

#|

Copyright (C) 1990-1994 Computational Logic, Inc. All Rights Reserved.

FM9001 PUBLIC SOFTWARE LICENSE
Computational Logic, Inc.
1717 West Sixth, Suite 290
Austin, Texas 78703-4776

Please read this license carefully before using the FM9001 Software. By using the FM9001 Software, you are agreeing to be bound by the terms of this license. If you do not agree to the terms of this license, promptly return the FM9001 Software to the place where you obtained it.

The FM9001 Software was developed by Computational Logic, Inc.(CLI). You own the disk or other medium on which the FM9001 Software is recorded, but CLI retains title to the FM9001 Software. The purposes of this license are to identify the FM9001 Software and to make the FM9001 Software, including its source code, freely available. This license allows you to use, copy, distribute and modify the FM9001 Software, on the condition that you comply with all the Copying Policies set out below.

COPYING POLICIES

1. You may copy and distribute verbatim copies of the FM9001 Software as you receive it, in any medium, including embedding it verbatim in derivative works, provided that you a) conspicuously and appropriately publish on each copy a valid copyright notice "Copyright (C) 1990-1994 by Computational Logic, Inc. All Rights Reserved.", b) keep intact on all files the notices that refer to this License Agreement and to the absence of any warranty, and c) give all recipients of the FM9001 Software a copy of this License Agreement along with the program.

2. You may modify your copy or copies of the FM9001 Software or any portion of it, and copy and distribute such modifications provided you tell recipients that what they have is a modification by your organization of the CLI version of the FM9001 Software.

3. You may incorporate parts of the FM9001 Software into other programs provided that you acknowledge Computational Logic Inc. in the program documentation.

CLI also requests, but does not require, that any improvements or extensions to the FM9001 Software be returned to one of the addresses below, so that they

may be shared with other FM9001 users. The FM9001 Software, including its source, can be obtained by contacting one of these addresses.

Software-Request or Software-Request@CLI.COM
Computational Logic Inc.
1717 West Sixth, Suite 290
Austin, TX 78703-4776

NO WARRANTY

BECAUSE THE FM9001 SOFTWARE IS LICENSED FREE OF CHARGE, WE PROVIDE ABSOLUTELY NO WARRANTY. THE SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE FM9001 SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT WILL COMPUTATIONAL LOGIC INC. BE LIABLE TO YOU FOR ANY DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE FM9001 SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

| #

EVENT: Start with the initial **nqthm** theory.

EVENT: Give the Nqthm control variable `reduce-term-clock` the value 2000.

DEFINITION:

```

delete( $x, l$ )
=  if listp( $l$ )
    then if  $x = \text{car}(l)$  then cdr( $l$ )
        else cons(car( $l$ ), delete( $x$ , cdr( $l$ ))) endif
    else  $l$  endif

```

DEFINITION:

```

bagdiff(x, y)
=  if listp(y)
    then if car(y) ∈ x then bagdiff(delete(car(y), x), cdr(y))

```

```

        else bagdiff(x, cdr(y)) endif
    else x endif

```

DEFINITION:

```

bagint(x, y)
= if listp(x)
    then if car(x) ∈ y
        then cons(car(x), bagint(cdr(x), delete(car(x), y)))
        else bagint(cdr(x), y) endif
    else nil endif

```

DEFINITION:

```

occurrences(x, l)
= if listp(l)
    then if x = car(l) then 1 + occurrences(x, cdr(l))
        else occurrences(x, cdr(l)) endif
    else 0 endif

```

DEFINITION:

```

subbagp(x, y)
= if listp(x)
    then if car(x) ∈ y then subbagp(cdr(x), delete(car(x), y))
        else f endif
    else t endif

```

THEOREM: listp-delete

```

listp(delete(x, l))
= if listp(l) then (x ≠ car(l)) ∨ listp(cdr(l))
  else f endif

```

EVENT: Enable listp-delete; name this event 'listp-delete-off'.

THEOREM: delete-non-member

$$(x \notin y) \rightarrow (\text{delete}(x, y) = y)$$

THEOREM: delete-delete

$$\text{delete}(y, \text{delete}(x, z)) = \text{delete}(x, \text{delete}(y, z))$$

THEOREM: equal-occurrences-zero

$$(\text{occurrences}(x, l) = 0) = (x \notin l)$$

THEOREM: member-non-list

$$(\neg \text{listp}(l)) \rightarrow (x \notin l)$$

THEOREM: member-delete

$(x \in \text{delete}(y, l))$
= **if** $x \in l$
 then if $x = y$ **then** $1 < \text{occurrences}(x, l)$
 else t endif
 else f endif

THEOREM: member-delete-implies-membership

$(x \in \text{delete}(y, l)) \rightarrow (x \in l)$

THEOREM: occurrences-delete

$\text{occurrences}(x, \text{delete}(y, l))$
= **if** $x = y$
 then if $x \in l$ **then** $\text{occurrences}(x, l) - 1$
 else 0 endif
 else occurrences}(x, l) endif

THEOREM: member-bagdiff

$(x \in \text{bagdiff}(a, b)) = (\text{occurrences}(x, b) < \text{occurrences}(x, a))$

THEOREM: bagdiff-delete

$\text{bagdiff}(\text{delete}(e, x), y) = \text{delete}(e, \text{bagdiff}(x, y))$

THEOREM: subbagp-delete

$\text{subbagp}(x, \text{delete}(u, y)) \rightarrow \text{subbagp}(x, y)$

THEOREM: subbagp-cdr1

$\text{subbagp}(x, y) \rightarrow \text{subbagp}(\text{cdr}(x), y)$

THEOREM: subbagp-cdr2

$\text{subbagp}(x, \text{cdr}(y)) \rightarrow \text{subbagp}(x, y)$

THEOREM: subbagp-bagint1

$\text{subbagp}(\text{bagint}(x, y), x)$

THEOREM: subbagp-bagint2

$\text{subbagp}(\text{bagint}(x, y), y)$

THEOREM: occurrences-bagint

$\text{occurrences}(x, \text{bagint}(a, b))$
= **if** $\text{occurrences}(x, a) < \text{occurrences}(x, b)$ **then** $\text{occurrences}(x, a)$
 else occurrences}(x, b) endif

THEOREM: occurrences-bagdiff

$\text{occurrences}(x, \text{bagdiff}(a, b)) = (\text{occurrences}(x, a) - \text{occurrences}(x, b))$

THEOREM: member-bagint

$$(x \in \text{bagint}(a, b)) = ((x \in a) \wedge (x \in b))$$

EVENT: Let us define the theory *bags* to consist of the following events: occurrences-bagint, bagdiff-delete, occurrences-bagdiff, member-bagint, member-bagdiff, subbagp-bagint2, subbagp-bagint1, subbagp-cdr2, subbagp-cdr1, subbagp-delete.

THEOREM: equal-plus-0

$$((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$$

THEOREM: plus-cancellation

$$((a + b) = (a + c)) = (\text{fix}(b) = \text{fix}(c))$$

EVENT: Enable plus-cancellation; name this event ‘plus-cancellation-off’.

THEOREM: equal-difference-0

$$(((x - y) = 0) = (y \not< x)) \wedge ((0 = (x - y)) = (y \not< x))$$

THEOREM: difference-cancellation

$$\begin{aligned} & ((x - y) = (z - y)) \\ = & \text{ if } x < y \text{ then } y \not< z \\ & \text{ elseif } z < y \text{ then } y \not< x \\ & \text{ else } \text{fix}(x) = \text{fix}(z) \text{ endif} \end{aligned}$$

EVENT: Enable difference-cancellation; name this event ‘difference-cancellation-off’.

THEOREM: commutativity-of-plus

$$(x + y) = (y + x)$$

THEOREM: commutativity2-of-plus

$$(x + y + z) = (y + x + z)$$

THEOREM: plus-zero-arg2

$$(y \simeq 0) \rightarrow ((x + y) = \text{fix}(x))$$

THEOREM: plus-add1-arg1

$$((1 + a) + b) = (1 + (a + b))$$

THEOREM: plus-add1-arg2

$$\begin{aligned} & (x + (1 + y)) \\ = & \text{ if } y \in \mathbf{N} \text{ then } 1 + (x + y) \\ & \text{ else } 1 + x \text{ endif} \end{aligned}$$

THEOREM: associativity-of-plus

$$((x + y) + z) = (x + y + z)$$

THEOREM: plus-difference-arg1

$$\begin{aligned} & ((a - b) + c) \\ = & \text{ if } b < a \text{ then } (a + c) - b \\ & \text{ else } 0 + c \text{ endif} \end{aligned}$$

THEOREM: plus-difference-arg2

$$\begin{aligned} & (a + (b - c)) \\ = & \text{ if } c < b \text{ then } (a + b) - c \\ & \text{ else } a + 0 \text{ endif} \end{aligned}$$

THEOREM: difference-plus-cancellation-proof

$$((x + y) - x) = \text{fix}(y)$$

THEOREM: difference-plus-cancellation

$$(((x + y) - x) = \text{fix}(y)) \wedge (((y + x) - x) = \text{fix}(y))$$

EVENT: Enable difference-plus-cancellation; name this event ‘difference-plus-cancellation-off’.

THEOREM: difference-plus-plus-cancellation-proof

$$((x + y) - (x + z)) = (y - z)$$

THEOREM: difference-plus-plus-cancellation

$$\begin{aligned} & (((x + y) - (x + z)) = (y - z)) \\ \wedge & (((y + x) - (x + z)) = (y - z)) \\ \wedge & (((x + y) - (z + x)) = (y - z)) \\ \wedge & (((y + x) - (z + x)) = (y - z)) \end{aligned}$$

EVENT: Enable difference-plus-plus-cancellation; name this event ‘difference-plus-plus-cancellation-off’.

THEOREM: difference-plus-plus-cancellation-hack

$$((w + x + a) - (y + z + a)) = ((w + x) - (y + z))$$

EVENT: Enable difference-plus-plus-cancellation-hack; name this event ‘difference-plus-plus-cancellation-hack-off’.

THEOREM: diff-sub1-arg2

$$\begin{aligned} & (a - (b - 1)) \\ = & \text{ if } b \simeq 0 \text{ then } \text{fix}(a) \\ & \text{ elseif } a < b \text{ then } 0 \\ & \text{ else } 1 + (a - b) \text{ endif} \end{aligned}$$

EVENT: Enable diff-sub1-arg2; name this event ‘diff-sub1-arg2-off’.

THEOREM: diff-diff-arg1
 $((x - y) - z) = (x - (y + z))$

THEOREM: diff-diff-arg2
 $(a - (b - c))$
 $= \text{if } b < c \text{ then fix}(a)$
 $\quad \text{else } (a + c) - b \text{ endif}$

THEOREM: diff-diff-diff
 $(\text{if } a < b \text{ then f}$
 $\quad \text{else t endif}$
 $\wedge \text{ if } c < d \text{ then f}$
 $\quad \text{else t endif})$
 $\rightarrow (((a - b) - (c - d)) = ((a + d) - (b + c)))$

EVENT: Enable diff-diff-diff; name this event ‘diff-diff-diff-off’.

THEOREM: difference-lessp-arg1
 $(a < b) \rightarrow ((a - b) = 0)$

EVENT: Enable difference-lessp-arg1; name this event ‘difference-lessp-arg1-off’.

DEFINITION:
plus-fringe(x)
 $= \text{if listp}(x) \wedge (\text{car}(x) = \text{'plus})$
 $\quad \text{then append(plus-fringe(cadr}(x)), \text{plus-fringe(caddr}(x)))$
 $\quad \text{else list}(x) \text{ endif}$

DEFINITION:
plus-tree(l)
 $= \text{if } l \simeq \text{nil} \text{ then '0}$
 $\quad \text{elseif cdr}(l) \simeq \text{nil} \text{ then list('fix, car}(l))$
 $\quad \text{elseif caddr}(l) \simeq \text{nil} \text{ then list('plus, car}(l), \text{cadr}(l))$
 $\quad \text{else list('plus, car}(l), \text{plus-tree(cdr}(l)) \text{ endif}$

THEOREM: numberp-eval\$-plus
 $(\text{listp}(x) \wedge (\text{car}(x) = \text{'plus})) \rightarrow (\text{eval}(\mathbf{t}, x, a) \in \mathbf{N})$

EVENT: Enable numberp-eval\$-plus; name this event ‘numberp-eval\$-plus-off’.

THEOREM: numberp-eval\$-plus-tree
 $\text{eval}(\mathbf{t}, \text{plus-tree}(l), a) \in \mathbf{N}$

EVENT: Enable numberp-eval\$-plus-tree; name this event ‘numberp-eval\$-plus-tree-off’.

THEOREM: member-implies-plus-tree-greatereqp
 $(x \in y) \rightarrow (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \not\leq \text{eval\$}(\mathbf{t}, x, a))$

EVENT: Enable member-implies-plus-tree-greatereqp; name this event ‘member-implies-plus-tree-greatereqp-off’.

THEOREM: plus-tree-delete
 $\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{delete}(x, y)), a)$
 $= \text{ if } x \in y \text{ then } \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval\$}(\mathbf{t}, x, a)$
 $\text{ else } \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \text{ endif}$

EVENT: Enable plus-tree-delete; name this event ‘plus-tree-delete-off’.

THEOREM: subbagp-implies-plus-tree-greatereqp
 $\text{subbagp}(x, y) \rightarrow (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) \not\leq \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a))$

EVENT: Enable subbagp-implies-plus-tree-greatereqp; name this event ‘subbagp-implies-plus-tree-greatereqp-off’.

THEOREM: plus-tree-bagdiff
 $\text{subbagp}(x, y)$
 $\rightarrow (\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{bagdiff}(y, x)), a)$
 $= (\text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a)))$

EVENT: Enable plus-tree-bagdiff; name this event ‘plus-tree-bagdiff-off’.

THEOREM: numberp-eval\$-bridge
 $(\text{eval\$}(\mathbf{t}, z, a) = \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a)) \rightarrow (\text{eval\$}(\mathbf{t}, z, a) \in \mathbf{N})$

EVENT: Enable numberp-eval\$-bridge; name this event ‘numberp-eval\$-bridge-off’.

THEOREM: bridge-to-subbagp-implies-plus-tree-greatereqp
 $(\text{subbagp}(y, \text{plus-fringe}(z))$
 $\wedge (\text{eval\$}(\mathbf{t}, z, a) = \text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(z)), a)))$
 $\rightarrow ((\text{eval\$}(\mathbf{t}, z, a) < \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a)) = \mathbf{f})$

EVENT: Enable bridge-to-subbagp-implies-plus-tree-greatereqp; name this event ‘bridge-to-subbagp-implies-plus-tree-greatereqp-off’.

THEOREM: eval\$-plus-tree-append

$$\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{append}(x, y), a)) \\ = \text{eval\$}(\mathbf{t}, \text{plus-tree}(x), a) + \text{eval\$}(\mathbf{t}, \text{plus-tree}(y), a))$$

EVENT: Enable eval\$-plus-tree-append; name this event ‘eval\$-plus-tree-append-off’.

THEOREM: plus-tree-plus-fringe

$$\text{eval\$}(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(x)), a) = \text{fix}(\text{eval\$}(\mathbf{t}, x, a))$$

EVENT: Enable plus-tree-plus-fringe; name this event ‘plus-tree-plus-fringe-off’.

THEOREM: member-implies-numberp

$$((c \in \text{plus-fringe}(x)) \wedge (\text{eval\$}(\mathbf{t}, c, a) \in \mathbf{N})) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) \in \mathbf{N})$$

EVENT: Enable member-implies-numberp; name this event ‘member-implies-numberp-off’.

THEOREM: cadr-eval\$list

$$(\text{car}(\text{eval\$}(\mathbf{'list}, x, a)) = \text{eval\$}(\mathbf{t}, \text{car}(x), a)) \\ \wedge (\text{cdr}(\text{eval\$}(\mathbf{'list}, x, a)) \\ = \text{if listp}(x) \text{ then eval\$}(\mathbf{'list}, \text{cdr}(x), a) \\ \text{else } 0 \text{ endif})$$

EVENT: Enable cadr-eval\$list; name this event ‘cadr-eval\$list-off’.

THEOREM: eval\$-quote

$$\text{eval\$}(\mathbf{t}, \text{cons}(\mathbf{'quote}, args), a) = \text{car}(args)$$

EVENT: Enable eval\$-quote; name this event ‘eval\$-quote-off’.

THEOREM: listp-eval\$

$$\text{listp}(\text{eval\$}(\mathbf{'list}, x, a)) = \text{listp}(x)$$

EVENT: Enable listp-eval\$; name this event ‘listp-eval\$-off’.

DEFINITION:

$$\text{cancel-equal-plus}(x) \\ = \text{if listp}(x) \wedge (\text{car}(x) = \mathbf{'equal'}) \\ \text{then if listp}(\text{cadr}(x)) \\ \quad \wedge (\text{caadr}(x) = \mathbf{'plus'}) \\ \quad \wedge \text{listp}(\text{caddr}(x))$$

```

      ∧ (caaddr(x) = 'plus)
then list('equal,
      plus-tree(bagdiff(plus-fringe(cadr(x)),
                        bagint(plus-fringe(cadr(x)),
                              plus-fringe(caddr(x))))),
      plus-tree(bagdiff(plus-fringe(caddr(x)),
                        bagint(plus-fringe(cadr(x)),
                              plus-fringe(caddr(x))))))

elseif listp(cadr(x))
      ∧ (caaddr(x) = 'plus)
      ∧ (caddr(x) ∈ plus-fringe(cadr(x)))
then list('if,
      list('numberp, caddr(x)),
      cons('equal,
      cons(plus-tree(delete(caddr(x),
                           plus-fringe(cadr(x)))),
        '('0))),
      list('quote, f))
elseif listp(caddr(x))
      ∧ (caaddr(x) = 'plus)
      ∧ (cadr(x) ∈ plus-fringe(caddr(x)))
then list('if,
      list('numberp, cadr(x)),
      list('equal,
        '0,
        plus-tree(delete(cadr(x), plus-fringe(caddr(x))))),
      list('quote, f))
else x endif
else x endif

```

THEOREM: correctness-of-cancel-equal-plus
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-equal-plus}(x), a)$

DEFINITION:

```

cancel-difference-plus(x)
= if listp(x) ∧ (car(x) = 'difference)
  then if listp(cadr(x))
    ∧ (caaddr(x) = 'plus)
    ∧ listp(caddr(x))
    ∧ (caaddr(x) = 'plus)
  then list('difference,
    plus-tree(bagdiff(plus-fringe(cadr(x)),
                      bagint(plus-fringe(cadr(x)),
                            plus-fringe(caddr(x))))),

```

```

        plus-tree (bagdiff (plus-fringe (caddr (x)),
                           bagint (plus-fringe (cadr (x)),
                                   plus-fringe (caddr (x)))))
    elseif listp (cadr (x))
        ∧ (caaddr (x) = 'plus)
        ∧ (caddr (x) ∈ plus-fringe (cadr (x)))
    then plus-tree (delete (caddr (x), plus-fringe (cadr (x))))
    elseif listp (caddr (x))
        ∧ (caaddr (x) = 'plus)
        ∧ (cadr (x) ∈ plus-fringe (caddr (x))) then '0
    else x endif
else x endif

```

THEOREM: correctness-of-cancel-difference-plus
 $\text{eval\$}(\mathbf{t}, x, a) = \text{eval\$}(\mathbf{t}, \text{cancel-difference-plus}(x), a)$

THEOREM: difference-elim
 $((y \in \mathbf{N}) \wedge (y \not\prec x)) \rightarrow ((x + (y - x)) = y)$

THEOREM: difference-leq-arg1
if $b < a$ **then** $\mathbf{f}$
else $\mathbf{t}$ **endif**
 $\rightarrow ((a - b) = 0)$

THEOREM: difference-add1-arg2
 $(a - (1 + b))$
 $=$ **if** $b < a$ **then** $(a - b) - 1$
else 0 **endif**

THEOREM: difference-sub1-arg2
 $(a - (b - 1))$
 $=$ **if** $b \simeq 0$ **then** $\text{fix}(a)$
elseif $a < b$ **then** 0
else $1 + (a - b)$ **endif**

THEOREM: difference-difference-arg1
 $((x - y) - z) = (x - (y + z))$

THEOREM: difference-difference-arg2
 $(a - (b - c))$
 $=$ **if** $b < c$ **then** $\text{fix}(a)$
else $(a + c) - b$ **endif**

THEOREM: difference-x-x
 $(x - x) = 0$

THEOREM: lessp-difference-cancellation

$$((a - c) < (b - c))$$

```

=  if if  $a < c$  then  $f$ 
    else  $t$  endif then  $a < b$ 
    else  $c < b$  endif

```

EVENT: Enable lessp-difference-cancellation; name this event 'lessp-difference-cancellation-off'.

DEFINITION:

```

cancel-lessp-plus( $x$ )
=  if listp( $x$ )  $\wedge$  (car( $x$ ) = 'lessp)
    then if listp(cadr( $x$ ))
           $\wedge$  (caadr( $x$ ) = 'plus)
           $\wedge$  listp(caddr( $x$ ))
           $\wedge$  (caaddr( $x$ ) = 'plus)
        then list('lessp,
                  plus-tree(bagdiff(plus-fringe(cadr( $x$ )),
                                     bagint(plus-fringe(cadr( $x$ )),
                                             plus-fringe(caddr( $x$ ))))),
                  plus-tree(bagdiff(plus-fringe(caddr( $x$ )),
                                     bagint(plus-fringe(cadr( $x$ )),
                                             plus-fringe(caddr( $x$ ))))))
        elseif listp(cadr( $x$ ))
           $\wedge$  (caadr( $x$ ) = 'plus)
           $\wedge$  (caddr( $x$ )  $\in$  plus-fringe(cadr( $x$ )))
        then list('quote,  $f$ )
        elseif listp(caddr( $x$ ))
           $\wedge$  (caaddr( $x$ ) = 'plus)
           $\wedge$  (cadr( $x$ )  $\in$  plus-fringe(caddr( $x$ )))
        then list('not,
                  list('zerop,
                      plus-tree(delete(cadr( $x$ ), plus-fringe(caddr( $x$ ))))))
        else  $x$  endif
    else  $x$  endif

```

THEOREM: correctness-of-cancel-lessp-plus

$$\text{eval}(\mathbf{t}, x, a) = \text{eval}(\mathbf{t}, \text{cancel-lessp-plus}(x), a)$$

EVENT: Let us define the theory *addition* to consist of the following events: equal-plus-0, equal-difference-0, commutativity-of-plus, commutativity2-of-plus, plus-zero-arg2, plus-add1-arg2, plus-add1-arg1, associativity-of-plus, plus-difference-arg1, plus-difference-arg2, diff-diff-arg1, diff-diff-arg2, correctness-of-cancel-equal-plus, correctness-of-cancel-difference-plus, difference-elim, difference-leq-arg1, difference-

add1-arg2, difference-sub1-arg2, difference-difference-arg1, difference-difference-arg2, difference-x-x, correctness-of-cancel-lessp-plus.

THEOREM: equal-times-0
 $((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$

THEOREM: equal-times-1
 $((a * b) = 1) = ((a = 1) \wedge (b = 1))$

THEOREM: equal-sub1-0
 $((x - 1) = 0) = ((x \simeq 0) \vee (x = 1))$

THEOREM: times-zero
 $(y \simeq 0) \rightarrow ((x * y) = 0)$

THEOREM: times-add1
 $(x * (1 + y))$
 $= \text{if } y \in \mathbf{N} \text{ then } x + (x * y)$
 $\text{else fix}(x) \text{ endif}$

THEOREM: commutativity-of-times
 $(y * x) = (x * y)$

THEOREM: times-distributes-over-plus-proof
 $(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-distributes-over-plus
 $((x * (y + z)) = ((x * y) + (x * z)))$
 $\wedge (((x + y) * z) = ((x * z) + (y * z)))$

THEOREM: commutativity2-of-times
 $(x * y * z) = (y * x * z)$

THEOREM: associativity-of-times
 $((x * y) * z) = (x * y * z)$

THEOREM: times-distributes-over-difference-proof
 $((a - b) * c) = ((a * c) - (b * c))$

THEOREM: times-distributes-over-difference
 $((((a - b) * c) = ((a * c) - (b * c))))$
 $\wedge ((a * (b - c)) = ((a * b) - (a * c)))$

THEOREM: times-quotient-proof
 $((x \neq 0) \wedge ((y \bmod x) = 0)) \rightarrow (((y \div x) * x) = \text{fix}(y))$

THEOREM: times-quotient

$$\begin{aligned} & ((y \neq 0) \wedge ((x \bmod y) = 0)) \\ \rightarrow & (((x \div y) * y) = \text{fix}(x)) \wedge ((y * (x \div y)) = \text{fix}(x)) \end{aligned}$$

THEOREM: times-1-arg1

$$(1 * x) = \text{fix}(x)$$

THEOREM: lessp-times1-proof

$$((a < b) \wedge (c \neq 0)) \rightarrow ((a < (b * c)) = \mathbf{t})$$

THEOREM: lessp-times1

$$\begin{aligned} & ((a < b) \wedge (c \neq 0)) \\ \rightarrow & (((a < (b * c)) = \mathbf{t}) \wedge ((a < (c * b)) = \mathbf{t})) \end{aligned}$$

THEOREM: lessp-times2-proof

$$\begin{aligned} & (\text{if } b < a \text{ then } \mathbf{f} \\ & \quad \text{else } \mathbf{t} \text{ endif} \\ & \wedge (c \neq 0)) \\ \rightarrow & (((b * c) < a) = \mathbf{f}) \end{aligned}$$

THEOREM: lessp-times2

$$\begin{aligned} & (\text{if } b < a \text{ then } \mathbf{f} \\ & \quad \text{else } \mathbf{t} \text{ endif} \\ & \wedge (c \neq 0)) \\ \rightarrow & (((b * c) < a) = \mathbf{f}) \wedge (((c * b) < a) = \mathbf{f}) \end{aligned}$$

THEOREM: lessp-times3-proof1

$$((a \neq 0) \wedge (1 < b)) \rightarrow (a < (a * b))$$

THEOREM: lessp-times3-proof2

$$(a < (a * b)) \rightarrow ((a \neq 0) \wedge (1 < b))$$

THEOREM: lessp-times3

$$\begin{aligned} & ((a < (a * b)) = ((a \neq 0) \wedge (1 < b))) \\ \wedge & ((a < (b * a)) = ((a \neq 0) \wedge (1 < b))) \end{aligned}$$

THEOREM: lessp-times-cancellation-proof

$$((x * z) < (y * z)) = ((z \neq 0) \wedge (x < y))$$

THEOREM: lessp-times-cancellation1

$$\begin{aligned} & (((x * z) < (y * z)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((z * x) < (y * z)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((x * z) < (z * y)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((z * x) < (z * y)) = ((z \neq 0) \wedge (x < y))) \end{aligned}$$

EVENT: Enable lessp-times-cancellation1; name this event ‘lessp-times-cancellation1-off’.

THEOREM: lessp-plus-times-proof

$$(x < a) \rightarrow (((x + (a * b)) < (a * c)) = (b < c))$$

THEOREM: lessp-plus-times1

$$\begin{aligned} & (((a + (b * c)) < b) = ((a < b) \wedge (c \simeq 0))) \\ \wedge & \quad (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq 0))) \\ \wedge & \quad (((c * b) + a) < b) = ((a < b) \wedge (c \simeq 0)) \\ \wedge & \quad (((b * c) + a) < b) = ((a < b) \wedge (c \simeq 0)) \end{aligned}$$

THEOREM: lessp-plus-times2

$$\begin{aligned} & ((a \not\simeq 0) \wedge (x < a)) \\ \rightarrow & \quad (((x + (a * b)) < (a * c)) = (b < c)) \\ & \quad \wedge \quad (((x + (b * a)) < (a * c)) = (b < c)) \\ & \quad \wedge \quad (((x + (a * b)) < (c * a)) = (b < c)) \\ & \quad \wedge \quad (((x + (b * a)) < (c * a)) = (b < c)) \\ & \quad \wedge \quad (((a * b) + x) < (a * c)) = (b < c)) \\ & \quad \wedge \quad (((b * a) + x) < (a * c)) = (b < c)) \\ & \quad \wedge \quad (((a * b) + x) < (c * a)) = (b < c)) \\ & \quad \wedge \quad (((b * a) + x) < (c * a)) = (b < c)) \end{aligned}$$

THEOREM: lessp-1-times

$$\begin{aligned} & (1 < (a * b)) \\ = & \quad (\neg ((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1)))) \end{aligned}$$

DEFINITION:

times-tree(x)
= **if** $x \simeq \mathbf{nil}$ **then** '1
 elseif $\text{cdr}(x) \simeq \mathbf{nil}$ **then** list('fix, car(x))
 elseif $\text{caddr}(x) \simeq \mathbf{nil}$ **then** list('times, car(x), caddr(x))
 else list('times, car(x), times-tree(cdr(x))) **endif**

DEFINITION:

times-fringe(x)
= **if** listp(x) $\wedge$ (car(x) = 'times)
 then append(times-fringe(cadr(x)), times-fringe(caddr(x)))
 else list(x) **endif**

DEFINITION:

or-zero-tree(x)
= **if** $x \simeq \mathbf{nil}$ **then** '(false)
 elseif $\text{cdr}(x) \simeq \mathbf{nil}$ **then** list('zerop, car(x))
 elseif $\text{caddr}(x) \simeq \mathbf{nil}$
 then list('or, list('zerop, car(x)), list('zerop, caddr(x)))
 else list('or, list('zerop, car(x)), or-zero-tree(cdr(x))) **endif**

DEFINITION:

and-not-zero-tree (x)
 $=$ **if** $x \simeq \text{nil}$ **then** 'true)
 elseif $\text{cdr}(x) \simeq \text{nil}$ **then** list('not, list('zerop, car(x)))
 else list('and,
 list('not, list('zerop, car(x))),
 and-not-zero-tree(cdr(x))) **endif**

THEOREM: numberp-eval\$-times
 $(\text{car}(x) = \text{'times}) \rightarrow (\text{eval}\$(\mathbf{t}, x, a) \in \mathbf{N})$

EVENT: Enable numberp-eval\$-times; name this event 'numberp-eval\$-times-off'.

THEOREM: eval\$-times
 $(\text{car}(x) = \text{'times})$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = (\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) * \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Enable eval\$-times; name this event 'eval\$-times-off'.

THEOREM: eval\$-or
 $(\text{car}(x) = \text{'or})$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = (\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) \vee \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Enable eval\$-or; name this event 'eval\$-or-off'.

THEOREM: eval\$-equal
 $(\text{car}(x) = \text{'equal})$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = (\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) = \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Enable eval\$-equal; name this event 'eval\$-equal-off'.

THEOREM: eval\$-lessp
 $(\text{car}(x) = \text{'lessp})$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = (\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) < \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Enable eval\$-lessp; name this event 'eval\$-lessp-off'.

THEOREM: eval\$-quotient
 $(\text{car}(x) = \text{'quotient})$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = (\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) \div \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

EVENT: Enable eval\$-quotient; name this event 'eval\$-quotient-off'.

THEOREM: eval\$-if
 $(\text{car}(x) = \text{'if})$
 $\rightarrow (\text{eval\$}(\mathbf{t}, x, a)$
 $\quad = \text{if eval\$}(\mathbf{t}, \text{cadr}(x), a) \text{ then eval\$}(\mathbf{t}, \text{caddr}(x), a)$
 $\quad \text{else eval\$}(\mathbf{t}, \text{caddr}(x), a) \text{ endif})$

EVENT: Enable eval\$-if; name this event 'eval\$-if-off'.

THEOREM: numberp-eval\$-times-tree
 $\text{eval\$}(\mathbf{t}, \text{times-tree}(x), a) \in \mathbf{N}$

EVENT: Enable numberp-eval\$-times-tree; name this event 'numberp-eval\$-times-tree-off'.

THEOREM: lessp-times-arg1
 $(a \not\leq 0) \rightarrow (((a * x) \not\leq (a * y)) = (x \not\leq y))$

THEOREM: infer-equality-from-not-lessp
 $((a \in \mathbf{N}) \wedge (b \in \mathbf{N})) \rightarrow (((a \not\leq b) \wedge (b \not\leq a)) = (a = b))$

THEOREM: equal-times-arg1
 $(a \not\leq 0) \rightarrow (((a * x) = (a * y)) = (\text{fix}(x) = \text{fix}(y)))$

EVENT: Enable equal-times-arg1; name this event 'equal-times-arg1-off'.

THEOREM: equal-times-bridge
 $((a * b) = (c * a * d)) = ((a \simeq 0) \vee (\text{fix}(b) = (c * d)))$

EVENT: Enable equal-times-bridge; name this event 'equal-times-bridge-off'.

THEOREM: eval\$-times-member
 $(e \in x)$
 $\rightarrow (\text{eval\$}(\mathbf{t}, \text{times-tree}(x), a)$
 $\quad = (\text{eval\$}(\mathbf{t}, e, a) * \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{delete}(e, x)), a)))$

EVENT: Enable eval\$-times-member; name this event 'eval\$-times-member-off'.

THEOREM: zerop-makes-times-tree-zero
 $((\neg \text{eval\$}(\mathbf{t}, \text{and-not-zerop-tree}(x), a)) \wedge \text{subbagp}(x, y))$
 $\rightarrow (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) = 0)$

EVENT: Enable zerop-makes-times-tree-zero; name this event 'zerop-makes-times-tree-zero-off'.

THEOREM: or-zero-tree-is-not-zero-tree

$$\text{eval\$}(\mathbf{t}, \text{or-zero-tree}(x), a) = (\neg \text{eval\$}(\mathbf{t}, \text{and-not-zero-tree}(x), a))$$

EVENT: Enable or-zero-tree-is-not-zero-tree; name this event ‘or-zero-tree-is-not-zero-tree-off’.

THEOREM: zero-tree-makes-times-tree-zero2

$$\begin{aligned} &(\text{eval\$}(\mathbf{t}, \text{or-zero-tree}(x), a) \wedge \text{subbagp}(x, y)) \\ \rightarrow &(\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) = 0) \end{aligned}$$

EVENT: Enable zero-tree-makes-times-tree-zero2; name this event ‘zero-tree-makes-times-tree-zero2-off’.

THEOREM: times-tree-append

$$\begin{aligned} &\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{append}(x, y)), a) \\ = &(\text{eval\$}(\mathbf{t}, \text{times-tree}(x), a) * \text{eval\$}(\mathbf{t}, \text{times-tree}(y), a)) \end{aligned}$$

EVENT: Enable times-tree-append; name this event ‘times-tree-append-off’.

THEOREM: times-tree-of-times-fringe

$$\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{times-fringe}(x)), a) = \text{fix}(\text{eval\$}(\mathbf{t}, x, a))$$

EVENT: Enable times-tree-of-times-fringe; name this event ‘times-tree-of-times-fringe-off’.

DEFINITION:

cancel-lessp-times(x)

```
=  if (car(x) = 'lessp)
    ∧ (caadr(x) = 'times)
    ∧ (caaddr(x) = 'times)
  then if listp(bagint(times-fringe(cadr(x)), times-fringe(caddr(x))))
      then list('and,
                and-not-zero-tree(bagint(times-fringe(cadr(x)),
                                           times-fringe(caddr(x)))),
                list('lessp,
                    times-tree(bagdiff(times-fringe(cadr(x)),
                                         bagint(times-fringe(cadr(x)),
                                                times-fringe(caddr(x))))),
                    times-tree(bagdiff(times-fringe(caddr(x)),
                                         bagint(times-fringe(cadr(x)),
                                                times-fringe(caddr(x)))))))
      else x endif
  else x endif
```

THEOREM: eval\$-lessp-times-tree-bagdiff

$$\begin{aligned} & (\text{subbagp}(x, y) \wedge \text{subbagp}(x, z) \wedge \text{eval\$}(\mathbf{t}, \text{and-not-zerop-tree}(x), a)) \\ \rightarrow & ((\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a) \\ & < \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a)) \\ = & (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) < \text{eval\$}(\mathbf{t}, \text{times-tree}(z), a))) \end{aligned}$$

EVENT: Enable eval\$-lessp-times-tree-bagdiff; name this event ‘eval\$-lessp-times-tree-bagdiff-off’.

THEOREM: zerop-makes-lessp-false-bridge

$$\begin{aligned} & ((\text{car}(x) = \text{'times}) \\ & \wedge (\text{car}(y) = \text{'times}) \\ & \wedge (\neg \text{eval\$}(\mathbf{t}, \\ & \quad \text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y))), \\ & \quad a))) \\ \rightarrow & (((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)) \\ & < (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a))) \\ = & \mathbf{f}) \end{aligned}$$

EVENT: Enable zerop-makes-lessp-false-bridge; name this event ‘zerop-makes-lessp-false-bridge-off’.

THEOREM: correctness-of-cancel-lessp-times

$$\text{eval\$}(\mathbf{t}, x, a) = \text{eval\$}(\mathbf{t}, \text{cancel-lessp-times}(x), a)$$

DEFINITION:
cancel-equal-times(x)
= **if** ($\text{car}(x) = \text{'equal}$)
 $\quad \wedge (\text{caadr}(x) = \text{'times})$
 $\quad \wedge (\text{caaddr}(x) = \text{'times})$
then if listp($\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \text{times-fringe}(\text{caddr}(x)))$)
 $\quad \text{then list}(\text{'or},$
 $\quad \quad \text{or-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \quad \quad \text{times-fringe}(\text{caddr}(x))),$
 $\quad \text{list}(\text{'equal},$
 $\quad \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \quad \quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \quad \quad \text{times-fringe}(\text{caddr}(x)))))$,
 $\quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{caddr}(x)),$
 $\quad \quad \text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \quad \quad \text{times-fringe}(\text{caddr}(x)))))$)
else x **endif**
else x **endif**

THEOREM: zerop-makes-equal-true-bridge

$$\begin{aligned}
& ((\text{car}(x) = \text{'times}) \\
& \quad \wedge (\text{car}(y) = \text{'times}) \\
& \quad \wedge \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y))), a)) \\
\rightarrow & (((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)) \\
& \quad = (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a))) \\
& \quad = \mathbf{t})
\end{aligned}$$

EVENT: Enable zerop-makes-equal-true-bridge; name this event ‘zerop-makes-equal-true-bridge-off’.

THEOREM: eval\$-equal-times-tree-bagdiff

$$\begin{aligned}
& (\text{subbagp}(x, y) \wedge \text{subbagp}(x, z) \wedge (\neg \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(x), a))) \\
\rightarrow & ((\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a) \\
& \quad = \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a)) \\
& \quad = (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) = \text{eval\$}(\mathbf{t}, \text{times-tree}(z), a)))
\end{aligned}$$

EVENT: Enable eval\$-equal-times-tree-bagdiff; name this event ‘eval\$-equal-times-tree-bagdiff-off’.

THEOREM: cancel-equal-times-preserves-inequality

$$\begin{aligned}
& (\text{subbagp}(z, x) \\
& \quad \wedge \text{subbagp}(z, y) \\
& \quad \wedge (\text{eval\$}(\mathbf{t}, \text{times-tree}(x), a) \neq \text{eval\$}(\mathbf{t}, \text{times-tree}(y), a))) \\
\rightarrow & (\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(x, z)), a) \\
& \quad \neq \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, z)), a))
\end{aligned}$$

EVENT: Enable cancel-equal-times-preserves-inequality; name this event ‘cancel-equal-times-preserves-inequality-off’.

THEOREM: cancel-equal-times-preserves-inequality-bridge

$$\begin{aligned}
& ((\text{car}(x) = \text{'times}) \\
& \quad \wedge (\text{car}(y) = \text{'times}) \\
& \quad \wedge ((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)) \\
& \quad \quad \neq (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a)))) \\
\rightarrow & (\text{eval\$}(\mathbf{t}, \\
& \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(x), \\
& \quad \quad \text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y)))), \\
& \quad a) \\
& \quad \neq \text{eval\$}(\mathbf{t}, \\
& \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(y), \\
& \quad \quad \text{bagint}(\text{times-fringe}(x), \\
& \quad \quad \quad \text{times-fringe}(y)))), \\
& \quad a))
\end{aligned}$$

EVENT: Enable cancel-equal-times-preserves-inequality-bridge; name this event ‘cancel-equal-times-preserves-inequality-bridge-off’.

THEOREM: correctness-of-cancel-equal-times
 $\text{eval}\$ (\mathbf{t}, x, a) = \text{eval}\$ (\mathbf{t}, \text{cancel-equal-times}(x), a)$

EVENT: Let us define the theory *multiplication* to consist of the following events: equal-times-0, equal-times-1, equal-sub1-0, times-zero, times-add1, commutativity-of-times, times-distributes-over-plus, commutativity2-of-times, associativity-of-times, times-distributes-over-difference, times-quotient, times-1-arg1, lessp-times1, lessp-times2, lessp-times3, lessp-plus-times1, lessp-plus-times2, lessp-1-times, correctness-of-cancel-lessp-times, correctness-of-cancel-equal-times.

THEOREM: lessp-remainder
 $((x \mathbf{mod} y) < y) = (y \not\leq 0)$

THEOREM: remainder-noop
 $(a < b) \rightarrow ((a \mathbf{mod} b) = \text{fix}(a))$

THEOREM: remainder-of-non-number
 $(a \notin \mathbf{N}) \rightarrow ((a \mathbf{mod} n) = (0 \mathbf{mod} n))$

THEOREM: remainder-zero
 $(x \simeq 0) \rightarrow ((y \mathbf{mod} x) = \text{fix}(y))$

THEOREM: plus-remainder-times-quotient
 $((x \mathbf{mod} y) + (y * (x \div y))) = \text{fix}(x)$

EVENT: Enable plus-remainder-times-quotient; name this event ‘plus-remainder-times-quotient-off’.

THEOREM: remainder-quotient-elim
 $((y \not\leq 0) \wedge (x \in \mathbf{N})) \rightarrow (((x \mathbf{mod} y) + (y * (x \div y))) = x)$

THEOREM: remainder-add1
 $((a \mathbf{mod} b) = 0) \rightarrow (((1 + a) \mathbf{mod} b) = (1 \mathbf{mod} b))$

THEOREM: remainder-plus-proof
 $((b \mathbf{mod} c) = 0) \rightarrow (((a + b) \mathbf{mod} c) = (a \mathbf{mod} c))$

THEOREM: remainder-plus
 $((a \mathbf{mod} c) = 0)$
 $\rightarrow (((a + b) \mathbf{mod} c) = (b \mathbf{mod} c))$
 $\wedge (((b + a) \mathbf{mod} c) = (b \mathbf{mod} c))$
 $\wedge (((x + y + a) \mathbf{mod} c) = ((x + y) \mathbf{mod} c)))$

THEOREM: equal-remainder-plus-0-proof

$$((a \bmod c) = 0) \rightarrow (((a + b) \bmod c) = 0) = ((b \bmod c) = 0))$$

THEOREM: equal-remainder-plus-0

$$\begin{aligned} & ((a \bmod c) = 0) \\ \rightarrow & (((((a + b) \bmod c) = 0) = ((b \bmod c) = 0)) \\ & \wedge (((b + a) \bmod c) = 0) = ((b \bmod c) = 0)) \\ & \wedge (((x + y + a) \bmod c) = 0) \\ & = (((x + y) \bmod c) = 0))) \end{aligned}$$

THEOREM: equal-remainder-plus-remainder-proof

$$(a < c) \rightarrow (((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0))$$

THEOREM: equal-remainder-plus-remainder

$$\begin{aligned} & (a < c) \\ \rightarrow & (((((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0)) \\ & \wedge (((b + a) \bmod c) = (b \bmod c)) = (a \simeq 0))) \end{aligned}$$

EVENT: Enable equal-remainder-plus-remainder; name this event ‘equal-remainder-plus-remainder-off’.

THEOREM: remainder-times1-proof

$$((b \bmod c) = 0) \rightarrow (((a * b) \bmod c) = 0)$$

THEOREM: remainder-times1

$$\begin{aligned} & ((b \bmod c) = 0) \\ \rightarrow & (((a * b) \bmod c) = 0) \wedge (((b * a) \bmod c) = 0)) \end{aligned}$$

THEOREM: remainder-times1-instance-proof

$$((x * y) \bmod y) = 0$$

THEOREM: remainder-times1-instance

$$(((x * y) \bmod y) = 0) \wedge (((x * y) \bmod x) = 0)$$

THEOREM: remainder-times-times-proof

$$((x * y) \bmod (x * z)) = (x * (y \bmod z))$$

THEOREM: remainder-times-times

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

EVENT: Enable remainder-times-times; name this event ‘remainder-times-times-off’.

THEOREM: remainder-times2-proof

$$((a \bmod z) = 0) \rightarrow ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))$$

THEOREM: remainder-times2

$$\begin{aligned} & ((a \bmod z) = 0) \\ \rightarrow & (((a \bmod (y * z)) = (z * ((a \div z) \bmod y))) \\ & \wedge ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))) \end{aligned}$$

THEOREM: remainder-times2-instance

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

THEOREM: remainder-difference1

$$\begin{aligned} & ((a \bmod c) = (b \bmod c)) \\ \rightarrow & (((a - b) \bmod c) = ((a \bmod c) - (b \bmod c))) \end{aligned}$$

DEFINITION:

double-remainder-induction (a, b, c)

```
=  if c  $\simeq$  0 then 0
    elseif a < c then 0
    elseif b < c then 0
    else double-remainder-induction (a - c, b - c, c) endif
```

THEOREM: remainder-difference2

$$\begin{aligned} & (((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \bmod c) \\ & = \text{if } b < a \text{ then } c - (b \bmod c) \\ & \text{else } 0 \text{ endif}) \end{aligned}$$

THEOREM: remainder-difference3

$$\begin{aligned} & (((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \bmod c) \\ & = \text{if } b < a \text{ then } a \bmod c \\ & \text{else } 0 \text{ endif}) \end{aligned}$$

EVENT: Enable remainder-difference3; name this event ‘remainder-difference3-off’.

THEOREM: equal-remainder-difference-0

$$\begin{aligned} & (((a - b) \bmod c) = 0) \\ = & \text{if if } a < b \text{ then f} \\ & \text{else t endif then } (a \bmod c) = (b \bmod c) \\ & \text{else t endif} \end{aligned}$$

EVENT: Enable equal-remainder-difference-0; name this event ‘equal-remainder-difference-0-off’.

THEOREM: lessp-plus-fact

$$\begin{aligned} & (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (b < c) \wedge (a < x)) \\ \rightarrow & (((a + b) < c) = \mathbf{t}) \end{aligned}$$

EVENT: Enable lessp-plus-fact; name this event ‘lessp-plus-fact-off’.

THEOREM: remainder-plus-fact

$$\begin{aligned} & (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)) \\ \rightarrow & (((a + b) \bmod c) = (a + (b \bmod c))) \end{aligned}$$

THEOREM: remainder-plus-times-times-proof

$$\begin{aligned} & (a < b) \\ \rightarrow & (((a + (b * c)) \bmod (b * d)) \\ & = (a + ((b * c) \bmod (b * d)))) \end{aligned}$$

THEOREM: remainder-plus-times-times

$$\begin{aligned} & (a < b) \\ \rightarrow & (((((a + (b * c)) \bmod (b * d)) \\ & = (a + ((b * c) \bmod (b * d)))) \\ \wedge & (((a + (c * b)) \bmod (d * b)) \\ & = (a + ((c * b) \bmod (d * b))))) \end{aligned}$$

THEOREM: remainder-plus-times-times-instance

$$\begin{aligned} & (a < b) \\ \rightarrow & (((((a + (b * c) + (b * d)) \bmod (b * e)) \\ & = (a + (b * ((c + d) \bmod e)))) \\ \wedge & (((a + (c * b) + (d * b)) \bmod (e * b)) \\ & = (a + (b * ((c + d) \bmod e))))) \end{aligned}$$

THEOREM: remainder-remainder

$$((b \bmod a) = 0) \rightarrow (((n \bmod b) \bmod a) = (n \bmod a))$$

THEOREM: remainder-1-arg1

$$\begin{aligned} & (1 \bmod x) \\ = & \text{if } x = 1 \text{ then } 0 \\ & \text{else } 1 \text{ endif} \end{aligned}$$

THEOREM: remainder-1-arg2

$$(y \bmod 1) = 0$$

THEOREM: remainder-x-x

$$(x \bmod x) = 0$$

THEOREM: transitivity-of-divides

$$(((a \bmod b) = 0) \wedge ((b \bmod c) = 0)) \rightarrow ((a \bmod c) = 0)$$

EVENT: Let us define the theory *remainders* to consist of the following events:
 lessp-remainder, remainder-noop, remainder-of-non-number, remainder-zero, remainder-quotient-elim, remainder-add1, remainder-plus, equal-remainder-plus-0, remainder-times1, remainder-times1-instance, remainder-times2, remainder-times2-instance, remainder-difference1, remainder-difference2, remainder-plus-times-times, remainder-plus-times-times-instance, remainder-remainder, remainder-1-arg1, remainder-1-arg2, remainder-x-x.

THEOREM: quotient-noop
 $(b = 1) \rightarrow ((a \div b) = \text{fix}(a))$

THEOREM: quotient-of-non-number
 $(a \notin \mathbf{N}) \rightarrow ((a \div n) = (0 \div n))$

THEOREM: quotient-zero
 $(x \simeq 0) \rightarrow ((y \div x) = 0)$

THEOREM: quotient-add1
 $((a \bmod b) = 0)$
 $\rightarrow (((1 + a) \div b)$
 $\quad = \text{ if } b = 1 \text{ then } 1 + (a \div b)$
 $\quad \text{ else } a \div b \text{ endif})$

THEOREM: equal-quotient-0
 $((a \div b) = 0) = ((b \simeq 0) \vee (a < b))$

THEOREM: quotient-sub1
 $((a \not\simeq 0) \wedge (b \not\simeq 0))$
 $\rightarrow (((a - 1) \div b)$
 $\quad = \text{ if } (a \bmod b) = 0 \text{ then } (a \div b) - 1$
 $\quad \text{ else } a \div b \text{ endif})$

THEOREM: quotient-plus-proof
 $((b \bmod c) = 0) \rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$

THEOREM: quotient-plus
 $((a \bmod c) = 0)$
 $\rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((b + a) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((x + y + a) \div c)$
 $\quad \quad = (((x + y) \div c) + (a \div c)))$

THEOREM: quotient-times-instance-temp-proof
 $((y * x) \div y)$
 $= \text{ if } y \simeq 0 \text{ then } 0$
 $\quad \text{ else } \text{fix}(x) \text{ endif}$

THEOREM: quotient-times-instance-temp

$$\begin{aligned}
& (((y * x) \div y) \\
& = \quad \text{if } y \simeq 0 \text{ then } 0 \\
& \quad \text{else fix}(x) \text{ endif} \\
\wedge \quad & (((x * y) \div y) \\
& = \quad \text{if } y \simeq 0 \text{ then } 0 \\
& \quad \text{else fix}(x) \text{ endif}
\end{aligned}$$

EVENT: Enable quotient-times-instance-temp; name this event ‘quotient-times-instance-temp-off’.

THEOREM: quotient-times-proof

$$((a \bmod c) = 0) \rightarrow (((a * b) \div c) = (b * (a \div c)))$$

THEOREM: quotient-times

$$\begin{aligned}
& ((a \bmod c) = 0) \\
& \rightarrow \quad (((a * b) \div c) = (b * (a \div c))) \\
& \quad \wedge \quad (((b * a) \div c) = (b * (a \div c)))
\end{aligned}$$

THEOREM: quotient-times-instance

$$\begin{aligned}
& (((y * x) \div y) \\
& = \quad \text{if } y \simeq 0 \text{ then } 0 \\
& \quad \text{else fix}(x) \text{ endif} \\
\wedge \quad & (((x * y) \div y) \\
& = \quad \text{if } y \simeq 0 \text{ then } 0 \\
& \quad \text{else fix}(x) \text{ endif}
\end{aligned}$$

THEOREM: quotient-times-times-proof

$$\begin{aligned}
& ((x * y) \div (x * z)) \\
& = \quad \text{if } x \simeq 0 \text{ then } 0 \\
& \quad \text{else } y \div z \text{ endif}
\end{aligned}$$

THEOREM: quotient-times-times

$$\begin{aligned}
& (((x * y) \div (x * z)) \\
& = \quad \text{if } x \simeq 0 \text{ then } 0 \\
& \quad \text{else } y \div z \text{ endif} \\
\wedge \quad & (((x * z) \div (y * z)) \\
& = \quad \text{if } z \simeq 0 \text{ then } 0 \\
& \quad \text{else } x \div y \text{ endif}
\end{aligned}$$

EVENT: Enable quotient-times-times; name this event ‘quotient-times-times-off’.

THEOREM: quotient-difference1

$$\begin{aligned}
& ((a \bmod c) = (b \bmod c)) \\
& \rightarrow \quad (((a - b) \div c) = ((a \div c) - (b \div c)))
\end{aligned}$$

THEOREM: quotient-lessp-arg1

$$(a < b) \rightarrow ((a \div b) = 0)$$

THEOREM: quotient-difference2

$$\begin{aligned} & (((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \div c) \\ & = \text{if } b < a \text{ then } (a \div c) - (1 + (b \div c)) \\ & \text{else } 0 \text{ endif}) \end{aligned}$$

THEOREM: quotient-difference3

$$\begin{aligned} & (((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)) \\ \rightarrow & (((a - b) \div c) \\ & = \text{if } b < a \text{ then } (a \div c) - (b \div c) \\ & \text{else } 0 \text{ endif}) \end{aligned}$$

THEOREM: remainder-equals-its-first-argument

$$(a = (a \bmod b)) = ((a \in \mathbf{N}) \wedge ((b \simeq 0) \vee (a < b)))$$

EVENT: Enable remainder-equals-its-first-argument; name this event ‘remainder-equals-its-first-argument-off’.

THEOREM: quotient-remainder-times

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-remainder

$$((c \bmod a) = 0) \rightarrow (((b \bmod c) \div a) = ((b \div a) \bmod (c \div a)))$$

THEOREM: quotient-remainder-instance

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-plus-fact

$$\begin{aligned} & (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)) \\ \rightarrow & (((a + b) \div c) = (b \div c)) \end{aligned}$$

THEOREM: quotient-plus-times-times-proof

$$\begin{aligned} & (a < b) \\ \rightarrow & (((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d))) \end{aligned}$$

THEOREM: quotient-plus-times-times

$$\begin{aligned} & (a < b) \\ \rightarrow & (((((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d))) \\ & \wedge (((a + (b * c)) \div (b * d)) \\ & = ((b * c) \div (b * d)))) \end{aligned}$$

THEOREM: quotient-plus-times-times-instance

$$\begin{aligned}
& (a < b) \\
\rightarrow & (((a + (b * c) + (b * d)) \div (b * e)) \\
& = \text{if } b \simeq 0 \text{ then } 0 \\
& \quad \text{else } (c + d) \div e \text{ endif}) \\
\wedge & (((a + (c * b) + (d * b)) \div (e * b)) \\
& = \text{if } b \simeq 0 \text{ then } 0 \\
& \quad \text{else } (d + c) \div e \text{ endif})
\end{aligned}$$

THEOREM: quotient-quotient

$$((b \div a) \div c) = (b \div (a * c))$$

THEOREM: leq-quotient

$$\begin{aligned}
& (a < b) \\
\rightarrow & \text{if } (b \div c) < (a \div c) \text{ then f} \\
& \quad \text{else t endif}
\end{aligned}$$

THEOREM: quotient-1-arg2

$$(n \div 1) = \text{fix}(n)$$

THEOREM: quotient-1-arg1-casesplit

$$(n \simeq 0) \vee (n = 1) \vee (1 < n)$$

THEOREM: quotient-1-arg1

$$\begin{aligned}
& (1 \div n) \\
= & \text{if } n = 1 \text{ then } 1 \\
& \quad \text{else } 0 \text{ endif}
\end{aligned}$$

THEOREM: quotient-x-x

$$(x \not\simeq 0) \rightarrow ((x \div x) = 1)$$

THEOREM: lessp-quotient

$$((i \div j) < i) = ((i \not\simeq 0) \wedge (j \neq 1))$$

DEFINITION:

cancel-quotient-times(x)

$$\begin{aligned}
= & \text{if } (\text{car}(x) = \text{'quotient'}) \\
& \quad \wedge (\text{caadr}(x) = \text{'times'}) \\
& \quad \wedge (\text{caaddr}(x) = \text{'times'}) \\
& \text{then if listp}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \text{times-fringe}(\text{caddr}(x)))) \\
& \quad \text{then cons('if,} \\
& \quad \quad \text{cons}(\text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(\text{cadr}(x)), \\
& \quad \quad \quad \text{times-fringe}(\text{caddr}(x)))), \\
& \quad \quad \text{cons}(\text{list}(\text{'quotient,} \\
& \quad \quad \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)),
\end{aligned}$$

bagint (times-fringe (cadr (x)),
 times-fringe (caddr (x))))),
 times-tree (bagdiff (times-fringe (caddr (x)),
 bagint (times-fringe (cadr (x)),
 times-fringe (caddr (x))))),
 ' ((zero))))))
 else x endif
 else x endif

THEOREM: zerop-makes-quotient-zero-bridge

((car (x) = 'times)
 ∧ (car (y) = 'times)
 ∧ (¬ eval\$ (t,
 and-not-zerop-tree (bagint (times-fringe (x), times-fringe (y))),
 a)))
 → (((eval\$ (t, cadr (x), a) * eval\$ (t, caddr (x), a))
 ÷ (eval\$ (t, cadr (y), a) * eval\$ (t, caddr (y), a)))
 = 0)

EVENT: Enable zerop-makes-quotient-zero-bridge; name this event 'zerop-makes-quotient-zero-bridge-off'.

THEOREM: eval\$-quotient-times-tree-bagdiff

(subbagp (x, y) ∧ subbagp (x, z) ∧ eval\$ (t, and-not-zerop-tree (x), a))
 → ((eval\$ (t, times-tree (bagdiff (y, x)), a)
 ÷ eval\$ (t, times-tree (bagdiff (z, x)), a))
 = (eval\$ (t, times-tree (y), a) ÷ eval\$ (t, times-tree (z), a)))

EVENT: Enable eval\$-quotient-times-tree-bagdiff; name this event 'eval\$-quotient-times-tree-bagdiff-off'.

THEOREM: correctness-of-cancel-quotient-times

eval\$ (t, x, a) = eval\$ (t, cancel-quotient-times (x), a)

EVENT: Let us define the theory *quotients* to consist of the following events: quotient-noop, quotient-of-non-number, quotient-zero, quotient-add1, equal-quotient-0, quotient-sub1, quotient-plus, quotient-times, quotient-times-instance, quotient-difference1, quotient-lessp-arg1, quotient-difference2, quotient-difference3, quotient-remainder-times, quotient-remainder, quotient-remainder-instance, quotient-plus-times-times, quotient-plus-times-times-instance, quotient-quotient, quotient-1-arg2, quotient-1-arg1, quotient-x-x, lessp-quotient, correctness-of-cancel-quotient-times.

DEFINITION:

$\text{exp}(i, j)$
 $=$ **if** $j \simeq 0$ **then** 1
 else $i * \text{exp}(i, j - 1)$ **endif**

DEFINITION:

$\log(\text{base}, n)$
 $=$ **if** $\text{base} < 2$ **then** 0
 elseif $n \simeq 0$ **then** 0
 else $1 + \log(\text{base}, n \div \text{base})$ **endif**

DEFINITION:

$\text{gcd}(x, y)$
 $=$ **if** $x \simeq 0$ **then** $\text{fix}(y)$
 elseif $y \simeq 0$ **then** x
 elseif $x < y$ **then** $\text{gcd}(x, y - x)$
 else $\text{gcd}(x - y, y)$ **endif**

THEOREM: remainder-exp

$(k \not\simeq 0) \rightarrow ((\text{exp}(n, k) \bmod n) = 0)$

DEFINITION:

$\text{double-number-induction}(i, j)$
 $=$ **if** $i \simeq 0$ **then** 0
 elseif $j \simeq 0$ **then** 0
 else $\text{double-number-induction}(i - 1, j - 1)$ **endif**

THEOREM: remainder-exp-exp

if $j < i$ **then** **f**
 else **t** **endif**
 $\rightarrow ((\text{exp}(a, j) \bmod \text{exp}(a, i)) = 0)$

THEOREM: quotient-exp

$(k \not\simeq 0)$
 $\rightarrow ((\text{exp}(n, k) \div n)$
 $=$ **if** $n \simeq 0$ **then** 0
 else $\text{exp}(n, k - 1)$ **endif)**

THEOREM: exp-zero

$(k \simeq 0) \rightarrow (\text{exp}(n, k) = 1)$

THEOREM: exp-add1

$\text{exp}(n, 1 + k) = (n * \text{exp}(n, k))$

THEOREM: exp-plus

$\text{exp}(i, j + k) = (\text{exp}(i, j) * \text{exp}(i, k))$

THEOREM: exp-0-arg1

$$\begin{aligned} & \exp(0, k) \\ &= \text{if } k \simeq 0 \text{ then } 1 \\ & \quad \text{else } 0 \text{ endif} \end{aligned}$$

THEOREM: exp-1-arg1

$$\exp(1, k) = 1$$

THEOREM: exp-0-arg2

$$\exp(n, 0) = 1$$

THEOREM: exp-times

$$\exp(i * j, k) = (\exp(i, k) * \exp(j, k))$$

THEOREM: exp-exp

$$\exp(\exp(i, j), k) = \exp(i, j * k)$$

THEOREM: equal-exp-0

$$(\exp(n, k) = 0) = ((n \simeq 0) \wedge (k \not\simeq 0))$$

THEOREM: equal-exp-1

$$\begin{aligned} & (\exp(n, k) = 1) \\ &= \text{if } k \simeq 0 \text{ then } t \\ & \quad \text{else } n = 1 \text{ endif} \end{aligned}$$

THEOREM: exp-difference

$$\begin{aligned} & (\text{if } b < c \text{ then } f \\ & \quad \text{else } t \text{ endif} \\ & \wedge (a \not\simeq 0)) \\ & \rightarrow (\exp(a, b - c) = (\exp(a, b) \div \exp(a, c))) \end{aligned}$$

EVENT: Let us define the theory *exponentiation* to consist of the following events: equal-exp-0, equal-exp-1, exp-exp, exp-add1, exp-times, exp-1-arg1, exp-zero, exp-0-arg2, exp-0-arg1, exp-difference, exp-plus, quotient-exp, remainder-exp-exp, remainder-exp.

THEOREM: equal-log-0

$$(\log(base, n) = 0) = ((base < 2) \vee (n \simeq 0))$$

THEOREM: log-0

$$(n \simeq 0) \rightarrow (\log(base, n) = 0)$$

THEOREM: log-1

$$(1 < base) \rightarrow (\log(base, 1) = 1)$$

DEFINITION:

double-log-induction($base, a, b$)

= **if** $base < 2$ **then** 0
 elseif $a \simeq 0$ **then** 0
 elseif $b \simeq 0$ **then** 0
 else double-log-induction($base, a \div base, b \div base$) **endif**

THEOREM: leq-log-log

if $m < n$ **then** **f**
else **t** **endif**
 $\rightarrow$ **if** $\log(c, m) < \log(c, n)$ **then** **f**
 else **t** **endif**

THEOREM: log-quotient

$(1 < c) \rightarrow (\log(c, n \div c) = (\log(c, n) - 1))$

THEOREM: log-quotient-times-proof

$(1 < c) \rightarrow (\log(c, n \div (c * m)) = (\log(c, n \div m) - 1))$

THEOREM: log-quotient-times

$(1 < c)$
 $\rightarrow ((\log(c, n \div (c * m)) = (\log(c, n \div m) - 1))$
 $\wedge (\log(c, n \div (m * c)) = (\log(c, n \div m) - 1)))$

THEOREM: log-quotient-exp

$(1 < c) \rightarrow (\log(c, n \div \exp(c, m)) = (\log(c, n) - m))$

THEOREM: log-times-proof

$((1 < c) \wedge (n \not\simeq 0)) \rightarrow (\log(c, c * n) = (1 + \log(c, n)))$

THEOREM: log-times

$((1 < c) \wedge (n \not\simeq 0))$
 $\rightarrow ((\log(c, c * n) = (1 + \log(c, n)))$
 $\wedge (\log(c, n * c) = (1 + \log(c, n))))$

THEOREM: log-times-exp-proof

$((1 < c) \wedge (n \not\simeq 0)) \rightarrow (\log(c, n * \exp(c, m)) = (\log(c, n) + m))$

THEOREM: log-times-exp

$((1 < c) \wedge (n \not\simeq 0))$
 $\rightarrow ((\log(c, n * \exp(c, m)) = (\log(c, n) + m))$
 $\wedge (\log(c, \exp(c, m) * n) = (\log(c, n) + m)))$

THEOREM: log-exp

$(1 < c) \rightarrow (\log(c, \exp(c, n)) = (1 + n))$

EVENT: Let us define the theory *logs* to consist of the following events: log-exp, log-times-exp, log-times, log-quotient-exp, log-quotient-times, log-quotient, log-1, log-0, equal-log-0, exp-exp.

THEOREM: commutativity-of-gcd
 $\text{gcd}(b, a) = \text{gcd}(a, b)$

DEFINITION:
single-number-induction(n)
= **if** $n \simeq 0$ **then** 0
else single-number-induction($n - 1$) **endif**

THEOREM: gcd-0
 $(\text{gcd}(0, x) = \text{fix}(x)) \wedge (\text{gcd}(x, 0) = \text{fix}(x))$

THEOREM: gcd-1
 $(\text{gcd}(1, x) = 1) \wedge (\text{gcd}(x, 1) = 1)$

THEOREM: equal-gcd-0
 $(\text{gcd}(a, b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: lessp-gcd
 $(b \not\simeq 0) \rightarrow (((b < \text{gcd}(a, b)) = \mathbf{f}) \wedge ((b < \text{gcd}(b, a)) = \mathbf{f}))$

THEOREM: gcd-plus-instance-temp-proof
 $\text{gcd}(a, a + b) = \text{gcd}(a, b)$

THEOREM: gcd-plus-instance-temp
 $(\text{gcd}(a, a + b) = \text{gcd}(a, b)) \wedge (\text{gcd}(a, b + a) = \text{gcd}(a, b))$

THEOREM: gcd-plus-proof
 $((b \mathbf{mod} a) = 0) \rightarrow (\text{gcd}(a, b + c) = \text{gcd}(a, c))$

THEOREM: gcd-plus
 $((b \mathbf{mod} a) = 0)$
 $\rightarrow ((\text{gcd}(a, b + c) = \text{gcd}(a, c))$
 $\quad \wedge (\text{gcd}(a, c + b) = \text{gcd}(a, c))$
 $\quad \wedge (\text{gcd}(b + c, a) = \text{gcd}(a, c))$
 $\quad \wedge (\text{gcd}(c + b, a) = \text{gcd}(a, c)))$

THEOREM: gcd-plus-instance
 $(\text{gcd}(a, a + b) = \text{gcd}(a, b)) \wedge (\text{gcd}(a, b + a) = \text{gcd}(a, b))$

THEOREM: remainder-gcd
 $((a \mathbf{mod} \text{gcd}(a, b)) = 0) \wedge ((b \mathbf{mod} \text{gcd}(a, b)) = 0)$

THEOREM: distributivity-of-times-over-gcd-proof

$$\text{gcd}(x * z, y * z) = (z * \text{gcd}(x, y))$$

THEOREM: distributivity-of-times-over-gcd

$$\begin{aligned} &(\text{gcd}(x * z, y * z) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(z * x, y * z) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(x * z, z * y) = (z * \text{gcd}(x, y))) \\ \wedge &(\text{gcd}(z * x, z * y) = (z * \text{gcd}(x, y))) \end{aligned}$$

THEOREM: gcd-is-the-greatest

$$\begin{aligned} &((x \neq 0) \wedge (y \neq 0) \wedge ((x \bmod z) = 0) \wedge ((y \bmod z) = 0)) \\ \rightarrow &\text{if } \text{gcd}(x, y) < z \text{ then } \mathbf{f} \\ &\text{else } \mathbf{t} \text{ endif} \end{aligned}$$

THEOREM: common-divisor-divides-gcd

$$(((x \bmod z) = 0) \wedge ((y \bmod z) = 0)) \rightarrow ((\text{gcd}(x, y) \bmod z) = 0)$$

THEOREM: associativity-of-gcd-zero-case

$$((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c)))$$

THEOREM: associativity-of-gcd

$$\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c))$$

THEOREM: commutativity2-of-gcd-zero-case

$$((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(b, \text{gcd}(a, c)) = \text{gcd}(a, \text{gcd}(b, c)))$$

THEOREM: commutativity2-of-gcd

$$\text{gcd}(b, \text{gcd}(a, c)) = \text{gcd}(a, \text{gcd}(b, c))$$

THEOREM: gcd-x-x

$$\text{gcd}(x, x) = \text{fix}(x)$$

THEOREM: gcd-idempotence

$$(\text{gcd}(x, \text{gcd}(x, y)) = \text{gcd}(x, y)) \wedge (\text{gcd}(y, \text{gcd}(x, y)) = \text{gcd}(x, y))$$

EVENT: Let us define the theory *gcds* to consist of the following events: commutativity2-of-gcd, associativity-of-gcd, common-divisor-divides-gcd, distributivity-of-times-over-gcd, lessp-gcd, equal-gcd-0, gcd-0, gcd-idempotence, gcd-x-x, remainder-gcd, gcd-plus, gcd-plus-instance, gcd-1, commutativity-of-gcd.

EVENT: Let us define the theory *naturals* to consist of the following events: addition, multiplication, remainders, quotients, exponentiation, logs, gcds.

DEFINITION:

$$\begin{aligned} &\text{integerp}(x) \\ = &\text{if } x \in \mathbf{N} \text{ then } \mathbf{t} \\ &\text{elseif } \text{negativep}(x) \text{ then } \text{negative-guts}(x) \neq 0 \\ &\text{else } \mathbf{f} \text{ endif} \end{aligned}$$

DEFINITION:

$\text{fix-int}(x)$
= **if** $\text{integerp}(x)$ **then** x
 else 0 **endif**

DEFINITION: $\text{izerop}(i) = (\text{fix-int}(i) = 0)$

DEFINITION:

$\text{ilessp}(i, j)$
= **if** $\text{negativep}(i)$
 then if $\text{negativep}(j)$ **then** $\text{negative-guts}(j) < \text{negative-guts}(i)$
 elseif $i = (-\ 0)$ **then** $0 < j$
 else t endif
 elseif $\text{negativep}(j)$ **then f**
 else $i < j$ **endif**

DEFINITION: $\text{ileq}(i, j) = (\neg \text{ilessp}(j, i))$

DEFINITION:

$\text{iplus}(x, y)$
= **if** $\text{negativep}(x)$
 then if $\text{negativep}(y)$
 then if $(\text{negative-guts}(x) \simeq 0) \wedge (\text{negative-guts}(y) \simeq 0)$ **then** 0
 else $-(\text{negative-guts}(x) + \text{negative-guts}(y))$ **endif**
 elseif $y < \text{negative-guts}(x)$ **then** $-(\text{negative-guts}(x) - y)$
 else $y - \text{negative-guts}(x)$ **endif**
 elseif $\text{negativep}(y)$
 then if $x < \text{negative-guts}(y)$ **then** $-(\text{negative-guts}(y) - x)$
 else $x - \text{negative-guts}(y)$ **endif**
 else $x + y$ **endif**

DEFINITION:

$\text{ineg}(x)$
= **if** $\text{negativep}(x)$ **then** $\text{negative-guts}(x)$
 elseif $x \simeq 0$ **then** 0
 else $-x$ **endif**

DEFINITION: $\text{idifference}(x, y) = \text{iplus}(x, \text{ineg}(y))$

DEFINITION:

$\text{iabs}(i)$
= **if** $\text{negativep}(i)$ **then** $\text{negative-guts}(i)$
 else $\text{fix}(i)$ **endif**

DEFINITION:

```

itimes( $i, j$ )
=  if negativep( $i$ )
    then if negativep( $j$ ) then negative-guts( $i$ ) * negative-guts( $j$ )
        else fix-int( $-($  negative-guts( $i$ ) *  $j$   $)$ ) endif
    elseif negativep( $j$ ) then fix-int( $-($   $i$  * negative-guts( $j$ )  $)$ )
    else  $i$  *  $j$  endif

```

DEFINITION:

```

iquotient( $i, j$ )
=  if fix-int( $j$ ) = 0 then 0
    elseif negativep( $i$ )
    then if negativep( $j$ )
        then if (negative-guts( $i$ ) mod negative-guts( $j$ )) = 0
            then negative-guts( $i$ )  $\div$  negative-guts( $j$ )
            else 1 + (negative-guts( $i$ )  $\div$  negative-guts( $j$ )) endif
        elseif (negative-guts( $i$ ) mod  $j$ ) = 0
            then fix-int( $-($  negative-guts( $i$ )  $\div$   $j$   $)$ )
            else fix-int( $-($  1 + (negative-guts( $i$ )  $\div$   $j$ )  $)$ ) endif
        elseif negativep( $j$ ) then fix-int( $-($   $i$   $\div$  negative-guts( $j$ )  $)$ )
        else  $i$   $\div$   $j$  endif

```

DEFINITION:

iremainder(i, j) = idifference(i , itimes(j , quotient(i, j)))

DEFINITION:

```

idiv( $i, j$ )
=  if fix-int( $j$ ) = 0 then 0
    elseif negativep( $i$ )
    then if negativep( $j$ ) then negative-guts( $i$ )  $\div$  negative-guts( $j$ )
        elseif (negative-guts( $i$ ) mod  $j$ ) = 0
            then fix-int( $-($  negative-guts( $i$ )  $\div$   $j$   $)$ )
            else fix-int( $-($  1 + (negative-guts( $i$ )  $\div$   $j$ )  $)$ ) endif
        elseif negativep( $j$ )
        then if ( $i$  mod negative-guts( $j$ )) = 0
            then fix-int( $-($   $i$   $\div$  negative-guts( $j$ )  $)$ )
            else fix-int( $-($  1 + ( $i$   $\div$  negative-guts( $j$ ))  $)$ ) endif
        else  $i$   $\div$   $j$  endif

```

DEFINITION:

imod(i, j) = idifference(fix-int(i), itimes(j , idiv(i, j)))

DEFINITION:

```

iquo( $i, j$ )
=  if fix-int( $j$ ) = 0 then 0
    elseif negativep( $i$ )

```

```

then if negativep( $j$ ) then negative-guts( $i$ )  $\div$  negative-guts( $j$ )
      else fix-int( $-(\text{negative-guts}(i) \div j)$ ) endif
elseif negativep( $j$ ) then fix-int( $-(i \div \text{negative-guts}(j))$ )
else  $i \div j$  endif

```

DEFINITION:

$\text{irem}(i, j) = \text{idifference}(\text{fix-int}(i), \text{itimes}(j, \text{iquo}(i, j)))$

EVENT: Let us define the theory *integer-defns* to consist of the following events: integerp, fix-int, illessp, iplus, ineg, iabs, itimes, iquotient, iremainder, idiv, imod, iquo, irem.

EVENT: Let us define the theory *all-integer-defns* to consist of the following events: integerp, fix-int, izerop, illessp, ileq, iplus, ineg, idifference, iabs, itimes, iquotient, iremainder, idiv, imod, iquo, irem.

EVENT: Enable integerp; name this event ‘integerp-off’.

EVENT: Enable fix-int; name this event ‘fix-int-off’.

EVENT: Enable illessp; name this event ‘illessp-off’.

EVENT: Enable iplus; name this event ‘iplus-off’.

EVENT: Enable ineg; name this event ‘ineg-off’.

EVENT: Enable iabs; name this event ‘iabs-off’.

EVENT: Enable itimes; name this event ‘itimes-off’.

THEOREM: integerp-fix-int
integerp(fix-int(x))

THEOREM: integerp-iplus
integerp(iplus(x, y))

THEOREM: integerp-idifference
integerp(idifference(x, y))

THEOREM: integerp-ineg
integerp(ineg(x))

THEOREM: integerp-iabs
 $\text{integerp}(\text{iabs}(x))$

THEOREM: integerp-itimes
 $\text{integerp}(\text{itimes}(x, y))$

THEOREM: fix-int-remover
 $\text{integerp}(x) \rightarrow (\text{fix-int}(x) = x)$

THEOREM: fix-int-fix-int
 $\text{fix-int}(\text{fix-int}(x)) = \text{fix-int}(x)$

THEOREM: fix-int-iplus
 $\text{fix-int}(\text{iplus}(a, b)) = \text{iplus}(a, b)$

THEOREM: fix-int-idifference
 $\text{fix-int}(\text{idifference}(a, b)) = \text{idifference}(a, b)$

THEOREM: fix-int-ineg
 $\text{fix-int}(\text{ineg}(x)) = \text{ineg}(x)$

THEOREM: fix-int-iabs
 $\text{fix-int}(\text{iabs}(x)) = \text{iabs}(x)$

THEOREM: fix-int-itimes
 $\text{fix-int}(\text{itimes}(x, y)) = \text{itimes}(x, y)$

THEOREM: neg-iplus
 $\text{neg}(\text{iplus}(a, b)) = \text{iplus}(\text{neg}(a), \text{neg}(b))$

THEOREM: neg-ineg
 $\text{neg}(\text{ineg}(x)) = \text{fix-int}(x)$

THEOREM: neg-fix-int
 $\text{neg}(\text{fix-int}(x)) = \text{ineg}(x)$

THEOREM: neg-of-non-integerp
 $(\neg \text{integerp}(x)) \rightarrow (\text{neg}(x) = 0)$

EVENT: Enable neg-of-non-integerp; name this event ‘neg-of-non-integerp-off’.

THEOREM: neg-0
 $\text{neg}(0) = 0$

THEOREM: iplus-left-id
 $(\neg \text{integerp}(x)) \rightarrow (\text{iplus}(x, y) = \text{fix-int}(y))$

EVENT: Enable iplus-left-id; name this event ‘iplus-left-id-off’.

THEOREM: iplus-right-id
 $(\neg \text{integerp}(y)) \rightarrow (\text{ipplus}(x, y) = \text{fix-int}(x))$

EVENT: Enable iplus-right-id; name this event ‘iplus-right-id-off’.

THEOREM: iplus-0-left
 $\text{ipplus}(0, x) = \text{fix-int}(x)$

THEOREM: iplus-0-right
 $\text{ipplus}(x, 0) = \text{fix-int}(x)$

THEOREM: commutativity2-of-ipplus
 $\text{ipplus}(x, \text{ipplus}(y, z)) = \text{ipplus}(y, \text{ipplus}(x, z))$

THEOREM: commutativity-of-ipplus
 $\text{ipplus}(x, y) = \text{ipplus}(y, x)$

THEOREM: associativity-of-ipplus
 $\text{ipplus}(\text{ipplus}(x, y), z) = \text{ipplus}(x, \text{ipplus}(y, z))$

THEOREM: iplus-cancellation-1
 $(\text{ipplus}(a, b) = \text{ipplus}(a, c)) = (\text{fix-int}(b) = \text{fix-int}(c))$

THEOREM: iplus-cancellation-2
 $(\text{ipplus}(b, a) = \text{ipplus}(c, a)) = (\text{fix-int}(b) = \text{fix-int}(c))$

THEOREM: iplus-ineg1
 $\text{ipplus}(\text{ineg}(a), a) = 0$

THEOREM: iplus-ineg2
 $\text{ipplus}(a, \text{ineg}(a)) = 0$

THEOREM: iplus-fix-int1
 $\text{ipplus}(\text{fix-int}(a), b) = \text{ipplus}(a, b)$

THEOREM: iplus-fix-int2
 $\text{ipplus}(a, \text{fix-int}(b)) = \text{ipplus}(a, b)$

THEOREM: idifference-fix-int1
 $\text{idifference}(\text{fix-int}(a), b) = \text{idifference}(a, b)$

THEOREM: idifference-fix-int2
 $\text{idifference}(a, \text{fix-int}(b)) = \text{idifference}(a, b)$

DEFINITION:

cancel-ineg-aux(x, y)

```
=  if listp( $x$ )
    then if car( $x$ ) = 'ineg
        then if  $y$  = cadr( $x$ ) then ''0
            elseif listp( $y$ )
                then if car( $y$ ) = 'iplus
                    then if cadr( $y$ ) = cadr( $x$ )
                        then list('fix-int, cadr( $y$ ))
                        elseif cancel-ineg-aux( $x$ , cadr( $y$ ))
                        then list('iplus,
                                cadr( $y$ ),
                                cancel-ineg-aux( $x$ , cadr( $y$ )))
                    else f endif
                else f endif
            else f endif
        elseif  $y \simeq \text{nil}$  then f
        elseif car( $y$ ) = 'ineg
            then if  $x$  = cadr( $y$ ) then ''0
                else f endif
            elseif car( $y$ ) = 'iplus
                then if listp(cadr( $y$ ))
                    then if caadr( $y$ ) = 'ineg
                        then if  $x$  = cadadr( $y$ )
                            then list('fix-int, cadr( $y$ ))
                            elseif cancel-ineg-aux( $x$ , cadr( $y$ ))
                            then list('iplus,
                                    cadr( $y$ ),
                                    cancel-ineg-aux( $x$ , cadr( $y$ )))
                        else f endif
                    elseif cancel-ineg-aux( $x$ , cadr( $y$ ))
                        then list('iplus, cadr( $y$ ), cancel-ineg-aux( $x$ , cadr( $y$ )))
                    else f endif
                elseif cancel-ineg-aux( $x$ , cadr( $y$ ))
                    then list('iplus, cadr( $y$ ), cancel-ineg-aux( $x$ , cadr( $y$ )))
                else f endif
            else f endif
        elseif  $y \simeq \text{nil}$  then f
        elseif car( $y$ ) = 'ineg
            then if  $x$  = cadr( $y$ ) then ''0
                else f endif
            elseif car( $y$ ) = 'iplus
                then if listp(cadr( $y$ ))
                    then if caadr( $y$ ) = 'ineg
```



```

      then if  $x = \text{cadadr}(y)$  then list('fix-int, caddr( $y$ ))
        elseif cancel-ineg-aux( $x$ , caddr( $y$ ))
          then list('iplus, cadr( $y$ ), cancel-ineg-aux( $x$ , caddr( $y$ )))
          else f endif
        elseif cancel-ineg-aux( $x$ , caddr( $y$ ))
          then list('iplus, cadr( $y$ ), cancel-ineg-aux( $x$ , caddr( $y$ )))
          else f endif
        elseif cancel-ineg-aux( $x$ , caddr( $y$ ))
          then list('iplus, cadr( $y$ ), cancel-ineg-aux( $x$ , caddr( $y$ )))
          else f endif
      else f endif

```

DEFINITION:

```

cancel-ineg( $x$ )
= if listp( $x$ )
  then if car( $x$ ) = 'iplus
    then if cancel-ineg-aux(cadr( $x$ ), caddr( $x$ ))
      then cancel-ineg-aux(cadr( $x$ ), caddr( $x$ ))
      else  $x$  endif
    else  $x$  endif
  else  $x$  endif

```

THEOREM: eval\$-list-cons

$\text{eval\$}('list, \text{cons}(x, y), a) = \text{cons}(\text{eval\$}(\mathbf{t}, x, a), \text{eval\$}('list, y, a))$

THEOREM: eval\$-list-nlistp

$(x \simeq \mathbf{nil}) \rightarrow (\text{eval\$}('list, x, a) = \mathbf{nil})$

THEOREM: eval\$-litatom

$\text{litatom}(x) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) = \text{cdr}(\text{assoc}(x, a)))$

EVENT: Disable eval\$-quote; name this event 'eval\$-quote-on'.

THEOREM: eval\$-other

$((\neg \text{litatom}(x)) \wedge (x \simeq \mathbf{nil})) \rightarrow (\text{eval\$}(\mathbf{t}, x, a) = x)$

EVENT: Enable eval\$; name this event 'eval\$-off'.

DEFINITION:

```

eval$-cancel-ineg-aux-fn( $x, y, a$ )
= if listp( $x$ )  $\wedge$  (car( $x$ ) = 'ineg)
  then if  $y = \text{cadr}(x)$  then 0
    elseif cadr( $y$ ) = cadr( $x$ ) then fix-int(eval$(\mathbf{t}, caddr( $y$ ),  $a$ ))
    elseif listp( $y$ )

```

```

      then iplus(eval$(t, cadr(y), a),
                 eval$-cancel-ineg-aux-fn(x, caddr(y), a))
    else 0 endif
  elseif car(y) = 'ineg then 0
elseif listp(cadr(y))
  ∧ (caadr(y) = 'ineg)
  ∧ (x = cadadr(y)) then fix-int(eval$(t, caddr(y), a))
elseif listp(y)
then iplus(eval$(t, cadr(y), a), eval$-cancel-ineg-aux-fn(x, caddr(y), a))
else 0 endif

```

THEOREM: eval\$-cancel-ineg-aux-is-its-fn
 $(\text{cancel-ineg-aux}(x, y) \neq \mathbf{f})$
 $\rightarrow (\text{eval}(\mathbf{t}, \text{cancel-ineg-aux}(x, y), a) = \text{eval$-cancel-ineg-aux-fn}(x, y, a))$

THEOREM: iplus-ineg3
 $\text{ipus}(\text{ineg}(x), \text{ipus}(x, y)) = \text{fix-int}(y)$

THEOREM: iplus-ineg4
 $\text{ipus}(x, \text{ipus}(\text{ineg}(x), y)) = \text{fix-int}(y)$

THEOREM: iplus-ineg-promote
 $\text{ipus}(y, \text{ineg}(x)) = \text{ipus}(\text{ineg}(x), y)$

THEOREM: iplus-x-y-ineg-x
 $\text{ipus}(x, \text{ipus}(y, \text{ineg}(x))) = \text{fix-int}(y)$

EVENT: Enable iplus-ineg-promote; name this event 'ipus-ineg-promote-off'.

THEOREM: correctness-of-cancel-ineg-aux
 $(\text{cancel-ineg-aux}(x, y) \neq \mathbf{f})$
 $\rightarrow (\text{eval$-cancel-ineg-aux-fn}(x, y, a) = \text{ipus}(\text{eval}(\mathbf{t}, x, a), \text{eval}(\mathbf{t}, y, a)))$

THEOREM: correctness-of-cancel-ineg
 $\text{eval}(\mathbf{t}, x, a) = \text{eval}(\mathbf{t}, \text{cancel-ineg}(x), a)$

EVENT: Enable correctness-of-cancel-ineg-aux; name this event 'correctness-of-cancel-ineg-aux-off'.

DEFINITION:
 $\text{ipus-fringe}(x)$
 $=$ **if** listp(x)
 then if car(x) = 'ipus
 then append(ipus-fringe(cadr(x)), ipus-fringe(caddr(x)))
 else list(x) **endif**
 else list(x) **endif**

THEOREM: lessp-count-listp-cdr
 $\text{listp}(\text{cdr}(x)) \rightarrow (\text{count}(\text{cdr}(x)) < \text{count}(x))$

DEFINITION:
 $\text{iplus-tree-rec}(l)$
 $=$ **if** $\text{cdr}(l) \simeq \text{nil}$ **then** $\text{car}(l)$
 else $\text{list}(' \text{iplus}, \text{car}(l), \text{iplus-tree-rec}(\text{cdr}(l)))$ **endif**

DEFINITION:
 $\text{iplus-tree}(l)$
 $=$ **if** $\text{listp}(l)$
 then if $\text{listp}(\text{cdr}(l))$ **then** $\text{iplus-tree-rec}(l)$
 else $\text{list}(' \text{fix-int}, \text{car}(l))$ **endif**
 else $'0$ **endif**

DEFINITION:
 $\text{iplus-list}(x)$
 $=$ **if** $\text{listp}(x)$ **then** $\text{iplus}(\text{car}(x), \text{iplus-list}(\text{cdr}(x)))$
 else 0 **endif**

THEOREM: integerp-iplus-list
 $\text{integerp}(\text{iplus-list}(x))$

THEOREM: eval\$-iplus-tree-rec
 $\text{eval}\$(\mathbf{t}, \text{iplus-tree-rec}(x), a)$
 $=$ **if** $\text{listp}(x)$
 then if $\text{listp}(\text{cdr}(x))$ **then** $\text{iplus-list}(\text{eval}\$(' \text{list}, x, a))$
 else $\text{eval}\$(\mathbf{t}, \text{car}(x), a)$ **endif**
 else 0 **endif**

THEOREM: eval\$-iplus-tree
 $\text{eval}\$(\mathbf{t}, \text{iplus-tree}(x), a) = \text{iplus-list}(\text{eval}\$(' \text{list}, x, a))$

THEOREM: eval\$-list-append
 $\text{eval}\$(' \text{list}, \text{append}(x, y), a)$
 $=$ $\text{append}(\text{eval}\$(' \text{list}, x, a), \text{eval}\$(' \text{list}, y, a))$

DEFINITION:
 $\text{cancel-iplus}(x)$
 $=$ **if** $\text{listp}(x)$
 then if $\text{car}(x) = ' \text{equal}$
 then if $\text{listp}(\text{cadr}(x))$
 then if $\text{caadr}(x) = ' \text{iplus}$
 then if $\text{listp}(\text{caddr}(x))$
 then if $\text{caaddr}(x) = ' \text{iplus}$

```

then if listp (bagint (iplus-fringe (cadr (x)),
                        iplus-fringe (caddr (x))))
    then list ('equal,
              iplus-tree (bagdiff (iplus-fringe (cadr (x)),
                                   bagint (iplus-fringe (cadr (x)),
                                           iplus-fringe (caddr (x))))),
              iplus-tree (bagdiff (iplus-fringe (caddr (x)),
                                   bagint (iplus-fringe (cadr (x)),
                                           iplus-fringe (caddr (x))))))
    else x endif
elseif caddr (x) ∈ iplus-fringe (cadr (x))
then list ('if,
            list ('integerp, caddr (x)),
            cons ('equal,
                  cons (iplus-tree (delete (caddr (x),
                                             iplus-fringe (cadr (x)))),
                        ' ('0))),
            list ('quote, f))
    else x endif
elseif caddr (x) ∈ iplus-fringe (cadr (x))
then list ('if,
            list ('integerp, caddr (x)),
            cons ('equal,
                  cons (iplus-tree (delete (caddr (x),
                                             iplus-fringe (cadr (x)))),
                        ' ('0))),
            list ('quote, f))
    else x endif
elseif listp (caddr (x))
then if caaddr (x) = 'iplus
    then if cadr (x) ∈ iplus-fringe (caddr (x))
        then list ('if,
                    list ('integerp, cadr (x)),
                    list ('equal,
                          ' '0,
                          iplus-tree (delete (cadr (x),
                                              iplus-fringe (caddr (x))))),
                    list ('quote, f))
        else x endif
    else x endif
elseif listp (caddr (x))
then if caaddr (x) = 'iplus
    then if cadr (x) ∈ iplus-fringe (caddr (x))

```

```

        then list('if,
                  list('integerp, cadr(x)),
                  list('equal,
                      '0,
                      iplus-tree(delete(cadr(x),
                                      iplus-fringe(caddr(x))))),
                  list('quote, f))
        else x endif
    else x endif
else x endif
else x endif

```

THEOREM: eval\$-cancel-iplus

```

eval$(t, cancel-iplus(x), a)
=  if listp(x) ∧ (car(x) = 'equal)
    then if listp(cadr(x))
        ∧ (caadr(x) = 'iplus)
        ∧ listp(caddr(x))
        ∧ (caaddr(x) = 'iplus)
    then if listp(bagint(iplus-fringe(cadr(x)),
                        iplus-fringe(caddr(x))))
        then iplus-list(eval$('list,
                                bagdiff(iplus-fringe(cadr(x)),
                                    bagint(iplus-fringe(cadr(x)),
                                        iplus-fringe(caddr(x)))),
                                a))
        =  iplus-list(eval$('list,
                                bagdiff(iplus-fringe(caddr(x)),
                                    bagint(iplus-fringe(cadr(x)),
                                        iplus-fringe(caddr(x)))),
                                a))
        else eval$(t, x, a) endif
    elseif listp(cadr(x))
        ∧ (caadr(x) = 'iplus)
        ∧ (caddr(x) ∈ iplus-fringe(cadr(x)))
    then if integerp(eval$(t, caddr(x), a))
        then iplus-list(eval$('list,
                                delete(caddr(x), iplus-fringe(cadr(x))),
                                a))
        =  0
        else f endif
    elseif listp(caddr(x))
        ∧ (caaddr(x) = 'iplus)

```

```

       $\wedge$  (cadr( $x$ )  $\in$  iplus-fringe(caddr( $x$ )))
then if integerp(eval$(t, cadr( $x$ ),  $a$ ))
  then 0 = iplus-list(eval$('list,
                                delete(cadr( $x$ ),
                                      iplus-fringe(caddr( $x$ ))),
                                 $a$ ))
  else f endif
else eval$(t,  $x$ ,  $a$ ) endif
else eval$(t,  $x$ ,  $a$ ) endif

```

EVENT: Enable cancel-iplus; name this event 'cancel-iplus-off'.

THEOREM: eval\$-iplus-list-delete

```

( $z \in y$ )
 $\rightarrow$  (iplus-list(eval$('list, delete( $z$ ,  $y$ ),  $a$ ))
      = idifference(iplus-list(eval$('list,  $y$ ,  $a$ )), eval$(t,  $z$ ,  $a$ )))

```

THEOREM: eval\$-iplus-list-bagdiff

```

subbagp( $x$ ,  $y$ )
 $\rightarrow$  (iplus-list(eval$('list, bagdiff( $y$ ,  $x$ ),  $a$ ))
      = idifference(iplus-list(eval$('list,  $y$ ,  $a$ )),
                    iplus-list(eval$('list,  $x$ ,  $a$ ))))

```

THEOREM: iplus-list-append

```

iplus-list(append( $x$ ,  $y$ )) = iplus(iplus-list( $x$ ), iplus-list( $y$ ))

```

EVENT: Enable iplus-tree; name this event 'iplus-tree-off'.

THEOREM: not-integerp-implies-not-equal-iplus

```

( $\neg$  integerp( $a$ ))  $\rightarrow$  (( $a$  = iplus( $b$ ,  $c$ )) = f)

```

THEOREM: iplus-list-eval\$-fringe

```

iplus-list(eval$('list, iplus-fringe( $x$ ),  $a$ )) = fix-int(eval$(t,  $x$ ,  $a$ ))

```

THEOREM: iplus-ineg5-lemma-1

```

integerp( $x$ )
 $\rightarrow$  (( $x$  = iplus( $y$ , iplus(ineg( $z$ ),  $w$ )))
      = ( $x$  = iplus(ineg( $z$ ), iplus( $y$ ,  $w$ ))))

```

THEOREM: iplus-ineg5-lemma-2

```

(integerp( $x$ )  $\wedge$  integerp( $v$ ))
 $\rightarrow$  (( $x$  = iplus(ineg( $z$ ),  $v$ )) = (iplus( $x$ ,  $z$ ) =  $v$ ))

```

THEOREM: iplus-ineg5

```

integerp( $x$ )
 $\rightarrow$  (( $x$  = iplus( $y$ , iplus(ineg( $z$ ),  $w$ ))) = (iplus( $x$ ,  $z$ ) = iplus( $y$ ,  $w$ )))

```

EVENT: Enable iplus-ineg5-lemma-1; name this event ‘iplus-ineg5-lemma-1-off’.

EVENT: Enable iplus-ineg5-lemma-2; name this event ‘iplus-ineg5-lemma-2-off’.

THEOREM: iplus-ineg6

integerp (x)
 $\rightarrow ((x = \text{iplus}(y, \text{iplus}(w, \text{ineg}(z)))) = (\text{iplus}(x, z) = \text{iplus}(y, w)))$

THEOREM: eval\$-iplus

(listp (x) $\wedge$ (car (x) = ‘**iplus**’))
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, a) = \text{iplus}(\text{eval}\$(\mathbf{t}, \text{cadr}(x), a), \text{eval}\$(\mathbf{t}, \text{caddr}(x), a)))$

THEOREM: iplus-ineg7

(0 = $\text{iplus}(x, \text{ineg}(y))$) = (fix-int (y) = fix-int (x))

THEOREM: correctness-of-cancel-iplus

$\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-iplus}(x), a)$

EVENT: Enable iplus-ineg5; name this event ‘iplus-ineg5-off’.

EVENT: Enable iplus-ineg6; name this event ‘iplus-ineg6-off’.

THEOREM: illessp-fix-int-1

illessp (fix-int (x), y) = illessp (x , y)

THEOREM: illessp-fix-int-2

illessp (x , fix-int (y)) = illessp (x , y)

DEFINITION:

make-cancel-iplus-inequality-1 (x , y)
 $= \text{list}(\text{‘illessp’},$
 $\text{iplus-tree}(\text{bagdiff}(x, \text{bagint}(x, y))),$
 $\text{iplus-tree}(\text{bagdiff}(y, \text{bagint}(x, y))))$

DEFINITION:

cancel-iplus-illessp-1 (x)
 $= \text{if listp}(x)$
 then if car (x) = ‘**illessp**’
 then make-cancel-iplus-inequality-1 (iplus-fringe (cadr (x)),
 iplus-fringe (caddr (x)))
 else x **endif**
 else x **endif**

THEOREM: lessp-difference-plus-arg1

$$(w < ((w + y) - x)) = (x < y)$$

THEOREM: lessp-difference-plus-arg1-commuted

$$(w < ((y + w) - x)) = (x < y)$$

THEOREM: iplus-cancellation-1-for-ilessp

$$\text{ilessp}(\text{iplus}(a, b), \text{iplus}(a, c)) = \text{ilessp}(b, c)$$

THEOREM: iplus-cancellation-2-for-ilessp

$$\text{ilessp}(\text{iplus}(b, a), \text{iplus}(c, a)) = \text{ilessp}(b, c)$$

THEOREM: correctness-of-cancel-iplus-ilessp-lemma

$$\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-iplus-ilessp-1}(x), a)$$

DEFINITION:

$$\begin{aligned} &\text{iplus-tree-no-fix-int}(l) \\ &= \text{if listp}(l) \text{ then iplus-tree-rec}(l) \\ &\quad \text{else } '0 \text{ endif} \end{aligned}$$

THEOREM: eval\$-ilessp-iplus-tree-no-fix-int

$$\begin{aligned} &\text{ilessp}(\text{eval}\$(\mathbf{t}, \text{iplus-tree-no-fix-int}(x), a), \\ &\quad \text{eval}\$(\mathbf{t}, \text{iplus-tree-no-fix-int}(y), a)) \\ &= \text{ilessp}(\text{eval}\$(\mathbf{t}, \text{iplus-tree}(x), a), \text{eval}\$(\mathbf{t}, \text{iplus-tree}(y), a)) \end{aligned}$$

EVENT: Enable iplus-tree-no-fix-int; name this event 'iplus-tree-no-fix-int-off'.

THEOREM: make-cancel-iplus-inequality-simplifier

$$\begin{aligned} &\text{eval}\$(\mathbf{t}, \text{make-cancel-iplus-inequality-1}(x, y), a) \\ &= \text{eval}\$(\mathbf{t}, \\ &\quad \text{list}('ilessp, \\ &\quad \quad \text{iplus-tree-no-fix-int}(\text{bagdiff}(x, \text{bagint}(x, y))), \\ &\quad \quad \text{iplus-tree-no-fix-int}(\text{bagdiff}(y, \text{bagint}(x, y)))), \\ &\quad a) \end{aligned}$$

DEFINITION:

$$\begin{aligned} &\text{cancel-iplus-ilessp}(x) \\ &= \text{if listp}(x) \\ &\quad \text{then if car}(x) = 'ilessp \\ &\quad \quad \text{then if listp}(\text{bagint}(\text{iplus-fringe}(\text{cadr}(x)), \\ &\quad \quad \quad \text{iplus-fringe}(\text{caddr}(x)))) \\ &\quad \quad \text{then list}('ilessp, \\ &\quad \quad \quad \text{iplus-tree-no-fix-int}(\text{bagdiff}(\text{iplus-fringe}(\text{cadr}(x)), \\ &\quad \quad \quad \quad \text{bagint}(\text{iplus-fringe}(\text{cadr}(x)), \\ &\quad \quad \quad \quad \text{iplus-fringe}(\text{caddr}(x))))), \end{aligned}$$


```

                                iplus-tree-no-fix-int (bagdiff (iplus-fringe (caddr (x)),
                                                                bagint (iplus-fringe (cadr (x)),
                                                                iplus-fringe (caddr (x))))))
                                else x endif
                                else x endif
                                else x endif

```

EVENT: Enable make-cancel-iplus-inequality-1; name this event ‘make-cancel-iplus-inequality-1-off’.

THEOREM: correctness-of-cancel-iplus-ilessp
 $\text{eval}\$ (\mathbf{t}, x, a) = \text{eval}\$ (\mathbf{t}, \text{cancel-iplus-ilessp}(x), a)$

THEOREM: itimes-zero1
 $(\text{fix-int}(x) = 0) \rightarrow (\text{itimes}(x, y) = 0)$

THEOREM: itimes-0-left
 $\text{itimes}(0, y) = 0$

EVENT: Enable itimes-zero1; name this event ‘itimes-zero1-off’.

THEOREM: itimes-zero2
 $(\text{fix-int}(y) = 0) \rightarrow (\text{itimes}(x, y) = 0)$

THEOREM: itimes-0-right
 $\text{itimes}(x, 0) = 0$

EVENT: Enable itimes-zero2; name this event ‘itimes-zero2-off’.

THEOREM: itimes-fix-int1
 $\text{itimes}(\text{fix-int}(a), b) = \text{itimes}(a, b)$

THEOREM: itimes-fix-int2
 $\text{itimes}(a, \text{fix-int}(b)) = \text{itimes}(a, b)$

THEOREM: commutativity-of-itimes
 $\text{itimes}(x, y) = \text{itimes}(y, x)$

THEOREM: itimes-distributes-over-iplus-proof
 $\text{itimes}(x, \text{iplus}(y, z)) = \text{iplus}(\text{itimes}(x, y), \text{itimes}(x, z))$

THEOREM: itimes-distributes-over-iplus
 $(\text{itimes}(x, \text{iplus}(y, z)) = \text{iplus}(\text{itimes}(x, y), \text{itimes}(x, z)))$
 $\wedge (\text{itimes}(\text{iplus}(x, y), z) = \text{iplus}(\text{itimes}(x, z), \text{itimes}(y, z)))$

THEOREM: commutativity2-of-itimes

$$\text{itimes}(x, \text{itimes}(y, z)) = \text{itimes}(y, \text{itimes}(x, z))$$

THEOREM: associativity-of-itimes

$$\text{itimes}(\text{itimes}(x, y), z) = \text{itimes}(x, \text{itimes}(y, z))$$

THEOREM: equal-itimes-0

$$(\text{itimes}(x, y) = 0) = ((\text{fix-int}(x) = 0) \vee (\text{fix-int}(y) = 0))$$

THEOREM: equal-itimes-1

$$\begin{aligned} &(\text{itimes}(a, b) = 1) \\ &= (((a = 1) \wedge (b = 1)) \vee ((a = -1) \wedge (b = -1))) \end{aligned}$$

THEOREM: equal-itimes-minus-1

$$\begin{aligned} &(\text{itimes}(a, b) = -1) \\ &= (((a = -1) \wedge (b = 1)) \vee ((a = 1) \wedge (b = -1))) \end{aligned}$$

THEOREM: itimes-1-arg1

$$\text{itimes}(1, x) = \text{fix-int}(x)$$

THEOREM: quotient-remainder-uniqueness

$$\begin{aligned} &((a = (r + (b * q))) \wedge (r < b)) \\ &\rightarrow ((\text{fix}(r) = (a \mathbf{mod} b)) \wedge (\text{fix}(q) = (a \div b))) \end{aligned}$$

THEOREM: division-theorem-part1

$$\text{integerp}(i) \rightarrow (\text{iplus}(\text{iremainder}(i, j), \text{itimes}(j, \text{iquotient}(i, j))) = i)$$

THEOREM: division-theorem-part2

$$(\text{integerp}(j) \wedge (j \neq 0)) \rightarrow (\neg \text{ilessp}(\text{iremainder}(i, j), 0))$$

THEOREM: division-theorem-part3

$$(\text{integerp}(j) \wedge (j \neq 0)) \rightarrow \text{ilessp}(\text{iremainder}(i, j), \text{iabs}(j))$$

THEOREM: division-theorem

$$\begin{aligned} &(\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0)) \\ &\rightarrow ((\text{iplus}(\text{iremainder}(i, j), \text{itimes}(j, \text{iquotient}(i, j))) = i) \\ &\quad \wedge (\neg \text{ilessp}(\text{iremainder}(i, j), 0)) \\ &\quad \wedge \text{ilessp}(\text{iremainder}(i, j), \text{iabs}(j))) \end{aligned}$$

THEOREM: quotient-difference-lessp-arg2

$$\begin{aligned} &(((a \mathbf{mod} c) = 0) \wedge (b < c)) \\ &\rightarrow (((a - b) \div c) \\ &\quad = \mathbf{if} \ b \simeq 0 \ \mathbf{then} \ a \div c \\ &\quad \mathbf{elseif} \ b < a \ \mathbf{then} \ (a \div c) - (1 + (b \div c)) \\ &\quad \mathbf{else} \ 0 \ \mathbf{endif}) \end{aligned}$$

THEOREM: iquotient-iremainder-uniqueness

```

(integerp (i)
  ∧ integerp (j)
  ∧ integerp (r)
  ∧ integerp (q)
  ∧ (j ≠ 0)
  ∧ (i = iplus (r, itimes (j, q)))
  ∧ (¬ illessp (r, 0))
  ∧ illessp (r, iabs (j)))
→ ((r = iremainder (i, j)) ∧ (q = iquotient (i, j)))

```

THEOREM: division-theorem-for-truncate-to-neginf-part1

```

(integerp (i) → (iplus (imod (i, j), itimes (j, idiv (i, j))) = i)

```

THEOREM: division-theorem-for-truncate-to-neginf-part2

```

illessp (0, j) → ((¬ illessp (imod (i, j), 0)) ∧ illessp (imod (i, j), j))

```

THEOREM: division-theorem-for-truncate-to-neginf-part3

```

(integerp (j) ∧ illessp (j, 0))
→ ((¬ illessp (0, imod (i, j))) ∧ illessp (j, imod (i, j)))

```

THEOREM: division-theorem-for-truncate-to-neginf

```

(integerp (i) ∧ integerp (j) ∧ (j ≠ 0))
→ ((iplus (imod (i, j), itimes (j, idiv (i, j))) = i)
  ∧ if illessp (0, j)
    then (¬ illessp (imod (i, j), 0)) ∧ illessp (imod (i, j), j)
    else (¬ illessp (0, imod (i, j)))
    ∧ illessp (j, imod (i, j)) endif)

```

THEOREM: idiv-imod-uniqueness

```

(integerp (i)
  ∧ integerp (j)
  ∧ integerp (r)
  ∧ integerp (q)
  ∧ (j ≠ 0)
  ∧ (i = iplus (r, itimes (j, q)))
  ∧ if illessp (0, j) then (¬ illessp (r, 0)) ∧ illessp (r, j)
  else (¬ illessp (0, r)) ∧ illessp (j, r) endif)
→ ((r = imod (i, j)) ∧ (q = idiv (i, j)))

```

THEOREM: division-theorem-for-truncate-to-zero-part1

```

(integerp (i) → (iplus (irem (i, j), itimes (j, iquo (i, j))) = i)

```

THEOREM: division-theorem-for-truncate-to-zero-part2

```

(integerp (i) ∧ integerp (j) ∧ (j ≠ 0) ∧ (¬ illessp (i, 0)))
→ ((¬ illessp (irem (i, j), 0)) ∧ illessp (irem (i, j), iabs (j)))

```

THEOREM: division-theorem-for-truncate-to-zero-part3
 $(\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0) \wedge \text{ilessp}(i, 0))$
 $\rightarrow ((\neg \text{ilessp}(0, \text{irem}(i, j))) \wedge \text{ilessp}(\text{ineg}(\text{iabs}(j)), \text{irem}(i, j)))$

THEOREM: division-theorem-for-truncate-to-zero
 $(\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0))$
 $\rightarrow ((\text{ipplus}(\text{irem}(i, j), \text{itimes}(j, \text{iquo}(i, j))) = i)$
 $\wedge \text{if } \neg \text{ilessp}(i, 0)$
 $\text{ then } (\neg \text{ilessp}(\text{irem}(i, j), 0))$
 $\wedge \text{ilessp}(\text{irem}(i, j), \text{iabs}(j))$
 $\text{ else } (\neg \text{ilessp}(0, \text{irem}(i, j)))$
 $\wedge \text{ilessp}(\text{ineg}(\text{iabs}(j)), \text{irem}(i, j)) \text{ endif})$

THEOREM: iquo-irem-uniqueness
 $(\text{integerp}(i)$
 $\wedge \text{integerp}(j)$
 $\wedge \text{integerp}(r)$
 $\wedge \text{integerp}(q)$
 $\wedge (j \neq 0)$
 $\wedge (i = \text{ipplus}(r, \text{itimes}(j, q)))$
 $\wedge \text{if } \neg \text{ilessp}(i, 0) \text{ then } (\neg \text{ilessp}(r, 0)) \wedge \text{ilessp}(r, \text{iabs}(j))$
 $\text{ else } (\neg \text{ilessp}(0, r)) \wedge \text{ilessp}(\text{ineg}(\text{iabs}(j)), r) \text{ endif})$
 $\rightarrow ((r = \text{irem}(i, j)) \wedge (q = \text{iquo}(i, j)))$

THEOREM: itimes-ineg-1
 $\text{itimes}(\text{ineg}(x), y) = \text{ineg}(\text{itimes}(x, y))$

THEOREM: itimes-ineg-2
 $\text{itimes}(x, \text{ineg}(y)) = \text{ineg}(\text{itimes}(x, y))$

THEOREM: itimes-cancellation-1
 $(\text{itimes}(a, b) = \text{itimes}(a, c))$
 $= ((\text{fix-int}(a) = 0) \vee (\text{fix-int}(b) = \text{fix-int}(c)))$

THEOREM: itimes-cancellation-2
 $(\text{itimes}(b, a) = \text{itimes}(c, a))$
 $= ((\text{fix-int}(a) = 0) \vee (\text{fix-int}(b) = \text{fix-int}(c)))$

THEOREM: itimes-cancellation-3
 $(\text{itimes}(a, b) = \text{itimes}(c, a))$
 $= ((\text{fix-int}(a) = 0) \vee (\text{fix-int}(b) = \text{fix-int}(c)))$

THEOREM: integerp-iquotient
 $\text{integerp}(\text{iquotient}(i, j))$

THEOREM: integerp-iremainder
 $\text{integerp}(\text{iremainder}(i, j))$

THEOREM: integerp-idiv
 $\text{integerp}(\text{idiv}(i, j))$

THEOREM: integerp-imod
 $\text{integerp}(\text{imod}(i, j))$

THEOREM: integerp-iquo
 $\text{integerp}(\text{iquo}(i, j))$

THEOREM: integerp-irem
 $\text{integerp}(\text{irem}(i, j))$

THEOREM: iquotient-fix-int1
 $\text{iquotient}(\text{fix-int}(i), j) = \text{iquotient}(i, j)$

THEOREM: iquotient-fix-int2
 $\text{iquotient}(i, \text{fix-int}(j)) = \text{iquotient}(i, j)$

THEOREM: iremainder-fix-int1
 $\text{iremainder}(\text{fix-int}(i), j) = \text{iremainder}(i, j)$

THEOREM: iremainder-fix-int2
 $\text{iremainder}(i, \text{fix-int}(j)) = \text{iremainder}(i, j)$

THEOREM: idiv-fix-int1
 $\text{idiv}(\text{fix-int}(i), j) = \text{idiv}(i, j)$

THEOREM: idiv-fix-int2
 $\text{idiv}(i, \text{fix-int}(j)) = \text{idiv}(i, j)$

THEOREM: imod-fix-int1
 $\text{imod}(\text{fix-int}(i), j) = \text{imod}(i, j)$

THEOREM: imod-fix-int2
 $\text{imod}(i, \text{fix-int}(j)) = \text{imod}(i, j)$

THEOREM: iquo-fix-int1
 $\text{iquo}(\text{fix-int}(i), j) = \text{iquo}(i, j)$

THEOREM: iquo-fix-int2
 $\text{iquo}(i, \text{fix-int}(j)) = \text{iquo}(i, j)$

THEOREM: irem-fix-int1
 $\text{irem}(\text{fix-int}(i), j) = \text{irem}(i, j)$

THEOREM: irem-fix-int2
 $\text{irem}(i, \text{fix-int}(j)) = \text{irem}(i, j)$

THEOREM: fix-int-iquotient
 $\text{fix-int}(\text{iquotient}(i, j)) = \text{iquotient}(i, j)$

THEOREM: fix-int-iremainder
 $\text{fix-int}(\text{iremainder}(i, j)) = \text{iremainder}(i, j)$

THEOREM: fix-int-idiv
 $\text{fix-int}(\text{idiv}(i, j)) = \text{idiv}(i, j)$

THEOREM: fix-int-imod
 $\text{fix-int}(\text{imod}(i, j)) = \text{imod}(i, j)$

THEOREM: fix-int-iquo
 $\text{fix-int}(\text{iquo}(i, j)) = \text{iquo}(i, j)$

THEOREM: fix-int-irem
 $\text{fix-int}(\text{irem}(i, j)) = \text{irem}(i, j)$

EVENT: Enable iquotient; name this event ‘iquotient-off’.

EVENT: Enable iremainder; name this event ‘iremainder-off’.

EVENT: Enable idiv; name this event ‘idiv-off’.

EVENT: Enable imod; name this event ‘imod-off’.

EVENT: Enable iquo; name this event ‘iquo-off’.

EVENT: Enable irem; name this event ‘irem-off’.

DEFINITION:
 $\text{itimes-fringe}(x)$
 $=$ **if** $\text{listp}(x)$
 then if $\text{car}(x) = \text{'itimes}$
 then $\text{append}(\text{itimes-fringe}(\text{cadr}(x)), \text{itimes-fringe}(\text{caddr}(x)))$
 else $\text{list}(x)$ **endif**
 else $\text{list}(x)$ **endif**

DEFINITION:
 $\text{itimes-tree-rec}(l)$
 $=$ **if** $\text{cdr}(l) \simeq \text{nil}$ **then** $\text{car}(l)$
 else $\text{list}(\text{'itimes}, \text{car}(l), \text{itimes-tree-rec}(\text{cdr}(l)))$ **endif**

DEFINITION:

```
itimes-tree (l)
=  if listp (l)
    then if listp (cdr (l)) then itimes-tree-rec (l)
        else list ('fix-int, car (l)) endif
    else ''1 endif
```

DEFINITION:

```
itimes-list (x)
=  if listp (x) then itimes (car (x), itimes-list (cdr (x)))
    else 1 endif
```

THEOREM: integerp-itimes-list

integerp (itimes-list (x))

THEOREM: eval\$-itimes-tree-rec

```
listp (x)
→ (eval$ (t, itimes-tree-rec (x), a)
   =  if listp (cdr (x)) then itimes-list (eval$ ('list, x, a))
       else eval$ (t, car (x), a) endif)
```

THEOREM: eval\$-itimes-tree

eval\$ (t, itimes-tree (x), a) = itimes-list (eval\$ ('list, x, a))

EVENT: Enable itimes-tree; name this event 'itimes-tree-off'.

DEFINITION:

```
make-cancel-itimes-equality (x, y, in-both)
=  list ('if,
        cons ('equal, cons (itimes-tree (in-both), ' ('0))),
        list ('quote, t),
        list ('equal,
              itimes-tree (bagdiff (x, in-both)),
              itimes-tree (bagdiff (y, in-both))))
```

DEFINITION:

```
cancel-itimes (x)
=  if listp (x)
    then if car (x) = 'equal
        then if listp (cadr (x))
            then if caadr (x) = 'itimes
                then if listp (caddr (x))
                    then if caaddr (x) = 'itimes
                        then if listp (bagint (itimes-fringe (cadr (x)),
                                                itimes-fringe (caddr (x))))
```

```

        then make-cancel-itimes-equality (itimes-fringe (cadr (x)),
                                           itimes-fringe (caddr (x)),
                                           bagint (itimes-fringe (cadr (x)),
                                                    itimes-fringe (caddr (x))))

      else x endif
    elseif caddr (x) ∈ itimes-fringe (cadr (x))
    then list ('if,
              list ('integerp, caddr (x)),
              make-cancel-itimes-equality (itimes-fringe (cadr (x)),
                                           list (caddr (x)),
                                           list (caddr (x))),

              list ('quote, f))
    else x endif
  elseif caddr (x) ∈ itimes-fringe (cadr (x))
  then list ('if,
            list ('integerp, caddr (x)),
            make-cancel-itimes-equality (itimes-fringe (cadr (x)),
                                         list (caddr (x)),
                                         list (caddr (x))),

            list ('quote, f))
  else x endif
elseif listp (caddr (x))
then if caaddr (x) = 'itimes
    then if cadr (x) ∈ itimes-fringe (caddr (x))
        then list ('if,
                  list ('integerp, cadr (x)),
                  make-cancel-itimes-equality (list (cadr (x)),
                                               itimes-fringe (caddr (x)),
                                               list (cadr (x))),

                  list ('quote, f))
        else x endif
    else x endif
else x endif
elseif listp (caddr (x))
then if caaddr (x) = 'itimes
    then if cadr (x) ∈ itimes-fringe (caddr (x))
        then list ('if,
                  list ('integerp, cadr (x)),
                  make-cancel-itimes-equality (list (cadr (x)),
                                               itimes-fringe (caddr (x)),
                                               list (cadr (x))),

                  list ('quote, f))
        else x endif
    else x endif

```


else x endif
else x endif
else x endif

THEOREM: itimes-list-append

$$\text{itimes-list}(\text{append}(x, y)) = \text{itimes}(\text{itimes-list}(x), \text{itimes-list}(y))$$

THEOREM: itimes-list-eval\$-fringe

$$\text{itimes-list}(\text{eval\$}('list, \text{itimes-fringe}(x), a)) = \text{fix-int}(\text{eval\$}(\mathbf{t}, x, a))$$

THEOREM: integerp-eval\$-itimes

$$(\text{car}(x) = \text{'itimes}) \rightarrow \text{integerp}(\text{eval\$}(\mathbf{t}, x, a))$$

THEOREM: not-integerp-implies-not-equal-itimes

$$(\neg \text{integerp}(a)) \rightarrow ((a = \text{itimes}(b, c)) = \mathbf{f})$$

THEOREM: itimes-list-eval\$-delete

$$\begin{aligned}
&(z \in y) \\
&\rightarrow (\text{itimes-list}(\text{eval\$}('list, y, a)) \\
&\quad = \text{itimes}(\text{eval\$}(\mathbf{t}, z, a), \text{itimes-list}(\text{eval\$}('list, \text{delete}(z, y), a))))
\end{aligned}$$

THEOREM: itimes-list-bagdiff

$$\begin{aligned}
&\text{subbagp}(x, y) \\
&\rightarrow (\text{itimes-list}(\text{eval\$}('list, y, a)) \\
&\quad = \text{itimes}(\text{itimes-list}(\text{eval\$}('list, \text{bagdiff}(y, x), a)), \\
&\quad \quad \text{itimes-list}(\text{eval\$}('list, x, a))))
\end{aligned}$$

THEOREM: equal-itimes-list-eval\$-list-delete

$$\begin{aligned}
&((c \in y) \wedge (\text{fix-int}(\text{eval\$}(\mathbf{t}, c, a)) \neq 0)) \\
&\rightarrow ((x = \text{itimes-list}(\text{eval\$}('list, \text{delete}(c, y), a))) \\
&\quad = (\text{integerp}(x) \\
&\quad \quad \wedge (\text{itimes}(x, \text{eval\$}(\mathbf{t}, c, a)) \\
&\quad \quad = \text{itimes-list}(\text{eval\$}('list, y, a))))))
\end{aligned}$$

EVENT: Enable itimes-list-eval\$-delete; name this event 'itimes-list-eval\$-delete-off'.

THEOREM: member-append

$$(a \in \text{append}(x, y)) = ((a \in x) \vee (a \in y))$$

THEOREM: member-izerop-itimes-fringe

$$\begin{aligned}
&((z \in \text{itimes-fringe}(x)) \wedge (\text{fix-int}(\text{eval\$}(\mathbf{t}, z, a)) = 0)) \\
&\rightarrow (\text{fix-int}(\text{eval\$}(\mathbf{t}, x, a)) = 0)
\end{aligned}$$

THEOREM: correctness-of-cancel-itimes-hack-1

```
((w ∈ itimes-fringe (cons ('itimes, x1)))
 ∧ (fix-int (eval$ (t, w, a)) = 0)
 ∧ (fix-int (eval$ (t, car (x1), a)) ≠ 0))
 → (fix-int (eval$ (t, cadr (x1), a)) = 0)
```

EVENT: Disable eval\$-equal; name this event 'eval\$-equal-on'.

THEOREM: eval\$-make-cancel-itimes-equality

```
eval$ (t, make-cancel-itimes-equality (x, y, in-both), a)
 = if eval$ (t, cons ('equal, cons (itimes-tree (in-both), ' ('0))), a)
   then t
   else itimes-list (eval$ ('list, bagdiff (x, in-both), a))
     = itimes-list (eval$ ('list, bagdiff (y, in-both), a)) endif
```

EVENT: Enable make-cancel-itimes-equality; name this event 'make-cancel-itimes-equality-off'.

THEOREM: eval\$-make-cancel-itimes-equality-1

```
eval$ (t, make-cancel-itimes-equality (list (x), y, list (x)), a)
 = if fix-int (eval$ (t, x, a)) = 0 then t
   else 1 = itimes-list (eval$ ('list, delete (x, y), a)) endif
```

THEOREM: equal-fix-int

```
(fix-int (x) = x) = integerp (x)
```

THEOREM: eval\$-make-cancel-itimes-equality-2

```
eval$ (t, make-cancel-itimes-equality (x, list (y), list (y)), a)
 = if fix-int (eval$ (t, y, a)) = 0 then t
   else 1 = itimes-list (eval$ ('list, delete (y, x), a)) endif
```

THEOREM: eval\$-equal-itimes-tree-itimes-fringe-0

```
(eval$ (t, cons ('equal, cons (itimes-tree (itimes-fringe (x)), ' ('0))), a)
 ∧ (car (x) = 'itimes))
 → (eval$ (t, x, a) = 0)
```

THEOREM: izerop-eval-of-member-implies-itimes-list-0

```
((z ∈ y) ∧ (fix-int (eval$ (t, z, a)) = 0))
 → (itimes-list (eval$ ('list, y, a)) = 0)
```

DEFINITION:

subsetp (x, y)

```
= if x ≈ nil then t
   elseif car (x) ∈ y then subsetp (cdr (x), y)
   else f endif
```

THEOREM: subsetp-implies-itimes-list-eval\$-equals-0
 $(\text{subsetp}(x, y) \wedge (\text{itimes-list}(\text{eval}(\$('list, x, a)) = 0))$
 $\rightarrow (\text{itimes-list}(\text{eval}(\$('list, y, a)) = 0)$

THEOREM: subbagp-subsetp
 $\text{subbagp}(x, y) \rightarrow \text{subsetp}(x, y)$

THEOREM: equal-0-itimes-list-eval\$-bagint-1
 $(\text{itimes-list}(\text{eval}(\$('list, \text{bagint}(x, y), a)) = 0)$
 $\rightarrow (\text{itimes-list}(\text{eval}(\$('list, x, a)) = 0)$

THEOREM: equal-0-itimes-list-eval\$-bagint-2
 $(\text{itimes-list}(\text{eval}(\$('list, \text{bagint}(x, y), a)) = 0)$
 $\rightarrow (\text{itimes-list}(\text{eval}(\$('list, y, a)) = 0)$

THEOREM: correctness-of-cancel-itimes-hack-2
 $(\text{listp}(u)$
 $\wedge (\text{car}(u) = 'itimes)$
 $\wedge \text{listp}(v)$
 $\wedge (\text{car}(v) = 'itimes)$
 $\wedge (\text{eval}(\$ (t, u, a) \neq \text{eval}(\$ (t, v, a)))$
 $\rightarrow (\text{itimes-list}(\text{eval}(\$('list,$
 $\quad \text{bagdiff}(\text{itimes-fringe}(u),$
 $\quad \text{bagint}(\text{itimes-fringe}(u), \text{itimes-fringe}(v))),$
 $\quad a))$
 $\neq \text{itimes-list}(\text{eval}(\$('list,$
 $\quad \text{bagdiff}(\text{itimes-fringe}(v),$
 $\quad \text{bagint}(\text{itimes-fringe}(u),$
 $\quad \text{itimes-fringe}(v))),$
 $\quad a)))$

THEOREM: correctness-of-cancel-itimes-hack-3-lemma
 $((u = \text{itimes}(a, b)) \wedge (\text{fix-int}(a) \neq 0))$
 $\rightarrow ((u = \text{itimes}(a, c)) = (\text{fix-int}(b) = \text{fix-int}(c)))$

THEOREM: correctness-of-cancel-itimes-hack-3
 $(\text{listp}(u)$
 $\wedge (\text{car}(u) = 'itimes)$
 $\wedge \text{listp}(v)$
 $\wedge (\text{car}(v) = 'itimes)$
 $\wedge (\text{eval}(\$ (t, u, a) = \text{eval}(\$ (t, v, a)))$
 $\wedge (\neg \text{eval}(\$ (t,$
 $\quad \text{cons}('equal,$
 $\quad \text{cons}(\text{itimes-tree}(\text{bagint}(\text{itimes-fringe}(u),$
 $\quad \text{itimes-fringe}(v))),$

$$\begin{aligned}
& \quad \quad \quad ' (' 0))), \\
& \quad \quad \quad a))) \\
\rightarrow & \quad ((\text{itimes-list} (\text{eval\$} (' \text{list}, \\
& \quad \quad \quad \text{bagdiff} (\text{itimes-fringe} (u), \\
& \quad \quad \quad \text{bagint} (\text{itimes-fringe} (u), \text{itimes-fringe} (v))), \\
& \quad \quad \quad a)) \\
& = \quad \text{itimes-list} (\text{eval\$} (' \text{list}, \\
& \quad \quad \quad \text{bagdiff} (\text{itimes-fringe} (v), \\
& \quad \quad \quad \text{bagint} (\text{itimes-fringe} (u), \\
& \quad \quad \quad \text{itimes-fringe} (v))), \\
& \quad \quad \quad a))) \\
& = \quad \mathbf{t})
\end{aligned}$$

EVENT: Enable correctness-of-cancel-itimes-hack-3-lemma; name this event 'correctness-of-cancel-itimes-hack-3-lemma-off'.

THEOREM: correctness-of-cancel-itimes
 $\text{eval\$} (\mathbf{t}, x, a) = \text{eval\$} (\mathbf{t}, \text{cancel-itimes} (x), a)$

DEFINITION:
 $\text{itimes-tree-no-fix-int} (l)$
 $= \text{if listp} (l) \text{ then } \text{itimes-tree-rec} (l)$
 $\text{else } ' ' 1 \text{ endif}$

THEOREM: eval\$-itimes-tree-no-fix-int-1
 $\text{ilessp} (\text{eval\$} (\mathbf{t}, \text{itimes-tree-no-fix-int} (x), a), y)$
 $= \text{ilessp} (\text{eval\$} (\mathbf{t}, \text{itimes-tree} (x), a), y)$

THEOREM: eval\$-itimes-tree-no-fix-int-2
 $\text{ilessp} (y, \text{eval\$} (\mathbf{t}, \text{itimes-tree-no-fix-int} (x), a))$
 $= \text{ilessp} (y, \text{eval\$} (\mathbf{t}, \text{itimes-tree} (x), a))$

EVENT: Enable itimes-tree-no-fix-int; name this event 'itimes-tree-no-fix-int-off'.

DEFINITION:
 $\text{make-cancel-itimes-inequality} (x, y, \text{in-both})$
 $= \text{list} (' \text{if},$
 $\quad \text{cons} (' \text{ilessp}, \text{cons} (\text{itimes-tree-no-fix-int} (\text{in-both}), ' (' 0))),$
 $\quad \text{list} (' \text{ilessp},$
 $\quad \quad \text{itimes-tree-no-fix-int} (\text{bagdiff} (y, \text{in-both})),$
 $\quad \quad \text{itimes-tree-no-fix-int} (\text{bagdiff} (x, \text{in-both})),$
 $\quad \text{cons} (' \text{if},$
 $\quad \quad \text{cons} (\text{list} (' \text{ilessp}, ' ' 0, \text{itimes-tree-no-fix-int} (\text{in-both})),$

```

cons (list ('ilessp,
           itimes-tree-no-fix-int (bagdiff (x, in-both)),
           itimes-tree-no-fix-int (bagdiff (y, in-both))),
      ' (false))))))

```

DEFINITION:

cancel-itimes-ilessp (x)

```

=  if listp (x)
    then if car (x) = 'ilessp
          then if listp (bagint (itimes-fringe (cadr (x)),
                                itimes-fringe (caddr (x))))
                then make-cancel-itimes-inequality (itimes-fringe (cadr (x)),
                                                            itimes-fringe (caddr (x)),
                                                            bagint (itimes-fringe (cadr (x)),
                                                                itimes-fringe (caddr (x))))
                else x
          else x
    else x

```

THEOREM: eval\$-make-cancel-itimes-inequality

eval\$ (t, make-cancel-itimes-inequality (x , y , *in-both*), a)

```

=  if eval$ (t,
            cons ('ilessp, cons (itimes-tree-no-fix-int (in-both), ' (0))),
    a)
    then illessp (eval$ (t, itimes-tree-no-fix-int (bagdiff (y, in-both)), a),
                  eval$ (t, itimes-tree-no-fix-int (bagdiff (x, in-both)), a))
    elseif eval$ (t, list ('ilessp, ' 0, itimes-tree-no-fix-int (in-both)), a)
    then illessp (eval$ (t, itimes-tree-no-fix-int (bagdiff (x, in-both)), a),
                  eval$ (t, itimes-tree-no-fix-int (bagdiff (y, in-both)), a))
    else f

```

EVENT: Enable make-cancel-itimes-inequality; name this event 'make-cancel-itimes-inequality-off'.

THEOREM: listp-bagint-with-singleton-implies-member

listp (bagint (y , list (z))) $\rightarrow (z \in y)$

THEOREM: itimes-list-eval\$-list-0

$(0 \in x) \rightarrow (\text{itimes-list} (\text{eval\$} ('list, x, a)) = 0)$

THEOREM: illessp-itimes-right-positive

illessp ($0, x$) $\rightarrow (\text{illessp} (y, z) = \text{illessp} (\text{itimes} (y, x), \text{itimes} (z, x)))$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-1

(subbagp (bag, x))

$$\begin{aligned}
& \wedge \text{subbagp}(bag, y) \\
& \wedge \text{ilessp}(0, \text{itimes-list}(\text{eval}\$('list, bag, a)))) \\
\rightarrow & (\text{ilessp}(\text{itimes-list}(\text{eval}\$('list, \text{bagdiff}(x, bag), a)), \\
& \quad \text{itimes-list}(\text{eval}\$('list, \text{bagdiff}(y, bag), a)))) \\
& = \text{ilessp}(\text{itimes-list}(\text{eval}\$('list, x, a)), \\
& \quad \text{itimes-list}(\text{eval}\$('list, y, a))))
\end{aligned}$$

THEOREM: listp-bagint-with-singleton-member
 $\text{listp}(\text{bagint}(y, \text{list}(z))) = (z \in y)$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-2-lemma
 $(0 \in \text{itimes-fringe}(w)) \rightarrow (\text{eval}\$(t, w, a) = 0)$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-2
 $(0 \in \text{itimes-fringe}(w)) \rightarrow (\neg \text{ilessp}(\text{eval}\$(t, w, a), 0))$

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-2-lemma; name this event
 ‘correctness-of-cancel-itimes-ilessp-hack-2-lemma-off’.

THEOREM: illessp-trichotomy
 $(\neg \text{ilessp}(x, y)) \rightarrow (\text{ilessp}(y, x) = (\text{fix-int}(x) \neq \text{fix-int}(y)))$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-3-lemma-1
 $((0 = \text{itimes-list}(\text{eval}\$('list, bag, a))) \wedge \text{subsetp}(bag, z))$
 $\rightarrow (\text{itimes-list}(\text{eval}\$('list, z, a)) = 0)$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-3-lemma-2
 $((0 = \text{itimes-list}(\text{eval}\$('list, bag, a))) \wedge \text{subsetp}(bag, \text{itimes-fringe}(x)))$
 $\rightarrow (\text{fix-int}(\text{eval}\$(t, x, a)) = 0)$

THEOREM: same-fix-int-implies-not-ilessp
 $(\text{fix-int}(x) = \text{fix-int}(y)) \rightarrow (\neg \text{ilessp}(x, y))$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-3
 $((\neg \text{ilessp}(\text{itimes-list}(\text{eval}\$('list, bag, a)), 0))$
 $\wedge (\neg \text{ilessp}(0, \text{itimes-list}(\text{eval}\$('list, bag, a))))$
 $\wedge \text{subbagp}(bag, \text{itimes-fringe}(w))$
 $\wedge \text{subbagp}(bag, \text{itimes-fringe}(v)))$
 $\rightarrow (\neg \text{ilessp}(\text{eval}\$(t, w, a), \text{eval}\$(t, v, a)))$

THEOREM: illessp-itimes-right-negative
 $\text{ilessp}(x, 0) \rightarrow (\text{ilessp}(y, z) = \text{ilessp}(\text{itimes}(z, x), \text{itimes}(y, x)))$

THEOREM: correctness-of-cancel-itimes-ilessp-hack-4
 $(\text{subbagp}(bag, x)$

$$\begin{aligned}
& \wedge \text{subbagp}(bag, y) \\
& \wedge \text{ilessp}(\text{itimes-list}(\text{eval\$}('list, bag, a)), 0)) \\
\rightarrow & (\text{ilessp}(\text{itimes-list}(\text{eval\$}('list, \text{bagdiff}(x, bag), a)), \\
& \quad \text{itimes-list}(\text{eval\$}('list, \text{bagdiff}(y, bag), a))) \\
& = \text{ilessp}(\text{itimes-list}(\text{eval\$}('list, y, a)), \\
& \quad \text{itimes-list}(\text{eval\$}('list, x, a))))
\end{aligned}$$

EVENT: Enable `ilessp-trichotomy`; name this event ‘`ilessp-trichotomy-off`’.

EVENT: Enable `same-fix-int-implies-not-ilessp`; name this event ‘`same-fix-int-implies-not-ilessp-off`’.

THEOREM: `correctness-of-cancel-itimes-ilessp`
 $\text{eval\$}(\mathbf{t}, x, a) = \text{eval\$}(\mathbf{t}, \text{cancel-itimes-ilessp}(x), a)$

THEOREM: `ilessp-strict`
 $\text{ilessp}(x, y) \rightarrow (\neg \text{ilessp}(y, x))$

EVENT: Enable `eval$-cancel-iplus`; name this event ‘`eval$-cancel-iplus-off`’.

EVENT: Enable `eval$-iplus`; name this event ‘`eval$-iplus-off`’.

EVENT: Enable `lessp-count-listp-cdr`; name this event ‘`lessp-count-listp-cdr-off`’.

EVENT: Enable `eval$-iplus-tree-rec`; name this event ‘`eval$-iplus-tree-rec-off`’.

EVENT: Enable `eval$-iplus-tree`; name this event ‘`eval$-iplus-tree-off`’.

EVENT: Enable `iplus-list-eval$-fringe`; name this event ‘`iplus-list-eval$-fringe-off`’.

EVENT: Enable `eval$-iplus-list-bagdiff`; name this event ‘`eval$-iplus-list-bagdiff-off`’.

EVENT: Enable `lessp-difference-plus-arg1`; name this event ‘`lessp-difference-plus-arg1-off`’.

EVENT: Enable `lessp-difference-plus-arg1-commuted`; name this event ‘`lessp-difference-plus-arg1-commuted-off`’.

EVENT: Enable correctness-of-cancel-iplus-ilessp-lemma; name this event ‘correctness-of-cancel-iplus-ilessp-lemma-off’.

EVENT: Enable eval\$-ilessp-iplus-tree-no-fix-int; name this event ‘eval\$-ilessp-iplus-tree-no-fix-int-off’.

EVENT: Enable make-cancel-iplus-inequality-simplifier; name this event ‘make-cancel-iplus-inequality-simplifier-off’.

EVENT: Enable quotient-difference-lessp-arg2; name this event ‘quotient-difference-lessp-arg2-off’.

EVENT: Enable eval\$-itimes-tree-rec; name this event ‘eval\$-itimes-tree-rec-off’.

EVENT: Enable eval\$-itimes-tree; name this event ‘eval\$-itimes-tree-off’.

EVENT: Enable itimes-list-eval\$-fringe; name this event ‘itimes-list-eval\$-fringe-off’.

EVENT: Enable integerp-eval\$-itimes; name this event ‘integerp-eval\$-itimes-off’.

EVENT: Enable itimes-list-bagdiff; name this event ‘itimes-list-bagdiff-off’.

EVENT: Enable equal-itimes-list-eval\$-list-delete; name this event ‘equal-itimes-list-eval\$-list-delete-off’.

EVENT: Enable member-izerop-itimes-fringe; name this event ‘member-izerop-itimes-fringe-off’.

EVENT: Enable correctness-of-cancel-itimes-hack-1; name this event ‘correctness-of-cancel-itimes-hack-1-off’.

EVENT: Enable eval\$-make-cancel-itimes-equality; name this event ‘eval\$-make-cancel-itimes-equality-off’.

EVENT: Enable eval\$-make-cancel-itimes-equality-1; name this event ‘eval\$-

make-cancel-itimes-equality-1-off’.

EVENT: Enable eval\$-make-cancel-itimes-equality-2; name this event ‘eval\$-make-cancel-itimes-equality-2-off’.

EVENT: Enable eval\$-equal-itimes-tree-itimes-fringe-0; name this event ‘eval\$-equal-itimes-tree-itimes-fringe-0-off’.

EVENT: Enable izerop-eval-of-member-implies-itimes-list-0; name this event ‘izerop-eval-of-member-implies-itimes-list-0-off’.

EVENT: Enable subsetp-implies-itimes-list-eval\$-equals-0; name this event ‘subsetp-implies-itimes-list-eval\$-equals-0-off’.

EVENT: Enable equal-0-itimes-list-eval\$-bagint-1; name this event ‘equal-0-itimes-list-eval\$-bagint-1-off’.

EVENT: Enable equal-0-itimes-list-eval\$-bagint-2; name this event ‘equal-0-itimes-list-eval\$-bagint-2-off’.

EVENT: Enable correctness-of-cancel-itimes-hack-2; name this event ‘correctness-of-cancel-itimes-hack-2-off’.

EVENT: Enable correctness-of-cancel-itimes-hack-3-lemma; name this event ‘correctness-of-cancel-itimes-hack-3-lemma-off1’.

EVENT: Enable correctness-of-cancel-itimes-hack-3; name this event ‘correctness-of-cancel-itimes-hack-3-off’.

EVENT: Enable eval\$-itimes-tree-no-fix-int-1; name this event ‘eval\$-itimes-tree-no-fix-int-1-off’.

EVENT: Enable eval\$-itimes-tree-no-fix-int-2; name this event ‘eval\$-itimes-tree-no-fix-int-2-off’.

EVENT: Enable eval\$-make-cancel-itimes-inequality; name this event ‘eval\$-make-cancel-itimes-inequality-off’.

EVENT: Enable listp-bagint-with-singleton-implies-member; name this event 'listp-bagint-with-singleton-implies-member-off'.

EVENT: Enable itimes-list-eval\$list-0; name this event 'itimes-list-eval\$list-0-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-1; name this event 'correctness-of-cancel-itimes-ilessp-hack-1-off'.

EVENT: Enable listp-bagint-with-singleton-member; name this event 'listp-bagint-with-singleton-member-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-2; name this event 'correctness-of-cancel-itimes-ilessp-hack-2-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-3-lemma-1; name this event 'correctness-of-cancel-itimes-ilessp-hack-3-lemma-1-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-3-lemma-2; name this event 'correctness-of-cancel-itimes-ilessp-hack-3-lemma-2-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-3; name this event 'correctness-of-cancel-itimes-ilessp-hack-3-off'.

EVENT: Enable correctness-of-cancel-itimes-ilessp-hack-4; name this event 'correctness-of-cancel-itimes-ilessp-hack-4-off'.

DEFINITION:

itimes-tree-ineg(l)

```
=  if listp( $l$ )
    then if car( $l$ ) = '-1
          then if listp(cdr( $l$ )) then list('neg, itimes-tree-rec(cdr( $l$ )))
                else car( $l$ ) endif
          else itimes-tree-rec( $l$ ) endif
    else '1 endif
```

DEFINITION:

itimes-factors(x)

```
=  if listp( $x$ )
```

```

then case on car(x):
  case = itimes
  then append(itimes-factors(cadr(x)), itimes-factors(caddr(x)))
  case = iplus
  then if listp(bagint(itimes-factors(cadr(x)),
    itimes-factors(caddr(x))))
    then cons(list('iplus,
      itimes-tree-ineg(bagdiff(itimes-factors(cadr(x)),
        bagint(itimes-factors(cadr(x)),
          itimes-factors(caddr(x))))),
      itimes-tree-ineg(bagdiff(itimes-factors(caddr(x)),
        bagint(itimes-factors(cadr(x)),
          itimes-factors(caddr(x))))),
      bagint(itimes-factors(cadr(x)),
        itimes-factors(caddr(x))))))
    else list(x) endif
  case = ineg
  then cons(' -1, itimes-factors(cadr(x)))
  otherwise list(x) endcase
else list(x) endif

```

THEOREM: itimes-1
 $\text{itimes}(-1, x) = \text{ineg}(x)$

THEOREM: equal-ineg-ineg
 $(\text{ineg}(x) = \text{ineg}(y)) = (\text{fix-int}(x) = \text{fix-int}(y))$

THEOREM: illessp-ineg-ineg
 $\text{illessp}(\text{ineg}(x), \text{ineg}(y)) = \text{illessp}(y, x)$

THEOREM: fix-int-eval\$-itimes-tree-rec
 $\text{listp}(x)$
 $\rightarrow (\text{fix-int}(\text{eval}(\mathbf{t}, \text{itimes-tree-rec}(x), a))$
 $\quad = \text{itimes-list}(\text{eval}(\mathbf{'list}, x, a)))$

THEOREM: eval\$-itimes-tree-ineg
 $\text{fix-int}(\text{eval}(\mathbf{t}, \text{itimes-tree-ineg}(x), a)) = \text{itimes-list}(\text{eval}(\mathbf{'list}, x, a))$

THEOREM: neg-eval\$-itimes-tree-ineg
 $\text{ineg}(\text{eval}(\mathbf{t}, \text{itimes-tree-ineg}(x), a))$
 $= \text{ineg}(\text{itimes-list}(\text{eval}(\mathbf{'list}, x, a)))$

THEOREM: iplus-eval\$-itimes-tree-ineg
 $(\text{iplus}(\text{eval}(\mathbf{t}, \text{itimes-tree-ineg}(x), a), y)$
 $\quad = \text{iplus}(\text{itimes-list}(\text{eval}(\mathbf{'list}, x, a)), y))$
 $\wedge (\text{iplus}(y, \text{eval}(\mathbf{t}, \text{itimes-tree-ineg}(x), a))$
 $\quad = \text{iplus}(y, \text{itimes-list}(\text{eval}(\mathbf{'list}, x, a))))$

THEOREM: itimes-eval\$-itimes-tree-ineg
 (itimes (eval\$ (t, itimes-tree-ineg (x), a), y)
 = itimes (itimes-list (eval\$ ('list, x, a)), y))
 $\wedge$ (itimes (y, eval\$ (t, itimes-tree-ineg (x), a))
 = itimes (y, itimes-list (eval\$ ('list, x, a))))

EVENT: Enable itimes-tree-ineg; name this event 'itimes-tree-ineg-off'.

DEFINITION:

iplus-or-itimes-term (x)
 = **if** listp (x)
 then case on car (x):
 case = *iphus*
 then t
 case = *itimes*
 then t
 case = *ineg*
 then if listp (cadr (x)) **then** caadr (x) = 'itimes
 else f endif
 otherwise f endcase
else f endif

DEFINITION:

cancel-itimes-factors (x)
 = **if** listp (x) $\wedge$ (car (x) = 'equal)
 then if iplus-or-itimes-term (cadr (x))
 then if listp (bagint (itimes-factors (cadr (x)),
 itimes-factors (caddr (x))))
 then if iplus-or-itimes-term (caddr (x))
 then make-cancel-itimes-equality (itimes-factors (cadr (x)),
 itimes-factors (caddr (x)),
 bagint (itimes-factors (cadr (x)),
 itimes-factors (caddr (x))))
 else list ('if,
 list ('integerp, caddr (x)),
 make-cancel-itimes-equality (itimes-factors (cadr (x)),
 itimes-factors (caddr (x)),
 bagint (itimes-factors (cadr (x)),
 itimes-factors (caddr (x))))),
 list ('quote, f)) **endif**
 else x endif
 elseif iplus-or-itimes-term (caddr (x))
 then if listp (bagint (itimes-factors (cadr (x)),
 itimes-factors (caddr (x))))

```

    then list ('if,
              list ('integerp, cadr (x)),
              make-cancel-itimes-equality (itimes-factors (cadr (x)),
                                           itimes-factors (caddr (x)),
                                           bagint (itimes-factors (cadr (x)),
                                                    itimes-factors (caddr (x)))),
              list ('quote, f))
    else x endif
  else x endif
else x endif

```

```

(DEFN CANCEL-ITIMES-FACTORS-EXPANDED
(X)
(IF
 (LISTP X)
 (IF
  (EQUAL (CAR X) 'EQUAL)
  (COND
   ((LISTP (CADR X))
    (CASE
     (CAADR X)
     (IPLUS
      (IF
       (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                     (ITIMES-FACTORS (CADDR X))))
       (IF
        (LISTP (CADDR X))
        (CASE
         (CAADDR X)
         (IPLUS
          (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                       (ITIMES-FACTORS (CADDR X))
                                       (BAGINT (ITIMES-FACTORS (CADR X))
                                              (ITIMES-FACTORS (CADDR X))))))
        (ITIMES
         (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                       (ITIMES-FACTORS (CADDR X))
                                       (BAGINT (ITIMES-FACTORS (CADR X))
                                              (ITIMES-FACTORS (CADDR X))))))
        (INEG
         (IF
          (LISTP (CADADDR X))

```

```

(IF
  (EQUAL (CAADADDR X) 'ITIMES)
    (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
      (ITIMES-FACTORS (CADDR X))
      (BAGINT (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))))
    (LIST 'IF
      (LIST 'INTEGERP (CADDR X))
      (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))
        (BAGINT (ITIMES-FACTORS (CADR X))
          (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F)))
  (LIST 'IF
    (LIST 'INTEGERP (CADDR X))
    (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
      (ITIMES-FACTORS (CADDR X))
      (BAGINT (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))))
    (LIST 'QUOTE F))))
(OTHERWISE
  (LIST 'IF
    (LIST 'INTEGERP (CADDR X))
    (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
      (ITIMES-FACTORS (CADDR X))
      (BAGINT (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))))
    (LIST 'QUOTE F))))
  (LIST 'IF
    (LIST 'INTEGERP (CADDR X))
    (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
      (ITIMES-FACTORS (CADDR X))
      (BAGINT (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))))
    (LIST 'QUOTE F))))
  X))
(ITIMES
  (IF
    (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
      (ITIMES-FACTORS (CADDR X))))
    (IF
      (LISTP (CADDR X))
      (CASE
        (CAADDR X)

```



```

                                                    (ITIMES-FACTORS (CADDR X))
                                                    (BAGINT (ITIMES-FACTORS (CADR X))
                                                     (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F)))
X))
(INEG
(COND
  ((LISTP (CADADR X))
   (COND
    ((EQUAL (CAADADR X) 'ITIMES)
     (IF
      (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                     (ITIMES-FACTORS (CADDR X))))
      (IF
       (LISTP (CADDR X))
       (CASE
        (CAADDR X)
        (IPLUS
         (MAKE-CANCEL-ITIMES-EQUALITY
          (ITIMES-FACTORS (CADR X))
          (ITIMES-FACTORS (CADDR X))
          (BAGINT (ITIMES-FACTORS (CADR X))
                   (ITIMES-FACTORS (CADDR X))))))
        (ITIMES
         (MAKE-CANCEL-ITIMES-EQUALITY
          (ITIMES-FACTORS (CADR X))
          (ITIMES-FACTORS (CADDR X))
          (BAGINT (ITIMES-FACTORS (CADR X))
                   (ITIMES-FACTORS (CADDR X))))))
      (INEG
       (IF
        (LISTP (CADADDR X))
        (IF
         (EQUAL (CAADADDR X) 'ITIMES)
         (MAKE-CANCEL-ITIMES-EQUALITY
          (ITIMES-FACTORS (CADR X))
          (ITIMES-FACTORS (CADDR X))
          (BAGINT (ITIMES-FACTORS (CADR X))
                   (ITIMES-FACTORS (CADDR X))))))
        (LIST 'IF
         (LIST 'INTEGERP (CADDR X))
         (MAKE-CANCEL-ITIMES-EQUALITY
          (ITIMES-FACTORS (CADR X))
          (ITIMES-FACTORS (CADDR X))

```



```

(ITIMES
  (IF
    (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                  (ITIMES-FACTORS (CADDR X))))
    (LIST 'IF
      (LIST 'INTEGERP (CADR X))
      (MAKE-CANCEL-ITIMES-EQUALITY
        (ITIMES-FACTORS (CADR X))
        (ITIMES-FACTORS (CADDR X))
        (BAGINT (ITIMES-FACTORS (CADR X))
                  (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F))
    X))
(INEG
  (IF
    (LISTP (CADADDR X))
    (IF
      (EQUAL (CAADDR X) 'ITIMES)
      (IF
        (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                      (ITIMES-FACTORS (CADDR X))))
        (LIST 'IF
          (LIST 'INTEGERP (CADR X))
          (MAKE-CANCEL-ITIMES-EQUALITY
            (ITIMES-FACTORS (CADR X))
            (ITIMES-FACTORS (CADDR X))
            (BAGINT (ITIMES-FACTORS (CADR X))
                    (ITIMES-FACTORS (CADDR X))))
          (LIST 'QUOTE F))
          X)
        X)
      X))
    (OTHERWISE X)))
(T X))
((LISTP (CADDR X))
 (CASE
  (CAADDR X)
  (IPLUS
   (IF
    (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                  (ITIMES-FACTORS (CADDR X))))
    (LIST 'IF
      (LIST 'INTEGERP (CADR X))
      (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))

```

```

                                                    (ITIMES-FACTORS (CADDR X))
                                                    (BAGINT (ITIMES-FACTORS (CADR X))
                                                    (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F))
    X))
  (ITIMES
  (IF
    (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                  (ITIMES-FACTORS (CADDR X))))
    (LIST 'IF
      (LIST 'INTEGERP (CADR X))
      (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                    (ITIMES-FACTORS (CADDR X))
                                    (BAGINT (ITIMES-FACTORS (CADR X))
                                            (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F))
    X))
  (INEG
  (IF
    (LISTP (CADADDR X))
    (IF
      (EQUAL (CAADDR X) 'ITIMES)
      (IF
        (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                      (ITIMES-FACTORS (CADDR X))))
        (LIST 'IF
          (LIST 'INTEGERP (CADR X))
          (MAKE-CANCEL-ITIMES-EQUALITY
            (ITIMES-FACTORS (CADR X))
            (ITIMES-FACTORS (CADDR X))
            (BAGINT (ITIMES-FACTORS (CADR X))
                    (ITIMES-FACTORS (CADDR X))))
          (LIST 'QUOTE F))
        X)
        X)
      (OTHERWISE X)))
    (T X)))
  (OTHERWISE
  (IF
    (LISTP (CADDR X))
    (CASE
      (CAADDR X)
      (IPLUS

```

```

(IF
  (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                (ITIMES-FACTORS (CADDR X))))
  (LIST 'IF
    (LIST 'INTEGERP (CADR X))
    (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                  (ITIMES-FACTORS (CADDR X))
                                  (BAGINT (ITIMES-FACTORS (CADR X))
                                           (ITIMES-FACTORS (CADDR X))))
    (LIST 'QUOTE F))
  X))
(ITIMES
  (IF
    (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                  (ITIMES-FACTORS (CADDR X))))
    (LIST 'IF
      (LIST 'INTEGERP (CADR X))
      (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                    (ITIMES-FACTORS (CADDR X))
                                    (BAGINT (ITIMES-FACTORS (CADR X))
                                             (ITIMES-FACTORS (CADDR X))))
      (LIST 'QUOTE F))
    X))
(INEG
  (IF
    (LISTP (CADADDR X))
    (IF
      (EQUAL (CAADADDR X) 'ITIMES)
      (IF
        (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                      (ITIMES-FACTORS (CADDR X))))
        (LIST 'IF
          (LIST 'INTEGERP (CADR X))
          (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                        (ITIMES-FACTORS (CADDR X))
                                        (BAGINT (ITIMES-FACTORS (CADR X))
                                                 (ITIMES-FACTORS (CADDR X))))
          (LIST 'QUOTE F))
        X)
        X)
      (OTHERWISE X))
    X)))
((LISTP (CADDR X))

```

```

(CASE
  (CAADDR X)
  (IPLUS
    (IF
      (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                    (ITIMES-FACTORS (CADDR X))))
      (LIST 'IF
        (LIST 'INTEGERP (CADR X))
        (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                      (ITIMES-FACTORS (CADDR X))
                                      (BAGINT (ITIMES-FACTORS (CADR X))
                                              (ITIMES-FACTORS (CADDR X))))
        (LIST 'QUOTE F))
      X))
  (ITIMES
    (IF
      (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                    (ITIMES-FACTORS (CADDR X))))
      (LIST 'IF
        (LIST 'INTEGERP (CADR X))
        (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                      (ITIMES-FACTORS (CADDR X))
                                      (BAGINT (ITIMES-FACTORS (CADR X))
                                              (ITIMES-FACTORS (CADDR X))))
        (LIST 'QUOTE F))
      X))
  (INEG
    (IF
      (LISTP (CADADDR X))
      (IF
        (EQUAL (CAADADDR X) 'ITIMES)
        (IF
          (LISTP (BAGINT (ITIMES-FACTORS (CADR X))
                        (ITIMES-FACTORS (CADDR X))))
          (LIST 'IF
            (LIST 'INTEGERP (CADR X))
            (MAKE-CANCEL-ITIMES-EQUALITY (ITIMES-FACTORS (CADR X))
                                          (ITIMES-FACTORS (CADDR X))
                                          (BAGINT (ITIMES-FACTORS (CADR X))
                                                  (ITIMES-FACTORS (CADDR X))))
            (LIST 'QUOTE F))
          X)
          X)
      X))

```

```

      (OTHERWISE X)))
    (T X))
  X)
X))

```

THEOREM: cancel-itimes-factors-expanded-cancel-itimes-factors
cancel-itimes-factors-expanded (x) = cancel-itimes-factors (x)

EVENT: Enable cancel-itimes-factors-expanded; name this event ‘cancel-itimes-factors-expanded-off’.

EVENT: Enable iplus-or-itimes-term; name this event ‘iplus-or-itimes-term-off’.

THEOREM: equal-itimes-list-eval\$list-delete-new-1
(fix-int (eval\$ (**t**, *elt*, *a*)) $\neq 0$)
 $\rightarrow$ (($x = \text{itimes-list}(\text{eval}\$('list, \text{delete}(elt, bag), a))$)
= **if** $elt \in bag$
then integerp(x)
 $\wedge$ (itimes(x , eval\$ (**t**, *elt*, *a*))
= itimes-list (eval\$ ('list, *bag*, *a*)))
else $x = \text{itimes-list}(\text{eval}\$('list, bag, a))$ **endif**)

THEOREM: equal-itimes-list-eval\$list-delete-new-2
(fix-int (eval\$ (**t**, *elt*, *a*)) $\neq 0$)
 $\rightarrow$ ((itimes-list (eval\$ ('list, delete(*elt*, *bag*), *a*)) = x)
= **if** $elt \in bag$
then integerp(x)
 $\wedge$ (itimes(x , eval\$ (**t**, *elt*, *a*))
= itimes-list (eval\$ ('list, *bag*, *a*)))
else $x = \text{itimes-list}(\text{eval}\$('list, bag, a))$ **endif**)

THEOREM: itimes-itimes-list-eval\$list-delete
($x \in bag$)
 $\rightarrow$ (itimes (eval\$ (**t**, x , *a*), itimes-list (eval\$ ('list, delete(x , *bag*), *a*)))
= itimes-list (eval\$ ('list, *bag*, *a*)))

THEOREM: equal-itimes-list-eval\$list-bagdiff
(subbagp (*in-both*, *bag1*)
 $\wedge$ subbagp (*in-both*, *bag2*)
 $\wedge$ (itimes-list (eval\$ ('list, *in-both*, *a*)) $\neq 0$))
 $\rightarrow$ ((itimes-list (eval\$ ('list, bagdiff(*bag1*, *in-both*), *a*))
= itimes-list (eval\$ ('list, bagdiff(*bag2*, *in-both*), *a*)))
= (itimes-list (eval\$ ('list, *bag1*, *a*))
= itimes-list (eval\$ ('list, *bag2*, *a*))))

THEOREM: membership-of-0-implies-itimes-list-is-0

$$(0 \in x) \rightarrow (\text{itimes-list}(x) = 0)$$

THEOREM: member-0-eval\$-list

$$(0 \in x) \rightarrow (0 \in \text{eval\$}('list, x, a))$$

THEOREM: itimes-list-eval\$-factors-lemma

$$\begin{aligned} & \text{itimes}(\text{itimes-list}(\text{eval\$}('list, \text{bagint}(bag1, bag2), a)), \\ & \quad \text{itimes-list}(\text{eval\$}('list, \text{bagdiff}(bag2, \text{bagint}(bag1, bag2)), a))) \\ &= \text{itimes-list}(\text{eval\$}('list, bag2, a)) \end{aligned}$$

THEOREM: itimes-list-eval\$-factors-lemma-prime

$$\begin{aligned} & \text{itimes}(\text{itimes-list}(\text{eval\$}('list, \text{bagint}(bag1, bag2), a)), \\ & \quad \text{itimes-list}(\text{eval\$}('list, \text{bagdiff}(bag1, \text{bagint}(bag1, bag2)), a))) \\ &= \text{itimes-list}(\text{eval\$}('list, bag1, a)) \end{aligned}$$

THEOREM: itimes-list-eval\$-factors

$$\text{itimes-list}(\text{eval\$}('list, \text{itimes-factors}(x), a)) = \text{fix-int}(\text{eval\$}(\mathbf{t}, x, a))$$

THEOREM: iplus-or-itimes-term-integerp-eval\$

$$\text{iplus-or-itimes-term}(x) \rightarrow \text{integerp}(\text{eval\$}(\mathbf{t}, x, a))$$

THEOREM: eval\$-list-bagint-0

$$\begin{aligned} & (\text{itimes-list}(\text{eval\$}('list, \text{bagint}(x, y), a)) = 0) \\ & \rightarrow ((\text{itimes-list}(\text{eval\$}('list, x, a)) = 0) \\ & \quad \wedge (\text{itimes-list}(\text{eval\$}('list, y, a)) = 0)) \end{aligned}$$

THEOREM: eval\$-list-bagint-0-implies-equal

$$\begin{aligned} & ((\text{itimes-list}(\text{eval\$}('list, \text{bagint}(\text{itimes-factors}(v), \text{itimes-factors}(w)), a)) \\ & \quad = 0) \\ & \wedge \text{integerp}(\text{eval\$}(\mathbf{t}, v, a)) \\ & \wedge \text{integerp}(\text{eval\$}(\mathbf{t}, w, a))) \\ & \rightarrow ((\text{eval\$}(\mathbf{t}, v, a) = \text{eval\$}(\mathbf{t}, w, a)) = \mathbf{t}) \end{aligned}$$

THEOREM: correctness-of-cancel-itimes-factors

$$\text{eval\$}(\mathbf{t}, x, a) = \text{eval\$}(\mathbf{t}, \text{cancel-itimes-factors-expanded}(x), a)$$

DEFINITION:

cancel-itimes-ilessp-factors(x)

$$\begin{aligned} &= \text{if listp}(x) \\ & \quad \text{then if car}(x) = \text{'ilessp} \\ & \quad \quad \text{then if listp}(\text{bagint}(\text{itimes-factors}(\text{cadr}(x)), \\ & \quad \quad \quad \text{itimes-factors}(\text{caddr}(x)))) \\ & \quad \quad \text{then make-cancel-itimes-inequality}(\text{itimes-factors}(\text{cadr}(x)), \\ & \quad \quad \quad \text{itimes-factors}(\text{caddr}(x)), \\ & \quad \quad \quad \text{bagint}(\text{itimes-factors}(\text{cadr}(x)), \end{aligned}$$

itimes-factors (caddr (x)))

else x endif
else x endif
else x endif

THEOREM: bagint-singleton

bagint (x, list (y))
= **if** y ∈ x **then** list (y)
else nil endif

THEOREM: izerop-ilessp-0-relationship

(fix-int (x) = 0) = ((¬ illessp (x, 0)) ∧ (¬ illessp (0, x)))

THEOREM: illessp-itimes-list-eval\$-list-delete-helper-1

illessp (0, w) → (illessp (itimes (x, w), itimes (w, u)) = illessp (x, u))

THEOREM: illessp-itimes-list-eval\$-list-delete-helper-2

illessp (w, 0) → (illessp (itimes (w, u), itimes (x, w)) = illessp (x, u))

THEOREM: illessp-itimes-list-eval\$-list-delete

((z ∈ y) ∧ (fix-int (eval\$ (t, z, a)) ≠ 0))
→ (illessp (x, itimes-list (eval\$ ('list, delete (z, y), a)))
= **if** illessp (0, eval\$ (t, z, a))
then illessp (itimes (x, eval\$ (t, z, a)),
itimes-list (eval\$ ('list, y, a)))
elseif illessp (eval\$ (t, z, a), 0)
then illessp (itimes-list (eval\$ ('list, y, a)),
itimes (x, eval\$ (t, z, a)))
else f endif)

THEOREM: illessp-itimes-list-eval\$-list-delete-prime-helper-1

illessp (0, w) → (illessp (itimes (w, u), itimes (x, w)) = illessp (u, x))

THEOREM: illessp-itimes-list-eval\$-list-delete-prime-helper-2

illessp (w, 0) → (illessp (itimes (x, w), itimes (w, u)) = illessp (u, x))

THEOREM: illessp-itimes-list-eval\$-list-delete-prime

((z ∈ y) ∧ (fix-int (eval\$ (t, z, a)) ≠ 0))
→ (illessp (itimes-list (eval\$ ('list, delete (z, y), a)), x)
= **if** illessp (0, eval\$ (t, z, a))
then illessp (itimes-list (eval\$ ('list, y, a)),
itimes (x, eval\$ (t, z, a)))
elseif illessp (eval\$ (t, z, a), 0)
then illessp (itimes (x, eval\$ (t, z, a)),
itimes-list (eval\$ ('list, y, a)))
else f endif)

THEOREM: ilessp-0-itimes
 ilessp(0, itimes(x , y))
 = ((ilessp(0, x) $\wedge$ ilessp(0, y)) $\vee$ (ilessp(x , 0) $\wedge$ ilessp(y , 0)))

THEOREM: ilessp-itimes-0
 ilessp(itimes(x , y), 0)
 = ((ilessp(0, x) $\wedge$ ilessp(y , 0)) $\vee$ (ilessp(x , 0) $\wedge$ ilessp(0, y)))

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff
 (subbagp($in-both$, $bag1$)
 $\wedge$ subbagp($in-both$, $bag2$)
 $\wedge$ (itimes-list(eval\$('list', $in-both$, a)) $\neq$ 0))
 $\rightarrow$ (ilessp(itimes-list(eval\$('list', bagdiff($bag1$, $in-both$), a)),
 itimes-list(eval\$('list', bagdiff($bag2$, $in-both$), a)))
 = **if** ilessp(0, itimes-list(eval\$('list', $in-both$, a)))
 then ilessp(itimes-list(eval\$('list', $bag1$, a)),
 itimes-list(eval\$('list', $bag2$, a)))
 else ilessp(itimes-list(eval\$('list', $bag2$, a)),
 itimes-list(eval\$('list', $bag1$, a))) **endif**)

THEOREM: zero-ilessp-implies-not-equal
 ilessp(0, x) $\rightarrow$ (0 $\neq x$)

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff-corollary-1
 (subbagp($in-both$, $bag1$)
 $\wedge$ subbagp($in-both$, $bag2$)
 $\wedge$ ilessp(0, itimes-list(eval\$('list', $in-both$, a))))
 $\rightarrow$ (ilessp(itimes-list(eval\$('list', bagdiff($bag1$, $in-both$), a)),
 itimes-list(eval\$('list', bagdiff($bag2$, $in-both$), a)))
 = ilessp(itimes-list(eval\$('list', $bag1$, a)),
 itimes-list(eval\$('list', $bag2$, a))))

THEOREM: ilessp-zero-implies-not-equal
 ilessp(x , 0) $\rightarrow$ (0 $\neq x$)

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff-corollary-2
 (subbagp($in-both$, $bag1$)
 $\wedge$ subbagp($in-both$, $bag2$)
 $\wedge$ ilessp(itimes-list(eval\$('list', $in-both$, a)), 0))
 $\rightarrow$ (ilessp(itimes-list(eval\$('list', bagdiff($bag1$, $in-both$), a)),
 itimes-list(eval\$('list', bagdiff($bag2$, $in-both$), a)))
 = ilessp(itimes-list(eval\$('list', $bag2$, a)),
 itimes-list(eval\$('list', $bag1$, a))))

THEOREM: member-0-itimes-factors-yields-0
 (eval\$(t , w , a) $\neq$ 0) $\rightarrow$ (0 $\notin$ itimes-factors(w))

THEOREM: member-0-itimes-factors-yields-0-ilessp-consequence-1
 $\text{ilessp}(\text{eval}\$(\mathbf{t}, w, a), 0) \rightarrow (0 \notin \text{itimes-factors}(w))$

THEOREM: member-0-itimes-factors-yields-0-ilessp-consequence-2
 $\text{ilessp}(0, \text{eval}\$(\mathbf{t}, w, a)) \rightarrow (0 \notin \text{itimes-factors}(w))$

EVENT: Disable ilessp-trichotomy; name this event ‘ilessp-trichotomy-on’.

EVENT: Enable izerop-ilessp-0-relationship; name this event ‘izerop-ilessp-0-relationship-off’.

THEOREM: eval\$-list-bagint-0-for-ilessp
 $((\neg \text{ilessp}(\text{itimes-list}(\text{eval}\$('list, \text{bagint}(x, y, a)), 0))$
 $\wedge (\neg \text{ilessp}(0, \text{itimes-list}(\text{eval}\$('list, \text{bagint}(x, y, a))))))$
 $\rightarrow ((\text{fix-int}(\text{itimes-list}(\text{eval}\$('list, x, a))) = 0)$
 $\wedge (\text{fix-int}(\text{itimes-list}(\text{eval}\$('list, y, a))) = 0))$

THEOREM: eval\$-list-bagint-0-implies-equal-for-ilessp-lemma
 $((\neg \text{ilessp}(\text{itimes-list}(\text{eval}\$('list,$
 $\text{bagint}(\text{itimes-factors}(v), \text{itimes-factors}(w)),$
 $a)),$
 $0))$
 $\wedge (\neg \text{ilessp}(0,$
 $\text{itimes-list}(\text{eval}\$('list,$
 $\text{bagint}(\text{itimes-factors}(v),$
 $\text{itimes-factors}(w)),$
 $a))))))$
 $\rightarrow (\text{fix-int}(\text{eval}\$(\mathbf{t}, v, a)) = \text{fix-int}(\text{eval}\$(\mathbf{t}, w, a)))$

THEOREM: equal-fix-int-to-ilessp
 $(\text{fix-int}(x) = \text{fix-int}(y)) \rightarrow (\neg \text{ilessp}(x, y))$

THEOREM: eval\$-list-bagint-0-implies-equal-for-ilessp
 $((\neg \text{ilessp}(\text{itimes-list}(\text{eval}\$('list,$
 $\text{bagint}(\text{itimes-factors}(v), \text{itimes-factors}(w)),$
 $a)),$
 $0))$
 $\wedge (\neg \text{ilessp}(0,$
 $\text{itimes-list}(\text{eval}\$('list,$
 $\text{bagint}(\text{itimes-factors}(v),$
 $\text{itimes-factors}(w)),$
 $a))))))$
 $\rightarrow ((\neg \text{ilessp}(\text{eval}\$(\mathbf{t}, v, a), \text{eval}\$(\mathbf{t}, w, a)))$
 $\wedge (\neg \text{ilessp}(\text{eval}\$(\mathbf{t}, w, a), \text{eval}\$(\mathbf{t}, v, a))))$

EVENT: Enable illessp-trichotomy; name this event ‘illessp-trichotomy-off1’.

THEOREM: correctness-of-cancel-itimes-illessp-factors
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-itimes-illessp-factors}(x), a)$

EVENT: Disable lessp-count-listp-cdr; name this event ‘lessp-count-listp-cdr-on’.

DEFINITION:
 $\text{disjoin-equalities-with-0}(factors)$
 $=$ **if** listp(cdr(factors))
 then list('or,
 cons('equal, cons(list('fix-int, car(factors)), '('0))),
 disjoin-equalities-with-0(cdr(factors)))
 else cons('equal, cons(list('fix-int, car(factors)), '('0)) **endif**

EVENT: Enable lessp-count-listp-cdr; name this event ‘lessp-count-listp-cdr-off1’.

DEFINITION:
 $\text{cancel-factors-0}(x)$
 $=$ **if** listp(x)
 then if car(x) = 'equal
 then if cadr(x) = ''0
 then if listp(cdr(itimes-factors(caddr(x))))
 then disjoin-equalities-with-0(itimes-factors(caddr(x)))
 else x **endif**
 elseif caddr(x) = ''0
 then if listp(cdr(itimes-factors(cadr(x))))
 then disjoin-equalities-with-0(itimes-factors(cadr(x)))
 else x **endif**
 else x **endif**
 else x **endif**
 else x **endif**

DEFINITION:
 $\text{some-eval}\$s\text{-to-0}(x, a)$
 $=$ **if** listp(x)
 then (fix-int(eval\$(t, car(x), a)) = 0)
 $\vee$ some-eval\$s-to-0(cdr(x), a)
 else f endif

THEOREM: eval\$-disjoin-equalities-with-0
 $\text{listp}(lst)$

→ (eval\$ (t, disjoin-equalities-with-0 (lst), a)
= some-eval\$s-to-0 (lst, a))

THEOREM: some-eval\$s-to-0-append
some-eval\$s-to-0 (append (x, y), a)
= (some-eval\$s-to-0 (x, a) ∨ some-eval\$s-to-0 (y, a))

THEOREM: some-eval\$s-to-0-eliminator
some-eval\$s-to-0 (x, a) = (itimes-list (eval\$ ('list, x, a)) = 0)

THEOREM: listp-cdr-factors-implies-integerp
listp (cdr (itimes-factors (v))) → integerp (eval\$ (t, v, a))

THEOREM: correctness-of-cancel-factors-0
eval\$ (t, x, a) = eval\$ (t, cancel-factors-0 (x), a)

EVENT: Disable lessp-count-listp-cdr; name this event 'lessp-count-listp-cdr-on1'.

DEFINITION:
conjoin-inequalities-with-0 (factors, parity)
= **if** listp (cdr (factors))
 then if parity
 then list ('or,
 list ('and,
 list ('ilessp, ''0, car (factors)),
 conjoin-inequalities-with-0 (cdr (factors), t)),
 list ('and,
 cons ('ilessp, cons (car (factors), ' ('0))),
 conjoin-inequalities-with-0 (cdr (factors), f)))
 else list ('or,
 list ('and,
 cons ('ilessp, cons (car (factors), ' ('0))),
 conjoin-inequalities-with-0 (cdr (factors), t)),
 list ('and,
 list ('ilessp, ''0, car (factors)),
 conjoin-inequalities-with-0 (cdr (factors), f))) **endif**
 elseif parity **then** list ('ilessp, ''0, car (factors))
 else cons ('ilessp, cons (car (factors), ' ('0))) **endif**

EVENT: Enable lessp-count-listp-cdr; name this event 'lessp-count-listp-cdr-off2'.

DEFINITION:

```

cancel-factors-ilessp-0( $x$ )
=  if listp( $x$ )
    then if car( $x$ ) = 'ilessp
        then if cadr( $x$ ) = ''0
            then if listp(cdr(itimes-factors(caddr( $x$ ))))
                then conjoin-inequalities-with-0(itimes-factors(caddr( $x$ )),
                                                    t)

                else  $x$  endif
            elseif cadr( $x$ ) = ''0
                then if listp(cdr(itimes-factors(cadr( $x$ ))))
                    then conjoin-inequalities-with-0(itimes-factors(cadr( $x$ )),
                                                        f)

                    else  $x$  endif
                else  $x$  endif
        else  $x$  endif
    else  $x$  endif

```

THEOREM: conjoin-inequalities-with-0-eliminator

```

listp( $x$ )
→ (eval$(t, conjoin-inequalities-with-0( $x$ , parity),  $a$ )
   =  if parity then illessp(0, itimes-list(eval$('list,  $x$ ,  $a$ )))
      else illessp(itimes-list(eval$('list,  $x$ ,  $a$ )), 0) endif)

```

THEOREM: correctness-of-cancel-factors-ilessp-0

eval\$(t, x , a) = eval\$(t, cancel-factors-ilessp-0(x), a)

EVENT: Enable equal-itimes-list-eval\$list-delete-new-1; name this event 'equal-itimes-list-eval\$list-delete-new-1-off'.

EVENT: Enable equal-itimes-list-eval\$list-delete-new-2; name this event 'equal-itimes-list-eval\$list-delete-new-2-off'.

EVENT: Enable itimes-itimes-list-eval\$list-delete; name this event 'itimes-itimes-list-eval\$list-delete-off'.

EVENT: Enable equal-itimes-list-eval\$list-bagdiff; name this event 'equal-itimes-list-eval\$list-bagdiff-off'.

EVENT: Enable itimes-list-eval\$factors-lemma; name this event 'itimes-list-eval\$factors-lemma-off'.

EVENT: Enable itimes-list-eval\$factors-lemma-prime; name this event 'itimes-list-eval\$factors-lemma-prime-off'.

EVENT: Enable itimes-list-eval\$-factors; name this event ‘itimes-list-eval\$-factors-off’.

EVENT: Enable iplus-or-itimes-term-integerp-eval\$; name this event ‘iplus-or-itimes-term-integerp-eval\$-off’.

EVENT: Enable eval\$list-bagint-0; name this event ‘eval\$list-bagint-0-off’.

EVENT: Enable eval\$list-bagint-0-implies-equal; name this event ‘eval\$list-bagint-0-implies-equal-off’.

EVENT: Enable izerop-ilessp-0-relationship; name this event ‘izerop-ilessp-0-relationship-off1’.

EVENT: Enable illessp-itimes-list-eval\$list-delete-helper-1; name this event ‘illessp-itimes-list-eval\$list-delete-helper-1-off’.

EVENT: Enable illessp-itimes-list-eval\$list-delete-helper-2; name this event ‘illessp-itimes-list-eval\$list-delete-helper-2-off’.

EVENT: Enable illessp-itimes-list-eval\$list-delete; name this event ‘illessp-itimes-list-eval\$list-delete-off’.

EVENT: Enable illessp-itimes-list-eval\$list-delete-prime-helper-1; name this event ‘illessp-itimes-list-eval\$list-delete-prime-helper-1-off’.

EVENT: Enable illessp-itimes-list-eval\$list-delete-prime-helper-2; name this event ‘illessp-itimes-list-eval\$list-delete-prime-helper-2-off’.

EVENT: Enable illessp-itimes-list-eval\$list-delete-prime; name this event ‘illessp-itimes-list-eval\$list-delete-prime-off’.

EVENT: Enable illessp-0-itimes; name this event ‘illessp-0-itimes-off’.

EVENT: Enable illessp-itimes-0; name this event ‘illessp-itimes-0-off’.

EVENT: Enable listp-cdr-factors-implies-integerp; name this event ‘listp-cdr-

factors-implies-integerp-off’.

EVENT: Enable correctness-of-cancel-itimes; name this event ‘correctness-of-cancel-itimes-off’.

EVENT: Enable correctness-of-cancel-itimes-ilessp; name this event ‘correctness-of-cancel-itimes-ilessp-off’.

EVENT: Enable not-integerp-implies-not-equal-iplus; name this event ‘not-integerp-implies-not-equal-iplus-off’.

EVENT: Enable not-integerp-implies-not-equal-itimes; name this event ‘not-integerp-implies-not-equal-itimes-off’.

EVENT: Enable subbagp-subsetp; name this event ‘subbagp-subsetp-off’.

EVENT: Enable eval\$-list-bagint-0-implies-equal-for-ilessp; name this event ‘eval\$-list-bagint-0-implies-equal-for-ilessp-off’.

DEFINITION:

```
split-out-ineg-terms (x)
=  if listp (x)
    then if listp (car (x))
        then if caar (x) = 'ineg
            then cons (car (split-out-ineg-terms (cdr (x))),
                        cons (cadar (x), cdr (split-out-ineg-terms (cdr (x)))))
            elseif (caar (x) = 'quote
                & negativep (cadar (x))
                & (negative-guts (cadar (x)) ≠ 0)
            then cons (car (split-out-ineg-terms (cdr (x))),
                        cons (list ('quote, negative-guts (cadar (x))),
                            cdr (split-out-ineg-terms (cdr (x)))))
            else cons (cons (car (x),
                            car (split-out-ineg-terms (cdr (x)))),
                        cdr (split-out-ineg-terms (cdr (x)))) endif
        else cons (cons (car (x), car (split-out-ineg-terms (cdr (x)))),
                        cdr (split-out-ineg-terms (cdr (x)))) endif
    else '(nil) endif
```

DEFINITION:

```

remove-inegs (x, y)
=  if listp (cdr (split-out-ineg-terms (x)))
     $\vee$  listp (cdr (split-out-ineg-terms (y)))
    then cons (append (cdr (split-out-ineg-terms (y)),
                      car (split-out-ineg-terms (x))),
              append (cdr (split-out-ineg-terms (x)),
                      car (split-out-ineg-terms (y))))
    else f endif

```

DEFINITION:

```

iplus-or-ineg-term (x)
=  (listp (x)  $\wedge$  ((car (x) = 'ineg)  $\vee$  (car (x) = 'iplus)))

```

DEFINITION:

```

make-cancel-ineg-terms-equality (x)
=  if remove-inegs (iplus-fringe (cadr (x)), iplus-fringe (caddr (x)))
    then if iplus-or-ineg-term (cadr (x))
        then if iplus-or-ineg-term (caddr (x))
            then list ('equal,
                      iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))),
                      iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))))
            else list ('if,
                      list ('integerp, caddr (x)),
                      list ('equal,
                            iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                            iplus-fringe (caddr (x))))),
                            iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                            iplus-fringe (caddr (x)))))),
                      list ('quote, f)) endif
        elseif iplus-or-ineg-term (caddr (x))
            then list ('if,
                      list ('integerp, cadr (x)),
                      list ('equal,
                            iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                            iplus-fringe (caddr (x))))),
                            iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                            iplus-fringe (caddr (x)))))),
                      list ('quote, f))
        else x endif
    else x endif

```

DEFINITION:

```

cancel-ineg-terms-from-equality (x)

```



```
=  if listp( $x$ )  $\wedge$  (car( $x$ ) = 'equal)
    then make-cancel-ineg-terms-equality( $x$ )
    else  $x$  endif
```

DEFINITION:

cancel-ineg-terms-from-equality-expanded(x)

```
=  if listp( $x$ )
    then if car( $x$ ) = 'equal
        then if remove-inegs(iplus-fringe(cadr( $x$ )),
                               iplus-fringe(caddr( $x$ )))
            then if listp(cadr( $x$ ))
                then case on caadr( $x$ ):
                    case = ineg
                    then if listp(caddr( $x$ ))
                        then case on caaddr( $x$ ):
                            case = ineg
                            then list('equal,
                                iplus-tree(car(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))),
                                iplus-tree(cdr(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))))
                            case = iplus
                            then list('equal,
                                iplus-tree(car(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))),
                                iplus-tree(cdr(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))))
                        otherwise list('if,
                            list('integerp,
                                cadr( $x$ )),
                            list('equal,
                                iplus-tree(car(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))),
                                iplus-tree(cdr(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))))
                            list('quote, f)) endcase
                    else list('if,
                        list('integerp, cadr( $x$ )),
                        list('equal,
                            iplus-tree(car(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))),
                            iplus-tree(cdr(remove-inegs(iplus-fringe(cadr( $x$ )),
                                                                iplus-fringe(caddr( $x$ ))))))
                        list('quote, f)) endif
```



```

                                case = ipplus
                                then list ('if,
                                    list ('integerp,
                                        cadr (x)),
                                    list ('equal,
                                        iplus-tree (car (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x))),
                                        iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x))),
                                    list ('quote, f))
                                otherwise x endcase
                            else x endif endcase
                    elseif listp (caddr (x))
                    then case on caaddr (x):
                        case = ineg
                        then list ('if,
                            list ('integerp, cadr (x)),
                            list ('equal,
                                iplus-tree (car (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x)))),
                                iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x)))),
                            list ('quote, f))
                        case = ipplus
                        then list ('if,
                            list ('integerp, cadr (x)),
                            list ('equal,
                                iplus-tree (car (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x)))),
                                iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x))),
                                                                    iplus-fringe (caddr (x)))),
                            list ('quote, f))
                        otherwise x endcase
                    else x endif
                else x endif
            else x endif
        else x endif

```

THEOREM: cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded
cancel-ineg-terms-from-equality-expanded (*x*)
= cancel-ineg-terms-from-equality (*x*)

EVENT: Enable cancel-ineg-terms-from-equality-expanded; name this event 'cancel-ineg-terms-from-equality-expanded-off'.

THEOREM: integerp-eval\$-iplus-or-ineg-term
 $\text{iplus-or-ineg-term}(x) \rightarrow \text{integerp}(\text{eval}(\mathbf{t}, x, a))$

EVENT: Enable iplus-or-ineg-term; name this event ‘iplus-or-ineg-term-off’.

THEOREM: eval\$iplus-list-car-remove-inegs
 $\text{remove-inegs}(x, y)$
 $\rightarrow (\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{car}(\text{remove-inegs}(x, y)), a))$
 $= \text{iplus}(\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{car}(\text{split-out-ineg-terms}(x)), a)),$
 $\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{cdr}(\text{split-out-ineg-terms}(y)), a)))$

THEOREM: eval\$iplus-list-cdr-remove-inegs
 $\text{remove-inegs}(x, y)$
 $\rightarrow (\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{cdr}(\text{remove-inegs}(x, y)), a))$
 $= \text{iplus}(\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{car}(\text{split-out-ineg-terms}(y)), a)),$
 $\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{cdr}(\text{split-out-ineg-terms}(x)), a)))$

THEOREM: minus-ineg
 $((x \in \mathbf{N}) \wedge (x \neq 0)) \rightarrow ((-x) = \text{ineg}(x))$

THEOREM: iplus-list-eval\$-car-split-out-ineg-terms
 $\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{car}(\text{split-out-ineg-terms}(x)), a))$
 $= \text{iplus}(\text{iplus-list}(\text{eval}(\mathbf{'list}, x, a)),$
 $\text{iplus-list}(\text{eval}(\mathbf{'list}, \text{cdr}(\text{split-out-ineg-terms}(x)), a)))$

EVENT: Enable remove-inegs; name this event ‘remove-inegs-off’.

THEOREM: correctness-of-cancel-ineg-terms-from-equality
 $\text{eval}(\mathbf{t}, x, a) = \text{eval}(\mathbf{t}, \text{cancel-ineg-terms-from-equality-expanded}(x), a)$

DEFINITION:
 $\text{make-cancel-ineg-terms-inequality}(x)$
 $= \text{if } \text{remove-inegs}(\text{iplus-fringe}(\text{cadr}(x)), \text{iplus-fringe}(\text{caddr}(x)))$
 $\text{ then list}(\mathbf{'ilessp},$
 $\text{ iplus-tree}(\text{car}(\text{remove-inegs}(\text{iplus-fringe}(\text{cadr}(x)),$
 $\text{ iplus-fringe}(\text{caddr}(x))))),$
 $\text{ iplus-tree}(\text{cdr}(\text{remove-inegs}(\text{iplus-fringe}(\text{cadr}(x)),$
 $\text{ iplus-fringe}(\text{caddr}(x))))))$
 $\text{ else } x \text{ endif}$

DEFINITION:
 $\text{cancel-ineg-terms-from-inequality}(x)$
 $= \text{if listp}(x) \wedge (\text{car}(x) = \mathbf{'ilessp})$
 $\text{ then if } \text{iplus-or-ineg-term}(\text{cadr}(x))$

```

    then make-cancel-ineg-terms-inequality (x)
    elseif iplus-or-ineg-term (caddr (x))
    then make-cancel-ineg-terms-inequality (x)
    else x endif
else x endif

```

DEFINITION:

cancel-ineg-terms-from-inequality-expanded (x)

```

= if listp (x)
  then if car (x) = 'ilessp
    then if listp (cadr (x))
      then case on caadr (x):
        case = neg
        then if remove-inegs (iplus-fringe (cadr (x)),
                               iplus-fringe (caddr (x)))
          then list ('ilessp,
                    iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))),
                    iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))))
          else x endif
        case = plus
        then if remove-inegs (iplus-fringe (cadr (x)),
                               iplus-fringe (caddr (x)))
          then list ('ilessp,
                    iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))),
                    iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                    iplus-fringe (caddr (x))))))
          else x endif
      otherwise if listp (caddr (x))
        then case on caaddr (x):
          case = neg
          then if remove-inegs (iplus-fringe (cadr (x)),
                                iplus-fringe (caddr (x)))
            then list ('ilessp,
                      iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                                      iplus-fringe (caddr (x))))),
                      iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                                      iplus-fringe (caddr (x))))))
            else x endif
          case = plus
          then if remove-inegs (iplus-fringe (cadr (x)),
                                iplus-fringe (caddr (x)))

```

```

                                then list ('ilessp,
                                           iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
                                           iplus-fringe (caddr (x))),
                                           iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
                                           iplus-fringe (caddr (x)))

                                else x endif
                                otherwise x endcase
                                else x endif endcase
elseif listp (caddr (x))
then case on caaddr (x):
  case = ineg
  then if remove-inegs (iplus-fringe (cadr (x)),
                        iplus-fringe (caddr (x)))
    then list ('ilessp,
               iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
               iplus-fringe (caddr (x))))),
               iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
               iplus-fringe (caddr (x))))))

    else x endif
  case = iplus
  then if remove-inegs (iplus-fringe (cadr (x)),
                        iplus-fringe (caddr (x)))
    then list ('ilessp,
               iplus-tree (car (remove-inegs (iplus-fringe (cadr (x)),
               iplus-fringe (caddr (x))))),
               iplus-tree (cdr (remove-inegs (iplus-fringe (cadr (x)),
               iplus-fringe (caddr (x))))))

    else x endif
  otherwise x endcase
  else x endif
else x endif
else x endif

```

THEOREM: cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded
cancel-ineg-terms-from-inequality-expanded (x)
= cancel-ineg-terms-from-inequality (x)

EVENT: Enable cancel-ineg-terms-from-inequality-expanded; name this event
‘cancel-ineg-terms-from-inequality-expanded-off’.

THEOREM: correctness-of-cancel-ineg-terms-from-inequality
eval\$ ($\mathbf{t}$, x , a) = eval\$ ($\mathbf{t}$, cancel-ineg-terms-from-inequality-expanded (x), a)

EVENT: Enable minus-ineg; name this event ‘minus-ineg-off’.

EVENT: Enable integerp-eval\$-iplus-or-ineg-term; name this event ‘integerp-eval\$-iplus-or-ineg-term-off’.

THEOREM: plus-iplus

$$((i \in \mathbf{N}) \wedge (j \in \mathbf{N})) \rightarrow ((i + j) = \text{iplus}(i, j))$$

THEOREM: iplus-constants

$$\text{iplus}(1 + i, \text{iplus}(1 + j, x)) = \text{iplus}((1 + i) + (1 + j), x)$$

THEOREM: numberp-is-integerp

$$(w \in \mathbf{N}) \rightarrow \text{integerp}(w)$$

THEOREM: difference-idifference

$$\begin{aligned} & ((x \in \mathbf{N}) \\ & \wedge (y \in \mathbf{N}) \\ & \wedge \text{if } y < x \text{ then f} \\ & \quad \text{else t endif}) \\ & \rightarrow ((y - x) = \text{idifference}(y, x)) \end{aligned}$$

THEOREM: cancel-constants-equal-lemma

$$\begin{aligned} & ((m \in \mathbf{N}) \wedge (n \in \mathbf{N})) \\ & \rightarrow ((\text{iplus}(m, x) = \text{iplus}(n, y)) \\ & \quad = \text{if } m < n \text{ then fix-int}(x) = \text{iplus}(n - m, y) \\ & \quad \text{else iplus}(m - n, x) = \text{fix-int}(y) \text{ endif}) \end{aligned}$$

THEOREM: cancel-constants-equal

$$\begin{aligned} & (\text{iplus}(1 + i, x) = \text{iplus}(1 + j, y)) \\ & = \text{if } i < j \text{ then fix-int}(x) = \text{iplus}(j - i, y) \\ & \quad \text{else iplus}(i - j, x) = \text{fix-int}(y) \text{ endif} \end{aligned}$$

THEOREM: ilessp-add1

$$(y \in \mathbf{N}) \rightarrow (\text{ilessp}(x, 1 + y) = (\neg \text{ilessp}(y, x)))$$

THEOREM: ilessp-add1-iplus

$$(y \in \mathbf{N}) \rightarrow (\text{ilessp}(x, \text{iplus}(1 + y, z)) = (\neg \text{ilessp}(\text{iplus}(y, z), x)))$$

THEOREM: cancel-constants-ilessp-lemma-1

$$\begin{aligned} & ((m \in \mathbf{N}) \wedge (n \in \mathbf{N})) \\ & \rightarrow (\text{ilessp}(\text{iplus}(m, x), \text{iplus}(n, y)) \\ & \quad = \text{if } m < n \text{ then ilessp}(x, \text{iplus}(n - m, y)) \\ & \quad \text{else ilessp}(\text{iplus}(m - n, x), y) \text{ endif}) \end{aligned}$$

THEOREM: cancel-constants-ilessp-lemma-2

$$\begin{aligned} & ((m \in \mathbf{N}) \wedge (n \in \mathbf{N})) \\ & \rightarrow (\text{ilessp}(\text{iplus}(m, x), \text{iplus}(n, y)) \\ & \quad = \text{if } m < n \text{ then } \neg \text{ilessp}(\text{iplus}((n - m) - 1, y), x) \\ & \quad \text{else ilessp}(\text{iplus}(m - n, x), y) \text{ endif}) \end{aligned}$$

THEOREM: cancel-constants-ilessp
 illessp (iplus (1 + i , x), iplus (1 + j , y))
 = **if** $i < j$ **then** $\neg$ illessp (iplus ($j - i - 1$, y), x)
 else illessp (iplus ($i - j$, x), y) **endif**

EVENT: Enable plus-iplus; name this event ‘plus-iplus-off’.

EVENT: Enable numberp-is-integerp; name this event ‘numberp-is-integerp-off’.

EVENT: Enable difference-idifference; name this event ‘difference-idifference-off’.

EVENT: Let us define the theory *integers* to consist of the following events:
 ileq, idifference, integerp-fix-int, integerp-iplus, integerp-idifference, integerp-ineg, integerp-iabs, integerp-itimes, fix-int-remover, fix-int-fix-int, fix-int-iplus, fix-int-idifference, fix-int-ineg, fix-int-iabs, fix-int-itimes, neg-iplus, neg-ineg, neg-fix-int, neg-of-non-integerp, neg-0, iplus-left-id, iplus-right-id, iplus-0-left, iplus-0-right, commutativity2-of-iplus, associativity-of-iplus, iplus-cancellation-1, iplus-cancellation-2, iplus-ineg1, iplus-ineg2, iplus-fix-int1, iplus-fix-int2, idifference-fix-int1, idifference-fix-int2, iplus-list, eval\$list-append, iplus-list-append, iplus-ineg3, iplus-ineg4, correctness-of-cancel-iplus, illessp-fix-int-1, illessp-fix-int-2, iplus-cancellation-1-for-illessp, iplus-cancellation-2-for-illessp, correctness-of-cancel-iplus-illessp, itimes-0-left, itimes-0-right, itimes-fix-int1, itimes-fix-int2, commutativity-of-itimes, itimes-distributes-over-iplus-proof, itimes-distributes-over-iplus, commutativity2-of-itimes, associativity-of-itimes, equal-itimes-0, equal-itimes-1, equal-itimes-minus-1, itimes-1-arg1, quotient-remainder-uniqueness, division-theorem, itimes-ineg-1, itimes-ineg-2, itimes-cancellation-1, itimes-cancellation-2, itimes-cancellation-3, integerp-iquotient, integerp-iremainder, integerp-idiv, integerp-imod, integerp-iquo, integerp-irem, iquotient-fix-int1, iquotient-fix-int2, iremainder-fix-int1, iremainder-fix-int2, idiv-fix-int1, idiv-fix-int2, imod-fix-int1, imod-fix-int2, iquo-fix-int1, iquo-fix-int2, irem-fix-int1, irem-fix-int2, fix-int-iquotient, fix-int-iremainder, fix-int-idiv, fix-int-imod, fix-int-iquo, fix-int-irem, itimes-list, itimes-list-append, member-append, equal-fix-int, subsetp, correctness-of-cancel-itimes, correctness-of-cancel-itimes-illessp, illessp-strict, eval\$list-cons, eval\$list-nlistp, eval\$list-atom, eval\$list-quote, eval\$list-other, iplus-x-y-ineg-x, correctness-of-cancel-ineg, integerp-iplus-list, eval\$list-iplus-list-delete, eval\$list-iplus-list-bagdiff, itimes-tree-ineg, itimes-factors, itimes-1, equal-ineg-ineg, illessp-ineg-ineg, fix-int-eval\$list-itimes-tree-rec, eval\$list-itimes-tree-ineg, neg-eval\$list-itimes-tree-ineg, iplus-eval\$list-itimes-tree-ineg, itimes-eval\$list-itimes-tree-ineg, iplus-or-itimes-term, cancel-itimes-factors, cancel-itimes-factors-expanded, cancel-itimes-factors-expanded-cancel-itimes-factors, membership-of-0-implies-itimes-list-is-0, member-0-eval\$list, correctness-of-cancel-itimes-factors, cancel-itimes-illessp-factors, bagint-singleton, illessp-itimes-list-eval\$list-bagdiff, illessp-itimes-list-eval\$list-bagdiff

corollary-1, member-0-itimes-factors-yields-0, member-0-itimes-factors-yields-0-ilessp-consequence-1, member-0-itimes-factors-yields-0-ilessp-consequence-2, ilessp-itimes-list-eval\$-list-bagdiff-corollary-2, correctness-of-cancel-itimes-ilessp-factors, disjoin-equalities-with-0, cancel-factors-0, some-eval\$s-to-0, eval\$-disjoin-equalities-with-0, some-eval\$s-to-0-append, some-eval\$s-to-0-eliminator, correctness-of-cancel-factors-0, conjoin-inequalities-with-0, cancel-factors-ilessp-0, split-out-ineg-terms, correctness-of-cancel-ineg-terms-from-equality, correctness-of-cancel-ineg-terms-from-inequality, iplus-constants, cancel-constants-equal, ilessp-add1, ilessp-add1-iplus, cancel-constants-ilessp.

EVENT: Disable eval\$; name this event ‘eval\$-on’.

EVENT: Let us define the theory *math-theory* to consist of the following events: cancel-constants-ilessp, cancel-constants-ilessp-lemma-2, cancel-constants-ilessp-lemma-1, ilessp-add1-iplus, ilessp-add1, cancel-constants-equal, cancel-constants-equal-lemma, iplus-constants, correctness-of-cancel-ineg-terms-from-inequality, cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded, *1*cancel-ineg-terms-from-inequality-expanded, cancel-ineg-terms-from-inequality, *1*cancel-ineg-terms-from-inequality, make-cancel-ineg-terms-inequality, *1*make-cancel-ineg-terms-inequality, correctness-of-cancel-ineg-terms-from-equality, iplus-list-eval\$-car-split-out-ineg-terms, eval\$-iplus-list-cdr-remove-inegs, eval\$-iplus-list-car-remove-inegs, cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded, *1*cancel-ineg-terms-from-equality-expanded, cancel-ineg-terms-from-equality, *1*cancel-ineg-terms-from-equality, make-cancel-ineg-terms-equality, *1*make-cancel-ineg-terms-equality, *1*iplus-or-ineg-term, *1*remove-inegs, split-out-ineg-terms, *1*split-out-ineg-terms, correctness-of-cancel-factors-ilessp-0, conjoin-inequalities-with-0-eliminator, cancel-factors-ilessp-0, *1*cancel-factors-ilessp-0, conjoin-inequalities-with-0, *1*conjoin-inequalities-with-0, correctness-of-cancel-factors-0, some-eval\$s-to-0-eliminator, some-eval\$s-to-0-append, eval\$-disjoin-equalities-with-0, some-eval\$s-to-0, *1*some-eval\$s-to-0, cancel-factors-0, *1*cancel-factors-0, disjoin-equalities-with-0, *1*disjoin-equalities-with-0, correctness-of-cancel-itimes-ilessp-factors, equal-fix-int-to-ilessp, eval\$-list-bagint-0-implies-equal-for-ilessp-lemma, eval\$-list-bagint-0-for-ilessp, member-0-itimes-factors-yields-0-ilessp-consequence-2, member-0-itimes-factors-yields-0-ilessp-consequence-1, member-0-itimes-factors-yields-0, ilessp-itimes-list-eval\$-list-bagdiff-corollary-2, ilessp-zero-implies-not-equal, ilessp-itimes-list-eval\$-list-bagdiff-corollary-1, zero-ilessp-implies-not-equal, ilessp-itimes-list-eval\$-list-bagdiff, bagint-singleton, cancel-itimes-ilessp-factors, *1*cancel-itimes-ilessp-factors, correctness-of-cancel-itimes-factors, member-0-eval\$-list, membership-of-0-implies-itimes-list-is-0, cancel-itimes-factors-expanded-cancel-itimes-factors, *1*cancel-itimes-factors-expanded, cancel-itimes-factors, *1*cancel-itimes-factors, *1*iplus-or-itimes-term, itimes-eval\$-itimes-tree-ineg, iplus-eval\$-itimes-tree-ineg, neg-eval\$-itimes-tree-ineg, eval\$-itimes-tree-ineg, fix-int-eval\$-itimes-tree-rec, ilessp-ineg-ineg, equal-ineg-ineg, itimes-1, itimes-factors, *1*itimes-factors, *1*itimes-

tree-ineg, illessp-strict, illessp-itimes-right-negative, illessp-itimes-right-positive,
 cancel-itimes-illessp, *1*cancel-itimes-illessp, *1*make-cancel-itimes-inequality, *1*itimes-
 tree-no-fix-int, subsetp, *1*subsetp, equal-fix-int, member-append, itimes-list-
 append, cancel-itimes, *1*cancel-itimes, *1*make-cancel-itimes-equality, integerp-
 itimes-list, itimes-list, *1*itimes-list, *1*itimes-tree, itimes-tree-rec, *1*itimes-
 tree-rec, itimes-fringe, *1*itimes-fringe, fix-int-irem, fix-int-iquo, fix-int-imod,
 fix-int-idiv, fix-int-iremremainder, fix-int-iquotient, irem-fix-int2, irem-fix-int1, iquo-
 fix-int2, iquo-fix-int1, imod-fix-int2, imod-fix-int1, idiv-fix-int2, idiv-fix-int1,
 iremainder-fix-int2, iremainder-fix-int1, iquotient-fix-int2, iquotient-fix-int1, integerp-
 irem, integerp-iquo, integerp-imod, integerp-idiv, integerp-iremremainder, integerp-
 iquotient, itimes-cancellation-3, itimes-cancellation-2, itimes-cancellation-1, itimes-
 neg-2, itimes-ineg-1, iquo-irem-uniqueness, division-theorem-for-truncate-to-zero,
 division-theorem-for-truncate-to-zero-part3, division-theorem-for-truncate-to-zero-
 part2, division-theorem-for-truncate-to-zero-part1, idiv-imod-uniqueness, division-
 theorem-for-truncate-to-neginf, division-theorem-for-truncate-to-neginf-part3, division-
 theorem-for-truncate-to-neginf-part2, division-theorem-for-truncate-to-neginf-part1,
 iquotient-iremremainder-uniqueness, division-theorem, division-theorem-part3, division-
 theorem-part2, division-theorem-part1, quotient-remainder-uniqueness, itimes-
 1-arg1, equal-itimes-minus-1, equal-itimes-1, equal-itimes-0, associativity-of-itimes,
 commutativity2-of-itimes, itimes-distributes-over-iplus, itimes-distributes-over-
 iplus-proof, commutativity-of-itimes, itimes-fix-int2, itimes-fix-int1, itimes-0-
 right, itimes-0-left, correctness-of-cancel-iplus-illessp, cancel-iplus-illessp, *1*cancel-
 iplus-illessp, *1*iplus-tree-no-fix-int, iplus-cancellation-2-for-illessp, iplus-cancellation-
 1-for-illessp, cancel-iplus-illessp-1, *1*cancel-iplus-illessp-1, *1*make-cancel-iplus-
 inequality-1, illessp-fix-int-2, illessp-fix-int-1, correctness-of-cancel-iplus, iplus-
 neg7, iplus-list-append, eval\$-iplus-list-delete, *1*cancel-iplus, eval\$-list-append,
 integerp-iplus-list, iplus-list, *1*iplus-list, *1*iplus-tree, iplus-tree-rec, *1*iplus-
 tree-rec, iplus-fringe, *1*iplus-fringe, correctness-of-cancel-ineg, iplus-x-y-ineg-
 x, iplus-ineg4, iplus-ineg3, eval\$-cancel-ineg-aux-is-its-fn, eval\$-cancel-ineg-aux-
 fn, *1*eval\$-cancel-ineg-aux-fn, eval\$-other, eval\$-litatom, eval\$-list-nlistp, eval\$-
 list-cons, cancel-ineg, *1*cancel-ineg, cancel-ineg-aux, *1*cancel-ineg-aux, idifference-
 fix-int2, idifference-fix-int1, iplus-fix-int2, iplus-fix-int1, iplus-ineg2, iplus-ineg1,
 iplus-cancellation-2, iplus-cancellation-1, associativity-of-iplus, commutativity-
 of-iplus, commutativity2-of-iplus, iplus-0-right, iplus-0-left, ineg-0, ineg-fix-int,
 ineg-ineg, ineg-iplus, fix-int-itimes, fix-int-iabs, fix-int-ineg, fix-int-idifference,
 fix-int-iplus, fix-int-fix-int, fix-int-remover, integerp-itimes, integerp-iabs, integerp-
 ineg, integerp-idifference, integerp-iplus, integerp-fix-int, *1*irem, *1*iquo, *1*imod,
 *1*idiv, *1*iremremainder, *1*iquotient, *1*itimes, *1*iabs, idifference, *1*idiffer-
 ence, *1*ineg, *1*iplus, ileq, *1*ileq, *1*illessp, izerop, *1*izerop, *1*fix-int,
 *1*integerp, gcd-idempotence, gcd-x-x, commutativity2-of-gcd, commutativity2-
 of-gcd-zero-case, associativity-of-gcd, associativity-of-gcd-zero-case, common-divisor-
 divides-gcd, gcd-is-the-greatest, distributivity-of-times-over-gcd, distributivity-
 of-times-over-gcd-proof, remainder-gcd, gcd-plus-instance, gcd-plus, gcd-plus-
 proof, gcd-plus-instance-temp, gcd-plus-instance-temp-proof, lessp-gcd, equal-

gcd-0, gcd-1, gcd-0, single-number-induction, *1*single-number-induction, commutativity-
of-gcd, log-exp, log-times-exp, log-times-exp-proof, log-times, log-times-proof,
log-quotient-exp, log-quotient-times, log-quotient-times-proof, log-quotient, leq-
log-log, double-log-induction, *1*double-log-induction, log-1, log-0, equal-log-0,
exp-difference, equal-exp-1, equal-exp-0, exp-exp, exp-times, exp-0-arg2, exp-
1-arg1, exp-0-arg1, exp-plus, exp-add1, exp-zero, quotient-exp, remainder-exp-
exp, double-number-induction, *1*double-number-induction, remainder-exp, gcd,
*1*gcd, log, *1*log, exp, *1*exp, correctness-of-cancel-quotient-times, cancel-
quotient-times, *1*cancel-quotient-times, lessp-quotient, quotient-x-x, quotient-
1-arg1, quotient-1-arg1-casesplit, quotient-1-arg2, leq-quotient, quotient-quotient,
quotient-plus-times-times-instance, quotient-plus-times-times, quotient-plus-times-
times-proof, quotient-plus-fact, quotient-remainder-instance, quotient-remainder,
quotient-remainder-times, quotient-difference3, quotient-difference2, quotient-
lessp-arg1, quotient-difference1, quotient-times-times-proof, quotient-times-instance,
quotient-times, quotient-times-proof, quotient-times-instance-temp-proof, quotient-
plus, quotient-plus-proof, quotient-sub1, equal-quotient-0, quotient-add1, quotient-
zero, quotient-of-non-number, quotient-noop, transitivity-of-divides, remainder-
x-x, remainder-1-arg2, remainder-1-arg1, remainder-remainder, remainder-plus-
times-times-instance, remainder-plus-times-times, remainder-plus-times-times-
proof, remainder-plus-fact, remainder-difference2, double-remainder-induction,
*1*double-remainder-induction, remainder-difference1, remainder-times2-instance,
remainder-times2, remainder-times2-proof, remainder-times-times-proof, remainder-
times1-instance, remainder-times1-instance-proof, remainder-times1, remainder-
times1-proof, equal-remainder-plus-remainder-proof, equal-remainder-plus-0, equal-
remainder-plus-0-proof, remainder-plus, remainder-plus-proof, remainder-add1,
remainder-quotient-elim, remainder-zero, remainder-of-non-number, remainder-
noop, lessp-remainder, correctness-of-cancel-equal-times, cancel-equal-times, *1*cancel-
equal-times, correctness-of-cancel-lessp-times, cancel-lessp-times, *1*cancel-lessp-
times, infer-equality-from-not-lessp, lessp-times-arg1, eval\$-equal, and-not-zero-
tree, *1*and-not-zero-tree, or-zero-tree, *1*or-zero-tree, times-fringe, *1*times-
fringe, times-tree, *1*times-tree, lessp-1-times, lessp-plus-times2, lessp-plus-times1,
lessp-plus-times-proof, lessp-times-cancellation-proof, lessp-times3, lessp-times3-
proof2, lessp-times3-proof1, lessp-times2, lessp-times2-proof, lessp-times1, lessp-
times1-proof, times-1-arg1, times-quotient, times-quotient-proof, times-distributes-
over-difference, times-distributes-over-difference-proof, associativity-of-times, commutativity2-
of-times, times-distributes-over-plus, times-distributes-over-plus-proof, commutativity-
of-times, times-add1, times-zero, equal-sub1-0, equal-times-1, equal-times-0, correctness-
of-cancel-lessp-plus, cancel-lessp-plus, *1*cancel-lessp-plus, difference-x-x, difference-
difference-arg2, difference-difference-arg1, difference-sub1-arg2, difference-add1-
arg2, difference-leq-arg1, difference-elim, correctness-of-cancel-difference-plus,
cancel-difference-plus, *1*cancel-difference-plus, correctness-of-cancel-equal-plus,
cancel-equal-plus, *1*cancel-equal-plus, eval\$-quote, plus-tree, *1*plus-tree, plus-
fringe, *1*plus-fringe, diff-diff-arg2, diff-diff-arg1, difference-plus-plus-cancellation-
proof, difference-plus-cancellation-proof, plus-difference-arg2, plus-difference-arg1,

associativity-of-plus, plus-add1-arg2, plus-add1-arg1, plus-zero-arg2, commutativity2-of-plus, commutativity-of-plus, equal-difference-0, equal-plus-0, member-bagint, occurrences-bagdiff, occurrences-bagint, subbagp-bagint2, subbagp-bagint1, subbagp-cdr2, subbagp-cdr1, subbagp-delete, bagdiff-delete, member-bagdiff, occurrences-delete, member-delete-implies-membership, member-delete, member-non-list, equal-occurrences-zero, delete-delete, delete-non-member, subbagp, *1*subbagp, occurrences, *1*occurrences, bagint, *1*bagint, bagdiff, *1*bagdiff, delete, *1*delete.

EVENT: Enable cancel-constants-ilessp; name this event ‘cancel-constants-ilessp-off’.

EVENT: Enable cancel-constants-ilessp-lemma-2; name this event ‘cancel-constants-ilessp-lemma-2-off’.

EVENT: Enable cancel-constants-ilessp-lemma-1; name this event ‘cancel-constants-ilessp-lemma-1-off’.

EVENT: Enable illessp-add1-iplus; name this event ‘illessp-add1-iplus-off’.

EVENT: Enable illessp-add1; name this event ‘illessp-add1-off’.

EVENT: Enable cancel-constants-equal; name this event ‘cancel-constants-equal-off’.

EVENT: Enable cancel-constants-equal-lemma; name this event ‘cancel-constants-equal-lemma-off’.

EVENT: Enable iplus-constants; name this event ‘iplus-constants-off’.

EVENT: Enable correctness-of-cancel-ineg-terms-from-inequality; name this event ‘correctness-of-cancel-ineg-terms-from-inequality-off’.

EVENT: Enable cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded; name this event ‘cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded-off’.

EVENT: Enable *1*cancel-ineg-terms-from-inequality-expanded; name this event ‘g*1*cancel-ineg-terms-from-inequality-expanded-off’.

EVENT: Enable cancel-ineg-terms-from-inequality; name this event ‘cancel-ineg-terms-from-inequality-off’.

EVENT: Enable *1*cancel-ineg-terms-from-inequality; name this event ‘g*1*cancel-ineg-terms-from-inequality-off’.

EVENT: Enable make-cancel-ineg-terms-inequality; name this event ‘make-cancel-ineg-terms-inequality-off’.

EVENT: Enable *1*make-cancel-ineg-terms-inequality; name this event ‘g*1*make-cancel-ineg-terms-inequality-off’.

EVENT: Enable correctness-of-cancel-ineg-terms-from-equality; name this event ‘correctness-of-cancel-ineg-terms-from-equality-off’.

EVENT: Enable iplus-list-eval\$-car-split-out-ineg-terms; name this event ‘iplus-list-eval\$-car-split-out-ineg-terms-off’.

EVENT: Enable eval\$-iplus-list-cdr-remove-inegs; name this event ‘eval\$-iplus-list-cdr-remove-inegs-off’.

EVENT: Enable eval\$-iplus-list-car-remove-inegs; name this event ‘eval\$-iplus-list-car-remove-inegs-off’.

EVENT: Enable cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded; name this event ‘cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded-off’.

EVENT: Enable *1*cancel-ineg-terms-from-equality-expanded; name this event ‘g*1*cancel-ineg-terms-from-equality-expanded-off’.

EVENT: Enable cancel-ineg-terms-from-equality; name this event ‘cancel-ineg-terms-from-equality-off’.

EVENT: Enable *1*cancel-ineg-terms-from-equality; name this event ‘g*1*cancel-ineg-terms-from-equality-off’.

EVENT: Enable make-cancel-ineg-terms-equality; name this event ‘make-cancel-

ineg-terms-equality-off’.

EVENT: Enable *1*make-cancel-ineg-terms-equality; name this event ‘g*1*make-cancel-ineg-terms-equality-off’.

EVENT: Enable *1*iplus-or-ineg-term; name this event ‘g*1*iplus-or-ineg-term-off’.

EVENT: Enable *1*remove-inegs; name this event ‘g*1*remove-inegs-off’.

EVENT: Enable split-out-ineg-terms; name this event ‘split-out-ineg-terms-off’.

EVENT: Enable *1*split-out-ineg-terms; name this event ‘g*1*split-out-ineg-terms-off’.

EVENT: Enable correctness-of-cancel-factors-ilessp-0; name this event ‘correctness-of-cancel-factors-ilessp-0-off’.

EVENT: Enable conjoin-inequalities-with-0-eliminator; name this event ‘conjoin-inequalities-with-0-eliminator-off’.

EVENT: Enable cancel-factors-ilessp-0; name this event ‘cancel-factors-ilessp-0-off’.

EVENT: Enable *1*cancel-factors-ilessp-0; name this event ‘g*1*cancel-factors-ilessp-0-off’.

EVENT: Enable conjoin-inequalities-with-0; name this event ‘conjoin-inequalities-with-0-off’.

EVENT: Enable *1*conjoin-inequalities-with-0; name this event ‘g*1*conjoin-inequalities-with-0-off’.

EVENT: Enable correctness-of-cancel-factors-0; name this event ‘correctness-of-cancel-factors-0-off’.

EVENT: Enable some-eval\$s-to-0-eliminator; name this event ‘some-eval\$s-to-0-

eliminator-off’.

EVENT: Enable some-eval\$s-to-0-append; name this event ‘some-eval\$s-to-0-append-off’.

EVENT: Enable eval\$-disjoin-equalities-with-0; name this event ‘eval\$-disjoin-equalities-with-0-off’.

EVENT: Enable some-eval\$s-to-0; name this event ‘some-eval\$s-to-0-off’.

EVENT: Enable *1*some-eval\$s-to-0; name this event ‘g*1*some-eval\$s-to-0-off’.

EVENT: Enable cancel-factors-0; name this event ‘cancel-factors-0-off’.

EVENT: Enable *1*cancel-factors-0; name this event ‘g*1*cancel-factors-0-off’.

EVENT: Enable disjoin-equalities-with-0; name this event ‘disjoin-equalities-with-0-off’.

EVENT: Enable *1*disjoin-equalities-with-0; name this event ‘g*1*disjoin-equalities-with-0-off’.

EVENT: Enable correctness-of-cancel-itimes-ilessp-factors; name this event ‘correctness-of-cancel-itimes-ilessp-factors-off’.

EVENT: Enable equal-fix-int-to-ilessp; name this event ‘equal-fix-int-to-ilessp-off’.

EVENT: Enable eval\$list-bagint-0-implies-equal-for-ilessp-lemma; name this event ‘eval\$list-bagint-0-implies-equal-for-ilessp-lemma-off’.

EVENT: Enable eval\$list-bagint-0-for-ilessp; name this event ‘eval\$list-bagint-0-for-ilessp-off’.

EVENT: Enable member-0-itimes-factors-yields-0-ilessp-consequence-2; name this event ‘member-0-itimes-factors-yields-0-ilessp-consequence-2-off’.

EVENT: Enable member-0-itimes-factors-yields-0-ilessp-consequence-1; name this event ‘member-0-itimes-factors-yields-0-ilessp-consequence-1-off’.

EVENT: Enable member-0-itimes-factors-yields-0; name this event ‘member-0-itimes-factors-yields-0-off’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff-corollary-2; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-2-off’.

EVENT: Enable ilessp-zero-implies-not-equal; name this event ‘ilessp-zero-implies-not-equal-off’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff-corollary-1; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-1-off’.

EVENT: Enable zero-ilessp-implies-not-equal; name this event ‘zero-ilessp-implies-not-equal-off’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-off’.

EVENT: Enable bagint-singleton; name this event ‘bagint-singleton-off’.

EVENT: Enable cancel-itimes-ilessp-factors; name this event ‘cancel-itimes-ilessp-factors-off’.

EVENT: Enable *1*cancel-itimes-ilessp-factors; name this event ‘g*1*cancel-itimes-ilessp-factors-off’.

EVENT: Enable correctness-of-cancel-itimes-factors; name this event ‘correctness-of-cancel-itimes-factors-off’.

EVENT: Enable member-0-eval\$list; name this event ‘member-0-eval\$list-off’.

EVENT: Enable membership-of-0-implies-itimes-list-is-0; name this event ‘membership-of-0-implies-itimes-list-is-0-off’.

EVENT: Enable cancel-itimes-factors-expanded-cancel-itimes-factors; name this event 'cancel-itimes-factors-expanded-cancel-itimes-factors-off'.

EVENT: Enable $*1*$ cancel-itimes-factors-expanded; name this event ' $g*1*$ cancel-itimes-factors-expanded-off'.

EVENT: Enable cancel-itimes-factors; name this event 'cancel-itimes-factors-off'.

EVENT: Enable $*1*$ cancel-itimes-factors; name this event ' $g*1*$ cancel-itimes-factors-off'.

EVENT: Enable $*1*$ iplus-or-itimes-term; name this event ' $g*1*$ iplus-or-itimes-term-off'.

EVENT: Enable itimes-eval\$-itimes-tree-ineg; name this event 'itimes-eval\$-itimes-tree-ineg-off'.

EVENT: Enable iplus-eval\$-itimes-tree-ineg; name this event 'iplus-eval\$-itimes-tree-ineg-off'.

EVENT: Enable neg-eval\$-itimes-tree-ineg; name this event 'neg-eval\$-itimes-tree-ineg-off'.

EVENT: Enable eval\$-itimes-tree-ineg; name this event 'eval\$-itimes-tree-ineg-off'.

EVENT: Enable fix-int-eval\$-itimes-tree-rec; name this event 'fix-int-eval\$-itimes-tree-rec-off'.

EVENT: Enable illessp-ineg-ineg; name this event 'illessp-ineg-ineg-off'.

EVENT: Enable equal-ineg-ineg; name this event 'equal-ineg-ineg-off'.

EVENT: Enable itimes-1; name this event 'itimes-1-off'.

EVENT: Enable itimes-factors; name this event 'itimes-factors-off'.

EVENT: Enable $*1*$ itimes-factors; name this event ' $g*1*$ itimes-factors-off'.

EVENT: Enable $*1*$ itimes-tree-ineg; name this event ' $g*1*$ itimes-tree-ineg-off'.

EVENT: Enable ilssp-strict; name this event 'ilssp-strict-off'.

EVENT: Enable ilssp-itimes-right-negative; name this event 'ilssp-itimes-right-negative-off'.

EVENT: Enable ilssp-itimes-right-positive; name this event 'ilssp-itimes-right-positive-off'.

EVENT: Enable cancel-itimes-ilssp; name this event 'cancel-itimes-ilssp-off'.

EVENT: Enable $*1*$ cancel-itimes-ilssp; name this event ' $g*1*$ cancel-itimes-ilssp-off'.

EVENT: Enable $*1*$ make-cancel-itimes-inequality; name this event ' $g*1*$ make-cancel-itimes-inequality-off'.

EVENT: Enable $*1*$ itimes-tree-no-fix-int; name this event ' $g*1*$ itimes-tree-no-fix-int-off'.

EVENT: Enable subsetp; name this event 'subsetp-off'.

EVENT: Enable $*1*$ subsetp; name this event ' $g*1*$ subsetp-off'.

EVENT: Enable equal-fix-int; name this event 'equal-fix-int-off'.

EVENT: Enable member-append; name this event 'member-append-off'.

EVENT: Enable itimes-list-append; name this event 'itimes-list-append-off'.

EVENT: Enable cancel-itimes; name this event 'cancel-itimes-off'.

EVENT: Enable $*1*$ cancel-itimes; name this event ' $g*1*$ cancel-itimes-off'.

EVENT: Enable *1*make-cancel-itimes-equality; name this event 'g*1*make-cancel-itimes-equality-off'.

EVENT: Enable integerp-itimes-list; name this event 'integerp-itimes-list-off'.

EVENT: Enable itimes-list; name this event 'itimes-list-off'.

EVENT: Enable *1*itimes-list; name this event 'g*1*itimes-list-off'.

EVENT: Enable *1*itimes-tree; name this event 'g*1*itimes-tree-off'.

EVENT: Enable itimes-tree-rec; name this event 'itimes-tree-rec-off'.

EVENT: Enable *1*itimes-tree-rec; name this event 'g*1*itimes-tree-rec-off'.

EVENT: Enable itimes-fringe; name this event 'itimes-fringe-off'.

EVENT: Enable *1*itimes-fringe; name this event 'g*1*itimes-fringe-off'.

EVENT: Enable fix-int-irem; name this event 'fix-int-irem-off'.

EVENT: Enable fix-int-iquo; name this event 'fix-int-iquo-off'.

EVENT: Enable fix-int-imod; name this event 'fix-int-imod-off'.

EVENT: Enable fix-int-idiv; name this event 'fix-int-idiv-off'.

EVENT: Enable fix-int-iremainder; name this event 'fix-int-iremainder-off'.

EVENT: Enable fix-int-iquotient; name this event 'fix-int-iquotient-off'.

EVENT: Enable irem-fix-int2; name this event 'irem-fix-int2-off'.

EVENT: Enable irem-fix-int1; name this event 'irem-fix-int1-off'.

EVENT: Enable iquo-fix-int2; name this event 'iquo-fix-int2-off'.

EVENT: Enable iquo-fix-int1; name this event 'iquo-fix-int1-off'.

EVENT: Enable imod-fix-int2; name this event 'imod-fix-int2-off'.

EVENT: Enable imod-fix-int1; name this event 'imod-fix-int1-off'.

EVENT: Enable idiv-fix-int2; name this event 'idiv-fix-int2-off'.

EVENT: Enable idiv-fix-int1; name this event 'idiv-fix-int1-off'.

EVENT: Enable iremainder-fix-int2; name this event 'iremainder-fix-int2-off'.

EVENT: Enable iremainder-fix-int1; name this event 'iremainder-fix-int1-off'.

EVENT: Enable iquotient-fix-int2; name this event 'iquotient-fix-int2-off'.

EVENT: Enable iquotient-fix-int1; name this event 'iquotient-fix-int1-off'.

EVENT: Enable integerp-irem; name this event 'integerp-irem-off'.

EVENT: Enable integerp-iquo; name this event 'integerp-iquo-off'.

EVENT: Enable integerp-imod; name this event 'integerp-imod-off'.

EVENT: Enable integerp-idiv; name this event 'integerp-idiv-off'.

EVENT: Enable integerp-iremainder; name this event 'integerp-iremainder-off'.

EVENT: Enable integerp-iquotient; name this event 'integerp-iquotient-off'.

EVENT: Enable itimes-cancellation-3; name this event 'itimes-cancellation-3-off'.

EVENT: Enable itimes-cancellation-2; name this event ‘itimes-cancellation-2-off’.

EVENT: Enable itimes-cancellation-1; name this event ‘itimes-cancellation-1-off’.

EVENT: Enable itimes-ineg-2; name this event ‘itimes-ineg-2-off’.

EVENT: Enable itimes-ineg-1; name this event ‘itimes-ineg-1-off’.

EVENT: Enable iquo-irem-uniqueness; name this event ‘iquo-irem-uniqueness-off’.

EVENT: Enable division-theorem-for-truncate-to-zero; name this event ‘division-theorem-for-truncate-to-zero-off’.

EVENT: Enable division-theorem-for-truncate-to-zero-part3; name this event ‘division-theorem-for-truncate-to-zero-part3-off’.

EVENT: Enable division-theorem-for-truncate-to-zero-part2; name this event ‘division-theorem-for-truncate-to-zero-part2-off’.

EVENT: Enable division-theorem-for-truncate-to-zero-part1; name this event ‘division-theorem-for-truncate-to-zero-part1-off’.

EVENT: Enable idiv-imod-uniqueness; name this event ‘idiv-imod-uniqueness-off’.

EVENT: Enable division-theorem-for-truncate-to-neginf; name this event ‘division-theorem-for-truncate-to-neginf-off’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part3; name this event ‘division-theorem-for-truncate-to-neginf-part3-off’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part2; name this event ‘division-theorem-for-truncate-to-neginf-part2-off’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part1; name this event 'division-theorem-for-truncate-to-neginf-part1-off'.

EVENT: Enable iquotient-iremainder-uniqueness; name this event 'iquotient-iremainder-uniqueness-off'.

EVENT: Enable division-theorem; name this event 'division-theorem-off'.

EVENT: Enable division-theorem-part3; name this event 'division-theorem-part3-off'.

EVENT: Enable division-theorem-part2; name this event 'division-theorem-part2-off'.

EVENT: Enable division-theorem-part1; name this event 'division-theorem-part1-off'.

EVENT: Enable quotient-remainder-uniqueness; name this event 'quotient-remainder-uniqueness-off'.

EVENT: Enable itimes-1-arg1; name this event 'itimes-1-arg1-off'.

EVENT: Enable equal-itimes-minus-1; name this event 'equal-itimes-minus-1-off'.

EVENT: Enable equal-itimes-1; name this event 'equal-itimes-1-off'.

EVENT: Enable equal-itimes-0; name this event 'equal-itimes-0-off'.

EVENT: Enable associativity-of-itimes; name this event 'associativity-of-itimes-off'.

EVENT: Enable commutativity2-of-itimes; name this event 'commutativity2-of-itimes-off'.

EVENT: Enable itimes-distributes-over-iplus; name this event 'itimes-distributes-over-iplus-off'.

EVENT: Enable itimes-distributes-over-iplus-proof; name this event ‘itimes-distributes-over-iplus-proof-off’.

EVENT: Enable commutativity-of-itimes; name this event ‘commutativity-of-itimes-off’.

EVENT: Enable itimes-fix-int2; name this event ‘itimes-fix-int2-off’.

EVENT: Enable itimes-fix-int1; name this event ‘itimes-fix-int1-off’.

EVENT: Enable itimes-0-right; name this event ‘itimes-0-right-off’.

EVENT: Enable itimes-0-left; name this event ‘itimes-0-left-off’.

EVENT: Enable correctness-of-cancel-iplus-ilessp; name this event ‘correctness-of-cancel-iplus-ilessp-off’.

EVENT: Enable cancel-iplus-ilessp; name this event ‘cancel-iplus-ilessp-off’.

EVENT: Enable $*1*$ cancel-iplus-ilessp; name this event ‘ $g*1*$ cancel-iplus-ilessp-off’.

EVENT: Enable $*1*$ iplus-tree-no-fix-int; name this event ‘ $g*1*$ iplus-tree-no-fix-int-off’.

EVENT: Enable iplus-cancellation-2-for-ilessp; name this event ‘iplus-cancellation-2-for-ilessp-off’.

EVENT: Enable iplus-cancellation-1-for-ilessp; name this event ‘iplus-cancellation-1-for-ilessp-off’.

EVENT: Enable cancel-iplus-ilessp-1; name this event ‘cancel-iplus-ilessp-1-off’.

EVENT: Enable $*1*$ cancel-iplus-ilessp-1; name this event ‘ $g*1*$ cancel-iplus-ilessp-1-off’.

EVENT: Enable $*1*$ make-cancel-iplus-inequality-1; name this event ‘ $g*1*$ make-

cancel-iplus-inequality-1-off’.

EVENT: Enable ilssp-fix-int-2; name this event ‘ilssp-fix-int-2-off’.

EVENT: Enable ilssp-fix-int-1; name this event ‘ilssp-fix-int-1-off’.

EVENT: Enable correctness-of-cancel-iplus; name this event ‘correctness-of-cancel-iplus-off’.

EVENT: Enable iplus-ineg7; name this event ‘iplus-ineg7-off’.

EVENT: Enable iplus-list-append; name this event ‘iplus-list-append-off’.

EVENT: Enable eval\$-iplus-list-delete; name this event ‘eval\$-iplus-list-delete-off’.

EVENT: Enable *1*cancel-iplus; name this event ‘g*1*cancel-iplus-off’.

EVENT: Enable eval\$list-append; name this event ‘eval\$list-append-off’.

EVENT: Enable integerp-iplus-list; name this event ‘integerp-iplus-list-off’.

EVENT: Enable iplus-list; name this event ‘iplus-list-off’.

EVENT: Enable *1*iplus-list; name this event ‘g*1*iplus-list-off’.

EVENT: Enable *1*iplus-tree; name this event ‘g*1*iplus-tree-off’.

EVENT: Enable iplus-tree-rec; name this event ‘iplus-tree-rec-off’.

EVENT: Enable *1*iplus-tree-rec; name this event ‘g*1*iplus-tree-rec-off’.

EVENT: Enable iplus-fringe; name this event ‘iplus-fringe-off’.

EVENT: Enable *1*iplus-fringe; name this event ‘g*1*iplus-fringe-off’.

EVENT: Enable correctness-of-cancel-ineg; name this event ‘correctness-of-cancel-ineg-off’.

EVENT: Enable iplus-x-y-ineg-x; name this event ‘iplus-x-y-ineg-x-off’.

EVENT: Enable iplus-ineg4; name this event ‘iplus-ineg4-off’.

EVENT: Enable iplus-ineg3; name this event ‘iplus-ineg3-off’.

EVENT: Enable eval\$-cancel-ineg-aux-is-its-fn; name this event ‘eval\$-cancel-ineg-aux-is-its-fn-off’.

EVENT: Enable eval\$-cancel-ineg-aux-fn; name this event ‘eval\$-cancel-ineg-aux-fn-off’.

EVENT: Enable *1*eval\$-cancel-ineg-aux-fn; name this event ‘g*1*eval\$-cancel-ineg-aux-fn-off’.

EVENT: Enable eval\$-other; name this event ‘eval\$-other-off’.

EVENT: Enable eval\$-litatom; name this event ‘eval\$-litatom-off’.

EVENT: Enable eval\$-list-nlistp; name this event ‘eval\$-list-nlistp-off’.

EVENT: Enable eval\$-list-cons; name this event ‘eval\$-list-cons-off’.

EVENT: Enable cancel-ineg; name this event ‘cancel-ineg-off’.

EVENT: Enable *1*cancel-ineg; name this event ‘g*1*cancel-ineg-off’.

EVENT: Enable cancel-ineg-aux; name this event ‘cancel-ineg-aux-off’.

EVENT: Enable *1*cancel-ineg-aux; name this event ‘g*1*cancel-ineg-aux-off’.

EVENT: Enable idifference-fix-int2; name this event ‘idifference-fix-int2-off’.

EVENT: Enable idifference-fix-int1; name this event 'idifference-fix-int1-off'.

EVENT: Enable iplus-fix-int2; name this event 'iplus-fix-int2-off'.

EVENT: Enable iplus-fix-int1; name this event 'iplus-fix-int1-off'.

EVENT: Enable iplus-ineg2; name this event 'iplus-ineg2-off'.

EVENT: Enable iplus-ineg1; name this event 'iplus-ineg1-off'.

EVENT: Enable iplus-cancellation-2; name this event 'iplus-cancellation-2-off'.

EVENT: Enable iplus-cancellation-1; name this event 'iplus-cancellation-1-off'.

EVENT: Enable associativity-of-iplus; name this event 'associativity-of-iplus-off'.

EVENT: Enable commutativity-of-iplus; name this event 'commutativity-of-iplus-off'.

EVENT: Enable commutativity2-of-iplus; name this event 'commutativity2-of-iplus-off'.

EVENT: Enable iplus-0-right; name this event 'iplus-0-right-off'.

EVENT: Enable iplus-0-left; name this event 'iplus-0-left-off'.

EVENT: Enable ineg-0; name this event 'ineg-0-off'.

EVENT: Enable ineg-fix-int; name this event 'ineg-fix-int-off'.

EVENT: Enable ineg-ineg; name this event 'ineg-ineg-off'.

EVENT: Enable ineg-iplus; name this event 'ineg-iplus-off'.

EVENT: Enable fix-int-itimes; name this event 'fix-int-itimes-off'.

EVENT: Enable fix-int-iabs; name this event 'fix-int-iabs-off'.

EVENT: Enable fix-int-ineg; name this event 'fix-int-ineg-off'.

EVENT: Enable fix-int-idifference; name this event 'fix-int-idifference-off'.

EVENT: Enable fix-int-iplus; name this event 'fix-int-iplus-off'.

EVENT: Enable fix-int-fix-int; name this event 'fix-int-fix-int-off'.

EVENT: Enable fix-int-remover; name this event 'fix-int-remover-off'.

EVENT: Enable integerp-itimes; name this event 'integerp-itimes-off'.

EVENT: Enable integerp-iabs; name this event 'integerp-iabs-off'.

EVENT: Enable integerp-ineg; name this event 'integerp-ineg-off'.

EVENT: Enable integerp-idifference; name this event 'integerp-idifference-off'.

EVENT: Enable integerp-iplus; name this event 'integerp-iplus-off'.

EVENT: Enable integerp-fix-int; name this event 'integerp-fix-int-off'.

EVENT: Enable *1*irem; name this event 'g*1*irem-off'.

EVENT: Enable *1*iquote; name this event 'g*1*iquote-off'.

EVENT: Enable *1*imod; name this event 'g*1*imod-off'.

EVENT: Enable *1*idiv; name this event 'g*1*idiv-off'.

EVENT: Enable *1*iremainder; name this event 'g*1*iremainder-off'.

EVENT: Enable *1*iquote; name this event 'g*1*iquote-off'.

EVENT: Enable $*1*$ itimes; name this event ' $g*1*$ itimes-off'.

EVENT: Enable $*1*$ iabs; name this event ' $g*1*$ iabs-off'.

EVENT: Enable idifference; name this event 'idifference-off'.

EVENT: Enable $*1*$ idifference; name this event ' $g*1*$ idifference-off'.

EVENT: Enable $*1*$ ineg; name this event ' $g*1*$ ineg-off'.

EVENT: Enable $*1*$ iplus; name this event ' $g*1*$ iplus-off'.

EVENT: Enable ileq; name this event 'ileq-off'.

EVENT: Enable $*1*$ ileq; name this event ' $g*1*$ ileq-off'.

EVENT: Enable $*1*$ ilessp; name this event ' $g*1*$ ilessp-off'.

EVENT: Enable izerop; name this event 'izerop-off'.

EVENT: Enable $*1*$ izerop; name this event ' $g*1*$ izerop-off'.

EVENT: Enable $*1*$ fix-int; name this event ' $g*1*$ fix-int-off'.

EVENT: Enable $*1*$ integerp; name this event ' $g*1*$ integerp-off'.

EVENT: Enable gcd-idempotence; name this event 'gcd-idempotence-off'.

EVENT: Enable gcd-x-x; name this event 'gcd-x-x-off'.

EVENT: Enable commutativity2-of-gcd; name this event 'commutativity2-of-gcd-off'.

EVENT: Enable commutativity2-of-gcd-zero-case; name this event 'commutativity2-of-gcd-zero-case-off'.

EVENT: Enable associativity-of-gcd; name this event ‘associativity-of-gcd-off’.

EVENT: Enable associativity-of-gcd-zero-case; name this event ‘associativity-of-gcd-zero-case-off’.

EVENT: Enable common-divisor-divides-gcd; name this event ‘common-divisor-divides-gcd-off’.

EVENT: Enable gcd-is-the-greatest; name this event ‘gcd-is-the-greatest-off’.

EVENT: Enable distributivity-of-times-over-gcd; name this event ‘distributivity-of-times-over-gcd-off’.

EVENT: Enable distributivity-of-times-over-gcd-proof; name this event ‘distributivity-of-times-over-gcd-proof-off’.

EVENT: Enable remainder-gcd; name this event ‘remainder-gcd-off’.

EVENT: Enable gcd-plus-instance; name this event ‘gcd-plus-instance-off’.

EVENT: Enable gcd-plus; name this event ‘gcd-plus-off’.

EVENT: Enable gcd-plus-proof; name this event ‘gcd-plus-proof-off’.

EVENT: Enable gcd-plus-instance-temp; name this event ‘gcd-plus-instance-temp-off’.

EVENT: Enable gcd-plus-instance-temp-proof; name this event ‘gcd-plus-instance-temp-proof-off’.

EVENT: Enable lessp-gcd; name this event ‘lessp-gcd-off’.

EVENT: Enable equal-gcd-0; name this event ‘equal-gcd-0-off’.

EVENT: Enable gcd-1; name this event ‘gcd-1-off’.

EVENT: Enable gcd-0; name this event 'gcd-0-off'.

EVENT: Enable single-number-induction; name this event 'single-number-induction-off'.

EVENT: Enable *1*single-number-induction; name this event 'g*1*single-number-induction-off'.

EVENT: Enable commutativity-of-gcd; name this event 'commutativity-of-gcd-off'.

EVENT: Enable log-exp; name this event 'log-exp-off'.

EVENT: Enable log-times-exp; name this event 'log-times-exp-off'.

EVENT: Enable log-times-exp-proof; name this event 'log-times-exp-proof-off'.

EVENT: Enable log-times; name this event 'log-times-off'.

EVENT: Enable log-times-proof; name this event 'log-times-proof-off'.

EVENT: Enable log-quotient-exp; name this event 'log-quotient-exp-off'.

EVENT: Enable log-quotient-times; name this event 'log-quotient-times-off'.

EVENT: Enable log-quotient-times-proof; name this event 'log-quotient-times-proof-off'.

EVENT: Enable log-quotient; name this event 'log-quotient-off'.

EVENT: Enable leq-log-log; name this event 'leq-log-log-off'.

EVENT: Enable double-log-induction; name this event 'double-log-induction-off'.

EVENT: Enable *1*double-log-induction; name this event 'g*1*double-log-induction-off'.

off’.

EVENT: Enable log-1; name this event ‘log-1-off’.

EVENT: Enable log-0; name this event ‘log-0-off’.

EVENT: Enable equal-log-0; name this event ‘equal-log-0-off’.

EVENT: Enable exp-difference; name this event ‘exp-difference-off’.

EVENT: Enable equal-exp-1; name this event ‘equal-exp-1-off’.

EVENT: Enable equal-exp-0; name this event ‘equal-exp-0-off’.

EVENT: Enable exp-exp; name this event ‘exp-exp-off’.

EVENT: Enable exp-times; name this event ‘exp-times-off’.

EVENT: Enable exp-0-arg2; name this event ‘exp-0-arg2-off’.

EVENT: Enable exp-1-arg1; name this event ‘exp-1-arg1-off’.

EVENT: Enable exp-0-arg1; name this event ‘exp-0-arg1-off’.

EVENT: Enable exp-plus; name this event ‘exp-plus-off’.

EVENT: Enable exp-add1; name this event ‘exp-add1-off’.

EVENT: Enable exp-zero; name this event ‘exp-zero-off’.

EVENT: Enable quotient-exp; name this event ‘quotient-exp-off’.

EVENT: Enable remainder-exp-exp; name this event ‘remainder-exp-exp-off’.

EVENT: Enable double-number-induction; name this event ‘double-number-

induction-off’.

EVENT: Enable *1*double-number-induction; name this event ‘g*1*double-number-induction-off’.

EVENT: Enable remainder-exp; name this event ‘remainder-exp-off’.

EVENT: Enable gcd; name this event ‘gcd-off’.

EVENT: Enable *1*gcd; name this event ‘g*1*gcd-off’.

EVENT: Enable log; name this event ‘log-off’.

EVENT: Enable *1*log; name this event ‘g*1*log-off’.

EVENT: Enable exp; name this event ‘exp-off’.

EVENT: Enable *1*exp; name this event ‘g*1*exp-off’.

EVENT: Enable correctness-of-cancel-quotient-times; name this event ‘correctness-of-cancel-quotient-times-off’.

EVENT: Enable cancel-quotient-times; name this event ‘cancel-quotient-times-off’.

EVENT: Enable *1*cancel-quotient-times; name this event ‘g*1*cancel-quotient-times-off’.

EVENT: Enable lessp-quotient; name this event ‘lessp-quotient-off’.

EVENT: Enable quotient-x-x; name this event ‘quotient-x-x-off’.

EVENT: Enable quotient-1-arg1; name this event ‘quotient-1-arg1-off’.

EVENT: Enable quotient-1-arg1-casesplit; name this event ‘quotient-1-arg1-casesplit-off’.

EVENT: Enable quotient-1-arg2; name this event 'quotient-1-arg2-off'.

EVENT: Enable leq-quotient; name this event 'leq-quotient-off'.

EVENT: Enable quotient-quotient; name this event 'quotient-quotient-off'.

EVENT: Enable quotient-plus-times-times-instance; name this event 'quotient-plus-times-times-instance-off'.

EVENT: Enable quotient-plus-times-times; name this event 'quotient-plus-times-times-off'.

EVENT: Enable quotient-plus-times-times-proof; name this event 'quotient-plus-times-times-proof-off'.

EVENT: Enable quotient-plus-fact; name this event 'quotient-plus-fact-off'.

EVENT: Enable quotient-remainder-instance; name this event 'quotient-remainder-instance-off'.

EVENT: Enable quotient-remainder; name this event 'quotient-remainder-off'.

EVENT: Enable quotient-remainder-times; name this event 'quotient-remainder-times-off'.

EVENT: Enable quotient-difference3; name this event 'quotient-difference3-off'.

EVENT: Enable quotient-difference2; name this event 'quotient-difference2-off'.

EVENT: Enable quotient-lessp-arg1; name this event 'quotient-lessp-arg1-off'.

EVENT: Enable quotient-difference1; name this event 'quotient-difference1-off'.

EVENT: Enable quotient-times-times-proof; name this event 'quotient-times-times-proof-off'.

EVENT: Enable quotient-times-instance; name this event ‘quotient-times-instance-off’.

EVENT: Enable quotient-times; name this event ‘quotient-times-off’.

EVENT: Enable quotient-times-proof; name this event ‘quotient-times-proof-off’.

EVENT: Enable quotient-times-instance-temp-proof; name this event ‘quotient-times-instance-temp-proof-off’.

EVENT: Enable quotient-plus; name this event ‘quotient-plus-off’.

EVENT: Enable quotient-plus-proof; name this event ‘quotient-plus-proof-off’.

EVENT: Enable quotient-sub1; name this event ‘quotient-sub1-off’.

EVENT: Enable equal-quotient-0; name this event ‘equal-quotient-0-off’.

EVENT: Enable quotient-add1; name this event ‘quotient-add1-off’.

EVENT: Enable quotient-zero; name this event ‘quotient-zero-off’.

EVENT: Enable quotient-of-non-number; name this event ‘quotient-of-non-number-off’.

EVENT: Enable quotient-noop; name this event ‘quotient-noop-off’.

EVENT: Enable transitivity-of-divides; name this event ‘transitivity-of-divides-off’.

EVENT: Enable remainder-x-x; name this event ‘remainder-x-x-off’.

EVENT: Enable remainder-1-arg2; name this event ‘remainder-1-arg2-off’.

EVENT: Enable remainder-1-arg1; name this event ‘remainder-1-arg1-off’.

EVENT: Enable remainder-remainder; name this event ‘remainder-remainder-off’.

EVENT: Enable remainder-plus-times-times-instance; name this event ‘remainder-plus-times-times-instance-off’.

EVENT: Enable remainder-plus-times-times; name this event ‘remainder-plus-times-times-off’.

EVENT: Enable remainder-plus-times-times-proof; name this event ‘remainder-plus-times-times-proof-off’.

EVENT: Enable remainder-plus-fact; name this event ‘remainder-plus-fact-off’.

EVENT: Enable remainder-difference2; name this event ‘remainder-difference2-off’.

EVENT: Enable double-remainder-induction; name this event ‘double-remainder-induction-off’.

EVENT: Enable *1*double-remainder-induction; name this event ‘g*1*double-remainder-induction-off’.

EVENT: Enable remainder-difference1; name this event ‘remainder-difference1-off’.

EVENT: Enable remainder-times2-instance; name this event ‘remainder-times2-instance-off’.

EVENT: Enable remainder-times2; name this event ‘remainder-times2-off’.

EVENT: Enable remainder-times2-proof; name this event ‘remainder-times2-proof-off’.

EVENT: Enable remainder-times-times-proof; name this event ‘remainder-times-times-proof-off’.

EVENT: Enable remainder-times1-instance; name this event ‘remainder-times1-instance-off’.

EVENT: Enable remainder-times1-instance-proof; name this event ‘remainder-times1-instance-proof-off’.

EVENT: Enable remainder-times1; name this event ‘remainder-times1-off’.

EVENT: Enable remainder-times1-proof; name this event ‘remainder-times1-proof-off’.

EVENT: Enable equal-remainder-plus-remainder-proof; name this event ‘equal-remainder-plus-remainder-proof-off’.

EVENT: Enable equal-remainder-plus-0; name this event ‘equal-remainder-plus-0-off’.

EVENT: Enable equal-remainder-plus-0-proof; name this event ‘equal-remainder-plus-0-proof-off’.

EVENT: Enable remainder-plus; name this event ‘remainder-plus-off’.

EVENT: Enable remainder-plus-proof; name this event ‘remainder-plus-proof-off’.

EVENT: Enable remainder-add1; name this event ‘remainder-add1-off’.

EVENT: Enable remainder-quotient-elim; name this event ‘remainder-quotient-elim-off’.

EVENT: Enable remainder-zero; name this event ‘remainder-zero-off’.

EVENT: Enable remainder-of-non-number; name this event ‘remainder-of-non-number-off’.

EVENT: Enable remainder-noop; name this event ‘remainder-noop-off’.

EVENT: Enable lessp-remainder; name this event 'lessp-remainder-off'.

EVENT: Enable correctness-of-cancel-equal-times; name this event 'correctness-of-cancel-equal-times-off'.

EVENT: Enable cancel-equal-times; name this event 'cancel-equal-times-off'.

EVENT: Enable *1*cancel-equal-times; name this event 'g*1*cancel-equal-times-off'.

EVENT: Enable correctness-of-cancel-lessp-times; name this event 'correctness-of-cancel-lessp-times-off'.

EVENT: Enable cancel-lessp-times; name this event 'cancel-lessp-times-off'.

EVENT: Enable *1*cancel-lessp-times; name this event 'g*1*cancel-lessp-times-off'.

EVENT: Enable infer-equality-from-not-lessp; name this event 'infer-equality-from-not-lessp-off'.

EVENT: Enable lessp-times-arg1; name this event 'lessp-times-arg1-off'.

EVENT: Enable eval\$-equal; name this event 'eval\$-equal-off1'.

EVENT: Enable and-not-zerop-tree; name this event 'and-not-zerop-tree-off'.

EVENT: Enable *1*and-not-zerop-tree; name this event 'g*1*and-not-zerop-tree-off'.

EVENT: Enable or-zerop-tree; name this event 'or-zerop-tree-off'.

EVENT: Enable *1*or-zerop-tree; name this event 'g*1*or-zerop-tree-off'.

EVENT: Enable times-fringe; name this event 'times-fringe-off'.

EVENT: Enable $*1*$ times-fringe; name this event 'g*1*times-fringe-off'.

EVENT: Enable times-tree; name this event 'times-tree-off'.

EVENT: Enable $*1*$ times-tree; name this event 'g*1*times-tree-off'.

EVENT: Enable lessp-1-times; name this event 'lessp-1-times-off'.

EVENT: Enable lessp-plus-times2; name this event 'lessp-plus-times2-off'.

EVENT: Enable lessp-plus-times1; name this event 'lessp-plus-times1-off'.

EVENT: Enable lessp-plus-times-proof; name this event 'lessp-plus-times-proof-off'.

EVENT: Enable lessp-times-cancellation-proof; name this event 'lessp-times-cancellation-proof-off'.

EVENT: Enable lessp-times3; name this event 'lessp-times3-off'.

EVENT: Enable lessp-times3-proof2; name this event 'lessp-times3-proof2-off'.

EVENT: Enable lessp-times3-proof1; name this event 'lessp-times3-proof1-off'.

EVENT: Enable lessp-times2; name this event 'lessp-times2-off'.

EVENT: Enable lessp-times2-proof; name this event 'lessp-times2-proof-off'.

EVENT: Enable lessp-times1; name this event 'lessp-times1-off'.

EVENT: Enable lessp-times1-proof; name this event 'lessp-times1-proof-off'.

EVENT: Enable times-1-arg1; name this event 'times-1-arg1-off'.

EVENT: Enable times-quotient; name this event 'times-quotient-off'.

EVENT: Enable times-quotient-proof; name this event ‘times-quotient-proof-off’.

EVENT: Enable times-distributes-over-difference; name this event ‘times-distributes-over-difference-off’.

EVENT: Enable times-distributes-over-difference-proof; name this event ‘times-distributes-over-difference-proof-off’.

EVENT: Enable associativity-of-times; name this event ‘associativity-of-times-off’.

EVENT: Enable commutativity2-of-times; name this event ‘commutativity2-of-times-off’.

EVENT: Enable times-distributes-over-plus; name this event ‘times-distributes-over-plus-off’.

EVENT: Enable times-distributes-over-plus-proof; name this event ‘times-distributes-over-plus-proof-off’.

EVENT: Enable commutativity-of-times; name this event ‘commutativity-of-times-off’.

EVENT: Enable times-add1; name this event ‘times-add1-off’.

EVENT: Enable times-zero; name this event ‘times-zero-off’.

EVENT: Enable equal-sub1-0; name this event ‘equal-sub1-0-off’.

EVENT: Enable equal-times-1; name this event ‘equal-times-1-off’.

EVENT: Enable equal-times-0; name this event ‘equal-times-0-off’.

EVENT: Enable correctness-of-cancel-lessp-plus; name this event ‘correctness-of-cancel-lessp-plus-off’.

EVENT: Enable cancel-lessp-plus; name this event ‘cancel-lessp-plus-off’.

EVENT: Enable *1*cancel-lessp-plus; name this event ‘g*1*cancel-lessp-plus-off’.

EVENT: Enable difference-x-x; name this event ‘difference-x-x-off’.

EVENT: Enable difference-difference-arg2; name this event ‘difference-difference-arg2-off’.

EVENT: Enable difference-difference-arg1; name this event ‘difference-difference-arg1-off’.

EVENT: Enable difference-sub1-arg2; name this event ‘difference-sub1-arg2-off’.

EVENT: Enable difference-add1-arg2; name this event ‘difference-add1-arg2-off’.

EVENT: Enable difference-leq-arg1; name this event ‘difference-leq-arg1-off’.

EVENT: Enable difference-elim; name this event ‘difference-elim-off’.

EVENT: Enable correctness-of-cancel-difference-plus; name this event ‘correctness-of-cancel-difference-plus-off’.

EVENT: Enable cancel-difference-plus; name this event ‘cancel-difference-plus-off’.

EVENT: Enable *1*cancel-difference-plus; name this event ‘g*1*cancel-difference-plus-off’.

EVENT: Enable correctness-of-cancel-equal-plus; name this event ‘correctness-of-cancel-equal-plus-off’.

EVENT: Enable cancel-equal-plus; name this event ‘cancel-equal-plus-off’.

EVENT: Enable *1*cancel-equal-plus; name this event ‘g*1*cancel-equal-plus-off’.

EVENT: Enable eval\$-quote; name this event 'eval\$-quote-off1'.

EVENT: Enable plus-tree; name this event 'plus-tree-off'.

EVENT: Enable *1*plus-tree; name this event 'g*1*plus-tree-off'.

EVENT: Enable plus-fringe; name this event 'plus-fringe-off'.

EVENT: Enable *1*plus-fringe; name this event 'g*1*plus-fringe-off'.

EVENT: Enable diff-diff-arg2; name this event 'diff-diff-arg2-off'.

EVENT: Enable diff-diff-arg1; name this event 'diff-diff-arg1-off'.

EVENT: Enable difference-plus-plus-cancellation-proof; name this event 'difference-plus-plus-cancellation-proof-off'.

EVENT: Enable difference-plus-cancellation-proof; name this event 'difference-plus-cancellation-proof-off'.

EVENT: Enable plus-difference-arg2; name this event 'plus-difference-arg2-off'.

EVENT: Enable plus-difference-arg1; name this event 'plus-difference-arg1-off'.

EVENT: Enable associativity-of-plus; name this event 'associativity-of-plus-off'.

EVENT: Enable plus-add1-arg2; name this event 'plus-add1-arg2-off'.

EVENT: Enable plus-add1-arg1; name this event 'plus-add1-arg1-off'.

EVENT: Enable plus-zero-arg2; name this event 'plus-zero-arg2-off'.

EVENT: Enable commutativity2-of-plus; name this event 'commutativity2-of-plus-off'.

EVENT: Enable commutativity-of-plus; name this event 'commutativity-of-plus-off'.

off’.

EVENT: Enable equal-difference-0; name this event ‘equal-difference-0-off’.

EVENT: Enable equal-plus-0; name this event ‘equal-plus-0-off’.

EVENT: Enable member-bagint; name this event ‘member-bagint-off’.

EVENT: Enable occurrences-bagdiff; name this event ‘occurrences-bagdiff-off’.

EVENT: Enable occurrences-bagint; name this event ‘occurrences-bagint-off’.

EVENT: Enable subbagp-bagint2; name this event ‘subbagp-bagint2-off’.

EVENT: Enable subbagp-bagint1; name this event ‘subbagp-bagint1-off’.

EVENT: Enable subbagp-cdr2; name this event ‘subbagp-cdr2-off’.

EVENT: Enable subbagp-cdr1; name this event ‘subbagp-cdr1-off’.

EVENT: Enable subbagp-delete; name this event ‘subbagp-delete-off’.

EVENT: Enable bagdiff-delete; name this event ‘bagdiff-delete-off’.

EVENT: Enable member-bagdiff; name this event ‘member-bagdiff-off’.

EVENT: Enable occurrences-delete; name this event ‘occurrences-delete-off’.

EVENT: Enable member-delete-implies-membership; name this event ‘member-delete-implies-membership-off’.

EVENT: Enable member-delete; name this event ‘member-delete-off’.

EVENT: Enable member-non-list; name this event ‘member-non-list-off’.

EVENT: Enable equal-occurrences-zero; name this event ‘equal-occurrences-zero-off’.

EVENT: Enable delete-delete; name this event ‘delete-delete-off’.

EVENT: Enable delete-non-member; name this event ‘delete-non-member-off’.

EVENT: Enable subbagp; name this event ‘subbagp-off’.

EVENT: Enable *1*subbagp; name this event ‘g*1*subbagp-off’.

EVENT: Enable occurrences; name this event ‘occurrences-off’.

EVENT: Enable *1*occurrences; name this event ‘g*1*occurrences-off’.

EVENT: Enable bagint; name this event ‘bagint-off’.

EVENT: Enable *1*bagint; name this event ‘g*1*bagint-off’.

EVENT: Enable bagdiff; name this event ‘bagdiff-off’.

EVENT: Enable *1*bagdiff; name this event ‘g*1*bagdiff-off’.

EVENT: Enable delete; name this event ‘delete-off’.

EVENT: Enable *1*delete; name this event ‘g*1*delete-off’.

DEFINITION:
 $\text{if}^*(a, b, c)$
 $= \text{if } a \text{ then } b$
 $\quad \text{else } c \text{ endif}$

EVENT: Enable if*; name this event ‘if*-off’.

DEFINITION: $\text{or}^*(a, b) = \text{if}^*(a, \text{t}, \text{if}^*(b, \text{t}, \text{f}))$

EVENT: Enable or*; name this event ‘or*-off’.

DEFINITION: $\text{and}^*(a, b) = \text{if}^*(a, \text{if}^*(b, \mathbf{t}, \mathbf{f}), \mathbf{f})$

EVENT: Enable and^* ; name this event ‘ and^* -off’.

DEFINITION: $\text{not}^*(a) = \text{if}^*(a, \mathbf{f}, \mathbf{t})$

EVENT: Enable not^* ; name this event ‘ not^* -off’.

EVENT: Let us define the theory *prop^{*}-theory* to consist of the following events:
 or^* , and^* , not^* .

THEOREM: simplify- if^*

$$(c \rightarrow (\text{if}^*(c, a, b) = a)) \wedge ((\neg c) \rightarrow (\text{if}^*(c, a, b) = b))$$

THEOREM: if^* -c-x-x

$$\text{if}^*(c, x, x) = x$$

THEOREM: if^* -cons

$$\text{if}^*(a, \text{cons}(b, c), \text{cons}(d, e)) = \text{cons}(\text{if}^*(a, b, d), \text{if}^*(a, c, e))$$

THEOREM: rewrite- not^*

$$(x \rightarrow (\text{not}^*(x) = \mathbf{f})) \wedge ((\neg x) \rightarrow (\text{not}^*(x) = \mathbf{t}))$$

THEOREM: rewrite- and^*

$$\begin{aligned} &(\text{and}^*(\mathbf{f}, x) = \mathbf{f}) \\ \wedge &(\text{and}^*(x, \mathbf{f}) = \mathbf{f}) \\ \wedge &(y \rightarrow ((\text{and}^*(y, x) = \text{if}^*(x, \mathbf{t}, \mathbf{f})) \wedge (\text{and}^*(x, y) = \text{if}^*(x, \mathbf{t}, \mathbf{f})))) \end{aligned}$$

THEOREM: rewrite- or^*

$$\begin{aligned} &(\text{or}^*(\mathbf{f}, x) = \text{if}^*(x, \mathbf{t}, \mathbf{f})) \\ \wedge &(\text{or}^*(x, \mathbf{f}) = \text{if}^*(x, \mathbf{t}, \mathbf{f})) \\ \wedge &(y \rightarrow ((\text{or}^*(y, x) = \mathbf{t}) \wedge (\text{or}^*(x, y) = \mathbf{t}))) \end{aligned}$$

THEOREM: expand- or^* -connectives

$$\begin{aligned} &(\text{if}^*(c, a, b) \\ &= \text{if } c \text{ then } a \\ &\quad \text{else } b \text{ endif}) \\ \wedge &(\text{or}^*(a, b) \\ &= \text{if } a \text{ then } \mathbf{t} \\ &\quad \text{elseif } b \text{ then } \mathbf{t} \\ &\quad \text{else } \mathbf{f} \text{ endif}) \\ \wedge &(\text{and}^*(a, b) \\ &= \text{if } a \\ &\quad \text{then if } b \text{ then } \mathbf{t} \end{aligned}$$

else f endif
else f endif
 $\wedge \quad (\text{not}^*(a) = (\neg a))$

EVENT: Enable expand-*-connectives; name this event ‘expand-*-connectives-off’.

THEOREM: car-cdr-if-cons
 $(\text{car}(\text{if } x \text{ then cons}(a, b)$
 $\quad \text{else cons}(c, d) \text{ endif})$
 $= \text{if } x \text{ then } a$
 $\quad \text{else } c \text{ endif})$
 $\wedge \quad (\text{cdr}(\text{if } x \text{ then cons}(a, b)$
 $\quad \text{else cons}(c, d) \text{ endif})$
 $= \text{if } x \text{ then } b$
 $\quad \text{else } d \text{ endif})$

THEOREM: our-car-cdr-elim
 $(l = \text{cons}(\text{car}(l), x)) = (\text{listp}(l) \wedge (x = \text{cdr}(l)))$

EVENT: Enable our-car-cdr-elim; name this event ‘our-car-cdr-elim-off’.

EVENT: Enable plus; name this event ‘plus-off’.

THEOREM: plus-0
 $((\text{zero} \simeq 0) \rightarrow ((x + \text{zero}) = \text{fix}(x)))$
 $\wedge \quad ((\text{zero} \simeq 0) \rightarrow ((\text{zero} + x) = \text{fix}(x)))$

THEOREM: plus-add1
 $(((1 + x) + y) = (1 + (x + y)))$
 $\wedge \quad ((x + (1 + y)) = (1 + (x + y)))$

THEOREM: plus-bottom
 $((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: plus-add1-sub1
 $((n \not\simeq 0) \rightarrow (((n - 1) + (1 + m)) = (n + m)))$
 $\wedge \quad ((n \not\simeq 0) \rightarrow (((1 + m) + (n - 1)) = (m + n)))$

EVENT: Enable difference; name this event ‘difference-off’.

THEOREM: difference-x-1
 $(x - 1) = (x - 1)$

THEOREM: our-difference-x-x

$$(x - x) = 0$$

THEOREM: our-equal-difference-0

$$\begin{aligned} & ((x - y) = 0) \\ = & \text{ if } y < x \text{ then f} \\ & \text{ else t endif} \end{aligned}$$

THEOREM: difference-0

$$(z \simeq 0) \rightarrow ((x - z) = \text{fix}(x))$$

THEOREM: difference-add1-add1

$$((1 + x) - (1 + y)) = (x - y)$$

THEOREM: not-lessp-difference

$$(x < (x - y)) = \mathbf{f}$$

THEOREM: lessp-difference=0

$$\begin{aligned} & \text{ if } y < x \text{ then f} \\ & \text{ else t endif} \\ \rightarrow & ((x - y) = 0) \end{aligned}$$

THEOREM: times-bottom

$$((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$$

THEOREM: times-1

$$((a * 1) = \text{fix}(a)) \wedge ((1 * a) = \text{fix}(a))$$

THEOREM: times-add1-again

$$(a * (1 + b)) = (a + (a * b))$$

THEOREM: times-commutes

$$(a * b) = (b * a)$$

THEOREM: quotient-lessp

$$((1 < \text{base}) \wedge (n \not\simeq 0)) \rightarrow ((n \div \text{base}) < n)$$

THEOREM: zerop-quotient

$$((n \div m) = 0) = ((n \simeq 0) \vee (m \simeq 0) \vee (n < m))$$

THEOREM: lessp-x-x

$$(x < x) = \mathbf{f}$$

THEOREM: lessp-sub1-x-y-crock

$$(x = y) \rightarrow (((x - 1) < y) = (x \not\simeq 0))$$

THEOREM: lessp-sub1-x-x
 $((x - 1) < x) = (x \not\succeq 0)$

THEOREM: lessp-x-1
 $(x < 1) = (x \simeq 0)$

THEOREM: leq-lessp-difference
(if $y < x$ **then f**
else t endif
 $\wedge (z \not\succeq 0)$
 $\wedge (y \not\succeq 0))$
 $\rightarrow (((x - z) < y) = \mathbf{t})$

EVENT: Enable append; name this event ‘append-off’.

DEFINITION:
 $\text{append5}(a, b, c, d, e) = \text{append}(a, \text{append}(b, \text{append}(c, \text{append}(d, e))))$

DEFINITION:
 $\text{append6}(a, b, c, d, e, g)$
 $= \text{append}(a, \text{append}(b, \text{append}(c, \text{append}(d, \text{append}(e, g)))))$

DEFINITION:
 $\text{append7}(a, b, c, d, e, g, h)$
 $= \text{append}(a, \text{append}(b, \text{append}(c, \text{append}(d, \text{append}(e, \text{append}(g, h)))))$

DEFINITION:
 $\text{append8}(a, b, c, d, e, g, h, i)$
 $= \text{append}(a,$
 $\quad \text{append}(b, \text{append}(c, \text{append}(d, \text{append}(e, \text{append}(g, \text{append}(h, i)))))$

THEOREM: append-nlistp
 $(a \simeq \mathbf{nil}) \rightarrow (\text{append}(a, b) = b)$

THEOREM: append-cons
 $\text{append}(\text{cons}(a, b), c) = \text{cons}(a, \text{append}(b, c))$

THEOREM: our-member-append
 $(a \in \text{append}(b, c)) = ((a \in b) \vee (a \in c))$

THEOREM: associativity-of-append
 $\text{append}(\text{append}(a, b), c) = \text{append}(a, \text{append}(b, c))$

THEOREM: equal-append-x-x
 $(\text{append}(a, b) = \text{append}(a, c)) = (b = c)$

DEFINITION:

$\text{properp}(l)$
= **if** $\text{listp}(l)$ **then** $\text{properp}(\text{cdr}(l))$
 else $l = \text{nil}$ **endif**

EVENT: Enable properp; name this event ‘properp-off’.

THEOREM: properp-if
 $(\text{properp}(a) \wedge \text{properp}(b))$
 $\rightarrow \text{properp}(\text{if } c \text{ then } a$
 else b **endif**)

THEOREM: properp-nlistp
 $(l \simeq \text{nil}) \rightarrow (\text{properp}(l) = (l = \text{nil}))$

THEOREM: properp-cons
 $\text{properp}(\text{cons}(x, y)) = \text{properp}(y)$

THEOREM: properp-append
 $\text{properp}(\text{append}(a, b)) = \text{properp}(b)$

THEOREM: properp-append-nil
 $\text{properp}(a) \rightarrow (\text{append}(a, \text{nil}) = a)$

THEOREM: properp-if*
 $(\text{properp}(a) \wedge \text{properp}(b)) \rightarrow \text{properp}(\text{if}^*(c, a, b))$

DEFINITION:

$\text{position}(x, l)$
= **if** $l \simeq \text{nil}$ **then** f
 elseif $x = \text{car}(l)$ **then** 0
 elseif $\text{position}(x, \text{cdr}(l))$ **then** $1 + \text{position}(x, \text{cdr}(l))$
 else f **endif**

EVENT: Enable position; name this event ‘position-off’.

THEOREM: member==>position
 $(x \in l) \rightarrow \text{position}(x, l)$

DEFINITION:

$\text{length}(l)$
= **if** $\text{listp}(l)$ **then** $1 + \text{length}(\text{cdr}(l))$
 else 0 **endif**

EVENT: Enable length; name this event ‘length-off’.

THEOREM: length-nlistp
 $(l \simeq \mathbf{nil}) \rightarrow (\text{length}(l) = 0)$

THEOREM: length-bottom
 $(\text{length}(x) = 0) = (x \simeq \mathbf{nil})$

THEOREM: length-1
 $(\text{length}(a) = 1) = (\text{listp}(a) \wedge (\text{cdr}(a) \simeq \mathbf{nil}))$

THEOREM: length-cons
 $\text{length}(\text{cons}(x, y)) = (1 + \text{length}(y))$

THEOREM: length-append
 $\text{length}(\text{append}(a, b)) = (\text{length}(a) + \text{length}(b))$

THEOREM: length-cdr-lemmas
 $((\text{length}(\text{cdr}(x)) = \text{length}(x)) = (x \simeq \mathbf{nil}))$
 $\wedge ((\text{length}(\text{cdr}(x)) < \text{length}(x)) = \text{listp}(x))$

THEOREM: equal-length-add1
 $(\text{length}(x) = (1 + y)) = (\text{listp}(x) \wedge (\text{length}(\text{cdr}(x)) = \text{fix}(y)))$

EVENT: Enable equal-length-add1; name this event ‘equal-length-add1-off’.

THEOREM: length-cdr
 $(\text{length}(\text{cdr}(v)) = (\text{length}(v) - 1)) = \mathbf{t}$

THEOREM: equal-length-cdr
 $(\text{length}(a) = \text{length}(b)) \rightarrow ((\text{length}(\text{cdr}(a)) = \text{length}(\text{cdr}(b))) = \mathbf{t})$

THEOREM: length-if
 $(\text{length}(a) = \text{length}(b))$
 $\rightarrow (\text{length}(\mathbf{if } c \mathbf{ then } a$
 $\qquad \qquad \mathbf{else } b \mathbf{ endif})$
 $\qquad = \text{length}(a))$

THEOREM: length-if*
 $(\text{length}(a) = \text{length}(b)) \rightarrow (\text{length}(\text{if}^*(c, a, b)) = \text{length}(a))$

DEFINITION:
 $\text{rev1}(x, \text{ sponge})$
 $= \mathbf{if } x \simeq \mathbf{nil} \mathbf{ then } \text{ sponge}$
 $\qquad \mathbf{else } \text{rev1}(\text{cdr}(x), \text{cons}(\text{car}(x), \text{ sponge})) \mathbf{ endif}$

EVENT: Enable rev1; name this event ‘rev1-off’.

DEFINITION: $\text{reverse}(x) = \text{rev1}(x, \mathbf{nil})$

EVENT: Enable reverse; name this event ‘reverse-off’.

THEOREM: length-rev1

$$\text{length}(\text{rev1}(x, \text{sponge})) = (\text{length}(x) + \text{length}(\text{sponge}))$$

THEOREM: length-reverse

$$\text{length}(\text{reverse}(x)) = \text{length}(x)$$

DEFINITION:

$\text{delete}^*(x, l)$

= **if** $l \simeq \mathbf{nil}$ **then** l
 elseif $x = \text{car}(l)$ **then** $\text{delete}^*(x, \text{cdr}(l))$
 else $\text{cons}(\text{car}(l), \text{delete}^*(x, \text{cdr}(l)))$ **endif**

EVENT: Enable delete*; name this event ‘delete*-off’.

DEFINITION:

$\text{subset}(l1, l2)$

= **if** $l1 \simeq \mathbf{nil}$ **then** **t**
 else $(\text{car}(l1) \in l2) \wedge \text{subset}(\text{cdr}(l1), l2)$ **endif**

EVENT: Enable subset; name this event ‘subset-off’.

THEOREM: subset-nlistp

$$(x \simeq \mathbf{nil}) \rightarrow \text{subset}(x, y)$$

THEOREM: subset-cons

$$\text{subset}(\text{cons}(x, y), z) = ((x \in z) \wedge \text{subset}(y, z))$$

THEOREM: subset-x-cons-y-z

$$\begin{aligned} &\text{subset}(x, \text{cons}(y, z)) \\ &= (\text{subset}(x, z) \vee ((y \in x) \wedge \text{subset}(\text{delete}^*(y, x), z))) \end{aligned}$$

THEOREM: subset-append

$$(\text{subset}(a, b) \vee \text{subset}(a, c)) \rightarrow \text{subset}(a, \text{append}(b, c))$$

THEOREM: subset-x-x

$$\text{subset}(x, x)$$

DEFINITION:

$\text{disjoint}(l1, l2)$

= **if** $l1 \simeq \mathbf{nil}$ **then** **t**
 else $(\text{car}(l1) \notin l2) \wedge \text{disjoint}(\text{cdr}(l1), l2)$ **endif**

EVENT: Enable disjoint; name this event ‘disjoint-off’.

THEOREM: disjoint-nlistp
 $((x \simeq \mathbf{nil}) \vee (y \simeq \mathbf{nil})) \rightarrow \text{disjoint}(x, y)$

THEOREM: disjoint-cons
 $(\text{disjoint}(\text{cons}(x, y), z) = ((x \notin z) \wedge \text{disjoint}(y, z)))$
 $\wedge (\text{disjoint}(z, \text{cons}(x, y)) = ((x \notin z) \wedge \text{disjoint}(z, y)))$

THEOREM: disjoint-append
 $(\text{disjoint}(x, \text{append}(y, z)) = (\text{disjoint}(x, y) \wedge \text{disjoint}(x, z)))$
 $\wedge (\text{disjoint}(\text{append}(y, z), x) = (\text{disjoint}(y, x) \wedge \text{disjoint}(z, x)))$

DEFINITION:
 $\text{duplicates?}(l)$
 $=$ **if** $l \simeq \mathbf{nil}$ **then f**
 else $(\text{car}(l) \in \text{cdr}(l)) \vee \text{duplicates?}(\text{cdr}(l))$ **endif**

EVENT: Enable duplicates?; name this event ‘duplicates?-off’.

THEOREM: duplicates?-cons
 $\text{duplicates?}(\text{cons}(x, y)) = ((x \in y) \vee \text{duplicates?}(y))$

THEOREM: duplicates-append
 $\text{duplicates?}(\text{append}(a, b))$
 $= (\text{duplicates?}(a) \vee \text{duplicates?}(b) \vee (\neg \text{disjoint}(a, b)))$

DEFINITION:
 $\text{firstn}(n, l)$
 $=$ **if** $\text{listp}(l)$
 then if $n \simeq 0$ **then nil**
 else $\text{cons}(\text{car}(l), \text{firstn}(n - 1, \text{cdr}(l)))$ **endif**
 else nil **endif**

EVENT: Enable firstn; name this event ‘firstn-off’.

THEOREM: length-firstn
 $\text{length}(\text{firstn}(n, l))$
 $=$ **if if** $\text{length}(l) < n$ **then f**
 else t **endif then** $\text{fix}(n)$
 else $\text{length}(l)$ **endif**

THEOREM: length-firstn1
if $\text{length}(l) < n$ **then f**
 else t **endif**
 $\rightarrow (\text{length}(\text{firstn}(n, l)) = \text{fix}(n))$

THEOREM: length-firstn2
 $(\neg \text{if length}(l) < n \text{ then } f$
 $\quad \text{else } t \text{ endif})$
 $\rightarrow (\text{length}(\text{firstn}(n, l)) = \text{length}(l))$

THEOREM: car-firstn
 $\text{car}(\text{firstn}(n, a))$
 $= \text{if listp}(a) \wedge (n \neq 0) \text{ then } \text{car}(a)$
 $\quad \text{else } 0 \text{ endif}$

THEOREM: not-member-firstn
 $(x \notin y) \rightarrow (x \notin \text{firstn}(n, y))$

THEOREM: not-duplicates?-firstn
 $(\neg \text{duplicates?}(x)) \rightarrow (\neg \text{duplicates?}(\text{firstn}(n, x)))$

THEOREM: subset-firstn
 $\text{subset}(\text{firstn}(n, l), l)$

THEOREM: firstn-bottom
 $(\text{firstn}(n, l) = \mathbf{nil}) = ((l \simeq \mathbf{nil}) \vee (n \simeq 0))$

THEOREM: disjoint-firstn
 $\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{firstn}(n, x), y)$

THEOREM: disjoint-firstn1
 $\text{disjoint}(y, x) \rightarrow \text{disjoint}(y, \text{firstn}(n, x))$

THEOREM: properp-firstn
 $\text{properp}(\text{firstn}(n, l))$

THEOREM: firstn-append
 $\text{firstn}(n, \text{append}(a, b)) = \text{append}(\text{firstn}(n, a), \text{firstn}(n - \text{length}(a), b))$

DEFINITION:
 $\text{restn}(n, l)$
 $= \text{if listp}(l)$
 $\quad \text{then if } n \simeq 0 \text{ then } l$
 $\quad \quad \text{else } \text{restn}(n - 1, \text{cdr}(l)) \text{ endif}$
 $\quad \text{else } l \text{ endif}$

EVENT: Enable restn; name this event 'restn-off'.

THEOREM: length-restn
 $\text{length}(\text{restn}(n, l)) = (\text{length}(l) - n)$

THEOREM: not-duplicates?-restn
 $(\neg \text{duplicates?}(x)) \rightarrow (\neg \text{duplicates?}(\text{restn}(n, x)))$

THEOREM: not-member-restn
 $(x \notin y) \rightarrow (x \notin \text{restn}(n, y))$

THEOREM: subset-restn
 $\text{subset}(\text{restn}(n, l), l)$

THEOREM: disjoint-restn
 $\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{restn}(n, x), y)$

THEOREM: disjoint-restn1
 $\text{disjoint}(y, x) \rightarrow \text{disjoint}(y, \text{restn}(n, x))$

THEOREM: properp-restn
 $\text{properp}(\text{restn}(n, l)) = \text{properp}(l)$

THEOREM: restn-append
 $\text{restn}(n, \text{append}(a, b)) = \text{append}(\text{restn}(n, a), \text{restn}(n - \text{length}(a), b))$

THEOREM: too-many-restns
 $(\text{properp}(l) \wedge \text{if } n < \text{length}(l) \text{ then f else t endif}) \rightarrow (\text{restn}(n, l) = \text{nil})$

THEOREM: cdr-restn
 $(n < \text{length}(v)) \rightarrow (\text{cdr}(\text{restn}(n, v)) = \text{restn}(1 + n, v))$

EVENT: Enable cdr-restn; name this event 'cdr-restn-off'.

THEOREM: disjoint-firstn-restn-lemmas
 $(\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{firstn}(n, x), \text{firstn}(m, y)))$
 $\wedge (\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{firstn}(n, x), \text{restn}(m, y)))$
 $\wedge (\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{restn}(n, x), \text{firstn}(m, y)))$
 $\wedge (\text{disjoint}(x, y) \rightarrow \text{disjoint}(\text{restn}(n, x), \text{restn}(m, y)))$

THEOREM: no-duplicates-disjoint-firstn-restn
 $((\neg \text{duplicates?}(x)) \rightarrow \text{disjoint}(\text{firstn}(n, x), \text{restn}(n, x)))$
 $\wedge ((\neg \text{duplicates?}(x)) \rightarrow \text{disjoint}(\text{restn}(n, x), \text{firstn}(n, x)))$

THEOREM: append-firstn-restn
 $\text{append}(\text{firstn}(n, l), \text{restn}(n, l)) = l$

EVENT: Enable pairlist; name this event 'pairlist-off'.

THEOREM: pairlist-nlistp
 $(a \simeq \mathbf{nil}) \rightarrow (\text{pairlist}(a, b) = \mathbf{nil})$

THEOREM: pairlist-cons
 $\text{pairlist}(\text{cons}(x, y), z) = \text{cons}(\text{cons}(x, \text{car}(z)), \text{pairlist}(y, \text{cdr}(z)))$

THEOREM: pairlist-append
 $(\text{length}(a) = \text{length}(c))$
 $\rightarrow (\text{pairlist}(\text{append}(a, b), \text{append}(c, d))$
 $\quad = \text{append}(\text{pairlist}(a, c), \text{pairlist}(b, d)))$

THEOREM: properp-pairlist
 $\text{properp}(\text{pairlist}(x, y))$

THEOREM: pairlists-are-equal-when-their-2nd-lists-are-nlistp
 $((l1 \simeq \mathbf{nil}) \wedge (l2 \simeq \mathbf{nil})) \rightarrow ((\text{pairlist}(a, l1) = \text{pairlist}(a, l2)) = \mathbf{t})$

DEFINITION:
 $\text{nth}(n, \text{list})$
 $= \text{if } n \simeq 0 \text{ then } \text{car}(\text{list})$
 $\quad \text{else } \text{nth}(n - 1, \text{cdr}(\text{list})) \text{ endif}$

EVENT: Enable nth; name this event ‘nth-off’.

THEOREM: open-nth
 $((n \simeq 0) \rightarrow (\text{nth}(n, \text{list}) = \text{car}(\text{list})))$
 $\wedge ((n \not\simeq 0) \rightarrow (\text{nth}(n, \text{list}) = \text{nth}(n - 1, \text{cdr}(\text{list}))))$

EVENT: Enable open-nth; name this event ‘open-nth-off’.

THEOREM: nth-if
 $\text{nth}(n,$
 $\quad \text{if } a \text{ then } b$
 $\quad \text{else } c \text{ endif})$
 $= \text{if } a \text{ then } \text{nth}(n, b)$
 $\quad \text{else } \text{nth}(n, c) \text{ endif}$

EVENT: Enable nth-if; name this event ‘nth-if-off’.

THEOREM: nth-restn
 $\text{nth}(n, v) = \text{car}(\text{restn}(n, v))$

EVENT: Enable nth-restn; name this event ‘nth-restn-off’.

THEOREM: nth-append
 $(n < \text{length}(a)) \rightarrow (\text{nth}(n, \text{append}(a, b)) = \text{nth}(n, a))$

THEOREM: nth-append-too
if $n < \text{length}(a)$ **then f**
else t endif
 $\rightarrow (\text{nth}(n, \text{append}(a, b)) = \text{nth}(n - \text{length}(a), b))$

DEFINITION:
 $\text{nthcdr}(n, l)$
 $=$ **if** $n \simeq 0$ **then** l
else $\text{nthcdr}(n - 1, \text{cdr}(l))$ **endif**

EVENT: Enable nthcdr; name this event ‘nthcdr-off’.

THEOREM: open-nthcdr
 $((n \simeq 0) \rightarrow (\text{nthcdr}(n, l) = l))$
 $\wedge ((n \not\simeq 0) \rightarrow (\text{nthcdr}(n, l) = \text{nthcdr}(n - 1, \text{cdr}(l))))$

THEOREM: properp-as-null-nthcdr
 $\text{properp}(l) = (\text{nthcdr}(\text{length}(l), l) = \text{nil})$

EVENT: Enable properp-as-null-nthcdr; name this event ‘properp-as-null-nthcdr-off’.

THEOREM: cdr-nthcdr
 $\text{cdr}(\text{nthcdr}(n, l)) = \text{nthcdr}(1 + n, l)$

EVENT: Enable cdr-nthcdr; name this event ‘cdr-nthcdr-off’.

THEOREM: listp-nthcdr
 $\text{listp}(\text{nthcdr}(n, l)) = (n < \text{length}(l))$

DEFINITION:
 $\text{subrange}(l, n, m)$
 $=$ **if** $m < n$ **then** **nil**
elseif $n \simeq 0$
then if $m \simeq 0$ **then** $\text{list}(\text{car}(l))$
else $\text{cons}(\text{car}(l), \text{subrange}(\text{cdr}(l), 0, m - 1))$ **endif**
else $\text{subrange}(\text{cdr}(l), n - 1, m - 1)$ **endif**

EVENT: Enable subrange; name this event ‘subrange-off’.

THEOREM: properp-subrange
 $\text{properp}(\text{subrange}(v, n, m))$

THEOREM: subrange-cons
 $\text{subrange}(\text{cons}(car, cdr), n, m)$
 $=$ **if** $m < n$ **then** **nil**
 elseif $n \simeq 0$
 then if $m \simeq 0$ **then** $\text{list}(car)$
 else $\text{cons}(car, \text{subrange}(cdr, 0, m - 1))$ **endif**
 else $\text{subrange}(cdr, n - 1, m - 1)$ **endif**

EVENT: Enable subrange-cons; name this event ‘subrange-cons-off’.

THEOREM: length-subrange
 $\text{length}(\text{subrange}(l, n, m))$
 $=$ **if** $m < n$ **then** 0
 else $1 + (m - n)$ **endif**

THEOREM: subrange-append-right
 $((m \not< n)$
 $\wedge$ **if** $n < \text{length}(a)$ **then** **f**
 else **t** **endif**)
 $\rightarrow$ $(\text{subrange}(\text{append}(a, b), n, m)$
 $= \text{subrange}(b, n - \text{length}(a), m - \text{length}(a)))$

THEOREM: subrange-append-left
 $((m \not< n) \wedge (n < \text{length}(a)) \wedge (m < \text{length}(a)))$
 $\rightarrow (\text{subrange}(\text{append}(a, b), n, m) = \text{subrange}(a, n, m))$

THEOREM: subrange-0
 $((m = (\text{length}(a) - 1)) \wedge \text{properp}(a) \wedge (\text{length}(a) \neq 0))$
 $\rightarrow (\text{subrange}(a, 0, m) = a)$

THEOREM: listp-subrange
 $\text{listp}(\text{subrange}(l, n, m)) = (m \not< n)$

THEOREM: subset-subrange
 $(n < \text{length}(v)) \rightarrow \text{subset}(\text{subrange}(v, m, n), v)$

THEOREM: not-member-subrange
 $((n < \text{length}(v)) \wedge (x \notin v)) \rightarrow (x \notin \text{subrange}(v, m, n))$

THEOREM: disjoint-subrange
 $(\text{disjoint}(x, y) \wedge (m < \text{length}(y)))$
 $\rightarrow (\text{disjoint}(x, \text{subrange}(y, n, m)) \wedge \text{disjoint}(\text{subrange}(y, n, m), x))$

THEOREM: open-subrange
 $((m < n) \rightarrow (\text{subrange}(l, n, m) = \text{nil}))$

$$\begin{aligned}
& \wedge \quad (((m \not\succeq n) \wedge (n \simeq 0) \wedge (m \simeq 0)) \\
& \quad \rightarrow \quad (\text{subrange}(l, n, m) = \text{list}(\text{car}(l)))) \\
& \wedge \quad (((m \not\succeq n) \wedge (n \simeq 0) \wedge (m \not\simeq 0)) \\
& \quad \rightarrow \quad (\text{subrange}(l, n, m) = \text{cons}(\text{car}(l), \text{subrange}(\text{cdr}(l), 0, m - 1)))) \\
& \wedge \quad (((m \not\succeq n) \wedge (n \not\simeq 0)) \\
& \quad \rightarrow \quad (\text{subrange}(l, n, m) = \text{subrange}(\text{cdr}(l), n - 1, m - 1)))
\end{aligned}$$

EVENT: Enable open-subrange; name this event ‘open-subrange-off’.

