#|

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

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 = 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) $\in x$ **then** bagdiff (delete (car (y), x), cdr (y))

else bagdiff $(x,\ \text{cdr}\,(y))$ **endif**
else x **endif**

DEFINITION:
bagint $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$
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 $(x,\ \text{cdr}\,(l))$
else occurrences $(x,\ \text{cdr}\,(l))$ **endif**
else 0 **endif**

DEFINITION:
subbagp $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$ **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** $(x \neq \text{car}\,(l)) \vee \text{listp}\,(\text{cdr}\,(l))$
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
$(x \in \text{delete}(y, l))$
$= \quad$ **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
$\text{occurrences}(x, \text{delete}(y, l))$
$= \quad$ **if** $x = y$
then if $x \in l$ **then** $\text{occurrences}(x, l) - 1$
else 0 **endif**
else $\text{occurrences}(x, l)$ **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
$\text{occurrences}(x, \text{bagint}(a, b))$
$= \quad$ **if** $\text{occurrences}(x, a) < \text{occurrences}(x, b)$ **then** $\text{occurrences}(x, a)$
else $\text{occurrences}(x, b)$ **endif**

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

THEOREM: member-bagint
$(x \in \text{bagint}(a,\ b)) = ((x \in a) \land (x \in b))$

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
$((a + b) = \mathsf{0}) = ((a \simeq \mathsf{0}) \land (b \simeq \mathsf{0}))$

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
$(((x - y) = \mathsf{0}) = (y \not< x)) \land ((\mathsf{0} = (x - y)) = (y \not< x))$

THEOREM: difference-cancellation
$((x - y) = (z - y))$
$= \ \ \textbf{if } x < y \textbf{ then } y \not< z$
$\quad\ \ \textbf{elseif } z < y \textbf{ then } y \not< x$
$\quad\ \ \textbf{else } \text{fix}(x) = \text{fix}(z) \textbf{ endif}$

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

THEOREM: plus-add1-arg1
$((1 + a) + b) = (1 + (a + b))$

THEOREM: plus-add1-arg2
$(x + (1 + y))$
$= \ \ \textbf{if } y \in \mathbf{N} \textbf{ then } 1 + (x + y)$
$\quad\ \ \textbf{else } 1 + x \textbf{ endif}$

THEOREM: associativity-of-plus
$((x + y) + z) = (x + y + z)$

THEOREM: plus-difference-arg1
$((a - b) + c)$
$= \quad$ **if** $b < a$ **then** $(a + c) - b$
else $0 + c$ **endif**

THEOREM: plus-difference-arg2
$(a + (b - c))$
$= \quad$ **if** $c < b$ **then** $(a + b) - c$
else $a + 0$ **endif**

THEOREM: difference-plus-cancellation-proof
$((x + y) - x) = \mathrm{fix}\,(y)$

THEOREM: difference-plus-cancellation
$(((x + y) - x) = \mathrm{fix}\,(y)) \wedge (((y + x) - x) = \mathrm{fix}\,(y))$

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
$(((x + y) - (x + z)) = (y - z))$
$\wedge \quad (((y + x) - (x + z)) = (y - z))$
$\wedge \quad (((x + y) - (z + x)) = (y - z))$
$\wedge \quad (((y + x) - (z + x)) = (y - z))$

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
$(a - (b - 1))$
$= \quad$ **if** $b \simeq 0$ **then** $\mathrm{fix}\,(a)$
elseif $a < b$ **then** 0
else $1 + (a - b)$ **endif**

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
$(a - (b - c))$
$=$ **if** $b < c$ **then** fix (a)
else $(a + c) - b$ **endif**

THEOREM: diff-diff-diff
(**if** $a < b$ **then f**
else t endif
$\wedge$ **if** $c < d$ **then f**
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
$(a < b) \rightarrow ((a - b) = \mathtt{0})$

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

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

DEFINITION:
plus-tree (l)
$=$ **if** $l \simeq$ **nil** **then** `''0`
elseif cdr $(l) \simeq$ **nil** **then** list (`'fix`, car (l))
elseif cddr $(l) \simeq$ **nil** **then** list (`'plus`, car (l), cadr (l))
else list (`'plus`, car (l), plus-tree (cdr (l))) **endif**

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

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

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

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

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

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

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

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

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

Event: Enable member-implies-numberp; name this event 'member-implies-numberp-off'.

Theorem: cadr-eval\$-list
$(\text{car}(\text{eval\$}(\texttt{'list},\ x,\ a)) = \text{eval\$}(\mathbf{t},\ \text{car}(x),\ a))$
$\land \quad (\text{cdr}(\text{eval\$}(\texttt{'list},\ x,\ a))$
$\quad = \quad \textbf{if } \text{listp}(x) \textbf{ then } \text{eval\$}(\texttt{'list},\ \text{cdr}(x),\ a)$
$\quad\quad \textbf{else } \texttt{0} \textbf{ endif})$

Event: Enable cadr-eval\$-list; name this event 'cadr-eval\$-list-off'.

Theorem: eval\$-quote
$\text{eval\$}(\mathbf{t},\ \text{cons}(\texttt{'quote},\ \mathit{args}),\ a) = \text{car}(\mathit{args})$

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

Theorem: listp-eval\$
$\text{listp}(\text{eval\$}(\texttt{'list},\ x,\ a)) = \text{listp}(x)$

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

Definition:
$\text{cancel-equal-plus}(x)$
$= \quad \textbf{if } \text{listp}(x) \land (\text{car}(x) = \texttt{'equal})$
$\quad \textbf{then if } \text{listp}(\text{cadr}(x))$
$\quad\quad\quad \land \quad (\text{caadr}(x) = \texttt{'plus})$
$\quad\quad\quad \land \quad \text{listp}(\text{caddr}(x))$

```
            ∧  (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))
               ∧  (caadr (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))
            ∧  (caadr (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))
$\land$ (caadr (x) = `'plus`)
$\land$ (caddr (x) $\in$ plus-fringe (cadr (x)))
then plus-tree (delete (caddr (x), plus-fringe (cadr (x))))
elseif listp (caddr (x))
$\land$ (caaddr (x) = `'plus`)
$\land$ (cadr (x) $\in$ plus-fringe (caddr (x))) **then** `''0`
else x **endif**
else x **endif**

THEOREM: correctness-of-cancel-difference-plus
eval\$ (**t**, x, a) = eval\$ (**t**, cancel-difference-plus (x), a)

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

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

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** $b \simeq \mathtt{0}$ **then** 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** fix (a)
else $(a + c) - b$ **endif**

THEOREM: difference-x-x
$(x - x) = \mathtt{0}$

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) ∧ (car (x) = 'lessp)
   then if listp (cadr (x))
           ∧  (caadr (x) = 'plus)
           ∧  listp (caddr (x))
           ∧  (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))
               ∧  (caadr (x) = 'plus)
               ∧  (caddr (x) ∈ plus-fringe (cadr (x)))
        then list ('quote, f)
        elseif listp (caddr (x))
               ∧  (caaddr (x) = 'plus)
               ∧  (cadr (x) ∈ 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

eval\$ $(\mathbf{t}, x, a) =$ eval\$ $(\mathbf{t},$ 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
$(x * (1 + y))$
$= \quad$ **if** $y \in \mathbf{N}$ **then** $x + (x * y)$
else $\mathrm{fix}(x)$ **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
$((x * (y + z)) = ((x * y) + (x * z)))$
$\wedge \quad (((x + y) * z) = ((x * z) + (y * z)))$

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
$(((a - b) * c) = ((a * c) - (b * c)))$
$\wedge \quad ((a * (b - c)) = ((a * b) - (a * c)))$

THEOREM: times-quotient-proof
$((x \not\simeq 0) \wedge ((y \ \mathbf{mod}\ x) = 0)) \rightarrow (((y \div x) * x) = \mathrm{fix}(y))$

THEOREM: times-quotient
$((y \not\simeq 0) \land ((x \ \mathbf{mod} \ y) = 0))$
$\rightarrow \quad ((((x \div y) * y) = \mathrm{fix}(x)) \land ((y * (x \div y)) = \mathrm{fix}(x)))$

THEOREM: times-1-arg1
$(1 * x) = \mathrm{fix}(x)$

THEOREM: lessp-times1-proof
$((a < b) \land (c \not\simeq 0)) \rightarrow ((a < (b * c)) = \mathbf{t})$

THEOREM: lessp-times1
$((a < b) \land (c \not\simeq 0))$
$\rightarrow \quad (((a < (b * c)) = \mathbf{t}) \land ((a < (c * b)) = \mathbf{t}))$

THEOREM: lessp-times2-proof
(**if** $b < a$ **then f**
else t endif
$\land \quad (c \not\simeq 0))$
$\rightarrow \quad (((b * c) < a) = \mathbf{f})$

THEOREM: lessp-times2
(**if** $b < a$ **then f**
else t endif
$\land \quad (c \not\simeq 0))$
$\rightarrow \quad ((((b * c) < a) = \mathbf{f}) \land (((c * b) < a) = \mathbf{f}))$

THEOREM: lessp-times3-proof1
$((a \not\simeq 0) \land (1 < b)) \rightarrow (a < (a * b))$

THEOREM: lessp-times3-proof2
$(a < (a * b)) \rightarrow ((a \not\simeq 0) \land (1 < b))$

THEOREM: lessp-times3
$((a < (a * b)) = ((a \not\simeq 0) \land (1 < b)))$
$\land \quad ((a < (b * a)) = ((a \not\simeq 0) \land (1 < b)))$

THEOREM: lessp-times-cancellation-proof
$((x * z) < (y * z)) = ((z \not\simeq 0) \land (x < y))$

THEOREM: lessp-times-cancellation1
$(((x * z) < (y * z)) = ((z \not\simeq 0) \land (x < y)))$
$\land \quad (((z * x) < (y * z)) = ((z \not\simeq 0) \land (x < y)))$
$\land \quad (((x * z) < (z * y)) = ((z \not\simeq 0) \land (x < y)))$
$\land \quad (((z * x) < (z * y)) = ((z \not\simeq 0) \land (x < y)))$

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
$(((a + (b * c)) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad ((((c * b) + a) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$
$\wedge \quad ((((b * c) + a) < b) = ((a < b) \wedge (c \simeq \mathtt{0})))$

THEOREM: lessp-plus-times2
$((a \not\simeq \mathtt{0}) \wedge (x < a))$
$\rightarrow \quad ((((x + (a * b)) < (a * c)) = (b < c))$
$\wedge \quad (((x + (b * a)) < (a * c)) = (b < c))$
$\wedge \quad (((x + (a * b)) < (c * a)) = (b < c))$
$\wedge \quad (((x + (b * a)) < (c * a)) = (b < c))$
$\wedge \quad ((((a * b) + x) < (a * c)) = (b < c))$
$\wedge \quad ((((b * a) + x) < (a * c)) = (b < c))$
$\wedge \quad ((((a * b) + x) < (c * a)) = (b < c))$
$\wedge \quad ((((b * a) + x) < (c * a)) = (b < c)))$

THEOREM: lessp-1-times
$(\mathtt{1} < (a * b))$
$= \quad (\neg ((a \simeq \mathtt{0}) \vee (b \simeq \mathtt{0}) \vee ((a = \mathtt{1}) \wedge (b = \mathtt{1}))))$

DEFINITION:
times-tree (x)
= **if** $x \simeq$ **nil then** `''1`
elseif cdr $(x) \simeq$ **nil then** list (`'fix`, car (x))
elseif cddr $(x) \simeq$ **nil then** list (`'times`, car (x), cadr (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-zerop-tree (x)
= **if** $x \simeq$ **nil then** `'(false)`
elseif cdr $(x) \simeq$ **nil then** list (`'zerop`, car (x))
elseif cddr $(x) \simeq$ **nil**
then list (`'or`, list (`'zerop`, car (x)), list (`'zerop`, cadr (x)))
else list (`'or`, list (`'zerop`, car (x)), or-zerop-tree (cdr (x))) **endif**

DEFINITION:
and-not-zerop-tree (x)
$=$ **if** $x \simeq$ **nil then** `'(true)`
elseif cdr $(x) \simeq$ **nil then** list (`'not`, list (`'zerop`, car (x)))
else list (`'and`,
list (`'not`, list (`'zerop`, car (x))),
and-not-zerop-tree (cdr (x))) **endif**

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

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

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

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

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

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

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

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

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

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

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

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

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

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
$(a \not\simeq \mathsf{0}) \rightarrow (((a * x) \not< (a * y)) = (x \not< y))$

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

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

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

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

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

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

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

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

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

THEOREM: or-zerop-tree-is-not-zerop-tree
eval\$ (**t**, or-zerop-tree (x), a) = (¬ eval\$ (**t**, and-not-zerop-tree (x), a))

EVENT: Enable or-zerop-tree-is-not-zerop-tree; name this event 'or-zerop-tree-is-not-zerop-tree-off'.

THEOREM: zerop-makes-times-tree-zero2
(eval\$ (**t**, or-zerop-tree (x), a) ∧ subbagp (x, y))
→ (eval\$ (**t**, times-tree (y), a) = `0`)

EVENT: Enable zerop-makes-times-tree-zero2; name this event 'zerop-makes-times-tree-zero2-off'.

THEOREM: times-tree-append
eval\$ (**t**, times-tree (append (x, y)), a)
= (eval\$ (**t**, times-tree (x), a) ∗ eval\$ (**t**, times-tree (y), a))

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

THEOREM: times-tree-of-times-fringe
eval\$ (**t**, times-tree (times-fringe (x)), a) = fix (eval\$ (**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-zerop-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

(subbagp (x, y) $\land$ subbagp (x, z) $\land$ eval\$ (**t**, and-not-zerop-tree (x), a))
$\rightarrow$ ((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\$-lessp-times-tree-bagdiff; name this event 'eval\$-lessp-times-tree-bagdiff-off'.

THEOREM: zerop-makes-lessp-false-bridge

((car (x) = `'times`)
$\land$ (car (y) = `'times`)
$\land$ ($\neg$ eval\$ (**t**,
and-not-zerop-tree (bagint (times-fringe (x), times-fringe (y))),
a)))
$\rightarrow$ (((eval\$ (**t**, cadr (x), a) $*$ eval\$ (**t**, caddr (x), a))
$<$ (eval\$ (**t**, cadr (y), a) $*$ eval\$ (**t**, caddr (y), a)))
$=$ **f**)

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

THEOREM: correctness-of-cancel-lessp-times

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

DEFINITION:

cancel-equal-times (x)
= **if** (car (x) = `'equal`)
$\land$ (caadr (x) = `'times`)
$\land$ (caaddr (x) = `'times`)
then if listp (bagint (times-fringe (cadr (x)), times-fringe (caddr (x))))
then list (`'or`,
or-zerop-tree (bagint (times-fringe (cadr (x)),
times-fringe (caddr (x)))),
list (`'equal`,
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: zerop-makes-equal-true-bridge
$((\text{car}(x) = \texttt{'times})$
$\land\ (\text{car}(y) = \texttt{'times})$
$\land\ \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y))), a))$
$\rightarrow\ (((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a))$
$= (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a)))$
$= \mathbf{t})$

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

THEOREM: eval$-equal-times-tree-bagdiff
$(\text{subbagp}(x, y) \land \text{subbagp}(x, z) \land (\neg\ \text{eval\$}(\mathbf{t}, \text{or-zerop-tree}(x), a)))$
$\rightarrow\ ((\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a)$
$= \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a))$
$= (\text{eval\$}(\mathbf{t}, \text{times-tree}(y), a) = \text{eval\$}(\mathbf{t}, \text{times-tree}(z), a)))$

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

THEOREM: cancel-equal-times-preserves-inequality
$(\text{subbagp}(z, x)$
$\land\ \text{subbagp}(z, y)$
$\land\ (\text{eval\$}(\mathbf{t}, \text{times-tree}(x), a) \neq \text{eval\$}(\mathbf{t}, \text{times-tree}(y), a)))$
$\rightarrow\ (\text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(x, z)), a)$
$\neq \text{eval\$}(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, z)), a))$

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

THEOREM: cancel-equal-times-preserves-inequality-bridge
$((\text{car}(x) = \texttt{'times})$
$\land\ (\text{car}(y) = \texttt{'times})$
$\land\ ((\text{eval\$}(\mathbf{t}, \text{cadr}(x), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(x), a))$
$\neq (\text{eval\$}(\mathbf{t}, \text{cadr}(y), a) * \text{eval\$}(\mathbf{t}, \text{caddr}(y), a))))$
$\rightarrow\ (\text{eval\$}(\mathbf{t},$
$\text{times-tree}(\text{bagdiff}(\text{times-fringe}(x),$
$\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y)))),$
$a)$
$\neq \text{eval\$}(\mathbf{t},$
$\text{times-tree}(\text{bagdiff}(\text{times-fringe}(y),$
$\text{bagint}(\text{times-fringe}(x),$
$\text{times-fringe}(y)))),$
$a))$

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
$((x \textbf{ mod } y) < y) = (y \not\simeq 0)$

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

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

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

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

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

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

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

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

THEOREM: remainder-plus
$$\begin{array}{l} ((a \textbf{ mod } c) = 0) \\ \rightarrow \quad ((((a + b) \textbf{ mod } c) = (b \textbf{ mod } c)) \\ \qquad \wedge \quad (((b + a) \textbf{ mod } c) = (b \textbf{ mod } c)) \\ \qquad \wedge \quad (((x + y + a) \textbf{ mod } c) = ((x + y) \textbf{ mod } c))) \end{array}$$

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

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

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

THEOREM: equal-remainder-plus-remainder
$(a < c)$
$\rightarrow \quad (((((a + b) \ \mathbf{mod}\ c) = (b \ \mathbf{mod}\ c)) = (a \simeq 0))$
$\quad \wedge \quad ((((b + a) \ \mathbf{mod}\ c) = (b \ \mathbf{mod}\ c)) = (a \simeq 0)))$

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

THEOREM: remainder-times1-proof
$((b \ \mathbf{mod}\ c) = 0) \rightarrow (((a * b) \ \mathbf{mod}\ c) = 0)$

THEOREM: remainder-times1
$((b \ \mathbf{mod}\ c) = 0)$
$\rightarrow \quad ((((a * b) \ \mathbf{mod}\ c) = 0) \wedge (((b * a) \ \mathbf{mod}\ c) = 0))$

THEOREM: remainder-times1-instance-proof
$((x * y) \ \mathbf{mod}\ y) = 0$

THEOREM: remainder-times1-instance
$(((x * y) \ \mathbf{mod}\ y) = 0) \wedge (((x * y) \ \mathbf{mod}\ x) = 0)$

THEOREM: remainder-times-times-proof
$((x * y) \ \mathbf{mod}\ (x * z)) = (x * (y \ \mathbf{mod}\ z))$

THEOREM: remainder-times-times
$(((x * y) \ \mathbf{mod}\ (x * z)) = (x * (y \ \mathbf{mod}\ z)))$
$\wedge \quad (((x * z) \ \mathbf{mod}\ (y * z)) = ((x \ \mathbf{mod}\ y) * z))$

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

THEOREM: remainder-times2-proof
$((a \ \mathbf{mod}\ z) = 0) \rightarrow ((a \ \mathbf{mod}\ (z * y)) = (z * ((a \div z) \ \mathbf{mod}\ y)))$

THEOREM: remainder-times2
$((a \textbf{ mod } z) = 0)$
$\rightarrow \quad (((a \textbf{ mod } (y * z)) = (z * ((a \div z) \textbf{ mod } y)))$
$\quad\quad \wedge \quad ((a \textbf{ mod } (z * y)) = (z * ((a \div z) \textbf{ mod } y))))$

THEOREM: remainder-times2-instance
$(((x * y) \textbf{ mod } (x * z)) = (x * (y \textbf{ mod } z)))$
$\wedge \quad (((x * z) \textbf{ mod } (y * z)) = ((x \textbf{ mod } y) * z))$

THEOREM: remainder-difference1
$((a \textbf{ mod } c) = (b \textbf{ mod } c))$
$\rightarrow \quad (((a - b) \textbf{ mod } c) = ((a \textbf{ mod } c) - (b \textbf{ mod } c)))$

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
$(((a \textbf{ mod } c) = 0) \wedge ((b \textbf{ mod } c) \neq 0))$
$\rightarrow \quad (((a - b) \textbf{ mod } c)$
$= \textbf{if } b < a \textbf{ then } c - (b \textbf{ mod } c)$
else 0 **endif**)

THEOREM: remainder-difference3
$(((b \textbf{ mod } c) = 0) \wedge ((a \textbf{ mod } c) \neq 0))$
$\rightarrow \quad (((a - b) \textbf{ mod } c)$
$= \textbf{if } b < a \textbf{ then } a \textbf{ mod } c$
else 0 **endif**)

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

THEOREM: equal-remainder-difference-0
$(((a - b) \textbf{ mod } c) = 0)$
$=$ **if if** $a < b$ **then f**
else t endif then $(a \textbf{ mod } c) = (b \textbf{ mod } c)$
else t endif

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

THEOREM: lessp-plus-fact
$(((b \textbf{ mod } x) = 0) \land ((c \textbf{ mod } x) = 0) \land (b < c) \land (a < x))$
$\rightarrow \quad (((a + b) < c) = \mathbf{t})$

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

THEOREM: remainder-plus-fact
$(((b \textbf{ mod } x) = 0) \land ((c \textbf{ mod } x) = 0) \land (a < x))$
$\rightarrow \quad (((a + b) \textbf{ mod } c) = (a + (b \textbf{ mod } c)))$

THEOREM: remainder-plus-times-times-proof
$(a < b)$
$\rightarrow \quad (((a + (b * c)) \textbf{ mod } (b * d))$
$\quad\quad = \quad (a + ((b * c) \textbf{ mod } (b * d))))$

THEOREM: remainder-plus-times-times
$(a < b)$
$\rightarrow \quad ((((a + (b * c)) \textbf{ mod } (b * d))$
$\quad\quad = \quad (a + ((b * c) \textbf{ mod } (b * d))))$
$\quad\quad \land \quad (((a + (c * b)) \textbf{ mod } (d * b))$
$\quad\quad\quad = \quad (a + ((c * b) \textbf{ mod } (d * b)))))$

THEOREM: remainder-plus-times-times-instance
$(a < b)$
$\rightarrow \quad ((((a + (b * c) + (b * d)) \textbf{ mod } (b * e))$
$\quad\quad = \quad (a + (b * ((c + d) \textbf{ mod } e))))$
$\quad\quad \land \quad (((a + (c * b) + (d * b)) \textbf{ mod } (e * b))$
$\quad\quad\quad = \quad (a + (b * ((c + d) \textbf{ mod } e)))))$

THEOREM: remainder-remainder
$((b \textbf{ mod } a) = 0) \rightarrow (((n \textbf{ mod } b) \textbf{ mod } a) = (n \textbf{ mod } a))$

THEOREM: remainder-1-arg1
$(1 \textbf{ mod } x)$
$= \quad \textbf{if } x = 1 \textbf{ then } 0$
$\quad\quad \textbf{else } 1 \textbf{ endif}$

THEOREM: remainder-1-arg2
$(y \textbf{ mod } 1) = 0$

THEOREM: remainder-x-x
$(x \textbf{ mod } x) = 0$

THEOREM: transitivity-of-divides
$(((a \textbf{ mod } b) = 0) \land ((b \textbf{ mod } c) = 0)) \rightarrow ((a \textbf{ mod } c) = 0)$

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
$(b = \mathtt{1}) \rightarrow ((a \div b) = \mathrm{fix}\,(a))$

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

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

THEOREM: quotient-add1
$$\begin{array}{ll} ((a \textbf{ mod } b) = \mathtt{0}) & \\ \rightarrow & (((1 + a) \div b) \\ & \quad = \textbf{if } b = \mathtt{1} \textbf{ then } 1 + (a \div b) \\ & \qquad \textbf{else } a \div b \textbf{ endif}) \end{array}$$

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

THEOREM: quotient-sub1
$$\begin{array}{ll} ((a \not\simeq \mathtt{0}) \wedge (b \not\simeq \mathtt{0})) & \\ \rightarrow & (((a - 1) \div b) \\ & \quad = \textbf{if } (a \textbf{ mod } b) = \mathtt{0} \textbf{ then } (a \div b) - 1 \\ & \qquad \textbf{else } a \div b \textbf{ endif}) \end{array}$$

THEOREM: quotient-plus-proof
$((b \textbf{ mod } c) = \mathtt{0}) \rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$

THEOREM: quotient-plus
$$\begin{array}{ll} ((a \textbf{ mod } c) = \mathtt{0}) & \\ \rightarrow & ((((a + b) \div c) = ((a \div c) + (b \div c))) \\ & \quad \wedge \quad (((b + a) \div c) = ((a \div c) + (b \div c))) \\ & \quad \wedge \quad (((x + y + a) \div c) \\ & \qquad\quad = \quad (((x + y) \div c) + (a \div c)))) \end{array}$$

THEOREM: quotient-times-instance-temp-proof
$$\begin{array}{ll} ((y * x) \div y) & \\ = & \textbf{if } y \simeq \mathtt{0} \textbf{ then } \mathtt{0} \\ & \textbf{else } \mathrm{fix}\,(x) \textbf{ endif} \end{array}$$

THEOREM: quotient-times-instance-temp

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

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

THEOREM: quotient-times-proof

$$((a \textbf{ mod } c) = 0) \rightarrow (((a * b) \div c) = (b * (a \div c)))$$

THEOREM: quotient-times

$$\begin{aligned}&((a \textbf{ mod } c) = 0)\\ &\rightarrow\ \ ((((a * b) \div c) = (b * (a \div c)))\\ &\quad\quad \land\ \ (((b * a) \div c) = (b * (a \div c))))\end{aligned}$$

THEOREM: quotient-times-instance

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

THEOREM: quotient-times-times-proof

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

THEOREM: quotient-times-times

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

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

THEOREM: quotient-difference1

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

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

THEOREM: quotient-difference2
$$\begin{array}{l} (((a \textbf{ mod } c) = 0) \wedge ((b \textbf{ mod } c) \neq 0)) \\ \rightarrow \quad (((a - b) \div c) \\ \qquad = \quad \textbf{if } b < a \textbf{ then } (a \div c) - (1 + (b \div c)) \\ \qquad\qquad \textbf{else } 0 \textbf{ endif}) \end{array}$$

THEOREM: quotient-difference3
$$\begin{array}{l} (((b \textbf{ mod } c) = 0) \wedge ((a \textbf{ mod } c) \neq 0)) \\ \rightarrow \quad (((a - b) \div c) \\ \qquad = \quad \textbf{if } b < a \textbf{ then } (a \div c) - (b \div c) \\ \qquad\qquad \textbf{else } 0 \textbf{ endif}) \end{array}$$

THEOREM: remainder-equals-its-first-argument
$(a = (a \textbf{ mod } b)) = ((a \in \mathbf{N}) \wedge ((b \simeq 0) \vee (a < b)))$

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

THEOREM: quotient-remainder-times
$((x \textbf{ mod } (a * b)) \div a) = ((x \div a) \textbf{ mod } b)$

THEOREM: quotient-remainder
$((c \textbf{ mod } a) = 0) \rightarrow (((b \textbf{ mod } c) \div a) = ((b \div a) \textbf{ mod } (c \div a)))$

THEOREM: quotient-remainder-instance
$((x \textbf{ mod } (a * b)) \div a) = ((x \div a) \textbf{ mod } b)$

THEOREM: quotient-plus-fact
$$\begin{array}{l} (((b \textbf{ mod } x) = 0) \wedge ((c \textbf{ mod } x) = 0) \wedge (a < x)) \\ \rightarrow \quad (((a + b) \div c) = (b \div c)) \end{array}$$

THEOREM: quotient-plus-times-times-proof
$$\begin{array}{l} (a < b) \\ \rightarrow \quad (((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d))) \end{array}$$

THEOREM: quotient-plus-times-times
$$\begin{array}{l} (a < b) \\ \rightarrow \quad ((((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d))) \\ \qquad \wedge \quad (((a + (b * c)) \div (b * d)) \\ \qquad\qquad = \quad ((b * c) \div (b * d)))) \end{array}$$

THEOREM: quotient-plus-times-times-instance

$$
\begin{array}{l}
(a < b) \\
\rightarrow \quad ((((a + (b * c) + (b * d)) \div (b * e)) \\
\qquad = \quad \textbf{if } b \simeq \texttt{0} \textbf{ then } \texttt{0} \\
\qquad\qquad \textbf{else } (c + d) \div e \textbf{ endif}) \\
\qquad \wedge \quad (((a + (c * b) + (d * b)) \div (e * b)) \\
\qquad\qquad = \quad \textbf{if } b \simeq \texttt{0} \textbf{ then } \texttt{0} \\
\qquad\qquad\qquad \textbf{else } (d + c) \div e \textbf{ endif}))
\end{array}
$$

THEOREM: quotient-quotient

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

THEOREM: leq-quotient

$$
\begin{array}{l}
(a < b) \\
\rightarrow \quad \textbf{if } (b \div c) < (a \div c) \textbf{ then f} \\
\qquad \textbf{else t endif}
\end{array}
$$

THEOREM: quotient-1-arg2

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

THEOREM: quotient-1-arg1-casesplit

$$(n \simeq \texttt{0}) \vee (n = \texttt{1}) \vee (\texttt{1} < n)$$

THEOREM: quotient-1-arg1

$$
\begin{array}{l}
(\texttt{1} \div n) \\
= \quad \textbf{if } n = \texttt{1} \textbf{ then } \texttt{1} \\
\qquad \textbf{else } \texttt{0} \textbf{ endif}
\end{array}
$$

THEOREM: quotient-x-x

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

THEOREM: lessp-quotient

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

DEFINITION:

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

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:

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

DEFINITION:
$\log(base,\ n)$
$= \ \ \textbf{if}\ base < 2\ \ \textbf{then}\ 0$
$\qquad \textbf{elseif}\ n \simeq 0\ \ \textbf{then}\ 0$
$\qquad \textbf{else}\ 1 + \log(base,\ n \div base)\ \textbf{endif}$

DEFINITION:
$\mathrm{gcd}(x,\ y)$
$= \ \ \textbf{if}\ x \simeq 0\ \ \textbf{then}\ \mathrm{fix}(y)$
$\qquad \textbf{elseif}\ y \simeq 0\ \ \textbf{then}\ x$
$\qquad \textbf{elseif}\ x < y\ \ \textbf{then}\ \mathrm{gcd}(x,\ y - x)$
$\qquad \textbf{else}\ \mathrm{gcd}(x - y,\ y)\ \textbf{endif}$

THEOREM: remainder-exp
$(k \not\simeq 0) \rightarrow ((\exp(n,\ k)\ \textbf{mod}\ n) = 0)$

DEFINITION:
double-number-induction$(i,\ j)$
$= \ \ \textbf{if}\ i \simeq 0\ \ \textbf{then}\ 0$
$\qquad \textbf{elseif}\ j \simeq 0\ \ \textbf{then}\ 0$
$\qquad \textbf{else}$ double-number-induction$(i - 1,\ j - 1)$ $\textbf{endif}$

THEOREM: remainder-exp-exp
$\textbf{if}\ j < i\ \ \textbf{then}\ \textbf{f}$
$\textbf{else}\ \textbf{t}\ \textbf{endif}$
$\rightarrow \quad ((\exp(a,\ j)\ \textbf{mod}\ \exp(a,\ i)) = 0)$

THEOREM: quotient-exp
$(k \not\simeq 0)$
$\rightarrow \quad ((\exp(n,\ k) \div n)$
$\qquad = \ \ \textbf{if}\ n \simeq 0\ \ \textbf{then}\ 0$
$\qquad\qquad \textbf{else}\ \exp(n,\ k - 1)\ \textbf{endif})$

THEOREM: exp-zero
$(k \simeq 0) \rightarrow (\exp(n,\ k) = 1)$

THEOREM: exp-add1
$\exp(n,\ 1 + k) = (n * \exp(n,\ k))$

THEOREM: exp-plus
$\exp(i,\ j + k) = (\exp(i,\ j) * \exp(i,\ k))$

THEOREM: exp-0-arg1
$\exp(0, k)$
$= \ \textbf{if } k \simeq 0 \textbf{ then } 1$
$\quad \textbf{else } 0 \textbf{ endif}$

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
$(\exp(n, k) = 1)$
$= \ \textbf{if } k \simeq 0 \textbf{ then t}$
$\quad \textbf{else } n = 1 \textbf{ endif}$

THEOREM: exp-difference
$(\textbf{if } b < c \textbf{ then f}$
$\textbf{else t endif}$
$\wedge \quad (a \not\simeq 0))$
$\rightarrow \quad (\exp(a, b - c) = (\exp(a, b) \div \exp(a, c)))$

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

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

Theorem: log-times-proof
$((1 < c) \land (n \not\simeq 0)) \rightarrow (\log(c,\ c * n) = (1 + \log(c,\ n)))$

Theorem: log-times
$((1 < c) \land (n \not\simeq 0))$
$\rightarrow ((\log(c,\ c * n) = (1 + \log(c,\ n)))$
$\land (\log(c,\ n * c) = (1 + \log(c,\ n))))$

Theorem: log-times-exp-proof
$((1 < c) \land (n \not\simeq 0)) \rightarrow (\log(c,\ n * \exp(c,\ m)) = (\log(c,\ n) + m))$

Theorem: log-times-exp
$((1 < c) \land (n \not\simeq 0))$
$\rightarrow ((\log(c,\ n * \exp(c,\ m)) = (\log(c,\ n) + m))$
$\land (\log(c,\ \exp(c,\ m) * n) = (\log(c,\ n) + m)))$

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
$\mathrm{gcd}(b, a) = \mathrm{gcd}(a, b)$

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

THEOREM: gcd-0
$(\mathrm{gcd}(0, x) = \mathrm{fix}(x)) \land (\mathrm{gcd}(x, 0) = \mathrm{fix}(x))$

THEOREM: gcd-1
$(\mathrm{gcd}(1, x) = 1) \land (\mathrm{gcd}(x, 1) = 1)$

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

THEOREM: lessp-gcd
$(b \not\simeq 0) \to (((b < \mathrm{gcd}(a, b)) = \mathbf{f}) \land ((b < \mathrm{gcd}(b, a)) = \mathbf{f}))$

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

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

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

THEOREM: gcd-plus
$$\begin{aligned}
&((b \ \mathbf{mod}\ a) = 0)\\
\to\quad &((\mathrm{gcd}(a, b + c) = \mathrm{gcd}(a, c))\\
&\quad \land\quad (\mathrm{gcd}(a, c + b) = \mathrm{gcd}(a, c))\\
&\quad \land\quad (\mathrm{gcd}(b + c, a) = \mathrm{gcd}(a, c))\\
&\quad \land\quad (\mathrm{gcd}(c + b, a) = \mathrm{gcd}(a, c)))
\end{aligned}$$

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

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

THEOREM: distributivity-of-times-over-gcd-proof
$\text{gcd}(x * z,\ y * z) = (z * \text{gcd}(x,\ y))$

THEOREM: distributivity-of-times-over-gcd
$(\text{gcd}(x * z,\ y * z) = (z * \text{gcd}(x,\ y)))$
$\land\quad (\text{gcd}(z * x,\ y * z) = (z * \text{gcd}(x,\ y)))$
$\land\quad (\text{gcd}(x * z,\ z * y) = (z * \text{gcd}(x,\ y)))$
$\land\quad (\text{gcd}(z * x,\ z * y) = (z * \text{gcd}(x,\ y)))$

THEOREM: gcd-is-the-greatest
$((x \not\simeq 0) \land (y \not\simeq 0) \land ((x \ \mathbf{mod}\ z) = 0) \land ((y \ \mathbf{mod}\ z) = 0))$
$\rightarrow\quad$ **if** $\text{gcd}(x,\ y) < z$ **then f**
else t endif

THEOREM: common-divisor-divides-gcd
$(((x \ \mathbf{mod}\ z) = 0) \land ((y \ \mathbf{mod}\ z) = 0)) \rightarrow ((\text{gcd}(x,\ y) \ \mathbf{mod}\ z) = 0)$

THEOREM: associativity-of-gcd-zero-case
$((a \simeq 0) \lor (b \simeq 0) \lor (c \simeq 0)) \rightarrow (\text{gcd}(\text{gcd}(a,\ b),\ c) = \text{gcd}(a,\ \text{gcd}(b,\ c)))$

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

THEOREM: commutativity2-of-gcd-zero-case
$((a \simeq 0) \lor (b \simeq 0) \lor (c \simeq 0)) \rightarrow (\text{gcd}(b,\ \text{gcd}(a,\ c)) = \text{gcd}(a,\ \text{gcd}(b,\ c)))$

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

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:
$\text{integerp}(x)$
$=\quad$ **if** $x \in \mathbf{N}$ **then t**
elseif $\text{negativep}(x)$ **then** $\text{negative-guts}(x) \not\simeq 0$
else f endif

DEFINITION:
fix-int (x)
$=$ **if** integerp (x) **then** x
else 0 **endif**

DEFINITION: izerop (i) = (fix-int $(i) = 0$)

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

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

DEFINITION:
iplus (x, y)
$=$ **if** negativep (x)
then if negativep (y)
then if (negative-guts $(x) \simeq 0$) $\wedge$ (negative-guts $(y) \simeq 0$) **then** 0
else $-$ (negative-guts (x) + negative-guts (y)) **endif**
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:
ineg (x)
$=$ **if** negativep (x) **then** negative-guts (x)
elseif $x \simeq 0$ **then** 0
else $-\ x$ **endif**

DEFINITION: idifference (x, y) = iplus $(x$, ineg $(y))$

DEFINITION:
iabs (i)
$=$ **if** negativep (i) **then** negative-guts (i)
else 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 $(j,$ iquotient $(i, j)))$

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 $(-$ (negative-guts $(i) \div j))$ **endif**
elseif negativep (j) **then** fix-int $(-$ $(i \div$ negative-guts $(j)))$
else $i \div j$ **endif**

DEFINITION:
irem (i, j) = idifference (fix-int (i), itimes $(j$, iquo $(i, j)))$

EVENT: Let us define the theory *integer-defns* to consist of the following events: integerp, fix-int, ilessp, 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, ilessp, 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 ilessp; name this event 'ilessp-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
integerp (iabs (x))

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

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

THEOREM: fix-int-fix-int
fix-int (fix-int (x)) $=$ fix-int (x)

THEOREM: fix-int-iplus
fix-int (iplus (a, b)) $=$ iplus (a, b)

THEOREM: fix-int-idifference
fix-int (idifference (a, b)) $=$ idifference (a, b)

THEOREM: fix-int-ineg
fix-int (ineg (x)) $=$ ineg (x)

THEOREM: fix-int-iabs
fix-int (iabs (x)) $=$ iabs (x)

THEOREM: fix-int-itimes
fix-int (itimes (x, y)) $=$ itimes (x, y)

THEOREM: ineg-iplus
ineg (iplus (a, b)) $=$ iplus (ineg (a), ineg (b))

THEOREM: ineg-ineg
ineg (ineg (x)) $=$ fix-int (x)

THEOREM: ineg-fix-int
ineg (fix-int (x)) $=$ ineg (x)

THEOREM: ineg-of-non-integerp
($\neg$ integerp (x)) $\rightarrow$ (ineg (x) $=$ 0)

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

THEOREM: ineg-0
ineg (0) $=$ 0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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, caddr (y))
                        elseif cancel-ineg-aux (x, caddr (y))
                        then list ('iplus,
                                   cadr (y),
                                   cancel-ineg-aux (x, caddr (y)))
                        else f endif
                   else f endif
              else f endif
         elseif y ≃ 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, 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
    elseif y ≃ 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, 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$-list-cons
eval$ ('list, cons (x, y), a) = cons (eval$ (**t**, x, a), eval$ ('list, y, a))

THEOREM: eval$-list-nlistp
($x \simeq$ **nil**) $\rightarrow$ (eval$ ('list, x, a) = **nil**)

THEOREM: eval$-litatom
litatom (x) $\rightarrow$ (eval$ (**t**, x, a) = cdr (assoc (x, a)))

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

THEOREM: eval$-other
(($\neg$ litatom (x)) $\wedge$ ($x \simeq$ **nil**)) $\rightarrow$ (eval$ (**t**, x, a) = x)

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

DEFINITION:
```
eval$-cancel-ineg-aux-fn (x, y, a)
=   if listp (x) ∧ (car (x) = 'ineg)
    then if y = cadr (x)  then 0
         elseif cadr (y) = cadr (x)  then fix-int (eval$ (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**
elseif car (y) = **`'ineg`** **then** `0`
elseif listp (cadr (y))
$\wedge$ (caadr (y) = **`'ineg`**)
$\wedge$ (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
(cancel-ineg-aux (x, y) $\neq$ **f**)
$\rightarrow$ (eval\$ (**t**, cancel-ineg-aux (x, y), a) = eval\$-cancel-ineg-aux-fn (x, y, a))

THEOREM: iplus-ineg3
iplus (ineg (x), iplus (x, y)) = fix-int (y)

THEOREM: iplus-ineg4
iplus (x, iplus (ineg (x), y)) = fix-int (y)

THEOREM: iplus-ineg-promote
iplus (y, ineg (x)) = iplus (ineg (x), y)

THEOREM: iplus-x-y-ineg-x
iplus (x, iplus (y, ineg (x))) = fix-int (y)

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

THEOREM: correctness-of-cancel-ineg-aux
(cancel-ineg-aux (x, y) $\neq$ **f**)
$\rightarrow$ (eval\$-cancel-ineg-aux-fn (x, y, a) = iplus (eval\$ (**t**, x, a), eval\$ (**t**, y, a)))

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

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

DEFINITION:
iplus-fringe (x)
= **if** listp (x)
then if car (x) = **`'iplus`**
then append (iplus-fringe (cadr (x)), iplus-fringe (caddr (x)))
else list (x) **endif**
else list (x) **endif**

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

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

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

DEFINITION:
iplus-list (x)
= **if** listp (x) **then** iplus (car (x), iplus-list (cdr (x)))
else `0` **endif**

THEOREM: integerp-iplus-list
integerp (iplus-list (x))

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

THEOREM: eval\$-iplus-tree
eval\$ (**t**, iplus-tree (x), a) = iplus-list (eval\$ (`'list`, x, a))

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

DEFINITION:
cancel-iplus (x)
= **if** listp (x)
then **if** car (x) = `'equal`
then **if** listp (cadr (x))
then **if** caadr (x) = `'iplus`
then **if** listp (caddr (x))
then **if** caaddr (x) = `'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
    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
            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
$(\text{listp}(x) \wedge (\text{car}(x) = \texttt{'iplus}))$
$\rightarrow$ $(\text{eval\$}(\mathbf{t}, x, a) = \text{iplus}(\text{eval\$}(\mathbf{t}, \text{cadr}(x), a), \text{eval\$}(\mathbf{t}, \text{caddr}(x), a)))$

THEOREM: iplus-ineg7
$(\mathtt{0} = \text{iplus}(x, \text{ineg}(y))) = (\text{fix-int}(y) = \text{fix-int}(x))$

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: ilessp-fix-int-1
$\text{ilessp}(\text{fix-int}(x), y) = \text{ilessp}(x, y)$

THEOREM: ilessp-fix-int-2
$\text{ilessp}(x, \text{fix-int}(y)) = \text{ilessp}(x, y)$

DEFINITION:
make-cancel-iplus-inequality-1 (x, y)
$=$ list ($\texttt{'ilessp}$,
iplus-tree (bagdiff $(x$, bagint $(x, y)))$,
iplus-tree (bagdiff $(y$, bagint $(x, y))))$

DEFINITION:
cancel-iplus-ilessp-1 (x)
$=$ **if** listp (x)
then if car $(x) = \texttt{'ilessp}$
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
ilessp (iplus (a, b), iplus (a, c)) = ilessp (b, c)

THEOREM: iplus-cancellation-2-for-ilessp
ilessp (iplus (b, a), iplus (c, a)) = ilessp (b, c)

THEOREM: correctness-of-cancel-iplus-ilessp-lemma
eval$ (**t**, x, a) = eval$ (**t**, cancel-iplus-ilessp-1 (x), a)

DEFINITION:
iplus-tree-no-fix-int (l)
= **if** listp (l) **then** iplus-tree-rec (l)
else `''0` **endif**

THEOREM: eval$-ilessp-iplus-tree-no-fix-int
ilessp (eval$ (**t**, iplus-tree-no-fix-int (x), a),
eval$ (**t**, iplus-tree-no-fix-int (y), a))
= ilessp (eval$ (**t**, iplus-tree (x), a), eval$ (**t**, iplus-tree (y), a))

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

THEOREM: make-cancel-iplus-inequality-simplifier
eval$ (**t**, make-cancel-iplus-inequality-1 (x, y), a)
= eval$ (**t**,
list (**`'ilessp`**,
iplus-tree-no-fix-int (bagdiff (x, bagint (x, y))),
iplus-tree-no-fix-int (bagdiff (y, bagint (x, y)))),
a)

DEFINITION:
cancel-iplus-ilessp (x)
= **if** listp (x)
then if car (x) = **`'ilessp`**
then if listp (bagint (iplus-fringe (cadr (x)),
iplus-fringe (caddr (x))))
then list (**`'ilessp`**,
iplus-tree-no-fix-int (bagdiff (iplus-fringe (cadr (x)),
bagint (iplus-fringe (cadr (x)),
iplus-fringe (caddr (x))))),

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
eval\$ (**t**, x, a) = eval\$ (**t**, cancel-iplus-ilessp (x), a)

THEOREM: itimes-zero1
(fix-int (x) = 0) → (itimes (x, y) = 0)

THEOREM: itimes-0-left
itimes (0, y) = 0

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

THEOREM: itimes-zero2
(fix-int (y) = 0) → (itimes (x, y) = 0)

THEOREM: itimes-0-right
itimes (x, 0) = 0

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

THEOREM: itimes-fix-int1
itimes (fix-int (a), b) = itimes (a, b)

THEOREM: itimes-fix-int2
itimes (a, fix-int (b)) = itimes (a, b)

THEOREM: commutativity-of-itimes
itimes (x, y) = itimes (y, x)

THEOREM: itimes-distributes-over-iplus-proof
itimes (x, iplus (y, z)) = iplus (itimes (x, y), itimes (x, z))

THEOREM: itimes-distributes-over-iplus
(itimes (x, iplus (y, z)) = iplus (itimes (x, y), itimes (x, z)))
∧ (itimes (iplus (x, y), z) = iplus (itimes (x, z), 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
$(\text{itimes}(x,\ y) = \mathtt{0}) = ((\text{fix-int}(x) = \mathtt{0}) \lor (\text{fix-int}(y) = \mathtt{0}))$

THEOREM: equal-itimes-1
$(\text{itimes}(a,\ b) = \mathtt{1})$
$= \quad (((a = \mathtt{1}) \land (b = \mathtt{1})) \lor ((a = \mathtt{-1}) \land (b = \mathtt{-1})))$

THEOREM: equal-itimes-minus-1
$(\text{itimes}(a,\ b) = \mathtt{-1})$
$= \quad (((a = \mathtt{-1}) \land (b = \mathtt{1})) \lor ((a = \mathtt{1}) \land (b = \mathtt{-1})))$

THEOREM: itimes-1-arg1
$\text{itimes}(\mathtt{1},\ x) = \text{fix-int}(x)$

THEOREM: quotient-remainder-uniqueness
$((a = (r + (b * q))) \land (r < b))$
$\rightarrow \quad ((\text{fix}(r) = (a \ \mathbf{mod}\ b)) \land (\text{fix}(q) = (a \div b)))$

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

THEOREM: division-theorem-part3
$(\text{integerp}(j) \land (j \neq \mathtt{0})) \rightarrow \text{ilessp}(\text{iremainder}(i,\ j),\ \text{iabs}(j))$

THEOREM: division-theorem
$(\text{integerp}(i) \land \text{integerp}(j) \land (j \neq \mathtt{0}))$
$\rightarrow \quad ((\text{iplus}(\text{iremainder}(i,\ j),\ \text{itimes}(j,\ \text{iquotient}(i,\ j))) = i)$
$\quad\quad \land \quad (\neg\ \text{ilessp}(\text{iremainder}(i,\ j),\ \mathtt{0}))$
$\quad\quad \land \quad \text{ilessp}(\text{iremainder}(i,\ j),\ \text{iabs}(j)))$

THEOREM: quotient-difference-lessp-arg2
$(((a \ \mathbf{mod}\ c) = \mathtt{0}) \land (b < c))$
$\rightarrow \quad (((a - b) \div c)$
$\quad\quad = \quad \mathbf{if}\ b \simeq \mathtt{0}\ \mathbf{then}\ a \div c$
$\quad\quad\quad\quad \mathbf{elseif}\ b < a\ \mathbf{then}\ (a \div c) - (1 + (b \div c))$
$\quad\quad\quad\quad \mathbf{else}\ \mathtt{0}\ \mathbf{endif})$

THEOREM: iquotient-iremainder-uniqueness

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{integerp}(i)} \\
\wedge & \text{integerp}(j) \\
\wedge & \text{integerp}(r) \\
\wedge & \text{integerp}(q) \\
\wedge & (j \neq 0) \\
\wedge & (i = \text{iplus}(r,\ \text{itimes}(j,\ q))) \\
\wedge & (\neg\ \text{ilessp}(r,\ 0)) \\
\wedge & \text{ilessp}(r,\ \text{iabs}(j))) \\
\rightarrow & ((r = \text{iremainder}(i,\ j)) \wedge (q = \text{iquotient}(i,\ j)))
\end{array}
$$

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

$$\text{integerp}(i) \rightarrow (\text{iplus}(\text{imod}(i,\ j),\ \text{itimes}(j,\ \text{idiv}(i,\ j))) = i)$$

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

$$\text{ilessp}(0,\ j) \rightarrow ((\neg\ \text{ilessp}(\text{imod}(i,\ j),\ 0)) \wedge \text{ilessp}(\text{imod}(i,\ j),\ j))$$

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

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{integerp}(j) \wedge \text{ilessp}(j,\ 0))} \\
\rightarrow & ((\neg\ \text{ilessp}(0,\ \text{imod}(i,\ j))) \wedge \text{ilessp}(j,\ \text{imod}(i,\ j)))
\end{array}
$$

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

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0))} \\
\rightarrow & ((\text{iplus}(\text{imod}(i,\ j),\ \text{itimes}(j,\ \text{idiv}(i,\ j))) = i) \\
& \ \wedge\ \ \textbf{if}\ \text{ilessp}(0,\ j) \\
& \qquad \textbf{then}\ (\neg\ \text{ilessp}(\text{imod}(i,\ j),\ 0)) \wedge \text{ilessp}(\text{imod}(i,\ j),\ j) \\
& \qquad \textbf{else}\ (\neg\ \text{ilessp}(0,\ \text{imod}(i,\ j))) \\
& \qquad\qquad \wedge\ \ \text{ilessp}(j,\ \text{imod}(i,\ j))\ \textbf{endif})
\end{array}
$$

THEOREM: idiv-imod-uniqueness

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{integerp}(i)} \\
\wedge & \text{integerp}(j) \\
\wedge & \text{integerp}(r) \\
\wedge & \text{integerp}(q) \\
\wedge & (j \neq 0) \\
\wedge & (i = \text{iplus}(r,\ \text{itimes}(j,\ q))) \\
\wedge & \textbf{if}\ \text{ilessp}(0,\ j)\ \ \textbf{then}\ (\neg\ \text{ilessp}(r,\ 0)) \wedge \text{ilessp}(r,\ j) \\
& \textbf{else}\ (\neg\ \text{ilessp}(0,\ r)) \wedge \text{ilessp}(j,\ r)\ \textbf{endif}) \\
\rightarrow & ((r = \text{imod}(i,\ j)) \wedge (q = \text{idiv}(i,\ j)))
\end{array}
$$

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

$$\text{integerp}(i) \rightarrow (\text{iplus}(\text{irem}(i,\ j),\ \text{itimes}(j,\ \text{iquo}(i,\ j))) = i)$$

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

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{integerp}(i) \wedge \text{integerp}(j) \wedge (j \neq 0) \wedge (\neg\ \text{ilessp}(i,\ 0)))} \\
\rightarrow & ((\neg\ \text{ilessp}(\text{irem}(i,\ j),\ 0)) \wedge \text{ilessp}(\text{irem}(i,\ j),\ \text{iabs}(j)))
\end{array}
$$

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

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

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

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
$(\text{itimes}\,(a,\ b) = \text{itimes}\,(a,\ c))$
$= \quad ((\text{fix-int}\,(a) = 0) \lor (\text{fix-int}\,(b) = \text{fix-int}\,(c)))$

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

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

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

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

THEOREM: integerp-idiv
integerp (idiv (i, j))

THEOREM: integerp-imod
integerp (imod (i, j))

THEOREM: integerp-iquo
integerp (iquo (i, j))

THEOREM: integerp-irem
integerp (irem (i, j))

THEOREM: iquotient-fix-int1
iquotient (fix-int (i), j) $=$ iquotient (i, j)

THEOREM: iquotient-fix-int2
iquotient (i, fix-int (j)) $=$ iquotient (i, j)

THEOREM: iremainder-fix-int1
iremainder (fix-int (i), j) $=$ iremainder (i, j)

THEOREM: iremainder-fix-int2
iremainder (i, fix-int (j)) $=$ iremainder (i, j)

THEOREM: idiv-fix-int1
idiv (fix-int (i), j) $=$ idiv (i, j)

THEOREM: idiv-fix-int2
idiv (i, fix-int (j)) $=$ idiv (i, j)

THEOREM: imod-fix-int1
imod (fix-int (i), j) $=$ imod (i, j)

THEOREM: imod-fix-int2
imod (i, fix-int (j)) $=$ imod (i, j)

THEOREM: iquo-fix-int1
iquo (fix-int (i), j) $=$ iquo (i, j)

THEOREM: iquo-fix-int2
iquo (i, fix-int (j)) $=$ iquo (i, j)

THEOREM: irem-fix-int1
irem (fix-int (i), j) $=$ irem (i, j)

THEOREM: irem-fix-int2
irem (i, fix-int (j)) $=$ irem (i, j)

THEOREM: fix-int-iquotient
fix-int (iquotient (i, j)) = iquotient (i, j)

THEOREM: fix-int-iremainder
fix-int (iremainder (i, j)) = iremainder (i, j)

THEOREM: fix-int-idiv
fix-int (idiv (i, j)) = idiv (i, j)

THEOREM: fix-int-imod
fix-int (imod (i, j)) = imod (i, j)

THEOREM: fix-int-iquo
fix-int (iquo (i, j)) = iquo (i, j)

THEOREM: fix-int-irem
fix-int (irem (i, j)) = 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:
itimes-fringe (x)
= **if** listp (x)
  then if car (x) = `'itimes`
    then append (itimes-fringe (cadr (x)), itimes-fringe (caddr (x)))
    else list (x) **endif**
  else list (x) **endif**

DEFINITION:
itimes-tree-rec (l)
= **if** cdr $(l) \simeq$ **nil then** car (l)
  else list (`'itimes`, car (l), itimes-tree-rec (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
itimes-list (append (x, y)) = itimes (itimes-list (x), itimes-list (y))

THEOREM: itimes-list-eval\$-fringe
itimes-list (eval\$ (`'list`, itimes-fringe (x), a)) = fix-int (eval\$ (**t**, x, a))

THEOREM: integerp-eval\$-itimes
(car (x) = `'itimes`) → integerp (eval\$ (**t**, x, a))

THEOREM: not-integerp-implies-not-equal-itimes
(¬ integerp (a)) → ((a = itimes (b, c)) = **f**)

THEOREM: itimes-list-eval\$-delete
($z \in y$)
→ (itimes-list (eval\$ (`'list`, y, a))
  = itimes (eval\$ (**t**, z, a), itimes-list (eval\$ (`'list`, delete (z, y), a))))

THEOREM: itimes-list-bagdiff
subbagp (x, y)
→ (itimes-list (eval\$ (`'list`, y, a))
  = itimes (itimes-list (eval\$ (`'list`, bagdiff (y, x), a)),
    itimes-list (eval\$ (`'list`, x, a))))

THEOREM: equal-itimes-list-eval\$-list-delete
(($c \in y$) ∧ (fix-int (eval\$ (**t**, c, a)) ≠ 0))
→ ((x = itimes-list (eval\$ (`'list`, delete (c, y), a)))
  = (integerp (x)
    ∧ (itimes (x, eval\$ (**t**, c, a))
      = itimes-list (eval\$ (`'list`, y, a)))))

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

THEOREM: member-append
($a \in$ append (x, y)) = (($a \in x$) ∨ ($a \in y$))

THEOREM: member-izerop-itimes-fringe
(($z \in$ itimes-fringe (x)) ∧ (fix-int (eval\$ (**t**, z, a)) = 0))
→ (fix-int (eval\$ (**t**, x, a)) = 0)

THEOREM: correctness-of-cancel-itimes-hack-1
$((w \in \text{itimes-fringe}(\text{cons}(\texttt{'itimes}, \mathit{x1})))$
$\land \quad (\text{fix-int}(\text{eval\$}(\mathbf{t}, w, a)) = \texttt{0})$
$\land \quad (\text{fix-int}(\text{eval\$}(\mathbf{t}, \text{car}(\mathit{x1}), a)) \neq \texttt{0}))$
$\rightarrow \quad (\text{fix-int}(\text{eval\$}(\mathbf{t}, \text{cadr}(\mathit{x1}), a)) = \texttt{0})$

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

THEOREM: eval\$-make-cancel-itimes-equality
$\text{eval\$}(\mathbf{t}, \text{make-cancel-itimes-equality}(x, y, \mathit{in\text{-}both}), a)$
$= \quad \textbf{if } \text{eval\$}(\mathbf{t}, \text{cons}(\texttt{'equal}, \text{cons}(\text{itimes-tree}(\mathit{in\text{-}both}), \texttt{'('0)})), a)$
$\quad \textbf{then } \mathbf{t}$
$\quad \textbf{else } \text{itimes-list}(\text{eval\$}(\texttt{'list}, \text{bagdiff}(x, \mathit{in\text{-}both}), a))$
$\qquad = \quad \text{itimes-list}(\text{eval\$}(\texttt{'list}, \text{bagdiff}(y, \mathit{in\text{-}both}), a)) \textbf{ endif}$

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

THEOREM: eval\$-make-cancel-itimes-equality-1
$\text{eval\$}(\mathbf{t}, \text{make-cancel-itimes-equality}(\text{list}(x), y, \text{list}(x)), a)$
$= \quad \textbf{if } \text{fix-int}(\text{eval\$}(\mathbf{t}, x, a)) = \texttt{0} \textbf{ then } \mathbf{t}$
$\quad \textbf{else } \texttt{1} = \text{itimes-list}(\text{eval\$}(\texttt{'list}, \text{delete}(x, y), a)) \textbf{ endif}$

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

THEOREM: eval\$-make-cancel-itimes-equality-2
$\text{eval\$}(\mathbf{t}, \text{make-cancel-itimes-equality}(x, \text{list}(y), \text{list}(y)), a)$
$= \quad \textbf{if } \text{fix-int}(\text{eval\$}(\mathbf{t}, y, a)) = \texttt{0} \textbf{ then } \mathbf{t}$
$\quad \textbf{else } \texttt{1} = \text{itimes-list}(\text{eval\$}(\texttt{'list}, \text{delete}(y, x), a)) \textbf{ endif}$

THEOREM: eval\$-equal-itimes-tree-itimes-fringe-0
$(\text{eval\$}(\mathbf{t}, \text{cons}(\texttt{'equal}, \text{cons}(\text{itimes-tree}(\text{itimes-fringe}(x)), \texttt{'('0)})), a)$
$\land \quad (\text{car}(x) = \texttt{'itimes}))$
$\rightarrow \quad (\text{eval\$}(\mathbf{t}, x, a) = \texttt{0})$

THEOREM: izerop-eval-of-member-implies-itimes-list-0
$((z \in y) \land (\text{fix-int}(\text{eval\$}(\mathbf{t}, z, a)) = \texttt{0}))$
$\rightarrow \quad (\text{itimes-list}(\text{eval\$}(\texttt{'list}, y, a)) = \texttt{0})$

DEFINITION:
$\text{subsetp}(x, y)$
$= \quad \textbf{if } x \simeq \textbf{nil} \textbf{ then } \mathbf{t}$
$\quad \textbf{elseif } \text{car}(x) \in y \textbf{ then } \text{subsetp}(\text{cdr}(x), y)$
$\quad \textbf{else } \mathbf{f} \textbf{ endif}$

THEOREM: subsetp-implies-itimes-list-eval\$-equals-0

$$\begin{array}{l}(\text{subsetp}\,(x,\ y) \land (\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ x,\ a)) = \texttt{0}))\\ \to\quad (\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ y,\ a)) = \texttt{0})\end{array}$$

THEOREM: subbagp-subsetp

$$\text{subbagp}\,(x,\ y) \to \text{subsetp}\,(x,\ y)$$

THEOREM: equal-0-itimes-list-eval\$-bagint-1

$$\begin{array}{l}(\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagint}\,(x,\ y),\ a)) = \texttt{0})\\ \to\quad (\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ x,\ a)) = \texttt{0})\end{array}$$

THEOREM: equal-0-itimes-list-eval\$-bagint-2

$$\begin{array}{l}(\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagint}\,(x,\ y),\ a)) = \texttt{0})\\ \to\quad (\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ y,\ a)) = \texttt{0})\end{array}$$

THEOREM: correctness-of-cancel-itimes-hack-2

$$\begin{array}{ll}(\text{listp}\,(u) & \\ \land & (\text{car}\,(u) = \texttt{'itimes})\\ \land & \text{listp}\,(v)\\ \land & (\text{car}\,(v) = \texttt{'itimes})\\ \land & (\text{eval\$}\,(\texttt{t},\ u,\ a) \neq \text{eval\$}\,(\texttt{t},\ v,\ a)))\\ \to & (\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\\ & \qquad\qquad \text{bagdiff}\,(\text{itimes-fringe}\,(u),\\ & \qquad\qquad\qquad \text{bagint}\,(\text{itimes-fringe}\,(u),\ \text{itimes-fringe}\,(v))),\\ & \qquad\qquad a))\\ & \quad \neq \quad \text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\\ & \qquad\qquad\qquad \text{bagdiff}\,(\text{itimes-fringe}\,(v),\\ & \qquad\qquad\qquad\qquad \text{bagint}\,(\text{itimes-fringe}\,(u),\\ & \qquad\qquad\qquad\qquad\qquad \text{itimes-fringe}\,(v))),\\ & \qquad\qquad\qquad a)))\end{array}$$

THEOREM: correctness-of-cancel-itimes-hack-3-lemma

$$\begin{array}{l}((u = \text{itimes}\,(a,\ b)) \land (\text{fix-int}\,(a) \neq \texttt{0}))\\ \to\quad ((u = \text{itimes}\,(a,\ c)) = (\text{fix-int}\,(b) = \text{fix-int}\,(c)))\end{array}$$

THEOREM: correctness-of-cancel-itimes-hack-3

$$\begin{array}{ll}(\text{listp}\,(u) & \\ \land & (\text{car}\,(u) = \texttt{'itimes})\\ \land & \text{listp}\,(v)\\ \land & (\text{car}\,(v) = \texttt{'itimes})\\ \land & (\text{eval\$}\,(\texttt{t},\ u,\ a) = \text{eval\$}\,(\texttt{t},\ v,\ a))\\ \land & (\neg\ \text{eval\$}\,(\texttt{t},\\ & \qquad \text{cons}\,(\texttt{'equal},\\ & \qquad\quad \text{cons}\,(\text{itimes-tree}\,(\text{bagint}\,(\text{itimes-fringe}\,(u),\\ & \qquad\qquad\qquad\qquad \text{itimes-fringe}\,(v))),\end{array}$$

```
                                          '('0))),
                      a)))
→  ((itimes-list (eval$ ('list,
                            bagdiff (itimes-fringe (u),
                                     bagint (itimes-fringe (u), itimes-fringe (v))),
                            a))
      =   itimes-list (eval$ ('list,
                               bagdiff (itimes-fringe (v),
                                        bagint (itimes-fringe (u),
                                                itimes-fringe (v))),
                               a)))
     =   t)
```

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
eval$ (**t**, x, a) = eval$ (**t**, cancel-itimes (x), a)

Definition:
itimes-tree-no-fix-int (l)
= **if** listp (l) **then** itimes-tree-rec (l)
  else ''1 **endif**

Theorem: eval$-itimes-tree-no-fix-int-1
ilessp (eval$ (**t**, itimes-tree-no-fix-int (x), a), y)
= ilessp (eval$ (**t**, itimes-tree (x), a), y)

Theorem: eval$-itimes-tree-no-fix-int-2
ilessp (y, eval$ (**t**, itimes-tree-no-fix-int (x), a))
= ilessp (y, eval$ (**t**, itimes-tree (x), a))

Event: Enable itimes-tree-no-fix-int; name this event ‘itimes-tree-no-fix-int-off’.

Definition:
make-cancel-itimes-inequality (x, y, *in-both*)

```
=  list ('if,
         cons ('ilessp, cons (itimes-tree-no-fix-int (in-both), '('0))),
         list ('ilessp,
               itimes-tree-no-fix-int (bagdiff (y, in-both)),
               itimes-tree-no-fix-int (bagdiff (x, in-both))),
         cons ('if,
               cons (list ('ilessp, ''0, itimes-tree-no-fix-int (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 endif
         else x endif
    else x endif
```

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 ilessp (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 ilessp (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 endif
```

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

THEOREM: ilessp-itimes-right-positive
ilessp (0, x) $\rightarrow$ (ilessp (y, z) = ilessp (itimes (y, x), itimes (z, x)))

THEOREM: correctness-of-cancel-itimes-ilessp-hack-1
(subbagp (*bag*, x)

$\land$ subbagp $(bag,\ y)$
$\land$ ilessp $(0,$ itimes-list (eval\$ ('list, $bag,\ a))))$
$\rightarrow$ (ilessp (itimes-list (eval\$ ('list, bagdiff $(x,\ bag),\ a)),$
itimes-list (eval\$ ('list, bagdiff $(y,\ bag),\ a)))$
$=$ ilessp (itimes-list (eval\$ ('list, $x,\ a)),$
itimes-list (eval\$ ('list, $y,\ a))))$

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

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

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

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

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

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

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

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

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

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

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

$\wedge$ subbagp (*bag*, *y*)
$\wedge$ ilessp (itimes-list (eval\$ (`'list`, *bag*, *a*)), `0`))
$\rightarrow$ (ilessp (itimes-list (eval\$ (`'list`, bagdiff (*x*, *bag*), *a*)),
itimes-list (eval\$ (`'list`, bagdiff (*y*, *bag*), *a*)))
= ilessp (itimes-list (eval\$ (`'list`, *y*, *a*)),
itimes-list (eval\$ (`'list`, *x*, *a*))))

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
eval\$ (**t**, *x*, *a*) = eval\$ (**t**, cancel-itimes-ilessp (*x*), *a*)

THEOREM: ilessp-strict
ilessp (*x*, *y*) $\rightarrow$ ($\neg$ 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 (`'ineg`, 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
itimes (-1, x) = ineg (x)

THEOREM: equal-ineg-ineg
(ineg (x) = ineg (y)) = (fix-int (x) = fix-int (y))

THEOREM: ilessp-ineg-ineg
ilessp (ineg (x), ineg (y)) = ilessp (y, x)

THEOREM: fix-int-eval$-itimes-tree-rec
listp (x)
→ (fix-int (eval$ (t, itimes-tree-rec (x), a))
   = itimes-list (eval$ ('list, x, a)))

THEOREM: eval$-itimes-tree-ineg
fix-int (eval$ (t, itimes-tree-ineg (x), a)) = itimes-list (eval$ ('list, x, a))

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

THEOREM: iplus-eval$-itimes-tree-ineg
(iplus (eval$ (t, itimes-tree-ineg (x), a), y)
 = iplus (itimes-list (eval$ ('list, x, a)), y))
∧ (iplus (y, eval$ (t, itimes-tree-ineg (x), a))
   = iplus (y, itimes-list (eval$ ('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))
 ∧   (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 = iplus
           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) ∧ (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)
```

```
 (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))
(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))
```

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

THEOREM: itimes-list-eval\$-factors-lemma
$$\begin{array}{l}\text{itimes}\,(\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagint}\,(bag1,\ bag2),\ a)),\\ \qquad\quad \text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagdiff}\,(bag2,\ \text{bagint}\,(bag1,\ bag2)),\ a)))\\ = \quad \text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ bag2,\ a))\end{array}$$

THEOREM: itimes-list-eval\$-factors-lemma-prime
$$\begin{array}{l}\text{itimes}\,(\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagint}\,(bag1,\ bag2),\ a)),\\ \qquad\quad \text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagdiff}\,(bag1,\ \text{bagint}\,(bag1,\ bag2)),\ a)))\\ = \quad \text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ bag1,\ a))\end{array}$$

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

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

THEOREM: eval\$-list-bagint-0-implies-equal
$$\begin{array}{l}((\text{itimes-list}\,(\text{eval\$}\,(\texttt{'list},\ \text{bagint}\,(\text{itimes-factors}\,(v),\ \text{itimes-factors}\,(w)),\ a))\\ \quad = \quad 0)\\ \ \wedge \quad \text{integerp}\,(\text{eval\$}\,(\mathbf{t},\ v,\ a))\\ \ \wedge \quad \text{integerp}\,(\text{eval\$}\,(\mathbf{t},\ w,\ a)))\\ \rightarrow \quad ((\text{eval\$}\,(\mathbf{t},\ v,\ a) = \text{eval\$}\,(\mathbf{t},\ w,\ a)) = \mathbf{t})\end{array}$$

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

itimes-factors (caddr (x))))
else x **endif**
else x **endif**
else x **endif**

THEOREM: bagint-singleton
bagint (x, list (y))
= **if** $y \in x$ **then** list (y)
else nil endif

THEOREM: izerop-ilessp-0-relationship
(fix-int (x) = 0) = (($\neg$ ilessp (x, 0)) $\wedge$ ($\neg$ ilessp (0, x)))

THEOREM: ilessp-itimes-list-eval$-list-delete-helper-1
ilessp (0, w) $\rightarrow$ (ilessp (itimes (x, w), itimes (w, u)) = ilessp (x, u))

THEOREM: ilessp-itimes-list-eval$-list-delete-helper-2
ilessp (w, 0) $\rightarrow$ (ilessp (itimes (w, u), itimes (x, w)) = ilessp (x, u))

THEOREM: ilessp-itimes-list-eval$-list-delete
(($z \in y$) $\wedge$ (fix-int (eval$ (**t**, z, a)) $\neq$ 0))
$\rightarrow$ (ilessp (x, itimes-list (eval$ ('list, delete (z, y), a)))
= **if** ilessp (0, eval$ (**t**, z, a))
then ilessp (itimes (x, eval$ (**t**, z, a)),
itimes-list (eval$ ('list, y, a)))
elseif ilessp (eval$ (**t**, z, a), 0)
then ilessp (itimes-list (eval$ ('list, y, a)),
itimes (x, eval$ (**t**, z, a)))
else f endif)

THEOREM: ilessp-itimes-list-eval$-list-delete-prime-helper-1
ilessp (0, w) $\rightarrow$ (ilessp (itimes (w, u), itimes (x, w)) = ilessp (u, x))

THEOREM: ilessp-itimes-list-eval$-list-delete-prime-helper-2
ilessp (w, 0) $\rightarrow$ (ilessp (itimes (x, w), itimes (w, u)) = ilessp (u, x))

THEOREM: ilessp-itimes-list-eval$-list-delete-prime
(($z \in y$) $\wedge$ (fix-int (eval$ (**t**, z, a)) $\neq$ 0))
$\rightarrow$ (ilessp (itimes-list (eval$ ('list, delete (z, y), a)), x)
= **if** ilessp (0, eval$ (**t**, z, a))
then ilessp (itimes-list (eval$ ('list, y, a)),
itimes (x, eval$ (**t**, z, a)))
elseif ilessp (eval$ (**t**, z, a), 0)
then ilessp (itimes (x, eval$ (**t**, z, a)),
itimes-list (eval$ ('list, y, a)))
else f endif)

THEOREM: ilessp-0-itimes
$\text{ilessp}(\mathtt{0},\ \text{itimes}(x,\ y))$
$= \quad ((\text{ilessp}(\mathtt{0},\ x) \land \text{ilessp}(\mathtt{0},\ y)) \lor (\text{ilessp}(x,\ \mathtt{0}) \land \text{ilessp}(y,\ \mathtt{0})))$

THEOREM: ilessp-itimes-0
$\text{ilessp}(\text{itimes}(x,\ y),\ \mathtt{0})$
$= \quad ((\text{ilessp}(\mathtt{0},\ x) \land \text{ilessp}(y,\ \mathtt{0})) \lor (\text{ilessp}(x,\ \mathtt{0}) \land \text{ilessp}(\mathtt{0},\ y)))$

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff
$(\text{subbagp}(\textit{in-both},\ \textit{bag1})$
$\land \quad \text{subbagp}(\textit{in-both},\ \textit{bag2})$
$\land \quad (\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{in-both},\ a)) \neq \mathtt{0}))$
$\rightarrow \quad (\text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag1},\ \textit{in-both}),\ a)),$
$\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag2},\ \textit{in-both}),\ a)))$
$\qquad = \quad \textbf{if}\ \text{ilessp}(\mathtt{0},\ \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{in-both},\ a)))$
$\qquad\quad \textbf{then}\ \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag1},\ a)),$
$\qquad\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag2},\ a)))$
$\qquad\quad \textbf{else}\ \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag2},\ a)),$
$\qquad\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag1},\ a)))\ \textbf{endif})$

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

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff-corollary-1
$(\text{subbagp}(\textit{in-both},\ \textit{bag1})$
$\land \quad \text{subbagp}(\textit{in-both},\ \textit{bag2})$
$\land \quad \text{ilessp}(\mathtt{0},\ \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{in-both},\ a))))$
$\rightarrow \quad (\text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag1},\ \textit{in-both}),\ a)),$
$\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag2},\ \textit{in-both}),\ a)))$
$\qquad = \quad \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag1},\ a)),$
$\qquad\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag2},\ a))))$

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

THEOREM: ilessp-itimes-list-eval\$-list-bagdiff-corollary-2
$(\text{subbagp}(\textit{in-both},\ \textit{bag1})$
$\land \quad \text{subbagp}(\textit{in-both},\ \textit{bag2})$
$\land \quad \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{in-both},\ a)),\ \mathtt{0}))$
$\rightarrow \quad (\text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag1},\ \textit{in-both}),\ a)),$
$\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagdiff}(\textit{bag2},\ \textit{in-both}),\ a)))$
$\qquad = \quad \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag2},\ a)),$
$\qquad\qquad \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \textit{bag1},\ a))))$

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

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

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

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
$$\begin{array}{ll}
((\neg\ \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagint}(x,\ y),\ a)),\ \mathtt{0})) & \\
\ \wedge\quad (\neg\ \text{ilessp}(\mathtt{0},\ \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagint}(x,\ y),\ a))))) & \\
\rightarrow\quad ((\text{fix-int}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ x,\ a))) = \mathtt{0}) & \\
\qquad \wedge\quad (\text{fix-int}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ y,\ a))) = \mathtt{0})) &
\end{array}$$

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

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
$$\begin{array}{l}
((\neg\ \text{ilessp}(\text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagint}(\text{itimes-factors}(v),\ \text{itimes-factors}(w)),\ a)),\ \mathtt{0})) \\
\ \wedge\quad (\neg\ \text{ilessp}(\mathtt{0},\ \text{itimes-list}(\text{eval\$}(\mathtt{'list},\ \text{bagint}(\text{itimes-factors}(v),\ \text{itimes-factors}(w)),\ a))))) \\
\rightarrow\quad ((\neg\ \text{ilessp}(\text{eval\$}(\mathtt{t},\ v,\ a),\ \text{eval\$}(\mathtt{t},\ w,\ a))) \\
\qquad \wedge\quad (\neg\ \text{ilessp}(\text{eval\$}(\mathtt{t},\ w,\ a),\ \text{eval\$}(\mathtt{t},\ v,\ a))))
\end{array}$$

EVENT: Enable ilessp-trichotomy; name this event 'ilessp-trichotomy-off1'.

THEOREM: correctness-of-cancel-itimes-ilessp-factors
eval$ (**t**, x, a) = eval$ (**t**, cancel-itimes-ilessp-factors (x), a)

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

DEFINITION:
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:
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:
some-eval$s-to-0 ($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
listp (*lst*)

$\rightarrow$ (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) $\vee$ 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))) $\rightarrow$ 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 caddr (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 ilessp (0, itimes-list (eval$ ('list, x, a)))
          else ilessp (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 ilessp-itimes-list-eval$-list-delete-helper-1; name this event 'ilessp-itimes-list-eval$-list-delete-helper-1-off'.

EVENT: Enable ilessp-itimes-list-eval$-list-delete-helper-2; name this event 'ilessp-itimes-list-eval$-list-delete-helper-2-off'.

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

EVENT: Enable ilessp-itimes-list-eval$-list-delete-prime-helper-1; name this event 'ilessp-itimes-list-eval$-list-delete-prime-helper-1-off'.

EVENT: Enable ilessp-itimes-list-eval$-list-delete-prime-helper-2; name this event 'ilessp-itimes-list-eval$-list-delete-prime-helper-2-off'.

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

EVENT: Enable ilessp-0-itimes; name this event 'ilessp-0-itimes-off'.

EVENT: Enable ilessp-itimes-0; name this event 'ilessp-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`)
    $\land$ negativep (cadar (x))
    $\land$ (negative-guts (cadar (x)) $\neq$ `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)))
       ∨   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) ∧ ((car (x) = 'ineg) ∨ (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) ∧ (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,
                                                        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)) endcase
                             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
```

```
case = iplus
 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,
                            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)) endcase
      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
otherwise if 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 = iplus
                         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 = iplus
             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
iplus-or-ineg-term $(x) \rightarrow$ integerp (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
remove-inegs (x, y)
$\rightarrow$ (iplus-list (eval$ (`'list`, car (remove-inegs (x, y)), a))
= iplus (iplus-list (eval$ (`'list`, car (split-out-ineg-terms (x)), a)),
iplus-list (eval$ (`'list`, cdr (split-out-ineg-terms (y)), a))))

THEOREM: eval$-iplus-list-cdr-remove-inegs
remove-inegs (x, y)
$\rightarrow$ (iplus-list (eval$ (`'list`, cdr (remove-inegs (x, y)), a))
= iplus (iplus-list (eval$ (`'list`, car (split-out-ineg-terms (y)), a)),
iplus-list (eval$ (`'list`, cdr (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$-car-split-out-ineg-terms
iplus-list (eval$ (`'list`, car (split-out-ineg-terms (x)), a))
= iplus (iplus-list (eval$ (`'list`, x, a)),
iplus-list (eval$ (`'list`, cdr (split-out-ineg-terms (x)), a)))

EVENT: Enable remove-inegs; name this event 'remove-inegs-off'.

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

DEFINITION:
make-cancel-ineg-terms-inequality (x)
= **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**

DEFINITION:
cancel-ineg-terms-from-inequality (x)
= **if** listp (x) $\wedge$ (car (x) = `'ilessp`)
then if iplus-or-ineg-term (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 = 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 if 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 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\$ (**t**, x, a) $=$ eval\$ (**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
$((i \in \mathbf{N}) \land (j \in \mathbf{N})) \rightarrow ((i + j) = \text{iplus}(i, j))$

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
$$
\begin{array}{ll}
((x \in \mathbf{N}) & \\
\land & (y \in \mathbf{N}) \\
\land & \mathbf{if}\ y < x\ \mathbf{then}\ \mathbf{f} \\
& \mathbf{else}\ \mathbf{t}\ \mathbf{endif}) \\
\rightarrow & ((y - x) = \text{idifference}(y, x))
\end{array}
$$

THEOREM: cancel-constants-equal-lemma
$$
\begin{array}{lll}
((m \in \mathbf{N}) \land (n \in \mathbf{N})) & & \\
\rightarrow & ((\text{iplus}(m, x) = \text{iplus}(n, y)) & \\
& = & \mathbf{if}\ m < n\ \mathbf{then}\ \text{fix-int}(x) = \text{iplus}(n - m, y) \\
& & \mathbf{else}\ \text{iplus}(m - n, x) = \text{fix-int}(y)\ \mathbf{endif})
\end{array}
$$

THEOREM: cancel-constants-equal
$$
\begin{array}{ll}
(\text{iplus}(1 + i, x) = \text{iplus}(1 + j, y)) & \\
= & \mathbf{if}\ i < j\ \mathbf{then}\ \text{fix-int}(x) = \text{iplus}(j - i, y) \\
& \mathbf{else}\ \text{iplus}(i - j, x) = \text{fix-int}(y)\ \mathbf{endif}
\end{array}
$$

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
$$
\begin{array}{lll}
((m \in \mathbf{N}) \land (n \in \mathbf{N})) & & \\
\rightarrow & (\text{ilessp}(\text{iplus}(m, x), \text{iplus}(n, y)) & \\
& = & \mathbf{if}\ m < n\ \mathbf{then}\ \text{ilessp}(x, \text{iplus}(n - m, y)) \\
& & \mathbf{else}\ \text{ilessp}(\text{iplus}(m - n, x), y)\ \mathbf{endif})
\end{array}
$$

THEOREM: cancel-constants-ilessp-lemma-2
$$
\begin{array}{lll}
((m \in \mathbf{N}) \land (n \in \mathbf{N})) & & \\
\rightarrow & (\text{ilessp}(\text{iplus}(m, x), \text{iplus}(n, y)) & \\
& = & \mathbf{if}\ m < n\ \mathbf{then}\ \neg\ \text{ilessp}(\text{iplus}((n - m) - 1, y), x) \\
& & \mathbf{else}\ \text{ilessp}(\text{iplus}(m - n, x), y)\ \mathbf{endif})
\end{array}
$$

THEOREM: cancel-constants-ilessp
ilessp (iplus $(1 + i,\ x)$, iplus $(1 + j,\ y)$)
$=$ **if** $i < j$ **then** $\neg$ ilessp (iplus $((j - i) - 1,\ y),\ x)$
else ilessp (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, ineg-iplus, ineg-ineg, ineg-fix-int, ineg-of-non-integerp, ineg-0, iplus-left-id, iplus-right-id, iplus-0-left, iplus-0-right, commutativity2-of-iplus, commutativity-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$-list-append, iplus-list-append, iplus-ineg3, iplus-ineg4, correctness-of-cancel-iplus, ilessp-fix-int-1, ilessp-fix-int-2, iplus-cancellation-1-for-ilessp, iplus-cancellation-2-for-ilessp, correctness-of-cancel-iplus-ilessp, itimes-0-left, itimes-0-right, itimes-fix-int1, itimes-fix-int2, commutativity-of-itimes, itimes-distributes-over-iplus-proof, itimes-distributes-over-iplus, commutativity2-of-itimes, associativity-of-itimes, equal-itimes-0, equal-itimes-1, equal-itimes-minus-1, itimes-1-arg1, quotient-remainder-uniqueness, division-theorem, itimes-ineg-1, itimes-ineg-2, itimes-cancellation-1, itimes-cancellation-2, itimes-cancellation-3, integerp-iquotient, integerp-iremainder, integerp-idiv, integerp-imod, integerp-iquo, integerp-irem, iquotient-fix-int1, iquotient-fix-int2, iremainder-fix-int1, iremainder-fix-int2, idiv-fix-int1, idiv-fix-int2, imod-fix-int1, imod-fix-int2, iquo-fix-int1, iquo-fix-int2, irem-fix-int1, irem-fix-int2, fix-int-iquotient, fix-int-iremainder, fix-int-idiv, fix-int-imod, fix-int-iquo, fix-int-irem, itimes-list, itimes-list-append, member-append, equal-fix-int, subsetp, correctness-of-cancel-itimes, correctness-of-cancel-itimes-ilessp, ilessp-strict, eval$-list-cons, eval$-list-nlistp, eval$-litatom, eval$-quote, eval$-other, iplus-x-y-ineg-x, correctness-of-cancel-ineg, integerp-iplus-list, eval$-iplus-list-delete, eval$-iplus-list-bagdiff, itimes-tree-ineg, itimes-factors, itimes–1, equal-ineg-ineg, ilessp-ineg-ineg, fix-int-eval$-itimes-tree-rec, eval$-itimes-tree-ineg, ineg-eval$-itimes-tree-ineg, iplus-eval$-itimes-tree-ineg, itimes-eval$-itimes-tree-ineg, iplus-or-itimes-term, cancel-itimes-factors, cancel-itimes-factors-expanded, cancel-itimes-factors-expanded-cancel-itimes-factors, membership-of-0-implies-itimes-list-is-0, member-0-eval$-list, correctness-of-cancel-itimes-factors, cancel-itimes-ilessp-factors, bagint-singleton, ilessp-itimes-list-eval$-list-bagdiff, ilessp-itimes-list-eval$-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, ineg-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, ilessp-strict, ilessp-itimes-right-negative, ilessp-itimes-right-positive, cancel-itimes-ilessp, *1*cancel-itimes-ilessp, *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-iremainder, 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-iremainder, integerp-iquotient, itimes-cancellation-3, itimes-cancellation-2, itimes-cancellation-1, itimes-ineg-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-iremainder-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-ilessp, cancel-iplus-ilessp, *1*cancel-iplus-ilessp, *1*iplus-tree-no-fix-int, iplus-cancellation-2-for-ilessp, iplus-cancellation-1-for-ilessp, cancel-iplus-ilessp-1, *1*cancel-iplus-ilessp-1, *1*make-cancel-iplus-inequality-1, ilessp-fix-int-2, ilessp-fix-int-1, correctness-of-cancel-iplus, iplus-ineg7, 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*iremainder, *1*iquotient, *1*itimes, *1*iabs, idifference, *1*idifference, *1*ineg, *1*iplus, ileq, *1*ileq, *1*ilessp, 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$-equal, and-not-zerop-tree, *1*and-not-zerop-tree, or-zerop-tree, *1*or-zerop-tree, times-fringe, *1*times-fringe, times-tree, *1*times-tree, lessp-1-times, lessp-plus-times2, lessp-plus-times1, lessp-plus-times-proof, lessp-times-cancellation-proof, lessp-times3, lessp-times3-proof2, lessp-times3-proof1, lessp-times2, lessp-times2-proof, lessp-times1, lessp-times1-proof, times-1-arg1, times-quotient, times-quotient-proof, times-distributes-over-difference, times-distributes-over-difference-proof, associativity-of-times, commutativity2-of-times, times-distributes-over-plus, times-distributes-over-plus-proof, commutativity-of-times, times-add1, times-zero, equal-sub1-0, equal-times-1, equal-times-0, correctness-of-cancel-lessp-plus, cancel-lessp-plus, *1*cancel-lessp-plus, difference-x-x, difference-difference-arg2, difference-difference-arg1, difference-sub1-arg2, difference-add1-arg2, difference-leq-arg1, difference-elim, correctness-of-cancel-difference-plus, cancel-difference-plus, *1*cancel-difference-plus, correctness-of-cancel-equal-plus, cancel-equal-plus, *1*cancel-equal-plus, eval$-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 ilessp-add1-iplus; name this event 'ilessp-add1-iplus-off'.

EVENT: Enable ilessp-add1; name this event 'ilessp-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 ineg-eval$-itimes-tree-ineg; name this event 'ineg-eval$-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 ilessp-ineg-ineg; name this event 'ilessp-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 ilessp-strict; name this event ‘ilessp-strict-off’.

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

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

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

EVENT: Enable *1*cancel-itimes-ilessp; name this event ‘g*1*cancel-itimes-ilessp-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 ilessp-fix-int-2; name this event 'ilessp-fix-int-2-off'.

EVENT: Enable ilessp-fix-int-1; name this event 'ilessp-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*iquo; name this event 'g*1*iquo-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*iquotient; name this event 'g*1*iquotient-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'.

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'.

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:
if* $(a,\ b,\ c)$
= **if** a **then** b
else c **endif**

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

DEFINITION: or* $(a,\ b)$ = if* $(a,\ \mathbf{t},$ if* $(b,\ \mathbf{t},\ \mathbf{f}))$

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*
$(\text{and*}(\mathbf{f},\ x) = \mathbf{f})$
$\wedge \quad (\text{and*}(x,\ \mathbf{f}) = \mathbf{f})$
$\wedge \quad (y \rightarrow ((\text{and*}(y,\ x) = \text{if*}(x, \mathbf{t}, \mathbf{f})) \wedge (\text{and*}(x,\ y) = \text{if*}(x, \mathbf{t}, \mathbf{f}))))$

THEOREM: rewrite-or*
$(\text{or*}(\mathbf{f},\ x) = \text{if*}(x, \mathbf{t}, \mathbf{f}))$
$\wedge \quad (\text{or*}(x,\ \mathbf{f}) = \text{if*}(x, \mathbf{t}, \mathbf{f}))$
$\wedge \quad (y \rightarrow ((\text{or*}(y,\ x) = \mathbf{t}) \wedge (\text{or*}(x,\ y) = \mathbf{t})))$

THEOREM: expand-*-connectives
(if* $(c,\ a,\ b)$
 = **if** c **then** a
 else b **endif**)
$\wedge$ (or* $(a,\ b)$
 = **if** a **then t**
 elseif b **then t**
 else f endif)
$\wedge$ (and* $(a,\ b)$
 = **if** a
 then if b **then t**

else f endif
else f endif)
$\wedge$ (not* $(a) = (\neg\ a)$)

EVENT: Enable expand-*-connectives; name this event 'expand-*-connectives-off'.

THEOREM: car-cdr-if-cons
(car (**if** x **then** cons $(a,\ b)$
else cons $(c,\ d)$ **endif**)
= **if** x **then** a
else c **endif**)
$\wedge$ (cdr (**if** x **then** cons $(a,\ b)$
else cons $(c,\ d)$ **endif**)
= **if** x **then** b
else d **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
$((zero \simeq 0) \rightarrow ((x + zero) = \text{fix}(x)))$
$\wedge\ ((zero \simeq 0) \rightarrow ((zero + x) = \text{fix}(x)))$

THEOREM: plus-add1
$(((1 + x) + y) = (1 + (x + y)))$
$\wedge\ ((x + (1 + y)) = (1 + (x + y)))$

THEOREM: plus-bottom
$((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$

THEOREM: plus-add1-sub1
$((n \not\simeq 0) \rightarrow (((n - 1) + (1 + m)) = (n + m)))$
$\wedge\ ((n \not\simeq 0) \rightarrow (((1 + m) + (n - 1)) = (m + n)))$

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
$((x - y) = 0)$
$= \quad$ **if** $y < x$ **then f**
else t endif

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
if $y < x$ **then f**
else t endif
$\rightarrow \quad ((x - y) = 0)$

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
$((1 < base) \wedge (n \not\simeq 0)) \rightarrow ((n \div base) < n)$

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
$((x - 1) < x) = (x \not\simeq 0)$

THEOREM: lessp-x-1
$(x < 1) = (x \simeq 0)$

THEOREM: leq-lessp-difference
(**if** $y < x$ **then f**
else t endif
$\wedge \quad (z \not\simeq 0)$
$\wedge \quad (y \not\simeq 0))$
$\rightarrow \quad (((x - z) < y) = \mathbf{t})$

EVENT: Enable append; name this event 'append-off'.

DEFINITION:
append5 $(a, b, c, d, e) =$ append $(a,$ append $(b,$ append $(c,$ append $(d, e))))$

DEFINITION:
append6 (a, b, c, d, e, g)
$=$ append $(a,$ append $(b,$ append $(c,$ append $(d,$ append $(e, g)))))$

DEFINITION:
append7 (a, b, c, d, e, g, h)
$=$ append $(a,$ append $(b,$ append $(c,$ append $(d,$ append $(e,$ append $(g, h))))))$

DEFINITION:
append8 (a, b, c, d, e, g, h, i)
$=$ append $(a,$
append $(b,$ append $(c,$ append $(d,$ append $(e,$ append $(g,$ append $(h, i)))))))$

THEOREM: append-nlistp
$(a \simeq \mathbf{nil}) \rightarrow$ (append $(a, b) = b)$

THEOREM: append-cons
append (cons $(a, b), c) =$ cons $(a,$ append $(b, c))$

THEOREM: our-member-append
$(a \in$ append $(b, c)) = ((a \in b) \vee (a \in c))$

THEOREM: associativity-of-append
append (append $(a, b), c) =$ append $(a,$ append $(b, c))$

THEOREM: equal-append-x-x
(append $(a, b) =$ append $(a, c)) = (b = c)$

DEFINITION:
properp (l)
$=$ **if** listp (l) **then** properp (cdr (l))
else $l =$ **nil endif**

EVENT: Enable properp; name this event 'properp-off'.

THEOREM: properp-if
(properp $(a) \wedge$ properp (b))
$\rightarrow$ properp (**if** c **then** a
else b **endif**)

THEOREM: properp-nlistp
$(l \simeq$ **nil**$) \rightarrow$ (properp $(l) = (l =$ **nil**$)$)

THEOREM: properp-cons
properp (cons (x, y)) $=$ properp (y)

THEOREM: properp-append
properp (append (a, b)) $=$ properp (b)

THEOREM: properp-append-nil
properp $(a) \rightarrow$ (append (a, **nil**) $= a$)

THEOREM: properp-if*
(properp $(a) \wedge$ properp (b)) $\rightarrow$ properp (if* (c, a, b))

DEFINITION:
position (x, l)
$=$ **if** $l \simeq$ **nil** **then f**
elseif $x =$ car (l) **then** 0
elseif position (x, cdr (l)) **then** $1 +$ position (x, cdr (l))
else f endif

EVENT: Enable position; name this event 'position-off'.

THEOREM: member==>position
$(x \in l) \rightarrow$ position (x, l)

DEFINITION:
length (l)
$=$ **if** listp (l) **then** $1 +$ length (cdr (l))
else 0 **endif**

EVENT: Enable length; name this event 'length-off'.

THEOREM: length-nlistp
$(l \simeq \mathbf{nil}) \rightarrow (\text{length}\,(l) = \mathtt{0})$

THEOREM: length-bottom
$(\text{length}\,(x) = \mathtt{0}) = (x \simeq \mathbf{nil})$

THEOREM: length-1
$(\text{length}\,(a) = \mathtt{1}) = (\text{listp}\,(a) \wedge (\text{cdr}\,(a) \simeq \mathbf{nil}))$

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
$((\text{length}\,(\text{cdr}\,(x)) = \text{length}\,(x)) = (x \simeq \mathbf{nil}))$
$\wedge \quad ((\text{length}\,(\text{cdr}\,(x)) < \text{length}\,(x)) = \text{listp}\,(x))$

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
$(\text{length}\,(a) = \text{length}\,(b))$
$\rightarrow \quad (\text{length}\,(\mathbf{if}\ c\ \mathbf{then}\ a$
$\qquad\qquad\qquad \mathbf{else}\ b\ \mathbf{endif})$
$\qquad = \quad \text{length}\,(a))$

THEOREM: length-if*
$(\text{length}\,(a) = \text{length}\,(b)) \rightarrow (\text{length}\,(\text{if*}\,(c,\ a,\ b)) = \text{length}\,(a))$

DEFINITION:
$\text{rev1}\,(x,\ \textit{sponge})$
$= \quad \mathbf{if}\ x \simeq \mathbf{nil}\ \mathbf{then}\ \textit{sponge}$
$\qquad \mathbf{else}\ \text{rev1}\,(\text{cdr}\,(x),\ \text{cons}\,(\text{car}\,(x),\ \textit{sponge}))\ \mathbf{endif}$

EVENT: Enable rev1; name this event 'rev1-off'.

DEFINITION: $\text{reverse}(x) = \text{rev1}(x, \textbf{nil})$

EVENT: Enable reverse; name this event 'reverse-off'.

THEOREM: length-rev1
$\text{length}(\text{rev1}(x, \textit{sponge})) = (\text{length}(x) + \text{length}(\textit{sponge}))$

THEOREM: length-reverse
$\text{length}(\text{reverse}(x)) = \text{length}(x)$

DEFINITION:
$\text{delete*}(x, l)$
$= \ \textbf{if}\ l \simeq \textbf{nil}\ \textbf{then}\ l$
$\quad \textbf{elseif}\ x = \text{car}(l)\ \textbf{then}\ \text{delete*}(x, \text{cdr}(l))$
$\quad \textbf{else}\ \text{cons}(\text{car}(l), \text{delete*}(x, \text{cdr}(l)))\ \textbf{endif}$

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

DEFINITION:
$\text{subset}(l1, l2)$
$= \ \textbf{if}\ l1 \simeq \textbf{nil}\ \textbf{then}\ \textbf{t}$
$\quad \textbf{else}\ (\text{car}(l1) \in l2) \land \text{subset}(\text{cdr}(l1), l2)\ \textbf{endif}$

EVENT: Enable subset; name this event 'subset-off'.

THEOREM: subset-nlistp
$(x \simeq \textbf{nil}) \rightarrow \text{subset}(x, y)$

THEOREM: subset-cons
$\text{subset}(\text{cons}(x, y), z) = ((x \in z) \land \text{subset}(y, z))$

THEOREM: subset-x-cons-y-z
$\text{subset}(x, \text{cons}(y, z))$
$= \ (\text{subset}(x, z) \lor ((y \in x) \land \text{subset}(\text{delete*}(y, x), z)))$

THEOREM: subset-append
$(\text{subset}(a, b) \lor \text{subset}(a, c)) \rightarrow \text{subset}(a, \text{append}(b, c))$

THEOREM: subset-x-x
$\text{subset}(x, x)$

DEFINITION:
$\text{disjoint}(l1, l2)$
$= \ \textbf{if}\ l1 \simeq \textbf{nil}\ \textbf{then}\ \textbf{t}$
$\quad \textbf{else}\ (\text{car}(l1) \notin l2) \land \text{disjoint}(\text{cdr}(l1), l2)\ \textbf{endif}$

EVENT: Enable disjoint; name this event 'disjoint-off'.

THEOREM: disjoint-nlistp
$((x \simeq \textbf{nil}) \lor (y \simeq \textbf{nil})) \rightarrow \text{disjoint}(x, y)$

THEOREM: disjoint-cons
$(\text{disjoint}(\text{cons}(x, y), z) = ((x \notin z) \land \text{disjoint}(y, z)))$
$\land \quad (\text{disjoint}(z, \text{cons}(x, y)) = ((x \notin z) \land \text{disjoint}(z, y)))$

THEOREM: disjoint-append
$(\text{disjoint}(x, \text{append}(y, z)) = (\text{disjoint}(x, y) \land \text{disjoint}(x, z)))$
$\land \quad (\text{disjoint}(\text{append}(y, z), x) = (\text{disjoint}(y, x) \land \text{disjoint}(z, x)))$

DEFINITION:
duplicates? (l)
$= \quad$ **if** $l \simeq$ **nil then f**
else $(\text{car}(l) \in \text{cdr}(l)) \lor \text{duplicates?}(\text{cdr}(l))$ **endif**

EVENT: Enable duplicates?; name this event 'duplicates?-off'.

THEOREM: duplicates?-cons
$\text{duplicates?}(\text{cons}(x, y)) = ((x \in y) \lor \text{duplicates?}(y))$

THEOREM: duplicates-append
$\text{duplicates?}(\text{append}(a, b))$
$= \quad (\text{duplicates?}(a) \lor \text{duplicates?}(b) \lor (\neg\, \text{disjoint}(a, b)))$

DEFINITION:
firstn (n, l)
$= \quad$ **if** listp (l)
then if $n \simeq 0$ **then nil**
else cons (car (l), firstn $(n - 1$, cdr $(l)))$ **endif**
else nil endif

EVENT: Enable firstn; name this event 'firstn-off'.

THEOREM: length-firstn
length (firstn (n, l))
$= \quad$ **if if** length $(l) < n$ **then f**
else t endif then fix (n)
else length (l) **endif**

THEOREM: length-firstn1
if length $(l) < n$ **then f**
else t endif
$\rightarrow \quad (\text{length}(\text{firstn}(n, l)) = \text{fix}(n))$

THEOREM: length-firstn2
$(\neg$ **if** $\text{length}(l) < n$ **then f**
else t endif$)$
$\rightarrow \quad (\text{length}(\text{firstn}(n, l)) = \text{length}(l))$

THEOREM: car-firstn
$\text{car}(\text{firstn}(n, a))$
$= \quad$ **if** $\text{listp}(a) \wedge (n \not\simeq 0)$ **then** $\text{car}(a)$
else 0 **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
$(\text{firstn}(n, l) = \textbf{nil}) = ((l \simeq \textbf{nil}) \vee (n \simeq 0))$

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:
$\text{restn}(n, l)$
$= \quad$ **if** $\text{listp}(l)$
then if $n \simeq 0$ **then** l
else $\text{restn}(n - 1, \text{cdr}(l))$ **endif**
else l **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
$(\text{properp}\,(l)$
$\wedge\quad \textbf{if}\ n < \text{length}\,(l)\ \ \textbf{then f}$
$\qquad \textbf{else t endif})$
$\rightarrow\quad (\text{restn}\,(n,\ l) = \textbf{nil})$

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
$(\text{disjoint}\,(x,\ y) \rightarrow \text{disjoint}\,(\text{firstn}\,(n,\ x),\ \text{firstn}\,(m,\ y)))$
$\wedge\quad (\text{disjoint}\,(x,\ y) \rightarrow \text{disjoint}\,(\text{firstn}\,(n,\ x),\ \text{restn}\,(m,\ y)))$
$\wedge\quad (\text{disjoint}\,(x,\ y) \rightarrow \text{disjoint}\,(\text{restn}\,(n,\ x),\ \text{firstn}\,(m,\ y)))$
$\wedge\quad (\text{disjoint}\,(x,\ y) \rightarrow \text{disjoint}\,(\text{restn}\,(n,\ x),\ \text{restn}\,(m,\ y)))$

THEOREM: no-duplicates-disjoint-firstn-restn
$((\neg\ \text{duplicates?}\,(x)) \rightarrow \text{disjoint}\,(\text{firstn}\,(n,\ x),\ \text{restn}\,(n,\ x)))$
$\wedge\quad ((\neg\ \text{duplicates?}\,(x)) \rightarrow \text{disjoint}\,(\text{restn}\,(n,\ x),\ \text{firstn}\,(n,\ x)))$

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
$$\begin{aligned}&(\text{length}(a) = \text{length}(c))\\ \rightarrow\quad &(\text{pairlist}(\text{append}(a,\ b),\ \text{append}(c,\ d))\\ &\quad = \quad \text{append}(\text{pairlist}(a,\ c),\ \text{pairlist}(b,\ d)))\end{aligned}$$

THEOREM: properp-pairlist
$\text{properp}(\text{pairlist}(x,\ y))$

THEOREM: pairlists-are-equal-when-their-2nd-lists-are-nlistp
$((\mathit{l1} \simeq \mathbf{nil}) \land (\mathit{l2} \simeq \mathbf{nil})) \rightarrow ((\text{pairlist}(a,\ \mathit{l1}) = \text{pairlist}(a,\ \mathit{l2})) = \mathbf{t})$

DEFINITION:
$$\begin{aligned}&\text{nth}(n,\ \mathit{list})\\ =\quad &\mathbf{if}\ n \simeq 0\ \ \mathbf{then}\ \text{car}(\mathit{list})\\ &\mathbf{else}\ \text{nth}(n-1,\ \text{cdr}(\mathit{list}))\ \mathbf{endif}\end{aligned}$$

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

THEOREM: open-nth
$$\begin{aligned}&((n \simeq 0) \rightarrow (\text{nth}(n,\ \mathit{list}) = \text{car}(\mathit{list})))\\ \land\quad &((n \not\simeq 0) \rightarrow (\text{nth}(n,\ \mathit{list}) = \text{nth}(n-1,\ \text{cdr}(\mathit{list}))))\end{aligned}$$

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

THEOREM: nth-if
$$\begin{aligned}&\text{nth}(n,\\ &\qquad \mathbf{if}\ a\ \ \mathbf{then}\ b\\ &\qquad \mathbf{else}\ c\ \mathbf{endif})\\ =\quad &\mathbf{if}\ a\ \ \mathbf{then}\ \text{nth}(n,\ b)\\ &\mathbf{else}\ \text{nth}(n,\ c)\ \mathbf{endif}\end{aligned}$$

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
$\textbf{if } n < \text{length}(a) \textbf{ then f}$
$\textbf{else t endif}$
$\rightarrow \quad (\text{nth}(n, \text{append}(a, b)) = \text{nth}(n - \text{length}(a), b))$

DEFINITION:
$\text{nthcdr}(n, l)$
$= \quad \textbf{if } n \simeq \mathtt{0} \textbf{ then } l$
$\quad\quad \textbf{else } \text{nthcdr}(n - 1, \text{cdr}(l)) \textbf{ endif}$

EVENT: Enable nthcdr; name this event 'nthcdr-off'.

THEOREM: open-nthcdr
$((n \simeq \mathtt{0}) \rightarrow (\text{nthcdr}(n, l) = l))$
$\wedge \quad ((n \not\simeq \mathtt{0}) \rightarrow (\text{nthcdr}(n, l) = \text{nthcdr}(n - 1, \text{cdr}(l))))$

THEOREM: properp-as-null-nthcdr
$\text{properp}(l) = (\text{nthcdr}(\text{length}(l), l) = \textbf{nil})$

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:
$\text{subrange}(l, n, m)$
$= \quad \textbf{if } m < n \textbf{ then nil}$
$\quad\quad \textbf{elseif } n \simeq \mathtt{0}$
$\quad\quad \textbf{then if } m \simeq \mathtt{0} \textbf{ then } \text{list}(\text{car}(l))$
$\quad\quad\quad\quad \textbf{else } \text{cons}(\text{car}(l), \text{subrange}(\text{cdr}(l), \mathtt{0}, m - 1)) \textbf{ endif}$
$\quad\quad \textbf{else } \text{subrange}(\text{cdr}(l), n - 1, m - 1) \textbf{ endif}$

EVENT: Enable subrange; name this event 'subrange-off'.

THEOREM: properp-subrange
$\text{properp}(\text{subrange}(v, n, m))$

THEOREM: subrange-cons
subrange (cons (car, cdr), n, m)
= **if** $m < n$ **then nil**
elseif $n \simeq 0$
then if $m \simeq 0$ **then** list (car)
else cons (car, subrange (cdr, 0, $m - 1$)) **endif**
else subrange (cdr, $n - 1$, $m - 1$) **endif**

EVENT: Enable subrange-cons; name this event 'subrange-cons-off'.

THEOREM: length-subrange
length (subrange (l, n, m))
= **if** $m < n$ **then** 0
else $1 + (m - n)$ **endif**

THEOREM: subrange-append-right
(($m \not< n$)
∧ **if** $n <$ length (a) **then f**
else t endif)
→ (subrange (append (a, b), n, m)
= subrange (b, $n -$ length (a), $m -$ length (a)))

THEOREM: subrange-append-left
(($m \not< n$) ∧ ($n <$ length (a)) ∧ ($m <$ length (a)))
→ (subrange (append (a, b), n, m) = subrange (a, n, m))

THEOREM: subrange-0
(($m =$ (length (a) $- 1$)) ∧ properp (a) ∧ (length (a) $\not\simeq 0$))
→ (subrange (a, 0, m) = a)

THEOREM: listp-subrange
listp (subrange (l, n, m)) = ($m \not< n$)

THEOREM: subset-subrange
($n <$ length (v)) → subset (subrange (v, m, n), v)

THEOREM: not-member-subrange
(($n <$ length (v)) ∧ ($x \notin v$)) → ($x \notin$ subrange (v, m, n))

THEOREM: disjoint-subrange
(disjoint (x, y) ∧ ($m <$ length (y)))
→ (disjoint (x, subrange (y, n, m)) ∧ disjoint (subrange (y, n, m), x))

THEOREM: open-subrange
(($m < n$) → (subrange (l, n, m) = **nil**))

$\wedge\quad (((m \not< n) \wedge (n \simeq 0) \wedge (m \simeq 0))$
$\quad\to\quad (\text{subrange}(l,\ n,\ m) = \text{list}(\text{car}(l))))$
$\wedge\quad (((m \not< n) \wedge (n \simeq 0) \wedge (m \not\simeq 0))$
$\quad\to\quad (\text{subrange}(l,\ n,\ m) = \text{cons}(\text{car}(l),\ \text{subrange}(\text{cdr}(l),\ 0,\ m - 1))))$
$\wedge\quad (((m \not< n) \wedge (n \not\simeq 0))$
$\quad\to\quad (\text{subrange}(l,\ n,\ m) = \text{subrange}(\text{cdr}(l),\ n - 1,\ m - 1)))$

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
$((n \simeq 0) \to (\text{make-list}(n,\ x) = \textbf{nil}))$
$\wedge\quad ((n \not\simeq 0) \to (\text{make-list}(n,\ x) = \text{cons}(x,\ \text{make-list}(n - 1,\ x))))$

EVENT: Enable open-make-list; name this event 'open-make-list-off'.

THEOREM: length-make-list
length (make-list (n, $value$)) = fix (n)

THEOREM: make-list-append
append (make-list (n, $value$), make-list (m, $value$)) = make-list ($n + m$, $value$)

THEOREM: properp-make-list
properp (make-list (n, $value$))

THEOREM: firstn-make-list
if $m < n$ **then f**
else t endif
$\to\quad (\text{firstn}(n,\ \text{make-list}(m,\ v)) = \text{make-list}(n,\ v))$

DEFINITION:
tree-size $(tree)$
$=$ **if** $tree \simeq$ **nil** **then** `1`
else tree-size (car $(tree)$) + tree-size (cdr $(tree)$) **endif**

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

THEOREM: tree-size-nlistp
$(tree \simeq \mathbf{nil}) \rightarrow (\text{tree-size}\,(tree) = \mathtt{1})$

THEOREM: not-equal-tree-size-tree-0
$\text{tree-size}\,(tree) \neq \mathtt{0}$

THEOREM: tree-size-1-crock
$\mathtt{1} \neq \text{tree-size}\,(\text{cons}\,(a,\ b))$

THEOREM: a-helpful-lemma-for-tree-inductions
$(\text{length}\,(a) = \text{tree-size}\,(tree))$
$\rightarrow \quad ((\text{length}\,(a) < \text{tree-size}\,(\text{car}\,(tree))) = \mathbf{f})$

THEOREM: tree-size-lemmas
$((\text{listp}\,(tree) \land (size = \text{tree-size}\,(tree)))$
$\rightarrow \quad ((size - \text{tree-size}\,(\text{car}\,(tree))) = \text{tree-size}\,(\text{cdr}\,(tree))))$
$\land \quad ((\text{listp}\,(tree) \land (size = \text{tree-size}\,(tree)))$
$\rightarrow \quad ((size - \text{tree-size}\,(\text{cdr}\,(tree))) = \text{tree-size}\,(\text{car}\,(tree))))$

THEOREM: make-list-append-tree-crock
$\text{listp}\,(tree)$
$\rightarrow \quad (\text{make-list}\,(\text{tree-size}\,(\text{car}\,(tree)) + \text{tree-size}\,(\text{cdr}\,(tree)),\ value)$
$= \quad \text{make-list}\,(\text{tree-size}\,(tree),\ value))$

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:
tree-height $(tree)$
$=$ **if** $tree \simeq$ **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
$(n \not\simeq 0) \rightarrow$ (tree-size (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), cadddr (a)) $= a$)

EVENT: Enable list-rewrite-4; name this event 'list-rewrite-4-off'.

THEOREM: list-elim-4
($l =$ list (a, b, c, d))
$=$ ((car $(l) = a$)
$\wedge$ (cadr $(l) = b$)
$\wedge$ (caddr $(l) = c$)
$\wedge$ (cadddr $(l) = d$)
$\wedge$ (cddddr $(l) =$ **nil**))

EVENT: Enable list-elim-4; name this event 'list-elim-4-off'.

DEFINITION:
list-as-collected-nth (l, length, n)
$=$ **if** $\mathit{length} \simeq 0$ **then nil**
else cons (nth (n, l), list-as-collected-nth $(l, \mathit{length} - 1, 1 + n)$) **endif**

EVENT: Enable list-as-collected-nth; name this event 'list-as-collected-nth-off'.

THEOREM: open-list-as-collected-nth
(($\mathit{length} \simeq 0$) $\rightarrow$ (list-as-collected-nth $(l, \mathit{length}, n) =$ **nil**))

$\wedge \quad ((\mathit{length} \not\simeq 0)$
$\quad \rightarrow \quad (\text{list-as-collected-nth}\,(l,\ \mathit{length},\ n)$
$\qquad = \quad \text{cons}\,(\text{nth}\,(n,\ l),$
$\qquad\qquad \text{list-as-collected-nth}\,(l,\ \mathit{length} - 1,\ 1 + n))))$

THEOREM: equal-length-4-as-collected-nth
$((\text{length}\,(l) = 4) \wedge \text{properp}\,(l))$
$\rightarrow \quad (l = \text{list-as-collected-nth}\,(l,\ 4,\ 0))$

THEOREM: equal-length-32-as-collected-nth
$((\text{length}\,(l) = 32) \wedge \text{properp}\,(l))$
$\rightarrow \quad (l = \text{list-as-collected-nth}\,(l,\ 32,\ 0))$

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:
$\text{lnfix}\,(x)$
$= \quad \textbf{if}\ (x \in \mathbf{N}) \vee \text{litatom}\,(x)\ \ \textbf{then}\ x$
$\quad\ \ \textbf{else}\ 0\ \textbf{endif}$

DEFINITION:
$\text{indices}\,(\mathit{name},\ \mathit{from},\ n)$
$= \quad \textbf{if}\ n \simeq 0\ \ \textbf{then nil}$
$\quad\ \ \textbf{else}\ \text{cons}\,(\text{index}\,(\mathit{name},\ \mathit{from}),\ \text{indices}\,(\mathit{name},\ 1 + \mathit{from},\ n - 1))\ \textbf{endif}$

EVENT: Enable indices; name this event 'indices-off'.

THEOREM: indices-zerop
$(n \simeq 0) \rightarrow (\text{indices}\,(\mathit{name},\ \mathit{from},\ n) = \textbf{nil})$

THEOREM: open-indices
$(n \not\simeq 0)$
$\rightarrow \quad (\text{indices}\,(\mathit{name},\ \mathit{from},\ n)$
$\qquad = \quad \text{cons}\,(\text{index}\,(\mathit{name},\ \mathit{from}),\ \text{indices}\,(\mathit{name},\ 1 + \mathit{from},\ n - 1)))$

THEOREM: indices-as-append
$(n \not\simeq 0)$
$\rightarrow \quad (\text{indices}\,(\mathit{name},\ \mathit{from},\ n)$
$\qquad = \quad \text{append}\,(\text{indices}\,(\mathit{name},\ \mathit{from},\ n - 1),$
$\qquad\qquad \text{list}\,(\text{index}\,(\mathit{name},\ (n - 1) + \mathit{from}))))$

EVENT: Enable indices-as-append; name this event 'indices-as-append-off'.

THEOREM: length-indices
length (indices (*name*, *from*, *n*)) = fix (*n*)

THEOREM: listp-indices
listp (indices (*name*, *from*, *n*)) = ($n \not\simeq 0$)

THEOREM: properp-indices
properp (indices (*name*, *from*, *n*))

THEOREM: member-indices
($x \in$ indices (*name*, *from*, *n*))
= (($n \not\simeq 0$)
  ∧ indexp (x)
  ∧ (i-name (x) = lnfix (*name*))
  ∧ **if** i-num (x) < *from* **then f**
    else t endif
  ∧ (i-num (x) < (*from* + *n*)))

THEOREM: disjoint-indices-different-names
(lnfix (*name1*) ≠ lnfix (*name2*))
→ disjoint (indices (*name1*, *from1*, *n1*), indices (*name2*, *from2*, *n2*))

THEOREM: no-duplicates-in-indices
¬ duplicates? (indices (*name*, *from*, *n*))

THEOREM: position-name-indices
(*index* ∈ indices (*name*, *from*, *n*))
→ (position (*index*, indices (*name*, *from*, *n*)) = (i-num (*index*) − *from*))

THEOREM: nth-indices
($n < m$) → (nth (n, indices (*name*, *from*, m)) = index (*name*, n + *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: boolp (x) = ((x = **t**) ∨ (x = **f**))

EVENT: Enable boolp; name this event 'boolp-off'.

THEOREM: prove-boolp
(truep (x) ∨ falsep (x)) → boolp (x)

THEOREM: boolp-lemmas
$\text{boolp}(\mathbf{t})$
$\wedge \quad \text{boolp}(\mathbf{f})$
$\wedge \quad (\text{boolp}(b) \rightarrow ((b = \mathbf{t}) = b))$
$\wedge \quad (\text{boolp}(b) \rightarrow ((b = \mathbf{f}) = (\neg\ b)))$

THEOREM: boolp-implies-not-equal-z
$\text{boolp}(x) \rightarrow (x \neq \text{z})$

EVENT: Enable boolp-implies-not-equal-z; name this event 'boolp-implies-not-equal-z-off'.

DEFINITION:
$\text{boolfix}(x)$
$= \quad$ **if** x **then t**
else f endif

DEFINITION: $\text{threep}(x) = ((x = \mathbf{t}) \vee (x = \mathbf{f}) \vee (x = \text{X}))$

DEFINITION:
$\text{threefix}(x)$
$= \quad$ **if** $\text{boolp}(x)$ **then** x
else X **endif**

THEOREM: threefix-idempotent
$\text{threefix}(\text{threefix}(x)) = \text{threefix}(x)$

THEOREM: threefix=x
$(\text{X} = \text{threefix}(x)) = (\neg\ \text{boolp}(x))$

EVENT: Enable threefix=x; name this event 'threefix=x-off'.

DEFINITION:
$\text{fourp}(x) = ((x = \mathbf{t}) \vee (\mathbf{f} = x) \vee (x = \text{X}) \vee (x = \text{z}))$

DEFINITION:
$\text{fourfix}(x)$
$= \quad$ **if** $\text{fourp}(x)$ **then** x
else X **endif**

DEFINITION:
$\text{bvp}(x)$
$= \quad$ **if** $x \simeq \textbf{nil}$ **then** $x = \textbf{nil}$
else $\text{boolp}(\text{car}(x)) \wedge \text{bvp}(\text{cdr}(x))$ **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
$\text{bvp}\,(\text{cons}\,(x,\ y)) = (\text{boolp}\,(x) \land \text{bvp}\,(y))$

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
$\text{bvp}\,(\text{make-list}\,(n,\ v)) = ((n \simeq \mathtt{0}) \lor \text{boolp}\,(v))$

THEOREM: bvp-nthcdr
$(\text{bvp}\,(l)$
$\land\ \ \mathbf{if}\ \text{length}\,(l) < n\ \ \mathbf{then\ f}$
$\quad\ \ \mathbf{else\ t\ endif})$
$\rightarrow\ \ \text{bvp}\,(\text{nthcdr}\,(n,\ l))$

THEOREM: bvp-if
$(\text{bvp}\,(a) \land \text{bvp}\,(b))$
$\rightarrow\ \ \text{bvp}\,(\mathbf{if}\ c\ \ \mathbf{then}\ a$
$\qquad\qquad \mathbf{else}\ b\ \mathbf{endif})$

THEOREM: bvp-if*
$(\text{bvp}\,(a) \land \text{bvp}\,(b)) \rightarrow \text{bvp}\,(\text{if*}\,(c,\ a,\ b))$

DEFINITION:
$\text{bv2p}\,(x,\ y) = (\text{bvp}\,(x) \land \text{bvp}\,(y) \land (\text{length}\,(x) = \text{length}\,(y)))$

DEFINITION:
$\text{bvp-length}\,(bvp,\ n)$
$=\ \ (\text{bvp}\,(bvp)$
$\quad \land\ \ \mathbf{if}\ \text{length}\,(bvp) < n\ \ \mathbf{then\ f}$
$\qquad\ \ \mathbf{else\ t\ endif})$

EVENT: Enable bvp-length; name this event 'bvp-length-off'.

THEOREM: bvp-length-cdr
bvp-length $(x,\ 1 + n) \rightarrow$ bvp-length (cdr $(x),\ n)$

THEOREM: boolfix-car-x=x
bvp-length $(x,\ \mathtt{1}) \rightarrow$ (boolfix (car (x)) $=$ car (x))

THEOREM: boolp-car-x
bvp-length $(x,\ \mathtt{1}) \rightarrow$ boolp (car (x))

THEOREM: bvp-length-restn
bvp (bvp)
$\rightarrow$ (bvp-length (restn $(n,\ bvp),\ m$)
$=$ **if** (length $(bvp) - n) < m$ **then f**
else t endif)

THEOREM: show-that-nth=t
(nth $(n,\ a) \wedge$ bvp $(a) \wedge (n <$ length $(a))) \rightarrow$ (nth $(n,\ a) =$ **t**)

EVENT: Enable show-that-nth=t; name this event 'show-that-nth=t-off'.

THEOREM: boolp-nth
bvp $(a) \rightarrow$ (boolp (nth $(n,\ a)) = (n <$ length $(a)))$

THEOREM: another-way-to-look-at-boolp-nth
(bvp $(v) \wedge (n <$ length $(v)))$
$\rightarrow$ ((**if** nth $(n,\ v)$ **then t**
else f endif
$=$ nth $(n,\ v))$
$=$ **t**)

DEFINITION:
xor $(a,\ b)$
$=$ **if** a
then if b **then f**
else t endif
elseif b **then t**
else f endif

DEFINITION:
b-buf (x)
$=$ **if** x **then t**
else f endif

DEFINITION: b-not$(x) = (\neg\ x)$

DEFINITION: b-nand$(a,\ b) = (\neg\ (a \land b))$

DEFINITION: b-nand3$(a,\ b,\ c) = (\neg\ (a \land b \land c))$

DEFINITION: b-nand4$(a,\ b,\ c,\ d) = (\neg\ (a \land b \land c \land d))$

DEFINITION:
b-nand5$(a,\ b,\ c,\ d,\ e) = (\neg\ (a \land b \land c \land d \land e))$

DEFINITION:
b-nand6$(a,\ b,\ c,\ d,\ e,\ g) = (\neg\ (a \land b \land c \land d \land e \land g))$

DEFINITION:
b-nand8$(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)$
$= \quad (\neg\ (a \land b \land c \land d \land e \land g \land h \land i))$

DEFINITION: b-or$(a,\ b) = (a \lor b)$

DEFINITION: b-or3$(a,\ b,\ c) = (a \lor b \lor c)$

DEFINITION: b-or4$(a,\ b,\ c,\ d) = (a \lor b \lor c \lor d)$

DEFINITION:
b-xor$(x,\ y)$
= **if** x
then if y **then f**
else t endif
elseif y **then t**
else f endif

DEFINITION: b-xor3$(a,\ b,\ c) =$ b-xor (b-xor$(a,\ b),\ c)$

DEFINITION:
b-equv$(x,\ y)$
= **if** x
then if y **then t**
else f endif
elseif y **then f**
else t endif

DEFINITION: b-equv3$(a,\ b,\ c) =$ b-equv $(a,$ b-xor$(b,\ c))$

DEFINITION: b-and$(a,\ b) = (a \land b)$

DEFINITION: b-and3$(a,\ b,\ c) = (a \land b \land c)$

DEFINITION: b-and4 $(a, b, c, d) = (a \wedge b \wedge c \wedge d)$

DEFINITION: b-nor $(a, b) = (\neg (a \vee b))$

DEFINITION: b-nor3 $(a, b, c) = (\neg (a \vee b \vee c))$

DEFINITION: b-nor4 $(a, b, c, d) = (\neg (a \vee b \vee c \vee d))$

DEFINITION:
b-nor5 $(a, b, c, d, e) = (\neg (a \vee b \vee c \vee d \vee e))$

DEFINITION:
b-nor6 $(a, b, c, d, e, g) = (\neg (a \vee b \vee c \vee d \vee e \vee g))$

DEFINITION:
b-nor8 (a, b, c, d, e, g, h, i)
$= \quad (\neg (a \vee b \vee c \vee d \vee e \vee g \vee h \vee i))$

DEFINITION:
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
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
b-xor3 (a, b, c) = b-xor $(a$, b-xor $(b, c))$

THEOREM: open-b-equv
b-equv (a, b)
= **if** a
then if b **then t**
else f endif
elseif b **then f**
else t endif

THEOREM: open-b-equv3
b-equv3 (a, b, c) = b-equv $(a$, b-xor $(b, c))$

THEOREM: b-and-rewrite
$(\neg\ \text{b-and}\,(\mathbf{f},\ x)) \land (\neg\ \text{b-and}\,(x,\ \mathbf{f})) \land ((x \land y) \rightarrow \text{b-and}\,(x,\ y))$

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-equv, b-equv3, 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-equv, open-b-equv3, b-and-rewrite, b-buf-x=x.

THEOREM: boolp-b-gates
$$
\begin{aligned}
& \text{boolp}\,(\text{b-buf}\,(x)) \\
\land\quad & \text{boolp}\,(\text{b-not}\,(x)) \\
\land\quad & \text{boolp}\,(\text{b-nand}\,(a,\ b)) \\
\land\quad & \text{boolp}\,(\text{b-nand3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-nand4}\,(a,\ b,\ c,\ d)) \\
\land\quad & \text{boolp}\,(\text{b-nand5}\,(a,\ b,\ c,\ d,\ e)) \\
\land\quad & \text{boolp}\,(\text{b-nand6}\,(a,\ b,\ c,\ d,\ e,\ g)) \\
\land\quad & \text{boolp}\,(\text{b-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)) \\
\land\quad & \text{boolp}\,(\text{b-or}\,(a,\ b)) \\
\land\quad & \text{boolp}\,(\text{b-or3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-or4}\,(a,\ b,\ c,\ d)) \\
\land\quad & \text{boolp}\,(\text{b-xor}\,(x,\ y)) \\
\land\quad & \text{boolp}\,(\text{b-xor3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-equv}\,(x,\ y)) \\
\land\quad & \text{boolp}\,(\text{b-equv3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-and}\,(a,\ b)) \\
\land\quad & \text{boolp}\,(\text{b-and3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-and4}\,(a,\ b,\ c,\ d)) \\
\land\quad & \text{boolp}\,(\text{b-nor}\,(a,\ b)) \\
\land\quad & \text{boolp}\,(\text{b-nor3}\,(a,\ b,\ c)) \\
\land\quad & \text{boolp}\,(\text{b-nor4}\,(a,\ b,\ c,\ d)) \\
\land\quad & \text{boolp}\,(\text{b-nor5}\,(a,\ b,\ c,\ d,\ e)) \\
\land\quad & \text{boolp}\,(\text{b-nor6}\,(a,\ b,\ c,\ d,\ e,\ g)) \\
\land\quad & \text{boolp}\,(\text{b-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)) \\
\land\quad & \text{boolp}\,(\text{b-if}\,(c,\ a,\ b))
\end{aligned}
$$

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: $\mathrm{ao4}(a,\ b,\ c,\ d) = \mathrm{b\text{-}nand}(\mathrm{b\text{-}or}(a,\ b),\ \mathrm{b\text{-}or}(c,\ d))$

DEFINITION: $\mathrm{ao6}(a,\ b,\ c) = \mathrm{b\text{-}nor}(\mathrm{b\text{-}and}(a,\ b),\ c)$

DEFINITION: $\mathrm{ao7}(a,\ b,\ c) = \mathrm{b\text{-}nand}(\mathrm{b\text{-}or}(a,\ b),\ c)$

DEFINITION: $\mathrm{VSS} = \mathbf{f}$

DEFINITION: $\mathrm{VDD} = \mathbf{t}$

DEFINITION:
v-buf (x)
$=$ **if** $x \simeq$ **nil then nil**
else cons (b-buf (car (x)), v-buf (cdr (x))) **endif**

EVENT: Enable v-buf; name this event 'v-buf-off'.

DEFINITION:
v-not (x)
$=$ **if** $x \simeq$ **nil then nil**
else cons (b-not (car (x)), v-not (cdr (x))) **endif**

EVENT: Enable v-not; name this event 'v-not-off'.

DEFINITION:
v-and $(x,\ y)$
$=$ **if** $x \simeq$ **nil then nil**
else cons (b-and (car (x), car (y)), v-and (cdr (x), cdr (y))) **endif**

EVENT: Enable v-and; name this event 'v-and-off'.

DEFINITION:
v-or $(x,\ y)$
$=$ **if** $x \simeq$ **nil then nil**
else cons (b-or (car (x), car (y)), v-or (cdr (x), cdr (y))) **endif**

EVENT: Enable v-or; name this event 'v-or-off'.

DEFINITION:
v-xor $(x,\ y)$
$=$ **if** $x \simeq$ **nil then nil**
else cons (b-xor (car (x), car (y)), v-xor (cdr (x), cdr (y))) **endif**

EVENT: Enable v-xor; name this event 'v-xor-off'.

DEFINITION:
v-shift-right $(a,\ si)$
$=$ **if** $a \simeq$ **nil then nil**
else append (v-buf (cdr (a)), list (boolfix (si))) **endif**

EVENT: Enable v-shift-right; name this event 'v-shift-right-off'.

DEFINITION: v-lsr (a) = v-shift-right $(a,$ **f**$)$

DEFINITION: v-ror $(a,\ si)$ = v-shift-right $(a,\ si)$

DEFINITION: v-asr (a) = v-shift-right $(a,$ nth (length $(a) - 1,\ a))$

DEFINITION:
v-if $(c,\ a,\ b)$
$=$ **if** $a \simeq$ **nil then nil**
else cons (**if if** c **then** car (a)
else car (b) **endif then t**
else f endif,
v-if $(c$, cdr (a), cdr (b))) **endif**

EVENT: Enable v-if; name this event 'v-if-off'.

THEOREM: bvp-v-buf
bvp (v-buf (a))

THEOREM: bvp-v-not
bvp (v-not (a))

THEOREM: bvp-v-and
bvp (v-and $(a,\ b)$)

THEOREM: bvp-v-or
bvp (v-or $(a,\ b)$)

THEOREM: bvp-v-xor
bvp (v-xor $(a,\ b)$)

THEOREM: bvp-v-shift-right
bvp (v-shift-right $(a,\ si)$)

THEOREM: bvp-v-lsr
bvp (v-lsr (a))

THEOREM: bvp-v-asr
bvp (v-asr (a))

THEOREM: bvp-v-ror
bvp (v-ror (a, c))

THEOREM: bvp-v-if
bvp (v-if (c, a, b))

THEOREM: length-v-buf
length (v-buf (a)) = length (a)

THEOREM: length-v-not
length (v-not (a)) = length (a)

THEOREM: length-v-and
length (v-and (a, b)) = length (a)

THEOREM: length-v-or
length (v-or (a, b)) = length (a)

THEOREM: length-v-xor
length (v-xor (a, b)) = length (a)

THEOREM: length-v-shift-right
length (v-shift-right (a, b)) = length (a)

THEOREM: length-v-lsr
length (v-lsr (a)) = length (a)

THEOREM: length-v-asr
length (v-asr (a)) = length (a)

THEOREM: length-v-ror
length (v-ror (a, b)) = length (a)

THEOREM: length-v-if
length (v-if (c, a, b)) = length (a)

THEOREM: v-and-append-help
(length (a) = length (b))
$\rightarrow$ (append (v-and (a, b), v-and (d, e)) = v-and (append (a, d), append (b, e)))

THEOREM: v-or-append-help
(length (a) = length (b))
$\rightarrow$ (append (v-or (a, b), v-or (d, e)) = v-or (append (a, d), append (b, e)))

THEOREM: v-xor-append-help
(length (a) = length (b))
$\rightarrow$ (append (v-xor (a, b), v-xor (d, e)) = v-xor (append (a, d), 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
$(\text{length}(a) = \text{length}(b))$
$\rightarrow\ \ (\text{append}(\text{v-if}(c,\ a,\ b),\ \text{v-if}(c,\ d,\ e)) = \text{v-if}(c,\ \text{append}(a,\ d),\ \text{append}(b,\ e)))$

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
$(\text{bvp}(a) \land (\text{length}(a) = n)) \rightarrow (\text{v-or}(\text{make-list}(n,\ \mathbf{f}),\ a) = a)$

DEFINITION:
v-to-nat (v)
= **if** $v \simeq$ **nil** **then** 0
else if car (v) **then** 1
else 0 **endif**
$+\ \ (2 * \text{v-to-nat}(\text{cdr}(v)))$ **endif**

EVENT: Enable v-to-nat; name this event 'v-to-nat-off'.

DEFINITION:
nat-to-v (x, n)
$=$ **if** $n \simeq 0$ **then nil**
else cons $((x \textbf{ mod } 2) \not\simeq 0$, 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
$\rightarrow$ (firstn (n, nat-to-v (nat, m)) = nat-to-v (nat, n))

THEOREM: restn-nat-to-v-0-hack
if $m < n$ **then f**
else t endif
$\rightarrow$ (restn (n, nat-to-v (0, m)) = nat-to-v (0, $m - n$))

THEOREM: length-nat-to-v
length (nat-to-v (n, $length$)) = fix ($length$)

THEOREM: bvp-nat-to-v
bvp (nat-to-v (n, $length$))

THEOREM: car-nat-to-v-0-is-f
$(n \not\simeq 0) \rightarrow (\neg$ car (nat-to-v (0, n)))

THEOREM: any-of-nat-to-v-0-is-f
$(n < m) \rightarrow (\neg$ nth (n, nat-to-v (0, m)))

DEFINITION: v-nth (v-n, lst) = nth (v-to-nat (v-n), lst)

EVENT: Enable v-nth; name this event 'v-nth-off'.

DEFINITION:
update-v-nth (v-n, lst, $value$) = update-nth (v-to-nat (v-n), lst, $value$)

EVENT: Enable update-v-nth; name this event 'update-v-nth-off'.

DEFINITION:
v-nzerop (x)
$=$ **if** $x \simeq$ **nil then f**
else car $(x) \vee$ v-nzerop (cdr (x)) **endif**

EVENT: Enable v-nzerop; name this event 'v-nzerop-off'.

DEFINITION: v-zerop (x) = (¬ v-nzerop (x))

THEOREM: v-nzerop-as-or-crock
(v-nzerop (firstn $(n,\ a)$) → v-nzerop (a))
∧ (v-nzerop (restn $(n,\ a)$) → v-nzerop (a))

THEOREM: not-v-nzerop-as-and-crock
((¬ v-nzerop (firstn $(n,\ a)$)) ∧ (¬ v-nzerop (restn $(n,\ a)$)))
→ (¬ v-nzerop (a))

THEOREM: not-v-nzerop-v-xor-x-x
¬ v-nzerop (v-xor $(x,\ x)$)

THEOREM: v-xor-nzerop=not-equal
bv2p $(a,\ b)$ → (v-nzerop (v-xor $(a,\ b)$) = $(a \neq b)$)

THEOREM: v-zerop-make-list-f
v-zerop (make-list (n, **f**)) = **t**

THEOREM: not-v-nzerop-all-f
¬ v-nzerop (make-list (n, **f**))

DEFINITION:
v-negp (x)
= **if** $x \simeq$ **nil** **then** **f**
elseif cdr $(x) \simeq$ **nil** **then** car (x)
else v-negp (cdr (x)) **endif**

EVENT: Enable v-negp; name this event 'v-negp-off'.

THEOREM: boolp-v-negp
bvp (v) → boolp (v-negp (v))

THEOREM: v-negp-as-nth
(length $(bv) \neq 0$) → (v-negp (bv) = nth (length $(bv) - 1,\ bv$))

EVENT: Enable v-negp-as-nth; name this event 'v-negp-as-nth-off'.

DEFINITION:
sign-extend $(v,\ n)$
= **if** $n \simeq 0$ **then** **nil**
elseif $v \simeq$ **nil** **then** make-list (n, **f**)
elseif cdr $(v) \simeq$ **nil**
then cons (boolfix (car (v)), make-list ($n - 1$, boolfix (car (v))))
else cons (boolfix (car (v)), sign-extend (cdr (v), $n - 1$)) **endif**

EVENT: Enable sign-extend; name this event 'sign-extend-off'.

THEOREM: length-sign-extend
length (sign-extend (v, n)) $=$ fix (n)

THEOREM: bvp-sign-extend
bvp (sign-extend (v, n))

THEOREM: sign-extend-as-append
(bvp (v)
$\land$ **if** $n <$ length (v) **then f**
else t endif
$\land$ (length (v) $\neq$ 0))
$\rightarrow$ (sign-extend (v, n)
$=$ append (v, make-list ($n -$ length (v), nth (length (v) $-$ 1, v))))

EVENT: Enable sign-extend-as-append; name this event 'sign-extend-as-append-off'.

DEFINITION:
v-adder (c, a, b)
$=$ **if** $a \simeq$ **nil then** list (boolfix (c))
else cons (b-xor3 (c, car (a), car (b)),
v-adder (b-or (b-and (car (a), car (b)),
b-or (b-and (car (a), c), b-and (car (b), c))),
cdr (a),
cdr (b))) **endif**

EVENT: Enable v-adder; name this event 'v-adder-off'.

DEFINITION:
v-adder-output (c, a, b) $=$ firstn (length (a), v-adder (c, a, b))

DEFINITION:
v-adder-carry-out (c, a, b) $=$ nth (length (a), v-adder (c, a, b))

DEFINITION:
v-adder-overflowp (c, a, b)
$=$ b-and (b-equv (nth (length (a) $-$ 1, a), nth (length (b) $-$ 1, b)),
b-xor (nth (length (a) $-$ 1, a),
nth (length (a) $-$ 1, v-adder-output (c, a, b))))

DEFINITION:
v-subtracter-output (c, a, b) $=$ v-adder-output (b-not (c), v-not (a), b)

DEFINITION:
v-subtracter-carry-out $(c,\ a,\ b)$
$=$ b-not (v-adder-carry-out (b-not (c), v-not $(a),\ b)$)

DEFINITION:
v-subtracter-overflowp $(c,\ a,\ b)$ = v-adder-overflowp (b-not (c), v-not $(a),\ b)$

DEFINITION:
v-inc (x) = v-adder-output (**t**, x, nat-to-v (0, length (x)))

EVENT: Enable v-inc; name this event 'v-inc-off'.

DEFINITION:
v-dec (x) = v-subtracter-output (**t**, nat-to-v (0, length (x)), x)

EVENT: Enable v-dec; name this event 'v-dec-off'.

THEOREM: length-of-v-adder
length (v-adder $(c,\ a,\ b)$) = (1 + length (a))

THEOREM: bvp-v-adder
bvp (v-adder $(c,\ a,\ b)$)

THEOREM: length-of-v-adder-output
length (v-adder-output $(c,\ a,\ b)$) = length (a)

THEOREM: length-of-v-subtracter-output
length (v-subtracter-output $(c,\ a,\ b)$) = length (a)

THEOREM: bvp-length-v-inc-v-dec
bvp (v-inc (x))
$\wedge$ bvp (v-dec (x))
$\wedge$ (length (v-inc (x)) = length (x))
$\wedge$ (length (v-dec (x)) = length (x))

THEOREM: v-buf-works
bvp $(x) \rightarrow$ (v-buf $(x) = x$)

THEOREM: v-if-works
bv2p $(x,\ y)$
$\rightarrow$ (v-if $(c,\ x,\ y)$
$=$ **if** c **then** x
else y **endif**)

THEOREM: v-adder-works
$\text{bv2p}(x, y)$
$\rightarrow$ (v-to-nat (v-adder (c, x, y))
= (**if** c **then** 1
else 0 **endif**
+ v-to-nat (x)
+ v-to-nat (y)))

DEFINITION:
v-threefix (v)
= **if** $v \simeq$ **nil** **then** **nil**
else cons (threefix (car (v)), v-threefix (cdr (v))) **endif**

EVENT: Enable v-threefix; name this event 'v-threefix-off'.

THEOREM: open-v-threefix
(($v \simeq$ **nil**) $\rightarrow$ (v-threefix (v) = **nil**))
$\wedge$ (listp (v)
$\rightarrow$ (v-threefix (v) = cons (threefix (car (v)), v-threefix (cdr (v)))))

THEOREM: v-threefix-bvp
bvp (v) $\rightarrow$ (v-threefix $(v) = v$)

THEOREM: properp-v-threefix
properp (v-threefix (x))

THEOREM: length-v-threefix
length (v-threefix (x)) = length (x)

THEOREM: append-v-threefix
append (v-threefix (a), v-threefix (b)) = v-threefix (append (a, b))

THEOREM: v-threefix-append
v-threefix (append (a, b)) = append (v-threefix (a), v-threefix (b))

EVENT: Enable v-threefix-append; name this event 'v-threefix-append-off'.

THEOREM: v-threefix-idempotence
v-threefix (v-threefix (x)) = v-threefix (x)

THEOREM: bvp-v-threefix
properp (v) $\rightarrow$ (bvp (v-threefix (v)) = bvp (v))

THEOREM: v-threefix-make-list-x
v-threefix (make-list $(n$, X)) = make-list $(n$, X)

DEFINITION:
v-fourfix (v)
$=$ **if** $v \simeq$ **nil then nil**
else cons (fourfix (car (v)), v-fourfix (cdr (v))) **endif**

EVENT: Enable v-fourfix; name this event 'v-fourfix-off'.

THEOREM: bvp-v-fourfix
bvp $(v) \rightarrow$ (v-fourfix $(v) = v$)

THEOREM: v-fourfix-make-list
$((x = \mathbf{t}) \vee (x = \mathbf{f}) \vee (x = \mathrm{x}) \vee (x = \mathrm{z}))$
$\rightarrow$ (v-fourfix (make-list (n, x)) = make-list (n, x))

THEOREM: v-threefix-v-fourfix
v-threefix (v-fourfix (v)) = v-threefix (v)

DEFINITION:
v-iff (a, b)
$=$ **if** $a \simeq$ **nil then t**
else (car $(a) \leftrightarrow$ car (b)) $\wedge$ v-iff (cdr (a), cdr (b)) **endif**

EVENT: Enable v-iff; name this event 'v-iff-off'.

THEOREM: v-iff-x-x
v-iff (x, x)

THEOREM: v-iff-rev1
((length (a) = length (b)) $\wedge$ (length (x) = length (y)))
$\rightarrow$ (v-iff (rev1 (a, x), rev1 (b, y)) = (v-iff (a, b) $\wedge$ v-iff (x, y)))

THEOREM: v-iff-reverse
(length (a) = length (b)) $\rightarrow$ (v-iff (reverse (a), reverse (b)) = v-iff (a, b))

THEOREM: v-iff=equal
bv2p $(a, b) \rightarrow$ (v-iff $(a, b) = (a = b)$)

THEOREM: bvp-subrange
(bvp $(v) \wedge (n <$ length (v))) $\rightarrow$ bvp (subrange (v, m, n))

THEOREM: boolp-if*
(boolp $(a) \wedge$ boolp (b)) $\rightarrow$ boolp (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
$(\neg\ \text{duplicates?}\,(args))$
$\rightarrow\ \ (\text{collect-value}\,(\text{restn}\,(n,\ args),\ \text{pairlist}\,(args,\ answer))$
$\quad =\ \ \text{collect-value}\,(\text{restn}\,(n,\ args),$
$\qquad\qquad \text{pairlist}\,(\text{restn}\,(n,\ args),\ \text{restn}\,(n,\ answer))))$

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
$(\text{length}\,(a) = \mathtt{1})$
$\rightarrow\ \ (\text{collect-value}\,(a,\ alist) = \text{list}\,(\text{value}\,(\text{car}\,(a),\ alist)))$

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
$\text{collect-value}\,(\text{append}\,(\text{firstn}\,(n,\ l),\ \text{restn}\,(n,\ l)),\ alist)$
$=\ \ \text{append}\,(\text{collect-value}\,(\text{firstn}\,(n,\ l),\ alist),$
$\qquad\qquad \text{collect-value}\,(\text{restn}\,(n,\ l),\ alist))$

THEOREM: collect-value-splitting-crock
$\text{collect-value}\,(l,\ alist)$
$=\ \ \text{append}\,(\text{collect-value}\,(\text{firstn}\,(n,\ l),\ alist),$
$\qquad\qquad \text{collect-value}\,(\text{restn}\,(n,\ l),\ alist))$

THEOREM: collect-value-make-list
$\text{collect-value}\,(\text{make-list}\,(n,\ name),\ alist) = \text{make-list}\,(n,\ \text{value}\,(name,\ alist))$

DEFINITION:
$\text{cdr-cdr-sub1-sub1-induction}\,(l1,\ n,\ m,\ l2)$
$=\ \ \textbf{if}\ m < n\ \textbf{then}\ \mathbf{t}$
$\quad \textbf{elseif}\ n \simeq \mathtt{0}$
$\quad \textbf{then}\ \textbf{if}\ m \simeq \mathtt{0}\ \textbf{then}\ \mathbf{t}$
$\qquad\quad \textbf{else}\ \text{cdr-cdr-sub1-sub1-induction}\,(\text{cdr}\,(l1),\ \mathtt{0},\ m-1,\ \text{cdr}\,(l2))\ \textbf{endif}$
$\quad \textbf{else}\ \text{cdr-cdr-sub1-sub1-induction}\,(\text{cdr}\,(l1),\ n-1,\ m-1,\ \text{cdr}\,(l2))\ \textbf{endif}$

THEOREM: collect-value-subrange-args-pairlist-args

$((\neg \text{ duplicates?}(args))$
$\land \quad (\text{length}(args) = \text{length}(list))$
$\land \quad \text{properp}(list)$
$\land \quad (n < \text{length}(args)))$
$\rightarrow \quad (\text{collect-value}(\text{subrange}(args,\ m,\ n),\ \text{pairlist}(args,\ list))$
$\qquad = \quad \text{subrange}(list,\ m,\ n))$

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)) $\land$ (length (stub-guts (mem)) $=$ $width$)
elseif $n \simeq 0$
then if ramp (mem)
then properp (ram-guts (mem))
$\land$ (length (ram-guts (mem)) $=$ $width$)
elseif romp (mem)
then properp (rom-guts (mem))
$\land$ (length (rom-guts (mem)) $=$ $width$)
else f endif
else listp (mem)
$\land$ memory-properp $(n - 1,\ width,\ \text{car}(mem))$
$\land$ memory-properp $(n - 1,\ width,\ \text{cdr}(mem))$ **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)) $\land$ (length (stub-guts (mem)) $=$ $width$)
elseif $n \simeq 0$
then if ramp (mem)
then bvp (ram-guts (mem)) $\land$ (length (ram-guts (mem)) $=$ $width$)

elseif romp (mem)
then bvp (rom-guts (mem)) $\wedge$ (length (rom-guts (mem)) $=$ $width$)
else f 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 ($v\text{-}addr$, mem)
= **if** stubp (mem) **then** stub-guts (mem)
elseif $v\text{-}addr \simeq$ **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 ($v\text{-}addr$) **then** read-mem1 (cdr ($v\text{-}addr$), cdr (mem))
else read-mem1 (cdr ($v\text{-}addr$), car (mem)) **endif**

EVENT: Enable read-mem1; name this event 'read-mem1-off'.

DEFINITION:
read-mem ($v\text{-}addr$, mem) = read-mem1 (reverse ($v\text{-}addr$), mem)

EVENT: Enable read-mem; name this event 'read-mem-off'.

DEFINITION:
write-mem1 ($v\text{-}addr$, mem, $value$)
= **if** stubp (mem) **then** mem
elseif $v\text{-}addr \simeq$ **nil**
then if ramp (mem) **then** ram ($value$)
else mem **endif**
elseif $mem \simeq$ **nil then** mem
elseif car ($v\text{-}addr$)
then cons (car (mem), write-mem1 (cdr ($v\text{-}addr$), cdr (mem), $value$))
else cons (write-mem1 (cdr ($v\text{-}addr$), car (mem), $value$), cdr (mem)) **endif**

EVENT: Enable write-mem1; name this event 'write-mem1-off'.

DEFINITION:
write-mem ($v\text{-}addr$, mem, $value$) = write-mem1 (reverse ($v\text{-}addr$), 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 $v\text{-}addr \simeq$ **nil then** ramp (mem)
elseif $mem \simeq$ **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 $mem \simeq$ **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 $mem \simeq$ **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) $\land$ 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)
$\land$ properp ($value$)
$\land$ ($width$ = length ($value$)))
$\rightarrow$ memory-properp (n, $width$, constant-ram (mem, $value$))

THEOREM: memory-properp-after-write-mem1
(memory-properp (n, $width$, mem)
$\land$ properp ($value$)
$\land$ ($width$ = length ($value$))
$\land$ (n = length (v-$addr$)))
$\rightarrow$ memory-properp (n, $width$, write-mem1 (v-$addr$, mem, $value$))

THEOREM: memory-properp-after-write-mem
(memory-properp (n, $width$, mem)
$\land$ properp ($value$)
$\land$ ($width$ = length ($value$))
$\land$ (n = length (v-$addr$)))
$\rightarrow$ memory-properp (n, $width$, write-mem (v-$addr$, mem, $value$))

THEOREM: memory-okp-after-write-mem1
(memory-okp (n, $width$, mem)
$\land$ bvp ($value$)
$\land$ ($width$ = length ($value$))
$\land$ (n = length (v-$addr$)))
$\rightarrow$ memory-okp (n, $width$, write-mem1 (v-$addr$, mem, $value$))

THEOREM: memory-okp-after-write-mem
(memory-okp (n, $width$, mem)
$\land$ bvp ($value$)
$\land$ ($width$ = length ($value$))
$\land$ (n = length (v-$addr$)))
$\rightarrow$ memory-okp (n, $width$, write-mem (v-$addr$, mem, $value$))

THEOREM: v-iff-v-addr1-v-addr2-read-mem1-write-mem1
(v-iff (v-$addr1$, v-$addr2$)
$\land$ ramp-mem1 (v-$addr2$, mem)
$\land$ (length (v-$addr1$) = length (v-$addr2$)))
$\rightarrow$ (read-mem1 (v-$addr1$, write-mem1 (v-$addr2$, mem, $value$)) = $value$)

THEOREM: v-iff-v-addr1-v-addr2-read-mem1-write-mem1-not-ram
$((\neg \text{ramp-mem1}(\textit{v-addr2},\ \textit{mem})) \land (\text{length}(\textit{v-addr1}) = \text{length}(\textit{v-addr2})))$
$\to\ (\text{read-mem1}(\textit{v-addr1},\ \text{write-mem1}(\textit{v-addr2},\ \textit{mem},\ \textit{value}))$
$\quad = \ \text{read-mem1}(\textit{v-addr1},\ \textit{mem}))$

THEOREM: not-v-iff-v-addr1-v-addr2-read-mem1-write-mem1
$(\neg \text{v-iff}(\textit{v-addr1},\ \textit{v-addr2}))$
$\to\ (\text{read-mem1}(\textit{v-addr1},\ \text{write-mem1}(\textit{v-addr2},\ \textit{mem},\ \textit{value}))$
$\quad = \ \text{read-mem1}(\textit{v-addr1},\ \textit{mem}))$

THEOREM: read-mem-write-mem
$(\text{length}(\textit{v-addr1}) = \text{length}(\textit{v-addr2}))$
$\to\ (\text{read-mem}(\textit{v-addr1},\ \text{write-mem}(\textit{v-addr2},\ \textit{mem},\ \textit{value}))$
$\quad = \ \textbf{if}\ \text{v-iff}(\textit{v-addr1},\ \textit{v-addr2}) \land \text{ramp-mem}(\textit{v-addr2},\ \textit{mem})$
$\quad\quad \textbf{then}\ \textit{value}$
$\quad\quad \textbf{else}\ \text{read-mem}(\textit{v-addr1},\ \textit{mem})\ \textbf{endif})$

THEOREM: properp-read-mem1
$\text{memory-properp}(\text{length}(\textit{v-addr}),\ \textit{size},\ \textit{mem})$
$\to\ (\text{properp}(\text{read-mem1}(\textit{v-addr},\ \textit{mem}))$
$\quad \land\ (\text{length}(\text{read-mem1}(\textit{v-addr},\ \textit{mem})) = \textit{size}))$

THEOREM: properp-read-mem
$\text{memory-properp}(\text{length}(\text{reverse}(\textit{v-addr})),\ \textit{size},\ \textit{mem})$
$\to\ (\text{properp}(\text{read-mem}(\textit{v-addr},\ \textit{mem}))$
$\quad \land\ (\text{length}(\text{read-mem}(\textit{v-addr},\ \textit{mem})) = \textit{size}))$

THEOREM: properp-read-mem-32
$\text{memory-properp}(\text{length}(\textit{v-addr}),\ 32,\ \textit{mem})$
$\to\ (\text{properp}(\text{read-mem}(\textit{v-addr},\ \textit{mem}))$
$\quad \land\ (\text{length}(\text{read-mem}(\textit{v-addr},\ \textit{mem})) = 32))$

THEOREM: bvp-read-mem1
$\text{memory-okp}(\text{length}(\textit{v-addr}),\ \textit{size},\ \textit{mem})$
$\to\ (\text{bvp}(\text{read-mem1}(\textit{v-addr},\ \textit{mem}))$
$\quad \land\ (\text{length}(\text{read-mem1}(\textit{v-addr},\ \textit{mem})) = \textit{size}))$

THEOREM: bvp-read-mem
$\text{memory-okp}(\text{length}(\text{reverse}(\textit{v-addr})),\ \textit{size},\ \textit{mem})$
$\to\ (\text{bvp}(\text{read-mem}(\textit{v-addr},\ \textit{mem}))$
$\quad \land\ (\text{length}(\text{read-mem}(\textit{v-addr},\ \textit{mem})) = \textit{size}))$

THEOREM: bvp-read-mem-32
$\text{memory-okp}(\text{length}(\textit{v-addr}),\ 32,\ \textit{mem})$
$\to\ (\text{bvp}(\text{read-mem}(\textit{v-addr},\ \textit{mem})) \land (\text{length}(\text{read-mem}(\textit{v-addr},\ \textit{mem})) = 32))$

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))
 ∧  (length (dual-port-ram-value (bits, address-lines, args, state))
     =  fix (bits))
```

DEFINITION:

```
next-memory-state (state, strobe-, rw-, address, data)
=   if boolp (strobe-) ∧ 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 (cadddddddr (state))
                              else v-threefix (data) endif
                         else make-list (length (data), x) endif)
              elseif rw-
              then if caddddddr (state) ∧ boolp (caddddddr (state))
                   then if bvp (address)
                           ∧   bvp (cadddddddr (state))
                           ∧   (address = caddddddr (state))
                        then list (car (state),
                                   1,
                                   cdaddr (state),
                                   caaddr (state) - 1,
                                   caaddr (state) ≃ 0,
                                   threefix (rw-),
```

```
                    v-threefix (address),
                    if boolp (rw-)
                    then if rw-
                         then v-threefix (caddddddddr (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) ≃ 0,
                   threefix (rw-),
                   v-threefix (address),
                   if boolp (rw-)
                   then if rw-
                        then v-threefix (caddddddddr (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) ≃ 0,
              threefix (rw-),
              v-threefix (address),
              if boolp (rw-)
              then if rw-
                   then v-threefix (caddddddddr (state))
                   else v-threefix (data) endif
              else make-list (length (data), x) endif) endif
elseif (¬ caddddddr (state))
     ∧ (bvp (address)
        ∧ bvp (cadddddddr (state))
        ∧ (address = cadddddddr (state)))
     ∧ bvp (data)
     ∧ bvp (caddddddddr (state))
     ∧ (data = caddddddddr (state))
then list (car (state),
          2,
          cdaddr (state),
          caaddr (state) − 1,
          caaddr (state) ≃ 0,
```

```
                threefix (rw-),
                v-threefix (address),
                if boolp (rw-)
                then if rw-  then v-threefix (caddddddddr (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) ≃ 0,
               threefix (rw-),
               v-threefix (address),
               if boolp (rw-)
               then if rw-
                     then v-threefix (caddddddddr (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 (caddddddddr (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 (caddddddddr (state))
```

```
                    else v-threefix (data) endif
                else make-list (length (data), x) endif) endif
    elseif rw-
    then if bvp (address)
          ∧  bvp (caddddddr (state))
          ∧  (address = caddddddr (state))
        then list (car (state),
                   1,
                   caddr (state),
                   cadddr (state) − 1,
                   cadddr (state) ≃ 0,
                   threefix (rw-),
                   v-threefix (address),
                   if boolp (rw-)
                   then if rw-
                        then v-threefix (cadddddddr (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) ≃ 0,
                  threefix (rw-),
                  v-threefix (address),
                  if boolp (rw-)
                  then if rw-
                       then v-threefix (cadddddddr (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) ≃ 0,
              threefix (rw-),
              v-threefix (address),
              if boolp (rw-)
              then if rw-
                   then v-threefix (cadddddddr (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 (caddddddddr (*state*))
        else v-threefix (*data*) **endif**
      else make-list (length (*data*), x) **endif**)
  elseif (bvp (*address*)
      $\land$ bvp (cadddddddr (*state*))
      $\land$ (*address* = cadddddddr (*state*)))
    $\land$ (bvp (*data*)
      $\land$ bvp (caddddddddr (*state*))
      $\land$ (*data* = caddddddddr (*state*)))
    $\land$ (cadddr (*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 (caddddddddr (*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 (caddddddddr (*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) ≃ 0,
                   threefix (rw-),
                   v-threefix (address),
                   if boolp (rw-)
                   then if rw- then v-threefix (caddddddddr (state))
                        else v-threefix (data) endif
                   else make-list (length (data), x) endif)
        elseif (bvp (address)
                 ∧  bvp (caddddddr (state))
                 ∧  (address = caddddddr (state)))
                ∧  bvp (data)
                ∧  bvp (caddddddddr (state))
                ∧  (data = caddddddddr (state))
        then list (car (state),
                   2,
                   caddr (state),
                   cadddr (state) − 1,
                   cadddr (state) ≃ 0,
                   threefix (rw-),
                   v-threefix (address),
                   if boolp (rw-)
                   then if rw- then v-threefix (caddddddddr (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) ≃ 0,
                  threefix (rw-),
                  v-threefix (address),
                  if boolp (rw-)
                  then if rw-
                       then v-threefix (caddddddddr (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 (caddddddddr (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 (caddddddddr (state))
                   else v-threefix (data) endif
              else make-list (length (data),
                              x) endif) endif
elseif rw-
then list (car (state),
           3,
           caddr (state),
           cadddr (state) - 1,
           cadddr (state) ≃ 0,
           threefix (rw-),
           v-threefix (address),
           if boolp (rw-)
           then if rw-
                then v-threefix (caddddddddr (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) ≃ 0,
                                   threefix (rw-),
                                   v-threefix (address),
                                   if boolp (rw-)
                                   then if rw-
                                        then v-threefix (cadddddddr (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 (cadddddddr (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) ≃ 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 = caddddddr (state),
                                 and* (bvp (address), bvp (caddddddr (state)))),
                           if* (cadddr (state) ≃ 0,
                                cons (f,
                                      if* (caddddr (state),
                                           read-mem (address, car (state)),
                                           make-list (length (data), x))),
```

```
                              cons (t, make-list (length (data), x))),
                          if* (cadddr (state) ≃ 0,
                              cons (f, make-list (length (data), x)),
                              cons (t, make-list (length (data), x)))),
                      if* (cadddr (state) ≃ 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* (cadddr (state) ≃ 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* (cadddr (state) ≃ 0,
                             cons (f, make-list (length (data), x)),
                             cons (t, make-list (length (data), x))),
                         if* (cadddr (state) ≃ 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*))
 ∧ (length (*address*) = 32)
 ∧ (length (*data*) = 32))
→ (properp (cdr (memory-value (*state*, *strobe*, *rw-*, *address*, *data*)))
  ∧ (length (cdr (memory-value (*state*, *strobe*, *rw-*, *address*, *data*)))
   = 32))

THEOREM: equal-memory-value
(length (*data1*) = length (*data2*))
→ ((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-*) ∧ 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 (cadddddddr (*state*))
            else v-threefix (*data*) **endif**
          else make-list (length (*data*), X) **endif**)
      elseif *rw-*
      then if caddddr (*state*) ∧ boolp (caddddr (*state*))
        then if bvp (*address*)
            ∧ bvp (cadddddr (*state*))
            ∧ (*address* = cadddddr (*state*))
          then list (car (*state*),
              1,
              cdaddr (*state*),
              caaddr (*state*) − 1,
              caaddr (*state*) ≃ 0,
              threefix (*rw-*),

```
                        v-threefix (address),
                        if boolp (rw-)
                        then if rw-
                             then v-threefix (cadddddddr (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) ≃ 0,
                       threefix (rw-),
                       v-threefix (address),
                       if boolp (rw-)
                       then if rw-
                            then v-threefix (cadddddddr (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) ≃ 0,
                     threefix (rw-),
                     v-threefix (address),
                     if boolp (rw-)
                     then if rw-
                          then v-threefix (cadddddddr (state))
                          else v-threefix (data) endif
                     else make-list (length (data), x) endif) endif
elseif (¬ cadddddr (state))
       ∧  (bvp (address)
           ∧  bvp (caddddddr (state))
           ∧  (address = caddddddr (state)))
       ∧  bvp (data)
       ∧  bvp (cadddddddr (state))
       ∧  (data = cadddddddr (state))
then list (car (state),
           2,
           cdaddr (state),
           caaddr (state) − 1,
           caaddr (state) ≃ 0,
```

```
threefix (rw-),
v-threefix (address),
if boolp (rw-)
then if rw-  then v-threefix (caddddddddr (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) ≃ 0,
          threefix (rw-),
          v-threefix (address),
          if boolp (rw-)
          then if rw-
                  then v-threefix (caddddddddr (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 (caddddddddr (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 (caddddddddr (state))
```

```
                        else v-threefix (data) endif
                   else make-list (length (data), x) endif) endif
     elseif rw-
     then if bvp (address)
             ∧  bvp (caddddddr (state))
             ∧  (address = caddddddr (state))
          then list (car (state),
                    1,
                    caddr (state),
                    cadddr (state) − 1,
                    cadddr (state) ≃ 0,
                    threefix (rw-),
                    v-threefix (address),
                    if boolp (rw-)
                    then if rw-
                         then v-threefix (cadddddddr (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) ≃ 0,
                   threefix (rw-),
                   v-threefix (address),
                   if boolp (rw-)
                   then if rw-
                        then v-threefix (cadddddddr (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) ≃ 0,
              threefix (rw-),
              v-threefix (address),
              if boolp (rw-)
              then if rw-
                   then v-threefix (cadddddddr (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 (caddddddddr (state))
                     else v-threefix (data) endif
                else make-list (length (data), x) endif)
     elseif (bvp (address)
              ∧  bvp (cadddddddr (state))
              ∧  (address = cadddddddr (state)))
             ∧  (bvp (data)
                  ∧  bvp (caddddddddr (state))
                  ∧  (data = caddddddddr (state)))
             ∧  (cadddr (state) ≃ 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 (caddddddddr (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 (caddddddddr (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) ≃ 0,
              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 (bvp (address)
             ∧  bvp (cadddddr (state))
             ∧  (address = cadddddr (state)))
            ∧  bvp (data)
            ∧  bvp (caddddddr (state))
            ∧  (data = caddddddr (state))
    then list (car (state),
              2,
              caddr (state),
              cadddr (state) − 1,
              cadddr (state) ≃ 0,
              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)
    else list (constant-ram (car (state), make-list (32, x)),
             3,
             cdaddr (state),
             caaddr (state) − 1,
             caaddr (state) ≃ 0,
             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) 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 (caddddddr (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 (caddddddr (state))
                      else v-threefix (data) endif
                 else make-list (length (data),
                                 x) endif) endif
elseif rw-
then list (car (state),
           3,
           caddr (state),
           cadddr (state) - 1,
           cadddr (state) ≃ 0,
           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)
else list (constant-ram (car (state),
                         make-list (32, x)),
           3,
           cdaddr (state),
           caaddr (state) - 1,
```

```
                                        caaddr (state) ≃ 0,
                                        threefix (rw-),
                                        v-threefix (address),
                                        if boolp (rw-)
                                        then if rw-
                                             then v-threefix (caddddddddr (state))
                                             else v-threefix (data) endif
                                        else make-list (length (data), x) endif) endif endcase
   elseif ¬ (boolp (strobe-) ∧ 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 (caddddddddr (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))
 ∧   (length (next-memory-state (state, strobe-, rw-, address, data)) = 8)
```

DEFINITION:

```
fix-breadth-tree-stack (stack, n)
=  if stack ≃ 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 ≃ nil
   then cons (cons (t,
                    if stack ≃ nil then make-list (n, f)
                    else car (stack) endif),
              fix-breadth-tree-stack (cdr (stack), 2))
   else cons (cons (t,
                    if stack ≃ 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 ≃ 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: f-buf (a) = threefix (a)

DEFINITION:

```
f-and (a, b)
=   if (a = f) ∨ (b = f) then f
    elseif (a = t) ∧ (b = t) then t
    else x endif
```

DEFINITION: f-and3 (a, b, c) = f-and (a, f-and (b, c))

DEFINITION: f-and4 (a, b, c, d) = f-and (a, f-and (b, f-and (c, d)))

DEFINITION:

```
f-not (a)
=   if boolp (a) then ¬ a
    else x endif
```

DEFINITION: f-nand (a, b) = f-not (f-and (a, b))

DEFINITION: f-nand3 (a, b, c) = f-not (f-and (a, f-and (b, c)))

DEFINITION:
f-nand4 (a, b, c, d) = f-not (f-and (a, f-and (b, 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:
$\text{f-nand6}(a,\ b,\ c,\ d,\ e,\ g)$
$= \quad \text{f-not}(\text{f-and}(a,\ \text{f-and}(b,\ \text{f-and}(c,\ \text{f-and}(d,\ \text{f-and}(e,\ g))))))$

DEFINITION:
$\text{f-nand8}(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)$
$= \quad \text{f-not}(\text{f-and}(a,\ \text{f-and}(b,\ \text{f-and}(c,\ \text{f-and}(d,\ \text{f-and}(e,\ \text{f-and}(g,\ \text{f-and}(h,\ i))))))))$

DEFINITION:
$\text{f-or}(a,\ b)$
$= \quad \textbf{if}\ (a = \mathbf{t}) \lor (b = \mathbf{t})\ \ \textbf{then}\ \mathbf{t}$
$\quad\ \ \textbf{elseif}\ (a = \mathbf{f}) \land (b = \mathbf{f})\ \ \textbf{then}\ \mathbf{f}$
$\quad\ \ \textbf{else}\ \mathrm{X}\ \textbf{endif}$

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:
$\text{f-nor8}(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)$
$= \quad \text{f-not}(\text{f-or}(a,\ \text{f-or}(b,\ \text{f-or}(c,\ \text{f-or}(d,\ \text{f-or}(e,\ \text{f-or}(g,\ \text{f-or}(h,\ i))))))))$

DEFINITION:
$\text{f-xor}(a,\ b)$
$= \quad \textbf{if}\ a = \mathbf{t}\ \ \textbf{then}\ \text{f-not}(b)$
$\quad\ \ \textbf{elseif}\ a = \mathbf{f}\ \textbf{then}\ \text{threefix}(b)$
$\quad\ \ \textbf{else}\ \mathrm{X}\ \textbf{endif}$

DEFINITION: $\text{f-xor3}(a,\ b,\ c) = \text{f-xor}(a,\ \text{f-xor}(b,\ c))$

DEFINITION:
f-equv $(a,\ b)$
$=$ **if** $a = \mathbf{t}$ **then** threefix (b)
elseif $a = \mathbf{f}$ **then** f-not (b)
else X **endif**

DEFINITION: f-equv3 $(a,\ b,\ c)$ = f-equv $(a$, 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 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 f**
elseif $(a = \mathbf{t}) \wedge (b = \mathbf{t})$ **then t**
else X **endif**)
$\wedge$ (f-and3 $(a,\ b,\ c)$ = f-and $(a$, f-and $(b,\ c)))$
$\wedge$ (f-and4 $(a,\ b,\ c,\ d)$ = f-and $(a$, f-and $(b$, f-and $(c,\ d))))$
$\wedge$ (f-not (a)
$=$ **if** boolp (a) **then** $\neg\ a$
else X **endif**)

$$
\begin{array}{ll}
\wedge & (\text{f-nand}\,(a,\ b) = \text{f-not}\,(\text{f-and}\,(a,\ b))) \\
\wedge & (\text{f-nand3}\,(a,\ b,\ c) = \text{f-not}\,(\text{f-and}\,(a,\ \text{f-and}\,(b,\ c)))) \\
\wedge & (\text{f-nand4}\,(a,\ b,\ c,\ d) = \text{f-not}\,(\text{f-and}\,(a,\ \text{f-and}\,(b,\ \text{f-and}\,(c,\ d))))) \\
\wedge & (\text{f-or}\,(a,\ b) \\
 & \ = \ \ \textbf{if}\ (a = \mathbf{t}) \vee (b = \mathbf{t})\ \ \textbf{then}\ \mathbf{t} \\
 & \qquad \textbf{elseif}\ (a = \mathbf{f}) \wedge (b = \mathbf{f})\ \ \textbf{then}\ \mathbf{f} \\
 & \qquad \textbf{else}\ \mathrm{X}\ \textbf{endif}) \\
\wedge & (\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)))))) \\
\wedge & (\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))))))) \\
\wedge & (\text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i) \\
 & \ = \ \ \text{f-not}\,(\text{f-and}\,(a, \\
 & \qquad\qquad\qquad \text{f-and}\,(b, \\
 & \qquad\qquad\qquad\qquad \text{f-and}\,(c, \\
 & \qquad\qquad\qquad\qquad\qquad \text{f-and}\,(d,\ \text{f-and}\,(e,\ \text{f-and}\,(g,\ \text{f-and}\,(h,\ i))))))))) \\
\wedge & (\text{f-or3}\,(a,\ b,\ c) = \text{f-or}\,(a,\ \text{f-or}\,(b,\ c))) \\
\wedge & (\text{f-or4}\,(a,\ b,\ c,\ d) = \text{f-or}\,(a,\ \text{f-or}\,(b,\ \text{f-or}\,(c,\ d)))) \\
\wedge & (\text{f-nor}\,(a,\ b) = \text{f-not}\,(\text{f-or}\,(a,\ b))) \\
\wedge & (\text{f-nor3}\,(a,\ b,\ c) = \text{f-not}\,(\text{f-or}\,(a,\ \text{f-or}\,(b,\ c)))) \\
\wedge & (\text{f-nor4}\,(a,\ b,\ c,\ d) = \text{f-not}\,(\text{f-or}\,(a,\ \text{f-or}\,(b,\ \text{f-or}\,(c,\ d))))) \\
\wedge & (\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)))))) \\
\wedge & (\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))))))) \\
\wedge & (\text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i) \\
 & \ = \ \ \text{f-not}\,(\text{f-or}\,(a, \\
 & \qquad\qquad\qquad \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))))))))) \\
\wedge & (\text{f-xor}\,(a,\ b) \\
 & \ = \ \ \textbf{if}\ a = \mathbf{t}\ \ \textbf{then}\ \text{f-not}\,(b) \\
 & \qquad \textbf{elseif}\ a = \mathbf{f}\ \ \textbf{then}\ \text{threefix}\,(b) \\
 & \qquad \textbf{else}\ \mathrm{X}\ \textbf{endif}) \\
\wedge & (\text{f-xor3}\,(a,\ b,\ c) = \text{f-xor}\,(a,\ \text{f-xor}\,(b,\ c))) \\
\wedge & (\text{f-equv}\,(a,\ b) \\
 & \ = \ \ \textbf{if}\ a = \mathbf{t}\ \ \textbf{then}\ \text{threefix}\,(b) \\
 & \qquad \textbf{elseif}\ a = \mathbf{f}\ \ \textbf{then}\ \text{f-not}\,(b) \\
 & \qquad \textbf{else}\ \mathrm{X}\ \textbf{endif}) \\
\wedge & (\text{f-equv3}\,(a,\ b,\ c) = \text{f-equv}\,(a,\ \text{f-xor}\,(b,\ c))) \\
\wedge & (\text{f-if}\,(c,\ a,\ b) \\
 & \ = \ \ \textbf{if}\ c = \mathbf{t}\ \ \textbf{then}\ \text{threefix}\,(a) \\
 & \qquad \textbf{elseif}\ c = \mathbf{f}\ \ \textbf{then}\ \text{threefix}\,(b) \\
 & \qquad \textbf{else}\ \mathrm{X}\ \textbf{endif}) \\
\wedge & (\text{ft-buf}\,(c,\ a) \\
 & \ = \ \ \textbf{if}\ c = \mathbf{t}\ \ \textbf{then}\ \text{threefix}\,(a) \\
 & \qquad \textbf{elseif}\ c = \mathbf{f}\ \ \textbf{then}\ \mathrm{Z} \\
 & \qquad \textbf{else}\ \mathrm{X}\ \textbf{endif})
\end{array}
$$

$\land$ (ft-wire $(a,\ b)$
  $=$ **if** $a = b$ **then** fourfix (a)
  elseif $a = \mathrm{Z}$ **then** fourfix (b)
  elseif $b = \mathrm{Z}$ **then** fourfix (a)
  else X **endif**)
$\land$ (f-pullup (a)
  $=$ **if** $a = \mathrm{Z}$ **then** **t**
  else threefix (a) **endif**)

EVENT: Enable expand-f-functions; name this event 'expand-f-functions-off'.

THEOREM: f-buf-lemma
boolp $(x) \rightarrow$ (f-buf $(x) = x$)

THEOREM: f-and-rewrite
($\neg$ f-and (**f**, x))
$\land$ ($\neg$ f-and (x, **f**))
$\land$ (f-and (x, **t**) $=$ f-buf (x))
$\land$ (f-and (**t**, x) $=$ f-buf (x))
$\land$ (($x \land y \land$ boolp $(x) \land$ boolp (y)) $\rightarrow$ (f-and $(x,\ y) =$ **t**))

THEOREM: f-or-rewrite
(f-or (**f**, **f**) $=$ **f**)
$\land$ (($x \land$ boolp (x)) $\rightarrow$ (f-or $(x,\ y) =$ **t**))
$\land$ (($x \land$ boolp (x)) $\rightarrow$ (f-or $(y,\ x) =$ **t**))

THEOREM: f-not-rewrite
(f-not (**t**) $=$ **f**) $\land$ (f-not (**f**) $=$ **t**)

THEOREM: ft-buf-rewrite
(ft-buf (**t**, x) $=$ threefix (x))
$\land$ (ft-buf (**f**, x) $=$ Z)
$\land$ (ft-buf (c, threefix (x)) $=$ ft-buf $(c,\ x)$)
$\land$ (ft-buf (X, x) $=$ X)

THEOREM: f-if-rewrite
(f-if (**t**, a, b) $=$ f-buf (a)) $\land$ (f-if (**f**, a, b) $=$ f-buf (b))

THEOREM: ft-wire-rewrite
(ft-wire (X, x) $=$ X)
$\land$ (ft-wire (x, X) $=$ X)
$\land$ (ft-wire (x, Z) $=$ fourfix (x))
$\land$ (ft-wire (Z, x) $=$ fourfix (x))

THEOREM: f-pullup-rewrite
(f-pullup (**t**) = **t**)
∧ (f-pullup (**f**) = **f**)
∧ (f-pullup (x) = x)
∧ (f-pullup (z) = **t**)

THEOREM: threefix-help-lemma
(threefix (**f**) = **f**)
∧ (threefix (**t**) = **t**)
∧ (threefix (x) = x)
∧ (threefix (f-buf (x)) = threefix (x))
∧ (threefix (f-not (x)) = f-not (x))
∧ (threefix (f-and (x, y)) = f-and (x, y))
∧ (threefix (f-or (x, y)) = f-or (x, y))
∧ (threefix (f-xor (x, y)) = f-xor (x, y))
∧ (threefix (f-equv (x, y)) = f-equv (x, y))

EVENT: Enable threefix-help-lemma; name this event ‘threefix-help-lemma-off’.

THEOREM: f-not-f-not=f-buf
f-not (f-not (x)) = 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
(f-buf (f-buf (a)) = f-buf (a))
∧ (f-buf (f-not (a)) = f-not (a))
∧ (f-buf (f-nand (a, b)) = f-nand (a, b))
∧ (f-buf (f-nand3 (a, b, c)) = f-nand3 (a, b, c))
∧ (f-buf (f-nand4 (a, b, c, d)) = f-nand4 (a, b, c, d))
∧ (f-buf (f-nand5 (a, b, c, d, e)) = f-nand5 (a, b, c, d, e))
∧ (f-buf (f-nand6 (a, b, c, d, e, g)) = f-nand6 (a, b, c, d, e, g))
∧ (f-buf (f-nand8 (a, b, c, d, e, g, h, i)) = f-nand8 (a, b, c, d, e, g, h, i))
∧ (f-buf (f-or (a, b)) = f-or (a, b))
∧ (f-buf (f-or3 (a, b, c)) = f-or3 (a, b, c))
∧ (f-buf (f-or4 (a, b, c, d)) = f-or4 (a, b, c, d))
∧ (f-buf (f-xor (a, b)) = f-xor (a, b))
∧ (f-buf (f-xor3 (a, b, c)) = f-xor3 (a, b, c))
∧ (f-buf (f-equv (a, b)) = f-equv (a, b))
∧ (f-buf (f-equv3 (a, b, c)) = f-equv3 (a, b, c))
∧ (f-buf (f-and (a, b)) = f-and (a, b))
∧ (f-buf (f-and3 (a, b, c)) = f-and3 (a, b, c))
∧ (f-buf (f-and4 (a, b, c, d)) = f-and4 (a, b, c, d))
∧ (f-buf (f-nor (a, b)) = f-nor (a, b))

$\wedge$ (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-equv (threefix (a), b) = f-equv $(a,\ b)$)
$\wedge$ (f-equv (a, threefix (b)) = f-equv $(a,\ b)$)
$\wedge$ (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 $(a,\ b,\ c,\ d,\ e)$)

$\wedge\quad (\text{f-nand5}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e)) = \text{f-nand5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge\quad (\text{f-nand6}\,(\text{threefix}\,(a),\ b,\ c,\ d,\ e,\ g) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand6}\,(a,\ \text{threefix}\,(b),\ c,\ d,\ e,\ g) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand6}\,(a,\ b,\ \text{threefix}\,(c),\ d,\ e,\ g) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand6}\,(a,\ b,\ c,\ \text{threefix}\,(d),\ e,\ g) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand6}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e),\ g) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ \text{threefix}\,(g)) = \text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge\quad (\text{f-nand8}\,(\text{threefix}\,(a),\ b,\ c,\ d,\ e,\ g,\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ \text{threefix}\,(b),\ c,\ d,\ e,\ g,\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ \text{threefix}\,(c),\ d,\ e,\ g,\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ c,\ \text{threefix}\,(d),\ e,\ g,\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e),\ g,\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ \text{threefix}\,(g),\ h,\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ \text{threefix}\,(h),\ i) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ \text{threefix}\,(i)) = \text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge\quad (\text{f-or}\,(\text{threefix}\,(a),\ b) = \text{f-or}\,(a,\ b))$
$\wedge\quad (\text{f-or}\,(a,\ \text{threefix}\,(b)) = \text{f-or}\,(a,\ b))$
$\wedge\quad (\text{f-or3}\,(\text{threefix}\,(a),\ b,\ c) = \text{f-or3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-or3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-or3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-or3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-or3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-or4}\,(\text{threefix}\,(a),\ b,\ c,\ d) = \text{f-or4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-or4}\,(a,\ \text{threefix}\,(b),\ c,\ d) = \text{f-or4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-or4}\,(a,\ b,\ \text{threefix}\,(c),\ d) = \text{f-or4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-or4}\,(a,\ b,\ c,\ \text{threefix}\,(d)) = \text{f-or4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-xor}\,(\text{threefix}\,(a),\ b) = \text{f-xor}\,(a,\ b))$
$\wedge\quad (\text{f-xor}\,(a,\ \text{threefix}\,(b)) = \text{f-xor}\,(a,\ b))$
$\wedge\quad (\text{f-xor3}\,(\text{threefix}\,(a),\ b,\ c) = \text{f-xor3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-xor3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-xor3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-xor3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-xor3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-equv}\,(\text{threefix}\,(a),\ b) = \text{f-equv}\,(a,\ b))$
$\wedge\quad (\text{f-equv}\,(a,\ \text{threefix}\,(b)) = \text{f-equv}\,(a,\ b))$
$\wedge\quad (\text{f-equv3}\,(\text{threefix}\,(a),\ b,\ c) = \text{f-equv3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-equv3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-equv3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-equv3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-equv3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-and}\,(\text{threefix}\,(a),\ b) = \text{f-and}\,(a,\ b))$
$\wedge\quad (\text{f-and}\,(a,\ \text{threefix}\,(b)) = \text{f-and}\,(a,\ b))$
$\wedge\quad (\text{f-and3}\,(\text{threefix}\,(a),\ b,\ c) = \text{f-and3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-and3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-and3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-and3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-and3}\,(a,\ b,\ c))$
$\wedge\quad (\text{f-and4}\,(\text{threefix}\,(a),\ b,\ c,\ d) = \text{f-and4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-and4}\,(a,\ \text{threefix}\,(b),\ c,\ d) = \text{f-and4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-and4}\,(a,\ b,\ \text{threefix}\,(c),\ d) = \text{f-and4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-and4}\,(a,\ b,\ c,\ \text{threefix}\,(d)) = \text{f-and4}\,(a,\ b,\ c,\ d))$
$\wedge\quad (\text{f-nor}\,(\text{threefix}\,(a),\ b) = \text{f-nor}\,(a,\ b))$

$\wedge \quad (\text{f-nor}\,(a,\ \text{threefix}\,(b)) = \text{f-nor}\,(a,\ b))$
$\wedge \quad (\text{f-nor3}\,(\text{threefix}\,(a),\ b,\ c) = \text{f-nor3}\,(a,\ b,\ c))$
$\wedge \quad (\text{f-nor3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-nor3}\,(a,\ b,\ c))$
$\wedge \quad (\text{f-nor3}\,(a,\ \text{threefix}\,(b),\ c) = \text{f-nor3}\,(a,\ b,\ c))$
$\wedge \quad (\text{f-nor4}\,(\text{threefix}\,(a),\ b,\ c,\ d) = \text{f-nor4}\,(a,\ b,\ c,\ d))$
$\wedge \quad (\text{f-nor4}\,(a,\ \text{threefix}\,(b),\ c,\ d) = \text{f-nor4}\,(a,\ b,\ c,\ d))$
$\wedge \quad (\text{f-nor4}\,(a,\ b,\ \text{threefix}\,(c),\ d) = \text{f-nor4}\,(a,\ b,\ c,\ d))$
$\wedge \quad (\text{f-nor4}\,(a,\ b,\ c,\ \text{threefix}\,(d)) = \text{f-nor4}\,(a,\ b,\ c,\ d))$
$\wedge \quad (\text{f-nor5}\,(\text{threefix}\,(a),\ b,\ c,\ d,\ e) = \text{f-nor5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge \quad (\text{f-nor5}\,(a,\ \text{threefix}\,(b),\ c,\ d,\ e) = \text{f-nor5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge \quad (\text{f-nor5}\,(a,\ b,\ \text{threefix}\,(c),\ d,\ e) = \text{f-nor5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge \quad (\text{f-nor5}\,(a,\ b,\ c,\ \text{threefix}\,(d),\ e) = \text{f-nor5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge \quad (\text{f-nor5}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e)) = \text{f-nor5}\,(a,\ b,\ c,\ d,\ e))$
$\wedge \quad (\text{f-nor6}\,(\text{threefix}\,(a),\ b,\ c,\ d,\ e,\ g) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor6}\,(a,\ \text{threefix}\,(b),\ c,\ d,\ e,\ g) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor6}\,(a,\ b,\ \text{threefix}\,(c),\ d,\ e,\ g) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor6}\,(a,\ b,\ c,\ \text{threefix}\,(d),\ e,\ g) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor6}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e),\ g) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ \text{threefix}\,(g)) = \text{f-nor6}\,(a,\ b,\ c,\ d,\ e,\ g))$
$\wedge \quad (\text{f-nor8}\,(\text{threefix}\,(a),\ b,\ c,\ d,\ e,\ g,\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ \text{threefix}\,(b),\ c,\ d,\ e,\ g,\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ \text{threefix}\,(c),\ d,\ e,\ g,\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ c,\ \text{threefix}\,(d),\ e,\ g,\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ c,\ d,\ \text{threefix}\,(e),\ g,\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ \text{threefix}\,(g),\ h,\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ \text{threefix}\,(h),\ i) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ \text{threefix}\,(i)) = \text{f-nor8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))$
$\wedge \quad (\text{f-if}\,(\text{threefix}\,(c),\ a,\ b) = \text{f-if}\,(c,\ a,\ b))$
$\wedge \quad (\text{f-if}\,(c,\ \text{threefix}\,(a),\ b) = \text{f-if}\,(c,\ a,\ b))$
$\wedge \quad (\text{f-if}\,(c,\ a,\ \text{threefix}\,(b)) = \text{f-if}\,(c,\ a,\ b))$

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-equv, f-equv3, 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
$(\text{boolp}\,(a) \rightarrow (\text{f-buf}\,(a) = \text{b-buf}\,(a)))$
$\wedge \quad (\text{boolp}\,(a) \rightarrow (\text{f-not}\,(a) = \text{b-not}\,(a)))$
$\wedge \quad ((\text{boolp}\,(a) \wedge \text{boolp}\,(b)) \rightarrow (\text{f-nand}\,(a,\ b) = \text{b-nand}\,(a,\ b)))$

$$
\begin{array}{ll}
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-nand3}\,(a,\ b,\ c) = \text{b-nand3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d)) \\
 & \quad \rightarrow \quad (\text{f-nand4}\,(a,\ b,\ c,\ d) = \text{b-nand4}\,(a,\ b,\ c,\ d))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d) \wedge \mathrm{boolp}\,(e)) \\
 & \quad \rightarrow \quad (\text{f-nand5}\,(a,\ b,\ c,\ d,\ e) = \text{b-nand5}\,(a,\ b,\ c,\ d,\ e))) \\
\wedge & ((\mathrm{boolp}\,(a) \\
 & \quad \wedge \quad \mathrm{boolp}\,(b) \\
 & \quad \wedge \quad \mathrm{boolp}\,(c) \\
 & \quad \wedge \quad \mathrm{boolp}\,(d) \\
 & \quad \wedge \quad \mathrm{boolp}\,(e) \\
 & \quad \wedge \quad \mathrm{boolp}\,(g)) \\
 & \quad \rightarrow \quad (\text{f-nand6}\,(a,\ b,\ c,\ d,\ e,\ g) = \text{b-nand6}\,(a,\ b,\ c,\ d,\ e,\ g))) \\
\wedge & ((\mathrm{boolp}\,(a) \\
 & \quad \wedge \quad \mathrm{boolp}\,(b) \\
 & \quad \wedge \quad \mathrm{boolp}\,(c) \\
 & \quad \wedge \quad \mathrm{boolp}\,(d) \\
 & \quad \wedge \quad \mathrm{boolp}\,(e) \\
 & \quad \wedge \quad \mathrm{boolp}\,(g) \\
 & \quad \wedge \quad \mathrm{boolp}\,(h) \\
 & \quad \wedge \quad \mathrm{boolp}\,(i)) \\
 & \quad \rightarrow \quad (\text{f-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i) = \text{b-nand8}\,(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b)) \rightarrow (\text{f-or}\,(a,\ b) = \text{b-or}\,(a,\ b))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-or3}\,(a,\ b,\ c) = \text{b-or3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d)) \\
 & \quad \rightarrow \quad (\text{f-or4}\,(a,\ b,\ c,\ d) = \text{b-or4}\,(a,\ b,\ c,\ d))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b)) \rightarrow (\text{f-xor}\,(a,\ b) = \text{b-xor}\,(a,\ b))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-xor3}\,(a,\ b,\ c) = \text{b-xor3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b)) \rightarrow (\text{f-equv}\,(a,\ b) = \text{b-equv}\,(a,\ b))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-equv3}\,(a,\ b,\ c) = \text{b-equv3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b)) \rightarrow (\text{f-and}\,(a,\ b) = \text{b-and}\,(a,\ b))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-and3}\,(a,\ b,\ c) = \text{b-and3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d)) \\
 & \quad \rightarrow \quad (\text{f-and4}\,(a,\ b,\ c,\ d) = \text{b-and4}\,(a,\ b,\ c,\ d))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b)) \rightarrow (\text{f-nor}\,(a,\ b) = \text{b-nor}\,(a,\ b))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c)) \\
 & \quad \rightarrow \quad (\text{f-nor3}\,(a,\ b,\ c) = \text{b-nor3}\,(a,\ b,\ c))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d)) \\
 & \quad \rightarrow \quad (\text{f-nor4}\,(a,\ b,\ c,\ d) = \text{b-nor4}\,(a,\ b,\ c,\ d))) \\
\wedge & ((\mathrm{boolp}\,(a) \wedge \mathrm{boolp}\,(b) \wedge \mathrm{boolp}\,(c) \wedge \mathrm{boolp}\,(d) \wedge \mathrm{boolp}\,(e))
\end{array}
$$

$\quad\rightarrow\quad$ (f-nor5 $(a,\ b,\ c,\ d,\ e)$ = b-nor5 $(a,\ b,\ c,\ d,\ e)$))

$\wedge$ ((boolp (a)
$\quad\wedge$ boolp (b)
$\quad\wedge$ boolp (c)
$\quad\wedge$ boolp (d)
$\quad\wedge$ boolp (e)
$\quad\wedge$ boolp (g))
$\quad\rightarrow$ (f-nor6 $(a,\ b,\ c,\ d,\ e,\ g)$ = b-nor6 $(a,\ b,\ c,\ d,\ e,\ g)$))

$\wedge$ ((boolp (a)
$\quad\wedge$ boolp (b)
$\quad\wedge$ boolp (c)
$\quad\wedge$ boolp (d)
$\quad\wedge$ boolp (e)
$\quad\wedge$ boolp (g)
$\quad\wedge$ boolp (h)
$\quad\wedge$ boolp (i))
$\quad\rightarrow$ (f-nor8 $(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)$ = b-nor8 $(a,\ b,\ c,\ d,\ e,\ g,\ h,\ i)$))

$\wedge$ ((boolp (c) $\wedge$ boolp (a) $\wedge$ boolp (b))
$\quad\rightarrow$ (f-if $(c,\ a,\ b)$ = b-if $(c,\ a,\ b)$)))

DEFINITION:
fv-or $(a,\ b)$
= **if** $a \simeq$ **nil then nil**
else cons (f-or (car (a), car (b)), fv-or (cdr (a), cdr (b))) **endif**

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

THEOREM: properp-length-fv-or
properp (fv-or $(a,\ b)$) $\wedge$ (length (fv-or $(a,\ b)$) = length (a))

THEOREM: fv-or=v-or
(bvp (a) $\wedge$ bvp (b) $\wedge$ (length (a) = length (b)))
$\rightarrow$ (fv-or $(a,\ b)$ = v-or $(a,\ b)$)

DEFINITION:
fv-xor $(a,\ b)$
= **if** $a \simeq$ **nil then nil**
else cons (f-xor (car (a), car (b)), fv-xor (cdr (a), cdr (b))) **endif**

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

THEOREM: properp-length-fv-xor
properp (fv-xor $(a,\ b)$) $\wedge$ (length (fv-xor $(a,\ b)$) = length (a))

THEOREM: fv-xor=v-xor
(bvp (a) $\wedge$ bvp (b) $\wedge$ (length (a) = length (b)))
$\rightarrow$ (fv-xor $(a,\ b)$ = v-xor $(a,\ b)$)

DEFINITION:
fv-if $(c,\ a,\ b)$
= **if** $a \simeq$ **nil** **then** **nil**
else cons (f-if $(c$, car (a), car $(b))$, fv-if $(c$, cdr (a), cdr (b))) **endif**

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

THEOREM: length-fv-if
length (fv-if $(c,\ a,\ b)$) = length (a)

THEOREM: properp-fv-if
properp (fv-if $(c,\ a,\ b)$)

THEOREM: v-threefix-fv-if
v-threefix (fv-if $(c,\ a,\ b)$) = fv-if $(c,\ a,\ b)$

THEOREM: fv-if-when-boolp-c
(length (a) = length (b))
$\rightarrow$ ((fv-if (**t**, $a,\ b$) = v-threefix (a))
$\wedge$ (fv-if (**f**, $a,\ b$) = v-threefix (b)))

THEOREM: fv-if-when-bvp
(boolp (c) $\wedge$ bvp (a) $\wedge$ bvp (b) $\wedge$ (length (a) = length (b)))
$\rightarrow$ (fv-if $(c,\ a,\ b)$ = if* $(c,\ a,\ b)$)

THEOREM: fv-if-rewrite
(length (a) = length (b))
$\rightarrow$ (fv-if $(c,\ a,\ b)$
= **if** boolp (c)
then **if** c **then** v-threefix (a)
else v-threefix (b) **endif**
else make-list (length (a), X) **endif**)

EVENT: Enable fv-if-rewrite; name this event ‘fv-if-rewrite-off’.

THEOREM: fv-if-v-threefix
(fv-if $(c$, v-threefix (a), $b)$ = fv-if $(c,\ a,\ b)$)
$\wedge$ (fv-if $(c,\ a$, v-threefix (b)) = fv-if $(c,\ a,\ b)$)

DEFINITION:
v-wire $(a,\ b)$
= **if** $a \simeq$ **nil** **then** **nil**
else cons (ft-wire (car (a), car (b)), v-wire (cdr (a), cdr (b))) **endif**

EVENT: Enable v-wire; name this event 'v-wire-off'.

THEOREM: v-wire-x-x=x
v-wire $(x,\ x)$ = v-fourfix (x)

THEOREM: v-wire-make-list-z
(length $(v) = n$)
$\rightarrow$ ((v-wire (make-list $(n,$ z$),\ v)$ = v-fourfix (v))
$\wedge$ (v-wire $(v,$ make-list $(n,$ z$))$ = v-fourfix (v)))

THEOREM: properp-length-v-wire
properp (v-wire $(a,\ b)$) $\wedge$ (length (v-wire $(a,\ b)$) = length (a))

THEOREM: v-wire-make-list-x
$(n =$ length $(x)) \rightarrow$ (v-wire $(x,$ make-list $(n,$ x$))$ = make-list $(n,$ x$)$)

DEFINITION:
v-pullup (v)
= **if** $v \simeq$ **nil** **then** **nil**
else cons (f-pullup (car (v)), v-pullup (cdr (v))) **endif**

EVENT: Enable v-pullup; name this event 'v-pullup-off'.

THEOREM: properp-length-v-pullup
properp (v-pullup (v)) $\wedge$ (length (v-pullup (v)) = length (v))

THEOREM: v-pullup-bvp
bvp $(v) \rightarrow$ (v-pullup $(v) = v$)

THEOREM: v-pullup-make-list-z
v-pullup (make-list $(n,$ z$)$) = make-list $(n,$ **t**$)$

DEFINITION:
fv-shift-right $(a,\ si)$
= **if** $a \simeq$ **nil** **then** **nil**
else append (v-threefix (cdr (a)), list (threefix (si))) **endif**

EVENT: Enable fv-shift-right; name this event 'fv-shift-right-off'.

THEOREM: fv-shift-right=v-shift-right
(boolp (si) $\wedge$ bvp (a)) $\rightarrow$ (fv-shift-right $(a,\ si)$ = v-shift-right $(a,\ si)$)

THEOREM: properp-length-fv-shift-right
properp (fv-shift-right $(a,\ si)$)
$\wedge$ (length (fv-shift-right $(a,\ si)$) = length (a))

THEOREM: v-threefix-fv-shift-right
v-threefix (fv-shift-right (a, si)) = fv-shift-right (a, si)

DEFINITION:
vft-buf (c, a)
= **if** $a \simeq$ **nil** **then nil**
else cons (ft-buf (c, car (a)), vft-buf (c, cdr (a))) **endif**

EVENT: Enable vft-buf; name this event 'vft-buf-off'.

THEOREM: properp-length-vft-buf
properp (vft-buf (c, a)) $\wedge$ (length (vft-buf (c, a)) = length (a))

THEOREM: vft-buf-lemmas
(vft-buf (**t**, a) = v-threefix (a))
$\wedge$ (vft-buf (**f**, a) = make-list (length (a), z))

THEOREM: vft-buf-rewrite
vft-buf (c, a)
= **if** c = **t** **then** v-threefix (a)
elseif c = **f** **then** make-list (length (a), z)
else make-list (length (a), x) **endif**

EVENT: Enable vft-buf-rewrite; name this event 'vft-buf-rewrite-off'.

```
(DEFN PRIMP-DATABASE NIL
 (LIST
 '(AO2 (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 . AO2)
       (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))
```

```
'(AO4 (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 . AO4)
      (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))
'(AO6 (DELAYS ((260 745) (82 202)))
      (DRIVES 10)
      (INPUT-TYPES BOOLP BOOLP BOOLP)
      (INPUTS A B C)
      (LOADINGS 1 1 1)
      (LSI-NAME . AO6)
      (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))
'(AO7 (DELAYS ((261 486) (83 293)))
      (DRIVES 10)
      (INPUT-TYPES BOOLP BOOLP BOOLP)
      (INPUTS A B C)
      (LOADINGS 1 1 1)
      (LSI-NAME . AO7)
      (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)
```

```
        (INPUTS A)
        (LOADINGS 1)
        (LSI-NAME . DEL10)
        (OUT-DEPENDS (A))
        (OUTPUT-TYPES BOOLP)
        (OUTPUTS Z)
        (RESULTS CONS (F-BUF A) 'NIL)
        (GATES . 19)
        (PRIMITIVES . 1)
        (TRANSISTORS . 76))
(LIST 'PROCMON
      '(DELAYS ((70 25000) (60 25000)))
      '(DRIVES 10)
      '(INPUT-TYPES PARAMETRIC PARAMETRIC PARAMETRIC PARAMETRIC)
      '(INPUTS A E S N)
      '(LOADINGS 2 2 2 1)
      '(LSI-NAME . PROCMON)
      '(OUT-DEPENDS (A E S N))
      '(OUTPUT-TYPES PARAMETRIC)
      '(OUTPUTS Z)
      (LIST 'RESULTS
            'CONS
            (LIST 'F-IF
                  'S
                  (LIST 'F-IF
                        'E
                        (LIST 'F-IF
                              'A
                              (LIST 'QUOTE F)
                              (LIST 'QUOTE F))
                        'A)
                  'N)
            ''NIL)
      '(GATES . 100)
      '(PRIMITIVES . 1)
      '(TRANSISTORS . 400))
'(DP-RAM-16X32
  (DELAYS ((35 7500) (21 7500))
          ((35 7500) (21 7500))
          ((35 7500) (21 7500))
          ((35 7500) (21 7500))
          ((35 7500) (21 7500))
          ((35 7500) (21 7500))
          ((35 7500) (21 7500))
```

```
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500))
        ((35 7500) (21 7500)))
(DRIVES 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)
(INPUT-TYPES BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP BOOLP LEVEL 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 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)
(LOADINGS 2 2 2 2 2 2 2 2 4 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 . CMRB100A)
(NEW-STATES DUAL-PORT-RAM-STATE '32
 '4
 (CONS READ-A0
  (CONS READ-A1
   (CONS READ-A2
    (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)
(OUTPUT-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)
(OUT-DEPENDS (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3)
             (READ-A0 READ-A1 READ-A2 READ-A3))
(OUTPUTS O0 O1 O2 O3 O4 O5 O6 O7 O8 O9 O10 O11 O12 O13 O14 O15 O16 O17 O18
         O19 O20 O21 O22 O23 O24 O25 O26 O27 O28 O29 O30 O31)
(RESULTS DUAL-PORT-RAM-VALUE '32
 '4
 (CONS READ-A0
  (CONS READ-A1
   (CONS READ-A2
```

```
(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)
```

```
         (TRANSISTORS . 40))
'(ID (DELAYS A)
     (DRIVES A)
     (INPUT-TYPES FREE)
     (INPUTS A)
     (LOADINGS 0)
     (LSI-NAME . ID)
     (OUT-DEPENDS (A))
     (OUTPUT-TYPES (A))
     (OUTPUTS Z)
     (RESULTS CONS A 'NIL)
     (GATES . 0)
     (PRIMITIVES . 0)
     (TRANSISTORS . 0))
'(MEM-32X32
  (DELAYS ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
          ((1410 20000) (820 20000))
```

```
        ((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 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 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 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
                                                                                                                                                          'NIL)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
 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
```

```
     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
     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
     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
     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
     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
     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
     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
     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
     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 A29 A30 A31))
```

```
(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 TRI-STATE TRI-STATE TRI-STATE
              TRI-STATE TRI-STATE TRI-STATE)
(OUTPUTS DTACK- O0 O1 O2 O3 O4 O5 O6 O7 O8 O9 O10 O11 O12 O13 O14 O15 O16
         O17 O18 O19 O20 O21 O22 O23 O24 O25 O26 O27 O28 O29 O30 O31)
(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
                                                                                                                                'NIL)))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))
   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 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
                      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*)) $\land$ (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*)) $\land$ (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*) $\lor$ (*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*) = cadddr (*module*)

DEFINITION: module-statenames (*module*) = caddddr (*module*)

DEFINITION: module-annotation (*module*) = cadddddr (*module*)

DEFINITION: occ-name (*occurrence*) = car (*occurrence*)

DEFINITION: occ-outputs (*occurrence*) = cadr (*occurrence*)

DEFINITION: occ-function (*occurrence*) = caddr (*occurrence*)

DEFINITION: occ-inputs (*occurrence*) = cadddr (*occurrence*)

DEFINITION: occ-annotation (*occurrence*) = caddddr (*occurrence*)

DEFINITION: primp (*fn*) = lookup-module (*fn*, PRIMP-DATABASE)

EVENT: Enable primp; name this event 'primp-off'.

THEOREM: not-litatom==>not-primp
($\neg$ litatom (*fn*)) $\rightarrow$ ($\neg$ 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 f endif
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

```
 ((¬ primp (fn)) ∧ listp (lookup-module (fn, netlist)))
 →  (dual-eval (0, fn, args, state, netlist)
     =  collect-value (module-outputs (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))))
```

THEOREM: expand-top-level-dual-eval-2-calls

```
((¬ primp (fn)) ∧ listp (lookup-module (fn, netlist)))
→  (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 ≃ nil)
 →  (dual-eval (1, body, bindings, state-bindings, netlist) = bindings))
∧  (listp (body)
    →  (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
$((body \simeq \mathbf{nil}) \rightarrow (\text{dual-eval}\,(3,\ body,\ bindings,\ state\text{-}bindings,\ netlist) = \mathbf{nil}))$
$\land$ (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*)) $\land$ listp (lookup-module (*fn*, *netlist*)))
$\rightarrow$ properp (dual-eval (0, *fn*, *args*, *state*, *netlist*))

THEOREM: length-dual-eval-0
(($\neg$ primp (*fn*)) $\land$ 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*))
$\land$ listp (lookup-module (*fn*, *netlist*))
$\land$ properp (module-statenames (lookup-module (*fn*, *netlist*))))
$\rightarrow$ properp (dual-eval (2, *fn*, *args*, *state*, *netlist*))

THEOREM: length-dual-eval-2
(($\neg$ primp (*fn*))
$\land$ listp (lookup-module (*fn*, *netlist*))
$\land$ 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 ≃ 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 ≃ 0)
  →  (dual-eval-body-bindings (n, body, bindings, state-bindings, netlist)
        =  bindings))
 ∧  ((n ≄ 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 ≃ 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 ≃ 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 ≃ 0) → (simulate-dual-eval-2(module, inputs, state, netlist, n) = state))
 ∧   ((n ≄ 0)
      →   (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** $e \in x$ **then** x
else cons $(e,\ x)$ **endif**

EVENT: Enable insert; name this event 'insert-off'.

DEFINITION: set-equal $(x,\ y) = (\text{subset}\,(x,\ y) \land \text{subset}\,(y,\ x))$

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

DEFINITION:
set-diff $(x,\ y)$
$=$ **if** listp (x)
then if car $(x) \in y$ **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 $(x) \in y$ **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 $(x,\ \mathit{duplicates})$
$=$ **if** listp (x)
then list-duplicates (cdr (x),
if car $(x) \in$ cdr (x)
then insert (car (x), *duplicates*)
else *duplicates* **endif**)
else *duplicates* **endif**

EVENT: Enable list-duplicates; name this event 'list-duplicates-off'.

DEFINITION:
remove-duplicates(x)
$=$ **if** listp(x)
then if car$(x) \in$ cdr(x) **then** remove-duplicates(cdr(x))
else cons(car(x), remove-duplicates(cdr(x))) **endif**
else x **endif**

EVENT: Enable remove-duplicates; name this event 'remove-duplicates-off'.

DEFINITION:
append-list(x)
$=$ **if** listp(x) **then** append(car(x), append-list(cdr(x)))
else nil endif

EVENT: Enable append-list; name this event 'append-list-off'.

DEFINITION: flatten-list(x) $=$ remove-duplicates(append-list(x))

EVENT: Enable flatten-list; name this event 'flatten-list-off'.

DEFINITION:
is-head(x, y)
$=$ **if** listp$(x) \land$ listp(y)
then (car(x) $=$ car(y)) $\land$ is-head(cdr(x), cdr(y))
else $x \simeq$ **nil endif**

EVENT: Enable is-head; name this event 'is-head-off'.

DEFINITION:
listify(x)
$=$ **if** listp(x) **then** cons(list(car(x)), listify(cdr(x)))
else nil endif

EVENT: Enable listify; name this event 'listify-off'.

DEFINITION:
alistp(x)
$=$ **if** listp(x) **then** listp(car(x)) $\land$ alistp(cdr(x))
else $x =$ **nil endif**

EVENT: Enable alistp; name this event 'alistp-off'.

DEFINITION: boundp (*key*, *alist*) = listp (lookup-module (*key*, *alist*))

DEFINITION: alist-entry (*key*, *alist*) = lookup-module (*key*, *alist*)

EVENT: Enable alist-entry; name this event 'alist-entry-off'.

DEFINITION: unbind (*key*, *alist*) = delete-module (*key*, *alist*)

DEFINITION:
set-value (*key*, *value*, *alist*)
= **if** alist-entry (*key*, *alist*) $\simeq$ **nil** **then** cons (cons (*key*, *value*), *alist*)
elseif *value* = cdr (alist-entry (*key*, *alist*)) **then** *alist*
else cons (cons (*key*, *value*), unbind (*key*, *alist*)) **endif**

EVENT: Enable set-value; name this event 'set-value-off'.

DEFINITION:
union-values (*keys*, *alist*)
= **if** listp (*keys*)
then value (car (*keys*), *alist*) $\cup$ union-values (cdr (*keys*), *alist*)
else nil endif

EVENT: Enable union-values; name this event 'union-values-off'.

DEFINITION:
list-union-values (*list-of-args*, *alist*)
= **if** listp (*list-of-args*)
then cons (union-values (car (*list-of-args*), *alist*),
list-union-values (cdr (*list-of-args*), *alist*))
else nil endif

EVENT: Enable list-union-values; name this event 'list-union-values-off'.

DEFINITION:
unbound-keys (*keys*, *alist*)
= **if** listp (*keys*)
then if boundp (car (*keys*), *alist*) **then** unbound-keys (cdr (*keys*), *alist*)
else insert (car (*keys*), unbound-keys (cdr (*keys*), *alist*)) **endif**
else nil endif

EVENT: Enable unbound-keys; name this event 'unbound-keys-off'.

DEFINITION:
unbind-list ($keys$, $alist$)
= **if** listp ($keys$) **then** unbind (car ($keys$), unbind-list (cdr ($keys$), $alist$))
else $alist$ **endif**

THEOREM: lookup-module-in-delete-module
(($k1 \neq k2$) $\wedge$ lookup-module ($k1$, m))
$\rightarrow$ lookup-module ($k1$, delete-module ($k2$, m))

THEOREM: lookup-module-in-unbind-list
(($k \notin u$) $\wedge$ lookup-module (k, m)) $\rightarrow$ lookup-module (k, unbind-list (u, m))

EVENT: Enable lookup-module-in-delete-module; name this event 'lookup-module-in-delete-module-off'.

THEOREM: lessp-unbind-list-count
(($k \notin u$) $\wedge$ boundp (k, m))
$\rightarrow$ (count (unbind-list (cons (k, u), m)) $<$ count (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 lookup-module-in-unbind-list; name this event 'lookup-module-in-unbind-list-off'.

DEFINITION:
m-states-list (m-$states$)
= **if** listp (m-$states$) $\vee$ (m-$states$ = **nil**) **then** m-$states$
else list (m-$states$) **endif**

EVENT: Enable m-states-list; name this event 'm-states-list-off'.

DEFINITION:
list-lessp (x, y)
= **if** listp (x)
then if listp (y)
then if car (x) $<$ car (y) **then t**
elseif car (x) = car (y) **then** list-lessp (cdr (x), 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 $(x) \land$ litatom (y) **then** list-lessp (unpack (x), unpack (y))
elseif litatom $(x) \land$ indexp (y) **then t**
elseif indexp $(x) \land$ litatom (y) **then f**
elseif indexp $(x) \land$ 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 $(seed,\ list) = seed$)
$\lor$ (min-token-help $(seed,\ list) \in list$)

THEOREM: min-token-lemma
listp $(lst) \rightarrow$ (min-token $(lst) \in lst$)

THEOREM: delete*-count
count (delete* (e, x)) < count (cons (e, x))

THEOREM: delete*-lemma
($e \in lst$) $\rightarrow$ (count (delete* (e, lst)) < count (lst))

DEFINITION:
list-sort ($list$)
= **if** listp ($list$)
then cons (min-token ($list$), list-sort (delete* (min-token ($list$), $list$)))
else nil endif

EVENT: Enable list-sort; name this event 'list-sort-off'.

DEFINITION:
list-list-sort ($list$)
= **if** listp ($list$)
then cons (list-sort (car ($list$)), list-list-sort (cdr ($list$)))
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: report-error (err) = err

EVENT: Enable report-error; name this event 'report-error-off'.

DEFINITION:
pred-error ($label$, $body$) = report-error (net-error ($label$, $body$))

EVENT: Enable pred-error; name this event 'pred-error-off'.

DEFINITION:
error-entry (lst, $label$)
= **if** listp (lst)
then if net-errorp (car (lst))
then if error-label (car (lst)) = $label$ **then** car (lst)
else error-entry (cdr (lst), $label$) **endif**
else error-entry (cdr (lst), $label$) **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** $p \land (\neg$ 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 = **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) $\simeq$ **nil then t**
else pred-error (*label*, 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, **nil**) $\simeq$ **nil then t**
else pred-error (*label*, list-duplicates (x, **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) $\simeq$ **nil then t**
else pred-error (*label*, 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*) $\simeq$ **nil then t**
else pred-error (*label*, 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*) $\lor$ (indexp (*name*) $\land$ 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*))
$\land$ (car (*name-list*) $\notin$ cdr (*name-list*))
$\land$ 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*))
$\lor$ ((5 = length (*occ*)) $\land$ alistp (occ-annotation (*occ*))))
$\land$ properp (*occ*)
$\land$ name-simple-okp (occ-name (*occ*))
$\land$ name-list-simple-okp (occ-outputs (*occ*))
$\land$ properp (occ-inputs (*occ*))
$\land$ **if** primp (occ-function (*occ*))
then (length (primp2 (occ-function (*occ*), `'inputs`))
= length (occ-inputs (*occ*)))
$\land$ (length (primp2 (occ-function (*occ*), `'outputs`))
= length (occ-outputs (*occ*)))
else listp (lookup-module (occ-function (*occ*), *netlist*))
$\land$ (length (module-inputs (lookup-module (occ-function (*occ*),
netlist)))
= length (occ-inputs (*occ*)))
$\land$ (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:

$$
\begin{array}{l}
\text{occ-body-syntax-simple-okp}\,(\mathit{body}, \mathit{signals}, \mathit{non\text{-}depends}, \mathit{occ\text{-}names}, \mathit{outputs}, \mathit{states}, \mathit{netlist}) \\
= \ \ \textbf{if } \mathit{body} \simeq \textbf{nil} \\
\quad \textbf{then } (\mathit{body} = \textbf{nil}) \\
\qquad \land \ \text{subset}\,(\mathit{outputs}, \mathit{signals}) \\
\qquad \land \ \text{subset}\,(\mathit{non\text{-}depends}, \mathit{signals}) \\
\qquad \land \ \text{subset}\,(\mathit{states}, \mathit{occ\text{-}names}) \\
\qquad \land \ (\neg \text{ duplicates?}\,(\mathit{signals})) \\
\qquad \land \ (\neg \text{ duplicates?}\,(\mathit{occ\text{-}names})) \\
\quad \textbf{elseif } \text{occ-syntax-simple-okp}\,(\text{car}\,(\mathit{body}), \mathit{netlist}) \\
\quad \textbf{then } \text{occ-body-syntax-simple-okp}\,(\text{cdr}\,(\mathit{body}), \\
\qquad\qquad \text{append}\,(\text{occ-outputs}\,(\text{car}\,(\mathit{body})), \mathit{signals}), \\
\qquad\qquad \text{append}\,(\text{set-diff}\,(\text{occ-inputs}\,(\text{car}\,(\mathit{body})), \mathit{signals}), \mathit{non\text{-}depends}), \\
\qquad\qquad \text{cons}\,(\text{occ-name}\,(\text{car}\,(\mathit{body})), \mathit{occ\text{-}names}), \\
\qquad\qquad \mathit{outputs}, \\
\qquad\qquad \mathit{states}, \\
\qquad\qquad \mathit{netlist}) \\
\quad \textbf{else f endif}
\end{array}
$$

EVENT: Enable occ-body-syntax-simple-okp; name this event 'occ-body-syntax-simple-okp-off'.

DEFINITION:

$$
\begin{array}{l}
\text{module-syntax-simple-okp}\,(\mathit{netlist}) \\
= \ \ (((\text{length}\,(\text{car}\,(\mathit{netlist})) = 5) \\
\qquad \lor \ ((\text{length}\,(\text{car}\,(\mathit{netlist})) = 6) \\
\qquad\qquad \land \ \text{alistp}\,(\text{module-annotation}\,(\text{car}\,(\mathit{netlist}))))) \\
\quad \land \ \text{properp}\,(\text{car}\,(\mathit{netlist})) \\
\quad \land \ \text{name-simple-okp}\,(\text{module-name}\,(\text{car}\,(\mathit{netlist}))) \\
\quad \land \ (\neg \text{ lookup-module}\,(\text{module-name}\,(\text{car}\,(\mathit{netlist})), \text{cdr}\,(\mathit{netlist}))) \\
\quad \land \ (\neg \text{ primp}\,(\text{module-name}\,(\text{car}\,(\mathit{netlist})))) \\
\quad \land \ \text{name-list-simple-okp}\,(\text{module-inputs}\,(\text{car}\,(\mathit{netlist})))
\end{array}
$$

```
      ∧  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 ≃ 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 ≃ 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 (x0)),
                                                                                                  listify (occ-inputs (car (x0))),
                                                                                                  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 ≃ 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)
       ∧  (length (args) = length (primp2 (module-name, 'inputs))))
      ∨  (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 ≃ 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 (x0)),
                                                                                       collect-types (occ-inputs (car (x0)),
                                                                                                      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-inputs (car (x0)),
                                                                                                         x1,
                                                                                                         f),
                                                                                          netlist,
                                                                                          type-table))),
                                                         update-known-types (pairlist (occ-inputs (car (x0)),
                                                                                       car (io-types-simple (0,
                                                                                                             occ-function (car (x0)),
```

```
                                                                                    collect-types (occ-inputs (car (x0)),
                                                                                                  x1,
                                                                                                  f),
                                                                                    netlist,
                                                                                    type-table))),
                                                                    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 ≃ 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-acceptablep (flg, x0, netlist, acceptable-types, checked-modules)
=   case on flg:
    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-acceptablep (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-acceptablep (0,
                                    occ-function (car (x0)),
                                    netlist,
                                    acceptable-types,
                                    checked-modules)
          then types-acceptablep (1,
                                  cdr (x0),
                                  netlist,
                                  acceptable-types,
                                  checked-modules)
          else f endif
    otherwise f endcase
```

EVENT: Enable types-acceptablep; name this event 'types-acceptablep-off'.

DEFINITION:

```
netlist-types-acceptable-list (netlist)
=   if netlist ≃ nil then nil
    elseif netlist-types-acceptable-list (cdr (netlist))
    then if types-acceptablep (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 ≃ nil) ∨ (formals ≃ nil) then actuals = formals
    else ((car (actuals) = car (formals))
            ∨  (car (formals) = 'free))
           ∧  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 ¬ 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)))
              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 (x0,
                                                                                                       netlist)),
                                                                    nil,
                                                                    delete-module (x0,
                                                                                   netlist),
                                                                    type-table)) endif
                else f endif
            else f endif
     case = 1
      then if x0 ≃ 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* ≃ **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*)
        ∧ state-simple-okp (ram-guts (*state*), ram-guts (*type*))
    elseif romp (*type*)
    then romp (*state*)
        ∧ state-simple-okp (rom-guts (*state*), rom-guts (*type*))
    elseif stubp (*type*)
    then stubp (*state*)
        ∧ state-simple-okp (stub-guts (*state*), stub-guts (*type*))
    elseif listp (*type*)
    then listp (*state*)

$\land$ state-simple-okp (car (*state*), car (*type*))
$\land$ 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*)) $\lor$ 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*)))
$\lor$ 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*) $\in$ **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 ≃ nil then nil
    elseif cdar (drs) ∈ N
    then if cdar (drs) < sum-numbers (value (caar (drs), lds)) then f
         elseif net-drives (cdr (drs), lds)
         then cons (cons (caar (drs),
                          cdar (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 ≃ nil then ld-lst
    elseif (litatom (car (ld-lst)) ∨ indexp (car (ld-lst)))
           ∧  (car (ld-lst) ∉ 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 ≃ nil then nil
    elseif lookup-module (car (names), loadings)
    then cons (if external-loading (value (car (names), loadings),
                                     external-names) ≃ nil then 0
               elseif cdr (external-loading (value (car (names), loadings),
                                                external-names)) ≃ 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 ≃ nil then nil
    elseif (¬ (litatom (value (car (names), drives))
                 ∨  indexp (value (car (names), drives))))
            ∨  (value (car (names), drives) ∈ 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)))
     ∧  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-occurrences (lookup-module (x0,
                                                                                                                            netlist)),
                                                                                          nil,
                                                                                          nil,
```

```
                                                                              delete-module (x0,
                                                                                             netlist),
                                                                              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-occurrences (lookup-module (x0,
                                                                                                                           netlist)),
                                                                                        nil,
                                                                                        nil,
                                                                                        delete-module (x0,
                                                                                                       netlist),
                                                                                        lds-table)),
                                                         append (module-inputs (lookup-module (x0,
                                                                                               netlist)),
                                                                 module-outputs (lookup-module (x0,
                                                                                                netlist)))),
                                           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 (car (x0)),
                                                                     occ-inputs (car (x0)),
                                                                     occ-outputs (car (x0)),
                                                                     netlist,
                                                                     lds-table)),
                                                        x1),
                                   append (pairlist (occ-outputs (car (x0)),
                                                     cdr (loadings-and-drives-simple (0,
                                                                     occ-function (car (x0)),
                                                                     occ-inputs (car (x0)),
                                                                     occ-outputs (car (x0)),
                                                                     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 ≃ 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 ≃ 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) ∨ (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 (¬ 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 chksum 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: name-length (*name*) = length (unpack (*name*))

EVENT: Enable name-length; name this event 'name-length-off'.

DEFINITION:
hierarchical-name-max (*n1*, *n2*)
= **if** (*n1* ≃ **nil**) ∧ (*n2* ≃ **nil**) **then** 0
  elseif *n1* ≃ **nil** **then** *n2*
  elseif *n2* ≃ **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 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 ≃ 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-table,
                                                                                                hierarchical-size),
                                                                     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 (¬ 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* $\in$ *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*))
$\land$ (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 ≃ 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)
                                   ∈  inputs)
                                 ∨  (parent-synonym (car (names), alist)
                                      ∈  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)),
                                               unbind (car (names),
                                                       unbind (parent-synonym (car (names),
                                                                               alist),
                                                               alist)))) endif,
                              inputs,
                              outputs)
    else alist endif
```

EVENT: Enable externalize-parents; name this event 'externalize-parents-off'.

DEFINITION:

subtype (x, y)

= **if** y = `'free` **then t**
elseif x = `'boolp`
then $y \in$ `'(boolp tri-state ttl-tri-state ttl)`
elseif x = `'ttl-tri-state`
then $y \in$ `'(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 ($t1$ = `'free`) $\vee$ ($t2$ = `'free`) **then t**
elseif $t1$ = `'tri-state` **then** $t2$ = `'ttl-tri-state`
elseif $t2$ = `'tri-state` **then** $t1$ = `'ttl-tri-state`
else f endif

EVENT: Enable types-compatiblep; name this event 'types-compatiblep-off'.

DEFINITION:
tri-state-typep (*type*) = (*type* $\in$ `'(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*) $\vee$ ($\neg$ 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*, 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*, **nil**)

EVENT: Enable type-value; name this event 'type-value-off'.

DEFINITION:
input-type (*name*, *alist*) = type-value (unmark-io-out (*name*), *alist*, **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*) $\land$ subtype (*out-type*, *in-type*)
then t
else io-type-error (*name*, *in-type*, *out-type*) **endif**
elseif *name* $\in$ *t-wire-ins*
then if tri-state-typep (*in-type*)
$\land$ 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** ($\neg$ *ok*) $\lor$ 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,
```

cdr (*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*))
$\land$ (car (output-type (*name*, *out-map*, *in-map*)) $\notin$ *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 ($outputs$, $out\text{-}map$, $inputs$, $in\text{-}map$)
= **if** listp ($outputs$)
then cons (collect-out-type (car ($outputs$), $out\text{-}map$, $inputs$, $in\text{-}map$),
collect-out-types (cdr ($outputs$), $out\text{-}map$, $inputs$, $in\text{-}map$))
else nil endif

EVENT: Enable collect-out-types; name this event 'collect-out-types-off'.

DEFINITION:
in-types-error ($in\text{-}map$) = err-and ($in\text{-}map$, `'input-type-conflicts`)

EVENT: Enable in-types-error; name this event 'in-types-error-off'.

DEFINITION:
out-types-error ($out\text{-}map$) = err-and ($out\text{-}map$, `'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) $\in$ **N**) $\land$ (cadr (dr) = `'ma`) $\land$ (cddr (dr) = **nil**)
case = 3
then (car (dr) $\in$ **N**)
$\land$ ((length (cadr (dr)) = 2)
$\land$ (caadr (dr) = `'point`)
$\land$ (cadadr (dr) $\in$ **N**))
$\land$ (caddr (dr) = `'ma`)
$\land$ (cdddr (dr) = **nil**)
otherwise f endcase

EVENT: Enable map; name this event 'map-off'.

DEFINITION:
pfp (ld)
= **if** length (ld) = 2
then (car (ld) $\in$ **N**) $\land$ (cadr (ld) = `'pf`) $\land$ (cddr (ld) = **nil**)
else f endif

EVENT: Enable pfp; name this event 'pfp-off'.

DEFINITION:
ma-to-pf (x) = cons ((car (x) * 10) + cadadr (x), '(pf))

EVENT: Enable ma-to-pf; name this event 'ma-to-pf-off'.

DEFINITION:
pf-to-ma (x)
= **if** (car (x) **mod** 10) $\simeq$ 0 **then** cons (car (x) $\div$ 10, '(ma))
else cons (car (x) $\div$ 10,
cons (list ('point, car (x) **mod** 10), '(ma))) **endif**

EVENT: Enable pf-to-ma; name this event 'pf-to-ma-off'.

DEFINITION: std-load-to-pf (x) = cons (x * 10, '(pf))

EVENT: Enable std-load-to-pf; name this event 'std-load-to-pf-off'.

DEFINITION:
pf-to-std-load (x)
= **if** (car (x) **mod** 10) $<$ 5 **then** car (x) $\div$ 10
else (car (x) $\div$ 10) + 1 **endif**

EVENT: Enable pf-to-std-load; name this event 'pf-to-std-load-off'.

DEFINITION: ma-to-std-drive (x) = car (x)

EVENT: Enable ma-to-std-drive; name this event 'ma-to-std-drive-off'.

DEFINITION: std-drive-to-ma (x) = cons (x, '(ma))

EVENT: Enable std-drive-to-ma; name this event 'std-drive-to-ma-off'.

DEFINITION:
zero-loadingp (a)
= **if** pfp (a) **then** car (a) $\simeq$ 0
else a $\simeq$ 0 **endif**

EVENT: Enable zero-loadingp; name this event 'zero-loadingp-off'.

DEFINITION: pf-plus (a, b) = cons (car (a) + car (b), '(pf))

EVENT: Enable pf-plus; name this event 'pf-plus-off'.

DEFINITION:
pf-difference (a, b) = cons (car (a) − car (b), `'(pf)`)

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

DEFINITION: pf-lessp (a, b) = (car (a) < car (b))

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

DEFINITION:
ma-lessp (a, b)
= **if** car (a) < car (b) **then t**
elseif car (a) = car (b) **then** cadadr (a) < cadadr (b)
else f endif

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

DEFINITION:
loading-plus (x, y)
= **if** $x \in \mathbf{N}$
then if $y \in \mathbf{N}$ **then** $x + y$
elseif pfp (y) **then** pf-plus (std-load-to-pf (x), y)
else x **endif**
elseif pfp (x)
then if pfp (y) **then** pf-plus (x, y)
elseif $y \in \mathbf{N}$ **then** pf-plus (x, std-load-to-pf (y))
else x **endif**
elseif $(y \in \mathbf{N}) \vee$ pfp (y) **then** y
else 0 **endif**

EVENT: Enable loading-plus; name this event 'loading-plus-off'.

DEFINITION: INITIAL-OCC-LOADINGS = **nil**

DEFINITION: INITIAL-OCC-DRIVES = **nil**

DEFINITION:
add-loading (*name*, *loading*, *load-map*)
= **if** zero-loadingp (*loading*) **then** *load-map*
elseif boundp (*name*, *load-map*)
then cons (cons (*name*, loading-plus (*loading*, value (*name*, *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*) $\land$ (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) ∧ (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 (cdar (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) $\wedge$ (car (a) = `'min`)
then collect-min-args (cdr (a), **t**, *drives*, *used-names*, *inputs*, *outputs*)
elseif ($a \in \mathbf{N}$) $\vee$ 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*) $\in \mathbf{N}$) $\vee$ 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* $\in$ `'(boolp clk level parametric tri-state)`
then if map (*drive*) **then** ma-to-std-drive (*drive*)
else *drive* **endif**
elseif *type* $\in$ `'(ttl ttl-tri-state)`
then if *drive* $\in \mathbf{N}$ **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** ($x \in \mathbf{N}$) $\wedge$ ($y \in \mathbf{N}$) **then** $x < y$
elseif map (x) $\wedge$ 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* ≃ **nil** **then** UNKNOWN
elseif cdr (*lst*) ≃ **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*)) ≃ **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)) ≃ **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)
$\land$ (length (x) = 2)
$\land$ (car (x) $\in \mathbf{N}$)
$\land$ (cadr (x) = `'ps-pf`))

EVENT: Enable ps-pfp; name this event 'ps-pfp-off'.

DEFINITION:
delay-lh (d)
= **if** listp (d) $\land$ (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) $\land$ (car (d) $\neq$ `'or`) **then** cadr (d)
else f endif

EVENT: Enable delay-hl; name this event 'delay-hl-off'.

Definition:
delay-slope-lh (d)
$=$ **if** listp $(d) \wedge (\text{car}(d) \neq$ `'or`$)$ **then** caar (d)
else f endif

Event: Enable delay-slope-lh; name this event ‘delay-slope-lh-off’.

Definition:
delay-slope-hl (d)
$=$ **if** listp $(d) \wedge (\text{car}(d) \neq$ `'or`$)$ **then** caadr (d)
else f endif

Event: Enable delay-slope-hl; name this event ‘delay-slope-hl-off’.

Definition:
delay-intercept-lh (d)
$=$ **if** listp $(d) \wedge (\text{car}(d) \neq$ `'or`$)$ **then** cadar (d)
else f endif

Event: Enable delay-intercept-lh; name this event ‘delay-intercept-lh-off’.

Definition:
delay-intercept-hl (d)
$=$ **if** listp $(d) \wedge (\text{car}(d) \neq$ `'or`$)$ **then** cadadr (d)
else f endif

Event: Enable delay-intercept-hl; name this event ‘delay-intercept-hl-off’.

Definition:
delay-dependencies (d)
$=$ **if** listp $(d) \wedge (\text{car}(d) \neq$ `'or`$)$ **then** cddr (d)
else nil endif

Event: Enable delay-dependencies; name this event ‘delay-dependencies-off’.

Definition: 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:
make-delay $(s\text{-}lh,\ i\text{-}lh,\ s\text{-}hl,\ i\text{-}hl,\ deps)$
$=$ make-delay-0 $(\text{list}(s\text{-}lh,\ i\text{-}lh),\ \text{list}(s\text{-}hl,\ i\text{-}hl),\ deps)$

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-rangep (x)
= **if** rangep (x) **then** range-dependencies (x) $\simeq$ **nil**
else $x \in \mathbf{N}$ **endif**

EVENT: Enable determined-rangep; name this event 'determined-rangep-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* **mod** `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* $*$ 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
listp $(x) \rightarrow$ (count (x) = (1 + (count (car (x)) + count (cdr (x)))))

THEOREM: local-delay-count-help
listp $(x) \rightarrow$ ((count (x) = 0) = **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 ≃ nil then delay
    elseif car (delay) = 'or
    then cons ('or, local-delay (cdr (delay), f, t, in-map, out-map, delay-map))
    elseif rangep (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 ≃ 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) ≃ 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) ∨ (r1 ∈ N)
    then if rangep (r2) ∨ (r2 ∈ 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)
                         ∪   range-dependencies (r2))
        else make-range (range-min (r1),
                         range-max (r1),
                         insert (r2, range-dependencies (r1))) endif
    elseif rangep (r2) ∨ (r2 ∈ 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, *approxp*)
= **if** *approxp* **then** rangep (d) $\vee$ ($d \in \mathbf{N}$) $\vee$ signal-namep (d)
else determined-rangep (d) **endif**

EVENT: Enable mergeable-input-delay-p; name this event 'mergeable-input-delay-p-off'.

DEFINITION:
insert-input-delay (d, *dlist*, *or-args?*, *approxp*)
= **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?,
approxp)
elseif mergeable-input-delay-p (d, *approxp* $\wedge$ ($\neg$ *or-args?*))
then if mergeable-input-delay-p (car (*dlist*), *approxp* $\wedge$ ($\neg$ *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*), *approxp* $\wedge$ ($\neg$ *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*) ≃ **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* ≃ **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* ≃ **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*, *approxp*)
= **if** listp (*in-delays*) ∧ (cdr (*in-delays*) ≃ **nil**)

then if if *approxp*
then rangep (car (*in-delays*)) $\vee$ (car (*in-delays*) $\in$ **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* $\in$ **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),
                                          approxp)
    else UNKNOWN endif
elseif car (dep) = 'or
then make-delay-or (input-delay0 (cdr (dep),
                                      delays,
                                      inputs,
                                      outputs,
                                      t,
                                      t,
                                      used-names,
                                      approxp))
elseif rangep (dep)
then if addable-input-delay (input-delay0 (range-dependencies (dep),
                                                delays,
                                                inputs,
                                                outputs,
                                                f,
                                                t,
                                                used-names,
                                                approxp),
                                  approxp)
    then range-plus (dep,
                     addable-input-delay (input-delay0 (range-dependencies (dep),
                                                           delays,
                                                           inputs,
                                                           outputs,
                                                           f,
                                                           t,
                                                           used-names,
                                                           approxp),
                                             approxp),
                     range-dependencies (addable-input-delay (input-delay0 (range-dependencies (dep),
                                                                               delays,
                                                                               inputs,
                                                                               outputs,
                                                                               f,
                                                                               t,
                                                                               used-names,
                                                                               approxp),
                                                                 approxp)))
    else make-range (range-min (dep),
```

range-max (*dep*),
input-delay0 (range-dependencies (*dep*),
delays,
inputs,
outputs,
f,
t,
used-names,
approxp)) **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*, *approxp*)
= **if** listp (input-delay0 (*dep*,
delays,
inputs,
outputs,
or-args?,
list?,
used-names,
approxp))
∧ rangep (car (input-delay0 (*dep*,
delays,
inputs,
outputs,
or-args?,
list?,
used-names,
approxp)))
then if (range-min (car (input-delay0 (*dep*,
delays,
inputs,
outputs,
or-args?,
list?,
used-names,
approxp)))
= 0)
∧ (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, approxp)
=  if addable-input-delay (input-delay-list, approxp)
   then make-delay (type-delay-slope (delay-slope-lh (delay), type),
                    range-plus (delay-intercept-lh (delay),
                                addable-input-delay (input-delay-list,
                                                     approxp),
                                nil),
                    type-delay-slope (delay-slope-hl (delay), type),
                    range-plus (delay-intercept-hl (delay),
                                addable-input-delay (input-delay-list,
                                                     approxp),
                                nil),
                    range-dependencies (addable-input-delay (input-delay-list,
                                                             approxp)))
   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, approxp)
=  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*, *approxp*)
= **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,
approxp))
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*),
approxp),
approxp) **endif**

EVENT: Enable output-delay; name this event 'output-delay-off'.

DEFINITION:
collect-delays-0 (*names*, *types*, *delays*, *inputs*, *outputs*, *approxp*)
= **if** listp (*names*)
then cons (output-delay (car (*names*),
car (*types*),
delays,
inputs,
outputs,
approxp),
collect-delays-0 (cdr (*names*),
cdr (*types*),
delays,

inputs,
outputs,
approxp))
else nil endif

EVENT: Enable collect-delays-0; name this event 'collect-delays-0-off'.

DEFINITION:
collect-delays (*outputs*, *output-types*, *delays*, *inputs*, *approxp*)
= collect-delays-0 (*outputs*,
output-types,
externalize-parents (*outputs*, *delays*, *inputs*, *outputs*),
inputs,
outputs,
approxp)

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,

```
                                                 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) ∨ (¬ boundp (prop-name, bindings))
   then result
   elseif prop-name ∈ '(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 ∈ '(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) ≃ 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))
  $\land$ (car (x) $\notin$ cdr (x))
  $\land$ 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* $\in$ **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* $\in$ **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 $args \in \mathbf{N}$ **then** $args$
else 1 **endif**

Event: Enable io-car; name this event 'io-car-off'.

Definition:
io-cdr $(args)$
= **if** $args \in \mathbf{N}$ **then** $1 + args$
elseif $args \simeq$ **nil** **then** 2
elseif cdr $(args) \in \mathbf{N}$ **then nil**
else cdr $(args)$ **endif**

Event: Enable io-cdr; name this event 'io-cdr-off'.

Definition:
out-prop-signals $(out\text{-}depends, inputs)$
= **if** unknownp $(out\text{-}depends)$ **then** $inputs$
else $out\text{-}depends$ **endif**

Event: Enable out-prop-signals; name this event 'out-prop-signals-off'.

Definition:
id-out-prop-signals $(output\text{-}type, out\text{-}depends, inputs)$
= **if** listp $(output\text{-}type)$ **then** $output\text{-}type$
else out-prop-signals $(out\text{-}depends, inputs)$ **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:

```
pdelays-errors (delays, outputs, types, out-depends, inputs)
=   if listp (delays)
    then cons (t-or-err (listp (outputs) ∨ (outputs ∈ N),
                         'too-many-delays,
                         cons (length (delays), '(extras))),
               cons (pdelay-ok (car (delays),
                                io-car (outputs),
                                ucar (types),
                                ucar (out-depends),
                                inputs),
                     pdelays-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:

```
pdelays-ok (delays, alist)
=   err-and (pdelays-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) ∨ 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:

```
ploading-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 ploading-ok; name this event 'ploading-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 (ploading-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 ≃ nil)
                      ∨  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))
                      ∨  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) ∨ (outputs ∈ 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 (caddr (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* $\in$ **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*))
$\lor$ (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**)
$\cup$ unknown-args (cdr (x), *signals*, *states*, **t**)
else nil endif
elseif $x \simeq$ **nil**
then if ($x \in$ *signals*) $\lor$ ($x \in$ *states*) **then nil**
elseif unknownp (*signals*) $\lor$ 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 ≃ 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) ∨ (outputs ∈ 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) ∨ (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 presults-ok; name this event 'presults-ok-off'.

DEFINITION:
pnew-states-ok (*new-states*, *alist*)

```
=   err-and (if listp (value ('states, alist))
                ∨  (value ('states, alist) = nil)
             then presults-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 presults-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 pnew-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 ≃ 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*) ∨ (*x0* ∈ *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* ≃ **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 (flg, x0, type, netlist)
=   case on flg:
    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 ('flg, flg), list ('x0, x0))) endcase
```

EVENT: Enable primitive-count; name this event 'primitive-count-off'.

DEFINITION:

```
out-depends (flg, x0, x1, x2, netlist)
=   case on flg:
    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))),
                                                  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 ('flg, flg),
                                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 (flg, x0, x1, x2, netlist)
=  case on flg:
   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 ('flg, flg),
                               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 ≃ nil) ∨ (formals ≃ nil) then actuals = formals
    elseif unknownp (car (actuals)) ∨ 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 (flg, x0, netlist)
=   case on flg:
    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 ('flg, *flg*), 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)
          ∧  state-okp-0 (ram-guts (state),
                          cadr (type),
                          0,
                          f)
    case = rom
     then romp (state)
          ∧  state-okp-0 (rom-guts (state),
                          cadr (type),
                          0,
                          f)
    case = stub
     then stubp (state)
          ∧  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)
                   ∧  state-okp-0 (cdr (state),
                                   cdr (type),
                                   0,
                                   f)
              else f endif endcase
elseif type = 'addressed-state
then if address-bits ≃ 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)
         ∧  state-okp-0 (cdr (state),
                         type,
                         address-bits − 1,
                         word-type)
    else f endif
elseif type = 'numberp then state ∈ N
elseif type = 'number-listp
then if state ≃ nil then state = nil
    elseif car (state) ∈ 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'.

```
(DEFN LOADINGS-AND-DRIVES
 (FLG X0 X1 X2 NETLIST)
 (CASE FLG
  (0
   (COND
    ((PRIMP X0)
     (LIST (PRIMP-LOOKUP X0 'LOADINGS)
           (CONS 'DRIVES
                 (FIX-DEPENDENT-DRS (PRIMP2 X0 'DRIVES)
                                    (APPEND (PAIRLIST (PRIMP2 X0 'INPUTS) X1)
                                            (PAIRLIST (PRIMP2 X0 'OUTPUTS)
                                                      X2))))))
    ((LOOKUP-MODULE X0 NETLIST)
     (IF
      (NET-ERRORP (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
                                      '(LOADINGS DRIVES INPUTS OUTPUTS)
                                      T))
      (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
                          '(LOADINGS DRIVES INPUTS OUTPUTS)
                          T)
      (LIST
       (ALIST-ENTRY 'LOADINGS
                    (VALUE X0
                           (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
                                               '(LOADINGS DRIVES INPUTS OUTPUTS)
                                               T)))
       (CONS 'DRIVES
        (FIX-DEPENDENT-DRS
         (VALUE 'DRIVES
                (VALUE X0
                       (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
                                           '(LOADINGS DRIVES INPUTS OUTPUTS)
                                           T)))
         (APPEND
          (PAIRLIST
            (VALUE 'INPUTS
                   (VALUE X0
                          (NETLIST-PROPERTIES (SUBNET 0 X0 NETLIST)
                                              '(LOADINGS DRIVES INPUTS OUTPUTS)
                                              T)))
            X1)
          (PAIRLIST
```

```
            (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: AO2$NETLIST = **nil**

THEOREM: ao2$value
ao2& (*netlist*)
→ (dual-eval (0, 'ao2, list (a, b, c, d), *state*, *netlist*)
= list (f-nor (f-and (a, b), f-and (c, d))))

EVENT: Enable ao2$value; name this event 'ao2$value-off'.

DEFINITION: ao4& (*netlist*) = **t**

EVENT: Enable ao4&; name this event 'ao4&-off'.

DEFINITION: AO4$NETLIST = **nil**

THEOREM: ao4$value
ao4& (*netlist*)
→ (dual-eval (0, 'ao4, list (a, b, c, d), *state*, *netlist*)
= list (f-nand (f-or (a, b), f-or (c, d))))

EVENT: Enable ao4$value; name this event 'ao4$value-off'.

DEFINITION: ao6& (*netlist*) = **t**

EVENT: Enable ao6&; name this event 'ao6&-off'.

DEFINITION: AO6$NETLIST = **nil**

THEOREM: ao6$value
ao6& (*netlist*)
→ (dual-eval (0, 'ao6, list (a, b, c), *state*, *netlist*)
= list (f-nor (f-and (a, b), c)))

EVENT: Enable ao6$value; name this event 'ao6$value-off'.

DEFINITION: ao7& (*netlist*) = **t**

EVENT: Enable ao7&; name this event 'ao7&-off'.

DEFINITION: AO7$NETLIST = **nil**

THEOREM: ao7$value
ao7& (*netlist*)
→ (dual-eval (0, 'ao7, list (a, b, c), *state*, *netlist*)
= list (f-nand (f-or (a, b), c)))

EVENT: Enable ao7$value; name this event 'ao7$value-off'.

DEFINITION: b-and& (*netlist*) = **t**

EVENT: Enable b-and&; name this event 'b-and&-off'.

DEFINITION: B-AND$NETLIST = **nil**

THEOREM: b-and$value
b-and& (*netlist*)
$\rightarrow$ (dual-eval (`0`, `'b-and`, list (a, b), *state*, *netlist*) = list (f-and (a, b)))

EVENT: Enable b-and$value; name this event 'b-and$value-off'.

DEFINITION: b-and3& (*netlist*) = **t**

EVENT: Enable b-and3&; name this event 'b-and3&-off'.

DEFINITION: B-AND3$NETLIST = **nil**

THEOREM: b-and3$value
b-and3& (*netlist*)
$\rightarrow$ (dual-eval (`0`, `'b-and3`, list (a, b, c), *state*, *netlist*)
= list (f-and3 (a, b, c)))

EVENT: Enable b-and3$value; name this event 'b-and3$value-off'.

DEFINITION: b-and4& (*netlist*) = **t**

EVENT: Enable b-and4&; name this event 'b-and4&-off'.

DEFINITION: B-AND4$NETLIST = **nil**

THEOREM: b-and4$value
b-and4& (*netlist*)
$\rightarrow$ (dual-eval (`0`, `'b-and4`, list (a, b, c, d), *state*, *netlist*)
= list (f-and4 (a, b, c, d)))

EVENT: Enable b-and4$value; name this event 'b-and4$value-off'.

DEFINITION: b-equv& (*netlist*) = **t**

EVENT: Enable b-equv&; name this event 'b-equv&-off'.

DEFINITION: B-EQUV$NETLIST = **nil**

THEOREM: b-equv$value
b-equv& (*netlist*)
→ (dual-eval (`0`, `'b-equv`, list (*a*, *b*), *state*, *netlist*)
= list (f-equv (*a*, *b*)))

EVENT: Enable b-equv$value; name this event 'b-equv$value-off'.

DEFINITION: b-equv3& (*netlist*) = **t**

EVENT: Enable b-equv3&; name this event 'b-equv3&-off'.

DEFINITION: B-EQUV3$NETLIST = **nil**

THEOREM: b-equv3$value
b-equv3& (*netlist*)
→ (dual-eval (`0`, `'b-equv3`, list (*a*, *b*, *c*), *state*, *netlist*)
= list (f-equv3 (*a*, *b*, *c*)))

EVENT: Enable b-equv3$value; name this event 'b-equv3$value-off'.

DEFINITION: b-if& (*netlist*) = **t**

EVENT: Enable b-if&; name this event 'b-if&-off'.

DEFINITION: B-IF$NETLIST = **nil**

THEOREM: b-if$value
b-if& (*netlist*)
→ (dual-eval (`0`, `'b-if`, list (*s*, *a*, *b*), *state*, *netlist*) = list (f-if (*s*, *a*, *b*)))

EVENT: Enable b-if$value; name this event 'b-if$value-off'.

DEFINITION: b-nand& (*netlist*) = **t**

EVENT: Enable b-nand&; name this event 'b-nand&-off'.

DEFINITION: B-NAND$NETLIST = **nil**

THEOREM: b-nand$value
b-nand& (*netlist*)
→ (dual-eval (`0`, `'b-nand`, list (*a*, *b*), *state*, *netlist*)
= list (f-nand (*a*, *b*)))

EVENT: Enable b-nand$value; name this event 'b-nand$value-off'.

DEFINITION: b-nand3& (*netlist*) = **t**

EVENT: Enable b-nand3&; name this event 'b-nand3&-off'.

DEFINITION: B-NAND3$NETLIST = **nil**

THEOREM: b-nand3$value
b-nand3& (*netlist*)
→ (dual-eval (`0`, `'b-nand3`, list (*a*, *b*, *c*), *state*, *netlist*)
= list (f-nand3 (*a*, *b*, *c*)))

EVENT: Enable b-nand3$value; name this event 'b-nand3$value-off'.

DEFINITION: b-nand4& (*netlist*) = **t**

EVENT: Enable b-nand4&; name this event 'b-nand4&-off'.

DEFINITION: B-NAND4$NETLIST = **nil**

THEOREM: b-nand4$value
b-nand4& (*netlist*)
→ (dual-eval (`0`, `'b-nand4`, list (*a*, *b*, *c*, *d*), *state*, *netlist*)
= list (f-nand4 (*a*, *b*, *c*, *d*)))

EVENT: Enable b-nand4$value; name this event 'b-nand4$value-off'.

DEFINITION: b-nand5& (*netlist*) = **t**

EVENT: Enable b-nand5&; name this event 'b-nand5&-off'.

DEFINITION: B-NAND5$NETLIST = **nil**

THEOREM: b-nand5$value
b-nand5& (*netlist*)
→ (dual-eval (`0`, `'b-nand5`, list (*a*, *b*, *c*, *d*, *e*), *state*, *netlist*)
= list (f-nand5 (*a*, *b*, *c*, *d*, *e*)))

EVENT: Enable b-nand5$value; name this event 'b-nand5$value-off'.

DEFINITION: b-nand6& (*netlist*) = **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: b-nor$value
b-nor& (*netlist*)
→ (dual-eval (`0`, `'b-nor`, list (a, b), *state*, *netlist*) = list (f-nor (a, b)))

EVENT: Enable b-nor$value; name this event 'b-nor$value-off'.

DEFINITION: b-nor3& (*netlist*) = **t**

EVENT: Enable b-nor3&; name this event 'b-nor3&-off'.

DEFINITION: B-NOR3$NETLIST = **nil**

THEOREM: b-nor3$value
b-nor3& (*netlist*)
→ (dual-eval (`0`, `'b-nor3`, list (a, b, c), *state*, *netlist*)
= list (f-nor3 (a, b, c)))

EVENT: Enable b-nor3$value; name this event 'b-nor3$value-off'.

DEFINITION: b-nor4& (*netlist*) = **t**

EVENT: Enable b-nor4&; name this event 'b-nor4&-off'.

DEFINITION: B-NOR4$NETLIST = **nil**

THEOREM: b-nor4$value
b-nor4& (*netlist*)
→ (dual-eval (`0`, `'b-nor4`, list (a, b, c, d), *state*, *netlist*)
= list (f-nor4 (a, b, c, d)))

EVENT: Enable b-nor4$value; name this event 'b-nor4$value-off'.

DEFINITION: b-nor5& (*netlist*) = **t**

EVENT: Enable b-nor5&; name this event 'b-nor5&-off'.

DEFINITION: B-NOR5$NETLIST = **nil**

THEOREM: b-nor5$value
b-nor5& (*netlist*)
→ (dual-eval (`0`, `'b-nor5`, list (a, b, c, d, e), *state*, *netlist*)
= list (f-nor5 (a, b, c, d, e)))

EVENT: Enable b-nor5$value; name this event 'b-nor5$value-off'.

DEFINITION: b-nor6& (*netlist*) = **t**

EVENT: Enable b-nor6&; name this event 'b-nor6&-off'.

DEFINITION: B-NOR6$NETLIST = **nil**

THEOREM: b-nor6$value
b-nor6& (*netlist*)
→ (dual-eval (`0`, `'b-nor6`, list (a, b, c, d, e, g), *state*, *netlist*)
= list (f-nor6 (a, b, c, d, e, g)))

EVENT: Enable b-nor6$value; name this event 'b-nor6$value-off'.

DEFINITION: b-nor8& (*netlist*) = **t**

EVENT: Enable b-nor8&; name this event 'b-nor8&-off'.

DEFINITION: B-NOR8$NETLIST = **nil**

THEOREM: b-nor8$value
b-nor8& (*netlist*)
→ (dual-eval (`0`, `'b-nor8`, list (a, b, c, d, e, g, h, i), *state*, *netlist*)
= list (f-nor8 (a, b, c, d, e, g, h, i)))

EVENT: Enable b-nor8$value; name this event 'b-nor8$value-off'.

DEFINITION: b-not& (*netlist*) = **t**

EVENT: Enable b-not&; name this event 'b-not&-off'.

DEFINITION: B-NOT$NETLIST = **nil**

THEOREM: b-not$value
b-not& (*netlist*)
→ (dual-eval (`0`, `'b-not`, list (a), *state*, *netlist*) = list (f-not (a)))

EVENT: Enable b-not$value; name this event 'b-not$value-off'.

DEFINITION: b-not-b4ip& (*netlist*) = **t**

EVENT: Enable b-not-b4ip&; name this event 'b-not-b4ip&-off'.

DEFINITION: B-NOT-B4IP$NETLIST = **nil**

THEOREM: b-not-b4ip$value
b-not-b4ip& (*netlist*)
→ (dual-eval (`0`, `'b-not-b4ip`, list (a), *state*, *netlist*)
= list (f-not (a)))

EVENT: Enable b-not-b4ip$value; name this event 'b-not-b4ip$value-off'.

DEFINITION: b-not-ivap& (*netlist*) = **t**

EVENT: Enable b-not-ivap&; name this event 'b-not-ivap&-off'.

DEFINITION: B-NOT-IVAP$NETLIST = **nil**

THEOREM: b-not-ivap$value
b-not-ivap& (*netlist*)
→ (dual-eval (`0`, `'b-not-ivap`, list (a), *state*, *netlist*)
= list (f-not (a)))

EVENT: Enable b-not-ivap$value; name this event 'b-not-ivap$value-off'.

DEFINITION: b-or& (*netlist*) = **t**

EVENT: Enable b-or&; name this event 'b-or&-off'.

DEFINITION: B-OR$NETLIST = **nil**

THEOREM: b-or$value
b-or& (*netlist*)
→ (dual-eval (`0`, `'b-or`, list (a, b), *state*, *netlist*) = list (f-or (a, b)))

EVENT: Enable b-or$value; name this event 'b-or$value-off'.

DEFINITION: b-or3& (*netlist*) = **t**

EVENT: Enable b-or3&; name this event 'b-or3&-off'.

DEFINITION: B-OR3$NETLIST = **nil**

THEOREM: b-or3$value
b-or3& (*netlist*)
→ (dual-eval (`0`, `'b-or3`, list (a, b, c), *state*, *netlist*)
= list (f-or3 (a, b, c)))

EVENT: Enable b-or3$value; name this event ‘b-or3$value-off’.

DEFINITION: b-or4& (*netlist*) = **t**

EVENT: Enable b-or4&; name this event ‘b-or4&-off’.

DEFINITION: B-OR4$NETLIST = **nil**

THEOREM: b-or4$value
b-or4& (*netlist*)
→ (dual-eval (`0`, `'b-or4`, list (*a*, *b*, *c*, *d*), *state*, *netlist*)
= list (f-or4 (*a*, *b*, *c*, *d*)))

EVENT: Enable b-or4$value; name this event ‘b-or4$value-off’.

DEFINITION: b-xor& (*netlist*) = **t**

EVENT: Enable b-xor&; name this event ‘b-xor&-off’.

DEFINITION: B-XOR$NETLIST = **nil**

THEOREM: b-xor$value
b-xor& (*netlist*)
→ (dual-eval (`0`, `'b-xor`, list (*a*, *b*), *state*, *netlist*) = list (f-xor (*a*, *b*)))

EVENT: Enable b-xor$value; name this event ‘b-xor$value-off’.

DEFINITION: b-xor3& (*netlist*) = **t**

EVENT: Enable b-xor3&; name this event ‘b-xor3&-off’.

DEFINITION: B-XOR3$NETLIST = **nil**

THEOREM: b-xor3$value
b-xor3& (*netlist*)
→ (dual-eval (`0`, `'b-xor3`, list (*a*, *b*, *c*), *state*, *netlist*)
= list (f-xor3 (*a*, *b*, *c*)))

EVENT: Enable b-xor3$value; name this event ‘b-xor3$value-off’.

DEFINITION: del2& (*netlist*) = **t**

EVENT: Enable del2&; name this event ‘del2&-off’.

DEFINITION: DEL2\$NETLIST = **nil**

THEOREM: del2\$value
del2& (*netlist*)
→ (dual-eval (`0`, `'del2`, list (a), *state*, *netlist*) = list (f-buf (a)))

EVENT: Enable del2\$value; name this event 'del2\$value-off'.

DEFINITION: del4& (*netlist*) = **t**

EVENT: Enable del4&; name this event 'del4&-off'.

DEFINITION: DEL4\$NETLIST = **nil**

THEOREM: del4\$value
del4& (*netlist*)
→ (dual-eval (`0`, `'del4`, list (a), *state*, *netlist*) = list (f-buf (a)))

EVENT: Enable del4\$value; name this event 'del4\$value-off'.

DEFINITION: del10& (*netlist*) = **t**

EVENT: Enable del10&; name this event 'del10&-off'.

DEFINITION: DEL10\$NETLIST = **nil**

THEOREM: del10\$value
del10& (*netlist*)
→ (dual-eval (`0`, `'del10`, list (a), *state*, *netlist*) = list (f-buf (a)))

EVENT: Enable del10\$value; name this event 'del10\$value-off'.

DEFINITION: procmon& (*netlist*) = **t**

EVENT: Enable procmon&; name this event 'procmon&-off'.

DEFINITION: PROCMON\$NETLIST = **nil**

THEOREM: procmon\$value
procmon& (*netlist*)
→ (dual-eval (`0`, `'procmon`, list (a, e, s, n), *state*, *netlist*)
= list (f-if (s, f-if (e, f-if (a, **f**, **f**), a), n)))

EVENT: Enable procmon$value; name this event 'procmon$value-off'.

DEFINITION: dp-ram-16x32& (*netlist*) = **t**

EVENT: Enable dp-ram-16x32&; name this event 'dp-ram-16x32&-off'.

DEFINITION: DP-RAM-16X32$NETLIST = **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,

$$
\begin{array}{l}
\quad\quad\quad\quad d22, \\
\quad\quad\quad\quad d23, \\
\quad\quad\quad\quad d24, \\
\quad\quad\quad\quad d25, \\
\quad\quad\quad\quad d26, \\
\quad\quad\quad\quad d27, \\
\quad\quad\quad\quad d28, \\
\quad\quad\quad\quad d29, \\
\quad\quad\quad\quad d30, \\
\quad\quad\quad\quad d31), \\
\quad\quad\quad state, \\
\quad\quad\quad netlist)
\end{array}
$$

$$
\begin{array}{ll}
= & \text{dual-port-ram-value}(\texttt{32}, \\
& \quad \texttt{4}, \\
& \quad \text{list}(read\text{-}a0, \\
& \quad\quad read\text{-}a1, \\
& \quad\quad read\text{-}a2, \\
& \quad\quad read\text{-}a3, \\
& \quad\quad write\text{-}b0, \\
& \quad\quad write\text{-}b1, \\
& \quad\quad write\text{-}b2, \\
& \quad\quad write\text{-}b3, \\
& \quad\quad wen, \\
& \quad\quad d0, \\
& \quad\quad d1, \\
& \quad\quad d2, \\
& \quad\quad d3, \\
& \quad\quad d4, \\
& \quad\quad d5, \\
& \quad\quad d6, \\
& \quad\quad d7, \\
& \quad\quad d8, \\
& \quad\quad d9, \\
& \quad\quad d10, \\
& \quad\quad d11, \\
& \quad\quad d12, \\
& \quad\quad d13, \\
& \quad\quad d14, \\
& \quad\quad d15, \\
& \quad\quad d16, \\
& \quad\quad d17, \\
& \quad\quad d18, \\
& \quad\quad d19, \\
& \quad\quad d20,
\end{array}
$$

$d21$,
$d22$,
$d23$,
$d24$,
$d25$,
$d26$,
$d27$,
$d28$,
$d29$,
$d30$,
$d31$),
state))

EVENT: Enable dp-ram-16x32$value; name this event ‘dp-ram-16x32$value-off’.

DEFINITION: fd1& (*netlist*) = **t**

EVENT: Enable fd1&; name this event ‘fd1&-off’.

DEFINITION: FD1$NETLIST = **nil**

THEOREM: fd1$value
fd1& (*netlist*)
→ (dual-eval (0, 'fd1, list (*d*, *cp*), *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*)))

EVENT: Enable fd1$value; name this event ‘fd1$value-off’.

DEFINITION: fd1s& (*netlist*) = **t**

EVENT: Enable fd1s&; name this event ‘fd1s&-off’.

DEFINITION: FD1S$NETLIST = **nil**

THEOREM: fd1s$value
fd1s& (*netlist*)
→ (dual-eval (0, 'fd1s, list (*d*, *cp*, *ti*, *te*), *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*)))

EVENT: Enable fd1s$value; name this event ‘fd1s$value-off’.

DEFINITION: fd1sp& (*netlist*) = **t**

EVENT: Enable fd1sp&; name this event 'fd1sp&-off'.

DEFINITION: FD1SP$NETLIST = **nil**

THEOREM: fd1sp$value
fd1sp& (*netlist*)
→ (dual-eval (`0`, `'fd1sp`, list (d, cp, ti, te), *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*)))

EVENT: Enable fd1sp$value; name this event 'fd1sp$value-off'.

DEFINITION: fd1slp& (*netlist*) = **t**

EVENT: Enable fd1slp&; name this event 'fd1slp&-off'.

DEFINITION: FD1SLP$NETLIST = **nil**

THEOREM: fd1slp$value
fd1slp& (*netlist*)
→ (dual-eval (`0`, `'fd1slp`, list (d, cp, ld, ti, te), *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*)))

EVENT: Enable fd1slp$value; name this event 'fd1slp$value-off'.

DEFINITION: id& (*netlist*) = **t**

EVENT: Enable id&; name this event 'id&-off'.

DEFINITION: ID$NETLIST = **nil**

THEOREM: id$value
id& (*netlist*) → (dual-eval (`0`, `'id`, list (a), *state*, *netlist*) = list (a))

EVENT: Enable id$value; name this event 'id$value-off'.

DEFINITION: mem-32x32& (*netlist*) = **t**

EVENT: Enable mem-32x32&; name this event 'mem-32x32&-off'.

DEFINITION: MEM-32X32$NETLIST = **nil**

THEOREM: mem-32x32$value
mem-32x32& (*netlist*)
$\rightarrow$ (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: ram-enable-circuit& (*netlist*) = **t**

Event: Enable ram-enable-circuit&; name this event 'ram-enable-circuit&-off'.

Definition: RAM-ENABLE-CIRCUIT$NETLIST = **nil**

Theorem: ram-enable-circuit$value
ram-enable-circuit& (*netlist*)
→ (dual-eval (`0`,
`'ram-enable-circuit`,
list (*clk*, *test-regfile-*, *disable-regfile-*, *we*),
state,
netlist)
= list (f-nand (*disable-regfile-*, *we*)))

Event: Enable ram-enable-circuit$value; name this event 'ram-enable-circuit$value-off'.

Definition: t-buf& (*netlist*) = **t**

Event: Enable t-buf&; name this event 't-buf&-off'.

Definition: T-BUF$NETLIST = **nil**

Theorem: t-buf$value
t-buf& (*netlist*)
→ (dual-eval (`0`, `'t-buf`, list (*e*, *a*), *state*, *netlist*) = list (ft-buf (*e*, *a*)))

Event: Enable t-buf$value; name this event 't-buf$value-off'.

Definition: t-wire& (*netlist*) = **t**

Event: Enable t-wire&; name this event 't-wire&-off'.

DEFINITION: T-WIRE$NETLIST = **nil**

THEOREM: t-wire$value
t-wire& (*netlist*)
→ (dual-eval (0, 't-wire, list (a, b), *state*, *netlist*)
= list (ft-wire (a, b)))

EVENT: Enable t-wire$value; name this event 't-wire$value-off'.

DEFINITION: pullup& (*netlist*) = **t**

EVENT: Enable pullup&; name this event 'pullup&-off'.

DEFINITION: PULLUP$NETLIST = **nil**

THEOREM: pullup$value
pullup& (*netlist*)
→ (dual-eval (0, 'pullup, list (a), *state*, *netlist*) = list (f-pullup (a)))

EVENT: Enable pullup$value; name this event 'pullup$value-off'.

DEFINITION: ttl-bidirect& (*netlist*) = **t**

EVENT: Enable ttl-bidirect&; name this event 'ttl-bidirect&-off'.

DEFINITION: TTL-BIDIRECT$NETLIST = **nil**

THEOREM: ttl-bidirect$value
ttl-bidirect& (*netlist*)
→ (dual-eval (0, 'ttl-bidirect, list (*io*, a, *en*, *pi*), *state*, *netlist*)
= list (ft-buf (f-not (*en*), a),
f-buf (ft-wire (*io*, ft-buf (f-not (*en*), a))),
f-nand (ft-wire (*io*, ft-buf (f-not (*en*), a)), *pi*)))

EVENT: Enable ttl-bidirect$value; name this event 'ttl-bidirect$value-off'.

DEFINITION: ttl-clk-input& (*netlist*) = **t**

EVENT: Enable ttl-clk-input&; name this event 'ttl-clk-input&-off'.

DEFINITION: TTL-CLK-INPUT$NETLIST = **nil**

THEOREM: ttl-clk-input$value
ttl-clk-input& (*netlist*)
→ (dual-eval (0, `'ttl-clk-input`, list (*a*, *pi*), *state*, *netlist*)
= list (f-buf (*a*), f-nand (*a*, *pi*)))

EVENT: Enable ttl-clk-input$value; name this event 'ttl-clk-input$value-off'.

DEFINITION: ttl-input& (*netlist*) = **t**

EVENT: Enable ttl-input&; name this event 'ttl-input&-off'.

DEFINITION: TTL-INPUT$NETLIST = **nil**

THEOREM: ttl-input$value
ttl-input& (*netlist*)
→ (dual-eval (0, `'ttl-input`, list (*a*, *pi*), *state*, *netlist*)
= list (f-buf (*a*), f-nand (*a*, *pi*)))

EVENT: Enable ttl-input$value; name this event 'ttl-input$value-off'.

DEFINITION: ttl-output& (*netlist*) = **t**

EVENT: Enable ttl-output&; name this event 'ttl-output&-off'.

DEFINITION: TTL-OUTPUT$NETLIST = **nil**

THEOREM: ttl-output$value
ttl-output& (*netlist*)
→ (dual-eval (0, `'ttl-output`, list (*a*), *state*, *netlist*)
= list (f-buf (*a*)))

EVENT: Enable ttl-output$value; name this event 'ttl-output$value-off'.

DEFINITION: ttl-output-parametric& (*netlist*) = **t**

EVENT: Enable ttl-output-parametric&; name this event 'ttl-output-parametric&-off'.

DEFINITION: TTL-OUTPUT-PARAMETRIC$NETLIST = **nil**

THEOREM: ttl-output-parametric$value
ttl-output-parametric& (*netlist*)
→ (dual-eval (0, `'ttl-output-parametric`, list (*a*), *state*, *netlist*)
= list (f-buf (*a*)))

EVENT: Enable ttl-output-parametric$value; name this event 'ttl-output-parametric$value-off'.

DEFINITION: ttl-output-fast& (*netlist*) = **t**

EVENT: Enable ttl-output-fast&; name this event 'ttl-output-fast&-off'.

DEFINITION: TTL-OUTPUT-FAST$NETLIST = **nil**

THEOREM: ttl-output-fast$value
ttl-output-fast& (*netlist*)
→ (dual-eval (0, `'ttl-output-fast`, list (*a*), *state*, *netlist*)
= list (f-buf (*a*)))

EVENT: Enable ttl-output-fast$value; name this event 'ttl-output-fast$value-off'.

DEFINITION: ttl-tri-output& (*netlist*) = **t**

EVENT: Enable ttl-tri-output&; name this event 'ttl-tri-output&-off'.

DEFINITION: TTL-TRI-OUTPUT$NETLIST = **nil**

THEOREM: ttl-tri-output$value
ttl-tri-output& (*netlist*)
→ (dual-eval (0, `'ttl-tri-output`, list (*a*, *en*), *state*, *netlist*)
= list (ft-buf (f-not (*en*), *a*)))

EVENT: Enable ttl-tri-output$value; name this event 'ttl-tri-output$value-off'.

DEFINITION: ttl-tri-output-fast& (*netlist*) = **t**

EVENT: Enable ttl-tri-output-fast&; name this event 'ttl-tri-output-fast&-off'.

DEFINITION: TTL-TRI-OUTPUT-FAST$NETLIST = **nil**

THEOREM: ttl-tri-output-fast$value
ttl-tri-output-fast& (*netlist*)
→ (dual-eval (0, `'ttl-tri-output-fast`, list (*a*, *en*), *state*, *netlist*)
= list (ft-buf (f-not (*en*), *a*)))

EVENT: Enable ttl-tri-output-fast$value; name this event 'ttl-tri-output-fast$value-off'.

DEFINITION: vdd& (*netlist*) = **t**

EVENT: Enable vdd&; name this event 'vdd&-off'.

DEFINITION: VDD$NETLIST = **nil**

THEOREM: vdd$value
vdd& (*netlist*) → (dual-eval (0, `'vdd`, **nil**, *state*, *netlist*) = list (**t**))

EVENT: Enable vdd$value; name this event 'vdd$value-off'.

DEFINITION: vdd-parametric& (*netlist*) = **t**

EVENT: Enable vdd-parametric&; name this event 'vdd-parametric&-off'.

DEFINITION: VDD-PARAMETRIC$NETLIST = **nil**

THEOREM: vdd-parametric$value
vdd-parametric& (*netlist*)
→ (dual-eval (0, `'vdd-parametric`, **nil**, *state*, *netlist*) = list (**t**))

EVENT: Enable vdd-parametric$value; name this event 'vdd-parametric$value-off'.

DEFINITION: vss& (*netlist*) = **t**

EVENT: Enable vss&; name this event 'vss&-off'.

DEFINITION: VSS$NETLIST = **nil**

THEOREM: vss$value
vss& (*netlist*) → (dual-eval (0, `'vss`, **nil**, *state*, *netlist*) = list (**f**))

EVENT: Enable vss$value; name this event 'vss$value-off'.

THEOREM: dp-ram-16x32$state
dp-ram-16x32& (*netlist*)
→ (dual-eval (2,
`'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-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*)

$\rightarrow$ (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$)
$\rightarrow$ (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$)
$\rightarrow$ (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$)
$\rightarrow$ (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$)
$\rightarrow$ (dual-eval (2,
'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-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:
b-buf& (*netlist*)
= ((lookup-module (`'b-buf`, *netlist*) = B-BUF*)
  ∧ b-nbuf& (delete-module (`'b-buf`, *netlist*)))

EVENT: Enable b-buf&; name this event 'b-buf&-off'.

DEFINITION: B-BUF$NETLIST = cons (B-BUF*, B-NBUF$NETLIST)

THEOREM: b-buf$value
b-buf& (*netlist*)
→ (dual-eval (`0`, `'b-buf`, list (*in*), *state*, *netlist*) = list (f-buf (*in*)))

EVENT: Enable b-buf$value; name this event 'b-buf$value-off'.

DEFINITION:
B-BUF-PWR*
=
```
'(b-buf-pwr
  (in)
  (out)
  ((g0 (out-) b-not (in))
   (g1 (out) b-not-b4ip (out-)))
  nil)
```

DEFINITION:
b-buf-pwr& (*netlist*)
= ((lookup-module (`'b-buf-pwr`, *netlist*) = B-BUF-PWR*)
  ∧ b-not& (delete-module (`'b-buf-pwr`, *netlist*))
  ∧ b-not-b4ip& (delete-module (`'b-buf-pwr`, *netlist*)))

EVENT: Enable b-buf-pwr&; name this event 'b-buf-pwr&-off'.

DEFINITION:
B-BUF-PWR$NETLIST = cons (B-BUF-PWR*, B-NOT$NETLIST ∪ B-NOT-B4IP$NETLIST)

THEOREM: b-buf-pwr$value
b-buf-pwr& (*netlist*)
→ (dual-eval (`0`, `'b-buf-pwr`, list (*in*), *state*, *netlist*)
   = list (f-buf (*in*)))

EVENT: Enable b-buf-pwr$value; name this event 'b-buf-pwr$value-off'.

THEOREM: dp-ram-16x32-args-crock

((length (a) = 4)
 ∧ properp (a)
 ∧ (length (b) = 4)
 ∧ properp (b)
 ∧ (length (c) = 32)
 ∧ properp (c))
→ (append (a, append (b, cons (x, c)))
    = append (list-as-collected-nth (a, 4, 0),
        append (list-as-collected-nth (b, 4, 0),
            cons (x, list-as-collected-nth (c, 32, 0)))))

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*)
 ∧ (length (a) = 4)
 ∧ properp (a)
 ∧ (length (b) = 4)
 ∧ properp (b)
 ∧ (length (c) = 32)
 ∧ properp (c))
→ (dual-eval (0,
        'dp-ram-16x32,
        append (a, append (b, cons (x, c))),
        state,
        netlist)
    = dual-port-ram-value (32, 4, append (a, append (b, cons (x, c))), *state*))

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*)
 ∧ (length (a) = 4)
 ∧ properp (a)
 ∧ (length (b) = 4)
 ∧ properp (b)
 ∧ (length (c) = 32)
 ∧ properp (c))
→ (dual-eval (2,
        'dp-ram-16x32,
        append (a, append (b, cons (x, c))),

$state$,
$netlist$)
= dual-port-ram-state(32, 4, append(a, append(b, cons(x, c))), $state$))

EVENT: Enable dp-ram-16x32$structured-state; name this event 'dp-ram-16x32$structured-state-off'.

THEOREM: mem-32x32-args-crock
((length($addr$) = 32)
∧ properp($addr$)
∧ (length($data$) = 32)
∧ properp($data$))
→ (cons(rw-, cons($strobe$-, append($addr$, $data$)))
= cons(rw-,
cons($strobe$-,
append(list-as-collected-nth($addr$, 32, 0),
list-as-collected-nth($data$, 32, 0)))))

EVENT: Enable mem-32x32-args-crock; name this event 'mem-32x32-args-crock-off'.

THEOREM: list-32-nth-collapse
(properp($args$) ∧ (length($args$) = 32))
→ (list(nth(0, $args$),
nth(1, $args$),
nth(2, $args$),
nth(3, $args$),
nth(4, $args$),
nth(5, $args$),
nth(6, $args$),
nth(7, $args$),
nth(8, $args$),
nth(9, $args$),
nth(10, $args$),
nth(11, $args$),
nth(12, $args$),
nth(13, $args$),
nth(14, $args$),
nth(15, $args$),
nth(16, $args$),
nth(17, $args$),
nth(18, $args$),
nth(19, $args$),

nth (20, *args*),
nth (21, *args*),
nth (22, *args*),
nth (23, *args*),
nth (24, *args*),
nth (25, *args*),
nth (26, *args*),
nth (27, *args*),
nth (28, *args*),
nth (29, *args*),
nth (30, *args*),
nth (31, *args*))
= *args*)

EVENT: Enable list-32-nth-collapse; name this event 'list-32-nth-collapse-off'.

THEOREM: mem-32x32$structured-value-1
(mem-32x32& (*netlist*)
∧ (length (*addr*) = 32)
∧ properp (*addr*)
∧ (length (*data*) = 32)
∧ properp (*data*))
→ (dual-eval (0,
'mem-32x32,
cons (*rw-*, cons (*strobe-*, append (*addr*, *data*))),
state,
netlist)
= memory-value (*state*,
strobe-,
rw-,
list (nth (0, *addr*),
nth (1, *addr*),
nth (2, *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*),

$$
\begin{array}{l}
\quad\operatorname{nth}(\mathtt{14},\ addr),\\
\quad\operatorname{nth}(\mathtt{15},\ addr),\\
\quad\operatorname{nth}(\mathtt{16},\ addr),\\
\quad\operatorname{nth}(\mathtt{17},\ addr),\\
\quad\operatorname{nth}(\mathtt{18},\ addr),\\
\quad\operatorname{nth}(\mathtt{19},\ addr),\\
\quad\operatorname{nth}(\mathtt{20},\ addr),\\
\quad\operatorname{nth}(\mathtt{21},\ addr),\\
\quad\operatorname{nth}(\mathtt{22},\ addr),\\
\quad\operatorname{nth}(\mathtt{23},\ addr),\\
\quad\operatorname{nth}(\mathtt{24},\ addr),\\
\quad\operatorname{nth}(\mathtt{25},\ addr),\\
\quad\operatorname{nth}(\mathtt{26},\ addr),\\
\quad\operatorname{nth}(\mathtt{27},\ addr),\\
\quad\operatorname{nth}(\mathtt{28},\ addr),\\
\quad\operatorname{nth}(\mathtt{29},\ addr),\\
\quad\operatorname{nth}(\mathtt{30},\ addr),\\
\quad\operatorname{nth}(\mathtt{31},\ addr)),\\
\operatorname{list}(\operatorname{nth}(\mathtt{0},\ data),\\
\quad\operatorname{nth}(\mathtt{1},\ data),\\
\quad\operatorname{nth}(\mathtt{2},\ data),\\
\quad\operatorname{nth}(\mathtt{3},\ data),\\
\quad\operatorname{nth}(\mathtt{4},\ data),\\
\quad\operatorname{nth}(\mathtt{5},\ data),\\
\quad\operatorname{nth}(\mathtt{6},\ data),\\
\quad\operatorname{nth}(\mathtt{7},\ data),\\
\quad\operatorname{nth}(\mathtt{8},\ data),\\
\quad\operatorname{nth}(\mathtt{9},\ data),\\
\quad\operatorname{nth}(\mathtt{10},\ data),\\
\quad\operatorname{nth}(\mathtt{11},\ data),\\
\quad\operatorname{nth}(\mathtt{12},\ data),\\
\quad\operatorname{nth}(\mathtt{13},\ data),\\
\quad\operatorname{nth}(\mathtt{14},\ data),\\
\quad\operatorname{nth}(\mathtt{15},\ data),\\
\quad\operatorname{nth}(\mathtt{16},\ data),\\
\quad\operatorname{nth}(\mathtt{17},\ data),\\
\quad\operatorname{nth}(\mathtt{18},\ data),\\
\quad\operatorname{nth}(\mathtt{19},\ data),\\
\quad\operatorname{nth}(\mathtt{20},\ data),\\
\quad\operatorname{nth}(\mathtt{21},\ data),\\
\quad\operatorname{nth}(\mathtt{22},\ data),\\
\quad\operatorname{nth}(\mathtt{23},\ data),\\
\quad\operatorname{nth}(\mathtt{24},\ data),\\
\quad\operatorname{nth}(\mathtt{25},\ data),
\end{array}
$$

nth (26, *data*),
nth (27, *data*),
nth (28, *data*),
nth (29, *data*),
nth (30, *data*),
nth (31, *data*))))

EVENT: Enable mem-32x32$structured-value-1; name this event 'mem-32x32$structured-value-1-off'.

THEOREM: mem-32x32$structured-value
(mem-32x32& (*netlist*)
∧ (length (*addr*) = 32)
∧ properp (*addr*)
∧ (length (*data*) = 32)
∧ properp (*data*))
→ (dual-eval (0,
'mem-32x32,
cons (*rw-*, cons (*strobe-*, append (*addr*, *data*))),
state,
netlist)
= memory-value (*state*, *strobe-*, *rw-*, *addr*, *data*))

EVENT: Enable mem-32x32$structured-value; name this event 'mem-32x32$structured-value-off'.

THEOREM: mem-32x32$structured-state-1
(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-,
list (nth (0, *addr*),
nth (1, *addr*),
nth (2, *addr*),

$$
\begin{aligned}
&\quad \mathrm{nth}\,(3,\ addr),\\
&\quad \mathrm{nth}\,(4,\ addr),\\
&\quad \mathrm{nth}\,(5,\ addr),\\
&\quad \mathrm{nth}\,(6,\ addr),\\
&\quad \mathrm{nth}\,(7,\ addr),\\
&\quad \mathrm{nth}\,(8,\ addr),\\
&\quad \mathrm{nth}\,(9,\ addr),\\
&\quad \mathrm{nth}\,(10,\ addr),\\
&\quad \mathrm{nth}\,(11,\ addr),\\
&\quad \mathrm{nth}\,(12,\ addr),\\
&\quad \mathrm{nth}\,(13,\ addr),\\
&\quad \mathrm{nth}\,(14,\ addr),\\
&\quad \mathrm{nth}\,(15,\ addr),\\
&\quad \mathrm{nth}\,(16,\ addr),\\
&\quad \mathrm{nth}\,(17,\ addr),\\
&\quad \mathrm{nth}\,(18,\ addr),\\
&\quad \mathrm{nth}\,(19,\ addr),\\
&\quad \mathrm{nth}\,(20,\ addr),\\
&\quad \mathrm{nth}\,(21,\ addr),\\
&\quad \mathrm{nth}\,(22,\ addr),\\
&\quad \mathrm{nth}\,(23,\ addr),\\
&\quad \mathrm{nth}\,(24,\ addr),\\
&\quad \mathrm{nth}\,(25,\ addr),\\
&\quad \mathrm{nth}\,(26,\ addr),\\
&\quad \mathrm{nth}\,(27,\ addr),\\
&\quad \mathrm{nth}\,(28,\ addr),\\
&\quad \mathrm{nth}\,(29,\ addr),\\
&\quad \mathrm{nth}\,(30,\ addr),\\
&\quad \mathrm{nth}\,(31,\ addr)),\\
&\mathrm{list}\,(\mathrm{nth}\,(0,\ data),\\
&\quad \mathrm{nth}\,(1,\ data),\\
&\quad \mathrm{nth}\,(2,\ data),\\
&\quad \mathrm{nth}\,(3,\ data),\\
&\quad \mathrm{nth}\,(4,\ data),\\
&\quad \mathrm{nth}\,(5,\ data),\\
&\quad \mathrm{nth}\,(6,\ data),\\
&\quad \mathrm{nth}\,(7,\ data),\\
&\quad \mathrm{nth}\,(8,\ data),\\
&\quad \mathrm{nth}\,(9,\ data),\\
&\quad \mathrm{nth}\,(10,\ data),\\
&\quad \mathrm{nth}\,(11,\ data),\\
&\quad \mathrm{nth}\,(12,\ data),\\
&\quad \mathrm{nth}\,(13,\ data),\\
&\quad \mathrm{nth}\,(14,\ data),
\end{aligned}
$$

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
$(body \simeq \textbf{nil}) \rightarrow \text{unbound-in-body}\,(name,\ body)$

THEOREM: unbound-in-body-listp
$\text{unbound-in-body}\,(name,\ \text{cons}\,(occurrence,\ rest))$
$= \quad ((name \notin \text{occ-outputs}\,(occurrence)) \land \text{unbound-in-body}\,(name,\ rest))$

DEFINITION:
$\text{all-unbound-in-body}\,(names,\ body)$
$= \quad \textbf{if}\ body \simeq \textbf{nil}\ \textbf{then}\ \textbf{t}$
$\quad\quad \textbf{else}\ \text{disjoint}\,(names,\ \text{occ-outputs}\,(\text{car}\,(body)))$
$\quad\quad\quad\quad \land \quad \text{all-unbound-in-body}\,(names,\ \text{cdr}\,(body))\ \textbf{endif}$

EVENT: Enable all-unbound-in-body; name this event ‘all-unbound-in-body-off’.

THEOREM: all-unbound-in-body-nlistp
$(body \simeq \textbf{nil}) \rightarrow \text{all-unbound-in-body}\,(names,\ body)$

THEOREM: all-unbound-in-body-listp
$\text{all-unbound-in-body}\,(names,\ \text{cons}\,(occurrence,\ rest))$
$= \quad (\text{disjoint}\,(names,\ \text{occ-outputs}\,(occurrence))$
$\quad\quad \land \quad \text{all-unbound-in-body}\,(names,\ rest))$

THEOREM: all-unbound-in-body-append
$\text{all-unbound-in-body}\,(\text{append}\,(names1,\ names2),\ body)$
$= \quad (\text{all-unbound-in-body}\,(names1,\ body)$
$\quad\quad \land \quad \text{all-unbound-in-body}\,(names2,\ body))$

THEOREM: all-unbound-in-body-cons
$\text{all-unbound-in-body}\,(\text{cons}\,(name,\ names),\ body)$
$= \quad (\text{unbound-in-body}\,(name,\ body) \land \text{all-unbound-in-body}\,(names,\ body))$

THEOREM: all-unbound-in-body-nlistp-names
$(names \simeq \textbf{nil}) \rightarrow \text{all-unbound-in-body}\,(names,\ body)$

THEOREM: unbound-in-body-dual-eval-1
$((flag = \texttt{1}) \land \text{unbound-in-body}\,(name,\ body))$
$\rightarrow \quad (\text{value}\,(name,\ \text{dual-eval}\,(flag,\ body,\ bindings,\ state\text{-}bindings,\ netlist))$
$\quad\quad = \quad \text{value}\,(name,\ bindings))$

THEOREM: all-unbound-in-body-dual-eval-1
$((flag = \texttt{1}) \land \text{all-unbound-in-body}\,(names,\ body))$
$\rightarrow \quad (\text{collect-value}\,(names,$
$\quad\quad\quad\quad \text{dual-eval}\,(flag,\ body,\ bindings,\ state\text{-}bindings,\ netlist))$
$\quad\quad = \quad \text{collect-value}\,(names,\ bindings))$

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*$destructure
(car (v-buf* (*n*)) = index (`'v-buf`, *n*))
∧ (cadr (v-buf* (*n*)) = indices (`'a`, `0`, *n*))
∧ (caddr (v-buf* (*n*)) = indices (`'y`, `0`, *n*))
∧ (cadddr (v-buf* (*n*)) = v-buf$body (`0`, *n*))
∧ (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-buf& ($netlist$, n)
$=$ ((lookup-module (index (`'v-buf`, n), $netlist$) = v-buf* (n))
  $\land$ b-buf& (delete-module (index (`'v-buf`, n), $netlist$)))

EVENT: Enable v-buf&; name this event 'v-buf&-off'.

DEFINITION: v-buf$netlist ($n$) = cons (v-buf* ($n$), B-BUF$NETLIST)

THEOREM: v-buf$unbound-in-body
($l < m$) $\rightarrow$ unbound-in-body (index (`'y`, l), v-buf$body ($m$, n))

EVENT: Enable v-buf$unbound-in-body; name this event 'v-buf$unbound-in-body-off'.

THEOREM: v-buf$body-value
(b-buf& ($netlist$) $\land$ ($body$ = v-buf$body ($m$, n)))
$\rightarrow$ (collect-value (indices (`'y`, m, n),
  dual-eval (`1`, $body$, $bindings$, $state\text{-}bindings$, $netlist$))
  $=$ v-threefix (collect-value (indices (`'a`, m, n), $bindings$)))

EVENT: Enable v-buf$body-value; name this event 'v-buf$body-value-off'.

THEOREM: v-buf$value
(v-buf& ($netlist$, n) $\land$ properp (a) $\land$ (length (a) = n))
$\rightarrow$ (dual-eval (`0`, index (`'v-buf`, n), a, $state$, $netlist$) = v-threefix (a))

EVENT: Enable v-buf$value; name this event 'v-buf$value-off'.

DEFINITION:
v-or$body ($m$, n)
$=$ **if** $n \simeq$ `0` **then nil**
  else cons (list (index (`'g`, m),
    list (index (`'y`, m)),
    `'b-or`,
    list (index (`'a`, m), index (`'b`, m))),
   v-or$body ($1 + m$, $n - 1$)) **endif**

EVENT: Enable v-or$body; name this event 'v-or$body-off'.

DEFINITION:
v-or* (n)
$=$ cons (index (`'v-or`, n),
  cons (append (indices (`'a`, `0`, n), indices (`'b`, `0`, n)),
   cons (indices (`'y`, `0`, n), cons (v-or$body (`0`, n), `'(nil)`))))

THEOREM: v-or*$destructure
(car (v-or* (n)) = index ('`v-or`, n))
∧ (cadr (v-or* (n)) = append (indices ('`a`, `0`, n), indices ('`b`, `0`, n)))
∧ (caddr (v-or* (n)) = indices ('`y`, `0`, n))
∧ (cadddr (v-or* (n)) = v-or$body (`0`, n))
∧ (caddddr (v-or* (n)) = **nil**)

EVENT: Enable v-or*; name this event 'v-or*-off'.

EVENT: Enable v-or*$destructure; name this event 'v-or*$destructure-off'.

DEFINITION:
v-or& (*netlist*, n)
= ((lookup-module (index ('`v-or`, n), *netlist*) = v-or* (n))
∧ b-or& (delete-module (index ('`v-or`, n), *netlist*)))

EVENT: Enable v-or&; name this event 'v-or&-off'.

DEFINITION: v-or$netlist ($n$) = cons (v-or* ($n$), B-OR$NETLIST)

THEOREM: v-or$unbound-in-body
($l < m$) → unbound-in-body (index ('`y`, l), v-or$body ($m$, n))

EVENT: Enable v-or$unbound-in-body; name this event 'v-or$unbound-in-body-off'.

THEOREM: v-or$body-value
(b-or& (*netlist*) ∧ (*body* = v-or$body ($m$, n)))
→ (collect-value (indices ('`y`, m, n),
dual-eval (`1`, *body*, *bindings*, *state-bindings*, *netlist*))
= fv-or (collect-value (indices ('`a`, m, n), *bindings*),
collect-value (indices ('`b`, m, n), *bindings*)))

EVENT: Enable v-or$body-value; name this event 'v-or$body-value-off'.

THEOREM: v-or$value
(v-or& (*netlist*, n)
∧ (properp (a) ∧ (length (a) = n))
∧ properp (b)
∧ (length (b) = n))
→ (dual-eval (`0`, index ('`v-or`, n), append (a, b), *state*, *netlist*)
= fv-or (a, b))

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*$destructure
(car (v-xor* (n)) = index (`'v-xor`, n))
∧ (cadr (v-xor* (n)) = append (indices (`'a`, `0`, n), indices (`'b`, `0`, n)))
∧ (caddr (v-xor* (n)) = indices (`'y`, `0`, n))
∧ (cadddr (v-xor* (n)) = v-xor$body (`0`, n))
∧ (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))
∧ 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$) → unbound-in-body (index (`'y`, l), v-xor$body ($m$, n))

EVENT: Enable v-xor$unbound-in-body; name this event 'v-xor$unbound-in-body-off'.

THEOREM: v-xor$body-value
(b-xor& (*netlist*) ∧ (*body* = v-xor$body (*m*, *n*)))
→ (collect-value (indices ('y, *m*, *n*),
dual-eval (1, *body*, *bindings*, *state-bindings*, *netlist*))
= fv-xor (collect-value (indices ('a, *m*, *n*), *bindings*),
collect-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-xor& (*netlist*, *n*)
∧ (properp (*a*) ∧ (length (*a*) = *n*))
∧ properp (*b*)
∧ (length (*b*) = *n*))
→ (dual-eval (0, index ('v-xor, *n*), append (*a*, *b*), *state*, *netlist*)
= fv-xor (*a*, *b*))

EVENT: Enable v-xor$value; name this event 'v-xor$value-off'.

DEFINITION:
v-pullup$body (*m*, *n*)
= **if** *n* ≃ 0 **then nil**
else cons (list (index ('g, *m*),
list (index ('y, *m*)),
'pullup,
list (index ('a, *m*))),
v-pullup$body (1 + *m*, *n* − 1)) **endif**

EVENT: Enable v-pullup$body; name this event 'v-pullup$body-off'.

DEFINITION:
v-pullup* (*n*)
= cons (index ('v-pullup, *n*),
cons (indices ('a, 0, *n*),
cons (indices ('y, 0, *n*), cons (v-pullup$body (0, *n*), '(nil)))))

THEOREM: v-pullup*$destructure
(car (v-pullup* (*n*)) = index ('v-pullup, *n*))
∧ (cadr (v-pullup* (*n*)) = indices ('a, 0, *n*))
∧ (caddr (v-pullup* (*n*)) = indices ('y, 0, *n*))

$\wedge$ (cadddr (v-pullup* (n)) = v-pullup\$body (0, n))
$\wedge$ (caddddr (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) ∨ (¬ 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 ≃ 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 (*flg*, *x0*, *x1*, *type*, *netlist*)

```
=  case on flg:
   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 ≃ 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 ∪ b-and$netlist)

Definition:
half-adder& (*netlist*)
= ((lookup-module ('`half-adder`, *netlist*) = half-adder*)
  ∧ b-xor& (delete-module ('`half-adder`, *netlist*))
  ∧ b-and& (delete-module ('`half-adder`, *netlist*)))

Event: Enable half-adder&; name this event 'half-adder&-off'.

Theorem: half-adder$value
half-adder& (*netlist*)
→ (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 ∪ B-OR$NETLIST)

DEFINITION:
full-adder& (*netlist*)
= ((lookup-module (`'full-adder`, *netlist*) = FULL-ADDER*)
  ∧ half-adder& (delete-module (`'full-adder`, *netlist*))
  ∧ 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$value
full-adder& (*netlist*)
→ (dual-eval (`0`, `'full-adder`, list (a, b, c), *state*, *netlist*)
  = f$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$full-adder=full-adder
(boolp (a) ∧ boolp (b) ∧ boolp (c))
→ (f$full-adder ($a$, b, c) = full-adder (a, b, c))

DEFINITION:
M1*
=
```
'(m1
  (clk en sel d q)
  (q)
  ((mux (b) b-if (sel d q))
   (latch (a an) fd1 (b clk))
   (tbuf (q) t-buf (en a)))
  latch)
```

DEFINITION:
M1$NETLIST = cons (M1*, (B-IF$NETLIST ∪ FD1$NETLIST) ∪ T-BUF$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*)
$\rightarrow$ (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** $a \simeq$ **nil** **then** list (c)
else cons (xor (c, xor (car (a), car (b))),
bv-adder ((car (a) $\wedge$ car (b))
$\vee$ (car (a) $\wedge$ c)
$\vee$ (car (b) $\wedge$ c),
cdr (a),
cdr (b))) **endif**

DEFINITION:
bv-adder-body (m, n)
= **if** $n \simeq$ 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: bv-adder*$destructure
(car (bv-adder* (n)) = index ('**bv-adder**, n))
∧ (cadr (bv-adder* (n))
  = cons (index ('**carry**, 0),
      append (indices ('**a**, 0, n), indices ('**b**, 0, n))))
∧ (caddr (bv-adder* (n))
  = append (indices ('**sum**, 0, n), list (index ('**carry**, n))))
∧ (cadddr (bv-adder* (n)) = bv-adder-body (0, n))
∧ (caddddr (bv-adder* (n)) = **nil**)

EVENT: Enable bv-adder*; name this event 'bv-adder*-off'.

EVENT: Enable bv-adder*$destructure; name this event 'bv-adder*$destructure-off'.

DEFINITION:
bv-adder& (*netlist*, n)
= ((lookup-module (index ('**bv-adder**, n), *netlist*) = bv-adder* (n))
  ∧ full-adder& (delete-module (index ('**bv-adder**, n), *netlist*)))

EVENT: Enable bv-adder&; name this event 'bv-adder&-off'.

DEFINITION:
bv-adder$netlist ($n$) = cons (bv-adder* ($n$), FULL-ADDER$NETLIST)

EVENT: Enable bv-adder$netlist; name this event 'bv-adder$netlist-off'.

DEFINITION:
bv-adder-body$induction ($m$, n, *bindings*, *state-bindings*, *netlist*)
= **if** $n \simeq 0$ **then** *bindings*
  else bv-adder-body$induction (1 + m,
      $n - 1$,
      append (pairlist (list (index ('**sum**,
          m),
        index ('**carry**,
          1 + m)),
        dual-eval (0,
          '**full-adder**,
          collect-value (list (index ('**a**,
              m),
            index ('**b**,
              m),

```
                                                    index ('carry,
                                                           m)),
                                                bindings),
                                         value (index ('g,
                                                       m),
                                                state-bindings),
                                         netlist)),
                           bindings),
                       state-bindings,
                       netlist) endif
```

Theorem: bv-adder$unbound-in-body-sum
$(l < m) \rightarrow$ unbound-in-body (index ('sum, l), bv-adder-body (m, n))

Theorem: bv-adder$unbound-in-body-carry
$(l < (1 + m)) \rightarrow$ unbound-in-body (index ('carry, l), bv-adder-body (m, n))

Event: Enable bv-adder$unbound-in-body-sum; name this event 'bv-adder$unbound-in-body-sum-off'.

Event: Enable bv-adder$unbound-in-body-carry; name this event 'bv-adder$unbound-in-body-carry-off'.

Theorem: bv-adder-body$value

```
 (full-adder& (netlist)
  ∧   boolp (value (index ('carry, m), bindings))
  ∧   bvp (collect-value (indices ('a, m, n), bindings))
  ∧   bvp (collect-value (indices ('b, m, n), bindings)))
 →    (collect-value (append (indices ('sum, m, n),
                             list (index ('carry, n + m))),
                     dual-eval (1,
                                bv-adder-body (m, n),
                                bindings,
                                state-bindings,
                                netlist))
        =   bv-adder (value (index ('carry, m), bindings),
                      collect-value (indices ('a, m, n), bindings),
                      collect-value (indices ('b, m, n), bindings)))
```

Theorem: bv-adder-body-special-case$value

```
 (full-adder& (netlist)
  ∧   boolp (value (index ('carry, 0), bindings))
  ∧   bvp (collect-value (indices ('a, 0, n), bindings))
  ∧   bvp (collect-value (indices ('b, 0, n), bindings)))
```

$\rightarrow$ (collect-value (append (indices (`'sum`, `0`, n), list (index (`'carry`, n))),
dual-eval (`1`,
bv-adder-body (`0`, n),
bindings,
state-bindings,
netlist))
= bv-adder (value (index (`'carry`, `0`), *bindings*),
collect-value (indices (`'a`, `0`, n), *bindings*),
collect-value (indices (`'b`, `0`, n), *bindings*)))

THEOREM: bv-adder$value
(bv-adder& (*netlist*, n)
$\wedge$ boolp (c)
$\wedge$ bvp (a)
$\wedge$ bvp (b)
$\wedge$ (length (a) = n)
$\wedge$ (length (b) = n))
$\rightarrow$ (dual-eval (`0`, index (`'bv-adder`, n), cons (c, append (a, b)), *state*, *netlist*)
= bv-adder (c, a, b))

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:
v-sum (c, a, b)
= **if** $a \simeq$ **nil then nil**
else cons (b-xor (c, b-xor (car (a), car (b))),
v-sum (b-or3 (b-and (car (a), car (b)),
b-and (car (a), c),
b-and (car (b), c)),
cdr (a),
cdr (b))) **endif**

EVENT: Enable v-sum; name this event 'v-sum-off'.

THEOREM: bvp-sum
bvp (v-sum (c, a, b))

THEOREM: length-sum
length (v-sum (c, a, b)) = length (a)

THEOREM: v-sum-congruence
c → ((v-sum (c, a, b) = v-sum (**t**, a, b)) = **t**)

THEOREM: a-1+1=a
(bvp (a) ∧ c) → (v-sum (c, v-not (nat-to-v (0, length (a))), a) = a)

DEFINITION:
v-carry (c, a, b)
= **if** $a \simeq$ **nil then** boolfix (c)
else v-carry ((car (a) ∧ car (b))
∨ ((car (a) ∨ car (b)) ∧ c),
cdr (a),
cdr (b)) **endif**

EVENT: Enable v-carry; name this event 'v-carry-off'.

DEFINITION:
v-propagate (a, b)
= **if** $a \simeq$ **nil then t**
else (car (a) ∨ car (b)) ∧ v-propagate (cdr (a), cdr (b)) **endif**

EVENT: Enable v-propagate; name this event 'v-propagate-off'.

DEFINITION:
v-generate (a, b)
= **if** $a \simeq$ **nil then f**
else v-generate (cdr (a), cdr (b))
∨ (car (a) ∧ car (b) ∧ v-propagate (cdr (a), cdr (b))) **endif**

EVENT: Enable v-generate; name this event 'v-generate-off'.

DEFINITION: f$t-carry ($c$, $prop$, gen) = f-or (f-and (c, $prop$), gen)

EVENT: Enable f$t-carry; name this event 'f$t-carry-off'.

DEFINITION: t-carry (c, $prop$, gen) = b-not (ao6 (c, $prop$, gen))

EVENT: Enable t-carry; name this event 't-carry-off'.

THEOREM: f$t-carry=t-carry
(boolp (c) ∧ boolp ($prop$) ∧ boolp (gen))
→ (f$t-carry ($c$, $prop$, gen) = t-carry (c, $prop$, 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*
```
=   '(t-carry
      (c prop gen)
      (z)
      ((g0 (z-) ao6 (c prop gen)) (g1 (z) b-not (z-)))
      nil)
```

DEFINITION:
t-carry& (*netlist*)
= ((lookup-module (`'t-carry`, *netlist*) = T-CARRY*)
  ∧ ao6& (delete-module (`'t-carry`, *netlist*))
  ∧ b-not& (delete-module (`'t-carry`, *netlist*)))

EVENT: Enable t-carry&; name this event 't-carry&-off'.

DEFINITION:
T-CARRY$NETLIST = cons (T-CARRY*, AO6$NETLIST ∪ B-NOT$NETLIST)

THEOREM: t-carry$value
t-carry& (*netlist*)
→ (dual-eval (0, `'t-carry`, list (*c*, *prop*, *gen*), *state*, *netlist*)
  = list (f$t-carry (*c*, *prop*, *gen*)))

EVENT: Enable t-carry$value; name this event 't-carry$value-off'.

THEOREM: v-append-propagate
(length (a) = length (c))
→ (v-propagate (append (a, b), append (c, d))
  = (v-propagate (a, c) ∧ v-propagate (b, d)))

THEOREM: v-propagate-append
(length (a) = length (c))
→ (b-and (v-propagate (a, c), v-propagate (b, d))
  = v-propagate (append (a, b), append (c, d)))

EVENT: Enable v-propagate-append; name this event 'v-propagate-append-off'.

THEOREM: generate-append
(length (a) = length (c))
→ (t-carry (v-generate (a, c), v-propagate (b, d), v-generate (b, d))
  = v-generate (append (a, b), append (c, d)))

THEOREM: append-sum
((length (a) = length (b)) $\wedge$ ($c2$ = v-carry ($c1$, a, b)))
$\rightarrow$ (append (v-sum ($c1$, a, b), v-sum ($c2$, c, d))
= v-sum ($c1$, append (a, c), append (b, d)))

THEOREM: t-carry-p-g-carry
t-carry (c, v-propagate (a, b), v-generate (a, b)) = v-carry (c, a, b)

THEOREM: v-adder-output=v-sum
v-adder-output (c, a, b) = v-sum (c, a, b)

THEOREM: v-adder-carry-out=v-carry
v-adder-carry-out (c, a, b) = v-carry (c, a, b)

DEFINITION:
tv-adder (c, a, b, *tree*)
= **if** *tree* $\simeq$ **nil**
then list (b-or (car (a), car (b)),
b-and (car (a), car (b)),
b-xor (car (a), b-xor (car (b), c)))
else cons (b-and (car (tv-adder (c,
tfirstn (a, *tree*),
tfirstn (b, *tree*),
car (*tree*))),
car (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*))))),
cons (t-carry (cadr (tv-adder (c,
tfirstn (a, *tree*),
tfirstn (b, *tree*),
car (*tree*))),
car (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))),
  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 (cddr (tv-adder (c,
                        tfirstn (a, tree),
                        tfirstn (b, tree),
                        car (tree))),
        cddr (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
((length (a) = tree-size (*tree*)) $\wedge$ (length (b) = tree-size (*tree*)))
$\rightarrow$ (car (tv-adder (c, a, b, *tree*)) = v-propagate (a, b))

THEOREM: cadr-tv-adder-as-g
((length (a) = tree-size (*tree*)) $\wedge$ (length (b) = tree-size (*tree*)))
$\rightarrow$ (cadr (tv-adder (c, a, b, *tree*)) = v-generate (a, b))

THEOREM: cddr-tv-adder-as-sum
((length (a) = tree-size (*tree*)) $\wedge$ (length (b) = tree-size (*tree*)))
$\rightarrow$ (cddr (tv-adder (c, a, b, *tree*)) = v-sum (c, a, b))

THEOREM: equality-of-three-element-lists
(($x \neq 0$) $\wedge$ ($y \neq 0$) $\wedge$ ($z \neq 0$))
$\rightarrow$ ((w = cons (x, cons (y, z)))
= ((car (w) = x) $\wedge$ (cadr (w) = y) $\wedge$ (cddr (w) = z)))

THEOREM: tv-adder-as-p-g-sum
((length (a) = tree-size (*tree*)) $\wedge$ (length (b) = tree-size (*tree*)))
$\rightarrow$ (tv-adder (c, a, b, *tree*)
= cons (v-propagate (a, b), cons (v-generate (a, b), 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 ≃ 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) ≃ 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) ≃ 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) ≃ 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)),
```

indices ('b, 0, tree-size (*tree*)))),
cons (indices ('out, 0, tree-size (*tree*)),
cons (tv-if-body (*tree*), '(nil))))

THEOREM: tv-if*$destructure
(car (tv-if* (*tree*)) = index ('tv-if, tree-number (*tree*)))
$\wedge$ (cadr (tv-if* (*tree*))
= cons ('c,
append (indices ('a, 0, tree-size (*tree*)),
indices ('b, 0, tree-size (*tree*)))))
$\wedge$ (caddr (tv-if* (*tree*)) = indices ('out, 0, tree-size (*tree*)))
$\wedge$ (cadddr (tv-if* (*tree*)) = tv-if-body (*tree*))
$\wedge$ (caddddr (tv-if* (*tree*)) = **nil**)

EVENT: Enable tv-if*; name this event 'tv-if*-off'.

EVENT: Enable tv-if*$destructure; name this event 'tv-if*$destructure-off'.

DEFINITION:
tv-if& (*netlist*, *tree*)
= **if** *tree* $\simeq$ **nil**
then (lookup-module (index ('tv-if, tree-number (*tree*)), *netlist*)
= tv-if* (*tree*))
$\wedge$ b-if& (delete-module (index ('tv-if, tree-number (*tree*)),
netlist))
else (lookup-module (index ('tv-if, tree-number (*tree*)), *netlist*)
= tv-if* (*tree*))
$\wedge$ b-buf& (delete-module (index ('tv-if,
tree-number (*tree*)),
netlist))
$\wedge$ tv-if& (delete-module (index ('tv-if,
tree-number (*tree*)),
netlist),
car (*tree*))
$\wedge$ tv-if& (delete-module (index ('tv-if,
tree-number (*tree*)),
netlist),
cdr (*tree*)) **endif**

EVENT: Enable tv-if&; name this event 'tv-if&-off'.

DEFINITION:
tv-if$netlist (*tree*)

= **if** *tree* $\simeq$ **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* $\simeq$ **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))
$\land$ tv-if-induction (car (*tree*),
`x`,
tfirstn (*a*, *tree*),
tfirstn (*b*, *tree*),
`0`,
delete-module (index (`'tv-if`,
tree-number (*tree*)),
netlist))
$\land$ tv-if-induction (cdr (*tree*),
c,
trestn (*a*, *tree*),
trestn (*b*, *tree*),
`0`,
delete-module (index (`'tv-if`,
tree-number (*tree*)),
netlist))
$\land$ 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)
  ∧  properp (a)
  ∧  properp (b)
  ∧  (length (a) = tree-size (tree))
  ∧  (length (b) = tree-size (tree)))
 →  (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 ≃ nil
    then list (list ('leaf,
                     '(out),
                     if parity then 'b-not
                     else 'b-buf endif,
                     list (index ('a, 0))))
    elseif (car (tree) ≃ nil) ∧ (cdr (tree) ≃ 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*),

```
                                    ¬ parity,
                                    if ¬ 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 ≃ nil
    then if parity then f-not (car (a))
         else f-buf (car (a)) endif
    elseif (car (tree) ≃ nil) ∧ (cdr (tree) ≃ 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), ¬ parity, car (tree)),
                tr-or-nor (trestn (a, tree), ¬ parity, cdr (tree)))
    else f-nand (tr-or-nor (tfirstn (a, tree), ¬ parity, car (tree)),
                 tr-or-nor (trestn (a, tree), ¬ 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)
  ∧  (call-name = if parity then 't-nor
                  else 't-or endif)
  ∧  properp (a)
  ∧  (length (a) = tree-size (tree)))
 →   (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 ≃ nil
    then if parity then b-not (car (a))
         else b-buf (car (a)) endif
    elseif (car (tree) ≃ nil) ∧ (cdr (tree) ≃ 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 (tfirstn (a, *tree*), ¬ *parity*, car (*tree*)),
btr-or-nor (trestn (a, *tree*), ¬ *parity*, cdr (*tree*)))
else b-nand (btr-or-nor (tfirstn (a, *tree*), ¬ *parity*, car (*tree*)),
btr-or-nor (trestn (a, *tree*), ¬ *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) ∧ (length (a) = tree-size (*tree*)))
→ (tr-or-nor (a, *parity*, *tree*) = btr-or-nor (a, *parity*, *tree*))

THEOREM: btr-or-is-v-nzerop
(length (a) = tree-size (*tree*))
→ (btr-or-nor (a, *parity*, *tree*)
= **if** *parity* **then** v-zerop (a)
else v-nzerop (a) **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*)))
∧ (cadr (tv-zerop* (*tree*)) = indices (`'in`, `0`, tree-size (*tree*)))
∧ (caddr (tv-zerop* (*tree*)) = `'(out)`)

∧ (cadddr (tv-zerop* (*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**)
∧ (caddddr (tv-zerop* (*tree*)) = **nil**)

EVENT: Enable tv-zerop*; name this event 'tv-zerop*-off'.

EVENT: Enable tv-zerop*$destructure; name this event 'tv-zerop*$destructure-off'.

DEFINITION:
tv-zerop& (*netlist*, *tree*)
= ((lookup-module (index (`'tv-zerop`, tree-number (*tree*)), *netlist*)
  = tv-zerop* (*tree*))
 ∧ t-or-nor& (delete-module (index (`'tv-zerop`, tree-number (*tree*)),
   netlist),
   tree,
   (tree-height (*tree*) **mod** 2) ≠ `1`)
 ∧ b-not& (delete-module (index (`'tv-zerop`, tree-number (*tree*)),
   netlist)))

EVENT: Enable tv-zerop&; name this event 'tv-zerop&-off'.

DEFINITION:
tv-zerop$netlist (*tree*)
= cons (tv-zerop* (*tree*),
  t-or-nor$netlist (*tree*, (tree-height (*tree*) **mod** 2) ≠ `1`)
  ∪ B-NOT$NETLIST)

DEFINITION:
f$tv-zerop (*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$tv-zerop; name this event 'f$tv-zerop-off'.

THEOREM: tv-zerop$value
(tv-zerop& (*netlist*, *tree*) ∧ (length (*a*) = tree-size (*tree*)) ∧ properp (*a*))
→ (dual-eval (0, index ('**tv-zerop**, tree-number (*tree*)), *a*, *state*, *netlist*)
  = list (f$tv-zerop (*a*, *tree*)))

EVENT: Enable tv-zerop$value; name this event 'tv-zerop$value-off'.

THEOREM: f$tv-zerop=v-zerop
((length (*a*) = tree-size (*tree*)) ∧ bvp (*a*))
→ (f$tv-zerop (*a*, *tree*) = v-zerop (*a*))

DEFINITION:
f$fast-zero (*v*)
= f-nor3 (tr-or-nor (firstn ((length (*v*) − 1) − 1, *v*),
  f,
  make-tree ((length (*v*) − 1) − 1)),
  nth ((length (*v*) − 1) − 1, *v*),
  nth (length (*v*) − 1, *v*))

EVENT: Enable f$fast-zero; name this event 'f$fast-zero-off'.

THEOREM: f$fast-zero=tr-or-nor
(properp (*v*)
∧ **if** length (*v*) < 3 **then f**
else t endif)
→ (f$fast-zero (*v*)
  = tr-or-nor (*v*,
  t,
  cons (make-tree ((length (*v*) − 1) − 1), '(0 . 0))))

THEOREM: f$fast-zero=v-zerop
(bvp (*v*)
∧ **if** length (*v*) < 3 **then f**
else t endif)
→ (f$fast-zero (*v*) = v-zerop (*v*))

DEFINITION:
fast-zero* (*n*)
= cons (index ('**fast-zero**, *n*),
  cons (indices ('**a**, 0, *n*),
  cons ('**(z)**,
  cons (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)))),
              '(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-zerop`,
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-zerop`, tree-number (make-tree (n))),

indices ('x, 0, n))))
$\wedge$ (caddddr (v-equal* (n)) = **nil**)

EVENT: Enable v-equal*; name this event 'v-equal*-off'.

EVENT: Enable v-equal*$destructure; name this event 'v-equal*$destructure-off'.

DEFINITION:
v-equal& (*netlist*, n)
= ((lookup-module (index ('v-equal, n), *netlist*) = v-equal* (n))
  $\wedge$ v-xor& (delete-module (index ('v-equal, n), *netlist*), n)
  $\wedge$ tv-zerop& (delete-module (index ('v-equal, n), *netlist*),
   make-tree (n)))

EVENT: Enable v-equal&; name this event 'v-equal&-off'.

DEFINITION:
v-equal$netlist ($n$)
= cons (v-equal* (n), v-xor$netlist ($n$) $\cup$ tv-zerop$netlist (make-tree ($n$)))

DEFINITION:
f$v-equal ($a$, b) = f$tv-zerop (fv-xor (a, b), make-tree (length (a)))

EVENT: Enable f$v-equal; name this event 'f$v-equal-off'.

THEOREM: v-equal$value
(v-equal& (*netlist*, n)
$\wedge$ ($n \not\simeq$ 0)
$\wedge$ properp (a)
$\wedge$ properp (b)
$\wedge$ (length (a) = n)
$\wedge$ (length (b) = n))
$\rightarrow$ (dual-eval (0, index ('v-equal, n), append (a, b), *state*, *netlist*)
  = list (f$v-equal ($a$, b)))

EVENT: Enable v-equal$value; name this event 'v-equal$value-off'.

THEOREM: f$v-equal=equal*
((length (a) $\not\simeq$ 0) $\wedge$ bvp (a) $\wedge$ bvp (b) $\wedge$ (length (a) = length (b)))
$\rightarrow$ (f$v-equal ($a$, b) = (a = b))

Definition:
v-inc4 (*a0*, *a1*, *a2*, *a3*)
= list (b-not (*a0*),
b-xor (b-not (*a1*), b-not (*a0*)),
b-equv (b-not (*a2*), b-nor (b-not (*a0*), b-not (*a1*))),
b-equv (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-equv (f-not (*a2*), f-nor (f-not (*a0*), f-not (*a1*))),
f-equv (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-equv (w-9 w-13))
     (g-7 (w-7) b-nor (w-10 w-11))
     (g-8 (w-8) b-equv (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-equv& (delete-module ('v-inc4, *netlist*))
∧ b-xor& (delete-module ('v-inc4, *netlist*)))

Event: Enable v-inc4&; name this event 'v-inc4&-off'.

Definition:

V-INC4$NETLIST
= cons (V-INC4*,
    (((B-NOR3$NETLIST ∪ B-NOT$NETLIST) ∪ B-NOR$NETLIST)
     ∪ B-EQUV$NETLIST)
    ∪ B-XOR$NETLIST)

THEOREM: v-inc4$value
v-inc4& (*netlist*)
→ (dual-eval (0, 'v-inc4, list (*a0*, *a1*, *a2*, *a3*), *state*, *netlist*)
    = f$v-inc4 (*a0*, *a1*, *a2*, *a3*))

EVENT: Enable v-inc4$value; name this event 'v-inc4$value-off'.

THEOREM: f$v-inc4=v-inc4
(boolp (*a0*) ∧ boolp (*a1*) ∧ boolp (*a2*) ∧ boolp (*a3*))
→ (f$v-inc4 (*a0*, *a1*, *a2*, *a3*) = v-inc4 (*a0*, *a1*, *a2*, *a3*))

DEFINITION:
f$v-inc4$v (*a*)
= list (f-not (car (*a*)),
    f-xor (f-not (cadr (*a*)), f-not (car (*a*))),
    f-equv (f-not (caddr (*a*)), f-nor (f-not (car (*a*)), f-not (cadr (*a*)))),
    f-equv (f-not (cadddr (*a*)),
        f-nor3 (f-not (car (*a*)), f-not (cadr (*a*)), f-not (caddr (*a*)))))

EVENT: Enable f$v-inc4$v; name this event 'f$v-inc4$v-off'.

THEOREM: properp-length-f$v-inc4$v
properp (f$v-inc4$v (*a*)) ∧ (length (f$v-inc4$v (*a*)) = 4)

THEOREM: v-inc4$value-as-v-inc
(v-inc4& (*netlist*) ∧ properp (*a*) ∧ (length (*a*) = 4))
→ (dual-eval (0, 'v-inc4, *a*, *state*, *netlist*) = f$v-inc4$v (*a*))

THEOREM: f$v-inc4$v=v-inc
(bvp (*a*) ∧ (length (*a*) = 4)) → (f$v-inc4$v (*a*) = v-inc (*a*))

DEFINITION:
tv-dec-pass (*c*, *a*, *tree*)
= **if** *tree* ≃ **nil** **then** list (b-buf (car (*a*)), b-equv (car (*a*), *c*))
  else cons (b-or (car (tv-dec-pass (*c*, tfirstn (*a*, *tree*), car (*tree*))),
        car (tv-dec-pass (b-or (*c*,
              car (tv-dec-pass (*c*,
                  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
length (cdr (tv-dec-pass (c, a, tree))) = tree-size (tree)

THEOREM: tv-dec-pass-length
length (tv-dec-pass (c, a, tree)) = (1 + tree-size (tree))

THEOREM: bvp-cdr-tv-dec-pass
bvp (cdr (tv-dec-pass (c, a, tree)))

THEOREM: bvp-tv-dec-pass
bvp (tv-dec-pass (c, a, tree))

THEOREM: bvp-length-tv-dec-pass
bvp-length (tv-dec-pass (c, a, tree), n)
= **if** (1 + tree-size (tree)) < n **then f**
else t endif

THEOREM: tv-dec-pass-crock-1
((length (a) = tree-size (tree)) ∧ bvp (a))
→ (tv-adder (c, v-not (nat-to-v (0, length (a))), a, tree)
= cons (**t**, tv-dec-pass (c, a, tree)))

THEOREM: tv-dec-pass-crock-2
((length (a) = tree-size (tree)) ∧ bvp (a))
→ (cdr (tv-dec-pass (c, a, tree))
= cddr (tv-adder (c, v-not (nat-to-v (0, length (a))), a, tree)))

THEOREM: tv-dec-pass-works
((length (a) = tree-size (tree)) ∧ bvp (a))
→ (cdr (tv-dec-pass (c, a, tree))
= **if** c **then** v-buf (a)
else v-dec (a) **endif**)

Definition:

```
tv-dec-pass-ng (c, a, tree, make-g)
=   if tree ≃ nil
    then if make-g  then list (b-buf (car (a)), b-equv (car (a), c))
           else list (b-equv (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 $\rightarrow$ (length (cdr (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*))) = tree-size (*tree*))

THEOREM: bvp-tv-dec-pass-ng
bvp (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*))

THEOREM: bvp-cdr-tv-dec-pass-ng
make-g $\rightarrow$ bvp (cdr (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*)))

THEOREM: bvp-length-tv-dec-pass-ng
bvp-length (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*), *n*)
= **if** *make-g*
  then if (1 + tree-size (*tree*)) < *n* **then f**
    else t endif
  else *n* < (1 + tree-size (*tree*)) **endif**

THEOREM: tv-dec-pass-ng-is-cdr-tv-dec-pass
tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*)
= **if** *make-g* **then** tv-dec-pass (*c*, *a*, *tree*)
  else cdr (tv-dec-pass (*c*, *a*, *tree*)) **endif**

THEOREM: boolp-car-tv-dec-pass-ng
boolp (car (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*)))

THEOREM: tv-dec-pass-ng-works-1
(bvp (*a*) $\wedge$ (length (*a*) = tree-size (*tree*)) $\wedge$ *make-g*)
$\rightarrow$ (cdr (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*))
  = **if** *c* **then** v-buf (*a*)
    else v-dec (*a*) **endif**)

THEOREM: tv-dec-pass-ng-works-2
(bvp (*a*) $\wedge$ (length (*a*) = tree-size (*tree*)) $\wedge$ ($\neg$ *make-g*))
$\rightarrow$ (tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*)
  = **if** *c* **then** v-buf (*a*)
    else v-dec (*a*) **endif**)

DEFINITION:
f$tv-dec-pass-ng (*c*, *a*, *tree*, *make-g*)
= **if** *tree* $\simeq$ **nil**
  then if *make-g* **then** list (car (*a*), f-equv (car (*a*), *c*))
    else list (f-equv (car (*a*), *c*)) **endif**
  elseif *make-g*
  then cons (f-or (car (f$tv-dec-pass-ng (*c*, tfirstn (*a*, *tree*), car (*tree*), **t**)),
      car (f$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))),
     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)`
`((g0 (g) id (a)) (g1 (z) b-equv (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-equv& (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-equv (*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 ≃ 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*))
∧ (cadr (tv-dec-pass-ng* (*tree*, *make-g*))
= cons (`'c`, indices (`'a`, `0`, tree-size (*tree*))))
∧ (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**)
∧ (cadddr (tv-dec-pass-ng* (*tree*, *make-g*))
= tv-dec-pass-ng-body (*tree*, *make-g*))
∧ (caddddr (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* ≃ **nil**
then (lookup-module (tv-dec-pass-name (*tree*, *make-g*), *netlist*)
= tv-dec-pass-ng* (*tree*, *make-g*))
∧ 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*))
∧ tv-dec-pass-ng& (delete-module (tv-dec-pass-name (*tree*,
make-g),
netlist),
car (*tree*),
t)
∧ tv-dec-pass-ng& (delete-module (tv-dec-pass-name (*tree*,
make-g),
netlist),
cdr (*tree*),
make-g)
∧ 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 ≃ 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

(tv-dec-pass-ng& (delete-module (index (`'tv-dec-pass-g`, tree-number (*tree*)),
netlist),
car (*tree*),
make-g)
→ (collect-value (indices (`'z`, `0`, *n*), *bindings*)
= append (collect-value (firstn (tree-size (car (*tree*)),
indices (`'z`, `0`, *n*)),
bindings),
collect-value (restn (tree-size (car (*tree*)),
indices (`'z`, `0`, *n*)),
bindings))))
∧ (tv-dec-pass-ng& (delete-module (index (`'tv-dec-pass-ng`,
tree-number (*tree*)),
netlist),
car (*tree*),
make-g)
→ (collect-value (indices (`'z`, `0`, *n*), *bindings*)
= append (collect-value (firstn (tree-size (car (*tree*)),
indices (`'z`, `0`, *n*)),
bindings),
collect-value (restn (tree-size (car (*tree*)),
indices (`'z`, `0`, *n*)),
bindings))))

Event: Enable tv-dec-pass-ng-lemma-crock; name this event 'tv-dec-pass-ng-lemma-crock-off'.

Theorem: tv-dec-pass-ng$value
(tv-dec-pass-ng& (*netlist*, *tree*, *make-g*)
∧ (length (a) = tree-size (*tree*))
∧ properp (a)
∧ boolp (*make-g*)
∧ (*name* = tv-dec-pass-name (*tree*, *make-g*)))
→ (dual-eval (`0`, *name*, cons (c, a), *state*, *netlist*)
= f$tv-dec-pass-ng ($c$, a, *tree*, *make-g*))

Event: Enable tv-dec-pass-ng$value; name this event 'tv-dec-pass-ng$value-off'.

Definition:
tv-dec-pass-ng$netlist (*tree*, *make-g*)
= **if** *tree* ≃ **nil** **then** list (tv-dec-pass-ng* (*tree*, *make-g*), DEC-PASS-CELL*)
else cons (tv-dec-pass-ng* (*tree*, *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
boolp (*make-g*)
→ tv-dec-pass-ng& (tv-dec-pass-ng\$netlist (`'(0 0 . 0)`, *make-g*),
`'(0 0 . 0)`,
make-g)

DEFINITION:
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
(car (dec-pass* (*n*)) = index (`'dec-pass`, *n*))
∧ (cadr (dec-pass* (*n*)) = cons (`'c`, indices (`'a`, `0`, *n*)))
∧ (caddr (dec-pass* (*n*)) = indices (`'z`, `0`, *n*))
∧ (cadddr (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*)))))
∧ (caddddr (dec-pass* (*n*)) = **nil**)

EVENT: Enable dec-pass*; name this event 'dec-pass*-off'.

EVENT: Enable dec-pass*\$destructure; name this event 'dec-pass*\$destructure-off'.

DEFINITION:
dec-pass& (*netlist*, *n*)
= ((lookup-module (index (`'dec-pass`, *n*), *netlist*) = dec-pass* (*n*))
∧ b-not& (delete-module (index (`'dec-pass`, *n*), *netlist*))

$\wedge$ tv-dec-pass-ng& (delete-module (index ('`dec-pass`, n), *netlist*),
make-tree (n),
f))

EVENT: Enable dec-pass&; name this event 'dec-pass&-off'.

DEFINITION:
f\$dec-pass (c, a) = f\$tv-dec-pass-ng (f-not (c), a, make-tree (length (a)), **f**)

EVENT: Enable f\$dec-pass; name this event 'f\$dec-pass-off'.

THEOREM: properp-length-f\$dec-pass
properp (f\$dec-pass (c, v))
$\wedge$ ((length (v) $\neq$ `0`) $\rightarrow$ (length (f\$dec-pass (c, v)) = length (v)))

THEOREM: f\$dec-pass=dec-or-pass
(bvp (v) $\wedge$ boolp (*dec*) $\wedge$ (length (v) $\neq$ `0`))
$\rightarrow$ (f\$dec-pass (*dec*, v) = if* (*dec*, v-dec (v), v))

THEOREM: dec-pass\$value
(dec-pass& (*netlist*, n) $\wedge$ ($n \not\simeq$ `0`) $\wedge$ (length (a) = n) $\wedge$ properp (a))
$\rightarrow$ (dual-eval (`0`, index ('`dec-pass`, n), cons (c, a), *state*, *netlist*)
= f\$dec-pass (c, a))

EVENT: Enable dec-pass\$value; name this event 'dec-pass\$value-off'.

DEFINITION:
dec-pass\$netlist (n)
= cons (dec-pass* (n),
B-NOT\$NETLIST $\cup$ tv-dec-pass-ng\$netlist (make-tree (n), **f**))

THEOREM: check-dec-pass\$netlist
dec-pass& (dec-pass\$netlist (`7`), `7`)

DEFINITION:
reg-body (m, n, *ti*, *te*)
= **if** $n \simeq$ `0` **then nil**
else cons (list (index ('`g`, m),
list (index ('`q`, m), index ('`qb`, m)),
'`fd1s`,
list (index ('`d`, m), '`clk`, *ti*, *te*)),
reg-body ($1 + m$, $n - 1$, index ('`q`, m), *te*)) **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))
∧ (cadddr (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))))
∧ (caddddr (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** $n \simeq 0$ **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*)

$\rightarrow$ (collect-value (indices ('**q**, m, n),
dual-eval (**1**,
reg-body (m, n, ti, te),
bindings,
state-bindings,
netlist))
= v-threefix (collect-value (indices ('**g**, m, n), *state-bindings*)))

THEOREM: reg$value
(reg& (*netlist*, n) $\wedge$ (length (*state*) = n) $\wedge$ properp (*state*))
$\rightarrow$ (dual-eval (**0**, index ('**reg**, n), *inputs*, *state*, *netlist*)
= v-threefix (*state*))

EVENT: Enable reg$value; name this event 'reg$value-off'.

THEOREM: reg-body$state
(fd1s& (*netlist*) $\wedge$ ($\neg$ value (te, *bindings*)))
$\rightarrow$ (collect-value (indices ('**g**, m, n),
dual-eval (**3**,
reg-body (m, n, ti, te),
bindings,
state-bindings,
netlist))
= v-threefix (collect-value (indices ('**d**, m, n), *bindings*)))

THEOREM: reg$state
(reg& (*netlist*, n) $\wedge$ (length (d) = n) $\wedge$ properp (d) $\wedge$ ($\neg$ te))
$\rightarrow$ (dual-eval (**2**,
index ('**reg**, n),
cons (clk, cons (te, cons (ti, d))),
state,
netlist)
= v-threefix (d))

EVENT: Enable reg$state; name this event 'reg$state-off'.

DEFINITION:
we-reg-body (m, n, ti)
= **if** $n \simeq$ **0** **then nil**
else cons (list (index ('**g**, m),
list (index ('**q**, m), index ('**qb**, m)),
'**fd1slp**,
cons (index ('**d**, m),
cons ('**clk**,

```
                        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:
we-reg$netlist (*n*)
= cons (we-reg* (*n*),
    if $n < 8$ **then** B-BUF$NETLIST
    else B-BUF-PWR$NETLIST **endif**
    ∪ (DEL4$NETLIST ∪ FD1SLP$NETLIST))

THEOREM: we-reg-body$unbound-in-body
unbound-in-body (`'clk`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'we`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'we-buf`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'ti`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'ti-buf`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'te`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (`'te-buf`, we-reg-body (*m*, *n*, *ti*))
∧ unbound-in-body (index (`'d`, *l*), we-reg-body (*m*, *n*, *ti*))
∧ (($l < m$) → unbound-in-body (index (`'q`, *l*), we-reg-body (*m*, *n*, *ti*)))

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:
we-reg-body-induction (*m*, *n*, *ti*, *bindings*, *state-bindings*, *netlist*)

```
=  if n ≃ 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) ∧ (length (state) = n) ∧ 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) ∧ (¬ 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$, n)
∧ (length (d) = n)
∧ properp (d)
∧ (length ($state$) = n)
∧ properp ($state$)
∧ (¬ te))
→ (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 ($cvzbv$) = cdddr ($cvzbv$)

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
bvp (bv (v)) $\rightarrow$ boolp (n (v))

THEOREM: boolp-bvp-cvzbv
(boolp (c (cvzbv (c, v, bv))) = boolp (c))
$\land$ (boolp (v (cvzbv (c, v, bv))) = boolp (v))
$\land$ boolp (zb (cvzbv (c, v, bv)))
$\land$ (bvp (bv (cvzbv (c, v, bv))) = bvp (bv))

THEOREM: bvp-cvzbv
(boolp (c ($cvzbv$)) $\land$ boolp (v ($cvzbv$)) $\land$ boolp (zb ($cvzbv$)) $\land$ bvp (bv ($cvzbv$)))
$\rightarrow$ bvp ($cvzbv$)

DEFINITION:
cvzbv-v-ror (c, a)
= cvzbv (**if** $a \simeq$ **nil then** c
else nth (0, a) **endif**,
f,
v-ror (a, c))

DEFINITION:
cvzbv-v-adder (c, a, b)
= cvzbv (v-adder-carry-out (c, a, b),
v-adder-overflowp (c, a, b),
v-adder-output (c, a, b))

DEFINITION: cvzbv-v-lsl (a) = cvzbv-v-adder (**f**, a, a)

DEFINITION:
cvzbv-v-subtracter (c, a, b)
= cvzbv (v-subtracter-carry-out (c, a, b),
v-subtracter-overflowp (c, a, b),
v-subtracter-output (c, a, b))

DEFINITION:
cvzbv-inc (a) = cvzbv-v-adder (**t**, a, nat-to-v (0, length (a)))

DEFINITION:
cvzbv-neg (a) = cvzbv-v-subtracter (**f**, a, nat-to-v (0, length (a)))

DEFINITION:
cvzbv-dec (a) = cvzbv-v-subtracter (**t**, nat-to-v (0, length (a)), a)

DEFINITION: cvzbv-v-not (a) = cvzbv (**f**, **f**, 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-subtracter (c, a, b)
elseif op = list (**t**, **t**, **t**, **f**) **then** cvzbv-v-subtracter (**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
(boolp (c) $\land$ bvp (a) $\land$ (length $(a) \neq 0$)) $\rightarrow$ boolp (c (v-alu (c, a, b, op)))

THEOREM: boolp-v-v-alu
(boolp (c) $\land$ bvp (a) $\land$ (length $(a) \neq 0$)) $\rightarrow$ boolp (v (v-alu (c, a, b, op)))

THEOREM: boolp-zb-v-alu
boolp (zb (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-subtracter
$\text{length}(\text{cvzbv-v-subtracter}(c,\ a,\ b)) = (1 + (1 + (1 + \text{length}(a))))$

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
$(\text{length}(\text{bv}(x)) = (\text{length}(x) - 3))$
$\wedge\quad ((\text{bvp}(x)$
$\qquad \wedge\quad$ **if** $\text{length}(x) < 3$ **then f**
$\qquad\qquad$ **else t endif**)
$\qquad \rightarrow\quad \text{bvp}(\text{bv}(x)))$

DEFINITION:
unary-op-code-p (*op-code*)
$= \quad ((\mathit{op\text{-}code} = \text{list}(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{f}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{t},\ \mathbf{f},\ \mathbf{f},\ \mathbf{f}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{f}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{t},\ \mathbf{f},\ \mathbf{t},\ \mathbf{f}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{t},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{f},\ \mathbf{t},\ \mathbf{f},\ \mathbf{t}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{f},\ \mathbf{t},\ \mathbf{t},\ \mathbf{t}))$
$\qquad \vee \quad (\mathit{op\text{-}code} = \text{list}(\mathbf{t},\ \mathbf{t},\ \mathbf{t},\ \mathbf{t})))$

EVENT: Enable unary-op-code-p; name this event ‘unary-op-code-p-off’.

DEFINITION: $\text{v-alu-1}(c,\ a,\ \mathit{op\text{-}code}) = \text{v-alu}(c,\ a,\ a,\ \mathit{op\text{-}code})$

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*)
→ (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) ∧ 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) ∧ bvp (ALU-DEC-OP)

THEOREM: bv-v-alu-alu-inc-alu-dec
(bv (v-alu (*c*, *a*, *b*, ALU-INC-OP)) = v-inc (*a*))
∧ (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*)
∧ b-nor3& (delete-module ('`decode-mode`, *netlist*))
∧ 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 ∪ B-NOR$NETLIST)

THEOREM: decode-mode$value
decode-mode& (*netlist*)
→ (dual-eval (`0`, '`decode-mode`, list (*op0*, *op1*, *op2*, *op3*), *state*, *netlist*)
= list (f$decode-mode (*op0*, *op1*, *op2*, *op3*)))

EVENT: Enable decode-mode$value; name this event 'decode-mode$value-off'.

THEOREM: f$decode-mode=decode-mode
(boolp (*op0*) ∧ boolp (*op1*) ∧ boolp (*op2*) ∧ boolp (*op3*))
→ (f$decode-mode (*op0*, *op1*, *op2*, *op3*) = decode-mode (*op0*, *op1*, *op2*, *op3*))

THEOREM: decode-mode$value-zero
decode-mode& (*netlist*)
→ (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-equv (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*))),

```
                    b-nand4 (b-not (b-not (op0)),
                            b-not (op1),
                            b-not (b-not (op2)),
                            b-not (op3))),
            b-not (zero)))
```

DEFINITION:
f$decode-prop (*zero*, *swap*, *op0*, *op1*, *op2*, *op3*)

```
=   list (f-nand3 (f-nand4 (f-not (op0),
                            f-not (op1),
                            f-not (f-not (op2)),
                            f-not (f-not (op3))),
                   f-nand (f-not (f-not (op1)), f-not (op3)),
                   f-nand3 (f-not (op2), f-not (op3), f-not (f-not (swap)))),
         f-nor (f-not (op2),
                f-nor (f-not (op3), f-nor (f-not (f-not (op0)), f-not (op1)))),
         f-and (f-nand3 (f-nand (f-not (f-not (op3)),
                                 f-equv (f-not (f-not (op0)), f-not (f-not (op1)))),
                         f-nand (f-not (op2),
                                 f-nand (f-not (f-not (swap)), f-not (op3))),
                         f-nand4 (f-not (f-not (op0)),
                                  f-not (op1),
                                  f-not (f-not (op2)),
                                  f-not (op3))),
                f-not (zero)))
```

EVENT: Enable f$decode-prop; name this event 'f$decode-prop-off'.

DEFINITION:
DECODE-PROP*

```
=   '(decode-prop
      (zero swap op0 op1 op2 op3)
      (w-16 w-24 w-48)
      ((g-47 (w-47) b-not (zero))
       (g-44 (w-44) b-not (op3))
       (g-42 (w-42) b-not (op2))
       (g-43 (w-43) b-not (w-42))
       (g-41 (w-41) b-not (op1))
       (g-39 (w-39) b-not (op0))
       (g-40 (w-40) b-not (w-39))
       (g-45 (w-45) b-nand4 (w-40 w-41 w-43 w-44))
       (g-34 (w-34) b-not (swap))
       (g-35 (w-35) b-not (w-34))
       (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-equv (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-equv& (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$NETLIST
= cons (DECODE-PROP*,
 (((((B-EQUV$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,

netlist)
= f$decode-prop (*zero*, *swap*, *op0*, *op1*, *op2*, *op3*))

EVENT: Enable decode-prop$value; name this event 'decode-prop$value-off'.

THEOREM: f$decode-prop=decode-prop
(boolp (*zero*)
∧ boolp (*swap*)
∧ boolp (*op0*)
∧ boolp (*op1*)
∧ boolp (*op2*)
∧ boolp (*op3*))
→ (f$decode-prop (*zero*, *swap*, *op0*, *op1*, *op2*, *op3*)
= decode-prop (*zero*, *swap*, *op0*, *op1*, *op2*, *op3*))

THEOREM: decode-prop$value-zero
decode-prop& (*netlist*)
→ (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*),
cadddr (*zsop*),
caddddr (*zsop*),
cadddddr (*zsop*)),
append (f$decode-prop (car (*zsop*),
cadr (*zsop*),
caddr (*zsop*),
cadddr (*zsop*),
caddddr (*zsop*),
cadddddr (*zsop*)),
list (f$decode-mode (caddr (*zsop*),
cadddr (*zsop*),

caddddr ($zsop$),
cadddddr ($zsop$)))))

Event: Enable f$mpg; name this event 'f$mpg-off'.

Definition:
mpg ($zsop$)
= append (decode-gen (car ($zsop$),
cadr ($zsop$),
caddr ($zsop$),
cadddr ($zsop$),
caddddr ($zsop$),
cadddddr ($zsop$)),
append (decode-prop (car ($zsop$),
cadr ($zsop$),
caddr ($zsop$),
cadddr ($zsop$),
caddddr ($zsop$),
cadddddr ($zsop$)),
list (decode-mode (caddr ($zsop$),
cadddr ($zsop$),
caddddr ($zsop$),
cadddddr ($zsop$)))))

Event: Enable mpg; name this event 'mpg-off'.

Theorem: f$mpg=mpg
(bvp ($zsop$) $\land$ (length ($zsop$) = 6)) $\rightarrow$ (f$mpg ($zsop$) = mpg ($zsop$))

Theorem: properp-length-f$mpg
properp (f$mpg ($zsop$)) $\land$ (length (f$mpg ($zsop$)) = 7)

Theorem: length-mpg
length (mpg ($zsop$)) = 7

Theorem: bvp-mpg
bvp (mpg ($zsop$))

Theorem: properp-mpg
properp (mpg ($zsop$))

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**, **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
(mpg& (*netlist*) ∧ (*zsop* = cons (**t**, cons (**f**, ALU-INC-OP))))
→ (dual-eval (0, 'mpg, *zsop*, *state*, *netlist*) = list (**f**, **f**, **f**, **f**, **f**, **f**, **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 (cadddr (*czop*)),
b-not (caddddr (*czop*)),
b-not (cadddddr (*czop*))),
b-nand3 (b-not (caddr (*czop*)),
b-not (cadddr (*czop*)),
b-not (b-not (caddddr (*czop*)))),
b-nand3 (b-not (b-not (caddr (*czop*))),
b-not (b-not (cadddr (*czop*))),
b-not (b-not (caddddr (*czop*))))),
b-nand3 (b-nand (b-not (b-not (cadddddr (*czop*))),
b-not (b-not (car (*czop*)))),
b-nand3 (b-not (caddr (*czop*)),
b-not (caddddr (*czop*)),
b-not (b-not (car (*czop*)))),
b-nand3 (b-not (caddr (*czop*)),
b-not (b-not (caddddr (*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 (cadddr (*czop*)),
f-not (caddddr (*czop*)),
f-not (cadddddr (*czop*))),
f-nand3 (f-not (caddr (*czop*)),
f-not (cadddr (*czop*)),
f-not (f-not (caddddr (*czop*)))),
f-nand3 (f-not (f-not (caddr (*czop*))),
f-not (f-not (cadddr (*czop*))),
f-not (f-not (caddddr (*czop*))))),
f-nand3 (f-nand (f-not (f-not (cadddddr (*czop*))),
f-not (f-not (car (*czop*)))),

f-nand3 (f-not (caddr (*czop*)),
f-not (caddddr (*czop*)),
f-not (f-not (car (*czop*)))),
f-nand3 (f-not (caddr (*czop*)),
f-not (f-not (caddddr (*czop*))),
f-not (car (*czop*)))))

EVENT: Enable f$carry-in-help; name this event 'f$carry-in-help-off'.

THEOREM: f$carry-in-help=carry-in-help
(bvp (*czop*) ∧ (length (*czop*) = 6))
→ (f$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*)
∧ b-nbuf& (delete-module ('carry-in-help, *netlist*))
∧ b-nand& (delete-module ('carry-in-help, *netlist*))
∧ b-nand3& (delete-module ('carry-in-help, *netlist*))
∧ 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 ∪ B-NAND$NETLIST) ∪ B-NAND3$NETLIST)
        ∪ 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$value
carry-in-help& (*netlist*)
→ (dual-eval (0, `'carry-in-help`, *czop*, *state*, *netlist*)
   = list (f$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*) ∧ (*czop* = cons (c, cons (**t**, ALU-INC-OP))))
→ (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*, cadddddr (*mpg*), caddddr (*mpg*), cadddr (*mpg*)),
g-cell (*a*, b-not (*a*), b-not (*b*), caddr (*mpg*), cadr (*mpg*), car (*mpg*)),
b-equv3 (b-nand (*c*, caddddddr (*mpg*)),
p-cell (*a*,
b-not (*a*),
b,
cadddddr (*mpg*),
caddddr (*mpg*),
cadddr (*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*, cadddddr (*mpg*), caddddr (*mpg*), cadddr (*mpg*)),
f$g-cell (*a*, f-not (*a*), f-not (*b*), caddr (*mpg*), cadr (*mpg*), car (*mpg*)),
f-equv3 (f-nand (*c*, caddddddr (*mpg*)),
f$p-cell (*a*,
f-not (*a*),
b,
cadddddr (*mpg*),
caddddr (*mpg*),
cadddr (*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-equv3 (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-equv3& (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$NETLIST)
  ∪ B-EQUV3$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**, **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

```
(length (mpg) = 7)
→   (f$alu-cell (c, a, b, v-threefix (mpg)) = f$alu-cell (c, a, b, mpg))
```

DEFINITION:

```
tv-alu-help (c, a, b, mpg, tree)
=   if tree ≃ nil then alu-cell (c, car (a), car (b), mpg)
    else cons (b-and (car (tv-alu-help (c,
                                        tfirstn (a, tree),
                                        tfirstn (b, tree),
                                        mpg,
                                        car (tree))),
                      car (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)))),
               cons (t-carry (cadr (tv-alu-help (c,
                                                 tfirstn (a, tree),
                                                 tfirstn (b, tree),
                                                 mpg,
                                                 car (tree))),
                              car (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))),
           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 (cddr (tv-alu-help (c,
                          tfirstn (a, tree),
                          tfirstn (b, tree),
                          mpg,
                          car (tree))),
           cddr (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*)))))) **endif**

Event: Enable tv-alu-help; name this event 'tv-alu-help-off'.

Theorem: cddr-tv-alu-help-length
length (cddr (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))) = tree-size (*tree*)

Theorem: tv-alu-help-length
length (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)) = (1 + (1 + tree-size (*tree*)))

Theorem: bvp-cddr-tv-alu-help
bvp (cddr (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)))

Theorem: bvp-tv-alu-help
bvp (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))

Theorem: bvp-length-tv-alu-help
bvp-length (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*), *n*)
= **if** (1 + (1 + tree-size (*tree*))) < *n* **then f**
else t endif

Theorem: tv-alu-help-zero
((*mpg* = list (**f**, **f**, **f**, **f**, **f**, **f**, **t**))
∧ (length (*a*) = tree-size (*tree*))
∧ (*c* = **t**))
→ (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)
= cons (**f**, cons (**t**, make-list (length (*a*), **f**))))

Event: Enable tv-alu-help-zero; name this event 'tv-alu-help-zero-off'.

Theorem: tv-alu-help-v-and-works
(bv2p (*a*, *b*)
∧ (length (*a*) = tree-size (*tree*))
∧ (*mpg* = list (**t**, **t**, **f**, **f**, **f**, **f**, **f**)))
→ (cddr (tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)) = 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
$((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
$\land \quad \text{bv2p}(a, b)$
$\land \quad (\text{length}(a) = \text{tree-size}(tree)))$
$\rightarrow \quad (\text{cddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-or}(a, b))$

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
$((mpg = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
$\land \quad \text{bv2p}(a, b)$
$\land \quad (\text{length}(a) = \text{tree-size}(tree)))$
$\rightarrow \quad (\text{cddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-xor}(a, b))$

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
$((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}))$
$\land \quad \text{bvp}(a)$
$\land \quad (\text{length}(a) = \text{tree-size}(tree)))$
$\rightarrow \quad (\text{cddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-not}(a))$

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
$((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}))$
$\land \quad \text{bvp}(a)$
$\land \quad (\text{length}(a) = \text{tree-size}(tree)))$
$\rightarrow \quad (\text{cddr}(\text{tv-alu-help}(c, a, b, mpg, tree)) = \text{v-buf}(a))$

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
$((mpg = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}))$
$\land \quad \text{bvp}(a)$

$\wedge$ (length (a) = tree-size ($tree$)))
$\rightarrow$ (tv-alu-help (c, a, b, mpg, $tree$) = tv-adder (c, a, b, $tree$))

Event: Enable tv-alu-help-tv-adder-works; name this event 'tv-alu-help-tv-adder-works-off'.

Theorem: tv-alu-help-tv-subtracter-works
((mpg = list (**t**, **f**, **t**, **t**, **f**, **t**))
$\wedge$ bvp (a)
$\wedge$ (length (a) = tree-size ($tree$)))
$\rightarrow$ (tv-alu-help (c, a, b, mpg, $tree$) = tv-adder (c, v-not (a), b, $tree$))

Event: Enable tv-alu-help-tv-subtracter-works; name this event 'tv-alu-help-tv-subtracter-works-off'.

Theorem: tv-alu-help-tv-inc-a-works
((mpg = list (**f**, **t**, **t**, **f**, **f**, **t**, **t**))
$\wedge$ bvp (a)
$\wedge$ (length (a) = tree-size ($tree$)))
$\rightarrow$ (tv-alu-help (c, a, b, mpg, $tree$)
= tv-adder (c, a, nat-to-v (0, length (a)), $tree$))

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
((mpg = list (**f**, **t**, **t**, **t**, **f**, **f**, **t**))
$\wedge$ bvp (b)
$\wedge$ (length (b) = tree-size ($tree$)))
$\rightarrow$ (tv-alu-help (c, a, b, mpg, $tree$)
= tv-adder (c, b, nat-to-v (0, length (b)), $tree$))

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
((mpg = list (**f**, **t**, **f**, **f**, **t**, **t**, **t**))
$\wedge$ bvp (a)
$\wedge$ (length (a) = tree-size ($tree$)))
$\rightarrow$ (tv-alu-help (c, a, b, mpg, $tree$)
= tv-adder (c, v-not (nat-to-v (0, length (a))), a, $tree$))

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
$((mpg = \text{list}(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{t}))$
$\land \quad \text{bvp}(b)$
$\land \quad (\text{length}(b) = \text{tree-size}(tree)))$
$\to \quad (\text{tv-alu-help}(c,\ a,\ b,\ mpg,\ tree)$
$\qquad = \quad \text{tv-adder}(c,\ \text{v-not}(\text{nat-to-v}(0,\ \text{length}(b))),\ b,\ tree))$

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
$((mpg = \text{list}(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}))$
$\land \quad \text{bvp}(a)$
$\land \quad (\text{length}(a) = \text{tree-size}(tree)))$
$\to \quad (\text{tv-alu-help}(c,\ a,\ b,\ mpg,\ tree)$
$\qquad = \quad \text{tv-adder}(c,\ \text{v-not}(a),\ \text{nat-to-v}(0,\ \text{length}(a)),\ tree))$

EVENT: Enable tv-alu-help-tv-neg-works; name this event 'tv-alu-help-tv-neg-works-off'.

DEFINITION:
```
f$tv-alu-help (c, a, b, mpg, tree)
=    if tree ≃ nil then f$alu-cell (c, car (a), car (b), mpg)
     else cons (f-and (car (f$tv-alu-help (c,
                                           tfirstn (a, tree),
                                           tfirstn (b, tree),
                                           mpg,
                                           car (tree))),
                       car (f$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)))),
cons (f$t-carry (cadr (f$tv-alu-help (c,
                                  tfirstn (a, tree),
                                  tfirstn (b, tree),
                                  mpg,
                                  car (tree))),
            car (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))),
            cadr (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)))),
                   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$tv-alu-help-length
length (cddr (f$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))) = tree-size (*tree*)

THEOREM: f$tv-alu-help-length
length (f$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)) = (1 + (1 + tree-size (*tree*)))

THEOREM: properp-cddr-f$tv-alu-help
properp (cddr (f$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*)))

THEOREM: properp-f$tv-alu-help
properp (f$tv-alu-help (*c*, *a*, *b*, *mpg*, *tree*))

THEOREM: boolp-car-f$tv-alu-help
(boolp (*c*)

$\land$ bvp (a)
$\land$ (length (a) = tree-size ($tree$))
$\land$ bvp (b)
$\land$ (length (b) = tree-size ($tree$))
$\land$ bvp (mpg)
$\land$ (length (mpg) = 7))
$\rightarrow$ (boolp (car (f$tv-alu-help ($c$, a, b, mpg, $tree$)))
  $\land$ boolp (cadr (f$tv-alu-help ($c$, a, b, mpg, $tree$))))

THEOREM: bvp-cddr-f$tv-alu-help
(boolp (c)
$\land$ bvp (a)
$\land$ (length (a) = tree-size ($tree$))
$\land$ bvp (b)
$\land$ (length (b) = tree-size ($tree$))
$\land$ bvp (mpg)
$\land$ (length (mpg) = 7))
$\rightarrow$ bvp (cddr (f$tv-alu-help ($c$, a, b, mpg, $tree$)))

THEOREM: bvp-f$tv-alu-help
(boolp (c)
$\land$ bvp (a)
$\land$ (length (a) = tree-size ($tree$))
$\land$ bvp (b)
$\land$ (length (b) = tree-size ($tree$))
$\land$ bvp (mpg)
$\land$ (length (mpg) = 7))
$\rightarrow$ bvp (f$tv-alu-help ($c$, a, b, mpg, $tree$))

THEOREM: f$tv-alu-help=tv-alu-help
(boolp (c)
$\land$ bvp (a)
$\land$ (length (a) = tree-size ($tree$))
$\land$ bvp (b)
$\land$ (length (b) = tree-size ($tree$))
$\land$ bvp (mpg)
$\land$ (length (mpg) = 7))
$\rightarrow$ (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
                    (TRESTN (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*)))))
∧ (cadddr (tv-alu-help* (*tree*)) = tv-alu-help-body (*tree*))
∧ (caddddr (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*))
∧ 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*))
∧ t-carry& (delete-module (index (`'tv-alu-help`,
tree-number (*tree*)),
netlist))
∧ b-and& (delete-module (index (`'tv-alu-help`,
tree-number (*tree*)),
netlist))
∧ v-buf& (delete-module (index (`'tv-alu-help`,
tree-number (*tree*)),
netlist),
`7`)
∧ tv-alu-help& (delete-module (index (`'tv-alu-help`,
tree-number (*tree*)),
netlist),
car (*tree*))
∧ 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$netlist (*tree*)
= **if** *tree* ≃ **nil** **then** cons (tv-alu-help* (*tree*), ALU-CELL$NETLIST)
else cons (tv-alu-help* (*tree*),
tv-alu-help$netlist (car (*tree*))
∪ (tv-alu-help$netlist (cdr (*tree*))
∪ (T-CARRY$NETLIST
∪ (B-AND$NETLIST
∪ v-buf$netlist (`7`))))) **endif**

DEFINITION:
tv-alu-help-induction (*tree*, *c*, *a*, *b*, *mpg*, *state*, *netlist*)
= **if** *tree* ≃ **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))
∧   tv-alu-help-induction (cdr (tree),
                  f$t-carry (c,
                             car (f$tv-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 (f$tv-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*))
$\rightarrow$ (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**)
$\land$ tv-alu-help& (*netlist*, *tree*)
$\land$ (length (*a*) = 1)
$\land$ properp (*a*)
$\land$ (length (*b*) = 1)
$\land$ properp (*b*)
$\land$ (length (*mpg*) = 7)
$\land$ properp (*mpg*))
$\rightarrow$ (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$value-base-case; name this event 'tv-alu-help$value-base-case-off'.