DEFINITION:

```

update-nth(n, lst, value)
=  if listp(lst)
    then if n  $\simeq$  0 then cons(value, cdr(lst))
        else cons(car(lst), update-nth(n - 1, cdr(lst), value)) endif
    else lst endif

```

EVENT: Enable update-nth; name this event ‘update-nth-off’.

DEFINITION:

```

make-list(n, value)
=  if n  $\simeq$  0 then nil
    else cons(value, make-list(n - 1, value)) endif

```

EVENT: Enable make-list; name this event ‘make-list-off’.

THEOREM: open-make-list

$$\begin{aligned}
& ((n \simeq 0) \rightarrow (\text{make-list}(n, x) = \text{nil})) \\
& \wedge \quad ((n \not\simeq 0) \rightarrow (\text{make-list}(n, x) = \text{cons}(x, \text{make-list}(n - 1, x))))
\end{aligned}$$

EVENT: Enable open-make-list; name this event ‘open-make-list-off’.

THEOREM: length-make-list

$$\text{length}(\text{make-list}(n, \text{value})) = \text{fix}(n)$$

THEOREM: make-list-append

$$\text{append}(\text{make-list}(n, \text{value}), \text{make-list}(m, \text{value})) = \text{make-list}(n + m, \text{value})$$

THEOREM: properp-make-list

$$\text{properp}(\text{make-list}(n, \text{value}))$$

THEOREM: firstn-make-list

```

if m < n then f
else t endif
→ (firstn(n, make-list(m, v)) = make-list(n, v))

```

DEFINITION:

$\text{tree-size}(tree)$
= **if** $tree \simeq \mathbf{nil}$ **then** 1
 else $\text{tree-size}(\text{car}(tree)) + \text{tree-size}(\text{cdr}(tree))$ **endif**

EVENT: Enable tree-size; name this event ‘tree-size-off’.

THEOREM: tree-size-nlistp

$(tree \simeq \mathbf{nil}) \rightarrow (\text{tree-size}(tree) = 1)$

THEOREM: not-equal-tree-size-tree-0

$\text{tree-size}(tree) \neq 0$

THEOREM: tree-size-1-crock

$1 \neq \text{tree-size}(\text{cons}(a, b))$

THEOREM: a-helpful-lemma-for-tree-inductions

$(\text{length}(a) = \text{tree-size}(tree))$
 $\rightarrow ((\text{length}(a) < \text{tree-size}(\text{car}(tree))) = \mathbf{f})$

THEOREM: tree-size-lemmas

$((\text{listp}(tree) \wedge (size = \text{tree-size}(tree)))$
 $\rightarrow ((size - \text{tree-size}(\text{car}(tree))) = \text{tree-size}(\text{cdr}(tree))))$
 $\wedge ((\text{listp}(tree) \wedge (size = \text{tree-size}(tree)))$
 $\rightarrow ((size - \text{tree-size}(\text{cdr}(tree))) = \text{tree-size}(\text{car}(tree))))$

THEOREM: make-list-append-tree-crock

$\text{listp}(tree)$
 $\rightarrow (\text{make-list}(\text{tree-size}(\text{car}(tree)) + \text{tree-size}(\text{cdr}(tree)), value)$
 $= \text{make-list}(\text{tree-size}(tree), value)$

EVENT: Enable make-list-append-tree-crock; name this event ‘make-list-append-tree-crock-off’.

DEFINITION:

$\text{tfirstn}(list, tree) = \text{firstn}(\text{tree-size}(\text{car}(tree)), list)$

DEFINITION:

$\text{trestn}(list, tree) = \text{restn}(\text{tree-size}(\text{car}(tree)), list)$

DEFINITION:

$\text{tree-height}(tree)$
= **if** $tree \simeq \mathbf{nil}$ **then** 1
 else $1 + \max(\text{tree-height}(\text{car}(tree)), \text{tree-height}(\text{cdr}(tree)))$ **endif**

EVENT: Enable tree-height; name this event ‘tree-height-off’.

DEFINITION:

make-tree (n)

= **if** $n \simeq 0$ **then** 0
 elseif $n = 1$ **then** 0
 else cons (make-tree ($n \div 2$), make-tree ($n - (n \div 2)$)) **endif**

EVENT: Enable make-tree; name this event ‘make-tree-off’.

THEOREM: tree-size-make-tree

$(n \not\simeq 0) \rightarrow (\text{tree-size} (\text{make-tree} (n)) = n)$

THEOREM: listp-make-tree

listp (make-tree (n))

= **if** $n < 2$ **then** **f**
 else **t** **endif**

THEOREM: list-rewrite-4

(properp (a) $\wedge$ (length (a) = 4))

$\rightarrow$ (list (car (a), cadr (a), caddr (a), caddr (a)) = a)

EVENT: Enable list-rewrite-4; name this event ‘list-rewrite-4-off’.

THEOREM: list-elim-4

$(l = \text{list} (a, b, c, d))$

= ((car (l) = a)
 $\wedge$ (cadr (l) = b)
 $\wedge$ (caddr (l) = c)
 $\wedge$ (caddr (l) = d)
 $\wedge$ (caddr (l) = **nil**))

EVENT: Enable list-elim-4; name this event ‘list-elim-4-off’.

DEFINITION:

list-as-collected-nth (l , $length$, n)

= **if** $length \simeq 0$ **then** **nil**
 else cons (nth (n , l), list-as-collected-nth (l , $length - 1$, $1 + n$)) **endif**

EVENT: Enable list-as-collected-nth; name this event ‘list-as-collected-nth-off’.

THEOREM: open-list-as-collected-nth

$((length \simeq 0) \rightarrow (\text{list-as-collected-nth} (l, length, n) = \text{nil}))$

$$\begin{aligned}
& \wedge ((length \neq 0) \\
& \quad \rightarrow (list-as-collected-nth(l, length, n) \\
& \quad \quad = cons(nth(n, l), \\
& \quad \quad \quad list-as-collected-nth(l, length - 1, 1 + n))))
\end{aligned}$$

THEOREM: equal-length-4-as-collected-nth
 $((length(l) = 4) \wedge properp(l))$
 $\rightarrow (l = list-as-collected-nth(l, 4, 0))$

THEOREM: equal-length-32-as-collected-nth
 $((length(l) = 32) \wedge properp(l))$
 $\rightarrow (l = list-as-collected-nth(l, 32, 0))$

EVENT: Add the shell *index*, with recognizer function symbol *indexp* and 2 accessors: *i-name*, with type restriction (one-of numberp litatom) and default value zero; *i-num*, with type restriction (one-of numberp) and default value zero.

DEFINITION:
 $infix(x)$
 $=$ **if** $(x \in \mathbf{N}) \vee litatom(x)$ **then** x
 else 0 **endif**

DEFINITION:
 $indices(name, from, n)$
 $=$ **if** $n \simeq 0$ **then** **nil**
 else $cons(index(name, from), indices(name, 1 + from, n - 1))$ **endif**

EVENT: Enable indices; name this event ‘indices-off’.

THEOREM: indices-zero
 $(n \simeq 0) \rightarrow (indices(name, from, n) = \mathbf{nil})$

THEOREM: open-indices
 $(n \neq 0)$
 $\rightarrow (indices(name, from, n)$
 $= cons(index(name, from), indices(name, 1 + from, n - 1)))$

THEOREM: indices-as-append
 $(n \neq 0)$
 $\rightarrow (indices(name, from, n)$
 $= append(indices(name, from, n - 1),$
 $list(index(name, (n - 1) + from)))$

EVENT: Enable indices-as-append; name this event ‘indices-as-append-off’.

THEOREM: length-indices
 $\text{length}(\text{indices}(\text{name}, \text{from}, n)) = \text{fix}(n)$

THEOREM: listp-indices
 $\text{listp}(\text{indices}(\text{name}, \text{from}, n)) = (n \neq 0)$

THEOREM: properp-indices
 $\text{properp}(\text{indices}(\text{name}, \text{from}, n))$

THEOREM: member-indices
 $(x \in \text{indices}(\text{name}, \text{from}, n))$
 $= ((n \neq 0)$
 $\quad \wedge \text{indexp}(x)$
 $\quad \wedge (\text{i-name}(x) = \text{Infix}(\text{name}))$
 $\quad \wedge \text{if } \text{i-num}(x) < \text{from} \text{ then } \mathbf{f}$
 $\quad \quad \mathbf{else } \mathbf{t} \text{ endif}$
 $\quad \wedge (\text{i-num}(x) < (\text{from} + n)))$

THEOREM: disjoint-indices-different-names
 $(\text{Infix}(\text{name1}) \neq \text{Infix}(\text{name2}))$
 $\rightarrow \text{disjoint}(\text{indices}(\text{name1}, \text{from1}, n1), \text{indices}(\text{name2}, \text{from2}, n2))$

THEOREM: no-duplicates-in-indices
 $\neg \text{duplicates?}(\text{indices}(\text{name}, \text{from}, n))$

THEOREM: position-name-indices
 $(\text{index} \in \text{indices}(\text{name}, \text{from}, n))$
 $\rightarrow (\text{position}(\text{index}, \text{indices}(\text{name}, \text{from}, n)) = (\text{i-num}(\text{index}) - \text{from}))$

THEOREM: nth-indices
 $(n < m) \rightarrow (\text{nth}(n, \text{indices}(\text{name}, \text{from}, m)) = \text{index}(\text{name}, n + \text{from}))$

EVENT: Enable nth-indices; name this event 'nth-indices-off'.

EVENT: Add the shell x , with recognizer function symbol xp and no accessors.

EVENT: Add the shell z , with recognizer function symbol zp and no accessors.

DEFINITION: $\text{boolp}(x) = ((x = \mathbf{t}) \vee (x = \mathbf{f}))$

EVENT: Enable boolp; name this event 'boolp-off'.

THEOREM: prove-boolp
 $(\text{truep}(x) \vee \text{falsep}(x)) \rightarrow \text{boolp}(x)$

THEOREM: boolp-lemmas

boolp(**t**)
 $\wedge$ boolp(**f**)
 $\wedge$ (boolp(b) $\rightarrow$ (($b = \mathbf{t}$) = b))
 $\wedge$ (boolp(b) $\rightarrow$ (($b = \mathbf{f}$) = ($\neg b$)))

THEOREM: boolp-implies-not-equal-z

boolp(x) $\rightarrow$ ($x \neq \mathbf{z}$)

EVENT: Enable boolp-implies-not-equal-z; name this event ‘boolp-implies-not-equal-z-off’.

DEFINITION:

boolfix(x)
= **if** x **then** **t**
 else **f** **endif**

DEFINITION: threep(x) = (($x = \mathbf{t}$) $\vee$ ($x = \mathbf{f}$) $\vee$ ($x = \mathbf{x}$))

DEFINITION:

threefix(x)
= **if** boolp(x) **then** x
 else $\mathbf{x}$ **endif**

THEOREM: threefix-idempotent

threefix(threefix(x)) = threefix(x)

THEOREM: threefix=x

($\mathbf{x} = \text{threefix}(x)$) = ($\neg \text{boolp}(x)$)

EVENT: Enable threefix=x; name this event ‘threefix=x-off’.

DEFINITION:

fourp(x) = (($x = \mathbf{t}$) $\vee$ ($\mathbf{f} = x$) $\vee$ ($x = \mathbf{x}$) $\vee$ ($x = \mathbf{z}$))

DEFINITION:

fourfix(x)
= **if** fourp(x) **then** x
 else $\mathbf{x}$ **endif**

DEFINITION:

bvp(x)
= **if** $x \simeq \mathbf{nil}$ **then** $x = \mathbf{nil}$
 else boolp(car(x)) $\wedge$ bvp(cdr(x)) **endif**

EVENT: Enable bvp; name this event ‘bvp-off’.

THEOREM: bvp-nlistp
 $(x \simeq \mathbf{nil}) \rightarrow (\text{bvp}(x) = (x = \mathbf{nil}))$

THEOREM: bvp-cons
 $\text{bvp}(\text{cons}(x, y)) = (\text{boolp}(x) \wedge \text{bvp}(y))$

THEOREM: bvp-firstn
 $\text{bvp}(x) \rightarrow \text{bvp}(\text{firstn}(n, x))$

THEOREM: bvp-restn
 $\text{bvp}(x) \rightarrow \text{bvp}(\text{restn}(n, x))$

THEOREM: bvp-append
 $\text{bvp}(a) \rightarrow (\text{bvp}(\text{append}(a, b)) = \text{bvp}(b))$

THEOREM: bvp-is-properp
 $\text{bvp}(v) \rightarrow \text{properp}(v)$

THEOREM: bvp-make-list
 $\text{bvp}(\text{make-list}(n, v)) = ((n \simeq 0) \vee \text{boolp}(v))$

THEOREM: bvp-nthcdr
 $(\text{bvp}(l) \wedge \text{if length}(l) < n \text{ then f else t endif}) \rightarrow \text{bvp}(\text{nthcdr}(n, l))$

THEOREM: bvp-if
 $(\text{bvp}(a) \wedge \text{bvp}(b)) \rightarrow \text{bvp}(\text{if } c \text{ then } a \text{ else } b \text{ endif})$

THEOREM: bvp-if*
 $(\text{bvp}(a) \wedge \text{bvp}(b)) \rightarrow \text{bvp}(\text{if}^*(c, a, b))$

DEFINITION:
 $\text{bv2p}(x, y) = (\text{bvp}(x) \wedge \text{bvp}(y) \wedge (\text{length}(x) = \text{length}(y)))$

DEFINITION:
 $\text{bvp-length}(bvp, n) = (\text{bvp}(bvp) \wedge \text{if length}(bvp) < n \text{ then f else t endif})$

EVENT: Enable bvp-length; name this event 'bvp-length-off'.

THEOREM: bvp-length-cdr
 $\text{bvp-length}(x, 1 + n) \rightarrow \text{bvp-length}(\text{cdr}(x), n)$

THEOREM: boolfix-car-x=x
 $\text{bvp-length}(x, 1) \rightarrow (\text{boolfix}(\text{car}(x)) = \text{car}(x))$

THEOREM: boolp-car-x
 $\text{bvp-length}(x, 1) \rightarrow \text{boolp}(\text{car}(x))$

THEOREM: bvp-length-restn
 $\text{bvp}(bvp)$
 $\rightarrow (\text{bvp-length}(\text{restn}(n, bvp), m)$
 $\quad = \text{if } (\text{length}(bvp) - n) < m \text{ then f}$
 $\quad \text{else t endif})$

THEOREM: show-that-nth=t
 $(\text{nth}(n, a) \wedge \text{bvp}(a) \wedge (n < \text{length}(a))) \rightarrow (\text{nth}(n, a) = \mathbf{t})$

EVENT: Enable show-that-nth=t; name this event 'show-that-nth=t-off'.

THEOREM: boolp-nth
 $\text{bvp}(a) \rightarrow (\text{boolp}(\text{nth}(n, a)) = (n < \text{length}(a)))$

THEOREM: another-way-to-look-at-boolp-nth
 $(\text{bvp}(v) \wedge (n < \text{length}(v)))$
 $\rightarrow ((\text{if } \text{nth}(n, v) \text{ then t}$
 $\quad \text{else f endif}$
 $\quad = \text{nth}(n, v))$
 $\quad = \mathbf{t})$

DEFINITION:
 $\text{xor}(a, b)$
 $= \text{if } a$
 $\quad \text{then if } b \text{ then f}$
 $\quad \quad \text{else t endif}$
 $\quad \text{elseif } b \text{ then t}$
 $\quad \text{else f endif}$

DEFINITION:
 $\text{b-buf}(x)$
 $= \text{if } x \text{ then t}$
 $\quad \text{else f endif}$

DEFINITION: $\text{b-not}(x) = (\neg x)$

DEFINITION: $\text{b-nand}(a, b) = (\neg (a \wedge b))$

DEFINITION: $\text{b-nand3}(a, b, c) = (\neg (a \wedge b \wedge c))$

DEFINITION: $\text{b-nand4}(a, b, c, d) = (\neg (a \wedge b \wedge c \wedge d))$

DEFINITION:

$\text{b-nand5}(a, b, c, d, e) = (\neg (a \wedge b \wedge c \wedge d \wedge e))$

DEFINITION:

$\text{b-nand6}(a, b, c, d, e, g) = (\neg (a \wedge b \wedge c \wedge d \wedge e \wedge g))$

DEFINITION:

$\text{b-nand8}(a, b, c, d, e, g, h, i)$
 $= (\neg (a \wedge b \wedge c \wedge d \wedge e \wedge g \wedge h \wedge i))$

DEFINITION: $\text{b-or}(a, b) = (a \vee b)$

DEFINITION: $\text{b-or3}(a, b, c) = (a \vee b \vee c)$

DEFINITION: $\text{b-or4}(a, b, c, d) = (a \vee b \vee c \vee d)$

DEFINITION:

$\text{b-xor}(x, y)$
 $=$ **if** x
 then if y **then** f
 else t **endif**
 elseif y **then** t
 else f **endif**

DEFINITION: $\text{b-xor3}(a, b, c) = \text{b-xor}(\text{b-xor}(a, b), c)$

DEFINITION:

$\text{b-equiv}(x, y)$
 $=$ **if** x
 then if y **then** t
 else f **endif**
 elseif y **then** f
 else t **endif**

DEFINITION: $\text{b-equiv3}(a, b, c) = \text{b-equiv}(a, \text{b-xor}(b, c))$

DEFINITION: $\text{b-and}(a, b) = (a \wedge b)$

DEFINITION: $\text{b-and3}(a, b, c) = (a \wedge b \wedge c)$

DEFINITION: $\text{b-and4}(a, b, c, d) = (a \wedge b \wedge c \wedge d)$

DEFINITION: $\text{b-nor}(a, b) = (\neg (a \vee b))$

DEFINITION: $\text{b-nor3}(a, b, c) = (\neg (a \vee b \vee c))$

DEFINITION: $\text{b-nor4}(a, b, c, d) = (\neg (a \vee b \vee c \vee d))$

DEFINITION:

$\text{b-nor5}(a, b, c, d, e) = (\neg (a \vee b \vee c \vee d \vee e))$

DEFINITION:

$\text{b-nor6}(a, b, c, d, e, g) = (\neg (a \vee b \vee c \vee d \vee e \vee g))$

DEFINITION:

$\text{b-nor8}(a, b, c, d, e, g, h, i)$
 $= (\neg (a \vee b \vee c \vee d \vee e \vee g \vee h \vee i))$

DEFINITION:

$\text{b-if}(c, a, b)$
 $=$ **if** c
 then if a **then** t
 else f endif
 elseif b **then** t
 else f endif

THEOREM: open-b-xor

$\text{b-xor}(a, b)$
 $=$ **if** a
 then if b **then** f
 else t endif
 elseif b **then** t
 else f endif

THEOREM: open-b-xor3

$\text{b-xor3}(a, b, c) = \text{b-xor}(a, \text{b-xor}(b, c))$

THEOREM: open-b-equiv

$\text{b-equiv}(a, b)$
 $=$ **if** a
 then if b **then** t
 else f endif
 elseif b **then** f
 else t endif

THEOREM: open-b-equiv3

$\text{b-equiv3}(a, b, c) = \text{b-equiv}(a, \text{b-xor}(b, c))$

THEOREM: b-and-rewrite
 $(\neg \text{b-and}(\mathbf{f}, x)) \wedge (\neg \text{b-and}(x, \mathbf{f})) \wedge ((x \wedge y) \rightarrow \text{b-and}(x, y))$

THEOREM: b-buf-x=x
 $\text{boolp}(x) \rightarrow (\text{b-buf}(x) = x)$

EVENT: Let us define the theory *b-gates* to consist of the following events: b-buf, b-not, b-nand, b-nand3, b-nand4, b-nand5, b-nand6, b-nand8, b-or, b-or3, b-or4, b-xor, b-xor3, b-equiv, b-equiv3, b-and, b-and3, b-and4, b-nor, b-nor3, b-nor4, b-nor5, b-nor6, b-nor8, b-if, open-b-xor, open-b-xor3, open-b-equiv, open-b-equiv3, b-and-rewrite, b-buf-x=x.

THEOREM: boolp-b-gates
 $\text{boolp}(\text{b-buf}(x))$
 $\wedge \text{boolp}(\text{b-not}(x))$
 $\wedge \text{boolp}(\text{b-nand}(a, b))$
 $\wedge \text{boolp}(\text{b-nand3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-nand4}(a, b, c, d))$
 $\wedge \text{boolp}(\text{b-nand5}(a, b, c, d, e))$
 $\wedge \text{boolp}(\text{b-nand6}(a, b, c, d, e, g))$
 $\wedge \text{boolp}(\text{b-nand8}(a, b, c, d, e, g, h, i))$
 $\wedge \text{boolp}(\text{b-or}(a, b))$
 $\wedge \text{boolp}(\text{b-or3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-or4}(a, b, c, d))$
 $\wedge \text{boolp}(\text{b-xor}(x, y))$
 $\wedge \text{boolp}(\text{b-xor3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-equiv}(x, y))$
 $\wedge \text{boolp}(\text{b-equiv3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-and}(a, b))$
 $\wedge \text{boolp}(\text{b-and3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-and4}(a, b, c, d))$
 $\wedge \text{boolp}(\text{b-nor}(a, b))$
 $\wedge \text{boolp}(\text{b-nor3}(a, b, c))$
 $\wedge \text{boolp}(\text{b-nor4}(a, b, c, d))$
 $\wedge \text{boolp}(\text{b-nor5}(a, b, c, d, e))$
 $\wedge \text{boolp}(\text{b-nor6}(a, b, c, d, e, g))$
 $\wedge \text{boolp}(\text{b-nor8}(a, b, c, d, e, g, h, i))$
 $\wedge \text{boolp}(\text{b-if}(c, a, b))$

EVENT: Enable boolp-b-gates; name this event ‘boolp-b-gates-off’.

DEFINITION: $\text{id}(x) = x$

DEFINITION: $\text{ao2}(a, b, c, d) = \text{b-nor}(\text{b-and}(a, b), \text{b-and}(c, d))$

DEFINITION: $\text{ao4}(a, b, c, d) = \text{b-nand}(\text{b-or}(a, b), \text{b-or}(c, d))$

DEFINITION: $\text{ao6}(a, b, c) = \text{b-nor}(\text{b-and}(a, b), c)$

DEFINITION: $\text{ao7}(a, b, c) = \text{b-nand}(\text{b-or}(a, b), c)$

DEFINITION: $\text{vss} = \mathbf{f}$

DEFINITION: $\text{vdd} = \mathbf{t}$

DEFINITION:

$\text{v-buf}(x)$

$= \text{if } x \simeq \mathbf{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{b-buf}(\text{car}(x)), \text{v-buf}(\text{cdr}(x))) \text{ endif}$

EVENT: Enable v-buf; name this event ‘v-buf-off’.

DEFINITION:

$\text{v-not}(x)$

$= \text{if } x \simeq \mathbf{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{b-not}(\text{car}(x)), \text{v-not}(\text{cdr}(x))) \text{ endif}$

EVENT: Enable v-not; name this event ‘v-not-off’.

DEFINITION:

$\text{v-and}(x, y)$

$= \text{if } x \simeq \mathbf{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{b-and}(\text{car}(x), \text{car}(y)), \text{v-and}(\text{cdr}(x), \text{cdr}(y))) \text{ endif}$

EVENT: Enable v-and; name this event ‘v-and-off’.

DEFINITION:

$\text{v-or}(x, y)$

$= \text{if } x \simeq \mathbf{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{b-or}(\text{car}(x), \text{car}(y)), \text{v-or}(\text{cdr}(x), \text{cdr}(y))) \text{ endif}$

EVENT: Enable v-or; name this event ‘v-or-off’.

DEFINITION:

$\text{v-xor}(x, y)$

$= \text{if } x \simeq \mathbf{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{b-xor}(\text{car}(x), \text{car}(y)), \text{v-xor}(\text{cdr}(x), \text{cdr}(y))) \text{ endif}$

EVENT: Enable v-xor; name this event ‘v-xor-off’.

DEFINITION:

$\text{v-shift-right}(a, si)$
= **if** $a \simeq \text{nil}$ **then nil**
 else $\text{append}(\text{v-buf}(\text{cdr}(a)), \text{list}(\text{boolfix}(si)))$ **endif**

EVENT: Enable v-shift-right; name this event ‘v-shift-right-off’.

DEFINITION: $\text{v-lsr}(a) = \text{v-shift-right}(a, \mathbf{f})$

DEFINITION: $\text{v-ror}(a, si) = \text{v-shift-right}(a, si)$

DEFINITION: $\text{v-asr}(a) = \text{v-shift-right}(a, \text{nth}(\text{length}(a) - 1, a))$

DEFINITION:

$\text{v-if}(c, a, b)$
= **if** $a \simeq \text{nil}$ **then nil**
 else $\text{cons}(\text{if } c \text{ then } \text{car}(a)$
 else $\text{car}(b)$ **endif** **then t**
 else f **endif**,
 $\text{v-if}(c, \text{cdr}(a), \text{cdr}(b)))$ **endif**

EVENT: Enable v-if; name this event ‘v-if-off’.

THEOREM: bvp-v-buf
 $\text{bvp}(\text{v-buf}(a))$

THEOREM: bvp-v-not
 $\text{bvp}(\text{v-not}(a))$

THEOREM: bvp-v-and
 $\text{bvp}(\text{v-and}(a, b))$

THEOREM: bvp-v-or
 $\text{bvp}(\text{v-or}(a, b))$

THEOREM: bvp-v-xor
 $\text{bvp}(\text{v-xor}(a, b))$

THEOREM: bvp-v-shift-right
 $\text{bvp}(\text{v-shift-right}(a, si))$

THEOREM: bvp-v-lsr
 $\text{bvp}(\text{v-lsr}(a))$

THEOREM: bvp-v-asr
 $\text{bvp}(\text{v-asr}(a))$

THEOREM: bvp-v-ror

$$\text{bvp}(\text{v-ror}(a, c))$$

THEOREM: bvp-v-if

$$\text{bvp}(\text{v-if}(c, a, b))$$

THEOREM: length-v-buf

$$\text{length}(\text{v-buf}(a)) = \text{length}(a)$$

THEOREM: length-v-not

$$\text{length}(\text{v-not}(a)) = \text{length}(a)$$

THEOREM: length-v-and

$$\text{length}(\text{v-and}(a, b)) = \text{length}(a)$$

THEOREM: length-v-or

$$\text{length}(\text{v-or}(a, b)) = \text{length}(a)$$

THEOREM: length-v-xor

$$\text{length}(\text{v-xor}(a, b)) = \text{length}(a)$$

THEOREM: length-v-shift-right

$$\text{length}(\text{v-shift-right}(a, b)) = \text{length}(a)$$

THEOREM: length-v-lsr

$$\text{length}(\text{v-lsr}(a)) = \text{length}(a)$$

THEOREM: length-v-asr

$$\text{length}(\text{v-asr}(a)) = \text{length}(a)$$

THEOREM: length-v-ror

$$\text{length}(\text{v-ror}(a, b)) = \text{length}(a)$$

THEOREM: length-v-if

$$\text{length}(\text{v-if}(c, a, b)) = \text{length}(a)$$

THEOREM: v-and-append-help

$$(\text{length}(a) = \text{length}(b))$$

$$\rightarrow (\text{append}(\text{v-and}(a, b), \text{v-and}(d, e)) = \text{v-and}(\text{append}(a, d), \text{append}(b, e)))$$

THEOREM: v-or-append-help

$$(\text{length}(a) = \text{length}(b))$$

$$\rightarrow (\text{append}(\text{v-or}(a, b), \text{v-or}(d, e)) = \text{v-or}(\text{append}(a, d), \text{append}(b, e)))$$

THEOREM: v-xor-append-help

$$(\text{length}(a) = \text{length}(b))$$

$$\rightarrow (\text{append}(\text{v-xor}(a, b), \text{v-xor}(d, e)) = \text{v-xor}(\text{append}(a, d), \text{append}(b, e)))$$

THEOREM: v-not-append-help

$$\text{append}(\text{v-not}(a), \text{v-not}(b)) = \text{v-not}(\text{append}(a, b))$$

THEOREM: v-buf-append-help

$$\text{append}(\text{v-buf}(a), \text{v-buf}(b)) = \text{v-buf}(\text{append}(a, b))$$

THEOREM: v-if-append-help

$$\begin{aligned} &(\text{length}(a) = \text{length}(b)) \\ \rightarrow &(\text{append}(\text{v-if}(c, a, b), \text{v-if}(c, d, e)) = \text{v-if}(c, \text{append}(a, d), \text{append}(b, e))) \end{aligned}$$

THEOREM: v-if-c-congruence

$$c \rightarrow ((\text{v-if}(c, a, b) = \text{v-if}(\mathbf{t}, a, b)) = \mathbf{t})$$

THEOREM: v-not-firstn

$$\text{v-not}(\text{firstn}(n, l)) = \text{firstn}(n, \text{v-not}(l))$$

THEOREM: v-not-restn

$$\text{v-not}(\text{restn}(n, l)) = \text{restn}(n, \text{v-not}(l))$$

THEOREM: firstn-v-not

$$\text{firstn}(n, \text{v-not}(l)) = \text{v-not}(\text{firstn}(n, l))$$

EVENT: Enable firstn-v-not; name this event ‘firstn-v-not-off’.

THEOREM: restn-v-not

$$\text{restn}(n, \text{v-not}(l)) = \text{v-not}(\text{restn}(n, l))$$

EVENT: Enable restn-v-not; name this event ‘restn-v-not-off’.

THEOREM: v-not-inverts-all

$$(n < \text{length}(bvp)) \rightarrow (\text{nth}(n, \text{v-not}(bvp)) = \text{b-not}(\text{nth}(n, bvp)))$$

THEOREM: v-or-make-list-f

$$(\text{bvp}(a) \wedge (\text{length}(a) = n)) \rightarrow (\text{v-or}(\text{make-list}(n, \mathbf{f}), a) = a)$$

DEFINITION:

v-to-nat(v)

$$\begin{aligned} = & \text{ if } v \simeq \mathbf{nil} \text{ then } 0 \\ & \text{ else if } \text{car}(v) \text{ then } 1 \\ & \quad \text{ else } 0 \text{ endif} \\ & + (2 * \text{v-to-nat}(\text{cdr}(v))) \text{ endif} \end{aligned}$$

EVENT: Enable v-to-nat; name this event ‘v-to-nat-off’.

DEFINITION:

$\text{nat-to-v}(x, n)$
 $=$ **if** $n \simeq 0$ **then** **nil**
 else $\text{cons}((x \bmod 2) \neq 0, \text{nat-to-v}(x \div 2, n - 1))$ **endif**

EVENT: Enable nat-to-v; name this event ‘nat-to-v-off’.

THEOREM: firstn-nat-to-v

if $m < n$ **then** **f**
 else **t** **endif**
 $\rightarrow$ $(\text{firstn}(n, \text{nat-to-v}(\text{nat}, m)) = \text{nat-to-v}(\text{nat}, n))$

THEOREM: restn-nat-to-v-0-hack

if $m < n$ **then** **f**
 else **t** **endif**
 $\rightarrow$ $(\text{restn}(n, \text{nat-to-v}(0, m)) = \text{nat-to-v}(0, m - n))$

THEOREM: length-nat-to-v

$\text{length}(\text{nat-to-v}(n, \text{length})) = \text{fix}(\text{length})$

THEOREM: bvp-nat-to-v

$\text{bvp}(\text{nat-to-v}(n, \text{length}))$

THEOREM: car-nat-to-v-0-is-f

$(n \neq 0) \rightarrow (\neg \text{car}(\text{nat-to-v}(0, n)))$

THEOREM: any-of-nat-to-v-0-is-f

$(n < m) \rightarrow (\neg \text{nth}(n, \text{nat-to-v}(0, m)))$

DEFINITION: $\text{v-nth}(v\text{-}n, \text{lst}) = \text{nth}(\text{v-to-nat}(v\text{-}n), \text{lst})$

EVENT: Enable v-nth; name this event ‘v-nth-off’.

DEFINITION:

$\text{update-v-nth}(v\text{-}n, \text{lst}, \text{value}) = \text{update-nth}(\text{v-to-nat}(v\text{-}n), \text{lst}, \text{value})$

EVENT: Enable update-v-nth; name this event ‘update-v-nth-off’.

DEFINITION:

$\text{v-nzerop}(x)$
 $=$ **if** $x \simeq \text{nil}$ **then** **f**
 else $\text{car}(x) \vee \text{v-nzerop}(\text{cdr}(x))$ **endif**

EVENT: Enable v-nzerop; name this event ‘v-nzerop-off’.

DEFINITION: $\text{v-zerop}(x) = (\neg \text{v-nzerop}(x))$

THEOREM: v-nzerop-as-or-crock
 $(\text{v-nzerop}(\text{firstn}(n, a)) \rightarrow \text{v-nzerop}(a))$
 $\wedge (\text{v-nzerop}(\text{restn}(n, a)) \rightarrow \text{v-nzerop}(a))$

THEOREM: not-v-nzerop-as-and-crock
 $((\neg \text{v-nzerop}(\text{firstn}(n, a))) \wedge (\neg \text{v-nzerop}(\text{restn}(n, a))))$
 $\rightarrow (\neg \text{v-nzerop}(a))$

THEOREM: not-v-nzerop-v-xor-x-x
 $\neg \text{v-nzerop}(\text{v-xor}(x, x))$

THEOREM: v-xor-nzerop=not-equal
 $\text{bv2p}(a, b) \rightarrow (\text{v-nzerop}(\text{v-xor}(a, b)) = (a \neq b))$

THEOREM: v-zerop-make-list-f
 $\text{v-zerop}(\text{make-list}(n, \mathbf{f})) = \mathbf{t}$

THEOREM: not-v-nzerop-all-f
 $\neg \text{v-nzerop}(\text{make-list}(n, \mathbf{f}))$

DEFINITION:
 $\text{v-negp}(x)$
 $=$ **if** $x \simeq \mathbf{nil}$ **then** $\mathbf{f}$
 elseif $\text{cdr}(x) \simeq \mathbf{nil}$ **then** $\text{car}(x)$
 else $\text{v-negp}(\text{cdr}(x))$ **endif**

EVENT: Enable v-negp; name this event ‘v-negp-off’.

THEOREM: boolp-v-negp
 $\text{bvp}(v) \rightarrow \text{boolp}(\text{v-negp}(v))$

THEOREM: v-negp-as-nth
 $(\text{length}(bv) \neq 0) \rightarrow (\text{v-negp}(bv) = \text{nth}(\text{length}(bv) - 1, bv))$

EVENT: Enable v-negp-as-nth; name this event ‘v-negp-as-nth-off’.

DEFINITION:
 $\text{sign-extend}(v, n)$
 $=$ **if** $n \simeq 0$ **then** $\mathbf{nil}$
 elseif $v \simeq \mathbf{nil}$ **then** $\text{make-list}(n, \mathbf{f})$
 elseif $\text{cdr}(v) \simeq \mathbf{nil}$
 then $\text{cons}(\text{boolfix}(\text{car}(v)), \text{make-list}(n - 1, \text{boolfix}(\text{car}(v))))$
 else $\text{cons}(\text{boolfix}(\text{car}(v)), \text{sign-extend}(\text{cdr}(v), n - 1))$ **endif**

EVENT: Enable sign-extend; name this event ‘sign-extend-off’.

THEOREM: length-sign-extend
 $\text{length}(\text{sign-extend}(v, n)) = \text{fix}(n)$

THEOREM: bvp-sign-extend
 $\text{bvp}(\text{sign-extend}(v, n))$

THEOREM: sign-extend-as-append
 $(\text{bvp}(v)$
 $\wedge \text{ if } n < \text{length}(v) \text{ then f}$
 else t endif
 $\wedge (\text{length}(v) \neq 0))$
 $\rightarrow (\text{sign-extend}(v, n)$
 $= \text{append}(v, \text{make-list}(n - \text{length}(v), \text{nth}(\text{length}(v) - 1, v))))$

EVENT: Enable sign-extend-as-append; name this event ‘sign-extend-as-append-off’.

DEFINITION:
 $\text{v-adder}(c, a, b)$
 $= \text{ if } a \simeq \text{nil} \text{ then list}(\text{boolfix}(c))$
 $\text{ else cons}(\text{b-xor3}(c, \text{car}(a), \text{car}(b)),$
 $\text{ v-adder}(\text{b-or}(\text{b-and}(\text{car}(a), \text{car}(b)),$
 $\text{ b-or}(\text{b-and}(\text{car}(a), c), \text{b-and}(\text{car}(b), c))),$
 $\text{cdr}(a),$
 $\text{cdr}(b))) \text{ endif}$

EVENT: Enable v-adder; name this event ‘v-adder-off’.

DEFINITION:
 $\text{v-adder-output}(c, a, b) = \text{firstn}(\text{length}(a), \text{v-adder}(c, a, b))$

DEFINITION:
 $\text{v-adder-carry-out}(c, a, b) = \text{nth}(\text{length}(a), \text{v-adder}(c, a, b))$

DEFINITION:
 $\text{v-adder-overflowp}(c, a, b)$
 $= \text{b-and}(\text{b-equiv}(\text{nth}(\text{length}(a) - 1, a), \text{nth}(\text{length}(b) - 1, b)),$
 $\text{b-xor}(\text{nth}(\text{length}(a) - 1, a),$
 $\text{nth}(\text{length}(a) - 1, \text{v-adder-output}(c, a, b))))$

DEFINITION:
 $\text{v-subtractor-output}(c, a, b) = \text{v-adder-output}(\text{b-not}(c), \text{v-not}(a), b)$

DEFINITION:

$$\begin{aligned} & \text{v-subtractor-carry-out}(c, a, b) \\ &= \text{b-not}(\text{v-adder-carry-out}(\text{b-not}(c), \text{v-not}(a), b)) \end{aligned}$$

DEFINITION:

$$\text{v-subtractor-overflowp}(c, a, b) = \text{v-adder-overflowp}(\text{b-not}(c), \text{v-not}(a), b)$$

DEFINITION:

$$\text{v-inc}(x) = \text{v-adder-output}(\mathbf{t}, x, \text{nat-to-v}(0, \text{length}(x)))$$

EVENT: Enable v-inc; name this event ‘v-inc-off’.

DEFINITION:

$$\text{v-dec}(x) = \text{v-subtractor-output}(\mathbf{t}, \text{nat-to-v}(0, \text{length}(x)), x)$$

EVENT: Enable v-dec; name this event ‘v-dec-off’.

THEOREM: length-of-v-adder

$$\text{length}(\text{v-adder}(c, a, b)) = (1 + \text{length}(a))$$

THEOREM: bvp-v-adder

$$\text{bvp}(\text{v-adder}(c, a, b))$$

THEOREM: length-of-v-adder-output

$$\text{length}(\text{v-adder-output}(c, a, b)) = \text{length}(a)$$

THEOREM: length-of-v-subtractor-output

$$\text{length}(\text{v-subtractor-output}(c, a, b)) = \text{length}(a)$$

THEOREM: bvp-length-v-inc-v-dec

$$\begin{aligned} & \text{bvp}(\text{v-inc}(x)) \\ & \wedge \text{bvp}(\text{v-dec}(x)) \\ & \wedge (\text{length}(\text{v-inc}(x)) = \text{length}(x)) \\ & \wedge (\text{length}(\text{v-dec}(x)) = \text{length}(x)) \end{aligned}$$

THEOREM: v-buf-works

$$\text{bvp}(x) \rightarrow (\text{v-buf}(x) = x)$$

THEOREM: v-if-works

$$\begin{aligned} & \text{bv2p}(x, y) \\ & \rightarrow (\text{v-if}(c, x, y) \\ & \quad = \text{if } c \text{ then } x \\ & \quad \text{else } y \text{ endif}) \end{aligned}$$

THEOREM: v-adder-works

$$\begin{aligned} & \text{bv2p}(x, y) \\ & \rightarrow (\text{v-to-nat}(\text{v-adder}(c, x, y))) \\ & = (\text{if } c \text{ then } 1 \\ & \quad \text{else } 0 \text{ endif} \\ & \quad + \text{v-to-nat}(x) \\ & \quad + \text{v-to-nat}(y)) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-threefix}(v) \\ & = \text{if } v \simeq \mathbf{nil} \text{ then } \mathbf{nil} \\ & \quad \text{else } \text{cons}(\text{threefix}(\text{car}(v)), \text{v-threefix}(\text{cdr}(v))) \text{ endif} \end{aligned}$$

EVENT: Enable v-threefix; name this event ‘v-threefix-off’.

THEOREM: open-v-threefix

$$\begin{aligned} & ((v \simeq \mathbf{nil}) \rightarrow (\text{v-threefix}(v) = \mathbf{nil})) \\ & \wedge (\text{listp}(v) \\ & \quad \rightarrow (\text{v-threefix}(v) = \text{cons}(\text{threefix}(\text{car}(v)), \text{v-threefix}(\text{cdr}(v))))) \end{aligned}$$

THEOREM: v-threefix-bvp

$$\text{bvp}(v) \rightarrow (\text{v-threefix}(v) = v)$$

THEOREM: properp-v-threefix

$$\text{properp}(\text{v-threefix}(x))$$

THEOREM: length-v-threefix

$$\text{length}(\text{v-threefix}(x)) = \text{length}(x)$$

THEOREM: append-v-threefix

$$\text{append}(\text{v-threefix}(a), \text{v-threefix}(b)) = \text{v-threefix}(\text{append}(a, b))$$

THEOREM: v-threefix-append

$$\text{v-threefix}(\text{append}(a, b)) = \text{append}(\text{v-threefix}(a), \text{v-threefix}(b))$$

EVENT: Enable v-threefix-append; name this event ‘v-threefix-append-off’.

THEOREM: v-threefix-idempotence

$$\text{v-threefix}(\text{v-threefix}(x)) = \text{v-threefix}(x)$$

THEOREM: bvp-v-threefix

$$\text{properp}(v) \rightarrow (\text{bvp}(\text{v-threefix}(v)) = \text{bvp}(v))$$

THEOREM: v-threefix-make-list-x

$$\text{v-threefix}(\text{make-list}(n, x)) = \text{make-list}(n, x)$$

DEFINITION:

$\text{v-fourfix}(v)$
 $=$ **if** $v \simeq \text{nil}$ **then nil**
 else $\text{cons}(\text{fourfix}(\text{car}(v)), \text{v-fourfix}(\text{cdr}(v)))$ **endif**

EVENT: Enable v-fourfix; name this event ‘v-fourfix-off’.

THEOREM: bvp-v-fourfix
 $\text{bvp}(v) \rightarrow (\text{v-fourfix}(v) = v)$

THEOREM: $\text{v-fourfix-make-list}$
 $((x = \mathbf{t}) \vee (x = \mathbf{f}) \vee (x = \mathbf{x}) \vee (x = \mathbf{z}))$
 $\rightarrow (\text{v-fourfix}(\text{make-list}(n, x)) = \text{make-list}(n, x))$

THEOREM: $\text{v-threefix-v-fourfix}$
 $\text{v-threefix}(\text{v-fourfix}(v)) = \text{v-threefix}(v)$

DEFINITION:

$\text{v-iff}(a, b)$
 $=$ **if** $a \simeq \text{nil}$ **then t**
 else $(\text{car}(a) \leftrightarrow \text{car}(b)) \wedge \text{v-iff}(\text{cdr}(a), \text{cdr}(b))$ **endif**

EVENT: Enable v-iff; name this event ‘v-iff-off’.

THEOREM: v-iff-x-x
 $\text{v-iff}(x, x)$

THEOREM: v-iff-rev1
 $((\text{length}(a) = \text{length}(b)) \wedge (\text{length}(x) = \text{length}(y)))$
 $\rightarrow (\text{v-iff}(\text{rev1}(a, x), \text{rev1}(b, y)) = (\text{v-iff}(a, b) \wedge \text{v-iff}(x, y)))$

THEOREM: v-iff-reverse
 $(\text{length}(a) = \text{length}(b)) \rightarrow (\text{v-iff}(\text{reverse}(a), \text{reverse}(b)) = \text{v-iff}(a, b))$

THEOREM: v-iff=equal
 $\text{bv2p}(a, b) \rightarrow (\text{v-iff}(a, b) = (a = b))$

THEOREM: bvp-subrange
 $(\text{bvp}(v) \wedge (n < \text{length}(v))) \rightarrow \text{bvp}(\text{subrange}(v, m, n))$

THEOREM: boolp-if^*
 $(\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow \text{boolp}(\text{if}^*(c, a, b))$

DEFINITION:

```
value(name, alist)
=  if alist  $\simeq$  nil then 0
   elseif listp(car(alist))
   then if caar(alist) = name then cdar(alist)
        else value(name, cdr(alist)) endif
   else value(name, cdr(alist)) endif
```

EVENT: Enable value; name this event 'value-off'.

DEFINITION:

```
collect-value(args, alist)
=  if args  $\simeq$  nil then nil
   else cons(value(car(args), alist), collect-value(cdr(args), alist)) endif
```

EVENT: Enable collect-value; name this event 'collect-value-off'.

THEOREM: rewrite-value

```
value(a, cons(cons(b, c), d))
=  if a = b then c
   else value(a, d) endif
```

THEOREM: rewrite-value-4x

```
value(a, cons(cons(b, c), cons(cons(d, e), cons(cons(g, h), cons(cons(i, j), k)))))
=  if a = b then c
   elseif a = d then e
   elseif a = g then h
   elseif a = i then j
   else value(a, k) endif
```

THEOREM: value-append-pairlist

```
value(a, append(pairlist(b, c), d))
=  if a  $\in$  b then value(a, pairlist(b, c))
   else value(a, d) endif
```

THEOREM: value-pairlist

```
(name  $\in$  names)
 $\rightarrow$  (value(name, pairlist(names, values))
      = car(restn(position(name, names), values)))
```

EVENT: Enable value-pairlist; name this event 'value-pairlist-off'.

THEOREM: value-indices-hack

```
((the-name  $\in$  indices(name, n, m))
```

$\wedge$ **if** length(*values*) < *m* **then f**
 else t endif
 $\rightarrow$ (value(*the-name*, pairlist(indices(*name*, *n*, *m*), *values*))
 = nth(i-num(*the-name*) - *n*, *values*))

THEOREM: collect-value-nlistp
 (*args* $\simeq$ **nil**) $\rightarrow$ (collect-value(*args*, *alist*) = **nil**)

THEOREM: listp-collect-value
 listp(collect-value(*args*, *alist*)) = listp(*args*)

THEOREM: collect-value-append
 collect-value(append(*a*, *b*), *alist*)
 = append(collect-value(*a*, *alist*), collect-value(*b*, *alist*))

THEOREM: length-collect-value
 length(collect-value(*args*, *alist*)) = length(*args*)

THEOREM: collect-value-append-pairlist-when-subset
 subset(*args1*, *args2*)
 $\rightarrow$ (collect-value(*args1*, append(pairlist(*args2*, *answer*), *x*))
 = collect-value(*args1*, pairlist(*args2*, *answer*)))

THEOREM: collect-value-append-pairlist-when-disjoint
 disjoint(*args1*, *args2*)
 $\rightarrow$ (collect-value(*args1*, append(pairlist(*args2*, *a*), *b*))
 = collect-value(*args1*, *b*))

THEOREM: collect-value-disjoint-pairlist
 (*a* $\notin$ *args*)
 $\rightarrow$ (collect-value(*args*, cons(cons(*a*, *b*), *c*)) = collect-value(*args*, *c*))

THEOREM: collect-value-litatom-indices-speedup
 litatom(*a*)
 $\rightarrow$ (collect-value(indices(*name*, *from*, *to*), cons(cons(*a*, *b*), *c*))
 = collect-value(indices(*name*, *from*, *to*), *c*))

THEOREM: equal-collect-value-promote-alists
 (*alist1* = *alist2*)
 $\rightarrow$ ((collect-value(*args*, *alist1*) = collect-value(*args*, *alist2*)) = **t**)

THEOREM: collect-value-firstn
 ($\neg$ duplicates?(*args*))
 $\rightarrow$ (collect-value(firstn(*n*, *args*), pairlist(*args*, *answer*))
 = collect-value(firstn(*n*, *args*),
 pairlist(firstn(*n*, *args*), firstn(*n*, *answer*))))

THEOREM: collect-value-restn
 $(\neg \text{duplicates?}(args))$
 $\rightarrow (\text{collect-value}(\text{restn}(n, args), \text{pairlist}(args, answer)))$
 $= \text{collect-value}(\text{restn}(n, args),$
 $\text{pairlist}(\text{restn}(n, args), \text{restn}(n, answer)))$

THEOREM: collect-value-cons
 $\text{collect-value}(\text{cons}(a, b), alist)$
 $= \text{cons}(\text{value}(a, alist), \text{collect-value}(b, alist))$

THEOREM: singleton-collect-value
 $(\text{length}(a) = 1)$
 $\rightarrow (\text{collect-value}(a, alist) = \text{list}(\text{value}(\text{car}(a), alist)))$

THEOREM: properp-collect-value
 $\text{properp}(\text{collect-value}(args, alist))$

THEOREM: collect-value-args-pairlist-args
 $((\neg \text{duplicates?}(args))$
 $\wedge (\text{length}(args) = \text{length}(list))$
 $\wedge \text{properp}(list))$
 $\rightarrow (\text{collect-value}(args, \text{pairlist}(args, list)) = list)$

THEOREM: collect-value-splitting-crock-helper
 $\text{collect-value}(\text{append}(\text{firstn}(n, l), \text{restn}(n, l)), alist)$
 $= \text{append}(\text{collect-value}(\text{firstn}(n, l), alist),$
 $\text{collect-value}(\text{restn}(n, l), alist))$

THEOREM: collect-value-splitting-crock
 $\text{collect-value}(l, alist)$
 $= \text{append}(\text{collect-value}(\text{firstn}(n, l), alist),$
 $\text{collect-value}(\text{restn}(n, l), alist))$

THEOREM: collect-value-make-list
 $\text{collect-value}(\text{make-list}(n, name), alist) = \text{make-list}(n, \text{value}(name, alist))$

DEFINITION:
 $\text{cdr-cdr-sub1-sub1-induction}(l1, n, m, l2)$
 $= \text{if } m < n \text{ then } t$
 $\text{elseif } n \simeq 0$
 $\text{then if } m \simeq 0 \text{ then } t$
 $\text{else } \text{cdr-cdr-sub1-sub1-induction}(\text{cdr}(l1), 0, m - 1, \text{cdr}(l2)) \text{ endif}$
 $\text{else } \text{cdr-cdr-sub1-sub1-induction}(\text{cdr}(l1), n - 1, m - 1, \text{cdr}(l2)) \text{ endif}$

THEOREM: collect-value-subrange-args-pairlist-args

$$\begin{aligned}
& ((\neg \text{duplicates?}(args)) \\
& \wedge (\text{length}(args) = \text{length}(list)) \\
& \wedge \text{properp}(list) \\
& \wedge (n < \text{length}(args))) \\
\rightarrow & (\text{collect-value}(\text{subrange}(args, m, n), \text{pairlist}(args, list)) \\
& = \text{subrange}(list, m, n))
\end{aligned}$$

EVENT: Add the shell *rom*, with recognizer function symbol *romp* and 1 accessor: *rom-guts*, with type restriction (none-of) and default value zero.

EVENT: Add the shell *ram*, with recognizer function symbol *ramp* and 1 accessor: *ram-guts*, with type restriction (none-of) and default value zero.

EVENT: Add the shell *stub*, with recognizer function symbol *stubp* and 1 accessor: *stub-guts*, with type restriction (none-of) and default value zero.

DEFINITION:

```

memory-properp(n, width, mem)
=  if stubp(mem)
    then properp(stub-guts(mem))  $\wedge$  (length(stub-guts(mem)) = width)
    elseif n  $\simeq$  0
    then if ramp(mem)
        then properp(ram-guts(mem))
             $\wedge$  (length(ram-guts(mem)) = width)
        elseif romp(mem)
        then properp(rom-guts(mem))
             $\wedge$  (length(rom-guts(mem)) = width)
        else f endif
    else listp(mem)
         $\wedge$  memory-properp(n - 1, width, car(mem))
         $\wedge$  memory-properp(n - 1, width, cdr(mem)) endif

```

EVENT: Enable memory-properp; name this event ‘memory-properp-off’.

DEFINITION:

```

memory-okp(n, width, mem)
=  if stubp(mem)
    then bvp(stub-guts(mem))  $\wedge$  (length(stub-guts(mem)) = width)
    elseif n  $\simeq$  0
    then if ramp(mem)
        then bvp(ram-guts(mem))  $\wedge$  (length(ram-guts(mem)) = width)

```

```

    elseif romp(mem)
    then bvp(rom-guts(mem))  $\wedge$  (length(rom-guts(mem)) = width)
    else endif
else listp(mem)
 $\wedge$  memory-okp( $n - 1$ , width, car(mem))
 $\wedge$  memory-okp( $n - 1$ , width, cdr(mem)) endif

```

EVENT: Enable memory-okp; name this event ‘memory-okp-off’.

DEFINITION:

```

read-mem1(v-addr, mem)
= if stubp(mem) then stub-guts(mem)
  elseif v-addr  $\simeq$  nil
  then if ramp(mem) then ram-guts(mem)
        elseif romp(mem) then rom-guts(mem)
        else 0 endif
  elseif mem  $\simeq$  nil then 0
  elseif car(v-addr) then read-mem1(cdr(v-addr), cdr(mem))
  else read-mem1(cdr(v-addr), car(mem)) endif

```

EVENT: Enable read-mem1; name this event ‘read-mem1-off’.

DEFINITION:

```

read-mem(v-addr, mem) = read-mem1(reverse(v-addr), mem)

```

EVENT: Enable read-mem; name this event ‘read-mem-off’.

DEFINITION:

```

write-mem1(v-addr, mem, value)
= if stubp(mem) then mem
  elseif v-addr  $\simeq$  nil
  then if ramp(mem) then ram(value)
        else mem endif
  elseif mem  $\simeq$  nil then mem
  elseif car(v-addr)
  then cons(car(mem), write-mem1(cdr(v-addr), cdr(mem), value))
  else cons(write-mem1(cdr(v-addr), car(mem), value), cdr(mem)) endif

```

EVENT: Enable write-mem1; name this event ‘write-mem1-off’.

DEFINITION:

```

write-mem(v-addr, mem, value) = write-mem1(reverse(v-addr), mem, value)

```

EVENT: Enable write-mem; name this event ‘write-mem-off’.

DEFINITION:

ramp-mem1 ($v\text{-}addr$, mem)
= **if** stubp(mem) **then f**
 elseif $v\text{-}addr \simeq \text{nil}$ **then** ramp(mem)
 elseif $mem \simeq \text{nil}$ **then f**
 elseif car($v\text{-}addr$) **then** ramp-mem1 (cdr($v\text{-}addr$), cdr(mem))
 else ramp-mem1 (cdr($v\text{-}addr$), car(mem)) **endif**

EVENT: Enable ramp-mem1; name this event ‘ramp-mem1-off’.

DEFINITION:

ramp-mem ($v\text{-}addr$, mem) = ramp-mem1 (reverse($v\text{-}addr$), mem)

EVENT: Enable ramp-mem; name this event ‘ramp-mem-off’.

DEFINITION:

all-ramp-mem (n , mem)
= **if** stubp(mem) **then f**
 elseif $n \simeq 0$ **then** ramp(mem)
 elseif $mem \simeq \text{nil}$ **then f**
 else all-ramp-mem ($n - 1$, car(mem))
 $\wedge$ all-ramp-mem ($n - 1$, cdr(mem)) **endif**

EVENT: Enable all-ramp-mem; name this event ‘all-ramp-mem-off’.

DEFINITION:

constant-ram (mem , $value$)
= **if** ramp(mem) **then** ram($value$)
 elseif $mem \simeq \text{nil}$ **then** mem
 else cons(constant-ram (car(mem), $value$),
 constant-ram (cdr(mem), $value$)) **endif**

EVENT: Enable constant-ram; name this event ‘constant-ram-off’.

THEOREM: memory-properp-if

(memory-properp (n , $width$, a) $\wedge$ memory-properp (n , $width$, b))
 $\rightarrow$ memory-properp (n ,
 $width$,
 if c **then** a
 else b **endif**)

THEOREM: memory-okp-if
 (memory-okp (n , $width$, a) $\wedge$ memory-okp (n , $width$, b))
 $\rightarrow$ memory-okp (n ,
 $width$,
 if c **then** a
 else b **endif**)

THEOREM: memory-properp-constant-ram
 (memory-properp (n , $width$, mem)
 $\wedge$ properp ($value$)
 $\wedge$ ($width = \text{length}(value)$))
 $\rightarrow$ memory-properp (n , $width$, constant-ram (mem , $value$))

THEOREM: memory-properp-after-write-mem1
 (memory-properp (n , $width$, mem)
 $\wedge$ properp ($value$)
 $\wedge$ ($width = \text{length}(value)$)
 $\wedge$ ($n = \text{length}(v\text{-}addr)$))
 $\rightarrow$ memory-properp (n , $width$, write-mem1 ($v\text{-}addr$, mem , $value$))

THEOREM: memory-properp-after-write-mem
 (memory-properp (n , $width$, mem)
 $\wedge$ properp ($value$)
 $\wedge$ ($width = \text{length}(value)$)
 $\wedge$ ($n = \text{length}(v\text{-}addr)$))
 $\rightarrow$ memory-properp (n , $width$, write-mem ($v\text{-}addr$, mem , $value$))

THEOREM: memory-okp-after-write-mem1
 (memory-okp (n , $width$, mem)
 $\wedge$ bvp ($value$)
 $\wedge$ ($width = \text{length}(value)$)
 $\wedge$ ($n = \text{length}(v\text{-}addr)$))
 $\rightarrow$ memory-okp (n , $width$, write-mem1 ($v\text{-}addr$, mem , $value$))

THEOREM: memory-okp-after-write-mem
 (memory-okp (n , $width$, mem)
 $\wedge$ bvp ($value$)
 $\wedge$ ($width = \text{length}(value)$)
 $\wedge$ ($n = \text{length}(v\text{-}addr)$))
 $\rightarrow$ memory-okp (n , $width$, write-mem ($v\text{-}addr$, mem , $value$))

THEOREM: v-iff-v-addr1-v-addr2-read-mem1-write-mem1
 (v-iff ($v\text{-}addr1$, $v\text{-}addr2$)
 $\wedge$ ramp-mem1 ($v\text{-}addr2$, mem)
 $\wedge$ ($\text{length}(v\text{-}addr1) = \text{length}(v\text{-}addr2)$))
 $\rightarrow$ (read-mem1 ($v\text{-}addr1$, write-mem1 ($v\text{-}addr2$, mem , $value$)) = $value$)

THEOREM: v-iff-v-addr1-v-addr2-read-mem1-write-mem1-not-ram
 $((\neg \text{ramp-mem1}(v\text{-addr2}, mem)) \wedge (\text{length}(v\text{-addr1}) = \text{length}(v\text{-addr2})))$
 $\rightarrow (\text{read-mem1}(v\text{-addr1}, \text{write-mem1}(v\text{-addr2}, mem, value))$
 $= \text{read-mem1}(v\text{-addr1}, mem))$

THEOREM: not-v-iff-v-addr1-v-addr2-read-mem1-write-mem1
 $(\neg \text{v-iff}(v\text{-addr1}, v\text{-addr2}))$
 $\rightarrow (\text{read-mem1}(v\text{-addr1}, \text{write-mem1}(v\text{-addr2}, mem, value))$
 $= \text{read-mem1}(v\text{-addr1}, mem))$

THEOREM: read-mem-write-mem
 $(\text{length}(v\text{-addr1}) = \text{length}(v\text{-addr2}))$
 $\rightarrow (\text{read-mem}(v\text{-addr1}, \text{write-mem}(v\text{-addr2}, mem, value))$
 $= \text{if } \text{v-iff}(v\text{-addr1}, v\text{-addr2}) \wedge \text{ramp-mem}(v\text{-addr2}, mem)$
 $\text{ then } value$
 $\text{ else read-mem}(v\text{-addr1}, mem) \text{ endif})$

THEOREM: properp-read-mem1
 $\text{memory-properp}(\text{length}(v\text{-addr}), size, mem)$
 $\rightarrow (\text{properp}(\text{read-mem1}(v\text{-addr}, mem))$
 $\wedge (\text{length}(\text{read-mem1}(v\text{-addr}, mem)) = size))$

THEOREM: properp-read-mem
 $\text{memory-properp}(\text{length}(\text{reverse}(v\text{-addr})), size, mem)$
 $\rightarrow (\text{properp}(\text{read-mem}(v\text{-addr}, mem))$
 $\wedge (\text{length}(\text{read-mem}(v\text{-addr}, mem)) = size))$

THEOREM: properp-read-mem-32
 $\text{memory-properp}(\text{length}(v\text{-addr}), 32, mem)$
 $\rightarrow (\text{properp}(\text{read-mem}(v\text{-addr}, mem))$
 $\wedge (\text{length}(\text{read-mem}(v\text{-addr}, mem)) = 32))$

THEOREM: bvp-read-mem1
 $\text{memory-okp}(\text{length}(v\text{-addr}), size, mem)$
 $\rightarrow (\text{bvp}(\text{read-mem1}(v\text{-addr}, mem))$
 $\wedge (\text{length}(\text{read-mem1}(v\text{-addr}, mem)) = size))$

THEOREM: bvp-read-mem
 $\text{memory-okp}(\text{length}(\text{reverse}(v\text{-addr})), size, mem)$
 $\rightarrow (\text{bvp}(\text{read-mem}(v\text{-addr}, mem))$
 $\wedge (\text{length}(\text{read-mem}(v\text{-addr}, mem)) = size))$

THEOREM: bvp-read-mem-32
 $\text{memory-okp}(\text{length}(v\text{-addr}), 32, mem)$
 $\rightarrow (\text{bvp}(\text{read-mem}(v\text{-addr}, mem)) \wedge (\text{length}(\text{read-mem}(v\text{-addr}, mem)) = 32))$

THEOREM: all-ramp-mem->ramp-mem1
all-ramp-mem (length (*v-addr*), *mem*) → ramp-mem1 (*v-addr*, *mem*)

THEOREM: all-ramp-mem->ramp-mem
all-ramp-mem (length (*v-addr*), *mem*) → ramp-mem (*v-addr*, *mem*)

THEOREM: all-ramp-mem-after-write-mem1
(all-ramp-mem (*n*, *mem*) ∧ (*n* = length (*v-addr*)))
→ all-ramp-mem (*n*, write-mem1 (*v-addr*, *mem*, *value*))

THEOREM: all-ramp-mem-after-write-mem
(all-ramp-mem (*n*, *mem*) ∧ (*n* = length (*v-addr*)))
→ all-ramp-mem (*n*, write-mem (*v-addr*, *mem*, *value*))

THEOREM: all-ramp-mem-constant-ram
all-ramp-mem (*n*, constant-ram (*mem*, *value*)) = all-ramp-mem (*n*, *mem*)

THEOREM: memory-okp==>memory-properp
memory-okp (*n*, *m*, *mem*) → memory-properp (*n*, *m*, *mem*)

DEFINITION:

dual-port-ram-value (*bits*, *address-lines*, *args*, *state*)
= **if** (¬ bvp (subrange (*args*, 0, *address-lines* - 1)))
 ∨ ((nth (2 * *address-lines*, *args*) ≠ **t**)
 ∧ ((¬ bvp (subrange (*args*,
 address-lines,
 (2 * *address-lines*) - 1)))
 ∨ (subrange (*args*, 0, *address-lines* - 1)
 = subrange (*args*,
 address-lines,
 (2 * *address-lines*) - 1))))
 then make-list (*bits*, x)
 elseif properp (read-mem (subrange (*args*, 0, *address-lines* - 1), *state*))
 ∧ (length (read-mem (subrange (*args*, 0, *address-lines* - 1),
 state))
 = *bits*)
 then read-mem (subrange (*args*, 0, *address-lines* - 1), *state*)
 else make-list (*bits*, x) **endif**

DEFINITION:

dual-port-ram-state (*bits*, *address-lines*, *args*, *state*)
= **if** nth (2 * *address-lines*, *args*) = **t** **then** *state*
 elseif ¬ bvp (subrange (*args*, *address-lines*, (2 * *address-lines*) - 1))
 then constant-ram (*state*, make-list (*bits*, x))
 elseif nth (2 * *address-lines*, *args*) = **f**

```

then write-mem (subrange (args, address-lines, (2 * address-lines) - 1),
                      state,
                      subrange (args,
                                1 + (2 * address-lines),
                                (2 * address-lines) + bits))
else write-mem (subrange (args,
                          address-lines,
                          (2 * address-lines) - 1),
                  state,
                  make-list (bits, x)) endif

```

THEOREM: properp-length-dual-port-ram-value
properp (dual-port-ram-value (bits, address-lines, args, state))
 $\wedge$ (length (dual-port-ram-value (bits, address-lines, args, state))
= fix (bits))

DEFINITION:

```

next-memory-state (state, strobe-, rw-, address, data)
= if boolp (strobe-)  $\wedge$  boolp (rw-)
  then case on cadr (state):
    case = 0
    then if strobe-
      then list (car (state),
                0,
                caddr (state),
                0,
                t,
                threefix (rw-),
                v-threefix (address),
                if boolp (rw-)
                  then if rw- then v-threefix (caddr (state))
                  else v-threefix (data) endif
                else make-list (length (data), x) endif)
    elseif rw-
      then if caddr (state)  $\wedge$  boolp (caddr (state))
        then if bvp (address)
           $\wedge$  bvp (caddr (state))
           $\wedge$  (address = caddr (state))
          then list (car (state),
                    1,
                    cdaddr (state),
                    caaddr (state) - 1,
                    caaddr (state)  $\simeq$  0,
                    threefix (rw-),

```

```

v-threefix (address),
if boolp (rw-)
then if rw-
    then v-threefix (caddddddd (state))
    else v-threefix (data) endif
else make-list (length (data), x) endif
else list (car (state),
    3,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw-
        then v-threefix (caddddddd (state))
        else v-threefix (data) endif
    else make-list (length (data),
        x) endif endif
else list (constant-ram (car (state),
    make-list (32, x)),
    3,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw-
        then v-threefix (caddddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif endif
elseif ( $\neg$  caddddddd (state))
     $\wedge$  (bvp (address)
     $\wedge$  bvp (caddddddd (state))
     $\wedge$  (address = caddddddd (state)))
     $\wedge$  bvp (data)
     $\wedge$  bvp (caddddddd (state))
     $\wedge$  (data = caddddddd (state))
then list (car (state),
    2,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,

```



```

    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw- then v-threefix (caddddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif
else list (constant-ram (car (state), make-list (32, x)),
    3,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw-
        then v-threefix (caddddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif) endif
case = 1
then if strobe-
    then if rw-
        then list (car (state),
            0,
            caddr (state),
            0,
            t,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddddddd (state))
                else v-threefix (data) endif
            else make-list (length (data), x) endif)
        else list (constant-ram (car (state),
            make-list (32, x)),
            0,
            caddr (state),
            0,
            t,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddddddd (state))

```

```

                                else v-threefix(data) endif
                                else make-list(length(data), x) endif) endif
elseif rw-
then if bvp(address)
    ∧ bvp(caddddddd(state))
    ∧ (address = caddddddd(state))
    then list(car(state),
              1,
              caddr(state),
              caddr(state) - 1,
              caddr(state)  $\simeq$  0,
              threefix(rw-),
              v-threefix(address),
              if boolp(rw-)
              then if rw-
                  then v-threefix(caddddddd(state))
                  else v-threefix(data) endif
                  else make-list(length(data), x) endif)
    else list(car(state),
              3,
              cdaddr(state),
              caaddr(state) - 1,
              caaddr(state)  $\simeq$  0,
              threefix(rw-),
              v-threefix(address),
              if boolp(rw-)
              then if rw-
                  then v-threefix(caddddddd(state))
                  else v-threefix(data) endif
                  else make-list(length(data), x) endif) endif
    else list(constant-ram(car(state), make-list(32, x)),
              3,
              cdaddr(state),
              caaddr(state) - 1,
              caaddr(state)  $\simeq$  0,
              threefix(rw-),
              v-threefix(address),
              if boolp(rw-)
              then if rw-
                  then v-threefix(caddddddd(state))
                  else v-threefix(data) endif
                  else make-list(length(data), x) endif) endif
case = 2
then if strobe-

```

```

then if rw-
  then list (constant-ram (car (state), make-list (32, x)),
    0,
    caddr (state),
    0,
    t,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
      then if rw-
        then v-threefix (caddrdddr (state))
        else v-threefix (data) endif
      else make-list (length (data), x) endif
    elseif (bvp (address)
       $\wedge$  bvp (caddrdddr (state))
       $\wedge$  (address = caddrdddr (state)))
       $\wedge$  (bvp (data)
         $\wedge$  bvp (caddrdddr (state))
         $\wedge$  (data = caddrdddr (state)))
       $\wedge$  (caddr (state)  $\simeq$  0)
    then list (write-mem (address, car (state), data),
      0,
      caddr (state),
      0,
      t,
      threefix (rw-),
      v-threefix (address),
      if boolp (rw-)
        then if rw-
          then v-threefix (caddrdddr (state))
          else v-threefix (data) endif
        else make-list (length (data), x) endif
    else list (constant-ram (car (state),
      make-list (32, x)),
      0,
      caddr (state),
      0,
      t,
      threefix (rw-),
      v-threefix (address),
      if boolp (rw-)
        then if rw-
          then v-threefix (caddrdddr (state))
          else v-threefix (data) endif

```

```

        else make-list (length (data), x) endif) endif
elseif rw-
then list (constant-ram (car (state), make-list (32, x)),
          3,
          cdaddr (state),
          caaddr (state) - 1,
          caaddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw- then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif)
elseif (bvp (address)
         $\wedge$  bvp (caddddddd (state))
         $\wedge$  (address = caddddddd (state)))
         $\wedge$  bvp (data)
         $\wedge$  bvp (caddddddd (state))
         $\wedge$  (data = caddddddd (state))
then list (car (state),
          2,
          caddr (state),
          caddr (state) - 1,
          caddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw- then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif)
else list (constant-ram (car (state), make-list (32, x)),
          3,
          cdaddr (state),
          caaddr (state) - 1,
          caaddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw-
                  then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif) endif
otherwise if strobe-
then if rw-

```

```

then list (car (state),
           0,
           caddr (state),
           0,
           t,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data), x) endif)
else list (constant-ram (car (state),
                        make-list (32, x)),
           0,
           caddr (state),
           0,
           t,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data),
                        x) endif) endif
elseif rw-
then list (car (state),
           3,
           caddr (state),
           caddr (state) - 1,
           caddr (state)  $\simeq$  0,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data), x) endif)
else list (constant-ram (car (state),
                        make-list (32, x)),
           3,
           cdaddr (state),
           caaddr (state) - 1,

```

```

caaddr (state)  $\simeq$  0,
threefix (rw-),
v-threefix (address),
if boolp (rw-)
then if rw-
    then v-threefix (caddddddd (state))
    else v-threefix (data) endif
else make-list (length (data), x) endif endif endcase
else list (constant-ram (car (state), make-list (32, x)),
0,
caddr (state),
0,
t,
threefix (rw-),
v-threefix (address),
if boolp (rw-)
then if rw- then v-threefix (caddddddd (state))
    else v-threefix (data) endif
else make-list (length (data), x) endif endif

```

EVENT: Enable next-memory-state; name this event 'next-memory-state-off'.

DEFINITION:

memory-value (state, strobe-, rw-, address, data)

```

= if* (and* (boolp (strobe-), boolp (rw-)),
case on cadr (state):
case = 0
then if* (strobe-,
    cons (X, make-list (length (data), z)),
    cons (if* (caaddr (state)  $\simeq$  0, f, t),
        if* (rw-,
            make-list (length (data), x),
            make-list (length (data), z))))
case = 1
then if* (strobe-,
    cons (X, make-list (length (data), z)),
    if* (rw-,
        if* (and* (address = caddddddd (state),
            and* (bvp (address), bvp (caddddddd (state)))),
            if* (caddddr (state)  $\simeq$  0,
                cons (f,
                    if* (caddddr (state),
                        read-mem (address, car (state)),
                        make-list (length (data), x))),

```

```

        cons (t, make-list (length (data), x))),
    if* (caddr (state)  $\simeq$  0,
        cons (f, make-list (length (data), x)),
        cons (t, make-list (length (data), x))),
    if* (caddr (state)  $\simeq$  0,
        cons (f, make-list (length (data), z)),
        cons (t, make-list (length (data), z))))
case = 2
    then if* (strobe-,
        cons (x, make-list (length (data), z)),
        if* (caddr (state)  $\simeq$  0,
            cons (f, make-list (length (data), z)),
            cons (t, make-list (length (data), z))))
    otherwise if* (strobe-,
        cons (x, make-list (length (data), z)),
        if* (rw-,
            if* (caddr (state)  $\simeq$  0,
                cons (f, make-list (length (data), x)),
                cons (t, make-list (length (data), x))),
            if* (caddr (state)  $\simeq$  0,
                cons (f, make-list (length (data), z)),
                cons (t, make-list (length (data), z)))))) endcase,
    cons (x, make-list (length (data), x))

```

EVENT: Enable memory-value; name this event ‘memory-value-off’.

DEFINITION:

```

mem-value (args, state)
=   memory-value (state,
    cadr (args),
    car (args),
    subrange (args, 2, 33),
    subrange (args, 34, 65))

```

EVENT: Enable mem-value; name this event ‘mem-value-off’.

DEFINITION:

```

mem-state (args, state)
=   next-memory-state (state,
    cadr (args),
    car (args),
    subrange (args, 2, 33),
    subrange (args, 34, 65))

```

EVENT: Enable mem-state; name this event ‘mem-state-off’.

THEOREM: properp-length-memory-value
 (memory-properp (32, 32, car (state))
 $\wedge$ (length (address) = 32)
 $\wedge$ (length (data) = 32))
 $\rightarrow$ (properp (cdr (memory-value (state, strobe, rw-, address, data)))
 $\wedge$ (length (cdr (memory-value (state, strobe, rw-, address, data)))
 = 32))

THEOREM: equal-memory-value
 (length (data1) = length (data2))
 $\rightarrow$ ((memory-value (state, strobe, rw-, address, data1)
 = memory-value (state, strobe, rw-, address, data2))
 = t)

DEFINITION:

next-memory-state\$induction (state, strobe-, rw-, address, data)
 = **if** boolp (strobe-) $\wedge$ boolp (rw-)
 then case on cadr (state):
 case = 0
 then if strobe-
 then list (car (state),
 0,
 caddr (state),
 0,
 t,
 threefix (rw-),
 v-threefix (address),
 if boolp (rw-)
 then if rw- **then** v-threefix (caddddddr (state))
 else v-threefix (data) **endif**
 else make-list (length (data), x) **endif**)
 elseif rw-
 then if cadddddr (state) $\wedge$ boolp (cadddddr (state))
 then if bvp (address)
 $\wedge$ bvp (cadddddr (state))
 $\wedge$ (address = cadddddr (state))
 then list (car (state),
 1,
 cdaddr (state),
 caaddr (state) - 1,
 caaddr (state) $\simeq$ 0,
 threefix (rw-),


```

        v-threefix (address),
        if boolp (rw-)
        then if rw-
            then v-threefix (caddddddd (state))
            else v-threefix (data) endif
        else make-list (length (data), x) endif
    else list (car (state),
        3,
        cdaddr (state),
        caaddr (state) - 1,
        caaddr (state)  $\simeq$  0,
        threefix (rw-),
        v-threefix (address),
        if boolp (rw-)
        then if rw-
            then v-threefix (caddddddd (state))
            else v-threefix (data) endif
        else make-list (length (data),
            x) endif endif
    else list (constant-ram (car (state),
        make-list (32, x)),
        3,
        cdaddr (state),
        caaddr (state) - 1,
        caaddr (state)  $\simeq$  0,
        threefix (rw-),
        v-threefix (address),
        if boolp (rw-)
        then if rw-
            then v-threefix (caddddddd (state))
            else v-threefix (data) endif
        else make-list (length (data), x) endif) endif
elseif ( $\neg$  caddddddd (state))
     $\wedge$  (bvp (address)
         $\wedge$  bvp (caddddddd (state))
         $\wedge$  (address = caddddddd (state)))
     $\wedge$  bvp (data)
     $\wedge$  bvp (caddddddd (state))
     $\wedge$  (data = caddddddd (state))
then list (car (state),
    2,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,

```

```

    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw- then v-threefix (caddrddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif
else list (constant-ram (car (state), make-list (32, x)),
    3,
    cdaddr (state),
    caaddr (state) - 1,
    caaddr (state)  $\simeq$  0,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw-
        then v-threefix (caddrddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif) endif
case = 1
then if strobe-
    then if rw-
        then list (car (state),
            0,
            caddr (state),
            0,
            t,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddrddddd (state))
                else v-threefix (data) endif
            else make-list (length (data), x) endif)
        else list (constant-ram (car (state),
            make-list (32, x)),
            0,
            caddr (state),
            0,
            t,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddrddddd (state))

```

```

                                else v-threefix (data) endif
                                else make-list (length (data), x) endif) endif
elseif rw-
then if bvp (address)
    ∧ bvp (caddddddd (state))
    ∧ (address = caddddddd (state))
then list (car (state),
            1,
            caddr (state),
            caddr (state) - 1,
            caddr (state)  $\simeq$  0,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddddddd (state))
                else v-threefix (data) endif
            else make-list (length (data), x) endif)
else list (car (state),
            3,
            cdaddr (state),
            caaddr (state) - 1,
            caaddr (state)  $\simeq$  0,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddddddd (state))
                else v-threefix (data) endif
            else make-list (length (data), x) endif) endif
else list (constant-ram (car (state), make-list (32, x)),
            3,
            cdaddr (state),
            caaddr (state) - 1,
            caaddr (state)  $\simeq$  0,
            threefix (rw-),
            v-threefix (address),
            if boolp (rw-)
            then if rw-
                then v-threefix (caddddddd (state))
                else v-threefix (data) endif
            else make-list (length (data), x) endif) endif
case = 2
then if strobe-

```

```

then if rw-
  then list (constant-ram (car (state), make-list (32, x)),
    0,
    caddr (state),
    0,
    t,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
      then if rw-
        then v-threefix (caddrddddd (state))
        else v-threefix (data) endif
      else make-list (length (data), x) endif
    elseif (bvp (address)
       $\wedge$  bvp (caddrddddd (state))
       $\wedge$  (address = caddrddddd (state)))
       $\wedge$  (bvp (data)
         $\wedge$  bvp (caddrddddd (state))
         $\wedge$  (data = caddrddddd (state)))
       $\wedge$  (caddr (state)  $\simeq$  0)
    then list (write-mem (address, car (state), data),
      0,
      caddr (state),
      0,
      t,
      threefix (rw-),
      v-threefix (address),
      if boolp (rw-)
        then if rw-
          then v-threefix (caddrddddd (state))
          else v-threefix (data) endif
        else make-list (length (data), x) endif
    else list (constant-ram (car (state),
      make-list (32, x)),
      0,
      caddr (state),
      0,
      t,
      threefix (rw-),
      v-threefix (address),
      if boolp (rw-)
        then if rw-
          then v-threefix (caddrddddd (state))
          else v-threefix (data) endif

```

```

        else make-list (length (data), x) endif) endif
elseif rw-
then list (constant-ram (car (state), make-list (32, x)),
          3,
          cdaddr (state),
          caaddr (state) - 1,
          caaddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw- then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif)
elseif (bvp (address)
         $\wedge$  bvp (caddddddd (state))
         $\wedge$  (address = caddddddd (state)))
         $\wedge$  bvp (data)
         $\wedge$  bvp (caddddddd (state))
         $\wedge$  (data = caddddddd (state))
then list (car (state),
          2,
          caddr (state),
          caddr (state) - 1,
          caddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw- then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif)
else list (constant-ram (car (state), make-list (32, x)),
          3,
          cdaddr (state),
          caaddr (state) - 1,
          caaddr (state)  $\simeq$  0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
            then if rw-
                  then v-threefix (caddddddd (state))
                  else v-threefix (data) endif
            else make-list (length (data), x) endif) endif
otherwise if strobe-
          then if rw-

```

```

then list (car (state),
           0,
           caddr (state),
           0,
           t,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data), x) endif)
else list (constant-ram (car (state),
                        make-list (32, x)),
           0,
           caddr (state),
           0,
           t,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data),
                        x) endif) endif
elseif rw-
then list (car (state),
           3,
           caddr (state),
           caddr (state) - 1,
           caddr (state)  $\simeq$  0,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
               then v-threefix (caddrdddr (state))
               else v-threefix (data) endif
           else make-list (length (data), x) endif)
else list (constant-ram (car (state),
                        make-list (32, x)),
           3,
           cdaddr (state),
           caaddr (state) - 1,

```

```

caaddr (state)  $\simeq$  0,
threefix (rw-),
v-threefix (address),
if boolp (rw-)
then if rw-
    then v-threefix (caddddddd (state))
    else v-threefix (data) endif
else make-list (length (data), x) endif endcase
elseif  $\neg$  (boolp (strobe-)  $\wedge$  boolp (rw-))
then list (constant-ram (car (state), make-list (32, x)),
    0,
    caddr (state),
    0,
    t,
    threefix (rw-),
    v-threefix (address),
    if boolp (rw-)
    then if rw- then v-threefix (caddddddd (state))
        else v-threefix (data) endif
    else make-list (length (data), x) endif)
else next-memory-state$induction (state - 1,
    strobe-,
    rw-,
    address,
    data) endif

```

THEOREM: properp-length-next-memory-state
properp (next-memory-state (state, strobe-, rw-, address, data))
 $\wedge$ (length (next-memory-state (state, strobe-, rw-, address, data)) = 8)

DEFINITION:

```

fix-breadth-tree-stack (stack, n)
= if stack  $\simeq$  nil then nil
  else cons (append (make-list (n, f), car (stack)),
    fix-breadth-tree-stack (cdr (stack), 2 * n)) endif

```

DEFINITION:

```

breadth-tree (tree, stack, n)
= if tree  $\simeq$  nil
  then cons (cons (t,
    if stack  $\simeq$  nil then make-list (n, f)
    else car (stack) endif),
    fix-breadth-tree-stack (cdr (stack), 2))
  else cons (cons (t,
    if stack  $\simeq$  nil then make-list (n, f)

```

```

      else car(stack) endif),
breadth-tree(cdr(tree),
  breadth-tree(car(tree),
    cdr(stack),
      2 * n),
    1 + (2 * n))) endif

```

DEFINITION:

```

collect-breadth-tree(stack, n)
= if stack  $\simeq$  nil then nil
  elseif car(stack) = make-list(n, t)
  then cons(t, collect-breadth-tree(cdr(stack), 2 * n))
  else cons(f,
    append(car(stack),
      collect-breadth-tree(cdr(stack), 2 * n))) endif

```

DEFINITION:

```

tree-number(tree)
= ((1 + v-to-nat(collect-breadth-tree(breadth-tree(tree, nil, 0), 1)))
  ÷ 2)

```

EVENT: Enable tree-number; name this event ‘tree-number-off’.

DEFINITION: $\text{f-buf}(a) = \text{threefix}(a)$

DEFINITION:

```

f-and(a, b)
= if (a = f)  $\vee$  (b = f) then f
  elseif (a = t)  $\wedge$  (b = t) then t
  else x endif

```

DEFINITION: $\text{f-and3}(a, b, c) = \text{f-and}(a, \text{f-and}(b, c))$

DEFINITION: $\text{f-and4}(a, b, c, d) = \text{f-and}(a, \text{f-and}(b, \text{f-and}(c, d)))$

DEFINITION:

```

f-not(a)
= if boolp(a) then  $\neg a$ 
  else x endif

```

DEFINITION: $\text{f-nand}(a, b) = \text{f-not}(\text{f-and}(a, b))$

DEFINITION: $\text{f-nand3}(a, b, c) = \text{f-not}(\text{f-and}(a, \text{f-and}(b, c)))$

DEFINITION:

$\text{f-nand4}(a, b, c, d) = \text{f-not}(\text{f-and}(a, \text{f-and}(b, \text{f-and}(c, d))))$

DEFINITION:

$\text{f-nand5}(a, b, c, d, e) = \text{f-not}(\text{f-and}(a, \text{f-and}(b, \text{f-and}(c, \text{f-and}(d, e))))))$

DEFINITION:

$\text{f-nand6}(a, b, c, d, e, g)$
 $= \text{f-not}(\text{f-and}(a, \text{f-and}(b, \text{f-and}(c, \text{f-and}(d, \text{f-and}(e, g))))))$

DEFINITION:

$\text{f-nand8}(a, b, c, d, e, g, h, i)$
 $= \text{f-not}(\text{f-and}(a, \text{f-and}(b, \text{f-and}(c, \text{f-and}(d, \text{f-and}(e, \text{f-and}(g, \text{f-and}(h, i))))))))$

DEFINITION:

$\text{f-or}(a, b)$
 $= \text{if } (a = \mathbf{t}) \vee (b = \mathbf{t}) \text{ then } \mathbf{t}$
 $\quad \text{elseif } (a = \mathbf{f}) \wedge (b = \mathbf{f}) \text{ then } \mathbf{f}$
 $\quad \text{else } \mathbf{x} \text{ endif}$

DEFINITION: $\text{f-or3}(a, b, c) = \text{f-or}(a, \text{f-or}(b, c))$

DEFINITION: $\text{f-or4}(a, b, c, d) = \text{f-or}(a, \text{f-or}(b, \text{f-or}(c, d)))$

DEFINITION: $\text{f-nor}(a, b) = \text{f-not}(\text{f-or}(a, b))$

DEFINITION: $\text{f-nor3}(a, b, c) = \text{f-not}(\text{f-or}(a, \text{f-or}(b, c)))$

DEFINITION: $\text{f-nor4}(a, b, c, d) = \text{f-not}(\text{f-or}(a, \text{f-or}(b, \text{f-or}(c, d))))$

DEFINITION:

$\text{f-nor5}(a, b, c, d, e) = \text{f-not}(\text{f-or}(a, \text{f-or}(b, \text{f-or}(c, \text{f-or}(d, e))))))$

DEFINITION:

$\text{f-nor6}(a, b, c, d, e, g) = \text{f-not}(\text{f-or}(a, \text{f-or}(b, \text{f-or}(c, \text{f-or}(d, \text{f-or}(e, g))))))$

DEFINITION:

$\text{f-nor8}(a, b, c, d, e, g, h, i)$
 $= \text{f-not}(\text{f-or}(a, \text{f-or}(b, \text{f-or}(c, \text{f-or}(d, \text{f-or}(e, \text{f-or}(g, \text{f-or}(h, i))))))))$

DEFINITION:

$\text{f-xor}(a, b)$
 $= \text{if } a = \mathbf{t} \text{ then } \text{f-not}(b)$
 $\quad \text{elseif } a = \mathbf{f} \text{ then } \text{threefix}(b)$
 $\quad \text{else } \mathbf{x} \text{ endif}$

DEFINITION: $\text{f-xor3}(a, b, c) = \text{f-xor}(a, \text{f-xor}(b, c))$

DEFINITION:

f-equiv(a, b)
 = **if** $a = \mathbf{t}$ **then** threefix(b)
 elseif $a = \mathbf{f}$ **then** f-not(b)
 else x **endif**

DEFINITION: f-equiv3(a, b, c) = f-equiv($a, \text{f-xor}(b, c)$)

DEFINITION:

f-if(c, a, b)
 = **if** $c = \mathbf{t}$ **then** threefix(a)
 elseif $c = \mathbf{f}$ **then** threefix(b)
 else x **endif**

DEFINITION:

ft-buf(c, a)
 = **if** $c = \mathbf{t}$ **then** threefix(a)
 elseif $c = \mathbf{f}$ **then** z
 else x **endif**

DEFINITION:

ft-wire(a, b)
 = **if** $a = b$ **then** fourfix(a)
 elseif $a = z$ **then** fourfix(b)
 elseif $b = z$ **then** fourfix(a)
 else x **endif**

DEFINITION:

f-pullup(a)
 = **if** $a = z$ **then** $\mathbf{t}$
 else threefix(a) **endif**

DEFINITION: f\$ao2(a, b, c, d) = f-nor(f-and(a, b), f-and(c, d))

THEOREM: expand-f-functions

(f-buf(a) = threefix(a))
 $\wedge$ (f-and(a, b)
 = **if** ($a = \mathbf{f}$) $\vee$ ($b = \mathbf{f}$) **then** $\mathbf{f}$
 elseif ($a = \mathbf{t}$) $\wedge$ ($b = \mathbf{t}$) **then** $\mathbf{t}$
 else x **endif**)
 $\wedge$ (f-and3(a, b, c) = f-and($a, \text{f-and}(b, c)$))
 $\wedge$ (f-and4(a, b, c, d) = f-and($a, \text{f-and}(b, \text{f-and}(c, d))$))
 $\wedge$ (f-not(a)
 = **if** boolp(a) **then** $\neg a$
 else x **endif**)

```

 $\wedge$  (f-nand( $a, b$ ) = f-not(f-and( $a, b$ )))
 $\wedge$  (f-nand3( $a, b, c$ ) = f-not(f-and( $a, f$ -and( $b, c$ ))))
 $\wedge$  (f-nand4( $a, b, c, d$ ) = f-not(f-and( $a, f$ -and( $b, f$ -and( $c, d$ ))))))
 $\wedge$  (f-or( $a, b$ )
    = if ( $a = \mathbf{t}$ )  $\vee$  ( $b = \mathbf{t}$ ) then  $\mathbf{t}$ 
      elseif ( $a = \mathbf{f}$ )  $\wedge$  ( $b = \mathbf{f}$ ) then  $\mathbf{f}$ 
      else  $\mathbf{x}$  endif)
 $\wedge$  (f-nand5( $a, b, c, d, e$ ) = f-not(f-and( $a, f$ -and( $b, f$ -and( $c, f$ -and( $d, e$ ))))))
 $\wedge$  (f-nand6( $a, b, c, d, e, g$ )
    = f-not(f-and( $a, f$ -and( $b, f$ -and( $c, f$ -and( $d, f$ -and( $e, g$ ))))))
 $\wedge$  (f-nand8( $a, b, c, d, e, g, h, i$ )
    = f-not(f-and( $a, f$ -and( $b, f$ -and( $c, f$ -and( $d, f$ -and( $e, f$ -and( $g, f$ -and( $h, i$ ))))))))))
 $\wedge$  (f-or3( $a, b, c$ ) = f-or( $a, f$ -or( $b, c$ )))
 $\wedge$  (f-or4( $a, b, c, d$ ) = f-or( $a, f$ -or( $b, f$ -or( $c, d$ ))))
 $\wedge$  (f-nor( $a, b$ ) = f-not(f-or( $a, b$ )))
 $\wedge$  (f-nor3( $a, b, c$ ) = f-not(f-or( $a, f$ -or( $b, c$ ))))
 $\wedge$  (f-nor4( $a, b, c, d$ ) = f-not(f-or( $a, f$ -or( $b, f$ -or( $c, d$ ))))
 $\wedge$  (f-nor5( $a, b, c, d, e$ ) = f-not(f-or( $a, f$ -or( $b, f$ -or( $c, f$ -or( $d, e$ ))))))
 $\wedge$  (f-nor6( $a, b, c, d, e, g$ )
    = f-not(f-or( $a, f$ -or( $b, f$ -or( $c, f$ -or( $d, f$ -or( $e, g$ ))))))
 $\wedge$  (f-nor8( $a, b, c, d, e, g, h, i$ )
    = f-not(f-or( $a, f$ -or( $b, f$ -or( $c, f$ -or( $d, f$ -or( $e, f$ -or( $g, f$ -or( $h, i$ ))))))))))
 $\wedge$  (f-xor( $a, b$ )
    = if  $a = \mathbf{t}$  then f-not( $b$ )
      elseif  $a = \mathbf{f}$  then threefix( $b$ )
      else  $\mathbf{x}$  endif)
 $\wedge$  (f-xor3( $a, b, c$ ) = f-xor( $a, f$ -xor( $b, c$ )))
 $\wedge$  (f-equiv( $a, b$ )
    = if  $a = \mathbf{t}$  then threefix( $b$ )
      elseif  $a = \mathbf{f}$  then f-not( $b$ )
      else  $\mathbf{x}$  endif)
 $\wedge$  (f-equiv3( $a, b, c$ ) = f-equiv( $a, f$ -xor( $b, c$ )))
 $\wedge$  (f-if( $c, a, b$ )
    = if  $c = \mathbf{t}$  then threefix( $a$ )
      elseif  $c = \mathbf{f}$  then threefix( $b$ )
      else  $\mathbf{x}$  endif)
 $\wedge$  (ft-buf( $c, a$ )
    = if  $c = \mathbf{t}$  then threefix( $a$ )
      elseif  $c = \mathbf{f}$  then  $\mathbf{z}$ 
      else  $\mathbf{x}$  endif)

```

$$\begin{aligned}
& \wedge \quad (\text{ft-wire}(a, b) \\
& \quad = \quad \text{if } a = b \text{ then fourfix}(a) \\
& \quad \quad \text{elseif } a = Z \text{ then fourfix}(b) \\
& \quad \quad \text{elseif } b = Z \text{ then fourfix}(a) \\
& \quad \quad \text{else } x \text{ endif}) \\
& \wedge \quad (\text{f-pullup}(a) \\
& \quad = \quad \text{if } a = Z \text{ then } t \\
& \quad \quad \text{else threefix}(a) \text{ endif})
\end{aligned}$$

EVENT: Enable expand-f-functions; name this event ‘expand-f-functions-off’.

THEOREM: f-buf-lemma
 $\text{boolp}(x) \rightarrow (\text{f-buf}(x) = x)$

THEOREM: f-and-rewrite
 $(\neg \text{f-and}(\mathbf{f}, x))$
 $\wedge \quad (\neg \text{f-and}(x, \mathbf{f}))$
 $\wedge \quad (\text{f-and}(x, \mathbf{t}) = \text{f-buf}(x))$
 $\wedge \quad (\text{f-and}(\mathbf{t}, x) = \text{f-buf}(x))$
 $\wedge \quad ((x \wedge y \wedge \text{boolp}(x) \wedge \text{boolp}(y)) \rightarrow (\text{f-and}(x, y) = \mathbf{t}))$

THEOREM: f-or-rewrite
 $(\text{f-or}(\mathbf{f}, \mathbf{f}) = \mathbf{f})$
 $\wedge \quad ((x \wedge \text{boolp}(x)) \rightarrow (\text{f-or}(x, y) = \mathbf{t}))$
 $\wedge \quad ((x \wedge \text{boolp}(x)) \rightarrow (\text{f-or}(y, x) = \mathbf{t}))$

THEOREM: f-not-rewrite
 $(\text{f-not}(\mathbf{t}) = \mathbf{f}) \wedge (\text{f-not}(\mathbf{f}) = \mathbf{t})$

THEOREM: ft-buf-rewrite
 $(\text{ft-buf}(\mathbf{t}, x) = \text{threefix}(x))$
 $\wedge \quad (\text{ft-buf}(\mathbf{f}, x) = Z)$
 $\wedge \quad (\text{ft-buf}(c, \text{threefix}(x)) = \text{ft-buf}(c, x))$
 $\wedge \quad (\text{ft-buf}(X, x) = X)$

THEOREM: f-if-rewrite
 $(\text{f-if}(\mathbf{t}, a, b) = \text{f-buf}(a)) \wedge (\text{f-if}(\mathbf{f}, a, b) = \text{f-buf}(b))$

THEOREM: ft-wire-rewrite
 $(\text{ft-wire}(X, x) = X)$
 $\wedge \quad (\text{ft-wire}(x, X) = X)$
 $\wedge \quad (\text{ft-wire}(x, Z) = \text{fourfix}(x))$
 $\wedge \quad (\text{ft-wire}(Z, x) = \text{fourfix}(x))$

THEOREM: f-pullup-rewrite

$$\begin{aligned} & (\text{f-pullup}(\mathbf{t}) = \mathbf{t}) \\ & \wedge (\text{f-pullup}(\mathbf{f}) = \mathbf{f}) \\ & \wedge (\text{f-pullup}(x) = x) \\ & \wedge (\text{f-pullup}(z) = \mathbf{t}) \end{aligned}$$

THEOREM: threefix-help-lemma

$$\begin{aligned} & (\text{threefix}(\mathbf{f}) = \mathbf{f}) \\ & \wedge (\text{threefix}(\mathbf{t}) = \mathbf{t}) \\ & \wedge (\text{threefix}(x) = x) \\ & \wedge (\text{threefix}(\text{f-buf}(x)) = \text{threefix}(x)) \\ & \wedge (\text{threefix}(\text{f-not}(x)) = \text{f-not}(x)) \\ & \wedge (\text{threefix}(\text{f-and}(x, y)) = \text{f-and}(x, y)) \\ & \wedge (\text{threefix}(\text{f-or}(x, y)) = \text{f-or}(x, y)) \\ & \wedge (\text{threefix}(\text{f-xor}(x, y)) = \text{f-xor}(x, y)) \\ & \wedge (\text{threefix}(\text{f-equiv}(x, y)) = \text{f-equiv}(x, y)) \end{aligned}$$

EVENT: Enable threefix-help-lemma; name this event ‘threefix-help-lemma-off’.

THEOREM: f-not-f-not=f-buf

$$\text{f-not}(\text{f-not}(x)) = \text{f-buf}(x)$$

EVENT: Enable f-not-f-not=f-buf; name this event ‘f-not-f-not=f-buf-off’.

THEOREM: f-buf-delete-lemmas

$$\begin{aligned} & (\text{f-buf}(\text{f-buf}(a)) = \text{f-buf}(a)) \\ & \wedge (\text{f-buf}(\text{f-not}(a)) = \text{f-not}(a)) \\ & \wedge (\text{f-buf}(\text{f-nand}(a, b)) = \text{f-nand}(a, b)) \\ & \wedge (\text{f-buf}(\text{f-nand3}(a, b, c)) = \text{f-nand3}(a, b, c)) \\ & \wedge (\text{f-buf}(\text{f-nand4}(a, b, c, d)) = \text{f-nand4}(a, b, c, d)) \\ & \wedge (\text{f-buf}(\text{f-nand5}(a, b, c, d, e)) = \text{f-nand5}(a, b, c, d, e)) \\ & \wedge (\text{f-buf}(\text{f-nand6}(a, b, c, d, e, g)) = \text{f-nand6}(a, b, c, d, e, g)) \\ & \wedge (\text{f-buf}(\text{f-nand8}(a, b, c, d, e, g, h, i)) = \text{f-nand8}(a, b, c, d, e, g, h, i)) \\ & \wedge (\text{f-buf}(\text{f-or}(a, b)) = \text{f-or}(a, b)) \\ & \wedge (\text{f-buf}(\text{f-or3}(a, b, c)) = \text{f-or3}(a, b, c)) \\ & \wedge (\text{f-buf}(\text{f-or4}(a, b, c, d)) = \text{f-or4}(a, b, c, d)) \\ & \wedge (\text{f-buf}(\text{f-xor}(a, b)) = \text{f-xor}(a, b)) \\ & \wedge (\text{f-buf}(\text{f-xor3}(a, b, c)) = \text{f-xor3}(a, b, c)) \\ & \wedge (\text{f-buf}(\text{f-equiv}(a, b)) = \text{f-equiv}(a, b)) \\ & \wedge (\text{f-buf}(\text{f-equiv3}(a, b, c)) = \text{f-equiv3}(a, b, c)) \\ & \wedge (\text{f-buf}(\text{f-and}(a, b)) = \text{f-and}(a, b)) \\ & \wedge (\text{f-buf}(\text{f-and3}(a, b, c)) = \text{f-and3}(a, b, c)) \\ & \wedge (\text{f-buf}(\text{f-and4}(a, b, c, d)) = \text{f-and4}(a, b, c, d)) \\ & \wedge (\text{f-buf}(\text{f-nor}(a, b)) = \text{f-nor}(a, b)) \end{aligned}$$

$\wedge$ (f-buf (f-nor3 (a, b, c)) = f-nor3 (a, b, c))
 $\wedge$ (f-buf (f-nor4 (a, b, c, d)) = f-nor4 (a, b, c, d))
 $\wedge$ (f-buf (f-nor5 (a, b, c, d, e)) = f-nor5 (a, b, c, d, e))
 $\wedge$ (f-buf (f-nor6 (a, b, c, d, e, g)) = f-nor6 (a, b, c, d, e, g))
 $\wedge$ (f-buf (f-nor8 (a, b, c, d, e, g, h, i)) = f-nor8 (a, b, c, d, e, g, h, i))
 $\wedge$ (f-buf (f-if (c, a, b)) = f-if (c, a, b))

EVENT: Enable f-buf-delete-lemmas; name this event ‘f-buf-delete-lemmas-off’.

THEOREM: f-gate-threefix-congruence-lemmas\$help

(f-buf (threefix (a)) = f-buf (a))
 $\wedge$ (f-not (threefix (a)) = f-not (a))
 $\wedge$ (f-or (threefix (a), b) = f-or (a, b))
 $\wedge$ (f-or (a, threefix (b)) = f-or (a, b))
 $\wedge$ (f-xor (threefix (a), b) = f-xor (a, b))
 $\wedge$ (f-xor (a, threefix (b)) = f-xor (a, b))
 $\wedge$ (f-equiv (threefix (a), b) = f-equiv (a, b))
 $\wedge$ (f-equiv (a, threefix (b)) = f-equiv (a, b))
 $\wedge$ (f-and (threefix (a), b) = f-and (a, b))
 $\wedge$ (f-and (a, threefix (b)) = f-and (a, b))
 $\wedge$ (f-if (threefix (c), a, b) = f-if (c, a, b))
 $\wedge$ (f-if (c, threefix (a), b) = f-if (c, a, b))
 $\wedge$ (f-if (c, a, threefix (b)) = f-if (c, a, b))

EVENT: Enable f-gate-threefix-congruence-lemmas\$help; name this event ‘f-gate-threefix-congruence-lemmas\$help-off’.

THEOREM: f-gate-threefix-congruence-lemmas

(f-buf (threefix (a)) = f-buf (a))
 $\wedge$ (f-not (threefix (a)) = f-not (a))
 $\wedge$ (f-nand (threefix (a), b) = f-nand (a, b))
 $\wedge$ (f-nand (a, threefix (b)) = f-nand (a, b))
 $\wedge$ (f-nand3 (threefix (a), b, c) = f-nand3 (a, b, c))
 $\wedge$ (f-nand3 (a, threefix (b), c) = f-nand3 (a, b, c))
 $\wedge$ (f-nand3 (a, threefix (b), c) = f-nand3 (a, b, c))
 $\wedge$ (f-nand4 (threefix (a), b, c, d) = f-nand4 (a, b, c, d))
 $\wedge$ (f-nand4 (a, threefix (b), c, d) = f-nand4 (a, b, c, d))
 $\wedge$ (f-nand4 (a, b, threefix (c), d) = f-nand4 (a, b, c, d))
 $\wedge$ (f-nand4 (a, b, c, threefix (d)) = f-nand4 (a, b, c, d))
 $\wedge$ (f-nand5 (threefix (a), b, c, d, e) = f-nand5 (a, b, c, d, e))
 $\wedge$ (f-nand5 (a, threefix (b), c, d, e) = f-nand5 (a, b, c, d, e))
 $\wedge$ (f-nand5 (a, b, threefix (c), d, e) = f-nand5 (a, b, c, d, e))
 $\wedge$ (f-nand5 (a, b, c, threefix (d), e) = f-nand5 (a, b, c, d, e))

$\wedge$ (f-nand5($a, b, c, d, \text{threefix}(e)$) = f-nand5(a, b, c, d, e))
 $\wedge$ (f-nand6($\text{threefix}(a), b, c, d, e, g$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand6($a, \text{threefix}(b), c, d, e, g$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand6($a, b, \text{threefix}(c), d, e, g$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand6($a, b, c, \text{threefix}(d), e, g$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand6($a, b, c, d, \text{threefix}(e), g$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand6($a, b, c, d, e, \text{threefix}(g)$) = f-nand6(a, b, c, d, e, g))
 $\wedge$ (f-nand8($\text{threefix}(a), b, c, d, e, g, h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, \text{threefix}(b), c, d, e, g, h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, \text{threefix}(c), d, e, g, h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, c, \text{threefix}(d), e, g, h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, c, d, \text{threefix}(e), g, h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, c, d, e, \text{threefix}(g), h, i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, c, d, e, g, \text{threefix}(h), i$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-nand8($a, b, c, d, e, g, h, \text{threefix}(i)$) = f-nand8(a, b, c, d, e, g, h, i))
 $\wedge$ (f-or($\text{threefix}(a), b$) = f-or(a, b))
 $\wedge$ (f-or($a, \text{threefix}(b)$) = f-or(a, b))
 $\wedge$ (f-or3($\text{threefix}(a), b, c$) = f-or3(a, b, c))
 $\wedge$ (f-or3($a, \text{threefix}(b), c$) = f-or3(a, b, c))
 $\wedge$ (f-or3($a, b, \text{threefix}(c)$) = f-or3(a, b, c))
 $\wedge$ (f-or4($\text{threefix}(a), b, c, d$) = f-or4(a, b, c, d))
 $\wedge$ (f-or4($a, \text{threefix}(b), c, d$) = f-or4(a, b, c, d))
 $\wedge$ (f-or4($a, b, \text{threefix}(c), d$) = f-or4(a, b, c, d))
 $\wedge$ (f-or4($a, b, c, \text{threefix}(d)$) = f-or4(a, b, c, d))
 $\wedge$ (f-xor($\text{threefix}(a), b$) = f-xor(a, b))
 $\wedge$ (f-xor($a, \text{threefix}(b)$) = f-xor(a, b))
 $\wedge$ (f-xor3($\text{threefix}(a), b, c$) = f-xor3(a, b, c))
 $\wedge$ (f-xor3($a, \text{threefix}(b), c$) = f-xor3(a, b, c))
 $\wedge$ (f-xor3($a, b, \text{threefix}(c)$) = f-xor3(a, b, c))
 $\wedge$ (f-equiv($\text{threefix}(a), b$) = f-equiv(a, b))
 $\wedge$ (f-equiv($a, \text{threefix}(b)$) = f-equiv(a, b))
 $\wedge$ (f-equiv3($\text{threefix}(a), b, c$) = f-equiv3(a, b, c))
 $\wedge$ (f-equiv3($a, \text{threefix}(b), c$) = f-equiv3(a, b, c))
 $\wedge$ (f-equiv3($a, b, \text{threefix}(c)$) = f-equiv3(a, b, c))
 $\wedge$ (f-and($\text{threefix}(a), b$) = f-and(a, b))
 $\wedge$ (f-and($a, \text{threefix}(b)$) = f-and(a, b))
 $\wedge$ (f-and3($\text{threefix}(a), b, c$) = f-and3(a, b, c))
 $\wedge$ (f-and3($a, \text{threefix}(b), c$) = f-and3(a, b, c))
 $\wedge$ (f-and3($a, b, \text{threefix}(c)$) = f-and3(a, b, c))
 $\wedge$ (f-and4($\text{threefix}(a), b, c, d$) = f-and4(a, b, c, d))
 $\wedge$ (f-and4($a, \text{threefix}(b), c, d$) = f-and4(a, b, c, d))
 $\wedge$ (f-and4($a, b, \text{threefix}(c), d$) = f-and4(a, b, c, d))
 $\wedge$ (f-and4($a, b, c, \text{threefix}(d)$) = f-and4(a, b, c, d))
 $\wedge$ (f-nor($\text{threefix}(a), b$) = f-nor(a, b))

$$\begin{aligned}
& \wedge \text{ (f-nor } (a, \text{threefix } (b)) = \text{f-nor } (a, b)) \\
& \wedge \text{ (f-nor3 } (\text{threefix } (a), b, c) = \text{f-nor3 } (a, b, c)) \\
& \wedge \text{ (f-nor3 } (a, \text{threefix } (b), c) = \text{f-nor3 } (a, b, c)) \\
& \wedge \text{ (f-nor3 } (a, \text{threefix } (b), c) = \text{f-nor3 } (a, b, c)) \\
& \wedge \text{ (f-nor4 } (\text{threefix } (a), b, c, d) = \text{f-nor4 } (a, b, c, d)) \\
& \wedge \text{ (f-nor4 } (a, \text{threefix } (b), c, d) = \text{f-nor4 } (a, b, c, d)) \\
& \wedge \text{ (f-nor4 } (a, b, \text{threefix } (c), d) = \text{f-nor4 } (a, b, c, d)) \\
& \wedge \text{ (f-nor4 } (a, b, c, \text{threefix } (d)) = \text{f-nor4 } (a, b, c, d)) \\
& \wedge \text{ (f-nor5 } (\text{threefix } (a), b, c, d, e) = \text{f-nor5 } (a, b, c, d, e)) \\
& \wedge \text{ (f-nor5 } (a, \text{threefix } (b), c, d, e) = \text{f-nor5 } (a, b, c, d, e)) \\
& \wedge \text{ (f-nor5 } (a, b, \text{threefix } (c), d, e) = \text{f-nor5 } (a, b, c, d, e)) \\
& \wedge \text{ (f-nor5 } (a, b, c, \text{threefix } (d), e) = \text{f-nor5 } (a, b, c, d, e)) \\
& \wedge \text{ (f-nor5 } (a, b, c, d, \text{threefix } (e)) = \text{f-nor5 } (a, b, c, d, e)) \\
& \wedge \text{ (f-nor6 } (\text{threefix } (a), b, c, d, e, g) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor6 } (a, \text{threefix } (b), c, d, e, g) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor6 } (a, b, \text{threefix } (c), d, e, g) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor6 } (a, b, c, \text{threefix } (d), e, g) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor6 } (a, b, c, d, \text{threefix } (e), g) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor6 } (a, b, c, d, e, \text{threefix } (g)) = \text{f-nor6 } (a, b, c, d, e, g)) \\
& \wedge \text{ (f-nor8 } (\text{threefix } (a), b, c, d, e, g, h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, \text{threefix } (b), c, d, e, g, h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, \text{threefix } (c), d, e, g, h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, c, \text{threefix } (d), e, g, h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, c, d, \text{threefix } (e), g, h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, c, d, e, \text{threefix } (g), h, i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, c, d, e, g, \text{threefix } (h), i) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-nor8 } (a, b, c, d, e, g, h, \text{threefix } (i)) = \text{f-nor8 } (a, b, c, d, e, g, h, i)) \\
& \wedge \text{ (f-if } (\text{threefix } (c), a, b) = \text{f-if } (c, a, b)) \\
& \wedge \text{ (f-if } (c, \text{threefix } (a), b) = \text{f-if } (c, a, b)) \\
& \wedge \text{ (f-if } (c, a, \text{threefix } (b)) = \text{f-if } (c, a, b))
\end{aligned}$$

EVENT: Enable f-gate-threefix-congruence-lemmas; name this event ‘f-gate-threefix-congruence-lemmas-off’.

EVENT: Let us define the theory *f-gates* to consist of the following events: f-buf, f-not, f-nand, f-nand3, f-nand4, f-nand5, f-nand6, f-nand8, f-or, f-or3, f-or4, f-xor, f-xor3, f-equiv, f-equiv3, f-and, f-and3, f-and4, f-nor, f-nor3, f-nor4, f-nor5, f-nor6, f-nor8, f-if, ft-buf, ft-wire, f-pullup.

THEOREM: f-gates=b-gates
 $(\text{boolp } (a) \rightarrow (\text{f-buf } (a) = \text{b-buf } (a)))$
 $\wedge (\text{boolp } (a) \rightarrow (\text{f-not } (a) = \text{b-not } (a)))$
 $\wedge ((\text{boolp } (a) \wedge \text{boolp } (b)) \rightarrow (\text{f-nand } (a, b) = \text{b-nand } (a, b)))$

$$\begin{aligned}
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-nand3}(a, b, c) = \text{b-nand3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d)) \\
& \quad \rightarrow (\text{f-nand4}(a, b, c, d) = \text{b-nand4}(a, b, c, d))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d) \wedge \text{boolp}(e)) \\
& \quad \rightarrow (\text{f-nand5}(a, b, c, d, e) = \text{b-nand5}(a, b, c, d, e))) \\
& \wedge ((\text{boolp}(a) \\
& \quad \wedge \text{boolp}(b) \\
& \quad \wedge \text{boolp}(c) \\
& \quad \wedge \text{boolp}(d) \\
& \quad \wedge \text{boolp}(e) \\
& \quad \wedge \text{boolp}(g)) \\
& \quad \rightarrow (\text{f-nand6}(a, b, c, d, e, g) = \text{b-nand6}(a, b, c, d, e, g))) \\
& \wedge ((\text{boolp}(a) \\
& \quad \wedge \text{boolp}(b) \\
& \quad \wedge \text{boolp}(c) \\
& \quad \wedge \text{boolp}(d) \\
& \quad \wedge \text{boolp}(e) \\
& \quad \wedge \text{boolp}(g) \\
& \quad \wedge \text{boolp}(h) \\
& \quad \wedge \text{boolp}(i)) \\
& \quad \rightarrow (\text{f-nand8}(a, b, c, d, e, g, h, i) = \text{b-nand8}(a, b, c, d, e, g, h, i))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow (\text{f-or}(a, b) = \text{b-or}(a, b))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-or3}(a, b, c) = \text{b-or3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d)) \\
& \quad \rightarrow (\text{f-or4}(a, b, c, d) = \text{b-or4}(a, b, c, d))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow (\text{f-xor}(a, b) = \text{b-xor}(a, b))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-xor3}(a, b, c) = \text{b-xor3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow (\text{f-equiv}(a, b) = \text{b-equiv}(a, b))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-equiv3}(a, b, c) = \text{b-equiv3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow (\text{f-and}(a, b) = \text{b-and}(a, b))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-and3}(a, b, c) = \text{b-and3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d)) \\
& \quad \rightarrow (\text{f-and4}(a, b, c, d) = \text{b-and4}(a, b, c, d))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b)) \rightarrow (\text{f-nor}(a, b) = \text{b-nor}(a, b))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{f-nor3}(a, b, c) = \text{b-nor3}(a, b, c))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d)) \\
& \quad \rightarrow (\text{f-nor4}(a, b, c, d) = \text{b-nor4}(a, b, c, d))) \\
& \wedge ((\text{boolp}(a) \wedge \text{boolp}(b) \wedge \text{boolp}(c) \wedge \text{boolp}(d) \wedge \text{boolp}(e))
\end{aligned}$$

$$\begin{aligned}
& \rightarrow (\text{f-nor5}(a, b, c, d, e) = \text{b-nor5}(a, b, c, d, e))) \\
\wedge & ((\text{boolp}(a) \\
& \wedge \text{boolp}(b) \\
& \wedge \text{boolp}(c) \\
& \wedge \text{boolp}(d) \\
& \wedge \text{boolp}(e) \\
& \wedge \text{boolp}(g)) \\
& \rightarrow (\text{f-nor6}(a, b, c, d, e, g) = \text{b-nor6}(a, b, c, d, e, g))) \\
\wedge & ((\text{boolp}(a) \\
& \wedge \text{boolp}(b) \\
& \wedge \text{boolp}(c) \\
& \wedge \text{boolp}(d) \\
& \wedge \text{boolp}(e) \\
& \wedge \text{boolp}(g) \\
& \wedge \text{boolp}(h) \\
& \wedge \text{boolp}(i)) \\
& \rightarrow (\text{f-nor8}(a, b, c, d, e, g, h, i) = \text{b-nor8}(a, b, c, d, e, g, h, i))) \\
\wedge & ((\text{boolp}(c) \wedge \text{boolp}(a) \wedge \text{boolp}(b)) \\
& \rightarrow (\text{f-if}(c, a, b) = \text{b-if}(c, a, b)))
\end{aligned}$$

DEFINITION:

$\text{fv-or}(a, b)$
 $=$ **if** $a \simeq \text{nil}$ **then nil**
else $\text{cons}(\text{f-or}(\text{car}(a), \text{car}(b)), \text{fv-or}(\text{cdr}(a), \text{cdr}(b)))$ **endif**

EVENT: Enable fv-or; name this event ‘fv-or-off’.

THEOREM: properp-length-fv-or
 $\text{properp}(\text{fv-or}(a, b)) \wedge (\text{length}(\text{fv-or}(a, b)) = \text{length}(a))$

THEOREM: fv-or=v-or
 $(\text{bvp}(a) \wedge \text{bvp}(b) \wedge (\text{length}(a) = \text{length}(b)))$
 $\rightarrow (\text{fv-or}(a, b) = \text{v-or}(a, b))$

DEFINITION:

$\text{fv-xor}(a, b)$
 $=$ **if** $a \simeq \text{nil}$ **then nil**
else $\text{cons}(\text{f-xor}(\text{car}(a), \text{car}(b)), \text{fv-xor}(\text{cdr}(a), \text{cdr}(b)))$ **endif**

EVENT: Enable fv-xor; name this event ‘fv-xor-off’.

THEOREM: properp-length-fv-xor
 $\text{properp}(\text{fv-xor}(a, b)) \wedge (\text{length}(\text{fv-xor}(a, b)) = \text{length}(a))$

THEOREM: fv-xor=v-xor
 $(\text{bvp}(a) \wedge \text{bvp}(b) \wedge (\text{length}(a) = \text{length}(b)))$
 $\rightarrow (\text{fv-xor}(a, b) = \text{v-xor}(a, b))$

DEFINITION:
 $\text{fv-if}(c, a, b)$
 $= \text{if } a \simeq \text{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{f-if}(c, \text{car}(a), \text{car}(b)), \text{fv-if}(c, \text{cdr}(a), \text{cdr}(b))) \text{ endif}$

EVENT: Enable fv-if; name this event 'fv-if-off'.

THEOREM: length-fv-if
 $\text{length}(\text{fv-if}(c, a, b)) = \text{length}(a)$

THEOREM: properp-fv-if
 $\text{properp}(\text{fv-if}(c, a, b))$

THEOREM: v-threefix-fv-if
 $\text{v-threefix}(\text{fv-if}(c, a, b)) = \text{fv-if}(c, a, b)$

THEOREM: fv-if-when-boolp-c
 $(\text{length}(a) = \text{length}(b))$
 $\rightarrow ((\text{fv-if}(\mathbf{t}, a, b) = \text{v-threefix}(a))$
 $\quad \wedge (\text{fv-if}(\mathbf{f}, a, b) = \text{v-threefix}(b)))$

THEOREM: fv-if-when-bvp
 $(\text{boolp}(c) \wedge \text{bvp}(a) \wedge \text{bvp}(b) \wedge (\text{length}(a) = \text{length}(b)))$
 $\rightarrow (\text{fv-if}(c, a, b) = \text{if}^*(c, a, b))$

THEOREM: fv-if-rewrite
 $(\text{length}(a) = \text{length}(b))$
 $\rightarrow (\text{fv-if}(c, a, b)$
 $\quad = \text{if boolp}(c)$
 $\quad \quad \text{then if } c \text{ then v-threefix}(a)$
 $\quad \quad \quad \text{else v-threefix}(b) \text{ endif}$
 $\quad \text{else make-list}(\text{length}(a), \mathbf{x}) \text{ endif})$

EVENT: Enable fv-if-rewrite; name this event 'fv-if-rewrite-off'.

THEOREM: fv-if-v-threefix
 $(\text{fv-if}(c, \text{v-threefix}(a), b) = \text{fv-if}(c, a, b))$
 $\wedge (\text{fv-if}(c, a, \text{v-threefix}(b)) = \text{fv-if}(c, a, b))$

DEFINITION:
 $\text{v-wire}(a, b)$
 $= \text{if } a \simeq \text{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{ft-wire}(\text{car}(a), \text{car}(b)), \text{v-wire}(\text{cdr}(a), \text{cdr}(b))) \text{ endif}$

EVENT: Enable v-wire; name this event ‘v-wire-off’.

THEOREM: v-wire-x-x=x
 $\text{v-wire}(x, x) = \text{v-fourfix}(x)$

THEOREM: v-wire-make-list-z
 $(\text{length}(v) = n)$
 $\rightarrow ((\text{v-wire}(\text{make-list}(n, Z), v) = \text{v-fourfix}(v))$
 $\quad \wedge (\text{v-wire}(v, \text{make-list}(n, Z)) = \text{v-fourfix}(v)))$

THEOREM: properp-length-v-wire
 $\text{properp}(\text{v-wire}(a, b)) \wedge (\text{length}(\text{v-wire}(a, b)) = \text{length}(a))$

THEOREM: v-wire-make-list-x
 $(n = \text{length}(x)) \rightarrow (\text{v-wire}(x, \text{make-list}(n, x)) = \text{make-list}(n, x))$

DEFINITION:
 $\text{v-pullup}(v)$
 $= \text{if } v \simeq \text{nil} \text{ then nil}$
 $\quad \text{else cons}(\text{f-pullup}(\text{car}(v)), \text{v-pullup}(\text{cdr}(v))) \text{ endif}$

EVENT: Enable v-pullup; name this event ‘v-pullup-off’.

THEOREM: properp-length-v-pullup
 $\text{properp}(\text{v-pullup}(v)) \wedge (\text{length}(\text{v-pullup}(v)) = \text{length}(v))$

THEOREM: v-pullup-bvp
 $\text{bvp}(v) \rightarrow (\text{v-pullup}(v) = v)$

THEOREM: v-pullup-make-list-z
 $\text{v-pullup}(\text{make-list}(n, Z)) = \text{make-list}(n, \mathbf{t})$

DEFINITION:
 $\text{fv-shift-right}(a, si)$
 $= \text{if } a \simeq \text{nil} \text{ then nil}$
 $\quad \text{else append}(\text{v-threefix}(\text{cdr}(a)), \text{list}(\text{threefix}(si))) \text{ endif}$

EVENT: Enable fv-shift-right; name this event ‘fv-shift-right-off’.

THEOREM: fv-shift-right=v-shift-right
 $(\text{boolp}(si) \wedge \text{bvp}(a)) \rightarrow (\text{fv-shift-right}(a, si) = \text{v-shift-right}(a, si))$

THEOREM: properp-length-fv-shift-right
 $\text{properp}(\text{fv-shift-right}(a, si))$
 $\wedge (\text{length}(\text{fv-shift-right}(a, si)) = \text{length}(a))$

THEOREM: v-threefix-fv-shift-right
 $\text{v-threefix}(\text{fv-shift-right}(a, si)) = \text{fv-shift-right}(a, si)$

DEFINITION:
 $\text{vft-buf}(c, a)$
 $=$ **if** $a \simeq \text{nil}$ **then nil**
 else $\text{cons}(\text{ft-buf}(c, \text{car}(a)), \text{vft-buf}(c, \text{cdr}(a)))$ **endif**

EVENT: Enable vft-buf; name this event 'vft-buf-off'.

THEOREM: properp-length-vft-buf
 $\text{properp}(\text{vft-buf}(c, a)) \wedge (\text{length}(\text{vft-buf}(c, a)) = \text{length}(a))$

THEOREM: vft-buf-lemmas
 $(\text{vft-buf}(\mathbf{t}, a) = \text{v-threefix}(a))$
 $\wedge (\text{vft-buf}(\mathbf{f}, a) = \text{make-list}(\text{length}(a), z))$

THEOREM: vft-buf-rewrite
 $\text{vft-buf}(c, a)$
 $=$ **if** $c = \mathbf{t}$ **then** $\text{v-threefix}(a)$
 elseif $c = \mathbf{f}$ **then** $\text{make-list}(\text{length}(a), z)$
 else $\text{make-list}(\text{length}(a), x)$ **endif**

EVENT: Enable vft-buf-rewrite; name this event 'vft-buf-rewrite-off'.

```
(DEFN PRIMP-DATABASE NIL
 (LIST
  '(A02 (DELAYS ((263 737) (83 392)))
    (DRIVES 10)
    (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
    (INPUTS A B C D)
    (LOADINGS 1 1 1 1)
    (LSI-NAME . A02)
    (OUT-DEPENDS (A B C D))
    (OUTPUT-TYPES BOOLP)
    (OUTPUTS Z)
    (RESULTS CONS
      (F-NOR (F-AND A B) (F-AND C D))
      'NIL)
    (GATES . 2)
    (PRIMITIVES . 1)
    (TRANSISTORS . 8))
```

```

'(A04 (DELAYS ((260 895) (82 311)))
      (DRIVES 10)
      (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
      (INPUTS A B C D)
      (LOADINGS 1 1 1 1)
      (LSI-NAME . A04)
      (OUT-DEPENDS (A B C D))
      (OUTPUT-TYPES BOOLP)
      (OUTPUTS Z)
      (RESULTS CONS
        (F-NAND (F-OR A B) (F-OR C D))
        'NIL)
      (GATES . 2)
      (PRIMITIVES . 1)
      (TRANSISTORS . 8))
'(A06 (DELAYS ((260 745) (82 202)))
      (DRIVES 10)
      (INPUT-TYPES BOOLP BOOLP BOOLP)
      (INPUTS A B C)
      (LOADINGS 1 1 1)
      (LSI-NAME . A06)
      (OUT-DEPENDS (A B C))
      (OUTPUT-TYPES BOOLP)
      (OUTPUTS Z)
      (RESULTS CONS
        (F-NOR (F-AND A B) C)
        'NIL)
      (GATES . 2)
      (PRIMITIVES . 1)
      (TRANSISTORS . 6))
'(A07 (DELAYS ((261 486) (83 293)))
      (DRIVES 10)
      (INPUT-TYPES BOOLP BOOLP BOOLP)
      (INPUTS A B C)
      (LOADINGS 1 1 1)
      (LSI-NAME . A07)
      (OUT-DEPENDS (A B C))
      (OUTPUT-TYPES BOOLP)
      (OUTPUTS Z)
      (RESULTS CONS
        (F-NAND (F-OR A B) C)
        'NIL)
      (GATES . 2)
      (PRIMITIVES . 1)

```

```

      (TRANSISTORS . 6))
'(B-AND (DELAYS ((144 422) (54 707)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A B)
  (LOADINGS 1 1)
  (LSI-NAME . AN2)
  (OUT-DEPENDS (A B))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-AND A B) 'NIL)
  (GATES . 2)
  (PRIMITIVES . 1)
  (TRANSISTORS . 6))
'(B-AND3 (DELAYS ((146 663) (58 809)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP)
  (INPUTS A B C)
  (LOADINGS 1 1 1)
  (LSI-NAME . AN3)
  (OUT-DEPENDS (A B C))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-AND3 A B C) 'NIL)
  (GATES . 2)
  (PRIMITIVES . 1)
  (TRANSISTORS . 8))
'(B-AND4 (DELAYS ((149 934) (60 870)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
  (INPUTS A B C D)
  (LOADINGS 1 1 1 1)
  (LSI-NAME . AN4)
  (OUT-DEPENDS (A B C D))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-AND4 A B C D) 'NIL)
  (GATES . 3)
  (PRIMITIVES . 1)
  (TRANSISTORS . 10))
'(B-EQUV (DELAYS ((145 742) (67 973)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A B)

```

```

        (LOADINGS 1 2)
        (LSI-NAME . EN)
        (OUT-DEPENDS (A B))
        (OUTPUT-TYPES BOOLP)
        (OUTPUTS Z)
        (RESULTS CONS (F-EQUV A B) 'NIL)
        (GATES . 3)
        (PRIMITIVES . 1)
        (TRANSISTORS . 12))
'(B-EQUV3 (DELAYS ((151 1806) (79 1580)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP)
  (INPUTS A B C)
  (LOADINGS 1 3 2)
  (LSI-NAME . EN3)
  (OUT-DEPENDS (A B C))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-EQUV3 A B C) 'NIL)
  (GATES . 7)
  (PRIMITIVES . 1)
  (TRANSISTORS . 22))
'(B-IF (DELAYS ((70 775) (60 1040)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP)
  (INPUTS S A B)
  (LOADINGS 2 1 1)
  (LSI-NAME MUX21H B A S)
  (OUT-DEPENDS (A B S))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-IF S A B) 'NIL)
  (GATES . 4)
  (PRIMITIVES . 1)
  (TRANSISTORS . 12))
'(B-NAND (DELAYS ((141 420) (82 161)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A B)
  (LOADINGS 1 1)
  (LSI-NAME . ND2)
  (OUT-DEPENDS (A B))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)

```



```

      (RESULTS CONS (F-NAND A B) 'NIL)
      (GATES . 1)
      (PRIMITIVES . 1)
      (TRANSISTORS . 4))
'(B-NAND3 (DELAYS ((142 537) (109 322)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP)
  (INPUTS A B C)
  (LOADINGS 1 1 1)
  (LSI-NAME . ND3)
  (OUT-DEPENDS (A B C))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-NAND3 A B C) 'NIL)
  (GATES . 2)
  (PRIMITIVES . 1)
  (TRANSISTORS . 6))
'(B-NAND4 (DELAYS ((144 588) (144 418)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
  (INPUTS A B C D)
  (LOADINGS 1 1 1 1)
  (LSI-NAME . ND4)
  (OUT-DEPENDS (A B C D))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-NAND4 A B C D) 'NIL)
  (GATES . 2)
  (PRIMITIVES . 1)
  (TRANSISTORS . 8))
'(B-NAND5 (DELAYS ((144 1002) (59 1120)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP)
  (INPUTS A B C D E)
  (LOADINGS 1 1 1 1 1)
  (LSI-NAME . ND5)
  (OUT-DEPENDS (A B C D E))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS
    (F-NAND5 A B C D E)
    'NIL)
  (GATES . 4)
  (PRIMITIVES . 1)

```

```

        (TRANSISTORS . 16))
'(B-NAND6 (DELAYS ((144 982) (59 1090)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP)
  (INPUTS A B C D E G)
  (LOADINGS 1 1 1 1 1 1)
  (LSI-NAME . ND6)
  (OUT-DEPENDS (A B C D E G))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS
    (F-NAND6 A B C D E G)
    'NIL)
  (GATES . 5)
  (PRIMITIVES . 1)
  (TRANSISTORS . 18))
'(B-NAND8 (DELAYS ((144 1042) (60 1360)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP)
  (INPUTS A B C D E G H I)
  (LOADINGS 1 1 1 1 1 1 1 1)
  (LSI-NAME . ND8)
  (OUT-DEPENDS (A B C D E G H I))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS
    (F-NAND8 A B C D E G H I)
    'NIL)
  (GATES . 6)
  (PRIMITIVES . 1)
  (TRANSISTORS . 22))
'(B-NBUF (DELAYS ((142 447) (57 213))
  ((143 302) (52 366)))
  (DRIVES 9 10)
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 1)
  (LSI-NAME . IVDA)
  (OUT-DEPENDS (A) (A))
  (OUTPUT-TYPES BOOLP BOOLP)
  (OUTPUTS Y Z)
  (RESULTS CONS
    (F-NOT A)
    (CONS (F-BUF A) 'NIL)))

```

```

(GATES . 1)
(PRIMITIVES . 1)
(TRANSISTORS . 4))
'(B-NOR (DELAYS ((260 460) (59 170)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP)
(INPUTS A B)
(LOADINGS 1 1)
(LSI-NAME . NR2)
(OUT-DEPENDS (A B))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-NOR A B) 'NIL)
(GATES . 1)
(PRIMITIVES . 1)
(TRANSISTORS . 4))
'(B-NOR3 (DELAYS ((384 798) (59 224)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP)
(INPUTS A B C)
(LOADINGS 1 1 1)
(LSI-NAME . NR3)
(OUT-DEPENDS (A B C))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-NOR3 A B C) 'NIL)
(GATES . 2)
(PRIMITIVES . 1)
(TRANSISTORS . 6))
'(B-NOR4 (DELAYS ((510 1072) (59 225)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
(INPUTS A B C D)
(LOADINGS 1 1 1 1)
(LSI-NAME . NR4)
(OUT-DEPENDS (A B C D))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-NOR4 A B C D) 'NIL)
(GATES . 2)
(PRIMITIVES . 1)
(TRANSISTORS . 8))
'(B-NOR5 (DELAYS ((145 1493) (55 767)))
(DRIVES 10)

```

```

(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP)
(INPUTS A B C D E)
(LOADINGS 1 1 1 1 1)
(LSI-NAME . NR5)
(OUT-DEPENDS (A B C D E))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS
  (F-NOR5 A B C D E)
  'NIL)
(GATES . 4)
(PRIMITIVES . 1)
(TRANSISTORS . 16))
'(B-NOR6 (DELAYS ((146 1593) (55 807))))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP)
(INPUTS A B C D E G)
(LOADINGS 1 1 1 1 1 1)
(LSI-NAME . NR6)
(OUT-DEPENDS (A B C D E G))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS
  (F-NOR6 A B C D E G)
  'NIL)
(GATES . 5)
(PRIMITIVES . 1)
(TRANSISTORS . 18))
'(B-NOR8 (DELAYS ((146 1853) (54 767))))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP)
(INPUTS A B C D E G H I)
(LOADINGS 1 1 1 1 1 1 1 1)
(LSI-NAME . NR8)
(OUT-DEPENDS (A B C D E G H I))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS
  (F-NOR8 A B C D E G H I)
  'NIL)
(GATES . 6)
(PRIMITIVES . 1)
(TRANSISTORS . 22))
'(B-NOT (DELAYS ((70 235) (57 198))))

```

```

(DRIVES 10)
(INPUT-TYPES BOOLP)
(INPUTS A)
(LOADINGS 2)
(LSI-NAME . IVA)
(OUT-DEPENDS (A))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-NOT A) 'NIL)
(GATES . 1)
(PRIMITIVES . 1)
(TRANSISTORS . 3))
'(B-NOT-B4IP (DELAYS ((17 333) (12 124)))
  (DRIVES 128)
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 8)
  (LSI-NAME . B4IP)
  (OUT-DEPENDS (A))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-NOT A) 'NIL)
  (GATES . 4)
  (PRIMITIVES . 1)
  (TRANSISTORS . 12))
'(B-NOT-IVAP (DELAYS ((38 208) (35 126)))
  (DRIVES 20)
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 3)
  (LSI-NAME . IVAP)
  (OUT-DEPENDS (A))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-NOT A) 'NIL)
  (GATES . 2)
  (PRIMITIVES . 1)
  (TRANSISTORS . 6))
'(B-OR (DELAYS ((143 332) (58 819)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A B)
  (LOADINGS 1 1)
  (LSI-NAME . OR2)

```

```

(OUT-DEPENDS (A B))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-OR A B) 'NIL)
(GATES . 2)
(PRIMITIVES . 1)
(TRANSISTORS . 6))
'(B-OR3 (DELAYS ((144 422) (70 1185)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP)
(INPUTS A B C)
(LOADINGS 1 1 1)
(LSI-NAME . OR3)
(OUT-DEPENDS (A B C))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-OR3 A B C) 'NIL)
(GATES . 2)
(PRIMITIVES . 1)
(TRANSISTORS . 8))
'(B-OR4 (DELAYS ((143 352) (78 1329)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP)
(INPUTS A B C D)
(LOADINGS 1 1 1 1)
(LSI-NAME . OR4)
(OUT-DEPENDS (A B C D))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-OR4 A B C D) 'NIL)
(GATES . 3)
(PRIMITIVES . 1)
(TRANSISTORS . 10))
'(B-XOR (DELAYS ((145 742) (67 973)))
(DRIVES 10)
(INPUT-TYPES BOOLP BOOLP)
(INPUTS A B)
(LOADINGS 1 2)
(LSI-NAME . EO)
(OUT-DEPENDS (A B))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-XOR A B) 'NIL)
(GATES . 3)

```

```

        (PRIMITIVES . 1)
        (TRANSISTORS . 12))
'(B-XOR3 (DELAYS ((151 1806) (79 1580)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP BOOLP)
  (INPUTS A B C)
  (LOADINGS 1 3 2)
  (LSI-NAME . EO3)
  (OUT-DEPENDS (A B C))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-XOR3 A B C) 'NIL)
  (GATES . 7)
  (PRIMITIVES . 1)
  (TRANSISTORS . 22))
'(DEL2 (DELAYS ((70 2035) (38 2199)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 1)
  (LSI-NAME . DEL2)
  (OUT-DEPENDS (A))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-BUF A) 'NIL)
  (GATES . 4)
  (PRIMITIVES . 1)
  (TRANSISTORS . 16))
'(DEL4 (DELAYS ((73 4017) (38 4179)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 1)
  (LSI-NAME . DEL4)
  (OUT-DEPENDS (A))
  (OUTPUT-TYPES BOOLP)
  (OUTPUTS Z)
  (RESULTS CONS (F-BUF A) 'NIL)
  (GATES . 7)
  (PRIMITIVES . 1)
  (TRANSISTORS . 28))
'(DEL10 (DELAYS ((60 10530) (49 10424)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP)

```



```

(CONS WRITE-B1
(CONS WRITE-B2
(CONS WRITE-B3
(CONS WEN
(CONS D0
(CONS D1
(CONS D2
(CONS D3
(CONS D4
(CONS D5
(CONS D6
(CONS D7
(CONS D8
(CONS D9
(CONS D10
(CONS D11
(CONS D12
(CONS D13
(CONS D14
(CONS D15
(CONS D16
(CONS D17
(CONS D18
(CONS D19
(CONS D20
(CONS D21
(CONS D22
(CONS D23
(CONS
D24
(CONS
D25
(CONS
D26
(CONS
D27
(CONS
D28
(CONS
D29
(CONS
D30
(CONS D31
'NIL))))))))))))))))))))))))))))))))))

```



```

(CONS READ-A3
(CONS WRITE-B0
(CONS WRITE-B1
(CONS WRITE-B2
(CONS WRITE-B3
(CONS WEN
(CONS D0
(CONS D1
(CONS D2
(CONS D3
(CONS D4
(CONS D5
(CONS D6
(CONS D7
(CONS D8
(CONS D9
(CONS D10
(CONS D11
(CONS D12
(CONS D13
(CONS D14
(CONS D15
(CONS D16
(CONS D17
(CONS D18
(CONS D19
(CONS D20
(CONS D21
(CONS D22
(CONS D23
(CONS
D24
(CONS
D25
(CONS
D26
(CONS
D27
(CONS
D28
(CONS
D29
(CONS
D30

```

```

(CONS D31
'NIL))))))))))))))))))))))))))))))))))))))

STATE)
(STATE-TYPES ADDRESSED-STATE 4
  (RAM '(BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
        BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
        BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
        BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP))

(STATES . STATE)
(GATES . 2368)
(PRIMITIVES . 1)
(TRANSISTORS . 9472))
'(FD1 (DELAYS ((147 1024) (55 1288))
              ((145 1432) (53 1447)))
  (DRIVES 10 10)
  (INPUT-TYPES BOOLP CLK)
  (INPUTS D CP)
  (LOADINGS 1 1)
  (LSI-NAME . FD1)
  (NEW-STATES F-BUF D)
  (OUT-DEPENDS NIL NIL)
  (OUTPUT-TYPES BOOLP BOOLP)
  (OUTPUTS Q QN)
  (RESULTS CONS
    (F-BUF STATE)
    (CONS (F-NOT STATE) 'NIL))
  (STATE-TYPES . BOOLP)
  (STATES . STATE)
  (GATES . 7)
  (PRIMITIVES . 1)
  (TRANSISTORS . 26))
'(FD1S (DELAYS ((147 1024) (55 1288))
               ((145 1432) (53 1447)))
  (DRIVES 10 10)
  (INPUT-TYPES BOOLP CLK BOOLP BOOLP)
  (INPUTS D CP TI TE)
  (LOADINGS 1 1 1 2)
  (LSI-NAME . FD1S)
  (NEW-STATES F-IF TE TI D)
  (OUT-DEPENDS NIL NIL)
  (OUTPUT-TYPES BOOLP BOOLP)
  (OUTPUTS Q QN)
  (RESULTS CONS
    (F-BUF STATE)

```

```

(CONS (F-NOT STATE) 'NIL))
(STATE-TYPES . BOOLP)
(STATES . STATE)
(GATES . 9)
(PRIMITIVES . 1)
(TRANSISTORS . 34))
'(FD1SP (DELAYS ((68 1084) (34 1327))
                ((65 1712) (32 1596))))
(DRIVES 16 16)
(INPUT-TYPES BOOLP CLK BOOLP BOOLP)
(INPUTS D CP TI TE)
(LOADINGS 1 1 1 2)
(LSI-NAME . FD1SP)
(NEW-STATES F-IF TE TI D)
(OUT-DEPENDS NIL NIL)
(OUTPUT-TYPES BOOLP BOOLP)
(OUTPUTS Q QN)
(RESULTS CONS
  (F-BUF STATE)
  (CONS (F-NOT STATE) 'NIL))
(STATE-TYPES . BOOLP)
(STATES . STATE)
(GATES . 10)
(PRIMITIVES . 1)
(TRANSISTORS . 38))
'(FD1SLP (DELAYS ((70 1085) (45 982))
                 ((67 1493) (35 1568))))
(DRIVES 12 12)
(INPUT-TYPES BOOLP CLK BOOLP BOOLP BOOLP)
(INPUTS D CP LD TI TE)
(LOADINGS 1 1 2 1 2)
(LSI-NAME . FD1SLP)
(NEW-STATES F-IF TE TI
  (F-IF LD D STATE))
(OUT-DEPENDS NIL NIL)
(OUTPUT-TYPES BOOLP BOOLP)
(OUTPUTS Q QN)
(RESULTS CONS
  (F-BUF STATE)
  (CONS (F-NOT STATE) 'NIL))
(STATE-TYPES . BOOLP)
(STATES . STATE)
(GATES . 12)
(PRIMITIVES . 1)

```

[illegible]

```

((1410 20000) (820 20000))
((1410 20000) (820 20000))
((1410 20000) (820 20000))
((1410 20000) (820 20000))
(DRIVES 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
BOOLP BOOLP BOOLP)
(INPUTS RW- STROBE- A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15
A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28 A29 A30 A31 D0
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19
D20 D21 D22 D23 D24 D25 D26 D27 D28 D29 D30 D31)
(LOADINGS 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(LSI-NAME . MEM-32X32)
(NEW-STATES MEM-STATE
(CONS RW-
(CONS STROBE-
(CONS A0
(CONS A1
(CONS A2
(CONS A3
(CONS A4
(CONS A5
(CONS A6
(CONS A7
(CONS A8
(CONS A9
(CONS A10
(CONS A11
(CONS A12
(CONS A13
(CONS A14
(CONS A15
(CONS A16
(CONS A17
(CONS A18
(CONS A19
(CONS A20

```



```

(CONS A21
  (CONS A22
    (CONS A23
      (CONS A24
        (CONS A25
          (CONS A26
            (CONS A27
              (CONS A28
                (CONS A29
                  (CONS A30
                    (CONS
                      A31
                      (CONS
                        D0
                        (CONS
                          D1
                          (CONS
                            D2
                            (CONS
                              D3
                              (CONS
                                D4
                                (CONS
                                  D5
                                  (CONS
                                    D6
                                    (CONS
                                      D7
                                      (CONS
                                        D8
                                        (CONS
                                          D9
                                          (CONS
                                            D10
                                            (CONS
                                              D11
                                              (CONS
                                                D12
                                                (CONS
                                                  D13
                                                  (CONS
                                                    D14
                                                    (CONS
                                                      D15

```

```

(CONS
D16
(CONS
D17
(CONS
D18
(CONS
D19
(CONS
D20
(CONS
D21
(CONS
D22
(CONS
D23
(CONS
D24
(CONS
D25
(CONS
D26
(CONS
D27
(CONS
D28
(CONS
D29
(CONS
D30
(CONS D31

```

```

STATE)
(OUT-DEPENDS (RW- STROBE- A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31)
(RW- STROBE- A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31)
(RW- STROBE- A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28
A29 A30 A31)
(RW- STROBE- A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 A25 A26 A27 A28

```

[illegible]


```

(OUTPUT-TYPES BOOLP TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE
    TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE
    TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE
    TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE TRI-STATE
    TRI-STATE TRI-STATE TRI-STATE)
(OUTPUTS DTACK- 00 01 02 03 04 05 06 07 08 09 010 011 012 013 014 015 016
    017 018 019 020 021 022 023 024 025 026 027 028 029 030 031)
(RESULTS MEM-VALUE
    (CONS RW-
        (CONS STROBE-
            (CONS A0
                (CONS A1
                    (CONS A2
                        (CONS A3
                            (CONS A4
                                (CONS A5
                                    (CONS A6
                                        (CONS A7
                                            (CONS A8
                                                (CONS A9
                                                    (CONS A10
                                                        (CONS A11
                                                            (CONS A12
                                                                (CONS A13
                                                                    (CONS A14
                                                                        (CONS A15
                                                                            (CONS A16
                                                                                (CONS A17
                                                                                    (CONS A18
                                                                                        (CONS A19
                                                                                            (CONS A20
                                                                                                (CONS A21
                                                                                                    (CONS A22
                                                                                                        (CONS A23
                                                                                                            (CONS A24
                                                                                                                (CONS A25
                                                                                                                    (CONS A26
                                                                                                                        (CONS A27
                                                                                                                            (CONS A28
                                                                                                                                (CONS A29
                                                                                                                                    (CONS A30
                                                                                                                                        (CONS
                                                                                                                                            A31

```

```
(CONS
D0
(CONS
D1
(CONS
D2
(CONS
D3
(CONS
D4
(CONS
D5
(CONS
D6
(CONS
D7
(CONS
D8
(CONS
D9
(CONS
D10
(CONS
D11
(CONS
D12
(CONS
D13
(CONS
D14
(CONS
D15
(CONS
D16
(CONS
D17
(CONS
D18
(CONS
D19
(CONS
D20
(CONS
D21
```

```

(CONS
D22
(CONS
D23
(CONS
D24
(CONS
D25
(CONS
D26
(CONS
D27
(CONS
D28
(CONS
D29
(CONS
D30
(CONS D31

STATE)
(STATE-TYPES (ADDRESSED-STATE 32
(RAM '(BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
      BOOLP BOOLP BOOLP BOOLP BOOLP
      BOOLP BOOLP BOOLP BOOLP BOOLP
      BOOLP BOOLP BOOLP BOOLP BOOLP
      BOOLP BOOLP BOOLP BOOLP BOOLP
      BOOLP))
NUMBERP NUMBER-LISTP NUMBERP BOOLP BOOLP
(BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP)
(BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP
  BOOLP)
(STATES . STATE)
(GATES . 0)
(PRIMITIVES . 1)
(TRANSISTORS . 0))
'(RAM-ENABLE-CIRCUIT (DELAYS ((68 12000) (54 12000)))
  (DRIVES 10)

```

```

        (INPUT-TYPES CLK BOOLP BOOLP BOOLP)
        (INPUTS CLK TEST-REGFILE- DISABLE-REGFILE- WE)
        (LOADINGS 1 2 2 2)
        (LSI-NAME . RAM-ENABLE-CIRCUIT)
        (OUT-DEPENDS (CLK TEST-REGFILE- DISABLE-REGFILE- WE))
        (OUTPUT-TYPES LEVEL)
        (OUTPUTS Z)
        (RESULTS CONS
          (F-NAND DISABLE-REGFILE- WE)
          'NIL)
        (GATES . 24)
        (PRIMITIVES . 1)
        (TRANSISTORS . 97))
'(T-BUF (DELAYS ((146 313) (57 728)))
  (DRIVES 10)
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS E A)
  (LOADINGS 2 2)
  (LSI-NAME BTS4 A E)
  (OUT-DEPENDS (A E))
  (OUTPUT-TYPES TRI-STATE)
  (OUTPUTS Z)
  (RESULTS CONS (FT-BUF E A) 'NIL)
  (GATES . 3)
  (PRIMITIVES . 1)
  (TRANSISTORS . 12))
'(T-WIRE (DELAYS (OR A B))
  (DRIVES (MIN A B))
  (INPUT-TYPES TRI-STATE TRI-STATE)
  (INPUTS A B)
  (LOADINGS 1 1)
  (LSI-NAME . T-WIRE)
  (OUT-DEPENDS (A B))
  (OUTPUT-TYPES TRI-STATE)
  (OUTPUTS Z)
  (RESULTS CONS (FT-WIRE A B) 'NIL)
  (GATES . 0)
  (PRIMITIVES . 0)
  (TRANSISTORS . 0))
'(PULLUP (DELAYS A)
  (DRIVES A)
  (INPUT-TYPES TRI-STATE)
  (INPUTS A)
  (LOADINGS 1)

```



```

(LSI-NAME . PULLUP)
(OUT-DEPENDS (A))
(OUTPUT-TYPES BOOLP)
(OUTPUTS Z)
(RESULTS CONS (F-PULLUP A) 'NIL)
(GATES . 0)
(PRIMITIVES . 0)
(TRANSISTORS . 0))
'(TTL-BIDIRECT (DELAYS (((61 PS-PF) 1633) ((41 PS-PF) 2253))
  ((43 633) (20 638))
  ((141 1053) (82 799))))
  (DRIVES (8 MA) 10 10)
  (INPUT-TYPES TTL-TRI-STATE BOOLP BOOLP PARAMETRIC)
  (INPUTS IO A EN PI)
  (LOADINGS (3 PF) 1 1 1)
  (LSI-NAME . BD8TRP)
  (OUT-DEPENDS (EN A)
    (EN A IO)
    (EN A IO PI))
  (OUTPUT-TYPES TTL-TRI-STATE BOOLP PARAMETRIC)
  (OUTPUTS IO ZI PO)
  (RESULTS CONS
    (FT-BUF (F-NOT EN) A)
    (CONS (F-BUF (FT-WIRE IO (FT-BUF (F-NOT EN) A)))
      (CONS (F-NAND (FT-WIRE IO (FT-BUF (F-NOT EN) A))
        PI)
        'NIL)))
    (GATES . 0)
    (PADS . 1)
    (PRIMITIVES . 1)
    (TRANSISTORS . 0))
'(TTL-CLK-INPUT (DELAYS ((4 1225) (4 1152))
  ((202 1050) (117 741))))
  (DRIVES 400 10)
  (INPUT-TYPES TTL PARAMETRIC)
  (INPUTS A PI)
  (LOADINGS (3 PF) 1)
  (LSI-NAME . DRVT8)
  (OUT-DEPENDS (A) (A PI))
  (OUTPUT-TYPES CLK PARAMETRIC)
  (OUTPUTS Z PO)
  (RESULTS CONS
    (F-BUF A)
    (CONS (F-NAND A PI) 'NIL))

```

```

        (GATES . 0)
        (PADS . 2)
        (PRIMITIVES . 1)
        (TRANSISTORS . 0))
'(TTL-INPUT (DELAYS ((43 633) (20 638))
                  ((141 1053) (82 799))))
(DRIVES 10 10)
(INPUT-TYPES TTL PARAMETRIC)
(INPUTS A PI)
(LOADINGS (3 PF) 1)
(LSI-NAME . TLCHT)
(OUT-DEPENDS (A) (A PI))
(OUTPUT-TYPES BOOLP PARAMETRIC)
(OUTPUTS Z PO)
(RESULTS CONS
  (F-BUF A)
  (CONS (F-NAND A PI) 'NIL))
(GATES . 0)
(PADS . 1)
(PRIMITIVES . 1)
(TRANSISTORS . 0))
'(TTL-OUTPUT (DELAYS (((61 PS-PF) 812) ((42 PS-PF) 1155)))
              (DRIVES (8 MA))
              (INPUT-TYPES BOOLP)
              (INPUTS A)
              (LOADINGS 5)
              (LSI-NAME . B8RP)
              (OUT-DEPENDS (A))
              (OUTPUT-TYPES TTL)
              (OUTPUTS Z)
              (RESULTS CONS (F-BUF A) 'NIL)
              (GATES . 0)
              (PADS . 1)
              (PRIMITIVES . 1)
              (TRANSISTORS . 0))
'(TTL-OUTPUT-PARAMETRIC (DELAYS (((64 PS-PF) 737) ((42 PS-PF) 1125)))
                        (DRIVES (4 MA))
                        (INPUT-TYPES PARAMETRIC)
                        (INPUTS A)
                        (LOADINGS 3)
                        (LSI-NAME . B4)
                        (OUT-DEPENDS (A))
                        (OUTPUT-TYPES TTL)
                        (OUTPUTS Z)

```

```

        (RESULTS CONS (F-BUF A) 'NIL)
        (GATES . 0)
        (PADS . 1)
        (PRIMITIVES . 1)
        (TRANSISTORS . 0))
'(TTL-OUTPUT-FAST (DELAYS (((36 PS-PF) 991) ((24 PS-PF) 1488)))
  (DRIVES (8 MA))
  (INPUT-TYPES BOOLP)
  (INPUTS A)
  (LOADINGS 3)
  (LSI-NAME . B8)
  (OUT-DEPENDS (A))
  (OUTPUT-TYPES TTL)
  (OUTPUTS Z)
  (RESULTS CONS (F-BUF A) 'NIL)
  (GATES . 0)
  (PADS . 1)
  (PRIMITIVES . 1)
  (TRANSISTORS . 0))
'(TTL-TRI-OUTPUT (DELAYS (((61 PS-PF) 1602) ((41 PS-PF) 2233)))
  (DRIVES (8 MA))
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A EN)
  (LOADINGS 1 1)
  (LSI-NAME . BT8RP)
  (OUT-DEPENDS (A EN))
  (OUTPUT-TYPES TTL-TRI-STATE)
  (OUTPUTS Z)
  (RESULTS CONS
    (FT-BUF (F-NOT EN) A)
    'NIL)
  (GATES . 0)
  (PADS . 1)
  (PRIMITIVES . 1)
  (TRANSISTORS . 0))
'(TTL-TRI-OUTPUT-FAST (DELAYS (((36 PS-PF) 1581) ((24 PS-PF) 2198)))
  (DRIVES (8 MA))
  (INPUT-TYPES BOOLP BOOLP)
  (INPUTS A EN)
  (LOADINGS 1 1)
  (LSI-NAME . BT8)
  (OUT-DEPENDS (A EN))
  (OUTPUT-TYPES TTL-TRI-STATE)
  (OUTPUTS Z)

```

```

                                (RESULTS CONS
                                (FT-BUF (F-NOT EN) A)
                                'NIL)
                                (GATES . 0)
                                (PADS . 1)
                                (PRIMITIVES . 1)
                                (TRANSISTORS . 0))
(LIST 'VDD
      '(DELAYS ((0 0) (0 0)))
      '(DRIVES 50)
      '(INPUT-TYPES)
      '(INPUTS)
      '(LOADINGS)
      '(LSI-NAME . VDD)
      '(OUT-DEPENDS NIL)
      '(OUTPUT-TYPES BOOLP)
      '(OUTPUTS Z)
      (LIST 'RESULTS
            'CONS
            (LIST 'QUOTE T)
            ''NIL)
      '(GATES . 0)
      '(PRIMITIVES . 1)
      '(TRANSISTORS . 0))
(LIST 'VDD-PARAMETRIC
      '(DELAYS ((0 0) (0 0)))
      '(DRIVES 50)
      '(INPUT-TYPES)
      '(INPUTS)
      '(LOADINGS)
      '(LSI-NAME . VDD)
      '(OUT-DEPENDS NIL)
      '(OUTPUT-TYPES PARAMETRIC)
      '(OUTPUTS Z)
      (LIST 'RESULTS
            'CONS
            (LIST 'QUOTE T)
            ''NIL)
      '(GATES . 0)
      '(PRIMITIVES . 1)
      '(TRANSISTORS . 0))
(LIST 'VSS
      '(DELAYS ((0 0) (0 0)))
      '(DRIVES 50)

```

```

' (INPUT-TYPES)
' (INPUTS)
' (LOADINGS)
' (LSI-NAME . VSS)
' (OUT-DEPENDS NIL)
' (OUTPUT-TYPES BOOLP)
' (OUTPUTS Z)
(LIST 'RESULTS
      'CONS
      (LIST 'QUOTE F)
      'NIL)
' (GATES . 0)
' (PRIMITIVES . 1)
' (TRANSISTORS . 0)))

```

DEFINITION:

```

lookup-module(name, netlist)
=  if netlist  $\simeq$  nil then f
    elseif listp(car(netlist))  $\wedge$  (caar(netlist) = name)
    then car(netlist)
    else lookup-module(name, cdr(netlist)) endif

```

EVENT: Enable lookup-module; name this event 'lookup-module-off'.

DEFINITION:

```

delete-module(name, netlist)
=  if netlist  $\simeq$  nil then netlist
    elseif listp(car(netlist))  $\wedge$  (caar(netlist) = name)
    then cdr(netlist)
    else cons(car(netlist), delete-module(name, cdr(netlist))) endif

```

EVENT: Enable delete-module; name this event 'delete-module-off'.

THEOREM: delete-module-lookup-module-count

```

lookup-module(name, netlist)
 $\rightarrow$  (count(delete-module(name, netlist)) < count(netlist))

```

DEFINITION:

```

pairstates(statenames, state)
=  if listp(statenames)  $\vee$  (statenames = nil)
    then pairlist(statenames, state)
    else list(cons(statenames, state)) endif

```

DEFINITION:

```

collect-states(statenames, state-bindings)
=  if listp(statenames)  $\vee$  (statenames = nil)
    then collect-value(statenames, state-bindings)
    else value(statenames, state-bindings) endif

```

DEFINITION: module-name(*module*) = car(*module*)

DEFINITION: module-inputs(*module*) = cadr(*module*)

DEFINITION: module-outputs(*module*) = caddr(*module*)

DEFINITION: module-occurrences(*module*) = caddr(*module*)

DEFINITION: module-statenames(*module*) = caddr(*module*)

DEFINITION: module-annotation(*module*) = caddr(*module*)

DEFINITION: occ-name(*occurrence*) = car(*occurrence*)

DEFINITION: occ-outputs(*occurrence*) = cadr(*occurrence*)

DEFINITION: occ-function(*occurrence*) = caddr(*occurrence*)

DEFINITION: occ-inputs(*occurrence*) = caddr(*occurrence*)

DEFINITION: occ-annotation(*occurrence*) = caddr(*occurrence*)

DEFINITION: primp(*fn*) = lookup-module(*fn*, PRIMP-DATABASE)

EVENT: Enable primp; name this event 'primp-off'.

THEOREM: not-litatom ==> not-primp
 $(\neg \text{litatom}(fn)) \rightarrow (\neg \text{primp}(fn))$

DEFINITION:
primp-lookup(*fn*, *name*) = lookup-module(*name*, cdr(primp(*fn*)))

DEFINITION: primp2(*fn*, *name*) = cdr(primp-lookup(*fn*, *name*))

DEFINITION:
dual-apply-value(*fn*, *args*, *state*)
= eval\$(t,
primp2(*fn*, 'results'),
append(pairlist(primp2(*fn*, 'inputs'), *args*),
pairstates(primp2(*fn*, 'states'), *state*)))

EVENT: Enable dual-apply-value; name this event 'dual-apply-value-off'.

DEFINITION:

```
dual-apply-state(fn, args, state)
= eval$(t,
    primp2(fn, 'new-states'),
    append(pairlist(primp2(fn, 'inputs'), args),
        pairstates(primp2(fn, 'states'), state)))
```

EVENT: Enable dual-apply-state; name this event 'dual-apply-state-off'.

DEFINITION:

```
dual-eval(flag, x0, x1, x2, netlist)
= case on flag:
    case = 0
    then if primp(x0) then dual-apply-value(x0, x1, x2)
        elseif listp(lookup-module(x0, netlist))
            then collect-value(module-outputs(lookup-module(x0, netlist)),
                dual-eval(1,
                    module-occurrences(lookup-module(x0,
                        netlist)),
                    pairlist(module-inputs(lookup-module(x0,
                        netlist)),
                        x1),
                    pairstates(module-statenames(lookup-module(x0,
                        netlist)),
                        x2),
                    delete-module(x0, netlist)))
        else fendif
    case = 1
    then if listp(x0)
        then dual-eval(1,
            cdr(x0),
            append(pairlist(occ-outputs(car(x0)),
                dual-eval(0,
                    occ-function(car(x0)),
                    collect-value(occ-inputs(car(x0)),
                        x1),
                    value(occ-name(car(x0)),
                        x2),
                        netlist)),
                    x1),
                x2,
                netlist))
            else x1 endif
    case = 2
```

```

then if primp( $x0$ ) then dual-apply-state( $x0, x1, x2$ )
  elseif listp(lookup-module( $x0, netlist$ ))
    then collect-states(module-statenames(lookup-module( $x0, netlist$ )),
      dual-eval(3,
        module-occurrences(lookup-module( $x0, netlist$ )),
        dual-eval(1,
          module-occurrences(lookup-module( $x0, netlist$ )),
          pairlist(module-inputs(lookup-module( $x0, netlist$ )),
             $x1$ ),
          pairstates(module-statenames(lookup-module( $x0, netlist$ )),
             $x2$ ),
          delete-module( $x0, netlist$ )),
          pairstates(module-statenames(lookup-module( $x0, netlist$ )),
             $x2$ ),
          delete-module( $x0, netlist$ )))
    else f endif
case = 3
  then if listp( $x0$ )
    then cons(cons(occ-name(car( $x0$ )),
      dual-eval(2,
        occ-function(car( $x0$ )),
        collect-value(occ-inputs(car( $x0$ )),  $x1$ ),
        value(occ-name(car( $x0$ )),  $x2$ ),
         $netlist$ )),
      dual-eval(3, cdr( $x0$ ),  $x1, x2, netlist$ ))
    else nil endif
  otherwise f endcase

```

EVENT: Enable dual-eval; name this event ‘dual-eval-off’.

THEOREM: expand-top-level-dual-eval-0-calls

$$\begin{aligned}
 & ((\neg \text{primp}(fn)) \wedge \text{listp}(\text{lookup-module}(fn, netlist))) \\
 \rightarrow & \quad (\text{dual-eval}(0, fn, args, state, netlist) \\
 = & \quad \text{collect-value}(\text{module-outputs}(\text{lookup-module}(fn, netlist)), \\
 & \quad \text{dual-eval}(1, \\
 & \quad \quad \text{module-occurrences}(\text{lookup-module}(fn, netlist)), \\
 & \quad \text{pairlist}(\text{module-inputs}(\text{lookup-module}(fn,
 \end{aligned}$$

netlist)),
args),
 pairstates (module-statenames (lookup-module (*fn*,
netlist)),
state),
 delete-module (*fn*, *netlist*))))

THEOREM: expand-top-level-dual-eval-2-calls

(($\neg$ primp (*fn*)) $\wedge$ listp (lookup-module (*fn*, *netlist*)))
 $\rightarrow$ (dual-eval (2, *fn*, *args*, *state*, *netlist*)
 = collect-states (module-statenames (lookup-module (*fn*, *netlist*)),
 dual-eval (3,
 module-occurrences (lookup-module (*fn*,
netlist)),
 dual-eval (1,
 module-occurrences (lookup-module (*fn*,
netlist)),
 pairlist (module-inputs (lookup-module (*fn*,
netlist)),
args),
 pairstates (module-statenames (lookup-module (*fn*,
netlist)),
state),
 delete-module (*fn*, *netlist*)),
 pairstates (module-statenames (lookup-module (*fn*,
netlist)),
state),
 delete-module (*fn*, *netlist*))))

THEOREM: open-dual-eval-with-flag-1

((*body* $\simeq$ **nil**)
 $\rightarrow$ (dual-eval (1, *body*, *bindings*, *state-bindings*, *netlist*) = *bindings*)
 $\wedge$ (listp (*body*)
 $\rightarrow$ (dual-eval (1, *body*, *bindings*, *state-bindings*, *netlist*)
 = dual-eval (1,
 cdr (*body*),
 append (pairlist (occ-outputs (car (*body*)),
 dual-eval (0,
 occ-function (car (*body*)),
 collect-value (occ-inputs (car (*body*)),
bindings),
 value (occ-name (car (*body*)),
state-bindings),
netlist)),

bindings),
state-bindings,
netlist)))

THEOREM: open-dual-eval-with-flag-3

((*body* $\simeq$ **nil**) $\rightarrow$ (dual-eval (3, *body*, *bindings*, *state-bindings*, *netlist*) = **nil**))
 $\wedge$ (listp (*body*)
 $\rightarrow$ (dual-eval (3, *body*, *bindings*, *state-bindings*, *netlist*)
 $=$ cons (cons (occ-name (car (*body*)),
dual-eval (2,
occ-function (car (*body*)),
collect-value (occ-inputs (car (*body*)),
bindings),
value (occ-name (car (*body*)),
state-bindings),
netlist)),
dual-eval (3,
cdr (*body*),
bindings,
state-bindings,
netlist))))))

THEOREM: properp-dual-eval-0

(($\neg$ primp (*fn*)) $\wedge$ listp (lookup-module (*fn*, *netlist*)))
 $\rightarrow$ properp (dual-eval (0, *fn*, *args*, *state*, *netlist*))

THEOREM: length-dual-eval-0

(($\neg$ primp (*fn*)) $\wedge$ listp (lookup-module (*fn*, *netlist*)))
 $\rightarrow$ (length (dual-eval (0, *fn*, *args*, *state*, *netlist*))
 $=$ length (module-outputs (lookup-module (*fn*, *netlist*))))

THEOREM: properp-dual-eval-2

(($\neg$ primp (*fn*))
 $\wedge$ listp (lookup-module (*fn*, *netlist*))
 $\wedge$ properp (module-statenames (lookup-module (*fn*, *netlist*))))
 $\rightarrow$ properp (dual-eval (2, *fn*, *args*, *state*, *netlist*))

THEOREM: length-dual-eval-2

(($\neg$ primp (*fn*))
 $\wedge$ listp (lookup-module (*fn*, *netlist*))
 $\wedge$ properp (module-statenames (lookup-module (*fn*, *netlist*))))
 $\rightarrow$ (length (dual-eval (2, *fn*, *args*, *state*, *netlist*))
 $=$ length (module-statenames (lookup-module (*fn*, *netlist*))))

DEFINITION:

```

dual-eval-body-bindings(n, body, bindings, state-bindings, netlist)
=  if n  $\simeq$  0 then bindings
    else dual-eval-body-bindings(n - 1,
                                cdr(body),
                                append(pairlist(occ-outputs(car(body)),
                                                dual-eval(0,
                                                            occ-function(car(body)),
                                                            collect-value(occ-inputs(car(body)),
                                                                bindings),
                                                            value(occ-name(car(body)),
                                                                state-bindings),
                                                                netlist))),
                                bindings),
                                state-bindings,
                                netlist) endif

```

EVENT: Enable dual-eval-body-bindings; name this event ‘dual-eval-body-bindings-off’.

THEOREM: open-dual-eval-body-bindings

```

((n  $\simeq$  0)
 → (dual-eval-body-bindings(n, body, bindings, state-bindings, netlist)
   = bindings))
 $\wedge$  ((n  $\not\simeq$  0)
 → (dual-eval-body-bindings(n,
                             body,
                             bindings,
                             state-bindings,
                             netlist)
   = dual-eval-body-bindings(n - 1,
                             cdr(body),
                             append(pairlist(occ-outputs(car(body)),
                                                dual-eval(0,
                                                            occ-function(car(body)),
                                                            collect-value(occ-inputs(car(body)),
                                                                bindings),
                                                            value(occ-name(car(body)),
                                                                state-bindings),
                                                                netlist))),
                             bindings),
                             state-bindings,
                             netlist)))

```

DEFINITION:

```

simulate(fn, inputs, state, netlist)
=  if inputs  $\simeq$  nil then nil
    else cons(list(dual-eval(0, fn, car(inputs), state, netlist),
                    dual-eval(2, fn, car(inputs), state, netlist)),
              simulate(fn,
                      cdr(inputs),
                      dual-eval(2, fn, car(inputs), state, netlist),
                      netlist)) endif

```

DEFINITION:

```

simulate-dual-eval-2(module, inputs, state, netlist, n)
=  if n  $\simeq$  0 then state
    else simulate-dual-eval-2(module,
                              cdr(inputs),
                              dual-eval(2,
                                         module,
                                         car(inputs),
                                         state,
                                         netlist),
                              netlist,
                              n - 1) endif

```

EVENT: Enable simulate-dual-eval-2; name this event ‘simulate-dual-eval-2-off’.

THEOREM: open-simulate-dual-eval-2

```

((n  $\simeq$  0)  $\rightarrow$  (simulate-dual-eval-2(module, inputs, state, netlist, n) = state))
 $\wedge$  ((n  $\not\simeq$  0)
       $\rightarrow$  (simulate-dual-eval-2(module, inputs, state, netlist, n)
          = simulate-dual-eval-2(module,
                                cdr(inputs),
                                dual-eval(2,
                                           module,
                                           car(inputs),
                                           state,
                                           netlist),
                                netlist,
                                n - 1))))

```

DEFINITION:

```

min(x, y)
=  if y < x then y
    else x endif

```

EVENT: Enable min; name this event ‘min-off’.

DEFINITION:

```
insert(e, x)  
=  if e ∈ x then x  
    else cons(e, x) endif
```

EVENT: Enable insert; name this event ‘insert-off’.

DEFINITION: $\text{set-equal}(x, y) = (\text{subset}(x, y) \wedge \text{subset}(y, x))$

EVENT: Enable set-equal; name this event ‘set-equal-off’.

DEFINITION:

```
set-diff(x, y)  
=  if listp(x)  
    then if car(x) ∈ y then set-diff(cdr(x), y)  
        else cons(car(x), set-diff(cdr(x), y)) endif  
    else nil endif
```

EVENT: Enable set-diff; name this event ‘set-diff-off’.

DEFINITION:

```
intersection(x, y)  
=  if listp(x)  
    then if car(x) ∈ y then cons(car(x), intersection(cdr(x), y))  
        else intersection(cdr(x), y) endif  
    else nil endif
```

EVENT: Enable intersection; name this event ‘intersection-off’.

DEFINITION:

```
last-cdr(x)  
=  if listp(x) then last-cdr(cdr(x))  
    else x endif
```

EVENT: Enable last-cdr; name this event ‘last-cdr-off’.

DEFINITION:

```
list-duplicates(x, duplicates)  
=  if listp(x)  
    then list-duplicates(cdr(x),  
                        if car(x) ∈ cdr(x)  
                        then insert(car(x), duplicates)  
                        else duplicates endif)  
    else duplicates endif
```

EVENT: Enable list-duplicates; name this event ‘list-duplicates-off’.

DEFINITION:

$\text{remove-duplicates}(x)$

= **if** $\text{listp}(x)$ **then** **if** $\text{car}(x) \in \text{cdr}(x)$ **then** $\text{remove-duplicates}(\text{cdr}(x))$
 else $\text{cons}(\text{car}(x), \text{remove-duplicates}(\text{cdr}(x)))$ **endif**
else x **endif**

EVENT: Enable remove-duplicates; name this event ‘remove-duplicates-off’.

DEFINITION:

$\text{append-list}(x)$

= **if** $\text{listp}(x)$ **then** $\text{append}(\text{car}(x), \text{append-list}(\text{cdr}(x)))$
 else nil **endif**

EVENT: Enable append-list; name this event ‘append-list-off’.

DEFINITION: $\text{flatten-list}(x) = \text{remove-duplicates}(\text{append-list}(x))$

EVENT: Enable flatten-list; name this event ‘flatten-list-off’.

DEFINITION:

$\text{is-head}(x, y)$

= **if** $\text{listp}(x) \wedge \text{listp}(y)$ **then** $(\text{car}(x) = \text{car}(y)) \wedge \text{is-head}(\text{cdr}(x), \text{cdr}(y))$
 else $x \simeq \text{nil}$ **endif**

EVENT: Enable is-head; name this event ‘is-head-off’.

DEFINITION:

$\text{listify}(x)$

= **if** $\text{listp}(x)$ **then** $\text{cons}(\text{list}(\text{car}(x)), \text{listify}(\text{cdr}(x)))$
 else nil **endif**

EVENT: Enable listify; name this event ‘listify-off’.

DEFINITION:

$\text{alistp}(x)$

= **if** $\text{listp}(x)$ **then** $\text{listp}(\text{car}(x)) \wedge \text{alistp}(\text{cdr}(x))$
 else $x = \text{nil}$ **endif**

EVENT: Enable alistp; name this event ‘alistp-off’.

DEFINITION: $\text{boundp}(key, alist) = \text{listp}(\text{lookup-module}(key, alist))$

DEFINITION: $\text{alist-entry}(key, alist) = \text{lookup-module}(key, alist)$

EVENT: Enable alist-entry; name this event ‘alist-entry-off’.

DEFINITION: $\text{unbind}(key, alist) = \text{delete-module}(key, alist)$

DEFINITION:

$\text{set-value}(key, value, alist)$

= **if** $\text{alist-entry}(key, alist) \simeq \text{nil}$ **then** $\text{cons}(\text{cons}(key, value), alist)$
 elseif $value = \text{cdr}(\text{alist-entry}(key, alist))$ **then** $alist$
 else $\text{cons}(\text{cons}(key, value), \text{unbind}(key, alist))$ **endif**

EVENT: Enable set-value; name this event ‘set-value-off’.

DEFINITION:

$\text{union-values}(keys, alist)$

= **if** $\text{listp}(keys)$
 then $\text{value}(\text{car}(keys), alist) \cup \text{union-values}(\text{cdr}(keys), alist)$
 else nil endif

EVENT: Enable union-values; name this event ‘union-values-off’.

DEFINITION:

$\text{list-union-values}(list-of-args, alist)$

= **if** $\text{listp}(list-of-args)$
 then $\text{cons}(\text{union-values}(\text{car}(list-of-args), alist),$
 $\text{list-union-values}(\text{cdr}(list-of-args), alist))$
 else nil endif

EVENT: Enable list-union-values; name this event ‘list-union-values-off’.

DEFINITION:

$\text{unbound-keys}(keys, alist)$

= **if** $\text{listp}(keys)$
 then if $\text{boundp}(\text{car}(keys), alist)$ **then** $\text{unbound-keys}(\text{cdr}(keys), alist)$
 else $\text{insert}(\text{car}(keys), \text{unbound-keys}(\text{cdr}(keys), alist))$ **endif**
 else nil endif

EVENT: Enable unbound-keys; name this event ‘unbound-keys-off’.

DEFINITION:

$\text{unbind-list}(keys, alist)$
= **if** $\text{listp}(keys)$ **then** $\text{unbind}(\text{car}(keys), \text{unbind-list}(\text{cdr}(keys), alist))$
 else $alist$ **endif**

THEOREM: $\text{lookup-module-in-delete-module}$

$((k1 \neq k2) \wedge \text{lookup-module}(k1, m))$
 $\rightarrow \text{lookup-module}(k1, \text{delete-module}(k2, m))$

THEOREM: $\text{lookup-module-in-unbind-list}$

$((k \notin u) \wedge \text{lookup-module}(k, m)) \rightarrow \text{lookup-module}(k, \text{unbind-list}(u, m))$

EVENT: Enable $\text{lookup-module-in-delete-module}$; name this event ‘ $\text{lookup-module-in-delete-module-off}$ ’.

THEOREM: $\text{lessp-unbind-list-count}$

$((k \notin u) \wedge \text{boundp}(k, m))$
 $\rightarrow (\text{count}(\text{unbind-list}(\text{cons}(k, u), m)) < \text{count}(\text{unbind-list}(u, m)))$

EVENT: Enable boundp ; name this event ‘ boundp-off ’.

EVENT: Enable unbind ; name this event ‘ unbind-off ’.

EVENT: Enable unbind-list ; name this event ‘ unbind-list-off ’.

EVENT: Enable $\text{lookup-module-in-unbind-list}$; name this event ‘ $\text{lookup-module-in-unbind-list-off}$ ’.

DEFINITION:

$\text{m-states-list}(m\text{-states})$
= **if** $\text{listp}(m\text{-states}) \vee (m\text{-states} = \text{nil})$ **then** $m\text{-states}$
 else $\text{list}(m\text{-states})$ **endif**

EVENT: Enable m-states-list ; name this event ‘ m-states-list-off ’.

DEFINITION:

$\text{list-lessp}(x, y)$
= **if** $\text{listp}(x)$
 then if $\text{listp}(y)$
 then if $\text{car}(x) < \text{car}(y)$ **then** **t**
 elseif $\text{car}(x) = \text{car}(y)$ **then** $\text{list-lessp}(\text{cdr}(x), \text{cdr}(y))$
 else f **endif**


```

        else f endif
    else listp(y) endif

```

EVENT: Enable list-lessp; name this event ‘list-lessp-off’.

DEFINITION:

```

token-lessp(x, y)
=  if litatom(x) ∧ litatom(y) then list-lessp(unpack(x), unpack(y))
    elseif litatom(x) ∧ indexp(y) then t
    elseif indexp(x) ∧ litatom(y) then f
    elseif indexp(x) ∧ indexp(y)
    then if i-name(x) = i-name(y) then i-num(x) < i-num(y)
        else list-lessp(unpack(i-name(x)), unpack(i-name(y))) endif
    else f endif

```

EVENT: Enable token-lessp; name this event ‘token-lessp-off’.

DEFINITION:

```

min-token-help(seed, list)
=  if listp(list)
    then if token-lessp(seed, car(list))
        then min-token-help(seed, cdr(list))
        else min-token-help(car(list), cdr(list)) endif
    else seed endif

```

EVENT: Enable min-token-help; name this event ‘min-token-help-off’.

DEFINITION:

```

min-token(list)
=  if listp(list)
    then if listp(cdr(list)) then min-token-help(car(list), cdr(list))
        else car(list) endif
    else 0 endif

```

EVENT: Enable min-token; name this event ‘min-token-off’.

THEOREM: min-token-help-lemma

```

(min-token-help(seed, list) = seed)
∨ (min-token-help(seed, list) ∈ list)

```

THEOREM: min-token-lemma

```

listp(lst) → (min-token(lst) ∈ lst)

```

THEOREM: delete*-count
 $\text{count}(\text{delete}^*(e, x)) < \text{count}(\text{cons}(e, x))$

THEOREM: delete*-lemma
 $(e \in \text{lst}) \rightarrow (\text{count}(\text{delete}^*(e, \text{lst})) < \text{count}(\text{lst}))$

DEFINITION:
 $\text{list-sort}(\text{list})$
 $=$ **if** $\text{listp}(\text{list})$
 then $\text{cons}(\text{min-token}(\text{list}), \text{list-sort}(\text{delete}^*(\text{min-token}(\text{list}), \text{list})))$
 else nil endif

EVENT: Enable list-sort; name this event 'list-sort-off'.

DEFINITION:
 $\text{list-list-sort}(\text{list})$
 $=$ **if** $\text{listp}(\text{list})$
 then $\text{cons}(\text{list-sort}(\text{car}(\text{list})), \text{list-list-sort}(\text{cdr}(\text{list})))$
 else nil endif

EVENT: Enable list-list-sort; name this event 'list-list-sort-off'.

EVENT: Add the shell *net-error*, with recognizer function symbol *net-errorp* and 2 accessors: *error-label*, with type restriction (none-of) and default value false; *error-body*, with type restriction (none-of) and default value true.

DEFINITION: $\text{report-error}(\text{err}) = \text{err}$

EVENT: Enable report-error; name this event 'report-error-off'.

DEFINITION:
 $\text{pred-error}(\text{label}, \text{body}) = \text{report-error}(\text{net-error}(\text{label}, \text{body}))$

EVENT: Enable pred-error; name this event 'pred-error-off'.

DEFINITION:
 $\text{error-entry}(\text{lst}, \text{label})$
 $=$ **if** $\text{listp}(\text{lst})$
 then if $\text{net-errorp}(\text{car}(\text{lst}))$
 then if $\text{error-label}(\text{car}(\text{lst})) = \text{label}$ **then** $\text{car}(\text{lst})$
 else $\text{error-entry}(\text{cdr}(\text{lst}), \text{label})$ **endif**
 else $\text{error-entry}(\text{cdr}(\text{lst}), \text{label})$ **endif**
 else f endif

EVENT: Enable error-entry; name this event ‘error-entry-off’.

DEFINITION:

```
net-errors(lst)  
=  if listp(lst)  
    then if net-errorp(car(lst)) then cons(car(lst), net-errors(cdr(lst)))  
        else net-errors(cdr(lst)) endif  
    else nil endif
```

EVENT: Enable net-errors; name this event ‘net-errors-off’.

DEFINITION:

```
collect-net-errors(lst)  
=  if listp(lst)  
    then append(net-errors(car(lst)), collect-net-errors(cdr(lst)))  
    else nil endif
```

EVENT: Enable collect-net-errors; name this event ‘collect-net-errors-off’.

DEFINITION:

```
err-and(lst, label)  
=  if net-errors(lst)  $\simeq$  nil then t  
    else pred-error(label, net-errors(lst)) endif
```

EVENT: Enable err-and; name this event ‘err-and-off’.

DEFINITION:

```
t-or-err(p, label, body)  
=  if  $p \wedge (\neg \text{net-errorp}(p))$  then t  
    else pred-error(label, body) endif
```

EVENT: Enable t-or-err; name this event ‘t-or-err-off’.

DEFINITION:

```
nlistp-or-err(x, label)  
=  if  $x \simeq$  nil then t  
    else pred-error(label, x) endif
```

EVENT: Enable nlistp-or-err; name this event ‘nlistp-or-err-off’.

DEFINITION:

```
nil-or-err(x, label)
```

```
=  if  $x = \mathbf{nil}$  then t
    else pred-error ( $label, x$ ) endif
```

EVENT: Enable nil-or-err; name this event ‘nil-or-err-off’.

DEFINITION:

```
subset-or-err ( $x, y, label$ )
=  if set-diff ( $x, y$ )  $\simeq \mathbf{nil}$  then t
    else pred-error ( $label, \text{set-diff}(x, y)$ ) endif
```

EVENT: Enable subset-or-err; name this event ‘subset-or-err-off’.

DEFINITION:

```
no-duplicates-or-err ( $x, label$ )
=  if list-duplicates ( $x, \mathbf{nil}$ )  $\simeq \mathbf{nil}$  then t
    else pred-error ( $label, \text{list-duplicates}(x, \mathbf{nil})$ ) endif
```

EVENT: Enable no-duplicates-or-err; name this event ‘no-duplicates-or-err-off’.

DEFINITION:

```
disjoint-or-err ( $x, y, label$ )
=  if intersection ( $x, y$ )  $\simeq \mathbf{nil}$  then t
    else pred-error ( $label, \text{intersection}(x, y)$ ) endif
```

EVENT: Enable disjoint-or-err; name this event ‘disjoint-or-err-off’.

DEFINITION:

```
all-bound-or-err ( $keys, alist, label$ )
=  if unbound-keys ( $keys, alist$ )  $\simeq \mathbf{nil}$  then t
    else pred-error ( $label, \text{unbound-keys}(keys, alist)$ ) endif
```

EVENT: Enable all-bound-or-err; name this event ‘all-bound-or-err-off’.

DEFINITION:

```
label-error ( $x, label$ )
=  if net-errorp ( $x$ ) then pred-error ( $label, x$ )
    else  $x$  endif
```

EVENT: Enable label-error; name this event ‘label-error-off’.

DEFINITION:

```
list-collect-value ( $list-of-args, alist$ )
```

```
=  if list-of-args  $\simeq$  nil then nil
    else cons (flatten-list (collect-value (car (list-of-args), alist)),
               list-collect-value (cdr (list-of-args), alist)) endif
```

EVENT: Enable list-collect-value; name this event ‘list-collect-value-off’.

DEFINITION:

```
name-simple-okp (name)
=  (litatom (name)  $\vee$  (inexp (name)  $\wedge$  litatom (i-name (name))))
```

EVENT: Enable name-simple-okp; name this event ‘name-simple-okp-off’.

DEFINITION:

```
name-list-simple-okp (name-list)
=  if name-list  $\simeq$  nil then name-list = nil
    else name-simple-okp (car (name-list))
         $\wedge$  (car (name-list)  $\notin$  cdr (name-list))
         $\wedge$  name-list-simple-okp (cdr (name-list)) endif
```

EVENT: Enable name-list-simple-okp; name this event ‘name-list-simple-okp-off’.

DEFINITION:

```
occ-syntax-simple-okp (occ, netlist)
=  (((4 = length (occ))
       $\vee$  ((5 = length (occ))  $\wedge$  alistp (occ-annotation (occ))))
     $\wedge$  properp (occ)
     $\wedge$  name-simple-okp (occ-name (occ))
     $\wedge$  name-list-simple-okp (occ-outputs (occ))
     $\wedge$  properp (occ-inputs (occ))
     $\wedge$  if primp (occ-function (occ))
        then (length (primp2 (occ-function (occ), 'inputs))
                  = length (occ-inputs (occ)))
               $\wedge$  (length (primp2 (occ-function (occ), 'outputs))
                        = length (occ-outputs (occ)))
        else listp (lookup-module (occ-function (occ), netlist))
               $\wedge$  (length (module-inputs (lookup-module (occ-function (occ),
                                                         netlist)))
                    = length (occ-inputs (occ)))
               $\wedge$  (length (module-outputs (lookup-module (occ-function (occ),
                                                         netlist)))
                    = length (occ-outputs (occ))) endif)
```

EVENT: Enable occ-syntax-simple-okp; name this event ‘occ-syntax-simple-okp-off’.

DEFINITION:

```
occ-body-syntax-simple-okp (body,
                           signals,
                           non-depends,
                           occ-names,
                           outputs,
                           states,
                           netlist)
=  if body  $\simeq$  nil
    then (body = nil)
         $\wedge$  subset (outputs, signals)
         $\wedge$  subset (non-depends, signals)
         $\wedge$  subset (states, occ-names)
         $\wedge$  ( $\neg$  duplicates? (signals))
         $\wedge$  ( $\neg$  duplicates? (occ-names))
    elseif occ-syntax-simple-okp (car (body), netlist)
    then occ-body-syntax-simple-okp (cdr (body),
                                     append (occ-outputs (car (body)), signals),
                                     append (set-diff (occ-inputs (car (body)),
                                                         signals),
                                                         non-depends),
                                     cons (occ-name (car (body)), occ-names),
                                     outputs,
                                     states,
                                     netlist)

    else f endif
```

EVENT: Enable occ-body-syntax-simple-okp; name this event ‘occ-body-syntax-simple-okp-off’.

DEFINITION:

```
module-syntax-simple-okp (netlist)
=  (((length (car (netlist)) = 5)
      $\vee$  ((length (car (netlist)) = 6)
          $\wedge$  alistp (module-annotation (car (netlist))))))
 $\wedge$  properp (car (netlist))
 $\wedge$  name-simple-okp (module-name (car (netlist)))
 $\wedge$  ( $\neg$  lookup-module (module-name (car (netlist)), cdr (netlist)))
 $\wedge$  ( $\neg$  primp (module-name (car (netlist))))
 $\wedge$  name-list-simple-okp (module-inputs (car (netlist)))
```

```

^ name-list-simple-okp (module-outputs (car (netlist)))
^ name-list-simple-okp (m-states-list (module-statenames (car (netlist))))
^ disjoint (module-inputs (car (netlist)),
            module-outputs (car (netlist)))
^ listp (module-occurrences (car (netlist)))
^ occ-body-syntax-simple-okp (module-occurrences (car (netlist)),
                             module-inputs (car (netlist)),
                             nil,
                             nil,
                             module-outputs (car (netlist)),
                             m-states-list (module-statenames (car (netlist))),
                             cdr (netlist)))

```

EVENT: Enable module-syntax-simple-okp; name this event ‘module-syntax-simple-okp-off’.

DEFINITION:

```

netlist-syntax-simple-okp (netlist)
=  if netlist  $\simeq$  nil then netlist = nil
    elseif module-syntax-simple-okp (netlist)
    then netlist-syntax-simple-okp (cdr (netlist))
    else f endif

```

EVENT: Enable netlist-syntax-simple-okp; name this event ‘netlist-syntax-simple-okp-off’.

DEFINITION:

```

out-depends-simple (flg, x0, x1, netlist, module-database)
=  case on flg:
    case = 0
    then if primp (x0)
        then list-collect-value (primp2 (x0, 'out-depends),
                                         pairlist (primp2 (x0, 'inputs), x1))
        elseif lookup-module ('out-depends,
                               cdr (lookup-module (x0, module-database)))
        then list-collect-value (cdr (lookup-module ('out-depends,
                                                         cdr (lookup-module (x0,
                                                         module-database)))),
                                pairlist (module-inputs (lookup-module (x0,
                                                         netlist)),
                                         x1))
        elseif listp (lookup-module (x0, netlist))
        then if out-depends-simple (1,

```

```

                                module-occurrences (lookup-module (x0,
                                                                    netlist)),
                                pairlist (module-inputs (lookup-module (x0,
                                                                    netlist)),
                                x1),
                                delete-module (x0, netlist),
                                module-database)
    ∧ subset (module-outputs (lookup-module (x0, netlist)),
              strip-cars (out-depends-simple (1,
                                              module-occurrences (lookup-module (x0,
                                                                    netlist)),
                                              pairlist (module-inputs (lookup-module (x0,
                                                                    netlist)),
                                                                    x1),
                                              delete-module (x0,
                                                                    netlist),
                                                                    module-database)))
    then collect-value (module-outputs (lookup-module (x0, netlist)),
                        out-depends-simple (1,
                                              module-occurrences (lookup-module (x0,
                                                                    netlist)),
                                              pairlist (module-inputs (lookup-module (x0,
                                                                    netlist)),
                                                                    x1),
                                              delete-module (x0,
                                                                    netlist),
                                                                    module-database))

    else f endif
else f endif
case = 1
    then if x0  $\simeq$  nil then x1
        elseif subset (flatten-list (out-depends-simple (0,
                                                            occ-function (car (x0)),
                                                            listify (occ-inputs (car (x0))),
                                                            netlist,
                                                            module-database)),
                        strip-cars (x1))
        ∧ out-depends-simple (0,
                              occ-function (car (x0)),
                              listify (occ-inputs (car (x0))),
                              netlist,
                              module-database)
        then out-depends-simple (1,
                                cdr (x0),

```



```

                                append (pairlist (occ-outputs (car (x0)),
                                                list-collect-value (out-depends-simple (0,
                                                                occ-function (car
                                                                listify (occ-inputs
                                                                netlist,
                                                                module-database)
                                                                x1)),
                                x1),
                                netlist,
                                module-database)
                                else f endif
otherwise f endcase

```

EVENT: Enable out-depends-simple; name this event ‘out-depends-simple-off’.

DEFINITION:

simple-dependency-table (*netlist*)

```

=  if netlist  $\simeq$  nil then nil
    elseif simple-dependency-table (cdr (netlist))
    then if out-depends-simple (0,
                                module-name (car (netlist)),
                                listify (module-inputs (car (netlist))),
                                netlist,
                                simple-dependency-table (cdr (netlist)))
    then cons (list (module-name (car (netlist)),
                    cons ('out-depends,
                        out-depends-simple (0,
                                                module-name (car (netlist)),
                                                listify (module-inputs (car (netlist))),
                                                netlist,
                                                simple-dependency-table (cdr (netlist))))),
                    simple-dependency-table (cdr (netlist)))
    else f endif
    else f endif

```

EVENT: Enable simple-dependency-table; name this event ‘simple-dependency-table-off’.

DEFINITION:

output-dependencies-simple (*module-name*, *args*, *netlist*)

```

=  if (primp (module-name)
     $\wedge$  (length (args) = length (primp2 (module-name, 'inputs))))
     $\vee$  (lookup-module (module-name, netlist)

```

```

       $\wedge$  (length (args)
        = length (module-inputs (lookup-module (module-name,
                                                netlist))))))
then list-list-sort (out-depends-simple (0,
                                         module-name,
                                         args,
                                         netlist,
                                         simple-dependency-table (netlist)))
else f endif

```

EVENT: Enable output-dependencies-simple; name this event ‘output-dependencies-simple-off’.

DEFINITION:

```

type-value-simple (n, alist, default)
= if lookup-module (n, alist)  $\simeq$  nil
  then if default then 'free
    else list (n) endif
  elseif cdr (lookup-module (n, alist))  $\simeq$  nil
  then cdr (lookup-module (n, alist))
  elseif cadr (lookup-module (n, alist)) = n
  then if default then 'free
    else cdr (lookup-module (n, alist)) endif
  else type-value-simple (cadr (lookup-module (n, alist)),
                        delete-module (n, alist),
                        default) endif

```

EVENT: Enable type-value-simple; name this event ‘type-value-simple-off’.

DEFINITION:

```

collect-types (names, alist, default)
= if names  $\simeq$  nil then nil
  else cons (type-value-simple (car (names), alist, default),
            collect-types (cdr (names), alist, default)) endif

```

EVENT: Enable collect-types; name this event ‘collect-types-off’.

DEFINITION:

```

add-new-type (new-type-entry, known-types)
= if cdr (new-type-entry) = 'free then known-types
  elseif lookup-module (car (new-type-entry), known-types)  $\simeq$  nil
  then cons (new-type-entry, known-types)
  elseif cdr (new-type-entry)

```

```

      = cdr(lookup-module(car(new-type-entry), known-types))
then known-types
elseif listp(cdr(new-type-entry))
then cons(cons(cadr(new-type-entry),
               cdr(lookup-module(car(new-type-entry), known-types))),
          known-types)
elseif listp(cdr(lookup-module(car(new-type-entry), known-types)))
then if add-new-type(cons(cadr(lookup-module(car(new-type-entry),
                                             known-types)),
                          cdr(new-type-entry)),
                    delete-module(car(new-type-entry), known-types))
then cons(new-type-entry,
          add-new-type(cons(cadr(lookup-module(car(new-type-entry),
                                             known-types)),
                          cdr(new-type-entry)),
                    delete-module(car(new-type-entry),
                                   known-types)))
else f endif
else f endif

```

EVENT: Enable add-new-type; name this event ‘add-new-type-off’.

DEFINITION:

```

update-known-types(new-types, known-types)
= if (new-types  $\simeq$  nil)  $\vee$  falsep(known-types) then known-types
  else update-known-types(cdr(new-types),
                          add-new-type(car(new-types), known-types)) endif

```

EVENT: Enable update-known-types; name this event ‘update-known-types-off’.

DEFINITION:

```

fix-dependent-types(types, alist)
= if types  $\simeq$  nil then nil
  elseif listp(car(types))
  then cons(value(caar(types), alist),
            fix-dependent-types(cdr(types), alist))
  else cons(car(types), fix-dependent-types(cdr(types), alist)) endif

```

EVENT: Enable fix-dependent-types; name this event ‘fix-dependent-types-off’.

DEFINITION:

```

io-types-simple(flg, x0, x1, netlist, type-table)
= case on flg:

```

```

case = 0
then if primp (x0)
    then cons (primp2 (x0, 'input-types),
               fix-dependent-types (primp2 (x0, 'output-types),
                                           pairlist (primp2 (x0, 'inputs), x1)))
    elseif lookup-module ('input-types,
                          cdr (lookup-module (x0, type-table)))
    ^ lookup-module ('output-types,
                    cdr (lookup-module (x0, type-table)))
    then cons (cdr (lookup-module ('input-types,
                                   cdr (lookup-module (x0, type-table)))),
               fix-dependent-types (cdr (lookup-module ('output-types,
                                                         cdr (lookup-module (x0,
                                                                 type-table)))),
                                   pairlist (module-inputs (lookup-module (x0,
                                                                 netlist)),
                                             x1)))
    elseif listp (lookup-module (x0, netlist))
    then if io-types-simple (1,
                             module-occurrences (lookup-module (x0,
                                                                    netlist)),
                             nil,
                             delete-module (x0, netlist),
                             type-table)
    then cons (collect-types (module-inputs (lookup-module (x0,
                                                             netlist)),
                             io-types-simple (1,
                                                module-occurrences (lookup-module (x0,
                                                                 netlist)),
                                                nil,
                                                delete-module (x0,
                                                                netlist),
                                                                type-table),
                             t),
               fix-dependent-types (collect-types (module-outputs (lookup-module (x0,
                                                                 netlist)),
                                                  io-types-simple (1,
                                                                     module-occurrences (lookup-module (x0,
                                                                 netlist),
                                                                 nil,
                                                                 delete-module (x0,
                                                                 netlist),
                                                                 type-table),
                                                                     f),

```

```

pairlist (module-inputs (lookup-module (x0,
                                         netlist)),
          x1)))

    else f endif
  else f endif
case = 1
  then if x0  $\simeq$  nil then x1
    elseif listp (io-types-simple (0,
                                   occ-function (car (x0)),
                                   collect-types (occ-inputs (car (x0)),
                                                         x1,
                                                         f),
                                   netlist,
                                   type-table))
    then if update-known-types (pairlist (occ-outputs (car (x0)),
                                              cdr (io-types-simple (0,
                                                                      occ-function (car (x0)),
                                                                      collect-types (occ-inputs (car (x0)),
                                                                      x1,
                                                                      f),
                                                                      netlist,
                                                                      type-table))),
update-known-types (pairlist (occ-inputs (car (x0)),
                                   car (io-types-simple (0,
                                                           occ-function (car (
                                                           collect-types (occ-in
                                                                      x1,
                                                                      f),
                                                                      netlist,
                                                                      type-table))),
x1))
    then io-types-simple (1,
                          cdr (x0),
                          update-known-types (pairlist (occ-outputs (car (x0)),
                                                            cdr (io-types-simple (0,
                                                                occ-function (car (x0)),
                                                                collect-types (occ-in
                                                                x1,
                                                                f),
                                                                netlist,
                                                                type-table))),
update-known-types (pairlist (occ-inputs (car (x0)),
                                   car (io-types-simple

```

$x1))$,

$netlist$,

$type-table$)

else f endif

else f endif

otherwise f endcase

EVENT: Enable io-types-simple; name this event ‘io-types-simple-off’.

DEFINITION:

netlist-type-table-simple ($netlist$)

= **if** $netlist \simeq nil$ **then** nil

elseif netlist-type-table-simple (cdr ($netlist$))

then if listp (io-types-simple (0,

module-name (car ($netlist$)),

listify (module-inputs (car ($netlist$))),

$netlist$,

netlist-type-table-simple (cdr ($netlist$))))

then cons (list (module-name (car ($netlist$)),

cons (‘input-types,

car (io-types-simple (0,

module-name (car ($netlist$)),

listify (module-inputs (car ($netlist$))),

$netlist$,

netlist-type-table-simple (cdr ($netlist$))))),

cons (‘output-types,

cdr (io-types-simple (0,

module-name (car ($netlist$)),

listify (module-inputs (car ($netlist$))),

$netlist$,

netlist-type-table-simple (cdr ($netlist$))))),

netlist-type-table-simple (cdr ($netlist$)))

else f endif

else f endif

EVENT: Enable netlist-type-table-simple; name this event ‘netlist-type-table-simple-off’.

DEFINITION:

```

types-acceptable(flag, x0, netlist, acceptable-types, checked-modules)
=  case on flag:
    case = 0
    then if primp(x0)
        then subset(primp2(x0, 'input-types), acceptable-types)
            ∧ subset(fix-dependent-types(primp2(x0,
                                                'output-types),
                                                pairlist(primp2(x0,
                                                                'inputs),
                                                                primp2(x0,
                                                                'input-types))),
                                                                acceptable-types)
        elseif x0 ∈ checked-modules then t
        elseif listp(lookup-module(x0, netlist))
        then types-acceptable(1,
                                module-occurrences(lookup-module(x0,
                                                                    netlist)),
                                delete-module(x0, netlist),
                                acceptable-types,
                                checked-modules)
        else f endif
    case = 1
    then if x0 ≈ nil then t
        elseif types-acceptable(0,
                                occ-function(car(x0),
                                                netlist,
                                                acceptable-types,
                                                checked-modules)
                                then types-acceptable(1,
                                                        cdr(x0),
                                                        netlist,
                                                        acceptable-types,
                                                        checked-modules)
                                else f endif
    otherwise f endcase

```

EVENT: Enable types-acceptable; name this event 'types-acceptable-off'.

DEFINITION:

```

netlist-types-acceptable-list(netlist)
=  if netlist ≈ nil then nil
    elseif netlist-types-acceptable-list(cdr(netlist))
    then if types-acceptable(0,
                            module-name(car(netlist)),

```

```

                                netlist,
                                '(boolp clk level clk-10 free),
                                netlist-types-acceptable-list (cdr (netlist)))
    then cons (module-name (car (netlist)),
              netlist-types-acceptable-list (cdr (netlist)))
    else f endif
  else f endif

```

EVENT: Enable netlist-types-acceptable-list; name this event 'netlist-types-acceptable-list-off'.

DEFINITION:

```

netlist-type-check-simple-okp (netlist)
=  if netlist-types-acceptable-list (netlist)
    then netlist-type-table-simple (netlist)
    else f endif

```

EVENT: Enable netlist-type-check-simple-okp; name this event 'netlist-type-check-simple-okp-off'.

DEFINITION:

```

arg-types-match-simplep (actuals, formals)
=  if (actuals  $\simeq$  nil)  $\vee$  (formals  $\simeq$  nil) then actuals = formals
    else ((car (actuals) = car (formals))
           $\vee$  (car (formals) = 'free))
           $\wedge$  arg-types-match-simplep (cdr (actuals), cdr (formals)) endif

```

EVENT: Enable arg-types-match-simplep; name this event 'arg-types-match-simplep-off'.

DEFINITION:

```

arg-types-simple-okp (fn, arg-types, netlist)
=  if  $\neg$  subset (arg-types, '(boolp clk level clk-10)) then f
    elseif listp (io-types-simple (0,
                                    fn,
                                    arg-types,
                                    netlist,
                                    netlist-type-table-simple (netlist)))
    then arg-types-match-simplep (arg-types,
                                  car (io-types-simple (0,
                                                          fn,
                                                          arg-types,
                                                          netlist,

```


netlist-type-table-simple (*netlist*)))

else f endif

EVENT: Enable arg-types-simple-okp; name this event ‘arg-types-simple-okp-off’.

DEFINITION:

make-ram-state (*structure*, *width*, *bit-value*)
 = **if** *structure* $\simeq$ **nil** **then** ram (make-list (*width*, *bit-value*))
 else cons (make-ram-state (car (*structure*), *width*, *bit-value*),
 make-ram-state (cdr (*structure*), *width*, *bit-value*)) **endif**

EVENT: Enable make-ram-state; name this event ‘make-ram-state-off’.

DEFINITION:

state-type-requirement-simple (*flg*, *x0*, *x1*, *netlist*, *type-table*)
 = **case on** *flg*:
 case = 0
 then if primp (*x0*)
 then if lookup-module (‘state-types, cdr (primp (*x0*)))
 then eval\$ (**t**,
 cdr (lookup-module (‘state-types,
 cdr (primp (*x0*))),
 nil)
 else t endif
 elseif lookup-module (*x0*, *type-table*)
 then if lookup-module (‘state-types,
 cdr (lookup-module (*x0*, *type-table*)))
 then cdr (lookup-module (‘state-types,
 cdr (lookup-module (*x0*, *type-table*))))
 else t endif
 elseif listp (lookup-module (*x0*, *netlist*))
 then if set-equal (m-states-list (module-statenames (lookup-module (*x0*,
 netlist))),
 strip-cars (state-type-requirement-simple (1,
 module-occurrences (lookup-module (*x0*,
 netlist),
 nil,
 delete-module (*x0*,
 netlist),
 type-table)))
 type-table)))
 then if module-statenames (lookup-module (*x0*, *netlist*))
 = **nil** **then t**

```

        else collect-states (module-statenames (lookup-module (x0,
                                                                netlist)),
                             state-type-requirement-simple (1,
                                                             module-occurrences (lookup-module (
                                                                 nil,
                                                                 delete-module (x0,
                                                                 netlist),
                                                                 type-table)) endif
                                                             )
                             )
        else f endif
    else f endif
case = 1
    then if x0  $\simeq$  nil then x1
        elseif truep (state-type-requirement-simple (0,
                                                       occ-function (car (x0)),
                                                       nil,
                                                       netlist,
                                                       type-table))
            then state-type-requirement-simple (1,
                                                 cdr (x0),
                                                 x1,
                                                 netlist,
                                                 type-table)
            elseif state-type-requirement-simple (0,
                                                  occ-function (car (x0)),
                                                  nil,
                                                  netlist,
                                                  type-table)
            then state-type-requirement-simple (1,
                                                 cdr (x0),
                                                 cons (cons (occ-name (car (x0)),
                                                             state-type-requirement-simple (0,
                                                                 occ-function (car (x0),
                                                                 nil,
                                                                 netlist,
                                                                 type-table)),
                                                             x1),
                                                 netlist,
                                                 type-table)
            else f endif
    otherwise f endcase

```

EVENT: Enable state-type-requirement-simple; name this event ‘state-type-requirement-simple-off’.

DEFINITION:

netlist-state-types-simple (*netlist*)

```
=  if netlist  $\simeq$  nil then nil
    elseif netlist-state-types-simple (cdr (netlist))
    then if truep (state-type-requirement-simple (0,
                                                    module-name (car (netlist)),
                                                    nil,
                                                    netlist,
                                                    netlist-state-types-simple (cdr (netlist))))
        then netlist-state-types-simple (cdr (netlist))
        elseif state-type-requirement-simple (0,
                                                module-name (car (netlist)),
                                                nil,
                                                netlist,
                                                netlist-state-types-simple (cdr (netlist)))
        then cons (list (module-name (car (netlist)),
                        cons ('state-types,
                            state-type-requirement-simple (0,
                                                            module-name (car (netlist)),
                                                            nil,
                                                            netlist,
                                                            netlist-state-types-simple (cdr (netlist))))),
                    netlist-state-types-simple (cdr (netlist)))
    else f endif
else f endif
```

EVENT: Enable netlist-state-types-simple; name this event 'netlist-state-types-simple-off'.

DEFINITION:

state-simple-okp (*state*, *type*)

```
=  if truep (type) then state = nil
    elseif type = 'boolp then boolp (state)
    elseif ramp (type)
    then ramp (state)
         $\wedge$  state-simple-okp (ram-guts (state), ram-guts (type))
    elseif romp (type)
    then romp (state)
         $\wedge$  state-simple-okp (rom-guts (state), rom-guts (type))
    elseif stubp (type)
    then stubp (state)
         $\wedge$  state-simple-okp (stub-guts (state), stub-guts (type))
    elseif listp (type)
    then listp (state)
```

```

      ∧ state-simple-okp (car (state), car (type))
      ∧ state-simple-okp (cdr (state), cdr (type))
elseif type = nil then state = nil
else f endif

```

EVENT: Enable state-simple-okp; name this event ‘state-simple-okp-off’.

DEFINITION:

```

apply-state-simple-okp (fn, state, netlist)
= state-simple-okp (state,
                    state-type-requirement-simple (0,
                                                    fn,
                                                    nil,
                                                    netlist,
                                                    netlist-state-types-simple (netlist)))

```

EVENT: Enable apply-state-simple-okp; name this event ‘apply-state-simple-okp-off’.

DEFINITION:

```

fix-dependent-lds (lds, alist)
= if lds  $\simeq$  nil then lds
  else cons (if litatom (car (lds))  $\vee$  indexp (car (lds))
            then value (car (lds), alist)
            elseif car (lds)  $\simeq$  nil then car (lds)
            else fix-dependent-lds (car (lds), alist) endif,
            fix-dependent-lds (cdr (lds), alist)) endif

```

EVENT: Enable fix-dependent-lds; name this event ‘fix-dependent-lds-off’.

DEFINITION:

```

parent-synonym-simple (name, slist)
= if listp (lookup-module (name, slist))
  then if litatom (cdr (lookup-module (name, slist)))
     $\vee$  indexp (cdr (lookup-module (name, slist)))
    then parent-synonym-simple (cdr (lookup-module (name, slist)),
                                delete-module (name, slist))
    else name endif
  else name endif

```

EVENT: Enable parent-synonym-simple; name this event ‘parent-synonym-simple-off’.

DEFINITION:

```

add-loading-simple (name, loading, ld-list)
=  if lookup-module (name, ld-list)  $\simeq$  nil
    then cons (cons (name,
                    if listp (loading) then loading
                    else list (loading) endif),
              ld-list)
    else cons (cons (name,
                    if listp (loading)
                    then append (loading,
                                cdr (lookup-module (name, ld-list)))
                    else cons (loading,
                                cdr (lookup-module (name, ld-list))) endif),
              delete-module (name, ld-list)) endif

```

EVENT: Enable add-loading-simple; name this event ‘add-loading-simple-off’.

DEFINITION:

```

add-loading-simples (names, loadings, ld-list)
=  if names  $\simeq$  nil then ld-list
    else add-loading-simples (cdr (names),
                              cdr (loadings),
                              add-loading-simple (car (names),
                                                    car (loadings),
                                                    ld-list)) endif

```

EVENT: Enable add-loading-simples; name this event ‘add-loading-simples-off’.

DEFINITION:

```

sum-numbers (lst)
=  if lst  $\simeq$  nil then 0
    elseif car (lst)  $\in \mathbf{N}$  then car (lst) + sum-numbers (cdr (lst))
    else sum-numbers (cdr (lst)) endif

```

EVENT: Enable sum-numbers; name this event ‘sum-numbers-off’.

DEFINITION:

```

extract-names-simple (lst)
=  if lst  $\simeq$  nil then nil
    elseif (litatom (car (lst))  $\vee$  indexp (car (lst)))
           $\wedge$  (car (lst)  $\notin$  cdr (lst))
    then cons (car (lst), extract-names-simple (cdr (lst)))
    else extract-names-simple (cdr (lst)) endif

```

EVENT: Enable extract-names-simple; name this event ‘extract-names-simple-off’.

DEFINITION:

```
sum-loading(ld)
=  cons(car(ld),
      if extract-names-simple(cdr(ld))  $\simeq$  nil
      then list(sum-numbers(cdr(ld)))
      elseif sum-numbers(cdr(ld))  $\simeq$  0
      then extract-names-simple(cdr(ld))
      else cons(sum-numbers(cdr(ld)),
                extract-names-simple(cdr(ld))) endif)
```

EVENT: Enable sum-loading; name this event ‘sum-loading-off’.

DEFINITION:

```
sum-loadings(lds)
=  if lds  $\simeq$  nil then nil
    else cons(sum-loading(car(lds)), sum-loadings(cdr(lds))) endif
```

EVENT: Enable sum-loadings; name this event ‘sum-loadings-off’.

DEFINITION:

```
fix-loadings(slist, lds)
=  if slist  $\simeq$  nil then sum-loadings(lds)
    elseif litatom(cdar(slist))  $\vee$  indexp(cdar(slist))
    then fix-loadings(cdr(slist),
                      add-loading-simple(cdar(slist),
                                          value(caar(slist), lds),
                                          delete-module(caar(slist), lds)))
    else fix-loadings(cdr(slist), lds) endif
```

EVENT: Enable fix-loadings; name this event ‘fix-loadings-off’.

DEFINITION:

```
fix-drives(drs, slist)
=  if drs  $\simeq$  nil then nil
    elseif litatom(cdar(drs))  $\vee$  indexp(cdar(drs))
    then cons(cons(caar(drs), parent-synonym-simple(cdar(drs), slist)),
              fix-drives(cdr(drs), slist))
    else cons(car(drs), fix-drives(cdr(drs), slist)) endif
```

EVENT: Enable fix-drives; name this event ‘fix-drives-off’.

DEFINITION:

```

net-drives(drs, lds)
=  if drs  $\simeq$  nil then nil
    elseif cdr(drs)  $\in$  N
    then if cdr(drs) < sum-numbers(value(caar(drs), lds)) then f
        elseif net-drives(cdr(drs), lds)
        then cons(cons(caar(drs),
                        cdr(drs) - sum-numbers(value(caar(drs), lds))),
                  net-drives(cdr(drs), lds))
        else f endif
    elseif net-drives(cdr(drs), lds)
    then cons(car(drs), net-drives(cdr(drs), lds))
    else f endif

```

EVENT: Enable net-drives; name this event ‘net-drives-off’.

DEFINITION:

```

external-loading(ld-lst, external-names)
=  if ld-lst  $\simeq$  nil then ld-lst
    elseif (litatom(car(ld-lst))  $\vee$  indexp(car(ld-lst)))
         $\wedge$  (car(ld-lst)  $\notin$  external-names)
    then external-loading(cdr(ld-lst), external-names)
    else cons(car(ld-lst),
              external-loading(cdr(ld-lst), external-names)) endif

```

EVENT: Enable external-loading; name this event ‘external-loading-off’.

DEFINITION:

```

collect-loadings-simple(names, loadings, external-names)
=  if names  $\simeq$  nil then nil
    elseif lookup-module(car(names), loadings)
    then cons(if external-loading(value(car(names), loadings),
                                   external-names)  $\simeq$  nil then 0
              elseif cdr(external-loading(value(car(names), loadings),
                                             external-names))  $\simeq$  nil
              then car(external-loading(value(car(names), loadings),
                                             external-names))
              else external-loading(value(car(names), loadings),
                                     external-names) endif,
              collect-loadings-simple(cdr(names), loadings, external-names))
    else cons(0,
              collect-loadings-simple(cdr(names),
                                       loadings,
                                       external-names)) endif

```

EVENT: Enable collect-loadings-simple; name this event ‘collect-loadings-simple-off’.

DEFINITION:

```
collect-drives-simple (names, drives, external-names)
=  if names  $\simeq$  nil then nil
    elseif ( $\neg$  (litatom (value (car (names), drives)))
               $\vee$  (indexp (value (car (names), drives))))
               $\vee$  (value (car (names), drives)  $\in$  external-names)
    then cons (value (car (names), drives),
               collect-drives-simple (cdr (names), drives, external-names))
    elseif parent-synonym-simple (value (car (names), drives), drives)
      = value (car (names), drives)
    then cons (value (value (car (names), drives), drives),
               collect-drives-simple (cdr (names),
                                       cons (cons (value (car (names), drives),
                                                  car (names)),
                                             cons (cons (car (names),
                                                         value (value (car (names),
                                                                    drives),
                                                                    drives))),
                                                delete-module (car (names),
                                                                delete-module (value (car (names),
                                                                                   drives),
                                                                                   drives))))),
                                       external-names))
    else cons (parent-synonym-simple (value (car (names), drives), drives),
               collect-drives-simple (cdr (names),
                                       drives,
                                       external-names)) endif
```

EVENT: Enable collect-drives-simple; name this event ‘collect-drives-simple-off’.

DEFINITION:

```
loadings-and-drives-simple (flg, x0, x1, x2, netlist, lds-table)
=  case on flg:
    case = 0
    then if primp (x0)
      then cons (fix-dependent-lds (primp2 (x0, 'loadings),
                                           append (pairlist (primp2 (x0, 'inputs),
                                                                    x1),
                                                  pairlist (primp2 (x0,
                                                                    'outputs),
```



```

                                x2))),
fix-dependent-lds (primp2 (x0, 'drives),
                      append (pairlist (primp2 (x0, 'inputs),
                                                x1),
                                pairlist (primp2 (x0,
                                                    'outputs),
                                                x2))))
elseif lookup-module ('loadings,
                      cdr (lookup-module (x0, lds-table)))
   $\wedge$  lookup-module ('drives,
                      cdr (lookup-module (x0, lds-table)))
then cons (fix-dependent-lds (cdr (lookup-module ('loadings,
                                                  cdr (lookup-module (x0,
                                                                      lds-table))))),
            append (pairlist (module-inputs (lookup-module (x0,
                                                            netlist)),
                                x1),
                    pairlist (module-outputs (lookup-module (x0,
                                                                netlist)),
                                x2))),
            fix-dependent-lds (cdr (lookup-module ('drives,
                                                  cdr (lookup-module (x0,
                                                                      lds-table))))),
            append (pairlist (module-inputs (lookup-module (x0,
                                                            netlist)),
                                x1),
                    pairlist (module-outputs (lookup-module (x0,
                                                                netlist)),
                                x2))))
elseif listp (lookup-module (x0, netlist))
then if loadings-and-drives-simple (1,
                                     module-occurrences (lookup-module (x0,
                                                                           netlist)),
                                     nil,
                                     nil,
                                     delete-module (x0, netlist),
                                     lds-table)
then cons (fix-dependent-lds (collect-loadings-simple (module-inputs (lookup-module (x0,
                                                                                      netlist))),
                                                                    car (loadings-and-drives-simple (1,
                                                                                      module-
                                                                                      nil,
                                                                                      nil,

```

```

delete-m
lds-table
module-outputs (lookup-module (x0,
netlist
append (pairlist (module-inputs (lookup-module (x0,
netlist)),
x1),
pairlist (module-outputs (lookup-module (x0,
netlist)),
x2))),
fix-dependent-lds (collect-drives-simple (module-outputs (lookup-module (x0,
netlist)),
cdr (loadings-and-drives-simple (1,
module-oc
nil,
nil,
delete-mo
lds-table))
append (module-inputs (lookup-module (x0,
n
module-outputs (lookup-module (x0,
n
append (pairlist (module-inputs (lookup-module (x0,
netlist)),
x1),
pairlist (module-outputs (lookup-module (x0,
netlist)),
x2))))))
else f endif
else f endif
case = 1
then if x0 ≈ nil
then if fix-loadings (fix-drives (x2, x2), x1)
∧ net-drives (fix-drives (x2, x2),
fix-loadings (fix-drives (x2, x2), x1))
then cons (fix-loadings (fix-drives (x2, x2), x1),
net-drives (fix-drives (x2, x2),
fix-loadings (fix-drives (x2, x2), x1)))
else f endif
elseif loadings-and-drives-simple (0,
occ-function (car (x0)),

```

```

                                occ-inputs (car (x0)),
                                occ-outputs (car (x0)),
                                netlist,
                                lds-table)
then loadings-and-drives-simple (1,
                                cdr (x0),
                                add-loading-simples (occ-inputs (car (x0)),
                                                    car (loadings-and-drives-simple (0,
                                                    occ-function (ca
                                                    occ-inputs (ca
                                                    occ-outputs (ca
                                                    netlist,
                                                    lds-table))),
                                                    x1),
                                append (pairlist (occ-outputs (car (x0)),
                                                    cdr (loadings-and-drives-simple (0,
                                                    occ-function (ca
                                                    occ-inputs (ca
                                                    occ-outputs (ca
                                                    netlist,
                                                    lds-table)))),
                                                    x2),
                                netlist,
                                lds-table)
else f endif
otherwise f endcase

```

EVENT: Enable loadings-and-drives-simple; name this event ‘loadings-and-drives-simple-off’.

DEFINITION:

netlist-loadings-and-drives-simple (*netlist*)

= **if** *netlist* $\simeq$ **nil** **then** **nil**

elseif netlist-loadings-and-drives-simple (cdr (*netlist*))

then if loadings-and-drives-simple (0,

module-name (car (*netlist*)),
module-inputs (car (*netlist*)),
module-outputs (car (*netlist*)),
netlist,
netlist-loadings-and-drives-simple (cdr (*netlist*)))

then cons (list (module-name (car (*netlist*)),

cons (‘loadings,

car (loadings-and-drives-simple (0,

module-name (car (*netlist*))),

```

module-inputs (car (netlist)),
module-outputs (car (netlist)),
netlist,
netlist-loadings-and-drives-simple (cdr (netlist))
cons ('drives,
      cdr (loadings-and-drives-simple (0,
module-name (car (netlist)),
module-inputs (car (netlist)),
module-outputs (car (netlist)),
netlist,
netlist-loadings-and-drives-simple (cdr (netlist))
netlist-loadings-and-drives-simple (cdr (netlist)))
else f endif
else f endif

```

EVENT: Enable netlist-loadings-and-drives-simple; name this event 'netlist-loadings-and-drives-simple-off'.

DEFINITION:

```

top-level-predicate-simple (netlist)
= if ¬ netlist-syntax-simple-okp (netlist)
  then 'netlist-syntax-simple-okp-error
  elseif ¬ simple-dependency-table (netlist)
  then 'dependency-table-simple-error
  elseif ¬ netlist-type-check-simple-okp (netlist)
  then 'netlist-type-check-simple-okp-error
  elseif ¬ netlist-state-types-simple (netlist)
  then 'netlist-state-types-simple-error
  elseif ¬ netlist-loadings-and-drives-simple (netlist)
  then 'netlist-loadings-and-drives-simple-error
  else t endif

```

DEFINITION:

```

name-okp (name)
= if litatom (name) then t
  else indexp (name) ∧ litatom (i-name (name)) endif

```

EVENT: Enable name-okp; name this event 'name-okp-off'.

DEFINITION:

```

bad-names (name-list)
= if listp (name-list)
  then if name-okp (car (name-list)) then bad-names (cdr (name-list))
       else insert (car (name-list), bad-names (cdr (name-list))) endif

```

else nil endif

EVENT: Enable bad-names; name this event 'bad-names-off'.

DEFINITION:

```
name-list-errors (name-list, duplicates-ok?)  
= list (nlistp-or-err (bad-names (name-list), 'bad-names),  
       if duplicates-ok? then t  
       else no-duplicates-or-err (name-list, 'duplicates) endif,  
       nil-or-err (last-cdr (name-list), 'not-proper-list))
```

EVENT: Enable name-list-errors; name this event 'name-list-errors-off'.

DEFINITION:

```
name-list-ok (name-list, label, duplicates-ok?)  
= err-and (name-list-errors (name-list, duplicates-ok?), label)
```

EVENT: Enable name-list-ok; name this event 'name-list-ok-off'.

DEFINITION:

```
occ-arg-length-error (fn, expected-args, actual-args, label)  
= t-or-err (length (expected-args) = length (actual-args),  
           label,  
           list ('occ-function,  
               fn,  
               list ('expected, length (expected-args)),  
               list ('got, length (actual-args))))
```

EVENT: Enable occ-arg-length-error; name this event 'occ-arg-length-error-off'.

DEFINITION:

```
occ-form-ok (occ, o-length)  
= err-and (list (t-or-err ((4 = o-length) ∨ (5 = o-length),  
                          'bad-occurrence-length,  
                          o-length),  
                nil-or-err (last-cdr (occ), 'not-proper-list)),  
          'occurrence-form)
```

EVENT: Enable occ-form-ok; name this event 'occ-form-ok-off'.

DEFINITION:

```
occ-name-ok (o-name, o-length)
```

```

=  if if o-length < 1 then f
    else t endif
    then label-error (t-or-err (name-okp (o-name), 'bad-name, o-name),
                          'occurrence-name)
    else t endif

```

EVENT: Enable occ-name-ok; name this event 'occ-name-ok-off'.

DEFINITION:

```

occ-outputs-ok (o-outs, o-length)
=  if if o-length < 2 then f
    else t endif
    then name-list-ok (o-outs, 'occurrence-outputs, f)
    else t endif

```

EVENT: Enable occ-outputs-ok; name this event 'occ-outputs-ok-off'.

DEFINITION:

```

occ-inputs-ok (o-ins, o-length)
=  if if o-length < 4 then f
    else t endif then name-list-ok (o-ins, 'occurrence-inputs, t)
    else t endif

```

EVENT: Enable occ-inputs-ok; name this event 'occ-inputs-ok-off'.

DEFINITION:

```

occ-function-ok (o-fn, o-ins, o-outs, o-length, netlist)
=  if o-length < 3 then t
    elseif primp (o-fn)
    then err-and (list (if if o-length < 4 then f
                        else t endif
                        then occ-arg-length-error (o-fn,
                                                    primp2 (o-fn, 'inputs),
                                                    o-ins,
                                                    'wrong-number-of-inputs)
                        else t endif,
                        occ-arg-length-error (o-fn,
                                                    primp2 (o-fn, 'outputs),
                                                    o-outs,
                                                    'wrong-number-of-outputs)),
                  'occurrence-function)
    elseif listp (lookup-module (o-fn, netlist))
    then err-and (list (if if o-length < 4 then f

```

```

        else t endif
    then occ-arg-length-error (o-fn,
                               module-inputs (lookup-module (o-fn,
                                                                netlist)),
                               o-ins,
                               'wrong-number-of-inputs)
    else t endif,
    occ-arg-length-error (o-fn,
                          module-outputs (lookup-module (o-fn,
                                                            netlist)),
                          o-outs,
                          'wrong-number-of-outputs)),
    'occurrence-function)
else pred-error ('unknown-occurrence-function, o-fn) endif

```

EVENT: Enable occ-function-ok; name this event 'occ-function-ok-off'.

DEFINITION:

```

occ-annotation-ok (o-anno, o-length)
=   if o-length = 5
    then label-error (t-or-err (alistp (o-anno), 'not-alist, o-anno),
                       'occurrence-annotation)
    else t endif

```

EVENT: Enable occ-annotation-ok; name this event 'occ-annotation-ok-off'.

DEFINITION:

```

occ-syntax-errors (occ, netlist)
=   list (occ-form-ok (occ, length (occ)),
          occ-name-ok (occ-name (occ), length (occ)),
          occ-outputs-ok (occ-outputs (occ), length (occ)),
          occ-inputs-ok (occ-inputs (occ), length (occ)),
          occ-function-ok (occ-function (occ),
                           occ-inputs (occ),
                           occ-outputs (occ),
                           length (occ),
                           netlist),
          occ-annotation-ok (occ-annotation (occ), length (occ)))

```

EVENT: Enable occ-syntax-errors; name this event 'occ-syntax-errors-off'.

DEFINITION:

```

occ-syntax-ok (occ, netlist)

```

```
= err-and (occ-syntax-errors (occ, netlist),
          list ('occurrence, occ-name (occ)))
```

EVENT: Enable occ-syntax-ok; name this event 'occ-syntax-ok-off'.

DEFINITION:

```
initial-occ-syntax-data (m-ins, m-outs)
= cons (cons ('signals, m-ins),
        cons (cons ('pending-io, intersection (m-ins, m-outs)),
              '((non-depends) (occ-names))))
```

EVENT: Enable initial-occ-syntax-data; name this event 'initial-occ-syntax-data-off'.

DEFINITION:

```
update-occ-syntax-data (occ, occ-data)
= list (cons ('signals,
              append (if listp (value ('pending-io, occ-data))
                      then set-diff (occ-outputs (occ),
                                     value ('pending-io, occ-data))
                      else occ-outputs (occ) endif,
                      value ('signals, occ-data))),
        cons ('pending-io,
              set-diff (value ('pending-io, occ-data), occ-outputs (occ))),
        cons ('non-depends,
              append (set-diff (occ-inputs (occ), value ('signals, occ-data)),
                      value ('non-depends, occ-data))),
        cons ('occ-names,
              cons (occ-name (occ), value ('occ-names, occ-data))))
```

EVENT: Enable update-occ-syntax-data; name this event 'update-occ-syntax-data-off'.

DEFINITION:

```
composite-occ-body-syntax-errors (body, occ-data, outputs, states)
= list (nil-or-err (body, 'not-proper-list),
        nlistp-or-err (value ('pending-io, occ-data),
                       'io-signals-not-in-outputs),
        subset-or-err (outputs,
                       value ('signals, occ-data),
                       'outputs-not-in-signals),
        subset-or-err (value ('non-depends, occ-data),
                       value ('signals, occ-data),
```



```

                                'non-depends-not-in-signals),
subset-or-err (states,
              value ('occ-names, occ-data),
              'states-not-in-occ-names),
no-duplicates-or-err (value ('signals, occ-data),
                    'duplicates-in-signals),
no-duplicates-or-err (value ('occ-names, occ-data),
                    'duplicates-in-occ-names))

```

EVENT: Enable composite-occ-body-syntax-errors; name this event 'composite-occ-body-syntax-errors-off'.

DEFINITION:

```

occ-body-syntax-errors (body, occ-data, outputs, states, netlist)
=  if body  $\simeq$  nil
    then composite-occ-body-syntax-errors (body, occ-data, outputs, states)
    else cons (occ-syntax-ok (car (body), netlist),
              occ-body-syntax-errors (cdr (body),
                                     update-occ-syntax-data (car (body),
                                                             occ-data),
                                     outputs,
                                     states,
                                     netlist)) endif

```

EVENT: Enable occ-body-syntax-errors; name this event 'occ-body-syntax-errors-off'.

DEFINITION:

```

module-form-ok (module, m-length)
=  err-and (list (t-or-err ((m-length = 5)  $\vee$  (m-length = 6),
                          'bad-module-length,
                          m-length),
                nil-or-err (last-cdr (module), 'not-proper-list)),
            'module-form)

```

EVENT: Enable module-form-ok; name this event 'module-form-ok-off'.

DEFINITION:

```

module-name-ok (m-name, m-length, netlist)
=  if if m-length < 1 then f
    else t endif
    then err-and (list (t-or-err (name-okp (m-name), 'bad-name, m-name),
                              t-or-err ( $\neg$  lookup-module (m-name, netlist),

```

```

        'duplicate-module-defns,
        m-name),
    t-or-err (¬ primp (m-name),
        'module-name-same-as-primitive,
        m-name)),
    'module-name)
else t endif

```

EVENT: Enable module-name-ok; name this event 'module-name-ok-off'.

DEFINITION:

```

module-inputs-ok (m-ins, m-length)
=  if if m-length < 2 then f
    else t endif then name-list-ok (m-ins, 'module-inputs, f)
    else t endif

```

EVENT: Enable module-inputs-ok; name this event 'module-inputs-ok-off'.

DEFINITION:

```

module-outputs-ok (m-outs, m-length)
=  if if m-length < 3 then f
    else t endif then name-list-ok (m-outs, 'module-outputs, f)
    else t endif

```

EVENT: Enable module-outputs-ok; name this event 'module-outputs-ok-off'.

DEFINITION:

```

states-list-or-nil (m-states, m-length)
=  if if m-length < 5 then f
    else t endif then m-states-list (m-states)
    else nil endif

```

EVENT: Enable states-list-or-nil; name this event 'states-list-or-nil-off'.

DEFINITION:

```

module-occurrences-ok (m-occs, m-ins, m-outs, m-states, m-length, netlist)
=  if if m-length < 4 then f
    else t endif
    then err-and (cons (t-or-err (listp (m-occs), 'no-occurrences, m-occs),
        occ-body-syntax-errors (m-occs,
            initial-occ-syntax-data (m-ins,
                                    m-outs),
            m-outs,

```

```

states-list-or-nil (m-states,
                   m-length),
netlist)),
'module-occurrences)
else t endif

```

EVENT: Enable module-occurrences-ok; name this event 'module-occurrences-ok-off'.

DEFINITION:

```

module-statenames-ok (m-states, m-length)
=  if if m-length < 5 then f
    else t endif
    then err-and (cons (t-or-err (if listp (m-states)
                                then listp (cdr (m-states))
                                else t endif,
                                'list-of-length-1-not-allowed,
                                m-states),
                      name-list-errors (m-states-list (m-states), f)),
                  'module-statenames)
    else t endif

```

EVENT: Enable module-statenames-ok; name this event 'module-statenames-ok-off'.

DEFINITION:

```

module-annotation-ok (m-anno, m-length)
=  if m-length = 6
    then label-error (t-or-err (alistp (m-anno), 'not-alist, m-anno),
                      'module-annotation)
    else t endif

```

EVENT: Enable module-annotation-ok; name this event 'module-annotation-ok-off'.

DEFINITION:

```

module-syntax-errors (module, netlist)
=  list (module-form-ok (module, length (module)),
        module-name-ok (module-name (module), length (module), netlist),
        module-inputs-ok (module-inputs (module), length (module)),
        module-outputs-ok (module-outputs (module), length (module)),
        module-occurrences-ok (module-occurrences (module),
                                                    module-inputs (module),

```

```

module-outputs (module),
module-statenames (module),
length (module),
netlist),
module-statenames-ok (module-statenames (module), length (module)),
module-annotation-ok (module-annotation (module), length (module)))

```

EVENT: Enable module-syntax-errors; name this event ‘module-syntax-errors-off’.

DEFINITION:

```

module-syntax-ok (module, netlist)
= err-and (module-syntax-errors (module, netlist),
          list ('module', module-name (module)))

```

EVENT: Enable module-syntax-ok; name this event ‘module-syntax-ok-off’.

DEFINITION:

```

netlist-syntax-errors (netlist)
= if listp (netlist)
  then cons (module-syntax-ok (car (netlist), cdr (netlist)),
            netlist-syntax-errors (cdr (netlist)))
  else list (nil-or-err (netlist, 'not-proper-list')) endif

```

EVENT: Enable netlist-syntax-errors; name this event ‘netlist-syntax-errors-off’.

DEFINITION:

```

netlist-syntax-ok (netlist)
= err-and (netlist-syntax-errors (netlist), 'netlist-syntax-errors)

```

EVENT: Enable netlist-syntax-ok; name this event ‘netlist-syntax-ok-off’.

DEFINITION:

```

letterp (char) = (char ∈ unpack ('abcdefghijklmnopqrstuvwxyz))

```

EVENT: Enable letterp; name this event ‘letterp-off’.

DEFINITION:

```

digitp (char) = (char ∈ cdr (unpack ('a0123456789)))

```

EVENT: Enable digitp; name this event ‘digitp-off’.

DEFINITION:

```
bad-lsi-name-chars (char-list, special-characters)
=  if listp (char-list)
    then if letterp (car (char-list))
        ∨ digitp (car (char-list))
        ∨ (car (char-list) ∈ special-characters)
    then bad-lsi-name-chars (cdr (char-list), special-characters)
    else insert (car (char-list),
                bad-lsi-name-chars (cdr (char-list),
                                     special-characters)) endif
    else nil endif
```

EVENT: Enable bad-lsi-name-chars; name this event 'bad-lsi-name-chars-off'.

DEFINITION:

LSI-KEYWORDS

```
=  '(action angle annotations bloat block bus cell celstat
    checksum clk_line direction dummy end flip function
    hm_bidi hm_input hm_output library name nc net pin
    powercells priority psecured region scale technology
    tsecured via wire)
```

EVENT: Enable lsi-keywords; name this event 'lsi-keywords-off'.

EVENT: Enable *1*lsi-keywords; name this event 'g*1*lsi-keywords-off'.

DEFINITION:

```
lsi-name-ok (name, token-size)
=  if name ∈ LSI-KEYWORDS
    then pred-error ('lsi-name-is-keyword, name)
    elseif litatom (name)
    then err-and (list (t-or-err (if token-size < length (unpack (name))
                                then f
                                else t endif,
                                'name-too-long,
                                list (list ('name-length,
                                             length (unpack (name))),
                                             list ('max-allowed-length,
                                                  token-size))),
                                t-or-err (letterp (car (unpack (name))),
                                             'name-begins-with-nonletter,
                                             car (unpack (name))),
                                nlistp-or-err (bad-lsi-name-chars (cdr (unpack (name))),
```

```

                                cdr (unpack ('a-)),
                                'illegal-chars-in-name),
                                t-or-err (last-cdr (unpack (name)) = 0,
                                'not-standard-litatom,
                                list ('last-cdr, last-cdr (unpack (name))))),
                                list ('bad-lsi-name, name))
    else pred-error ('lsi-name-not-litatom, name) endif

```

EVENT: Enable lsi-name-ok; name this event 'lsi-name-ok-off'.

DEFINITION:

```

lsi-bad-names (name-list, token-size)
=  if listp (name-list)
    then cons (lsi-name-ok (car (name-list), token-size),
               lsi-bad-names (cdr (name-list), token-size))
    else nil endif

```

EVENT: Enable lsi-bad-names; name this event 'lsi-bad-names-off'.

DEFINITION:

```

lsi-name-list-errors (name-list, token-size, duplicates-ok?)
=  list (err-and (lsi-bad-names (name-list, token-size), 'bad-lsi-names),
         if duplicates-ok? then t
         else no-duplicates-or-err (name-list, 'duplicates) endif,
         nil-or-err (last-cdr (name-list), 'not-proper-list))

```

EVENT: Enable lsi-name-list-errors; name this event 'lsi-name-list-errors-off'.

DEFINITION:

```

lsi-name-list-ok (name-list, token-size, label, duplicates-ok?)
=  err-and (lsi-name-list-errors (name-list, token-size, duplicates-ok?),
            label)

```

EVENT: Enable lsi-name-list-ok; name this event 'lsi-name-list-ok-off'.

DEFINITION:

```

lsi-function-name (fname)
=  if primp (fname) ^ primp-lookup (fname, 'lsi-name)
    then if listp (primp2 (fname, 'lsi-name))
         then car (primp2 (fname, 'lsi-name))
         else primp2 (fname, 'lsi-name) endif
    else fname endif

```

EVENT: Enable lsi-function-name; name this event 'lsi-function-name-off'.

DEFINITION: $\text{name-length}(name) = \text{length}(\text{unpack}(name))$

EVENT: Enable name-length; name this event 'name-length-off'.

DEFINITION:

```
hierarchical-name-max(n1, n2)
=  if (n1  $\simeq$  nil)  $\wedge$  (n2  $\simeq$  nil) then 0
    elseif n1  $\simeq$  nil then n2
    elseif n2  $\simeq$  nil then n1
    elseif car(n1) < car(n2) then n2
    else n1 endif
```

EVENT: Enable hierarchical-name-max; name this event 'hierarchical-name-max-off'.

DEFINITION:

```
max-hierarchical-length-and-name(names)
=  if listp(names)
    then hierarchical-name-max(list(1 + name-length(car(names)),
                                     car(names)),
                               max-hierarchical-length-and-name(cdr(names)))
    else 0 endif
```

EVENT: Enable max-hierarchical-length-and-name; name this event 'max-hierarchical-length-and-name-off'.

DEFINITION:

```
max-occ-hierarchical-name(occ, hierarchical-table, hierarchical-size)
=  if listp(if primp(occ-function(occ))
    then max-hierarchical-length-and-name(append(occ-inputs(occ),
                                                    occ-outputs(occ)))
    else value(occ-function(occ), hierarchical-table) endif)
  then if if hierarchical-size < ((1 + name-length(occ-name(occ)))
    + car(if primp(occ-function(occ))
    then max-hierarchical-length-and-name(append(occ-inputs(occ),
                                                    occ-outputs(occ)))
    else value(occ-function(occ),
                hierarchical-table) endif))
    then f
    else t endif
```

```

then cons((1 + name-length(occ-name(occ)))
+ car(if primp(occ-function(occ))
then max-hierarchical-length-and-name(append(occ-inputs(occ),
occ-outputs(occ)))

else value(occ-function(occ),
hierarchical-table) endif),
cons(occ-name(occ),
cdr(if primp(occ-function(occ))
then max-hierarchical-length-and-name(append(occ-inputs(occ),
occ-outputs(occ)))

else value(occ-function(occ),
hierarchical-table) endif)))
else pred-error('hierarchical-name-too-long,
list(list('name-length,
(1 + name-length(occ-name(occ)))
+ car(if primp(occ-function(occ))
then max-hierarchical-length-and-name(append(occ-inputs(occ),
occ-outputs(occ)))

else value(occ-function(occ),
hierarchical-table) endif)),
list('max-allowed-length,
hierarchical-size),
list('name,
cons(occ-name(occ),
cdr(if primp(occ-function(occ))
then max-hierarchical-length-and-name(append(occ-inputs(occ),
occ-outputs(occ)))

else value(occ-function(occ),
hierarchical-table) endif)))) endif

else 0 endif

```

EVENT: Enable max-occ-hierarchical-name; name this event 'max-occ-hierarchical-name-off'.

DEFINITION:

lsi-occ-name-ok(*o-name*, *o-length*, *o-outs*, *occ-data*, *token-size*)

```

= if o-length < 1 then f
else t endif
then err-and(list(lsi-name-ok(o-name, token-size),
t-or-err(o-name ∉ value('signals, occ-data),
'occ-name-is-signal,
o-name),
t-or-err((cdr(o-outs) = nil)
∨ (o-name ∉ o-outs),

```



```

                                'occ-name-is-same-as-output-name,
                                o-name)),
                                'occurrence-name)
else t endif

```

EVENT: Enable lsi-occ-name-ok; name this event 'lsi-occ-name-ok-off'.

DEFINITION:

```

lsi-occ-outputs-ok (o-outs, o-length, occ-data, token-size)
=  if if o-length < 2 then f
    else t endif
    then err-and (list (lsi-name-list-ok (o-outs,
                                          token-size,
                                          'occ-outputs,
                                          f),
                        disjoint-or-err (o-outs,
                                         value ('occ-names, occ-data),
                                         'occ-outputs-in-occ-names)),
                'occurrence-outputs)
    else t endif

```

EVENT: Enable lsi-occ-outputs-ok; name this event 'lsi-occ-outputs-ok-off'.

DEFINITION:

```

lsi-occ-inputs-ok (o-ins, o-length, token-size)
=  if if o-length < 4 then f
    else t endif
    then lsi-name-list-ok (o-ins, token-size, 'occurrence-inputs, t)
    else t endif

```

EVENT: Enable lsi-occ-inputs-ok; name this event 'lsi-occ-inputs-ok-off'.

DEFINITION:

```

lsi-occ-syntax-ok (occ, hierarchical-name-error, occ-data, netlist, token-size)
=  err-and (list (occ-form-ok (occ, length (occ)),
                  lsi-occ-name-ok (occ-name (occ),
                                   length (occ),
                                   occ-outputs (occ),
                                   occ-data,
                                   token-size),
                  lsi-occ-outputs-ok (occ-outputs (occ),
                                       length (occ),
                                       occ-data,

```

```

                                token-size),
lsi-occ-inputs-ok (occ-inputs (occ), length (occ), token-size),
occ-function-ok (occ-function (occ),
                    occ-inputs (occ),
                    occ-outputs (occ),
                    length (occ),
                    netlist),
occ-annotation-ok (occ-annotation (occ), length (occ),
                    hierarchical-name-error),
list ('occurrence, occ-name (occ)))

```

EVENT: Enable lsi-occ-syntax-ok; name this event 'lsi-occ-syntax-ok-off'.

DEFINITION:

```

initial-lsi-occ-syntax-data (m-ins, m-outs)
=  cons ('occ-fns,
        cons ('hierarchical-name . 0),
        initial-occ-syntax-data (m-ins, m-outs)))

```

EVENT: Enable initial-lsi-occ-syntax-data; name this event 'initial-lsi-occ-syntax-data-off'.

DEFINITION:

```

update-lsi-occ-syntax-data (occ, o-hname, occ-data)
=  cons (cons ('occ-fns,
              cons (lsi-function-name (occ-function (occ)),
                    value ('occ-fns, occ-data))),
        cons (cons ('hierarchical-name,
                    hierarchical-name-max (value ('hierarchical-name,
                                                    occ-data),
                                                    o-hname)),
              update-occ-syntax-data (occ, occ-data)))

```

EVENT: Enable update-lsi-occ-syntax-data; name this event 'update-lsi-occ-syntax-data-off'.

DEFINITION:

```

lsi-occ-body-syntax-check (body,
                           occ-data,
                           outputs,
                           states,
                           netlist,
                           token-size,

```

```

                                hierarchical-table,
                                hierarchical-size)
=  if body  $\simeq$  nil
    then append (composite-occ-body-syntax-errors (body,
                                                    occ-data,
                                                    outputs,
                                                    states),
                list (alist-entry ('hierarchical-name, occ-data),
                      disjoint-or-err (value ('signals, occ-data),
                                         value ('occ-fns, occ-data),
                                         'signal-occ-fn-overlap),
                      disjoint-or-err (value ('occ-names, occ-data),
                                         value ('occ-fns, occ-data),
                                         'occ-name-occ-fn-overlap)))
    else cons (lsi-occ-syntax-ok (car (body),
                                   max-occ-hierarchical-name (car (body),
                                                                hierarchical-table,
                                                                hierarchical-size),
                                   occ-data,
                                   netlist,
                                   token-size),
              lsi-occ-body-syntax-check (cdr (body),
                                         update-lsi-occ-syntax-data (car (body),
                                                                       max-occ-hierarchical-name (car (body),
                                                                 hierarchical-
                                                                 hierarchical-
                                                                 occ-data),
                                                                       outputs,
                                                                       states,
                                                                       netlist,
                                                                       token-size,
                                                                       hierarchical-table,
                                                                       hierarchical-size)) endif

```

EVENT: Enable lsi-occ-body-syntax-check; name this event 'lsi-occ-body-syntax-check-off'.

DEFINITION:

```

lsi-module-name-ok (m-name, m-length, token-size, netlist)
=  if if m-length < 1 then f
    else t endif
    then err-and (list (lsi-name-ok (m-name, token-size),
                                   t-or-err ( $\neg$  lookup-module (m-name, netlist),
                                   'duplicate-module-defns,

```

```

                                m-name),
t-or-err (⊃ primp (m-name),
          'module-name-same-as-primitive,
          m-name)),
'module-name)
else t endif

```

EVENT: Enable lsi-module-name-ok; name this event 'lsi-module-name-ok-off'.

DEFINITION:

```

lsi-module-inputs-ok (m-ins, m-length, token-size)
=  if if m-length < 2 then f
    else t endif
    then lsi-name-list-ok (m-ins, token-size, 'module-inputs, f)
    else t endif

```

EVENT: Enable lsi-module-inputs-ok; name this event 'lsi-module-inputs-ok-off'.

DEFINITION:

```

lsi-module-outputs-ok (m-outs, m-length, token-size)
=  if if m-length < 3 then f
    else t endif
    then lsi-name-list-ok (m-outs, token-size, 'module-outputs, f)
    else t endif

```

EVENT: Enable lsi-module-outputs-ok; name this event 'lsi-module-outputs-ok-off'.

DEFINITION:

```

lsi-module-occurrences-check (m-occs,
                               m-ins,
                               m-outs,
                               m-states,
                               m-length,
                               netlist,
                               token-size,
                               hierarchical-table,
                               hierarchical-size)
=  if if m-length < 4 then f
    else t endif
    then cons (t-or-err (listp (m-occs), 'no-occurrences, m-occs),
               lsi-occ-body-syntax-check (m-occs,

```

```

initial-lsi-occ-syntax-data (m-ins,
                             m-outs),
m-outs,
states-list-or-nil (m-states,
                   m-length),
netlist,
token-size,
hierarchical-table,
hierarchical-size))

else t endif

```

EVENT: Enable lsi-module-occurrences-check; name this event 'lsi-module-occurrences-check-off'.

DEFINITION:

```

lsi-module-syntax-check (module,
                        netlist,
                        token-size,
                        hierarchical-table,
                        hierarchical-size)
= list (cons (module-name (module),
                        value ('hierarchical-name,
                            lsi-module-occurrences-check (module-occurrences (module),
                                module-inputs (module),
                                module-outputs (module),
                                module-statenames (module),
                                length (module),
                                netlist,
                                token-size,
                                hierarchical-table,
                                hierarchical-size))),
            err-and (list (module-form-ok (module, length (module)),
                lsi-module-name-ok (module-name (module),
                    length (module),
                    token-size,
                    netlist),
                lsi-module-inputs-ok (module-inputs (module),
                    length (module),
                    token-size),
                lsi-module-outputs-ok (module-outputs (module),
                    length (module),
                    token-size),
                err-and (lsi-module-occurrences-check (module-occurrences (module),
                    module-inputs (module),

```

```

module-outputs(module),
module-statenames(module),
length(module),
netlist,
token-size,
hierarchical-table,
hierarchical-size),
'module-occurrences'),
module-statenames-ok (module-statenames(module),
length(module)),
module-annotation-ok (module-annotation(module),
length(module))),
list('module, module-name(module)))

```

EVENT: Enable lsi-module-syntax-check; name this event 'lsi-module-syntax-check-off'.

DEFINITION:

```

lsi-netlist-syntax-check(netlist, token-size, hierarchical-size)
=  if listp(netlist)
    then append (lsi-module-syntax-check (car(netlist),
                                           cdr(netlist),
                                           token-size,
                                           lsi-netlist-syntax-check (cdr(netlist),
                                                                           token-size,
                                                                           hierarchical-size),
                                                                           hierarchical-size),
                                           lsi-netlist-syntax-check (cdr(netlist),
                                                                           token-size,
                                                                           hierarchical-size))
    else list (nil-or-err(netlist, 'not-proper-list)) endif

```

EVENT: Enable lsi-netlist-syntax-check; name this event 'lsi-netlist-syntax-check-off'.

DEFINITION:

```

lsi-netlist-syntax-ok(netlist, token-size, hierarchical-size)
=  err-and (lsi-netlist-syntax-check(netlist, token-size, hierarchical-size),
           'lsi-netlist-syntax-errors)

```

EVENT: Enable lsi-netlist-syntax-ok; name this event 'lsi-netlist-syntax-ok-off'.

DEFINITION:

PRIMITIVE-PROPERTIES

```
= '(delays drives input-types inputs loadings lsi-name
    new-states out-depends output-types outputs results
    state-types states gates pads primitives transistors)
```

EVENT: Enable primitive-properties; name this event 'primitive-properties-off'.

EVENT: Enable *1*primitive-properties; name this event 'g*1*primitive-properties-off'.

DEFINITION:

ALL-MODULE-PROPS

```
= '(delays drives input-types inputs loadings out-depends
    output-types outputs state-types states gates pads
    primitives transistors)
```

EVENT: Enable all-module-props; name this event 'all-module-props-off'.

EVENT: Enable *1*all-module-props; name this event 'g*1*all-module-props-off'.

EVENT: Add the shell *unknown*, with recognizer function symbol *unknownp* and no accessors.

DEFINITION:

value-or-unknown(*key*, *alist*)

```
= if boundp(key, alist) then value(key, alist)
   else UNKNOWN endif
```

EVENT: Enable value-or-unknown; name this event 'value-or-unknown-off'.

DEFINITION:

collect-value-or-unknown(*keys*, *alist*)

```
= if listp(keys)
   then cons(value-or-unknown(car(keys), alist),
             collect-value-or-unknown(cdr(keys), alist))
   else nil endif
```

EVENT: Enable collect-value-or-unknown; name this event 'collect-value-or-unknown-off'.

EVENT: Add the shell *io-out*, with recognizer function symbol *io-outp* and 1 accessor: *io-out-signal*, with type restriction (none-of) and default value false.

DEFINITION:

```
mark-io-out(name, io-outs)  
=  if name ∈ io-outs then io-out(name)  
    else name endif
```

EVENT: Enable mark-io-out; name this event ‘mark-io-out-off’.

DEFINITION:

```
mark-io-outs-0(names, io-outs)  
=  if listp(names)  
    then cons(mark-io-out(car(names), io-outs),  
              mark-io-outs-0(cdr(names), io-outs))  
    else nil endif
```

EVENT: Enable mark-io-outs-0; name this event ‘mark-io-outs-0-off’.

DEFINITION:

```
mark-io-outs(names, io-outs)  
=  if disjoint(io-outs, names) then names  
    else mark-io-outs-0(names, io-outs) endif
```

EVENT: Enable mark-io-outs; name this event ‘mark-io-outs-off’.

DEFINITION:

```
unmark-io-out(name)  
=  if io-outp(name) then io-out-signal(name)  
    else name endif
```

EVENT: Enable unmark-io-out; name this event ‘unmark-io-out-off’.

DEFINITION:

```
unmark-io-outs(names)  
=  if listp(names)  
    then if (unmark-io-out(car(names)) = car(names))  
            ∧ (unmark-io-outs(cdr(names)) = cdr(names)) then names  
            else cons(unmark-io-out(car(names)),  
                      unmark-io-outs(cdr(names))) endif  
    else names endif
```

EVENT: Enable unmark-io-outs; name this event ‘unmark-io-outs-off’.

DEFINITION:

```
value2(s, in-map, out-map)  
=  if boundp(s, in-map) then value(s, in-map)  
    else value(unmark-io-out(s), out-map) endif
```

EVENT: Enable value2; name this event 'value2-off'.

DEFINITION:

```
collect-value2(lst, in-map, out-map)  
=  if listp(lst)  
    then cons(value2(car(lst), in-map, out-map),  
              collect-value2(cdr(lst), in-map, out-map))  
    else nil endif
```

EVENT: Enable collect-value2; name this event 'collect-value2-off'.

DEFINITION:

```
signal-namep(x) = (litatom(x)  $\vee$  indexp(x)  $\vee$  io-outp(x))
```

EVENT: Enable signal-namep; name this event 'signal-namep-off'.

DEFINITION:

```
parent-synonym0(name, slist, used-names)  
=  if name  $\in$  used-names then name  
    elseif boundp(name, slist)  
        then if signal-namep(value(name, slist))  
            then parent-synonym0(value(name, slist),  
                                   slist,  
                                   cons(name, used-names))  
            else name endif  
    else name endif
```

EVENT: Enable parent-synonym0; name this event 'parent-synonym0-off'.

DEFINITION:

```
parent-synonym(name, slist) = parent-synonym0(name, slist, nil)
```

EVENT: Enable parent-synonym; name this event 'parent-synonym-off'.

DEFINITION:

```
parent-synonyms-list(lst, slist)  
=  if listp(lst)
```

```

then cons (parent-synonym (car (lst), slist),
            parent-synonyms-list (cdr (lst), slist))
else lst endif

```

EVENT: Enable parent-synonyms-list; name this event ‘parent-synonyms-list-off’.

DEFINITION:

```

extract-names (lst)
= if lst  $\simeq$  nil then nil
  elseif signal-namep (car (lst))
  then insert (car (lst), extract-names (cdr (lst)))
  else extract-names (cdr (lst)) endif

```

EVENT: Enable extract-names; name this event ‘extract-names-off’.

DEFINITION:

```

externalize-parents (names, alist, inputs, outputs)
= if listp (names)
  then externalize-parents (cdr (names),
                           if (parent-synonym (car (names), alist)
                                $\in$  inputs)
                                $\vee$  (parent-synonym (car (names), alist)
                                    $\in$  outputs) then alist
                           else cons (cons (parent-synonym (car (names),
                                                           alist),
                                                           car (names)),
                                       cons (cons (car (names),
                                                   value-or-unknown (parent-synonym (car (names),
                                                                 alist),
                                                                 alist),
                                                   alist)),
                                       unbind (car (names),
                                                unbind (parent-synonym (car (names),
                                                                 alist),
                                                                 alist))),
                                       inputs,
                                       outputs)) endif,
  else alist endif

```

EVENT: Enable externalize-parents; name this event ‘externalize-parents-off’.

DEFINITION:

```

subtype (x, y)

```

```

=  if  $y = \text{'free'}$  then t
    elseif  $x = \text{'boolp'}$ 
        then  $y \in \text{'(boolp tri-state ttl-tri-state ttl)}$ 
        elseif  $x = \text{'ttl-tri-state'}$ 
            then  $y \in \text{'(tri-state ttl-tri-state)}$ 
            else  $x = y$  endif

```

EVENT: Enable subtype; name this event 'subtype-off'.

DEFINITION:

```

types-compatiblep( $t1, t2$ )
=  if  $t1 = t2$  then t
    elseif  $(t1 = \text{'free'}) \vee (t2 = \text{'free'})$  then t
    elseif  $t1 = \text{'tri-state'}$  then  $t2 = \text{'ttl-tri-state'}$ 
    elseif  $t2 = \text{'tri-state'}$  then  $t1 = \text{'ttl-tri-state'}$ 
    else f endif

```

EVENT: Enable types-compatiblep; name this event 'types-compatiblep-off'.

DEFINITION:

```

tri-state-typep( $type$ ) =  $(type \in \text{'(tri-state ttl-tri-state)})$ 

```

EVENT: Enable tri-state-typep; name this event 'tri-state-typep-off'.

DEFINITION:

```

type-value0( $n, alist, free, used-names$ )
=  if  $(n \in used-names) \vee (\neg \text{boundp}(n, alist))$ 
    then if  $free$  then 'free'
        else list( $n$ ) endif
    elseif listp(value( $n, alist$ ))
        then type-value0(car(value( $n, alist$ )),  $alist, free, \text{cons}(n, used-names)$ )
        else value( $n, alist$ ) endif

```

EVENT: Enable type-value0; name this event 'type-value0-off'.

DEFINITION:

```

type-value( $n, alist, free$ ) = type-value0( $n, alist, free, \text{nil}$ )

```

EVENT: Enable type-value; name this event 'type-value-off'.

DEFINITION:

```

input-type( $name, alist$ ) = type-value(unmark-io-out( $name$ ),  $alist, \text{t}$ )

```

EVENT: Enable input-type; name this event 'input-type-off'.

DEFINITION:

```
output-type(name, out-map, in-map)
=  if listp(type-value(name, out-map, f))
    then type-value(car(type-value(name, out-map, f)), in-map, f)
    else type-value(name, out-map, f) endif
```

EVENT: Enable output-type; name this event 'output-type-off'.

DEFINITION: INITIAL-OCC-IN-TYPES = **nil**

DEFINITION: INITIAL-OCC-OUT-TYPES = **nil**

DEFINITION:

```
add-in-type(name, type, type-map)
=  if type = 'free then type-map
    elseif boundp(unmark-io-out(name), type-map)
    then if unknownp(value(unmark-io-out(name), type-map))
        ∨ subtype(value(unmark-io-out(name), type-map), type)
        then type-map
        elseif unknownp(type)
        ∨ subtype(type, value(unmark-io-out(name), type-map))
        then cons(cons(unmark-io-out(name), type),
                    unbind(unmark-io-out(name), type-map))
        else cons(pred-error(list('signal, unmark-io-out(name)),
                                list(list('old-type,
                                             value(unmark-io-out(name),
                                             type-map)),
                                list('new-type, type))),
                    cons(cons(unmark-io-out(name), UNKNOWN),
                        unbind(unmark-io-out(name), type-map))) endif
    else cons(cons(unmark-io-out(name), type), type-map) endif
```

EVENT: Enable add-in-type; name this event 'add-in-type-off'.

DEFINITION:

```
add-in-types(inputs, types, type-map)
=  if listp(inputs)
    then add-in-types(cdr(inputs),
                      cdr(types),
                      add-in-type(car(inputs), car(types), type-map))
    else type-map endif
```

EVENT: Enable add-in-types; name this event ‘add-in-types-off’.

DEFINITION:

```
update-in-types(in-types, o-ins, o-fn-props)  
= add-in-types(o-ins, value('input-types, o-fn-props), in-types)
```

EVENT: Enable update-in-types; name this event ‘update-in-types-off’.

DEFINITION:

```
add-out-type(name, type, arg-map, type-map)  
= cons(cons(name,  
            if listp(type) then collect-value(type, arg-map)  
            else type endif),  
       type-map)
```

EVENT: Enable add-out-type; name this event ‘add-out-type-off’.

DEFINITION:

```
add-out-types(outputs, types, arg-map, type-map)  
= if listp(outputs)  
  then add-out-types(cdr(outputs),  
                     cdr(types),  
                     arg-map,  
                     add-out-type(car(outputs),  
                                   car(types),  
                                   arg-map,  
                                   type-map))  
  else type-map endif
```

EVENT: Enable add-out-types; name this event ‘add-out-types-off’.

DEFINITION:

```
update-out-types(out-types, o-outs, o-fn-props, in-map)  
= add-out-types(o-outs,  
                value('output-types, o-fn-props),  
                in-map,  
                out-types)
```

EVENT: Enable update-out-types; name this event ‘update-out-types-off’.

DEFINITION:

```
io-type-error(name, in-type, out-type)
```

```
=  pred-error (list (if io-outp (name) then 'io-signal
                      else 'signal endif,
                      unmark-io-out (name)),
                  list (list ('input-type, in-type),
                          list ('output-type, out-type)))
```

EVENT: Enable io-type-error; name this event 'io-type-error-off'.

DEFINITION:

```
io-types-compatible (name, in-type, out-type, t-wire-ins)
=  if unknownp (in-type) then unknownp (out-type)
    elseif unknownp (out-type) then unknownp (in-type)
    elseif io-outp (name)
    then if tri-state-typep (out-type) ∧ subtype (out-type, in-type)
         then t
         else io-type-error (name, in-type, out-type) endif
    elseif name ∈ t-wire-ins
    then if tri-state-typep (in-type)
         ∧ types-compatiblep (in-type, out-type) then t
         else io-type-error (name, in-type, out-type) endif
    elseif subtype (out-type, in-type) then t
    else io-type-error (name, in-type, out-type) endif
```

EVENT: Enable io-types-compatible; name this event 'io-types-compatible-off'.

DEFINITION:

```
ok-in-type (ok, name, out-type, in-map)
=  if (¬ ok) ∨ net-errorp (ok)
    then set-value (unmark-io-out (name), UNKNOWN, in-map)
    elseif io-outp (name)
    then set-value (unmark-io-out (name), out-type, in-map)
    else in-map endif
```

EVENT: Enable ok-in-type; name this event 'ok-in-type-off'.

DEFINITION:

```
ok-out-type (ok, name, out-map)
=  if net-errorp (ok) then cons (ok, set-value (name, UNKNOWN, out-map))
    elseif ok then out-map
    else set-value (name, UNKNOWN, out-map) endif
```

EVENT: Enable ok-out-type; name this event 'ok-out-type-off'.

DEFINITION:

```
transfer-in-type (name, out-type, in-map)
=  if listp (out-type)
    then add-in-type (car (out-type), input-type (name, in-map), in-map)
    else in-map endif
```

EVENT: Enable transfer-in-type; name this event ‘transfer-in-type-off’.

DEFINITION:

```
compose-type (name, type, maps, t-wire-ins)
=  cons (ok-in-type (io-types-compatible (name,
                                           input-type (name, car (maps)),
                                           if listp (if listp (type)
                                               then output-type (car (type),
                                                                    cdr (maps),
                                                                    car (maps))
                                               else type endif)
                                           then 'free
                                           elseif listp (type)
                                           then output-type (car (type),
                                                                    cdr (maps),
                                                                    car (maps))
                                           else type endif,
                                           t-wire-ins),
        name,
        if listp (type)
        then output-type (car (type), cdr (maps), car (maps))
        else type endif,
        car (maps)),
    ok-out-type (io-types-compatible (name,
                                       input-type (name, car (maps)),
                                       if listp (if listp (type)
                                           then output-type (car (type),
                                                                cdr (maps),
                                                                car (maps))
                                           else type endif)
                                       then 'free
                                       elseif listp (type)
                                       then output-type (car (type),
                                                                cdr (maps),
                                                                car (maps))
                                       else type endif,
                                       t-wire-ins),
        name,
```

$\text{cdr}(\text{maps}))$

EVENT: Enable compose-type; name this event ‘compose-type-off’.

DEFINITION:

```
composed-type-maps(out-types, in-map, out-map, t-wire-ins)
=  if listp(out-types)
    then compose-type(caar(out-types),
                      cdar(out-types),
                      composed-type-maps(cdr(out-types),
                                         transfer-in-type(caar(out-types),
                                                             cdar(out-types),
                                                             in-map),
                                         out-map,
                                         t-wire-ins),
                      t-wire-ins)
    else cons(in-map, out-map) endif
```

EVENT: Enable composed-type-maps; name this event ‘composed-type-maps-off’.

DEFINITION:

```
compose-io-types(in-map, out-map, t-wire-ins)
=  composed-type-maps(out-map, in-map, out-map, t-wire-ins)
```

EVENT: Enable compose-io-types; name this event ‘compose-io-types-off’.

DEFINITION:

```
collect-in-types(names, alist)
=  if listp(names)
    then cons(type-value(car(names), alist, t),
              collect-in-types(cdr(names), alist))
    else nil endif
```

EVENT: Enable collect-in-types; name this event ‘collect-in-types-off’.

DEFINITION:

```
collect-out-type(name, out-map, inputs, in-map)
=  if listp(output-type(name, out-map, in-map))
    ∧ (car(output-type(name, out-map, in-map)) ∉ inputs)
    then UNKNOWN
    else output-type(name, out-map, in-map) endif
```


EVENT: Enable collect-out-type; name this event ‘collect-out-type-off’.

DEFINITION:

```
collect-out-types(outputs, out-map, inputs, in-map)
=  if listp(outputs)
    then cons(collect-out-type(car(outputs), out-map, inputs, in-map),
              collect-out-types(cdr(outputs), out-map, inputs, in-map))
    else nil endif
```

EVENT: Enable collect-out-types; name this event ‘collect-out-types-off’.

DEFINITION:

```
in-types-error(in-map) = err-and(in-map, 'input-type-conflicts)
```

EVENT: Enable in-types-error; name this event ‘in-types-error-off’.

DEFINITION:

```
out-types-error(out-map) = err-and(out-map, 'io-type-conflicts)
```

EVENT: Enable out-types-error; name this event ‘out-types-error-off’.

DEFINITION:

```
map(dr)
=  case on length(dr):
    case = 2
    then (car(dr) ∈ N) ∧ (cadr(dr) = 'ma) ∧ (cddr(dr) = nil)
    case = 3
    then (car(dr) ∈ N)
        ∧ ((length(cadr(dr)) = 2)
            ∧ (caadr(dr) = 'point)
            ∧ (cadadr(dr) ∈ N))
        ∧ (caddr(dr) = 'ma)
        ∧ (cdddr(dr) = nil)
    otherwise f endcase
```

EVENT: Enable map; name this event ‘map-off’.

DEFINITION:

```
pf(ld)
=  if length(ld) = 2
    then (car(ld) ∈ N) ∧ (cadr(ld) = 'pf) ∧ (cddr(ld) = nil)
    else f endif
```

EVENT: Enable pfp; name this event ‘pfp-off’.

DEFINITION:

$\text{ma-to-pf}(x) = \text{cons}((\text{car}(x) * 10) + \text{cadadr}(x), '(\text{pf}))$

EVENT: Enable ma-to-pf; name this event ‘ma-to-pf-off’.

DEFINITION:

$\text{pf-to-ma}(x)$
= **if** $(\text{car}(x) \bmod 10) \simeq 0$ **then** $\text{cons}(\text{car}(x) \div 10, '(\text{ma}))$
 else $\text{cons}(\text{car}(x) \div 10,$
 $\text{cons}(\text{list}('point, \text{car}(x) \bmod 10), '(\text{ma})))$ **endif**

EVENT: Enable pf-to-ma; name this event ‘pf-to-ma-off’.

DEFINITION: $\text{std-load-to-pf}(x) = \text{cons}(x * 10, '(\text{pf}))$

EVENT: Enable std-load-to-pf; name this event ‘std-load-to-pf-off’.

DEFINITION:

$\text{pf-to-std-load}(x)$
= **if** $(\text{car}(x) \bmod 10) < 5$ **then** $\text{car}(x) \div 10$
 else $(\text{car}(x) \div 10) + 1$ **endif**

EVENT: Enable pf-to-std-load; name this event ‘pf-to-std-load-off’.

DEFINITION: $\text{ma-to-std-drive}(x) = \text{car}(x)$

EVENT: Enable ma-to-std-drive; name this event ‘ma-to-std-drive-off’.

DEFINITION: $\text{std-drive-to-ma}(x) = \text{cons}(x, '(\text{ma}))$

EVENT: Enable std-drive-to-ma; name this event ‘std-drive-to-ma-off’.

DEFINITION:

$\text{zero-loadingp}(a)$
= **if** $\text{pfp}(a)$ **then** $\text{car}(a) \simeq 0$
 else $a \simeq 0$ **endif**

EVENT: Enable zero-loadingp; name this event ‘zero-loadingp-off’.

DEFINITION: $\text{pf-plus}(a, b) = \text{cons}(\text{car}(a) + \text{car}(b), '(\text{pf}))$

EVENT: Enable pf-plus; name this event ‘pf-plus-off’.

DEFINITION:

$\text{pf-difference}(a, b) = \text{cons}(\text{car}(a) - \text{car}(b), '(\text{pf}))$

EVENT: Enable pf-difference; name this event ‘pf-difference-off’.

DEFINITION: $\text{pf-lessp}(a, b) = (\text{car}(a) < \text{car}(b))$

EVENT: Enable pf-lessp; name this event ‘pf-lessp-off’.

DEFINITION:

$\text{ma-lessp}(a, b)$

$=$ **if** $\text{car}(a) < \text{car}(b)$ **then** **t**
 elseif $\text{car}(a) = \text{car}(b)$ **then** $\text{cadadr}(a) < \text{cadadr}(b)$
 else **f** **endif**

EVENT: Enable ma-lessp; name this event ‘ma-lessp-off’.

DEFINITION:

$\text{loading-plus}(x, y)$

$=$ **if** $x \in \mathbf{N}$
 then **if** $y \in \mathbf{N}$ **then** $x + y$
 elseif $\text{pfp}(y)$ **then** $\text{pf-plus}(\text{std-load-to-pf}(x), y)$
 else x **endif**
 elseif $\text{pfp}(x)$
 then **if** $\text{pfp}(y)$ **then** $\text{pf-plus}(x, y)$
 elseif $y \in \mathbf{N}$ **then** $\text{pf-plus}(x, \text{std-load-to-pf}(y))$
 else x **endif**
 elseif $(y \in \mathbf{N}) \vee \text{pfp}(y)$ **then** y
 else 0 **endif**

EVENT: Enable loading-plus; name this event ‘loading-plus-off’.

DEFINITION: $\text{INITIAL-OCC-LOADINGS} = \mathbf{nil}$

DEFINITION: $\text{INITIAL-OCC-DRIVES} = \mathbf{nil}$

DEFINITION:

$\text{add-loading}(\text{name}, \text{loading}, \text{load-map})$

$=$ **if** $\text{zero-loadingp}(\text{loading})$ **then** load-map
 elseif $\text{boundp}(\text{name}, \text{load-map})$
 then $\text{cons}(\text{cons}(\text{name}, \text{loading-plus}(\text{loading}, \text{value}(\text{name}, \text{load-map}))),$

```

        unbind(name, load-map)
    else cons(cons(name, loading), load-map) endif

```

EVENT: Enable add-loading; name this event ‘add-loading-off’.

DEFINITION:

```

add-loadings(names, loadings, load-map)
=  if listp(names)
    then add-loadings(cdr(names),
                      cdr(loadings),
                      add-loading(car(names), car(loadings), load-map))
    else load-map endif

```

EVENT: Enable add-loadings; name this event ‘add-loadings-off’.

DEFINITION:

```

update-loadings(loadings, o-ins, o-fn-props)
=  add-loadings(o-ins, value('loadings, o-fn-props), loadings)

```

EVENT: Enable update-loadings; name this event ‘update-loadings-off’.

DEFINITION:

```

local-drive(drive, listp, in-map, out-map, drive-map)
=  if listp
    then if listp(drive)
        then cons(local-drive(car(drive), f, in-map, out-map, drive-map),
                  local-drive(cdr(drive), t, in-map, out-map, drive-map))
        else nil endif
    elseif signal-namep(drive)
        then parent-synonym(value2(drive, in-map, out-map), drive-map)
    elseif listp(drive) ∧ (car(drive) = 'min)
        then cons('min, local-drive(cdr(drive), t, in-map, out-map, drive-map))
    else drive endif

```

EVENT: Enable local-drive; name this event ‘local-drive-off’.

DEFINITION:

```

add-drives(outputs, drives, in-map, out-map, drive-map)
=  if listp(outputs)
    then add-drives(cdr(outputs),
                  cdr(drives),
                  in-map,
                  out-map,

```

```

cons (cons (car (outputs),
             local-drive (car (drives),
                             f,
                             in-map,
                             out-map,
                             drive-map)),
      drive-map))
else drive-map endif

```

EVENT: Enable add-drives; name this event ‘add-drives-off’.

DEFINITION:

```

update-drives (drives, o-outs, o-fn-props, in-map, out-map)
= add-drives (o-outs, value ('drives, o-fn-props), in-map, out-map, drives)

```

EVENT: Enable update-drives; name this event ‘update-drives-off’.

DEFINITION:

```

transfer-loading (drive, listp, load, loadings)
= if listp
  then if listp (drive)
    then transfer-loading (cdr (drive),
                          t,
                          load,
                          transfer-loading (car (drive),
                                              f,
                                              load,
                                              loadings))
    else loadings endif
  elseif signal-namep (drive) then add-loading (drive, load, loadings)
  elseif listp (drive)  $\wedge$  (car (drive) = 'min)
    then transfer-loading (cdr (drive), t, load, loadings)
  else loadings endif

```

EVENT: Enable transfer-loading; name this event ‘transfer-loading-off’.

DEFINITION:

```

transfer-loadings (loadings, drives)
= if listp (drives)
  then transfer-loadings (transfer-loading (cдар (drives),
                                             f,
                                             value (caar (drives), loadings),
                                             loadings),

```

```

                                cdr (drives))
    else loadings endif

```

EVENT: Enable transfer-loadings; name this event ‘transfer-loadings-off’.

DEFINITION:

literal-net-drive (drive, load, label)

```

=  if drive ∈ N
    then if drive < if load ∈ N then load
        elseif pfp (load) then pf-to-std-load (load)
        else 0 endif
    then pred-error (label,
                     list (list ('drive, drive),
                              cons ('loading,
                                    cons (load,
                                          if load = if load ∈ N then load
                                              elseif pfp (load)
                                              then pf-to-std-load (load)
                                              else 0 endif
                                          then nil
                                          else list (if load ∈ N
                                              then load
                                              elseif pfp (load)
                                              then pf-to-std-load (load)
                                              else 0 endif) endif)))
    else drive – if load ∈ N then load
        elseif pfp (load) then pf-to-std-load (load)
        else 0 endif endif
elseif map (drive)
then if pf-lessp (ma-to-pf (drive),
                 if pfp (load) then load
                 elseif load ∈ N then std-load-to-pf (load)
                 else '(0 pf) endif)
then pred-error (label,
                 list (list ('drive, drive, ma-to-pf (drive)),
                        cons ('loading,
                              cons (load,
                                    if load = if pfp (load)
                                        then load
                                        elseif load ∈ N
                                        then std-load-to-pf (load)
                                        else '(0 pf) endif
                                    then nil
                                    else list (if pfp (load)

```

```

                                then load
                                elseif load ∈ N
                                then std-load-to-pf (load)
                                else '(0
                                    pf) endif) endif))))
else pf-to-ma (pf-difference (ma-to-pf (drive),
                                if pfp (load) then load
                                elseif load ∈ N
                                then std-load-to-pf (load)
                                else '(0 pf) endif) endif
else UNKNOWN endif

```

EVENT: Enable literal-net-drive; name this event 'literal-net-drive-off'.

DEFINITION:

```

net-min-args (a, load, listp)
= if listp
  then if listp (a)
    then net-min-args (car (a), load, f)
      ∪ net-min-args (cdr (a), load, t)
    else nil endif
  elseif (a ∈ N) ∨ map (a)
  then list (literal-net-drive (a, load, 'min-arg))
  elseif listp (a) ∧ (car (a) = 'min)
  then net-min-args (cdr (a), load, t)
  else list (a) endif

```

EVENT: Enable net-min-args; name this event 'net-min-args-off'.

DEFINITION:

```

add-net-min-drive (name, args, load, drive-map)
= if net-errorp (err-and (net-min-args (args, load, t), list ('signal, name)))
  then if listp (extract-names (net-min-args (args, load, t)))
    then cons (err-and (net-min-args (args, load, t),
      list ('signal, name)),
      cons (cons (name,
        cons ('min,
          cons (UNKNOWN,
            extract-names (net-min-args (args,
              load,
              t))))),
        drive-map))
    else cons (err-and (net-min-args (args, load, t),

```

```

                                list('signal, name)),
                                drive-map) endif
else cons(cons(name, cons('min, net-min-args(args, load, t))),
                                drive-map) endif

```

EVENT: Enable add-net-min-drive; name this event 'add-net-min-drive-off'.

DEFINITION:

```

add-net-drive(drive, loadings, drive-map)
= if listp(drive)
  then if (cdr(drive) ∈ N) ∨ map(cdr(drive))
    then cons(if net-errorp(literal-net-drive(cdr(drive),
                                              value(car(drive),
                                              loadings),
                                              list('signal,
                                              car(drive))))
    then literal-net-drive(cdr(drive),
                          value(car(drive), loadings),
                          list('signal, car(drive)))
    else cons(car(drive),
              literal-net-drive(cdr(drive),
                                value(car(drive),
                                loadings),
                                list('signal,
                                car(drive)))) endif,
              drive-map)
  elseif cadr(drive) = 'min
    then add-net-min-drive(car(drive),
                           cddr(drive),
                           value(car(drive), loadings),
                           drive-map)
    else cons(drive, drive-map) endif
else cons(drive, drive-map) endif

```

EVENT: Enable add-net-drive; name this event 'add-net-drive-off'.

DEFINITION:

```

net-drives-simple(drs, lds)
= if listp(drs)
  then add-net-drive(car(drs), lds, net-drives-simple(cdr(drs), lds))
  else nil endif

```

EVENT: Enable net-drives-simple; name this event 'net-drives-simple-off'.

DEFINITION:

```
type-loading(load, type)
=  if type ∈ '(boolp clk level parametric tri-state)
    then if pfp(load) then pf-to-std-load(load)
        else load endif
    elseif type ∈ '(ttl ttl-tri-state)
    then if load ∈ N then std-load-to-pf(load)
        else load endif
    else load endif
```

EVENT: Enable type-loading; name this event 'type-loading-off'.

DEFINITION:

```
collect-loadings(names, types, loadings)
=  if listp(names)
    then cons(type-loading(value(car(names), loadings), car(types)),
              collect-loadings(cdr(names), cdr(types), loadings))
    else nil endif
```

EVENT: Enable collect-loadings; name this event 'collect-loadings-off'.

DEFINITION:

```
loadings-error(loadings) = err-and(loadings, 'bad-loadings)
```

EVENT: Enable loadings-error; name this event 'loadings-error-off'.

DEFINITION:

```
collect-min-args(a, listp, drives, used-names, inputs, outputs)
=  if listp
    then if listp(a)
        then collect-min-args(car(a), f, drives, used-names, inputs, outputs)
            ∪ collect-min-args(cdr(a),
                               t,
                               drives,
                               used-names,
                               inputs,
                               outputs)
        else nil endif
    elseif signal-namep(a)
    then if a ∈ inputs then list(a)
        elseif (¬ ((a ∈ outputs) ∨ (a ∈ used-names)))
            ∧ boundp(a, drives)
        then collect-min-args(value(a, drives),
```

```

                                f,
                                drives,
                                cons(a, used-names),
                                inputs,
                                outputs)
                                else list(UNKNOWN) endif
elseif listp(a) ∧ (car(a) = 'min)
then collect-min-args(cdr(a), t, drives, used-names, inputs, outputs)
elseif (a ∈ N) ∨ map(a) then list(a)
else list(UNKNOWN) endif

```

EVENT: Enable collect-min-args; name this event 'collect-min-args-off'.

DEFINITION:

```

numeric-drives(drs)
= if listp(drs)
  then if (car(drs) ∈ N) ∨ map(car(drs))
        then cons(car(drs), numeric-drives(cdr(drs)))
        else numeric-drives(cdr(drs)) endif
  else nil endif

```

EVENT: Enable numeric-drives; name this event 'numeric-drives-off'.

DEFINITION:

```

type-drive(drive, type)
= if type ∈ '(boolp clk level parametric tri-state)
  then if map(drive) then ma-to-std-drive(drive)
        else drive endif
  elseif type ∈ '(ttl ttl-tri-state)
  then if drive ∈ N then std-drive-to-ma(drive)
        else drive endif
  else drive endif

```

EVENT: Enable type-drive; name this event 'type-drive-off'.

DEFINITION:

```

drive-lessp(x, y)
= if (x ∈ N) ∧ (y ∈ N) then x < y
  elseif map(x) ∧ map(y) then ma-lessp(x, y)
  else f endif

```

EVENT: Enable drive-lessp; name this event 'drive-lessp-off'.

DEFINITION:

drive-min (*lst*, *type*)

```
=  if lst  $\simeq$  nil then UNKNOWN
    elseif cdr (lst)  $\simeq$  nil then type-drive (car (lst), type)
    elseif drive-lessp (type-drive (car (lst), type), drive-min (cdr (lst), type))
    then type-drive (car (lst), type)
    else drive-min (cdr (lst), type) endif
```

EVENT: Enable drive-min; name this event 'drive-min-off'.

DEFINITION:

make-drive-min (*args*, *type*, *drives*, *inputs*, *outputs*)

```
=  if extract-names (collect-min-args (args, t, drives, nil, inputs, outputs))  $\simeq$  nil
    then drive-min (numeric-drives (collect-min-args (args,
                                                         t,
                                                         drives,
                                                         nil,
                                                         inputs,
                                                         outputs))),
                    type)
    elseif numeric-drives (collect-min-args (args,
                                              t,
                                              drives,
                                              nil,
                                              inputs,
                                              outputs))  $\simeq$  nil
    then if listp (cdr (extract-names (collect-min-args (args,
                                                         t,
                                                         drives,
                                                         nil,
                                                         inputs,
                                                         outputs))))
        then cons ('min,
                    extract-names (collect-min-args (args,
                                                         t,
                                                         drives,
                                                         nil,
                                                         inputs,
                                                         outputs))))
        else cons ('min,
                    cons (UNKNOWN,
                          extract-names (collect-min-args (args,
                                                             t,
                                                             drives,
```

```

                                nil,
                                inputs,
                                outputs)))) endif

else cons('min,
          cons(drive-min (numeric-drives (collect-min-args (args,
                                                            t,
                                                            drives,
                                                            nil,
                                                            inputs,
                                                            outputs)),
                                                            type),
          extract-names (collect-min-args (args,
                                            t,
                                            drives,
                                            nil,
                                            inputs,
                                            outputs)))) endif

```

EVENT: Enable make-drive-min; name this event 'make-drive-min-off'.

DEFINITION:

```

collect-drives-0 (names, output-types, drives, inputs, outputs)
= if listp (names)
  then cons (if (value-or-unknown (car (names), drives) ∈ N)
    ∨ map (value-or-unknown (car (names), drives))
    then type-drive (value-or-unknown (car (names), drives),
                    car (output-types))
    elseif signal-namep (value-or-unknown (car (names), drives))
    then parent-synonym (value-or-unknown (car (names), drives),
                      drives)
    elseif listp (value-or-unknown (car (names), drives))
      ∧ (car (value-or-unknown (car (names), drives))
        = 'min)
    then make-drive-min (cdr (value-or-unknown (car (names), drives)),
                        car (output-types),
                        drives,
                        inputs,
                        outputs)
    else UNKNOWN endif,
  collect-drives-0 (cdr (names),
                    cdr (output-types),
                    drives,
                    inputs,
                    outputs))

```

else nil endif

EVENT: Enable collect-drives-0; name this event 'collect-drives-0-off'.

DEFINITION:

collect-drives(*outputs*, *output-types*, *drives*, *inputs*)
= collect-drives-0(*outputs*,
 output-types,
 externalize-parents(*outputs*, *drives*, *inputs*, *outputs*),
 inputs,
 outputs)

EVENT: Enable collect-drives; name this event 'collect-drives-off'.

DEFINITION:

drives-error(*drives*) = err-and(*drives*, 'bad-drives')

EVENT: Enable drives-error; name this event 'drives-error-off'.

DEFINITION:

ps-pfp(*x*)
= (properp(*x*)
 $\wedge$ (length(*x*) = 2)
 $\wedge$ (car(*x*) $\in$ **N**)
 $\wedge$ (cadr(*x*) = 'ps-pf'))

EVENT: Enable ps-pfp; name this event 'ps-pfp-off'.

DEFINITION:

delay-lh(*d*)
= **if** listp(*d*) $\wedge$ (car(*d*) $\neq$ 'or) **then** car(*d*)
 else f endif

EVENT: Enable delay-lh; name this event 'delay-lh-off'.

DEFINITION:

delay-hl(*d*)
= **if** listp(*d*) $\wedge$ (car(*d*) $\neq$ 'or) **then** cadr(*d*)
 else f endif

EVENT: Enable delay-hl; name this event 'delay-hl-off'.