THEOREM: f$tv-alu-help-v-threefix-mpg
(length (*mpg*) = 7)
$\rightarrow$ (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*)
$\land$ properp (*a*)
$\land$ (length (*a*) = tree-size (*tree*))
$\land$ properp (*b*)
$\land$ (length (*b*) = tree-size (*tree*))

```
 ∧   properp (mpg)
 ∧   (length (mpg) = 7))
→   (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$carry-out-help=carry-out-help
(boolp (*a0*)
∧ boolp (*result*)
∧ boolp (*zero*)
∧ boolp (*op0*)
∧ boolp (*op1*)
∧ boolp (*op2*)
∧ boolp (*op3*))
→ (f$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 → ((carry-out-help (*x*, *a*, *zero*, *op0*, *op1*, *op2*, *op3*)
= carry-out-help (**t**, *a*, *zero*, *op0*, *op1*, *op2*, *op3*))
= **t**)

THEOREM: carry-out-help$value-zero
carry-out-help& (*netlist*)
→ (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*)

$\wedge$ boolp (*bn*)
$\wedge$ boolp (*zero*)
$\wedge$ boolp (*op0*)
$\wedge$ boolp (*op1*)
$\wedge$ boolp (*op2*)
$\wedge$ boolp (*op3*))
$\rightarrow$ (f$overflow-help (*rn*, *an*, *bn*, *zero*, *op0*, *op1*, *op2*, *op3*)
= overflow-help (*rn*, *an*, *bn*, *zero*, *op0*, *op1*, *op2*, *op3*))

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-cntl (*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-cntl (*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,

netlist)
    = f$shift-or-buf-cntl (*c*, *an*, *zero*, *op0*, *op1*, *op2*, *op3*))

EVENT: Enable shift-or-buf-cntl$value; name this event 'shift-or-buf-cntl$value-off'.

THEOREM: f$shift-or-buf-cntl=shift-or-buf-cntl
(boolp (*c*)
∧ boolp (*an*)
∧ boolp (*zero*)
∧ boolp (*op0*)
∧ boolp (*op1*)
∧ boolp (*op2*)
∧ boolp (*op3*))
→ (f$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*)
→ (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,
v-shift-right (a,
cadr (shift-or-buf-cntl (c, an, $zero$, $op0$, $op1$, $op2$, $op3$))))

Event: Enable shift-or-buf; name this event 'shift-or-buf-off'.

Theorem: properp-length-f$shift-or-buf
properp (f$shift-or-buf ($c$, a, an, $zero$, $op0$, $op1$, $op2$, $op3$))
$\wedge$ (length (f$shift-or-buf ($c$, a, an, $zero$, $op0$, $op1$, $op2$, $op3$)) = length (a))

Theorem: length-shift-or-buf
length (shift-or-buf (c, a, an, $zero$, $op1$, $op2$, $op3$, $op4$)) = length (a)

Theorem: bvp-shift-or-buf
bvp (shift-or-buf (c, a, an, $zero$, $op1$, $op2$, $op3$, $op4$))

Theorem: f$shift-or-buf=shift-or-buf
(boolp (c)
$\wedge$ boolp (an)
$\wedge$ boolp ($zero$)
$\wedge$ boolp ($op1$)
$\wedge$ boolp ($op2$)
$\wedge$ boolp ($op3$)
$\wedge$ boolp ($op4$)
$\wedge$ bvp (a))
$\rightarrow$ (f$shift-or-buf ($c$, a, an, $zero$, $op1$, $op2$, $op3$, $op4$)
= shift-or-buf (c, a, an, $zero$, $op1$, $op2$, $op3$, $op4$))

Theorem: shift-or-buf-is-buf
(bvp (a) $\wedge$ (($op0$ $\wedge$ $op1$) $\vee$ $op2$ $\vee$ ($\neg$ $op3$) $\vee$ $zero$))
$\rightarrow$ (shift-or-buf (c, a, an, $zero$, $op0$, $op1$, $op2$, $op3$) = a)

Theorem: shift-or-buf-is-asr
(bvp (a)
$\wedge$ (an = nth (length (a) $-$ 1, a))
$\wedge$ $op0$
$\wedge$ ($\neg$ $op1$)
$\wedge$ ($\neg$ $op2$)
$\wedge$ $op3$
$\wedge$ ($\neg$ $zero$))
$\rightarrow$ (shift-or-buf (c, a, an, $zero$, $op0$, $op1$, $op2$, $op3$) = v-asr (a))

Theorem: shift-or-buf-is-ror
(bvp (a)
$\wedge$ boolp (c)

```
 ∧   (¬ op0)
 ∧   (¬ op1)
 ∧   (¬ op2)
 ∧   op3
 ∧   (¬ zero))
 →   (shift-or-buf (c, a, an, zero, op0, op1, op2, op3) = v-ror (a, c))
```

THEOREM: shift-or-buf-is-lsr

```
 (bvp (a) ∧ (¬ op0) ∧ op1 ∧ (¬ op2) ∧ op3 ∧ (¬ zero))
 →   (shift-or-buf (c, a, an, zero, op0, op1, op2, op3) = v-lsr (a))
```

DEFINITION:

```
tv-shift-or-buf* (tree)
=   cons (index ('tv-shift-or-buf, tree-number (tree)),
          cons (cons ('c,
                      append (indices ('a, 0, tree-size (tree)),
                              '(an zero op0 op1 op2 op3))),
                cons (indices ('out, 0, tree-size (tree)),
                      cons (list ('(cntl
                                    (ctl si)
                                    shift-or-buf-cntl
                                    (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))))),
                            '(nil)))))
```

THEOREM: tv-shift-or-buf*$destructure

```
 (car (tv-shift-or-buf* (tree))
  =   index ('tv-shift-or-buf, tree-number (tree)))
 ∧   (cadr (tv-shift-or-buf* (tree))
      =   cons ('c,
                append (indices ('a, 0, tree-size (tree)),
                        '(an zero op0 op1 op2 op3))))
 ∧   (caddr (tv-shift-or-buf* (tree)) = indices ('out, 0, tree-size (tree)))
 ∧   (cadddr (tv-shift-or-buf* (tree))
      =   list ('(cntl
```

```
                    (ctl si)
                    shift-or-buf-cntl
                    (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))))))
∧   (caddddr (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))
     ∧   shift-or-buf-cntl& (delete-module (index ('tv-shift-or-buf,
                                                  tree-number (tree)),
                                           netlist))
     ∧   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) ∪ SHIFT-OR-BUF-CNTL$NETLIST)
```

THEOREM: tv-shift-or-buf$value

```
(tv-shift-or-buf& (netlist, tree)
 ∧   (length (a) = tree-size (tree))
 ∧   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*)
$\land$ (length (*a*) = tree-size (*tree*))
$\land$ 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,
      cddr (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*),

```
                        cadddr (op)),
             cons (f$fast-zero (f$shift-or-buf (c,
                                                cddr (f$tv-alu-help (c,
                                                                     a,
                                                                     b,
                                                                     mpg,
                                                                     tree)),
                                                nth (length (a) - 1, a),
                                                zero,
                                                car (op),
                                                cadr (op),
                                                caddr (op),
                                                cadddr (op))),
                 f$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),
                                 cadddr (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),
                          cadddr (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),
                               cadddr (op)),
```

```
cons (v-zerop (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),
                              cadddr (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),
                    cadddr (op)))))
```

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

THEOREM: properp-length-f$core-alu

```
 properp (f$core-alu (c, a, b, zero, mpg, op, tree))
 ∧   (length (f$core-alu (c, a, b, zero, mpg, op, tree))
       =   (1 + (1 + (1 + tree-size (tree)))))
```

THEOREM: f$core-alu=core-alu

```
 (boolp (c)
  ∧   bvp (a)
  ∧   (length (a) = tree-size (tree))
  ∧   bvp (b)
  ∧   (length (b) = tree-size (tree))
  ∧   if length (a) < 3 then f
      else t endif
  ∧   boolp (zero)
  ∧   bvp (mpg)
  ∧   (length (mpg) = 7)
  ∧   bvp (op)
  ∧   (length (op) = 4))
 →   (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

boolp (c (core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*)))
$\land$ boolp (v (core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*)))
$\land$ boolp (zb (core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*)))
$\land$ bvp (bv (core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*)))
$\land$ bvp (core-alu (*c*, *a*, *b*, *zero*, *mpg*, *op*, *tree*))

THEOREM: core-alu-works-for-all-normal-cases$crock

(bvp (*a*) $\land$ (length (*a*) = tree-size (*tree*)) $\land$ nth (0, *a*))
$\rightarrow$ (nth (0, *a*) = **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

((bv2p (*a*, *b*) $\land$ (length (*a*) = tree-size (*tree*)) $\land$ boolp (*c*))
$\rightarrow$ (v-alu (*c*, *a*, *b*, list (**f**, **f**, **f**, **f**))
= core-alu (carry-in-help (list (*c*, **f**, **f**, **f**, **f**, **f**)),
a,
b,
f,
mpg (list (**f**, **f**, **f**, **f**, **f**, **f**)),
list (**f**, **f**, **f**, **f**),
tree)))
$\land$ ((bv2p (*a*, *b*) $\land$ (length (*a*) = tree-size (*tree*)) $\land$ boolp (*c*))
$\rightarrow$ (v-alu (*c*, *a*, *b*, list (**t**, **f**, **f**, **f**))
= core-alu (carry-in-help (list (*c*, **f**, **t**, **f**, **f**, **f**)),
a,
b,
f,
mpg (list (**f**, **f**, **t**, **f**, **f**, **f**)),
list (**t**, **f**, **f**, **f**),
tree)))
$\land$ ((bv2p (*a*, *b*) $\land$ (length (*a*) = tree-size (*tree*)) $\land$ boolp (*c*))
$\rightarrow$ (v-alu (*c*, *a*, *b*, list (**f**, **t**, **f**, **f**))
= core-alu (carry-in-help (list (*c*, **f**, **f**, **t**, **f**, **f**)),
a,
b,
f,
mpg (list (**f**, **f**, **f**, **t**, **f**, **f**)),
list (**f**, **t**, **f**, **f**),
tree)))
$\land$ ((bv2p (*a*, *b*) $\land$ (length (*a*) = tree-size (*tree*)) $\land$ boolp (*c*))
$\rightarrow$ (v-alu (*c*, *a*, *b*, list (**t**, **t**, **f**, **f**))

$$
\begin{array}{lll}
& = & \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 \mathit{tree}))) \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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 \mathit{tree}))) \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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 \mathit{tree}))) \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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 \mathit{tree}))) \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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 \mathit{tree}))) \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{tree})) \wedge \text{boolp}\,(c)) &
\end{array}
$$

$$
\begin{array}{lll}
\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})), & \\
& \qquad\qquad a, & \\
& \qquad\qquad b, & \\
& \qquad\qquad \mathbf{f}, & \\
& \qquad\qquad \text{mpg}\,(\text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t})), & \\
& \qquad\qquad \text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t}), & \\
& \qquad\qquad \mathit{tree}))) & \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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})), & \\
& \qquad\qquad a, & \\
& \qquad\qquad b, & \\
& \qquad\qquad \mathbf{f}, & \\
& \qquad\qquad \text{mpg}\,(\text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t})), & \\
& \qquad\qquad \text{list}\,(\mathbf{t},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t}), & \\
& \qquad\qquad \mathit{tree}))) & \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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})), & \\
& \qquad\qquad a, & \\
& \qquad\qquad b, & \\
& \qquad\qquad \mathbf{f}, & \\
& \qquad\qquad \text{mpg}\,(\text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{f},\ \mathbf{t})), & \\
& \qquad\qquad \text{list}\,(\mathbf{f},\ \mathbf{t},\ \mathbf{f},\ \mathbf{t}), & \\
& \qquad\qquad \mathit{tree}))) & \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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})), & \\
& \qquad\qquad a, & \\
& \qquad\qquad b, & \\
& \qquad\qquad \mathbf{f}, & \\
& \qquad\qquad \text{mpg}\,(\text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{t},\ \mathbf{f},\ \mathbf{t})), & \\
& \qquad\qquad \text{list}\,(\mathbf{t},\ \mathbf{t},\ \mathbf{f},\ \mathbf{t}), & \\
& \qquad\qquad \mathit{tree}))) & \\
\wedge & ((\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) = \text{tree-size}\,(\mathit{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})), & \\
& \qquad\qquad a, & \\
& \qquad\qquad b, & \\
& \qquad\qquad \mathbf{f}, & \\
& \qquad\qquad \text{mpg}\,(\text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{t})), & \\
& \qquad\qquad \text{list}\,(\mathbf{f},\ \mathbf{f},\ \mathbf{t},\ \mathbf{t}), & \\
& \qquad\qquad \mathit{tree}))) &
\end{array}
$$

$\wedge$ ((bv2p (a, b) $\wedge$ (length (a) = tree-size ($tree$)) $\wedge$ boolp (c))
  $\rightarrow$ (v-alu (c, a, b, list (**t**, **f**, **t**, **t**))
    = core-alu (carry-in-help (list (c, **f**, **t**, **f**, **t**, **t**)),
      a,
      b,
      f,
      mpg (list (**f**, **f**, **t**, **f**, **t**, **t**)),
      list (**t**, **f**, **t**, **t**),
      $tree$)))
$\wedge$ ((bv2p (a, b) $\wedge$ (length (a) = tree-size ($tree$)) $\wedge$ boolp (c))
  $\rightarrow$ (v-alu (c, a, b, list (**f**, **t**, **t**, **t**))
    = core-alu (carry-in-help (list (c, **f**, **f**, **t**, **t**, **t**)),
      a,
      b,
      f,
      mpg (list (**f**, **f**, **f**, **t**, **t**, **t**)),
      list (**f**, **t**, **t**, **t**),
      $tree$)))
$\wedge$ ((bv2p (a, b) $\wedge$ (length (a) = tree-size ($tree$)) $\wedge$ boolp (c))
  $\rightarrow$ (v-alu (c, a, b, list (**t**, **t**, **t**, **t**))
    = core-alu (carry-in-help (list (c, **f**, **t**, **t**, **t**, **t**)),
      a,
      b,
      f,
      mpg (list (**f**, **f**, **t**, **t**, **t**, **t**)),
      list (**t**, **t**, **t**, **t**),
      $tree$)))

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
(bvp (op)
 $\wedge$ (length (op) = 4)
 $\wedge$ **if** op = list (**f**, **f**, **f**, **f**) **then** p
  elseif op = list (**t**, **f**, **f**, **f**) **then** p
  elseif op = list (**f**, **t**, **f**, **f**) **then** p
  elseif op = list (**t**, **t**, **f**, **f**) **then** p
  elseif op = list (**f**, **f**, **t**, **f**) **then** p
  elseif op = list (**t**, **f**, **t**, **f**) **then** p
  elseif op = list (**f**, **t**, **t**, **f**) **then** p
  elseif op = list (**t**, **t**, **t**, **f**) **then** p
  elseif op = list (**f**, **f**, **f**, **t**) **then** p
  elseif op = list (**t**, **f**, **f**, **t**) **then** p

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**, **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))))))
∧  (cadddr (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)
          ∪   (T-CARRY$NETLIST
               ∪   (CARRY-OUT-HELP$NETLIST
```

$\cup$ (OVERFLOW-HELP$NETLIST
$\cup$ (tv-shift-or-buf$netlist (*tree*)
$\cup$ fast-zero$netlist (tree-size (*tree*))))))

THEOREM: core-alu$value
(core-alu& (*netlist*, *tree*)
$\land$ (length (a) = tree-size (*tree*))
$\land$ (length (b) = tree-size (*tree*))
$\land$ **if** length (a) < 3 **then f**
else t endif
$\land$ properp (a)
$\land$ properp (b)
$\land$ properp (*op*)
$\land$ (length (*op*) = 4)
$\land$ properp (*mpg*)
$\land$ (length (*mpg*) = 7))
$\rightarrow$ (dual-eval (0,
index ('core-alu, tree-number (*tree*)),
cons (c, append (a, append (b, cons (*zero*, append (*mpg*, *op*))))),
state,
netlist)
= f$core-alu ($c$, a, b, *zero*, *mpg*, *op*, *tree*))

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:
op-code (*instruction*) = 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: z-set (*set-flags*) = nth (0, *set-flags*)

DEFINITION: n-set (*set-flags*) = nth (1, *set-flags*)

DEFINITION: v-set (*set-flags*) = nth (2, *set-flags*)

DEFINITION: c-set (*set-flags*) = nth (3, *set-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: $\text{z-flag}\,(\mathit{flags}) = \text{nth}\,(\mathtt{0},\ \mathit{flags})$

DEFINITION: $\text{n-flag}\,(\mathit{flags}) = \text{nth}\,(\mathtt{1},\ \mathit{flags})$

DEFINITION: $\text{v-flag}\,(\mathit{flags}) = \text{nth}\,(\mathtt{2},\ \mathit{flags})$

DEFINITION: $\text{c-flag}\,(\mathit{flags}) = \text{nth}\,(\mathtt{3},\ \mathit{flags})$

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: $\text{reg-direct-p}\,(\mathit{mode}) = (\mathit{mode} = \text{list}\,(\mathbf{f},\ \mathbf{f}))$

DEFINITION: $\text{reg-indirect-p}\,(\mathit{mode}) = (\mathit{mode} = \text{list}\,(\mathbf{t},\ \mathbf{f}))$

DEFINITION: $\text{pre-dec-p}\,(\mathit{mode}) = (\mathit{mode} = \text{list}\,(\mathbf{f},\ \mathbf{t}))$

DEFINITION: $\text{post-inc-p}\,(\mathit{mode}) = (\mathit{mode} = \text{list}\,(\mathbf{t},\ \mathbf{t}))$

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: regs (*state*) = nth (0, *state*)

DEFINITION: flags (*state*) = nth (1, *state*)

EVENT: Enable regs; name this event 'regs-off'.

EVENT: Enable flags; name this event 'flags-off'.

DEFINITION:

```
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:

```
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 ($pc\text{-}reg$, $regs$), mem))

DEFINITION:
fm9001-step ($state$, $pc\text{-}reg$)
= fm9001-fetch (regs (car ($state$)), flags (car ($state$)), cadr ($state$), $pc\text{-}reg$)

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 ($state$, $pc\text{-}reg$, n)
= **if** $n \simeq 0$ **then** $state$
else fm9001-interpreter (fm9001-step ($state$, $pc\text{-}reg$), $pc\text{-}reg$, $n - 1$) **endif**

EVENT: Enable fm9001-interpreter; name this event 'fm9001-interpreter-off'.

THEOREM: open-fm9001-interpreter
(($n \simeq 0$) → (fm9001-interpreter ($state$, $pc\text{-}reg$, n) = $state$))
∧ (($n \not\simeq 0$)
→ (fm9001-interpreter ($state$, $pc\text{-}reg$, n)
= fm9001-interpreter (fm9001-step ($state$, $pc\text{-}reg$),
$pc\text{-}reg$,
$n - 1$)))

DEFINITION:
fm9001-intr ($state$, $pc\text{-}reg\text{-}input$)
= **if** $pc\text{-}reg\text{-}input$ ≃ **nil** **then** $state$
else fm9001-intr (fm9001-step ($state$, car ($pc\text{-}reg\text{-}input$)),
cdr ($pc\text{-}reg\text{-}input$)) **endif**

EVENT: Enable fm9001-intr; name this event 'fm9001-intr-off'.

THEOREM: properp-length-ir-accessors
properp (a-immediate ($i\text{-}reg$))
∧ (length (a-immediate ($i\text{-}reg$)) = 9)
∧ properp (rn-a ($i\text{-}reg$))

$\wedge$ (length (rn-a (*i-reg*)) = 4)
$\wedge$ properp (mode-a (*i-reg*))
$\wedge$ (length (mode-a (*i-reg*)) = 2)
$\wedge$ properp (rn-b (*i-reg*))
$\wedge$ (length (rn-b (*i-reg*)) = 4)
$\wedge$ properp (mode-b (*i-reg*))
$\wedge$ (length (mode-b (*i-reg*)) = 2)
$\wedge$ properp (set-flags (*i-reg*))
$\wedge$ (length (set-flags (*i-reg*)) = 4)
$\wedge$ properp (store-cc (*i-reg*))
$\wedge$ (length (store-cc (*i-reg*)) = 4)
$\wedge$ properp (op-code (*i-reg*))
$\wedge$ (length (op-code (*i-reg*)) = 4)

THEOREM: bvp-ir-accessors
(bvp (*i-reg*) $\wedge$ (length (*i-reg*) = 32))
$\rightarrow$ (bvp (a-immediate (*i-reg*))
$\wedge$ bvp (rn-a (*i-reg*))
$\wedge$ bvp (mode-a (*i-reg*))
$\wedge$ bvp (rn-b (*i-reg*))
$\wedge$ bvp (mode-b (*i-reg*))
$\wedge$ bvp (set-flags (*i-reg*))
$\wedge$ bvp (store-cc (*i-reg*))
$\wedge$ bvp (op-code (*i-reg*)))

THEOREM: boolp-a-immediate-p
(bvp (*ir*) $\wedge$ (length (*ir*) = 32)) $\rightarrow$ boolp (a-immediate-p (*ir*))

THEOREM: bvp-length-update-flags
bvp (update-flags (*flags*, *set-flags*, *cvzbv*))
$\wedge$ (length (update-flags (*flags*, *set-flags*, *cvzbv*)) = 4)

THEOREM: boolp-c-flag-update-flags
boolp (c-flag (update-flags (*flags*, *set-flags*, *cvzbv*)))

THEOREM: bvp-a-immediate
(bvp (*ir*) $\wedge$ (8 < length (*ir*))) $\rightarrow$ bvp (a-immediate (*ir*))

THEOREM: reg-direct->not-reg-indirect
reg-direct-p (*mode*)
$\rightarrow$ ((reg-indirect-p (*mode*) = **f**)
$\wedge$ (pre-dec-p (*mode*) = **f**)
$\wedge$ (post-inc-p (*mode*) = **f**))

THEOREM: boolp-store-result-p
(bvp (*flags*) $\wedge$ (length (*flags*) = 4))
$\rightarrow$ boolp (store-resultp (*store-cc*, *flags*))

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: intp (x) $=$ (($x \in \mathbf{N}$) $\vee$ negativep (x))

DEFINITION:
int-to-bv (x, $size$)
$=$ **if** negativep (x)
then v-subtracter-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: c10-int (x) $=$ bv-to-int (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: sub (x, y) $=$ add (x, integer-minus (y))

DEFINITION:
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:
log2 (n)
$=$ **if** $(n \simeq 0) \vee (n = 1)$ **then** 0
else $1 + \text{log2}(n \div 2)$ **endif**

DEFINITION:
asm-register (x)
$=$ **case on** x:
case $=$ *r0*
then list $(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r1*
then list $(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r2*
then list $(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r3*
then list $(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r4*
then list $(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r5*
then list $(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r6*
then list $(\mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r7*
then list $(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{f})$
case $=$ *r8*
then list $(\mathbf{f}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
case $=$ *r9*
then list $(\mathbf{t}, \mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
case $=$ *r10*
then list $(\mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
case $=$ *r11*
then list $(\mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
case $=$ *r12*
then list $(\mathbf{f}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
case $=$ *r13*
then list $(\mathbf{t}, \mathbf{f}, \mathbf{t}, \mathbf{t}, \mathbf{f}, \mathbf{f})$
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 = *lsr*
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:
v-to-nat-all (x)
= **if** $x \simeq$ **nil then nil**
else cons (v-to-nat (car (x)), v-to-nat-all (cdr (x))) **endif**

Definition:
v-to-0s-through-fs (x, *collect*)
= **if** $x \simeq$ **nil then** *collect*
else v-to-0s-through-fs (cddddr (x),
cons (**case on** v-to-nat (firstn (4, x)):
case = 0
then cadr (unpack ('x0))
case = 1
then cadr (unpack ('x1))
case = 2
then cadr (unpack ('x2))
case = 3
then cadr (unpack ('x3))
case = 4
then cadr (unpack ('x4))
case = 5
then cadr (unpack ('x5))
case = 6
then cadr (unpack ('x6))
case = 7
then cadr (unpack ('x7))
case = 8
then cadr (unpack ('x8))
case = 9
then cadr (unpack ('x9))
case = 10
then cadr (unpack ('xa))
case = 11
then cadr (unpack ('xb))
case = 12
then cadr (unpack ('xc))
case = 13
then cadr (unpack ('xd))
case = 14
then cadr (unpack ('xe))
otherwise cadr (unpack ('xf)) **endcase**,
collect)) **endif**

Definition:
v-to-hex (x) = pack (cons (car (unpack ('x)), v-to-0s-through-fs (x, 0)))

DEFINITION:
v-to-hex-all(x)
$=$ **if** $x \simeq$ **nil then nil**
else cons(v-to-hex(car(x)), v-to-hex-all(cdr(x))) **endif**

DEFINITION:
make-pairs(*list*)
$=$ **if** (*list* $\simeq$ **nil**) $\vee$ (cdr(*list*) $\simeq$ **nil**) **then nil**
else cons(cons(car(*list*), cadr(*list*)), make-pairs(cddr(*list*))) **endif**

DEFINITION:
add-ram-marker(*list-mem*)
$=$ **if** *list-mem* $\simeq$ **nil then nil**
else cons(ram(car(*list-mem*)), add-ram-marker(cdr(*list-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 + log2(length(*mem*)))
then `'(list-to-mem= insufficient-size)`
else stub-right(list-to-mem1(*mem*, *default*),
size $-$ (1 + log2(length(*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$store-resultp-mux; name this event 'f$store-resultp-mux-off'.

DEFINITION:
STORE-RESULTP-MUX*
= '(store-resultp-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-resultp-mux& (*netlist*)
= ((lookup-module ('store-resultp-mux, *netlist*)
     = STORE-RESULTP-MUX*)
   ∧ b-nand& (delete-module ('store-resultp-mux, *netlist*))
   ∧ b-not& (delete-module ('store-resultp-mux, *netlist*))
   ∧ ao2& (delete-module ('store-resultp-mux, *netlist*)))

EVENT: Enable store-resultp-mux&; name this event 'store-resultp-mux&-off'.

DEFINITION:
STORE-RESULTP-MUX$NETLIST
= cons (STORE-RESULTP-MUX*,
        (B-NAND$NETLIST ∪ B-NOT$NETLIST) ∪ AO2$NETLIST)

THEOREM: store-resultp-mux$value
store-resultp-mux& (*netlist*)
→ (dual-eval (0,
              'store-resultp-mux,

list ($s0$, $s1$, $s2$, $d0$, $d1$, $d2$, $d3$, $d4$, $d5$, $d6$),
state,
netlist)
= list (f$store-resultp-mux ($s0$, $s1$, $s2$, $d0$, $d1$, $d2$, $d3$, $d4$, $d5$, $d6$)))

EVENT: Enable store-resultp-mux$value; name this event ‘store-resultp-mux$value-off’.

THEOREM: f$store-resultp-mux=store-resultp-mux
(boolp ($s0$)
∧ boolp ($s1$)
∧ boolp ($s2$)
∧ boolp ($d0$)
∧ boolp ($d1$)
∧ boolp ($d2$)
∧ boolp ($d3$)
∧ boolp ($d4$)
∧ boolp ($d5$)
∧ boolp ($d6$))
→ (f$store-resultp-mux ($s0$, $s1$, $s2$, $d0$, $d1$, $d2$, $d3$, $d4$, $d5$, $d6$)
= store-resultp-mux ($s0$, $s1$, $s2$, $d0$, $d1$, $d2$, $d3$, $d4$, $d5$, $d6$))

DEFINITION:
b-store-resultp (*store-cc*, *flags*)
= b-xor (car (*store-cc*),
store-resultp-mux (cadr (*store-cc*),
caddr (*store-cc*),
cadddr (*store-cc*),
cadddr (*flags*),
caddr (*flags*),
cadr (*flags*),
car (*flags*),
b-or (cadddr (*flags*), car (*flags*)),
b-xor (cadr (*flags*), caddr (*flags*)),
b-or (car (*flags*),
b-xor (cadr (*flags*), caddr (*flags*)))))

EVENT: Enable b-store-resultp; name this event ‘b-store-resultp-off’.

DEFINITION:
f$b-store-resultp (*store-cc*, *flags*)
= f-xor (car (*store-cc*),
f$store-resultp-mux (cadr (*store-cc*),
caddr (*store-cc*),

cadddr (*store-cc*),
cadddr (*flags*),
caddr (*flags*),
cadr (*flags*),
car (*flags*),
f-or (cadddr (*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 ∪ STORE-RESULTP-MUX$NETLIST) ∪ B-XOR$NETLIST)

THEOREM: b-store-resultp=store-resultp$help
(boolp (*s0*)
∧ boolp (*s1*)
∧ boolp (*s2*)
∧ boolp (*s3*)
∧ boolp (*s4*)
∧ boolp (*s5*)
∧ boolp (*s6*)
∧ boolp (*s7*))
→ (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*)
∧ (length (*store-cc*) = 4)
∧ bvp (*flags*)
∧ (length (*flags*) = 4))
→ (b-store-resultp (*store-cc*, *flags*) = store-resultp (*store-cc*, *flags*))

THEOREM: b-store-resultp$value
(b-store-resultp& (*netlist*)
∧ properp (*store-cc*)
∧ (length (*store-cc*) = 4)
∧ properp (*store-cc*)
∧ (length (*flags*) = 4))
→ (dual-eval (0,
`'b-store-resultp`,
append (*store-cc*, *flags*),
state,
netlist)
= list (f$b-store-resultp (*store-cc*, *flags*)))

DEFINITION:
f$unary-op-code-p (*op*)
= f-nand4 (f-nand (f-not (cadddr (*op*)), f-not (cadr (*op*))),
f-nand (f-not (caddr (*op*)), f-not (cadr (*op*))),
f-nand3 (cadddr (*op*), cadr (*op*), f-not (car (*op*))),
f-nand3 (cadddr (*op*), caddr (*op*), cadr (*op*)))

EVENT: Enable f$unary-op-code-p; name this event 'f$unary-op-code-p-off'.

THEOREM: f$unary-op-code-p=unary-op-code-p
(bvp (*op*) ∧ (length (*op*) = 4))
→ (f$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 ∪ B-NAND$NETLIST) ∪ B-NAND3$NETLIST)
∪ 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$)
= 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*

```
=  '(decode-reg-mode
     (m0 m1)
     (direct pre-dec side-effect)
     ((g0 (direct) b-nor (m0 m1))
      (g1 (m1-) b-not (m1))
      (g2 (pre-dec) b-nor (m0 m1-))
      (g3 (side-effect) id (m1)))
     nil)
```

DEFINITION:
decode-reg-mode& (*netlist*)
= ((lookup-module (`'decode-reg-mode`, *netlist*) = DECODE-REG-MODE*)
  ∧ b-not& (delete-module (`'decode-reg-mode`, *netlist*))
  ∧ b-nor& (delete-module (`'decode-reg-mode`, *netlist*))
  ∧ id& (delete-module (`'decode-reg-mode`, *netlist*)))

EVENT: Enable decode-reg-mode&; name this event 'decode-reg-mode&-off'.

DEFINITION:
DECODE-REG-MODE$NETLIST
= cons (DECODE-REG-MODE*,
  (B-NOT$NETLIST ∪ B-NOR$NETLIST) ∪ ID$NETLIST)

THEOREM: decode-reg-mode$value
decode-reg-mode& (*netlist*)
→ (dual-eval (0, `'decode-reg-mode`, *mode*, *state*, *netlist*)
  = f$decode-reg-mode (*mode*))

EVENT: Enable decode-reg-mode$value; name this event 'decode-reg-mode$value-off'.

THEOREM: f$decode-reg-mode-as-reg-mode
(bvp (*mode*) ∧ (length (*mode*) = 2))
→ (f$decode-reg-mode (*mode*)
  = list (reg-direct-p (*mode*),
    pre-dec-p (*mode*),
    pre-dec-p (*mode*) ∨ 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*, cadddr (*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*))
∧ (length (f$select-op-code (*select*, *dec*, *op*)) = 4)

THEOREM: f$select-op-code-selects
(boolp (*select*) ∧ boolp (*dec*) ∧ bvp (*op*) ∧ (length (*op*) = 4))
→ (f$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*)
∧ b-not& (delete-module (`'select-op-code`, *netlist*))
∧ b-nand& (delete-module (`'select-op-code`, *netlist*))
∧ b-if& (delete-module (`'select-op-code`, *netlist*))
∧ 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 ∪ B-NAND$NETLIST) ∪ B-IF$NETLIST)
∪ B-AND$NETLIST)

THEOREM: select-op-code$value
(select-op-code& (*netlist*) ∧ properp (*op*) ∧ (length (*op*) = 4))
→ (dual-eval (0,
'select-op-code,
cons (*select*, cons (*dec*, *op*)),
state,
netlist)
= f$select-op-code (*select*, *dec*, *op*))

EVENT: Enable select-op-code$value; name this event 'select-op-code$value-off'.

DEFINITION:
V-IF-F-4*
= '(v-if-f-4

```
    (c a0 a1 a2 a3)
    (z0 z1 z2 z3)
    ((cb (c-) b-not (c))
     (g0 (z0) b-and (c- a0))
     (g1 (z1) b-and (c- a1))
     (g2 (z2) b-and (c- a2))
     (g3 (z3) b-and (c- a3)))
    nil)
```

DEFINITION:
v-if-f-4& (*netlist*)
= ((lookup-module ('v-if-f-4, *netlist*) = V-IF-F-4*)
∧ b-not& (delete-module ('v-if-f-4, *netlist*))
∧ b-and& (delete-module ('v-if-f-4, *netlist*)))

EVENT: Enable v-if-f-4&; name this event 'v-if-f-4&-off'.

DEFINITION:
V-IF-F-4$NETLIST = cons (V-IF-F-4*, B-NOT$NETLIST ∪ B-AND$NETLIST)

DEFINITION:
f$v-if-f-4 (*c*, *a*)
= list (f-and (f-not (*c*), car (*a*)),
f-and (f-not (*c*), cadr (*a*)),
f-and (f-not (*c*), caddr (*a*)),
f-and (f-not (*c*), cadddr (*a*)))

EVENT: Enable f$v-if-f-4; name this event 'f$v-if-f-4-off'.

THEOREM: properp-length-f$v-if-f-4
properp (f$v-if-f-4 (*c*, *a*)) ∧ (length (f$v-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 ∪ 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))),

$$
\begin{aligned}
&\qquad\qquad \text{f-not}\,(\text{caddddr}\,(s)))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{cadddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{caddddr}\,(s)))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{f-not}\,(\text{car}\,(s))),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{cadddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{caddddr}\,(s)))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{f-not}\,(\text{cadr}\,(s)))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{cadddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{caddddr}\,(s)))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{f-not}\,(\text{car}\,(s))),\ \text{f-not}\,(\text{f-not}\,(\text{cadr}\,(s)))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{cadddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{caddddr}\,(s)))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{caddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{f-not}\,(\text{car}\,(s))),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{caddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{f-not}\,(\text{cadr}\,(s)))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{caddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{f-not}\,(\text{car}\,(s))),\ \text{f-not}\,(\text{f-not}\,(\text{cadr}\,(s)))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{caddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{f-not}\,(\text{car}\,(s))),\ \text{f-not}\,(\text{cadr}\,(s))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),\\
&\qquad\qquad \text{f-not}\,(\text{f-not}\,(\text{caddddr}\,(s))))),\\
&\text{f-nor}\,(\text{f-nand}\,(\text{f-not}\,(\text{car}\,(s)),\ \text{f-not}\,(\text{f-not}\,(\text{cadr}\,(s)))),\\
&\quad \text{f-nand3}\,(\text{f-not}\,(\text{f-not}\,(\text{caddr}\,(s))),\\
&\qquad\qquad \text{f-not}\,(\text{cadddr}\,(s)),
\end{aligned}
$$

```
                  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 (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 (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-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 (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 (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 (cadddr (s)),
                 b-not (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 (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 (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 (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 (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 (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 (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 (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 (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 (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 (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 (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 (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 (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 (b-not (cadddr (s))),
b-not (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 (b-not (cadddr (s))),
b-not (b-not (caddddr (s))))))

EVENT: Enable decode-5; name this event 'decode-5-off'.

THEOREM: bvp-length-decode-5
bvp (decode-5 (s)) $\land$ (length (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 (l3 h5))
    (gx06 (x06) b-nor (l0 h6))
    (gx16 (x16) b-nor (l1 h6))
    (gx26 (x26) b-nor (l2 h6))
    (gx36 (x36) b-nor (l3 h6))
    (gx07 (x07) b-nor (l0 h7))
    (gx17 (x17) b-nor (l1 h7))
    (gx27 (x27) b-nor (l2 h7))
    (gx37 (x37) b-nor (l3 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,

$$
\begin{array}{ll}
 & s4, \\
 & s5, \\
 & s6, \\
 & s7, \\
 & s8, \\
 & s9, \\
 & s10, \\
 & s11, \\
 & s12, \\
 & s13, \\
 & s14, \\
 & s15, \\
 & s16, \\
 & s17, \\
 & s18, \\
 & s19, \\
 & s20, \\
 & s21, \\
 & s22, \\
 & s23, \\
 & s24, \\
 & s25, \\
 & s26, \\
 & s27, \\
 & s28, \\
 & s29, \\
 & s30, \\
 & s31) \\
= & \text{list}(\text{b-nand4}(\text{b-nor4}(s1, s3, s5, s7), \\
 & \qquad\qquad \text{b-nor4}(s9, s11, s13, s15), \\
 & \qquad\qquad \text{b-nor4}(s17, s19, s21, s23), \\
 & \qquad\qquad \text{b-nor}(s25, s29)), \\
 & \quad \text{b-nand4}(\text{b-nor4}(s2, s3, s6, s7), \\
 & \qquad\qquad \text{b-nor4}(s10, s11, s14, s15), \\
 & \qquad\qquad \text{b-nor4}(s18, s19, s22, s23), \\
 & \qquad\qquad \text{b-not}(s30)), \\
 & \quad \text{b-nand4}(\text{b-nor4}(s4, s5, s6, s7), \\
 & \qquad\qquad \text{b-nor4}(s12, s13, s14, s15), \\
 & \qquad\qquad \text{b-nor4}(s20, s21, s22, s23), \\
 & \qquad\qquad \text{b-nor3}(s28, s29, s30)), \\
 & \quad \text{b-nand4}(\text{b-nor4}(s8, s9, s10, s11), \\
 & \qquad\qquad \text{b-nor4}(s12, s13, s14, s15), \\
 & \qquad\qquad \text{b-nor}(s24, s25), \\
 & \qquad\qquad \text{b-nor3}(s28, s29, s30)),
\end{array}
$$

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,

$$
\begin{aligned}
&\qquad\qquad s18,\\
&\qquad\qquad s19,\\
&\qquad\qquad s20,\\
&\qquad\qquad s21,\\
&\qquad\qquad s22,\\
&\qquad\qquad s23,\\
&\qquad\qquad s24,\\
&\qquad\qquad s25,\\
&\qquad\qquad s26,\\
&\qquad\qquad s27,\\
&\qquad\qquad s28,\\
&\qquad\qquad s29,\\
&\qquad\qquad s30,\\
&\qquad\qquad s31),\\
&\qquad state,\\
&\qquad netlist)\\
= \;& \mathrm{f\$encode\text{-}32}\,(s0,\\
&\qquad\qquad s1,\\
&\qquad\qquad s2,\\
&\qquad\qquad s3,\\
&\qquad\qquad s4,\\
&\qquad\qquad s5,\\
&\qquad\qquad s6,\\
&\qquad\qquad s7,\\
&\qquad\qquad s8,\\
&\qquad\qquad s9,\\
&\qquad\qquad s10,\\
&\qquad\qquad s11,\\
&\qquad\qquad s12,\\
&\qquad\qquad s13,\\
&\qquad\qquad s14,\\
&\qquad\qquad s15,\\
&\qquad\qquad s16,\\
&\qquad\qquad s17,\\
&\qquad\qquad s18,\\
&\qquad\qquad s19,\\
&\qquad\qquad s20,\\
&\qquad\qquad s21,\\
&\qquad\qquad s22,\\
&\qquad\qquad s23,\\
&\qquad\qquad s24,\\
&\qquad\qquad s25,\\
&\qquad\qquad s26,\\
&\qquad\qquad s27,
\end{aligned}
$$

$s28$,
$s29$,
$s30$,
$s31$))

EVENT: Enable encode-32$value; name this event 'encode-32$value-off'.

THEOREM: f$encode-32=encode-32
(boolp ($s0$)
∧ boolp ($s1$)
∧ boolp ($s2$)
∧ boolp ($s3$)
∧ boolp ($s4$)
∧ boolp ($s5$)
∧ boolp ($s6$)
∧ boolp ($s7$)
∧ boolp ($s8$)
∧ boolp ($s9$)
∧ boolp ($s10$)
∧ boolp ($s11$)
∧ boolp ($s12$)
∧ boolp ($s13$)
∧ boolp ($s14$)
∧ boolp ($s15$)
∧ boolp ($s16$)
∧ boolp ($s17$)
∧ boolp ($s18$)
∧ boolp ($s19$)
∧ boolp ($s20$)
∧ boolp ($s21$)
∧ boolp ($s22$)
∧ boolp ($s23$)
∧ boolp ($s24$)
∧ boolp ($s25$)
∧ boolp ($s26$)
∧ boolp ($s27$)
∧ boolp ($s28$)
∧ boolp ($s29$)
∧ boolp ($s30$)
∧ boolp ($s31$))
→ (f$encode-32 ($s0$,
$s1$,
$s2$,
$s3$,

$$
\begin{aligned}
&\qquad\quad s4, \\
&\qquad\quad s5, \\
&\qquad\quad s6, \\
&\qquad\quad s7, \\
&\qquad\quad s8, \\
&\qquad\quad s9, \\
&\qquad\quad s10, \\
&\qquad\quad s11, \\
&\qquad\quad s12, \\
&\qquad\quad s13, \\
&\qquad\quad s14, \\
&\qquad\quad s15, \\
&\qquad\quad s16, \\
&\qquad\quad s17, \\
&\qquad\quad s18, \\
&\qquad\quad s19, \\
&\qquad\quad s20, \\
&\qquad\quad s21, \\
&\qquad\quad s22, \\
&\qquad\quad s23, \\
&\qquad\quad s24, \\
&\qquad\quad s25, \\
&\qquad\quad s26, \\
&\qquad\quad s27, \\
&\qquad\quad s28, \\
&\qquad\quad s29, \\
&\qquad\quad s30, \\
&\qquad\quad s31) \\
= \;& \text{encode-32}(s0, \\
&\qquad\quad s1, \\
&\qquad\quad s2, \\
&\qquad\quad s3, \\
&\qquad\quad s4, \\
&\qquad\quad s5, \\
&\qquad\quad s6, \\
&\qquad\quad s7, \\
&\qquad\quad s8, \\
&\qquad\quad s9, \\
&\qquad\quad s10, \\
&\qquad\quad s11, \\
&\qquad\quad s12, \\
&\qquad\quad s13, \\
&\qquad\quad s14, \\
&\qquad\quad s15,
\end{aligned}
$$

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,

$s26$,
$s27$,
$s28$,
$s29$,
$s30$,
$s31$))
$\land$ (length (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: properp-length-f$encode-32
properp (f$encode-32 ($s0$,
$s1$,
$s2$,

$s3$,
$s4$,
$s5$,
$s6$,
$s7$,
$s8$,
$s9$,
$s10$,
$s11$,
$s12$,
$s13$,
$s14$,
$s15$,
$s16$,
$s17$,
$s18$,
$s19$,
$s20$,
$s21$,
$s22$,
$s23$,
$s24$,
$s25$,
$s26$,
$s27$,
$s28$,
$s29$,
$s30$,
$s31$))
$\wedge$ (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
((length ($args$) = 32) ∧ properp ($args$))
→ (dual-eval ($flag$, $name$, $args$, $state$, $netlist$)
= dual-eval ($flag$,
$name$,
list (nth (0, $args$),
nth (1, $args$),
nth (2, $args$),
nth (3, $args$),
nth (4, $args$),
nth (5, $args$),
nth (6, $args$),
nth (7, $args$),
nth (8, $args$),
nth (9, $args$),
nth (10, $args$),
nth (11, $args$),
nth (12, $args$),
nth (13, $args$),
nth (14, $args$),
nth (15, $args$),
nth (16, $args$),
nth (17, $args$),
nth (18, $args$),
nth (19, $args$),

nth (20, *args*),
nth (21, *args*),
nth (22, *args*),
nth (23, *args*),
nth (24, *args*),
nth (25, *args*),
nth (26, *args*),
nth (27, *args*),
nth (28, *args*),
nth (29, *args*),
nth (30, *args*),
nth (31, *args*)),
state,
netlist))

THEOREM: encode-32$value-on-a-vector
(encode-32& (*netlist*) ∧ properp (*args*) ∧ (length (*args*) = 32))
→ (dual-eval (0, 'encode-32, *args*, *state*, *netlist*)
= f$encode-32 (nth (0, *args*),
nth (1, *args*),
nth (2, *args*),
nth (3, *args*),
nth (4, *args*),
nth (5, *args*),
nth (6, *args*),
nth (7, *args*),
nth (8, *args*),
nth (9, *args*),
nth (10, *args*),
nth (11, *args*),
nth (12, *args*),
nth (13, *args*),
nth (14, *args*),
nth (15, *args*),
nth (16, *args*),
nth (17, *args*),
nth (18, *args*),
nth (19, *args*),
nth (20, *args*),
nth (21, *args*),
nth (22, *args*),
nth (23, *args*),
nth (24, *args*),
nth (25, *args*),

nth (26, *args*),
nth (27, *args*),
nth (28, *args*),
nth (29, *args*),
nth (30, *args*),
nth (31, *args*)))

EVENT: Enable encode-32$value-on-a-vector; name this event ‘encode-32$value-on-a-vector-off’.

DEFINITION:
set-some-flags (*set-flags*)
= (z-set (*set-flags*)
∨ n-set (*set-flags*)
∨ v-set (*set-flags*)
∨ c-set (*set-flags*))

EVENT: Enable set-some-flags; name this event ‘set-some-flags-off’.

THEOREM: not-set-some-flags-make-list-4-f
¬ set-some-flags (make-list (4, **f**))

DEFINITION:
flags-hyps (*flags*) = ((length (*flags*) = 4) ∧ bvp (*flags*))

THEOREM: not-set-some-flags-update-flags
(flags-hyps (*flags*) ∧ (¬ set-some-flags (*set-flags*)))
→ (update-flags (*flags*, *set-flags*, *cvzbv*) = *flags*)

THEOREM: boolp-c-flag
flags-hyps (*flags*) → boolp (c-flag (*flags*))

DEFINITION:
f$set-some-flags (*set-flags*)
= f-or4 (car (*set-flags*),
cadr (*set-flags*),
caddr (*set-flags*),
cadddr (*set-flags*))

EVENT: Enable f$set-some-flags; name this event ‘f$set-some-flags-off’.

THEOREM: f$set-some-flags=set-some=flags
(bvp (*set-flags*) ∧ (length (*set-flags*) = 4))
→ (f$set-some-flags (*set-flags*) = set-some-flags (*set-flags*))

THEOREM: b-or4$value-as-f$set-some-flags
(b-or4& (*netlist*) ∧ properp (*set-flags*) ∧ (length (*set-flags*) = `4`))
→ (dual-eval (`0`, `'b-or4`, *set-flags*, *state*, *netlist*)
  = list (f$set-some-flags (*set-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-t-regs-address (*regs-address*)
= f-and4 (car (*regs-address*),
  cadr (*regs-address*),
  caddr (*regs-address*),
  cadddr (*regs-address*))

EVENT: Enable f$all-t-regs-address; name this event 'f$all-t-regs-address-off'.

THEOREM: f$all-t-regs-address=set-some=flags
(bvp (*regs-address*) ∧ (length (*regs-address*) = `4`))
→ (f$all-t-regs-address (*regs-address*)
  = (*regs-address* = make-list (`4`, `t`)))

THEOREM: b-and4$value-as-f$all-t-regs-address
(b-and4& (*netlist*)
∧ properp (*regs-address*)
∧ (length (*regs-address*) = `4`))
→ (dual-eval (`0`, `'b-and4`, *regs-address*, *state*, *netlist*)
  = list (f$all-t-regs-address (*regs-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-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
= (boolp (*rw-*)
  ∧ (length (*regs-address*) = `4`)
  ∧ bvp (*regs-address*)
  ∧ (length (*i-reg*) = `32`)
  ∧ bvp (*i-reg*)
  ∧ flags-hyps (*flags*)
  ∧ (length (*pc-reg*) = `4`)
  ∧ bvp (*pc-reg*))

DEFINITION: V_FETCH0 = 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-to-nat (list (**f**, **f**, **f**, **f**, **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 = list (**t**, **f**, **f**, **f**, **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-to-nat (list (**t**, **f**, **f**, **f**, **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 = list (**f**, **t**, **f**, **f**, **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-to-nat (list (**f**, **t**, **f**, **f**, **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 = list (**t**, **t**, **f**, **f**, **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: N_FETCH3 = v-to-nat (list (**t**, **t**, **f**, **f**, **f**))

EVENT: Enable n_fetch3; name this event 'n_fetch3-off'.

EVENT: Enable *1*n_fetch3; name this event 'g*1*n_fetch3-off'.

DEFINITION: V_DECODE = list (**f**, **f**, **t**, **f**, **f**)

EVENT: Enable v_decode; name this event 'v_decode-off'.

EVENT: Enable *1*v_decode; name this event 'g*1*v_decode-off'.

DEFINITION: N_DECODE = v-to-nat (list (**f**, **f**, **t**, **f**, **f**))

EVENT: Enable n_decode; name this event 'n_decode-off'.

EVENT: Enable *1*n_decode; name this event 'g*1*n_decode-off'.

DEFINITION: V_REGA = list (**t**, **f**, **t**, **f**, **f**)

EVENT: Enable v_rega; name this event 'v_rega-off'.

EVENT: Enable *1*v_rega; name this event 'g*1*v_rega-off'.

DEFINITION: N_REGA = v-to-nat (list (**t**, **f**, **t**, **f**, **f**))

EVENT: Enable n_rega; name this event 'n_rega-off'.

EVENT: Enable *1*n_rega; name this event 'g*1*n_rega-off'.

DEFINITION: V_REGB = list (**f**, **t**, **t**, **f**, **f**)

EVENT: Enable v_regb; name this event 'v_regb-off'.

EVENT: Enable *1*v_regb; name this event 'g*1*v_regb-off'.

DEFINITION: N_REGB = v-to-nat (list (**f**, **t**, **t**, **f**, **f**))

EVENT: Enable n_regb; name this event 'n_regb-off'.

EVENT: Enable *1*n_regb; name this event 'g*1*n_regb-off'.

DEFINITION: V_UPDATE = list (**t**, **t**, **t**, **f**, **f**)

EVENT: Enable v_update; name this event 'v_update-off'.

EVENT: Enable *1*v_update; name this event 'g*1*v_update-off'.

DEFINITION: N_UPDATE = v-to-nat (list (**t**, **t**, **t**, **f**, **f**))

EVENT: Enable n_update; name this event 'n_update-off'.

EVENT: Enable *1*n_update; name this event 'g*1*n_update-off'.

DEFINITION: V_READA0 = list (**f**, **f**, **f**, **t**, **f**)

EVENT: Enable v_reada0; name this event 'v_reada0-off'.

EVENT: Enable *1*v_reada0; name this event 'g*1*v_reada0-off'.

DEFINITION: N_READA0 = v-to-nat (list (**f**, **f**, **f**, **t**, **f**))

EVENT: Enable n_reada0; name this event 'n_reada0-off'.

EVENT: Enable *1*n_reada0; name this event 'g*1*n_reada0-off'.

DEFINITION: V_READA1 = list (**t**, **f**, **f**, **t**, **f**)

EVENT: Enable v_reada1; name this event 'v_reada1-off'.

EVENT: Enable *1*v_reada1; name this event 'g*1*v_reada1-off'.

DEFINITION: N_READA1 = v-to-nat (list (**t**, **f**, **f**, **t**, **f**))

EVENT: Enable n_reada1; name this event 'n_reada1-off'.

EVENT: Enable *1*n_reada1; name this event 'g*1*n_reada1-off'.

DEFINITION: V_READA2 = list (**f**, **t**, **f**, **t**, **f**)

EVENT: Enable v_reada2; name this event 'v_reada2-off'.

EVENT: Enable *1*v_reada2; name this event 'g*1*v_reada2-off'.

DEFINITION: N_READA2 = v-to-nat (list (**f**, **t**, **f**, **t**, **f**))

EVENT: Enable n_reada2; name this event 'n_reada2-off'.

EVENT: Enable *1*n_reada2; name this event 'g*1*n_reada2-off'.

DEFINITION: V_READA3 = list (**t**, **t**, **f**, **t**, **f**)

EVENT: Enable v_reada3; name this event 'v_reada3-off'.

EVENT: Enable *1*v_reada3; name this event 'g*1*v_reada3-off'.

DEFINITION: N_READA3 = v-to-nat (list (**t**, **t**, **f**, **t**, **f**))

EVENT: Enable n_reada3; name this event 'n_reada3-off'.

EVENT: Enable *1*n_reada3; name this event 'g*1*n_reada3-off'.

DEFINITION: V_READB0 = list (**f**, **f**, **t**, **t**, **f**)

EVENT: Enable v_readb0; name this event 'v_readb0-off'.

EVENT: Enable *1*v_readb0; name this event 'g*1*v_readb0-off'.

DEFINITION: N_READB0 = v-to-nat (list (**f**, **f**, **t**, **t**, **f**))

EVENT: Enable n_readb0; name this event 'n_readb0-off'.

EVENT: Enable *1*n_readb0; name this event 'g*1*n_readb0-off'.

DEFINITION: V_READB1 = list (**t**, **f**, **t**, **t**, **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-to-nat (list (**f**, **f**, **f**, **f**, **t**))

EVENT: Enable n_write0; name this event 'n_write0-off'.

EVENT: Enable *1*n_write0; name this event 'g*1*n_write0-off'.

DEFINITION: V_WRITE1 = list (**t**, **f**, **f**, **f**, **t**)

EVENT: Enable v_write1; name this event 'v_write1-off'.

EVENT: Enable *1*v_write1; name this event 'g*1*v_write1-off'.

DEFINITION: N_WRITE1 = v-to-nat (list (**t**, **f**, **f**, **f**, **t**))

EVENT: Enable n_write1; name this event 'n_write1-off'.

EVENT: Enable *1*n_write1; name this event 'g*1*n_write1-off'.

DEFINITION: V_WRITE2 = list (**f**, **t**, **f**, **f**, **t**)

EVENT: Enable v_write2; name this event 'v_write2-off'.

EVENT: Enable *1*v_write2; name this event 'g*1*v_write2-off'.

DEFINITION: N_WRITE2 = v-to-nat (list (**f**, **t**, **f**, **f**, **t**))

EVENT: Enable n_write2; name this event 'n_write2-off'.

EVENT: Enable *1*n_write2; name this event 'g*1*n_write2-off'.

DEFINITION: V_WRITE3 = list (**t**, **t**, **f**, **f**, **t**)

EVENT: Enable v_write3; name this event 'v_write3-off'.

EVENT: Enable *1*v_write3; name this event 'g*1*v_write3-off'.

DEFINITION: N_WRITE3 = v-to-nat (list (**t**, **t**, **f**, **f**, **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 = list (**t**, **t**, **t**, **f**, **t**)

EVENT: Enable v_sefb1; name this event 'v_sefb1-off'.

EVENT: Enable *1*v_sefb1; name this event 'g*1*v_sefb1-off'.

DEFINITION: N_SEFB1 = v-to-nat (list (**t**, **t**, **t**, **f**, **t**))

EVENT: Enable n_sefb1; name this event 'n_sefb1-off'.

EVENT: Enable *1*n_sefb1; name this event 'g*1*n_sefb1-off'.

DEFINITION: V_HOLD0 = list (**f**, **f**, **f**, **t**, **t**)

EVENT: Enable v_hold0; name this event 'v_hold0-off'.

EVENT: Enable *1*v_hold0; name this event 'g*1*v_hold0-off'.

DEFINITION: N_HOLD0 = v-to-nat (list (**f**, **f**, **f**, **t**, **t**))

EVENT: Enable n_hold0; name this event 'n_hold0-off'.

EVENT: Enable *1*n_hold0; name this event 'g*1*n_hold0-off'.

DEFINITION: V_HOLD1 = list (**t**, **f**, **f**, **t**, **t**)

EVENT: Enable v_hold1; name this event 'v_hold1-off'.

EVENT: Enable *1*v_hold1; name this event 'g*1*v_hold1-off'.

DEFINITION: N_HOLD1 = v-to-nat (list (**t**, **f**, **f**, **t**, **t**))

EVENT: Enable n_hold1; name this event 'n_hold1-off'.

EVENT: Enable *1*n_hold1; name this event 'g*1*n_hold1-off'.

DEFINITION: V_V11010 = list (**f**, **t**, **f**, **t**, **t**)

EVENT: Enable v_v11010; name this event 'v_v11010-off'.

EVENT: Enable *1*v_v11010; name this event 'g*1*v_v11010-off'.

DEFINITION: N_V11010 = v-to-nat (list (**f**, **t**, **f**, **t**, **t**))

EVENT: Enable n_v11010; name this event 'n_v11010-off'.

EVENT: Enable *1*n_v11010; name this event 'g*1*n_v11010-off'.

DEFINITION: V_V11011 = list (**t**, **t**, **f**, **t**, **t**)

EVENT: Enable v_v11011; name this event 'v_v11011-off'.

EVENT: Enable *1*v_v11011; name this event 'g*1*v_v11011-off'.

DEFINITION: N_V11011 = v-to-nat (list (**t**, **t**, **f**, **t**, **t**))

EVENT: Enable n_v11011; name this event 'n_v11011-off'.

EVENT: Enable *1*n_v11011; name this event 'g*1*n_v11011-off'.

DEFINITION: V_RESET0 = list (**f**, **f**, **t**, **t**, **t**)

EVENT: Enable v_reset0; name this event 'v_reset0-off'.

EVENT: Enable *1*v_reset0; name this event 'g*1*v_reset0-off'.

DEFINITION: N_RESET0 = v-to-nat (list (**f**, **f**, **t**, **t**, **t**))

EVENT: Enable n_reset0; name this event 'n_reset0-off'.

EVENT: Enable *1*n_reset0; name this event 'g*1*n_reset0-off'.

DEFINITION: V_RESET1 = list (**t**, **f**, **t**, **t**, **t**)

EVENT: Enable v_reset1; name this event 'v_reset1-off'.

EVENT: Enable *1*v_reset1; name this event 'g*1*v_reset1-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*n_reset1; name this event 'g*1*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*v_reset2; name this event 'g*1*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*n_reset2; name this event 'g*1*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*v_v11111; name this event 'g*1*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*n_v11111; name this event 'g*1*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)
∧ bvp (V_FETCH0)
∧ (length (V_FETCH1) = 5)
∧ bvp (V_FETCH1)
∧ (length (V_FETCH2) = 5)
∧ bvp (V_FETCH2)
∧ (length (V_FETCH3) = 5)
∧ bvp (V_FETCH3)
∧ (length (V_DECODE) = 5)
∧ bvp (V_DECODE)
∧ (length (V_REGA) = 5)
∧ bvp (V_REGA)
∧ (length (V_REGB) = 5)
∧ bvp (V_REGB)
∧ (length (V_UPDATE) = 5)
∧ bvp (V_UPDATE)
∧ (length (V_READA0) = 5)
∧ bvp (V_READA0)
∧ (length (V_READA1) = 5)
∧ bvp (V_READA1)
∧ (length (V_READA2) = 5)
∧ bvp (V_READA2)
∧ (length (V_READA3) = 5)
∧ bvp (V_READA3)
∧ (length (V_READB0) = 5)
∧ bvp (V_READB0)
∧ (length (V_READB1) = 5)
∧ bvp (V_READB1)
∧ (length (V_READB2) = 5)
∧ bvp (V_READB2)
∧ (length (V_READB3) = 5)
∧ bvp (V_READB3)
∧ (length (V_WRITE0) = 5)
∧ bvp (V_WRITE0)
∧ (length (V_WRITE1) = 5)
∧ bvp (V_WRITE1)
∧ (length (V_WRITE2) = 5)
∧ bvp (V_WRITE2)
∧ (length (V_WRITE3) = 5)

$\wedge$ bvp (V_WRITE3)
$\wedge$ (length (V_SEFA0) = 5)
$\wedge$ bvp (V_SEFA0)
$\wedge$ (length (V_SEFA1) = 5)
$\wedge$ bvp (V_SEFA1)
$\wedge$ (length (V_SEFB0) = 5)
$\wedge$ bvp (V_SEFB0)
$\wedge$ (length (V_SEFB1) = 5)
$\wedge$ bvp (V_SEFB1)
$\wedge$ (length (V_HOLD0) = 5)
$\wedge$ bvp (V_HOLD0)
$\wedge$ (length (V_HOLD1) = 5)
$\wedge$ bvp (V_HOLD1)
$\wedge$ (length (V_V11010) = 5)
$\wedge$ bvp (V_V11010)
$\wedge$ (length (V_V11011) = 5)
$\wedge$ bvp (V_V11011)
$\wedge$ (length (V_RESET0) = 5)
$\wedge$ bvp (V_RESET0)
$\wedge$ (length (V_RESET1) = 5)
$\wedge$ bvp (V_RESET1)
$\wedge$ (length (V_RESET2) = 5)
$\wedge$ bvp (V_RESET2)
$\wedge$ (length (V_V11111) = 5)
$\wedge$ bvp (V_V11111)

EVENT: Enable bvp-length-state-vectors; name this event ‘bvp-length-state-vectors-off’.

DEFINITION: rw- (*cntl-state*) = nth (0, *cntl-state*)

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

DEFINITION: strobe- (*cntl-state*) = nth (1, *cntl-state*)

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

DEFINITION: hdack- (*cntl-state*) = nth (2, *cntl-state*)

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

DEFINITION: we-regs (*cntl-state*) = nth (3, *cntl-state*)

EVENT: Enable we-regs; name this event 'we-regs-off'.

DEFINITION: we-a-reg (*cntl-state*) = nth (`4`, *cntl-state*)

EVENT: Enable we-a-reg; name this event 'we-a-reg-off'.

DEFINITION: we-b-reg (*cntl-state*) = nth (`5`, *cntl-state*)

EVENT: Enable we-b-reg; name this event 'we-b-reg-off'.

DEFINITION: we-i-reg (*cntl-state*) = nth (`6`, *cntl-state*)

EVENT: Enable we-i-reg; name this event 'we-i-reg-off'.

DEFINITION: we-data-out (*cntl-state*) = nth (`7`, *cntl-state*)

EVENT: Enable we-data-out; name this event 'we-data-out-off'.

DEFINITION: we-addr-out (*cntl-state*) = nth (`8`, *cntl-state*)

EVENT: Enable we-addr-out; name this event 'we-addr-out-off'.

DEFINITION: we-hold- (*cntl-state*) = nth (`9`, *cntl-state*)

EVENT: Enable we-hold-; name this event 'we-hold–off'.

DEFINITION: we-pc-reg (*cntl-state*) = nth (`10`, *cntl-state*)

EVENT: Enable we-pc-reg; name this event 'we-pc-reg-off'.

DEFINITION: data-in-select (*cntl-state*) = nth (`11`, *cntl-state*)

EVENT: Enable data-in-select; name this event 'data-in-select-off'.

DEFINITION: dec-addr-out (*cntl-state*) = nth (`12`, *cntl-state*)

EVENT: Enable dec-addr-out; name this event 'dec-addr-out-off'.

DEFINITION: select-immediate (*cntl-state*) = nth (`13`, *cntl-state*)

EVENT: Enable select-immediate; name this event 'select-immediate-off'.

DEFINITION: alu-c (*cntl-state*) = nth (14, *cntl-state*)

EVENT: Enable alu-c; name this event 'alu-c-off'.

DEFINITION: alu-zero (*cntl-state*) = nth (15, *cntl-state*)

EVENT: Enable alu-zero; name this event 'alu-zero-off'.

DEFINITION: state (*cntl-state*) = subrange (*cntl-state*, 16, 20)

EVENT: Enable state; name this event 'state-off'.

DEFINITION: we-flags (*cntl-state*) = subrange (*cntl-state*, 21, 24)

EVENT: Enable we-flags; name this event 'we-flags-off'.

DEFINITION:
regs-address (*cntl-state*) = subrange (*cntl-state*, 25, 28)

EVENT: Enable regs-address; name this event 'regs-address-off'.

DEFINITION: alu-op (*cntl-state*) = subrange (*cntl-state*, 29, 32)

EVENT: Enable alu-op; name this event 'alu-op-off'.

DEFINITION: alu-mpg (*cntl-state*) = subrange (*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_FETCH0
 (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 F
               (CONS
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (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*)
→ ((rw- (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = *rw-*)
∧ (strobe- (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (hdack- (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (we-regs (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-a-reg (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-b-reg (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-i-reg (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-data-out (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-addr-out (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)

$\wedge$ (we-hold- (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = **t**)
$\wedge$ (we-pc-reg (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = **f**)
$\wedge$ (data-in-select (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = **f**)
$\wedge$ (dec-addr-out (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = **f**)
$\wedge$ (select-immediate (cv_fetch0 (*rw-*,
      regs-address,
      i-reg,
      flags,
      pc-reg))
  = **f**)
$\wedge$ (alu-c (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = carry-in-help (cons (c-flag (*flags*), cons (**t**, ALU-INC-OP))))
$\wedge$ (alu-zero (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = **t**)
$\wedge$ (state (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = V_FETCH0)
$\wedge$ (we-flags (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = make-list (**4**, **f**))
$\wedge$ (regs-address (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = *pc-reg*)
$\wedge$ (alu-op (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = ALU-INC-OP)
$\wedge$ (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 F
      (CONS
       (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                            (CONS F (OP-CODE I-REG))))
       (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*)
$\rightarrow$ ((rw- (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-a-reg (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-b-reg (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-i-reg (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-pc-reg (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (data-in-select (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \quad$ **f**)

$\wedge$ (dec-addr-out (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)

$\wedge$ (select-immediate (cv_fetch1 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \quad$ **f**)

$\wedge$ (alu-c (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ carry-in-help (cons (c-flag (*flags*),
cons (**f**, op-code (*i-reg*)))))

$\wedge$ (alu-zero (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)

$\wedge$ (state (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ V_FETCH1)

$\wedge$ (we-flags (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ make-list (**4**, **f**))

$\wedge$ (regs-address (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ *pc-reg*)

$\wedge$ (alu-op (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ op-code (*i-reg*))

$\wedge$ (alu-mpg (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ 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 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 F
            (CONS T
             (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*)
$\rightarrow$ ((rw- (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (strobe- (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (hdack- (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-a-reg (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-b-reg (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-i-reg (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\land$ (we-data-out (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-addr-out (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-hold- (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-pc-reg (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (data-in-select (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\land$ (dec-addr-out (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (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 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*))

$$
\begin{array}{ll}
& = \quad \mathbf{f}) \\
\wedge & (\text{state}\,(\text{cv\_decode}\,(rw\text{-},\ regs\text{-}address,\ i\text{-}reg,\ flags,\ pc\text{-}reg)) \\
& = \quad \textsc{v\_decode}) \\
\wedge & (\text{we-flags}\,(\text{cv\_decode}\,(rw\text{-},\ regs\text{-}address,\ i\text{-}reg,\ flags,\ pc\text{-}reg)) \\
& = \quad \text{make-list}\,(\mathbf{4},\ \mathbf{f})) \\
\wedge & (\text{regs-address}\,(\text{cv\_decode}\,(rw\text{-},\ regs\text{-}address,\ i\text{-}reg,\ flags,\ pc\text{-}reg)) \\
& = \quad pc\text{-}reg) \\
\wedge & (\text{alu-op}\,(\text{cv\_decode}\,(rw\text{-},\ regs\text{-}address,\ i\text{-}reg,\ flags,\ pc\text{-}reg)) \\
& = \quad \text{op-code}\,(i\text{-}reg)) \\
\wedge & (\text{alu-mpg}\,(\text{cv\_decode}\,(rw\text{-},\ regs\text{-}address,\ i\text{-}reg,\ flags,\ pc\text{-}reg)) \\
& = \quad \text{mpg}\,(\text{cons}\,(\mathbf{f},\ \text{cons}\,(\mathbf{f},\ \text{op-code}\,(i\text{-}reg))))))
\end{array}
$$

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 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'.

```
(DEFN CV_REGB
 (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 F
              (CONS F
               (CONS
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (CONS F (OP-CODE I-REG))))
                (CONS F
                 (APPEND
                  (V_REGB)
                  (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_regb; name this event 'cv_regb-off'.

THEOREM: cv_regb$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\to$ ((rw- (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (strobe- (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (hdack- (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\land$ (we-a-reg (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\land$ (we-b-reg (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-i-reg (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\land$ (we-data-out (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \mathbf{f})$
$\wedge$ (we-addr-out (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
$= \mathbf{f})$
$\wedge$ (we-hold- (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)) $= \mathbf{f})$
$\wedge$ (we-pc-reg (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
$= \mathbf{f})$
$\wedge$ (data-in-select (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
$= \mathbf{f})$
$\wedge$ (dec-addr-out (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
$= \mathbf{f})$
$\wedge$ (select-immediate (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
$= \mathbf{f})$
$\wedge$ (alu-c (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= carry-in-help (cons (c-flag ($flags$),
cons ($\mathbf{f}$, op-code ($i\text{-}reg$)))))
$\wedge$ (alu-zero (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)) $= \mathbf{f})$
$\wedge$ (state (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= V_REGB)
$\wedge$ (we-flags (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= rn-b ($i\text{-}reg$))
$\wedge$ (alu-op (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= op-code ($i\text{-}reg$))
$\wedge$ (alu-mpg (cv_regb (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= mpg (cons (**f**, cons (**f**, op-code ($i\text{-}reg$))))))

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*)
$\rightarrow$ ((rw- (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= store-resultp (store-cc (*i-reg*), *flags*))
$\wedge$ (we-a-reg (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-b-reg (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (we-i-reg (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-pc-reg (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (data-in-select (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)

$\wedge$ (dec-addr-out (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (select-immediate (cv_update (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (state (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_UPDATE)
$\wedge$ (we-flags (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= set-flags (*i-reg*))
$\wedge$ (regs-address (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-b (*i-reg*))
$\wedge$ (alu-op (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (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_READA0
 (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 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*)))

$\wedge$ (select-immediate (cv_reada0 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
$\wedge$ (alu-c (cv_reada0 (*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_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (state (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_READA0)
$\wedge$ (we-flags (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-a (*i-reg*))
$\wedge$ (alu-op (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (alu-mpg (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= mpg (cons (**f**, cons (**f**, op-code (*i-reg*))))))

EVENT: Enable cv_reada0$destructure; name this event 'cv_reada0$destructure-off'.

```
(DEFN CV_READA1
 (RW- REGS-ADDRESS I-REG FLAGS PC-REG)
 (CONS T
  (CONS T
   (CONS T
    (CONS
     (AND* (NOT* (A-IMMEDIATE-P I-REG))
           (OR* (PRE-DEC-P (MODE-A I-REG))
                (POST-INC-P (MODE-A I-REG))))
     (CONS F
      (CONS F
       (CONS F
        (CONS F
         (CONS F
          (CONS F
           (CONS F
            (CONS F
```

```
(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*)
$\rightarrow$ ((rw- (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (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*)))))
$\wedge$ (we-a-reg (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-b-reg (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-i-reg (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \mathbf{f}$)

∧ (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 F
            (CONS F
             (CONS
              (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                   (CONS F (OP-CODE I-REG))))
              (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-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)
$\rightarrow$ ((rw- (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)) = **t**)
$\wedge$ (strobe- (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= **f**)
$\wedge$ (hdack- (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)) = **t**)
$\wedge$ (we-regs (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= **f**)
$\wedge$ (we-a-reg (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= **f**)
$\wedge$ (we-b-reg (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= **t**)
$\wedge$ (we-i-reg (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))
= **f**)
$\wedge$ (we-data-out (cv_reada2 (rw-, $regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$))

$= \mathbf{f})$
$\wedge$ (we-addr-out (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-hold- (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-pc-reg (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (data-in-select (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (dec-addr-out (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (select-immediate (cv_reada2 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \mathbf{f})$
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (state (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_READA2)
$\wedge$ (we-flags (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-b (*i-reg*))
$\wedge$ (alu-op (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (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 F
       (CONS T
        (CONS F
         (CONS F
          (CONS
           (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                (CONS F (OP-CODE I-REG))))
           (CONS F
            (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*)) = **t**)
$\land$ (strobe- (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (hdack- (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-a-reg (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\land$ (we-b-reg (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-i-reg (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-data-out (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-addr-out (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-hold- (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$=$ **f**)

$\wedge$ (we-pc-reg (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)

$\wedge$ (data-in-select (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **t**)

$\wedge$ (dec-addr-out (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)

$\wedge$ (select-immediate (cv_reada3 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$=$ **f**)

$\wedge$ (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*)))))

$\wedge$ (alu-zero (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)

$\wedge$ (state (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ V_READA3)

$\wedge$ (we-flags (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ make-list (**4**, **f**))

$\wedge$ (regs-address (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ *pc-reg*)

$\wedge$ (alu-op (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ op-code (*i-reg*))

$\wedge$ (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_READB0
 (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 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*)
$\rightarrow$ ((rw- (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (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*))))
$\wedge$ (we-b-reg (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-i-reg (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-pc-reg (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \quad \mathbf{f})$
$\wedge$ (data-in-select (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$
$\wedge$ (dec-addr-out (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$
$\wedge$ (select-immediate (cv_readb1 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$=$ a-immediate-p (*i-reg*))
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$
$\wedge$ (state (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ V_READB1)
$\wedge$ (we-flags (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ make-list (**4**, **f**))
$\wedge$ (regs-address (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ rn-a (*i-reg*))
$\wedge$ (alu-op (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ op-code (*i-reg*))
$\wedge$ (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 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*)
$\rightarrow$ ((rw- (cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (hdack- (cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (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*)))))
$\wedge$ (we-a-reg (cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (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'.

```
(DEFN CV_READB3
 (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 T
             (CONS F
              (CONS F
               (CONS
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (CONS F (OP-CODE I-REG))))
                (CONS F
                 (APPEND
                  (V_READB3)
                  (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_readb3; name this event 'cv_readb3-off'.

Theorem: cv_readb3$destructure

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})} \\
\rightarrow & ((\text{rw-}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) = \mathbf{t}) \\
& \quad \land \quad (\text{strobe-}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
& \qquad\quad = \ \mathbf{f}) \\
& \quad \land \quad (\text{hdack-}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) = \mathbf{t}) \\
& \quad \land \quad (\text{we-regs}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
& \qquad\quad = \ \mathbf{f}) \\
& \quad \land \quad (\text{we-a-reg}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
& \qquad\quad = \ \mathbf{f}) \\
& \quad \land \quad (\text{we-b-reg}\,(\text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))
\end{array}
$$

$= \mathbf{t})$
$\wedge$ (we-i-reg (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-data-out (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-addr-out (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-hold- (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-pc-reg (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (data-in-select (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{t})$
$\wedge$ (dec-addr-out (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (select-immediate (cv_readb3 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \mathbf{f})$
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (state (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_READB3)
$\wedge$ (we-flags (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= *pc-reg*)
$\wedge$ (alu-op (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (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*)
$\rightarrow$ ((rw- (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-a-reg (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-b-reg (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (we-i-reg (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \mathbf{f})$

$\wedge$ (we-data-out (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{t})$

$\wedge$ (we-addr-out (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{t})$

$\wedge$ (we-hold- (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$

$\wedge$ (we-pc-reg (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$

$\wedge$ (data-in-select (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$

$\wedge$ (dec-addr-out (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= pre-dec-p (mode-b (*i-reg*)))

$\wedge$ (select-immediate (cv_write0 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \mathbf{f})$

$\wedge$ (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*)))))

$\wedge$ (alu-zero (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$

$\wedge$ (state (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_WRITE0)

$\wedge$ (we-flags (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= set-flags (*i-reg*))

$\wedge$ (regs-address (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-b (*i-reg*))

$\wedge$ (alu-op (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))

$\wedge$ (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 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*)
$\rightarrow$ ((rw- (cv_write1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\wedge$ (strobe- (cv_write1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (hdack- (cv_write1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_write1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$$
\begin{array}{lll}
 & = & \text{or* (pre-dec-p (mode-b (}\textit{i-reg}\text{)),} \\
 & & \quad\text{post-inc-p (mode-b (}\textit{i-reg}\text{))))} \\
\wedge & \multicolumn{2}{l}{\text{(we-a-reg (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-b-reg (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-i-reg (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-data-out (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-addr-out (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-hold- (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(we-pc-reg (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(data-in-select (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(dec-addr-out (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(select-immediate (cv\_write1 (}\textit{rw-},} \\
 & & \qquad\qquad\qquad\qquad\textit{regs-address}, \\
 & & \qquad\qquad\qquad\qquad\textit{i-reg}, \\
 & & \qquad\qquad\qquad\qquad\textit{flags}, \\
 & & \qquad\qquad\qquad\qquad\textit{pc-reg}\text{))} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(alu-c (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{if* (pre-dec-p (mode-b (}\textit{i-reg}\text{)),} \\
 & & \quad\text{carry-in-help (cons (c-flag (}\textit{flags}\text{),} \\
 & & \qquad\qquad\qquad\text{cons (}\mathbf{f},\ \text{ALU-DEC-OP))),} \\
 & & \quad\text{carry-in-help (cons (c-flag (}\textit{flags}\text{),} \\
 & & \qquad\qquad\qquad\text{cons (}\mathbf{f},\ \text{ALU-INC-OP)))))} \\
\wedge & \multicolumn{2}{l}{\text{(alu-zero (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \mathbf{f}) \\
\wedge & \multicolumn{2}{l}{\text{(state (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{V\_WRITE1)} \\
\wedge & \multicolumn{2}{l}{\text{(we-flags (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{make-list (}\mathbf{4},\ \mathbf{f}\text{))} \\
\wedge & \multicolumn{2}{l}{\text{(regs-address (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{rn-b (}\textit{i-reg}\text{))} \\
\wedge & \multicolumn{2}{l}{\text{(alu-op (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{if* (pre-dec-p (mode-b (}\textit{i-reg}\text{)), ALU-DEC-OP, ALU-INC-OP))} \\
\wedge & \multicolumn{2}{l}{\text{(alu-mpg (cv\_write1 (}\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}\text{))}} \\
 & = & \text{if* (pre-dec-p (mode-b (}\textit{i-reg}\text{)),}
\end{array}
$$

mpg (cons (**f**, cons (**t**, ALU-DEC-OP))),
mpg (cons (**f**, cons (**t**, ALU-INC-OP))))))

EVENT: Enable cv_write1$destructure; name this event 'cv_write1$destructure-off'.

```
(DEFN CV_WRITE2
 (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
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (CONS F (OP-CODE I-REG))))
                (CONS F
                 (APPEND
                  (V_WRITE2)
                  (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_write2; name this event 'cv_write2-off'.

THEOREM: cv_write2$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\wedge$ (strobe- (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \ \mathbf{f})$
$\wedge$ (hdack- (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = $\mathbf{t}$)
$\wedge$ (we-regs (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-a-reg (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-b-reg (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-i-reg (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-data-out (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-addr-out (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-hold- (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (we-pc-reg (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (data-in-select (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (dec-addr-out (cv_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (select-immediate (cv_write2 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \ \mathbf{f})$
$\wedge$ (alu-c (cv_write2 (*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_write2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\wedge$ (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**, $\mathbf{f}$))
$\wedge$ (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 ($\mathbf{f}$, cons ($\mathbf{f}$, 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
                (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’.

Theorem: cv_write3$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\wedge$ (strobe- (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (hdack- (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$=$ **f**)
$\wedge$ (we-a-reg (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-b-reg (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-i-reg (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-data-out (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-addr-out (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-hold- (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (we-pc-reg (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (data-in-select (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (dec-addr-out (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (select-immediate (cv_write3 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$=$ **f**)
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ **f**)
$\wedge$ (state (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ V_WRITE3)
$\wedge$ (we-flags (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ make-list (**4**, **f**))
$\wedge$ (regs-address (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ *pc-reg*)
$\wedge$ (alu-op (cv_write3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$=$ op-code (*i-reg*))
$\wedge$ (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 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*)
$\rightarrow$ ((rw- (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (strobe- (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (hdack- (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\land$ (we-a-reg (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\land$ (we-b-reg (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)

$\wedge$ (we-i-reg (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-pc-reg (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (data-in-select (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (dec-addr-out (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (select-immediate (cv_sefa0 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (state (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_SEFA0)
$\wedge$ (we-flags (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-a (*i-reg*))
$\wedge$ (alu-op (cv_sefa0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (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
```

```
(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
              (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_SEFA1)
                (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_sefa1; name this event 'cv_sefa1-off'.

THEOREM: cv_sefa1$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\land$ (strobe- (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\land$ (hdack- (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\land$ (we-regs (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\land$ (we-a-reg (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)

$\wedge$ (we-b-reg (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-i-reg (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-addr-out (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-pc-reg (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (data-in-select (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (dec-addr-out (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (select-immediate (cv_sefa1 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
$\wedge$ (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)))))
$\wedge$ (alu-zero (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (state (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_SEFA1)
$\wedge$ (we-flags (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_sefa1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-a (*i-reg*))
$\wedge$ (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))
$\wedge$ (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 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*))

$= \ \mathbf{f})$
$\land \ $ (we-b-reg (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{t})$
$\land \ $ (we-i-reg (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (we-data-out (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (we-addr-out (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (we-hold- (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (we-pc-reg (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (data-in-select (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (dec-addr-out (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (select-immediate (cv_sefb0 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \ \mathbf{f})$
$\land \ $ (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*)))))
$\land \ $ (alu-zero (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \ \mathbf{f})$
$\land \ $ (state (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_SEFB0)
$\land \ $ (we-flags (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\land \ $ (regs-address (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= rn-b (*i-reg*))
$\land \ $ (alu-op (cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\land \ $ (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'.