DEFINITION:

$\text{delay-slope-lh}(d)$
= **if** $\text{listp}(d) \wedge (\text{car}(d) \neq \text{'or'})$ **then** $\text{caar}(d)$
 else f endif

EVENT: Enable delay-slope-lh; name this event 'delay-slope-lh-off'.

DEFINITION:

$\text{delay-slope-hl}(d)$
= **if** $\text{listp}(d) \wedge (\text{car}(d) \neq \text{'or'})$ **then** $\text{caadr}(d)$
 else f endif

EVENT: Enable delay-slope-hl; name this event 'delay-slope-hl-off'.

DEFINITION:

$\text{delay-intercept-lh}(d)$
= **if** $\text{listp}(d) \wedge (\text{car}(d) \neq \text{'or'})$ **then** $\text{cadar}(d)$
 else f endif

EVENT: Enable delay-intercept-lh; name this event 'delay-intercept-lh-off'.

DEFINITION:

$\text{delay-intercept-hl}(d)$
= **if** $\text{listp}(d) \wedge (\text{car}(d) \neq \text{'or'})$ **then** $\text{cadadr}(d)$
 else f endif

EVENT: Enable delay-intercept-hl; name this event 'delay-intercept-hl-off'.

DEFINITION:

$\text{delay-dependencies}(d)$
= **if** $\text{listp}(d) \wedge (\text{car}(d) \neq \text{'or'})$ **then** $\text{cddr}(d)$
 else nil endif

EVENT: Enable delay-dependencies; name this event 'delay-dependencies-off'.

DEFINITION: $\text{make-delay-0}(lh, hl, deps) = \text{cons}(lh, \text{cons}(hl, deps))$

EVENT: Enable make-delay-0; name this event 'make-delay-0-off'.

DEFINITION:

$\text{make-delay}(s-lh, i-lh, s-hl, i-hl, deps)$
= $\text{make-delay-0}(\text{list}(s-lh, i-lh), \text{list}(s-hl, i-hl), deps)$

EVENT: Enable make-delay; name this event ‘make-delay-off’.

DEFINITION:

determined-delayp(d)
= (listp(d) $\wedge$ (car(d) $\neq$ 'or) $\wedge$ (delay-dependencies(d) $\simeq$ nil))

EVENT: Enable determined-delayp; name this event ‘determined-delayp-off’.

DEFINITION:

make-range(lo , hi , $deps$)
= **if** ($lo = hi$) $\wedge$ ($deps \simeq$ nil) **then** lo
 else cons('range, cons(lo , cons(hi , $deps$))) **endif**

EVENT: Enable make-range; name this event ‘make-range-off’.

DEFINITION:

rangep(x)
= **if** listp(x) **then** car(x) = 'range
 else f endif

EVENT: Enable rangep; name this event ‘rangep-off’.

DEFINITION:

range-min(x)
= **if** rangep(x) **then** cadr(x)
 else x **endif**

EVENT: Enable range-min; name this event ‘range-min-off’.

DEFINITION:

range-max(x)
= **if** rangep(x) **then** caddr(x)
 else x **endif**

EVENT: Enable range-max; name this event ‘range-max-off’.

DEFINITION:

range-dependencies(x)
= **if** rangep(x) **then** cdddr(x)
 else nil **endif**

EVENT: Enable range-dependencies; name this event ‘range-dependencies-off’.

DEFINITION:

determined-range(x)

= **if** range(x) **then** range-dependencies(x) $\simeq$ **nil**
 else $x \in \mathbf{N}$ **endif**

EVENT: Enable determined-range; name this event ‘determined-range-off’.

DEFINITION:

per-std-load-delay-slope($slope$)

= **if** ps-pfp($slope$) **then** car($slope$) * 10
 else $slope$ **endif**

EVENT: Enable per-std-load-delay-slope; name this event ‘per-std-load-delay-slope-off’.

DEFINITION:

per-pf-delay-slope($slope$)

= **if** $slope \in \mathbf{N}$
 then cons(**if** ($slope \bmod 10$) < 5 **then** $slope \div 10$
 else $1 + (slope \div 10)$ **endif**,
 ’(ps-pf))
 else $slope$ **endif**

EVENT: Enable per-pf-delay-slope; name this event ‘per-pf-delay-slope-off’.

DEFINITION:

range-plus($i1$, $i2$, $deps$)

= make-range(range-min($i1$) + range-min($i2$),
 range-max($i1$) + range-max($i2$),
 $deps$)

EVENT: Enable range-plus; name this event ‘range-plus-off’.

DEFINITION:

slope-times-load($slope$, $load$)

= **if** $slope \in \mathbf{N}$
 then if $load \in \mathbf{N}$ **then** $slope * load$
 elseif pfp($load$) **then** $slope * \text{pf-to-std-load}(load)$
 else 0 **endif**
 elseif ps-pfp($slope$)
 then if pfp($load$) **then** car($slope$) * car($load$)
 elseif $load \in \mathbf{N}$ **then** car($slope$) * car(std-load-to-pf($load$))
 else 0 **endif**
 else 0 **endif**

EVENT: Enable slope-times-load; name this event ‘slope-times-load-off’.

DEFINITION: INITIAL-OCC-DELAYS = **nil**

THEOREM: local-delay-count-help-0

$\text{listp}(x) \rightarrow (\text{count}(x) = (1 + (\text{count}(\text{car}(x)) + \text{count}(\text{cdr}(x))))))$

THEOREM: local-delay-count-help

$\text{listp}(x) \rightarrow ((\text{count}(x) = 0) = \mathbf{f})$

EVENT: Enable local-delay-count-help-0; name this event ‘local-delay-count-help-0-off’.

EVENT: Disable range-dependencies; name this event ‘range-dependencies-on’.

EVENT: Disable delay-dependencies; name this event ‘delay-dependencies-on’.

DEFINITION:

```
local-delay(delay, out-depends, listp, in-map, out-map, delay-map)
= if listp
  then if listp(delay)
    then cons(local-delay(car(delay),
                          out-depends,
                          f,
                          in-map,
                          out-map,
                          delay-map),
              local-delay(cdr(delay),
                          out-depends,
                          t,
                          in-map,
                          out-map,
                          delay-map))
    else nil endif
  elseif signal-namep(delay)
    then parent-synonym(value2(delay, in-map, out-map), delay-map)
  elseif delay  $\simeq$  nil then delay
  elseif car(delay) = 'or'
    then cons('or', local-delay(cdr(delay), f, t, in-map, out-map, delay-map))
  elseif range(delay)
    then if listp(range-dependencies(delay))
      then make-range(range-min(delay),
```

```

range-max (delay),
local-delay (range-dependencies (delay),
             f,
             t,
             in-map,
             out-map,
             delay-map))

else delay endif
elseif listp (if listp (delay-dependencies (delay))
                 then delay-dependencies (delay)
                 else out-depends endif)
then make-delay-0 (delay-lh (delay),
                    delay-hl (delay),
                    local-delay (if listp (delay-dependencies (delay))
                                then delay-dependencies (delay)
                                else out-depends endif,
                                f,
                                t,
                                in-map,
                                out-map,
                                delay-map))

else delay endif

```

EVENT: Enable local-delay-count-help; name this event ‘local-delay-count-help-off’.

EVENT: Enable range-dependencies; name this event ‘range-dependencies-off1’.

EVENT: Enable delay-dependencies; name this event ‘delay-dependencies-off1’.

EVENT: Enable local-delay; name this event ‘local-delay-off’.

DEFINITION:

```

add-delays (outputs, delays, out-depends, in-map, out-map, delay-map)
= if listp (outputs)
  then add-delays (cdr (outputs),
                  cdr (delays),
                  cdr (out-depends),
                  in-map,
                  out-map,
                  cons (cons (car (outputs),
                             local-delay (car (delays),

```

```

                                car (out-depends),
                                f,
                                in-map,
                                out-map,
                                delay-map)),
                                delay-map))
    else delay-map endif

```

EVENT: Enable add-delays; name this event ‘add-delays-off’.

DEFINITION:

```

update-delays (delays, o-outs, o-fn-props, in-map, out-map)
=  add-delays (o-outs,
               value ('delays, o-fn-props),
               value ('out-depends, o-fn-props),
               in-map,
               out-map,
               delays)

```

EVENT: Enable update-delays; name this event ‘update-delays-off’.

DEFINITION:

```

add-load-delay (delay, load)
=  if zero-loadingp (load) then delay
    else make-delay (delay-slope-lh (delay),
                    range-plus (delay-intercept-lh (delay),
                                slope-times-load (delay-slope-lh (delay),
                                                    load),
                                nil),
                    delay-slope-hl (delay),
                    range-plus (delay-intercept-hl (delay),
                                slope-times-load (delay-slope-hl (delay),
                                                    load),
                                nil),
                    delay-dependencies (delay)) endif

```

EVENT: Enable add-load-delay; name this event ‘add-load-delay-off’.

DEFINITION:

```

loaded-delay (delay, load, listp)
=  if listp
    then if listp (delay)
         then cons (loaded-delay (car (delay), load, f),

```

```

                                loaded-delay (cdr (delay), load, t))
      else nil endif
    elseif delay  $\simeq$  nil then delay
    elseif car (delay) = 'or
    then cons ('or, loaded-delay (cdr (delay), load, t))
    else add-load-delay (delay, load) endif

```

EVENT: Enable loaded-delay; name this event 'loaded-delay-off'.

DEFINITION:

```

loaded-delays (delays, loadings)
= if listp (delays)
  then if car (delays)  $\simeq$  nil
        then cons (car (delays), loaded-delays (cdr (delays), loadings))
        else cons (cons (caar (delays),
                          loaded-delay (cdar (delays),
                                          value (caar (delays), loadings),
                                          f)),
                    loaded-delays (cdr (delays), loadings)) endif
  else delays endif

```

EVENT: Enable loaded-delays; name this event 'loaded-delays-off'.

DEFINITION:

```

merge-input-delays (r1, r2, or-args?)
= if rangep (r1)  $\vee$  (r1  $\in \mathbf{N}$ )
  then if rangep (r2)  $\vee$  (r2  $\in \mathbf{N}$ )
        then make-range (if or-args?
                           then min (range-min (r1), range-min (r2))
                           else max (range-min (r1), range-min (r2)) endif,
                           max (range-max (r1), range-max (r2)),
                           range-dependencies (r1)
                            $\cup$  range-dependencies (r2))
        else make-range (range-min (r1),
                          range-max (r1),
                          insert (r2, range-dependencies (r1))) endif
  elseif rangep (r2)  $\vee$  (r2  $\in \mathbf{N}$ )
  then make-range (range-min (r2),
                    range-max (r2),
                    insert (r1, range-dependencies (r2)))
  else make-range (0, 0, insert (r1, list (r2))) endif

```

EVENT: Enable merge-input-delays; name this event 'merge-input-delays-off'.

DEFINITION:

```
mergeable-input-delay-p(d, approx)
=  if approx then range-p(d)  $\vee$  (d  $\in$  N)  $\vee$  signal-namep(d)
    else determined-range-p(d) endif
```

EVENT: Enable mergeable-input-delay-p; name this event ‘mergeable-input-delay-p-off’.

DEFINITION:

```
insert-input-delay(d, dlist, or-args?, approx)
=  if unknownp(d) then dlist
    elseif dlist  $\simeq$  nil
    then if or-args?  $\wedge$  (car(d) = 'or) then cdr(d)
         else list(d) endif
    elseif or-args?  $\wedge$  listp(d)  $\wedge$  (car(d) = 'or)
    then insert-input-delay(cadr(d),
                           insert(car(dlist), cddr(d)  $\cup$  cdr(dlist)),
                           or-args?,
                           approx)
    elseif mergeable-input-delay-p(d, approx  $\wedge$  ( $\neg$  or-args?))
    then if mergeable-input-delay-p(car(dlist), approx  $\wedge$  ( $\neg$  or-args?))
         then cons(merge-input-delays(d, car(dlist), or-args?), cdr(dlist))
         else cons(d, dlist) endif
    elseif mergeable-input-delay-p(car(dlist), approx  $\wedge$  ( $\neg$  or-args?))
    then cons(car(dlist), insert(d, cdr(dlist)))
    else insert(d, dlist) endif
```

EVENT: Enable insert-input-delay; name this event ‘insert-input-delay-off’.

DEFINITION:

```
literal-delay-to-range(d)
=  make-range(min(range-min(delay-intercept-lh(d),
                    range-min(delay-intercept-hl(d))),
              max(range-max(delay-intercept-lh(d),
                    range-max(delay-intercept-hl(d))),
              delay-dependencies(d))
```

EVENT: Enable literal-delay-to-range; name this event ‘literal-delay-to-range-off’.

DEFINITION:

```
or-delay-args-to-ranges(args, result)
=  if listp(args)
```

```

then if car(args)  $\simeq$  nil
  then insert-input-delay(car(args),
    or-delay-args-to-ranges(cdr(args), result),
    t,
    f)
  elseif caar(args) = 'or'
  then or-delay-args-to-ranges(cdar(args),
    or-delay-args-to-ranges(cdr(args),
      result))
  else insert-input-delay(literal-delay-to-range(car(args)),
    or-delay-args-to-ranges(cdr(args),
      result),
    t,
    f) endif
else result endif

```

EVENT: Enable or-delay-args-to-ranges; name this event 'or-delay-args-to-ranges-off'.

DEFINITION:

```

make-delay-or(args)
= if args  $\simeq$  nil then UNKNOWN
  elseif listp(cdr(args)) then cons('or, args)
  elseif signal-namep(car(args)) then cons('or, cons(UNKNOWN, args))
  else car(args) endif

```

EVENT: Enable make-delay-or; name this event 'make-delay-or-off'.

DEFINITION:

```

delay-to-range(d)
= if d  $\simeq$  nil then d
  elseif car(d) = 'or'
  then make-delay-or(or-delay-args-to-ranges(cdr(d), nil))
  else literal-delay-to-range(d) endif

```

EVENT: Enable delay-to-range; name this event 'delay-to-range-off'.

EVENT: Disable range-dependencies; name this event 'range-dependencies-on1'.

DEFINITION:

```

addable-input-delay(in-delays, approx)
= if listp(in-delays)  $\wedge$  (cdr(in-delays)  $\simeq$  nil)

```

```

then if if approxp
  then rangep (car (in-delays))  $\vee$  (car (in-delays)  $\in \mathbf{N}$ )
  else determined-rangep (car (in-delays)) endif
  then car (in-delays)
  else f endif
else f endif

```

EVENT: Enable addable-input-delay; name this event ‘addable-input-delay-off’.

DEFINITION:

input-delay0 (*dep*, *delays*, *inputs*, *outputs*, *or-args?*, *list?*, *used-names*, *approxp*)
 = **if** *list?*

```

  then if listp (dep)
    then insert-input-delay (input-delay0 (car (dep),
                                             delays,
                                             inputs,
                                             outputs,
                                             or-args?,
                                             f,
                                             used-names,
                                             approxp),
                             input-delay0 (cdr (dep),
                                             delays,
                                             inputs,
                                             outputs,
                                             or-args?,
                                             t,
                                             used-names,
                                             approxp),
                             or-args?,
                             approxp)
    else nil endif
  elseif dep  $\simeq$  nil
  then if dep  $\in \mathbf{N}$  then dep
    elseif dep  $\in$  inputs then dep
    elseif dep  $\in$  used-names then UNKNOWN
    elseif dep  $\in$  outputs
    then if or-args? then UNKNOWN
      else parent-synonym (dep, delays) endif
    elseif boundp (dep, delays)
    then input-delay0 (delay-to-range (value (dep, delays)),
                        delays,
                        inputs,
                        outputs,

```

```

                                or-args?,
                                f,
                                cons(dep, used-names),
                                approx)
    else UNKNOWN endif
elseif car(dep) = 'or
then make-delay-or(input-delay0(cdr(dep),
                                delays,
                                inputs,
                                outputs,
                                t,
                                t,
                                used-names,
                                approx))

elseif range(dep)
then if addable-input-delay(input-delay0(range-dependencies(dep),
                                delays,
                                inputs,
                                outputs,
                                f,
                                t,
                                used-names,
                                approx),
                                approx)
    then range-plus(dep,
                    addable-input-delay(input-delay0(range-dependencies(dep),
                                delays,
                                inputs,
                                outputs,
                                f,
                                t,
                                used-names,
                                approx),
                                approx),
                    range-dependencies(addable-input-delay(input-delay0(range-dependencies(dep),
                                delays,
                                inputs,
                                outputs,
                                f,
                                t,
                                used-names,
                                approx),
                                approx)))

    else make-range(range-min(dep),

```



```

range-max(dep),
input-delay0(range-dependencies(dep),
             delays,
             inputs,
             outputs,
             f,
             t,
             used-names,
             approx)) endif

else UNKNOWN endif

```

EVENT: Enable range-dependencies; name this event ‘range-dependencies-off2’.

EVENT: Enable input-delay0; name this event ‘input-delay0-off’.

DEFINITION:

```

input-delay(dep, delays, inputs, outputs, or-args?, list?, used-names, approx)
= if listp(input-delay0(dep,
                       delays,
                       inputs,
                       outputs,
                       or-args?,
                       list?,
                       used-names,
                       approx))
   $\wedge$  rangep(car(input-delay0(dep,
                             delays,
                             inputs,
                             outputs,
                             or-args?,
                             list?,
                             used-names,
                             approx)))
then if (range-min(car(input-delay0(dep,
                                   delays,
                                   inputs,
                                   outputs,
                                   or-args?,
                                   list?,
                                   used-names,
                                   approx)))
        = 0)
   $\wedge$  (range-max(car(input-delay0(dep,

```

```

                                delays,
                                inputs,
                                outputs,
                                or-args?,
                                list?,
                                used-names,
                                approxp)))
    = 0)
then range-dependencies (car (input-delay0 (dep,
                                delays,
                                inputs,
                                outputs,
                                or-args?,
                                list?,
                                used-names,
                                approxp)))
    ∪ cdr (input-delay0 (dep,
                                delays,
                                inputs,
                                outputs,
                                or-args?,
                                list?,
                                used-names,
                                approxp)))
else input-delay0 (dep,
                    delays,
                    inputs,
                    outputs,
                    or-args?,
                    list?,
                    used-names,
                    approxp) endif
else input-delay0 (dep,
                    delays,
                    inputs,
                    outputs,
                    or-args?,
                    list?,
                    used-names,
                    approxp) endif

```

EVENT: Enable input-delay; name this event ‘input-delay-off’.

DEFINITION:

```

type-delay-slope (slope, type)
=  if type ∈ '(boolp clk level parametric tri-state)
    then per-std-load-delay-slope (slope)
    elseif type ∈ '(ttl ttl-tri-state)
    then per-pf-delay-slope (slope)
    else slope endif

```

EVENT: Enable type-delay-slope; name this event 'type-delay-slope-off'.

DEFINITION:

```

make-output-delay (delay, type, input-delay-list, approx)
=  if addable-input-delay (input-delay-list, approx)
    then make-delay (type-delay-slope (delay-slope-lh (delay), type),
                        range-plus (delay-intercept-lh (delay),
                                addable-input-delay (input-delay-list,
                                                approx)),
                        nil),
                        type-delay-slope (delay-slope-hl (delay), type),
                        range-plus (delay-intercept-hl (delay),
                                addable-input-delay (input-delay-list,
                                                approx)),
                        nil),
                        range-dependencies (addable-input-delay (input-delay-list,
                                                approx)))
    else make-delay (type-delay-slope (delay-slope-lh (delay), type),
                        delay-intercept-lh (delay),
                        type-delay-slope (delay-slope-hl (delay), type),
                        delay-intercept-hl (delay),
                        input-delay-list) endif

```

EVENT: Enable make-output-delay; name this event 'make-output-delay-off'.

DEFINITION:

```

or-delay-args (a, listp, type, delays, used-names, inputs, outputs, approx)
=  if listp
    then if listp (a)
        then or-delay-args (car (a),
                            f,
                            type,
                            delays,
                            used-names,
                            inputs,
                            outputs,

```

```

                                approxp)
      ∪ or-delay-args (cdr (a),
                        t,
                        type,
                        delays,
                        used-names,
                        inputs,
                        outputs,
                        approxp)

    else nil endif
elseif signal-namep (a)
then if a ∈ inputs then list (a)
      elseif (¬ ((a ∈ outputs) ∨ (a ∈ used-names)))
              ∧ boundp (a, delays)
      then or-delay-args (value (a, delays),
                          f,
                          type,
                          delays,
                          cons (a, used-names),
                          inputs,
                          outputs,
                          approxp)
      else list (UNKNOWN) endif
elseif a ≃ nil then list (UNKNOWN)
elseif car (a) = 'or
then or-delay-args (cdr (a),
                    t,
                    type,
                    delays,
                    used-names,
                    inputs,
                    outputs,
                    approxp)
else list (make-output-delay (a,
                              type,
                              input-delay (delay-dependencies (a),
                                              delays,
                                              inputs,
                                              outputs,
                                              f,
                                              t,
                                              used-names,
                                              approxp),
                              approxp)) endif

```

EVENT: Enable or-delay-args; name this event ‘or-delay-args-off’.

DEFINITION:

```

output-delay (name, type, delays, inputs, outputs, approx)
=  if signal-namep (value-or-unknown (name, delays))
    then parent-synonym (value-or-unknown (name, delays), delays)
    elseif value-or-unknown (name, delays)  $\simeq$  nil then UNKNOWN
    elseif car (value-or-unknown (name, delays)) = 'or
    then make-delay-or (or-delay-args (cdr (value-or-unknown (name, delays))),
                                t,
                                type,
                                delays,
                                list (name),
                                inputs,
                                outputs,
                                approx))
    else make-output-delay (value-or-unknown (name, delays),
                                type,
                                input-delay (delay-dependencies (value-or-unknown (name,
                                                                                      delays)),
                                                                                      delays,
                                                                                      inputs,
                                                                                      outputs,
                                                                                      f,
                                                                                      t,
                                                                                      list (name),
                                                                                      approx),
                                approx) endif

```

EVENT: Enable output-delay; name this event ‘output-delay-off’.

DEFINITION:

```

collect-delays-0 (names, types, delays, inputs, outputs, approx)
=  if listp (names)
    then cons (output-delay (car (names),
                                car (types),
                                delays,
                                inputs,
                                outputs,
                                approx),
              collect-delays-0 (cdr (names),
                                cdr (types),
                                delays,

```

```

                                inputs,
                                outputs,
                                approx))
else nil endif

```

EVENT: Enable collect-delays-0; name this event ‘collect-delays-0-off’.

DEFINITION:

```

collect-delays (outputs, output-types, delays, inputs, approx)
=   collect-delays-0 (outputs,
                      output-types,
                      externalize-parents (outputs, delays, inputs, outputs),
                      inputs,
                      outputs,
                      approx)

```

EVENT: Enable collect-delays; name this event ‘collect-delays-off’.

DEFINITION:

```

delays-error (delays) = err-and (delays, 'bad-delays)

```

EVENT: Enable delays-error; name this event ‘delays-error-off’.

DEFINITION:

```

initial-occ-out-depends (m-ins, m-outs)
=   cons (m-outs, pairlist (m-ins, listify (m-ins)))

```

EVENT: Enable initial-occ-out-depends; name this event ‘initial-occ-out-depends-off’.

DEFINITION:

```

add-out-depends (outputs, out-depends, arg-map, deps-map)
=   if listp (outputs)
      then add-out-depends (cdr (outputs),
                           cdr (out-depends),
                           arg-map,
                           cons (cons (car (outputs),
                                         union-values (collect-value (car (out-depends),
                                                                    arg-map),
                                                                    deps-map)),
                                   cons (all-bound-or-err (collect-value (car (out-depends),
                                                                    arg-map),
                                                                    deps-map),
                                         deps-map),

```

```

list ('output,
      car (outputs))),
      deps-map)))
else deps-map endif

```

EVENT: Enable add-out-depends; name this event 'add-out-depends-off'.

DEFINITION:

```

update-out-depends (out-depends, o-outs, o-fn-props, o-ins, arg-map)
= cons (o-ins ∪ car (out-depends),
      add-out-depends (o-outs,
                       value ('out-depends, o-fn-props),
                       arg-map,
                       cdr (out-depends)))

```

EVENT: Enable update-out-depends; name this event 'update-out-depends-off'.

DEFINITION:

```

collect-out-depends (outputs, dependencies)
= collect-value (outputs, cdr (dependencies))

```

EVENT: Enable collect-out-depends; name this event 'collect-out-depends-off'.

DEFINITION:

```

collect-out-depends-errors (deps-map, used-signals)
= if listp (deps-map)
  then if net-errorp (car (deps-map))
    then if cadr (error-label (car (deps-map))) ∈ used-signals
      then cons (car (deps-map),
                 collect-out-depends-errors (cdr (deps-map),
                                              used-signals))
    else collect-out-depends-errors (cdr (deps-map),
                                     used-signals) endif
  else collect-out-depends-errors (cdr (deps-map), used-signals) endif
else nil endif

```

EVENT: Enable collect-out-depends-errors; name this event 'collect-out-depends-errors-off'.

DEFINITION:

```

out-depends-error (dependencies)
= nlistp-or-err (collect-out-depends-errors (cdr (dependencies),
                                              car (dependencies)),
                'unbound-output-dependencies)

```

EVENT: Enable out-depends-error; name this event ‘out-depends-error-off’.

DEFINITION: INITIAL-OCC-STATE-TYPES = **nil**

DEFINITION:

```
update-state-types(state-types, o-name, o-fn-props)
=  if boundp('state-types, o-fn-props)
    then cons(cons(o-name, value('state-types, o-fn-props)), state-types)
    else state-types endif
```

EVENT: Enable update-state-types; name this event ‘update-state-types-off’.

DEFINITION:

```
collect-state-types(states, state-types)
=  if listp(states) ∨ (states = nil)
    then collect-value-or-unknown(states, state-types)
    else value-or-unknown(states, state-types) endif
```

EVENT: Enable collect-state-types; name this event ‘collect-state-types-off’.

DEFINITION:

```
state-types-error(state-types, states)
=  t-or-err(set-equal(strip-cars(state-types), m-states-list(states)),
            'bad-state-types,
            list(list('module-states, states),
                  list('computed-states, strip-cars(state-types))))
```

EVENT: Enable state-types-error; name this event ‘state-types-error-off’.

DEFINITION:

```
initial-occ-tri-state-data(m-outs)
=  cons(cons('free-signals, m-outs),
        '((t-wire-ins) (synonyms)))
```

EVENT: Enable initial-occ-tri-state-data; name this event ‘initial-occ-tri-state-data-off’.

DEFINITION:

```
function-t-wire-ins(fn, data)
=  if boundp('tri-state-data, data)
    then value('t-wire-ins, value('tri-state-data, data))
    elseif fn = 't-wire then value('inputs, data)
    else nil endif
```


EVENT: Enable function-t-wire-ins; name this event 'function-t-wire-ins-off'.

DEFINITION:

```
function-synonyms(fn, data)
=  if boundp('tri-state-data, data)
    then value('synonyms, value('tri-state-data, data))
    elseif fn = 'id
    then pairlist(value('outputs, data), value('inputs, data))
    else nil endif
```

EVENT: Enable function-synonyms; name this event 'function-synonyms-off'.

DEFINITION:

```
add-synonyms(f-synonyms, in-map, out-map, slist)
=  if listp(f-synonyms)
    then add-synonyms(cdr(f-synonyms),
                      in-map,
                      out-map,
                      cons(cons(value(caar(f-synonyms), out-map),
                                value2(cdar(f-synonyms), in-map, out-map))),
                          slist))
    else slist endif
```

EVENT: Enable add-synonyms; name this event 'add-synonyms-off'.

DEFINITION:

```
update-tri-state-data(data, o-ins, o-fn, o-fn-data, in-map, out-map)
=  list(cons('free-signals,
             if listp(collect-value2(function-t-wire-ins(o-fn, o-fn-data),
                                     in-map,
                                     out-map))
             then set-diff(o-ins,
                           collect-value2(function-t-wire-ins(o-fn,
                                                                o-fn-data),
                                                                in-map,
                                                                out-map))
             elseif o-fn = 'id then nil
             else o-ins endif
             ∪ value('free-signals, data)),
        cons('t-wire-ins,
             append(collect-value2(function-t-wire-ins(o-fn, o-fn-data),
                                                         in-map,
                                                         out-map),
```

```

value('t-wire-ins, data))),
cons('synonyms,
add-synonyms (function-synonyms (o-fn, o-fn-data),
in-map,
out-map,
value('synonyms, data))))

```

EVENT: Enable update-tri-state-data; name this event 'update-tri-state-data-off'.

DEFINITION:

```

io-signal-renames (synonyms, io-signals, slist)
= if listp (synonyms)
  then if (io-outp (caar (synonyms))
    ∨ io-outp (parent-synonym (cdar (synonyms), slist))
    ∨ (parent-synonym (cdar (synonyms), slist) ∈ io-signals)
    ∧ (caar (synonyms)
      ≠ parent-synonym (cdar (synonyms), slist))
  then cons (cons (caar (synonyms),
    parent-synonym (cdar (synonyms), slist)),
    io-signal-renames (cdr (synonyms), io-signals, slist))
  else io-signal-renames (cdr (synonyms), io-signals, slist) endif
else nil endif

```

EVENT: Enable io-signal-renames; name this event 'io-signal-renames-off'.

DEFINITION:

```

io-rename-error (synonyms, io-signals)
= nlistp-or-err (io-signal-renames (synonyms, io-signals, synonyms),
'renamed-io-signals)

```

EVENT: Enable io-rename-error; name this event 'io-rename-error-off'.

DEFINITION:

```

t-wire-error (free-signals, t-wire-ins)
= err-and (list (disjoint-or-err (remove-duplicates (t-wire-ins),
free-signals,
't-wire-ins-used-elsewhere),
no-duplicates-or-err (t-wire-ins,
'duplicate-t-wire-inputs)),
't-wire-errors)

```

EVENT: Enable t-wire-error; name this event 't-wire-error-off'.

DEFINITION:

```
compose-tri-state-data(data, io-signals)
= list (cons ('free-signals,
              parent-synonyms-list (value ('free-signals, data),
                                           value ('synonyms, data))),
        cons ('t-wire-ins,
              parent-synonyms-list (value ('t-wire-ins, data),
                                           value ('synonyms, data))),
        cons ('synonyms, value ('synonyms, data)),
        io-rename-error (value ('synonyms, data), io-signals),
        t-wire-error (parent-synonyms-list (value ('free-signals, data),
                                                  value ('synonyms, data)),
                      parent-synonyms-list (value ('t-wire-ins, data),
                                              value ('synonyms, data))))
```

EVENT: Enable compose-tri-state-data; name this event 'compose-tri-state-data-off'.

DEFINITION:

```
collect-synonyms(m-outs, synonyms)
= if listp(m-outs)
  then if car(m-outs) = parent-synonym(car(m-outs), synonyms)
    then collect-synonyms(cdr(m-outs), synonyms)
    else cons (cons (car(m-outs),
                    parent-synonym(car(m-outs), synonyms)),
              collect-synonyms(cdr(m-outs), synonyms)) endif
  else nil endif
```

EVENT: Enable collect-synonyms; name this event 'collect-synonyms-off'.

DEFINITION:

```
collect-tri-state-data(m-ins, m-outs, data)
= list (cons ('t-wire-ins,
              intersection(m-ins, value ('t-wire-ins, data))),
        cons ('synonyms,
              collect-synonyms(m-outs,
                              externalize-parents(m-outs,
                                                  value ('synonyms,
                                                          data),
                                                  m-ins,
                                                  m-outs))))
```

EVENT: Enable collect-tri-state-data; name this event 'collect-tri-state-data-off'.

DEFINITION:

```
tri-state-errors (data)
=  if net-errorp (error-entry (data, 'renamed-io-signals))
    ∧ net-errorp (error-entry (data, 't-wire-errors))
  then list (error-entry (data, 'renamed-io-signals),
             error-entry (data, 't-wire-errors))
  elseif net-errorp (error-entry (data, 'renamed-io-signals))
  then error-entry (data, 'renamed-io-signals)
  elseif net-errorp (error-entry (data, 't-wire-errors))
  then error-entry (data, 't-wire-errors)
  else nil endif
```

EVENT: Enable tri-state-errors; name this event 'tri-state-errors-off'.

DEFINITION:

```
initial-occ-data (props, m-ins, m-outs)
=  if listp (props)
  then if listp (case on car (props):
    case = delays
    then cons ('delays, INITIAL-OCC-DELAYS)
    case = drives
    then cons ('drives, INITIAL-OCC-DRIVES)
    case = input-types
    then cons ('input-types, INITIAL-OCC-IN-TYPES)
    case = loadings
    then cons ('loadings, INITIAL-OCC-LOADINGS)
    case = out-depends
    then cons ('out-depends,
               initial-occ-out-depends (m-ins, m-outs))
    case = output-types
    then cons ('output-types, INITIAL-OCC-OUT-TYPES)
    case = state-types
    then cons ('state-types, INITIAL-OCC-STATE-TYPES)
    case = tri-state-data
    then cons ('tri-state-data,
               initial-occ-tri-state-data (m-outs))
    otherwise if car (props)
      ∈ ' (gates pads primitives
            transistors)
    then cons (car (props), 0)
    else nil endif endcase)
  then cons (case on car (props):
    case = delays
    then cons ('delays, INITIAL-OCC-DELAYS)
```

```

case = drives
  then cons('drives, INITIAL-OCC-DRIVES)
case = input-types
  then cons('input-types, INITIAL-OCC-IN-TYPES)
case = loadings
  then cons('loadings, INITIAL-OCC-LOADINGS)
case = out-depends
  then cons('out-depends,
            initial-occ-out-depends(m-ins, m-outs))
case = output-types
  then cons('output-types, INITIAL-OCC-OUT-TYPES)
case = state-types
  then cons('state-types, INITIAL-OCC-STATE-TYPES)
case = tri-state-data
  then cons('tri-state-data,
            initial-occ-tri-state-data(m-outs))
otherwise if car(props)
  ∈ '(gates pads primitives
      transistors)
  then cons(car(props), 0)
  else nil endif endcase,
  initial-occ-data(cdr(props), m-ins, m-outs))
else initial-occ-data(cdr(props), m-ins, m-outs) endif
else nil endif

```

EVENT: Enable initial-occ-data; name this event 'initial-occ-data-off'.

DEFINITION:

```

update-oprop-binding(o-name,
  o-ins,
  o-outs,
  o-fn,
  o-fn-props,
  in-map,
  out-map,
  binding)
= if listp(binding)
  then cons(car(binding),
    case on car(binding):
    case = delays
    then update-delays(cdr(binding),
      o-outs,
      o-fn-props,
      in-map,

```

```

                                out-map)
case = drives
  then update-drives (cdr (binding),
                        o-outs,
                        o-fn-props,
                        in-map,
                        out-map)
case = input-types
  then update-in-types (cdr (binding), o-ins, o-fn-props)
case = loadings
  then update-loadings (cdr (binding), o-ins, o-fn-props)
case = out-depends
  then update-out-depends (cdr (binding),
                              o-outs,
                              o-fn-props,
                              o-ins,
                              in-map)
case = output-types
  then update-out-types (cdr (binding),
                          o-outs,
                          o-fn-props,
                          in-map)
case = state-types
  then update-state-types (cdr (binding), o-name, o-fn-props)
case = tri-state-data
  then update-tri-state-data (cdr (binding),
                              o-ins,
                              o-fn,
                              o-fn-props,
                              in-map,
                              out-map)
otherwise if car (binding)
  ∈ '(gates pads primitives
       transistors)
  then value (car (binding), o-fn-props)
        + cdr (binding)
  else cdr (binding) endif endcase
else binding endif

```

EVENT: Enable update-oprop-binding; name this event 'update-oprop-binding-off'.

DEFINITION:
update-occ-bindings (*o-name*,

```

        o-ins,
        o-outs,
        o-fn,
        o-fn-props,
        in-map,
        out-map,
        bindings)
=  if listp (bindings)
    then cons (update-oprop-binding (o-name,
                                   o-ins,
                                   o-outs,
                                   o-fn,
                                   o-fn-props,
                                   in-map,
                                   out-map,
                                   car (bindings)),
              update-occ-bindings (o-name,
                                   o-ins,
                                   o-outs,
                                   o-fn,
                                   o-fn-props,
                                   in-map,
                                   out-map,
                                   cdr (bindings)))
    else nil endif

```

EVENT: Enable update-occ-bindings; name this event ‘update-occ-bindings-off’.

DEFINITION:

```

function-properties (fn, database)
=  if primp (fn) then cdr (primp (fn))
    else value (fn, database) endif

```

EVENT: Enable function-properties; name this event ‘function-properties-off’.

DEFINITION:

```

collect-occurrence-data (body, bindings, io-outputs, io-signals, database)
=  if listp (body)
    then collect-occurrence-data (cdr (body),
                                update-occ-bindings (occ-name (car (body)),
                                                        mark-io-outs (occ-inputs (car (body)),
                                                                    io-outputs),
                                                        mark-io-outs (occ-outputs (car (body)),

```

```

                                io-signals),
occ-function (car (body)),
function-properties (occ-function (car (body)),
                                database),
pairlist (value ('inputs,
                function-properties (occ-function (car (body)),
                                database)),
                mark-io-outs (occ-inputs (car (body)),
                                io-outputs)),
pairlist (value ('outputs,
                function-properties (occ-function (car (body)),
                                database)),
                mark-io-outs (occ-outputs (car (body)),
                                io-signals)),
                                bindings),
append (intersection (io-signals,
                    occ-outputs (car (body))),
        io-outputs),
        io-signals,
        database)

else bindings endif

```

EVENT: Enable collect-occurrence-data; name this event 'collect-occurrence-data-off'.

DEFINITION:

```

composed-io-types (in-types, out-types, tri-state-data, io-signals)
= list (cons ('tri-state-data,
            compose-tri-state-data (tri-state-data, io-signals)),
      cons ('input-types,
            car (compose-io-types (in-types,
                                out-types,
                                value ('t-wire-ins, tri-state-data)))),
      cons ('output-types,
            cdr (compose-io-types (in-types,
                                out-types,
                                value ('t-wire-ins, tri-state-data))))))

```

EVENT: Enable composed-io-types; name this event 'composed-io-types-off'.

DEFINITION:

```

composed-loadings-drives-delays (loadings, drives, delays)
= list (cons ('delays,

```



```

loaded-delays (delays, transfer-loadings (loadings, drives)),
cons ('drives,
      net-drives-simple (drives, transfer-loadings (loadings, drives))),
cons ('loadings, transfer-loadings (loadings, drives)))

```

EVENT: Enable composed-loadings-drives-delays; name this event 'composed-loadings-drives-delays-off'.

DEFINITION:

```

add-composed-data (prop-name, bindings, io-signals, result)
=  if boundp (prop-name, result)  $\vee$  ( $\neg$  boundp (prop-name, bindings))
    then result
    elseif prop-name  $\in$  ' (input-types output-types
                          tri-state-data)
    then append (composed-io-types (value ('input-types, bindings),
                                          value ('output-types, bindings),
                                          value ('tri-state-data, bindings),
                                          io-signals),
                result)
    elseif prop-name  $\in$  ' (loadings drives delays)
    then append (composed-loadings-drives-delays (value ('loadings,
                                                         bindings),
                                                         value ('drives, bindings),
                                                         value ('delays, bindings)),
                result)
    else cons (alist-entry (prop-name, bindings), result) endif

```

EVENT: Enable add-composed-data; name this event 'add-composed-data-off'.

DEFINITION:

```

composed-occurrence-data (bindings, io-signals, result)
=  if listp (bindings)
    then composed-occurrence-data (cdr (bindings),
                                     io-signals,
                                     if car (bindings)  $\simeq$  nil
                                     then cons (car (bindings), result)
                                     else add-composed-data (caar (bindings),
                                                              bindings,
                                                              io-signals,
                                                              result) endif)
    else result endif

```

EVENT: Enable composed-occurrence-data; name this event 'composed-occurrence-data-off'.

DEFINITION:

```
compose-occurrence-data (bindings, io-signals)  
=   composed-occurrence-data (bindings, io-signals, nil)
```

EVENT: Enable compose-occurrence-data; name this event ‘compose-occurrence-data-off’.

DEFINITION:

```
occurrence-data (props, body, m-ins, m-outs, io-signals, database)  
=   compose-occurrence-data (collect-occurrence-data (body,  
                                                         initial-occ-data (props,  
                                                         m-ins,  
                                                         m-outs),  
                                                         nil,  
                                                         io-signals,  
                                                         database),  
                             io-signals)
```

EVENT: Enable occurrence-data; name this event ‘occurrence-data-off’.

DEFINITION:

```
collect-module-prop (prop-name,  
                    m-ins,  
                    in-types,  
                    m-outs,  
                    out-types,  
                    m-states,  
                    obindings,  
                    approximate-delays-p)  
=   case on prop-name:  
     case = delays  
     then cons ('delays,  
                collect-delays (m-outs,  
                                out-types,  
                                cdr (alist-entry (prop-name, obindings)),  
                                m-ins,  
                                approximate-delays-p))  
  
     case = drives  
     then cons ('drives,  
                collect-drives (m-outs,  
                                out-types,  
                                cdr (alist-entry (prop-name, obindings)),  
                                m-ins))
```

```

case = input-types
  then cons ('input-types, in-types)
case = inputs
  then cons ('inputs, m-ins)
case = loadings
  then cons ('loadings,
             collect-loadings (m-ins,
                               in-types,
                               cdr (alist-entry (prop-name, obindings))))
case = out-depends
  then cons ('out-depends,
             collect-out-depends (m-outs,
                                  cdr (alist-entry (prop-name, obindings))))
case = output-types
  then cons ('output-types, out-types)
case = outputs
  then cons ('outputs, unmark-io-outs (m-outs))
case = state-types
  then if m-states = nil then nil
        else cons ('state-types,
                   collect-state-types (m-states,
                                         cdr (alist-entry (prop-name,
                                                             obindings)))) endif
case = states
  then if m-states = nil then nil
        else cons ('states, m-states) endif
case = tri-state-data
  then cons ('tri-state-data,
             collect-tri-state-data (m-ins,
                                     m-outs,
                                     cdr (alist-entry (prop-name,
                                                         obindings))))
otherwise alist-entry (prop-name, obindings) endcase

```

EVENT: Enable collect-module-prop; name this event 'collect-module-prop-off'.

DEFINITION:

```

collect-module-props (props,
                     m-ins,
                     in-types,
                     m-outs,
                     out-types,
                     m-states,
                     obindings,

```

```

                                approximate-delays-p)
=  if listp (props)
    then if listp (collect-module-prop (car (props),
                                         m-ins,
                                         in-types,
                                         m-outs,
                                         out-types,
                                         m-states,
                                         obindings,
                                         approximate-delays-p))
        then cons (collect-module-prop (car (props),
                                             m-ins,
                                             in-types,
                                             m-outs,
                                             out-types,
                                             m-states,
                                             obindings,
                                             approximate-delays-p),
                    collect-module-props (cdr (props),
                                             m-ins,
                                             in-types,
                                             m-outs,
                                             out-types,
                                             m-states,
                                             obindings,
                                             approximate-delays-p))
        else collect-module-props (cdr (props),
                                         m-ins,
                                         in-types,
                                         m-outs,
                                         out-types,
                                         m-states,
                                         obindings,
                                         approximate-delays-p) endif
    else nil endif

```

EVENT: Enable collect-module-props; name this event ‘collect-module-props-off’.

DEFINITION:

```

module-prop-error (obinding, m-states)
=  if listp (obinding)
    then case on car (obinding):
        case = delays

```

```

then delays-error (cdr (obinding))
case = drives
  then drives-error (cdr (obinding))
case = input-types
  then in-types-error (cdr (obinding))
case = loadings
  then loadings-error (cdr (obinding))
case = out-depends
  then out-depends-error (cdr (obinding))
case = output-types
  then out-types-error (cdr (obinding))
case = state-types
  then state-types-error (cdr (obinding), m-states)
case = tri-state-data
  then tri-state-errors (cdr (obinding))
otherwise nil endcase
else obinding endif

```

EVENT: Enable module-prop-error; name this event ‘module-prop-error-off’.

DEFINITION:

```

collect-module-errors (obindings, m-states)
= if listp (obindings)
  then if net-errorp (module-prop-error (car (obindings), m-states))
    then cons (module-prop-error (car (obindings), m-states),
      collect-module-errors (cdr (obindings), m-states))
    elseif listp (module-prop-error (car (obindings), m-states))
    then append (module-prop-error (car (obindings), m-states),
      collect-module-errors (cdr (obindings), m-states))
    else collect-module-errors (cdr (obindings), m-states) endif
  else nil endif

```

EVENT: Enable collect-module-errors; name this event ‘collect-module-errors-off’.

DEFINITION:

```

collect-module-data (props,
  m-name,
  m-ins,
  m-outs,
  m-states,
  obindings,
  approximate-delays-p)

```

```

=  if net-errorp (nlistp-or-err (collect-module-errors (obindings, m-states),
                                                    list ('module, m-name)))
then cons(m-name,
          cons(nlistp-or-err (collect-module-errors (obindings, m-states),
                                                    list ('module, m-name)),
              collect-module-props(props,
                                   m-ins,
                                   if 'input-types ∈ props
                                   then collect-in-types(m-ins,
                                                         value ('input-types,
                                                                obindings))

                                   else nil endif,
                                   m-outs,
                                   if 'output-types ∈ props
                                   then collect-out-types(m-outs,
                                                         value ('output-types,
                                                                obindings),
                                                         m-ins,
                                                         value ('input-types,
                                                                obindings))

                                   else nil endif,
                                   m-states,
                                   obindings,
                                   approximate-delays-p)))
else cons(m-name,
          collect-module-props(props,
                               m-ins,
                               if 'input-types ∈ props
                               then collect-in-types(m-ins,
                                                         value ('input-types,
                                                                obindings))

                               else nil endif,
                               m-outs,
                               if 'output-types ∈ props
                               then collect-out-types(m-outs,
                                                         value ('output-types,
                                                                obindings),
                                                         m-ins,
                                                         value ('input-types,
                                                                obindings))

                               else nil endif,
                               m-states,
                               obindings,
                               approximate-delays-p)) endif

```

EVENT: Enable collect-module-data; name this event ‘collect-module-data-off’.

DEFINITION:

```
module-data (module, props, database, approximate-delays-p)
= collect-module-data (props,
    module-name (module),
    module-inputs (module),
    mark-io-outs (module-outputs (module),
        intersection (module-inputs (module),
            module-outputs (module))),
    module-statenames (module),
    occurrence-data (props,
        module-occurrences (module),
        module-inputs (module),
        mark-io-outs (module-outputs (module),
            intersection (module-inputs (module),
                module-outputs (module))),
        intersection (module-inputs (module),
            module-outputs (module)),
        database),
    approximate-delays-p)
```

EVENT: Enable module-data; name this event ‘module-data-off’.

DEFINITION:

```
module-database (netlist, props, approximate-delays-p)
= if listp (netlist)
    then cons (module-data (car (netlist),
        props,
        module-database (cdr (netlist),
            props,
            approximate-delays-p),
        approximate-delays-p),
        module-database (cdr (netlist), props, approximate-delays-p))
    else nil endif
```

EVENT: Enable module-database; name this event ‘module-database-off’.

DEFINITION:

```
required-for-prop (prop)
= case on prop:
    case = delays
    then '(delays loadings drives input-types output-types
```

```

        inputs outputs tri-state-data)
case = drives
  then '(drives loadings input-types output-types inputs
        outputs tri-state-data)
case = input-types
  then '(input-types output-types inputs outputs
        tri-state-data)
case = loadings
  then '(loadings drives input-types output-types inputs
        outputs tri-state-data)
case = out-depends
  then '(out-depends inputs outputs)
case = output-types
  then '(output-types input-types inputs outputs
        tri-state-data)
otherwise if prop ∈ ALL-MODULE-PROPS then list(prop)
        else nil endif endcase

```

EVENT: Enable required-for-prop; name this event 'required-for-prop-off'.

DEFINITION:

```

get-required-props(props, wanted-props)
= if listp(props)
  then required-for-prop(car(props))
    ∪ get-required-props(cdr(props), wanted-props)
  else wanted-props endif

```

EVENT: Enable get-required-props; name this event 'get-required-props-off'.

DEFINITION:

```

required-props(props) = get-required-props(props, props)

```

EVENT: Enable required-props; name this event 'required-props-off'.

DEFINITION:

```

unbind-netlist-props-0(props, database)
= if listp(database)
  then cons(cons(caar(database),
                if net-errorp(cadar(database))
                then cons(cadar(database),
                          unbind-list(props, cddar(database)))
                else unbind-list(props, cdar(database)) endif),
            unbind-netlist-props-0(props, cdr(database)))
  else nil endif

```


EVENT: Enable unbind-netlist-props-0; name this event ‘unbind-netlist-props-0-off’.

DEFINITION:

```
unbind-netlist-props(props, database)  
=  if listp(props) then unbind-netlist-props-0(reverse(props), database)  
   else database endif
```

EVENT: Enable unbind-netlist-props; name this event ‘unbind-netlist-props-off’.

DEFINITION:

```
netlist-properties(netlist, props, approximate-delays-p)  
=  if net-errorp(subset-or-err(props,  
                               ALL-MODULE-PROPS,  
                               'unknown-properties'))  
   then subset-or-err(props, ALL-MODULE-PROPS, 'unknown-properties')  
   elseif listp(collect-net-errors(module-database(netlist,  
                                                    required-props(props),  
                                                    approximate-delays-p)))  
   then pred-error('netlist-errors,  
                   collect-net-errors(module-database(netlist,  
                                                       required-props(props),  
                                                       approximate-delays-p)))  
   else unbind-netlist-props(set-diff(required-props(props), props),  
                              module-database(netlist,  
                                              required-props(props),  
                                              approximate-delays-p)) endif
```

EVENT: Enable netlist-properties; name this event ‘netlist-properties-off’.

DEFINITION:

```
netlist-database(netlist, approximate-delays-p)  
=  unbind-netlist-props(set-diff(required-props(ALL-MODULE-PROPS),  
                                  ALL-MODULE-PROPS),  
                          module-database(netlist,  
                                          required-props(ALL-MODULE-PROPS),  
                                          approximate-delays-p))
```

EVENT: Enable netlist-database; name this event ‘netlist-database-off’.

DEFINITION:

PREDICATE-PROPERTIES

```
= '(drives input-types loadings out-depends output-types
    state-types)
```

EVENT: Enable predicate-properties; name this event 'predicate-properties-off'.

EVENT: Enable *1*predicate-properties; name this event 'g*1*predicate-properties-off'.

DEFINITION:

top-level-predicate (*netlist*)

```
= if net-errorp (netlist-syntax-ok (netlist))
    then netlist-syntax-ok (netlist)
    elseif net-errorp (netlist-properties (netlist, PREDICATE-PROPERTIES, t))
    then netlist-properties (netlist, PREDICATE-PROPERTIES, t)
    else t endif
```

EVENT: Enable top-level-predicate; name this event 'top-level-predicate-off'.

DEFINITION:

lsi-top-level-predicate (*netlist*, *token-size*, *max-hierarchical-name-length*)

```
= if net-errorp (lsi-netlist-syntax-ok (netlist,
                                         token-size,
                                         max-hierarchical-name-length))
    then lsi-netlist-syntax-ok (netlist,
                                token-size,
                                max-hierarchical-name-length)
    elseif net-errorp (netlist-properties (netlist, PREDICATE-PROPERTIES, t))
    then netlist-properties (netlist, PREDICATE-PROPERTIES, t)
    else t endif
```

EVENT: Enable lsi-top-level-predicate; name this event 'lsi-top-level-predicate-off'.

DEFINITION:

pname-listp (*x*)

```
= if listp (x)
    then name-okp (car (x))
        ∧ (car (x) ∉ cdr (x))
        ∧ pname-listp (cdr (x))
    else x = nil endif
```

EVENT: Enable pname-listp; name this event 'pname-listp-off'.

DEFINITION:

```

function-call-ok( $x$ ,  $number-of-args$ )
=  if listp( $x$ )  $\wedge$  litatom(car( $x$ ))
    then if properp( $x$ )
        then if  $number-of-args \in \mathbf{N}$ 
            then t-or-err(length(cdr( $x$ )) =  $number-of-args$ ,
                cons('function,
                    cons(car( $x$ ),
                        cons('has,
                            cons( $number-of-args$ , '(args))))),
                    list('it, 'is, 'given, length(cdr( $x$ )), 'in,  $x$ ))
            else t endif
        else pred-error('ill-formed-function-call,  $x$ ) endif
    else pred-error('not-function-call,  $x$ ) endif

```

EVENT: Enable function-call-ok; name this event 'function-call-ok-off'.

DEFINITION:

```

io-label( $arg$ ,  $what$ )
=  if  $arg \in \mathbf{N}$  then list('unknown,  $what$ ,  $arg$ )
    else list( $what$ ,  $arg$ ) endif

```

EVENT: Enable io-label; name this event 'io-label-off'.

DEFINITION: input-label($input$) = io-label($input$, 'input)

EVENT: Enable input-label; name this event 'input-label-off'.

DEFINITION: output-label($output$) = io-label($output$, 'output)

EVENT: Enable output-label; name this event 'output-label-off'.

DEFINITION: state-label($state$) = io-label($state$, 'state)

EVENT: Enable state-label; name this event 'state-label-off'.

DEFINITION:

```

ucar( $x$ )
=  if listp( $x$ ) then car( $x$ )
    else UNKNOWN endif

```

EVENT: Enable ucar; name this event 'ucar-off'.

DEFINITION:

```
ucdr(x)  
=  if listp(x) then cdr(x)  
    else UNKNOWN endif
```

EVENT: Enable ucdr; name this event ‘ucdr-off’.

DEFINITION:

```
io-car(args)  
=  if listp(args) then car(args)  
    elseif args ∈ N then args  
    else 1 endif
```

EVENT: Enable io-car; name this event ‘io-car-off’.

DEFINITION:

```
io-cdr(args)  
=  if args ∈ N then 1 + args  
    elseif args ≈ nil then 2  
    elseif cdr(args) ∈ N then nil  
    else cdr(args) endif
```

EVENT: Enable io-cdr; name this event ‘io-cdr-off’.

DEFINITION:

```
out-prop-signals(out-depends, inputs)  
=  if unknownp(out-depends) then inputs  
    else out-depends endif
```

EVENT: Enable out-prop-signals; name this event ‘out-prop-signals-off’.

DEFINITION:

```
id-out-prop-signals(output-type, out-depends, inputs)  
=  if listp(output-type) then output-type  
    else out-prop-signals(out-depends, inputs) endif
```

EVENT: Enable id-out-prop-signals; name this event ‘id-out-prop-signals-off’.

DEFINITION:

```
ALL-INPUT-TYPES  
=  '(boolp clk free level parametric tri-state ttl  
    ttl-tri-state)
```

EVENT: Enable all-input-types; name this event 'all-input-types-off'.

EVENT: Enable *1*all-input-types; name this event 'g*1*all-input-types-off'.

DEFINITION:

```
all-output-types(inputs)  
= append(listify(inputs), delete*('free, ALL-INPUT-TYPES))
```

EVENT: Enable all-output-types; name this event 'all-output-types-off'.

DEFINITION:

```
pin-type-ok(type, input, io-signals, fname)  
= label-error(if fname = 't-wire  
    then t-or-err(type = 'tri-state,  
        't-wire-input-type-not-tri-state,  
        type)  
    elseif input ∈ io-signals  
    then t-or-err(tri-state-typep(type),  
        'bad-io-signal-type,  
        type)  
    else t-or-err(type ∈ ALL-INPUT-TYPES,  
        'unknown-type,  
        type) endif,  
    input-label(input))
```

EVENT: Enable pin-type-ok; name this event 'pin-type-ok-off'.

DEFINITION:

```
pin-type-errors(types, inputs, io-signals, fname)  
= if listp(types)  
    then cons(t-or-err(listp(inputs) ∨ (inputs ∈ N),  
        'too-many-input-types,  
        cons(length(types), '(extras))),  
        cons(pin-type-ok(car(types), io-car(inputs), io-signals, fname),  
            pin-type-errors(cdr(types),  
                io-cdr(inputs),  
                io-signals,  
                fname)))  
    else list(nil-or-err(types, 'not-proper-list),  
        nlistp-or-err(inputs, 'inputs-without-types)) endif
```

EVENT: Enable pin-type-errors; name this event 'pin-type-errors-off'.

DEFINITION:

```
pin-types-ok(in-types, alist)
=  err-and (pin-type-errors(in-types,
    if unknownp (value('inputs, alist)) then 1
    else value('inputs, alist) endif,
    intersection (value('inputs, alist),
        value('outputs, alist)),
    value('primp-name, alist)),
    'input-types)
```

EVENT: Enable pin-types-ok; name this event 'pin-types-ok-off'.

DEFINITION:

```
pout-type-ok(type, output, out-depends, inputs, io-signals, fname, in-type-map)
=  label-error (if fname = 't-wire
    then t-or-err (type = 'tri-state,
        't-wire-output-type-not-tri-state,
        type)
    elseif output ∈ io-signals
    then if tri-state-typep (type)
        then t-or-err ((¬ tri-state-typep (value (output,
            in-type-map)))
            ∨ (type = value (output,
                in-type-map)),
            'different-io-signal-in-and-out-types,
            list (list ('input-type,
                value (output, in-type-map)),
                list ('output-type, type)))
        else pred-error ('bad-io-signal-type, type) endif
    elseif listp (type)
    then t-or-err (properp (type)
        ∧ (length (type) = 1)
        ∧ if unknownp (out-prop-signals (out-depends,
            inputs))
            then name-okp (car (type))
            else car (type)
                ∈ out-prop-signals (out-depends,
                    inputs) endif,
        'unknown-type,
        type)
    else t-or-err (type ∈ all-output-types (nil),
        'unknown-type,
        type) endif,
    output-label (output))
```

EVENT: Enable pout-type-ok; name this event ‘pout-type-ok-off’.

DEFINITION:

```

pout-type-errors (types,
                  outputs,
                  out-depends,
                  inputs,
                  io-signals,
                  fname,
                  in-type-map)
=  if listp (types)
    then cons (t-or-err (listp (outputs)  $\vee$  (outputs  $\in$   $\mathbf{N}$ ),
                        'too-many-output-types,
                        cons (length (types), 'extras))),
              cons (pout-type-ok (car (types),
                                io-car (outputs),
                                ucar (out-depends),
                                inputs,
                                io-signals,
                                fname,
                                in-type-map),
                    pout-type-errors (cdr (types),
                                      io-cdr (outputs),
                                      ucdr (out-depends),
                                      inputs,
                                      io-signals,
                                      fname,
                                      in-type-map)))
    else list (nil-or-err (types, 'not-proper-list),
              nlistp-or-err (outputs, 'outputs-without-types)) endif

```

EVENT: Enable pout-type-errors; name this event ‘pout-type-errors-off’.

DEFINITION:

```

pout-types-ok (out-types, alist)
=  err-and (pout-type-errors (out-types,
                              if unknownp (value ('outputs, alist)) then 1
                              else value ('outputs, alist) endif,
                              value ('out-depends, alist),
                              value ('inputs, alist),
                              intersection (value ('inputs, alist),
                                             value ('outputs, alist)),
                              value ('primp-name, alist),

```

```

pairlist (value ('inputs, alist),
           value ('input-types, alist)),
'output-types)

```

EVENT: Enable pout-types-ok; name this event 'pout-types-ok-off'.

DEFINITION:

```

p-name-prop-value-ok (name, out-type, out-depends, inputs, label)
=  t-or-err (if unknownp (id-out-prop-signals (out-type, out-depends, inputs))
             then name-okp (name)
             else name ∈ id-out-prop-signals (out-type,
                                                out-depends,
                                                inputs) endif,
             label,
             name)

```

EVENT: Enable p-name-prop-value-ok; name this event 'p-name-prop-value-ok-off'.

DEFINITION:

```

p-t-wire-args-ok (args, out-depends, inputs, label)
=  err-and (if unknownp (out-prop-signals (out-depends, inputs))
            then list (name-list-ok (args, 'ill-formed-args, f))
            else list (subset-or-err (args,
                                      out-prop-signals (out-depends,
                                                         inputs),
                                                         'unknown-args),
                                      no-duplicates-or-err (args,
                                                            'duplicate-args),
                                      nil-or-err (last-cdr (args),
                                                  'not-proper-list)) endif,
            label)

```

EVENT: Enable p-t-wire-args-ok; name this event 'p-t-wire-args-ok-off'.

DEFINITION:

```

phalf-delay-ok (d, type, label)
=  if length (d) = 2
    then err-and (list (if type ∈ ' (boolp clk level parametric
                                   tri-state)
                        then t-or-err (car (d) ∈ N,
                                       'slope-not-number,
                                       list (list ('slope, car (d)),

```



```

                                list ('output-type, type)))
elseif type ∈ '(ttl ttl-tri-state)
then t-or-err (ps-pfp (car (d)),
                'slope-not-ps-pf,
                list (list ('slope, car (d)),
                        list ('output-type, type)))
else t-or-err ((car (d) ∈ N)
                ∨ ps-pfp (car (d)),
                'slope-not-number-or-ps-pf,
                car (d)) endif,
t-or-err (cadr (d) ∈ N,
          'intercept-not-number,
          cadr (d)),
nil-or-err (last-cdr (d), 'not-proper-list)),
label)
else pred-error (label, list ('length-not-2, d)) endif

```

EVENT: Enable phalf-delay-ok; name this event 'phalf-delay-ok-off'.

DEFINITION:

```

pdelay-ok (delay, output, type, out-depends, inputs)
= label-error (if litatom (delay) ∨ indexp (delay)
then p-name-prop-value-ok (delay,
                            type,
                            out-depends,
                            inputs,
                            'unknown-delay)
elseif listp (delay) ∧ (car (delay) = 'or)
then p-t-wire-args-ok (cdr (delay),
                       out-depends,
                       inputs,
                       'bad-or-delay)
elseif length (delay) = 2
then err-and (list (nlistp-or-err (type,
                                    'expected-literal-type),
                                phalf-delay-ok (car (delay),
                                                  type,
                                                  'low-to-high),
                                phalf-delay-ok (cadr (delay),
                                                  type,
                                                  'high-to-low),
                                nil-or-err (last-cdr (delay),
                                             'not-proper-list)),
              'bad-literal-delay)

```

```

    else pred-error ('unknown-delay, delay) endif,
    output-label (output))

```

EVENT: Enable pdelay-ok; name this event 'pdelay-ok-off'.

DEFINITION:

```

pdelay-errors (delays, outputs, types, out-depends, inputs)
=  if listp (delays)
    then cons (t-or-err (listp (outputs)  $\vee$  (outputs  $\in \mathbf{N}$ ),
                        'too-many-delays,
                        cons (length (delays), ' (extras))),
              cons (pdelay-ok (car (delays),
                              io-car (outputs),
                              ucar (types),
                              ucar (out-depends),
                              inputs),
                    pdelay-errors (cdr (delays),
                                    io-cdr (outputs),
                                    ucdr (types),
                                    ucdr (out-depends),
                                    inputs)))
    else list (nil-or-err (delays, 'not-proper-list),
              nlistp-or-err (outputs, 'outputs-without-delays)) endif

```

EVENT: Enable pdelays-errors; name this event 'pdelays-errors-off'.

DEFINITION:

```

pdelay-ok (delays, alist)
=  err-and (pdelay-errors (delays,
                          if unknownp (value ('outputs, alist)) then 1
                          else value ('outputs, alist) endif,
                          value ('output-types, alist),
                          value ('out-depends, alist),
                          value ('inputs, alist)),
            'delays)

```

EVENT: Enable pdelays-ok; name this event 'pdelays-ok-off'.

DEFINITION:

```

pdrive-ok (drive, output, type, out-depends, inputs)
=  label-error (if litatom (drive)  $\vee$  indexp (drive)
                then p-name-prop-value-ok (drive,
                                             type,

```

```

                                out-depends,
                                inputs,
                                'unknown-drive)
elseif listp (drive) ∧ (car (drive) = 'min)
then p-t-wire-args-ok (cdr (drive),
                        out-depends,
                        inputs,
                        'bad-min-drive)
elseif type ∈ '(boolp clk level parametric
                tri-state)
then t-or-err (drive ∈ ℕ,
               'bad-drive,
               list (list ('drive, drive),
                           list ('type, type)))
elseif type ∈ '(ttl ttl-tri-state)
then t-or-err (map (drive),
                 'bad-drive,
                 list (list ('drive, drive),
                             list ('type, type)))
elseif (drive ∈ ℕ) ∨ map (drive)
then t-or-err (type ≈ nil,
               'expected-literal-type,
               list (list ('drive, drive),
                           list ('type, type)))
else pred-error ('unknown-drive, drive) endif,
output-label (output))

```

EVENT: Enable pdrive-ok; name this event 'pdrive-ok-off'.

DEFINITION:

```

pdrives-errors (drives, outputs, types, out-depends, inputs)
= if listp (drives)
  then cons (t-or-err (listp (outputs) ∨ (outputs ∈ ℕ),
                       'too-many-drives,
                       cons (length (drives), ' (extras))),
             cons (pdrive-ok (car (drives),
                               io-car (outputs),
                               ucar (types),
                               ucar (out-depends),
                               inputs),
                   pdrives-errors (cdr (drives),
                                    io-cdr (outputs),
                                    ucdr (types),
                                    ucdr (out-depends)),

```

```

                                inputs)))
  else list (nil-or-err (drives, 'not-proper-list),
                nlistp-or-err (outputs, 'outputs-without-drives)) endif

```

EVENT: Enable pdrives-errors; name this event 'pdrives-errors-off'.

DEFINITION:

```

pdrives-ok (drives, alist)
=  err-and (pdrives-errors (drives,
                            if unknownp (value ('outputs, alist)) then 1
                            else value ('outputs, alist) endif,
                            value ('output-types, alist),
                            value ('out-depends, alist),
                            value ('inputs, alist)),
            'drives)

```

EVENT: Enable pdrives-ok; name this event 'pdrives-ok-off'.

DEFINITION:

```

ploadings-ok (loading, input, type)
=  label-error (t-or-err (if type ∈ ' (boolp clk free level
                                parametric tri-state)
                        then loading ∈ N
                        elseif type ∈ ' (ttl ttl-tri-state)
                        then pfp (loading)
                        else (loading ∈ N) ∨ pfp (loading) endif,
                        'bad-loading,
                        list (list ('loading, loading),
                                list ('input-type, type))),
                input-label (input))

```

EVENT: Enable ploadings-ok; name this event 'ploadings-ok-off'.

DEFINITION:

```

ploadings-errors (loadings, inputs, types)
=  if listp (loadings)
    then cons (t-or-err (listp (inputs) ∨ (inputs ∈ N),
                            'too-many-loadings,
                            cons (length (loadings), ' (extras))),
                cons (ploadings-ok (car (loadings), io-car (inputs), ucar (types)),
                        ploadings-errors (cdr (loadings),
                                            io-cdr (inputs),
                                            ucdr (types))))

```

```

else list (nil-or-err (loadings, 'not-proper-list),
  nlistp-or-err (inputs, 'inputs-without-loadings)) endif

```

EVENT: Enable ploadings-errors; name this event 'ploadings-errors-off'.

DEFINITION:

```

ploadings-ok (loadings, alist)
= err-and (ploadings-errors (loadings,
  if unknownp (value ('inputs, alist)) then 1
  else value ('inputs, alist) endif,
  value ('input-types, alist)),
  'loadings)

```

EVENT: Enable ploadings-ok; name this event 'ploadings-ok-off'.

DEFINITION:

```

plsi-name-ok (lsi-name-entry, alist)
= err-and (list (lsi-name-ok (if listp (lsi-name-entry)
  then car (lsi-name-entry)
  else lsi-name-entry endif,
  64),
  if (lsi-name-entry  $\simeq$  nil)
     $\vee$  unknownp (value ('inputs, alist)) then t
  else t-or-err (set-equal (cdr (lsi-name-entry),
    value ('inputs, alist)),
    'input-conflict,
    list (list ('inputs,
      value ('inputs, alist)),
      list ('lsi-inputs,
        cdr (lsi-name-entry)))) endif,
  if unknownp (value ('inputs, alist))
     $\vee$  unknownp (value ('outputs, alist)) then t
  else t-or-err (listp (intersection (value ('inputs,
    alist),
    value ('outputs,
    alist))))
  = is-head (unpack ('bd),
    unpack (if listp (lsi-name-entry)
      then car (lsi-name-entry)
      else lsi-name-entry endif)),
  'bidirect-conflict,
  list (list ('lsi-name,
    if listp (lsi-name-entry)

```

```

        then car (lsi-name-entry)
        else lsi-name-entry endif),
list ('io-signals,
      intersection (value ('inputs,
                           alist),
                     value ('outputs,
                           alist)))) endif),
'lsi-name)

```

EVENT: Enable plsi-name-ok; name this event 'plsi-name-ok-off'.

DEFINITION:

```

pout-depends-one-ok (out-depends, output, inputs)
= if unknownp (inputs)
  then name-list-ok (out-depends, output-label (output), f)
  else err-and (list (subset-or-err (out-depends,
                                     inputs,
                                     'unknown-output-dependencies),
                       no-duplicates-or-err (out-depends,
                                             'duplicates),
                       nil-or-err (last-cdr (out-depends),
                                    'not-proper-list)),
                output-label (output)) endif

```

EVENT: Enable pout-depends-one-ok; name this event 'pout-depends-one-ok-off'.

DEFINITION:

```

pout-depends-errors (out-depends, outputs, inputs)
= if listp (out-depends)
  then cons (t-or-err (listp (outputs)  $\vee$  ( $outputs \in \mathbf{N}$ ),
                       'too-many-out-depends,
                       cons (length (out-depends), ' (extras))),
            cons (pout-depends-one-ok (car (out-depends),
                                         io-car (outputs),
                                         inputs),
                  pout-depends-errors (cdr (out-depends),
                                         io-cdr (outputs),
                                         inputs)))
  else list (nil-or-err (out-depends, 'not-proper-list),
            nlistp-or-err (outputs,
                           'outputs-without-out-depends)) endif

```

EVENT: Enable pout-depends-errors; name this event 'pout-depends-errors-off'.

DEFINITION:

```
pout-depends-ok(out-depends, alist)
= err-and(pout-depends-errors(out-depends,
    if unknownp(value('outputs, alist))
    then 1
    else value('outputs, alist) endif,
    value('inputs, alist)),
    'out-depends)
```

EVENT: Enable pout-depends-ok; name this event 'pout-depends-ok-off'.

DEFINITION:

```
pstates-ok(states)
= if listp(states)
    then err-and(cons(t-or-err(listp(cdr(states)),
        'list-of-length-1-not-allowed,
        states),
        name-list-errors(states, f)),
        'states)
    elseif states = nil then t
    else t-or-err(name-okp(states),
        'states,
        list('bad-name, states)) endif
```

EVENT: Enable pstates-ok; name this event 'pstates-ok-off'.

DEFINITION:

```
addressed-state(address-bits, word-value)
= if address-bits  $\simeq$  0 then word-value
    else cons(addressed-state(address-bits - 1, word-value),
        addressed-state(address-bits - 1, word-value)) endif
```

EVENT: Enable addressed-state; name this event 'addressed-state-off'.

DEFINITION:

```
boolp-list-ok(lst, label)
= err-and(list(nlistp-or-err(remove-duplicates(delete*('boolp, lst)),
    'not-boolp),
    nil-or-err(last-cdr(lst), 'not-proper-list)),
    label)
```

EVENT: Enable boolp-list-ok; name this event 'boolp-list-ok-off'.

DEFINITION:

```
literal-state-type-ok (type)
=  if ramp (type) then boolp-list-ok (ram-guts (type), 'bad-ram-type)
    elseif romp (type)
    then boolp-list-ok (rom-guts (type), 'bad-rom-type)
    elseif stubp (type)
    then boolp-list-ok (stub-guts (type), 'bad-stub-type)
    else t-or-err (type = 'boolp, 'unknown-type, type) endif
```

EVENT: Enable literal-state-type-ok; name this event 'literal-state-type-ok-off'.

DEFINITION:

```
pmemory-word-call-ok (type)
=  if listp (type) ∧ (car (type) ∈ ' (ram rom stub))
    then if net-errorp (function-call-ok (type, 1))
        then function-call-ok (type, 1)
        else label-error (if caadr (type) = 'quote
                            then if net-errorp (function-call-ok (cadr (type),
                                                                    1))
                                then function-call-ok (cadr (type), 1)
                                else boolp-list-ok (cadadr (type),
                                                         'bad-memory-bit-types) endif
                            else pred-error ('unknown-memory-bits-call,
                                             cadr (type)) endif,
                                             cons ('bad, cons (car (type), ' (call)))) endif
    else pred-error ('not-memory-word-call, type) endif
```

EVENT: Enable pmemory-word-call-ok; name this event 'pmemory-word-call-ok-off'.

DEFINITION:

```
paddressed-state-call-ok (type)
=  if listp (type) ∧ (car (type) = 'addressed-state)
    then if net-errorp (function-call-ok (type, 2))
        then function-call-ok (type, 2)
        else err-and (list (t-or-err (cadr (type) ∈ N,
                                         'address-bits-not-number,
                                         cadr (type)),
                             pmemory-word-call-ok (cadadr (type))),
                        '(bad addressed-state call)) endif
    else pred-error ('not-addressed-state-call, type) endif
```

EVENT: Enable paddressed-state-call-ok; name this event 'paddressed-state-call-ok-off'.

DEFINITION:

```

one-pstate-type-ok(type)
=  if type  $\simeq$  nil then literal-state-type-ok(type)
   elseif car(type)  $\in$  '(ram rom stub)
   then pmemory-word-call-ok(type)
   elseif car(type) = 'addressed-state
   then paddressed-state-call-ok(type)
   else boolp-list-ok(type, 'bad-type-list) endif

```

EVENT: Enable one-pstate-type-ok; name this event 'one-pstate-type-ok-off'.

DEFINITION:

```

pstate-type-list-errors(types, fname)
=  if listp(types)
   then if (fname = 'mem-32x32)
         $\wedge$  (car(types)  $\in$  '(numberp number-listp))
        then pstate-type-list-errors(cdr(types), fname)
        else insert(one-pstate-type-ok(car(types)),
                    pstate-type-list-errors(cdr(types), fname)) endif
   else list(nil-or-err(types, 'not-proper-list)) endif

```

EVENT: Enable pstate-type-list-errors; name this event 'pstate-type-list-errors-off'.

DEFINITION:

```

pstate-type-ok(type, state, fname)
=  if (type  $\simeq$  nil)
     $\vee$  (car(type)  $\in$  '(ram rom stub addressed-state))
  then label-error(one-pstate-type-ok(type), state-label(state))
  else err-and(pstate-type-list-errors(type, fname),
              state-label(state)) endif

```

EVENT: Enable pstate-type-ok; name this event 'pstate-type-ok-off'.

DEFINITION:

```

pstate-types-errors(types, states, fname)
=  if listp(types)
   then cons(t-or-err(listp(states)  $\vee$  (states  $\in$   $\mathbf{N}$ ),
                    'too-many-state-types,
                    cons(length(types), 'extras)),
             cons(pstate-type-ok(car(types), io-car(states), fname),
                 pstate-types-errors(cdr(types), io-cdr(states), fname)))
   else list(nil-or-err(types, 'not-proper-list),
            nlistp-or-err(states, 'states-without-types)) endif

```

EVENT: Enable pstate-types-errors; name this event ‘pstate-types-errors-off’.

DEFINITION:

```
pstate-types-ok(state-types, alist)
=  err-and (if listp (value('states, alist))
              ∨ (value('states, alist) = nil)
            then pstate-types-errors(state-types,
                                     value('states, alist),
                                     value('primp-name, alist))
            else list(pstate-type-ok(state-types,
                                     value('states, alist),
                                     value('primp-name, alist))) endif,
              'state-types)
```

EVENT: Enable pstate-types-ok; name this event ‘pstate-types-ok-off’.

DEFINITION:

```
unknown-args(x, signals, states, arg-listp)
=  if arg-listp
    then if listp(x)
          then unknown-args(car(x), signals, states, f)
              ∪ unknown-args(cdr(x), signals, states, t)
          else nil endif
    elseif x ≈ nil
    then if (x ∈ signals) ∨ (x ∈ states) then nil
          elseif unknownp(signals) ∨ unknownp(states)
          then if name-okp(x) then nil
                else list(x) endif
          else list(x) endif
    elseif car(x) = 'quote then nil
    elseif litatom(car(x)) then unknown-args(cdr(x), signals, states, t)
    else nil endif
```

EVENT: Enable unknown-args; name this event ‘unknown-args-off’.

DEFINITION:

```
presult-form-errors(x, arg-listp)
=  if arg-listp
    then if listp(x)
          then if net-errorp(presult-form-errors(car(x), f))
                then cons(presult-form-errors(car(x), f),
                           presult-form-errors(cdr(x), t))
                else append(presult-form-errors(car(x), f),
```

```

                                presult-form-errors (cdr (x), t)) endif
elseif x = nil then nil
    else list (pred-error ('not-proper-list, x)) endif
elseif listp (x)
then if net-errorp (function-call-ok (x,
                                case on car (x):
                                case = quote
                                then 1
                                case = cons
                                then 2
                                otherwise UNKNOWN endcase))
    then function-call-ok (x,
                                case on car (x):
                                case = quote
                                then 1
                                case = cons
                                then 2
                                otherwise UNKNOWN endcase)
    elseif car (x) = 'quote then nil
    else presult-form-errors (cdr (x), t) endif
else nil endif

```

EVENT: Enable presult-form-errors; name this event 'presult-form-errors-off'.

DEFINITION:

```

presult-ok (result, signals, states, label)
= err-and (cons (nlistp-or-err (unknown-args (result, signals, states, f),
                                'unknown-args),
                presult-form-errors (list (result), t)),
            label)

```

EVENT: Enable presult-ok; name this event 'presult-ok-off'.

DEFINITION:

```

results-length (x)
= if x ≈ nil then 0
  elseif car (x) = 'cons then 1 + results-length (caddr (x))
  else 0 endif

```

EVENT: Enable results-length; name this event 'results-length-off'.

DEFINITION:

```

presults-errors (results, outputs, out-depends, inputs, states, label)

```

```

=  if results  $\simeq$  nil
    then list (pred-error ('unknown-dual-eval-form, results))
    elseif results = ''nil
    then list (nlistp-or-err (outputs,
                             cons ('missing, cons (label, '(values))))))
    elseif car (results) = 'cons
    then if net-errorp (function-call-ok (results, 2))
          then list (function-call-ok (results, 2))
          else cons (t-or-err (listp (outputs)  $\vee$  (outputs  $\in$   $\mathbf{N}$ ),
                               'too-many-values,
                               cons (results-length (results),
                                     ' (or more extras))),
                     cons (presult-ok (cadr (results),
                                         out-prop-signals (ucar (out-depends),
                                                                inputs),
                                                                states,
                                                                io-label (io-car (outputs), label)),
                           presults-errors (caddr (results),
                                                io-cdr (outputs),
                                                ucdr (out-depends),
                                                inputs,
                                                states,
                                                label))) endif
    elseif listp (outputs)  $\vee$  (outputs = nil)
    then list (pred-error (cons ('evaluation,
                                cons ('should,
                                        cons ('return,
                                                cons (length (outputs),
                                                            ' (results))))),
                                results),
              presult-ok (results,
                          inputs,
                          states,
                          list (label, 'list, outputs)))
    else list (presult-ok (results,
                          inputs,
                          states,
                          'unknown-outputs)) endif

```

EVENT: Enable presults-errors; name this event 'presults-errors-off'.

DEFINITION:

presults-ok (*results*, *alist*)

= err-and (presults-errors (*results*,

```

        if unknownp (value ('outputs, alist)) then 1
        else value ('outputs, alist) endif,
        value ('out-depends, alist),
        value ('inputs, alist),
        if unknownp (value ('states, alist))
        then value ('states, alist)
        else m-states-list (value ('states,
                                   alist)) endif,
        'output),
    'results)

```

EVENT: Enable results-ok; name this event 'results-ok-off'.

DEFINITION:

```

pnew-states-ok (new-states, alist)
=   err-and (if listp (value ('states, alist))
             ∨ (value ('states, alist) = nil)
             then results-errors (new-states,
                                   value ('states, alist),
                                   UNKNOWN,
                                   value ('inputs, alist),
                                   value ('states, alist),
                                   'state)
             elseif litatom (value ('states, alist))
             ∨ indexp (value ('states, alist))
             then list (presult-ok (new-states,
                                     value ('inputs, alist),
                                     list (value ('states, alist)),
                                     state-label (value ('states, alist))))
             else results-errors (new-states,
                                   1,
                                   UNKNOWN,
                                   value ('inputs, alist),
                                   UNKNOWN,
                                   'state) endif,
    'new-states)

```

EVENT: Enable pnew-states-ok; name this event 'pnew-states-ok-off'.

DEFINITION:

```

primitive-prop-ok (prop, alist, keys)
=   if listp (prop)
    then if car (prop) ∈ keys

```

```

then pred-error ('duplicate-property, prop)
elseif car(prop) = 'delays then pdelays-ok(cdr(prop), alist)
elseif car(prop) = 'drives then pdrives-ok(cdr(prop), alist)
elseif car(prop) = 'input-types
then pin-types-ok(cdr(prop), alist)
elseif car(prop) = 'inputs
then name-list-ok(cdr(prop), 'inputs, f)
elseif car(prop) = 'loadings
then ploadings-ok(cdr(prop), alist)
elseif car(prop) = 'lsi-name
then plsi-name-ok(cdr(prop), alist)
elseif car(prop) = 'new-states
then pnnew-states-ok(cdr(prop), alist)
elseif car(prop) = 'out-depends
then pout-depends-ok(cdr(prop), alist)
elseif car(prop) = 'output-types
then pout-types-ok(cdr(prop), alist)
elseif car(prop) = 'outputs
then name-list-ok(cdr(prop), 'outputs, f)
elseif car(prop) = 'results
then presults-ok(cdr(prop), alist)
elseif car(prop) = 'state-types
then pstate-types-ok(cdr(prop), alist)
elseif car(prop) = 'states then pstates-ok(cdr(prop))
elseif car(prop)
    ∈ ' (gates pads primitives transistors)
then t-or-err(cdr(prop) ∈ N,
    car(prop),
    list('value-not-number, cdr(prop)))
else pred-error('unknown-property, car(prop)) endif
else pred-error('not-key-value-pair, prop) endif

```

EVENT: Enable primitive-prop-ok; name this event 'primitive-prop-ok-off'.

DEFINITION:

```

primitive-prop-errors(props, alist, keys)
= if listp(props)
  then cons(primitive-prop-ok(car(props), alist, keys),
    primitive-prop-errors(cdr(props),
      alist,
      cons(caar(props), keys)))
  else list(nlistp-or-err(delete*('pads,
    if subset(' (new-states
      state-types

```

```

states),
set-diff (PRIMITIVE-PROPERTIES,
keys))
then set-diff (set-diff (PRIMITIVE-PROPERTIES,
keys),
'(new-states
state-types
states))
else set-diff (PRIMITIVE-PROPERTIES,
keys) endif,
'missing-properties),
nil-or-err (props, 'not-proper-list)) endif

```

EVENT: Enable primitive-prop-errors; name this event 'primitive-prop-errors-off'.

DEFINITION:

```

ok-name-list (pname, props)
= if pname-listp (value-or-unknown (pname, props))
then value-or-unknown (pname, props)
else UNKNOWN endif

```

EVENT: Enable ok-name-list; name this event 'ok-name-list-off'.

DEFINITION:

```

ok-states (props)
= if boundp ('states, props)
then if listp (value ('states, props))
    ∨ (value ('states, props) = nil)
then if pname-listp (value ('states, props))
    then value ('states, props)
    else UNKNOWN endif
elseif name-okp (value ('states, props))
then value ('states, props)
else UNKNOWN endif
else nil endif

```

EVENT: Enable ok-states; name this event 'ok-states-off'.

DEFINITION:

```

ok-out-depends (ods, inputs)
= if listp (ods)
then cons (if pname-listp (car (ods))

```

```

       $\wedge$  (unknownp (inputs)  $\vee$  subset (car (ods), inputs))
    then car (ods)
    else UNKNOWN endif,
    ok-out-depends (cdr (ods), inputs))
  else nil endif

```

EVENT: Enable ok-out-depends; name this event ‘ok-out-depends-off’.

DEFINITION:

```

ok-input-types (types)
= if listp (types)
  then cons (if car (types)  $\in$  ALL-INPUT-TYPES then car (types)
    else UNKNOWN endif,
    ok-input-types (cdr (types)))
  else nil endif

```

EVENT: Enable ok-input-types; name this event ‘ok-input-types-off’.

DEFINITION:

```

ok-output-types (out-types, out-depends, inputs)
= if listp (out-types)
  then cons (if if listp (car (out-types))
    then properp (car (out-types))
       $\wedge$  (length (car (out-types)) = 1)
       $\wedge$  if unknownp (out-prop-signals (ucar (out-depends),
        inputs))
        then name-okp (caar (out-types))
        else caar (out-types)
           $\in$  out-prop-signals (ucar (out-depends),
            inputs) endif
      else car (out-types)  $\in$  all-output-types (nil) endif
    then car (out-types)
    else UNKNOWN endif,
    ok-output-types (cdr (out-types), ucdr (out-depends), inputs))
  else nil endif

```

EVENT: Enable ok-output-types; name this event ‘ok-output-types-off’.

DEFINITION:

```

parent-props-alist (primp-entry)
= list (cons ('primp-name, car (primp-entry)),
  cons ('inputs, ok-name-list ('inputs, cdr (primp-entry))),
  cons ('outputs, ok-name-list ('outputs, cdr (primp-entry))),

```



```

cons ('states, ok-states (cdr (primp-entry))),
cons ('out-depends,
      ok-out-depends (value ('out-depends, cdr (primp-entry)),
                          ok-name-list ('inputs, cdr (primp-entry)))),
cons ('input-types,
      ok-input-types (value ('input-types, cdr (primp-entry)))),
cons ('output-types,
      ok-output-types (value ('output-types, cdr (primp-entry)),
                           ok-out-depends (value ('out-depends,
                                                    cdr (primp-entry)),
                                                ok-name-list ('inputs,
                                                                cdr (primp-entry)))),
                           ok-name-list ('inputs, cdr (primp-entry))))

```

EVENT: Enable parent-props-alist; name this event 'parent-props-alist-off'.

DEFINITION:

```

primitive-ok (p)
=  if listp (p)
    then err-and (cons (t-or-err (name-okp (car (p)),
                                   'bad-primitive-name,
                                   car (p)),
                        primitive-prop-errors (cdr (p),
                                                parent-props-alist (p),
                                                nil)),
                  list ('primitive, car (p)))
    else pred-error ('not-a-primitive, p) endif

```

EVENT: Enable primitive-ok; name this event 'primitive-ok-off'.

DEFINITION:

```

primp-database-errors (d)
=  if d  $\simeq$  nil then nil
    else cons (primitive-ok (car (d)), primp-database-errors (cdr (d))) endif

```

EVENT: Enable primp-database-errors; name this event 'primp-database-errors-off'.

DEFINITION:

```

PRIMP-DATABASE-PREDICATE
=  err-and (primp-database-errors (PRIMP-DATABASE),
            'primp-database-errors)

```

EVENT: Enable primp-database-predicate; name this event ‘primp-database-predicate-off’.

EVENT: Disable boundp; name this event ‘boundp-on’.

DEFINITION:

```

subnet0(flg, x0, netlist, used-modules, result)
=  case on flg:
    case = 0
    then if lookup-module(x0, result)  $\vee$  (x0  $\in$  used-modules)
        then result
        elseif listp(lookup-module(x0, netlist))
        then cons(lookup-module(x0, netlist),
            subnet0(1,
                module-occurrences(lookup-module(x0, netlist)),
                netlist,
                cons(x0, used-modules),
                result))
        else result endif
    case = 1
    then if x0  $\simeq$  nil then result
    else subnet0(1,
        cdr(x0),
        netlist,
        used-modules,
        subnet0(0,
            occ-function(car(x0)),
            netlist,
            used-modules,
            result)) endif
    otherwise f endcase

```

EVENT: Enable boundp; name this event ‘boundp-off1’.

EVENT: Enable subnet0; name this event ‘subnet0-off’.

DEFINITION:

```

subnet(flg, x0, netlist) = subnet0(flg, x0, netlist, nil, nil)

```

EVENT: Enable subnet; name this event ‘subnet-off’.

DEFINITION:

```

delete-null-entries(database)
=  if listp(database)
    then if listp(cdar(database))
        then cons(car(database), delete-null-entries(cdr(database)))
        else delete-null-entries(cdr(database)) endif
    else nil endif

```

EVENT: Enable delete-null-entries; name this event ‘delete-null-entries-off’.

DEFINITION:

```

primitive-count(flag, x0, type, netlist)
=  case on flag:
    case = 0
    then if primp(x0) then primp2(x0, type)
        elseif lookup-module(x0, netlist)
            then if net-errorp(netlist-properties(subnet(0, x0, netlist),
                                                    list(type),
                                                    t))
                then netlist-properties(subnet(0, x0, netlist), list(type), t)
            else value(type,
                    value(x0,
                        netlist-properties(subnet(0,
                                                x0,
                                                netlist),
                                                list(type),
                                                t))) endif
        else pred-error('unknown-module, x0) endif
    case = 1
    then value(type,
        occurrence-data(required-props(list(type)),
            x0,
            nil,
            nil,
            nil,
            module-database(subnet(1, x0, netlist),
                required-props(list(type),
                    t))))
    otherwise pred-error('unknown-flag,
        list(list('flag, flag), list('x0, x0))) endcase

```

EVENT: Enable primitive-count; name this event ‘primitive-count-off’.

DEFINITION:

```

out-depends(flag, x0, x1, x2, netlist)
=  case on flag:
    case = 0
    then if primp(x0)
        then list-union-values(primp2(x0, 'out-depends),
                                pairlist(primp2(x0, 'inputs), x1))
        elseif lookup-module(x0, netlist)
        then if net-errorp(netlist-properties(subnet(0, x0, netlist),
                                                '(out-depends inputs),
                                                t))
            then netlist-properties(subnet(0, x0, netlist),
                                    '(out-depends inputs),
                                    t)
            else list-union-values(value('out-depends,
                                         value(x0,
                                               netlist-properties(subnet(0,
                                                                           x0,
                                                                           netlist),
                                                                           '(out-depends
                                                                           inputs),
                                                                           t))),
                                pairlist(value('inputs,
                                                value(x0,
                                                      netlist-properties(subnet(0,
                                                                           x0,
                                                                           netlist),
                                                                           '(out-depends
                                                                           inputs),
                                                                           t))),
                                pairlist(value('inputs,
                                                value(x0,
                                                      netlist-properties(subnet(0,
                                                                           x0,
                                                                           netlist),
                                                                           '(out-depends
                                                                           inputs),
                                                                           t))),
                                x1)) endif
        else pred-error('unknown-module, x0) endif
    case = 1
    then value('out-depends,
               occurrence-data(required-props('out-depends),
                               x0,
                               x1,
                               mark-io-outs(x2, intersection(x1, x2)),
                               intersection(x1, x2),
                               module-database(subnet(1, x0, netlist),
                                                  required-props('out-depends),
                                                  t)))
    otherwise pred-error('unknown-flag,
                          list(list('flag, flag),
                                list('x0, x0),

```

```

list ('x1, x1),
list ('x2, x2))) endcase

```

EVENT: Enable out-depends; name this event 'out-depends-off'.

DEFINITION:

```

dependency-table (netlist)
= netlist-properties (netlist, ' (out-depends), t)

```

EVENT: Enable dependency-table; name this event 'dependency-table-off'.

DEFINITION:

```

output-dependencies (module-name, args, netlist)
= if net-errorp (if primp (module-name)
    then length (primp2 (module-name, 'inputs))
    elseif lookup-module (module-name, netlist)
    then length (module-inputs (lookup-module (module-name,
                                                netlist)))
    else pred-error ('unknown-module, module-name) endif)
then if primp (module-name)
    then length (primp2 (module-name, 'inputs))
    elseif lookup-module (module-name, netlist)
    then length (module-inputs (lookup-module (module-name, netlist)))
    else pred-error ('unknown-module, module-name) endif
elseif length (args)
    = if primp (module-name)
        then length (primp2 (module-name, 'inputs))
        elseif lookup-module (module-name, netlist)
        then length (module-inputs (lookup-module (module-name,
                                                    netlist)))
        else pred-error ('unknown-module, module-name) endif
    then if net-errorp (out-depends (0, module-name, args, nil, netlist))
        then out-depends (0, module-name, args, nil, netlist)
        else list-list-sort (out-depends (0,
                                          module-name,
                                          args,
                                          nil,
                                          netlist)) endif
    else pred-error ('wrong-number-of-args,
        list (list (module-name,
                    'expects,
                    if primp (module-name)
                    then length (primp2 (module-name,
```

```

                                'inputs))
elseif lookup-module (module-name,
                      netlist)
then length (module-inputs (lookup-module (module-name,
                                           netlist)))

else pred-error ('unknown-module,
                 module-name) endif,
list ('got, length (args), args))) endif

```

EVENT: Enable output-dependencies; name this event 'output-dependencies-off'.

DEFINITION:

```

io-types-collector (types, alist)
= if listp (types)
  then cons (if listp (car (types)) then value (caar (types), alist)
            else car (types) endif,
            io-types-collector (cdr (types), alist))
  else nil endif

```

EVENT: Enable io-types-collector; name this event 'io-types-collector-off'.

DEFINITION:

```

io-types (flag, x0, x1, x2, netlist)
= case on flag:
  case = 0
  then if primp (x0)
        then list (primp-lookup (x0, 'input-types),
                    cons ('output-types,
                          io-types-collector (primp2 (x0, 'output-types),
                                                  pairlist (primp2 (x0, 'inputs),
                                                                x1))))
        elseif lookup-module (x0, netlist)
        then if net-errorp (netlist-properties (subnet (0, x0, netlist),
                                                         '(input-types
                                                           output-types inputs),
                                                         t))
              then netlist-properties (subnet (0, x0, netlist),
                                         '(input-types output-types
                                           inputs),
                                         t)
              else list (alist-entry ('input-types,
                                       value (x0,

```

```

netlist-properties (subnet (0,
                             x0,
                             netlist),
                    ' (input-types
                      output-types
                      inputs),
                    t))),
cons ('output-types,
     io-types-collector (value ('output-types,
                                value (x0,
                                        netlist-properties (subnet (0,
                                                                    x0,
                                                                    netlist),
                                                                    ' (input-types
                                                                      output-types
                                                                      inputs),
                                                                      t))),
                                pairlist (value ('inputs,
                                                value (x0,
                                                        netlist-properties (subnet (0,
                                                                                    x0,
                                                                                    netlist),
                                                                                    ' (input-types
                                                                                      output-types
                                                                                      inputs),
                                                                                      t))),
                                                x1)))) endif
else pred-error ('unknown-module, x0) endif
case = 1
then list (alist-entry ('input-types,
                        occurrence-data (required-props (' (input-types
                                                            output-types)),
                        x0,
                        x1,
                        mark-io-outs (x2,
                                     intersection (x1,
                                                    x2))),
                        intersection (x1, x2),
                        module-database (subnet (1,
                                                  x0,
                                                  netlist),
                                          required-props (' (input-types
                                                                output-types)),
                                          t))),

```

```

alist-entry ('output-types,
             occurrence-data (required-props ('(input-types
                                                output-types)),
                               x0,
                               x1,
                               mark-io-outs (x2,
                                              intersection (x1,
                                                            x2)),
                               intersection (x1, x2),
                               module-database (subnet (1,
                                                         x0,
                                                         netlist),
                                                  required-props ('(input-types
                                                                    output-types)),
                                                  t))))
otherwise pred-error ('unknown-flag,
                       list (list ('flag, flag),
                                  list ('x0, x0),
                                  list ('x1, x1),
                                  list ('x2, x2))) endcase

```

EVENT: Enable io-types; name this event 'io-types-off'.

DEFINITION:

```

netlist-type-table (netlist)
= netlist-properties (netlist, '(input-types output-types), t)

```

EVENT: Enable netlist-type-table; name this event 'netlist-type-table-off'.

DEFINITION:

```

arg-types-matchp (actuals, formals)
= if (actuals  $\simeq$  nil)  $\vee$  (formals  $\simeq$  nil) then actuals = formals
  elseif unknownp (car (actuals))  $\vee$  unknownp (car (formals)) then f
  elseif types-compatiblep (car (actuals), car (formals))
  then arg-types-matchp (cdr (actuals), cdr (formals))
  else f endif

```

EVENT: Enable arg-types-matchp; name this event 'arg-types-matchp-off'.

DEFINITION:

```

arg-types-okp (fn, arg-types, netlist)
= if boundp ('input-types, io-types (0, fn, arg-types, nil, netlist))
  then arg-types-matchp (arg-types,

```



```

                                value('input-types,
                                      io-types(0, fn, arg-types, nil, netlist)))
else f endif

```

EVENT: Enable arg-types-okp; name this event 'arg-types-okp-off'.

DEFINITION:

```

state-type-requirement(flag, x0, netlist)
= case on flag:
  case = 0
  then if primp(x0)
    then if primp-lookup(x0, 'state-types)
      then primp2(x0, 'state-types)
      else nil endif
    elseif lookup-module(x0, netlist)
    then if net-errorp(netlist-properties(subnet(0, x0, netlist),
                                             '(state-types),
                                             t))
      then netlist-properties(subnet(0, x0, netlist),
                               '(state-types),
                               t)
      elseif boundp('state-types,
                    value(x0,
                          netlist-properties(subnet(0, x0, netlist),
                                              '(state-types),
                                              t)))
      then value('state-types,
                  value(x0,
                        netlist-properties(subnet(0, x0, netlist),
                                          '(state-types),
                                          t)))
      else nil endif
    else pred-error('unknown-module, x0) endif
  case = 1
  then value('state-types,
              occurrence-data(required-props('state-types),
                              x0,
                              nil,
                              nil,
                              nil,
                              module-database(subnet(1, x0, netlist),
                                                  required-props('state-types),
                                                  t)))
  otherwise pred-error('unknown-flag,

```

list (list ('flag, flag), list ('x0, x0))) **endcase**

EVENT: Enable state-type-requirement; name this event 'state-type-requirement-off'.

DEFINITION:

```
netlist-state-types (netlist)
=  if net-errorp (netlist-properties (netlist, '(state-types), t))
    then netlist-properties (netlist, '(state-types), t)
    else delete-null-entries (netlist-properties (netlist,
                                                    '(state-types),
                                                    t)) endif
```

EVENT: Enable netlist-state-types; name this event 'netlist-state-types-off'.

DEFINITION:

type-count (*type*, *word-type*) = (count (*type*) + count (*word-type*))

THEOREM: type-count-lessp1

listp (*type*) → (type-count (car (*type*), caddr (*type*)) < type-count (*type*, *x*))

THEOREM: type-count-lessp2

listp (*type*) → (type-count (cadr (*type*), f) < type-count (*type*, *x*))

THEOREM: type-count-lessp3

(count (*t2*) < count (*t1*))
→ ((type-count (*wtype*, *t2*) < type-count (*t1*, *wtype*)) = t)

EVENT: Enable type-count; name this event 'type-count-off'.

DEFINITION:

state-okp-0 (*state*, *type*, *address-bits*, *word-type*)

```
=  case on type:
    case = nil
    then state = nil
    case = boolp
    then boolp (state)
    otherwise if listp (type)
        then case on car (type):
            case = addressed-state
            then state-okp-0 (state,
                             car (type),
                             cadr (type),
                             caddr (type))
```

```

case = ram
  then ramp (state)
     $\wedge$  state-okp-0 (ram-guts (state),
                      cadr (type),
                      0,
                      f)

case = rom
  then romp (state)
     $\wedge$  state-okp-0 (rom-guts (state),
                      cadr (type),
                      0,
                      f)

case = stub
  then stubp (state)
     $\wedge$  state-okp-0 (stub-guts (state),
                      cadr (type),
                      0,
                      f)

case = quote
  then state-okp-0 (state, cadr (type), 0, f)
otherwise if listp (state)
  then state-okp-0 (car (state), car (type), 0, f)
     $\wedge$  state-okp-0 (cdr (state),
                      cdr (type),
                      0,
                      f)
    else f endif endcase
elseif type = 'addressed-state'
then if address-bits  $\simeq$  0
  then state-okp-0 (state, word-type, 0, f)
  elseif listp (state)
  then state-okp-0 (car (state),
                    type,
                    address-bits - 1,
                    word-type)
     $\wedge$  state-okp-0 (cdr (state),
                    type,
                    address-bits - 1,
                    word-type)
    else f endif
elseif type = 'numberp' then state  $\in$  N
elseif type = 'number-listp'
then if state  $\simeq$  nil then state = nil
  elseif car (state)  $\in$  N

```

```

        then state-okp-0 (cdr (state), type, 0, f)
      else f endif
    elseif ramp (type)
    then if ramp (state)
      then state-okp-0 (ram-guts (state), ram-guts (type), 0, f)
      else f endif
    elseif romp (type)
    then if romp (state)
      then state-okp-0 (rom-guts (state), rom-guts (type), 0, f)
      else f endif
    elseif stubp (type)
    then if stubp (state)
      then state-okp-0 (stub-guts (state), stub-guts (type), 0, f)
      else f endif
    else f endif endcase

```

EVENT: Enable state-okp-0; name this event 'state-okp-0-off'.

EVENT: Enable type-count-lessp1; name this event 'type-count-lessp1-off'.

EVENT: Enable type-count-lessp2; name this event 'type-count-lessp2-off'.

EVENT: Enable type-count-lessp3; name this event 'type-count-lessp3-off'.

DEFINITION:

```

state-okp (fn, state, netlist)
= state-okp-0 (state, state-type-requirement (0, fn, netlist), 0, f)

```

EVENT: Enable state-okp; name this event 'state-okp-off'.

DEFINITION:

```

fix-dependent-drs (drs, alist)
= if listp (drs)
  then cons (if signal-namep (car (drs)) then value (car (drs), alist)
    elseif listp (car (drs)) ^ (caar (drs) = 'min)
    then cons ('min, fix-dependent-drs (cdar (drs), alist))
    else car (drs) endif,
    fix-dependent-drs (cdr (drs), alist))
  else nil endif

```

EVENT: Enable fix-dependent-drs; name this event 'fix-dependent-drs-off'.


```

(VALUE 'OUTPUTS
  (VALUE X0
    (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
      '(LOADINGS DRIVES INPUTS OUTPUTS)
      T)))
  X2))))))
(T (PRED-ERROR 'UNKNOWN-MODULE X0)))
(1
  (LIST
    (ALIST-ENTRY 'LOADINGS
      (OCCURRENCE-DATA (REQUIRED-PROPS '(LOADINGS DRIVES))
        X0 X1
        (MARK-IO-OUTS X2 (INTERSECTION X1 X2))
        (INTERSECTION X1 X2)
        (MODULE-DATABASE (SUBNET 1 X0 NETLIST)
          (REQUIRED-PROPS '(LOADINGS DRIVES))
          T)))
    (ALIST-ENTRY 'DRIVES
      (OCCURRENCE-DATA (REQUIRED-PROPS '(LOADINGS DRIVES))
        X0 X1
        (MARK-IO-OUTS X2 (INTERSECTION X1 X2))
        (INTERSECTION X1 X2)
        (MODULE-DATABASE (SUBNET 1 X0 NETLIST)
          (REQUIRED-PROPS '(LOADINGS DRIVES))
          T))))
    (OTHERWISE (PRED-ERROR 'UNKNOWN-FLAG
      (LIST (LIST 'FLG FLG)
        (LIST 'X0 X0)
        (LIST 'X1 X1)
        (LIST 'X2 X2))))))

```

EVENT: Enable loadings-and-drives; name this event 'loadings-and-drives-off'.

DEFINITION:

```

netlist-loadings-and-drives(netlist)
= netlist-properties(netlist, '(loadings drives), t)

```

EVENT: Enable netlist-loadings-and-drives; name this event 'netlist-loadings-and-drives-off'.

DEFINITION: $ao2\&(netlist) = t$

EVENT: Enable ao2&; name this event 'ao2&-off'.

DEFINITION: $\text{AO2\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{ao2\$value}$

$\text{ao2\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ao2'}, \text{list}(a, b, c, d), state, netlist)$
 $\quad = \text{list}(\text{f-nor}(\text{f-and}(a, b), \text{f-and}(c, d))))$

EVENT: Enable $\text{ao2\$value}$; name this event ' $\text{ao2\$value-off}$ '.

DEFINITION: $\text{ao4\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{ao4\&}$; name this event ' $\text{ao4\&-off}$ '.

DEFINITION: $\text{AO4\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{ao4\$value}$

$\text{ao4\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ao4'}, \text{list}(a, b, c, d), state, netlist)$
 $\quad = \text{list}(\text{f-nand}(\text{f-or}(a, b), \text{f-or}(c, d))))$

EVENT: Enable $\text{ao4\$value}$; name this event ' $\text{ao4\$value-off}$ '.

DEFINITION: $\text{ao6\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{ao6\&}$; name this event ' $\text{ao6\&-off}$ '.

DEFINITION: $\text{AO6\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{ao6\$value}$

$\text{ao6\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ao6'}, \text{list}(a, b, c), state, netlist)$
 $\quad = \text{list}(\text{f-nor}(\text{f-and}(a, b), c)))$

EVENT: Enable $\text{ao6\$value}$; name this event ' $\text{ao6\$value-off}$ '.

DEFINITION: $\text{ao7\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{ao7\&}$; name this event ' $\text{ao7\&-off}$ '.

DEFINITION: $\text{AO7\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{ao7\$value}$

$\text{ao7\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ao7'}, \text{list}(a, b, c), state, netlist)$
 $\quad = \text{list}(\text{f-nand}(\text{f-or}(a, b), c)))$

EVENT: Enable $\text{ao7\$value}$; name this event ' $\text{ao7\$value-off}$ '.

DEFINITION: $\text{b-and\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{b-and\&}$; name this event ' $\text{b-and\&-off}$ '.

DEFINITION: $\text{B-AND\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{b-and\$value}$

$\text{b-and\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-and'}, \text{list}(a, b), state, netlist) = \text{list}(\text{f-and}(a, b)))$

EVENT: Enable $\text{b-and\$value}$; name this event ' $\text{b-and\$value-off}$ '.

DEFINITION: $\text{b-and3\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{b-and3\&}$; name this event ' $\text{b-and3\&-off}$ '.

DEFINITION: $\text{B-AND3\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{b-and3\$value}$

$\text{b-and3\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-and3'}, \text{list}(a, b, c), state, netlist) = \text{list}(\text{f-and3}(a, b, c)))$

EVENT: Enable $\text{b-and3\$value}$; name this event ' $\text{b-and3\$value-off}$ '.

DEFINITION: $\text{b-and4\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{b-and4\&}$; name this event ' $\text{b-and4\&-off}$ '.

DEFINITION: $\text{B-AND4\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{b-and4\$value}$

$\text{b-and4\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-and4'}, \text{list}(a, b, c, d), state, netlist) = \text{list}(\text{f-and4}(a, b, c, d)))$

EVENT: Enable $\text{b-and4\$value}$; name this event ' $\text{b-and4\$value-off}$ '.

DEFINITION: $\text{b-equiv\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{b-equiv\&}$; name this event ' $\text{b-equiv\&-off}$ '.

DEFINITION: $\text{B-EQV}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-equiv}\$value$
 $\text{b-equiv}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-equiv'}, \text{list}(a, b), state, netlist)$
 $= \text{list}(\text{f-equiv}(a, b)))$

EVENT: Enable $\text{b-equiv}\$value$; name this event 'b-equiv\$value-off'.

DEFINITION: $\text{b-equiv3}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-equiv3}\&$; name this event 'b-equiv3&-off'.

DEFINITION: $\text{B-EQV3}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-equiv3}\$value$
 $\text{b-equiv3}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-equiv3'}, \text{list}(a, b, c), state, netlist)$
 $= \text{list}(\text{f-equiv3}(a, b, c)))$

EVENT: Enable $\text{b-equiv3}\$value$; name this event 'b-equiv3\$value-off'.

DEFINITION: $\text{b-if}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-if}\&$; name this event 'b-if&-off'.

DEFINITION: $\text{B-IF}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-if}\$value$
 $\text{b-if}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-if'}, \text{list}(s, a, b), state, netlist) = \text{list}(\text{f-if}(s, a, b)))$

EVENT: Enable $\text{b-if}\$value$; name this event 'b-if\$value-off'.

DEFINITION: $\text{b-nand}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-nand}\&$; name this event 'b-nand&-off'.

DEFINITION: $\text{B-NAND}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-nand}\$value$
 $\text{b-nand}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-nand'}, \text{list}(a, b), state, netlist)$
 $= \text{list}(\text{f-nand}(a, b)))$

EVENT: Enable b-nand\$value; name this event 'b-nand\$value-off'.

DEFINITION: $\text{b-nand3\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nand3\&; name this event 'b-nand3\&-off'.

DEFINITION: $\text{B-NAND3\$NETLIST} = \mathbf{nil}$

THEOREM: b-nand3\$value

$\text{b-nand3\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-nand3'}, \text{list}(a, b, c), state, netlist)$
 $= \text{list}(\text{f-nand3}(a, b, c)))$

EVENT: Enable b-nand3\$value; name this event 'b-nand3\$value-off'.

DEFINITION: $\text{b-nand4\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nand4\&; name this event 'b-nand4\&-off'.

DEFINITION: $\text{B-NAND4\$NETLIST} = \mathbf{nil}$

THEOREM: b-nand4\$value

$\text{b-nand4\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-nand4'}, \text{list}(a, b, c, d), state, netlist)$
 $= \text{list}(\text{f-nand4}(a, b, c, d)))$

EVENT: Enable b-nand4\$value; name this event 'b-nand4\$value-off'.

DEFINITION: $\text{b-nand5\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nand5\&; name this event 'b-nand5\&-off'.

DEFINITION: $\text{B-NAND5\$NETLIST} = \mathbf{nil}$

THEOREM: b-nand5\$value

$\text{b-nand5\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-nand5'}, \text{list}(a, b, c, d, e), state, netlist)$
 $= \text{list}(\text{f-nand5}(a, b, c, d, e)))$

EVENT: Enable b-nand5\$value; name this event 'b-nand5\$value-off'.

DEFINITION: $\text{b-nand6\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nand6&; name this event 'b-nand6&-off'.

DEFINITION: B-NAND6\$NETLIST = **nil**

THEOREM: b-nand6\$value

b-nand6& (*netlist*)

→ (dual-eval (0, 'b-nand6, list (*a*, *b*, *c*, *d*, *e*, *g*), *state*, *netlist*)
= list (f-nand6 (*a*, *b*, *c*, *d*, *e*, *g*)))

EVENT: Enable b-nand6\$value; name this event 'b-nand6\$value-off'.

DEFINITION: b-nand8& (*netlist*) = **t**

EVENT: Enable b-nand8&; name this event 'b-nand8&-off'.

DEFINITION: B-NAND8\$NETLIST = **nil**

THEOREM: b-nand8\$value

b-nand8& (*netlist*)

→ (dual-eval (0, 'b-nand8, list (*a*, *b*, *c*, *d*, *e*, *g*, *h*, *i*), *state*, *netlist*)
= list (f-nand8 (*a*, *b*, *c*, *d*, *e*, *g*, *h*, *i*)))

EVENT: Enable b-nand8\$value; name this event 'b-nand8\$value-off'.

DEFINITION: b-nbuf& (*netlist*) = **t**

EVENT: Enable b-nbuf&; name this event 'b-nbuf&-off'.

DEFINITION: B-NBUF\$NETLIST = **nil**

THEOREM: b-nbuf\$value

b-nbuf& (*netlist*)

→ (dual-eval (0, 'b-nbuf, list (*a*), *state*, *netlist*)
= list (f-not (*a*), f-buf (*a*)))

EVENT: Enable b-nbuf\$value; name this event 'b-nbuf\$value-off'.

DEFINITION: b-nor& (*netlist*) = **t**

EVENT: Enable b-nor&; name this event 'b-nor&-off'.

DEFINITION: B-NOR\$NETLIST = **nil**

THEOREM: $\text{b-nor}\$value$
 $\text{b-nor}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-nor'}, \text{list}(a, b), state, netlist) = \text{list}(\text{f-nor}(a, b)))$

EVENT: Enable $\text{b-nor}\$value$; name this event ' $\text{b-nor}\$value\text{-off}$ '.

DEFINITION: $\text{b-nor3}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-nor3}\&$; name this event ' $\text{b-nor3}\&\text{-off}$ '.

DEFINITION: $\text{B-NOR3}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-nor3}\$value$
 $\text{b-nor3}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-nor3'}, \text{list}(a, b, c), state, netlist)$
 $= \text{list}(\text{f-nor3}(a, b, c)))$

EVENT: Enable $\text{b-nor3}\$value$; name this event ' $\text{b-nor3}\$value\text{-off}$ '.

DEFINITION: $\text{b-nor4}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-nor4}\&$; name this event ' $\text{b-nor4}\&\text{-off}$ '.

DEFINITION: $\text{B-NOR4}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-nor4}\$value$
 $\text{b-nor4}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-nor4'}, \text{list}(a, b, c, d), state, netlist)$
 $= \text{list}(\text{f-nor4}(a, b, c, d)))$

EVENT: Enable $\text{b-nor4}\$value$; name this event ' $\text{b-nor4}\$value\text{-off}$ '.

DEFINITION: $\text{b-nor5}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-nor5}\&$; name this event ' $\text{b-nor5}\&\text{-off}$ '.

DEFINITION: $\text{B-NOR5}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-nor5}\$value$
 $\text{b-nor5}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-nor5'}, \text{list}(a, b, c, d, e), state, netlist)$
 $= \text{list}(\text{f-nor5}(a, b, c, d, e)))$

EVENT: Enable b-nor5\$value; name this event 'b-nor5\$value-off'.

DEFINITION: $\text{b-nor6\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nor6\&; name this event 'b-nor6\&-off'.

DEFINITION: $\text{B-NOR6\$NETLIST} = \mathbf{nil}$

THEOREM: b-nor6\$value

$\text{b-nor6\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-nor6'}, \text{list}(a, b, c, d, e, g), state, netlist)$
 $= \text{list}(\text{f-nor6}(a, b, c, d, e, g)))$

EVENT: Enable b-nor6\$value; name this event 'b-nor6\$value-off'.

DEFINITION: $\text{b-nor8\&}(netlist) = \mathbf{t}$

EVENT: Enable b-nor8\&; name this event 'b-nor8\&-off'.

DEFINITION: $\text{B-NOR8\$NETLIST} = \mathbf{nil}$

THEOREM: b-nor8\$value

$\text{b-nor8\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-nor8'}, \text{list}(a, b, c, d, e, g, h, i), state, netlist)$
 $= \text{list}(\text{f-nor8}(a, b, c, d, e, g, h, i)))$

EVENT: Enable b-nor8\$value; name this event 'b-nor8\$value-off'.

DEFINITION: $\text{b-not\&}(netlist) = \mathbf{t}$

EVENT: Enable b-not\&; name this event 'b-not\&-off'.

DEFINITION: $\text{B-NOT\$NETLIST} = \mathbf{nil}$

THEOREM: b-not\$value

$\text{b-not\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-not'}, \text{list}(a), state, netlist) = \text{list}(\text{f-not}(a)))$

EVENT: Enable b-not\$value; name this event 'b-not\$value-off'.

DEFINITION: $\text{b-not-b4ip\&}(netlist) = \mathbf{t}$

EVENT: Enable b-not-b4ip\&; name this event 'b-not-b4ip\&-off'.

DEFINITION: $\text{B-NOT-B4IP}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-not-b4ip}\$value$
 $\text{b-not-b4ip}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-not-b4ip'}, \text{list}(a), state, netlist)$
 $= \text{list}(\text{f-not}(a)))$

EVENT: Enable $\text{b-not-b4ip}\$value$; name this event ' $\text{b-not-b4ip}\$value\text{-off}$ '.

DEFINITION: $\text{b-not-ivap}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-not-ivap}\&$; name this event ' $\text{b-not-ivap}\&\text{-off}$ '.

DEFINITION: $\text{B-NOT-IVAP}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-not-ivap}\$value$
 $\text{b-not-ivap}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-not-ivap'}, \text{list}(a), state, netlist)$
 $= \text{list}(\text{f-not}(a)))$

EVENT: Enable $\text{b-not-ivap}\$value$; name this event ' $\text{b-not-ivap}\$value\text{-off}$ '.

DEFINITION: $\text{b-or}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-or}\&$; name this event ' $\text{b-or}\&\text{-off}$ '.

DEFINITION: $\text{B-OR}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-or}\$value$
 $\text{b-or}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-or'}, \text{list}(a, b), state, netlist) = \text{list}(\text{f-or}(a, b)))$

EVENT: Enable $\text{b-or}\$value$; name this event ' $\text{b-or}\$value\text{-off}$ '.

DEFINITION: $\text{b-or3}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{b-or3}\&$; name this event ' $\text{b-or3}\&\text{-off}$ '.

DEFINITION: $\text{B-OR3}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{b-or3}\$value$
 $\text{b-or3}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'b-or3'}, \text{list}(a, b, c), state, netlist)$
 $= \text{list}(\text{f-or3}(a, b, c)))$

EVENT: Enable b-or3\$value; name this event ‘b-or3\$value-off’.

DEFINITION: $\text{b-or4\&}(netlist) = \mathbf{t}$

EVENT: Enable b-or4\&; name this event ‘b-or4\&-off’.

DEFINITION: $\text{B-OR4\$NETLIST} = \mathbf{nil}$

THEOREM: b-or4\$value

$\text{b-or4\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-or4'}, \text{list}(a, b, c, d), state, netlist)$
 $= \text{list}(\text{f-or4}(a, b, c, d)))$

EVENT: Enable b-or4\$value; name this event ‘b-or4\$value-off’.

DEFINITION: $\text{b-xor\&}(netlist) = \mathbf{t}$

EVENT: Enable b-xor\&; name this event ‘b-xor\&-off’.

DEFINITION: $\text{B-XOR\$NETLIST} = \mathbf{nil}$

THEOREM: b-xor\$value

$\text{b-xor\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-xor'}, \text{list}(a, b), state, netlist) = \text{list}(\text{f-xor}(a, b)))$

EVENT: Enable b-xor\$value; name this event ‘b-xor\$value-off’.

DEFINITION: $\text{b-xor3\&}(netlist) = \mathbf{t}$

EVENT: Enable b-xor3\&; name this event ‘b-xor3\&-off’.

DEFINITION: $\text{B-XOR3\$NETLIST} = \mathbf{nil}$

THEOREM: b-xor3\$value

$\text{b-xor3\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'b-xor3'}, \text{list}(a, b, c), state, netlist)$
 $= \text{list}(\text{f-xor3}(a, b, c)))$

EVENT: Enable b-xor3\$value; name this event ‘b-xor3\$value-off’.

DEFINITION: $\text{del2\&}(netlist) = \mathbf{t}$

EVENT: Enable del2\&; name this event ‘del2\&-off’.

DEFINITION: $\text{DEL2\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{del2\$value}$

$\text{del2\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, 'de12, list(a), state, netlist) = list(f\text{-buf}(a)))$

EVENT: Enable $\text{del2\$value}$; name this event ' $\text{del2\$value-off}$ '.

DEFINITION: $\text{del4\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{del4\&}$; name this event ' $\text{del4\&-off}$ '.

DEFINITION: $\text{DEL4\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{del4\$value}$

$\text{del4\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, 'de14, list(a), state, netlist) = list(f\text{-buf}(a)))$

EVENT: Enable $\text{del4\$value}$; name this event ' $\text{del4\$value-off}$ '.

DEFINITION: $\text{del10\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{del10\&}$; name this event ' $\text{del10\&-off}$ '.

DEFINITION: $\text{DEL10\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{del10\$value}$

$\text{del10\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, 'de110, list(a), state, netlist) = list(f\text{-buf}(a)))$

EVENT: Enable $\text{del10\$value}$; name this event ' $\text{del10\$value-off}$ '.

DEFINITION: $\text{procmon\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{procmon\&}$; name this event ' $\text{procmon\&-off}$ '.

DEFINITION: $\text{PROCMON\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{procmon\$value}$

$\text{procmon\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, 'procmon, list(a, e, s, n), state, netlist)$
 $= list(f\text{-if}(s, f\text{-if}(e, f\text{-if}(a, \mathbf{f}, \mathbf{f}), a), n)))$

EVENT: Enable procmon\$value; name this event ‘procmon\$value-off’.

DEFINITION: $\text{dp-ram-16x32\&}(netlist) = \mathbf{t}$

EVENT: Enable dp-ram-16x32\&; name this event ‘dp-ram-16x32\&-off’.

DEFINITION: $\text{DP-RAM-16X32\$NETLIST} = \mathbf{nil}$

THEOREM: dp-ram-16x32\$value

dp-ram-16x32\&(*netlist*)

→ (dual-eval(0,
 ’dp-ram-16x32,
 list(read-a0,
 read-a1,
 read-a2,
 read-a3,
 write-b0,
 write-b1,
 write-b2,
 write-b3,
 wen,
 d0,
 d1,
 d2,
 d3,
 d4,
 d5,
 d6,
 d7,
 d8,
 d9,
 d10,
 d11,
 d12,
 d13,
 d14,
 d15,
 d16,
 d17,
 d18,
 d19,
 d20,
 d21,

```

        d22,
        d23,
        d24,
        d25,
        d26,
        d27,
        d28,
        d29,
        d30,
        d31),
    state,
    netlist)
= dual-port-ram-value (32,
    4,
    list (read-a0,
        read-a1,
        read-a2,
        read-a3,
        write-b0,
        write-b1,
        write-b2,
        write-b3,
        wen,
        d0,
        d1,
        d2,
        d3,
        d4,
        d5,
        d6,
        d7,
        d8,
        d9,
        d10,
        d11,
        d12,
        d13,
        d14,
        d15,
        d16,
        d17,
        d18,
        d19,
        d20,
```

$d21,$
 $d22,$
 $d23,$
 $d24,$
 $d25,$
 $d26,$
 $d27,$
 $d28,$
 $d29,$
 $d30,$
 $d31),$
 $state))$

EVENT: Enable dp-ram-16x32\$value; name this event ‘dp-ram-16x32\$value-off’.

DEFINITION: $fd1\&(netlist) = \mathbf{t}$

EVENT: Enable fd1&; name this event ‘fd1&-off’.

DEFINITION: $FD1\$NETLIST = \mathbf{nil}$

THEOREM: fd1\$value
 $fd1\&(netlist)$
 $\rightarrow (\text{dual-eval}(0, 'fd1, \text{list}(d, cp), state, netlist)$
 $\quad = \text{list}(f\text{-buf}(state), f\text{-not}(state)))$

EVENT: Enable fd1\$value; name this event ‘fd1\$value-off’.

DEFINITION: $fd1s\&(netlist) = \mathbf{t}$

EVENT: Enable fd1s&; name this event ‘fd1s&-off’.

DEFINITION: $FD1S\$NETLIST = \mathbf{nil}$

THEOREM: fd1s\$value
 $fd1s\&(netlist)$
 $\rightarrow (\text{dual-eval}(0, 'fd1s, \text{list}(d, cp, ti, te), state, netlist)$
 $\quad = \text{list}(f\text{-buf}(state), f\text{-not}(state)))$

EVENT: Enable fd1s\$value; name this event ‘fd1s\$value-off’.

DEFINITION: $fd1sp\&(netlist) = \mathbf{t}$

EVENT: Enable $\text{fd1sp\&}$; name this event ‘ $\text{fd1sp\&-off}$ ’.

DEFINITION: $\text{FD1SP\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{fd1sp\$value}$

$\text{fd1sp\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'fd1sp'}, \text{list}(d, cp, ti, te), state, netlist)$
 $= \text{list}(\text{f-buf}(state), \text{f-not}(state)))$

EVENT: Enable $\text{fd1sp\$value}$; name this event ‘ $\text{fd1sp\$value-off}$ ’.

DEFINITION: $\text{fd1slp\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{fd1slp\&}$; name this event ‘ $\text{fd1slp\&-off}$ ’.

DEFINITION: $\text{FD1SLP\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{fd1slp\$value}$

$\text{fd1slp\&}(netlist)$

$\rightarrow (\text{dual-eval}(0, \text{'fd1slp'}, \text{list}(d, cp, ld, ti, te), state, netlist)$
 $= \text{list}(\text{f-buf}(state), \text{f-not}(state)))$

EVENT: Enable $\text{fd1slp\$value}$; name this event ‘ $\text{fd1slp\$value-off}$ ’.

DEFINITION: $\text{id\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{id\&}$; name this event ‘ $\text{id\&-off}$ ’.

DEFINITION: $\text{ID\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{id\$value}$

$\text{id\&}(netlist) \rightarrow (\text{dual-eval}(0, \text{'id'}, \text{list}(a), state, netlist) = \text{list}(a))$

EVENT: Enable $\text{id\$value}$; name this event ‘ $\text{id\$value-off}$ ’.

DEFINITION: $\text{mem-32x32\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{mem-32x32\&}$; name this event ‘ $\text{mem-32x32\&-off}$ ’.

DEFINITION: $\text{MEM-32X32\$NETLIST} = \mathbf{nil}$

```

THEOREM: mem-32x32$value
mem-32x32& (netlist)
→ (dual-eval (0,
               'mem-32x32,
               list (rw-,
                    strobe-,
                    a0,
                    a1,
                    a2,
                    a3,
                    a4,
                    a5,
                    a6,
                    a7,
                    a8,
                    a9,
                    a10,
                    a11,
                    a12,
                    a13,
                    a14,
                    a15,
                    a16,
                    a17,
                    a18,
                    a19,
                    a20,
                    a21,
                    a22,
                    a23,
                    a24,
                    a25,
                    a26,
                    a27,
                    a28,
                    a29,
                    a30,
                    a31,
                    d0,
                    d1,
                    d2,
                    d3,
                    d4,
                    d5,

```

```

        d6,
        d7,
        d8,
        d9,
        d10,
        d11,
        d12,
        d13,
        d14,
        d15,
        d16,
        d17,
        d18,
        d19,
        d20,
        d21,
        d22,
        d23,
        d24,
        d25,
        d26,
        d27,
        d28,
        d29,
        d30,
        d31),
    state,
    netlist)
= mem-value (list (rw-,
                   strobe-,
                   a0,
                   a1,
                   a2,
                   a3,
                   a4,
                   a5,
                   a6,
                   a7,
                   a8,
                   a9,
                   a10,
                   a11,
                   a12,
                   a13,

```

$a14,$
 $a15,$
 $a16,$
 $a17,$
 $a18,$
 $a19,$
 $a20,$
 $a21,$
 $a22,$
 $a23,$
 $a24,$
 $a25,$
 $a26,$
 $a27,$
 $a28,$
 $a29,$
 $a30,$
 $a31,$
 $d0,$
 $d1,$
 $d2,$
 $d3,$
 $d4,$
 $d5,$
 $d6,$
 $d7,$
 $d8,$
 $d9,$
 $d10,$
 $d11,$
 $d12,$
 $d13,$
 $d14,$
 $d15,$
 $d16,$
 $d17,$
 $d18,$
 $d19,$
 $d20,$
 $d21,$
 $d22,$
 $d23,$
 $d24,$
 $d25,$

$d26,$
 $d27,$
 $d28,$
 $d29,$
 $d30,$
 $d31),$
 $state))$

EVENT: Enable mem-32x32\$value; name this event ‘mem-32x32\$value-off’.

DEFINITION: $\text{ram-enable-circuit\&}(netlist) = \mathbf{t}$

EVENT: Enable ram-enable-circuit\&; name this event ‘ram-enable-circuit\&-off’.

DEFINITION: $\text{RAM-ENABLE-CIRCUIT\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{ram-enable-circuit\$value}$
 $\text{ram-enable-circuit\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0,$
 $\quad \text{'ram-enable-circuit},$
 $\quad \text{list}(clk, test-regfile-, disable-regfile-, we),$
 $\quad state,$
 $\quad netlist)$
 $= \text{list}(\text{f-nand}(disable-regfile-, we)))$

EVENT: Enable ram-enable-circuit\$value; name this event ‘ram-enable-circuit\$value-off’.

DEFINITION: $\text{t-buf\&}(netlist) = \mathbf{t}$

EVENT: Enable t-buf\&; name this event ‘t-buf\&-off’.

DEFINITION: $\text{T-BUF\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{t-buf\$value}$
 $\text{t-buf\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'t-buf}, \text{list}(e, a), state, netlist) = \text{list}(\text{ft-buf}(e, a)))$

EVENT: Enable t-buf\$value; name this event ‘t-buf\$value-off’.

DEFINITION: $\text{t-wire\&}(netlist) = \mathbf{t}$

EVENT: Enable t-wire\&; name this event ‘t-wire\&-off’.

DEFINITION: $\text{T-WIRE}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{t-wire}\$value$

$\text{t-wire}\& (netlist)$

$$\rightarrow (\text{dual-eval}(0, \text{'t-wire'}, \text{list}(a, b), state, netlist) \\ = \text{list}(\text{ft-wire}(a, b)))$$

EVENT: Enable $\text{t-wire}\$value$; name this event ' $\text{t-wire}\$value\text{-off}$ '.

DEFINITION: $\text{pullup}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{pullup}\&$; name this event ' $\text{pullup}\&\text{-off}$ '.

DEFINITION: $\text{PULLUP}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{pullup}\$value$

$\text{pullup}\& (netlist)$

$$\rightarrow (\text{dual-eval}(0, \text{'pullup'}, \text{list}(a), state, netlist) = \text{list}(\text{f-pullup}(a)))$$

EVENT: Enable $\text{pullup}\$value$; name this event ' $\text{pullup}\$value\text{-off}$ '.

DEFINITION: $\text{ttl-bidirect}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{ttl-bidirect}\&$; name this event ' $\text{ttl-bidirect}\&\text{-off}$ '.

DEFINITION: $\text{TTL-BIDIRECT}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{ttl-bidirect}\$value$

$\text{ttl-bidirect}\& (netlist)$

$$\rightarrow (\text{dual-eval}(0, \text{'ttl-bidirect'}, \text{list}(io, a, en, pi), state, netlist) \\ = \text{list}(\text{ft-buf}(\text{f-not}(en), a), \\ \text{f-buf}(\text{ft-wire}(io, \text{ft-buf}(\text{f-not}(en), a))), \\ \text{f-nand}(\text{ft-wire}(io, \text{ft-buf}(\text{f-not}(en), a)), pi)))$$

EVENT: Enable $\text{ttl-bidirect}\$value$; name this event ' $\text{ttl-bidirect}\$value\text{-off}$ '.

DEFINITION: $\text{ttl-clk-input}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{ttl-clk-input}\&$; name this event ' $\text{ttl-clk-input}\&\text{-off}$ '.

DEFINITION: $\text{TTL-CLK-INPUT}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{ttl-clk-input}\$value$
 $\text{ttl-clk-input}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-clk-input'}, \text{list}(a, pi), state, netlist)$
 $= \text{list}(\text{f-buf}(a), \text{f-nand}(a, pi)))$

EVENT: Enable $\text{ttl-clk-input}\$value$; name this event ' $\text{ttl-clk-input}\$value\text{-off}$ '.

DEFINITION: $\text{ttl-input}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{ttl-input}\&$; name this event ' $\text{ttl-input}\&\text{-off}$ '.

DEFINITION: $\text{TTL-INPUT}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{ttl-input}\$value$
 $\text{ttl-input}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-input'}, \text{list}(a, pi), state, netlist)$
 $= \text{list}(\text{f-buf}(a), \text{f-nand}(a, pi)))$

EVENT: Enable $\text{ttl-input}\$value$; name this event ' $\text{ttl-input}\$value\text{-off}$ '.

DEFINITION: $\text{ttl-output}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{ttl-output}\&$; name this event ' $\text{ttl-output}\&\text{-off}$ '.

DEFINITION: $\text{TTL-OUTPUT}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{ttl-output}\$value$
 $\text{ttl-output}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-output'}, \text{list}(a), state, netlist)$
 $= \text{list}(\text{f-buf}(a)))$

EVENT: Enable $\text{ttl-output}\$value$; name this event ' $\text{ttl-output}\$value\text{-off}$ '.

DEFINITION: $\text{ttl-output-parametric}\& (netlist) = \mathbf{t}$

EVENT: Enable $\text{ttl-output-parametric}\&$; name this event ' $\text{ttl-output-parametric}\&\text{-off}$ '.

DEFINITION: $\text{TTL-OUTPUT-PARAMETRIC}\$NETLIST = \mathbf{nil}$

THEOREM: $\text{ttl-output-parametric}\$value$
 $\text{ttl-output-parametric}\& (netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-output-parametric'}, \text{list}(a), state, netlist)$
 $= \text{list}(\text{f-buf}(a)))$

EVENT: Enable ttl-output-parametric\$value; name this event ‘ttl-output-parametric\$value-off’.

DEFINITION: $\text{ttl-output-fast\&}(netlist) = \mathbf{t}$

EVENT: Enable ttl-output-fast\&; name this event ‘ttl-output-fast\&-off’.

DEFINITION: $\text{TTL-OUTPUT-FAST\$NETLIST} = \mathbf{nil}$

THEOREM: ttl-output-fast\$value
 $\text{ttl-output-fast\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-output-fast'}, \text{list}(a), state, netlist)$
 $\quad = \text{list}(\text{f-buf}(a)))$

EVENT: Enable ttl-output-fast\$value; name this event ‘ttl-output-fast\$value-off’.

DEFINITION: $\text{ttl-tri-output\&}(netlist) = \mathbf{t}$

EVENT: Enable ttl-tri-output\&; name this event ‘ttl-tri-output\&-off’.

DEFINITION: $\text{TTL-TRI-OUTPUT\$NETLIST} = \mathbf{nil}$

THEOREM: ttl-tri-output\$value
 $\text{ttl-tri-output\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-tri-output'}, \text{list}(a, en), state, netlist)$
 $\quad = \text{list}(\text{ft-buf}(\text{f-not}(en), a)))$

EVENT: Enable ttl-tri-output\$value; name this event ‘ttl-tri-output\$value-off’.

DEFINITION: $\text{ttl-tri-output-fast\&}(netlist) = \mathbf{t}$

EVENT: Enable ttl-tri-output-fast\&; name this event ‘ttl-tri-output-fast\&-off’.

DEFINITION: $\text{TTL-TRI-OUTPUT-FAST\$NETLIST} = \mathbf{nil}$

THEOREM: ttl-tri-output-fast\$value
 $\text{ttl-tri-output-fast\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, \text{'ttl-tri-output-fast'}, \text{list}(a, en), state, netlist)$
 $\quad = \text{list}(\text{ft-buf}(\text{f-not}(en), a)))$

EVENT: Enable ttl-tri-output-fast\$value; name this event ‘ttl-tri-output-fast\$value-off’.

DEFINITION: $\text{vdd\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{vdd\&}$; name this event ‘ $\text{vdd\&-off}$ ’.

DEFINITION: $\text{VDD\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{vdd\$value}$

$\text{vdd\&}(netlist) \rightarrow (\text{dual-eval}(0, \text{'vdd}, \mathbf{nil}, state, netlist) = \text{list}(\mathbf{t}))$

EVENT: Enable $\text{vdd\$value}$; name this event ‘ $\text{vdd\$value-off}$ ’.

DEFINITION: $\text{vdd-parametric\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{vdd-parametric\&}$; name this event ‘ $\text{vdd-parametric\&-off}$ ’.

DEFINITION: $\text{VDD-PARAMETRIC\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{vdd-parametric\$value}$

$\text{vdd-parametric\&}(netlist) \rightarrow (\text{dual-eval}(0, \text{'vdd-parametric}, \mathbf{nil}, state, netlist) = \text{list}(\mathbf{t}))$

EVENT: Enable $\text{vdd-parametric\$value}$; name this event ‘ $\text{vdd-parametric\$value-off}$ ’.

DEFINITION: $\text{vss\&}(netlist) = \mathbf{t}$

EVENT: Enable $\text{vss\&}$; name this event ‘ $\text{vss\&-off}$ ’.

DEFINITION: $\text{VSS\$NETLIST} = \mathbf{nil}$

THEOREM: $\text{vss\$value}$

$\text{vss\&}(netlist) \rightarrow (\text{dual-eval}(0, \text{'vss}, \mathbf{nil}, state, netlist) = \text{list}(\mathbf{f}))$

EVENT: Enable $\text{vss\$value}$; name this event ‘ $\text{vss\$value-off}$ ’.

THEOREM: $\text{dp-ram-16x32\$state}$

$\text{dp-ram-16x32\&}(netlist) \rightarrow (\text{dual-eval}(2, \text{'dp-ram-16x32}, \text{list}(\text{read-a0}, \text{read-a1}, \text{read-a2}, \text{read-a3},$

```

        write-b0,
        write-b1,
        write-b2,
        write-b3,
        wen,
        d0,
        d1,
        d2,
        d3,
        d4,
        d5,
        d6,
        d7,
        d8,
        d9,
        d10,
        d11,
        d12,
        d13,
        d14,
        d15,
        d16,
        d17,
        d18,
        d19,
        d20,
        d21,
        d22,
        d23,
        d24,
        d25,
        d26,
        d27,
        d28,
        d29,
        d30,
        d31),
    state,
    netlist)
= dual-port-ram-state (32,
                        4,
                        list (read-a0,
                            read-a1,
                            read-a2,
```

```

    read-a3,
    write-b0,
    write-b1,
    write-b2,
    write-b3,
    wen,
    d0,
    d1,
    d2,
    d3,
    d4,
    d5,
    d6,
    d7,
    d8,
    d9,
    d10,
    d11,
    d12,
    d13,
    d14,
    d15,
    d16,
    d17,
    d18,
    d19,
    d20,
    d21,
    d22,
    d23,
    d24,
    d25,
    d26,
    d27,
    d28,
    d29,
    d30,
    d31),
state))

```

EVENT: Enable dp-ram-16x32\$state; name this event ‘dp-ram-16x32\$state-off’.

THEOREM: fd1\$state
fd1& (*netlist*)

→ (dual-eval (2, 'fd1, list (d, cp), state, netlist) = f-buf (d))

EVENT: Enable fd1\$state; name this event 'fd1\$state-off'.

THEOREM: fd1s\$state

fd1s& (netlist)

→ (dual-eval (2, 'fd1s, list (d, cp, ti, te), state, netlist)
= f-if (te, ti, d))

EVENT: Enable fd1s\$state; name this event 'fd1s\$state-off'.

THEOREM: fd1sp\$state

fd1sp& (netlist)

→ (dual-eval (2, 'fd1sp, list (d, cp, ti, te), state, netlist)
= f-if (te, ti, d))

EVENT: Enable fd1sp\$state; name this event 'fd1sp\$state-off'.

THEOREM: fd1slp\$state

fd1slp& (netlist)

→ (dual-eval (2, 'fd1slp, list (d, cp, ld, ti, te), state, netlist)
= f-if (te, ti, f-if (ld, d, state)))

EVENT: Enable fd1slp\$state; name this event 'fd1slp\$state-off'.

THEOREM: mem-32x32\$state

mem-32x32& (netlist)

→ (dual-eval (2,
 'mem-32x32,
 list (rw-,
 strobe-,
 a0,
 a1,
 a2,
 a3,
 a4,
 a5,
 a6,
 a7,
 a8,
 a9,
 a10,
 a11,

$a_{12},$
 $a_{13},$
 $a_{14},$
 $a_{15},$
 $a_{16},$
 $a_{17},$
 $a_{18},$
 $a_{19},$
 $a_{20},$
 $a_{21},$
 $a_{22},$
 $a_{23},$
 $a_{24},$
 $a_{25},$
 $a_{26},$
 $a_{27},$
 $a_{28},$
 $a_{29},$
 $a_{30},$
 $a_{31},$
 $d_0,$
 $d_1,$
 $d_2,$
 $d_3,$
 $d_4,$
 $d_5,$
 $d_6,$
 $d_7,$
 $d_8,$
 $d_9,$
 $d_{10},$
 $d_{11},$
 $d_{12},$
 $d_{13},$
 $d_{14},$
 $d_{15},$
 $d_{16},$
 $d_{17},$
 $d_{18},$
 $d_{19},$
 $d_{20},$
 $d_{21},$
 $d_{22},$
 $d_{23},$


```

        d24,
        d25,
        d26,
        d27,
        d28,
        d29,
        d30,
        d31),
    state,
    netlist)
= mem-state(list(rw-,
                 strobe-,
                 a0,
                 a1,
                 a2,
                 a3,
                 a4,
                 a5,
                 a6,
                 a7,
                 a8,
                 a9,
                 a10,
                 a11,
                 a12,
                 a13,
                 a14,
                 a15,
                 a16,
                 a17,
                 a18,
                 a19,
                 a20,
                 a21,
                 a22,
                 a23,
                 a24,
                 a25,
                 a26,
                 a27,
                 a28,
                 a29,
                 a30,
                 a31,

```

```

d0,
d1,
d2,
d3,
d4,
d5,
d6,
d7,
d8,
d9,
d10,
d11,
d12,
d13,
d14,
d15,
d16,
d17,
d18,
d19,
d20,
d21,
d22,
d23,
d24,
d25,
d26,
d27,
d28,
d29,
d30,
d31),
state))

```

EVENT: Enable mem-32x32\$state; name this event ‘mem-32x32\$state-off’.

DEFINITION:

B-BUF*

```

=  '(b-buf
    (in)
    (out)
    ((g0 (out- out) b-nbuf (in)))
    nil)

```

DEFINITION:

$\text{b-buf\&}(netlist)$
= $((\text{lookup-module}('b\text{-buf}, netlist) = B\text{-BUF}^*)$
 $\wedge \text{b-nbuf\&}(\text{delete-module}('b\text{-buf}, netlist)))$

EVENT: Enable b-buf&; name this event 'b-buf&-off'.

DEFINITION: $B\text{-BUF}\$NETLIST = \text{cons}(B\text{-BUF}^*, B\text{-NBUF}\$NETLIST)$

THEOREM: b-buf\$value

$\text{b-buf\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, 'b\text{-buf}, \text{list}(in), state, netlist) = \text{list}(\text{f-buf}(in)))$

EVENT: Enable b-buf\$value; name this event 'b-buf\$value-off'.

DEFINITION:

$B\text{-BUF-PWR}^*$
= $'(b\text{-buf-pwr}$
 (in)
 (out)
 $((g0(out-) b\text{-not}(in))$
 $(g1(out) b\text{-not-b4ip}(out-)))$
 $nil)$

DEFINITION:

$\text{b-buf-pwr\&}(netlist)$
= $((\text{lookup-module}('b\text{-buf-pwr}, netlist) = B\text{-BUF-PWR}^*)$
 $\wedge \text{b-not\&}(\text{delete-module}('b\text{-buf-pwr}, netlist))$
 $\wedge \text{b-not-b4ip\&}(\text{delete-module}('b\text{-buf-pwr}, netlist)))$

EVENT: Enable b-buf-pwr&; name this event 'b-buf-pwr&-off'.

DEFINITION:

$B\text{-BUF-PWR}\$NETLIST = \text{cons}(B\text{-BUF-PWR}^*, B\text{-NOT}\$NETLIST \cup B\text{-NOT-B4IP}\$NETLIST)$

THEOREM: b-buf-pwr\$value

$\text{b-buf-pwr\&}(netlist)$
 $\rightarrow (\text{dual-eval}(0, 'b\text{-buf-pwr}, \text{list}(in), state, netlist)$
 $= \text{list}(\text{f-buf}(in)))$

EVENT: Enable b-buf-pwr\$value; name this event 'b-buf-pwr\$value-off'.

THEOREM: dp-ram-16x32-args-crock

$$\begin{aligned}
& ((\text{length}(a) = 4) \\
& \wedge \text{properp}(a) \\
& \wedge (\text{length}(b) = 4) \\
& \wedge \text{properp}(b) \\
& \wedge (\text{length}(c) = 32) \\
& \wedge \text{properp}(c)) \\
\rightarrow & (\text{append}(a, \text{append}(b, \text{cons}(x, c))) \\
& = \text{append}(\text{list-as-collected-nth}(a, 4, 0), \\
& \quad \text{append}(\text{list-as-collected-nth}(b, 4, 0), \\
& \quad \quad \text{cons}(x, \text{list-as-collected-nth}(c, 32, 0))))))
\end{aligned}$$

EVENT: Enable dp-ram-16x32-args-crock; name this event ‘dp-ram-16x32-args-crock-off’.

THEOREM: dp-ram-16x32\$structured-value
(dp-ram-16x32& (*netlist*)

$$\begin{aligned}
& \wedge (\text{length}(a) = 4) \\
& \wedge \text{properp}(a) \\
& \wedge (\text{length}(b) = 4) \\
& \wedge \text{properp}(b) \\
& \wedge (\text{length}(c) = 32) \\
& \wedge \text{properp}(c)) \\
\rightarrow & (\text{dual-eval}(0, \\
& \quad \text{'dp-ram-16x32}, \\
& \quad \text{append}(a, \text{append}(b, \text{cons}(x, c))), \\
& \quad \text{state}, \\
& \quad \text{netlist}) \\
& = \text{dual-port-ram-value}(32, 4, \text{append}(a, \text{append}(b, \text{cons}(x, c))), \text{state}))
\end{aligned}$$

EVENT: Enable dp-ram-16x32\$structured-value; name this event ‘dp-ram-16x32\$structured-value-off’.

THEOREM: dp-ram-16x32\$structured-state
(dp-ram-16x32& (*netlist*)

$$\begin{aligned}
& \wedge (\text{length}(a) = 4) \\
& \wedge \text{properp}(a) \\
& \wedge (\text{length}(b) = 4) \\
& \wedge \text{properp}(b) \\
& \wedge (\text{length}(c) = 32) \\
& \wedge \text{properp}(c)) \\
\rightarrow & (\text{dual-eval}(2, \\
& \quad \text{'dp-ram-16x32}, \\
& \quad \text{append}(a, \text{append}(b, \text{cons}(x, c))),
\end{aligned}$$

$$\begin{aligned}
& \text{state,} \\
& \text{netlist}) \\
= & \text{dual-port-ram-state}(32, 4, \text{append}(a, \text{append}(b, \text{cons}(x, c))), \text{state}))
\end{aligned}$$

EVENT: Enable dp-ram-16x32\$structured-state; name this event ‘dp-ram-16x32\$structured-state-off’.

$$\begin{aligned}
& \text{THEOREM: mem-32x32-args-crock} \\
& ((\text{length}(addr) = 32) \\
& \wedge \text{properp}(addr) \\
& \wedge (\text{length}(data) = 32) \\
& \wedge \text{properp}(data)) \\
\rightarrow & (\text{cons}(rw-, \text{cons}(strobe-, \text{append}(addr, data)))) \\
& = \text{cons}(rw-, \\
& \quad \text{cons}(strobe-, \\
& \quad \quad \text{append}(\text{list-as-collected-nth}(addr, 32, 0), \\
& \quad \quad \text{list-as-collected-nth}(data, 32, 0))))))
\end{aligned}$$

EVENT: Enable mem-32x32-args-crock; name this event ‘mem-32x32-args-crock-off’.

$$\begin{aligned}
& \text{THEOREM: list-32-nth-collapse} \\
& (\text{properp}(args) \wedge (\text{length}(args) = 32)) \\
\rightarrow & (\text{list}(\text{nth}(0, args), \\
& \quad \text{nth}(1, args), \\
& \quad \text{nth}(2, args), \\
& \quad \text{nth}(3, args), \\
& \quad \text{nth}(4, args), \\
& \quad \text{nth}(5, args), \\
& \quad \text{nth}(6, args), \\
& \quad \text{nth}(7, args), \\
& \quad \text{nth}(8, args), \\
& \quad \text{nth}(9, args), \\
& \quad \text{nth}(10, args), \\
& \quad \text{nth}(11, args), \\
& \quad \text{nth}(12, args), \\
& \quad \text{nth}(13, args), \\
& \quad \text{nth}(14, args), \\
& \quad \text{nth}(15, args), \\
& \quad \text{nth}(16, args), \\
& \quad \text{nth}(17, args), \\
& \quad \text{nth}(18, args), \\
& \quad \text{nth}(19, args),
\end{aligned}$$

$$\begin{aligned}
& \text{nth}(20, \text{args}), \\
& \text{nth}(21, \text{args}), \\
& \text{nth}(22, \text{args}), \\
& \text{nth}(23, \text{args}), \\
& \text{nth}(24, \text{args}), \\
& \text{nth}(25, \text{args}), \\
& \text{nth}(26, \text{args}), \\
& \text{nth}(27, \text{args}), \\
& \text{nth}(28, \text{args}), \\
& \text{nth}(29, \text{args}), \\
& \text{nth}(30, \text{args}), \\
& \text{nth}(31, \text{args})) \\
= & \text{args})
\end{aligned}$$

EVENT: Enable list-32-nth-collapse; name this event 'list-32-nth-collapse-off'.

THEOREM: mem-32x32\$structured-value-1

$$\begin{aligned}
& (\text{mem-32x32} \& (\text{netlist}) \\
& \wedge \text{length}(\text{addr}) = 32) \\
& \wedge \text{properp}(\text{addr}) \\
& \wedge \text{length}(\text{data}) = 32) \\
& \wedge \text{properp}(\text{data})) \\
\rightarrow & (\text{dual-eval}(0, \\
& \quad \text{'mem-32x32}, \\
& \quad \text{cons}(\text{rw-}, \text{cons}(\text{strobe-}, \text{append}(\text{addr}, \text{data}))), \\
& \quad \text{state}, \\
& \quad \text{netlist}) \\
= & \text{memory-value}(\text{state}, \\
& \quad \text{strobe-}, \\
& \quad \text{rw-}, \\
& \quad \text{list}(\text{nth}(0, \text{addr}), \\
& \quad \quad \text{nth}(1, \text{addr}), \\
& \quad \quad \text{nth}(2, \text{addr}), \\
& \quad \quad \text{nth}(3, \text{addr}), \\
& \quad \quad \text{nth}(4, \text{addr}), \\
& \quad \quad \text{nth}(5, \text{addr}), \\
& \quad \quad \text{nth}(6, \text{addr}), \\
& \quad \quad \text{nth}(7, \text{addr}), \\
& \quad \quad \text{nth}(8, \text{addr}), \\
& \quad \quad \text{nth}(9, \text{addr}), \\
& \quad \quad \text{nth}(10, \text{addr}), \\
& \quad \quad \text{nth}(11, \text{addr}), \\
& \quad \quad \text{nth}(12, \text{addr}), \\
& \quad \quad \text{nth}(13, \text{addr}),
\end{aligned}$$

```

    nth (14, addr),
    nth (15, addr),
    nth (16, addr),
    nth (17, addr),
    nth (18, addr),
    nth (19, addr),
    nth (20, addr),
    nth (21, addr),
    nth (22, addr),
    nth (23, addr),
    nth (24, addr),
    nth (25, addr),
    nth (26, addr),
    nth (27, addr),
    nth (28, addr),
    nth (29, addr),
    nth (30, addr),
    nth (31, addr)),
list (nth (0, data),
      nth (1, data),
      nth (2, data),
      nth (3, data),
      nth (4, data),
      nth (5, data),
      nth (6, data),
      nth (7, data),
      nth (8, data),
      nth (9, data),
      nth (10, data),
      nth (11, data),
      nth (12, data),
      nth (13, data),
      nth (14, data),
      nth (15, data),
      nth (16, data),
      nth (17, data),
      nth (18, data),
      nth (19, data),
      nth (20, data),
      nth (21, data),
      nth (22, data),
      nth (23, data),
      nth (24, data),
      nth (25, data),

```

$$\begin{aligned} & \text{nth}(26, \text{data}), \\ & \text{nth}(27, \text{data}), \\ & \text{nth}(28, \text{data}), \\ & \text{nth}(29, \text{data}), \\ & \text{nth}(30, \text{data}), \\ & \text{nth}(31, \text{data}))) \end{aligned}$$

EVENT: Enable mem-32x32\$structured-value-1; name this event ‘mem-32x32\$structured-value-1-off’.

THEOREM: mem-32x32\$structured-value
 $(\text{mem-32x32} \& (\text{netlist})$
 $\wedge (\text{length}(\text{addr}) = 32)$
 $\wedge \text{properp}(\text{addr})$
 $\wedge (\text{length}(\text{data}) = 32)$
 $\wedge \text{properp}(\text{data}))$
 $\rightarrow (\text{dual-eval}(0,$
 $\quad \text{'mem-32x32},$
 $\quad \text{cons}(\text{rw-}, \text{cons}(\text{strobe-}, \text{append}(\text{addr}, \text{data}))),$
 $\quad \text{state},$
 $\quad \text{netlist})$
 $= \text{memory-value}(\text{state}, \text{strobe-}, \text{rw-}, \text{addr}, \text{data}))$

EVENT: Enable mem-32x32\$structured-value; name this event ‘mem-32x32\$structured-value-off’.

THEOREM: mem-32x32\$structured-state-1
 $(\text{mem-32x32} \& (\text{netlist})$
 $\wedge (\text{length}(\text{addr}) = 32)$
 $\wedge \text{properp}(\text{addr})$
 $\wedge (\text{length}(\text{data}) = 32)$
 $\wedge \text{properp}(\text{data}))$
 $\rightarrow (\text{dual-eval}(2,$
 $\quad \text{'mem-32x32},$
 $\quad \text{cons}(\text{rw-}, \text{cons}(\text{strobe-}, \text{append}(\text{addr}, \text{data}))),$
 $\quad \text{state},$
 $\quad \text{netlist})$
 $= \text{next-memory-state}(\text{state},$
 $\quad \text{strobe-},$
 $\quad \text{rw-},$
 $\quad \text{list}(\text{nth}(0, \text{addr}),$
 $\quad \quad \text{nth}(1, \text{addr}),$
 $\quad \quad \text{nth}(2, \text{addr}),$


```

nth(3, addr),
nth(4, addr),
nth(5, addr),
nth(6, addr),
nth(7, addr),
nth(8, addr),
nth(9, addr),
nth(10, addr),
nth(11, addr),
nth(12, addr),
nth(13, addr),
nth(14, addr),
nth(15, addr),
nth(16, addr),
nth(17, addr),
nth(18, addr),
nth(19, addr),
nth(20, addr),
nth(21, addr),
nth(22, addr),
nth(23, addr),
nth(24, addr),
nth(25, addr),
nth(26, addr),
nth(27, addr),
nth(28, addr),
nth(29, addr),
nth(30, addr),
nth(31, addr)),
list(nth(0, data),
nth(1, data),
nth(2, data),
nth(3, data),
nth(4, data),
nth(5, data),
nth(6, data),
nth(7, data),
nth(8, data),
nth(9, data),
nth(10, data),
nth(11, data),
nth(12, data),
nth(13, data),
nth(14, data),

```

```

nth (15, data),
nth (16, data),
nth (17, data),
nth (18, data),
nth (19, data),
nth (20, data),
nth (21, data),
nth (22, data),
nth (23, data),
nth (24, data),
nth (25, data),
nth (26, data),
nth (27, data),
nth (28, data),
nth (29, data),
nth (30, data),
nth (31, data))))

```

EVENT: Enable mem-32x32\$structured-state-1; name this event ‘mem-32x32\$structured-state-1-off’.

THEOREM: mem-32x32\$structured-state

```

(mem-32x32& (netlist)
  ∧ (length (addr) = 32)
  ∧ properp (addr)
  ∧ (length (data) = 32)
  ∧ properp (data))
→ (dual-eval (2,
              'mem-32x32,
              cons (rw-, cons (strobe-, append (addr, data))),
              state,
              netlist)
    = next-memory-state (state, strobe-, rw-, addr, data))

```

EVENT: Enable mem-32x32\$structured-state; name this event ‘mem-32x32\$structured-state-off’.

DEFINITION:

```

unbound-in-body (name, body)
= if body ≈ nil then t
  else (name ∉ occ-outputs (car (body)))
       ∧ unbound-in-body (name, cdr (body)) endif

```

EVENT: Enable unbound-in-body; name this event ‘unbound-in-body-off’.

THEOREM: unbound-in-body-nlistp
 $(body \simeq \mathbf{nil}) \rightarrow \text{unbound-in-body}(name, body)$

THEOREM: unbound-in-body-listp
 $\text{unbound-in-body}(name, \text{cons}(occurrence, rest))$
 $= ((name \notin \text{occ-outputs}(occurrence)) \wedge \text{unbound-in-body}(name, rest))$

DEFINITION:
 $\text{all-unbound-in-body}(names, body)$
 $= \text{if } body \simeq \mathbf{nil} \text{ then } \mathbf{t}$
 $\quad \text{else } \text{disjoint}(names, \text{occ-outputs}(\text{car}(body)))$
 $\quad \wedge \text{all-unbound-in-body}(names, \text{cdr}(body)) \text{ endif}$

EVENT: Enable all-unbound-in-body; name this event ‘all-unbound-in-body-off’.

THEOREM: all-unbound-in-body-nlistp
 $(body \simeq \mathbf{nil}) \rightarrow \text{all-unbound-in-body}(names, body)$

THEOREM: all-unbound-in-body-listp
 $\text{all-unbound-in-body}(names, \text{cons}(occurrence, rest))$
 $= (\text{disjoint}(names, \text{occ-outputs}(occurrence))$
 $\quad \wedge \text{all-unbound-in-body}(names, rest))$

THEOREM: all-unbound-in-body-append
 $\text{all-unbound-in-body}(\text{append}(names1, names2), body)$
 $= (\text{all-unbound-in-body}(names1, body)$
 $\quad \wedge \text{all-unbound-in-body}(names2, body))$

THEOREM: all-unbound-in-body-cons
 $\text{all-unbound-in-body}(\text{cons}(name, names), body)$
 $= (\text{unbound-in-body}(name, body) \wedge \text{all-unbound-in-body}(names, body))$

THEOREM: all-unbound-in-body-nlistp-names
 $(names \simeq \mathbf{nil}) \rightarrow \text{all-unbound-in-body}(names, body)$

THEOREM: unbound-in-body-dual-eval-1
 $((flag = 1) \wedge \text{unbound-in-body}(name, body))$
 $\rightarrow (\text{value}(name, \text{dual-eval}(flag, body, bindings, state-bindings, netlist))$
 $\quad = \text{value}(name, bindings))$

THEOREM: all-unbound-in-body-dual-eval-1
 $((flag = 1) \wedge \text{all-unbound-in-body}(names, body))$
 $\rightarrow (\text{collect-value}(names,$
 $\quad \text{dual-eval}(flag, body, bindings, state-bindings, netlist))$
 $\quad = \text{collect-value}(names, bindings))$

DEFINITION:

```
vector-module-induction(body, m, n, bindings, state-bindings, netlist)
=  if n  $\simeq$  0 then t
    else vector-module-induction(cdr(body),
                                  1 + m,
                                  n - 1,
                                  dual-eval-body-bindings(1,
                                                            body,
                                                            bindings,
                                                            state-bindings,
                                                            netlist),
                                  state-bindings,
                                  netlist) endif
```

DEFINITION:

```
v-buf$body(m, n)
=  if n  $\simeq$  0 then nil
    else cons(list(index('g, m),
                    list(index('y, m)),
                    'b-buf,
                    list(index('a, m))),
              v-buf$body(1 + m, n - 1)) endif
```

EVENT: Enable v-buf\$body; name this event 'v-buf\$body-off'.

DEFINITION:

```
v-buf*(n)
=  cons(index('v-buf, n),
        cons(indices('a, 0, n),
              cons(indices('y, 0, n), cons(v-buf$body(0, n), '(nil))))))
```

THEOREM: v-buf*\$destructure

```
(car(v-buf*(n)) = index('v-buf, n))
 $\wedge$  (cadr(v-buf*(n)) = indices('a, 0, n))
 $\wedge$  (caddr(v-buf*(n)) = indices('y, 0, n))
 $\wedge$  (cadddr(v-buf*(n)) = v-buf$body(0, n))
 $\wedge$  (caddddr(v-buf*(n)) = nil)
```

EVENT: Enable v-buf*; name this event 'v-buf*-off'.

EVENT: Enable v-buf*\$destructure; name this event 'v-buf*\$destructure-off'.

DEFINITION:

$v\text{-buf\&}(netlist, n)$
 $= ((\text{lookup-module}(\text{index}('v\text{-buf}, n), netlist) = v\text{-buf}^*(n))$
 $\wedge \text{b-buf\&}(\text{delete-module}(\text{index}('v\text{-buf}, n), netlist)))$

EVENT: Enable $v\text{-buf\&}$; name this event ' $v\text{-buf\&-off}$ '.

DEFINITION: $v\text{-buf}\$netlist(n) = \text{cons}(v\text{-buf}^*(n), \text{B-BUF}\$NETLIST)$

THEOREM: $v\text{-buf}\$unbound\text{-in-body}$
 $(l < m) \rightarrow \text{unbound-in-body}(\text{index}('y, l), v\text{-buf}\$body(m, n))$

EVENT: Enable $v\text{-buf}\$unbound\text{-in-body}$; name this event ' $v\text{-buf}\$unbound\text{-in-body-off}$ '.

THEOREM: $v\text{-buf}\$body\text{-value}$
 $(b\text{-buf\&}(netlist) \wedge (body = v\text{-buf}\$body(m, n)))$
 $\rightarrow (\text{collect-value}(\text{indices}('y, m, n),$
 $\quad \text{dual-eval}(1, body, bindings, state\text{-}bindings, netlist))$
 $= v\text{-threefix}(\text{collect-value}(\text{indices}('a, m, n), bindings)))$

EVENT: Enable $v\text{-buf}\$body\text{-value}$; name this event ' $v\text{-buf}\$body\text{-value-off}$ '.

THEOREM: $v\text{-buf}\$value$
 $(v\text{-buf\&}(netlist, n) \wedge \text{properp}(a) \wedge (\text{length}(a) = n))$
 $\rightarrow (\text{dual-eval}(0, \text{index}('v\text{-buf}, n), a, state, netlist) = v\text{-threefix}(a))$

EVENT: Enable $v\text{-buf}\$value$; name this event ' $v\text{-buf}\$value-off$ '.

DEFINITION:
 $v\text{-or}\$body(m, n)$
 $= \text{if } n \simeq 0 \text{ then nil}$
 $\quad \text{else cons}(\text{list}(\text{index}('g, m),$
 $\quad \text{list}(\text{index}('y, m)),$
 $\quad 'b\text{-or},$
 $\quad \text{list}(\text{index}('a, m), \text{index}('b, m))),$
 $\quad v\text{-or}\$body(1 + m, n - 1)) \text{ endif}$

EVENT: Enable $v\text{-or}\$body$; name this event ' $v\text{-or}\$body-off$ '.

DEFINITION:
 $v\text{-or}^*(n)$
 $= \text{cons}(\text{index}('v\text{-or}, n),$
 $\quad \text{cons}(\text{append}(\text{indices}('a, 0, n), \text{indices}('b, 0, n)),$
 $\quad \text{cons}(\text{indices}('y, 0, n), \text{cons}(v\text{-or}\$body(0, n), '(\text{nil}))))))$

THEOREM: $v\text{-or}^*\$destructure$

$$\begin{aligned} & (\text{car}(v\text{-or}^*(n)) = \text{index}('v\text{-or}', n)) \\ & \wedge (\text{cadr}(v\text{-or}^*(n)) = \text{append}(\text{indices}('a', 0, n), \text{indices}('b', 0, n))) \\ & \wedge (\text{caddr}(v\text{-or}^*(n)) = \text{indices}('y', 0, n)) \\ & \wedge (\text{cadddr}(v\text{-or}^*(n)) = v\text{-or}\$body(0, n)) \\ & \wedge (\text{caddddr}(v\text{-or}^*(n)) = \mathbf{nil}) \end{aligned}$$

EVENT: Enable $v\text{-or}^*$; name this event ' $v\text{-or}^*\text{-off}$ '.

EVENT: Enable $v\text{-or}^*\$destructure$; name this event ' $v\text{-or}^*\$destructure\text{-off}$ '.

DEFINITION:

$$\begin{aligned} & v\text{-or}\& (netlist, n) \\ & = ((\text{lookup-module}(\text{index}('v\text{-or}', n), netlist) = v\text{-or}^*(n)) \\ & \quad \wedge \text{b-or}\& (\text{delete-module}(\text{index}('v\text{-or}', n), netlist))) \end{aligned}$$

EVENT: Enable $v\text{-or}\&$; name this event ' $v\text{-or}\&\text{-off}$ '.

DEFINITION: $v\text{-or}\$netlist(n) = \text{cons}(v\text{-or}^*(n), \text{B-OR}\$NETLIST)$

THEOREM: $v\text{-or}\$unbound\text{-in-body}$

$$(l < m) \rightarrow \text{unbound-in-body}(\text{index}('y', l), v\text{-or}\$body(m, n))$$

EVENT: Enable $v\text{-or}\$unbound\text{-in-body}$; name this event ' $v\text{-or}\$unbound\text{-in-body}\text{-off}$ '.

THEOREM: $v\text{-or}\$body\text{-value}$

$$\begin{aligned} & (\text{b-or}\& (netlist) \wedge (body = v\text{-or}\$body(m, n))) \\ & \rightarrow (\text{collect-value}(\text{indices}('y', m, n), \\ & \quad \text{dual-eval}(1, body, bindings, state\text{-}bindings, netlist)) \\ & = \text{fv-or}(\text{collect-value}(\text{indices}('a', m, n), bindings), \\ & \quad \text{collect-value}(\text{indices}('b', m, n), bindings))) \end{aligned}$$

EVENT: Enable $v\text{-or}\$body\text{-value}$; name this event ' $v\text{-or}\$body\text{-value}\text{-off}$ '.

THEOREM: $v\text{-or}\$value$

$$\begin{aligned} & (v\text{-or}\& (netlist, n) \\ & \wedge (\text{properp}(a) \wedge (\text{length}(a) = n)) \\ & \wedge \text{properp}(b) \\ & \wedge (\text{length}(b) = n)) \\ & \rightarrow (\text{dual-eval}(0, \text{index}('v\text{-or}', n), \text{append}(a, b), state, netlist) \\ & = \text{fv-or}(a, b)) \end{aligned}$$

EVENT: Enable v-or\$value; name this event 'v-or\$value-off'.

DEFINITION:

v-xor\$body(m, n)
 = **if** $n \simeq 0$ **then** **nil**
 else cons(list(index('g, m),
 list(index('y, m),
 'b-xor,
 list(index('a, m), index('b, m))),
 v-xor\$body($1 + m, n - 1$)) **endif**

EVENT: Enable v-xor\$body; name this event 'v-xor\$body-off'.

DEFINITION:

v-xor*(n)
 = cons(index('v-xor, n),
 cons(append(indices('a, 0, n), indices('b, 0, n)),
 cons(indices('y, 0, n), cons(v-xor\$body(0, n), ' (nil))))))

THEOREM: v-xor*\$destructure

(car(v-xor*(n)) = index('v-xor, n))
 $\wedge$ (cadr(v-xor*(n)) = append(indices('a, 0, n), indices('b, 0, n)))
 $\wedge$ (caddr(v-xor*(n)) = indices('y, 0, n))
 $\wedge$ (cadddr(v-xor*(n)) = v-xor\$body(0, n))
 $\wedge$ (caddddr(v-xor*(n)) = **nil**)

EVENT: Enable v-xor*; name this event 'v-xor*-off'.

EVENT: Enable v-xor*\$destructure; name this event 'v-xor*\$destructure-off'.

DEFINITION:

v-xor&($netlist, n$)
 = ((lookup-module(index('v-xor, n), $netlist$) = v-xor*(n))
 $\wedge$ b-xor&(delete-module(index('v-xor, n), $netlist$)))

EVENT: Enable v-xor&; name this event 'v-xor&-off'.

DEFINITION: v-xor\$netlist(n) = cons(v-xor*(n), B-XOR\$NETLIST)

THEOREM: v-xor\$unbound-in-body

($l < m$) $\rightarrow$ unbound-in-body(index('y, l), v-xor\$body(m, n))

EVENT: Enable v-xor\$unbound-in-body; name this event 'v-xor\$unbound-in-body-off'.

THEOREM: v-xor\$body-value
 $(b\text{-xor}\& (netlist) \wedge (body = v\text{-xor}\$body(m, n)))$
 $\rightarrow (collect\text{-value} (indices('y, m, n),$
 $\quad dual\text{-eval}(1, body, bindings, state\text{-}bindings, netlist))$
 $= fv\text{-xor} (collect\text{-value} (indices('a, m, n), bindings),$
 $\quad collect\text{-value} (indices('b, m, n), bindings)))$

EVENT: Enable v-xor\$body-value; name this event 'v-xor\$body-value-off'.

THEOREM: v-xor\$value
 $(v\text{-xor}\& (netlist, n)$
 $\wedge (properp(a) \wedge (length(a) = n))$
 $\wedge properp(b)$
 $\wedge (length(b) = n))$
 $\rightarrow (dual\text{-eval}(0, index('v\text{-xor}, n), append(a, b), state, netlist)$
 $= fv\text{-xor}(a, b))$

EVENT: Enable v-xor\$value; name this event 'v-xor\$value-off'.

DEFINITION:
v-pullup\$body(m, n)
 $= \text{if } n \simeq 0 \text{ then nil}$
 $\quad \text{else cons} (list (index('g, m),$
 $\quad \quad list (index('y, m)),$
 $\quad \quad 'pullup,$
 $\quad \quad list (index('a, m))),$
 $\quad v\text{-pullup}\$body(1 + m, n - 1)) \text{ endif}$

EVENT: Enable v-pullup\$body; name this event 'v-pullup\$body-off'.

DEFINITION:
v-pullup*(n)
 $= cons (index('v\text{-pullup}, n),$
 $\quad cons (indices('a, 0, n),$
 $\quad \quad cons (indices('y, 0, n), cons (v\text{-pullup}\$body(0, n), ' (nil))))$

THEOREM: v-pullup*\$destructure
 $(car(v\text{-pullup}^*(n)) = index('v\text{-pullup}, n))$
 $\wedge (cadr(v\text{-pullup}^*(n)) = indices('a, 0, n))$
 $\wedge (caddr(v\text{-pullup}^*(n)) = indices('y, 0, n))$

$\wedge$ (caddr (v-pullup* (n)) = v-pullup\$body (0, n))
 $\wedge$ (caddr (v-pullup* (n)) = **nil**)

EVENT: Enable v-pullup*; name this event ‘v-pullup*-off’.

EVENT: Enable v-pullup*\$destructure; name this event ‘v-pullup*\$destructure-off’.

DEFINITION:

v-pullup& (netlist, n)
 = ((lookup-module (index ('v-pullup, n), netlist) = v-pullup* (n))
 $\wedge$ pullup& (delete-module (index ('v-pullup, n), netlist)))

EVENT: Enable v-pullup&; name this event ‘v-pullup&-off’.

DEFINITION:

v-pullup\$netlist (n) = cons (v-pullup* (n), PULLUP\$NETLIST)

THEOREM: v-pullup\$unbound-in-body
 $(l < m) \rightarrow$ unbound-in-body (index ('y, l), v-pullup\$body (m, n))

EVENT: Enable v-pullup\$unbound-in-body; name this event ‘v-pullup\$unbound-in-body-off’.

THEOREM: v-pullup\$body-value

(pullup& (netlist) $\wedge$ (body = v-pullup\$body (m, n)))
 $\rightarrow$ (collect-value (indices ('y, m, n),
 dual-eval (1, body, bindings, state-bindings, netlist))
 = v-pullup (collect-value (indices ('a, m, n), bindings)))

EVENT: Enable v-pullup\$body-value; name this event ‘v-pullup\$body-value-off’.

THEOREM: v-pullup\$value

(v-pullup& (netlist, n) $\wedge$ properp (a) $\wedge$ (length (a) = n))
 $\rightarrow$ (dual-eval (0, index ('v-pullup, n), a, state, netlist) = v-pullup (a))

EVENT: Enable v-pullup\$value; name this event ‘v-pullup\$value-off’.

DEFINITION:

v-wire\$body (m, n)
 = **if** $n \simeq 0$ **then** **nil**

```

else cons (list (index ('g, m),
                 list (index ('y, m)),
                 't-wire,
                 list (index ('a, m), index ('b, m))),
            v-wire$body (1 + m, n - 1)) endif

```

EVENT: Enable v-wire\$body; name this event 'v-wire\$body-off'.

DEFINITION:

```

v-wire* (n)
= cons (index ('v-wire, n),
        cons (append (indices ('a, 0, n), indices ('b, 0, n)),
              cons (indices ('y, 0, n), cons (v-wire$body (0, n), ' (nil)))))

```

THEOREM: v-wire*\$destructure

```

(car (v-wire* (n)) = index ('v-wire, n))
^ (cadr (v-wire* (n)) = append (indices ('a, 0, n), indices ('b, 0, n)))
^ (caddr (v-wire* (n)) = indices ('y, 0, n))
^ (cadddr (v-wire* (n)) = v-wire$body (0, n))
^ (caddddr (v-wire* (n)) = nil)

```

EVENT: Enable v-wire*; name this event 'v-wire*-off'.

EVENT: Enable v-wire*\$destructure; name this event 'v-wire*\$destructure-off'.

DEFINITION:

```

v-wire& (netlist, n)
= ((lookup-module (index ('v-wire, n), netlist) = v-wire* (n))
   ^ t-wire& (delete-module (index ('v-wire, n), netlist)))

```

EVENT: Enable v-wire&; name this event 'v-wire&-off'.

DEFINITION: v-wire\$netlist (n) = cons (v-wire* (n), T-WIRE\$NETLIST)

THEOREM: v-wire\$unbound-in-body

```

(l < m) → unbound-in-body (index ('y, l), v-wire$body (m, n))

```

EVENT: Enable v-wire\$unbound-in-body; name this event 'v-wire\$unbound-in-body-off'.

THEOREM: v-wire\$body-value

```

(t-wire& (netlist) ^ (body = v-wire$body (m, n)))
→ (collect-value (indices ('y, m, n),

```

```

      dual-eval (1, body, bindings, state-bindings, netlist))
=   v-wire (collect-value (indices ('a, m, n), bindings),
      collect-value (indices ('b, m, n), bindings)))

```

EVENT: Enable v-wire\$body-value; name this event 'v-wire\$body-value-off'.

THEOREM: v-wire\$value

```

(v-wire& (netlist, n)
  ^ (properp (a) ^ (length (a) = n))
  ^ properp (b)
  ^ (length (b) = n))
→ (dual-eval (0, index ('v-wire, n), append (a, b), state, netlist)
    = v-wire (a, b))

```

EVENT: Enable v-wire\$value; name this event 'v-wire\$value-off'.

DEFINITION: rev-0 (x) = rev1 (x, 0)

DEFINITION:

```

number-to-digit (number) = nth (number, cdr (unpack ('a0123456789)))

```

DEFINITION:

```

number-to-list1 (number)
=   if number ≈ 0 then 0
    else cons (number-to-digit (number mod 10),
              number-to-list1 (number ÷ 10)) endif

```

DEFINITION:

```

number-to-list (number)
=   if number ≈ 0 then cons (cadr (unpack ('a0)), 0)
    else rev-0 (number-to-list1 (number)) endif

```

DEFINITION:

```

lisp-netlist (netlist)
=   if indexp (netlist)
    then pack (append (unpack (i-name (netlist)),
                      cons (cadr (unpack ('a_)),
                          number-to-list (i-num (netlist)))))
    elseif netlist ≈ nil then netlist
    else cons (lisp-netlist (car (netlist)), lisp-netlist (cdr (netlist))) endif

```

DEFINITION:

```

collect-primitives (flg, x0, netlist)
=   case on flg:

```

```

case = 0
then if primp( $x0$ )  $\vee$  ( $\neg$  lookup-module( $x0$ ,  $netlist$ )) then list( $x0$ )
      else flatten-list(collect-primitives(1,
                                         module-occurrences(lookup-module( $x0$ ,
                                                                               $netlist$ )),
                                         delete-module( $x0$ ,
                                                          $netlist$ ))) endif

case = 1
  then if  $x0 \simeq \text{nil}$  then nil
    else cons(collect-primitives(0,
                                occ-function(car( $x0$ )),
                                 $netlist$ ),
              collect-primitives(1, cdr( $x0$ ),  $netlist$ )) endif
otherwise f endcase

```

DEFINITION:

count-primitives($flag$, $x0$, $x1$, $type$, $netlist$)

= **case on** $flag$:

case = 0

then if primp($x0$)

then if primp-lookup($x0$, $type$) **then** primp2($x0$, $type$)

else f endif

elseif listp(lookup-module($x0$, $netlist$))

then count-primitives(1,

module-occurrences(lookup-module($x0$,
 $netlist$)),

0,

$type$,

delete-module($x0$, $netlist$))

else f endif

case = 1

then if $x0 \simeq \text{nil}$ **then** $x1$

elseif count-primitives(0,

occ-function(car($x0$)),

nil,

$type$,

$netlist$)

then count-primitives(1,

cdr($x0$),

$x1$ + count-primitives(0,

occ-function(car($x0$)),

nil,

$type$,

$netlist$),

```

                                type,
                                netlist)

    else f endif
otherwise f endcase

```

EVENT: Enable count-primitives; name this event 'count-primitives-off'.

DEFINITION:
 HALF-ADDER*
 = '(half-adder
 (a b)
 (sum carry)
 ((g0 (sum) b-xor (a b)) (g1 (carry) b-and (a b)))
 nil)

DEFINITION:
 HALF-ADDER\$NETLIST = cons (HALF-ADDER*, B-XOR\$NETLIST $\cup$ B-AND\$NETLIST)

DEFINITION:
 half-adder& (netlist)
 = ((lookup-module ('half-adder, netlist) = HALF-ADDER*)
 $\wedge$ b-xor& (delete-module ('half-adder, netlist))
 $\wedge$ b-and& (delete-module ('half-adder, netlist)))

EVENT: Enable half-adder&; name this event 'half-adder&-off'.

THEOREM: half-adder\$value
 half-adder& (netlist)
 $\rightarrow$ (dual-eval (0, 'half-adder, list (a, b), state, netlist)
 = list (f-xor (a, b), f-and (a, b)))

EVENT: Enable half-adder\$value; name this event 'half-adder\$value-off'.

DEFINITION:
 FULL-ADDER*
 = '(full-adder
 (a b c)
 (sum carry)
 ((t0 (sum1 carry1) half-adder (a b))
 (t1 (sum carry2) half-adder (sum1 c))
 (t2 (carry) b-or (carry1 carry2)))
 nil)

DEFINITION:

FULL-ADDER\$NETLIST

= cons (FULL-ADDER*, HALF-ADDER\$NETLIST $\cup$ B-OR\$NETLIST)

DEFINITION:

full-adder& (*netlist*)

= ((lookup-module ('full-adder, *netlist*) = FULL-ADDER*)
 $\wedge$ half-adder& (delete-module ('full-adder, *netlist*))
 $\wedge$ b-or& (delete-module ('full-adder, *netlist*)))

EVENT: Enable full-adder&; name this event 'full-adder&-off'.

DEFINITION:

f\$full-adder (*a*, *b*, *c*)

= list (f-xor (f-xor (*a*, *b*), *c*), f-or (f-and (*a*, *b*), f-and (f-xor (*a*, *b*), *c*)))

THEOREM: full-adder\$value

full-adder& (*netlist*)

$\rightarrow$ (dual-eval (0, 'full-adder, list (*a*, *b*, *c*), *state*, *netlist*)
= f\$full-adder (*a*, *b*, *c*))

DEFINITION:

full-adder (*a*, *b*, *c*)

= list (b-xor3 (*a*, *b*, *c*), b-or (b-and (*a*, b-or (*b*, *c*)), b-and (*b*, *c*)))

THEOREM: f\$full-adder=full-adder

(boolp (*a*) $\wedge$ boolp (*b*) $\wedge$ boolp (*c*))

$\rightarrow$ (f\$full-adder (*a*, *b*, *c*) = full-adder (*a*, *b*, *c*))

DEFINITION:

M1*

= '(m1
(c1k en sel d q)
(q)
((mux (b) b-if (sel d q))
(latch (a an) fd1 (b c1k))
(tbuf (q) t-buf (en a)))
latch)

DEFINITION:

M1\$NETLIST = cons (M1*, (B-IF\$NETLIST $\cup$ FD1\$NETLIST) $\cup$ T-BUF\$NETLIST)

DEFINITION:

m1& (*netlist*)

= ((lookup-module ('m1, *netlist*) = M1*)

$\wedge$ b-if&(delete-module('m1, netlist))
 $\wedge$ fd1&(delete-module('m1, netlist))
 $\wedge$ t-buf&(delete-module('m1, netlist)))

EVENT: Enable m1&; name this event 'm1&-off'.

THEOREM: m1\$value

m1&(netlist)
 $\rightarrow$ (dual-eval(0, 'm1, list(clk, en, sel, d, q), state, netlist)
 = list(ft-buf(en, state)))

EVENT: Enable m1\$value; name this event 'm1\$value-off'.

THEOREM: m1\$state

m1&(netlist)
 $\rightarrow$ (dual-eval(2, 'm1, list(clk, en, sel, d, q), state, netlist)
 = f-if(sel, d, q))

EVENT: Enable m1\$state; name this event 'm1\$state-off'.

DEFINITION:

M2*
 = '(m2
 (clk en0 en1 sel0 sel1 d0 d1)
 (q)
 ((occ0 (q0) m1 (clk en0 sel0 d0 q))
 (occ1 (q1) m1 (clk en1 sel1 d1 q))
 (wire (q) t-wire (q0 q1)))
 (occ0 occ1))

DEFINITION:

M2\$NETLIST = cons(M2*, M1\$NETLIST $\cup$ T-WIRE\$NETLIST)

DEFINITION:

m2&(netlist)
 = ((lookup-module('m2, netlist) = M2*)
 $\wedge$ m1&(delete-module('m2, netlist))
 $\wedge$ t-wire&(delete-module('m2, netlist)))

EVENT: Enable m2&; name this event 'm2&-off'.

THEOREM: m2\$value

m2&(netlist)
 $\rightarrow$ (dual-eval(0, 'm2, list(clk, en0, en1, sel0, sel1, d0, d1), state, netlist)
 = list(ft-wire(ft-buf(en0, car(state)), ft-buf(en1, cadr(state)))))

EVENT: Enable m2\$value; name this event 'm2\$value-off'.

THEOREM: m2\$state

```
m2& (netlist)
→ (dual-eval (2, 'm2, list (clk, en0, en1, sel0, sel1, d0, d1), state, netlist)
   = list (f-if (sel0,
                 d0,
                 ft-wire (ft-buf (en0, car (state)),
                           ft-buf (en1, cadr (state))))) ,
            f-if (sel1,
                 d1,
                 ft-wire (ft-buf (en0, car (state)),
                           ft-buf (en1, cadr (state)))))
```

EVENT: Enable m2\$state; name this event 'm2\$state-off'.

DEFINITION:

```
bv-adder (c, a, b)
= if a ≈ nil then list (c)
  else cons (xor (c, xor (car (a), car (b))),
             bv-adder ((car (a) ∧ car (b))
                       ∨ (car (a) ∧ c)
                       ∨ (car (b) ∧ c),
                       cdr (a),
                       cdr (b))) endif
```

DEFINITION:

```
bv-adder-body (m, n)
= if n ≈ 0 then nil
  else cons (list (index ('g, m),
                   list (index ('sum, m), index ('carry, 1 + m)),
                   'full-adder,
                   list (index ('a, m), index ('b, m), index ('carry, m))),
            bv-adder-body (1 + m, n - 1)) endif
```

EVENT: Enable bv-adder-body; name this event 'bv-adder-body-off'.

DEFINITION:

```
bv-adder* (n)
= cons (index ('bv-adder, n),
       cons (cons (index ('carry, 0),
                   append (indices ('a, 0, n), indices ('b, 0, n))),
             cons (append (indices ('sum, 0, n), list (index ('carry, n))),
                   cons (bv-adder-body (0, n), ' (nil))))))
```


THEOREM: $\text{bv-adder}^*\$destructure$
 $(\text{car}(\text{bv-adder}^*(n)) = \text{index}(' \mathbf{bv-adder}, n))$
 $\wedge (\text{cadr}(\text{bv-adder}^*(n))$
 $\quad = \text{cons}(\text{index}(' \mathbf{carry}, 0),$
 $\quad \quad \text{append}(\text{indices}(' \mathbf{a}, 0, n), \text{indices}(' \mathbf{b}, 0, n))))$
 $\wedge (\text{caddr}(\text{bv-adder}^*(n))$
 $\quad = \text{append}(\text{indices}(' \mathbf{sum}, 0, n), \text{list}(\text{index}(' \mathbf{carry}, n))))$
 $\wedge (\text{caddr}(\text{bv-adder}^*(n)) = \text{bv-adder-body}(0, n))$
 $\wedge (\text{caddr}(\text{bv-adder}^*(n)) = \mathbf{nil})$

EVENT: Enable bv-adder^* ; name this event ' $\text{bv-adder}^*\text{-off}$ '.

EVENT: Enable $\text{bv-adder}^*\$destructure$; name this event ' $\text{bv-adder}^*\$destructure\text{-off}$ '.

DEFINITION:
 $\text{bv-adder}\&(\text{netlist}, n)$
 $= ((\text{lookup-module}(\text{index}(' \mathbf{bv-adder}, n), \text{netlist}) = \text{bv-adder}^*(n))$
 $\quad \wedge \text{full-adder}\&(\text{delete-module}(\text{index}(' \mathbf{bv-adder}, n), \text{netlist})))$

EVENT: Enable $\text{bv-adder}\&$; name this event ' $\text{bv-adder}\&\text{-off}$ '.

DEFINITION:
 $\text{bv-adder}\$netlist(n) = \text{cons}(\text{bv-adder}^*(n), \text{FULL-ADDER}\$NETLIST)$

EVENT: Enable $\text{bv-adder}\$netlist$; name this event ' $\text{bv-adder}\$netlist\text{-off}$ '.

DEFINITION:
 $\text{bv-adder-body}\$induction(m, n, \text{bindings}, \text{state-bindings}, \text{netlist})$
 $= \text{if } n \simeq 0 \text{ then } \text{bindings}$
 $\quad \text{else } \text{bv-adder-body}\$induction(1 + m,$
 $\quad \quad n - 1,$
 $\quad \quad \text{append}(\text{pairlist}(\text{list}(\text{index}(' \mathbf{sum},$
 $\quad \quad \quad m),$
 $\quad \quad \quad \text{index}(' \mathbf{carry},$
 $\quad \quad \quad 1 + m))),$
 $\quad \quad \text{dual-eval}(0,$
 $\quad \quad \quad ' \mathbf{full-adder},$
 $\quad \quad \quad \text{collect-value}(\text{list}(\text{index}(' \mathbf{a},$
 $\quad \quad \quad \quad m),$
 $\quad \quad \quad \quad \text{index}(' \mathbf{b},$
 $\quad \quad \quad \quad \quad m),$

$\text{index}(' \text{carry}, m),$
 $\text{bindings}),$
 $\text{value}(\text{index}(' \text{g}, m),$
 $\text{state-bindings}),$
 $\text{netlist}),$
 $\text{bindings}),$
 $\text{state-bindings},$
 $\text{netlist}) \text{ endif}$

THEOREM: $\text{bv-adder\$unbound-in-body-sum}$
 $(l < m) \rightarrow \text{unbound-in-body}(\text{index}(' \text{sum}, l), \text{bv-adder-body}(m, n))$

THEOREM: $\text{bv-adder\$unbound-in-body-carry}$
 $(l < (1 + m)) \rightarrow \text{unbound-in-body}(\text{index}(' \text{carry}, l), \text{bv-adder-body}(m, n))$

EVENT: Enable $\text{bv-adder\$unbound-in-body-sum}$; name this event ‘ $\text{bv-adder\$unbound-in-body-sum-off}$ ’.

EVENT: Enable $\text{bv-adder\$unbound-in-body-carry}$; name this event ‘ $\text{bv-adder\$unbound-in-body-carry-off}$ ’.

THEOREM: $\text{bv-adder-body\$value}$
 $(\text{full-adder\& } (netlist))$
 $\wedge \text{ boolp}(\text{value}(\text{index}(' \text{carry}, m), \text{bindings}))$
 $\wedge \text{ bvp}(\text{collect-value}(\text{indices}(' \text{a}, m, n), \text{bindings}))$
 $\wedge \text{ bvp}(\text{collect-value}(\text{indices}(' \text{b}, m, n), \text{bindings}))$
 $\rightarrow (\text{collect-value}(\text{append}(\text{indices}(' \text{sum}, m, n),$
 $\text{list}(\text{index}(' \text{carry}, n + m))),$
 $\text{dual-eval}(1,$
 $\text{bv-adder-body}(m, n),$
 $\text{bindings},$
 $\text{state-bindings},$
 $\text{netlist}))$
 $= \text{bv-adder}(\text{value}(\text{index}(' \text{carry}, m), \text{bindings}),$
 $\text{collect-value}(\text{indices}(' \text{a}, m, n), \text{bindings}),$
 $\text{collect-value}(\text{indices}(' \text{b}, m, n), \text{bindings}))$

THEOREM: $\text{bv-adder-body-special-case\$value}$
 $(\text{full-adder\& } (netlist))$
 $\wedge \text{ boolp}(\text{value}(\text{index}(' \text{carry}, 0), \text{bindings}))$
 $\wedge \text{ bvp}(\text{collect-value}(\text{indices}(' \text{a}, 0, n), \text{bindings}))$
 $\wedge \text{ bvp}(\text{collect-value}(\text{indices}(' \text{b}, 0, n), \text{bindings}))$

$$\begin{aligned}
&\rightarrow (\text{collect-value}(\text{append}(\text{indices}(' \text{sum}, 0, n), \text{list}(\text{index}(' \text{carry}, n))), \\
&\quad \text{dual-eval}(1, \\
&\quad \quad \text{bv-adder-body}(0, n), \\
&\quad \quad \text{bindings}, \\
&\quad \quad \text{state-bindings}, \\
&\quad \quad \text{netlist})) \\
&= \text{bv-adder}(\text{value}(\text{index}(' \text{carry}, 0), \text{bindings}), \\
&\quad \text{collect-value}(\text{indices}(' \text{a}, 0, n), \text{bindings}), \\
&\quad \text{collect-value}(\text{indices}(' \text{b}, 0, n), \text{bindings}))
\end{aligned}$$

THEOREM: bv-adder\$value

$$\begin{aligned}
&(\text{bv-adder}\&(\text{netlist}, n) \\
&\wedge \text{boolp}(c) \\
&\wedge \text{bvp}(a) \\
&\wedge \text{bvp}(b) \\
&\wedge (\text{length}(a) = n) \\
&\wedge (\text{length}(b) = n)) \\
&\rightarrow (\text{dual-eval}(0, \text{index}(' \text{bv-adder}, n), \text{cons}(c, \text{append}(a, b)), \text{state}, \text{netlist}) \\
&\quad = \text{bv-adder}(c, a, b))
\end{aligned}$$

EVENT: Enable bv-adder-body-special-case\$value; name this event 'bv-adder-body-special-case\$value-off'.

EVENT: Enable bv-adder-body\$value; name this event 'bv-adder-body\$value-off'.

EVENT: Enable bv-adder\$value; name this event 'bv-adder\$value-off'.

DEFINITION:

$$\begin{aligned}
&\text{v-sum}(c, a, b) \\
&= \text{if } a \simeq \text{nil} \text{ then nil} \\
&\quad \text{else cons}(\text{b-xor}(c, \text{b-xor}(\text{car}(a), \text{car}(b))), \\
&\quad \quad \text{v-sum}(\text{b-or3}(\text{b-and}(\text{car}(a), \text{car}(b)), \\
&\quad \quad \quad \text{b-and}(\text{car}(a), c), \\
&\quad \quad \quad \text{b-and}(\text{car}(b), c)), \\
&\quad \quad \text{cdr}(a), \\
&\quad \quad \text{cdr}(b))) \text{ endif}
\end{aligned}$$

EVENT: Enable v-sum; name this event 'v-sum-off'.

THEOREM: bvp-sum

$$\text{bvp}(\text{v-sum}(c, a, b))$$

THEOREM: length-sum
 $\text{length}(\text{v-sum}(c, a, b)) = \text{length}(a)$

THEOREM: v-sum-congruence
 $c \rightarrow ((\text{v-sum}(c, a, b) = \text{v-sum}(\mathbf{t}, a, b)) = \mathbf{t})$

THEOREM: a-1+1=a
 $(\text{bvp}(a) \wedge c) \rightarrow (\text{v-sum}(c, \text{v-not}(\text{nat-to-v}(0, \text{length}(a))), a) = a)$

DEFINITION:
 $\text{v-carry}(c, a, b)$
 $=$ **if** $a \simeq \mathbf{nil}$ **then** $\text{boolfix}(c)$
 else $\text{v-carry}((\text{car}(a) \wedge \text{car}(b))$
 $\vee ((\text{car}(a) \vee \text{car}(b)) \wedge c),$
 $\text{cdr}(a),$
 $\text{cdr}(b))$ **endif**

EVENT: Enable v-carry; name this event ‘v-carry-off’.

DEFINITION:
 $\text{v-propagate}(a, b)$
 $=$ **if** $a \simeq \mathbf{nil}$ **then** $\mathbf{t}$
 else $(\text{car}(a) \vee \text{car}(b)) \wedge \text{v-propagate}(\text{cdr}(a), \text{cdr}(b))$ **endif**

EVENT: Enable v-propagate; name this event ‘v-propagate-off’.

DEFINITION:
 $\text{v-generate}(a, b)$
 $=$ **if** $a \simeq \mathbf{nil}$ **then** $\mathbf{f}$
 else $\text{v-generate}(\text{cdr}(a), \text{cdr}(b))$
 $\vee (\text{car}(a) \wedge \text{car}(b) \wedge \text{v-propagate}(\text{cdr}(a), \text{cdr}(b)))$ **endif**

EVENT: Enable v-generate; name this event ‘v-generate-off’.

DEFINITION: $\text{f\$t-carry}(c, \text{prop}, \text{gen}) = \text{f-or}(\text{f-and}(c, \text{prop}), \text{gen})$

EVENT: Enable f\$t-carry; name this event ‘f\$t-carry-off’.

DEFINITION: $\text{t-carry}(c, \text{prop}, \text{gen}) = \text{b-not}(\text{ao6}(c, \text{prop}, \text{gen}))$

EVENT: Enable t-carry; name this event ‘t-carry-off’.

THEOREM: f\$t-carry=t-carry
 $(\text{boolp}(c) \wedge \text{boolp}(\text{prop}) \wedge \text{boolp}(\text{gen}))$
 $\rightarrow (\text{f\$t-carry}(c, \text{prop}, \text{gen}) = \text{t-carry}(c, \text{prop}, \text{gen}))$

THEOREM: t-carry-congruence

$$x \rightarrow ((\text{t-carry}(x, p, g) = \text{t-carry}(\mathbf{t}, p, g)) = \mathbf{t})$$

DEFINITION:

T-CARRY*

$$\begin{aligned} = & \text{'(t-carry} \\ & \quad (\text{c prop gen}) \\ & \quad (\text{z}) \\ & \quad ((\text{g0 (z-)} \text{ ao6 (c prop gen)}) (\text{g1 (z) b-not (z-)})) \\ & \quad \text{nil)} \end{aligned}$$

DEFINITION:

t-carry& (*netlist*)

$$\begin{aligned} = & ((\text{lookup-module}(\text{'t-carry}, \text{netlist}) = \text{T-CARRY*}) \\ & \wedge \text{ao6\& (delete-module}(\text{'t-carry}, \text{netlist})) \\ & \wedge \text{b-not\& (delete-module}(\text{'t-carry}, \text{netlist}))) \end{aligned}$$

EVENT: Enable t-carry&; name this event 't-carry&-off'.

DEFINITION:

$$\text{T-CARRY\$NETLIST} = \text{cons}(\text{T-CARRY*}, \text{AO6\$NETLIST} \cup \text{B-NOT\$NETLIST})$$

THEOREM: t-carry\$value

$$\begin{aligned} & \text{t-carry\& (netlist)} \\ \rightarrow & (\text{dual-eval}(0, \text{'t-carry}, \text{list}(c, \text{prop}, \text{gen}), \text{state}, \text{netlist}) \\ & = \text{list}(\text{f\$t-carry}(c, \text{prop}, \text{gen}))) \end{aligned}$$

EVENT: Enable t-carry\$value; name this event 't-carry\$value-off'.

THEOREM: v-append-propagate

$$\begin{aligned} & (\text{length}(a) = \text{length}(c)) \\ \rightarrow & (\text{v-propagate}(\text{append}(a, b), \text{append}(c, d)) \\ & = (\text{v-propagate}(a, c) \wedge \text{v-propagate}(b, d))) \end{aligned}$$

THEOREM: v-propagate-append

$$\begin{aligned} & (\text{length}(a) = \text{length}(c)) \\ \rightarrow & (\text{b-and}(\text{v-propagate}(a, c), \text{v-propagate}(b, d)) \\ & = \text{v-propagate}(\text{append}(a, b), \text{append}(c, d))) \end{aligned}$$

EVENT: Enable v-propagate-append; name this event 'v-propagate-append-off'.

THEOREM: generate-append

$$\begin{aligned} & (\text{length}(a) = \text{length}(c)) \\ \rightarrow & (\text{t-carry}(\text{v-generate}(a, c), \text{v-propagate}(b, d), \text{v-generate}(b, d)) \\ & = \text{v-generate}(\text{append}(a, b), \text{append}(c, d))) \end{aligned}$$

THEOREM: append-sum
 $((\text{length}(a) = \text{length}(b)) \wedge (c2 = \text{v-carry}(c1, a, b)))$
 $\rightarrow (\text{append}(\text{v-sum}(c1, a, b), \text{v-sum}(c2, c, d)))$
 $= \text{v-sum}(c1, \text{append}(a, c), \text{append}(b, d))$

THEOREM: t-carry-p-g-carry
 $\text{t-carry}(c, \text{v-propagate}(a, b), \text{v-generate}(a, b)) = \text{v-carry}(c, a, b)$

THEOREM: v-adder-output=v-sum
 $\text{v-adder-output}(c, a, b) = \text{v-sum}(c, a, b)$

THEOREM: v-adder-carry-out=v-carry
 $\text{v-adder-carry-out}(c, a, b) = \text{v-carry}(c, a, b)$

DEFINITION:

$\text{tv-adder}(c, a, b, \text{tree})$
 $=$ **if** $\text{tree} \simeq \text{nil}$
 then $\text{list}(\text{b-or}(\text{car}(a), \text{car}(b)),$
 $\text{b-and}(\text{car}(a), \text{car}(b)),$
 $\text{b-xor}(\text{car}(a), \text{b-xor}(\text{car}(b), c)))$
 else $\text{cons}(\text{b-and}(\text{car}(\text{tv-adder}(c,$
 $\text{tfirstn}(a, \text{tree}),$
 $\text{tfirstn}(b, \text{tree}),$
 $\text{car}(\text{tree}))),$
 $\text{car}(\text{tv-adder}(\text{t-carry}(c,$
 $\text{car}(\text{tv-adder}(c,$
 $\text{tfirstn}(a,$
 $\text{tree}),$
 $\text{tfirstn}(b,$
 $\text{tree}),$
 $\text{car}(\text{tree}))),$
 $\text{cadr}(\text{tv-adder}(c,$
 $\text{tfirstn}(a,$
 $\text{tree}),$
 $\text{tfirstn}(b,$
 $\text{tree}),$
 $\text{car}(\text{tree}))),$
 $\text{trestn}(a, \text{tree}),$
 $\text{trestn}(b, \text{tree}),$
 $\text{cdr}(\text{tree}))),$
 $\text{cons}(\text{t-carry}(\text{cadr}(\text{tv-adder}(c,$
 $\text{tfirstn}(a, \text{tree}),$
 $\text{tfirstn}(b, \text{tree}),$
 $\text{car}(\text{tree}))),$
 $\text{car}(\text{tv-adder}(\text{t-carry}(c,$

```

                                car (tv-adder (c,
                                                tfirstn (a,
                                                            tree),
                                                tfirstn (b,
                                                            tree),
                                                car (tree))),
                                cadr (tv-adder (c,
                                                tfirstn (a,
                                                            tree),
                                                tfirstn (b,
                                                            tree),
                                                car (tree))),
                                trestn (a, tree),
                                trestn (b, tree),
                                cdr (tree))),
                                cadr (tv-adder (t-carry (c,
                                                        car (tv-adder (c,
                                                                    tfirstn (a,
                                                                    tree),
                                                                    tfirstn (b,
                                                                    tree),
                                                                    car (tree))),
                                                        cadr (tv-adder (c,
                                                                    tfirstn (a,
                                                                    tree),
                                                                    tfirstn (b,
                                                                    tree),
                                                                    car (tree))),
                                                        trestn (a, tree),
                                                        trestn (b, tree),
                                                        cdr (tree))),
                                append (caddr (tv-adder (c,
                                                        tfirstn (a, tree),
                                                        tfirstn (b, tree),
                                                        car (tree))),
                                caddr (tv-adder (t-carry (c,
                                                        car (tv-adder (c,
                                                                    tfirstn (a,
                                                                    tree),
                                                                    tfirstn (b,
                                                                    tree),
                                                                    car (tree))),
                                                        cadr (tv-adder (c,
                                                                    tfirstn (a,

```

```

                                tree),
tfirstn (b,
                                tree),
                                car (tree))))),
trestn (a, tree),
trestn (b, tree),
cdr (tree)))))) endif

```

EVENT: Enable tv-adder; name this event ‘tv-adder-off’.

THEOREM: car-tv-adder-as-p
 $((\text{length}(a) = \text{tree-size}(tree)) \wedge (\text{length}(b) = \text{tree-size}(tree)))$
 $\rightarrow (\text{car}(\text{tv-adder}(c, a, b, tree)) = \text{v-propagate}(a, b))$

THEOREM: cadr-tv-adder-as-g
 $((\text{length}(a) = \text{tree-size}(tree)) \wedge (\text{length}(b) = \text{tree-size}(tree)))$
 $\rightarrow (\text{cadr}(\text{tv-adder}(c, a, b, tree)) = \text{v-generate}(a, b))$

THEOREM: cddr-tv-adder-as-sum
 $((\text{length}(a) = \text{tree-size}(tree)) \wedge (\text{length}(b) = \text{tree-size}(tree)))$
 $\rightarrow (\text{cddr}(\text{tv-adder}(c, a, b, tree)) = \text{v-sum}(c, a, b))$

THEOREM: equality-of-three-element-lists
 $((x \neq 0) \wedge (y \neq 0) \wedge (z \neq 0))$
 $\rightarrow ((w = \text{cons}(x, \text{cons}(y, z)))$
 $= ((\text{car}(w) = x) \wedge (\text{cadr}(w) = y) \wedge (\text{cddr}(w) = z)))$

THEOREM: tv-adder-as-p-g-sum
 $((\text{length}(a) = \text{tree-size}(tree)) \wedge (\text{length}(b) = \text{tree-size}(tree)))$
 $\rightarrow (\text{tv-adder}(c, a, b, tree)$
 $= \text{cons}(\text{v-propagate}(a, b), \text{cons}(\text{v-generate}(a, b), \text{v-sum}(c, a, b))))$

EVENT: Enable car-tv-adder-as-p; name this event ‘car-tv-adder-as-p-off’.

EVENT: Enable cadr-tv-adder-as-g; name this event ‘cadr-tv-adder-as-g-off’.

EVENT: Enable cddr-tv-adder-as-sum; name this event ‘cddr-tv-adder-as-sum-off’.

EVENT: Enable tv-adder-as-p-g-sum; name this event ‘tv-adder-as-p-g-sum-off’.

DEFINITION:
tv-if-body (*tree*)


```

=  if tree  $\simeq$  nil
    then list (list ('leaf,
                    list (index ('out, 0)),
                    'b-if,
                    list ('c, index ('a, 0), index ('b, 0))))
    else append (if (tree-height (tree) mod 3)  $\simeq$  0
                  then '((c-buf (c-buf) b-buf (c)))
                  else nil endif,
                  list (list ('left,
                              tfirstn (indices ('out, 0, tree-size (tree)),
                                                tree),
                              index ('tv-if, tree-number (car (tree))),
                              cons (if (tree-height (tree) mod 3)  $\simeq$  0
                                      then 'c-buf
                                      else 'c endif,
                                      append (tfirstn (indices ('a,
                                                                0,
                                                                tree-size (tree)),
                                                                tree),
                                                                tfirstn (indices ('b,
                                                                0,
                                                                tree-size (tree)),
                                                                tree))))),
                              list ('right,
                                  trestn (indices ('out, 0, tree-size (tree)),
                                                tree),
                                  index ('tv-if, tree-number (cdr (tree))),
                                  cons (if (tree-height (tree) mod 3)  $\simeq$  0
                                          then 'c-buf
                                          else 'c endif,
                                          append (trestn (indices ('a,
                                                                    0,
                                                                    tree-size (tree)),
                                                                    tree),
                                                                    trestn (indices ('b,
                                                                    0,
                                                                    tree-size (tree)),
                                                                    tree)))))) endif

```

DEFINITION:

```

tv-if* (tree)
=  cons (index ('tv-if, tree-number (tree)),
        cons (cons ('c,
                    append (indices ('a, 0, tree-size (tree)),

```

$$\begin{aligned} & \text{indices}('b, 0, \text{tree-size}(tree))), \\ & \text{cons}(\text{indices}('out, 0, \text{tree-size}(tree)), \\ & \quad \text{cons}(\text{tv-if-body}(tree), '(\text{nil})))) \end{aligned}$$

THEOREM: $\text{tv-if}^*\$destructure$
 $(\text{car}(\text{tv-if}^*(tree)) = \text{index}('tv-if, \text{tree-number}(tree)))$
 $\wedge (\text{cadr}(\text{tv-if}^*(tree))$
 $\quad = \text{cons}('c,$
 $\quad \quad \text{append}(\text{indices}('a, 0, \text{tree-size}(tree)),$
 $\quad \quad \quad \text{indices}('b, 0, \text{tree-size}(tree))))$
 $\wedge (\text{caddr}(\text{tv-if}^*(tree)) = \text{indices}('out, 0, \text{tree-size}(tree)))$
 $\wedge (\text{cadddr}(\text{tv-if}^*(tree)) = \text{tv-if-body}(tree))$
 $\wedge (\text{caddddr}(\text{tv-if}^*(tree)) = \text{nil})$

EVENT: Enable tv-if^* ; name this event ' $\text{tv-if}^*\text{-off}$ '.

EVENT: Enable $\text{tv-if}^*\$destructure$; name this event ' $\text{tv-if}^*\$destructure\text{-off}$ '.

DEFINITION:

$\text{tv-if}\& (netlist, tree)$
 $= \text{if } tree \simeq \text{nil}$
 $\quad \text{then } (\text{lookup-module}(\text{index}('tv-if, \text{tree-number}(tree)), netlist)$
 $\quad \quad = \text{tv-if}^*(tree))$
 $\quad \quad \wedge \text{b-if}\& (\text{delete-module}(\text{index}('tv-if, \text{tree-number}(tree)),$
 $\quad \quad \quad netlist))$
 $\quad \text{else } (\text{lookup-module}(\text{index}('tv-if, \text{tree-number}(tree)), netlist)$
 $\quad \quad = \text{tv-if}^*(tree))$
 $\quad \quad \wedge \text{b-buf}\& (\text{delete-module}(\text{index}('tv-if,$
 $\quad \quad \quad \text{tree-number}(tree)),$
 $\quad \quad \quad netlist))$
 $\quad \quad \wedge \text{tv-if}\& (\text{delete-module}(\text{index}('tv-if,$
 $\quad \quad \quad \text{tree-number}(tree)),$
 $\quad \quad \quad netlist),$
 $\quad \quad \quad \text{car}(tree))$
 $\quad \quad \wedge \text{tv-if}\& (\text{delete-module}(\text{index}('tv-if,$
 $\quad \quad \quad \text{tree-number}(tree)),$
 $\quad \quad \quad netlist),$
 $\quad \quad \quad \text{cdr}(tree)) \text{ endif}$

EVENT: Enable $\text{tv-if}\&$; name this event ' $\text{tv-if}\&\text{-off}$ '.

DEFINITION:

$\text{tv-if}\$netlist (tree)$

```

=  if  $tree \simeq \mathbf{nil}$  then cons (tv-if* ( $tree$ ), B-IF$NETLIST)
    else cons (tv-if* ( $tree$ ),
                tv-if$netlist (car ( $tree$ ))
                 $\cup$  (tv-if$netlist (cdr ( $tree$ ))  $\cup$  B-BUF$NETLIST)) endif

```

DEFINITION:

tv-if-induction ($tree, c, a, b, state, netlist$)

```

=  if  $tree \simeq \mathbf{nil}$  then t
    else tv-if-induction (car ( $tree$ ),
                          c,
                          tfirstn ( $a, tree$ ),
                          tfirstn ( $b, tree$ ),
                          0,
                          delete-module (index ('tv-if,
                                                tree-number ( $tree$ )),
                                                netlist))
     $\wedge$  tv-if-induction (car ( $tree$ ),
                        X,
                        tfirstn ( $a, tree$ ),
                        tfirstn ( $b, tree$ ),
                        0,
                        delete-module (index ('tv-if,
                                                tree-number ( $tree$ )),
                                                netlist))
     $\wedge$  tv-if-induction (cdr ( $tree$ ),
                        c,
                        trestn ( $a, tree$ ),
                        trestn ( $b, tree$ ),
                        0,
                        delete-module (index ('tv-if,
                                                tree-number ( $tree$ )),
                                                netlist))
     $\wedge$  tv-if-induction (cdr ( $tree$ ),
                        X,
                        trestn ( $a, tree$ ),
                        trestn ( $b, tree$ ),
                        0,
                        delete-module (index ('tv-if,
                                                tree-number ( $tree$ )),
                                                netlist)) endif

```

THEOREM: tv-if-lemma-crock

```

tv-if& (delete-module (index ('tv-if, tree-number ( $tree$ )), netlist), car ( $tree$ ))
 $\rightarrow$  (collect-value (indices ('out, 0,  $n$ ), alist)

```

```

=  append (collect-value (firstn (tree-size (car (tree))),
                                indices ('out, 0, n)),
          alist),
      collect-value (restn (tree-size (car (tree))),
                    indices ('out, 0, n)),
          alist)))

```

EVENT: Enable tv-if-lemma-crock; name this event 'tv-if-lemma-crock-off'.

THEOREM: tv-if\$value
 (tv-if& (netlist, tree)
 $\wedge$ properp (a)
 $\wedge$ properp (b)
 $\wedge$ (length (a) = tree-size (tree))
 $\wedge$ (length (b) = tree-size (tree)))
 $\rightarrow$ (dual-eval (0,
 index ('tv-if, tree-number (tree)),
 cons (c, append (a, b)),
 state,
 netlist)
 = fv-if (c, a, b))

EVENT: Enable tv-if\$value; name this event 'tv-if\$value-off'.

DEFINITION:
 t-or-nor-body (tree, parity)
 = if tree $\simeq$ nil
 then list (list ('leaf,
 ' (out),
 if parity then 'b-not
 else 'b-buf endif,
 list (index ('a, 0))))
 elseif (car (tree) $\simeq$ nil) $\wedge$ (cdr (tree) $\simeq$ nil)
 then list (list ('leaf,
 ' (out),
 if parity then 'b-nor
 else 'b-or endif,
 list (index ('a, 0), index ('a, 1))))
 else list (list ('left,
 ' (left-out),
 index (if parity then 't-or
 else 't-nor endif,
 tree-number (car (tree))))

```

    tfirstn (indices ('a, 0, tree-size (tree)), tree)),
  list ('right,
    ' (right-out),
    index (if parity then 't-or
      else 't-nor endif,
      tree-number (cdr (tree))),
    trestn (indices ('a, 0, tree-size (tree)), tree)),
  cons ('output,
    cons (' (out),
      cons (if parity then 'b-nor
        else 'b-nand endif,
        ' ((left-out right-out)))))) endif

```

DEFINITION:

```

t-or-nor* (tree, parity)
= cons (index (if parity then 't-nor
  else 't-or endif,
  tree-number (tree)),
  cons (indices ('a, 0, tree-size (tree)),
    cons (' (out), cons (t-or-nor-body (tree, parity), ' (nil)))))

```

THEOREM: t-or-nor*\$destructure

```

(car (t-or-nor* (tree, parity))
= index (if parity then 't-nor
  else 't-or endif,
  tree-number (tree)))
^ (cadr (t-or-nor* (tree, parity)) = indices ('a, 0, tree-size (tree)))
^ (caddr (t-or-nor* (tree, parity)) = ' (out))
^ (cadddr (t-or-nor* (tree, parity)) = t-or-nor-body (tree, parity))
^ (caddddr (t-or-nor* (tree, parity)) = nil)

```

EVENT: Enable t-or-nor*; name this event 't-or-nor*-off'.

EVENT: Enable t-or-nor*\$destructure; name this event 't-or-nor*\$destructure-off'.

DEFINITION:

```

t-or-nor& (netlist, tree, parity)
= if (tree ≃ nil) ∨ ((car (tree) ≃ nil) ∧ (cdr (tree) ≃ nil))
  then (lookup-module (index (if parity then 't-nor
    else 't-or endif,
    tree-number (tree)),
    netlist)
  = t-or-nor* (tree, parity))

```

```

      ^ b-not& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
      ^ b-buf& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
      ^ b-nor& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
      ^ b-or& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
      ^ b-nand& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
else (lookup-module (index (if parity then 't-nor
                             else 't-or endif,
                             tree-number (tree)),
                             netlist)
      = t-or-nor* (tree, parity))
      ^ t-or-nor& (delete-module (index (if parity
                                     then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist),
                  car (tree),
                  ¬ parity)
      ^ t-or-nor& (delete-module (index (if parity
                                     then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist),
                  cdr (tree),
                  ¬ parity)
      ^ b-not& (delete-module (index (if parity then 't-nor
                                     else 't-or endif,
                                     tree-number (tree)),
                                     netlist))
      ^ b-buf& (delete-module (index (if parity then 't-nor

```

```

                                else 't-or endif,
                                tree-number (tree)),
                                netlist))
  ∧ b-nor& (delete-module (index (if parity then 't-nor
                                else 't-or endif,
                                tree-number (tree)),
                                netlist))
  ∧ b-or& (delete-module (index (if parity then 't-nor
                                else 't-or endif,
                                tree-number (tree)),
                                netlist))
  ∧ b-nand& (delete-module (index (if parity then 't-nor
                                else 't-or endif,
                                tree-number (tree)),
                                netlist)) endif

```

EVENT: Enable t-or-nor&; name this event 't-or-nor&-off'.

DEFINITION:

```

t-or-nor$netlist (tree, parity)
= if (tree ≃ nil) ∨ ((car (tree) ≃ nil) ∧ (cdr (tree) ≃ nil))
  then cons (t-or-nor* (tree, parity),
            ((B-NOT$NETLIST ∪ B-BUF$NETLIST)
             ∪ (B-NOR$NETLIST ∪ B-OR$NETLIST))
             ∪ B-NAND$NETLIST)
  else cons (t-or-nor* (tree, parity),
            t-or-nor$netlist (car (tree), ¬ parity)
            ∪ t-or-nor$netlist (cdr (tree), ¬ parity)) endif

```

DEFINITION:

```

t-or-nor-induction (tree, parity, call-name, a, state, netlist)
= if (tree ≃ nil) ∨ ((car (tree) ≃ nil) ∧ (cdr (tree) ≃ nil)) then t
  else t-or-nor-induction (car (tree),
                            ¬ parity,
                            if ¬ parity then 't-nor
                            else 't-or endif,
                            tfirstn (a, tree),
                            0,
                            delete-module (index (if parity
                                                    then 't-nor
                                                    else 't-or endif,
                                                    tree-number (tree)),
                                                    netlist))
  ∧ t-or-nor-induction (cdr (tree),

```

```

 $\neg$  parity,
if  $\neg$  parity then 't-nor
else 't-or endif,
trestn(a, tree),
0,
delete-module(index(if parity
                    then 't-nor
                    else 't-or endif,
                    tree-number(tree)),
netlist)) endif

```

DEFINITION:

```

tr-or-nor(a, parity, tree)
= if tree  $\simeq$  nil
  then if parity then f-not(car(a))
    else f-buf(car(a)) endif
  elseif (car(tree)  $\simeq$  nil)  $\wedge$  (cdr(tree)  $\simeq$  nil)
  then if parity then f-nor(car(a), cadr(a))
    else f-or(car(a), cadr(a)) endif
  elseif parity
  then f-nor(tr-or-nor(tfirstn(a, tree),  $\neg$  parity, car(tree)),
            tr-or-nor(trestn(a, tree),  $\neg$  parity, cdr(tree)))
  else f-nand(tr-or-nor(tfirstn(a, tree),  $\neg$  parity, car(tree)),
            tr-or-nor(trestn(a, tree),  $\neg$  parity, cdr(tree))) endif

```

EVENT: Enable tr-or-nor; name this event 'tr-or-nor-off'.

THEOREM: t-or-nor\$value

```

(t-or-nor&(netlist, tree, parity)
 $\wedge$  (call-name = if parity then 't-nor
    else 't-or endif)
 $\wedge$  properp(a)
 $\wedge$  (length(a) = tree-size(tree)))
 $\rightarrow$  (dual-eval(0, index(call-name, tree-number(tree)), a, state, netlist)
    = list(tr-or-nor(a, parity, tree)))

```

EVENT: Enable t-or-nor\$value; name this event 't-or-nor\$value-off'.

DEFINITION:

```

btr-or-nor(a, parity, tree)
= if tree  $\simeq$  nil
  then if parity then b-not(car(a))
    else b-buf(car(a)) endif
  elseif (car(tree)  $\simeq$  nil)  $\wedge$  (cdr(tree)  $\simeq$  nil)

```



```

then if parity then b-nor(car(a), cadr(a))
      else b-or(car(a), cadr(a)) endif
elseif parity
then b-nor(btr-or-nor(tfirstrn(a, tree),  $\neg$  parity, car(tree)),
            btr-or-nor(trestn(a, tree),  $\neg$  parity, cdr(tree)))
else b-nand(btr-or-nor(tfirstrn(a, tree),  $\neg$  parity, car(tree)),
            btr-or-nor(trestn(a, tree),  $\neg$  parity, cdr(tree))) endif

```

EVENT: Enable btr-or-nor; name this event ‘btr-or-nor-off’.

THEOREM: tr-or-nor=btr-or-nor
 $(bvp(a) \wedge (length(a) = tree-size(tree)))$
 $\rightarrow (tr-or-nor(a, parity, tree) = btr-or-nor(a, parity, tree))$

THEOREM: btr-or-is-v-zerop
 $(length(a) = tree-size(tree))$
 $\rightarrow (btr-or-nor(a, parity, tree)$
 $= \text{if } parity \text{ then } v\text{-zerop}(a)$
 $\text{else } v\text{-zerop}(a) \text{ endif})$

DEFINITION:

```

tv-zerop*(tree)
= cons(index('tv-zerop, tree-number(tree)),
      cons(indices('in, 0, tree-size(tree)),
            cons('out,
                  cons(if (tree-height(tree) mod 2) = 1
                        then cons(list('g0,
                                      'out-),
                                      index('t-or, tree-number(tree)),
                                      indices('in, 0, tree-size(tree))),
                        '((g1 (out) b-not (out-))))
                        else list(list('g0,
                                      'out),
                                      index('t-nor,
                                              tree-number(tree)),
                                      indices('in,
                                              0,
                                              tree-size(tree)))) endif,
            ' (nil))))))

```

THEOREM: tv-zerop*\$destructure
 $(car(tv-zerop*(tree)) = index('tv-zerop, tree-number(tree)))$
 $\wedge (cadr(tv-zerop*(tree)) = indices('in, 0, tree-size(tree)))$
 $\wedge (caddr(tv-zerop*(tree)) = 'out)$

```

 $\wedge$  (caddr (tv-zero* (tree))
    =  if (tree-height (tree) mod 2) = 1
        then cons (list ('g0,
                        ' (out-),
                        index ('t-or, tree-number (tree)),
                        indices ('in, 0, tree-size (tree))),
                    ' ((g1 (out) b-not (out-)))
        else list (list ('g0,
                        ' (out),
                        index ('t-nor, tree-number (tree)),
                        indices ('in, 0, tree-size (tree)))) endif)
 $\wedge$  (caddr (tv-zero* (tree)) = nil)

```

EVENT: Enable tv-zero^{*}; name this event 'tv-zero^{*}-off'.

EVENT: Enable tv-zero^{*}\$destructure; name this event 'tv-zero^{*}\$destructure-off'.

DEFINITION:

```

tv-zero& (netlist, tree)
= ((lookup-module (index ('tv-zero, tree-number (tree)), netlist)
   = tv-zero* (tree))
 $\wedge$  t-or-nor& (delete-module (index ('tv-zero, tree-number (tree)),
                                   netlist),
               tree,
               (tree-height (tree) mod 2)  $\neq$  1)
 $\wedge$  b-not& (delete-module (index ('tv-zero, tree-number (tree)),
                                   netlist)))

```

EVENT: Enable tv-zero&; name this event 'tv-zero&-off'.

DEFINITION:

```

tv-zero$netlist (tree)
= cons (tv-zero* (tree),
        t-or-nor$netlist (tree, (tree-height (tree) mod 2)  $\neq$  1)
         $\cup$  B-NOT$NETLIST)

```

DEFINITION:

```

f$tv-zero (a, tree)
=  if (tree-height (tree) mod 2) = 1 then f-not (tr-or-nor (a, f, tree))
    else tr-or-nor (a, t, tree) endif

```

EVENT: Enable f\$tv-zero; name this event 'f\$tv-zero-off'.

THEOREM: tv-zero\$value
 $(\text{tv-zero}(\text{netlist}, \text{tree}) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{properp}(a))$
 $\rightarrow (\text{dual-eval}(0, \text{index}(' \mathbf{tv-zero}, \text{tree-number}(\text{tree})), a, \text{state}, \text{netlist}))$
 $= \text{list}(\text{f\$tv-zero}(a, \text{tree}))$

EVENT: Enable tv-zero\$value; name this event 'tv-zero\$value-off'.

THEOREM: f\$tv-zero=v-zero
 $((\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{bvp}(a))$
 $\rightarrow (\text{f\$tv-zero}(a, \text{tree}) = \text{v-zero}(a))$

DEFINITION:
 $\text{f\$fast-zero}(v)$
 $= \text{f-nor3}(\text{tr-or-nor}(\text{firstn}((\text{length}(v) - 1) - 1, v),$
 $\quad \mathbf{f},$
 $\quad \text{make-tree}((\text{length}(v) - 1) - 1)),$
 $\quad \text{nth}((\text{length}(v) - 1) - 1, v),$
 $\quad \text{nth}(\text{length}(v) - 1, v))$

EVENT: Enable f\$fast-zero; name this event 'f\$fast-zero-off'.

THEOREM: f\$fast-zero=tr-or-nor
 $(\text{properp}(v)$
 $\wedge \text{if } \text{length}(v) < 3 \text{ then } \mathbf{f}$
 $\quad \mathbf{else } \mathbf{t} \text{ endif})$
 $\rightarrow (\text{f\$fast-zero}(v)$
 $= \text{tr-or-nor}(v,$
 $\quad \mathbf{t},$
 $\quad \text{cons}(\text{make-tree}((\text{length}(v) - 1) - 1), ' (0 \ . \ 0)))$

THEOREM: f\$fast-zero=v-zero
 $(\text{bvp}(v)$
 $\wedge \text{if } \text{length}(v) < 3 \text{ then } \mathbf{f}$
 $\quad \mathbf{else } \mathbf{t} \text{ endif})$
 $\rightarrow (\text{f\$fast-zero}(v) = \text{v-zero}(v))$

DEFINITION:
 $\text{fast-zero}^*(n)$
 $= \text{cons}(\text{index}(' \mathbf{fast-zero}, n),$
 $\quad \text{cons}(\text{indices}(' \mathbf{a}, 0, n),$
 $\quad \text{cons}(' (\mathbf{z}),$
 $\quad \text{cons}(\text{list}(\text{list}(' \mathbf{front},$
 $\quad \quad ' (\mathbf{zfront}),$
 $\quad \text{index}(' \mathbf{t-or},$

```

tree-number (make-tree ((n - 1) - 1)),
firstn ((n - 1) - 1, indices ('a, 0, n))),
list ('result,
      '(z),
      'b-nor3,
      list ('zfront,
            index ('a, (n - 1) - 1),
            index ('a, n - 1))))),
'(nil))))))

```

THEOREM: fast-zero*\$destructure

```

(car (fast-zero* (n)) = index ('fast-zero, n))
^ (cadr (fast-zero* (n)) = indices ('a, 0, n))
^ (caddr (fast-zero* (n)) = '(z))
^ (cadddr (fast-zero* (n))
    = list (list ('front,
                  '(zfront),
                  index ('t-or, tree-number (make-tree ((n - 1) - 1)),
                  firstn ((n - 1) - 1, indices ('a, 0, n))),
            list ('result,
                  '(z),
                  'b-nor3,
                  list ('zfront,
                        index ('a, (n - 1) - 1),
                        index ('a, n - 1))))))
^ (caddddr (fast-zero* (n)) = nil)

```

EVENT: Enable fast-zero*; name this event 'fast-zero*-off'.

EVENT: Enable fast-zero*\$destructure; name this event 'fast-zero*\$destructure-off'.

DEFINITION:

```

fast-zero& (netlist, n)
= ((lookup-module (index ('fast-zero, n), netlist) = fast-zero* (n))
   ^ t-or-nor& (delete-module (index ('fast-zero, n), netlist),
                    make-tree ((n - 1) - 1),
                    f)
   ^ b-nor3& (delete-module (index ('fast-zero, n), netlist)))

```

EVENT: Enable fast-zero&; name this event 'fast-zero&-off'.

DEFINITION:

```

fast-zero$netlist (n)
= cons (fast-zero* (n),
        t-or-nor$netlist (make-tree ((n - 1) - 1), f) ∪ B-NOR3$NETLIST)

```

THEOREM: check-fast-zero\$netlist
fast-zero& (fast-zero\$netlist (5), 5)

THEOREM: fast-zero\$value
(fast-zero& (netlist, n)
 ∧ properp (v)
 ∧ (length (v) = n)
 ∧ **if** n < 3 **then** f
 else t **endif**)
→ (dual-eval (0, index ('fast-zero, n), v, state, netlist)
 = list (f\$fast-zero (v)))

DEFINITION:
v-equal* (n)
= cons (index ('v-equal, n),
 cons (append (indices ('a, 0, n), indices ('b, 0, n)),
 cons ('equal),
 cons (list (list ('g0,
 indices ('x, 0, n),
 index ('v-xor, n),
 append (indices ('a, 0, n),
 indices ('b, 0, n))),
 list ('g1,
 'equal),
 index ('tv-zero, p,
 tree-number (make-tree (n))),
 indices ('x, 0, n))),
 ' (nil))))))

THEOREM: v-equal*\$destructure
(car (v-equal* (n)) = index ('v-equal, n))
 ∧ (cadr (v-equal* (n)) = append (indices ('a, 0, n), indices ('b, 0, n)))
 ∧ (caddr (v-equal* (n)) = 'equal)
 ∧ (cadddr (v-equal* (n))
 = list (list ('g0,
 indices ('x, 0, n),
 index ('v-xor, n),
 append (indices ('a, 0, n), indices ('b, 0, n))),
 list ('g1,
 'equal),
 index ('tv-zero, p, tree-number (make-tree (n))),

$$\wedge \quad (\text{caddddr}(\text{v-equal}^*(n)) = \mathbf{nil})$$

EVENT: Enable v-equal*; name this event ‘v-equal*-off’.

EVENT: Enable v-equal*\$destructure; name this event ‘v-equal*\$destructure-off’.

DEFINITION:

$$\begin{aligned} & \text{v-equal\&}(netlist, n) \\ = & ((\text{lookup-module}(\text{index}('v\text{-equal}, n), netlist) = \text{v-equal}^*(n)) \\ & \wedge \quad \text{v-xor\&}(\text{delete-module}(\text{index}('v\text{-equal}, n), netlist), n) \\ & \wedge \quad \text{tv-zero\&}(\text{delete-module}(\text{index}('v\text{-equal}, n), netlist), \\ & \quad \text{make-tree}(n))) \end{aligned}$$

EVENT: Enable v-equal\&; name this event ‘v-equal\&-off’.

DEFINITION:

$$\begin{aligned} & \text{v-equal}\$netlist(n) \\ = & \text{cons}(\text{v-equal}^*(n), \text{v-xor}\$netlist(n) \cup \text{tv-zero}\$netlist(\text{make-tree}(n))) \end{aligned}$$

DEFINITION:

$$\text{f}\$v\text{-equal}(a, b) = \text{f}\$tv\text{-zero}(\text{fv-xor}(a, b), \text{make-tree}(\text{length}(a)))$$

EVENT: Enable f\$v-equal; name this event ‘f\$v-equal-off’.

THEOREM: v-equal\$value

$$\begin{aligned} & (\text{v-equal\&}(netlist, n) \\ & \wedge \quad (n \neq 0) \\ & \wedge \quad \text{properp}(a) \\ & \wedge \quad \text{properp}(b) \\ & \wedge \quad (\text{length}(a) = n) \\ & \wedge \quad (\text{length}(b) = n)) \\ \rightarrow & (\text{dual-eval}(0, \text{index}('v\text{-equal}, n), \text{append}(a, b), state, netlist) \\ & \quad = \text{list}(\text{f}\$v\text{-equal}(a, b))) \end{aligned}$$

EVENT: Enable v-equal\$value; name this event ‘v-equal\$value-off’.

THEOREM: f\$v-equal=equal*

$$\begin{aligned} & ((\text{length}(a) \neq 0) \wedge \text{bvp}(a) \wedge \text{bvp}(b) \wedge (\text{length}(a) = \text{length}(b))) \\ \rightarrow & (\text{f}\$v\text{-equal}(a, b) = (a = b)) \end{aligned}$$

DEFINITION:

```
v-inc4(a0, a1, a2, a3)
= list(b-not(a0),
      b-xor(b-not(a1), b-not(a0)),
      b-equiv(b-not(a2), b-nor(b-not(a0), b-not(a1))),
      b-equiv(b-not(a3), b-nor3(b-not(a0), b-not(a1), b-not(a2)))))
```

DEFINITION:

```
f$v-inc4(a0, a1, a2, a3)
= list(f-not(a0),
      f-xor(f-not(a1), f-not(a0)),
      f-equiv(f-not(a2), f-nor(f-not(a0), f-not(a1))),
      f-equiv(f-not(a3), f-nor3(f-not(a0), f-not(a1), f-not(a2)))))
```

EVENT: Enable f\$v-inc4; name this event 'f\$v-inc4-off'.

DEFINITION:

```
v-INC4*
= '(v-inc4
   (a0 a1 a2 a3)
   (w-10 w-3 w-8 w-14)
   ((g-12 (w-12) b-not (a2))
    (g-11 (w-11) b-not (a1))
    (g-10 (w-10) b-not (a0))
    (g-13 (w-13) b-nor3 (w-10 w-11 w-12))
    (g-9 (w-9) b-not (a3))
    (g-14 (w-14) b-equiv (w-9 w-13))
    (g-7 (w-7) b-nor (w-10 w-11))
    (g-8 (w-8) b-equiv (w-12 w-7))
    (g-3 (w-3) b-xor (w-11 w-10)))
   nil)
```

DEFINITION:

```
v-inc4&(netlist)
= ((lookup-module('v-inc4, netlist) = v-INC4*)
   ^ b-nor3&(delete-module('v-inc4, netlist))
   ^ b-not&(delete-module('v-inc4, netlist))
   ^ b-nor&(delete-module('v-inc4, netlist))
   ^ b-equiv&(delete-module('v-inc4, netlist))
   ^ b-xor&(delete-module('v-inc4, netlist)))
```

EVENT: Enable v-inc4&; name this event 'v-inc4&-off'.

DEFINITION:

$$\begin{aligned} & \text{V-INC4}\$NETLIST \\ = & \text{cons}(\text{V-INC4}^*, \\ & \quad (((\text{B-NOR3}\$NETLIST \cup \text{B-NOT}\$NETLIST) \cup \text{B-NOR}\$NETLIST) \\ & \quad \cup \text{B-EQUV}\$NETLIST) \\ & \quad \cup \text{B-XOR}\$NETLIST) \end{aligned}$$

THEOREM: $\text{v-inc4}\$value$
 $\text{v-inc4}\&(\text{netlist})$
 $\rightarrow (\text{dual-eval}(0, \text{'v-inc4'}, \text{list}(a0, a1, a2, a3), \text{state}, \text{netlist})$
 $= \text{f}\$v\text{-inc4}(a0, a1, a2, a3))$

EVENT: Enable $\text{v-inc4}\$value$; name this event ' $\text{v-inc4}\$value\text{-off}$ '.

THEOREM: $\text{f}\$v\text{-inc4} = \text{v-inc4}$
 $(\text{boolp}(a0) \wedge \text{boolp}(a1) \wedge \text{boolp}(a2) \wedge \text{boolp}(a3))$
 $\rightarrow (\text{f}\$v\text{-inc4}(a0, a1, a2, a3) = \text{v-inc4}(a0, a1, a2, a3))$

DEFINITION:
 $\text{f}\$v\text{-inc4}\$v(a)$
 $= \text{list}(\text{f-not}(\text{car}(a)),$
 $\quad \text{f-xor}(\text{f-not}(\text{cadr}(a)), \text{f-not}(\text{car}(a))),$
 $\quad \text{f-equiv}(\text{f-not}(\text{caddr}(a)), \text{f-nor}(\text{f-not}(\text{car}(a)), \text{f-not}(\text{cadr}(a)))),$
 $\quad \text{f-equiv}(\text{f-not}(\text{caddr}(a)),$
 $\quad \quad \text{f-nor3}(\text{f-not}(\text{car}(a)), \text{f-not}(\text{cadr}(a)), \text{f-not}(\text{caddr}(a))))$

EVENT: Enable $\text{f}\$v\text{-inc4}\$v$; name this event ' $\text{f}\$v\text{-inc4}\$v\text{-off}$ '.

THEOREM: $\text{properp-length-f}\$v\text{-inc4}\$v$
 $\text{properp}(\text{f}\$v\text{-inc4}\$v(a)) \wedge (\text{length}(\text{f}\$v\text{-inc4}\$v(a)) = 4)$

THEOREM: $\text{v-inc4}\$value\text{-as-v-inc}$
 $(\text{v-inc4}\&(\text{netlist}) \wedge \text{properp}(a) \wedge (\text{length}(a) = 4))$
 $\rightarrow (\text{dual-eval}(0, \text{'v-inc4'}, a, \text{state}, \text{netlist}) = \text{f}\$v\text{-inc4}\$v(a))$

THEOREM: $\text{f}\$v\text{-inc4}\$v = \text{v-inc}$
 $(\text{bvp}(a) \wedge (\text{length}(a) = 4)) \rightarrow (\text{f}\$v\text{-inc4}\$v(a) = \text{v-inc}(a))$

DEFINITION:
 $\text{tv-dec-pass}(c, a, \text{tree})$
 $= \text{if } \text{tree} \simeq \text{nil} \text{ then } \text{list}(\text{b-buf}(\text{car}(a)), \text{b-equiv}(\text{car}(a), c))$
 $\quad \text{else } \text{cons}(\text{b-or}(\text{car}(\text{tv-dec-pass}(c, \text{tfirstn}(a, \text{tree})), \text{car}(\text{tree}))),$
 $\quad \quad \text{car}(\text{tv-dec-pass}(\text{b-or}(c,$
 $\quad \quad \quad \text{car}(\text{tv-dec-pass}(c,$
 $\quad \quad \quad \text{tfirstn}(a,$


```

                                tree),
                                car (tree))),
                                trestn (a, tree),
                                cdr (tree))),
                                append (cdr (tv-dec-pass (c, tfirstn (a, tree), car (tree))),
                                cdr (tv-dec-pass (b-or (c,
                                car (tv-dec-pass (c,
                                tfirstn (a,
                                tree),
                                car (tree)))))
                                trestn (a, tree),
                                cdr (tree))))) endif

```

EVENT: Enable tv-dec-pass; name this event ‘tv-dec-pass-off’.

THEOREM: cdr-tv-dec-pass-length
 $\text{length}(\text{cdr}(\text{tv-dec-pass}(c, a, \text{tree}))) = \text{tree-size}(\text{tree})$

THEOREM: tv-dec-pass-length
 $\text{length}(\text{tv-dec-pass}(c, a, \text{tree})) = (1 + \text{tree-size}(\text{tree}))$

THEOREM: bvp-cdr-tv-dec-pass
 $\text{bvp}(\text{cdr}(\text{tv-dec-pass}(c, a, \text{tree})))$

THEOREM: bvp-tv-dec-pass
 $\text{bvp}(\text{tv-dec-pass}(c, a, \text{tree}))$

THEOREM: bvp-length-tv-dec-pass
 $\text{bvp-length}(\text{tv-dec-pass}(c, a, \text{tree}), n)$
 $= \text{if } (1 + \text{tree-size}(\text{tree})) < n \text{ then f}$
 else t endif

THEOREM: tv-dec-pass-crock-1
 $((\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{bvp}(a))$
 $\rightarrow (\text{tv-adder}(c, \text{v-not}(\text{nat-to-v}(0, \text{length}(a))), a, \text{tree}))$
 $= \text{cons}(\mathbf{t}, \text{tv-dec-pass}(c, a, \text{tree}))$

THEOREM: tv-dec-pass-crock-2
 $((\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{bvp}(a))$
 $\rightarrow (\text{cdr}(\text{tv-dec-pass}(c, a, \text{tree})))$
 $= \text{cddr}(\text{tv-adder}(c, \text{v-not}(\text{nat-to-v}(0, \text{length}(a))), a, \text{tree}))$

THEOREM: tv-dec-pass-works
 $((\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{bvp}(a))$
 $\rightarrow (\text{cdr}(\text{tv-dec-pass}(c, a, \text{tree})))$
 $= \text{if } c \text{ then v-buf}(a)$
 $\text{else v-dec}(a) \text{ endif}$

DEFINITION:

```

tv-dec-pass-ng (c, a, tree, make-g)
=  if tree  $\simeq$  nil
    then if make-g then list (b-buf (car (a)), b-equiv (car (a), c))
        else list (b-equiv (car (a), c)) endif
    elseif make-g
    then cons (b-or (car (tv-dec-pass-ng (c, tfirstn (a, tree), car (tree), t)),
        car (tv-dec-pass-ng (b-or (c,
            car (tv-dec-pass-ng (c,
                tfirstn (a,
                    tree),
                car (tree),
                t))),
            trestn (a, tree),
            cdr (tree),
            make-g))),
        append (cdr (tv-dec-pass-ng (c, tfirstn (a, tree), car (tree), t)),
            cdr (tv-dec-pass-ng (b-or (c,
                car (tv-dec-pass-ng (c,
                    tfirstn (a,
                        tree),
                    car (tree),
                    t))),
                    trestn (a, tree),
                    cdr (tree),
                    make-g))))))
    else append (cdr (tv-dec-pass-ng (c, tfirstn (a, tree), car (tree), t)),
        tv-dec-pass-ng (b-or (c,
            car (tv-dec-pass-ng (c,
                tfirstn (a,
                    tree),
                car (tree),
                t))),
            trestn (a, tree),
            cdr (tree),
            make-g)) endif

```

EVENT: Enable tv-dec-pass-ng; name this event 'tv-dec-pass-ng-off'.

THEOREM: tv-dec-pass-ng-length

```

length (tv-dec-pass-ng (c, a, tree, make-g))
=  if make-g then 1 + tree-size (tree)
    else tree-size (tree) endif

```

THEOREM: tv-dec-pass-ng-length-1

$make-g \rightarrow (\text{length}(\text{cdr}(\text{tv-dec-pass-ng}(c, a, tree, make-g))) = \text{tree-size}(tree))$

THEOREM: bvp-tv-dec-pass-ng

$\text{bvp}(\text{tv-dec-pass-ng}(c, a, tree, make-g))$

THEOREM: bvp-cdr-tv-dec-pass-ng

$make-g \rightarrow \text{bvp}(\text{cdr}(\text{tv-dec-pass-ng}(c, a, tree, make-g)))$

THEOREM: bvp-length-tv-dec-pass-ng

$\text{bvp-length}(\text{tv-dec-pass-ng}(c, a, tree, make-g), n)$

$=$ **if** $make-g$
 then if $(1 + \text{tree-size}(tree)) < n$ **then f**
 else t endif
 else $n < (1 + \text{tree-size}(tree))$ **endif**

THEOREM: tv-dec-pass-ng-is-cdr-tv-dec-pass

$\text{tv-dec-pass-ng}(c, a, tree, make-g)$

$=$ **if** $make-g$ **then** $\text{tv-dec-pass}(c, a, tree)$
 else $\text{cdr}(\text{tv-dec-pass}(c, a, tree))$ **endif**

THEOREM: boolp-car-tv-dec-pass-ng

$\text{boolp}(\text{car}(\text{tv-dec-pass-ng}(c, a, tree, make-g)))$

THEOREM: tv-dec-pass-ng-works-1

$(\text{bvp}(a) \wedge (\text{length}(a) = \text{tree-size}(tree)) \wedge make-g)$
 $\rightarrow (\text{cdr}(\text{tv-dec-pass-ng}(c, a, tree, make-g))$
 $=$ **if** c **then** $\text{v-buf}(a)$
 else $\text{v-dec}(a)$ **endif**

THEOREM: tv-dec-pass-ng-works-2

$(\text{bvp}(a) \wedge (\text{length}(a) = \text{tree-size}(tree)) \wedge (\neg make-g))$
 $\rightarrow (\text{tv-dec-pass-ng}(c, a, tree, make-g)$
 $=$ **if** c **then** $\text{v-buf}(a)$
 else $\text{v-dec}(a)$ **endif**

DEFINITION:

$\text{f\$tv-dec-pass-ng}(c, a, tree, make-g)$

$=$ **if** $tree \simeq \text{nil}$
 then if $make-g$ **then** $\text{list}(\text{car}(a), \text{f-equiv}(\text{car}(a), c))$
 else $\text{list}(\text{f-equiv}(\text{car}(a), c))$ **endif**
 elseif $make-g$
 then $\text{cons}(\text{f-or}(\text{car}(\text{f\$tv-dec-pass-ng}(c, \text{tfirstn}(a, tree), \text{car}(tree), \mathbf{t})),$
 $\text{car}(\text{f\$tv-dec-pass-ng}(\text{f-or}(c,$
 $\text{car}(\text{f\$tv-dec-pass-ng}(c,$

```

                                tfirstn (a,
                                    tree),
                                car (tree),
                                t))),
                                trestn (a, tree),
                                cdr (tree),
                                make-g))),
append (cdr (f$tv-dec-pass-ng (c, tfirstn (a, tree), car (tree), t)),
        cdr (f$tv-dec-pass-ng (f-or (c,
                                    car (f$tv-dec-pass-ng (c,
                                                            tfirstn (a,
                                                                tree),
                                                                car (tree),
                                                                t))),
                                    trestn (a, tree),
                                    cdr (tree),
                                    make-g))))))
else append (cdr (f$tv-dec-pass-ng (c, tfirstn (a, tree), car (tree), t)),
              f$tv-dec-pass-ng (f-or (c,
                                      car (f$tv-dec-pass-ng (c,
                                                              tfirstn (a,
                                                                  tree),
                                                                  car (tree),
                                                                  t))),
                                      trestn (a, tree),
                                      cdr (tree),
                                      make-g)) endif

```

EVENT: Enable f\$tv-dec-pass-ng; name this event ‘f\$tv-dec-pass-ng-off’.

THEOREM: f\$tv-dec-pass-ng-length
length (f\$tv-dec-pass-ng (c, a, tree, make-g))
= **if** make-g **then** 1 + tree-size (tree)
else tree-size (tree) **endif**

THEOREM: f\$tv-dec-pass-ng-length-1
make-g → (length (cdr (f\$tv-dec-pass-ng (c, a, tree, make-g)))
= tree-size (tree))

THEOREM: properp-f\$tv-dec-pass-ng
properp (f\$tv-dec-pass-ng (c, a, tree, make-g))

THEOREM: properp-cdr-f\$tv-dec-pass-ng
make-g → properp (cdr (f\$tv-dec-pass-ng (c, a, tree, make-g)))

THEOREM: f\$tv-dec-pass-ng=tv-dec-pass-ng\$super-crock
 (f\$tv-dec-pass-ng(*c*, *a*, *tree*, *make-g*) = tv-dec-pass-ng(*c*, *a*, *tree*, *make-g*))
 → boolp(car(f\$tv-dec-pass-ng(*c*, *a*, *tree*, *make-g*)))

EVENT: Enable f\$tv-dec-pass-ng=tv-dec-pass-ng\$super-crock; name this event
 ‘f\$tv-dec-pass-ng=tv-dec-pass-ng\$super-crock-off’.

THEOREM: f\$tv-dec-pass-ng=tv-dec-pass-ng
 (boolp(*c*) ∧ bvp(*a*) ∧ (length(*a*) = tree-size(*tree*)))
 → (f\$tv-dec-pass-ng(*c*, *a*, *tree*, *make-g*)
 = tv-dec-pass-ng(*c*, *a*, *tree*, *make-g*))

DEFINITION:

DEC-PASS-CELL*

= '(dec-pass-cell
 (*c* *a*)
 (*g* *z*)
 ((*g*0 (*g*) id(*a*)) (*g*1 (*z*) b-equiv(*a* *c*)))
 nil)

DEFINITION:

dec-pass-cell&(*netlist*)

= ((lookup-module('dec-pass-cell, *netlist*) = DEC-PASS-CELL*)
 ∧ id&(delete-module('dec-pass-cell, *netlist*))
 ∧ b-equiv&(delete-module('dec-pass-cell, *netlist*)))

EVENT: Enable dec-pass-cell&; name this event ‘dec-pass-cell&-off’.

DEFINITION:

DEC-PASS-CELL\$NETLIST

= cons(DEC-PASS-CELL*, ID\$NETLIST ∪ B-EQUV\$NETLIST)

THEOREM: dec-pass-cell\$value

dec-pass-cell&(*netlist*)
 → (dual-eval(0, 'dec-pass-cell, list(*c*, *a*), *state*, *netlist*)
 = list(*a*, f-equiv(*a*, *c*)))

DEFINITION:

tv-dec-pass-name(*tree*, *make-g*)

= **if** *make-g* **then** index('tv-dec-pass-g, tree-number(*tree*))
 else index('tv-dec-pass-ng, tree-number(*tree*)) **endif**

DEFINITION:

tv-dec-pass-ng-body(*tree*, *make-g*)

```

=  if tree  $\simeq$  nil
    then list (list ('leaf,
                    list ('g, index ('z, 0)),
                    'dec-pass-cell,
                    list ('c, index ('a, 0))))
    elseif make-g
    then cons (list ('left,
                    cons ('gl, tfirstn (indices ('z, 0, tree-size (tree)), tree)),
                    index ('tv-dec-pass-g, tree-number (car (tree))),
                    cons ('c, tfirstn (indices ('a, 0, tree-size (tree)), tree))),
                    cons ('(carry (cx) b-or (c gl)),
                          cons (list ('right,
                                      cons ('gr,
                                            trestn (indices ('z, 0, tree-size (tree)),
                                                            tree)),
                                      index ('tv-dec-pass-g,
                                            tree-number (cdr (tree))),
                                      cons ('cx,
                                            trestn (indices ('a, 0, tree-size (tree)),
                                                            tree))),
                                      '(generate (g) b-or (gl gr))))))
    else list (list ('left,
                    cons ('gl,
                          tfirstn (indices ('z, 0, tree-size (tree)), tree)),
                    index ('tv-dec-pass-g, tree-number (car (tree))),
                    cons ('c,
                          tfirstn (indices ('a, 0, tree-size (tree)), tree))),
                    '(carry (cx) b-or (c gl)),
                    list ('right,
                          trestn (indices ('z, 0, tree-size (tree)), tree),
                          index ('tv-dec-pass-ng,
                                tree-number (cdr (tree))),
                          cons ('cx,
                                trestn (indices ('a, 0, tree-size (tree)), tree)))) endif

```

EVENT: Enable tv-dec-pass-ng-body; name this event 'tv-dec-pass-ng-body-off'.

DEFINITION:

```

tv-dec-pass-ng* (tree, make-g)
=  cons (tv-dec-pass-name (tree, make-g),
        cons (cons ('c, indices ('a, 0, tree-size (tree))),
              cons (if make-g
                    then cons ('g, indices ('z, 0, tree-size (tree)))
                    else indices ('z, 0, tree-size (tree)) endif,

```

cons (tv-dec-pass-ng-body (*tree*, *make-g*), ' (nil)))))

THEOREM: tv-dec-pass-ng*\$destructure
(car (tv-dec-pass-ng* (*tree*, *make-g*)) = tv-dec-pass-name (*tree*, *make-g*))
 $\wedge$ (cadr (tv-dec-pass-ng* (*tree*, *make-g*))
= cons ('c, indices ('a, 0, tree-size (*tree*))))
 $\wedge$ (caddr (tv-dec-pass-ng* (*tree*, *make-g*))
= **if** *make-g* **then** cons ('g, indices ('z, 0, tree-size (*tree*)))
else indices ('z, 0, tree-size (*tree*)) **endif**)
 $\wedge$ (caddr (tv-dec-pass-ng* (*tree*, *make-g*))
= tv-dec-pass-ng-body (*tree*, *make-g*))
 $\wedge$ (caddr (tv-dec-pass-ng* (*tree*, *make-g*)) = **nil**)

EVENT: Enable tv-dec-pass-ng*; name this event 'tv-dec-pass-ng*-off'.

EVENT: Enable tv-dec-pass-ng*\$destructure; name this event 'tv-dec-pass-ng*\$destructure-off'.

DEFINITION:

tv-dec-pass-ng& (*netlist*, *tree*, *make-g*)
= **if** *tree* $\simeq$ **nil**
then (lookup-module (tv-dec-pass-name (*tree*, *make-g*), *netlist*)
= tv-dec-pass-ng* (*tree*, *make-g*))
 $\wedge$ dec-pass-cell& (delete-module (tv-dec-pass-name (*tree*, *make-g*),
netlist))
else (lookup-module (tv-dec-pass-name (*tree*, *make-g*), *netlist*)
= tv-dec-pass-ng* (*tree*, *make-g*))
 $\wedge$ tv-dec-pass-ng& (delete-module (tv-dec-pass-name (*tree*,
make-g),
netlist),
car (*tree*),
t)
 $\wedge$ tv-dec-pass-ng& (delete-module (tv-dec-pass-name (*tree*,
make-g),
netlist),
cdr (*tree*),
make-g)
 $\wedge$ b-or& (delete-module (tv-dec-pass-name (*tree*, *make-g*),
netlist)) **endif**

EVENT: Enable tv-dec-pass-ng&; name this event 'tv-dec-pass-ng&-off'.

DEFINITION:

```

tv-dec-pass-ng-induction(tree, c, a, make-g, name, state, netlist)
=  if tree  $\simeq$  nil then t
    else tv-dec-pass-ng-induction(car(tree),
                                   c,
                                   tfirstn(a, tree),
                                   t,
                                   index('tv-dec-pass-g,
                                         tree-number(car(tree))),
                                   0,
                                   delete-module(tv-dec-pass-name(tree,
                                                                    make-g),
                                                                    netlist))
    ∧ tv-dec-pass-ng-induction(cdr(tree),
                               f-or(c,
                                     car(f$tv-dec-pass-ng(c,
                                                            tfirstn(a,
                                                                tree),
                                                                car(tree),
                                                                t))),
                               trestn(a, tree),
                               t,
                               index('tv-dec-pass-g,
                                     tree-number(cdr(tree))),
                               0,
                               delete-module(tv-dec-pass-name(tree,
                                                                    make-g),
                                                                    netlist))
    ∧ tv-dec-pass-ng-induction(cdr(tree),
                               f-or(c,
                                     car(f$tv-dec-pass-ng(c,
                                                            tfirstn(a,
                                                                tree),
                                                                car(tree),
                                                                t))),
                               trestn(a, tree),
                               f,
                               index('tv-dec-pass-ng,
                                     tree-number(cdr(tree))),
                               0,
                               delete-module(tv-dec-pass-name(tree,
                                                                    make-g),
                                                                    netlist)) endif

```

THEOREM: tv-dec-pass-ng-lemma-crock

$$\begin{aligned}
& (\text{tv-dec-pass-ng\&} (\text{delete-module} (\text{index} (' \text{tv-dec-pass-g}, \text{tree-number} (tree)), \\
& \quad \text{netlist}), \\
& \quad \text{car} (tree), \\
& \quad \text{make-g}) \\
\rightarrow & (\text{collect-value} (\text{indices} (' \mathbf{z}, 0, n), \text{bindings}) \\
& = \text{append} (\text{collect-value} (\text{firstn} (\text{tree-size} (\text{car} (tree)), \\
& \quad \text{indices} (' \mathbf{z}, 0, n)), \\
& \quad \text{bindings}), \\
& \text{collect-value} (\text{restn} (\text{tree-size} (\text{car} (tree)), \\
& \quad \text{indices} (' \mathbf{z}, 0, n)), \\
& \quad \text{bindings}))) \\
\wedge & (\text{tv-dec-pass-ng\&} (\text{delete-module} (\text{index} (' \text{tv-dec-pass-ng}, \\
& \quad \text{tree-number} (tree)), \\
& \quad \text{netlist}), \\
& \quad \text{car} (tree), \\
& \quad \text{make-g}) \\
\rightarrow & (\text{collect-value} (\text{indices} (' \mathbf{z}, 0, n), \text{bindings}) \\
& = \text{append} (\text{collect-value} (\text{firstn} (\text{tree-size} (\text{car} (tree)), \\
& \quad \text{indices} (' \mathbf{z}, 0, n)), \\
& \quad \text{bindings}), \\
& \text{collect-value} (\text{restn} (\text{tree-size} (\text{car} (tree)), \\
& \quad \text{indices} (' \mathbf{z}, 0, n)), \\
& \quad \text{bindings})))
\end{aligned}$$

EVENT: Enable tv-dec-pass-ng-lemma-crock; name this event ‘tv-dec-pass-ng-lemma-crock-off’.

THEOREM: tv-dec-pass-ng\$value
 $(\text{tv-dec-pass-ng\&} (\text{netlist}, \text{tree}, \text{make-g})$
 $\wedge (\text{length} (a) = \text{tree-size} (tree))$
 $\wedge \text{properp} (a)$
 $\wedge \text{boolp} (\text{make-g})$
 $\wedge (\text{name} = \text{tv-dec-pass-name} (tree, \text{make-g}))$
 $\rightarrow (\text{dual-eval} (0, \text{name}, \text{cons} (c, a), \text{state}, \text{netlist})$
 $= \text{f\$tv-dec-pass-ng} (c, a, \text{tree}, \text{make-g}))$

EVENT: Enable tv-dec-pass-ng\$value; name this event ‘tv-dec-pass-ng\$value-off’.

DEFINITION:
 $\text{tv-dec-pass-ng\$netlist} (tree, \text{make-g})$
 $= \text{if } tree \simeq \text{nil} \text{ then list} (\text{tv-dec-pass-ng}^* (tree, \text{make-g}), \text{DEC-PASS-CELL}^*)$
 $\text{else cons} (\text{tv-dec-pass-ng}^* (tree, \text{make-g}),$

```

tv-dec-pass-ng$netlist (car (tree), t)
∪ tv-dec-pass-ng$netlist (cdr (tree), make-g)) endif

```

THEOREM: check-tv-dec-pass-ng\$netlist

$$\text{boolp}(make-g)$$
$$\rightarrow \text{tv-dec-pass-ng} \& (\text{tv-dec-pass-ng} \$ \text{netlist} \text{ ' (0 0 . 0) , make-g) ,}$$

DEFINITION:

$$\text{dec-pass}^*(n)$$

```
= cons (index ('dec-pass, n),
      cons (cons ('c, indices ('a, 0, n)),
            cons (indices ('z, 0, n),
                  cons (list ('m0 (cn) b-not (c)),
                        list ('m1,
                              indices ('z, 0, n),
                              index ('tv-dec-pass-ng,
                                      tree-number (make-tree (n))),
                              cons ('cn, indices ('a, 0, n))))),
            ' (nil))))))
```

THEOREM: dec-pass*\$destructure

$$(\text{car}(\text{dec-pass}^*(n)) = \text{index}(' \text{dec-pass}, n))$$
$$\wedge \quad (\text{cadr}(\text{dec-pass}^*(n)) = \text{cons}('c, \text{indices}('a, 0, n)))$$
$$\wedge \quad (\text{caddr}(\text{dec-pass}^*(n)) = \text{indices}('z, 0, n))$$
$$\wedge \quad (\text{caddr}(\text{dec-pass}^*(n)))$$

```
= list ('m0 (cn) b-not (c)),
      list ('m1,
            indices ('z, 0, n),
            index ('tv-dec-pass-ng,
                  tree-number (make-tree (n))),
            cons ('cn, indices ('a, 0, n))))
```

$$\wedge \quad (\text{caddddr}(\text{dec-pass}^*(n)) = \mathbf{nil})$$

EVENT: Enable dec-pass*; name this event 'dec-pass*-off'.

EVENT: Enable dec-pass*\$destructure; name this event ‘dec-pass*\$destructure-off’.

DEFINITION:

$$\text{dec-pass\&}(netlist, n)$$
$$= ((\text{lookup-module}(\text{index}(' \text{dec-pass}, n), \text{netlist}) = \text{dec-pass}^*(n)) \wedge \text{b-not\&}(\text{delete-module}(\text{index}(' \text{dec-pass}, n), \text{netlist})))$$

$$\wedge \text{ tv-dec-pass-ng\& } (\text{delete-module}(\text{index}(' \text{dec-pass}, n), \text{netlist}), \\ \text{make-tree}(n), \\ \mathbf{f}))$$

EVENT: Enable dec-pass&; name this event 'dec-pass&-off'.

DEFINITION:

$$\text{f\$dec-pass}(c, a) = \text{f\$tv-dec-pass-ng}(\text{f-not}(c), a, \text{make-tree}(\text{length}(a)), \mathbf{f})$$

EVENT: Enable f\\$dec-pass; name this event 'f\\$dec-pass-off'.

THEOREM: properp-length-f\\$dec-pass

$$\text{properp}(\text{f\$dec-pass}(c, v))$$

$$\wedge ((\text{length}(v) \neq 0) \rightarrow (\text{length}(\text{f\$dec-pass}(c, v)) = \text{length}(v)))$$

THEOREM: f\\$dec-pass=dec-or-pass

$$(\text{bvp}(v) \wedge \text{boolp}(dec) \wedge (\text{length}(v) \neq 0))$$

$$\rightarrow (\text{f\$dec-pass}(dec, v) = \text{if}^*(dec, \text{v-dec}(v), v))$$

THEOREM: dec-pass\$value

$$(\text{dec-pass\&}(netlist, n) \wedge (n \neq 0) \wedge (\text{length}(a) = n) \wedge \text{properp}(a))$$

$$\rightarrow (\text{dual-eval}(0, \text{index}(' \text{dec-pass}, n), \text{cons}(c, a), state, netlist) \\ = \text{f\$dec-pass}(c, a))$$

EVENT: Enable dec-pass\$value; name this event 'dec-pass\$value-off'.

DEFINITION:

$$\text{dec-pass\$netlist}(n)$$

$$= \text{cons}(\text{dec-pass}^*(n),$$

$$\text{B-NOT\$NETLIST} \cup \text{tv-dec-pass-ng\$netlist}(\text{make-tree}(n), \mathbf{f}))$$

THEOREM: check-dec-pass\\$netlist

$$\text{dec-pass\&}(\text{dec-pass\$netlist}(7), 7)$$

DEFINITION:

$$\text{reg-body}(m, n, ti, te)$$

$$= \text{if } n \simeq 0 \text{ then nil}$$

$$\text{else cons}(\text{list}(\text{index}(' \mathbf{g}, m),$$

$$\text{list}(\text{index}(' \mathbf{q}, m), \text{index}(' \mathbf{qb}, m)),$$

$$' \mathbf{fd1s},$$

$$\text{list}(\text{index}(' \mathbf{d}, m), ' \mathbf{clk}, ti, te)),$$

$$\text{reg-body}(1 + m, n - 1, \text{index}(' \mathbf{q}, m), te)) \text{ endif}$$

EVENT: Enable reg-body; name this event 'reg-body-off'.

DEFINITION:

```

reg*(n)
= list(index('reg, n),
      cons('clk, cons('te, cons('ti, indices('d, 0, n)))),
      indices('q, 0, n),
      cons(cons('te-buffer,
                cons('te-buf,
                    cons(if n < 8 then 'b-buf
                        else 'b-buf-pwr endif,
                        '((te))))),
            cons('ti-del (ti-buf) del4 (ti)),
            reg-body(0, n, 'ti-buf, 'te-buf))),
      indices('g, 0, n))

```

THEOREM: reg*\$destructure

```

(car(reg*(n)) = index('reg, n))
^ (cadr(reg*(n)) = cons('clk, cons('te, cons('ti, indices('d, 0, n))))
^ (caddr(reg*(n)) = indices('q, 0, n))
^ (caddr(reg*(n))
  = cons(cons('te-buffer,
              cons('te-buf,
                  cons(if n < 8 then 'b-buf
                      else 'b-buf-pwr endif,
                      '((te))))),
          cons('ti-del (ti-buf) del4 (ti)),
          reg-body(0, n, 'ti-buf, 'te-buf)))
^ (caddr(reg*(n)) = indices('g, 0, n))

```

EVENT: Enable reg*; name this event 'reg*-off'.

EVENT: Enable reg*\$destructure; name this event 'reg*\$destructure-off'.

DEFINITION:

```

reg&(netlist, n)
= ((lookup-module(index('reg, n), netlist) = reg*(n))
  ^ if n < 8
    then b-buf&(delete-module(index('reg, n), netlist))
    else b-buf-pwr&(delete-module(index('reg, n), netlist)) endif
  ^ del4&(delete-module(index('reg, n), netlist))
  ^ fd1s&(delete-module(index('reg, n), netlist)))

```

EVENT: Enable reg&; name this event 'reg&-off'.

DEFINITION:

```

reg$netlist(n)
=  cons(reg*(n),
      if n < 8 then B-BUF$NETLIST
      else B-BUF-PWR$NETLIST endif
      ∪ (DEL4$NETLIST ∪ FD1S$NETLIST))

```

THEOREM: reg-body\$unbound-in-body
 unbound-in-body('clk, reg-body(*m*, *n*, *ti*, *te*))
 ∧ unbound-in-body('ti, reg-body(*m*, *n*, *ti*, *te*))
 ∧ unbound-in-body('ti-del, reg-body(*m*, *n*, *ti*, *te*))
 ∧ unbound-in-body('te, reg-body(*m*, *n*, *ti*, *te*))
 ∧ unbound-in-body('te-buf, reg-body(*m*, *n*, *ti*, *te*))
 ∧ unbound-in-body(index('d, *l*), reg-body(*m*, *n*, *ti*, *te*))
 ∧ ((*l* < *m*) → unbound-in-body(index('q, *l*), reg-body(*m*, *n*, *ti*, *te*)))

EVENT: Enable reg-body\$unbound-in-body; name this event 'reg-body\$unbound-in-body-off'.

THEOREM: reg-body\$all-unbound-in-body
 all-unbound-in-body(indices('d, *x*, *y*), reg-body(*m*, *n*, *ti*, *te*))

EVENT: Enable reg-body\$all-unbound-in-body; name this event 'reg-body\$all-unbound-in-body-off'.

DEFINITION:
 reg-body-induction(*m*, *n*, *ti*, *te*, *bindings*, *state-bindings*, *netlist*)
 = if *n* ≃ 0 then **t**
 else reg-body-induction(1 + *m*,
 n − 1,
 index('q, *m*),
 te,
 dual-eval-body-bindings(1,
 reg-body(*m*,
 n,
 ti,
 te),
 bindings,
 state-bindings,
 netlist),
 state-bindings,
 netlist) endif

THEOREM: reg-body\$value
 fd1s&(*netlist*)

$$\begin{aligned}
& \rightarrow (\text{collect-value}(\text{indices}('q, m, n), \\
& \quad \text{dual-eval}(1, \\
& \quad \quad \text{reg-body}(m, n, ti, te), \\
& \quad \quad \text{bindings}, \\
& \quad \quad \text{state-bindings}, \\
& \quad \quad \text{netlist})) \\
& = \text{v-threefix}(\text{collect-value}(\text{indices}('g, m, n), \text{state-bindings}))
\end{aligned}$$

THEOREM: reg\$value

$$\begin{aligned}
& (\text{reg}\&(\text{netlist}, n) \wedge (\text{length}(\text{state}) = n) \wedge \text{properp}(\text{state})) \\
& \rightarrow (\text{dual-eval}(0, \text{index}('reg, n), \text{inputs}, \text{state}, \text{netlist}) \\
& \quad = \text{v-threefix}(\text{state}))
\end{aligned}$$

EVENT: Enable reg\$value; name this event 'reg\$value-off'.

THEOREM: reg\$body\$state

$$\begin{aligned}
& (\text{fd1s}\&(\text{netlist}) \wedge (\neg \text{value}(te, \text{bindings}))) \\
& \rightarrow (\text{collect-value}(\text{indices}('g, m, n), \\
& \quad \text{dual-eval}(3, \\
& \quad \quad \text{reg-body}(m, n, ti, te), \\
& \quad \quad \text{bindings}, \\
& \quad \quad \text{state-bindings}, \\
& \quad \quad \text{netlist})) \\
& = \text{v-threefix}(\text{collect-value}(\text{indices}('d, m, n), \text{bindings}))
\end{aligned}$$

THEOREM: reg\$state

$$\begin{aligned}
& (\text{reg}\&(\text{netlist}, n) \wedge (\text{length}(d) = n) \wedge \text{properp}(d) \wedge (\neg te)) \\
& \rightarrow (\text{dual-eval}(2, \\
& \quad \text{index}('reg, n), \\
& \quad \text{cons}(\text{clk}, \text{cons}(te, \text{cons}(ti, d))), \\
& \quad \text{state}, \\
& \quad \text{netlist}) \\
& = \text{v-threefix}(d))
\end{aligned}$$

EVENT: Enable reg\$state; name this event 'reg\$state-off'.

DEFINITION:

we-reg-body(m, n, ti)

$$\begin{aligned}
& = \text{if } n \simeq 0 \text{ then nil} \\
& \quad \text{else cons}(\text{list}(\text{index}('g, m), \\
& \quad \quad \text{list}(\text{index}('q, m), \text{index}('qb, m)), \\
& \quad \quad 'fd1slp, \\
& \quad \quad \text{cons}(\text{index}('d, m), \\
& \quad \quad \quad \text{cons}('clk,
\end{aligned}$$

cons('we-buf, cons(ti, '(te-buf))))),
 we-reg-body(1 + m, n - 1, index('q, m)) **endif**

EVENT: Enable we-reg-body; name this event 'we-reg-body-off'.

DEFINITION:

we-reg*(n)
 = list(index('we-reg, n),
 cons('clk, cons('we, cons('te, cons('ti, indices('d, 0, n))))),
 indices('q, 0, n),
 cons(cons('we-buffer,
 cons('we-buf),
 cons(if n < 8 then 'b-buf
 else 'b-buf-pwr endif,
 '((we))))),
 cons(cons('te-buffer,
 cons('te-buf),
 cons(if n < 8 then 'b-buf
 else 'b-buf-pwr endif,
 '((te))))),
 cons('ti-del (ti-buf) del4 (ti)),
 we-reg-body(0, n, 'ti-buf))),
 indices('g, 0, n))

THEOREM: we-reg*\$destructure

(car (we-reg*(n)) = index('we-reg, n))
 ∧ (cadr (we-reg*(n))
 = cons('clk, cons('we, cons('te, cons('ti, indices('d, 0, n))))))
 ∧ (caddr (we-reg*(n)) = indices('q, 0, n))
 ∧ (cadddr (we-reg*(n))
 = cons(cons('we-buffer,
 cons('we-buf),
 cons(if n < 8 then 'b-buf
 else 'b-buf-pwr endif,
 '((we))))),
 cons(cons('te-buffer,
 cons('te-buf),
 cons(if n < 8 then 'b-buf
 else 'b-buf-pwr endif,
 '((te))))),
 cons('ti-del (ti-buf) del4 (ti)),
 we-reg-body(0, n, 'ti-buf))))
 ∧ (caddddr (we-reg*(n)) = indices('g, 0, n))

EVENT: Enable we-reg*; name this event 'we-reg*-off'.

EVENT: Enable we-reg*\$destructure; name this event ‘we-reg*\$destructure-off’.

DEFINITION:

```

we-reg& (netlist, n)
= ((lookup-module (index ('we-reg, n), netlist) = we-reg* (n))
  ∧  if n < 8
      then b-buf& (delete-module (index ('we-reg, n), netlist))
      else b-buf-pwr& (delete-module (index ('we-reg, n),
                                             netlist)) endif
  ∧  del4& (delete-module (index ('we-reg, n), netlist))
  ∧  fd1slp& (delete-module (index ('we-reg, n), netlist)))

```

EVENT: Enable we-reg&; name this event 'we-reg&-off'.

DEFINITION:

$$\begin{aligned} & \text{we-reg}\$netlist(n) \\ = & \text{cons}(\text{we-reg}^*(n), \\ & \quad \text{if } n < 8 \text{ then B-BUF}\$NETLIST \\ & \quad \text{else B-BUF-PWR}\$NETLIST \text{ endif} \\ & \quad \cup (\text{DEL4}\$NETLIST \cup \text{FD1SLP}\$NETLIST)) \end{aligned}$$
$$\begin{aligned} & \text{THEOREM: } \text{we-reg-body} \text{unbound-in-body} \\ & \text{unbound-in-body} (' \text{clk}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{we}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{we-buf}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{ti}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{ti-buf}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{te}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (' \text{te-buf}, \text{we-reg-body} (m, n, ti)) \\ & \wedge \text{unbound-in-body} (\text{index} (' \text{d}, l), \text{we-reg-body} (m, n, ti)) \\ & \wedge ((l < m) \rightarrow \text{unbound-in-body} (\text{index} (' \text{q}, l), \text{we-reg-body} (m, n, ti))) \end{aligned}$$

EVENT: Enable we-reg-body\$unbound-in-body; name this event ‘we-reg-body\$unbound-in-body-off’.

THEOREM: we-reg-body\$all-unbound-in-body
all-unbound-in-body (indices ('d, x, y), we-reg-body (m, n, ti))

EVENT: Enable we-reg-body\$all-unbound-in-body; name this event ‘we-reg-body\$all-unbound-in-body-off’.

DEFINITION:

$$\text{we-reg-body-induction}(m, n, ti, \text{bindings}, \text{state-bindings}, \text{netlist})$$

= **if** $n \simeq 0$ **then** **t**
 else we-reg-body-induction($1 + m$,
 $n - 1$,
 index('q, m),
 dual-eval-body-bindings(1 ,
 we-reg-body(m ,
 n ,
 ti),
 bindings,
 state-bindings,
 netlist),
 state-bindings,
 netlist) **endif**

THEOREM: we-reg-body\$value

fd1slp& (netlist)

→ (collect-value (indices ('q, m , n),
 dual-eval(1 ,
 we-reg-body(m , n , ti),
 bindings,
 state-bindings,
 netlist))
 = v-threefix (collect-value (indices ('g, m , n), state-bindings)))

EVENT: Enable we-reg-body\$value; name this event 'we-reg-body\$value-off'.

THEOREM: we-reg\$value

(we-reg& (netlist, n) $\wedge$ (length (state) = n) $\wedge$ properp (state))

→ (dual-eval(0 , index('we-reg, n), inputs, state, netlist)
 = v-threefix (state))

EVENT: Enable we-reg\$value; name this event 'we-reg\$value-off'.

THEOREM: we-reg-body\$state

(fd1slp& (netlist) $\wedge$ ($\neg$ value('te-buf, bindings)))

→ (collect-value (indices ('g, m , n),
 dual-eval(3 ,
 we-reg-body(m , n , ti),
 bindings,
 state-bindings,
 netlist))
 = fv-if (value ('we-buf, bindings),
 collect-value (indices ('d, m , n), bindings),
 collect-value (indices ('g, m , n), state-bindings)))

EVENT: Enable we-reg-body\$state; name this event ‘we-reg-body\$state-off’.

THEOREM: we-reg\$state
 (we-reg& (*netlist*, *n*)
 $\wedge$ (length (*d*) = *n*)
 $\wedge$ properp (*d*)
 $\wedge$ (length (*state*) = *n*)
 $\wedge$ properp (*state*)
 $\wedge$ ($\neg$ *te*))
 $\rightarrow$ (dual-eval (2,
 index ('we-reg, *n*),
 cons (*clk*, cons (*we*, cons (*te*, cons (*ti*, *d*)))),
 state,
 netlist)
 = fv-if (*we*, *d*, *state*))

EVENT: Enable we-reg\$state; name this event ‘we-reg\$state-off’.

DEFINITION:
 cvzbv (*carry*, *overflow*, *vector*)
 = cons (*carry*, cons (*overflow*, cons (v-zerop (*vector*), *vector*)))

DEFINITION: c (*cvzbv*) = car (*cvzbv*)

DEFINITION: v (*cvzbv*) = cadr (*cvzbv*)

DEFINITION: bv (*cvzbv*) = caddr (*cvzbv*)

DEFINITION: n (*cvzbv*) = v-negp (bv (*cvzbv*))

DEFINITION: zb (*cvzbv*) = caddr (*cvzbv*)

EVENT: Enable c; name this event ‘c-off’.

EVENT: Enable v; name this event ‘v-off’.

EVENT: Enable n; name this event ‘n-off’.

EVENT: Enable zb; name this event ‘zb-off’.

EVENT: Enable bv; name this event ‘bv-off’.

THEOREM: boolp-n
 $\text{bvp}(\text{bv}(v)) \rightarrow \text{boolp}(\text{n}(v))$

THEOREM: boolp-bvp-cvzbv
 $(\text{boolp}(c(\text{cvzbv}(c, v, bv))) = \text{boolp}(c))$
 $\wedge (\text{boolp}(v(\text{cvzbv}(c, v, bv))) = \text{boolp}(v))$
 $\wedge \text{boolp}(\text{zb}(\text{cvzbv}(c, v, bv)))$
 $\wedge (\text{bvp}(\text{bv}(\text{cvzbv}(c, v, bv))) = \text{bvp}(bv))$

THEOREM: bvp-cvzbv
 $(\text{boolp}(c(\text{cvzbv})) \wedge \text{boolp}(v(\text{cvzbv})) \wedge \text{boolp}(\text{zb}(\text{cvzbv})) \wedge \text{bvp}(\text{bv}(\text{cvzbv})))$
 $\rightarrow \text{bvp}(\text{cvzbv})$

DEFINITION:
 $\text{cvzbv-v-ror}(c, a)$
 $= \text{cvzbv}(\text{if } a \simeq \text{nil} \text{ then } c$
 $\quad \text{else } \text{nth}(0, a) \text{ endif,}$
 $\quad \mathbf{f},$
 $\quad \text{v-ror}(a, c))$

DEFINITION:
 $\text{cvzbv-v-adder}(c, a, b)$
 $= \text{cvzbv}(\text{v-adder-carry-out}(c, a, b),$
 $\quad \text{v-adder-overflowp}(c, a, b),$
 $\quad \text{v-adder-output}(c, a, b))$

DEFINITION: $\text{cvzbv-v-lsl}(a) = \text{cvzbv-v-adder}(\mathbf{f}, a, a)$

DEFINITION:
 $\text{cvzbv-v-subtractor}(c, a, b)$
 $= \text{cvzbv}(\text{v-subtractor-carry-out}(c, a, b),$
 $\quad \text{v-subtractor-overflowp}(c, a, b),$
 $\quad \text{v-subtractor-output}(c, a, b))$

DEFINITION:
 $\text{cvzbv-inc}(a) = \text{cvzbv-v-adder}(\mathbf{t}, a, \text{nat-to-v}(0, \text{length}(a)))$

DEFINITION:
 $\text{cvzbv-neg}(a) = \text{cvzbv-v-subtractor}(\mathbf{f}, a, \text{nat-to-v}(0, \text{length}(a)))$

DEFINITION:
 $\text{cvzbv-dec}(a) = \text{cvzbv-v-subtractor}(\mathbf{t}, \text{nat-to-v}(0, \text{length}(a)), a)$

DEFINITION: $\text{cvzbv-v-not}(a) = \text{cvzbv}(\mathbf{f}, \mathbf{f}, \text{v-not}(a))$

DEFINITION:

```

cvzbv-v-asr (a)
=  cvzbv (if listp (a) then nth (0, a)
           else f endif,
         f,
         v-asr (a))

```

DEFINITION:

```

cvzbv-v-lsr (a)
=  cvzbv (if listp (a) then nth (0, a)
           else f endif,
         f,
         v-lsr (a))

```

DEFINITION:

```

v-alu (c, a, b, op)
=  if op = list (f, f, f, f) then cvzbv (f, f, v-buf (a))
    elseif op = list (t, f, f, f) then cvzbv-inc (a)
    elseif op = list (f, t, f, f) then cvzbv-v-adder (c, a, b)
    elseif op = list (t, t, f, f) then cvzbv-v-adder (f, a, b)
    elseif op = list (f, f, t, f) then cvzbv-neg (a)
    elseif op = list (t, f, t, f) then cvzbv-dec (a)
    elseif op = list (f, t, t, f) then cvzbv-v-subtractor (c, a, b)
    elseif op = list (t, t, t, f) then cvzbv-v-subtractor (f, a, b)
    elseif op = list (f, f, f, t) then cvzbv-v-ror (c, a)
    elseif op = list (t, f, f, t) then cvzbv-v-asr (a)
    elseif op = list (f, t, f, t) then cvzbv-v-lsr (a)
    elseif op = list (t, t, f, t) then cvzbv (f, f, v-xor (a, b))
    elseif op = list (f, f, t, t) then cvzbv (f, f, v-or (a, b))
    elseif op = list (t, f, t, t) then cvzbv (f, f, v-and (a, b))
    elseif op = list (f, t, t, t) then cvzbv-v-not (a)
    else cvzbv (f, f, v-buf (a)) endif

```

EVENT: Enable v-alu; name this event ‘v-alu-off’.

THEOREM: boolp-c-v-alu

$$(\text{boolp}(c) \wedge \text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow \text{boolp}(c(\text{v-alu}(c, a, b, op)))$$

THEOREM: boolp-v-v-alu

$$(\text{boolp}(c) \wedge \text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow \text{boolp}(v(\text{v-alu}(c, a, b, op)))$$

THEOREM: boolp-zb-v-alu

$$\text{boolp}(\text{zb}(\text{v-alu}(c, a, b, op)))$$

THEOREM: bvp-bv-v-alu
 $\text{bvp}(a) \rightarrow \text{bvp}(\text{bv}(\text{v-alu}(c, a, b, op)))$

THEOREM: bvp-v-alu
 $(\text{bvp}(a) \wedge \text{boolp}(c) \wedge (\text{length}(a) \neq 0)) \rightarrow \text{bvp}(\text{v-alu}(c, a, b, op))$

THEOREM: length-cvzbv-subtractor
 $\text{length}(\text{cvzbv-v-subtractor}(c, a, b)) = (1 + (1 + (1 + \text{length}(a))))$

THEOREM: length-cvzbv-adder
 $\text{length}(\text{cvzbv-v-adder}(c, a, b)) = (1 + (1 + (1 + \text{length}(a))))$

THEOREM: length-v-alu
 $\text{length}(\text{v-alu}(c, a, b, op)) = (1 + (1 + (1 + \text{length}(a))))$

THEOREM: bvp-length-bv
 $(\text{length}(\text{bv}(x)) = (\text{length}(x) - 3))$
 $\wedge ((\text{bvp}(x)$
 $\quad \wedge \text{ if length}(x) < 3 \text{ then f}$
 $\quad \text{ else t endif})$
 $\rightarrow \text{bvp}(\text{bv}(x)))$

DEFINITION:
 $\text{unary-op-code-p}(op\text{-code})$
 $= ((op\text{-code} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$
 $\quad \vee (op\text{-code} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})))$

EVENT: Enable unary-op-code-p; name this event ‘unary-op-code-p-off’.

DEFINITION: $\text{v-alu-1}(c, a, op\text{-code}) = \text{v-alu}(c, a, a, op\text{-code})$

EVENT: Enable v-alu-1; name this event ‘v-alu-1-off’.

THEOREM: bvp-v-alu-1
 $(\text{bvp}(a) \wedge \text{boolp}(c) \wedge (\text{length}(a) \neq 0)) \rightarrow \text{bvp}(\text{v-alu-1}(c, a, op))$

THEOREM: length-v-alu-1
 $\text{length}(\text{v-alu-1}(c, a, op)) = (1 + (1 + (1 + \text{length}(a))))$

THEOREM: unary-op-code-p->v-alu=v-alu-1
 unary-op-code-p (*op-code*)
 $\rightarrow$ (v-alu (*c*, *a*, *b*, *op-code*) = v-alu-1 (*c*, *a*, *op-code*))

EVENT: Enable unary-op-code-p->v-alu=v-alu-1; name this event 'unary-op-code-p->v-alu=v-alu-1-off'.

DEFINITION: ALU-INC-OP = list (**t**, **f**, **f**, **f**)

EVENT: Enable alu-inc-op; name this event 'alu-inc-op-off'.

EVENT: Enable *1*alu-inc-op; name this event 'g*1*alu-inc-op-off'.

THEOREM: bvp-length-alu-inc-op
 (length (ALU-INC-OP) = 4) $\wedge$ bvp (ALU-INC-OP)

DEFINITION: ALU-DEC-OP = list (**t**, **f**, **t**, **f**)

EVENT: Enable alu-dec-op; name this event 'alu-dec-op-off'.

EVENT: Enable *1*alu-dec-op; name this event 'g*1*alu-dec-op-off'.

THEOREM: bvp-length-alu-dec-op
 (length (ALU-DEC-OP) = 4) $\wedge$ bvp (ALU-DEC-OP)

THEOREM: bv-v-alu-alu-inc-alu-dec
 (bv (v-alu (*c*, *a*, *b*, ALU-INC-OP)) = v-inc (*a*))
 $\wedge$ (bv (v-alu (*c*, *a*, *b*, ALU-DEC-OP)) = v-dec (*a*))

DEFINITION:
 decode-mode (*op0*, *op1*, *op2*, *op3*) = b-nor (b-nor3 (*op0*, *op1*, *op2*), *op3*)

DEFINITION:
 f\$decode-mode (*op0*, *op1*, *op2*, *op3*) = f-nor (f-nor3 (*op0*, *op1*, *op2*), *op3*)

EVENT: Enable f\$decode-mode; name this event 'f\$decode-mode-off'.