```
(DEFN CV_SEFB1
 (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 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_SEFB1)
                  (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_sefb1; name this event 'cv_sefb1-off'.

Theorem: cv_sefb1$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_sefb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_sefb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (hdack- (cv_sefb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)

$\wedge$ (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 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
                 (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**)

∧ (we-regs (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
∧ (we-a-reg (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-b-reg (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-i-reg (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-data-out (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-addr-out (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-hold- (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-pc-reg (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (data-in-select (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (dec-addr-out (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (select-immediate (cv_hold0 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
∧ (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*)))))
∧ (alu-zero (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (state (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_HOLD0)
∧ (we-flags (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
∧ (regs-address (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= *pc-reg*)
∧ (alu-op (cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
∧ (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’.

```
(DEFN CV_HOLD1
 (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 F
               (CONS
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (CONS F (OP-CODE I-REG))))
                (CONS F
                 (APPEND
                  (V_HOLD1)
                  (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_hold1; name this event 'cv_hold1-off'.

THEOREM: cv_hold1$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (strobe- (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (hdack- (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
$\land$ (we-a-reg (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-b-reg (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-i-reg (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \mathbf{f})$
$\wedge$ (we-data-out (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-addr-out (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-hold- (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (we-pc-reg (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (data-in-select (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (dec-addr-out (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (select-immediate (cv_hold1 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \mathbf{f})$
$\wedge$ (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*)))))
$\wedge$ (alu-zero (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \mathbf{f})$
$\wedge$ (state (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_HOLD1)
$\wedge$ (we-flags (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))
$\wedge$ (regs-address (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= *pc-reg*)
$\wedge$ (alu-op (cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))
$\wedge$ (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
```

```
(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
             (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                  (CONS F (OP-CODE I-REG))))
             (CONS F
              (APPEND
               (V_V11010)
               (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_v11010; name this event 'cv_v11010-off'.

Theorem: cv_v11010$destructure
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ ((rw- (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\wedge$ (strobe- (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **t**)
 $\wedge$ (hdack- (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
 $\wedge$ (we-regs (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)
 $\wedge$ (we-a-reg (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)
 $\wedge$ (we-b-reg (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)
 $\wedge$ (we-i-reg (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)
 $\wedge$ (we-data-out (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
 = **f**)
 $\wedge$ (we-addr-out (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \quad$ **f**)
$\wedge \quad$ (we-hold- (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)
$\wedge \quad$ (we-pc-reg (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)
$\wedge \quad$ (data-in-select (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)
$\wedge \quad$ (dec-addr-out (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)
$\wedge \quad$ (select-immediate (cv_v11010 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \quad$ **f**)
$\wedge \quad$ (alu-c (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ carry-in-help (cons (c-flag (*flags*),
cons (**f**, op-code (*i-reg*)))))
$\wedge \quad$ (alu-zero (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ **f**)
$\wedge \quad$ (state (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ v_v11010)
$\wedge \quad$ (we-flags (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ make-list (**4**, **f**))
$\wedge \quad$ (regs-address (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ *pc-reg*)
$\wedge \quad$ (alu-op (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ op-code (*i-reg*))
$\wedge \quad$ (alu-mpg (cv_v11010 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ 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 F
              (CONS F
               (CONS
                (CARRY-IN-HELP (CONS (C-FLAG FLAGS)
                                     (CONS F (OP-CODE I-REG))))
                (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*)
$\rightarrow$ ((rw- (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (strobe- (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\land$ (hdack- (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\land$ (we-regs (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-a-reg (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-b-reg (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-i-reg (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-data-out (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-addr-out (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-hold- (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\land$ (we-pc-reg (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \quad \mathbf{f})$

$\wedge$ (data-in-select (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$

$\wedge$ (dec-addr-out (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$

$\wedge$ (select-immediate (cv_v11011 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
$= \quad \mathbf{f})$

$\wedge$ (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*)))))

$\wedge$ (alu-zero (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad \mathbf{f})$

$\wedge$ (state (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= v_v11011)

$\wedge$ (we-flags (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (**4**, **f**))

$\wedge$ (regs-address (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= *pc-reg*)

$\wedge$ (alu-op (cv_v11011 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= op-code (*i-reg*))

$\wedge$ (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
```

```
(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_RESET0)
       (APPEND
        (MAKE-LIST 4 T)
        (APPEND (MAKE-LIST 4 F)
                (APPEND (ALU-INC-OP)
                        (APPEND (MPG (CONS T (CONS F (ALU-INC-OP))))
                                NIL))))))))))))))))))))))
```

EVENT: Enable cv_reset0; name this event 'cv_reset0-off'.

THEOREM: cv_reset0$destructure
$\mathbf{t} \rightarrow$ ((rw- (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (strobe- (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (hdack- (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-regs (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
$\wedge$ (we-a-reg (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-b-reg (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-i-reg (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-data-out (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (we-addr-out (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (we-hold- (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (we-pc-reg (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
$\wedge$ (data-in-select (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (dec-addr-out (cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
$\wedge$ (select-immediate (cv_reset0 (*rw-*,
regs-address,

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 T
        (CONS F
         (CONS T
          (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_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
t → ((rw- (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (strobe- (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (hdack- (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (we-regs (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **f**)
∧ (we-a-reg (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-b-reg (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-i-reg (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-data-out (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-addr-out (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-hold- (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-pc-reg (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (data-in-select (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (dec-addr-out (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (select-immediate (cv_reset1 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
∧ (alu-c (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= carry-in-help (cons (c-flag (*flags*), cons (**t**, ALU-INC-OP))))
∧ (alu-zero (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)

$\wedge$ (state (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = V_RESET1)
$\wedge$ (we-flags (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = make-list (**4**, **f**))
$\wedge$ (regs-address (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = make-list (**4**, **f**))
$\wedge$ (alu-op (cv_reset1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
  = ALU-INC-OP)
$\wedge$ (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 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*)
→ ((rw- (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (strobe- (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (hdack- (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)) = **t**)
∧ (we-regs (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (we-a-reg (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-b-reg (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-i-reg (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-data-out (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-addr-out (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-hold- (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (we-pc-reg (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (data-in-select (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (dec-addr-out (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **f**)
∧ (select-immediate (cv_reset2 (*rw-*,
regs-address,
i-reg,
flags,
pc-reg))
= **f**)
∧ (alu-c (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= carry-in-help (cons (c-flag (*flags*), cons (**t**, ALU-INC-OP))))
∧ (alu-zero (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= **t**)
∧ (state (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= V_RESET2)
∧ (we-flags (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= make-list (4, **f**))
∧ (regs-address (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$= \quad$ v-inc (*regs-address*))
$\wedge \quad$ (alu-op (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ ALU-INC-OP)
$\wedge \quad$ (alu-mpg (cv_reset2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$= \quad$ mpg (cons (**t**, cons (**f**, ALU-INC-OP)))))

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
             (CONS F
              (CONS F
               (CONS
                (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

$$
\begin{array}{l}
\text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}) \\
\rightarrow \quad ((\text{rw-}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) = \mathbf{t}) \\
\qquad \land \quad (\text{strobe-}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{t}) \\
\qquad \land \quad (\text{hdack-}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) = \mathbf{t}) \\
\qquad \land \quad (\text{we-regs}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-a-reg}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-b-reg}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-i-reg}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-data-out}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-addr-out}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-hold-}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{we-pc-reg}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{data-in-select}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{dec-addr-out}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{select-immediate}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{regs\text{-}address}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{alu-c}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \text{carry-in-help}\,(\text{cons}\,(\text{c-flag}\,(\mathit{flags}), \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{cons}\,(\mathbf{f},\ \text{op-code}\,(\mathit{i\text{-}reg}))))) \\
\qquad \land \quad (\text{alu-zero}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathbf{f}) \\
\qquad \land \quad (\text{state}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \text{v\_v11111}) \\
\qquad \land \quad (\text{we-flags}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \text{make-list}\,(\mathbf{4},\ \mathbf{f})) \\
\qquad \land \quad (\text{regs-address}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \mathit{pc\text{-}reg}) \\
\qquad \land \quad (\text{alu-op}\,(\text{cv\_v11111}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\qquad\qquad = \quad \text{op-code}\,(\mathit{i\text{-}reg}))
\end{array}
$$

$\wedge$ (alu-mpg (cv_v11111 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
= mpg (cons (**f**, cons (**f**, op-code (*i-reg*))))))

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
cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)
$\rightarrow$ (bvp (cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_fetch1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_fetch2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_fetch3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_decode (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_rega (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_reada0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_reada1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_reada2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_reada3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_readb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_readb3 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\wedge$ bvp (cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

$\wedge$ 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 'G0
            '(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$value
(control-let& (*netlist*)
∧ properp (*i-reg*)
∧ (length (*i-reg*) = 32)
∧ properp (*flags*)
∧ (length (*flags*) = 4)
∧ properp (*regs-address*)
∧ (length (*regs-address*) = 4))
→ (dual-eval (0,
'**control-let**,
append (*i-reg*, append (*flags*, *regs-address*)),
state,
netlist)
= f$control-let (*i-reg*, *flags*, *regs-address*))

EVENT: Enable control-let$value; name this event 'control-let$value-off'.

THEOREM: f$control-let=control-let
(bvp (*i-reg*)
∧ (length (*i-reg*) = 32)
∧ bvp (*flags*)
∧ (length (*flags*) = 4)
∧ bvp (*regs-address*)
∧ (length (*regs-address*) = 4))
→ (f$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)),
```

nth (4, *decoded-state*)),
nth (20, *decoded-state*),
f-nand (f-nand (*side-effect-b*, nth (21, *decoded-state*)),
f-nand4 (*side-effect-b*,
f-not (*side-effect-a*),
f-not (f-or (*store*, *set-some-flags*)),
nth (4, *decoded-state*))),
f-nand (f-not (nth (22, *decoded-state*)),
f-nand (f-and (*side-effect-b*, *unary*), nth (7, *decoded-state*))),
f-nand (f-nand (f-not (*hold-*), nth (24, *decoded-state*)),
f-nand (f-not (*hold-*), nth (1, *decoded-state*))),
f-and (*hold-*, nth (24, *decoded-state*)),
vss,
vss,
f-nand3 (f-not (nth (31, *decoded-state*)),
f-not (nth (27, *decoded-state*)),
f-not (nth (26, *decoded-state*))),
nth (28, *decoded-state*),
f-nand (f-nand (f-not (*all-t-regs-address*), nth (30, *decoded-state*)),
f-not (nth (29, *decoded-state*))),
vss)

EVENT: Enable f$next-state; name this event 'f$next-state-off'.

THEOREM: f$next-state=next-state
(bvp (*decoded-state*)
∧ (length (*decoded-state*) = 32)
∧ boolp (*store*)
∧ boolp (*set-some-flags*)
∧ boolp (*unary*)
∧ boolp (*direct-a*)
∧ boolp (*direct-b*)
∧ boolp (*side-effect-a*)
∧ boolp (*side-effect-b*)
∧ boolp (*all-t-regs-address*)
∧ boolp (*dtack-*)
∧ boolp (*hold-*))
→ (f$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-)
    =  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,

direct-a,
direct-b,
side-effect-a,
side-effect-b,
all-t-regs-address,
dtack-,
hold-))
= 32)
∧ ((bvp (*decoded-state*) ∧ (length (*decoded-state*) = 32))
→ bvp (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-)))

THEOREM: properp-length-f$next-state
(length (f$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-))
= 32)
∧ properp (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-))

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{array}{ll}
\wedge & \mathrm{bvp}\,(\mathit{pc\text{-}reg}) \\
\wedge & (\mathrm{length}\,(\mathit{pc\text{-}reg}) = \mathtt{4}) \\
\wedge & \mathrm{bvp}\,(\mathit{regs\text{-}address}) \\
\wedge & (\mathrm{length}\,(\mathit{regs\text{-}address}) = \mathtt{4}) \\
\wedge & \mathrm{bvp}\,(\mathit{set\text{-}flags}) \\
\wedge & (\mathrm{length}\,(\mathit{set\text{-}flags}) = \mathtt{4}) \\
\wedge & \mathrm{boolp}\,(\mathit{store}) \\
\wedge & \mathrm{boolp}\,(c) \\
\wedge & \mathrm{boolp}\,(\mathit{a\text{-}immediate\text{-}p}) \\
\wedge & \mathrm{boolp}\,(\mathit{rw\text{-}}) \\
\wedge & \mathrm{boolp}\,(\mathit{direct\text{-}a}) \\
\wedge & \mathrm{boolp}\,(\mathit{side\text{-}effect\text{-}a}) \\
\wedge & \mathrm{boolp}\,(\mathit{side\text{-}effect\text{-}b}) \\
\wedge & \mathrm{boolp}\,(\mathit{pre\text{-}dec\text{-}a}) \\
\wedge & \mathrm{boolp}\,(\mathit{pre\text{-}dec\text{-}b})) \\
\to & (\mathrm{f\$cv}\,(\mathit{decoded\text{-}state}, \\
& \qquad \mathit{rn\text{-}a}, \\
& \qquad \mathit{rn\text{-}b}, \\
& \qquad \mathit{op\text{-}code}, \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \mathit{regs\text{-}address}, \\
& \qquad \mathit{set\text{-}flags}, \\
& \qquad \mathit{store}, \\
& \qquad c, \\
& \qquad \mathit{a\text{-}immediate\text{-}p}, \\
& \qquad \mathit{rw\text{-}}, \\
& \qquad \mathit{direct\text{-}a}, \\
& \qquad \mathit{side\text{-}effect\text{-}a}, \\
& \qquad \mathit{side\text{-}effect\text{-}b}, \\
& \qquad \mathit{pre\text{-}dec\text{-}a}, \\
& \qquad \mathit{pre\text{-}dec\text{-}b}) \\
& \quad = \ \mathrm{cv}\,(\mathit{decoded\text{-}state}, \\
& \qquad\quad \mathit{rn\text{-}a}, \\
& \qquad\quad \mathit{rn\text{-}b}, \\
& \qquad\quad \mathit{op\text{-}code}, \\
& \qquad\quad \mathit{pc\text{-}reg}, \\
& \qquad\quad \mathit{regs\text{-}address}, \\
& \qquad\quad \mathit{set\text{-}flags}, \\
& \qquad\quad \mathit{store}, \\
& \qquad\quad c, \\
& \qquad\quad \mathit{a\text{-}immediate\text{-}p}, \\
& \qquad\quad \mathit{rw\text{-}}, \\
& \qquad\quad \mathit{direct\text{-}a}, \\
& \qquad\quad \mathit{side\text{-}effect\text{-}a},
\end{array}
$$

side-effect-b,
pre-dec-a,
pre-dec-b))

THEOREM: bvp-length-cv

$$
\begin{array}{l}
(((\text{length}\,(\textit{set-flags}) = \mathtt{4}) \\
\quad\land\quad (\text{length}\,(\textit{rn-a}) = \mathtt{4}) \\
\quad\land\quad (\text{length}\,(\textit{rn-b}) = \mathtt{4}) \\
\quad\land\quad (\text{length}\,(\textit{op-code}) = \mathtt{4})) \\
\quad\to\quad (\text{length}\,(\text{cv}\,(\textit{decoded-state}, \\
\qquad\qquad\qquad \textit{rn-a}, \\
\qquad\qquad\qquad \textit{rn-b}, \\
\qquad\qquad\qquad \textit{op-code}, \\
\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad \textit{regs-address}, \\
\qquad\qquad\qquad \textit{set-flags}, \\
\qquad\qquad\qquad \textit{store}, \\
\qquad\qquad\qquad \textit{c}, \\
\qquad\qquad\qquad \textit{a-immediate-p}, \\
\qquad\qquad\qquad \textit{rw-}, \\
\qquad\qquad\qquad \textit{direct-a}, \\
\qquad\qquad\qquad \textit{side-effect-a}, \\
\qquad\qquad\qquad \textit{side-effect-b}, \\
\qquad\qquad\qquad \textit{pre-dec-a}, \\
\qquad\qquad\qquad \textit{pre-dec-b})) \\
\qquad = \quad \mathtt{40})) \\
\land\quad ((\text{bvp}\,(\textit{decoded-state}) \land (\text{length}\,(\textit{decoded-state}) = \mathtt{32})) \\
\qquad \to\quad \text{bvp}\,(\text{cv}\,(\textit{decoded-state}, \\
\qquad\qquad\qquad \textit{rn-a}, \\
\qquad\qquad\qquad \textit{rn-b}, \\
\qquad\qquad\qquad \textit{op-code}, \\
\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad \textit{regs-address}, \\
\qquad\qquad\qquad \textit{set-flags}, \\
\qquad\qquad\qquad \textit{store}, \\
\qquad\qquad\qquad \textit{c}, \\
\qquad\qquad\qquad \textit{a-immediate-p}, \\
\qquad\qquad\qquad \textit{rw-}, \\
\qquad\qquad\qquad \textit{direct-a}, \\
\qquad\qquad\qquad \textit{side-effect-a}, \\
\qquad\qquad\qquad \textit{side-effect-b}, \\
\qquad\qquad\qquad \textit{pre-dec-a}, \\
\qquad\qquad\qquad \textit{pre-dec-b})))
\end{array}
$$

THEOREM: properp-length-f$cv

$$
\begin{array}{l}
(((\text{length}\,(\textit{set-flags}) = \texttt{4}) \\
\quad \land \quad (\text{length}\,(\textit{rn-a}) = \texttt{4}) \\
\quad \land \quad (\text{length}\,(\textit{rn-b}) = \texttt{4}) \\
\quad \land \quad (\text{length}\,(\textit{op-code}) = \texttt{4})) \\
\quad \to \quad (\text{length}\,(\text{f\$cv}\,(\textit{decoded-state}, \\
\qquad\qquad\qquad \textit{rn-a}, \\
\qquad\qquad\qquad \textit{rn-b}, \\
\qquad\qquad\qquad \textit{op-code}, \\
\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad \textit{regs-address}, \\
\qquad\qquad\qquad \textit{set-flags}, \\
\qquad\qquad\qquad \textit{store}, \\
\qquad\qquad\qquad \textit{c}, \\
\qquad\qquad\qquad \textit{a-immediate-p}, \\
\qquad\qquad\qquad \textit{rw-}, \\
\qquad\qquad\qquad \textit{direct-a}, \\
\qquad\qquad\qquad \textit{side-effect-a}, \\
\qquad\qquad\qquad \textit{side-effect-b}, \\
\qquad\qquad\qquad \textit{pre-dec-a}, \\
\qquad\qquad\qquad \textit{pre-dec-b})) \\
\qquad = \quad \texttt{40})) \\
\land \quad ((\text{properp}\,(\textit{decoded-state}) \land (\text{length}\,(\textit{decoded-state}) = \texttt{32})) \\
\qquad \to \quad \text{properp}\,(\text{f\$cv}\,(\textit{decoded-state}, \\
\qquad\qquad\qquad\quad \textit{rn-a}, \\
\qquad\qquad\qquad\quad \textit{rn-b}, \\
\qquad\qquad\qquad\quad \textit{op-code}, \\
\qquad\qquad\qquad\quad \textit{pc-reg}, \\
\qquad\qquad\qquad\quad \textit{regs-address}, \\
\qquad\qquad\qquad\quad \textit{set-flags}, \\
\qquad\qquad\qquad\quad \textit{store}, \\
\qquad\qquad\qquad\quad \textit{c}, \\
\qquad\qquad\qquad\quad \textit{a-immediate-p}, \\
\qquad\qquad\qquad\quad \textit{rw-}, \\
\qquad\qquad\qquad\quad \textit{direct-a}, \\
\qquad\qquad\qquad\quad \textit{side-effect-a}, \\
\qquad\qquad\qquad\quad \textit{side-effect-b}, \\
\qquad\qquad\qquad\quad \textit{pre-dec-a}, \\
\qquad\qquad\qquad\quad \textit{pre-dec-b})))
\end{array}
$$

DEFINITION:
CV*

$$
\begin{array}{l}
= \quad \text{cons}\,(\texttt{'cv}, \\
\qquad\quad \text{cons}\,(\text{append8}\,(\text{indices}\,(\texttt{'decoded-state}, \texttt{0}, \texttt{32}), \\
\qquad\qquad\qquad\qquad\quad \text{indices}\,(\texttt{'rn-a}, \texttt{0}, \texttt{4}),
\end{array}
$$

```
            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*))

```
     ∧  b-nor3& (delete-module ('cv, netlist))
     ∧  b-nor4& (delete-module ('cv, netlist))
     ∧  b-nor5& (delete-module ('cv, netlist))
     ∧  b-nor6& (delete-module ('cv, netlist))
     ∧  b-nor8& (delete-module ('cv, netlist))
     ∧  b-and& (delete-module ('cv, netlist))
     ∧  select-op-code& (delete-module ('cv, netlist))
     ∧  tv-if& (delete-module ('cv, netlist), make-tree (4))
     ∧  carry-in-help& (delete-module ('cv, netlist))
     ∧  encode-32& (delete-module ('cv, netlist))
     ∧  fanout-4& (delete-module ('cv, netlist))
     ∧  v-inc4& (delete-module ('cv, netlist))
     ∧  v-if-f-4& (delete-module ('cv, netlist))
     ∧  mpg& (delete-module ('cv, netlist)))
```

EVENT: Enable cv&; name this event 'cv&-off'.

DEFINITION:

```
CV$NETLIST
=  cons (CV*,
        ((((ID$NETLIST ∪ B-NOT$NETLIST)
           ∪  (B-OR$NETLIST ∪ (B-OR3$NETLIST ∪ B-OR4$NETLIST)))
          ∪  ((B-NAND$NETLIST ∪ B-NAND3$NETLIST)
              ∪  (B-NOR$NETLIST ∪ B-NOR3$NETLIST)))
         ∪  (((B-NOR4$NETLIST ∪ B-NOR5$NETLIST)
              ∪  (B-NOR6$NETLIST ∪ B-NOR8$NETLIST))
             ∪  ((B-AND$NETLIST ∪ SELECT-OP-CODE$NETLIST)
                 ∪  (tv-if$netlist (make-tree (4))
                     ∪  CARRY-IN-HELP$NETLIST))))
        ∪  (((ENCODE-32$NETLIST ∪ FANOUT-4$NETLIST)
             ∪  (V-INC4$NETLIST ∪ V-IF-F-4$NETLIST))
            ∪  (MPG$NETLIST ∪ B-NAND5$NETLIST)))
```

THEOREM: check-cv$netlist

cv& (CV$NETLIST)

THEOREM: cv$value

```
(cv& (netlist)
 ∧  properp (decoded-state)
 ∧  (length (decoded-state) = 32)
 ∧  properp (rn-a)
 ∧  (length (rn-a) = 4)
 ∧  properp (rn-b)
 ∧  (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

```
(length (next-cntl-state (reset-,
                          dtack-,
                          hold-,
                          rw-,
                          state,
                          i-reg,
                          flags,
                          pc-reg,
                          regs-address))
 =  40)
∧   bvp (next-cntl-state (reset-,
                          dtack-,
                          hold-,
                          rw-,
                          state,
                          i-reg,
                          flags,
                          pc-reg,
                          regs-address))
```

DEFINITION:

```
f$next-cntl-state (reset-,
                   dtack-,
                   hold-,
                   rw-,
                   state,
                   i-reg,
                   flags,
                   pc-reg,
                   regs-address)
=   f$cv (f$next-state (f$decode-5 (fv-or (make-list (5, f-not (reset-)), state)),
                        cadr (f$control-let (i-reg, flags, regs-address)),
                        caddr (f$control-let (i-reg, flags, regs-address)),
                        caddddddr (f$control-let (i-reg, flags, regs-address)),
                        cadddr (f$control-let (i-reg, flags, regs-address)),
                        caddddr (f$control-let (i-reg, flags, regs-address)),
                        caddddddddddr (f$control-let (i-reg,
                                                      flags,
                                                      regs-address)),
                        cadddddddddddddr (f$control-let (i-reg,
                                                         flags,
                                                         regs-address)),
                        caddddddddddr (f$control-let (i-reg, flags, regs-address)),
```

dtack-,
hold-),
rn-a (*i-reg*),
rn-b (*i-reg*),
op-code (*i-reg*),
pc-reg,
regs-address,
set-flags (*i-reg*),
cadr (f$control-let (*i-reg*, *flags*, *regs-address*)),
caddddddddr (f$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*)),
caddddddddddr (f$control-let (*i-reg*, *flags*, *regs-address*)),
cadddddddddddr (f$control-let (*i-reg*, *flags*, *regs-address*)),
caddddddr (f$control-let (*i-reg*, *flags*, *regs-address*)),
cadddddddr (f$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)
∧ 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) ∧ bvp (*state*))
→ ((v-or (list (**f**, **f**, **f**, **f**, **f**), *state*) = *state*)

$\wedge$ (v-or (list (**t**, **t**, **t**, **t**, **t**), *state*) = list (**t**, **t**, **t**, **t**, **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
(boolp (*reset-*)
$\wedge$ boolp (*dtack-*)
$\wedge$ boolp (*hold-*)
$\wedge$ boolp (*rw-*)
$\wedge$ bvp (*state*)
$\wedge$ (length (*state*) = 5)
$\wedge$ bvp (*i-reg*)
$\wedge$ (length (*i-reg*) = 32)
$\wedge$ bvp (*flags*)
$\wedge$ (length (*flags*) = 4)
$\wedge$ bvp (*pc-reg*)
$\wedge$ (length (*pc-reg*) = 4)
$\wedge$ bvp (*regs-address*)
$\wedge$ (length (*regs-address*) = 4))
$\rightarrow$ (f$next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
state,
i-reg,
flags,
pc-reg,
regs-address)
= next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
state,
i-reg,
flags,
pc-reg,
regs-address))

DEFINITION:
NEXT-CNTL-STATE*
= cons (`'next-cntl-state`,
cons (append6 (`'(reset- dtack- hold- rw-)`,
indices (`'state`, `0`, `5`),

```
        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

```
 (next-cntl-state& (netlist)
 ∧   properp (state)
 ∧   (length (state) = 5)
 ∧   properp (i-reg)
 ∧   (length (i-reg) = 32)
 ∧   properp (flags)
 ∧   (length (flags) = 4)
 ∧   properp (pc-reg)
 ∧   (length (pc-reg) = 4)
 ∧   properp (regs-address)
 ∧   (length (regs-address) = 4))
 →   (dual-eval (0,
                 'next-cntl-state,
                 cons (reset-,
                       cons (dtack-,
                             cons (hold-,
                                   cons (rw-,
                                         append (state,
                                                 append (i-reg,
                                                         append (flags,
                                                                 append (pc-reg,
                                                                         regs-address)))))))),
                 dual-eval-state,
                 netlist)
      =   f$next-cntl-state (reset-,
                             dtack-,
                             hold-,
                             rw-,
                             state,
                             i-reg,
                             flags,
                             pc-reg,
                             regs-address))
```

EVENT: Enable next-cntl-state$value; name this event 'next-cntl-state$value-off'.

THEOREM: next-cntl-state$fetch0

```
 ((reset- = t) ∧ cv-hyps (rw-, regs-address, i-reg, flags, pc-reg))
 →   (next-cntl-state (reset-,
                       dtack-,
                       hold-,
                       rw-,
```

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad \textsc{v\_fetch0},\\
\qquad\qquad\qquad\qquad \textit{i-reg},\\
\qquad\qquad\qquad\qquad \textit{flags},\\
\qquad\qquad\qquad\qquad \textit{pc-reg},\\
\qquad\qquad\qquad\qquad \textit{regs-address})\\
\quad = \quad \text{cv\_fetch1}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}))
\end{array}
$$

THEOREM: next-cntl-state$fetch1

$$
\begin{array}{l}
((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}))\\
\rightarrow \quad (\text{next-cntl-state}\,(\textit{reset-},\\
\qquad\qquad\qquad\qquad \textit{dtack-},\\
\qquad\qquad\qquad\qquad \textit{hold-},\\
\qquad\qquad\qquad\qquad \textit{rw-},\\
\qquad\qquad\qquad\qquad \textsc{v\_fetch1},\\
\qquad\qquad\qquad\qquad \textit{i-reg},\\
\qquad\qquad\qquad\qquad \textit{flags},\\
\qquad\qquad\qquad\qquad \textit{pc-reg},\\
\qquad\qquad\qquad\qquad \textit{regs-address})\\
\quad = \quad \text{if*}\,(\textit{hold-},\\
\qquad\qquad \text{cv\_fetch2}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}),\\
\qquad\qquad \text{cv\_hold0}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg})))
\end{array}
$$

THEOREM: next-cntl-state$fetch2

$$
\begin{array}{l}
((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}))\\
\rightarrow \quad (\text{next-cntl-state}\,(\textit{reset-},\\
\qquad\qquad\qquad\qquad \textit{dtack-},\\
\qquad\qquad\qquad\qquad \textit{hold-},\\
\qquad\qquad\qquad\qquad \textit{rw-},\\
\qquad\qquad\qquad\qquad \textsc{v\_fetch2},\\
\qquad\qquad\qquad\qquad \textit{i-reg},\\
\qquad\qquad\qquad\qquad \textit{flags},\\
\qquad\qquad\qquad\qquad \textit{pc-reg},\\
\qquad\qquad\qquad\qquad \textit{regs-address})\\
\quad = \quad \text{cv\_fetch3}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}))
\end{array}
$$

THEOREM: next-cntl-state$fetch3

$$
\begin{array}{l}
((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}\,(\textit{rw-},\ \textit{regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}))\\
\rightarrow \quad (\text{next-cntl-state}\,(\textit{reset-},\\
\qquad\qquad\qquad\qquad \textit{dtack-},\\
\qquad\qquad\qquad\qquad \textit{hold-},\\
\qquad\qquad\qquad\qquad \textit{rw-},\\
\qquad\qquad\qquad\qquad \textsc{v\_fetch3},\\
\qquad\qquad\qquad\qquad \textit{i-reg},\\
\qquad\qquad\qquad\qquad \textit{flags},\\
\qquad\qquad\qquad\qquad \textit{pc-reg},
\end{array}
$$

```
                      regs-address)
    =   if* (dtack-,
            cv_fetch3 (rw-, regs-address, i-reg, flags, pc-reg),
            cv_decode (rw-, regs-address, i-reg, flags, pc-reg)))
```

THEOREM: next-cntl-state$decode

```
 ((reset- = t) ∧ cv-hyps (rw-, regs-address, i-reg, flags, pc-reg))
 →   (next-cntl-state (reset-,
                       dtack-,
                       hold-,
                       rw-,
                       V_DECODE,
                       i-reg,
                       flags,
                       pc-reg,
                       regs-address)
    =   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 (rw-, regs-address, i-reg, flags, pc-reg),
                    cv_readb0 (rw-, regs-address, i-reg, flags, pc-reg)),
                cv_reada0 (rw-, regs-address, 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 (rw-, regs-address, i-reg, flags, pc-reg),
                if* (or* (pre-dec-p (mode-b (i-reg)),
                         post-inc-p (mode-b (i-reg))),
                    cv_sefb0 (rw-, regs-address, i-reg, flags, pc-reg),
                    cv_fetch0 (rw-, regs-address, i-reg, flags, pc-reg)))))
```

THEOREM: next-cntl-state$rega

```
 ((reset- = t) ∧ cv-hyps (rw-, regs-address, i-reg, flags, pc-reg))
 →   (next-cntl-state (reset-,
                       dtack-,
                       hold-,
                       rw-,
                       V_REGA,
                       i-reg,
                       flags,
                       pc-reg,
                       regs-address)
```

$=$ if* (reg-direct-p (mode-b (*i-reg*)),
  if* (unary-op-code-p (op-code (*i-reg*)),
    cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
    cv_regb (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)),
  if* (store-resultp (store-cc (*i-reg*), *flags*),
    cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
    cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))))

THEOREM: next-cntl-state$regb
((*reset-* $=$ **t**) $\wedge$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_REGB,
i-reg,
flags,
pc-reg,
regs-address)
$=$ cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$update
((*reset-* $=$ **t**) $\wedge$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_UPDATE,
i-reg,
flags,
pc-reg,
regs-address)
$=$ 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 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
  cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)))

THEOREM: next-cntl-state$reada0
((*reset-* $=$ **t**) $\wedge$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_READA0,

i-reg,
$flags$,
pc-reg,
$regs$-$address$)
= cv_reada1 (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))

THEOREM: next-cntl-state$reada1
(($reset$- = **t**) ∧ cv-hyps (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))
→ (next-cntl-state ($reset$-,
$dtack$-,
$hold$-,
rw-,
V_READA1,
i-reg,
$flags$,
pc-reg,
$regs$-$address$)
= cv_reada2 (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))

THEOREM: next-cntl-state$reada2
(($reset$- = **t**) ∧ cv-hyps (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))
→ (next-cntl-state ($reset$-,
$dtack$-,
$hold$-,
rw-,
V_READA2,
i-reg,
$flags$,
pc-reg,
$regs$-$address$)
= cv_reada3 (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))

THEOREM: next-cntl-state$reada3
(($reset$- = **t**) ∧ cv-hyps (rw-, $regs$-$address$, i-reg, $flags$, pc-reg))
→ (next-cntl-state ($reset$-,
$dtack$-,
$hold$-,
rw-,
V_READA3,
i-reg,
$flags$,
pc-reg,
$regs$-$address$)
= if* ($dtack$-,
cv_reada3 (rw-, $regs$-$address$, i-reg, $flags$, pc-reg),

if* (reg-direct-p (mode-b (*i-reg*)),
    cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
    if* (unary-op-code-p (op-code (*i-reg*)),
        if* (store-resultp (store-cc (*i-reg*), *flags*),
            cv_write0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
            cv_update (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)),
        cv_readb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)))))

THEOREM: next-cntl-state$readb0
$((reset\text{-} = \mathbf{t}) \land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_READB0,
i-reg,
flags,
pc-reg,
regs-address)
= cv_readb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$readb1
$((reset\text{-} = \mathbf{t}) \land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_READB1,
i-reg,
flags,
pc-reg,
regs-address)
= cv_readb2 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$readb2
$((reset\text{-} = \mathbf{t}) \land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_READB2,
i-reg,
flags,
pc-reg,
regs-address)

$= \quad \text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))$

Theorem: next-cntl-state$readb3

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to \quad (\text{next-cntl-state}\,(\mathit{reset\text{-}},\ \mathit{dtack\text{-}},\ \mathit{hold\text{-}},\ \mathit{rw\text{-}},\ \textsc{v\_readb3},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg},\ \mathit{regs\text{-}address}) \\
\qquad = \quad \text{if*}\,(\mathit{dtack\text{-}}, \\
\qquad\qquad \text{cv\_readb3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}), \\
\qquad\qquad \text{if*}\,(\text{store-resultp}\,(\text{store-cc}\,(\mathit{i\text{-}reg}),\ \mathit{flags}), \\
\qquad\qquad\qquad \text{cv\_write0}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}), \\
\qquad\qquad\qquad \text{cv\_update}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))))
\end{array}
$$

Theorem: next-cntl-state$write0

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to \quad (\text{next-cntl-state}\,(\mathit{reset\text{-}},\ \mathit{dtack\text{-}},\ \mathit{hold\text{-}},\ \mathit{rw\text{-}},\ \textsc{v\_write0},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg},\ \mathit{regs\text{-}address}) \\
\qquad = \quad \text{cv\_write1}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))
\end{array}
$$

Theorem: next-cntl-state$write1

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to \quad (\text{next-cntl-state}\,(\mathit{reset\text{-}},\ \mathit{dtack\text{-}},\ \mathit{hold\text{-}},\ \mathit{rw\text{-}},\ \textsc{v\_write1},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg},\ \mathit{regs\text{-}address}) \\
\qquad = \quad \text{cv\_write2}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))
\end{array}
$$

THEOREM: next-cntl-state\$write2

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to\quad (\text{next-cntl-state}\,(\mathit{reset\text{-}}, \mathit{dtack\text{-}}, \mathit{hold\text{-}}, \mathit{rw\text{-}}, \text{V\_WRITE2}, \mathit{i\text{-}reg}, \mathit{flags}, \mathit{pc\text{-}reg}, \mathit{regs\text{-}address}) \\
\qquad = \quad \text{cv\_write3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))
\end{array}
$$

THEOREM: next-cntl-state\$write3

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to\quad (\text{next-cntl-state}\,(\mathit{reset\text{-}}, \mathit{dtack\text{-}}, \mathit{hold\text{-}}, \mathit{rw\text{-}}, \text{V\_WRITE3}, \mathit{i\text{-}reg}, \mathit{flags}, \mathit{pc\text{-}reg}, \mathit{regs\text{-}address}) \\
\qquad = \quad \text{if*}\,(\mathit{dtack\text{-}}, \\
\qquad\qquad\quad \text{cv\_write3}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}), \\
\qquad\qquad\quad \text{cv\_fetch0}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})))
\end{array}
$$

THEOREM: next-cntl-state\$sefa0

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to\quad (\text{next-cntl-state}\,(\mathit{reset\text{-}}, \mathit{dtack\text{-}}, \mathit{hold\text{-}}, \mathit{rw\text{-}}, \text{V\_SEFA0}, \mathit{i\text{-}reg}, \mathit{flags}, \mathit{pc\text{-}reg}, \mathit{regs\text{-}address}) \\
\qquad = \quad \text{cv\_sefa1}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}))
\end{array}
$$

THEOREM: next-cntl-state\$sefa1

$$
\begin{array}{l}
((\mathit{reset\text{-}} = \mathbf{t}) \land \text{cv-hyps}\,(\mathit{rw\text{-}},\ \mathit{regs\text{-}address},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})) \\
\to\quad (\text{next-cntl-state}\,(\mathit{reset\text{-}}, \\
\qquad\qquad\qquad\qquad \mathit{dtack\text{-}},
\end{array}
$$

hold-,
rw-,
V_SEFA1,
i-reg,
flags,
pc-reg,
regs-address)
= if* (or* (pre-dec-p (mode-b (*i-reg*)), post-inc-p (mode-b (*i-reg*))),
cv_sefb0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)))

THEOREM: next-cntl-state$sefb0
((*reset-* = **t**) $\land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_SEFB0,
i-reg,
flags,
pc-reg,
regs-address)
= cv_sefb1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$sefb1
((*reset-* = **t**) $\land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_SEFB1,
i-reg,
flags,
pc-reg,
regs-address)
= cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$hold0
((*reset-* = **t**) $\land$ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
$\rightarrow$ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_HOLD0,
i-reg,

$\quad\quad\quad\quad$ *flags*,
$\quad\quad\quad\quad$ *pc-reg*,
$\quad\quad\quad\quad$ *regs-address*)

= if* (*hold-*,
$\quad$ cv_hold1 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*),
$\quad$ cv_hold0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*)))

THEOREM: next-cntl-state$hold1
((*reset-* = **t**) ∧ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
→ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_HOLD1,
i-reg,
flags,
pc-reg,
regs-address)
= cv_fetch0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$v11010
((*reset-* = **t**) ∧ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
→ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_V11010,
i-reg,
flags,
pc-reg,
regs-address)
= cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$v11011
((*reset-* = **t**) ∧ cv-hyps (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))
→ (next-cntl-state (*reset-*,
dtack-,
hold-,
rw-,
V_V11011,
i-reg,
flags,
pc-reg,
regs-address)
= cv_reset0 (*rw-*, *regs-address*, *i-reg*, *flags*, *pc-reg*))

THEOREM: next-cntl-state$reset0

$$((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$
$$\rightarrow \quad (\text{next-cntl-state}(\textit{reset-}, \textit{dtack-}, \textit{hold-}, \textit{rw-}, \text{V\_RESET0}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}, \textit{regs-address})$$
$$= \quad \text{cv\_reset1}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$

THEOREM: next-cntl-state$reset1

$$((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$
$$\rightarrow \quad (\text{next-cntl-state}(\textit{reset-}, \textit{dtack-}, \textit{hold-}, \textit{rw-}, \text{V\_RESET1}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}, \textit{regs-address})$$
$$= \quad \text{cv\_reset2}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$

THEOREM: next-cntl-state$reset2

$$((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$
$$\rightarrow \quad (\text{next-cntl-state}(\textit{reset-}, \textit{dtack-}, \textit{hold-}, \textit{rw-}, \text{V\_RESET2}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}, \textit{regs-address})$$
$$= \quad \text{if*}(\textit{regs-address} = \text{list}(\mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}), \text{cv\_fetch0}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}), \text{cv\_reset2}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg})))$$

THEOREM: next-cntl-state$v11111

$$((\textit{reset-} = \mathbf{t}) \land \text{cv-hyps}(\textit{rw-}, \textit{regs-address}, \textit{i-reg}, \textit{flags}, \textit{pc-reg}))$$
$$\rightarrow \quad (\text{next-cntl-state}(\textit{reset-}, \textit{dtack-},$$

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*, cadddr (*regs*)) **then** caddr (*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 (cadddr (*regs*), car (*regs*), caddr (*regs*))
else car (*regs*) **endif**,
we,
if *we* **then** *data*
else caddr (*regs*) **endif**,
if *we* **then** *address*
else cadddr (*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,

$data$)
= **if** c **then** write-regs (a, $address$, $regs$, $data$)
else write-regs (b, $address$, $regs$, $data$) **endif**

THEOREM: write-regs-ok
(memory-okp (n, m, car (write-regs (we, $address$, $regs$, $data$)))
= **if** cadr ($regs$)
then memory-okp (n, m, write-mem (cadddr ($regs$), car ($regs$), caddr ($regs$)))
else memory-okp (n, m, car ($regs$)) **endif**)
∧ (boolp (cadr (write-regs (we, $address$, $regs$, $data$))) = boolp (we))
∧ (bvp (caddr (write-regs (we, $address$, $regs$, $data$)))
= **if** we **then** bvp ($data$)
else bvp (caddr ($regs$)) **endif**)
∧ (length (caddr (write-regs (we, $address$, $regs$, $data$)))
= **if** we **then** length ($data$)
else length (caddr ($regs$)) **endif**)
∧ (bvp (cadddr (write-regs (we, $address$, $regs$, $data$)))
= **if** we **then** bvp ($address$)
else bvp (cadddr ($regs$)) **endif**)
∧ (length (cadddr (write-regs (we, $address$, $regs$, $data$)))
= **if** we **then** length ($address$)
else length (cadddr ($regs$)) **endif**)

THEOREM: read-regs-write-regs-f
(all-ramp-mem (length ($addr$), car ($regs$))
∧ (length ($addr$) = length (cadddr ($regs$)))
∧ bvp ($addr$))
→ (read-regs ($addr$, write-regs (**f**, $address$, $regs$, $value$))
= read-regs ($addr$, $regs$))

THEOREM: all-ramp-mem-after-write-regs
(all-ramp-mem (n, car ($regs$)) ∧ (length (cadddr ($regs$)) = n))
→ all-ramp-mem (n, car (write-regs (we, $address$, $regs$, $data$)))

THEOREM: read-regs=read-mem
(¬ cadr ($regs$))
→ (read-regs ($v\text{-}addr$, $regs$) = read-mem ($v\text{-}addr$, car ($regs$)))

THEOREM: read-regs=read-mem-write-mem
(all-ramp-mem (length ($v\text{-}addr1$), car ($regs$))
∧ (length ($v\text{-}addr1$) = length (cadddr ($regs$)))
∧ cadr ($regs$))
→ (read-regs ($v\text{-}addr1$, $regs$)
= read-mem ($v\text{-}addr1$,
write-mem (cadddr ($regs$), car ($regs$), caddr ($regs$))))

THEOREM: bvp-length-read-regs-32

```
 (memory-okp (length (v-addr), 32, car (regs))
  ∧   bvp (caddr (regs))
  ∧   (length (caddr (regs)) = 32))
 →   (bvp (read-regs (v-addr, regs))
        ∧   (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-)
                                    fd1s
                                    (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)))
∧   (cadddr (REGFILE*)
      =   list ('(we-latch
                  (we-dp-ram we-dp-ram-)
                  fd1s
                  (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*)
    ∧ fd1s& (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$NETLIST
= cons (REGFILE*,
(((FD1S$NETLIST ∪ we-reg$netlist (4))
∪ (we-reg$netlist (32) ∪ RAM-ENABLE-CIRCUIT$NETLIST))
∪ (DP-RAM-16X32$NETLIST
∪ (v-equal$netlist (4) ∪ B-AND3$NETLIST)))
∪ (tv-if$netlist (make-tree (32)) ∪ ID$NETLIST))

THEOREM: check-regfile$netlist
regfile& (REGFILE$NETLIST)

DEFINITION:
f$read-regs (*address*, *regs*)
= fv-if (f-and3 (f-buf (cadr (*regs*)),
f$v-equal (*address*, v-threefix (cadddr (*regs*))),
t),
v-threefix (caddr (*regs*)),
dual-port-ram-value (32,
4,
append (*address*,
append (v-threefix (cadddr (*regs*)),
cons (f-nand (**t**,
f-buf (cadr (*regs*))),
v-threefix (caddr (*regs*))))),
car (*regs*)))

EVENT: Enable f$read-regs; name this event 'f$read-regs-off'.

THEOREM: properp-length-f$read-regs
properp (f$read-regs (*address*, *mem*))
∧ ((length (caddr (*mem*)) = 32)
→ (length (f$read-regs (*address*, *mem*)) = 32))

THEOREM: f$read-regs=read-regs
(memory-okp (4, 32, car (*regs*))
∧ boolp (cadr (*regs*))
∧ bvp (caddr (*regs*))
∧ (length (caddr (*regs*)) = 32)
∧ bvp (cadddr (*regs*))
∧ (length (cadddr (*regs*)) = 4)
∧ bvp (*address*)
∧ (length (*address*) = 4))
→ (f$read-regs (*address*, *regs*) = read-regs (*address*, *regs*))

THEOREM: regfile$value
(regfile& (*netlist*)
∧ (*test-regfile-* = `t`)
∧ (*disable-regfile-* = `t`)
∧ memory-properp (`4`, `32`, *mem*)
∧ properp (*last-data*)
∧ (length (*last-data*) = `32`)
∧ properp (*address*)
∧ (length (*address*) = `4`)
∧ properp (*last-address*)
∧ (length (*last-address*) = `4`))
→ (cdr (dual-eval (`0`,
                 'regfile,
                 cons (*clk*,
                   cons (*te*,
                     cons (*ti*,
                       cons (*we*,
                         cons (*disable-regfile-*,
                           cons (*test-regfile-*,
                             append (*address*, *data*))))))),
                 list (*mem*, *last-we*, *last-data*, *last-address*),
                 netlist))
  = f$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 (cadddr (*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*, cadddr (*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*))
∧ (length (f\$write-regs (*we*, *address*, *regs*, *data*)) = 4)

THEOREM: f\$write-regs=write-regs
(boolp (*we*)
∧ boolp (cadr (*regs*))
∧ bvp (caddr (*regs*))
∧ (length (caddr (*regs*)) = 32)
∧ bvp (cadddr (*regs*))
∧ (length (cadddr (*regs*)) = 4)
∧ bvp (*address*)
∧ (length (*address*) = 4)
∧ bvp (*data*)
∧ (length (*data*) = 32))
→ (f\$write-regs (*we*, *address*, *regs*, *data*)
  = write-regs (*we*, *address*, *regs*, *data*))

THEOREM: regfile\$state
(regfile& (*netlist*)
∧ (¬ *te*)
∧ (*test-regfile-* = **t**)
∧ (*disable-regfile-* = **t**)
∧ properp (*data*)
∧ (length (*data*) = 32)
∧ properp (*last-data*)
∧ (length (*last-data*) = 32)
∧ properp (*address*)
∧ (length (*address*) = 4)
∧ properp (*last-address*)
∧ (length (*last-address*) = 4))
→ (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))
∧    (cadddr (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))))))
```

∧ (cadddddr (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 (`'flags`, *netlist*) = FLAGS*)
∧ fd1slp& (delete-module (`'flags`, *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*) ∧ properp (*flags*) ∧ (length (*flags*) = 4))
→ (dual-eval (0,
`'flags`,
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*))
∧ (length (f$update-flags (*flags*, *set-flags*, *cvzbv*)) = 4)

THEOREM: f$update-flags=update-flags

$$
\begin{array}{ll}
& (\text{bvp}\,(\mathit{flags}) \\
\wedge & (\text{length}\,(\mathit{flags}) = \mathtt{4}) \\
\wedge & \text{bvp}\,(\mathit{set\text{-}flags}) \\
\wedge & (\text{length}\,(\mathit{set\text{-}flags}) = \mathtt{4}) \\
\wedge & \text{bvp}\,(\mathit{cvzbv}) \\
\wedge & (\text{length}\,(\mathit{cvzbv}) = \mathtt{35})) \\
\rightarrow & (\text{f\$update-flags}\,(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv}) \\
& \quad = \quad \text{update-flags}\,(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv}))
\end{array}
$$

THEOREM: flags$state-help

$$
\begin{array}{ll}
& (\text{flags\&}\,(\mathit{netlist}) \\
\wedge & \text{properp}\,(\mathit{flags}) \\
\wedge & (\text{length}\,(\mathit{flags}) = \mathtt{4}) \\
\wedge & (\mathit{set\text{-}flags} = \text{list}\,(\mathit{set\text{-}z},\ \mathit{set\text{-}n},\ \mathit{set\text{-}v},\ \mathit{set\text{-}c})) \\
\wedge & (\mathit{cvzbv} = \text{cons}\,(c,\ \text{cons}\,(v,\ \text{cons}\,(z,\ \mathit{bv})))) \\
\wedge & \text{properp}\,(\mathit{cvzbv}) \\
\wedge & (\text{length}\,(\mathit{cvzbv}) = \mathtt{35}) \\
\wedge & (\neg\ \mathit{te})) \\
\rightarrow & (\text{dual-eval}\,(\mathtt{2}, \\
& \qquad\qquad \mathtt{'flags}, \\
& \qquad\qquad \text{cons}\,(\mathit{clk},\ \text{cons}\,(\mathit{te},\ \text{cons}\,(\mathit{ti},\ \text{append}\,(\mathit{set\text{-}flags},\ \mathit{cvzbv})))), \\
& \qquad\qquad \mathit{flags}, \\
& \qquad\qquad \mathit{netlist}) \\
& \quad = \quad \text{f\$update-flags}\,(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv}))
\end{array}
$$

EVENT: Enable flags$state-help; name this event ‘flags$state-help-off’.

THEOREM: flags$state

$$
\begin{array}{ll}
& (\text{flags\&}\,(\mathit{netlist}) \\
\wedge & \text{properp}\,(\mathit{flags}) \\
\wedge & (\text{length}\,(\mathit{flags}) = \mathtt{4}) \\
\wedge & \text{properp}\,(\mathit{set\text{-}flags}) \\
\wedge & (\text{length}\,(\mathit{set\text{-}flags}) = \mathtt{4}) \\
\wedge & \text{properp}\,(\mathit{cvzbv}) \\
\wedge & (\text{length}\,(\mathit{cvzbv}) = \mathtt{35}) \\
\wedge & (\neg\ \mathit{te})) \\
\rightarrow & (\text{dual-eval}\,(\mathtt{2}, \\
& \qquad\qquad \mathtt{'flags}, \\
& \qquad\qquad \text{cons}\,(\mathit{clk},\ \text{cons}\,(\mathit{te},\ \text{cons}\,(\mathit{ti},\ \text{append}\,(\mathit{set\text{-}flags},\ \mathit{cvzbv})))), \\
& \qquad\qquad \mathit{flags}, \\
& \qquad\qquad \mathit{netlist}) \\
& \quad = \quad \text{f\$update-flags}\,(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv}))
\end{array}
$$

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))
∧ (cadddr (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))))))
∧ (caddddr (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$value
(extend-immediate& (*netlist*)
∧ properp (*immediate*)
∧ (length (*immediate*) = `9`)
∧ properp (*reg-data*)
∧ (length (*reg-data*) = `32`))
→ (dual-eval (`0`,
    `'extend-immediate`,
    cons (*select-immediate*, append (*immediate*, *reg-data*)),
    state,
    netlist)
  = f$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,

netlist),
state-bindings,
netlist) **endif**

DEFINITION:
ttl-input-pads* (n)
= cons (index (`'ttl-input-pads`, n),
cons (cons (`'pi`, indices (`'in`, `0`, n)),
cons (cons (index (`'po`, $n - 1$), indices (`'b-out`, `0`, n)),
cons (ttl-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)))
∧ (cadddr (ttl-input-pads* (n)) = ttl-input-pads-body (`0`, n, `'pi`))
∧ (caddddr (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:
ttl-input-pads& (*netlist*, n)
= ((lookup-module (index (`'ttl-input-pads`, n), *netlist*)
= ttl-input-pads* (n))
∧ ttl-input& (delete-module (index (`'ttl-input-pads`, n),
netlist))
∧ b-buf& (delete-module (index (`'ttl-input-pads`, n), *netlist*)))

EVENT: Enable ttl-input-pads&; name this event 'ttl-input-pads&-off'.

DEFINITION:
ttl-input-pads$netlist ($n$)
= cons (ttl-input-pads* (n), TTL-INPUT$NETLIST ∪ B-BUF$NETLIST)

THEOREM: ttl-input-pads-body$unbound-in-body
($l < m$)
→ unbound-in-body (index (`'b-out`, l), ttl-input-pads-body (m, n, pi))

EVENT: Enable ttl-input-pads-body$unbound-in-body; name this event 'ttl-input-pads-body$unbound-in-body-off'.

THEOREM: ttl-input-pads-body$value
(ttl-input& (*netlist*) ∧ b-buf& (*netlist*))
→ (collect-value (indices ('b-out, *m*, *n*),
dual-eval (1,
ttl-input-pads-body (*m*, *n*, *pi*),
bindings,
state-bindings,
netlist))
= v-threefix (collect-value (indices ('in, *m*, *n*), *bindings*)))

EVENT: Enable ttl-input-pads-body$value; name this event 'ttl-input-pads-body$value-off'.

THEOREM: ttl-input-pads$value
(ttl-input-pads& (*netlist*, *n*) ∧ (length (*inputs*) = *n*) ∧ properp (*inputs*))
→ (cdr (dual-eval (0,
index ('ttl-input-pads, *n*),
cons (*pi*, *inputs*),
state,
netlist))
= v-threefix (*inputs*))

EVENT: Enable ttl-input-pads$value; name this event 'ttl-input-pads$value-off'.

DEFINITION:
ttl-output-pads-body (*m*, *n*)
= **if** $n \simeq 0$ **then nil**
else cons (list (index ('b, *m*),
list (index ('b-in, *m*)),
'b-buf,
list (index ('in, *m*))),
cons (list (index ('g, *m*),
list (index ('out, *m*)),
'ttl-output,
list (index ('b-in, *m*))),
ttl-output-pads-body ($1 + m$, $n - 1$))) **endif**

EVENT: Enable ttl-output-pads-body; name this event 'ttl-output-pads-body-off'.

DEFINITION:
ttl-output-pads-body$induction (*m*, *n*, *bindings*, *state-bindings*, *netlist*)
= **if** $n \simeq 0$ **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: ttl-output-pads-body$unbound-in-body
$(l < m) \rightarrow$ unbound-in-body (index (`'out`, l), ttl-output-pads-body (m, n))

EVENT: Enable ttl-output-pads-body$unbound-in-body; name this event 'ttl-output-pads-body$unbound-in-body-off'.

THEOREM: ttl-output-pads-body$value
(ttl-output& (*netlist*) $\land$ b-buf& (*netlist*))
$\rightarrow$ (collect-value (indices (`'out`, m, n),
dual-eval (`1`,
ttl-output-pads-body (m, n),
bindings,
state-bindings,
netlist))
= v-threefix (collect-value (indices (`'in`, m, n), *bindings*)))

EVENT: Enable ttl-output-pads-body$value; name this event 'ttl-output-pads-body$value-off'.

THEOREM: ttl-output-pads$value
(ttl-output-pads& (*netlist*, n) $\land$ (length (*inputs*) $=$ n) $\land$ properp (*inputs*))
$\rightarrow$ (dual-eval (`0`, index (`'ttl-output-pads`, n), *inputs*, *state*, *netlist*)
= v-threefix (*inputs*))

DEFINITION:
ttl-tri-output-pads-body (m, n)
= **if** $n \simeq$ `0` **then nil**
else cons (list (index (`'g`, m),
list (index (`'out`, m)),
`'ttl-tri-output`,
cons (index (`'in`, m), `'(enable-buf)`)),
ttl-tri-output-pads-body ($1 + m$, $n - 1$)) **endif**

EVENT: Enable ttl-tri-output-pads-body; name this event 'ttl-tri-output-pads-body-off'.

DEFINITION:
ttl-tri-output-pads-body$induction ($m$, n, *bindings*, *state-bindings*, *netlist*)
= **if** $n \simeq$ `0` **then t**
else ttl-tri-output-pads-body$induction ($1 + m$,
$n - 1$,
dual-eval-body-bindings (`1`,
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: 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))
∧ (cadddr (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)))
∧ (caddddr (ttl-tri-output-pads* (n)) = **nil**)

EVENT: Enable ttl-tri-output-pads*; name this event 'ttl-tri-output-pads*-off'.

EVENT: Enable 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))
∧ **if** $n < 8$
then b-buf& (delete-module (index (`'ttl-tri-output-pads`, n),
netlist))
else b-buf-pwr& (delete-module (index (`'ttl-tri-output-pads`,
n),
netlist)) **endif**)

EVENT: Enable ttl-tri-output-pads&; name this event 'ttl-tri-output-pads&-off'.

DEFINITION:
ttl-tri-output-pads$netlist ($n$)
= cons (ttl-tri-output-pads* (n),
if $n < 8$ **then** B-BUF$NETLIST
else B-BUF-PWR$NETLIST **endif**
∪ TTL-TRI-OUTPUT$NETLIST)

THEOREM: ttl-tri-output-pads-body$unbound-in-body
($l < m$)
→ unbound-in-body (index (`'out`, l), ttl-tri-output-pads-body (m, n))

EVENT: Enable ttl-tri-output-pads-body$unbound-in-body; name this event 'ttl-tri-output-pads-body$unbound-in-body-off'.

THEOREM: ttl-tri-output-pads-body$value
ttl-tri-output& (*netlist*)
→ (collect-value (indices (`'out`, m, n),
dual-eval (`1`,
ttl-tri-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 ttl-tri-output-pads-body$value; name this event 'ttl-tri-output-pads-body$value-off'.

THEOREM: ttl-tri-output-pads$value
(ttl-tri-output-pads& (*netlist*, n)
∧ (length (*inputs*) = n)
∧ properp (*inputs*))
→ (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** $n \simeq$ `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** $n \simeq$ `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: 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 ttl-bidirect-pads*; name this event 'ttl-bidirect-pads*-off'.

EVENT: Enable ttl-bidirect-pads*$destructure; name this event 'ttl-bidirect-pads*$destructure-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 ('`ttl-bidirect-pads`, n),
netlist))
else b-buf-pwr& (delete-module (index ('`ttl-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)
  ∧  (length(data) = n)
  ∧  properp(data)
  ∧  (length(inputs) = n)
  ∧  properp(inputs))
 →  (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: reset- (*state*) = nth (7, *state*)

DEFINITION: dtack- (*state*) = nth (8, *state*)

DEFINITION: hold- (*state*) = nth (9, *state*)

DEFINITION: pc-reg (*state*) = nth (10, *state*)

DEFINITION: cntl-state (*state*) = 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: reset–input (*ext-in*) = nth (0, *ext-in*)

DEFINITION: hold–input (*ext-in*) = nth (1, *ext-in*)

DEFINITION: pc-reg-input (*ext-in*) = 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$write-regs (we-regs (v-threefix (cntl-state (car (total-state)))),
                              regs-address (v-threefix (cntl-state (car (total-state)))),
                              regs (car (total-state)),
                              bv (f$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)))),
          f$update-flags (flags (car (total-state)),
                          we-flags (v-threefix (cntl-state (car (total-state)))),
                          f$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))),
          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-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))))),
                                    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 (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))))),
                                    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 (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))))),
                                 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** $n \simeq 0$ **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
($n \simeq 0$) $\rightarrow$ (run-fm9001 (*state*, *inputs*, *n*) = *state*)

THEOREM: run-fm9001-step-case
($n \not\simeq 0$)
$\rightarrow$ (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
run-fm9001 ($state$, $inputs$, $n + m$)
= run-fm9001 (run-fm9001 ($state$, $inputs$, n), nthcdr (n, $inputs$), m)

DEFINITION:
fm9001-state-structure ($state$)
= ((length ($state$) = 2)
∧ properp ($state$)
∧ (length (car ($state$)) = 12)
∧ properp (car ($state$))
∧ (length (cadr ($state$)) = 8)
∧ properp (cadr ($state$))
∧ (length (caar ($state$)) = 4)
∧ properp (caar ($state$)))

THEOREM: fm9001-state-structure\$step
fm9001-state-structure ($state$)
→ fm9001-state-structure (fm9001-next-state ($state$, $external\text{-}inputs$))

THEOREM: fm9001-state-structure\$induction
fm9001-state-structure ($state$)
→ fm9001-state-structure (run-fm9001 ($state$, $inputs$, $steps$))

THEOREM: fm9001-state-as-a-list
fm9001-state-structure ($state$)
→ ($state$ = list (list (list (caaar ($state$),
caadaar ($state$),
caddaar ($state$),
cadddaar ($state$)),
cadar ($state$),
caddar ($state$),
cadddar ($state$),
caddddar ($state$),
cadddddar ($state$),
caddddddar ($state$),
cadddddddar ($state$),
caddddddddar ($state$),
cadddddddddar ($state$),
caddddddddddar ($state$),
cadddddddddddar ($state$)),
list (caadr ($state$),

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
 (CADDDR (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`)
  ∧ fd1s& (delete-module (`'chip-module`, *netlist*))
  ∧ fd1slp& (delete-module (`'chip-module`, *netlist*))
  ∧ we-reg& (delete-module (`'chip-module`, *netlist*), `4`)
  ∧ reg& (delete-module (`'chip-module`, *netlist*), `40`)

$\wedge$ id& (delete-module (`'chip-module`, *netlist*))
$\wedge$ b-buf-pwr& (delete-module (`'chip-module`, *netlist*))
$\wedge$ b-buf& (delete-module (`'chip-module`, *netlist*))
$\wedge$ b-not& (delete-module (`'chip-module`, *netlist*))
$\wedge$ extend-immediate& (delete-module (`'chip-module`, *netlist*))
$\wedge$ dec-pass& (delete-module (`'chip-module`, *netlist*), `32`)
$\wedge$ tv-if& (delete-module (`'chip-module`, *netlist*), make-tree (`32`))
$\wedge$ core-alu& (delete-module (`'chip-module`, *netlist*),
make-tree (`32`))
$\wedge$ 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
∪ ((((FLAGS$NETLIST ∪ we-reg$netlist (`32`))
∪ (FD1S$NETLIST ∪ FD1SLP$NETLIST))
∪ (((we-reg$netlist (`4`) ∪ reg$netlist (`40`))
∪ (ID$NETLIST
∪ EXTEND-IMMEDIATE$NETLIST))
∪ ((dec-pass$netlist (`32`)
∪ tv-if$netlist (make-tree (`32`)))
∪ (core-alu$netlist (make-tree (`32`))
∪ NEXT-CNTL-STATE$NETLIST))))
∪ (B-BUF-PWR$NETLIST
∪ (B-BUF$NETLIST ∪ B-NOT$NETLIST))))

THEOREM: equal-length-40-as-collected-nth+subrange
((length (l) = `40`) $\wedge$ properp (l))
$\rightarrow$ (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
$\land$ properp (x))
$\rightarrow$ (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*))
$\land$ memory-properp (4, 32, caar (*machine-state*))
$\land$ properp (caddar (*machine-state*))
$\land$ (length (caddar (*machine-state*)) = 32)
$\land$ properp (cadddar (*machine-state*))
$\land$ (length (cadddar (*machine-state*)) = 4)
$\land$ properp (cadr (*machine-state*))
$\land$ (length (cadr (*machine-state*)) = 4)
$\land$ properp (caddr (*machine-state*))
$\land$ (length (caddr (*machine-state*)) = 32)
$\land$ properp (cadddr (*machine-state*))
$\land$ (length (cadddr (*machine-state*)) = 32)
$\land$ properp (caddddr (*machine-state*))
$\land$ (length (caddddr (*machine-state*)) = 32)
$\land$ properp (cadddddr (*machine-state*))
$\land$ (length (cadddddr (*machine-state*)) = 32)
$\land$ properp (caddddddr (*machine-state*))
$\land$ (length (caddddddr (*machine-state*)) = 32)
$\land$ properp (cadddddddddddr (*machine-state*))
$\land$ (length (cadddddddddddr (*machine-state*)) = 4)
$\land$ properp (caddddddddddddr (*machine-state*))
$\land$ (length (caddddddddddddr (*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$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))),
                  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)
```

```
                                          (INDICES 'DATA-BUS 0 32))))))))))
(EQUAL
 (CADDR (CHIP*))
 (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))))))))))))
(EQUAL
 (CADDDR (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 (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))))
 (EQUAL (CADDDDR (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)

∧ chip-module& (delete-module (`'chip`, *netlist*))
∧ ttl-bidirect-pads& (delete-module (`'chip`, *netlist*), `32`)
∧ ttl-output-parametric& (delete-module (`'chip`, *netlist*))
∧ ttl-output& (delete-module (`'chip`, *netlist*))
∧ ttl-output-fast& (delete-module (`'chip`, *netlist*))
∧ ttl-tri-output-fast& (delete-module (`'chip`, *netlist*))
∧ ttl-tri-output-pads& (delete-module (`'chip`, *netlist*), `32`)
∧ ttl-output-pads& (delete-module (`'chip`, *netlist*), `4`)
∧ ttl-output-pads& (delete-module (`'chip`, *netlist*), `5`)
∧ procmon& (delete-module (`'chip`, *netlist*)))

EVENT: Enable chip&; name this event 'chip&-off'.

DEFINITION:
CHIP$NETLIST
= cons (CHIP*,
(((ttl-bidirect-pads$netlist (`32`) ∪ CHIP-MODULE$NETLIST)
∪ (TTL-INPUT$NETLIST ∪ ttl-input-pads$netlist (`4`)))
∪ ((TTL-CLK-INPUT$NETLIST ∪ VDD-PARAMETRIC$NETLIST)
∪ (TTL-OUTPUT-PARAMETRIC$NETLIST
∪ TTL-OUTPUT$NETLIST)))
∪ (((TTL-OUTPUT-FAST$NETLIST
∪ TTL-TRI-OUTPUT-FAST$NETLIST)
∪ (ttl-tri-output-pads$netlist (`32`)
∪ ttl-output-pads$netlist (`4`)))
∪ (ttl-output-pads$netlist (`5`) ∪ PROCMON$NETLIST)))

THEOREM: chip$state
(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))

$\land \quad (te = \mathbf{f})$
$\land \quad (\mathit{disable\text{-}regfile\text{-}} = \mathtt{t})$
$\land \quad (\mathit{test\text{-}regfile\text{-}} = \mathtt{t})$
$\land \quad \mathrm{properp}\,(\mathit{pc\text{-}reg\text{-}in})$
$\land \quad (\mathrm{length}\,(\mathit{pc\text{-}reg\text{-}in}) = \mathtt{4})$
$\land \quad \mathrm{properp}\,(\mathit{data\text{-}in})$
$\land \quad (\mathrm{length}\,(\mathit{data\text{-}in}) = \mathtt{32}))$
$\to \quad (\mathrm{dual\text{-}eval}\,(\mathtt{2},$
$\qquad\qquad \mathtt{'chip},$
$\qquad\qquad \mathrm{cons}\,(\mathit{clk},$
$\qquad\qquad\quad \mathrm{cons}\,(\mathit{ti},$
$\qquad\qquad\qquad \mathrm{cons}\,(\mathit{te},$
$\qquad\qquad\qquad\quad \mathrm{cons}\,(\mathit{dtack\text{-}},$
$\qquad\qquad\qquad\qquad \mathrm{cons}\,(\mathit{reset\text{-}},$
$\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(\mathit{hold\text{-}},$
$\qquad\qquad\qquad\qquad\qquad \mathrm{cons}\,(\mathit{disable\text{-}regfile\text{-}},$
$\qquad\qquad\qquad\qquad\qquad\quad \mathrm{cons}\,(\mathit{test\text{-}regfile\text{-}},$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{append}\,(\mathit{pc\text{-}reg\text{-}in},$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{data\text{-}in})))))))))),$
$\qquad\qquad \mathrm{list}\,(\mathrm{list}\,(\mathit{regs\text{-}regs},\ \mathit{regs\text{-}we},\ \mathit{regs\text{-}data},\ \mathit{regs\text{-}addr}),$
$\qquad\qquad\quad \mathit{flags},$
$\qquad\qquad\quad \mathit{a\text{-}reg},$
$\qquad\qquad\quad \mathit{b\text{-}reg},$
$\qquad\qquad\quad \mathit{i\text{-}reg},$
$\qquad\qquad\quad \mathit{data\text{-}out},$
$\qquad\qquad\quad \mathit{addr\text{-}out},$
$\qquad\qquad\quad \mathit{last\text{-}reset\text{-}},$
$\qquad\qquad\quad \mathit{last\text{-}dtack\text{-}},$
$\qquad\qquad\quad \mathit{last\text{-}hold\text{-}},$
$\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg},$
$\qquad\qquad\quad \mathit{cntl\text{-}state}),$
$\qquad\qquad \mathit{netlist})$
$\quad = \quad \mathrm{list}\,(\mathrm{f\$write\text{-}regs}\,(\mathrm{we\text{-}regs}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad \mathrm{regs\text{-}address}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad \mathrm{list}\,(\mathit{regs\text{-}regs},\ \mathit{regs\text{-}we},\ \mathit{regs\text{-}data},\ \mathit{regs\text{-}addr}),$
$\qquad\qquad\qquad\qquad \mathrm{bv}\,(\mathrm{f\$core\text{-}alu}\,(\mathrm{alu\text{-}c}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{v\text{-}threefix}\,(\mathit{a\text{-}reg}),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{v\text{-}threefix}\,(\mathit{b\text{-}reg}),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{alu\text{-}zero}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{alu\text{-}mpg}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{alu\text{-}op}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$
$\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{make\text{-}tree}\,(\mathtt{32})))),$
$\qquad\qquad \mathrm{f\$update\text{-}flags}\,(\mathit{flags},$
$\qquad\qquad\qquad\qquad\quad \mathrm{we\text{-}flags}\,(\mathrm{v\text{-}threefix}\,(\mathit{cntl\text{-}state})),$

```
                  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),
     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$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)))),
               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
∪ (v-pullup$netlist (32)
∪ (MEM-32X32$NETLIST
∪ (v-wire$netlist (32) ∪ CHIP$NETLIST))))

DEFINITION:
memory-state-invariant (*mem-state*)
= (memory-properp (32, 32, car (*mem-state*))
$\wedge$ properp (caddddddr (*mem-state*))
$\wedge$ properp (cadddddddr (*mem-state*))
$\wedge$ (length (cadddddddr (*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 (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)))),
                                  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 (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))))),
                           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 (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))))),
                           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$read-regs (regs-address (v-threefix (cntl-state)),
                                                         list (regs-regs,
                                                               regs-we,
                                                               regs-data,
                                                               regs-addr)))),
            addr-out),
      f-buf (reset-),
      f-or (strobe- (v-threefix (cntl-state)),
            f-buf (car (memory-value (mem-state,
                                      f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                        strobe- (v-threefix (cntl-state)))),
                                      f-pullup (ft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                        f-buf (rw- (v-threefix (cntl-state))))),
                                      v-pullup (vft-buf (f-buf (hdack- (v-threefix (cntl-state))),
                                                         v-threefix (addr-out))),
                                      make-list (32, x))))),
      f-if (we-hold- (v-threefix (cntl-state)),
            f-buf (hold-),
            last-hold-),
      fv-if (we-pc-reg (v-threefix (cntl-state)),
             pc-reg-in,
             last-pc-reg),
      v-threefix (f$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))),
```

```
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 (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))))))))
```

Event: Enable chip-system$state-help; name this event 'chip-system$state-help-off'.

Theorem: chip-system$state

```
 (chip-system& (netlist)
 ∧   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-,
```

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad\qquad\text{cons}\,(\textit{hold-},\\
\qquad\qquad\qquad\qquad\qquad\qquad\quad\text{cons}\,(\textit{disable-regfile-},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{cons}\,(\textit{test-regfile-},\ \textit{pc-reg-in}))))))),\\
\qquad\quad\text{list}\,(\text{list}\,(\text{list}\,(\textit{regs-regs},\ \textit{regs-we},\ \textit{regs-data},\ \textit{regs-addr}),\\
\qquad\qquad\qquad\textit{flags},\\
\qquad\qquad\qquad\textit{a-reg},\\
\qquad\qquad\qquad\textit{b-reg},\\
\qquad\qquad\qquad\textit{i-reg},\\
\qquad\qquad\qquad\textit{data-out},\\
\qquad\qquad\qquad\textit{addr-out},\\
\qquad\qquad\qquad\textit{last-reset-},\\
\qquad\qquad\qquad\textit{last-dtack-},\\
\qquad\qquad\qquad\textit{last-hold-},\\
\qquad\qquad\qquad\textit{last-pc-reg},\\
\qquad\qquad\qquad\textit{cntl-state}),\\
\qquad\qquad\quad\textit{mem-state}),\\
\qquad\quad\textit{netlist})\\
= \ \text{fm9001-next-state}\,(\text{list}\,(\text{list}\,(\text{list}\,(\textit{regs-regs},\\
\qquad\qquad\qquad\qquad\qquad\qquad\textit{regs-we},\\
\qquad\qquad\qquad\qquad\qquad\qquad\textit{regs-data},\\
\qquad\qquad\qquad\qquad\qquad\qquad\textit{regs-addr}),\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{flags},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{a-reg},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{b-reg},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{i-reg},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{data-out},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{addr-out},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{last-reset-},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{last-dtack-},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{last-hold-},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{last-pc-reg},\\
\qquad\qquad\qquad\qquad\qquad\quad\textit{cntl-state}),\\
\qquad\qquad\qquad\qquad\qquad\textit{mem-state}),\\
\qquad\qquad\qquad\qquad\text{cons}\,(\textit{reset-},\ \text{cons}\,(\textit{hold-},\ \textit{pc-reg-in}))))
\end{array}
$$

EVENT: Enable chip-system$state; name this event 'chip-system$state-off'.

THEOREM: next-memory-state-preserves-memory-invariant

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{memory-state-invariant}\,(\textit{mem-state})}\\
\land & \text{properp}\,(\textit{data})\\
\land & (\text{length}\,(\textit{data}) = 32)\\
\land & (\text{length}\,(\textit{address}) = 32))\\
\rightarrow & \text{memory-state-invariant}\,(\text{next-memory-state}\,(\textit{mem-state},\\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\textit{strobe-},
\end{array}
$$

rw-,
address,
data))

THEOREM: memory-properp-dual-port-ram-state-crock
(memory-properp (4, 32, *regs*)
∧ (length (*a-address*) = 4)
∧ (length (*b-address*) = 4)
∧ properp (*data*)
∧ (length (*data*) = 32))
→ 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*)
∧ (length (*a-address*) = 4)
∧ (length (*b-address*) = 4)
∧ properp (*data*)
∧ (length (*data*) = 32))
→ 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*)
∧ properp (pc-reg-input (*inputs*))
∧ (length (pc-reg-input (*inputs*)) = 4))
→ chip-system-invariant (fm9001-next-state (*state*, *inputs*))

DEFINITION:
chip-system-input-invariant (*inputs*)
= ((caddr (*inputs*) = **f**)
∧ (caddddddr (*inputs*) = **t**)
∧ (cadddddddr (*inputs*) = **t**)
∧ properp (cddddddddr (*inputs*))
∧ (length (cddddddddr (*inputs*)) = 4))

THEOREM: rewrite-chip-system-input-invariant

$$
\begin{array}{l}
\text{chip-system-input-invariant}\,(inputs) \\
= \quad ((inputs = \text{cons}\,(\text{car}\,(inputs), \\
\qquad\qquad \text{cons}\,(\text{cadr}\,(inputs), \\
\qquad\qquad\quad \text{cons}\,(\text{caddr}\,(inputs), \\
\qquad\qquad\qquad \text{cons}\,(\text{cadddr}\,(inputs), \\
\qquad\qquad\qquad\quad \text{cons}\,(\text{caddddr}\,(inputs), \\
\qquad\qquad\qquad\qquad \text{cons}\,(\text{cadddddr}\,(inputs), \\
\qquad\qquad\qquad\qquad\quad \text{cons}\,(\text{caddddddr}\,(inputs), \\
\qquad\qquad\qquad\qquad\qquad \text{cddddddddr}\,(inputs)))))))) \\
\quad \land \quad (\text{caddr}\,(inputs) = \mathbf{f}) \\
\quad \land \quad (\text{cadddddr}\,(inputs) = \mathbf{t}) \\
\quad \land \quad (\text{caddddddr}\,(inputs) = \mathbf{t}) \\
\quad \land \quad \text{properp}\,(\text{cddddddddr}\,(inputs)) \\
\quad \land \quad (\text{length}\,(\text{cddddddddr}\,(inputs)) = \mathbf{4}))
\end{array}
$$

DEFINITION:

$$
\begin{array}{l}
\text{chip-system-operating-inputs-p}\,(inputs,\ n) \\
= \quad \textbf{if } n \simeq 0 \textbf{ then t} \\
\quad\ \ \textbf{else } \text{chip-system-input-invariant}\,(\text{car}\,(inputs)) \\
\qquad\quad \land \quad \text{chip-system-operating-inputs-p}\,(\text{cdr}\,(inputs),\ n - 1) \textbf{ endif}
\end{array}
$$

EVENT: Enable chip-system-operating-inputs-p; name this event ‘chip-system-operating-inputs-p-off’.

THEOREM: open-chip-system-operating-inputs-p

$$
\begin{array}{l}
((n \simeq 0) \rightarrow (\text{chip-system-operating-inputs-p}\,(inputs,\ n) = \mathbf{t})) \\
\land \quad ((n \not\simeq 0) \\
\qquad \rightarrow \quad (\text{chip-system-operating-inputs-p}\,(inputs,\ n) \\
\qquad\qquad = \quad (\text{chip-system-input-invariant}\,(\text{car}\,(inputs)) \\
\qquad\qquad\quad \land \quad \text{chip-system-operating-inputs-p}\,(\text{cdr}\,(inputs), \\
\qquad\qquad\qquad\qquad n - 1))))
\end{array}
$$

DEFINITION:

$$
\begin{array}{l}
\text{map-up-1-input}\,(inputs) \\
= \quad \text{cons}\,(\text{cadddr}\,(inputs),\ \text{cons}\,(\text{caddddr}\,(inputs),\ \text{cddddddddr}\,(inputs)))
\end{array}
$$

DEFINITION:

$$
\begin{array}{l}
\text{map-up-inputs}\,(inputs) \\
= \quad \textbf{if } inputs \simeq \textbf{nil} \textbf{ then nil} \\
\quad\ \ \textbf{else } \text{cons}\,(\text{map-up-1-input}\,(\text{car}\,(inputs)), \\
\qquad\qquad \text{map-up-inputs}\,(\text{cdr}\,(inputs))) \textbf{ endif}
\end{array}
$$

EVENT: Enable map-up-inputs; name this event ‘map-up-inputs-off’.