DEFINITION:
 DECODE-MODE*
 = ' (decode-mode
 (*op0* *op1* *op2* *op3*)
 (*w-1*)
 ((*g-0* (*w-0*) b-nor3 (*op0* *op1* *op2*))
 (*g-1* (*w-1*) b-nor (*w-0* *op3*)))
 nil)

DEFINITION:

decode-mode& (*netlist*)
 = ((lookup-module ('**decode-mode**, *netlist*) = DECODE-MODE*)
 $\wedge$ b-nor3& (delete-module ('**decode-mode**, *netlist*))
 $\wedge$ b-nor& (delete-module ('**decode-mode**, *netlist*)))

EVENT: Enable decode-mode&; name this event 'decode-mode&-off'.

DEFINITION:

DECODE-MODE\$NETLIST = cons (DECODE-MODE*, B-NOR3\$NETLIST $\cup$ B-NOR\$NETLIST)

THEOREM: decode-mode\$value

decode-mode& (*netlist*)
 $\rightarrow$ (dual-eval (0, '**decode-mode**, list (*op0*, *op1*, *op2*, *op3*), *state*, *netlist*)
 = list (f\$decode-mode (*op0*, *op1*, *op2*, *op3*)))

EVENT: Enable decode-mode\$value; name this event 'decode-mode\$value-off'.

THEOREM: f\$decode-mode=decode-mode

(boolp (*op0*) $\wedge$ boolp (*op1*) $\wedge$ boolp (*op2*) $\wedge$ boolp (*op3*))
 $\rightarrow$ (f\$decode-mode (*op0*, *op1*, *op2*, *op3*) = decode-mode (*op0*, *op1*, *op2*, *op3*))

THEOREM: decode-mode\$value-zero

decode-mode& (*netlist*)
 $\rightarrow$ (dual-eval (0, '**decode-mode**, ALU-INC-OP, *state*, *netlist*) = list (**t**))

EVENT: Enable decode-mode\$value-zero; name this event 'decode-mode\$value-zero-off'.

DEFINITION:

decode-prop (*zero*, *swap*, *op0*, *op1*, *op2*, *op3*)
 = list (b-nand3 (b-nand4 (b-not (*op0*),
 b-not (*op1*),
 b-not (b-not (*op2*)),
 b-not (b-not (*op3*))),
 b-nand (b-not (b-not (*op1*)), b-not (*op3*)),
 b-nand3 (b-not (*op2*), b-not (*op3*), b-not (b-not (*swap*)))),
 b-nor (b-not (*op2*),
 b-nor (b-not (*op3*), b-nor (b-not (b-not (*op0*)), b-not (*op1*))),
 b-and (b-nand3 (b-nand (b-not (b-not (*op3*)),
 b-equiv (b-not (b-not (*op0*)), b-not (b-not (*op1*))),
 b-nand (b-not (*op2*),
 b-nand (b-not (b-not (*swap*)), b-not (*op3*))),

$$\begin{aligned} & \text{b-nand4}(\text{b-not}(\text{b-not}(\text{op0})), \\ & \quad \text{b-not}(\text{op1}), \\ & \quad \text{b-not}(\text{b-not}(\text{op2})), \\ & \quad \text{b-not}(\text{op3}))), \\ & \text{b-not}(\text{zero})) \end{aligned}$$

DEFINITION:

$\text{f\$decode-prop}(\text{zero}, \text{swap}, \text{op0}, \text{op1}, \text{op2}, \text{op3})$
 $= \text{list}(\text{f-nand3}(\text{f-nand4}(\text{f-not}(\text{op0}),$
 $\quad \text{f-not}(\text{op1}),$
 $\quad \text{f-not}(\text{f-not}(\text{op2})),$
 $\quad \text{f-not}(\text{f-not}(\text{op3}))),$
 $\quad \text{f-nand}(\text{f-not}(\text{f-not}(\text{op1})), \text{f-not}(\text{op3})),$
 $\quad \text{f-nand3}(\text{f-not}(\text{op2}), \text{f-not}(\text{op3}), \text{f-not}(\text{f-not}(\text{swap})))),$
 $\text{f-nor}(\text{f-not}(\text{op2}),$
 $\quad \text{f-nor}(\text{f-not}(\text{op3}), \text{f-nor}(\text{f-not}(\text{f-not}(\text{op0})), \text{f-not}(\text{op1}))),$
 $\text{f-and}(\text{f-nand3}(\text{f-nand}(\text{f-not}(\text{f-not}(\text{op3})),$
 $\quad \text{f-equiv}(\text{f-not}(\text{f-not}(\text{op0})), \text{f-not}(\text{f-not}(\text{op1}))),$
 $\quad \text{f-nand}(\text{f-not}(\text{op2}),$
 $\quad \text{f-nand}(\text{f-not}(\text{f-not}(\text{swap})), \text{f-not}(\text{op3}))),$
 $\text{f-nand4}(\text{f-not}(\text{f-not}(\text{op0})),$
 $\quad \text{f-not}(\text{op1}),$
 $\quad \text{f-not}(\text{f-not}(\text{op2})),$
 $\quad \text{f-not}(\text{op3}))),$
 $\text{f-not}(\text{zero})))$

EVENT: Enable f\$decode-prop; name this event ‘f\$decode-prop-off’.

DEFINITION:

DECODE-PROP*

$= \text{'(decode-prop}$
 $\quad (\text{zero swap op0 op1 op2 op3})$
 $\quad (\text{w-16 w-24 w-48})$
 $\quad ((\text{g-47 (w-47) b-not (zero)})$
 $\quad (\text{g-44 (w-44) b-not (op3)})$
 $\quad (\text{g-42 (w-42) b-not (op2)})$
 $\quad (\text{g-43 (w-43) b-not (w-42)})$
 $\quad (\text{g-41 (w-41) b-not (op1)})$
 $\quad (\text{g-39 (w-39) b-not (op0)})$
 $\quad (\text{g-40 (w-40) b-not (w-39)})$
 $\quad (\text{g-45 (w-45) b-nand4 (w-40 w-41 w-43 w-44)})$
 $\quad (\text{g-34 (w-34) b-not (swap)})$
 $\quad (\text{g-35 (w-35) b-not (w-34)})$
 $\quad (\text{g-37 (w-37) b-nand (w-35 w-44)})$


```

(g-38 (w-38) b-nand (w-42 w-37))
(g-30 (w-30) b-not (w-41))
(g-31 (w-31) b-equiv (w-40 w-30))
(g-26 (w-26) b-not (w-44))
(g-32 (w-32) b-nand (w-26 w-31))
(g-46 (w-46) b-nand3 (w-32 w-38 w-45))
(g-48 (w-48) b-and (w-46 w-47))
(g-22 (w-22) b-nor (w-40 w-41))
(g-23 (w-23) b-nor (w-44 w-22))
(g-24 (w-24) b-nor (w-42 w-23))
(g-15 (w-15) b-nand3 (w-42 w-44 w-35))
(g-10 (w-10) b-nand (w-30 w-44))
(g-6 (w-6) b-nand4 (w-39 w-41 w-43 w-26))
(g-16 (w-16) b-nand3 (w-6 w-10 w-15)))
nil)

```

DEFINITION:

```

decode-prop& (netlist)
= ((lookup-module ('decode-prop, netlist) = DECODE-PROP*)
  ∧ b-equiv& (delete-module ('decode-prop, netlist))
  ∧ b-not& (delete-module ('decode-prop, netlist))
  ∧ b-and& (delete-module ('decode-prop, netlist))
  ∧ b-nor& (delete-module ('decode-prop, netlist))
  ∧ b-nand& (delete-module ('decode-prop, netlist))
  ∧ b-nand4& (delete-module ('decode-prop, netlist))
  ∧ b-nand3& (delete-module ('decode-prop, netlist)))

```

EVENT: Enable decode-prop&; name this event 'decode-prop&-off'.

DEFINITION:

```

DECODE-PROP$NETLIST
= cons (DECODE-PROP*,
  (((((B-EQUIV$NETLIST ∪ B-NOT$NETLIST) ∪ B-AND$NETLIST)
    ∪ B-NOR$NETLIST)
    ∪ B-NAND$NETLIST)
    ∪ B-NAND4$NETLIST)
    ∪ B-NAND3$NETLIST)

```

THEOREM: decode-prop\$value

```

decode-prop& (netlist)
→ (dual-eval (0,
  'decode-prop,
  list (zero, swap, op0, op1, op2, op3),
  state,

```

$$\begin{aligned}
& \text{netlist}) \\
& = \text{f\$decode-prop}(\text{zero}, \text{swap}, \text{op0}, \text{op1}, \text{op2}, \text{op3})
\end{aligned}$$

EVENT: Enable decode-prop\$value; name this event ‘decode-prop\$value-off’.

THEOREM: f\$decode-prop=decode-prop
 (boolp (zero)
 $\wedge$ boolp (swap)
 $\wedge$ boolp (op0)
 $\wedge$ boolp (op1)
 $\wedge$ boolp (op2)
 $\wedge$ boolp (op3))
 $\rightarrow$ (f\$decode-prop (zero, swap, op0, op1, op2, op3)
 = decode-prop (zero, swap, op0, op1, op2, op3))

THEOREM: decode-prop\$value-zero
 decode-prop& (netlist)
 $\rightarrow$ (dual-eval (0,
 'decode-prop,
 cons (t, cons (f, ALU-INC-OP)),
 state,
 netlist)
 = list (f, f, f))

EVENT: Enable decode-prop\$value-zero; name this event ‘decode-prop\$value-zero-off’.

DEFINITION:
 decode-gen (zero, swap, op0, op1, op2, op3)
 = list (b-nand3 (b-nand3 (b-not (b-not (op0)),
 b-not (b-not (op3)),
 b-xor (b-not (b-not (op1)), b-not (b-not (op2)))),
 b-nand3 (b-not (b-not (op2)),
 b-not (op3),
 b-nand (b-not (op1), b-not (swap))),
 b-nand3 (b-not (b-not (op1)), b-not (op2), b-not (op3))),
 b-nor (b-nand (b-nand4 (b-not (b-not (op0)),
 b-not (b-not (op1)),
 b-not (op2),
 b-not (b-not (op3))),
 b-nand3 (b-not (b-not (op2)),
 b-not (op3),
 b-nand (b-not (op1), b-not (swap))))),
 b-not (b-not (zero))),

```

b-nor (b-nand3 (b-nand3 (b-not (b-not (op0)),
                        b-not (b-not (op3))),
        b-xor (b-not (b-not (op1)), b-not (b-not (op2)))))
b-nand3 (b-not (b-not (op0)),
        b-not (op1),
        b-not (b-not (op2))),
b-nand3 (b-not (b-not (op1)), b-not (op2), b-not (op3)),
b-not (b-not (zero)))

```

DEFINITION:

```

f$decode-gen (zero, swap, op0, op1, op2, op3)
= list (f-nand3 (f-nand3 (f-not (f-not (op0)),
                        f-not (f-not (op3))),
        f-xor (f-not (f-not (op1)), f-not (f-not (op2)))))
f-nand3 (f-not (f-not (op2)),
        f-not (op3),
        f-nand (f-not (op1), f-not (swap))),
f-nand3 (f-not (f-not (op1)), f-not (op2), f-not (op3))),
f-nor (f-nand (f-nand4 (f-not (f-not (op0)),
                        f-not (f-not (op1)),
                        f-not (op2),
                        f-not (f-not (op3)))),
        f-nand3 (f-not (f-not (op2)),
                f-not (op3),
                f-nand (f-not (op1), f-not (swap)))),
        f-not (f-not (zero))),
f-nor (f-nand3 (f-nand3 (f-not (f-not (op0)),
                        f-not (f-not (op3))),
        f-xor (f-not (f-not (op1)), f-not (f-not (op2)))))
f-nand3 (f-not (f-not (op0)),
        f-not (op1),
        f-not (f-not (op2))),
f-nand3 (f-not (f-not (op1)), f-not (op2), f-not (op3))),
f-not (f-not (zero)))

```

EVENT: Enable f\$decode-gen; name this event 'f\$decode-gen-off'.

DEFINITION:

DECODE-GEN*

```

= ' (decode-gen
    (zero swap op0 op1 op2 op3)
    (w-22 w-41 w-66)
    ((g-64 (w-64) b-not (zero))
     (g-65 (w-65) b-not (w-64))

```

```

(g-61 (w-61) b-not (op3))
(g-60 (w-60) b-not (op2))
(g-58 (w-58) b-not (op1))
(g-59 (w-59) b-not (w-58))
(g-62 (w-62) b-nand3 (w-59 w-60 w-61))
(g-56 (w-56) b-not (w-60))
(g-52 (w-52) b-not (op0))
(g-53 (w-53) b-not (w-52))
(g-57 (w-57) b-nand3 (w-53 w-58 w-56))
(g-50 (w-50) b-xor (w-59 w-56))
(g-45 (w-45) b-not (w-61))
(g-51 (w-51) b-nand3 (w-53 w-45 w-50))
(g-63 (w-63) b-nand3 (w-51 w-57 w-62))
(g-66 (w-66) b-nor (w-63 w-65))
(g-35 (w-35) b-not (swap))
(g-36 (w-36) b-nand (w-58 w-35))
(g-37 (w-37) b-nand3 (w-56 w-61 w-36))
(g-30 (w-30) b-nand4 (w-53 w-59 w-60 w-45))
(g-38 (w-38) b-nand (w-30 w-37))
(g-41 (w-41) b-nor (w-38 w-65))
(g-22 (w-22) b-nand3 (w-51 w-37 w-62))
nil)

```

DEFINITION:

decode-gen& (*netlist*)

```

= ((lookup-module ('decode-gen, netlist) = DECODE-GEN*)
  ∧ b-xor& (delete-module ('decode-gen, netlist))
  ∧ b-not& (delete-module ('decode-gen, netlist))
  ∧ b-nand4& (delete-module ('decode-gen, netlist))
  ∧ b-nand& (delete-module ('decode-gen, netlist))
  ∧ b-nor& (delete-module ('decode-gen, netlist))
  ∧ b-nand3& (delete-module ('decode-gen, netlist)))

```

EVENT: Enable decode-gen&; name this event 'decode-gen&-off'.

DEFINITION:

DECODE-GEN\$NETLIST

```

= cons (DECODE-GEN*,
  (((B-XOR$NETLIST ∪ B-NOT$NETLIST) ∪ B-NAND4$NETLIST)
   ∪ B-NAND$NETLIST)
  ∪ B-NOR$NETLIST)
  ∪ B-NAND3$NETLIST)

```

THEOREM: decode-gen\$value

```

decode-gen& (netlist)
→ (dual-eval (0,
               'decode-gen,
               list (zero, swap, op0, op1, op2, op3),
               state,
               netlist)
    = f$decode-gen (zero, swap, op0, op1, op2, op3))

```

EVENT: Enable decode-gen\$value; name this event 'decode-gen\$value-off'.

THEOREM: f\$decode-gen=decode-gen

```

(boolp (zero)
 ∧ boolp (swap)
 ∧ boolp (op0)
 ∧ boolp (op1)
 ∧ boolp (op2)
 ∧ boolp (op3))
→ (f$decode-gen (zero, swap, op0, op1, op2, op3)
    = decode-gen (zero, swap, op0, op1, op2, op3))

```

THEOREM: decode-gen\$value-zero

```

decode-gen& (netlist)
→ (dual-eval (0, 'decode-gen, cons (t, cons (f, ALU-INC-OP)), state, netlist)
    = list (f, f, f))

```

EVENT: Enable decode-gen\$value-zero; name this event 'decode-gen\$value-zero-off'.

DEFINITION:

```

f$mpg (zsop)
= append (f$decode-gen (car (zsop),
                        cadr (zsop),
                        caddr (zsop),
                        caddr (zsop),
                        caddr (zsop),
                        caddr (zsop)),
          append (f$decode-prop (car (zsop),
                                cadr (zsop),
                                caddr (zsop),
                                caddr (zsop),
                                caddr (zsop),
                                caddr (zsop)),
                  list (f$decode-mode (caddr (zsop),
                                       caddr (zsop)),

```

caddrddr (zso),
caddrddr (zso))))))

EVENT: Enable f\$mpg; name this event ‘f\$mpg-off’.

DEFINITION:

mpg (zso)
= append (decode-gen (car (zso),
 cadr (zso),
 caddr (zso),
 caddrddr (zso),
 caddrddr (zso),
 caddrddr (zso)),
 append (decode-prop (car (zso),
 cadr (zso),
 caddr (zso),
 caddrddr (zso),
 caddrddr (zso)),
 list (decode-mode (caddr (zso),
 caddrddr (zso),
 caddrddr (zso),
 caddrddr (zso))))))

EVENT: Enable mpg; name this event ‘mpg-off’.

THEOREM: f\$mpg=mpg

(bvp (zso) $\wedge$ (length (zso) = 6)) $\rightarrow$ (f\$mpg (zso) = mpg (zso))

THEOREM: properp-length-f\$mpg

properp (f\$mpg (zso)) $\wedge$ (length (f\$mpg (zso)) = 7)

THEOREM: length-mpg

length (mpg (zso)) = 7

THEOREM: bvp-mpg

bvp (mpg (zso))

THEOREM: properp-mpg

properp (mpg (zso))

THEOREM: mpg-if-op-code

mpg (cons (a,
 cons (b,

if c then d
else e endif))
 = **if c then** mpg (cons (a , cons (b , d)))
else mpg (cons (a , cons (b , e))) **endif**

THEOREM: mpg-zero

mpg (cons (t , cons (f , ALU-INC-OP))) = list (f , f , f , f , f , t)

EVENT: Enable mpg-zero; name this event 'mpg-zero-off'.

DEFINITION:

MPG*

= ' (mpg
 (zero swap op0 op1 op2 op3)
 (gbn gan ga pb pan pa mode)
 ((m (mode) decode-mode (op0 op1 op2 op3))
 (p
 (pb pan pa)
 decode-prop
 (zero swap op0 op1 op2 op3))
 (g
 (gbn gan ga)
 decode-gen
 (zero swap op0 op1 op2 op3)))
 nil)

DEFINITION:

mpg& (*netlist*)

= ((lookup-module ('mpg, *netlist*) = MPG*)
 ∧ decode-mode& (delete-module ('mpg, *netlist*))
 ∧ decode-prop& (delete-module ('mpg, *netlist*))
 ∧ decode-gen& (delete-module ('mpg, *netlist*)))

EVENT: Enable mpg&; name this event 'mpg&-off'.

DEFINITION:

MPG\$NETLIST

= cons (MPG*,
 (DECODE-MODE\$NETLIST ∪ DECODE-PROP\$NETLIST)
 ∪ DECODE-GEN\$NETLIST)

THEOREM: mpg\$value

(mpg& (*netlist*) ∧ properp (*zsop*) ∧ (length (*zsop*) = 6))
 → (dual-eval (0, 'mpg, *zsop*, *state*, *netlist*) = f\$mpg (*zsop*))

EVENT: Enable mpg\$value; name this event ‘mpg\$value-off’.

THEOREM: mpg\$value-zero

$(\text{mpg} \& (\text{netlist}) \wedge (\text{zsop} = \text{cons}(\mathbf{t}, \text{cons}(\mathbf{f}, \text{ALU-INC-OP}))))$
 $\rightarrow (\text{dual-eval}(0, \text{'mpg'}, \text{zsop}, \text{state}, \text{netlist}) = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable mpg\$value-zero; name this event ‘mpg\$value-zero-off’.

DEFINITION:

carry-in-help (*czop*)

= b-or (b-nand3 (b-nand3 (b-not (caddr (*czop*)),
 b-not (caddr (*czop*)),
 b-not (caddr (*czop*))),
 b-nand3 (b-not (caddr (*czop*)),
 b-not (caddr (*czop*)),
 b-not (b-not (caddr (*czop*)))),
 b-nand3 (b-not (b-not (caddr (*czop*))),
 b-not (b-not (caddr (*czop*))),
 b-not (b-not (caddr (*czop*)))),
 b-nand3 (b-nand (b-not (b-not (caddr (*czop*))),
 b-not (b-not (car (*czop*))),
 b-nand3 (b-not (caddr (*czop*)),
 b-not (caddr (*czop*)),
 b-not (b-not (car (*czop*))),
 b-nand3 (b-not (caddr (*czop*)),
 b-not (b-not (caddr (*czop*))),
 b-not (car (*czop*))))))

EVENT: Enable carry-in-help; name this event ‘carry-in-help-off’.

DEFINITION:

f\$carry-in-help (*czop*)

= f-or (f-nand3 (f-nand3 (f-not (caddr (*czop*)),
 f-not (caddr (*czop*)),
 f-not (caddr (*czop*))),
 f-nand3 (f-not (caddr (*czop*)),
 f-not (caddr (*czop*)),
 f-not (f-not (caddr (*czop*)))),
 f-nand3 (f-not (f-not (caddr (*czop*))),
 f-not (f-not (caddr (*czop*))),
 f-not (f-not (caddr (*czop*)))),
 f-nand3 (f-nand (f-not (f-not (caddr (*czop*))),
 f-not (f-not (car (*czop*))),
 f-not (f-not (car (*czop*))))))


```

f-nand3 (f-not (caddr (czop)),
         f-not (caddrdr (czop)),
         f-not (f-not (car (czop))))),
f-nand3 (f-not (caddr (czop)),
         f-not (f-not (caddrdr (czop))),
         f-not (car (czop))))))

```

EVENT: Enable f\$carry-in-help; name this event 'f\$carry-in-help-off'.

THEOREM: f\$carry-in-help=carry-in-help
 (bvp (czop) $\wedge$ (length (czop) = 6))
 $\rightarrow$ (f\$carry-in-help (czop) = carry-in-help (czop))

DEFINITION:

```

CARRY-IN-HELP*
= '(carry-in-help
   (cin z op0in op1in op2in op3in)
   (cout)
   ((g0 (c- c) b-nbuf (cin))
    (g1 (op0- op0) b-nbuf (op0in))
    (g2 (op1- op1) b-nbuf (op1in))
    (g3 (op2- op2) b-nbuf (op2in))
    (g4 (op3- op3) b-nbuf (op3in))
    (g5 (s5) b-nand3 (op1- op2- op3-))
    (g6 (s6) b-nand3 (op0- op1- op2))
    (g7 (s7) b-nand3 (op0 op1 op2))
    (g8 (s8) b-nand3 (s5 s6 s7))
    (g9 (s9) b-nand (op3 c))
    (g10 (s10) b-nand3 (op0- op2- c))
    (g11 (s11) b-nand3 (op0- op2 c-))
    (g12 (s12) b-nand3 (s9 s10 s11))
    (g13 (cout) b-or (s8 s12)))
   nil)

```

DEFINITION:

```

carry-in-help& (netlist)
= ((lookup-module ('carry-in-help, netlist) = CARRY-IN-HELP*)
    $\wedge$  b-nbuf& (delete-module ('carry-in-help, netlist))
    $\wedge$  b-nand& (delete-module ('carry-in-help, netlist))
    $\wedge$  b-nand3& (delete-module ('carry-in-help, netlist))
    $\wedge$  b-or& (delete-module ('carry-in-help, netlist)))

```

EVENT: Enable carry-in-help&; name this event 'carry-in-help&-off'.

DEFINITION:

CARRY-IN-HELP\$NETLIST
 = cons (CARRY-IN-HELP*,
 ((B-NBUF\$NETLIST $\cup$ B-NAND\$NETLIST) $\cup$ B-NAND3\$NETLIST)
 $\cup$ B-OR\$NETLIST)

THEOREM: carry-in-help-zero

carry-in-help (cons (c, cons (t, ALU-INC-OP))) = t

EVENT: Enable carry-in-help-zero; name this event ‘carry-in-help-zero-off’.

THEOREM: carry-in-help-if-op-code

carry-in-help (cons (a,
 cons (b,
 if c **then** d
 else e **endif**)))
 = **if** c **then** carry-in-help (cons (a, cons (b, d)))
 else carry-in-help (cons (a, cons (b, e))) **endif**

THEOREM: carry-in-help\$value

carry-in-help& (netlist)
 $\rightarrow$ (dual-eval (0, 'carry-in-help, czop, state, netlist)
 = list (f\$carry-in-help (czop)))

EVENT: Enable carry-in-help\$value; name this event ‘carry-in-help\$value-off’.

THEOREM: carry-in-help\$value-zero

(carry-in-help& (netlist) $\wedge$ (czop = cons (c, cons (t, ALU-INC-OP))))
 $\rightarrow$ (dual-eval (0, 'carry-in-help, czop, state, netlist) = list (t))

EVENT: Enable carry-in-help\$value-zero; name this event ‘carry-in-help\$value-zero-off’.

DEFINITION:

p-cell (a, an, b, pa, pan, pb)
 = b-nand3 (b-nand (a, pa), b-nand (an, pan), b-nand (b, pb))

DEFINITION:

f\$p-cell (a, an, b, pa, pan, pb)
 = f-nand3 (f-nand (a, pa), f-nand (an, pan), f-nand (b, pb))

EVENT: Enable f\$p-cell; name this event ‘f\$p-cell-off’.

DEFINITION:

P-CELL*

```
= 'p-cell
  (a an b pa pan pb)
  (w-3)
  ((g-2 (w-2) b-nand (b pb))
   (g-1 (w-1) b-nand (an pan))
   (g-0 (w-0) b-nand (a pa))
   (g-3 (w-3) b-nand3 (w-0 w-1 w-2)))
  nil)
```

DEFINITION:

p-cell& (*netlist*)

```
= ((lookup-module ('p-cell, netlist) = P-CELL*)
   ∧ b-nand& (delete-module ('p-cell, netlist))
   ∧ b-nand3& (delete-module ('p-cell, netlist)))
```

EVENT: Enable p-cell&; name this event 'p-cell&-off'.

DEFINITION:

P-CELL\$NETLIST = cons (P-CELL*, B-NAND\$NETLIST ∪ B-NAND3\$NETLIST)

THEOREM: p-cell\$value

p-cell& (*netlist*)

```
→ (dual-eval (0, 'p-cell, list (a, an, b, pa, pan, pb), state, netlist)
   = list (f$p-cell (a, an, b, pa, pan, pb)))
```

EVENT: Enable p-cell\$value; name this event 'p-cell\$value-off'.

THEOREM: f\$p-cell=p-cell

```
(boolp (a)
 ∧ boolp (an)
 ∧ boolp (b)
 ∧ boolp (pa)
 ∧ boolp (pan)
 ∧ boolp (pb))
→ (f$p-cell (a, an, b, pa, pan, pb) = p-cell (a, an, b, pa, pan, pb))
```

THEOREM: p-cell\$value-zero

p-cell& (*netlist*)

```
→ (dual-eval (0, 'p-cell, list (a, an, b, f, f, f), state, netlist) = list (f))
```

DEFINITION:

g-cell (*a, an, bn, ga, gan, gbn*)

```
= b-and3 (b-nand (a, ga), b-nand (an, gan), b-nand (bn, gbn))
```

DEFINITION:

f\$g-cell(*a, an, bn, ga, gan, gbn*)
 = f-and3(f-nand(*a, ga*), f-nand(*an, gan*), f-nand(*bn, gbn*))

EVENT: Enable f\$g-cell; name this event 'f\$g-cell-off'.

DEFINITION:

G-CELL*
 = '(g-cell
 (a an bn ga gan gbn)
 (w-3)
 ((g-2 (w-2) b-nand (bn gbn))
 (g-1 (w-1) b-nand (an gan))
 (g-0 (w-0) b-nand (a ga))
 (g-3 (w-3) b-and3 (w-0 w-1 w-2)))
 nil)

DEFINITION:

g-cell&(*netlist*)
 = ((lookup-module('g-cell, *netlist*) = G-CELL*)
 ∧ b-nand&(delete-module('g-cell, *netlist*))
 ∧ b-and3&(delete-module('g-cell, *netlist*)))

EVENT: Enable g-cell&; name this event 'g-cell&-off'.

DEFINITION:

G-CELL\$NETLIST = cons(G-CELL*, B-NAND\$NETLIST ∪ B-AND3\$NETLIST)

THEOREM: g-cell\$value

g-cell&(*netlist*)
 → (dual-eval(0, 'g-cell, list(*a, an, bn, ga, gan, gbn*), *state, netlist*)
 = list(f\$g-cell(*a, an, bn, ga, gan, gbn*)))

EVENT: Enable g-cell\$value; name this event 'g-cell\$value-off'.

THEOREM: f\$g-cell=g-cell

(boolp(*a*)
 ∧ boolp(*an*)
 ∧ boolp(*bn*)
 ∧ boolp(*ga*)
 ∧ boolp(*gan*)
 ∧ boolp(*gbn*))
 → (f\$g-cell(*a, an, bn, ga, gan, gbn*) = g-cell(*a, an, bn, ga, gan, gbn*))

THEOREM: g-cell\$value-zero
g-cell& (netlist)
→ (dual-eval (0, 'g-cell, list (a, an, bn, f, f, f), state, netlist)
= list (t))

DEFINITION:

alu-cell (c, a, b, mpg)
= list (p-cell (a, b-not (a), b, caddrddr (mpg), caddrdr (mpg), caddrdr (mpg)),
g-cell (a, b-not (a), b-not (b), caddr (mpg), cadr (mpg), car (mpg)),
b-equiv3 (b-nand (c, caddrddr (mpg)),
p-cell (a,
b-not (a),
b,
caddrddr (mpg),
caddrdr (mpg),
caddrdr (mpg)),
g-cell (a, b-not (a), b-not (b), caddr (mpg), cadr (mpg), car (mpg))))

DEFINITION:

f\$alu-cell (c, a, b, mpg)
= list (f\$p-cell (a, f-not (a), b, caddrddr (mpg), caddrdr (mpg), caddrdr (mpg)),
f\$g-cell (a, f-not (a), f-not (b), caddr (mpg), cadr (mpg), car (mpg)),
f-equiv3 (f-nand (c, caddrddr (mpg)),
f\$p-cell (a,
f-not (a),
b,
caddrddr (mpg),
caddrdr (mpg),
caddrdr (mpg)),
f\$g-cell (a,
f-not (a),
f-not (b),
caddr (mpg),
cadr (mpg),
car (mpg))))

EVENT: Enable f\$alu-cell; name this event 'f\$alu-cell-off'.

THEOREM: f\$alu-cell=alu-cell

(boolp (c) ∧ boolp (a) ∧ boolp (b) ∧ bvp (mpg) ∧ (length (mpg) = 7))
→ (f\$alu-cell (c, a, b, mpg) = alu-cell (c, a, b, mpg))

DEFINITION:

ALU-CELL*

```

= 'alu-cell
  (c a b gbn gan ga pb pan pa m)
  (p g z)
  ((n0 (an) b-not (a))
   (n1 (bn) b-not (b))
   (p0 (p) p-cell (a an b pa pan pb))
   (g0 (g) g-cell (a an bn ga gan gbn))
   (m0 (mc) b-nand (c m))
   (z0 (z) b-equiv3 (mc p g)))
  nil)

```

DEFINITION:

alu-cell& (*netlist*)

```

= ((lookup-module ('alu-cell, netlist) = ALU-CELL*)
   ^ b-not& (delete-module ('alu-cell, netlist))
   ^ p-cell& (delete-module ('alu-cell, netlist))
   ^ g-cell& (delete-module ('alu-cell, netlist))
   ^ b-nand& (delete-module ('alu-cell, netlist))
   ^ b-equiv3& (delete-module ('alu-cell, netlist)))

```

EVENT: Enable alu-cell&; name this event 'alu-cell&-off'.

DEFINITION:

ALU-CELL\$NETLIST

```

= cons (ALU-CELL*,
        (((B-NOT$NETLIST ∪ P-CELL$NETLIST) ∪ G-CELL$NETLIST)
         ∪ B-NAND$NETLIST)
         ∪ B-EQV3$NETLIST)

```

THEOREM: alu-cell\$value

```

alu-cell& (netlist)
→ (dual-eval (0, 'alu-cell, cons (c, cons (a, cons (b, mpg))), state, netlist)
   = f$alu-cell (c, a, b, mpg))

```

EVENT: Enable alu-cell\$value; name this event 'alu-cell\$value-off'.

THEOREM: alu-cell\$value-zero

```

(alu-cell& (netlist) ∧ (mpg = list (f, f, f, f, f, t)))
→ (dual-eval (0, 'alu-cell, cons (t, cons (a, cons (b, mpg))), state, netlist)
   = list (f, t, f))

```

EVENT: Enable alu-cell\$value-zero; name this event 'alu-cell\$value-zero-off'.

THEOREM: f\$alu-cell-v-threefix-mpg

$(\text{length}(mpg) = 7)$
 $\rightarrow (\text{f\$alu-cell}(c, a, b, \text{v-threefix}(mpg)) = \text{f\$alu-cell}(c, a, b, mpg))$

DEFINITION:

$\text{tv-alu-help}(c, a, b, mpg, tree)$

$=$ **if** $tree \simeq \text{nil}$ **then** $\text{alu-cell}(c, \text{car}(a), \text{car}(b), mpg)$
else $\text{cons}(\text{b-and}(\text{car}(\text{tv-alu-help}(c,$
 $\quad \text{tfirstn}(a, tree),$
 $\quad \text{tfirstn}(b, tree),$
 $\quad mpg,$
 $\quad \text{car}(tree))),$
 $\text{car}(\text{tv-alu-help}(\text{t-carry}(c,$
 $\quad \text{car}(\text{tv-alu-help}(c,$
 $\quad \quad \text{tfirstn}(a,$
 $\quad \quad \quad tree),$
 $\quad \text{tfirstn}(b,$
 $\quad \quad tree),$
 $\quad \quad mpg,$
 $\quad \quad \text{car}(tree))),$
 $\text{cadr}(\text{tv-alu-help}(c,$
 $\quad \text{tfirstn}(a,$
 $\quad \quad tree),$
 $\quad \text{tfirstn}(b,$
 $\quad \quad tree),$
 $\quad \quad mpg,$
 $\quad \quad \text{car}(tree))),$
 $\quad \text{trestn}(a, tree),$
 $\quad \text{trestn}(b, tree),$
 $\quad \quad mpg,$
 $\quad \quad \text{cdr}(tree))),$
 $\text{cons}(\text{t-carry}(\text{cadr}(\text{tv-alu-help}(c,$
 $\quad \text{tfirstn}(a, tree),$
 $\quad \text{tfirstn}(b, tree),$
 $\quad \quad mpg,$
 $\quad \quad \text{car}(tree))),$
 $\text{car}(\text{tv-alu-help}(\text{t-carry}(c,$
 $\quad \text{car}(\text{tv-alu-help}(c,$
 $\quad \quad \text{tfirstn}(a,$
 $\quad \quad \quad tree),$
 $\quad \text{tfirstn}(b,$
 $\quad \quad tree),$
 $\quad \quad mpg,$
 $\quad \quad \text{car}(tree))),$
 $\text{cadr}(\text{tv-alu-help}(c,$

```

tfirstn (a,
         tree),
tfirstn (b,
         tree),
mpg,
car (tree))))),

trestn (a, tree),
trestn (b, tree),
mpg,
cdr (tree))),
cadr (tv-alu-help (t-carry (c,
                           car (tv-alu-help (c,
                                             tfirstn (a,
                                                         tree),
                                             tfirstn (b,
                                                         tree),
                                             mpg,
                                             car (tree))),
cadr (tv-alu-help (c,
                  tfirstn (a,
                              tree),
                  tfirstn (b,
                              tree),
                  mpg,
                  car (tree))))),

trestn (a, tree),
trestn (b, tree),
mpg,
cdr (tree))))),
append (caddr (tv-alu-help (c,
                           tfirstn (a, tree),
                           tfirstn (b, tree),
                           mpg,
                           car (tree))),
caddr (tv-alu-help (t-carry (c,
                           car (tv-alu-help (c,
                                             tfirstn (a,
                                                         tree),
                                             tfirstn (b,
                                                         tree),
                                             mpg,
                                             car (tree))),
caddr (tv-alu-help (c,
                  tfirstn (a,

```



```

                                tree),
tfirstn (b,
         tree),
mpg,
car (tree))))),
trestn (a, tree),
trestn (b, tree),
mpg,
cdr (tree)))))) endif

```

EVENT: Enable tv-alu-help; name this event ‘tv-alu-help-off’.

THEOREM: cddr-tv-alu-help-length
 $\text{length}(\text{cddr}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree}))) = \text{tree-size}(\text{tree})$

THEOREM: tv-alu-help-length
 $\text{length}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree})) = (1 + (1 + \text{tree-size}(\text{tree})))$

THEOREM: bvp-cddr-tv-alu-help
 $\text{bvp}(\text{cddr}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree})))$

THEOREM: bvp-tv-alu-help
 $\text{bvp}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree}))$

THEOREM: bvp-length-tv-alu-help
 $\text{bvp-length}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree}), n)$
 $= \text{if } (1 + (1 + \text{tree-size}(\text{tree}))) < n \text{ then } \mathbf{f}$
 $\text{else } \mathbf{t} \text{ endif}$

THEOREM: tv-alu-help-zero
 $((\text{mpg} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$
 $\wedge (\text{length}(a) = \text{tree-size}(\text{tree}))$
 $\wedge (c = \mathbf{t}))$
 $\rightarrow (\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree})$
 $= \text{cons}(\mathbf{f}, \text{cons}(\mathbf{t}, \text{make-list}(\text{length}(a), \mathbf{f}))))$

EVENT: Enable tv-alu-help-zero; name this event ‘tv-alu-help-zero-off’.

THEOREM: tv-alu-help-v-and-works
 $(\text{bv2p}(a, b)$
 $\wedge (\text{length}(a) = \text{tree-size}(\text{tree}))$
 $\wedge (\text{mpg} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})))$
 $\rightarrow (\text{cddr}(\text{tv-alu-help}(c, a, b, \text{mpg}, \text{tree})) = \text{v-and}(a, b))$

EVENT: Enable tv-alu-help-v-and-works; name this event ‘tv-alu-help-v-and-works-off’.

THEOREM: tv-alu-help-v-or-works
 $((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
 $\wedge \text{bv2p}(a, b)$
 $\wedge (\text{length}(a) = \text{tree-size}(tree)))$
 $\rightarrow (\text{caddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-or}(a, b))$

EVENT: Enable tv-alu-help-v-or-works; name this event ‘tv-alu-help-v-or-works-off’.

THEOREM: tv-alu-help-v-xor-works
 $((mpg = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
 $\wedge \text{bv2p}(a, b)$
 $\wedge (\text{length}(a) = \text{tree-size}(tree)))$
 $\rightarrow (\text{caddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-xor}(a, b))$

EVENT: Enable tv-alu-help-v-xor-works; name this event ‘tv-alu-help-v-xor-works-off’.

THEOREM: tv-alu-help-v-not-works
 $((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}))$
 $\wedge \text{bvp}(a)$
 $\wedge (\text{length}(a) = \text{tree-size}(tree)))$
 $\rightarrow (\text{caddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-not}(a))$

EVENT: Enable tv-alu-help-v-not-works; name this event ‘tv-alu-help-v-not-works-off’.

THEOREM: tv-alu-help-v-buf-works
 $((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
 $\wedge \text{bvp}(a)$
 $\wedge (\text{length}(a) = \text{tree-size}(tree)))$
 $\rightarrow (\text{caddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-buf}(a))$

EVENT: Enable tv-alu-help-v-buf-works; name this event ‘tv-alu-help-v-buf-works-off’.

THEOREM: tv-alu-help-tv-adder-works
 $((mpg = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$
 $\wedge \text{bvp}(a)$

$$\begin{aligned} & \wedge (\text{length}(a) = \text{tree-size}(tree))) \\ \rightarrow & (\text{tv-alu-help}(c, a, b, mpg, tree) = \text{tv-adder}(c, a, b, tree)) \end{aligned}$$

EVENT: Enable tv-alu-help-tv-adder-works; name this event ‘tv-alu-help-tv-adder-works-off’.

THEOREM: tv-alu-help-tv-subtractor-works

$$\begin{aligned} & ((mpg = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})) \\ & \wedge \text{bvp}(a) \\ & \wedge (\text{length}(a) = \text{tree-size}(tree))) \\ \rightarrow & (\text{tv-alu-help}(c, a, b, mpg, tree) = \text{tv-adder}(c, \text{v-not}(a), b, tree)) \end{aligned}$$

EVENT: Enable tv-alu-help-tv-subtractor-works; name this event ‘tv-alu-help-tv-subtractor-works-off’.

THEOREM: tv-alu-help-tv-inc-a-works

$$\begin{aligned} & ((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})) \\ & \wedge \text{bvp}(a) \\ & \wedge (\text{length}(a) = \text{tree-size}(tree))) \\ \rightarrow & (\text{tv-alu-help}(c, a, b, mpg, tree) \\ & = \text{tv-adder}(c, a, \text{nat-to-v}(0, \text{length}(a)), tree)) \end{aligned}$$

EVENT: Enable tv-alu-help-tv-inc-a-works; name this event ‘tv-alu-help-tv-inc-a-works-off’.

THEOREM: tv-alu-help-tv-inc-b-works

$$\begin{aligned} & ((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})) \\ & \wedge \text{bvp}(b) \\ & \wedge (\text{length}(b) = \text{tree-size}(tree))) \\ \rightarrow & (\text{tv-alu-help}(c, a, b, mpg, tree) \\ & = \text{tv-adder}(c, b, \text{nat-to-v}(0, \text{length}(b)), tree)) \end{aligned}$$

EVENT: Enable tv-alu-help-tv-inc-b-works; name this event ‘tv-alu-help-tv-inc-b-works-off’.

THEOREM: tv-alu-help-tv-dec-a-works

$$\begin{aligned} & ((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})) \\ & \wedge \text{bvp}(a) \\ & \wedge (\text{length}(a) = \text{tree-size}(tree))) \\ \rightarrow & (\text{tv-alu-help}(c, a, b, mpg, tree) \\ & = \text{tv-adder}(c, \text{v-not}(\text{nat-to-v}(0, \text{length}(a))), a, tree)) \end{aligned}$$

EVENT: Enable tv-alu-help-tv-dec-a-works; name this event ‘tv-alu-help-tv-dec-a-works-off’.

THEOREM: tv-alu-help-tv-dec-b-works

$$\begin{aligned}
& ((mpg = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})) \\
& \quad \wedge \text{bvp}(b) \\
& \quad \wedge (\text{length}(b) = \text{tree-size}(tree))) \\
& \rightarrow (\text{tv-alu-help}(c, a, b, mpg, tree) \\
& \quad = \text{tv-adder}(c, \text{v-not}(\text{nat-to-v}(0, \text{length}(b))), b, tree))
\end{aligned}$$

EVENT: Enable tv-alu-help-tv-dec-b-works; name this event ‘tv-alu-help-tv-dec-b-works-off’.

THEOREM: tv-alu-help-tv-neg-works

$$\begin{aligned}
& ((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})) \\
& \quad \wedge \text{bvp}(a) \\
& \quad \wedge (\text{length}(a) = \text{tree-size}(tree))) \\
& \rightarrow (\text{tv-alu-help}(c, a, b, mpg, tree) \\
& \quad = \text{tv-adder}(c, \text{v-not}(a), \text{nat-to-v}(0, \text{length}(a)), tree))
\end{aligned}$$

EVENT: Enable tv-alu-help-tv-neg-works; name this event ‘tv-alu-help-tv-neg-works-off’.

DEFINITION:

$$\begin{aligned}
& \text{f\$tv-alu-help}(c, a, b, mpg, tree) \\
& = \text{if } tree \simeq \mathbf{nil} \text{ then f\$alu-cell}(c, \text{car}(a), \text{car}(b), mpg) \\
& \quad \text{else cons}(\text{f-and}(\text{car}(\text{f\$tv-alu-help}(c, \\
& \qquad \qquad \qquad \text{tfirstn}(a, tree), \\
& \qquad \qquad \qquad \text{tfirstn}(b, tree), \\
& \qquad \qquad \qquad mpg, \\
& \qquad \qquad \qquad \text{car}(tree))), \\
& \qquad \text{car}(\text{f\$tv-alu-help}(\text{f\$t-carry}(c, \\
& \qquad \qquad \qquad \text{car}(\text{f\$tv-alu-help}(c, \\
& \qquad \qquad \qquad \text{tfirstn}(a, \\
& \qquad \qquad \qquad \text{tree}), \\
& \qquad \qquad \qquad \text{tfirstn}(b, \\
& \qquad \qquad \qquad \text{tree}), \\
& \qquad \qquad \qquad mpg, \\
& \qquad \qquad \qquad \text{car}(tree))), \\
& \qquad \text{cadr}(\text{f\$tv-alu-help}(c, \\
& \qquad \qquad \qquad \text{tfirstn}(a, \\
& \qquad \qquad \qquad \text{tree}), \\
& \qquad \qquad \qquad \text{tfirstn}(b, \\
& \qquad \qquad \qquad \text{tree}), \\
& \qquad \qquad \qquad mpg, \\
& \qquad \qquad \qquad \text{car}(tree))))),
\end{aligned}$$

```

trestn (a, tree),
trestn (b, tree),
mpg,
cdr (tree))))),
cons (fst-carry (cadr (fstv-alu-help (c,
tfirstn (a, tree),
tfirstn (b, tree),
mpg,
car (tree))),
car (fstv-alu-help (fst-carry (c,
car (fstv-alu-help (c,
tfirstn (a,
tree),
tfirstn (b,
tree),
mpg,
car (tree))),
cadr (fstv-alu-help (c,
tfirstn (a,
tree),
tfirstn (b,
tree),
mpg,
car (tree))))),
trestn (a, tree),
trestn (b, tree),
mpg,
cdr (tree))),
cadr (fstv-alu-help (fst-carry (c,
car (fstv-alu-help (c,
tfirstn (a,
tree),
tfirstn (b,
tree),
mpg,
car (tree))),
cadr (fstv-alu-help (c,
tfirstn (a,
tree),
tfirstn (b,
tree),
mpg,
car (tree))))),
trestn (a, tree),

```

```

                                trestn (b, tree),
                                mpg,
                                cdr (tree))),
append (cddr (f$tv-alu-help (c,
                                tfirstn (a, tree),
                                tfirstn (b, tree),
                                mpg,
                                car (tree))),
cddr (f$tv-alu-help (f$t-carry (c,
                                car (f$tv-alu-help (c,
                                                        tfirstn (a,
                                                            tree),
                                                        tfirstn (b,
                                                            tree),
                                                        mpg,
                                                        car (tree))),
                                cadr (f$tv-alu-help (c,
                                                        tfirstn (a,
                                                            tree),
                                                        tfirstn (b,
                                                            tree),
                                                        mpg,
                                                        car (tree))))),
                                trestn (a, tree),
                                trestn (b, tree),
                                mpg,
                                cdr (tree)))))) endif

```

EVENT: Enable f\$tv-alu-help; name this event 'f\$tv-alu-help-off'.

THEOREM: cddr-f\$tv-alu-help-length
 $\text{length}(\text{cddr}(\text{f\$tv-alu-help}(c, a, b, \text{mpg}, \text{tree}))) = \text{tree-size}(\text{tree})$

THEOREM: f\$tv-alu-help-length
 $\text{length}(\text{f\$tv-alu-help}(c, a, b, \text{mpg}, \text{tree})) = (1 + (1 + \text{tree-size}(\text{tree})))$

THEOREM: properp-cddr-f\$tv-alu-help
 $\text{properp}(\text{cddr}(\text{f\$tv-alu-help}(c, a, b, \text{mpg}, \text{tree})))$

THEOREM: properp-f\$tv-alu-help
 $\text{properp}(\text{f\$tv-alu-help}(c, a, b, \text{mpg}, \text{tree}))$

THEOREM: boolp-car-f\$tv-alu-help
 $\text{boolp}(c)$

```

^   bvp (a)
^   (length (a) = tree-size (tree))
^   bvp (b)
^   (length (b) = tree-size (tree))
^   bvp (mpg)
^   (length (mpg) = 7))
→   (boolp (car (f$tv-alu-help (c, a, b, mpg, tree))))
      ^   boolp (cadr (f$tv-alu-help (c, a, b, mpg, tree))))

```

THEOREM: bvp-cddr-f\$tv-alu-help

```

(boolp (c)
 ^   bvp (a)
 ^   (length (a) = tree-size (tree))
 ^   bvp (b)
 ^   (length (b) = tree-size (tree))
 ^   bvp (mpg)
 ^   (length (mpg) = 7))
→   bvp (cddr (f$tv-alu-help (c, a, b, mpg, tree))))

```

THEOREM: bvp-f\$tv-alu-help

```

(boolp (c)
 ^   bvp (a)
 ^   (length (a) = tree-size (tree))
 ^   bvp (b)
 ^   (length (b) = tree-size (tree))
 ^   bvp (mpg)
 ^   (length (mpg) = 7))
→   bvp (f$tv-alu-help (c, a, b, mpg, tree))

```

THEOREM: f\$tv-alu-help=tv-alu-help

```

(boolp (c)
 ^   bvp (a)
 ^   (length (a) = tree-size (tree))
 ^   bvp (b)
 ^   (length (b) = tree-size (tree))
 ^   bvp (mpg)
 ^   (length (mpg) = 7))
→   (f$tv-alu-help (c, a, b, mpg, tree) = tv-alu-help (c, a, b, mpg, tree))

```

(DEFN TV-ALU-HELP-BODY

(TREE)

(IF

(NLISTP TREE)

```

(LIST (LIST 'LEAF
          (LIST 'P 'G (INDEX 'OUT 0))
          'ALU-CELL
          (CONS 'C
                (CONS (INDEX 'A 0)
                      (CONS (INDEX 'B 0)
                            (INDICES 'MPG 0 7)))))))
(APPEND
 (IF (EQUAL (REMAINDER (SUB1 (TREE-HEIGHT TREE))
                      3)
          0)
  (LIST (LIST 'BUFFERMPG
            (INDICES 'MPG- 0 7)
            (INDEX 'V-BUF 7)
            (INDICES 'MPG 0 7)))
  NIL)
(CONS
 (LIST 'LHS
       (CONS 'PL
             (CONS 'GL
                   (TFIRSTN (INDICES 'OUT 0 (TREE-SIZE TREE))
                           TREE)))
       (INDEX 'TV-ALU-HELP
              (TREE-NUMBER (CAR TREE)))
       (CONS 'C
             (APPEND (TFIRSTN (INDICES 'A 0 (TREE-SIZE TREE))
                           TREE)
                     (APPEND (TFIRSTN (INDICES 'B 0 (TREE-SIZE TREE))
                           TREE)
                             (IF (EQUAL (REMAINDER (SUB1 (TREE-HEIGHT TREE))
                                              3)
                                  0)
                                 (INDICES 'MPG- 0 7)
                                 (INDICES 'MPG 0 7)))))))
(CONS
 ' (LHS-CARRY (CL) T-CARRY (C PL GL))
(CONS
 (LIST 'RHS
       (CONS 'PR
             (CONS 'GR
                   (TRESTDN (INDICES 'OUT 0 (TREE-SIZE TREE))
                           TREE)))
       (INDEX 'TV-ALU-HELP
              (TREE-NUMBER (CDR TREE)))

```



```

(CONS 'CL
  (APPEND (TRESTN (INDICES 'A 0 (TREE-SIZE TREE))
    TREE)
    (APPEND (TRESTN (INDICES 'B 0 (TREE-SIZE TREE))
      TREE)
      (IF (EQUAL (REMAINDER (SUB1 (TREE-HEIGHT TREE))
        3)
        0)
        (INDICES 'MPG- 0 7)
        (INDICES 'MPG 0 7))))))
'((P (P) B-AND (PL PR))
  (G (G) T-CARRY (GL PR GR))))))

```

DEFINITION:

```

tv-alu-help* (tree)
= cons (index ('tv-alu-help, tree-number (tree)),
  cons (cons ('c,
    append (indices ('a, 0, tree-size (tree)),
      append (indices ('b, 0, tree-size (tree)),
        indices ('mpg, 0, 7))),
    cons (cons ('p, cons ('g, indices ('out, 0, tree-size (tree))),
      cons (tv-alu-help-body (tree), ' (nil))))))

```

THEOREM: tv-alu-help*\$destructure

```

(car (tv-alu-help* (tree))) = index ('tv-alu-help, tree-number (tree))
^ (cadr (tv-alu-help* (tree)))
= cons ('c,
  append (indices ('a, 0, tree-size (tree)),
    append (indices ('b, 0, tree-size (tree)),
      indices ('mpg, 0, 7))))
^ (caddr (tv-alu-help* (tree)))
= cons ('p, cons ('g, indices ('out, 0, tree-size (tree))))
^ (caddr (tv-alu-help* (tree))) = tv-alu-help-body (tree)
^ (caddr (tv-alu-help* (tree))) = nil

```

EVENT: Enable tv-alu-help*; name this event 'tv-alu-help*-off'.

EVENT: Enable tv-alu-help*\$destructure; name this event 'tv-alu-help*\$destructure-off'.

DEFINITION:

```

tv-alu-help& (netlist, tree)
= if tree  $\simeq$  nil
  then (lookup-module (index ('tv-alu-help, tree-number (tree)), netlist)

```

```

= tv-alu-help* (tree))
 $\wedge$  alu-cell& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist))
else (lookup-module (index ('tv-alu-help, tree-number (tree)),
                             netlist))
= tv-alu-help* (tree))
 $\wedge$  t-carry& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist))
 $\wedge$  b-and& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist))
 $\wedge$  v-buf& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist),
7)
 $\wedge$  tv-alu-help& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist),
car (tree))
 $\wedge$  tv-alu-help& (delete-module (index ('tv-alu-help,
                                     tree-number (tree)),
                                     netlist),
cdr (tree)) endif

```

EVENT: Enable tv-alu-help&; name this event 'tv-alu-help&-off'.

DEFINITION:

```

tv-alu-help$netlist (tree)
= if tree  $\simeq$  nil then cons (tv-alu-help* (tree), ALU-CELL$NETLIST)
  else cons (tv-alu-help* (tree),
            tv-alu-help$netlist (car (tree))
             $\cup$  (tv-alu-help$netlist (cdr (tree))
             $\cup$  (T-CARRY$NETLIST
             $\cup$  (B-AND$NETLIST
             $\cup$  v-buf$netlist (7)))))) endif

```

DEFINITION:

```

tv-alu-help-induction (tree, c, a, b, mpg, state, netlist)
= if tree  $\simeq$  nil then t
  else tv-alu-help-induction (car (tree),
                              c,
                              tfirstn (a, tree),

```

```

tfirstn (b, tree),
if ((tree-height (tree) - 1) mod 3)
  = 0 then v-threefix (mpg)
else mpg endif,
0,
delete-module (index ('tv-alu-help,
                      tree-number (tree)),
                netlist))
 $\wedge$  tv-alu-help-induction (cdr (tree),
                           fst-carry (c,
                                       car (fstv-alu-help (c,
                                                             tfirstn (a,
                                                             tree),
                                                             tfirstn (b,
                                                             tree),
                                                             if ((tree-height (tree) - 1)
                                                             mod 3)
                                                             = 0
                                                             then v-threefix (mpg)
                                                             else mpg endif,
                                                             car (tree))),
                                       cadr (fstv-alu-help (c,
                                                             tfirstn (a,
                                                             tree),
                                                             tfirstn (b,
                                                             tree),
                                                             if ((tree-height (tree) - 1)
                                                             mod 3)
                                                             = 0
                                                             then v-threefix (mpg)
                                                             else mpg endif,
                                                             car (tree))))),
                           trestn (a, tree),
                           trestn (b, tree),
                           if ((tree-height (tree) - 1)
                               mod 3)
                               = 0
                           then v-threefix (mpg)
                           else mpg endif,
                           0,
                           delete-module (index ('tv-alu-help,
                                                  tree-number (tree)),
                                          netlist)) endif

```

THEOREM: tv-alu-help-lemma-crock

tv-alu-help& (delete-module (index ('**tv-alu-help**, tree-number (*tree*)), *netlist*),
 car (*tree*))
 → (collect-value (indices ('**out**, 0, *n*), *bindings*)
 = append (collect-value (firstn (tree-size (car (*tree*)),
 indices ('**out**, 0, *n*)),
 bindings),
 collect-value (restn (tree-size (car (*tree*)),
 indices ('**out**, 0, *n*)),
 bindings)))

EVENT: Enable tv-alu-help-lemma-crock; name this event 'tv-alu-help-lemma-crock-off'.

THEOREM: tv-alu-help\$value-base-case

((*tree* $\simeq$ **nil**)
 ∧ tv-alu-help& (*netlist*, *tree*)
 ∧ (length (*a*) = 1)
 ∧ properp (*a*)
 ∧ (length (*b*) = 1)
 ∧ properp (*b*)
 ∧ (length (*mpg*) = 7)
 ∧ properp (*mpg*))
 → (dual-eval (0,
 index ('**tv-alu-help**, tree-number (*tree*)),
 cons (*c*, append (*a*, append (*b*, *mpg*))),
 state,
 netlist)
 = f\$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))

EVENT: Enable tv-alu-help\$value-base-case; name this event 'tv-alu-help\$value-base-case-off'.

THEOREM: f\$tv-alu-help-v-threefix-mpg

(length (*mpg*) = 7)
 → (f\$tv-alu-help (*c*, *a*, *b*, v-threefix (*mpg*), *tree*)
 = f\$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))

THEOREM: tv-alu-help\$value

(tv-alu-help& (*netlist*, *tree*)
 ∧ properp (*a*)
 ∧ (length (*a*) = tree-size (*tree*))
 ∧ properp (*b*)
 ∧ (length (*b*) = tree-size (*tree*))

```

 $\wedge$  properp(mpg)
 $\wedge$  (length(mpg) = 7))
 $\rightarrow$  (dual-eval(0,
              index('tv-alu-help', tree-number(tree)),
              cons(c, append(a, append(b, mpg))),
              state,
              netlist)
      = f$tv-alu-help(c, a, b, mpg, tree))

```

EVENT: Enable tv-alu-help\$value; name this event 'tv-alu-help\$value-off'.

DEFINITION:

```

carry-out-help(a0, result, zero, op0, op1, op2, op3)
= b-and(b-nand3(b-nand4(b-not(op3),
                        b-nand(b-not(op0), b-not(op1)),
                        b-not(op2),
                        result),
        b-nand3(b-not(op3), b-not(b-not(op2)), b-not(result)),
        b-nand4(b-not(b-not(op3)),
                  b-not(op2),
                  b-nand(b-not(b-not(op0)), b-not(b-not(op1))),
                  a0)),
        b-not(zero))

```

DEFINITION:

```

f$carry-out-help(a0, result, zero, op0, op1, op2, op3)
= f-and(f-nand3(f-nand4(f-not(op3),
                        f-nand(f-not(op0), f-not(op1)),
                        f-not(op2),
                        result),
        f-nand3(f-not(op3), f-not(f-not(op2)), f-not(result)),
        f-nand4(f-not(f-not(op3)),
                  f-not(op2),
                  f-nand(f-not(f-not(op0)), f-not(f-not(op1))),
                  a0)),
        f-not(zero))

```

EVENT: Enable f\$carry-out-help; name this event 'f\$carry-out-help-off'.

DEFINITION:

```

CARRY-OUT-HELP*
= '(carry-out-help
   (a0 result zero op0 op1 op2 op3)
   (w-22)

```

```

((g-21 (w-21) b-not (zero))
 (g-16 (w-16) b-not (op1))
 (g-17 (w-17) b-not (w-16))
 (g-14 (w-14) b-not (op0))
 (g-15 (w-15) b-not (w-14))
 (g-18 (w-18) b-nand (w-15 w-17))
 (g-13 (w-13) b-not (op2))
 (g-11 (w-11) b-not (op3))
 (g-12 (w-12) b-not (w-11))
 (g-19 (w-19) b-nand4 (w-12 w-13 w-18 a0))
 (g-9 (w-9) b-not (result))
 (g-8 (w-8) b-not (w-13))
 (g-10 (w-10) b-nand3 (w-11 w-8 w-9))
 (g-3 (w-3) b-nand (w-14 w-16))
 (g-5 (w-5) b-nand4 (w-11 w-3 w-13 result))
 (g-20 (w-20) b-nand3 (w-5 w-10 w-19))
 (g-22 (w-22) b-and (w-20 w-21)))
nil)

```

DEFINITION:

```

carry-out-help& (netlist)
= ((lookup-module ('carry-out-help, netlist) = CARRY-OUT-HELP*)
   ∧ b-not& (delete-module ('carry-out-help, netlist))
   ∧ b-nand& (delete-module ('carry-out-help, netlist))
   ∧ b-nand4& (delete-module ('carry-out-help, netlist))
   ∧ b-nand3& (delete-module ('carry-out-help, netlist))
   ∧ b-and& (delete-module ('carry-out-help, netlist)))

```

EVENT: Enable carry-out-help&; name this event 'carry-out-help&-off'.

DEFINITION:

```

CARRY-OUT-HELP$NETLIST
= cons (CARRY-OUT-HELP*,
        (((B-NOT$NETLIST ∪ B-NAND$NETLIST) ∪ B-NAND4$NETLIST)
         ∪ B-NAND3$NETLIST)
         ∪ B-AND$NETLIST)

```

THEOREM: carry-out-help\$value

```

carry-out-help& (netlist)
→ (dual-eval (0,
               'carry-out-help,
               list (a0, result, zero, op0, op1, op2, op3),
               state,
               netlist)
   = list (f$carry-out-help (a0, result, zero, op0, op1, op2, op3)))

```

EVENT: Enable carry-out-help\$value; name this event 'carry-out-help\$value-off'.

THEOREM: f\$carry-out-help=carry-out-help
 (boolp (*a0*)
 $\wedge$ boolp (*result*)
 $\wedge$ boolp (*zero*)
 $\wedge$ boolp (*op0*)
 $\wedge$ boolp (*op1*)
 $\wedge$ boolp (*op2*)
 $\wedge$ boolp (*op3*))
 $\rightarrow$ (f\$carry-out-help (*a0*, *result*, *zero*, *op0*, *op1*, *op2*, *op3*)
 = carry-out-help (*a0*, *result*, *zero*, *op0*, *op1*, *op2*, *op3*))

THEOREM: carry-out-help-congruence
 $x \rightarrow ((\text{carry-out-help}(x, a, \text{zero}, \text{op0}, \text{op1}, \text{op2}, \text{op3})$
 = carry-out-help(**t**, *a*, *zero*, *op0*, *op1*, *op2*, *op3*))
 = **t**)

THEOREM: carry-out-help\$value-zero
 carry-out-help& (*netlist*)
 $\rightarrow$ (dual-eval (0,
 'carry-out-help,
 list (*a0*, *result*, **t**, *op0*, *op1*, *op2*, *op3*),
 state,
 netlist)
 = list (**f**))

EVENT: Enable carry-out-help\$value-zero; name this event 'carry-out-help\$value-zero-off'.

DEFINITION:
 overflow-help (*rn*, *an*, *bn*, *zero*, *op0*, *op1*, *op2*, *op3*)
 = b-if (*rn*,
 b-nor (b-nand (b-nor (b-nand3 (b-not (*op3*),
 b-or3 (b-not (*op1*),
 b-not (*op2*),
 b-xor (b-not (b-not (*an*)), *bn*)),
 b-nand3 (b-not (*op1*,
 b-not (b-not (*op2*)),
 b-not (*an*))),
 b-nand (b-nand3 (*op1*,
 b-not (*op2*),
 b-xor (b-not (b-not (*an*)), *bn*)),

```

                                b-nand3 (b-not (op1),
                                        b-not (op2),
                                        b-not (b-not (an))))) ,
                                b-not (zero)),
    b-nand3 (b-nand (b-not (b-not (op2)), b-not (an)),
            b-nand3 (op0, b-not (op1), b-not (b-not (an))),
            b-nand (b-not (op2), b-not (b-not (an))))),
    b-nor (b-nand (b-nor (b-nand3 (b-not (op3),
                                b-or3 (b-not (op1),
                                        b-not (op2),
                                        b-xor (b-not (b-not (an)), bn)),
                                b-nand3 (b-not (op1),
                                        b-not (b-not (op2)),
                                        b-not (an))),
            b-nand (b-nand3 (op1,
                                b-not (op2),
                                b-xor (b-not (b-not (an)), bn)),
            b-nand3 (b-not (op1),
                    b-not (op2),
                    b-not (b-not (an))))),
                                b-not (zero)),
    b-not (b-nand3 (b-nand (b-not (b-not (op2)), b-not (an)),
                    b-nand3 (op0, b-not (op1), b-not (b-not (an))),
                    b-nand (b-not (op2), b-not (b-not (an))))))

```

DEFINITION:

f\$overflow-help (*rn*, *an*, *bn*, *zero*, *op0*, *op1*, *op2*, *op3*)

```

=   f-if (rn,
        f-nor (f-nand (f-nor (f-nand3 (f-not (op3),
                                f-or3 (f-not (op1),
                                        f-not (op2),
                                        f-xor (f-not (f-not (an)), bn)),
                                f-nand3 (f-not (op1),
                                        f-not (f-not (op2)),
                                        f-not (an))),
            f-nand (f-nand3 (op1,
                                f-not (op2),
                                f-xor (f-not (f-not (an)), bn)),
            f-nand3 (f-not (op1),
                    f-not (op2),
                    f-not (f-not (an))))),
        f-not (zero)),
    f-nand3 (f-nand (f-not (f-not (op2)), f-not (an)),
            f-nand3 (op0, f-not (op1), f-not (f-not (an))))

```



```

        f-nand (f-not (op2), f-not (f-not (an))))) ,
f-nor (f-nand (f-nor (f-nand3 (f-not (op3),
        f-or3 (f-not (op1),
        f-not (op2),
        f-xor (f-not (f-not (an)), bn)),
        f-nand3 (f-not (op1),
        f-not (f-not (op2)),
        f-not (an))),
        f-nand (f-nand3 (op1,
        f-not (op2),
        f-xor (f-not (f-not (an)), bn)),
        f-nand3 (f-not (op1),
        f-not (op2),
        f-not (f-not (an))))) ,
        f-not (zero)),
f-not (f-nand3 (f-nand (f-not (f-not (op2)), f-not (an)),
        f-nand3 (op0, f-not (op1), f-not (f-not (an))),
        f-nand (f-not (op2), f-not (f-not (an))))) )

```

EVENT: Enable f\$overflow-help; name this event 'f\$overflow-help-off'.

DEFINITION:

OVERFLOW-HELP*

```

=  '(overflow-help
    (rn an bn zero op0 op1 op2 op3)
    (w-83)
    ((g-77 (w-77) b-not (an))
     (g-78 (w-78) b-not (w-77))
     (g-76 (w-76) b-not (op2))
     (g-79 (w-79) b-nand (w-76 w-78))
     (g-72 (w-72) b-not (op1))
     (g-75 (w-75) b-nand3 (op0 w-72 w-78))
     (g-69 (w-69) b-not (w-76))
     (g-71 (w-71) b-nand (w-69 w-77))
     (g-80 (w-80) b-nand3 (w-71 w-75 w-79))
     (g-81 (w-81) b-not (w-80))
     (g-66 (w-66) b-not (zero))
     (g-63 (w-63) b-nand3 (w-72 w-76 w-78))
     (g-57 (w-57) b-xor (w-78 bn))
     (g-58 (w-58) b-nand3 (op1 w-76 w-57))
     (g-64 (w-64) b-nand (w-58 w-63))
     (g-52 (w-52) b-nand3 (w-72 w-69 w-77))
     (g-47 (w-47) b-or3 (w-72 w-76 w-57))
     (g-41 (w-41) b-not (op3))

```

```

(g-53 (w-53) b-nand3 (w-41 w-47 w-52))
(g-65 (w-65) b-nor (w-53 w-64))
(g-67 (w-67) b-nand (w-65 w-66))
(g-82 (w-82) b-nor (w-67 w-81))
(g-40 (w-40) b-nor (w-67 w-80))
(g-83 (w-83) b-if (rn w-40 w-82)))
nil)

```

DEFINITION:

```

overflow-help& (netlist)
= ((lookup-module ('overflow-help, netlist) = OVERFLOW-HELP*)
   ∧ b-xor& (delete-module ('overflow-help, netlist))
   ∧ b-or3& (delete-module ('overflow-help, netlist))
   ∧ b-not& (delete-module ('overflow-help, netlist))
   ∧ b-nand3& (delete-module ('overflow-help, netlist))
   ∧ b-nand& (delete-module ('overflow-help, netlist))
   ∧ b-nor& (delete-module ('overflow-help, netlist))
   ∧ b-if& (delete-module ('overflow-help, netlist)))

```

EVENT: Enable overflow-help&; name this event 'overflow-help&-off'.

DEFINITION:

```

OVERFLOW-HELP$NETLIST
= cons (OVERFLOW-HELP*,
        (((((B-XOR$NETLIST ∪ B-OR3$NETLIST) ∪ B-NOT$NETLIST)
            ∪ B-NAND3$NETLIST)
          ∪ B-NAND$NETLIST)
        ∪ B-NOR$NETLIST)
        ∪ B-IF$NETLIST)

```

THEOREM: overflow-help\$value

```

overflow-help& (netlist)
→ (dual-eval (0,
              'overflow-help,
              list (rn, an, bn, zero, op0, op1, op2, op3),
              state,
              netlist)
   = list (f$overflow-help (rn, an, bn, zero, op0, op1, op2, op3)))

```

EVENT: Enable overflow-help\$value; name this event 'overflow-help\$value-off'.

THEOREM: f\$overflow-help=overflow-help

```

(boolp (rn)
 ∧ boolp (an)

```

$$\begin{aligned}
& \wedge \text{ boolp}(bn) \\
& \wedge \text{ boolp}(zero) \\
& \wedge \text{ boolp}(op0) \\
& \wedge \text{ boolp}(op1) \\
& \wedge \text{ boolp}(op2) \\
& \wedge \text{ boolp}(op3) \\
\rightarrow & (\text{f}\$overflow\text{-help}(rn, an, bn, zero, op0, op1, op2, op3) \\
& = \text{ overflow-help}(rn, an, bn, zero, op0, op1, op2, op3))
\end{aligned}$$

THEOREM: overflow-help\$value-zero
(overflow-help& (*netlist*) $\wedge$ boolp(*rn*))
 $\rightarrow$ (dual-eval(0,
 'overflow-help,
 list(*rn*, *an*, *bn*, **t**, *op0*, *op1*, *op2*, *op3*),
 state,
 netlist)
 = list(**f**))

EVENT: Enable overflow-help\$value-zero; name this event 'overflow-help\$value-zero-off'.

DEFINITION:
v-shift-right-names(*a*, *si*)
= **if** *a* $\simeq$ **nil** **then nil**
 else append(cdr(*a*), list(*si*)) **endif**

THEOREM: length-v-shift-right-names
length(v-shift-right-names(*a*, *si*)) = length(*a*)

THEOREM: properp-shift-right-names
properp(v-shift-right-names(*a*, *si*))

THEOREM: collect-value-v-shift-right
v-threefix(collect-value(v-shift-right-names(*a*, *si*), *alist*))
= fv-shift-right(collect-value(*a*, *alist*), value(*si*, *alist*))

DEFINITION:
shift-or-buf-ctrl(*c*, *an*, *zero*, *op0*, *op1*, *op2*, *op3*)
= list(b-or3(b-and(*op0*, *op1*), b-nand(b-not(*op2*), *op3*), *zero*),
 b-or(b-and(*op0*, *an*), b-and(b-and(b-not(*op0*), b-not(*op1*)), *c*)))

DEFINITION:
f\$shift-or-buf-ctrl(*c*, *an*, *zero*, *op0*, *op1*, *op2*, *op3*)
= list(f-or3(f-and(*op0*, *op1*), f-nand(f-not(*op2*), *op3*), *zero*),
 f-or(f-and(*op0*, *an*), f-and(f-and(f-not(*op0*), f-not(*op1*)), *c*)))

EVENT: Enable f\$shift-or-buf-cntl; name this event 'f\$shift-or-buf-cntl-off'.

DEFINITION:

SHIFT-OR-BUF-CNTL*

```
= ' (shift-or-buf-cntl
    (c an zero op0 op1 op2 op3)
    (w-3 w-9)
    ((g-6 (w-6) b-not (op1))
     (g-5 (w-5) b-not (op0))
     (g-7 (w-7) b-and (w-5 w-6))
     (g-8 (w-8) b-and (w-7 c))
     (g-4 (w-4) b-and (op0 an))
     (g-9 (w-9) b-or (w-4 w-8))
     (g-1 (w-1) b-not (op2))
     (g-2 (w-2) b-nand (w-1 op3))
     (g-0 (w-0) b-and (op0 op1))
     (g-3 (w-3) b-or3 (w-0 w-2 zero)))
    nil)
```

DEFINITION:

shift-or-buf-cntl& (*netlist*)

```
= ((lookup-module ('shift-or-buf-cntl, netlist)
   =  SHIFT-OR-BUF-CNTL*)
   ^ b-or& (delete-module ('shift-or-buf-cntl, netlist))
   ^ b-not& (delete-module ('shift-or-buf-cntl, netlist))
   ^ b-nand& (delete-module ('shift-or-buf-cntl, netlist))
   ^ b-and& (delete-module ('shift-or-buf-cntl, netlist))
   ^ b-or3& (delete-module ('shift-or-buf-cntl, netlist)))
```

EVENT: Enable shift-or-buf-cntl&; name this event 'shift-or-buf-cntl&-off'.

DEFINITION:

SHIFT-OR-BUF-CNTL\$NETLIST

```
= cons (SHIFT-OR-BUF-CNTL*,
       (((B-OR$NETLIST ∪ B-NOT$NETLIST) ∪ B-NAND$NETLIST)
        ∪ B-AND$NETLIST)
        ∪ B-OR3$NETLIST)
```

THEOREM: shift-or-buf-cntl\$value

shift-or-buf-cntl& (*netlist*)

```
→ (dual-eval (0,
              'shift-or-buf-cntl,
              list (c, an, zero, op0, op1, op2, op3),
              state,
```

$$\begin{aligned}
& \text{netlist}) \\
= & \text{f\$shift-or-buf-cntl}(c, an, zero, op0, op1, op2, op3))
\end{aligned}$$

EVENT: Enable shift-or-buf-cntl\$value; name this event ‘shift-or-buf-cntl\$value-off’.

THEOREM: f\$shift-or-buf-cntl=shift-or-buf-cntl
 (boolp (c)
 $\wedge$ boolp (an)
 $\wedge$ boolp (zero)
 $\wedge$ boolp (op0)
 $\wedge$ boolp (op1)
 $\wedge$ boolp (op2)
 $\wedge$ boolp (op3))
 $\rightarrow$ (f\$shift-or-buf-cntl (c, an, zero, op0, op1, op2, op3)
 = shift-or-buf-cntl (c, an, zero, op0, op1, op2, op3))

THEOREM: shift-or-buf-cntl\$value-zero
 shift-or-buf-cntl& (netlist)
 $\rightarrow$ (car (dual-eval (0,
 ’shift-or-buf-cntl,
 list (c, an, t, op0, op1, op2, op3),
 state,
 netlist))
 = t)

DEFINITION:
 f\$shift-or-buf (c, a, an, zero, op0, op1, op2, op3)
 = fv-if (car (f\$shift-or-buf-cntl (c, an, zero, op0, op1, op2, op3)),
 a,
 fv-shift-right (a,
 cadr (f\$shift-or-buf-cntl (c,
 an,
 zero,
 op0,
 op1,
 op2,
 op3))))))

EVENT: Enable f\$shift-or-buf; name this event ‘f\$shift-or-buf-off’.

DEFINITION:
 shift-or-buf (c, a, an, zero, op0, op1, op2, op3)
 = v-if (car (shift-or-buf-cntl (c, an, zero, op0, op1, op2, op3)),

$a,$
 $\text{v-shift-right}(a,$
 $\quad \text{cadr}(\text{shift-or-buf-ctrl}(c, an, zero, op0, op1, op2, op3))))$

EVENT: Enable shift-or-buf; name this event ‘shift-or-buf-off’.

THEOREM: properp-length-f\$shift-or-buf
 $\text{properp}(\text{f\$shift-or-buf}(c, a, an, zero, op0, op1, op2, op3))$
 $\wedge (\text{length}(\text{f\$shift-or-buf}(c, a, an, zero, op0, op1, op2, op3)) = \text{length}(a))$

THEOREM: length-shift-or-buf
 $\text{length}(\text{shift-or-buf}(c, a, an, zero, op1, op2, op3, op4)) = \text{length}(a)$

THEOREM: bvp-shift-or-buf
 $\text{bvp}(\text{shift-or-buf}(c, a, an, zero, op1, op2, op3, op4))$

THEOREM: f\$shift-or-buf=shift-or-buf
 $(\text{boolp}(c)$
 $\wedge \text{boolp}(an)$
 $\wedge \text{boolp}(zero)$
 $\wedge \text{boolp}(op1)$
 $\wedge \text{boolp}(op2)$
 $\wedge \text{boolp}(op3)$
 $\wedge \text{boolp}(op4)$
 $\wedge \text{bvp}(a))$
 $\rightarrow (\text{f\$shift-or-buf}(c, a, an, zero, op1, op2, op3, op4)$
 $\quad = \text{shift-or-buf}(c, a, an, zero, op1, op2, op3, op4))$

THEOREM: shift-or-buf-is-buf
 $(\text{bvp}(a) \wedge ((op0 \wedge op1) \vee op2 \vee (\neg op3) \vee zero))$
 $\rightarrow (\text{shift-or-buf}(c, a, an, zero, op0, op1, op2, op3) = a)$

THEOREM: shift-or-buf-is-asr
 $(\text{bvp}(a)$
 $\wedge (an = \text{nth}(\text{length}(a) - 1, a))$
 $\wedge op0$
 $\wedge (\neg op1)$
 $\wedge (\neg op2)$
 $\wedge op3$
 $\wedge (\neg zero))$
 $\rightarrow (\text{shift-or-buf}(c, a, an, zero, op0, op1, op2, op3) = \text{v-asr}(a))$

THEOREM: shift-or-buf-is-ror
 $(\text{bvp}(a)$
 $\wedge \text{boolp}(c)$

$$\begin{aligned}
& \wedge (\neg op0) \\
& \wedge (\neg op1) \\
& \wedge (\neg op2) \\
& \wedge op3 \\
& \wedge (\neg zero) \\
\rightarrow & \text{shift-or-buf}(c, a, an, zero, op0, op1, op2, op3) = \text{v-ror}(a, c)
\end{aligned}$$

THEOREM: shift-or-buf-is-lsr

$$\begin{aligned}
& (\text{bvp}(a) \wedge (\neg op0) \wedge op1 \wedge (\neg op2) \wedge op3 \wedge (\neg zero)) \\
\rightarrow & \text{shift-or-buf}(c, a, an, zero, op0, op1, op2, op3) = \text{v-lsr}(a)
\end{aligned}$$

DEFINITION:

tv-shift-or-buf* (*tree*)

$$\begin{aligned}
= & \text{cons}(\text{index}('tv\text{-}shift\text{-}or\text{-}buf, \text{tree-number}(tree)), \\
& \text{cons}(\text{cons}('c, \\
& \quad \text{append}(\text{indices}('a, 0, \text{tree-size}(tree)), \\
& \quad \quad '(an\ zero\ op0\ op1\ op2\ op3))), \\
& \text{cons}(\text{indices}('out, 0, \text{tree-size}(tree)), \\
& \quad \text{cons}(\text{list}('cntl \\
& \quad \quad \text{cntl\ si}) \\
& \quad \quad \text{shift-or-buf-cntl} \\
& \quad \quad (c\ an\ zero\ op0\ op1\ op2\ op3)), \\
& \text{list}('mux, \\
& \quad \text{indices}('out, 0, \text{tree-size}(tree)), \\
& \quad \text{index}('tv\text{-}if, \text{tree-number}(tree)), \\
& \quad \text{cons}('cntl, \\
& \quad \quad \text{append}(\text{indices}('a, \\
& \quad \quad \quad 0, \\
& \quad \quad \quad \text{tree-size}(tree)), \\
& \quad \quad \text{v-shift-right-names}(\text{indices}('a, \\
& \quad \quad \quad 0, \\
& \quad \quad \quad \text{tree-size}(tree)), \\
& \quad \quad 'si))))), \\
& 'nil))))))
\end{aligned}$$

THEOREM: tv-shift-or-buf*\$destructure

$$\begin{aligned}
& (\text{car}(\text{tv-shift-or-buf}^*(tree)) \\
& = \text{index}('tv\text{-}shift\text{-}or\text{-}buf, \text{tree-number}(tree))) \\
\wedge & (\text{cadr}(\text{tv-shift-or-buf}^*(tree)) \\
& = \text{cons}('c, \\
& \quad \text{append}(\text{indices}('a, 0, \text{tree-size}(tree)), \\
& \quad \quad '(an\ zero\ op0\ op1\ op2\ op3)))) \\
\wedge & (\text{caddr}(\text{tv-shift-or-buf}^*(tree)) = \text{indices}('out, 0, \text{tree-size}(tree))) \\
\wedge & (\text{cadddr}(\text{tv-shift-or-buf}^*(tree)) \\
& = \text{list}('cntl
\end{aligned}$$

```

      (ctl si)
      shift-or-buf-ctl
      (c an zero op0 op1 op2 op3)),
list ('mux,
      indices ('out, 0, tree-size (tree)),
      index ('tv-if, tree-number (tree)),
      cons ('ctl,
            append (indices ('a, 0, tree-size (tree)),
                    v-shift-right-names (indices ('a,
                                                    0,
                                                    tree-size (tree)),
                                                    'si))))))
 $\wedge$  (caddr (tv-shift-or-buf* (tree)) = nil)

```

EVENT: Enable tv-shift-or-buf*; name this event 'tv-shift-or-buf*-off'.

EVENT: Enable tv-shift-or-buf*\$destructure; name this event 'tv-shift-or-buf*\$destructure-off'.

DEFINITION:

```

tv-shift-or-buf& (netlist, tree)
= ((lookup-module (index ('tv-shift-or-buf, tree-number (tree)),
                          netlist)
   = tv-shift-or-buf* (tree))
 $\wedge$  shift-or-buf-ctl& (delete-module (index ('tv-shift-or-buf,
                                              tree-number (tree)),
                                              netlist))
 $\wedge$  tv-if& (delete-module (index ('tv-shift-or-buf,
                                tree-number (tree)),
                                netlist),
            tree))

```

EVENT: Enable tv-shift-or-buf&; name this event 'tv-shift-or-buf&-off'.

DEFINITION:

```

tv-shift-or-buf$netlist (tree)
= cons (tv-shift-or-buf* (tree),
        tv-if$netlist (tree)  $\cup$  SHIFT-OR-BUF-CNTL$NETLIST)

```

THEOREM: tv-shift-or-buf\$value

```

(tv-shift-or-buf& (netlist, tree)
 $\wedge$  (length (a) = tree-size (tree))
 $\wedge$  properp (a))

```


$\rightarrow$ (dual-eval (0,
 index ('**tv-shift-or-buf**, tree-number (*tree*)),
 cons (*c*, append (*a*, list (*an*, *zero*, *op0*, *op1*, *op2*, *op3*))),
 state,
 netlist)
 = f\$shift-or-buf (*c*, *a*, *an*, *zero*, *op0*, *op1*, *op2*, *op3*))

EVENT: Enable tv-shift-or-buf\$value; name this event 'tv-shift-or-buf\$value-off'.

THEOREM: tv-shift-or-buf\$value-zero
 (tv-shift-or-buf& (*netlist*, *tree*)
 ^ (length (*a*) = tree-size (*tree*))
 ^ bvp (*a*)
 $\rightarrow$ (dual-eval (0,
 index ('**tv-shift-or-buf**, tree-number (*tree*)),
 cons (*c*, append (*a*, list (*an*, **t**, *op0*, *op1*, *op2*, *op3*))),
 state,
 netlist)
 = *a*)

EVENT: Enable tv-shift-or-buf\$value-zero; name this event 'tv-shift-or-buf\$value-zero-off'.

DEFINITION:
 f\$core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*)
 = cons (f\$carry-out-help (nth (0, *a*),
 f\$t-carry (*c*,
 car (f\$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)),
 cadr (f\$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))),
 zero,
 car (*op*),
 cadr (*op*),
 caddr (*op*),
 cadddr (*op*)),
 cons (f\$overflow-help (nth (length (*a*) - 1,
 caddr (f\$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))),
 nth (length (*a*) - 1, *a*),
 nth (length (*a*) - 1, *b*),
 zero,
 car (*op*),
 cadr (*op*),
 caddr (*op*),

```

                                caddr (op)),
cons (f$fast-zero (f$shift-or-buf (c,
                                cddr (f$tv-alu-help (c,
                                                        a,
                                                        b,
                                                        mpg,
                                                        tree))),
                                nth (length (a) - 1, a),
                                zero,
                                car (op),
                                cadr (op),
                                caddr (op),
                                caddr (op))),
f$shift-or-buf (c,
                cddr (f$tv-alu-help (c, a, b, mpg, tree)),
                nth (length (a) - 1, a),
                zero,
                car (op),
                cadr (op),
                caddr (op),
                caddr (op))))

```

EVENT: Enable f\$core-alu; name this event ‘f\$core-alu-off’.

DEFINITION:

```

core-alu (c, a, b, zero, mpg, op, tree)
=  cons (carry-out-help (nth (0, a),
                            t-carry (c,
                                    car (tv-alu-help (c, a, b, mpg, tree)),
                                    cadr (tv-alu-help (c, a, b, mpg, tree))),
                            zero,
                            car (op),
                            cadr (op),
                            caddr (op),
                            caddr (op)),
cons (overflow-help (nth (length (a) - 1,
                        cddr (tv-alu-help (c, a, b, mpg, tree))),
                    nth (length (a) - 1, a),
                    nth (length (a) - 1, b),
                    zero,
                    car (op),
                    cadr (op),
                    caddr (op),
                    caddr (op)),

```

```

cons (v-zero (shift-or-buf (c,
                             cddr (tv-alu-help (c, a, b, mpg, tree)),
                             nth (length (a) - 1, a),
                             zero,
                             car (op),
                             cadr (op),
                             caddr (op),
                             caddr (op))),
      shift-or-buf (c,
                    cddr (tv-alu-help (c, a, b, mpg, tree)),
                    nth (length (a) - 1, a),
                    zero,
                    car (op),
                    cadr (op),
                    caddr (op),
                    caddr (op))))

```

EVENT: Enable core-alu; name this event ‘core-alu-off’.

THEOREM: properp-length-f\$core-alu

```

properp (f$core-alu (c, a, b, zero, mpg, op, tree))
 $\wedge$  (length (f$core-alu (c, a, b, zero, mpg, op, tree))
      = (1 + (1 + (1 + tree-size (tree)))))

```

THEOREM: f\$core-alu=core-alu

```

(boolp (c)
 $\wedge$  bvp (a)
 $\wedge$  (length (a) = tree-size (tree))
 $\wedge$  bvp (b)
 $\wedge$  (length (b) = tree-size (tree))
 $\wedge$  if length (a) < 3 then f
    else t endif
 $\wedge$  boolp (zero)
 $\wedge$  bvp (mpg)
 $\wedge$  (length (mpg) = 7)
 $\wedge$  bvp (op)
 $\wedge$  (length (op) = 4))
 $\rightarrow$  (f$core-alu (c, a, b, zero, mpg, op, tree)
      = core-alu (c, a, b, zero, mpg, op, tree))

```

THEOREM: length-core-alu

```

length (core-alu (c, a, b, zero, mpg, op, tree))
= (1 + (1 + (1 + tree-size (tree))))

```

THEOREM: boolp-bvp-core-alu
 $\text{boolp}(c(\text{core-alu}(c, a, b, \text{zero}, \text{mpg}, \text{op}, \text{tree})))$
 $\wedge \text{boolp}(v(\text{core-alu}(c, a, b, \text{zero}, \text{mpg}, \text{op}, \text{tree})))$
 $\wedge \text{boolp}(zb(\text{core-alu}(c, a, b, \text{zero}, \text{mpg}, \text{op}, \text{tree})))$
 $\wedge \text{bvp}(\text{bv}(\text{core-alu}(c, a, b, \text{zero}, \text{mpg}, \text{op}, \text{tree})))$
 $\wedge \text{bvp}(\text{core-alu}(c, a, b, \text{zero}, \text{mpg}, \text{op}, \text{tree}))$

THEOREM: core-alu-works-for-all-normal-cases\$crock
 $(\text{bvp}(a) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{nth}(0, a))$
 $\rightarrow (\text{nth}(0, a) = \mathbf{t})$

EVENT: Enable core-alu-works-for-all-normal-cases\$crock; name this event ‘core-alu-works-for-all-normal-cases\$crock-off’.

THEOREM: core-alu-works-for-all-normal-cases
 $((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c))$
 $\rightarrow (v\text{-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})))$
 $= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}),$
 $\quad a,$
 $\quad b,$
 $\quad \mathbf{f},$
 $\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})),$
 $\quad \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}),$
 $\quad \text{tree})))$
 $\wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c))$
 $\rightarrow (v\text{-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})))$
 $= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}),$
 $\quad a,$
 $\quad b,$
 $\quad \mathbf{f},$
 $\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})),$
 $\quad \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}),$
 $\quad \text{tree})))$
 $\wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c))$
 $\rightarrow (v\text{-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})))$
 $= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}),$
 $\quad a,$
 $\quad b,$
 $\quad \mathbf{f},$
 $\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})),$
 $\quad \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}),$
 $\quad \text{tree})))$
 $\wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c))$
 $\rightarrow (v\text{-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})))$

$$\begin{aligned}
&= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})), \\
&\quad a, \\
&\quad b, \\
&\quad \mathbf{f}, \\
&\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})), \\
&\quad \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}), \\
&\quad \text{tree})) \\
\wedge \quad &((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
&\rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))) \\
&= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f})), \\
&\quad a, \\
&\quad b, \\
&\quad \mathbf{f}, \\
&\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f})), \\
&\quad \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}), \\
&\quad \text{tree})) \\
\wedge \quad &((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
&\rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))) \\
&= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})), \\
&\quad a, \\
&\quad b, \\
&\quad \mathbf{f}, \\
&\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})), \\
&\quad \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}), \\
&\quad \text{tree})) \\
\wedge \quad &((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
&\rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}))) \\
&= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})), \\
&\quad a, \\
&\quad b, \\
&\quad \mathbf{f}, \\
&\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})), \\
&\quad \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}), \\
&\quad \text{tree})) \\
\wedge \quad &((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
&\rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}))) \\
&= \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})), \\
&\quad a, \\
&\quad b, \\
&\quad \mathbf{f}, \\
&\quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})), \\
&\quad \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}), \\
&\quad \text{tree})) \\
\wedge \quad &((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c))
\end{aligned}$$

$$\begin{aligned}
& \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}), \\
& \quad \quad \text{tree})) \\
\wedge & ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
& \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}), \\
& \quad \quad \text{tree})) \\
\wedge & ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
& \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}), \\
& \quad \quad \text{tree})) \\
\wedge & ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
& \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}), \\
& \quad \quad \text{tree})) \\
\wedge & ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(\text{tree})) \wedge \text{boolp}(c)) \\
& \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}), \\
& \quad \quad \text{tree}))
\end{aligned}$$

$$\begin{aligned}
& \wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(tree)) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}), \\
& \quad \quad tree))) \\
& \wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(tree)) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}), \\
& \quad \quad tree))) \\
& \wedge ((\text{bv2p}(a, b) \wedge (\text{length}(a) = \text{tree-size}(tree)) \wedge \text{boolp}(c)) \\
& \quad \rightarrow (\text{v-alu}(c, a, b, \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}))) \\
& \quad = \text{core-alu}(\text{carry-in-help}(\text{list}(c, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad a, \\
& \quad \quad b, \\
& \quad \quad \mathbf{f}, \\
& \quad \quad \text{mpg}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})), \\
& \quad \quad \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}), \\
& \quad \quad tree)))
\end{aligned}$$

EVENT: Enable core-alu-works-for-all-normal-cases; name this event ‘core-alu-works-for-all-normal-cases-off’.

THEOREM: cases-on-a-4-bit-bvp

$$\begin{aligned}
& (\text{bvp}(op) \\
& \quad \wedge (\text{length}(op) = 4) \\
& \quad \wedge \text{if } op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}) \text{ then } p \\
& \quad \quad \text{elseif } op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}) \text{ then } p
\end{aligned}$$

```

elseif op = list (f, t, f, t) then p
elseif op = list (t, t, f, t) then p
elseif op = list (f, f, t, t) then p
elseif op = list (t, f, t, t) then p
elseif op = list (f, t, t, t) then p
elseif op = list (t, t, t, t) then p
else t endif)

```

→ p

THEOREM: core-alu-is-v-alu

```

(bv2p (a, b)
  ∧ (length (a) = tree-size (tree))
  ∧ bvp (op)
  ∧ (length (op) = 4)
  ∧ (¬ zero)
  ∧ (mpg = mpg (cons (zero, cons (f, op))))
  ∧ boolp (c))
→ (core-alu (carry-in-help (cons (c, cons (f, op))), a, b, zero, mpg, op, tree)
   = v-alu (c, a, b, op))

```

THEOREM: core-alu-works-for-zero-case

```

((length (a) = tree-size (tree)) ∧ zero)
→ (core-alu (t, a, b, zero, list (f, f, f, f, f, f, t), op, tree)
   = cvzbv (f, f, make-list (length (a), f)))

```

THEOREM: core-alu-works-as-inc-b

```

(bv2p (a, b) ∧ (length (a) = tree-size (tree)) ∧ (¬ zero) ∧ swap)
→ (bv (core-alu (carry-in-help (cons (c, cons (zero, ALU-INC-OP))),
  a,
  b,
  zero,
  mpg (cons (zero, cons (swap, ALU-INC-OP))),
  ALU-INC-OP,
  tree))
   = v-inc (b))

```

THEOREM: core-alu-works-as-dec-b

```

(bv2p (a, b) ∧ (length (a) = tree-size (tree)) ∧ (¬ zero) ∧ swap)
→ (bv (core-alu (carry-in-help (cons (c, cons (zero, ALU-DEC-OP))),
  a,
  b,
  zero,
  mpg (cons (zero, cons (swap, ALU-DEC-OP))),
  ALU-DEC-OP,
  tree))
   = v-dec (b))

```


DEFINITION:

```

core-alu* (tree)
=  cons (index ('core-alu, tree-number (tree)),
      cons (cons ('c,
        append (indices ('a, 0, tree-size (tree)),
          append (indices ('b, 0, tree-size (tree)),
            cons ('zero,
              append (indices ('mpg, 0, 7),
                indices ('op, 0, 4)))))),
        cons (cons ('carry,
          cons ('overflow,
            cons ('zerop,
              indices ('out, 0, tree-size (tree))))),
          cons (list (list ('m-alu,
            cons ('p,
              cons ('g,
                indices ('alu-out,
                  0,
                    tree-size (tree))))),
                index ('tv-alu-help,
                  tree-number (tree)),
                cons ('c,
                  append (indices ('a,
                    0,
                      tree-size (tree)),
                    append (indices ('b,
                      0,
                        tree-size (tree)),
                      indices ('mpg, 0, 7))))),
                ' (m-alu-carry
                  (alu-carry
                    t-carry
                    (c p g)),
                list ('m-carry-out-help,
                  ' (carry),
                  ' carry-out-help,
                  list (index ('a, 0),
                    'alu-carry,
                    'zero,
                    index ('op, 0),
                    index ('op, 1),
                    index ('op, 2),
                    index ('op, 3))),
                list ('m-overflow-help,

```

```

'overflow),
'overflow-help,
list(index('alu-out,
          tree-size(tree) - 1),
      index('a, tree-size(tree) - 1),
      index('b, tree-size(tree) - 1),
      'zero,
      index('op, 0),
      index('op, 1),
      index('op, 2),
      index('op, 3))),
list('m-shift,
      indices('out, 0, tree-size(tree)),
      index('tv-shift-or-buf,
            tree-number(tree)),
      cons('c,
            append(indices('alu-out,
                           0,
                           tree-size(tree)),
                  list(index('a,
                              tree-size(tree) - 1),
                        'zero,
                        index('op, 0),
                        index('op, 1),
                        index('op, 2),
                        index('op, 3)))))),
list('m-zerop,
      '(zerop),
      index('fast-zero, tree-size(tree)),
      indices('out, 0, tree-size(tree))),
'(nil))))))

```

THEOREM: core-alu*\$destructure

```

(car (core-alu*(tree)) = index('core-alu, tree-number(tree)))
^  (cadr (core-alu*(tree))
    =  cons('c,
            append(indices('a, 0, tree-size(tree)),
                  append(indices('b, 0, tree-size(tree)),
                        cons('zero,
                            append(indices('mpg, 0, 7),
                                indices('op, 0, 4)))))))
^  (caddr (core-alu*(tree))
    =  cons('carry,
            cons('overflow,

```

```

      cons('zerop, indices('out, 0, tree-size(tree))))))
^ (caddr (core-alu*(tree))
  = list(list('m-alu,
              cons('p,
                    cons('g,
                          indices('alu-out, 0, tree-size(tree))),
                    index('tv-alu-help, tree-number(tree)),
                    cons('c,
                          append(indices('a, 0, tree-size(tree)),
                                append(indices('b, 0, tree-size(tree)),
                                      indices('mpg, 0, 7))))),
              '(m-alu-carry (alu-carry) t-carry (c p g)),
              list('m-carry-out-help,
                    '(carry,
                      'carry-out-help,
                      list(index('a, 0),
                            'alu-carry,
                            'zero,
                            index('op, 0),
                            index('op, 1),
                            index('op, 2),
                            index('op, 3))),
                    list('m-overflow-help,
                          '(overflow,
                            'overflow-help,
                            list(index('alu-out, tree-size(tree) - 1),
                                  index('a, tree-size(tree) - 1),
                                  index('b, tree-size(tree) - 1),
                                  'zero,
                                  index('op, 0),
                                  index('op, 1),
                                  index('op, 2),
                                  index('op, 3))),
                          list('m-shift,
                                indices('out, 0, tree-size(tree)),
                                index('tv-shift-or-buf, tree-number(tree)),
                                cons('c,
                                      append(indices('alu-out, 0, tree-size(tree)),
                                            list(index('a, tree-size(tree) - 1),
                                                  'zero,
                                                  index('op, 0),
                                                  index('op, 1),
                                                  index('op, 2),
                                                  index('op, 3))))),

```

```

      list ('m-zerop,
            '(zerop),
            index ('fast-zero, tree-size (tree)),
            indices ('out, 0, tree-size (tree))))
  ^  (caddddr (core-alu* (tree)) = nil)

```

EVENT: Enable core-alu*; name this event 'core-alu*-off'.

EVENT: Enable core-alu*\$destructure; name this event 'core-alu*\$destructure-off'.

DEFINITION:

```

core-alu& (netlist, tree)
=  ((lookup-module (index ('core-alu, tree-number (tree)), netlist)
    =  core-alu* (tree))
  ^  tv-alu-help& (delete-module (index ('core-alu,
                                         tree-number (tree)),
                                         netlist),
                    tree)
  ^  t-carry& (delete-module (index ('core-alu, tree-number (tree)),
                                     netlist))
  ^  carry-out-help& (delete-module (index ('core-alu,
                                             tree-number (tree)),
                                             netlist))
  ^  overflow-help& (delete-module (index ('core-alu,
                                           tree-number (tree)),
                                           netlist))
  ^  tv-shift-or-buf& (delete-module (index ('core-alu,
                                              tree-number (tree)),
                                              netlist),
                       tree)
  ^  fast-zero& (delete-module (index ('core-alu, tree-number (tree)),
                                      netlist),
                tree-size (tree)))

```

EVENT: Enable core-alu&; name this event 'core-alu&-off'.

DEFINITION:

```

core-alu$netlist (tree)
=  cons (core-alu* (tree),
        tv-alu-help$netlist (tree)
        U  (T-CARRY$NETLIST
            U  (CARRY-OUT-HELP$NETLIST

```

$$\begin{aligned} & \cup \text{ (OVERFLOW-HELP\$NETLIST} \\ & \quad \cup \text{ (tv-shift-or-buf\$netlist (tree)} \\ & \quad \quad \cup \text{ fast-zero\$netlist (tree-size (tree)))))) \end{aligned}$$

THEOREM: core-alu\$value

$$\begin{aligned} & (\text{core-alu\& (netlist, tree)} \\ & \quad \wedge \text{ (length (a) = tree-size (tree))} \\ & \quad \wedge \text{ (length (b) = tree-size (tree))} \\ & \quad \wedge \text{ if length (a) < 3 then f} \\ & \quad \quad \text{else t endif} \\ & \quad \wedge \text{ properp (a)} \\ & \quad \wedge \text{ properp (b)} \\ & \quad \wedge \text{ properp (op)} \\ & \quad \wedge \text{ (length (op) = 4)} \\ & \quad \wedge \text{ properp (mpg)} \\ & \quad \wedge \text{ (length (mpg) = 7)} \\ & \rightarrow \text{ (dual-eval (0,} \\ & \quad \text{index ('core-alu, tree-number (tree)),} \\ & \quad \text{cons (c, append (a, append (b, cons (zero, append (mpg, op))))),} \\ & \quad \text{state,} \\ & \quad \text{netlist)} \\ & \quad = \text{ f\$core-alu (c, a, b, zero, mpg, op, tree))} \end{aligned}$$

EVENT: Enable core-alu\$value; name this event 'core-alu\$value-off'.

DEFINITION: REG-SIZE = 4

DEFINITION: a-immediate-p (*instruction*) = nth (9, *instruction*)

DEFINITION:

a-immediate (*instruction*) = subrange (*instruction*, 0, 8)

DEFINITION: rn-a (*instruction*) = subrange (*instruction*, 0, 3)

DEFINITION: mode-a (*instruction*) = subrange (*instruction*, 4, 5)

DEFINITION: rn-b (*instruction*) = subrange (*instruction*, 10, 13)

DEFINITION: mode-b (*instruction*) = subrange (*instruction*, 14, 15)

DEFINITION:

set-flags (*instruction*) = subrange (*instruction*, 16, 19)

DEFINITION:

store-cc (*instruction*) = subrange (*instruction*, 20, 23)

DEFINITION:

$\text{op-code}(instruction) = \text{subrange}(instruction, 24, 27)$

EVENT: Enable a-immediate-p; name this event ‘a-immediate-p-off’.

EVENT: Enable a-immediate; name this event ‘a-immediate-off’.

EVENT: Enable rn-a; name this event ‘rn-a-off’.

EVENT: Enable rn-b; name this event ‘rn-b-off’.

EVENT: Enable mode-a; name this event ‘mode-a-off’.

EVENT: Enable mode-b; name this event ‘mode-b-off’.

EVENT: Enable set-flags; name this event ‘set-flags-off’.

EVENT: Enable store-cc; name this event ‘store-cc-off’.

EVENT: Enable op-code; name this event ‘op-code-off’.

EVENT: Let us define the theory *ir-fields-theory* to consist of the following events: a-immediate-p, a-immediate, rn-a, mode-a, rn-b, mode-b, set-flags, store-cc, op-code.

DEFINITION: $\text{z-set}(set\text{-}flags) = \text{nth}(0, set\text{-}flags)$

DEFINITION: $\text{n-set}(set\text{-}flags) = \text{nth}(1, set\text{-}flags)$

DEFINITION: $\text{v-set}(set\text{-}flags) = \text{nth}(2, set\text{-}flags)$

DEFINITION: $\text{c-set}(set\text{-}flags) = \text{nth}(3, set\text{-}flags)$

EVENT: Enable z-set; name this event ‘z-set-off’.

EVENT: Enable n-set; name this event ‘n-set-off’.

EVENT: Enable v-set; name this event ‘v-set-off’.

EVENT: Enable c-set; name this event ‘c-set-off’.

EVENT: Let us define the theory *set-flags-theory* to consist of the following events: z-set, n-set, v-set, c-set.

DEFINITION: $\text{z-flag}(flags) = \text{nth}(0, flags)$

DEFINITION: $\text{n-flag}(flags) = \text{nth}(1, flags)$

DEFINITION: $\text{v-flag}(flags) = \text{nth}(2, flags)$

DEFINITION: $\text{c-flag}(flags) = \text{nth}(3, flags)$

EVENT: Enable z-flag; name this event ‘z-flag-off’.

EVENT: Enable n-flag; name this event ‘n-flag-off’.

EVENT: Enable v-flag; name this event ‘v-flag-off’.

EVENT: Enable c-flag; name this event ‘c-flag-off’.

EVENT: Let us define the theory *flags-theory* to consist of the following events: z-flag, n-flag, v-flag, c-flag.

DEFINITION: $\text{reg-direct-p}(mode) = (mode = \text{list}(\mathbf{f}, \mathbf{f}))$

DEFINITION: $\text{reg-indirect-p}(mode) = (mode = \text{list}(\mathbf{t}, \mathbf{f}))$

DEFINITION: $\text{pre-dec-p}(mode) = (mode = \text{list}(\mathbf{f}, \mathbf{t}))$

DEFINITION: $\text{post-inc-p}(mode) = (mode = \text{list}(\mathbf{t}, \mathbf{t}))$

EVENT: Enable reg-direct-p; name this event ‘reg-direct-p-off’.

EVENT: Enable reg-indirect-p; name this event ‘reg-indirect-p-off’.

EVENT: Enable pre-dec-p; name this event ‘pre-dec-p-off’.

EVENT: Enable post-inc-p; name this event ‘post-inc-p-off’.

EVENT: Let us define the theory *reg-mode-theory* to consist of the following events: reg-direct-p, reg-indirect-p, pre-dec-p, post-inc-p.

DEFINITION:

```

store-resultp(store-cc, flags)
=  if store-cc = list(f, f, f, f) then ¬ c-flag(flags)
    elseif store-cc = list(t, f, f, f) then c-flag(flags)
    elseif store-cc = list(f, t, f, f) then ¬ v-flag(flags)
    elseif store-cc = list(t, t, f, f) then v-flag(flags)
    elseif store-cc = list(f, f, t, f) then ¬ n-flag(flags)
    elseif store-cc = list(t, f, t, f) then n-flag(flags)
    elseif store-cc = list(f, t, t, f) then ¬ z-flag(flags)
    elseif store-cc = list(t, t, t, f) then z-flag(flags)
    elseif store-cc = list(f, f, f, t)
    then (¬ c-flag(flags)) ∧ (¬ z-flag(flags))
    elseif store-cc = list(t, f, f, t)
    then c-flag(flags) ∨ z-flag(flags)
    elseif store-cc = list(f, t, f, t)
    then (n-flag(flags) ∧ v-flag(flags))
        ∨ ((¬ n-flag(flags)) ∧ (¬ v-flag(flags)))
    elseif store-cc = list(t, t, f, t)
    then (n-flag(flags) ∧ (¬ v-flag(flags)))
        ∨ ((¬ n-flag(flags)) ∧ v-flag(flags))
    elseif store-cc = list(f, f, t, t)
    then (n-flag(flags) ∧ v-flag(flags) ∧ (¬ z-flag(flags)))
        ∨ ((¬ n-flag(flags))
            ∧ (¬ v-flag(flags))
            ∧ (¬ z-flag(flags)))
    elseif store-cc = list(t, f, t, t)
    then z-flag(flags)
        ∨ (n-flag(flags) ∧ (¬ v-flag(flags)))
        ∨ ((¬ n-flag(flags)) ∧ v-flag(flags))
    elseif store-cc = list(f, t, t, t) then t
    else f endif

```

EVENT: Enable store-resultp; name this event ‘store-resultp-off’.

DEFINITION:

```

update-flags(flags, set-flags, cvzbv)
=  list(b-if(z-set(set-flags), zb(cvzbv), z-flag(flags)),
        b-if(n-set(set-flags), n(cvzbv), n-flag(flags)),
        b-if(v-set(set-flags), v(cvzbv), v-flag(flags)),
        b-if(c-set(set-flags), c(cvzbv), c-flag(flags)))

```

EVENT: Enable update-flags; name this event ‘update-flags-off’.

DEFINITION: $\text{regs}(\text{state}) = \text{nth}(0, \text{state})$

DEFINITION: $\text{flags}(state) = \text{nth}(1, state)$

EVENT: Enable regs; name this event ‘regs-off’.

EVENT: Enable flags; name this event ‘flags-off’.

DEFINITION:

$\text{fm9001-alu-operation}(regs, flags, mem, ins, operand-a, operand-b, b-address)$

```
= list (list (if store-resultp (store-cc (ins), flags)
    ∧ reg-direct-p (mode-b (ins))
    then write-mem (rn-b (ins),
        regs,
        bv (v-alu (c-flag (flags),
            operand-a,
            operand-b,
            op-code (ins))))
    else regs endif,
    update-flags (flags,
        set-flags (ins),
        v-alu (c-flag (flags),
            operand-a,
            operand-b,
            op-code (ins))))),
    if store-resultp (store-cc (ins), flags)
    ∧ (¬ reg-direct-p (mode-b (ins)))
    then write-mem (b-address,
        mem,
        bv (v-alu (c-flag (flags),
            operand-a,
            operand-b,
            op-code (ins))))
    else mem endif)
```

DEFINITION:

$\text{fm9001-operand-b}(regs, flags, mem, ins, operand-a)$

```
= fm9001-alu-operation (if pre-dec-p (mode-b (ins))
    then write-mem (rn-b (ins),
        regs,
        v-dec (read-mem (rn-b (ins), regs)))
    elseif post-inc-p (mode-b (ins))
    then write-mem (rn-b (ins),
        regs,
        v-inc (read-mem (rn-b (ins), regs)))
```

```

else regs endif,
flags,
mem,
ins,
operand-a,
if reg-direct-p (mode-b (ins))
then read-mem (rn-b (ins), regs)
else read-mem (if pre-dec-p (mode-b (ins))
               then v-dec (read-mem (rn-b (ins),
                                     regs))
               else read-mem (rn-b (ins),
                               regs) endif,
               mem) endif,
if pre-dec-p (mode-b (ins))
then v-dec (read-mem (rn-b (ins), regs))
else read-mem (rn-b (ins), regs) endif

```

DEFINITION:

```

fm9001-operand-a (regs, flags, mem, ins)
= fm9001-operand-b (if a-immediate-p (ins) then regs
                    elseif pre-dec-p (mode-a (ins))
                    then write-mem (rn-a (ins),
                                     regs,
                                     v-dec (read-mem (rn-a (ins), regs)))
                    elseif post-inc-p (mode-a (ins))
                    then write-mem (rn-a (ins),
                                     regs,
                                     v-inc (read-mem (rn-a (ins), regs)))
                    else regs endif,
                    flags,
                    mem,
                    ins,
                    if a-immediate-p (ins)
                    then sign-extend (a-immediate (ins), 32)
                    elseif reg-direct-p (mode-a (ins))
                    then read-mem (rn-a (ins), regs)
                    elseif pre-dec-p (mode-a (ins))
                    then read-mem (v-dec (read-mem (rn-a (ins), regs)), mem)
                    else read-mem (read-mem (rn-a (ins), regs), mem) endif)

```

DEFINITION:

```

fm9001-fetch (regs, flags, mem, pc-reg)
= fm9001-operand-a (write-mem (pc-reg, regs, v-inc (read-mem (pc-reg, regs))),
                    flags,

```

mem,
 read-mem (read-mem (*pc-reg*, *regs*), *mem*))

DEFINITION:

fm9001-step (*state*, *pc-reg*)
 = fm9001-fetch (regs (car (*state*)), flags (car (*state*)), cadr (*state*), *pc-reg*)

EVENT: Enable fm9001-step; name this event 'fm9001-step-off'.

DEFINITION:

fm9001 (*state*, *n*)
 = **if** *n* $\simeq$ 0 **then** *state*
 else fm9001 (fm9001-step (*state*, nat-to-v (15, REG-SIZE)), *n* - 1) **endif**

EVENT: Enable fm9001; name this event 'fm9001-off'.

DEFINITION:

fm9001-interpreter (*state*, *pc-reg*, *n*)
 = **if** *n* $\simeq$ 0 **then** *state*
 else fm9001-interpreter (fm9001-step (*state*, *pc-reg*), *pc-reg*, *n* - 1) **endif**

EVENT: Enable fm9001-interpreter; name this event 'fm9001-interpreter-off'.

THEOREM: open-fm9001-interpreter

((*n* $\simeq$ 0) $\rightarrow$ (fm9001-interpreter (*state*, *pc-reg*, *n*) = *state*))
 $\wedge$ ((*n* $\not\simeq$ 0)
 $\rightarrow$ (fm9001-interpreter (*state*, *pc-reg*, *n*)
 = fm9001-interpreter (fm9001-step (*state*, *pc-reg*),
 pc-reg,
 n - 1)))

DEFINITION:

fm9001-intr (*state*, *pc-reg-input*)
 = **if** *pc-reg-input* $\simeq$ nil **then** *state*
 else fm9001-intr (fm9001-step (*state*, car (*pc-reg-input*)),
 cdr (*pc-reg-input*)) **endif**

EVENT: Enable fm9001-intr; name this event 'fm9001-intr-off'.

THEOREM: properp-length-ir-accessors

properp (a-immediate (*i-reg*))
 $\wedge$ (length (a-immediate (*i-reg*)) = 9)
 $\wedge$ properp (rn-a (*i-reg*))

$$\begin{aligned}
& \wedge \text{length}(\text{rn-a}(i\text{-reg})) = 4 \\
& \wedge \text{properp}(\text{mode-a}(i\text{-reg})) \\
& \wedge \text{length}(\text{mode-a}(i\text{-reg})) = 2 \\
& \wedge \text{properp}(\text{rn-b}(i\text{-reg})) \\
& \wedge \text{length}(\text{rn-b}(i\text{-reg})) = 4 \\
& \wedge \text{properp}(\text{mode-b}(i\text{-reg})) \\
& \wedge \text{length}(\text{mode-b}(i\text{-reg})) = 2 \\
& \wedge \text{properp}(\text{set-flags}(i\text{-reg})) \\
& \wedge \text{length}(\text{set-flags}(i\text{-reg})) = 4 \\
& \wedge \text{properp}(\text{store-cc}(i\text{-reg})) \\
& \wedge \text{length}(\text{store-cc}(i\text{-reg})) = 4 \\
& \wedge \text{properp}(\text{op-code}(i\text{-reg})) \\
& \wedge \text{length}(\text{op-code}(i\text{-reg})) = 4
\end{aligned}$$

THEOREM: bvp-ir-accessors

$$\begin{aligned}
& (\text{bvp}(i\text{-reg}) \wedge (\text{length}(i\text{-reg}) = 32)) \\
& \rightarrow (\text{bvp}(\text{a-immediate}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{rn-a}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{mode-a}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{rn-b}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{mode-b}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{set-flags}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{store-cc}(i\text{-reg})) \\
& \quad \wedge \text{bvp}(\text{op-code}(i\text{-reg})))
\end{aligned}$$

THEOREM: boolp-a-immediate-p

$$(\text{bvp}(ir) \wedge (\text{length}(ir) = 32)) \rightarrow \text{boolp}(\text{a-immediate-p}(ir))$$

THEOREM: bvp-length-update-flags

$$\begin{aligned}
& \text{bvp}(\text{update-flags}(\text{flags}, \text{set-flags}, \text{cvzbu})) \\
& \wedge (\text{length}(\text{update-flags}(\text{flags}, \text{set-flags}, \text{cvzbu})) = 4)
\end{aligned}$$

THEOREM: boolp-c-flag-update-flags

$$\text{boolp}(\text{c-flag}(\text{update-flags}(\text{flags}, \text{set-flags}, \text{cvzbu})))$$

THEOREM: bvp-a-immediate

$$(\text{bvp}(ir) \wedge (8 < \text{length}(ir))) \rightarrow \text{bvp}(\text{a-immediate}(ir))$$

THEOREM: reg-direct->not-reg-indirect

$$\begin{aligned}
& \text{reg-direct-p}(\text{mode}) \\
& \rightarrow ((\text{reg-indirect-p}(\text{mode}) = \mathbf{f}) \\
& \quad \wedge (\text{pre-dec-p}(\text{mode}) = \mathbf{f}) \\
& \quad \wedge (\text{post-inc-p}(\text{mode}) = \mathbf{f}))
\end{aligned}$$

THEOREM: boolp-store-result-p

$$\begin{aligned}
& (\text{bvp}(\text{flags}) \wedge (\text{length}(\text{flags}) = 4)) \\
& \rightarrow \text{boolp}(\text{store-resultp}(\text{store-cc}, \text{flags}))
\end{aligned}$$

THEOREM: unary-op-code-p-op-code->v-alu=v-alu-1
 unary-op-code-p (op-code (*i-reg*))
 $\rightarrow$ (v-alu (*c*, *a*, *b*, op-code (*i-reg*))) = v-alu-1 (*c*, *a*, op-code (*i-reg*)))

DEFINITION: $\text{intp}(x) = ((x \in \mathbf{N}) \vee \text{negativep}(x))$

DEFINITION:

int-to-bv (*x*, *size*)
 = **if** negativep (*x*)
 then v-subtractor-output (**f**,
 nat-to-v (negative-guts (*x*), *size*),
 nat-to-v (0, *size*))
 else nat-to-v (*x*, *size*) **endif**

DEFINITION:

bv-to-int (*x*)
 = **if** nth (length (*x*) - 1, *x*) **then** - v-to-nat (v-inc (v-not (*x*)))
 else v-to-nat (*x*) **endif**

DEFINITION:

c10-tf (*x*)
 = **if** *x* $\simeq$ nil **then** nil
 else cons (**if** car (*x*) = 0 **then** **f**
 else **t** **endif**,
 c10-tf (cdr (*x*))) **endif**

DEFINITION: $\text{c10-int}(x) = \text{bv-to-int}(\text{c10-tf}(x))$

DEFINITION:

add (*x*, *y*)
 = **if** negativep (*x*)
 then **if** negativep (*y*) **then** - (negative-guts (*x*) + negative-guts (*y*))
 elseif *y* < negative-guts (*x*) **then** - (negative-guts (*x*) - *y*)
 else *y* - negative-guts (*x*) **endif**
 elseif negativep (*y*)
 then **if** *x* < negative-guts (*y*) **then** - (negative-guts (*y*) - *x*)
 else *x* - negative-guts (*y*) **endif**
 else *x* + *y* **endif**

DEFINITION:

integer-minus (*x*)
 = **if** negativep (*x*) **then** negative-guts (*x*)
 elseif *x* $\simeq$ 0 **then** 0
 else - *x* **endif**

DEFINITION: $\text{sub}(x, y) = \text{add}(x, \text{integer-minus}(y))$

DEFINITION:

$\text{exp2}(y)$
= **if** $y \simeq 0$ **then** 1
 else $2 * \text{exp2}(y - 1)$ **endif**

THEOREM: quotient-stuff

$((n \neq 0) \wedge (n \in \mathbf{N})) \rightarrow ((n \div 2) < n)$

DEFINITION:

$\text{log2}(n)$
= **if** $(n \simeq 0) \vee (n = 1)$ **then** 0
 else $1 + \text{log2}(n \div 2)$ **endif**

DEFINITION:

$\text{asm-register}(x)$
= **case on** x :
 case = $r0$
 then list (f, f, f, f, f, f)
 case = $r1$
 then list (t, f, f, f, f, f)
 case = $r2$
 then list (f, t, f, f, f, f)
 case = $r3$
 then list (t, t, f, f, f, f)
 case = $r4$
 then list (f, f, t, f, f, f)
 case = $r5$
 then list (t, f, t, f, f, f)
 case = $r6$
 then list (f, t, t, f, f, f)
 case = $r7$
 then list (t, t, t, f, f, f)
 case = $r8$
 then list (f, f, f, t, f, f)
 case = $r9$
 then list (t, f, f, t, f, f)
 case = $r10$
 then list (f, t, f, t, f, f)
 case = $r11$
 then list (t, t, f, t, f, f)
 case = $r12$
 then list (f, f, t, t, f, f)
 case = $r13$
 then list (t, f, t, t, f, f)
 case = lnk

```

    then list (t, f, t, t, f, f)
case = r14
    then list (f, t, t, t, f, f)
case = tos
    then list (f, t, t, t, f, f)
case = r15
    then list (t, t, t, t, f, f)
case = pc
    then list (t, t, t, t, f, f)
case = R0
    then list (f, f, f, f, t, f)
case = R1
    then list (t, f, f, f, t, f)
case = R2
    then list (f, t, f, f, t, f)
case = R3
    then list (t, t, f, f, t, f)
case = R4
    then list (f, f, t, f, t, f)
case = R5
    then list (t, f, t, f, t, f)
case = R6
    then list (f, t, t, f, t, f)
case = R7
    then list (t, t, t, f, t, f)
case = R8
    then list (f, f, f, t, t, f)
case = R9
    then list (t, f, f, t, t, f)
case = R10
    then list (f, t, f, t, t, f)
case = R11
    then list (t, t, f, t, t, f)
case = R12
    then list (f, f, t, t, t, f)
case = R13
    then list (t, f, t, t, t, f)
case = LNK
    then list (t, f, t, t, t, f)
case = R14
    then list (f, t, t, t, t, f)
case = TOS
    then list (f, t, t, t, t, f)
case = R15

```

```

    then list (t, t, t, t, t, f)
case = PC
    then list (t, t, t, t, t, f)
case = R0-
    then list (f, f, f, f, f, t)
case = R1-
    then list (t, f, f, f, f, t)
case = R2-
    then list (f, t, f, f, f, t)
case = R3-
    then list (t, t, f, f, f, t)
case = R4-
    then list (f, f, t, f, f, t)
case = R5-
    then list (t, f, t, f, f, t)
case = R6-
    then list (f, t, t, f, f, t)
case = R7-
    then list (t, t, t, f, f, t)
case = R8-
    then list (f, f, f, t, f, t)
case = R9-
    then list (t, f, f, t, f, t)
case = R10-
    then list (f, t, f, t, f, t)
case = R11-
    then list (t, t, f, t, f, t)
case = R12-
    then list (f, f, t, t, f, t)
case = R13-
    then list (t, f, t, t, f, t)
case = LNK-
    then list (t, f, t, t, f, t)
case = R14-
    then list (f, t, t, t, f, t)
case = TOS-
    then list (f, t, t, t, f, t)
case = R15-
    then list (t, t, t, t, f, t)
case = PC-
    then list (t, t, t, t, f, t)
case = R0+
    then list (f, f, f, f, t, t)
case = R1+

```



```

    then list (t, f, f, f, t, t)
case = R2+
    then list (f, t, f, f, t, t)
case = R3+
    then list (t, t, f, f, t, t)
case = R4+
    then list (f, f, t, f, t, t)
case = R5+
    then list (t, f, t, f, t, t)
case = R6+
    then list (f, t, t, f, t, t)
case = R7+
    then list (t, t, t, f, t, t)
case = R8+
    then list (f, f, f, t, t, t)
case = R9+
    then list (t, f, f, t, t, t)
case = R10+
    then list (f, t, f, t, t, t)
case = R11+
    then list (t, t, f, t, t, t)
case = R12+
    then list (f, f, t, t, t, t)
case = R13+
    then list (t, f, t, t, t, t)
case = LNK+
    then list (t, f, t, t, t, t)
case = R14+
    then list (f, t, t, t, t, t)
case = TOS+
    then list (f, t, t, t, t, t)
case = R15+
    then list (t, t, t, t, t, t)
case = PC+
    then list (t, t, t, t, t, t)
otherwise list (f, f, f, f, f, f) endcase

```

DEFINITION:

asm-register-a (x)

```

= if intp ( $x$ ) then append (int-to-bv ( $x$ , 9), list (t))
  else append (asm-register ( $x$ ), list (f, f, f, f)) endif

```

DEFINITION:

asm-op-code (x)

```

=  case on  $x$ :
    case = move
    then list (f, f, f, f)
    case = inc
    then list (t, f, f, f)
    case = addc
    then list (f, t, f, f)
    case = add
    then list (t, t, f, f)
    case = neg
    then list (f, f, t, f)
    case = dec
    then list (t, f, t, f)
    case = subb
    then list (f, t, t, f)
    case = sub
    then list (t, t, t, f)
    case = ror
    then list (f, f, f, t)
    case = asr
    then list (t, f, f, t)
    case = lsh
    then list (f, t, f, t)
    case = xor
    then list (t, t, f, t)
    case = or
    then list (f, f, t, t)
    case = and
    then list (t, f, t, t)
    case = not
    then list (f, t, t, t)
    case = m15
    then list (t, t, t, t)
    otherwise list (f, f, f, f) endcase

```

DEFINITION:

asm-store-cc (x)

```

=  case on  $x$ :
    case = cc
    then list (f, f, f, f)
    case = cs
    then list (t, f, f, f)
    case = vc
    then list (f, t, f, f)

```

```

case = vs
  then list (t, t, f, f)
case = pl
  then list (f, f, t, f)
case = nc
  then list (f, f, t, f)
case = mi
  then list (t, f, t, f)
case = ns
  then list (t, f, t, f)
case = ne
  then list (f, t, t, f)
case = zc
  then list (f, t, t, f)
case = eq
  then list (t, t, t, f)
case = zs
  then list (t, t, t, f)
case = hi
  then list (f, f, f, t)
case = ls
  then list (t, f, f, t)
case = ge
  then list (f, t, f, t)
case = lt
  then list (t, t, f, t)
case = gt
  then list (f, f, t, t)
case = le
  then list (t, f, t, t)
case = t
  then list (f, t, t, t)
case = f
  then list (t, t, t, t)
otherwise list (f, f, f, f) endcase

```

DEFINITION:

asm-flags (*x*)

```

= if x = 't' then list (t, t, t, t)
  else list (car (unpack ('z')) ∈ unpack (x),
    car (unpack ('n')) ∈ unpack (x),
    car (unpack ('v')) ∈ unpack (x),
    car (unpack ('c')) ∈ unpack (x)) endif

```

DEFINITION:

```

resolve-names(list, cnt, alist)
=  if list  $\simeq$  nil then alist
    elseif litatom(car(list))
    then resolve-names(cdr(list), cnt, cons(cons(car(list), cnt), alist))
    else resolve-names(cdr(list), 1 + cnt, alist) endif

```

DEFINITION:

```

update-list(list, alist)
=  if list  $\simeq$  nil then nil
    else cons(if listp(car(list))  $\wedge$  (caar(list) = 'value)
        then eval$(t, cadar(list), alist)
        else car(list) endif,
        update-list(cdr(list), alist) endif

```

DEFINITION:

```

asm-line(line)
=  if intp(line) then int-to-bv(line, 32)
    else append(asm-register-a(caddddr(line)),
        append(asm-register(cadddr(line)),
            append(asm-flags(caddr(line)),
                append(asm-store-cc(cadr(line)),
                    append(asm-op-code(car(line)),
                        list(f, f, f, f)))))) endif

```

DEFINITION:

```

asm-list(list)
=  if list  $\simeq$  nil then nil
    elseif litatom(car(list)) then asm-list(cdr(list))
    else cons(asm-line(car(list)), asm-list(cdr(list))) endif

```

DEFINITION:

```

asm(list) = asm-list(update-list(list, resolve-names(list, 0, nil)))

```

DEFINITION:

```

t-and-f-to-1-and-0(list)
=  if list  $\simeq$  nil then nil
    else cons(if falsep(car(list)) then 0
        else 1 endif,
        t-and-f-to-1-and-0(cdr(list))) endif

```

DEFINITION:

```

asm-to-1-and-0(list)
=  if list  $\simeq$  nil then nil
    else cons(t-and-f-to-1-and-0(car(list)),
        asm-to-1-and-0(cdr(list))) endif

```

DEFINITION:

$\text{v-to-nat-all}(x)$

= **if** $x \simeq \text{nil}$ **then** **nil**
 else $\text{cons}(\text{v-to-nat}(\text{car}(x)), \text{v-to-nat-all}(\text{cdr}(x)))$ **endif**

DEFINITION:

$\text{v-to-0s-through-fs}(x, \text{collect})$

= **if** $x \simeq \text{nil}$ **then** collect
 else $\text{v-to-0s-through-fs}(\text{cddddr}(x),$
 $\text{cons}(\text{case on v-to-nat}(\text{firstn}(4, x)):$
 case = 0
 then $\text{cadr}(\text{unpack}('x0))$
 case = 1
 then $\text{cadr}(\text{unpack}('x1))$
 case = 2
 then $\text{cadr}(\text{unpack}('x2))$
 case = 3
 then $\text{cadr}(\text{unpack}('x3))$
 case = 4
 then $\text{cadr}(\text{unpack}('x4))$
 case = 5
 then $\text{cadr}(\text{unpack}('x5))$
 case = 6
 then $\text{cadr}(\text{unpack}('x6))$
 case = 7
 then $\text{cadr}(\text{unpack}('x7))$
 case = 8
 then $\text{cadr}(\text{unpack}('x8))$
 case = 9
 then $\text{cadr}(\text{unpack}('x9))$
 case = 10
 then $\text{cadr}(\text{unpack}('xa))$
 case = 11
 then $\text{cadr}(\text{unpack}('xb))$
 case = 12
 then $\text{cadr}(\text{unpack}('xc))$
 case = 13
 then $\text{cadr}(\text{unpack}('xd))$
 case = 14
 then $\text{cadr}(\text{unpack}('xe))$
 otherwise $\text{cadr}(\text{unpack}('xf))$ **endcase,**
 $\text{collect}))$ **endif**

DEFINITION:

$\text{v-to-hex}(x) = \text{pack}(\text{cons}(\text{car}(\text{unpack}('x)), \text{v-to-0s-through-fs}(x, 0)))$

DEFINITION:

v-to-hex-all(x)

= **if** $x \simeq \mathbf{nil}$ **then** $\mathbf{nil}$
 else cons(v-to-hex(car(x)), v-to-hex-all(cdr(x))) **endif**

DEFINITION:

make-pairs($list$)

= **if** ($list \simeq \mathbf{nil}$) $\vee$ (cdr($list$) $\simeq \mathbf{nil}$) **then** $\mathbf{nil}$
 else cons(cons(car($list$), cadr($list$)), make-pairs(cddr($list$))) **endif**

DEFINITION:

add-ram-marker($list\text{-}mem$)

= **if** $list\text{-}mem \simeq \mathbf{nil}$ **then** $\mathbf{nil}$
 else cons(ram(car($list\text{-}mem$)), add-ram-marker(cdr($list\text{-}mem$))) **endif**

DEFINITION:

list-to-mem3(mem , $depth$)

= **if** $depth \simeq 0$ **then** mem
 else list-to-mem3(make-pairs(mem), $depth - 1$) **endif**

DEFINITION:

list-to-mem2(mem , $depth$) = car(list-to-mem3(mem , $depth$))

DEFINITION:

list-to-mem1(mem , $default$)

= list-to-mem2(add-ram-marker(append(mem ,
 make-list(exp2(1 + log2(length(mem)))
 - length(mem),
 $default$))),
 1 + log2(length(mem)))

DEFINITION:

stub-right(mem , $levels$, $default$)

= **if** $levels \simeq 0$ **then** mem
 else cons(stub-right(mem , $levels - 1$, $default$), stub($default$)) **endif**

DEFINITION:

list-to-mem(mem , $size$, $default$)

= **if** $size < (1 + \log_2(\text{length}(\mathbf{mem})))$
 then '(list-to-mem= insufficient-size)
 else stub-right(list-to-mem1(mem , $default$),
 $size - (1 + \log_2(\text{length}(\mathbf{mem})))$,
 $default$) **endif**

DEFINITION:

list-to-tree-mem(mem) = list-to-mem(mem , 32, make-list(32, f))

DEFINITION:

```

mem-to-list (mem)
=  if listp (mem)
    then append (mem-to-list (car (mem)), mem-to-list (cdr (mem)))
    elseif stubp (mem) then nil
    elseif ramp (mem) then list (ram-guts (mem))
    elseif romp (mem) then list (rom-guts (mem))
    else mem endif

```

DEFINITION:

```

un-fm9001 (state)
=  list (list (v-to-nat-all (mem-to-list (regs (car (state)))),
               z-flag (flags (car (state))),
               n-flag (flags (car (state))),
               v-flag (flags (car (state))),
               c-flag (flags (car (state))),
               v-to-hex-all (mem-to-list (cadr (state))))))

```

DEFINITION:

```

execute-fm9001 (n, tree-pgm)
=  un-fm9001 (fm9001 (list (list (list-to-mem (make-list (15, nat-to-v (0, 32)),
                                                         4,
                                                         make-list (32, f)),
                               list (t, f, f, f)),
                             tree-pgm),
              n))

```

DEFINITION:

```

asm-and-fm9001 (n, program)
=  execute-fm9001 (n, list-to-mem (asm (program), 32, make-list (32, f)))

```

DEFINITION:

```

store-resultp-mux (s0, s1, s2, d0, d1, d2, d3, d4, d5, d6)
=  ao2 (b-not (s2),
        ao2 (b-not (s1), ao2 (b-not (s0), d0, s0, d1), s1, ao2 (b-not (s0), d2, s0, d3)),
        s2,
        ao2 (b-not (s1), ao2 (b-not (s0), d4, s0, d5), s1, b-nand (b-not (s0), d6)))

```

DEFINITION:

```

f$store-resultp-mux (s0, s1, s2, d0, d1, d2, d3, d4, d5, d6)
=  f$ao2 (f-not (s2),
          f$ao2 (f-not (s1),
                  f$ao2 (f-not (s0), d0, s0, d1),
                  s1,
                  f$ao2 (f-not (s0), d2, s0, d3)),

```

```

s2,
f$ao2 (f-not (s1),
      f$ao2 (f-not (s0), d4, s0, d5),
      s1,
      f-nand (f-not (s0), d6)))

```

EVENT: Enable f\$store-resulttp-mux; name this event ‘f\$store-resulttp-mux-off’.

DEFINITION:

STORE-RESULTTP-MUX*

```

= '(store-resulttp-mux
   (s0 s1 s2 d0 d1 d2 d3 d4 d5 d6)
   (w-13)
   ((g-10 (w-10) b-not (s0))
    (g-11 (w-11) b-nand (w-10 d6))
    (g-9 (w-9) ao2 (w-10 d4 s0 d5))
    (g-7 (w-7) b-not (s1))
    (g-12 (w-12) ao2 (w-7 w-9 s1 w-11))
    (g-5 (w-5) ao2 (w-10 d2 s0 d3))
    (g-3 (w-3) ao2 (w-10 d0 s0 d1))
    (g-6 (w-6) ao2 (w-7 w-3 s1 w-5))
    (g-0 (w-0) b-not (s2))
    (g-13 (w-13) ao2 (w-0 w-6 s2 w-12)))
   nil)

```

DEFINITION:

store-resulttp-mux& (*netlist*)

```

= ((lookup-module ('store-resulttp-mux, netlist)
   = STORE-RESULTTP-MUX*)
   ^ b-nand& (delete-module ('store-resulttp-mux, netlist))
   ^ b-not& (delete-module ('store-resulttp-mux, netlist))
   ^ ao2& (delete-module ('store-resulttp-mux, netlist)))

```

EVENT: Enable store-resulttp-mux&; name this event ‘store-resulttp-mux&-off’.

DEFINITION:

STORE-RESULTTP-MUX\$NETLIST

```

= cons (STORE-RESULTTP-MUX*,
      (B-NAND$NETLIST ∪ B-NOT$NETLIST) ∪ AO2$NETLIST)

```

THEOREM: store-resulttp-mux\$value

store-resulttp-mux& (*netlist*)

```

→ (dual-eval (0,
              'store-resulttp-mux,

```


$$\begin{aligned}
& \text{list}(s0, s1, s2, d0, d1, d2, d3, d4, d5, d6), \\
& \text{state}, \\
& \text{netlist}) \\
= & \text{list}(\text{f\$store-resultp-mux}(s0, s1, s2, d0, d1, d2, d3, d4, d5, d6)))
\end{aligned}$$

EVENT: Enable store-resultp-mux\$value; name this event 'store-resultp-mux\$value-off'.

THEOREM: f\$store-resultp-mux=store-resultp-mux

$$\begin{aligned}
& (\text{boolp}(s0) \\
& \wedge \text{boolp}(s1) \\
& \wedge \text{boolp}(s2) \\
& \wedge \text{boolp}(d0) \\
& \wedge \text{boolp}(d1) \\
& \wedge \text{boolp}(d2) \\
& \wedge \text{boolp}(d3) \\
& \wedge \text{boolp}(d4) \\
& \wedge \text{boolp}(d5) \\
& \wedge \text{boolp}(d6)) \\
\rightarrow & (\text{f\$store-resultp-mux}(s0, s1, s2, d0, d1, d2, d3, d4, d5, d6) \\
& = \text{store-resultp-mux}(s0, s1, s2, d0, d1, d2, d3, d4, d5, d6))
\end{aligned}$$

DEFINITION:

$$\begin{aligned}
& \text{b-store-resultp}(store-cc, flags) \\
= & \text{b-xor}(\text{car}(store-cc), \\
& \text{store-resultp-mux}(\text{cadr}(store-cc), \\
& \text{caddr}(store-cc), \\
& \text{cadddr}(store-cc), \\
& \text{cadddr}(flags), \\
& \text{caddr}(flags), \\
& \text{cadr}(flags), \\
& \text{car}(flags), \\
& \text{b-or}(\text{cadddr}(flags), \text{car}(flags)), \\
& \text{b-xor}(\text{cadr}(flags), \text{caddr}(flags)), \\
& \text{b-or}(\text{car}(flags), \\
& \text{b-xor}(\text{cadr}(flags), \text{caddr}(flags))))))
\end{aligned}$$

EVENT: Enable b-store-resultp; name this event 'b-store-resultp-off'.

DEFINITION:

$$\begin{aligned}
& \text{f$b-store-resultp}(store-cc, flags) \\
= & \text{f-xor}(\text{car}(store-cc), \\
& \text{f\$store-resultp-mux}(\text{cadr}(store-cc), \\
& \text{caddr}(store-cc),
\end{aligned}$$

```

caddr (store-cc),
caddr (flags),
caddr (flags),
cadr (flags),
car (flags),
f-or (caddr (flags), car (flags)),
f-xor (cadr (flags), caddr (flags)),
f-or (car (flags),
      f-xor (cadr (flags), caddr (flags))))

```

EVENT: Enable f\$b-store-resultp; name this event 'f\$b-store-resultp-off'.

THEOREM: f\$b-store-resultp=b-store-resultp

```

(bvp (store-cc)
  ^ (length (store-cc) = 4)
  ^ bvp (flags)
  ^ (length (flags) = 4))
→ (f$b-store-resultp (store-cc, flags)
    = b-store-resultp (store-cc, flags))

```

DEFINITION:

B-STORE-RESULTP*

```

= ' (b-store-resultp
    (s0 s1 s2 s3 z n v c)
    (result)
    ((g0 (cz) b-or (c z))
     (g1 (nv) b-xor (n v))
     (g2 (znv) b-or (z nv))
     (g3
      (mux)
      store-resultp-mux
      (s1 s2 s3 c v n z cz nv znv))
     (g4 (result) b-xor (s0 mux)))
    nil)

```

DEFINITION:

b-store-resultp& (netlist)

```

= ((lookup-module ('b-store-resultp, netlist) = B-STORE-RESULTP*)
   ^ b-or& (delete-module ('b-store-resultp, netlist))
   ^ store-resultp-mux& (delete-module ('b-store-resultp,
                                         netlist))
   ^ b-xor& (delete-module ('b-store-resultp, netlist)))

```

EVENT: Enable b-store-resultp&; name this event 'b-store-resultp&-off'.

DEFINITION:

B-STORE-RESULTP\$NETLIST
= cons (B-STORE-RESULTP*,
(B-OR\$NETLIST $\cup$ STORE-RESULTP-MUX\$NETLIST) $\cup$ B-XOR\$NETLIST)

THEOREM: b-store-resultp=store-resultp\$help

(boolp (*s0*)
 $\wedge$ boolp (*s1*)
 $\wedge$ boolp (*s2*)
 $\wedge$ boolp (*s3*)
 $\wedge$ boolp (*s4*)
 $\wedge$ boolp (*s5*)
 $\wedge$ boolp (*s6*)
 $\wedge$ boolp (*s7*))
 $\rightarrow$ (b-store-resultp (list (*s0*, *s1*, *s2*, *s3*), list (*s4*, *s5*, *s6*, *s7*))
= store-resultp (list (*s0*, *s1*, *s2*, *s3*), list (*s4*, *s5*, *s6*, *s7*)))

THEOREM: b-store-resultp=store-resultp

(bvp (*store-cc*)
 $\wedge$ (length (*store-cc*) = 4)
 $\wedge$ bvp (*flags*)
 $\wedge$ (length (*flags*) = 4))
 $\rightarrow$ (b-store-resultp (*store-cc*, *flags*) = store-resultp (*store-cc*, *flags*))

THEOREM: b-store-resultp\$value

(b-store-resultp& (*netlist*)
 $\wedge$ properp (*store-cc*)
 $\wedge$ (length (*store-cc*) = 4)
 $\wedge$ properp (*store-cc*)
 $\wedge$ (length (*flags*) = 4))
 $\rightarrow$ (dual-eval (0,
' b-store-resultp,
append (*store-cc*, *flags*),
state,
netlist)
= list (f\$b-store-resultp (*store-cc*, *flags*)))

DEFINITION:

f\$unary-op-code-p (*op*)
= f-nand4 (f-nand (f-not (caddr (*op*)), f-not (cadr (*op*))),
f-nand (f-not (caddr (*op*)), f-not (cadr (*op*))),
f-nand3 (caddr (*op*), cadr (*op*), f-not (car (*op*))),
f-nand3 (caddr (*op*), caddr (*op*), cadr (*op*)))

EVENT: Enable f\$unary-op-code-p; name this event 'f\$unary-op-code-p-off'.

THEOREM: $f\$unary-op-code-p = unary-op-code-p$
 $(bvp(op) \wedge (length(op) = 4))$
 $\rightarrow (f\$unary-op-code-p(op) = unary-op-code-p(op))$

DEFINITION:

UNARY-OP-CODE-P*

```
= ' (unary-op-code-p
    (op0 op1 op2 op3)
    (z)
    ((g0 (op0-) b-not (op0))
     (g1 (op1-) b-not (op1))
     (g2 (op2-) b-not (op2))
     (g3 (op3-) b-not (op3))
     (g4 (s0) b-nand (op3- op1-))
     (g5 (s1) b-nand (op2- op1-))
     (g6 (s2) b-nand3 (op3 op1 op0-))
     (g7 (s3) b-nand3 (op3 op2 op1))
     (g8 (z) b-nand4 (s0 s1 s2 s3)))
    nil)
```

DEFINITION:

```
unary-op-code-p& (netlist)
= ((lookup-module ('unary-op-code-p, netlist) = UNARY-OP-CODE-P*)
   ^ b-not& (delete-module ('unary-op-code-p, netlist))
   ^ b-nand& (delete-module ('unary-op-code-p, netlist))
   ^ b-nand3& (delete-module ('unary-op-code-p, netlist))
   ^ b-nand4& (delete-module ('unary-op-code-p, netlist)))
```

EVENT: Enable unary-op-code-p&; name this event 'unary-op-code-p&-off'.

DEFINITION:

UNARY-OP-CODE-P\$NETLIST

```
= cons (UNARY-OP-CODE-P*,
        ((B-NOT$NETLIST U B-NAND$NETLIST) U B-NAND3$NETLIST)
        U B-NAND4$NETLIST)
```

THEOREM: unary-op-code-p\$value

```
unary-op-code-p& (netlist)
→ (dual-eval (0, 'unary-op-code-p, op-code, state, netlist)
    = list (f$unary-op-code-p (op-code)))
```

EVENT: Enable unary-op-code-p\$value; name this event 'unary-op-code-p\$value-off'.

DEFINITION:

$f\$decode-reg-mode(m)$
 $= \text{list}(f-nor(car(m), cadr(m)), f-nor(car(m), f-not(cadr(m))), id(cadr(m)))$

EVENT: Enable $f\$decode-reg-mode$; name this event ' $f\$decode-reg-mode-off$ '.

DEFINITION:

DECODE-REG-MODE*

$= \text{'(decode-reg-mode}$
 $\quad (m0\ m1)$
 $\quad (direct\ pre-dec\ side-effect)$
 $\quad ((g0\ (direct)\ b-nor\ (m0\ m1))$
 $\quad \quad (g1\ (m1-) \ b-not\ (m1))$
 $\quad \quad (g2\ (pre-dec)\ b-nor\ (m0\ m1-))$
 $\quad \quad (g3\ (side-effect)\ id\ (m1)))$
 $\quad nil)$

DEFINITION:

decode-reg-mode& (*netlist*)

$= ((lookup-module('decode-reg-mode, netlist) = DECODE-REG-MODE*)$
 $\quad \wedge\ b-not\&\ (delete-module('decode-reg-mode, netlist))$
 $\quad \wedge\ b-nor\&\ (delete-module('decode-reg-mode, netlist))$
 $\quad \wedge\ id\&\ (delete-module('decode-reg-mode, netlist)))$

EVENT: Enable decode-reg-mode&; name this event ' $decode-reg-mode\&-off$ '.

DEFINITION:

DECODE-REG-MODE\$NETLIST

$= \text{cons}(DECODE-REG-MODE*,$
 $\quad (B-NOT\$NETLIST \cup B-NOR\$NETLIST) \cup ID\$NETLIST)$

THEOREM: decode-reg-mode\$value

decode-reg-mode& (*netlist*)
 $\rightarrow (\text{dual-eval}(0, 'decode-reg-mode, mode, state, netlist)$
 $\quad = f\$decode-reg-mode(mode))$

EVENT: Enable decode-reg-mode\$value; name this event ' $decode-reg-mode\$value-off$ '.

THEOREM: $f\$decode-reg-mode-as-reg-mode$

$(bvp(mode) \wedge (\text{length}(mode) = 2))$
 $\rightarrow (f\$decode-reg-mode(mode)$
 $\quad = \text{list}(\text{reg-direct-p}(mode),$
 $\quad \quad \text{pre-dec-p}(mode),$
 $\quad \quad \text{pre-dec-p}(mode) \vee \text{post-inc-p}(mode)))$

DEFINITION:

```
f$select-op-code (select, dec, op)
= list (f-nand (select, f-not (car (op))),
        f-and (select, cadr (op)),
        f-if (select, caddr (op), dec),
        f-and (select, caddr (op)))
```

EVENT: Enable f\$select-op-code; name this event 'f\$select-op-code-off'.

THEOREM: properp-length-f\$select-op-code

```
properp (f$select-op-code (select, dec, op))
 $\wedge$  (length (f$select-op-code (select, dec, op)) = 4)
```

THEOREM: f\$select-op-code-selects

```
(boolp (select)  $\wedge$  boolp (dec)  $\wedge$  bvp (op)  $\wedge$  (length (op) = 4))
 $\rightarrow$  (f$select-op-code (select, dec, op)
     = v-if (select, op, v-if (dec, ALU-DEC-OP, ALU-INC-OP)))
```

DEFINITION:

```
SELECT-OP-CODE*
= '(select-op-code
   (select dec op0 op1 op2 op3)
   (z0 z1 z2 z3)
   ((i0 (op0-) b-not (op0))
    (g0 (z0) b-nand (select op0-))
    (g1 (z1) b-and (select op1))
    (g2 (z2) b-if (select op2 dec))
    (g3 (z3) b-and (select op3)))
   nil)
```

DEFINITION:

```
select-op-code& (netlist)
= ((lookup-module ('select-op-code, netlist) = SELECT-OP-CODE*)
    $\wedge$  b-not& (delete-module ('select-op-code, netlist))
    $\wedge$  b-nand& (delete-module ('select-op-code, netlist))
    $\wedge$  b-if& (delete-module ('select-op-code, netlist))
    $\wedge$  b-and& (delete-module ('select-op-code, netlist)))
```

EVENT: Enable select-op-code&; name this event 'select-op-code&-off'.

DEFINITION:

```
SELECT-OP-CODE$NETLIST
= cons (SELECT-OP-CODE*,
        ((B-NOT$NETLIST  $\cup$  B-NAND$NETLIST)  $\cup$  B-IF$NETLIST)
         $\cup$  B-AND$NETLIST)
```

THEOREM: $\text{select-op-code}\$value$
 $(\text{select-op-code}\&(\text{netlist}) \wedge \text{properp}(op) \wedge (\text{length}(op) = 4))$
 $\rightarrow (\text{dual-eval}(0,$
 $\quad \text{'select-op-code},$
 $\quad \text{cons}(\text{select}, \text{cons}(\text{dec}, op)),$
 $\quad \text{state},$
 $\quad \text{netlist})$
 $= \text{f}\$select-op-code(\text{select}, \text{dec}, op))$

EVENT: Enable $\text{select-op-code}\$value$; name this event ' $\text{select-op-code}\$value\text{-off}$ '.

DEFINITION:

V-IF-F-4^*
 $= \text{'(v-if-f-4}$
 $\quad (c\ a0\ a1\ a2\ a3)$
 $\quad (z0\ z1\ z2\ z3)$
 $\quad ((cb\ (c-) \text{ b-not } (c))$
 $\quad \quad (g0\ (z0) \text{ b-and } (c- \ a0))$
 $\quad \quad (g1\ (z1) \text{ b-and } (c- \ a1))$
 $\quad \quad (g2\ (z2) \text{ b-and } (c- \ a2))$
 $\quad \quad (g3\ (z3) \text{ b-and } (c- \ a3)))$
 $\quad \text{nil})$

DEFINITION:

$\text{v-if-f-4}\&(\text{netlist})$
 $= ((\text{lookup-module}(\text{'v-if-f-4}, \text{netlist}) = \text{V-IF-F-4}^*)$
 $\quad \wedge \text{ b-not}\&(\text{delete-module}(\text{'v-if-f-4}, \text{netlist}))$
 $\quad \wedge \text{ b-and}\&(\text{delete-module}(\text{'v-if-f-4}, \text{netlist})))$

EVENT: Enable $\text{v-if-f-4}\&$; name this event ' $\text{v-if-f-4}\&\text{-off}$ '.

DEFINITION:

$\text{V-IF-F-4}\$NETLIST = \text{cons}(\text{V-IF-F-4}^*, \text{B-NOT}\$NETLIST \cup \text{B-AND}\$NETLIST)$

DEFINITION:

$\text{f}\$v\text{-if-f-4}(c, a)$
 $= \text{list}(\text{f-and}(\text{f-not}(c), \text{car}(a)),$
 $\quad \text{f-and}(\text{f-not}(c), \text{cadr}(a)),$
 $\quad \text{f-and}(\text{f-not}(c), \text{caddr}(a)),$
 $\quad \text{f-and}(\text{f-not}(c), \text{caddr}(a)))$

EVENT: Enable $\text{f}\$v\text{-if-f-4}$; name this event ' $\text{f}\$v\text{-if-f-4}\text{-off}$ '.

THEOREM: $\text{properp-length-f}\$v\text{-if-f-4}$

$\text{properp}(\text{f}\$v\text{-if-f-4}(c, a)) \wedge (\text{length}(\text{f}\$v\text{-if-f-4}(c, a)) = 4)$

THEOREM: v-if-f-4\$value
v-if-f-4& (*netlist*)
→ (dual-eval (0, 'v-if-f-4, cons (*c*, *a*), *state*, *netlist*)
= f\$v-if-f-4 (*c*, *a*))

THEOREM: f\$v-if-f-4=fv-if
boolp (*c*) → (f\$v-if-f-4 (*c*, *a*) = fv-if (*c*, make-list (4, f), *a*))

THEOREM: v-if-f-4\$reset-value
v-if-f-4& (*netlist*)
→ (dual-eval (0, 'v-if-f-4, cons (t, *a*), *state*, *netlist*)
= make-list (4, f))

EVENT: Enable v-if-f-4\$reset-value; name this event 'v-if-f-4\$reset-value-off'.

DEFINITION:
FANOUT-4*
= ' (fanout-4
(a)
(z0 z1 z2 z3)
((aa (aa) b-buf (a))
(g0 (z0) id (aa))
(g1 (z1) id (aa))
(g2 (z2) id (aa))
(g3 (z3) id (aa)))
nil)

DEFINITION:
fanout-4& (*netlist*)
= ((lookup-module ('fanout-4, *netlist*) = FANOUT-4*)
∧ b-buf& (delete-module ('fanout-4, *netlist*))
∧ id& (delete-module ('fanout-4, *netlist*)))

EVENT: Enable fanout-4&; name this event 'fanout-4&-off'.

DEFINITION:
FANOUT-4\$NETLIST = cons (FANOUT-4*, B-BUF\$NETLIST ∪ ID\$NETLIST)

THEOREM: fanout-4\$value
fanout-4& (*netlist*)
→ (dual-eval (0, 'fanout-4, list (*a*), *state*, *netlist*)
= make-list (4, threefix (*a*)))

EVENT: Enable fanout-4\$value; name this event 'fanout-4\$value-off'.

DEFINITION:

FANOUT-5*

```
= ' (fanout-5
    (a)
    (z0 z1 z2 z3 z4)
    ((aa (aa) b-buf (a))
     (g0 (z0) id (aa))
     (g1 (z1) id (aa))
     (g2 (z2) id (aa))
     (g3 (z3) id (aa))
     (g4 (z4) id (aa)))
    nil)
```

DEFINITION:

fanout-5& (*netlist*)

```
= ((lookup-module ('fanout-5, netlist) = FANOUT-5*)
   ^ b-buf& (delete-module ('fanout-5, netlist))
   ^ id& (delete-module ('fanout-5, netlist)))
```

EVENT: Enable fanout-5&; name this event 'fanout-5&-off'.

DEFINITION:

FANOUT-5\$NETLIST = cons (FANOUT-5*, B-BUF\$NETLIST $\cup$ ID\$NETLIST)

THEOREM: fanout-5\$value

fanout-5& (*netlist*)

```
→ (dual-eval (0, 'fanout-5, list (a), state, netlist)
   = make-list (5, threefix (a)))
```

EVENT: Enable fanout-5\$value; name this event 'fanout-5\$value-off'.

DEFINITION:

FANOUT-32*

```
= ' (fanout-32
    (a)
    (s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14
     s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25 s26 s27
     s28 s29 s30 s31)
    ((ga (aa) b-buf-pwr (a))
     (g0 (s0) id (aa))
     (g1 (s1) id (aa))
     (g2 (s2) id (aa))
     (g3 (s3) id (aa))
     (g4 (s4) id (aa)))
```

```

(g5 (s5) id (aa))
(g6 (s6) id (aa))
(g7 (s7) id (aa))
(g8 (s8) id (aa))
(g9 (s9) id (aa))
(g10 (s10) id (aa))
(g11 (s11) id (aa))
(g12 (s12) id (aa))
(g13 (s13) id (aa))
(g14 (s14) id (aa))
(g15 (s15) id (aa))
(g16 (s16) id (aa))
(g17 (s17) id (aa))
(g18 (s18) id (aa))
(g19 (s19) id (aa))
(g20 (s20) id (aa))
(g21 (s21) id (aa))
(g22 (s22) id (aa))
(g23 (s23) id (aa))
(g24 (s24) id (aa))
(g25 (s25) id (aa))
(g26 (s26) id (aa))
(g27 (s27) id (aa))
(g28 (s28) id (aa))
(g29 (s29) id (aa))
(g30 (s30) id (aa))
(g31 (s31) id (aa))
nil)

```

DEFINITION:

```

fanout-32& (netlist)
= ((lookup-module ('fanout-32, netlist) = FANOUT-32*)
   ∧ b-buf-pwr& (delete-module ('fanout-32, netlist))
   ∧ id& (delete-module ('fanout-32, netlist)))

```

EVENT: Enable fanout-32&; name this event 'fanout-32&-off'.

DEFINITION:

```

FANOUT-32$NETLIST = cons (FANOUT-32*, B-BUF-PWR$NETLIST ∪ ID$NETLIST)

```

THEOREM: fanout-32\$value

```

fanout-32& (netlist)
→ (dual-eval (0, 'fanout-32, list (a), state, netlist)
   = make-list (32, threefix (a)))

```

EVENT: Enable fanout-32\$value; name this event 'fanout-32\$value-off'.

DEFINITION:

f\$decode-5 (s)

```
= list (f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
                  f-nand3 (f-not (caddr (s)), f-not (cadddr (s)), f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
                  f-nand3 (f-not (caddr (s)), f-not (cadddr (s)), f-not (caddddr (s)))),
        f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (caddr (s)), f-not (cadddr (s)), f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (caddr (s)), f-not (cadddr (s)), f-not (caddddr (s)))),
        f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
                  f-nand3 (f-not (f-not (caddr (s))),
                          f-not (cadddr (s)),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
                  f-nand3 (f-not (f-not (caddr (s))),
                          f-not (cadddr (s)),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (f-not (caddr (s))),
                          f-not (cadddr (s)),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (f-not (caddr (s))),
                          f-not (cadddr (s)),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
                  f-nand3 (f-not (caddr (s)),
                          f-not (f-not (cadddr (s))),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
                  f-nand3 (f-not (caddr (s)),
                          f-not (f-not (cadddr (s))),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (caddr (s)),
                          f-not (f-not (cadddr (s))),
                          f-not (caddddr (s)))),
        f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
                  f-nand3 (f-not (caddr (s)),
                          f-not (f-not (cadddr (s))),
                          f-not (caddddr (s))))).
```

```
f-not (caddr (s))),
f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (f-not (caddr (s))),
                  f-not (caddr (s)))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (f-not (caddr (s))),
                  f-not (caddr (s)))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s))))),
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (f-not (caddr (s))),
                  f-not (caddr (s)))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s))))) ,
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (f-not (caddr (s))),
                  f-not (caddr (s)))),
f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
        f-nand3 (f-not (caddr (s)),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
        f-nand3 (f-not (caddr (s)),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s))))),
        f-nand3 (f-not (caddr (s)),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s))))) ,
        f-nand3 (f-not (caddr (s)),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
        f-nand3 (f-not (f-not (caddr (s))),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s))))),
        f-nand3 (f-not (caddr (s)),
                  f-not (caddr (s)),
                  f-not (f-not (caddr (s))))),
```

```

        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (f-not (caddr (s))),
        f-not (cadddr (s)),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (cadr (s))),
        f-nand3 (f-not (caddr (s)),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (cadr (s))),
        f-nand3 (f-not (caddr (s)),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (caddr (s)),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (caddr (s)),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (caddr (s)),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (f-not (caddr (s))),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (car (s)), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (f-not (caddr (s))),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))),
f-nor (f-nand (f-not (f-not (car (s))), f-not (f-not (cadr (s)))),
        f-nand3 (f-not (f-not (caddr (s))),
        f-not (f-not (cadddr (s))),
        f-not (f-not (caddddr (s))))))

```

EVENT: Enable f\$decode-5; name this event 'f\$decode-5-off'.

DEFINITION:

decode-5 (s)

```

= list (b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)), b-not (cadddr (s)), b-not (caddddr (s))))),

```

```

b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)), b-not (cadddr (s)), b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)), b-not (cadddr (s)), b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)), b-not (cadddr (s)), b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (cadddr (s)),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (cadddr (s)),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (cadddr (s)),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (cadddr (s)),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                    b-not (b-not (cadddr (s))),
                    b-not (caddddr (s))))

```

```

        b-not (b-not (caddr (s))),
        b-not (caddrdr (s))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (caddrdr (s)))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (caddrdr (s))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (caddrdr (s)),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),

```

```

        b-not (b-not (caddr (s))),
        b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (caddr (s)),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (caddr (s)),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (cadr (s))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (car (s)), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))),
b-nor (b-nand (b-not (b-not (car (s))), b-not (b-not (cadr (s)))),
        b-nand3 (b-not (b-not (caddr (s))),
                  b-not (b-not (caddrdr (s))),
                  b-not (b-not (caddrdr (s))))))

```

EVENT: Enable decode-5; name this event ‘decode-5-off’.

THEOREM: bvp-length-decode-5
 $\text{bvp}(\text{decode-5}(s)) \wedge (\text{length}(\text{decode-5}(s)) = 32)$

DEFINITION:

DECODE-5*

```

=  ' (decode-5
      (s0 s1 s2 s3 s4)
      (x00 x10 x20 x30 x01 x11 x21 x31 x02 x12 x22 x32 x03
       x13 x23 x33 x04 x14 x24 x34 x05 x15 x25 x35 x06 x16
       x26 x36 x07 x17 x27 x37)
      ((gs0- (s0-) b-not (s0)))

```



```

(gs1- (s1-) b-not (s1))
(gs2- (s2-) b-not (s2))
(gs3- (s3-) b-not (s3))
(gs4- (s4-) b-not (s4))
(gs0 (bs0) b-not (s0-))
(gs1 (bs1) b-not (s1-))
(gs2 (bs2) b-not (s2-))
(gs3 (bs3) b-not (s3-))
(gs4 (bs4) b-not (s4-))
(gl0 (l0) b-nand (s0- s1-))
(gl1 (l1) b-nand (bs0 s1-))
(gl2 (l2) b-nand (s0- bs1))
(gl3 (l3) b-nand (bs0 bs1))
(gh0 (h0) b-nand3 (s2- s3- s4-))
(gh1 (h1) b-nand3 (bs2 s3- s4-))
(gh2 (h2) b-nand3 (s2- bs3 s4-))
(gh3 (h3) b-nand3 (bs2 bs3 s4-))
(gh4 (h4) b-nand3 (s2- s3- bs4))
(gh5 (h5) b-nand3 (bs2 s3- bs4))
(gh6 (h6) b-nand3 (s2- bs3 bs4))
(gh7 (h7) b-nand3 (bs2 bs3 bs4))
(gx00 (x00) b-nor (l0 h0))
(gx10 (x10) b-nor (l1 h0))
(gx20 (x20) b-nor (l2 h0))
(gx30 (x30) b-nor (l3 h0))
(gx01 (x01) b-nor (l0 h1))
(gx11 (x11) b-nor (l1 h1))
(gx21 (x21) b-nor (l2 h1))
(gx31 (x31) b-nor (l3 h1))
(gx02 (x02) b-nor (l0 h2))
(gx12 (x12) b-nor (l1 h2))
(gx22 (x22) b-nor (l2 h2))
(gx32 (x32) b-nor (l3 h2))
(gx03 (x03) b-nor (l0 h3))
(gx13 (x13) b-nor (l1 h3))
(gx23 (x23) b-nor (l2 h3))
(gx33 (x33) b-nor (l3 h3))
(gx04 (x04) b-nor (l0 h4))
(gx14 (x14) b-nor (l1 h4))
(gx24 (x24) b-nor (l2 h4))
(gx34 (x34) b-nor (l3 h4))
(gx05 (x05) b-nor (l0 h5))
(gx15 (x15) b-nor (l1 h5))
(gx25 (x25) b-nor (l2 h5))

```

```

(gx35 (x35) b-nor (13 h5))
(gx06 (x06) b-nor (10 h6))
(gx16 (x16) b-nor (11 h6))
(gx26 (x26) b-nor (12 h6))
(gx36 (x36) b-nor (13 h6))
(gx07 (x07) b-nor (10 h7))
(gx17 (x17) b-nor (11 h7))
(gx27 (x27) b-nor (12 h7))
(gx37 (x37) b-nor (13 h7))
nil)

```

DEFINITION:

decode-5& (*netlist*)

```

= ((lookup-module ('decode-5, netlist) = DECODE-5*)
   ∧ b-not& (delete-module ('decode-5, netlist))
   ∧ b-nand& (delete-module ('decode-5, netlist))
   ∧ b-nand3& (delete-module ('decode-5, netlist))
   ∧ b-nor& (delete-module ('decode-5, netlist)))

```

EVENT: Enable decode-5&; name this event 'decode-5&-off'.

DEFINITION:

DECODE-5\$NETLIST

```

= cons (DECODE-5*,
        ((B-NOT$NETLIST ∪ B-NAND$NETLIST) ∪ B-NAND3$NETLIST)
        ∪ B-NOR$NETLIST)

```

THEOREM: properp-length-f\$decode-5

properp (f\$decode-5 (*s*)) ∧ (length (f\$decode-5 (*s*)) = 32)

THEOREM: decode-5\$value

(decode-5& (*netlist*) ∧ (length (*s*) = 5))
→ (dual-eval (0, 'decode-5, *s*, *state*, *netlist*) = f\$decode-5 (*s*))

THEOREM: f\$decode-5=decode-5

(bvp (*s*) ∧ (length (*s*) = 5)) → (f\$decode-5 (*s*) = decode-5 (*s*))

EVENT: Enable decode-5\$value; name this event 'decode-5\$value-off'.

DEFINITION:

```

encode-32 (s0,
           s1,
           s2,
           s3,

```

```

s4,
s5,
s6,
s7,
s8,
s9,
s10,
s11,
s12,
s13,
s14,
s15,
s16,
s17,
s18,
s19,
s20,
s21,
s22,
s23,
s24,
s25,
s26,
s27,
s28,
s29,
s30,
s31)
= list (b-nand4 (b-nor4 (s1, s3, s5, s7),
                      b-nor4 (s9, s11, s13, s15),
                      b-nor4 (s17, s19, s21, s23),
                      b-nor (s25, s29)),
      b-nand4 (b-nor4 (s2, s3, s6, s7),
              b-nor4 (s10, s11, s14, s15),
              b-nor4 (s18, s19, s22, s23),
              b-not (s30)),
      b-nand4 (b-nor4 (s4, s5, s6, s7),
              b-nor4 (s12, s13, s14, s15),
              b-nor4 (s20, s21, s22, s23),
              b-nor3 (s28, s29, s30)),
      b-nand4 (b-nor4 (s8, s9, s10, s11),
              b-nor4 (s12, s13, s14, s15),
              b-nor (s24, s25),
              b-nor3 (s28, s29, s30)),

```

```

b-nand4 (b-nor4 (s16, s17, s18, s19),
               b-nor4 (s20, s21, s22, s23),
               b-nor (s24, s25),
               b-nor3 (s28, s29, s30)))

```

EVENT: Enable encode-32; name this event ‘encode-32-off’.

DEFINITION:

```

f$encode-32 (s0,
             s1,
             s2,
             s3,
             s4,
             s5,
             s6,
             s7,
             s8,
             s9,
             s10,
             s11,
             s12,
             s13,
             s14,
             s15,
             s16,
             s17,
             s18,
             s19,
             s20,
             s21,
             s22,
             s23,
             s24,
             s25,
             s26,
             s27,
             s28,
             s29,
             s30,
             s31)
= list (f-nand4 (f-nor4 (s1, s3, s5, s7),
                  f-nor4 (s9, s11, s13, s15),
                  f-nor4 (s17, s19, s21, s23),
                  f-nor (s25, s29)),

```

```

f-nand4 (f-nor4 (s2, s3, s6, s7),
              f-nor4 (s10, s11, s14, s15),
              f-nor4 (s18, s19, s22, s23),
              f-not (s30)),
f-nand4 (f-nor4 (s4, s5, s6, s7),
              f-nor4 (s12, s13, s14, s15),
              f-nor4 (s20, s21, s22, s23),
              f-nor3 (s28, s29, s30)),
f-nand4 (f-nor4 (s8, s9, s10, s11),
              f-nor4 (s12, s13, s14, s15),
              f-nor (s24, s25),
              f-nor3 (s28, s29, s30)),
f-nand4 (f-nor4 (s16, s17, s18, s19),
              f-nor4 (s20, s21, s22, s23),
              f-nor (s24, s25),
              f-nor3 (s28, s29, s30)))

```

EVENT: Enable f\$encode-32; name this event 'f\$encode-32-off'.

DEFINITION:

ENCODE-32*

```

=      '(encode-32
      (s0 s1 s2 s3 s4 s5 s6 s7 s8 s9 s10 s11 s12 s13 s14
       s15 s16 s17 s18 s19 s20 s21 s22 s23 s24 s25 s26 s27
       s28 s29 s30 s31)
      (w-4 w-9 w-14 w-19 w-24)
      ((g-23 (w-23) b-nor3 (s28 s29 s30))
       (g-22 (w-22) b-nor (s24 s25))
       (g-21 (w-21) b-nor4 (s20 s21 s22 s23))
       (g-20 (w-20) b-nor4 (s16 s17 s18 s19))
       (g-24 (w-24) b-nand4 (w-20 w-21 w-22 w-23))
       (g-16 (w-16) b-nor4 (s12 s13 s14 s15))
       (g-15 (w-15) b-nor4 (s8 s9 s10 s11))
       (g-19 (w-19) b-nand4 (w-15 w-16 w-22 w-23))
       (g-10 (w-10) b-nor4 (s4 s5 s6 s7))
       (g-14 (w-14) b-nand4 (w-10 w-16 w-21 w-23))
       (g-8 (w-8) b-not (s30))
       (g-7 (w-7) b-nor4 (s18 s19 s22 s23))
       (g-6 (w-6) b-nor4 (s10 s11 s14 s15))
       (g-5 (w-5) b-nor4 (s2 s3 s6 s7))
       (g-9 (w-9) b-nand4 (w-5 w-6 w-7 w-8))
       (g-3 (w-3) b-nor (s25 s29))
       (g-2 (w-2) b-nor4 (s17 s19 s21 s23))
       (g-1 (w-1) b-nor4 (s9 s11 s13 s15))

```

```

      (g-0 (w-0) b-nor4 (s1 s3 s5 s7))
      (g-4 (w-4) b-nand4 (w-0 w-1 w-2 w-3)))
    nil)

```

DEFINITION:

```

encode-32& (netlist)
= ((lookup-module ('encode-32, netlist) = ENCODE-32*)
   ∧ b-nor3& (delete-module ('encode-32, netlist))
   ∧ b-not& (delete-module ('encode-32, netlist))
   ∧ b-nor& (delete-module ('encode-32, netlist))
   ∧ b-nor4& (delete-module ('encode-32, netlist))
   ∧ b-nand4& (delete-module ('encode-32, netlist)))

```

EVENT: Enable encode-32&; name this event 'encode-32&-off'.

DEFINITION:

```

ENCODE-32$NETLIST
= cons (ENCODE-32*,
        (((B-NOR3$NETLIST ∪ B-NOT$NETLIST) ∪ B-NOR$NETLIST)
         ∪ B-NOR4$NETLIST)
         ∪ B-NAND4$NETLIST)

```

THEOREM: encode-32\$value

```

encode-32& (netlist)
→ (dual-eval (0,
               'encode-32,
               list (s0,
                     s1,
                     s2,
                     s3,
                     s4,
                     s5,
                     s6,
                     s7,
                     s8,
                     s9,
                     s10,
                     s11,
                     s12,
                     s13,
                     s14,
                     s15,
                     s16,
                     s17,

```

```

        s18,
        s19,
        s20,
        s21,
        s22,
        s23,
        s24,
        s25,
        s26,
        s27,
        s28,
        s29,
        s30,
        s31),
    state,
    netlist)
= f$encode-32(s0,
    s1,
    s2,
    s3,
    s4,
    s5,
    s6,
    s7,
    s8,
    s9,
    s10,
    s11,
    s12,
    s13,
    s14,
    s15,
    s16,
    s17,
    s18,
    s19,
    s20,
    s21,
    s22,
    s23,
    s24,
    s25,
    s26,
    s27,

```

$s28,$
 $s29,$
 $s30,$
 $s31))$

EVENT: Enable encode-32\$value; name this event 'encode-32\$value-off'.

THEOREM: f\$encode-32=encode-32

$(\text{boolp}(s0)$
 $\wedge \text{boolp}(s1)$
 $\wedge \text{boolp}(s2)$
 $\wedge \text{boolp}(s3)$
 $\wedge \text{boolp}(s4)$
 $\wedge \text{boolp}(s5)$
 $\wedge \text{boolp}(s6)$
 $\wedge \text{boolp}(s7)$
 $\wedge \text{boolp}(s8)$
 $\wedge \text{boolp}(s9)$
 $\wedge \text{boolp}(s10)$
 $\wedge \text{boolp}(s11)$
 $\wedge \text{boolp}(s12)$
 $\wedge \text{boolp}(s13)$
 $\wedge \text{boolp}(s14)$
 $\wedge \text{boolp}(s15)$
 $\wedge \text{boolp}(s16)$
 $\wedge \text{boolp}(s17)$
 $\wedge \text{boolp}(s18)$
 $\wedge \text{boolp}(s19)$
 $\wedge \text{boolp}(s20)$
 $\wedge \text{boolp}(s21)$
 $\wedge \text{boolp}(s22)$
 $\wedge \text{boolp}(s23)$
 $\wedge \text{boolp}(s24)$
 $\wedge \text{boolp}(s25)$
 $\wedge \text{boolp}(s26)$
 $\wedge \text{boolp}(s27)$
 $\wedge \text{boolp}(s28)$
 $\wedge \text{boolp}(s29)$
 $\wedge \text{boolp}(s30)$
 $\wedge \text{boolp}(s31))$
 $\rightarrow (\text{f\$encode-32}(s0,$
 $s1,$
 $s2,$
 $s3,$


```

s4,
s5,
s6,
s7,
s8,
s9,
s10,
s11,
s12,
s13,
s14,
s15,
s16,
s17,
s18,
s19,
s20,
s21,
s22,
s23,
s24,
s25,
s26,
s27,
s28,
s29,
s30,
s31)
= encode-32(s0,
s1,
s2,
s3,
s4,
s5,
s6,
s7,
s8,
s9,
s10,
s11,
s12,
s13,
s14,
s15,

```

$s16,$
 $s17,$
 $s18,$
 $s19,$
 $s20,$
 $s21,$
 $s22,$
 $s23,$
 $s24,$
 $s25,$
 $s26,$
 $s27,$
 $s28,$
 $s29,$
 $s30,$
 $s31))$

THEOREM: bvp-length-encode-32

bvp (encode-32 ($s0,$
 $s1,$
 $s2,$
 $s3,$
 $s4,$
 $s5,$
 $s6,$
 $s7,$
 $s8,$
 $s9,$
 $s10,$
 $s11,$
 $s12,$
 $s13,$
 $s14,$
 $s15,$
 $s16,$
 $s17,$
 $s18,$
 $s19,$
 $s20,$
 $s21,$
 $s22,$
 $s23,$
 $s24,$
 $s25,$

$$\begin{aligned}
& s26, \\
& s27, \\
& s28, \\
& s29, \\
& s30, \\
& s31)) \\
\wedge \quad & (\text{length}(\text{encode-32}(s0, \\
& s1, \\
& s2, \\
& s3, \\
& s4, \\
& s5, \\
& s6, \\
& s7, \\
& s8, \\
& s9, \\
& s10, \\
& s11, \\
& s12, \\
& s13, \\
& s14, \\
& s15, \\
& s16, \\
& s17, \\
& s18, \\
& s19, \\
& s20, \\
& s21, \\
& s22, \\
& s23, \\
& s24, \\
& s25, \\
& s26, \\
& s27, \\
& s28, \\
& s29, \\
& s30, \\
& s31))) \\
& = \quad 5)
\end{aligned}$$

THEOREM: properp-length-f\$encode-32
properp (f\$encode-32 (s0,
s1,
s2,

```

s3,
s4,
s5,
s6,
s7,
s8,
s9,
s10,
s11,
s12,
s13,
s14,
s15,
s16,
s17,
s18,
s19,
s20,
s21,
s22,
s23,
s24,
s25,
s26,
s27,
s28,
s29,
s30,
s31))
^ (length (f$encode-32 (s0,
s1,
s2,
s3,
s4,
s5,
s6,
s7,
s8,
s9,
s10,
s11,
s12,
s13,
s14,

```

$s15,$
 $s16,$
 $s17,$
 $s18,$
 $s19,$
 $s20,$
 $s21,$
 $s22,$
 $s23,$
 $s24,$
 $s25,$
 $s26,$
 $s27,$
 $s28,$
 $s29,$
 $s30,$
 $s31))$
 $= 5)$

THEOREM: dual-eval-on-collected-nth-32
 $((\text{length}(args) = 32) \wedge \text{properp}(args))$
 $\rightarrow (\text{dual-eval}(flag, name, args, state, netlist)$
 $= \text{dual-eval}(flag,$
 $name,$
 $\text{list}(\text{nth}(0, args),$
 $\text{nth}(1, args),$
 $\text{nth}(2, args),$
 $\text{nth}(3, args),$
 $\text{nth}(4, args),$
 $\text{nth}(5, args),$
 $\text{nth}(6, args),$
 $\text{nth}(7, args),$
 $\text{nth}(8, args),$
 $\text{nth}(9, args),$
 $\text{nth}(10, args),$
 $\text{nth}(11, args),$
 $\text{nth}(12, args),$
 $\text{nth}(13, args),$
 $\text{nth}(14, args),$
 $\text{nth}(15, args),$
 $\text{nth}(16, args),$
 $\text{nth}(17, args),$
 $\text{nth}(18, args),$
 $\text{nth}(19, args),$

$\text{nth}(20, \text{args}),$
 $\text{nth}(21, \text{args}),$
 $\text{nth}(22, \text{args}),$
 $\text{nth}(23, \text{args}),$
 $\text{nth}(24, \text{args}),$
 $\text{nth}(25, \text{args}),$
 $\text{nth}(26, \text{args}),$
 $\text{nth}(27, \text{args}),$
 $\text{nth}(28, \text{args}),$
 $\text{nth}(29, \text{args}),$
 $\text{nth}(30, \text{args}),$
 $\text{nth}(31, \text{args})),$
 $\text{state},$
 $\text{netlist}))$

THEOREM: $\text{encode-32}\$ \text{value-on-a-vector}$
 $(\text{encode-32}\& (\text{netlist}) \wedge \text{properp}(\text{args}) \wedge (\text{length}(\text{args}) = 32))$
 $\rightarrow (\text{dual-eval}(0, \text{'encode-32}, \text{args}, \text{state}, \text{netlist})$
 $= \text{f}\$ \text{encode-32}(\text{nth}(0, \text{args}),$
 $\text{nth}(1, \text{args}),$
 $\text{nth}(2, \text{args}),$
 $\text{nth}(3, \text{args}),$
 $\text{nth}(4, \text{args}),$
 $\text{nth}(5, \text{args}),$
 $\text{nth}(6, \text{args}),$
 $\text{nth}(7, \text{args}),$
 $\text{nth}(8, \text{args}),$
 $\text{nth}(9, \text{args}),$
 $\text{nth}(10, \text{args}),$
 $\text{nth}(11, \text{args}),$
 $\text{nth}(12, \text{args}),$
 $\text{nth}(13, \text{args}),$
 $\text{nth}(14, \text{args}),$
 $\text{nth}(15, \text{args}),$
 $\text{nth}(16, \text{args}),$
 $\text{nth}(17, \text{args}),$
 $\text{nth}(18, \text{args}),$
 $\text{nth}(19, \text{args}),$
 $\text{nth}(20, \text{args}),$
 $\text{nth}(21, \text{args}),$
 $\text{nth}(22, \text{args}),$
 $\text{nth}(23, \text{args}),$
 $\text{nth}(24, \text{args}),$
 $\text{nth}(25, \text{args}),$

$$\begin{aligned} & \text{nth}(26, \text{args}), \\ & \text{nth}(27, \text{args}), \\ & \text{nth}(28, \text{args}), \\ & \text{nth}(29, \text{args}), \\ & \text{nth}(30, \text{args}), \\ & \text{nth}(31, \text{args})) \end{aligned}$$

EVENT: Enable encode-32\$value-on-a-vector; name this event ‘encode-32\$value-on-a-vector-off’.

DEFINITION:

$$\begin{aligned} & \text{set-some-flags}(\text{set-flags}) \\ = & \quad (\text{z-set}(\text{set-flags}) \\ & \quad \vee \quad \text{n-set}(\text{set-flags}) \\ & \quad \vee \quad \text{v-set}(\text{set-flags}) \\ & \quad \vee \quad \text{c-set}(\text{set-flags})) \end{aligned}$$

EVENT: Enable set-some-flags; name this event ‘set-some-flags-off’.

THEOREM: not-set-some-flags-make-list-4-f
 $\neg \text{set-some-flags}(\text{make-list}(4, \mathbf{f}))$

DEFINITION:

$$\text{flags-hyps}(\text{flags}) = ((\text{length}(\text{flags}) = 4) \wedge \text{bvp}(\text{flags}))$$

THEOREM: not-set-some-flags-update-flags
 $(\text{flags-hyps}(\text{flags}) \wedge (\neg \text{set-some-flags}(\text{set-flags})))$
 $\rightarrow \quad (\text{update-flags}(\text{flags}, \text{set-flags}, \text{cvzbv}) = \text{flags})$

THEOREM: boolp-c-flag
 $\text{flags-hyps}(\text{flags}) \rightarrow \text{boolp}(\text{c-flag}(\text{flags}))$

DEFINITION:

$$\begin{aligned} & \text{f\$set-some-flags}(\text{set-flags}) \\ = & \quad \text{f-or4}(\text{car}(\text{set-flags}), \\ & \quad \text{cadr}(\text{set-flags}), \\ & \quad \text{caddr}(\text{set-flags}), \\ & \quad \text{cadddr}(\text{set-flags})) \end{aligned}$$

EVENT: Enable f\$set-some-flags; name this event ‘f\$set-some-flags-off’.

THEOREM: f\$set-some-flags=set-some=flags
 $(\text{bvp}(\text{set-flags}) \wedge (\text{length}(\text{set-flags}) = 4))$
 $\rightarrow \quad (\text{f\$set-some-flags}(\text{set-flags}) = \text{set-some-flags}(\text{set-flags}))$

THEOREM: b-or4\$value-as-f\$set-some-flags
 $(b\text{-or}4 \& (netlist) \wedge \text{properp}(set\text{-}flags) \wedge (\text{length}(set\text{-}flags) = 4))$
 $\rightarrow (\text{dual-eval}(0, 'b\text{-or}4, set\text{-}flags, state, netlist)$
 $= \text{list}(f\$set\text{-}some\text{-}flags(set\text{-}flags)))$

EVENT: Enable b-or4\$value-as-f\$set-some-flags; name this event 'b-or4\$value-as-f\$set-some-flags-off'.

DEFINITION:
 $f\$all\text{-}t\text{-}regs\text{-}address(regs\text{-}address)$
 $= f\text{-and}4(car(regs\text{-}address),$
 $\quad \quad \quad cadr(regs\text{-}address),$
 $\quad \quad \quad caddr(regs\text{-}address),$
 $\quad \quad \quad caddr(regs\text{-}address))$

EVENT: Enable f\$all-t-regs-address; name this event 'f\$all-t-regs-address-off'.

THEOREM: f\$all-t-regs-address=set-some=flags
 $(bvp(regs\text{-}address) \wedge (\text{length}(regs\text{-}address) = 4))$
 $\rightarrow (f\$all\text{-}t\text{-}regs\text{-}address(regs\text{-}address)$
 $= (regs\text{-}address = \text{make-list}(4, t)))$

THEOREM: b-and4\$value-as-f\$all-t-regs-address
 $(b\text{-and}4 \& (netlist)$
 $\wedge \text{properp}(regs\text{-}address)$
 $\wedge (\text{length}(regs\text{-}address) = 4))$
 $\rightarrow (\text{dual-eval}(0, 'b\text{-and}4, regs\text{-}address, state, netlist)$
 $= \text{list}(f\$all\text{-}t\text{-}regs\text{-}address(regs\text{-}address)))$

EVENT: Enable b-and4\$value-as-f\$all-t-regs-address; name this event 'b-and4\$value-as-f\$all-t-regs-address-off'.

DEFINITION:
 $cv\text{-}hyps(rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg)$
 $= (\text{boolp}(rw-)$
 $\quad \wedge (\text{length}(regs\text{-}address) = 4)$
 $\quad \wedge bvp(regs\text{-}address)$
 $\quad \wedge (\text{length}(i\text{-}reg) = 32)$
 $\quad \wedge bvp(i\text{-}reg)$
 $\quad \wedge \text{flags-hyps}(flags)$
 $\quad \wedge (\text{length}(pc\text{-}reg) = 4)$
 $\quad \wedge bvp(pc\text{-}reg))$

DEFINITION: $v\_FETCH0 = \text{list}(f, f, f, f, f)$

EVENT: Enable v_fetch0 ; name this event ' v_fetch0 -off'.

EVENT: Enable $*1*v\_fetch0$; name this event ' $g*1*v\_fetch0$ -off'.

DEFINITION: $N\_FETCH0 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_fetch0 ; name this event ' n_fetch0 -off'.

EVENT: Enable $*1*n\_fetch0$; name this event ' $g*1*n\_fetch0$ -off'.

DEFINITION: $V\_FETCH1 = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_fetch1 ; name this event ' v_fetch1 -off'.

EVENT: Enable $*1*v\_fetch1$; name this event ' $g*1*v\_fetch1$ -off'.

DEFINITION: $N\_FETCH1 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_fetch1 ; name this event ' n_fetch1 -off'.

EVENT: Enable $*1*n\_fetch1$; name this event ' $g*1*n\_fetch1$ -off'.

DEFINITION: $V\_FETCH2 = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_fetch2 ; name this event ' v_fetch2 -off'.

EVENT: Enable $*1*v\_fetch2$; name this event ' $g*1*v\_fetch2$ -off'.

DEFINITION: $N\_FETCH2 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_fetch2 ; name this event ' n_fetch2 -off'.

EVENT: Enable $*1*n\_fetch2$; name this event ' $g*1*n\_fetch2$ -off'.

DEFINITION: $V\_FETCH3 = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_fetch3 ; name this event ' v_fetch3 -off'.

EVENT: Enable $*1*v\_fetch3$; name this event ' $g*1*v\_fetch3$ -off'.

DEFINITION: $\text{N\_FETCH3} = \text{v-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_fetch3 ; name this event ‘ n_fetch3-off ’.

EVENT: Enable $*1*\text{n\_fetch3}$; name this event ‘ $\text{g}*1*\text{n\_fetch3-off}$ ’.

DEFINITION: $\text{V\_DECODE} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_decode ; name this event ‘ v_decode-off ’.

EVENT: Enable $*1*\text{v\_decode}$; name this event ‘ $\text{g}*1*\text{v\_decode-off}$ ’.

DEFINITION: $\text{N\_DECODE} = \text{v-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_decode ; name this event ‘ n_decode-off ’.

EVENT: Enable $*1*\text{n\_decode}$; name this event ‘ $\text{g}*1*\text{n\_decode-off}$ ’.

DEFINITION: $\text{V\_REGA} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_rega ; name this event ‘ v_rega-off ’.

EVENT: Enable $*1*\text{v\_rega}$; name this event ‘ $\text{g}*1*\text{v\_rega-off}$ ’.

DEFINITION: $\text{N\_REGA} = \text{v-to-nat}(\text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_rega ; name this event ‘ n_rega-off ’.

EVENT: Enable $*1*\text{n\_rega}$; name this event ‘ $\text{g}*1*\text{n\_rega-off}$ ’.

DEFINITION: $\text{V\_REGB} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

EVENT: Enable v_regb ; name this event ‘ v_regb-off ’.

EVENT: Enable $*1*\text{v\_regb}$; name this event ‘ $\text{g}*1*\text{v\_regb-off}$ ’.

DEFINITION: $\text{N\_REGB} = \text{v-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}))$

EVENT: Enable n_regb ; name this event ' $n\_regb\text{-off}$ '.

EVENT: Enable $*1*n\_regb$; name this event ' $g*1*n\_regb\text{-off}$ '.

DEFINITION: $v\_update = \text{list}(t, t, t, f, f)$

EVENT: Enable v_update ; name this event ' $v\_update\text{-off}$ '.

EVENT: Enable $*1*v\_update$; name this event ' $g*1*v\_update\text{-off}$ '.

DEFINITION: $n\_update = v\text{-to-nat}(\text{list}(t, t, t, f, f))$

EVENT: Enable n_update ; name this event ' $n\_update\text{-off}$ '.

EVENT: Enable $*1*n\_update$; name this event ' $g*1*n\_update\text{-off}$ '.

DEFINITION: $v\_reada0 = \text{list}(f, f, f, t, f)$

EVENT: Enable v_reada0 ; name this event ' $v\_reada0\text{-off}$ '.

EVENT: Enable $*1*v\_reada0$; name this event ' $g*1*v\_reada0\text{-off}$ '.

DEFINITION: $n\_reada0 = v\text{-to-nat}(\text{list}(f, f, f, t, f))$

EVENT: Enable n_reada0 ; name this event ' $n\_reada0\text{-off}$ '.

EVENT: Enable $*1*n\_reada0$; name this event ' $g*1*n\_reada0\text{-off}$ '.

DEFINITION: $v\_reada1 = \text{list}(t, f, f, t, f)$

EVENT: Enable v_reada1 ; name this event ' $v\_reada1\text{-off}$ '.

EVENT: Enable $*1*v\_reada1$; name this event ' $g*1*v\_reada1\text{-off}$ '.

DEFINITION: $n\_reada1 = v\text{-to-nat}(\text{list}(t, f, f, t, f))$

EVENT: Enable n_reada1 ; name this event ' $n\_reada1\text{-off}$ '.

EVENT: Enable $*1*n\_reada1$; name this event ' $g*1*n\_reada1\text{-off}$ '.

DEFINITION: $v\_reada2 = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

EVENT: Enable v_reada2 ; name this event ' $v\_reada2\text{-off}$ '.

EVENT: Enable $*1*v\_reada2$; name this event ' $g*1*v\_reada2\text{-off}$ '.

DEFINITION: $n\_reada2 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$

EVENT: Enable n_reada2 ; name this event ' $n\_reada2\text{-off}$ '.

EVENT: Enable $*1*n\_reada2$; name this event ' $g*1*n\_reada2\text{-off}$ '.

DEFINITION: $v\_reada3 = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

EVENT: Enable v_reada3 ; name this event ' $v\_reada3\text{-off}$ '.

EVENT: Enable $*1*v\_reada3$; name this event ' $g*1*v\_reada3\text{-off}$ '.

DEFINITION: $n\_reada3 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$

EVENT: Enable n_reada3 ; name this event ' $n\_reada3\text{-off}$ '.

EVENT: Enable $*1*n\_reada3$; name this event ' $g*1*n\_reada3\text{-off}$ '.

DEFINITION: $v\_readb0 = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

EVENT: Enable v_readb0 ; name this event ' $v\_readb0\text{-off}$ '.

EVENT: Enable $*1*v\_readb0$; name this event ' $g*1*v\_readb0\text{-off}$ '.

DEFINITION: $n\_readb0 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}))$

EVENT: Enable n_readb0 ; name this event ' $n\_readb0\text{-off}$ '.

EVENT: Enable $*1*n\_readb0$; name this event ' $g*1*n\_readb0\text{-off}$ '.

DEFINITION: $v\_readb1 = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

EVENT: Enable `v_readb1`; name this event '`v_readb1-off`'.

EVENT: Enable `*1*v_readb1`; name this event '`g*1*v_readb1-off`'.

DEFINITION: `N_READB1` = `v-to-nat` (`list` (`t`, `f`, `t`, `t`, `f`))

EVENT: Enable `n_readb1`; name this event '`n_readb1-off`'.

EVENT: Enable `*1*n_readb1`; name this event '`g*1*n_readb1-off`'.

DEFINITION: `V_READB2` = `list` (`f`, `t`, `t`, `t`, `f`)

EVENT: Enable `v_readb2`; name this event '`v_readb2-off`'.

EVENT: Enable `*1*v_readb2`; name this event '`g*1*v_readb2-off`'.

DEFINITION: `N_READB2` = `v-to-nat` (`list` (`f`, `t`, `t`, `t`, `f`))

EVENT: Enable `n_readb2`; name this event '`n_readb2-off`'.

EVENT: Enable `*1*n_readb2`; name this event '`g*1*n_readb2-off`'.

DEFINITION: `V_READB3` = `list` (`t`, `t`, `t`, `t`, `f`)

EVENT: Enable `v_readb3`; name this event '`v_readb3-off`'.

EVENT: Enable `*1*v_readb3`; name this event '`g*1*v_readb3-off`'.

DEFINITION: `N_READB3` = `v-to-nat` (`list` (`t`, `t`, `t`, `t`, `f`))

EVENT: Enable `n_readb3`; name this event '`n_readb3-off`'.

EVENT: Enable `*1*n_readb3`; name this event '`g*1*n_readb3-off`'.

DEFINITION: `V_WRITE0` = `list` (`f`, `f`, `f`, `f`, `t`)

EVENT: Enable `v_write0`; name this event '`v_write0-off`'.

EVENT: Enable `*1*v_write0`; name this event '`g*1*v_write0-off`'.

DEFINITION: $N\_WRITE0 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable n_write0 ; name this event ' $n\_write0\text{-off}$ '.

EVENT: Enable $*1*n\_write0$; name this event ' $g*1*n\_write0\text{-off}$ '.

DEFINITION: $V\_WRITE1 = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

EVENT: Enable v_write1 ; name this event ' $v\_write1\text{-off}$ '.

EVENT: Enable $*1*v\_write1$; name this event ' $g*1*v\_write1\text{-off}$ '.

DEFINITION: $N\_WRITE1 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable n_write1 ; name this event ' $n\_write1\text{-off}$ '.

EVENT: Enable $*1*n\_write1$; name this event ' $g*1*n\_write1\text{-off}$ '.

DEFINITION: $V\_WRITE2 = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

EVENT: Enable v_write2 ; name this event ' $v\_write2\text{-off}$ '.

EVENT: Enable $*1*v\_write2$; name this event ' $g*1*v\_write2\text{-off}$ '.

DEFINITION: $N\_WRITE2 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable n_write2 ; name this event ' $n\_write2\text{-off}$ '.

EVENT: Enable $*1*n\_write2$; name this event ' $g*1*n\_write2\text{-off}$ '.

DEFINITION: $V\_WRITE3 = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

EVENT: Enable v_write3 ; name this event ' $v\_write3\text{-off}$ '.

EVENT: Enable $*1*v\_write3$; name this event ' $g*1*v\_write3\text{-off}$ '.

DEFINITION: $N\_WRITE3 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable n_write3 ; name this event ' $n\_write3-off$ '.

EVENT: Enable $*1*n\_write3$; name this event ' $g*1*n\_write3-off$ '.

DEFINITION: $v\_sefa0 = list(f, f, t, f, t)$

EVENT: Enable v_sefa0 ; name this event ' $v\_sefa0-off$ '.

EVENT: Enable $*1*v\_sefa0$; name this event ' $g*1*v\_sefa0-off$ '.

DEFINITION: $n\_sefa0 = v-to-nat(list(f, f, t, f, t))$

EVENT: Enable n_sefa0 ; name this event ' $n\_sefa0-off$ '.

EVENT: Enable $*1*n\_sefa0$; name this event ' $g*1*n\_sefa0-off$ '.

DEFINITION: $v\_sefa1 = list(t, f, t, f, t)$

EVENT: Enable v_sefa1 ; name this event ' $v\_sefa1-off$ '.

EVENT: Enable $*1*v\_sefa1$; name this event ' $g*1*v\_sefa1-off$ '.

DEFINITION: $n\_sefa1 = v-to-nat(list(t, f, t, f, t))$

EVENT: Enable n_sefa1 ; name this event ' $n\_sefa1-off$ '.

EVENT: Enable $*1*n\_sefa1$; name this event ' $g*1*n\_sefa1-off$ '.

DEFINITION: $v\_sefb0 = list(f, t, t, f, t)$

EVENT: Enable v_sefb0 ; name this event ' $v\_sefb0-off$ '.

EVENT: Enable $*1*v\_sefb0$; name this event ' $g*1*v\_sefb0-off$ '.

DEFINITION: $n\_sefb0 = v-to-nat(list(f, t, t, f, t))$

EVENT: Enable n_sefb0 ; name this event ' $n\_sefb0-off$ '.

EVENT: Enable $*1*n\_sefb0$; name this event ' $g*1*n\_sefb0-off$ '.

DEFINITION: $v\_sefb1 = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})$

EVENT: Enable v_sefb1 ; name this event ' $v\_sefb1\text{-off}$ '.

EVENT: Enable $*1*v\_sefb1$; name this event ' $g*1*v\_sefb1\text{-off}$ '.

DEFINITION: $n\_sefb1 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}))$

EVENT: Enable n_sefb1 ; name this event ' $n\_sefb1\text{-off}$ '.

EVENT: Enable $*1*n\_sefb1$; name this event ' $g*1*n\_sefb1\text{-off}$ '.

DEFINITION: $v\_hold0 = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_hold0 ; name this event ' $v\_hold0\text{-off}$ '.

EVENT: Enable $*1*v\_hold0$; name this event ' $g*1*v\_hold0\text{-off}$ '.

DEFINITION: $n\_hold0 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_hold0 ; name this event ' $n\_hold0\text{-off}$ '.

EVENT: Enable $*1*n\_hold0$; name this event ' $g*1*n\_hold0\text{-off}$ '.

DEFINITION: $v\_hold1 = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_hold1 ; name this event ' $v\_hold1\text{-off}$ '.

EVENT: Enable $*1*v\_hold1$; name this event ' $g*1*v\_hold1\text{-off}$ '.

DEFINITION: $n\_hold1 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_hold1 ; name this event ' $n\_hold1\text{-off}$ '.

EVENT: Enable $*1*n\_hold1$; name this event ' $g*1*n\_hold1\text{-off}$ '.

DEFINITION: $v\_v11010 = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_v11010 ; name this event ' $v\_v11010\text{-off}$ '.

EVENT: Enable $*1*v\_v11010$; name this event ' $g*1*v\_v11010\text{-off}$ '.

DEFINITION: $n\_v11010 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_v11010 ; name this event ' $n\_v11010\text{-off}$ '.

EVENT: Enable $*1*n\_v11010$; name this event ' $g*1*n\_v11010\text{-off}$ '.

DEFINITION: $v\_v11011 = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_v11011 ; name this event ' $v\_v11011\text{-off}$ '.

EVENT: Enable $*1*v\_v11011$; name this event ' $g*1*v\_v11011\text{-off}$ '.

DEFINITION: $n\_v11011 = v\text{-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_v11011 ; name this event ' $n\_v11011\text{-off}$ '.

EVENT: Enable $*1*n\_v11011$; name this event ' $g*1*n\_v11011\text{-off}$ '.

DEFINITION: $v\_RESET0 = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_reset0 ; name this event ' $v\_reset0\text{-off}$ '.

EVENT: Enable $*1*v\_reset0$; name this event ' $g*1*v\_reset0\text{-off}$ '.

DEFINITION: $n\_RESET0 = v\text{-to-nat}(\text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_reset0 ; name this event ' $n\_reset0\text{-off}$ '.

EVENT: Enable $*1*n\_reset0$; name this event ' $g*1*n\_reset0\text{-off}$ '.

DEFINITION: $v\_RESET1 = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_reset1 ; name this event ' $v\_reset1\text{-off}$ '.

EVENT: Enable $*1*v\_reset1$; name this event ' $g*1*v\_reset1\text{-off}$ '.

DEFINITION: $\text{N\_RESET1} = \text{v-to-nat}(\text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_reset1 ; name this event ‘ n_reset1-off ’.

EVENT: Enable $*1*\text{n\_reset1}$; name this event ‘ $g*1*\text{n\_reset1-off}$ ’.

DEFINITION: $\text{V\_RESET2} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_reset2 ; name this event ‘ v_reset2-off ’.

EVENT: Enable $*1*\text{v\_reset2}$; name this event ‘ $g*1*\text{v\_reset2-off}$ ’.

DEFINITION: $\text{N\_RESET2} = \text{v-to-nat}(\text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_reset2 ; name this event ‘ n_reset2-off ’.

EVENT: Enable $*1*\text{n\_reset2}$; name this event ‘ $g*1*\text{n\_reset2-off}$ ’.

DEFINITION: $\text{V\_V11111} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Enable v_v11111 ; name this event ‘ v_v11111-off ’.

EVENT: Enable $*1*\text{v\_v11111}$; name this event ‘ $g*1*\text{v\_v11111-off}$ ’.

DEFINITION: $\text{N\_V11111} = \text{v-to-nat}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$

EVENT: Enable n_v11111 ; name this event ‘ n_v11111-off ’.

EVENT: Enable $*1*\text{n\_v11111}$; name this event ‘ $g*1*\text{n\_v11111-off}$ ’.

EVENT: Let us define the theory *vector-state-theory* to consist of the following events: v_fetch0 , v_fetch1 , v_fetch2 , v_fetch3 , v_decode , v_rega , v_regb , v_update , v_reada0 , v_reada1 , v_reada2 , v_reada3 , v_readb0 , v_readb1 , v_readb2 , v_readb3 , v_write0 , v_write1 , v_write2 , v_write3 , v_sefa0 , v_sefa1 , v_sefb0 , v_sefb1 , v_hold0 , v_hold1 , v_v11010 , v_v11011 , v_reset0 , v_reset1 , v_reset2 , v_v11111 .

EVENT: Let us define the theory *natural-state-theory* to consist of the following events: n_fetch0 , n_fetch1 , n_fetch2 , n_fetch3 , n_decode , n_rega , n_regb , n_update , n_reada0 , n_reada1 , n_reada2 , n_reada3 , n_readb0 , n_readb1 , n_readb2 ,

n_readb3, n_write0, n_write1, n_write2, n_write3, n_sefa0, n_sefa1, n_sefb0, n_sefb1,
n_hold0, n_hold1, n_v11010, n_v11011, n_reset0, n_reset1, n_reset2, n_v11111.

THEOREM: bvp-length-state-vectors

(length (V_FETCH0) = 5)
 $\wedge$ bvp (V_FETCH0)
 $\wedge$ (length (V_FETCH1) = 5)
 $\wedge$ bvp (V_FETCH1)
 $\wedge$ (length (V_FETCH2) = 5)
 $\wedge$ bvp (V_FETCH2)
 $\wedge$ (length (V_FETCH3) = 5)
 $\wedge$ bvp (V_FETCH3)
 $\wedge$ (length (V_DECODE) = 5)
 $\wedge$ bvp (V_DECODE)
 $\wedge$ (length (V_REGA) = 5)
 $\wedge$ bvp (V_REGA)
 $\wedge$ (length (V_REGB) = 5)
 $\wedge$ bvp (V_REGB)
 $\wedge$ (length (V_UPDATE) = 5)
 $\wedge$ bvp (V_UPDATE)
 $\wedge$ (length (V_READA0) = 5)
 $\wedge$ bvp (V_READA0)
 $\wedge$ (length (V_READA1) = 5)
 $\wedge$ bvp (V_READA1)
 $\wedge$ (length (V_READA2) = 5)
 $\wedge$ bvp (V_READA2)
 $\wedge$ (length (V_READA3) = 5)
 $\wedge$ bvp (V_READA3)
 $\wedge$ (length (V_READB0) = 5)
 $\wedge$ bvp (V_READB0)
 $\wedge$ (length (V_READB1) = 5)
 $\wedge$ bvp (V_READB1)
 $\wedge$ (length (V_READB2) = 5)
 $\wedge$ bvp (V_READB2)
 $\wedge$ (length (V_READB3) = 5)
 $\wedge$ bvp (V_READB3)
 $\wedge$ (length (V_WRITE0) = 5)
 $\wedge$ bvp (V_WRITE0)
 $\wedge$ (length (V_WRITE1) = 5)
 $\wedge$ bvp (V_WRITE1)
 $\wedge$ (length (V_WRITE2) = 5)
 $\wedge$ bvp (V_WRITE2)
 $\wedge$ (length (V_WRITE3) = 5)

```

^  bvp (V_WRITE3)
^  (length (V_SEFA0) = 5)
^  bvp (V_SEFA0)
^  (length (V_SEFA1) = 5)
^  bvp (V_SEFA1)
^  (length (V_SEFB0) = 5)
^  bvp (V_SEFB0)
^  (length (V_SEFB1) = 5)
^  bvp (V_SEFB1)
^  (length (V_HOLD0) = 5)
^  bvp (V_HOLD0)
^  (length (V_HOLD1) = 5)
^  bvp (V_HOLD1)
^  (length (V_V11010) = 5)
^  bvp (V_V11010)
^  (length (V_V11011) = 5)
^  bvp (V_V11011)
^  (length (V_RESET0) = 5)
^  bvp (V_RESET0)
^  (length (V_RESET1) = 5)
^  bvp (V_RESET1)
^  (length (V_RESET2) = 5)
^  bvp (V_RESET2)
^  (length (V_V11111) = 5)
^  bvp (V_V11111)

```

EVENT: Enable bvp-length-state-vectors; name this event ‘bvp-length-state-vectors-off’.

DEFINITION: $rw\text{-}(cntl\text{-}state) = nth(0, cntl\text{-}state)$

EVENT: Enable rw-; name this event ‘rw-off’.

DEFINITION: $strobe\text{-}(cntl\text{-}state) = nth(1, cntl\text{-}state)$

EVENT: Enable strobe-; name this event ‘strobe-off’.

DEFINITION: $hdack\text{-}(cntl\text{-}state) = nth(2, cntl\text{-}state)$

EVENT: Enable hdack-; name this event ‘hdack-off’.

DEFINITION: $we\text{-}regs(cntl\text{-}state) = nth(3, cntl\text{-}state)$

EVENT: Enable we-regs; name this event ‘we-regs-off’.

DEFINITION: $\text{we-a-reg}(cntl\text{-}state) = \text{nth}(4, cntl\text{-}state)$

EVENT: Enable we-a-reg; name this event ‘we-a-reg-off’.

DEFINITION: $\text{we-b-reg}(cntl\text{-}state) = \text{nth}(5, cntl\text{-}state)$

EVENT: Enable we-b-reg; name this event ‘we-b-reg-off’.

DEFINITION: $\text{we-i-reg}(cntl\text{-}state) = \text{nth}(6, cntl\text{-}state)$

EVENT: Enable we-i-reg; name this event ‘we-i-reg-off’.

DEFINITION: $\text{we-data-out}(cntl\text{-}state) = \text{nth}(7, cntl\text{-}state)$

EVENT: Enable we-data-out; name this event ‘we-data-out-off’.

DEFINITION: $\text{we-addr-out}(cntl\text{-}state) = \text{nth}(8, cntl\text{-}state)$

EVENT: Enable we-addr-out; name this event ‘we-addr-out-off’.

DEFINITION: $\text{we-hold-}(cntl\text{-}state) = \text{nth}(9, cntl\text{-}state)$

EVENT: Enable we-hold-; name this event ‘we-hold-off’.

DEFINITION: $\text{we-pc-reg}(cntl\text{-}state) = \text{nth}(10, cntl\text{-}state)$

EVENT: Enable we-pc-reg; name this event ‘we-pc-reg-off’.

DEFINITION: $\text{data-in-select}(cntl\text{-}state) = \text{nth}(11, cntl\text{-}state)$

EVENT: Enable data-in-select; name this event ‘data-in-select-off’.

DEFINITION: $\text{dec-addr-out}(cntl\text{-}state) = \text{nth}(12, cntl\text{-}state)$

EVENT: Enable dec-addr-out; name this event ‘dec-addr-out-off’.

DEFINITION: $\text{select-immediate}(cntl\text{-}state) = \text{nth}(13, cntl\text{-}state)$

EVENT: Enable select-immediate; name this event ‘select-immediate-off’.

DEFINITION: $\text{alu-c}(\text{cntl-state}) = \text{nth}(14, \text{cntl-state})$

EVENT: Enable alu-c; name this event ‘alu-c-off’.

DEFINITION: $\text{alu-zero}(\text{cntl-state}) = \text{nth}(15, \text{cntl-state})$

EVENT: Enable alu-zero; name this event ‘alu-zero-off’.

DEFINITION: $\text{state}(\text{cntl-state}) = \text{subrange}(\text{cntl-state}, 16, 20)$

EVENT: Enable state; name this event ‘state-off’.

DEFINITION: $\text{we-flags}(\text{cntl-state}) = \text{subrange}(\text{cntl-state}, 21, 24)$

EVENT: Enable we-flags; name this event ‘we-flags-off’.

DEFINITION:

$\text{regs-address}(\text{cntl-state}) = \text{subrange}(\text{cntl-state}, 25, 28)$

EVENT: Enable regs-address; name this event ‘regs-address-off’.

DEFINITION: $\text{alu-op}(\text{cntl-state}) = \text{subrange}(\text{cntl-state}, 29, 32)$

EVENT: Enable alu-op; name this event ‘alu-op-off’.

DEFINITION: $\text{alu-mpg}(\text{cntl-state}) = \text{subrange}(\text{cntl-state}, 33, 39)$

EVENT: Enable alu-mpg; name this event ‘alu-mpg-off’.

EVENT: Let us define the theory *control-state-accessor-theory* to consist of the following events: rw-, strobe-, hdack-, we-regs, we-a-reg, we-b-reg, we-i-reg, we-data-out, we-addr-out, we-hold-, we-pc-reg, data-in-select, dec-addr-out, select-immediate, alu-c, alu-zero, state, we-flags, regs-address, alu-op, alu-mpg.

```
(DEFN CV_FETCHO
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS RW-
    (CONS T
      (CONS T
```

```

(CONS F
  (CONS T
    (CONS F
      (CONS F
        (CONS F
          (CONS T
            (CONS T
              (CONS F
                (CONS F
                  (CONS F
                    (CONS
                      (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                            (CONS T (ALU-INC-OP))))
                      (CONS T (ALU-INC-OP))))
                    (CONS T
                      (APPEND
                        (V_FETCH0)
                        (APPEND
                          (MAKE-LIST 4 F)
                          (APPEND PC-REG
                                (APPEND (ALU-INC-OP)
                                          (APPEND (MPG (CONS T (CONS F (ALU-INC-OP))))
                                                    NIL))))))))))))))))))

```

EVENT: Enable cv_fetch0; name this event 'cv_fetch0-off'.

THEOREM: cv_fetch0\$destructure

cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)

$$\begin{aligned}
\rightarrow & ((rw- (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) = rw-) \\
& \wedge (strobe- (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{t}) \\
& \wedge (hdack- (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) = \mathbf{t}) \\
& \wedge (we-regs (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{f}) \\
& \wedge (we-a-reg (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{t}) \\
& \wedge (we-b-reg (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{f}) \\
& \wedge (we-i-reg (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{f}) \\
& \wedge (we-data-out (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{f}) \\
& \wedge (we-addr-out (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)) \\
& \quad = \mathbf{t})
\end{aligned}$$

```

^ (we-hold- (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = t)
^ (we-pc-reg (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (data-in-select (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (dec-addr-out (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (select-immediate (cv_fetch0 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
  = f)
^ (alu-c (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = carry-in-help (cons (c-flag (flags), cons (t, ALU-INC-OP))))
^ (alu-zero (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = t)
^ (state (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = V_FETCH0)
^ (we-flags (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = make-list (4, f))
^ (regs-address (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = pc-reg)
^ (alu-op (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = ALU-INC-OP)
^ (alu-mpg (cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))
  = mpg (cons (t, cons (f, ALU-INC-OP))))

```

EVENT: Enable cv_fetch0\$destructure; name this event 'cv_fetch0\$destructure-off'.

```

(DEFN CV_FETCH1
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS F
            (CONS F
              (CONS F
                (CONS F

```



```

(CONS F
  (CONS F
    (CONS F
      (CONS F
        (CONS F
          (CONS
            (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                (CONS F (OP-CODE I-REG))))
            (CONS F
              (CONS F
                (APPEND
                  (V_FETCH1)
                  (APPEND
                    (MAKE-LIST 4 F)
                    (APPEND PC-REG
                      (APPEND (OP-CODE I-REG)
                        (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                          NIL))))))))))))))))))

```

EVENT: Enable cv_fetch1; name this event 'cv_fetch1-off'.

THEOREM: cv_fetch1\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-i-reg (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-hold- (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (data-in-select (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ^ (dec-addr-out (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_fetch1 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                             cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_FETCH1)
  ^ (we-flags (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
  ^ (alu-op (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_fetch1\$destructure; name this event 'cv_fetch1\$destructure-off'.

```

(DEFN CV_FETCH2
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS F
      (CONS T
        (CONS T
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F

```

```

(CONS F
  (CONS F
    (CONS
      (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                           (CONS F (ALU-INC-OP))))
      (CONS F
        (APPEND
          (V_FETCH2)
          (APPEND
            (MAKE-LIST 4 F)
            (APPEND PC-REG
              (APPEND (ALU-INC-OP)
                (APPEND (MPG (CONS F (CONS F (ALU-INC-OP))))
                  NIL)))))))))))))

```

EVENT: Enable cv_fetch2; name this event 'cv_fetch2-off'.

THEOREM: cv_fetch2\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (hdack- (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-a-reg (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-i-reg (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-hold- (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (data-in-select (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (dec-addr-out (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (select-immediate (cv_fetch2 (rw-,

```

```

                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags), cons (f, ALU-INC-OP))))
  ^ (alu-zero (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_FETCH2)
  ^ (we-flags (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
  ^ (alu-op (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = ALU-INC-OP)
  ^ (alu-mpg (cv_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, ALU-INC-OP))))

```

EVENT: Enable cv_fetch2\$destructure; name this event 'cv_fetch2\$destructure-off'.

```

(DEFN CV_FETCH3
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS F
      (CONS T
        (CONS F
          (CONS F
            (CONS F
              (CONS T
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS
                        (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                              (CONS F (OP-CODE I-REG))))

```

```

(CONS F
 (APPEND
  (V_FETCH3)
  (APPEND
   (MAKE-LIST 4 F)
   (APPEND PC-REG
    (APPEND (OP-CODE I-REG)
     (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
      NIL))))))))))

```

EVENT: Enable `cv_fetch3`; name this event '`cv_fetch3-off`'.

THEOREM: `cv_fetch3$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))) = t)
  ∧ (strobe- (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (hdack- (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))) = t)
  ∧ (we-regs (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-a-reg (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-b-reg (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-i-reg (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = t)
  ∧ (we-data-out (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-addr-out (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-hold- (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (we-pc-reg (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (data-in-select (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = t)
  ∧ (dec-addr-out (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg)))
    = f)
  ∧ (select-immediate (cv_fetch3 (rw-,
                                   regs-address,
                                   i-reg,
                                   flags,
                                   pc-reg)))
    = f)

```

```

^ (alu-c (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = carry-in-help (cons (c-flag (flags),
                        cons (f, op-code (i-reg)))))
^ (alu-zero (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (state (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = V_FETCH3)
^ (we-flags (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = make-list (4, f))
^ (regs-address (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = pc-reg)
^ (alu-op (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = op-code (i-reg))
^ (alu-mpg (cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))
  = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_fetch3\$destructure; name this event 'cv_fetch3\$destructure-off'.

```

(DEFN CV_DECODE
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS
                        (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                              (CONS F (OP-CODE I-REG))))
                      (CONS F
                        (APPEND
                          (V_DECODE)
                          (APPEND

```

```

(MAKE-LIST 4 F)
(APPEND PC-REG
  (APPEND (OP-CODE I-REG)
    (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
      NIL))))))

```

EVENT: Enable cv_decode; name this event 'cv_decode-off'.

THEOREM: cv_decode\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_decode (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (strobe- (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = t)
  ∧ (hdack- (cv_decode (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (we-regs (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-a-reg (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-b-reg (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-i-reg (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-data-out (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-addr-out (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-hold- (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-pc-reg (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (data-in-select (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (dec-addr-out (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (select-immediate (cv_decode (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ∧ (alu-c (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ^ (state (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = V_DECODE)
  ^ (we-flags (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
  ^ (alu-op (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_decode (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg))))))

```

EVENT: Enable cv_decode\$destructure; name this event 'cv_decode\$destructure-off'.

```

(DEFN CV_REGA
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS T
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS
                        (A-IMMEDIATE-P I-REG)
                        (CONS
                          (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                (CONS F (OP-CODE I-REG))))
                          (CONS F
                            (APPEND
                              (V_REGA)
                              (APPEND
                                (MAKE-LIST 4 F)
                                (APPEND
                                  (RN-A I-REG)

```



```

(APPEND (OP-CODE I-REG)
  (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
    NIL)))))))))

```

EVENT: Enable `cv_rega`; name this event ‘`cv_rega-off`’.

THEOREM: `cv_rega$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (strobe- (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (hdack- (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (we-regs (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ∧ (we-a-reg (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (we-b-reg (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ∧ (we-i-reg (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ∧ (we-data-out (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-addr-out (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-hold- (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ∧ (we-pc-reg (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (data-in-select (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (dec-addr-out (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (select-immediate (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = a-immediate-p (i-reg))
  ∧ (alu-c (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_rega (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ∧ (state (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = V_REGA)
  ∧ (we-flags (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ∧ (regs-address (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-a (i-reg))
  ∧ (alu-op (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ∧ (alu-mpg (cv_rega (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable `cv_rega$destructure`; name this event ‘`cv_rega$destructure-off`’.


```

      = f)
  ^ (we-addr-out (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (we-hold- (cv_regb (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ^ (we-pc-reg (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (data-in-select (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (alu-c (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                          cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_regb (rw-, regs-address, i-reg, flags, pc-reg)) = f)
  ^ (state (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = V_REGB)
  ^ (we-flags (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-b (i-reg))
  ^ (alu-op (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_regb (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_regb\$destructure; name this event 'cv_regb\$destructure-off'.

```

(DEFN CV_UPDATE
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS
          (STORE-RESULTP (STORE-CC I-REG) FLAGS)
          (CONS F
            (CONS T
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F

```

```

(CONS F
  (CONS F
    (CONS F
      (CONS F
        (CONS
          (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                (CONS F (OP-CODE I-REG))))
          (CONS F
            (APPEND
              (V_UPDATE)
              (APPEND
                (SET-FLAGS I-REG)
                (APPEND
                  (RN-B I-REG)
                  (APPEND (OP-CODE I-REG)
                        (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))
                                NIL))))))))))))))

```

EVENT: Enable cv_update; name this event 'cv_update-off'.

THEOREM: cv_update\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_update (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_update (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = store-resultp (store-cc (i-reg), flags))
   ∧ (we-a-reg (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-i-reg (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-hold- (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (data-in-select (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
      = f)

```

```

^ (dec-addr-out (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (select-immediate (cv_update (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
  = f)
^ (alu-c (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
^ (alu-zero (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (state (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = V_UPDATE)
^ (we-flags (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = set-flags (i-reg))
^ (regs-address (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = rn-b (i-reg))
^ (alu-op (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = op-code (i-reg))
^ (alu-mpg (cv_update (rw-, regs-address, i-reg, flags, pc-reg))
  = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_update\$destructure; name this event 'cv_update\$destructure-off'.

```

(DEFN CV_READAO
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS T
            (CONS F
              (CONS F
                (CONS T
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS

```

```

(PRE-DEC-P (MODE-A I-REG))
(CONS F
  (CONS
    (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                          (CONS F (OP-CODE I-REG))))
    (CONS F
      (APPEND
        (V_READA0)
        (APPEND
          (MAKE-LIST 4 F)
          (APPEND
            (RN-A I-REG)
            (APPEND (OP-CODE I-REG)
                    (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                            NIL))))))))))

```

EVENT: Enable `cv_reada0`; name this event ‘`cv_reada0-off`’.

THEOREM: `cv_reada0$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-b-reg (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-i-reg (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-hold- (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (data-in-select (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (dec-addr-out (cv_reada0 (rw-, regs-address, i-reg, flags, pc-reg))
      = pre-dec-p (mode-a (i-reg)))

```



```

(CONS F
  (CONS F
    (CONS
      (IF* (PRE-DEC-P (MODE-A I-REG))
        (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
          (CONS F (ALU-DEC-OP))))
        (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
          (CONS F (ALU-INC-OP))))))
    (CONS F
      (APPEND
        (V_READA1)
        (APPEND
          (MAKE-LIST 4 F)
          (APPEND
            (RN-A I-REG)
            (APPEND (IF* (PRE-DEC-P (MODE-A I-REG))
              (ALU-DEC-OP)
              (ALU-INC-OP))
              (APPEND (IF* (PRE-DEC-P (MODE-A I-REG))
                (MPG (CONS F (CONS F (ALU-DEC-OP))))
                (MPG (CONS F (CONS F (ALU-INC-OP))))
                NIL))))))))))

```

EVENT: Enable `cv_reada1`; name this event ‘`cv_reada1-off`’.

THEOREM: `cv_reada1$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (strobe- (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = t)
  ∧ (hdack- (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
  ∧ (we-regs (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = and* (not* (a-immediate-p (i-reg)),
      or* (pre-dec-p (mode-a (i-reg)),
        post-inc-p (mode-a (i-reg)))))
  ∧ (we-a-reg (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-b-reg (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-i-reg (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-data-out (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ∧ (we-addr-out (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))

```



```

      = f)
  ∧ (we-hold- (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (dec-addr-out (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_reada1 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
  ∧ (alu-c (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-a (i-reg)),
              carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-DEC-OP))),
              carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-INC-OP))))))
  ∧ (alu-zero (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_READA1)
  ∧ (we-flags (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = rn-a (i-reg))
  ∧ (alu-op (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-a (i-reg)), ALU-DEC-OP, ALU-INC-OP))
  ∧ (alu-mpg (cv_reada1 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-a (i-reg)),
              mpg (cons (f, cons (f, ALU-DEC-OP))),
              mpg (cons (f, cons (f, ALU-INC-OP))))))

```

EVENT: Enable cv_reada1\$destructure; name this event 'cv_reada1\$destructure-off'.

```

(DEFN CV_READA2
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T

```

```

(CONS F
  (CONS T
    (CONS F
      (CONS F
        (CONS T
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS
                        (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                              (CONS F (OP-CODE I-REG))))
                      (CONS F
                        (CONS F
                          (APPEND
                            (V_READA2)
                            (APPEND
                              (MAKE-LIST 4 F)
                              (APPEND
                                (RN-B I-REG)
                                (APPEND (OP-CODE I-REG)
                                          (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                                                    NIL))))))))))))))))))

```

EVENT: Enable cv_reada2; name this event 'cv_reada2-off'.

THEOREM: cv_reada2\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (hdack- (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-i-reg (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ^ (we-addr-out (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (we-hold- (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (we-pc-reg (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (data-in-select (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_reada2 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_READA2)
  ^ (we-flags (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-b (i-reg))
  ^ (alu-op (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_reada2 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_reada2\$destructure; name this event 'cv_reada2\$destructure-off'.

```

(DEFN CV_READA3
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS F
      (CONS T
        (CONS F

```

```

(CONS T
  (CONS F
    (CONS F
      (CONS F
        (CONS F
          (CONS F
            (CONS T
              (CONS F
                (CONS F
                  (CONS
                    (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                          (CONS F (OP-CODE I-REG))))
                    (CONS F
                      (CONS F
                        (APPEND
                          (APPEND
                            (V_READA3)
                            (APPEND
                              (MAKE-LIST 4 F)
                              (APPEND PC-REG
                                (APPEND (OP-CODE I-REG)
                                  (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))
                                    NIL))))))))))))))))))

```

EVENT: Enable `cv_reada3`; name this event ‘`cv_reada3-off`’.

THEOREM: `cv_reada3$destructure`

`cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)`
 $\rightarrow ((rw- (cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg))) = \mathbf{t})$
 $\wedge (\text{strobe-}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{hdack-}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg))) = \mathbf{t})$
 $\wedge (\text{we-regs}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{we-a-reg}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{t})$
 $\wedge (\text{we-b-reg}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{we-i-reg}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{we-data-out}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{we-addr-out}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$
 $= \mathbf{f})$
 $\wedge (\text{we-hold-}(cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg)))$

```

      = f)
  ∧ (we-pc-reg (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
  ∧ (dec-addr-out (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_reada3 (rw-,
                                  regs-address,
                                  i-reg,
                                  flags,
                                  pc-reg))
      = f)
  ∧ (alu-c (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                                cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_READA3)
  ∧ (we-flags (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = pc-reg)
  ∧ (alu-op (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
  ∧ (alu-mpg (cv_reada3 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_reada3\$destructure; name this event 'cv_reada3\$destructure-off'.

```

(DEFN CV_READBO
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS F
            (CONS T
              (CONS F
                (CONS F

```

```

(CONS T
  (CONS F
    (CONS F
      (CONS F
        (CONS
          (PRE-DEC-P (MODE-B I-REG))
          (CONS F
            (CONS
              (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                   (CONS F (OP-CODE I-REG))))
              (CONS F
                (APPEND
                  (V_READB0)
                  (APPEND
                    (MAKE-LIST 4 F)
                    (APPEND
                      (RN-B I-REG)
                      (APPEND (OP-CODE I-REG)
                            (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))
                                          NIL))))))))))))))))))

```

EVENT: Enable `cv_readb0`; name this event ‘`cv_readb0-off`’.

THEOREM: `cv_readb0$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-i-reg (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-hold- (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ^ (data-in-select (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = pre-dec-p (mode-b (i-reg)))
  ^ (select-immediate (cv_readb0 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_READB0)
  ^ (we-flags (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-b (i-reg))
  ^ (alu-op (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_readb0\$destructure; name this event 'cv_readb0\$destructure-off'.

```

(DEFN CV_READB1
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS
            (OR* (A-IMMEDIATE-P I-REG)
                 (REG-DIRECT-P (MODE-A I-REG)))
            (CONS F
              (CONS F
                (CONS F

```

```

(CONS F
  (CONS F
    (CONS F
      (CONS F
        (CONS
          (A-IMMEDIATE-P I-REG)
          (CONS
            (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                  (CONS F (OP-CODE I-REG))))
            (CONS F
              (APPEND
                (V_READB1)
                (APPEND
                  (MAKE-LIST 4 F)
                  (APPEND
                    (RN-A I-REG)
                    (APPEND (OP-CODE I-REG)
                           (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))
                                         NIL))))))))))))))))))

```

EVENT: Enable cv_readb1; name this event 'cv_readb1-off'.

THEOREM: cv_readb1\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = or* (a-immediate-p (i-reg), reg-direct-p (mode-a (i-reg))))
   ∧ (we-b-reg (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-i-reg (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-hold- (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))

```



```

      = f)
  ^ (data-in-select (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_readb1 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = a-immediate-p (i-reg))
  ^ (alu-c (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_READB1)
  ^ (we-flags (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-a (i-reg))
  ^ (alu-op (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_readb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_readb1\$destructure; name this event 'cv_readb1\$destructure-off'.

```

(DEFN CV_READB2
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS F
      (CONS T
        (CONS
          (AND* (NOT* (STORE-RESULTP (STORE-CC I-REG)
                                FLAGS))
            (OR* (PRE-DEC-P (MODE-B I-REG))
              (POST-INC-P (MODE-B I-REG)))))
          (CONS F
            (CONS F

```

```

(CONS F
  (CONS F
    (CONS F
      (CONS F
        (CONS F
          (CONS F
            (CONS
              (IF* (PRE-DEC-P (MODE-B I-REG))
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                      (CONS F (ALU-DEC-OP))))
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                      (CONS F (ALU-INC-OP))))))
            (CONS F
              (APPEND
                (V_READB2)
                (APPEND
                  (MAKE-LIST 4 F)
                  (APPEND
                    (RN-B I-REG)
                    (APPEND (IF* (PRE-DEC-P (MODE-B I-REG))
                                (ALU-DEC-OP)
                                (ALU-INC-OP))
                      (APPEND (IF* (PRE-DEC-P (MODE-B I-REG))
                                (MPG (CONS F (CONS T (ALU-DEC-OP))))
                                (MPG (CONS F (CONS T (ALU-INC-OP))))
                                NIL))))))))))))))

```

EVENT: Enable `cv_readb2`; name this event ‘`cv_readb2-off`’.

THEOREM: `cv_readb2$destructure`

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (hdack- (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = and* (not* (store-resultp (store-cc (i-reg), flags)),
              or* (pre-dec-p (mode-b (i-reg)),
                  post-inc-p (mode-b (i-reg)))))
   ∧ (we-a-reg (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ∧ (we-i-reg (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-data-out (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-addr-out (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-hold- (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (dec-addr-out (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_readb2 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
  ∧ (alu-c (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)),
              carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-DEC-OP))),
              carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-INC-OP))))))
  ∧ (alu-zero (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_READB2)
  ∧ (we-flags (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = rn-b (i-reg))
  ∧ (alu-op (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)), ALU-DEC-OP, ALU-INC-OP))
  ∧ (alu-mpg (cv_readb2 (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)),
              mpg (cons (f, cons (t, ALU-DEC-OP))),
              mpg (cons (f, cons (t, ALU-INC-OP)))))

```

EVENT: Enable cv_readb2\$destructure; name this event 'cv_readb2\$destructure-off'.


```

      = t)
  ∧ (we-i-reg (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-data-out (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-addr-out (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-hold- (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
  ∧ (dec-addr-out (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_readb3 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
  ∧ (alu-c (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                              cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_READB3)
  ∧ (we-flags (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = pc-reg)
  ∧ (alu-op (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
  ∧ (alu-mpg (cv_readb3 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_readb3\$destructure; name this event 'cv_readb3\$destructure-off'.

```

(DEFN CV_WRITE0
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)

```

```

(CONS T
  (CONS T
    (CONS T
      (CONS F
        (CONS F
          (CONS T
            (CONS F
              (CONS T
                (CONS T
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS
                          (PRE-DEC-P (MODE-B I-REG))
                          (CONS F
                            (CONS
                              (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                    (CONS F (OP-CODE I-REG))))
                              (CONS F
                                (APPEND
                                  (V_WRITE0)
                                  (APPEND
                                    (SET-FLAGS I-REG)
                                    (APPEND
                                      (RN-B I-REG)
                                      (APPEND (OP-CODE I-REG)
                                                (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))
                                                            NIL))))))))))))))))))))))

```

EVENT: Enable cv_write0; name this event 'cv_write0-off'.

THEOREM: cv_write0\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-i-reg (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ∧ (we-data-out (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = t)
  ∧ (we-addr-out (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = t)
  ∧ (we-hold- (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = f)
  ∧ (we-pc-reg (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = f)
  ∧ (data-in-select (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = f)
  ∧ (dec-addr-out (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = pre-dec-p (mode-b (i-reg)))
  ∧ (select-immediate (cv_write0 (rw-,
                                   regs-address,
                                   i-reg,
                                   flags,
                                   pc-reg))
     = f)
  ∧ (alu-c (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = carry-in-help (cons (c-flag (flags),
                               cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = f)
  ∧ (state (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = V_WRITE0)
  ∧ (we-flags (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = set-flags (i-reg))
  ∧ (regs-address (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = rn-b (i-reg))
  ∧ (alu-op (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = op-code (i-reg))
  ∧ (alu-mpg (cv_write0 (rw-, regs-address, i-reg, flags, pc-reg))
     = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_write0\$destructure; name this event 'cv_write0\$destructure-off'.

```

(DEFN CV_WRITE1
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS F
    (CONS T

```

```

(CONS T
  (CONS
    (OR* (PRE-DEC-P (MODE-B I-REG))
      (POST-INC-P (MODE-B I-REG)))
    (CONS F
      (CONS F
        (CONS F
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS
                    (IF* (PRE-DEC-P (MODE-B I-REG))
                      (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                            (CONS F (ALU-DEC-OP))))
                      (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                            (CONS F (ALU-INC-OP))))))
                    (CONS F
                      (APPEND
                        (V_WRITE1)
                        (APPEND
                          (MAKE-LIST 4 F)
                          (APPEND
                            (RN-B I-REG)
                            (APPEND (IF* (PRE-DEC-P (MODE-B I-REG))
                                          (ALU-DEC-OP)
                                          (ALU-INC-OP))
                              (APPEND (IF* (PRE-DEC-P (MODE-B I-REG))
                                            (MPG (CONS F (CONS T (ALU-DEC-OP))))
                                            (MPG (CONS F (CONS T (ALU-INC-OP))))
                                            NIL))))))))))))))))))))))
  )
)

```

EVENT: Enable cv_write1; name this event 'cv_write1-off'.

THEOREM: cv_write1\$destructure

cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)

$$\begin{aligned}
 \rightarrow & ((rw-(cv\_write1\ (rw-, regs-address, i-reg, flags, pc-reg)) = \mathbf{f}) \\
 & \wedge \text{ (strobe-}(cv\_write1\ (rw-, regs-address, i-reg, flags, pc-reg)) \\
 & \quad = \mathbf{t}) \\
 & \wedge \text{ (hdack-}(cv\_write1\ (rw-, regs-address, i-reg, flags, pc-reg)) = \mathbf{t}) \\
 & \wedge \text{ (we-regs}(cv\_write1\ (rw-, regs-address, i-reg, flags, pc-reg))
 \end{aligned}$$


```

      = or* (pre-dec-p (mode-b (i-reg)),
             post-inc-p (mode-b (i-reg))))
  ∧ (we-a-reg (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-b-reg (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-i-reg (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-data-out (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-addr-out (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-hold- (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (dec-addr-out (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_writel (rw-,
                                   regs-address,
                                   i-reg,
                                   flags,
                                   pc-reg))
      = f)
  ∧ (alu-c (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)),
             carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-DEC-OP))),
             carry-in-help (cons (c-flag (flags),
                                   cons (f, ALU-INC-OP)))))
  ∧ (alu-zero (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = V_WRITE1)
  ∧ (we-flags (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = rn-b (i-reg))
  ∧ (alu-op (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)), ALU-DEC-OP, ALU-INC-OP))
  ∧ (alu-mpg (cv_writel (rw-, regs-address, i-reg, flags, pc-reg))
      = if* (pre-dec-p (mode-b (i-reg)),

```

$$\begin{aligned} & \text{mpg}(\text{cons}(\mathbf{f}, \text{cons}(\mathbf{t}, \text{ALU-DEC-OP}))), \\ & \text{mpg}(\text{cons}(\mathbf{f}, \text{cons}(\mathbf{t}, \text{ALU-INC-OP})))))) \end{aligned}$$

EVENT: Enable `cv_write1$destructure`; name this event ‘`cv_write1$destructure-off`’.

[illegible]

EVENT: Enable cv_write2; name this event 'cv_write2-off'.

THEOREM: `cv_write2$destructure`
`cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)`
 $\rightarrow ((\text{rw-}(\text{cv\_write2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, \text{pc-reg})) = \mathbf{f})$
 $\wedge (\text{strobe-}(\text{cv\_write2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, \text{pc-reg})))$

```

      = f)
 $\wedge$  (hdack- (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
 $\wedge$  (we-regs (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-a-reg (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-b-reg (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-i-reg (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-data-out (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-addr-out (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-hold- (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (we-pc-reg (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (data-in-select (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (dec-addr-out (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (select-immediate (cv_write2 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
 $\wedge$  (alu-c (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                              cons (f, op-code (i-reg)))))
 $\wedge$  (alu-zero (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
 $\wedge$  (state (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_WRITE2)
 $\wedge$  (we-flags (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
 $\wedge$  (regs-address (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = pc-reg)
 $\wedge$  (alu-op (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
 $\wedge$  (alu-mpg (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable `cv_write2$destructure`; name this event ‘`cv_write2$destructure-off`’.

```
(DEFN CV_WRITE3
(RW- REGS-ADDRESS I-REG FLAGS PC-REG)
(CONS F
(CONS F
(CONS T
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS F
(CONS
(CARRY-IN-HELP (CONS (C-FLAG FLAGS)
(CONS F (OP-CODE I-REG))))))
(CONS F
(APPEND
(V_WRITE3)
(APPEND
(MAKE-LIST 4 F)
(APPEND PC-REG
(APPEND (OP-CODE I-REG)
(APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
NIL)))))))))
```

EVENT: Enable cv_write3; name this event 'cv_write3-off'.

$$\begin{aligned} & \text{THEOREM: cv\_write3}\$destructure \\ & \text{cv-hyps } (rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg) \\ & \rightarrow ((rw\text{-}(\text{cv\_write3}(rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg))) = \mathbf{f}) \\ & \quad \wedge (\text{strobe}\text{-}(\text{cv\_write3}(rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg))) \\ & \quad = \mathbf{f}) \\ & \quad \wedge (\text{hdack}\text{-}(\text{cv\_write3}(rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg))) = \mathbf{t}) \\ & \quad \wedge (\text{we}\text{-}regs(\text{cv\_write3}(rw-, regs\text{-}address, i\text{-}reg, flags, pc\text{-}reg))) \end{aligned}$$

```

      = f)
  ∧ (we-a-reg (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-b-reg (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-i-reg (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-data-out (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-addr-out (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-hold- (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (dec-addr-out (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_write3 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
  ∧ (alu-c (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                                cons (f, op-code (i-reg))))))
  ∧ (alu-zero (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_WRITE3)
  ∧ (we-flags (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = pc-reg)
  ∧ (alu-op (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
  ∧ (alu-mpg (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg))))))

```

EVENT: Enable cv_write3\$destructure; name this event 'cv_write3\$destructure-off'.

```

(DEFN CV_SEFA0
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS T
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS
                          (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                (CONS F (OP-CODE I-REG))))
                        (CONS F
                          (APPEND
                            (V_SEFA0)
                            (APPEND
                              (MAKE-LIST 4 F)
                              (APPEND
                                (RN-A I-REG)
                                (APPEND (OP-CODE I-REG)
                                          (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                                                    NIL))))))))))))))))))

```

EVENT: Enable cv_sefa0; name this event 'cv_sefa0-off'.

THEOREM: cv_sefa0\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (hdack- (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg)) = f)
   ∧ (we-a-reg (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (we-b-reg (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)

```

```

^ (we-i-reg (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-data-out (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-addr-out (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-hold- (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-pc-reg (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (data-in-select (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (dec-addr-out (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (select-immediate (cv_sefa0 (rw-,
                               regs-address,
                               i-reg,
                               flags,
                               pc-reg))
  = f)
^ (alu-c (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = carry-in-help (cons (c-flag (flags),
                             cons (f, op-code (i-reg)))))
^ (alu-zero (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (state (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = V_SEFA0)
^ (we-flags (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = make-list (4, f))
^ (regs-address (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = rn-a (i-reg))
^ (alu-op (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = op-code (i-reg))
^ (alu-mpg (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
  = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_sefa0\$destructure; name this event 'cv_sefa0\$destructure-off'.

```

(DEFN CV_SEFA1
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T

```



```

^ (we-b-reg (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-i-reg (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-data-out (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-addr-out (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-hold- (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (we-pc-reg (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (data-in-select (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (dec-addr-out (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (select-immediate (cv_sefa1 (rw-,
                               regs-address,
                               i-reg,
                               flags,
                               pc-reg))
  = f)
^ (alu-c (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = if* (pre-dec-p (mode-a (i-reg)),
        carry-in-help (cons (c-flag (flags),
                              cons (f, ALU-DEC-OP))),
        carry-in-help (cons (c-flag (flags),
                              cons (f, ALU-INC-OP)))))
^ (alu-zero (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = f)
^ (state (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = V_SEFA1)
^ (we-flags (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = make-list (4, f))
^ (regs-address (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = rn-a (i-reg))
^ (alu-op (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = if* (pre-dec-p (mode-a (i-reg)), ALU-DEC-OP, ALU-INC-OP))
^ (alu-mpg (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
  = if* (pre-dec-p (mode-a (i-reg)),
        mpg (cons (f, cons (f, ALU-DEC-OP))),
        mpg (cons (f, cons (f, ALU-INC-OP)))))

```

EVENT: Enable cv_sefa1\$destructure; name this event 'cv_sefa1\$destructure-

off’.

```
(DEFN CV_SEFB0
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS F
            (CONS T
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS F
                          (CONS
                            (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                  (CONS F (OP-CODE I-REG))))
                            (CONS F
                              (APPEND
                                (V_SEFB0)
                                (APPEND
                                  (MAKE-LIST 4 F)
                                  (APPEND
                                    (RN-B I-REG)
                                    (APPEND (OP-CODE I-REG)
                                          (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                                                    NIL))))))))))))))))))))))
```

EVENT: Enable cv_sefb0; name this event ‘cv_sefb0-off’.

THEOREM: cv_sefb0\$destructure

```
cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (hdack- (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg)) = f)
   ∧ (we-a-reg (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
```

```

      = f)
  ∧ (we-b-reg (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
  ∧ (we-i-reg (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-data-out (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-addr-out (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-hold- (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (we-pc-reg (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (data-in-select (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (dec-addr-out (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (select-immediate (cv_sefb0 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
      = f)
  ∧ (alu-c (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                              cons (f, op-code (i-reg)))))
  ∧ (alu-zero (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
  ∧ (state (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_SEFB0)
  ∧ (we-flags (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
  ∧ (regs-address (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = rn-b (i-reg))
  ∧ (alu-op (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
  ∧ (alu-mpg (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_sefb0\$destructure; name this event 'cv_sefb0\$destructure-off'.


```

 $\wedge$  (we-regs (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
 $\wedge$  (we-a-reg (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-b-reg (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-i-reg (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-data-out (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-addr-out (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-hold- (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-pc-reg (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (data-in-select (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (dec-addr-out (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (select-immediate (cv_sefb1 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
 $\wedge$  (alu-c (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = if* (pre-dec-p (mode-b (i-reg)),
           carry-in-help (cons (c-flag (flags),
                                cons (f, ALU-DEC-OP))),
           carry-in-help (cons (c-flag (flags),
                                cons (f, ALU-INC-OP)))))
 $\wedge$  (alu-zero (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (state (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = v_SEFB1)
 $\wedge$  (we-flags (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
 $\wedge$  (regs-address (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = rn-b (i-reg))
 $\wedge$  (alu-op (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = if* (pre-dec-p (mode-b (i-reg)), ALU-DEC-OP, ALU-INC-OP))
 $\wedge$  (alu-mpg (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
    = if* (pre-dec-p (mode-b (i-reg)),
           mpg (cons (f, cons (t, ALU-DEC-OP))),

```

mpg (cons (f, cons (t, ALU-INC-OP))))))

EVENT: Enable cv_sefb1\$destructure; name this event 'cv_sefb1\$destructure-off'.

```
(DEFN CV_HOLD0
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS F
        (CONS F
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS T
                    (CONS T
                      (CONS F
                        (CONS F
                          (CONS F
                            (CONS
                              (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                    (CONS F (OP-CODE I-REG))))
                              (CONS F
                                (CONS F
                                  (APPEND
                                    (V_HOLD0)
                                    (APPEND
                                      (MAKE-LIST 4 F)
                                      (APPEND PC-REG
                                        (APPEND (OP-CODE I-REG)
                                          (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                                            NIL))))))))))))))))))))))
```

EVENT: Enable cv_hold0; name this event 'cv_hold0-off'.

THEOREM: cv_hold0\$destructure

cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
 $\rightarrow ((rw-(cv\_hold0(rw-, regs-address, i-reg, flags, pc-reg)) = t)$
 $\wedge (strobe-(cv\_hold0(rw-, regs-address, i-reg, flags, pc-reg)) = t)$
 $\wedge (hdack-(cv\_hold0(rw-, regs-address, i-reg, flags, pc-reg)) = f)$

```

 $\wedge$  (we-regs (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg)) = f)
 $\wedge$  (we-a-reg (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-b-reg (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-i-reg (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-data-out (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-addr-out (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (we-hold- (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = t)
 $\wedge$  (we-pc-reg (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = t)
 $\wedge$  (data-in-select (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (dec-addr-out (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (select-immediate (cv_hold0 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
 $\wedge$  (alu-c (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                            cons (f, op-code (i-reg)))))
 $\wedge$  (alu-zero (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
 $\wedge$  (state (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = v_HOLD0)
 $\wedge$  (we-flags (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
 $\wedge$  (regs-address (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
 $\wedge$  (alu-op (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
 $\wedge$  (alu-mpg (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_hold0\$destructure; name this event 'cv_hold0\$destructure-off'.


```

      = f)
    ∧ (we-data-out (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (we-addr-out (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (we-hold- (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (we-pc-reg (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (data-in-select (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (dec-addr-out (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (select-immediate (cv_hold1 (rw-,
                                   regs-address,
                                   i-reg,
                                   flags,
                                   pc-reg))
      = f)
    ∧ (alu-c (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = carry-in-help (cons (c-flag (flags),
                              cons (f, op-code (i-reg)))))
    ∧ (alu-zero (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
    ∧ (state (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = V_HOLD1)
    ∧ (we-flags (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = make-list (4, f))
    ∧ (regs-address (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = pc-reg)
    ∧ (alu-op (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = op-code (i-reg))
    ∧ (alu-mpg (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
      = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_hold1\$destructure; name this event 'cv_hold1\$destructure-off'.

```

(DEFN CV_V11010
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T

```



```

      = f)
  ^ (we-hold- (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (we-pc-reg (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (data-in-select (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_v11010 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = v_v11010)
  ^ (we-flags (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
  ^ (alu-op (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_v11010\$destructure; name this event 'cv_v11010\$destructure-off'.

```

(DEFN CV_V11011
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS F
            (CONS F

```

```

(CONS F
  (CONS F
    (CONS F
      (CONS F
        (CONS F
          (CONS F
            (CONS
              (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                   (CONS F (OP-CODE I-REG))))
              (CONS F
                (CONS F
                  (APPEND
                    (V_V11011)
                    (APPEND
                     (MAKE-LIST 4 F)
                     (APPEND PC-REG
                           (APPEND (OP-CODE I-REG)
                                   (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG))))
                                           NIL))))))))))))))))))

```

EVENT: Enable cv_v11011; name this event 'cv_v11011-off'.

THEOREM: cv_v11011\$destructure

```

cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
→ ((rw- (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (strobe- (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = t)
   ∧ (hdack- (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg)) = t)
   ∧ (we-regs (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-a-reg (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-b-reg (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-i-reg (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-data-out (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-addr-out (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-hold- (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
      = f)
   ∧ (we-pc-reg (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))

```

```

      = f)
  ^ (data-in-select (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (dec-addr-out (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (select-immediate (cv_v11011 (rw-,
                                regs-address,
                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags),
                           cons (f, op-code (i-reg)))))
  ^ (alu-zero (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = f)
  ^ (state (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = v_v11011)
  ^ (we-flags (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (regs-address (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = pc-reg)
  ^ (alu-op (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = op-code (i-reg))
  ^ (alu-mpg (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (f, cons (f, op-code (i-reg)))))

```

EVENT: Enable cv_v11011\$destructure; name this event 'cv_v11011\$destructure-off'.

```

(DEFN CV_RESET0
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS T
          (CONS F
            (CONS F
              (CONS F
                (CONS T
                  (CONS F
                    (CONS T

```



```

                                i-reg,
                                flags,
                                pc-reg))
    = f)
  ^ (alu-c (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = carry-in-help (cons (c-flag (flags), cons (t, ALU-INC-OP))))
  ^ (alu-zero (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = t)
  ^ (state (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = V_RESET0)
  ^ (we-flags (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, t))
  ^ (regs-address (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = make-list (4, f))
  ^ (alu-op (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = ALU-INC-OP)
  ^ (alu-mpg (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
    = mpg (cons (t, cons (f, ALU-INC-OP))))

```

EVENT: Enable cv_reset0\$destructure; name this event 'cv_reset0\$destructure-off'.

```

(DEFN CV_RESET1
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS F
          (CONS T
            (CONS T
              (CONS F
                (CONS T
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS
                          (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                (CONS T (ALU-INC-OP))))
                        (CONS T

```

```

(APPEND
 (V_RESET1)
 (APPEND
  (MAKE-LIST 4 F)
  (APPEND (MAKE-LIST 4 F)
   (APPEND (ALU-INC-OP)
    (APPEND (MPG (CONS T (CONS F (ALU-INC-OP))))
     NIL)))))))))

```

EVENT: Enable `cv_reset1`; name this event '`cv_reset1-off`'.

THEOREM: `cv_reset1$destructure`

$$\begin{aligned}
\mathbf{t} \rightarrow & ((\text{rw-}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) = \mathbf{t}) \\
& \wedge (\text{strobe-}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) = \mathbf{t}) \\
& \wedge (\text{hdack-}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) = \mathbf{t}) \\
& \wedge (\text{we-regs}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) = \mathbf{f}) \\
& \wedge (\text{we-a-reg}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{t}) \\
& \wedge (\text{we-b-reg}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{t}) \\
& \wedge (\text{we-i-reg}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{t}) \\
& \wedge (\text{we-data-out}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{we-addr-out}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{t}) \\
& \wedge (\text{we-hold-}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{we-pc-reg}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{data-in-select}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{dec-addr-out}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{select-immediate}(\text{cv\_reset1}(\text{rw-}, \\
& \quad \text{regs-address}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg})) \\
& \quad = \mathbf{f}) \\
& \wedge (\text{alu-c}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \text{carry-in-help}(\text{cons}(\text{c-flag}(\text{flags}), \text{cons}(\mathbf{t}, \text{ALU-INC-OP})))) \\
& \wedge (\text{alu-zero}(\text{cv\_reset1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
& \quad = \mathbf{t})
\end{aligned}$$


```

^ (state (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
   = V_RESET1)
^ (we-flags (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
   = make-list (4, f))
^ (regs-address (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
   = make-list (4, f))
^ (alu-op (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
   = ALU-INC-OP)
^ (alu-mpg (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
   = mpg (cons (t, cons (f, ALU-INC-OP))))

```

EVENT: Enable cv_reset1\$destructure; name this event 'cv_reset1\$destructure-off'.

```

(DEFN CV_RESET2
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
  (CONS T
    (CONS T
      (CONS T
        (CONS T
          (CONS F
            (CONS F
              (CONS F
                (CONS F
                  (CONS F
                    (CONS F
                      (CONS F
                        (CONS F
                          (CONS
                            (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                                  (CONS T (ALU-INC-OP))))
                        (CONS T
                          (APPEND
                            (V_RESET2)
                            (APPEND
                              (MAKE-LIST 4 F)
                              (APPEND (V-INC REGS-ADDRESS)
                                        (APPEND (ALU-INC-OP)
                                                  (APPEND (MPG (CONS T (CONS F (ALU-INC-OP))))
                                                            NIL))))))))))))))))))

```

EVENT: Enable `cv_reset2`; name this event '`cv_reset2-off`'.

THEOREM: `cv_reset2$destructure`

`cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)`
 $\rightarrow$ $((rw-(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg)) = \mathbf{t})$
 $\wedge$ $(strobe-(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{t})$
 $\wedge$ $(hdack-(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg)) = \mathbf{t})$
 $\wedge$ $(we-regs(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{t})$
 $\wedge$ $(we-a-reg(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-b-reg(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-i-reg(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-data-out(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-addr-out(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-hold-(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(we-pc-reg(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(data-in-select(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(dec-addr-out(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(select-immediate(cv\_reset2(rw-,$
 $regs-address,$
 $i-reg,$
 $flags,$
 $pc-reg))$
 $= \mathbf{f})$
 $\wedge$ $(alu-c(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \text{carry-in-help}(\text{cons}(\text{c-flag}(flags), \text{cons}(\mathbf{t}, \text{ALU-INC-OP}))))$
 $\wedge$ $(alu-zero(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \mathbf{t})$
 $\wedge$ $(state(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \text{V\_RESET2})$
 $\wedge$ $(we-flags(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$
 $= \text{make-list}(4, \mathbf{f}))$
 $\wedge$ $(regs-address(cv\_reset2(rw-, regs-address, i-reg, flags, pc-reg))$

$$\begin{aligned} &= \text{v-inc}(\text{regs-address})) \\ \wedge \quad &(\text{alu-op}(\text{cv\_reset2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\ &= \text{ALU-INC-OP}) \\ \wedge \quad &(\text{alu-mpg}(\text{cv\_reset2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\ &= \text{mpg}(\text{cons}(\mathbf{t}, \text{cons}(\mathbf{f}, \text{ALU-INC-OP})))) \end{aligned}$$

EVENT: Enable `cv_reset2$destructure`; name this event ‘`cv_reset2$destructure-off`’.

```
(DEFN CV_V11111  
  (RW- REGS-ADDRESS I-REG FLAGS PC-REG)  
  (CONS T  
    (CONS T  
      (CONS T  
        (CONS F  
          (CONS F  
            (CONS F  
              (CONS F  
                (CONS F  
                  (CONS F  
                    (CONS F  
                      (CONS F  
                        (CONS F  
                          (CARRY-IN-HELP (CONS (C-FLAG FLAGS)  
                                                (CONS F (OP-CODE I-REG))))  
                      (CONS F  
                        (APPEND  
                          (V_V11111)  
                          (APPEND  
                            (MAKE-LIST 4 F)  
                            (APPEND PC-REG  
                              (APPEND (OP-CODE I-REG)  
                                (APPEND (MPG (CONS F (CONS F (OP-CODE I-REG)))  
                                          NIL)))))
```

EVENT: Enable cv_v11111; name this event 'cv_v11111-off'.

THEOREM: cv_v11111\$destructure

$cv\_hyps(rw-, regs\_address, i-reg, flags, pc-reg)$
 $\rightarrow ((rw-(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg)) = \mathbf{t})$
 $\wedge (strobe-(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{t})$
 $\wedge (hdack-(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg)) = \mathbf{t})$
 $\wedge (we\_regs(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_a-reg(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_b-reg(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_i-reg(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_data-out(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_addr-out(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_hold-(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (we\_pc-reg(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (data-in-select(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (dec-addr-out(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (select-immediate(cv\_v11111(rw-,$
 $regs\_address,$
 $i-reg,$
 $flags,$
 $pc-reg))$
 $= \mathbf{f})$
 $\wedge (alu-c(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= carry-in-help(cons(c-flag(flags),$
 $cons(\mathbf{f}, op-code(i-reg))))$
 $\wedge (alu-zero(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= \mathbf{f})$
 $\wedge (state(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= v\_v11111)$
 $\wedge (we\_flags(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= make-list(4, \mathbf{f}))$
 $\wedge (regs\_address(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= pc-reg)$
 $\wedge (alu-op(cv\_v11111(rw-, regs\_address, i-reg, flags, pc-reg))$
 $= op-code(i-reg))$

$$\begin{aligned} & \wedge \text{ (alu-mpg (cv\_v11111 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad = \text{ mpg (cons (f, cons (f, op-code (i-reg))))))} \end{aligned}$$

EVENT: Enable cv_v11111\$destructure; name this event 'cv_v11111\$destructure-off'.

EVENT: Let us define the theory *cv-theory* to consist of the following events: cv_fetch0, cv_fetch1, cv_fetch2, cv_fetch3, cv_decode, cv_rega, cv_regb, cv_update, cv_reada0, cv_reada1, cv_reada2, cv_reada3, cv_readb0, cv_readb1, cv_readb2, cv_readb3, cv_write0, cv_write1, cv_write2, cv_write3, cv_sefa0, cv_sefa1, cv_sefb0, cv_sefb1, cv_hold0, cv_hold1, cv_v11010, cv_v11011, cv_reset0, cv_reset1, cv_reset2, cv_v11111.

EVENT: Let us define the theory *cv-destructure-theory* to consist of the following events: cv_fetch0\$destructure, cv_fetch1\$destructure, cv_fetch2\$destructure, cv_fetch3\$destructure, cv_decode\$destructure, cv_rega\$destructure, cv_regb\$destructure, cv_update\$destructure, cv_reada0\$destructure, cv_reada1\$destructure, cv_reada2\$destructure, cv_reada3\$destructure, cv_readb0\$destructure, cv_readb1\$destructure, cv_readb2\$destructure, cv_readb3\$destructure, cv_write0\$destructure, cv_write1\$destructure, cv_write2\$destructure, cv_write3\$destructure, cv_sefa0\$destructure, cv_sefa1\$destructure, cv_sefb0\$destructure, cv_sefb1\$destructure, cv_hold0\$destructure, cv_hold1\$destructure, cv_v11010\$destructure, cv_v11011\$destructure, cv_reset0\$destructure, cv_reset1\$destructure, cv_reset2\$destructure, cv_v11111\$destructure.

THEOREM: bvp-cv-vectors

$$\begin{aligned} & \text{cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)} \\ \rightarrow & \text{ (bvp (cv\_fetch0 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_fetch2 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_fetch3 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_decode (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_rega (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_regb (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_update (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_reada0 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_reada1 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_reada2 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_reada3 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_readb0 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_readb1 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_readb2 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_readb3 (rw-, regs-address, i-reg, flags, pc-reg))} \\ & \quad \wedge \text{ bvp (cv\_write0 (rw-, regs-address, i-reg, flags, pc-reg))} \end{aligned}$$

$\wedge$ bvp (cv_write1 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_write2 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_write3 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_sefa0 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_sefa1 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_sefb1 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_hold0 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_hold1 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_v11010 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_v11011 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_reset0 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_reset1 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_reset2 (rw-, regs-address, i-reg, flags, pc-reg))
 $\wedge$ bvp (cv_v11111 (rw-, regs-address, i-reg, flags, pc-reg)))

EVENT: Enable bvp-cv-vectors; name this event ‘bvp-cv-vectors-off’.

THEOREM: cv_fetch1\$rw-doan-matta
 cv_fetch1 (f, regs-address, i-reg, flags, pc-reg)
 = cv_fetch1 (t, regs-address, i-reg, flags, pc-reg)

THEOREM: properp-length-cv_fetch1
 (cv-hyps (rw-, regs-address, i-reg, flags, pc-reg)
 $\rightarrow$ (length (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg)) = 40))
 $\wedge$ properp (cv_fetch1 (rw-, regs-address, i-reg, flags, pc-reg))

DEFINITION:
 f\$control-let (i-reg, flags, regs-address)
 = list (id (a-immediate-p (i-reg)),
 f\$b-store-resultp (store-cc (i-reg), flags),
 f\$set-some-flags (set-flags (i-reg)),
 f-or (id (a-immediate-p (i-reg)),
 car (f\$decode-reg-mode (mode-a (i-reg)))),
 car (f\$decode-reg-mode (mode-b (i-reg))),
 f\$unary-op-code-p (op-code (i-reg)),
 cadr (f\$decode-reg-mode (mode-a (i-reg))),
 cadr (f\$decode-reg-mode (mode-b (i-reg))),
 id (c-flag (flags)),
 f\$all-t-regs-address (regs-address),
 f-and (f-not (id (a-immediate-p (i-reg))),
 caddr (f\$decode-reg-mode (mode-a (i-reg))),
 caddr (f\$decode-reg-mode (mode-b (i-reg))))

EVENT: Enable f\$control-let; name this event ‘f\$control-let-off’.

```

(DEFN CONTROL-LET* NIL
  (CONS 'CONTROL-LET
    (CONS
      (APPEND (INDICES 'I-REG 0 32)
        (APPEND (INDICES 'FLAGS 0 4)
          (INDICES 'REGS-ADDRESS 0 4)))
      (CONS
        '(A-IMMEDIATE-P STORE SET-SOME-FLAGS DIRECT-A DIRECT-B UNARY PRE-DEC-A
          PRE-DEC-B C ALL-T-REGS-ADDRESS SIDE-EFFECT-A SIDE-EFFECT-B)
        (CONS
          (CONS
            (LIST 'GO
              '(A-IMMEDIATE-P)
              'ID
              (LIST (A-IMMEDIATE-P (INDICES 'I-REG 0 32))))
            (CONS
              '(G1 (A-IMMEDIATE-P-)
                B-NOT
                (A-IMMEDIATE-P))
              (CONS
                (LIST 'G2
                  '(STORE)
                  'B-STORE-RESULTP
                  (APPEND (STORE-CC (INDICES 'I-REG 0 32))
                    (INDICES 'FLAGS 0 4)))
                (CONS
                  (LIST 'G3
                    '(SET-SOME-FLAGS)
                    'B-OR4
                    (SET-FLAGS (INDICES 'I-REG 0 32)))
                  (CONS
                    (LIST 'G4
                      '(ALMOST-DIRECT-A PRE-DEC-A ALMOST-SIDE-EFFECT-A)
                      'DECODE-REG-MODE
                      (MODE-A (INDICES 'I-REG 0 32)))
                    (CONS
                      (LIST 'G5
                        '(DIRECT-B PRE-DEC-B SIDE-EFFECT-B)
                        'DECODE-REG-MODE
                        (MODE-B (INDICES 'I-REG 0 32)))
                      (CONS (LIST 'G6
                        '(UNARY)

```

```

'UNARY-OP-CODE-P
(OP-CODE (INDICES 'I-REG 0 32)))
(CONS (LIST 'G7
            '(C)
            'ID
            (LIST (C-FLAG (INDICES 'FLAGS 0 4))))
(CONS (LIST 'G8
            '(ALL-T-REGS-ADDRESS)
            'B-AND4
            (INDICES 'REGS-ADDRESS 0 4))
      '((G9 (SIDE-EFFECT-A)
            B-AND
            (A-IMMEDIATE-P- ALMOST-SIDE-EFFECT-A))
        (GA (DIRECT-A)
            B-OR
            (A-IMMEDIATE-P ALMOST-DIRECT-A)))))))))
'(NIL))))))

```

DEFINITION:

control-let& (*netlist*)

```

= ((lookup-module ('control-let, netlist) = CONTROL-LET*)
   ^ b-not& (delete-module ('control-let, netlist))
   ^ b-store-resultp& (delete-module ('control-let, netlist))
   ^ b-or4& (delete-module ('control-let, netlist))
   ^ decode-reg-mode& (delete-module ('control-let, netlist))
   ^ unary-op-code-p& (delete-module ('control-let, netlist))
   ^ id& (delete-module ('control-let, netlist))
   ^ b-and4& (delete-module ('control-let, netlist))
   ^ b-and& (delete-module ('control-let, netlist))
   ^ b-or& (delete-module ('control-let, netlist)))

```

EVENT: Enable control-let&; name this event 'control-let&-off'.

DEFINITION:

CONTROL-LET\$NETLIST

```

= cons (CONTROL-LET*,
        ((((((B-NOT$NETLIST ∪ B-STORE-RESULTP$NETLIST)
              ∪ B-OR4$NETLIST)
          ∪ DECODE-REG-MODE$NETLIST)
        ∪ UNARY-OP-CODE-P$NETLIST)
      ∪ ID$NETLIST)
    ∪ B-AND4$NETLIST)
  ∪ B-AND$NETLIST)
  ∪ B-OR$NETLIST)

```


THEOREM: control-let\$value
 (control-let& (*netlist*)
 $\wedge$ properp (*i-reg*)
 $\wedge$ (length (*i-reg*) = 32)
 $\wedge$ properp (*flags*)
 $\wedge$ (length (*flags*) = 4)
 $\wedge$ properp (*regs-address*)
 $\wedge$ (length (*regs-address*) = 4))
 $\rightarrow$ (dual-eval (0,
 'control-let,
 append (*i-reg*, append (*flags*, *regs-address*)),
 state,
 netlist)
 = f\$control-let (*i-reg*, *flags*, *regs-address*))

EVENT: Enable control-let\$value; name this event 'control-let\$value-off'.

THEOREM: f\$control-let=control-let
 (bvp (*i-reg*)
 $\wedge$ (length (*i-reg*) = 32)
 $\wedge$ bvp (*flags*)
 $\wedge$ (length (*flags*) = 4)
 $\wedge$ bvp (*regs-address*)
 $\wedge$ (length (*regs-address*) = 4))
 $\rightarrow$ (f\$control-let (*i-reg*, *flags*, *regs-address*)
 = list (a-immediate-p (*i-reg*),
 store-resultp (store-cc (*i-reg*), *flags*),
 set-some-flags (set-flags (*i-reg*)),
 or* (a-immediate-p (*i-reg*), reg-direct-p (mode-a (*i-reg*))),
 reg-direct-p (mode-b (*i-reg*)),
 unary-op-code-p (op-code (*i-reg*)),
 pre-dec-p (mode-a (*i-reg*)),
 pre-dec-p (mode-b (*i-reg*)),
 c-flag (*flags*),
 regs-address = list (**t**, **t**, **t**, **t**),
 and* (not* (a-immediate-p (*i-reg*)),
 or* (pre-dec-p (mode-a (*i-reg*)),
 post-inc-p (mode-a (*i-reg*)))),
 or* (pre-dec-p (mode-b (*i-reg*)), post-inc-p (mode-b (*i-reg*))))))

DEFINITION:
 next-state (*decoded-state*,
 store,
 set-some-flags,

```

    unary,
    direct-a,
    direct-b,
    side-effect-a,
    side-effect-b,
    all-t-regs-address,
    dtack-,
    hold-)
= list (b-nand (b-and4 (b-nand (all-t-regs-address, nth (30, decoded-state)),
    b-not (nth (25, decoded-state)),
    b-not (nth (23, decoded-state)),
    b-nand (b-not (side-effect-b), nth (21, decoded-state))),
    b-and3 (b-nand (b-not (dtack-), nth (19, decoded-state)),
    b-nand (b-not (b-and (side-effect-b, unary)),
    nth (7, decoded-state)),
    b-nand4 (b-not (side-effect-b),
    b-not (side-effect-a),
    b-not (b-or (store, set-some-flags)),
    nth (4, decoded-state)))),
    nth (0, decoded-state),
    b-and (hold-, nth (1, decoded-state)),
    b-nand (b-nand (dtack-, nth (3, decoded-state)),
    b-not (nth (2, decoded-state))),
    b-and (b-not (dtack-), nth (3, decoded-state)),
    b-and4 (b-or (direct-b, unary),
    direct-a,
    b-or (store, set-some-flags),
    nth (4, decoded-state)),
    b-and3 (b-not (unary), direct-b, nth (5, decoded-state)),
    b-nand6 (b-nand3 (b-not (store),
    b-not (dtack-),
    nth (15, decoded-state)),
    b-nand5 (b-not (store),
    unary,
    b-not (direct-b),
    b-not (dtack-),
    nth (11, decoded-state)),
    b-nand3 (direct-b, b-not (dtack-), nth (11, decoded-state)),
    b-not (nth (6, decoded-state)),
    b-nand3 (b-not (store),
    b-not (direct-b),
    nth (5, decoded-state)),
    b-nand3 (unary, direct-b, nth (5, decoded-state))),
    b-and3 (b-not (direct-a),

```

```

        b-or (store, set-some-flags),
        nth (4, decoded-state)),
nth (8, decoded-state),
nth (9, decoded-state),
b-nand (b-nand (dtack-, nth (11, decoded-state)),
        b-not (nth (10, decoded-state))),
b-nand (b-nand4 (b-not (unary),
                b-not (direct-b),
                b-not (dtack-),
                nth (11, decoded-state))),
        b-nand4 (b-not (b-or (direct-b, unary),
                direct-a,
                b-or (store, set-some-flags),
                nth (4, decoded-state)))),
nth (12, decoded-state),
nth (13, decoded-state),
b-nand (b-nand (dtack-, nth (15, decoded-state)),
        b-not (nth (14, decoded-state))),
b-nand3 (b-nand3 (store, b-not (dtack-), nth (15, decoded-state)),
        b-nand5 (store,
                unary,
                b-not (direct-b),
                b-not (dtack-),
                nth (11, decoded-state))),
        b-nand3 (store, b-not (direct-b), nth (5, decoded-state))),
nth (16, decoded-state),
nth (17, decoded-state),
b-nand (b-nand (dtack-, nth (19, decoded-state)),
        b-not (nth (18, decoded-state))),
b-and3 (side-effect-a,
        b-not (b-or (store, set-some-flags)),
        nth (4, decoded-state)),
nth (20, decoded-state),
b-nand (b-nand (side-effect-b, nth (21, decoded-state)),
        b-nand4 (side-effect-b,
                b-not (side-effect-a),
                b-not (b-or (store, set-some-flags)),
                nth (4, decoded-state))),
b-nand (b-not (nth (22, decoded-state)),
        b-nand (b-and (side-effect-b, unary), nth (7, decoded-state))),
b-nand (b-nand (b-not (hold-), nth (24, decoded-state)),
        b-nand (b-not (hold-), nth (1, decoded-state))),
b-and (hold-, nth (24, decoded-state)),
vss,

```

VSS,
 b-nand3 (b-not (nth (31, *decoded-state*)),
 b-not (nth (27, *decoded-state*)),
 b-not (nth (26, *decoded-state*))),
 nth (28, *decoded-state*),
 b-nand (b-nand (b-not (*all-t-regs-address*), nth (30, *decoded-state*)),
 b-not (nth (29, *decoded-state*))),
 VSS)

DEFINITION:

f\$next-state (*decoded-state*,
 store,
 set-some-flags,
 unary,
 direct-a,
 direct-b,
 side-effect-a,
 side-effect-b,
 all-t-regs-address,
 dtack-,
 hold-)
 = list (f-nand (f-and4 (f-nand (*all-t-regs-address*, nth (30, *decoded-state*)),
 f-not (nth (25, *decoded-state*)),
 f-not (nth (23, *decoded-state*)),
 f-nand (f-not (*side-effect-b*), nth (21, *decoded-state*))),
 f-and3 (f-nand (f-not (*dtack-*), nth (19, *decoded-state*)),
 f-nand (f-not (f-and (*side-effect-b*, *unary*)),
 nth (7, *decoded-state*)),
 f-nand4 (f-not (*side-effect-b*),
 f-not (*side-effect-a*),
 f-not (f-or (*store*, *set-some-flags*)),
 nth (4, *decoded-state*))),
 nth (0, *decoded-state*),
 f-and (*hold-*, nth (1, *decoded-state*)),
 f-nand (f-nand (*dtack-*, nth (3, *decoded-state*)),
 f-not (nth (2, *decoded-state*))),
 f-and (f-not (*dtack-*), nth (3, *decoded-state*)),
 f-and4 (f-or (*direct-b*, *unary*),
 direct-a,
 f-or (*store*, *set-some-flags*),
 nth (4, *decoded-state*)),
 f-and3 (f-not (*unary*), *direct-b*, nth (5, *decoded-state*)),
 f-nand6 (f-nand3 (f-not (*store*),
 f-not (*dtack-*),

```

        nth (15, decoded-state)),
f-nand5 (f-not (store),
        unary,
        f-not (direct-b),
        f-not (dtack-),
        nth (11, decoded-state)),
f-nand3 (direct-b, f-not (dtack-), nth (11, decoded-state)),
f-not (nth (6, decoded-state)),
f-nand3 (f-not (store),
        f-not (direct-b),
        nth (5, decoded-state)),
f-nand3 (unary, direct-b, nth (5, decoded-state))),
f-and3 (f-not (direct-a),
        f-or (store, set-some-flags),
        nth (4, decoded-state)),
nth (8, decoded-state),
nth (9, decoded-state),
f-nand (f-nand (dtack-, nth (11, decoded-state)),
        f-not (nth (10, decoded-state))),
f-nand (f-nand4 (f-not (unary),
                f-not (direct-b),
                f-not (dtack-),
                nth (11, decoded-state)),
        f-nand4 (f-not (f-or (direct-b, unary)),
                direct-a,
                f-or (store, set-some-flags),
                nth (4, decoded-state))),
nth (12, decoded-state),
nth (13, decoded-state),
f-nand (f-nand (dtack-, nth (15, decoded-state)),
        f-not (nth (14, decoded-state))),
f-nand3 (f-nand3 (store, f-not (dtack-), nth (15, decoded-state)),
        f-nand5 (store,
                unary,
                f-not (direct-b),
                f-not (dtack-),
                nth (11, decoded-state)),
        f-nand3 (store, f-not (direct-b), nth (5, decoded-state))),
nth (16, decoded-state),
nth (17, decoded-state),
f-nand (f-nand (dtack-, nth (19, decoded-state)),
        f-not (nth (18, decoded-state))),
f-and3 (side-effect-a,
        f-not (f-or (store, set-some-flags))),

```

$\text{nth}(4, \text{decoded-state}),$
 $\text{nth}(20, \text{decoded-state}),$
 $\text{f-nand}(\text{f-nand}(\text{side-effect-b}, \text{nth}(21, \text{decoded-state})),$
 $\text{f-nand4}(\text{side-effect-b},$
 $\text{f-not}(\text{side-effect-a}),$
 $\text{f-not}(\text{f-or}(\text{store}, \text{set-some-flags})),$
 $\text{nth}(4, \text{decoded-state}))),$
 $\text{f-nand}(\text{f-not}(\text{nth}(22, \text{decoded-state})),$
 $\text{f-nand}(\text{f-and}(\text{side-effect-b}, \text{unary}), \text{nth}(7, \text{decoded-state}))),$
 $\text{f-nand}(\text{f-nand}(\text{f-not}(\text{hold-}), \text{nth}(24, \text{decoded-state})),$
 $\text{f-nand}(\text{f-not}(\text{hold-}), \text{nth}(1, \text{decoded-state}))),$
 $\text{f-and}(\text{hold-}, \text{nth}(24, \text{decoded-state})),$
 $\text{VSS},$
 $\text{VSS},$
 $\text{f-nand3}(\text{f-not}(\text{nth}(31, \text{decoded-state})),$
 $\text{f-not}(\text{nth}(27, \text{decoded-state})),$
 $\text{f-not}(\text{nth}(26, \text{decoded-state}))),$
 $\text{nth}(28, \text{decoded-state}),$
 $\text{f-nand}(\text{f-nand}(\text{f-not}(\text{all-t-regs-address}), \text{nth}(30, \text{decoded-state})),$
 $\text{f-not}(\text{nth}(29, \text{decoded-state}))),$
 $\text{VSS})$

EVENT: Enable $\text{f\$next-state}$; name this event ' $\text{f\$next-state-off}$ '.

THEOREM: $\text{f\$next-state} = \text{next-state}$

$(\text{bvp}(\text{decoded-state})$
 $\wedge \text{length}(\text{decoded-state}) = 32)$
 $\wedge \text{boolp}(\text{store})$
 $\wedge \text{boolp}(\text{set-some-flags})$
 $\wedge \text{boolp}(\text{unary})$
 $\wedge \text{boolp}(\text{direct-a})$
 $\wedge \text{boolp}(\text{direct-b})$
 $\wedge \text{boolp}(\text{side-effect-a})$
 $\wedge \text{boolp}(\text{side-effect-b})$
 $\wedge \text{boolp}(\text{all-t-regs-address})$
 $\wedge \text{boolp}(\text{dtack-})$
 $\wedge \text{boolp}(\text{hold-})$
 $\rightarrow (\text{f\$next-state}(\text{decoded-state},$
 $\text{store},$
 $\text{set-some-flags},$
 $\text{unary},$
 $\text{direct-a},$
 $\text{direct-b},$
 $\text{side-effect-a},$

```

      side-effect-b,
      all-t-regs-address,
      dtack-,
      hold-)
= next-state (decoded-state,
              store,
              set-some-flags,
              unary,
              direct-a,
              direct-b,
              side-effect-a,
              side-effect-b,
              all-t-regs-address,
              dtack-,
              hold-))

```

DEFINITION:

NEXT-STATE*

```

= cons ('next-state,
        cons (append (indices ('s, 0, 32),
                          '(store set-some-flags unary direct-a
                                direct-b side-effect-a side-effect-b
                                all-t-regs-address dtack- hold-)),
              cons ('(w-17 w-18 w-19 w-22 w-24 w-27 w-29 w-44
                      w-47 w-48 w-49 w-52 w-61 w-62 w-63 w-66
                      w-74 w-75 w-76 w-79 w-82 w-83 w-89 w-93
                      w-98 w-99 w-112 w-113 w-105 w-106 w-110
                      w-111),
                  cons (append (list (list ('r-0,
                                              '(fetch0),
                                              'id,
                                              list (index ('s, 0))),
                                      list ('r-1,
                                              '(fetch1),
                                              'id,
                                              list (index ('s, 1))),
                                      list ('r-2,
                                              '(fetch2),
                                              'id,
                                              list (index ('s, 2))),
                                      list ('r-3,
                                              '(fetch3),
                                              'id,
                                              list (index ('s, 3))),

```

```

list ('r-4,
      '(decode),
      'id,
      list (index ('s, 4))),
list ('r-5,
      '(rega),
      'id,
      list (index ('s, 5))),
list ('r-6,
      '(regb),
      'id,
      list (index ('s, 6))),
list ('r-7,
      '(update),
      'id,
      list (index ('s, 7))),
list ('r-8,
      '(reada0),
      'id,
      list (index ('s, 8))),
list ('r-9,
      '(reada1),
      'id,
      list (index ('s, 9))),
list ('r-10,
      '(reada2),
      'id,
      list (index ('s, 10))),
list ('r-11,
      '(reada3),
      'id,
      list (index ('s, 11))),
list ('r-12,
      '(readb0),
      'id,
      list (index ('s, 12))),
list ('r-13,
      '(readb1),
      'id,
      list (index ('s, 13))),
list ('r-14,
      '(readb2),
      'id,
      list (index ('s, 14))),

```



```

list ('r-15,
      '(readb3),
      'id,
      list (index ('s, 15))),
list ('r-16,
      '(write0),
      'id,
      list (index ('s, 16))),
list ('r-17,
      '(write1),
      'id,
      list (index ('s, 17))),
list ('r-18,
      '(write2),
      'id,
      list (index ('s, 18))),
list ('r-19,
      '(write3),
      'id,
      list (index ('s, 19))),
list ('r-20,
      '(sefa0),
      'id,
      list (index ('s, 20))),
list ('r-21,
      '(sefa1),
      'id,
      list (index ('s, 21))),
list ('r-22,
      '(sefb0),
      'id,
      list (index ('s, 22))),
list ('r-23,
      '(sefb1),
      'id,
      list (index ('s, 23))),
list ('r-24,
      '(hold0),
      'id,
      list (index ('s, 24))),
list ('r-25,
      '(hold1),
      'id,
      list (index ('s, 25))),

```

```

list ('r-26,
      '(v11010),
      'id,
      list (index ('s, 26))),
list ('r-27,
      '(v11011),
      'id,
      list (index ('s, 27))),
list ('r-28,
      '(reset0),
      'id,
      list (index ('s, 28))),
list ('r-29,
      '(reset1),
      'id,
      list (index ('s, 29))),
list ('r-30,
      '(reset2),
      'id,
      list (index ('s, 30))),
list ('r-31,
      '(v11111),
      'id,
      list (index ('s, 31))),
'((g-111 (w-111) vss nil)
 (g-109 (w-109) b-not (reset1))
 (g-107
  (w-107)
  b-not
  (all-t-regs-address))
 (g-108
  (w-108)
  b-nand
  (w-107 reset2))
 (g-110
  (w-110)
  b-nand
  (w-108 w-109))
 (g-106 (w-106) id (reset0))
 (g-104 (w-104) b-not (v11010))
 (g-103 (w-103) b-not (v11011))
 (g-102 (w-102) b-not (v11111))
 (g-105
  (w-105)

```

```

b-nand3
(w-102 w-103 w-104))
(g-99
(w-99)
b-and
(hold- hold0))
(g-96 (w-96) b-not (hold-))
(g-97
(w-97)
b-nand
(w-96 fetch1))
(g-95
(w-95)
b-nand
(w-96 hold0))
(g-98
(w-98)
b-nand
(w-95 w-97))
(g-91
(w-91)
b-and
(side-effect-b unary))
(g-92
(w-92)
b-nand
(w-91 update))
(g-90 (w-90) b-not (sefb0))
(g-93
(w-93)
b-nand
(w-90 w-92))
(g-86
(w-86)
b-or
(store set-some-flags))
(g-87 (w-87) b-not (w-86))
(g-85
(w-85)
b-not
(side-effect-a))
(g-88
(w-88)
b-nand4

```

```

(side-effect-b w-85 w-87 decode))
(g-84
 (w-84)
 b-nand
 (side-effect-b sefa1))
(g-89
 (w-89)
 b-nand
 (w-84 w-88))
(g-83 (w-83) id (sefa0))
(g-82
 (w-82)
 b-and3
 (side-effect-a w-87 decode))
(g-78 (w-78) b-not (write2))
(g-77
 (w-77)
 b-nand
 (dtack- write3))
(g-79
 (w-79)
 b-nand
 (w-77 w-78))
(g-76 (w-76) id (write1))
(g-75 (w-75) id (write0))
(g-72 (w-72) b-not (direct-b))
(g-73
 (w-73)
 b-nand3
 (store w-72 rega))
(g-70 (w-70) b-not (dtack-))
(g-71
 (w-71)
 b-nand5
 (store unary w-72 w-70 reada3))
(g-68
 (w-68)
 b-nand3
 (store w-70 readb3))
(g-74
 (w-74)
 b-nand3
 (w-68 w-71 w-73))
(g-65 (w-65) b-not (readb2))

```

```

(g-64
 (w-64)
 b-nand
 (dtack- readb3))
(g-66
 (w-66)
 b-nand
 (w-64 w-65))
(g-63 (w-63) id (readb1))
(g-62 (w-62) id (readb0))
(g-57
 (w-57)
 b-or
 (direct-b unary))
(g-58 (w-58) b-not (w-57))
(g-60
 (w-60)
 b-nand4
 (w-58 direct-a w-86 decode))
(g-53 (w-53) b-not (unary))
(g-56
 (w-56)
 b-nand4
 (w-53 w-72 w-70 reada3))
(g-61
 (w-61)
 b-nand
 (w-56 w-60))
(g-51 (w-51) b-not (reada2))
(g-50
 (w-50)
 b-nand
 (dtack- reada3))
(g-52
 (w-52)
 b-nand
 (w-50 w-51))
(g-49 (w-49) id (reada1))
(g-48 (w-48) id (reada0))
(g-45 (w-45) b-not (direct-a))
(g-47
 (w-47)
 b-and3
 (w-45 w-86 decode))

```

```

(g-43
  (w-43)
  b-nand3
  (unary direct-b rega))
(g-40 (w-40) b-not (store))
(g-42
  (w-42)
  b-nand3
  (w-40 w-72 rega))
(g-39 (w-39) b-not (regb))
(g-38
  (w-38)
  b-nand3
  (direct-b w-70 reada3))
(g-36
  (w-36)
  b-nand5
  (w-40 unary w-72 w-70 reada3))
(g-32
  (w-32)
  b-nand3
  (w-40 w-70 readb3))
(g-44
  (w-44)
  b-nand6
  (w-32 w-36 w-38 w-39 w-42 w-43))
(g-29
  (w-29)
  b-and3
  (w-53 direct-b rega))
(g-27
  (w-27)
  b-and4
  (w-57 direct-a w-86 decode))
(g-24
  (w-24)
  b-and
  (w-70 fetch3))
(g-21 (w-21) b-not (fetch2))
(g-20
  (w-20)
  b-nand
  (dtack- fetch3))
(g-22

```

```

(w-22)
b-nand
(w-20 w-21))
(g-19
(w-19)
b-and
(hold- fetch1))
(g-18 (w-18) id (fetch0))
(g-11
(w-11)
b-not
(side-effect-b))
(g-15
(w-15)
b-nand4
(w-11 w-85 w-87 decode))
(g-9 (w-9) b-not (w-91))
(g-10
(w-10)
b-nand
(w-9 update))
(g-7
(w-7)
b-nand
(w-70 write3))
(g-16
(w-16)
b-and3
(w-7 w-10 w-15))
(g-4
(w-4)
b-nand
(w-11 sefa1))
(g-2 (w-2) b-not (sefb1))
(g-1 (w-1) b-not (hold1))
(g-0
(w-0)
b-nand
(all-t-regs-address reset2))
(g-5
(w-5)
b-and4
(w-0 w-1 w-2 w-4))
(g-17

```

```

(w-17)
b-nand
(w-5 w-16))
(g-112 (w-112) id (w-111))
(g-113 (w-113) id (w-111))),
'(nil))))))

```

DEFINITION:

next-state& (*netlist*)

```

= ((lookup-module ('next-state, netlist) = NEXT-STATE*)
  ∧ vss& (delete-module ('next-state, netlist))
  ∧ b-or& (delete-module ('next-state, netlist))
  ∧ b-nand5& (delete-module ('next-state, netlist))
  ∧ b-nand3& (delete-module ('next-state, netlist))
  ∧ b-nand6& (delete-module ('next-state, netlist))
  ∧ b-and& (delete-module ('next-state, netlist))
  ∧ b-nand4& (delete-module ('next-state, netlist))
  ∧ b-and3& (delete-module ('next-state, netlist))
  ∧ b-not& (delete-module ('next-state, netlist))
  ∧ b-and4& (delete-module ('next-state, netlist))
  ∧ b-nand& (delete-module ('next-state, netlist))
  ∧ id& (delete-module ('next-state, netlist)))

```

EVENT: Enable next-state&; name this event 'next-state&-off'.

DEFINITION:

NEXT-STATE\$NETLIST

```

= cons (NEXT-STATE*,
  ((((((((((VSS$NETLIST ∪ B-OR$NETLIST) ∪ B-NAND5$NETLIST)
    ∪ B-NAND3$NETLIST)
    ∪ B-NAND6$NETLIST)
    ∪ B-AND$NETLIST)
    ∪ B-NAND4$NETLIST)
    ∪ B-AND3$NETLIST)
    ∪ B-NOT$NETLIST)
    ∪ B-AND4$NETLIST)
    ∪ B-NAND$NETLIST)
    ∪ ID$NETLIST)

```

THEOREM: bvp-length-next-state

```

(length (next-state (decoded-state,
                    store,
                    set-some-flags,
                    unary,

```


$$\begin{aligned}
& \text{direct-}a, \\
& \text{direct-}b, \\
& \text{side-effect-}a, \\
& \text{side-effect-}b, \\
& \text{all-t-regs-address}, \\
& \text{dtack-}, \\
& \text{hold-})) \\
= & \text{32}) \\
\wedge & ((\text{bvp}(\text{decoded-state}) \wedge (\text{length}(\text{decoded-state}) = 32)) \\
& \rightarrow \text{bvp}(\text{next-state}(\text{decoded-state}, \\
& \quad \text{store}, \\
& \quad \text{set-some-flags}, \\
& \quad \text{unary}, \\
& \quad \text{direct-}a, \\
& \quad \text{direct-}b, \\
& \quad \text{side-effect-}a, \\
& \quad \text{side-effect-}b, \\
& \quad \text{all-t-regs-address}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-})))
\end{aligned}$$

THEOREM: properp-length-f\$next-state

$$\begin{aligned}
& (\text{length}(\text{f\$next-state}(\text{decoded-state}, \\
& \quad \text{store}, \\
& \quad \text{set-some-flags}, \\
& \quad \text{unary}, \\
& \quad \text{direct-}a, \\
& \quad \text{direct-}b, \\
& \quad \text{side-effect-}a, \\
& \quad \text{side-effect-}b, \\
& \quad \text{all-t-regs-address}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-})) \\
= & \text{32}) \\
\wedge & \text{properp}(\text{f\$next-state}(\text{decoded-state}, \\
& \quad \text{store}, \\
& \quad \text{set-some-flags}, \\
& \quad \text{unary}, \\
& \quad \text{direct-}a, \\
& \quad \text{direct-}b, \\
& \quad \text{side-effect-}a, \\
& \quad \text{side-effect-}b, \\
& \quad \text{all-t-regs-address}, \\
& \quad \text{dtack-},
\end{aligned}$$

hold-))

THEOREM: next-state\$value
 (next-state& (*netlist*)
 ^ properp (*decoded-state*)
 ^ (length (*decoded-state*) = 32))
 → (dual-eval (0,
 'next-state,
 append (*decoded-state*,
 list (*store*,
 set-some-flags,
 unary,
 direct-a,
 direct-b,
 side-effect-a,
 side-effect-b,
 all-t-regs-address,
 dtack-,
 hold-)),
 state,
 netlist)
 = f\$next-state (*decoded-state*,
 store,
 set-some-flags,
 unary,
 direct-a,
 direct-b,
 side-effect-a,
 side-effect-b,
 all-t-regs-address,
 dtack-,
 hold-))

EVENT: Enable next-state\$value; name this event 'next-state\$value-off'.

```
(DEFN CV
  (DECODED-STATE RN-A RN-B OP-CODE PC-REG REGS-ADDRESS SET-FLAGS STORE C
    A-IMMEDIATE-P RW- DIRECT-A SIDE-EFFECT-A SIDE-EFFECT-B
    PRE-DEC-A PRE-DEC-B)
  (CONS
    (B-AND (B-NAND (B-NOT RW-)
      (NTH 0 DECODED-STATE))
```

```

        (B-NOR3 (NTH 17 DECODED-STATE)
                 (NTH 18 DECODED-STATE)
                 (NTH 19 DECODED-STATE)))
(CONS
 (B-NOR8 (NTH 2 DECODED-STATE)
          (NTH 3 DECODED-STATE)
          (NTH 10 DECODED-STATE)
          (NTH 11 DECODED-STATE)
          (NTH 14 DECODED-STATE)
          (NTH 15 DECODED-STATE)
          (NTH 18 DECODED-STATE)
          (NTH 19 DECODED-STATE))
(CONS
 (B-NOT (NTH 24 DECODED-STATE))
(CONS
 (B-NAND5 (B-NAND STORE (NTH 7 DECODED-STATE))
           (B-NAND SIDE-EFFECT-A
                    (NTH 9 DECODED-STATE))
           (B-NAND3 (B-NOT STORE)
                     SIDE-EFFECT-B
                     (NTH 14 DECODED-STATE))
           (B-NAND SIDE-EFFECT-B
                    (NTH 17 DECODED-STATE))
           (B-NOR5 (NTH 2 DECODED-STATE)
                    (NTH 21 DECODED-STATE)
                    (NTH 23 DECODED-STATE)
                    (NTH 28 DECODED-STATE)
                    (NTH 30 DECODED-STATE))))
(CONS
 (B-NAND (B-NAND DIRECT-A
              (NTH 13 DECODED-STATE))
          (B-NOR6 (NTH 0 DECODED-STATE)
                   (NTH 5 DECODED-STATE)
                   (NTH 8 DECODED-STATE)
                   (NTH 11 DECODED-STATE)
                   (NTH 20 DECODED-STATE)
                   (NTH 29 DECODED-STATE))))
(CONS
 (B-NAND (B-NOR4 (NTH 6 DECODED-STATE)
                  (NTH 7 DECODED-STATE)
                  (NTH 10 DECODED-STATE)
                  (NTH 12 DECODED-STATE))
          (B-NOR4 (NTH 15 DECODED-STATE)
                   (NTH 16 DECODED-STATE))

```

```

(NTH 22 DECODED-STATE)
(NTH 29 DECODED-STATE)))
(CONS
  (B-OR (NTH 3 DECODED-STATE)
    (NTH 29 DECODED-STATE))
  (CONS
    (B-OR (NTH 16 DECODED-STATE)
      (NTH 28 DECODED-STATE))
    (CONS
      (B-NAND (B-NOR3 (NTH 0 DECODED-STATE)
        (NTH 8 DECODED-STATE)
        (NTH 12 DECODED-STATE))
        (B-NOR (NTH 16 DECODED-STATE)
          (NTH 29 DECODED-STATE)))
      (CONS
        (B-OR3 (NTH 0 DECODED-STATE)
          (NTH 24 DECODED-STATE)
          (NTH 28 DECODED-STATE))
        (CONS
          (B-OR (NTH 24 DECODED-STATE)
            (NTH 28 DECODED-STATE))
          (CONS
            (B-OR3 (NTH 3 DECODED-STATE)
              (NTH 11 DECODED-STATE)
              (NTH 15 DECODED-STATE))
            (CONS
              (B-NAND (B-NAND PRE-DEC-A
                (NTH 8 DECODED-STATE))
                (B-NAND PRE-DEC-B
                  (B-OR (NTH 12 DECODED-STATE)
                    (NTH 16 DECODED-STATE))))
              (CONS
                (B-AND A-IMMEDIATE-P
                  (B-OR (NTH 5 DECODED-STATE)
                    (NTH 13 DECODED-STATE)))
                (CONS
                  (CARRY-IN-HELP
                    (CONS C
                      (CONS
                        (B-OR4 (NTH 0 DECODED-STATE)
                          (NTH 28 DECODED-STATE)
                          (NTH 29 DECODED-STATE)
                          (NTH 30 DECODED-STATE))
                        (V-IF (B-NOR4 (NTH 2 DECODED-STATE)

```

```

        (B-OR (NTH 9 DECODED-STATE)
              (NTH 21 DECODED-STATE))
        (B-OR3 (NTH 14 DECODED-STATE)
              (NTH 17 DECODED-STATE)
              (NTH 23 DECODED-STATE))
        (B-OR4 (NTH 0 DECODED-STATE)
              (NTH 28 DECODED-STATE)
              (NTH 29 DECODED-STATE)
              (NTH 30 DECODED-STATE)))
OP-CODE
(V-IF (B-NAND (B-NAND (B-OR (NTH 9 DECODED-STATE)
                          (NTH 21 DECODED-STATE))
                    PRE-DEC-A)
      (B-NAND (B-OR3 (NTH 14 DECODED-STATE)
                    (NTH 17 DECODED-STATE)
                    (NTH 23 DECODED-STATE))
            PRE-DEC-B))
      (ALU-DEC-OP)
      (ALU-INC-OP))))))
(CONS
 (B-OR4 (NTH 0 DECODED-STATE)
       (NTH 28 DECODED-STATE)
       (NTH 29 DECODED-STATE)
       (NTH 30 DECODED-STATE))
 (APPEND
  (ENCODE-32 (NTH 0 DECODED-STATE)
            (NTH 1 DECODED-STATE)
            (NTH 2 DECODED-STATE)
            (NTH 3 DECODED-STATE)
            (NTH 4 DECODED-STATE)
            (NTH 5 DECODED-STATE)
            (NTH 6 DECODED-STATE)
            (NTH 7 DECODED-STATE)
            (NTH 8 DECODED-STATE)
            (NTH 9 DECODED-STATE)
            (NTH 10 DECODED-STATE)
            (NTH 11 DECODED-STATE)
            (NTH 12 DECODED-STATE)
            (NTH 13 DECODED-STATE)
            (NTH 14 DECODED-STATE)
            (NTH 15 DECODED-STATE)
            (NTH 16 DECODED-STATE)
            (NTH 17 DECODED-STATE)
            (NTH 18 DECODED-STATE)

```

```

(NTH 19 DECODED-STATE)
(NTH 20 DECODED-STATE)
(NTH 21 DECODED-STATE)
(NTH 22 DECODED-STATE)
(NTH 23 DECODED-STATE)
(NTH 24 DECODED-STATE)
(NTH 25 DECODED-STATE)
(NTH 26 DECODED-STATE)
(NTH 27 DECODED-STATE)
(NTH 28 DECODED-STATE)
(NTH 29 DECODED-STATE)
(NTH 30 DECODED-STATE)
(NTH 31 DECODED-STATE))

(APPEND
(V-IF (B-NOR (NTH 7 DECODED-STATE)
              (NTH 16 DECODED-STATE)))
(V-IF (NTH 28 DECODED-STATE)
      (MAKE-LIST 4 T)
      (MAKE-LIST 4 F))
SET-FLAGS)

(APPEND
(V-IF (B-NAND (B-NOR3 (NTH 5 DECODED-STATE)
                      (NTH 8 DECODED-STATE)
                      (NTH 9 DECODED-STATE))
          (B-NOR3 (NTH 13 DECODED-STATE)
                  (NTH 20 DECODED-STATE)
                  (NTH 21 DECODED-STATE))))

RN-A
(V-IF (B-NAND (B-NOR5 (NTH 6 DECODED-STATE)
                      (NTH 7 DECODED-STATE)
                      (NTH 10 DECODED-STATE)
                      (NTH 12 DECODED-STATE)
                      (NTH 14 DECODED-STATE))
          (B-NOR4 (NTH 16 DECODED-STATE)
                  (NTH 17 DECODED-STATE)
                  (NTH 22 DECODED-STATE)
                  (NTH 23 DECODED-STATE))))

RN-B
(V-IF (B-OR (NTH 28 DECODED-STATE)
           (NTH 29 DECODED-STATE))
      (MAKE-LIST 4 F)
      (V-IF (NTH 30 DECODED-STATE)
            (V-INC REGS-ADDRESS)
            PC-REG))))

```

```

(APPEND
(V-IF (B-NOR4 (NTH 2 DECODED-STATE)
              (B-OR (NTH 9 DECODED-STATE)
                    (NTH 21 DECODED-STATE))
              (B-OR3 (NTH 14 DECODED-STATE)
                    (NTH 17 DECODED-STATE)
                    (NTH 23 DECODED-STATE))
              (B-OR4 (NTH 0 DECODED-STATE)
                    (NTH 28 DECODED-STATE)
                    (NTH 29 DECODED-STATE)
                    (NTH 30 DECODED-STATE)))

OP-CODE
(V-IF (B-NAND (B-NAND (B-OR (NTH 9 DECODED-STATE)
                          (NTH 21 DECODED-STATE))
                PRE-DEC-A)
      (B-NAND (B-OR3 (NTH 14 DECODED-STATE)
                    (NTH 17 DECODED-STATE)
                    (NTH 23 DECODED-STATE))
            PRE-DEC-B))
      (ALU-DEC-OP)
      (ALU-INC-OP)))

(MPG
(CONS
(B-OR4 (NTH 0 DECODED-STATE)
      (NTH 28 DECODED-STATE)
      (NTH 29 DECODED-STATE)
      (NTH 30 DECODED-STATE))

(CONS
(B-OR3 (NTH 14 DECODED-STATE)
      (NTH 17 DECODED-STATE)
      (NTH 23 DECODED-STATE))

(V-IF
(B-NOR4 (NTH 2 DECODED-STATE)
        (B-OR (NTH 9 DECODED-STATE)
              (NTH 21 DECODED-STATE))
        (B-OR3 (NTH 14 DECODED-STATE)
              (NTH 17 DECODED-STATE)
              (NTH 23 DECODED-STATE))
        (B-OR4 (NTH 0 DECODED-STATE)
              (NTH 28 DECODED-STATE)
              (NTH 29 DECODED-STATE)
              (NTH 30 DECODED-STATE)))

OP-CODE
(V-IF (B-NAND (B-NAND (B-OR (NTH 9 DECODED-STATE)

```

```

(NTH 21 DECODED-STATE))
PRE-DEC-A)
(B-NAND (B-OR3 (NTH 14 DECODED-STATE)
(NTH 17 DECODED-STATE)
(NTH 23 DECODED-STATE))
PRE-DEC-B))
(ALU-DEC-OP)
(ALU-INC-OP))))))))))))))))))))))

```

EVENT: Enable cv; name this event 'cv-off'.

```

(DEFN F$CV
(DECODED-STATE RN-A RN-B OP-CODE PC-REG REGS-ADDRESS SET-FLAGS STORE C
A-IMMEDIATE-P RW- DIRECT-A SIDE-EFFECT-A SIDE-EFFECT-B
PRE-DEC-A PRE-DEC-B)
(CONS
(F-AND (F-NAND (F-NOT RW-)
(NTH 0 DECODED-STATE))
(F-NOR3 (NTH 17 DECODED-STATE)
(NTH 18 DECODED-STATE)
(NTH 19 DECODED-STATE)))
(CONS
(F-NOR8 (NTH 2 DECODED-STATE)
(NTH 3 DECODED-STATE)
(NTH 10 DECODED-STATE)
(NTH 11 DECODED-STATE)
(NTH 14 DECODED-STATE)
(NTH 15 DECODED-STATE)
(NTH 18 DECODED-STATE)
(NTH 19 DECODED-STATE))
(CONS
(F-NOT (NTH 24 DECODED-STATE))
(CONS
(F-NAND5 (F-NAND STORE (NTH 7 DECODED-STATE))
(F-NAND SIDE-EFFECT-A
(NTH 9 DECODED-STATE))
(F-NAND3 (F-NOT STORE)
SIDE-EFFECT-B
(NTH 14 DECODED-STATE))
(F-NAND SIDE-EFFECT-B
(NTH 17 DECODED-STATE))
(F-NOR5 (NTH 2 DECODED-STATE)

```



```

(NTH 21 DECODED-STATE)
(NTH 23 DECODED-STATE)
(NTH 28 DECODED-STATE)
(NTH 30 DECODED-STATE)))
(CONS
  (F-NAND (F-NAND DIRECT-A
    (NTH 13 DECODED-STATE))
    (F-NOR6 (NTH 0 DECODED-STATE)
      (NTH 5 DECODED-STATE)
      (NTH 8 DECODED-STATE)
      (NTH 11 DECODED-STATE)
      (NTH 20 DECODED-STATE)
      (NTH 29 DECODED-STATE))))
  (CONS
    (F-NAND (F-NOR4 (NTH 6 DECODED-STATE)
      (NTH 7 DECODED-STATE)
      (NTH 10 DECODED-STATE)
      (NTH 12 DECODED-STATE))
      (F-NOR4 (NTH 15 DECODED-STATE)
        (NTH 16 DECODED-STATE)
        (NTH 22 DECODED-STATE)
        (NTH 29 DECODED-STATE))))
    (CONS
      (F-OR (NTH 3 DECODED-STATE)
        (NTH 29 DECODED-STATE))
      (CONS
        (F-OR (NTH 16 DECODED-STATE)
          (NTH 28 DECODED-STATE))
        (CONS
          (F-NAND (F-NOR3 (NTH 0 DECODED-STATE)
            (NTH 8 DECODED-STATE)
            (NTH 12 DECODED-STATE))
            (F-NOR (NTH 16 DECODED-STATE)
              (NTH 29 DECODED-STATE))))
          (CONS
            (F-OR3 (NTH 0 DECODED-STATE)
              (NTH 24 DECODED-STATE)
              (NTH 28 DECODED-STATE))
            (CONS
              (F-OR (NTH 24 DECODED-STATE)
                (NTH 28 DECODED-STATE))
              (CONS
                (F-OR3 (NTH 3 DECODED-STATE)
                  (NTH 11 DECODED-STATE)

```

```

(NTH 15 DECODED-STATE))
(CONS
  (F-NAND (F-NAND PRE-DEC-A
            (NTH 8 DECODED-STATE))
    (F-NAND PRE-DEC-B
      (F-OR (NTH 12 DECODED-STATE)
            (NTH 16 DECODED-STATE))))))
(CONS
  (F-AND A-IMMEDIATE-P
    (F-OR (NTH 5 DECODED-STATE)
          (NTH 13 DECODED-STATE))))
(CONS
  (F$CARRY-IN-HELP
    (CONS C
      (CONS
        (F-OR4 (NTH 0 DECODED-STATE)
              (NTH 28 DECODED-STATE)
              (NTH 29 DECODED-STATE)
              (NTH 30 DECODED-STATE))
        (F$SELECT-OP-CODE
          (F-NOR4 (NTH 2 DECODED-STATE)
                (F-OR (NTH 9 DECODED-STATE)
                      (NTH 21 DECODED-STATE))
                (F-OR3 (NTH 14 DECODED-STATE)
                      (NTH 17 DECODED-STATE)
                      (NTH 23 DECODED-STATE))
                (F-OR4 (NTH 0 DECODED-STATE)
                      (NTH 28 DECODED-STATE)
                      (NTH 29 DECODED-STATE)
                      (NTH 30 DECODED-STATE))))
          (F-NAND (F-NAND (F-OR (NTH 9 DECODED-STATE)
                                (NTH 21 DECODED-STATE))
                      PRE-DEC-A)
                    (F-NAND (F-OR3 (NTH 14 DECODED-STATE)
                                (NTH 17 DECODED-STATE)
                                (NTH 23 DECODED-STATE))
                      PRE-DEC-B))
          OP-CODE))))))
(CONS
  (F-OR4 (NTH 0 DECODED-STATE)
        (NTH 28 DECODED-STATE)
        (NTH 29 DECODED-STATE)
        (NTH 30 DECODED-STATE))
  (APPEND

```

```

(F$ENCODE-32 (NTH 0 DECODED-STATE)
              (NTH 1 DECODED-STATE)
              (NTH 2 DECODED-STATE)
              (NTH 3 DECODED-STATE)
              (NTH 4 DECODED-STATE)
              (NTH 5 DECODED-STATE)
              (NTH 6 DECODED-STATE)
              (NTH 7 DECODED-STATE)
              (NTH 8 DECODED-STATE)
              (NTH 9 DECODED-STATE)
              (NTH 10 DECODED-STATE)
              (NTH 11 DECODED-STATE)
              (NTH 12 DECODED-STATE)
              (NTH 13 DECODED-STATE)
              (NTH 14 DECODED-STATE)
              (NTH 15 DECODED-STATE)
              (NTH 16 DECODED-STATE)
              (NTH 17 DECODED-STATE)
              (NTH 18 DECODED-STATE)
              (NTH 19 DECODED-STATE)
              (NTH 20 DECODED-STATE)
              (NTH 21 DECODED-STATE)
              (NTH 22 DECODED-STATE)
              (NTH 23 DECODED-STATE)
              (NTH 24 DECODED-STATE)
              (NTH 25 DECODED-STATE)
              (NTH 26 DECODED-STATE)
              (NTH 27 DECODED-STATE)
              (NTH 28 DECODED-STATE)
              (NTH 29 DECODED-STATE)
              (NTH 30 DECODED-STATE)
              (NTH 31 DECODED-STATE))
(APPEND
 (FV-IF (F-NOR (NTH 7 DECODED-STATE)
               (NTH 16 DECODED-STATE))
  (MAKE-LIST 4
    (THREEFIX (NTH 28 DECODED-STATE)))
  SET-FLAGS)
(APPEND
 (FV-IF (F-NAND (F-NOR3 (NTH 5 DECODED-STATE)
                        (NTH 8 DECODED-STATE)
                        (NTH 9 DECODED-STATE))
              (F-NOR3 (NTH 13 DECODED-STATE)
                      (NTH 20 DECODED-STATE))

```

```

(NTH 21 DECODED-STATE)))

RN-A
(FV-IF (F-NAND (F-NOR5 (NTH 6 DECODED-STATE)
(NTH 7 DECODED-STATE)
(NTH 10 DECODED-STATE)
(NTH 12 DECODED-STATE)
(NTH 14 DECODED-STATE))
(F-NOR4 (NTH 16 DECODED-STATE)
(NTH 17 DECODED-STATE)
(NTH 22 DECODED-STATE)
(NTH 23 DECODED-STATE))))

RN-B
(F$V-IF-F-4 (F-OR (NTH 28 DECODED-STATE)
(NTH 29 DECODED-STATE))
(FV-IF (NTH 30 DECODED-STATE)
(F$V-INC4$V REGS-ADDRESS)
PC-REG))))

(APPEND
(F$SELECT-OP-CODE
(F-NOR4 (NTH 2 DECODED-STATE)
(F-OR (NTH 9 DECODED-STATE)
(NTH 21 DECODED-STATE))
(F-OR3 (NTH 14 DECODED-STATE)
(NTH 17 DECODED-STATE)
(NTH 23 DECODED-STATE))
(F-OR4 (NTH 0 DECODED-STATE)
(NTH 28 DECODED-STATE)
(NTH 29 DECODED-STATE)
(NTH 30 DECODED-STATE)))
(F-NAND (F-NAND (F-OR (NTH 9 DECODED-STATE)
(NTH 21 DECODED-STATE))
PRE-DEC-A)
(F-NAND (F-OR3 (NTH 14 DECODED-STATE)
(NTH 17 DECODED-STATE)
(NTH 23 DECODED-STATE))
PRE-DEC-B))
OP-CODE)

(F$MPG
(CONS
(F-OR4 (NTH 0 DECODED-STATE)
(NTH 28 DECODED-STATE)
(NTH 29 DECODED-STATE)
(NTH 30 DECODED-STATE))
(CONS

```

```

(F-OR3 (NTH 14 DECODED-STATE)
      (NTH 17 DECODED-STATE)
      (NTH 23 DECODED-STATE))
(F$SELECT-OP-CODE
  (F-NOR4 (NTH 2 DECODED-STATE)
    (F-OR (NTH 9 DECODED-STATE)
      (NTH 21 DECODED-STATE))
    (F-OR3 (NTH 14 DECODED-STATE)
      (NTH 17 DECODED-STATE)
      (NTH 23 DECODED-STATE))
    (F-OR4 (NTH 0 DECODED-STATE)
      (NTH 28 DECODED-STATE)
      (NTH 29 DECODED-STATE)
      (NTH 30 DECODED-STATE)))
  (F-NAND (F-NAND (F-OR (NTH 9 DECODED-STATE)
    (NTH 21 DECODED-STATE))
    PRE-DEC-A)
    (F-NAND (F-OR3 (NTH 14 DECODED-STATE)
      (NTH 17 DECODED-STATE)
      (NTH 23 DECODED-STATE))
    PRE-DEC-B))
  OP-CODE))))))))))))))))))))))

```

EVENT: Enable f\$cv; name this event ‘f\$cv-off’.

THEOREM: make-list-crock-for-f\$cv=cv

```

boolp (v)
→ (make-list (4, v)
  = if v then list (t, t, t, t)
  else list (f, f, f, f) endif)

```

EVENT: Enable make-list-crock-for-f\$cv=cv; name this event ‘make-list-crock-for-f\$cv=cv-off’.

THEOREM: f\$cv=cv

```

(bvp (decoded-state)
  ∧ (length (decoded-state) = 32)
  ∧ bvp (rn-a)
  ∧ (length (rn-a) = 4)
  ∧ bvp (rn-b)
  ∧ (length (rn-b) = 4)
  ∧ bvp (op-code)
  ∧ (length (op-code) = 4)

```

$$\begin{aligned}
& \wedge \text{bvp}(pc\text{-reg}) \\
& \wedge (\text{length}(pc\text{-reg}) = 4) \\
& \wedge \text{bvp}(regs\text{-address}) \\
& \wedge (\text{length}(regs\text{-address}) = 4) \\
& \wedge \text{bvp}(set\text{-flags}) \\
& \wedge (\text{length}(set\text{-flags}) = 4) \\
& \wedge \text{boolp}(store) \\
& \wedge \text{boolp}(c) \\
& \wedge \text{boolp}(a\text{-immediate}\text{-}p) \\
& \wedge \text{boolp}(rw\text{-}) \\
& \wedge \text{boolp}(direct\text{-}a) \\
& \wedge \text{boolp}(side\text{-effect}\text{-}a) \\
& \wedge \text{boolp}(side\text{-effect}\text{-}b) \\
& \wedge \text{boolp}(pre\text{-dec}\text{-}a) \\
& \wedge \text{boolp}(pre\text{-dec}\text{-}b) \\
\rightarrow & (\text{f\$cv}(\text{decoded}\text{-}state, \\
& \quad rn\text{-}a, \\
& \quad rn\text{-}b, \\
& \quad op\text{-}code, \\
& \quad pc\text{-reg}, \\
& \quad regs\text{-address}, \\
& \quad set\text{-flags}, \\
& \quad store, \\
& \quad c, \\
& \quad a\text{-immediate}\text{-}p, \\
& \quad rw\text{-}, \\
& \quad direct\text{-}a, \\
& \quad side\text{-effect}\text{-}a, \\
& \quad side\text{-effect}\text{-}b, \\
& \quad pre\text{-dec}\text{-}a, \\
& \quad pre\text{-dec}\text{-}b) \\
= & \text{cv}(\text{decoded}\text{-}state, \\
& \quad rn\text{-}a, \\
& \quad rn\text{-}b, \\
& \quad op\text{-}code, \\
& \quad pc\text{-reg}, \\
& \quad regs\text{-address}, \\
& \quad set\text{-flags}, \\
& \quad store, \\
& \quad c, \\
& \quad a\text{-immediate}\text{-}p, \\
& \quad rw\text{-}, \\
& \quad direct\text{-}a, \\
& \quad side\text{-effect}\text{-}a,
\end{aligned}$$

side-effect-b,
pre-dec-a,
pre-dec-b))

THEOREM: bvp-length-cv

$$\begin{aligned}
& (((\text{length}(\text{set-flags}) = 4) \\
& \quad \wedge (\text{length}(\text{rn-a}) = 4) \\
& \quad \wedge (\text{length}(\text{rn-b}) = 4) \\
& \quad \wedge (\text{length}(\text{op-code}) = 4)) \\
& \rightarrow (\text{length}(\text{cv}(\text{decoded-state}, \\
& \quad \text{rn-a}, \\
& \quad \text{rn-b}, \\
& \quad \text{op-code}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}, \\
& \quad \text{set-flags}, \\
& \quad \text{store}, \\
& \quad \text{c}, \\
& \quad \text{a-immediate-p}, \\
& \quad \text{rw-}, \\
& \quad \text{direct-a}, \\
& \quad \text{side-effect-a}, \\
& \quad \text{side-effect-b}, \\
& \quad \text{pre-dec-a}, \\
& \quad \text{pre-dec-b})) \\
& \quad = 40)) \\
& \wedge ((\text{bvp}(\text{decoded-state}) \wedge (\text{length}(\text{decoded-state}) = 32)) \\
& \quad \rightarrow \text{bvp}(\text{cv}(\text{decoded-state}, \\
& \quad \text{rn-a}, \\
& \quad \text{rn-b}, \\
& \quad \text{op-code}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}, \\
& \quad \text{set-flags}, \\
& \quad \text{store}, \\
& \quad \text{c}, \\
& \quad \text{a-immediate-p}, \\
& \quad \text{rw-}, \\
& \quad \text{direct-a}, \\
& \quad \text{side-effect-a}, \\
& \quad \text{side-effect-b}, \\
& \quad \text{pre-dec-a}, \\
& \quad \text{pre-dec-b})))
\end{aligned}$$

THEOREM: properp-length-f\$cv

$$\begin{aligned}
& (((\text{length}(\text{set-flags}) = 4) \\
& \quad \wedge (\text{length}(\text{rn-a}) = 4) \\
& \quad \wedge (\text{length}(\text{rn-b}) = 4) \\
& \quad \wedge (\text{length}(\text{op-code}) = 4)) \\
& \rightarrow (\text{length}(\text{f\$cv}(\text{decoded-state}, \\
& \quad \text{rn-a}, \\
& \quad \text{rn-b}, \\
& \quad \text{op-code}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}, \\
& \quad \text{set-flags}, \\
& \quad \text{store}, \\
& \quad \text{c}, \\
& \quad \text{a-immediate-p}, \\
& \quad \text{rw-}, \\
& \quad \text{direct-a}, \\
& \quad \text{side-effect-a}, \\
& \quad \text{side-effect-b}, \\
& \quad \text{pre-dec-a}, \\
& \quad \text{pre-dec-b})) \\
& \quad = 40)) \\
& \wedge ((\text{properp}(\text{decoded-state}) \wedge (\text{length}(\text{decoded-state}) = 32)) \\
& \quad \rightarrow \text{properp}(\text{f\$cv}(\text{decoded-state}, \\
& \quad \text{rn-a}, \\
& \quad \text{rn-b}, \\
& \quad \text{op-code}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}, \\
& \quad \text{set-flags}, \\
& \quad \text{store}, \\
& \quad \text{c}, \\
& \quad \text{a-immediate-p}, \\
& \quad \text{rw-}, \\
& \quad \text{direct-a}, \\
& \quad \text{side-effect-a}, \\
& \quad \text{side-effect-b}, \\
& \quad \text{pre-dec-a}, \\
& \quad \text{pre-dec-b})))
\end{aligned}$$

DEFINITION:

CV*

$$\begin{aligned}
& = \text{cons}('cv, \\
& \quad \text{cons}(\text{append8}(\text{indices}('decoded-state, 0, 32), \\
& \quad \quad \text{indices}('rn-a, 0, 4),
\end{aligned}$$


```

indices('rn-b, 0, 4),
indices('op-code, 0, 4),
indices('pc-reg, 0, 4),
indices('regs-address, 0, 4),
indices('set-flags, 0, 4),
'(store c a-immediate-p rw- direct-a
  side-effect-a side-effect-b pre-dec-a
  pre-dec-b)),
cons(append6(' (new-rw- strobe- hdack- we-regs
  we-a-reg we-b-reg we-i-reg
  we-data-out we-addr-out we-hold-
  we-pc-reg data-in-select
  dec-addr-out select-immediate alu-c
  alu-zero),
indices('state, 0, 5),
indices('we-flags, 0, 4),
indices('new-regs-address, 0, 4),
indices('alu-op, 0, 4),
indices('alu-mpg, 0, 7)),
cons(list(list('g-fetch0,
  '(fetch0),
  'id,
  list(index('decoded-state, 0))),
list('g-fetch1,
  '(fetch1),
  'id,
  list(index('decoded-state, 1))),
list('g-fetch2,
  '(fetch2),
  'id,
  list(index('decoded-state, 2))),
list('g-fetch3,
  '(fetch3),
  'id,
  list(index('decoded-state, 3))),
list('g-decode,
  '(decode),
  'id,
  list(index('decoded-state, 4))),
list('g-rega,
  '(rega),
  'id,
  list(index('decoded-state, 5))),
list('g-regb,

```

```

        'regb),
        'id,
        list(index('decoded-state, 6))),
list('g-update,
      'update),
      'id,
      list(index('decoded-state, 7))),
list('g-reada0,
      'reada0),
      'id,
      list(index('decoded-state, 8))),
list('g-reada1,
      'reada1),
      'id,
      list(index('decoded-state, 9))),
list('g-reada2,
      'reada2),
      'id,
      list(index('decoded-state, 10))),
list('g-reada3,
      'reada3),
      'id,
      list(index('decoded-state, 11))),
list('g-readb0,
      'readb0),
      'id,
      list(index('decoded-state, 12))),
list('g-readb1,
      'readb1),
      'id,
      list(index('decoded-state, 13))),
list('g-readb2,
      'readb2),
      'id,
      list(index('decoded-state, 14))),
list('g-readb3,
      'readb3),
      'id,
      list(index('decoded-state, 15))),
list('g-write0,
      'write0),
      'id,
      list(index('decoded-state, 16))),
list('g-write1,

```

```

        'write1),
        'id,
        list(index('decoded-state, 17))),
list('g-write2,
      'write2),
      'id,
      list(index('decoded-state, 18))),
list('g-write3,
      'write3),
      'id,
      list(index('decoded-state, 19))),
list('g-sefa0,
      'sefa0),
      'id,
      list(index('decoded-state, 20))),
list('g-sefa1,
      'sefa1),
      'id,
      list(index('decoded-state, 21))),
list('g-sefb0,
      'sefb0),
      'id,
      list(index('decoded-state, 22))),
list('g-sefb1,
      'sefb1),
      'id,
      list(index('decoded-state, 23))),
list('g-hold0,
      'hold0),
      'id,
      list(index('decoded-state, 24))),
list('g-hold1,
      'hold1),
      'id,
      list(index('decoded-state, 25))),
list('g-v11010,
      'v11010),
      'id,
      list(index('decoded-state, 26))),
list('g-v11011,
      'v11011),
      'id,
      list(index('decoded-state, 27))),
list('g-reset0,

```

```

        ' (reset0),
        ' id,
        list (index ('decoded-state, 28))),
list ('g-reset1,
      ' (reset1),
      ' id,
      list (index ('decoded-state, 29))),
list ('g-reset2,
      ' (reset2),
      ' id,
      list (index ('decoded-state, 30))),
list ('g-v11111,
      ' (v11111),
      ' id,
      list (index ('decoded-state, 31))),
' (g0 (store-) b-not (store)),
' (g1
  (alu-zero)
  b-or4
  (fetch0 reset0 reset1 reset2)),
' (g2
  (alu-swap)
  b-or3
  (readb2 write1 sefb1)),
' (g3
  (incdeca)
  b-or
  (reada1 sefa1)),
' (g4
  (s4)
  b-nor4
  (fetch2 incdeca alu-swap alu-zero)),
' (g5
  (s5)
  b-nand
  (incdeca pre-dec-a)),
' (g6
  (s6)
  b-nand
  (alu-swap pre-dec-b)),
' (g7 (s7) b-nand (s5 s6)),
list ('g8,
      indices ('alu-op, 0, 4),
      'select-op-code,

```

```

      cons('s4,
        cons('s7, indices('op-code, 0, 4)))),
'(g10 (s10) b-not (rw-)),
'(g11 (s11) b-nand (s10 fetch0)),
'(g12
  (s12)
  b-nor3
  (write1 write2 write3)),
'(g13 (new-rw-) b-and (s11 s12)),
'(g14
  (strobe-)
  b-nor8
  (fetch2 fetch3 reada2 reada3
    readb2 readb3 write2 write3)),
'(g15 (hdack-) b-not (hold0)),
'(g17
  (s17)
  b-nand
  (store update)),
'(g18
  (s18)
  b-nand
  (side-effect-a reada1)),
'(g19
  (s19)
  b-nand3
  (store- side-effect-b readb2)),
'(g20
  (s20)
  b-nand
  (side-effect-b write1)),
'(g21
  (s21)
  b-nor5
  (fetch2 sefa1 sefb1 reset0 reset2)),
'(g22
  (we-regs)
  b-nand5
  (s17 s18 s19 s20 s21)),
'(g23
  (s23)
  b-nand
  (direct-a readb1)),
'(g24

```

```

(s24)
b-nor6
(fetch0 rega reada0 reada3 sefa0
reset1)),
'(g25
(we-a-reg)
b-nand
(s23 s24)),
'(g26
(s26)
b-nor4
(regb update reada2 readb0)),
'(g27
(s27)
b-nor4
(readb3 write0 sefb0 reset1)),
'(g28
(we-b-reg)
b-nand
(s26 s27)),
'(g29
(we-i-reg)
b-or
(fetch3 reset1)),
'(g30
(we-data-out)
b-or
(write0 reset0)),
'(g31
(s31)
b-nor3
(fetch0 reada0 readb0)),
'(g32
(s32)
b-nor
(write0 reset1)),
'(g33
(we-addr-out)
b-nand
(s31 s32)),
'(g34
(we-hold-)
b-or3
(fetch0 hold0 reset0)),

```

```

'(g35
  (we-pc-reg)
  b-or
  (hold0 reset0)),
'(g36
  (data-in-select)
  b-or3
  (fetch3 reada3 readb3)),
'(g37
  (s37)
  b-nand
  (pre-dec-a reada0)),
'(g38 (s38) b-or (readb0 write0)),
'(g39
  (s39)
  b-nand
  (pre-dec-b s38)),
'(g40
  (dec-addr-out)
  b-nand
  (s37 s39)),
'(g41 (s41) b-or (rega readb1)),
'(g42
  (select-immediate)
  b-and
  (a-immediate-p s41)),
list('g43,
      '(alu-c),
      'carry-in-help,
      cons('c,
            cons('alu-zero,
                  indices('alu-op, 0, 4)))))
list('g44,
      indices('state, 0, 5),
      'encode-32,
      indices('decoded-state, 0, 32)),
cons('g45,
      cons(indices('fanout-reset0, 0, 4),
            '(fanout-4 (reset0)))),
'(g46
  (s46)
  b-nor
  (update write0)),
list('g47,

```

```

indices('we-flags, 0, 4),
index('tv-if,
      tree-number(make-tree(4))),
cons('s46,
      append(indices('fanout-reset0,
                     0,
                     4),
            indices('set-flags, 0, 4)))),
'(g48
  (s48)
  b-nor3
  (rega reada0 reada1)),
'(g49
  (s49)
  b-nor3
  (readb1 sefa0 sefa1)),
'(g50
  (select-rn-a)
  b-nand
  (s48 s49)),
'(g51
  (s51)
  b-nor5
  (regb update reada2 readb0 readb2)),
'(g52
  (s52)
  b-nor4
  (write0 write1 sefb0 sefb1)),
'(g53
  (select-rn-b)
  b-nand
  (s51 s52)),
'(g54
  (select-all-f)
  b-or
  (reset0 reset1)),
list('g55,
      indices('v-inc-regs-address, 0, 4),
      'v-inc4,
      indices('regs-address, 0, 4)),
list('g56,
      indices('s56, 0, 4),
      index('tv-if,
            tree-number(make-tree(4))),

```



```

cons('reset2,
      append(indices('v-inc-regs-address,
                     0,
                     4),
              indices('pc-reg, 0, 4))))),
list('g57,
      indices('s57, 0, 4),
      'v-if-f-4,
      cons('select-all-f,
            indices('s56, 0, 4))),
list('g58,
      indices('s58, 0, 4),
      index('tv-if,
             tree-number(make-tree(4))),
      cons('select-rn-b,
            append(indices('rn-b, 0, 4),
                    indices('s57, 0, 4))))),
list('g59,
      indices('new-regs-address, 0, 4),
      index('tv-if,
             tree-number(make-tree(4))),
      cons('select-rn-a,
            append(indices('rn-a, 0, 4),
                    indices('s58, 0, 4))))),
list('g60,
      indices('alu-mpg, 0, 7),
      'mpg,
      cons('alu-zero,
            cons('alu-swap,
                  indices('alu-op, 0, 4))))),
'(nil))))

```

DEFINITION:

$cv\&(netlist)$

```

= ((lookup-module('cv, netlist) = CV*)
   ^ id&(delete-module('cv, netlist))
   ^ b-not&(delete-module('cv, netlist))
   ^ b-or&(delete-module('cv, netlist))
   ^ b-or3&(delete-module('cv, netlist))
   ^ b-or4&(delete-module('cv, netlist))
   ^ b-nand&(delete-module('cv, netlist))
   ^ b-nand3&(delete-module('cv, netlist))
   ^ b-nand5&(delete-module('cv, netlist))
   ^ b-nor&(delete-module('cv, netlist))

```

```

 $\wedge$  b-nor3& (delete-module ('cv, netlist))
 $\wedge$  b-nor4& (delete-module ('cv, netlist))
 $\wedge$  b-nor5& (delete-module ('cv, netlist))
 $\wedge$  b-nor6& (delete-module ('cv, netlist))
 $\wedge$  b-nor8& (delete-module ('cv, netlist))
 $\wedge$  b-and& (delete-module ('cv, netlist))
 $\wedge$  select-op-code& (delete-module ('cv, netlist))
 $\wedge$  tv-if& (delete-module ('cv, netlist), make-tree (4))
 $\wedge$  carry-in-help& (delete-module ('cv, netlist))
 $\wedge$  encode-32& (delete-module ('cv, netlist))
 $\wedge$  fanout-4& (delete-module ('cv, netlist))
 $\wedge$  v-inc4& (delete-module ('cv, netlist))
 $\wedge$  v-if-f-4& (delete-module ('cv, netlist))
 $\wedge$  mpg& (delete-module ('cv, netlist)))

```

EVENT: Enable cv&; name this event 'cv&-off'.

DEFINITION:

```

CV$NETLIST
= cons (CV*,
      (((ID$NETLIST  $\cup$  B-NOT$NETLIST)
         $\cup$  (B-OR$NETLIST  $\cup$  (B-OR3$NETLIST  $\cup$  B-OR4$NETLIST)))
         $\cup$  ((B-NAND$NETLIST  $\cup$  B-NAND3$NETLIST)
           $\cup$  (B-NOR$NETLIST  $\cup$  B-NOR3$NETLIST))))
       $\cup$  (((B-NOR4$NETLIST  $\cup$  B-NOR5$NETLIST)
         $\cup$  (B-NOR6$NETLIST  $\cup$  B-NOR8$NETLIST))
         $\cup$  ((B-AND$NETLIST  $\cup$  SELECT-OP-CODE$NETLIST)
           $\cup$  (tv-if$netlist (make-tree (4))
             $\cup$  CARRY-IN-HELP$NETLIST))))))
       $\cup$  (((ENCODE-32$NETLIST  $\cup$  FANOUT-4$NETLIST)
         $\cup$  (V-INC4$NETLIST  $\cup$  V-IF-F-4$NETLIST))
         $\cup$  (MPG$NETLIST  $\cup$  B-NAND5$NETLIST)))

```

THEOREM: check-cv\$netlist

cv& (CV\$NETLIST)

THEOREM: cv\$value

```

(cv& (netlist)
 $\wedge$  properp (decoded-state)
 $\wedge$  (length (decoded-state) = 32)
 $\wedge$  properp (rn-a)
 $\wedge$  (length (rn-a) = 4)
 $\wedge$  properp (rn-b)
 $\wedge$  (length (rn-b) = 4)

```

```

^  properp (op-code)
^  (length (op-code) = 4)
^  properp (pc-reg)
^  (length (pc-reg) = 4)
^  properp (regs-address)
^  (length (regs-address) = 4)
^  properp (set-flags)
^  (length (set-flags) = 4))
→  (dual-eval (0,
               'cv,
               append (decoded-state,
                       append (rn-a,
                              append (rn-b,
                                      append (op-code,
                                              append (pc-reg,
                                                      append (regs-address,
                                                              append (set-flags,
                                                                      list (store,
                                                                           c,
                                                                           a-immediate-p,
                                                                           rw-,
                                                                           direct-a,
                                                                           side-effect-a,
                                                                           side-effect-b,
                                                                           pre-dec-a,
                                                                           pre-dec-b)))))))))
               state,
               netlist)
    =  f$cv (decoded-state,
            rn-a,
            rn-b,
            op-code,
            pc-reg,
            regs-address,
            set-flags,
            store,
            c,
            a-immediate-p,
            rw-,
            direct-a,
            side-effect-a,
            side-effect-b,
            pre-dec-a,
            pre-dec-b))

```

EVENT: Enable cv\$value; name this event ‘cv\$value-off’.

DEFINITION:

```

next-cntl-state (reset-,
                 dtack-,
                 hold-,
                 rw-,
                 state,
                 i-reg,
                 flags,
                 pc-reg,
                 regs-address)
=  cv (next-state (decode-5 (v-if (reset-, state, v_v11111)),
                    store-resultp (store-cc (i-reg), flags),
                    set-some-flags (set-flags (i-reg)),
                    unary-op-code-p (op-code (i-reg)),
                    or* (a-immediate-p (i-reg), reg-direct-p (mode-a (i-reg))),
                    reg-direct-p (mode-b (i-reg)),
                    and* (not* (a-immediate-p (i-reg)),
                        or* (pre-dec-p (mode-a (i-reg)),
                            post-inc-p (mode-a (i-reg))))) ,
                    or* (pre-dec-p (mode-b (i-reg)), post-inc-p (mode-b (i-reg))),
                    regs-address = list (t, t, t, t),
                    dtack-,
                    hold-),
    rn-a (i-reg),
    rn-b (i-reg),
    op-code (i-reg),
    pc-reg,
    regs-address,
    set-flags (i-reg),
    store-resultp (store-cc (i-reg), flags),
    c-flag (flags),
    a-immediate-p (i-reg),
    rw-,
    or* (a-immediate-p (i-reg), reg-direct-p (mode-a (i-reg))),
    and* (not* (a-immediate-p (i-reg)),
        or* (pre-dec-p (mode-a (i-reg)), post-inc-p (mode-a (i-reg))))) ,
    or* (pre-dec-p (mode-b (i-reg)), post-inc-p (mode-b (i-reg))),
    pre-dec-p (mode-a (i-reg)),
    pre-dec-p (mode-b (i-reg)))

```

EVENT: Enable next-cntl-state; name this event ‘next-cntl-state-off’.

THEOREM: bvp-length-next-cntl-state

$$\begin{aligned}
& (\text{length} (\text{next-cntl-state} (\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address})) \\
& = 40) \\
\wedge \quad & \text{bvp} (\text{next-cntl-state} (\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}))
\end{aligned}$$

DEFINITION:

$$\begin{aligned}
& \text{f\$next-cntl-state} (\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}) \\
& = \text{f\$cv} (\text{f\$next-state} (\text{f\$decode-5} (\text{fv-or} (\text{make-list} (5, \text{f-not} (\text{reset-})), \text{state})), \\
& \quad \text{cadr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})), \\
& \quad \text{caddr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})), \\
& \quad \text{cadddddr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})), \\
& \quad \text{caddddr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})), \\
& \quad \text{caddddr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})), \\
& \quad \text{cadddddddddr} (\text{f\$control-let} (\text{i-reg}, \\
& \quad \quad \text{flags}, \\
& \quad \quad \text{regs-address})), \\
& \quad \text{cadddddddddr} (\text{f\$control-let} (\text{i-reg}, \\
& \quad \quad \text{flags}, \\
& \quad \quad \text{regs-address})), \\
& \quad \text{cadddddddddr} (\text{f\$control-let} (\text{i-reg}, \text{flags}, \text{regs-address})),
\end{aligned}$$

$dtack-$,
 $hold-$),
 $rn-a(i-reg)$,
 $rn-b(i-reg)$,
 $op-code(i-reg)$,
 $pc-reg$,
 $regs-address$,
 $set-flags(i-reg)$,
 $cadr(f\$control-let(i-reg, flags, regs-address))$,
 $caddddddd(f\$control-let(i-reg, flags, regs-address))$,
 $car(f\$control-let(i-reg, flags, regs-address))$,
 $rw-$,
 $cadddr(f\$control-let(i-reg, flags, regs-address))$,
 $caddddddd(f\$control-let(i-reg, flags, regs-address))$,
 $caddddddd(f\$control-let(i-reg, flags, regs-address))$,
 $caddddddd(f\$control-let(i-reg, flags, regs-address))$,
 $caddddddd(f\$control-let(i-reg, flags, regs-address))$

EVENT: Enable $f\$next-cntl-state$; name this event ' $f\$next-cntl-state-off$ '.

THEOREM: $properp-length-f\$next-cntl-state$

$(length(f\$next-cntl-state(reset-$
 $dtack-$
 $hold-$
 $rw-$
 $state,$
 $i-reg,$
 $flags,$
 $pc-reg,$
 $regs-address)))$
 $= 40)$
 $\wedge properp(f\$next-cntl-state(reset-$
 $dtack-$
 $hold-$
 $rw-$
 $state,$
 $i-reg,$
 $flags,$
 $pc-reg,$
 $regs-address)))$

THEOREM: $v-or-crock-for-f\$next-cntl-state$

$((length(state) = 5) \wedge bvp(state))$
 $\rightarrow ((v-or(list(f, f, f, f, f), state) = state)$

$$\wedge \quad (\text{v-or}(\text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}), \text{state}) = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})))$$

EVENT: Enable v-or-crock-for-f\$next-cntl-state; name this event ‘v-or-crock-for-f\$next-cntl-state-off’.

THEOREM: f\$next-cntl-state=next-cntl-state

$$\begin{aligned}
& (\text{boolp}(\text{reset-}) \\
& \wedge \quad \text{boolp}(\text{dtack-}) \\
& \wedge \quad \text{boolp}(\text{hold-}) \\
& \wedge \quad \text{boolp}(\text{rw-}) \\
& \wedge \quad \text{bvp}(\text{state}) \\
& \wedge \quad (\text{length}(\text{state}) = 5) \\
& \wedge \quad \text{bvp}(\text{i-reg}) \\
& \wedge \quad (\text{length}(\text{i-reg}) = 32) \\
& \wedge \quad \text{bvp}(\text{flags}) \\
& \wedge \quad (\text{length}(\text{flags}) = 4) \\
& \wedge \quad \text{bvp}(\text{pc-reg}) \\
& \wedge \quad (\text{length}(\text{pc-reg}) = 4) \\
& \wedge \quad \text{bvp}(\text{regs-address}) \\
& \wedge \quad (\text{length}(\text{regs-address}) = 4)) \\
\rightarrow & \quad (\text{f\$next-cntl-state}(\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address})) \\
= & \quad \text{next-cntl-state}(\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}))
\end{aligned}$$

DEFINITION:

NEXT-CNTL-STATE*

$$\begin{aligned}
= & \quad \text{cons}(\text{'next-cntl-state}, \\
& \quad \text{cons}(\text{append6}(\text{'(reset- dtack- hold- rw-)}, \\
& \quad \quad \text{indices}(\text{'state}, 0, 5),
\end{aligned}$$

```

indices('i-reg, 0, 32),
indices('flags, 0, 4),
indices('pc-reg, 0, 4),
indices('regs-address, 0, 4)),
cons(indices('next-cntl-state, 0, 40),
cons(list(list('control-signals,
              '(a-immediate-p store
                set-some-flags direct-a
                direct-b unary pre-dec-a
                pre-dec-b c all-t-regs-address
                side-effect-a side-effect-b),
              'control-let,
              append(indices('i-reg, 0, 32),
                        append(indices('flags, 0, 4),
                                indices('regs-address,
                                      0,
                                      4))))),
      '(not-reset
        (reset)
        b-not
        (reset-)),
cons('reset5x,
cons(indices('reset5x, 0, 5),
      '(fanout-5 (reset))))),
list('xstate,
indices('xstate, 0, 5),
index('v-or, 5),
append(indices('reset5x, 0, 5),
        indices('state, 0, 5))),
list('dstate,
indices('decoded-state, 0, 32),
'decode-5,
indices('xstate, 0, 5)),
list('nxstate,
indices('next-state, 0, 32),
'next-state,
append(indices('decoded-state,
              0,
              32),
        '(store set-some-flags
          unary direct-a direct-b
          side-effect-a
          side-effect-b
          all-t-regs-address

```



```

                                dtack- hold-))),
list ('cvector,
      indices ('next-cntl-state, 0, 40),
      'cv,
      append8 (indices ('next-state, 0, 32),
                rn-a (indices ('i-reg, 0, 32)),
                rn-b (indices ('i-reg, 0, 32)),
                op-code (indices ('i-reg, 0, 32)),
                indices ('pc-reg, 0, 4),
                indices ('regs-address, 0, 4),
                set-flags (indices ('i-reg,
                                    0,
                                    32)),
                '(store c a-immediate-p
                      rw- direct-a
                      side-effect-a
                      side-effect-b pre-dec-a
                      pre-dec-b))))),
' (nil))))))

```

DEFINITION:

```

next-cntl-state& (netlist)
= ((lookup-module ('next-cntl-state, netlist) = NEXT-CNTL-STATE*)
   ^ control-let& (delete-module ('next-cntl-state, netlist))
   ^ b-not& (delete-module ('next-cntl-state, netlist))
   ^ fanout-5& (delete-module ('next-cntl-state, netlist))
   ^ v-or& (delete-module ('next-cntl-state, netlist), 5)
   ^ decode-5& (delete-module ('next-cntl-state, netlist))
   ^ next-state& (delete-module ('next-cntl-state, netlist))
   ^ cv& (delete-module ('next-cntl-state, netlist)))

```

EVENT: Enable next-cntl-state&; name this event 'next-cntl-state&-off'.

DEFINITION:

```

NEXT-CNTL-STATE$NETLIST
= cons (NEXT-CNTL-STATE*,
        (((CONTROL-LET$NETLIST ∪ B-NOT$NETLIST)
          ∪ (FANOUT-5$NETLIST ∪ v-or$netlist (5)))
         ∪ (DECODE-5$NETLIST ∪ NEXT-STATE$NETLIST))
        ∪ CV$NETLIST)

```

THEOREM: check-next-cntl-state\$netlist
next-cntl-state& (NEXT-CNTL-STATE\$NETLIST)

THEOREM: next-cntl-state\$value

$$\begin{aligned}
& (\text{next-cntl-state} \& (\text{netlist}) \\
& \wedge \text{properp}(\text{state}) \\
& \wedge (\text{length}(\text{state}) = 5) \\
& \wedge \text{properp}(i\text{-reg}) \\
& \wedge (\text{length}(i\text{-reg}) = 32) \\
& \wedge \text{properp}(\text{flags}) \\
& \wedge (\text{length}(\text{flags}) = 4) \\
& \wedge \text{properp}(pc\text{-reg}) \\
& \wedge (\text{length}(pc\text{-reg}) = 4) \\
& \wedge \text{properp}(\text{regs-address}) \\
& \wedge (\text{length}(\text{regs-address}) = 4)) \\
\rightarrow & (\text{dual-eval}(0, \\
& \quad \text{'next-cntl-state',} \\
& \quad \text{cons}(\text{reset-}, \\
& \quad \quad \text{cons}(\text{dtack-}, \\
& \quad \quad \quad \text{cons}(\text{hold-}, \\
& \quad \quad \quad \quad \text{cons}(rw\text{-}, \\
& \quad \quad \quad \quad \quad \text{append}(\text{state}, \\
& \quad \quad \quad \quad \quad \quad \text{append}(i\text{-reg}, \\
& \quad \quad \quad \quad \quad \quad \quad \text{append}(\text{flags}, \\
& \quad \quad \quad \quad \quad \quad \quad \quad \text{append}(pc\text{-reg}, \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{regs-address})))))), \\
& \quad \text{dual-eval-state}, \\
& \quad \text{netlist}) \\
= & \text{f\$next-cntl-state}(\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-}, \\
& \quad \text{state}, \\
& \quad \text{i-reg}, \\
& \quad \text{flags}, \\
& \quad \text{pc-reg}, \\
& \quad \text{regs-address}))
\end{aligned}$$

EVENT: Enable next-cntl-state\$value; name this event 'next-cntl-state\$value-off'.

THEOREM: next-cntl-state\$fetch0

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(\text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{rw-},
\end{aligned}$$

$$\begin{aligned}
& \text{V\_FETCH0}, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_fetch1}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))
\end{aligned}$$

$$\begin{aligned}
& \text{THEOREM: next-cntl-state}\$fetch1 \\
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset\text{-}, \\
& dtack\text{-}, \\
& hold\text{-}, \\
& rw\text{-}, \\
& \text{V\_FETCH1}, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg}, \\
& \text{regs-address}) \\
= & \text{if}^*(hold\text{-}, \\
& \text{cv\_fetch2}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_hold0}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})))
\end{aligned}$$

$$\begin{aligned}
& \text{THEOREM: next-cntl-state}\$fetch2 \\
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset\text{-}, \\
& dtack\text{-}, \\
& hold\text{-}, \\
& rw\text{-}, \\
& \text{V\_FETCH2}, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_fetch3}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))
\end{aligned}$$

$$\begin{aligned}
& \text{THEOREM: next-cntl-state}\$fetch3 \\
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw\text{-}, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset\text{-}, \\
& dtack\text{-}, \\
& hold\text{-}, \\
& rw\text{-}, \\
& \text{V\_FETCH3}, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg},
\end{aligned}$$

$$\begin{aligned}
& \text{regs-address}) \\
= & \text{if}^* (\text{dtack-}, \\
& \text{cv\_fetch3} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_decode} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$decode

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& rw-, \\
& V\_DECODE, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg}, \\
& \text{regs-address}) \\
= & \text{if}^* (\text{or}^* (\text{store-resultp} (\text{store-cc} (i\text{-reg}), \text{flags}), \\
& \text{set-some-flags} (\text{set-flags} (i\text{-reg}))), \\
& \text{if}^* (\text{or}^* (\text{a-immediate-p} (i\text{-reg}), \text{reg-direct-p} (\text{mode-a} (i\text{-reg}))), \\
& \text{if}^* (\text{or}^* (\text{reg-direct-p} (\text{mode-b} (i\text{-reg})), \\
& \text{unary-op-code-p} (\text{op-code} (i\text{-reg}))), \\
& \text{cv\_rega} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_readb0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_reada0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{if}^* (\text{and}^* (\text{not}^* (\text{a-immediate-p} (i\text{-reg})), \\
& \text{or}^* (\text{pre-dec-p} (\text{mode-a} (i\text{-reg})), \\
& \text{post-inc-p} (\text{mode-a} (i\text{-reg})))), \\
& \text{cv\_sefa0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{if}^* (\text{or}^* (\text{pre-dec-p} (\text{mode-b} (i\text{-reg})), \\
& \text{post-inc-p} (\text{mode-b} (i\text{-reg}))), \\
& \text{cv\_sefb0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_fetch0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))))
\end{aligned}$$

THEOREM: next-cntl-state\$rega

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& rw-, \\
& V\_REGA, \\
& i\text{-reg}, \\
& \text{flags}, \\
& pc\text{-reg}, \\
& \text{regs-address})
\end{aligned}$$

$$\begin{aligned}
= & \text{if}^* (\text{reg-direct-p} (\text{mode-b} (i\text{-reg})), \\
& \text{if}^* (\text{unary-op-code-p} (\text{op-code} (i\text{-reg})), \\
& \quad \text{cv\_update} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \quad \text{cv\_regb} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})), \\
& \text{if}^* (\text{store-resultp} (\text{store-cc} (i\text{-reg}), \text{flags}), \\
& \quad \text{cv\_write0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \quad \text{cv\_update} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))))
\end{aligned}$$

THEOREM: next-cntl-state\$regb

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \quad dtack-, \\
& \quad hold-, \\
& \quad rw-, \\
& \quad V\_REGB, \\
& \quad i\text{-reg}, \\
& \quad \text{flags}, \\
& \quad pc\text{-reg}, \\
& \quad \text{regs-address}) \\
= & \text{cv\_update} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$update

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \quad dtack-, \\
& \quad hold-, \\
& \quad rw-, \\
& \quad V\_UPDATE, \\
& \quad i\text{-reg}, \\
& \quad \text{flags}, \\
& \quad pc\text{-reg}, \\
& \quad \text{regs-address}) \\
= & \text{if}^* (\text{and}^* (\text{or}^* (\text{pre-dec-p} (\text{mode-b} (i\text{-reg})), \\
& \quad \text{post-inc-p} (\text{mode-b} (i\text{-reg}))), \\
& \quad \text{unary-op-code-p} (\text{op-code} (i\text{-reg}))), \\
& \text{cv\_sefb1} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\
& \text{cv\_fetch0} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})))
\end{aligned}$$

THEOREM: next-cntl-state\$reada0

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps} (rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \quad dtack-, \\
& \quad hold-, \\
& \quad rw-, \\
& \quad V\_READA0,
\end{aligned}$$

$$\begin{aligned}
& i\text{-reg}, \\
& flags, \\
& pc\text{-reg}, \\
& regs\text{-address}) \\
= & \text{cv\_reada1}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})
\end{aligned}$$

THEOREM: next-cntl-state\$reada1

$$\begin{aligned}
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset-, \\
& dtack-, \\
& hold-, \\
& rw-, \\
& V\_READA1, \\
& i\text{-reg}, \\
& flags, \\
& pc\text{-reg}, \\
& regs\text{-address}) \\
= & \text{cv\_reada2}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})
\end{aligned}$$

THEOREM: next-cntl-state\$reada2

$$\begin{aligned}
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset-, \\
& dtack-, \\
& hold-, \\
& rw-, \\
& V\_READA2, \\
& i\text{-reg}, \\
& flags, \\
& pc\text{-reg}, \\
& regs\text{-address}) \\
= & \text{cv\_reada3}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})
\end{aligned}$$

THEOREM: next-cntl-state\$reada3

$$\begin{aligned}
& ((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg})) \\
\rightarrow & (\text{next-cntl-state}(reset-, \\
& dtack-, \\
& hold-, \\
& rw-, \\
& V\_READA3, \\
& i\text{-reg}, \\
& flags, \\
& pc\text{-reg}, \\
& regs\text{-address}) \\
= & \text{if}^*(dtack-, \\
& \text{cv\_reada3}(rw-, regs\text{-address}, i\text{-reg}, flags, pc\text{-reg}),
\end{aligned}$$

$\text{if}^*(\text{reg-direct-p}(\text{mode-b}(i\text{-reg})),$
 $\quad \text{cv\_update}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}),$
 $\text{if}^*(\text{unary-op-code-p}(\text{op-code}(i\text{-reg})),$
 $\quad \text{if}^*(\text{store-resultp}(\text{store-cc}(i\text{-reg}), \text{flags}),$
 $\quad \quad \text{cv\_write0}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}),$
 $\quad \quad \text{cv\_update}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})),$
 $\quad \text{cv\_readb0}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))))$

THEOREM: next-cntl-state\$readb0

$((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow (\text{next-cntl-state}(\text{reset-},$
 $\quad dtack-,$
 $\quad hold-,$
 $\quad rw-,$
 $\quad V\_READB0,$
 $\quad i\text{-reg},$
 $\quad \text{flags},$
 $\quad pc\text{-reg},$
 $\quad \text{regs-address})$
 $= \text{cv\_readb1}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$

THEOREM: next-cntl-state\$readb1

$((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow (\text{next-cntl-state}(\text{reset-},$
 $\quad dtack-,$
 $\quad hold-,$
 $\quad rw-,$
 $\quad V\_READB1,$
 $\quad i\text{-reg},$
 $\quad \text{flags},$
 $\quad pc\text{-reg},$
 $\quad \text{regs-address})$
 $= \text{cv\_readb2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$

THEOREM: next-cntl-state\$readb2

$((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow (\text{next-cntl-state}(\text{reset-},$
 $\quad dtack-,$
 $\quad hold-,$
 $\quad rw-,$
 $\quad V\_READB2,$
 $\quad i\text{-reg},$
 $\quad \text{flags},$
 $\quad pc\text{-reg},$
 $\quad \text{regs-address})$

$$= \text{cv\_readb3}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$$

THEOREM: next-cntl-state\$readb3

$$\begin{aligned} & ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\ \rightarrow & (\text{next-cntl-state}(\text{reset-}, \\ & \quad dtack-, \\ & \quad hold-, \\ & \quad rw-, \\ & \quad V\_READB3, \\ & \quad i\text{-reg}, \\ & \quad \text{flags}, \\ & \quad pc\text{-reg}, \\ & \quad \text{regs-address})) \\ = & \text{if}^*(dtack-, \\ & \quad \text{cv\_readb3}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\ & \quad \text{if}^*(\text{store-resultp}(\text{store-cc}(i\text{-reg}), \text{flags}), \\ & \quad \quad \text{cv\_write0}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}), \\ & \quad \quad \text{cv\_update}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})))) \end{aligned}$$

THEOREM: next-cntl-state\$write0

$$\begin{aligned} & ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\ \rightarrow & (\text{next-cntl-state}(\text{reset-}, \\ & \quad dtack-, \\ & \quad hold-, \\ & \quad rw-, \\ & \quad V\_WRITE0, \\ & \quad i\text{-reg}, \\ & \quad \text{flags}, \\ & \quad pc\text{-reg}, \\ & \quad \text{regs-address})) \\ = & \text{cv\_write1}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \end{aligned}$$

THEOREM: next-cntl-state\$write1

$$\begin{aligned} & ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \\ \rightarrow & (\text{next-cntl-state}(\text{reset-}, \\ & \quad dtack-, \\ & \quad hold-, \\ & \quad rw-, \\ & \quad V\_WRITE1, \\ & \quad i\text{-reg}, \\ & \quad \text{flags}, \\ & \quad pc\text{-reg}, \\ & \quad \text{regs-address})) \\ = & \text{cv\_write2}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg})) \end{aligned}$$

THEOREM: next-cntl-state\$write2
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_WRITE2 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ cv_write3($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$))

THEOREM: next-cntl-state\$write3
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_WRITE3 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ if* ($dtack-$,
cv_write3($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$),
cv_fetch0($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$)))

THEOREM: next-cntl-state\$sefa0
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_SEFA0 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ cv_sefa1($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$))

THEOREM: next-cntl-state\$sefa1
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,

$$\begin{aligned}
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_SEFA1}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{if}^* (\text{or}^* (\text{pre-dec-p} (\text{mode-b} (\text{i-reg})), \text{post-inc-p} (\text{mode-b} (\text{i-reg}))), \\
& \text{cv\_sefb0} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}), \\
& \text{cv\_fetch0} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$sefb0

$$\begin{aligned}
& ((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_SEFB0}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_sefb1} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$sefb1

$$\begin{aligned}
& ((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_SEFB1}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_fetch0} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$hold0

$$\begin{aligned}
& ((\text{reset} = \mathbf{t}) \wedge \text{cv-hyps} (\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state} (\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_HOLD0}, \\
& \text{i-reg},
\end{aligned}$$

$$\begin{aligned}
& \text{flags,} \\
& \text{pc-reg,} \\
& \text{regs-address}) \\
= & \text{if}^*(\text{hold-}, \\
& \text{cv\_hold1}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}), \\
& \text{cv\_hold0}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$hold1

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state}(\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_HOLD1}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_fetch0}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$V11010

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state}(\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_V11010}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_reset0}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$V11011

$$\begin{aligned}
& ((\text{reset-} = \mathbf{t}) \wedge \text{cv-hyps}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg})) \\
\rightarrow & (\text{next-cntl-state}(\text{reset-}, \\
& \text{dtack-}, \\
& \text{hold-}, \\
& \text{rw-}, \\
& \text{V\_V11011}, \\
& \text{i-reg}, \\
& \text{flags}, \\
& \text{pc-reg}, \\
& \text{regs-address}) \\
= & \text{cv\_reset0}(\text{rw-}, \text{regs-address}, \text{i-reg}, \text{flags}, \text{pc-reg}))
\end{aligned}$$

THEOREM: next-cntl-state\$reset0
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_RESET0 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ cv_reset1($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$))

THEOREM: next-cntl-state\$reset1
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_RESET1 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ cv_reset2($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$))

THEOREM: next-cntl-state\$reset2
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,
 $hold-$,
 $rw-$,
 V_RESET2 ,
 $i\text{-reg}$,
 $flags$,
 $pc\text{-reg}$,
 $regs\text{-address}$)
 $=$ if* ($regs\text{-address} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})$,
cv_fetch0($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$),
cv_reset2($rw-$, $regs\text{-address}$, $i\text{-reg}$, $flags$, $pc\text{-reg}$)))

THEOREM: next-cntl-state\$V11111
 $((reset = \mathbf{t}) \wedge \text{cv-hyps}(rw-, \text{regs-address}, i\text{-reg}, \text{flags}, pc\text{-reg}))$
 $\rightarrow$ (next-cntl-state($reset-$,
 $dtack-$,

```

        hold-,
        rw-,
        V_V11111,
        i-reg,
        flags,
        pc-reg,
        regs-address)
=   cv_reset0(rw-, regs-address, i-reg, flags, pc-reg))

```

DEFINITION:

```

read-regs(address, regs)
=   if cadr(regs) ∧ v-iff(address, caddr(regs)) then cadr(regs)
    else read-mem(address, car(regs)) endif

```

EVENT: Enable read-regs; name this event ‘read-regs-off’.

DEFINITION:

```

write-regs(we, address, regs, data)
=   list(if cadr(regs) then write-mem(caddr(regs), car(regs), cadr(regs))
        else car(regs) endif,
        we,
        if we then data
        else cadr(regs) endif,
        if we then address
        else caddr(regs) endif)

```

EVENT: Enable write-regs; name this event ‘write-regs-off’.

THEOREM: write-regs-f-write-regs-f

```

write-regs(f, addr1, write-regs(f, addr2, regs, data2), data1)
=   write-regs(f, addr1, regs, data1)

```

THEOREM: write-regs-f-bv-crock

```

write-regs(f, addr, regs, bv(x)) = write-regs(f, 0, regs, 0)

```

THEOREM: properp-length-write-regs

```

properp(write-regs(we, address, regs, data))
∧   (length(write-regs(we, address, regs, data)) = 4)

```

THEOREM: write-regs-if

```

write-regs(if c then a
           else b endif,
           address,
           regs,

```

$$= \text{if } c \text{ then write-regs}(a, \text{address}, \text{regs}, \text{data}) \\ \text{else write-regs}(b, \text{address}, \text{regs}, \text{data}) \text{ endif}$$

THEOREM: write-regs-ok

$$(\text{memory-okp}(n, m, \text{car}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))) \\ = \text{if } \text{cadr}(\text{regs}) \\ \text{then memory-okp}(n, m, \text{write-mem}(\text{caddr}(\text{regs}), \text{car}(\text{regs}), \text{caddr}(\text{regs}))) \\ \text{else memory-okp}(n, m, \text{car}(\text{regs})) \text{ endif} \\ \wedge (\text{boolp}(\text{cadr}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data}))) = \text{boolp}(we)) \\ \wedge (\text{bvp}(\text{caddr}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))) \\ = \text{if } we \text{ then bvp}(\text{data}) \\ \text{else bvp}(\text{caddr}(\text{regs})) \text{ endif} \\ \wedge (\text{length}(\text{caddr}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))) \\ = \text{if } we \text{ then length}(\text{data}) \\ \text{else length}(\text{caddr}(\text{regs})) \text{ endif} \\ \wedge (\text{bvp}(\text{caddr}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))) \\ = \text{if } we \text{ then bvp}(\text{address}) \\ \text{else bvp}(\text{caddr}(\text{regs})) \text{ endif} \\ \wedge (\text{length}(\text{caddr}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))) \\ = \text{if } we \text{ then length}(\text{address}) \\ \text{else length}(\text{caddr}(\text{regs})) \text{ endif}$$

THEOREM: read-regs-write-regs-f

$$(\text{all-ramp-mem}(\text{length}(\text{addr}), \text{car}(\text{regs}))) \\ \wedge (\text{length}(\text{addr}) = \text{length}(\text{caddr}(\text{regs}))) \\ \wedge \text{bvp}(\text{addr}) \\ \rightarrow (\text{read-regs}(\text{addr}, \text{write-regs}(\mathbf{f}, \text{address}, \text{regs}, \text{value})) \\ = \text{read-regs}(\text{addr}, \text{regs}))$$

THEOREM: all-ramp-mem-after-write-regs

$$(\text{all-ramp-mem}(n, \text{car}(\text{regs})) \wedge (\text{length}(\text{caddr}(\text{regs})) = n)) \\ \rightarrow \text{all-ramp-mem}(n, \text{car}(\text{write-regs}(we, \text{address}, \text{regs}, \text{data})))$$

THEOREM: read-regs=read-mem

$$(\neg \text{cadr}(\text{regs})) \\ \rightarrow (\text{read-regs}(v\text{-addr}, \text{regs}) = \text{read-mem}(v\text{-addr}, \text{car}(\text{regs})))$$

THEOREM: read-regs=read-mem-write-mem

$$(\text{all-ramp-mem}(\text{length}(v\text{-addr1}), \text{car}(\text{regs}))) \\ \wedge (\text{length}(v\text{-addr1}) = \text{length}(\text{caddr}(\text{regs}))) \\ \wedge \text{cadr}(\text{regs}) \\ \rightarrow (\text{read-regs}(v\text{-addr1}, \text{regs}) \\ = \text{read-mem}(v\text{-addr1}, \\ \text{write-mem}(\text{caddr}(\text{regs}), \text{car}(\text{regs}), \text{caddr}(\text{regs}))))$$

THEOREM: bvp-length-read-regs-32
 (memory-okp (length (*v-addr*), 32, car (*regs*))
 $\wedge$ bvp (caddr (*regs*))
 $\wedge$ (length (caddr (*regs*)) = 32))
 $\rightarrow$ (bvp (read-regs (*v-addr*, *regs*))
 $\wedge$ (length (read-regs (*v-addr*, *regs*)) = 32))

DEFINITION:

REGFILE*

```
= cons ('regfile,
      cons (cons ('clk,
                  cons ('te,
                        cons ('ti,
                              cons ('we,
                                    cons ('disable-regfile-,
                                          cons ('test-regfile-,
                                                append (indices ('address,
                                                                0,
                                                                4),
                                                                indices ('data, 0, 32)))))),
            cons (cons ('to, indices ('out, 0, 32)),
                  cons (list ('(we-latch
                              (we-dp-ram we-dp-ram-)
                              fdls
                              (we clk ti te)),
                            list ('address-latch,
                                  indices ('address-dp-ram, 0, 4),
                                  index ('we-reg, 4),
                                  cons ('clk,
                                        cons ('we,
                                              cons ('te,
                                                    cons ('we-dp-ram,
                                                          indices ('address,
                                                                0,
                                                                4))))),
                            list ('data-latch,
                                  indices ('data-dp-ram, 0, 32),
                                  index ('we-reg, 32),
                                  cons ('clk,
                                        cons ('we,
                                              cons ('te,
                                                    cons (index ('address-dp-ram,
                                                                3),
                                                                indices ('data,
```

```

                                0,
                                32)))))),
'(reg-en-circuit
  (we-)
  ram-enable-circuit
  (clk test-regfile- disable-regfile-
   we-dp-ram)),
list('ram,
      indices('ramout, 0, 32),
      'dp-ram-16x32,
      append(indices('address, 0, 4),
              append(indices('address-dp-ram,
                              0,
                              4),
                    cons('we-,
                        indices('data-dp-ram,
                              0,
                              32)))))),
list('compare,
      '(read-equal-write),
      index('v-equal, 4),
      append(indices('address, 0, 4),
              indices('address-dp-ram,
                      0,
                      4))),
'(mux-control
  (s)
  b-and3
  (we-dp-ram read-equal-write
   test-regfile-)),
list('mux,
      indices('out, 0, 32),
      index('tv-if,
            tree-number(make-tree(32))),
      cons('s,
            append(indices('data-dp-ram,
                            0,
                            32),
                  indices('ramout, 0, 32))))),
list('scanout,
      '(to),
      'id,
      list(index('data-dp-ram, 31))),
'((ram we-latch data-latch address-latch))))))

```



```

THEOREM: regfile*$destructure
(car (REGFILE*) = 'regfile)
^  (cadr (REGFILE*)
    =  cons('clk,
            cons('te,
                  cons('ti,
                        cons('we,
                              cons('disable-regfile-,
                                    cons('test-regfile-,
                                          append(indices('address, 0, 4),
                                                    indices('data, 0, 32))))))))))
^  (caddr (REGFILE*) = cons('to, indices('out, 0, 32)))
^  (caddr (REGFILE*)
    =  list('we-latch
            (we-dp-ram we-dp-ram-)
            fdls
            (we clk ti te)),
      list('address-latch,
            indices('address-dp-ram, 0, 4),
            index('we-reg, 4),
            cons('clk,
                  cons('we,
                        cons('te,
                              cons('we-dp-ram,
                                    indices('address, 0, 4)))))),
      list('data-latch,
            indices('data-dp-ram, 0, 32),
            index('we-reg, 32),
            cons('clk,
                  cons('we,
                        cons('te,
                              cons(index('address-dp-ram, 3),
                                    indices('data, 0, 32)))))),
      '(reg-en-circuit
        (we-)
        ram-enable-circuit
        (clk test-regfile- disable-regfile-
         we-dp-ram)),
      list('ram,
            indices('ramout, 0, 32),
            'dp-ram-16x32,
            append(indices('address, 0, 4),
                    append(indices('address-dp-ram, 0, 4),
                            cons('we-,

```

```

                                indices('data-dp-ram, 0, 32))))),
list('compare,
      '(read-equal-write),
      index('v-equal, 4),
      append(indices('address, 0, 4),
              indices('address-dp-ram, 0, 4))),
'(mux-control
  (s)
  b-and3
  (we-dp-ram read-equal-write test-regfile-)),
list('mux,
      indices('out, 0, 32),
      index('tv-if, tree-number(make-tree(32))),
      cons('s,
            append(indices('data-dp-ram, 0, 32),
                    indices('ramout, 0, 32)))))
list('scanout,
      '(to),
      'id,
      list(index('data-dp-ram, 31))))
^ (caddddr(REGFILE*)
   = '(ram we-latch data-latch address-latch))

```

EVENT: Enable regfile*; name this event 'regfile*-off'.

EVENT: Enable regfile*\$destructure; name this event 'regfile*\$destructure-off'.

EVENT: Enable *1*regfile*; name this event 'g*1*regfile*-off'.

DEFINITION:

```

regfile& (netlist)
= ((lookup-module('regfile, netlist) = REGFILE*)
   ^ fdls&(delete-module('regfile, netlist))
   ^ we-reg&(delete-module('regfile, netlist), 4)
   ^ we-reg&(delete-module('regfile, netlist), 32)
   ^ ram-enable-circuit&(delete-module('regfile, netlist))
   ^ dp-ram-16x32&(delete-module('regfile, netlist))
   ^ v-equal&(delete-module('regfile, netlist), 4)
   ^ b-and3&(delete-module('regfile, netlist))
   ^ tv-if&(delete-module('regfile, netlist), make-tree(32))
   ^ id&(delete-module('regfile, netlist)))

```

EVENT: Enable regfile&; name this event 'regfile&-off'.

DEFINITION:

REGFILE\$NETLIST

$$= \text{cons}(\text{REGFILE}^*,$$

$$\quad (((\text{FD1S\$NETLIST} \cup \text{we-reg\$netlist}(4))$$

$$\quad \cup (\text{we-reg\$netlist}(32) \cup \text{RAM-ENABLE-CIRCUIT\$NETLIST}))$$

$$\quad \cup (\text{DP-RAM-16X32\$NETLIST}$$

$$\quad \quad \cup (\text{v-equal\$netlist}(4) \cup \text{B-AND3\$NETLIST})))$$

$$\quad \cup (\text{tv-if\$netlist}(\text{make-tree}(32)) \cup \text{ID\$NETLIST}))$$

THEOREM: check-regfile\$netlist
 regfile&(REGFILE\$NETLIST)

DEFINITION:

f\$read-regs(*address*, *regs*)

$$= \text{fv-if}(\text{f-and3}(\text{f-buf}(\text{cadr}(\text{regs})),$$

$$\quad \text{f\$v-equal}(\text{address}, \text{v-threefix}(\text{caddr}(\text{regs}))),$$

$$\quad \mathbf{t}),$$

$$\quad \text{v-threefix}(\text{caddr}(\text{regs})),$$

$$\quad \text{dual-port-ram-value}(32,$$

$$\quad \quad 4,$$

$$\quad \quad \text{append}(\text{address},$$

$$\quad \quad \quad \text{append}(\text{v-threefix}(\text{caddr}(\text{regs})),$$

$$\quad \quad \quad \text{cons}(\text{f-nand}(\mathbf{t},$$

$$\quad \quad \quad \quad \text{f-buf}(\text{cadr}(\text{regs}))),$$

$$\quad \quad \quad \quad \text{v-threefix}(\text{caddr}(\text{regs}))))),$$

$$\quad \text{car}(\text{regs}))$$

EVENT: Enable f\$read-regs; name this event 'f\$read-regs-off'.

THEOREM: properp-length-f\$read-regs

$$\text{properp}(\text{f\$read-regs}(\text{address}, \text{mem}))$$

$$\wedge ((\text{length}(\text{caddr}(\text{mem})) = 32)$$

$$\rightarrow (\text{length}(\text{f\$read-regs}(\text{address}, \text{mem})) = 32))$$

THEOREM: f\$read-regs=read-regs

$$(\text{memory-okp}(4, 32, \text{car}(\text{regs}))$$

$$\wedge \text{boolp}(\text{cadr}(\text{regs}))$$

$$\wedge \text{bvp}(\text{caddr}(\text{regs}))$$

$$\wedge (\text{length}(\text{caddr}(\text{regs})) = 32)$$

$$\wedge \text{bvp}(\text{caddr}(\text{regs}))$$

$$\wedge (\text{length}(\text{caddr}(\text{regs})) = 4)$$

$$\wedge \text{bvp}(\text{address})$$

$$\wedge (\text{length}(\text{address}) = 4))$$

$$\rightarrow (\text{f\$read-regs}(\text{address}, \text{regs}) = \text{read-regs}(\text{address}, \text{regs}))$$

THEOREM: regfile\$value
 (regfile& (*netlist*)
 $\wedge$ (*test-regfile-* = **t**)
 $\wedge$ (*disable-regfile-* = **t**)
 $\wedge$ memory-properp (4, 32, *mem*)
 $\wedge$ properp (*last-data*)
 $\wedge$ (length (*last-data*) = 32)
 $\wedge$ properp (*address*)
 $\wedge$ (length (*address*) = 4)
 $\wedge$ properp (*last-address*)
 $\wedge$ (length (*last-address*) = 4))
 $\rightarrow$ (cdr (dual-eval (0,
 '**regfile**,
 cons (*clk*,
 cons (*te*,
 cons (*ti*,
 cons (*we*,
 cons (*disable-regfile-*,
 cons (*test-regfile-*,
 append (*address*, *data*)))))),
 list (*mem*, *last-we*, *last-data*, *last-address*),
 netlist))
 = f\$read-regs (*address*, list (*mem*, *last-we*, *last-data*, *last-address*)))

EVENT: Enable regfile\$value; name this event 'regfile\$value-off'.

DEFINITION:
 f\$write-regs (*we*, *address*, *regs*, *data*)
 = list (dual-port-ram-state (32,
 4,
 append (*address*,
 append (v-threefix (caddr (*regs*)),
 cons (f-nand (**t**,
 f-buf (cadr (*regs*))),
 v-threefix (caddr (*regs*))))),
 car (*regs*)),
 threefix (*we*),
 fv-if (*we*, *data*, caddr (*regs*)),
 fv-if (*we*, *address*, caddr (*regs*)))

EVENT: Enable f\$write-regs; name this event 'f\$write-regs-off'.

THEOREM: properp-length-f\$write-regs

```

properp (f$write-regs (we, address, regs, data))
 $\wedge$  (length (f$write-regs (we, address, regs, data)) = 4)

```

THEOREM: f\$write-regs=write-regs

```

(boolp (we)
 $\wedge$  boolp (cadr (regs))
 $\wedge$  bvp (caddr (regs))
 $\wedge$  (length (caddr (regs)) = 32)
 $\wedge$  bvp (caddr (regs))
 $\wedge$  (length (caddr (regs)) = 4)
 $\wedge$  bvp (address)
 $\wedge$  (length (address) = 4)
 $\wedge$  bvp (data)
 $\wedge$  (length (data) = 32))
 $\rightarrow$  (f$write-regs (we, address, regs, data)
    = write-regs (we, address, regs, data))

```

THEOREM: regfile\$state

```

(regfile& (netlist)
 $\wedge$  ( $\neg$  te)
 $\wedge$  (test-regfile- = t)
 $\wedge$  (disable-regfile- = t)
 $\wedge$  properp (data)
 $\wedge$  (length (data) = 32)
 $\wedge$  properp (last-data)
 $\wedge$  (length (last-data) = 32)
 $\wedge$  properp (address)
 $\wedge$  (length (address) = 4)
 $\wedge$  properp (last-address)
 $\wedge$  (length (last-address) = 4))
 $\rightarrow$  (dual-eval (2,
    'regfile,
    cons (clk,
        cons (te,
            cons (ti,
                cons (we,
                    cons (disable-regfile-,
                        cons (test-regfile-,
                            append (address, data)))))),
                list (regfile, last-we, last-data, last-address),
                netlist)
    = f$write-regs (we,
        address,
        list (regfile, last-we, last-data, last-address),
        data))

```

EVENT: Enable regfile\$state; name this event 'regfile\$state-off'.

DEFINITION:

FLAGS*

```
= cons('flags,
      cons(cons('clk,
                cons('te,
                    cons('ti,
                        append(indices('set-flags, 0, 4),
                               indices('cvzbv, 0, 35))))),
      cons('z n v c),
      cons(list(list('z-latch,
                    'z zb),
                  'fd1slp,
                  cons(zb(indices('cvzbv, 0, 35)),
                        cons('clk,
                            cons(z-set(indices('set-flags,
                                                0,
                                                4)),
                                '(ti te))))),
      list('n-latch,
          'n nb),
          'fd1slp,
          cons(index('cvzbv, 34),
                cons('clk,
                    cons(n-set(indices('set-flags,
                                        0,
                                        4)),
                        '(z te))))),
      list('v-latch,
          'v vb),
          'fd1slp,
          cons(v(indices('cvzbv, 0, 35)),
                cons('clk,
                    cons(v-set(indices('set-flags,
                                        0,
                                        4)),
                        '(n te))))),
      list('c-latch,
          'c cb),
          'fd1slp,
          cons(c(indices('cvzbv, 0, 35)),
                cons('clk,
```

```

cons (c-set (indices ('set-flags,
                      0,
                      4)),
      '(v te))))),
'((z-latch n-latch v-latch c-latch))))))

```

THEOREM: flags*\$destructure

```

(car (FLAGS*) = 'flags)
^ (cadr (FLAGS*)
   = cons ('clk,
            cons ('te,
                  cons ('ti,
                        append (indices ('set-flags, 0, 4),
                                indices ('cvzbv, 0, 35))))))
^ (caddr (FLAGS*) = '(z n v c))
^ (caddr (FLAGS*)
   = list (list ('z-latch,
                  '(z zb),
                  'fd1slp,
                  cons (zb (indices ('cvzbv, 0, 35)),
                        cons ('clk,
                              cons (z-set (indices ('set-flags, 0, 4)),
                                      '(ti te))))),
            list ('n-latch,
                  '(n nb),
                  'fd1slp,
                  cons (index ('cvzbv, 34),
                        cons ('clk,
                              cons (n-set (indices ('set-flags, 0, 4)),
                                      '(z te))))),
            list ('v-latch,
                  '(v vb),
                  'fd1slp,
                  cons (v (indices ('cvzbv, 0, 35)),
                        cons ('clk,
                              cons (v-set (indices ('set-flags, 0, 4)),
                                      '(n te))))),
            list ('c-latch,
                  '(c cb),
                  'fd1slp,
                  cons (c (indices ('cvzbv, 0, 35)),
                        cons ('clk,
                              cons (c-set (indices ('set-flags, 0, 4)),
                                      '(v te))))))

```

$\wedge$ (caddddr (FLAGS*) = '(z-latch n-latch v-latch c-latch))

EVENT: Enable flags*; name this event 'flags*-off'.

EVENT: Enable flags*\$destructure; name this event 'flags*\$destructure-off'.

EVENT: Enable *1*flags*; name this event 'g*1*flags*-off'.

DEFINITION:

flags& (netlist)
 = ((lookup-module ('f**lags**, netlist) = FLAGS*)
 $\wedge$ fd1slp& (delete-module ('f**lags**, netlist)))

EVENT: Enable flags&; name this event 'flags&-off'.

DEFINITION: FLAGS\$NETLIST = cons (FLAGS*, FD1SLP\$NETLIST)

THEOREM: check-flags\$netlist
 flags& (FLAGS\$NETLIST)

THEOREM: flags\$value
 (flags& (netlist) $\wedge$ properp (flags) $\wedge$ (length (flags) = 4))
 $\rightarrow$ (dual-eval (0,
 'f**lags**,
 cons (clk, cons (te, cons (ti, append (set-flags, cvzbv)))),
 flags,
 netlist)
 = v-threefix (flags))

EVENT: Enable flags\$value; name this event 'flags\$value-off'.

DEFINITION:

f\$update-flags (flags, set-flags, cvzbv)
 = list (f-if (z-set (set-flags), zb (cvzbv), z-flag (flags)),
 f-if (n-set (set-flags), n (cvzbv), n-flag (flags)),
 f-if (v-set (set-flags), v (cvzbv), v-flag (flags)),
 f-if (c-set (set-flags), c (cvzbv), c-flag (flags)))

EVENT: Enable f\$update-flags; name this event 'f\$update-flags-off'.

THEOREM: properp-length-f\$update-flags
 properp (f\$update-flags (flags, set-flags, cvzbv))
 $\wedge$ (length (f\$update-flags (flags, set-flags, cvzbv)) = 4)

THEOREM: f\$update-flags=update-flags
 (bvp (*flags*)
 $\wedge$ (length (*flags*) = 4)
 $\wedge$ bvp (*set-flags*)
 $\wedge$ (length (*set-flags*) = 4)
 $\wedge$ bvp (*cvzbv*)
 $\wedge$ (length (*cvzbv*) = 35))
 $\rightarrow$ (f\$update-flags (*flags*, *set-flags*, *cvzbv*)
 = update-flags (*flags*, *set-flags*, *cvzbv*))

THEOREM: flags\$state-help
 (flags& (*netlist*)
 $\wedge$ properp (*flags*)
 $\wedge$ (length (*flags*) = 4)
 $\wedge$ (*set-flags* = list (*set-z*, *set-n*, *set-v*, *set-c*))
 $\wedge$ (*cvzbv* = cons (*c*, cons (*v*, cons (*z*, *bv*))))
 $\wedge$ properp (*cvzbv*)
 $\wedge$ (length (*cvzbv*) = 35)
 $\wedge$ ($\neg$ *te*)
 $\rightarrow$ (dual-eval (2,
 'f\$flags,
 cons (*clk*, cons (*te*, cons (*ti*, append (*set-flags*, *cvzbv*)))),
 flags,
 netlist)
 = f\$update-flags (*flags*, *set-flags*, *cvzbv*))

EVENT: Enable flags\$state-help; name this event 'flags\$state-help-off'.

THEOREM: flags\$state
 (flags& (*netlist*)
 $\wedge$ properp (*flags*)
 $\wedge$ (length (*flags*) = 4)
 $\wedge$ properp (*set-flags*)
 $\wedge$ (length (*set-flags*) = 4)
 $\wedge$ properp (*cvzbv*)
 $\wedge$ (length (*cvzbv*) = 35)
 $\wedge$ ($\neg$ *te*)
 $\rightarrow$ (dual-eval (2,
 'f\$flags,
 cons (*clk*, cons (*te*, cons (*ti*, append (*set-flags*, *cvzbv*)))),
 flags,
 netlist)
 = f\$update-flags (*flags*, *set-flags*, *cvzbv*))

EVENT: Enable flags\$state; name this event 'flags\$state-off'.

THEOREM: flags\$partial-state-help

```

(flags& (netlist)
  ^ properp (flags)
  ^ (length (flags) = 4)
  ^ (set-flags = make-list (4, t))
  ^ (cvzbv = cons (c, cons (v, cons (z, bv))))
  ^ bvp (cvzbv)
  ^ (length (cvzbv) = 35)
  ^ (¬ te))
→ (dual-eval (2,
               'flags,
               cons (clk, cons (te, cons (ti, append (set-flags, cvzbv)))),
               flags,
               netlist)
    = update-flags (flags, set-flags, cvzbv))

```

EVENT: Enable flags\$partial-state-help; name this event 'flags\$partial-state-help-off'.

THEOREM: flags\$partial-state

```

(flags& (netlist)
  ^ properp (flags)
  ^ (length (flags) = 4)
  ^ (set-flags = make-list (4, t))
  ^ bvp (cvzbv)
  ^ (length (cvzbv) = 35)
  ^ (¬ te))
→ (dual-eval (2,
               'flags,
               cons (clk, cons (te, cons (ti, append (set-flags, cvzbv)))),
               flags,
               netlist)
    = update-flags (flags, set-flags, cvzbv))

```

EVENT: Enable flags\$partial-state; name this event 'flags\$partial-state-off'.

DEFINITION:

EXTEND-IMMEDIATE*

```

= cons ('extend-immediate,
        cons (cons ('select-immediate,
                    append (indices ('immediate, 0, 9),
                                   indices ('reg-data, 0, 32))),
              cons (indices ('z, 0, 32),

```

```

cons (list (list ('buffer,
                  '(sign-bit),
                  'b-buf-pwr,
                  list (index ('immediate, 8))),
            list ('mux,
                  indices ('z, 0, 32),
                  index ('tv-if,
                          tree-number (make-tree (32))),
                  cons ('select-immediate,
                        append (append (indices ('immediate,
                                                  0,
                                                  9),
                                                  make-list (23,
                                                              'sign-bit)),
                                      indices ('reg-data, 0, 32))))),
      '(nil))))

```

THEOREM: extend-immediate*\$destructure

```

(car (EXTEND-IMMEDIATE*) = 'extend-immediate)
^  (cadr (EXTEND-IMMEDIATE*)
    = cons ('select-immediate,
            append (indices ('immediate, 0, 9),
                    indices ('reg-data, 0, 32))))
^  (caddr (EXTEND-IMMEDIATE*) = indices ('z, 0, 32))
^  (caddrdr (EXTEND-IMMEDIATE*)
    = list (list ('buffer,
                  '(sign-bit),
                  'b-buf-pwr,
                  list (index ('immediate, 8))),
            list ('mux,
                  indices ('z, 0, 32),
                  index ('tv-if, tree-number (make-tree (32))),
                  cons ('select-immediate,
                        append (append (indices ('immediate, 0, 9),
                                                  make-list (23, 'sign-bit)),
                                      indices ('reg-data, 0, 32))))))
^  (caddrdrdr (EXTEND-IMMEDIATE*) = nil)

```

EVENT: Enable extend-immediate*; name this event 'extend-immediate*-off'.

EVENT: Enable extend-immediate*\$destructure; name this event 'extend-immediate*\$destructure-off'.

EVENT: Enable *1*extend-immediate*; name this event 'g*1*extend-immediate*-

off’.

DEFINITION:

```
extend-immediate& (netlist)
= ((lookup-module ('extend-immediate, netlist)
   = EXTEND-IMMEDIATE*)
   ∧ b-buf-pwr& (delete-module ('extend-immediate, netlist))
   ∧ tv-if& (delete-module ('extend-immediate, netlist),
              make-tree (32)))
```

EVENT: Enable extend-immediate&; name this event ‘extend-immediate&-off’.

DEFINITION:

```
EXTEND-IMMEDIATE$NETLIST
= cons (EXTEND-IMMEDIATE*,
        B-BUF-PWR$NETLIST ∪ tv-if$netlist (make-tree (32)))
```

THEOREM: check-extend-immediate\$netlist
extend-immediate& (EXTEND-IMMEDIATE\$NETLIST)

DEFINITION:

```
f$extend-immediate (select-immediate, immediate, reg-data)
= fv-if (select-immediate,
        append (immediate,
                if boolp (nth (8, immediate))
                  then make-list (23, nth (8, immediate))
                  else make-list (23, x) endif),
        reg-data)
```

EVENT: Enable f\$extend-immediate; name this event ‘f\$extend-immediate-off’.

THEOREM: properp-length-f\$extend-immediate
properp (f\$extend-immediate (*select-immediate*, *immediate*, *reg-bus*))
∧ ((length (*immediate*) = 9)
→ (length (f\$extend-immediate (*select-immediate*, *immediate*, *reg-bus*))
= 32))

THEOREM: f\$extend-immediate=extend-immediate

```
(bvp (immediate)
 ∧ (length (immediate) = 9)
 ∧ bvp (reg-data)
 ∧ (length (reg-data) = 32)
 ∧ boolp (select-immediate))
→ (f$extend-immediate (select-immediate, immediate, reg-data)
   = if* (select-immediate, sign-extend (immediate, 32), reg-data))
```

THEOREM: extend-immediate\$value
 (extend-immediate& (*netlist*)
 $\wedge$ properp (*immediate*)
 $\wedge$ (length (*immediate*) = 9)
 $\wedge$ properp (*reg-data*)
 $\wedge$ (length (*reg-data*) = 32))
 $\rightarrow$ (dual-eval (0,
 'extend-immediate,
 cons (select-immediate, append (*immediate*, *reg-data*)),
 state,
 netlist)
 = f\$extend-immediate (select-immediate, *immediate*, *reg-data*))

EVENT: Enable extend-immediate\$value; name this event 'extend-immediate\$value-off'.

DEFINITION:
 ttl-input-pads-body (*m*, *n*, *pi*)
 = **if** *n* $\simeq$ 0 **then** **nil**
 else cons (list (index ('g, *m*),
 list (index ('out, *m*), index ('po, *m*)),
 'ttl-input,
 list (index ('in, *m*), *pi*)),
 cons (list (index ('b, *m*),
 list (index ('b-out, *m*)),
 'b-buf,
 list (index ('out, *m*))),
 ttl-input-pads-body (1 + *m*, *n* - 1, index ('po, *m*))) **endif**

EVENT: Enable ttl-input-pads-body; name this event 'ttl-input-pads-body-off'.

DEFINITION:
 ttl-input-pads-body\$induction (*m*, *n*, *pi*, *bindings*, *state-bindings*, *netlist*)
 = **if** *n* $\simeq$ 0 **then** **t**
 else ttl-input-pads-body\$induction (1 + *m*,
 n - 1,
 index ('po, *m*),
 dual-eval-body-bindings (2,
 ttl-input-pads-body (*m*,
 n,
 pi),
 bindings,
 state-bindings,

```

state-bindings,
netlist) endif

```

DEFINITION:

```
ttn-input-pads*(n)
= cons(index('ttn-input-pads', n),
      cons(cons('pi', indices('in', 0, n)),
            cons(cons(index('po', n - 1), indices('b-out', 0, n)),
                  cons(ttn-input-pads-body(0, n, 'pi'), 'nil')))))
```

THEOREM: `ttl-input-pads*`\$destructure

```

(car (ttl-input-pads* (n)) = index ('ttl-input-pads, n))
^  (cadr (ttl-input-pads* (n)) = cons ('pi, indices ('in, 0, n)))
^  (caddr (ttl-input-pads* (n))
    = cons (index ('po, n - 1), indices ('b-out, 0, n)))
^  (caddddr (ttl-input-pads* (n)) = ttl-input-pads-body (0, n, 'pi))
^  (cadddddr (ttl-input-pads* (n)) = nil)

```

EVENT: Enable ttl-input-pads*; name this event 'ttl-input-pads*-off'.

EVENT: Enable ttl-input-pads*\$destructure; name this event ‘ttl-input-pads*\$destructure-off’.

DEFINITION:

$$\begin{aligned} & \text{ttl-input-pads} \& (\text{netlist}, n) \\ = & ((\text{lookup-module}(\text{index}(' \text{ttl-input-pads}, n), \text{netlist}) \\ & = \text{ttl-input-pads}^*(n)) \\ & \wedge \text{ttl-input} \& (\text{delete-module}(\text{index}(' \text{ttl-input-pads}, n), \\ & \quad \text{netlist})) \\ & \wedge \text{b-buf} \& (\text{delete-module}(\text{index}(' \text{ttl-input-pads}, n), \text{netlist}))) \end{aligned}$$

EVENT: Enable ttl-input-pads&; name this event 'ttl-input-pads&-off'.

DEFINITION:

$$\begin{aligned} & \text{ttl-input-pads}\$netlist(n) \\ &= \text{cons}(\text{ttl-input-pads}^*(n), \text{TTL-INPUT}\$NETLIST \cup \text{B-BUF}\$NETLIST) \end{aligned}$$

THEOREM: $\text{ttl-input-pads-body} \leq \text{unbound-in-body}$

$$\begin{aligned} & (l < m) \\ & \rightarrow \text{unbound-in-body}(\text{index}('b\text{-out}, l), \text{ttl-input-pads-body}(m, n, pi)) \end{aligned}$$

EVENT: Enable `ttl-input-pads-body$unbound-in-body`; name this event ‘`ttl-input-pads-body$unbound-in-body-off`’.

THEOREM: $\text{ttl-input-pads-body}\$value$
 $(\text{ttl-input}\&(\text{netlist}) \wedge \text{b-buf}\&(\text{netlist}))$
 $\rightarrow (\text{collect-value}(\text{indices}('b-out, m, n),$
 $\quad \text{dual-eval}(1,$
 $\quad \quad \text{ttl-input-pads-body}(m, n, pi),$
 $\quad \quad \text{bindings},$
 $\quad \quad \text{state-bindings},$
 $\quad \quad \text{netlist}))$
 $= \text{v-threefix}(\text{collect-value}(\text{indices}('in, m, n), \text{bindings}))$

EVENT: Enable $\text{ttl-input-pads-body}\$value$; name this event ' $\text{ttl-input-pads-body}\$value-off$ '.

THEOREM: $\text{ttl-input-pads}\$value$
 $(\text{ttl-input-pads}\&(\text{netlist}, n) \wedge (\text{length}(\text{inputs}) = n) \wedge \text{properp}(\text{inputs}))$
 $\rightarrow (\text{cdr}(\text{dual-eval}(0,$
 $\quad \text{index}('ttl-input-pads, n),$
 $\quad \text{cons}(pi, \text{inputs}),$
 $\quad \text{state},$
 $\quad \text{netlist}))$
 $= \text{v-threefix}(\text{inputs}))$

EVENT: Enable $\text{ttl-input-pads}\$value$; name this event ' $\text{ttl-input-pads}\$value-off$ '.

DEFINITION:
 $\text{ttl-output-pads-body}(m, n)$
 $= \text{if } n \simeq 0 \text{ then nil}$
 $\quad \text{else cons}(\text{list}(\text{index}('b, m),$
 $\quad \quad \text{list}(\text{index}('b-in, m)),$
 $\quad \quad 'b-buf,$
 $\quad \quad \text{list}(\text{index}('in, m))),$
 $\quad \text{cons}(\text{list}(\text{index}('g, m),$
 $\quad \quad \text{list}(\text{index}('out, m)),$
 $\quad \quad 'ttl-output,$
 $\quad \quad \text{list}(\text{index}('b-in, m))),$
 $\quad \text{ttl-output-pads-body}(1 + m, n - 1))) \text{ endif}$

EVENT: Enable $\text{ttl-output-pads-body}$; name this event ' $\text{ttl-output-pads-body-off}$ '.

DEFINITION:
 $\text{ttl-output-pads-body}\$induction(m, n, \text{bindings}, \text{state-bindings}, \text{netlist})$
 $= \text{if } n \simeq 0 \text{ then t}$

```

else ttl-output-pads-body$induction (1 + m,
                                     n - 1,
                                     dual-eval-body-bindings (2,
                                                                ttl-output-pads-body (m,
                                                                                   n),
                                                                bindings,
                                                                state-bindings,
                                                                netlist),
                                     state-bindings,
                                     netlist) endif

```

DEFINITION:

```

ttl-output-pads* (n)
=  cons (index ('ttl-output-pads, n),
        cons (indices ('in, 0, n),
              cons (indices ('out, 0, n),
                    cons (ttl-output-pads-body (0, n), ' (nil))))))

```

THEOREM: ttl-output-pads*\$destructure

```

(car (ttl-output-pads* (n)) = index ('ttl-output-pads, n))
^  (cadr (ttl-output-pads* (n)) = indices ('in, 0, n))
^  (caddr (ttl-output-pads* (n)) = indices ('out, 0, n))
^  (cadddr (ttl-output-pads* (n)) = ttl-output-pads-body (0, n))
^  (caddddr (ttl-output-pads* (n)) = nil)

```

EVENT: Enable ttl-output-pads*; name this event 'ttl-output-pads*-off'.

EVENT: Enable ttl-output-pads*\$destructure; name this event 'ttl-output-pads*\$destructure-off'.

DEFINITION:

```

ttl-output-pads& (netlist, n)
=  ((lookup-module (index ('ttl-output-pads, n), netlist)
                   =  ttl-output-pads* (n))
   ^  ttl-output& (delete-module (index ('ttl-output-pads, n),
                                         netlist))
   ^  b-buf& (delete-module (index ('ttl-output-pads, n), netlist)))

```

EVENT: Enable ttl-output-pads&; name this event 'ttl-output-pads&-off'.

DEFINITION:

```

ttl-output-pads$netlist (n)
=  cons (ttl-output-pads* (n), TTL-OUTPUT$NETLIST ∪ B-BUF$NETLIST)

```


THEOREM: $\text{ttl-output-pads-body}\$unbound\text{-in-body}$
 $(l < m) \rightarrow \text{unbound-in-body}(\text{index}('out, l), \text{ttl-output-pads-body}(m, n))$

EVENT: Enable $\text{ttl-output-pads-body}\$unbound\text{-in-body}$; name this event ' $\text{ttl-output-pads-body}\$unbound\text{-in-body-off}$ '.

THEOREM: $\text{ttl-output-pads-body}\$value$
 $(\text{ttl-output-pads}\&(\text{netlist}) \wedge \text{b-buf}\&(\text{netlist}))$
 $\rightarrow (\text{collect-value}(\text{indices}('out, m, n),$
 $\quad \text{dual-eval}(1,$
 $\quad \quad \text{ttl-output-pads-body}(m, n),$
 $\quad \quad \text{bindings},$
 $\quad \quad \text{state-bindings},$
 $\quad \quad \text{netlist}))$
 $= \text{v-threefix}(\text{collect-value}(\text{indices}('in, m, n), \text{bindings}))$

EVENT: Enable $\text{ttl-output-pads-body}\$value$; name this event ' $\text{ttl-output-pads-body}\$value-off$ '.

THEOREM: $\text{ttl-output-pads}\$value$
 $(\text{ttl-output-pads}\&(\text{netlist}, n) \wedge (\text{length}(\text{inputs}) = n) \wedge \text{properp}(\text{inputs}))$
 $\rightarrow (\text{dual-eval}(0, \text{index}('ttl-output-pads, n), \text{inputs}, \text{state}, \text{netlist})$
 $\quad = \text{v-threefix}(\text{inputs}))$

DEFINITION:
 $\text{ttl-tri-output-pads-body}(m, n)$
 $= \text{if } n \simeq 0 \text{ then nil}$
 $\quad \text{else cons}(\text{list}(\text{index}('g, m),$
 $\quad \quad \text{list}(\text{index}('out, m)),$
 $\quad \quad 'ttl-tri-output,$
 $\quad \quad \text{cons}(\text{index}('in, m), 'enable-buf)),$
 $\quad \text{ttl-tri-output-pads-body}(1 + m, n - 1)) \text{ endif}$

EVENT: Enable $\text{ttl-tri-output-pads-body}$; name this event ' $\text{ttl-tri-output-pads-body-off}$ '.

DEFINITION:
 $\text{ttl-tri-output-pads-body}\$induction(m, n, \text{bindings}, \text{state-bindings}, \text{netlist})$
 $= \text{if } n \simeq 0 \text{ then t}$
 $\quad \text{else ttl-tri-output-pads-body}\$induction(1 + m,$
 $\quad \quad \quad n - 1,$
 $\quad \quad \quad \text{dual-eval-body-bindings}(1,$
 $\quad \quad \quad \text{ttl-tri-output-pads-body}(m,$

n),

bindings,
state-bindings,
netlist),

state-bindings,
netlist) **endif**

DEFINITION:

```
ttl-tri-output-pads*(n)
=  cons(index('ttl-tri-output-pads, n),
        cons(cons('enable, indices('in, 0, n)),
              cons(indices('out, 0, n),
                    cons(cons(cons('enable-buffer,
                                  cons('enable-buf),
                                      cons(if n < 8 then 'b-buf
                                             else 'b-buf-pwr endif,
                                             '((enable))))),
                          ttl-tri-output-pads-body(0, n)),
                    '(nil))))))
```

THEOREM: $\text{ttl-tri-output-pads}^*\$destructure$

```
(car(ttl-tri-output-pads*(n)) = index('ttl-tri-output-pads, n))
^ (cadr(ttl-tri-output-pads*(n)) = cons('enable, indices('in, 0, n)))
^ (caddr(ttl-tri-output-pads*(n)) = indices('out, 0, n))
^ (caddr(ttl-tri-output-pads*(n))
   =  cons(cons('enable-buffer,
                 cons('enable-buf),
                     cons(if n < 8 then 'b-buf
                           else 'b-buf-pwr endif,
                           '((enable))))),
         ttl-tri-output-pads-body(0, n)))
^ (caddr(ttl-tri-output-pads*(n)) = nil)
```

EVENT: Enable $\text{ttl-tri-output-pads}^*$; name this event 'ttl-tri-output-pads*-off'.

EVENT: Enable $\text{ttl-tri-output-pads}^*\$destructure$; name this event 'ttl-tri-output-pads*destructure-off'.

DEFINITION:

```
ttl-tri-output-pads&(netlist, n)
=  ((lookup-module(index('ttl-tri-output-pads, n), netlist)
                  =  ttl-tri-output-pads*(n))
   ^  ttl-tri-output&(delete-module(index('ttl-tri-output-pads,
                                           n),
```

netlist))

$\wedge$ **if** $n < 8$
then b-buf& (delete-module (index ('**t1-t1-output-pads**, n),
netlist))
else b-buf-pwr& (delete-module (index ('**t1-t1-output-pads**,
 n),
netlist)) **endif**)

EVENT: Enable t1-t1-output-pads&; name this event 't1-t1-output-pads&-off'.

DEFINITION:

t1-t1-output-pads\$*netlist* (n)
 $=$ cons (t1-t1-output-pads* (n),
if $n < 8$ **then** B-BUF\$NETLIST
else B-BUF-PWR\$NETLIST **endif**
 $\cup$ T1-T1-OUTPUT\$NETLIST)

THEOREM: t1-t1-output-pads-body\$unbound-in-body

($l < m$)
 $\rightarrow$ unbound-in-body (index ('**out**, l), t1-t1-output-pads-body (m , n))

EVENT: Enable t1-t1-output-pads-body\$unbound-in-body; name this event 't1-t1-output-pads-body\$unbound-in-body-off'.

THEOREM: t1-t1-output-pads-body\$value

t1-t1-output& (*netlist*)
 $\rightarrow$ (collect-value (indices ('**out**, m , n),
dual-eval (1,
t1-t1-output-pads-body (m , n),
bindings,
state-bindings,
netlist))
 $=$ vft-buf (f-not (value ('**enable-buf**, *bindings*)),
collect-value (indices ('**in**, m , n), *bindings*)))

EVENT: Enable t1-t1-output-pads-body\$value; name this event 't1-t1-output-pads-body\$value-off'.

THEOREM: t1-t1-output-pads\$value

(t1-t1-output-pads& (*netlist*, n)
 $\wedge$ (length (*inputs*) = n)
 $\wedge$ properp (*inputs*)
 $\rightarrow$ (dual-eval (0,

```

index('ttl-tri-output-pads, n),
cons(enable, inputs),
state,
netlist)
= vft-buf(f-not(enable), inputs))

```

EVENT: Enable ttl-tri-output-pads\$value; name this event 'ttl-tri-output-pads\$value-off'.

DEFINITION:

```

ttl-bidirect-pads-body(m, n, pi)
= if n ≈ 0 then nil
  else cons(list(index('g, m),
list(index('data, m),
index('out, m),
index('po, m)),
'ttl-bidirect,
list(index('data, m),
index('in, m),
'buf-enable,
pi)),
ttl-bidirect-pads-body(1 + m, n - 1, index('po, m))) endif

```

EVENT: Enable ttl-bidirect-pads-body; name this event 'ttl-bidirect-pads-body-off'.

DEFINITION:

```

ttl-bidirect-pads-body$induction(m, n, pi, bindings, state-bindings, netlist)
= if n ≈ 0 then t
  else ttl-bidirect-pads-body$induction(1 + m,
n - 1,
index('po, m),
dual-eval-body-bindings(1,
ttl-bidirect-pads-body(m,
n,
pi),
bindings,
state-bindings,
netlist),
state-bindings,
netlist) endif

```

DEFINITION:

```

ttl-bidirect-pads*(n)

```

```

= cons(index('ttl-bidirect-pads, n),
      cons(cons('enable,
                cons('pi,
                    append(indices('data, 0, n), indices('in, 0, n))))),
      cons(cons(index('po, n - 1),
                append(indices('data, 0, n), indices('out, 0, n))),
            cons(cons(cons('enable-buf,
                          cons('buf-enable,
                              cons(if n < 8 then 'b-buf
                                  else 'b-buf-pwr endif,
                                      '((enable))))),
                    ttl-bidirect-pads-body(0, n, 'pi)),
              '(nil))))))

```

THEOREM: $\text{ttl-bidirect-pads}^*\$destructure$

```

(car(ttl-bidirect-pads*(n)) = index('ttl-bidirect-pads, n))
 $\wedge$  (cadr(ttl-bidirect-pads*(n))
      = cons('enable,
              cons('pi, append(indices('data, 0, n), indices('in, 0, n))))))
 $\wedge$  (caddr(ttl-bidirect-pads*(n))
      = cons(index('po, n - 1),
              append(indices('data, 0, n), indices('out, 0, n))))
 $\wedge$  (cadddr(ttl-bidirect-pads*(n))
      = cons(cons('enable-buf,
                  cons('buf-enable,
                      cons(if n < 8 then 'b-buf
                          else 'b-buf-pwr endif,
                              '((enable))))),
              ttl-bidirect-pads-body(0, n, 'pi)))
 $\wedge$  (caddddr(ttl-bidirect-pads*(n)) = nil)

```

EVENT: Enable $\text{ttl-bidirect-pads}^*$; name this event ' $\text{ttl-bidirect-pads}^*\text{-off}$ '.

EVENT: Enable $\text{ttl-bidirect-pads}^*\$destructure$; name this event ' $\text{ttl-bidirect-pads}^*\$destructure\text{-off}$ '.

DEFINITION:

```

ttl-bidirect-pads&(netlist, n)
= ((lookup-module(index('ttl-bidirect-pads, n), netlist)
  = ttl-bidirect-pads*(n))
 $\wedge$  ttl-bidirect&(delete-module(index('ttl-bidirect-pads, n),
                                     netlist))
 $\wedge$  if n < 8

```

```

then b-buf& (delete-module (index ('t1-bidirect-pads, n),
                                netlist))
else b-buf-pwr& (delete-module (index ('t1-bidirect-pads,
                                n),
                                netlist)) endif

```

EVENT: Enable ttl-bidirect-pads&; name this event 'ttl-bidirect-pads&-off'.

DEFINITION:

```

ttl-bidirect-pads$netlist (n)
=  cons (ttl-bidirect-pads* (n),
        TTL-BIDIRECT$NETLIST
        ∪  if n < 8 then B-BUF$NETLIST
            else B-BUF-PWR$NETLIST endif)

```

THEOREM: ttl-bidirect-pads-body\$unbound-in-body

```

(l < m)
→  (unbound-in-body (index ('data, l), ttl-bidirect-pads-body (m, n, pi))
    ∧  unbound-in-body (index ('out, l),
                            ttl-bidirect-pads-body (m, n, pi))
    ∧  unbound-in-body (index ('po, l), ttl-bidirect-pads-body (m, n, pi))
    ∧  unbound-in-body (index ('in, l), ttl-bidirect-pads-body (m, n, pi)))

```

EVENT: Enable ttl-bidirect-pads-body\$unbound-in-body; name this event 'ttl-bidirect-pads-body\$unbound-in-body-off'.

THEOREM: ttl-bidirect-pads-body\$value

```

ttl-bidirect& (netlist)
→  ((collect-value (indices ('data, m, n),
                            dual-eval (1,
                                        ttl-bidirect-pads-body (m, n, pi),
                                        bindings,
                                        state-bindings,
                                        netlist)))
    =  if value ('buf-enable, bindings) = f
        then v-threefix (collect-value (indices ('in, m, n), bindings))
        elseif value ('buf-enable, bindings) = t
        then make-list (n, z)
        else make-list (n, x) endif)
    ∧  (collect-value (indices ('out, m, n),
                            dual-eval (1,
                                        ttl-bidirect-pads-body (m, n, pi),
                                        bindings,
                                        state-bindings,

```

```

                                netlist))
=  if value ('buf-enable, bindings) = f
    then v-threefix (v-wire (collect-value (indices ('data,
                                                    m,
                                                    n),
                                                    bindings),
                                v-threefix (collect-value (indices ('in,
                                                                    m,
                                                                    n),
                                                                    bindings))))
    elseif value ('buf-enable, bindings) = t
    then v-threefix (collect-value (indices ('data, m, n),
                                            bindings))
    else make-list (n, x) endif))

```

EVENT: Enable ttl-bidirect-pads-body\$value; name this event 'ttl-bidirect-pads-body\$value-off'.

THEOREM: ttl-bidirect-pads\$value
 (ttl-bidirect-pads& (*netlist*, *n*)
 $\wedge$ (length (*data*) = *n*)
 $\wedge$ properp (*data*)
 $\wedge$ (length (*inputs*) = *n*)
 $\wedge$ properp (*inputs*)
 $\rightarrow$ (cdr (dual-eval (0,
 index ('ttl-bidirect-pads, *n*),
 cons (*enable*, cons (*pi*, append (*data*, *inputs*))),
 state-bindings,
 netlist))
 = append (vft-buf (f-not (*enable*), *inputs*),
 v-threefix (v-wire (*data*, vft-buf (f-not (*enable*), *inputs*))))))

EVENT: Enable ttl-bidirect-pads\$value; name this event 'ttl-bidirect-pads\$value-off'.

DEFINITION: a-reg (*state*) = nth (2, *state*)

DEFINITION: b-reg (*state*) = nth (3, *state*)

DEFINITION: i-reg (*state*) = nth (4, *state*)

DEFINITION: data-out (*state*) = nth (5, *state*)

DEFINITION: addr-out (*state*) = nth (6, *state*)

DEFINITION: $\text{reset-}(state) = \text{nth}(7, state)$

DEFINITION: $\text{dtack-}(state) = \text{nth}(8, state)$

DEFINITION: $\text{hold-}(state) = \text{nth}(9, state)$

DEFINITION: $\text{pc-reg}(state) = \text{nth}(10, state)$

DEFINITION: $\text{cntl-state}(state) = \text{nth}(11, state)$

EVENT: Enable a-reg; name this event 'a-reg-off'.

EVENT: Enable b-reg; name this event 'b-reg-off'.

EVENT: Enable i-reg; name this event 'i-reg-off'.

EVENT: Enable data-out; name this event 'data-out-off'.

EVENT: Enable addr-out; name this event 'addr-out-off'.

EVENT: Enable reset-; name this event 'reset-off'.

EVENT: Enable dtack-; name this event 'dtack-off'.

EVENT: Enable hold-; name this event 'hold-off'.

EVENT: Enable pc-reg; name this event 'pc-reg-off'.

EVENT: Enable cntl-state; name this event 'cntl-state-off'.

EVENT: Let us define the theory *fm9001-hardware-state-accessors* to consist of the following events: regs, flags, a-reg, b-reg, i-reg, data-out, addr-out, reset-, dtack-, hold-, pc-reg, cntl-state.

DEFINITION: $\text{reset-input}(ext-in) = \text{nth}(0, ext-in)$

DEFINITION: $\text{hold-input}(ext-in) = \text{nth}(1, ext-in)$

DEFINITION: $\text{pc-reg-input}(ext-in) = \text{subrange}(ext-in, 2, 5)$

EVENT: Enable reset-input; name this event 'reset-input-off'.

EVENT: Enable hold-input; name this event ‘hold-input-off’.

EVENT: Enable pc-reg-input; name this event ‘pc-reg-input-off’.

EVENT: Let us define the theory *fm9001-external-input-accessors* to consist of the following events: reset-input, hold-input, pc-reg-input.

DEFINITION:

fm9001-next-state (*total-state*, *external-inputs*)

```

= list (list (f$write-regs (we-regs (v-threefix (cntl-state (car (total-state)))),
      regs-address (v-threefix (cntl-state (car (total-state)))),
      regs (car (total-state)),
      bv (f$score-alu (alu-c (v-threefix (cntl-state (car (total-state)))),
        v-threefix (a-reg (car (total-state))),
        v-threefix (b-reg (car (total-state))),
        alu-zero (v-threefix (cntl-state (car (total-state)))),
        alu-mpg (v-threefix (cntl-state (car (total-state)))),
        alu-op (v-threefix (cntl-state (car (total-state)))),
        make-tree (32))))),
    f$update-flags (flags (car (total-state)),
      we-flags (v-threefix (cntl-state (car (total-state)))),
      f$score-alu (alu-c (v-threefix (cntl-state (car (total-state)))),
        v-threefix (a-reg (car (total-state))),
        v-threefix (b-reg (car (total-state))),
        alu-zero (v-threefix (cntl-state (car (total-state)))),
        alu-mpg (v-threefix (cntl-state (car (total-state)))),
        alu-op (v-threefix (cntl-state (car (total-state)))),
        make-tree (32))),
    fv-if (we-a-reg (v-threefix (cntl-state (car (total-state)))),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state (car (total-state)))),
        f-not (dtack- (car (total-state)))),
      f$extend-immediate (select-immediate (v-threefix (cntl-state (car (total-state)))),
        a-immediate (v-threefix (i-reg (car (total-state)))),
        f$read-regs (regs-address (v-threefix (cntl-state (car (total-state)))),
          regs (car (total-state))),
        v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (total-state)))),
          v-threefix (data-out (car (total-state))))),
          cdr (memory-value (cadr (total-state)),
          f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state)))),
          strobe- (v-threefix (cntl-state (car (total-state)))))))))

```

```

f-pullup (ft-buf (f-buf (hdack- (v-
                                f-buf (rw- (v-thr
v-pullup (vft-buf (f-buf (hdack- (
                                v-threefix (ad
make-list (32,
            x))))),
vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (total-state)))))),
v-threefix (data-out (car (total-state)))))),
a-reg (car (total-state))),
fv-if (we-b-reg (v-threefix (cntl-state (car (total-state))))),
fv-if (f-nand (data-in-select (v-threefix (cntl-state (car (total-state))))),
f-not (dtack- (car (total-state))))),
f$extend-immediate (select-immediate (v-threefix (cntl-state (car (total-state))))),
a-immediate (v-threefix (i-reg (car (total-state))))),
f$read-regs (regs-address (v-threefix (cntl-state (car (total-state))))),
regs (car (total-state))),
v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (
v-threefix (data-out (car (total-state))))),
cdr (memory-value (cadr (total-state),
f-pullup (ft-buf (f-buf (hdack- (v-
                                strobe- (v-three
f-pullup (ft-buf (f-buf (hdack- (v-
                                f-buf (rw- (v-thr
v-pullup (vft-buf (f-buf (hdack- (
                                v-threefix (ad
make-list (32,
            x))))),
vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (total-state)))))),
v-threefix (data-out (car (total-state)))))),
b-reg (car (total-state))),
fv-if (we-i-reg (v-threefix (cntl-state (car (total-state))))),
fv-if (f-nand (data-in-select (v-threefix (cntl-state (car (total-state))))),
f-not (dtack- (car (total-state))))),
f$extend-immediate (select-immediate (v-threefix (cntl-state (car (total-state))))),
a-immediate (v-threefix (i-reg (car (total-state))))),
f$read-regs (regs-address (v-threefix (cntl-state (car (total-state))))),
regs (car (total-state))),
v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (
v-threefix (data-out (car (total-state))))),
cdr (memory-value (cadr (total-state),
f-pullup (ft-buf (f-buf (hdack- (v-
                                strobe- (v-three
f-pullup (ft-buf (f-buf (hdack- (v-
                                f-buf (rw- (v-thr

```

```

v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
v-threefix (addr-out (car (total-state))))),
make-list (32,
x))))),
vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (total-state))))),
v-threefix (data-out (car (total-state)))))),
i-reg (car (total-state))),
fv-if (we-data-out (v-threefix (cntl-state (car (total-state))))),
bv (f$core-alu (alu-c (v-threefix (cntl-state (car (total-state))))),
v-threefix (a-reg (car (total-state))),
v-threefix (b-reg (car (total-state))),
alu-zero (v-threefix (cntl-state (car (total-state))))),
alu-mpg (v-threefix (cntl-state (car (total-state))))),
alu-op (v-threefix (cntl-state (car (total-state))))),
make-tree (32))),
data-out (car (total-state))),
fv-if (we-addr-out (v-threefix (cntl-state (car (total-state))))),
f$dec-pass (dec-addr-out (v-threefix (cntl-state (car (total-state))))),
f$extend-immediate (select-immediate (v-threefix (cntl-state (car (total-state))))),
a-immediate (v-threefix (i-reg (car (total-state))))),
f$read-regs (regs-address (v-threefix (cntl-state (car (total-state))))),
regs (car (total-state))))),
addr-out (car (total-state))),
f-buf (reset-input (external-inputs)),
f-or (strobe- (v-threefix (cntl-state (car (total-state))))),
f-buf (car (memory-value (cadr (total-state),
f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
strobe- (v-threefix (cntl-state (car (total-state))))),
f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
f-buf (rw- (v-threefix (cntl-state (car (total-state))))),
v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
v-threefix (addr-out (car (total-state))))),
make-list (32, x))))),
f-if (we-hold- (v-threefix (cntl-state (car (total-state))))),
f-buf (hold-input (external-inputs)),
hold- (car (total-state))),
fv-if (we-pc-reg (v-threefix (cntl-state (car (total-state))))),
pc-reg-input (external-inputs),
pc-reg (car (total-state))),
v-threefix (f$next-cntl-state (f-buf (reset- (car (total-state))),
f-buf (dtack- (car (total-state))),
f-buf (hold- (car (total-state))),
rw- (v-threefix (cntl-state (car (total-state))))),
state (v-threefix (cntl-state (car (total-state))))),

```

```

v-threefix (i-reg (car (total-state))),
v-threefix (flags (car (total-state))),
v-threefix (pc-reg (car (total-state))),
regs-address (v-threefix (cntl-state (car (total-state))))),
next-memory-state (cadr (total-state),
  f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
    strobe- (v-threefix (cntl-state (car (total-state))))),
  f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
    f-buf (rw- (v-threefix (cntl-state (car (total-state))))),
  v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
    v-threefix (addr-out (car (total-state))))),
  v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state (car (total-state))))),
    v-threefix (data-out (car (total-state))))),
  cdr (memory-value (cadr (total-state),
    f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
      strobe- (v-threefix (cntl-state (car (total-state))))),
    f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
      f-buf (rw- (v-threefix (cntl-state (car (total-state))))),
    v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state (car (total-state))))),
      v-threefix (addr-out (car (total-state))))),
  make-list (32,
    x))))))

```

EVENT: Enable fm9001-next-state; name this event ‘fm9001-next-state-off’.

DEFINITION:

```

run-fm9001 (state, inputs, n)
=  if n ≈ 0 then state
   else run-fm9001 (fm9001-next-state (state, car (inputs)),
     cdr (inputs),
     n - 1) endif

```

EVENT: Enable run-fm9001; name this event ‘run-fm9001-off’.

THEOREM: run-fm9001-base-case

$(n \simeq 0) \rightarrow (\text{run-fm9001} (state, inputs, n) = state)$

THEOREM: run-fm9001-step-case

```

(n ≠ 0)
→  (run-fm9001 (state, inputs, n)
   =  run-fm9001 (fm9001-next-state (state, car (inputs)),
     cdr (inputs),
     n - 1))

```

EVENT: Enable run-fm9001-step-case; name this event ‘run-fm9001-step-case-off’.

THEOREM: run-fm9001-plus
 $\text{run-fm9001}(\text{state}, \text{inputs}, n + m)$
 $= \text{run-fm9001}(\text{run-fm9001}(\text{state}, \text{inputs}, n), \text{nthcdr}(n, \text{inputs}), m)$

DEFINITION:
 $\text{fm9001-state-structure}(\text{state})$
 $= ((\text{length}(\text{state}) = 2)$
 $\wedge \text{properp}(\text{state})$
 $\wedge (\text{length}(\text{car}(\text{state})) = 12)$
 $\wedge \text{properp}(\text{car}(\text{state}))$
 $\wedge (\text{length}(\text{cadr}(\text{state})) = 8)$
 $\wedge \text{properp}(\text{cadr}(\text{state}))$
 $\wedge (\text{length}(\text{caar}(\text{state})) = 4)$
 $\wedge \text{properp}(\text{caar}(\text{state})))$

THEOREM: fm9001-state-structure\$step
 $\text{fm9001-state-structure}(\text{state})$
 $\rightarrow \text{fm9001-state-structure}(\text{fm9001-next-state}(\text{state}, \text{external-inputs}))$

THEOREM: fm9001-state-structure\$induction
 $\text{fm9001-state-structure}(\text{state})$
 $\rightarrow \text{fm9001-state-structure}(\text{run-fm9001}(\text{state}, \text{inputs}, \text{steps}))$

THEOREM: fm9001-state-as-a-list
 $\text{fm9001-state-structure}(\text{state})$
 $\rightarrow (\text{state} = \text{list}(\text{list}(\text{list}(\text{caaar}(\text{state}),$
 $\text{cadaar}(\text{state}),$
 $\text{caddaar}(\text{state}),$
 $\text{cadddaar}(\text{state})),$
 $\text{cadar}(\text{state}),$
 $\text{caddar}(\text{state}),$
 $\text{cadddar}(\text{state}),$
 $\text{caddddar}(\text{state}),$
 $\text{cadddddar}(\text{state}),$
 $\text{cadddddar}(\text{state}),$
 $\text{caddddddar}(\text{state}),$
 $\text{cadddddddar}(\text{state}),$
 $\text{cadddddddar}(\text{state}),$
 $\text{cadddddddar}(\text{state}),$
 $\text{cadddddddar}(\text{state}),$
 $\text{cadddddddar}(\text{state})),$
 $\text{list}(\text{caadr}(\text{state}),$

```

cadadr (state),
caddadr (state),
cadddadr (state),
caddddadr (state),
cadddddadr (state),
caddddddadr (state),
cadddddddadr (state))))

```

```

(DEFN CHIP-MODULE* NIL
  (CONS 'CHIP-MODULE
    (CONS
      (CONS 'CLK
        (CONS 'TI
          (CONS 'TE
            (CONS 'DTACK-
              (CONS 'RESET-
                (CONS 'HOLD-
                  (CONS 'DISABLE-REGFILE-
                    (CONS 'TEST-REGFILE-
                      (APPEND (INDICES 'PC-REG-IN 0 4)
                        (INDICES 'DATA-IN 0 32))))))))))
        (CONS
          (CONS 'TO
            (CONS 'TIMING
              (CONS 'HDACK-
                (CONS 'EN-ADDR-OUT-
                  (CONS 'RW-
                    (CONS 'STROBE-
                      (APPEND (INDICES 'ADDR-OUT 0 32)
                        (APPEND (INDICES 'DATA-OUT 0 32)
                          (APPEND (INDICES 'FLAGS 0 4)
                            (APPEND (INDICES 'CNTL-STATE 0 5)
                              (SUBRANGE (INDICES 'I-REG 0 32)
                                28 31))))))))))
                    (CONS
                      (LIST
                        (LIST 'CNTL-STATE
                          (CONS 'RW-SIG-
                            (CONS 'STROBE-
                              (CONS 'HDACK-
                                (CONS 'WE-REGS
                                  (CONS 'WE-A-REG

```

```

(CONS 'WE-B-REG
(CONS 'WE-I-REG
(CONS 'WE-DATA-OUT
(CONS 'WE-ADDR-OUT
(CONS 'WE-HOLD-
(CONS 'WE-PC-REG
(CONS 'DATA-IN-SELECT
(CONS 'DEC-ADDR-OUT
(CONS 'SELECT-IMMEDIATE
(CONS 'ALU-C
(CONS 'ALU-ZERO
(APPEND
  (INDICES 'CNTL-STATE 0 5)
  (APPEND (INDICES 'WE-FLAGS 0 4)
    (APPEND (INDICES 'REGS-ADDRESS 0 4)
      (APPEND (INDICES 'ALU-OP 0 4)
        (INDICES 'ALU-MPG
          0
          7))))))))))))))))))
(INDEX 'REG 40)
(CONS 'CLK
  (CONS 'TE-SIG
    (CONS 'TI
      (INDICES 'NEXT-STATE 0 40))))
(LIST 'REGS
  (CONS 'REGFILE-TO
    (INDICES 'REGFILE-OUT 0 32))
  'REGFILE
  (CONS 'CLK
    (CONS 'TE-SIG
      (CONS (INDEX 'ALU-MPG 6)
        (CONS 'WE-REGS
          (CONS 'DISABLE-REGFILE-
            (CONS 'TEST-REGFILE-
              (APPEND (INDICES 'REGS-ADDRESS 0 4)
                (BV (INDICES 'ALU-BUS 0 35))))))))))
(LIST 'CVNZ-FLAGS
  (INDICES 'FLAGS 0 4)
  'FLAGS
  (CONS 'CLK
    (CONS 'TE-SIG
      (CONS 'REGFILE-TO
        (APPEND (INDICES 'WE-FLAGS 0 4)
          (INDICES 'ALU-BUS 0 35))))))

```

```

(LIST 'A-REG
      (INDICES 'A-REG 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
            (CONS 'WE-A-REG
                  (CONS 'TE-SIG
                        (CONS (INDEX 'FLAGS 3)
                              (INDICES 'ABI-BUS 0 32)))))))
(LIST 'B-REG
      (INDICES 'B-REG 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
            (CONS 'WE-B-REG
                  (CONS 'TE-SIG
                        (CONS (INDEX 'A-REG 31)
                              (INDICES 'ABI-BUS 0 32)))))))
(LIST 'I-REG
      (INDICES 'I-REG 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
            (CONS 'WE-I-REG
                  (CONS 'TE-SIG
                        (CONS (INDEX 'B-REG 31)
                              (INDICES 'ABI-BUS 0 32)))))))
(LIST 'DATA-OUT
      (INDICES 'DATA-OUT 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
            (CONS 'WE-DATA-OUT
                  (CONS 'TE-SIG
                        (CONS (INDEX 'I-REG 31)
                              (BV (INDICES 'ALU-BUS 0 35)))))))
(LIST 'ADDR-OUT
      (INDICES 'ADDR-OUT 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
            (CONS 'WE-ADDR-OUT
                  (CONS 'TE-SIG
                        (CONS (INDEX 'DATA-OUT 31)
                              (INDICES 'ADDR-OUT-BUS 0 32)))))))
(LIST 'RESET-LATCH
      '(LAST-RESET- LAST-RESET-INV)
      'FD1S
      (LIST 'RESET-

```



```

        'CLK
        (INDEX 'ADDR-OUT 31)
        'TE-SIG))
' (DTACK--OR (DTACK--OR-STROBE-)
  B-OR
  (STROBE- DTACK-))
' (DTACK-LATCH (LAST-DTACK- LAST-DTACK-INV)
  FD1S
  (DTACK--OR-STROBE- CLK LAST-RESET- TE-SIG))
' (HOLD-LATCH (LAST-HOLD- LAST-HOLD-INV)
  FD1SLP
  (HOLD- CLK WE-HOLD- LAST-DTACK- TE-SIG))
(LIST 'PC-REG
  (INDICES 'PC-REG 0 4)
  (INDEX 'WE-REG 4)
  (CONS 'CLK
    (CONS 'WE-PC-REG
      (CONS 'TE-SIG
        (CONS 'LAST-HOLD-
          (INDICES 'PC-REG-IN 0 4))))))
(LIST 'IMMEDIATE-PASS
  (INDICES 'REG-BUS 0 32)
  'EXTEND-IMMEDIATE
  (CONS 'SELECT-IMMEDIATE
    (APPEND (A-IMMEDIATE (INDICES 'I-REG 0 32))
      (INDICES 'REGFILE-OUT 0 32))))
(LIST 'DEC-PASS
  (INDICES 'ADDR-OUT-BUS 0 32)
  (INDEX 'DEC-PASS 32)
  (CONS 'DEC-ADDR-OUT
    (INDICES 'REG-BUS 0 32)))
' (MUX-CNTL (ABI-CNTL)
  B-NAND
  (DATA-IN-SELECT LAST-DTACK-INV))
(LIST 'DATA-IN-MUX
  (INDICES 'ABI-BUS 0 32)
  (INDEX 'TV-IF
    (TREE-NUMBER (MAKE-TREE 32)))
  (CONS 'ABI-CNTL
    (APPEND (INDICES 'REG-BUS 0 32)
      (INDICES 'DATA-IN 0 32))))
(LIST 'ALU
  (INDICES 'ALU-BUS 0 35)
  (INDEX 'CORE-ALU

```

```

        (TREE-NUMBER (MAKE-TREE 32)))
(CONS 'ALU-C
      (APPEND (INDICES 'A-REG 0 32)
                (APPEND (INDICES 'B-REG 0 32)
                        (CONS 'ALU-ZERO
                            (APPEND (INDICES 'ALU-MPG 0 7)
                                    (INDICES 'ALU-OP 0 4)))))))
(LIST 'NEXT-STATE
      (INDICES 'NEXT-STATE 0 40)
      'NEXT-CNTL-STATE
      (CONS 'LAST-RESET-
            (CONS 'LAST-DTACK-
                  (CONS 'LAST-HOLD-
                        (CONS 'RW-SIG-
                              (APPEND
                                (INDICES 'CNTL-STATE 0 5)
                                (APPEND (INDICES 'I-REG 0 32)
                                        (APPEND (INDICES 'FLAGS 0 4)
                                                (APPEND (INDICES 'PC-REG 0 4)
                                                        (INDICES 'REGS-ADDRESS 0 4))))))))))
      ' (TE-BUFFER (TE-SIG) B-BUF-PWR (TE))
      ' (RW-BUFFER (RW-) B-BUF (RW-SIG-))
      ' (EN-ADDR-OUT-GATE (EN-ADDR-OUT-)
        B-NOT
        (HDACK-))
(LIST 'TIMING-GATE
      '(TIMING)
      'ID
      (LIST (INDEX 'ALU-BUS 2)))
(LIST 'SCANOUT
      '(TO)
      'ID
      (LIST (INDEX 'PC-REG 3)))
' ((REGS CVNZ-FLAGS A-REG B-REG I-REG DATA-OUT ADDR-OUT RESET-LATCH
      DTACK-LATCH HOLD-LATCH PC-REG CNTL-STATE))))))

```

```

(PROVE-LEMMA CHIP-MODULE*$DESTRUCTURE
  (REWRITE)
  (AND
    (EQUAL (CAR (CHIP-MODULE*))
            'CHIP-MODULE)
    (EQUAL

```

```

(CADR (CHIP-MODULE*))
(CONS 'CLK
(CONS 'TI
(CONS 'TE
  (CONS 'DTACK-
    (CONS 'RESET-
      (CONS 'HOLD-
        (CONS 'DISABLE-REGFILE-
          (CONS 'TEST-REGFILE-
            (APPEND (INDICES 'PC-REG-IN 0 4)
              (INDICES 'DATA-IN 0 32))))))))))
(EQUAL
(CADDR (CHIP-MODULE*))
(CONS 'TO
(CONS 'TIMING
(CONS 'HDACK-
(CONS 'EN-ADDR-OUT-
(CONS 'RW-
(CONS 'STROBE-
  (APPEND (INDICES 'ADDR-OUT 0 32)
    (APPEND (INDICES 'DATA-OUT 0 32)
      (APPEND (INDICES 'FLAGS 0 4)
        (APPEND (INDICES 'CNTL-STATE 0 5)
          (SUBRANGE (INDICES 'I-REG 0 32)
            28 31))))))))))
(EQUAL
(CADDR (CHIP-MODULE*))
(LIST
(LIST 'CNTL-STATE
(CONS 'RW-SIG-
(CONS 'STROBE-
(CONS 'HDACK-
(CONS 'WE-REGS
(CONS 'WE-A-REG
(CONS 'WE-B-REG
(CONS 'WE-I-REG
(CONS 'WE-DATA-OUT
(CONS 'WE-ADDR-OUT
(CONS 'WE-HOLD-
(CONS 'WE-PC-REG
(CONS 'DATA-IN-SELECT
(CONS 'DEC-ADDR-OUT
(CONS 'SELECT-IMMEDIATE
(CONS 'ALU-C

```

```

(CONS 'ALU-ZERO
  (APPEND
    (INDICES 'CNTL-STATE 0 5)
    (APPEND (INDICES 'WE-FLAGS 0 4)
      (APPEND (INDICES 'REGS-ADDRESS 0 4)
        (APPEND (INDICES 'ALU-OP 0 4)
          (INDICES 'ALU-MPG
            0
            7))))))))))))))))))
(INDEX 'REG 40)
(CONS 'CLK
  (CONS 'TE-SIG
    (CONS 'TI
      (INDICES 'NEXT-STATE 0 40))))))
(LIST 'REGS
  (CONS 'REGFILE-TO
    (INDICES 'REGFILE-OUT 0 32))
  'REGFILE
  (CONS 'CLK
    (CONS 'TE-SIG
      (CONS (INDEX 'ALU-MPG 6)
        (CONS 'WE-REGS
          (CONS 'DISABLE-REGFILE-
            (CONS 'TEST-REGFILE-
              (APPEND (INDICES 'REGS-ADDRESS 0 4)
                (BV (INDICES 'ALU-BUS 0 35))))))))))
  (LIST 'CVNZ-FLAGS
    (INDICES 'FLAGS 0 4)
    'FLAGS
    (CONS 'CLK
      (CONS 'TE-SIG
        (CONS 'REGFILE-TO
          (APPEND (INDICES 'WE-FLAGS 0 4)
            (INDICES 'ALU-BUS 0 35))))))
  (LIST 'A-REG
    (INDICES 'A-REG 0 32)
    (INDEX 'WE-REG 32)
    (CONS 'CLK
      (CONS 'WE-A-REG
        (CONS 'TE-SIG
          (CONS (INDEX 'FLAGS 3)
            (INDICES 'ABI-BUS 0 32))))))
  (LIST 'B-REG
    (INDICES 'B-REG 0 32)

```

```

      (INDEX 'WE-REG 32)
      (CONS 'CLK
        (CONS 'WE-B-REG
          (CONS 'TE-SIG
            (CONS (INDEX 'A-REG 31)
              (INDICES 'ABI-BUS 0 32))))))
    (LIST 'I-REG
      (INDICES 'I-REG 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
        (CONS 'WE-I-REG
          (CONS 'TE-SIG
            (CONS (INDEX 'B-REG 31)
              (INDICES 'ABI-BUS 0 32))))))
    (LIST 'DATA-OUT
      (INDICES 'DATA-OUT 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
        (CONS 'WE-DATA-OUT
          (CONS 'TE-SIG
            (CONS (INDEX 'I-REG 31)
              (BV (INDICES 'ALU-BUS 0 35))))))
    (LIST 'ADDR-OUT
      (INDICES 'ADDR-OUT 0 32)
      (INDEX 'WE-REG 32)
      (CONS 'CLK
        (CONS 'WE-ADDR-OUT
          (CONS 'TE-SIG
            (CONS (INDEX 'DATA-OUT 31)
              (INDICES 'ADDR-OUT-BUS 0 32))))))
    (LIST 'RESET-LATCH
      ' (LAST-RESET- LAST-RESET-INV)
      ' FD1S
      (LIST 'RESET-
        ' CLK
        (INDEX 'ADDR-OUT 31)
        ' TE-SIG))
    ' (DTACK--OR (DTACK--OR-STROBE-)
      B-OR
      (STROBE- DTACK-))
    ' (DTACK-LATCH (LAST-DTACK- LAST-DTACK-INV)
      FD1S
      (DTACK--OR-STROBE- CLK LAST-RESET- TE-SIG))
    ' (HOLD-LATCH (LAST-HOLD- LAST-HOLD-INV)

```

```

        FD1SLP
        (HOLD- CLK WE-HOLD- LAST-DTACK- TE-SIG))
(LIST 'PC-REG
      (INDICES 'PC-REG 0 4)
      (INDEX 'WE-REG 4)
      (CONS 'CLK
            (CONS 'WE-PC-REG
                  (CONS 'TE-SIG
                        (CONS 'LAST-HOLD-
                              (INDICES 'PC-REG-IN 0 4)))))))
(LIST 'IMMEDIATE-PASS
      (INDICES 'REG-BUS 0 32)
      'EXTEND-IMMEDIATE
      (CONS 'SELECT-IMMEDIATE
            (APPEND (A-IMMEDIATE (INDICES 'I-REG 0 32))
                    (INDICES 'REGFILE-OUT 0 32))))
(LIST 'DEC-PASS
      (INDICES 'ADDR-OUT-BUS 0 32)
      (INDEX 'DEC-PASS 32)
      (CONS 'DEC-ADDR-OUT
            (INDICES 'REG-BUS 0 32)))
' (MUX-CNTL (ABI-CNTL)
          B-NAND
          (DATA-IN-SELECT LAST-DTACK-INV))
(LIST 'DATA-IN-MUX
      (INDICES 'ABI-BUS 0 32)
      (INDEX 'TV-IF
            (TREE-NUMBER (MAKE-TREE 32)))
      (CONS 'ABI-CNTL
            (APPEND (INDICES 'REG-BUS 0 32)
                    (INDICES 'DATA-IN 0 32))))
(LIST 'ALU
      (INDICES 'ALU-BUS 0 35)
      (INDEX 'CORE-ALU
            (TREE-NUMBER (MAKE-TREE 32)))
      (CONS 'ALU-C
            (APPEND (INDICES 'A-REG 0 32)
                    (APPEND (INDICES 'B-REG 0 32)
                            (CONS 'ALU-ZERO
                                  (APPEND (INDICES 'ALU-MPG 0 7)
                                            (INDICES 'ALU-OP 0 4)))))))
(LIST 'NEXT-STATE
      (INDICES 'NEXT-STATE 0 40)
      'NEXT-CNTL-STATE

```

```

(CONS 'LAST-RESET-
  (CONS 'LAST-DTACK-
    (CONS 'LAST-HOLD-
      (CONS 'RW-SIG-
        (APPEND (INDICES 'CNTL-STATE 0 5)
          (APPEND (INDICES 'I-REG 0 32)
            (APPEND (INDICES 'FLAGS 0 4)
              (APPEND (INDICES 'PC-REG 0 4)
                (INDICES 'REGS-ADDRESS 0 4))))))))))
' (TE-BUFFER (TE-SIG) B-BUF-PWR (TE))
' (RW-BUFFER (RW-) B-BUF (RW-SIG-))
' (EN-ADDR-OUT-GATE (EN-ADDR-OUT-)
  B-NOT
  (HDACK-))
(LIST 'TIMING-GATE
  ' (TIMING)
  ' ID
  (LIST (INDEX 'ALU-BUS 2)))
(LIST 'SCANOUT
  ' (TO)
  ' ID
  (LIST (INDEX 'PC-REG 3))))
(EQUAL (CADDDDR (CHIP-MODULE*))
  ' (REGS CVNZ-FLAGS A-REG B-REG I-REG DATA-OUT ADDR-OUT RESET-LATCH
    DTACK-LATCH HOLD-LATCH PC-REG CNTL-STATE))))

```

EVENT: Enable chip-module*; name this event 'chip-module*-off'.

EVENT: Enable chip-module*\$destructure; name this event 'chip-module*\$destructure-off'.

EVENT: Enable *1*chip-module*; name this event 'g*1*chip-module*-off'.

DEFINITION:

chip-module& (*netlist*)

```

= ((lookup-module('chip-module, netlist) = CHIP-MODULE*)
  ^ regfile& (delete-module('chip-module, netlist))
  ^ flags& (delete-module('chip-module, netlist))
  ^ we-reg& (delete-module('chip-module, netlist), 32)
  ^ fdls& (delete-module('chip-module, netlist))
  ^ fdlslp& (delete-module('chip-module, netlist))
  ^ we-reg& (delete-module('chip-module, netlist), 4)
  ^ reg& (delete-module('chip-module, netlist), 40)

```

```

^ id& (delete-module ('chip-module, netlist))
^ b-buf-pwr& (delete-module ('chip-module, netlist))
^ b-buf& (delete-module ('chip-module, netlist))
^ b-not& (delete-module ('chip-module, netlist))
^ extend-immediate& (delete-module ('chip-module, netlist))
^ dec-pass& (delete-module ('chip-module, netlist), 32)
^ tv-if& (delete-module ('chip-module, netlist), make-tree (32))
^ core-alu& (delete-module ('chip-module, netlist),
              make-tree (32))
^ next-cntl-state& (delete-module ('chip-module, netlist)))

```

EVENT: Enable chip-module&; name this event 'chip-module&-off'.

DEFINITION:

```

CHIP-MODULE$NETLIST
= cons (CHIP-MODULE*,
        REGFILE$NETLIST
        U (((((FLAGS$NETLIST U we-reg$netlist (32))
              U (FD1S$NETLIST U FD1SLP$NETLIST))
              U (((we-reg$netlist (4) U reg$netlist (40))
                  U (ID$NETLIST
                    U EXTEND-IMMEDIATE$NETLIST)))
              U ((dec-pass$netlist (32)
                  U tv-if$netlist (make-tree (32)))
                  U (core-alu$netlist (make-tree (32))
                    U NEXT-CNTL-STATE$NETLIST))))))
        U (B-BUF-PWR$NETLIST
          U (B-BUF$NETLIST U B-NOT$NETLIST))))

```

THEOREM: equal-length-40-as-collected-nth+subrange

```

((length (l) = 40) ^ properp (l))
→ (l = append (list-as-collected-nth (l, 16, 0), subrange (l, 16, 39)))

```

```

(PROVE-LEMMA LIST-AS-CNTL-STATE-CROCK NIL
 (IMPLIES
  (AND (PROPERP LIST)
        (EQUAL (LENGTH LIST) 40))
  (EQUAL LIST
   (CONS
    (RW- LIST)
    (CONS
     (STROBE- LIST)

```



```

(CONS
  (HDACK- LIST)
  (CONS
    (WE-REGS LIST)
    (CONS
      (WE-A-REG LIST)
      (CONS
        (WE-B-REG LIST)
        (CONS
          (WE-I-REG LIST)
          (CONS
            (WE-DATA-OUT LIST)
            (CONS
              (WE-ADDR-OUT LIST)
              (CONS
                (WE-HOLD- LIST)
                (CONS
                  (WE-PC-REG LIST)
                  (CONS
                    (DATA-IN-SELECT LIST)
                    (CONS
                      (DEC-ADDR-OUT LIST)
                      (CONS
                        (SELECT-IMMEDIATE LIST)
                        (CONS
                          (ALU-C LIST)
                          (CONS
                            (ALU-ZERO LIST)
                            (APPEND
                              (STATE LIST)
                              (APPEND
                                (WE-FLAGS LIST)
                                (APPEND (REGS-ADDRESS LIST)
                                  (APPEND (ALU-OP LIST)
                                    (ALU-MPG LIST))))))))))))))))))
  ((ENABLE-THEORY CONTROL-STATE-ACCESSOR-THEORY)
   (ENABLE OPEN-SUBRANGE)
   (USE (EQUAL-LENGTH-40-AS-COLLECTED-NTH+SUBRANGE (L LIST))))))

```

```

(PROVE-LEMMA REG-40$VALUE-AS-CNTL-STATE
  (REWRITE)
  (IMPLIES

```

```

(AND (REG& NETLIST 40)
      (PROPERP STATE)
      (EQUAL (LENGTH STATE) 40))
(EQUAL
  (DUAL-EVAL 0
    (INDEX 'REG 40)
    ARGS STATE NETLIST)
  (CONS
    (RW- (V-THREEFIX STATE))
    (CONS
      (STROBE- (V-THREEFIX STATE))
      (CONS
        (HDACK- (V-THREEFIX STATE))
        (CONS
          (WE-REGS (V-THREEFIX STATE))
          (CONS
            (WE-A-REG (V-THREEFIX STATE))
            (CONS
              (WE-B-REG (V-THREEFIX STATE))
              (CONS
                (WE-I-REG (V-THREEFIX STATE))
                (CONS
                  (WE-DATA-OUT (V-THREEFIX STATE))
                  (CONS
                    (WE-ADDR-OUT (V-THREEFIX STATE))
                    (CONS
                      (WE-HOLD- (V-THREEFIX STATE))
                      (CONS
                        (WE-PC-REG (V-THREEFIX STATE))
                        (CONS
                          (DATA-IN-SELECT (V-THREEFIX STATE))
                          (CONS
                            (DEC-ADDR-OUT (V-THREEFIX STATE))
                            (CONS
                              (SELECT-IMMEDIATE (V-THREEFIX STATE))
                              (CONS
                                (ALU-C (V-THREEFIX STATE))
                                (CONS
                                  (ALU-ZERO (V-THREEFIX STATE))
                                  (APPEND
                                    (STATE (V-THREEFIX STATE))
                                    (APPEND
                                      (WE-FLAGS (V-THREEFIX STATE))
                                      (APPEND

```

```

( (REGS-ADDRESS (V-THREEFIX STATE))
  (APPEND
    (ALU-OP (V-THREEFIX STATE))
    (ALU-MPG (V-THREEFIX STATE))))))))))))))
((ENABLE REG$VALUE)
 (USE (LIST-AS-CNTL-STATE-CROCK (LIST (V-THREEFIX STATE)))))

```

EVENT: Enable reg-40\$value-as-cntl-state; name this event ‘reg-40\$value-as-cntl-state-off’.

THEOREM: bv-as-subrange

```

(if length(x) < 3 then f
  else t endif
  ^ properp(x))
→ (bv(x) = subrange(x, 3, length(x) - 1))

```

EVENT: Enable bv-as-subrange; name this event ‘bv-as-subrange-off’.

DEFINITION:

```

machine-state-invariant(machine-state)
= (all-ramp-mem(4, caar(machine-state))
  ^ memory-properp(4, 32, caar(machine-state))
  ^ properp(caddar(machine-state))
  ^ (length(caddar(machine-state)) = 32)
  ^ properp(caddddar(machine-state))
  ^ (length(caddddar(machine-state)) = 4)
  ^ properp(cadr(machine-state))
  ^ (length(cadr(machine-state)) = 4)
  ^ properp(caddr(machine-state))
  ^ (length(caddr(machine-state)) = 32)
  ^ properp(caddddr(machine-state))
  ^ (length(caddddr(machine-state)) = 32)
  ^ properp(cadddddr(machine-state))
  ^ (length(cadddddr(machine-state)) = 32)
  ^ properp(caddddddr(machine-state))
  ^ (length(caddddddr(machine-state)) = 32)
  ^ properp(cadddddddrr(machine-state))
  ^ (length(cadddddddrr(machine-state)) = 4)
  ^ properp(cadddddddrrr(machine-state))
  ^ (length(cadddddddrrr(machine-state)) = 40))

```

THEOREM: chip-module\$state

```

(chip-module& (netlist)
  ^ machine-state-invariant (list (list (regs-regs,
                                         regs-we,
                                         regs-data,
                                         regs-addr),
                                         flags,
                                         a-reg,
                                         b-reg,
                                         i-reg,
                                         data-out,
                                         addr-out,
                                         last-reset-,
                                         last-dtack-,
                                         last-hold-,
                                         last-pc-reg,
                                         cntl-state))
  ^ (te = f)
  ^ (disable-regfile- = t)
  ^ (test-regfile- = t)
  ^ properp (pc-reg-in)
  ^ (length (pc-reg-in) = 4)
  ^ properp (data-in)
  ^ (length (data-in) = 32))
→ (dual-eval (2,
              'chip-module,
              cons (clk,
                    cons (ti,
                          cons (te,
                                cons (dtack-,
                                      cons (reset-,
                                            cons (hold-,
                                                  cons (disable-regfile-,
                                                        cons (test-regfile-,
                                                                append (pc-reg-in,
                                                                    data-in))))))))),
              list (list (regs-regs, regs-we, regs-data, regs-addr),
                        flags,
                        a-reg,
                        b-reg,
                        i-reg,
                        data-out,
                        addr-out,
                        last-reset-,
                        last-dtack-,

```

```

        last-hold-,
        last-pc-reg,
        cntl-state),
    netlist)
= list (f$write-regs (we-regs (v-threefix (cntl-state)),
    regs-address (v-threefix (cntl-state)),
    list (regs-regs, regs-we, regs-data, regs-addr),
    bv (f$core-alu (alu-c (v-threefix (cntl-state)),
        v-threefix (a-reg),
        v-threefix (b-reg),
        alu-zero (v-threefix (cntl-state)),
        alu-mpg (v-threefix (cntl-state)),
        alu-op (v-threefix (cntl-state)),
        make-tree (32))))),
    f$update-flags (flags,
        we-flags (v-threefix (cntl-state)),
        f$core-alu (alu-c (v-threefix (cntl-state)),
            v-threefix (a-reg),
            v-threefix (b-reg),
            alu-zero (v-threefix (cntl-state)),
            alu-mpg (v-threefix (cntl-state)),
            alu-op (v-threefix (cntl-state)),
            make-tree (32))),
    fv-if (we-a-reg (v-threefix (cntl-state)),
        fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
            f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                a-immediate (v-threefix (i-reg)),
                f$read-regs (regs-address (v-threefix (cntl-state)),
                    list (regs-regs,
                        regs-we,
                        regs-data,
                        regs-addr))),
            data-in),
        a-reg),
    fv-if (we-b-reg (v-threefix (cntl-state)),
        fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
            f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                a-immediate (v-threefix (i-reg)),
                f$read-regs (regs-address (v-threefix (cntl-state)),
                    list (regs-regs,
                        regs-we,
                        regs-data,

```

```

                                regs-addr))),
                                data-in),
                                b-reg),
fv-if (we-i-reg (v-threefix (cntl-state)),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state))),
              f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                  a-immediate (v-threefix (i-reg)),
                                  f$read-regs (regs-address (v-threefix (cntl-state)),
                                                list (regs-regs,
                                                      regs-we,
                                                      regs-data,
                                                      regs-addr))))),
                                data-in),
                                i-reg),
fv-if (we-data-out (v-threefix (cntl-state)),
      bv (f$core-alu (alu-c (v-threefix (cntl-state)),
                      v-threefix (a-reg),
                      v-threefix (b-reg),
                      alu-zero (v-threefix (cntl-state)),
                      alu-mpg (v-threefix (cntl-state)),
                      alu-op (v-threefix (cntl-state)),
                      make-tree (32)))),
                                data-out),
fv-if (we-addr-out (v-threefix (cntl-state)),
      f$dec-pass (dec-addr-out (v-threefix (cntl-state)),
                  f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                        a-immediate (v-threefix (i-reg)),
                                        f$read-regs (regs-address (v-threefix (cntl-state)),
                                                      list (regs-regs,
                                                            regs-we,
                                                            regs-data,
                                                            regs-addr))))),
                                addr-out),
f-buf (reset-),
f-or (strobe- (v-threefix (cntl-state)), dtack-),
f-if (we-hold- (v-threefix (cntl-state)), hold-, last-hold-),
fv-if (we-pc-reg (v-threefix (cntl-state)),
      pc-reg-in,
      last-pc-reg),
v-threefix (f$next-cntl-state (f-buf (last-reset-),
                                  f-buf (last-dtack-),
                                  f-buf (last-hold-),
                                  rw- (v-threefix (cntl-state))),

```

```

state (v-threefix (cntl-state)),
v-threefix (i-reg),
v-threefix (flags),
v-threefix (last-pc-reg),
regs-address (v-threefix (cntl-state))))))

```

THEOREM: chip-module\$value
(chip-module& (netlist)

```

^ machine-state-invariant (list (list (regs-regs,
                                     regs-we,
                                     regs-data,
                                     regs-addr),
                                     flags,
                                     a-reg,
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     addr-out,
                                     last-reset-,
                                     last-dtack-,
                                     last-hold-,
                                     last-pc-reg,
                                     cntl-state))
^ properp (pc-reg-in)
^ (length (pc-reg-in) = 4)
^ properp (data-in)
^ (length (data-in) = 32))
→ (dual-eval (0,
              'chip-module,
              cons (clk,
                    cons (ti,
                          cons (te,
                                cons (dtack-,
                                      cons (reset-,
                                            cons (hold-,
                                                  cons (disable-regfile-,
                                                        cons (test-regfile-,
                                                                append (pc-reg-in,
                                                                    data-in))))))))),
              list (list (regs-regs, regs-we, regs-data, regs-addr),
                        flags,
                        a-reg,
                        b-reg,
                        i-reg,

```

```

      data-out,
      addr-out,
      last-reset-,
      last-dtack-,
      last-hold-,
      last-pc-reg,
      cntl-state),
    netlist)
= cons (nth (3, v-threefix (last-pc-reg)),
      cons (nth (2,
        f$score-alu (alu-c (v-threefix (cntl-state)),
          v-threefix (a-reg),
          v-threefix (b-reg),
          alu-zero (v-threefix (cntl-state)),
          alu-mpg (v-threefix (cntl-state)),
          alu-op (v-threefix (cntl-state)),
          make-tree (32))),
        cons (hdack- (v-threefix (cntl-state)),
          cons (f-not (hdack- (v-threefix (cntl-state))),
            cons (f-buf (rw- (v-threefix (cntl-state))),
              cons (strobe- (v-threefix (cntl-state)),
                append5 (v-threefix (addr-out),
                  v-threefix (data-out),
                  v-threefix (flags),
                  state (v-threefix (cntl-state)),
                  subrange (v-threefix (i-reg),
                    28,
                    31))))))))))

```

```

(DEFN CHIP* NIL
  (CONS 'CHIP
    (CONS
      (CONS 'CLK
        (CONS 'TI
          (CONS 'TE
            (CONS 'DTACK-
              (CONS 'RESET-
                (CONS 'HOLD-
                  (CONS 'DISABLE-REGFILE-
                    (CONS 'TEST-REGFILE-
                      (APPEND (INDICES 'PC-REG-IN 0 4)
                        (INDICES 'DATA-BUS 0 32))))))))))))

```



```

(CONS
  (CONS 'PO
    (CONS 'TO
      (CONS 'TIMING
        (CONS 'HDACK-
          (CONS 'RW-
            (CONS 'STROBE-
              (APPEND (INDICES 'ADDR-OUT 0 32)
                (APPEND (INDICES 'DATA-BUS 0 32)
                  (APPEND (INDICES 'FLAGS 0 4)
                    (APPEND (INDICES 'CNTL-STATE 0 5)
                      (INDICES 'I-REG 28 4))))))))))
    (CONS
      (LIST
        (LIST 'BODY
          (CONS 'I-TO
            (CONS 'I-TIMING
              (CONS 'I-HDACK-
                (CONS 'I-EN-ADDR-OUT-
                  (CONS 'I-RW-
                    (CONS 'I-STROBE-
                      (APPEND (INDICES 'I-ADDR-OUT 0 32)
                        (APPEND (INDICES 'I-DATA-OUT 0 32)
                          (APPEND (INDICES 'I-FLAGS 0 4)
                            (APPEND (INDICES 'I-CNTL-STATE 0 5)
                              (INDICES 'I-I-REG 28 4))))))))))
                    'CHIP-MODULE
          (CONS 'I-CLK
            (CONS 'I-TI
              (CONS 'I-TE
                (CONS 'I-DTACK-
                  (CONS 'I-RESET-
                    (CONS 'I-HOLD-
                      (CONS 'I-DISABLE-REGFILE-
                        (CONS 'I-TEST-REGFILE-
                          (APPEND (INDICES 'I-PC-REG 0 4)
                            (INDICES 'I-DATA-IN 0 32))))))))))
          ' (PLUS-5 (B-TRUE-P) VDD-PARAMETRIC NIL)
          ' (CLOCK-PAD (I-CLK CLK-PO)
              TTL-CLK-INPUT
              (CLK B-TRUE-P))
          ' (TI-PAD (I-TI TI-PO)
              TTL-INPUT
              (TI B-TRUE-P))

```

```

'(TE-PAD (I-TE TE-PO)
          TTL-INPUT
          (TE TI-PO))
'(DTACK-PAD (I-DTACK- DTACK-PO)
             TTL-INPUT
             (DTACK- TE-PO))
'(RESET-PAD (I-RESET- RESET-PO)
             TTL-INPUT
             (RESET- DTACK-PO))
'(HOLD-PAD (I-HOLD- HOLD-PO)
            TTL-INPUT
            (HOLD- RESET-PO))
'(DISABLE-REGFILE-PAD (I-DISABLE-REGFILE- DISABLE-REGFILE-PO)
                       TTL-INPUT
                       (DISABLE-REGFILE- HOLD-PO))
'(TEST-REGFILE-PAD (I-TEST-REGFILE- TEST-REGFILE-PO)
                   TTL-INPUT
                   (TEST-REGFILE- DISABLE-REGFILE-PO))
(LIST 'DATA-BUS-PADS
      (CONS 'DATA-BUS-PO
             (APPEND (INDICES 'DATA-BUS 0 32)
                     (INDICES 'I-DATA-IN 0 32)))
      (INDEX 'TTL-BIDIRECT-PADS 32)
      (CONS 'I-RW-
             (CONS 'TEST-REGFILE-PO
                    (APPEND (INDICES 'DATA-BUS 0 32)
                            (INDICES 'I-DATA-OUT 0 32))))))
(LIST 'PC-REG-PADS
      (CONS 'PC-REG-PO
             (INDICES 'I-PC-REG 0 4))
      (INDEX 'TTL-INPUT-PADS 4)
      (CONS 'DATA-BUS-PO
             (INDICES 'PC-REG-IN 0 4)))
'(MONITOR (I-PO)
           PROCMON
           (PC-REG-PO CLK-PO B-TRUE-P B-TRUE-P))
'(PO-PAD (PO)
          TTL-OUTPUT-PARAMETRIC
          (I-PO))
'(TO-PAD (TO) TTL-OUTPUT (I-TO))
'(TIMING-PAD (TIMING)
             TTL-OUTPUT-FAST
             (I-TIMING))
'(HDACK-PAD (HDACK-))

```

```

        TTL-OUTPUT-FAST
        (I-HDACK-))
' (RW-PAD (RW-)
    TTL-TRI-OUTPUT-FAST
    (I-RW- I-EN-ADDR-OUT-))
' (STROBE-PAD (STROBE-)
    TTL-TRI-OUTPUT-FAST
    (I-STROBE- I-EN-ADDR-OUT-))
(LIST 'ADDR-OUT-PADS
    (INDICES 'ADDR-OUT 0 32)
    (INDEX 'TTL-TRI-OUTPUT-PADS 32)
    (CONS 'I-EN-ADDR-OUT-
        (INDICES 'I-ADDR-OUT 0 32)))
(LIST 'FLAGS-PADS
    (INDICES 'FLAGS 0 4)
    (INDEX 'TTL-OUTPUT-PADS 4)
    (INDICES 'I-FLAGS 0 4))
(LIST 'CNTL-STATE-PADS
    (INDICES 'CNTL-STATE 0 5)
    (INDEX 'TTL-OUTPUT-PADS 5)
    (INDICES 'I-CNTL-STATE 0 5))
(LIST 'I-REG-PADS
    (INDICES 'I-REG 28 4)
    (INDEX 'TTL-OUTPUT-PADS 4)
    (INDICES 'I-I-REG 28 4))
' (BODY))))))

```

```

(PROVE-LEMMA CHIP*$DESTRUCTURE
  (REWRITE)
  (AND
    (EQUAL (CAR (CHIP*)) 'CHIP)
    (EQUAL
      (CADR (CHIP*))
      (CONS 'CLK
        (CONS 'TI
          (CONS 'TE
            (CONS 'DTACK-
              (CONS 'RESET-
                (CONS 'HOLD-
                  (CONS 'DISABLE-REGFILE-
                    (CONS 'TEST-REGFILE-
                      (APPEND (INDICES 'PC-REG-IN 0 4)

```

```

(INDEXES 'DATA-BUS 0 32))))))))))
(EQUAL
  (CADDR (CHIP*))
  (CONS 'PO
    (CONS 'TO
      (CONS 'TIMING
        (CONS 'HDACK-
          (CONS 'RW-
            (CONS 'STROBE-
              (APPEND (INDEXES 'ADDR-OUT 0 32)
                (APPEND (INDEXES 'DATA-BUS 0 32)
                  (APPEND (INDEXES 'FLAGS 0 4)
                    (APPEND (INDEXES 'CNTL-STATE 0 5)
                      (INDEXES 'I-REG 28 4))))))))))))
(EQUAL
  (CADDR (CHIP*))
  (LIST
    (LIST 'BODY
      (CONS 'I-TO
        (CONS 'I-TIMING
          (CONS 'I-HDACK-
            (CONS 'I-EN-ADDR-OUT-
              (CONS 'I-RW-
                (CONS 'I-STROBE-
                  (APPEND (INDEXES 'I-ADDR-OUT 0 32)
                    (APPEND (INDEXES 'I-DATA-OUT 0 32)
                      (APPEND (INDEXES 'I-FLAGS 0 4)
                        (APPEND (INDEXES 'I-CNTL-STATE 0 5)
                          (INDEXES 'I-I-REG 28 4))))))))))))
    'CHIP-MODULE
    (CONS 'I-CLK
      (CONS 'I-TI
        (CONS 'I-TE
          (CONS 'I-DTACK-
            (CONS 'I-RESET-
              (CONS 'I-HOLD-
                (CONS 'I-DISABLE-REGFILE-
                  (CONS 'I-TEST-REGFILE-
                    (APPEND (INDEXES 'I-PC-REG 0 4)
                      (INDEXES 'I-DATA-IN 0 32))))))))))))
    ' (PLUS-5 (B-TRUE-P) VDD-PARAMETRIC NIL)
    ' (CLOCK-PAD (I-CLK CLK-PO)
      TTL-CLK-INPUT
      (CLK B-TRUE-P))

```

```

'(TI-PAD (I-TI TI-PO)
  TTL-INPUT
  (TI B-TRUE-P))
'(TE-PAD (I-TE TE-PO)
  TTL-INPUT
  (TE TI-PO))
'(DTACK-PAD (I-DTACK- DTACK-PO)
  TTL-INPUT
  (DTACK- TE-PO))
'(RESET-PAD (I-RESET- RESET-PO)
  TTL-INPUT
  (RESET- DTACK-PO))
'(HOLD-PAD (I-HOLD- HOLD-PO)
  TTL-INPUT
  (HOLD- RESET-PO))
'(DISABLE-REGFILE-PAD (I-DISABLE-REGFILE- DISABLE-REGFILE-PO)
  TTL-INPUT
  (DISABLE-REGFILE- HOLD-PO))
'(TEST-REGFILE-PAD (I-TEST-REGFILE- TEST-REGFILE-PO)
  TTL-INPUT
  (TEST-REGFILE- DISABLE-REGFILE-PO))
(LIST 'DATA-BUS-PADS
  (CONS 'DATA-BUS-PO
    (APPEND (INDICES 'DATA-BUS 0 32)
      (INDICES 'I-DATA-IN 0 32)))
  (INDEX 'TTL-BIDIRECT-PADS 32)
  (CONS 'I-RW-
    (CONS 'TEST-REGFILE-PO
      (APPEND (INDICES 'DATA-BUS 0 32)
        (INDICES 'I-DATA-OUT 0 32)))))
(LIST 'PC-REG-PADS
  (CONS 'PC-REG-PO
    (INDICES 'I-PC-REG 0 4))
  (INDEX 'TTL-INPUT-PADS 4)
  (CONS 'DATA-BUS-PO
    (INDICES 'PC-REG-IN 0 4)))
'(MONITOR (I-PO)
  PROCMON
  (PC-REG-PO CLK-PO B-TRUE-P B-TRUE-P))
'(PO-PAD (PO)
  TTL-OUTPUT-PARAMETRIC
  (I-PO))
'(TO-PAD (TO) TTL-OUTPUT (I-TO))
'(TIMING-PAD (TIMING)

```

```

        TTL-OUTPUT-FAST
        (I-TIMING))
' (HDACK-PAD (HDACK-)
    TTL-OUTPUT-FAST
    (I-HDACK-))
' (RW-PAD (RW-)
    TTL-TRI-OUTPUT-FAST
    (I-RW- I-EN-ADDR-OUT-))
' (STROBE-PAD (STROBE-)
    TTL-TRI-OUTPUT-FAST
    (I-STROBE- I-EN-ADDR-OUT-))
(LIST 'ADDR-OUT-PADS
    (INDICES 'ADDR-OUT 0 32)
    (INDEX 'TTL-TRI-OUTPUT-PADS 32)
    (CONS 'I-EN-ADDR-OUT-
        (INDICES 'I-ADDR-OUT 0 32)))
(LIST 'FLAGS-PADS
    (INDICES 'FLAGS 0 4)
    (INDEX 'TTL-OUTPUT-PADS 4)
    (INDICES 'I-FLAGS 0 4))
(LIST 'CNTL-STATE-PADS
    (INDICES 'CNTL-STATE 0 5)
    (INDEX 'TTL-OUTPUT-PADS 5)
    (INDICES 'I-CNTL-STATE 0 5))
(LIST 'I-REG-PADS
    (INDICES 'I-REG 28 4)
    (INDEX 'TTL-OUTPUT-PADS 4)
    (INDICES 'I-I-REG 28 4)))
(EQUAL (CADDR (CHIP*)) 'BODY))

```

EVENT: Enable chip*; name this event 'chip*-off'.

EVENT: Enable chip*\$destructure; name this event 'chip*\$destructure-off'.

EVENT: Enable *1*chip*; name this event 'g*1*chip*-off'.

DEFINITION:

chip& (*netlist*)

```

= ((lookup-module ('chip, netlist) = CHIP*)
  ^ vdd-parametric& (delete-module ('chip, netlist))
  ^ ttl-clk-input& (delete-module ('chip, netlist))
  ^ ttl-input& (delete-module ('chip, netlist))
  ^ ttl-input-pads& (delete-module ('chip, netlist), 4)

```

$\wedge$ chip-module& (delete-module ('chip, netlist))
 $\wedge$ ttl-bidirect-pads& (delete-module ('chip, netlist), 32)
 $\wedge$ ttl-output-parametric& (delete-module ('chip, netlist))
 $\wedge$ ttl-output& (delete-module ('chip, netlist))
 $\wedge$ ttl-output-fast& (delete-module ('chip, netlist))
 $\wedge$ ttl-tri-output-fast& (delete-module ('chip, netlist))
 $\wedge$ ttl-tri-output-pads& (delete-module ('chip, netlist), 32)
 $\wedge$ ttl-output-pads& (delete-module ('chip, netlist), 4)
 $\wedge$ ttl-output-pads& (delete-module ('chip, netlist), 5)
 $\wedge$ procmon& (delete-module ('chip, netlist))

EVENT: Enable chip&; name this event 'chip&-off'.

DEFINITION:

CHIP\$NETLIST

= cons (CHIP*,
 (((ttl-bidirect-pads\$netlist (32) $\cup$ CHIP-MODULE\$NETLIST)
 $\cup$ (TTL-INPUT\$NETLIST $\cup$ ttl-input-pads\$netlist (4)))
 $\cup$ ((TTL-CLK-INPUT\$NETLIST $\cup$ VDD-PARAMETRIC\$NETLIST)
 $\cup$ (TTL-OUTPUT-PARAMETRIC\$NETLIST
 $\cup$ TTL-OUTPUT\$NETLIST)))
 $\cup$ (((TTL-OUTPUT-FAST\$NETLIST
 $\cup$ TTL-TRI-OUTPUT-FAST\$NETLIST)
 $\cup$ (ttl-tri-output-pads\$netlist (32)
 $\cup$ ttl-output-pads\$netlist (4)))
 $\cup$ (ttl-output-pads\$netlist (5) $\cup$ PROCMON\$NETLIST)))

THEOREM: chip\$state

(chip& (netlist)

$\wedge$ machine-state-invariant (list (list (regs-regs,
 regs-we,
 regs-data,
 regs-addr),
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,
 last-reset-,
 last-dtack-,
 last-hold-,
 last-pc-reg,
 cntl-state))

```

^ (te = f)
^ (disable-regfile- = t)
^ (test-regfile- = t)
^ properp (pc-reg-in)
^ (length (pc-reg-in) = 4)
^ properp (data-in)
^ (length (data-in) = 32))
→ (dual-eval (2,
    'chip,
    cons (clk,
        cons (ti,
            cons (te,
                cons (dtack-,
                    cons (reset-,
                        cons (hold-,
                            cons (disable-regfile-,
                                cons (test-regfile-,
                                    append (pc-reg-in,
                                        data-in))))))))),
        list (list (regs-regs, regs-we, regs-data, regs-addr),
            flags,
            a-reg,
            b-reg,
            i-reg,
            data-out,
            addr-out,
            last-reset-,
            last-dtack-,
            last-hold-,
            last-pc-reg,
            cntl-state),
            netlist)
    = list (f$write-regs (we-regs (v-threefix (cntl-state)),
        regs-address (v-threefix (cntl-state)),
        list (regs-regs, regs-we, regs-data, regs-addr),
        bv (f$core-alu (alu-c (v-threefix (cntl-state)),
            v-threefix (a-reg),
            v-threefix (b-reg),
            alu-zero (v-threefix (cntl-state)),
            alu-mpg (v-threefix (cntl-state)),
            alu-op (v-threefix (cntl-state)),
            make-tree (32))))),
        f$update-flags (flags,
            we-flags (v-threefix (cntl-state))),

```



```

f$core-alu (alu-c (v-threefix (cntl-state)),
            v-threefix (a-reg),
            v-threefix (b-reg),
            alu-zero (v-threefix (cntl-state)),
            alu-mpg (v-threefix (cntl-state)),
            alu-op (v-threefix (cntl-state)),
            make-tree (32))),
fv-if (we-a-reg (v-threefix (cntl-state)),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                    f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                   a-immediate (v-threefix (i-reg)),
                                   f$read-regs (regs-address (v-threefix (cntl-state)),
                                                list (regs-regs,
                                                    regs-we,
                                                    regs-data,
                                                    regs-addr))),
            v-wire (data-in,
                    vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
                    v-threefix (data-out))),
      a-reg),
fv-if (we-b-reg (v-threefix (cntl-state)),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                    f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                   a-immediate (v-threefix (i-reg)),
                                   f$read-regs (regs-address (v-threefix (cntl-state)),
                                                list (regs-regs,
                                                    regs-we,
                                                    regs-data,
                                                    regs-addr))),
            v-wire (data-in,
                    vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
                    v-threefix (data-out))),
      b-reg),
fv-if (we-i-reg (v-threefix (cntl-state)),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                    f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                   a-immediate (v-threefix (i-reg)),
                                   f$read-regs (regs-address (v-threefix (cntl-state)),
                                                list (regs-regs,
                                                    regs-we,
                                                    regs-data,
                                                    regs-addr))),
            v-wire (data-in,
                    vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
                    v-threefix (data-out))),
      i-reg),

```

```

                                regs-addr))),
v-wire (data-in,
        vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
        v-threefix (data-out))),
i-reg),
fv-if (we-data-out (v-threefix (cntl-state)),
      bv (f$core-alu (alu-c (v-threefix (cntl-state)),
        v-threefix (a-reg),
        v-threefix (b-reg),
        alu-zero (v-threefix (cntl-state)),
        alu-mpg (v-threefix (cntl-state)),
        alu-op (v-threefix (cntl-state)),
        make-tree (32))),
      data-out),
fv-if (we-addr-out (v-threefix (cntl-state)),
      f$dec-pass (dec-addr-out (v-threefix (cntl-state)),
        f$extend-immediate (select-immediate (v-threefix (cntl-state)),
          a-immediate (v-threefix (i-reg)),
          f$read-regs (regs-address (v-threefix (cntl-state)),
            list (regs-regs,
                  regs-we,
                  regs-data,
                  regs-addr))))),
      addr-out),
f-buf (reset-),
f-or (strobe- (v-threefix (cntl-state)), f-buf (dtack-)),
f-if (we-hold- (v-threefix (cntl-state)),
      f-buf (hold-),
      last-hold-),
fv-if (we-pc-reg (v-threefix (cntl-state)),
      pc-reg-in,
      last-pc-reg),
v-threefix (f$next-cntl-state (f-buf (last-reset-),
                                f-buf (last-dtack-),
                                f-buf (last-hold-),
                                rw- (v-threefix (cntl-state)),
                                state (v-threefix (cntl-state)),
                                v-threefix (i-reg),
                                v-threefix (flags),
                                v-threefix (last-pc-reg),
                                regs-address (v-threefix (cntl-state))))))

```

THEOREM: chip\$value
(chip& (netlist)

```

^ machine-state-invariant (list (list (regs-regs,
                                     regs-we,
                                     regs-data,
                                     regs-addr),
                                     flags,
                                     a-reg,
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     addr-out,
                                     last-reset-,
                                     last-dtack-,
                                     last-hold-,
                                     last-pc-reg,
                                     cntl-state))
^ properp (pc-reg-in)
^ (length (pc-reg-in) = 4)
^ properp (data-in)
^ (length (data-in) = 32))
→ (cdr (dual-eval (0,
                  'chip,
                  cons (clk,
                        cons (ti,
                              cons (te,
                                    cons (dtack-,
                                          cons (reset-,
                                                cons (hold-,
                                                      cons (disable-regfile-,
                                                            cons (test-regfile-,
                                                                append (pc-reg-in,
                                                                    data-in))))))))),
                  list (list (regs-regs, regs-we, regs-data, regs-addr),
                          flags,
                          a-reg,
                          b-reg,
                          i-reg,
                          data-out,
                          addr-out,
                          last-reset-,
                          last-dtack-,
                          last-hold-,
                          last-pc-reg,
                          cntl-state),
                          netlist))

```

```

= cons (f-buf (nth (3, v-threefix (last-pc-reg))),
        cons (f-buf (nth (2,
                          f$score-alu (alu-c (v-threefix (cntl-state)),
                                      v-threefix (a-reg),
                                      v-threefix (b-reg),
                                      alu-zero (v-threefix (cntl-state)),
                                      alu-mpg (v-threefix (cntl-state)),
                                      alu-op (v-threefix (cntl-state)),
                                      make-tree (32))),
              cons (f-buf (hdack- (v-threefix (cntl-state))),
                    cons (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                              f-buf (rw- (v-threefix (cntl-state)))),
                    cons (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                              strobe- (v-threefix (cntl-state))),
                    append5 (vft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                     v-threefix (addr-out)),
                             vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))),
                                     v-threefix (data-out)),
                             v-threefix (flags),
                             v-threefix (state (v-threefix (cntl-state))),
                             v-threefix (subrange (v-threefix (i-reg),
                                                         28,
                                                         31))))))))))

```

```

(DEFN CHIP-SYSTEM* NIL
  (CONS 'CHIP-SYSTEM
    (CONS
      (CONS 'CLK
        (CONS 'TI
          (CONS 'TE
            (CONS 'RESET-
              (CONS 'HOLD-
                (CONS 'DISABLE-REGFILE-
                  (CONS 'TEST-REGFILE-
                    (INDICES 'PC-REG 0 4))))))))))
    (CONS NIL
      (CONS
        (LIST
          (LIST 'FM9001
            (CONS 'PO
              (CONS 'TO
                (CONS 'TIMING

```

```

(CONS 'HDACK-
(CONS 'RW-
(CONS 'STROBE-
  (APPEND (INDICES 'ADDR-OUT 0 32)
    (APPEND (INDICES 'FM9001-DATA 0 32)
      (APPEND (INDICES 'FLAGS 0 4)
        (APPEND (INDICES 'CNTL-STATE 0 5)
          (INDICES 'I-REG 28 4))))))))))
'CHIP
(CONS 'CLK
(CONS 'TI
(CONS 'TE
(CONS 'DTACK-
(CONS 'RESET-
  (CONS 'HOLD-
    (CONS 'DISABLE-REGFILE-
      (CONS 'TEST-REGFILE-
        (APPEND (INDICES 'PC-REG 0 4)
          (INDICES 'DATA-BUS 0 32))))))))))
'(PULLUP-RW- (RW-BUS) PULLUP (RW-))
'(PULLUP-STROBE- (STROBE-BUS)
  PULLUP
  (STROBE-))
(LIST 'ADDRESS
  (INDICES 'ADDR-BUS 0 32)
  (INDEX 'V-PULLUP 32)
  (INDICES 'ADDR-OUT 0 32))
(LIST 'MEM
  (CONS 'DTACK-
    (INDICES 'MEM-DATA 0 32))
  'MEM-32X32
  (CONS 'RW-BUS
    (CONS 'STROBE-BUS
      (APPEND (INDICES 'ADDR-BUS 0 32)
        (INDICES 'DATA-BUS 0 32))))))
(LIST 'DATA-WIRE
  (INDICES 'DATA-WIRE 0 32)
  (INDEX 'V-WIRE 32)
  (APPEND (INDICES 'FM9001-DATA 0 32)
    (INDICES 'MEM-DATA 0 32)))
(LIST 'DATA
  (INDICES 'DATA-BUS 0 32)
  (INDEX 'V-PULLUP 32)
  (INDICES 'DATA-WIRE 0 32)))

```

```
'((FM9001 MEM))))))
```

```
(PROVE-LEMMA CHIP-SYSTEM*$DESTRUCTURE
(REWRITE)
(AND
(EQUAL (CAR (CHIP-SYSTEM*))
'CHIP-SYSTEM)
(EQUAL
(CADR (CHIP-SYSTEM*))
(CONS 'CLK
(CONS 'TI
(CONS 'TE
(CONS 'RESET-
(CONS 'HOLD-
(CONS 'DISABLE-REGFILE-
(CONS 'TEST-REGFILE-
(INDICES 'PC-REG 0 4))))))))))
(EQUAL (CADDR (CHIP-SYSTEM*)) NIL)
(EQUAL
(CADDDR (CHIP-SYSTEM*))
(LIST
(LIST 'FM9001
(CONS 'PO
(CONS 'TO
(CONS 'TIMING
(CONS 'HDACK-
(CONS 'RW-
(CONS 'STROBE-
(APPEND (INDICES 'ADDR-OUT 0 32)
(APPEND (INDICES 'FM9001-DATA 0 32)
(APPEND (INDICES 'FLAGS 0 4)
(APPEND (INDICES 'CNTL-STATE 0 5)
(INDICES 'I-REG 28 4))))))))))
'CHIP
(CONS 'CLK
(CONS 'TI
(CONS 'TE
(CONS 'DTACK-
(CONS 'RESET-
(CONS 'HOLD-
(CONS 'DISABLE-REGFILE-
(CONS 'TEST-REGFILE-
```

```

                                (APPEND (INDICES 'PC-REG 0 4)
                                (INDICES 'DATA-BUS 0 32)))))))))
' (PULLUP-RW- (RW-BUS) PULLUP (RW-))
' (PULLUP-STROBE- (STROBE-BUS)
  PULLUP
  (STROBE-))
(LIST 'ADDRESS
  (INDICES 'ADDR-BUS 0 32)
  (INDEX 'V-PULLUP 32)
  (INDICES 'ADDR-OUT 0 32))
(LIST 'MEM
  (CONS 'DTACK-
    (INDICES 'MEM-DATA 0 32))
  'MEM-32X32
  (CONS 'RW-BUS
    (CONS 'STROBE-BUS
      (APPEND (INDICES 'ADDR-BUS 0 32)
        (INDICES 'DATA-BUS 0 32)))))
(LIST 'DATA-WIRE
  (INDICES 'DATA-WIRE 0 32)
  (INDEX 'V-WIRE 32)
  (APPEND (INDICES 'FM9001-DATA 0 32)
    (INDICES 'MEM-DATA 0 32)))
(LIST 'DATA
  (INDICES 'DATA-BUS 0 32)
  (INDEX 'V-PULLUP 32)
  (INDICES 'DATA-WIRE 0 32)))
(EQUAL (CADDDDR (CHIP-SYSTEM*))
  '(FM9001 MEM)))

```

EVENT: Enable chip-system*; name this event 'chip-system*-off'.

EVENT: Enable chip-system*\$destructure; name this event 'chip-system*\$destructure-off'.

EVENT: Enable *1*chip-system*; name this event 'g*1*chip-system*-off'.

DEFINITION:

chip-system& (*netlist*)

```

= ((lookup-module ('chip-system, netlist) = CHIP-SYSTEM*)
  ^ chip& (delete-module ('chip-system, netlist))
  ^ pullup& (delete-module ('chip-system, netlist))
  ^ v-pullup& (delete-module ('chip-system, netlist), 32)

```

$\wedge$ mem-32x32& (delete-module ('chip-system, *netlist*))
 $\wedge$ v-wire& (delete-module ('chip-system, *netlist*), 32))

EVENT: Enable chip-system&; name this event 'chip-system&-off'.

DEFINITION:

CHIP-SYSTEM\$NETLIST
 = cons (CHIP-SYSTEM*,
 PULLUP\$NETLIST
 $\cup$ (v-pullup\$netlist (32)
 $\cup$ (MEM-32X32\$NETLIST
 $\cup$ (v-wire\$netlist (32) $\cup$ CHIP\$NETLIST))))

DEFINITION:

memory-state-invariant (*mem-state*)
 = (memory-properp (32, 32, car (*mem-state*))
 $\wedge$ properp (caddrdddr (*mem-state*))
 $\wedge$ properp (caddrdddr (*mem-state*))
 $\wedge$ (length (caddrdddr (*mem-state*)) = 32))

DEFINITION:

chip-system-invariant (*state*)
 = (machine-state-invariant (car (*state*))
 $\wedge$ memory-state-invariant (cadr (*state*)))

THEOREM: equal-memory-value-for-chip-system\$state

memory-value (*state*, *strobe*, *rw-*, *address*, collect-value (*args*, *alist*))
 = memory-value (*state*, *strobe*, *rw-*, *address*, make-list (length (*args*), X))

THEOREM: chip-system\$state-help

(chip-system& (*netlist*)
 $\wedge$ chip-system-invariant (list (list (list (*regs-regs*,
 regs-we,
 regs-data,
 regs-addr),
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,
 last-reset,
 last-dtack,
 last-hold-,


```

                                last-pc-reg,
                                cntl-state),
                                mem-state))
^  (te = f)
^  (disable-regfile- = t)
^  (test-regfile- = t)
^  properp (pc-reg-in)
^  (length (pc-reg-in) = 4))
→  (dual-eval (2,
              'chip-system,
              cons (clk,
                    cons (ti,
                          cons (te,
                                cons (reset-,
                                      cons (hold-,
                                            cons (disable-regfile-,
                                                  cons (test-regfile-, pc-reg-in)))))),
                    list (list (list (regs-regs, regs-we, regs-data, regs-addr),
                                    flags,
                                    a-reg,
                                    b-reg,
                                    i-reg,
                                    data-out,
                                    addr-out,
                                    last-reset-,
                                    last-dtack-,
                                    last-hold-,
                                    last-pc-reg,
                                    cntl-state),
                                mem-state),
                          netlist)
              =  list (list (f$write-regs (we-regs (v-threefix (cntl-state))),
                              regs-address (v-threefix (cntl-state))),
                          list (regs-regs,
                              regs-we,
                              regs-data,
                              regs-addr),
                              bv (f$core-alu (alu-c (v-threefix (cntl-state))),
                                  v-threefix (a-reg),
                                  v-threefix (b-reg),
                                  alu-zero (v-threefix (cntl-state)),
                                  alu-mpg (v-threefix (cntl-state)),
                                  alu-op (v-threefix (cntl-state)),
                                  make-tree (32))))),

```

```

f$update-flags (flags,
               we-flags (v-threefix (cntl-state)),
               f$core-alu (alu-c (v-threefix (cntl-state)),
                           v-threefix (a-reg),
                           v-threefix (b-reg),
                           alu-zero (v-threefix (cntl-state)),
                           alu-mpg (v-threefix (cntl-state)),
                           alu-op (v-threefix (cntl-state)),
                           make-tree (32))),
fv-if (we-a-reg (v-threefix (cntl-state)),
      fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                    f-not (last-dtack-)),
            f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                a-immediate (v-threefix (i-reg)),
                                f$read-regs (regs-address (v-threefix (cntl-state)),
                                              list (regs-regs,
                                                  regs-we,
                                                  regs-data,
                                                  regs-addr))),
                                v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))
                                                                 v-threefix (data-out))),
                                                                cdr (memory-value (mem-state,
                                                                 f-pullup (ft-buf (f-buf (hd
                                                                 strobe- (
                                                                 f-pullup (ft-buf (f-buf (hd
                                                                 f-buf (rw
                                                                 v-pullup (vft-buf (f-buf (h
                                                                 v-three
                                                                 make-list (32,
                                                                 x))))))),
                                                                vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state)))))
                                                                v-threefix (data-out))))),
                                a-reg)),
      fv-if (we-b-reg (v-threefix (cntl-state)),
            fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                          f-not (last-dtack-)),
                f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                        a-immediate (v-threefix (i-reg)),
                                        f$read-regs (regs-address (v-threefix (cntl-state)),
                                                    list (regs-regs,
                                                        regs-we,
                                                        regs-data,
                                                        regs-addr))),
                                        v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))

```

```

                                v-threefix (data-out)),
                                cdr (memory-value (mem-state,
                                f-pullup (ft-buf (f-buf (hd
                                strobe- (
                                f-pullup (ft-buf (f-buf (hd
                                f-buf (rw
                                v-pullup (vft-buf (f-buf (h
                                v-three
                                make-list (32,
                                x))))),
                                vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
                                v-threefix (data-out))))),
                                b-reg),
                                fv-if (we-i-reg (v-threefix (cntl-state)),
                                fv-if (f-nand (data-in-select (v-threefix (cntl-state)),
                                f-not (last-dtack-)),
                                f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                a-immediate (v-threefix (i-reg)),
                                f$read-regs (regs-address (v-threefix (cntl-state)),
                                list (regs-regs,
                                regs-we,
                                regs-data,
                                regs-addr))),
                                v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state)
                                v-threefix (data-out)),
                                cdr (memory-value (mem-state,
                                f-pullup (ft-buf (f-buf (hd
                                strobe- (
                                f-pullup (ft-buf (f-buf (hd
                                f-buf (rw
                                v-pullup (vft-buf (f-buf (h
                                v-three
                                make-list (32,
                                x))))),
                                vft-buf (f-not (f-buf (rw- (v-threefix (cntl-state))))),
                                v-threefix (data-out))))),
                                i-reg),
                                fv-if (we-data-out (v-threefix (cntl-state)),
                                bv (f$score-alu (alu-c (v-threefix (cntl-state)),
                                v-threefix (a-reg),
                                v-threefix (b-reg),
                                alu-zero (v-threefix (cntl-state)),
                                alu-mpg (v-threefix (cntl-state)),
                                alu-op (v-threefix (cntl-state)),

```

```

                                make-tree (32))),
                                data-out),
fv-if (we-addr-out (v-threefix (cntl-state)),
      f$dec-pass (dec-addr-out (v-threefix (cntl-state)),
                  f$extend-immediate (select-immediate (v-threefix (cntl-state)),
                                      a-immediate (v-threefix (i-reg)),
                                      f$read-regs (regs-address (v-threefix (cntl-state)),
                                                  list (regs-regs,
                                                      regs-we,
                                                      regs-data,
                                                      regs-addr))))),
                                addr-out),
f-buf (reset-),
f-or (strobe- (v-threefix (cntl-state)),
     f-buf (car (memory-value (mem-state,
                              f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                              strobe- (v-threefix (cntl-state)))),
                              f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                              f-buf (rw- (v-threefix (cntl-state)))),
                              v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                              v-threefix (addr-out))),
                              make-list (32, x))))),
f-if (we-hold- (v-threefix (cntl-state)),
     f-buf (hold-),
     last-hold-),
fv-if (we-pc-reg (v-threefix (cntl-state)),
     pc-reg-in,
     last-pc-reg),
v-threefix (f$next-cntl-state (f-buf (last-reset-),
                                   f-buf (last-dtack-),
                                   f-buf (last-hold-),
                                   rw- (v-threefix (cntl-state)),
                                   state (v-threefix (cntl-state)),
                                   v-threefix (i-reg),
                                   v-threefix (flags),
                                   v-threefix (last-pc-reg),
                                   regs-address (v-threefix (cntl-state))))),
next-memory-state (mem-state,
                  f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                  strobe- (v-threefix (cntl-state)))),
                  f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                  f-buf (rw- (v-threefix (cntl-state))))),
                  v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                  v-threefix (addr-out))),

```

EVENT: Enable chip-system\$state-help; name this event ‘chip-system\$state-help-off’.

$$\begin{aligned} &\wedge (te = \mathbf{f}) \\ &\wedge (disable-regfile- = \mathbf{t}) \\ &\wedge (test-regfile- = \mathbf{t}) \\ &\wedge \text{properp}(pc-reg-in) \\ &\wedge (\text{length}(pc-reg-in) = 4)) \\ \rightarrow &(\text{dual-eval}(2, \\ &\quad \text{'chip-system',} \\ &\quad \text{cons}(clk, \\ &\quad \quad \text{cons}(ti, \\ &\quad \quad \quad \text{cons}(te, \\ &\quad \quad \quad \quad \text{cons}(reset-, \end{aligned}$$

$$\begin{aligned}
& \text{cons}(\text{hold-}, \\
& \quad \text{cons}(\text{disable-regfile-}, \\
& \quad \quad \text{cons}(\text{test-regfile-}, \text{pc-reg-in}))))), \\
& \text{list}(\text{list}(\text{list}(\text{regs-regs}, \text{regs-we}, \text{regs-data}, \text{regs-addr}), \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{b-reg}, \\
& \quad \text{i-reg}, \\
& \quad \text{data-out}, \\
& \quad \text{addr-out}, \\
& \quad \text{last-reset-}, \\
& \quad \text{last-dtack-}, \\
& \quad \text{last-hold-}, \\
& \quad \text{last-pc-reg}, \\
& \quad \text{cntl-state}), \\
& \quad \text{mem-state}), \\
& \quad \text{netlist}) \\
= & \text{fm9001-next-state}(\text{list}(\text{list}(\text{list}(\text{regs-regs}, \\
& \quad \text{regs-we}, \\
& \quad \text{regs-data}, \\
& \quad \text{regs-addr}), \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{b-reg}, \\
& \quad \text{i-reg}, \\
& \quad \text{data-out}, \\
& \quad \text{addr-out}, \\
& \quad \text{last-reset-}, \\
& \quad \text{last-dtack-}, \\
& \quad \text{last-hold-}, \\
& \quad \text{last-pc-reg}, \\
& \quad \text{cntl-state}), \\
& \quad \text{mem-state}), \\
& \quad \text{cons}(\text{reset-}, \text{cons}(\text{hold-}, \text{pc-reg-in}))))
\end{aligned}$$

EVENT: Enable chip-system\$state; name this event ‘chip-system\$state-off’.

THEOREM: next-memory-state-preserves-memory-invariant
(memory-state-invariant (*mem-state*)

$\wedge$ properp(*data*)

$\wedge$ (length(*data*) = 32)

$\wedge$ (length(*address*) = 32))

$\rightarrow$ memory-state-invariant (next-memory-state (*mem-state*,
strobe-,

rw-,
address,
data))

THEOREM: memory-properp-dual-port-ram-state-crock

(memory-properp (4, 32, *regs*)
 $\wedge$ (length (*a-address*) = 4)
 $\wedge$ (length (*b-address*) = 4)
 $\wedge$ properp (*data*)
 $\wedge$ (length (*data*) = 32))
 $\rightarrow$ memory-properp (4,
32,
dual-port-ram-state (32,
4,
append (*a-address*,
append (*b-address*,
cons (*we*, *data*))),
regs))

THEOREM: all-ramp-mem-dual-port-ram-state-crock

(all-ramp-mem (4, *regs*)
 $\wedge$ (length (*a-address*) = 4)
 $\wedge$ (length (*b-address*) = 4)
 $\wedge$ properp (*data*)
 $\wedge$ (length (*data*) = 32))
 $\rightarrow$ all-ramp-mem (4,
dual-port-ram-state (32,
4,
append (*a-address*,
append (*b-address*,
cons (*we*, *data*))),
regs))

THEOREM: fm9001-preserves-chip-system-invariant

(chip-system-invariant (*state*)
 $\wedge$ properp (pc-reg-input (*inputs*))
 $\wedge$ (length (pc-reg-input (*inputs*)) = 4))
 $\rightarrow$ chip-system-invariant (fm9001-next-state (*state*, *inputs*))

DEFINITION:

chip-system-input-invariant (*inputs*)
= ((caddr (*inputs*) = **f**)
 $\wedge$ (cadddddr (*inputs*) = **t**)
 $\wedge$ (caddddddr (*inputs*) = **t**)
 $\wedge$ properp (cdddddrr (*inputs*))
 $\wedge$ (length (cdddddrr (*inputs*)) = 4))

THEOREM: rewrite-chip-system-input-invariant
chip-system-input-invariant (*inputs*)
= ((*inputs* = cons (car (*inputs*),
 cons (cadr (*inputs*),
 cons (caddr (*inputs*),
 cons (cadddr (*inputs*),
 cons (caddddr (*inputs*),
 cons (cadddddr (*inputs*),
 cons (caddddddd (*inputs*),
 caddddddd (*inputs*))))))))))
 $\wedge$ (caddr (*inputs*) = **f**)
 $\wedge$ (cadddddr (*inputs*) = **t**)
 $\wedge$ (caddddddd (*inputs*) = **t**)
 $\wedge$ properp (caddddddd (*inputs*))
 $\wedge$ (length (caddddddd (*inputs*)) = 4))

DEFINITION:
chip-system-operating-inputs-p (*inputs*, *n*)
= **if** *n* $\simeq$ 0 **then** **t**
 else chip-system-input-invariant (car (*inputs*))
 $\wedge$ chip-system-operating-inputs-p (cdr (*inputs*), *n* - 1) **endif**

EVENT: Enable chip-system-operating-inputs-p; name this event ‘chip-system-operating-inputs-p-off’.

THEOREM: open-chip-system-operating-inputs-p
((*n* $\simeq$ 0) $\rightarrow$ (chip-system-operating-inputs-p (*inputs*, *n*) = **t**))
 $\wedge$ ((*n* $\not\simeq$ 0)
 $\rightarrow$ (chip-system-operating-inputs-p (*inputs*, *n*)
 = (chip-system-input-invariant (car (*inputs*))
 $\wedge$ chip-system-operating-inputs-p (cdr (*inputs*),
 n - 1))))

DEFINITION:
map-up-1-input (*inputs*)
= cons (cadddr (*inputs*), cons (caddddr (*inputs*), caddddddd (*inputs*)))

DEFINITION:
map-up-inputs (*inputs*)
= **if** *inputs* $\simeq$ **nil** **then** **nil**
 else cons (map-up-1-input (car (*inputs*)),
 map-up-inputs (cdr (*inputs*))) **endif**

EVENT: Enable map-up-inputs; name this event ‘map-up-inputs-off’.

THEOREM: open-map-up-inputs
 $((inputs \simeq \mathbf{nil}) \rightarrow (map-up-inputs(inputs) = \mathbf{nil}))$
 $\wedge$ (listp($inputs$)
 $\rightarrow$ (map-up-inputs($inputs$)
 $=$ cons(map-up-1-input(car($inputs$)),
map-up-inputs(cdr($inputs$))))))

THEOREM: chip-system=fm9001\$step
(chip-system&($netlist$)
 $\wedge$ fm9001-state-structure($state$)
 $\wedge$ chip-system-invariant($state$)
 $\wedge$ chip-system-input-invariant($inputs$)
 $\rightarrow$ (dual-eval(2, 'chip-system, $inputs$, $state$, $netlist$)
 $=$ fm9001-next-state($state$, map-up-1-input($inputs$)))

EVENT: Enable chip-system=fm9001\$step; name this event 'chip-system=fm9001\$step-off'.

DEFINITION:
chip-system=run-fm9001\$induction($state$, $inputs$, $fm9001-inputs$, n)
 $=$ **if** $n \simeq 0$ **then** **t**
else chip-system=run-fm9001\$induction(fm9001-next-state($state$,
car($fm9001-inputs$)),
cdr($inputs$),
cdr($fm9001-inputs$),
 $n - 1$) **endif**

THEOREM: chip-system=run-fm9001
(chip-system&($netlist$)
 $\wedge$ fm9001-state-structure($state$)
 $\wedge$ chip-system-invariant($state$)
 $\wedge$ chip-system-operating-inputs-p($inputs$, n)
 $\wedge$ ($fm9001-inputs = map-up-inputs(inputs)$))
 $\rightarrow$ (simulate-dual-eval-2('chip-system, $inputs$, $state$, $netlist$, n)
 $=$ run-fm9001($state$, $fm9001-inputs$, n))

DEFINITION:
run-inputs-p($inputs$, n)
 $=$ **if** $n \simeq 0$ **then** **t**
else (reset-input(car($inputs$)) = **t**)
 $\wedge$ properp(pc-reg-input(car($inputs$)))
 $\wedge$ (length(pc-reg-input(car($inputs$))) = 4)
 $\wedge$ run-inputs-p(cdr($inputs$), $n - 1$) **endif**

EVENT: Enable run-inputs-p; name this event 'run-inputs-p-off'.

THEOREM: open-run-inputs-p
 $((n \simeq 0) \rightarrow (\text{run-inputs-p } (inputs, n) = \mathbf{t}))$
 $\wedge ((n \not\simeq 0)$
 $\rightarrow (\text{run-inputs-p } (inputs, n)$
 $= ((\text{reset-input } (\text{car } (inputs)) = \mathbf{t})$
 $\wedge \text{properp } (\text{pc-reg-input } (\text{car } (inputs)))$
 $\wedge (\text{length } (\text{pc-reg-input } (\text{car } (inputs))) = 4)$
 $\wedge \text{run-inputs-p } (\text{cdr } (inputs), n - 1)))$

THEOREM: run-inputs-p-plus
 $\text{run-inputs-p } (inputs, n + m)$
 $= (\text{run-inputs-p } (inputs, n) \wedge \text{run-inputs-p } (\text{nthcdr } (n, inputs), m))$

THEOREM: run-inputs-p-1
 $(\text{run-inputs-p } (inputs, n) \wedge (n \not\simeq 0)) \rightarrow \text{run-inputs-p } (inputs, 1)$

THEOREM: open-run-inputs-p-add1
 $\text{run-inputs-p } (inputs, 1 + n)$
 $= ((\text{reset-input } (\text{car } (inputs)) = \mathbf{t})$
 $\wedge \text{properp } (\text{pc-reg-input } (\text{car } (inputs)))$
 $\wedge (\text{length } (\text{pc-reg-input } (\text{car } (inputs))) = 4)$
 $\wedge \text{run-inputs-p } (\text{cdr } (inputs), n))$

DEFINITION:
 $\text{regfile-okp } (regs)$
 $= ((\text{length } (regs) = 4)$
 $\wedge \text{memory-okp } (4, 32, \text{car } (regs))$
 $\wedge \text{all-ramp-mem } (4, \text{car } (regs))$
 $\wedge \text{boolp } (\text{cadr } (regs))$
 $\wedge (\text{bvp } (\text{caddr } (regs)) \wedge (\text{length } (\text{caddr } (regs)) = 32))$
 $\wedge \text{bvp } (\text{cadddr } (regs))$
 $\wedge (\text{length } (\text{cadddr } (regs)) = 4))$

THEOREM: rewrite-fm9001-next-state-for-step-lemmas
 $(\text{bvp } (flags)$
 $\wedge \text{bvp } (a\text{-reg})$
 $\wedge \text{bvp } (b\text{-reg})$
 $\wedge \text{bvp } (i\text{-reg})$
 $\wedge \text{bvp } (data\text{-out})$
 $\wedge \text{bvp } (addr\text{-out})$
 $\wedge \text{boolp } (last\text{-reset-})$
 $\wedge \text{boolp } (last\text{-dtack-})$
 $\wedge \text{boolp } (last\text{-hold-})$
 $\wedge \text{bvp } (last\text{-pc-reg})$
 $\wedge \text{bvp } (cntl\text{-state}))$

```

→ (fm9001-next-state (list (list (regs,
                                   flags,
                                   a-reg,
                                   b-reg,
                                   i-reg,
                                   data-out,
                                   addr-out,
                                   last-reset,
                                   last-dtack,
                                   last-hold,
                                   last-pc-reg,
                                   cntl-state),
                               mem-state),
    external-inputs)
=  list (list (f$write-regs (we-regs (cntl-state),
                                     regs-address (cntl-state),
                                     regs,
                                     bv (f$core-alu (alu-c (cntl-state),
                                                         a-reg,
                                                         b-reg,
                                                         alu-zero (cntl-state),
                                                         alu-mpg (cntl-state),
                                                         alu-op (cntl-state),
                                                         make-tree (32))))),
              f$update-flags (flags,
                              we-flags (cntl-state),
                              f$core-alu (alu-c (cntl-state),
                                                  a-reg,
                                                  b-reg,
                                                  alu-zero (cntl-state),
                                                  alu-mpg (cntl-state),
                                                  alu-op (cntl-state),
                                                  make-tree (32))),
              fv-if (we-a-reg (cntl-state),
                    fv-if (f-nand (data-in-select (cntl-state),
                                                  f-not (last-dtack-)),
                          f$extend-immediate (select-immediate (cntl-state),
                                                              a-immediate (i-reg),
                                                              f$read-regs (regs-address (cntl-state),
                                                              regs)),
                    v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (cntl-state))),
                                                              data-out),
                                                              cdr (memory-value (mem-state,
                                                              f-pullup (ft-buf (f-buf (hd

```

```

                                strobe- (
f-pullup (ft-buf (f-buf (hd
                                f-buf (rw
v-pullup (vft-buf (f-buf (h
                                addr-o
make-list (32,
                                x))))),
vft-buf (f-not (f-buf (rw- (cntl-state))),
data-out))))),
a-reg),
fv-if (we-b-reg (cntl-state),
fv-if (f-nand (data-in-select (cntl-state),
f-not (last-dtack-)),
f$extend-immediate (select-immediate (cntl-state),
a-immediate (i-reg),
f$read-regs (regs-address (cntl-state),
regs)),
v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (cntl-state))),
data-out),
cdr (memory-value (mem-state,
f-pullup (ft-buf (f-buf (hd
                                strobe- (
f-pullup (ft-buf (f-buf (hd
                                f-buf (rw
v-pullup (vft-buf (f-buf (h
                                addr-o
make-list (32,
                                x))))),
vft-buf (f-not (f-buf (rw- (cntl-state))),
data-out))))),
b-reg),
fv-if (we-i-reg (cntl-state),
fv-if (f-nand (data-in-select (cntl-state),
f-not (last-dtack-)),
f$extend-immediate (select-immediate (cntl-state),
a-immediate (i-reg),
f$read-regs (regs-address (cntl-state),
regs)),
v-threefix (v-wire (v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (cntl-state))),
data-out),
cdr (memory-value (mem-state,
f-pullup (ft-buf (f-buf (hd
                                strobe- (
f-pullup (ft-buf (f-buf (hd

```

```

f-buf (rw-
v-pullup (vft-buf (f-buf (h
addr-out
make-list (32,
x))))),
vft-buf (f-not (f-buf (rw- (cntl-state))),
data-out))),
i-reg),
fv-if (we-data-out (cntl-state),
bv (f$core-alu (alu-c (cntl-state),
a-reg,
b-reg,
alu-zero (cntl-state),
alu-mpg (cntl-state),
alu-op (cntl-state),
make-tree (32))),
data-out),
fv-if (we-addr-out (cntl-state),
f$dec-pass (dec-addr-out (cntl-state),
f$extend-immediate (select-immediate (cntl-state),
a-immediate (i-reg),
f$read-regs (regs-address (cntl-state),
regs))),
addr-out),
f-buf (reset-input (external-inputs)),
f-or (strobe- (cntl-state),
f-buf (car (memory-value (mem-state,
f-pullup (ft-buf (f-buf (hdack- (cntl-state)),
strobe- (cntl-state))),
f-pullup (ft-buf (f-buf (hdack- (cntl-state)),
f-buf (rw- (cntl-state)))),
v-pullup (vft-buf (f-buf (hdack- (cntl-state)),
addr-out))),
make-list (32, x))))),
f-if (we-hold- (cntl-state),
f-buf (hold-input (external-inputs)),
last-hold-),
fv-if (we-pc-reg (cntl-state),
pc-reg-input (external-inputs),
last-pc-reg),
v-threefix (f$next-cntl-state (last-reset-,
last-dtack-,
last-hold-,
rw- (cntl-state),

```

```

state (cntl-state),
i-reg,
flags,
last-pc-reg,
regs-address (cntl-state))),
next-memory-state (mem-state,
  f-pullup (ft-buf (f-buf (hdack- (cntl-state))),
    strobe- (cntl-state))),
  f-pullup (ft-buf (f-buf (hdack- (cntl-state))),
    f-buf (rw- (cntl-state)))),
  v-pullup (vft-buf (f-buf (hdack- (cntl-state))),
    addr-out)),
  v-pullup (v-wire (vft-buf (f-not (f-buf (rw- (cntl-state))),
    data-out)),
    cdr (memory-value (mem-state,
      f-pullup (ft-buf (f-buf (hdack- (cntl-state))),
        strobe- (cntl-state))),
      f-pullup (ft-buf (f-buf (hdack- (cntl-state))),
        f-buf (rw- (cntl-state))),
      v-pullup (vft-buf (f-buf (hdack- (cntl-state))),
        addr-out))),
      make-list (32,
        x))))))

```

THEOREM: fetch0\$step0

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,

```

```

      flags,
      a-reg,
      b-reg,
      i-reg,
      data-out,
      addr-out,
      reset-,
      dtack-,
      hold-,
      pc-reg,
      cv_fetch0 (t,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  last-pc-reg)),
    list (mem,
          0,
          clock,
          mem-count,
          mem-dtack,
          t,
          last-address,
          last-data)),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     read-regs (last-pc-reg, regs),
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     read-regs (last-pc-reg, regs),
                                     t,
                                     t,
                                     f-buf (f-buf (hold-input (car (inputs))))),
                                     pc-reg,
                                     cv_fetch1 (t,
                                                  last-pc-reg,
                                                  i-reg,
                                                  flags,
                                                  pc-reg)),
                                list (mem,
                                      0,
                                      clock,

```

```

0,
t,
t,
addr-out,
v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: fetch0\$step1

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_fetch0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg))),
list (mem,

```



```

1,
clock,
mem-count,
mem-dtack,
t,
last-address,
last-data)),
inputs,
n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
flags,
read-regs (last-pc-reg, regs),
b-reg,
i-reg,
data-out,
read-regs (last-pc-reg, regs),
t,
t,
f-buf (f-buf (hold-input (car (inputs))))),
pc-reg,
cv_fetch1 (t,
last-pc-reg,
i-reg,
flags,
pc-reg)),
list (mem,
0,
clock,
0,
t,
t,
addr-out,
v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: fetch0\$step2

```

(cv-hyps (f, last-regs-address, last-i-reg, last-flags, last-pc-reg)
^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
^ memory-okp (32, 32, mem)
^ regfile-okp (regs)
^ bvp (a-reg)
^ (length (a-reg) = 32)
^ bvp (b-reg)

```

```

^ (length (b-reg) = 32)
^ bvp (addr-out)
^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ (n ≠ 0)
^ run-inputs-p (inputs, 1)
^ boolp (dtack-)
^ boolp (hold-)
^ (reset- = t)
^ (mem-count ≅ 0))
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_fetch0 (f,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg)),
                           list (mem,
                                2,
                                clock,
                                mem-count,
                                mem-dtack,
                                f,
                                addr-out,
                                data-out))),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     read-regs (last-pc-reg, regs),
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     read-regs (last-pc-reg, regs),

```

```

      t,
      t,
      f-buf (f-buf (hold-input (car (inputs)))),
      pc-reg,
      cv_fetch1 (t,
                  last-pc-reg,
                  i-reg,
                  flags,
                  pc-reg)),
      list (write-mem (addr-out, mem, data-out),
            0,
            clock,
            0,
            t,
            f,
            addr-out,
            data-out)),
      cdr (inputs),
      n - 1))

```

THEOREM: fetch1\$step

```

(cv-hyps (last-rw-, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ boolp (last-rw-)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,

```

```

    data-out,
    addr-out,
    reset-,
    dtack-,
    hold-,
    pc-reg,
    cv_fetch1 (last-rw-,
                last-regs-address,
                last-i-reg,
                last-flags,
                last-pc-reg)),
  list (mem,
        0,
        clock,
        mem-count,
        mem-dtack,
        last-rw-,
        last-address,
        last-data)),
  inputs,
  n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
    flags,
    a-reg,
    b-reg,
    i-reg,
    data-out,
    addr-out,
    t,
    t,
    hold-,
    pc-reg,
    if* (hold-,
        cv_fetch2 (t,
                    last-pc-reg,
                    i-reg,
                    flags,
                    pc-reg),
        cv_hold0 (t,
                   last-pc-reg,
                   i-reg,
                   flags,
                   pc-reg))),
    list (mem,

```

```

0,
clock,
0,
t,
t,
addr-out,
v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: fetch2\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_fetch2 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,

```

```

                                last-pc-reg)),
      list (mem,
            0,
            clock,
            mem-count,
            mem-dtack,
            t,
            addr-out,
            last-data)),
      inputs,
      n)
=  if car (clock)  $\simeq$  0
    then run-fm9001 (list (list (write-regs (t,
                                              last-pc-reg,
                                              regs,
                                              v-inc (a-reg)),
                                              flags,
                                              a-reg,
                                              b-reg,
                                              i-reg,
                                              data-out,
                                              addr-out,
                                              t,
                                              f,
                                              hold-,
                                              pc-reg,
                                              cv_fetch3 (t,
                                                         last-pc-reg,
                                                         i-reg,
                                                         flags,
                                                         pc-reg))),
                            list (mem,
                                  1,
                                  cdr (clock),
                                  0,
                                  t,
                                  t,
                                  addr-out,
                                  v-threefix (last-data))),
                            cdr (inputs),
                            n - 1)
    else run-fm9001 (list (list (write-regs (t,
                                              last-pc-reg,
                                              regs,

```

```

                                v-inc(a-reg)),
                                flags,
                                a-reg,
                                b-reg,
                                i-reg,
                                data-out,
                                addr-out,
                                t,
                                t,
                                hold-,
                                pc-reg,
                                cv_fetch3(t,
                                last-pc-reg,
                                i-reg,
                                flags,
                                pc-reg)),
                                list(mem,
                                1,
                                cdr(clock),
                                car(clock) - 1,
                                f,
                                t,
                                addr-out,
                                v-threefix(last-data))),
                                cdr(inputs),
                                n - 1) endif)

```

THEOREM: fetch3\$step0
 (cv-hyps(**t**, *last-regs-address*, *last-i-reg*, *last-flags*, *last-pc-reg*)
 ^ cv-hyps(**t**, *pc-reg*, *i-reg*, *flags*, *pc-reg*)
 ^ memory-okp(32, 32, *mem*)
 ^ regfile-okp(*regs*)
 ^ bvp(*a-reg*)
 ^ (length(*a-reg*) = 32)
 ^ bvp(*b-reg*)
 ^ (length(*b-reg*) = 32)
 ^ bvp(*addr-out*)
 ^ (length(*addr-out*) = 32)
 ^ bvp(*data-out*)
 ^ (length(*data-out*) = 32)
 ^ (*n* ≠ 0)
 ^ run-inputs-p(*inputs*, 1)
 ^ boolp(**f**)
 ^ boolp(*hold-*)

$$\begin{aligned}
& \wedge \quad (\text{reset} = \mathbf{t}) \\
& \wedge \quad (\text{mem-count} \simeq 0) \\
\rightarrow & \quad (\text{run-fm9001} (\text{list} (\text{list} (\text{regs}, \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{b-reg}, \\
& \quad \text{i-reg}, \\
& \quad \text{data-out}, \\
& \quad \text{addr-out}, \\
& \quad \text{reset-}, \\
& \quad \mathbf{f}, \\
& \quad \text{hold-}, \\
& \quad \text{pc-reg}, \\
& \quad \text{cv\_fetch3} (\mathbf{t}, \\
& \quad \quad \text{last-regs-address}, \\
& \quad \quad \text{last-i-reg}, \\
& \quad \quad \text{last-flags}, \\
& \quad \quad \text{last-pc-reg})), \\
& \quad \text{list} (\text{mem}, 1, \text{clock}, \text{mem-count}, \mathbf{t}, \mathbf{t}, \text{addr-out}, \text{last-data})), \\
& \quad \text{inputs}, \\
& \quad n) \\
= & \quad \text{run-fm9001} (\text{list} (\text{list} (\text{write-regs} (\mathbf{f}, 0, \text{regs}, 0), \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{b-reg}, \\
& \quad \text{read-mem} (\text{addr-out}, \text{mem}), \\
& \quad \text{data-out}, \\
& \quad \text{addr-out}, \\
& \quad \mathbf{t}, \\
& \quad \mathbf{f}, \\
& \quad \text{hold-}, \\
& \quad \text{pc-reg}, \\
& \quad \text{cv\_decode} (\mathbf{t}, \\
& \quad \quad \text{last-pc-reg}, \\
& \quad \quad \text{i-reg}, \\
& \quad \quad \text{flags}, \\
& \quad \quad \text{pc-reg})), \\
& \quad \text{list} (\text{mem}, \\
& \quad \quad 1, \\
& \quad \quad \text{clock}, \\
& \quad \quad 0, \\
& \quad \quad \mathbf{t}, \\
& \quad \quad \mathbf{t}, \\
& \quad \quad \text{addr-out},
\end{aligned}$$

$$\text{v-threefix}(\text{last-data})),$$

$$\text{cdr}(\text{inputs}),$$

$$n - 1))$$

THEOREM: fetch3\$step1

$$(\text{cv-hyps}(\mathbf{t}, \text{last-regs-address}, \text{last-i-reg}, \text{last-flags}, \text{last-pc-reg})$$

$$\wedge \text{cv-hyps}(\mathbf{t}, \text{pc-reg}, \text{i-reg}, \text{flags}, \text{pc-reg})$$

$$\wedge \text{memory-okp}(32, 32, \text{mem})$$

$$\wedge \text{regfile-okp}(\text{regs})$$

$$\wedge \text{bvp}(\text{a-reg})$$

$$\wedge (\text{length}(\text{a-reg}) = 32)$$

$$\wedge \text{bvp}(\text{b-reg})$$

$$\wedge (\text{length}(\text{b-reg}) = 32)$$

$$\wedge \text{bvp}(\text{addr-out})$$

$$\wedge (\text{length}(\text{addr-out}) = 32)$$

$$\wedge \text{bvp}(\text{data-out})$$

$$\wedge (\text{length}(\text{data-out}) = 32)$$

$$\wedge (n \neq 0)$$

$$\wedge \text{run-inputs-p}(\text{inputs}, 1)$$

$$\wedge \text{boolp}(\mathbf{t})$$

$$\wedge \text{boolp}(\text{hold-})$$

$$\wedge (\text{reset-} = \mathbf{t})$$

$$\wedge \mathbf{nil})$$

$$\rightarrow (\text{run-fm9001}(\text{list}(\text{list}(\text{regs},$$

$$\text{flags},$$

$$\text{a-reg},$$

$$\text{b-reg},$$

$$\text{i-reg},$$

$$\text{data-out},$$

$$\text{addr-out},$$

$$\text{reset-},$$

$$\mathbf{t},$$

$$\text{hold-},$$

$$\text{pc-reg},$$

$$\text{cv\_fetch3}(\mathbf{t},$$

$$\text{last-regs-address},$$

$$\text{last-i-reg},$$

$$\text{last-flags},$$

$$\text{last-pc-reg})),$$

$$\text{list}(\text{mem}, 1, \text{clock}, \text{mem-count}, \mathbf{f}, \mathbf{t}, \text{addr-out}, \text{last-data})),$$

$$\text{inputs},$$

$$n)$$

$$= \text{if } \text{mem-count} \simeq 0$$

$$\text{then } \text{run-fm9001}(\text{list}(\text{list}(\text{write-regs}(\mathbf{f}, 0, \text{regs}, 0),$$

```

        flags,
        a-reg,
        b-reg,
        read-regs (last-pc-reg, regs),
        data-out,
        addr-out,
        t,
        f,
        hold-,
        pc-reg,
        cv_fetch3 (t,
                    last-pc-reg,
                    i-reg,
                    flags,
                    pc-reg)),
    list (mem,
          1,
          clock,
          0,
          t,
          t,
          addr-out,
          v-threefix (last-data))),
    cdr (inputs),
    n - 1)
else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
        flags,
        a-reg,
        b-reg,
        read-regs (last-pc-reg, regs),
        data-out,
        addr-out,
        t,
        t,
        hold-,
        pc-reg,
        cv_fetch3 (t,
                    last-pc-reg,
                    i-reg,
                    flags,
                    pc-reg)),
    list (mem,
          1,
          clock,

```

```

                                mem-count - 1,
                                f,
                                t,
                                addr-out,
                                v-threefix(last-data))),
cdr(inputs),
n - 1) endif)

```

THEOREM: decode\$step

```

(cv-hyps(t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps(t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp(32, 32, mem)
  ^ regfile-okp(regs)
  ^ bvp(a-reg)
  ^ (length(a-reg) = 32)
  ^ bvp(b-reg)
  ^ (length(b-reg) = 32)
  ^ bvp(addr-out)
  ^ (length(addr-out) = 32)
  ^ bvp(data-out)
  ^ (length(data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p(inputs, 1)
  ^ boolp(dtack-)
  ^ boolp(hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001(list(list(regs,
                        flags,
                        a-reg,
                        b-reg,
                        i-reg,
                        data-out,
                        addr-out,
                        reset-,
                        dtack-,
                        hold-,
                        pc-reg,
                        cv_decode(t,
                                last-regs-address,
                                last-i-reg,
                                last-flags,
                                last-pc-reg))),
list(mem,

```

```

1,
clock,
mem-count,
mem-dtack,
t,
last-address,
last-data)),
inputs,
n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
t,
t,
hold-,
pc-reg,
if* (or* (store-resultp (store-cc (i-reg),
flags),
set-some-flags (set-flags (i-reg))),
if* (or* (a-immediate-p (i-reg),
reg-direct-p (mode-a (i-reg))),
if* (or* (reg-direct-p (mode-b (i-reg)),
unary-op-code-p (op-code (i-reg))),
cv_rega (t,
last-pc-reg,
i-reg,
flags,
pc-reg),
cv_readb0 (t,
last-pc-reg,
i-reg,
flags,
pc-reg)),
cv_reada0 (t,
last-pc-reg,
i-reg,
flags,
pc-reg)),
if* (and* (not* (a-immediate-p (i-reg)),
or* (pre-dec-p (mode-a (i-reg))),

```

```

                                post-inc-p (mode-a (i-reg))),
cv_sefa0 (t,
          last-pc-reg,
          i-reg,
          flags,
          pc-reg),
if* (or* (pre-dec-p (mode-b (i-reg)),
                  post-inc-p (mode-b (i-reg))),
      cv_sefb0 (t,
                last-pc-reg,
                i-reg,
                flags,
                pc-reg),
      cv_fetch0 (t,
                  last-pc-reg,
                  i-reg,
                  flags,
                  pc-reg))))),
list (mem,
      0,
      clock,
      0,
      t,
      t,
      addr-out,
      v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: rega\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ∧ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ∧ memory-okp (32, 32, mem)
  ∧ regfile-okp (regs)
  ∧ bvp (a-reg)
  ∧ (length (a-reg) = 32)
  ∧ bvp (b-reg)
  ∧ (length (b-reg) = 32)
  ∧ bvp (addr-out)
  ∧ (length (addr-out) = 32)
  ∧ bvp (data-out)
  ∧ (length (data-out) = 32)
  ∧ (n ≠ 0)
  ∧ run-inputs-p (inputs, 1)

```

```

^ boolp (dtack-)
^ boolp (hold-)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv-rega (t,
                                   last-regs-address,
                                   last-i-reg,
                                   last-flags,
                                   last-pc-reg)),
                      list (mem,
                            0,
                            clock,
                            mem-count,
                            mem-dtack,
                            t,
                            last-address,
                            last-data))),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     if* (a-immediate-p (last-i-reg),
                                             sign-extend (a-immediate (i-reg), 32),
                                             read-regs (rn-a (last-i-reg), regs)),
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     addr-out,
                                     t,
                                     t,
                                     hold-,
                                     pc-reg,
                                     if* (reg-direct-p (mode-b (i-reg))),

```

```

if* (unary-op-code-p (op-code (i-reg)),
     cv_update (t,
                 rn-a (last-i-reg),
                 i-reg,
                 flags,
                 pc-reg),
     cv_regb (t,
              rn-a (last-i-reg),
              i-reg,
              flags,
              pc-reg)),
if* (store-resultp (store-cc (i-reg),
                             flags),
     cv_write0 (t,
                rn-a (last-i-reg),
                i-reg,
                flags,
                pc-reg),
     cv_update (t,
                 rn-a (last-i-reg),
                 i-reg,
                 flags,
                 pc-reg))))),
list (mem,
      0,
      clock,
      0,
      t,
      t,
      addr-out,
      v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: regb\$step
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 $\wedge$ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 $\wedge$ memory-okp (32, 32, mem)
 $\wedge$ regfile-okp (regs)
 $\wedge$ bvp (a-reg)
 $\wedge$ (length (a-reg) = 32)
 $\wedge$ bvp (b-reg)
 $\wedge$ (length (b-reg) = 32)
 $\wedge$ bvp (addr-out)

```

^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ (n ≠ 0)
^ run-inputs-p (inputs, 1)
^ boolp (dtack-)
^ boolp (hold-)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_regb (t,
                                   last-regs-address,
                                   last-i-reg,
                                   last-flags,
                                   last-pc-reg)),
                        list (mem,
                              0,
                              clock,
                              mem-count,
                              mem-dtack,
                              t,
                              last-address,
                              last-data))),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                   flags,
                                   a-reg,
                                   read-regs (rn-b (last-i-reg), regs),
                                   i-reg,
                                   data-out,
                                   addr-out,
                                   t,
                                   t),

```



```

      hold-,
      pc-reg,
      cv_update (t,
                  rn-b (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg)),
      list (mem,
            0,
            clock,
            0,
            t,
            t,
            addr-out,
            v-threefix (last-data))),
      cdr (inputs),
      n - 1))

```

THEOREM: update\$step
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 $\wedge$ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 $\wedge$ memory-okp (32, 32, mem)
 $\wedge$ regfile-okp (regs)
 $\wedge$ bvp (a-reg)
 $\wedge$ (length (a-reg) = 32)
 $\wedge$ bvp (b-reg)
 $\wedge$ (length (b-reg) = 32)
 $\wedge$ bvp (addr-out)
 $\wedge$ (length (addr-out) = 32)
 $\wedge$ bvp (data-out)
 $\wedge$ (length (data-out) = 32)
 $\wedge$ (n $\neq$ 0)
 $\wedge$ run-inputs-p (inputs, 1)
 $\wedge$ boolp (dtack-)
 $\wedge$ boolp (hold-)
 $\wedge$ (reset- = t)
 $\wedge$ ((mem-state = 0) $\vee$ (mem-state = 1)))
 $\rightarrow$ (run-fm9001 (list (list (regs,
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,

```

reset-,
dtack-,
hold-,
pc-reg,
cv_update (t,
            last-regs-address,
            last-i-reg,
            last-flags,
            last-pc-reg)),
list (mem,
      mem-state,
      clock,
      mem-count,
      mem-dtack,
      t,
      last-address,
      last-data)),
inputs,
n)
= run-fm9001 (list (list (write-regs (store-resultp (store-cc (last-i-reg),
                                                         last-flags),
                                                         rn-b (last-i-reg),
                                                         regs,
                                                         bv (v-alu (c-flag (last-flags),
                                                         a-reg,
                                                         b-reg,
                                                         op-code (last-i-reg))))),
                        update-flags (flags,
                                     set-flags (last-i-reg),
                                     v-alu (c-flag (last-flags),
                                     a-reg,
                                     b-reg,
                                     op-code (last-i-reg))),
                        a-reg,
                        read-regs (rn-b (last-i-reg), regs),
                        i-reg,
                        data-out,
                        addr-out,
                        t,
                        t,
                        hold-,
                        pc-reg,
                        if* (and* (or* (pre-dec-p (mode-b (i-reg)),
                                           post-inc-p (mode-b (i-reg))),

```

```

        unary-op-code-p (op-code (i-reg))),
cv_sefb1 (t,
        rn-b (last-i-reg),
        i-reg,
        flags,
        pc-reg),
cv_fetch0 (t,
        rn-b (last-i-reg),
        i-reg,
        flags,
        pc-reg))),
list (mem,
      0,
      clock,
      0,
      t,
      t,
      addr-out,
      v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: reada0\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,

```

```

      b-reg,
      i-reg,
      data-out,
      addr-out,
      reset-,
      dtack-,
      hold-,
      pc-reg,
      cv_reada0 (t,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  last-pc-reg)),
    list (mem,
          0,
          clock,
          mem-count,
          mem-dtack,
          t,
          last-address,
          last-data)),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     read-regs (rn-a (last-i-reg), regs),
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     if* (pre-dec-p (mode-a (last-i-reg)),
                                           v-dec (read-regs (rn-a (last-i-reg),
                                                             regs))),
                                     read-regs (rn-a (last-i-reg), regs)),
                                     t,
                                     t,
                                     hold-,
                                     pc-reg,
                                     cv_reada1 (t,
                                                  rn-a (last-i-reg),
                                                  i-reg,
                                                  flags,
                                                  pc-reg))),
               list (mem,
                     0,

```

```

        clock,
        0,
        t,
        t,
        addr-out,
        v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: reada1\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_reada1 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg))),

```

```

list (mem,
      0,
      clock,
      mem-count,
      mem-dtack,
      t,
      last-address,
      last-data)),
,
n)
= run-fm9001 (list (list (write-regs (and* (not* (a-immediate-p (last-i-reg)),
                                         or* (pre-dec-p (mode-a (last-i-reg)),
                                         post-inc-p (mode-a (last-i-reg)))),
rn-a (last-i-reg),
regs,
bv (core-alu (if* (pre-dec-p (mode-a (last-i-reg)),
                           carry-in-help (cons (c-flag (last-flags),
                                                    cons (f,
                                                           ALU-DEC-OP))),
                           carry-in-help (cons (c-flag (last-flags),
                                                    cons (f,
                                                           ALU-INC-OP))))),
a-reg,
b-reg,
f,
if* (pre-dec-p (mode-a (last-i-reg)),
    mpg (cons (f,
               cons (f,
                     ALU-DEC-OP))),
    mpg (cons (f,
               cons (f,
                     ALU-INC-OP))))),
if* (pre-dec-p (mode-a (last-i-reg)),
    ALU-DEC-OP,
    ALU-INC-OP),
make-tree (32))))),
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
t,
t,

```

```

      hold-,
      pc-reg,
      cv_reada2 (t,
                  rn-a (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg)),
      list (mem,
            0,
            clock,
            0,
            t,
            t,
            addr-out,
            v-threefix (last-data))),
      cdr (inputs),
      n - 1))

```

THEOREM: reada2\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,

```

```

reset-,
dtack-,
hold-,
pc-reg,
cv_reada2 (t,
            last-regs-address,
            last-i-reg,
            last-flags,
            last-pc-reg)),
list (mem,
      0,
      clock,
      mem-count,
      mem-dtack,
      t,
      addr-out,
      last-data)),
inputs,
n)
= if car (clock)  $\simeq$  0
  then run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                           flags,
                                           a-reg,
                                           read-regs (rn-b (last-i-reg), regs),
                                           i-reg,
                                           data-out,
                                           addr-out,
                                           t,
                                           f,
                                           hold-,
                                           pc-reg,
                                           cv_reada3 (t,
                                                         rn-b (last-i-reg),
                                                         i-reg,
                                                         flags,
                                                         pc-reg)),
list (mem,
      1,
      cdr (clock),
      0,
      t,
      t,
      addr-out,
      v-threefix (last-data))),

```



```

      cdr (inputs),
      n - 1)
else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
      flags,
      a-reg,
      read-regs (rn-b (last-i-reg),
      regs),
      i-reg,
      data-out,
      addr-out,
      t,
      t,
      hold-,
      pc-reg,
      cv_reada3 (t,
      rn-b (last-i-reg),
      i-reg,
      flags,
      pc-reg))),
      list (mem,
      1,
      cdr (clock),
      car (clock) - 1,
      f,
      t,
      addr-out,
      v-threefix (last-data))),
      cdr (inputs),
      n - 1) endif)

```

THEOREM: reada3\$step0
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 ^ memory-okp (32, 32, mem)
 ^ regfile-okp (regs)
 ^ bvp (a-reg)
 ^ (length (a-reg) = 32)
 ^ bvp (b-reg)
 ^ (length (b-reg) = 32)
 ^ bvp (addr-out)
 ^ (length (addr-out) = 32)
 ^ bvp (data-out)
 ^ (length (data-out) = 32)
 ^ (n $\neq$ 0)

```

 $\wedge$  run-inputs-p (inputs, 1)
 $\wedge$  boolp (f)
 $\wedge$  boolp (hold-)
 $\wedge$  (reset- = t)
 $\wedge$  (mem-count  $\simeq$  0))
 $\rightarrow$  (run-fm9001 (list (list (regs,
                             flags,
                             a-reg,
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,
                             reset-,
                             f,
                             hold-,
                             pc-reg,
                             cv_reada3 (t,
                                         last-regs-address,
                                         last-i-reg,
                                         last-flags,
                                         last-pc-reg))),
                  list (mem, 1, clock, mem-count, t, t, addr-out, last-data)),
      inputs,
      n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     read-mem (addr-out, mem),
                                     b-reg,
                                     i-reg,
                                     data-out,
                                     addr-out,
                                     t,
                                     f,
                                     hold-,
                                     pc-reg,
                                     if* (reg-direct-p (mode-b (i-reg))),
                                     cv_update (t,
                                                last-pc-reg,
                                                i-reg,
                                                flags,
                                                pc-reg),
                                     if* (unary-op-code-p (op-code (i-reg))),
                                     if* (store-resultp (store-cc (i-reg),
                                                                    flags))),

```

```

cv_write0(t,
          last-pc-reg,
          i-reg,
          flags,
          pc-reg),
cv_update(t,
          last-pc-reg,
          i-reg,
          flags,
          pc-reg)),
cv_readb0(t,
          last-pc-reg,
          i-reg,
          flags,
          pc-reg))))),
list(mem,
      1,
      clock,
      0,
      t,
      t,
      addr-out,
      v-threefix(last-data))),
cdr(inputs),
n - 1))

```

THEOREM: reada3\$step1
 (cv-hyps(**t**, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 ^ cv-hyps(**t**, pc-reg, i-reg, flags, pc-reg)
 ^ memory-okp(32, 32, mem)
 ^ regfile-okp(regs)
 ^ bvp(a-reg)
 ^ (length(a-reg) = 32)
 ^ bvp(b-reg)
 ^ (length(b-reg) = 32)
 ^ bvp(addr-out)
 ^ (length(addr-out) = 32)
 ^ bvp(data-out)
 ^ (length(data-out) = 32)
 ^ (n ≠ 0)
 ^ run-inputs-p(inputs, 1)
 ^ boolp(**t**)
 ^ boolp(hold-)
 ^ (reset- = **t**))

```

 $\wedge$  nil)
 $\rightarrow$  (run-fm9001 (list (list (regs,
                             flags,
                             a-reg,
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,
                             reset-,
                             t,
                             hold-,
                             pc-reg,
                             cv_reada3 (t,
                                         last-regs-address,
                                         last-i-reg,
                                         last-flags,
                                         last-pc-reg))),
                    list (mem, 1, clock, mem-count, f, t, addr-out, last-data)),
    inputs,
    n)
= if mem-count  $\simeq$  0
  then run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                           flags,
                                           read-regs (last-pc-reg, regs),
                                           b-reg,
                                           i-reg,
                                           data-out,
                                           addr-out,
                                           t,
                                           f,
                                           hold-,
                                           pc-reg,
                                           cv_reada3 (t,
                                                       last-pc-reg,
                                                       i-reg,
                                                       flags,
                                                       pc-reg))),
                          list (mem,
                                1,
                                clock,
                                0,
                                t,
                                t,
                                addr-out,

```

```

                                v-threefix (last-data))),
                                cdr (inputs),
                                n - 1)
else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                flags,
                                read-regs (last-pc-reg, regs),
                                b-reg,
                                i-reg,
                                data-out,
                                addr-out,
                                t,
                                t,
                                hold-,
                                pc-reg,
                                cv_reada3 (t,
                                last-pc-reg,
                                i-reg,
                                flags,
                                pc-reg))),
                                list (mem,
                                1,
                                clock,
                                mem-count - 1,
                                f,
                                t,
                                addr-out,
                                v-threefix (last-data))),
                                cdr (inputs),
                                n - 1) endif)

```

THEOREM: readb0\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)

```

$$\begin{aligned}
& \wedge \text{run-inputs-p}(\text{inputs}, 1) \\
& \wedge \text{boolp}(\text{dtack-}) \\
& \wedge \text{boolp}(\text{hold-}) \\
& \wedge (\text{reset-} = \mathbf{t}) \\
& \wedge ((\text{mem-state} = 0) \vee (\text{mem-state} = 1)) \\
\rightarrow & (\text{run-fm9001}(\text{list}(\text{list}(\text{regs}, \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{b-reg}, \\
& \quad \text{i-reg}, \\
& \quad \text{data-out}, \\
& \quad \text{addr-out}, \\
& \quad \text{reset-}, \\
& \quad \text{dtack-}, \\
& \quad \text{hold-}, \\
& \quad \text{pc-reg}, \\
& \quad \text{cv\_readb0}(\mathbf{t}, \\
& \quad \quad \text{last-regs-address}, \\
& \quad \quad \text{last-i-reg}, \\
& \quad \quad \text{last-flags}, \\
& \quad \quad \text{last-pc-reg})), \\
& \text{list}(\text{mem}, \\
& \quad \text{mem-state}, \\
& \quad \text{clock}, \\
& \quad \text{mem-count}, \\
& \quad \text{mem-dtack}, \\
& \quad \mathbf{t}, \\
& \quad \text{last-address}, \\
& \quad \text{last-data})), \\
& \text{inputs}, \\
& n) \\
= & \text{run-fm9001}(\text{list}(\text{list}(\text{write-regs}(\mathbf{f}, 0, \text{regs}, 0), \\
& \quad \text{flags}, \\
& \quad \text{a-reg}, \\
& \quad \text{read-regs}(\text{rn-b}(\text{last-i-reg}), \text{regs}), \\
& \quad \text{i-reg}, \\
& \quad \text{data-out}, \\
& \quad \text{if*}(\text{pre-dec-p}(\text{mode-b}(\text{last-i-reg})), \\
& \quad \quad \text{v-dec}(\text{read-regs}(\text{rn-b}(\text{last-i-reg}), \\
& \quad \quad \quad \text{regs})), \\
& \quad \quad \text{read-regs}(\text{rn-b}(\text{last-i-reg}), \text{regs})), \\
& \quad \mathbf{t}, \\
& \quad \mathbf{t}, \\
& \quad \text{hold-},
\end{aligned}$$

```

      pc-reg,
      cv_readb1 (t,
                  rn-b (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg)),
      list (mem,
            0,
            clock,
            0,
            t,
            t,
            addr-out,
            v-threefix (last-data))),
      cdr (inputs),
      n - 1))

```

THEOREM: readb1\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,

```

```

    dtack-,
    hold-,
    pc-reg,
    cv_readb1 (t,
                last-regs-address,
                last-i-reg,
                last-flags,
                last-pc-reg)),
  list (mem,
        0,
        clock,
        mem-count,
        mem-dtack,
        t,
        last-address,
        last-data)),
  inputs,
  n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
    flags,
    if* (or* (a-immediate-p (last-i-reg),
               reg-direct-p (mode-a (last-i-reg)))),
    if* (a-immediate-p (last-i-reg),
          sign-extend (a-immediate (i-reg),
                        32),
          read-regs (rn-a (last-i-reg), regs)),
    a-reg),
    b-reg,
    i-reg,
    data-out,
    addr-out,
    t,
    t,
    hold-,
    pc-reg,
    cv_readb2 (t,
                rn-a (last-i-reg),
                i-reg,
                flags,
                pc-reg)),
  list (mem,
        0,
        clock,
        0,

```



```

      t,
      t,
      addr-out,
      v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: readb2\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_readb2 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg))),
list (mem,
      0,

```

```

                                clock,
                                mem-count,
                                mem-dtack,
                                t,
                                addr-out,
                                last-data)),
                                inputs,
                                n)
=  if car (clock)  $\simeq$  0
    then run-fm9001 (list (list (write-regs (and* (not* (store-resultp (store-cc (last-i-reg),
                                                                    last-flags))),
                                                or* (pre-dec-p (mode-b (last-i-reg)),
                                                                post-inc-p (mode-b (last-i-reg)))),
                                                rn-b (last-i-reg),
                                                regs,
                                                bv (core-alu (if* (pre-dec-p (mode-b (last-i-reg)),
                                                                carry-in-help (cons (c-flag (last-flags),
                                                                    cons (f,
                                                                    ALU-DEC-OP)))),
                                                                carry-in-help (cons (c-flag (last-flags),
                                                                    cons (f,
                                                                    ALU-INC-OP))))),
                                                                    a-reg,
                                                                    b-reg,
                                                                    f,
                                                                    if* (pre-dec-p (mode-b (last-i-reg)),
                                                                mpg (cons (f,
                                                                    cons (t,
                                                                    ALU-DEC-OP))),
                                                                mpg (cons (f,
                                                                    cons (t,
                                                                    ALU-INC-OP))))),
                                                                    if* (pre-dec-p (mode-b (last-i-reg)),
                                                                ALU-DEC-OP,
                                                                ALU-INC-OP),
                                                                make-tree (32))))),
                                flags,
                                a-reg,
                                b-reg,
                                i-reg,
                                data-out,
                                addr-out,
                                t,
                                f,

```

```

        hold-,
        pc-reg,
        cv_readb3 (t,
                    rn-b (last-i-reg),
                    i-reg,
                    flags,
                    pc-reg)),
        list (mem,
              1,
              cdr (clock),
              0,
              t,
              t,
              addr-out,
              v-threefix (last-data))),
        cdr (inputs),
        n - 1)
else run-fm9001 (list (list (write-regs (and* (not* (store-resultp (store-cc (last-i-reg),
                                                                              last-flags)),
or* (pre-dec-p (mode-b (last-i-reg)),
post-inc-p (mode-b (last-i-reg))))),
rn-b (last-i-reg),
regs,
bv (core-alu (if* (pre-dec-p (mode-b (last-i-reg)),
carry-in-help (cons (c-flag (last-flags),
cons (f,
ALU-DEC-OP)))),
carry-in-help (cons (c-flag (last-flags),
cons (f,
ALU-INC-OP))))),
a-reg,
b-reg,
f,
if* (pre-dec-p (mode-b (last-i-reg)),
mpg (cons (f,
cons (t,
ALU-DEC-OP))),
mpg (cons (f,
cons (t,
ALU-INC-OP))))),
if* (pre-dec-p (mode-b (last-i-reg)),
ALU-DEC-OP,
ALU-INC-OP),
make-tree (32))))),

```

```

      flags,
      a-reg,
      b-reg,
      i-reg,
      data-out,
      addr-out,
      t,
      t,
      hold-,
      pc-reg,
      cv_readb3 (t,
                  rn-b (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg)),
    list (mem,
          1,
          cdr (clock),
          car (clock) - 1,
          f,
          t,
          addr-out,
          v-threefix (last-data))),
  cdr (inputs),
  n - 1) endif)

```

THEOREM: readb3\$step0
 (cv-hyps (**t**, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 ^ cv-hyps (**t**, pc-reg, i-reg, flags, pc-reg)
 ^ memory-okp (32, 32, mem)
 ^ regfile-okp (regs)
 ^ bvp (a-reg)
 ^ (length (a-reg) = 32)
 ^ bvp (b-reg)
 ^ (length (b-reg) = 32)
 ^ bvp (addr-out)
 ^ (length (addr-out) = 32)
 ^ bvp (data-out)
 ^ (length (data-out) = 32)
 ^ (n $\neq$ 0)
 ^ run-inputs-p (inputs, 1)
 ^ boolp (**f**)
 ^ boolp (hold-)
 ^ (reset- = **t**)

$$\begin{aligned}
& \wedge \quad (mem\text{-}count \simeq 0)) \\
\rightarrow & \quad (run\text{-}fm9001 \text{ (list (list (regs,} \\
& \quad \quad \quad flags, \\
& \quad \quad \quad a\text{-reg,} \\
& \quad \quad \quad b\text{-reg,} \\
& \quad \quad \quad i\text{-reg,} \\
& \quad \quad \quad data\text{-out,} \\
& \quad \quad \quad addr\text{-out,} \\
& \quad \quad \quad reset\text{-,} \\
& \quad \quad \quad \mathbf{f}, \\
& \quad \quad \quad hold\text{-,} \\
& \quad \quad \quad pc\text{-reg,} \\
& \quad \quad \quad cv\_readb3 \text{ (t,} \\
& \quad \quad \quad \quad \quad last\text{-regs-address,} \\
& \quad \quad \quad \quad \quad last\text{-i-reg,} \\
& \quad \quad \quad \quad \quad last\text{-flags,} \\
& \quad \quad \quad \quad \quad last\text{-pc-reg}),} \\
& \quad \quad \quad list \text{ (mem, 1, clock, mem-count, t, t, addr-out, last-data)),} \\
& \quad \quad \quad inputs, \\
& \quad \quad \quad n) \\
= & \quad run\text{-fm9001 \text{ (list (list (write-regs (\mathbf{f}, 0, regs, 0),} \\
& \quad \quad \quad flags, \\
& \quad \quad \quad a\text{-reg,} \\
& \quad \quad \quad read\text{-mem (addr-out, mem),} \\
& \quad \quad \quad i\text{-reg,} \\
& \quad \quad \quad data\text{-out,} \\
& \quad \quad \quad addr\text{-out,} \\
& \quad \quad \quad \mathbf{t}, \\
& \quad \quad \quad \mathbf{f}, \\
& \quad \quad \quad hold\text{-,} \\
& \quad \quad \quad pc\text{-reg,} \\
& \quad \quad \quad if* \text{ (store-resultp (store-cc (i-reg), flags),} \\
& \quad \quad \quad \quad \quad cv\_write0 \text{ (t,} \\
& \quad \quad \quad \quad \quad \quad \quad last\text{-pc-reg,} \\
& \quad \quad \quad \quad \quad \quad \quad i\text{-reg,} \\
& \quad \quad \quad \quad \quad \quad \quad flags,} \\
& \quad \quad \quad \quad \quad \quad \quad pc\text{-reg}),} \\
& \quad \quad \quad \quad \quad cv\_update \text{ (t,} \\
& \quad \quad \quad \quad \quad \quad \quad last\text{-pc-reg,} \\
& \quad \quad \quad \quad \quad \quad \quad i\text{-reg,} \\
& \quad \quad \quad \quad \quad \quad \quad flags,} \\
& \quad \quad \quad \quad \quad \quad \quad pc\text{-reg))),} \\
& \quad \quad \quad list \text{ (mem,} \\
& \quad \quad \quad \quad \quad 1,}
\end{aligned}$$

```

        clock,
        0,
        t,
        t,
        addr-out,
        v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: readb3\$step1

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (t)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           t,
                           hold-,
                           pc-reg,
                           cv_readb3 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg))),

```

```

        list (mem, 1, clock, mem-count, f, t, addr-out, last-data)),
        inputs,
        n)
=   if mem-count  $\simeq$  0
    then run-fm9001 (list (list (write-regs (f, 0, regs, 0),
        flags,
        a-reg,
        read-regs (last-pc-reg, regs),
        i-reg,
        data-out,
        addr-out,
        t,
        f,
        hold-,
        pc-reg,
        cv_readb3 (t,
                    last-pc-reg,
                    i-reg,
                    flags,
                    pc-reg))),
        list (mem,
        1,
        clock,
        0,
        t,
        t,
        addr-out,
        v-threefix (last-data))),
        cdr (inputs),
        n - 1)
    else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
        flags,
        a-reg,
        read-regs (last-pc-reg, regs),
        i-reg,
        data-out,
        addr-out,
        t,
        t,
        hold-,
        pc-reg,
        cv_readb3 (t,
                    last-pc-reg,
                    i-reg,

```

```

                                flags,
                                pc-reg)),
list (mem,
      1,
      clock,
      mem-count - 1,
      f,
      t,
      addr-out,
      v-threefix (last-data))),
cdr (inputs),
n - 1) endif)

```

THEOREM: write0\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ ((mem-state = 0) ∨ (mem-state = 1)))
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_write0 (t,

```



```

                                last-regs-address,
                                last-i-reg,
                                last-flags,
                                last-pc-reg)),
list (mem,
      mem-state,
      clock,
      mem-count,
      mem-dtack,
      t,
      last-address,
      last-data)),
inputs,
n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                   update-flags (flags,
                                                  set-flags (last-i-reg),
                                                  v-alu (c-flag (last-flags),
                                                         a-reg,
                                                         b-reg,
                                                         op-code (last-i-reg))),
                                   a-reg,
                                   read-regs (rn-b (last-i-reg), regs),
                                   i-reg,
                                   bv (v-alu (c-flag (last-flags),
                                                         a-reg,
                                                         b-reg,
                                                         op-code (last-i-reg))),
                                   if* (pre-dec-p (mode-b (last-i-reg)),
                                       v-dec (read-regs (rn-b (last-i-reg),
                                                         regs)),
                                       read-regs (rn-b (last-i-reg), regs)),
                                   t,
                                   t,
                                   hold-,
                                   pc-reg,
                                   cv_writel (t,
                                              rn-b (last-i-reg),
                                              i-reg,
                                              flags,
                                              pc-reg)),
list (mem,
      0,
      clock,

```

```

0,
t,
t,
addr-out,
v-threefix (last-data))),
cdr (inputs),
n - 1))

```

THEOREM: write1\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv-write1 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg)),
                        list (mem,

```

```

0,
clock,
mem-count,
mem-dtack,
t,
last-address,
last-data)),
inputs,
n)
= run-fm9001 (list (list (write-regs (or* (pre-dec-p (mode-b (last-i-reg))),
                                         post-inc-p (mode-b (last-i-reg))),
                                         rn-b (last-i-reg),
                                         regs,
                                         bv (core-alu (if* (pre-dec-p (mode-b (last-i-reg)),
                                                             carry-in-help (cons (c-flag (last-flags),
                                                             cons (f,
                                                             ALU-DEC-OP))),
                                                             carry-in-help (cons (c-flag (last-flags),
                                                             cons (f,
                                                             ALU-INC-OP))))),
                                         a-reg,
                                         b-reg,
                                         f,
                                         if* (pre-dec-p (mode-b (last-i-reg)),
                                         mpg (cons (f,
                                         cons (t,
                                         ALU-DEC-OP))),
                                         mpg (cons (f,
                                         cons (t,
                                         ALU-INC-OP))))),
                                         if* (pre-dec-p (mode-b (last-i-reg)),
                                         ALU-DEC-OP,
                                         ALU-INC-OP),
                                         make-tree (32))))),
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
t,
t,
hold-,
pc-reg,

```

```

cv_write2(f,
          rn-b(last-i-reg),
          i-reg,
          flags,
          pc-reg)),
list(mem, 0, clock, 0, t, f, addr-out, data-out)),
cdr(inputs),
n - 1))

```

THEOREM: write2\$step

```

(cv-hyps(f, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps(t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp(32, 32, mem)
  ^ regfile-okp(regs)
  ^ bvp(a-reg)
  ^ (length(a-reg) = 32)
  ^ bvp(b-reg)
  ^ (length(b-reg) = 32)
  ^ bvp(addr-out)
  ^ (length(addr-out) = 32)
  ^ bvp(data-out)
  ^ (length(data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p(inputs, 1)
  ^ boolp(dtack-)
  ^ boolp(hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001(list(list(regs,
                        flags,
                        a-reg,
                        b-reg,
                        i-reg,
                        data-out,
                        addr-out,
                        reset-,
                        dtack-,
                        hold-,
                        pc-reg,
                        cv_write2(f,
                                last-regs-address,
                                last-i-reg,
                                last-flags,
                                last-pc-reg))),

```

```

list (mem,
      0,
      clock,
      mem-count,
      mem-dtack,
      f,
      addr-out,
      data-out)),
inputs,
n)
= if car (clock)  $\simeq$  0
  then run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                           flags,
                                           a-reg,
                                           b-reg,
                                           i-reg,
                                           data-out,
                                           addr-out,
                                           t,
                                           f,
                                           hold-,
                                           pc-reg,
                                           cv_write3 (f,
                                                         last-pc-reg,
                                                         i-reg,
                                                         flags,
                                                         pc-reg))),
list (mem,
      2,
      cdr (clock),
      0,
      t,
      f,
      addr-out,
      data-out)),
cdr (inputs),
n - 1)
  else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                           flags,
                                           a-reg,
                                           b-reg,
                                           i-reg,
                                           data-out,
                                           addr-out,

```

```

      t,
      t,
      hold-,
      pc-reg,
      cv_write3 (f,
                  last-pc-reg,
                  i-reg,
                  flags,
                  pc-reg)),
      list (mem,
            2,
            cdr (clock),
            car (clock) - 1,
            f,
            f,
            addr-out,
            data-out)),
      cdr (inputs),
      n - 1) endif)

```

THEOREM: write3\$step0
 (cv-hyps (**f**, *last-regs-address*, *last-i-reg*, *last-flags*, *last-pc-reg*)
 ^ cv-hyps (**t**, *pc-reg*, *i-reg*, *flags*, *pc-reg*)
 ^ memory-okp (32, 32, *mem*)
 ^ regfile-okp (*regs*)
 ^ bvp (*a-reg*)
 ^ (length (*a-reg*) = 32)
 ^ bvp (*b-reg*)
 ^ (length (*b-reg*) = 32)
 ^ bvp (*addr-out*)
 ^ (length (*addr-out*) = 32)
 ^ bvp (*data-out*)
 ^ (length (*data-out*) = 32)
 ^ (*n* $\neq$ 0)
 ^ run-inputs-p (*inputs*, 1)
 ^ boolp (**f**)
 ^ boolp (*hold-*)
 ^ (*reset* = **t**)
 ^ (*mem-count* $\simeq$ 0))
 → (run-fm9001 (list (list (*regs*,
 flags,
 a-reg,
 b-reg,
 i-reg,

$$\begin{aligned}
& \text{data-out,} \\
& \text{addr-out,} \\
& \text{reset-,} \\
& \mathbf{f}, \\
& \text{hold-,} \\
& \text{pc-reg,} \\
& \text{cv\_write3}(\mathbf{f}, \\
& \quad \text{last-regs-address,} \\
& \quad \text{last-i-reg,} \\
& \quad \text{last-flags,} \\
& \quad \text{last-pc-reg}), \\
& \text{list}(\text{mem}, 2, \text{clock}, \text{mem-count}, \mathbf{t}, \mathbf{f}, \text{addr-out}, \text{data-out}), \\
& \text{inputs,} \\
& n) \\
= & \text{run-fm9001}(\text{list}(\text{list}(\text{write-regs}(\mathbf{f}, 0, \text{regs}, 0), \\
& \quad \text{flags,} \\
& \quad \text{a-reg,} \\
& \quad \text{b-reg,} \\
& \quad \text{i-reg,} \\
& \quad \text{data-out,} \\
& \quad \text{addr-out,} \\
& \quad \mathbf{t}, \\
& \quad \mathbf{f}, \\
& \quad \text{hold-,} \\
& \quad \text{pc-reg,} \\
& \quad \text{cv\_fetch0}(\mathbf{f}, \\
& \quad \quad \text{last-pc-reg,} \\
& \quad \quad \text{i-reg,} \\
& \quad \quad \text{flags,} \\
& \quad \quad \text{pc-reg}), \\
& \quad \text{list}(\text{mem}, 2, \text{clock}, 0, \mathbf{t}, \mathbf{f}, \text{addr-out}, \text{data-out}), \\
& \text{cdr}(\text{inputs}), \\
& n - 1))
\end{aligned}$$

THEOREM: write3\$step1

$$\begin{aligned}
& (\text{cv-hyps}(\mathbf{f}, \text{last-regs-address}, \text{last-i-reg}, \text{last-flags}, \text{last-pc-reg}) \\
& \wedge \text{cv-hyps}(\mathbf{t}, \text{pc-reg}, \text{i-reg}, \text{flags}, \text{pc-reg}) \\
& \wedge \text{memory-okp}(32, 32, \text{mem}) \\
& \wedge \text{regfile-okp}(\text{regs}) \\
& \wedge \text{bvp}(\text{a-reg}) \\
& \wedge (\text{length}(\text{a-reg}) = 32) \\
& \wedge \text{bvp}(\text{b-reg}) \\
& \wedge (\text{length}(\text{b-reg}) = 32) \\
& \wedge \text{bvp}(\text{addr-out})
\end{aligned}$$

```

^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ (n ≠ 0)
^ run-inputs-p (inputs, 1)
^ boolp (t)
^ boolp (hold-)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           t,
                           hold-,
                           pc-reg,
                           cv_write3 (f,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg)),
                           list (mem, 2, clock, mem-count, f, f, addr-out, data-out)),
    inputs,
    n)
= if mem-count ≈ 0
  then run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                           flags,
                                           a-reg,
                                           b-reg,
                                           i-reg,
                                           data-out,
                                           addr-out,
                                           t,
                                           f,
                                           hold-,
                                           pc-reg,
                                           cv_write3 (f,
                                                         last-pc-reg,
                                                         i-reg,
                                                         flags,

```



```

                                pc-reg)),
list (mem,
      2,
      clock,
      0,
      t,
      f,
      addr-out,
      data-out)),
cdr (inputs),
  n - 1)
else run-fm9001 (list (list (write-regs (f, 0, regs, 0),
      flags,
      a-reg,
      b-reg,
      i-reg,
      data-out,
      addr-out,
      t,
      t,
      hold-,
      pc-reg,
      cv_write3 (f,
                  last-pc-reg,
                  i-reg,
                  flags,
                  pc-reg)),
list (mem,
      2,
      clock,
      mem-count - 1,
      f,
      f,
      addr-out,
      data-out)),
cdr (inputs),
  n - 1) endif)

```

THEOREM: sefa0\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ∧ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ∧ memory-okp (32, 32, mem)
  ∧ regfile-okp (regs)
  ∧ bvp (a-reg)

```

```

^ (length (a-reg) = 32)
^ bvp (b-reg)
^ (length (b-reg) = 32)
^ bvp (addr-out)
^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ (n ≠ 0)
^ run-inputs-p (inputs, 1)
^ boolp (dtack-)
^ boolp (hold-)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_sefa0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg)),
                        list (mem,
                              0,
                              clock,
                              mem-count,
                              mem-dtack,
                              t,
                              last-address,
                              last-data))),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     read-regs (rn-a (last-i-reg), regs),
                                     b-reg,
                                     i-reg,

```

```

      data-out,
      addr-out,
      t,
      t,
      hold-,
      pc-reg,
      cv_sefal (t,
                rn-a (last-i-reg),
                i-reg,
                flags,
                pc-reg)),
      list (mem,
            0,
            clock,
            0,
            t,
            t,
            addr-out,
            v-threefix (last-data))),
      cdr (inputs),
      n - 1))

```

THEOREM: sefal\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ (n ≠ 0)
  ^ run-inputs-p (inputs, 1)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,

```

```

    b-reg,
    i-reg,
    data-out,
    addr-out,
    reset-,
    dtack-,
    hold-,
    pc-reg,
    cv_sefal (t,
              last-regs-address,
              last-i-reg,
              last-flags,
              last-pc-reg)),
  list (mem,
        0,
        clock,
        mem-count,
        mem-dtack,
        t,
        last-address,
        last-data)),
  inputs,
  n)
= run-fm9001 (list (list (write-regs (t,
                                     rn-a (last-i-reg),
                                     regs,
                                     bv (core-alu (if* (pre-dec-p (mode-a (last-i-reg)),
                                                         carry-in-help (cons (c-flag (last-flags),
                                                         cons (f,
                                                         ALU-DEC-OP)))),
                                                         carry-in-help (cons (c-flag (last-flags),
                                                         cons (f,
                                                         ALU-INC-OP))))),
                                     a-reg,
                                     b-reg,
                                     f,
                                     if* (pre-dec-p (mode-a (last-i-reg)),
                                                         mpg (cons (f,
                                                         cons (f,
                                                         ALU-DEC-OP)))),
                                                         mpg (cons (f,
                                                         cons (f,
                                                         ALU-INC-OP))))),
                                     if* (pre-dec-p (mode-a (last-i-reg)),

```

```

        ALU-DEC-OP,
        ALU-INC-OP),
    make-tree (32)))))

    flags,
    a-reg,
    b-reg,
    i-reg,
    data-out,
    addr-out,
    t,
    t,
    hold-,
    pc-reg,
    if* (or* (pre-dec-p (mode-b (i-reg)),
                  post-inc-p (mode-b (i-reg))),
        cv_sefb0 (t,
                  rn-a (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg),
        cv_fetch0 (t,
                  rn-a (last-i-reg),
                  i-reg,
                  flags,
                  pc-reg))),
    list (mem,
          0,
          clock,
          0,
          t,
          t,
          addr-out,
          v-threefix (last-data))),
    cdr (inputs),
    n - 1))

```

THEOREM: sefb0\$step

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
  ∧ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ∧ memory-okp (32, 32, mem)
  ∧ regfile-okp (regs)
  ∧ bvp (a-reg)
  ∧ (length (a-reg) = 32)
  ∧ bvp (b-reg)

```

```

^ (length (b-reg) = 32)
^ bvp (addr-out)
^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ (n ≠ 0)
^ run-inputs-p (inputs, 1)
^ boolp (dtack-)
^ boolp (hold-)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_sefb0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       last-pc-reg)),
                        list (mem,
                              0,
                              clock,
                              mem-count,
                              mem-dtack,
                              t,
                              last-address,
                              last-data))),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (f, 0, regs, 0),
                                     flags,
                                     a-reg,
                                     read-regs (rn-b (last-i-reg), regs),
                                     i-reg,
                                     data-out,
                                     addr-out,

```

```

      t,
      t,
      hold-,
      pc-reg,
      cv_sefb1 (t,
                 rn-b (last-i-reg),
                 i-reg,
                 flags,
                 pc-reg)),
      list (mem,
            0,
            clock,
            0,
            t,
            t,
            addr-out,
            v-threefix (last-data))),
      cdr (inputs),
      n - 1))

```

THEOREM: sefb1\$step
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, last-pc-reg)
 ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 ^ memory-okp (32, 32, mem)
 ^ regfile-okp (regs)
 ^ bvp (a-reg)
 ^ (length (a-reg) = 32)
 ^ bvp (b-reg)
 ^ (length (b-reg) = 32)
 ^ bvp (addr-out)
 ^ (length (addr-out) = 32)
 ^ bvp (data-out)
 ^ (length (data-out) = 32)
 ^ (n ≠ 0)
 ^ run-inputs-p (inputs, 1)
 ^ boolp (dtack-)
 ^ boolp (hold-)
 ^ (reset- = t)
 ^ nil)
 → (run-fm9001 (list (list (regs,
 flags,
 a-reg,
 b-reg,
 i-reg,

```

      data-out,
      addr-out,
      reset-,
      dtack-,
      hold-,
      pc-reg,
      cv_sefb1 (t,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  last-pc-reg)),
    list (mem,
          0,
          clock,
          mem-count,
          mem-dtack,
          t,
          last-address,
          last-data)),
    inputs,
    n)
= run-fm9001 (list (list (write-regs (t,
                                     rn-b (last-i-reg),
                                     regs,
                                     bv (core-alu (if* (pre-dec-p (mode-b (last-i-reg)),
                                                         carry-in-help (cons (c-flag (last-flags),
                                                         cons (f,
                                                         ALU-DEC-OP)))),
                                                         carry-in-help (cons (c-flag (last-flags),
                                                         cons (f,
                                                         ALU-INC-OP))))),
                                     a-reg,
                                     b-reg,
                                     f,
                                     if* (pre-dec-p (mode-b (last-i-reg)),
                                     mpg (cons (f,
                                                         cons (t,
                                                         ALU-DEC-OP))),
                                     mpg (cons (f,
                                                         cons (t,
                                                         ALU-INC-OP))))),
                                     if* (pre-dec-p (mode-b (last-i-reg)),
                                     ALU-DEC-OP,
                                     ALU-INC-OP),

```



```

make-tree(32))),
  flags,
  a-reg,
  b-reg,
  i-reg,
  data-out,
  addr-out,
  t,
  t,
  hold-,
  pc-reg,
  cv_fetch0(t,
    rn-b(last-i-reg),
    i-reg,
    flags,
    pc-reg)),
list(mem,
  0,
  clock,
  0,
  t,
  t,
  addr-out,
  v-threefix(last-data))),
cdr(inputs),
n - 1))

```

EVENT: Let us define the theory *fm9001-step-theory* to consist of the following events: sefb1\$step, sefb0\$step, sefa1\$step, sefa0\$step, write3\$step1, write3\$step0, write2\$step, write1\$step, write0\$step, readb3\$step1, readb3\$step0, readb2\$step, readb1\$step, readb0\$step, reada3\$step1, reada3\$step0, reada2\$step, reada1\$step, reada0\$step, update\$step, regb\$step, rega\$step, decode\$step, fetch3\$step1, fetch3\$step0, fetch2\$step, fetch1\$step, fetch0\$step2, fetch0\$step1, fetch0\$step0.

DEFINITION:

```

read-fn(reg0, reg1, n)
=  if n ≃ 0 then reg0
   else read-fn(reg1, reg1, n - 1) endif

```

EVENT: Enable read-fn; name this event 'read-fn-off'.