THEOREM: open-map-up-inputs
$((inputs \simeq \textbf{nil}) \rightarrow (\text{map-up-inputs}\,(inputs) = \textbf{nil}))$
$\land\ \ (\text{listp}\,(inputs)$
$\quad\rightarrow\ \ (\text{map-up-inputs}\,(inputs)$
$\qquad = \ \ \text{cons}\,(\text{map-up-1-input}\,(\text{car}\,(inputs)),$
$\qquad\qquad \text{map-up-inputs}\,(\text{cdr}\,(inputs)))))$

THEOREM: chip-system=fm9001\$step
$(\text{chip-system\&}\,(netlist)$
$\ \land\ \ \text{fm9001-state-structure}\,(state)$
$\ \land\ \ \text{chip-system-invariant}\,(state)$
$\ \land\ \ \text{chip-system-input-invariant}\,(inputs))$
$\rightarrow\ \ (\text{dual-eval}\,(2,\ \texttt{'chip-system},\ inputs,\ state,\ netlist)$
$\qquad = \ \ \text{fm9001-next-state}\,(state,\ \text{map-up-1-input}\,(inputs)))$

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** $n \simeq \texttt{0}$ **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
$(\text{chip-system\&}\,(netlist)$
$\ \land\ \ \text{fm9001-state-structure}\,(state)$
$\ \land\ \ \text{chip-system-invariant}\,(state)$
$\ \land\ \ \text{chip-system-operating-inputs-p}\,(inputs,\ n)$
$\ \land\ \ (fm9001\text{-}inputs = \text{map-up-inputs}\,(inputs)))$
$\rightarrow\ \ (\text{simulate-dual-eval-2}\,(\texttt{'chip-system},\ inputs,\ state,\ netlist,\ n)$
$\qquad = \ \ \text{run-fm9001}\,(state,\ fm9001\text{-}inputs,\ n))$

DEFINITION:
run-inputs-p (*inputs*, *n*)
$=\ \ $ **if** $n \simeq \texttt{0}$ **then t**
else (reset–input (car (*inputs*)) = **t**)
$\land$ properp (pc-reg-input (car (*inputs*)))
$\land$ (length (pc-reg-input (car (*inputs*))) = **4**)
$\land$ 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

$$((n \simeq \mathtt{0}) \rightarrow (\text{run-inputs-p}\,(inputs,\ n) = \mathbf{t}))$$
$$\land\ ((n \not\simeq \mathtt{0}) \rightarrow (\text{run-inputs-p}\,(inputs,\ n) = ((\text{reset–input}\,(\text{car}\,(inputs)) = \mathbf{t}) \land \text{properp}\,(\text{pc-reg-input}\,(\text{car}\,(inputs))) \land (\text{length}\,(\text{pc-reg-input}\,(\text{car}\,(inputs))) = \mathtt{4}) \land \text{run-inputs-p}\,(\text{cdr}\,(inputs),\ n - 1))))$$

THEOREM: run-inputs-p-plus

$$\text{run-inputs-p}\,(inputs,\ n + m) = (\text{run-inputs-p}\,(inputs,\ n) \land \text{run-inputs-p}\,(\text{nthcdr}\,(n,\ inputs),\ m))$$

THEOREM: run-inputs-p-1

$$(\text{run-inputs-p}\,(inputs,\ n) \land (n \not\simeq \mathtt{0})) \rightarrow \text{run-inputs-p}\,(inputs,\ \mathtt{1})$$

THEOREM: open-run-inputs-p-add1

$$\text{run-inputs-p}\,(inputs,\ 1 + n) = ((\text{reset–input}\,(\text{car}\,(inputs)) = \mathbf{t}) \land \text{properp}\,(\text{pc-reg-input}\,(\text{car}\,(inputs))) \land (\text{length}\,(\text{pc-reg-input}\,(\text{car}\,(inputs))) = \mathtt{4}) \land \text{run-inputs-p}\,(\text{cdr}\,(inputs),\ n))$$

DEFINITION:

$$\text{regfile-okp}\,(regs) = ((\text{length}\,(regs) = \mathtt{4}) \land \text{memory-okp}\,(\mathtt{4},\ \mathtt{32},\ \text{car}\,(regs)) \land \text{all-ramp-mem}\,(\mathtt{4},\ \text{car}\,(regs)) \land \text{boolp}\,(\text{cadr}\,(regs)) \land (\text{bvp}\,(\text{caddr}\,(regs)) \land (\text{length}\,(\text{caddr}\,(regs)) = \mathtt{32})) \land \text{bvp}\,(\text{cadddr}\,(regs)) \land (\text{length}\,(\text{cadddr}\,(regs)) = \mathtt{4}))$$

THEOREM: rewrite-fm9001-next-state-for-step-lemmas

$$(\text{bvp}\,(flags) \land \text{bvp}\,(a\text{-}reg) \land \text{bvp}\,(b\text{-}reg) \land \text{bvp}\,(i\text{-}reg) \land \text{bvp}\,(data\text{-}out) \land \text{bvp}\,(addr\text{-}out) \land \text{boolp}\,(last\text{-}reset\text{-}) \land \text{boolp}\,(last\text{-}dtack\text{-}) \land \text{boolp}\,(last\text{-}hold\text{-}) \land \text{bvp}\,(last\text{-}pc\text{-}reg) \land \text{bvp}\,(cntl\text{-}state))$$

```
→   (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 (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))))),
                           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 (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))))),
                           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 (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))))),
                                      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 \not\simeq 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,
```

$\mathtt{0}$,
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*)
$\wedge$ cv-hyps (**t**, *pc-reg*, *i-reg*, *flags*, *pc-reg*)
$\wedge$ memory-okp ($\mathtt{32}$, $\mathtt{32}$, *mem*)
$\wedge$ regfile-okp (*regs*)
$\wedge$ bvp (*a-reg*)
$\wedge$ (length (*a-reg*) $= \mathtt{32}$)
$\wedge$ bvp (*b-reg*)
$\wedge$ (length (*b-reg*) $= \mathtt{32}$)
$\wedge$ bvp (*addr-out*)
$\wedge$ (length (*addr-out*) $= \mathtt{32}$)
$\wedge$ bvp (*data-out*)
$\wedge$ (length (*data-out*) $= \mathtt{32}$)
$\wedge$ ($n \not\simeq \mathtt{0}$)
$\wedge$ run-inputs-p (*inputs*, $\mathtt{1}$)
$\wedge$ boolp (*dtack-*)
$\wedge$ boolp (*hold-*)
$\wedge$ (*reset-* $=$ **t**)
$\wedge$ **nil**)
$\rightarrow$ (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)
```

$$
\begin{array}{ll}
\wedge & (\text{length}\,(b\text{-}reg) = \mathtt{32}) \\
\wedge & \text{bvp}\,(addr\text{-}out) \\
\wedge & (\text{length}\,(addr\text{-}out) = \mathtt{32}) \\
\wedge & \text{bvp}\,(data\text{-}out) \\
\wedge & (\text{length}\,(data\text{-}out) = \mathtt{32}) \\
\wedge & (n \not\simeq \mathtt{0}) \\
\wedge & \text{run-inputs-p}\,(inputs,\ \mathtt{1}) \\
\wedge & \text{boolp}\,(dtack\text{-}) \\
\wedge & \text{boolp}\,(hold\text{-}) \\
\wedge & (reset\text{-} = \mathbf{t}) \\
\wedge & (mem\text{-}count \simeq \mathtt{0})) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(regs,\ flags,\ a\text{-}reg,\ b\text{-}reg,\ i\text{-}reg,\ data\text{-}out,\ addr\text{-}out,\ reset\text{-},\ dtack\text{-},\ hold\text{-},\ pc\text{-}reg, \\
& \qquad \text{cv\_fetch0}\,(\mathbf{f},\ last\text{-}regs\text{-}address,\ last\text{-}i\text{-}reg,\ last\text{-}flags,\ last\text{-}pc\text{-}reg)), \\
& \qquad \text{list}\,(mem,\ \mathtt{2},\ clock,\ mem\text{-}count,\ mem\text{-}dtack,\ \mathbf{f},\ addr\text{-}out,\ data\text{-}out)), \\
& \qquad inputs, \\
& \qquad n) \\
& = \ \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ regs,\ \mathtt{0}), \\
& \qquad flags, \\
& \qquad \text{read-regs}\,(last\text{-}pc\text{-}reg,\ regs), \\
& \qquad b\text{-}reg, \\
& \qquad i\text{-}reg, \\
& \qquad data\text{-}out, \\
& \qquad \text{read-regs}\,(last\text{-}pc\text{-}reg,\ regs),
\end{array}
$$

```
                        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) ≃ 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 \not\simeq 0$)
∧ run-inputs-p (*inputs*, 1)
∧ boolp (**f**)
∧ boolp (*hold-*)

$$
\begin{array}{ll}
\wedge & (\mathit{reset\text{-}} = \mathbf{t}) \\
\wedge & (\mathit{mem\text{-}count} \simeq \mathtt{0})) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
& \qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad \mathit{reset\text{-}}, \\
& \qquad\qquad\qquad \mathbf{f}, \\
& \qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad \text{cv\_fetch3}\,(\mathbf{t}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{last\text{-}flags}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg})), \\
& \qquad\qquad\quad \text{list}\,(\mathit{mem},\ \mathtt{1},\ \mathit{clock},\ \mathit{mem\text{-}count},\ \mathbf{t},\ \mathbf{t},\ \mathit{addr\text{-}out},\ \mathit{last\text{-}data})), \\
& \qquad\qquad \mathit{inputs}, \\
& \qquad\qquad n) \\
& \quad = \quad \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad\qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \text{read-mem}\,(\mathit{addr\text{-}out},\ \mathit{mem}), \\
& \qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad \mathbf{f}, \\
& \qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \text{cv\_decode}\,(\mathbf{t}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg})), \\
& \qquad\qquad\qquad \text{list}\,(\mathit{mem}, \\
& \qquad\qquad\qquad\qquad \mathtt{1}, \\
& \qquad\qquad\qquad\qquad \mathit{clock}, \\
& \qquad\qquad\qquad\qquad \mathtt{0}, \\
& \qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad \mathit{addr\text{-}out},
\end{array}
$$

```
                        v-threefix (last-data))),
                    cdr (inputs),
                    n − 1))
```

THEOREM: fetch3$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_fetch3 (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 ≃ 0
           then  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,
                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 \not\simeq 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*)
∧ 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*)

$$
\begin{array}{ll}
\land & (\mathrm{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \mathrm{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\mathrm{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & (n \not\simeq \mathtt{0}) \\
\land & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\land & \mathrm{boolp}\,(\mathit{dtack\text{-}}) \\
\land & \mathrm{boolp}\,(\mathit{hold\text{-}}) \\
\land & (\mathit{reset\text{-}} = \mathbf{t}) \\
\land & \mathbf{nil}) \\
\to & (\text{run-fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathit{regs}, \mathit{flags}, \mathit{a\text{-}reg}, \mathit{b\text{-}reg}, \mathit{i\text{-}reg}, \mathit{data\text{-}out}, \mathit{addr\text{-}out}, \mathit{reset\text{-}}, \mathit{dtack\text{-}}, \mathit{hold\text{-}}, \mathit{pc\text{-}reg}, \\
& \qquad \text{cv\_regb}\,(\mathbf{t}, \mathit{last\text{-}regs\text{-}address}, \mathit{last\text{-}i\text{-}reg}, \mathit{last\text{-}flags}, \mathit{last\text{-}pc\text{-}reg})), \\
& \qquad \mathrm{list}\,(\mathit{mem}, \mathtt{0}, \mathit{clock}, \mathit{mem\text{-}count}, \mathit{mem\text{-}dtack}, \mathbf{t}, \mathit{last\text{-}address}, \mathit{last\text{-}data})), \\
& \qquad \mathit{inputs}, \\
& \qquad n) \\
& = \text{run-fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\text{write-regs}\,(\mathbf{f}, \mathtt{0}, \mathit{regs}, \mathtt{0}), \\
& \qquad \mathit{flags}, \\
& \qquad \mathit{a\text{-}reg}, \\
& \qquad \text{read-regs}\,(\text{rn-b}\,(\mathit{last\text{-}i\text{-}reg}), \mathit{regs}), \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \mathit{addr\text{-}out}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathbf{t},
\end{array}
$$

```
                              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)
  ∧   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_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 \not\simeq 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)),
       inputs,
       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) ≃ 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 \not\simeq 0$)

$$
\begin{array}{ll}
\wedge & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\wedge & \text{boolp}\,(\mathbf{f}) \\
\wedge & \text{boolp}\,(\mathit{hold\text{-}}) \\
\wedge & (\mathit{reset\text{-}} = \mathbf{t}) \\
\wedge & (\mathit{mem\text{-}count} \simeq \mathtt{0})) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
& \quad \mathit{flags}, \\
& \quad \mathit{a\text{-}reg}, \\
& \quad \mathit{b\text{-}reg}, \\
& \quad \mathit{i\text{-}reg}, \\
& \quad \mathit{data\text{-}out}, \\
& \quad \mathit{addr\text{-}out}, \\
& \quad \mathit{reset\text{-}}, \\
& \quad \mathbf{f}, \\
& \quad \mathit{hold\text{-}}, \\
& \quad \mathit{pc\text{-}reg}, \\
& \quad \text{cv\_reada3}\,(\mathbf{t}, \\
& \qquad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad \mathit{last\text{-}flags}, \\
& \qquad \mathit{last\text{-}pc\text{-}reg})), \\
& \quad \text{list}\,(\mathit{mem},\ \mathtt{1},\ \mathit{clock},\ \mathit{mem\text{-}count},\ \mathbf{t},\ \mathbf{t},\ \mathit{addr\text{-}out},\ \mathit{last\text{-}data})), \\
& \quad \mathit{inputs}, \\
& \quad n) \\
& = \ \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \quad \mathit{flags}, \\
& \quad \text{read-mem}\,(\mathit{addr\text{-}out},\ \mathit{mem}), \\
& \quad \mathit{b\text{-}reg}, \\
& \quad \mathit{i\text{-}reg}, \\
& \quad \mathit{data\text{-}out}, \\
& \quad \mathit{addr\text{-}out}, \\
& \quad \mathbf{t}, \\
& \quad \mathbf{f}, \\
& \quad \mathit{hold\text{-}}, \\
& \quad \mathit{pc\text{-}reg}, \\
& \quad \text{if*}\,(\text{reg-direct-p}\,(\text{mode-b}\,(\mathit{i\text{-}reg})), \\
& \qquad \text{cv\_update}\,(\mathbf{t}, \\
& \qquad\quad \mathit{last\text{-}pc\text{-}reg}, \\
& \qquad\quad \mathit{i\text{-}reg}, \\
& \qquad\quad \mathit{flags}, \\
& \qquad\quad \mathit{pc\text{-}reg}), \\
& \qquad \text{if*}\,(\text{unary-op-code-p}\,(\text{op-code}\,(\mathit{i\text{-}reg})), \\
& \qquad\quad \text{if*}\,(\text{store-resultp}\,(\text{store-cc}\,(\mathit{i\text{-}reg}), \\
& \qquad\qquad \mathit{flags}),
\end{array}
$$

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 \not\simeq 0$)
∧ run-inputs-p (*inputs*, 1)
∧ boolp (**t**)
∧ boolp (*hold-*)
∧ (*reset-* = **t**)

$$
\begin{array}{ll}
\wedge & \textbf{nil}) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \mathit{flags}, \mathit{a\text{-}reg}, \mathit{b\text{-}reg}, \mathit{i\text{-}reg}, \mathit{data\text{-}out}, \mathit{addr\text{-}out}, \mathit{reset\text{-}}, \mathbf{t}, \mathit{hold\text{-}}, \mathit{pc\text{-}reg}, \\
& \quad \text{cv\_reada3}\,(\mathbf{t}, \mathit{last\text{-}regs\text{-}address}, \mathit{last\text{-}i\text{-}reg}, \mathit{last\text{-}flags}, \mathit{last\text{-}pc\text{-}reg})), \\
& \quad \text{list}\,(\mathit{mem}, \texttt{1}, \mathit{clock}, \mathit{mem\text{-}count}, \mathbf{f}, \mathbf{t}, \mathit{addr\text{-}out}, \mathit{last\text{-}data})), \\
& \quad \mathit{inputs}, \\
& \quad n) \\
& = \ \textbf{if}\ \mathit{mem\text{-}count} \simeq \texttt{0} \\
& \quad \textbf{then}\ \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f}, \texttt{0}, \mathit{regs}, \texttt{0}), \\
& \qquad \mathit{flags}, \\
& \qquad \text{read-regs}\,(\mathit{last\text{-}pc\text{-}reg}, \mathit{regs}), \\
& \qquad \mathit{b\text{-}reg}, \mathit{i\text{-}reg}, \mathit{data\text{-}out}, \mathit{addr\text{-}out}, \mathbf{t}, \mathbf{f}, \mathit{hold\text{-}}, \mathit{pc\text{-}reg}, \\
& \qquad \text{cv\_reada3}\,(\mathbf{t}, \mathit{last\text{-}pc\text{-}reg}, \mathit{i\text{-}reg}, \mathit{flags}, \mathit{pc\text{-}reg})), \\
& \qquad \text{list}\,(\mathit{mem}, \texttt{1}, \mathit{clock}, \texttt{0}, \mathbf{t}, \mathbf{t}, \mathit{addr\text{-}out},
\end{array}
$$

```
                        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{array}{ll}
\wedge & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\wedge & \text{boolp}\,(\mathit{dtack\text{-}}) \\
\wedge & \text{boolp}\,(\mathit{hold\text{-}}) \\
\wedge & (\mathit{reset\text{-}} = \mathbf{t}) \\
\wedge & ((\mathit{mem\text{-}state} = \mathtt{0}) \vee (\mathit{mem\text{-}state} = \mathtt{1}))) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
& \qquad \mathit{flags}, \\
& \qquad \mathit{a\text{-}reg}, \\
& \qquad \mathit{b\text{-}reg}, \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \mathit{addr\text{-}out}, \\
& \qquad \mathit{reset\text{-}}, \\
& \qquad \mathit{dtack\text{-}}, \\
& \qquad \mathit{hold\text{-}}, \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \text{cv\_readb0}\,(\mathbf{t}, \\
& \qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad \mathit{last\text{-}flags}, \\
& \qquad\qquad \mathit{last\text{-}pc\text{-}reg})), \\
& \quad \text{list}\,(\mathit{mem}, \\
& \qquad \mathit{mem\text{-}state}, \\
& \qquad \mathit{clock}, \\
& \qquad \mathit{mem\text{-}count}, \\
& \qquad \mathit{mem\text{-}dtack}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathit{last\text{-}address}, \\
& \qquad \mathit{last\text{-}data})), \\
& \quad \mathit{inputs}, \\
& \quad n) \\
= & \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad \mathit{flags}, \\
& \qquad \mathit{a\text{-}reg}, \\
& \qquad \text{read-regs}\,(\text{rn-b}\,(\mathit{last\text{-}i\text{-}reg}),\ \mathit{regs}), \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \text{if*}\,(\text{pre-dec-p}\,(\text{mode-b}\,(\mathit{last\text{-}i\text{-}reg})), \\
& \qquad\quad \text{v-dec}\,(\text{read-regs}\,(\text{rn-b}\,(\mathit{last\text{-}i\text{-}reg}), \\
& \qquad\qquad\qquad \mathit{regs})), \\
& \qquad\quad \text{read-regs}\,(\text{rn-b}\,(\mathit{last\text{-}i\text{-}reg}),\ \mathit{regs})), \\
& \qquad \mathbf{t}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathit{hold\text{-}},
\end{array}
$$

```
                    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,
```

$\mathbf{t}$,
$\mathbf{t}$,
addr-out,
v-threefix (*last-data*))),
cdr (*inputs*),
$n - 1$))

THEOREM: readb2$step
(cv-hyps ($\mathbf{t}$, *last-regs-address*, *last-i-reg*, *last-flags*, *last-pc-reg*)
∧ cv-hyps ($\mathbf{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 \not\simeq 0$)
∧ run-inputs-p (*inputs*, 1)
∧ boolp (*dtack-*)
∧ boolp (*hold-*)
∧ (*reset-* = $\mathbf{t}$)
∧ **nil**)
→ (run-fm9001 (list (list (*regs*,
flags,
a-reg,
b-reg,
i-reg,
data-out,
addr-out,
reset-,
dtack-,
hold-,
pc-reg,
cv_readb2 ($\mathbf{t}$,
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) ≃ 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 ≄ 0)
  ∧    run-inputs-p (inputs, 1)
  ∧    boolp (f)
  ∧    boolp (hold-)
  ∧    (reset- = t)
```

$$
\begin{array}{l}
\wedge \quad (\mathit{mem\text{-}count} \simeq \mathtt{0})) \\
\rightarrow \quad (\mathrm{run\text{-}fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathit{regs}, \\
\qquad\qquad\qquad\qquad \mathit{flags}, \\
\qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
\qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
\qquad\qquad\qquad\qquad \mathit{reset\text{-}}, \\
\qquad\qquad\qquad\qquad \mathbf{f}, \\
\qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathrm{cv\_readb3}\,(\mathtt{t}, \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}regs\text{-}address}, \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}flags}, \\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg})), \\
\qquad\qquad\qquad \mathrm{list}\,(\mathit{mem},\ \mathtt{1},\ \mathit{clock},\ \mathit{mem\text{-}count},\ \mathtt{t},\ \mathtt{t},\ \mathit{addr\text{-}out},\ \mathit{last\text{-}data})), \\
\qquad\qquad\quad \mathit{inputs}, \\
\qquad\qquad\quad n) \\
\quad = \quad \mathrm{run\text{-}fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathrm{write\text{-}regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
\qquad\qquad\qquad\qquad\quad \mathit{flags}, \\
\qquad\qquad\qquad\qquad\quad \mathit{a\text{-}reg}, \\
\qquad\qquad\qquad\qquad\quad \mathrm{read\text{-}mem}\,(\mathit{addr\text{-}out},\ \mathit{mem}), \\
\qquad\qquad\qquad\qquad\quad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\quad \mathit{data\text{-}out}, \\
\qquad\qquad\qquad\qquad\quad \mathit{addr\text{-}out}, \\
\qquad\qquad\qquad\qquad\quad \mathtt{t}, \\
\qquad\qquad\qquad\qquad\quad \mathbf{f}, \\
\qquad\qquad\qquad\qquad\quad \mathit{hold\text{-}}, \\
\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad\quad \mathrm{if^*}\,(\mathrm{store\text{-}resultp}\,(\mathrm{store\text{-}cc}\,(\mathit{i\text{-}reg}),\ \mathit{flags}), \\
\qquad\qquad\qquad\qquad\qquad \mathrm{cv\_write0}\,(\mathtt{t}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg}), \\
\qquad\qquad\qquad\qquad\qquad \mathrm{cv\_update}\,(\mathtt{t}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg}))), \\
\qquad\qquad\qquad\qquad \mathrm{list}\,(\mathit{mem}, \\
\qquad\qquad\qquad\qquad\quad \mathtt{1},
\end{array}
$$

```
                                 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 ≃ 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,
```

$$
\begin{array}{l}
\hspace{17em} \textit{last-regs-address},\\
\hspace{17em} \textit{last-i-reg},\\
\hspace{17em} \textit{last-flags},\\
\hspace{17em} \textit{last-pc-reg})),\\
\hspace{8em} \text{list}\,(\textit{mem},\\
\hspace{10em} \textit{mem-state},\\
\hspace{10em} \textit{clock},\\
\hspace{10em} \textit{mem-count},\\
\hspace{10em} \textit{mem-dtack},\\
\hspace{10em} \mathbf{t},\\
\hspace{10em} \textit{last-address},\\
\hspace{10em} \textit{last-data})),\\
\hspace{6em} \textit{inputs},\\
\hspace{6em} n)\\
= \quad \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \textit{regs},\ \mathtt{0}),\\
\hspace{12em} \text{update-flags}\,(\textit{flags},\\
\hspace{18em} \text{set-flags}\,(\textit{last-i-reg}),\\
\hspace{18em} \text{v-alu}\,(\text{c-flag}\,(\textit{last-flags}),\\
\hspace{21em} \textit{a-reg},\\
\hspace{21em} \textit{b-reg},\\
\hspace{21em} \text{op-code}\,(\textit{last-i-reg}))),\\
\hspace{12em} \textit{a-reg},\\
\hspace{12em} \text{read-regs}\,(\text{rn-b}\,(\textit{last-i-reg}),\ \textit{regs}),\\
\hspace{12em} \textit{i-reg},\\
\hspace{12em} \text{bv}\,(\text{v-alu}\,(\text{c-flag}\,(\textit{last-flags}),\\
\hspace{16.5em} \textit{a-reg},\\
\hspace{16.5em} \textit{b-reg},\\
\hspace{16.5em} \text{op-code}\,(\textit{last-i-reg}))),\\
\hspace{12em} \text{if*}\,(\text{pre-dec-p}\,(\text{mode-b}\,(\textit{last-i-reg})),\\
\hspace{13.5em} \text{v-dec}\,(\text{read-regs}\,(\text{rn-b}\,(\textit{last-i-reg}),\\
\hspace{21em} \textit{regs})),\\
\hspace{13.5em} \text{read-regs}\,(\text{rn-b}\,(\textit{last-i-reg}),\ \textit{regs})),\\
\hspace{12em} \mathbf{t},\\
\hspace{12em} \mathbf{t},\\
\hspace{12em} \textit{hold-},\\
\hspace{12em} \textit{pc-reg},\\
\hspace{12em} \text{cv\_write1}\,(\mathbf{t},\\
\hspace{16.5em} \text{rn-b}\,(\textit{last-i-reg}),\\
\hspace{16.5em} \textit{i-reg},\\
\hspace{16.5em} \textit{flags},\\
\hspace{16.5em} \textit{pc-reg})),\\
\hspace{10em} \text{list}\,(\textit{mem},\\
\hspace{12em} \mathtt{0},\\
\hspace{12em} \textit{clock},
\end{array}
$$

```
                                    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) ≃ 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 \not\simeq 0$)
∧ run-inputs-p (*inputs*, 1)
∧ boolp (**f**)
∧ boolp (*hold-*)
∧ (*reset-* = **t**)
∧ (*mem-count* $\simeq$ 0))
→ (run-fm9001 (list (list (list (*regs*,
flags,
a-reg,
b-reg,
i-reg,

$$
\begin{array}{l}
\qquad\qquad\qquad\quad \mathit{data\text{-}out}, \\
\qquad\qquad\qquad\quad \mathit{addr\text{-}out}, \\
\qquad\qquad\qquad\quad \mathit{reset\text{-}}, \\
\qquad\qquad\qquad\quad \mathbf{f}, \\
\qquad\qquad\qquad\quad \mathit{hold\text{-}}, \\
\qquad\qquad\qquad\quad \mathit{pc\text{-}reg}, \\
\qquad\qquad\qquad\quad \text{cv\_write3}\,(\mathbf{f}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}flags}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}pc\text{-}reg})), \\
\qquad\qquad\qquad \text{list}\,(\mathit{mem}, \mathtt{2}, \mathit{clock}, \mathit{mem\text{-}count}, \mathbf{t}, \mathbf{f}, \mathit{addr\text{-}out}, \mathit{data\text{-}out})), \\
\qquad\qquad \mathit{inputs}, \\
\qquad\qquad n) \\
= \quad \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f}, \mathtt{0}, \mathit{regs}, \mathtt{0}), \\
\qquad\qquad\qquad\qquad \mathit{flags}, \\
\qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
\qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
\qquad\qquad\qquad\qquad \mathbf{t}, \\
\qquad\qquad\qquad\qquad \mathbf{f}, \\
\qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad \text{cv\_fetch0}\,(\mathbf{f}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}pc\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg})), \\
\qquad\qquad\qquad \text{list}\,(\mathit{mem}, \mathtt{2}, \mathit{clock}, \mathtt{0}, \mathbf{t}, \mathbf{f}, \mathit{addr\text{-}out}, \mathit{data\text{-}out})), \\
\qquad\qquad \text{cdr}\,(\mathit{inputs}), \\
\qquad\qquad n - 1))
\end{array}
$$

THEOREM: write3$step1

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{cv-hyps}\,(\mathbf{f}, \mathit{last\text{-}regs\text{-}address}, \mathit{last\text{-}i\text{-}reg}, \mathit{last\text{-}flags}, \mathit{last\text{-}pc\text{-}reg})} \\
\land & \text{cv-hyps}\,(\mathbf{t}, \mathit{pc\text{-}reg}, \mathit{i\text{-}reg}, \mathit{flags}, \mathit{pc\text{-}reg}) \\
\land & \text{memory-okp}\,(\mathtt{32}, \mathtt{32}, \mathit{mem}) \\
\land & \text{regfile-okp}\,(\mathit{regs}) \\
\land & \text{bvp}\,(\mathit{a\text{-}reg}) \\
\land & (\text{length}\,(\mathit{a\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{b\text{-}reg}) \\
\land & (\text{length}\,(\mathit{b\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{addr\text{-}out})
\end{array}
$$

$$
\begin{array}{ll}
\land & (\mathrm{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \mathrm{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\mathrm{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & (n \not\simeq \mathtt{0}) \\
\land & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\land & \mathrm{boolp}\,(\mathbf{t}) \\
\land & \mathrm{boolp}\,(\mathit{hold\text{-}}) \\
\land & (\mathit{reset\text{-}} = \mathbf{t}) \\
\land & \mathbf{nil}) \\
\to & (\text{run-fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathit{regs}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{reset\text{-}}, \\
& \qquad\qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad \text{cv\_write3}\,(\mathbf{f}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}flags}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}pc\text{-}reg})), \\
& \qquad\qquad\qquad\qquad \mathrm{list}\,(\mathit{mem},\ \mathtt{2},\ \mathit{clock},\ \mathit{mem\text{-}count},\ \mathbf{f},\ \mathbf{f},\ \mathit{addr\text{-}out},\ \mathit{data\text{-}out})), \\
& \qquad\qquad\qquad \mathit{inputs}, \\
& \qquad\qquad\qquad n) \\
& \quad = \ \ \mathbf{if}\ \mathit{mem\text{-}count} \simeq \mathtt{0} \\
& \qquad\quad \mathbf{then}\ \text{run-fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbf{f}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{cv\_write3}\,(\mathbf{f}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{last\text{-}pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{flags},
\end{array}
$$

```
                                                    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*)

$$
\begin{array}{ll}
\wedge & (\text{length}\,(a\text{-}reg) = \mathtt{32}) \\
\wedge & \text{bvp}\,(b\text{-}reg) \\
\wedge & (\text{length}\,(b\text{-}reg) = \mathtt{32}) \\
\wedge & \text{bvp}\,(addr\text{-}out) \\
\wedge & (\text{length}\,(addr\text{-}out) = \mathtt{32}) \\
\wedge & \text{bvp}\,(data\text{-}out) \\
\wedge & (\text{length}\,(data\text{-}out) = \mathtt{32}) \\
\wedge & (n \not\simeq \mathtt{0}) \\
\wedge & \text{run-inputs-p}\,(inputs,\ \mathtt{1}) \\
\wedge & \text{boolp}\,(dtack\text{-}) \\
\wedge & \text{boolp}\,(hold\text{-}) \\
\wedge & (reset\text{-} = \mathbf{t}) \\
\wedge & \mathbf{nil}) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(regs, \\
& \quad flags, \\
& \quad a\text{-}reg, \\
& \quad b\text{-}reg, \\
& \quad i\text{-}reg, \\
& \quad data\text{-}out, \\
& \quad addr\text{-}out, \\
& \quad reset\text{-}, \\
& \quad dtack\text{-}, \\
& \quad hold\text{-}, \\
& \quad pc\text{-}reg, \\
& \quad \text{cv\_sefa0}\,(\mathbf{t}, \\
& \qquad last\text{-}regs\text{-}address, \\
& \qquad last\text{-}i\text{-}reg, \\
& \qquad last\text{-}flags, \\
& \qquad last\text{-}pc\text{-}reg)), \\
& \quad \text{list}\,(mem, \\
& \qquad \mathtt{0}, \\
& \qquad clock, \\
& \qquad mem\text{-}count, \\
& \qquad mem\text{-}dtack, \\
& \qquad \mathbf{t}, \\
& \qquad last\text{-}address, \\
& \qquad last\text{-}data)), \\
& \quad inputs, \\
& \quad n) \\
& = \ \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ regs,\ \mathtt{0}), \\
& \qquad flags, \\
& \qquad \text{read-regs}\,(\text{rn-a}\,(last\text{-}i\text{-}reg),\ regs), \\
& \qquad b\text{-}reg, \\
& \qquad i\text{-}reg,
\end{array}
$$

```
                                  data-out,
                                  addr-out,
                                  t,
                                  t,
                                  hold-,
                                  pc-reg,
                                  cv_sefa1 (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: sefa1$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_sefa1 (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*)

$$
\begin{array}{ll}
\land & (\text{length}\,(\mathit{b\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{addr\text{-}out}) \\
\land & (\text{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\text{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & (n \not\simeq \mathtt{0}) \\
\land & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\land & \text{boolp}\,(\mathit{dtack\text{-}}) \\
\land & \text{boolp}\,(\mathit{hold\text{-}}) \\
\land & (\mathit{reset\text{-}} = \mathbf{t}) \\
\land & \mathbf{nil}) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
& \qquad\qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathit{reset\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{dtack\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \text{cv\_sefb0}\,(\mathbf{t}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}flags}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}pc\text{-}reg})), \\
& \qquad\qquad\qquad \text{list}\,(\mathit{mem}, \\
& \qquad\qquad\qquad\qquad \mathtt{0}, \\
& \qquad\qquad\qquad\qquad \mathit{clock}, \\
& \qquad\qquad\qquad\qquad \mathit{mem\text{-}count}, \\
& \qquad\qquad\qquad\qquad \mathit{mem\text{-}dtack}, \\
& \qquad\qquad\qquad\qquad \mathbf{t}, \\
& \qquad\qquad\qquad\qquad \mathit{last\text{-}address}, \\
& \qquad\qquad\qquad\qquad \mathit{last\text{-}data})), \\
& \qquad\qquad \mathit{inputs}, \\
& \qquad\qquad n) \\
& = \ \text{run-fm9001}\,(\text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad\qquad\qquad\qquad\quad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\quad \text{read-regs}\,(\text{rn-b}\,(\mathit{last\text{-}i\text{-}reg}),\ \mathit{regs}), \\
& \qquad\qquad\qquad\qquad\quad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad\quad \mathit{addr\text{-}out},
\end{array}
$$

```
                    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** $n \simeq 0$ **then** *reg0*
else read-fn (*reg1*, *reg1*, $n - 1$) **endif**

Event: Enable read-fn; name this event 'read-fn-off'.

Theorem: open-read-fn
$((n \simeq 0) \rightarrow (\text{read-fn}(reg0, reg1, n) = reg0))$
$\land \quad ((n \not\simeq 0) \rightarrow (\text{read-fn}(reg0, reg1, n) = \text{read-fn}(reg1, reg1, n - 1)))$

THEOREM: length-read-fn
$((\text{length}(reg0) = 32) \land (\text{length}(reg1) = 32))$
$\rightarrow \quad (\text{length}(\text{read-fn}(reg0,\ reg1,\ n)) = 32)$

THEOREM: properp-read-fn
$(\text{properp}(reg0) \land \text{properp}(reg1)) \rightarrow \text{properp}(\text{read-fn}(reg0,\ reg1,\ n))$

THEOREM: bvp-read-fn
$(\text{bvp}(reg0) \land \text{bvp}(reg1)) \rightarrow \text{bvp}(\text{read-fn}(reg0,\ reg1,\ n))$

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 \not\simeq 0$))
→ (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 − 1, f, t, addr-out, last-data)),
        inputs,
        count)
=   list (list (write-regs (f, 0, regs, 0),
                flags,
                a-reg,
                b-reg,
                read-fn (i-reg, read-regs (pc-reg, regs), count),
                data-out,
                addr-out,
                t,
                f,
                hold-,
                pc-reg,
                cv_fetch3 (t,
                           pc-reg,
                           read-fn (i-reg,
                                    read-regs (pc-reg, regs),
                                    count − 1),
                           flags,
                           pc-reg)),
          list (mem, 1, clock, 0, t, t, addr-out, v-threefix (last-data))))
```

Definition:

```
reada3$induction (n,
                  clock,
                  count,
                  inputs,
                  regs,
                  a-reg,
                  b-reg,
                  rw-,
                  regs-address,
                  i-reg,
                  flags,
```

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad\qquad \textit{last-rw-}, \\
\qquad\qquad\qquad\qquad \textit{last-regs-address}, \\
\qquad\qquad\qquad\qquad \textit{last-i-reg}, \\
\qquad\qquad\qquad\qquad \textit{last-flags}, \\
\qquad\qquad\qquad\qquad \textit{last-data}) \\
= \quad \textbf{if } n \simeq 0 \textbf{ then t} \\
\qquad \textbf{else } \text{reada3\$induction}\,(n-1, \\
\qquad\qquad\qquad\qquad\qquad \textit{clock}, \\
\qquad\qquad\qquad\qquad\qquad \textit{count} - 1, \\
\qquad\qquad\qquad\qquad\qquad \text{cdr}\,(\textit{inputs}), \\
\qquad\qquad\qquad\qquad\qquad \text{write-regs}\,(\mathbf{f},\ 0,\ \textit{regs},\ 0), \\
\qquad\qquad\qquad\qquad\qquad \text{read-regs}\,(\textit{pc-reg},\ \textit{regs}), \\
\qquad\qquad\qquad\qquad\qquad \textit{b-reg}, \\
\qquad\qquad\qquad\qquad\qquad \mathbf{t}, \\
\qquad\qquad\qquad\qquad\qquad \textit{regs-address}, \\
\qquad\qquad\qquad\qquad\qquad \textit{i-reg}, \\
\qquad\qquad\qquad\qquad\qquad \textit{flags}, \\
\qquad\qquad\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad\qquad\qquad \textit{rw-}, \\
\qquad\qquad\qquad\qquad\qquad \textit{pc-reg}, \\
\qquad\qquad\qquad\qquad\qquad \textit{i-reg}, \\
\qquad\qquad\qquad\qquad\qquad \textit{flags}, \\
\qquad\qquad\qquad\qquad\qquad \text{v-threefix}\,(\textit{last-data}))\ \textbf{endif}
\end{array}
$$

THEOREM: reada3$progress-help

$$
\begin{array}{ll}
\multicolumn{2}{l}{(\text{cv-hyps}\,(\mathbf{t},\ \textit{last-regs-address},\ \textit{last-i-reg},\ \textit{last-flags},\ \textit{pc-reg})} \\
\land & \text{cv-hyps}\,(\mathbf{t},\ \textit{last-regs-address},\ \textit{i-reg},\ \textit{flags},\ \textit{pc-reg}) \\
\land & \text{memory-okp}\,(32,\ 32,\ \textit{mem}) \\
\land & \text{regfile-okp}\,(\textit{regs}) \\
\land & \text{bvp}\,(\textit{a-reg}) \\
\land & (\text{length}\,(\textit{a-reg}) = 32) \\
\land & \text{bvp}\,(\textit{b-reg}) \\
\land & (\text{length}\,(\textit{b-reg}) = 32) \\
\land & \text{bvp}\,(\textit{addr-out}) \\
\land & (\text{length}\,(\textit{addr-out}) = 32) \\
\land & \text{bvp}\,(\textit{data-out}) \\
\land & (\text{length}\,(\textit{data-out}) = 32) \\
\land & \text{boolp}\,(\textit{hold-}) \\
\land & \text{run-inputs-p}\,(\textit{inputs},\ 1 + n) \\
\land & \textbf{if } \textit{count} < n \textbf{ then f} \\
 & \textbf{else t endif}) \\
\rightarrow & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\textit{regs}, \\
 & \qquad\qquad\qquad\qquad \textit{flags},
\end{array}
$$

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*)

$$
\begin{array}{ll}
\land & \text{regfile-okp}\,(\mathit{regs}) \\
\land & \text{bvp}\,(\mathit{a\text{-}reg}) \\
\land & (\text{length}\,(\mathit{a\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{b\text{-}reg}) \\
\land & (\text{length}\,(\mathit{b\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{addr\text{-}out}) \\
\land & (\text{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\text{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & \text{boolp}\,(\mathit{hold\text{-}}) \\
\land & \text{run-inputs-p}\,(\mathit{inputs},\ \mathit{count}) \\
\land & (\mathit{count} \not\simeq \mathtt{0})) \\
\to & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
& \qquad \mathit{flags}, \\
& \qquad \mathit{a\text{-}reg}, \\
& \qquad \mathit{b\text{-}reg}, \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \mathit{addr\text{-}out}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathit{hold\text{-}}, \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \text{cv\_reada3}\,(\mathbf{t}, \\
& \qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad \mathit{last\text{-}flags}, \\
& \qquad\qquad \mathit{pc\text{-}reg})), \\
& \quad \text{list}\,(\mathit{mem},\ \mathtt{1},\ \mathit{clock},\ \mathit{count} - 1,\ \mathbf{f},\ \mathbf{t},\ \mathit{addr\text{-}out},\ \mathit{last\text{-}data})), \\
& \quad \mathit{inputs}, \\
& \quad \mathit{count}) \\
& = \ \text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad \mathit{flags}, \\
& \qquad \text{read-fn}\,(\mathit{a\text{-}reg},\ \text{read-regs}\,(\mathit{pc\text{-}reg},\ \mathit{regs}),\ \mathit{count}), \\
& \qquad \mathit{b\text{-}reg}, \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \mathit{addr\text{-}out}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathbf{f}, \\
& \qquad \mathit{hold\text{-}}, \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \text{cv\_reada3}\,(\mathbf{t},\ \mathit{pc\text{-}reg},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})), \\
& \quad \text{list}\,(\mathit{mem},\ \mathtt{1},\ \mathit{clock},\ \mathtt{0},\ \mathbf{t},\ \mathbf{t},\ \mathit{addr\text{-}out},\ \text{v-threefix}\,(\mathit{last\text{-}data}))))
\end{array}
$$

DEFINITION:
readb3$induction ($n$,
$clock$,
$count$,
$inputs$,
$regs$,
$a\text{-}reg$,
$b\text{-}reg$,
$rw\text{-}$,
$regs\text{-}address$,
$i\text{-}reg$,
$flags$,
$pc\text{-}reg$,
$last\text{-}rw\text{-}$,
$last\text{-}regs\text{-}address$,
$last\text{-}i\text{-}reg$,
$last\text{-}flags$,
$last\text{-}data$)

= **if** $n \simeq 0$ **then** **t**
else readb3$induction ($n - 1$,
$clock$,
$count - 1$,
cdr ($inputs$),
write-regs (**f**, 0, $regs$, 0),
$a\text{-}reg$,
read-regs ($pc\text{-}reg$, $regs$),
t,
$regs\text{-}address$,
$i\text{-}reg$,
$flags$,
$pc\text{-}reg$,
$rw\text{-}$,
$pc\text{-}reg$,
$i\text{-}reg$,
$flags$,
v-threefix ($last\text{-}data$)) **endif**

THEOREM: readb3$progress-help
(cv-hyps (**t**, $last\text{-}regs\text{-}address$, $last\text{-}i\text{-}reg$, $last\text{-}flags$, $pc\text{-}reg$)
$\land$ cv-hyps (**t**, $last\text{-}regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)
$\land$ memory-okp (32, 32, mem)
$\land$ regfile-okp ($regs$)
$\land$ bvp ($a\text{-}reg$)
$\land$ (length ($a\text{-}reg$) = 32)

$\land$ bvp (*b-reg*)
$\land$ (length (*b-reg*) $=$ `32`)
$\land$ bvp (*addr-out*)
$\land$ (length (*addr-out*) $=$ `32`)
$\land$ bvp (*data-out*)
$\land$ (length (*data-out*) $=$ `32`)
$\land$ boolp (*hold-*)
$\land$ run-inputs-p (*inputs*, $1 + n$)
$\land$ **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,
$addr\text{-}out$,
v-threefix ($last\text{-}data$))))

THEOREM: readb3$progress
(cv-hyps (**t**, $last\text{-}regs\text{-}address$, $last\text{-}i\text{-}reg$, $last\text{-}flags$, $pc\text{-}reg$)
$\land$ cv-hyps (**t**, $last\text{-}regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)
$\land$ memory-okp (32, 32, mem)
$\land$ regfile-okp ($regs$)
$\land$ bvp ($a\text{-}reg$)
$\land$ (length ($a\text{-}reg$) = 32)
$\land$ bvp ($b\text{-}reg$)
$\land$ (length ($b\text{-}reg$) = 32)
$\land$ bvp ($addr\text{-}out$)
$\land$ (length ($addr\text{-}out$) = 32)
$\land$ bvp ($data\text{-}out$)
$\land$ (length ($data\text{-}out$) = 32)
$\land$ boolp ($hold\text{-}$)
$\land$ run-inputs-p ($inputs$, $count$)
$\land$ ($count \not\simeq 0$))
$\to$ (run-fm9001 (list (list ($regs$,
$flags$,
$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,
t,
t,
$hold\text{-}$,
$pc\text{-}reg$,
cv_readb3 (**t**,
$last\text{-}regs\text{-}address$,
$last\text{-}i\text{-}reg$,
$last\text{-}flags$,
$pc\text{-}reg$)),
list (mem, 1, $clock$, $count - 1$, **f**, **t**, $addr\text{-}out$, $last\text{-}data$)),
$inputs$,
$count$)
= 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*)
$\land$ cv-hyps (**t**, *last-regs-address*, *i-reg*, *flags*, *pc-reg*)
$\land$ memory-okp (32, 32, *mem*)
$\land$ regfile-okp (*regs*)
$\land$ bvp (*a-reg*)
$\land$ (length (*a-reg*) = 32)
$\land$ bvp (*b-reg*)
$\land$ (length (*b-reg*) = 32)
$\land$ bvp (*addr-out*)
$\land$ (length (*addr-out*) = 32)
$\land$ bvp (*data-out*)
$\land$ (length (*data-out*) = 32)
$\land$ boolp (*hold-*)
$\land$ run-inputs-p (*inputs*, 1 + *n*)
$\land$ **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_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,

$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,
t,
¬ **if** $n < count$ **then f**
else t endif,
$hold\text{-}$,
$pc\text{-}reg$,
cv_write3 (**f**, $pc\text{-}reg$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)),
list (mem,
2,
$clock$,
$count - (1 + n)$,
if $n < count$ **then f**
else t endif,
f,
$addr\text{-}out$,
$data\text{-}out$)))

THEOREM: write3\$progress
(cv-hyps (**t**, $last\text{-}regs\text{-}address$, $last\text{-}i\text{-}reg$, $last\text{-}flags$, $pc\text{-}reg$)
∧ cv-hyps (**t**, $last\text{-}regs\text{-}address$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)
∧ memory-okp (32, 32, mem)
∧ regfile-okp ($regs$)
∧ bvp ($a\text{-}reg$)
∧ (length ($a\text{-}reg$) = 32)
∧ bvp ($b\text{-}reg$)
∧ (length ($b\text{-}reg$) = 32)
∧ bvp ($addr\text{-}out$)
∧ (length ($addr\text{-}out$) = 32)
∧ bvp ($data\text{-}out$)
∧ (length ($data\text{-}out$) = 32)
∧ boolp ($hold\text{-}$)
∧ run-inputs-p ($inputs$, $count$)
∧ ($count \not\simeq 0$))
→ (run-fm9001 (list (list ($regs$,
$flags$,
$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,

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** $n \simeq 0$ **then t**
elseif $n \not\simeq 0$ **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*)

$$
\begin{array}{ll}
\wedge & (\mathrm{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\wedge & \mathrm{boolp}\,(\mathit{dtack\text{-}}) \\
\wedge & \mathrm{boolp}\,(\mathit{hold\text{-}}) \\
\wedge & \text{run-inputs-p}\,(\mathit{inputs},\ \mathtt{1}) \\
\wedge & (\mathit{reset\text{-}} = \mathbf{t}) \\
\wedge & \mathbf{nil}) \\
\rightarrow & (\text{run-fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathit{regs}, \\
& \qquad \mathit{flags}, \\
& \qquad \mathit{a\text{-}reg}, \\
& \qquad \mathit{b\text{-}reg}, \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \mathit{addr\text{-}out}, \\
& \qquad \mathit{reset\text{-}}, \\
& \qquad \mathit{dtack\text{-}}, \\
& \qquad \mathit{hold\text{-}}, \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \mathrm{cv\_fetch0}\,(\mathbf{t}, \\
& \qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad \mathit{last\text{-}flags}, \\
& \qquad\qquad \mathit{pc\text{-}reg})), \\
& \quad \mathrm{list}\,(\mathit{mem}, \\
& \qquad \mathtt{0}, \\
& \qquad \mathit{clock}, \\
& \qquad \mathit{mem\text{-}count}, \\
& \qquad \mathit{mem\text{-}dtack}, \\
& \qquad \mathbf{t}, \\
& \qquad \mathit{last\text{-}address}, \\
& \qquad \mathit{last\text{-}data})), \\
& \quad \mathit{inputs}, \\
& \quad \mathtt{1}) \\
& = \ \mathrm{list}\,(\mathrm{list}\,(\text{write-regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}), \\
& \qquad \mathit{flags}, \\
& \qquad \text{read-regs}\,(\mathit{pc\text{-}reg},\ \mathit{regs}), \\
& \qquad \mathit{b\text{-}reg}, \\
& \qquad \mathit{i\text{-}reg}, \\
& \qquad \mathit{data\text{-}out}, \\
& \qquad \text{read-regs}\,(\mathit{pc\text{-}reg},\ \mathit{regs}), \\
& \qquad \mathbf{t}, \\
& \qquad \mathbf{t}, \\
& \qquad \text{f-buf}\,(\text{f-buf}\,(\text{hold--input}\,(\mathrm{car}\,(\mathit{inputs})))), \\
& \qquad \mathit{pc\text{-}reg}, \\
& \qquad \mathrm{cv\_fetch1}\,(\mathbf{t},\ \mathit{pc\text{-}reg},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg})),
\end{array}
$$

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 ($\mathbf{f}$, 0, $regs$, 0),
$flags$,
read-regs ($pc\text{-}reg$, $regs$),
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
read-regs ($pc\text{-}reg$, $regs$),
$\mathbf{t}$,
$\mathbf{t}$,
f-buf (f-buf (hold–input (car ($inputs$)))),
$pc\text{-}reg$,
cv_fetch1 ($\mathbf{t}$, $pc\text{-}reg$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)),
list (mem, 0, $clock$, 0, $\mathbf{t}$, $\mathbf{t}$, $addr\text{-}out$, v-threefix ($last\text{-}data$))))

THEOREM: fetch0->fetch0$sim2
(cv-hyps ($\mathbf{f}$, $last\text{-}regs\text{-}address$, $last\text{-}i\text{-}reg$, $last\text{-}flags$, $pc\text{-}reg$)
$\land$ cv-hyps ($\mathbf{t}$, $pc\text{-}reg$, $i\text{-}reg$, $flags$, $pc\text{-}reg$)
$\land$ memory-okp (32, 32, mem)
$\land$ regfile-okp ($regs$)
$\land$ bvp ($a\text{-}reg$)
$\land$ (length ($a\text{-}reg$) $= 32$)
$\land$ bvp ($b\text{-}reg$)
$\land$ (length ($b\text{-}reg$) $= 32$)
$\land$ bvp ($addr\text{-}out$)
$\land$ (length ($addr\text{-}out$) $= 32$)
$\land$ bvp ($data\text{-}out$)
$\land$ (length ($data\text{-}out$) $= 32$)
$\land$ boolp ($dtack\text{-}$)
$\land$ boolp ($hold\text{-}$)
$\land$ run-inputs-p ($inputs$, 1)
$\land$ ($reset\text{-} = \mathbf{t}$)
$\land$ ($mem\text{-}count \simeq 0$))
$\rightarrow$ (run-fm9001 (list (list ($regs$,
$flags$,
$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,
$reset\text{-}$,
$dtack\text{-}$,
$hold\text{-}$,

```
                        pc-reg,
                        cv_fetch0 (f,
                                   last-regs-address,
                                   last-i-reg,
                                   last-flags,
                                   pc-reg)),
                    list (mem,
                          2,
                          clock,
                          mem-count,
                          mem-dtack,
                          f,
                          addr-out,
                          data-out)),
                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 (write-mem (addr-out, mem, data-out),
                0,
                clock,
                0,
                t,
                f,
                addr-out,
                data-out)))
```

THEOREM: fetch1->decode$sim

```
((dtack-wait = car (clock))
 ∧  cv-hyps (last-rw-, 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)
```

$$
\begin{array}{ll}
\land & (\mathrm{length}\,(\mathit{a\text{-}reg}) = \mathtt{32}) \\
\land & \mathrm{bvp}\,(\mathit{b\text{-}reg}) \\
\land & (\mathrm{length}\,(\mathit{b\text{-}reg}) = \mathtt{32}) \\
\land & \mathrm{bvp}\,(\mathit{addr\text{-}out}) \\
\land & (\mathrm{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \mathrm{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\mathrm{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & \mathrm{boolp}\,(\mathit{dtack\text{-}}) \\
\land & \mathrm{boolp}\,(\mathit{hold\text{-}}) \\
\land & \mathrm{run\text{-}inputs\text{-}p}\,(\mathit{inputs},\ 1 + (1 + (\mathrm{car}\,(\mathit{clock}) + \mathtt{2}))) \\
\land & (\mathit{reset\text{-}} = \mathbf{t}) \\
\land & (\mathit{hold\text{-}} = \mathbf{t}) \\
\land & \mathrm{boolp}\,(\mathit{last\text{-}rw\text{-}})) \\
\rightarrow & (\mathrm{run\text{-}fm9001}\,(\mathrm{list}\,(\mathrm{list}\,(\mathit{regs}, \\
& \qquad\qquad\qquad\qquad \mathit{flags}, \\
& \qquad\qquad\qquad\qquad \mathit{a\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{b\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathit{data\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathit{addr\text{-}out}, \\
& \qquad\qquad\qquad\qquad \mathit{reset\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{dtack\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{hold\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad \mathrm{cv\_fetch1}\,(\mathit{last\text{-}rw\text{-}}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}regs\text{-}address}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}i\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{last\text{-}flags}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg})), \\
& \qquad\qquad\qquad \mathrm{list}\,(\mathit{mem}, \\
& \qquad\qquad\qquad\qquad \mathtt{0}, \\
& \qquad\qquad\qquad\qquad \mathit{clock}, \\
& \qquad\qquad\qquad\qquad \mathit{mem\text{-}count}, \\
& \qquad\qquad\qquad\qquad \mathit{mem\text{-}dtack}, \\
& \qquad\qquad\qquad\qquad \mathit{last\text{-}rw\text{-}}, \\
& \qquad\qquad\qquad\qquad \mathit{last\text{-}address}, \\
& \qquad\qquad\qquad\qquad \mathit{last\text{-}data})), \\
& \qquad\qquad \mathit{inputs}, \\
& \qquad\qquad 1 + (1 + (\mathrm{car}\,(\mathit{clock}) + \mathtt{2}))) \\
& = \ \mathrm{list}\,(\mathrm{list}\,(\mathrm{write\text{-}regs}\,(\mathbf{f}, \\
& \qquad\qquad\qquad\qquad \mathtt{0}, \\
& \qquad\qquad\qquad\qquad \mathrm{write\text{-}regs}\,(\mathbf{t}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathit{pc\text{-}reg}, \\
& \qquad\qquad\qquad\qquad\qquad\quad \mathrm{write\text{-}regs}\,(\mathbf{f},\ \mathtt{0},\ \mathit{regs},\ \mathtt{0}),
\end{array}
$$

```
                                        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*)
$\land$ cv-hyps (**t**, *pc-reg*, *i-reg*, *flags*, *pc-reg*)
$\land$ memory-okp (32, 32, *mem*)
$\land$ regfile-okp (*regs*)
$\land$ bvp (*a-reg*)
$\land$ (length (*a-reg*) = 32)
$\land$ bvp (*b-reg*)
$\land$ (length (*b-reg*) = 32)
$\land$ bvp (*addr-out*)
$\land$ (length (*addr-out*) = 32)
$\land$ bvp (*data-out*)
$\land$ (length (*data-out*) = 32)
$\land$ boolp (*dtack-*)
$\land$ boolp (*hold-*)
$\land$ run-inputs-p (*inputs*, 1)
$\land$ (*reset-* = **t**)
$\land$ ((*mem-state* = 0) $\lor$ (*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,
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)
```

$$
\begin{array}{ll}
\land & \text{cv-hyps}\,(\mathtt{t},\ \mathit{pc\text{-}reg},\ \mathit{i\text{-}reg},\ \mathit{flags},\ \mathit{pc\text{-}reg}) \\
\land & \text{memory-okp}\,(\mathtt{32},\ \mathtt{32},\ \mathit{mem}) \\
\land & \text{regfile-okp}\,(\mathit{regs}) \\
\land & \text{bvp}\,(\mathit{a\text{-}reg}) \\
\land & (\text{length}\,(\mathit{a\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{b\text{-}reg}) \\
\land & (\text{length}\,(\mathit{b\text{-}reg}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{addr\text{-}out}) \\
\land & (\text{length}\,(\mathit{addr\text{-}out}) = \mathtt{32}) \\
\land & \text{bvp}\,(\mathit{data\text{-}out}) \\
\land & (\text{length}\,(\mathit{data\text{-}out}) = \mathtt{32}) \\
\land & \text{boolp}\,(\mathit{dtack\text{-}}) \\
\land & \text{boolp}\,(\mathit{hold\text{-}}) \\
\land & \text{run-inputs-p}\,(\mathit{inputs},\ 1 + (1 + (1 + (\text{car}\,(\mathit{clock}) + \mathtt{1})))) \\
\land & (\mathit{reset\text{-}} = \mathbf{t}) \\
\land & ((\mathit{mem\text{-}state} = \mathtt{0}) \lor (\mathit{mem\text{-}state} = \mathtt{1}))) \\
\to & (\text{run-fm9001}\,(\text{list}\,(\text{list}\,(\mathit{regs}, \\
 & \qquad \mathit{flags}, \\
 & \qquad \mathit{a\text{-}reg}, \\
 & \qquad \mathit{b\text{-}reg}, \\
 & \qquad \mathit{i\text{-}reg}, \\
 & \qquad \mathit{data\text{-}out}, \\
 & \qquad \mathit{addr\text{-}out}, \\
 & \qquad \mathit{reset\text{-}}, \\
 & \qquad \mathit{dtack\text{-}}, \\
 & \qquad \mathit{hold\text{-}}, \\
 & \qquad \mathit{pc\text{-}reg}, \\
 & \qquad \text{cv\_readb0}\,(\mathbf{t}, \\
 & \qquad\qquad \mathit{last\text{-}regs\text{-}address}, \\
 & \qquad\qquad \mathit{last\text{-}i\text{-}reg}, \\
 & \qquad\qquad \mathit{last\text{-}flags}, \\
 & \qquad\qquad \mathit{pc\text{-}reg})), \\
 & \quad \text{list}\,(\mathit{mem}, \\
 & \qquad \mathit{mem\text{-}state}, \\
 & \qquad \mathit{clock}, \\
 & \qquad \mathit{mem\text{-}count}, \\
 & \qquad \mathit{mem\text{-}dtack}, \\
 & \qquad \mathbf{t}, \\
 & \qquad \mathit{last\text{-}address}, \\
 & \qquad \mathit{last\text{-}data})), \\
 & \quad \mathit{inputs}, \\
 & \quad 1 + (1 + (1 + (\text{car}\,(\mathit{clock}) + \mathtt{1})))) \\
 & = \ \text{list}\,(\text{list}\,(\text{write-regs}\,(\mathbf{f}, \\
 & \qquad\qquad \mathtt{0},
\end{array}
$$

```
          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**)

```
∧   ((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,
                                        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,
```

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: sefb1->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*, 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_sefb1 (**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 (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 + **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 + **if** b-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))))
\+ **if** store-resultp (store-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))))
\+ **if** reg-direct-p (mode-b (*i-reg*))
then t_update (cdr (*clock*), *i-reg*, *flags*)
elseif unary-op-code-p (op-code (*i-reg*))
then if store-resultp (store-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 + **if** b-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-resultp (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-resultp (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**)
∧ (hold–input (car (*inputs*)) = **t**)
∧ bvp (pc-reg-input (car (*inputs*)))
∧ (length (pc-reg-input (car (*inputs*))) = 4)
∧ 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
(($n \simeq 0$) → (operating-inputs-p (*inputs*, *n*) = **t**))
∧ (($n \not\simeq 0$)
→ (operating-inputs-p (*inputs*, *n*)

$=$ ((reset–input (car (*inputs*)) = **t**)
$\land$ (hold–input (car (*inputs*)) = **t**)
$\land$ bvp (pc-reg-input (car (*inputs*)))
$\land$ (length (pc-reg-input (car (*inputs*))) = 4)
$\land$ operating-inputs-p (cdr (*inputs*), $n - 1$))))

THEOREM: operating-inputs-p-implies-run-inputs-p
operating-inputs-p (*inputs*, *n*) $\rightarrow$ run-inputs-p (*inputs*, *n*)

THEOREM: operating-inputs-p-plus
operating-inputs-p (*inputs*, $n + m$)
$=$ (operating-inputs-p (*inputs*, *n*)
$\land$ operating-inputs-p (nthcdr (*n*, *inputs*), *m*))

THEOREM: operating-inputs-p-1
(operating-inputs-p (*inputs*, *n*) $\land$ ($n \not\simeq 0$))
$\rightarrow$ operating-inputs-p (*inputs*, 1)

THEOREM: operating-inputs-p-implies-hold–input
(operating-inputs-p (*inputs*, *n*) $\land$ ($n \not\simeq 0$))
$\rightarrow$ (hold–input (car (*inputs*)) = **t**)

DEFINITION:
microcycles (*state*)
$=$ t_fetch1 (caddadr (*state*),
read-mem (read-mem (cadddddddddar (*state*), caaar (*state*)),
caadr (*state*)),
cadar (*state*))

DEFINITION:
total-microcycles (*state*, *inputs*, *n*)
$=$ **if** $n \simeq 0$ **then** 0
else microcycles (*state*)
$+$ total-microcycles (run-fm9001 (*state*,
inputs,
microcycles (*state*)),
nthcdr (microcycles (*state*), *inputs*),
$n - 1$) **endif**

EVENT: Enable total-microcycles; name this event 'total-microcycles-off'.

THEOREM: open-total-microcycles
(($n \simeq 0$) $\rightarrow$ (total-microcycles (*state*, *inputs*, *n*) = 0))
$\land$ (($n \not\simeq 0$)
$\rightarrow$ (total-microcycles (*state*, *inputs*, *n*)

$=$ (microcycles ($state$)
$+$ total-microcycles (run-fm9001 ($state$,
$inputs$,
microcycles ($state$)),
nthcdr (microcycles ($state$),
$inputs$),
$n - 1$))))

DEFINITION:
macrocycle-invariant ($state$, pc)
$=$ (all-ramp-mem (4, caaar ($state$))
$\wedge$ memory-okp (4, 32, caaar ($state$))
$\wedge$ (cadaar ($state$) $=$ **f**)
$\wedge$ bvp (caddaar ($state$))
$\wedge$ (length (caddaar ($state$)) $=$ 32)
$\wedge$ bvp (cadddaar ($state$))
$\wedge$ (length (cadddaar ($state$)) $=$ 4)
$\wedge$ bvp (cadar ($state$))
$\wedge$ (length (cadar ($state$)) $=$ 4)
$\wedge$ (caddar ($state$)
$=$ read-regs (caddddddddddar ($state$), caar ($state$)))
$\wedge$ bvp (cadddar ($state$))
$\wedge$ (length (cadddar ($state$)) $=$ 32)
$\wedge$ bvp (caddddar ($state$))
$\wedge$ (length (caddddar ($state$)) $=$ 32)
$\wedge$ bvp (cadddddar ($state$))
$\wedge$ (length (cadddddar ($state$)) $=$ 32)
$\wedge$ (caddddddar ($state$)
$=$ read-regs (caddddddddddar ($state$), caar ($state$)))
$\wedge$ (cadddddddar ($state$) $=$ **t**)
$\wedge$ boolp (caddddddddar ($state$))
$\wedge$ (cadddddddddar ($state$) $=$ **t**)
$\wedge$ bvp (caddddddddddar ($state$))
$\wedge$ (length (caddddddddddar ($state$)) $=$ 4)
$\wedge$ (caddddddddddar ($state$) $=$ pc)
$\wedge$ (cadddddddddddar ($state$)
$=$ cv_fetch1 (cadddddadr ($state$),
caddddddddddar ($state$),
caddddar ($state$),
cadar ($state$),
caddddddddddar ($state$)))
$\wedge$ memory-okp (32, 32, caadr ($state$))
$\wedge$ (cadadr ($state$) $=$ 0)
$\wedge$ (cadddadr ($state$) $\in$ **N**)

$\wedge$ boolp (caddddddadr ($state$))
$\wedge$ bvp (cadddddddadr ($state$))
$\wedge$ (length (cadddddddadr ($state$)) $=$ 32)
$\wedge$ properp (caddddddddadr ($state$))
$\wedge$ (length (caddddddddadr ($state$)) $=$ 32))

THEOREM: macrocycle-invariant==>chip-system-invariant$help
macrocycle-invariant (list (list (list ($regs\text{-}regs$, $regs\text{-}we$, $regs\text{-}data$, $regs\text{-}address$),
$flags$,
$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,
$last\text{-}reset\text{-}$,
$last\text{-}dtack\text{-}$,
$last\text{-}hold\text{-}$,
$pc\text{-}reg$,
$cntl\text{-}state$),
list (mem,
$cntl$,
$clock$,
$count$,
$dtack$,
$last\text{-}rw\text{-}$,
$last\text{-}address$,
$last\text{-}data$)),
pc)
$\rightarrow$ chip-system-invariant (list (list (list (list ($regs\text{-}regs$,
$regs\text{-}we$,
$regs\text{-}data$,
$regs\text{-}address$),
$flags$,
$a\text{-}reg$,
$b\text{-}reg$,
$i\text{-}reg$,
$data\text{-}out$,
$addr\text{-}out$,
$last\text{-}reset\text{-}$,
$last\text{-}dtack\text{-}$,
$last\text{-}hold\text{-}$,
$pc\text{-}reg$,
$cntl\text{-}state$),
list (mem,

cntl,
clock,
count,
dtack,
last-rw-,
last-address,
last-data)))

THEOREM: macrocycle-invariant==>chip-system-invariant
(fm9001-state-structure (*state*) $\land$ macrocycle-invariant (*state*, *pc*))
$\rightarrow$ chip-system-invariant (*state*)

DEFINITION:
macrocycle-invariant* (*state*, *pc*) = macrocycle-invariant (*state*, *pc*)

EVENT: Enable macrocycle-invariant*; name this event 'macrocycle-invariant*-off'.

THEOREM: macrocycle-invariant*=macrocycle-invariant
macrocycle-invariant* (cons (x, y), *pc*) = macrocycle-invariant (cons (x, y), *pc*)

THEOREM: macrocycle-invariant-is-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)
∧ 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)))))
→ 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*)
$\land$ macrocycle-invariant (*state*, *pc*)
$\land$ 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
map-up (cons (x, y))
= list (list (car (regs (car (cons (x, y)))), flags (car (cons (x, y)))),
caadr (cons (x, y)))

DEFINITION:
fm9001-step* (*state*, *pc-reg*) = fm9001-step (*state*, *pc-reg*)

EVENT: Enable fm9001-step*; name this event ‘fm9001-step*-off’.

THEOREM: fm9001-step*-lemma
(cons (x, y) = fm9001-step* (*state*, *pc-reg*))
= (cons (x, y) = fm9001-step (*state*, *pc-reg*))

THEOREM: middle=high$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-reg)
∧ 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)))))
→   (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)),
```

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)))))
= fm9001-step* (list (list (*regs-regs*, *flags*), *mem*), *pc-reg*))

THEOREM: macrocycle-invariant==>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*)
 ∧ fm9001-state-structure (*state*)
 ∧ macrocycle-invariant (*state*, *pc*)
 ∧ chip-system-operating-inputs-p (*inputs*,
 total-microcycles (*state*,
 map-up-inputs (*inputs*),
 instructions))
 ∧ operating-inputs-p (map-up-inputs (*inputs*),
 total-microcycles (*state*,
 map-up-inputs (*inputs*),
 instructions)))
→ (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*))))
 ∧ (flags (car (*high-level-state*)) = flags (car (*low-level-state*)))
 ∧ (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*)
 ∧ properp (car (*state*))
 ∧ (length (car (*state*)) = `2`)
 ∧ (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*) → (map-up (map-down (*state*)) = *state*)

Theorem: fm9001-statep-implies-fm9001-state-structure
fm9001-statep (*state*) → 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*)
→ 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*)
∧ fm9001-statep (*state*)
∧ chip-system-operating-inputs-p (*inputs*,
total-microcycles (map-down (*state*),
map-up-inputs (*inputs*),
n))
∧ operating-inputs-p (map-up-inputs (*inputs*),
total-microcycles (map-down (*state*),
map-up-inputs (*inputs*),
n)))
→ (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)
∧ operating-inputs-p (map-up-inputs (i), c))

THEOREM: fm9001=chip-system-summary
(fm9001-statep (s)
∧ no-holds-reset-or-test (i,
total-microcycles (map-down (s),
map-up-inputs (i),
n)))
→ (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 = **t**) ∨ (x = **f**) ∨ (x = z))

THEOREM: b-knownp-compound-recognizer
b-knownp (x) = (truep (x) ∨ falsep (x) ∨ 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*)
= **if** b-knownp (*a1*) **then** *a1* = *a2*
else t endif

DEFINITION:
v-approx (*v1*, *v2*)
= **if** listp (*v1*)
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
$((x \neq y) \wedge$ v-knownp $(x)) \rightarrow (\neg$ 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)) $\land$ good-s (cdr (s))
elseif ramp (s)
then (length (ram-guts (s)) = MEM-WIDTH) $\land$ properp (ram-guts (s))
elseif romp (s)
then (length (rom-guts (s)) = MEM-WIDTH) $\land$ properp (rom-guts (s))
elseif stubp (s)
then (length (stub-guts (s)) = MEM-WIDTH) $\land$ properp (stub-guts (s))
else t endif

DEFINITION:
s-knownp (s)
= **if** listp (s) **then** s-knownp (car (s)) $\land$ 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) $\land$ ($x \neq y$)) $\rightarrow$ ($\neg$ 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*)
$\land$ (caar (*alist1*) = caar (*alist2*))
$\land$ b-approx (cdar (*alist1*), cdar (*alist2*))
$\land$ v-approx-alist (cdr (*alist1*), cdr (*alist2*))
else *alist2* $\simeq$ **nil endif**

DEFINITION:
s-approx-alist (*alist1*, *alist2*)
= **if** listp (*alist1*)
then listp (*alist2*)
$\land$ (caar (*alist1*) = caar (*alist2*))
$\land$ s-approx (cdar (*alist1*), cdar (*alist2*))
$\land$ s-approx-alist (cdr (*alist1*), cdr (*alist2*))
else *alist2* $\simeq$ **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 ($a2$)

$\wedge$ s-approx-alist (*s1*, *s2*)
$\wedge$ alistp (*s1*)
$\wedge$ alistp (*s2*))
$\rightarrow$ v-approx-alist (dual-eval (1, *fn*, *a1*, *s1*, *netlist*),
dual-eval (1, *fn*, *a2*, *s2*, *netlist*))

THEOREM: monotonicity-property-consequence-2
(monotonicity-property (2, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\wedge$ v-approx (*a1*, *a2*)
$\wedge$ s-approx (*s1*, *s2*))
$\rightarrow$ s-approx (dual-eval (2, *fn*, *a1*, *s1*, *netlist*), dual-eval (2, *fn*, *a2*, *s2*, *netlist*))

THEOREM: monotonicity-property-consequence-3
(monotonicity-property (3, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\wedge$ v-approx-alist (*a1*, *a2*)
$\wedge$ alistp (*a1*)
$\wedge$ alistp (*a2*)
$\wedge$ s-approx-alist (*s1*, *s2*)
$\wedge$ alistp (*s1*)
$\wedge$ alistp (*s2*))
$\rightarrow$ s-approx-alist (dual-eval (3, *fn*, *a1*, *s1*, *netlist*),
dual-eval (3, *fn*, *a2*, *s2*, *netlist*))

THEOREM: monotonicity-property-opener-0
monotonicity-property (0, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\leftrightarrow$ ((v-approx (*a1*, *a2*) $\wedge$ s-approx (*s1*, *s2*))
$\rightarrow$ v-approx (dual-eval (0, *fn*, *a1*, *s1*, *netlist*),
dual-eval (0, *fn*, *a2*, *s2*, *netlist*)))

THEOREM: monotonicity-property-opener-1
monotonicity-property (1, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\leftrightarrow$ ((v-approx-alist (*a1*, *a2*)
$\wedge$ alistp (*a1*)
$\wedge$ alistp (*a2*)
$\wedge$ s-approx-alist (*s1*, *s2*)
$\wedge$ alistp (*s1*)
$\wedge$ alistp (*s2*))
$\rightarrow$ v-approx-alist (dual-eval (1, *fn*, *a1*, *s1*, *netlist*),
dual-eval (1, *fn*, *a2*, *s2*, *netlist*)))

THEOREM: monotonicity-property-opener-2
monotonicity-property (2, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\leftrightarrow$ ((v-approx (*a1*, *a2*) $\wedge$ s-approx (*s1*, *s2*))
$\rightarrow$ s-approx (dual-eval (2, *fn*, *a1*, *s1*, *netlist*),
dual-eval (2, *fn*, *a2*, *s2*, *netlist*)))

THEOREM: monotonicity-property-opener-3
monotonicity-property (3, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\leftrightarrow$ ((v-approx-alist (*a1*, *a2*)
$\wedge$ alistp (*a1*)
$\wedge$ alistp (*a2*)
$\wedge$ s-approx-alist (*s1*, *s2*)
$\wedge$ alistp (*s1*)
$\wedge$ alistp (*s2*))
$\rightarrow$ s-approx-alist (dual-eval (3, *fn*, *a1*, *s1*, *netlist*),
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
b-approx (*a1*, *a2*) $\rightarrow$ b-approx (f-buf (*a1*), f-buf (*a2*))

THEOREM: f-and-monotone
(b-approx (*a1*, *a2*) $\wedge$ b-approx (*b1*, *b2*))
$\rightarrow$ b-approx (f-and (*a1*, *b1*), f-and (*a2*, *b2*))

THEOREM: f-and3-monotone
(b-approx (*a1*, *a2*) $\wedge$ b-approx (*b1*, *b2*) $\wedge$ b-approx (*c1*, *c2*))
$\rightarrow$ b-approx (f-and3 (*a1*, *b1*, *c1*), f-and3 (*a2*, *b2*, *c2*))

THEOREM: f-and4-monotone
(b-approx (*a1*, *a2*) $\wedge$ b-approx (*b1*, *b2*) $\wedge$ b-approx (*c1*, *c2*) $\wedge$ b-approx (*d1*, *d2*))
$\rightarrow$ b-approx (f-and4 (*a1*, *b1*, *c1*, *d1*), f-and4 (*a2*, *b2*, *c2*, *d2*))

THEOREM: f-not-monotone
b-approx (*a1*, *a2*) $\rightarrow$ b-approx (f-not (*a1*), f-not (*a2*))

THEOREM: f-nand-monotone
(b-approx (*a1*, *a2*) $\wedge$ b-approx (*b1*, *b2*))
$\rightarrow$ b-approx (f-nand (*a1*, *b1*), f-nand (*a2*, *b2*))

Theorem: f-nand3-monotone
(b-approx $(a1,\ a2)$ ∧ b-approx $(b1,\ b2)$ ∧ b-approx $(c1,\ c2)$)
→ b-approx (f-nand3 $(a1,\ b1,\ c1)$, f-nand3 $(a2,\ b2,\ c2)$)

Theorem: f-nand4-monotone
(b-approx $(a1,\ a2)$ ∧ b-approx $(b1,\ b2)$ ∧ b-approx $(c1,\ c2)$ ∧ b-approx $(d1,\ d2)$)
→ b-approx (f-nand4 $(a1,\ b1,\ c1,\ d1)$, f-nand4 $(a2,\ b2,\ c2,\ d2)$)

Theorem: f-nand5-monotone
(b-approx $(a1,\ a2)$
∧ b-approx $(b1,\ b2)$
∧ b-approx $(c1,\ c2)$
∧ b-approx $(d1,\ d2)$
∧ b-approx $(e1,\ e2)$)
→ b-approx (f-nand5 $(a1,\ b1,\ c1,\ d1,\ e1)$, f-nand5 $(a2,\ b2,\ c2,\ d2,\ e2)$)

Theorem: f-nand6-monotone
(b-approx $(a1,\ a2)$
∧ b-approx $(b1,\ b2)$
∧ b-approx $(c1,\ c2)$
∧ b-approx $(d1,\ d2)$
∧ b-approx $(e1,\ e2)$
∧ b-approx $(g1,\ g2)$)
→ b-approx (f-nand6 $(a1,\ b1,\ c1,\ d1,\ e1,\ g1)$, f-nand6 $(a2,\ b2,\ c2,\ d2,\ e2,\ g2)$)

Theorem: f-nand8-monotone
(b-approx $(a1,\ a2)$
∧ b-approx $(b1,\ b2)$
∧ b-approx $(c1,\ c2)$
∧ b-approx $(d1,\ d2)$
∧ b-approx $(e1,\ e2)$
∧ b-approx $(g1,\ g2)$
∧ b-approx $(h1,\ h2)$
∧ b-approx $(i1,\ i2)$)
→ b-approx (f-nand8 $(a1,\ b1,\ c1,\ d1,\ e1,\ g1,\ h1,\ i1)$,
f-nand8 $(a2,\ b2,\ c2,\ d2,\ e2,\ g2,\ h2,\ i2)$)

Theorem: f-or-monotone
(b-approx $(a1,\ a2)$ ∧ b-approx $(b1,\ b2)$) → b-approx (f-or $(a1,\ b1)$, f-or $(a2,\ b2)$)

Theorem: f-or3-monotone
(b-approx $(a1,\ a2)$ ∧ b-approx $(b1,\ b2)$ ∧ b-approx $(c1,\ c2)$)
→ b-approx (f-or3 $(a1,\ b1,\ c1)$, f-or3 $(a2,\ b2,\ c2)$)

Theorem: f-or4-monotone
(b-approx $(a1,\ a2)$ ∧ b-approx $(b1,\ b2)$ ∧ b-approx $(c1,\ c2)$ ∧ b-approx $(d1,\ d2)$)
→ b-approx (f-or4 $(a1,\ b1,\ c1,\ d1)$, f-or4 $(a2,\ b2,\ c2,\ d2)$)

THEOREM: f-nor-monotone
(b-approx $(a1, a2)$ ∧ b-approx $(b1, b2)$)
→ b-approx (f-nor $(a1, b1)$, f-nor $(a2, b2)$)

THEOREM: f-nor3-monotone
(b-approx $(a1, a2)$ ∧ b-approx $(b1, b2)$ ∧ b-approx $(c1, c2)$)
→ b-approx (f-nor3 $(a1, b1, c1)$, f-nor3 $(a2, b2, c2)$)

THEOREM: f-nor4-monotone
(b-approx $(a1, a2)$ ∧ b-approx $(b1, b2)$ ∧ b-approx $(c1, c2)$ ∧ b-approx $(d1, d2)$)
→ b-approx (f-nor4 $(a1, b1, c1, d1)$, f-nor4 $(a2, b2, c2, d2)$)

THEOREM: f-nor5-monotone
(b-approx $(a1, a2)$
∧ b-approx $(b1, b2)$
∧ b-approx $(c1, c2)$
∧ b-approx $(d1, d2)$
∧ b-approx $(e1, e2)$)
→ b-approx (f-nor5 $(a1, b1, c1, d1, e1)$, f-nor5 $(a2, b2, c2, d2, e2)$)

THEOREM: f-nor6-monotone
(b-approx $(a1, a2)$
∧ b-approx $(b1, b2)$
∧ b-approx $(c1, c2)$
∧ b-approx $(d1, d2)$
∧ b-approx $(e1, e2)$
∧ b-approx $(g1, g2)$)
→ b-approx (f-nor6 $(a1, b1, c1, d1, e1, g1)$, f-nor6 $(a2, b2, c2, d2, e2, g2)$)

THEOREM: f-nor8-monotone
(b-approx $(a1, a2)$
∧ b-approx $(b1, b2)$
∧ b-approx $(c1, c2)$
∧ b-approx $(d1, d2)$
∧ b-approx $(e1, e2)$
∧ b-approx $(g1, g2)$
∧ b-approx $(h1, h2)$
∧ b-approx $(i1, i2)$)
→ b-approx (f-nor8 $(a1, b1, c1, d1, e1, g1, h1, i1)$,
f-nor8 $(a2, b2, c2, d2, e2, g2, h2, i2)$)

THEOREM: f-xor-monotone
(b-approx $(a1, a2)$ ∧ b-approx $(b1, b2)$)
→ b-approx (f-xor $(a1, b1)$, f-xor $(a2, b2)$)

THEOREM: f-equv-monotone
(b-approx ($a1$, $a2$) $\land$ b-approx ($b1$, $b2$))
$\rightarrow$ b-approx (f-equv ($a1$, $b1$), f-equv ($a2$, $b2$))

THEOREM: f-xor3-monotone
(b-approx ($a1$, $a2$) $\land$ b-approx ($b1$, $b2$) $\land$ b-approx ($c1$, $c2$))
$\rightarrow$ b-approx (f-xor3 ($a1$, $b1$, $c1$), f-xor3 ($a2$, $b2$, $c2$))

THEOREM: f-equv3-monotone
(b-approx ($a1$, $a2$) $\land$ b-approx ($b1$, $b2$) $\land$ b-approx ($c1$, $c2$))
$\rightarrow$ b-approx (f-equv3 ($a1$, $b1$, $c1$), f-equv3 ($a2$, $b2$, $c2$))

THEOREM: f-if-monotone
(b-approx ($c1$, $c2$) $\land$ b-approx ($a1$, $a2$) $\land$ b-approx ($b1$, $b2$))
$\rightarrow$ b-approx (f-if ($c1$, $a1$, $b1$), f-if ($c2$, $a2$, $b2$))

THEOREM: ft-buf-monotone
(b-approx ($c1$, $c2$) $\land$ b-approx ($a1$, $a2$))
$\rightarrow$ b-approx (ft-buf ($c1$, $a1$), ft-buf ($c2$, $a2$))

THEOREM: ft-wire-monotone
(b-approx ($a1$, $a2$) $\land$ b-approx ($b1$, $b2$))
$\rightarrow$ b-approx (ft-wire ($a1$, $b1$), ft-wire ($a2$, $b2$))

THEOREM: f-pullup-monotone
b-approx ($a1$, $a2$) $\rightarrow$ b-approx (f-pullup ($a1$), f-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-equv; name this event ‘f-equv-off’.

EVENT: Enable f-equv3; name this event ‘f-equv3-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-equv-monotone, f-equv3-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*)
$\rightarrow$ (dual-eval (0, *name*, *args*, *state*, *netlist*)
= dual-apply-value (*name*, *args*, *state*))

THEOREM: dual-eval-2-primp
primp (*name*)
$\rightarrow$ (dual-eval (2, *name*, *args*, *state*, *netlist*)
= dual-apply-state (*name*, *args*, *state*))

THEOREM: s-approx-implies-b-approx
s-approx (x, y) $\rightarrow$ b-approx (x, y)

THEOREM: f-buf-type-set
truep (f-buf (x)) $\vee$ falsep (f-buf (x)) $\vee$ 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) $\wedge$ fourp (y) $\wedge$ b-approx (x, y)) $\rightarrow$ (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*), cadddr (*args*))))

THEOREM: dual-eval-ao2-state
dual-eval (2, 'ao2, *args*, *state*, *netlist*) = 0

THEOREM: ao2-monotone
monotonicity-property (0, 'ao2, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'ao2, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ao4-value
dual-eval (0, 'ao4, *args*, *state*, *netlist*)
= list (f-nand (f-or (car (*args*), cadr (*args*)), f-or (caddr (*args*), cadddr (*args*))))

THEOREM: dual-eval-ao4-state
dual-eval (2, 'ao4, *args*, *state*, *netlist*) = 0

THEOREM: ao4-monotone
monotonicity-property (0, 'ao4, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'ao4, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ao6-value
dual-eval (0, 'ao6, *args*, *state*, *netlist*)
= list (f-nor (f-and (car (*args*), cadr (*args*)), caddr (*args*)))

THEOREM: dual-eval-ao6-state
dual-eval (2, 'ao6, *args*, *state*, *netlist*) = 0

THEOREM: ao6-monotone
monotonicity-property (0, 'ao6, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'ao6, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ao7-value
dual-eval (0, 'ao7, *args*, *state*, *netlist*)
= list (f-nand (f-or (car (*args*), cadr (*args*)), caddr (*args*)))

THEOREM: dual-eval-ao7-state
dual-eval (2, 'ao7, *args*, *state*, *netlist*) = 0

THEOREM: ao7-monotone
monotonicity-property (0, 'ao7, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'ao7, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-and-value
dual-eval (0, 'b-and, *args*, *state*, *netlist*)
= list (f-and (car (*args*), cadr (*args*)))

THEOREM: dual-eval-b-and-state
dual-eval (2, 'b-and, *args*, *state*, *netlist*) = 0

THEOREM: b-and-monotone
monotonicity-property (0, 'b-and, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ 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*)
∧ 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*), cadddr (*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*)
∧ monotonicity-property (2, 'b-and4, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-equv-value
dual-eval (0, 'b-equv, *args*, *state*, *netlist*)
= list (f-equv (car (*args*), cadr (*args*)))

THEOREM: dual-eval-b-equv-state
dual-eval (2, 'b-equv, *args*, *state*, *netlist*) = 0

THEOREM: b-equv-monotone
monotonicity-property (0, 'b-equv, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-equv, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-equv3-value
dual-eval (0, 'b-equv3, *args*, *state*, *netlist*)
= list (f-equv3 (car (*args*), cadr (*args*), caddr (*args*)))

THEOREM: dual-eval-b-equv3-state
dual-eval (2, 'b-equv3, *args*, *state*, *netlist*) = 0

THEOREM: b-equv3-monotone
monotonicity-property (0, 'b-equv3, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-equv3, *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*), cadddr (*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*),
cadddr (*args*),
caddddr (*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*)
∧ 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*),
cadddr (*args*),
caddddr (*args*),
cadddddr (*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*)
∧ 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*),
cadddr (*args*),
caddddr (*args*),
cadddddr (*args*),
caddddddr (*args*),
cadddddddr (*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*)
∧ 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*)
∧ 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*)
∧ 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*)
∧ 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*), cadddr (*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*)
∧ 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*)
∧ 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*)
∧ 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*),
cadddddddr (*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*)
∧ 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*)
∧ 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*)
∧ 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*)
∧ 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*)
∧ 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
dual-eval (2, 'b-or3, *args*, *state*, *netlist*) = 0

THEOREM: b-or3-monotone
monotonicity-property (0, 'b-or3, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-or3, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-or4-value
dual-eval (0, 'b-or4, *args*, *state*, *netlist*)
= list (f-or4 (car (*args*), cadr (*args*), caddr (*args*), cadddr (*args*)))

THEOREM: dual-eval-b-or4-state
dual-eval (2, 'b-or4, *args*, *state*, *netlist*) = 0

THEOREM: b-or4-monotone
monotonicity-property (0, 'b-or4, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-or4, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-xor-value
dual-eval (0, 'b-xor, *args*, *state*, *netlist*)
= list (f-xor (car (*args*), cadr (*args*)))

THEOREM: dual-eval-b-xor-state
dual-eval (2, 'b-xor, *args*, *state*, *netlist*) = 0

THEOREM: b-xor-monotone
monotonicity-property (0, 'b-xor, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-xor, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-b-xor3-value
dual-eval (0, 'b-xor3, *args*, *state*, *netlist*)
= list (f-xor3 (car (*args*), cadr (*args*), caddr (*args*)))

THEOREM: dual-eval-b-xor3-state
dual-eval (2, 'b-xor3, *args*, *state*, *netlist*) = 0

THEOREM: b-xor3-monotone
monotonicity-property (0, 'b-xor3, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'b-xor3, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-del2-value
dual-eval (0, 'del2, *args*, *state*, *netlist*) = list (f-buf (car (*args*)))

THEOREM: dual-eval-del2-state
dual-eval (2, 'del2, *args*, *state*, *netlist*) = 0

THEOREM: del2-monotone
monotonicity-property (0, 'del2, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ 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*)
∧ 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*)
∧ 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*)),
cadddr (*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*)
∧ 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*)
∧ monotonicity-property (2, 'fd1, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-fd1s-value
dual-eval (0, 'fd1s, *args*, *state*, *netlist*) = list (f-buf (*state*), f-not (*state*))

THEOREM: dual-eval-fd1s-state
dual-eval (2, 'fd1s, *args*, *state*, *netlist*)
= f-if (cadddr (*args*), caddr (*args*), car (*args*))

THEOREM: fd1s-monotone
monotonicity-property (0, 'fd1s, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'fd1s, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-fd1sp-value
dual-eval (0, 'fd1sp, *args*, *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*))

THEOREM: dual-eval-fd1sp-state
dual-eval (2, 'fd1sp, *args*, *state*, *netlist*)
= f-if (cadddr (*args*), caddr (*args*), car (*args*))

THEOREM: fd1sp-monotone
monotonicity-property (0, 'fd1sp, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'fd1sp, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-fd1slp-value
dual-eval (0, 'fd1slp, *args*, *state*, *netlist*)
= list (f-buf (*state*), f-not (*state*))

THEOREM: dual-eval-fd1slp-state
dual-eval (2, 'fd1slp, *args*, *state*, *netlist*)
= f-if (caddddr (*args*), cadddr (*args*), f-if (caddr (*args*), car (*args*), *state*))

THEOREM: fd1slp-monotone
monotonicity-property (0, 'fd1slp, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'fd1slp, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-id-value
dual-eval (0, 'id, *args*, *state*, *netlist*) = list (car (*args*))

THEOREM: dual-eval-id-state
dual-eval (2, 'id, *args*, *state*, *netlist*) = 0

THEOREM: id-monotone
monotonicity-property (0, 'id, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'id, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ram-enable-circuit-value
dual-eval (0, 'ram-enable-circuit, *args*, *state*, *netlist*)
= list (f-nand (caddr (*args*), cadddr (*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*)
$\land$ 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*)
$\land$ 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*)
$\land$ 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*)
$\land$ 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*))),
cadddr (*args*)))

THEOREM: dual-eval-ttl-bidirect-state
dual-eval (2, `'ttl-bidirect`, *args*, *state*, *netlist*) = `0`

THEOREM: ttl-bidirect-monotone
monotonicity-property (`0`, `'ttl-bidirect`, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (`2`, `'ttl-bidirect`, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-clk-input-value
dual-eval (`0`, `'ttl-clk-input`, *args*, *state*, *netlist*)
= list (f-buf (car (*args*)), f-nand (car (*args*), cadr (*args*)))

THEOREM: dual-eval-ttl-clk-input-state
dual-eval (`2`, `'ttl-clk-input`, *args*, *state*, *netlist*) = `0`

THEOREM: ttl-clk-input-monotone
monotonicity-property (`0`, `'ttl-clk-input`, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (`2`, `'ttl-clk-input`, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-input-value
dual-eval (`0`, `'ttl-input`, *args*, *state*, *netlist*)
= list (f-buf (car (*args*)), f-nand (car (*args*), cadr (*args*)))

THEOREM: dual-eval-ttl-input-state
dual-eval (`2`, `'ttl-input`, *args*, *state*, *netlist*) = `0`

THEOREM: ttl-input-monotone
monotonicity-property (`0`, `'ttl-input`, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (`2`, `'ttl-input`, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-output-value
dual-eval (`0`, `'ttl-output`, *args*, *state*, *netlist*) = list (f-buf (car (*args*)))

THEOREM: dual-eval-ttl-output-state
dual-eval (`2`, `'ttl-output`, *args*, *state*, *netlist*) = `0`

THEOREM: ttl-output-monotone
monotonicity-property (`0`, `'ttl-output`, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (`2`, `'ttl-output`, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-ttl-output-parametric-value
dual-eval (`0`, `'ttl-output-parametric`, *args*, *state*, *netlist*)
= list (f-buf (car (*args*)))

THEOREM: dual-eval-ttl-output-parametric-state
dual-eval (`2`, `'ttl-output-parametric`, *args*, *state*, *netlist*) = `0`

THEOREM: ttl-output-parametric-monotone
monotonicity-property (0, **'ttl-output-parametric**, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ 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*)
∧ 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*)
∧ 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*)
∧ monotonicity-property (2,
'ttl-tri-output-fast,
netlist,
a1,
a2,
s1,
s2)

THEOREM: dual-eval-vdd-value
dual-eval (0, 'vdd, *args*, *state*, *netlist*) = list (t)

THEOREM: dual-eval-vdd-state
dual-eval (2, 'vdd, *args*, *state*, *netlist*) = 0

THEOREM: vdd-monotone
monotonicity-property (0, 'vdd, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'vdd, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-vdd-parametric-value
dual-eval (0, 'vdd-parametric, *args*, *state*, *netlist*) = list (t)

THEOREM: dual-eval-vdd-parametric-state
dual-eval (2, 'vdd-parametric, *args*, *state*, *netlist*) = 0

THEOREM: vdd-parametric-monotone
monotonicity-property (0, 'vdd-parametric, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'vdd-parametric, *netlist*, *a1*, *a2*, *s1*, *s2*)

THEOREM: dual-eval-vss-value
dual-eval (0, 'vss, *args*, *state*, *netlist*) = list (f)

THEOREM: dual-eval-vss-state
dual-eval (2, 'vss, *args*, *state*, *netlist*) = 0

THEOREM: vss-monotone
monotonicity-property (0, 'vss, *netlist*, *a1*, *a2*, *s1*, *s2*)
∧ monotonicity-property (2, 'vss, *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-equv-monotone, b-equv3-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
$(\text{bvp}(x) \land \text{bvp}(acc)) \rightarrow \text{bvp}(\text{rev1}(x, acc))$

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:
read-mem1-monotone-induction ($v\text{-}addr$, $mem1$, $mem2$)
= **if** stubp ($mem1$)
  ∨ stubp ($mem2$)
  ∨ ($v\text{-}addr \simeq$ **nil**)
  ∨ ($mem1 \simeq$ **nil**)
  ∨ ($mem2 \simeq$ **nil**) **then t**
  elseif car ($v\text{-}addr$)
  then read-mem1-monotone-induction (cdr ($v\text{-}addr$), cdr ($mem1$), cdr ($mem2$))
  else read-mem1-monotone-induction (cdr ($v\text{-}addr$),
  car ($mem1$),
  car ($mem2$)) **endif**

THEOREM: read-mem1-monotone
$(\text{bvp}(v\text{-}addr) \land \text{s-approx}(mem1, mem2))$
$\rightarrow \text{v-approx}(\text{read-mem1}(v\text{-}addr, mem1), \text{read-mem1}(v\text{-}addr, mem2))$

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
$((\text{nth}(n, x) = c) \land (\text{nth}(n, y) \neq c) \land \text{b-knownp}(c))$
$\rightarrow (\neg \text{v-approx}(x, y))$

EVENT: Disable *1*b-knownp; name this event 'g*1*b-knownp-on'.