THEOREM: open-read-fn

```

((n ≃ 0) → (read-fn(reg0, reg1, n) = reg0))
∧ ((n ≠ 0) → (read-fn(reg0, reg1, n) = read-fn(reg1, reg1, n - 1)))

```

THEOREM: length-read-fn
 $((\text{length}(\text{reg0}) = 32) \wedge (\text{length}(\text{reg1}) = 32))$
 $\rightarrow (\text{length}(\text{read-fn}(\text{reg0}, \text{reg1}, n)) = 32)$

THEOREM: properp-read-fn
 $(\text{properp}(\text{reg0}) \wedge \text{properp}(\text{reg1})) \rightarrow \text{properp}(\text{read-fn}(\text{reg0}, \text{reg1}, n))$

THEOREM: bvp-read-fn
 $(\text{bvp}(\text{reg0}) \wedge \text{bvp}(\text{reg1})) \rightarrow \text{bvp}(\text{read-fn}(\text{reg0}, \text{reg1}, n))$

DEFINITION:

fetch3\$induction(*n*,
 clock,
 count,
 inputs,
 regs,
 a-reg,
 b-reg,
 rw-,
 regs-address,
 i-reg,
 flags,
 pc-reg,
 last-rw-,
 last-regs-address,
 last-i-reg,
 last-flags,
 last-data)
 = **if** *n* $\simeq$ 0 **then** **t**
 else fetch3\$induction(*n* - 1,
 clock,
 count - 1,
 cdr(*inputs*),
 write-regs(**f**, 0, *regs*, 0),
 a-reg,
 b-reg,
 t,
 regs-address,
 read-regs(*pc-reg*, *regs*),
 flags,
 pc-reg,
 rw-,
 pc-reg,
 i-reg,
 flags,

v-threefix (*last-data*) **endif**

THEOREM: fetch3\$progress-help

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, last-regs-address, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, 1 + n)
  ^ if count < n then f
    else t endif)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           t,
                           t,
                           hold-,
                           pc-reg,
                           cv_fetch3 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg)),
                     list (mem, 1, clock, count, f, t, addr-out, last-data)),
    inputs,
    1 + n)
= list (list (write-regs (f, 0, regs, 0),
                  flags,
                  a-reg,
                  b-reg,
                  read-fn (i-reg, read-regs (pc-reg, regs), 1 + n),
                  data-out,
```

```

    addr-out,
    t,
    ¬ if n < count then f
      else t endif,
    hold-,
    pc-reg,
    cv_fetch3 (t,
                pc-reg,
                read-fn (i-reg, read-regs (pc-reg, regs), n),
                flags,
                pc-reg)),
  list (mem,
        1,
        clock,
        count - (1 + n),
        if n < count then f
        else t endif,
        t,
        addr-out,
        v-threefix (last-data))))

```

THEOREM: fetch3\$progress

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ∧ cv-hyps (t, last-regs-address, i-reg, flags, pc-reg)
  ∧ memory-okp (32, 32, mem)
  ∧ regfile-okp (regs)
  ∧ bvp (a-reg)
  ∧ (length (a-reg) = 32)
  ∧ bvp (b-reg)
  ∧ (length (b-reg) = 32)
  ∧ bvp (addr-out)
  ∧ (length (addr-out) = 32)
  ∧ bvp (data-out)
  ∧ (length (data-out) = 32)
  ∧ boolp (hold-)
  ∧ run-inputs-p (inputs, count)
  ∧ (count ≠ 0))
→ (run-fm9001 (list (list (regs,
                             flags,
                             a-reg,
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,

```

$$\begin{aligned}
& \text{t,} \\
& \text{t,} \\
& \text{hold-,} \\
& \text{pc-reg,} \\
& \text{cv\_fetch3 (t,} \\
& \quad \text{last-regs-address,} \\
& \quad \text{last-i-reg,} \\
& \quad \text{last-flags,} \\
& \quad \text{pc-reg)),} \\
& \text{list (mem, 1, clock, count - 1, f, t, addr-out, last-data)),} \\
& \text{inputs,} \\
& \text{count)} \\
= & \text{list (list (write-regs (f, 0, regs, 0),} \\
& \quad \text{flags,} \\
& \quad \text{a-reg,} \\
& \quad \text{b-reg,} \\
& \quad \text{read-fn (i-reg, read-regs (pc-reg, regs), count),} \\
& \quad \text{data-out,} \\
& \quad \text{addr-out,} \\
& \quad \text{t,} \\
& \quad \text{f,} \\
& \quad \text{hold-,} \\
& \quad \text{pc-reg,} \\
& \quad \text{cv\_fetch3 (t,} \\
& \quad \quad \text{pc-reg,} \\
& \quad \quad \text{read-fn (i-reg,} \\
& \quad \quad \quad \text{read-regs (pc-reg, regs),} \\
& \quad \quad \quad \text{count - 1),} \\
& \quad \quad \text{flags,} \\
& \quad \quad \text{pc-reg)),} \\
& \text{list (mem, 1, clock, 0, t, t, addr-out, v-threefix (last-data))))
\end{aligned}$$

DEFINITION:

$\text{reada3\$induction (n,}$
 $\quad \text{clock,}$
 $\quad \text{count,}$
 $\quad \text{inputs,}$
 $\quad \text{regs,}$
 $\quad \text{a-reg,}$
 $\quad \text{b-reg,}$
 $\quad \text{rw-,}$
 $\quad \text{regs-address,}$
 $\quad \text{i-reg,}$
 $\quad \text{flags,}$

```

      pc-reg,
      last-rw-,
      last-regs-address,
      last-i-reg,
      last-flags,
      last-data)
=  if  $n \simeq 0$  then t
    else reada3$induction ( $n - 1$ ,
      clock,
      count - 1,
      cdr (inputs),
      write-regs (f, 0, regs, 0),
      read-regs (pc-reg, regs),
      b-reg,
      t,
      regs-address,
      i-reg,
      flags,
      pc-reg,
      rw-,
      pc-reg,
      i-reg,
      flags,
      v-threefix (last-data)) endif

```

THEOREM: reada3\$progress-help
 (cv-hyps (**t**, last-regs-address, last-i-reg, last-flags, pc-reg)
 $\wedge$ cv-hyps (**t**, last-regs-address, i-reg, flags, pc-reg)
 $\wedge$ memory-okp (32, 32, mem)
 $\wedge$ regfile-okp (regs)
 $\wedge$ bvp (a-reg)
 $\wedge$ (length (a-reg) = 32)
 $\wedge$ bvp (b-reg)
 $\wedge$ (length (b-reg) = 32)
 $\wedge$ bvp (addr-out)
 $\wedge$ (length (addr-out) = 32)
 $\wedge$ bvp (data-out)
 $\wedge$ (length (data-out) = 32)
 $\wedge$ boolp (hold-)
 $\wedge$ run-inputs-p (inputs, 1 + n)
 $\wedge$ **if** count < n **then f**
 else t endif)
 $\rightarrow$ (run-fm9001 (list (list (regs,
 flags,

```

      a-reg,
      b-reg,
      i-reg,
      data-out,
      addr-out,
      t,
      t,
      hold-,
      pc-reg,
      cv_reada3 (t,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  pc-reg)),
      list (mem, 1, clock, count, f, t, addr-out, last-data)),
    inputs,
    1 + n)
= list (list (write-regs (f, 0, regs, 0),
                      flags,
                      read-fn (a-reg, read-regs (pc-reg, regs), 1 + n),
                      b-reg,
                      i-reg,
                      data-out,
                      addr-out,
                      t,
                      ¬ if n < count then f
                        else t endif,
                      hold-,
                      pc-reg,
                      cv_reada3 (t, pc-reg, i-reg, flags, pc-reg)),
    list (mem,
          1,
          clock,
          count - (1 + n),
          if n < count then f
            else t endif,
          t,
          addr-out,
          v-threefix (last-data))))

```

THEOREM: reada3\$progress
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ∧ cv-hyps (t, last-regs-address, i-reg, flags, pc-reg)
 ∧ memory-okp (32, 32, mem))

```

 $\wedge$  regfile-okp (regs)
 $\wedge$  bvp (a-reg)
 $\wedge$  (length (a-reg) = 32)
 $\wedge$  bvp (b-reg)
 $\wedge$  (length (b-reg) = 32)
 $\wedge$  bvp (addr-out)
 $\wedge$  (length (addr-out) = 32)
 $\wedge$  bvp (data-out)
 $\wedge$  (length (data-out) = 32)
 $\wedge$  boolp (hold-)
 $\wedge$  run-inputs-p (inputs, count)
 $\wedge$  (count  $\neq$  0))
 $\rightarrow$  (run-fm9001 (list (list (regs,
                             flags,
                             a-reg,
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,
                             t,
                             t,
                             hold-,
                             pc-reg,
                             cv_reada3 (t,
                                         last-regs-address,
                                         last-i-reg,
                                         last-flags,
                                         pc-reg)),
                             list (mem, 1, clock, count - 1, f, t, addr-out, last-data)),
                             inputs,
                             count)
      = list (list (write-regs (f, 0, regs, 0),
                             flags,
                             read-fn (a-reg, read-regs (pc-reg, regs), count),
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,
                             t,
                             f,
                             hold-,
                             pc-reg,
                             cv_reada3 (t, pc-reg, i-reg, flags, pc-reg)),
                             list (mem, 1, clock, 0, t, t, addr-out, v-threefix (last-data))))

```


DEFINITION:

```

readb3$induction (n,
                  clock,
                  count,
                  inputs,
                  regs,
                  a-reg,
                  b-reg,
                  rw-,
                  regs-address,
                  i-reg,
                  flags,
                  pc-reg,
                  last-rw-,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  last-data)
=  if n  $\simeq$  0 then t
   else readb3$induction (n - 1,
                         clock,
                         count - 1,
                         cdr (inputs),
                         write-regs (f, 0, regs, 0),
                         a-reg,
                         read-regs (pc-reg, regs),
                         t,
                         regs-address,
                         i-reg,
                         flags,
                         pc-reg,
                         rw-,
                         pc-reg,
                         i-reg,
                         flags,
                         v-threefix (last-data)) endif

```

THEOREM: readb3\$progress-help

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
 $\wedge$  cv-hyps (t, last-regs-address, i-reg, flags, pc-reg)
 $\wedge$  memory-okp (32, 32, mem)
 $\wedge$  regfile-okp (regs)
 $\wedge$  bvp (a-reg)
 $\wedge$  (length (a-reg) = 32)

```

```

 $\wedge$    bvp(b-reg)
 $\wedge$    (length(b-reg) = 32)
 $\wedge$    bvp(addr-out)
 $\wedge$    (length(addr-out) = 32)
 $\wedge$    bvp(data-out)
 $\wedge$    (length(data-out) = 32)
 $\wedge$    boolp(hold-)
 $\wedge$    run-inputs-p(inputs, 1 + n)
 $\wedge$    if count < n then f
      else t endif
 $\rightarrow$   (run-fm9001 (list (list (regs,
                               flags,
                               a-reg,
                               b-reg,
                               i-reg,
                               data-out,
                               addr-out,
                               t,
                               t,
                               hold-,
                               pc-reg,
                               cv_readb3(t,
                                           last-regs-address,
                                           last-i-reg,
                                           last-flags,
                                           pc-reg)),
                          list(mem, 1, clock, count, f, t, addr-out, last-data)),
                               inputs,
                               1 + n)
      =  list (list (write-regs(f, 0, regs, 0),
                      flags,
                      a-reg,
                      read-fn(b-reg, read-regs(pc-reg, regs), 1 + n),
                      i-reg,
                      data-out,
                      addr-out,
                      t,
                       $\neg$  if n < count then f
                        else t endif,
                      hold-,
                      pc-reg,
                      cv_readb3(t, pc-reg, i-reg, flags, pc-reg)),
                  list(mem,
                      1,

```

```

    clock,
    count - (1 + n),
    if n < count then f
    else t endif,
    t,
    addr-out,
    v-threefix (last-data))))

```

THEOREM: readb3\$progress

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, last-regs-address, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, count)
  ^ (count ≠ 0))
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           t,
                           t,
                           hold-,
                           pc-reg,
                           cv_readb3 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg)),
                        list (mem, 1, clock, count - 1, f, t, addr-out, last-data)),
    inputs,
    count)
= list (list (write-regs (f, 0, regs, 0),

```

```

    flags,
    a-reg,
    read-fn (b-reg, read-regs (pc-reg, regs), count),
    i-reg,
    data-out,
    addr-out,
    t,
    f,
    hold-,
    pc-reg,
    cv_readb3 (t, pc-reg, i-reg, flags, pc-reg)),
list (mem, 1, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

DEFINITION:

```

write3$induction (n,
    clock,
    count,
    inputs,
    regs,
    a-reg,
    b-reg,
    rw-,
    regs-address,
    i-reg,
    flags,
    pc-reg,
    last-rw-,
    last-regs-address,
    last-i-reg,
    last-flags,
    last-data)
=  if n  $\simeq$  0 then t
    else write3$induction (n - 1,
        clock,
        count - 1,
        cdr (inputs),
        write-regs (f, 0, regs, 0),
        a-reg,
        b-reg,
        f,
        regs-address,
        i-reg,
        flags,
        pc-reg,

```

```

    rw-,
    pc-reg,
    i-reg,
    flags,
    v-threefix(last-data)) endif

```

THEOREM: write3\$progress-help

```

(cv-hyps(t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps(t, last-regs-address, i-reg, flags, pc-reg)
  ^ memory-okp(32, 32, mem)
  ^ regfile-okp(regs)
  ^ bvp(a-reg)
  ^ (length(a-reg) = 32)
  ^ bvp(b-reg)
  ^ (length(b-reg) = 32)
  ^ bvp(addr-out)
  ^ (length(addr-out) = 32)
  ^ bvp(data-out)
  ^ (length(data-out) = 32)
  ^ boolp(hold-)
  ^ run-inputs-p(inputs, 1 + n)
  ^ if count < n then f
    else t endif)
→ (run-fm9001(list(list(regs,
    flags,
    a-reg,
    b-reg,
    i-reg,
    data-out,
    addr-out,
    t,
    t,
    hold-,
    pc-reg,
    cv_write3(f,
    last-regs-address,
    last-i-reg,
    last-flags,
    pc-reg)),
    list(mem, 2, clock, count, f, f, addr-out, data-out)),
    inputs,
    1 + n)
= list(list(write-regs(f, 0, regs, 0),
    flags,

```

```

    a-reg,
    b-reg,
    i-reg,
    data-out,
    addr-out,
    t,
    ¬ if n < count then f
      else t endif,
    hold-,
    pc-reg,
    cv_write3(f, pc-reg, i-reg, flags, pc-reg)),
list(mem,
    2,
    clock,
    count - (1 + n),
    if n < count then f
    else t endif,
    f,
    addr-out,
    data-out)))

```

THEOREM: write3\$progress
 (cv-hyps(t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ∧ cv-hyps(t, last-regs-address, i-reg, flags, pc-reg)
 ∧ memory-okp(32, 32, mem)
 ∧ regfile-okp(regs)
 ∧ bvp(a-reg)
 ∧ (length(a-reg) = 32)
 ∧ bvp(b-reg)
 ∧ (length(b-reg) = 32)
 ∧ bvp(addr-out)
 ∧ (length(addr-out) = 32)
 ∧ bvp(data-out)
 ∧ (length(data-out) = 32)
 ∧ boolp(hold-)
 ∧ run-inputs-p(inputs, count)
 ∧ (count ≠ 0))
 → (run-fm9001(list(list(regs,
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,

```

      t,
      t,
      hold-,
      pc-reg,
      cv_write3(f,
                  last-regs-address,
                  last-i-reg,
                  last-flags,
                  pc-reg)),
      list(mem, 2, clock, count - 1, f, f, addr-out, data-out)),
      inputs,
      count)
=   list(list(write-regs(f, 0, regs, 0),
              flags,
              a-reg,
              b-reg,
              i-reg,
              data-out,
              addr-out,
              t,
              f,
              hold-,
              pc-reg,
              cv_write3(f, pc-reg, i-reg, flags, pc-reg)),
          list(mem, 2, clock, 0, t, f, addr-out, data-out)))

```

DEFINITION:

```

zerop-not-zerop-cases(n)
=   if n ≈ 0 then t
    elseif n ≠ 0 then t
    else zerop-not-zerop-cases(n) endif

```

THEOREM: fetch0->fetch0\$sim0

```

(cv-hyps(t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ∧ cv-hyps(t, pc-reg, i-reg, flags, pc-reg)
 ∧ memory-okp(32, 32, mem)
 ∧ regfile-okp(regs)
 ∧ bvp(a-reg)
 ∧ (length(a-reg) = 32)
 ∧ bvp(b-reg)
 ∧ (length(b-reg) = 32)
 ∧ bvp(addr-out)
 ∧ (length(addr-out) = 32)
 ∧ bvp(data-out)

```

```

^ (length (data-out) = 32)
^ boolp (dtack-)
^ boolp (hold-)
^ run-inputs-p (inputs, 1)
^ (reset- = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_fetch0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg))),
                  list (mem,
                        0,
                        clock,
                        mem-count,
                        mem-dtack,
                        t,
                        last-address,
                        last-data))),
    inputs,
    1)
= list (list (write-regs (f, 0, regs, 0),
                      flags,
                      read-regs (pc-reg, regs),
                      b-reg,
                      i-reg,
                      data-out,
                      read-regs (pc-reg, regs),
                      t,
                      t,
                      f-buf (f-buf (hold-input (car (inputs))))),
                      pc-reg,
                      cv_fetch1 (t, pc-reg, i-reg, flags, pc-reg)),

```


list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

THEOREM: fetch0->fetch0\$sim1

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)

 ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)

 ^ memory-okp (32, 32, mem)

 ^ regfile-okp (regs)

 ^ bvp (a-reg)

 ^ (length (a-reg) = 32)

 ^ bvp (b-reg)

 ^ (length (b-reg) = 32)

 ^ bvp (addr-out)

 ^ (length (addr-out) = 32)

 ^ bvp (data-out)

 ^ (length (data-out) = 32)

 ^ boolp (dtack-)

 ^ boolp (hold-)

 ^ run-inputs-p (inputs, 1)

 ^ (reset- = t)

 ^ nil)

→ (run-fm9001 (list (list (regs,
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,
 reset-
 dtack-
 hold-
 pc-reg,
 cv_fetch0 (t,
 last-regs-address,
 last-i-reg,
 last-flags,
 pc-reg)),

list (mem,
 1,
 clock,
 mem-count,
 mem-dtack,
 t,
 last-address,
 last-data))),

```

      inputs,
      1)
=  list (list (write-regs (f, 0, regs, 0),
      flags,
      read-regs (pc-reg, regs),
      b-reg,
      i-reg,
      data-out,
      read-regs (pc-reg, regs),
      t,
      t,
      f-buf (f-buf (hold-input (car (inputs))))),
      pc-reg,
      cv_fetch1 (t, pc-reg, i-reg, flags, pc-reg)),
      list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

THEOREM: fetch0->fetch0\$sim2

```

(cv-hyps (f, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, 1)
  ^ (reset- = t)
  ^ (mem-count  $\simeq$  0))
→ (run-fm9001 (list (list (regs,
      flags,
      a-reg,
      b-reg,
      i-reg,
      data-out,
      addr-out,
      reset-,
      dtack-,
      hold-,

```

$$\begin{aligned}
& \text{pc-reg,} \\
& \text{cv\_fetch0}(\mathbf{f}, \\
& \quad \text{last-regs-address,} \\
& \quad \text{last-i-reg,} \\
& \quad \text{last-flags,} \\
& \quad \text{pc-reg}), \\
& \text{list}(\text{mem,} \\
& \quad 2, \\
& \quad \text{clock,} \\
& \quad \text{mem-count,} \\
& \quad \text{mem-dtack,} \\
& \quad \mathbf{f}, \\
& \quad \text{addr-out,} \\
& \quad \text{data-out}), \\
& \text{inputs,} \\
& 1) \\
= & \text{list}(\text{list}(\text{write-regs}(\mathbf{f}, 0, \text{regs}, 0), \\
& \quad \text{flags,} \\
& \quad \text{read-regs}(\text{pc-reg}, \text{regs}), \\
& \quad \text{b-reg,} \\
& \quad \text{i-reg,} \\
& \quad \text{data-out,} \\
& \quad \text{read-regs}(\text{pc-reg}, \text{regs}), \\
& \quad \mathbf{t}, \\
& \quad \mathbf{t}, \\
& \quad \text{f-buf}(\text{f-buf}(\text{hold-input}(\text{car}(\text{inputs})))), \\
& \quad \text{pc-reg,} \\
& \quad \text{cv\_fetch1}(\mathbf{t}, \text{pc-reg}, \text{i-reg}, \text{flags}, \text{pc-reg}), \\
& \text{list}(\text{write-mem}(\text{addr-out}, \text{mem}, \text{data-out}), \\
& \quad 0, \\
& \quad \text{clock,} \\
& \quad 0, \\
& \quad \mathbf{t}, \\
& \quad \mathbf{f}, \\
& \quad \text{addr-out,} \\
& \quad \text{data-out})))
\end{aligned}$$

THEOREM: fetch1->decode\$sim

$$\begin{aligned}
& ((\text{dtack-wait} = \text{car}(\text{clock})) \\
& \wedge \text{cv-hyps}(\text{last-rw-}, \text{last-regs-address}, \text{last-i-reg}, \text{last-flags}, \text{pc-reg}) \\
& \wedge \text{cv-hyps}(\mathbf{t}, \text{pc-reg}, \text{i-reg}, \text{flags}, \text{pc-reg}) \\
& \wedge \text{memory-okp}(32, 32, \text{mem}) \\
& \wedge \text{regfile-okp}(\text{regs}) \\
& \wedge \text{bvp}(\text{a-reg})
\end{aligned}$$

```

^ (length (a-reg) = 32)
^ bvp (b-reg)
^ (length (b-reg) = 32)
^ bvp (addr-out)
^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ boolp (dtack-)
^ boolp (hold-)
^ run-inputs-p (inputs, 1 + (1 + (car (clock) + 2)))
^ (reset- = t)
^ (hold- = t)
^ boolp (last-rw-)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_fetch1 (last-rw-,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg)),
                        list (mem,
                              0,
                              clock,
                              mem-count,
                              mem-dtack,
                              last-rw-,
                              last-address,
                              last-data))),
    inputs,
    1 + (1 + (car (clock) + 2)))
= list (list (write-regs (f,
                        0,
                        write-regs (t,
                                    pc-reg,
                                    write-regs (f, 0, regs, 0),

```

```

                                v-inc (a-reg)),
                                0),
    flags,
    a-reg,
    b-reg,
    read-mem (addr-out, mem),
    data-out,
    addr-out,
    t,
    t,
    t,
    pc-reg,
    if* (or* (store-resultp (store-cc (read-mem (addr-out,
                                                mem)),
                                                flags),
              set-some-flags (set-flags (read-mem (addr-out,
                                                mem)))),
        if* (or* (a-immediate-p (read-mem (addr-out, mem)),
                  reg-direct-p (mode-a (read-mem (addr-out,
                                                mem)))),
              if* (or* (reg-direct-p (mode-b (read-mem (addr-out,
                                                mem))),
                        unary-op-code-p (op-code (read-mem (addr-out,
                                                mem))))),
        cv_rega (t,
                  pc-reg,
                  read-mem (addr-out, mem),
                  flags,
                  pc-reg),
        cv_readb0 (t,
                   pc-reg,
                   read-mem (addr-out, mem),
                   flags,
                   pc-reg)),
        cv_reada0 (t,
                   pc-reg,
                   read-mem (addr-out, mem),
                   flags,
                   pc-reg)),
    if* (and* (not* (a-immediate-p (read-mem (addr-out,
                                                mem))),
              or* (pre-dec-p (mode-a (read-mem (addr-out,
                                                mem))),
                  post-inc-p (mode-a (read-mem (addr-out,

```

```

mem))))),
cv_sefa0 (t,
  pc-reg,
  read-mem (addr-out, mem),
  flags,
  pc-reg),
if* (or* (pre-dec-p (mode-b (read-mem (addr-out,
  mem))),
  post-inc-p (mode-b (read-mem (addr-out,
  mem))))),
cv_sefb0 (t,
  pc-reg,
  read-mem (addr-out, mem),
  flags,
  pc-reg),
cv_fetch0 (t,
  pc-reg,
  read-mem (addr-out, mem),
  flags,
  pc-reg))))),
list (mem,
  0,
  cdr (clock),
  0,
  t,
  t,
  addr-out,
  v-threefix (last-data))))

```

THEOREM: rega->rega\$sim
 (cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 ^ memory-okp (32, 32, mem)
 ^ regfile-okp (regs)
 ^ bvp (a-reg)
 ^ (length (a-reg) = 32)
 ^ bvp (b-reg)
 ^ (length (b-reg) = 32)
 ^ bvp (addr-out)
 ^ (length (addr-out) = 32)
 ^ bvp (data-out)
 ^ (length (data-out) = 32)
 ^ boolp (dtack-)
 ^ boolp (hold-))

```

^ run-inputs-p (inputs, 1)
^ (reset = t)
^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_rega (t,
                                   last-regs-address,
                                   last-i-reg,
                                   last-flags,
                                   pc-reg))),
                list (mem,
                    0,
                    clock,
                    mem-count,
                    mem-dtack,
                    t,
                    last-address,
                    last-data))),
    inputs,
    1)
= list (list (write-regs (f, 0, regs, 0),
                     flags,
                     if* (a-immediate-p (last-i-reg),
                        sign-extend (a-immediate (i-reg), 32),
                        read-regs (rn-a (last-i-reg), regs)),
                     b-reg,
                     i-reg,
                     data-out,
                     addr-out,
                     t,
                     t,
                     hold-,
                     pc-reg,
                     if* (reg-direct-p (mode-b (i-reg)),
                        if* (unary-op-code-p (op-code (i-reg)),

```

```

cv_update(t,
          rn-a(last-i-reg),
          i-reg,
          flags,
          pc-reg),
cv_regb(t,
        rn-a(last-i-reg),
        i-reg,
        flags,
        pc-reg)),
if* (store-resultp (store-cc (i-reg), flags),
    cv_write0(t,
              rn-a(last-i-reg),
              i-reg,
              flags,
              pc-reg),
    cv_update(t,
              rn-a(last-i-reg),
              i-reg,
              flags,
              pc-reg))),
list(mem, 0, clock, 0, t, t, addr-out, v-threefix(last-data))))

```

THEOREM: regb->update\$sim
 (cv-hyps(**t**, *last-regs-address*, *last-i-reg*, *last-flags*, *pc-reg*)
 ^ cv-hyps(**t**, *pc-reg*, *i-reg*, *flags*, *pc-reg*)
 ^ memory-okp(32, 32, *mem*)
 ^ regfile-okp(*regs*)
 ^ bvp(*a-reg*)
 ^ (length(*a-reg*) = 32)
 ^ bvp(*b-reg*)
 ^ (length(*b-reg*) = 32)
 ^ bvp(*addr-out*)
 ^ (length(*addr-out*) = 32)
 ^ bvp(*data-out*)
 ^ (length(*data-out*) = 32)
 ^ boolp(*dtack*-)
 ^ boolp(*hold*-)
 ^ run-inputs-p(*inputs*, 2)
 ^ (*reset* = **t**)
 ^ **nil**)
 → (run-fm9001(list(list(*regs*,
 flags,
 a-reg,


```

    b-reg,
    i-reg,
    data-out,
    addr-out,
    reset-,
    dtack-,
    hold-,
    pc-reg,
    cv_regb (t,
              last-regs-address,
              last-i-reg,
              last-flags,
              pc-reg)),
  list (mem,
        0,
        clock,
        mem-count,
        mem-dtack,
        t,
        last-address,
        last-data)),
  inputs,
  2)
= list (list (write-regs (store-resultp (store-cc (i-reg), flags),
    rn-b (i-reg),
    write-regs (f, 0, regs, 0),
    bv (v-alu (c-flag (flags),
              a-reg,
              read-regs (rn-b (last-i-reg), regs),
              op-code (i-reg))))),
  update-flags (flags,
    set-flags (i-reg),
    v-alu (c-flag (flags),
          a-reg,
          read-regs (rn-b (last-i-reg), regs),
          op-code (i-reg))),
  a-reg,
  read-regs (rn-b (i-reg), regs),
  i-reg,
  data-out,
  addr-out,
  t,
  t,
  hold-,

```

```

pc-reg,
if* (and* (or* (pre-dec-p (mode-b (i-reg))),
               post-inc-p (mode-b (i-reg))),
      unary-op-code-p (op-code (i-reg))),
cv_sefb1 (t, rn-b (i-reg), i-reg, flags, pc-reg),
cv_fetch0 (t, rn-b (i-reg), i-reg, flags, pc-reg)),
list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

THEOREM: update->update\$sim

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, 1)
  ^ (reset- = t)
  ^ ((mem-state = 0) ∨ (mem-state = 1)))
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_update (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg))),
                  list (mem,
                        mem-state,

```

```

                                clock,
                                mem-count,
                                mem-dtack,
                                t,
                                last-address,
                                last-data)),
                                inputs,
                                1)
=  list (list (write-regs (store-resultp (store-cc (last-i-reg),
                                                last-flags),
                                                rn-b (last-i-reg),
                                                regs,
                                                bv (v-alu (c-flag (last-flags),
                                                                a-reg,
                                                                b-reg,
                                                                op-code (last-i-reg))))),
                                update-flags (flags,
                                                set-flags (last-i-reg),
                                                v-alu (c-flag (last-flags),
                                                                a-reg,
                                                                b-reg,
                                                                op-code (last-i-reg))),
                                a-reg,
                                read-regs (rn-b (last-i-reg), regs),
                                i-reg,
                                data-out,
                                addr-out,
                                t,
                                t,
                                hold-,
                                pc-reg,
                                if* (and* (or* (pre-dec-p (mode-b (i-reg)),
                                                        post-inc-p (mode-b (i-reg))),
                                                unary-op-code-p (op-code (i-reg))),
                                cv_sefb1 (t, rn-b (last-i-reg), i-reg, flags, pc-reg),
                                cv_fetch0 (t, rn-b (last-i-reg), i-reg, flags, pc-reg))),
                                list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

THEOREM: reada0->reada3\$sim

```

((dtack-wait = car (clock))
 ∧  cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ∧  cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 ∧  memory-okp (32, 32, mem)
 ∧  regfile-okp (regs)

```

```

^  bvp(a-reg)
^  (length(a-reg) = 32)
^  bvp(b-reg)
^  (length(b-reg) = 32)
^  bvp(addr-out)
^  (length(addr-out) = 32)
^  bvp(data-out)
^  (length(data-out) = 32)
^  boolp(dtack-)
^  boolp(hold-)
^  run-inputs-p(inputs, 1 + (1 + (1 + (car(clock) + 1))))
^  (reset- = t)
^  nil
→  (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_reada0(t,
                                     last-regs-address,
                                     last-i-reg,
                                     last-flags,
                                     pc-reg))),
              list (mem,
                    0,
                    clock,
                    mem-count,
                    mem-dtack,
                    t,
                    last-address,
                    last-data))),
      inputs,
      1 + (1 + (1 + (car(clock) + 1))))
=  list (list (write-regs (f,
                          0,
                          write-regs (and* (not* (a-immediate-p (i-reg))),
                                          or* (pre-dec-p (mode-a (i-reg))),
                                          post-inc-p (mode-a (i-reg)))),

```

```

rn-a (i-reg),
write-regs (f, 0, regs, 0),
bv (core-alu (if* (pre-dec-p (mode-a (i-reg)),
                    carry-in-help (cons (c-flag (flags),
                                           cons (f,
                                                ALU-DEC-OP)))),
                    carry-in-help (cons (c-flag (flags),
                                           cons (f,
                                                ALU-INC-OP))))),
read-regs (rn-a (last-i-reg),
            regs),
b-reg,
f,
if* (pre-dec-p (mode-a (i-reg)),
     mpg (cons (f,
                 cons (f,
                      ALU-DEC-OP)))),
     mpg (cons (f,
                 cons (f,
                      ALU-INC-OP))))),
if* (pre-dec-p (mode-a (i-reg)),
     ALU-DEC-OP,
     ALU-INC-OP),
make-tree (32))))),
0),
flags,
read-mem (if* (pre-dec-p (mode-a (last-i-reg)),
                 v-dec (read-regs (rn-a (last-i-reg), regs)),
                 read-regs (rn-a (last-i-reg), regs)),
            mem),
read-regs (rn-b (i-reg),
            write-regs (and* (not* (a-immediate-p (i-reg)),
                                or* (pre-dec-p (mode-a (i-reg)),
                                       post-inc-p (mode-a (i-reg)))),
                        rn-a (i-reg),
                        write-regs (f, 0, regs, 0),
                        bv (core-alu (if* (pre-dec-p (mode-a (i-reg)),
                                                carry-in-help (cons (c-flag (flags),
                                                                           cons (f,
                                                                                ALU-DEC-OP)))),
                                                carry-in-help (cons (c-flag (flags),
                                                                           cons (f,
                                                                                ALU-INC-OP))))),
                        read-regs (rn-a (last-i-reg),

```

```

                                regs),
                                b-reg,
                                f,
                                if* (pre-dec-p (mode-a (i-reg)),
                                      mpg (cons (f,
                                                  cons (f,
                                                        ALU-DEC-OP))),
                                      mpg (cons (f,
                                                  cons (f,
                                                        ALU-INC-OP))))),
                                if* (pre-dec-p (mode-a (i-reg)),
                                      ALU-DEC-OP,
                                      ALU-INC-OP),
                                make-tree (32))))),
                                i-reg,
                                data-out,
                                if* (pre-dec-p (mode-a (last-i-reg)),
                                      v-dec (read-regs (rn-a (last-i-reg), regs)),
                                      read-regs (rn-a (last-i-reg), regs)),
                                t,
                                f,
                                hold-,
                                pc-reg,
                                if* (reg-direct-p (mode-b (i-reg)),
                                      cv_update (t, pc-reg, i-reg, flags, pc-reg),
                                      if* (unary-op-code-p (op-code (i-reg)),
                                            if* (store-resultp (store-cc (i-reg), flags),
                                                  cv_write0 (t, pc-reg, i-reg, flags, pc-reg),
                                                  cv_update (t, pc-reg, i-reg, flags, pc-reg)),
                                            cv_readb0 (t, pc-reg, i-reg, flags, pc-reg))),
                                list (mem,
                                      1,
                                      cdr (clock),
                                      0,
                                      t,
                                      t,
                                      if* (pre-dec-p (mode-a (last-i-reg)),
                                            v-dec (read-regs (rn-a (last-i-reg), regs)),
                                            read-regs (rn-a (last-i-reg), regs)),
                                      v-threefix (last-data))))

```

THEOREM: readb0->readb3\$sim

((dtack-wait = car (clock))

∧ cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)

```

^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
^ memory-okp (32, 32, mem)
^ regfile-okp (regs)
^ bvp (a-reg)
^ (length (a-reg) = 32)
^ bvp (b-reg)
^ (length (b-reg) = 32)
^ bvp (addr-out)
^ (length (addr-out) = 32)
^ bvp (data-out)
^ (length (data-out) = 32)
^ boolp (dtack-)
^ boolp (hold-)
^ run-inputs-p (inputs, 1 + (1 + (1 + (car (clock) + 1))))
^ (reset- = t)
^ ((mem-state = 0) ∨ (mem-state = 1)))
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_readb0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg))),
      list (mem,
            mem-state,
            clock,
            mem-count,
            mem-dtack,
            t,
            last-address,
            last-data))),
      inputs,
      1 + (1 + (1 + (car (clock) + 1))))
= list (list (write-regs (f,
                          0,

```

```

write-regs (and* (not* (store-resultp (store-cc (i-reg),
                                                flags)),
                 or* (pre-dec-p (mode-b (i-reg)),
                     post-inc-p (mode-b (i-reg)))),
            rn-b (i-reg),
            write-regs (f, 0, regs, 0),
            bv (core-alu (if* (pre-dec-p (mode-b (i-reg)),
                                     carry-in-help (cons (c-flag (flags),
                                                             cons (f,
                                                                    ALU-DEC-OP))),
                             carry-in-help (cons (c-flag (flags),
                                                             cons (f,
                                                                    ALU-INC-OP))),
                if* (or* (a-immediate-p (i-reg),
                               reg-direct-p (mode-a (i-reg))),
                    if* (a-immediate-p (i-reg),
                        sign-extend (a-immediate (i-reg),
                                      32),
                        read-regs (rn-a (i-reg),
                                    regs)),
                    a-reg),
                read-regs (rn-b (last-i-reg),
                            regs),
                f,
                if* (pre-dec-p (mode-b (i-reg)),
                    mpg (cons (f,
                                cons (t,
                                       ALU-DEC-OP))),
                    mpg (cons (f,
                                cons (t,
                                       ALU-INC-OP))))),
                if* (pre-dec-p (mode-b (i-reg)),
                    ALU-DEC-OP,
                    ALU-INC-OP),
                make-tree (32))))),
0),
flags,
if* (or* (a-immediate-p (i-reg),
                        reg-direct-p (mode-a (i-reg))),
    if* (a-immediate-p (i-reg),
        sign-extend (a-immediate (i-reg), 32),
        read-regs (rn-a (i-reg), regs)),
    a-reg),
read-mem (if* (pre-dec-p (mode-b (last-i-reg)),

```



```

                                v-dec (read-regs (rn-b (last-i-reg), regs)),
                                read-regs (rn-b (last-i-reg), regs)),
                                mem),
    i-reg,
    data-out,
    if* (pre-dec-p (mode-b (last-i-reg)),
        v-dec (read-regs (rn-b (last-i-reg), regs)),
        read-regs (rn-b (last-i-reg), regs)),
    t,
    f,
    hold-,
    pc-reg,
    if* (store-resultp (store-cc (i-reg), flags),
        cv-write0 (t, pc-reg, i-reg, flags, pc-reg),
        cv-update (t, pc-reg, i-reg, flags, pc-reg))),
    list (mem,
        1,
        cdr (clock),
        0,
        t,
        t,
        if* (pre-dec-p (mode-b (last-i-reg)),
            v-dec (read-regs (rn-b (last-i-reg), regs)),
            read-regs (rn-b (last-i-reg), regs)),
        v-threefix (last-data)))

```

THEOREM: write0->write3\$sim

```

((dtack-wait = car (clock))
 ^  cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
 ^  cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
 ^  memory-okp (32, 32, mem)
 ^  regfile-okp (regs)
 ^  bvp (a-reg)
 ^  (length (a-reg) = 32)
 ^  bvp (b-reg)
 ^  (length (b-reg) = 32)
 ^  bvp (addr-out)
 ^  (length (addr-out) = 32)
 ^  bvp (data-out)
 ^  (length (data-out) = 32)
 ^  boolp (dtack-)
 ^  boolp (hold-)
 ^  run-inputs-p (inputs, 1 + (1 + (1 + (car (clock) + 1))))
 ^  (reset- = t)

```

```

 $\wedge$  ((mem-state = 0)  $\vee$  (mem-state = 1)))
 $\rightarrow$  (run-fin9001 (list (list (regs,
                             flags,
                             a-reg,
                             b-reg,
                             i-reg,
                             data-out,
                             addr-out,
                             reset-,
                             dtack-,
                             hold-,
                             pc-reg,
                             cv_write0 (t,
                                         last-regs-address,
                                         last-i-reg,
                                         last-flags,
                                         pc-reg)),
                        list (mem,
                              mem-state,
                              clock,
                              mem-count,
                              mem-dtack,
                              t,
                              last-address,
                              last-data)),
    inputs,
    1 + (1 + (1 + (car (clock) + 1))))
= list (list (write-regs (f,
                        0,
                        write-regs (or* (pre-dec-p (mode-b (i-reg)),
                                             post-inc-p (mode-b (i-reg))),
                        rn-b (i-reg),
                        write-regs (f, 0, regs, 0),
                        bv (core-alu (if* (pre-dec-p (mode-b (i-reg)),
                                                  carry-in-help (cons (c-flag (flags),
                                                                      cons (f,
                                                                      ALU-DEC-OP))),
                                                  carry-in-help (cons (c-flag (flags),
                                                                      cons (f,
                                                                      ALU-INC-OP))))),
                        a-reg,
                        read-regs (rn-b (last-i-reg),
                                     regs),
                        f,

```

```

                                if* (pre-dec-p (mode-b (i-reg)),
                                    mpg (cons (f,
                                                cons (t,
                                                    ALU-DEC-OP))),
                                    mpg (cons (f,
                                                cons (t,
                                                    ALU-INC-OP))))),
                                if* (pre-dec-p (mode-b (i-reg)),
                                    ALU-DEC-OP,
                                    ALU-INC-OP),
                                make-tree (32))),
                                0),
    update-flags (flags,
                  set-flags (last-i-reg),
                  v-alu (c-flag (last-flags),
                        a-reg,
                        b-reg,
                        op-code (last-i-reg))),
    a-reg,
    read-regs (rn-b (last-i-reg), regs),
    i-reg,
    bv (v-alu (c-flag (last-flags),
                a-reg,
                b-reg,
                op-code (last-i-reg))),
    if* (pre-dec-p (mode-b (last-i-reg)),
        v-dec (read-regs (rn-b (last-i-reg), regs)),
        read-regs (rn-b (last-i-reg), regs)),
    t,
    f,
    hold-,
    pc-reg,
    cv_fetch0 (f,
                pc-reg,
                i-reg,
                update-flags (flags,
                              set-flags (last-i-reg),
                              v-alu (c-flag (last-flags),
                                      a-reg,
                                      b-reg,
                                      op-code (last-i-reg))),
                              pc-reg)),
    list (mem,
          2,

```

```

cdr (clock),
0,
t,
f,
if* (pre-dec-p (mode-b (last-i-reg)),
      v-dec (read-regs (rn-b (last-i-reg), regs)),
      read-regs (rn-b (last-i-reg), regs)),
bv (v-alu (c-flag (last-flags),
            a-reg,
            b-reg,
            op-code (last-i-reg))))))

```

THEOREM: sefa0->sefa1\$sim

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, 2)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_sefa0 (t,
                                       last-regs-address,
                                       last-i-reg,

```

```

                                last-flags,
                                pc-reg)),
list (mem,
      0,
      clock,
      mem-count,
      mem-dtack,
      t,
      last-address,
      last-data)),
inputs,
2)
= list (list (write-regs (t,
                        rn-a (i-reg),
                        write-regs (f, 0, regs, 0),
                        bv (core-alu (if* (pre-dec-p (mode-a (i-reg)),
                                                carry-in-help (cons (c-flag (flags),
                                                                    cons (f,
                                                                    ALU-DEC-OP))),
                                                carry-in-help (cons (c-flag (flags),
                                                                    cons (f,
                                                                    ALU-INC-OP))))),
                        read-regs (rn-a (last-i-reg),
                                    regs),
                        b-reg,
                        f,
                        if* (pre-dec-p (mode-a (i-reg)),
                            mpg (cons (f,
                                        cons (f,
                                        ALU-DEC-OP))),
                            mpg (cons (f,
                                        cons (f,
                                        ALU-INC-OP))))),
                        if* (pre-dec-p (mode-a (i-reg)),
                            ALU-DEC-OP,
                            ALU-INC-OP),
                        make-tree (32))))),
flags,
read-regs (rn-a (last-i-reg), regs),
b-reg,
i-reg,
data-out,
addr-out,
t,

```

```

    t,
    hold-,
    pc-reg,
    if* (or* (pre-dec-p (mode-b (i-reg)),
               post-inc-p (mode-b (i-reg))),
         cv_sefb0 (t, rn-a (i-reg), i-reg, flags, pc-reg),
         cv_fetch0 (t, rn-a (i-reg), i-reg, flags, pc-reg))),
    list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

THEOREM: sefb0->sefb1\$sim

```

(cv-hyps (t, last-regs-address, last-i-reg, last-flags, pc-reg)
  ^ cv-hyps (t, pc-reg, i-reg, flags, pc-reg)
  ^ memory-okp (32, 32, mem)
  ^ regfile-okp (regs)
  ^ bvp (a-reg)
  ^ (length (a-reg) = 32)
  ^ bvp (b-reg)
  ^ (length (b-reg) = 32)
  ^ bvp (addr-out)
  ^ (length (addr-out) = 32)
  ^ bvp (data-out)
  ^ (length (data-out) = 32)
  ^ boolp (dtack-)
  ^ boolp (hold-)
  ^ run-inputs-p (inputs, 2)
  ^ (reset- = t)
  ^ nil)
→ (run-fm9001 (list (list (regs,
                           flags,
                           a-reg,
                           b-reg,
                           i-reg,
                           data-out,
                           addr-out,
                           reset-,
                           dtack-,
                           hold-,
                           pc-reg,
                           cv_sefb0 (t,
                                       last-regs-address,
                                       last-i-reg,
                                       last-flags,
                                       pc-reg))),
                  list (mem,

```

```

0,
clock,
mem-count,
mem-dtack,
t,
last-address,
last-data)),
inputs,
2)
= list (list (write-regs (t,
rn-b (i-reg),
write-regs (f, 0, regs, 0),
bv (core-alu (if* (pre-dec-p (mode-b (i-reg)),
carry-in-help (cons (c-flag (flags),
cons (f,
ALU-DEC-OP))),
carry-in-help (cons (c-flag (flags),
cons (f,
ALU-INC-OP))))),
a-reg,
read-regs (rn-b (last-i-reg),
regs),
f,
if* (pre-dec-p (mode-b (i-reg)),
mpg (cons (f,
cons (t,
ALU-DEC-OP))),
mpg (cons (f,
cons (t,
ALU-INC-OP))))),
if* (pre-dec-p (mode-b (i-reg)),
ALU-DEC-OP,
ALU-INC-OP),
make-tree (32))))),
flags,
a-reg,
read-regs (rn-b (last-i-reg), regs),
i-reg,
data-out,
addr-out,
t,
t,
hold-,
pc-reg,

```

$$\text{cv\_fetch0}(\mathbf{t}, \text{rn-b}(i\text{-reg}), i\text{-reg}, \text{flags}, \text{pc-reg}),$$

$$\text{list}(\text{mem}, 0, \text{clock}, 0, \mathbf{t}, \mathbf{t}, \text{addr-out}, \text{v-threefix}(\text{last-data}))))$$

THEOREM: sefb1->sefb1\$sim

$$(\text{cv-hyps}(\mathbf{t}, \text{last-regs-address}, \text{last-i-reg}, \text{last-flags}, \text{pc-reg})$$

$$\wedge \text{cv-hyps}(\mathbf{t}, \text{pc-reg}, i\text{-reg}, \text{flags}, \text{pc-reg})$$

$$\wedge \text{memory-okp}(32, 32, \text{mem})$$

$$\wedge \text{regfile-okp}(\text{regs})$$

$$\wedge \text{bvp}(a\text{-reg})$$

$$\wedge (\text{length}(a\text{-reg}) = 32)$$

$$\wedge \text{bvp}(b\text{-reg})$$

$$\wedge (\text{length}(b\text{-reg}) = 32)$$

$$\wedge \text{bvp}(\text{addr-out})$$

$$\wedge (\text{length}(\text{addr-out}) = 32)$$

$$\wedge \text{bvp}(\text{data-out})$$

$$\wedge (\text{length}(\text{data-out}) = 32)$$

$$\wedge \text{boolp}(\text{dtack-})$$

$$\wedge \text{boolp}(\text{hold-})$$

$$\wedge \text{run-inputs-p}(\text{inputs}, 1)$$

$$\wedge (\text{reset-} = \mathbf{t})$$

$$\wedge \mathbf{nil})$$

$$\rightarrow (\text{run-fm9001}(\text{list}(\text{list}(\text{regs},$$

$$\text{flags},$$

$$a\text{-reg},$$

$$b\text{-reg},$$

$$i\text{-reg},$$

$$\text{data-out},$$

$$\text{addr-out},$$

$$\text{reset-},$$

$$\text{dtack-},$$

$$\text{hold-},$$

$$\text{pc-reg},$$

$$\text{cv\_sefb1}(\mathbf{t},$$

$$\text{last-regs-address},$$

$$\text{last-i-reg},$$

$$\text{last-flags},$$

$$\text{pc-reg}))),$$

$$\text{list}(\text{mem},$$

$$0,$$

$$\text{clock},$$

$$\text{mem-count},$$

$$\text{mem-dtack},$$

$$\mathbf{t},$$

$$\text{last-address},$$


```

                                last-data)),
                                inputs,
                                1)
=  list (list (write-regs (t,
                                rn-b (last-i-reg),
                                regs,
                                bv (core-alu (if* (pre-dec-p (mode-b (last-i-reg)),
                                                carry-in-help (cons (c-flag (last-flags),
                                                                cons (f,
                                                                    ALU-DEC-OP))),
                                                carry-in-help (cons (c-flag (last-flags),
                                                                cons (f,
                                                                    ALU-INC-OP))))),
                                a-reg,
                                b-reg,
                                f,
                                if* (pre-dec-p (mode-b (last-i-reg)),
                                    mpg (cons (f,
                                                cons (t,
                                                    ALU-DEC-OP))),
                                    mpg (cons (f,
                                                cons (t,
                                                    ALU-INC-OP))))),
                                if* (pre-dec-p (mode-b (last-i-reg)),
                                    ALU-DEC-OP,
                                    ALU-INC-OP),
                                make-tree (32))))),
                                flags,
                                a-reg,
                                b-reg,
                                i-reg,
                                data-out,
                                addr-out,
                                t,
                                t,
                                hold-,
                                pc-reg,
                                cv_fetch0 (t, rn-b (last-i-reg), i-reg, flags, pc-reg)),
                                list (mem, 0, clock, 0, t, t, addr-out, v-threefix (last-data))))

```

DEFINITION: $t\_fetch0 (clock, i-reg, flags) = 1$

DEFINITION:

$t\_sefb1 (clock, i-reg, flags) = (1 + t\_fetch0 (clock, i-reg, flags))$

DEFINITION:

$t\_sefb0(clock, i-reg, flags) = (2 + t\_fetch0(clock, i-reg, flags))$

DEFINITION:

$t\_sefa0(clock, i-reg, flags)$
= $(2 + \text{if } or^*(pre-dec-p(mode-b(i-reg)), post-inc-p(mode-b(i-reg))))$
 then $t\_sefb0(clock, i-reg, flags)$
 else $t\_fetch0(clock, i-reg, flags)$ **endif**

DEFINITION:

$t\_write0(clock, i-reg, flags)$
= $((1 + (1 + (1 + (car(clock) + 1))))$
 + $t\_fetch0(cdr(clock), i-reg, flags)$)

DEFINITION:

$t\_update(clock, i-reg, flags)$
= $(1 + \text{if } b\text{-and}(or^*(pre-dec-p(mode-b(i-reg)),$
 $post-inc-p(mode-b(i-reg))),$
 $unary-op-code-p(op-code(i-reg))))$
 then $t\_sefb1(clock, i-reg, flags)$
 else $t\_fetch0(clock, i-reg, flags)$ **endif**

DEFINITION:

$t\_readb0(clock, i-reg, flags)$
= $((1 + (1 + (1 + (car(clock) + 1))))$
 + $\text{if } store\text{-resultp}(store\text{-cc}(i-reg), flags)$
 then $t\_write0(cdr(clock), i-reg, flags)$
 else $t\_update(cdr(clock), i-reg, flags)$ **endif**)

DEFINITION:

$t\_reada0(clock, i-reg, flags)$
= $((1 + (1 + (1 + (car(clock) + 1))))$
 + $\text{if } reg\text{-direct-p}(mode-b(i-reg))$
 then $t\_update(cdr(clock), i-reg, flags)$
 elseif $unary\text{-op-code-p}(op-code(i-reg))$
 then if $store\text{-resultp}(store\text{-cc}(i-reg), flags)$
 then $t\_write0(cdr(clock), i-reg, flags)$
 else $t\_update(cdr(clock), i-reg, flags)$ **endif**
 else $t\_readb0(cdr(clock), i-reg, flags)$ **endif**)

DEFINITION:

$t\_regb(clock, i-reg, flags)$
= $(2 + \text{if } b\text{-and}(or^*(pre-dec-p(mode-b(i-reg)),$
 $post-inc-p(mode-b(i-reg))),$
 $unary-op-code-p(op-code(i-reg))))$
 then $t\_sefb1(clock, i-reg, flags)$
 else $t\_fetch0(clock, i-reg, flags)$ **endif**)

DEFINITION:

```
t_rega (clock, i-reg, flags)
= (1 + if reg-direct-p (mode-b (i-reg))
    then if unary-op-code-p (op-code (i-reg))
        then t_update (clock, i-reg, flags)
        else t_regb (clock, i-reg, flags) endif
    elseif store-result-p (store-cc (i-reg), flags)
    then t_write0 (clock, i-reg, flags)
    else t_update (clock, i-reg, flags) endif)
```

DEFINITION:

```
t_fetch1 (clock, i-reg, flags)
= ((1 + (1 + (car (clock) + 2)))
   + if b-or (store-result-p (store-cc (i-reg), flags),
              set-some-flags (set-flags (i-reg)))
   then if or* (a-immediate-p (i-reg), reg-direct-p (mode-a (i-reg)))
       then if b-or (reg-direct-p (mode-b (i-reg)),
                    unary-op-code-p (op-code (i-reg)))
           then t_rega (cdr (clock), i-reg, flags)
           else t_readb0 (cdr (clock), i-reg, flags) endif
       else t_reada0 (cdr (clock), i-reg, flags) endif
   elseif and* (not* (a-immediate-p (i-reg)),
                or* (pre-dec-p (mode-a (i-reg)),
                    post-inc-p (mode-a (i-reg))))
   then t_sefa0 (cdr (clock), i-reg, flags)
   elseif or* (pre-dec-p (mode-b (i-reg)), post-inc-p (mode-b (i-reg)))
   then t_sefb0 (cdr (clock), i-reg, flags)
   else t_fetch0 (cdr (clock), i-reg, flags) endif)
```

DEFINITION:

```
operating-inputs-p (inputs, n)
= if n  $\simeq$  0 then t
  else (reset-input (car (inputs)) = t)
     $\wedge$  (hold-input (car (inputs)) = t)
     $\wedge$  bvp (pc-reg-input (car (inputs)))
     $\wedge$  (length (pc-reg-input (car (inputs))) = 4)
     $\wedge$  operating-inputs-p (cdr (inputs), n - 1) endif
```

EVENT: Enable operating-inputs-p; name this event ‘operating-inputs-p-off’.

THEOREM: open-operating-inputs-p

```
((n  $\simeq$  0)  $\rightarrow$  (operating-inputs-p (inputs, n) = t))
 $\wedge$  ((n  $\not\simeq$  0)
 $\rightarrow$  (operating-inputs-p (inputs, n)
```

$$\begin{aligned}
= & ((\text{reset-input}(\text{car}(\text{inputs})) = \mathbf{t}) \\
& \wedge (\text{hold-input}(\text{car}(\text{inputs})) = \mathbf{t}) \\
& \wedge \text{bvp}(\text{pc-reg-input}(\text{car}(\text{inputs}))) \\
& \wedge (\text{length}(\text{pc-reg-input}(\text{car}(\text{inputs}))) = 4) \\
& \wedge \text{operating-inputs-p}(\text{cdr}(\text{inputs}), n - 1)))
\end{aligned}$$

THEOREM: operating-inputs-p-implies-run-inputs-p
 $\text{operating-inputs-p}(\text{inputs}, n) \rightarrow \text{run-inputs-p}(\text{inputs}, n)$

THEOREM: operating-inputs-p-plus
 $\text{operating-inputs-p}(\text{inputs}, n + m)$
 $= (\text{operating-inputs-p}(\text{inputs}, n)$
 $\quad \wedge \text{operating-inputs-p}(\text{nthcdr}(n, \text{inputs}), m))$

THEOREM: operating-inputs-p-1
 $(\text{operating-inputs-p}(\text{inputs}, n) \wedge (n \neq 0))$
 $\rightarrow \text{operating-inputs-p}(\text{inputs}, 1)$

THEOREM: operating-inputs-p-implies-hold-input
 $(\text{operating-inputs-p}(\text{inputs}, n) \wedge (n \neq 0))$
 $\rightarrow (\text{hold-input}(\text{car}(\text{inputs})) = \mathbf{t})$

DEFINITION:
 $\text{microcycles}(\text{state})$
 $= \text{t\_fetch1}(\text{caddadr}(\text{state}),$
 $\quad \text{read-mem}(\text{read-mem}(\text{cadddddddardar}(\text{state}), \text{caaar}(\text{state})),$
 $\quad \text{caadr}(\text{state})),$
 $\quad \text{cadar}(\text{state}))$

DEFINITION:
 $\text{total-microcycles}(\text{state}, \text{inputs}, n)$
 $= \text{if } n \simeq 0 \text{ then } 0$
 $\quad \text{else } \text{microcycles}(\text{state})$
 $\quad + \text{total-microcycles}(\text{run-fm9001}(\text{state},$
 $\quad \quad \text{inputs},$
 $\quad \quad \text{microcycles}(\text{state})),$
 $\quad \text{nthcdr}(\text{microcycles}(\text{state}), \text{inputs}),$
 $\quad n - 1) \text{ endif}$

EVENT: Enable total-microcycles; name this event ‘total-microcycles-off’.

THEOREM: open-total-microcycles
 $((n \simeq 0) \rightarrow (\text{total-microcycles}(\text{state}, \text{inputs}, n) = 0))$
 $\wedge ((n \neq 0)$
 $\quad \rightarrow (\text{total-microcycles}(\text{state}, \text{inputs}, n)$

$$\begin{aligned}
&= \text{microcycles}(state) \\
&\quad + \text{total-microcycles}(\text{run-fm9001}(state, \\
&\quad \quad \quad inputs, \\
&\quad \quad \quad \text{microcycles}(state)), \\
&\quad \quad \quad \text{nthcdr}(\text{microcycles}(state), \\
&\quad \quad \quad inputs), \\
&\quad \quad \quad n - 1)))
\end{aligned}$$

DEFINITION:

macrocycle-invariant(*state*, *pc*)

$$\begin{aligned}
&= (\text{all-ramp-mem}(4, \text{caaar}(state)) \\
&\quad \wedge \text{memory-okp}(4, 32, \text{caaar}(state)) \\
&\quad \wedge (\text{cadaar}(state) = \mathbf{f}) \\
&\quad \wedge \text{bvp}(\text{caddaar}(state)) \\
&\quad \wedge (\text{length}(\text{caddaar}(state)) = 32) \\
&\quad \wedge \text{bvp}(\text{cadddaar}(state)) \\
&\quad \wedge (\text{length}(\text{cadddaar}(state)) = 4) \\
&\quad \wedge \text{bvp}(\text{cadar}(state)) \\
&\quad \wedge (\text{length}(\text{cadar}(state)) = 4) \\
&\quad \wedge (\text{caddar}(state) \\
&\quad \quad = \text{read-regs}(\text{cadddddddardar}(state), \text{caar}(state))) \\
&\quad \wedge \text{bvp}(\text{cadddar}(state)) \\
&\quad \wedge (\text{length}(\text{cadddar}(state)) = 32) \\
&\quad \wedge \text{bvp}(\text{caddddar}(state)) \\
&\quad \wedge (\text{length}(\text{caddddar}(state)) = 32) \\
&\quad \wedge \text{bvp}(\text{cadddddar}(state)) \\
&\quad \wedge (\text{length}(\text{cadddddar}(state)) = 32) \\
&\quad \wedge (\text{cadddddardar}(state) \\
&\quad \quad = \text{read-regs}(\text{cadddddddardar}(state), \text{caar}(state))) \\
&\quad \wedge (\text{cadddddardar}(state) = \mathbf{t}) \\
&\quad \wedge \text{boolp}(\text{cadddddddardar}(state)) \\
&\quad \wedge (\text{cadddddddardar}(state) = \mathbf{t}) \\
&\quad \wedge \text{bvp}(\text{cadddddddardar}(state)) \\
&\quad \wedge (\text{length}(\text{cadddddddardar}(state)) = 4) \\
&\quad \wedge (\text{cadddddddardar}(state) = pc) \\
&\quad \wedge (\text{cadddddddardar}(state) \\
&\quad \quad = \text{cv\_fetch1}(\text{cadddddadr}(state), \\
&\quad \quad \quad \text{cadddddddardar}(state), \\
&\quad \quad \quad \text{caddddar}(state), \\
&\quad \quad \quad \text{cadar}(state), \\
&\quad \quad \quad \text{cadddddddardar}(state))) \\
&\quad \wedge \text{memory-okp}(32, 32, \text{caadr}(state)) \\
&\quad \wedge (\text{cadadr}(state) = 0) \\
&\quad \wedge (\text{cadddadr}(state) \in \mathbf{N})
\end{aligned}$$

$\wedge$ boolp (cadddddadr (state))
 $\wedge$ bvp (cadddddadr (state))
 $\wedge$ (length (cadddddadr (state)) = 32)
 $\wedge$ properp (cadddddadr (state))
 $\wedge$ (length (cadddddadr (state)) = 32))

THEOREM: macrocycle-invariant $\Rightarrow$ chip-system-invariant\$help

macrocycle-invariant (list (list (list (regs-regs, regs-we, regs-data, regs-address),
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,
 last-reset-,
 last-dtack-,
 last-hold-,
 pc-reg,
 cntl-state),
 list (mem,
 cntl,
 clock,
 count,
 dtack,
 last-rw-,
 last-address,
 last-data)),
 pc)
 $\rightarrow$ chip-system-invariant (list (list (list (regs-regs,
 regs-we,
 regs-data,
 regs-address),
 flags,
 a-reg,
 b-reg,
 i-reg,
 data-out,
 addr-out,
 last-reset-,
 last-dtack-,
 last-hold-,
 pc-reg,
 cntl-state),
 list (mem,

cntl,
clock,
count,
dtack,
last-rw-,
last-address,
last-data)))

THEOREM: $\text{macrocycle-invariant} \Rightarrow \text{chip-system-invariant}$
 $(\text{fm9001-state-structure}(\text{state}) \wedge \text{macrocycle-invariant}(\text{state}, \text{pc}))$
 $\rightarrow \text{chip-system-invariant}(\text{state})$

DEFINITION:

$\text{macrocycle-invariant}^*(\text{state}, \text{pc}) = \text{macrocycle-invariant}(\text{state}, \text{pc})$

EVENT: Enable $\text{macrocycle-invariant}^*$; name this event ‘ $\text{macrocycle-invariant}^*$ -off’.

THEOREM: $\text{macrocycle-invariant}^* = \text{macrocycle-invariant}$
 $\text{macrocycle-invariant}^*(\text{cons}(x, y), \text{pc}) = \text{macrocycle-invariant}(\text{cons}(x, y), \text{pc})$

THEOREM: $\text{macrocycle-invariant-is-invariant}$ \$help

$(\text{macrocycle-invariant}(\text{list}(\text{list}(\text{list}(\text{regs-regs},$
 $\text{regs-we},$
 $\text{regs-data},$
 $\text{regs-address}),$

$\text{flags},$
 $\text{a-reg},$
 $\text{b-reg},$
 $\text{i-reg},$
 $\text{data-out},$
 $\text{addr-out},$
 $\text{last-reset},$
 $\text{last-dtack},$
 $\text{last-hold},$
 $\text{pc-reg},$
 $\text{cntl-state}),$

$\text{list}(\text{mem},$
 $\text{cntl},$
 $\text{clock},$
 $\text{count},$
 $\text{dtack},$
 $\text{last-rw},$
 $\text{last-address},$

```

                                last-data)),
                                pc)
 $\wedge$  operating-inputs-p (inputs,
                        microcycles (list (list (list (regs-regs,
                                                    regs-we,
                                                    regs-data,
                                                    regs-address),
                                                    flags,
                                                    a-reg,
                                                    b-reg,
                                                    i-reg,
                                                    data-out,
                                                    addr-out,
                                                    last-reset-,
                                                    last-dtack-,
                                                    last-hold-,
                                                    pc-reg,
                                                    cntl-state),
list (mem,
      cntl,
      clock,
      count,
      dtack,
      last-rw-,
      last-address,
      last-data))))))
 $\rightarrow$  macrocycle-invariant* (run-fm9001 (list (list (list (regs-regs,
                                                    regs-we,
                                                    regs-data,
                                                    regs-address),
                                                    flags,
                                                    a-reg,
                                                    b-reg,
                                                    i-reg,
                                                    data-out,
                                                    addr-out,
                                                    last-reset-,
                                                    last-dtack-,
                                                    last-hold-,
                                                    pc-reg,
                                                    cntl-state),
list (mem,
      cntl,
      clock,
```


$count,$
 $dtack,$
 $last-rw-,$
 $last-address,$
 $last-data)),$
 $inputs,$
 $microcycles (list (list (list (regs-regs,$
 $regs-we,$
 $regs-data,$
 $regs-address),$
 $flags,$
 $a-reg,$
 $b-reg,$
 $i-reg,$
 $data-out,$
 $addr-out,$
 $last-reset-,$
 $last-dtack-,$
 $last-hold-,$
 $pc-reg,$
 $cntl-state),$
 $list (mem,$
 $cntl,$
 $clock,$
 $count,$
 $dtack,$
 $last-rw-,$
 $last-address,$
 $last-data))))),$
 $pc)$

THEOREM: macrocycle-invariant-is-invariant
 $(fm9001-state-structure (state)$
 $\wedge$ macrocycle-invariant $(state, pc)$
 $\wedge$ operating-inputs-p $(inputs, microcycles (state))$
 $\rightarrow$ macrocycle-invariant $(run-fm9001 (state, inputs, microcycles (state)), pc)$

DEFINITION:
 $p-map-up (p-state) = list (car (regs (p-state)), flags (p-state))$

DEFINITION: $mem-map-up (mem-state) = car (mem-state)$

DEFINITION:
 $map-up (state) = list (p-map-up (car (state)), mem-map-up (cadr (state)))$

EVENT: Enable map-up; name this event ‘map-up-off’.

THEOREM: open-map-up
 $\text{map-up}(\text{cons}(x, y))$
 $= \text{list}(\text{list}(\text{car}(\text{regs}(\text{car}(\text{cons}(x, y)))), \text{flags}(\text{car}(\text{cons}(x, y)))),$
 $\text{caadr}(\text{cons}(x, y)))$

DEFINITION:
 $\text{fm9001-step}^*(\text{state}, \text{pc-reg}) = \text{fm9001-step}(\text{state}, \text{pc-reg})$

EVENT: Enable fm9001-step*; name this event ‘fm9001-step*-off’.

THEOREM: fm9001-step*-lemma
 $(\text{cons}(x, y) = \text{fm9001-step}^*(\text{state}, \text{pc-reg}))$
 $= (\text{cons}(x, y) = \text{fm9001-step}(\text{state}, \text{pc-reg}))$

THEOREM: middle=high\$help
 $(\text{macrocycle-invariant}(\text{list}(\text{list}(\text{list}(\text{regs-regs},$
 $\text{regs-we},$
 $\text{regs-data},$
 $\text{regs-address}),$
 $\text{flags},$
 $\text{a-reg},$
 $\text{b-reg},$
 $\text{i-reg},$
 $\text{data-out},$
 $\text{addr-out},$
 $\text{last-reset-},$
 $\text{last-dtack-},$
 $\text{last-hold-},$
 $\text{pc-reg},$
 $\text{cntl-state}),$
 $\text{list}(\text{mem},$
 $\text{cntl},$
 $\text{clock},$
 $\text{count},$
 $\text{dtack},$
 $\text{last-rw-},$
 $\text{last-address},$
 $\text{last-data}),$
 $\text{pc-reg})$
 $\wedge \text{operating-inputs-p}(\text{inputs},$
 $\text{microcycles}(\text{list}(\text{list}(\text{list}(\text{regs-regs},$
 $\text{regs-we},$

```

regs-data,
regs-address),
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
last-reset-,
last-dtack-,
last-hold-,
pc-reg,
cntl-state),
list (mem,
cntl,
clock,
count,
dtack,
last-rw-,
last-address,
last-data))))
→ (map-up (run-fm9001 (list (list (list (regs-regs,
regs-we,
regs-data,
regs-address),
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
last-reset-,
last-dtack-,
last-hold-,
pc-reg,
cntl-state),
list (mem,
cntl,
clock,
count,
dtack,
last-rw-,
last-address,
last-data))),

```

$$\begin{aligned}
& \text{inputs,} \\
& \text{microcycles (list (list (list (regs-regs,} \\
& \hspace{10em} \text{regs-we,} \\
& \hspace{10em} \text{regs-data,} \\
& \hspace{10em} \text{regs-address}),} \\
& \hspace{4em} \text{flags,} \\
& \hspace{4em} \text{a-reg,} \\
& \hspace{4em} \text{b-reg,} \\
& \hspace{4em} \text{i-reg,} \\
& \hspace{4em} \text{data-out,} \\
& \hspace{4em} \text{addr-out,} \\
& \hspace{4em} \text{last-reset-,} \\
& \hspace{4em} \text{last-dtack-,} \\
& \hspace{4em} \text{last-hold-,} \\
& \hspace{4em} \text{pc-reg,} \\
& \hspace{4em} \text{cntl-state}),} \\
& \text{list (mem,} \\
& \hspace{4em} \text{cntl,} \\
& \hspace{4em} \text{clock,} \\
& \hspace{4em} \text{count,} \\
& \hspace{4em} \text{dtack,} \\
& \hspace{4em} \text{last-rw-,} \\
& \hspace{4em} \text{last-address,} \\
& \hspace{4em} \text{last-data))))) \\
= & \text{ fm9001-step* (list (list (regs-regs, flags), mem), pc-reg))}
\end{aligned}$$

THEOREM: macrocycle-invariant $\Rightarrow$ pc-reg
macrocycle-invariant (*state*, *pc-reg*) $\rightarrow$ (pc-reg (car (*state*)) = *pc-reg*)

THEOREM: middle=high
(fm9001-state-structure (*state*)
 $\wedge$ macrocycle-invariant (*state*, *pc*)
 $\wedge$ operating-inputs-p (*inputs*, microcycles (*state*)))
 $\rightarrow$ (map-up (run-fm9001 (*state*, *inputs*, microcycles (*state*)))
= fm9001-step (map-up (*state*), *pc*))

THEOREM: fm9001-interpreter-correct
(fm9001-state-structure (*state*)
 $\wedge$ macrocycle-invariant (*state*, *pc*)
 $\wedge$ operating-inputs-p (*inputs*,
total-microcycles (*state*, *inputs*, *instructions*)))
 $\rightarrow$ (map-up (run-fm9001 (*state*,
inputs,
total-microcycles (*state*, *inputs*, *instructions*)))
= fm9001-interpreter (map-up (*state*), *pc*, *instructions*))

THEOREM: chip-system=fm9001-interpreter
 (chip-system& (*netlist*)
 $\wedge$ fm9001-state-structure (*state*)
 $\wedge$ macrocycle-invariant (*state*, *pc*)
 $\wedge$ chip-system-operating-inputs-p (*inputs*,
 total-microcycles (*state*,
 map-up-inputs (*inputs*),
 instructions))
 $\wedge$ operating-inputs-p (map-up-inputs (*inputs*),
 total-microcycles (*state*,
 map-up-inputs (*inputs*),
 instructions)))
 $\rightarrow$ (map-up (simulate-dual-eval-2 ('**chip-system**,
 inputs,
 state,
 netlist,
 total-microcycles (*state*,
 map-up-inputs (*inputs*),
 instructions)))
 = fm9001-interpreter (map-up (*state*), *pc*, *instructions*))

THEOREM: fm9001=fm9001-interpreter
 fm9001 (*state*, *n*) = fm9001-interpreter (*state*, make-list (4, **t**), *n*)

DEFINITION:
 map-down-relation (*high-level-state*, *low-level-state*)
 = ((regs (car (*high-level-state*)) = car (regs (car (*low-level-state*))))
 $\wedge$ (flags (car (*high-level-state*)) = flags (car (*low-level-state*)))
 $\wedge$ (cadr (*high-level-state*) = caadr (*low-level-state*)))

EVENT: Enable map-down-relation; name this event 'map-down-relation-off'.

DEFINITION:
 high-level-state-structure (*state*)
 = (properp (*state*)
 $\wedge$ properp (car (*state*))
 $\wedge$ (length (car (*state*)) = 2)
 $\wedge$ (length (*state*) = 2))

EVENT: Enable high-level-state-structure; name this event 'high-level-state-structure-off'.

THEOREM: map-down-relation-lemma
 (high-level-state-structure (*high-level-state*)

$\wedge$ map-down-relation (*high-level-state*, *low-level-state*)
 $\rightarrow$ (map-up (*low-level-state*) = *high-level-state*)

THEOREM: fm9001-interpreter-correct\$map-down
 (high-level-state-structure (*high-level-state*)
 $\wedge$ map-down-relation (*high-level-state*, *low-level-state*)
 $\wedge$ fm9001-state-structure (*low-level-state*)
 $\wedge$ macrocycle-invariant (*low-level-state*, *pc*)
 $\wedge$ operating-inputs-p (*inputs*,
 total-microcycles (*low-level-state*, *inputs*, *n*)))
 $\rightarrow$ (map-up (run-fm9001 (*low-level-state*,
 inputs,
 total-microcycles (*low-level-state*, *inputs*, *n*)))
 = fm9001-interpreter (*high-level-state*, *pc*, *n*))

DEFINITION:
 fm9001-machine-statep (*p-state*)
 = (properp (*p-state*)
 $\wedge$ (length (*p-state*) = 2)
 $\wedge$ all-ramp-mem (4, car (*p-state*))
 $\wedge$ memory-okp (4, 32, car (*p-state*))
 $\wedge$ bvp (cadr (*p-state*))
 $\wedge$ (length (cadr (*p-state*)) = 4))

DEFINITION:
 fm9001-statep (*state*)
 = (properp (*state*)
 $\wedge$ (length (*state*) = 2)
 $\wedge$ fm9001-machine-statep (car (*state*))
 $\wedge$ memory-okp (32, 32, cadr (*state*)))

DEFINITION:
 p-map-down (*p-state*)
 = list (list (car (*p-state*), **f**, make-list (32, **f**), make-list (4, **t**)),
 cadr (*p-state*),
 read-mem (make-list (4, **t**), car (*p-state*)),
 make-list (32, **f**),
 make-list (32, **f**),
 make-list (32, **f**),
 read-mem (make-list (4, **t**), car (*p-state*)),
 t,
 t,
 t,
 make-list (4, **t**),
 cv_fetch1 (**t**,

list (**t**, **t**, **t**, **t**),
 make-list (32, **f**),
 list (**t**, **f**, **f**, **f**),
 make-list (4, **t**)))

DEFINITION:

mem-map-down (*memory*)
 = list (*memory*, 0, 0, 0, **t**, **t**, make-list (32, **f**), make-list (32, **x**))

DEFINITION:

map-down (*state*) = list (p-map-down (car (*state*)), mem-map-down (cadr (*state*)))

THEOREM: map-up-inverts-map-down

fm9001-statep (*state*) $\rightarrow$ (map-up (map-down (*state*)) = *state*)

THEOREM: fm9001-statep-implies-fm9001-state-structure

fm9001-statep (*state*) $\rightarrow$ fm9001-state-structure (map-down (*state*))

THEOREM: fm9001-statep-implies-macrocycle-invariant-lemma1

(cv_fetch1 (**t**, list (**t**, **t**, **t**, **t**), make-list (32, **f**), list (**t**, **f**, **f**, **f**), make-list (4, **t**))
 = cv_fetch1 (**t**, make-list (4, **t**), make-list (32, **f**), *z*, make-list (4, **t**))
 = **t**)

THEOREM: fm9001-statep-implies-macrocycle-invariant

fm9001-statep (*state*)
 $\rightarrow$ macrocycle-invariant (map-down (*state*), make-list (4, **t**))

THEOREM: fm9001=chip-system-lemma1

nth (10, car (map-down (*state*))) = list (**t**, **t**, **t**, **t**)

THEOREM: fm9001=chip-system

(chip-system& (*netlist*)
 $\wedge$ fm9001-statep (*state*)
 $\wedge$ chip-system-operating-inputs-p (*inputs*,
 total-microcycles (map-down (*state*),
 map-up-inputs (*inputs*),
 n))
 $\wedge$ operating-inputs-p (map-up-inputs (*inputs*),
 total-microcycles (map-down (*state*),
 map-up-inputs (*inputs*),
 n)))
 $\rightarrow$ (fm9001 (*state*, *n*)
 = map-up (simulate-dual-eval-2 ('**chip-system**,
 inputs,
 map-down (*state*),

$netlist,$
 $total\_microcycles (map\_down (state),$
 $map\_up\_inputs (inputs),$
 $n))))$

DEFINITION:

$no\_holds\_reset\_or\_test (i, c)$
 $= (chip\_system\_operating\_inputs\_p (i, c)$
 $\wedge operating\_inputs\_p (map\_up\_inputs (i), c))$

THEOREM: $fm9001 = chip\_system\_summary$

$(fm9001\_statep (s)$
 $\wedge no\_holds\_reset\_or\_test (i,$
 $total\_microcycles (map\_down (s),$
 $map\_up\_inputs (i),$
 $n)))$

$\rightarrow (fm9001 (s, n)$
 $= map\_up (simulate\_dual\_eval\_2 ('chip\_system,$
 $i,$
 $map\_down (s),$
 $CHIP\_SYSTEM\$NETLIST,$
 $total\_microcycles (map\_down (s),$
 $map\_up\_inputs (i),$
 $n))))$

DEFINITION:

$b\_knownp (x) = ((x = \mathbf{t}) \vee (x = \mathbf{f}) \vee (x = \mathbf{z}))$

THEOREM: $b\_knownp\_compound\_recognizer$

$b\_knownp (x) = (truep (x) \vee falsep (x) \vee zp (x))$

EVENT: Enable b_knownp ; name this event ' b_knownp_off '.

EVENT: Enable $*1*b\_knownp$; name this event ' $g*1*b\_knownp\_off$ '.

DEFINITION:

$b\_approx (a1, a2)$
 $= \mathbf{if} \ b\_knownp (a1) \ \mathbf{then} \ a1 = a2$
 $\quad \mathbf{else} \ \mathbf{t} \ \mathbf{endif}$

DEFINITION:

$v\_approx (v1, v2)$
 $= \mathbf{if} \ listp (v1)$
 $\quad \mathbf{then} \ listp (v2)$


```

       $\wedge$  b-approx(car(v1), car(v2))
       $\wedge$  v-approx(cdr(v1), cdr(v2))
    else v1 = v2 endif

```

DEFINITION:

```

v-knownp(x)
= if listp(x) then b-knownp(car(x))  $\wedge$  v-knownp(cdr(x))
  else t endif

```

THEOREM: v-knownp-implies-v-approx-is-equal
 $((x \neq y) \wedge \text{v-knownp}(x)) \rightarrow (\neg \text{v-approx}(x, y))$

EVENT: Enable plus-add1; name this event 'plus-add1-off'.

DEFINITION: MEM-WIDTH = 32

DEFINITION:

```

s-approx(s1, s2)
= if listp(s1)  $\vee$  listp(s2)
  then if listp(s1)
    then if listp(s2)
      then s-approx(car(s1), car(s2))
             $\wedge$  s-approx(cdr(s1), cdr(s2))
      else f endif
    else f endif
  elseif (s1 = nil)  $\vee$  (s2 = nil) then s1 = s2
  elseif ramp(s1)  $\vee$  ramp(s2)
  then if ramp(s1)
    then if ramp(s2) then v-approx(ram-guts(s1), ram-guts(s2))
      else f endif
    else f endif
  elseif romp(s1)  $\vee$  romp(s2)
  then if romp(s1)
    then if romp(s2) then v-approx(rom-guts(s1), rom-guts(s2))
      else f endif
    else f endif
  elseif stubp(s1)  $\vee$  stubp(s2)
  then if stubp(s1)
    then if stubp(s2) then v-approx(stub-guts(s1), stub-guts(s2))
      else f endif
    else f endif
  else b-approx(s1, s2) endif

```

THEOREM: s-approx-x-x

s-approx(x, x)

DEFINITION:

```
good-s (s)
=  if listp (s) then good-s (car (s)) ∧ good-s (cdr (s))
    elseif ramp (s)
        then (length (ram-guts (s)) = MEM-WIDTH) ∧ properp (ram-guts (s))
    elseif romp (s)
        then (length (rom-guts (s)) = MEM-WIDTH) ∧ properp (rom-guts (s))
    elseif stubp (s)
        then (length (stub-guts (s)) = MEM-WIDTH) ∧ properp (stub-guts (s))
    else t endif
```

DEFINITION:

```
s-knownp (s)
=  if listp (s) then s-knownp (car (s)) ∧ s-knownp (cdr (s))
    elseif s = nil then t
    elseif ramp (s) then v-knownp (ram-guts (s))
    elseif romp (s) then v-knownp (rom-guts (s))
    elseif stubp (s) then v-knownp (stub-guts (s))
    else b-knownp (s) endif
```

THEOREM: s-knownp-implies-s-approx-is-equal
(s-knownp (x) ∧ (x ≠ y)) → (¬ s-approx (x, y))

EVENT: Disable plus-add1; name this event 'plus-add1-on'.

DEFINITION:

```
v-approx-alist (alist1, alist2)
=  if listp (alist1)
    then listp (alist2)
        ∧ (caar (alist1) = caar (alist2))
        ∧ b-approx (cdar (alist1), cdar (alist2))
        ∧ v-approx-alist (cdr (alist1), cdr (alist2))
    else alist2 ≈ nil endif
```

DEFINITION:

```
s-approx-alist (alist1, alist2)
=  if listp (alist1)
    then listp (alist2)
        ∧ (caar (alist1) = caar (alist2))
        ∧ s-approx (cdar (alist1), cdar (alist2))
        ∧ s-approx-alist (cdr (alist1), cdr (alist2))
    else alist2 ≈ nil endif
```

DEFINITION:

```

monotonicity-property (flag, fn, netlist, a1, a2, s1, s2)
=  case on flag:
    case = 0
    then (v-approx (a1, a2) ∧ s-approx (s1, s2))
        → v-approx (dual-eval (0, fn, a1, s1, netlist),
                      dual-eval (0, fn, a2, s2, netlist))
    case = 1
    then (v-approx-alist (a1, a2)
          ∧ alistp (a1)
          ∧ alistp (a2)
          ∧ s-approx-alist (s1, s2)
          ∧ alistp (s1)
          ∧ alistp (s2))
        → v-approx-alist (dual-eval (1, fn, a1, s1, netlist),
                          dual-eval (1, fn, a2, s2, netlist))
    case = 2
    then (v-approx (a1, a2) ∧ s-approx (s1, s2))
        → s-approx (dual-eval (2, fn, a1, s1, netlist),
                    dual-eval (2, fn, a2, s2, netlist))
    case = 3
    then (v-approx-alist (a1, a2)
          ∧ alistp (a1)
          ∧ alistp (a2)
          ∧ s-approx-alist (s1, s2)
          ∧ alistp (s1)
          ∧ alistp (s2))
        → s-approx-alist (dual-eval (3, fn, a1, s1, netlist),
                          dual-eval (3, fn, a2, s2, netlist))
    otherwise t endcase

```

EVENT: Enable monotonicity-property; name this event 'monotonicity-property-off'.

THEOREM: monotonicity-property-consequence-0
(monotonicity-property (0, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
 ∧ v-approx (*a1*, *a2*)
 ∧ s-approx (*s1*, *s2*))
 → v-approx (dual-eval (0, *fn*, *a1*, *s1*, *netlist*), dual-eval (0, *fn*, *a2*, *s2*, *netlist*))

THEOREM: monotonicity-property-consequence-1
(monotonicity-property (1, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
 ∧ v-approx-alist (*a1*, *a2*)
 ∧ alistp (*a1*)
 ∧ alistp (*a2*)

$$\begin{aligned}
& \wedge \text{ s-approx-alist } (s1, s2) \\
& \wedge \text{ alistp } (s1) \\
& \wedge \text{ alistp } (s2)) \\
\rightarrow & \text{ v-approx-alist } (\text{dual-eval } (1, fn, a1, s1, netlist), \\
& \text{dual-eval } (1, fn, a2, s2, netlist))
\end{aligned}$$

THEOREM: monotonicity-property-consequence-2
 $(\text{monotonicity-property } (2, fn, netlist, a1, a2, s1, s2)$
 $\wedge \text{ v-approx } (a1, a2)$
 $\wedge \text{ s-approx } (s1, s2))$
 $\rightarrow \text{ s-approx } (\text{dual-eval } (2, fn, a1, s1, netlist), \text{dual-eval } (2, fn, a2, s2, netlist))$

THEOREM: monotonicity-property-consequence-3
 $(\text{monotonicity-property } (3, fn, netlist, a1, a2, s1, s2)$
 $\wedge \text{ v-approx-alist } (a1, a2)$
 $\wedge \text{ alistp } (a1)$
 $\wedge \text{ alistp } (a2)$
 $\wedge \text{ s-approx-alist } (s1, s2)$
 $\wedge \text{ alistp } (s1)$
 $\wedge \text{ alistp } (s2))$
 $\rightarrow \text{ s-approx-alist } (\text{dual-eval } (3, fn, a1, s1, netlist),$
 $\text{dual-eval } (3, fn, a2, s2, netlist))$

THEOREM: monotonicity-property-opener-0
 $\text{monotonicity-property } (0, fn, netlist, a1, a2, s1, s2)$
 $\leftrightarrow ((\text{v-approx } (a1, a2) \wedge \text{s-approx } (s1, s2))$
 $\rightarrow \text{ v-approx } (\text{dual-eval } (0, fn, a1, s1, netlist),$
 $\text{dual-eval } (0, fn, a2, s2, netlist)))$

THEOREM: monotonicity-property-opener-1
 $\text{monotonicity-property } (1, fn, netlist, a1, a2, s1, s2)$
 $\leftrightarrow ((\text{v-approx-alist } (a1, a2)$
 $\wedge \text{ alistp } (a1)$
 $\wedge \text{ alistp } (a2)$
 $\wedge \text{ s-approx-alist } (s1, s2)$
 $\wedge \text{ alistp } (s1)$
 $\wedge \text{ alistp } (s2))$
 $\rightarrow \text{ v-approx-alist } (\text{dual-eval } (1, fn, a1, s1, netlist),$
 $\text{dual-eval } (1, fn, a2, s2, netlist)))$

THEOREM: monotonicity-property-opener-2
 $\text{monotonicity-property } (2, fn, netlist, a1, a2, s1, s2)$
 $\leftrightarrow ((\text{v-approx } (a1, a2) \wedge \text{s-approx } (s1, s2))$
 $\rightarrow \text{ s-approx } (\text{dual-eval } (2, fn, a1, s1, netlist),$
 $\text{dual-eval } (2, fn, a2, s2, netlist)))$

THEOREM: monotonicity-property-opener-3
monotonicity-property ($3, fn, netlist, a1, a2, s1, s2$)
 $\leftrightarrow ((v\text{-approx-alist } (a1, a2)$
 $\wedge \text{ alistp } (a1)$
 $\wedge \text{ alistp } (a2)$
 $\wedge \text{ s-approx-alist } (s1, s2)$
 $\wedge \text{ alistp } (s1)$
 $\wedge \text{ alistp } (s2))$
 $\rightarrow \text{ s-approx-alist } (\text{dual-eval } (3, fn, a1, s1, netlist),$
 $\text{dual-eval } (3, fn, a2, s2, netlist)))$

EVENT: Enable monotonicity-property-opener-0; name this event ‘monotonicity-property-opener-0-off’.

EVENT: Enable monotonicity-property-opener-1; name this event ‘monotonicity-property-opener-1-off’.

EVENT: Enable monotonicity-property-opener-2; name this event ‘monotonicity-property-opener-2-off’.

EVENT: Enable monotonicity-property-opener-3; name this event ‘monotonicity-property-opener-3-off’.

THEOREM: f-buf-monotone
 $\text{b-approx } (a1, a2) \rightarrow \text{b-approx } (\text{f-buf } (a1), \text{f-buf } (a2))$

THEOREM: f-and-monotone
 $(\text{b-approx } (a1, a2) \wedge \text{b-approx } (b1, b2))$
 $\rightarrow \text{b-approx } (\text{f-and } (a1, b1), \text{f-and } (a2, b2))$

THEOREM: f-and3-monotone
 $(\text{b-approx } (a1, a2) \wedge \text{b-approx } (b1, b2) \wedge \text{b-approx } (c1, c2))$
 $\rightarrow \text{b-approx } (\text{f-and3 } (a1, b1, c1), \text{f-and3 } (a2, b2, c2))$

THEOREM: f-and4-monotone
 $(\text{b-approx } (a1, a2) \wedge \text{b-approx } (b1, b2) \wedge \text{b-approx } (c1, c2) \wedge \text{b-approx } (d1, d2))$
 $\rightarrow \text{b-approx } (\text{f-and4 } (a1, b1, c1, d1), \text{f-and4 } (a2, b2, c2, d2))$

THEOREM: f-not-monotone
 $\text{b-approx } (a1, a2) \rightarrow \text{b-approx } (\text{f-not } (a1), \text{f-not } (a2))$

THEOREM: f-nand-monotone
 $(\text{b-approx } (a1, a2) \wedge \text{b-approx } (b1, b2))$
 $\rightarrow \text{b-approx } (\text{f-nand } (a1, b1), \text{f-nand } (a2, b2))$

THEOREM: f-nand3-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2)) \\ \rightarrow & \text{b-approx}(\text{f-nand3}(a1, b1, c1), \text{f-nand3}(a2, b2, c2)) \end{aligned}$$

THEOREM: f-nand4-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2) \wedge \text{b-approx}(d1, d2)) \\ \rightarrow & \text{b-approx}(\text{f-nand4}(a1, b1, c1, d1), \text{f-nand4}(a2, b2, c2, d2)) \end{aligned}$$

THEOREM: f-nand5-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2)) \\ \rightarrow & \text{b-approx}(\text{f-nand5}(a1, b1, c1, d1, e1), \text{f-nand5}(a2, b2, c2, d2, e2)) \end{aligned}$$

THEOREM: f-nand6-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2) \\ & \wedge \text{b-approx}(g1, g2)) \\ \rightarrow & \text{b-approx}(\text{f-nand6}(a1, b1, c1, d1, e1, g1), \text{f-nand6}(a2, b2, c2, d2, e2, g2)) \end{aligned}$$

THEOREM: f-nand8-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2) \\ & \wedge \text{b-approx}(g1, g2) \\ & \wedge \text{b-approx}(h1, h2) \\ & \wedge \text{b-approx}(i1, i2)) \\ \rightarrow & \text{b-approx}(\text{f-nand8}(a1, b1, c1, d1, e1, g1, h1, i1), \\ & \text{f-nand8}(a2, b2, c2, d2, e2, g2, h2, i2)) \end{aligned}$$

THEOREM: f-or-monotone

$$(\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2)) \rightarrow \text{b-approx}(\text{f-or}(a1, b1), \text{f-or}(a2, b2))$$

THEOREM: f-or3-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2)) \\ \rightarrow & \text{b-approx}(\text{f-or3}(a1, b1, c1), \text{f-or3}(a2, b2, c2)) \end{aligned}$$

THEOREM: f-or4-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2) \wedge \text{b-approx}(d1, d2)) \\ \rightarrow & \text{b-approx}(\text{f-or4}(a1, b1, c1, d1), \text{f-or4}(a2, b2, c2, d2)) \end{aligned}$$

THEOREM: f-nor-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2)) \\ \rightarrow & \text{b-approx}(\text{f-nor}(a1, b1), \text{f-nor}(a2, b2)) \end{aligned}$$

THEOREM: f-nor3-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2)) \\ \rightarrow & \text{b-approx}(\text{f-nor3}(a1, b1, c1), \text{f-nor3}(a2, b2, c2)) \end{aligned}$$

THEOREM: f-nor4-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2) \wedge \text{b-approx}(c1, c2) \wedge \text{b-approx}(d1, d2)) \\ \rightarrow & \text{b-approx}(\text{f-nor4}(a1, b1, c1, d1), \text{f-nor4}(a2, b2, c2, d2)) \end{aligned}$$

THEOREM: f-nor5-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2)) \\ \rightarrow & \text{b-approx}(\text{f-nor5}(a1, b1, c1, d1, e1), \text{f-nor5}(a2, b2, c2, d2, e2)) \end{aligned}$$

THEOREM: f-nor6-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2) \\ & \wedge \text{b-approx}(g1, g2)) \\ \rightarrow & \text{b-approx}(\text{f-nor6}(a1, b1, c1, d1, e1, g1), \text{f-nor6}(a2, b2, c2, d2, e2, g2)) \end{aligned}$$

THEOREM: f-nor8-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \\ & \wedge \text{b-approx}(b1, b2) \\ & \wedge \text{b-approx}(c1, c2) \\ & \wedge \text{b-approx}(d1, d2) \\ & \wedge \text{b-approx}(e1, e2) \\ & \wedge \text{b-approx}(g1, g2) \\ & \wedge \text{b-approx}(h1, h2) \\ & \wedge \text{b-approx}(i1, i2)) \\ \rightarrow & \text{b-approx}(\text{f-nor8}(a1, b1, c1, d1, e1, g1, h1, i1), \\ & \text{f-nor8}(a2, b2, c2, d2, e2, g2, h2, i2)) \end{aligned}$$

THEOREM: f-xor-monotone

$$\begin{aligned} & (\text{b-approx}(a1, a2) \wedge \text{b-approx}(b1, b2)) \\ \rightarrow & \text{b-approx}(\text{f-xor}(a1, b1), \text{f-xor}(a2, b2)) \end{aligned}$$

THEOREM: f-equiv-monotone
 $(b\text{-approx}(a1, a2) \wedge b\text{-approx}(b1, b2))$
 $\rightarrow b\text{-approx}(f\text{-equiv}(a1, b1), f\text{-equiv}(a2, b2))$

THEOREM: f-xor3-monotone
 $(b\text{-approx}(a1, a2) \wedge b\text{-approx}(b1, b2) \wedge b\text{-approx}(c1, c2))$
 $\rightarrow b\text{-approx}(f\text{-xor3}(a1, b1, c1), f\text{-xor3}(a2, b2, c2))$

THEOREM: f-equiv3-monotone
 $(b\text{-approx}(a1, a2) \wedge b\text{-approx}(b1, b2) \wedge b\text{-approx}(c1, c2))$
 $\rightarrow b\text{-approx}(f\text{-equiv3}(a1, b1, c1), f\text{-equiv3}(a2, b2, c2))$

THEOREM: f-if-monotone
 $(b\text{-approx}(c1, c2) \wedge b\text{-approx}(a1, a2) \wedge b\text{-approx}(b1, b2))$
 $\rightarrow b\text{-approx}(f\text{-if}(c1, a1, b1), f\text{-if}(c2, a2, b2))$

THEOREM: ft-buf-monotone
 $(b\text{-approx}(c1, c2) \wedge b\text{-approx}(a1, a2))$
 $\rightarrow b\text{-approx}(ft\text{-buf}(c1, a1), ft\text{-buf}(c2, a2))$

THEOREM: ft-wire-monotone
 $(b\text{-approx}(a1, a2) \wedge b\text{-approx}(b1, b2))$
 $\rightarrow b\text{-approx}(ft\text{-wire}(a1, b1), ft\text{-wire}(a2, b2))$

THEOREM: f-pullup-monotone
 $b\text{-approx}(a1, a2) \rightarrow b\text{-approx}(f\text{-pullup}(a1), f\text{-pullup}(a2))$

EVENT: Enable f-buf; name this event 'f-buf-off'.

EVENT: Enable f-and; name this event 'f-and-off'.

EVENT: Enable f-and3; name this event 'f-and3-off'.

EVENT: Enable f-and4; name this event 'f-and4-off'.

EVENT: Enable f-not; name this event 'f-not-off'.

EVENT: Enable f-nand; name this event 'f-nand-off'.

EVENT: Enable f-nand3; name this event 'f-nand3-off'.

EVENT: Enable f-nand4; name this event 'f-nand4-off'.

EVENT: Enable f-nand5; name this event 'f-nand5-off'.

EVENT: Enable f-nand6; name this event 'f-nand6-off'.

EVENT: Enable f-nand8; name this event 'f-nand8-off'.

EVENT: Enable f-or; name this event 'f-or-off'.

EVENT: Enable f-or3; name this event 'f-or3-off'.

EVENT: Enable f-or4; name this event 'f-or4-off'.

EVENT: Enable f-nor; name this event 'f-nor-off'.

EVENT: Enable f-nor3; name this event 'f-nor3-off'.

EVENT: Enable f-nor4; name this event 'f-nor4-off'.

EVENT: Enable f-nor5; name this event 'f-nor5-off'.

EVENT: Enable f-nor6; name this event 'f-nor6-off'.

EVENT: Enable f-nor8; name this event 'f-nor8-off'.

EVENT: Enable f-xor; name this event 'f-xor-off'.

EVENT: Enable f-xor3; name this event 'f-xor3-off'.

EVENT: Enable f-equiv; name this event 'f-equiv-off'.

EVENT: Enable f-equiv3; name this event 'f-equiv3-off'.

EVENT: Enable f-if; name this event 'f-if-off'.

EVENT: Enable ft-buf; name this event 'ft-buf-off'.

EVENT: Enable ft-wire; name this event ‘ft-wire-off’.

EVENT: Enable f-pullup; name this event ‘f-pullup-off’.

EVENT: Let us define the theory *monotonicity-lemmas* to consist of the following events: f-buf-monotone, f-and-monotone, f-and3-monotone, f-and4-monotone, f-not-monotone, f-nand-monotone, f-nand3-monotone, f-nand4-monotone, f-nand5-monotone, f-nand6-monotone, f-nand8-monotone, f-or-monotone, f-or3-monotone, f-or4-monotone, f-nor-monotone, f-nor3-monotone, f-nor4-monotone, f-nor5-monotone, f-nor6-monotone, f-nor8-monotone, f-xor-monotone, f-xor3-monotone, f-equiv-monotone, f-equiv3-monotone, f-if-monotone, ft-buf-monotone, ft-wire-monotone, f-pullup-monotone.

EVENT: Enable b-approx; name this event ‘b-approx-off’.

THEOREM: dual-eval-0-primp
primp(*name*)
→ (dual-eval(0, *name*, *args*, *state*, *netlist*)
= dual-apply-value(*name*, *args*, *state*))

THEOREM: dual-eval-2-primp
primp(*name*)
→ (dual-eval(2, *name*, *args*, *state*, *netlist*)
= dual-apply-state(*name*, *args*, *state*))

THEOREM: s-approx-implies-b-approx
s-approx(*x*, *y*) → b-approx(*x*, *y*)

THEOREM: f-buf-type-set
truep(f-buf(*x*)) ∨ falsep(f-buf(*x*)) ∨ xp(f-buf(*x*))

THEOREM: fourp-f-buf
fourp(f-buf(*x*))

THEOREM: fourp-f-if
fourp(f-if(*x*, *y*, *z*))

THEOREM: fourp-implies-s-approx-is-b-approx
(fourp(*x*) ∧ fourp(*y*) ∧ b-approx(*x*, *y*)) → (s-approx(*x*, *y*) = t)

THEOREM: dual-eval-ao2-value
dual-eval(0, 'ao2, *args*, *state*, *netlist*)
= list(f-nor(f-and(car(*args*), cadr(*args*)),
f-and(caddr(*args*), caddr(*args*))))

THEOREM: dual-eval-ao2-state

$\text{dual-eval}(2, \text{'ao2}, args, state, netlist) = 0$

THEOREM: ao2-monotone

$\text{monotonicity-property}(0, \text{'ao2}, netlist, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'ao2}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ao4-value

$\text{dual-eval}(0, \text{'ao4}, args, state, netlist)$

$= \text{list}(\text{f-nand}(\text{f-or}(\text{car}(args), \text{cadr}(args)), \text{f-or}(\text{caddr}(args), \text{caddr}(args))))$

THEOREM: dual-eval-ao4-state

$\text{dual-eval}(2, \text{'ao4}, args, state, netlist) = 0$

THEOREM: ao4-monotone

$\text{monotonicity-property}(0, \text{'ao4}, netlist, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'ao4}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ao6-value

$\text{dual-eval}(0, \text{'ao6}, args, state, netlist)$

$= \text{list}(\text{f-nor}(\text{f-and}(\text{car}(args), \text{cadr}(args)), \text{caddr}(args)))$

THEOREM: dual-eval-ao6-state

$\text{dual-eval}(2, \text{'ao6}, args, state, netlist) = 0$

THEOREM: ao6-monotone

$\text{monotonicity-property}(0, \text{'ao6}, netlist, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'ao6}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ao7-value

$\text{dual-eval}(0, \text{'ao7}, args, state, netlist)$

$= \text{list}(\text{f-nand}(\text{f-or}(\text{car}(args), \text{cadr}(args)), \text{caddr}(args)))$

THEOREM: dual-eval-ao7-state

$\text{dual-eval}(2, \text{'ao7}, args, state, netlist) = 0$

THEOREM: ao7-monotone

$\text{monotonicity-property}(0, \text{'ao7}, netlist, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'ao7}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-b-and-value

$\text{dual-eval}(0, \text{'b-and}, args, state, netlist)$

$= \text{list}(\text{f-and}(\text{car}(args), \text{cadr}(args)))$

THEOREM: dual-eval-b-and-state

$\text{dual-eval}(2, \text{'b-and}, args, state, netlist) = 0$

THEOREM: b-and-monotone
monotonicity-property (0, 'b-and, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-and, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-and3-value
dual-eval (0, 'b-and3, args, state, netlist)
= list (f-and3 (car (args), cadr (args), caddr (args)))

THEOREM: dual-eval-b-and3-state
dual-eval (2, 'b-and3, args, state, netlist) = 0

THEOREM: b-and3-monotone
monotonicity-property (0, 'b-and3, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-and3, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-and4-value
dual-eval (0, 'b-and4, args, state, netlist)
= list (f-and4 (car (args), cadr (args), caddr (args), caddr (args)))

THEOREM: dual-eval-b-and4-state
dual-eval (2, 'b-and4, args, state, netlist) = 0

THEOREM: b-and4-monotone
monotonicity-property (0, 'b-and4, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-and4, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-equiv-value
dual-eval (0, 'b-equiv, args, state, netlist)
= list (f-equiv (car (args), cadr (args)))

THEOREM: dual-eval-b-equiv-state
dual-eval (2, 'b-equiv, args, state, netlist) = 0

THEOREM: b-equiv-monotone
monotonicity-property (0, 'b-equiv, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-equiv, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-equiv3-value
dual-eval (0, 'b-equiv3, args, state, netlist)
= list (f-equiv3 (car (args), cadr (args), caddr (args)))

THEOREM: dual-eval-b-equiv3-state
dual-eval (2, 'b-equiv3, args, state, netlist) = 0

THEOREM: b-equiv3-monotone
monotonicity-property (0, 'b-equiv3, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-equiv3, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-if-value

dual-eval(0, 'b-if, args, state, netlist)
= list(f-if(car(args), cadr(args), caddr(args)))

THEOREM: dual-eval-b-if-state

dual-eval(2, 'b-if, args, state, netlist) = 0

THEOREM: b-if-monotone

monotonicity-property(0, 'b-if, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'b-if, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand-value

dual-eval(0, 'b-nand, args, state, netlist)
= list(f-nand(car(args), cadr(args)))

THEOREM: dual-eval-b-nand-state

dual-eval(2, 'b-nand, args, state, netlist) = 0

THEOREM: b-nand-monotone

monotonicity-property(0, 'b-nand, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'b-nand, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand3-value

dual-eval(0, 'b-nand3, args, state, netlist)
= list(f-nand3(car(args), cadr(args), caddr(args)))

THEOREM: dual-eval-b-nand3-state

dual-eval(2, 'b-nand3, args, state, netlist) = 0

THEOREM: b-nand3-monotone

monotonicity-property(0, 'b-nand3, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'b-nand3, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand4-value

dual-eval(0, 'b-nand4, args, state, netlist)
= list(f-nand4(car(args), cadr(args), caddr(args), caddr(args)))

THEOREM: dual-eval-b-nand4-state

dual-eval(2, 'b-nand4, args, state, netlist) = 0

THEOREM: b-nand4-monotone

monotonicity-property(0, 'b-nand4, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'b-nand4, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand5-value
dual-eval (0, 'b-nand5, args, state, netlist)
= list (f-nand5 (car (args),
 cadr (args),
 caddr (args),
 caddr (args),
 caddr (args)))

THEOREM: dual-eval-b-nand5-state
dual-eval (2, 'b-nand5, args, state, netlist) = 0

THEOREM: b-nand5-monotone
monotonicity-property (0, 'b-nand5, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-nand5, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand6-value
dual-eval (0, 'b-nand6, args, state, netlist)
= list (f-nand6 (car (args),
 cadr (args),
 caddr (args),
 caddr (args),
 caddr (args),
 caddr (args)))

THEOREM: dual-eval-b-nand6-state
dual-eval (2, 'b-nand6, args, state, netlist) = 0

THEOREM: b-nand6-monotone
monotonicity-property (0, 'b-nand6, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'b-nand6, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nand8-value
dual-eval (0, 'b-nand8, args, state, netlist)
= list (f-nand8 (car (args),
 cadr (args),
 caddr (args),
 caddr (args),
 caddr (args),
 caddr (args),
 caddr (args),
 caddr (args)))

THEOREM: dual-eval-b-nand8-state
dual-eval (2, 'b-nand8, args, state, netlist) = 0

THEOREM: b-nand8-monotone
monotonicity-property (0, 'b-nand8, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nand8, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-nbuf-value
dual-eval (0, 'b-nbuf, *args*, *state*, *netlist*)
= list (f-not (car (*args*)), f-buf (car (*args*)))

THEOREM: dual-eval-b-nbuf-state
dual-eval (2, 'b-nbuf, *args*, *state*, *netlist*) = 0

THEOREM: b-nbuf-monotone
monotonicity-property (0, 'b-nbuf, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nbuf, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-nor-value
dual-eval (0, 'b-nor, *args*, *state*, *netlist*)
= list (f-nor (car (*args*), cadr (*args*)))

THEOREM: dual-eval-b-nor-state
dual-eval (2, 'b-nor, *args*, *state*, *netlist*) = 0

THEOREM: b-nor-monotone
monotonicity-property (0, 'b-nor, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nor, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-nor3-value
dual-eval (0, 'b-nor3, *args*, *state*, *netlist*)
= list (f-nor3 (car (*args*), cadr (*args*), caddr (*args*)))

THEOREM: dual-eval-b-nor3-state
dual-eval (2, 'b-nor3, *args*, *state*, *netlist*) = 0

THEOREM: b-nor3-monotone
monotonicity-property (0, 'b-nor3, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nor3, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-nor4-value
dual-eval (0, 'b-nor4, *args*, *state*, *netlist*)
= list (f-nor4 (car (*args*), cadr (*args*), caddr (*args*), caddr (*args*)))

THEOREM: dual-eval-b-nor4-state
dual-eval (2, 'b-nor4, *args*, *state*, *netlist*) = 0

THEOREM: b-nor4-monotone
monotonicity-property (0, 'b-nor4, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nor4, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-nor5-value
dual-eval(0, 'b-nor5, args, state, netlist)
= list(f-nor5(car(args),
 cadr(args),
 caddr(args),
 cadddr(args),
 caddddr(args)))

THEOREM: dual-eval-b-nor5-state
dual-eval(2, 'b-nor5, args, state, netlist) = 0

THEOREM: b-nor5-monotone
monotonicity-property(0, 'b-nor5, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property(2, 'b-nor5, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nor6-value
dual-eval(0, 'b-nor6, args, state, netlist)
= list(f-nor6(car(args),
 cadr(args),
 caddr(args),
 cadddr(args),
 caddddr(args),
 cadddddr(args)))

THEOREM: dual-eval-b-nor6-state
dual-eval(2, 'b-nor6, args, state, netlist) = 0

THEOREM: b-nor6-monotone
monotonicity-property(0, 'b-nor6, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property(2, 'b-nor6, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-b-nor8-value
dual-eval(0, 'b-nor8, args, state, netlist)
= list(f-nor8(car(args),
 cadr(args),
 caddr(args),
 cadddr(args),
 caddddr(args),
 cadddddr(args),
 caddddddr(args),
 cadddddddrr(args)))

THEOREM: dual-eval-b-nor8-state
dual-eval(2, 'b-nor8, args, state, netlist) = 0

THEOREM: b-nor8-monotone
monotonicity-property (0, 'b-nor8, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-nor8, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-not-value
dual-eval (0, 'b-not, *args*, *state*, *netlist*) = list (f-not (car (*args*)))

THEOREM: dual-eval-b-not-state
dual-eval (2, 'b-not, *args*, *state*, *netlist*) = 0

THEOREM: b-not-monotone
monotonicity-property (0, 'b-not, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-not, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-not-b4ip-value
dual-eval (0, 'b-not-b4ip, *args*, *state*, *netlist*) = list (f-not (car (*args*)))

THEOREM: dual-eval-b-not-b4ip-state
dual-eval (2, 'b-not-b4ip, *args*, *state*, *netlist*) = 0

THEOREM: b-not-b4ip-monotone
monotonicity-property (0, 'b-not-b4ip, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-not-b4ip, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-not-ivap-value
dual-eval (0, 'b-not-ivap, *args*, *state*, *netlist*) = list (f-not (car (*args*)))

THEOREM: dual-eval-b-not-ivap-state
dual-eval (2, 'b-not-ivap, *args*, *state*, *netlist*) = 0

THEOREM: b-not-ivap-monotone
monotonicity-property (0, 'b-not-ivap, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-not-ivap, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-or-value
dual-eval (0, 'b-or, *args*, *state*, *netlist*)
= list (f-or (car (*args*), cadr (*args*)))

THEOREM: dual-eval-b-or-state
dual-eval (2, 'b-or, *args*, *state*, *netlist*) = 0

THEOREM: b-or-monotone
monotonicity-property (0, 'b-or, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'b-or, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-or3-value
dual-eval (0, 'b-or3, *args*, *state*, *netlist*)
= list (f-or3 (car (*args*), cadr (*args*), caddr (*args*)))

THEOREM: dual-eval-b-or3-state

$\text{dual-eval}(2, \text{'b-or3}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: b-or3-monotone

$\text{monotonicity-property}(0, \text{'b-or3}, \text{netlist}, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'b-or3}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-b-or4-value

$\text{dual-eval}(0, \text{'b-or4}, \text{args}, \text{state}, \text{netlist})$

$= \text{list}(\text{f-or4}(\text{car}(\text{args}), \text{cadr}(\text{args}), \text{caddr}(\text{args}), \text{cadddr}(\text{args})))$

THEOREM: dual-eval-b-or4-state

$\text{dual-eval}(2, \text{'b-or4}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: b-or4-monotone

$\text{monotonicity-property}(0, \text{'b-or4}, \text{netlist}, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'b-or4}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-b-xor-value

$\text{dual-eval}(0, \text{'b-xor}, \text{args}, \text{state}, \text{netlist})$

$= \text{list}(\text{f-xor}(\text{car}(\text{args}), \text{cadr}(\text{args})))$

THEOREM: dual-eval-b-xor-state

$\text{dual-eval}(2, \text{'b-xor}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: b-xor-monotone

$\text{monotonicity-property}(0, \text{'b-xor}, \text{netlist}, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'b-xor}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-b-xor3-value

$\text{dual-eval}(0, \text{'b-xor3}, \text{args}, \text{state}, \text{netlist})$

$= \text{list}(\text{f-xor3}(\text{car}(\text{args}), \text{cadr}(\text{args}), \text{caddr}(\text{args})))$

THEOREM: dual-eval-b-xor3-state

$\text{dual-eval}(2, \text{'b-xor3}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: b-xor3-monotone

$\text{monotonicity-property}(0, \text{'b-xor3}, \text{netlist}, a1, a2, s1, s2)$

$\wedge \text{monotonicity-property}(2, \text{'b-xor3}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-del2-value

$\text{dual-eval}(0, \text{'del2}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\text{f-buf}(\text{car}(\text{args})))$

THEOREM: dual-eval-del2-state

$\text{dual-eval}(2, \text{'del2}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: del2-monotone
monotonicity-property (0, 'del2, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'del2, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-del4-value
dual-eval (0, 'del4, args, state, netlist) = list (f-buf (car (args)))

THEOREM: dual-eval-del4-state
dual-eval (2, 'del4, args, state, netlist) = 0

THEOREM: del4-monotone
monotonicity-property (0, 'del4, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'del4, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-del10-value
dual-eval (0, 'del10, args, state, netlist) = list (f-buf (car (args)))

THEOREM: dual-eval-del10-state
dual-eval (2, 'del10, args, state, netlist) = 0

THEOREM: del10-monotone
monotonicity-property (0, 'del10, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'del10, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-procmon-value
dual-eval (0, 'procmon, args, state, netlist)
= list (f-if (caddr (args),
f-if (cadr (args), f-if (car (args), f, f), car (args)),
caddr (args)))

THEOREM: dual-eval-procmon-state
dual-eval (2, 'procmon, args, state, netlist) = 0

THEOREM: procmon-monotone
monotonicity-property (0, 'procmon, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'procmon, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-fd1-value
dual-eval (0, 'fd1, args, state, netlist) = list (f-buf (state), f-not (state))

THEOREM: dual-eval-fd1-state
dual-eval (2, 'fd1, args, state, netlist) = f-buf (car (args))

THEOREM: fd1-monotone
monotonicity-property (0, 'fd1, netlist, a1, a2, s1, s2)
 $\wedge$ monotonicity-property (2, 'fd1, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-fdls-value
 $\text{dual-eval}(0, \text{'fd1s}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\text{f-buf}(\text{state}), \text{f-not}(\text{state}))$

THEOREM: dual-eval-fdls-state
 $\text{dual-eval}(2, \text{'fd1s}, \text{args}, \text{state}, \text{netlist})$
 $= \text{f-if}(\text{caddr}(\text{args}), \text{caddr}(\text{args}), \text{car}(\text{args}))$

THEOREM: fdls-monotone
 $\text{monotonicity-property}(0, \text{'fd1s}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'fd1s}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-fd1sp-value
 $\text{dual-eval}(0, \text{'fd1sp}, \text{args}, \text{state}, \text{netlist})$
 $= \text{list}(\text{f-buf}(\text{state}), \text{f-not}(\text{state}))$

THEOREM: dual-eval-fd1sp-state
 $\text{dual-eval}(2, \text{'fd1sp}, \text{args}, \text{state}, \text{netlist})$
 $= \text{f-if}(\text{caddr}(\text{args}), \text{caddr}(\text{args}), \text{car}(\text{args}))$

THEOREM: fd1sp-monotone
 $\text{monotonicity-property}(0, \text{'fd1sp}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'fd1sp}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-fd1slp-value
 $\text{dual-eval}(0, \text{'fd1slp}, \text{args}, \text{state}, \text{netlist})$
 $= \text{list}(\text{f-buf}(\text{state}), \text{f-not}(\text{state}))$

THEOREM: dual-eval-fd1slp-state
 $\text{dual-eval}(2, \text{'fd1slp}, \text{args}, \text{state}, \text{netlist})$
 $= \text{f-if}(\text{caddr}(\text{args}), \text{caddr}(\text{args}), \text{f-if}(\text{caddr}(\text{args}), \text{car}(\text{args}), \text{state}))$

THEOREM: fd1slp-monotone
 $\text{monotonicity-property}(0, \text{'fd1slp}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'fd1slp}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-id-value
 $\text{dual-eval}(0, \text{'id}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\text{car}(\text{args}))$

THEOREM: dual-eval-id-state
 $\text{dual-eval}(2, \text{'id}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: id-monotone
 $\text{monotonicity-property}(0, \text{'id}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'id}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-ram-enable-circuit-value
 $\text{dual-eval}(0, \text{'ram-enable-circuit}, \text{args}, \text{state}, \text{netlist})$
 $= \text{list}(\text{f-nand}(\text{caddr}(\text{args}), \text{caddr}(\text{args})))$

THEOREM: dual-eval-ram-enable-circuit-state

dual-eval(2, 'ram-enable-circuit, args, state, netlist) = 0

THEOREM: ram-enable-circuit-monotone

monotonicity-property(0, 'ram-enable-circuit, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'ram-enable-circuit, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-t-buf-value

dual-eval(0, 't-buf, args, state, netlist)
= list(ft-buf(car(args), cadr(args)))

THEOREM: dual-eval-t-buf-state

dual-eval(2, 't-buf, args, state, netlist) = 0

THEOREM: t-buf-monotone

monotonicity-property(0, 't-buf, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 't-buf, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-t-wire-value

dual-eval(0, 't-wire, args, state, netlist)
= list(ft-wire(car(args), cadr(args)))

THEOREM: dual-eval-t-wire-state

dual-eval(2, 't-wire, args, state, netlist) = 0

THEOREM: t-wire-monotone

monotonicity-property(0, 't-wire, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 't-wire, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-pullup-value

dual-eval(0, 'pullup, args, state, netlist) = list(f-pullup(car(args)))

THEOREM: dual-eval-pullup-state

dual-eval(2, 'pullup, args, state, netlist) = 0

THEOREM: pullup-monotone

monotonicity-property(0, 'pullup, netlist, a1, a2, s1, s2)
∧ monotonicity-property(2, 'pullup, netlist, a1, a2, s1, s2)

THEOREM: dual-eval-ttl-bidirect-value

dual-eval(0, 'ttl-bidirect, args, state, netlist)
= list(ft-buf(f-not(caddr(args)), cadr(args)),
f-buf(ft-wire(car(args), ft-buf(f-not(caddr(args)), cadr(args)))),
f-nand(ft-wire(car(args), ft-buf(f-not(caddr(args)), cadr(args))),
caddr(args)))

THEOREM: dual-eval-ttl-bidirect-state
 $\text{dual-eval}(2, \text{'ttl-bidirect'}, args, state, netlist) = 0$

THEOREM: ttl-bidirect-monotone
 $\text{monotonicity-property}(0, \text{'ttl-bidirect'}, netlist, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'ttl-bidirect'}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ttl-clk-input-value
 $\text{dual-eval}(0, \text{'ttl-clk-input'}, args, state, netlist)$
 $= \text{list}(\text{f-buf}(\text{car}(args)), \text{f-nand}(\text{car}(args), \text{cadr}(args)))$

THEOREM: dual-eval-ttl-clk-input-state
 $\text{dual-eval}(2, \text{'ttl-clk-input'}, args, state, netlist) = 0$

THEOREM: ttl-clk-input-monotone
 $\text{monotonicity-property}(0, \text{'ttl-clk-input'}, netlist, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'ttl-clk-input'}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ttl-input-value
 $\text{dual-eval}(0, \text{'ttl-input'}, args, state, netlist)$
 $= \text{list}(\text{f-buf}(\text{car}(args)), \text{f-nand}(\text{car}(args), \text{cadr}(args)))$

THEOREM: dual-eval-ttl-input-state
 $\text{dual-eval}(2, \text{'ttl-input'}, args, state, netlist) = 0$

THEOREM: ttl-input-monotone
 $\text{monotonicity-property}(0, \text{'ttl-input'}, netlist, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'ttl-input'}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ttl-output-value
 $\text{dual-eval}(0, \text{'ttl-output'}, args, state, netlist) = \text{list}(\text{f-buf}(\text{car}(args)))$

THEOREM: dual-eval-ttl-output-state
 $\text{dual-eval}(2, \text{'ttl-output'}, args, state, netlist) = 0$

THEOREM: ttl-output-monotone
 $\text{monotonicity-property}(0, \text{'ttl-output'}, netlist, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'ttl-output'}, netlist, a1, a2, s1, s2)$

THEOREM: dual-eval-ttl-output-parametric-value
 $\text{dual-eval}(0, \text{'ttl-output-parametric'}, args, state, netlist)$
 $= \text{list}(\text{f-buf}(\text{car}(args)))$

THEOREM: dual-eval-ttl-output-parametric-state
 $\text{dual-eval}(2, \text{'ttl-output-parametric'}, args, state, netlist) = 0$

THEOREM: ttl-output-parametric-monotone
monotonicity-property (0, 'ttl-output-parametric, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2,
 'ttl-output-parametric,
 netlist,
 a1,
 a2,
 s1,
 s2)

THEOREM: dual-eval-ttl-output-fast-value
dual-eval (0, 'ttl-output-fast, *args*, *state*, *netlist*)
= list (f-buf (car (*args*)))

THEOREM: dual-eval-ttl-output-fast-state
dual-eval (2, 'ttl-output-fast, *args*, *state*, *netlist*) = 0

THEOREM: ttl-output-fast-monotone
monotonicity-property (0, 'ttl-output-fast, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'ttl-output-fast, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-tri-output-value
dual-eval (0, 'ttl-tri-output, *args*, *state*, *netlist*)
= list (ft-buf (f-not (cadr (*args*))), car (*args*)))

THEOREM: dual-eval-ttl-tri-output-state
dual-eval (2, 'ttl-tri-output, *args*, *state*, *netlist*) = 0

THEOREM: ttl-tri-output-monotone
monotonicity-property (0, 'ttl-tri-output, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2, 'ttl-tri-output, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-tri-output-fast-value
dual-eval (0, 'ttl-tri-output-fast, *args*, *state*, *netlist*)
= list (ft-buf (f-not (cadr (*args*))), car (*args*)))

THEOREM: dual-eval-ttl-tri-output-fast-state
dual-eval (2, 'ttl-tri-output-fast, *args*, *state*, *netlist*) = 0

THEOREM: ttl-tri-output-fast-monotone
monotonicity-property (0, 'ttl-tri-output-fast, *netlist*, *a1*, *a2*, *s1*, *s2*)
 $\wedge$ monotonicity-property (2,
 'ttl-tri-output-fast,
 netlist,
 a1,
 a2,
 s1,
 s2)

THEOREM: dual-eval-vdd-value
 $\text{dual-eval}(0, \text{'vdd}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\mathbf{t})$

THEOREM: dual-eval-vdd-state
 $\text{dual-eval}(2, \text{'vdd}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: vdd-monotone
 $\text{monotonicity-property}(0, \text{'vdd}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'vdd}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-vdd-parametric-value
 $\text{dual-eval}(0, \text{'vdd-parametric}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\mathbf{t})$

THEOREM: dual-eval-vdd-parametric-state
 $\text{dual-eval}(2, \text{'vdd-parametric}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: vdd-parametric-monotone
 $\text{monotonicity-property}(0, \text{'vdd-parametric}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'vdd-parametric}, \text{netlist}, a1, a2, s1, s2)$

THEOREM: dual-eval-vss-value
 $\text{dual-eval}(0, \text{'vss}, \text{args}, \text{state}, \text{netlist}) = \text{list}(\mathbf{f})$

THEOREM: dual-eval-vss-state
 $\text{dual-eval}(2, \text{'vss}, \text{args}, \text{state}, \text{netlist}) = 0$

THEOREM: vss-monotone
 $\text{monotonicity-property}(0, \text{'vss}, \text{netlist}, a1, a2, s1, s2)$
 $\wedge \text{monotonicity-property}(2, \text{'vss}, \text{netlist}, a1, a2, s1, s2)$

EVENT: Let us define the theory *primitives-monotone* to consist of the following events: ao2-monotone, ao4-monotone, ao6-monotone, ao7-monotone, b-and-monotone, b-and3-monotone, b-and4-monotone, b-equiv-monotone, b-equiv3-monotone, b-if-monotone, b-nand-monotone, b-nand3-monotone, b-nand4-monotone, b-nand5-monotone, b-nand6-monotone, b-nand8-monotone, b-nbuf-monotone, b-nor-monotone, b-nor3-monotone, b-nor4-monotone, b-nor5-monotone, b-nor6-monotone, b-nor8-monotone, b-not-monotone, b-not-b4ip-monotone, b-not-ivap-monotone, b-or-monotone, b-or3-monotone, b-or4-monotone, b-xor-monotone, b-xor3-monotone, del2-monotone, del4-monotone, del10-monotone, procmon-monotone, fd1-monotone, fd1s-monotone, fd1sp-monotone, fd1slp-monotone, id-monotone, ram-enable-circuit-monotone, t-buf-monotone, t-wire-monotone, pullup-monotone, ttl-bidirect-monotone, ttl-clk-input-monotone, ttl-input-monotone, ttl-output-monotone, ttl-output-parametric-monotone, ttl-output-fast-monotone, ttl-tri-output-monotone, ttl-tri-output-fast-monotone, vdd-monotone, vdd-parametric-monotone, vss-monotone.

THEOREM: bvp-rev1
 $(\text{bvp}(x) \wedge \text{bvp}(acc)) \rightarrow \text{bvp}(\text{rev1}(x, acc))$

THEOREM: bvp-reverse
 $\text{bvp}(x) \rightarrow \text{bvp}(\text{reverse}(x))$

THEOREM: v-approx-bvp
 $\text{bvp}(x) \rightarrow (\text{v-approx}(x, y) = (x = y))$

DEFINITION:
 $\text{read-mem1-monotone-induction}(v\text{-addr}, mem1, mem2)$
 $=$ **if** $\text{stulp}(mem1)$
 $\quad \vee \text{stulp}(mem2)$
 $\quad \vee (v\text{-addr} \simeq \mathbf{nil})$
 $\quad \vee (mem1 \simeq \mathbf{nil})$
 $\quad \vee (mem2 \simeq \mathbf{nil})$ **then t**
elseif $\text{car}(v\text{-addr})$
then $\text{read-mem1-monotone-induction}(\text{cdr}(v\text{-addr}), \text{cdr}(mem1), \text{cdr}(mem2))$
else $\text{read-mem1-monotone-induction}(\text{cdr}(v\text{-addr}),$
 $\quad \text{car}(mem1),$
 $\quad \text{car}(mem2))$ **endif**

THEOREM: read-mem1-monotone
 $(\text{bvp}(v\text{-addr}) \wedge \text{s-approx}(mem1, mem2))$
 $\rightarrow \text{v-approx}(\text{read-mem1}(v\text{-addr}, mem1), \text{read-mem1}(v\text{-addr}, mem2))$

THEOREM: bvp-implies-v-knownp
 $\text{bvp}(x) \rightarrow \text{v-knownp}(x)$

THEOREM: v-approx-x-x
 $\text{v-approx}(x, x)$

THEOREM: v-approx-implies-b-approx-nth
 $((\text{nth}(n, x) = c) \wedge (\text{nth}(n, y) \neq c) \wedge \text{b-knownp}(c))$
 $\rightarrow (\neg \text{v-approx}(x, y))$

EVENT: Disable *1*b-knownp; name this event 'g*1*b-knownp-on'.

THEOREM: v-approx-implies-subranges-equal
 $(\text{v-approx}(x, y) \wedge \text{bvp}(\text{subrange}(x, i, j)))$
 $\rightarrow (\text{subrange}(x, i, j) = \text{subrange}(y, i, j))$

THEOREM: v-approx-bvp-subrange
 $(\text{v-approx}(a1, a2) \wedge \text{bvp}(\text{subrange}(a1, i, j))) \rightarrow \text{bvp}(\text{subrange}(a2, i, j))$

THEOREM: v-approx-make-list-x

$((\text{length}(y) = \text{bits}) \wedge \text{properp}(y)) \rightarrow \text{v-approx}(\text{make-list}(\text{bits}, x), y)$

THEOREM: read-mem-monotone

$(\text{bvp}(v\text{-addr}) \wedge \text{s-approx}(mem1, mem2))$
 $\rightarrow \text{v-approx}(\text{read-mem}(v\text{-addr}, mem1), \text{read-mem}(v\text{-addr}, mem2))$

THEOREM: equal-length-read-mem1

$\text{s-approx}(s1, s2)$
 $\rightarrow ((\text{length}(\text{read-mem1}(a, s2)) = \text{length}(\text{read-mem1}(a, s1))) = \mathbf{t})$

THEOREM: equal-length-read-mem

$\text{s-approx}(s1, s2)$
 $\rightarrow ((\text{length}(\text{read-mem}(a, s2)) = \text{length}(\text{read-mem}(a, s1))) = \mathbf{t})$

THEOREM: s-approx-implies-properp-read-mem1

$(\text{s-approx}(s1, s2) \wedge \text{properp}(\text{read-mem1}(a, s1))) \rightarrow \text{properp}(\text{read-mem1}(a, s2))$

THEOREM: s-approx-implies-properp-read-mem

$(\text{s-approx}(s1, s2) \wedge \text{properp}(\text{read-mem}(a, s1))) \rightarrow \text{properp}(\text{read-mem}(a, s2))$

THEOREM: dual-port-ram-value-monotone

$((\text{address-lines} \in \mathbf{N}) \wedge \text{s-approx}(s1, s2) \wedge \text{v-approx}(a1, a2))$
 $\rightarrow \text{v-approx}(\text{dual-port-ram-value}(\text{bits}, \text{address-lines}, a1, s1),$
 $\text{dual-port-ram-value}(\text{bits}, \text{address-lines}, a2, s2))$

THEOREM: eval\$-append

$(\text{litatom}(x) \wedge (x \in lst))$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, \text{append}(\text{pairlist}(lst, y), z)) = \text{eval}\$(\mathbf{t}, x, \text{pairlist}(lst, y)))$

THEOREM: eval\$-pairlist-cons

$\text{litatom}(x)$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, \text{pairlist}(\text{cons}(c, lst), y))$
 $= \text{if } x = c \text{ then } \text{car}(y)$
 $\text{else } \text{eval}\$(\mathbf{t}, x, \text{pairlist}(lst, \text{cdr}(y))) \text{ endif})$

THEOREM: eval\$-append-2

$(\text{litatom}(x) \wedge (x \notin lst))$
 $\rightarrow (\text{eval}\$(\mathbf{t}, x, \text{append}(\text{pairlist}(lst, y), z)) = \text{eval}\$(\mathbf{t}, x, z))$

THEOREM: dual-apply-value-dp-ram-16x32-lemma-1

$\text{dual-apply-value}(\text{'dp-ram-16x32}, a, s)$
 $= \text{eval}\$(\mathbf{t},$
 $\text{cdr}(\text{lookup-module}(\text{'results}, \text{cdr}(\text{primp}(\text{'dp-ram-16x32})))),$
 $\text{append}(\text{pairlist}(\text{cdr}(\text{lookup-module}(\text{'inputs},$
 $\text{cdr}(\text{primp}(\text{'dp-ram-16x32})))),$

```

a),
pairstates(cdr(lookup-module('states,
                                cdr(primp('dp-ram-16x32)))),
s)))

```

THEOREM: dual-apply-value-dp-ram-16x32-lemma-2

```

dual-apply-value('dp-ram-16x32, a, s)
=  dual-port-ram-value(32,
4,
list(eval$(t,
' read-a0,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
' read-a1,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
' read-a2,
append(pairlist(' (read-a0 read-a1

```

```

read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'read-a3,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'write-b0,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),

```

```

                                pairstates('state, s))),
eval$(t,
      'write-b1,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                        a),
                                pairstates('state, s))),
eval$(t,
      'write-b2,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                        a),
                                pairstates('state, s))),
eval$(t,
      'write-b3,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20

```

```

                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'wen,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd0,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd1,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3

```

```

d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd2,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd3,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd4,
append(pairlist('(read-a0 read-a1

```

```

read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd5,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd6,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),

```



```

                                pairstates('state, s))),
eval$(t,
      'd7,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd8,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd9,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20

```

```

                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd10,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd11,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd12,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3

```

```

d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd13,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd14,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd15,
append(pairlist('(read-a0 read-a1

```

```

read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd16,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd17,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),

```

```

                                pairstates('state, s))),
eval$(t,
      'd18,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd19,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd20,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20

```

```

                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd21,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd22,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd23,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3

```

```

d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd24,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd25,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd26,
append(pairlist('(read-a0 read-a1

```

```

read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd27,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd28,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),

```



```

                                pairstates('state, s))),
eval$(t,
      'd29,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd30,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd31,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20

```

```

                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))))),
eval$(t,
      'state,
      append(pairlist('(read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3 d4
                        d5 d6 d7 d8 d9 d10
                        d11 d12 d13 d14 d15
                        d16 d17 d18 d19 d20
                        d21 d22 d23 d24 d25
                        d26 d27 d28 d29 d30
                        d31),
                        a),
                        pairstates('state, s))))))

```

THEOREM: dual-apply-value-dp-ram-16x32-lemma-3

```
dual-apply-value ('dp-ram-16x32, a, s)
```

$$= \text{dual-port-ram-value}(32,$$
[illegible]

THEOREM: dual-apply-value-dp-ram-16x32
dual-apply-value ('dp-ram-16x32, a , s) = dual-port-ram-value (32, 4, a , s)

THEOREM: dp-ram-16x32-monotone-value
monotonicity-property (0, 'dp-ram-16x32, $netlist$, $a1$, $a2$, $s1$, $s2$)

EVENT: Enable v-knownp; name this event 'v-knownp-off'.

EVENT: Enable s-knownp-implies-s-approx-is-equal; name this event 's-knownp-implies-s-approx-is-equal-off'.

EVENT: Enable bvp; name this event 'bvp-off1'.

EVENT: Enable fourp-implies-s-approx-is-b-approx; name this event 'fourp-implies-s-approx-is-b-approx-off'.

EVENT: Enable v-approx; name this event 'v-approx-off'.

EVENT: Enable s-knownp; name this event 's-knownp-off'.

THEOREM: s-approx-opener
 $((\text{listp}(s1) \vee \text{listp}(s2))$
 $\rightarrow (\text{s-approx}(s1, s2)$
 $= \text{if listp}(s1)$
 $\text{then if listp}(s2)$
 $\text{then s-approx}(\text{car}(s1), \text{car}(s2))$
 $\wedge \text{s-approx}(\text{cdr}(s1), \text{cdr}(s2))$
 else f endif
 $\text{else f endif}))$
 $\wedge (((s1 = \text{nil}) \vee (s2 = \text{nil}))$
 $\rightarrow (\text{s-approx}(s1, s2) = (s1 = s2)))$
 $\wedge ((\text{ramp}(s1) \vee \text{ramp}(s2))$
 $\rightarrow (\text{s-approx}(s1, s2)$
 $= \text{if ramp}(s1)$
 $\text{then if ramp}(s2)$
 $\text{then v-approx}(\text{ram-guts}(s1), \text{ram-guts}(s2))$
 else f endif
 $\text{else f endif}))$
 $\wedge ((\text{romp}(s1) \vee \text{romp}(s2))$
 $\rightarrow (\text{s-approx}(s1, s2)$
 $= \text{if romp}(s1)$

```

      then if romp (s2)
        then v-approx (rom-guts (s1), rom-guts (s2))
        else f endif
      else f endif))
 $\wedge$  ((stulp (s1)  $\vee$  stulp (s2))
   $\rightarrow$  (s-approx (s1, s2)
    = if stulp (s1)
      then if stulp (s2)
        then v-approx (stub-guts (s1), stub-guts (s2))
        else f endif
      else f endif))
 $\wedge$  (( $\neg$  (listp (s1)
   $\vee$  listp (s2)
   $\vee$  (s1 = nil)
   $\vee$  (s2 = nil)
   $\vee$  ramp (s1)
   $\vee$  ramp (s2)
   $\vee$  romp (s1)
   $\vee$  romp (s2)
   $\vee$  stulp (s1)
   $\vee$  stulp (s2)))
   $\rightarrow$  (s-approx (s1, s2) = b-approx (s1, s2)))

```

EVENT: Enable s-approx; name this event ‘s-approx-off’.

EVENT: Enable *1*mem-width; name this event ‘g*1*mem-width-off’.

EVENT: Enable mem-width; name this event ‘mem-width-off’.

THEOREM: v-approx-implies-nth-does-not-go-from-f-to-t
 $((\neg \text{nth}(n, a1)) \wedge \text{nth}(n, a2)) \rightarrow (\neg \text{v-approx}(a1, a2))$

THEOREM: write-mem1-opener
 $(\text{stulp}(mem) \rightarrow (\text{write-mem1}(v\text{-addr}, mem, value) = mem))$
 $\wedge$ ((v-addr $\simeq$ nil)
 $\rightarrow$ (write-mem1 (v-addr, mem, value)
 = if ramp (mem) then ram (value)
 else mem endif))
 $\wedge$ ((listp (v-addr) $\wedge$ (mem $\simeq$ nil))
 $\rightarrow$ (write-mem1 (v-addr, mem, value) = mem))
 $\wedge$ ((listp (mem) $\wedge$ listp (v-addr) $\wedge$ car (v-addr))
 $\rightarrow$ (write-mem1 (v-addr, mem, value)
 = cons (car (mem), write-mem1 (cdr (v-addr), cdr (mem), value))))

$$\begin{aligned}
& \wedge ((\text{listp}(mem) \wedge \text{listp}(v\text{-}addr) \wedge (\neg \text{car}(v\text{-}addr))) \\
& \quad \rightarrow (\text{write-mem1}(v\text{-}addr, mem, value) \\
& \quad = \text{cons}(\text{write-mem1}(\text{cdr}(v\text{-}addr), \text{car}(mem), value), \text{cdr}(mem))))
\end{aligned}$$

DEFINITION:

write-mem1-monotone-induction($v\text{-}addr, mem1, mem2$)
= **if** stubp($mem1$) $\vee$ stubp($mem2$) **then t**
 elseif $v\text{-}addr \simeq \text{nil}$ **then t**
 elseif ($mem1 \simeq \text{nil}$) $\vee$ ($mem2 \simeq \text{nil}$) **then t**
 elseif car($v\text{-}addr$)
 then write-mem1-monotone-induction(cdr($v\text{-}addr$), cdr($mem1$), cdr($mem2$))
 else write-mem1-monotone-induction(cdr($v\text{-}addr$),
 car($mem1$),
 car($mem2$)) **endif**

THEOREM: write-mem1-monotone

(s-approx($mem1, mem2$)
 $\wedge$ v-approx($data1, data2$)
 $\wedge$ properp($data1$)
 $\wedge$ properp($data2$)
 $\wedge$ (length($data1$) = MEM-WIDTH)
 $\wedge$ (length($data2$) = MEM-WIDTH))
 $\rightarrow$ s-approx(write-mem1($v\text{-}addr, mem1, data1$), write-mem1($v\text{-}addr, mem2, data2$))

THEOREM: write-mem-monotone

(s-approx($mem1, mem2$)
 $\wedge$ v-approx($data1, data2$)
 $\wedge$ properp($data1$)
 $\wedge$ properp($data2$)
 $\wedge$ (length($data1$) = MEM-WIDTH)
 $\wedge$ (length($data2$) = MEM-WIDTH))
 $\rightarrow$ s-approx(write-mem($v\text{-}addr, mem1, data1$), write-mem($v\text{-}addr, mem2, data2$))

THEOREM: v-approx-length

v-approx($v1, v2$) $\rightarrow$ (length($v1$) $\nless$ length($v2$))

DEFINITION:

write-mem1-double-induction($v\text{-}addr, mem1, mem2$)
= **if** stubp($mem1$) $\vee$ stubp($mem2$) **then t**
 elseif $v\text{-}addr \simeq \text{nil}$ **then t**
 elseif ($mem1 \simeq \text{nil}$) $\vee$ ($mem2 \simeq \text{nil}$) **then t**
 elseif car($v\text{-}addr$)
 then write-mem1-double-induction(cdr($v\text{-}addr$), cdr($mem1$), cdr($mem2$))
 else write-mem1-double-induction(cdr($v\text{-}addr$),
 car($mem1$),
 car($mem2$)) **endif**

THEOREM: s-approx-write-mem1-id
 $(\text{s-approx}(s1, s2) \wedge \text{good-s}(s2))$
 $\rightarrow \text{s-approx}(\text{write-mem1}(v\text{-addr}, s1, \text{make-list}(\text{MEM-WIDTH}, x)), s2)$

THEOREM: s-approx-write-mem-id
 $(\text{good-s}(s2) \wedge \text{s-approx}(s1, s2))$
 $\rightarrow \text{s-approx}(\text{write-mem}(v\text{-addr}, s1, \text{make-list}(\text{MEM-WIDTH}, x)), s2)$

THEOREM: s-approx-constant-ram-x-id
 $(\text{good-s}(s2) \wedge \text{s-approx}(s1, s2))$
 $\rightarrow \text{s-approx}(\text{constant-ram}(s1, \text{make-list}(\text{MEM-WIDTH}, x)), s2)$

THEOREM: s-approx-constant-ram-x-constant-ram-x
 $\text{s-approx}(s1, s2)$
 $\rightarrow \text{s-approx}(\text{constant-ram}(s1, \text{make-list}(\text{MEM-WIDTH}, x)),$
 $\text{constant-ram}(s2, \text{make-list}(\text{MEM-WIDTH}, x)))$

THEOREM: v-approx-preserves-length
 $\text{v-approx}(a1, a2) \rightarrow (\text{length}(a1) = \text{length}(a2))$

THEOREM: v-approx-subrange
 $\text{v-approx}(a1, a2) \rightarrow \text{v-approx}(\text{subrange}(a1, i, j), \text{subrange}(a2, i, j))$

THEOREM: mem-width-non-zero
 $\text{MEM-WIDTH} \neq 0$

THEOREM: s-approx-constant-ram-x-write-mem1-case-3
 $((\neg \text{stubp}(s1))$
 $\wedge (\neg \text{stubp}(s2))$
 $\wedge \text{listp}(v\text{-addr})$
 $\wedge (\neg \text{listp}(s1))$
 $\wedge \text{s-approx}(s1, s2)$
 $\wedge \text{properp}(v)$
 $\wedge (\text{length}(v) = \text{MEM-WIDTH})$
 $\wedge (\neg \text{ramp}(s1)))$
 $\rightarrow \text{s-approx}(s1, \text{write-mem1}(v\text{-addr}, s2, v))$

THEOREM: s-approx-constant-ram-x-write-mem1
 $(\text{s-approx}(s1, s2) \wedge \text{properp}(v) \wedge (\text{length}(v) = \text{MEM-WIDTH}) \wedge \text{good-s}(s2))$
 $\rightarrow \text{s-approx}(\text{constant-ram}(s1, \text{make-list}(\text{MEM-WIDTH}, x)),$
 $\text{write-mem1}(v\text{-addr}, s2, v))$

THEOREM: s-approx-constant-ram-x-write-mem
 $(\text{s-approx}(s1, s2) \wedge \text{properp}(v) \wedge (\text{length}(v) = \text{MEM-WIDTH}) \wedge \text{good-s}(s2))$
 $\rightarrow \text{s-approx}(\text{constant-ram}(s1, \text{make-list}(\text{MEM-WIDTH}, x)),$
 $\text{write-mem}(v\text{-addr}, s2, v))$

THEOREM: v-approx-preserves-properp
 $(\neg (\text{properp}(v1) \leftrightarrow \text{properp}(v2))) \rightarrow (\neg \text{v-approx}(v1, v2))$

THEOREM: mem-width-linear-facts
 $(31 < \text{MEM-WIDTH}) \wedge (\text{MEM-WIDTH} < 33)$

THEOREM: dual-port-ram-state-monotone
 $(\text{s-approx}(s1, s2) \wedge \text{good-s}(s2) \wedge \text{v-approx}(a1, a2))$
 $\rightarrow \text{s-approx}(\text{dual-port-ram-state}(\text{MEM-WIDTH}, 4, a1, s1),$
 $\text{dual-port-ram-state}(\text{MEM-WIDTH}, 4, a2, s2))$

THEOREM: dual-port-ram-state-monotone-rewrite
 $(\text{s-approx}(s1, s2) \wedge \text{good-s}(s2) \wedge \text{v-approx}(a1, a2))$
 $\rightarrow \text{s-approx}(\text{dual-port-ram-state}(32, 4, a1, s1),$
 $\text{dual-port-ram-state}(32, 4, a2, s2))$

THEOREM: dual-apply-state-dp-ram-16x32-lemma-1
 $\text{dual-apply-state}('dp-ram-16x32, a, s)$
 $= \text{eval\$}(\mathbf{t},$
 $\text{cdr}(\text{lookup-module}('new-states, \text{cdr}(\text{primp}('dp-ram-16x32))))),$
 $\text{append}(\text{pairlist}(\text{cdr}(\text{lookup-module}('inputs,$
 $\text{cdr}(\text{primp}('dp-ram-16x32))))),$
 $a),$
 $\text{pairstates}(\text{cdr}(\text{lookup-module}('states,$
 $\text{cdr}(\text{primp}('dp-ram-16x32))))),$
 $s)))$

THEOREM: dual-apply-state-dp-ram-16x32-lemma-2
 $\text{dual-apply-state}('dp-ram-16x32, a, s)$
 $= \text{dual-port-ram-state}(32,$
 $4,$
 $\text{list}(\text{eval\$}(\mathbf{t},$
 $'read-a0,$
 $\text{append}(\text{pairlist}('(\text{read-a0 read-a1}$
 read-a2 read-a3
 write-b0 write-b1
 write-b2 write-b3
 wen d0 d1 d2 d3
 d4 d5 d6 d7 d8
 d9 d10 d11 d12
 d13 d14 d15 d16
 d17 d18 d19 d20
 d21 d22 d23 d24
 d25 d26 d27 d28
 $\text{d29 d30 d31}),$


```

                                a),
                                pairstates('state, s))),
eval$(t,
      'read-a1,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'read-a2,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'read-a3,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16

```

```

d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'write-b0,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'write-b1,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'write-b2,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3

```

```

wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'write-b3,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'wen,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd0,
```

```

append(pairlist(' (read-a0 read-a1
                  read-a2 read-a3
                  write-b0 write-b1
                  write-b2 write-b3
                  wen d0 d1 d2 d3
                  d4 d5 d6 d7 d8
                  d9 d10 d11 d12
                  d13 d14 d15 d16
                  d17 d18 d19 d20
                  d21 d22 d23 d24
                  d25 d26 d27 d28
                  d29 d30 d31),
          a),
        pairstates('state, s))),
eval$(t,
      'd1,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                  a),
            pairstates('state, s))),
eval$(t,
      'd2,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                  a),
            pairstates('state, s))),

```

```

                                a),
                                pairstates('state, s))),
eval$(t,
      'd3,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd4,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd5,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16

```

```

                                d17 d18 d19 d20
                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd6,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd7,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd8,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3

```

```

wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd9,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd10,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd11,

```

```

append(pairlist(' (read-a0 read-a1
                  read-a2 read-a3
                  write-b0 write-b1
                  write-b2 write-b3
                  wen d0 d1 d2 d3
                  d4 d5 d6 d7 d8
                  d9 d10 d11 d12
                  d13 d14 d15 d16
                  d17 d18 d19 d20
                  d21 d22 d23 d24
                  d25 d26 d27 d28
                  d29 d30 d31),
          a),
        pairstates('state, s))),
eval$(t,
      'd12,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                      a),
            pairstates('state, s))),
eval$(t,
      'd13,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                      a),
            pairstates('state, s))),

```



```

                                a),
                                pairstates('state, s))),
eval$(t,
      'd14,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd15,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd16,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16

```

```

                                d17 d18 d19 d20
                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd17,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd18,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd19,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3

```

```

wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd20,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd21,
append(pairlist('(read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd22,

```

```

append(pairlist(' (read-a0 read-a1
                  read-a2 read-a3
                  write-b0 write-b1
                  write-b2 write-b3
                  wen d0 d1 d2 d3
                  d4 d5 d6 d7 d8
                  d9 d10 d11 d12
                  d13 d14 d15 d16
                  d17 d18 d19 d20
                  d21 d22 d23 d24
                  d25 d26 d27 d28
                  d29 d30 d31),
          a),
        pairstates('state, s))),
eval$(t,
      'd23,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                      a),
            pairstates('state, s))),
eval$(t,
      'd24,
      append(pairlist(' (read-a0 read-a1
                        read-a2 read-a3
                        write-b0 write-b1
                        write-b2 write-b3
                        wen d0 d1 d2 d3
                        d4 d5 d6 d7 d8
                        d9 d10 d11 d12
                        d13 d14 d15 d16
                        d17 d18 d19 d20
                        d21 d22 d23 d24
                        d25 d26 d27 d28
                        d29 d30 d31),
                      a),
            pairstates('state, s))),

```

```

                                a),
                                pairstates('state, s))),
eval$(t,
      'd25,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd26,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd27,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16

```

```

                                d17 d18 d19 d20
                                d21 d22 d23 d24
                                d25 d26 d27 d28
                                d29 d30 d31),
                                a),
                                pairstates('state, s))),
eval$(t,
      'd28,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd29,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3
                          wen d0 d1 d2 d3
                          d4 d5 d6 d7 d8
                          d9 d10 d11 d12
                          d13 d14 d15 d16
                          d17 d18 d19 d20
                          d21 d22 d23 d24
                          d25 d26 d27 d28
                          d29 d30 d31),
                          a),
                          pairstates('state, s))),
eval$(t,
      'd30,
      append(pairlist(' (read-a0 read-a1
                          read-a2 read-a3
                          write-b0 write-b1
                          write-b2 write-b3

```

```

wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'd31,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3
d4 d5 d6 d7 d8
d9 d10 d11 d12
d13 d14 d15 d16
d17 d18 d19 d20
d21 d22 d23 d24
d25 d26 d27 d28
d29 d30 d31),
a),
pairstates('state, s))),
eval$(t,
'state,
append(pairlist(' (read-a0 read-a1
read-a2 read-a3
write-b0 write-b1
write-b2 write-b3
wen d0 d1 d2 d3 d4
d5 d6 d7 d8 d9 d10
d11 d12 d13 d14 d15
d16 d17 d18 d19 d20
d21 d22 d23 d24 d25
d26 d27 d28 d29 d30
d31),
a),
pairstates('state, s))))

```

THEOREM: dual-apply-state-dp-ram-16x32-lemma-3
dual-apply-state('dp-ram-16x32, a, s)

DEFINITION:

```

dual-port-ram-state-body (bits, b-address, wen, data, state)
=  if wen = t then state
    elseif  $\neg$  bvp (b-address) then constant-ram (state, make-list (bits, x))
    elseif wen = f then write-mem (b-address, state, data)
    else write-mem (b-address, state, make-list (bits, x)) endif

```

THEOREM: dual-port-ram-state-is-dual-port-ram-state-body

```

dual-port-ram-state (bits, address-lines, args, state)
=  dual-port-ram-state-body (bits,
                               subrange (args,
                                           address-lines,
                                           (2 * address-lines) - 1),
                               nth (2 * address-lines, args),
                               subrange (args,
                                           1 + (2 * address-lines),
                                           (2 * address-lines) + bits),
                               state)

```

THEOREM: dual-apply-state-dp-ram-16x32

```

dual-apply-state ('dp-ram-16x32', a, s) = dual-port-ram-state (32, 4, a, s)

```

THEOREM: dp-ram-16x32-monotone-state

```

good-s (s2)
→  monotonicity-property (2, 'dp-ram-16x32', netlist, a1, a2, s1, s2)

```

THEOREM: dp-ram-16x32-monotone

```

monotonicity-property (0, 'dp-ram-16x32', netlist, a1, a2, s1, s2)
∧  (good-s (s2)
    →  monotonicity-property (2,
                                'dp-ram-16x32',
                                netlist,
                                a1,
                                a2,
                                s1,
                                s2))

```

EVENT: Enable *bvp-reverse*; name this event '*bvp-reverse-off*'.

EVENT: Enable *v-approx-bvp*; name this event '*v-approx-bvp-off*'.

EVENT: Enable *read-mem1-monotone*; name this event '*read-mem1-monotone-off*'.

EVENT: Enable bvp-implies-v-knownp; name this event 'bvp-implies-v-knownp-off'.

EVENT: Enable v-approx-x-x; name this event 'v-approx-x-x-off'.

EVENT: Enable v-approx-implies-b-approx-nth; name this event 'v-approx-implies-b-approx-nth-off'.

EVENT: Enable $*1*b$ -knownp; name this event ' $g*1*b$ -knownp-off1'.

EVENT: Enable v-approx-implies-subranges-equal; name this event 'v-approx-implies-subranges-equal-off'.

EVENT: Enable v-approx-bvp-subrange; name this event 'v-approx-bvp-subrange-off'.

EVENT: Enable v-approx-make-list-x; name this event 'v-approx-make-list-x-off'.

EVENT: Enable read-mem-monotone; name this event 'read-mem-monotone-off'.

EVENT: Enable equal-length-read-mem1; name this event 'equal-length-read-mem1-off'.

EVENT: Enable equal-length-read-mem; name this event 'equal-length-read-mem-off'.

EVENT: Enable s-approx-implies-properp-read-mem1; name this event 's-approx-implies-properp-read-mem1-off'.

EVENT: Enable s-approx-implies-properp-read-mem; name this event 's-approx-implies-properp-read-mem-off'.

EVENT: Enable dual-port-ram-value-monotone; name this event 'dual-port-ram-value-monotone-off'.

EVENT: Enable eval\$-append; name this event ‘eval\$-append-off’.

EVENT: Enable eval\$-pairlist-cons; name this event ‘eval\$-pairlist-cons-off’.

EVENT: Enable eval\$-append-2; name this event ‘eval\$-append-2-off’.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-1; name this event ‘dual-apply-value-dp-ram-16x32-lemma-1-off’.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-2; name this event ‘dual-apply-value-dp-ram-16x32-lemma-2-off’.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-3; name this event ‘dual-apply-value-dp-ram-16x32-lemma-3-off’.

EVENT: Enable dual-port-ram-value-is-dual-port-ram-value-body; name this event ‘dual-port-ram-value-is-dual-port-ram-value-body-off’.

EVENT: Enable dual-apply-value-dp-ram-16x32; name this event ‘dual-apply-value-dp-ram-16x32-off’.

EVENT: Enable dp-ram-16x32-monotone-value; name this event ‘dp-ram-16x32-monotone-value-off’.

EVENT: Disable v-knownp; name this event ‘v-knownp-on’.

EVENT: Disable s-knownp-implies-s-approx-is-equal; name this event ‘s-knownp-implies-s-approx-is-equal-on’.

EVENT: Disable bvp; name this event ‘bvp-on’.

EVENT: Disable fourp-implies-s-approx-is-b-approx; name this event ‘fourp-implies-s-approx-is-b-approx-on’.

EVENT: Disable v-approx; name this event ‘v-approx-on’.

EVENT: Disable s-knownp; name this event ‘s-knownp-on’.

EVENT: Enable s-approx-opener; name this event ‘s-approx-opener-off’.

EVENT: Disable s-approx; name this event ‘s-approx-on’.

EVENT: Disable *1*mem-width; name this event ‘g*1*mem-width-on’.

EVENT: Disable mem-width; name this event ‘mem-width-on’.

EVENT: Enable v-approx-implies-nth-does-not-go-from-f-to-t; name this event ‘v-approx-implies-nth-does-not-go-from-f-to-t-off’.

EVENT: Enable write-mem1-opener; name this event ‘write-mem1-opener-off’.

EVENT: Enable write-mem1-monotone; name this event ‘write-mem1-monotone-off’.

EVENT: Enable write-mem-monotone; name this event ‘write-mem-monotone-off’.

EVENT: Enable v-approx-length; name this event ‘v-approx-length-off’.

EVENT: Enable s-approx-write-mem1-id; name this event ‘s-approx-write-mem1-id-off’.

EVENT: Enable s-approx-write-mem-id; name this event ‘s-approx-write-mem-id-off’.

EVENT: Enable s-approx-constant-ram-x-id; name this event ‘s-approx-constant-ram-x-id-off’.

EVENT: Enable s-approx-constant-ram-x-constant-ram-x; name this event ‘s-approx-constant-ram-x-constant-ram-x-off’.

EVENT: Enable v-approx-preserves-length; name this event ‘v-approx-preserves-length-off’.

EVENT: Enable v-approx-subrange; name this event ‘v-approx-subrange-off’.

EVENT: Enable mem-width-non-zero; name this event ‘mem-width-non-zero-off’.

EVENT: Enable s-approx-constant-ram-x-write-mem1-case-3; name this event ‘s-approx-constant-ram-x-write-mem1-case-3-off’.

EVENT: Enable s-approx-constant-ram-x-write-mem1; name this event ‘s-approx-constant-ram-x-write-mem1-off’.

EVENT: Enable s-approx-constant-ram-x-write-mem; name this event ‘s-approx-constant-ram-x-write-mem-off’.

EVENT: Enable v-approx-preserves-properp; name this event ‘v-approx-preserves-properp-off’.

EVENT: Enable mem-width-linear-facts; name this event ‘mem-width-linear-facts-off’.

EVENT: Enable dual-port-ram-state-monotone-rewrite; name this event ‘dual-port-ram-state-monotone-rewrite-off’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-1; name this event ‘dual-apply-state-dp-ram-16x32-lemma-1-off’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-2; name this event ‘dual-apply-state-dp-ram-16x32-lemma-2-off’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-3; name this event ‘dual-apply-state-dp-ram-16x32-lemma-3-off’.

EVENT: Enable dual-port-ram-state-is-dual-port-ram-state-body; name this event ‘dual-port-ram-state-is-dual-port-ram-state-body-off’.

EVENT: Enable dual-apply-state-dp-ram-16x32; name this event ‘dual-apply-state-dp-ram-16x32-off’.

EVENT: Enable dp-ram-16x32-monotone-state; name this event ‘dp-ram-16x32-monotone-state-off’.

EVENT: Enable dp-ram-16x32-monotone; name this event ‘dp-ram-16x32-monotone-off’.

DEFINITION:

```

ok-netlistp (flag, fn, netlist, exceptions)
=  case on flag:
    case = 0
    then if primp (fn) then fn ∉ exceptions
        elseif listp (lookup-module (fn, netlist))
            then ok-netlistp (1,
                                module-occurrences (lookup-module (fn, netlist)),
                                delete-module (fn, netlist),
                                exceptions)
        else f endif
    case = 1
    then if listp (fn)
        then ok-netlistp (0, occ-function (car (fn)), netlist, exceptions)
            ∧ ok-netlistp (1, cdr (fn), netlist, exceptions)
        else t endif
    case = 2
    then if primp (fn) then fn ∉ exceptions
        elseif listp (lookup-module (fn, netlist))
            then ok-netlistp (3,
                                module-occurrences (lookup-module (fn, netlist)),
                                delete-module (fn, netlist),
                                exceptions)
        else f endif
    case = 3
    then if listp (fn)
        then ok-netlistp (2, occ-function (car (fn)), netlist, exceptions)
            ∧ ok-netlistp (3, cdr (fn), netlist, exceptions)
        else t endif
    otherwise f endcase

```

THEOREM: primp-monotone

$$\begin{aligned}
 & (\text{primp } (fn)) \\
 & \wedge ((flag = 0) \\
 & \quad \vee ((flag = 2) \\
 & \quad \quad \wedge ((fn = \text{'dp-ram-16x32'} \rightarrow \text{good-s}(s2)))))) \\
 & \wedge (fn \neq \text{'mem-32x32'}) \\
 \rightarrow & (\text{monotonicity-property } (flag, fn, netlist, a1, a2, s1, s2) \\
 & \quad \wedge \text{monotonicity-property } (flag, fn, netlist, a1, a2, s1, s2))
 \end{aligned}$$

DEFINITION:

```

dual-eval-monotone-induction (flag, fn, netlist, a1, a2, s1, s2)
=  case on flag:
    case = 0
    then if primp (fn) then t
        elseif listp (lookup-module (fn, netlist))
            then dual-eval-monotone-induction (1,
                module-occurrences (lookup-module (fn,
                                                            netlist)),
                delete-module (fn, netlist),
                pairlist (module-inputs (lookup-module (fn,
                                                            netlist)),
                    a1),
                pairlist (module-inputs (lookup-module (fn,
                                                            netlist)),
                    a2),
                pairstates (module-statenames (lookup-module (fn,
                                                            netlist)),
                    s1),
                pairstates (module-statenames (lookup-module (fn,
                                                            netlist)),
                    s2))
            else t endif
    case = 1
    then if listp (fn)
        then dual-eval-monotone-induction (0,
            occ-function (car (fn)),
            netlist,
            collect-value (occ-inputs (car (fn)),
                a1),
            collect-value (occ-inputs (car (fn)),
                a2),
            value (occ-name (car (fn)), s1),
            value (occ-name (car (fn)), s2))
        ^ dual-eval-monotone-induction (1,
            cdr (fn),
            netlist,
            append (pairlist (occ-outputs (car (fn)),
                dual-eval (0,
                    occ-function (car (fn)),
                    collect-value (occ-inputs (car (fn)),
                        a1),
                    value (occ-name (car (fn)),
                        s1),
                    netlist)),
                    s2))

```

```

                                a1),
                                append (pairlist (occ-outputs (car (fn)),
                                dual-eval (0,
                                occ-function (car (fn)),
                                collect-value (occ-inputs (car (fn)),
                                a2),
                                value (occ-name (car (fn)),
                                s2),
                                netlist)),
                                a2),
                                s1,
                                s2)

                                else t endif
case = 2
then if primp (fn) then t
    elseif listp (lookup-module (fn, netlist))
    then dual-eval-monotone-induction (1,
                                module-occurrences (lookup-module (fn,
                                netlist)),
                                delete-module (fn, netlist),
                                pairlist (module-inputs (lookup-module (fn,
                                netlist)),
                                a1),
                                pairlist (module-inputs (lookup-module (fn,
                                netlist)),
                                a2),
                                pairstates (module-statenames (lookup-module (fn,
                                netlist)),
                                s1),
                                pairstates (module-statenames (lookup-module (fn,
                                netlist)),
                                s2))
    ^ dual-eval-monotone-induction (3,
                                module-occurrences (lookup-module (fn,
                                netlist)),
                                delete-module (fn,
                                netlist),
                                dual-eval (1,
                                module-occurrences (lookup-module (fn,
                                netlist)),
                                pairlist (module-inputs (lookup-module (fn,
                                netlist),
                                a1),
                                pairstates (module-statenames (lookup-module

```



```

                                s1),
                                delete-module (fn,
                                                netlist)),
dual-eval (1,
           module-occurrences (lookup-module (fn,
                                                netlist)),
           pairlist (module-inputs (lookup-module (fn,
                                                netlist),
                                                a2),
                     pairstates (module-statenames (lookup-module
                                                s2),
                                                delete-module (fn,
                                                                netlist)),
                     pairstates (module-statenames (lookup-module (fn,
                                                                netlist)),
                                s1),
                     pairstates (module-statenames (lookup-module (fn,
                                                                netlist)),
                                s2))
           s2))
else t endif
case = 3
then if listp (fn)
    then dual-eval-monotone-induction (2,
                                       occ-function (car (fn)),
                                       netlist,
                                       collect-value (occ-inputs (car (fn)),
                                                       a1),
                                       collect-value (occ-inputs (car (fn)),
                                                       a2),
                                       value (occ-name (car (fn)), s1),
                                       value (occ-name (car (fn)), s2))
    ∧ dual-eval-monotone-induction (3,
                                     cdr (fn),
                                     netlist,
                                     a1,
                                     a2,
                                     s1,
                                     s2)
    else t endif
otherwise t endcase

```

EVENT: Disable v-approx-x-x; name this event ‘v-approx-x-x-on’.

THEOREM: v-approx-alist-implies-b-approx-value
 $(\text{alistp}(\text{alist1}) \wedge \text{alistp}(\text{alist2}) \wedge \text{v-approx-alist}(\text{alist1}, \text{alist2}))$
 $\rightarrow \text{b-approx}(\text{value}(x, \text{alist1}), \text{value}(x, \text{alist2}))$

THEOREM: alistp-pairlist
 $\text{alistp}(\text{pairlist}(x, y))$

THEOREM: alistp-append
 $\text{alistp}(x) \rightarrow (\text{alistp}(\text{append}(x, y)) = \text{alistp}(y))$

THEOREM: alistp-dual-eval-1
 $(\text{alistp}(\text{bindings}) \wedge (\text{flag} = 1))$
 $\rightarrow \text{alistp}(\text{dual-eval}(\text{flag}, \text{occurrences}, \text{bindings}, \text{state-bindints}, \text{netlist}))$

THEOREM: v-approx-alist-implies-v-approx-collect-value
 $(\text{alistp}(\text{alist1}) \wedge \text{alistp}(\text{alist2}) \wedge \text{v-approx-alist}(\text{alist1}, \text{alist2}))$
 $\rightarrow \text{v-approx}(\text{collect-value}(x, \text{alist1}), \text{collect-value}(x, \text{alist2}))$

DEFINITION:
 $\text{s-approx-list}(x, y)$
 $=$ **if** $\text{listp}(x)$
 then $\text{listp}(y)$
 $\wedge \text{s-approx}(\text{car}(x), \text{car}(y))$
 $\wedge \text{s-approx-list}(\text{cdr}(x), \text{cdr}(y))$
 else $y \simeq \text{nil}$ **endif**

THEOREM: s-approx-list-implies-s-approx-alist
 $\text{s-approx-list}(s1, s2) \rightarrow \text{s-approx-alist}(\text{pairlist}(x, s1), \text{pairlist}(x, s2))$

THEOREM: v-approx-implies-v-approx-alist
 $\text{v-approx}(v1, v2) \rightarrow \text{v-approx-alist}(\text{pairlist}(x, v1), \text{pairlist}(x, v2))$

THEOREM: alistp-opener
 $\text{alistp}(\text{cons}(a, z)) = (\text{listp}(a) \wedge \text{alistp}(z))$

THEOREM: s-approx-alist-implies-s-approx-value
 $(\text{s-approx-alist}(s1, s2) \wedge \text{alistp}(s1) \wedge \text{alistp}(s2))$
 $\rightarrow \text{s-approx}(\text{value}(w, s1), \text{value}(w, s2))$

THEOREM: v-approx-alist-append
 $(\text{v-approx-alist}(a, c) \wedge \text{v-approx-alist}(b, d))$
 $\rightarrow \text{v-approx-alist}(\text{append}(a, b), \text{append}(c, d))$

DEFINITION:
 $\text{double-cdr-induction}(x, y)$
 $=$ **if** $\text{listp}(x) \wedge \text{listp}(y)$ **then** $\text{double-cdr-induction}(\text{cdr}(x), \text{cdr}(y))$
 else t **endif**

THEOREM: s-approx-alist-implies-s-approx-list-collect-value
 $(\text{s-approx-alist}(x, y) \wedge \text{alistp}(x) \wedge \text{alistp}(y))$
 $\rightarrow \text{s-approx-list}(\text{collect-value}(a, x), \text{collect-value}(a, y))$

THEOREM: alistp-dual-eval-3
 $(\text{alistp}(\text{state-bindings}) \wedge (\text{flag} = 3))$
 $\rightarrow \text{alistp}(\text{dual-eval}(\text{flag}, \text{occurrences}, \text{bindings}, \text{state-bindings}, \text{netlist}))$

THEOREM: s-approx-implies-s-approx-alist
 $\text{s-approx}(s1, s2) \rightarrow \text{s-approx-alist}(\text{pairlist}(x, s1), \text{pairlist}(x, s2))$

THEOREM: s-approx-alist-implies-s-approx-collect-value
 $(\text{s-approx-alist}(x, y) \wedge \text{alistp}(x) \wedge \text{alistp}(y))$
 $\rightarrow \text{s-approx}(\text{collect-value}(a, x), \text{collect-value}(a, y))$

THEOREM: ok-netlistp-reduction
 $((\text{flag} = 2) \vee (\text{flag} = 3))$
 $\rightarrow (\text{ok-netlistp}(\text{flag}, \text{fn}, \text{netlist}, \text{exceptions})$
 $\quad = \text{if } \text{flag} = 2 \text{ then ok-netlistp}(0, \text{fn}, \text{netlist}, \text{exceptions})$
 $\quad \text{else ok-netlistp}(1, \text{fn}, \text{netlist}, \text{exceptions}) \text{ endif})$

THEOREM: ok-netlistp-reduction-rewrite
 $(\text{ok-netlistp}(2, \text{fn}, \text{netlist}, \text{exceptions})$
 $\quad = \text{ok-netlistp}(0, \text{fn}, \text{netlist}, \text{exceptions}))$
 $\wedge (\text{ok-netlistp}(3, \text{fn}, \text{netlist}, \text{exceptions})$
 $\quad = \text{ok-netlistp}(1, \text{fn}, \text{netlist}, \text{exceptions}))$

THEOREM: dual-eval-monotone-no-ram
 $\text{ok-netlistp}(\text{flag}, \text{fn}, \text{netlist}, '(\text{mem-32x32 dp-ram-16x32}))$
 $\rightarrow \text{monotonicity-property}(\text{flag}, \text{fn}, \text{netlist}, a1, a2, s1, s2)$

DEFINITION:
 $\text{good-s-alist}(x)$
 $= \text{if listp}(x)$
 $\quad \text{then listp}(\text{car}(x)) \wedge \text{good-s}(\text{cdar}(x)) \wedge \text{good-s-alist}(\text{cdr}(x))$
 $\quad \text{else } x = \text{nil} \text{ endif}$

THEOREM: good-s-alist-pairlist
 $\text{good-s}(s) \rightarrow \text{good-s-alist}(\text{pairlist}(x, s))$

THEOREM: good-s-value
 $\text{good-s-alist}(x) \rightarrow \text{good-s}(\text{value}(w, x))$

THEOREM: dual-eval-monotone
 $\text{ok-netlistp}(\text{flag}, \text{fn}, \text{netlist}, '(\text{mem-32x32}))$
 $\rightarrow (\text{if } (\text{flag} = 0) \vee (\text{flag} = 2) \text{ then good-s}(s2)$
 $\quad \text{else good-s-alist}(s2) \text{ endif})$
 $\rightarrow \text{monotonicity-property}(\text{flag}, \text{fn}, \text{netlist}, a1, a2, s1, s2))$

THEOREM: good-s-collect-value
 $\text{good-s-alist}(s) \rightarrow \text{good-s}(\text{collect-value}(keys, s))$

THEOREM: f-buf-preserves-good-s
 $\text{good-s}(\text{f-buf}(x))$

THEOREM: f-if-preserves-good-s
 $\text{good-s}(\text{f-if}(x, y, z))$

THEOREM: good-s-0
 $\text{good-s}(0)$

THEOREM: ao2-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ao2, args, s))$

THEOREM: ao4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ao4, args, s))$

THEOREM: ao6-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ao6, args, s))$

THEOREM: ao7-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ao7, args, s))$

THEOREM: b-and-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-and, args, s))$

THEOREM: b-and3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-and3, args, s))$

THEOREM: b-and4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-and4, args, s))$

THEOREM: b-equiv-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-equiv, args, s))$

THEOREM: b-equiv3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-equiv3, args, s))$

THEOREM: b-if-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-if, args, s))$

THEOREM: b-nand-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-nand, args, s))$

THEOREM: b-nand3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b-nand3, args, s))$

THEOREM: b-nand4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nand4}, args, s))$

THEOREM: b-nand5-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nand5}, args, s))$

THEOREM: b-nand6-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nand6}, args, s))$

THEOREM: b-nand8-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nand8}, args, s))$

THEOREM: b-nbuf-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nbuf}, args, s))$

THEOREM: b-nor-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor}, args, s))$

THEOREM: b-nor3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor3}, args, s))$

THEOREM: b-nor4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor4}, args, s))$

THEOREM: b-nor5-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor5}, args, s))$

THEOREM: b-nor6-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor6}, args, s))$

THEOREM: b-nor8-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-nor8}, args, s))$

THEOREM: b-not-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-not}, args, s))$

THEOREM: b-not-b4ip-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-not-b4ip}, args, s))$

THEOREM: b-not-ivap-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-not-ivap}, args, s))$

THEOREM: b-or-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-or}, args, s))$

THEOREM: b-or3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-or3}, args, s))$

THEOREM: b-or4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-or4}, args, s))$

THEOREM: b-xor-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-xor}, args, s))$

THEOREM: b-xor3-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('b\text{-xor3}, args, s))$

THEOREM: del2-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('del2, args, s))$

THEOREM: del4-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('del4, args, s))$

THEOREM: del10-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('del10, args, s))$

THEOREM: procmon-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('procmon, args, s))$

THEOREM: fd1-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('fd1, args, s))$

THEOREM: fd1s-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('fd1s, args, s))$

THEOREM: fd1sp-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('fd1sp, args, s))$

THEOREM: fd1slp-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('fd1slp, args, s))$

THEOREM: id-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('id, args, s))$

THEOREM: ram-enable-circuit-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ram\text{-enable-circuit}, args, s))$

THEOREM: t-buf-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('t\text{-buf}, args, s))$

THEOREM: t-wire-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('t\text{-wire}, args, s))$

THEOREM: pullup-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('pullup, args, s))$

THEOREM: ttl-bidirect-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-bidirect}, args, s))$

THEOREM: ttl-clk-input-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-clk-input}, args, s))$

THEOREM: ttl-input-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-input}, args, s))$

THEOREM: ttl-output-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-output}, args, s))$

THEOREM: ttl-output-parametric-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-output-parametric}, args, s))$

THEOREM: ttl-output-fast-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-output-fast}, args, s))$

THEOREM: ttl-tri-output-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-tri-output}, args, s))$

THEOREM: ttl-tri-output-fast-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('ttl\text{-tri-output-fast}, args, s))$

THEOREM: vdd-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('vdd, args, s))$

THEOREM: vdd-parametric-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('vdd\text{-parametric}, args, s))$

THEOREM: vss-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('vss, args, s))$

THEOREM: good-s-constant-ram
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{constant-ram}(s, \text{make-list}(32, x)))$

THEOREM: good-s-write-mem-1
 $(\text{good-s}(s) \wedge \text{properp}(value) \wedge (\text{length}(value) = 32)) \rightarrow \text{good-s}(\text{write-mem1}(v\text{-addr}, s, value))$

THEOREM: good-s-write-mem
 $(\text{good-s}(s) \wedge \text{properp}(value) \wedge (\text{length}(value) = 32)) \rightarrow \text{good-s}(\text{write-mem}(v\text{-addr}, s, value))$

THEOREM: dp-ram-16x32-preserves-good-s
 $\text{good-s}(s) \rightarrow \text{good-s}(\text{dual-apply-state}('dp\text{-ram-16x32}, args, s))$

EVENT: Let us define the theory *good-s-primitives-theory* to consist of the following events: ao2-preserves-good-s, ao4-preserves-good-s, ao6-preserves-good-s, ao7-preserves-good-s, b-and-preserves-good-s, b-and3-preserves-good-s, b-and4-preserves-good-s, b-equiv-preserves-good-s, b-equiv3-preserves-good-s, b-if-preserves-good-s, b-nand-preserves-good-s, b-nand3-preserves-good-s, b-nand4-preserves-good-s, b-nand5-preserves-good-s, b-nand6-preserves-good-s, b-nand8-preserves-good-s, b-nbuf-preserves-good-s, b-nor-preserves-good-s, b-nor3-preserves-good-s, b-nor4-preserves-good-s, b-nor5-preserves-good-s, b-nor6-preserves-good-s, b-nor8-preserves-good-s, b-not-preserves-good-s, b-not-b4ip-preserves-good-s, b-not-ivap-preserves-good-s, b-or-preserves-good-s, b-or3-preserves-good-s, b-or4-preserves-good-s, b-xor-preserves-good-s, b-xor3-preserves-good-s, del2-preserves-good-s, del4-preserves-good-s, del10-preserves-good-s, procmon-preserves-good-s, fd1-preserves-good-s, fd1s-preserves-good-s, fd1sp-preserves-good-s, fd1slp-preserves-good-s, id-preserves-good-s, ram-enable-circuit-preserves-good-s, t-buf-preserves-good-s, t-wire-preserves-good-s, pullup-preserves-good-s, ttl-bidirect-preserves-good-s, ttl-clk-input-preserves-good-s, ttl-input-preserves-good-s, ttl-output-preserves-good-s, ttl-output-parametric-preserves-good-s, ttl-output-fast-preserves-good-s, ttl-tri-output-preserves-good-s, ttl-tri-output-fast-preserves-good-s, vdd-preserves-good-s, vdd-parametric-preserves-good-s, vss-preserves-good-s, dp-ram-16x32-preserves-good-s.

THEOREM: primp-preserves-good-s
 $(\text{good-s}(s) \wedge \text{primp}(fn) \wedge (fn \neq \text{'mem-32x32}))$
 $\rightarrow \text{good-s}(\text{dual-apply-state}(fn, args, s))$

THEOREM: good-s-preserved
 $\text{ok-netlist}(flag, fn, netlist, \text{'(mem-32x32)})$
 $\rightarrow (((flag = 2) \wedge \text{good-s}(s))$
 $\quad \rightarrow \text{good-s}(\text{dual-eval}(flag, fn, a, s, netlist)))$
 $\wedge (((flag = 3) \wedge \text{good-s-alist}(s))$
 $\quad \rightarrow \text{good-s-alist}(\text{dual-eval}(flag, fn, a, s, netlist))))$

DEFINITION:
 $\text{v-approx-list}(x, y)$
 $= \text{if listp}(x)$
 $\quad \text{then listp}(y)$
 $\quad \quad \wedge \text{v-approx}(\text{car}(x), \text{car}(y))$
 $\quad \quad \wedge \text{v-approx-list}(\text{cdr}(x), \text{cdr}(y))$
 $\quad \text{else } y \simeq \text{nil endif}$

DEFINITION:
 $\text{v-s-approx-list}(x, y)$
 $= \text{if listp}(x)$
 $\quad \text{then listp}(y)$


```

      ∧ v-approx (caar (x), caar (y))
      ∧ s-approx (cadar (x), cadar (y))
      ∧ v-s-approx-list (cdr (x), cdr (y))
    else t endif

```

DEFINITION:

```

nat-lst-lst-induction (n, x, y)
=  if (n ≃ 0) ∨ (x ≃ nil) ∨ (y ≃ nil) then t
    else nat-lst-lst-induction (n - 1, cdr (x), cdr (y)) endif

```

EVENT: Disable open-nth; name this event ‘open-nth-on’.

THEOREM: v-approx-car-nth

```

(v-s-approx-list (final-1, final-2) ∧ (n < length (final-1)))
→ v-approx (car (nth (n, final-1)), car (nth (n, final-2)))

```

THEOREM: s-approx-cadr-nth

```

(v-s-approx-list (final-1, final-2) ∧ (n < length (final-1)))
→ s-approx (cadr (nth (n, final-1)), cadr (nth (n, final-2)))

```

DEFINITION:

simulate-monotone-induction (fn, inputs-1, state-1, netlist, inputs-2, state-2)

```

=  if listp (inputs-1)
    then simulate-monotone-induction (fn,
                                         cdr (inputs-1),
                                         dual-eval (2,
                                                         fn,
                                                         car (inputs-1),
                                                         state-1,
                                                         netlist),
                                         netlist,
                                         cdr (inputs-2),
                                         dual-eval (2,
                                                         fn,
                                                         car (inputs-2),
                                                         state-2,
                                                         netlist))
    else t endif

```

THEOREM: simulate-monotone

```

(v-approx-list (inputs-1, inputs-2)
 ∧ s-approx (state-1, state-2)
 ∧ good-s (state-2)
 ∧ ok-netlistp (0, fn, netlist, ' (mem-32x32))

```

$\wedge$ ok-netlistp(2, fn , $netlist$, 'mem-32x32'))
 $\rightarrow$ v-s-approx-list(simulate(fn , $inputs-1$, $state-1$, $netlist$),
 simulate(fn , $inputs-2$, $state-2$, $netlist$))

THEOREM: v-approx-list-x-x
 v-approx-list(x , x)

DEFINITION: doublet-p(x) = (x = list(car(x), cadr(x)))

DEFINITION:
 doublet-n-simulate-induction(fn , $inputs$, $state$, $netlist$, n)
 = **if** $n \simeq 0$ **then** **t**
 else doublet-n-simulate-induction(fn ,
 cdr($inputs$),
 dual-eval(2,
 fn ,
 car($inputs$),
 $state$,
 $netlist$),
 $netlist$,
 $n - 1$) **endif**

THEOREM: doublet-n-simulate
 ($n < \text{length}(inputs)$)
 $\rightarrow$ doublet-p(nth(n , simulate(fn , $inputs$, $state$, $netlist$)))

THEOREM: doublet-p-equal-approx
 (doublet-p(x)
 $\wedge$ doublet-p(y)
 $\wedge$ v-knownp(car(x))
 $\wedge$ s-knownp(cadr(x))
 $\wedge$ v-approx(car(x), car(y))
 $\wedge$ s-approx(cadr(x), cadr(y)))
 $\rightarrow$ ($x = y$)

EVENT: Enable doublet-p; name this event 'doublet-p-off'.

THEOREM: length-simulate
 length(simulate(fn , $inputs$, $state$, $netlist$)) = length($inputs$)

THEOREM: xs-suffice-for-reset-lemma-verbose
 (($n < \text{length}(inputs)$)
 $\wedge$ s-approx($state-1$, $state-2$)
 $\wedge$ v-knownp(car(nth(n , simulate(fn , $inputs$, $state-1$, $netlist$))))
 $\wedge$ s-knownp(cadr(nth(n , simulate(fn , $inputs$, $state-1$, $netlist$))))

```

^ good-s (state-2)
^ ok-netlistp (0, fn, netlist, ' (mem-32x32))
^ ok-netlistp (2, fn, netlist, ' (mem-32x32)))
→ (nth (n, simulate (fn, inputs, state-1, netlist))
    = nth (n, simulate (fn, inputs, state-2, netlist)))

```

THEOREM: xs-suffice-for-reset-lemma

```

((n < length (inputs))
 ∧ s-approx (state-1, state-2)
 ∧ v-knownp (car (nth (n, simulate (fn, inputs, state-1, netlist))))
 ∧ s-knownp (cadr (nth (n, simulate (fn, inputs, state-1, netlist))))
 ∧ good-s (state-2)
 ∧ ok-netlistp (0, fn, netlist, ' (mem-32x32)))
→ (nth (n, simulate (fn, inputs, state-1, netlist))
    = nth (n, simulate (fn, inputs, state-2, netlist)))

```

DEFINITION: $\text{COMPILE-UNCOMPILED-DEFNS} = \mathbf{t}$

EVENT: Disable compile-uncompiled-defns; name this event ‘compile-uncompiled-defns-on’.

EVENT: For efficiency, compile those definitions not yet compiled.

DEFINITION:

UNKNOWN-REGFILE

```
= list (cons (cons (cons (cons (cons (ram (make-list (32, x))), ram (make-list (32, x))),
                                cons (ram (make-list (32, x)), ram (make-list (32, x)))),
                cons (cons (ram (make-list (32, x)), ram (make-list (32, x))),
                    cons (ram (make-list (32, x)), ram (make-list (32, x))))),
    cons (cons (cons (ram (make-list (32, x)), ram (make-list (32, x))),
        cons (ram (make-list (32, x)), ram (make-list (32, x)))),
    cons (cons (ram (make-list (32, x)), ram (make-list (32, x))),
        cons (ram (make-list (32, x)), ram (make-list (32, x)))))
```

DEFINITION:

UNKNOWN-MACHINE-STATE

```
= list (UNKNOWN-REGFILE.  
      make-list (4, x),  
      make-list (32, x),  
      make-list (32, x),  
      make-list (32, x),
```

```

make-list (32, x),
make-list (32, x),
x,
x,
x,
make-list (4, x),
make-list (40, x))

```

DEFINITION:

UNKNOWN-MEMORY-STATE

```

= list (stub (make-list (32, f)),
0,
0,
0,
x,
x,
make-list (32, x),
make-list (32, x))

```

DEFINITION:

UNKNOWN-STATE = list (UNKNOWN-MACHINE-STATE, UNKNOWN-MEMORY-STATE)

THEOREM: chip-system-invariant-unknown-state

chip-system-invariant (UNKNOWN-STATE)

THEOREM: fm9001-state-structure-unknown-state

fm9001-state-structure (UNKNOWN-STATE)

DEFINITION:

RESET-VECTOR

```

= cons (x,
cons (x, cons (f, cons (f, cons (t, cons (t, cons (t, make-list (4, t))))))))

```

DEFINITION:

RUN-VECTOR

```

= cons (x,
cons (x, cons (f, cons (t, cons (t, cons (t, cons (t, make-list (4, t))))))))

```

DEFINITION:

RESET-SEQUENCE = cons (RESET-VECTOR, make-list (19, RUN-VECTOR))

THEOREM: chip-system-operating-inputs-p-reset-sequence

chip-system-operating-inputs-p (RESET-SEQUENCE, 20)

DEFINITION:

INITIALIZED-REGFILE

```
= list (cons (cons (cons (cons (ram (make-list (32, f)), ram (make-list (32, f))),
                        cons (ram (make-list (32, f)), ram (make-list (32, f)))),
                cons (cons (ram (make-list (32, f)), ram (make-list (32, f))),
                        cons (ram (make-list (32, f)), ram (make-list (32, f))))),
        cons (cons (cons (ram (make-list (32, f)), ram (make-list (32, f))),
                        cons (ram (make-list (32, f)), ram (make-list (32, f))))),
        cons (cons (ram (make-list (32, f)), ram (make-list (32, f))),
                cons (ram (make-list (32, f)), ram (make-list (32, f))))),
        f,
        make-list (32, f),
        make-list (4, t))
```

DEFINITION:

INITIALIZED-MACHINE-STATE

```
= list (INITIALIZED-REGFILE,
        list (t, f, f, f),
        make-list (32, f),
        make-list (32, f),
        make-list (32, f),
        make-list (32, f),
        make-list (32, f),
        t,
        t,
        t,
        make-list (4, t),
        cv_fetch1 (t,
                    list (t, t, t, t),
                    make-list (32, f),
                    list (t, f, f, f),
                    make-list (4, t)))
```

DEFINITION:

INITIALIZED-MEMORY-STATE

```
= list (stub (make-list (32, f)),
        0,
        0,
        0,
        t,
        t,
        make-list (32, f),
        make-list (32, x))
```

DEFINITION:

FINAL-STATE = list (INITIALIZED-MACHINE-STATE, INITIALIZED-MEMORY-STATE)

THEOREM: reset-works

run-fm9001 (UNKNOWN-STATE,
 map-up-inputs (RESET-SEQUENCE),
 length (RESET-SEQUENCE))
 = FINAL-STATE

THEOREM: unknown-state-okp

fm9001-state-structure (UNKNOWN-STATE)
 ∧ chip-system-invariant (UNKNOWN-STATE)
 ∧ chip-system-operating-inputs-p (RESET-SEQUENCE, 20)

THEOREM: final-state-okp

chip-system& (CHIP-SYSTEM\$NETLIST)
 ∧ fm9001-state-structure (FINAL-STATE)
 ∧ macrocycle-invariant (FINAL-STATE, make-list (4, t))

THEOREM: instance-theorem

(chip-system& (CHIP-SYSTEM\$NETLIST)
 ∧ fm9001-state-structure (FINAL-STATE)
 ∧ macrocycle-invariant (FINAL-STATE, pc)
 ∧ chip-system-operating-inputs-p (*inputs*,
 total-microcycles (FINAL-STATE,
 map-up-inputs (*inputs*),
 n))
 ∧ operating-inputs-p (map-up-inputs (*inputs*),
 total-microcycles (FINAL-STATE,
 map-up-inputs (*inputs*),
 n)))
 → (map-up (simulate-dual-eval-2 ('chip-system,
 inputs,
 FINAL-STATE,
 CHIP-SYSTEM\$NETLIST,
 total-microcycles (FINAL-STATE,
 map-up-inputs (*inputs*),
 n)))
 = fm9001-interpreter (map-up (FINAL-STATE), pc, *n*))

THEOREM: fm9001-statep-map-up-final-state

fm9001-statep (map-up (FINAL-STATE))

DEFINITION:

RESET-VECTOR-CHIP

= cons (X,

```

cons (X,
      cons (f,
            cons (X,
                  cons (f,
                        cons (t,
                              cons (t,
                                    cons (t,
                                          append (make-list (4, t),
                                                            make-list (32, x))))))))))

```

DEFINITION:

RUN-VECTOR-CHIP-1

```

= cons (t,
        cons (X,
              cons (f,
                    cons (t,
                          cons (t,
                                cons (t,
                                      cons (t,
                                            cons (t,
                                                  append (make-list (4, t),
                                                                    make-list (32, t))))))))))

```

DEFINITION:

RUN-VECTOR-CHIP-2

```

= cons (f,
        cons (X,
              cons (f,
                    cons (t,
                          cons (t,
                                cons (t,
                                      cons (t,
                                            cons (t,
                                                  append (make-list (4, t),
                                                                    make-list (32, t))))))))))

```

DEFINITION:

RESET-SEQUENCE-CHIP-1

```

= cons (RESET-VECTOR-CHIP, make-list (19, RUN-VECTOR-CHIP-1))

```

DEFINITION:

RESET-SEQUENCE-CHIP-2

```

= cons (RESET-VECTOR-CHIP, make-list (19, RUN-VECTOR-CHIP-2))

```

THEOREM: reset-sequence-chip-1-vs-2

```

simulate('chip, RESET-SEQUENCE-CHIP-1, UNKNOWN-MACHINE-STATE, CHIP$NETLIST)
=  simulate('chip,
           RESET-SEQUENCE-CHIP-2,
           UNKNOWN-MACHINE-STATE,
           CHIP$NETLIST)

```

THEOREM: simulate-reset-chip-final-state

```

cadr (nth (length (RESET-SEQUENCE-CHIP-1) - 1,
               simulate('chip,
                       RESET-SEQUENCE-CHIP-1,
                       UNKNOWN-MACHINE-STATE,
                       CHIP$NETLIST)))
=  car (FINAL-STATE)

```

THEOREM: for-final-1-of-reset-sequence-chip

```

(((length (RESET-SEQUENCE-CHIP-1) - 1) < length (RESET-SEQUENCE-CHIP-1))
 =  t)
^  v-knownp (car (nth (length (RESET-SEQUENCE-CHIP-1) - 1,
                       simulate('chip,
                               RESET-SEQUENCE-CHIP-1,
                               UNKNOWN-MACHINE-STATE,
                               CHIP$NETLIST))))
^  s-knownp (cadr (nth (length (RESET-SEQUENCE-CHIP-1) - 1,
                          simulate('chip,
                                  RESET-SEQUENCE-CHIP-1,
                                  UNKNOWN-MACHINE-STATE,
                                  CHIP$NETLIST))))
^  ok-netlistp (0, 'chip, CHIP$NETLIST, '(mem-32x32))

```

EVENT: Enable for-final-1-of-reset-sequence-chip; name this event 'for-final-1-of-reset-sequence-chip-off'.

THEOREM: xs-suffice-for-reset-chip-lemma-instance

```

(s-approx (UNKNOWN-MACHINE-STATE, state-2) ^ good-s (state-2))
→  (nth (length (RESET-SEQUENCE-CHIP-1) - 1,
            simulate('chip,
                    RESET-SEQUENCE-CHIP-1,
                    UNKNOWN-MACHINE-STATE,
                    CHIP$NETLIST))
=  nth (length (RESET-SEQUENCE-CHIP-1) - 1,
        simulate('chip,
                RESET-SEQUENCE-CHIP-1,
                state-2,
                CHIP$NETLIST)))

```


THEOREM: fm9001-machine-statep-p-map-up-initialized-machine-state
 fm9001-machine-statep (p-map-up (INITIALIZED-MACHINE-STATE))

DEFINITION:

v-fourp (x)
 = **if** $x \simeq \mathbf{nil}$ **then** $x = \mathbf{nil}$
 else fourp (car (x)) $\wedge$ v-fourp (cdr (x)) **endif**

DEFINITION:

all-xs (vec)
 = **if** $vec \simeq \mathbf{nil}$ **then** $vec = \mathbf{nil}$
 else (car (vec) = x) $\wedge$ all-xs (cdr (vec)) **endif**

EVENT: Disable b-knownp; name this event 'b-knownp-on'.

EVENT: Disable b-approx; name this event 'b-approx-on'.

THEOREM: all-xs-approximates
 (all-xs ($vec1$) $\wedge$ (length ($vec1$) = length ($vec2$)) $\wedge$ properp ($vec2$))
 $\rightarrow$ v-approx ($vec1$, $vec2$)

THEOREM: all-xs-make-list
 all-xs (make-list (n , x))

DEFINITION:

memory-v-fourp (n , $width$, mem)
 = **if** stubp (mem)
 then v-fourp (stub-guts (mem)) $\wedge$ (length (stub-guts (mem)) = $width$)
 elseif $n \simeq 0$
 then if ramp (mem)
 then v-fourp (ram-guts (mem))
 $\wedge$ (length (ram-guts (mem)) = $width$)
 elseif romp (mem)
 then v-fourp (rom-guts (mem))
 $\wedge$ (length (rom-guts (mem)) = $width$)
 else f endif
 else listp (mem)
 $\wedge$ memory-v-fourp ($n - 1$, $width$, car (mem))
 $\wedge$ memory-v-fourp ($n - 1$, $width$, cdr (mem)) **endif**

DEFINITION:

new-machine-state-invariant ($machine-state$)
 = ((length ($machine-state$) = 12)
 $\wedge$ properp ($machine-state$))

$\wedge$ (length (car (machine-state)) = 4)
 $\wedge$ properp (car (machine-state))
 $\wedge$ all-ramp-mem (4, caar (machine-state))
 $\wedge$ fourp (cadar (machine-state))
 $\wedge$ memory-v-fourp (4, 32, caar (machine-state))
 $\wedge$ v-fourp (caddar (machine-state))
 $\wedge$ (length (caddar (machine-state)) = 32)
 $\wedge$ v-fourp (caddddar (machine-state))
 $\wedge$ (length (caddddar (machine-state)) = 4)
 $\wedge$ v-fourp (cadr (machine-state))
 $\wedge$ (length (cadr (machine-state)) = 4)
 $\wedge$ v-fourp (caddr (machine-state))
 $\wedge$ (length (caddr (machine-state)) = 32)
 $\wedge$ v-fourp (cadddr (machine-state))
 $\wedge$ (length (cadddr (machine-state)) = 32)
 $\wedge$ v-fourp (caddddr (machine-state))
 $\wedge$ (length (caddddr (machine-state)) = 32)
 $\wedge$ v-fourp (cadddddr (machine-state))
 $\wedge$ (length (cadddddr (machine-state)) = 32)
 $\wedge$ v-fourp (caddddddr (machine-state))
 $\wedge$ (length (caddddddr (machine-state)) = 32)
 $\wedge$ fourp (cadddddddr (machine-state))
 $\wedge$ fourp (cadddddddrr (machine-state))
 $\wedge$ fourp (cadddddddrrr (machine-state))
 $\wedge$ v-fourp (cadddddddrrr (machine-state))
 $\wedge$ (length (cadddddddrrr (machine-state)) = 4)
 $\wedge$ v-fourp (cadddddddrrrr (machine-state))
 $\wedge$ (length (cadddddddrrrr (machine-state)) = 40)

THEOREM: s-approx-make-list

$((\text{length} (vec1) = \text{length} (vec2)) \wedge \text{v-fourp} (vec2) \wedge \text{all-xs} (vec1))$
 $\rightarrow \text{s-approx} (vec1, vec2)$

THEOREM: v-fourp-implies-properp

$\text{v-fourp} (x) \rightarrow \text{properp} (x)$

THEOREM: listp-implies-not-fourp

$\text{listp} (x) \rightarrow (\neg \text{fourp} (x))$

THEOREM: ramp-implies-not-fourp

$\text{ramp} (x) \rightarrow (\neg \text{fourp} (x))$

THEOREM: romp-implies-not-fourp

$\text{romp} (x) \rightarrow (\neg \text{fourp} (x))$

THEOREM: stubp-implies-not-fourp

$\text{stubp}(x) \rightarrow (\neg \text{fourp}(x))$

THEOREM: b-approx-x

$\text{b-approx}(x, y)$

THEOREM: machine-state-invariant-implies-s-approx-lemma

$(\text{new-machine-state-invariant}(\text{state}))$

$\wedge (\text{state} = \text{list}(\text{list}(s1-1, s1-2, s1-3, s1-4),$
 $s2,$
 $s3,$
 $s4,$
 $s5,$
 $s6,$
 $s7,$
 $s8,$
 $s9,$
 $s10,$
 $s11,$
 $s12)))$

$\rightarrow \text{s-approx}(\text{UNKNOWN-MACHINE-STATE}, \text{state})$

THEOREM: machine-state-invariant-implies-s-approx-lemma-2

$((\text{length}(\text{state}) = 12)$

$\wedge \text{properp}(\text{state})$

$\wedge (\text{length}(\text{car}(\text{state})) = 4)$

$\wedge \text{properp}(\text{car}(\text{state})))$

$\rightarrow (\text{state} = \text{list}(\text{list}(\text{caar}(\text{state}),$
 $\text{cadar}(\text{state}),$
 $\text{caddar}(\text{state}),$
 $\text{caddar}(\text{state})),$
 $\text{cadr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state}),$
 $\text{caddr}(\text{state})))$

THEOREM: machine-state-invariant-implies-s-approx-lemma-3

$\text{new-machine-state-invariant}(\text{state})$

$\rightarrow$ ((length(*state*) = 12)
 $\wedge$ properp(*state*)
 $\wedge$ (length(car(*state*)) = 4)
 $\wedge$ properp(car(*state*)))

THEOREM: machine-state-invariant-implies-s-approx
 new-machine-state-invariant(*state*)
 $\rightarrow$ s-approx(UNKNOWN-MACHINE-STATE, *state*)

THEOREM: good-s-opener
 (listp(*s*) $\rightarrow$ (good-s(*s*) = (good-s(car(*s*)) $\wedge$ good-s(cdr(*s*))))))
 $\wedge$ (ramp(*s*)
 $\rightarrow$ (good-s(*s*)
 $=$ ((length(ram-guts(*s*)) = MEM-WIDTH)
 $\wedge$ properp(ram-guts(*s*))))))
 $\wedge$ (romp(*s*)
 $\rightarrow$ (good-s(*s*)
 $=$ ((length(rom-guts(*s*)) = MEM-WIDTH)
 $\wedge$ properp(rom-guts(*s*))))))
 $\wedge$ (stubp(*s*)
 $\rightarrow$ (good-s(*s*)
 $=$ ((length(stub-guts(*s*)) = MEM-WIDTH)
 $\wedge$ properp(stub-guts(*s*))))))
 $\wedge$ (fourp(*s*) $\rightarrow$ good-s(*s*))

THEOREM: memory-v-fourp-implies-good-s
 memory-v-fourp(*n*, 32, *v*) $\rightarrow$ good-s(*v*)

THEOREM: v-fourp-is-good-s
 v-fourp(*v*) $\rightarrow$ good-s(*v*)

THEOREM: new-machine-state-invariant-implies-good-s
 new-machine-state-invariant(*state*) $\rightarrow$ good-s(*state*)

THEOREM: xs-suffice-for-reset-chip-final-state-for-any-unknown-state
 new-machine-state-invariant(*state*)
 $\rightarrow$ (cadr(nth(length(RESET-SEQUENCE-CHIP-1) - 1,
 simulate('chip,
 RESET-SEQUENCE-CHIP-1,
 UNKNOWN-MACHINE-STATE,
 CHIP\$NETLIST)))
 $=$ cadr(nth(length(RESET-SEQUENCE-CHIP-1) - 1,
 simulate('chip,
 RESET-SEQUENCE-CHIP-1,
state,
 CHIP\$NETLIST))))))

THEOREM: xs-suffice-for-reset-chip-final-state-for-any-unknown-state-better
 new-machine-state-invariant (*state*)
 $\rightarrow$ (cadr (nth (length (RESET-SEQUENCE-CHIP-1) - 1,
 simulate ('chip, RESET-SEQUENCE-CHIP-1, *state*, CHIP\$NETLIST))))
 = INITIALIZED-MACHINE-STATE)

THEOREM: new-machine-state-invariant-is-non-trivial
 new-machine-state-invariant (UNKNOWN-MACHINE-STATE)

THEOREM: memory-v-fourp-implies-memory-properp
 memory-v-fourp (*n*, *width*, *mem*) $\rightarrow$ memory-properp (*n*, *width*, *mem*)

THEOREM: new-machine-state-invariant-implies-machine-state-invariant
 new-machine-state-invariant (*machine-state*)
 $\rightarrow$ machine-state-invariant (*machine-state*)

THEOREM: fm9001-statep-implies-memory-ok-p-instance
 memory-okp (32,
 32,
 cadr (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*))))
 $\rightarrow$ fm9001-statep (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*)))

THEOREM: fm9001=chip-system-true-after-reset-lemma
 (chip-system& (*netlist*)
 $\wedge$ memory-okp (32,
 32,
 cadr (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*))))
 $\wedge$ chip-system-operating-inputs-p (*inputs*,
 total-microcycles (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
 machine-memory))),
 map-up-inputs (*inputs*),
 n))
 $\wedge$ operating-inputs-p (map-up-inputs (*inputs*),
 total-microcycles (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
 machine-memory))),
 map-up-inputs (*inputs*),
 n)))
 $\rightarrow$ (fm9001 (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*)), *n*)
 = map-up (simulate-dual-eval-2 ('chip-system,
 inputs,
 map-down (map-up (list (INITIALIZED-MACHINE-STATE,
 machine-memory))),
 netlist,
 total-microcycles (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
 machine-memory))),

map-up-inputs (*inputs*,
n))))

THEOREM: map-down-inverts-map-up
memory-okp (32,

32,
cadr (map-up (list (INITIALIZED-MACHINE-STATE,
list (*memory*,
0,
0,
0,
t,
t,
make-list (32, **f**),
make-list (32, *x*))))))
→ (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
list (*memory*,
0,
0,
0,
t,
t,
make-list (32, **f**),
make-list (32, *x*))))))
= list (INITIALIZED-MACHINE-STATE,
list (*memory*, 0, 0, 0, 0, **t**, **t**, make-list (32, **f**), make-list (32, *x*))))

THEOREM: cadr-map-up
cadr (map-up (list (*s*, *mem*))) = car (*mem*)

THEOREM: fm9001=chip-system-true-after-reset

(chip-system& (*netlist*)
∧ memory-okp (32, 32, *memory*)
∧ chip-system-operating-inputs-p (*inputs*,
total-microcycles (list (INITIALIZED-MACHINE-STATE,
list (*memory*,
0,
0,
0,
t,
t,
make-list (32,
f),
make-list (32,
x))),

```

map-up-inputs (inputs),
n))
^  operating-inputs-p (map-up-inputs (inputs),
total-microcycles (list (INITIALIZED-MACHINE-STATE,
list (memory,
0,
0,
0,
t,
t,
make-list (32, f),
make-list (32, x))),
map-up-inputs (inputs),
n)))
→  (fm9001 (map-up (list (INITIALIZED-MACHINE-STATE,
list (memory,
0,
0,
0,
t,
t,
make-list (32, f),
make-list (32, x)))),
n)
=  map-up (simulate-dual-eval-2 ('chip-system,
inputs,
list (INITIALIZED-MACHINE-STATE,
list (memory,
0,
0,
0,
t,
t,
make-list (32, f),
make-list (32, x))),
netlist,
total-microcycles (list (INITIALIZED-MACHINE-STATE,
list (memory,
0,
0,
0,
t,
t,
make-list (32,

```

```

                                f),
                                make-list (32,
                                              x))),
                                map-up-inputs (inputs),
                                n))))

```

EVENT: Enable map-down-inverts-map-up; name this event ‘map-down-inverts-map-up-off’.

EVENT: Enable b-knownp; name this event ‘b-knownp-off1’.

EVENT: Enable b-approx; name this event ‘b-approx-off1’.

THEOREM: simulate-contains-simulate-dual-eval-2

```

((n ≠ 0)
 ∧  if length (inputs) < n then f
    else t endif)
→  (cadr (nth (n - 1, simulate (fn, inputs, state, netlist)))
    =  simulate-dual-eval-2 (fn, inputs, state, netlist, n))

```

THEOREM: length-reset-sequence-chip-1

```

length (RESET-SEQUENCE-CHIP-1) = 20

```

THEOREM: reset-chip

```

simulate-dual-eval-2 ('chip,
                      RESET-SEQUENCE-CHIP-1,
                      UNKNOWN-MACHINE-STATE,
                      CHIP$NETLIST,
                      length (RESET-SEQUENCE-CHIP-1))
=  INITIALIZED-MACHINE-STATE

```

THEOREM: reset-chip-from-any-state

```

(s-approx (UNKNOWN-MACHINE-STATE, any-state) ∧ good-s (any-state))
→  (simulate-dual-eval-2 ('chip,
                          RESET-SEQUENCE-CHIP-1,
                          any-state,
                          CHIP$NETLIST,
                          length (RESET-SEQUENCE-CHIP-1))
    =  INITIALIZED-MACHINE-STATE)

```

THEOREM: chip-system=fm9001-interpret\$after-reset

```

(chip-system& (netlist)
 ∧  memory-okp (32, 32, memory)

```



```

^  chip-system-operating-inputs-p (inputs,
                                   total-microcycles (list (INITIALIZED-MACHINE-STATE,
                                                             list (memory,
                                                                0,
                                                                clock,
                                                                0,
                                                                X,
                                                                t,
                                                                make-list (32,
                                                                    f),
                                                                make-list (32,
                                                                    X))),
                                                             map-up-inputs (inputs,
                                                                    instructions)))

^  operating-inputs-p (map-up-inputs (inputs),
                           total-microcycles (list (INITIALIZED-MACHINE-STATE,
                                                       list (memory,
                                                            0,
                                                            clock,
                                                            0,
                                                            X,
                                                            t,
                                                            make-list (32, f),
                                                            make-list (32, X))),
                                                       map-up-inputs (inputs,
                                                            instructions)))

→  (map-up (simulate-dual-eval-2 ('chip-system,
                                   inputs,
                                   list (INITIALIZED-MACHINE-STATE,
                                         list (memory,
                                                0,
                                                clock,
                                                0,
                                                X,
                                                t,
                                                make-list (32, f),
                                                make-list (32, X))),
                                   netlist,
                                   total-microcycles (list (INITIALIZED-MACHINE-STATE,
                                                             list (memory,
                                                                0,
                                                                clock,
                                                                0,
                                                                X,

```

```

                                t,
                                make-list (32,
                                              f),
                                make-list (32,
                                              x))),
                                map-up-inputs (inputs),
                                instructions)))
=  fm9001-interpreter (map-up (list (INITIALIZED-MACHINE-STATE,
                                list (memory,
                                      0,
                                      clock,
                                      0,
                                      x,
                                      t,
                                      make-list (32, f),
                                      make-list (32, x))))),
                                make-list (4, t),
                                instructions))

```

EVENT: Disable compile-uncompiled-defns; name this event ‘compile-uncompiled-defns-on1’.

EVENT: For efficiency, compile those definitions not yet compiled.

THEOREM: chip-well-formed
 (top-level-predicate (CHIP-MODULE\$NETLIST) = **t**)
 ∧ (top-level-predicate (CHIP\$NETLIST) = **t**)

THEOREM: chip-well-formed-after-indexed-names-removed
 (top-level-predicate (lisp-netlist (CHIP-MODULE\$NETLIST)) = **t**)
 ∧ (top-level-predicate (lisp-netlist (CHIP\$NETLIST)) = **t**)

THEOREM: chip-well-formed-simple
 (top-level-predicate-simple (CHIP-MODULE\$NETLIST) = **t**)
 ∧ (top-level-predicate-simple (lisp-netlist (CHIP-MODULE\$NETLIST)) = **t**)

EVENT: Enable chip-well-formed-simple; name this event ‘chip-well-formed-simple-off’.

EVENT: Enable chip-well-formed-after-indexed-names-removed; name this event ‘chip-well-formed-after-indexed-names-removed-off’.

EVENT: Enable chip-well-formed; name this event ‘chip-well-formed-off’.

EVENT: Enable chip-system=fm9001-interpretter\$after-reset; name this event ‘chip-system=fm9001-interpretter\$after-reset-off’.

EVENT: Enable reset-chip-from-any-state; name this event ‘reset-chip-from-any-state-off’.

EVENT: Enable reset-chip; name this event ‘reset-chip-off’.

EVENT: Enable length-reset-sequence-chip-1; name this event ‘length-reset-sequence-chip-1-off’.

EVENT: Enable simulate-contains-simulate-dual-eval-2; name this event ‘simulate-contains-simulate-dual-eval-2-off’.

EVENT: Enable fm9001=chip-system-true-after-reset; name this event ‘fm9001=chip-system-true-after-reset-off’.

EVENT: Enable cadr-map-up; name this event ‘cadr-map-up-off’.

EVENT: Enable fm9001=chip-system-true-after-reset-lemma; name this event ‘fm9001=chip-system-true-after-reset-lemma-off’.

EVENT: Enable fm9001-statep-implies-memory-ok-p-instance; name this event ‘fm9001-statep-implies-memory-ok-p-instance-off’.

EVENT: Enable new-machine-state-invariant-implies-machine-state-invariant; name this event ‘new-machine-state-invariant-implies-machine-state-invariant-off’.

EVENT: Enable memory-v-fourp-implies-memory-properp; name this event ‘memory-v-fourp-implies-memory-properp-off’.

EVENT: Enable new-machine-state-invariant-is-non-trivial; name this event ‘new-machine-state-invariant-is-non-trivial-off’.

EVENT: Enable xs-suffice-for-reset-chip-final-state-for-any-unknown-state-better; name this event ‘xs-suffice-for-reset-chip-final-state-for-any-unknown-state-better-off’.

EVENT: Enable xs-suffice-for-reset-chip-final-state-for-any-unknown-state; name this event 'xs-suffice-for-reset-chip-final-state-for-any-unknown-state-off'.

EVENT: Enable new-machine-state-invariant-implies-good-s; name this event 'new-machine-state-invariant-implies-good-s-off'.

EVENT: Enable v-fourp-is-good-s; name this event 'v-fourp-is-good-s-off'.

EVENT: Enable memory-v-fourp-implies-good-s; name this event 'memory-v-fourp-implies-good-s-off'.

EVENT: Enable good-s-opener; name this event 'good-s-opener-off'.

EVENT: Enable machine-state-invariant-implies-s-approx; name this event 'machine-state-invariant-implies-s-approx-off'.

EVENT: Enable machine-state-invariant-implies-s-approx-lemma-3; name this event 'machine-state-invariant-implies-s-approx-lemma-3-off'.

EVENT: Enable machine-state-invariant-implies-s-approx-lemma-2; name this event 'machine-state-invariant-implies-s-approx-lemma-2-off'.

EVENT: Enable machine-state-invariant-implies-s-approx-lemma; name this event 'machine-state-invariant-implies-s-approx-lemma-off'.

EVENT: Enable b-approx-x; name this event 'b-approx-x-off'.

EVENT: Enable stubp-implies-not-fourp; name this event 'stubp-implies-not-fourp-off'.

EVENT: Enable romp-implies-not-fourp; name this event 'romp-implies-not-fourp-off'.

EVENT: Enable ramp-implies-not-fourp; name this event 'ramp-implies-not-fourp-off'.

EVENT: Enable listp-implies-not-fourp; name this event 'listp-implies-not-fourp-off'.

off’.

EVENT: Enable v-fourp-implies-properp; name this event ‘v-fourp-implies-properp-off’.

EVENT: Enable s-approx-make-list; name this event ‘s-approx-make-list-off’.

EVENT: Enable new-machine-state-invariant; name this event ‘new-machine-state-invariant-off’.

EVENT: Enable *1*new-machine-state-invariant; name this event ‘g*1*new-machine-state-invariant-off’.

EVENT: Enable memory-v-fourp; name this event ‘memory-v-fourp-off’.

EVENT: Enable *1*memory-v-fourp; name this event ‘g*1*memory-v-fourp-off’.

EVENT: Enable all-xs-make-list; name this event ‘all-xs-make-list-off’.

EVENT: Enable all-xs-approximates; name this event ‘all-xs-approximates-off’.

EVENT: Enable all-xs; name this event ‘all-xs-off’.

EVENT: Enable *1*all-xs; name this event ‘g*1*all-xs-off’.

EVENT: Enable v-fourp; name this event ‘v-fourp-off’.

EVENT: Enable *1*v-fourp; name this event ‘g*1*v-fourp-off’.

EVENT: Enable fm9001-machine-statep-p-map-up-initialized-machine-state; name this event ‘fm9001-machine-statep-p-map-up-initialized-machine-state-off’.

EVENT: Enable xs-suffice-for-reset-chip-lemma-instance; name this event ‘xs-suffice-for-reset-chip-lemma-instance-off’.

EVENT: Enable simulate-reset-chip-final-state; name this event ‘simulate-reset-

chip-final-state-off’.

EVENT: Enable reset-sequence-chip-1-vs-2; name this event ‘reset-sequence-chip-1-vs-2-off’.

EVENT: Enable reset-sequence-chip-2; name this event ‘reset-sequence-chip-2-off’.

EVENT: Enable *1*reset-sequence-chip-2; name this event ‘g*1*reset-sequence-chip-2-off’.

EVENT: Enable reset-sequence-chip-1; name this event ‘reset-sequence-chip-1-off’.

EVENT: Enable *1*reset-sequence-chip-1; name this event ‘g*1*reset-sequence-chip-1-off’.

EVENT: Enable run-vector-chip-2; name this event ‘run-vector-chip-2-off’.

EVENT: Enable *1*run-vector-chip-2; name this event ‘g*1*run-vector-chip-2-off’.

EVENT: Enable run-vector-chip-1; name this event ‘run-vector-chip-1-off’.

EVENT: Enable *1*run-vector-chip-1; name this event ‘g*1*run-vector-chip-1-off’.

EVENT: Enable reset-vector-chip; name this event ‘reset-vector-chip-off’.

EVENT: Enable *1*reset-vector-chip; name this event ‘g*1*reset-vector-chip-off’.

EVENT: Enable fm9001-statep-map-up-final-state; name this event ‘fm9001-statep-map-up-final-state-off’.

EVENT: Enable instance-theorem; name this event ‘instance-theorem-off’.

EVENT: Enable final-state-okp; name this event 'final-state-okp-off'.

EVENT: Enable unknown-state-okp; name this event 'unknown-state-okp-off'.

EVENT: Enable reset-works; name this event 'reset-works-off'.

EVENT: Enable final-state; name this event 'final-state-off'.

EVENT: Enable *1*final-state; name this event 'g*1*final-state-off'.

EVENT: Enable initialized-memory-state; name this event 'initialized-memory-state-off'.

EVENT: Enable *1*initialized-memory-state; name this event 'g*1*initialized-memory-state-off'.

EVENT: Enable initialized-machine-state; name this event 'initialized-machine-state-off'.

EVENT: Enable *1*initialized-machine-state; name this event 'g*1*initialized-machine-state-off'.

EVENT: Enable initialized-regfile; name this event 'initialized-regfile-off'.

EVENT: Enable *1*initialized-regfile; name this event 'g*1*initialized-regfile-off'.

EVENT: Enable chip-system-operating-inputs-p-reset-sequence; name this event 'chip-system-operating-inputs-p-reset-sequence-off'.

EVENT: Enable reset-sequence; name this event 'reset-sequence-off'.

EVENT: Enable *1*reset-sequence; name this event 'g*1*reset-sequence-off'.

EVENT: Enable run-vector; name this event 'run-vector-off'.

EVENT: Enable *1*run-vector; name this event 'g*1*run-vector-off'.

EVENT: Enable reset-vector; name this event 'reset-vector-off'.

EVENT: Enable *1*reset-vector; name this event 'g*1*reset-vector-off'.

EVENT: Enable fm9001-state-structure-unknown-state; name this event 'fm9001-state-structure-unknown-state-off'.

EVENT: Enable chip-system-invariant-unknown-state; name this event 'chip-system-invariant-unknown-state-off'.

EVENT: Enable unknown-state; name this event 'unknown-state-off'.

EVENT: Enable *1*unknown-state; name this event 'g*1*unknown-state-off'.

EVENT: Enable unknown-memory-state; name this event 'unknown-memory-state-off'.

EVENT: Enable *1*unknown-memory-state; name this event 'g*1*unknown-memory-state-off'.

EVENT: Enable unknown-machine-state; name this event 'unknown-machine-state-off'.

EVENT: Enable *1*unknown-machine-state; name this event 'g*1*unknown-machine-state-off'.

EVENT: Enable unknown-regfile; name this event 'unknown-regfile-off'.

EVENT: Enable *1*unknown-regfile; name this event 'g*1*unknown-regfile-off'.

EVENT: Enable compile-uncompiled-defns; name this event 'compile-uncompiled-defns-off'.

EVENT: Enable *1*compile-uncompiled-defns; name this event 'g*1*compile-uncompiled-defns-off'.

EVENT: Enable xs-suffice-for-reset-lemma; name this event ‘xs-suffice-for-reset-lemma-off’.

EVENT: Enable xs-suffice-for-reset-lemma-verbose; name this event ‘xs-suffice-for-reset-lemma-verbose-off’.

EVENT: Enable length-simulate; name this event ‘length-simulate-off’.

EVENT: Enable doublet-p-equal-approx; name this event ‘doublet-p-equal-approx-off’.

EVENT: Enable doublet-n-simulate; name this event ‘doublet-n-simulate-off’.

EVENT: Enable doublet-n-simulate-induction; name this event ‘doublet-n-simulate-induction-off’.

EVENT: Enable *1*doublet-n-simulate-induction; name this event ‘g*1*doublet-n-simulate-induction-off’.

EVENT: Enable *1*doublet-p; name this event ‘g*1*doublet-p-off’.

EVENT: Enable v-approx-list-x-x; name this event ‘v-approx-list-x-x-off’.

EVENT: Enable simulate-monotone; name this event ‘simulate-monotone-off’.

EVENT: Enable simulate-monotone-induction; name this event ‘simulate-monotone-induction-off’.

EVENT: Enable *1*simulate-monotone-induction; name this event ‘g*1*simulate-monotone-induction-off’.

EVENT: Enable s-approx-cadr-nth; name this event ‘s-approx-cadr-nth-off’.

EVENT: Enable v-approx-car-nth; name this event ‘v-approx-car-nth-off’.

EVENT: Enable nat-lst-lst-induction; name this event ‘nat-lst-lst-induction-off’.

EVENT: Enable *1*nat-lst-lst-induction; name this event ‘g*1*nat-lst-lst-induction-off’.

EVENT: Enable v-s-approx-list; name this event ‘v-s-approx-list-off’.

EVENT: Enable *1*v-s-approx-list; name this event ‘g*1*v-s-approx-list-off’.

EVENT: Enable v-approx-list; name this event ‘v-approx-list-off’.

EVENT: Enable *1*v-approx-list; name this event ‘g*1*v-approx-list-off’.

EVENT: Enable good-s-preserved; name this event ‘good-s-preserved-off’.

EVENT: Enable primp-preserves-good-s; name this event ‘primp-preserves-good-s-off’.

EVENT: Enable dp-ram-16x32-preserves-good-s; name this event ‘dp-ram-16x32-preserves-good-s-off’.

EVENT: Enable good-s-write-mem; name this event ‘good-s-write-mem-off’.

EVENT: Enable good-s-write-mem-1; name this event ‘good-s-write-mem-1-off’.

EVENT: Enable good-s-constant-ram; name this event ‘good-s-constant-ram-off’.

EVENT: Enable vss-preserves-good-s; name this event ‘vss-preserves-good-s-off’.

EVENT: Enable vdd-parametric-preserves-good-s; name this event ‘vdd-parametric-preserves-good-s-off’.

EVENT: Enable vdd-preserves-good-s; name this event ‘vdd-preserves-good-s-off’.

EVENT: Enable ttl-tri-output-fast-preserves-good-s; name this event ‘ttl-tri-output-fast-preserves-good-s-off’.

EVENT: Enable ttl-tri-output-preserves-good-s; name this event ‘ttl-tri-output-preserves-good-s-off’.

EVENT: Enable ttl-output-fast-preserves-good-s; name this event ‘ttl-output-fast-preserves-good-s-off’.

EVENT: Enable ttl-output-parametric-preserves-good-s; name this event ‘ttl-output-parametric-preserves-good-s-off’.

EVENT: Enable ttl-output-preserves-good-s; name this event ‘ttl-output-preserves-good-s-off’.

EVENT: Enable ttl-input-preserves-good-s; name this event ‘ttl-input-preserves-good-s-off’.

EVENT: Enable ttl-clk-input-preserves-good-s; name this event ‘ttl-clk-input-preserves-good-s-off’.

EVENT: Enable ttl-bidirect-preserves-good-s; name this event ‘ttl-bidirect-preserves-good-s-off’.

EVENT: Enable pullup-preserves-good-s; name this event ‘pullup-preserves-good-s-off’.

EVENT: Enable t-wire-preserves-good-s; name this event ‘t-wire-preserves-good-s-off’.

EVENT: Enable t-buf-preserves-good-s; name this event ‘t-buf-preserves-good-s-off’.

EVENT: Enable ram-enable-circuit-preserves-good-s; name this event ‘ram-enable-circuit-preserves-good-s-off’.

EVENT: Enable id-preserves-good-s; name this event ‘id-preserves-good-s-off’.

EVENT: Enable fd1slp-preserves-good-s; name this event ‘fd1slp-preserves-good-s-off’.

EVENT: Enable fd1sp-preserves-good-s; name this event 'fd1sp-preserves-good-s-off'.

EVENT: Enable fd1s-preserves-good-s; name this event 'fd1s-preserves-good-s-off'.

EVENT: Enable fd1-preserves-good-s; name this event 'fd1-preserves-good-s-off'.

EVENT: Enable procmon-preserves-good-s; name this event 'procmon-preserves-good-s-off'.

EVENT: Enable del10-preserves-good-s; name this event 'del10-preserves-good-s-off'.

EVENT: Enable del4-preserves-good-s; name this event 'del4-preserves-good-s-off'.

EVENT: Enable del2-preserves-good-s; name this event 'del2-preserves-good-s-off'.

EVENT: Enable b-xor3-preserves-good-s; name this event 'b-xor3-preserves-good-s-off'.

EVENT: Enable b-xor-preserves-good-s; name this event 'b-xor-preserves-good-s-off'.

EVENT: Enable b-or4-preserves-good-s; name this event 'b-or4-preserves-good-s-off'.

EVENT: Enable b-or3-preserves-good-s; name this event 'b-or3-preserves-good-s-off'.

EVENT: Enable b-or-preserves-good-s; name this event 'b-or-preserves-good-s-off'.

EVENT: Enable b-not-ivap-preserves-good-s; name this event 'b-not-ivap-preserves-good-s-off'.

EVENT: Enable b-not-b4ip-preserves-good-s; name this event ‘b-not-b4ip-preserves-good-s-off’.

EVENT: Enable b-not-preserves-good-s; name this event ‘b-not-preserves-good-s-off’.

EVENT: Enable b-nor8-preserves-good-s; name this event ‘b-nor8-preserves-good-s-off’.

EVENT: Enable b-nor6-preserves-good-s; name this event ‘b-nor6-preserves-good-s-off’.

EVENT: Enable b-nor5-preserves-good-s; name this event ‘b-nor5-preserves-good-s-off’.

EVENT: Enable b-nor4-preserves-good-s; name this event ‘b-nor4-preserves-good-s-off’.

EVENT: Enable b-nor3-preserves-good-s; name this event ‘b-nor3-preserves-good-s-off’.

EVENT: Enable b-nor-preserves-good-s; name this event ‘b-nor-preserves-good-s-off’.

EVENT: Enable b-nbuf-preserves-good-s; name this event ‘b-nbuf-preserves-good-s-off’.

EVENT: Enable b-nand8-preserves-good-s; name this event ‘b-nand8-preserves-good-s-off’.

EVENT: Enable b-nand6-preserves-good-s; name this event ‘b-nand6-preserves-good-s-off’.

EVENT: Enable b-nand5-preserves-good-s; name this event ‘b-nand5-preserves-good-s-off’.

EVENT: Enable b-nand4-preserves-good-s; name this event ‘b-nand4-preserves-good-s-off’.

EVENT: Enable b-nand3-preserves-good-s; name this event ‘b-nand3-preserves-good-s-off’.

EVENT: Enable b-nand-preserves-good-s; name this event ‘b-nand-preserves-good-s-off’.

EVENT: Enable b-if-preserves-good-s; name this event ‘b-if-preserves-good-s-off’.

EVENT: Enable b-equiv3-preserves-good-s; name this event ‘b-equiv3-preserves-good-s-off’.

EVENT: Enable b-equiv-preserves-good-s; name this event ‘b-equiv-preserves-good-s-off’.

EVENT: Enable b-and4-preserves-good-s; name this event ‘b-and4-preserves-good-s-off’.

EVENT: Enable b-and3-preserves-good-s; name this event ‘b-and3-preserves-good-s-off’.

EVENT: Enable b-and-preserves-good-s; name this event ‘b-and-preserves-good-s-off’.

EVENT: Enable ao7-preserves-good-s; name this event ‘ao7-preserves-good-s-off’.

EVENT: Enable ao6-preserves-good-s; name this event ‘ao6-preserves-good-s-off’.

EVENT: Enable ao4-preserves-good-s; name this event ‘ao4-preserves-good-s-off’.

EVENT: Enable ao2-preserves-good-s; name this event ‘ao2-preserves-good-s-off’.

EVENT: Enable good-s-0; name this event ‘good-s-0-off’.

EVENT: Enable f-if-preserves-good-s; name this event ‘f-if-preserves-good-s-off’.

EVENT: Enable f-buf-preserves-good-s; name this event ‘f-buf-preserves-good-s-off’.

EVENT: Enable good-s-collect-value; name this event ‘good-s-collect-value-off’.

EVENT: Enable dual-eval-monotone; name this event ‘dual-eval-monotone-off’.

EVENT: Enable good-s-value; name this event ‘good-s-value-off’.

EVENT: Enable good-s-alist-pairlist; name this event ‘good-s-alist-pairlist-off’.

EVENT: Enable good-s-alist; name this event ‘good-s-alist-off’.

EVENT: Enable *1*good-s-alist; name this event ‘g*1*good-s-alist-off’.

EVENT: Enable dual-eval-monotone-no-ram; name this event ‘dual-eval-monotone-no-ram-off’.

EVENT: Enable ok-netlistp-reduction-rewrite; name this event ‘ok-netlistp-reduction-rewrite-off’.

EVENT: Enable ok-netlistp-reduction; name this event ‘ok-netlistp-reduction-off’.

EVENT: Enable s-approx-alist-implies-s-approx-collect-value; name this event ‘s-approx-alist-implies-s-approx-collect-value-off’.

EVENT: Enable s-approx-implies-s-approx-alist; name this event ‘s-approx-implies-s-approx-alist-off’.

EVENT: Enable alistp-dual-eval-3; name this event ‘alistp-dual-eval-3-off’.

EVENT: Enable s-approx-alist-implies-s-approx-list-collect-value; name this event ‘s-approx-alist-implies-s-approx-list-collect-value-off’.

EVENT: Enable double-cdr-induction; name this event 'double-cdr-induction-off'.

EVENT: Enable *1*double-cdr-induction; name this event 'g*1*double-cdr-induction-off'.

EVENT: Enable v-approx-alist-append; name this event 'v-approx-alist-append-off'.

EVENT: Enable s-approx-alist-implies-s-approx-value; name this event 's-approx-alist-implies-s-approx-value-off'.

EVENT: Enable alistp-opener; name this event 'alistp-opener-off'.

EVENT: Enable v-approx-implies-v-approx-alist; name this event 'v-approx-implies-v-approx-alist-off'.

EVENT: Enable s-approx-list-implies-s-approx-alist; name this event 's-approx-list-implies-s-approx-alist-off'.

EVENT: Enable s-approx-list; name this event 's-approx-list-off'.

EVENT: Enable *1*s-approx-list; name this event 'g*1*s-approx-list-off'.

EVENT: Enable v-approx-alist-implies-v-approx-collect-value; name this event 'v-approx-alist-implies-v-approx-collect-value-off'.

EVENT: Enable alistp-dual-eval-1; name this event 'alistp-dual-eval-1-off'.

EVENT: Enable alistp-append; name this event 'alistp-append-off'.

EVENT: Enable alistp-pairlist; name this event 'alistp-pairlist-off'.

EVENT: Enable v-approx-alist-implies-b-approx-value; name this event 'v-approx-alist-implies-b-approx-value-off'.

EVENT: Enable dual-eval-monotone-induction; name this event 'dual-eval-monotone-

induction-off’.

EVENT: Enable *1*dual-eval-monotone-induction; name this event ‘g*1*dual-eval-monotone-induction-off’.

EVENT: Enable primp-monotone; name this event ‘primp-monotone-off’.

EVENT: Enable ok-netlistp; name this event ‘ok-netlistp-off’.

EVENT: Enable *1*ok-netlistp; name this event ‘g*1*ok-netlistp-off’.

EVENT: Enable dual-port-ram-state-body; name this event ‘dual-port-ram-state-body-off’.

EVENT: Enable *1*dual-port-ram-state-body; name this event ‘g*1*dual-port-ram-state-body-off’.

EVENT: Enable dual-port-ram-state-monotone; name this event ‘dual-port-ram-state-monotone-off’.

EVENT: Enable write-mem1-double-induction; name this event ‘write-mem1-double-induction-off’.

EVENT: Enable *1*write-mem1-double-induction; name this event ‘g*1*write-mem1-double-induction-off’.

EVENT: Enable write-mem1-monotone-induction; name this event ‘write-mem1-monotone-induction-off’.

EVENT: Enable *1*write-mem1-monotone-induction; name this event ‘g*1*write-mem1-monotone-induction-off’.

EVENT: Enable dual-port-ram-value-body; name this event ‘dual-port-ram-value-body-off’.

EVENT: Enable *1*dual-port-ram-value-body; name this event ‘g*1*dual-port-ram-value-body-off’.

EVENT: Enable v-approx-x-x; name this event ‘v-approx-x-x-off1’.

EVENT: Enable read-mem1-monotone-induction; name this event ‘read-mem1-monotone-induction-off’.

EVENT: Enable *1*read-mem1-monotone-induction; name this event ‘g*1*read-mem1-monotone-induction-off’.

EVENT: Enable bvp-rev1; name this event ‘bvp-rev1-off’.

EVENT: Enable vss-monotone; name this event ‘vss-monotone-off’.

EVENT: Enable dual-eval-vss-state; name this event ‘dual-eval-vss-state-off’.

EVENT: Enable dual-eval-vss-value; name this event ‘dual-eval-vss-value-off’.

EVENT: Enable vdd-parametric-monotone; name this event ‘vdd-parametric-monotone-off’.

EVENT: Enable dual-eval-vdd-parametric-state; name this event ‘dual-eval-vdd-parametric-state-off’.

EVENT: Enable dual-eval-vdd-parametric-value; name this event ‘dual-eval-vdd-parametric-value-off’.

EVENT: Enable vdd-monotone; name this event ‘vdd-monotone-off’.

EVENT: Enable dual-eval-vdd-state; name this event ‘dual-eval-vdd-state-off’.

EVENT: Enable dual-eval-vdd-value; name this event ‘dual-eval-vdd-value-off’.

EVENT: Enable ttl-tri-output-fast-monotone; name this event ‘ttl-tri-output-fast-monotone-off’.

EVENT: Enable dual-eval-ttl-tri-output-fast-state; name this event ‘dual-eval-ttl-tri-output-fast-state-off’.

EVENT: Enable dual-eval-ttl-tri-output-fast-value; name this event ‘dual-eval-ttl-tri-output-fast-value-off’.

EVENT: Enable ttl-tri-output-monotone; name this event ‘ttl-tri-output-monotone-off’.

EVENT: Enable dual-eval-ttl-tri-output-state; name this event ‘dual-eval-ttl-tri-output-state-off’.

EVENT: Enable dual-eval-ttl-tri-output-value; name this event ‘dual-eval-ttl-tri-output-value-off’.

EVENT: Enable ttl-output-fast-monotone; name this event ‘ttl-output-fast-monotone-off’.

EVENT: Enable dual-eval-ttl-output-fast-state; name this event ‘dual-eval-ttl-output-fast-state-off’.

EVENT: Enable dual-eval-ttl-output-fast-value; name this event ‘dual-eval-ttl-output-fast-value-off’.

EVENT: Enable ttl-output-parametric-monotone; name this event ‘ttl-output-parametric-monotone-off’.

EVENT: Enable dual-eval-ttl-output-parametric-state; name this event ‘dual-eval-ttl-output-parametric-state-off’.

EVENT: Enable dual-eval-ttl-output-parametric-value; name this event ‘dual-eval-ttl-output-parametric-value-off’.

EVENT: Enable ttl-output-monotone; name this event ‘ttl-output-monotone-off’.

EVENT: Enable dual-eval-ttl-output-state; name this event ‘dual-eval-ttl-output-state-off’.

EVENT: Enable dual-eval-ttl-output-value; name this event ‘dual-eval-ttl-output-value-off’.

EVENT: Enable ttl-input-monotone; name this event ‘ttl-input-monotone-off’.

EVENT: Enable dual-eval-ttl-input-state; name this event ‘dual-eval-ttl-input-state-off’.

EVENT: Enable dual-eval-ttl-input-value; name this event ‘dual-eval-ttl-input-value-off’.

EVENT: Enable ttl-clk-input-monotone; name this event ‘ttl-clk-input-monotone-off’.

EVENT: Enable dual-eval-ttl-clk-input-state; name this event ‘dual-eval-ttl-clk-input-state-off’.

EVENT: Enable dual-eval-ttl-clk-input-value; name this event ‘dual-eval-ttl-clk-input-value-off’.

EVENT: Enable ttl-bidirect-monotone; name this event ‘ttl-bidirect-monotone-off’.

EVENT: Enable dual-eval-ttl-bidirect-state; name this event ‘dual-eval-ttl-bidirect-state-off’.

EVENT: Enable dual-eval-ttl-bidirect-value; name this event ‘dual-eval-ttl-bidirect-value-off’.

EVENT: Enable pullup-monotone; name this event ‘pullup-monotone-off’.

EVENT: Enable dual-eval-pullup-state; name this event ‘dual-eval-pullup-state-off’.

EVENT: Enable dual-eval-pullup-value; name this event ‘dual-eval-pullup-value-off’.

EVENT: Enable t-wire-monotone; name this event ‘t-wire-monotone-off’.

EVENT: Enable dual-eval-t-wire-state; name this event ‘dual-eval-t-wire-state-

off’.

EVENT: Enable dual-eval-t-wire-value; name this event ‘dual-eval-t-wire-value-off’.

EVENT: Enable t-buf-monotone; name this event ‘t-buf-monotone-off’.

EVENT: Enable dual-eval-t-buf-state; name this event ‘dual-eval-t-buf-state-off’.

EVENT: Enable dual-eval-t-buf-value; name this event ‘dual-eval-t-buf-value-off’.

EVENT: Enable ram-enable-circuit-monotone; name this event ‘ram-enable-circuit-monotone-off’.

EVENT: Enable dual-eval-ram-enable-circuit-state; name this event ‘dual-eval-ram-enable-circuit-state-off’.

EVENT: Enable dual-eval-ram-enable-circuit-value; name this event ‘dual-eval-ram-enable-circuit-value-off’.

EVENT: Enable id-monotone; name this event ‘id-monotone-off’.

EVENT: Enable dual-eval-id-state; name this event ‘dual-eval-id-state-off’.

EVENT: Enable dual-eval-id-value; name this event ‘dual-eval-id-value-off’.

EVENT: Enable fd1slp-monotone; name this event ‘fd1slp-monotone-off’.

EVENT: Enable dual-eval-fd1slp-state; name this event ‘dual-eval-fd1slp-state-off’.

EVENT: Enable dual-eval-fd1slp-value; name this event ‘dual-eval-fd1slp-value-off’.

EVENT: Enable fd1sp-monotone; name this event ‘fd1sp-monotone-off’.

EVENT: Enable dual-eval-fd1sp-state; name this event ‘dual-eval-fd1sp-state-off’.

EVENT: Enable dual-eval-fd1sp-value; name this event ‘dual-eval-fd1sp-value-off’.

EVENT: Enable fd1s-monotone; name this event ‘fd1s-monotone-off’.

EVENT: Enable dual-eval-fd1s-state; name this event ‘dual-eval-fd1s-state-off’.

EVENT: Enable dual-eval-fd1s-value; name this event ‘dual-eval-fd1s-value-off’.

EVENT: Enable fd1-monotone; name this event ‘fd1-monotone-off’.

EVENT: Enable dual-eval-fd1-state; name this event ‘dual-eval-fd1-state-off’.

EVENT: Enable dual-eval-fd1-value; name this event ‘dual-eval-fd1-value-off’.

EVENT: Enable procmon-monotone; name this event ‘procmon-monotone-off’.

EVENT: Enable dual-eval-procmon-state; name this event ‘dual-eval-procmon-state-off’.

EVENT: Enable dual-eval-procmon-value; name this event ‘dual-eval-procmon-value-off’.

EVENT: Enable del10-monotone; name this event ‘del10-monotone-off’.

EVENT: Enable dual-eval-del10-state; name this event ‘dual-eval-del10-state-off’.

EVENT: Enable dual-eval-del10-value; name this event ‘dual-eval-del10-value-off’.

EVENT: Enable del4-monotone; name this event ‘del4-monotone-off’.

EVENT: Enable dual-eval-del4-state; name this event ‘dual-eval-del4-state-off’.

EVENT: Enable dual-eval-del4-value; name this event ‘dual-eval-del4-value-off’.

EVENT: Enable del2-monotone; name this event ‘del2-monotone-off’.

EVENT: Enable dual-eval-del2-state; name this event ‘dual-eval-del2-state-off’.

EVENT: Enable dual-eval-del2-value; name this event ‘dual-eval-del2-value-off’.

EVENT: Enable b-xor3-monotone; name this event ‘b-xor3-monotone-off’.

EVENT: Enable dual-eval-b-xor3-state; name this event ‘dual-eval-b-xor3-state-off’.

EVENT: Enable dual-eval-b-xor3-value; name this event ‘dual-eval-b-xor3-value-off’.

EVENT: Enable b-xor-monotone; name this event ‘b-xor-monotone-off’.

EVENT: Enable dual-eval-b-xor-state; name this event ‘dual-eval-b-xor-state-off’.

EVENT: Enable dual-eval-b-xor-value; name this event ‘dual-eval-b-xor-value-off’.

EVENT: Enable b-or4-monotone; name this event ‘b-or4-monotone-off’.

EVENT: Enable dual-eval-b-or4-state; name this event ‘dual-eval-b-or4-state-off’.

EVENT: Enable dual-eval-b-or4-value; name this event ‘dual-eval-b-or4-value-off’.

EVENT: Enable b-or3-monotone; name this event ‘b-or3-monotone-off’.

EVENT: Enable dual-eval-b-or3-state; name this event ‘dual-eval-b-or3-state-off’.

EVENT: Enable dual-eval-b-or3-value; name this event ‘dual-eval-b-or3-value-off’.

EVENT: Enable b-or-monotone; name this event ‘b-or-monotone-off’.

EVENT: Enable dual-eval-b-or-state; name this event ‘dual-eval-b-or-state-off’.

EVENT: Enable dual-eval-b-or-value; name this event ‘dual-eval-b-or-value-off’.

EVENT: Enable b-not-ivap-monotone; name this event ‘b-not-ivap-monotone-off’.

EVENT: Enable dual-eval-b-not-ivap-state; name this event ‘dual-eval-b-not-ivap-state-off’.

EVENT: Enable dual-eval-b-not-ivap-value; name this event ‘dual-eval-b-not-ivap-value-off’.

EVENT: Enable b-not-b4ip-monotone; name this event ‘b-not-b4ip-monotone-off’.

EVENT: Enable dual-eval-b-not-b4ip-state; name this event ‘dual-eval-b-not-b4ip-state-off’.

EVENT: Enable dual-eval-b-not-b4ip-value; name this event ‘dual-eval-b-not-b4ip-value-off’.

EVENT: Enable b-not-monotone; name this event ‘b-not-monotone-off’.

EVENT: Enable dual-eval-b-not-state; name this event ‘dual-eval-b-not-state-off’.

EVENT: Enable dual-eval-b-not-value; name this event ‘dual-eval-b-not-value-off’.

EVENT: Enable b-nor8-monotone; name this event ‘b-nor8-monotone-off’.

EVENT: Enable dual-eval-b-nor8-state; name this event ‘dual-eval-b-nor8-state-off’.

EVENT: Enable dual-eval-b-nor8-value; name this event ‘dual-eval-b-nor8-value-off’.

EVENT: Enable b-nor6-monotone; name this event ‘b-nor6-monotone-off’.

EVENT: Enable dual-eval-b-nor6-state; name this event ‘dual-eval-b-nor6-state-off’.

EVENT: Enable dual-eval-b-nor6-value; name this event ‘dual-eval-b-nor6-value-off’.

EVENT: Enable b-nor5-monotone; name this event ‘b-nor5-monotone-off’.

EVENT: Enable dual-eval-b-nor5-state; name this event ‘dual-eval-b-nor5-state-off’.

EVENT: Enable dual-eval-b-nor5-value; name this event ‘dual-eval-b-nor5-value-off’.

EVENT: Enable b-nor4-monotone; name this event ‘b-nor4-monotone-off’.

EVENT: Enable dual-eval-b-nor4-state; name this event ‘dual-eval-b-nor4-state-off’.

EVENT: Enable dual-eval-b-nor4-value; name this event ‘dual-eval-b-nor4-value-off’.

EVENT: Enable b-nor3-monotone; name this event ‘b-nor3-monotone-off’.

EVENT: Enable dual-eval-b-nor3-state; name this event ‘dual-eval-b-nor3-state-off’.

EVENT: Enable dual-eval-b-nor3-value; name this event ‘dual-eval-b-nor3-value-off’.

EVENT: Enable b-nor-monotone; name this event ‘b-nor-monotone-off’.

EVENT: Enable dual-eval-b-nor-state; name this event ‘dual-eval-b-nor-state-off’.

EVENT: Enable dual-eval-b-nor-value; name this event ‘dual-eval-b-nor-value-off’.

EVENT: Enable b-nbuf-monotone; name this event ‘b-nbuf-monotone-off’.

EVENT: Enable dual-eval-b-nbuf-state; name this event ‘dual-eval-b-nbuf-state-off’.

EVENT: Enable dual-eval-b-nbuf-value; name this event ‘dual-eval-b-nbuf-value-off’.

EVENT: Enable b-nand8-monotone; name this event ‘b-nand8-monotone-off’.

EVENT: Enable dual-eval-b-nand8-state; name this event ‘dual-eval-b-nand8-state-off’.

EVENT: Enable dual-eval-b-nand8-value; name this event ‘dual-eval-b-nand8-value-off’.

EVENT: Enable b-nand6-monotone; name this event ‘b-nand6-monotone-off’.

EVENT: Enable dual-eval-b-nand6-state; name this event ‘dual-eval-b-nand6-state-off’.

EVENT: Enable dual-eval-b-nand6-value; name this event ‘dual-eval-b-nand6-value-off’.

EVENT: Enable b-nand5-monotone; name this event ‘b-nand5-monotone-off’.

EVENT: Enable dual-eval-b-nand5-state; name this event ‘dual-eval-b-nand5-state-off’.

EVENT: Enable dual-eval-b-nand5-value; name this event ‘dual-eval-b-nand5-value-off’.

EVENT: Enable b-nand4-monotone; name this event ‘b-nand4-monotone-off’.

EVENT: Enable dual-eval-b-nand4-state; name this event ‘dual-eval-b-nand4-state-off’.

EVENT: Enable dual-eval-b-nand4-value; name this event ‘dual-eval-b-nand4-value-off’.

EVENT: Enable b-nand3-monotone; name this event ‘b-nand3-monotone-off’.

EVENT: Enable dual-eval-b-nand3-state; name this event ‘dual-eval-b-nand3-state-off’.

EVENT: Enable dual-eval-b-nand3-value; name this event ‘dual-eval-b-nand3-value-off’.

EVENT: Enable b-nand-monotone; name this event ‘b-nand-monotone-off’.

EVENT: Enable dual-eval-b-nand-state; name this event ‘dual-eval-b-nand-state-off’.

EVENT: Enable dual-eval-b-nand-value; name this event ‘dual-eval-b-nand-value-off’.

EVENT: Enable b-if-monotone; name this event ‘b-if-monotone-off’.

EVENT: Enable dual-eval-b-if-state; name this event ‘dual-eval-b-if-state-off’.

EVENT: Enable dual-eval-b-if-value; name this event ‘dual-eval-b-if-value-off’.

EVENT: Enable b-equiv3-monotone; name this event ‘b-equiv3-monotone-off’.

EVENT: Enable dual-eval-b-equiv3-state; name this event 'dual-eval-b-equiv3-state-off'.

EVENT: Enable dual-eval-b-equiv3-value; name this event 'dual-eval-b-equiv3-value-off'.

EVENT: Enable b-equiv-monotone; name this event 'b-equiv-monotone-off'.

EVENT: Enable dual-eval-b-equiv-state; name this event 'dual-eval-b-equiv-state-off'.

EVENT: Enable dual-eval-b-equiv-value; name this event 'dual-eval-b-equiv-value-off'.

EVENT: Enable b-and4-monotone; name this event 'b-and4-monotone-off'.

EVENT: Enable dual-eval-b-and4-state; name this event 'dual-eval-b-and4-state-off'.

EVENT: Enable dual-eval-b-and4-value; name this event 'dual-eval-b-and4-value-off'.

EVENT: Enable b-and3-monotone; name this event 'b-and3-monotone-off'.

EVENT: Enable dual-eval-b-and3-state; name this event 'dual-eval-b-and3-state-off'.

EVENT: Enable dual-eval-b-and3-value; name this event 'dual-eval-b-and3-value-off'.

EVENT: Enable b-and-monotone; name this event 'b-and-monotone-off'.

EVENT: Enable dual-eval-b-and-state; name this event 'dual-eval-b-and-state-off'.

EVENT: Enable dual-eval-b-and-value; name this event 'dual-eval-b-and-value-off'.

EVENT: Enable ao7-monotone; name this event ‘ao7-monotone-off’.

EVENT: Enable dual-eval-ao7-state; name this event ‘dual-eval-ao7-state-off’.

EVENT: Enable dual-eval-ao7-value; name this event ‘dual-eval-ao7-value-off’.

EVENT: Enable ao6-monotone; name this event ‘ao6-monotone-off’.

EVENT: Enable dual-eval-ao6-state; name this event ‘dual-eval-ao6-state-off’.

EVENT: Enable dual-eval-ao6-value; name this event ‘dual-eval-ao6-value-off’.

EVENT: Enable ao4-monotone; name this event ‘ao4-monotone-off’.

EVENT: Enable dual-eval-ao4-state; name this event ‘dual-eval-ao4-state-off’.

EVENT: Enable dual-eval-ao4-value; name this event ‘dual-eval-ao4-value-off’.

EVENT: Enable ao2-monotone; name this event ‘ao2-monotone-off’.

EVENT: Enable dual-eval-ao2-state; name this event ‘dual-eval-ao2-state-off’.

EVENT: Enable dual-eval-ao2-value; name this event ‘dual-eval-ao2-value-off’.

EVENT: Enable fourp-implies-s-approx-is-b-approx; name this event ‘fourp-implies-s-approx-is-b-approx-off1’.

EVENT: Enable fourp-f-if; name this event ‘fourp-f-if-off’.

EVENT: Enable fourp-f-buf; name this event ‘fourp-f-buf-off’.

EVENT: Enable f-buf-type-set; name this event ‘f-buf-type-set-off’.

EVENT: Enable s-approx-implies-b-approx; name this event ‘s-approx-implies-b-approx-off’.

EVENT: Enable dual-eval-2-primp; name this event 'dual-eval-2-primp-off'.

EVENT: Enable dual-eval-0-primp; name this event 'dual-eval-0-primp-off'.

EVENT: Enable f-pullup-monotone; name this event 'f-pullup-monotone-off'.

EVENT: Enable ft-wire-monotone; name this event 'ft-wire-monotone-off'.

EVENT: Enable ft-buf-monotone; name this event 'ft-buf-monotone-off'.

EVENT: Enable f-if-monotone; name this event 'f-if-monotone-off'.

EVENT: Enable f-equiv3-monotone; name this event 'f-equiv3-monotone-off'.

EVENT: Enable f-xor3-monotone; name this event 'f-xor3-monotone-off'.

EVENT: Enable f-equiv-monotone; name this event 'f-equiv-monotone-off'.

EVENT: Enable f-xor-monotone; name this event 'f-xor-monotone-off'.

EVENT: Enable f-nor8-monotone; name this event 'f-nor8-monotone-off'.

EVENT: Enable f-nor6-monotone; name this event 'f-nor6-monotone-off'.

EVENT: Enable f-nor5-monotone; name this event 'f-nor5-monotone-off'.

EVENT: Enable f-nor4-monotone; name this event 'f-nor4-monotone-off'.

EVENT: Enable f-nor3-monotone; name this event 'f-nor3-monotone-off'.

EVENT: Enable f-nor-monotone; name this event 'f-nor-monotone-off'.

EVENT: Enable f-or4-monotone; name this event 'f-or4-monotone-off'.

EVENT: Enable f-or3-monotone; name this event 'f-or3-monotone-off'.

EVENT: Enable f-or-monotone; name this event ‘f-or-monotone-off’.

EVENT: Enable f-nand8-monotone; name this event ‘f-nand8-monotone-off’.

EVENT: Enable f-nand6-monotone; name this event ‘f-nand6-monotone-off’.

EVENT: Enable f-nand5-monotone; name this event ‘f-nand5-monotone-off’.

EVENT: Enable f-nand4-monotone; name this event ‘f-nand4-monotone-off’.

EVENT: Enable f-nand3-monotone; name this event ‘f-nand3-monotone-off’.

EVENT: Enable f-nand-monotone; name this event ‘f-nand-monotone-off’.

EVENT: Enable f-not-monotone; name this event ‘f-not-monotone-off’.

EVENT: Enable f-and4-monotone; name this event ‘f-and4-monotone-off’.

EVENT: Enable f-and3-monotone; name this event ‘f-and3-monotone-off’.

EVENT: Enable f-and-monotone; name this event ‘f-and-monotone-off’.

EVENT: Enable f-buf-monotone; name this event ‘f-buf-monotone-off’.

EVENT: Enable monotonicity-property-consequence-3; name this event ‘monotonicity-property-consequence-3-off’.

EVENT: Enable monotonicity-property-consequence-2; name this event ‘monotonicity-property-consequence-2-off’.

EVENT: Enable monotonicity-property-consequence-1; name this event ‘monotonicity-property-consequence-1-off’.

EVENT: Enable monotonicity-property-consequence-0; name this event ‘monotonicity-property-consequence-0-off’.

EVENT: Enable *1*monotonicity-property; name this event 'g*1*monotonicity-property-off'.

EVENT: Enable s-approx-alist; name this event 's-approx-alist-off'.

EVENT: Enable *1*s-approx-alist; name this event 'g*1*s-approx-alist-off'.

EVENT: Enable v-approx-alist; name this event 'v-approx-alist-off'.

EVENT: Enable *1*v-approx-alist; name this event 'g*1*v-approx-alist-off'.

EVENT: Enable s-knownnp-implies-s-approx-is-equal; name this event 's-knownnp-implies-s-approx-is-equal-off1'.

EVENT: Enable s-knownnp; name this event 's-knownnp-off1'.

EVENT: Enable *1*s-knownnp; name this event 'g*1*s-knownnp-off'.

EVENT: Enable good-s; name this event 'good-s-off'.

EVENT: Enable *1*good-s; name this event 'g*1*good-s-off'.

EVENT: Enable s-approx-x-x; name this event 's-approx-x-x-off'.

EVENT: Enable s-approx; name this event 's-approx-off1'.

EVENT: Enable *1*s-approx; name this event 'g*1*s-approx-off'.

EVENT: Enable mem-width; name this event 'mem-width-off1'.

EVENT: Enable *1*mem-width; name this event 'g*1*mem-width-off1'.

EVENT: Enable v-knownnp-implies-v-approx-is-equal; name this event 'v-knownnp-implies-v-approx-is-equal-off'.

EVENT: Enable v-knownnp; name this event 'v-knownnp-off1'.

EVENT: Enable *1*v-knownp; name this event 'g*1*v-knownp-off'.

EVENT: Enable v-approx; name this event 'v-approx-off1'.

EVENT: Enable *1*v-approx; name this event 'g*1*v-approx-off'.

EVENT: Enable *1*b-approx; name this event 'g*1*b-approx-off'.

EVENT: Enable b-knownp-compound-recognizer; name this event 'b-knownp-compound-recognizer-off'.

EVENT: Disable monotonicity-property; name this event 'monotonicity-property-on'.

EVENT: Disable monotonicity-property-opener-0; name this event 'monotonicity-property-opener-0-on'.

EVENT: Disable monotonicity-property-opener-1; name this event 'monotonicity-property-opener-1-on'.

EVENT: Disable monotonicity-property-opener-2; name this event 'monotonicity-property-opener-2-on'.

EVENT: Disable monotonicity-property-opener-3; name this event 'monotonicity-property-opener-3-on'.

EVENT: Disable f-buf; name this event 'f-buf-on'.

EVENT: Disable f-and; name this event 'f-and-on'.

EVENT: Disable f-and3; name this event 'f-and3-on'.

EVENT: Disable f-and4; name this event 'f-and4-on'.

EVENT: Disable f-not; name this event 'f-not-on'.

EVENT: Disable f-nand; name this event 'f-nand-on'.

EVENT: Disable f-nand3; name this event 'f-nand3-on'.

EVENT: Disable f-nand4; name this event 'f-nand4-on'.

EVENT: Disable f-nand5; name this event 'f-nand5-on'.

EVENT: Disable f-nand6; name this event 'f-nand6-on'.

EVENT: Disable f-nand8; name this event 'f-nand8-on'.

EVENT: Disable f-or; name this event 'f-or-on'.

EVENT: Disable f-or3; name this event 'f-or3-on'.

EVENT: Disable f-or4; name this event 'f-or4-on'.

EVENT: Disable f-nor; name this event 'f-nor-on'.

EVENT: Disable f-nor3; name this event 'f-nor3-on'.

EVENT: Disable f-nor4; name this event 'f-nor4-on'.

EVENT: Disable f-nor5; name this event 'f-nor5-on'.

EVENT: Disable f-nor6; name this event 'f-nor6-on'.

EVENT: Disable f-nor8; name this event 'f-nor8-on'.

EVENT: Disable f-xor; name this event 'f-xor-on'.

EVENT: Disable f-xor3; name this event 'f-xor3-on'.

EVENT: Disable f-equiv; name this event 'f-equiv-on'.

EVENT: Disable f-equiv3; name this event 'f-equiv3-on'.

EVENT: Disable f-if; name this event 'f-if-on'.

EVENT: Disable ft-buf; name this event 'ft-buf-on'.

EVENT: Disable ft-wire; name this event 'ft-wire-on'.

EVENT: Disable f-pullup; name this event 'f-pullup-on'.

EVENT: Disable bvp-reverse; name this event 'bvp-reverse-on'.

EVENT: Disable v-approx-bvp; name this event 'v-approx-bvp-on'.

EVENT: Disable read-mem1-monotone; name this event 'read-mem1-monotone-on'.

EVENT: Disable bvp-implies-v-knownp; name this event 'bvp-implies-v-knownp-on'.

EVENT: Disable v-approx-implies-b-approx-nth; name this event 'v-approx-implies-b-approx-nth-on'.

EVENT: Disable *1*b-knownp; name this event 'g*1*b-knownp-on1'.

EVENT: Disable v-approx-implies-subranges-equal; name this event 'v-approx-implies-subranges-equal-on'.

EVENT: Disable v-approx-bvp-subrange; name this event 'v-approx-bvp-subrange-on'.

EVENT: Disable v-approx-make-list-x; name this event 'v-approx-make-list-x-on'.

EVENT: Disable read-mem-monotone; name this event 'read-mem-monotone-on'.

EVENT: Disable equal-length-read-mem1; name this event ‘equal-length-read-mem1-on’.

EVENT: Disable equal-length-read-mem; name this event ‘equal-length-read-mem-on’.

EVENT: Disable s-approx-implies-properp-read-mem1; name this event ‘s-approx-implies-properp-read-mem1-on’.

EVENT: Disable s-approx-implies-properp-read-mem; name this event ‘s-approx-implies-properp-read-mem-on’.

EVENT: Disable dual-port-ram-value-monotone; name this event ‘dual-port-ram-value-monotone-on’.

EVENT: Disable eval\$-append; name this event ‘eval\$-append-on’.

EVENT: Disable eval\$-pairlist-cons; name this event ‘eval\$-pairlist-cons-on’.

EVENT: Disable eval\$-append-2; name this event ‘eval\$-append-2-on’.

EVENT: Disable dual-apply-value-dp-ram-16x32-lemma-1; name this event ‘dual-apply-value-dp-ram-16x32-lemma-1-on’.

EVENT: Disable dual-apply-value-dp-ram-16x32-lemma-2; name this event ‘dual-apply-value-dp-ram-16x32-lemma-2-on’.

EVENT: Disable dual-apply-value-dp-ram-16x32-lemma-3; name this event ‘dual-apply-value-dp-ram-16x32-lemma-3-on’.

EVENT: Disable dual-port-ram-value-is-dual-port-ram-value-body; name this event ‘dual-port-ram-value-is-dual-port-ram-value-body-on’.

EVENT: Disable dual-apply-value-dp-ram-16x32; name this event ‘dual-apply-value-dp-ram-16x32-on’.

EVENT: Disable dp-ram-16x32-monotone-value; name this event ‘dp-ram-16x32-

monotone-value-on’.

EVENT: Enable bvp; name this event ‘bvp-off2’.

EVENT: Disable s-approx-opener; name this event ‘s-approx-opener-on’.

EVENT: Disable v-approx-implies-nth-does-not-go-from-f-to-t; name this event ‘v-approx-implies-nth-does-not-go-from-f-to-t-on’.

EVENT: Disable write-mem1-opener; name this event ‘write-mem1-opener-on’.

EVENT: Disable write-mem1-monotone; name this event ‘write-mem1-monotone-on’.

EVENT: Disable write-mem-monotone; name this event ‘write-mem-monotone-on’.

EVENT: Disable v-approx-length; name this event ‘v-approx-length-on’.

EVENT: Disable s-approx-write-mem1-id; name this event ‘s-approx-write-mem1-id-on’.

EVENT: Disable s-approx-write-mem-id; name this event ‘s-approx-write-mem-id-on’.

EVENT: Disable s-approx-constant-ram-x-id; name this event ‘s-approx-constant-ram-x-id-on’.

EVENT: Disable s-approx-constant-ram-x-constant-ram-x; name this event ‘s-approx-constant-ram-x-constant-ram-x-on’.

EVENT: Disable v-approx-preserves-length; name this event ‘v-approx-preserves-length-on’.

EVENT: Disable v-approx-subrange; name this event ‘v-approx-subrange-on’.

EVENT: Disable mem-width-non-zero; name this event ‘mem-width-non-zero-

on’.

EVENT: Disable s-approx-constant-ram-x-write-mem1-case-3; name this event ‘s-approx-constant-ram-x-write-mem1-case-3-on’.

EVENT: Disable s-approx-constant-ram-x-write-mem1; name this event ‘s-approx-constant-ram-x-write-mem1-on’.

EVENT: Disable s-approx-constant-ram-x-write-mem; name this event ‘s-approx-constant-ram-x-write-mem-on’.

EVENT: Disable v-approx-preserves-properp; name this event ‘v-approx-preserves-properp-on’.

EVENT: Disable mem-width-linear-facts; name this event ‘mem-width-linear-facts-on’.

EVENT: Disable dual-port-ram-state-monotone-rewrite; name this event ‘dual-port-ram-state-monotone-rewrite-on’.

EVENT: Disable dual-apply-state-dp-ram-16x32-lemma-1; name this event ‘dual-apply-state-dp-ram-16x32-lemma-1-on’.

EVENT: Disable dual-apply-state-dp-ram-16x32-lemma-2; name this event ‘dual-apply-state-dp-ram-16x32-lemma-2-on’.

EVENT: Disable dual-apply-state-dp-ram-16x32-lemma-3; name this event ‘dual-apply-state-dp-ram-16x32-lemma-3-on’.

EVENT: Disable dual-port-ram-state-is-dual-port-ram-state-body; name this event ‘dual-port-ram-state-is-dual-port-ram-state-body-on’.

EVENT: Disable dual-apply-state-dp-ram-16x32; name this event ‘dual-apply-state-dp-ram-16x32-on’.

EVENT: Disable dp-ram-16x32-monotone-state; name this event ‘dp-ram-16x32-monotone-state-on’.

EVENT: Disable dp-ram-16x32-monotone; name this event ‘dp-ram-16x32-monotone-on’.

EVENT: Enable open-nth; name this event ‘open-nth-off1’.

EVENT: Disable doublet-p; name this event ‘doublet-p-on’.

EVENT: Disable for-final-1-of-reset-sequence-chip; name this event ‘for-final-1-of-reset-sequence-chip-on’.

EVENT: Disable map-down-inverts-map-up; name this event ‘map-down-inverts-map-up-on’.

EVENT: Disable b-knownp; name this event ‘b-knownp-on1’.

EVENT: Disable b-approx; name this event ‘b-approx-on1’.

EVENT: Disable cancel-constants-ilessp; name this event ‘cancel-constants-ilessp-on’.

EVENT: Disable cancel-constants-ilessp-lemma-2; name this event ‘cancel-constants-ilessp-lemma-2-on’.

EVENT: Disable cancel-constants-ilessp-lemma-1; name this event ‘cancel-constants-ilessp-lemma-1-on’.

EVENT: Disable illessp-add1-iplus; name this event ‘illessp-add1-iplus-on’.

EVENT: Disable illessp-add1; name this event ‘illessp-add1-on’.

EVENT: Disable cancel-constants-equal; name this event ‘cancel-constants-equal-on’.

EVENT: Disable cancel-constants-equal-lemma; name this event ‘cancel-constants-equal-lemma-on’.

EVENT: Disable iplus-constants; name this event ‘iplus-constants-on’.

EVENT: Disable correctness-of-cancel-ineg-terms-from-inequality; name this event 'correctness-of-cancel-ineg-terms-from-inequality-on'.

EVENT: Disable cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded; name this event 'cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded-on'.

EVENT: Disable *1*cancel-ineg-terms-from-inequality-expanded; name this event 'g*1*cancel-ineg-terms-from-inequality-expanded-on'.

EVENT: Disable cancel-ineg-terms-from-inequality; name this event 'cancel-ineg-terms-from-inequality-on'.

EVENT: Disable *1*cancel-ineg-terms-from-inequality; name this event 'g*1*cancel-ineg-terms-from-inequality-on'.

EVENT: Disable make-cancel-ineg-terms-inequality; name this event 'make-cancel-ineg-terms-inequality-on'.

EVENT: Disable *1*make-cancel-ineg-terms-inequality; name this event 'g*1*make-cancel-ineg-terms-inequality-on'.

EVENT: Disable correctness-of-cancel-ineg-terms-from-equality; name this event 'correctness-of-cancel-ineg-terms-from-equality-on'.

EVENT: Disable iplus-list-eval\$-car-split-out-ineg-terms; name this event 'iplus-list-eval\$-car-split-out-ineg-terms-on'.

EVENT: Disable eval\$-iplus-list-cdr-remove-inegs; name this event 'eval\$-iplus-list-cdr-remove-inegs-on'.

EVENT: Disable eval\$-iplus-list-car-remove-inegs; name this event 'eval\$-iplus-list-car-remove-inegs-on'.

EVENT: Disable cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded; name this event 'cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded-on'.

EVENT: Disable *1*cancel-ineg-terms-from-equality-expanded; name this event 'g*1*cancel-ineg-terms-from-equality-expanded-on'.

EVENT: Disable cancel-ineg-terms-from-equality; name this event 'cancel-ineg-terms-from-equality-on'.

EVENT: Disable *1*cancel-ineg-terms-from-equality; name this event 'g*1*cancel-ineg-terms-from-equality-on'.

EVENT: Disable make-cancel-ineg-terms-equality; name this event 'make-cancel-ineg-terms-equality-on'.

EVENT: Disable *1*make-cancel-ineg-terms-equality; name this event 'g*1*make-cancel-ineg-terms-equality-on'.

EVENT: Disable *1*iplus-or-ineg-term; name this event 'g*1*iplus-or-ineg-term-on'.

EVENT: Disable *1*remove-inegs; name this event 'g*1*remove-inegs-on'.

EVENT: Disable split-out-ineg-terms; name this event 'split-out-ineg-terms-on'.

EVENT: Disable *1*split-out-ineg-terms; name this event 'g*1*split-out-ineg-terms-on'.

EVENT: Disable correctness-of-cancel-factors-ilessp-0; name this event 'correctness-of-cancel-factors-ilessp-0-on'.

EVENT: Disable conjoin-inequalities-with-0-eliminator; name this event 'conjoin-inequalities-with-0-eliminator-on'.

EVENT: Disable cancel-factors-ilessp-0; name this event 'cancel-factors-ilessp-0-on'.

EVENT: Disable *1*cancel-factors-ilessp-0; name this event 'g*1*cancel-factors-ilessp-0-on'.

EVENT: Disable conjoin-inequalities-with-0; name this event ‘conjoin-inequalities-with-0-on’.

EVENT: Disable *1*conjoin-inequalities-with-0; name this event ‘g*1*conjoin-inequalities-with-0-on’.

EVENT: Disable correctness-of-cancel-factors-0; name this event ‘correctness-of-cancel-factors-0-on’.

EVENT: Disable some-eval\$\$s-to-0-eliminator; name this event ‘some-eval\$\$s-to-0-eliminator-on’.

EVENT: Disable some-eval\$\$s-to-0-append; name this event ‘some-eval\$\$s-to-0-append-on’.

EVENT: Disable eval\$-disjoin-equalities-with-0; name this event ‘eval\$-disjoin-equalities-with-0-on’.

EVENT: Disable some-eval\$\$s-to-0; name this event ‘some-eval\$\$s-to-0-on’.

EVENT: Disable *1*some-eval\$\$s-to-0; name this event ‘g*1*some-eval\$\$s-to-0-on’.

EVENT: Disable cancel-factors-0; name this event ‘cancel-factors-0-on’.

EVENT: Disable *1*cancel-factors-0; name this event ‘g*1*cancel-factors-0-on’.

EVENT: Disable disjoin-equalities-with-0; name this event ‘disjoin-equalities-with-0-on’.

EVENT: Disable *1*disjoin-equalities-with-0; name this event ‘g*1*disjoin-equalities-with-0-on’.

EVENT: Disable correctness-of-cancel-itimes-ilessp-factors; name this event ‘correctness-of-cancel-itimes-ilessp-factors-on’.

EVENT: Disable equal-fix-int-to-ilessp; name this event ‘equal-fix-int-to-ilessp-

on’.

EVENT: Disable eval\$list-bagint-0-implies-equal-for-ilessp-lemma; name this event ‘eval\$list-bagint-0-implies-equal-for-ilessp-lemma-on’.

EVENT: Disable eval\$list-bagint-0-for-ilessp; name this event ‘eval\$list-bagint-0-for-ilessp-on’.

EVENT: Disable member-0-itimes-factors-yields-0-ilessp-consequence-2; name this event ‘member-0-itimes-factors-yields-0-ilessp-consequence-2-on’.

EVENT: Disable member-0-itimes-factors-yields-0-ilessp-consequence-1; name this event ‘member-0-itimes-factors-yields-0-ilessp-consequence-1-on’.

EVENT: Disable member-0-itimes-factors-yields-0; name this event ‘member-0-itimes-factors-yields-0-on’.

EVENT: Disable ilessp-itimes-list-eval\$list-bagdiff-corollary-2; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-2-on’.

EVENT: Disable ilessp-zero-implies-not-equal; name this event ‘ilessp-zero-implies-not-equal-on’.

EVENT: Disable ilessp-itimes-list-eval\$list-bagdiff-corollary-1; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-1-on’.

EVENT: Disable zero-ilessp-implies-not-equal; name this event ‘zero-ilessp-implies-not-equal-on’.

EVENT: Disable ilessp-itimes-list-eval\$list-bagdiff; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-on’.

EVENT: Disable bagint-singleton; name this event ‘bagint-singleton-on’.

EVENT: Disable cancel-itimes-ilessp-factors; name this event ‘cancel-itimes-ilessp-factors-on’.

EVENT: Disable $*1*$ cancel-itimes-ilessp-factors; name this event ' $g*1*$ cancel-itimes-ilessp-factors-on'.

EVENT: Disable correctness-of-cancel-itimes-factors; name this event 'correctness-of-cancel-itimes-factors-on'.

EVENT: Disable member-0-eval\$list; name this event 'member-0-eval\$list-on'.

EVENT: Disable membership-of-0-implies-itimes-list-is-0; name this event 'membership-of-0-implies-itimes-list-is-0-on'.

EVENT: Disable cancel-itimes-factors-expanded-cancel-itimes-factors; name this event 'cancel-itimes-factors-expanded-cancel-itimes-factors-on'.

EVENT: Disable $*1*$ cancel-itimes-factors-expanded; name this event ' $g*1*$ cancel-itimes-factors-expanded-on'.

EVENT: Disable cancel-itimes-factors; name this event 'cancel-itimes-factors-on'.

EVENT: Disable $*1*$ cancel-itimes-factors; name this event ' $g*1*$ cancel-itimes-factors-on'.

EVENT: Disable $*1*$ iplus-or-itimes-term; name this event ' $g*1*$ iplus-or-itimes-term-on'.

EVENT: Disable itimes-eval\$-itimes-tree-ineg; name this event 'itimes-eval\$-itimes-tree-ineg-on'.

EVENT: Disable iplus-eval\$-itimes-tree-ineg; name this event 'iplus-eval\$-itimes-tree-ineg-on'.

EVENT: Disable neg-eval\$-itimes-tree-ineg; name this event 'neg-eval\$-itimes-tree-ineg-on'.

EVENT: Disable eval\$-itimes-tree-ineg; name this event 'eval\$-itimes-tree-ineg-on'.

EVENT: Disable fix-int-eval\$-itimes-tree-rec; name this event ‘fix-int-eval\$-itimes-tree-rec-on’.

EVENT: Disable illessp-ineg-ineg; name this event ‘illessp-ineg-ineg-on’.

EVENT: Disable equal-ineg-ineg; name this event ‘equal-ineg-ineg-on’.

EVENT: Disable itimes-1; name this event ‘itimes-1-on’.

EVENT: Disable itimes-factors; name this event ‘itimes-factors-on’.

EVENT: Disable *1*itimes-factors; name this event ‘g*1*itimes-factors-on’.

EVENT: Disable *1*itimes-tree-ineg; name this event ‘g*1*itimes-tree-ineg-on’.

EVENT: Disable illessp-strict; name this event ‘illessp-strict-on’.

EVENT: Disable illessp-itimes-right-negative; name this event ‘illessp-itimes-right-negative-on’.

EVENT: Disable illessp-itimes-right-positive; name this event ‘illessp-itimes-right-positive-on’.

EVENT: Disable cancel-itimes-illessp; name this event ‘cancel-itimes-illessp-on’.

EVENT: Disable *1*cancel-itimes-illessp; name this event ‘g*1*cancel-itimes-illessp-on’.

EVENT: Disable *1*make-cancel-itimes-inequality; name this event ‘g*1*make-cancel-itimes-inequality-on’.

EVENT: Disable *1*itimes-tree-no-fix-int; name this event ‘g*1*itimes-tree-no-fix-int-on’.

EVENT: Disable subsetp; name this event ‘subsetp-on’.

EVENT: Disable *1*subsetp; name this event 'g*1*subsetp-on'.

EVENT: Disable equal-fix-int; name this event 'equal-fix-int-on'.

EVENT: Disable member-append; name this event 'member-append-on'.

EVENT: Disable itimes-list-append; name this event 'itimes-list-append-on'.

EVENT: Disable cancel-itimes; name this event 'cancel-itimes-on'.

EVENT: Disable *1*cancel-itimes; name this event 'g*1*cancel-itimes-on'.

EVENT: Disable *1*make-cancel-itimes-equality; name this event 'g*1*make-cancel-itimes-equality-on'.

EVENT: Disable integerp-itimes-list; name this event 'integerp-itimes-list-on'.

EVENT: Disable itimes-list; name this event 'itimes-list-on'.

EVENT: Disable *1*itimes-list; name this event 'g*1*itimes-list-on'.

EVENT: Disable *1*itimes-tree; name this event 'g*1*itimes-tree-on'.

EVENT: Disable itimes-tree-rec; name this event 'itimes-tree-rec-on'.

EVENT: Disable *1*itimes-tree-rec; name this event 'g*1*itimes-tree-rec-on'.

EVENT: Disable itimes-fringe; name this event 'itimes-fringe-on'.

EVENT: Disable *1*itimes-fringe; name this event 'g*1*itimes-fringe-on'.

EVENT: Disable fix-int-irem; name this event 'fix-int-irem-on'.

EVENT: Disable fix-int-iquo; name this event 'fix-int-iquo-on'.

EVENT: Disable fix-int-imod; name this event 'fix-int-imod-on'.

EVENT: Disable fix-int-idiv; name this event 'fix-int-idiv-on'.

EVENT: Disable fix-int-iremainder; name this event 'fix-int-iremainder-on'.

EVENT: Disable fix-int-iquotient; name this event 'fix-int-iquotient-on'.

EVENT: Disable irem-fix-int2; name this event 'irem-fix-int2-on'.

EVENT: Disable irem-fix-int1; name this event 'irem-fix-int1-on'.

EVENT: Disable iquo-fix-int2; name this event 'iquo-fix-int2-on'.

EVENT: Disable iquo-fix-int1; name this event 'iquo-fix-int1-on'.

EVENT: Disable imod-fix-int2; name this event 'imod-fix-int2-on'.

EVENT: Disable imod-fix-int1; name this event 'imod-fix-int1-on'.

EVENT: Disable idiv-fix-int2; name this event 'idiv-fix-int2-on'.

EVENT: Disable idiv-fix-int1; name this event 'idiv-fix-int1-on'.

EVENT: Disable iremainder-fix-int2; name this event 'iremainder-fix-int2-on'.

EVENT: Disable iremainder-fix-int1; name this event 'iremainder-fix-int1-on'.

EVENT: Disable iquotient-fix-int2; name this event 'iquotient-fix-int2-on'.

EVENT: Disable iquotient-fix-int1; name this event 'iquotient-fix-int1-on'.

EVENT: Disable integerp-irem; name this event 'integerp-irem-on'.

EVENT: Disable integerp-iquo; name this event 'integerp-iquo-on'.

EVENT: Disable integerp-imod; name this event 'integerp-imod-on'.

EVENT: Disable integerp-idiv; name this event 'integerp-idiv-on'.

EVENT: Disable integerp-remainder; name this event 'integerp-remainder-on'.

EVENT: Disable integerp-quotient; name this event 'integerp-quotient-on'.

EVENT: Disable itimes-cancellation-3; name this event 'itimes-cancellation-3-on'.

EVENT: Disable itimes-cancellation-2; name this event 'itimes-cancellation-2-on'.

EVENT: Disable itimes-cancellation-1; name this event 'itimes-cancellation-1-on'.

EVENT: Disable itimes-ineg-2; name this event 'itimes-ineg-2-on'.

EVENT: Disable itimes-ineg-1; name this event 'itimes-ineg-1-on'.

EVENT: Disable iquo-irem-uniqueness; name this event 'iquo-irem-uniqueness-on'.

EVENT: Disable division-theorem-for-truncate-to-zero; name this event 'division-theorem-for-truncate-to-zero-on'.

EVENT: Disable division-theorem-for-truncate-to-zero-part3; name this event 'division-theorem-for-truncate-to-zero-part3-on'.

EVENT: Disable division-theorem-for-truncate-to-zero-part2; name this event 'division-theorem-for-truncate-to-zero-part2-on'.

EVENT: Disable division-theorem-for-truncate-to-zero-part1; name this event 'division-theorem-for-truncate-to-zero-part1-on'.

EVENT: Disable idiv-imod-uniqueness; name this event 'idiv-imod-uniqueness-

on’.

EVENT: Disable division-theorem-for-truncate-to-neginf; name this event ‘division-theorem-for-truncate-to-neginf-on’.

EVENT: Disable division-theorem-for-truncate-to-neginf-part3; name this event ‘division-theorem-for-truncate-to-neginf-part3-on’.

EVENT: Disable division-theorem-for-truncate-to-neginf-part2; name this event ‘division-theorem-for-truncate-to-neginf-part2-on’.

EVENT: Disable division-theorem-for-truncate-to-neginf-part1; name this event ‘division-theorem-for-truncate-to-neginf-part1-on’.

EVENT: Disable iquotient-iremainder-uniqueness; name this event ‘iquotient-iremainder-uniqueness-on’.

EVENT: Disable division-theorem; name this event ‘division-theorem-on’.

EVENT: Disable division-theorem-part3; name this event ‘division-theorem-part3-on’.

EVENT: Disable division-theorem-part2; name this event ‘division-theorem-part2-on’.

EVENT: Disable division-theorem-part1; name this event ‘division-theorem-part1-on’.

EVENT: Disable quotient-remainder-uniqueness; name this event ‘quotient-remainder-uniqueness-on’.

EVENT: Disable itimes-1-arg1; name this event ‘itimes-1-arg1-on’.

EVENT: Disable equal-itimes-minus-1; name this event ‘equal-itimes-minus-1-on’.

EVENT: Disable equal-itimes-1; name this event ‘equal-itimes-1-on’.

EVENT: Disable equal-itimes-0; name this event 'equal-itimes-0-on'.

EVENT: Disable associativity-of-itimes; name this event 'associativity-of-itimes-on'.

EVENT: Disable commutativity2-of-itimes; name this event 'commutativity2-of-itimes-on'.

EVENT: Disable itimes-distributes-over-iplus; name this event 'itimes-distributes-over-iplus-on'.

EVENT: Disable itimes-distributes-over-iplus-proof; name this event 'itimes-distributes-over-iplus-proof-on'.

EVENT: Disable commutativity-of-itimes; name this event 'commutativity-of-itimes-on'.

EVENT: Disable itimes-fix-int2; name this event 'itimes-fix-int2-on'.

EVENT: Disable itimes-fix-int1; name this event 'itimes-fix-int1-on'.

EVENT: Disable itimes-0-right; name this event 'itimes-0-right-on'.

EVENT: Disable itimes-0-left; name this event 'itimes-0-left-on'.

EVENT: Disable correctness-of-cancel-iplus-ilessp; name this event 'correctness-of-cancel-iplus-ilessp-on'.

EVENT: Disable cancel-iplus-ilessp; name this event 'cancel-iplus-ilessp-on'.

EVENT: Disable *1*cancel-iplus-ilessp; name this event 'g*1*cancel-iplus-ilessp-on'.

EVENT: Disable *1*iplus-tree-no-fix-int; name this event 'g*1*iplus-tree-no-fix-int-on'.

EVENT: Disable iplus-cancellation-2-for-ilessp; name this event 'iplus-cancellation-

2-for-ilessp-on’.

EVENT: Disable iplus-cancellation-1-for-ilessp; name this event ‘iplus-cancellation-1-for-ilessp-on’.

EVENT: Disable cancel-iplus-ilessp-1; name this event ‘cancel-iplus-ilessp-1-on’.

EVENT: Disable *1*cancel-iplus-ilessp-1; name this event ‘g*1*cancel-iplus-ilessp-1-on’.

EVENT: Disable *1*make-cancel-iplus-inequality-1; name this event ‘g*1*make-cancel-iplus-inequality-1-on’.

EVENT: Disable ilessp-fix-int-2; name this event ‘ilessp-fix-int-2-on’.

EVENT: Disable ilessp-fix-int-1; name this event ‘ilessp-fix-int-1-on’.

EVENT: Disable correctness-of-cancel-iplus; name this event ‘correctness-of-cancel-iplus-on’.

EVENT: Disable iplus-ineg7; name this event ‘iplus-ineg7-on’.

EVENT: Disable iplus-list-append; name this event ‘iplus-list-append-on’.

EVENT: Disable eval\$-iplus-list-delete; name this event ‘eval\$-iplus-list-delete-on’.

EVENT: Disable *1*cancel-iplus; name this event ‘g*1*cancel-iplus-on’.

EVENT: Disable eval\$list-append; name this event ‘eval\$list-append-on’.

EVENT: Disable integerp-iplus-list; name this event ‘integerp-iplus-list-on’.

EVENT: Disable iplus-list; name this event ‘iplus-list-on’.

EVENT: Disable *1*iplus-list; name this event ‘g*1*iplus-list-on’.

EVENT: Disable $*1*$ iplus-tree; name this event ' $g*1*$ iplus-tree-on'.

EVENT: Disable iplus-tree-rec; name this event 'iplus-tree-rec-on'.

EVENT: Disable $*1*$ iplus-tree-rec; name this event ' $g*1*$ iplus-tree-rec-on'.

EVENT: Disable iplus-fringe; name this event 'iplus-fringe-on'.

EVENT: Disable $*1*$ iplus-fringe; name this event ' $g*1*$ iplus-fringe-on'.

EVENT: Disable correctness-of-cancel-ineg; name this event 'correctness-of-cancel-ineg-on'.

EVENT: Disable iplus-x-y-ineg-x; name this event 'iplus-x-y-ineg-x-on'.

EVENT: Disable iplus-ineg4; name this event 'iplus-ineg4-on'.

EVENT: Disable iplus-ineg3; name this event 'iplus-ineg3-on'.

EVENT: Disable eval\$-cancel-ineg-aux-is-its-fn; name this event 'eval\$-cancel-ineg-aux-is-its-fn-on'.

EVENT: Disable eval\$-cancel-ineg-aux-fn; name this event 'eval\$-cancel-ineg-aux-fn-on'.

EVENT: Disable $*1*$ eval\$-cancel-ineg-aux-fn; name this event ' $g*1*$ eval\$-cancel-ineg-aux-fn-on'.

EVENT: Disable eval\$-other; name this event 'eval\$-other-on'.

EVENT: Disable eval\$-litatom; name this event 'eval\$-litatom-on'.

EVENT: Disable eval\$-list-nlistp; name this event 'eval\$-list-nlistp-on'.

EVENT: Disable eval\$-list-cons; name this event 'eval\$-list-cons-on'.

EVENT: Disable cancel-ineg; name this event 'cancel-ineg-on'.

EVENT: Disable $*1*$ cancel-ineg; name this event ' $g*1*$ cancel-ineg-on'.

EVENT: Disable cancel-ineg-aux; name this event 'cancel-ineg-aux-on'.

EVENT: Disable $*1*$ cancel-ineg-aux; name this event ' $g*1*$ cancel-ineg-aux-on'.

EVENT: Disable idifference-fix-int2; name this event 'idifference-fix-int2-on'.

EVENT: Disable idifference-fix-int1; name this event 'idifference-fix-int1-on'.

EVENT: Disable iplus-fix-int2; name this event 'iplus-fix-int2-on'.

EVENT: Disable iplus-fix-int1; name this event 'iplus-fix-int1-on'.

EVENT: Disable iplus-ineg2; name this event 'iplus-ineg2-on'.

EVENT: Disable iplus-ineg1; name this event 'iplus-ineg1-on'.

EVENT: Disable iplus-cancellation-2; name this event 'iplus-cancellation-2-on'.

EVENT: Disable iplus-cancellation-1; name this event 'iplus-cancellation-1-on'.

EVENT: Disable associativity-of-iplus; name this event 'associativity-of-iplus-on'.

EVENT: Disable commutativity-of-iplus; name this event 'commutativity-of-iplus-on'.

EVENT: Disable commutativity2-of-iplus; name this event 'commutativity2-of-iplus-on'.

EVENT: Disable iplus-0-right; name this event 'iplus-0-right-on'.

EVENT: Disable iplus-0-left; name this event 'iplus-0-left-on'.

EVENT: Disable neg-0; name this event 'neg-0-on'.

EVENT: Disable neg-fix-int; name this event 'neg-fix-int-on'.

EVENT: Disable neg-neg; name this event 'neg-neg-on'.

EVENT: Disable neg-iplus; name this event 'neg-iplus-on'.

EVENT: Disable fix-int-itimes; name this event 'fix-int-itimes-on'.

EVENT: Disable fix-int-iabs; name this event 'fix-int-iabs-on'.

EVENT: Disable fix-int-neg; name this event 'fix-int-neg-on'.

EVENT: Disable fix-int-idifference; name this event 'fix-int-idifference-on'.

EVENT: Disable fix-int-iplus; name this event 'fix-int-iplus-on'.

EVENT: Disable fix-int-fix-int; name this event 'fix-int-fix-int-on'.

EVENT: Disable fix-int-remover; name this event 'fix-int-remover-on'.

EVENT: Disable integerp-itimes; name this event 'integerp-itimes-on'.

EVENT: Disable integerp-iabs; name this event 'integerp-iabs-on'.

EVENT: Disable integerp-neg; name this event 'integerp-neg-on'.

EVENT: Disable integerp-idifference; name this event 'integerp-idifference-on'.

EVENT: Disable integerp-iplus; name this event 'integerp-iplus-on'.

EVENT: Disable integerp-fix-int; name this event 'integerp-fix-int-on'.

EVENT: Disable *1*irem; name this event 'g*1*irem-on'.

EVENT: Disable $*1*$ quo; name this event ' $g*1*$ quo-on'.

EVENT: Disable $*1*$ mod; name this event ' $g*1*$ mod-on'.

EVENT: Disable $*1*$ div; name this event ' $g*1*$ div-on'.

EVENT: Disable $*1*$ remainder; name this event ' $g*1*$ remainder-on'.

EVENT: Disable $*1*$ quotient; name this event ' $g*1*$ quotient-on'.

EVENT: Disable $*1*$ times; name this event ' $g*1*$ times-on'.

EVENT: Disable $*1*$ iabs; name this event ' $g*1*$ iabs-on'.

EVENT: Disable idifference; name this event 'idifference-on'.

EVENT: Disable $*1*$ idifference; name this event ' $g*1*$ idifference-on'.

EVENT: Disable $*1*$ ineg; name this event ' $g*1*$ ineg-on'.

EVENT: Disable $*1*$ iplus; name this event ' $g*1*$ iplus-on'.

EVENT: Disable ileq; name this event 'ileq-on'.

EVENT: Disable $*1*$ ileq; name this event ' $g*1*$ ileq-on'.

EVENT: Disable $*1*$ ilessp; name this event ' $g*1*$ ilessp-on'.

EVENT: Disable izerop; name this event 'izerop-on'.

EVENT: Disable $*1*$ izerop; name this event ' $g*1*$ izerop-on'.

EVENT: Disable $*1*$ fix-int; name this event ' $g*1*$ fix-int-on'.

EVENT: Disable $*1*$ integerp; name this event ' $g*1*$ integerp-on'.

EVENT: Disable gcd-idempotence; name this event 'gcd-idempotence-on'.

EVENT: Disable gcd-x-x; name this event 'gcd-x-x-on'.

EVENT: Disable commutativity2-of-gcd; name this event 'commutativity2-of-gcd-on'.

EVENT: Disable commutativity2-of-gcd-zero-case; name this event 'commutativity2-of-gcd-zero-case-on'.

EVENT: Disable associativity-of-gcd; name this event 'associativity-of-gcd-on'.

EVENT: Disable associativity-of-gcd-zero-case; name this event 'associativity-of-gcd-zero-case-on'.

EVENT: Disable common-divisor-divides-gcd; name this event 'common-divisor-divides-gcd-on'.

EVENT: Disable gcd-is-the-greatest; name this event 'gcd-is-the-greatest-on'.

EVENT: Disable distributivity-of-times-over-gcd; name this event 'distributivity-of-times-over-gcd-on'.

EVENT: Disable distributivity-of-times-over-gcd-proof; name this event 'distributivity-of-times-over-gcd-proof-on'.

EVENT: Disable remainder-gcd; name this event 'remainder-gcd-on'.

EVENT: Disable gcd-plus-instance; name this event 'gcd-plus-instance-on'.

EVENT: Disable gcd-plus; name this event 'gcd-plus-on'.

EVENT: Disable gcd-plus-proof; name this event 'gcd-plus-proof-on'.

EVENT: Disable gcd-plus-instance-temp; name this event 'gcd-plus-instance-temp-on'.

EVENT: Disable gcd-plus-instance-temp-proof; name this event 'gcd-plus-instance-temp-proof-on'.

EVENT: Disable lessp-gcd; name this event 'lessp-gcd-on'.

EVENT: Disable equal-gcd-0; name this event 'equal-gcd-0-on'.

EVENT: Disable gcd-1; name this event 'gcd-1-on'.

EVENT: Disable gcd-0; name this event 'gcd-0-on'.

EVENT: Disable single-number-induction; name this event 'single-number-induction-on'.

EVENT: Disable *1*single-number-induction; name this event 'g*1*single-number-induction-on'.

EVENT: Disable commutativity-of-gcd; name this event 'commutativity-of-gcd-on'.

EVENT: Disable log-exp; name this event 'log-exp-on'.

EVENT: Disable log-times-exp; name this event 'log-times-exp-on'.

EVENT: Disable log-times-exp-proof; name this event 'log-times-exp-proof-on'.

EVENT: Disable log-times; name this event 'log-times-on'.

EVENT: Disable log-times-proof; name this event 'log-times-proof-on'.

EVENT: Disable log-quotient-exp; name this event 'log-quotient-exp-on'.

EVENT: Disable log-quotient-times; name this event 'log-quotient-times-on'.

EVENT: Disable log-quotient-times-proof; name this event 'log-quotient-times-proof-on'.

EVENT: Disable log-quotient; name this event 'log-quotient-on'.

EVENT: Disable leq-log-log; name this event 'leq-log-log-on'.

EVENT: Disable double-log-induction; name this event 'double-log-induction-on'.

EVENT: Disable $*1*$ double-log-induction; name this event ' $g*1*$ double-log-induction-on'.

EVENT: Disable log-1; name this event 'log-1-on'.

EVENT: Disable log-0; name this event 'log-0-on'.

EVENT: Disable equal-log-0; name this event 'equal-log-0-on'.

EVENT: Disable exp-difference; name this event 'exp-difference-on'.

EVENT: Disable equal-exp-1; name this event 'equal-exp-1-on'.

EVENT: Disable equal-exp-0; name this event 'equal-exp-0-on'.

EVENT: Disable exp-exp; name this event 'exp-exp-on'.

EVENT: Disable exp-times; name this event 'exp-times-on'.

EVENT: Disable exp-0-arg2; name this event 'exp-0-arg2-on'.

EVENT: Disable exp-1-arg1; name this event 'exp-1-arg1-on'.

EVENT: Disable exp-0-arg1; name this event 'exp-0-arg1-on'.

EVENT: Disable exp-plus; name this event 'exp-plus-on'.

EVENT: Disable exp-add1; name this event 'exp-add1-on'.

EVENT: Disable exp-zero; name this event 'exp-zero-on'.

EVENT: Disable quotient-exp; name this event 'quotient-exp-on'.

EVENT: Disable remainder-exp-exp; name this event 'remainder-exp-exp-on'.

EVENT: Disable double-number-induction; name this event 'double-number-induction-on'.

EVENT: Disable *1*double-number-induction; name this event 'g*1*double-number-induction-on'.

EVENT: Disable remainder-exp; name this event 'remainder-exp-on'.

EVENT: Disable gcd; name this event 'gcd-on'.

EVENT: Disable *1*gcd; name this event 'g*1*gcd-on'.

EVENT: Disable log; name this event 'log-on'.

EVENT: Disable *1*log; name this event 'g*1*log-on'.

EVENT: Disable exp; name this event 'exp-on'.

EVENT: Disable *1*exp; name this event 'g*1*exp-on'.

EVENT: Disable correctness-of-cancel-quotient-times; name this event 'correctness-of-cancel-quotient-times-on'.

EVENT: Disable cancel-quotient-times; name this event 'cancel-quotient-times-on'.

EVENT: Disable *1*cancel-quotient-times; name this event 'g*1*cancel-quotient-times-on'.

EVENT: Disable lessp-quotient; name this event 'lessp-quotient-on'.

EVENT: Disable quotient-x-x; name this event 'quotient-x-x-on'.

EVENT: Disable quotient-1-arg1; name this event 'quotient-1-arg1-on'.

EVENT: Disable quotient-1-arg1-casesplit; name this event 'quotient-1-arg1-casesplit-on'.

EVENT: Disable quotient-1-arg2; name this event 'quotient-1-arg2-on'.

EVENT: Disable leq-quotient; name this event 'leq-quotient-on'.

EVENT: Disable quotient-quotient; name this event 'quotient-quotient-on'.

EVENT: Disable quotient-plus-times-times-instance; name this event 'quotient-plus-times-times-instance-on'.

EVENT: Disable quotient-plus-times-times; name this event 'quotient-plus-times-times-on'.

EVENT: Disable quotient-plus-times-times-proof; name this event 'quotient-plus-times-times-proof-on'.

EVENT: Disable quotient-plus-fact; name this event 'quotient-plus-fact-on'.

EVENT: Disable quotient-remainder-instance; name this event 'quotient-remainder-instance-on'.

EVENT: Disable quotient-remainder; name this event 'quotient-remainder-on'.

EVENT: Disable quotient-remainder-times; name this event 'quotient-remainder-times-on'.

EVENT: Disable quotient-difference3; name this event 'quotient-difference3-on'.

EVENT: Disable quotient-difference2; name this event 'quotient-difference2-on'.

EVENT: Disable quotient-lessp-arg1; name this event 'quotient-lessp-arg1-on'.

EVENT: Disable quotient-difference1; name this event 'quotient-difference1-on'.

EVENT: Disable quotient-times-times-proof; name this event 'quotient-times-times-proof-on'.

EVENT: Disable quotient-times-instance; name this event 'quotient-times-instance-on'.

EVENT: Disable quotient-times; name this event 'quotient-times-on'.

EVENT: Disable quotient-times-proof; name this event 'quotient-times-proof-on'.

EVENT: Disable quotient-times-instance-temp-proof; name this event 'quotient-times-instance-temp-proof-on'.

EVENT: Disable quotient-plus; name this event 'quotient-plus-on'.

EVENT: Disable quotient-plus-proof; name this event 'quotient-plus-proof-on'.

EVENT: Disable quotient-sub1; name this event 'quotient-sub1-on'.

EVENT: Disable equal-quotient-0; name this event 'equal-quotient-0-on'.

EVENT: Disable quotient-add1; name this event 'quotient-add1-on'.

EVENT: Disable quotient-zero; name this event 'quotient-zero-on'.

EVENT: Disable quotient-of-non-number; name this event 'quotient-of-non-number-on'.

EVENT: Disable quotient-noop; name this event 'quotient-noop-on'.

EVENT: Disable transitivity-of-divides; name this event 'transitivity-of-divides-

on’.

EVENT: Disable remainder-x-x; name this event ‘remainder-x-x-on’.

EVENT: Disable remainder-1-arg2; name this event ‘remainder-1-arg2-on’.

EVENT: Disable remainder-1-arg1; name this event ‘remainder-1-arg1-on’.

EVENT: Disable remainder-remainder; name this event ‘remainder-remainder-on’.

EVENT: Disable remainder-plus-times-times-instance; name this event ‘remainder-plus-times-times-instance-on’.

EVENT: Disable remainder-plus-times-times; name this event ‘remainder-plus-times-times-on’.

EVENT: Disable remainder-plus-times-times-proof; name this event ‘remainder-plus-times-times-proof-on’.

EVENT: Disable remainder-plus-fact; name this event ‘remainder-plus-fact-on’.

EVENT: Disable remainder-difference2; name this event ‘remainder-difference2-on’.

EVENT: Disable double-remainder-induction; name this event ‘double-remainder-induction-on’.

EVENT: Disable *1*double-remainder-induction; name this event ‘g*1*double-remainder-induction-on’.

EVENT: Disable remainder-difference1; name this event ‘remainder-difference1-on’.

EVENT: Disable remainder-times2-instance; name this event ‘remainder-times2-instance-on’.

EVENT: Disable remainder-times2; name this event ‘remainder-times2-on’.

EVENT: Disable remainder-times2-proof; name this event ‘remainder-times2-proof-on’.

EVENT: Disable remainder-times-times-proof; name this event ‘remainder-times-times-proof-on’.

EVENT: Disable remainder-times1-instance; name this event ‘remainder-times1-instance-on’.

EVENT: Disable remainder-times1-instance-proof; name this event ‘remainder-times1-instance-proof-on’.

EVENT: Disable remainder-times1; name this event ‘remainder-times1-on’.

EVENT: Disable remainder-times1-proof; name this event ‘remainder-times1-proof-on’.

EVENT: Disable equal-remainder-plus-remainder-proof; name this event ‘equal-remainder-plus-remainder-proof-on’.

EVENT: Disable equal-remainder-plus-0; name this event ‘equal-remainder-plus-0-on’.

EVENT: Disable equal-remainder-plus-0-proof; name this event ‘equal-remainder-plus-0-proof-on’.

EVENT: Disable remainder-plus; name this event ‘remainder-plus-on’.

EVENT: Disable remainder-plus-proof; name this event ‘remainder-plus-proof-on’.

EVENT: Disable remainder-add1; name this event ‘remainder-add1-on’.

EVENT: Disable remainder-quotient-elim; name this event ‘remainder-quotient-elim-on’.

EVENT: Disable remainder-zero; name this event ‘remainder-zero-on’.

EVENT: Disable remainder-of-non-number; name this event ‘remainder-of-non-number-on’.

EVENT: Disable remainder-noop; name this event ‘remainder-noop-on’.

EVENT: Disable lessp-remainder; name this event ‘lessp-remainder-on’.

EVENT: Disable correctness-of-cancel-equal-times; name this event ‘correctness-of-cancel-equal-times-on’.

EVENT: Disable cancel-equal-times; name this event ‘cancel-equal-times-on’.

EVENT: Disable *1*cancel-equal-times; name this event ‘g*1*cancel-equal-times-on’.

EVENT: Disable correctness-of-cancel-lessp-times; name this event ‘correctness-of-cancel-lessp-times-on’.

EVENT: Disable cancel-lessp-times; name this event ‘cancel-lessp-times-on’.

EVENT: Disable *1*cancel-lessp-times; name this event ‘g*1*cancel-lessp-times-on’.

EVENT: Disable infer-equality-from-not-lessp; name this event ‘infer-equality-from-not-lessp-on’.

EVENT: Disable lessp-times-arg1; name this event ‘lessp-times-arg1-on’.

EVENT: Disable eval\$-equal; name this event ‘eval\$-equal-on1’.

EVENT: Disable and-not-zerop-tree; name this event ‘and-not-zerop-tree-on’.

EVENT: Disable *1*and-not-zerop-tree; name this event ‘g*1*and-not-zerop-tree-on’.

EVENT: Disable or-zero-tree; name this event 'or-zero-tree-on'.

EVENT: Disable $*1*$ or-zero-tree; name this event ' $g*1*$ or-zero-tree-on'.

EVENT: Disable times-fringe; name this event 'times-fringe-on'.

EVENT: Disable $*1*$ times-fringe; name this event ' $g*1*$ times-fringe-on'.

EVENT: Disable times-tree; name this event 'times-tree-on'.

EVENT: Disable $*1*$ times-tree; name this event ' $g*1*$ times-tree-on'.

EVENT: Disable lessp-1-times; name this event 'lessp-1-times-on'.

EVENT: Disable lessp-plus-times2; name this event 'lessp-plus-times2-on'.

EVENT: Disable lessp-plus-times1; name this event 'lessp-plus-times1-on'.

EVENT: Disable lessp-plus-times-proof; name this event 'lessp-plus-times-proof-on'.

EVENT: Disable lessp-times-cancellation-proof; name this event 'lessp-times-cancellation-proof-on'.

EVENT: Disable lessp-times3; name this event 'lessp-times3-on'.

EVENT: Disable lessp-times3-proof2; name this event 'lessp-times3-proof2-on'.

EVENT: Disable lessp-times3-proof1; name this event 'lessp-times3-proof1-on'.

EVENT: Disable lessp-times2; name this event 'lessp-times2-on'.

EVENT: Disable lessp-times2-proof; name this event 'lessp-times2-proof-on'.

EVENT: Disable lessp-times1; name this event 'lessp-times1-on'.

EVENT: Disable lessp-times1-proof; name this event 'lessp-times1-proof-on'.

EVENT: Disable times-1-arg1; name this event 'times-1-arg1-on'.

EVENT: Disable times-quotient; name this event 'times-quotient-on'.

EVENT: Disable times-quotient-proof; name this event 'times-quotient-proof-on'.

EVENT: Disable times-distributes-over-difference; name this event 'times-distributes-over-difference-on'.

EVENT: Disable times-distributes-over-difference-proof; name this event 'times-distributes-over-difference-proof-on'.

EVENT: Disable associativity-of-times; name this event 'associativity-of-times-on'.

EVENT: Disable commutativity2-of-times; name this event 'commutativity2-of-times-on'.

EVENT: Disable times-distributes-over-plus; name this event 'times-distributes-over-plus-on'.

EVENT: Disable times-distributes-over-plus-proof; name this event 'times-distributes-over-plus-proof-on'.

EVENT: Disable commutativity-of-times; name this event 'commutativity-of-times-on'.

EVENT: Disable times-add1; name this event 'times-add1-on'.

EVENT: Disable times-zero; name this event 'times-zero-on'.

EVENT: Disable equal-sub1-0; name this event 'equal-sub1-0-on'.

EVENT: Disable equal-times-1; name this event 'equal-times-1-on'.

EVENT: Disable equal-times-0; name this event ‘equal-times-0-on’.

EVENT: Disable correctness-of-cancel-lessp-plus; name this event ‘correctness-of-cancel-lessp-plus-on’.

EVENT: Disable cancel-lessp-plus; name this event ‘cancel-lessp-plus-on’.

EVENT: Disable $*1*$ cancel-lessp-plus; name this event ‘ $g*1*$ cancel-lessp-plus-on’.

EVENT: Disable difference-x-x; name this event ‘difference-x-x-on’.

EVENT: Disable difference-difference-arg2; name this event ‘difference-difference-arg2-on’.

EVENT: Disable difference-difference-arg1; name this event ‘difference-difference-arg1-on’.

EVENT: Disable difference-sub1-arg2; name this event ‘difference-sub1-arg2-on’.

EVENT: Disable difference-add1-arg2; name this event ‘difference-add1-arg2-on’.

EVENT: Disable difference-leq-arg1; name this event ‘difference-leq-arg1-on’.

EVENT: Disable difference-elim; name this event ‘difference-elim-on’.

EVENT: Disable correctness-of-cancel-difference-plus; name this event ‘correctness-of-cancel-difference-plus-on’.

EVENT: Disable cancel-difference-plus; name this event ‘cancel-difference-plus-on’.

EVENT: Disable $*1*$ cancel-difference-plus; name this event ‘ $g*1*$ cancel-difference-plus-on’.

EVENT: Disable correctness-of-cancel-equal-plus; name this event ‘correctness-

of-cancel-equal-plus-on’.

EVENT: Disable cancel-equal-plus; name this event ‘cancel-equal-plus-on’.

EVENT: Disable $*1*$ cancel-equal-plus; name this event ‘ $g*1*$ cancel-equal-plus-on’.

EVENT: Disable eval\$-quote; name this event ‘eval\$-quote-on1’.

EVENT: Disable plus-tree; name this event ‘plus-tree-on’.

EVENT: Disable $*1*$ plus-tree; name this event ‘ $g*1*$ plus-tree-on’.

EVENT: Disable plus-fringe; name this event ‘plus-fringe-on’.

EVENT: Disable $*1*$ plus-fringe; name this event ‘ $g*1*$ plus-fringe-on’.

EVENT: Disable diff-diff-arg2; name this event ‘diff-diff-arg2-on’.

EVENT: Disable diff-diff-arg1; name this event ‘diff-diff-arg1-on’.

EVENT: Disable difference-plus-plus-cancellation-proof; name this event ‘difference-plus-plus-cancellation-proof-on’.

EVENT: Disable difference-plus-cancellation-proof; name this event ‘difference-plus-cancellation-proof-on’.

EVENT: Disable plus-difference-arg2; name this event ‘plus-difference-arg2-on’.

EVENT: Disable plus-difference-arg1; name this event ‘plus-difference-arg1-on’.

EVENT: Disable associativity-of-plus; name this event ‘associativity-of-plus-on’.

EVENT: Disable plus-add1-arg2; name this event ‘plus-add1-arg2-on’.

EVENT: Disable plus-add1-arg1; name this event ‘plus-add1-arg1-on’.

EVENT: Disable plus-zero-arg2; name this event 'plus-zero-arg2-on'.

EVENT: Disable commutativity2-of-plus; name this event 'commutativity2-of-plus-on'.

EVENT: Disable commutativity-of-plus; name this event 'commutativity-of-plus-on'.

EVENT: Disable equal-difference-0; name this event 'equal-difference-0-on'.

EVENT: Disable equal-plus-0; name this event 'equal-plus-0-on'.

EVENT: Disable member-bagint; name this event 'member-bagint-on'.

EVENT: Disable occurrences-bagdiff; name this event 'occurrences-bagdiff-on'.

EVENT: Disable occurrences-bagint; name this event 'occurrences-bagint-on'.

EVENT: Disable subbagp-bagint2; name this event 'subbagp-bagint2-on'.

EVENT: Disable subbagp-bagint1; name this event 'subbagp-bagint1-on'.

EVENT: Disable subbagp-cdr2; name this event 'subbagp-cdr2-on'.

EVENT: Disable subbagp-cdr1; name this event 'subbagp-cdr1-on'.

EVENT: Disable subbagp-delete; name this event 'subbagp-delete-on'.

EVENT: Disable bagdiff-delete; name this event 'bagdiff-delete-on'.

EVENT: Disable member-bagdiff; name this event 'member-bagdiff-on'.

EVENT: Disable occurrences-delete; name this event 'occurrences-delete-on'.

EVENT: Disable member-delete-implies-membership; name this event 'member-delete-implies-membership-on'.

EVENT: Disable member-delete; name this event ‘member-delete-on’.

EVENT: Disable member-non-list; name this event ‘member-non-list-on’.

EVENT: Disable equal-occurrences-zero; name this event ‘equal-occurrences-zero-on’.

EVENT: Disable delete-delete; name this event ‘delete-delete-on’.

EVENT: Disable delete-non-member; name this event ‘delete-non-member-on’.

EVENT: Disable subbagp; name this event ‘subbagp-on’.

EVENT: Disable *1*subbagp; name this event ‘g*1*subbagp-on’.

EVENT: Disable occurrences; name this event ‘occurrences-on’.

EVENT: Disable *1*occurrences; name this event ‘g*1*occurrences-on’.

EVENT: Disable bagint; name this event ‘bagint-on’.

EVENT: Disable *1*bagint; name this event ‘g*1*bagint-on’.

EVENT: Disable bagdiff; name this event ‘bagdiff-on’.

EVENT: Disable *1*bagdiff; name this event ‘g*1*bagdiff-on’.

EVENT: Disable delete; name this event ‘delete-on’.

EVENT: Disable *1*delete; name this event ‘g*1*delete-on’.

DEFINITION:

$\text{b-to-nat}(c)$

= **if** c **then** 1
 else 0 **endif**

EVENT: Enable v-adder-output=v-sum; name this event ‘v-adder-output=v-sum-off’.

EVENT: Enable v-adder-carry-out=v-carry; name this event ‘v-adder-carry-out=v-carry-off’.

THEOREM: bvp-cdr
 $(\text{bvp}(x) \wedge \text{listp}(x)) \rightarrow \text{bvp}(\text{cdr}(x))$

EVENT: Let us define the theory *integer-metas* to consist of the following events: correctness-of-cancel-ineg, correctness-of-cancel-iplus, correctness-of-cancel-iplus-ilessp, correctness-of-cancel-itimes, correctness-of-cancel-itimes-ilessp, correctness-of-cancel-itimes-factors, correctness-of-cancel-itimes-ilessp-factors, correctness-of-cancel-factors-0, correctness-of-cancel-factors-ilessp-0, correctness-of-cancel-ineg-terms-from-equality, correctness-of-cancel-ineg-terms-from-inequality.

EVENT: Enable correctness-of-cancel-ineg; name this event ‘correctness-of-cancel-ineg-off1’.

EVENT: Enable correctness-of-cancel-iplus; name this event ‘correctness-of-cancel-iplus-off1’.

EVENT: Enable correctness-of-cancel-iplus-ilessp; name this event ‘correctness-of-cancel-iplus-ilessp-off1’.

EVENT: Enable correctness-of-cancel-itimes; name this event ‘correctness-of-cancel-itimes-off1’.

EVENT: Enable correctness-of-cancel-itimes-ilessp; name this event ‘correctness-of-cancel-itimes-ilessp-off1’.

EVENT: Enable correctness-of-cancel-itimes-factors; name this event ‘correctness-of-cancel-itimes-factors-off1’.

EVENT: Enable correctness-of-cancel-itimes-ilessp-factors; name this event ‘correctness-of-cancel-itimes-ilessp-factors-off1’.

EVENT: Enable correctness-of-cancel-factors-0; name this event ‘correctness-of-cancel-factors-0-off1’.

EVENT: Enable correctness-of-cancel-factors-ilessp-0; name this event ‘correctness-of-cancel-factors-ilessp-0-off1’.

EVENT: Enable correctness-of-cancel-ineg-terms-from-equality; name this event ‘correctness-of-cancel-ineg-terms-from-equality-off1’.

EVENT: Enable correctness-of-cancel-ineg-terms-from-inequality; name this event ‘correctness-of-cancel-ineg-terms-from-inequality-off1’.

EVENT: Let us define the theory *naturals-metas* to consist of the following events: correctness-of-cancel-equal-plus, correctness-of-cancel-difference-plus, correctness-of-cancel-lessp-plus, correctness-of-cancel-lessp-times, correctness-of-cancel-equal-times, correctness-of-cancel-quotient-times.

EVENT: Enable correctness-of-cancel-equal-plus; name this event ‘correctness-of-cancel-equal-plus-off1’.

EVENT: Enable correctness-of-cancel-difference-plus; name this event ‘correctness-of-cancel-difference-plus-off1’.

EVENT: Enable correctness-of-cancel-lessp-plus; name this event ‘correctness-of-cancel-lessp-plus-off1’.

EVENT: Enable correctness-of-cancel-lessp-times; name this event ‘correctness-of-cancel-lessp-times-off1’.

EVENT: Enable correctness-of-cancel-equal-times; name this event ‘correctness-of-cancel-equal-times-off1’.

EVENT: Enable correctness-of-cancel-quotient-times; name this event ‘correctness-of-cancel-quotient-times-off1’.

DEFINITION: $\text{v-alu-nat-int-buf}(a) = \text{cvzbv}(\mathbf{f}, \mathbf{f}, a)$

THEOREM: $\text{v-alu-correct-nat-int-buf}$
 $\text{bvp}(a) \rightarrow (\text{cvzbv}(\mathbf{f}, \mathbf{f}, \text{v-buf}(a)) = \text{v-alu-nat-int-buf}(a))$

THEOREM: nth-length-v
 $(\text{bvp}(bv) \wedge (\text{length}(bv) = (1 + n)))$
 $\rightarrow (\text{nth}(n, bv) = (\text{v-to-nat}(bv) \not\prec \exp(2, n)))$

THEOREM: remainder-plus-multiple

$$((z1 < y1) \wedge ((a \mathbf{mod} y1) = 0)) \rightarrow (((z1 + a) \mathbf{mod} y1) = \text{fix}(z1))$$

THEOREM: divides-plus-plus

$$((x \mathbf{mod} y) = 0) \rightarrow (((x + x) \mathbf{mod} (y + y)) = 0)$$

THEOREM: v-to-nat-firstn-hack1

$$(z < y) \\ \rightarrow (((z + z + (w * y) + (w * y)) \mathbf{mod} (y + y)) = (z + z))$$

THEOREM: remainder-add1-plus-multiple

$$(((1 + z1) < y1) \wedge ((a \mathbf{mod} y1) = 0)) \\ \rightarrow (((1 + (z1 + a)) \mathbf{mod} y1) = (1 + z1))$$

THEOREM: v-to-nat-firstn-hack2

$$(z < y) \\ \rightarrow (((1 + (z + z + (d * y) + (d * y))) \mathbf{mod} (y + y)) \\ = (1 + (z + z)))$$

THEOREM: v-to-nat-firstn

$$\text{v-to-nat}(\text{firstn}(n, v)) = (\text{v-to-nat}(v) \mathbf{mod} \exp(2, n))$$

EVENT: Enable v-to-nat-firstn-hack1; name this event ‘v-to-nat-firstn-hack1-off’.

EVENT: Enable v-to-nat-firstn-hack2; name this event ‘v-to-nat-firstn-hack2-off’.

THEOREM: v-to-nat-of-nat-to-v-hack

$$(v < \exp(2, x)) \\ \rightarrow (((1 + (v + v + (w * \exp(2, x)) + (w * \exp(2, x)))) \\ \mathbf{mod} (\exp(2, x) + \exp(2, x))) \\ = (1 + (v + v)))$$

THEOREM: v-to-nat-of-nat-to-v

$$\text{v-to-nat}(\text{nat-to-v}(n, \text{len})) = (n \mathbf{mod} \exp(2, \text{len}))$$

THEOREM: remainder-plus-x-x-2

$$((x + x) \mathbf{mod} 2) = 0$$

THEOREM: quotient-plus-x-x-2

$$((x + x) \div 2) = \text{fix}(x)$$

THEOREM: nat-to-v-plus-x-x

$$\text{nat-to-v}(x + x, 1 + n) = \text{cons}(\mathbf{f}, \text{nat-to-v}(x, n))$$

THEOREM: firstn-add1-cons

$$\text{firstn}(1 + n, \text{cons}(a, x)) = \text{cons}(a, \text{firstn}(n, x))$$

THEOREM: firstn-zero

$$(n \simeq 0) \rightarrow (\text{firstn}(n, x) = \mathbf{nil})$$

THEOREM: firstn-nlistp

$$(x \simeq \mathbf{nil}) \rightarrow (\text{firstn}(n, x) = \mathbf{nil})$$

THEOREM: nat-to-v-of-v-to-nat-general

$$\begin{aligned} &(\text{bvp}(v) \wedge (\text{length}(v) \not\leq n)) \\ &\rightarrow (\text{nat-to-v}(\text{v-to-nat}(v), n) = \text{firstn}(n, v)) \end{aligned}$$

THEOREM: firstn-length

$$\text{properp}(x) \rightarrow (\text{firstn}(\text{length}(x), x) = x)$$

THEOREM: nat-to-v-of-v-to-nat

$$\text{bvp}(v) \rightarrow (\text{nat-to-v}(\text{v-to-nat}(v), \text{length}(v)) = v)$$

THEOREM: nat-to-v-zero

$$(len \simeq 0) \rightarrow (\text{nat-to-v}(n, len) = \mathbf{nil})$$

THEOREM: equal-nat-to-v-inverter-hack1

$$\begin{aligned} &((len \in \mathbf{N}) \wedge (v = \text{nat-to-v}(n, len))) \\ &\rightarrow ((\text{v-to-nat}(v) = (n \mathbf{mod} \exp(2, len))) = \mathbf{t}) \end{aligned}$$

THEOREM: equal-nat-to-v-inverter-hack2-lemma

$$\begin{aligned} &(\text{bvp}(v) \wedge (\text{v-to-nat}(v) = (n \mathbf{mod} \exp(2, len)))) \\ &\rightarrow ((\text{nat-to-v}(\text{v-to-nat}(v), len) = \text{nat-to-v}(n \mathbf{mod} \exp(2, len), len)) \\ &\quad = \mathbf{t}) \end{aligned}$$

THEOREM: nat-to-v-remainder

$$\text{nat-to-v}(n \mathbf{mod} \exp(2, len), len) = \text{nat-to-v}(n, len)$$

THEOREM: equal-nat-to-v-inverter-hack2

$$\begin{aligned} &(\text{bvp}(v) \wedge (\text{v-to-nat}(v) = (n \mathbf{mod} \exp(2, \text{length}(v))))) \\ &\rightarrow ((v = \text{nat-to-v}(n, \text{length}(v))) = \mathbf{t}) \end{aligned}$$

THEOREM: equal-nat-to-v-inverter

$$\begin{aligned} &(v = \text{nat-to-v}(n, len)) \\ \leftrightarrow &(\text{bvp}(v) \\ &\quad \wedge (\text{length}(v) = \text{fix}(len)) \\ &\quad \wedge (\text{v-to-nat}(v) = (n \mathbf{mod} \exp(2, len)))) \end{aligned}$$

EVENT: Enable equal-nat-to-v-inverter-hack1; name this event ‘equal-nat-to-v-inverter-hack1-off’.

EVENT: Enable equal-nat-to-v-inverter-hack2; name this event ‘equal-nat-to-v-inverter-hack2-off’.

THEOREM: v-to-nat-v-not-lemma

$$\begin{aligned} & \text{bvp}(a) \\ \rightarrow & ((\text{v-to-nat}(a) + \text{v-to-nat}(\text{v-not}(a))) = (\exp(2, \text{length}(a)) - 1)) \end{aligned}$$

THEOREM: v-to-nat-v-not

$$\begin{aligned} & \text{bvp}(a) \\ \rightarrow & (\text{v-to-nat}(\text{v-not}(a)) = ((\exp(2, \text{length}(a)) - \text{v-to-nat}(a)) - 1)) \end{aligned}$$

THEOREM: lessp-v-to-nat-exp

$$\text{bvp}(a) \rightarrow (\text{v-to-nat}(a) < \exp(2, \text{length}(a)))$$

THEOREM: equal-iff

$$(\text{boolp}(x) \wedge \text{boolp}(y)) \rightarrow ((x = y) = (x \leftrightarrow y))$$

EVENT: Enable equal-iff; name this event ‘equal-iff-off’.

THEOREM: v-alu-correct-nat-subtractor-output-lemma

$$\begin{aligned} & ((a\text{-nat} < bound) \wedge (b\text{-nat} < bound)) \\ \rightarrow & ((b\text{-nat} + ((bound - a\text{-nat}) - 1)) \\ & = ((b\text{-nat} + bound) - a\text{-nat} - 1)) \end{aligned}$$

THEOREM: lessp-v-to-nat-exp-rewrite

$$(\text{bvp}(a) \wedge (\text{length}(a) = len)) \rightarrow ((\text{v-to-nat}(a) < \exp(2, len)) = \mathbf{t})$$

THEOREM: not-lessp-quotient

$$n \not< (n \div k)$$

THEOREM: v-to-nat-append

$$\begin{aligned} & (\text{bvp}(a) \wedge \text{bvp}(b)) \\ \rightarrow & (\text{v-to-nat}(\text{append}(a, b)) \\ & = (\text{v-to-nat}(a) + (\exp(2, \text{length}(a)) * \text{v-to-nat}(b)))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-adder-output}(c, a, b) \\ = & \text{nat-to-v}((\text{b-to-nat}(c) + \text{v-to-nat}(a) + \text{v-to-nat}(b)) \\ & \quad \mathbf{mod} \exp(2, \text{length}(a)), \\ & \quad \text{length}(a)) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-adder-carry-out}(c, a, b) \\ = & ((\text{b-to-nat}(c) + \text{v-to-nat}(a) + \text{v-to-nat}(b)) \not< \exp(2, \text{length}(a))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-adder}(c, a, b) \\ = & \text{cvzbv}(\text{v-alu-nat-adder-carry-out}(c, a, b), \\ & \quad \text{v-adder-overflowp}(c, a, b), \\ & \quad \text{v-alu-nat-adder-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-nat-adder-carry-out

$$\begin{aligned} & \text{bv2p}(a, b) \\ & \rightarrow (\text{v-adder-carry-out}(c, a, b) = \text{v-alu-nat-adder-carry-out}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-nat-adder-output

$$\text{bv2p}(a, b) \rightarrow (\text{v-adder-output}(c, a, b) = \text{v-alu-nat-adder-output}(c, a, b))$$

THEOREM: v-alu-correct-nat-adder

$$\text{bv2p}(a, b) \rightarrow (\text{cvzbv-v-adder}(c, a, b) = \text{v-alu-nat-adder}(c, a, b))$$

DEFINITION:

$$\text{v-alu-nat-inc}(a) = \text{v-alu-nat-adder}(\mathbf{t}, a, \text{nat-to-v}(0, \text{length}(a)))$$

THEOREM: v-alu-correct-nat-inc

$$\text{bvp}(a) \rightarrow (\text{cvzbv-inc}(a) = \text{v-alu-nat-inc}(a))$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-subtractor-output}(c, a, b) \\ & = \text{nat-to-v}(((\text{v-to-nat}(b) + \text{exp}(2, \text{length}(a))) \\ & \quad - (\text{v-to-nat}(a) + \text{b-to-nat}(c))) \\ & \quad \mathbf{mod} \text{exp}(2, \text{length}(a)), \\ & \quad \text{length}(a)) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-subtractor-carry-out}(c, a, b) \\ & = (\text{v-to-nat}(b) < (\text{v-to-nat}(a) + \text{b-to-nat}(c))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-nat-subtractor}(c, a, b) \\ & = \text{cvzbv}(\text{v-alu-nat-subtractor-carry-out}(c, a, b), \\ & \quad \text{v-subtractor-overflowp}(c, a, b), \\ & \quad \text{v-alu-nat-subtractor-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-nat-subtractor-carry-out

$$\begin{aligned} & \text{bv2p}(a, b) \\ & \rightarrow (\text{v-subtractor-carry-out}(c, a, b) \\ & \quad = \text{v-alu-nat-subtractor-carry-out}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-nat-subtractor-output

$$\begin{aligned} & \text{bv2p}(a, b) \\ & \rightarrow (\text{v-subtractor-output}(c, a, b) = \text{v-alu-nat-subtractor-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-nat-subtractor

$$\text{bv2p}(a, b) \rightarrow (\text{cvzbv-v-subtractor}(c, a, b) = \text{v-alu-nat-subtractor}(c, a, b))$$

DEFINITION:

$$\text{v-alu-nat-dec}(a) = \text{v-alu-nat-subtractor}(\mathbf{t}, \text{nat-to-v}(0, \text{length}(a)), a)$$

THEOREM: v-alu-correct-nat-dec
 $\text{bvp}(a) \rightarrow (\text{cvzbv-dec}(a) = \text{v-alu-nat-dec}(a))$

DEFINITION:
 $\text{v-alu-nat-lsr-output}(a) = \text{nat-to-v}(\text{v-to-nat}(a) \div 2, \text{length}(a))$

DEFINITION:
 $\text{v-alu-nat-lsr-carry-out}(a) = ((\text{v-to-nat}(a) \bmod 2) \neq 0)$

DEFINITION:
 $\text{v-alu-nat-lsr}(a)$
 $= \text{cvzbv}(\text{v-alu-nat-lsr-carry-out}(a), \mathbf{f}, \text{v-alu-nat-lsr-output}(a))$

THEOREM: v-alu-correct-nat-lsr-carry-out
 $(\text{bvp}(a) \wedge \text{listp}(a)) \rightarrow (\text{nth}(0, a) = \text{v-alu-nat-lsr-carry-out}(a))$

THEOREM: v-alu-correct-nat-lsr-output
 $\text{bvp}(a) \rightarrow (\text{v-lsr}(a) = \text{v-alu-nat-lsr-output}(a))$

THEOREM: v-alu-correct-nat-lsr
 $\text{bvp}(a) \rightarrow (\text{cvzbv-v-lsr}(a) = \text{v-alu-nat-lsr}(a))$

DEFINITION:
 $\text{v-alu-nat-not-output}(a)$
 $= \text{nat-to-v}((\text{exp}(2, \text{length}(a)) - \text{v-to-nat}(a)) - 1, \text{length}(a))$

DEFINITION:
 $\text{v-alu-nat-not}(a) = \text{cvzbv}(\mathbf{f}, \mathbf{f}, \text{v-alu-nat-not-output}(a))$

THEOREM: v-alu-correct-nat-not-output
 $\text{bvp}(a) \rightarrow (\text{v-not}(a) = \text{v-alu-nat-not-output}(a))$

THEOREM: v-alu-correct-nat-not
 $\text{bvp}(a) \rightarrow (\text{cvzbv-v-not}(a) = \text{v-alu-nat-not}(a))$

DEFINITION:
 $\text{v-alu-nat}(c, a, b, op)$
 $=$ **if** $op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-nat-int-buf}(a)$
 elseif $op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-nat-inc}(a)$
 elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-nat-adder}(c, a, b)$
 elseif $op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-nat-adder}(\mathbf{f}, a, b)$
 elseif $op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-nat-dec}(a)$
 elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-nat-subtractor}(c, a, b)$
 elseif $op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-nat-subtractor}(\mathbf{f}, a, b)$
 elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})$ **then** $\text{v-alu-nat-lsr}(a)$
 elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$ **then** $\text{v-alu-nat-not}(a)$
 else $\text{v-alu}(c, a, b, op)$ **endif**

THEOREM: v-alu-correct-nat

$$\text{bv2p}(a, b) \rightarrow (\text{v-alu}(c, a, b, \text{op}) = \text{v-alu-nat}(c, a, b, \text{op}))$$

DEFINITION:

int-to-v (i, l)

$$= \text{if } \text{ilessp}(i, 0) \text{ then nat-to-v}(\text{iplus}(i, \text{exp}(2, l)), l) \\ \text{else nat-to-v}(i, l) \text{ endif}$$

DEFINITION:

v-to-int (v)

$$= \text{if } \text{v-to-nat}(v) < \text{exp}(2, \text{length}(v) - 1) \text{ then v-to-nat}(v) \\ \text{else idifference}(\text{v-to-nat}(v), \text{exp}(2, \text{length}(v))) \text{ endif}$$

DEFINITION:

integer-in-range (i, l)

$$= (\text{ileq}(\text{ineg}(\text{exp}(2, l - 1)), i) \wedge \text{ilessp}(i, \text{exp}(2, l - 1)))$$

THEOREM: iplus-plus

$$((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \rightarrow (\text{iplus}(x, y) = (x + y))$$

THEOREM: illessp-lessp

$$((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \rightarrow (\text{illessp}(x, y) = (x < y))$$

THEOREM: times-2

$$(2 * x) = (x + x)$$

THEOREM: b-to-nat-leq

$$1 \not\leq \text{b-to-nat}(x)$$

THEOREM: lessp-v-to-nat-exp-with-exp-opened

$$(\text{bvp}(a) \wedge (\text{length}(a) = \text{length}(b)) \wedge (\text{length}(b) \neq 0)) \\ \rightarrow (\text{v-to-nat}(a) < (\text{exp}(2, \text{length}(b) - 1) + \text{exp}(2, \text{length}(b) - 1)))$$

THEOREM: fix-int-numberp

$$(x \in \mathbf{N}) \rightarrow (\text{fix-int}(x) = x)$$

THEOREM: iplus3-plus-difference

$$((c\theta \in \mathbf{N}) \\ \wedge (a\theta \in \mathbf{N}) \\ \wedge (b\theta \in \mathbf{N}) \\ \wedge (d \in \mathbf{N}) \\ \wedge ((c\theta + a\theta + b\theta) \not\leq d)) \\ \rightarrow (\text{iplus}(c\theta, \text{iplus}(a\theta, \text{iplus}(b\theta, \text{ineg}(d)))) = ((c\theta + a\theta + b\theta) - d))$$

THEOREM: v-to-int-v-not

$$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow (\text{v-to-int}(\text{v-not}(a)) = \text{iplus}(-1, \text{ineg}(\text{v-to-int}(a))))$$

EVENT: Enable v-alu-correct-nat-not-output; name this event ‘v-alu-correct-nat-not-output-off’.

THEOREM: v-alu-correct-int-subtractor-overflowp-lemma
 $(bv2p(a, b) \wedge (length(a) \neq 0))$
 $\rightarrow (iplus(b\text{-to-nat}(b\text{-not}(c)), iplus(v\text{-to-int}(v\text{-not}(a)), v\text{-to-int}(b)))$
 $= idifference(v\text{-to-int}(b), iplus(v\text{-to-int}(a), b\text{-to-nat}(c))))$

THEOREM: idifference-difference
 $((x \in \mathbf{N}) \wedge (y \in \mathbf{N}) \wedge (x \not< y))$
 $\rightarrow ((iplus(x, ineg(y)) = (x - y))$
 $\wedge (iplus(ineg(y), x) = (x - y)))$

THEOREM: idiv-quotient
 $((x \in \mathbf{N}) \wedge (y \in \mathbf{N})) \rightarrow (idiv(x, y) = (x \div y))$

THEOREM: idiv-ilessp-0
 $((y \in \mathbf{N}) \wedge (y \neq 0)) \rightarrow (ilessp(idiv(x, y), 0) = illessp(x, 0))$

THEOREM: quotient-plus-x-x-2-with-extra-arg
 $((x + x + y) \div 2) = (x + (y \div 2))$

THEOREM: plus-quotient-2-lemma
 $((v + (2 * z)) \div 2) = (z + (v \div 2))$

THEOREM: plus-quotient-2
 $(a0 \not< exp) \rightarrow ((exp + ((a0 - exp) \div 2)) = ((a0 + exp) \div 2))$

THEOREM: iplus-div-hack1
 $((a0 \in \mathbf{N}) \wedge (exp \in \mathbf{N}))$
 $\rightarrow (iplus(idiv(iplus(a0, ineg(exp)), 2), exp) = ((a0 + exp) \div 2))$

EVENT: Enable quotient-plus-x-x-2-with-extra-arg; name this event ‘quotient-plus-x-x-2-with-extra-arg-off’.

EVENT: Enable plus-quotient-2-lemma; name this event ‘plus-quotient-2-lemma-off’.

EVENT: Enable plus-quotient-2; name this event ‘plus-quotient-2-off’.

THEOREM: lessp-quotient-2
 $((x \div 2) < y) \leftrightarrow (x < (y + y))$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-adder-output}(c, a, b) \\ = & \text{int-to-v}(\text{iplus}(\text{b-to-nat}(c), \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))), \text{length}(a)) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-adder-overflowp}(c, a, b) \\ = & (\neg \text{integer-in-range}(\text{iplus}(\text{b-to-nat}(c), \\ & \qquad \qquad \qquad \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))), \\ & \qquad \qquad \qquad \text{length}(a))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-adder}(c, a, b) \\ = & \text{cvzbv}(\text{v-adder-carry-out}(c, a, b), \\ & \qquad \text{v-alu-int-adder-overflowp}(c, a, b), \\ & \qquad \text{v-alu-int-adder-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-adder-overflowp

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{v-adder-overflowp}(c, a, b) = \text{v-alu-int-adder-overflowp}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-adder-output

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{v-adder-output}(c, a, b) = \text{v-alu-int-adder-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-adder

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{cvzbv-v-adder}(c, a, b) = \text{v-alu-int-adder}(c, a, b)) \end{aligned}$$

DEFINITION:

$$\text{v-alu-int-inc}(a) = \text{v-alu-int-adder}(\mathbf{t}, a, \text{int-to-v}(0, \text{length}(a)))$$

THEOREM: v-alu-correct-int-inc

$$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{cvzbv-inc}(a) = \text{v-alu-int-inc}(a))$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-subtractor-output}(c, a, b) \\ = & \text{int-to-v}(\text{idifference}(\text{v-to-int}(b), \text{iplus}(\text{v-to-int}(a), \text{b-to-nat}(c))), \\ & \qquad \qquad \qquad \text{length}(a)) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-subtractor-overflowp}(c, a, b) \\ = & (\neg \text{integer-in-range}(\text{idifference}(\text{v-to-int}(b), \\ & \qquad \qquad \qquad \text{iplus}(\text{v-to-int}(a), \text{b-to-nat}(c))), \\ & \qquad \qquad \qquad \text{length}(a))) \end{aligned}$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-subtractor}(c, a, b) \\ = & \text{cvzbv}(\text{v-subtractor-carry-out}(c, a, b), \\ & \text{v-alu-int-subtractor-overflowp}(c, a, b), \\ & \text{v-alu-int-subtractor-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-subtractor-overflowp

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{v-subtractor-overflowp}(c, a, b) \\ = & \text{v-alu-int-subtractor-overflowp}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-subtractor-output

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{v-subtractor-output}(c, a, b) = \text{v-alu-int-subtractor-output}(c, a, b)) \end{aligned}$$

THEOREM: v-alu-correct-int-subtractor

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0)) \\ \rightarrow & (\text{cvzbv-v-subtractor}(c, a, b) = \text{v-alu-int-subtractor}(c, a, b)) \end{aligned}$$

DEFINITION:

$$\text{v-alu-int-dec}(a) = \text{v-alu-int-subtractor}(\mathbf{t}, \text{int-to-v}(0, \text{length}(a)), a)$$

THEOREM: v-alu-correct-int-dec

$$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{cvzbv-dec}(a) = \text{v-alu-int-dec}(a))$$

DEFINITION:

$$\text{v-alu-int-asr-output}(a) = \text{int-to-v}(\text{idiv}(\text{v-to-int}(a), 2), \text{length}(a))$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-asr}(a) \\ = & \text{cvzbv}(\text{v-alu-nat-lsr-carry-out}(a), \mathbf{f}, \text{v-alu-int-asr-output}(a)) \end{aligned}$$

THEOREM: v-alu-correct-int-asr-output

$$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{v-asr}(a) = \text{v-alu-int-asr-output}(a))$$

THEOREM: v-alu-correct-int-asr

$$\text{bvp}(a) \rightarrow (\text{cvzbv-v-asr}(a) = \text{v-alu-int-asr}(a))$$

DEFINITION:

$$\text{v-alu-int-neg}(a) = \text{v-alu-int-subtractor}(\mathbf{f}, a, \text{int-to-v}(0, \text{length}(a)))$$

THEOREM: v-alu-correct-int-neg

$$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{cvzbv-neg}(a) = \text{v-alu-int-neg}(a))$$

DEFINITION:

$$\begin{aligned} & \text{v-alu-int-not-output}(a) \\ = & \text{v-alu-int-subtractor-output}(\mathbf{t}, a, \text{int-to-v}(0, \text{length}(a))) \end{aligned}$$

DEFINITION:

$\text{v-alu-int-not}(a) = \text{cvzbv}(\mathbf{f}, \mathbf{f}, \text{v-alu-int-not-output}(a))$

THEOREM: v-alu-correct-int-not-output

$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{v-not}(a) = \text{v-alu-int-not-output}(a))$

THEOREM: v-alu-correct-int-not

$(\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \rightarrow (\text{cvzbv-v-not}(a) = \text{v-alu-int-not}(a))$

DEFINITION:

$\text{v-alu-int}(c, a, b, op)$

= **if** $op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-nat-int-buf}(a)$
elseif $op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-int-inc}(a)$
elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-int-adder}(c, a, b)$
elseif $op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$ **then** $\text{v-alu-int-adder}(\mathbf{f}, a, b)$
elseif $op = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-int-neg}(a)$
elseif $op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-int-dec}(a)$
elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-int-subtracter}(c, a, b)$
elseif $op = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})$ **then** $\text{v-alu-int-subtracter}(\mathbf{f}, a, b)$
elseif $op = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})$ **then** $\text{v-alu-int-asr}(a)$
elseif $op = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$ **then** $\text{v-alu-int-not}(a)$
else $\text{v-alu}(c, a, b, op)$ **endif**

THEOREM: v-alu-correct-int

$(\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0))$
 $\rightarrow (\text{v-alu}(c, a, b, op) = \text{v-alu-int}(c, a, b, op))$

EVENT: Enable v-alu-correct-nat-int-buf; name this event ‘v-alu-correct-nat-int-buf-off’.

EVENT: Enable v-alu-correct-nat-adder-carry-out; name this event ‘v-alu-correct-nat-adder-carry-out-off’.

EVENT: Enable v-alu-correct-nat-adder-output; name this event ‘v-alu-correct-nat-adder-output-off’.

EVENT: Enable v-alu-correct-nat-adder; name this event ‘v-alu-correct-nat-adder-off’.

EVENT: Enable v-alu-correct-nat-inc; name this event ‘v-alu-correct-nat-inc-off’.

EVENT: Enable v-alu-correct-nat-subtracter-carry-out; name this event ‘v-alu-correct-nat-subtracter-carry-out-off’.

EVENT: Enable v-alu-correct-nat-subtractor-output; name this event ‘v-alu-correct-nat-subtractor-output-off’.

EVENT: Enable v-alu-correct-nat-subtractor; name this event ‘v-alu-correct-nat-subtractor-off’.

EVENT: Enable v-alu-correct-nat-dec; name this event ‘v-alu-correct-nat-dec-off’.

EVENT: Enable v-alu-correct-nat-lsr-carry-out; name this event ‘v-alu-correct-nat-lsr-carry-out-off’.

EVENT: Enable v-alu-correct-nat-lsr-output; name this event ‘v-alu-correct-nat-lsr-output-off’.

EVENT: Enable v-alu-correct-nat-lsr; name this event ‘v-alu-correct-nat-lsr-off’.

EVENT: Enable v-alu-correct-nat-not-output; name this event ‘v-alu-correct-nat-not-output-off1’.

EVENT: Enable v-alu-correct-nat-not; name this event ‘v-alu-correct-nat-not-off’.

EVENT: Enable v-alu-correct-int-adder-overflowp; name this event ‘v-alu-correct-int-adder-overflowp-off’.

EVENT: Enable v-alu-correct-int-adder-output; name this event ‘v-alu-correct-int-adder-output-off’.

EVENT: Enable v-alu-correct-int-adder; name this event ‘v-alu-correct-int-adder-off’.

EVENT: Enable v-alu-correct-int-inc; name this event ‘v-alu-correct-int-inc-off’.

EVENT: Enable v-alu-correct-int-subtractor-overflowp; name this event ‘v-alu-correct-int-subtractor-overflowp-off’.

EVENT: Enable v-alu-correct-int-subtractor-output; name this event 'v-alu-correct-int-subtractor-output-off'.

EVENT: Enable v-alu-correct-int-subtractor; name this event 'v-alu-correct-int-subtractor-off'.

EVENT: Enable v-alu-correct-int-dec; name this event 'v-alu-correct-int-dec-off'.

EVENT: Enable v-alu-correct-int-asr-output; name this event 'v-alu-correct-int-asr-output-off'.

EVENT: Enable v-alu-correct-int-asr; name this event 'v-alu-correct-int-asr-off'.

EVENT: Enable v-alu-correct-int-neg; name this event 'v-alu-correct-int-neg-off'.

EVENT: Enable v-alu-correct-int-not; name this event 'v-alu-correct-int-not-off'.

EVENT: Enable v-alu-correct-int; name this event 'v-alu-correct-int-off'.

EVENT: Enable v-alu-int; name this event 'v-alu-int-off'.

EVENT: Enable *1*v-alu-int; name this event 'g*1*v-alu-int-off'.

EVENT: Enable v-alu-correct-int-not-output; name this event 'v-alu-correct-int-not-output-off'.

EVENT: Enable v-alu-int-not; name this event 'v-alu-int-not-off'.

EVENT: Enable *1*v-alu-int-not; name this event 'g*1*v-alu-int-not-off'.

EVENT: Enable v-alu-int-not-output; name this event 'v-alu-int-not-output-off'.

EVENT: Enable *1*v-alu-int-not-output; name this event 'g*1*v-alu-int-not-output-off'.

EVENT: Enable v-alu-int-neg; name this event 'v-alu-int-neg-off'.

EVENT: Enable *1*v-alu-int-neg; name this event 'g*1*v-alu-int-neg-off'.

EVENT: Enable v-alu-int-asr; name this event 'v-alu-int-asr-off'.

EVENT: Enable *1*v-alu-int-asr; name this event 'g*1*v-alu-int-asr-off'.

EVENT: Enable v-alu-int-asr-output; name this event 'v-alu-int-asr-output-off'.

EVENT: Enable *1*v-alu-int-asr-output; name this event 'g*1*v-alu-int-asr-output-off'.

EVENT: Enable v-alu-int-dec; name this event 'v-alu-int-dec-off'.

EVENT: Enable *1*v-alu-int-dec; name this event 'g*1*v-alu-int-dec-off'.

EVENT: Enable v-alu-int-subtractor; name this event 'v-alu-int-subtractor-off'.

EVENT: Enable *1*v-alu-int-subtractor; name this event 'g*1*v-alu-int-subtractor-off'.

EVENT: Enable v-alu-int-subtractor-overflowp; name this event 'v-alu-int-subtractor-overflowp-off'.

EVENT: Enable *1*v-alu-int-subtractor-overflowp; name this event 'g*1*v-alu-int-subtractor-overflowp-off'.

EVENT: Enable v-alu-int-subtractor-output; name this event 'v-alu-int-subtractor-output-off'.

EVENT: Enable *1*v-alu-int-subtractor-output; name this event 'g*1*v-alu-int-subtractor-output-off'.

EVENT: Enable v-alu-int-inc; name this event 'v-alu-int-inc-off'.

EVENT: Enable *1*v-alu-int-inc; name this event 'g*1*v-alu-int-inc-off'.

EVENT: Enable v-alu-int-adder; name this event 'v-alu-int-adder-off'.

EVENT: Enable *1*v-alu-int-adder; name this event 'g*1*v-alu-int-adder-off'.

EVENT: Enable v-alu-int-adder-overflowp; name this event 'v-alu-int-adder-overflowp-off'.

EVENT: Enable *1*v-alu-int-adder-overflowp; name this event 'g*1*v-alu-int-adder-overflowp-off'.

EVENT: Enable v-alu-int-adder-output; name this event 'v-alu-int-adder-output-off'.

EVENT: Enable *1*v-alu-int-adder-output; name this event 'g*1*v-alu-int-adder-output-off'.

EVENT: Enable lessp-quotient-2; name this event 'lessp-quotient-2-off'.

EVENT: Enable iplus-div-hack1; name this event 'iplus-div-hack1-off'.

EVENT: Enable idiv-ilessp-0; name this event 'idiv-ilessp-0-off'.

EVENT: Enable idiv-quotient; name this event 'idiv-quotient-off'.

EVENT: Enable idifference-difference; name this event 'idifference-difference-off'.

EVENT: Enable v-alu-correct-int-subtracter-overflowp-lemma; name this event 'v-alu-correct-int-subtracter-overflowp-lemma-off'.

EVENT: Enable v-to-int-v-not; name this event 'v-to-int-v-not-off'.

EVENT: Enable iplus3-plus-difference; name this event 'iplus3-plus-difference-off'.

EVENT: Enable fix-int-numberp; name this event 'fix-int-numberp-off'.

EVENT: Enable lessp-v-to-nat-exp-with-exp-opened; name this event 'lessp-v-to-nat-exp-with-exp-opened-off'.

EVENT: Enable b-to-nat-leq; name this event 'b-to-nat-leq-off'.

EVENT: Enable times-2; name this event 'times-2-off'.

EVENT: Enable illessp-lessp; name this event 'illessp-lessp-off'.

EVENT: Enable iplus-plus; name this event 'iplus-plus-off'.

EVENT: Enable integer-in-range; name this event 'integer-in-range-off'.

EVENT: Enable *1*integer-in-range; name this event 'g*1*integer-in-range-off'.

EVENT: Enable v-to-int; name this event 'v-to-int-off'.

EVENT: Enable *1*v-to-int; name this event 'g*1*v-to-int-off'.

EVENT: Enable int-to-v; name this event 'int-to-v-off'.

EVENT: Enable *1*int-to-v; name this event 'g*1*int-to-v-off'.

EVENT: Enable v-alu-correct-nat; name this event 'v-alu-correct-nat-off'.

EVENT: Enable v-alu-nat; name this event 'v-alu-nat-off'.

EVENT: Enable *1*v-alu-nat; name this event 'g*1*v-alu-nat-off'.

EVENT: Enable v-alu-nat-not; name this event 'v-alu-nat-not-off'.

EVENT: Enable *1*v-alu-nat-not; name this event 'g*1*v-alu-nat-not-off'.

EVENT: Enable v-alu-nat-not-output; name this event 'v-alu-nat-not-output-off'.

EVENT: Enable *1*v-alu-nat-not-output; name this event 'g*1*v-alu-nat-not-output-off'.

EVENT: Enable v-alu-nat-lsr; name this event 'v-alu-nat-lsr-off'.

EVENT: Enable *1*v-alu-nat-lsr; name this event 'g*1*v-alu-nat-lsr-off'.

EVENT: Enable v-alu-nat-lsr-carry-out; name this event 'v-alu-nat-lsr-carry-out-off'.

EVENT: Enable *1*v-alu-nat-lsr-carry-out; name this event 'g*1*v-alu-nat-lsr-carry-out-off'.

EVENT: Enable v-alu-nat-lsr-output; name this event 'v-alu-nat-lsr-output-off'.

EVENT: Enable *1*v-alu-nat-lsr-output; name this event 'g*1*v-alu-nat-lsr-output-off'.

EVENT: Enable v-alu-nat-dec; name this event 'v-alu-nat-dec-off'.

EVENT: Enable *1*v-alu-nat-dec; name this event 'g*1*v-alu-nat-dec-off'.

EVENT: Enable v-alu-nat-subtractor; name this event 'v-alu-nat-subtractor-off'.

EVENT: Enable *1*v-alu-nat-subtractor; name this event 'g*1*v-alu-nat-subtractor-off'.

EVENT: Enable v-alu-nat-subtractor-carry-out; name this event 'v-alu-nat-subtractor-carry-out-off'.

EVENT: Enable *1*v-alu-nat-subtractor-carry-out; name this event 'g*1*v-alu-nat-subtractor-carry-out-off'.

EVENT: Enable v-alu-nat-subtractor-output; name this event 'v-alu-nat-subtractor-

output-off’.

EVENT: Enable $*1*v\text{-alu-nat-subtracter-output}$; name this event ‘ $g*1*v\text{-alu-nat-subtracter-output-off}$ ’.

EVENT: Enable $v\text{-alu-nat-inc}$; name this event ‘ $v\text{-alu-nat-inc-off}$ ’.

EVENT: Enable $*1*v\text{-alu-nat-inc}$; name this event ‘ $g*1*v\text{-alu-nat-inc-off}$ ’.

EVENT: Enable $v\text{-alu-nat-adder}$; name this event ‘ $v\text{-alu-nat-adder-off}$ ’.

EVENT: Enable $*1*v\text{-alu-nat-adder}$; name this event ‘ $g*1*v\text{-alu-nat-adder-off}$ ’.

EVENT: Enable $v\text{-alu-nat-adder-carry-out}$; name this event ‘ $v\text{-alu-nat-adder-carry-out-off}$ ’.

EVENT: Enable $*1*v\text{-alu-nat-adder-carry-out}$; name this event ‘ $g*1*v\text{-alu-nat-adder-carry-out-off}$ ’.

EVENT: Enable $v\text{-alu-nat-adder-output}$; name this event ‘ $v\text{-alu-nat-adder-output-off}$ ’.

EVENT: Enable $*1*v\text{-alu-nat-adder-output}$; name this event ‘ $g*1*v\text{-alu-nat-adder-output-off}$ ’.

EVENT: Enable $v\text{-to-nat-append}$; name this event ‘ $v\text{-to-nat-append-off}$ ’.

EVENT: Enable $\text{not-lessp-quotient}$; name this event ‘ $\text{not-lessp-quotient-off}$ ’.

EVENT: Enable $\text{lessp-v-to-nat-exp-rewrite}$; name this event ‘ $\text{lessp-v-to-nat-exp-rewrite-off}$ ’.

EVENT: Enable $v\text{-alu-correct-nat-subtracter-output-lemma}$; name this event ‘ $v\text{-alu-correct-nat-subtracter-output-lemma-off}$ ’.

EVENT: Enable $\text{lessp-v-to-nat-exp}$; name this event ‘ $\text{lessp-v-to-nat-exp-off}$ ’.

EVENT: Enable v-to-nat-v-not; name this event ‘v-to-nat-v-not-off’.

EVENT: Enable v-to-nat-v-not-lemma; name this event ‘v-to-nat-v-not-lemma-off’.

EVENT: Enable equal-nat-to-v-inverter; name this event ‘equal-nat-to-v-inverter-off’.

EVENT: Enable nat-to-v-remainder; name this event ‘nat-to-v-remainder-off’.

EVENT: Enable equal-nat-to-v-inverter-hack2-lemma; name this event ‘equal-nat-to-v-inverter-hack2-lemma-off’.

EVENT: Enable nat-to-v-zero; name this event ‘nat-to-v-zero-off’.

EVENT: Enable nat-to-v-of-v-to-nat; name this event ‘nat-to-v-of-v-to-nat-off’.

EVENT: Enable firstn-length; name this event ‘firstn-length-off’.

EVENT: Enable nat-to-v-of-v-to-nat-general; name this event ‘nat-to-v-of-v-to-nat-general-off’.

EVENT: Enable firstn-nlistp; name this event ‘firstn-nlistp-off’.

EVENT: Enable firstn-zero; name this event ‘firstn-zero-off’.

EVENT: Enable firstn-add1-cons; name this event ‘firstn-add1-cons-off’.

EVENT: Enable nat-to-v-plus-x-x; name this event ‘nat-to-v-plus-x-x-off’.

EVENT: Enable quotient-plus-x-x-2; name this event ‘quotient-plus-x-x-2-off’.

EVENT: Enable remainder-plus-x-x-2; name this event ‘remainder-plus-x-x-2-off’.

EVENT: Enable v-to-nat-of-nat-to-v; name this event ‘v-to-nat-of-nat-to-v-off’.

EVENT: Enable v-to-nat-of-nat-to-v-hack; name this event 'v-to-nat-of-nat-to-v-hack-off'.

EVENT: Enable v-to-nat-firstn; name this event 'v-to-nat-firstn-off'.

EVENT: Enable remainder-add1-plus-multiple; name this event 'remainder-add1-plus-multiple-off'.

EVENT: Enable divides-plus-plus; name this event 'divides-plus-plus-off'.

EVENT: Enable remainder-plus-multiple; name this event 'remainder-plus-multiple-off'.

EVENT: Enable nth-length-v; name this event 'nth-length-v-off'.

EVENT: Enable v-alu-nat-int-buf; name this event 'v-alu-nat-int-buf-off'.

EVENT: Enable *1*v-alu-nat-int-buf; name this event 'g*1*v-alu-nat-int-buf-off'.

EVENT: Enable bvp-cdr; name this event 'bvp-cdr-off'.

EVENT: Enable b-to-nat; name this event 'b-to-nat-off'.

EVENT: Enable *1*b-to-nat; name this event 'g*1*b-to-nat-off'.

EVENT: Enable map-down-inverts-map-up; name this event 'map-down-inverts-map-up-off1'.

EVENT: Enable for-final-1-of-reset-sequence-chip; name this event 'for-final-1-of-reset-sequence-chip-off1'.

EVENT: Disable chip-well-formed-simple; name this event 'chip-well-formed-simple-on'.

EVENT: Disable chip-well-formed-after-indexed-names-removed; name this event 'chip-well-formed-after-indexed-names-removed-on'.

EVENT: Disable chip-well-formed; name this event ‘chip-well-formed-on’.

EVENT: Disable chip-system=fm9001-interpretter\$after-reset; name this event ‘chip-system=fm9001-interpretter\$after-reset-on’.

EVENT: Disable reset-chip-from-any-state; name this event ‘reset-chip-from-any-state-on’.

EVENT: Disable reset-chip; name this event ‘reset-chip-on’.

EVENT: Disable length-reset-sequence-chip-1; name this event ‘length-reset-sequence-chip-1-on’.

EVENT: Disable simulate-contains-simulate-dual-eval-2; name this event ‘simulate-contains-simulate-dual-eval-2-on’.

EVENT: Disable fm9001=chip-system-true-after-reset; name this event ‘fm9001=chip-system-true-after-reset-on’.

EVENT: Disable cadr-map-up; name this event ‘cadr-map-up-on’.

EVENT: Disable fm9001=chip-system-true-after-reset-lemma; name this event ‘fm9001=chip-system-true-after-reset-lemma-on’.

EVENT: Disable fm9001-statep-implies-memory-ok-p-instance; name this event ‘fm9001-statep-implies-memory-ok-p-instance-on’.

EVENT: Disable new-machine-state-invariant-implies-machine-state-invariant; name this event ‘new-machine-state-invariant-implies-machine-state-invariant-on’.

EVENT: Disable memory-v-fourp-implies-memory-properp; name this event ‘memory-v-fourp-implies-memory-properp-on’.

EVENT: Disable new-machine-state-invariant-is-non-trivial; name this event ‘new-machine-state-invariant-is-non-trivial-on’.

EVENT: Disable xs-suffice-for-reset-chip-final-state-for-any-unknown-state-better; name this event 'xs-suffice-for-reset-chip-final-state-for-any-unknown-state-better-on'.

EVENT: Disable xs-suffice-for-reset-chip-final-state-for-any-unknown-state; name this event 'xs-suffice-for-reset-chip-final-state-for-any-unknown-state-on'.

EVENT: Disable new-machine-state-invariant-implies-good-s; name this event 'new-machine-state-invariant-implies-good-s-on'.

EVENT: Disable v-fourp-is-good-s; name this event 'v-fourp-is-good-s-on'.

EVENT: Disable memory-v-fourp-implies-good-s; name this event 'memory-v-fourp-implies-good-s-on'.

EVENT: Disable good-s-opener; name this event 'good-s-opener-on'.

EVENT: Disable machine-state-invariant-implies-s-approx; name this event 'machine-state-invariant-implies-s-approx-on'.

EVENT: Disable machine-state-invariant-implies-s-approx-lemma-3; name this event 'machine-state-invariant-implies-s-approx-lemma-3-on'.

EVENT: Disable machine-state-invariant-implies-s-approx-lemma-2; name this event 'machine-state-invariant-implies-s-approx-lemma-2-on'.

EVENT: Disable machine-state-invariant-implies-s-approx-lemma; name this event 'machine-state-invariant-implies-s-approx-lemma-on'.

EVENT: Disable b-approx-x; name this event 'b-approx-x-on'.

EVENT: Disable stubp-implies-not-fourp; name this event 'stubp-implies-not-fourp-on'.

EVENT: Disable romp-implies-not-fourp; name this event 'romp-implies-not-fourp-on'.

EVENT: Disable ramp-implies-not-fourp; name this event ‘ramp-implies-not-fourp-on’.

EVENT: Disable listp-implies-not-fourp; name this event ‘listp-implies-not-fourp-on’.

EVENT: Disable v-fourp-implies-properp; name this event ‘v-fourp-implies-properp-on’.

EVENT: Disable s-approx-make-list; name this event ‘s-approx-make-list-on’.

EVENT: Disable new-machine-state-invariant; name this event ‘new-machine-state-invariant-on’.

EVENT: Disable *1*new-machine-state-invariant; name this event ‘g*1*new-machine-state-invariant-on’.

EVENT: Disable memory-v-fourp; name this event ‘memory-v-fourp-on’.

EVENT: Disable *1*memory-v-fourp; name this event ‘g*1*memory-v-fourp-on’.

EVENT: Disable all-xs-make-list; name this event ‘all-xs-make-list-on’.

EVENT: Disable all-xs-approximates; name this event ‘all-xs-approximates-on’.

EVENT: Disable all-xs; name this event ‘all-xs-on’.

EVENT: Disable *1*all-xs; name this event ‘g*1*all-xs-on’.

EVENT: Disable v-fourp; name this event ‘v-fourp-on’.

EVENT: Disable *1*v-fourp; name this event ‘g*1*v-fourp-on’.

EVENT: Disable fm9001-machine-statep-p-map-up-initialized-machine-state; name this event ‘fm9001-machine-statep-p-map-up-initialized-machine-state-on’.

EVENT: Disable xs-suffice-for-reset-chip-lemma-instance; name this event 'xs-suffice-for-reset-chip-lemma-instance-on'.

EVENT: Disable simulate-reset-chip-final-state; name this event 'simulate-reset-chip-final-state-on'.

EVENT: Disable reset-sequence-chip-1-vs-2; name this event 'reset-sequence-chip-1-vs-2-on'.

EVENT: Disable reset-sequence-chip-2; name this event 'reset-sequence-chip-2-on'.

EVENT: Disable *1*reset-sequence-chip-2; name this event 'g*1*reset-sequence-chip-2-on'.

EVENT: Disable reset-sequence-chip-1; name this event 'reset-sequence-chip-1-on'.

EVENT: Disable *1*reset-sequence-chip-1; name this event 'g*1*reset-sequence-chip-1-on'.

EVENT: Disable run-vector-chip-2; name this event 'run-vector-chip-2-on'.

EVENT: Disable *1*run-vector-chip-2; name this event 'g*1*run-vector-chip-2-on'.

EVENT: Disable run-vector-chip-1; name this event 'run-vector-chip-1-on'.

EVENT: Disable *1*run-vector-chip-1; name this event 'g*1*run-vector-chip-1-on'.

EVENT: Disable reset-vector-chip; name this event 'reset-vector-chip-on'.

EVENT: Disable *1*reset-vector-chip; name this event 'g*1*reset-vector-chip-on'.

EVENT: Disable fm9001-statep-map-up-final-state; name this event 'fm9001-

statep-map-up-final-state-on’.

EVENT: Disable instance-theorem; name this event ‘instance-theorem-on’.

EVENT: Disable final-state-okp; name this event ‘final-state-okp-on’.

EVENT: Disable unknown-state-okp; name this event ‘unknown-state-okp-on’.

EVENT: Disable reset-works; name this event ‘reset-works-on’.

EVENT: Disable final-state; name this event ‘final-state-on’.

EVENT: Disable *1*final-state; name this event ‘g*1*final-state-on’.

EVENT: Disable initialized-memory-state; name this event ‘initialized-memory-state-on’.

EVENT: Disable *1*initialized-memory-state; name this event ‘g*1*initialized-memory-state-on’.

EVENT: Disable initialized-machine-state; name this event ‘initialized-machine-state-on’.

EVENT: Disable *1*initialized-machine-state; name this event ‘g*1*initialized-machine-state-on’.

EVENT: Disable initialized-regfile; name this event ‘initialized-regfile-on’.

EVENT: Disable *1*initialized-regfile; name this event ‘g*1*initialized-regfile-on’.

EVENT: Disable chip-system-operating-inputs-p-reset-sequence; name this event ‘chip-system-operating-inputs-p-reset-sequence-on’.

EVENT: Disable reset-sequence; name this event ‘reset-sequence-on’.

EVENT: Disable *1*reset-sequence; name this event 'g*1*reset-sequence-on'.

EVENT: Disable run-vector; name this event 'run-vector-on'.

EVENT: Disable *1*run-vector; name this event 'g*1*run-vector-on'.

EVENT: Disable reset-vector; name this event 'reset-vector-on'.

EVENT: Disable *1*reset-vector; name this event 'g*1*reset-vector-on'.

EVENT: Disable fm9001-state-structure-unknown-state; name this event 'fm9001-state-structure-unknown-state-on'.

EVENT: Disable chip-system-invariant-unknown-state; name this event 'chip-system-invariant-unknown-state-on'.

EVENT: Disable unknown-state; name this event 'unknown-state-on'.

EVENT: Disable *1*unknown-state; name this event 'g*1*unknown-state-on'.

EVENT: Disable unknown-memory-state; name this event 'unknown-memory-state-on'.

EVENT: Disable *1*unknown-memory-state; name this event 'g*1*unknown-memory-state-on'.

EVENT: Disable unknown-machine-state; name this event 'unknown-machine-state-on'.

EVENT: Disable *1*unknown-machine-state; name this event 'g*1*unknown-machine-state-on'.

EVENT: Disable unknown-regfile; name this event 'unknown-regfile-on'.

EVENT: Disable *1*unknown-regfile; name this event 'g*1*unknown-regfile-on'.

EVENT: Disable compile-uncompiled-defns; name this event ‘compile-uncompiled-defns-on2’.

EVENT: For efficiency, compile those definitions not yet compiled.

EVENT: Disable *1*compile-uncompiled-defns; name this event ‘g*1*compile-uncompiled-defns-on’.

EVENT: Disable xs-suffice-for-reset-lemma; name this event ‘xs-suffice-for-reset-lemma-on’.

EVENT: Disable xs-suffice-for-reset-lemma-verbose; name this event ‘xs-suffice-for-reset-lemma-verbose-on’.

EVENT: Disable length-simulate; name this event ‘length-simulate-on’.

EVENT: Disable doublet-p-equal-approx; name this event ‘doublet-p-equal-approx-on’.

EVENT: Disable doublet-n-simulate; name this event ‘doublet-n-simulate-on’.

EVENT: Disable doublet-n-simulate-induction; name this event ‘doublet-n-simulate-induction-on’.

EVENT: Disable *1*doublet-n-simulate-induction; name this event ‘g*1*doublet-n-simulate-induction-on’.

EVENT: Disable *1*doublet-p; name this event ‘g*1*doublet-p-on’.

EVENT: Disable v-approx-list-x-x; name this event ‘v-approx-list-x-x-on’.

EVENT: Disable simulate-monotone; name this event ‘simulate-monotone-on’.

EVENT: Disable simulate-monotone-induction; name this event ‘simulate-monotone-induction-on’.

EVENT: Disable *1*simulate-monotone-induction; name this event ‘g*1*simulate-

monotone-induction-on’.

EVENT: Disable s-approx-cadr-nth; name this event ‘s-approx-cadr-nth-on’.

EVENT: Disable v-approx-car-nth; name this event ‘v-approx-car-nth-on’.

EVENT: Disable nat-lst-lst-induction; name this event ‘nat-lst-lst-induction-on’.

EVENT: Disable *1*nat-lst-lst-induction; name this event ‘g*1*nat-lst-lst-induction-on’.

EVENT: Disable v-s-approx-list; name this event ‘v-s-approx-list-on’.

EVENT: Disable *1*v-s-approx-list; name this event ‘g*1*v-s-approx-list-on’.

EVENT: Disable v-approx-list; name this event ‘v-approx-list-on’.

EVENT: Disable *1*v-approx-list; name this event ‘g*1*v-approx-list-on’.

EVENT: Disable good-s-preserved; name this event ‘good-s-preserved-on’.

EVENT: Disable primp-preserves-good-s; name this event ‘primp-preserves-good-s-on’.

EVENT: Disable dp-ram-16x32-preserves-good-s; name this event ‘dp-ram-16x32-preserves-good-s-on’.

EVENT: Disable good-s-write-mem; name this event ‘good-s-write-mem-on’.

EVENT: Disable good-s-write-mem-1; name this event ‘good-s-write-mem-1-on’.

EVENT: Disable good-s-constant-ram; name this event ‘good-s-constant-ram-on’.

EVENT: Disable vss-preserves-good-s; name this event ‘vss-preserves-good-s-on’.

EVENT: Disable vdd-parametric-preserves-good-s; name this event ‘vdd-parametric-preserves-good-s-on’.

EVENT: Disable vdd-preserves-good-s; name this event ‘vdd-preserves-good-s-on’.

EVENT: Disable ttl-tri-output-fast-preserves-good-s; name this event ‘ttl-tri-output-fast-preserves-good-s-on’.

EVENT: Disable ttl-tri-output-preserves-good-s; name this event ‘ttl-tri-output-preserves-good-s-on’.

EVENT: Disable ttl-output-fast-preserves-good-s; name this event ‘ttl-output-fast-preserves-good-s-on’.

EVENT: Disable ttl-output-parametric-preserves-good-s; name this event ‘ttl-output-parametric-preserves-good-s-on’.

EVENT: Disable ttl-output-preserves-good-s; name this event ‘ttl-output-preserves-good-s-on’.

EVENT: Disable ttl-input-preserves-good-s; name this event ‘ttl-input-preserves-good-s-on’.

EVENT: Disable ttl-clk-input-preserves-good-s; name this event ‘ttl-clk-input-preserves-good-s-on’.

EVENT: Disable ttl-bidirect-preserves-good-s; name this event ‘ttl-bidirect-preserves-good-s-on’.

EVENT: Disable pullup-preserves-good-s; name this event ‘pullup-preserves-good-s-on’.

EVENT: Disable t-wire-preserves-good-s; name this event ‘t-wire-preserves-good-s-on’.

EVENT: Disable t-buf-preserves-good-s; name this event ‘t-buf-preserves-good-s-on’.

EVENT: Disable ram-enable-circuit-preserves-good-s; name this event 'ram-enable-circuit-preserves-good-s-on'.

EVENT: Disable id-preserves-good-s; name this event 'id-preserves-good-s-on'.

EVENT: Disable fd1slp-preserves-good-s; name this event 'fd1slp-preserves-good-s-on'.

EVENT: Disable fd1sp-preserves-good-s; name this event 'fd1sp-preserves-good-s-on'.

EVENT: Disable fd1s-preserves-good-s; name this event 'fd1s-preserves-good-s-on'.

EVENT: Disable fd1-preserves-good-s; name this event 'fd1-preserves-good-s-on'.

EVENT: Disable procmon-preserves-good-s; name this event 'procmon-preserves-good-s-on'.

EVENT: Disable del10-preserves-good-s; name this event 'del10-preserves-good-s-on'.

EVENT: Disable del4-preserves-good-s; name this event 'del4-preserves-good-s-on'.

EVENT: Disable del2-preserves-good-s; name this event 'del2-preserves-good-s-on'.

EVENT: Disable b-xor3-preserves-good-s; name this event 'b-xor3-preserves-good-s-on'.

EVENT: Disable b-xor-preserves-good-s; name this event 'b-xor-preserves-good-s-on'.

EVENT: Disable b-or4-preserves-good-s; name this event 'b-or4-preserves-good-s-on'.

EVENT: Disable b-or3-preserves-good-s; name this event 'b-or3-preserves-good-s-on'.

EVENT: Disable b-or-preserves-good-s; name this event 'b-or-preserves-good-s-on'.

EVENT: Disable b-not-ivap-preserves-good-s; name this event 'b-not-ivap-preserves-good-s-on'.

EVENT: Disable b-not-b4ip-preserves-good-s; name this event 'b-not-b4ip-preserves-good-s-on'.

EVENT: Disable b-not-preserves-good-s; name this event 'b-not-preserves-good-s-on'.

EVENT: Disable b-nor8-preserves-good-s; name this event 'b-nor8-preserves-good-s-on'.

EVENT: Disable b-nor6-preserves-good-s; name this event 'b-nor6-preserves-good-s-on'.

EVENT: Disable b-nor5-preserves-good-s; name this event 'b-nor5-preserves-good-s-on'.

EVENT: Disable b-nor4-preserves-good-s; name this event 'b-nor4-preserves-good-s-on'.

EVENT: Disable b-nor3-preserves-good-s; name this event 'b-nor3-preserves-good-s-on'.

EVENT: Disable b-nor-preserves-good-s; name this event 'b-nor-preserves-good-s-on'.

EVENT: Disable b-nbuf-preserves-good-s; name this event 'b-nbuf-preserves-good-s-on'.

EVENT: Disable b-nand8-preserves-good-s; name this event 'b-nand8-preserves-good-s-on'.

EVENT: Disable b-nand6-preserves-good-s; name this event ‘b-nand6-preserves-good-s-on’.

EVENT: Disable b-nand5-preserves-good-s; name this event ‘b-nand5-preserves-good-s-on’.

EVENT: Disable b-nand4-preserves-good-s; name this event ‘b-nand4-preserves-good-s-on’.

EVENT: Disable b-nand3-preserves-good-s; name this event ‘b-nand3-preserves-good-s-on’.

EVENT: Disable b-nand-preserves-good-s; name this event ‘b-nand-preserves-good-s-on’.

EVENT: Disable b-if-preserves-good-s; name this event ‘b-if-preserves-good-s-on’.

EVENT: Disable b-equiv3-preserves-good-s; name this event ‘b-equiv3-preserves-good-s-on’.

EVENT: Disable b-equiv-preserves-good-s; name this event ‘b-equiv-preserves-good-s-on’.

EVENT: Disable b-and4-preserves-good-s; name this event ‘b-and4-preserves-good-s-on’.

EVENT: Disable b-and3-preserves-good-s; name this event ‘b-and3-preserves-good-s-on’.

EVENT: Disable b-and-preserves-good-s; name this event ‘b-and-preserves-good-s-on’.

EVENT: Disable ao7-preserves-good-s; name this event ‘ao7-preserves-good-s-on’.

EVENT: Disable ao6-preserves-good-s; name this event ‘ao6-preserves-good-s-on’.

EVENT: Disable ao4-preserves-good-s; name this event 'ao4-preserves-good-s-on'.

EVENT: Disable ao2-preserves-good-s; name this event 'ao2-preserves-good-s-on'.

EVENT: Disable good-s-0; name this event 'good-s-0-on'.

EVENT: Disable f-if-preserves-good-s; name this event 'f-if-preserves-good-s-on'.

EVENT: Disable f-buf-preserves-good-s; name this event 'f-buf-preserves-good-s-on'.

EVENT: Disable good-s-collect-value; name this event 'good-s-collect-value-on'.

EVENT: Disable dual-eval-monotone; name this event 'dual-eval-monotone-on'.

EVENT: Disable good-s-value; name this event 'good-s-value-on'.

EVENT: Disable good-s-alist-pairlist; name this event 'good-s-alist-pairlist-on'.

EVENT: Disable good-s-alist; name this event 'good-s-alist-on'.

EVENT: Disable *1*good-s-alist; name this event 'g*1*good-s-alist-on'.

EVENT: Disable dual-eval-monotone-no-ram; name this event 'dual-eval-monotone-no-ram-on'.

EVENT: Disable ok-netlistp-reduction-rewrite; name this event 'ok-netlistp-reduction-rewrite-on'.

EVENT: Disable ok-netlistp-reduction; name this event 'ok-netlistp-reduction-on'.

EVENT: Disable s-approx-alist-implies-s-approx-collect-value; name this event 's-approx-alist-implies-s-approx-collect-value-on'.

EVENT: Disable s-approx-implies-s-approx-alist; name this event 's-approx-implies-s-approx-alist-on'.

EVENT: Disable alistp-dual-eval-3; name this event 'alistp-dual-eval-3-on'.

EVENT: Disable s-approx-alist-implies-s-approx-list-collect-value; name this event 's-approx-alist-implies-s-approx-list-collect-value-on'.

EVENT: Disable double-cdr-induction; name this event 'double-cdr-induction-on'.

EVENT: Disable *1*double-cdr-induction; name this event 'g*1*double-cdr-induction-on'.

EVENT: Disable v-approx-alist-append; name this event 'v-approx-alist-append-on'.

EVENT: Disable s-approx-alist-implies-s-approx-value; name this event 's-approx-alist-implies-s-approx-value-on'.

EVENT: Disable alistp-opener; name this event 'alistp-opener-on'.

EVENT: Disable v-approx-implies-v-approx-alist; name this event 'v-approx-implies-v-approx-alist-on'.

EVENT: Disable s-approx-list-implies-s-approx-alist; name this event 's-approx-list-implies-s-approx-alist-on'.

EVENT: Disable s-approx-list; name this event 's-approx-list-on'.

EVENT: Disable *1*s-approx-list; name this event 'g*1*s-approx-list-on'.

EVENT: Disable v-approx-alist-implies-v-approx-collect-value; name this event 'v-approx-alist-implies-v-approx-collect-value-on'.

EVENT: Disable alistp-dual-eval-1; name this event 'alistp-dual-eval-1-on'.

EVENT: Disable alistp-append; name this event ‘alistp-append-on’.

EVENT: Disable alistp-pairlist; name this event ‘alistp-pairlist-on’.

EVENT: Disable v-approx-alist-implies-b-approx-value; name this event ‘v-approx-alist-implies-b-approx-value-on’.

EVENT: Disable dual-eval-monotone-induction; name this event ‘dual-eval-monotone-induction-on’.

EVENT: Disable *1*dual-eval-monotone-induction; name this event ‘g*1*dual-eval-monotone-induction-on’.

EVENT: Disable primp-monotone; name this event ‘primp-monotone-on’.

EVENT: Disable ok-netlistp; name this event ‘ok-netlistp-on’.

EVENT: Disable *1*ok-netlistp; name this event ‘g*1*ok-netlistp-on’.

EVENT: Disable dual-port-ram-state-body; name this event ‘dual-port-ram-state-body-on’.

EVENT: Disable *1*dual-port-ram-state-body; name this event ‘g*1*dual-port-ram-state-body-on’.

EVENT: Disable dual-port-ram-state-monotone; name this event ‘dual-port-ram-state-monotone-on’.

EVENT: Disable write-mem1-double-induction; name this event ‘write-mem1-double-induction-on’.

EVENT: Disable *1*write-mem1-double-induction; name this event ‘g*1*write-mem1-double-induction-on’.

EVENT: Disable write-mem1-monotone-induction; name this event ‘write-mem1-monotone-induction-on’.

EVENT: Disable *1*write-mem1-monotone-induction; name this event 'g*1*write-mem1-monotone-induction-on'.

EVENT: Disable dual-port-ram-value-body; name this event 'dual-port-ram-value-body-on'.

EVENT: Disable *1*dual-port-ram-value-body; name this event 'g*1*dual-port-ram-value-body-on'.

EVENT: Disable v-approx-x-x; name this event 'v-approx-x-x-on1'.

EVENT: Disable read-mem1-monotone-induction; name this event 'read-mem1-monotone-induction-on'.

EVENT: Disable *1*read-mem1-monotone-induction; name this event 'g*1*read-mem1-monotone-induction-on'.

EVENT: Disable bvp-rev1; name this event 'bvp-rev1-on'.

EVENT: Disable vss-monotone; name this event 'vss-monotone-on'.

EVENT: Disable dual-eval-vss-state; name this event 'dual-eval-vss-state-on'.

EVENT: Disable dual-eval-vss-value; name this event 'dual-eval-vss-value-on'.

EVENT: Disable vdd-parametric-monotone; name this event 'vdd-parametric-monotone-on'.

EVENT: Disable dual-eval-vdd-parametric-state; name this event 'dual-eval-vdd-parametric-state-on'.

EVENT: Disable dual-eval-vdd-parametric-value; name this event 'dual-eval-vdd-parametric-value-on'.

EVENT: Disable vdd-monotone; name this event 'vdd-monotone-on'.

EVENT: Disable dual-eval-vdd-state; name this event 'dual-eval-vdd-state-on'.

EVENT: Disable dual-eval-vdd-value; name this event ‘dual-eval-vdd-value-on’.

EVENT: Disable ttl-tri-output-fast-monotone; name this event ‘ttl-tri-output-fast-monotone-on’.

EVENT: Disable dual-eval-ttl-tri-output-fast-state; name this event ‘dual-eval-ttl-tri-output-fast-state-on’.

EVENT: Disable dual-eval-ttl-tri-output-fast-value; name this event ‘dual-eval-ttl-tri-output-fast-value-on’.

EVENT: Disable ttl-tri-output-monotone; name this event ‘ttl-tri-output-monotone-on’.

EVENT: Disable dual-eval-ttl-tri-output-state; name this event ‘dual-eval-ttl-tri-output-state-on’.

EVENT: Disable dual-eval-ttl-tri-output-value; name this event ‘dual-eval-ttl-tri-output-value-on’.

EVENT: Disable ttl-output-fast-monotone; name this event ‘ttl-output-fast-monotone-on’.

EVENT: Disable dual-eval-ttl-output-fast-state; name this event ‘dual-eval-ttl-output-fast-state-on’.

EVENT: Disable dual-eval-ttl-output-fast-value; name this event ‘dual-eval-ttl-output-fast-value-on’.

EVENT: Disable ttl-output-parametric-monotone; name this event ‘ttl-output-parametric-monotone-on’.

EVENT: Disable dual-eval-ttl-output-parametric-state; name this event ‘dual-eval-ttl-output-parametric-state-on’.

EVENT: Disable dual-eval-ttl-output-parametric-value; name this event ‘dual-eval-ttl-output-parametric-value-on’.

EVENT: Disable ttl-output-monotone; name this event ‘ttl-output-monotone-on’.

EVENT: Disable dual-eval-ttl-output-state; name this event ‘dual-eval-ttl-output-state-on’.

EVENT: Disable dual-eval-ttl-output-value; name this event ‘dual-eval-ttl-output-value-on’.

EVENT: Disable ttl-input-monotone; name this event ‘ttl-input-monotone-on’.

EVENT: Disable dual-eval-ttl-input-state; name this event ‘dual-eval-ttl-input-state-on’.

EVENT: Disable dual-eval-ttl-input-value; name this event ‘dual-eval-ttl-input-value-on’.

EVENT: Disable ttl-clk-input-monotone; name this event ‘ttl-clk-input-monotone-on’.

EVENT: Disable dual-eval-ttl-clk-input-state; name this event ‘dual-eval-ttl-clk-input-state-on’.

EVENT: Disable dual-eval-ttl-clk-input-value; name this event ‘dual-eval-ttl-clk-input-value-on’.

EVENT: Disable ttl-bidirect-monotone; name this event ‘ttl-bidirect-monotone-on’.

EVENT: Disable dual-eval-ttl-bidirect-state; name this event ‘dual-eval-ttl-bidirect-state-on’.

EVENT: Disable dual-eval-ttl-bidirect-value; name this event ‘dual-eval-ttl-bidirect-value-on’.

EVENT: Disable pullup-monotone; name this event ‘pullup-monotone-on’.

EVENT: Disable dual-eval-pullup-state; name this event 'dual-eval-pullup-state-on'.

EVENT: Disable dual-eval-pullup-value; name this event 'dual-eval-pullup-value-on'.

EVENT: Disable t-wire-monotone; name this event 't-wire-monotone-on'.

EVENT: Disable dual-eval-t-wire-state; name this event 'dual-eval-t-wire-state-on'.

EVENT: Disable dual-eval-t-wire-value; name this event 'dual-eval-t-wire-value-on'.

EVENT: Disable t-buf-monotone; name this event 't-buf-monotone-on'.

EVENT: Disable dual-eval-t-buf-state; name this event 'dual-eval-t-buf-state-on'.

EVENT: Disable dual-eval-t-buf-value; name this event 'dual-eval-t-buf-value-on'.

EVENT: Disable ram-enable-circuit-monotone; name this event 'ram-enable-circuit-monotone-on'.

EVENT: Disable dual-eval-ram-enable-circuit-state; name this event 'dual-eval-ram-enable-circuit-state-on'.

EVENT: Disable dual-eval-ram-enable-circuit-value; name this event 'dual-eval-ram-enable-circuit-value-on'.

EVENT: Disable id-monotone; name this event 'id-monotone-on'.

EVENT: Disable dual-eval-id-state; name this event 'dual-eval-id-state-on'.

EVENT: Disable dual-eval-id-value; name this event 'dual-eval-id-value-on'.

EVENT: Disable fd1slp-monotone; name this event 'fd1slp-monotone-on'.

EVENT: Disable dual-eval-fd1slp-state; name this event 'dual-eval-fd1slp-state-on'.

EVENT: Disable dual-eval-fd1slp-value; name this event 'dual-eval-fd1slp-value-on'.

EVENT: Disable fd1sp-monotone; name this event 'fd1sp-monotone-on'.

EVENT: Disable dual-eval-fd1sp-state; name this event 'dual-eval-fd1sp-state-on'.

EVENT: Disable dual-eval-fd1sp-value; name this event 'dual-eval-fd1sp-value-on'.

EVENT: Disable fd1s-monotone; name this event 'fd1s-monotone-on'.

EVENT: Disable dual-eval-fd1s-state; name this event 'dual-eval-fd1s-state-on'.

EVENT: Disable dual-eval-fd1s-value; name this event 'dual-eval-fd1s-value-on'.

EVENT: Disable fd1-monotone; name this event 'fd1-monotone-on'.

EVENT: Disable dual-eval-fd1-state; name this event 'dual-eval-fd1-state-on'.

EVENT: Disable dual-eval-fd1-value; name this event 'dual-eval-fd1-value-on'.

EVENT: Disable procmon-monotone; name this event 'procmon-monotone-on'.

EVENT: Disable dual-eval-procmon-state; name this event 'dual-eval-procmon-state-on'.

EVENT: Disable dual-eval-procmon-value; name this event 'dual-eval-procmon-value-on'.

EVENT: Disable del10-monotone; name this event ‘del10-monotone-on’.

EVENT: Disable dual-eval-del10-state; name this event ‘dual-eval-del10-state-on’.

EVENT: Disable dual-eval-del10-value; name this event ‘dual-eval-del10-value-on’.

EVENT: Disable del4-monotone; name this event ‘del4-monotone-on’.

EVENT: Disable dual-eval-del4-state; name this event ‘dual-eval-del4-state-on’.

EVENT: Disable dual-eval-del4-value; name this event ‘dual-eval-del4-value-on’.

EVENT: Disable del2-monotone; name this event ‘del2-monotone-on’.

EVENT: Disable dual-eval-del2-state; name this event ‘dual-eval-del2-state-on’.

EVENT: Disable dual-eval-del2-value; name this event ‘dual-eval-del2-value-on’.

EVENT: Disable b-xor3-monotone; name this event ‘b-xor3-monotone-on’.

EVENT: Disable dual-eval-b-xor3-state; name this event ‘dual-eval-b-xor3-state-on’.

EVENT: Disable dual-eval-b-xor3-value; name this event ‘dual-eval-b-xor3-value-on’.

EVENT: Disable b-xor-monotone; name this event ‘b-xor-monotone-on’.

EVENT: Disable dual-eval-b-xor-state; name this event ‘dual-eval-b-xor-state-on’.

EVENT: Disable dual-eval-b-xor-value; name this event ‘dual-eval-b-xor-value-on’.

EVENT: Disable b-or4-monotone; name this event ‘b-or4-monotone-on’.

EVENT: Disable dual-eval-b-or4-state; name this event ‘dual-eval-b-or4-state-on’.

EVENT: Disable dual-eval-b-or4-value; name this event ‘dual-eval-b-or4-value-on’.

EVENT: Disable b-or3-monotone; name this event ‘b-or3-monotone-on’.

EVENT: Disable dual-eval-b-or3-state; name this event ‘dual-eval-b-or3-state-on’.

EVENT: Disable dual-eval-b-or3-value; name this event ‘dual-eval-b-or3-value-on’.

EVENT: Disable b-or-monotone; name this event ‘b-or-monotone-on’.

EVENT: Disable dual-eval-b-or-state; name this event ‘dual-eval-b-or-state-on’.

EVENT: Disable dual-eval-b-or-value; name this event ‘dual-eval-b-or-value-on’.

EVENT: Disable b-not-ivap-monotone; name this event ‘b-not-ivap-monotone-on’.

EVENT: Disable dual-eval-b-not-ivap-state; name this event ‘dual-eval-b-not-ivap-state-on’.

EVENT: Disable dual-eval-b-not-ivap-value; name this event ‘dual-eval-b-not-ivap-value-on’.

EVENT: Disable b-not-b4ip-monotone; name this event ‘b-not-b4ip-monotone-on’.

EVENT: Disable dual-eval-b-not-b4ip-state; name this event ‘dual-eval-b-not-b4ip-state-on’.

EVENT: Disable dual-eval-b-not-b4ip-value; name this event ‘dual-eval-b-not-b4ip-value-on’.

EVENT: Disable b-not-monotone; name this event ‘b-not-monotone-on’.

EVENT: Disable dual-eval-b-not-state; name this event ‘dual-eval-b-not-state-on’.

EVENT: Disable dual-eval-b-not-value; name this event ‘dual-eval-b-not-value-on’.

EVENT: Disable b-nor8-monotone; name this event ‘b-nor8-monotone-on’.

EVENT: Disable dual-eval-b-nor8-state; name this event ‘dual-eval-b-nor8-state-on’.

EVENT: Disable dual-eval-b-nor8-value; name this event ‘dual-eval-b-nor8-value-on’.

EVENT: Disable b-nor6-monotone; name this event ‘b-nor6-monotone-on’.

EVENT: Disable dual-eval-b-nor6-state; name this event ‘dual-eval-b-nor6-state-on’.

EVENT: Disable dual-eval-b-nor6-value; name this event ‘dual-eval-b-nor6-value-on’.

EVENT: Disable b-nor5-monotone; name this event ‘b-nor5-monotone-on’.

EVENT: Disable dual-eval-b-nor5-state; name this event ‘dual-eval-b-nor5-state-on’.

EVENT: Disable dual-eval-b-nor5-value; name this event ‘dual-eval-b-nor5-value-on’.

EVENT: Disable b-nor4-monotone; name this event ‘b-nor4-monotone-on’.

EVENT: Disable dual-eval-b-nor4-state; name this event 'dual-eval-b-nor4-state-on'.

EVENT: Disable dual-eval-b-nor4-value; name this event 'dual-eval-b-nor4-value-on'.

EVENT: Disable b-nor3-monotone; name this event 'b-nor3-monotone-on'.

EVENT: Disable dual-eval-b-nor3-state; name this event 'dual-eval-b-nor3-state-on'.

EVENT: Disable dual-eval-b-nor3-value; name this event 'dual-eval-b-nor3-value-on'.

EVENT: Disable b-nor-monotone; name this event 'b-nor-monotone-on'.

EVENT: Disable dual-eval-b-nor-state; name this event 'dual-eval-b-nor-state-on'.

EVENT: Disable dual-eval-b-nor-value; name this event 'dual-eval-b-nor-value-on'.

EVENT: Disable b-nbuf-monotone; name this event 'b-nbuf-monotone-on'.

EVENT: Disable dual-eval-b-nbuf-state; name this event 'dual-eval-b-nbuf-state-on'.

EVENT: Disable dual-eval-b-nbuf-value; name this event 'dual-eval-b-nbuf-value-on'.

EVENT: Disable b-nand8-monotone; name this event 'b-nand8-monotone-on'.

EVENT: Disable dual-eval-b-nand8-state; name this event 'dual-eval-b-nand8-state-on'.

EVENT: Disable dual-eval-b-nand8-value; name this event 'dual-eval-b-nand8-value-on'.

EVENT: Disable b-nand6-monotone; name this event ‘b-nand6-monotone-on’.

EVENT: Disable dual-eval-b-nand6-state; name this event ‘dual-eval-b-nand6-state-on’.

EVENT: Disable dual-eval-b-nand6-value; name this event ‘dual-eval-b-nand6-value-on’.

EVENT: Disable b-nand5-monotone; name this event ‘b-nand5-monotone-on’.

EVENT: Disable dual-eval-b-nand5-state; name this event ‘dual-eval-b-nand5-state-on’.

EVENT: Disable dual-eval-b-nand5-value; name this event ‘dual-eval-b-nand5-value-on’.

EVENT: Disable b-nand4-monotone; name this event ‘b-nand4-monotone-on’.

EVENT: Disable dual-eval-b-nand4-state; name this event ‘dual-eval-b-nand4-state-on’.

EVENT: Disable dual-eval-b-nand4-value; name this event ‘dual-eval-b-nand4-value-on’.

EVENT: Disable b-nand3-monotone; name this event ‘b-nand3-monotone-on’.

EVENT: Disable dual-eval-b-nand3-state; name this event ‘dual-eval-b-nand3-state-on’.

EVENT: Disable dual-eval-b-nand3-value; name this event ‘dual-eval-b-nand3-value-on’.

EVENT: Disable b-nand-monotone; name this event ‘b-nand-monotone-on’.

EVENT: Disable dual-eval-b-nand-state; name this event ‘dual-eval-b-nand-state-on’.

EVENT: Disable dual-eval-b-nand-value; name this event ‘dual-eval-b-nand-value-on’.

EVENT: Disable b-if-monotone; name this event ‘b-if-monotone-on’.

EVENT: Disable dual-eval-b-if-state; name this event ‘dual-eval-b-if-state-on’.

EVENT: Disable dual-eval-b-if-value; name this event ‘dual-eval-b-if-value-on’.

EVENT: Disable b-equiv3-monotone; name this event ‘b-equiv3-monotone-on’.

EVENT: Disable dual-eval-b-equiv3-state; name this event ‘dual-eval-b-equiv3-state-on’.

EVENT: Disable dual-eval-b-equiv3-value; name this event ‘dual-eval-b-equiv3-value-on’.

EVENT: Disable b-equiv-monotone; name this event ‘b-equiv-monotone-on’.

EVENT: Disable dual-eval-b-equiv-state; name this event ‘dual-eval-b-equiv-state-on’.

EVENT: Disable dual-eval-b-equiv-value; name this event ‘dual-eval-b-equiv-value-on’.

EVENT: Disable b-and4-monotone; name this event ‘b-and4-monotone-on’.

EVENT: Disable dual-eval-b-and4-state; name this event ‘dual-eval-b-and4-state-on’.

EVENT: Disable dual-eval-b-and4-value; name this event ‘dual-eval-b-and4-value-on’.

EVENT: Disable b-and3-monotone; name this event ‘b-and3-monotone-on’.

EVENT: Disable dual-eval-b-and3-state; name this event ‘dual-eval-b-and3-state-on’.

EVENT: Disable dual-eval-b-and3-value; name this event ‘dual-eval-b-and3-value-on’.

EVENT: Disable b-and-monotone; name this event ‘b-and-monotone-on’.

EVENT: Disable dual-eval-b-and-state; name this event ‘dual-eval-b-and-state-on’.

EVENT: Disable dual-eval-b-and-value; name this event ‘dual-eval-b-and-value-on’.

EVENT: Disable ao7-monotone; name this event ‘ao7-monotone-on’.

EVENT: Disable dual-eval-ao7-state; name this event ‘dual-eval-ao7-state-on’.

EVENT: Disable dual-eval-ao7-value; name this event ‘dual-eval-ao7-value-on’.

EVENT: Disable ao6-monotone; name this event ‘ao6-monotone-on’.

EVENT: Disable dual-eval-ao6-state; name this event ‘dual-eval-ao6-state-on’.

EVENT: Disable dual-eval-ao6-value; name this event ‘dual-eval-ao6-value-on’.

EVENT: Disable ao4-monotone; name this event ‘ao4-monotone-on’.

EVENT: Disable dual-eval-ao4-state; name this event ‘dual-eval-ao4-state-on’.

EVENT: Disable dual-eval-ao4-value; name this event ‘dual-eval-ao4-value-on’.

EVENT: Disable ao2-monotone; name this event ‘ao2-monotone-on’.

EVENT: Disable dual-eval-ao2-state; name this event ‘dual-eval-ao2-state-on’.

EVENT: Disable dual-eval-ao2-value; name this event ‘dual-eval-ao2-value-on’.

EVENT: Disable fourp-implies-s-approx-is-b-approx; name this event ‘fourp-

implies-s-approx-is-b-approx-on1’.

EVENT: Disable fourp-f-if; name this event ‘fourp-f-if-on’.

EVENT: Disable fourp-f-buf; name this event ‘fourp-f-buf-on’.

EVENT: Disable f-buf-type-set; name this event ‘f-buf-type-set-on’.

EVENT: Disable s-approx-implies-b-approx; name this event ‘s-approx-implies-b-approx-on’.

EVENT: Disable dual-eval-2-primp; name this event ‘dual-eval-2-primp-on’.

EVENT: Disable dual-eval-0-primp; name this event ‘dual-eval-0-primp-on’.

EVENT: Disable f-pullup-monotone; name this event ‘f-pullup-monotone-on’.

EVENT: Disable ft-wire-monotone; name this event ‘ft-wire-monotone-on’.

EVENT: Disable ft-buf-monotone; name this event ‘ft-buf-monotone-on’.

EVENT: Disable f-if-monotone; name this event ‘f-if-monotone-on’.

EVENT: Disable f-equiv3-monotone; name this event ‘f-equiv3-monotone-on’.

EVENT: Disable f-xor3-monotone; name this event ‘f-xor3-monotone-on’.

EVENT: Disable f-equiv-monotone; name this event ‘f-equiv-monotone-on’.

EVENT: Disable f-xor-monotone; name this event ‘f-xor-monotone-on’.

EVENT: Disable f-nor8-monotone; name this event ‘f-nor8-monotone-on’.

EVENT: Disable f-nor6-monotone; name this event ‘f-nor6-monotone-on’.

EVENT: Disable f-nor5-monotone; name this event 'f-nor5-monotone-on'.

EVENT: Disable f-nor4-monotone; name this event 'f-nor4-monotone-on'.

EVENT: Disable f-nor3-monotone; name this event 'f-nor3-monotone-on'.

EVENT: Disable f-nor-monotone; name this event 'f-nor-monotone-on'.

EVENT: Disable f-or4-monotone; name this event 'f-or4-monotone-on'.

EVENT: Disable f-or3-monotone; name this event 'f-or3-monotone-on'.

EVENT: Disable f-or-monotone; name this event 'f-or-monotone-on'.

EVENT: Disable f-nand8-monotone; name this event 'f-nand8-monotone-on'.

EVENT: Disable f-nand6-monotone; name this event 'f-nand6-monotone-on'.

EVENT: Disable f-nand5-monotone; name this event 'f-nand5-monotone-on'.

EVENT: Disable f-nand4-monotone; name this event 'f-nand4-monotone-on'.

EVENT: Disable f-nand3-monotone; name this event 'f-nand3-monotone-on'.

EVENT: Disable f-nand-monotone; name this event 'f-nand-monotone-on'.

EVENT: Disable f-not-monotone; name this event 'f-not-monotone-on'.

EVENT: Disable f-and4-monotone; name this event 'f-and4-monotone-on'.

EVENT: Disable f-and3-monotone; name this event 'f-and3-monotone-on'.

EVENT: Disable f-and-monotone; name this event 'f-and-monotone-on'.

EVENT: Disable f-buf-monotone; name this event 'f-buf-monotone-on'.

EVENT: Disable monotonicity-property-consequence-3; name this event ‘monotonicity-property-consequence-3-on’.

EVENT: Disable monotonicity-property-consequence-2; name this event ‘monotonicity-property-consequence-2-on’.

EVENT: Disable monotonicity-property-consequence-1; name this event ‘monotonicity-property-consequence-1-on’.

EVENT: Disable monotonicity-property-consequence-0; name this event ‘monotonicity-property-consequence-0-on’.

EVENT: Disable *1*monotonicity-property; name this event ‘g*1*monotonicity-property-on’.

EVENT: Disable s-approx-alist; name this event ‘s-approx-alist-on’.

EVENT: Disable *1*s-approx-alist; name this event ‘g*1*s-approx-alist-on’.

EVENT: Disable v-approx-alist; name this event ‘v-approx-alist-on’.

EVENT: Disable *1*v-approx-alist; name this event ‘g*1*v-approx-alist-on’.

EVENT: Disable s-knownnp-implies-s-approx-is-equal; name this event ‘s-knownnp-implies-s-approx-is-equal-on1’.

EVENT: Disable s-knownnp; name this event ‘s-knownnp-on1’.

EVENT: Disable *1*s-knownnp; name this event ‘g*1*s-knownnp-on’.

EVENT: Disable good-s; name this event ‘good-s-on’.

EVENT: Disable *1*good-s; name this event ‘g*1*good-s-on’.

EVENT: Disable s-approx-x-x; name this event ‘s-approx-x-x-on’.

EVENT: Disable s-approx; name this event 's-approx-on1'.

EVENT: Disable *1*s-approx; name this event 'g*1*s-approx-on'.

EVENT: Disable mem-width; name this event 'mem-width-on1'.

EVENT: Disable *1*mem-width; name this event 'g*1*mem-width-on1'.

EVENT: Disable v-knownnp-implies-v-approx-is-equal; name this event 'v-knownnp-implies-v-approx-is-equal-on'.

EVENT: Disable v-knownnp; name this event 'v-knownnp-on1'.

EVENT: Disable *1*v-knownnp; name this event 'g*1*v-knownnp-on'.

EVENT: Disable v-approx; name this event 'v-approx-on1'.

EVENT: Disable *1*v-approx; name this event 'g*1*v-approx-on'.

EVENT: Disable *1*b-approx; name this event 'g*1*b-approx-on'.

EVENT: Disable b-knownnp-compound-recognizer; name this event 'b-knownnp-compound-recognizer-on'.

EVENT: Enable monotonicity-property; name this event 'monotonicity-property-off1'.

EVENT: Enable monotonicity-property-opener-0; name this event 'monotonicity-property-opener-0-off1'.

EVENT: Enable monotonicity-property-opener-1; name this event 'monotonicity-property-opener-1-off1'.

EVENT: Enable monotonicity-property-opener-2; name this event 'monotonicity-property-opener-2-off1'.

EVENT: Enable monotonicity-property-opener-3; name this event ‘monotonicity-property-opener-3-off1’.

EVENT: Enable f-buf; name this event ‘f-buf-off1’.

EVENT: Enable f-and; name this event ‘f-and-off1’.

EVENT: Enable f-and3; name this event ‘f-and3-off1’.

EVENT: Enable f-and4; name this event ‘f-and4-off1’.

EVENT: Enable f-not; name this event ‘f-not-off1’.

EVENT: Enable f-nand; name this event ‘f-nand-off1’.

EVENT: Enable f-nand3; name this event ‘f-nand3-off1’.

EVENT: Enable f-nand4; name this event ‘f-nand4-off1’.

EVENT: Enable f-nand5; name this event ‘f-nand5-off1’.

EVENT: Enable f-nand6; name this event ‘f-nand6-off1’.

EVENT: Enable f-nand8; name this event ‘f-nand8-off1’.

EVENT: Enable f-or; name this event ‘f-or-off1’.

EVENT: Enable f-or3; name this event ‘f-or3-off1’.

EVENT: Enable f-or4; name this event ‘f-or4-off1’.

EVENT: Enable f-nor; name this event ‘f-nor-off1’.

EVENT: Enable f-nor3; name this event ‘f-nor3-off1’.

EVENT: Enable f-nor4; name this event 'f-nor4-off1'.

EVENT: Enable f-nor5; name this event 'f-nor5-off1'.

EVENT: Enable f-nor6; name this event 'f-nor6-off1'.

EVENT: Enable f-nor8; name this event 'f-nor8-off1'.

EVENT: Enable f-xor; name this event 'f-xor-off1'.

EVENT: Enable f-xor3; name this event 'f-xor3-off1'.

EVENT: Enable f-equiv; name this event 'f-equiv-off1'.

EVENT: Enable f-equiv3; name this event 'f-equiv3-off1'.

EVENT: Enable f-if; name this event 'f-if-off1'.

EVENT: Enable ft-buf; name this event 'ft-buf-off1'.

EVENT: Enable ft-wire; name this event 'ft-wire-off1'.

EVENT: Enable f-pullup; name this event 'f-pullup-off1'.

EVENT: Enable bvp-reverse; name this event 'bvp-reverse-off1'.

EVENT: Enable v-approx-bvp; name this event 'v-approx-bvp-off1'.

EVENT: Enable read-mem1-monotone; name this event 'read-mem1-monotone-off1'.

EVENT: Enable bvp-implies-v-knownp; name this event 'bvp-implies-v-knownp-off1'.

EVENT: Enable v-approx-implies-b-approx-nth; name this event 'v-approx-implies-b-approx-nth-off1'.

EVENT: Enable *1*b-knownp; name this event 'g*1*b-knownp-off2'.

EVENT: Enable v-approx-implies-subranges-equal; name this event 'v-approx-implies-subranges-equal-off1'.

EVENT: Enable v-approx-bvp-subrange; name this event 'v-approx-bvp-subrange-off1'.

EVENT: Enable v-approx-make-list-x; name this event 'v-approx-make-list-x-off1'.

EVENT: Enable read-mem-monotone; name this event 'read-mem-monotone-off1'.

EVENT: Enable equal-length-read-mem1; name this event 'equal-length-read-mem1-off1'.

EVENT: Enable equal-length-read-mem; name this event 'equal-length-read-mem-off1'.

EVENT: Enable s-approx-implies-properp-read-mem1; name this event 's-approx-implies-properp-read-mem1-off1'.

EVENT: Enable s-approx-implies-properp-read-mem; name this event 's-approx-implies-properp-read-mem-off1'.

EVENT: Enable dual-port-ram-value-monotone; name this event 'dual-port-ram-value-monotone-off1'.

EVENT: Enable eval\$-append; name this event 'eval\$-append-off1'.

EVENT: Enable eval\$-pairlist-cons; name this event 'eval\$-pairlist-cons-off1'.

EVENT: Enable eval\$-append-2; name this event 'eval\$-append-2-off1'.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-1; name this event 'dual-apply-value-dp-ram-16x32-lemma-1-off1'.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-2; name this event ‘dual-apply-value-dp-ram-16x32-lemma-2-off1’.

EVENT: Enable dual-apply-value-dp-ram-16x32-lemma-3; name this event ‘dual-apply-value-dp-ram-16x32-lemma-3-off1’.

EVENT: Enable dual-port-ram-value-is-dual-port-ram-value-body; name this event ‘dual-port-ram-value-is-dual-port-ram-value-body-off1’.

EVENT: Enable dual-apply-value-dp-ram-16x32; name this event ‘dual-apply-value-dp-ram-16x32-off1’.

EVENT: Enable dp-ram-16x32-monotone-value; name this event ‘dp-ram-16x32-monotone-value-off1’.

EVENT: Disable bvp; name this event ‘bvp-on1’.

EVENT: Enable s-approx-opener; name this event ‘s-approx-opener-off1’.

EVENT: Enable v-approx-implies-nth-does-not-go-from-f-to-t; name this event ‘v-approx-implies-nth-does-not-go-from-f-to-t-off1’.

EVENT: Enable write-mem1-opener; name this event ‘write-mem1-opener-off1’.

EVENT: Enable write-mem1-monotone; name this event ‘write-mem1-monotone-off1’.

EVENT: Enable write-mem-monotone; name this event ‘write-mem-monotone-off1’.

EVENT: Enable v-approx-length; name this event ‘v-approx-length-off1’.

EVENT: Enable s-approx-write-mem1-id; name this event ‘s-approx-write-mem1-id-off1’.

EVENT: Enable s-approx-write-mem-id; name this event ‘s-approx-write-mem-id-off1’.

EVENT: Enable s-approx-constant-ram-x-id; name this event ‘s-approx-constant-ram-x-id-off1’.

EVENT: Enable s-approx-constant-ram-x-constant-ram-x; name this event ‘s-approx-constant-ram-x-constant-ram-x-off1’.

EVENT: Enable v-approx-preserves-length; name this event ‘v-approx-preserves-length-off1’.

EVENT: Enable v-approx-subrange; name this event ‘v-approx-subrange-off1’.

EVENT: Enable mem-width-non-zero; name this event ‘mem-width-non-zero-off1’.

EVENT: Enable s-approx-constant-ram-x-write-mem1-case-3; name this event ‘s-approx-constant-ram-x-write-mem1-case-3-off1’.

EVENT: Enable s-approx-constant-ram-x-write-mem1; name this event ‘s-approx-constant-ram-x-write-mem1-off1’.

EVENT: Enable s-approx-constant-ram-x-write-mem; name this event ‘s-approx-constant-ram-x-write-mem-off1’.

EVENT: Enable v-approx-preserves-properp; name this event ‘v-approx-preserves-properp-off1’.

EVENT: Enable mem-width-linear-facts; name this event ‘mem-width-linear-facts-off1’.

EVENT: Enable dual-port-ram-state-monotone-rewrite; name this event ‘dual-port-ram-state-monotone-rewrite-off1’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-1; name this event ‘dual-apply-state-dp-ram-16x32-lemma-1-off1’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-2; name this event ‘dual-apply-state-dp-ram-16x32-lemma-2-off1’.

EVENT: Enable dual-apply-state-dp-ram-16x32-lemma-3; name this event ‘dual-apply-state-dp-ram-16x32-lemma-3-off1’.

EVENT: Enable dual-port-ram-state-is-dual-port-ram-state-body; name this event ‘dual-port-ram-state-is-dual-port-ram-state-body-off1’.

EVENT: Enable dual-apply-state-dp-ram-16x32; name this event ‘dual-apply-state-dp-ram-16x32-off1’.

EVENT: Enable dp-ram-16x32-monotone-state; name this event ‘dp-ram-16x32-monotone-state-off1’.

EVENT: Enable dp-ram-16x32-monotone; name this event ‘dp-ram-16x32-monotone-off1’.

EVENT: Disable open-nth; name this event ‘open-nth-on1’.

EVENT: Enable doublet-p; name this event ‘doublet-p-off1’.

EVENT: Enable b-knownp; name this event ‘b-knownp-off2’.

EVENT: Enable b-approx; name this event ‘b-approx-off2’.

EVENT: Enable cancel-constants-ilessp; name this event ‘cancel-constants-ilessp-off1’.

EVENT: Enable cancel-constants-ilessp-lemma-2; name this event ‘cancel-constants-ilessp-lemma-2-off1’.

EVENT: Enable cancel-constants-ilessp-lemma-1; name this event ‘cancel-constants-ilessp-lemma-1-off1’.

EVENT: Enable illessp-add1-iplus; name this event ‘illessp-add1-iplus-off1’.

EVENT: Enable illessp-add1; name this event ‘illessp-add1-off1’.

EVENT: Enable cancel-constants-equal; name this event ‘cancel-constants-equal-

off1’.

EVENT: Enable cancel-constants-equal-lemma; name this event ‘cancel-constants-equal-lemma-off1’.

EVENT: Enable iplus-constants; name this event ‘iplus-constants-off1’.

EVENT: Enable cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded; name this event ‘cancel-ineg-terms-from-inequality-cancel-ineg-terms-from-inequality-expanded-off1’.

EVENT: Enable *1*canceled-ineg-terms-from-inequality-expanded; name this event ‘g*1*canceled-ineg-terms-from-inequality-expanded-off1’.

EVENT: Enable cancel-ineg-terms-from-inequality; name this event ‘cancel-ineg-terms-from-inequality-off1’.

EVENT: Enable *1*canceled-ineg-terms-from-inequality; name this event ‘g*1*canceled-ineg-terms-from-inequality-off1’.

EVENT: Enable make-cancel-ineg-terms-inequality; name this event ‘make-cancel-ineg-terms-inequality-off1’.

EVENT: Enable *1*make-cancel-ineg-terms-inequality; name this event ‘g*1*make-cancel-ineg-terms-inequality-off1’.

EVENT: Enable iplus-list-eval\$-car-split-out-ineg-terms; name this event ‘iplus-list-eval\$-car-split-out-ineg-terms-off1’.

EVENT: Enable eval\$-iplus-list-cdr-remove-inegs; name this event ‘eval\$-iplus-list-cdr-remove-inegs-off1’.

EVENT: Enable eval\$-iplus-list-car-remove-inegs; name this event ‘eval\$-iplus-list-car-remove-inegs-off1’.

EVENT: Enable cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded; name this event ‘cancel-ineg-terms-from-equality-cancel-ineg-terms-from-equality-expanded-off1’.

EVENT: Enable *1*cancel-ineg-terms-from-equality-expanded; name this event 'g*1*cancel-ineg-terms-from-equality-expanded-off1'.

EVENT: Enable cancel-ineg-terms-from-equality; name this event 'cancel-ineg-terms-from-equality-off1'.

EVENT: Enable *1*cancel-ineg-terms-from-equality; name this event 'g*1*cancel-ineg-terms-from-equality-off1'.

EVENT: Enable make-cancel-ineg-terms-equality; name this event 'make-cancel-ineg-terms-equality-off1'.

EVENT: Enable *1*make-cancel-ineg-terms-equality; name this event 'g*1*make-cancel-ineg-terms-equality-off1'.

EVENT: Enable *1*iplus-or-ineg-term; name this event 'g*1*iplus-or-ineg-term-off1'.

EVENT: Enable *1*remove-inegs; name this event 'g*1*remove-inegs-off1'.

EVENT: Enable split-out-ineg-terms; name this event 'split-out-ineg-terms-off1'.

EVENT: Enable *1*split-out-ineg-terms; name this event 'g*1*split-out-ineg-terms-off1'.

EVENT: Enable conjoin-inequalities-with-0-eliminator; name this event 'conjoin-inequalities-with-0-eliminator-off1'.

EVENT: Enable cancel-factors-ilessp-0; name this event 'cancel-factors-ilessp-0-off1'.

EVENT: Enable *1*cancel-factors-ilessp-0; name this event 'g*1*cancel-factors-ilessp-0-off1'.

EVENT: Enable conjoin-inequalities-with-0; name this event 'conjoin-inequalities-with-0-off1'.

EVENT: Enable *1*conjoin-inequalities-with-0; name this event 'g*1*conjoin-inequalities-with-0-off1'.

EVENT: Enable some-eval\$s-to-0-eliminator; name this event 'some-eval\$s-to-0-eliminator-off1'.

EVENT: Enable some-eval\$s-to-0-append; name this event 'some-eval\$s-to-0-append-off1'.

EVENT: Enable eval\$-disjoin-equalities-with-0; name this event 'eval\$-disjoin-equalities-with-0-off1'.

EVENT: Enable some-eval\$s-to-0; name this event 'some-eval\$s-to-0-off1'.

EVENT: Enable *1*some-eval\$s-to-0; name this event 'g*1*some-eval\$s-to-0-off1'.

EVENT: Enable cancel-factors-0; name this event 'cancel-factors-0-off1'.

EVENT: Enable *1*cancel-factors-0; name this event 'g*1*cancel-factors-0-off1'.

EVENT: Enable disjoin-equalities-with-0; name this event 'disjoin-equalities-with-0-off1'.

EVENT: Enable *1*disjoin-equalities-with-0; name this event 'g*1*disjoin-equalities-with-0-off1'.

EVENT: Enable equal-fix-int-to-ilessp; name this event 'equal-fix-int-to-ilessp-off1'.

EVENT: Enable eval\$list-bagint-0-implies-equal-for-ilessp-lemma; name this event 'eval\$list-bagint-0-implies-equal-for-ilessp-lemma-off1'.

EVENT: Enable eval\$list-bagint-0-for-ilessp; name this event 'eval\$list-bagint-0-for-ilessp-off1'.

EVENT: Enable member-0-itimes-factors-yields-0-ilessp-consequence-2; name this

event ‘member-0-itimes-factors-yields-0-ilessp-consequence-2-off1’.

EVENT: Enable member-0-itimes-factors-yields-0-ilessp-consequence-1; name this event ‘member-0-itimes-factors-yields-0-ilessp-consequence-1-off1’.

EVENT: Enable member-0-itimes-factors-yields-0; name this event ‘member-0-itimes-factors-yields-0-off1’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff-corollary-2; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-2-off1’.

EVENT: Enable ilessp-zero-implies-not-equal; name this event ‘ilessp-zero-implies-not-equal-off1’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff-corollary-1; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-corollary-1-off1’.

EVENT: Enable zero-ilessp-implies-not-equal; name this event ‘zero-ilessp-implies-not-equal-off1’.

EVENT: Enable ilessp-itimes-list-eval\$list-bagdiff; name this event ‘ilessp-itimes-list-eval\$list-bagdiff-off1’.

EVENT: Enable bagint-singleton; name this event ‘bagint-singleton-off1’.

EVENT: Enable cancel-itimes-ilessp-factors; name this event ‘cancel-itimes-ilessp-factors-off1’.

EVENT: Enable *1*cancel-itimes-ilessp-factors; name this event ‘g*1*cancel-itimes-ilessp-factors-off1’.

EVENT: Enable member-0-eval\$list; name this event ‘member-0-eval\$list-off1’.

EVENT: Enable membership-of-0-implies-itimes-list-is-0; name this event ‘membership-of-0-implies-itimes-list-is-0-off1’.

EVENT: Enable cancel-itimes-factors-expanded-cancel-itimes-factors; name this

event ‘cancel-itimes-factors-expanded-cancel-itimes-factors-off1’.

EVENT: Enable $*1*$ cancel-itimes-factors-expanded; name this event ‘g $*1*$ cancel-itimes-factors-expanded-off1’.

EVENT: Enable cancel-itimes-factors; name this event ‘cancel-itimes-factors-off1’.

EVENT: Enable $*1*$ cancel-itimes-factors; name this event ‘g $*1*$ cancel-itimes-factors-off1’.

EVENT: Enable $*1*$ iplus-or-itimes-term; name this event ‘g $*1*$ iplus-or-itimes-term-off1’.

EVENT: Enable itimes-eval\$-itimes-tree-ineg; name this event ‘itimes-eval\$-itimes-tree-ineg-off1’.

EVENT: Enable iplus-eval\$-itimes-tree-ineg; name this event ‘iplus-eval\$-itimes-tree-ineg-off1’.

EVENT: Enable neg-eval\$-itimes-tree-ineg; name this event ‘neg-eval\$-itimes-tree-ineg-off1’.

EVENT: Enable eval\$-itimes-tree-ineg; name this event ‘eval\$-itimes-tree-ineg-off1’.

EVENT: Enable fix-int-eval\$-itimes-tree-rec; name this event ‘fix-int-eval\$-itimes-tree-rec-off1’.

EVENT: Enable illessp-ineg-ineg; name this event ‘illessp-ineg-ineg-off1’.

EVENT: Enable equal-ineg-ineg; name this event ‘equal-ineg-ineg-off1’.

EVENT: Enable itimes-1; name this event ‘itimes-1-off1’.

EVENT: Enable itimes-factors; name this event ‘itimes-factors-off1’.

EVENT: Enable *1*itimes-factors; name this event 'g*1*itimes-factors-off1'.

EVENT: Enable *1*itimes-tree-ineg; name this event 'g*1*itimes-tree-ineg-off1'.

EVENT: Enable illessp-strict; name this event 'illessp-strict-off1'.

EVENT: Enable illessp-itimes-right-negative; name this event 'illessp-itimes-right-negative-off1'.

EVENT: Enable illessp-itimes-right-positive; name this event 'illessp-itimes-right-positive-off1'.

EVENT: Enable cancel-itimes-illessp; name this event 'cancel-itimes-illessp-off1'.

EVENT: Enable *1*cancel-itimes-illessp; name this event 'g*1*cancel-itimes-illessp-off1'.

EVENT: Enable *1*make-cancel-itimes-inequality; name this event 'g*1*make-cancel-itimes-inequality-off1'.

EVENT: Enable *1*itimes-tree-no-fix-int; name this event 'g*1*itimes-tree-no-fix-int-off1'.

EVENT: Enable subsetp; name this event 'subsetp-off1'.

EVENT: Enable *1*subsetp; name this event 'g*1*subsetp-off1'.

EVENT: Enable equal-fix-int; name this event 'equal-fix-int-off1'.

EVENT: Enable member-append; name this event 'member-append-off1'.

EVENT: Enable itimes-list-append; name this event 'itimes-list-append-off1'.

EVENT: Enable cancel-itimes; name this event 'cancel-itimes-off1'.

EVENT: Enable *1*cancel-itimes; name this event 'g*1*cancel-itimes-off1'.

EVENT: Enable *1*make-cancel-itimes-equality; name this event 'g*1*make-cancel-itimes-equality-off1'.

EVENT: Enable integerp-itimes-list; name this event 'integerp-itimes-list-off1'.

EVENT: Enable itimes-list; name this event 'itimes-list-off1'.

EVENT: Enable *1*itimes-list; name this event 'g*1*itimes-list-off1'.

EVENT: Enable *1*itimes-tree; name this event 'g*1*itimes-tree-off1'.

EVENT: Enable itimes-tree-rec; name this event 'itimes-tree-rec-off1'.

EVENT: Enable *1*itimes-tree-rec; name this event 'g*1*itimes-tree-rec-off1'.

EVENT: Enable itimes-fringe; name this event 'itimes-fringe-off1'.

EVENT: Enable *1*itimes-fringe; name this event 'g*1*itimes-fringe-off1'.

EVENT: Enable fix-int-irem; name this event 'fix-int-irem-off1'.

EVENT: Enable fix-int-iquo; name this event 'fix-int-iquo-off1'.

EVENT: Enable fix-int-imod; name this event 'fix-int-imod-off1'.

EVENT: Enable fix-int-idiv; name this event 'fix-int-idiv-off1'.

EVENT: Enable fix-int-iremainder; name this event 'fix-int-iremainder-off1'.

EVENT: Enable fix-int-iquotient; name this event 'fix-int-iquotient-off1'.

EVENT: Enable irem-fix-int2; name this event 'irem-fix-int2-off1'.

EVENT: Enable irem-fix-int1; name this event 'irem-fix-int1-off1'.

EVENT: Enable iquo-fix-int2; name this event 'iquo-fix-int2-off1'.

EVENT: Enable iquo-fix-int1; name this event 'iquo-fix-int1-off1'.

EVENT: Enable imod-fix-int2; name this event 'imod-fix-int2-off1'.

EVENT: Enable imod-fix-int1; name this event 'imod-fix-int1-off1'.

EVENT: Enable idiv-fix-int2; name this event 'idiv-fix-int2-off1'.

EVENT: Enable idiv-fix-int1; name this event 'idiv-fix-int1-off1'.

EVENT: Enable iremainder-fix-int2; name this event 'iremainder-fix-int2-off1'.

EVENT: Enable iremainder-fix-int1; name this event 'iremainder-fix-int1-off1'.

EVENT: Enable iquotient-fix-int2; name this event 'iquotient-fix-int2-off1'.

EVENT: Enable iquotient-fix-int1; name this event 'iquotient-fix-int1-off1'.

EVENT: Enable integerp-irem; name this event 'integerp-irem-off1'.

EVENT: Enable integerp-iquo; name this event 'integerp-iquo-off1'.

EVENT: Enable integerp-imod; name this event 'integerp-imod-off1'.

EVENT: Enable integerp-idiv; name this event 'integerp-idiv-off1'.

EVENT: Enable integerp-iremainder; name this event 'integerp-iremainder-off1'.

EVENT: Enable integerp-iquotient; name this event 'integerp-iquotient-off1'.

EVENT: Enable itimes-cancellation-3; name this event 'itimes-cancellation-3-off1'.

EVENT: Enable itimes-cancellation-2; name this event ‘itimes-cancellation-2-off1’.

EVENT: Enable itimes-cancellation-1; name this event ‘itimes-cancellation-1-off1’.

EVENT: Enable itimes-ineg-2; name this event ‘itimes-ineg-2-off1’.

EVENT: Enable itimes-ineg-1; name this event ‘itimes-ineg-1-off1’.

EVENT: Enable iquo-irem-uniqueness; name this event ‘iquo-irem-uniqueness-off1’.

EVENT: Enable division-theorem-for-truncate-to-zero; name this event ‘division-theorem-for-truncate-to-zero-off1’.

EVENT: Enable division-theorem-for-truncate-to-zero-part3; name this event ‘division-theorem-for-truncate-to-zero-part3-off1’.

EVENT: Enable division-theorem-for-truncate-to-zero-part2; name this event ‘division-theorem-for-truncate-to-zero-part2-off1’.

EVENT: Enable division-theorem-for-truncate-to-zero-part1; name this event ‘division-theorem-for-truncate-to-zero-part1-off1’.

EVENT: Enable idiv-imod-uniqueness; name this event ‘idiv-imod-uniqueness-off1’.

EVENT: Enable division-theorem-for-truncate-to-neginf; name this event ‘division-theorem-for-truncate-to-neginf-off1’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part3; name this event ‘division-theorem-for-truncate-to-neginf-part3-off1’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part2; name this event ‘division-theorem-for-truncate-to-neginf-part2-off1’.

EVENT: Enable division-theorem-for-truncate-to-neginf-part1; name this event 'division-theorem-for-truncate-to-neginf-part1-off1'.

EVENT: Enable iquotient-iremainder-uniqueness; name this event 'iquotient-iremainder-uniqueness-off1'.

EVENT: Enable division-theorem; name this event 'division-theorem-off1'.

EVENT: Enable division-theorem-part3; name this event 'division-theorem-part3-off1'.

EVENT: Enable division-theorem-part2; name this event 'division-theorem-part2-off1'.

EVENT: Enable division-theorem-part1; name this event 'division-theorem-part1-off1'.

EVENT: Enable quotient-remainder-uniqueness; name this event 'quotient-remainder-uniqueness-off1'.

EVENT: Enable itimes-1-arg1; name this event 'itimes-1-arg1-off1'.

EVENT: Enable equal-itimes-minus-1; name this event 'equal-itimes-minus-1-off1'.

EVENT: Enable equal-itimes-1; name this event 'equal-itimes-1-off1'.

EVENT: Enable equal-itimes-0; name this event 'equal-itimes-0-off1'.

EVENT: Enable associativity-of-itimes; name this event 'associativity-of-itimes-off1'.

EVENT: Enable commutativity2-of-itimes; name this event 'commutativity2-of-itimes-off1'.

EVENT: Enable itimes-distributes-over-iplus; name this event 'itimes-distributes-over-iplus-off1'.

EVENT: Enable itimes-distributes-over-iplus-proof; name this event ‘itimes-distributes-over-iplus-proof-off1’.

EVENT: Enable commutativity-of-itimes; name this event ‘commutativity-of-itimes-off1’.

EVENT: Enable itimes-fix-int2; name this event ‘itimes-fix-int2-off1’.

EVENT: Enable itimes-fix-int1; name this event ‘itimes-fix-int1-off1’.

EVENT: Enable itimes-0-right; name this event ‘itimes-0-right-off1’.

EVENT: Enable itimes-0-left; name this event ‘itimes-0-left-off1’.

EVENT: Enable cancel-iplus-ilessp; name this event ‘cancel-iplus-ilessp-off1’.

EVENT: Enable $*1*$ cancel-iplus-ilessp; name this event ‘ $g*1*$ cancel-iplus-ilessp-off1’.

EVENT: Enable $*1*$ iplus-tree-no-fix-int; name this event ‘ $g*1*$ iplus-tree-no-fix-int-off1’.

EVENT: Enable iplus-cancellation-2-for-ilessp; name this event ‘iplus-cancellation-2-for-ilessp-off1’.

EVENT: Enable iplus-cancellation-1-for-ilessp; name this event ‘iplus-cancellation-1-for-ilessp-off1’.

EVENT: Enable cancel-iplus-ilessp-1; name this event ‘cancel-iplus-ilessp-1-off1’.

EVENT: Enable $*1*$ cancel-iplus-ilessp-1; name this event ‘ $g*1*$ cancel-iplus-ilessp-1-off1’.

EVENT: Enable $*1*$ make-cancel-iplus-inequality-1; name this event ‘ $g*1*$ make-cancel-iplus-inequality-1-off1’.

EVENT: Enable illessp-fix-int-2; name this event ‘illessp-fix-int-2-off1’.

EVENT: Enable ilssp-fix-int-1; name this event 'ilssp-fix-int-1-off1'.

EVENT: Enable iplus-ineg7; name this event 'iplus-ineg7-off1'.

EVENT: Enable iplus-list-append; name this event 'iplus-list-append-off1'.

EVENT: Enable eval\$iplus-list-delete; name this event 'eval\$iplus-list-delete-off1'.

EVENT: Enable *1*cancel-iplus; name this event 'g*1*cancel-iplus-off1'.

EVENT: Enable eval\$list-append; name this event 'eval\$list-append-off1'.

EVENT: Enable integerp-iplus-list; name this event 'integerp-iplus-list-off1'.

EVENT: Enable iplus-list; name this event 'iplus-list-off1'.

EVENT: Enable *1*iplus-list; name this event 'g*1*iplus-list-off1'.

EVENT: Enable *1*iplus-tree; name this event 'g*1*iplus-tree-off1'.

EVENT: Enable iplus-tree-rec; name this event 'iplus-tree-rec-off1'.

EVENT: Enable *1*iplus-tree-rec; name this event 'g*1*iplus-tree-rec-off1'.

EVENT: Enable iplus-fringe; name this event 'iplus-fringe-off1'.

EVENT: Enable *1*iplus-fringe; name this event 'g*1*iplus-fringe-off1'.

EVENT: Enable iplus-x-y-ineg-x; name this event 'iplus-x-y-ineg-x-off1'.

EVENT: Enable iplus-ineg4; name this event 'iplus-ineg4-off1'.

EVENT: Enable iplus-ineg3; name this event 'iplus-ineg3-off1'.

EVENT: Enable eval\$-cancel-ineg-aux-is-its-fn; name this event ‘eval\$-cancel-ineg-aux-is-its-fn-off1’.

EVENT: Enable eval\$-cancel-ineg-aux-fn; name this event ‘eval\$-cancel-ineg-aux-fn-off1’.

EVENT: Enable *1*eval\$-cancel-ineg-aux-fn; name this event ‘g*1*eval\$-cancel-ineg-aux-fn-off1’.

EVENT: Enable eval\$-other; name this event ‘eval\$-other-off1’.

EVENT: Enable eval\$-litatom; name this event ‘eval\$-litatom-off1’.

EVENT: Enable eval\$-list-nlistp; name this event ‘eval\$-list-nlistp-off1’.

EVENT: Enable eval\$-list-cons; name this event ‘eval\$-list-cons-off1’.

EVENT: Enable cancel-ineg; name this event ‘cancel-ineg-off1’.

EVENT: Enable *1*cancel-ineg; name this event ‘g*1*cancel-ineg-off1’.

EVENT: Enable cancel-ineg-aux; name this event ‘cancel-ineg-aux-off1’.

EVENT: Enable *1*cancel-ineg-aux; name this event ‘g*1*cancel-ineg-aux-off1’.

EVENT: Enable idifference-fix-int2; name this event ‘idifference-fix-int2-off1’.

EVENT: Enable idifference-fix-int1; name this event ‘idifference-fix-int1-off1’.

EVENT: Enable iplus-fix-int2; name this event ‘iplus-fix-int2-off1’.

EVENT: Enable iplus-fix-int1; name this event ‘iplus-fix-int1-off1’.

EVENT: Enable iplus-ineg2; name this event ‘iplus-ineg2-off1’.

EVENT: Enable iplus-ineg1; name this event ‘iplus-ineg1-off1’.

EVENT: Enable iplus-cancellation-2; name this event ‘iplus-cancellation-2-off1’.

EVENT: Enable iplus-cancellation-1; name this event ‘iplus-cancellation-1-off1’.

EVENT: Enable associativity-of-iplus; name this event ‘associativity-of-iplus-off1’.

EVENT: Enable commutativity-of-iplus; name this event ‘commutativity-of-iplus-off1’.

EVENT: Enable commutativity2-of-iplus; name this event ‘commutativity2-of-iplus-off1’.

EVENT: Enable iplus-0-right; name this event ‘iplus-0-right-off1’.

EVENT: Enable iplus-0-left; name this event ‘iplus-0-left-off1’.

EVENT: Enable ineg-0; name this event ‘ineg-0-off1’.

EVENT: Enable ineg-fix-int; name this event ‘ineg-fix-int-off1’.

EVENT: Enable ineg-ineg; name this event ‘ineg-ineg-off1’.

EVENT: Enable ineg-iplus; name this event ‘ineg-iplus-off1’.

EVENT: Enable fix-int-itimes; name this event ‘fix-int-itimes-off1’.

EVENT: Enable fix-int-iabs; name this event ‘fix-int-iabs-off1’.

EVENT: Enable fix-int-ineg; name this event ‘fix-int-ineg-off1’.

EVENT: Enable fix-int-idifference; name this event ‘fix-int-idifference-off1’.

EVENT: Enable fix-int-iplus; name this event ‘fix-int-iplus-off1’.

EVENT: Enable fix-int-fix-int; name this event ‘fix-int-fix-int-off1’.

EVENT: Enable fix-int-remover; name this event 'fix-int-remover-off1'.

EVENT: Enable integerp-itimes; name this event 'integerp-itimes-off1'.

EVENT: Enable integerp-iabs; name this event 'integerp-iabs-off1'.

EVENT: Enable integerp-ineg; name this event 'integerp-ineg-off1'.

EVENT: Enable integerp-idifference; name this event 'integerp-idifference-off1'.

EVENT: Enable integerp-iplus; name this event 'integerp-iplus-off1'.

EVENT: Enable integerp-fix-int; name this event 'integerp-fix-int-off1'.

EVENT: Enable *1*irem; name this event 'g*1*irem-off1'.

EVENT: Enable *1*iquo; name this event 'g*1*iquo-off1'.

EVENT: Enable *1*imod; name this event 'g*1*imod-off1'.

EVENT: Enable *1*idiv; name this event 'g*1*idiv-off1'.

EVENT: Enable *1*iremainder; name this event 'g*1*iremainder-off1'.

EVENT: Enable *1*iquotient; name this event 'g*1*iquotient-off1'.

EVENT: Enable *1*itimes; name this event 'g*1*itimes-off1'.

EVENT: Enable *1*iabs; name this event 'g*1*iabs-off1'.

EVENT: Enable idifference; name this event 'idifference-off1'.

EVENT: Enable *1*idifference; name this event 'g*1*idifference-off1'.

EVENT: Enable *1*ineg; name this event 'g*1*ineg-off1'.

EVENT: Enable *1*iplus; name this event 'g*1*iplus-off1'.

EVENT: Enable ileq; name this event 'ileq-off1'.

EVENT: Enable *1*ileq; name this event 'g*1*ileq-off1'.

EVENT: Enable *1*ilessp; name this event 'g*1*ilessp-off1'.

EVENT: Enable izerop; name this event 'izerop-off1'.

EVENT: Enable *1*izerop; name this event 'g*1*izerop-off1'.

EVENT: Enable *1*fix-int; name this event 'g*1*fix-int-off1'.

EVENT: Enable *1*integerp; name this event 'g*1*integerp-off1'.

EVENT: Enable gcd-idempotence; name this event 'gcd-idempotence-off1'.

EVENT: Enable gcd-x-x; name this event 'gcd-x-x-off1'.

EVENT: Enable commutativity2-of-gcd; name this event 'commutativity2-of-gcd-off1'.

EVENT: Enable commutativity2-of-gcd-zero-case; name this event 'commutativity2-of-gcd-zero-case-off1'.

EVENT: Enable associativity-of-gcd; name this event 'associativity-of-gcd-off1'.

EVENT: Enable associativity-of-gcd-zero-case; name this event 'associativity-of-gcd-zero-case-off1'.

EVENT: Enable common-divisor-divides-gcd; name this event 'common-divisor-divides-gcd-off1'.

EVENT: Enable gcd-is-the-greatest; name this event 'gcd-is-the-greatest-off1'.

EVENT: Enable distributivity-of-times-over-gcd; name this event 'distributivity-of-times-over-gcd-off1'.

EVENT: Enable distributivity-of-times-over-gcd-proof; name this event 'distributivity-of-times-over-gcd-proof-off1'.

EVENT: Enable remainder-gcd; name this event 'remainder-gcd-off1'.

EVENT: Enable gcd-plus-instance; name this event 'gcd-plus-instance-off1'.

EVENT: Enable gcd-plus; name this event 'gcd-plus-off1'.

EVENT: Enable gcd-plus-proof; name this event 'gcd-plus-proof-off1'.

EVENT: Enable gcd-plus-instance-temp; name this event 'gcd-plus-instance-temp-off1'.

EVENT: Enable gcd-plus-instance-temp-proof; name this event 'gcd-plus-instance-temp-proof-off1'.

EVENT: Enable lessp-gcd; name this event 'lessp-gcd-off1'.

EVENT: Enable equal-gcd-0; name this event 'equal-gcd-0-off1'.

EVENT: Enable gcd-1; name this event 'gcd-1-off1'.

EVENT: Enable gcd-0; name this event 'gcd-0-off1'.

EVENT: Enable single-number-induction; name this event 'single-number-induction-off1'.

EVENT: Enable *1*single-number-induction; name this event 'g*1*single-number-induction-off1'.

EVENT: Enable commutativity-of-gcd; name this event 'commutativity-of-gcd-off1'.

EVENT: Enable log-exp; name this event 'log-exp-off1'.

EVENT: Enable log-times-exp; name this event 'log-times-exp-off1'.

EVENT: Enable log-times-exp-proof; name this event 'log-times-exp-proof-off1'.

EVENT: Enable log-times; name this event 'log-times-off1'.

EVENT: Enable log-times-proof; name this event 'log-times-proof-off1'.

EVENT: Enable log-quotient-exp; name this event 'log-quotient-exp-off1'.

EVENT: Enable log-quotient-times; name this event 'log-quotient-times-off1'.

EVENT: Enable log-quotient-times-proof; name this event 'log-quotient-times-proof-off1'.

EVENT: Enable log-quotient; name this event 'log-quotient-off1'.

EVENT: Enable leq-log-log; name this event 'leq-log-log-off1'.

EVENT: Enable double-log-induction; name this event 'double-log-induction-off1'.

EVENT: Enable *1*double-log-induction; name this event 'g*1*double-log-induction-off1'.

EVENT: Enable log-1; name this event 'log-1-off1'.

EVENT: Enable log-0; name this event 'log-0-off1'.

EVENT: Enable equal-log-0; name this event 'equal-log-0-off1'.

EVENT: Enable exp-difference; name this event 'exp-difference-off1'.

EVENT: Enable equal-exp-1; name this event 'equal-exp-1-off1'.

EVENT: Enable equal-exp-0; name this event 'equal-exp-0-off1'.

EVENT: Enable exp-exp; name this event 'exp-exp-off1'.

EVENT: Enable exp-times; name this event 'exp-times-off1'.

EVENT: Enable exp-0-arg2; name this event 'exp-0-arg2-off1'.

EVENT: Enable exp-1-arg1; name this event 'exp-1-arg1-off1'.

EVENT: Enable exp-0-arg1; name this event 'exp-0-arg1-off1'.

EVENT: Enable exp-plus; name this event 'exp-plus-off1'.

EVENT: Enable exp-add1; name this event 'exp-add1-off1'.

EVENT: Enable exp-zero; name this event 'exp-zero-off1'.

EVENT: Enable quotient-exp; name this event 'quotient-exp-off1'.

EVENT: Enable remainder-exp-exp; name this event 'remainder-exp-exp-off1'.

EVENT: Enable double-number-induction; name this event 'double-number-induction-off1'.

EVENT: Enable *1*double-number-induction; name this event 'g*1*double-number-induction-off1'.

EVENT: Enable remainder-exp; name this event 'remainder-exp-off1'.

EVENT: Enable gcd; name this event 'gcd-off1'.

EVENT: Enable *1*gcd; name this event 'g*1*gcd-off1'.

EVENT: Enable log; name this event 'log-off1'.

EVENT: Enable *1*log; name this event 'g*1*log-off1'.

EVENT: Enable exp; name this event 'exp-off1'.

EVENT: Enable *1*exp; name this event 'g*1*exp-off1'.

EVENT: Enable cancel-quotient-times; name this event 'cancel-quotient-times-off1'.

EVENT: Enable *1*cancel-quotient-times; name this event 'g*1*cancel-quotient-times-off1'.

EVENT: Enable lessp-quotient; name this event 'lessp-quotient-off1'.

EVENT: Enable quotient-x-x; name this event 'quotient-x-x-off1'.

EVENT: Enable quotient-1-arg1; name this event 'quotient-1-arg1-off1'.

EVENT: Enable quotient-1-arg1-casesplit; name this event 'quotient-1-arg1-casesplit-off1'.

EVENT: Enable quotient-1-arg2; name this event 'quotient-1-arg2-off1'.

EVENT: Enable leq-quotient; name this event 'leq-quotient-off1'.

EVENT: Enable quotient-quotient; name this event 'quotient-quotient-off1'.

EVENT: Enable quotient-plus-times-times-instance; name this event 'quotient-plus-times-times-instance-off1'.

EVENT: Enable quotient-plus-times-times; name this event 'quotient-plus-times-times-off1'.

EVENT: Enable quotient-plus-times-times-proof; name this event 'quotient-plus-times-times-proof-off1'.

EVENT: Enable quotient-plus-fact; name this event ‘quotient-plus-fact-off1’.

EVENT: Enable quotient-remainder-instance; name this event ‘quotient-remainder-instance-off1’.

EVENT: Enable quotient-remainder; name this event ‘quotient-remainder-off1’.

EVENT: Enable quotient-remainder-times; name this event ‘quotient-remainder-times-off1’.

EVENT: Enable quotient-difference3; name this event ‘quotient-difference3-off1’.

EVENT: Enable quotient-difference2; name this event ‘quotient-difference2-off1’.

EVENT: Enable quotient-lessp-arg1; name this event ‘quotient-lessp-arg1-off1’.

EVENT: Enable quotient-difference1; name this event ‘quotient-difference1-off1’.

EVENT: Enable quotient-times-times-proof; name this event ‘quotient-times-times-proof-off1’.

EVENT: Enable quotient-times-instance; name this event ‘quotient-times-instance-off1’.

EVENT: Enable quotient-times; name this event ‘quotient-times-off1’.

EVENT: Enable quotient-times-proof; name this event ‘quotient-times-proof-off1’.

EVENT: Enable quotient-times-instance-temp-proof; name this event ‘quotient-times-instance-temp-proof-off1’.

EVENT: Enable quotient-plus; name this event ‘quotient-plus-off1’.

EVENT: Enable quotient-plus-proof; name this event ‘quotient-plus-proof-off1’.

EVENT: Enable quotient-sub1; name this event 'quotient-sub1-off1'.

EVENT: Enable equal-quotient-0; name this event 'equal-quotient-0-off1'.

EVENT: Enable quotient-add1; name this event 'quotient-add1-off1'.

EVENT: Enable quotient-zero; name this event 'quotient-zero-off1'.

EVENT: Enable quotient-of-non-number; name this event 'quotient-of-non-number-off1'.

EVENT: Enable quotient-noop; name this event 'quotient-noop-off1'.

EVENT: Enable transitivity-of-divides; name this event 'transitivity-of-divides-off1'.

EVENT: Enable remainder-x-x; name this event 'remainder-x-x-off1'.

EVENT: Enable remainder-1-arg2; name this event 'remainder-1-arg2-off1'.

EVENT: Enable remainder-1-arg1; name this event 'remainder-1-arg1-off1'.

EVENT: Enable remainder-remainder; name this event 'remainder-remainder-off1'.

EVENT: Enable remainder-plus-times-times-instance; name this event 'remainder-plus-times-times-instance-off1'.

EVENT: Enable remainder-plus-times-times; name this event 'remainder-plus-times-times-off1'.

EVENT: Enable remainder-plus-times-times-proof; name this event 'remainder-plus-times-times-proof-off1'.

EVENT: Enable remainder-plus-fact; name this event 'remainder-plus-fact-off1'.

EVENT: Enable remainder-difference2; name this event ‘remainder-difference2-off1’.

EVENT: Enable double-remainder-induction; name this event ‘double-remainder-induction-off1’.

EVENT: Enable *1*double-remainder-induction; name this event ‘g*1*double-remainder-induction-off1’.

EVENT: Enable remainder-difference1; name this event ‘remainder-difference1-off1’.

EVENT: Enable remainder-times2-instance; name this event ‘remainder-times2-instance-off1’.

EVENT: Enable remainder-times2; name this event ‘remainder-times2-off1’.

EVENT: Enable remainder-times2-proof; name this event ‘remainder-times2-proof-off1’.

EVENT: Enable remainder-times-times-proof; name this event ‘remainder-times-times-proof-off1’.

EVENT: Enable remainder-times1-instance; name this event ‘remainder-times1-instance-off1’.

EVENT: Enable remainder-times1-instance-proof; name this event ‘remainder-times1-instance-proof-off1’.

EVENT: Enable remainder-times1; name this event ‘remainder-times1-off1’.

EVENT: Enable remainder-times1-proof; name this event ‘remainder-times1-proof-off1’.

EVENT: Enable equal-remainder-plus-remainder-proof; name this event ‘equal-remainder-plus-remainder-proof-off1’.

EVENT: Enable equal-remainder-plus-0; name this event 'equal-remainder-plus-0-off1'.

EVENT: Enable equal-remainder-plus-0-proof; name this event 'equal-remainder-plus-0-proof-off1'.

EVENT: Enable remainder-plus; name this event 'remainder-plus-off1'.

EVENT: Enable remainder-plus-proof; name this event 'remainder-plus-proof-off1'.

EVENT: Enable remainder-add1; name this event 'remainder-add1-off1'.

EVENT: Enable remainder-quotient-elim; name this event 'remainder-quotient-elim-off1'.

EVENT: Enable remainder-zero; name this event 'remainder-zero-off1'.

EVENT: Enable remainder-of-non-number; name this event 'remainder-of-non-number-off1'.

EVENT: Enable remainder-noop; name this event 'remainder-noop-off1'.

EVENT: Enable lessp-remainder; name this event 'lessp-remainder-off1'.

EVENT: Enable cancel-equal-times; name this event 'cancel-equal-times-off1'.

EVENT: Enable *1*cancel-equal-times; name this event 'g*1*cancel-equal-times-off1'.

EVENT: Enable cancel-lessp-times; name this event 'cancel-lessp-times-off1'.

EVENT: Enable *1*cancel-lessp-times; name this event 'g*1*cancel-lessp-times-off1'.

EVENT: Enable infer-equality-from-not-lessp; name this event 'infer-equality-from-not-lessp-off1'.

EVENT: Enable lessp-times-arg1; name this event 'lessp-times-arg1-off1'.

EVENT: Enable eval\$-equal; name this event 'eval\$-equal-off2'.

EVENT: Enable and-not-zerop-tree; name this event 'and-not-zerop-tree-off1'.

EVENT: Enable $*1*$ and-not-zerop-tree; name this event ' $g*1*$ and-not-zerop-tree-off1'.

EVENT: Enable or-zerop-tree; name this event 'or-zerop-tree-off1'.

EVENT: Enable $*1*$ or-zerop-tree; name this event ' $g*1*$ or-zerop-tree-off1'.

EVENT: Enable times-fringe; name this event 'times-fringe-off1'.

EVENT: Enable $*1*$ times-fringe; name this event ' $g*1*$ times-fringe-off1'.

EVENT: Enable times-tree; name this event 'times-tree-off1'.

EVENT: Enable $*1*$ times-tree; name this event ' $g*1*$ times-tree-off1'.

EVENT: Enable lessp-1-times; name this event 'lessp-1-times-off1'.

EVENT: Enable lessp-plus-times2; name this event 'lessp-plus-times2-off1'.

EVENT: Enable lessp-plus-times1; name this event 'lessp-plus-times1-off1'.

EVENT: Enable lessp-plus-times-proof; name this event 'lessp-plus-times-proof-off1'.

EVENT: Enable lessp-times-cancellation-proof; name this event 'lessp-times-cancellation-proof-off1'.

EVENT: Enable lessp-times3; name this event 'lessp-times3-off1'.

EVENT: Enable lessp-times3-proof2; name this event 'lessp-times3-proof2-off1'.

EVENT: Enable lessp-times3-proof1; name this event 'lessp-times3-proof1-off1'.

EVENT: Enable lessp-times2; name this event 'lessp-times2-off1'.

EVENT: Enable lessp-times2-proof; name this event 'lessp-times2-proof-off1'.

EVENT: Enable lessp-times1; name this event 'lessp-times1-off1'.

EVENT: Enable lessp-times1-proof; name this event 'lessp-times1-proof-off1'.

EVENT: Enable times-1-arg1; name this event 'times-1-arg1-off1'.

EVENT: Enable times-quotient; name this event 'times-quotient-off1'.

EVENT: Enable times-quotient-proof; name this event 'times-quotient-proof-off1'.

EVENT: Enable times-distributes-over-difference; name this event 'times-distributes-over-difference-off1'.

EVENT: Enable times-distributes-over-difference-proof; name this event 'times-distributes-over-difference-proof-off1'.

EVENT: Enable associativity-of-times; name this event 'associativity-of-times-off1'.

EVENT: Enable commutativity2-of-times; name this event 'commutativity2-of-times-off1'.

EVENT: Enable times-distributes-over-plus; name this event 'times-distributes-over-plus-off1'.

EVENT: Enable times-distributes-over-plus-proof; name this event 'times-distributes-over-plus-proof-off1'.

EVENT: Enable commutativity-of-times; name this event 'commutativity-of-times-off1'.

EVENT: Enable times-add1; name this event 'times-add1-off1'.

EVENT: Enable times-zero; name this event 'times-zero-off1'.

EVENT: Enable equal-sub1-0; name this event 'equal-sub1-0-off1'.

EVENT: Enable equal-times-1; name this event 'equal-times-1-off1'.

EVENT: Enable equal-times-0; name this event 'equal-times-0-off1'.

EVENT: Enable cancel-lessp-plus; name this event 'cancel-lessp-plus-off1'.

EVENT: Enable $*1*$ cancel-lessp-plus; name this event ' $g*1*$ cancel-lessp-plus-off1'.

EVENT: Enable difference-x-x; name this event 'difference-x-x-off1'.

EVENT: Enable difference-difference-arg2; name this event 'difference-difference-arg2-off1'.

EVENT: Enable difference-difference-arg1; name this event 'difference-difference-arg1-off1'.

EVENT: Enable difference-sub1-arg2; name this event 'difference-sub1-arg2-off1'.

EVENT: Enable difference-add1-arg2; name this event 'difference-add1-arg2-off1'.

EVENT: Enable difference-leq-arg1; name this event 'difference-leq-arg1-off1'.

EVENT: Enable difference-elim; name this event 'difference-elim-off1'.

EVENT: Enable cancel-difference-plus; name this event 'cancel-difference-plus-off1'.

EVENT: Enable *1*cancel-difference-plus; name this event 'g*1*cancel-difference-plus-off1'.

EVENT: Enable cancel-equal-plus; name this event 'cancel-equal-plus-off1'.

EVENT: Enable *1*cancel-equal-plus; name this event 'g*1*cancel-equal-plus-off1'.

EVENT: Enable eval\$-quote; name this event 'eval\$-quote-off2'.

EVENT: Enable plus-tree; name this event 'plus-tree-off1'.

EVENT: Enable *1*plus-tree; name this event 'g*1*plus-tree-off1'.

EVENT: Enable plus-fringe; name this event 'plus-fringe-off1'.

EVENT: Enable *1*plus-fringe; name this event 'g*1*plus-fringe-off1'.

EVENT: Enable diff-diff-arg2; name this event 'diff-diff-arg2-off1'.

EVENT: Enable diff-diff-arg1; name this event 'diff-diff-arg1-off1'.

EVENT: Enable difference-plus-plus-cancellation-proof; name this event 'difference-plus-plus-cancellation-proof-off1'.

EVENT: Enable difference-plus-cancellation-proof; name this event 'difference-plus-cancellation-proof-off1'.

EVENT: Enable plus-difference-arg2; name this event 'plus-difference-arg2-off1'.

EVENT: Enable plus-difference-arg1; name this event 'plus-difference-arg1-off1'.

EVENT: Enable associativity-of-plus; name this event 'associativity-of-plus-off1'.

EVENT: Enable plus-add1-arg2; name this event 'plus-add1-arg2-off1'.

EVENT: Enable plus-add1-arg1; name this event ‘plus-add1-arg1-off1’.

EVENT: Enable plus-zero-arg2; name this event ‘plus-zero-arg2-off1’.

EVENT: Enable commutativity2-of-plus; name this event ‘commutativity2-of-plus-off1’.

EVENT: Enable commutativity-of-plus; name this event ‘commutativity-of-plus-off1’.

EVENT: Enable equal-difference-0; name this event ‘equal-difference-0-off1’.

EVENT: Enable equal-plus-0; name this event ‘equal-plus-0-off1’.

EVENT: Enable member-bagint; name this event ‘member-bagint-off1’.

EVENT: Enable occurrences-bagdiff; name this event ‘occurrences-bagdiff-off1’.

EVENT: Enable occurrences-bagint; name this event ‘occurrences-bagint-off1’.

EVENT: Enable subbagp-bagint2; name this event ‘subbagp-bagint2-off1’.

EVENT: Enable subbagp-bagint1; name this event ‘subbagp-bagint1-off1’.

EVENT: Enable subbagp-cdr2; name this event ‘subbagp-cdr2-off1’.

EVENT: Enable subbagp-cdr1; name this event ‘subbagp-cdr1-off1’.

EVENT: Enable subbagp-delete; name this event ‘subbagp-delete-off1’.

EVENT: Enable bagdiff-delete; name this event ‘bagdiff-delete-off1’.

EVENT: Enable member-bagdiff; name this event ‘member-bagdiff-off1’.

EVENT: Enable occurrences-delete; name this event ‘occurrences-delete-off1’.

EVENT: Enable member-delete-implies-membership; name this event 'member-delete-implies-membership-off1'.

EVENT: Enable member-delete; name this event 'member-delete-off1'.

EVENT: Enable member-non-list; name this event 'member-non-list-off1'.

EVENT: Enable equal-occurrences-zero; name this event 'equal-occurrences-zero-off1'.

EVENT: Enable delete-delete; name this event 'delete-delete-off1'.

EVENT: Enable delete-non-member; name this event 'delete-non-member-off1'.

EVENT: Enable subbagp; name this event 'subbagp-off1'.

EVENT: Enable *1*subbagp; name this event 'g*1*subbagp-off1'.

EVENT: Enable occurrences; name this event 'occurrences-off1'.

EVENT: Enable *1*occurrences; name this event 'g*1*occurrences-off1'.

EVENT: Enable bagint; name this event 'bagint-off1'.

EVENT: Enable *1*bagint; name this event 'g*1*bagint-off1'.

EVENT: Enable bagdiff; name this event 'bagdiff-off1'.

EVENT: Enable *1*bagdiff; name this event 'g*1*bagdiff-off1'.

EVENT: Enable delete; name this event 'delete-off1'.

EVENT: Enable *1*delete; name this event 'g*1*delete-off1'.

EVENT: Disable v-adder-output=v-sum; name this event 'v-adder-output=v-sum-on'.

EVENT: Disable v-adder-carry-out=v-carry; name this event 'v-adder-carry-out=v-carry-on'.

EVENT: Disable v-to-nat-firstn-hack1; name this event 'v-to-nat-firstn-hack1-on'.

EVENT: Disable v-to-nat-firstn-hack2; name this event 'v-to-nat-firstn-hack2-on'.

EVENT: Disable equal-nat-to-v-inverter-hack1; name this event 'equal-nat-to-v-inverter-hack1-on'.

EVENT: Disable equal-nat-to-v-inverter-hack2; name this event 'equal-nat-to-v-inverter-hack2-on'.

EVENT: Disable equal-iff; name this event 'equal-iff-on'.

EVENT: Disable quotient-plus-x-x-2-with-extra-arg; name this event 'quotient-plus-x-x-2-with-extra-arg-on'.

EVENT: Disable plus-quotient-2-lemma; name this event 'plus-quotient-2-lemma-on'.

EVENT: Disable plus-quotient-2; name this event 'plus-quotient-2-on'.

EVENT: Disable v-alu-correct-nat-int-buf; name this event 'v-alu-correct-nat-int-buf-on'.

EVENT: Disable v-alu-correct-nat-adder-carry-out; name this event 'v-alu-correct-nat-adder-carry-out-on'.

EVENT: Disable v-alu-correct-nat-adder-output; name this event 'v-alu-correct-nat-adder-output-on'.

EVENT: Disable v-alu-correct-nat-adder; name this event 'v-alu-correct-nat-adder-on'.

EVENT: Disable v-alu-correct-nat-inc; name this event ‘v-alu-correct-nat-inc-on’.

EVENT: Disable v-alu-correct-nat-subtractor-carry-out; name this event ‘v-alu-correct-nat-subtractor-carry-out-on’.

EVENT: Disable v-alu-correct-nat-subtractor-output; name this event ‘v-alu-correct-nat-subtractor-output-on’.

EVENT: Disable v-alu-correct-nat-subtractor; name this event ‘v-alu-correct-nat-subtractor-on’.

EVENT: Disable v-alu-correct-nat-dec; name this event ‘v-alu-correct-nat-dec-on’.

EVENT: Disable v-alu-correct-nat-lsr-carry-out; name this event ‘v-alu-correct-nat-lsr-carry-out-on’.

EVENT: Disable v-alu-correct-nat-lsr-output; name this event ‘v-alu-correct-nat-lsr-output-on’.

EVENT: Disable v-alu-correct-nat-lsr; name this event ‘v-alu-correct-nat-lsr-on’.

EVENT: Disable v-alu-correct-nat-not-output; name this event ‘v-alu-correct-nat-not-output-on’.

EVENT: Disable v-alu-correct-nat-not; name this event ‘v-alu-correct-nat-not-on’.

EVENT: Disable v-alu-correct-int-adder-overflowp; name this event ‘v-alu-correct-int-adder-overflowp-on’.

EVENT: Disable v-alu-correct-int-adder-output; name this event ‘v-alu-correct-int-adder-output-on’.

EVENT: Disable v-alu-correct-int-adder; name this event ‘v-alu-correct-int-adder-on’.

EVENT: Disable v-alu-correct-int-inc; name this event ‘v-alu-correct-int-inc-on’.

EVENT: Disable v-alu-correct-int-subtractor-overflow; name this event ‘v-alu-correct-int-subtractor-overflow-on’.

EVENT: Disable v-alu-correct-int-subtractor-output; name this event ‘v-alu-correct-int-subtractor-output-on’.

EVENT: Disable v-alu-correct-int-subtractor; name this event ‘v-alu-correct-int-subtractor-on’.

EVENT: Disable v-alu-correct-int-dec; name this event ‘v-alu-correct-int-dec-on’.

EVENT: Disable v-alu-correct-int-asr-output; name this event ‘v-alu-correct-int-asr-output-on’.

EVENT: Disable v-alu-correct-int-asr; name this event ‘v-alu-correct-int-asr-on’.

EVENT: Disable v-alu-correct-int-neg; name this event ‘v-alu-correct-int-neg-on’.

EVENT: Disable v-alu-correct-int-not; name this event ‘v-alu-correct-int-not-on’.

EVENT: Enable v-alu-correct-int-not-output; name this event ‘v-alu-correct-int-not-output-off1’.

EVENT: Enable v-alu-correct-nat-not-output; name this event ‘v-alu-correct-nat-not-output-off2’.

EVENT: Enable v-alu-correct-nat-lsr-carry-out; name this event ‘v-alu-correct-nat-lsr-carry-out-off1’.

EVENT: Enable v-adder-output=v-sum; name this event ‘v-adder-output=v-sum-off1’.

EVENT: Enable v-alu-correct-nat-inc; name this event 'v-alu-correct-nat-inc-off1'.

EVENT: Enable v-alu-correct-nat-adder; name this event 'v-alu-correct-nat-adder-off1'.

EVENT: Enable v-alu-correct-int-asr; name this event 'v-alu-correct-int-asr-off1'.

EVENT: Enable v-alu-correct-nat-lsr; name this event 'v-alu-correct-nat-lsr-off1'.

EVENT: Disable v-to-nat-of-nat-to-v; name this event 'v-to-nat-of-nat-to-v-on'.

EVENT: Disable iplus-0-left; name this event 'iplus-0-left-on1'.

EVENT: Disable iplus-0-right; name this event 'iplus-0-right-on1'.

EVENT: Disable commutativity-of-iplus; name this event 'commutativity-of-iplus-on1'.

THEOREM: v-negp->v-nzerop
 $\text{v-negp}(v) \rightarrow \text{v-nzerop}(v)$

THEOREM: v-zerop-implies-v-to-nat-0
 $\text{v-zerop}(a) \rightarrow (\text{v-to-nat}(a) = 0)$

THEOREM: boolp-flag-extracters
 $\text{boolp}(\text{c-flag}(\text{update-flags}(flags, \text{set-flags}, cvzbv)))$
 $\wedge \text{boolp}(\text{v-flag}(\text{update-flags}(flags, \text{set-flags}, cvzbv)))$
 $\wedge \text{boolp}(\text{n-flag}(\text{update-flags}(flags, \text{set-flags}, cvzbv)))$
 $\wedge \text{boolp}(\text{z-flag}(\text{update-flags}(flags, \text{set-flags}, cvzbv)))$

THEOREM: v-to-nat-equal
 $\text{bv2p}(a, b) \rightarrow ((\text{v-to-nat}(a) = \text{v-to-nat}(b)) = (a = b))$

THEOREM: v-nzerop-nat-to-v-zero
 $(z \simeq 0) \rightarrow (\neg \text{v-nzerop}(\text{nat-to-v}(z, l)))$

THEOREM: v-not-v-not
 $\text{bvp}(a) \rightarrow (\text{v-not}(\text{v-not}(a)) = a)$

THEOREM: rewrite-equal-v-to-nat-0
 $(v\text{-to-nat}(a) = 0) = v\text{-zerop}(a)$

EVENT: Enable rewrite-equal-v-to-nat-0; name this event ‘rewrite-equal-v-to-nat-0-off’.

DEFINITION: $cc\text{-}cc = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

DEFINITION: $cc\text{-}cs = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

DEFINITION: $cc\text{-}vc = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

DEFINITION: $cc\text{-}vs = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

DEFINITION: $cc\text{-}pl = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

DEFINITION: $cc\text{-}mi = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

DEFINITION: $cc\text{-}ne = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

DEFINITION: $cc\text{-}eq = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

DEFINITION: $cc\text{-}hi = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

DEFINITION: $cc\text{-}ls = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

DEFINITION: $cc\text{-}ge = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})$

DEFINITION: $cc\text{-}lt = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})$

DEFINITION: $cc\text{-}gt = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

DEFINITION: $cc\text{-}le = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

DEFINITION: $cc\text{-}t = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

DEFINITION: $cc\text{-}f = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Let us define the theory *condition-code-theory* to consist of the following events: $cc\text{-}cc$, $cc\text{-}cs$, $cc\text{-}vc$, $cc\text{-}vs$, $cc\text{-}pl$, $cc\text{-}mi$, $cc\text{-}ne$, $cc\text{-}eq$, $cc\text{-}hi$, $cc\text{-}ls$, $cc\text{-}ge$, $cc\text{-}lt$, $cc\text{-}gt$, $cc\text{-}le$, $cc\text{-}t$, $cc\text{-}f$.

DEFINITION: $op\text{-}move = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

DEFINITION: $op\text{-}inc = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$

DEFINITION: $op\text{-}addc = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

DEFINITION: $\text{OP-ADD} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$

DEFINITION: $\text{OP-NEG} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

DEFINITION: $\text{OP-DEC} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f})$

DEFINITION: $\text{OP-SUBB} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

DEFINITION: $\text{OP-SUB} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f})$

DEFINITION: $\text{OP-ROR} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

DEFINITION: $\text{OP-ASR} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t})$

DEFINITION: $\text{OP-LSR} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t})$

DEFINITION: $\text{OP-XOR} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t})$

DEFINITION: $\text{OP-OR} = \text{list}(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

DEFINITION: $\text{OP-AND} = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t})$

DEFINITION: $\text{OP-NOT} = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

DEFINITION: $\text{OP-M15} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t})$

EVENT: Let us define the theory *opcode-theory* to consist of the following events:
 op-move, op-inc, op-addc, op-add, op-neg, op-dec, op-subb, op-sub, op-ror, op-
 asr, op-lsr, op-xor, op-or, op-and, op-not, op-m15.

THEOREM: zb-v-alu

$$\text{zb}(\text{v-alu}(c, a, b, \text{op-code})) = \text{v-zerop}(\text{bv}(\text{v-alu}(c, a, b, \text{op-code})))$$

THEOREM: n-v-alu

$$\text{n}(\text{v-alu}(c, a, b, \text{op-code})) = \text{v-negp}(\text{bv}(\text{v-alu}(c, a, b, \text{op-code})))$$

THEOREM: universal-flags-interpretations

$$\begin{aligned} & (\text{bvp}(a) \wedge \text{boolp}(c) \wedge (\text{length}(a) \neq 0)) \\ & \rightarrow ((\text{z-set}(\text{set-flags}) \\ & \quad \rightarrow ((\text{store-resultp}(\text{CC-NE}, \\ & \quad \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \quad \text{set-flags}, \\ & \quad \quad \quad \text{v-alu}(c, a, b, \text{op-code})))) \\ & \quad = \text{v-nzerop}(\text{bv}(\text{v-alu}(c, a, b, \text{op-code})))) \\ & \wedge (\text{store-resultp}(\text{CC-EQ}, \\ & \quad \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \quad \text{set-flags}, \end{aligned}$$

$$\begin{aligned}
&= (\neg \text{v-nzerop} (\text{bv} (\text{v-alu} (c, a, b, \text{op-code})))) \\
\wedge \quad &(\text{n-set} (\text{set-flags})) \\
&\rightarrow ((\text{store-resultp} (\text{CC-PL}, \\
&\quad \text{update-flags} (\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu} (c, a, b, \text{op-code})))) \\
&= (\neg \text{v-negp} (\text{bv} (\text{v-alu} (c, a, b, \text{op-code})))) \\
\wedge \quad &(\text{store-resultp} (\text{CC-MI}, \\
&\quad \text{update-flags} (\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu} (c, \\
&\quad \quad \quad a, \\
&\quad \quad \quad b, \\
&\quad \quad \quad \text{op-code}))) \\
&= \text{v-negp} (\text{bv} (\text{v-alu} (c, a, b, \text{op-code})))) \\
\wedge \quad &(\text{store-resultp} (\text{CC-T}, \\
&\quad \text{update-flags} (\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu} (c, a, b, \text{op-code}))) \\
&= \mathbf{t}) \\
\wedge \quad &(\text{store-resultp} (\text{CC-F}, \\
&\quad \text{update-flags} (\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu} (c, a, b, \text{op-code}))) \\
&= \mathbf{f}))
\end{aligned}$$

THEOREM: v-sum-on-not-a-a

$$\text{bv2p} (a, b) \rightarrow (\text{v-nzerop} (\text{v-sum} (\mathbf{t}, a, b)) = (\text{v-not} (a) \neq b))$$

THEOREM: if-v-subtractor-output=0-then-a=b

$$\text{bv2p} (a, b) \rightarrow (\text{v-nzerop} (\text{v-subtractor-output} (\mathbf{f}, a, b)) = (a \neq b))$$

THEOREM: zero-flag-interpretation-for-sub

$$\begin{aligned}
&(\text{bv2p} (a, b) \wedge \text{z-set} (\text{set-flags})) \\
&\rightarrow (\text{z-flag} (\text{update-flags} (\text{flags}, \text{set-flags}, \text{v-alu} (\text{c-flag} (\text{flags}), a, b, \text{OP-SUB})))) \\
&= (a = b))
\end{aligned}$$

THEOREM: flags-interpretation-sub

$$\begin{aligned}
&(\text{bv2p} (a, b) \wedge \text{z-set} (\text{set-flags})) \\
&\rightarrow ((\text{store-resultp} (\text{CC-NE}, \\
&\quad \text{update-flags} (\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu} (\text{c-flag} (\text{flags}), a, b, \text{OP-SUB})))) \\
&= (a \neq b))
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ (store-resultp (CC-EQ,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-SUB))} \\
& = \text{ (a = b))}
\end{aligned}$$

THEOREM: v-negp-v-shift-right

$$((\text{length (a)} \neq 0) \wedge \text{boolp (c)}) \rightarrow (\text{v-negp (v-shift-right (a, c))} = c)$$

THEOREM: flags-interpretation-ror

$$\begin{aligned}
& (\text{bvp (a)} \wedge \text{boolp (c-flag (flags))} \wedge (\text{length (a)} \neq 0)) \\
& \rightarrow ((\text{c-set (set-flags)} \\
& \quad \rightarrow ((\text{store-resultp (CC-CC,} \\
& \quad \quad \text{update-flags (flags,} \\
& \quad \quad \quad \text{set-flags,} \\
& \quad \quad \quad \text{v-alu (c-flag (flags), a, b, OP-ROR))} \\
& = \text{ (}\neg \text{ nth (0, a))}) \\
& \wedge \text{ (store-resultp (CC-CS,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad \text{a,} \\
& \quad \quad \quad \text{b,} \\
& \quad \quad \quad \text{OP-ROR))}) \\
& = \text{ nth (0, a))}) \\
& \wedge \text{ (n-set (set-flags)} \\
& \quad \rightarrow ((\text{store-resultp (CC-PL,} \\
& \quad \quad \text{update-flags (flags,} \\
& \quad \quad \quad \text{set-flags,} \\
& \quad \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad \text{a,} \\
& \quad \quad \quad \text{b,} \\
& \quad \quad \quad \text{OP-ROR))}) \\
& = \text{ (}\neg \text{ c-flag (flags))}) \\
& \wedge \text{ (store-resultp (CC-MI,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad \text{a,} \\
& \quad \quad \quad \text{b,} \\
& \quad \quad \quad \text{OP-ROR))}) \\
& = \text{ c-flag (flags))})
\end{aligned}$$

THEOREM: flags-interpretation-lsr

$$(\text{bvp (a)} \wedge (\text{length (a)} \neq 0) \wedge \text{c-set (set-flags)})$$

$$\begin{aligned}
& \rightarrow ((\text{store-resultp}(\text{CC-CC}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
& = (\neg \text{nth}(0, a))) \\
& \wedge (\text{store-resultp}(\text{CC-CS}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
& = \text{nth}(0, a)))
\end{aligned}$$

THEOREM: flags-interpretation-xor

$$\begin{aligned}
& (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{z-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-NE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-XOR}))) \\
& = (a \neq b)) \\
& \wedge (\text{store-resultp}(\text{CC-EQ}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-XOR}))) \\
& = (a = b)))
\end{aligned}$$

THEOREM: flags-interpretation-nat-move

$$\begin{aligned}
& (\text{bv2p}(a, b) \wedge \text{z-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-NE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-MOVE}))) \\
& = (\text{v-to-nat}(a) \neq 0)) \\
& \wedge (\text{store-resultp}(\text{CC-EQ}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-MOVE}))) \\
& = (\text{v-to-nat}(a) = 0)) \\
& \wedge (\text{store-resultp}(\text{CC-NE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-M15}))) \\
& = (\text{v-to-nat}(a) \neq 0)) \\
& \wedge (\text{store-resultp}(\text{CC-EQ}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags},
\end{aligned}$$

$$= \text{v-to-nat}(a) = 0))) \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-M15}))$$

THEOREM: flags-interpretation-nat-inc

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge \text{c-set}(set\text{-}flags)) \\ \rightarrow & ((\text{store-resultp}(\text{CC-CC}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-INC})))) \\ = & ((1 + \text{v-to-nat}(a)) < \exp(2, \text{length}(a))) \\ \wedge & (\text{store-resultp}(\text{CC-CS}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-INC})))) \\ = & ((1 + \text{v-to-nat}(a)) \not< \exp(2, \text{length}(a)))) \end{aligned}$$

THEOREM: flags-interpretation-nat-addc

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge \text{c-set}(set\text{-}flags)) \\ \rightarrow & ((\text{store-resultp}(\text{CC-CC}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-ADDC})))) \\ = & ((\text{b-to-nat}(\text{c-flag}(flags)) + \text{v-to-nat}(a) + \text{v-to-nat}(b)) \\ & \quad < \exp(2, \text{length}(a))) \\ \wedge & (\text{store-resultp}(\text{CC-CS}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-ADDC})))) \\ = & ((\text{b-to-nat}(\text{c-flag}(flags)) \\ & \quad + \text{v-to-nat}(a) \\ & \quad + \text{v-to-nat}(b)) \\ & \quad \not< \exp(2, \text{length}(a)))) \end{aligned}$$

THEOREM: flags-interpretation-nat-add

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge \text{c-set}(set\text{-}flags)) \\ \rightarrow & ((\text{store-resultp}(\text{CC-CC}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-ADD})))) \\ = & ((\text{v-to-nat}(a) + \text{v-to-nat}(b)) < \exp(2, \text{length}(a))) \\ \wedge & (\text{store-resultp}(\text{CC-CS}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\text{-}flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-ADD})))) \\ = & ((\text{v-to-nat}(a) + \text{v-to-nat}(b)) \not< \exp(2, \text{length}(a)))) \end{aligned}$$

THEOREM: when-the-mantissa-isnt-0-then-neither-is-the-exponential
 $(mantissa \neq 0) \rightarrow (\exp(mantissa, exponent) \neq 0)$

DEFINITION:

double-difference-induction (n, m, l)

= **if** $n \simeq 0$ **then** **t**
 elseif $m \simeq 0$ **then** **t**
 elseif $l \simeq 0$ **then** **t**
 else double-difference-induction $(n - l, m - l, l)$ **endif**

THEOREM: quotient-lessp-double-remainder\$help

$((n < m)$
 $\wedge (n \neq 0)$
 $\wedge (m \neq 0)$
 $\wedge (l \neq 0)$
 $\wedge ((n \bmod l) = 0)$
 $\wedge ((m \bmod l) = 0))$
 $\rightarrow ((n \div l) < (m \div l))$

THEOREM: v-nzerop-nat-to-v\$help

$((l \neq 0) \wedge (n \neq 0) \wedge ((n \bmod 2) = 0) \wedge (n < \exp(2, l)))$
 $\rightarrow ((n \div 2) < \exp(2, l - 1))$

EVENT: Enable v-nzerop-nat-to-v\$help; name this event ‘v-nzerop-nat-to-v\$help-off’.

THEOREM: v-nzerop-nat-to-v

$(n < \exp(2, l)) \rightarrow (\text{v-nzerop}(\text{nat-to-v}(n, l)) = (n \neq 0))$

THEOREM: lessp-sub1-plus-a-b-b

$((a + b) - 1 < b) = ((a \simeq 0) \wedge (b \neq 0))$

THEOREM: lessp-v-to-nat-exp-2-length-a

$(l = \text{length}(a)) \rightarrow ((\text{v-to-nat}(a) < \exp(2, l)) = \mathbf{t})$

THEOREM: lessp-v-to-nat-exp-2-length-a\$linear

$\text{v-to-nat}(a) < \exp(2, \text{length}(a))$

THEOREM: flags-interpretation-nat-dec

$\text{bv2p}(a, b)$
 $\rightarrow ((\text{c-set}(\text{set-flags})$
 $\rightarrow ((\text{store-resultp}(\text{CC-CC},$
 $\text{update-flags}(\text{flags},$
 $\text{set-flags},$
 $\text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC})))$

$$\begin{aligned}
&= (\text{v-to-nat}(a) \neq 0)) \\
\wedge \quad &(\text{store-resultp}(\text{CC-CS}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
&\quad \quad \quad a, \\
&\quad \quad \quad b, \\
&\quad \quad \quad \text{OP-DEC}))) \\
&= (\text{v-to-nat}(a) = 0))) \\
\wedge \quad &((\text{z-set}(\text{set-flags}) \wedge (\text{length}(a) \neq 0)) \\
&\rightarrow ((\text{store-resultp}(\text{CC-NE}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
&\quad \quad \quad a, \\
&\quad \quad \quad b, \\
&\quad \quad \quad \text{OP-DEC})))) \\
&= (\text{v-to-nat}(a) \neq 1)) \\
\wedge \quad &(\text{store-resultp}(\text{CC-EQ}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
&\quad \quad \quad a, \\
&\quad \quad \quad b, \\
&\quad \quad \quad \text{OP-DEC})))) \\
&= (\text{v-to-nat}(a) = 1))))
\end{aligned}$$

THEOREM: remainder-theorem-for-subb

$$((a < b) \wedge (b < c)) \rightarrow (((b + c) - a) \bmod c) = (b - a))$$

THEOREM: difference-linear

$$(a < b) \rightarrow (((a - c) < b) = \mathbf{t})$$

THEOREM: flags-interpretation-nat-subb-carry

$$\begin{aligned}
&(\text{bv2p}(a, b) \wedge \text{c-set}(\text{set-flags})) \\
&\rightarrow ((\text{store-resultp}(\text{CC-CC}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUBB}))) \\
&= (\text{v-to-nat}(b) \not< (\text{v-to-nat}(a) + \text{b-to-nat}(\text{c-flag}(\text{flags})))) \\
\wedge \quad &(\text{store-resultp}(\text{CC-CS}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUBB}))) \\
&= (\text{v-to-nat}(b)
\end{aligned}$$

$$< (\text{v-to-nat}(a) + \text{b-to-nat}(\text{c-flag}(flags))))))$$

THEOREM: flags-interpretation-nat-subb-higher

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge \text{c-set}(set\_flags) \wedge \text{z-set}(set\_flags)) \\ \rightarrow & ((\text{store-resultp}(\text{CC-HI}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\_flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-SUBB}))) \\ = & ((\text{v-to-nat}(a) + \text{b-to-nat}(\text{c-flag}(flags))) < \text{v-to-nat}(b))) \\ \wedge & (\text{store-resultp}(\text{CC-LS}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\_flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-SUBB}))) \\ = & ((\text{v-to-nat}(a) + \text{b-to-nat}(\text{c-flag}(flags))) \\ & \quad \not< \text{v-to-nat}(b)))) \end{aligned}$$

THEOREM: carry-flag-nat-interpretation-for-sub

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge \text{c-set}(set\_flags)) \\ \rightarrow & (\text{c-flag}(\text{update-flags}(flags, set\_flags, \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-SUB}))) \\ = & (\text{v-to-nat}(b) < \text{v-to-nat}(a))) \end{aligned}$$

THEOREM: flags-interpretation-nat-sub

$$\begin{aligned} & \text{bv2p}(a, b) \\ \rightarrow & ((\text{c-set}(set\_flags) \\ & \quad \rightarrow ((\text{store-resultp}(\text{CC-CC}, \\ & \quad \quad \text{update-flags}(flags, \\ & \quad \quad \quad set\_flags, \\ & \quad \quad \quad \text{v-alu}(\text{c-flag}(flags), a, b, \text{OP-SUB}))) \\ = & (\text{v-to-nat}(b) \not< \text{v-to-nat}(a)))) \\ \wedge & (\text{store-resultp}(\text{CC-CS}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\_flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), \\ & \quad \quad \quad a, \\ & \quad \quad \quad b, \\ & \quad \quad \quad \text{OP-SUB}))) \\ = & (\text{v-to-nat}(b) < \text{v-to-nat}(a)))) \\ \wedge & ((\text{c-set}(set\_flags) \wedge \text{z-set}(set\_flags)) \\ \rightarrow & ((\text{store-resultp}(\text{CC-HI}, \\ & \quad \text{update-flags}(flags, \\ & \quad \quad set\_flags, \\ & \quad \quad \text{v-alu}(\text{c-flag}(flags), \\ & \quad \quad \quad a, \\ & \quad \quad \quad b, \\ & \quad \quad \quad \text{OP-SUB})))) \end{aligned}$$

$$\begin{aligned}
&= (\text{v-to-nat}(a) < \text{v-to-nat}(b)) \\
\wedge \quad &(\text{store-resultp}(\text{CC-LS}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
&\quad \quad \quad a, \\
&\quad \quad \quad b, \\
&\quad \quad \quad \text{OP-SUB}))) \\
&= (\text{v-to-nat}(a) \not< \text{v-to-nat}(b))
\end{aligned}$$

THEOREM: flags-interpretation-nat-lsr-carry

$$\begin{aligned}
&(\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{c-set}(\text{set-flags})) \\
\rightarrow \quad &((\text{store-resultp}(\text{CC-CC}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
&= ((\text{v-to-nat}(a) \bmod 2) = 0)) \\
\wedge \quad &(\text{store-resultp}(\text{CC-CS}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
&= ((\text{v-to-nat}(a) \bmod 2) \neq 0))
\end{aligned}$$

THEOREM: flags-interpretation-nat-lsr-zero

$$\begin{aligned}
&(\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{z-set}(\text{set-flags})) \\
\rightarrow \quad &((\text{store-resultp}(\text{CC-NE}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
&= ((\text{v-to-nat}(a) \div 2) \neq 0)) \\
\wedge \quad &(\text{store-resultp}(\text{CC-EQ}, \\
&\quad \text{update-flags}(\text{flags}, \\
&\quad \quad \text{set-flags}, \\
&\quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-LSR}))) \\
&= ((\text{v-to-nat}(a) \div 2) = 0))
\end{aligned}$$

THEOREM: idifference-0

$$(\text{integerp}(a) \wedge \text{integerp}(b)) \rightarrow ((\text{idifference}(a, b) = 0) = (a = b))$$

THEOREM: v-to-nat-lessp-exp-2-length

$$\text{v-to-nat}(a) < \exp(2, \text{length}(a))$$

THEOREM: integerp-v-to-int

$$\text{integerp}(\text{v-to-int}(a))$$

THEOREM: v-negp-as-bounds

$$\text{bvp}(x) \rightarrow (\text{v-negp}(x) = (\text{v-to-nat}(x) \not\leq \exp(2, \text{length}(x) - 1)))$$

EVENT: Enable v-negp-as-bounds; name this event ‘v-negp-as-bounds-off’.

THEOREM: negativep-idifference-on-numberps

$$((a \in \mathbf{N}) \wedge (b \in \mathbf{N})) \rightarrow (\text{negativep}(\text{idifference}(a, b)) = (a < b))$$

THEOREM: v-negp->negativep-v-to-int

$$\text{bvp}(a) \rightarrow (\text{v-negp}(a) = \text{negativep}(\text{v-to-int}(a)))$$

EVENT: Enable v-negp->negativep-v-to-int; name this event ‘v-negp->negativep-v-to-int-off’.

THEOREM: rewrite-equal-v-to-int-0

$$(\text{v-to-int}(a) = 0) = \text{v-zerop}(a)$$

EVENT: Enable rewrite-equal-v-to-int-0; name this event ‘rewrite-equal-v-to-int-0-off’.

THEOREM: int-to-v-v-to-int-0

$$\text{v-to-int}(\text{int-to-v}(0, l)) = 0$$

THEOREM: bvp-int-to-v

$$\text{bvp}(\text{int-to-v}(i, l))$$

THEOREM: v-to-int-nat-to-v-0

$$\text{v-to-int}(\text{nat-to-v}(0, l)) = 0$$

THEOREM: integer-in-rangep-minus-0

$$\text{integer-in-rangep}(-0, l)$$

THEOREM: not-equal-iplus-minus-0

$$\text{iplus}(a, b) \neq (-0)$$

THEOREM: lessp-1-plus-x-x

$$(1 < (x + x)) = (x \not\leq 0)$$

THEOREM: integer-in-rangep-1

$$((l \not\leq 0) \wedge (l \neq 1)) \rightarrow \text{integer-in-rangep}(1, l)$$

THEOREM: lessp-a-plus-a-a

$$(a < (a + a)) = (a \not\leq 0)$$

THEOREM: exp-linear-bounds

(if $v < \exp(2, n - 1)$ then **f**
 else **t** endif
 $\wedge (v < \exp(2, n))$
 $\rightarrow (\exp(2, n - 1) \not< (\exp(2, n) - v))$

THEOREM: integer-in-range-v-to-int

$(l = \text{length}(a)) \rightarrow \text{integer-in-range}(\text{v-to-int}(a), l)$

THEOREM: addition-cases-for-integer-in-range-lemmas

$(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{iplus}(a, b) < \exp(2, l))$

THEOREM: subtraction-cases-for-integer-in-range-lemmas

$(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{iplus}(b, \text{neg}(a)) < \exp(2, l))$

THEOREM: exp-2-l-1<=exp-2-l

$(l \neq 0) \rightarrow (\exp(2, l - 1) < \exp(2, l))$

THEOREM: lessp-remainder-theorem

$(n < m) \rightarrow ((n \bmod r) < m)$

THEOREM: integer-in-range-the-obvious-way

$((l \neq 0) \wedge \text{integerp}(i) \wedge \text{integer-in-range}(i, l))$
 $\rightarrow (\text{v-negp}(\text{int-to-v}(i, l)) = \text{negativep}(i))$

EVENT: Enable integer-in-range-the-obvious-way; name this event ‘integer-in-range-the-obvious-way-off’.

THEOREM: integer-in-range-iplus

$(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{integer-in-range}(\text{iplus}(a, b), l)$
 $= (\text{v-negp}(\text{int-to-v}(\text{iplus}(a, b), l)) = \text{negativep}(\text{iplus}(a, b))))$

EVENT: Enable integer-in-range-iplus; name this event ‘integer-in-range-iplus-off’.

THEOREM: integer-in-range-iplus\$commuted

$(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{negativep}(\text{iplus}(a, b))$
 $= (\text{integer-in-range}(\text{iplus}(a, b), l)$
 $\leftrightarrow \text{v-negp}(\text{int-to-v}(\text{iplus}(a, b), l))))$

THEOREM: integer-in-range-iplus-carry
 $(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{integer-in-range}(\text{iplus}(1, \text{iplus}(a, b)), l)$
 $= (\text{v-negp}(\text{int-to-v}(\text{iplus}(1, \text{iplus}(a, b))), l))$
 $= \text{negativep}(\text{iplus}(1, \text{iplus}(a, b))))$

EVENT: Enable integer-in-range-iplus-carry; name this event ‘integer-in-range-iplus-carry-off’.

THEOREM: integer-in-range-iplus-carry\$commuted
 $(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{negativep}(\text{iplus}(1, \text{iplus}(a, b))))$
 $= (\text{integer-in-range}(\text{iplus}(1, \text{iplus}(a, b)), l)$
 $\leftrightarrow \text{v-negp}(\text{int-to-v}(\text{iplus}(1, \text{iplus}(a, b))), l)))$

THEOREM: not-integerp-minus-zero
 $\text{integerp}(-x) = (x \neq 0)$

THEOREM: integer-in-range-iplus-ineg
 $(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{integer-in-range}(\text{iplus}(b, \text{ineg}(a)), l)$
 $= (\text{v-negp}(\text{int-to-v}(\text{iplus}(b, \text{ineg}(a))), l))$
 $= \text{negativep}(\text{iplus}(b, \text{ineg}(a))))$

EVENT: Enable integer-in-range-iplus-ineg; name this event ‘integer-in-range-iplus-ineg-off’.

THEOREM: integer-in-range-iplus-ineg\$commuted
 $(\text{integer-in-range}(a, l) \wedge \text{integer-in-range}(b, l) \wedge (l \neq 0))$
 $\rightarrow (\text{negativep}(\text{iplus}(b, \text{ineg}(a))))$
 $= (\text{integer-in-range}(\text{iplus}(b, \text{ineg}(a)), l)$
 $\leftrightarrow \text{v-negp}(\text{int-to-v}(\text{iplus}(b, \text{ineg}(a))), l)))$

THEOREM: integer-in-range-iplus-ineg-carry
 $(\text{integer-in-range}(a, l)$
 $\wedge \text{integer-in-range}(b, l)$
 $\wedge (l \neq 0)$
 $\wedge (l \neq 1))$
 $\rightarrow (\text{integer-in-range}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a))), l)$
 $= (\text{v-negp}(\text{int-to-v}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a))), l))$
 $= \text{negativep}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a))))$

EVENT: Enable integer-in-range-iplus-ineg-carry; name this event ‘integer-in-range-iplus-ineg-carry-off’.

THEOREM: integer-in-range-iplus-ineg-carry\$commuted

$$\begin{aligned} & (\text{integer-in-range}(a, l) \\ & \wedge \text{integer-in-range}(b, l) \\ & \wedge (l \neq 0) \\ & \wedge (l \neq 1)) \\ \rightarrow & (\text{negativep}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a)))) \\ & = (\text{v-negp}(\text{int-to-v}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a))), l)) \\ & \leftrightarrow \text{integer-in-range}(\text{iplus}(b, \text{ineg}(\text{iplus}(1, a))), l)) \end{aligned}$$

THEOREM: lessp-1-exp-2

$$(l \neq 0) \rightarrow (1 < \text{exp}(2, l))$$

THEOREM: v-nzerop-int-to-v

$$\begin{aligned} & (\text{integer-in-range}(n, l) \wedge (l \neq 0)) \\ \rightarrow & (\text{v-nzerop}(\text{int-to-v}(n, l)) = (\neg \text{izerop}(n))) \end{aligned}$$

THEOREM: izerop-inc-v-nzerop

$$\begin{aligned} & (\text{integer-in-range}(a, l) \wedge (l \neq 0)) \\ \rightarrow & (\text{v-nzerop}(\text{int-to-v}(\text{iplus}(1, a), l)) = (\neg \text{izerop}(\text{iplus}(1, a)))) \end{aligned}$$

THEOREM: v-nzerop-int-to-v-0

$$\text{izerop}(z) \rightarrow (\neg \text{v-nzerop}(\text{int-to-v}(z, l)))$$

THEOREM: integer-in-range-0

$$\text{izerop}(a) \rightarrow \text{integer-in-range}(a, l)$$

THEOREM: not-v-negp-nat-to-v-0

$$(a \simeq 0) \rightarrow (\neg \text{v-negp}(\text{nat-to-v}(a, l)))$$

THEOREM: length-int-to-v

$$\text{length}(\text{int-to-v}(i, l)) = \text{fix}(l)$$

THEOREM: integer-in-range-1

$$(l \neq 0) \rightarrow \text{integer-in-range}(-1, l)$$

THEOREM: flags-interpretation-int-move

$$\begin{aligned} & \text{bvp}(a) \\ \rightarrow & ((\text{v-set}(\text{set-flags}) \\ & \rightarrow ((\text{store-resultp}(\text{CC-VC}, \\ & \qquad \qquad \text{update-flags}(\text{flags}, \\ & \qquad \qquad \text{set-flags}, \\ & \qquad \qquad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-MOVE}))) \\ & = \text{integer-in-range}(\text{v-to-int}(a), \text{length}(a))) \\ \wedge & (\text{store-resultp}(\text{CC-VC}, \\ & \qquad \qquad \text{update-flags}(\text{flags}, \\ & \qquad \qquad \text{set-flags}, \end{aligned}$$

$$\begin{aligned}
& \text{v-alu (c-flag (flags),} \\
& \quad a, \\
& \quad b, \\
& \quad \text{OP-M15})) \\
& = \text{integer-in-range (v-to-int (a), length (a))} \\
\wedge \quad & \text{(store-resultp (CC-VS,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))} \\
& = (\neg \text{integer-in-range (v-to-int (a), length (a)))) \\
\wedge \quad & \text{(store-resultp (CC-VS,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))} \\
& = (\neg \text{integer-in-range (v-to-int (a), length (a))})) \\
\wedge \quad & \text{(z-set (set-flags)} \\
& \rightarrow ((\text{store-resultp (CC-NE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))} \\
& = (\text{v-to-int (a) } \neq 0)) \\
\wedge \quad & \text{(store-resultp (CC-EQ,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))} \\
& = (\text{v-to-int (a) } = 0)) \\
\wedge \quad & \text{(store-resultp (CC-NE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b,
\end{aligned}$$

$$\begin{aligned}
& \text{OP-M15})) \\
& = (\text{v-to-int}(a) \neq 0) \\
\wedge \quad & (\text{store-resultp}(\text{CC-EQ}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))) \\
& = \text{izerop}(\text{v-to-int}(a))) \\
\wedge \quad & (\text{n-set}(\text{set-flags}) \\
& \rightarrow ((\text{store-resultp}(\text{CC-PL}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))) \\
& = (\neg \text{negativep}(\text{v-to-int}(a))) \\
\wedge \quad & (\text{store-resultp}(\text{CC-MI}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))) \\
& = \text{negativep}(\text{v-to-int}(a))) \\
\wedge \quad & (\text{store-resultp}(\text{CC-PL}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))) \\
& = (\neg \text{negativep}(\text{v-to-int}(a))) \\
\wedge \quad & (\text{store-resultp}(\text{CC-MI}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))) \\
& = \text{negativep}(\text{v-to-int}(a))) \\
\wedge \quad & ((\text{v-set}(\text{set-flags}) \wedge \text{n-set}(\text{set-flags}))
\end{aligned}$$

$$\begin{aligned}
& \rightarrow ((\text{store-resultp}(\text{CC-GE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))) \\
& = (\neg \text{negativep}(\text{v-to-int}(a))) \\
& \wedge (\text{store-resultp}(\text{CC-LT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))) \\
& = \text{negativep}(\text{v-to-int}(a)) \\
& \wedge (\text{store-resultp}(\text{CC-GE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))) \\
& = (\neg \text{negativep}(\text{v-to-int}(a))) \\
& \wedge (\text{store-resultp}(\text{CC-LT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))) \\
& = \text{negativep}(\text{v-to-int}(a))) \\
& \wedge ((\text{v-set}(\text{set-flags}) \wedge \text{n-set}(\text{set-flags}) \wedge \text{z-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-GT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-MOVE}))) \\
& = \text{ilessp}(0, \text{v-to-int}(a)) \\
& \wedge (\text{store-resultp}(\text{CC-LE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags},
\end{aligned}$$

$$\begin{aligned}
& \text{v-alu (c-flag (flags),} \\
& \quad a, \\
& \quad b, \\
& \quad \text{OP-MOVE})) \\
& = \text{ileq (v-to-int (a), 0))} \\
\wedge \quad & \text{(store-resultp (CC-GT,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))} \\
& = \text{ilessp (0, v-to-int (a))} \\
\wedge \quad & \text{(store-resultp (CC-LE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags),} \\
& \quad \quad \quad a, \\
& \quad \quad \quad b, \\
& \quad \quad \quad \text{OP-M15}))} \\
& = \text{ileq (v-to-int (a), 0))})
\end{aligned}$$

THEOREM: flags-interpretation-int-inc-overflow
 $(\text{bvp (a)} \wedge (\text{length (a)} \neq 0) \wedge \text{v-set (set-flags)})$
 $\rightarrow ((\text{store-resultp (CC-VC,}$
 $\quad \text{update-flags (flags,}$
 $\quad \quad \text{set-flags,}$
 $\quad \quad \text{v-alu (c-flag (flags), a, b, OP-INC))})$
 $= \text{integer-in-range (iplus (v-to-int (a), 1), length (a))})$
 $\wedge (\text{store-resultp (CC-VS,}$
 $\quad \text{update-flags (flags,}$
 $\quad \quad \text{set-flags,}$
 $\quad \quad \text{v-alu (c-flag (flags), a, b, OP-INC))})$
 $= (\neg \text{integer-in-range (iplus (v-to-int (a), 1),}$
 $\quad \text{length (a))}))$

THEOREM: flags-interpretation-int-inc-zero
 $(\text{bvp (a)} \wedge (\text{length (a)} \neq 0) \wedge \text{z-set (set-flags)})$
 $\rightarrow ((\text{store-resultp (CC-NE,}$
 $\quad \text{update-flags (flags,}$
 $\quad \quad \text{set-flags,}$
 $\quad \quad \text{v-alu (c-flag (flags), a, b, OP-INC))})$
 $= (\text{iplus (v-to-int (a), 1)} \neq 0))$
 $\wedge (\text{store-resultp (CC-EQ,}$

$$\begin{aligned}
& \text{update-flags } (flags, \\
& \quad \text{set-flags}, \\
& \quad \text{v-alu } (c\text{-flag } (flags), a, b, \text{OP-INC}))) \\
= & \text{ (iplus (v-to-int } (a), 1) = 0))
\end{aligned}$$

THEOREM: flags-interpretation-int-inc-negative

$$\begin{aligned}
& (\text{bvp } (a) \\
& \wedge \text{ (length } (a) \neq 0) \\
& \wedge \text{ (length } (a) \neq 1) \\
& \wedge \text{ v-set } (set\text{-flags}) \\
& \wedge \text{ n-set } (set\text{-flags})) \\
\rightarrow & ((\text{store-resultp } (\text{CC-GE}, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, \text{OP-INC})))) \\
& = \text{ (}\neg \text{ negativep (iplus (v-to-int } (a), 1))) \\
& \wedge \text{ (store-resultp } (\text{CC-LT}, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, \text{OP-INC})))) \\
& = \text{ negativep (iplus (v-to-int } (a), 1))})
\end{aligned}$$

THEOREM: flags-interpretation-int-inc-gt-0

$$\begin{aligned}
& (\text{bvp } (a) \\
& \wedge \text{ (length } (a) \neq 0) \\
& \wedge \text{ (length } (a) \neq 1) \\
& \wedge \text{ n-set } (set\text{-flags}) \\
& \wedge \text{ v-set } (set\text{-flags}) \\
& \wedge \text{ z-set } (set\text{-flags})) \\
\rightarrow & ((\text{store-resultp } (\text{CC-GT}, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, \text{OP-INC})))) \\
& = \text{ ilessp } (0, \text{iplus } (1, \text{v-to-int } (a))) \\
& \wedge \text{ (store-resultp } (\text{CC-LE}, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, \text{OP-INC})))) \\
& = \text{ ileq } (\text{iplus } (1, \text{v-to-int } (a)), 0))
\end{aligned}$$

THEOREM: flags-interpretation-int-addc-overflow

$$\begin{aligned}
& (\text{bv2p } (a, b) \wedge (\text{length } (a) \neq 0) \wedge \text{ v-set } (set\text{-flags})) \\
\rightarrow & ((\text{store-resultp } (\text{CC-VC}, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags},
\end{aligned}$$

$$\begin{aligned}
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC})) \\
= & \text{integer-in-range}(\text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))), \\
& \text{length}(a))) \\
\wedge & (\text{store-resultp}(\text{CC-VS}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC}))) \\
= & (\neg \text{integer-in-range}(\text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \text{iplus}(\text{v-to-int}(a), \\
& \text{v-to-int}(b))), \\
& \text{length}(a))))
\end{aligned}$$

THEOREM: flags-interpretation-int-addc-negative

$$\begin{aligned}
& (\text{bv2p}(a, b) \\
& \wedge (\text{length}(a) \neq 0) \\
& \wedge (\text{length}(a) \neq 1) \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{n-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-LT}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC}))) \\
= & \text{negativep}(\text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)))) \\
\wedge & (\text{store-resultp}(\text{CC-GE}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC}))) \\
= & (\neg \text{negativep}(\text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))))))
\end{aligned}$$

THEOREM: flags-interpretation-int-addc-gt-0

$$\begin{aligned}
& (\text{bv2p}(a, b) \\
& \wedge (\text{length}(a) \neq 0) \\
& \wedge (\text{length}(a) \neq 1) \\
& \wedge \text{n-set}(\text{set-flags}) \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{z-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-GT}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC}))) \\
= & \text{ilessp}(0,
\end{aligned}$$

$$\begin{aligned}
& \text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \quad \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)))) \\
\wedge & \text{ (store-resultp}(\text{CC-LE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADDC}))) \\
= & \text{ ileq}(\text{iplus}(\text{b-to-nat}(\text{c-flag}(\text{flags})), \\
& \quad \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))), \\
& \quad 0)))
\end{aligned}$$

THEOREM: flags-interpretation-int-add-overflow

$$\begin{aligned}
& (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{v-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-VC}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADD}))) \\
= & \text{ integer-in-range}(\text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)), \text{length}(a))) \\
\wedge & (\text{store-resultp}(\text{CC-VS}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADD}))) \\
= & (\neg \text{integer-in-range}(\text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)), \\
& \quad \text{length}(a))))
\end{aligned}$$

THEOREM: flags-interpretation-int-add-negative

$$\begin{aligned}
& (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{v-set}(\text{set-flags}) \wedge \text{n-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-LT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADD}))) \\
= & \text{ negativep}(\text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)))) \\
\wedge & (\text{store-resultp}(\text{CC-GE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ADD}))) \\
= & (\neg \text{negativep}(\text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))))
\end{aligned}$$

THEOREM: flags-interpretation-int-add-gt-0

$$\begin{aligned}
& (\text{bv2p}(a, b) \\
& \wedge (\text{length}(a) \neq 0) \\
& \wedge (\text{length}(a) \neq 1) \\
& \wedge \text{n-set}(\text{set-flags}) \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{z-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-GT},
\end{aligned}$$

$$\begin{aligned}
& \text{update-flags } (flags, \\
& \quad \text{set-flags,} \\
& \quad \text{v-alu (c-flag } (flags), a, b, \text{OP-ADD})) \\
= & \text{ ilespp } (0, \text{iplus } (\text{v-to-int } (a), \text{v-to-int } (b))) \\
\wedge & (\text{store-resultp } (\text{CC-LE,} \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag } (flags), a, b, \text{OP-ADD}))) \\
= & \text{ ileq } (\text{iplus } (\text{v-to-int } (a), \text{v-to-int } (b)), 0))
\end{aligned}$$

THEOREM: int-neg-range-implications

$$\begin{aligned}
& ((\text{negativep } (\text{v-to-int } (a))) \\
& \wedge (\neg \text{integer-in-rangep } (\text{negative-guts } (\text{v-to-int } (a)), \text{length } (a)))) \\
\rightarrow & \text{ v-negp } (\text{int-to-v } (\text{negative-guts } (\text{v-to-int } (a)), \text{length } (a))) \\
\wedge & (((\neg \text{negativep } (\text{v-to-int } (a))) \\
& \wedge (\neg \text{integer-in-rangep } (- \text{v-to-int } (a), \text{length } (a)))) \\
& \rightarrow (\neg \text{v-negp } (\text{int-to-v } (- \text{v-to-int } (a), \text{length } (a))))) \\
\wedge & (((\neg \text{negativep } (\text{v-to-int } (a))) \\
& \wedge (\neg \text{integer-in-rangep } (- \text{v-to-int } (a), \text{length } (a)))) \\
& \rightarrow \text{v-nzerop } (\text{int-to-v } (- \text{v-to-int } (a), \text{length } (a))))
\end{aligned}$$

THEOREM: flags-interpretation-int-neg-overflow

$$\begin{aligned}
& (\text{bvp } (a) \wedge (\text{length } (a) \neq 0) \wedge \text{v-set } (\text{set-flags})) \\
\rightarrow & ((\text{store-resultp } (\text{CC-VC,} \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag } (flags), a, b, \text{OP-NEG}))) \\
= & \text{ integer-in-rangep } (\text{ineg } (\text{v-to-int } (a)), \text{length } (a))) \\
\wedge & (\text{store-resultp } (\text{CC-VS,} \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag } (flags), a, b, \text{OP-NEG}))) \\
= & (\neg \text{integer-in-rangep } (\text{ineg } (\text{v-to-int } (a)), \text{length } (a))))
\end{aligned}$$

THEOREM: flags-interpretation-int-neg-negative

$$\begin{aligned}
& (\text{bvp } (a) \\
& \wedge (\text{length } (a) \neq 0) \\
& \wedge (\text{length } (a) \neq 1) \\
& \wedge \text{v-set } (\text{set-flags}) \\
& \wedge \text{n-set } (\text{set-flags})) \\
\rightarrow & ((\text{store-resultp } (\text{CC-LT,} \\
& \quad \text{update-flags } (flags, \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag } (flags), a, b, \text{OP-NEG}))) \\
= & \text{ negativep } (\text{ineg } (\text{v-to-int } (a))))
\end{aligned}$$

$$\begin{aligned}
& \wedge \text{ (store-resultp (CC-GE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-NEG))} \\
& = \text{ (}\neg \text{ negativep (ineg (v-to-int (a))))}
\end{aligned}$$

THEOREM: flags-interpretation-int-neg-zero

$$\begin{aligned}
& (\text{bvp (a)} \wedge (\text{length (a)} \neq 0) \wedge \text{z-set (set-flags)}) \\
& \rightarrow ((\text{store-resultp (CC-NE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-NEG))} \\
& = \text{ (ineg (v-to-int (a))} \neq 0)) \\
& \wedge \text{ (store-resultp (CC-EQ,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-NEG))} \\
& = \text{ (ineg (v-to-int (a))} = 0))
\end{aligned}$$

THEOREM: flags-interpretation-int-neg-gt-0

$$\begin{aligned}
& (\text{bvp (a)} \\
& \wedge (\text{length (a)} \neq 0) \\
& \wedge (\text{length (a)} \neq 1) \\
& \wedge \text{n-set (set-flags)} \\
& \wedge \text{v-set (set-flags)} \\
& \wedge \text{z-set (set-flags)}) \\
& \rightarrow ((\text{store-resultp (CC-GT,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-NEG))} \\
& = \text{ ilessp (0, ineg (v-to-int (a)))} \\
& \wedge \text{ (store-resultp (CC-LE,} \\
& \quad \text{update-flags (flags,} \\
& \quad \quad \text{set-flags,} \\
& \quad \quad \text{v-alu (c-flag (flags), a, b, OP-NEG))} \\
& = \text{ ileq (ineg (v-to-int (a)), 0))}
\end{aligned}$$

THEOREM: int-dec-rangep-lemmas

$$\begin{aligned}
& ((\neg \text{integer-in-rangep (v-to-int (a) - 1, length (a))} \\
& \rightarrow \text{v-nzerop (int-to-v (v-to-int (a) - 1, length (a))})) \\
& \wedge (((\text{length (a)} \neq 1) \\
& \quad \wedge (\neg \text{integer-in-rangep (- (1 + negative-guts (v-to-int (a))),} \\
& \quad \quad \text{length (a))})) \\
& \rightarrow \text{v-nzerop (int-to-v (- (1 + negative-guts (v-to-int (a))),} \\
& \quad \quad \text{length (a))}))
\end{aligned}$$

$$\begin{aligned}
& \wedge \quad (((\neg \text{negativep}(\text{iplus}(-1, \text{v-to-int}(a)))) \\
& \quad \wedge \quad \text{v-nzerop}(\text{int-to-v}(\text{iplus}(-1, \text{v-to-int}(a)), \text{length}(a)))) \\
& \quad \rightarrow \quad (\text{iplus}(-1, \text{v-to-int}(a)) \neq 0)) \\
& \wedge \quad (((\text{length}(a) \neq 0) \\
& \quad \wedge \quad (\neg \text{negativep}(\text{iplus}(-1, \text{v-to-int}(a)))) \\
& \quad \wedge \quad (\neg \text{v-nzerop}(\text{int-to-v}(\text{iplus}(-1, \text{v-to-int}(a)), \text{length}(a)))) \\
& \quad \rightarrow \quad (\text{iplus}(-1, \text{v-to-int}(a)) = 0))
\end{aligned}$$

THEOREM: flags-interpretation-int-dec-overflow

$$\begin{aligned}
& (\text{bvp}(a) \wedge (\text{length}(a) \neq 0) \wedge \text{v-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-VC}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\
& = \text{integer-in-range}(\text{idifference}(\text{v-to-int}(a), 1), \text{length}(a))) \\
& \wedge (\text{store-resultp}(\text{CC-VS}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\
& = (\neg \text{integer-in-range}(\text{idifference}(\text{v-to-int}(a), 1), \\
& \quad \text{length}(a))))
\end{aligned}$$

THEOREM: flags-interpretation-int-dec-negative

$$\begin{aligned}
& (\text{bvp}(a) \wedge (\text{length}(a) \neq 0) \wedge \text{v-set}(\text{set-flags}) \wedge \text{n-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-LT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\
& = \text{negativep}(\text{idifference}(\text{v-to-int}(a), 1))) \\
& \wedge (\text{store-resultp}(\text{CC-GE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\
& = (\neg \text{negativep}(\text{idifference}(\text{v-to-int}(a), 1))))
\end{aligned}$$

THEOREM: flags-interpretation-int-dec-zero

$$\begin{aligned}
& (\text{bvp}(a) \wedge (\text{length}(a) \neq 0) \wedge (\text{length}(a) \neq 1) \wedge \text{z-set}(\text{set-flags})) \\
& \rightarrow ((\text{store-resultp}(\text{CC-NE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\
& = (\text{idifference}(\text{v-to-int}(a), 1) \neq 0)) \\
& \wedge (\text{store-resultp}(\text{CC-EQ}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags},
\end{aligned}$$

$$= \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC})) \\ = (\text{idifference}(\text{v-to-int}(a), 1) = 0))$$

THEOREM: flags-interpretation-int-dec-gt-0

$$\begin{aligned} & (\text{bvp}(a) \\ & \wedge (\text{length}(a) \neq 0) \\ & \wedge (\text{length}(a) \neq 1) \\ & \wedge \text{n-set}(\text{set-flags}) \\ & \wedge \text{v-set}(\text{set-flags}) \\ & \wedge \text{z-set}(\text{set-flags})) \\ \rightarrow & ((\text{store-resultp}(\text{CC-GT}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\ & = \text{ilessp}(0, \text{idifference}(\text{v-to-int}(a), 1))) \\ & \wedge (\text{store-resultp}(\text{CC-LE}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-DEC}))) \\ & = \text{ileq}(\text{idifference}(\text{v-to-int}(a), 1), 0))) \end{aligned}$$

THEOREM: flags-interpretation-int-subbb-overflow

$$\begin{aligned} & (\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0) \wedge \text{v-set}(\text{set-flags})) \\ \rightarrow & ((\text{store-resultp}(\text{CC-VC}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUBB}))) \\ & = \text{integer-in-range}(\text{idifference}(\text{v-to-int}(b), \\ & \quad \text{iplus}(\text{v-to-int}(a), \\ & \quad \quad \text{b-to-nat}(\text{c-flag}(\text{flags}))), \\ & \quad \text{length}(a))) \\ & \wedge (\text{store-resultp}(\text{CC-VS}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUBB}))) \\ & = (\neg \text{integer-in-range}(\text{idifference}(\text{v-to-int}(b), \\ & \quad \text{iplus}(\text{v-to-int}(a), \\ & \quad \quad \text{b-to-nat}(\text{c-flag}(\text{flags}))), \\ & \quad \text{length}(a)))))) \end{aligned}$$

THEOREM: flags-interpretation-int-subbb-negative

$$\begin{aligned} & (\text{bv2p}(a, b) \\ & \wedge (\text{length}(a) \neq 0) \\ & \wedge (\text{length}(a) \neq 1) \\ & \wedge \text{v-set}(\text{set-flags})) \end{aligned}$$

$$\begin{aligned}
& \wedge \text{ n-set } (set\text{-}flags)) \\
\rightarrow & ((\text{store-resultp } (CC\text{-}LT, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad set\text{-}flags, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, OP\text{-}SUBB))) \\
& = \text{ negativep } (\text{idifference } (\text{v-to-int } (b), \\
& \quad \text{iplus } (\text{v-to-int } (a), \\
& \quad \quad \text{b-to-nat } (c\text{-flag } (flags)))))) \\
& \wedge (\text{store-resultp } (CC\text{-}GE, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad set\text{-}flags, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, OP\text{-}SUBB))) \\
& = (\neg \text{ negativep } (\text{idifference } (\text{v-to-int } (b), \\
& \quad \text{iplus } (\text{v-to-int } (a), \\
& \quad \quad \text{b-to-nat } (c\text{-flag } (flags))))))
\end{aligned}$$

THEOREM: flags-interpretation-int-subb-gt-0

$$\begin{aligned}
& (\text{bv2p } (a, b) \\
& \wedge (\text{length } (a) \neq 0) \\
& \wedge (\text{length } (a) \neq 1) \\
& \wedge \text{ n-set } (set\text{-}flags) \\
& \wedge \text{ v-set } (set\text{-}flags) \\
& \wedge \text{ z-set } (set\text{-}flags)) \\
\rightarrow & ((\text{store-resultp } (CC\text{-}GT, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad set\text{-}flags, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, OP\text{-}SUBB))) \\
& = \text{ ilessp } (0, \\
& \quad \text{idifference } (\text{v-to-int } (b), \\
& \quad \quad \text{iplus } (\text{v-to-int } (a), \text{b-to-nat } (c\text{-flag } (flags)))))) \\
& \wedge (\text{store-resultp } (CC\text{-}LE, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad set\text{-}flags, \\
& \quad \quad \text{v-alu } (c\text{-flag } (flags), a, b, OP\text{-}SUBB))) \\
& = \text{ ileq } (\text{idifference } (\text{v-to-int } (b), \\
& \quad \text{iplus } (\text{v-to-int } (a), \\
& \quad \quad \text{b-to-nat } (c\text{-flag } (flags))))), \\
& \quad 0)))
\end{aligned}$$

THEOREM: flags-interpretation-int-sub-overflow

$$\begin{aligned}
& (\text{bv2p } (a, b) \wedge (\text{length } (a) \neq 0) \wedge \text{ v-set } (set\text{-}flags)) \\
\rightarrow & ((\text{store-resultp } (CC\text{-}VC, \\
& \quad \text{update-flags } (flags, \\
& \quad \quad set\text{-}flags,
\end{aligned}$$

$$\begin{aligned}
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB})) \\
= & \text{integer-in-range}(\text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)), \\
& \text{length}(a))) \\
\wedge & (\text{store-resultp}(\text{CC-VS}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB}))) \\
= & (\neg \text{integer-in-range}(\text{idifference}(\text{v-to-int}(b), \\
& \text{v-to-int}(a)), \\
& \text{length}(a))))))
\end{aligned}$$

THEOREM: flags-interpretation-int-sub-negative

$$\begin{aligned}
& (\text{bv2p}(a, b) \\
& \wedge (\text{length}(a) \neq 0) \\
& \wedge (\text{length}(a) \neq 1) \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{n-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-LT}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB}))) \\
= & \text{negativep}(\text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)))) \\
\wedge & (\text{store-resultp}(\text{CC-GE}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB}))) \\
= & (\neg \text{negativep}(\text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)))))
\end{aligned}$$

THEOREM: flags-interpretation-int-sub-gt-0

$$\begin{aligned}
& (\text{bv2p}(a, b) \\
& \wedge (\text{length}(a) \neq 0) \\
& \wedge (\text{length}(a) \neq 1) \\
& \wedge \text{n-set}(\text{set-flags}) \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{z-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-GT}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB}))) \\
= & \text{ilessp}(0, \text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)))) \\
\wedge & (\text{store-resultp}(\text{CC-LE}, \\
& \text{update-flags}(\text{flags}, \\
& \text{set-flags}, \\
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-SUB}))) \\
= & \text{ileq}(\text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)), 0)))
\end{aligned}$$

THEOREM: equal-idiv-2-0

$$(\text{idiv}(i, 2) = 0) = (\text{izerop}(i) \vee (i = 1))$$

THEOREM: integer-in-range-idiv

$$\text{integer-in-range}(i, l) \rightarrow \text{integer-in-range}(\text{idiv}(i, 2), l)$$

THEOREM: flags-interpretation-int-asr-zero

$$\begin{aligned} & (\text{bvp}(a) \wedge (\text{length}(a) \neq 0) \wedge \text{z-set}(\text{set-flags})) \\ \rightarrow & ((\text{store-resultp}(\text{CC-NE}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\ = & (\text{idiv}(\text{v-to-int}(a), 2) \neq 0)) \\ \wedge & (\text{store-resultp}(\text{CC-EQ}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \\ & \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\ = & (\text{idiv}(\text{v-to-int}(a), 2) = 0))) \end{aligned}$$

THEOREM: equal-sub1-plus-x-x

$$(((x + x) - 1) = 0) = (x \simeq 0)$$

THEOREM: remainder-difference-plus-x-x-plus-y-y

$$(((x + x) - (y + y)) \mathbf{mod} 2) = 0$$

THEOREM: remainder-difference-plus-x-x-add1-plus-y-y

$$\begin{aligned} & ((1 + (y + y)) < (x + x)) \\ \rightarrow & (((x + x) - (1 + (y + y))) \mathbf{mod} 2) = 1 \end{aligned}$$

THEOREM: quotient-difference-plus-x-x-plus-y-y-2

$$(((x + x) - (y + y)) \div 2) = (x - y)$$

THEOREM: quotient-difference-plus-x-x-add1-plus-y-y-2

$$(((x + x) - (1 + (y + y))) \div 2) = ((x - y) - 1)$$

THEOREM: quotient-add1-plus-x-x-2

$$((1 + (x + x)) \div 2) = \text{fix}(x)$$

THEOREM: imod-v-to-int-2

$$\text{listp}(a) \rightarrow (\text{imod}(\text{v-to-int}(a), 2) = \text{b-to-nat}(\text{car}(a)))$$

THEOREM: flags-interpretation-int-asr-carry

$$\begin{aligned} & (\text{bvp}(a) \wedge (\text{length}(a) \neq 0) \wedge \text{c-set}(\text{set-flags})) \\ \rightarrow & ((\text{store-resultp}(\text{CC-CC}, \\ & \quad \text{update-flags}(\text{flags}, \\ & \quad \quad \text{set-flags}, \end{aligned}$$

$$\begin{aligned}
& \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR})) \\
= & (\text{imod}(\text{v-to-int}(a), 2) = 0) \\
\wedge & (\text{store-resultp}(\text{CC-CS}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
= & (\text{imod}(\text{v-to-int}(a), 2) \neq 0))
\end{aligned}$$

THEOREM: flags-interpretation-int-asr-negative

$$\begin{aligned}
& (\text{bvp}(a) \wedge (\text{length}(a) \neq 0)) \\
\rightarrow & ((\text{n-set}(\text{set-flags}) \\
& \quad \rightarrow ((\text{store-resultp}(\text{CC-PL}, \\
& \quad \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \quad \text{set-flags}, \\
& \quad \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
& \quad = (\neg \text{negativep}(\text{idiv}(\text{v-to-int}(a), 2)))) \\
& \quad \wedge (\text{store-resultp}(\text{CC-MI}, \\
& \quad \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \quad \text{set-flags}, \\
& \quad \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad \quad a, \\
& \quad \quad \quad \quad b, \\
& \quad \quad \quad \quad \text{OP-ASR}))) \\
& \quad = \text{negativep}(\text{idiv}(\text{v-to-int}(a), 2)))) \\
& \quad \wedge ((\text{n-set}(\text{set-flags}) \wedge \text{v-set}(\text{set-flags})) \\
& \quad \rightarrow ((\text{store-resultp}(\text{CC-GE}, \\
& \quad \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \quad \text{set-flags}, \\
& \quad \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad \quad a, \\
& \quad \quad \quad \quad b, \\
& \quad \quad \quad \quad \text{OP-ASR}))) \\
& \quad = (\neg \text{negativep}(\text{idiv}(\text{v-to-int}(a), 2)))) \\
& \quad \wedge (\text{store-resultp}(\text{CC-LT}, \\
& \quad \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \quad \text{set-flags}, \\
& \quad \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), \\
& \quad \quad \quad \quad a, \\
& \quad \quad \quad \quad b, \\
& \quad \quad \quad \quad \text{OP-ASR}))) \\
& \quad = \text{negativep}(\text{idiv}(\text{v-to-int}(a), 2))))))
\end{aligned}$$

THEOREM: flags-interpretation-int-asr-gt-0

$$(\text{bvp}(a))$$

$$\begin{aligned}
& \wedge \text{length}(a) \neq 0 \\
& \wedge \text{v-set}(\text{set-flags}) \\
& \wedge \text{n-set}(\text{set-flags}) \\
& \wedge \text{z-set}(\text{set-flags}) \\
\rightarrow & ((\text{store-resultp}(\text{CC-GT}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
& = \text{ilessp}(0, \text{idiv}(\text{v-to-int}(a), 2))) \\
& \wedge (\text{store-resultp}(\text{CC-LE}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
& = \text{ileq}(\text{idiv}(\text{v-to-int}(a), 2), 0)))
\end{aligned}$$

THEOREM: flags-interpretation-int-asr-overflow

$$\begin{aligned}
& (\text{bvp}(a) \wedge \text{v-set}(\text{set-flags})) \\
\rightarrow & ((\text{store-resultp}(\text{CC-VC}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
& = \text{integer-in-range}(\text{idiv}(\text{v-to-int}(a), 2), \text{length}(a))) \\
& \wedge (\text{store-resultp}(\text{CC-VS}, \\
& \quad \text{update-flags}(\text{flags}, \\
& \quad \quad \text{set-flags}, \\
& \quad \quad \text{v-alu}(\text{c-flag}(\text{flags}), a, b, \text{OP-ASR}))) \\
& = (\neg \text{integer-in-range}(\text{idiv}(\text{v-to-int}(a), 2), \text{length}(a))))
\end{aligned}$$

DEFINITION:

$$\text{v-neg}(a) = \text{v-subtractor-output}(\mathbf{f}, a, \text{nat-to-v}(0, \text{length}(a)))$$

THEOREM: bv-cvzbv

$$\text{bv}(\text{cvzbv}(c, v, bv)) = bv$$

THEOREM: c-cvzbv

$$c(\text{cvzbv}(c, v, bv)) = c$$

THEOREM: v-cvzbv

$$v(\text{cvzbv}(c, v, bv)) = v$$

THEOREM: zb-cvzbv

$$\text{zb}(\text{cvzbv}(c, v, bv)) = \text{v-zerop}(bv)$$

THEOREM: bv-v-alu

$$\text{bv}(\text{v-alu}(c, a, b, op))$$


```

=  if op = list (f, f, f, f) then v-buf(a)
    elseif op = list (t, f, f, f) then v-inc(a)
    elseif op = list (f, t, f, f) then v-adder-output(c, a, b)
    elseif op = list (t, t, f, f) then v-adder-output(f, a, b)
    elseif op = list (f, f, t, f) then v-neg(a)
    elseif op = list (t, f, t, f) then v-dec(a)
    elseif op = list (f, t, t, f) then v-subtractor-output(c, a, b)
    elseif op = list (t, t, t, f) then v-subtractor-output(f, a, b)
    elseif op = list (f, f, f, t) then v-ror(a, c)
    elseif op = list (t, f, f, t) then v-asr(a)
    elseif op = list (f, t, f, t) then v-lsr(a)
    elseif op = list (t, t, f, t) then v-xor(a, b)
    elseif op = list (f, f, t, t) then v-or(a, b)
    elseif op = list (t, f, t, t) then v-and(a, b)
    elseif op = list (f, t, t, t) then v-not(a)
    else v-buf(a) endif

```

THEOREM: c-v-alu

```

(((op = OP-MOVE)
  ∨ (op = OP-XOR)
  ∨ (op = OP-OR)
  ∨ (op = OP-AND)
  ∨ (op = OP-NOT)
  ∨ (op = OP-M15))
→ (c(v-alu(c, a, b, op)) = f))
∧ ((listp(a)
  ∧ ((op = OP-ROR) ∨ (op = OP-ASR) ∨ (op = OP-LSR)))
→ (c(v-alu(c, a, b, op)) = nth(0, a)))

```

THEOREM: v-v-alu

```

((op = OP-MOVE)
  ∨ (op = OP-ROR)
  ∨ (op = OP-ASR)
  ∨ (op = OP-ASR)
  ∨ (op = OP-LSR)
  ∨ (op = OP-XOR)
  ∨ (op = OP-OR)
  ∨ (op = OP-AND)
  ∨ (op = OP-NOT)
  ∨ (op = OP-M15))
→ (v(v-alu(c, a, b, op)) = f)

```

THEOREM: v-alu-correct-nat-rewrite

```

bv2p(a, b) → (v-alu(c, a, b, op) = v-alu-nat(c, a, b, op))

```

EVENT: Enable v-alu-correct-nat-rewrite; name this event ‘v-alu-correct-nat-rewrite-off’.

THEOREM: v-alu-correct-int-rewrite
 $(\text{bv2p}(a, b) \wedge (\text{length}(a) \neq 0))$
 $\rightarrow (\text{v-alu}(c, a, b, op) = \text{v-alu-int}(c, a, b, op))$

EVENT: Enable v-alu-correct-int-rewrite; name this event ‘v-alu-correct-int-rewrite-off’.

THEOREM: lessp-quotient-test
 $(n < m) \rightarrow (((n \div k) < m) = \mathbf{t})$

EVENT: Enable lessp-quotient-test; name this event ‘lessp-quotient-test-off’.

THEOREM: nat-to-v-as-remainder
 $\text{nat-to-v}(n \text{ mod } \exp(2, l), l) = \text{nat-to-v}(n, l)$

THEOREM: nat-to-v-of-v-to-nat*
 $(\text{bvp}(v) \wedge (l = \text{length}(v))) \rightarrow (\text{nat-to-v}(\text{v-to-nat}(v), l) = v)$

EVENT: Enable nat-to-v-of-v-to-nat*; name this event ‘nat-to-v-of-v-to-nat*-off’.

THEOREM: bv-v-alu-as-natural
 $(\text{bv2p}(a, b) \wedge (\text{length}(a) = 32))$
 $\rightarrow (\text{bv}(\text{v-alu}(c, a, b, op))$
 $= \text{if } op = \text{OP-INC} \text{ then nat-to-v}(1 + \text{v-to-nat}(a), 32)$
 $\text{elseif } op = \text{OP-ADDC}$
 $\text{then nat-to-v}(\text{b-to-nat}(c) + \text{v-to-nat}(a) + \text{v-to-nat}(b), 32)$
 $\text{elseif } op = \text{OP-ADD}$
 $\text{then nat-to-v}(\text{v-to-nat}(a) + \text{v-to-nat}(b), 32)$
 $\text{elseif } op = \text{OP-DEC}$
 $\text{then nat-to-v}((\text{v-to-nat}(a) + \exp(2, 32)) - 1, 32)$
 $\text{elseif } op = \text{OP-SUBB}$
 $\text{then nat-to-v}((\text{v-to-nat}(b) + \exp(2, 32))$
 $\quad - (\text{v-to-nat}(a) + \text{b-to-nat}(c)),$
 $\quad 32)$
 $\text{elseif } op = \text{OP-SUB}$
 $\text{then nat-to-v}((\text{v-to-nat}(b) + \exp(2, 32)) - \text{v-to-nat}(a),$
 $\quad 32)$
 $\text{elseif } op = \text{OP-LSR} \text{ then nat-to-v}(\text{v-to-nat}(a) \div 2, 32)$
 $\text{else bv}(\text{v-alu}(c, a, b, op)) \text{ endif})$

THEOREM: fix-int-int-to-v
 $\text{fix-int}(\text{v-to-int}(v)) = \text{v-to-int}(v)$

THEOREM: int-to-v-of-v-to-int
 $((l = \text{length}(v)) \wedge \text{bvp}(v)) \rightarrow (\text{int-to-v}(\text{v-to-int}(v), l) = v)$

THEOREM: ineg-as-idifference
 $\text{idifference}(0, i) = \text{ineg}(i)$

THEOREM: bv-v-alu-as-integer
 $(\text{bv2p}(a, b) \wedge (\text{length}(a) = 32))$
 $\rightarrow (\text{bv}(\text{v-alu}(c, a, b, op))$
 $=$
 $\quad \text{if } op = \text{OP-INC} \text{ then } \text{int-to-v}(\text{iplus}(1, \text{v-to-int}(a)), 32)$
 $\quad \text{elseif } op = \text{OP-ADDC}$
 $\quad \text{then } \text{int-to-v}(\text{iplus}(\text{b-to-nat}(c),$
 $\quad \quad \text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b))),$
 $\quad \quad 32)$
 $\quad \text{elseif } op = \text{OP-ADD}$
 $\quad \text{then } \text{int-to-v}(\text{iplus}(\text{v-to-int}(a), \text{v-to-int}(b)), 32)$
 $\quad \text{elseif } op = \text{OP-NEG} \text{ then } \text{int-to-v}(\text{ineg}(\text{v-to-int}(a)), 32)$
 $\quad \text{elseif } op = \text{OP-DEC}$
 $\quad \text{then } \text{int-to-v}(\text{idifference}(\text{v-to-int}(a), 1), 32)$
 $\quad \text{elseif } op = \text{OP-SUBB}$
 $\quad \text{then } \text{int-to-v}(\text{idifference}(\text{v-to-int}(b),$
 $\quad \quad \text{iplus}(\text{v-to-int}(a), \text{b-to-nat}(c))),$
 $\quad \quad 32)$
 $\quad \text{elseif } op = \text{OP-SUB}$
 $\quad \text{then } \text{int-to-v}(\text{idifference}(\text{v-to-int}(b), \text{v-to-int}(a)), 32)$
 $\quad \text{elseif } op = \text{OP-ASR} \text{ then } \text{int-to-v}(\text{idiv}(\text{v-to-int}(a), 2), 32)$
 $\quad \text{else } \text{bv}(\text{v-alu}(c, a, b, op)) \text{ endif})$

EVENT: Make the library "fm9001-replay" and compile it.

Index

- a-1+1=a, 444
- a-helpful-lemma-for-tree-inducti
ons, 146
- a-immediate, 541, 546–548, 761–763,
781, 782, 793, 794, 802–804,
811–813, 830, 848, 895, 904
- a-immediate-off, 542
- a-immediate-p, 541, 546, 548, 617,
624, 632, 633, 670, 673, 716,
724, 828, 830, 838, 848, 893,
895, 900, 901, 904, 915
- a-immediate-p-off, 542
- a-reg, 759, 761–763
- a-reg-off, 760
- add, 549
- add-composed-data, 345
- add-composed-data-off, 345
- add-delays, 322, 323
- add-delays-off, 323
- add-drives, 308, 309
- add-drives-off, 309
- add-in-type, 300, 303
- add-in-type-off, 300
- add-in-types, 300, 301
- add-in-types-off, 301
- add-load-delay, 323, 324
- add-load-delay-off, 323
- add-loading, 307–309
- add-loading-off, 308
- add-loading-simple, 269, 270
- add-loading-simple-off, 269
- add-loading-simples, 269, 275
- add-loading-simples-off, 269
- add-loadings, 308
- add-loadings-off, 308
- add-net-drive, 312
- add-net-drive-off, 312
- add-net-min-drive, 311, 312
- add-net-min-drive-off, 312
- add-new-type, 258, 259
- add-new-type-off, 259
- add-out-depends, 334, 335
- add-out-depends-off, 335
- add-out-type, 301
- add-out-type-off, 301
- add-out-types, 301
- add-out-types-off, 301
- add-ram-marker, 558
- add-synonyms, 337, 338
- add-synonyms-off, 337
- addable-input-delay, 326, 328, 331
- addable-input-delay-off, 327
- addition, 13
- addition-cases-for-integer-in-r
angep-lemmas, 1198
- addr-out, 759, 762–764
- addr-out-off, 760
- addressed-state, 367
- addressed-state-off, 367
- alist-entry, 247, 291, 345–347, 383,
384
- alist-entry-off, 247
- alistp, 246, 253, 254, 279, 283, 931–
933, 1002, 1003
- alistp-append, 1002
- alistp-append-off, 1040
- alistp-append-on, 1130
- alistp-dual-eval-1, 1002
- alistp-dual-eval-1-off, 1040
- alistp-dual-eval-1-on, 1129
- alistp-dual-eval-3, 1003
- alistp-dual-eval-3-off, 1039
- alistp-dual-eval-3-on, 1129
- alistp-off, 247
- alistp-opener, 1002
- alistp-opener-off, 1040
- alistp-opener-on, 1129
- alistp-pairlist, 1002
- alistp-pairlist-off, 1040
- alistp-pairlist-on, 1130
- all-bound-or-err, 252, 335
- all-bound-or-err-off, 252

- all-input-types, 356, 357, 376
- all-input-types-off, 357
- all-integer-defns, 37
- all-module-props, 295, 352, 353
- all-module-props-off, 295
- all-output-types, 357, 358, 376
- all-output-types-off, 357
- all-ramp-mem, 171, 174, 734, 779, 807, 810, 917, 926, 1018
- all-ramp-mem->ramp-mem, 174
- all-ramp-mem->ramp-mem1, 174
- all-ramp-mem-after-write-mem, 174
- all-ramp-mem-after-write-mem1, 174
- all-ramp-mem-after-write-regs, 734
- all-ramp-mem-constant-ram, 174
- all-ramp-mem-dual-port-ram-state-crock, 807
- all-ramp-mem-off, 171
- all-unbound-in-body, 427, 477, 480
- all-unbound-in-body-append, 427
- all-unbound-in-body-cons, 427
- all-unbound-in-body-dual-eval-1, 427
- all-unbound-in-body-listp, 427
- all-unbound-in-body-nlistp, 427
- all-unbound-in-body-nlistp-names, 427
- all-unbound-in-body-off, 427
- all-xs, 1017, 1018
- all-xs-approximates, 1017
- all-xs-approximates-off, 1029
- all-xs-approximates-on, 1118
- all-xs-make-list, 1017
- all-xs-make-list-off, 1029
- all-xs-make-list-on, 1118
- all-xs-off, 1029
- all-xs-on, 1118
- alu-c, 606, 608, 610, 612, 614, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 664, 666, 668, 761, 763, 781, 782, 784, 792–794, 796, 801–803, 811, 813
- alu-c-off, 606
- alu-cell, 501, 503
- alu-cell*, 501, 502
- alu-cell\$netlist, 502, 514
- alu-cell\$value, 502
- alu-cell\$value-off, 502
- alu-cell\$value-zero, 502
- alu-cell\$value-zero-off, 502
- alu-cell&, 502, 514
- alu-cell&-off, 502
- alu-dec-op, 486, 536, 566, 625, 635, 641, 642, 649, 653, 838, 850, 851, 859, 868, 869, 872, 901, 902, 904, 906, 907, 909, 911, 913
- alu-dec-op-off, 486
- alu-inc-op, 486, 487, 490, 493, 495, 496, 498, 536, 566, 608, 612, 625, 635, 641, 642, 649, 653, 654, 663–667, 838, 850, 851, 859, 868, 869, 872, 901, 902, 904, 906, 907, 909, 911, 913
- alu-inc-op-off, 486
- alu-mpg, 606, 608, 610, 612, 614, 616, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 669, 761, 763, 781, 782, 784, 792–794, 796, 801–803, 811, 813
- alu-mpg-off, 606
- alu-op, 606, 608, 610, 612, 614, 616, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 667, 668, 761, 763, 781, 782, 784, 792–794, 796, 801–803, 811, 813
- alu-op-off, 606
- alu-zero, 606, 608, 610, 612, 614, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637,

639, 641, 643, 645, 647, 649,
 651, 653, 655, 657, 659, 661,
 663, 664, 666, 668, 761, 763,
 781, 782, 784, 792–794, 796,
 801–803, 811, 813
 alu-zero-off, 606
 and*, 132, 182, 624, 634, 673, 716,
 724, 725, 829, 835, 838, 850,
 851, 894, 898–901, 904, 915
 and*-off, 132
 and-not-zero-p-tree, 16–19, 28, 29
 and-not-zero-p-tree-off, 125
 and-not-zero-p-tree-off1, 1177
 and-not-zero-p-tree-on, 1088
 another-way-to-look-at-boolp-nt
 h, 152
 any-of-nat-to-v-0-is-f, 160
 ao2, 155, 559
 ao2-monotone, 939
 ao2-monotone-off, 1053
 ao2-monotone-on, 1142
 ao2-preserved-good-s, 1004
 ao2-preserved-good-s-off, 1038
 ao2-preserved-good-s-on, 1128
 ao2\$netlist, 391, 560
 ao2\$value, 391
 ao2\$value-off, 391
 ao2&, 390, 391, 560
 ao2&-off, 390
 ao4, 156
 ao4-monotone, 939
 ao4-monotone-off, 1053
 ao4-monotone-on, 1142
 ao4-preserved-good-s, 1004
 ao4-preserved-good-s-off, 1038
 ao4-preserved-good-s-on, 1128
 ao4\$netlist, 391
 ao4\$value, 391
 ao4\$value-off, 391
 ao4&, 391
 ao4&-off, 391
 ao6, 156, 444
 ao6-monotone, 939
 ao6-monotone-off, 1053
 ao6-monotone-on, 1142
 ao6-preserved-good-s, 1004
 ao6-preserved-good-s-off, 1038
 ao6-preserved-good-s-on, 1127
 ao6\$netlist, 391, 445
 ao6\$value, 391
 ao6\$value-off, 391
 ao6&, 391, 445
 ao6&-off, 391
 ao7, 156
 ao7-monotone, 939
 ao7-monotone-off, 1053
 ao7-monotone-on, 1142
 ao7-preserved-good-s, 1004
 ao7-preserved-good-s-off, 1038
 ao7-preserved-good-s-on, 1127
 ao7\$netlist, 391
 ao7\$value, 391
 ao7\$value-off, 392
 ao7&, 391
 ao7&-off, 391
 append-cons, 135
 append-firstn-restn, 141
 append-list, 246
 append-list-off, 246
 append-nlistp, 135
 append-off, 135
 append-sum, 446
 append-v-threefix, 164
 append5, 135, 784, 796
 append6, 135, 705, 720
 append7, 135
 append8, 135, 705, 721
 apply-state-simple-okp, 268
 apply-state-simple-okp-off, 268
 arg-types-match-simplep, 264, 265
 arg-types-match-simplep-off, 264
 arg-types-matchp, 384, 385
 arg-types-matchp-off, 384
 arg-types-okp, 384
 arg-types-okp-off, 385
 arg-types-simple-okp, 264
 arg-types-simple-okp-off, 265
 asm, 556, 559

asm-and-fm9001, 559
asm-flags, 555, 556
asm-line, 556
asm-list, 556
asm-op-code, 553, 556
asm-register, 550, 553, 556
asm-register-a, 553, 556
asm-store-cc, 554, 556
asm-to-1-and-0, 556
associativity-of-append, 135
associativity-of-gcd, 34
associativity-of-gcd-off, 117
associativity-of-gcd-off1, 1168
associativity-of-gcd-on, 1080
associativity-of-gcd-zero-case, 34
 off, 117
 off1, 1168
 on, 1080
associativity-of-iplus, 39
associativity-of-iplus-off, 114
associativity-of-iplus-off1, 1166
associativity-of-iplus-on, 1077
associativity-of-itimes, 50
associativity-of-itimes-off, 110
associativity-of-itimes-off1, 1162
associativity-of-itimes-on, 1074
associativity-of-plus, 6
associativity-of-plus-off, 129
associativity-of-plus-off1, 1180
associativity-of-plus-on, 1092
associativity-of-times, 13
associativity-of-times-off, 127
associativity-of-times-off1, 1178
associativity-of-times-on, 1090

b-and, 153, 155, 156, 162, 201, 438,
 443, 445, 446, 488, 503, 517,
 523, 674, 675, 914
b-and-monotone, 940
b-and-monotone-off, 1052
b-and-monotone-on, 1142
b-and-preserves-good-s, 1004
b-and-preserves-good-s-off, 1038
b-and-preserves-good-s-on, 1127
b-and-rewrite, 155
b-and3, 153, 155, 201, 499, 674, 675
b-and3-monotone, 940
b-and3-monotone-off, 1052
b-and3-monotone-on, 1141
b-and3-preserves-good-s, 1004
b-and3-preserves-good-s-off, 1038
b-and3-preserves-good-s-on, 1127
b-and3\$netlist, 392, 500, 688, 739
b-and3\$value, 392
b-and3\$value-off, 392
b-and3&, 392, 500, 688, 738
b-and3&-off, 392
b-and4, 154, 155, 201, 674
b-and4-monotone, 940
b-and4-monotone-off, 1052
b-and4-monotone-on, 1141
b-and4-preserves-good-s, 1004
b-and4-preserves-good-s-off, 1038
b-and4-preserves-good-s-on, 1127
b-and4\$netlist, 392, 672, 688
b-and4\$value, 392
b-and4\$value-as-f\$all-t-regs-
 address, 592
 address-off, 592
b-and4\$value-off, 392
b-and4&, 392, 592, 672, 688
b-and4&-off, 392
b-and\$netlist, 392, 437, 489, 514,
 518, 524, 566, 567, 672, 688,
 714
b-and\$value, 392
b-and\$value-off, 392
b-and&, 392, 437, 489, 514, 518, 524,
 566, 567, 672, 688, 714
b-and&-off, 392
b-approx, 928–930, 933–936, 938, 973,
 1002, 1019
b-approx-off, 938
b-approx-off1, 1024
b-approx-off2, 1152
b-approx-on, 1017
b-approx-on1, 1063

b-approx-x, 1019
 b-approx-x-off, 1028
 b-approx-x-on, 1117
 b-buf, 152, 155, 156, 200, 456, 464, 466
 b-buf*, 418, 419
 b-buf-pwr*, 419
 b-buf-pwr\$netlist, 419, 477, 480, 570, 748, 755, 758, 776
 b-buf-pwr\$value, 419
 b-buf-pwr\$value-off, 419
 b-buf-pwr&, 419, 476, 480, 570, 748, 755, 758, 776
 b-buf-pwr&-off, 419
 b-buf-x=x, 155
 b-buf\$netlist, 419, 429, 451, 455, 477, 480, 568, 569, 750, 752, 755, 758, 776
 b-buf\$value, 419
 b-buf\$value-off, 419
 b-buf&, 419, 429, 450, 454, 455, 476, 480, 568, 569, 750–753, 755, 758, 776
 b-buf&-off, 419
 b-equiv, 153–155, 162, 201, 463, 464, 466, 487
 b-equiv-monotone, 940
 b-equiv-monotone-off, 1052
 b-equiv-monotone-on, 1141
 b-equiv-preserves-good-s, 1004
 b-equiv-preserves-good-s-off, 1038
 b-equiv-preserves-good-s-on, 1127
 b-equiv3, 153–155, 201, 501
 b-equiv3-monotone, 940
 b-equiv3-monotone-off, 1051
 b-equiv3-monotone-on, 1141
 b-equiv3-preserves-good-s, 1004
 b-equiv3-preserves-good-s-off, 1038
 b-equiv3-preserves-good-s-on, 1127
 b-equiv3\$netlist, 393, 502
 b-equiv3\$value, 393
 b-equiv3\$value-off, 393
 b-equiv3&, 393, 502
 b-equiv3&-off, 393
 b-equiv\$netlist, 393, 464, 469, 489
 b-equiv\$value, 393
 b-equiv\$value-off, 393
 b-equiv&, 392, 393, 463, 469, 489
 b-equiv&-off, 392
 b-gates, 155
 b-if, 154, 155, 202, 520, 544
 b-if-monotone, 941
 b-if-monotone-off, 1051
 b-if-monotone-on, 1141
 b-if-preserves-good-s, 1004
 b-if-preserves-good-s-off, 1038
 b-if-preserves-good-s-on, 1127
 b-if\$netlist, 393, 438, 451, 522, 566
 b-if\$value, 393
 b-if\$value-off, 393
 b-if&, 393, 439, 450, 522, 566
 b-if&-off, 393
 b-knownp, 928–930, 953
 b-knownp-compound-recognizer, 928
 b-knownp-compound-recognizer-of f, 1057
 b-knownp-compound-recognizer-on, 1146
 b-knownp-off, 928
 b-knownp-off1, 1024
 b-knownp-off2, 1152
 b-knownp-on, 1017
 b-knownp-on1, 1063
 b-nand, 153, 155, 156, 200, 457, 487, 490, 496, 498, 499, 501, 517, 520, 523, 559, 573–576, 674–676
 b-nand-monotone, 941
 b-nand-monotone-off, 1051
 b-nand-monotone-on, 1140
 b-nand-preserves-good-s, 1004
 b-nand-preserves-good-s-off, 1038
 b-nand-preserves-good-s-on, 1127
 b-nand3, 153, 155, 201, 487, 488, 490, 491, 496, 498, 517, 519, 520, 573–576, 674–676
 b-nand3-monotone, 941
 b-nand3-monotone-off, 1051

b-nand3-monotone-on, 1140
 b-nand3-preserves-good-s, 1004
 b-nand3-preserves-good-s-off, 1038
 b-nand3-preserves-good-s-on, 1127
 b-nand3\$netlist, 394, 489, 492, 498,
 499, 518, 522, 564, 578, 688,
 714
 b-nand3\$value, 394
 b-nand3\$value-off, 394
 b-nand3&, 394, 489, 492, 497, 499,
 518, 522, 564, 578, 688, 713
 b-nand3&-off, 394
 b-nand4, 153, 155, 201, 487, 488,
 490, 517, 579, 580, 674, 675
 b-nand4-monotone, 941
 b-nand4-monotone-off, 1051
 b-nand4-monotone-on, 1140
 b-nand4-preserves-good-s, 1005
 b-nand4-preserves-good-s-off, 1037
 b-nand4-preserves-good-s-on, 1127
 b-nand4\$netlist, 394, 489, 492, 518,
 564, 582, 688
 b-nand4\$value, 394
 b-nand4\$value-off, 394
 b-nand4&, 394, 489, 492, 518, 564,
 582, 688
 b-nand4&-off, 394
 b-nand5, 153, 155, 201, 674, 675
 b-nand5-monotone, 942
 b-nand5-monotone-off, 1050
 b-nand5-monotone-on, 1140
 b-nand5-preserves-good-s, 1005
 b-nand5-preserves-good-s-off, 1037
 b-nand5-preserves-good-s-on, 1127
 b-nand5\$netlist, 394, 688, 714
 b-nand5\$value, 394
 b-nand5\$value-off, 394
 b-nand5&, 394, 688, 713
 b-nand5&-off, 394
 b-nand6, 153, 155, 201, 674
 b-nand6-monotone, 942
 b-nand6-monotone-off, 1050
 b-nand6-monotone-on, 1140
 b-nand6-preserves-good-s, 1005
 b-nand6-preserves-good-s-off, 1037
 b-nand6-preserves-good-s-on, 1127
 b-nand6\$netlist, 395, 688
 b-nand6\$value, 395
 b-nand6\$value-off, 395
 b-nand6&, 394, 395, 688
 b-nand6&-off, 395
 b-nand8, 153, 155, 201
 b-nand8-monotone, 943
 b-nand8-monotone-off, 1050
 b-nand8-monotone-on, 1139
 b-nand8-preserves-good-s, 1005
 b-nand8-preserves-good-s-off, 1037
 b-nand8-preserves-good-s-on, 1126
 b-nand8\$netlist, 395
 b-nand8\$value, 395
 b-nand8\$value-off, 395
 b-nand8&, 395
 b-nand8&-off, 395
 b-nand\$netlist, 393, 455, 489, 492,
 498–500, 502, 518, 522, 524,
 560, 564, 566, 578, 688, 714
 b-nand\$value, 393
 b-nand\$value-off, 394
 b-nand&, 393, 454, 455, 489, 492,
 497, 499, 500, 502, 518, 522,
 524, 560, 564, 566, 578, 688,
 713
 b-nand&-off, 393
 b-nbuf-monotone, 943
 b-nbuf-monotone-off, 1050
 b-nbuf-monotone-on, 1139
 b-nbuf-preserves-good-s, 1005
 b-nbuf-preserves-good-s-off, 1037
 b-nbuf-preserves-good-s-on, 1126
 b-nbuf\$netlist, 395, 419, 498
 b-nbuf\$value, 395
 b-nbuf\$value-off, 395
 b-nbuf&, 395, 419, 497
 b-nbuf&-off, 395
 b-nor, 154–156, 201, 457, 463, 486,
 487, 490, 491, 520, 573–576,
 579, 580
 b-nor-monotone, 943

b-nor-monotone-off, 1050
 b-nor-monotone-on, 1139
 b-nor-preserves-good-s, 1005
 b-nor-preserves-good-s-off, 1037
 b-nor-preserves-good-s-on, 1126
 b-nor3, 154, 155, 201, 463, 486, 579, 580
 b-nor3-monotone, 943
 b-nor3-monotone-off, 1049
 b-nor3-monotone-on, 1139
 b-nor3-preserves-good-s, 1005
 b-nor3-preserves-good-s-off, 1037
 b-nor3-preserves-good-s-on, 1126
 b-nor3\$netlist, 396, 461, 464, 487, 582, 714
 b-nor3\$value, 396
 b-nor3\$value-off, 396
 b-nor3&, 396, 460, 463, 487, 582, 714
 b-nor3&-off, 396
 b-nor4, 154, 155, 201, 579, 580
 b-nor4-monotone, 943
 b-nor4-monotone-off, 1049
 b-nor4-monotone-on, 1138
 b-nor4-preserves-good-s, 1005
 b-nor4-preserves-good-s-off, 1037
 b-nor4-preserves-good-s-on, 1126
 b-nor4\$netlist, 396, 582, 714
 b-nor4\$value, 396
 b-nor4\$value-off, 396
 b-nor4&, 396, 582, 714
 b-nor4&-off, 396
 b-nor5, 154, 155, 202
 b-nor5-monotone, 944
 b-nor5-monotone-off, 1049
 b-nor5-monotone-on, 1138
 b-nor5-preserves-good-s, 1005
 b-nor5-preserves-good-s-off, 1037
 b-nor5-preserves-good-s-on, 1126
 b-nor5\$netlist, 396, 714
 b-nor5\$value, 396
 b-nor5\$value-off, 397
 b-nor5&, 396, 714
 b-nor5&-off, 396
 b-nor6, 154, 155, 202
 b-nor6-monotone, 944
 b-nor6-monotone-off, 1049
 b-nor6-monotone-on, 1138
 b-nor6-preserves-good-s, 1005
 b-nor6-preserves-good-s-off, 1037
 b-nor6-preserves-good-s-on, 1126
 b-nor6\$netlist, 397, 714
 b-nor6\$value, 397
 b-nor6\$value-off, 397
 b-nor6&, 397, 714
 b-nor6&-off, 397
 b-nor8, 154, 155, 202
 b-nor8-monotone, 945
 b-nor8-monotone-off, 1049
 b-nor8-monotone-on, 1138
 b-nor8-preserves-good-s, 1005
 b-nor8-preserves-good-s-off, 1037
 b-nor8-preserves-good-s-on, 1126
 b-nor8\$netlist, 397, 714
 b-nor8\$value, 397
 b-nor8\$value-off, 397
 b-nor8&, 397, 714
 b-nor8&-off, 397
 b-nor\$netlist, 395, 455, 464, 487, 489, 492, 522, 565, 578, 582, 714
 b-nor\$value, 396
 b-nor\$value-off, 396
 b-nor&, 395, 396, 454, 455, 463, 487, 489, 492, 522, 565, 578, 582, 713
 b-nor&-off, 395
 b-not, 153, 155, 156, 159, 162, 163, 200, 444, 456, 463, 487, 488, 490, 491, 496, 501, 517, 519, 520, 523, 559, 573–576, 579, 674–676, 1103
 b-not-b4ip-monotone, 945
 b-not-b4ip-monotone-off, 1048
 b-not-b4ip-monotone-on, 1137
 b-not-b4ip-preserves-good-s, 1005
 b-not-b4ip-preserves-good-s-off, 1037
 b-not-b4ip-preserves-good-s-on, 1126
 b-not-b4ip\$netlist, 398, 419

b-not-b4ip\$value, 398
 b-not-b4ip\$value-off, 398
 b-not-b4ip&, 397, 398, 419
 b-not-b4ip&-off, 397
 b-not-ivap-monotone, 945
 b-not-ivap-monotone-off, 1048
 b-not-ivap-monotone-on, 1137
 b-not-ivap-preserves-good-s, 1005
 b-not-ivap-preserves-good-s-off, 1036
 b-not-ivap-preserves-good-s-on, 1126
 b-not-ivap\$netlist, 398
 b-not-ivap\$value, 398
 b-not-ivap\$value-off, 398
 b-not-ivap&, 398
 b-not-ivap&-off, 398
 b-not-monotone, 945
 b-not-monotone-off, 1048
 b-not-monotone-on, 1138
 b-not-preserves-good-s, 1005
 b-not-preserves-good-s-off, 1037
 b-not-preserves-good-s-on, 1126
 b-not\$netlist, 397, 419, 445, 455, 458,
 464, 475, 489, 492, 502, 518,
 522, 524, 560, 564–567, 578,
 582, 672, 688, 714, 721, 776
 b-not\$value, 397
 b-not\$value-off, 397
 b-not&, 397, 419, 445, 454, 458, 463,
 474, 489, 492, 502, 518, 522,
 524, 560, 564–567, 578, 582,
 672, 688, 713, 721, 776
 b-not&-off, 397
 b-or, 153, 155, 156, 162, 201, 438,
 446, 457, 465, 466, 496, 523,
 561, 674, 675, 915
 b-or-monotone, 945
 b-or-monotone-off, 1048
 b-or-monotone-on, 1137
 b-or-preserves-good-s, 1005
 b-or-preserves-good-s-off, 1036
 b-or-preserves-good-s-on, 1126
 b-or3, 153, 155, 201, 443, 519, 520,
 523
 b-or3-monotone, 946
 b-or3-monotone-off, 1047
 b-or3-monotone-on, 1137
 b-or3-preserves-good-s, 1005
 b-or3-preserves-good-s-off, 1036
 b-or3-preserves-good-s-on, 1126
 b-or3\$netlist, 398, 522, 524, 714
 b-or3\$value, 398
 b-or3\$value-off, 399
 b-or3&, 398, 522, 524, 713
 b-or3&-off, 398
 b-or4, 153, 155, 201
 b-or4-monotone, 946
 b-or4-monotone-off, 1047
 b-or4-monotone-on, 1137
 b-or4-preserves-good-s, 1006
 b-or4-preserves-good-s-off, 1036
 b-or4-preserves-good-s-on, 1125
 b-or4\$netlist, 399, 672, 714
 b-or4\$value, 399
 b-or4\$value-as-f\$set-some-fla
 gs, 592
 gs-off, 592
 b-or4\$value-off, 399
 b-or4&, 399, 592, 672, 713
 b-or4&-off, 399
 b-or\$netlist, 398, 430, 438, 455, 498,
 524, 563, 672, 688, 714
 b-or\$value, 398
 b-or\$value-off, 398
 b-or&, 398, 430, 438, 454, 455, 471,
 497, 524, 562, 672, 688, 713
 b-or&-off, 398
 b-reg, 759, 761–763
 b-reg-off, 760
 b-store-resultp, 561–563
 b-store-resultp*, 562, 563
 b-store-resultp-off, 561
 b-store-resultp=store-resultp, 563
 b-store-resultp=store-resultp\$
 help, 563
 b-store-resultp\$netlist, 563, 672
 b-store-resultp\$value, 563
 b-store-resultp&, 562, 563, 672
 b-store-resultp&-off, 562

b-to-nat, 1094, 1099, 1100, 1102–1104, 1192, 1194, 1195, 1206, 1207, 1211, 1212, 1214, 1218, 1219
 b-to-nat-leq, 1102
 b-to-nat-leq-off, 1111
 b-to-nat-off, 1115
 b-xor, 153–156, 162, 201, 443, 446, 463, 490, 491, 519, 520, 561
 b-xor-monotone, 946
 b-xor-monotone-off, 1047
 b-xor-monotone-on, 1136
 b-xor-preserves-good-s, 1006
 b-xor-preserves-good-s-off, 1036
 b-xor-preserves-good-s-on, 1125
 b-xor3, 153–155, 162, 201, 438
 b-xor3-monotone, 946
 b-xor3-monotone-off, 1047
 b-xor3-monotone-on, 1136
 b-xor3-preserves-good-s, 1006
 b-xor3-preserves-good-s-off, 1036
 b-xor3-preserves-good-s-on, 1125
 b-xor3\$netlist, 399
 b-xor3\$value, 399
 b-xor3\$value-off, 399
 b-xor3&, 399
 b-xor3&-off, 399
 b-xor\$netlist, 399, 431, 437, 464, 492, 522, 563
 b-xor\$value, 399
 b-xor\$value-off, 399
 b-xor&, 399, 431, 432, 437, 463, 492, 522, 562
 b-xor&-off, 399
 bad-lsi-name-chars, 285, 286
 bad-lsi-name-chars-off, 285
 bad-names, 276, 277
 bad-names-off, 277
 bagdiff, 2–4, 8, 10–12, 18–20, 29, 44–49, 55, 57–63, 67, 78, 79, 81
 bagdiff-delete, 4
 bagdiff-delete-off, 130
 bagdiff-delete-off1, 1181
 bagdiff-delete-on, 1093
 bagdiff-off, 131
 bagdiff-off1, 1182
 bagdiff-on, 1094
 bagint, 3–5, 10–12, 18–20, 28, 29, 44, 45, 47–49, 55, 56, 59–62, 67–69, 79, 80, 82
 bagint-off, 131
 bagint-off1, 1182
 bagint-on, 1094
 bagint-singleton, 80
 bagint-singleton-off, 104
 bagint-singleton-off1, 1156
 bagint-singleton-on, 1067
 bags, 5
 boolfix, 150, 152, 157, 161, 162, 444
 boolfix-car-x=x, 152
 boolp, 149–152, 155, 161, 165, 175–182, 184–192, 194, 196, 200–204, 267, 386, 438, 442–444, 464, 467, 469, 473–475, 483–485, 487, 490, 493, 499–501, 510, 511, 519, 522, 523, 525, 526, 531–536, 548, 561, 563, 566, 568, 584, 591, 592, 678, 701, 702, 719, 734, 739, 741, 748, 810, 814, 816, 818, 819, 821, 823, 825, 827, 830, 832, 833, 835, 837, 839, 842, 843, 846, 847, 849, 852, 854, 856, 858, 860, 862, 864, 866, 867, 870, 871, 875, 876, 878, 880, 882, 883, 885, 886, 888–890, 892, 894, 896, 898, 900, 903, 905, 908, 910, 912, 917, 918, 1099, 1186, 1188, 1190
 boolp-a-immediate-p, 548
 boolp-b-gates, 155
 boolp-b-gates-off, 155
 boolp-bvp-core-alu, 532
 boolp-bvp-cvzbv, 483
 boolp-c-flag, 591
 boolp-c-flag-update-flags, 548
 boolp-c-v-alu, 484

boolp-car-f\$tv-alu-help, 510
 boolp-car-tv-dec-pass-ng, 467
 boolp-car-x, 152
 boolp-flag-extracters, 1186
 boolp-if*, 165
 boolp-implies-not-equal-z, 150
 boolp-implies-not-equal-z-off, 150
 boolp-lemmas, 150
 boolp-list-ok, 367–369
 boolp-list-ok-off, 367
 boolp-n, 483
 boolp-nth, 152
 boolp-off, 149
 boolp-store-result-p, 548
 boolp-v-negp, 161
 boolp-v-v-alu, 484
 boolp-zb-v-alu, 484
 boundp, 247, 248, 295, 297, 299, 300,
 307, 313, 327, 332, 336, 337,
 345, 375, 384, 385
 boundp-off, 248
 boundp-off1, 378
 boundp-on, 378
 breadth-tree, 191, 192
 bridge-to-subbagp-implies-plus-t
 ree-greatereqp, 8
 ree-greatereqp-off, 8
 btr-or-is-v-nzerop, 457
 btr-or-nor, 456, 457
 btr-or-nor-off, 457
 bv, 482, 483, 485, 486, 532, 536, 545,
 733, 761, 763, 779, 781, 782,
 792, 794, 801, 804, 811, 813,
 834, 838, 850, 851, 857, 859,
 869, 873, 897, 899, 901, 902,
 904, 907–909, 911, 913, 1188,
 1189, 1216, 1218, 1219
 bv-adder, 440, 442, 443
 bv-adder*, 440, 441
 bv-adder*-off, 441
 bv-adder*\$destructure, 441
 bv-adder*\$destructure-off, 441
 bv-adder-body, 440–443
 bv-adder-body-off, 440
 bv-adder-body-special-case\$val
 ue, 442
 ue-off, 443
 bv-adder-body\$induction, 441, 442
 bv-adder-body\$value, 442
 bv-adder-body\$value-off, 443
 bv-adder\$netlist, 441
 bv-adder\$netlist-off, 441
 bv-adder\$unbound-in-body-carry, 442
 -off, 442
 bv-adder\$unbound-in-body-sum, 442
 bv-adder\$unbound-in-body-sum-o
 ff, 442
 bv-adder\$value, 443
 bv-adder\$value-off, 443
 bv-adder&, 441, 443
 bv-adder&-off, 441
 bv-as-subrange, 779
 bv-as-subrange-off, 779
 bv-cvzbv, 1216
 bv-off, 482
 bv-to-int, 549
 bv-v-alu, 1216
 bv-v-alu-alu-inc-alu-dec, 486
 bv-v-alu-as-integer, 1219
 bv-v-alu-as-natural, 1218
 bv2p, 151, 161, 163–165, 505, 506,
 532–536, 1100, 1102–1106,
 1186, 1189, 1191–1196, 1205–
 1207, 1211–1213, 1217–1219
 bvp, 150–152, 157–165, 169, 170, 172–
 176, 178–180, 182, 184, 185,
 187–189, 202–204, 442–444,
 457, 459, 462, 464, 465, 467,
 469, 475, 483–486, 494, 497,
 501, 505–508, 511, 526, 527,
 529, 531, 532, 535, 536, 548,
 562–566, 576, 578, 587, 591,
 592, 603, 604, 669, 670, 673,
 678, 689, 701–703, 717–719,
 734, 735, 739, 741, 745, 746,
 748, 810, 814, 816–819, 821,
 823, 825, 827, 829, 831–833,
 835, 837, 839, 841, 843, 845,

847, 849, 852, 854, 856, 858,
 860, 862–867, 869–871, 874–
 876, 878, 880–883, 885–887,
 889–892, 894, 896, 898, 900,
 903, 905, 908, 910, 912, 915–
 918, 926, 953, 954, 971, 993,
 1095, 1096, 1098–1102, 1104–
 1106, 1186, 1188, 1190, 1197,
 1200, 1204, 1205, 1208–1211,
 1214–1216, 1218, 1219
 bvp-a-immediate, 548
 bvp-append, 151
 bvp-bv-v-alu, 485
 bvp-cddr-f\$tv-alu-help, 511
 bvp-cddr-tv-alu-help, 505
 bvp-cdr, 1095
 bvp-cdr-off, 1115
 bvp-cdr-tv-dec-pass, 465
 bvp-cdr-tv-dec-pass-ng, 467
 bvp-cons, 151
 bvp-cv-vectors, 669
 bvp-cv-vectors-off, 670
 bvp-cvzbv, 483
 bvp-f\$tv-alu-help, 511
 bvp-firstn, 151
 bvp-if, 151
 bvp-if*, 151
 bvp-implies-v-knownnp, 953
 bvp-implies-v-knownnp-off, 994
 bvp-implies-v-knownnp-off1, 1148
 bvp-implies-v-knownnp-on, 1059
 bvp-int-to-v, 1197
 bvp-ir-accessors, 548
 bvp-is-properp, 151
 bvp-length, 151, 152, 465, 467, 505
 bvp-length-alu-dec-op, 486
 bvp-length-alu-inc-op, 486
 bvp-length-bv, 485
 bvp-length-cdr, 152
 bvp-length-cv, 703
 bvp-length-decode-5, 576
 bvp-length-encode-32, 586
 bvp-length-next-cntl-state, 717
 bvp-length-next-state, 688
 bvp-length-off, 152
 bvp-length-read-regs-32, 735
 bvp-length-restn, 152
 bvp-length-state-vectors, 603
 bvp-length-state-vectors-off, 604
 bvp-length-tv-alu-help, 505
 bvp-length-tv-dec-pass, 465
 bvp-length-tv-dec-pass-ng, 467
 bvp-length-update-flags, 548
 bvp-length-v-inc-v-dec, 163
 bvp-make-list, 151
 bvp-mpg, 494
 bvp-nat-to-v, 160
 bvp-nlistp, 151
 bvp-nthcdr, 151
 bvp-off, 151
 bvp-off1, 972
 bvp-off2, 1061
 bvp-on, 995
 bvp-on1, 1150
 bvp-read-fn, 874
 bvp-read-mem, 173
 bvp-read-mem-32, 173
 bvp-read-mem1, 173
 bvp-restn, 151
 bvp-rev1, 953
 bvp-rev1-off, 1042
 bvp-rev1-on, 1131
 bvp-reverse, 953
 bvp-reverse-off, 993
 bvp-reverse-off1, 1148
 bvp-reverse-on, 1059
 bvp-shift-or-buf, 526
 bvp-sign-extend, 162
 bvp-subrange, 165
 bvp-sum, 443
 bvp-tv-alu-help, 505
 bvp-tv-dec-pass, 465
 bvp-tv-dec-pass-ng, 467
 bvp-v-adder, 163
 bvp-v-alu, 485
 bvp-v-alu-1, 485
 bvp-v-and, 157
 bvp-v-asr, 157

- bvp-v-buf, 157
- bvp-v-fourfix, 165
- bvp-v-if, 158
- bvp-v-lsr, 157
- bvp-v-not, 157
- bvp-v-or, 157
- bvp-v-ror, 158
- bvp-v-shift-right, 157
- bvp-v-threefix, 164
- bvp-v-xor, 157
- c, 482–484, 532, 544, 742–744, 1216, 1217
- c-cvzbv, 1216
- c-flag, 543–545, 548, 559, 591, 608, 610, 612, 614, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 664, 666, 668, 670, 673, 716, 744, 834, 838, 850, 851, 857, 859, 868, 872, 897, 899, 901, 904, 906–909, 911, 913, 1186, 1189–1196, 1200–1216
- c-flag-off, 543
- c-off, 482
- c-set, 542, 544, 591, 743, 744, 1190, 1192–1196, 1214
- c-set-off, 543
- c-v-alu, 1217
- c10-int, 549
- c10-tf, 549
- cadr-eval\$list, 9
- cadr-eval\$list-off, 9
- cadr-map-up, 1022
- cadr-map-up-off, 1027
- cadr-map-up-on, 1116
- cadr-tv-adder-as-g, 448
- cadr-tv-adder-as-g-off, 448
- cancel-constants-equal, 95
- cancel-constants-equal-lemma, 95
- cancel-constants-equal-lemma-of f, 100
 - f1, 1153
- cancel-constants-equal-lemma-on, 1063
- cancel-constants-equal-off, 100
- cancel-constants-equal-off1, 1153
- cancel-constants-equal-on, 1063
- cancel-constants-ilessp, 96
- cancel-constants-ilessp-lemma-1, 95
 - off, 100
 - off1, 1152
 - on, 1063
- cancel-constants-ilessp-lemma-2, 95
 - off, 100
 - off1, 1152
 - on, 1063
- cancel-constants-ilessp-off, 100
- cancel-constants-ilessp-off1, 1152
- cancel-constants-ilessp-on, 1063
- cancel-difference-plus, 10, 11
- cancel-difference-plus-off, 128
- cancel-difference-plus-off1, 1179
- cancel-difference-plus-on, 1091
- cancel-equal-plus, 9, 10
- cancel-equal-plus-off, 128
- cancel-equal-plus-off1, 1180
- cancel-equal-plus-on, 1092
- cancel-equal-times, 19, 21
- cancel-equal-times-off, 125
- cancel-equal-times-off1, 1176
- cancel-equal-times-on, 1088
- cancel-equal-times-preserves-inequality, 20
 - uality-bridge, 20
 - uality-bridge-off, 21
 - uality-off, 20
- cancel-factors-0, 83, 84
- cancel-factors-0-off, 103
- cancel-factors-0-off1, 1155
- cancel-factors-0-on, 1066
- cancel-factors-ilessp-0, 84, 85
- cancel-factors-ilessp-0-off, 102
- cancel-factors-ilessp-0-off1, 1154
- cancel-factors-ilessp-0-on, 1065
- cancel-ineq, 41, 42
- cancel-ineq-aux, 40–42

- cancel-ineg-aux-off, 113
- cancel-ineg-aux-off1, 1165
- cancel-ineg-aux-on, 1077
- cancel-ineg-off, 113
- cancel-ineg-off1, 1165
- cancel-ineg-on, 1077
- cancel-ineg-terms-from-equality, 88, 91
 - cancel-ineg-terms-from-equality-expanded, 91
 - cancel-ineg-terms-from-equality-expanded-off, 101
 - cancel-ineg-terms-from-equality-expanded-off1, 1153
 - cancel-ineg-terms-from-equality-expanded-on, 1064
 - expanded, 89, 91, 92
 - expanded-off, 91
 - off, 101
 - off1, 1154
 - on, 1065
- cancel-ineg-terms-from-inequality, 92, 94
 - y-cancel-ineg-terms-from-inequality-expanded, 94
 - y-cancel-ineg-terms-from-inequality-expanded-off, 100
 - y-cancel-ineg-terms-from-inequality-expanded-off1, 1153
 - y-cancel-ineg-terms-from-inequality-expanded-on, 1064
 - y-expanded, 93, 94
 - y-expanded-off, 94
 - y-off, 101
 - y-off1, 1153
 - y-on, 1064
- cancel-iplus, 43, 45, 47
- cancel-iplus-ilessp, 48, 49
- cancel-iplus-ilessp-1, 47, 48
- cancel-iplus-ilessp-1-off, 111
- cancel-iplus-ilessp-1-off1, 1163
- cancel-iplus-ilessp-1-on, 1075
- cancel-iplus-ilessp-off, 111
- cancel-iplus-ilessp-off1, 1163
- cancel-iplus-ilessp-on, 1074
- cancel-iplus-off, 46
- cancel-itimes, 55, 60
- cancel-itimes-factors, 68, 78
- cancel-itimes-factors-expanded, 78, 79
- cancel-itimes-factors-expanded-
 - cancel-itimes-factors, 78
 - cancel-itimes-factors-off, 105
 - cancel-itimes-factors-off1, 1157
 - cancel-itimes-factors-on, 1068
 - off, 78
- cancel-itimes-factors-off, 105
- cancel-itimes-factors-off1, 1157
- cancel-itimes-factors-on, 1068
- cancel-itimes-ilessp, 61, 63
- cancel-itimes-ilessp-factors, 79, 83
- cancel-itimes-ilessp-factors-of
 - f, 104
 - f1, 1156
- cancel-itimes-ilessp-factors-on, 1067
- cancel-itimes-ilessp-off, 106
- cancel-itimes-ilessp-off1, 1158
- cancel-itimes-ilessp-on, 1069
- cancel-itimes-off, 106
- cancel-itimes-off1, 1158
- cancel-itimes-on, 1070
- cancel-lessp-plus, 12
- cancel-lessp-plus-off, 128
- cancel-lessp-plus-off1, 1179
- cancel-lessp-plus-on, 1091
- cancel-lessp-times, 18, 19
- cancel-lessp-times-off, 125
- cancel-lessp-times-off1, 1176
- cancel-lessp-times-on, 1088
- cancel-quotient-times, 28, 29
- cancel-quotient-times-off, 120
- cancel-quotient-times-off1, 1172
- cancel-quotient-times-on, 1083
- car-cdr-if-cons, 133
- car-firstn, 140
- car-nat-to-v-0-is-f, 160
- car-tv-adder-as-p, 448
- car-tv-adder-as-p-off, 448
- carry-flag-nat-interpretation-f

- or-sub, 1195
- carry-in-help, 496–498, 532–536, 608, 610, 612, 614, 615, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 664, 666, 668, 838, 850, 851, 859, 868, 872, 901, 904, 906, 909, 911, 913
- carry-in-help*, 497, 498
- carry-in-help-if-op-code, 498
- carry-in-help-off, 496
- carry-in-help-zero, 498
- carry-in-help-zero-off, 498
- carry-in-help\$netlist, 498, 714
- carry-in-help\$value, 498
- carry-in-help\$value-off, 498
- carry-in-help\$value-zero, 498
- carry-in-help\$value-zero-off, 498
- carry-in-help&, 497, 498, 714
- carry-in-help&-off, 497
- carry-out-help, 517, 519, 530
- carry-out-help*, 517, 518
- carry-out-help-congruence, 519
- carry-out-help\$netlist, 518, 540
- carry-out-help\$value, 518
- carry-out-help\$value-off, 519
- carry-out-help\$value-zero, 519
- carry-out-help\$value-zero-off, 519
- carry-out-help&, 518, 519, 540
- carry-out-help&-off, 518
- cases-on-a-4-bit-bvp, 535
- cc-cc, 1187, 1190–1196, 1214
- cc-cs, 1187, 1190–1192, 1194–1196, 1215
- cc-eq, 1187, 1188, 1190, 1191, 1194, 1196, 1201, 1202, 1204, 1209, 1210, 1214
- cc-f, 1187, 1189
- cc-ge, 1187, 1203, 1205–1207, 1209, 1210, 1212, 1213, 1215
- cc-gt, 1187, 1203–1207, 1209, 1211–1213, 1216
- cc-hi, 1187, 1195
- cc-le, 1187, 1203–1205, 1207–1209, 1211–1213, 1216
- cc-ls, 1187, 1195, 1196
- cc-lt, 1187, 1203, 1205–1208, 1210, 1212, 1213, 1215
- cc-mi, 1187, 1189, 1190, 1202, 1215
- cc-ne, 1187–1189, 1191, 1194, 1196, 1201, 1204, 1209, 1210, 1214
- cc-pl, 1187, 1189, 1190, 1202, 1215
- cc-t, 1187, 1189
- cc-vc, 1187, 1200, 1204, 1205, 1207, 1208, 1210–1212, 1216
- cc-vs, 1187, 1201, 1204, 1206–1208, 1210, 1211, 1213, 1216
- cddr-f\$tv-alu-help-length, 510
- cddr-tv-adder-as-sum, 448
- cddr-tv-adder-as-sum-off, 448
- cddr-tv-alu-help-length, 505
- cdr-cdr-sub1-sub1-induction, 168
- cdr-nthcdr, 143
- cdr-nthcdr-off, 143
- cdr-restn, 141
- cdr-restn-off, 141
- cdr-tv-dec-pass-length, 465
- check-cv\$netlist, 714
- check-dec-pass\$netlist, 475
- check-extend-immediate\$netlist, 748
- check-fast-zero\$netlist, 461
- check-flags\$netlist, 744
- check-next-cntl-state\$netlist, 721
- check-regfile\$netlist, 739
- check-tv-dec-pass-ng\$netlist, 474
- chip*, 790, 791
- chip*-off, 790
- chip*\$destructure-off, 790
- chip-module*, 775, 776
- chip-module*-off, 775
- chip-module*\$destructure-off, 775
- chip-module\$netlist, 776, 791, 1026
- chip-module\$state, 780
- chip-module\$value, 783
- chip-module&, 775, 780, 783, 791
- chip-module&-off, 776

- chip-system*, 799, 800
- chip-system*-off, 799
- chip-system*\$destructure-off, 799
- chip-system-input-invariant, 807–809
- chip-system-invariant, 800, 801, 805, 807, 809, 919, 1012, 1014
- chip-system-invariant-unknown-state, 1012
 - ate, 1012
 - ate-off, 1032
 - ate-on, 1121
- chip-system-operating-inputs-p, 808, 809, 925, 927, 928, 1012, 1014, 1021, 1023, 1025
- chip-system-operating-inputs-p-off, 808
 - reset-sequence, 1012
 - reset-sequence-off, 1031
 - reset-sequence-on, 1120
- chip-system=fm9001-interpret, 925
- chip-system=fm9001-interpret\$
 - after-reset, 1024
 - after-reset-off, 1027
 - after-reset-on, 1116
- chip-system=fm9001\$step, 809
- chip-system=fm9001\$step-off, 809
- chip-system=run-fm9001, 809
- chip-system=run-fm9001\$inducti
 - on, 809
- chip-system\$netlist, 800, 928, 1014
- chip-system\$state, 805
- chip-system\$state-help, 800
- chip-system\$state-help-off, 805
- chip-system\$state-off, 806
- chip-system&, 799, 800, 805, 809, 925, 927, 1014, 1021, 1022, 1024
- chip-system&-off, 800
- chip-well-formed, 1026
- chip-well-formed-after-indexed-
 - names-removed, 1026
 - names-removed-off, 1026
 - names-removed-on, 1115
- chip-well-formed-off, 1026
- chip-well-formed-on, 1116
- chip-well-formed-simple, 1026
- chip-well-formed-simple-off, 1026
- chip-well-formed-simple-on, 1115
- chip\$netlist, 791, 800, 1016, 1020, 1021, 1024, 1026
- chip\$state, 791
- chip\$value, 794
- chip&, 790, 791, 794, 799
- chip&-off, 791
- cntl-state, 760–764
- cntl-state-off, 760
- collect-breadth-tree, 192
- collect-delays, 334, 346
- collect-delays-0, 333, 334
- collect-delays-0-off, 334
- collect-delays-off, 334
- collect-drives, 317, 346
- collect-drives-0, 316, 317
- collect-drives-0-off, 317
- collect-drives-off, 317
- collect-drives-simple, 272, 274
- collect-drives-simple-off, 272
- collect-in-types, 304, 350
- collect-in-types-off, 304
- collect-loadings, 313, 347
- collect-loadings-off, 313
- collect-loadings-simple, 271, 274
- collect-loadings-simple-off, 272
- collect-min-args, 313–316
- collect-min-args-off, 314
- collect-module-data, 349, 351
- collect-module-data-off, 351
- collect-module-errors, 349, 350
- collect-module-errors-off, 349
- collect-module-prop, 346, 348
- collect-module-prop-off, 347
- collect-module-props, 347, 348, 350
- collect-module-props-off, 348
- collect-net-errors, 251, 353
- collect-net-errors-off, 251
- collect-occurrence-data, 343, 344, 346
- collect-occurrence-data-off, 344
- collect-out-depends, 335, 347
- collect-out-depends-errors, 335

- collect-out-depends-errors-off, 335
- collect-out-depends-off, 335
- collect-out-type, 304, 305
- collect-out-type-off, 305
- collect-out-types, 305, 350
- collect-out-types-off, 305
- collect-primitives, 435, 436
- collect-state-types, 336, 347
- collect-state-types-off, 336
- collect-states, 237, 238, 240, 241, 266
- collect-synonyms, 339
- collect-synonyms-off, 339
- collect-tri-state-data, 339, 347
- collect-tri-state-data-off, 339
- collect-types, 258, 260–262
- collect-types-off, 258
- collect-value, 166–169, 238–243, 253, 256, 301, 334, 335, 427, 429, 430, 432, 433, 435, 442, 443, 451, 452, 473, 478, 481, 516, 523, 751, 753, 755, 758, 759, 800, 999–1004
- collect-value-append, 167
- collect-value-append-pairlist-w
 - hen-disjoint, 167
 - hen-subset, 167
- collect-value-args-pairlist-arg
 - s, 168
- collect-value-cons, 168
- collect-value-disjoint-pairlist, 167
- collect-value-firstn, 167
- collect-value-litatom-indices-spee
 - dup, 167
- collect-value-make-list, 168
- collect-value-nlistp, 167
- collect-value-off, 166
- collect-value-or-unknown, 295, 336
- collect-value-or-unknown-off, 295
- collect-value-restn, 168
- collect-value-splitting-crock, 168
- collect-value-splitting-crock-he
 - lper, 168
- collect-value-subrange-args-pai
 - rlist-args, 169
- collect-value-v-shift-right, 523
- collect-value2, 297, 337
- collect-value2-off, 297
- common-divisor-divides-gcd, 34
- common-divisor-divides-gcd-off, 117
- common-divisor-divides-gcd-off1, 1168
- common-divisor-divides-gcd-on, 1080
- commutativity-of-gcd, 33
- commutativity-of-gcd-off, 118
- commutativity-of-gcd-off1, 1169
- commutativity-of-gcd-on, 1081
- commutativity-of-iplus, 39
- commutativity-of-iplus-off, 114
- commutativity-of-iplus-off1, 1166
- commutativity-of-iplus-on, 1077
- commutativity-of-iplus-on1, 1186
- commutativity-of-itimes, 49
- commutativity-of-itimes-off, 111
- commutativity-of-itimes-off1, 1163
- commutativity-of-itimes-on, 1074
- commutativity-of-plus, 5
- commutativity-of-plus-off, 130
- commutativity-of-plus-off1, 1181
- commutativity-of-plus-on, 1093
- commutativity-of-times, 13
- commutativity-of-times-off, 127
- commutativity-of-times-off1, 1178
- commutativity-of-times-on, 1090
- commutativity2-of-gcd, 34
- commutativity2-of-gcd-off, 116
- commutativity2-of-gcd-off1, 1168
- commutativity2-of-gcd-on, 1080
- commutativity2-of-gcd-zero-case, 34
 - off, 116
 - off1, 1168
 - on, 1080
- commutativity2-of-iplus, 39
- commutativity2-of-iplus-off, 114
- commutativity2-of-iplus-off1, 1166
- commutativity2-of-iplus-on, 1077
- commutativity2-of-itimes, 50
- commutativity2-of-itimes-off, 110
- commutativity2-of-itimes-off1, 1162
- commutativity2-of-itimes-on, 1074

commutativity2-of-plus, 5
 commutativity2-of-plus-off, 129
 commutativity2-of-plus-off1, 1181
 commutativity2-of-plus-on, 1093
 commutativity2-of-times, 13
 commutativity2-of-times-off, 127
 commutativity2-of-times-off1, 1178
 commutativity2-of-times-on, 1090
 compile-uncompiled-defns, 1011
 compile-uncompiled-defns-off, 1032
 compile-uncompiled-defns-on, 1011
 compile-uncompiled-defns-on1, 1026
 compile-uncompiled-defns-on2, 1122
 compose-io-types, 304, 344
 compose-io-types-off, 304
 compose-occurrence-data, 346
 compose-occurrence-data-off, 346
 compose-tri-state-data, 339, 344
 compose-tri-state-data-off, 339
 compose-type, 303, 304
 compose-type-off, 304
 composed-io-types, 344, 345
 composed-io-types-off, 344
 composed-loadings-drives-delays, 344,
 345
 -off, 345
 composed-occurrence-data, 345, 346
 composed-occurrence-data-off, 345
 composed-type-maps, 304
 composed-type-maps-off, 304
 composite-occ-body-syntax-error
 s, 280, 281, 291
 s-off, 281
 condition-code-theory, 1187
 conjoin-inequalities-with-0, 84, 85
 conjoin-inequalities-with-0-eli
 minator, 85
 minator-off, 102
 minator-off1, 1154
 minator-on, 1065
 conjoin-inequalities-with-0-off, 102
 1, 1154
 conjoin-inequalities-with-0-on, 1066
 constant-ram, 171, 172, 174, 176–
 182, 185–191, 975, 993, 1007
 constant-ram-off, 171
 control-let*, 672
 control-let\$netlist, 672, 721
 control-let\$value, 673
 control-let\$value-off, 673
 control-let&, 672, 673, 721
 control-let&-off, 672
 control-state-accessor-theory, 606
 core-alu, 530–536, 838, 850, 851, 859,
 869, 873, 901, 902, 904, 907,
 909, 911, 913
 core-alu*, 537–540
 core-alu*-off, 540
 core-alu*\$destructure, 538
 core-alu*\$destructure-off, 540
 core-alu-is-v-alu, 536
 core-alu-off, 531
 core-alu-works-as-dec-b, 536
 core-alu-works-as-inc-b, 536
 core-alu-works-for-all-normal-c
 ases, 532
 ases-off, 535
 ases\$crock, 532
 ases\$crock-off, 532
 core-alu-works-for-zero-case, 536
 core-alu\$netlist, 540, 776
 core-alu\$value, 541
 core-alu\$value-off, 541
 core-alu&, 540, 541, 776
 core-alu&-off, 540
 correctness-of-cancel-difference
 -plus, 11
 -plus-off, 128
 -plus-off1, 1096
 -plus-on, 1091
 correctness-of-cancel-equal-plu
 s, 10
 s-off, 128
 s-off1, 1096
 s-on, 1092
 correctness-of-cancel-equal-time
 s, 21

- s-off, 125
- s-off1, 1096
- s-on, 1088
- correctness-of-cancel-factors-0, 84
 - off, 102
 - off1, 1095
 - on, 1066
- correctness-of-cancel-factors-i
 - lessp-0, 85
 - lessp-0-off, 102
 - lessp-0-off1, 1096
 - lessp-0-on, 1065
- correctness-of-cancel-ineg, 42
- correctness-of-cancel-ineg-aux, 42
- correctness-of-cancel-ineg-aux-off, 42
- correctness-of-cancel-ineg-off, 113
- correctness-of-cancel-ineg-off1, 1095
- correctness-of-cancel-ineg-on, 1076
- correctness-of-cancel-ineg-term
 - s-from-equality, 92
 - s-from-equality-off, 101
 - s-from-equality-off1, 1096
 - s-from-equality-on, 1064
 - s-from-inequality, 94
 - s-from-inequality-off, 100
 - s-from-inequality-off1, 1096
 - s-from-inequality-on, 1064
- correctness-of-cancel-iplus, 47
- correctness-of-cancel-iplus-ile
 - ssp, 49
 - ssp-lemma, 48
 - ssp-lemma-off, 64
 - ssp-off, 111
 - ssp-off1, 1095
 - ssp-on, 1074
- correctness-of-cancel-iplus-off, 112
 - 1, 1095
- correctness-of-cancel-iplus-on, 1075
- correctness-of-cancel-itimes, 60
- correctness-of-cancel-itimes-fa
 - ctors, 79
 - ctors-off, 104
 - ctors-off1, 1095
 - ctors-on, 1068
- correctness-of-cancel-itimes-ha
 - ck-1, 58
 - ck-1-off, 64
 - ck-2, 59
 - ck-2-off, 65
 - ck-3, 59
 - ck-3-lemma, 59
 - ck-3-lemma-off, 60
 - ck-3-lemma-off1, 65
 - ck-3-off, 65
- correctness-of-cancel-itimes-ile
 - ssp, 63
 - ssp-factors, 83
 - ssp-factors-off, 103
 - ssp-factors-off1, 1095
 - ssp-factors-on, 1066
 - ssp-hack-1, 61
 - ssp-hack-1-off, 66
 - ssp-hack-2, 62
 - ssp-hack-2-lemma, 62
 - ssp-hack-2-lemma-off, 62
 - ssp-hack-2-off, 66
 - ssp-hack-3, 62
 - ssp-hack-3-lemma-1, 62
 - ssp-hack-3-lemma-1-off, 66
 - ssp-hack-3-lemma-2, 62
 - ssp-hack-3-lemma-2-off, 66
 - ssp-hack-3-off, 66
 - ssp-hack-4, 62
 - ssp-hack-4-off, 66
 - ssp-off, 87
 - ssp-off1, 1095
- correctness-of-cancel-itimes-of
 - f, 87
 - f1, 1095
- correctness-of-cancel-lessp-plu
 - s, 12
 - s-off, 127
 - s-off1, 1096
 - s-on, 1091
- correctness-of-cancel-lessp-time
 - s, 19
 - s-off, 125

- s-off1, 1096
- s-on, 1088
- correctness-of-cancel-quotient-ti
 - mes, 29
 - mes-off, 120
 - mes-off1, 1096
 - mes-on, 1083
- count-primitives, 436, 437
- count-primitives-off, 437
- cv, 703, 716
- cv*, 704, 713, 714
- cv-destructure-theory, 669
- cv-hyps, 592, 607, 609, 611, 613, 615,
 - 617, 618, 620, 622, 624, 626,
 - 628, 630, 632, 634, 636, 638,
 - 640, 642, 644, 646, 648, 650,
 - 652, 654, 656, 658, 660, 666,
 - 668–670, 722–732, 814, 816,
 - 817, 819, 821, 823, 825, 827,
 - 829, 831, 833, 835, 837, 839,
 - 841, 843, 845, 847, 849, 852,
 - 854, 856, 858, 860, 862, 863,
 - 865, 867, 869, 871, 875, 876,
 - 878, 879, 881, 883, 885–887,
 - 889–891, 894, 896, 898, 899,
 - 902, 903, 905, 908, 910, 912
- cv-off, 696
- cv-theory, 669
- cv\$netlist, 714, 721
- cv\$value, 714
- cv\$value-off, 716
- cv&, 713, 714, 721
- cv&-off, 714
- cv_decode, 615, 616, 669, 724, 824,
 - 827
- cv_decode-off, 615
- cv_decode\$destructure, 615
- cv_decode\$destructure-off, 616
- cv_fetch0, 607, 608, 669, 724, 725,
 - 729–732, 815, 816, 818, 829,
 - 835, 863, 869, 873, 888, 889,
 - 891, 894, 898, 899, 907, 910,
 - 912, 913
- cv_fetch0-off, 607
- cv_fetch0\$destructure, 607
- cv_fetch0\$destructure-off, 608
- cv_fetch1, 609, 610, 669, 670, 723,
 - 815, 817, 819, 820, 888, 890–
 - 892, 917, 927, 1013
- cv_fetch1-off, 609
- cv_fetch1\$destructure, 609
- cv_fetch1\$destructure-off, 610
- cv_fetch1\$rw-doan-matta, 670
- cv_fetch2, 611, 612, 669, 723, 820,
 - 822
- cv_fetch2-off, 611
- cv_fetch2\$destructure, 611
- cv_fetch2\$destructure-off, 612
- cv_fetch3, 613, 614, 669, 723, 724,
 - 822–826, 875–877
- cv_fetch3-off, 613
- cv_fetch3\$destructure, 613
- cv_fetch3\$destructure-off, 614
- cv_hold0, 654, 655, 670, 723, 731,
 - 820
- cv_hold0-off, 654
- cv_hold0\$destructure, 654
- cv_hold0\$destructure-off, 655
- cv_hold1, 656, 657, 670, 731
- cv_hold1-off, 656
- cv_hold1\$destructure, 656
- cv_hold1\$destructure-off, 657
- cv_reada0, 622, 623, 669, 724, 828,
 - 836, 893, 900
- cv_reada0-off, 622
- cv_reada0\$destructure, 622
- cv_reada0\$destructure-off, 623
- cv_reada1, 624, 625, 669, 726, 836,
 - 837
- cv_reada1-off, 624
- cv_reada1\$destructure, 624
- cv_reada1\$destructure-off, 625
- cv_reada2, 626, 627, 669, 726, 839,
 - 840
- cv_reada2-off, 626
- cv_reada2\$destructure, 626
- cv_reada2\$destructure-off, 627

cv_reada3, 628, 629, 669, 726, 840–842, 844, 845, 879, 880
 cv_reada3-off, 628
 cv_reada3\$destructure, 628
 cv_reada3\$destructure-off, 629
 cv_readb0, 630, 631, 669, 724, 727, 828, 843, 846, 893, 902, 903
 cv_readb0-off, 630
 cv_readb0\$destructure, 630
 cv_readb0\$destructure-off, 631
 cv_readb1, 632, 633, 669, 727, 847, 848
 cv_readb1-off, 632
 cv_readb1\$destructure, 632
 cv_readb1\$destructure-off, 633
 cv_readb2, 634, 635, 669, 727, 848, 849
 cv_readb2-off, 634
 cv_readb2\$destructure, 634
 cv_readb2\$destructure-off, 635
 cv_readb3, 636, 637, 669, 728, 851–856, 882–884
 cv_readb3-off, 636
 cv_readb3\$destructure, 636
 cv_readb3\$destructure-off, 637
 cv_rega, 617, 669, 724, 828, 830, 893, 895
 cv_rega-off, 617
 cv_rega\$destructure, 617
 cv_rega\$destructure-off, 617
 cv_regb, 618, 619, 669, 725, 831, 832, 896, 897
 cv_regb-off, 618
 cv_regb\$destructure, 618
 cv_regb\$destructure-off, 619
 cv_reset0, 662, 663, 670, 731, 733
 cv_reset0-off, 662
 cv_reset0\$destructure, 662
 cv_reset0\$destructure-off, 663
 cv_reset1, 664, 665, 670, 732
 cv_reset1-off, 664
 cv_reset1\$destructure, 664
 cv_reset1\$destructure-off, 665
 cv_reset2, 666, 667, 670, 732
 cv_reset2-off, 666
 cv_reset2\$destructure, 666
 cv_reset2\$destructure-off, 667
 cv_sefa0, 646, 647, 670, 724, 829, 866, 894, 909
 cv_sefa0-off, 646
 cv_sefa0\$destructure, 646
 cv_sefa0\$destructure-off, 647
 cv_sefa1, 648, 649, 670, 729, 867, 868
 cv_sefa1-off, 648
 cv_sefa1\$destructure, 648
 cv_sefa1\$destructure-off, 650
 cv_sefb0, 650, 651, 670, 724, 730, 829, 869, 870, 894, 910
 cv_sefb0-off, 650
 cv_sefb0\$destructure, 650
 cv_sefb0\$destructure-off, 651
 cv_sefb1, 652, 653, 670, 725, 730, 835, 871, 872, 898, 899, 912
 cv_sefb1-off, 652
 cv_sefb1\$destructure, 652
 cv_sefb1\$destructure-off, 654
 cv_update, 620, 621, 669, 725, 727, 728, 831, 833, 834, 842, 843, 853, 896, 898, 902, 905
 cv_update-off, 620
 cv_update\$destructure, 620
 cv_update\$destructure-off, 621
 cv_v11010, 658, 659, 670
 cv_v11010-off, 658
 cv_v11010\$destructure, 658
 cv_v11010\$destructure-off, 659
 cv_v11011, 660, 661, 670
 cv_v11011-off, 660
 cv_v11011\$destructure, 660
 cv_v11011\$destructure-off, 661
 cv_v11111, 668–670
 cv_v11111-off, 667
 cv_v11111\$destructure, 668
 cv_v11111\$destructure-off, 669
 cv_write0, 638, 639, 669, 725, 727, 728, 831, 843, 853, 857, 896, 902, 905, 906

cv_write0-off, 638
 cv_write0\$destructure, 638
 cv_write0\$destructure-off, 639
 cv_write1, 640, 641, 670, 728, 857, 858
 cv_write1-off, 640
 cv_write1\$destructure, 640
 cv_write1\$destructure-off, 642
 cv_write2, 642, 643, 670, 728, 860
 cv_write2-off, 642
 cv_write2\$destructure, 642
 cv_write2\$destructure-off, 644
 cv_write3, 644, 645, 670, 729, 861–865, 885–887
 cv_write3-off, 644
 cv_write3\$destructure, 644
 cv_write3\$destructure-off, 645
 cvzbv, 482–484, 536, 1096, 1099–1101, 1104–1106, 1216
 cvzbv-dec, 483, 484, 1101, 1105
 cvzbv-inc, 483, 484, 1100, 1104
 cvzbv-neg, 483, 484, 1105
 cvzbv-v-adder, 483–485, 1100, 1104
 cvzbv-v-asr, 484, 1105
 cvzbv-v-lsl, 483
 cvzbv-v-lsr, 484, 1101
 cvzbv-v-not, 483, 484, 1101, 1106
 cvzbv-v-ror, 483, 484
 cvzbv-v-subtractor, 483–485, 1100, 1105

 data-in-select, 605, 608, 609, 611, 613, 615, 617, 619, 620, 622, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 662, 664, 666, 668, 761, 762, 781, 782, 793, 802, 803, 811, 812
 data-in-select-off, 605
 data-out, 759, 761–764
 data-out-off, 760
 dec-addr-out, 605, 608, 610, 611, 613, 615, 617, 619, 621, 622, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 662, 664, 666, 668, 763, 782, 794, 804, 813
 dec-addr-out-off, 605
 dec-pass*, 474, 475
 dec-pass*-off, 474
 dec-pass*\$destructure, 474
 dec-pass*\$destructure-off, 474
 dec-pass-cell*, 469, 473
 dec-pass-cell\$netlist, 469
 dec-pass-cell\$value, 469
 dec-pass-cell&, 469, 471
 dec-pass-cell&-off, 469
 dec-pass\$netlist, 475, 776
 dec-pass\$value, 475
 dec-pass\$value-off, 475
 dec-pass&, 474, 475, 776
 dec-pass&-off, 475
 decode-5, 573, 576, 578, 716
 decode-5*, 576, 578
 decode-5-off, 576
 decode-5\$netlist, 578, 721
 decode-5\$value, 578
 decode-5\$value-off, 578
 decode-5&, 578, 721
 decode-5&-off, 578
 decode-gen, 490, 493, 494
 decode-gen*, 491, 492
 decode-gen\$netlist, 492, 495
 decode-gen\$value, 493
 decode-gen\$value-off, 493
 decode-gen\$value-zero, 493
 decode-gen\$value-zero-off, 493
 decode-gen&, 492, 493, 495
 decode-gen&-off, 492
 decode-mode, 486, 487, 494
 decode-mode*, 486, 487
 decode-mode\$netlist, 487, 495
 decode-mode\$value, 487
 decode-mode\$value-off, 487
 decode-mode\$value-zero, 487
 decode-mode\$value-zero-off, 487

decode-mode&, 487, 495
 decode-mode&-off, 487
 decode-prop, 487, 490, 494
 decode-prop*, 488, 489
 decode-prop\$netlist, 489, 495
 decode-prop\$value, 489
 decode-prop\$value-off, 490
 decode-prop\$value-zero, 490
 decode-prop\$value-zero-off, 490
 decode-prop&, 489, 490, 495
 decode-prop&-off, 489
 decode-reg-mode*, 565
 decode-reg-mode\$netlist, 565, 672
 decode-reg-mode\$value, 565
 decode-reg-mode\$value-off, 565
 decode-reg-mode&, 565, 672
 decode-reg-mode&-off, 565
 decode\$step, 827
 del10-monotone, 947
 del10-monotone-off, 1046
 del10-monotone-on, 1136
 del10-preserves-good-s, 1006
 del10-preserves-good-s-off, 1036
 del10-preserves-good-s-on, 1125
 del10\$netlist, 400
 del10\$value, 400
 del10\$value-off, 400
 del10&, 400
 del10&-off, 400
 del2-monotone, 947
 del2-monotone-off, 1047
 del2-monotone-on, 1136
 del2-preserves-good-s, 1006
 del2-preserves-good-s-off, 1036
 del2-preserves-good-s-on, 1125
 del2\$netlist, 400
 del2\$value, 400
 del2\$value-off, 400
 del2&, 399, 400
 del2&-off, 399
 del4-monotone, 947
 del4-monotone-off, 1046
 del4-monotone-on, 1136
 del4-preserves-good-s, 1006
 del4-preserves-good-s-off, 1036
 del4-preserves-good-s-on, 1125
 del4\$netlist, 400, 477, 480
 del4\$value, 400
 del4\$value-off, 400
 del4&, 400, 476, 480
 del4&-off, 400
 delay-dependencies, 318, 319, 322, 323, 325, 332, 333
 delay-dependencies-off, 318
 delay-dependencies-off1, 322
 delay-dependencies-on, 321
 delay-hl, 317, 322
 delay-hl-off, 317
 delay-intercept-hl, 318, 323, 325, 331
 delay-intercept-hl-off, 318
 delay-intercept-lh, 318, 323, 325, 331
 delay-intercept-lh-off, 318
 delay-lh, 317, 322
 delay-lh-off, 317
 delay-slope-hl, 318, 323, 331
 delay-slope-hl-off, 318
 delay-slope-lh, 318, 323, 331
 delay-slope-lh-off, 318
 delay-to-range, 326, 327
 delay-to-range-off, 326
 delays-error, 334, 349
 delays-error-off, 334
 delete, 2–4, 8, 10–12, 17, 44–46, 57, 58, 78, 80
 delete*, 138, 250, 357, 367, 375
 delete*-count, 250
 delete*-lemma, 250
 delete*-off, 138
 delete-delete, 3
 delete-delete-off, 131
 delete-delete-off1, 1182
 delete-delete-on, 1094
 delete-module, 237, 239–241, 247, 248, 256, 258–260, 263, 265, 266, 268–270, 272–274, 419, 429–431, 433, 434, 436–439, 441, 445, 450, 451, 454–456, 458, 460, 462, 463, 469, 471–476,

480, 487, 489, 492, 495, 497,
 499, 500, 502, 514–516, 518,
 522, 524, 528, 540, 560, 562,
 564–570, 578, 582, 672, 688,
 713, 714, 721, 738, 744, 748,
 750, 752, 755, 757, 758, 775,
 776, 790, 791, 799, 800, 998–
 1001
 delete-module-lookup-module-cou
 nt, 237
 delete-module-off, 237
 delete-non-member, 3
 delete-non-member-off, 131
 delete-non-member-off1, 1182
 delete-non-member-on, 1094
 delete-null-entries, 378, 379, 386
 delete-null-entries-off, 379
 delete-off, 131
 delete-off1, 1182
 delete-on, 1094
 dependency-table, 381
 dependency-table-off, 381
 determined-delayp, 319
 determined-delayp-off, 319
 determined-rangep, 320, 325, 327
 determined-rangep-off, 320
 diff-diff-arg1, 7
 diff-diff-arg1-off, 129
 diff-diff-arg1-off1, 1180
 diff-diff-arg1-on, 1092
 diff-diff-arg2, 7
 diff-diff-arg2-off, 129
 diff-diff-arg2-off1, 1180
 diff-diff-arg2-on, 1092
 diff-diff-diff, 7
 diff-diff-diff-off, 7
 diff-sub1-arg2, 6
 diff-sub1-arg2-off, 7
 difference-0, 134
 difference-add1-add1, 134
 difference-add1-arg2, 11
 difference-add1-arg2-off, 128
 difference-add1-arg2-off1, 1179
 difference-add1-arg2-on, 1091
 difference-cancellation, 5
 difference-cancellation-off, 5
 difference-difference-arg1, 11
 difference-difference-arg1-off, 128
 difference-difference-arg1-off1, 1179
 difference-difference-arg1-on, 1091
 difference-difference-arg2, 11
 difference-difference-arg2-off, 128
 difference-difference-arg2-off1, 1179
 difference-difference-arg2-on, 1091
 difference-elim, 11
 difference-elim-off, 128
 difference-elim-off1, 1179
 difference-elim-on, 1091
 difference-idifference, 95
 difference-idifference-off, 96
 difference-leq-arg1, 11
 difference-leq-arg1-off, 128
 difference-leq-arg1-off1, 1179
 difference-leq-arg1-on, 1091
 difference-lessp-arg1, 7
 difference-lessp-arg1-off, 7
 difference-linear, 1194
 difference-off, 133
 difference-plus-cancellation, 6
 difference-plus-cancellation-of
 f, 6
 difference-plus-cancellation-pr
 oof, 6
 oof-off, 129
 oof-off1, 1180
 oof-on, 1092
 difference-plus-plus-cancellati
 on, 6
 on-hack, 6
 on-hack-off, 6
 on-off, 6
 on-proof, 6
 on-proof-off, 129
 on-proof-off1, 1180
 on-proof-on, 1092
 difference-sub1-arg2, 11
 difference-sub1-arg2-off, 128
 difference-sub1-arg2-off1, 1179

- difference-sub1-arg2-on, 1091
- difference-x-1, 133
- difference-x-x, 11
- difference-x-x-off, 128
- difference-x-x-off1, 1179
- difference-x-x-on, 1091
- digitp, 284, 285
- digitp-off, 284
- disjoin-equalities-with-0, 83, 84
- disjoin-equalities-with-0-off, 103
- disjoin-equalities-with-0-off1, 1155
- disjoin-equalities-with-0-on, 1066
- disjoint, 138–141, 144, 149, 167, 255, 296, 427
- disjoint-append, 139
- disjoint-cons, 139
- disjoint-firstn, 140
- disjoint-firstn-restn-lemmas, 141
- disjoint-firstn1, 140
- disjoint-indices-different-names, 149
- disjoint-nlistp, 139
- disjoint-off, 139
- disjoint-or-err, 252, 289, 291, 338
- disjoint-or-err-off, 252
- disjoint-restn, 141
- disjoint-restn1, 141
- disjoint-subrange, 144
- distributivity-of-times-over-gcd, 34
 - d-off, 117
 - d-off1, 1169
 - d-on, 1080
 - d-proof, 34
 - d-proof-off, 117
 - d-proof-off1, 1169
 - d-proof-on, 1080
- divides-plus-plus, 1097
- divides-plus-plus-off, 1115
- division-theorem, 50
- division-theorem-for-truncate-to-neginf, 51
 - o-neginf-off, 109
 - o-neginf-off1, 1161
 - o-neginf-on, 1073
 - o-neginf-part1, 51
 - o-neginf-part1-off, 110
 - o-neginf-part1-off1, 1162
 - o-neginf-part1-on, 1073
 - o-neginf-part2, 51
 - o-neginf-part2-off, 109
 - o-neginf-part2-off1, 1161
 - o-neginf-part2-on, 1073
 - o-neginf-part3, 51
 - o-neginf-part3-off, 109
 - o-neginf-part3-off1, 1161
 - o-neginf-part3-on, 1073
 - o-zero, 52
 - o-zero-off, 109
 - o-zero-off1, 1161
 - o-zero-on, 1072
 - o-zero-part1, 51
 - o-zero-part1-off, 109
 - o-zero-part1-off1, 1161
 - o-zero-part1-on, 1072
 - o-zero-part2, 51
 - o-zero-part2-off, 109
 - o-zero-part2-off1, 1161
 - o-zero-part2-on, 1072
 - o-zero-part3, 52
 - o-zero-part3-off, 109
 - o-zero-part3-off1, 1161
 - o-zero-part3-on, 1072
- division-theorem-off, 110
- division-theorem-off1, 1162
- division-theorem-on, 1073
- division-theorem-part1, 50
- division-theorem-part1-off, 110
- division-theorem-part1-off1, 1162
- division-theorem-part1-on, 1073
- division-theorem-part2, 50
- division-theorem-part2-off, 110
- division-theorem-part2-off1, 1162
- division-theorem-part2-on, 1073
- division-theorem-part3, 50
- division-theorem-part3-off, 110
- division-theorem-part3-off1, 1162
- division-theorem-part3-on, 1073

- double-cdr-induction, 1002
- double-cdr-induction-off, 1040
- double-cdr-induction-on, 1129
- double-difference-induction, 1193
- double-log-induction, 32
- double-log-induction-off, 118
- double-log-induction-off1, 1170
- double-log-induction-on, 1082
- double-number-induction, 30
- double-number-induction-off, 120
- double-number-induction-off1, 1171
- double-number-induction-on, 1083
- double-remainder-induction, 23
- double-remainder-induction-off, 123
- double-remainder-induction-off1, 1175
- double-remainder-induction-on, 1086
- doublet-n-simulate, 1010
- doublet-n-simulate-induction, 1010
- doublet-n-simulate-induction-of
f, 1033
- doublet-n-simulate-induction-on, 1122
- doublet-n-simulate-off, 1033
- doublet-n-simulate-on, 1122
- doublet-p, 1010
- doublet-p-equal-approx, 1010
- doublet-p-equal-approx-off, 1033
- doublet-p-equal-approx-on, 1122
- doublet-p-off, 1010
- doublet-p-off1, 1152
- doublet-p-on, 1063
- dp-ram-16x32-args-crock, 420
- dp-ram-16x32-args-crock-off, 420
- dp-ram-16x32-monotone, 993
- dp-ram-16x32-monotone-off, 998
- dp-ram-16x32-monotone-off1, 1152
- dp-ram-16x32-monotone-on, 1063
- dp-ram-16x32-monotone-state, 993
- dp-ram-16x32-monotone-state-off, 997
1, 1152
- dp-ram-16x32-monotone-state-on, 1062
- dp-ram-16x32-monotone-value, 972
- dp-ram-16x32-monotone-value-off, 995
1, 1150
- dp-ram-16x32-monotone-value-on, 1061
- dp-ram-16x32-preserves-good-s, 1007
- dp-ram-16x32-preserves-good-s-o
ff, 1034
n, 1123
- dp-ram-16x32\$netlist, 401, 739
- dp-ram-16x32\$state, 412
- dp-ram-16x32\$state-off, 414
- dp-ram-16x32\$structured-state, 420
- dp-ram-16x32\$structured-state-
off, 421
- dp-ram-16x32\$structured-value, 420
- dp-ram-16x32\$structured-value-
off, 420
- dp-ram-16x32\$value, 401
- dp-ram-16x32\$value-off, 403
- dp-ram-16x32&, 401, 412, 420, 738
- dp-ram-16x32&-off, 401
- drive-lessp, 314, 315
- drive-lessp-off, 314
- drive-min, 315, 316
- drive-min-off, 315
- drives-error, 317, 349
- drives-error-off, 317
- dtack-, 760–763
- dtack-off, 760
- dual-apply-state, 239, 240, 938, 976,
991, 993, 1004–1008
- dual-apply-state-dp-ram-16x32, 993
- dual-apply-state-dp-ram-16x32-le
mma-1, 976
mma-1-off, 997
mma-1-off1, 1151
mma-1-on, 1062
mma-2, 976
mma-2-off, 997
mma-2-off1, 1151
mma-2-on, 1062
mma-3, 991
mma-3-off, 997
mma-3-off1, 1152
mma-3-on, 1062
- dual-apply-state-dp-ram-16x32-o
ff, 997
ff1, 1152

- n, 1062
- dual-apply-state-off, 239
- dual-apply-value, 238, 239, 938, 954, 955, 970, 972
- dual-apply-value-dp-ram-16x32, 972
- dual-apply-value-dp-ram-16x32-le
 - mma-1, 954
 - mma-1-off, 995
 - mma-1-off1, 1149
 - mma-1-on, 1060
 - mma-2, 955
 - mma-2-off, 995
 - mma-2-off1, 1150
 - mma-2-on, 1060
 - mma-3, 970
 - mma-3-off, 995
 - mma-3-off1, 1150
 - mma-3-on, 1060
- dual-apply-value-dp-ram-16x32-off, 995
- ff1, 1150
- n, 1060
- dual-apply-value-off, 238
- dual-eval, 239–244, 391–400, 402–404, 406, 408–413, 415, 417, 419–422, 424, 426, 427, 429, 430, 432, 433, 435, 437–440, 442, 443, 445, 452, 456, 459, 461, 462, 464, 469, 473, 475, 478, 481, 482, 487, 490, 493, 495, 496, 498–502, 516–519, 522, 523, 525, 529, 541, 561, 563–565, 567–570, 578, 583, 589, 590, 592, 673, 690, 715, 722, 740, 741, 744–746, 749, 751, 753, 755, 756, 758, 759, 781, 784, 792, 795, 801, 806, 809, 931–933, 938–952, 999–1003, 1008–1010
- dual-eval-0-primp, 938
- dual-eval-0-primp-off, 1054
- dual-eval-0-primp-on, 1143
- dual-eval-2-primp, 938
- dual-eval-2-primp-off, 1054
- dual-eval-2-primp-on, 1143
- dual-eval-ao2-state, 939
- dual-eval-ao2-state-off, 1053
- dual-eval-ao2-state-on, 1142
- dual-eval-ao2-value, 938
- dual-eval-ao2-value-off, 1053
- dual-eval-ao2-value-on, 1142
- dual-eval-ao4-state, 939
- dual-eval-ao4-state-off, 1053
- dual-eval-ao4-state-on, 1142
- dual-eval-ao4-value, 939
- dual-eval-ao4-value-off, 1053
- dual-eval-ao4-value-on, 1142
- dual-eval-ao6-state, 939
- dual-eval-ao6-state-off, 1053
- dual-eval-ao6-state-on, 1142
- dual-eval-ao6-value, 939
- dual-eval-ao6-value-off, 1053
- dual-eval-ao6-value-on, 1142
- dual-eval-ao7-state, 939
- dual-eval-ao7-state-off, 1053
- dual-eval-ao7-state-on, 1142
- dual-eval-ao7-value, 939
- dual-eval-ao7-value-off, 1053
- dual-eval-ao7-value-on, 1142
- dual-eval-b-and-state, 939
- dual-eval-b-and-state-off, 1052
- dual-eval-b-and-state-on, 1142
- dual-eval-b-and-value, 939
- dual-eval-b-and-value-off, 1052
- dual-eval-b-and-value-on, 1142
- dual-eval-b-and3-state, 940
- dual-eval-b-and3-state-off, 1052
- dual-eval-b-and3-state-on, 1141
- dual-eval-b-and3-value, 940
- dual-eval-b-and3-value-off, 1052
- dual-eval-b-and3-value-on, 1142
- dual-eval-b-and4-state, 940
- dual-eval-b-and4-state-off, 1052
- dual-eval-b-and4-state-on, 1141
- dual-eval-b-and4-value, 940
- dual-eval-b-and4-value-off, 1052
- dual-eval-b-and4-value-on, 1141
- dual-eval-b-equiv-state, 940

dual-eval-b-equiv-state-off, 1052	dual-eval-b-nand6-value, 942
dual-eval-b-equiv-state-on, 1141	dual-eval-b-nand6-value-off, 1050
dual-eval-b-equiv-value, 940	dual-eval-b-nand6-value-on, 1140
dual-eval-b-equiv-value-off, 1052	dual-eval-b-nand8-state, 942
dual-eval-b-equiv-value-on, 1141	dual-eval-b-nand8-state-off, 1050
dual-eval-b-equiv3-state, 940	dual-eval-b-nand8-state-on, 1139
dual-eval-b-equiv3-state-off, 1052	dual-eval-b-nand8-value, 942
dual-eval-b-equiv3-state-on, 1141	dual-eval-b-nand8-value-off, 1050
dual-eval-b-equiv3-value, 940	dual-eval-b-nand8-value-on, 1139
dual-eval-b-equiv3-value-off, 1052	dual-eval-b-nbuf-state, 943
dual-eval-b-equiv3-value-on, 1141	dual-eval-b-nbuf-state-off, 1050
dual-eval-b-if-state, 941	dual-eval-b-nbuf-state-on, 1139
dual-eval-b-if-state-off, 1051	dual-eval-b-nbuf-value, 943
dual-eval-b-if-state-on, 1141	dual-eval-b-nbuf-value-off, 1050
dual-eval-b-if-value, 941	dual-eval-b-nbuf-value-on, 1139
dual-eval-b-if-value-off, 1051	dual-eval-b-nor-state, 943
dual-eval-b-if-value-on, 1141	dual-eval-b-nor-state-off, 1050
dual-eval-b-nand-state, 941	dual-eval-b-nor-state-on, 1139
dual-eval-b-nand-state-off, 1051	dual-eval-b-nor-value, 943
dual-eval-b-nand-state-on, 1140	dual-eval-b-nor-value-off, 1050
dual-eval-b-nand-value, 941	dual-eval-b-nor-value-on, 1139
dual-eval-b-nand-value-off, 1051	dual-eval-b-nor3-state, 943
dual-eval-b-nand-value-on, 1141	dual-eval-b-nor3-state-off, 1049
dual-eval-b-nand3-state, 941	dual-eval-b-nor3-state-on, 1139
dual-eval-b-nand3-state-off, 1051	dual-eval-b-nor3-value, 943
dual-eval-b-nand3-state-on, 1140	dual-eval-b-nor3-value-off, 1050
dual-eval-b-nand3-value, 941	dual-eval-b-nor3-value-on, 1139
dual-eval-b-nand3-value-off, 1051	dual-eval-b-nor4-state, 943
dual-eval-b-nand3-value-on, 1140	dual-eval-b-nor4-state-off, 1049
dual-eval-b-nand4-state, 941	dual-eval-b-nor4-state-on, 1139
dual-eval-b-nand4-state-off, 1051	dual-eval-b-nor4-value, 943
dual-eval-b-nand4-state-on, 1140	dual-eval-b-nor4-value-off, 1049
dual-eval-b-nand4-value, 941	dual-eval-b-nor4-value-on, 1139
dual-eval-b-nand4-value-off, 1051	dual-eval-b-nor5-state, 944
dual-eval-b-nand4-value-on, 1140	dual-eval-b-nor5-state-off, 1049
dual-eval-b-nand5-state, 942	dual-eval-b-nor5-state-on, 1138
dual-eval-b-nand5-state-off, 1051	dual-eval-b-nor5-value, 944
dual-eval-b-nand5-state-on, 1140	dual-eval-b-nor5-value-off, 1049
dual-eval-b-nand5-value, 942	dual-eval-b-nor5-value-on, 1138
dual-eval-b-nand5-value-off, 1051	dual-eval-b-nor6-state, 944
dual-eval-b-nand5-value-on, 1140	dual-eval-b-nor6-state-off, 1049
dual-eval-b-nand6-state, 942	dual-eval-b-nor6-state-on, 1138
dual-eval-b-nand6-state-off, 1050	dual-eval-b-nor6-value, 944
dual-eval-b-nand6-state-on, 1140	dual-eval-b-nor6-value-off, 1049

dual-eval-b-nor6-value-on, 1138
 dual-eval-b-nor8-state, 944
 dual-eval-b-nor8-state-off, 1049
 dual-eval-b-nor8-state-on, 1138
 dual-eval-b-nor8-value, 944
 dual-eval-b-nor8-value-off, 1049
 dual-eval-b-nor8-value-on, 1138
 dual-eval-b-not-b4ip-state, 945
 dual-eval-b-not-b4ip-state-off, 1048
 dual-eval-b-not-b4ip-state-on, 1137
 dual-eval-b-not-b4ip-value, 945
 dual-eval-b-not-b4ip-value-off, 1048
 dual-eval-b-not-b4ip-value-on, 1138
 dual-eval-b-not-ivap-state, 945
 dual-eval-b-not-ivap-state-off, 1048
 dual-eval-b-not-ivap-state-on, 1137
 dual-eval-b-not-ivap-value, 945
 dual-eval-b-not-ivap-value-off, 1048
 dual-eval-b-not-ivap-value-on, 1137
 dual-eval-b-not-state, 945
 dual-eval-b-not-state-off, 1048
 dual-eval-b-not-state-on, 1138
 dual-eval-b-not-value, 945
 dual-eval-b-not-value-off, 1048
 dual-eval-b-not-value-on, 1138
 dual-eval-b-or-state, 945
 dual-eval-b-or-state-off, 1048
 dual-eval-b-or-state-on, 1137
 dual-eval-b-or-value, 945
 dual-eval-b-or-value-off, 1048
 dual-eval-b-or-value-on, 1137
 dual-eval-b-or3-state, 946
 dual-eval-b-or3-state-off, 1048
 dual-eval-b-or3-state-on, 1137
 dual-eval-b-or3-value, 945
 dual-eval-b-or3-value-off, 1048
 dual-eval-b-or3-value-on, 1137
 dual-eval-b-or4-state, 946
 dual-eval-b-or4-state-off, 1047
 dual-eval-b-or4-state-on, 1137
 dual-eval-b-or4-value, 946
 dual-eval-b-or4-value-off, 1047
 dual-eval-b-or4-value-on, 1137
 dual-eval-b-xor-state, 946
 dual-eval-b-xor-state-off, 1047
 dual-eval-b-xor-state-on, 1136
 dual-eval-b-xor-value, 946
 dual-eval-b-xor-value-off, 1047
 dual-eval-b-xor-value-on, 1136
 dual-eval-b-xor3-state, 946
 dual-eval-b-xor3-state-off, 1047
 dual-eval-b-xor3-state-on, 1136
 dual-eval-b-xor3-value, 946
 dual-eval-b-xor3-value-off, 1047
 dual-eval-b-xor3-value-on, 1136
 dual-eval-body-bindings, 242, 243, 428,
 477, 481, 750, 752, 754, 756
 dual-eval-body-bindings-off, 243
 dual-eval-del10-state, 947
 dual-eval-del10-state-off, 1046
 dual-eval-del10-state-on, 1136
 dual-eval-del10-value, 947
 dual-eval-del10-value-off, 1046
 dual-eval-del10-value-on, 1136
 dual-eval-del2-state, 946
 dual-eval-del2-state-off, 1047
 dual-eval-del2-state-on, 1136
 dual-eval-del2-value, 946
 dual-eval-del2-value-off, 1047
 dual-eval-del2-value-on, 1136
 dual-eval-del4-state, 947
 dual-eval-del4-state-off, 1047
 dual-eval-del4-state-on, 1136
 dual-eval-del4-value, 947
 dual-eval-del4-value-off, 1047
 dual-eval-del4-value-on, 1136
 dual-eval-fd1-state, 947
 dual-eval-fd1-state-off, 1046
 dual-eval-fd1-state-on, 1135
 dual-eval-fd1-value, 947
 dual-eval-fd1-value-off, 1046
 dual-eval-fd1-value-on, 1135
 dual-eval-fd1s-state, 948
 dual-eval-fd1s-state-off, 1046
 dual-eval-fd1s-state-on, 1135
 dual-eval-fd1s-value, 948
 dual-eval-fd1s-value-off, 1046
 dual-eval-fd1s-value-on, 1135

dual-eval-fd1slp-state, 948
 dual-eval-fd1slp-state-off, 1045
 dual-eval-fd1slp-state-on, 1135
 dual-eval-fd1slp-value, 948
 dual-eval-fd1slp-value-off, 1045
 dual-eval-fd1slp-value-on, 1135
 dual-eval-fd1sp-state, 948
 dual-eval-fd1sp-state-off, 1046
 dual-eval-fd1sp-state-on, 1135
 dual-eval-fd1sp-value, 948
 dual-eval-fd1sp-value-off, 1046
 dual-eval-fd1sp-value-on, 1135
 dual-eval-id-state, 948
 dual-eval-id-state-off, 1045
 dual-eval-id-state-on, 1134
 dual-eval-id-value, 948
 dual-eval-id-value-off, 1045
 dual-eval-id-value-on, 1134
 dual-eval-monotone, 1003
 dual-eval-monotone-induction, 998–1001
 dual-eval-monotone-induction-of
 f, 1041
 dual-eval-monotone-induction-on, 1130
 dual-eval-monotone-no-ram, 1003
 dual-eval-monotone-no-ram-off, 1039
 dual-eval-monotone-no-ram-on, 1128
 dual-eval-monotone-off, 1039
 dual-eval-monotone-on, 1128
 dual-eval-off, 240
 dual-eval-on-collected-nth-32, 589
 dual-eval-procmon-state, 947
 dual-eval-procmon-state-off, 1046
 dual-eval-procmon-state-on, 1135
 dual-eval-procmon-value, 947
 dual-eval-procmon-value-off, 1046
 dual-eval-procmon-value-on, 1135
 dual-eval-pullup-state, 949
 dual-eval-pullup-state-off, 1044
 dual-eval-pullup-state-on, 1134
 dual-eval-pullup-value, 949
 dual-eval-pullup-value-off, 1044
 dual-eval-pullup-value-on, 1134
 dual-eval-ram-enable-circuit-st
 ate, 949
 ate-off, 1045
 ate-on, 1134
 dual-eval-ram-enable-circuit-va
 lue, 948
 lue-off, 1045
 lue-on, 1134
 dual-eval-t-buf-state, 949
 dual-eval-t-buf-state-off, 1045
 dual-eval-t-buf-state-on, 1134
 dual-eval-t-buf-value, 949
 dual-eval-t-buf-value-off, 1045
 dual-eval-t-buf-value-on, 1134
 dual-eval-t-wire-state, 949
 dual-eval-t-wire-state-off, 1045
 dual-eval-t-wire-state-on, 1134
 dual-eval-t-wire-value, 949
 dual-eval-t-wire-value-off, 1045
 dual-eval-t-wire-value-on, 1134
 dual-eval-ttl-bidirect-state, 950
 dual-eval-ttl-bidirect-state-of
 f, 1044
 dual-eval-ttl-bidirect-state-on, 1133
 dual-eval-ttl-bidirect-value, 949
 dual-eval-ttl-bidirect-value-of
 f, 1044
 dual-eval-ttl-bidirect-value-on, 1133
 dual-eval-ttl-clk-input-state, 950
 dual-eval-ttl-clk-input-state-o
 ff, 1044
 n, 1133
 dual-eval-ttl-clk-input-value, 950
 dual-eval-ttl-clk-input-value-o
 ff, 1044
 n, 1133
 dual-eval-ttl-input-state, 950
 dual-eval-ttl-input-state-off, 1044
 dual-eval-ttl-input-state-on, 1133
 dual-eval-ttl-input-value, 950
 dual-eval-ttl-input-value-off, 1044
 dual-eval-ttl-input-value-on, 1133
 dual-eval-ttl-output-fast-state, 951
 -off, 1043
 -on, 1132

- dual-eval-ttl-output-fast-value, 951
 - off, 1043
 - on, 1132
- dual-eval-ttl-output-parametric
 - state, 950
 - state-off, 1043
 - state-on, 1132
 - value, 950
 - value-off, 1043
 - value-on, 1132
- dual-eval-ttl-output-state, 950
- dual-eval-ttl-output-state-off, 1043
- dual-eval-ttl-output-state-on, 1133
- dual-eval-ttl-output-value, 950
- dual-eval-ttl-output-value-off, 1043
- dual-eval-ttl-output-value-on, 1133
- dual-eval-ttl-tri-output-fast-state, 951
 - ate-off, 1042
 - ate-on, 1132
- dual-eval-ttl-tri-output-fast-value, 951
 - alue-off, 1043
 - alue-on, 1132
- dual-eval-ttl-tri-output-state, 951
- dual-eval-ttl-tri-output-state-off, 1043
 - on, 1132
- dual-eval-ttl-tri-output-value, 951
- dual-eval-ttl-tri-output-value-off, 1043
 - on, 1132
- dual-eval-vdd-parametric-state, 952
- dual-eval-vdd-parametric-state-off, 1042
 - on, 1131
- dual-eval-vdd-parametric-value, 952
- dual-eval-vdd-parametric-value-off, 1042
 - on, 1131
- dual-eval-vdd-state, 952
- dual-eval-vdd-state-off, 1042
- dual-eval-vdd-state-on, 1131
- dual-eval-vdd-value, 952
- dual-eval-vdd-value-off, 1042
- dual-eval-vdd-value-on, 1132
- dual-eval-vss-state, 952
- dual-eval-vss-state-off, 1042
- dual-eval-vss-state-on, 1131
- dual-eval-vss-value, 952
- dual-eval-vss-value-off, 1042
- dual-eval-vss-value-on, 1131
- dual-port-ram-state, 174, 414, 421, 740, 807, 976, 991–993
- dual-port-ram-state-body, 993
- dual-port-ram-state-body-off, 1041
- dual-port-ram-state-body-on, 1130
- dual-port-ram-state-is-dual-port
 - ram-state-body, 993
 - ram-state-body-off, 997
 - ram-state-body-off1, 1152
 - ram-state-body-on, 1062
- dual-port-ram-state-monotone, 976
- dual-port-ram-state-monotone-off, 1041
- dual-port-ram-state-monotone-on, 1130
- dual-port-ram-state-monotone-re
 - write, 976
 - write-off, 997
 - write-off1, 1151
 - write-on, 1062
- dual-port-ram-value, 174, 175, 403, 420, 739, 954, 970–972
- dual-port-ram-value-body, 971
- dual-port-ram-value-body-off, 1041
- dual-port-ram-value-body-on, 1131
- dual-port-ram-value-is-dual-port
 - ram-value-body, 971
 - ram-value-body-off, 995
 - ram-value-body-off1, 1150
 - ram-value-body-on, 1060
- dual-port-ram-value-monotone, 954
- dual-port-ram-value-monotone-off, 994
 - f1, 1149
- dual-port-ram-value-monotone-on, 1060
- duplicates-append, 139

- duplicates?, 139–141, 149, 167–169, 254
- duplicates?-cons, 139
- duplicates?-off, 139
- encode-32, 578, 579, 586, 587
- encode-32*, 581, 582
- encode-32-off, 580
- encode-32\$netlist, 582, 714
- encode-32\$value, 582
- encode-32\$value-off, 584
- encode-32\$value-on-a-vector, 590
- encode-32\$value-on-a-vector-of f, 591
- encode-32&, 582, 590, 714
- encode-32&-off, 582
- equal-0-itimes-list-eval\$-bagi
 - nt-1, 59
 - nt-1-off, 65
 - nt-2, 59
 - nt-2-off, 65
- equal-append-x-x, 135
- equal-collect-value-promote-ali
 - sts, 167
- equal-difference-0, 5
- equal-difference-0-off, 130
- equal-difference-0-off1, 1181
- equal-difference-0-on, 1093
- equal-exp-0, 31
- equal-exp-0-off, 119
- equal-exp-0-off1, 1171
- equal-exp-0-on, 1082
- equal-exp-1, 31
- equal-exp-1-off, 119
- equal-exp-1-off1, 1170
- equal-exp-1-on, 1082
- equal-fix-int, 58
- equal-fix-int-off, 106
- equal-fix-int-off1, 1158
- equal-fix-int-on, 1070
- equal-fix-int-to-ilessp, 82
- equal-fix-int-to-ilessp-off, 103
- equal-fix-int-to-ilessp-off1, 1155
- equal-fix-int-to-ilessp-on, 1067
- equal-gcd-0, 33
- equal-gcd-0-off, 117
- equal-gcd-0-off1, 1169
- equal-gcd-0-on, 1081
- equal-idiv-2-0, 1214
- equal-iff, 1099
- equal-iff-off, 1099
- equal-iff-on, 1183
- equal-ineg-ineg, 67
- equal-ineg-ineg-off, 105
- equal-ineg-ineg-off1, 1157
- equal-ineg-ineg-on, 1069
- equal-itimes-0, 50
- equal-itimes-0-off, 110
- equal-itimes-0-off1, 1162
- equal-itimes-0-on, 1074
- equal-itimes-1, 50
- equal-itimes-1-off, 110
- equal-itimes-1-off1, 1162
- equal-itimes-1-on, 1073
- equal-itimes-list-eval\$-list-b
 - agdiff, 78
 - agdiff-off, 85
- equal-itimes-list-eval\$-list-de
 - lete, 57
 - lete-new-1, 78
 - lete-new-1-off, 85
 - lete-new-2, 78
 - lete-new-2-off, 85
 - lete-off, 64
- equal-itimes-minus-1, 50
- equal-itimes-minus-1-off, 110
- equal-itimes-minus-1-off1, 1162
- equal-itimes-minus-1-on, 1073
- equal-length-32-as-collected-nt
 - h, 148
- equal-length-4-as-collected-nth, 148
- equal-length-40-as-collected-nt
 - h+subrange, 776
- equal-length-add1, 137
- equal-length-add1-off, 137
- equal-length-cdr, 137
- equal-length-read-mem, 954
- equal-length-read-mem-off, 994

- equal-length-read-mem-off1, 1149
- equal-length-read-mem-on, 1060
- equal-length-read-mem1, 954
- equal-length-read-mem1-off, 994
- equal-length-read-mem1-off1, 1149
- equal-length-read-mem1-on, 1060
- equal-log-0, 31
- equal-log-0-off, 119
- equal-log-0-off1, 1170
- equal-log-0-on, 1082
- equal-memory-value, 184
- equal-memory-value-for-chip-syste
m\$state, 800
- equal-nat-to-v-inverter, 1098
- equal-nat-to-v-inverter-hack1, 1098
- equal-nat-to-v-inverter-hack1-o
ff, 1098
- n, 1183
- equal-nat-to-v-inverter-hack2, 1098
- equal-nat-to-v-inverter-hack2-le
mma, 1098
- mma-off, 1114
- equal-nat-to-v-inverter-hack2-o
ff, 1098
- n, 1183
- equal-nat-to-v-inverter-off, 1114
- equal-occurrences-zero, 3
- equal-occurrences-zero-off, 131
- equal-occurrences-zero-off1, 1182
- equal-occurrences-zero-on, 1094
- equal-plus-0, 5
- equal-plus-0-off, 130
- equal-plus-0-off1, 1181
- equal-plus-0-on, 1093
- equal-quotient-0, 25
- equal-quotient-0-off, 122
- equal-quotient-0-off1, 1174
- equal-quotient-0-on, 1085
- equal-remainder-difference-0, 23
- equal-remainder-difference-0-of
f, 23
- equal-remainder-plus-0, 22
- equal-remainder-plus-0-off, 124
- equal-remainder-plus-0-off1, 1176
- equal-remainder-plus-0-on, 1087
- equal-remainder-plus-0-proof, 22
- equal-remainder-plus-0-proof-of
f, 124
- f1, 1176
- equal-remainder-plus-0-proof-on, 1087
- equal-remainder-plus-remainder, 22
- equal-remainder-plus-remainder-
off, 22
- equal-remainder-plus-remainder-p
roof, 22
- roof-off, 124
- roof-off1, 1175
- roof-on, 1087
- equal-sub1-0, 13
- equal-sub1-0-off, 127
- equal-sub1-0-off1, 1179
- equal-sub1-0-on, 1090
- equal-sub1-plus-x-x, 1214
- equal-times-0, 13
- equal-times-0-off, 127
- equal-times-0-off1, 1179
- equal-times-0-on, 1091
- equal-times-1, 13
- equal-times-1-off, 127
- equal-times-1-off1, 1179
- equal-times-1-on, 1090
- equal-times-arg1, 17
- equal-times-arg1-off, 17
- equal-times-bridge, 17
- equal-times-bridge-off, 17
- equality-of-three-element-lists, 448
- err-and, 251, 277–284, 286, 289, 290,
292, 294, 305, 311–313, 317,
334, 338, 358, 360–362, 364–
371, 373, 377
- err-and-off, 251
- error-entry, 250, 340
- error-entry-off, 251
- error-label, 250, 335
- eval\$-append, 954
- eval\$-append-2, 954
- eval\$-append-2-off, 995
- eval\$-append-2-off1, 1149

- eval\$-append-2-on, 1060
- eval\$-append-off, 995
- eval\$-append-off1, 1149
- eval\$-append-on, 1060
- eval\$-cancel-ineg-aux-fn, 41, 42
- eval\$-cancel-ineg-aux-fn-off, 113
- eval\$-cancel-ineg-aux-fn-off1, 1165
- eval\$-cancel-ineg-aux-fn-on, 1076
- eval\$-cancel-ineg-aux-is-its-f
 - n, 42
 - n-off, 113
 - n-off1, 1165
 - n-on, 1076
- eval\$-cancel-iplus, 45
- eval\$-cancel-iplus-off, 63
- eval\$-disjoin-equalities-with-
 - 0-off, 103
 - 0-off1, 1155
 - 0-on, 1066
 - 0, 83
- eval\$-equal, 16
- eval\$-equal-itimes-tree-itimes
 - fringe-0, 58
 - fringe-0-off, 65
- eval\$-equal-off, 16
- eval\$-equal-off1, 125
- eval\$-equal-off2, 1177
- eval\$-equal-on, 58
- eval\$-equal-on1, 1088
- eval\$-equal-times-tree-bagdiff, 20
 - off, 20
- eval\$-if, 17
- eval\$-if-off, 17
- eval\$-ilessp-iplus-tree-no-fix
 - int, 48
 - int-off, 64
- eval\$-iplus, 47
- eval\$-iplus-list-bagdiff, 46
- eval\$-iplus-list-bagdiff-off, 63
- eval\$-iplus-list-car-remove-ine
 - gs, 92
 - gs-off, 101
 - gs-off1, 1153
 - gs-on, 1064
- eval\$-iplus-list-cdr-remove-ine
 - gs, 92
 - gs-off, 101
 - gs-off1, 1153
 - gs-on, 1064
- eval\$-iplus-list-delete, 46
- eval\$-iplus-list-delete-off, 112
- eval\$-iplus-list-delete-off1, 1164
- eval\$-iplus-list-delete-on, 1075
- eval\$-iplus-off, 63
- eval\$-iplus-tree, 43
- eval\$-iplus-tree-off, 63
- eval\$-iplus-tree-rec, 43
- eval\$-iplus-tree-rec-off, 63
- eval\$-itimes-tree, 55
- eval\$-itimes-tree-ineg, 67
- eval\$-itimes-tree-ineg-off, 105
- eval\$-itimes-tree-ineg-off1, 1157
- eval\$-itimes-tree-ineg-on, 1068
- eval\$-itimes-tree-no-fix-int-1, 60
 - off, 65
- eval\$-itimes-tree-no-fix-int-2, 60
 - off, 65
- eval\$-itimes-tree-off, 64
- eval\$-itimes-tree-rec, 55
- eval\$-itimes-tree-rec-off, 64
- eval\$-lessp, 16
- eval\$-lessp-off, 16
- eval\$-lessp-times-tree-bagdiff, 19
 - off, 19
- eval\$-list-append, 43
- eval\$-list-append-off, 112
- eval\$-list-append-off1, 1164
- eval\$-list-append-on, 1075
- eval\$-list-bagint-0, 79
- eval\$-list-bagint-0-for-ilessp, 82
 - off, 103
 - off1, 1155
 - on, 1067
- eval\$-list-bagint-0-implies-eq
 - ual, 79
 - ual-for-ilessp, 82
 - ual-for-ilessp-lemma, 82
 - ual-for-ilessp-lemma-off, 103

- ual-for-ilessp-lemma-off1, 1155
- ual-for-ilessp-lemma-on, 1067
- ual-for-ilessp-off, 87
- ual-off, 86
- eval\$-list-bagint-0-off, 86
- eval\$-list-cons, 41
- eval\$-list-cons-off, 113
- eval\$-list-cons-off1, 1165
- eval\$-list-cons-on, 1076
- eval\$-list-nlistp, 41
- eval\$-list-nlistp-off, 113
- eval\$-list-nlistp-off1, 1165
- eval\$-list-nlistp-on, 1076
- eval\$-litatom, 41
- eval\$-litatom-off, 113
- eval\$-litatom-off1, 1165
- eval\$-litatom-on, 1076
- eval\$-make-cancel-itimes-equalit
 - y, 58
 - y-1, 58
 - y-1-off, 65
 - y-2, 58
 - y-2-off, 65
 - y-off, 64
- eval\$-make-cancel-itimes-inequ
 - ality, 61
 - ality-off, 65
- eval\$-off, 41
- eval\$-on, 97
- eval\$-or, 16
- eval\$-or-off, 16
- eval\$-other, 41
- eval\$-other-off, 113
- eval\$-other-off1, 1165
- eval\$-other-on, 1076
- eval\$-pairlist-cons, 954
- eval\$-pairlist-cons-off, 995
- eval\$-pairlist-cons-off1, 1149
- eval\$-pairlist-cons-on, 1060
- eval\$-plus-tree-append, 9
- eval\$-plus-tree-append-off, 9
- eval\$-quote, 9
- eval\$-quote-off, 9
- eval\$-quote-off1, 129
- eval\$-quote-off2, 1180
- eval\$-quote-on, 41
- eval\$-quote-on1, 1092
- eval\$-quotient, 16
- eval\$-quotient-off, 16
- eval\$-quotient-times-tree-bagdi
 - ff, 29
 - ff-off, 29
- eval\$-times, 16
- eval\$-times-member, 17
- eval\$-times-member-off, 17
- eval\$-times-off, 16
- execute-fm9001, 559
- exp, 29–32, 1096–1102, 1192, 1193,
 - 1196–1198, 1200, 1218
- exp-0-arg1, 31
- exp-0-arg1-off, 119
- exp-0-arg1-off1, 1171
- exp-0-arg1-on, 1082
- exp-0-arg2, 31
- exp-0-arg2-off, 119
- exp-0-arg2-off1, 1171
- exp-0-arg2-on, 1082
- exp-1-arg1, 31
- exp-1-arg1-off, 119
- exp-1-arg1-off1, 1171
- exp-1-arg1-on, 1082
- exp-2-l-1 <= exp-2-l, 1198
- exp-add1, 30
- exp-add1-off, 119
- exp-add1-off1, 1171
- exp-add1-on, 1082
- exp-difference, 31
- exp-difference-off, 119
- exp-difference-off1, 1170
- exp-difference-on, 1082
- exp-exp, 31
- exp-exp-off, 119
- exp-exp-off1, 1171
- exp-exp-on, 1082
- exp-linear-bounds, 1198
- exp-off, 120
- exp-off1, 1172
- exp-on, 1083

exp-plus, 30
 exp-plus-off, 119
 exp-plus-off1, 1171
 exp-plus-on, 1082
 exp-times, 31
 exp-times-off, 119
 exp-times-off1, 1171
 exp-times-on, 1082
 exp-zero, 30
 exp-zero-off, 119
 exp-zero-off1, 1171
 exp-zero-on, 1083
 exp2, 550, 558
 expand-*-connectives, 132
 expand-*-connectives-off, 133
 expand-f-functions, 194
 expand-f-functions-off, 196
 expand-top-level-dual-eval-0-calls, 240
 expand-top-level-dual-eval-2-calls, 241
 exponentiation, 31
 extend-immediate*, 746–748
 extend-immediate*-off, 747
 extend-immediate*\$destructure, 747
 extend-immediate*\$destructure-off, 747
 extend-immediate\$netlist, 748, 776
 extend-immediate\$value, 749
 extend-immediate\$value-off, 749
 extend-immediate&, 748, 749, 776
 extend-immediate&-off, 748
 external-loading, 271
 external-loading-off, 271
 externalize-parents, 298, 317, 334, 339
 externalize-parents-off, 298
 extract-names, 298, 311, 315, 316
 extract-names-off, 298
 extract-names-simple, 269, 270
 extract-names-simple-off, 270
 f-and, 192–199, 201, 391, 392, 437, 438, 444, 488, 509, 517, 523, 566, 567, 670, 676, 678, 933, 938, 939
 f-and-monotone, 933
 f-and-monotone-off, 1055
 f-and-monotone-on, 1144
 f-and-off, 936
 f-and-off1, 1147
 f-and-on, 1057
 f-and-rewrite, 196
 f-and3, 192, 194, 197, 199, 201, 392, 500, 676–678, 739, 933, 940
 f-and3-monotone, 933
 f-and3-monotone-off, 1055
 f-and3-monotone-on, 1144
 f-and3-off, 936
 f-and3-off1, 1147
 f-and3-on, 1057
 f-and4, 192, 194, 197, 199, 201, 392, 592, 676, 933, 940
 f-and4-monotone, 933
 f-and4-monotone-off, 1055
 f-and4-monotone-on, 1144
 f-and4-off, 936
 f-and4-off1, 1147
 f-and4-on, 1057
 f-buf, 192, 194, 196–198, 200, 395, 400, 403, 404, 409–411, 415, 419, 456, 739, 740, 761–764, 782, 784, 793, 794, 796, 802–805, 811–815, 817, 819, 888, 890, 891, 933, 938, 943, 946–951, 1004
 f-buf-delete-lemmas, 197
 f-buf-delete-lemmas-off, 198
 f-buf-lemma, 196
 f-buf-monotone, 933
 f-buf-monotone-off, 1055
 f-buf-monotone-on, 1144
 f-buf-off, 936
 f-buf-off1, 1147
 f-buf-on, 1057
 f-buf-preserves-good-s, 1004
 f-buf-preserves-good-s-off, 1039
 f-buf-preserves-good-s-on, 1128

f-buf-type-set, 938
 f-buf-type-set-off, 1053
 f-buf-type-set-on, 1143
 f-equiv, 194, 195, 197–199, 201, 393,
 463, 464, 467, 469, 488, 936,
 940
 f-equiv-monotone, 936
 f-equiv-monotone-off, 1054
 f-equiv-monotone-on, 1143
 f-equiv-off, 937
 f-equiv-off1, 1148
 f-equiv-on, 1058
 f-equiv3, 194, 195, 197, 199, 201, 393,
 501, 936, 940
 f-equiv3-monotone, 936
 f-equiv3-monotone-off, 1054
 f-equiv3-monotone-on, 1143
 f-equiv3-off, 937
 f-equiv3-off1, 1148
 f-equiv3-on, 1059
 f-gate-threefix-congruence-lemm
 as, 198
 as-off, 200
 as\$help, 198
 as\$help-off, 198
 f-gates, 200
 f-gates=b-gates, 200
 f-if, 194–196, 198, 200, 202, 203, 393,
 400, 415, 439, 440, 521, 566,
 744, 763, 782, 794, 804, 813,
 936, 938, 941, 947, 948, 1004
 f-if-monotone, 936
 f-if-monotone-off, 1054
 f-if-monotone-on, 1143
 f-if-off, 937
 f-if-off1, 1148
 f-if-on, 1059
 f-if-preserves-good-s, 1004
 f-if-preserves-good-s-off, 1039
 f-if-preserves-good-s-on, 1128
 f-if-rewrite, 196
 f-nand, 192, 195, 197, 198, 200, 391,
 393, 408–410, 456, 488, 491,
 496, 498, 500, 501, 517, 520,
 521, 523, 560, 563, 566, 571–
 573, 676–678, 739, 740, 761,
 762, 781, 782, 793, 802, 803,
 811, 812, 933, 939, 941, 948–
 950
 f-nand-monotone, 933
 f-nand-monotone-off, 1055
 f-nand-monotone-on, 1144
 f-nand-off, 936
 f-nand-off1, 1147
 f-nand-on, 1058
 f-nand3, 192, 195, 197, 198, 201, 394,
 488, 491, 496–498, 517, 520,
 521, 563, 571–573, 677, 678,
 934, 941
 f-nand3-monotone, 934
 f-nand3-monotone-off, 1055
 f-nand3-monotone-on, 1144
 f-nand3-off, 936
 f-nand3-off1, 1147
 f-nand3-on, 1058
 f-nand4, 192, 195, 197, 198, 201, 394,
 488, 491, 517, 563, 580, 581,
 676–678, 934, 941
 f-nand4-monotone, 934
 f-nand4-monotone-off, 1055
 f-nand4-monotone-on, 1144
 f-nand4-off, 936
 f-nand4-off1, 1147
 f-nand4-on, 1058
 f-nand5, 193, 195, 197–199, 201, 394,
 677, 934, 942
 f-nand5-monotone, 934
 f-nand5-monotone-off, 1055
 f-nand5-monotone-on, 1144
 f-nand5-off, 937
 f-nand5-off1, 1147
 f-nand5-on, 1058
 f-nand6, 193, 195, 197, 199, 201, 395,
 677, 934, 942
 f-nand6-monotone, 934
 f-nand6-monotone-off, 1055
 f-nand6-monotone-on, 1144
 f-nand6-off, 937

f-nand6-off1, 1147
 f-nand6-on, 1058
 f-nand8, 193, 195, 197, 199, 201, 395, 934, 942
 f-nand8-monotone, 934
 f-nand8-monotone-off, 1055
 f-nand8-monotone-on, 1144
 f-nand8-off, 937
 f-nand8-off1, 1147
 f-nand8-on, 1058
 f-nor, 193–195, 197, 199–201, 391, 396, 456, 463, 464, 486, 488, 491, 520, 521, 565, 571–573, 580, 581, 935, 938, 939, 943
 f-nor-monotone, 935
 f-nor-monotone-off, 1054
 f-nor-monotone-on, 1144
 f-nor-off, 937
 f-nor-off1, 1147
 f-nor-on, 1058
 f-nor3, 193, 195, 198, 200, 201, 396, 459, 463, 464, 486, 581, 935, 943
 f-nor3-monotone, 935
 f-nor3-monotone-off, 1054
 f-nor3-monotone-on, 1144
 f-nor3-off, 937
 f-nor3-off1, 1147
 f-nor3-on, 1058
 f-nor4, 193, 195, 198, 200, 201, 396, 580, 581, 935, 943
 f-nor4-monotone, 935
 f-nor4-monotone-off, 1054
 f-nor4-monotone-on, 1144
 f-nor4-off, 937
 f-nor4-off1, 1148
 f-nor4-on, 1058
 f-nor5, 193, 195, 198, 200, 202, 396, 935, 944
 f-nor5-monotone, 935
 f-nor5-monotone-off, 1054
 f-nor5-monotone-on, 1144
 f-nor5-off, 937
 f-nor5-off1, 1148
 f-nor5-on, 1058
 f-nor6, 193, 195, 198, 200, 202, 397, 935, 944
 f-nor6-monotone, 935
 f-nor6-monotone-off, 1054
 f-nor6-monotone-on, 1143
 f-nor6-off, 937
 f-nor6-off1, 1148
 f-nor6-on, 1058
 f-nor8, 193, 195, 198, 200, 202, 397, 935, 944
 f-nor8-monotone, 935
 f-nor8-monotone-off, 1054
 f-nor8-monotone-on, 1143
 f-nor8-off, 937
 f-nor8-off1, 1148
 f-nor8-on, 1058
 f-not, 192–198, 200, 395, 397, 398, 403, 404, 409, 411, 456, 458, 463, 464, 475, 488, 491, 496, 497, 501, 517, 520, 521, 523, 559, 560, 563, 565–567, 571–573, 581, 670, 676–678, 717, 755, 756, 759, 761–764, 781, 782, 784, 793, 794, 796, 802, 803, 805, 811–814, 933, 943, 945, 947–949, 951
 f-not-f-not=f-buf, 197
 f-not-f-not=f-buf-off, 197
 f-not-monotone, 933
 f-not-monotone-off, 1055
 f-not-monotone-on, 1144
 f-not-off, 936
 f-not-off1, 1147
 f-not-on, 1057
 f-not-rewrite, 196
 f-or, 193, 195–199, 201, 202, 391, 398, 438, 444, 456, 468, 472, 497, 523, 562, 670, 676–678, 763, 782, 794, 804, 813, 934, 939, 945
 f-or-monotone, 934
 f-or-monotone-off, 1055
 f-or-monotone-on, 1144

f-or-off, 937
 f-or-off1, 1147
 f-or-on, 1058
 f-or-rewrite, 196
 f-or3, 193, 195, 197, 199, 201, 398,
 520, 521, 523, 934, 945
 f-or3-monotone, 934
 f-or3-monotone-off, 1054
 f-or3-monotone-on, 1144
 f-or3-off, 937
 f-or3-off1, 1147
 f-or3-on, 1058
 f-or4, 193, 195, 197, 199, 201, 399,
 591, 934, 946
 f-or4-monotone, 934
 f-or4-monotone-off, 1054
 f-or4-monotone-on, 1144
 f-or4-off, 937
 f-or4-off1, 1147
 f-or4-on, 1058
 f-pullup, 194, 196, 197, 204, 409,
 761–764, 802–805, 812–814,
 936, 949
 f-pullup-monotone, 936
 f-pullup-monotone-off, 1054
 f-pullup-monotone-on, 1143
 f-pullup-off, 938
 f-pullup-off1, 1148
 f-pullup-on, 1059
 f-pullup-rewrite, 197
 f-xor, 193–195, 197–199, 201, 202,
 399, 437, 438, 463, 464, 491,
 520, 521, 562, 935, 946
 f-xor-monotone, 935
 f-xor-monotone-off, 1054
 f-xor-monotone-on, 1143
 f-xor-off, 937
 f-xor-off1, 1148
 f-xor-on, 1058
 f-xor3, 193, 195, 197, 199, 201, 399,
 936, 946
 f-xor3-monotone, 936
 f-xor3-monotone-off, 1054
 f-xor3-monotone-on, 1143
 f-xor3-off, 937
 f-xor3-off1, 1148
 f-xor3-on, 1058
 f\$all-t-regs-address, 592, 670
 f\$all-t-regs-address-off, 592
 f\$all-t-regs-address=set-some=
 flags, 592
 f\$alu-cell, 501–503, 508
 f\$alu-cell-off, 501
 f\$alu-cell-v-threefix-mpg, 503
 f\$alu-cell=alu-cell, 501
 f\$ao2, 194, 559, 560
 f\$b-store-resultp, 561–563, 670
 f\$b-store-resultp-off, 562
 f\$b-store-resultp=b-store-resu
 ltp, 562
 f\$carry-in-help, 496–498
 f\$carry-in-help-off, 497
 f\$carry-in-help=carry-in-help, 497
 f\$carry-out-help, 517–519, 529
 f\$carry-out-help-off, 517
 f\$carry-out-help=carry-out-help, 519
 f\$control-let, 670, 673, 717, 718
 f\$control-let-off, 670
 f\$control-let=control-let, 673
 f\$score-alu, 529, 531, 541, 761, 763,
 781, 782, 784, 792–794, 796,
 801, 802, 804, 811, 813
 f\$score-alu-off, 530
 f\$score-alu=core-alu, 531
 f\$cv, 702, 704, 715, 718
 f\$cv-off, 701
 f\$cv=cv, 701
 f\$dec-pass, 475, 763, 782, 794, 804,
 813
 f\$dec-pass-off, 475
 f\$dec-pass=dec-or-pass, 475
 f\$decode-5, 571, 578, 717
 f\$decode-5-off, 573
 f\$decode-5=decode-5, 578
 f\$decode-gen, 491, 493
 f\$decode-gen-off, 491
 f\$decode-gen=decode-gen, 493
 f\$decode-mode, 486, 487, 494

f\$decode-mode-off, 486
 f\$decode-mode=decode-mode, 487
 f\$decode-prop, 488, 490, 493
 f\$decode-prop-off, 488
 f\$decode-prop=decode-prop, 490
 f\$decode-reg-mode, 565, 670
 f\$decode-reg-mode-as-reg-mode, 565
 f\$decode-reg-mode-off, 565
 f\$encode-32, 580, 584, 585, 588, 589, 591
 f\$encode-32-off, 581
 f\$encode-32=encode-32, 584
 f\$extend-immediate, 748, 749, 761–763, 781, 782, 793, 794, 802–804, 811–813
 f\$extend-immediate-off, 748
 f\$extend-immediate=extend-immediate, 748
 f\$fast-zero, 459, 461, 530
 f\$fast-zero-off, 459
 f\$fast-zero=tr-or-nor, 459
 f\$fast-zero=v-zerop, 459
 f\$full-adder, 438
 f\$full-adder=full-adder, 438
 f\$g-cell, 500, 501
 f\$g-cell-off, 500
 f\$g-cell=g-cell, 500
 f\$mpg, 493–495
 f\$mpg-off, 494
 f\$mpg=mpg, 494
 f\$next-cntl-state, 717–719, 722, 764, 783, 794, 804, 814
 f\$next-cntl-state-off, 718
 f\$next-cntl-state=next-cntl-state, 719
 f\$next-state, 676, 679, 689, 690, 718
 f\$next-state-off, 678
 f\$next-state=next-state, 678
 f\$overflow-help, 520, 522, 523, 530
 f\$overflow-help-off, 521
 f\$overflow-help=overflow-help, 522
 f\$p-cell, 498, 499, 501
 f\$p-cell-off, 498
 f\$p-cell=p-cell, 499
 f\$read-regs, 739, 740, 761–763, 781, 782, 793, 794, 802–804, 811–813
 f\$read-regs-off, 739
 f\$read-regs=read-regs, 739
 f\$select-op-code, 566, 567
 f\$select-op-code-off, 566
 f\$select-op-code-selects, 566
 f\$set-some-flags, 591, 592, 670
 f\$set-some-flags-off, 591
 f\$set-some-flags=set-some-flags, 591
 f\$shift-or-buf, 525, 526, 529, 530
 f\$shift-or-buf-cntl, 523, 525
 f\$shift-or-buf-cntl-off, 524
 f\$shift-or-buf-cntl=shift-or-buf-cntl, 525
 f\$shift-or-buf-off, 525
 f\$shift-or-buf=shift-or-buf, 526
 f\$store-resultp-mux, 559, 561, 562
 f\$store-resultp-mux-off, 560
 f\$store-resultp-mux=store-resultp-mux, 561
 f\$t-carry, 444, 445, 508–510, 515, 529
 f\$t-carry-off, 444
 f\$t-carry=t-carry, 444
 f\$tv-alu-help, 508–511, 515–517, 529, 530
 f\$tv-alu-help-length, 510
 f\$tv-alu-help-off, 510
 f\$tv-alu-help-v-threefix-mpg, 516
 f\$tv-alu-help=tv-alu-help, 511
 f\$tv-dec-pass-ng, 467–469, 472, 473, 475
 f\$tv-dec-pass-ng-length, 468
 f\$tv-dec-pass-ng-length-1, 468
 f\$tv-dec-pass-ng-off, 468
 f\$tv-dec-pass-ng=tv-dec-pass-ng, 469
 g\$super-crock, 469
 g\$super-crock-off, 469
 f\$tv-zerop, 458, 459, 462
 f\$tv-zerop-off, 458
 f\$tv-zerop=v-zerop, 459

f\$unary-op-code-p, 563, 564, 670
 f\$unary-op-code-p-off, 563
 f\$unary-op-code-p=unary-op-code
 -p, 564
 f\$update-flags, 744, 745, 761, 781,
 793, 802, 811
 f\$update-flags-off, 744
 f\$update-flags=update-flags, 745
 f\$v-equal, 462, 739
 f\$v-equal-off, 462
 f\$v-equal=equal*, 462
 f\$v-if-f-4, 567, 568
 f\$v-if-f-4-off, 567
 f\$v-if-f-4=fv-if, 568
 f\$v-inc4, 463, 464
 f\$v-inc4-off, 463
 f\$v-inc4=v-inc4, 464
 f\$v-inc4\$v, 464
 f\$v-inc4\$v-off, 464
 f\$v-inc4\$v=v-inc, 464
 f\$write-regs, 740, 741, 761, 781, 792,
 801, 811
 f\$write-regs-off, 740
 f\$write-regs=write-regs, 741
 fanout-32*, 569, 570
 fanout-32\$netlist, 570
 fanout-32\$value, 570
 fanout-32\$value-off, 571
 fanout-32&, 570
 fanout-32&-off, 570
 fanout-4*, 568
 fanout-4\$netlist, 568, 714
 fanout-4\$value, 568
 fanout-4\$value-off, 568
 fanout-4&, 568, 714
 fanout-4&-off, 568
 fanout-5*, 569
 fanout-5\$netlist, 569, 721
 fanout-5\$value, 569
 fanout-5\$value-off, 569
 fanout-5&, 569, 721
 fanout-5&-off, 569
 fast-zero*, 459–461
 fast-zero*-off, 460
 fast-zero*\$destructure, 460
 fast-zero*\$destructure-off, 460
 fast-zero\$netlist, 460, 461, 541
 fast-zero\$value, 461
 fast-zero&, 460, 461, 540
 fast-zero&-off, 460
 fd1-monotone, 947
 fd1-monotone-off, 1046
 fd1-monotone-on, 1135
 fd1-preserves-good-s, 1006
 fd1-preserves-good-s-off, 1036
 fd1-preserves-good-s-on, 1125
 fd1\$netlist, 403, 438
 fd1\$state, 414
 fd1\$state-off, 415
 fd1\$value, 403
 fd1\$value-off, 403
 fd1&, 403, 414, 439
 fd1&-off, 403
 fd1s-monotone, 948
 fd1s-monotone-off, 1046
 fd1s-monotone-on, 1135
 fd1s-preserves-good-s, 1006
 fd1s-preserves-good-s-off, 1036
 fd1s-preserves-good-s-on, 1125
 fd1s\$netlist, 403, 477, 739, 776
 fd1s\$state, 415
 fd1s\$state-off, 415
 fd1s\$value, 403
 fd1s\$value-off, 403
 fd1s&, 403, 415, 476–478, 738, 775
 fd1s&-off, 403
 fd1slp-monotone, 948
 fd1slp-monotone-off, 1045
 fd1slp-monotone-on, 1135
 fd1slp-preserves-good-s, 1006
 fd1slp-preserves-good-s-off, 1035
 fd1slp-preserves-good-s-on, 1125
 fd1slp\$netlist, 404, 480, 744, 776
 fd1slp\$state, 415
 fd1slp\$state-off, 415
 fd1slp\$value, 404
 fd1slp\$value-off, 404

fd1slp&, 404, 415, 480, 481, 744, 775
 fd1slp&-off, 404
 fd1sp-monotone, 948
 fd1sp-monotone-off, 1045
 fd1sp-monotone-on, 1135
 fd1sp-preserves-good-s, 1006
 fd1sp-preserves-good-s-off, 1036
 fd1sp-preserves-good-s-on, 1125
 fd1sp\$netlist, 404
 fd1sp\$state, 415
 fd1sp\$state-off, 415
 fd1sp\$value, 404
 fd1sp\$value-off, 404
 fd1sp&, 403, 404, 415
 fd1sp&-off, 404
 fetch0->fetch0\$sim0, 887
 fetch0->fetch0\$sim1, 889
 fetch0->fetch0\$sim2, 890
 fetch0\$step0, 814
 fetch0\$step1, 816
 fetch0\$step2, 817
 fetch1->decode\$sim, 891
 fetch1\$step, 819
 fetch2\$step, 821
 fetch3\$induction, 874, 875
 fetch3\$progress, 876
 fetch3\$progress-help, 875
 fetch3\$step0, 823
 fetch3\$step1, 825
 final-state, 1013, 1014, 1016
 final-state-off, 1031
 final-state-okp, 1014
 final-state-okp-off, 1031
 final-state-okp-on, 1120
 final-state-on, 1120
 firstn, 139–141, 145, 146, 151, 159–162, 167, 168, 452, 459, 460, 473, 516, 557, 1097, 1098
 firstn-add1-cons, 1098
 firstn-add1-cons-off, 1114
 firstn-append, 140
 firstn-bottom, 140
 firstn-length, 1098
 firstn-length-off, 1114
 firstn-make-list, 145
 firstn-nat-to-v, 160
 firstn-nlistp, 1098
 firstn-nlistp-off, 1114
 firstn-off, 139
 firstn-v-not, 159
 firstn-v-not-off, 159
 firstn-zerop, 1098
 firstn-zerop-off, 1114
 fix-breadth-tree-stack, 191
 fix-dependent-drs, 388
 fix-dependent-drs-off, 388
 fix-dependent-lds, 268, 273, 274
 fix-dependent-lds-off, 268
 fix-dependent-types, 259–261, 263
 fix-dependent-types-off, 259
 fix-drives, 270, 274
 fix-drives-off, 270
 fix-int, 35–39, 41, 42, 46, 47, 49, 50, 52–54, 57–59, 62, 67, 78–80, 82, 83, 95, 1102, 1219
 fix-int-eval\$-itimes-tree-rec, 67
 fix-int-eval\$-itimes-tree-rec-off, 105
 off1, 1157
 on, 1069
 fix-int-fix-int, 38
 fix-int-fix-int-off, 115
 fix-int-fix-int-off1, 1166
 fix-int-fix-int-on, 1078
 fix-int-iabs, 38
 fix-int-iabs-off, 115
 fix-int-iabs-off1, 1166
 fix-int-iabs-on, 1078
 fix-int-idifference, 38
 fix-int-idifference-off, 115
 fix-int-idifference-off1, 1166
 fix-int-idifference-on, 1078
 fix-int-idiv, 54
 fix-int-idiv-off, 107
 fix-int-idiv-off1, 1159
 fix-int-idiv-on, 1071
 fix-int-imod, 54

fix-int-imod-off, 107
 fix-int-imod-off1, 1159
 fix-int-imod-on, 1071
 fix-int-ineg, 38
 fix-int-ineg-off, 115
 fix-int-ineg-off1, 1166
 fix-int-ineg-on, 1078
 fix-int-int-to-v, 1219
 fix-int-iplus, 38
 fix-int-iplus-off, 115
 fix-int-iplus-off1, 1166
 fix-int-iplus-on, 1078
 fix-int-iquo, 54
 fix-int-iquo-off, 107
 fix-int-iquo-off1, 1159
 fix-int-iquo-on, 1070
 fix-int-iquotient, 54
 fix-int-iquotient-off, 107
 fix-int-iquotient-off1, 1159
 fix-int-iquotient-on, 1071
 fix-int-irem, 54
 fix-int-irem-off, 107
 fix-int-irem-off1, 1159
 fix-int-irem-on, 1070
 fix-int-iremainder, 54
 fix-int-iremainder-off, 107
 fix-int-iremainder-off1, 1159
 fix-int-iremainder-on, 1071
 fix-int-itimes, 38
 fix-int-itimes-off, 114
 fix-int-itimes-off1, 1166
 fix-int-itimes-on, 1078
 fix-int-numberp, 1102
 fix-int-numberp-off, 1111
 fix-int-off, 37
 fix-int-remover, 38
 fix-int-remover-off, 115
 fix-int-remover-off1, 1167
 fix-int-remover-on, 1078
 fix-loadings, 270, 274
 fix-loadings-off, 270
 flags, 545, 547, 559, 761, 764, 921,
 922, 925
 flags*, 742–744
 flags*-off, 744
 flags*\$destructure, 743
 flags*\$destructure-off, 744
 flags-hyps, 591, 592
 flags-interpretation-int-add-gt
 -0, 1207
 flags-interpretation-int-add-ne
 gative, 1207
 flags-interpretation-int-add-ove
 rflow, 1207
 flags-interpretation-int-addc-gt
 -0, 1206
 flags-interpretation-int-addc-ne
 gative, 1206
 flags-interpretation-int-addc-ove
 rflow, 1205
 flags-interpretation-int-asr-ca
 rry, 1214
 flags-interpretation-int-asr-gt
 -0, 1215
 flags-interpretation-int-asr-ne
 gative, 1215
 flags-interpretation-int-asr-ove
 rflow, 1216
 flags-interpretation-int-asr-ze
 ro, 1214
 flags-interpretation-int-dec-gt
 -0, 1211
 flags-interpretation-int-dec-ne
 gative, 1210
 flags-interpretation-int-dec-ove
 rflow, 1210
 flags-interpretation-int-dec-ze
 ro, 1210
 flags-interpretation-int-inc-gt
 -0, 1205
 flags-interpretation-int-inc-ne
 gative, 1205
 flags-interpretation-int-inc-ove
 rflow, 1204
 flags-interpretation-int-inc-ze
 ro, 1204
 flags-interpretation-int-move, 1200
 flags-interpretation-int-neg-gt

- 0, 1209
- flags-interpretation-int-neg-negative, 1208
- flags-interpretation-int-neg-overflow, 1208
- flags-interpretation-int-neg-zero, 1209
- flags-interpretation-int-sub-gt-0, 1213
- flags-interpretation-int-sub-negative, 1213
- flags-interpretation-int-sub-overflow, 1212
- flags-interpretation-int-subb-gt-0, 1212
- flags-interpretation-int-subb-negative, 1211
- flags-interpretation-int-subb-overflow, 1211
- flags-interpretation-lsr, 1190
- flags-interpretation-nat-add, 1192
- flags-interpretation-nat-addc, 1192
- flags-interpretation-nat-dec, 1193
- flags-interpretation-nat-inc, 1192
- flags-interpretation-nat-lsr-carry, 1196
- flags-interpretation-nat-lsr-zero, 1196
- flags-interpretation-nat-move, 1191
- flags-interpretation-nat-sub, 1195
- flags-interpretation-nat-subb-carry, 1194
- flags-interpretation-nat-subb-higher, 1195
- flags-interpretation-ror, 1190
- flags-interpretation-sub, 1189
- flags-interpretation-xor, 1191
- flags-off, 545
- flags-theory, 543
- flags\$netlist, 744, 776
- flags\$partial-state, 746
- flags\$partial-state-help, 746
- flags\$partial-state-help-off, 746
- flags\$partial-state-off, 746
- flags\$state, 745
- flags\$state-help, 745
- flags\$state-help-off, 745
- flags\$state-off, 745
- flags\$value, 744
- flags\$value-off, 744
- flags&, 744–746, 775
- flags&-off, 744
- flatten-list, 246, 253, 256, 436
- flatten-list-off, 246
- fm9001, 547, 559, 925, 927, 928, 1021, 1023
- fm9001-alu-operation, 545, 546
- fm9001-external-input-accessors, 761
- fm9001-fetch, 546, 547
- fm9001-hardware-state-accessors, 760
- fm9001-interpreter, 547, 924–926, 1014, 1026
- fm9001-interpreter-correct, 924
- fm9001-interpreter-correct\$map-down, 926
- fm9001-interpreter-off, 547
- fm9001-intr, 547
- fm9001-intr-off, 547
- fm9001-machine-statep, 926, 1017
- fm9001-machine-statep-p-map-up-initialized-machine-state, 1017
- fm9001-machine-state-off, 1029
- fm9001-machine-state-on, 1118
- fm9001-next-state, 761, 764, 765, 806, 807, 809, 811
- fm9001-next-state-off, 764
- fm9001-off, 547
- fm9001-operand-a, 546, 547
- fm9001-operand-b, 545, 546
- fm9001-preserves-chip-system-invariant, 807
- fm9001-state-as-a-list, 765
- fm9001-state-structure, 765, 809, 919, 921, 924–927, 1012, 1014
- fm9001-state-structure-unknown-state, 1012
- fm9001-state-off, 1032
- fm9001-state-on, 1121

fm9001-state-structure\$inducti
 on, 765
 fm9001-state-structure\$step, 765
 fm9001-statep, 926–928, 1014, 1021
 fm9001-statep-implies-fm9001-st
 ate-structure, 927
 fm9001-statep-implies-macrocycle
 -invariant, 927
 -invariant-lemma1, 927
 fm9001-statep-implies-memory-ok
 -p-instance, 1021
 -p-instance-off, 1027
 -p-instance-on, 1116
 fm9001-statep-map-up-final-state, 1014
 -off, 1030
 -on, 1120
 fm9001-step, 547, 922, 924
 fm9001-step*, 922, 924
 fm9001-step*-lemma, 922
 fm9001-step*-off, 922
 fm9001-step-off, 547
 fm9001-step-theory, 873
 fm9001=chip-system, 927
 fm9001=chip-system-lemma1, 927
 fm9001=chip-system-summary, 928
 fm9001=chip-system-true-after-re
 set, 1022
 set-lemma, 1021
 set-lemma-off, 1027
 set-lemma-on, 1116
 set-off, 1027
 set-on, 1116
 fm9001=fm9001-interpreter, 925
 for-final-1-of-reset-sequence-c
 hip, 1016
 hip-off, 1016
 hip-off1, 1115
 hip-on, 1063
 fourfix, 150, 165, 194, 196
 fourp, 150, 938, 1017–1020
 fourp-f-buf, 938
 fourp-f-buf-off, 1053
 fourp-f-buf-on, 1143
 fourp-f-if, 938
 fourp-f-if-off, 1053
 fourp-f-if-on, 1143
 fourp-implies-s-approx-is-b-app
 rox, 938
 rox-off, 972
 rox-off1, 1053
 rox-on, 995
 rox-on1, 1143
 ft-buf, 194–196, 205, 408, 409, 411,
 439, 440, 761–764, 796, 802–
 805, 812–814, 936, 949, 951
 ft-buf-monotone, 936
 ft-buf-monotone-off, 1054
 ft-buf-monotone-map-up-final-state, 1143
 ft-buf-off, 937
 ft-buf-off1, 1148
 ft-buf-on, 1059
 ft-buf-rewrite, 196
 ft-wire, 194, 196, 203, 409, 439, 440,
 936, 949
 ft-wire-monotone, 936
 ft-wire-monotone-off, 1054
 ft-wire-monotone-on, 1143
 ft-wire-off, 938
 ft-wire-off1, 1148
 ft-wire-on, 1059
 ft-wire-rewrite, 196
 full-adder, 438
 full-adder*, 437, 438
 full-adder\$netlist, 438, 441
 full-adder\$value, 438
 full-adder&, 438, 441, 442
 full-adder&-off, 438
 function-call-ok, 355, 368, 371, 372
 function-call-ok-off, 355
 function-properties, 343, 344
 function-properties-off, 343
 function-synonyms, 337, 338
 function-synonyms-off, 337
 function-t-wire-ins, 336, 337
 function-t-wire-ins-off, 337
 fv-if, 203, 452, 481, 482, 525, 568,
 739, 740, 748, 762, 763, 781,

782, 793, 794, 802–804, 812,
 813
 fv-if-off, 203
 fv-if-rewrite, 203
 fv-if-rewrite-off, 203
 fv-if-v-threefix, 203
 fv-if-when-boolp-c, 203
 fv-if-when-bvp, 203
 fv-or, 202, 430, 717
 fv-or-off, 202
 fv-or=v-or, 202
 fv-shift-right, 204, 205, 523, 525
 fv-shift-right-off, 204
 fv-shift-right=v-shift-right, 204
 fv-xor, 202, 203, 432, 462
 fv-xor-off, 202
 fv-xor=v-xor, 203

 g*1*all-input-types-off, 357
 g*1*all-module-props-off, 295
 g*1*all-xs-off, 1029
 g*1*all-xs-on, 1118
 g*1*alu-dec-op-off, 486
 g*1*alu-inc-op-off, 486
 g*1*and-not-zerop-tree-off, 125
 g*1*and-not-zerop-tree-off1, 1177
 g*1*and-not-zerop-tree-on, 1088
 g*1*b-approx-off, 1057
 g*1*b-approx-on, 1146
 g*1*b-knownp-off, 928
 g*1*b-knownp-off1, 994
 g*1*b-knownp-off2, 1149
 g*1*b-knownp-on, 953
 g*1*b-knownp-on1, 1059
 g*1*b-to-nat-off, 1115
 g*1*bagdiff-off, 131
 g*1*bagdiff-off1, 1182
 g*1*bagdiff-on, 1094
 g*1*bagint-off, 131
 g*1*bagint-off1, 1182
 g*1*bagint-on, 1094
 g*1*cancel-difference-plus-off, 128
 g*1*cancel-difference-plus-off1, 1180
 g*1*cancel-difference-plus-on, 1091

 g*1*cancel-equal-plus-off, 128
 g*1*cancel-equal-plus-off1, 1180
 g*1*cancel-equal-plus-on, 1092
 g*1*cancel-equal-times-off, 125
 g*1*cancel-equal-times-off1, 1176
 g*1*cancel-equal-times-on, 1088
 g*1*cancel-factors-0-off, 103
 g*1*cancel-factors-0-off1, 1155
 g*1*cancel-factors-0-on, 1066
 g*1*cancel-factors-ilessp-0-off, 102
 1, 1154
 g*1*cancel-factors-ilessp-0-on, 1065
 g*1*cancel-ineg-aux-off, 113
 g*1*cancel-ineg-aux-off1, 1165
 g*1*cancel-ineg-aux-on, 1077
 g*1*cancel-ineg-off, 113
 g*1*cancel-ineg-off1, 1165
 g*1*cancel-ineg-on, 1077
 g*1*cancel-ineg-terms-from-equa
 lity-expanded-off, 101
 lity-expanded-off1, 1154
 lity-expanded-on, 1065
 lity-off, 101
 lity-off1, 1154
 lity-on, 1065
 g*1*cancel-ineg-terms-from-ineq
 uality-expanded-off, 100
 uality-expanded-off1, 1153
 uality-expanded-on, 1064
 uality-off, 101
 uality-off1, 1153
 uality-on, 1064
 g*1*cancel-iplus-ilessp-1-off, 111
 g*1*cancel-iplus-ilessp-1-off1, 1163
 g*1*cancel-iplus-ilessp-1-on, 1075
 g*1*cancel-iplus-ilessp-off, 111
 g*1*cancel-iplus-ilessp-off1, 1163
 g*1*cancel-iplus-ilessp-on, 1074
 g*1*cancel-iplus-off, 112
 g*1*cancel-iplus-off1, 1164
 g*1*cancel-iplus-on, 1075
 g*1*cancel-itimes-factors-expan
 ded-off, 105
 ded-off1, 1157

ded-on, 1068
 g*1*cancel-itimes-factors-off, 105
 g*1*cancel-itimes-factors-off1, 1157
 g*1*cancel-itimes-factors-on, 1068
 g*1*cancel-itimes-ilessp-factor
 s-off, 104
 s-off1, 1156
 s-on, 1068
 g*1*cancel-itimes-ilessp-off, 106
 g*1*cancel-itimes-ilessp-off1, 1158
 g*1*cancel-itimes-ilessp-on, 1069
 g*1*cancel-itimes-off, 106
 g*1*cancel-itimes-off1, 1158
 g*1*cancel-itimes-on, 1070
 g*1*cancel-lessp-plus-off, 128
 g*1*cancel-lessp-plus-off1, 1179
 g*1*cancel-lessp-plus-on, 1091
 g*1*cancel-lessp-times-off, 125
 g*1*cancel-lessp-times-off1, 1176
 g*1*cancel-lessp-times-on, 1088
 g*1*cancel-quotient-times-off, 120
 g*1*cancel-quotient-times-off1, 1172
 g*1*cancel-quotient-times-on, 1083
 g*1*chip*-off, 790
 g*1*chip-module*-off, 775
 g*1*chip-system*-off, 799
 g*1*compile-uncompiled-defns-of
 f, 1032
 g*1*compile-uncompiled-defns-on, 1122
 g*1*conjoin-inequalities-with-0
 -off, 102
 -off1, 1155
 -on, 1066
 g*1*delete-off, 131
 g*1*delete-off1, 1182
 g*1*delete-on, 1094
 g*1*disjoin-equalities-with-0-o
 ff, 103
 ff1, 1155
 n, 1066
 g*1*double-cdr-induction-off, 1040
 g*1*double-cdr-induction-on, 1129
 g*1*double-log-induction-off, 119
 g*1*double-log-induction-off1, 1170
 g*1*double-log-induction-on, 1082
 g*1*double-number-induction-off, 120
 1, 1171
 g*1*double-number-induction-on, 1083
 g*1*double-remainder-induction-
 off, 123
 off1, 1175
 on, 1086
 g*1*doublet-n-simulate-inductio
 n-off, 1033
 n-on, 1122
 g*1*doublet-p-off, 1033
 g*1*doublet-p-on, 1122
 g*1*dual-eval-monotone-inductio
 n-off, 1041
 n-on, 1130
 g*1*dual-port-ram-state-body-of
 f, 1041
 g*1*dual-port-ram-state-body-on, 1130
 g*1*dual-port-ram-value-body-of
 f, 1041
 g*1*dual-port-ram-value-body-on, 1131
 g*1*eval\$-cancel-ineg-aux-fn-o
 ff, 113
 ff1, 1165
 n, 1076
 g*1*exp-off, 120
 g*1*exp-off1, 1172
 g*1*exp-on, 1083
 g*1*extend-immediate*-off, 748
 g*1*final-state-off, 1031
 g*1*final-state-on, 1120
 g*1*fix-int-off, 116
 g*1*fix-int-off1, 1168
 g*1*fix-int-on, 1079
 g*1*flags*-off, 744
 g*1*gcd-off, 120
 g*1*gcd-off1, 1171
 g*1*gcd-on, 1083
 g*1*good-s-alist-off, 1039
 g*1*good-s-alist-on, 1128
 g*1*good-s-off, 1056
 g*1*good-s-on, 1145
 g*1*iabs-off, 116

- g*1*iabs-off1, 1167
- g*1*iabs-on, 1079
- g*1*idifference-off, 116
- g*1*idifference-off1, 1167
- g*1*idifference-on, 1079
- g*1*idiv-off, 115
- g*1*idiv-off1, 1167
- g*1*idiv-on, 1079
- g*1*ileq-off, 116
- g*1*ileq-off1, 1168
- g*1*ileq-on, 1079
- g*1*ilessp-off, 116
- g*1*ilessp-off1, 1168
- g*1*ilessp-on, 1079
- g*1*imod-off, 115
- g*1*imod-off1, 1167
- g*1*imod-on, 1079
- g*1*ineg-off, 116
- g*1*ineg-off1, 1167
- g*1*ineg-on, 1079
- g*1*initialized-machine-state-off, 1031
- g*1*initialized-machine-state-on, 1120
- g*1*initialized-memory-state-off, 1031
- g*1*initialized-memory-state-on, 1120
- g*1*initialized-regfile-off, 1031
- g*1*initialized-regfile-on, 1120
- g*1*int-to-v-off, 1111
- g*1*integer-in-range-off, 1111
- g*1*integerp-off, 116
- g*1*integerp-off1, 1168
- g*1*integerp-on, 1079
- g*1*iplus-fringe-off, 112
- g*1*iplus-fringe-off1, 1164
- g*1*iplus-fringe-on, 1076
- g*1*iplus-list-off, 112
- g*1*iplus-list-off1, 1164
- g*1*iplus-list-on, 1075
- g*1*iplus-off, 116
- g*1*iplus-off1, 1168
- g*1*iplus-on, 1079
- g*1*iplus-or-ineg-term-off, 102
- g*1*iplus-or-ineg-term-off1, 1154

- g*1*iplus-or-ineg-term-on, 1065
- g*1*iplus-or-itimes-term-off, 105
- g*1*iplus-or-itimes-term-off1, 1157
- g*1*iplus-or-itimes-term-on, 1068
- g*1*iplus-tree-no-fix-int-off, 111
- g*1*iplus-tree-no-fix-int-off1, 1163
- g*1*iplus-tree-no-fix-int-on, 1074
- g*1*iplus-tree-off, 112
- g*1*iplus-tree-off1, 1164
- g*1*iplus-tree-on, 1076
- g*1*iplus-tree-rec-off, 112
- g*1*iplus-tree-rec-off1, 1164
- g*1*iplus-tree-rec-on, 1076
- g*1*iquo-off, 115
- g*1*iquo-off1, 1167
- g*1*iquo-on, 1079
- g*1*iquotient-off, 115
- g*1*iquotient-off1, 1167
- g*1*iquotient-on, 1079
- g*1*irem-off, 115
- g*1*irem-off1, 1167
- g*1*irem-on, 1078
- g*1*iremainder-off, 115
- g*1*iremainder-off1, 1167
- g*1*iremainder-on, 1079
- g*1*itimes-factors-off, 106
- g*1*itimes-factors-off1, 1158
- g*1*itimes-factors-on, 1069
- g*1*itimes-fringe-off, 107
- g*1*itimes-fringe-off1, 1159
- g*1*itimes-fringe-on, 1070
- g*1*itimes-list-off, 107
- g*1*itimes-list-off1, 1159
- g*1*itimes-list-on, 1070
- g*1*itimes-off, 116
- g*1*itimes-off1, 1167
- g*1*itimes-on, 1079
- g*1*itimes-tree-ineg-off, 106
- g*1*itimes-tree-ineg-off1, 1158
- g*1*itimes-tree-ineg-on, 1069
- g*1*itimes-tree-no-fix-int-off, 106
- g*1*itimes-tree-no-fix-int-off1, 1158
- g*1*itimes-tree-no-fix-int-on, 1069
- g*1*itimes-tree-off, 107

g*1*itimes-tree-off1, 1159	g*1*n_fetch3-off, 594
g*1*itimes-tree-on, 1070	g*1*n_hold0-off, 600
g*1*itimes-tree-rec-off, 107	g*1*n_hold1-off, 600
g*1*itimes-tree-rec-off1, 1159	g*1*n_reada0-off, 595
g*1*itimes-tree-rec-on, 1070	g*1*n_reada1-off, 595
g*1*izerop-off, 116	g*1*n_reada2-off, 596
g*1*izerop-off1, 1168	g*1*n_reada3-off, 596
g*1*izerop-on, 1079	g*1*n_readb0-off, 596
g*1*log-off, 120	g*1*n_readb1-off, 597
g*1*log-off1, 1172	g*1*n_readb2-off, 597
g*1*log-on, 1083	g*1*n_readb3-off, 597
g*1*lsi-keywords-off, 285	g*1*n_rega-off, 594
g*1*make-cancel-ineg-terms-equa	g*1*n_regb-off, 595
lity-off, 102	g*1*n_reset0-off, 601
lity-off1, 1154	g*1*n_reset1-off, 602
lity-on, 1065	g*1*n_reset2-off, 602
g*1*make-cancel-ineg-terms-ineq	g*1*n_sefa0-off, 599
uality-off, 101	g*1*n_sefa1-off, 599
uality-off1, 1153	g*1*n_sefb0-off, 599
uality-on, 1064	g*1*n_sefb1-off, 600
g*1*make-cancel-iplus-inequalit	g*1*n_update-off, 595
y-1-off, 112	g*1*n_v11010-off, 601
y-1-off1, 1163	g*1*n_v11011-off, 601
y-1-on, 1075	g*1*n_v11111-off, 602
g*1*make-cancel-itimes-equality	g*1*n_write0-off, 598
-off, 107	g*1*n_write1-off, 598
-off1, 1159	g*1*n_write2-off, 598
-on, 1070	g*1*n_write3-off, 599
g*1*make-cancel-itimes-inequalit	g*1*nat-lst-lst-induction-off, 1034
y-off, 106	g*1*nat-lst-lst-induction-on, 1123
y-off1, 1158	g*1*new-machine-state-invariant
y-on, 1069	-off, 1029
g*1*mem-width-off, 973	-on, 1118
g*1*mem-width-off1, 1056	g*1*occurrences-off, 131
g*1*mem-width-on, 996	g*1*occurrences-off1, 1182
g*1*mem-width-on1, 1146	g*1*occurrences-on, 1094
g*1*memory-v-fourp-off, 1029	g*1*ok-netlistp-off, 1041
g*1*memory-v-fourp-on, 1118	g*1*ok-netlistp-on, 1130
g*1*monotonicity-property-off, 1056	g*1*or-zerop-tree-off, 125
g*1*monotonicity-property-on, 1145	g*1*or-zerop-tree-off1, 1177
g*1*n_decode-off, 594	g*1*or-zerop-tree-on, 1089
g*1*n_fetch0-off, 593	g*1*plus-fringe-off, 129
g*1*n_fetch1-off, 593	g*1*plus-fringe-off1, 1180
g*1*n_fetch2-off, 593	g*1*plus-fringe-on, 1092

- g*1*plus-tree-off, 129
- g*1*plus-tree-off1, 1180
- g*1*plus-tree-on, 1092
- g*1*predicate-properties-off, 354
- g*1*primitive-properties-off, 295
- g*1*read-mem1-monotone-inductio
 - n-off, 1042
 - n-on, 1131
- g*1*regfile*-off, 738
- g*1*remove-inegs-off, 102
- g*1*remove-inegs-off1, 1154
- g*1*remove-inegs-on, 1065
- g*1*reset-sequence-chip-1-off, 1030
- g*1*reset-sequence-chip-1-on, 1119
- g*1*reset-sequence-chip-2-off, 1030
- g*1*reset-sequence-chip-2-on, 1119
- g*1*reset-sequence-off, 1031
- g*1*reset-sequence-on, 1121
- g*1*reset-vector-chip-off, 1030
- g*1*reset-vector-chip-on, 1119
- g*1*reset-vector-off, 1032
- g*1*reset-vector-on, 1121
- g*1*run-vector-chip-1-off, 1030
- g*1*run-vector-chip-1-on, 1119
- g*1*run-vector-chip-2-off, 1030
- g*1*run-vector-chip-2-on, 1119
- g*1*run-vector-off, 1032
- g*1*run-vector-on, 1121
- g*1*s-approx-alist-off, 1056
- g*1*s-approx-alist-on, 1145
- g*1*s-approx-list-off, 1040
- g*1*s-approx-list-on, 1129
- g*1*s-approx-off, 1056
- g*1*s-approx-on, 1146
- g*1*s-knownp-off, 1056
- g*1*s-knownp-on, 1145
- g*1*simulate-monotone-induction
 - off, 1033
 - on, 1123
- g*1*single-number-induction-off, 1181, 1169
- g*1*single-number-induction-on, 1081
- g*1*some-eval\$s-to-0-off, 103
- g*1*some-eval\$s-to-0-off1, 1155

- g*1*some-eval\$s-to-0-on, 1066
- g*1*split-out-ineg-terms-off, 102
- g*1*split-out-ineg-terms-off1, 1154
- g*1*split-out-ineg-terms-on, 1065
- g*1*subbagp-off, 131
- g*1*subbagp-off1, 1182
- g*1*subbagp-on, 1094
- g*1*subsetp-off, 106
- g*1*subsetp-off1, 1158
- g*1*subsetp-on, 1070
- g*1*times-fringe-off, 126
- g*1*times-fringe-off1, 1177
- g*1*times-fringe-on, 1089
- g*1*times-tree-off, 126
- g*1*times-tree-off1, 1177
- g*1*times-tree-on, 1089
- g*1*unknown-machine-state-off, 1032
- g*1*unknown-machine-state-on, 1121
- g*1*unknown-memory-state-off, 1032
- g*1*unknown-memory-state-on, 1121
- g*1*unknown-regfile-off, 1032
- g*1*unknown-regfile-on, 1121
- g*1*unknown-state-off, 1032
- g*1*unknown-state-on, 1121
- g*1*v-alu-int-adder-off, 1110
- g*1*v-alu-int-adder-output-off, 1110
- g*1*v-alu-int-adder-overflowp-off, 1110
- g*1*v-alu-int-asr-off, 1109
- g*1*v-alu-int-asr-output-off, 1109
- g*1*v-alu-int-dec-off, 1109
- g*1*v-alu-int-inc-off, 1110
- g*1*v-alu-int-neg-off, 1109
- g*1*v-alu-int-not-off, 1108
- g*1*v-alu-int-not-output-off, 1108
- g*1*v-alu-int-off, 1108
- g*1*v-alu-int-subtractor-off, 1109
- g*1*v-alu-int-subtractor-output-off, 1109
- g*1*v-alu-int-subtractor-overflowp-off, 1109
- g*1*v-alu-nat-adder-carry-out-off, 1113
- g*1*v-alu-nat-adder-off, 1113

g*1*v-alu-nat-adder-output-off, 1113
 g*1*v-alu-nat-dec-off, 1112
 g*1*v-alu-nat-inc-off, 1113
 g*1*v-alu-nat-int-buf-off, 1115
 g*1*v-alu-nat-lsr-carry-out-off, 1112
 g*1*v-alu-nat-lsr-off, 1112
 g*1*v-alu-nat-lsr-output-off, 1112
 g*1*v-alu-nat-not-off, 1111
 g*1*v-alu-nat-not-output-off, 1112
 g*1*v-alu-nat-off, 1111
 g*1*v-alu-nat-subtractor-carry-
 out-off, 1112
 g*1*v-alu-nat-subtractor-off, 1112
 g*1*v-alu-nat-subtractor-output
 -off, 1113
 g*1*v-approx-alist-off, 1056
 g*1*v-approx-alist-on, 1145
 g*1*v-approx-list-off, 1034
 g*1*v-approx-list-on, 1123
 g*1*v-approx-off, 1057
 g*1*v-approx-on, 1146
 g*1*v-fourp-off, 1029
 g*1*v-fourp-on, 1118
 g*1*v-knownp-off, 1057
 g*1*v-knownp-on, 1146
 g*1*v-s-approx-list-off, 1034
 g*1*v-s-approx-list-on, 1123
 g*1*v-to-int-off, 1111
 g*1*v_decode-off, 594
 g*1*v_fetch0-off, 593
 g*1*v_fetch1-off, 593
 g*1*v_fetch2-off, 593
 g*1*v_fetch3-off, 593
 g*1*v_hold0-off, 600
 g*1*v_hold1-off, 600
 g*1*v_reada0-off, 595
 g*1*v_reada1-off, 595
 g*1*v_reada2-off, 596
 g*1*v_reada3-off, 596
 g*1*v_readb0-off, 596
 g*1*v_readb1-off, 597
 g*1*v_readb2-off, 597
 g*1*v_readb3-off, 597
 g*1*v_rega-off, 594

g*1*v_regb-off, 594
 g*1*v_reset0-off, 601
 g*1*v_reset1-off, 601
 g*1*v_reset2-off, 602
 g*1*v_sefa0-off, 599
 g*1*v_sefa1-off, 599
 g*1*v_sefb0-off, 599
 g*1*v_sefb1-off, 600
 g*1*v_update-off, 595
 g*1*v_v11010-off, 601
 g*1*v_v11011-off, 601
 g*1*v_v11111-off, 602
 g*1*v_write0-off, 597
 g*1*v_write1-off, 598
 g*1*v_write2-off, 598
 g*1*v_write3-off, 598
 g*1*write-mem1-double-induction
 -off, 1041
 -on, 1130
 g*1*write-mem1-monotone-inducti
 on-off, 1041
 on-on, 1131
 g-cell, 499–501
 g-cell*, 500
 g-cell\$netlist, 500, 502
 g-cell\$value, 500
 g-cell\$value-off, 500
 g-cell\$value-zero, 501
 g-cell&, 500–502
 g-cell&-off, 500
 gcd, 30, 33, 34
 gcd-0, 33
 gcd-0-off, 118
 gcd-0-off1, 1169
 gcd-0-on, 1081
 gcd-1, 33
 gcd-1-off, 117
 gcd-1-off1, 1169
 gcd-1-on, 1081
 gcd-idempotence, 34
 gcd-idempotence-off, 116
 gcd-idempotence-off1, 1168
 gcd-idempotence-on, 1080
 gcd-is-the-greatest, 34

gcd-is-the-greatest-off, 117
gcd-is-the-greatest-off1, 1168
gcd-is-the-greatest-on, 1080
gcd-off, 120
gcd-off1, 1171
gcd-on, 1083
gcd-plus, 33
gcd-plus-instance, 33
gcd-plus-instance-off, 117
gcd-plus-instance-off1, 1169
gcd-plus-instance-on, 1080
gcd-plus-instance-temp, 33
gcd-plus-instance-temp-off, 117
gcd-plus-instance-temp-off1, 1169
gcd-plus-instance-temp-on, 1080
gcd-plus-instance-temp-proof, 33
gcd-plus-instance-temp-proof-of
 f, 117
 f1, 1169
gcd-plus-instance-temp-proof-on, 1081
gcd-plus-off, 117
gcd-plus-off1, 1169
gcd-plus-on, 1080
gcd-plus-proof, 33
gcd-plus-proof-off, 117
gcd-plus-proof-off1, 1169
gcd-plus-proof-on, 1080
gcd-x-x, 34
gcd-x-x-off, 116
gcd-x-x-off1, 1168
gcd-x-x-on, 1080
gcds, 34
generate-append, 445
get-required-props, 352
get-required-props-off, 352
good-s, 930, 975, 976, 993, 998, 1003–
 1009, 1011, 1016, 1020, 1024
good-s-0, 1004
good-s-0-off, 1038
good-s-0-on, 1128
good-s-alist, 1003, 1004, 1008
good-s-alist-off, 1039
good-s-alist-on, 1128
good-s-alist-pairlist, 1003
good-s-alist-pairlist-off, 1039
good-s-alist-pairlist-on, 1128
good-s-collect-value, 1004
good-s-collect-value-off, 1039
good-s-collect-value-on, 1128
good-s-constant-ram, 1007
good-s-constant-ram-off, 1034
good-s-constant-ram-on, 1123
good-s-off, 1056
good-s-on, 1145
good-s-opener, 1020
good-s-opener-off, 1028
good-s-opener-on, 1117
good-s-preserved, 1008
good-s-preserved-off, 1034
good-s-preserved-on, 1123
good-s-primitives-theory, 1008
good-s-value, 1003
good-s-value-off, 1039
good-s-value-on, 1128
good-s-write-mem, 1007
good-s-write-mem-1, 1007
good-s-write-mem-1-off, 1034
good-s-write-mem-1-on, 1123
good-s-write-mem-off, 1034
good-s-write-mem-on, 1123
half-adder*, 437
half-adder\$netlist, 437, 438
half-adder\$value, 437
half-adder\$value-off, 437
half-adder&, 437, 438
half-adder&-off, 437
hdack-, 604, 607, 609, 611, 613, 615,
 617, 618, 620, 622, 624, 626,
 628, 630, 632, 634, 636, 638,
 640, 643, 644, 646, 648, 650,
 652, 654, 656, 658, 660, 662,
 664, 666, 668, 761–764, 784,
 796, 802–805, 811–814
hdack-off, 604
hierarchical-name-max, 287, 290
hierarchical-name-max-off, 287
high-level-state-structure, 925, 926

high-level-state-structure-off, 925
 hold-, 760, 763
 hold-input, 760, 763, 813, 815, 817,
 819, 888, 890, 891, 915, 916
 hold-input-off, 761
 hold-off, 760

 i-name, 149, 249, 253, 276, 435
 i-num, 149, 167, 249, 435
 i-reg, 759, 761–764
 i-reg-off, 760
 iabs, 35, 38, 50–52
 iabs-off, 37
 id, 155, 565, 670
 id-monotone, 948
 id-monotone-off, 1045
 id-monotone-on, 1134
 id-out-prop-signals, 356, 360
 id-out-prop-signals-off, 356
 id-preserves-good-s, 1006
 id-preserves-good-s-off, 1035
 id-preserves-good-s-on, 1125
 id\$netlist, 404, 469, 565, 568–570,
 672, 688, 714, 739, 776
 id\$value, 404
 id\$value-off, 404
 id&, 404, 469, 565, 568–570, 672,
 688, 713, 738, 776
 id&-off, 404
 idifference, 35–39, 46, 95, 1102–1104,
 1196, 1197, 1210–1213, 1219
 idifference-0, 1196
 idifference-difference, 1103
 idifference-difference-off, 1110
 idifference-fix-int1, 39
 idifference-fix-int1-off, 114
 idifference-fix-int1-off1, 1165
 idifference-fix-int1-on, 1077
 idifference-fix-int2, 39
 idifference-fix-int2-off, 113
 idifference-fix-int2-off1, 1165
 idifference-fix-int2-on, 1077
 idifference-off, 116
 idifference-off1, 1167

idifference-on, 1079
 idiv, 36, 51, 53, 54, 1103, 1105, 1214–
 1216, 1219
 idiv-fix-int1, 53
 idiv-fix-int1-off, 108
 idiv-fix-int1-off1, 1160
 idiv-fix-int1-on, 1071
 idiv-fix-int2, 53
 idiv-fix-int2-off, 108
 idiv-fix-int2-off1, 1160
 idiv-fix-int2-on, 1071
 idiv-ilessp-0, 1103
 idiv-ilessp-0-off, 1110
 idiv-imod-uniqueness, 51
 idiv-imod-uniqueness-off, 109
 idiv-imod-uniqueness-off1, 1161
 idiv-imod-uniqueness-on, 1073
 idiv-off, 54
 idiv-quotient, 1103
 idiv-quotient-off, 1110
 if*, 131, 132, 136, 137, 151, 165,
 182, 183, 203, 475, 625, 635,
 641, 642, 649, 653, 654, 723–
 725, 727–732, 748, 820, 828–
 831, 835, 836, 838, 843, 846,
 848, 850, 851, 853, 857, 859,
 868, 869, 872, 893–896, 898,
 899, 901, 902, 904–911, 913
 if*-c-x-x, 132
 if*-cons, 132
 if*-off, 131
 if-v-subtractor-output=0-then-a
 =b, 1189
 ileq, 35, 1102, 1204, 1205, 1207–1209,
 1211–1213, 1216
 ileq-off, 116
 ileq-off1, 1168
 ileq-on, 1079
 illessp, 35, 47, 48, 50–52, 60–63, 67,
 80–82, 85, 95, 96, 1102, 1103,
 1203–1205, 1207–1209, 1211–
 1213, 1216
 illessp-0-itimes, 81
 illessp-0-itimes-off, 86

- ilessp-add1, 95
- ilessp-add1-iplus, 95
- ilessp-add1-iplus-off, 100
- ilessp-add1-iplus-off1, 1152
- ilessp-add1-iplus-on, 1063
- ilessp-add1-off, 100
- ilessp-add1-off1, 1152
- ilessp-add1-on, 1063
- ilessp-fix-int-1, 47
- ilessp-fix-int-1-off, 112
- ilessp-fix-int-1-off1, 1164
- ilessp-fix-int-1-on, 1075
- ilessp-fix-int-2, 47
- ilessp-fix-int-2-off, 112
- ilessp-fix-int-2-off1, 1163
- ilessp-fix-int-2-on, 1075
- ilessp-ineg-ineg, 67
- ilessp-ineg-ineg-off, 105
- ilessp-ineg-ineg-off1, 1157
- ilessp-ineg-ineg-on, 1069
- ilessp-itimes-0, 81
- ilessp-itimes-0-off, 86
- ilessp-itimes-list-eval\$list-
 - bagdiff, 81
 - bagdiff-corollary-1, 81
 - bagdiff-corollary-1-off, 104
 - bagdiff-corollary-1-off1, 1156
 - bagdiff-corollary-1-on, 1067
 - bagdiff-corollary-2, 81
 - bagdiff-corollary-2-off, 104
 - bagdiff-corollary-2-off1, 1156
 - bagdiff-corollary-2-on, 1067
 - bagdiff-off, 104
 - bagdiff-off1, 1156
 - bagdiff-on, 1067
- delete, 80
- delete-helper-1, 80
- delete-helper-1-off, 86
- delete-helper-2, 80
- delete-helper-2-off, 86
- delete-off, 86
- delete-prime, 80
- delete-prime-helper-1, 80
- delete-prime-helper-1-off, 86
- delete-prime-helper-2, 80
- delete-prime-helper-2-off, 86
- delete-prime-off, 86
- ilessp-itimes-right-negative, 62
- ilessp-itimes-right-negative-of
 - f, 106
 - f1, 1158
- ilessp-itimes-right-negative-on, 1069
- ilessp-itimes-right-positive, 61
- ilessp-itimes-right-positive-of
 - f, 106
 - f1, 1158
- ilessp-itimes-right-positive-on, 1069
- ilessp-lessp, 1102
- ilessp-lessp-off, 1111
- ilessp-off, 37
- ilessp-strict, 63
- ilessp-strict-off, 106
- ilessp-strict-off1, 1158
- ilessp-strict-on, 1069
- ilessp-trichotomy, 62
- ilessp-trichotomy-off, 63
- ilessp-trichotomy-off1, 83
- ilessp-trichotomy-on, 82
- ilessp-zero-implies-not-equal, 81
- ilessp-zero-implies-not-equal-o
 - ff, 104
 - ff1, 1156
 - n, 1067
- imod, 36, 51, 53, 54, 1214, 1215
- imod-fix-int1, 53
- imod-fix-int1-off, 108
- imod-fix-int1-off1, 1160
- imod-fix-int1-on, 1071
- imod-fix-int2, 53
- imod-fix-int2-off, 108
- imod-fix-int2-off1, 1160
- imod-fix-int2-on, 1071
- imod-off, 54
- imod-v-to-int-2, 1214
- in-types-error, 305, 349
- in-types-error-off, 305
- index, 148, 149, 428–435, 440–443,
 - 449–462, 469, 470, 472–482,

513–517, 527–529, 537–541,
 679–682, 705–708, 712, 713,
 720, 735–738, 742, 743, 747,
 749–759
 indexp, 149, 249, 253, 268–272, 276,
 297, 361, 362, 373, 435
 indices, 148, 149, 166, 167, 428–435,
 440–443, 449–453, 457–462,
 470, 471, 473, 474, 476–481,
 513, 516, 527, 528, 537–540,
 679, 704, 705, 708, 709, 711–
 713, 719–721, 735–738, 742,
 743, 746, 747, 750–755, 757–
 759
 indices-as-append, 148
 indices-as-append-off, 148
 indices-off, 148
 indices-zero, 148
 ineg, 35, 37–39, 42, 46, 47, 52, 67,
 92, 1102, 1103, 1198–1200,
 1208, 1209, 1219
 ineg-0, 38
 ineg-0-off, 114
 ineg-0-off1, 1166
 ineg-0-on, 1078
 ineg-as-idifference, 1219
 ineg-eval\$-itimes-tree-ineg, 67
 ineg-eval\$-itimes-tree-ineg-of
 f, 105
 f1, 1157
 ineg-eval\$-itimes-tree-ineg-on, 1068
 ineg-fix-int, 38
 ineg-fix-int-off, 114
 ineg-fix-int-off1, 1166
 ineg-fix-int-on, 1078
 ineg-ineg, 38
 ineg-ineg-off, 114
 ineg-ineg-off1, 1166
 ineg-ineg-on, 1078
 ineg-iplus, 38
 ineg-iplus-off, 114
 ineg-iplus-off1, 1166
 ineg-iplus-on, 1078
 ineg-of-non-integerp, 38
 ineg-of-non-integerp-off, 38
 ineg-off, 37
 infer-equality-from-not-lessp, 17
 infer-equality-from-not-lessp-o
 ff, 125
 ff1, 1176
 n, 1088
 initial-lsi-occ-syntax-data, 290, 293
 initial-lsi-occ-syntax-data-off, 290
 initial-occ-data, 340, 341, 346
 initial-occ-data-off, 341
 initial-occ-delays, 321, 340
 initial-occ-drives, 307, 340, 341
 initial-occ-in-types, 300, 340, 341
 initial-occ-loadings, 307, 340, 341
 initial-occ-out-depends, 334, 340, 341
 initial-occ-out-depends-off, 334
 initial-occ-out-types, 300, 340, 341
 initial-occ-state-types, 336, 340, 341
 initial-occ-syntax-data, 280, 282, 290
 initial-occ-syntax-data-off, 280
 initial-occ-tri-state-data, 336, 340, 341
 initial-occ-tri-state-data-off, 336
 initialized-machine-state, 1013, 1014,
 1017, 1021–1026
 initialized-machine-state-off, 1031
 initialized-machine-state-on, 1120
 initialized-memory-state, 1013, 1014
 initialized-memory-state-off, 1031
 initialized-memory-state-on, 1120
 initialized-regfile, 1013
 initialized-regfile-off, 1031
 initialized-regfile-on, 1120
 input-delay, 329, 332, 333
 input-delay-off, 330
 input-delay0, 327–330
 input-delay0-off, 329
 input-label, 355, 357, 364
 input-label-off, 355
 input-type, 299, 303
 input-type-off, 300
 insert, 245, 247, 276, 285, 298, 324,
 325, 369
 insert-input-delay, 325–327

insert-input-delay-off, 325
 insert-off, 245
 instance-theorem, 1014
 instance-theorem-off, 1030
 instance-theorem-on, 1120
 int-dec-range-lemmas, 1209
 int-neg-range-implications, 1208
 int-to-bv, 549, 553, 556
 int-to-v, 1102, 1104, 1105, 1197–1200,
 1208–1210, 1219
 int-to-v-of-v-to-int, 1219
 int-to-v-off, 1111
 int-to-v-v-to-int-0, 1197
 integer-defns, 37
 integer-in-range, 1102, 1104, 1197–
 1201, 1204, 1206–1211, 1213,
 1214, 1216
 integer-in-range-1, 1200
 integer-in-range-0, 1200
 integer-in-range-1, 1197
 integer-in-range-idiv, 1214
 integer-in-range-iplus, 1198
 integer-in-range-iplus-carry, 1199
 integer-in-range-iplus-carry-o
 ff, 1199
 integer-in-range-iplus-carry\$
 commuted, 1199
 integer-in-range-iplus-ineg, 1199
 integer-in-range-iplus-ineg-ca
 rry, 1199
 rry-off, 1199
 rry\$commuted, 1200
 integer-in-range-iplus-ineg-of
 f, 1199
 integer-in-range-iplus-ineg\$c
 ommutated, 1199
 integer-in-range-iplus-off, 1198
 integer-in-range-iplus\$commute
 d, 1198
 integer-in-range-minus-0, 1197
 integer-in-range-off, 1111
 integer-in-range-the-obvious-w
 ay, 1198
 ay-off, 1198
 integer-in-range-v-to-int, 1198
 integer-metas, 1095
 integer-minus, 549
 integerp, 34, 35, 37–39, 43, 45–47,
 50–53, 55, 57, 58, 78, 79,
 84, 92, 95, 1196, 1198, 1199
 integerp-eval\$-iplus-or-ineg-te
 rm, 92
 rm-off, 95
 integerp-eval\$-itimes, 57
 integerp-eval\$-itimes-off, 64
 integerp-fix-int, 37
 integerp-fix-int-off, 115
 integerp-fix-int-off1, 1167
 integerp-fix-int-on, 1078
 integerp-iabs, 38
 integerp-iabs-off, 115
 integerp-iabs-off1, 1167
 integerp-iabs-on, 1078
 integerp-idifference, 37
 integerp-idifference-off, 115
 integerp-idifference-off1, 1167
 integerp-idifference-on, 1078
 integerp-idiv, 53
 integerp-idiv-off, 108
 integerp-idiv-off1, 1160
 integerp-idiv-on, 1072
 integerp-imod, 53
 integerp-imod-off, 108
 integerp-imod-off1, 1160
 integerp-imod-on, 1072
 integerp-ineg, 37
 integerp-ineg-off, 115
 integerp-ineg-off1, 1167
 integerp-ineg-on, 1078
 integerp-iplus, 37
 integerp-iplus-list, 43
 integerp-iplus-list-off, 112
 integerp-iplus-list-off1, 1164
 integerp-iplus-list-on, 1075
 integerp-iplus-off, 115
 integerp-iplus-off1, 1167
 integerp-iplus-on, 1078
 integerp-iquo, 53

- integerp-iquo-off, 108
- integerp-iquo-off1, 1160
- integerp-iquo-on, 1071
- integerp-iquotient, 52
- integerp-iquotient-off, 108
- integerp-iquotient-off1, 1160
- integerp-iquotient-on, 1072
- integerp-irem, 53
- integerp-irem-off, 108
- integerp-irem-off1, 1160
- integerp-irem-on, 1071
- integerp-iremainder, 52
- integerp-iremainder-off, 108
- integerp-iremainder-off1, 1160
- integerp-iremainder-on, 1072
- integerp-itimes, 38
- integerp-itimes-list, 55
- integerp-itimes-list-off, 107
- integerp-itimes-list-off1, 1159
- integerp-itimes-list-on, 1070
- integerp-itimes-off, 115
- integerp-itimes-off1, 1167
- integerp-itimes-on, 1078
- integerp-off, 37
- integerp-v-to-int, 1196
- integers, 97
- intersection, 245, 252, 280, 339, 344,
351, 358, 359, 365, 366, 380,
383, 384
- intersection-off, 245
- intp, 549, 553, 556
- io-car, 356, 357, 359, 362–364, 366,
369, 372
- io-car-off, 356
- io-cdr, 356, 357, 359, 362–364, 366,
369, 372
- io-cdr-off, 356
- io-label, 355, 372
- io-label-off, 355
- io-out, 296
- io-out-signal, 296
- io-outp, 296, 297, 302, 338
- io-rename-error, 338, 339
- io-rename-error-off, 338
- io-signal-renames, 338
- io-signal-renames-off, 338
- io-type-error, 301, 302
- io-type-error-off, 302
- io-types, 382, 384, 385
- io-types-collector, 382, 383
- io-types-collector-off, 382
- io-types-compatible, 302, 303
- io-types-compatible-off, 302
- io-types-off, 384
- io-types-simple, 259–262, 264, 265
- io-types-simple-off, 262
- iplus, 35, 37–39, 42, 43, 46–52, 67,
92, 95, 96, 1102–1104, 1197–
1200, 1204–1208, 1210–1212,
1219
- iplus-0-left, 39
- iplus-0-left-off, 114
- iplus-0-left-off1, 1166
- iplus-0-left-on, 1077
- iplus-0-left-on1, 1186
- iplus-0-right, 39
- iplus-0-right-off, 114
- iplus-0-right-off1, 1166
- iplus-0-right-on, 1077
- iplus-0-right-on1, 1186
- iplus-cancellation-1, 39
- iplus-cancellation-1-for-ilessp, 48
-off, 111
-off1, 1163
-on, 1075
- iplus-cancellation-1-off, 114
- iplus-cancellation-1-off1, 1166
- iplus-cancellation-1-on, 1077
- iplus-cancellation-2, 39
- iplus-cancellation-2-for-ilessp, 48
-off, 111
-off1, 1163
-on, 1075
- iplus-cancellation-2-off, 114
- iplus-cancellation-2-off1, 1166
- iplus-cancellation-2-on, 1077
- iplus-constants, 95
- iplus-constants-off, 100

- iplus-constants-off1, 1153
- iplus-constants-on, 1063
- iplus-div-hack1, 1103
- iplus-div-hack1-off, 1110
- iplus-eval\$-itimes-tree-ineg, 67
- iplus-eval\$-itimes-tree-ineg-o
 - ff, 105
 - ff1, 1157
 - n, 1068
- iplus-fix-int1, 39
- iplus-fix-int1-off, 114
- iplus-fix-int1-off1, 1165
- iplus-fix-int1-on, 1077
- iplus-fix-int2, 39
- iplus-fix-int2-off, 114
- iplus-fix-int2-off1, 1165
- iplus-fix-int2-on, 1077
- iplus-fringe, 42, 44–49, 88–94
- iplus-fringe-off, 112
- iplus-fringe-off1, 1164
- iplus-fringe-on, 1076
- iplus-ineg-promote, 42
- iplus-ineg-promote-off, 42
- iplus-ineg1, 39
- iplus-ineg1-off, 114
- iplus-ineg1-off1, 1165
- iplus-ineg1-on, 1077
- iplus-ineg2, 39
- iplus-ineg2-off, 114
- iplus-ineg2-off1, 1165
- iplus-ineg2-on, 1077
- iplus-ineg3, 42
- iplus-ineg3-off, 113
- iplus-ineg3-off1, 1164
- iplus-ineg3-on, 1076
- iplus-ineg4, 42
- iplus-ineg4-off, 113
- iplus-ineg4-off1, 1164
- iplus-ineg4-on, 1076
- iplus-ineg5, 46
- iplus-ineg5-lemma-1, 46
- iplus-ineg5-lemma-1-off, 47
- iplus-ineg5-lemma-2, 46
- iplus-ineg5-lemma-2-off, 47
- iplus-ineg5-off, 47
- iplus-ineg6, 47
- iplus-ineg6-off, 47
- iplus-ineg7, 47
- iplus-ineg7-off, 112
- iplus-ineg7-off1, 1164
- iplus-ineg7-on, 1075
- iplus-left-id, 38
- iplus-left-id-off, 39
- iplus-list, 43, 45, 46, 92
- iplus-list-append, 46
- iplus-list-append-off, 112
- iplus-list-append-off1, 1164
- iplus-list-append-on, 1075
- iplus-list-eval\$-car-split-out
 - ineg-terms, 92
 - ineg-terms-off, 101
 - ineg-terms-off1, 1153
 - ineg-terms-on, 1064
- iplus-list-eval\$-fringe, 46
- iplus-list-eval\$-fringe-off, 63
- iplus-list-off, 112
- iplus-list-off1, 1164
- iplus-list-on, 1075
- iplus-off, 37
- iplus-or-ineg-term, 88, 92, 93
- iplus-or-ineg-term-off, 92
- iplus-or-itimes-term, 68, 79
- iplus-or-itimes-term-integerp-eval\$
 - al\$, 79
 - al\$-off, 86
- iplus-or-itimes-term-off, 78
- iplus-plus, 1102
- iplus-plus-off, 1111
- iplus-right-id, 39
- iplus-right-id-off, 39
- iplus-tree, 43–45, 47, 48, 88–94
- iplus-tree-no-fix-int, 48, 49
- iplus-tree-no-fix-int-off, 48
- iplus-tree-off, 46
- iplus-tree-rec, 43, 48
- iplus-tree-rec-off, 112
- iplus-tree-rec-off1, 1164
- iplus-tree-rec-on, 1076

- iplus-x-y-ineg-x, 42
- iplus-x-y-ineg-x-off, 113
- iplus-x-y-ineg-x-off1, 1164
- iplus-x-y-ineg-x-on, 1076
- iplus3-plus-difference, 1102
- iplus3-plus-difference-off, 1110
- iquo, 36, 37, 51–54
- iquo-fix-int1, 53
- iquo-fix-int1-off, 108
- iquo-fix-int1-off1, 1160
- iquo-fix-int1-on, 1071
- iquo-fix-int2, 53
- iquo-fix-int2-off, 108
- iquo-fix-int2-off1, 1160
- iquo-fix-int2-on, 1071
- iquo-irem-uniqueness, 52
- iquo-irem-uniqueness-off, 109
- iquo-irem-uniqueness-off1, 1161
- iquo-irem-uniqueness-on, 1072
- iquo-off, 54
- iquotient, 36, 50–54
- iquotient-fix-int1, 53
- iquotient-fix-int1-off, 108
- iquotient-fix-int1-off1, 1160
- iquotient-fix-int1-on, 1071
- iquotient-fix-int2, 53
- iquotient-fix-int2-off, 108
- iquotient-fix-int2-off1, 1160
- iquotient-fix-int2-on, 1071
- iquotient-remainder-uniqueness, 51
 - off, 110
 - off1, 1162
 - on, 1073
- iquotient-off, 54
- ir-fields-theory, 542
- irem, 37, 51–54
- irem-fix-int1, 53
- irem-fix-int1-off, 107
- irem-fix-int1-off1, 1159
- irem-fix-int1-on, 1071
- irem-fix-int2, 53
- irem-fix-int2-off, 107
- irem-fix-int2-off1, 1159
- irem-fix-int2-on, 1071
- irem-off, 54
- iremainder, 36, 50–54
- iremainder-fix-int1, 53
- iremainder-fix-int1-off, 108
- iremainder-fix-int1-off1, 1160
- iremainder-fix-int1-on, 1071
- iremainder-fix-int2, 53
- iremainder-fix-int2-off, 108
- iremainder-fix-int2-off1, 1160
- iremainder-fix-int2-on, 1071
- iremainder-off, 54
- is-head, 246, 365
- is-head-off, 246
- itimes, 35–38, 49–52, 55, 57, 59, 61, 62, 67, 68, 78–81
- itimes-1, 67
- itimes-1-off, 105
- itimes-1-off1, 1157
- itimes-1-on, 1069
- itimes-0-left, 49
- itimes-0-left-off, 111
- itimes-0-left-off1, 1163
- itimes-0-left-on, 1074
- itimes-0-right, 49
- itimes-0-right-off, 111
- itimes-0-right-off1, 1163
- itimes-0-right-on, 1074
- itimes-1-arg1, 50
- itimes-1-arg1-off, 110
- itimes-1-arg1-off1, 1162
- itimes-1-arg1-on, 1073
- itimes-cancellation-1, 52
- itimes-cancellation-1-off, 109
- itimes-cancellation-1-off1, 1161
- itimes-cancellation-1-on, 1072
- itimes-cancellation-2, 52
- itimes-cancellation-2-off, 109
- itimes-cancellation-2-off1, 1161
- itimes-cancellation-2-on, 1072
- itimes-cancellation-3, 52
- itimes-cancellation-3-off, 108
- itimes-cancellation-3-off1, 1160
- itimes-cancellation-3-on, 1072
- itimes-distributes-over-iplus, 49

- itimes-distributes-over-iplus-o
 - ff, 110
 - ff1, 1162
 - n, 1074
- itimes-distributes-over-iplus-p
 - roof, 49
 - roof-off, 111
 - roof-off1, 1163
 - roof-on, 1074
- itimes-eval\$-itimes-tree-ineg, 68
- itimes-eval\$-itimes-tree-ineg-
 - off, 105
 - off1, 1157
 - on, 1068
- itimes-factors, 66–69, 79–85
- itimes-factors-off, 105
- itimes-factors-off1, 1157
- itimes-factors-on, 1069
- itimes-fix-int1, 49
- itimes-fix-int1-off, 111
- itimes-fix-int1-off1, 1163
- itimes-fix-int1-on, 1074
- itimes-fix-int2, 49
- itimes-fix-int2-off, 111
- itimes-fix-int2-off1, 1163
- itimes-fix-int2-on, 1074
- itimes-fringe, 54–62
- itimes-fringe-off, 107
- itimes-fringe-off1, 1159
- itimes-fringe-on, 1070
- itimes-ineg-1, 52
- itimes-ineg-1-off, 109
- itimes-ineg-1-off1, 1161
- itimes-ineg-1-on, 1072
- itimes-ineg-2, 52
- itimes-ineg-2-off, 109
- itimes-ineg-2-off1, 1161
- itimes-ineg-2-on, 1072
- itimes-itimes-list-eval\$list-
 - delete, 78
 - delete-off, 85
- itimes-list, 55, 57–63, 67, 68, 78–82,
 - 84, 85
- itimes-list-append, 57
- itimes-list-append-off, 106
- itimes-list-append-off1, 1158
- itimes-list-append-on, 1070
- itimes-list-bagdiff, 57
- itimes-list-bagdiff-off, 64
- itimes-list-eval\$-delete, 57
- itimes-list-eval\$-delete-off, 57
- itimes-list-eval\$-factors, 79
- itimes-list-eval\$-factors-lemm
 - a, 79
 - a-off, 85
 - a-prime, 79
 - a-prime-off, 85
- itimes-list-eval\$-factors-off, 86
- itimes-list-eval\$-fringe, 57
- itimes-list-eval\$-fringe-off, 64
- itimes-list-eval\$list-0, 61
- itimes-list-eval\$list-0-off, 66
- itimes-list-off, 107
- itimes-list-off1, 1159
- itimes-list-on, 1070
- itimes-off, 37
- itimes-tree, 55, 58–60
- itimes-tree-ineg, 66–68
- itimes-tree-ineg-off, 68
- itimes-tree-no-fix-int, 60, 61
- itimes-tree-no-fix-int-off, 60
- itimes-tree-off, 55
- itimes-tree-rec, 54, 55, 60, 66, 67
- itimes-tree-rec-off, 107
- itimes-tree-rec-off1, 1159
- itimes-tree-rec-on, 1070
- itimes-zero1, 49
- itimes-zero1-off, 49
- itimes-zero2, 49
- itimes-zero2-off, 49
- izerop, 35, 1200, 1202, 1214
- izerop-eval-of-member-implies-iti
 - mes-list-0, 58
 - mes-list-0-off, 65
- izerop-ilessp-0-relationship, 80
- izerop-ilessp-0-relationship-of
 - f, 82
 - f1, 86

- izerop-inc-v-nzerop, 1200
- izerop-off, 116
- izerop-off1, 1168
- izerop-on, 1079

- label-error, 252, 278, 279, 283, 357, 358, 362–364, 368, 369
- label-error-off, 252
- last-cdr, 245, 277, 281, 286, 360, 361, 366, 367
- last-cdr-off, 245
- length, 136–149, 151, 152, 157–165, 167–170, 172–191, 202–205, 242, 253, 254, 257, 258, 277, 279, 283–285, 287, 289, 290, 293, 294, 305, 317, 355, 357–364, 366, 369, 372, 376, 381, 382, 420–422, 424, 426, 429, 430, 432, 433, 435, 443–446, 448, 452, 456, 457, 459, 461, 462, 464–469, 473, 475, 478, 481–486, 494, 495, 497, 501, 503, 505–508, 510, 511, 516, 517, 523, 526, 528–536, 541, 547–549, 558, 562–567, 576, 578, 587, 589–592, 603, 604, 670, 673, 678, 689, 690, 701–704, 714, 715, 717–719, 722, 733–735, 739–741, 744–746, 748, 749, 751, 753, 755, 759, 765, 776, 779, 780, 783, 792, 795, 800, 801, 805–810, 814, 816–819, 821, 823, 825, 827, 829, 831–833, 835, 837, 839, 841, 843, 845, 847, 849, 852, 854, 856, 858, 860, 862–864, 866, 867, 869–871, 874–876, 878, 880–883, 885–890, 892, 894, 896, 898, 900, 903, 905, 908, 910, 912, 915–918, 925, 926, 930, 954, 971, 974, 975, 1007, 1009–1011, 1014, 1016–1021, 1024, 1096, 1098–1106, 1188, 1190–1194, 1196–1198, 1200, 1201, 1204–1216, 1218, 1219
- length-1, 137
- length-append, 137
- length-bottom, 137
- length-cdr, 137
- length-cdr-lemmas, 137
- length-collect-value, 167
- length-cons, 137
- length-core-alu, 531
- length-cvzbv-adder, 485
- length-cvzbv-subtractor, 485
- length-dual-eval-0, 242
- length-dual-eval-2, 242
- length-firstn, 139
- length-firstn1, 139
- length-firstn2, 140
- length-fv-if, 203
- length-if, 137
- length-if*, 137
- length-indices, 149
- length-int-to-v, 1200
- length-make-list, 145
- length-mpg, 494
- length-nat-to-v, 160
- length-nlistp, 137
- length-of-v-adder, 163
- length-of-v-adder-output, 163
- length-of-v-subtractor-output, 163
- length-off, 136
- length-read-fn, 874
- length-reset-sequence-chip-1, 1024
- length-reset-sequence-chip-1-of f, 1027
- length-reset-sequence-chip-1-on, 1116
- length-restn, 140
- length-rev1, 138
- length-reverse, 138
- length-shift-or-buf, 526
- length-sign-extend, 162
- length-simulate, 1010
- length-simulate-off, 1033
- length-simulate-on, 1122
- length-subrange, 144

- length-sum, 444
- length-v-alu, 485
- length-v-alu-1, 485
- length-v-and, 158
- length-v-asr, 158
- length-v-buf, 158
- length-v-if, 158
- length-v-lsr, 158
- length-v-not, 158
- length-v-or, 158
- length-v-ror, 158
- length-v-shift-right, 158
- length-v-shift-right-names, 523
- length-v-threefix, 164
- length-v-xor, 158
- leq-lessp-difference, 135
- leq-log-log, 32
- leq-log-log-off, 118
- leq-log-log-off1, 1170
- leq-log-log-on, 1082
- leq-quotient, 28
- leq-quotient-off, 121
- leq-quotient-off1, 1172
- leq-quotient-on, 1084
- lessp-1-exp-2, 1200
- lessp-1-plus-x-x, 1197
- lessp-1-times, 15
- lessp-1-times-off, 126
- lessp-1-times-off1, 1177
- lessp-1-times-on, 1089
- lessp-a-plus-a-a, 1197
- lessp-count-listp-cdr, 43
- lessp-count-listp-cdr-off, 63
- lessp-count-listp-cdr-off1, 83
- lessp-count-listp-cdr-off2, 84
- lessp-count-listp-cdr-on, 83
- lessp-count-listp-cdr-on1, 84
- lessp-difference-cancellation, 12
- lessp-difference-cancellation-off, 12
- lessp-difference-plus-arg1, 48
- lessp-difference-plus-arg1-commuted, 48
- lessp-difference-plus-arg1-commuted-off, 63
- lessp-difference-plus-arg1-off, 63
- lessp-difference=0, 134
- lessp-gcd, 33
- lessp-gcd-off, 117
- lessp-gcd-off1, 1169
- lessp-gcd-on, 1081
- lessp-plus-fact, 24
- lessp-plus-fact-off, 24
- lessp-plus-times-proof, 15
- lessp-plus-times-proof-off, 126
- lessp-plus-times-proof-off1, 1177
- lessp-plus-times-proof-on, 1089
- lessp-plus-times1, 15
- lessp-plus-times1-off, 126
- lessp-plus-times1-off1, 1177
- lessp-plus-times1-on, 1089
- lessp-plus-times2, 15
- lessp-plus-times2-off, 126
- lessp-plus-times2-off1, 1177
- lessp-plus-times2-on, 1089
- lessp-quotient, 28
- lessp-quotient-2, 1103
- lessp-quotient-2-off, 1110
- lessp-quotient-off, 120
- lessp-quotient-off1, 1172
- lessp-quotient-on, 1083
- lessp-quotient-test, 1218
- lessp-quotient-test-off, 1218
- lessp-remainder, 21
- lessp-remainder-off, 125
- lessp-remainder-off1, 1176
- lessp-remainder-on, 1088
- lessp-remainder-theorem, 1198
- lessp-sub1-plus-a-b-b, 1193
- lessp-sub1-x-x, 135
- lessp-sub1-x-y-crock, 134
- lessp-times-arg1, 17
- lessp-times-arg1-off, 125
- lessp-times-arg1-off1, 1177
- lessp-times-arg1-on, 1088
- lessp-times-cancellation-proof, 14
- lessp-times-cancellation-proof-off, 126
- lessp-times-cancellation-proof-off1, 1177

- on, 1089
- lessp-times-cancellation1, 14
- lessp-times-cancellation1-off, 14
- lessp-times1, 14
- lessp-times1-off, 126
- lessp-times1-off1, 1178
- lessp-times1-on, 1089
- lessp-times1-proof, 14
- lessp-times1-proof-off, 126
- lessp-times1-proof-off1, 1178
- lessp-times1-proof-on, 1090
- lessp-times2, 14
- lessp-times2-off, 126
- lessp-times2-off1, 1178
- lessp-times2-on, 1089
- lessp-times2-proof, 14
- lessp-times2-proof-off, 126
- lessp-times2-proof-off1, 1178
- lessp-times2-proof-on, 1089
- lessp-times3, 14
- lessp-times3-off, 126
- lessp-times3-off1, 1177
- lessp-times3-on, 1089
- lessp-times3-proof1, 14
- lessp-times3-proof1-off, 126
- lessp-times3-proof1-off1, 1178
- lessp-times3-proof1-on, 1089
- lessp-times3-proof2, 14
- lessp-times3-proof2-off, 126
- lessp-times3-proof2-off1, 1177
- lessp-times3-proof2-on, 1089
- lessp-unbind-list-count, 248
- lessp-v-to-nat-exp, 1099
- lessp-v-to-nat-exp-2-length-a, 1193
- lessp-v-to-nat-exp-2-length-a\$
 - linear, 1193
- lessp-v-to-nat-exp-off, 1113
- lessp-v-to-nat-exp-rewrite, 1099
- lessp-v-to-nat-exp-rewrite-off, 1113
- lessp-v-to-nat-exp-with-exp-ope
 - ned, 1102
 - ned-off, 1111
- lessp-x-1, 135
- lessp-x-x, 134
- letterp, 284, 285
- letterp-off, 284
- lisp-netlist, 435, 1026
- list-32-nth-collapse, 421
- list-32-nth-collapse-off, 422
- list-as-collected-nth, 147, 148, 420, 421, 776
- list-as-collected-nth-off, 147
- list-collect-value, 252, 253, 255, 257
- list-collect-value-off, 253
- list-duplicates, 245, 252
- list-duplicates-off, 246
- list-elim-4, 147
- list-elim-4-off, 147
- list-lessp, 248, 249
- list-lessp-off, 249
- list-list-sort, 250, 258, 381
- list-list-sort-off, 250
- list-rewrite-4, 147
- list-rewrite-4-off, 147
- list-sort, 250
- list-sort-off, 250
- list-to-mem, 558, 559
- list-to-mem1, 558
- list-to-mem2, 558
- list-to-mem3, 558
- list-to-tree-mem, 558
- list-union-values, 247, 380
- list-union-values-off, 247
- listify, 246, 256, 257, 262, 334, 357
- listify-off, 246
- listp-bagint-with-singleton-imp
 - lies-member, 61
 - lies-member-off, 66
- listp-bagint-with-singleton-mem
 - ber, 62
 - ber-off, 66
- listp-cdr-factors-implies-intege
 - rp, 84
 - rp-off, 87
- listp-collect-value, 167
- listp-delete, 3
- listp-delete-off, 3
- listp-eval\$, 9

- listp-eval\$-off, 9
- listp-implies-not-fourp, 1018
- listp-implies-not-fourp-off, 1029
- listp-implies-not-fourp-on, 1118
- listp-indices, 149
- listp-make-tree, 147
- listp-nthcdr, 143
- listp-subrange, 144
- literal-delay-to-range, 325, 326
- literal-delay-to-range-off, 325
- literal-net-drive, 310–312
- literal-net-drive-off, 311
- literal-state-type-ok, 368, 369
- literal-state-type-ok-off, 368
- lnfix, 148, 149
- lnk, 551
- lnk+, 553
- lnk-, 552
- loaded-delay, 323, 324
- loaded-delay-off, 324
- loaded-delays, 324, 345
- loaded-delays-off, 324
- loading-plus, 307
- loading-plus-off, 307
- loadings-and-drives-off, 390
- loadings-and-drives-simple, 272–276
- loadings-and-drives-simple-off, 275
- loadings-error, 313, 349
- loadings-error-off, 313
- local-delay, 321–323
- local-delay-count-help, 321
- local-delay-count-help-0, 321
- local-delay-count-help-0-off, 321
- local-delay-count-help-off, 322
- local-delay-off, 322
- local-drive, 308, 309
- local-drive-off, 308
- log, 30–32
- log-0, 31
- log-0-off, 119
- log-0-off1, 1170
- log-0-on, 1082
- log-1, 31
- log-1-off, 119
- log-1-off1, 1170
- log-1-on, 1082
- log-exp, 32
- log-exp-off, 118
- log-exp-off1, 1170
- log-exp-on, 1081
- log-off, 120
- log-off1, 1171
- log-on, 1083
- log-quotient, 32
- log-quotient-exp, 32
- log-quotient-exp-off, 118
- log-quotient-exp-off1, 1170
- log-quotient-exp-on, 1081
- log-quotient-off, 118
- log-quotient-off1, 1170
- log-quotient-on, 1082
- log-quotient-times, 32
- log-quotient-times-off, 118
- log-quotient-times-off1, 1170
- log-quotient-times-on, 1081
- log-quotient-times-proof, 32
- log-quotient-times-proof-off, 118
- log-quotient-times-proof-off1, 1170
- log-quotient-times-proof-on, 1081
- log-times, 32
- log-times-exp, 32
- log-times-exp-off, 118
- log-times-exp-off1, 1170
- log-times-exp-on, 1081
- log-times-exp-proof, 32
- log-times-exp-proof-off, 118
- log-times-exp-proof-off1, 1170
- log-times-exp-proof-on, 1081
- log-times-off, 118
- log-times-off1, 1170
- log-times-on, 1081
- log-times-proof, 32
- log-times-proof-off, 118
- log-times-proof-off1, 1170
- log-times-proof-on, 1081
- log2, 550, 558
- logs, 33

- lookup-module, 237–242, 247, 248, 253–261, 263, 265, 266, 268, 269, 271, 273, 274, 278, 279, 281, 291, 378–382, 385, 419, 429–431, 433, 434, 436–439, 441, 445, 450, 453, 454, 458, 460, 462, 463, 469, 471, 474, 476, 480, 487, 489, 492, 495, 497, 499, 500, 502, 513, 514, 518, 522, 524, 528, 540, 560, 562, 564–570, 578, 582, 672, 688, 713, 721, 738, 744, 748, 750, 752, 754, 757, 775, 790, 799, 954, 955, 976, 998–1001
- lookup-module-in-delete-module, 248
- lookup-module-in-delete-module-off, 248
- lookup-module-in-unbind-list, 248
- lookup-module-in-unbind-list-of f, 248
- lookup-module-off, 237
- lsi-bad-names, 286
- lsi-bad-names-off, 286
- lsi-function-name, 286, 290
- lsi-function-name-off, 287
- lsi-keywords, 285
- lsi-keywords-off, 285
- lsi-module-inputs-ok, 292, 293
- lsi-module-inputs-ok-off, 292
- lsi-module-name-ok, 291, 293
- lsi-module-name-ok-off, 292
- lsi-module-occurrences-check, 292–294
- lsi-module-occurrences-check-of f, 293
- lsi-module-outputs-ok, 292, 293
- lsi-module-outputs-ok-off, 292
- lsi-module-syntax-check, 293, 294
- lsi-module-syntax-check-off, 294
- lsi-name-list-errors, 286
- lsi-name-list-errors-off, 286
- lsi-name-list-ok, 286, 289, 292
- lsi-name-list-ok-off, 286
- lsi-name-ok, 285, 286, 288, 291, 365
- lsi-name-ok-off, 286
- lsi-netlist-syntax-check, 294
- lsi-netlist-syntax-check-off, 294
- lsi-netlist-syntax-ok, 294, 354
- lsi-netlist-syntax-ok-off, 294
- lsi-occ-body-syntax-check, 290, 291, 293
- lsi-occ-body-syntax-check-off, 291
- lsi-occ-inputs-ok, 289, 290
- lsi-occ-inputs-ok-off, 289
- lsi-occ-name-ok, 288, 289
- lsi-occ-name-ok-off, 289
- lsi-occ-outputs-ok, 289, 290
- lsi-occ-outputs-ok-off, 289
- lsi-occ-syntax-ok, 289, 291
- lsi-occ-syntax-ok-off, 290
- lsi-top-level-predicate, 354
- lsi-top-level-predicate-off, 354
- m-states-list, 248, 255, 265, 282, 283, 336, 373
- m-states-list-off, 248
- m1*, 438
- m1\$netlist, 438, 439
- m1\$state, 439
- m1\$state-off, 439
- m1\$value, 439
- m1\$value-off, 439
- m1&, 438, 439
- m1&-off, 439
- m2*, 439
- m2\$netlist, 439
- m2\$state, 440
- m2\$state-off, 440
- m2\$value, 439
- m2\$value-off, 440
- m2&, 439, 440
- m2&-off, 439
- ma-lessp, 307, 314
- ma-lessp-off, 307
- ma-to-pf, 306, 310, 311
- ma-to-pf-off, 306
- ma-to-std-drive, 306, 314
- ma-to-std-drive-off, 306

machine-state-invariant, 779, 780, 783, 791, 795, 800, 1021
 machine-state-invariant-implies
 -s-approx, 1020
 -s-approx-lemma, 1019
 -s-approx-lemma-2, 1019
 -s-approx-lemma-2-off, 1028
 -s-approx-lemma-2-on, 1117
 -s-approx-lemma-3, 1019
 -s-approx-lemma-3-off, 1028
 -s-approx-lemma-3-on, 1117
 -s-approx-lemma-off, 1028
 -s-approx-lemma-on, 1117
 -s-approx-off, 1028
 -s-approx-on, 1117
 macrocycle-invariant, 917–922, 924–927, 1014
 macrocycle-invariant*, 919, 921
 macrocycle-invariant*-off, 919
 macrocycle-invariant*=macrocycle-invariant, 919
 macrocycle-invariant-is-invariant, 921
 nt\$help, 919
 macrocycle-invariant==>chip-system-invariant, 919
 ystem-invariant\$help, 918
 macrocycle-invariant==>pc-reg, 924
 make-cancel-ineg-terms-equality, 88, 89
 -off, 102
 -off1, 1154
 -on, 1065
 make-cancel-ineg-terms-inequality, 92, 93
 y-off, 101
 y-off1, 1153
 y-on, 1064
 make-cancel-iplus-inequality-1, 47, 48
 make-cancel-iplus-inequality-1-off, 49
 make-cancel-iplus-inequality-simplifier, 48
 mplifier-off, 64
 make-cancel-itimes-equality, 55, 56, 58, 68, 69
 make-cancel-itimes-equality-off, 58
 make-cancel-itimes-inequality, 60, 61, 80
 make-cancel-itimes-inequality-off, 61
 make-delay, 318, 323, 331
 make-delay-0, 318, 322
 make-delay-0-off, 318
 make-delay-off, 319
 make-delay-or, 326, 328, 333
 make-delay-or-off, 326
 make-drive-min, 315, 316
 make-drive-min-off, 316
 make-list, 145, 146, 151, 159, 161, 162, 164, 165, 168, 174–192, 203–205, 265, 505, 536, 558, 559, 568–570, 591, 592, 608, 610, 612, 614, 616, 617, 619, 623, 625, 627, 629, 631, 633, 635, 637, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 666, 668, 701, 717, 746–748, 758, 759, 762–764, 800, 802–805, 812–814, 925–927, 954, 971, 975, 993, 1007, 1011–1015, 1017, 1022–1026
 make-list-append, 145
 make-list-append-tree-crock, 146
 make-list-append-tree-crock-off, 146
 make-list-crock-for-f\$cv=cv, 701
 make-list-crock-for-f\$cv=cv-of-f, 701
 make-list-off, 145
 make-output-delay, 331–333
 make-output-delay-off, 331
 make-pairs, 558
 make-ram-state, 265
 make-ram-state-off, 265
 make-range, 319, 320, 322, 324, 325, 329
 make-range-off, 319

make-tree, 147, 459–462, 474, 475,
 712–714, 736, 738, 739, 747,
 748, 761, 763, 776, 781, 782,
 784, 792–794, 796, 801, 802,
 804, 811, 813, 838, 850, 851,
 859, 869, 873, 901, 902, 904,
 907, 909, 911, 913
 make-tree-off, 147
 map, 305, 310–312, 314, 316, 363
 map-down, 927, 928, 1021, 1022
 map-down-inverts-map-up, 1022
 map-down-inverts-map-up-off, 1024
 map-down-inverts-map-up-off1, 1115
 map-down-inverts-map-up-on, 1063
 map-down-relation, 925, 926
 map-down-relation-lemma, 925
 map-down-relation-off, 925
 map-off, 305
 map-up, 921, 922, 924–928, 1014,
 1021–1024, 1026
 map-up-1-input, 808, 809
 map-up-inputs, 808, 809, 925, 927,
 928, 1014, 1021–1026
 map-up-inputs-off, 808
 map-up-inverts-map-down, 927
 map-up-off, 922
 mark-io-out, 296
 mark-io-out-off, 296
 mark-io-outs, 296, 343, 344, 351, 380,
 383, 384
 mark-io-outs-0, 296
 mark-io-outs-0-off, 296
 mark-io-outs-off, 296
 math-theory, 100
 max-hierarchical-length-and-name, 287,
 288
 -off, 287
 max-occ-hierarchical-name, 287, 291
 max-occ-hierarchical-name-off, 288
 mem-32x32-args-crock, 421
 mem-32x32-args-crock-off, 421
 mem-32x32\$netlist, 404, 800
 mem-32x32\$state, 415
 mem-32x32\$state-off, 418
 mem-32x32\$structured-state, 426
 mem-32x32\$structured-state-1, 424
 mem-32x32\$structured-state-1-o
 ff, 426
 mem-32x32\$structured-state-off, 426
 mem-32x32\$structured-value, 424
 mem-32x32\$structured-value-1, 422
 mem-32x32\$structured-value-1-o
 ff, 424
 mem-32x32\$structured-value-off, 424
 mem-32x32\$value, 405
 mem-32x32\$value-off, 408
 mem-32x32&, 404, 405, 415, 422,
 424, 426, 800
 mem-32x32&-off, 404
 mem-map-down, 927
 mem-map-up, 921
 mem-state, 183, 418
 mem-state-off, 184
 mem-to-list, 559
 mem-value, 183, 408
 mem-value-off, 183
 mem-width, 929, 930, 974–976, 1020
 mem-width-linear-facts, 976
 mem-width-linear-facts-off, 997
 mem-width-linear-facts-off1, 1151
 mem-width-linear-facts-on, 1062
 mem-width-non-zero, 975
 mem-width-non-zero-off, 997
 mem-width-non-zero-off1, 1151
 mem-width-non-zero-on, 1062
 mem-width-off, 973
 mem-width-off1, 1056
 mem-width-on, 996
 mem-width-on1, 1146
 member-0-eval\$list, 79
 member-0-eval\$list-off, 104
 member-0-eval\$list-off1, 1156
 member-0-eval\$list-on, 1068
 member-0-itimes-factors-yields-
 0-ilessp-consequence-1, 82
 0-ilessp-consequence-1-off, 104
 0-ilessp-consequence-1-off1, 1156
 0-ilessp-consequence-1-on, 1067

- 0-ilessp-consequence-2, 82
- 0-ilessp-consequence-2-off, 103
- 0-ilessp-consequence-2-off1, 1156
- 0-ilessp-consequence-2-on, 1067
- 0-off, 104
- 0-off1, 1156
- 0-on, 1067
- 0, 81
- member-append, 57
- member-append-off, 106
- member-append-off1, 1158
- member-append-on, 1070
- member-bagdiff, 4
- member-bagdiff-off, 130
- member-bagdiff-off1, 1181
- member-bagdiff-on, 1093
- member-bagint, 5
- member-bagint-off, 130
- member-bagint-off1, 1181
- member-bagint-on, 1093
- member-delete, 4
- member-delete-implies-membership, 4
 - off, 130
 - off1, 1182
 - on, 1093
- member-delete-off, 130
- member-delete-off1, 1182
- member-delete-on, 1094
- member-implies-numberp, 9
- member-implies-numberp-off, 9
- member-implies-plus-tree-greate
 - reqp, 8
 - reqp-off, 8
- member-indices, 149
- member-izerop-itimes-fringe, 57
- member-izerop-itimes-fringe-off, 64
- member-non-list, 3
- member-non-list-off, 130
- member-non-list-off1, 1182
- member-non-list-on, 1094
- member==>position, 136
- membership-of-0-implies-itimes-
 - list-is-0, 79
 - list-is-0-off, 104
 - list-is-0-off1, 1156
 - list-is-0-on, 1068
- memory-okp, 169, 170, 172–174, 734, 735, 739, 810, 814, 816, 817, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 852, 854, 856, 858, 860, 862, 863, 865, 867, 869, 871, 875, 876, 878, 879, 881, 883, 885–887, 889–891, 894, 896, 898, 899, 903, 905, 908, 910, 912, 917, 926, 1021, 1022, 1024
- memory-okp-after-write-mem, 172
- memory-okp-after-write-mem1, 172
- memory-okp-if, 172
- memory-okp-off, 170
- memory-okp==>memory-properp, 174
- memory-properp, 169, 171–174, 184, 740, 779, 800, 807, 1021
- memory-properp-after-write-mem, 172
- memory-properp-after-write-mem1, 172
- memory-properp-constant-ram, 172
- memory-properp-dual-port-ram-state-crock, 807
- memory-properp-if, 171
- memory-properp-off, 169
- memory-state-invariant, 800, 806, 807
- memory-v-fourp, 1017, 1018, 1020, 1021
- memory-v-fourp-implies-good-s, 1020
- memory-v-fourp-implies-good-s-off, 1028
- n, 1117
- memory-v-fourp-implies-memory-properp, 1021
- roperp-off, 1027
- roperp-on, 1116
- memory-v-fourp-off, 1029
- memory-v-fourp-on, 1118
- memory-value, 182–184, 424, 762–764, 800, 802–805, 812–814
- memory-value-off, 183

- merge-input-delays, 324, 325
- merge-input-delays-off, 324
- mergeable-input-delay-p, 325
- mergeable-input-delay-p-off, 325
- microcycles, 916, 917, 920, 921, 923, 924
- middle=high, 924
- middle=high\$help, 922
- min, 244, 324, 325
- min-off, 244
- min-token, 249, 250
- min-token-help, 249
- min-token-help-lemma, 249
- min-token-help-off, 249
- min-token-lemma, 249
- min-token-off, 249
- minus-ineg, 92
- minus-ineg-off, 94
- mode-a, 541, 546, 548, 622, 624, 625, 632, 649, 670, 673, 716, 724, 828, 829, 836, 838, 848, 868, 893, 894, 900–902, 904, 909, 915
- mode-a-off, 542
- mode-b, 541, 545, 546, 548, 631, 634, 635, 639, 641, 653, 670, 673, 716, 724, 725, 727, 730, 828–830, 834, 842, 846, 850, 851, 857, 859, 869, 872, 893–895, 898, 899, 902, 904–908, 910, 911, 913–915
- mode-b-off, 542
- module-annotation, 238, 254, 284, 294
- module-annotation-ok, 283, 284, 294
- module-annotation-ok-off, 283
- module-data, 351
- module-data-off, 351
- module-database, 351, 353, 379, 380, 383–385
- module-database-off, 351
- module-form-ok, 281, 283, 293
- module-form-ok-off, 281
- module-inputs, 238–241, 253–258, 260–262, 273–276, 279, 283, 293, 351, 381, 382, 999–1001
- module-inputs-ok, 282, 283
- module-inputs-ok-off, 282
- module-name, 238, 254, 257, 262–264, 267, 275, 276, 283, 284, 293, 294, 351
- module-name-ok, 281, 283
- module-name-ok-off, 282
- module-occurrences, 238–241, 255, 256, 260, 263, 265, 266, 273, 274, 283, 293, 351, 378, 436, 998–1001
- module-occurrences-ok, 282, 284
- module-occurrences-ok-off, 283
- module-outputs, 238–240, 242, 253, 255, 256, 260, 273–276, 279, 283, 284, 293, 294, 351
- module-outputs-ok, 282, 283
- module-outputs-ok-off, 282
- module-prop-error, 348, 349
- module-prop-error-off, 349
- module-statenames, 238–242, 255, 265, 266, 284, 293, 294, 351, 999–1001
- module-statenames-ok, 283, 284, 294
- module-statenames-ok-off, 283
- module-syntax-errors, 283, 284
- module-syntax-errors-off, 284
- module-syntax-ok, 284
- module-syntax-ok-off, 284
- module-syntax-simple-okp, 254, 255
- module-syntax-simple-okp-off, 255
- monotonicity-lemmas, 938
- monotonicity-property, 930–933, 939–952, 972, 993, 998, 1003
- monotonicity-property-consequen
 - ce-0, 931
 - ce-0-off, 1055
 - ce-0-on, 1145
 - ce-1, 931
 - ce-1-off, 1055
 - ce-1-on, 1145
 - ce-2, 932

- ce-2-off, 1055
- ce-2-on, 1145
- ce-3, 932
- ce-3-off, 1055
- ce-3-on, 1145
- monotonicity-property-off, 931
- monotonicity-property-off1, 1146
- monotonicity-property-on, 1057
- monotonicity-property-opener-0, 932
- monotonicity-property-opener-0-
off, 933
- off1, 1146
- on, 1057
- monotonicity-property-opener-1, 932
- monotonicity-property-opener-1-
off, 933
- off1, 1146
- on, 1057
- monotonicity-property-opener-2, 932
- monotonicity-property-opener-2-
off, 933
- off1, 1146
- on, 1057
- monotonicity-property-opener-3, 933
- monotonicity-property-opener-3-
off, 933
- off1, 1147
- on, 1057
- mpg, 494, 495, 532–536, 608, 610,
612, 614, 616, 617, 619, 621,
623, 625, 627, 629, 631, 633,
635, 637, 639, 642, 643, 645,
647, 649, 651, 653–655, 657,
659, 661, 663, 665, 667, 669,
838, 850, 851, 859, 868, 872,
901, 902, 904, 907, 909, 911,
913
- mpg*, 495
- mpg-if-op-code, 494
- mpg-off, 494
- mpg-zero, 495
- mpg-zero-off, 495
- mpg\$netlist, 495, 714
- mpg\$value, 495
- mpg\$value-off, 496
- mpg\$value-zero, 496
- mpg\$value-zero-off, 496
- mpg&, 495, 496, 714
- mpg&-off, 495
- multiplication, 21
- n, 482, 483, 544, 744, 1188
- n-flag, 543, 544, 559, 744, 1186
- n-flag-off, 543
- n-off, 482
- n-set, 542, 544, 591, 742–744, 1189,
1190, 1202, 1203, 1205–1213,
1215, 1216
- n-set-off, 542
- n-v-alu, 1188
- n_decode, 594
- n_decode-off, 594
- n_fetch0, 593
- n_fetch0-off, 593
- n_fetch1, 593
- n_fetch1-off, 593
- n_fetch2, 593
- n_fetch2-off, 593
- n_fetch3, 594
- n_fetch3-off, 594
- n_hold0, 600
- n_hold0-off, 600
- n_hold1, 600
- n_hold1-off, 600
- n_reada0, 595
- n_reada0-off, 595
- n_reada1, 595
- n_reada1-off, 595
- n_reada2, 596
- n_reada2-off, 596
- n_reada3, 596
- n_reada3-off, 596
- n_readb0, 596
- n_readb0-off, 596
- n_readb1, 597
- n_readb1-off, 597
- n_readb2, 597
- n_readb2-off, 597

n_readb3, 597
n_readb3-off, 597
n_rega, 594
n_rega-off, 594
n_regb, 594
n_regb-off, 595
n_reset0, 601
n_reset0-off, 601
n_reset1, 602
n_reset1-off, 602
n_reset2, 602
n_reset2-off, 602
n_sefa0, 599
n_sefa0-off, 599
n_sefa1, 599
n_sefa1-off, 599
n_sefb0, 599
n_sefb0-off, 599
n_sefb1, 600
n_sefb1-off, 600
n_update, 595
n_update-off, 595
n_v11010, 601
n_v11010-off, 601
n_v11011, 601
n_v11011-off, 601
n_v11111, 602
n_v11111-off, 602
n_write0, 598
n_write0-off, 598
n_write1, 598
n_write1-off, 598
n_write2, 598
n_write2-off, 598
n_write3, 598
n_write3-off, 599
name-length, 287, 288
name-length-off, 287
name-list-errors, 277, 283, 367
name-list-errors-off, 277
name-list-ok, 277, 278, 282, 360, 366,
374
name-list-ok-off, 277
name-list-simple-okp, 253–255
name-list-simple-okp-off, 253
name-okp, 276, 278, 281, 354, 358,
360, 367, 370, 375–377
name-okp-off, 276
name-simple-okp, 253, 254
name-simple-okp-off, 253
nat-lst-lst-induction, 1009
nat-lst-lst-induction-off, 1033
nat-lst-lst-induction-on, 1123
nat-to-v, 160, 163, 444, 465, 483,
507, 508, 547, 549, 559, 1097–
1102, 1186, 1193, 1197, 1200,
1216, 1218
nat-to-v-as-remainder, 1218
nat-to-v-of-v-to-nat, 1098
nat-to-v-of-v-to-nat*, 1218
nat-to-v-of-v-to-nat*-off, 1218
nat-to-v-of-v-to-nat-general, 1098
nat-to-v-of-v-to-nat-general-of
f, 1114
nat-to-v-of-v-to-nat-off, 1114
nat-to-v-off, 160
nat-to-v-plus-x-x, 1097
nat-to-v-plus-x-x-off, 1114
nat-to-v-remainder, 1098
nat-to-v-remainder-off, 1114
nat-to-v-zerop, 1098
nat-to-v-zerop-off, 1114
natural-state-theory, 603
naturals, 34
naturals-metas, 1096
negativep-idifference-on-numberp
s, 1197
net-drives, 271, 274
net-drives-off, 271
net-drives-simple, 312, 345
net-drives-simple-off, 312
net-error, 250
net-errorp, 250–252, 302, 311, 312,
335, 340, 349, 350, 352–354,
368, 370–372, 379–382, 385,
386
net-errors, 251
net-errors-off, 251

- net-min-args, 311, 312
- net-min-args-off, 311
- netlist-database, 353
- netlist-database-off, 353
- netlist-loadings-and-drives, 390
- netlist-loadings-and-drives-off, 390
- netlist-loadings-and-drives-simple, 275, 276
- netlist-loadings-and-drives-simple-off, 276
- netlist-properties, 353, 354, 379–386, 390
- netlist-properties-off, 353
- netlist-state-types, 386
- netlist-state-types-off, 386
- netlist-state-types-simple, 267, 268, 276
- netlist-state-types-simple-off, 267
- netlist-syntax-errors, 284
- netlist-syntax-errors-off, 284
- netlist-syntax-ok, 284, 354
- netlist-syntax-ok-off, 284
- netlist-syntax-simple-okp, 255, 276
- netlist-syntax-simple-okp-off, 255
- netlist-type-check-simple-okp, 264, 276
- netlist-type-check-simple-okp-off, 264
- netlist-type-table, 384
- netlist-type-table-off, 384
- netlist-type-table-simple, 262, 264, 265
- netlist-type-table-simple-off, 262
- netlist-types-acceptable-list, 263, 264
- netlist-types-acceptable-list-off, 264
- new-machine-state-invariant, 1017, 1019–1021
- new-machine-state-invariant-imp
 - lies-good-s, 1020
 - lies-good-s-off, 1028
 - lies-good-s-on, 1117
 - lies-machine-state-invariant, 1021
 - lies-machine-state-invariant-off, 1027
 - lies-machine-state-invariant-on, 1116
- new-machine-state-invariant-is-non-trivial, 1021
- new-machine-state-invariant-is-non-trivial-off, 1027
- new-machine-state-invariant-is-non-trivial-on, 1116
- new-machine-state-invariant-off, 1029
- new-machine-state-invariant-on, 1118
- next-cntl-state, 716, 717, 719, 723–733
- next-cntl-state*, 719, 721
- next-cntl-state-off, 716
- next-cntl-state\$decode, 724
- next-cntl-state\$fetch0, 722
- next-cntl-state\$fetch1, 723
- next-cntl-state\$fetch2, 723
- next-cntl-state\$fetch3, 723
- next-cntl-state\$hold0, 730
- next-cntl-state\$hold1, 731
- next-cntl-state\$netlist, 721, 776
- next-cntl-state\$reada0, 725
- next-cntl-state\$reada1, 726
- next-cntl-state\$reada2, 726
- next-cntl-state\$reada3, 726
- next-cntl-state\$readb0, 727
- next-cntl-state\$readb1, 727
- next-cntl-state\$readb2, 727
- next-cntl-state\$readb3, 728
- next-cntl-state\$rega, 724
- next-cntl-state\$regb, 725
- next-cntl-state\$reset0, 732
- next-cntl-state\$reset1, 732
- next-cntl-state\$reset2, 732
- next-cntl-state\$sefa0, 729
- next-cntl-state\$sefa1, 729
- next-cntl-state\$safb0, 730
- next-cntl-state\$safb1, 730
- next-cntl-state\$update, 725
- next-cntl-state\$vl1010, 731
- next-cntl-state\$vl1011, 731
- next-cntl-state\$vl1111, 732
- next-cntl-state\$value, 722
- next-cntl-state\$value-off, 722
- next-cntl-state\$write0, 728

next-cntl-state\$write1, 728
 next-cntl-state\$write2, 729
 next-cntl-state\$write3, 729
 next-cntl-state&, 721, 722, 776
 next-cntl-state&-off, 721
 next-memory-state, 175, 183, 191,
 426, 764, 805, 807, 814
 next-memory-state-off, 182
 next-memory-state-preserves-memory-invariant, 806
 next-memory-state\$induction, 184, 191
 next-state, 673, 674, 679, 689, 716
 next-state*, 679, 688
 next-state\$netlist, 688, 721
 next-state\$value, 690
 next-state\$value-off, 690
 next-state&, 688, 690, 721
 next-state&-off, 688
 nil-or-err, 251, 277, 280, 281, 284,
 286, 294, 357, 359–362, 364–
 367, 369, 375
 nil-or-err-off, 252
 nlistp-or-err, 251, 277, 280, 286, 335,
 338, 350, 357, 359, 361, 362,
 364–367, 369, 371, 372, 375
 nlistp-or-err-off, 251
 no-duplicates-disjoint-firstn-restn, 141
 no-duplicates-in-indices, 149
 no-duplicates-or-err, 252, 277, 281,
 286, 338, 360, 366
 no-duplicates-or-err-off, 252
 no-holds-reset-or-test, 928
 not*, 132, 133, 624, 634, 673, 716,
 724, 828, 838, 850, 851, 893,
 900, 901, 904, 915
 not*-off, 132
 not-duplicates?-firstn, 140
 not-duplicates?-restn, 141
 not-equal-plus-minus-0, 1197
 not-equal-tree-size-tree-0, 146
 not-integerp-implies-not-equal-plus-minus-off, 46
 lus-off, 87
 not-integerp-implies-not-equal-iti-mes, 57
 mes-off, 87
 not-integerp-minus-zero, 1199
 not-lessp-difference, 134
 not-lessp-quotient, 1099
 not-lessp-quotient-off, 1113
 not-litatom==>not-primp, 238
 not-member-firstn, 140
 not-member-restn, 141
 not-member-subrange, 144
 not-set-some-flags-make-list-4-f, 591
 not-set-some-flags-update-flags, 591
 not-v-iff-v-addr1-v-addr2-read-mem1-write-mem1, 173
 not-v-negp-nat-to-v-0, 1200
 not-v-nzerop-all-f, 161
 not-v-nzerop-as-and-crock, 161
 not-v-nzerop-v-xor-x-x, 161
 nth, 142, 143, 147–149, 152, 157,
 159–162, 167, 174, 421–426,
 435, 459, 483, 484, 526, 529–
 532, 541–545, 549, 589–591,
 604–606, 674–678, 748, 759,
 760, 784, 796, 927, 953, 971,
 973, 993, 1009–1011, 1016,
 1020, 1021, 1024, 1096, 1101,
 1190, 1191, 1217
 nth-append, 143
 nth-append-too, 143
 nth-if, 142
 nth-if-off, 142
 nth-indices, 149
 nth-indices-off, 149
 nth-length-v, 1096
 nth-length-v-off, 1115
 nth-off, 142
 nth-restn, 142
 nth-restn-off, 142
 nthcdr, 143, 151, 765, 810, 916, 917
 nthcdr-off, 143
 number-to-digit, 435
 number-to-list, 435

- number-to-list1, 435
- numberp-eval\$-bridge, 8
- numberp-eval\$-bridge-off, 8
- numberp-eval\$-plus, 7
- numberp-eval\$-plus-off, 7
- numberp-eval\$-plus-tree, 7
- numberp-eval\$-plus-tree-off, 8
- numberp-eval\$-times, 16
- numberp-eval\$-times-off, 16
- numberp-eval\$-times-tree, 17
- numberp-eval\$-times-tree-off, 17
- numberp-is-integerp, 95
- numberp-is-integerp-off, 96
- numeric-drives, 314–316
- numeric-drives-off, 314

- occ-annotation, 238, 253, 279, 290
- occ-annotation-ok, 279, 290
- occ-annotation-ok-off, 279
- occ-arg-length-error, 277–279
- occ-arg-length-error-off, 277
- occ-body-syntax-errors, 281, 283
- occ-body-syntax-errors-off, 281
- occ-body-syntax-simple-okp, 254, 255
- occ-body-syntax-simple-okp-off, 254
- occ-form-ok, 277, 279, 289
- occ-form-ok-off, 277
- occ-function, 238–243, 253, 256, 257, 261, 263, 266, 274, 275, 279, 287, 288, 290, 344, 378, 436, 998–1001
- occ-function-ok, 278, 279, 290
- occ-function-ok-off, 279
- occ-inputs, 238–243, 253, 254, 256, 257, 261, 262, 275, 279, 280, 287, 288, 290, 343, 344, 999–1001
- occ-inputs-ok, 278, 279
- occ-inputs-ok-off, 278
- occ-name, 238–243, 253, 254, 266, 279, 280, 287–290, 343, 999–1001
- occ-name-ok, 277, 279
- occ-name-ok-off, 278

- occ-outputs, 238, 239, 241, 243, 253, 254, 257, 261, 275, 279, 280, 287–290, 343, 344, 426, 427, 999, 1000
- occ-outputs-ok, 278, 279
- occ-outputs-ok-off, 278
- occ-syntax-errors, 279, 280
- occ-syntax-errors-off, 279
- occ-syntax-ok, 279, 281
- occ-syntax-ok-off, 280
- occ-syntax-simple-okp, 253, 254
- occ-syntax-simple-okp-off, 254
- occurrence-data, 346, 351, 379, 380, 383–385
- occurrence-data-off, 346
- occurrences, 3, 4
- occurrences-bagdiff, 4
- occurrences-bagdiff-off, 130
- occurrences-bagdiff-off1, 1181
- occurrences-bagdiff-on, 1093
- occurrences-bagint, 4
- occurrences-bagint-off, 130
- occurrences-bagint-off1, 1181
- occurrences-bagint-on, 1093
- occurrences-delete, 4
- occurrences-delete-off, 130
- occurrences-delete-off1, 1181
- occurrences-delete-on, 1093
- occurrences-off, 131
- occurrences-off1, 1182
- occurrences-on, 1094
- ok-in-type, 302, 303
- ok-in-type-off, 302
- ok-input-types, 376, 377
- ok-input-types-off, 376
- ok-name-list, 375–377
- ok-name-list-off, 375
- ok-netlistp, 998, 1003, 1008–1011, 1016
- ok-netlistp-off, 1041
- ok-netlistp-on, 1130
- ok-netlistp-reduction, 1003
- ok-netlistp-reduction-off, 1039
- ok-netlistp-reduction-on, 1128

- ok-netlistp-reduction-rewrite, 1003
- ok-netlistp-reduction-rewrite-o
 - ff, 1039
 - n, 1128
- ok-out-depends, 375–377
- ok-out-depends-off, 376
- ok-out-type, 302, 304
- ok-out-type-off, 302
- ok-output-types, 376, 377
- ok-output-types-off, 376
- ok-states, 375, 377
- ok-states-off, 375
- one-pstate-type-ok, 369
- one-pstate-type-ok-off, 369
- op-add, 1188, 1192, 1207, 1208, 1218, 1219
- op-adc, 1187, 1192, 1206, 1207, 1218, 1219
- op-and, 1188, 1217
- op-asr, 1188, 1214–1217, 1219
- op-code, 542, 545, 548, 549, 610, 614–617, 619, 621, 623, 627, 629, 631, 633, 637, 639, 643, 645, 647, 651, 655, 657, 659, 661, 668–670, 673, 716, 718, 721, 724, 725, 727, 828, 831, 834, 835, 842, 857, 893, 895, 897–899, 902, 907, 908, 914, 915
- op-code-off, 542
- op-dec, 1188, 1193, 1194, 1210, 1211, 1218, 1219
- op-inc, 1187, 1192, 1204, 1205, 1218, 1219
- op-lsr, 1188, 1191, 1196, 1217, 1218
- op-m15, 1188, 1191, 1192, 1201–1204, 1217
- op-move, 1187, 1191, 1200–1204, 1217
- op-neg, 1188, 1208, 1209, 1219
- op-not, 1188, 1217
- op-or, 1188, 1217
- op-ror, 1188, 1190, 1217
- op-sub, 1188–1190, 1195, 1196, 1213, 1218, 1219
- op-subb, 1188, 1194, 1195, 1211, 1212, 1218, 1219
- op-xor, 1188, 1191, 1217
- opcode-theory, 1188
- open-b-equiv, 154
- open-b-equiv3, 154
- open-b-xor, 154
- open-b-xor3, 154
- open-chip-system-operating-input
 - s-p, 808
- open-dual-eval-body-bindings, 243
- open-dual-eval-with-flag-1, 241
- open-dual-eval-with-flag-3, 242
- open-fm9001-interpreter, 547
- open-indices, 148
- open-list-as-collected-nth, 147
- open-make-list, 145
- open-make-list-off, 145
- open-map-up, 922
- open-map-up-inputs, 809
- open-nth, 142
- open-nth-off, 142
- open-nth-off1, 1063
- open-nth-on, 1009
- open-nth-on1, 1152
- open-nthcdr, 143
- open-operating-inputs-p, 915
- open-read-fn, 873
- open-run-inputs-p, 810
- open-run-inputs-p-add1, 810
- open-simulate-dual-eval-2, 244
- open-subrange, 144
- open-subrange-off, 145
- open-total-microcycles, 916
- open-v-threefix, 164
- operating-inputs-p, 915, 916, 920, 921, 923–928, 1014, 1021, 1023, 1025
- operating-inputs-p-1, 916
- operating-inputs-p-implies-hold
 - input, 916
- operating-inputs-p-implies-run-i
 - nputs-p, 916
- operating-inputs-p-off, 915

- operating-inputs-p-plus, 916
- or*, 131, 132, 624, 632, 634, 641,
 - 673, 716, 724, 725, 730, 828,
 - 829, 834, 838, 848, 850, 851,
 - 859, 869, 893, 894, 898–901,
 - 904, 906, 910, 914, 915
- or*-off, 131
- or-delay-args, 331–333
- or-delay-args-off, 333
- or-delay-args-to-ranges, 325, 326
- or-delay-args-to-ranges-off, 326
- or-zerop-tree, 15, 18–20
- or-zerop-tree-is-not-zerop-tree, 18
 - off, 18
- or-zerop-tree-off, 125
- or-zerop-tree-off1, 1177
- or-zerop-tree-on, 1089
- our-car-cdr-elim, 133
- our-car-cdr-elim-off, 133
- our-difference-x-x, 134
- our-equal-difference-0, 134
- our-member-append, 135
- out-depends, 379–381
- out-depends-error, 335, 349
- out-depends-error-off, 336
- out-depends-off, 381
- out-depends-simple, 255–258
- out-depends-simple-off, 257
- out-prop-signals, 356, 358, 360, 372,
 - 376
- out-prop-signals-off, 356
- out-types-error, 305, 349
- out-types-error-off, 305
- output-delay, 333
- output-delay-off, 333
- output-dependencies, 381
- output-dependencies-off, 382
- output-dependencies-simple, 257
- output-dependencies-simple-off, 258
- output-label, 355, 358, 362, 363, 366
- output-label-off, 355
- output-type, 300, 303, 304
- output-type-off, 300
- overflow-help, 519, 523, 530
- overflow-help*, 521, 522
- overflow-help\$netlist, 522, 541
- overflow-help\$value, 522
- overflow-help\$value-off, 522
- overflow-help\$value-zero, 523
- overflow-help\$value-zero-off, 523
- overflow-help&, 522, 523, 540
- overflow-help&-off, 522
- p-cell, 498, 499, 501
- p-cell*, 499
- p-cell\$netlist, 499, 502
- p-cell\$value, 499
- p-cell\$value-off, 499
- p-cell\$value-zero, 499
- p-cell&, 499, 502
- p-cell&-off, 499
- p-map-down, 926, 927
- p-map-up, 921, 1017
- p-name-prop-value-ok, 360, 361, 363
- p-name-prop-value-ok-off, 360
- p-t-wire-args-ok, 360, 361, 363
- p-t-wire-args-ok-off, 360
- paddressed-state-call-ok, 368, 369
- paddressed-state-call-ok-off, 368
- pairlist-append, 142
- pairlist-cons, 142
- pairlist-nlistp, 142
- pairlist-off, 141
- pairlists-are-equal-when-their-
 - 2nd-lists-are-nlistp, 142
- pairstates, 237–241, 955–970, 976–
 - 991, 999–1001
- parent-props-alist, 376, 377
- parent-props-alist-off, 377
- parent-synonym, 297, 298, 308, 316,
 - 321, 327, 333, 338, 339
- parent-synonym-off, 297
- parent-synonym-simple, 268, 270, 272
- parent-synonym-simple-off, 268
- parent-synonym0, 297
- parent-synonym0-off, 297
- parent-synonyms-list, 297, 298, 339
- parent-synonyms-list-off, 298

pc, 552
 pc+, 553
 pc-, 552
 pc-reg, 760, 763, 764, 924
 pc-reg-input, 760, 763, 807, 809, 810, 813, 915, 916
 pc-reg-input-off, 761
 pc-reg-off, 760
 pdelay-ok, 361, 362
 pdelay-ok-off, 362
 pdelays-errors, 362
 pdelays-errors-off, 362
 pdelays-ok, 362, 374
 pdelays-ok-off, 362
 pdrive-ok, 362, 363
 pdrive-ok-off, 363
 pdrives-errors, 363, 364
 pdrives-errors-off, 364
 pdrives-ok, 364, 374
 pdrives-ok-off, 364
 per-pf-delay-slope, 320, 331
 per-pf-delay-slope-off, 320
 per-std-load-delay-slope, 320, 331
 per-std-load-delay-slope-off, 320
 pf-difference, 307, 311
 pf-difference-off, 307
 pf-lessp, 307, 310
 pf-lessp-off, 307
 pf-plus, 306, 307
 pf-plus-off, 307
 pf-to-ma, 306, 311
 pf-to-ma-off, 306
 pf-to-std-load, 306, 310, 313, 320
 pf-to-std-load-off, 306
 pfp, 305–307, 310, 311, 313, 320, 364
 pfp-off, 306
 phalf-delay-ok, 360, 361
 phalf-delay-ok-off, 361
 pin-type-errors, 357, 358
 pin-type-errors-off, 357
 pin-type-ok, 357
 pin-type-ok-off, 357
 pin-types-ok, 358, 374
 pin-types-ok-off, 358
 ploading-ok, 364
 ploading-ok-off, 364
 ploadings-errors, 364, 365
 ploadings-errors-off, 365
 ploadings-ok, 365, 374
 ploadings-ok-off, 365
 plsi-name-ok, 365, 374
 plsi-name-ok-off, 366
 plus-0, 133
 plus-add1, 133
 plus-add1-arg1, 5
 plus-add1-arg1-off, 129
 plus-add1-arg1-off1, 1181
 plus-add1-arg1-on, 1092
 plus-add1-arg2, 5
 plus-add1-arg2-off, 129
 plus-add1-arg2-off1, 1180
 plus-add1-arg2-on, 1092
 plus-add1-off, 929
 plus-add1-on, 930
 plus-add1-sub1, 133
 plus-bottom, 133
 plus-cancellation, 5
 plus-cancellation-off, 5
 plus-difference-arg1, 6
 plus-difference-arg1-off, 129
 plus-difference-arg1-off1, 1180
 plus-difference-arg1-on, 1092
 plus-difference-arg2, 6
 plus-difference-arg2-off, 129
 plus-difference-arg2-off1, 1180
 plus-difference-arg2-on, 1092
 plus-fringe, 7–12
 plus-fringe-off, 129
 plus-fringe-off1, 1180
 plus-fringe-on, 1092
 plus-iplus, 95
 plus-iplus-off, 96
 plus-off, 133
 plus-quotient-2, 1103
 plus-quotient-2-lemma, 1103
 plus-quotient-2-lemma-off, 1103
 plus-quotient-2-lemma-on, 1183

- plus-quotient-2-off, 1103
- plus-quotient-2-on, 1183
- plus-remainder-times-quotient, 21
- plus-remainder-times-quotient-off, 21
- plus-tree, 7–12
- plus-tree-bagdiff, 8
- plus-tree-bagdiff-off, 8
- plus-tree-delete, 8
- plus-tree-delete-off, 8
- plus-tree-off, 129
- plus-tree-off1, 1180
- plus-tree-on, 1092
- plus-tree-plus-fringe, 9
- plus-tree-plus-fringe-off, 9
- plus-zero-arg2, 5
- plus-zero-arg2-off, 129
- plus-zero-arg2-off1, 1181
- plus-zero-arg2-on, 1093
- pmemory-word-call-ok, 368, 369
- pmemory-word-call-ok-off, 368
- pname-listp, 354, 375
- pname-listp-off, 354
- pnew-states-ok, 373, 374
- pnew-states-ok-off, 373
- position, 136, 149, 166
- position-name-indices, 149
- position-off, 136
- post-inc-p, 543, 545, 546, 548, 565, 624, 634, 641, 673, 716, 724, 725, 730, 829, 834, 838, 850, 851, 859, 869, 894, 898–901, 904, 906, 910, 914, 915
- post-inc-p-off, 543
- pout-depends-errors, 366, 367
- pout-depends-errors-off, 366
- pout-depends-ok, 367, 374
- pout-depends-ok-off, 367
- pout-depends-one-ok, 366
- pout-depends-one-ok-off, 366
- pout-type-errors, 359, 360
- pout-type-errors-off, 359
- pout-type-ok, 358, 359
- pout-type-ok-off, 359
- pout-types-ok, 359, 374
- pout-types-ok-off, 360
- pre-dec-p, 543, 545, 546, 548, 565, 622, 624, 625, 631, 634, 635, 639, 641, 649, 653, 673, 716, 724, 725, 730, 828, 829, 834, 836, 838, 846, 850, 851, 857, 859, 868, 869, 872, 893, 894, 898–902, 904–911, 913–915
- pre-dec-p-off, 543
- pred-error, 250–252, 279, 285, 286, 288, 300, 302, 310, 311, 353, 355, 358, 361–363, 368, 371, 372, 374, 377, 379–386
- pred-error-off, 250
- predicate-properties, 353, 354
- predicate-properties-off, 354
- presult-form-errors, 370, 371
- presult-form-errors-off, 371
- presult-ok, 371–373
- presult-ok-off, 371
- presults-errors, 371–373
- presults-errors-off, 372
- presults-ok, 372, 374
- presults-ok-off, 373
- primitive-count, 379
- primitive-count-off, 379
- primitive-ok, 377
- primitive-ok-off, 377
- primitive-prop-errors, 374, 377
- primitive-prop-errors-off, 375
- primitive-prop-ok, 373, 374
- primitive-prop-ok-off, 374
- primitive-properties, 294, 295, 375
- primitive-properties-off, 295
- primitives-monotone, 952
- primp, 238–242, 253–255, 257, 260, 263, 265, 272, 278, 282, 286–288, 292, 343, 379–382, 385, 436, 938, 954, 955, 976, 998–1000, 1008
- primp-database, 238, 377
- primp-database-errors, 377
- primp-database-errors-off, 377

- primp-database-predicate, 377
- primp-database-predicate-off, 378
- primp-lookup, 238, 286, 382, 385, 436
- primp-monotone, 998
- primp-monotone-off, 1041
- primp-monotone-on, 1130
- primp-off, 238
- primp-preserves-good-s, 1008
- primp-preserves-good-s-off, 1034
- primp-preserves-good-s-on, 1123
- primp2, 238, 239, 253, 255, 257, 260, 263, 272, 273, 278, 286, 379–382, 385, 436
- procmon-monotone, 947
- procmon-monotone-off, 1046
- procmon-monotone-on, 1135
- procmon-preserves-good-s, 1006
- procmon-preserves-good-s-off, 1036
- procmon-preserves-good-s-on, 1125
- procmon\$netlist, 400, 791
- procmon\$value, 400
- procmon\$value-off, 401
- procmon&, 400, 791
- procmon&-off, 400
- prop*-theory, 132
- properp, 136, 140–145, 147–149, 151, 164, 168, 169, 172–175, 184, 191, 202–205, 242, 253, 254, 317, 355, 358, 376, 420–422, 424, 426, 429, 430, 432, 433, 435, 452, 456, 459, 461, 462, 464, 468, 473, 475, 478, 481, 482, 494, 495, 510, 516, 517, 523, 526, 528, 531, 541, 547, 548, 563, 566, 567, 578, 588–590, 592, 670, 673, 690, 704, 714, 715, 718, 722, 733, 739–741, 744–746, 748, 749, 751, 753, 755, 759, 765, 776, 779, 780, 783, 792, 795, 800, 801, 805–810, 874, 918, 925, 926, 930, 954, 971, 974–976, 1007, 1017–1020, 1098
- properp-append, 136
- properp-append-nil, 136
- properp-as-null-nthcdr, 143
- properp-as-null-nthcdr-off, 143
- properp-cddr-f\$tv-alu-help, 510
- properp-cdr-f\$tv-dec-pass-ng, 468
- properp-collect-value, 168
- properp-cons, 136
- properp-dual-eval-0, 242
- properp-dual-eval-2, 242
- properp-f\$tv-alu-help, 510
- properp-f\$tv-dec-pass-ng, 468
- properp-firstn, 140
- properp-fv-if, 203
- properp-if, 136
- properp-if*, 136
- properp-indices, 149
- properp-length-cv-fetch1, 670
- properp-length-dual-port-ram-value, 175
- properp-length-f\$core-alu, 531
- properp-length-f\$cv, 704
- properp-length-f\$dec-pass, 475
- properp-length-f\$decode-5, 578
- properp-length-f\$encode-32, 587
- properp-length-f\$extend-immediate, 748
- properp-length-f\$mpg, 494
- properp-length-f\$next-cntl-state, 718
- properp-length-f\$next-state, 689
- properp-length-f\$read-regs, 739
- properp-length-f\$select-op-code, 566
- properp-length-f\$shift-or-buf, 526
- properp-length-f\$update-flags, 744
- properp-length-f\$v-if-f-4, 567
- properp-length-f\$v-inc4\$v, 464
- properp-length-f\$write-regs, 741
- properp-length-fv-or, 202
- properp-length-fv-shift-right, 204
- properp-length-fv-xor, 202
- properp-length-ir-accessors, 547
- properp-length-memory-value, 184
- properp-length-next-memory-state, 191
- properp-length-v-pullup, 204

properp-length-v-wire, 204
 properp-length-vft-buf, 205
 properp-length-write-regs, 733
 properp-make-list, 145
 properp-mpg, 494
 properp-nlistp, 136
 properp-off, 136
 properp-pairlist, 142
 properp-read-fn, 874
 properp-read-mem, 173
 properp-read-mem-32, 173
 properp-read-mem1, 173
 properp-restn, 141
 properp-shift-right-names, 523
 properp-subrange, 143
 properp-v-threefix, 164
 prove-boolp, 149
 ps-pfp, 317, 320, 361
 ps-pfp-off, 317
 pstate-type-list-errors, 369
 pstate-type-list-errors-off, 369
 pstate-type-ok, 369, 370
 pstate-type-ok-off, 369
 pstate-types-errors, 369, 370
 pstate-types-errors-off, 370
 pstate-types-ok, 370, 374
 pstate-types-ok-off, 370
 pstates-ok, 367, 374
 pstates-ok-off, 367
 pullup-monotone, 949
 pullup-monotone-off, 1044
 pullup-monotone-on, 1133
 pullup-preserved-good-s, 1006
 pullup-preserved-good-s-off, 1035
 pullup-preserved-good-s-on, 1124
 pullup\$netlist, 409, 433, 800
 pullup\$value, 409
 pullup\$value-off, 409
 pullup&, 409, 433, 799
 pullup&-off, 409

 quotient-1-arg1, 28
 quotient-1-arg1-casesplit, 28
 quotient-1-arg1-casesplit-off, 120
 quotient-1-arg1-casesplit-off1, 1172
 quotient-1-arg1-casesplit-on, 1084
 quotient-1-arg1-off, 120
 quotient-1-arg1-off1, 1172
 quotient-1-arg1-on, 1084
 quotient-1-arg2, 28
 quotient-1-arg2-off, 121
 quotient-1-arg2-off1, 1172
 quotient-1-arg2-on, 1084
 quotient-add1, 25
 quotient-add1-off, 122
 quotient-add1-off1, 1174
 quotient-add1-on, 1085
 quotient-add1-plus-x-x-2, 1214
 quotient-difference-lessp-arg2, 50
 quotient-difference-lessp-arg2-off, 64
 quotient-difference-plus-x-x-add1-plus-y-y-2, 1214
 quotient-difference-plus-x-x-plus-y-y-2, 1214
 quotient-difference1, 26
 quotient-difference1-off, 121
 quotient-difference1-off1, 1173
 quotient-difference1-on, 1085
 quotient-difference2, 27
 quotient-difference2-off, 121
 quotient-difference2-off1, 1173
 quotient-difference2-on, 1084
 quotient-difference3, 27
 quotient-difference3-off, 121
 quotient-difference3-off1, 1173
 quotient-difference3-on, 1084
 quotient-exp, 30
 quotient-exp-off, 119
 quotient-exp-off1, 1171
 quotient-exp-on, 1083
 quotient-lessp, 134
 quotient-lessp-arg1, 27
 quotient-lessp-arg1-off, 121
 quotient-lessp-arg1-off1, 1173
 quotient-lessp-arg1-on, 1085
 quotient-lessp-double-remainder\$help, 1193

- quotient-noop, 25
- quotient-noop-off, 122
- quotient-noop-off1, 1174
- quotient-noop-on, 1085
- quotient-of-non-number, 25
- quotient-of-non-number-off, 122
- quotient-of-non-number-off1, 1174
- quotient-of-non-number-on, 1085
- quotient-plus, 25
- quotient-plus-fact, 27
- quotient-plus-fact-off, 121
- quotient-plus-fact-off1, 1173
- quotient-plus-fact-on, 1084
- quotient-plus-off, 122
- quotient-plus-off1, 1173
- quotient-plus-on, 1085
- quotient-plus-proof, 25
- quotient-plus-proof-off, 122
- quotient-plus-proof-off1, 1173
- quotient-plus-proof-on, 1085
- quotient-plus-times-times, 27
- quotient-plus-times-times-insta-
nce, 28
- nce-off, 121
- nce-off1, 1172
- nce-on, 1084
- quotient-plus-times-times-off, 121
- quotient-plus-times-times-off1, 1172
- quotient-plus-times-times-on, 1084
- quotient-plus-times-times-proof, 27
- off, 121
- off1, 1172
- on, 1084
- quotient-plus-x-x-2, 1097
- quotient-plus-x-x-2-off, 1114
- quotient-plus-x-x-2-with-extra-
arg, 1103
- arg-off, 1103
- arg-on, 1183
- quotient-quotient, 28
- quotient-quotient-off, 121
- quotient-quotient-off1, 1172
- quotient-quotient-on, 1084
- quotient-remainder, 27
- quotient-remainder-instance, 27
- quotient-remainder-instance-off, 121
- 1, 1173
- quotient-remainder-instance-on, 1084
- quotient-remainder-off, 121
- quotient-remainder-off1, 1173
- quotient-remainder-on, 1084
- quotient-remainder-times, 27
- quotient-remainder-times-off, 121
- quotient-remainder-times-off1, 1173
- quotient-remainder-times-on, 1084
- quotient-remainder-uniqueness, 50
- quotient-remainder-uniqueness-o
ff, 110
- ff1, 1162
- n, 1073
- quotient-stuff, 550
- quotient-sub1, 25
- quotient-sub1-off, 122
- quotient-sub1-off1, 1174
- quotient-sub1-on, 1085
- quotient-times, 26
- quotient-times-instance, 26
- quotient-times-instance-off, 122
- quotient-times-instance-off1, 1173
- quotient-times-instance-on, 1085
- quotient-times-instance-temp, 26
- quotient-times-instance-temp-of
f, 26
- quotient-times-instance-temp-pr
oof, 25
- oof-off, 122
- oof-off1, 1173
- oof-on, 1085
- quotient-times-off, 122
- quotient-times-off1, 1173
- quotient-times-on, 1085
- quotient-times-proof, 26
- quotient-times-proof-off, 122
- quotient-times-proof-off1, 1173
- quotient-times-proof-on, 1085
- quotient-times-times, 26
- quotient-times-times-off, 26
- quotient-times-times-proof, 26

- quotient-times-times-proof-off, 121
- quotient-times-times-proof-off1, 1173
- quotient-times-times-proof-on, 1085
- quotient-x-x, 28
- quotient-x-x-off, 120
- quotient-x-x-off1, 1172
- quotient-x-x-on, 1084
- quotient-zero, 25
- quotient-zero-off, 122
- quotient-zero-off1, 1174
- quotient-zero-on, 1085
- quotients, 29

- r0, 551
- r0+, 552
- r0-, 552
- r1, 551
- r1+, 552
- r1-, 552
- r10, 551
- r10+, 553
- r10-, 552
- r11, 551
- r11+, 553
- r11-, 552
- r12, 551
- r12+, 553
- r12-, 552
- r13, 551
- r13+, 553
- r13-, 552
- r14, 551
- r14+, 553
- r14-, 552
- r15, 551
- r15+, 553
- r15-, 552
- r2, 551
- r2+, 553
- r2-, 552
- r3, 551
- r3+, 553
- r3-, 552
- r4, 551

- r4+, 553
- r4-, 552
- r5, 551
- r5+, 553
- r5-, 552
- r6, 551
- r6+, 553
- r6-, 552
- r7, 551
- r7+, 553
- r7-, 552
- r8, 551
- r8+, 553
- r8-, 552
- r9, 551
- r9+, 553
- r9-, 552
- ram, 169–171, 265, 558, 973, 1011, 1013
- ram-enable-circuit-monotone, 949
- ram-enable-circuit-monotone-off, 1045
- ram-enable-circuit-monotone-on, 1134
- ram-enable-circuit-preserves-go
 - od-s, 1006
 - od-s-off, 1035
 - od-s-on, 1125
- ram-enable-circuit\$netlist, 408, 739
- ram-enable-circuit\$value, 408
- ram-enable-circuit\$value-off, 408
- ram-enable-circuit&, 408, 738
- ram-enable-circuit&-off, 408
- ram-guts, 169, 170, 267, 368, 387, 388, 559, 929, 930, 972, 1017, 1020
- ramp, 169–171, 267, 368, 387, 388, 559, 929, 930, 972, 973, 975, 1017, 1018, 1020
- ramp-implies-not-fourp, 1018
- ramp-implies-not-fourp-off, 1028
- ramp-implies-not-fourp-on, 1118
- ramp-mem, 171, 173, 174
- ramp-mem-off, 171
- ramp-mem1, 171–174
- ramp-mem1-off, 171

range-dependencies, 319–322, 324, 328–331
 range-dependencies-off, 319
 range-dependencies-off1, 322
 range-dependencies-off2, 329
 range-dependencies-on, 321
 range-dependencies-on1, 326
 range-max, 319, 320, 322, 324, 325, 329, 330
 range-max-off, 319
 range-min, 319–321, 324, 325, 328, 329
 range-min-off, 319
 range-plus, 320, 323, 328, 331
 range-plus-off, 320
 rangep, 319–321, 324, 325, 327–329
 rangep-off, 319
 read-fn, 873–877, 879, 880, 882, 884
 read-fn-off, 873
 read-mem, 170, 173, 174, 182, 545–547, 733, 734, 824, 842, 853, 893, 894, 901, 905, 916, 926, 954, 971
 read-mem-monotone, 954
 read-mem-monotone-off, 994
 read-mem-monotone-off1, 1149
 read-mem-monotone-on, 1059
 read-mem-off, 170
 read-mem-write-mem, 173
 read-mem1, 170, 172, 173, 953, 954
 read-mem1-monotone, 953
 read-mem1-monotone-induction, 953
 read-mem1-monotone-induction-of f, 1042
 read-mem1-monotone-induction-on, 1131
 read-mem1-monotone-off, 993
 read-mem1-monotone-off1, 1148
 read-mem1-monotone-on, 1059
 read-mem1-off, 170
 read-regs, 733–735, 739, 815, 817, 818, 826, 830, 832, 834, 836, 840, 841, 844–846, 848, 855, 857, 866, 870, 874–882, 884, 888, 890, 891, 895, 897, 899, 901, 902, 904–909, 911, 917
 read-regs-off, 733
 read-regs-write-regs-f, 734
 read-regs=read-mem, 734
 read-regs=read-mem-write-mem, 734
 reada0->reada3\$sim, 899
 reada0\$step, 835
 reada1\$step, 837
 reada2\$step, 839
 reada3\$induction, 877, 878
 reada3\$progress, 879
 reada3\$progress-help, 878
 reada3\$step0, 841
 reada3\$step1, 843
 readb0->readb3\$sim, 902
 readb0\$step, 845
 readb1\$step, 847
 readb2\$step, 849
 readb3\$induction, 881
 readb3\$progress, 883
 readb3\$progress-help, 881
 readb3\$step0, 852
 readb3\$step1, 854
 reg*, 476, 477
 reg*-off, 476
 reg*\$destructure, 476
 reg*\$destructure-off, 476
 reg-40\$value-as-cntl-state-off, 779
 reg-body, 475–478
 reg-body-induction, 477
 reg-body-off, 475
 reg-body\$all-unbound-in-body, 477
 reg-body\$all-unbound-in-body-off, 477
 reg-body\$state, 478
 reg-body\$unbound-in-body, 477
 reg-body\$unbound-in-body-off, 477
 reg-body\$value, 477
 reg-direct->not-reg-indirect, 548
 reg-direct-p, 543, 545, 546, 548, 565, 632, 673, 716, 724, 725, 727, 828, 830, 842, 848, 893, 895, 902, 904, 914, 915

- reg-direct-p-off, 543
- reg-indirect-p, 543, 548
- reg-indirect-p-off, 543
- reg-mode-theory, 543
- reg-size, 541, 547
- reg\$netlist, 476, 477, 776
- reg\$state, 478
- reg\$state-off, 478
- reg\$value, 478
- reg\$value-off, 478
- reg&, 476, 478, 775
- reg&-off, 476
- rega->rega\$sim, 894
- rega\$step, 829
- regb->update\$sim, 896
- regb\$step, 831
- regfile*, 735, 737–739
- regfile*-off, 738
- regfile*\$destructure, 737
- regfile*\$destructure-off, 738
- regfile-okp, 810, 814, 816, 817, 819, 821, 823, 825, 827, 829, 831, 833, 835, 837, 839, 841, 843, 845, 847, 849, 852, 854, 856, 858, 860, 862, 863, 865, 867, 869, 871, 875, 876, 878, 880, 881, 883, 885–887, 889–891, 894, 896, 898, 899, 903, 905, 908, 910, 912
- regfile\$netlist, 739, 776
- regfile\$state, 741
- regfile\$state-off, 742
- regfile\$value, 740
- regfile\$value-off, 740
- regfile&, 738–741, 775
- regfile&-off, 738
- regs, 544, 547, 559, 761–763, 921, 922, 925
- regs-address, 606, 608, 610, 612, 614, 616, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 659, 661, 663, 665, 666, 668, 761–764, 781–783, 792–794, 801–804, 811–814
- regs-address-off, 606
- regs-off, 545
- remainder-1-arg1, 24
- remainder-1-arg1-off, 122
- remainder-1-arg1-off1, 1174
- remainder-1-arg1-on, 1086
- remainder-1-arg2, 24
- remainder-1-arg2-off, 122
- remainder-1-arg2-off1, 1174
- remainder-1-arg2-on, 1086
- remainder-add1, 21
- remainder-add1-off, 124
- remainder-add1-off1, 1176
- remainder-add1-on, 1087
- remainder-add1-plus-multiple, 1097
- remainder-add1-plus-multiple-of f, 1115
- remainder-difference-plus-x-x-add1-plus-y-y, 1214
- remainder-difference-plus-x-x-plus-y-y, 1214
- remainder-difference1, 23
- remainder-difference1-off, 123
- remainder-difference1-off1, 1175
- remainder-difference1-on, 1086
- remainder-difference2, 23
- remainder-difference2-off, 123
- remainder-difference2-off1, 1175
- remainder-difference2-on, 1086
- remainder-difference3, 23
- remainder-difference3-off, 23
- remainder-equals-its-first-argument, 27
- remainder-off, 27
- remainder-exp, 30
- remainder-exp-exp, 30
- remainder-exp-exp-off, 119
- remainder-exp-exp-off1, 1171
- remainder-exp-exp-on, 1083
- remainder-exp-off, 120
- remainder-exp-off1, 1171
- remainder-exp-on, 1083

- remainder-gcd, 33
- remainder-gcd-off, 117
- remainder-gcd-off1, 1169
- remainder-gcd-on, 1080
- remainder-noop, 21
- remainder-noop-off, 124
- remainder-noop-off1, 1176
- remainder-noop-on, 1088
- remainder-of-non-number, 21
- remainder-of-non-number-off, 124
- remainder-of-non-number-off1, 1176
- remainder-of-non-number-on, 1088
- remainder-plus, 21
- remainder-plus-fact, 24
- remainder-plus-fact-off, 123
- remainder-plus-fact-off1, 1174
- remainder-plus-fact-on, 1086
- remainder-plus-multiple, 1097
- remainder-plus-multiple-off, 1115
- remainder-plus-off, 124
- remainder-plus-off1, 1176
- remainder-plus-on, 1087
- remainder-plus-proof, 21
- remainder-plus-proof-off, 124
- remainder-plus-proof-off1, 1176
- remainder-plus-proof-on, 1087
- remainder-plus-times-times, 24
- remainder-plus-times-times-instance, 24
 - ance-off, 123
 - ance-off1, 1174
 - ance-on, 1086
- remainder-plus-times-times-off, 123
- remainder-plus-times-times-off1, 1174
- remainder-plus-times-times-on, 1086
- remainder-plus-times-times-proof, 24
 - f, 24
 - f-off, 123
 - f-off1, 1174
 - f-on, 1086
- remainder-plus-x-x-2, 1097
- remainder-plus-x-x-2-off, 1114
- remainder-quotient-elim, 21
- remainder-quotient-elim-off, 124
- remainder-quotient-elim-off1, 1176
- remainder-quotient-elim-on, 1087
- remainder-remainder, 24
- remainder-remainder-off, 123
- remainder-remainder-off1, 1174
- remainder-remainder-on, 1086
- remainder-theorem-for-subb, 1194
- remainder-times-times, 22
- remainder-times-times-off, 22
- remainder-times-times-proof, 22
- remainder-times-times-proof-off, 123
 - 1, 1175
- remainder-times-times-proof-on, 1087
- remainder-times1, 22
- remainder-times1-instance, 22
- remainder-times1-instance-off, 124
- remainder-times1-instance-off1, 1175
- remainder-times1-instance-on, 1087
- remainder-times1-instance-proof, 22
 - off, 124
 - off1, 1175
 - on, 1087
- remainder-times1-off, 124
- remainder-times1-off1, 1175
- remainder-times1-on, 1087
- remainder-times1-proof, 22
- remainder-times1-proof-off, 124
- remainder-times1-proof-off1, 1175
- remainder-times1-proof-on, 1087
- remainder-times2, 23
- remainder-times2-instance, 23
- remainder-times2-instance-off, 123
- remainder-times2-instance-off1, 1175
- remainder-times2-instance-on, 1086
- remainder-times2-off, 123
- remainder-times2-off1, 1175
- remainder-times2-on, 1087
- remainder-times2-proof, 22
- remainder-times2-proof-off, 123
- remainder-times2-proof-off1, 1175
- remainder-times2-proof-on, 1087
- remainder-x-x, 24
- remainder-x-x-off, 122
- remainder-x-x-off1, 1174

- remainder-x-x-on, 1086
- remainder-zero, 21
- remainder-zero-off, 124
- remainder-zero-off1, 1176
- remainder-zero-on, 1088
- remainders, 25
- remove-duplicates, 246, 338, 367
- remove-duplicates-off, 246
- remove-inegs, 87–94
- remove-inegs-off, 92
- report-error, 250
- report-error-off, 250
- required-for-prop, 351, 352
- required-for-prop-off, 352
- required-props, 352, 353, 379, 380, 383–385
- required-props-off, 352
- reset-, 760, 763
- reset-input, 760, 763, 809, 810, 813, 915, 916
- reset-input-off, 761
- reset-off, 760
- reset-chip, 1024
- reset-chip-from-any-state, 1024
- reset-chip-from-any-state-off, 1027
- reset-chip-from-any-state-on, 1116
- reset-chip-off, 1027
- reset-chip-on, 1116
- reset-sequence, 1012, 1014
- reset-sequence-chip-1, 1015, 1016, 1020, 1021, 1024
- reset-sequence-chip-1-off, 1030
- reset-sequence-chip-1-on, 1119
- reset-sequence-chip-1-vs-2, 1016
- reset-sequence-chip-1-vs-2-off, 1030
- reset-sequence-chip-1-vs-2-on, 1119
- reset-sequence-chip-2, 1015, 1016
- reset-sequence-chip-2-off, 1030
- reset-sequence-chip-2-on, 1119
- reset-sequence-off, 1031
- reset-sequence-on, 1120
- reset-vector, 1012
- reset-vector-chip, 1014, 1015
- reset-vector-chip-off, 1030
- reset-vector-chip-on, 1119
- reset-vector-off, 1032
- reset-vector-on, 1121
- reset-works, 1014
- reset-works-off, 1031
- reset-works-on, 1120
- resolve-names, 555, 556
- restn, 140–142, 146, 151, 152, 159–161, 166, 168, 452, 473, 516
- restn-append, 141
- restn-nat-to-v-0-hack, 160
- restn-off, 140
- restn-v-not, 159
- restn-v-not-off, 159
- results-length, 371, 372
- results-length-off, 371
- rev-0, 435
- rev1, 137, 138, 165, 435, 953
- rev1-off, 137
- reverse, 138, 165, 170, 171, 173, 353, 953
- reverse-off, 138
- rewrite-and*, 132
- rewrite-chip-system-input-invariant, 808
- rewrite-equal-v-to-int-0, 1197
- rewrite-equal-v-to-int-0-off, 1197
- rewrite-equal-v-to-nat-0, 1187
- rewrite-equal-v-to-nat-0-off, 1187
- rewrite-fm9001-next-state-for-step-lemmas, 810
- rewrite-not*, 132
- rewrite-or*, 132
- rewrite-value, 166
- rewrite-value-4x, 166
- rn-a, 541, 546–548, 617, 623, 625, 633, 647, 649, 716, 718, 721, 830, 831, 836, 838, 839, 848, 866–869, 895, 896, 901, 902, 904, 909, 910
- rn-a-off, 542
- rn-b, 541, 545, 546, 548, 619, 621, 627, 631, 635, 639, 641, 651, 653, 716, 718, 721, 832–835,

840, 841, 846, 847, 850–852,
 857, 859, 860, 870–873, 897–
 899, 901, 904–908, 911–913
 rn-b-off, 542
 rom, 169
 rom-guts, 169, 170, 267, 368, 387,
 388, 559, 929, 930, 973, 1017,
 1020
 romp, 169, 170, 267, 368, 387, 388,
 559, 929, 930, 972, 973, 1017,
 1018, 1020
 romp-implies-not-fourp, 1018
 romp-implies-not-fourp-off, 1028
 romp-implies-not-fourp-on, 1117
 run-fm9001, 764, 765, 809, 815–873,
 875, 877, 879, 880, 882, 883,
 885, 887, 888, 890–892, 895,
 897, 899, 900, 903, 906, 909,
 911, 913, 916, 917, 921, 924,
 926, 1014
 run-fm9001-base-case, 764
 run-fm9001-off, 764
 run-fm9001-plus, 765
 run-fm9001-step-case, 764
 run-fm9001-step-case-off, 765
 run-inputs-p, 809, 810, 814, 816, 818,
 819, 821, 823, 825, 827, 829,
 832, 833, 835, 837, 839, 842,
 843, 846, 847, 849, 852, 854,
 856, 858, 860, 862, 864, 866,
 867, 870, 871, 875, 876, 878,
 880, 882, 883, 885, 886, 888–
 890, 892, 895, 896, 898, 900,
 903, 905, 908, 910, 912, 916
 run-inputs-p-1, 810
 run-inputs-p-off, 809
 run-inputs-p-plus, 810
 run-vector, 1012
 run-vector-chip-1, 1015
 run-vector-chip-1-off, 1030
 run-vector-chip-1-on, 1119
 run-vector-chip-2, 1015
 run-vector-chip-2-off, 1030
 run-vector-chip-2-on, 1119
 run-vector-off, 1031
 run-vector-on, 1121
 rw-, 604, 607, 609, 611, 613, 615,
 617, 618, 620, 622, 624, 626,
 628, 630, 632, 634, 636, 638,
 640, 642, 644, 646, 648, 650,
 652, 654, 656, 658, 660, 662,
 664, 666, 668, 761–764, 782,
 784, 793, 794, 796, 802–805,
 811–814
 rw-off, 604
 s-approx, 929–932, 938, 953, 954,
 972–976, 1002, 1003, 1009–
 1011, 1016, 1018–1020, 1024
 s-approx-alist, 930–933, 1002, 1003
 s-approx-alist-implies-s-approx
 -collect-value, 1003
 -collect-value-off, 1039
 -collect-value-on, 1128
 -list-collect-value, 1003
 -list-collect-value-off, 1039
 -list-collect-value-on, 1129
 -value, 1002
 -value-off, 1040
 -value-on, 1129
 s-approx-alist-off, 1056
 s-approx-alist-on, 1145
 s-approx-cadr-nth, 1009
 s-approx-cadr-nth-off, 1033
 s-approx-cadr-nth-on, 1123
 s-approx-constant-ram-x-constant
 -ram-x, 975
 -ram-x-off, 996
 -ram-x-off1, 1151
 -ram-x-on, 1061
 s-approx-constant-ram-x-id, 975
 s-approx-constant-ram-x-id-off, 996
 s-approx-constant-ram-x-id-off1, 1151
 s-approx-constant-ram-x-id-on, 1061
 s-approx-constant-ram-x-write-me
 m, 975
 m-off, 997
 m-off1, 1151

- m-on, 1062
- m1, 975
- m1-case-3, 975
- m1-case-3-off, 997
- m1-case-3-off1, 1151
- m1-case-3-on, 1062
- m1-off, 997
- m1-off1, 1151
- m1-on, 1062
- s-approx-implies-b-approx, 938
- s-approx-implies-b-approx-off, 1053
- s-approx-implies-b-approx-on, 1143
- s-approx-implies-properp-read-me
 - m, 954
 - m-off, 994
 - m-off1, 1149
 - m-on, 1060
 - m1, 954
 - m1-off, 994
 - m1-off1, 1149
 - m1-on, 1060
- s-approx-implies-s-approx-alist, 1003
 - off, 1039
 - on, 1129
- s-approx-list, 1002, 1003
- s-approx-list-implies-s-approx-
 - alist, 1002
 - alist-off, 1040
 - alist-on, 1129
- s-approx-list-off, 1040
- s-approx-list-on, 1129
- s-approx-make-list, 1018
- s-approx-make-list-off, 1029
- s-approx-make-list-on, 1118
- s-approx-off, 973
- s-approx-off1, 1056
- s-approx-on, 996
- s-approx-on1, 1146
- s-approx-opener, 972
- s-approx-opener-off, 996
- s-approx-opener-off1, 1150
- s-approx-opener-on, 1061
- s-approx-write-mem-id, 975
- s-approx-write-mem-id-off, 996
- s-approx-write-mem-id-off1, 1150
- s-approx-write-mem-id-on, 1061
- s-approx-write-mem1-id, 975
- s-approx-write-mem1-id-off, 996
- s-approx-write-mem1-id-off1, 1150
- s-approx-write-mem1-id-on, 1061
- s-approx-x-x, 929
- s-approx-x-x-off, 1056
- s-approx-x-x-on, 1145
- s-knownp, 930, 1010, 1011, 1016
- s-knownp-implies-s-approx-is-eq
 - ual, 930
 - ual-off, 972
 - ual-off1, 1056
 - ual-on, 995
 - ual-on1, 1145
- s-knownp-off, 972
- s-knownp-off1, 1056
- s-knownp-on, 995
- s-knownp-on1, 1145
- same-fix-int-implies-not-ilessp, 62
 - off, 63
- sefa0->sefa1\$sim, 908
- sefa0\$step, 865
- sefa1\$step, 867
- sefb0->sefb1\$sim, 910
- sefb0\$step, 869
- sefb1->sefb1\$sim, 912
- sefb1\$step, 871
- select-immediate, 605, 608, 610, 612,
 - 613, 615, 617, 619, 621, 623,
 - 625, 627, 629, 631, 633, 635,
 - 637, 639, 641, 643, 645, 647,
 - 649, 651, 653, 655, 657, 659,
 - 661, 663, 664, 666, 668, 761–
 - 763, 781, 782, 793, 794, 802–
 - 804, 811–813
- select-immediate-off, 605
- select-op-code*, 566
- select-op-code\$netlist, 566, 714
- select-op-code\$value, 567
- select-op-code\$value-off, 567
- select-op-code&, 566, 567, 714
- select-op-code&-off, 566

set-diff, 245, 252, 254, 280, 337, 353, 375
 set-diff-off, 245
 set-equal, 245, 265, 336, 365
 set-equal-off, 245
 set-flags, 541, 545, 548, 621, 639, 670, 673, 716, 718, 721, 724, 828, 834, 857, 893, 897, 899, 907, 915
 set-flags-off, 542
 set-flags-theory, 543
 set-some-flags, 591, 673, 716, 724, 828, 893, 915
 set-some-flags-off, 591
 set-value, 247, 302
 set-value-off, 247
 shift-or-buf, 525–527, 531
 shift-or-buf-cntl, 523, 525, 526
 shift-or-buf-cntl*, 524
 shift-or-buf-cntl\$netlist, 524, 528
 shift-or-buf-cntl\$value, 524
 shift-or-buf-cntl\$value-off, 525
 shift-or-buf-cntl\$value-zero, 525
 shift-or-buf-cntl&, 524, 525, 528
 shift-or-buf-cntl&-off, 524
 shift-or-buf-is-asr, 526
 shift-or-buf-is-buf, 526
 shift-or-buf-is-lsr, 527
 shift-or-buf-is-ror, 526
 shift-or-buf-off, 526
 show-that-nth=t, 152
 show-that-nth=t-off, 152
 sign-extend, 161, 162, 546, 748, 830, 848, 895, 904
 sign-extend-as-append, 162
 sign-extend-as-append-off, 162
 sign-extend-off, 162
 signal-namep, 297, 298, 308, 309, 313, 316, 321, 325, 326, 332, 333, 388
 signal-namep-off, 297
 simple-dependency-table, 257, 258, 276
 simple-dependency-table-off, 257
 simplify-if*, 132
 simulate, 243, 244, 1010, 1011, 1016, 1020, 1021, 1024
 simulate-contains-simulate-dual
 -eval-2, 1024
 -eval-2-off, 1027
 -eval-2-on, 1116
 simulate-dual-eval-2, 244, 809, 925, 928, 1014, 1022, 1024, 1026
 simulate-dual-eval-2-off, 244
 simulate-monotone, 1009
 simulate-monotone-induction, 1009
 simulate-monotone-induction-off, 1033
 simulate-monotone-induction-on, 1122
 simulate-monotone-off, 1033
 simulate-monotone-on, 1122
 simulate-reset-chip-final-state, 1016
 -off, 1030
 -on, 1119
 single-number-induction, 33
 single-number-induction-off, 118
 single-number-induction-off1, 1169
 single-number-induction-on, 1081
 singleton-collect-value, 168
 slope-times-load, 320, 323
 slope-times-load-off, 321
 some-eval\$s-to-0, 83, 84
 some-eval\$s-to-0-append, 84
 some-eval\$s-to-0-append-off, 103
 some-eval\$s-to-0-append-off1, 1155
 some-eval\$s-to-0-append-on, 1066
 some-eval\$s-to-0-eliminator, 84
 some-eval\$s-to-0-eliminator-of
 f, 103
 f1, 1155
 some-eval\$s-to-0-eliminator-on, 1066
 some-eval\$s-to-0-off, 103
 some-eval\$s-to-0-off1, 1155
 some-eval\$s-to-0-on, 1066
 split-out-ineg-terms, 87, 88, 92
 split-out-ineg-terms-off, 102
 split-out-ineg-terms-off1, 1154
 split-out-ineg-terms-on, 1065

state, 606, 608, 610, 612, 614, 616,
 617, 619, 621, 623, 625, 627,
 629, 631, 633, 635, 637, 639,
 641, 643, 645, 647, 649, 651,
 653, 655, 657, 659, 661, 663,
 665, 666, 668, 763, 783, 784,
 794, 796, 804, 814
 state-label, 355, 369, 373
 state-label-off, 355
 state-off, 606
 state-okp, 388
 state-okp-0, 386–388
 state-okp-0-off, 388
 state-okp-off, 388
 state-simple-okp, 267, 268
 state-simple-okp-off, 268
 state-type-requirement, 385, 388
 state-type-requirement-off, 386
 state-type-requirement-simple, 265–
 268
 state-type-requirement-simple-o
 ff, 266
 state-types-error, 336, 349
 state-types-error-off, 336
 states-list-or-nil, 282, 283, 293
 states-list-or-nil-off, 282
 std-drive-to-ma, 306, 314
 std-drive-to-ma-off, 306
 std-load-to-pf, 306, 307, 310, 311,
 313, 320
 std-load-to-pf-off, 306
 store-cc, 541, 545, 548, 620, 634, 670,
 673, 716, 724, 725, 727, 728,
 828, 831, 834, 842, 850, 851,
 853, 893, 896, 897, 899, 902,
 904, 905, 914, 915
 store-cc-off, 542
 store-resultp, 544, 545, 548, 563, 620,
 634, 673, 716, 724, 725, 727,
 728, 828, 831, 834, 842, 850,
 851, 853, 893, 896, 897, 899,
 902, 904, 905, 914, 915, 1188–
 1196, 1200–1216
 store-resultp-mux, 559, 561
 store-resultp-mux*, 560
 store-resultp-mux\$netlist, 560, 563
 store-resultp-mux\$value, 560
 store-resultp-mux\$value-off, 561
 store-resultp-mux&, 560, 562
 store-resultp-mux&-off, 560
 store-resultp-off, 544
 strobe-, 604, 607, 609, 611, 613, 615,
 617, 618, 620, 622, 624, 626,
 628, 630, 632, 634, 636, 638,
 640, 642, 644, 646, 648, 650,
 652, 654, 656, 658, 660, 662,
 664, 666, 668, 761–764, 782,
 784, 794, 796, 802–805, 812–
 814
 strobe-off, 604
 stub, 169, 558, 1012, 1013
 stub-guts, 169, 170, 267, 368, 387,
 388, 929, 930, 973, 1017,
 1020
 stub-right, 558
 stubp, 169–171, 267, 368, 387, 388,
 559, 929, 930, 953, 973–975,
 1017, 1019, 1020
 stubp-implies-not-fourp, 1019
 stubp-implies-not-fourp-off, 1028
 stubp-implies-not-fourp-on, 1117
 sub, 549
 subbagp, 3, 4, 8, 17–20, 29, 46, 57,
 59, 61–63, 78, 81
 subbagp-bagint1, 4
 subbagp-bagint1-off, 130
 subbagp-bagint1-off1, 1181
 subbagp-bagint1-on, 1093
 subbagp-bagint2, 4
 subbagp-bagint2-off, 130
 subbagp-bagint2-off1, 1181
 subbagp-bagint2-on, 1093
 subbagp-cdr1, 4
 subbagp-cdr1-off, 130
 subbagp-cdr1-off1, 1181
 subbagp-cdr1-on, 1093
 subbagp-cdr2, 4
 subbagp-cdr2-off, 130

- subbagp-cdr2-off1, 1181
- subbagp-cdr2-on, 1093
- subbagp-delete, 4
- subbagp-delete-off, 130
- subbagp-delete-off1, 1181
- subbagp-delete-on, 1093
- subbagp-implies-plus-tree-greate
 - reqp, 8
 - reqp-off, 8
- subbagp-off, 131
- subbagp-off1, 1182
- subbagp-on, 1094
- subbagp-subsetp, 59
- subbagp-subsetp-off, 87
- subnet, 378–380, 382–385
- subnet-off, 378
- subnet0, 378
- subnet0-off, 378
- subrange, 143–145, 165, 169, 174,
 - 175, 183, 541, 542, 606, 760,
 - 776, 779, 784, 796, 953, 971,
 - 975, 993
- subrange-0, 144
- subrange-append-left, 144
- subrange-append-right, 144
- subrange-cons, 144
- subrange-cons-off, 144
- subrange-off, 143
- subset, 138, 140, 141, 144, 167, 245,
 - 254, 256, 263, 264, 375, 376
- subset-append, 138
- subset-cons, 138
- subset-firstn, 140
- subset-nlistp, 138
- subset-off, 138
- subset-or-err, 252, 280, 281, 353, 360,
 - 366
- subset-or-err-off, 252
- subset-restn, 141
- subset-subrange, 144
- subset-x-cons-y-z, 138
- subset-x-x, 138
- subsetp, 58, 59, 62
- subsetp-implies-itimes-list-eva
 - l\$-equals-0, 59
 - l\$-equals-0-off, 65
- subsetp-off, 106
- subsetp-off1, 1158
- subsetp-on, 1069
- subtraction-cases-for-integer-i
 - n-rangep-lemmas, 1198
- subtype, 298, 300, 302
- subtype-off, 299
- sum-loading, 270
- sum-loading-off, 270
- sum-loadings, 270
- sum-loadings-off, 270
- sum-numbers, 269–271
- sum-numbers-off, 269
- t-and-f-to-1-and-0, 556
- t-buf-monotone, 949
- t-buf-monotone-off, 1045
- t-buf-monotone-on, 1134
- t-buf-preserves-good-s, 1006
- t-buf-preserves-good-s-off, 1035
- t-buf-preserves-good-s-on, 1124
- t-buf\$netlist, 408, 438
- t-buf\$value, 408
- t-buf\$value-off, 408
- t-buf&, 408, 439
- t-buf&-off, 408
- t-carry, 444–448, 503–505, 530
- t-carry*, 445
- t-carry-congruence, 445
- t-carry-off, 444
- t-carry-p-g-carry, 446
- t-carry\$netlist, 445, 514, 540
- t-carry\$value, 445
- t-carry\$value-off, 445
- t-carry&, 445, 514, 540
- t-carry&-off, 445
- t-or-err, 251, 277–279, 281–283, 285,
 - 286, 288, 289, 292, 336, 355,
 - 357–369, 372, 374, 377
- t-or-err-off, 251
- t-or-nor*, 453–455
- t-or-nor*-off, 453

t-or-nor*\$destructure, 453
 t-or-nor*\$destructure-off, 453
 t-or-nor-body, 452, 453
 t-or-nor-induction, 455, 456
 t-or-nor\$netlist, 455, 458, 461
 t-or-nor\$value, 456
 t-or-nor\$value-off, 456
 t-or-nor&, 453, 454, 456, 458, 460
 t-or-nor&-off, 455
 t-wire-error, 338, 339
 t-wire-error-off, 338
 t-wire-monotone, 949
 t-wire-monotone-off, 1044
 t-wire-monotone-on, 1134
 t-wire-preserves-good-s, 1006
 t-wire-preserves-good-s-off, 1035
 t-wire-preserves-good-s-on, 1124
 t-wire\$netlist, 409, 434, 439
 t-wire\$value, 409
 t-wire\$value-off, 409
 t-wire&, 408, 409, 434, 439
 t-wire&-off, 408
 t_fetch0, 913–915
 t_fetch1, 915, 916
 t_reada0, 914, 915
 t_readb0, 914, 915
 t_rega, 915
 t_regb, 914, 915
 t_sefa0, 914, 915
 t_sefb0, 914, 915
 t_sefb1, 913, 914
 t_update, 914, 915
 t_write0, 914, 915
 tfirstn, 146, 446–449, 451, 453, 455–
 457, 464–468, 470, 472, 503–
 505, 508–510, 514, 515
 threefix, 150, 164, 175–182, 184–200,
 204, 568–570, 740
 threefix-help-lemma, 197
 threefix-help-lemma-off, 197
 threefix-idempotent, 150
 threefix=x, 150
 threefix=x-off, 150
 threep, 150
 times-1, 134
 times-1-arg1, 14
 times-1-arg1-off, 126
 times-1-arg1-off1, 1178
 times-1-arg1-on, 1090
 times-2, 1102
 times-2-off, 1111
 times-add1, 13
 times-add1-again, 134
 times-add1-off, 127
 times-add1-off1, 1179
 times-add1-on, 1090
 times-bottom, 134
 times-commutes, 134
 times-distributes-over-differen
 ce, 13
 ce-off, 127
 ce-off1, 1178
 ce-on, 1090
 ce-proof, 13
 ce-proof-off, 127
 ce-proof-off1, 1178
 ce-proof-on, 1090
 times-distributes-over-plus, 13
 times-distributes-over-plus-off, 127
 1, 1178
 times-distributes-over-plus-on, 1090
 times-distributes-over-plus-pro
 of, 13
 of-off, 127
 of-off1, 1178
 of-on, 1090
 times-fringe, 15, 18–20, 28, 29
 times-fringe-off, 125
 times-fringe-off1, 1177
 times-fringe-on, 1089
 times-quotient, 14
 times-quotient-off, 126
 times-quotient-off1, 1178
 times-quotient-on, 1090
 times-quotient-proof, 13
 times-quotient-proof-off, 127
 times-quotient-proof-off1, 1178
 times-quotient-proof-on, 1090

times-tree, 15, 17–20, 29
 times-tree-append, 18
 times-tree-append-off, 18
 times-tree-of-times-fringe, 18
 times-tree-of-times-fringe-off, 18
 times-tree-off, 126
 times-tree-off1, 1177
 times-tree-on, 1089
 times-zero, 13
 times-zero-off, 127
 times-zero-off1, 1179
 times-zero-on, 1090
 token-less, 249
 token-less-off, 249
 too-many-restns, 141
 top-level-predicate, 354, 1026
 top-level-predicate-off, 354
 top-level-predicate-simple, 276, 1026
 tos, 551
 tos+, 553
 tos-, 552
 total-microcycles, 916, 917, 924–928,
 1014, 1021–1026
 total-microcycles-off, 916
 tr-or-nor, 456–459
 tr-or-nor-off, 456
 tr-or-nor=btr-or-nor, 457
 transfer-in-type, 303, 304
 transfer-in-type-off, 303
 transfer-loading, 309
 transfer-loading-off, 309
 transfer-loadings, 309, 310, 345
 transfer-loadings-off, 310
 transitivity-of-divides, 24
 transitivity-of-divides-off, 122
 transitivity-of-divides-off1, 1174
 transitivity-of-divides-on, 1086
 tree-height, 146, 449, 457, 458, 515
 tree-height-off, 147
 tree-number, 192, 449–461, 469, 470,
 472–474, 513–517, 527–529,
 537–541, 712, 713, 736, 738,
 747
 tree-number-off, 192
 tree-size, 146, 147, 448–450, 452, 453,
 456–459, 465–471, 473, 505–
 508, 510, 511, 513, 516, 527–
 529, 531–541
 tree-size-1-crock, 146
 tree-size-lemmas, 146
 tree-size-make-tree, 147
 tree-size-nlistp, 146
 tree-size-off, 146
 trestn, 146, 446–449, 451, 453, 456,
 457, 465, 466, 468, 470, 472,
 503–505, 509, 510, 515
 tri-state-errors, 340, 349
 tri-state-errors-off, 340
 tri-state-typep, 299, 302, 357, 358
 tri-state-typep-off, 299
 ttl-bidirect-monotone, 950
 ttl-bidirect-monotone-off, 1044
 ttl-bidirect-monotone-on, 1133
 ttl-bidirect-pads*, 756–758
 ttl-bidirect-pads*-off, 757
 ttl-bidirect-pads*\$destructure, 757
 -off, 757
 ttl-bidirect-pads-body, 756–758
 ttl-bidirect-pads-body-off, 756
 ttl-bidirect-pads-body\$inducti
 on, 756
 ttl-bidirect-pads-body\$unbound
 -in-body, 758
 -in-body-off, 758
 ttl-bidirect-pads-body\$value, 758
 ttl-bidirect-pads-body\$value-o
 ff, 759
 ttl-bidirect-pads\$netlist, 758, 791
 ttl-bidirect-pads\$value, 759
 ttl-bidirect-pads\$value-off, 759
 ttl-bidirect-pads&, 757, 759, 791
 ttl-bidirect-pads&-off, 758
 ttl-bidirect-preserves-good-s, 1007
 ttl-bidirect-preserves-good-s-o
 ff, 1035
 n, 1124
 ttl-bidirect\$netlist, 409, 758
 ttl-bidirect\$value, 409

- ttl-bidirect\$value-off, 409
- ttl-bidirect&, 409, 757, 758
- ttl-bidirect&-off, 409
- ttl-clk-input-monotone, 950
- ttl-clk-input-monotone-off, 1044
- ttl-clk-input-monotone-on, 1133
- ttl-clk-input-preserves-good-s, 1007
- ttl-clk-input-preserves-good-s-off, 1035
- on, 1124
- ttl-clk-input\$netlist, 409, 791
- ttl-clk-input\$value, 410
- ttl-clk-input\$value-off, 410
- ttl-clk-input&, 409, 410, 790
- ttl-clk-input&-off, 409
- ttl-input-monotone, 950
- ttl-input-monotone-off, 1044
- ttl-input-monotone-on, 1133
- ttl-input-pads*, 750
- ttl-input-pads*-off, 750
- ttl-input-pads*\$destructure, 750
- ttl-input-pads*\$destructure-of f, 750
- ttl-input-pads-body, 749–751
- ttl-input-pads-body-off, 749
- ttl-input-pads-body\$induction, 749, 750
- ttl-input-pads-body\$unbound-in-body, 750
- body-off, 750
- ttl-input-pads-body\$value, 751
- ttl-input-pads-body\$value-off, 751
- ttl-input-pads\$netlist, 750, 791
- ttl-input-pads\$value, 751
- ttl-input-pads\$value-off, 751
- ttl-input-pads&, 750, 751, 790
- ttl-input-pads&-off, 750
- ttl-input-preserves-good-s, 1007
- ttl-input-preserves-good-s-off, 1035
- ttl-input-preserves-good-s-on, 1124
- ttl-input\$netlist, 410, 750, 791
- ttl-input\$value, 410
- ttl-input\$value-off, 410
- ttl-input&, 410, 750, 751, 790

- ttl-input&-off, 410
- ttl-output-fast-monotone, 951
- ttl-output-fast-monotone-off, 1043
- ttl-output-fast-monotone-on, 1132
- ttl-output-fast-preserves-good-s, 1007
- s-off, 1035
- s-on, 1124
- ttl-output-fast\$netlist, 411, 791
- ttl-output-fast\$value, 411
- ttl-output-fast\$value-off, 411
- ttl-output-fast&, 411, 791
- ttl-output-fast&-off, 411
- ttl-output-monotone, 950
- ttl-output-monotone-off, 1043
- ttl-output-monotone-on, 1133
- ttl-output-pads*, 752
- ttl-output-pads*-off, 752
- ttl-output-pads*\$destructure, 752
- ttl-output-pads*\$destructure-off, 752
- ttl-output-pads-body, 751–753
- ttl-output-pads-body-off, 751
- ttl-output-pads-body\$induction, 751, 752
- ttl-output-pads-body\$unbound-in-body, 753
- n-body-off, 753
- ttl-output-pads-body\$value, 753
- ttl-output-pads-body\$value-off, 753
- ttl-output-pads\$netlist, 752, 791
- ttl-output-pads\$value, 753
- ttl-output-pads&, 752, 753, 791
- ttl-output-pads&-off, 752
- ttl-output-parametric-monotone, 951
- ttl-output-parametric-monotone-off, 1043
- on, 1132
- ttl-output-parametric-preserves-good-s, 1007
- good-s-off, 1035
- good-s-on, 1124
- ttl-output-parametric\$netlist, 410, 791
- ttl-output-parametric\$value, 410

ttl-output-parametric\$value-of
 f, 411
 ttl-output-parametric&, 410, 791
 ttl-output-parametric&-off, 410
 ttl-output-preserves-good-s, 1007
 ttl-output-preserves-good-s-off, 1035
 ttl-output-preserves-good-s-on, 1124
 ttl-output\$netlist, 410, 752, 791
 ttl-output\$value, 410
 ttl-output\$value-off, 410
 ttl-output&, 410, 752, 753, 791
 ttl-output&-off, 410
 ttl-tri-output-fast-monotone, 951
 ttl-tri-output-fast-monotone-of
 f, 1042
 ttl-tri-output-fast-monotone-on, 1132
 ttl-tri-output-fast-preserves-g
 ood-s, 1007
 ood-s-off, 1034
 ood-s-on, 1124
 ttl-tri-output-fast\$netlist, 411, 791
 ttl-tri-output-fast\$value, 411
 ttl-tri-output-fast\$value-off, 411
 ttl-tri-output-fast&, 411, 791
 ttl-tri-output-fast&-off, 411
 ttl-tri-output-monotone, 951
 ttl-tri-output-monotone-off, 1043
 ttl-tri-output-monotone-on, 1132
 ttl-tri-output-pads*, 754, 755
 ttl-tri-output-pads*-off, 754
 ttl-tri-output-pads*\$destructu
 re, 754
 re-off, 754
 ttl-tri-output-pads-body, 753–755
 ttl-tri-output-pads-body-off, 753
 ttl-tri-output-pads-body\$inducti
 on, 753, 754
 ttl-tri-output-pads-body\$unbou
 nd-in-body, 755
 nd-in-body-off, 755
 ttl-tri-output-pads-body\$value, 755
 -off, 755
 ttl-tri-output-pads\$netlist, 755, 791
 ttl-tri-output-pads\$value, 755
 ttl-tri-output-pads\$value-off, 756
 ttl-tri-output-pads&, 754, 755, 791
 ttl-tri-output-pads&-off, 755
 ttl-tri-output-preserves-good-s, 1007
 -off, 1035
 -on, 1124
 ttl-tri-output\$netlist, 411, 755
 ttl-tri-output\$value, 411
 ttl-tri-output\$value-off, 411
 ttl-tri-output&, 411, 755
 ttl-tri-output&-off, 411
 tv-adder, 446–448, 465, 507, 508
 tv-adder-as-p-g-sum, 448
 tv-adder-as-p-g-sum-off, 448
 tv-adder-off, 448
 tv-alu-help, 503–508, 511, 530, 531
 tv-alu-help*, 513, 514
 tv-alu-help*-off, 513
 tv-alu-help*\$destructure, 513
 tv-alu-help*\$destructure-off, 513
 tv-alu-help-body, 513
 tv-alu-help-induction, 514, 515
 tv-alu-help-lemma-crock, 516
 tv-alu-help-lemma-crock-off, 516
 tv-alu-help-length, 505
 tv-alu-help-off, 505
 tv-alu-help-tv-adder-works, 506
 tv-alu-help-tv-adder-works-off, 507
 tv-alu-help-tv-dec-a-works, 507
 tv-alu-help-tv-dec-a-works-off, 507
 tv-alu-help-tv-dec-b-works, 508
 tv-alu-help-tv-dec-b-works-off, 508
 tv-alu-help-tv-inc-a-works, 507
 tv-alu-help-tv-inc-a-works-off, 507
 tv-alu-help-tv-inc-b-works, 507
 tv-alu-help-tv-inc-b-works-off, 507
 tv-alu-help-tv-neg-works, 508
 tv-alu-help-tv-neg-works-off, 508
 tv-alu-help-tv-subtractor-works, 507
 -off, 507
 tv-alu-help-v-and-works, 505
 tv-alu-help-v-and-works-off, 506
 tv-alu-help-v-buf-works, 506
 tv-alu-help-v-buf-works-off, 506

tv-alu-help-v-not-works, 506
 tv-alu-help-v-not-works-off, 506
 tv-alu-help-v-or-works, 506
 tv-alu-help-v-or-works-off, 506
 tv-alu-help-v-xor-works, 506
 tv-alu-help-v-xor-works-off, 506
 tv-alu-help-zero, 505
 tv-alu-help-zero-off, 505
 tv-alu-help\$netlist, 514, 540
 tv-alu-help\$value, 516
 tv-alu-help\$value-base-case, 516
 tv-alu-help\$value-base-case-of
 f, 516
 tv-alu-help\$value-off, 517
 tv-alu-help&, 513, 514, 516, 540
 tv-alu-help&-off, 514
 tv-dec-pass, 464, 465, 467
 tv-dec-pass-crock-1, 465
 tv-dec-pass-crock-2, 465
 tv-dec-pass-length, 465
 tv-dec-pass-name, 469–473
 tv-dec-pass-ng, 466, 467, 469
 tv-dec-pass-ng*, 470, 471, 473
 tv-dec-pass-ng*-off, 471
 tv-dec-pass-ng*\$destructure, 471
 tv-dec-pass-ng*\$destructure-of
 f, 471
 tv-dec-pass-ng-body, 469, 471
 tv-dec-pass-ng-body-off, 470
 tv-dec-pass-ng-induction, 471, 472
 tv-dec-pass-ng-is-cdr-tv-dec-pa
 ss, 467
 tv-dec-pass-ng-lemma-crock, 473
 tv-dec-pass-ng-lemma-crock-off, 473
 tv-dec-pass-ng-length, 466
 tv-dec-pass-ng-length-1, 467
 tv-dec-pass-ng-off, 466
 tv-dec-pass-ng-works-1, 467
 tv-dec-pass-ng-works-2, 467
 tv-dec-pass-ng\$netlist, 473–475
 tv-dec-pass-ng\$value, 473
 tv-dec-pass-ng\$value-off, 473
 tv-dec-pass-ng&, 471, 473–475
 tv-dec-pass-ng&-off, 471
 tv-dec-pass-off, 465
 tv-dec-pass-works, 465
 tv-if*, 449–451
 tv-if*-off, 450
 tv-if*\$destructure, 450
 tv-if*\$destructure-off, 450
 tv-if-body, 448, 450
 tv-if-induction, 451
 tv-if-lemma-crock, 451
 tv-if-lemma-crock-off, 452
 tv-if\$netlist, 450, 451, 528, 714, 739,
 748, 776
 tv-if\$value, 452
 tv-if\$value-off, 452
 tv-if&, 450–452, 528, 714, 738, 748,
 776
 tv-if&-off, 450
 tv-shift-or-buf*, 527, 528
 tv-shift-or-buf*-off, 528
 tv-shift-or-buf*\$destructure, 527
 tv-shift-or-buf*\$destructure-o
 ff, 528
 tv-shift-or-buf\$netlist, 528, 541
 tv-shift-or-buf\$value, 528
 tv-shift-or-buf\$value-off, 529
 tv-shift-or-buf\$value-zero, 529
 tv-shift-or-buf\$value-zero-off, 529
 tv-shift-or-buf&, 528, 529, 540
 tv-shift-or-buf&-off, 528
 tv-zerop*, 457, 458
 tv-zerop*-off, 458
 tv-zerop*\$destructure, 457
 tv-zerop*\$destructure-off, 458
 tv-zerop\$netlist, 458, 462
 tv-zerop\$value, 459
 tv-zerop\$value-off, 459
 tv-zerop&, 458, 459, 462
 tv-zerop&-off, 458
 type-count, 386
 type-count-lessp1, 386
 type-count-lessp1-off, 388
 type-count-lessp2, 386
 type-count-lessp2-off, 388
 type-count-lessp3, 386

- type-count-lessp3-off, 388
- type-count-off, 386
- type-delay-slope, 330, 331
- type-delay-slope-off, 331
- type-drive, 314–316
- type-drive-off, 314
- type-loading, 313
- type-loading-off, 313
- type-value, 299, 300, 304
- type-value-off, 299
- type-value-simple, 258
- type-value-simple-off, 258
- type-value0, 299
- type-value0-off, 299
- types-acceptable, 262–264
- types-acceptable-off, 263
- types-compatible, 299, 302, 384
- types-compatible-off, 299

- ucar, 355, 359, 362–364, 372, 376
- ucar-off, 355
- ucdr, 356, 359, 362–364, 372, 376
- ucdr-off, 356
- un-fm9001, 559
- unary-op-code-p, 485, 486, 549, 564, 673, 716, 724, 725, 727, 828, 831, 835, 842, 893, 895, 898, 899, 902, 914, 915
- unary-op-code-p*, 564
- unary-op-code-p->v-alu=v-alu-1-off, 486
- 1, 486
- unary-op-code-p-off, 485
- unary-op-code-p-op-code->v-alu=v-alu-1, 549
- unary-op-code-p\$netlist, 564, 672
- unary-op-code-p\$value, 564
- unary-op-code-p\$value-off, 564
- unary-op-code-p&, 564, 672
- unary-op-code-p&-off, 564
- unbind, 247, 248, 298, 300, 308
- unbind-list, 248, 352
- unbind-list-off, 248
- unbind-netlist-props, 353

- unbind-netlist-props-0, 352, 353
- unbind-netlist-props-0-off, 353
- unbind-netlist-props-off, 353
- unbind-off, 248
- unbound-in-body, 426, 427, 429–431, 433, 434, 442, 477, 480, 750, 753, 755, 758
- unbound-in-body-dual-eval-1, 427
- unbound-in-body-listp, 427
- unbound-in-body-nlistp, 427
- unbound-in-body-off, 426
- unbound-keys, 247, 252
- unbound-keys-off, 247
- union-values, 247, 334
- union-values-off, 247
- universal-flags-interpretations, 1188
- unknown, 295, 300, 302, 304, 311, 314–316, 326–329, 332, 333, 355, 356, 371, 373, 375, 376
- unknown-args, 370, 371
- unknown-args-off, 370
- unknown-machine-state, 1011, 1012, 1016, 1019–1021, 1024
- unknown-machine-state-off, 1032
- unknown-machine-state-on, 1121
- unknown-memory-state, 1012
- unknown-memory-state-off, 1032
- unknown-memory-state-on, 1121
- unknown-regfile, 1011
- unknown-regfile-off, 1032
- unknown-regfile-on, 1121
- unknown-state, 1012, 1014
- unknown-state-off, 1032
- unknown-state-okp, 1014
- unknown-state-okp-off, 1031
- unknown-state-okp-on, 1120
- unknown-state-on, 1121
- unknownp, 300, 302, 325, 356, 358–360, 362, 364–367, 370, 373, 376, 384
- unmark-io-out, 296, 297, 299, 300, 302
- unmark-io-out-off, 296
- unmark-io-outs, 296, 347

- unmark-io-outs-off, 296
- update->update\$sim, 898
- update-delays, 323, 342
- update-delays-off, 323
- update-drives, 309, 342
- update-drives-off, 309
- update-flags, 544, 545, 548, 591, 745, 746, 834, 857, 897, 899, 907, 1186, 1188–1196, 1200–1216
- update-flags-off, 544
- update-in-types, 301, 342
- update-in-types-off, 301
- update-known-types, 259, 261, 262
- update-known-types-off, 259
- update-list, 556
- update-loadings, 308, 342
- update-loadings-off, 308
- update-lsi-occ-syntax-data, 290, 291
- update-lsi-occ-syntax-data-off, 290
- update-nth, 145, 160
- update-nth-off, 145
- update-occ-bindings, 342–344
- update-occ-bindings-off, 343
- update-occ-syntax-data, 280, 281, 290
- update-occ-syntax-data-off, 280
- update-oprop-binding, 341, 343
- update-oprop-binding-off, 342
- update-out-depends, 335, 342
- update-out-depends-off, 335
- update-out-types, 301, 342
- update-out-types-off, 301
- update-state-types, 336, 342
- update-state-types-off, 336
- update-tri-state-data, 337, 342
- update-tri-state-data-off, 338
- update-v-nth, 160
- update-v-nth-off, 160
- update\$step, 833
- v, 482–484, 532, 544, 742–744, 1216, 1217
- v-adder, 162–164
- v-adder-carry-out, 162, 163, 446, 483, 1100, 1104
- v-adder-carry-out=v-carry, 446
- v-adder-carry-out=v-carry-off, 1095
- v-adder-carry-out=v-carry-on, 1183
- v-adder-off, 162
- v-adder-output, 162, 163, 446, 483, 1100, 1104, 1217
- v-adder-output=v-sum, 446
- v-adder-output=v-sum-off, 1095
- v-adder-output=v-sum-off1, 1185
- v-adder-output=v-sum-on, 1182
- v-adder-overflowp, 162, 163, 483, 1099, 1104
- v-adder-works, 164
- v-alu, 484–486, 532–536, 545, 549, 834, 857, 897, 899, 907, 908, 1101, 1102, 1106, 1188–1196, 1200–1219
- v-alu-1, 485, 486, 549
- v-alu-1-off, 485
- v-alu-correct-int, 1106
- v-alu-correct-int-adder, 1104
- v-alu-correct-int-adder-off, 1107
- v-alu-correct-int-adder-on, 1184
- v-alu-correct-int-adder-output, 1104
- v-alu-correct-int-adder-output-off, 1107
- on, 1184
- v-alu-correct-int-adder-overflowp, 1104
- wp-off, 1107
- wp-on, 1184
- v-alu-correct-int-asr, 1105
- v-alu-correct-int-asr-off, 1108
- v-alu-correct-int-asr-off1, 1186
- v-alu-correct-int-asr-on, 1185
- v-alu-correct-int-asr-output, 1105
- v-alu-correct-int-asr-output-of-f, 1108
- v-alu-correct-int-asr-output-on, 1185
- v-alu-correct-int-dec, 1105
- v-alu-correct-int-dec-off, 1108
- v-alu-correct-int-dec-on, 1185
- v-alu-correct-int-inc, 1104
- v-alu-correct-int-inc-off, 1107

v-alu-correct-int-inc-on, 1185
v-alu-correct-int-neg, 1105
v-alu-correct-int-neg-off, 1108
v-alu-correct-int-neg-on, 1185
v-alu-correct-int-not, 1106
v-alu-correct-int-not-off, 1108
v-alu-correct-int-not-on, 1185
v-alu-correct-int-not-output, 1106
v-alu-correct-int-not-output-of
 f, 1108
 f1, 1185
v-alu-correct-int-off, 1108
v-alu-correct-int-rewrite, 1218
v-alu-correct-int-rewrite-off, 1218
v-alu-correct-int-subtractor, 1105
v-alu-correct-int-subtractor-of
 f, 1108
v-alu-correct-int-subtractor-on, 1185
v-alu-correct-int-subtractor-outp
 ut, 1105
 ut-off, 1108
 ut-on, 1185
v-alu-correct-int-subtractor-ove
 rflowp, 1105
 rflowp-lemma, 1103
 rflowp-lemma-off, 1110
 rflowp-off, 1107
 rflowp-on, 1185
v-alu-correct-nat, 1102
v-alu-correct-nat-adder, 1100
v-alu-correct-nat-adder-carry-o
 ut, 1100
 ut-off, 1106
 ut-on, 1183
v-alu-correct-nat-adder-off, 1106
v-alu-correct-nat-adder-off1, 1186
v-alu-correct-nat-adder-on, 1183
v-alu-correct-nat-adder-output, 1100
v-alu-correct-nat-adder-output-
 off, 1106
 on, 1183
v-alu-correct-nat-dec, 1101
v-alu-correct-nat-dec-off, 1107
v-alu-correct-nat-dec-on, 1184
v-alu-correct-nat-inc, 1100
v-alu-correct-nat-inc-off, 1106
v-alu-correct-nat-inc-off1, 1186
v-alu-correct-nat-inc-on, 1184
v-alu-correct-nat-int-buf, 1096
v-alu-correct-nat-int-buf-off, 1106
v-alu-correct-nat-int-buf-on, 1183
v-alu-correct-nat-lsr, 1101
v-alu-correct-nat-lsr-carry-out, 1101
 -off, 1107
 -off1, 1185
 -on, 1184
v-alu-correct-nat-lsr-off, 1107
v-alu-correct-nat-lsr-off1, 1186
v-alu-correct-nat-lsr-on, 1184
v-alu-correct-nat-lsr-output, 1101
v-alu-correct-nat-lsr-output-of
 f, 1107
v-alu-correct-nat-lsr-output-on, 1184
v-alu-correct-nat-not, 1101
v-alu-correct-nat-not-off, 1107
v-alu-correct-nat-not-on, 1184
v-alu-correct-nat-not-output, 1101
v-alu-correct-nat-not-output-of
 f, 1103
 f1, 1107
 f2, 1185
v-alu-correct-nat-not-output-on, 1184
v-alu-correct-nat-off, 1111
v-alu-correct-nat-rewrite, 1217
v-alu-correct-nat-rewrite-off, 1218
v-alu-correct-nat-subtractor, 1100
v-alu-correct-nat-subtractor-ca
 rry-out, 1100
 rry-out-off, 1106
 rry-out-on, 1184
v-alu-correct-nat-subtractor-of
 f, 1107
v-alu-correct-nat-subtractor-on, 1184
v-alu-correct-nat-subtractor-outp
 ut, 1100
 ut-lemma, 1099
 ut-lemma-off, 1113
 ut-off, 1107

- ut-on, 1184
- v-alu-int, 1106, 1218
- v-alu-int-adder, 1104, 1106
- v-alu-int-adder-off, 1110
- v-alu-int-adder-output, 1104
- v-alu-int-adder-output-off, 1110
- v-alu-int-adder-overflowp, 1104
- v-alu-int-adder-overflowp-off, 1110
- v-alu-int-asr, 1105, 1106
- v-alu-int-asr-off, 1109
- v-alu-int-asr-output, 1105
- v-alu-int-asr-output-off, 1109
- v-alu-int-dec, 1105, 1106
- v-alu-int-dec-off, 1109
- v-alu-int-inc, 1104, 1106
- v-alu-int-inc-off, 1109
- v-alu-int-neg, 1105, 1106
- v-alu-int-neg-off, 1109
- v-alu-int-not, 1106
- v-alu-int-not-off, 1108
- v-alu-int-not-output, 1105, 1106
- v-alu-int-not-output-off, 1108
- v-alu-int-off, 1108
- v-alu-int-subtractor, 1105, 1106
- v-alu-int-subtractor-off, 1109
- v-alu-int-subtractor-output, 1104, 1105
- v-alu-int-subtractor-output-off, 1109
- v-alu-int-subtractor-overflowp, 1104, 1105
- v-alu-int-subtractor-overflowp-off, 1109
- v-alu-nat, 1101, 1102, 1217
- v-alu-nat-adder, 1099–1101
- v-alu-nat-adder-carry-out, 1099, 1100
- v-alu-nat-adder-carry-out-off, 1113
- v-alu-nat-adder-off, 1113
- v-alu-nat-adder-output, 1099, 1100
- v-alu-nat-adder-output-off, 1113
- v-alu-nat-dec, 1100, 1101
- v-alu-nat-dec-off, 1112
- v-alu-nat-inc, 1100, 1101
- v-alu-nat-inc-off, 1113
- v-alu-nat-int-buf, 1096, 1101, 1106
- v-alu-nat-int-buf-off, 1115
- v-alu-nat-lsr, 1101
- v-alu-nat-lsr-carry-out, 1101, 1105
- v-alu-nat-lsr-carry-out-off, 1112
- v-alu-nat-lsr-off, 1112
- v-alu-nat-lsr-output, 1101
- v-alu-nat-lsr-output-off, 1112
- v-alu-nat-not, 1101
- v-alu-nat-not-off, 1111
- v-alu-nat-not-output, 1101
- v-alu-nat-not-output-off, 1112
- v-alu-nat-off, 1111
- v-alu-nat-subtractor, 1100, 1101
- v-alu-nat-subtractor-carry-out, 1100
- v-alu-nat-subtractor-carry-out-off, 1112
- v-alu-nat-subtractor-off, 1112
- v-alu-nat-subtractor-output, 1100
- v-alu-nat-subtractor-output-off, 1113
- v-alu-off, 484
- v-and, 156–158, 484, 505, 1217
- v-and-append-help, 158
- v-and-off, 156
- v-append-propagate, 445
- v-approx, 928, 929, 931, 932, 953, 954, 972–976, 1002, 1008–1010, 1017
- v-approx-alist, 930–933, 1002
- v-approx-alist-append, 1002
- v-approx-alist-append-off, 1040
- v-approx-alist-append-on, 1129
- v-approx-alist-implies-b-approx-value, 1002
- v-approx-alist-implies-b-approx-value-off, 1040
- v-approx-alist-implies-b-approx-value-on, 1130
- v-approx-alist-implies-v-approx-collect-value, 1002
- v-approx-alist-implies-v-approx-collect-value-off, 1040
- v-approx-alist-implies-v-approx-collect-value-on, 1129
- v-approx-alist-off, 1056
- v-approx-alist-on, 1145
- v-approx-bvp, 953
- v-approx-bvp-off, 993
- v-approx-bvp-off1, 1148
- v-approx-bvp-on, 1059

- v-approx-bvp-subrange, 953
- v-approx-bvp-subrange-off, 994
- v-approx-bvp-subrange-off1, 1149
- v-approx-bvp-subrange-on, 1059
- v-approx-car-nth, 1009
- v-approx-car-nth-off, 1033
- v-approx-car-nth-on, 1123
- v-approx-implies-b-approx-nth, 953
- v-approx-implies-b-approx-nth-off, 994
- v-approx-implies-b-approx-nth-off1, 1148
- v-approx-implies-b-approx-nth-on, 1059
- v-approx-implies-nth-does-not-g-o-from-f-to-t, 973
- v-approx-implies-nth-does-not-g-o-from-f-to-t-off, 996
- v-approx-implies-nth-does-not-g-o-from-f-to-t-off1, 1150
- v-approx-implies-nth-does-not-g-o-from-f-to-t-on, 1061
- v-approx-implies-subranges-equal, 953
- v-approx-implies-subranges-equal-off, 994
- v-approx-implies-subranges-equal-off1, 1149
- v-approx-implies-subranges-equal-on, 1059
- v-approx-implies-v-approx-alist, 1002
- v-approx-implies-v-approx-alist-off, 1040
- v-approx-implies-v-approx-alist-on, 1129
- v-approx-length, 974
- v-approx-length-off, 996
- v-approx-length-off1, 1150
- v-approx-length-on, 1061
- v-approx-list, 1008–1010
- v-approx-list-off, 1034
- v-approx-list-on, 1123
- v-approx-list-x-x, 1010
- v-approx-list-x-x-off, 1033
- v-approx-list-x-x-on, 1122
- v-approx-make-list-x, 954
- v-approx-make-list-x-off, 994
- v-approx-make-list-x-off1, 1149
- v-approx-make-list-x-on, 1059
- v-approx-off, 972
- v-approx-off1, 1057
- v-approx-on, 995
- v-approx-on1, 1146
- v-approx-preserves-length, 975
- v-approx-preserves-length-off, 996
- v-approx-preserves-length-off1, 1151
- v-approx-preserves-length-on, 1061
- v-approx-preserves-properp, 976
- v-approx-preserves-properp-off, 997
- v-approx-preserves-properp-off1, 1151
- v-approx-preserves-properp-on, 1062
- v-approx-subrange, 975
- v-approx-subrange-off, 996
- v-approx-subrange-off1, 1151
- v-approx-subrange-on, 1061
- v-approx-x-x, 953
- v-approx-x-x-off, 994
- v-approx-x-x-off1, 1042
- v-approx-x-x-on, 1001
- v-approx-x-x-on1, 1131
- v-asr, 157, 158, 484, 526, 1105, 1217
- v-buf, 156–159, 163, 465, 467, 484, 506, 1096, 1217
- v-buf*, 428, 429
- v-buf*-off, 428
- v-buf*\$destructure, 428
- v-buf*\$destructure-off, 428
- v-buf-append-help, 159
- v-buf-off, 156
- v-buf-works, 163
- v-buf\$body, 428, 429
- v-buf\$body-off, 428
- v-buf\$body-value, 429
- v-buf\$body-value-off, 429
- v-buf\$netlist, 429, 514
- v-buf\$unbound-in-body, 429
- v-buf\$unbound-in-body-off, 429
- v-buf\$value, 429
- v-buf\$value-off, 429
- v-buf&, 428, 429, 514
- v-buf&-off, 429
- v-carry, 444, 446
- v-carry-off, 444
- v-cvzbv, 1216
- v-dec, 163, 465, 467, 475, 486, 536, 545, 546, 836, 846, 857, 901, 902, 905, 907, 908, 1217
- v-dec-off, 163

- v-equal*, 461, 462
- v-equal*-off, 462
- v-equal*\$destructure, 461
- v-equal*\$destructure-off, 462
- v-equal\$netlist, 462, 739
- v-equal\$value, 462
- v-equal\$value-off, 462
- v-equal&, 462, 738
- v-equal&-off, 462
- v-flag, 543, 544, 559, 744, 1186
- v-flag-off, 543
- v-fourfix, 165, 204
- v-fourfix-make-list, 165
- v-fourfix-off, 165
- v-fourp, 1017, 1018, 1020
- v-fourp-implies-properp, 1018
- v-fourp-implies-properp-off, 1029
- v-fourp-implies-properp-on, 1118
- v-fourp-is-good-s, 1020
- v-fourp-is-good-s-off, 1028
- v-fourp-is-good-s-on, 1117
- v-fourp-off, 1029
- v-fourp-on, 1118
- v-generate, 444–446, 448
- v-generate-off, 444
- v-if, 157–159, 163, 526, 566, 716
- v-if-append-help, 159
- v-if-c-congruence, 159
- v-if-f-4*, 567
- v-if-f-4\$netlist, 567, 714
- v-if-f-4\$reset-value, 568
- v-if-f-4\$reset-value-off, 568
- v-if-f-4\$value, 568
- v-if-f-4&, 567, 568, 714
- v-if-f-4&-off, 567
- v-if-off, 157
- v-if-works, 163
- v-iff, 165, 172, 173, 733
- v-iff-off, 165
- v-iff-rev1, 165
- v-iff-reverse, 165
- v-iff-v-addr1-v-addr2-read-mem1
 - write-mem1, 172
 - write-mem1-not-ram, 173
- v-iff-x-x, 165
- v-iff=equal, 165
- v-inc, 163, 464, 486, 536, 545, 546, 549, 667, 822, 823, 893, 1217
- v-inc-off, 163
- v-inc4, 463, 464
- v-inc4*, 463, 464
- v-inc4\$netlist, 463, 464, 714
- v-inc4\$value, 464
- v-inc4\$value-as-v-inc, 464
- v-inc4\$value-off, 464
- v-inc4&, 463, 464, 714
- v-inc4&-off, 463
- v-knownnp, 929, 930, 953, 1010, 1011, 1016
- v-knownnp-implies-v-approx-is-equal, 929
 - ual-off, 1056
 - ual-on, 1146
- v-knownnp-off, 972
- v-knownnp-off1, 1056
- v-knownnp-on, 995
- v-knownnp-on1, 1146
- v-lsr, 157, 158, 484, 527, 1101, 1217
- v-neg, 1216, 1217
- v-negp, 161, 482, 1186, 1188–1190, 1197–1200, 1208
- v-negp->negativep-v-to-int, 1197
- v-negp->negativep-v-to-int-of-f, 1197
- v-negp->v-nzerop, 1186
- v-negp-as-bounds, 1197
- v-negp-as-bounds-off, 1197
- v-negp-as-nth, 161
- v-negp-as-nth-off, 161
- v-negp-off, 161
- v-negp-v-shift-right, 1190
- v-not, 156–159, 162, 163, 444, 465, 483, 506–508, 549, 1099, 1101–1103, 1106, 1186, 1189, 1217
- v-not-append-help, 159
- v-not-firstn, 159
- v-not-inverts-all, 159
- v-not-off, 156

v-not-restn, 159
v-not-v-not, 1186
v-nth, 160
v-nth-off, 160
v-nzerop, 160, 161, 457, 1186, 1188,
1189, 1193, 1200, 1208–1210
v-nzerop-as-or-crock, 161
v-nzerop-int-to-v, 1200
v-nzerop-int-to-v-0, 1200
v-nzerop-nat-to-v, 1193
v-nzerop-nat-to-v-zero, 1186
v-nzerop-nat-to-v\$help, 1193
v-nzerop-nat-to-v\$help-off, 1193
v-nzerop-off, 160
v-off, 482
v-or, 156–159, 202, 484, 506, 718,
719, 1217
v-or*, 429, 430
v-or*-off, 430
v-or*\$destructure, 430
v-or*\$destructure-off, 430
v-or-append-help, 158
v-or-crock-for-f\$next-cntl-state, 718
-off, 719
v-or-make-list-f, 159
v-or-off, 156
v-or\$body, 429, 430
v-or\$body-off, 429
v-or\$body-value, 430
v-or\$body-value-off, 430
v-or\$netlist, 430, 721
v-or\$unbound-in-body, 430
v-or\$unbound-in-body-off, 430
v-or\$value, 430
v-or\$value-off, 431
v-or&, 430, 721
v-or&-off, 430
v-propagate, 444–446, 448
v-propagate-append, 445
v-propagate-append-off, 445
v-propagate-off, 444
v-pullup, 204, 433, 762–764, 802–
805, 812–814
v-pullup*, 432, 433
v-pullup*-off, 433
v-pullup*\$destructure, 432
v-pullup*\$destructure-off, 433
v-pullup-bvp, 204
v-pullup-make-list-z, 204
v-pullup-off, 204
v-pullup\$body, 432, 433
v-pullup\$body-off, 432
v-pullup\$body-value, 433
v-pullup\$body-value-off, 433
v-pullup\$netlist, 433, 800
v-pullup\$unbound-in-body, 433
v-pullup\$unbound-in-body-off, 433
v-pullup\$value, 433
v-pullup\$value-off, 433
v-pullup&, 433, 799
v-pullup&-off, 433
v-ror, 157, 158, 483, 527, 1217
v-s-approx-list, 1008–1010
v-s-approx-list-off, 1034
v-s-approx-list-on, 1123
v-set, 542, 544, 591, 742–744, 1200,
1202–1213, 1215, 1216
v-set-off, 542
v-shift-right, 157, 158, 204, 526, 1190
v-shift-right-names, 523, 527, 528
v-shift-right-off, 157
v-subtractor-carry-out, 163, 483, 1100,
1105
v-subtractor-output, 162, 163, 483,
549, 1100, 1105, 1189, 1216,
1217
v-subtractor-overflowp, 163, 483, 1100,
1105
v-sum, 443, 444, 446, 448, 1189
v-sum-congruence, 444
v-sum-off, 443
v-sum-on-not-a-a, 1189
v-threefix, 164, 165, 175–182, 184–
191, 203–205, 429, 478, 481,
503, 515, 516, 523, 739, 740,
744, 751, 753, 758, 759, 761–
764, 781–784, 792–794, 796,
801–805, 812–814, 816, 817,

821–823, 825–827, 829, 831,
 833, 835, 837, 839–841, 843,
 845, 847, 849, 851, 852, 854–
 856, 858, 867, 869, 871, 873,
 875–881, 883–885, 889, 890,
 894, 896, 898, 899, 902, 905,
 910, 912, 913
 v-threefix-append, 164
 v-threefix-append-off, 164
 v-threefix-bvp, 164
 v-threefix-fv-if, 203
 v-threefix-fv-shift-right, 205
 v-threefix-idempotence, 164
 v-threefix-make-list-x, 164
 v-threefix-off, 164
 v-threefix-v-fourfix, 165
 v-to-0s-through-fs, 557
 v-to-hex, 557, 558
 v-to-hex-all, 558, 559
 v-to-int, 1102–1105, 1196–1198, 1200–
 1216, 1219
 v-to-int-nat-to-v-0, 1197
 v-to-int-off, 1111
 v-to-int-v-not, 1102
 v-to-int-v-not-off, 1110
 v-to-nat, 159, 160, 164, 192, 549,
 557, 593–602, 1096–1102,
 1186, 1187, 1191–1197, 1218
 v-to-nat-all, 557, 559
 v-to-nat-append, 1099
 v-to-nat-append-off, 1113
 v-to-nat-equal, 1186
 v-to-nat-firstn, 1097
 v-to-nat-firstn-hack1, 1097
 v-to-nat-firstn-hack1-off, 1097
 v-to-nat-firstn-hack1-on, 1183
 v-to-nat-firstn-hack2, 1097
 v-to-nat-firstn-hack2-off, 1097
 v-to-nat-firstn-hack2-on, 1183
 v-to-nat-firstn-off, 1115
 v-to-nat-lessp-exp-2-length, 1196
 v-to-nat-of-nat-to-v, 1097
 v-to-nat-of-nat-to-v-hack, 1097
 v-to-nat-of-nat-to-v-hack-off, 1115
 v-to-nat-of-nat-to-v-off, 1114
 v-to-nat-of-nat-to-v-on, 1186
 v-to-nat-off, 159
 v-to-nat-v-not, 1099
 v-to-nat-v-not-lemma, 1099
 v-to-nat-v-not-lemma-off, 1114
 v-to-nat-v-not-off, 1114
 v-v-alu, 1217
 v-wire, 203, 204, 435, 759, 762–764,
 793, 794, 802, 803, 805, 812–
 814
 v-wire*, 434
 v-wire*-off, 434
 v-wire*\$destructure, 434
 v-wire*\$destructure-off, 434
 v-wire-make-list-x, 204
 v-wire-make-list-z, 204
 v-wire-off, 204
 v-wire-x-x=x, 204
 v-wire\$body, 433, 434
 v-wire\$body-off, 434
 v-wire\$body-value, 434
 v-wire\$body-value-off, 435
 v-wire\$netlist, 434, 800
 v-wire\$unbound-in-body, 434
 v-wire\$unbound-in-body-off, 434
 v-wire\$value, 435
 v-wire\$value-off, 435
 v-wire&, 434, 435, 800
 v-wire&-off, 434
 v-xor, 156–158, 161, 203, 484, 506,
 1217
 v-xor*, 431
 v-xor*-off, 431
 v-xor*\$destructure, 431
 v-xor*\$destructure-off, 431
 v-xor-append-help, 158
 v-xor-nzerop=not-equal, 161
 v-xor-off, 156
 v-xor\$body, 431, 432
 v-xor\$body-off, 431
 v-xor\$body-value, 432
 v-xor\$body-value-off, 432
 v-xor\$netlist, 431, 462

v_xor\$unbound-in-body, 431
 v_xor\$unbound-in-body-off, 432
 v_xor\$value, 432
 v_xor\$value-off, 432
 v_xor&, 431, 432, 462
 v_xor&-off, 431
 v_zerop, 161, 457, 459, 482, 531, 1186–
 1188, 1197, 1216
 v_zerop-implies-v-to-nat-0, 1186
 v_zerop-make-list-f, 161
 v_decode, 594, 603, 616, 724
 v_decode-off, 594
 v_fetch0, 592, 603, 608, 723
 v_fetch0-off, 593
 v_fetch1, 593, 603, 610, 723
 v_fetch1-off, 593
 v_fetch2, 593, 603, 612, 723
 v_fetch2-off, 593
 v_fetch3, 593, 603, 614, 723
 v_fetch3-off, 593
 v_hold0, 600, 604, 655, 730
 v_hold0-off, 600
 v_hold1, 600, 604, 657, 731
 v_hold1-off, 600
 v_reada0, 595, 603, 623, 725
 v_reada0-off, 595
 v_reada1, 595, 603, 625, 726
 v_reada1-off, 595
 v_reada2, 596, 603, 627, 726
 v_reada2-off, 596
 v_reada3, 596, 603, 629, 726
 v_reada3-off, 596
 v_readb0, 596, 603, 631, 727
 v_readb0-off, 596
 v_readb1, 596, 603, 633, 727
 v_readb1-off, 597
 v_readb2, 597, 603, 635, 727
 v_readb2-off, 597
 v_readb3, 597, 603, 637, 728
 v_readb3-off, 597
 v_rega, 594, 603, 617, 724
 v_rega-off, 594
 v_regb, 594, 603, 619, 725
 v_regb-off, 594
 v_reset0, 601, 604, 663, 732
 v_reset0-off, 601
 v_reset1, 601, 604, 665, 732
 v_reset1-off, 601
 v_reset2, 602, 604, 666, 732
 v_reset2-off, 602
 v_sefa0, 599, 604, 647, 729
 v_sefa0-off, 599
 v_sefa1, 599, 604, 649, 730
 v_sefa1-off, 599
 v_sefb0, 599, 604, 651, 730
 v_sefb0-off, 599
 v_sefb1, 600, 604, 653, 730
 v_sefb1-off, 600
 v_update, 595, 603, 621, 725
 v_update-off, 595
 v_v11010, 600, 604, 659, 731
 v_v11010-off, 601
 v_v11011, 601, 604, 661, 731
 v_v11011-off, 601
 v_v11111, 602, 604, 668, 716, 733
 v_v11111-off, 602
 v_write0, 597, 603, 639, 728
 v_write0-off, 597
 v_write1, 598, 603, 641, 728
 v_write1-off, 598
 v_write2, 598, 603, 643, 729
 v_write2-off, 598
 v_write3, 598, 603, 604, 645, 729
 v_write3-off, 598
 value, 166–168, 238–243, 247, 259,
 268, 270–272, 280, 281, 287–
 291, 293, 295, 297, 299–301,
 307–309, 312, 313, 323, 324,
 327, 332, 335–339, 342–345,
 350, 358–360, 362, 364–367,
 370, 373, 375, 377, 379, 380,
 382, 383, 385, 388, 427, 442,
 443, 478, 481, 523, 755, 758,
 759, 999–1003
 value-append-pairlist, 166
 value-indices-hack, 166
 value-off, 166

- value-or-unknown, 295, 298, 316, 333, 336, 375
- value-or-unknown-off, 295
- value-pairlist, 166
- value-pairlist-off, 166
- value2, 297, 308, 321, 337
- value2-off, 297
- vdd, 156
- vdd-monotone, 952
- vdd-monotone-off, 1042
- vdd-monotone-on, 1131
- vdd-parametric-monotone, 952
- vdd-parametric-monotone-off, 1042
- vdd-parametric-monotone-on, 1131
- vdd-parametric-preserves-good-s, 1007
 - off, 1034
 - on, 1124
- vdd-parametric\$netlist, 412, 791
- vdd-parametric\$value, 412
- vdd-parametric\$value-off, 412
- vdd-parametric&, 412, 790
- vdd-parametric&-off, 412
- vdd-preserves-good-s, 1007
- vdd-preserves-good-s-off, 1034
- vdd-preserves-good-s-on, 1124
- vdd\$netlist, 412
- vdd\$value, 412
- vdd\$value-off, 412
- vdd&, 412
- vdd&-off, 412
- vector-module-induction, 428
- vector-state-theory, 602
- vft-buf, 205, 755, 756, 759, 761–764, 793, 794, 796, 802–805, 811–814
- vft-buf-lemmas, 205
- vft-buf-off, 205
- vft-buf-rewrite, 205
- vft-buf-rewrite-off, 205
- vss, 156, 675, 676, 678
- vss-monotone, 952
- vss-monotone-off, 1042
- vss-monotone-on, 1131
- vss-preserves-good-s, 1007
- vss-preserves-good-s-off, 1034
- vss-preserves-good-s-on, 1123
- vss\$netlist, 412, 688
- vss\$value, 412
- vss\$value-off, 412
- vss&, 412, 688
- vss&-off, 412
- we-a-reg, 605, 607, 609, 611, 613, 615, 617, 618, 620, 622, 624, 626, 628, 630, 632, 634, 636, 638, 641, 643, 645, 646, 648, 650, 653, 655, 656, 658, 660, 662, 664, 666, 668, 761, 781, 793, 802, 811
- we-a-reg-off, 605
- we-addr-out, 605, 607, 609, 611, 613, 615, 617, 619, 620, 622, 624, 627, 628, 630, 632, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 658, 660, 662, 664, 666, 668, 763, 782, 794, 804, 813
- we-addr-out-off, 605
- we-b-reg, 605, 607, 609, 611, 613, 615, 617, 618, 620, 622, 624, 626, 628, 630, 632, 634, 636, 638, 641, 643, 645, 646, 649, 651, 653, 655, 656, 658, 660, 662, 664, 666, 668, 762, 781, 793, 802, 812
- we-b-reg-off, 605
- we-data-out, 605, 607, 609, 611, 613, 615, 617, 618, 620, 622, 624, 626, 628, 630, 632, 635, 637, 639, 641, 643, 645, 647, 649, 651, 653, 655, 657, 658, 660, 662, 664, 666, 668, 763, 782, 794, 803, 813
- we-data-out-off, 605
- we-flags, 606, 608, 610, 612, 614, 616, 617, 619, 621, 623, 625, 627, 629, 631, 633, 635, 637, 639, 641, 643, 645, 647, 649,

651, 653, 655, 657, 659, 661,
 663, 665, 666, 668, 761, 781,
 792, 802, 811
 we-flags-off, 606
 we-hold-, 605, 608, 609, 611, 613,
 615, 617, 619, 620, 622, 625,
 627, 628, 630, 632, 635, 637,
 639, 641, 643, 645, 647, 649,
 651, 653, 655, 657, 659, 660,
 662, 664, 666, 668, 763, 782,
 794, 804, 813
 we-hold-off, 605
 we-i-reg, 605, 607, 609, 611, 613,
 615, 617, 618, 620, 622, 624,
 626, 628, 630, 632, 635, 637,
 638, 641, 643, 645, 647, 649,
 651, 653, 655, 656, 658, 660,
 662, 664, 666, 668, 762, 782,
 793, 803, 812
 we-i-reg-off, 605
 we-pc-reg, 605, 608, 609, 611, 613,
 615, 617, 619, 620, 622, 625,
 627, 629, 630, 632, 635, 637,
 639, 641, 643, 645, 647, 649,
 651, 653, 655, 657, 659, 660,
 662, 664, 666, 668, 763, 782,
 794, 804, 813
 we-pc-reg-off, 605
 we-reg*, 479, 480
 we-reg*-off, 479
 we-reg*\$destructure, 479
 we-reg*\$destructure-off, 480
 we-reg-body, 478–481
 we-reg-body-induction, 480, 481
 we-reg-body-off, 479
 we-reg-body\$all-unbound-in-bod
 y, 480
 y-off, 480
 we-reg-body\$state, 481
 we-reg-body\$state-off, 482
 we-reg-body\$unbound-in-body, 480
 we-reg-body\$unbound-in-body-of
 f, 480
 we-reg-body\$value, 481
 we-reg-body\$value-off, 481
 we-reg\$netlist, 480, 739, 776
 we-reg\$state, 482
 we-reg\$state-off, 482
 we-reg\$value, 481
 we-reg\$value-off, 481
 we-reg&, 480–482, 738, 775
 we-reg&-off, 480
 we-regs, 604, 607, 609, 611, 613, 615,
 617, 618, 620, 622, 624, 626,
 628, 630, 632, 634, 636, 638,
 640, 643, 644, 646, 648, 650,
 653, 655, 656, 658, 660, 662,
 664, 666, 668, 761, 781, 792,
 801, 811
 we-regs-off, 605
 when-the-mantissa-isnt-0-then-neit
 her-is-the-exponential, 1193
 write-mem, 170, 172–175, 179, 188,
 545, 546, 733, 734, 819, 891,
 974, 975, 993, 1007
 write-mem-monotone, 974
 write-mem-monotone-off, 996
 write-mem-monotone-off1, 1150
 write-mem-monotone-on, 1061
 write-mem-off, 171
 write-mem1, 170, 172–174, 973–975,
 1007
 write-mem1-double-induction, 974
 write-mem1-double-induction-off, 1041
 write-mem1-double-induction-on, 1130
 write-mem1-monotone, 974
 write-mem1-monotone-induction, 974
 write-mem1-monotone-induction-o
 ff, 1041
 n, 1130
 write-mem1-monotone-off, 996
 write-mem1-monotone-off1, 1150
 write-mem1-monotone-on, 1061
 write-mem1-off, 170
 write-mem1-opener, 973
 write-mem1-opener-off, 996
 write-mem1-opener-off1, 1150
 write-mem1-opener-on, 1061

write-regs, 733, 734, 741, 815, 817,
 818, 820, 822–826, 828, 830,
 832, 834, 836, 838, 840–842,
 844–846, 848, 850, 851, 853,
 855, 857, 859, 861, 863–866,
 869, 870, 873–875, 877–885,
 887, 888, 890–893, 895, 897,
 899, 901, 902, 904, 906, 907,
 909, 911, 913
 write-regs-f-bv-crock, 733
 write-regs-f-write-regs-f, 733
 write-regs-if, 733
 write-regs-off, 733
 write-regs-ok, 734
 write0->write3\$sim, 905
 write0\$step, 856
 write1\$step, 858
 write2\$step, 860
 write3\$induction, 884, 885
 write3\$progress, 886
 write3\$progress-help, 885
 write3\$step0, 862
 write3\$step1, 863

 x, 149, 150, 164, 165, 174–197, 203–
 205, 451, 748, 758, 759, 762–
 764, 800, 802–805, 812–814,
 927, 954, 971, 975, 993, 1007,
 1011–1015, 1017, 1019, 1022–
 1026
 xor, 152, 440
 xp, 938
 xs-suffice-for-reset-chip-final
 -state-for-any-unknown-state, 1020
 -state-for-any-unknown-state-better,
 1021
 -state-for-any-unknown-state-better-
 off, 1027
 -state-for-any-unknown-state-better-
 on, 1117
 -state-for-any-unknown-state-off,
 1028
 -state-for-any-unknown-state-on,
 1117

 xs-suffice-for-reset-chip-lemma
 -instance, 1016
 -instance-off, 1029
 -instance-on, 1119
 xs-suffice-for-reset-lemma, 1011
 xs-suffice-for-reset-lemma-off, 1033
 xs-suffice-for-reset-lemma-on, 1122
 xs-suffice-for-reset-lemma-verb
 ose, 1010
 ose-off, 1033
 ose-on, 1122

 z, 149, 150, 165, 182, 183, 194–197,
 204, 205, 758, 928
 z-flag, 543, 544, 559, 744, 1186, 1189
 z-flag-off, 543
 z-set, 542, 544, 591, 742–744, 1188,
 1189, 1191, 1194–1196, 1201,
 1203–1207, 1209–1214, 1216
 z-set-off, 542
 zb, 482–484, 532, 544, 742–744, 1188,
 1216
 zb-cvzbv, 1216
 zb-off, 482
 zb-v-alu, 1188
 zero-flag-interpretation-for-su
 b, 1189
 zero-ilessp-implies-not-equal, 81
 zero-ilessp-implies-not-equal-o
 ff, 104
 ff1, 1156
 n, 1067
 zero-loadingp, 306, 307, 323
 zero-loadingp-off, 306
 zerop-makes-equal-true-bridge, 20
 zerop-makes-equal-true-bridge-o
 ff, 20
 zerop-makes-lessp-false-bridge, 19
 zerop-makes-lessp-false-bridge-
 off, 19
 zerop-makes-quotient-zero-bridge, 29
 -off, 29
 zerop-makes-times-tree-zero, 17
 zerop-makes-times-tree-zero-off, 17

zerop-makes-times-tree-zero2, 18
zerop-makes-times-tree-zero2-of
f, 18
zerop-not-zerop-cases, 887
zerop-quotient, 134
zp, 928