THEOREM: v-approx-implies-subranges-equal
$(\text{v-approx}(x, y) \land \text{bvp}(\text{subrange}(x, i, j)))$
$\rightarrow (\text{subrange}(x, i, j) = \text{subrange}(y, i, j))$

THEOREM: v-approx-bvp-subrange
$(\text{v-approx}(a1, a2) \land \text{bvp}(\text{subrange}(a1, i, j))) \rightarrow \text{bvp}(\text{subrange}(a2, i, j))$

THEOREM: v-approx-make-list-x
((length (y) = *bits*) ∧ properp (y)) → v-approx (make-list (*bits*, X), y)

THEOREM: read-mem-monotone
(bvp (*v-addr*) ∧ s-approx (*mem1*, *mem2*))
→ v-approx (read-mem (*v-addr*, *mem1*), read-mem (*v-addr*, *mem2*))

THEOREM: equal-length-read-mem1
s-approx (*s1*, *s2*)
→ ((length (read-mem1 (a, *s2*)) = length (read-mem1 (a, *s1*))) = **t**)

THEOREM: equal-length-read-mem
s-approx (*s1*, *s2*)
→ ((length (read-mem (a, *s2*)) = length (read-mem (a, *s1*))) = **t**)

THEOREM: s-approx-implies-properp-read-mem1
(s-approx (*s1*, *s2*) ∧ properp (read-mem1 (a, *s1*))) → properp (read-mem1 (a, *s2*))

THEOREM: s-approx-implies-properp-read-mem
(s-approx (*s1*, *s2*) ∧ properp (read-mem (a, *s1*))) → properp (read-mem (a, *s2*))

THEOREM: dual-port-ram-value-monotone
((*address-lines* ∈ **N**) ∧ s-approx (*s1*, *s2*) ∧ v-approx (*a1*, *a2*))
→ v-approx (dual-port-ram-value (*bits*, *address-lines*, *a1*, *s1*),
dual-port-ram-value (*bits*, *address-lines*, *a2*, *s2*))

THEOREM: eval$-append
(litatom (x) ∧ (x ∈ *lst*))
→ (eval$ (**t**, x, append (pairlist (*lst*, y), z)) = eval$ (**t**, x, pairlist (*lst*, y)))

THEOREM: eval$-pairlist-cons
litatom (x)
→ (eval$ (**t**, x, pairlist (cons (c, *lst*), y))
= **if** $x = c$ **then** car (y)
else eval$ (**t**, x, pairlist (*lst*, cdr (y))) **endif**)

THEOREM: eval$-append-2
(litatom (x) ∧ (x ∉ *lst*))
→ (eval$ (**t**, x, append (pairlist (*lst*, y), z)) = eval$ (**t**, x, z))

THEOREM: dual-apply-value-dp-ram-16x32-lemma-1
dual-apply-value (`'dp-ram-16x32`, a, s)
= eval$ (**t**,
cdr (lookup-module (`'results`, cdr (primp (`'dp-ram-16x32`)))),
append (pairlist (cdr (lookup-module (`'inputs`,
cdr (primp (`'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)
=  dual-port-ram-value (32,
                        4,
                        list (car (a),
                              cadr (a),
                              caddr (a),
                              cadddr (a),
                              caddddr (a),
                              cadddddr (a),
                              caddddddr (a),
                              cadddddddr (a),
                              caddddddddr (a),
                              cadddddddddr (a),
                              caddddddddddr (a),
                              cadddddddddddr (a),
                              caddddddddddddr (a),
                              cadddddddddddddr (a),
                              caddddddddddddddr (a),
                              cadddddddddddddddr (a),
                              caddddddddddddddddr (a),
                              cadddddddddddddddddr (a),
                              caddddddddddddddddddr (a),
```

caddddddddddddddddddddr (a),
cadddddddddddddddddddddr (a),
caddddddddddddddddddddddr (a),
cadddddddddddddddddddddddr (a),
caddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddddddddddr (a),
caddddddddddddddddddddddddddddddddddddddddr (a),
cadddddddddddddddddddddddddddddddddddddddddr (a)),
s)

DEFINITION:
dual-port-ram-value-body ($bits$, a-$address$, b-$address$, wen, $state$)
= **if** ($\neg$ bvp (a-$address$))
  $\vee$ (($wen \neq$ **t**)
   $\wedge$ (($\neg$ bvp (b-$address$)) $\vee$ (a-$address$ = b-$address$)))
 then make-list ($bits$, X)
 elseif properp (read-mem (a-$address$, $state$))
   $\wedge$ (length (read-mem (a-$address$, $state$)) = $bits$)
 then read-mem (a-$address$, $state$)
 else make-list ($bits$, X) **endif**

THEOREM: dual-port-ram-value-is-dual-port-ram-value-body
dual-port-ram-value ($bits$, $address$-$lines$, $args$, $state$)
= dual-port-ram-value-body ($bits$,
  subrange ($args$, 0, $address$-$lines$ $-$ 1),
  subrange ($args$,
   $address$-$lines$,
   (2 $*$ $address$-$lines$) $-$ 1),
  nth (2 $*$ $address$-$lines$, $args$),
  $state$)

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
((listp (*s1*) ∨ listp (*s2*))
 → (s-approx (*s1*, *s2*)
   = **if** listp (*s1*)
     then if listp (*s2*)
       then s-approx (car (*s1*), car (*s2*))
         ∧ s-approx (cdr (*s1*), cdr (*s2*))
       else f endif
     else f endif))
∧ (((*s1* = **nil**) ∨ (*s2* = **nil**))
  → (s-approx (*s1*, *s2*) = (*s1* = *s2*)))
∧ ((ramp (*s1*) ∨ ramp (*s2*))
  → (s-approx (*s1*, *s2*)
    = **if** ramp (*s1*)
      then if ramp (*s2*)
        then v-approx (ram-guts (*s1*), ram-guts (*s2*))
        else f endif
      else f endif))
∧ ((romp (*s1*) ∨ romp (*s2*))
  → (s-approx (*s1*, *s2*)
    = **if** romp (*s1*)

then if romp (*s2*)
  then v-approx (rom-guts (*s1*), rom-guts (*s2*))
  else f endif
else f endif))

$\land$ ((stubp (*s1*) $\lor$ stubp (*s2*))
 $\rightarrow$ (s-approx (*s1*, *s2*)
  = **if** stubp (*s1*)
  then if stubp (*s2*)
   then v-approx (stub-guts (*s1*), stub-guts (*s2*))
   else f endif
  else f endif))

$\land$ (($\neg$ (listp (*s1*)
 $\lor$ listp (*s2*)
 $\lor$ (*s1* = **nil**)
 $\lor$ (*s2* = **nil**)
 $\lor$ ramp (*s1*)
 $\lor$ ramp (*s2*)
 $\lor$ romp (*s1*)
 $\lor$ romp (*s2*)
 $\lor$ stubp (*s1*)
 $\lor$ stubp (*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
(($\neg$ nth (*n*, *a1*)) $\land$ nth (*n*, *a2*)) $\rightarrow$ ($\neg$ v-approx (*a1*, *a2*))

THEOREM: write-mem1-opener
(stubp (*mem*) $\rightarrow$ (write-mem1 (*v-addr*, *mem*, *value*) = *mem*))
$\land$ ((*v-addr* $\simeq$ **nil**)
 $\rightarrow$ (write-mem1 (*v-addr*, *mem*, *value*)
  = **if** ramp (*mem*) **then** ram (*value*)
  else *mem* **endif**))
$\land$ ((listp (*v-addr*) $\land$ (*mem* $\simeq$ **nil**))
 $\rightarrow$ (write-mem1 (*v-addr*, *mem*, *value*) = *mem*))
$\land$ ((listp (*mem*) $\land$ listp (*v-addr*) $\land$ car (*v-addr*))
 $\rightarrow$ (write-mem1 (*v-addr*, *mem*, *value*)
  = cons (car (*mem*), write-mem1 (cdr (*v-addr*), cdr (*mem*), *value*))))

$\wedge$ ((listp (mem) $\wedge$ listp ($v\text{-}addr$) $\wedge$ ($\neg$ car ($v\text{-}addr$)))
$\rightarrow$ (write-mem1 ($v\text{-}addr$, mem, $value$)
$=$ cons (write-mem1 (cdr ($v\text{-}addr$), car (mem), $value$), cdr (mem))))

DEFINITION:
write-mem1-monotone-induction ($v\text{-}addr$, $mem1$, $mem2$)
$=$ **if** stubp ($mem1$) $\vee$ stubp ($mem2$) **then t**
elseif $v\text{-}addr \simeq$ **nil then t**
elseif ($mem1 \simeq$ **nil**) $\vee$ ($mem2 \simeq$ **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 ($v1$) $\not\lessdot$ length ($v2$))

DEFINITION:
write-mem1-double-induction ($v\text{-}addr$, $mem1$, $mem2$)
$=$ **if** stubp ($mem1$) $\vee$ stubp ($mem2$) **then t**
elseif $v\text{-}addr \simeq$ **nil then t**
elseif ($mem1 \simeq$ **nil**) $\vee$ ($mem2 \simeq$ **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
(s-approx ($s1$, $s2$) $\wedge$ good-s ($s2$))
$\rightarrow$ s-approx (write-mem1 ($v\text{-}addr$, $s1$, make-list (MEM-WIDTH, X)), $s2$)

THEOREM: s-approx-write-mem-id
(good-s ($s2$) $\wedge$ s-approx ($s1$, $s2$))
$\rightarrow$ s-approx (write-mem ($v\text{-}addr$, $s1$, make-list (MEM-WIDTH, X)), $s2$)

THEOREM: s-approx-constant-ram-x-id
(good-s ($s2$) $\wedge$ s-approx ($s1$, $s2$))
$\rightarrow$ s-approx (constant-ram ($s1$, make-list (MEM-WIDTH, X)), $s2$)

THEOREM: s-approx-constant-ram-x-constant-ram-x
s-approx ($s1$, $s2$)
$\rightarrow$ s-approx (constant-ram ($s1$, make-list (MEM-WIDTH, X)),
constant-ram ($s2$, make-list (MEM-WIDTH, X)))

THEOREM: v-approx-preserves-length
v-approx ($a1$, $a2$) $\rightarrow$ (length ($a1$) = length ($a2$))

THEOREM: v-approx-subrange
v-approx ($a1$, $a2$) $\rightarrow$ v-approx (subrange ($a1$, i, j), subrange ($a2$, i, j))

THEOREM: mem-width-non-zero
MEM-WIDTH $\neq$ 0

THEOREM: s-approx-constant-ram-x-write-mem1-case-3
(($\neg$ stubp ($s1$))
$\wedge$ ($\neg$ stubp ($s2$))
$\wedge$ listp ($v\text{-}addr$)
$\wedge$ ($\neg$ listp ($s1$))
$\wedge$ s-approx ($s1$, $s2$)
$\wedge$ properp (v)
$\wedge$ (length (v) = MEM-WIDTH)
$\wedge$ ($\neg$ ramp ($s1$)))
$\rightarrow$ s-approx ($s1$, write-mem1 ($v\text{-}addr$, $s2$, v))

THEOREM: s-approx-constant-ram-x-write-mem1
(s-approx ($s1$, $s2$) $\wedge$ properp (v) $\wedge$ (length (v) = MEM-WIDTH) $\wedge$ good-s ($s2$))
$\rightarrow$ s-approx (constant-ram ($s1$, make-list (MEM-WIDTH, X)),
write-mem1 ($v\text{-}addr$, $s2$, v))

THEOREM: s-approx-constant-ram-x-write-mem
(s-approx ($s1$, $s2$) $\wedge$ properp (v) $\wedge$ (length (v) = MEM-WIDTH) $\wedge$ good-s ($s2$))
$\rightarrow$ s-approx (constant-ram ($s1$, make-list (MEM-WIDTH, X)),
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
$(31 < \text{MEM-WIDTH}) \land (\text{MEM-WIDTH} < 33)$

THEOREM: dual-port-ram-state-monotone
(s-approx ($s1$, $s2$) ∧ good-s ($s2$) ∧ v-approx ($a1$, $a2$))
→ s-approx (dual-port-ram-state (MEM-WIDTH, 4, $a1$, $s1$),
 dual-port-ram-state (MEM-WIDTH, 4, $a2$, $s2$))

THEOREM: dual-port-ram-state-monotone-rewrite
(s-approx ($s1$, $s2$) ∧ good-s ($s2$) ∧ v-approx ($a1$, $a2$))
→ s-approx (dual-port-ram-state (32, 4, $a1$, $s1$),
 dual-port-ram-state (32, 4, $a2$, $s2$))

THEOREM: dual-apply-state-dp-ram-16x32-lemma-1
```
dual-apply-state ('dp-ram-16x32, a, s)
=   eval$ (t,
           cdr (lookup-module ('new-states, cdr (primp ('dp-ram-16x32)))),
           append (pairlist (cdr (lookup-module ('inputs,
                                                 cdr (primp ('dp-ram-16x32)))),
                             a),
                   pairstates (cdr (lookup-module ('states,
                                                   cdr (primp ('dp-ram-16x32)))),
                               s)))
```

THEOREM: dual-apply-state-dp-ram-16x32-lemma-2
```
dual-apply-state ('dp-ram-16x32, a, s)
=   dual-port-ram-state (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-state-dp-ram-16x32-lemma-3

```
dual-apply-state ('dp-ram-16x32, a, s)
```

= dual-port-ram-state(32,
4,
list(car(a),
cadr(a),
caddr(a),
cadddr(a),
caddddr(a),
cadddddr(a),
caddddddr(a),
cadddddddr(a),
caddddddddr(a),
cadddddddddr(a),
caddddddddddr(a),
cadddddddddddr(a),
caddddddddddddr(a),
cadddddddddddddr(a),
caddddddddddddddr(a),
cadddddddddddddddr(a),
caddddddddddddddddr(a),
cadddddddddddddddddr(a),
caddddddddddddddddddr(a),
cadddddddddddddddddddr(a),
caddddddddddddddddddddr(a),
cadddddddddddddddddddddr(a),
caddddddddddddddddddddddr(a),
cadddddddddddddddddddddddr(a),
caddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddddddddddr(a),
cadddddddddddddddddddddddddddddddddddddddr(a),
caddddddddddddddddddddddddddddddddddddddddr(a)),
s)

DEFINITION:
dual-port-ram-state-body (*bits*, *b-address*, *wen*, *data*, *state*)
= **if** *wen* = **t** **then** *state*
elseif ¬ 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* $\notin$ *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*)
$\land$ ok-netlistp (1, cdr (*fn*), *netlist*, *exceptions*)
else t endif
case = 2
then if primp (*fn*) **then** *fn* $\notin$ *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*)
$\land$ ok-netlistp (3, cdr (*fn*), *netlist*, *exceptions*)
else t endif
otherwise f endcase

THEOREM: primp-monotone
(primp (*fn*)
$\land$ ((*flag* = 0)
$\lor$ ((*flag* = 2)
$\land$ ((*fn* = `'dp-ram-16x32`) $\to$ good-s (*s2*))))
$\land$ (*fn* $\neq$ `'mem-32x32`))
$\to$ (monotonicity-property (*flag*, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)
$\land$ monotonicity-property (*flag*, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*))

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)),
```

```
                              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 (fn,
```

```
                                                                                        netlist)),
                                                              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 (fn,
                                                                                                     netlist)),
                                                              s2),
                                                       delete-module (fn,
                                                                      netlist)),
                                                  pairstates (module-statenames (lookup-module (fn,
                                                                                                netlist)),
                                                        s1),
                                                  pairstates (module-statenames (lookup-module (fn,
                                                                                                netlist)),
                                                        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
(alistp (*alist1*) $\wedge$ alistp (*alist2*) $\wedge$ v-approx-alist (*alist1*, *alist2*))
$\rightarrow$ b-approx (value (x, *alist1*), value (x, *alist2*))

THEOREM: alistp-pairlist
alistp (pairlist (x, y))

THEOREM: alistp-append
alistp (x) $\rightarrow$ (alistp (append (x, y)) = alistp (y))

THEOREM: alistp-dual-eval-1
(alistp (*bindings*) $\wedge$ (*flag* = 1))
$\rightarrow$ alistp (dual-eval (*flag*, *occurrences*, *bindings*, *state-bindints*, *netlist*))

THEOREM: v-approx-alist-implies-v-approx-collect-value
(alistp (*alist1*) $\wedge$ alistp (*alist2*) $\wedge$ v-approx-alist (*alist1*, *alist2*))
$\rightarrow$ v-approx (collect-value (x, *alist1*), collect-value (x, *alist2*))

DEFINITION:
s-approx-list (x, y)
= **if** listp (x)
then listp (y)
$\wedge$ s-approx (car (x), car (y))
$\wedge$ s-approx-list (cdr (x), cdr (y))
else $y \simeq$ **nil endif**

THEOREM: s-approx-list-implies-s-approx-alist
s-approx-list (*s1*, *s2*) $\rightarrow$ s-approx-alist (pairlist (x, *s1*), pairlist (x, *s2*))

THEOREM: v-approx-implies-v-approx-alist
v-approx (*v1*, *v2*) $\rightarrow$ v-approx-alist (pairlist (x, *v1*), pairlist (x, *v2*))

THEOREM: alistp-opener
alistp (cons (a, z)) = (listp (a) $\wedge$ alistp (z))

THEOREM: s-approx-alist-implies-s-approx-value
(s-approx-alist (*s1*, *s2*) $\wedge$ alistp (*s1*) $\wedge$ alistp (*s2*))
$\rightarrow$ s-approx (value (w, *s1*), value (w, *s2*))

THEOREM: v-approx-alist-append
(v-approx-alist (a, c) $\wedge$ v-approx-alist (b, d))
$\rightarrow$ v-approx-alist (append (a, b), append (c, d))

DEFINITION:
double-cdr-induction (x, y)
= **if** listp (x) $\wedge$ listp (y) **then** double-cdr-induction (cdr (x), cdr (y))
else t endif

THEOREM: s-approx-alist-implies-s-approx-list-collect-value
(s-approx-alist $(x, y) \wedge$ alistp $(x) \wedge$ alistp (y))
$\rightarrow$ s-approx-list (collect-value (a, x), collect-value (a, y))

THEOREM: alistp-dual-eval-3
(alistp (*state-bindings*) $\wedge$ (*flag* = 3))
$\rightarrow$ alistp (dual-eval (*flag*, *occurrences*, *bindings*, *state-bindings*, *netlist*))

THEOREM: s-approx-implies-s-approx-alist
s-approx (*s1*, *s2*) $\rightarrow$ s-approx-alist (pairlist (x, *s1*), pairlist (x, *s2*))

THEOREM: s-approx-alist-implies-s-approx-collect-value
(s-approx-alist $(x, y) \wedge$ alistp $(x) \wedge$ alistp (y))
$\rightarrow$ s-approx (collect-value (a, x), collect-value (a, y))

THEOREM: ok-netlistp-reduction
((*flag* = 2) $\vee$ (*flag* = 3))
$\rightarrow$ (ok-netlistp (*flag*, *fn*, *netlist*, *exceptions*)
= **if** *flag* = 2 **then** ok-netlistp (0, *fn*, *netlist*, *exceptions*)
else ok-netlistp (1, *fn*, *netlist*, *exceptions*) **endif**)

THEOREM: ok-netlistp-reduction-rewrite
(ok-netlistp (2, *fn*, *netlist*, *exceptions*)
= ok-netlistp (0, *fn*, *netlist*, *exceptions*))
$\wedge$ (ok-netlistp (3, *fn*, *netlist*, *exceptions*)
= ok-netlistp (1, *fn*, *netlist*, *exceptions*))

THEOREM: dual-eval-monotone-no-ram
ok-netlistp (*flag*, *fn*, *netlist*, `'(mem-32x32 dp-ram-16x32)`)
$\rightarrow$ monotonicity-property (*flag*, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*)

DEFINITION:
good-s-alist (x)
= **if** listp (x)
then listp (car (x)) $\wedge$ good-s (cdar (x)) $\wedge$ good-s-alist (cdr (x))
else x = **nil endif**

THEOREM: good-s-alist-pairlist
good-s (s) $\rightarrow$ good-s-alist (pairlist (x, s))

THEOREM: good-s-value
good-s-alist (x) $\rightarrow$ good-s (value (w, x))

THEOREM: dual-eval-monotone
ok-netlistp (*flag*, *fn*, *netlist*, `'(mem-32x32)`)
$\rightarrow$ (**if** (*flag* = 0) $\vee$ (*flag* = 2) **then** good-s (*s2*)
else good-s-alist (*s2*) **endif**
$\rightarrow$ monotonicity-property (*flag*, *fn*, *netlist*, *a1*, *a2*, *s1*, *s2*))

THEOREM: good-s-collect-value
good-s-alist (s) $\rightarrow$ good-s (collect-value ($keys$, s))

THEOREM: f-buf-preserves-good-s
good-s (f-buf (x))

THEOREM: f-if-preserves-good-s
good-s (f-if (x, y, z))

THEOREM: good-s-0
good-s (`0`)

THEOREM: ao2-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'ao2**, $args$, s))

THEOREM: ao4-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'ao4**, $args$, s))

THEOREM: ao6-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'ao6**, $args$, s))

THEOREM: ao7-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'ao7**, $args$, s))

THEOREM: b-and-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-and**, $args$, s))

THEOREM: b-and3-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-and3**, $args$, s))

THEOREM: b-and4-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-and4**, $args$, s))

THEOREM: b-equv-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-equv**, $args$, s))

THEOREM: b-equv3-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-equv3**, $args$, s))

THEOREM: b-if-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-if**, $args$, s))

THEOREM: b-nand-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-nand**, $args$, s))

THEOREM: b-nand3-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (**'b-nand3**, $args$, s))

THEOREM: b-nand4-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nand4`, $args$, s))

THEOREM: b-nand5-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nand5`, $args$, s))

THEOREM: b-nand6-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nand6`, $args$, s))

THEOREM: b-nand8-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nand8`, $args$, s))

THEOREM: b-nbuf-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nbuf`, $args$, s))

THEOREM: b-nor-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor`, $args$, s))

THEOREM: b-nor3-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor3`, $args$, s))

THEOREM: b-nor4-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor4`, $args$, s))

THEOREM: b-nor5-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor5`, $args$, s))

THEOREM: b-nor6-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor6`, $args$, s))

THEOREM: b-nor8-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-nor8`, $args$, s))

THEOREM: b-not-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-not`, $args$, s))

THEOREM: b-not-b4ip-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-not-b4ip`, $args$, s))

THEOREM: b-not-ivap-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-not-ivap`, $args$, s))

THEOREM: b-or-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-or`, $args$, s))

THEOREM: b-or3-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-or3`, $args$, s))

THEOREM: b-or4-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-or4`, *args*, *s*))

THEOREM: b-xor-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-xor`, *args*, *s*))

THEOREM: b-xor3-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'b-xor3`, *args*, *s*))

THEOREM: del2-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'del2`, *args*, *s*))

THEOREM: del4-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'del4`, *args*, *s*))

THEOREM: del10-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'del10`, *args*, *s*))

THEOREM: procmon-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'procmon`, *args*, *s*))

THEOREM: fd1-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'fd1`, *args*, *s*))

THEOREM: fd1s-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'fd1s`, *args*, *s*))

THEOREM: fd1sp-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'fd1sp`, *args*, *s*))

THEOREM: fd1slp-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'fd1slp`, *args*, *s*))

THEOREM: id-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'id`, *args*, *s*))

THEOREM: ram-enable-circuit-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'ram-enable-circuit`, *args*, *s*))

THEOREM: t-buf-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'t-buf`, *args*, *s*))

THEOREM: t-wire-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'t-wire`, *args*, *s*))

THEOREM: pullup-preserves-good-s
good-s $(s) \rightarrow$ good-s (dual-apply-state (`'pullup`, *args*, *s*))

THEOREM: ttl-bidirect-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-bidirect`, $args$, s))

THEOREM: ttl-clk-input-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-clk-input`, $args$, s))

THEOREM: ttl-input-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-input`, $args$, s))

THEOREM: ttl-output-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-output`, $args$, s))

THEOREM: ttl-output-parametric-preserves-good-s
good-s (s)
$\rightarrow$ good-s (dual-apply-state (`'ttl-output-parametric`, $args$, s))

THEOREM: ttl-output-fast-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-output-fast`, $args$, s))

THEOREM: ttl-tri-output-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-tri-output`, $args$, s))

THEOREM: ttl-tri-output-fast-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'ttl-tri-output-fast`, $args$, s))

THEOREM: vdd-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'vdd`, $args$, s))

THEOREM: vdd-parametric-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'vdd-parametric`, $args$, s))

THEOREM: vss-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'vss`, $args$, s))

THEOREM: good-s-constant-ram
good-s (s) $\rightarrow$ good-s (constant-ram (s, make-list (`32`, X)))

THEOREM: good-s-write-mem-1
(good-s (s) $\wedge$ properp ($value$) $\wedge$ (length ($value$) $=$ `32`))
$\rightarrow$ good-s (write-mem1 (v-$addr$, s, $value$))

THEOREM: good-s-write-mem
(good-s (s) $\wedge$ properp ($value$) $\wedge$ (length ($value$) $=$ `32`))
$\rightarrow$ good-s (write-mem (v-$addr$, s, $value$))

THEOREM: dp-ram-16x32-preserves-good-s
good-s (s) $\rightarrow$ good-s (dual-apply-state (`'dp-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-equv-preserves-good-s, b-equv3-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
(good-s (s) $\wedge$ primp (fn) $\wedge$ $(fn \neq$ `'mem-32x32`$)$)
$\rightarrow$ good-s (dual-apply-state $(fn,\ args,\ s)$)

THEOREM: good-s-preserved
ok-netlistp $(flag,\ fn,\ netlist,$ `'(mem-32x32)`$)$
$\rightarrow$ $((((flag = \texttt{2}) \wedge$ good-s $(s))$
    $\rightarrow$ good-s (dual-eval $(flag,\ fn,\ a,\ s,\ netlist)))$
  $\wedge$ $(((flag = \texttt{3}) \wedge$ good-s-alist $(s))$
      $\rightarrow$ good-s-alist (dual-eval $(flag,\ fn,\ a,\ s,\ netlist))))$

DEFINITION:
v-approx-list $(x,\ y)$
= **if** listp (x)
  then listp (y)
    $\wedge$ v-approx (car (x), car (y))
    $\wedge$ v-approx-list (cdr (x), cdr (y))
  else $y \simeq$ **nil endif**

DEFINITION:
v-s-approx-list $(x,\ y)$
= **if** listp (x)
  then listp (y)

$\wedge$ v-approx (caar (x), caar (y))
$\wedge$ s-approx (cadar (x), cadar (y))
$\wedge$ v-s-approx-list (cdr (x), cdr (y))
else t endif

DEFINITION:
nat-lst-lst-induction (n, x, y)
= **if** ($n \simeq 0$) $\vee$ ($x \simeq$ **nil**) $\vee$ ($y \simeq$ **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*) $\wedge$ ($n <$ length (*final-1*)))
$\rightarrow$ 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*) $\wedge$ ($n <$ length (*final-1*)))
$\rightarrow$ 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*)
$\wedge$ s-approx (*state-1*, *state-2*)
$\wedge$ good-s (*state-2*)
$\wedge$ 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 < 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 < 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*))))

$\wedge$ good-s (*state-2*)
$\wedge$ ok-netlistp (`0`, *fn*, *netlist*, `'(mem-32x32)`)
$\wedge$ ok-netlistp (`2`, *fn*, *netlist*, `'(mem-32x32)`))
$\rightarrow$ (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*))
$\wedge$ 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*))))
$\wedge$ good-s (*state-2*)
$\wedge$ ok-netlistp (`0`, *fn*, *netlist*, `'(mem-32x32)`))
$\rightarrow$ (nth (*n*, simulate (*fn*, *inputs*, *state-1*, *netlist*))
  = nth (*n*, simulate (*fn*, *inputs*, *state-2*, *netlist*)))

DEFINITION: COMPILE-UNCOMPILED-DEFNS = **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 (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)))))),
 x,
 make-list (`32`, x),
 make-list (`4`, 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** $x \simeq$ **nil** **then** $x =$ **nil**
else fourp (car (x)) $\wedge$ v-fourp (cdr (x)) **endif**

DEFINITION:
all-xs (vec)
= **if** $vec \simeq$ **nil** **then** $vec =$ **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 ($machine\text{-}state$)
= ((length ($machine\text{-}state$) = 12)
 $\wedge$ properp ($machine\text{-}state$)

$\wedge$ (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 (cadddar (*machine-state*))
$\wedge$ (length (cadddar (*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 (caddddddddr (*machine-state*))
$\wedge$ fourp (cadddddddddr (*machine-state*))
$\wedge$ v-fourp (caddddddddddr (*machine-state*))
$\wedge$ (length (caddddddddddr (*machine-state*)) = 4)
$\wedge$ v-fourp (cadddddddddddr (*machine-state*))
$\wedge$ (length (cadddddddddddr (*machine-state*)) = 40))

THEOREM: s-approx-make-list
((length (*vec1*) = length (*vec2*)) $\wedge$ v-fourp (*vec2*) $\wedge$ all-xs (*vec1*))
$\rightarrow$ s-approx (*vec1*, *vec2*)

THEOREM: v-fourp-implies-properp
v-fourp (x) $\rightarrow$ properp (x)

THEOREM: listp-implies-not-fourp
listp (x) $\rightarrow$ ($\neg$ fourp (x))

THEOREM: ramp-implies-not-fourp
ramp (x) $\rightarrow$ ($\neg$ fourp (x))

THEOREM: romp-implies-not-fourp
romp (x) $\rightarrow$ ($\neg$ fourp (x))

THEOREM: stubp-implies-not-fourp
stubp $(x) \rightarrow (\neg$ fourp $(x))$

THEOREM: b-approx-x
b-approx (X, y)

THEOREM: machine-state-invariant-implies-s-approx-lemma
(new-machine-state-invariant ($state$)
 $\wedge$ ($state$ = list (list ($s1\text{-}1$, $s1\text{-}2$, $s1\text{-}3$, $s1\text{-}4$),
 $s2$,
 $s3$,
 $s4$,
 $s5$,
 $s6$,
 $s7$,
 $s8$,
 $s9$,
 $s10$,
 $s11$,
 $s12$)))
$\rightarrow$ s-approx (UNKNOWN-MACHINE-STATE, $state$)

THEOREM: machine-state-invariant-implies-s-approx-lemma-2
((length ($state$) = 12)
 $\wedge$ properp ($state$)
 $\wedge$ (length (car ($state$)) = 4)
 $\wedge$ properp (car ($state$)))
$\rightarrow$ ($state$ = list (list (caar ($state$),
 cadar ($state$),
 caddar ($state$),
 cadddar ($state$)),
 cadr ($state$),
 caddr ($state$),
 cadddr ($state$),
 caddddr ($state$),
 cadddddr ($state$),
 caddddddr ($state$),
 cadddddddr ($state$),
 caddddddddr ($state$),
 cadddddddddr ($state$),
 caddddddddddr ($state$),
 cadddddddddddr ($state$)))

THEOREM: machine-state-invariant-implies-s-approx-lemma-3
new-machine-state-invariant ($state$)

$\rightarrow$ ((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*)
→ (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*) → memory-properp (*n*, *width*, *mem*)

THEOREM: new-machine-state-invariant-implies-machine-state-invariant
new-machine-state-invariant (*machine-state*)
→ 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*))))
→ fm9001-statep (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*)))

THEOREM: fm9001=chip-system-true-after-reset-lemma
(chip-system& (*netlist*)
∧ memory-okp (32,
  32,
  cadr (map-up (list (INITIALIZED-MACHINE-STATE, *machine-memory*))))
∧ chip-system-operating-inputs-p (*inputs*,
  total-microcycles (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
  machine-memory))),
  map-up-inputs (*inputs*),
  n))
∧ operating-inputs-p (map-up-inputs (*inputs*),
  total-microcycles (map-down (map-up (list (INITIALIZED-MACHINE-STATE,
  machine-memory))),
  map-up-inputs (*inputs*),
  n)))
→ (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, 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
(($n \not\simeq 0$)
∧ **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-interpreter$after-reset
(chip-system& (*netlist*)
∧ memory-okp (32, 32, *memory*)

$\wedge$ 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))
$\wedge$ 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)))
$\rightarrow$ (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-interpreter$after-reset; name this event 'chip-system=fm9001-interpreter$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'.

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-equv3-preserves-good-s; name this event ‘b-equv3-preserves-good-s-off’.

EVENT: Enable b-equv-preserves-good-s; name this event ‘b-equv-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-equv3-monotone; name this event 'b-equv3-monotone-off'.

EVENT: Enable dual-eval-b-equv3-state; name this event 'dual-eval-b-equv3-state-off'.

EVENT: Enable dual-eval-b-equv3-value; name this event 'dual-eval-b-equv3-value-off'.

EVENT: Enable b-equv-monotone; name this event 'b-equv-monotone-off'.

EVENT: Enable dual-eval-b-equv-state; name this event 'dual-eval-b-equv-state-off'.

EVENT: Enable dual-eval-b-equv-value; name this event 'dual-eval-b-equv-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-equv3-monotone; name this event 'f-equv3-monotone-off'.

EVENT: Enable f-xor3-monotone; name this event 'f-xor3-monotone-off'.

EVENT: Enable f-equv-monotone; name this event 'f-equv-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-knownp-implies-s-approx-is-equal; name this event ‘s-knownp-implies-s-approx-is-equal-off1’.

EVENT: Enable s-knownp; name this event ‘s-knownp-off1’.

EVENT: Enable *1*s-knownp; name this event ‘g*1*s-knownp-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-knownp-implies-v-approx-is-equal; name this event ‘v-knownp-implies-v-approx-is-equal-off’.

EVENT: Enable v-knownp; name this event ‘v-knownp-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-equv; name this event 'f-equv-on'.

EVENT: Disable f-equv3; name this event ‘f-equv3-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 ilessp-add1-iplus; name this event 'ilessp-add1-iplus-on'.

EVENT: Disable ilessp-add1; name this event 'ilessp-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 ineg-eval$-itimes-tree-ineg; name this event 'ineg-eval$-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 ilessp-ineg-ineg; name this event ‘ilessp-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 ilessp-strict; name this event ‘ilessp-strict-on’.

EVENT: Disable ilessp-itimes-right-negative; name this event ‘ilessp-itimes-right-negative-on’.

EVENT: Disable ilessp-itimes-right-positive; name this event ‘ilessp-itimes-right-positive-on’.

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

EVENT: Disable *1*cancel-itimes-ilessp; name this event ‘g*1*cancel-itimes-ilessp-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-iremainder; name this event 'integerp-iremainder-on'.

EVENT: Disable integerp-iquotient; name this event 'integerp-iquotient-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 ineg-0; name this event 'ineg-0-on'.

EVENT: Disable ineg-fix-int; name this event 'ineg-fix-int-on'.

EVENT: Disable ineg-ineg; name this event 'ineg-ineg-on'.

EVENT: Disable ineg-iplus; name this event 'ineg-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-ineg; name this event 'fix-int-ineg-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-ineg; name this event 'integerp-ineg-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*iquo; name this event ‘g*1*iquo-on’.

EVENT: Disable *1*imod; name this event ‘g*1*imod-on’.

EVENT: Disable *1*idiv; name this event ‘g*1*idiv-on’.

EVENT: Disable *1*iremainder; name this event ‘g*1*iremainder-on’.

EVENT: Disable *1*iquotient; name this event ‘g*1*iquotient-on’.

EVENT: Disable *1*itimes; name this event ‘g*1*itimes-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-zerop-tree; name this event ‘or-zerop-tree-on’.

EVENT: Disable *1*or-zerop-tree; name this event ‘g*1*or-zerop-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:
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
$(\mathrm{bvp}\,(x) \land \mathrm{listp}\,(x)) \rightarrow \mathrm{bvp}\,(\mathrm{cdr}\,(x))$

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: 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
$(\text{bvp}\,(bv) \land (\text{length}\,(bv) = (1 + n)))$
$\rightarrow \quad (\text{nth}\,(n,\ bv) = (\text{v-to-nat}\,(bv) \not< \text{exp}\,(2,\ n)))$

THEOREM: remainder-plus-multiple
$((z1 < y1) \wedge ((a \textbf{ mod } y1) = 0)) \rightarrow (((z1 + a) \textbf{ mod } y1) = \text{fix}(z1))$

THEOREM: divides-plus-plus
$((x \textbf{ mod } y) = 0) \rightarrow (((x + x) \textbf{ mod } (y + y)) = 0)$

THEOREM: v-to-nat-firstn-hack1
$(z < y)$
$\rightarrow \quad (((z + z + (w * y) + (w * y)) \textbf{ mod } (y + y)) = (z + z))$

THEOREM: remainder-add1-plus-multiple
$(((1 + z1) < y1) \wedge ((a \textbf{ mod } y1) = 0))$
$\rightarrow \quad (((1 + (z1 + a)) \textbf{ mod } y1) = (1 + z1))$

THEOREM: v-to-nat-firstn-hack2
$(z < y)$
$\rightarrow \quad (((1 + (z + z + (d * y) + (d * y))) \textbf{ mod } (y + y))$
$\quad\quad = \quad (1 + (z + z)))$

THEOREM: v-to-nat-firstn
v-to-nat (firstn (n, v)) = (v-to-nat (v) **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
$(v < \exp(2, x))$
$\rightarrow \quad (((1 + (v + v + (w * \exp(2, x)) + (w * \exp(2, x))))$
$\quad\quad \textbf{mod} \quad (\exp(2, x) + \exp(2, x)))$
$\quad\quad = \quad (1 + (v + v)))$

THEOREM: v-to-nat-of-nat-to-v
v-to-nat (nat-to-v (n, len)) = $(n$ **mod** exp $(2, len))$

THEOREM: remainder-plus-x-x-2
$((x + x) \textbf{ mod } 2) = 0$

THEOREM: quotient-plus-x-x-2
$((x + x) \div 2) = \text{fix}(x)$

THEOREM: nat-to-v-plus-x-x
nat-to-v $(x + x, 1 + n)$ = cons (**f**, nat-to-v (x, n))

THEOREM: firstn-add1-cons
firstn $(1 + n, \text{cons}(a, x)) = \text{cons}(a, \text{firstn}(n, x))$

THEOREM: firstn-zerop
$(n \simeq 0) \rightarrow (\text{firstn}(n, x) = \textbf{nil})$

THEOREM: firstn-nlistp
$(x \simeq \textbf{nil}) \rightarrow (\text{firstn}(n, x) = \textbf{nil})$

THEOREM: nat-to-v-of-v-to-nat-general
$(\text{bvp}(v) \land (\text{length}(v) \not< n))$
$\rightarrow \quad (\text{nat-to-v}(\text{v-to-nat}(v), n) = \text{firstn}(n, v))$

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-zerop
$(len \simeq 0) \rightarrow (\text{nat-to-v}(n, len) = \textbf{nil})$

THEOREM: equal-nat-to-v-inverter-hack1
$((len \in \mathbf{N}) \land (v = \text{nat-to-v}(n, len)))$
$\rightarrow \quad ((\text{v-to-nat}(v) = (n \textbf{ mod } \exp(2, len))) = \textbf{t})$

THEOREM: equal-nat-to-v-inverter-hack2-lemma
$(\text{bvp}(v) \land (\text{v-to-nat}(v) = (n \textbf{ mod } \exp(2, len))))$
$\rightarrow \quad ((\text{nat-to-v}(\text{v-to-nat}(v), len) = \text{nat-to-v}(n \textbf{ mod } \exp(2, len), len))$
$\quad\quad = \quad \textbf{t})$

THEOREM: nat-to-v-remainder
$\text{nat-to-v}(n \textbf{ mod } \exp(2, len), len) = \text{nat-to-v}(n, len)$

THEOREM: equal-nat-to-v-inverter-hack2
$(\text{bvp}(v) \land (\text{v-to-nat}(v) = (n \textbf{ mod } \exp(2, \text{length}(v)))))$
$\rightarrow \quad ((v = \text{nat-to-v}(n, \text{length}(v))) = \textbf{t})$

THEOREM: equal-nat-to-v-inverter
$(v = \text{nat-to-v}(n, len))$
$\leftrightarrow \quad (\text{bvp}(v)$
$\quad\quad \land \quad (\text{length}(v) = \text{fix}(len))$
$\quad\quad \land \quad (\text{v-to-nat}(v) = (n \textbf{ mod } \exp(2, len))))$

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
$\text{bvp}\,(a)$
$\rightarrow \quad ((\text{v-to-nat}\,(a) + \text{v-to-nat}\,(\text{v-not}\,(a))) = (\exp\,(2,\ \text{length}\,(a)) - 1))$

THEOREM: v-to-nat-v-not
$\text{bvp}\,(a)$
$\rightarrow \quad (\text{v-to-nat}\,(\text{v-not}\,(a)) = ((\exp\,(2,\ \text{length}\,(a)) - \text{v-to-nat}\,(a)) - 1))$

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-subtracter-output-lemma
$((\textit{a-nat} < \textit{bound}) \wedge (\textit{b-nat} < \textit{bound}))$
$\rightarrow \quad ((\textit{b-nat} + ((\textit{bound} - \textit{a-nat}) - 1))$
$\quad\quad = \quad (((\textit{b-nat} + \textit{bound}) - \textit{a-nat}) - 1))$

THEOREM: lessp-v-to-nat-exp-rewrite
$(\text{bvp}\,(a) \wedge (\text{length}\,(a) = \textit{len})) \rightarrow ((\text{v-to-nat}\,(a) < \exp\,(2,\ \textit{len})) = \mathbf{t})$

THEOREM: not-lessp-quotient
$n \not< (n \div k)$

THEOREM: v-to-nat-append
$(\text{bvp}\,(a) \wedge \text{bvp}\,(b))$
$\rightarrow \quad (\text{v-to-nat}\,(\text{append}\,(a,\ b))$
$\quad\quad = \quad (\text{v-to-nat}\,(a) + (\exp\,(2,\ \text{length}\,(a)) * \text{v-to-nat}\,(b))))$

DEFINITION:
$\text{v-alu-nat-adder-output}\,(c,\ a,\ b)$
$= \quad \text{nat-to-v}\,((\text{b-to-nat}\,(c) + \text{v-to-nat}\,(a) + \text{v-to-nat}\,(b))$
$\quad\quad\quad\quad \textbf{mod} \quad \exp\,(2,\ \text{length}\,(a)),$
$\quad\quad\quad\quad \text{length}\,(a))$

DEFINITION:
$\text{v-alu-nat-adder-carry-out}\,(c,\ a,\ b)$
$= \quad ((\text{b-to-nat}\,(c) + \text{v-to-nat}\,(a) + \text{v-to-nat}\,(b)) \not< \exp\,(2,\ \text{length}\,(a)))$

DEFINITION:
$\text{v-alu-nat-adder}\,(c,\ a,\ b)$
$= \quad \text{cvzbv}\,(\text{v-alu-nat-adder-carry-out}\,(c,\ a,\ b),$
$\quad\quad\quad \text{v-adder-overflowp}\,(c,\ a,\ b),$
$\quad\quad\quad \text{v-alu-nat-adder-output}\,(c,\ a,\ b))$

THEOREM: v-alu-correct-nat-adder-carry-out
bv2p $(a,\ b)$
$\rightarrow$ (v-adder-carry-out $(c,\ a,\ b)$ = v-alu-nat-adder-carry-out $(c,\ a,\ b)$)

THEOREM: v-alu-correct-nat-adder-output
bv2p $(a,\ b)$ $\rightarrow$ (v-adder-output $(c,\ a,\ b)$ = v-alu-nat-adder-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-nat-adder
bv2p $(a,\ b)$ $\rightarrow$ (cvzbv-v-adder $(c,\ a,\ b)$ = v-alu-nat-adder $(c,\ a,\ b)$)

DEFINITION:
v-alu-nat-inc (a) = v-alu-nat-adder (**t**, a, nat-to-v (0, length (a)))

THEOREM: v-alu-correct-nat-inc
bvp (a) $\rightarrow$ (cvzbv-inc (a) = v-alu-nat-inc (a))

DEFINITION:
v-alu-nat-subtracter-output $(c,\ a,\ b)$
= nat-to-v (((v-to-nat (b) + exp (2, length (a)))
− (v-to-nat (a) + b-to-nat (c)))
mod exp (2, length (a)),
length (a))

DEFINITION:
v-alu-nat-subtracter-carry-out $(c,\ a,\ b)$
= (v-to-nat (b) < (v-to-nat (a) + b-to-nat (c)))

DEFINITION:
v-alu-nat-subtracter $(c,\ a,\ b)$
= cvzbv (v-alu-nat-subtracter-carry-out $(c,\ a,\ b)$,
v-subtracter-overflowp $(c,\ a,\ b)$,
v-alu-nat-subtracter-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-nat-subtracter-carry-out
bv2p $(a,\ b)$
$\rightarrow$ (v-subtracter-carry-out $(c,\ a,\ b)$
= v-alu-nat-subtracter-carry-out $(c,\ a,\ b)$)

THEOREM: v-alu-correct-nat-subtracter-output
bv2p $(a,\ b)$
$\rightarrow$ (v-subtracter-output $(c,\ a,\ b)$ = v-alu-nat-subtracter-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-nat-subtracter
bv2p $(a,\ b)$ $\rightarrow$ (cvzbv-v-subtracter $(c,\ a,\ b)$ = v-alu-nat-subtracter $(c,\ a,\ b)$)

DEFINITION:
v-alu-nat-dec (a) = v-alu-nat-subtracter (**t**, nat-to-v (0, length (a)), a)

THEOREM: v-alu-correct-nat-dec
bvp $(a) \rightarrow$ (cvzbv-dec (a) = v-alu-nat-dec (a))

DEFINITION:
v-alu-nat-lsr-output (a) = nat-to-v (v-to-nat $(a) \div 2$, length (a))

DEFINITION:
v-alu-nat-lsr-carry-out (a) = ((v-to-nat (a) **mod** 2) $\neq 0$)

DEFINITION:
v-alu-nat-lsr (a)
= cvzbv (v-alu-nat-lsr-carry-out (a), **f**, v-alu-nat-lsr-output (a))

THEOREM: v-alu-correct-nat-lsr-carry-out
(bvp $(a) \land$ listp (a)) $\rightarrow$ (nth $(0, a)$ = v-alu-nat-lsr-carry-out (a))

THEOREM: v-alu-correct-nat-lsr-output
bvp $(a) \rightarrow$ (v-lsr (a) = v-alu-nat-lsr-output (a))

THEOREM: v-alu-correct-nat-lsr
bvp $(a) \rightarrow$ (cvzbv-v-lsr (a) = v-alu-nat-lsr (a))

DEFINITION:
v-alu-nat-not-output (a)
= nat-to-v ((exp (2, length (a)) $-$ v-to-nat (a)) $- 1$, length (a))

DEFINITION:
v-alu-nat-not (a) = cvzbv (**f**, **f**, v-alu-nat-not-output (a))

THEOREM: v-alu-correct-nat-not-output
bvp $(a) \rightarrow$ (v-not (a) = v-alu-nat-not-output (a))

THEOREM: v-alu-correct-nat-not
bvp $(a) \rightarrow$ (cvzbv-v-not (a) = v-alu-nat-not (a))

DEFINITION:
v-alu-nat (c, a, b, op)
= **if** op = list (**f**, **f**, **f**, **f**) **then** v-alu-nat-int-buf (a)
elseif op = list (**t**, **f**, **f**, **f**) **then** v-alu-nat-inc (a)
elseif op = list (**f**, **t**, **f**, **f**) **then** v-alu-nat-adder (c, a, b)
elseif op = list (**t**, **t**, **f**, **f**) **then** v-alu-nat-adder (**f**, a, b)
elseif op = list (**t**, **f**, **t**, **f**) **then** v-alu-nat-dec (a)
elseif op = list (**f**, **t**, **t**, **f**) **then** v-alu-nat-subtracter (c, a, b)
elseif op = list (**t**, **t**, **t**, **f**) **then** v-alu-nat-subtracter (**f**, a, b)
elseif op = list (**f**, **t**, **f**, **t**) **then** v-alu-nat-lsr (a)
elseif op = list (**f**, **t**, **t**, **t**) **then** v-alu-nat-not (a)
else v-alu (c, a, b, op) **endif**

THEOREM: v-alu-correct-nat
$\text{bv2p}(a, b) \rightarrow (\text{v-alu}(c, a, b, op) = \text{v-alu-nat}(c, a, b, op))$

DEFINITION:
$\text{int-to-v}(i, l)$
$= \ \ \textbf{if } \text{ilessp}(i, 0) \ \textbf{then } \text{nat-to-v}(\text{iplus}(i, \exp(2, l)), l)$
$\quad\ \ \textbf{else } \text{nat-to-v}(i, l) \ \textbf{endif}$

DEFINITION:
$\text{v-to-int}(v)$
$= \ \ \textbf{if } \text{v-to-nat}(v) < \exp(2, \text{length}(v) - 1) \ \textbf{then } \text{v-to-nat}(v)$
$\quad\ \ \textbf{else } \text{idifference}(\text{v-to-nat}(v), \exp(2, \text{length}(v))) \ \textbf{endif}$

DEFINITION:
$\text{integer-in-rangep}(i, l)$
$= \ \ (\text{ileq}(\text{ineg}(\exp(2, l - 1)), i) \land \text{ilessp}(i, \exp(2, l - 1)))$

THEOREM: iplus-plus
$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow (\text{iplus}(x, y) = (x + y))$

THEOREM: ilessp-lessp
$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow (\text{ilessp}(x, y) = (x < y))$

THEOREM: times-2
$(2 * x) = (x + x)$

THEOREM: b-to-nat-leq
$1 \not< \text{b-to-nat}(x)$

THEOREM: lessp-v-to-nat-exp-with-exp-opened
$(\text{bvp}(a) \land (\text{length}(a) = \text{length}(b)) \land (\text{length}(b) \neq 0))$
$\rightarrow \ \ (\text{v-to-nat}(a) < (\exp(2, \text{length}(b) - 1) + \exp(2, \text{length}(b) - 1)))$

THEOREM: fix-int-numberp
$(x \in \mathbf{N}) \rightarrow (\text{fix-int}(x) = x)$

THEOREM: iplus3-plus-difference
$((c0 \in \mathbf{N})$
$\land \ \ (a0 \in \mathbf{N})$
$\land \ \ (b0 \in \mathbf{N})$
$\land \ \ (d \in \mathbf{N})$
$\land \ \ ((c0 + a0 + b0) \not< d))$
$\rightarrow \ \ (\text{iplus}(c0, \text{iplus}(a0, \text{iplus}(b0, \text{ineg}(d)))) = ((c0 + a0 + b0) - d))$

THEOREM: v-to-int-v-not
$(\text{bvp}(a) \land (\text{length}(a) \neq 0))$
$\rightarrow \ \ (\text{v-to-int}(\text{v-not}(a)) = \text{iplus}(-1, \text{ineg}(\text{v-to-int}(a))))$

EVENT: Enable v-alu-correct-nat-not-output; name this event 'v-alu-correct-nat-not-output-off'.

THEOREM: v-alu-correct-int-subtracter-overflowp-lemma

$$(\text{bv2p}\,(a,\ b) \land (\text{length}\,(a) \neq 0))$$
$$\rightarrow \quad (\text{iplus}\,(\text{b-to-nat}\,(\text{b-not}\,(c)),\ \text{iplus}\,(\text{v-to-int}\,(\text{v-not}\,(a)),\ \text{v-to-int}\,(b)))$$
$$= \quad \text{idifference}\,(\text{v-to-int}\,(b),\ \text{iplus}\,(\text{v-to-int}\,(a),\ \text{b-to-nat}\,(c))))$$

THEOREM: idifference-difference

$$((x \in \mathbf{N}) \land (y \in \mathbf{N}) \land (x \not< y))$$
$$\rightarrow \quad ((\text{iplus}\,(x,\ \text{ineg}\,(y)) = (x - y))$$
$$\land \quad (\text{iplus}\,(\text{ineg}\,(y),\ x) = (x - y)))$$

THEOREM: idiv-quotient

$$((x \in \mathbf{N}) \land (y \in \mathbf{N})) \rightarrow (\text{idiv}\,(x,\ y) = (x \div y))$$

THEOREM: idiv-ilessp-0

$$((y \in \mathbf{N}) \land (y \neq 0)) \rightarrow (\text{ilessp}\,(\text{idiv}\,(x,\ y),\ 0) = \text{ilessp}\,(x,\ 0))$$

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

$$((a0 \in \mathbf{N}) \land (exp \in \mathbf{N}))$$
$$\rightarrow \quad (\text{iplus}\,(\text{idiv}\,(\text{iplus}\,(a0,\ \text{ineg}\,(exp)),\ 2),\ exp) = ((a0 + exp) \div 2))$$

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:
v-alu-int-adder-output $(c,\ a,\ b)$
$=$ int-to-v (iplus (b-to-nat (c), iplus (v-to-int (a), v-to-int (b))), length (a))

DEFINITION:
v-alu-int-adder-overflowp $(c,\ a,\ b)$
$=$ ($\neg$ integer-in-rangep (iplus (b-to-nat (c),
iplus (v-to-int (a), v-to-int (b))),
length (a)))

DEFINITION:
v-alu-int-adder $(c,\ a,\ b)$
$=$ cvzbv (v-adder-carry-out $(c,\ a,\ b)$,
v-alu-int-adder-overflowp $(c,\ a,\ b)$,
v-alu-int-adder-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-adder-overflowp
(bv2p $(a,\ b)$ $\land$ (length $(a) \neq 0$))
$\rightarrow$ (v-adder-overflowp $(c,\ a,\ b)$ $=$ v-alu-int-adder-overflowp $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-adder-output
(bv2p $(a,\ b)$ $\land$ (length $(a) \neq 0$))
$\rightarrow$ (v-adder-output $(c,\ a,\ b)$ $=$ v-alu-int-adder-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-adder
(bv2p $(a,\ b)$ $\land$ (length $(a) \neq 0$))
$\rightarrow$ (cvzbv-v-adder $(c,\ a,\ b)$ $=$ v-alu-int-adder $(c,\ a,\ b)$)

DEFINITION:
v-alu-int-inc (a) $=$ v-alu-int-adder ($\mathtt{t}$, a, int-to-v (0, length (a)))

THEOREM: v-alu-correct-int-inc
(bvp (a) $\land$ (length $(a) \neq 0$)) $\rightarrow$ (cvzbv-inc (a) $=$ v-alu-int-inc (a))

DEFINITION:
v-alu-int-subtracter-output $(c,\ a,\ b)$
$=$ int-to-v (idifference (v-to-int (b), iplus (v-to-int (a), b-to-nat (c))),
length (a))

DEFINITION:
v-alu-int-subtracter-overflowp $(c,\ a,\ b)$
$=$ ($\neg$ integer-in-rangep (idifference (v-to-int (b),
iplus (v-to-int (a), b-to-nat (c))),
length (a)))

DEFINITION:
v-alu-int-subtracter $(c,\ a,\ b)$
$=$ cvzbv (v-subtracter-carry-out $(c,\ a,\ b)$,
v-alu-int-subtracter-overflowp $(c,\ a,\ b)$,
v-alu-int-subtracter-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-subtracter-overflowp
(bv2p $(a,\ b)$ $\wedge$ (length $(a) \neq$ 0))
$\rightarrow$ (v-subtracter-overflowp $(c,\ a,\ b)$
$=$ v-alu-int-subtracter-overflowp $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-subtracter-output
(bv2p $(a,\ b)$ $\wedge$ (length $(a) \neq$ 0))
$\rightarrow$ (v-subtracter-output $(c,\ a,\ b)$ $=$ v-alu-int-subtracter-output $(c,\ a,\ b)$)

THEOREM: v-alu-correct-int-subtracter
(bv2p $(a,\ b)$ $\wedge$ (length $(a) \neq$ 0))
$\rightarrow$ (cvzbv-v-subtracter $(c,\ a,\ b)$ $=$ v-alu-int-subtracter $(c,\ a,\ b)$)

DEFINITION:
v-alu-int-dec (a) $=$ v-alu-int-subtracter (**t**, int-to-v (0, length (a)), a)

THEOREM: v-alu-correct-int-dec
(bvp (a) $\wedge$ (length $(a) \neq$ 0)) $\rightarrow$ (cvzbv-dec (a) $=$ v-alu-int-dec (a))

DEFINITION:
v-alu-int-asr-output (a) $=$ int-to-v (idiv (v-to-int (a), 2), length (a))

DEFINITION:
v-alu-int-asr (a)
$=$ cvzbv (v-alu-nat-lsr-carry-out (a), **f**, v-alu-int-asr-output (a))

THEOREM: v-alu-correct-int-asr-output
(bvp (a) $\wedge$ (length $(a) \neq$ 0)) $\rightarrow$ (v-asr (a) $=$ v-alu-int-asr-output (a))

THEOREM: v-alu-correct-int-asr
bvp (a) $\rightarrow$ (cvzbv-v-asr (a) $=$ v-alu-int-asr (a))

DEFINITION:
v-alu-int-neg (a) $=$ v-alu-int-subtracter (**f**, a, int-to-v (0, length (a)))

THEOREM: v-alu-correct-int-neg
(bvp (a) $\wedge$ (length $(a) \neq$ 0)) $\rightarrow$ (cvzbv-neg (a) $=$ v-alu-int-neg (a))

DEFINITION:
v-alu-int-not-output (a)
$=$ v-alu-int-subtracter-output (**t**, a, int-to-v (0, length (a)))

DEFINITION:
v-alu-int-not (a) = cvzbv (**f**, **f**, v-alu-int-not-output (a))

THEOREM: v-alu-correct-int-not-output
(bvp (a) $\wedge$ (length $(a) \neq 0$)) $\rightarrow$ (v-not (a) = v-alu-int-not-output (a))

THEOREM: v-alu-correct-int-not
(bvp (a) $\wedge$ (length $(a) \neq 0$)) $\rightarrow$ (cvzbv-v-not (a) = v-alu-int-not (a))

DEFINITION:
v-alu-int (c, a, b, op)
= **if** op = list (**f**, **f**, **f**, **f**) **then** v-alu-nat-int-buf (a)
elseif op = list (**t**, **f**, **f**, **f**) **then** v-alu-int-inc (a)
elseif op = list (**f**, **t**, **f**, **f**) **then** v-alu-int-adder (c, a, b)
elseif op = list (**t**, **t**, **f**, **f**) **then** v-alu-int-adder (**f**, a, b)
elseif op = list (**f**, **f**, **t**, **f**) **then** v-alu-int-neg (a)
elseif op = list (**t**, **f**, **t**, **f**) **then** v-alu-int-dec (a)
elseif op = list (**f**, **t**, **t**, **f**) **then** v-alu-int-subtracter (c, a, b)
elseif op = list (**t**, **t**, **t**, **f**) **then** v-alu-int-subtracter (**f**, a, b)
elseif op = list (**t**, **f**, **f**, **t**) **then** v-alu-int-asr (a)
elseif op = list (**f**, **t**, **t**, **t**) **then** v-alu-int-not (a)
else v-alu (c, a, b, op) **endif**

THEOREM: v-alu-correct-int
(bv2p (a, b) $\wedge$ (length $(a) \neq 0$))
$\rightarrow$ (v-alu (c, a, b, op) = 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-subtracter-output; name this event 'v-alu-correct-nat-subtracter-output-off'.

EVENT: Enable v-alu-correct-nat-subtracter; name this event 'v-alu-correct-nat-subtracter-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-subtracter-overflowp; name this event 'v-alu-correct-int-subtracter-overflowp-off'.

EVENT: Enable v-alu-correct-int-subtracter-output; name this event 'v-alu-correct-int-subtracter-output-off'.

EVENT: Enable v-alu-correct-int-subtracter; name this event 'v-alu-correct-int-subtracter-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-subtracter; name this event ‘v-alu-int-subtracter-off’.

EVENT: Enable *1*v-alu-int-subtracter; name this event ‘g*1*v-alu-int-subtracter-off’.

EVENT: Enable v-alu-int-subtracter-overflowp; name this event ‘v-alu-int-subtracter-overflowp-off’.

EVENT: Enable *1*v-alu-int-subtracter-overflowp; name this event ‘g*1*v-alu-int-subtracter-overflowp-off’.

EVENT: Enable v-alu-int-subtracter-output; name this event ‘v-alu-int-subtracter-output-off’.

EVENT: Enable *1*v-alu-int-subtracter-output; name this event ‘g*1*v-alu-int-subtracter-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 ilessp-lessp; name this event ‘ilessp-lessp-off’.

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

EVENT: Enable integer-in-rangep; name this event ‘integer-in-rangep-off’.

EVENT: Enable *1*integer-in-rangep; name this event ‘g*1*integer-in-rangep-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-subtracter; name this event 'v-alu-nat-subtracter-off'.

EVENT: Enable *1*v-alu-nat-subtracter; name this event 'g*1*v-alu-nat-subtracter-off'.

EVENT: Enable v-alu-nat-subtracter-carry-out; name this event 'v-alu-nat-subtracter-carry-out-off'.

EVENT: Enable *1*v-alu-nat-subtracter-carry-out; name this event 'g*1*v-alu-nat-subtracter-carry-out-off'.

EVENT: Enable v-alu-nat-subtracter-output; name this event 'v-alu-nat-subtracter-output-off'.

EVENT: Enable *1*v-alu-nat-subtracter-output; name this event 'g*1*v-alu-nat-subtracter-output-off'.

EVENT: Enable v-alu-nat-inc; name this event 'v-alu-nat-inc-off'.

EVENT: Enable *1*v-alu-nat-inc; name this event 'g*1*v-alu-nat-inc-off'.

EVENT: Enable v-alu-nat-adder; name this event 'v-alu-nat-adder-off'.

EVENT: Enable *1*v-alu-nat-adder; name this event 'g*1*v-alu-nat-adder-off'.

EVENT: Enable v-alu-nat-adder-carry-out; name this event 'v-alu-nat-adder-carry-out-off'.

EVENT: Enable *1*v-alu-nat-adder-carry-out; name this event 'g*1*v-alu-nat-adder-carry-out-off'.

EVENT: Enable v-alu-nat-adder-output; name this event 'v-alu-nat-adder-output-off'.

EVENT: Enable *1*v-alu-nat-adder-output; name this event 'g*1*v-alu-nat-adder-output-off'.

EVENT: Enable v-to-nat-append; name this event 'v-to-nat-append-off'.

EVENT: Enable not-lessp-quotient; name this event 'not-lessp-quotient-off'.

EVENT: Enable lessp-v-to-nat-exp-rewrite; name this event 'lessp-v-to-nat-exp-rewrite-off'.

EVENT: Enable v-alu-correct-nat-subtracter-output-lemma; name this event 'v-alu-correct-nat-subtracter-output-lemma-off'.

EVENT: Enable lessp-v-to-nat-exp; name this event '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-zerop; name this event ‘nat-to-v-zerop-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-zerop; name this event ‘firstn-zerop-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-interpreter$after-reset; name this event ‘chip-system=fm9001-interpreter$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-equv3-preserves-good-s; name this event 'b-equv3-preserves-good-s-on'.

EVENT: Disable b-equv-preserves-good-s; name this event 'b-equv-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-equv3-monotone; name this event 'b-equv3-monotone-on'.

EVENT: Disable dual-eval-b-equv3-state; name this event 'dual-eval-b-equv3-state-on'.

EVENT: Disable dual-eval-b-equv3-value; name this event 'dual-eval-b-equv3-value-on'.

EVENT: Disable b-equv-monotone; name this event 'b-equv-monotone-on'.

EVENT: Disable dual-eval-b-equv-state; name this event 'dual-eval-b-equv-state-on'.

EVENT: Disable dual-eval-b-equv-value; name this event 'dual-eval-b-equv-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-equv3-monotone; name this event 'f-equv3-monotone-on'.

EVENT: Disable f-xor3-monotone; name this event 'f-xor3-monotone-on'.

EVENT: Disable f-equv-monotone; name this event 'f-equv-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-knownp-implies-s-approx-is-equal; name this event 's-knownp-implies-s-approx-is-equal-on1'.

EVENT: Disable s-knownp; name this event 's-knownp-on1'.

EVENT: Disable *1*s-knownp; name this event 'g*1*s-knownp-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-knownp-implies-v-approx-is-equal; name this event 'v-knownp-implies-v-approx-is-equal-on'.

EVENT: Disable v-knownp; name this event 'v-knownp-on1'.

EVENT: Disable *1*v-knownp; name this event 'g*1*v-knownp-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-knownp-compound-recognizer; name this event 'b-knownp-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-equv; name this event 'f-equv-off1'.

EVENT: Enable f-equv3; name this event 'f-equv3-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 ilessp-add1-iplus; name this event 'ilessp-add1-iplus-off1'.

EVENT: Enable ilessp-add1; name this event 'ilessp-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*cancel-ineg-terms-from-inequality-expanded; name this event 'g*1*cancel-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*cancel-ineg-terms-from-inequality; name this event 'g*1*cancel-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 ineg-eval$-itimes-tree-ineg; name this event ‘ineg-eval$-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 ilessp-ineg-ineg; name this event ‘ilessp-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 ilessp-strict; name this event ‘ilessp-strict-off1’.

EVENT: Enable ilessp-itimes-right-negative; name this event ‘ilessp-itimes-right-negative-off1’.

EVENT: Enable ilessp-itimes-right-positive; name this event ‘ilessp-itimes-right-positive-off1’.

EVENT: Enable cancel-itimes-ilessp; name this event ‘cancel-itimes-ilessp-off1’.

EVENT: Enable *1*cancel-itimes-ilessp; name this event ‘g*1*cancel-itimes-ilessp-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 ilessp-fix-int-2; name this event 'ilessp-fix-int-2-off1'.

EVENT: Enable ilessp-fix-int-1; name this event 'ilessp-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-subtracter-carry-out; name this event ‘v-alu-correct-nat-subtracter-carry-out-on’.

EVENT: Disable v-alu-correct-nat-subtracter-output; name this event ‘v-alu-correct-nat-subtracter-output-on’.

EVENT: Disable v-alu-correct-nat-subtracter; name this event ‘v-alu-correct-nat-subtracter-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-subtracter-overflowp; name this event 'v-alu-correct-int-subtracter-overflowp-on'.

EVENT: Disable v-alu-correct-int-subtracter-output; name this event 'v-alu-correct-int-subtracter-output-on'.

EVENT: Disable v-alu-correct-int-subtracter; name this event 'v-alu-correct-int-subtracter-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
v-negp $(v) \rightarrow$ v-nzerop (v)

THEOREM: v-zerop-implies-v-to-nat-0
v-zerop $(a) \rightarrow$ (v-to-nat $(a) = 0$)

THEOREM: boolp-flag-extracters
boolp (c-flag (update-flags $(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv})$))
$\wedge$ boolp (v-flag (update-flags $(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv})$))
$\wedge$ boolp (n-flag (update-flags $(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv})$))
$\wedge$ boolp (z-flag (update-flags $(\mathit{flags},\ \mathit{set\text{-}flags},\ \mathit{cvzbv})$))

THEOREM: v-to-nat-equal
bv2p $(a,\ b) \rightarrow$ ((v-to-nat (a) = v-to-nat (b)) = $(a = b)$)

THEOREM: v-nzerop-nat-to-v-zero
$(z \simeq 0) \rightarrow (\neg$ v-nzerop (nat-to-v $(z,\ l)$))

THEOREM: v-not-v-not
bvp $(a) \rightarrow$ (v-not (v-not (a)) = a)

THEOREM: rewrite-equal-v-to-nat-0
(v-to-nat (a) = 0) = v-zerop (a)

EVENT: Enable rewrite-equal-v-to-nat-0; name this event 'rewrite-equal-v-to-nat-0-off'.

DEFINITION: CC-CC = list (**f**, **f**, **f**, **f**)

DEFINITION: CC-CS = list (**t**, **f**, **f**, **f**)

DEFINITION: CC-VC = list (**f**, **t**, **f**, **f**)

DEFINITION: CC-VS = list (**t**, **t**, **f**, **f**)

DEFINITION: CC-PL = list (**f**, **f**, **t**, **f**)

DEFINITION: CC-MI = list (**t**, **f**, **t**, **f**)

DEFINITION: CC-NE = list (**f**, **t**, **t**, **f**)

DEFINITION: CC-EQ = list (**t**, **t**, **t**, **f**)

DEFINITION: CC-HI = list (**f**, **f**, **f**, **t**)

DEFINITION: CC-LS = list (**t**, **f**, **f**, **t**)

DEFINITION: CC-GE = list (**f**, **t**, **f**, **t**)

DEFINITION: CC-LT = list (**t**, **t**, **f**, **t**)

DEFINITION: CC-GT = list (**f**, **f**, **t**, **t**)

DEFINITION: CC-LE = list (**t**, **f**, **t**, **t**)

DEFINITION: CC-T = list (**f**, **t**, **t**, **t**)

DEFINITION: CC-F = list (**t**, **t**, **t**, **t**)

EVENT: Let us define the theory *condition-code-theory* to consist of the following events: cc-cc, cc-cs, cc-vc, cc-vs, cc-pl, cc-mi, cc-ne, cc-eq, cc-hi, cc-ls, cc-ge, cc-lt, cc-gt, cc-le, cc-t, cc-f.

DEFINITION: OP-MOVE = list (**f**, **f**, **f**, **f**)

DEFINITION: OP-INC = list (**t**, **f**, **f**, **f**)

DEFINITION: OP-ADDC = list (**f**, **t**, **f**, **f**)

DEFINITION: OP-ADD = list (**t**, **t**, **f**, **f**)

DEFINITION: OP-NEG = list (**f**, **f**, **t**, **f**)

DEFINITION: OP-DEC = list (**t**, **f**, **t**, **f**)

DEFINITION: OP-SUBB = list (**f**, **t**, **t**, **f**)

DEFINITION: OP-SUB = list (**t**, **t**, **t**, **f**)

DEFINITION: OP-ROR = list (**f**, **f**, **f**, **t**)

DEFINITION: OP-ASR = list (**t**, **f**, **f**, **t**)

DEFINITION: OP-LSR = list (**f**, **t**, **f**, **t**)

DEFINITION: OP-XOR = list (**t**, **t**, **f**, **t**)

DEFINITION: OP-OR = list (**f**, **f**, **t**, **t**)

DEFINITION: OP-AND = list (**t**, **f**, **t**, **t**)

DEFINITION: OP-NOT = list (**f**, **t**, **t**, **t**)

DEFINITION: OP-M15 = list (**t**, **t**, **t**, **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
zb (v-alu (c, a, b, *op-code*)) = v-zerop (bv (v-alu (c, a, b, *op-code*)))

THEOREM: n-v-alu
n (v-alu (c, a, b, *op-code*)) = v-negp (bv (v-alu (c, a, b, *op-code*)))

THEOREM: universal-flags-interpretations
(bvp (a) ∧ boolp (c) ∧ (length (a) ≄ 0))
→ ((z-set (*set-flags*)
  → ((store-resultp (CC-NE,
      update-flags (*flags*,
        set-flags,
        v-alu (c, a, b, *op-code*)))
    = v-nzerop (bv (v-alu (c, a, b, *op-code*))))
   ∧ (store-resultp (CC-EQ,
      update-flags (*flags*,
        set-flags,

v-alu (*c*, *a*, *b*, *op-code*)))
$=$ ($\neg$ v-nzerop (bv (v-alu (*c*, *a*, *b*, *op-code*)))))))
$\wedge$ (n-set (*set-flags*)
$\rightarrow$ ((store-resultp (CC-PL,
update-flags (*flags*,
set-flags,
v-alu (*c*, *a*, *b*, *op-code*)))
$=$ ($\neg$ v-negp (bv (v-alu (*c*, *a*, *b*, *op-code*)))))
$\wedge$ (store-resultp (CC-MI,
update-flags (*flags*,
set-flags,
v-alu (*c*,
a,
b,
op-code)))
$=$ v-negp (bv (v-alu (*c*, *a*, *b*, *op-code*))))))
$\wedge$ (store-resultp (CC-T,
update-flags (*flags*,
set-flags,
v-alu (*c*, *a*, *b*, *op-code*)))
$=$ **t**)
$\wedge$ (store-resultp (CC-F,
update-flags (*flags*,
set-flags,
v-alu (*c*, *a*, *b*, *op-code*)))
$=$ **f**))

THEOREM: v-sum-on-not-a-a
bv2p (*a*, *b*) $\rightarrow$ (v-nzerop (v-sum (**t**, *a*, *b*)) = (v-not (*a*) $\neq$ *b*))

THEOREM: if-v-subtracter-output=0-then-a=b
bv2p (*a*, *b*) $\rightarrow$ (v-nzerop (v-subtracter-output (**f**, *a*, *b*)) = (*a* $\neq$ *b*))

THEOREM: zero-flag-interpretation-for-sub
(bv2p (*a*, *b*) $\wedge$ z-set (*set-flags*))
$\rightarrow$ (z-flag (update-flags (*flags*, *set-flags*, v-alu (c-flag (*flags*), *a*, *b*, OP-SUB)))
$=$ (*a* = *b*))

THEOREM: flags-interpretation-sub
(bv2p (*a*, *b*) $\wedge$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-NE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-SUB)))
$=$ (*a* $\neq$ *b*))

$$
\begin{array}{l}
\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-sub})))\\
\qquad = \quad (a = b)))
\end{array}
$$

THEOREM: v-negp-v-shift-right

$$((\text{length}\,(a) \neq \mathtt{0}) \wedge \text{boolp}\,(c)) \rightarrow (\text{v-negp}\,(\text{v-shift-right}\,(a,\ c)) = c)$$

THEOREM: flags-interpretation-ror

$$
\begin{array}{l}
(\text{bvp}\,(a) \wedge \text{boolp}\,(\text{c-flag}\,(\mathit{flags})) \wedge (\text{length}\,(a) \not\simeq \mathtt{0}))\\
\rightarrow \quad ((\text{c-set}\,(\mathit{set\text{-}flags})\\
\quad \rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-cc},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-ror})))\\
\qquad = \quad (\neg\ \text{nth}\,(\mathtt{0},\ a)))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-cs},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\quad \textsc{op-ror})))\\
\qquad\quad = \quad \text{nth}\,(\mathtt{0},\ a))))\\
\quad \wedge \quad (\text{n-set}\,(\mathit{set\text{-}flags})\\
\qquad \rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-pl},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\quad \textsc{op-ror})))\\
\qquad\quad = \quad (\neg\ \text{c-flag}\,(\mathit{flags})))\\
\qquad\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-mi},\\
\qquad\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-ror})))\\
\qquad\qquad = \quad \text{c-flag}\,(\mathit{flags})))))
\end{array}
$$

THEOREM: flags-interpretation-lsr

$$(\text{bvp}\,(a) \wedge (\text{length}\,(a) \not\simeq \mathtt{0}) \wedge \text{c-set}\,(\mathit{set\text{-}flags}))$$

$$
\begin{array}{l}
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-cc},\\
\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
\qquad = \quad (\neg\ \text{nth}\,(\mathtt{0},\ a)))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-cs},\\
\qquad\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
\qquad\qquad = \quad \text{nth}\,(\mathtt{0},\ a)))
\end{array}
$$

THEOREM: flags-interpretation-xor

$$
\begin{array}{l}
(\text{bv2p}\,(a,\ b) \wedge (\text{length}\,(a) \not\simeq \mathtt{0}) \wedge \text{z-set}\,(\mathit{set\text{-}flags}))\\
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-xor})))\\
\qquad = \quad (a \neq b))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-xor})))\\
\qquad\qquad = \quad (a = b)))
\end{array}
$$

THEOREM: flags-interpretation-nat-move

$$
\begin{array}{l}
(\text{bv2p}\,(a,\ b) \wedge \text{z-set}\,(\mathit{set\text{-}flags}))\\
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-move})))\\
\qquad = \quad (\text{v-to-nat}\,(a) \neq \mathtt{0}))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-move})))\\
\qquad\qquad = \quad (\text{v-to-nat}\,(a) = \mathtt{0}))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-m15})))\\
\qquad\qquad = \quad (\text{v-to-nat}\,(a) \neq \mathtt{0}))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},
\end{array}
$$

v-alu (c-flag (*flags*), a, b, OP-M15)))
$=$ (v-to-nat (a) $=$ 0)))

THEOREM: flags-interpretation-nat-inc
(bv2p (a, b) $\land$ c-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-CC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
$=$ ((1 $+$ v-to-nat (a)) $<$ exp (2, length (a))))
$\land$ (store-resultp (CC-CS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
$=$ ((1 $+$ v-to-nat (a)) $\not<$ exp (2, length (a)))))

THEOREM: flags-interpretation-nat-addc
(bv2p (a, b) $\land$ c-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-CC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ADDC)))
$=$ ((b-to-nat (c-flag (*flags*)) $+$ v-to-nat (a) $+$ v-to-nat (b))
$<$ exp (2, length (a))))
$\land$ (store-resultp (CC-CS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ADDC)))
$=$ ((b-to-nat (c-flag (*flags*))
$+$ v-to-nat (a)
$+$ v-to-nat (b))
$\not<$ exp (2, length (a)))))

THEOREM: flags-interpretation-nat-add
(bv2p (a, b) $\land$ c-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-CC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ADD)))
$=$ ((v-to-nat (a) $+$ v-to-nat (b)) $<$ exp (2, length (a))))
$\land$ (store-resultp (CC-CS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ADD)))
$=$ ((v-to-nat (a) $+$ v-to-nat (b)) $\not<$ exp (2, length (a)))))

THEOREM: when-the-mantissa-isnt-0-then-neither-is-the-exponential
$(mantissa \not\simeq 0) \rightarrow (\exp(mantissa,\ exponent) \neq 0)$

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
$((n < m)$
$\land \quad (n \not\simeq 0)$
$\land \quad (m \not\simeq 0)$
$\land \quad (l \not\simeq 0)$
$\land \quad ((n \ \mathbf{mod}\ l) = 0)$
$\land \quad ((m \ \mathbf{mod}\ l) = 0))$
$\rightarrow \quad ((n \div l) < (m \div l))$

THEOREM: v-nzerop-nat-to-v$help
$((l \not\simeq 0) \land (n \not\simeq 0) \land ((n \ \mathbf{mod}\ 2) = 0) \land (n < \exp(2,\ l)))$
$\rightarrow \quad ((n \div 2) < \exp(2,\ l - 1))$

EVENT: Enable v-nzerop-nat-to-v$help; name this event 'v-nzerop-nat-to-v$help-off'.

THEOREM: v-nzerop-nat-to-v
$(n < \exp(2,\ l)) \rightarrow (\text{v-nzerop}(\text{nat-to-v}(n,\ l)) = (n \not\simeq 0))$

THEOREM: lessp-sub1-plus-a-b-b
$(((a + b) - 1) < b) = ((a \simeq 0) \land (b \not\simeq 0))$

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
bv2p $(a,\ b)$
$\rightarrow$ ((c-set (*set-flags*)
$\rightarrow$ ((store-resultp (CC-CC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-DEC)))

$$
\begin{array}{l}
\quad = \quad (\text{v-to-nat}\,(a) \neq \mathtt{0}))\\
\quad \land \quad (\text{store-resultp}\,(\textsc{cc-cs},\\
\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad \textsc{op-dec})))\\
\qquad = \quad (\text{v-to-nat}\,(a) = \mathtt{0}))))\\
\land \quad ((\text{z-set}\,(\mathit{set\text{-}flags}) \land (\text{length}\,(a) \neq \mathtt{0}))\\
\quad \rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad \textsc{op-dec})))\\
\qquad = \quad (\text{v-to-nat}\,(a) \neq \mathtt{1}))\\
\qquad \land \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\quad \textsc{op-dec})))\\
\qquad\quad = \quad (\text{v-to-nat}\,(a) = \mathtt{1})))))
\end{array}
$$

Theorem: remainder-theorem-for-subb

$((a < b) \land (b < c)) \rightarrow ((((b + c) - a) \ \mathbf{mod}\ c) = (b - a))$

Theorem: difference-linear

$(a < b) \rightarrow (((a - c) < b) = \mathbf{t})$

Theorem: flags-interpretation-nat-subb-carry

$$
\begin{array}{l}
(\text{bv2p}\,(a,\ b) \land \text{c-set}\,(\mathit{set\text{-}flags}))\\
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-cc},\\
\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-subb})))\\
\quad = \quad (\text{v-to-nat}\,(b) \not< (\text{v-to-nat}\,(a) + \text{b-to-nat}\,(\text{c-flag}\,(\mathit{flags})))))\\
\land \quad (\text{store-resultp}\,(\textsc{cc-cs},\\
\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-subb})))\\
\quad = \quad (\text{v-to-nat}\,(b)
\end{array}
$$

$<$ (v-to-nat (a) + b-to-nat (c-flag (*flags*))))))

THEOREM: flags-interpretation-nat-subb-higher

(bv2p $(a,\ b)$ $\wedge$ c-set (*set-flags*) $\wedge$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-HI,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
= ((v-to-nat (a) + b-to-nat (c-flag (*flags*))) $<$ v-to-nat (b)))
$\wedge$ (store-resultp (CC-LS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
= ((v-to-nat (a) + b-to-nat (c-flag (*flags*)))
$\not<$ v-to-nat (b))))

THEOREM: carry-flag-nat-interpretation-for-sub

(bv2p $(a,\ b)$ $\wedge$ c-set (*set-flags*))
$\rightarrow$ (c-flag (update-flags (*flags*, *set-flags*, v-alu (c-flag (*flags*), a, b, OP-SUB)))
= (v-to-nat (b) $<$ v-to-nat (a)))

THEOREM: flags-interpretation-nat-sub

bv2p $(a,\ b)$
$\rightarrow$ ((c-set (*set-flags*)
$\rightarrow$ ((store-resultp (CC-CC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= (v-to-nat (b) $\not<$ v-to-nat (a)))
$\wedge$ (store-resultp (CC-CS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-SUB)))
= (v-to-nat (b) $<$ v-to-nat (a)))))
$\wedge$ ((c-set (*set-flags*) $\wedge$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-HI,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-SUB)))

$$
\begin{aligned}
&= \quad (\text{v-to-nat}(a) < \text{v-to-nat}(b)))\\
&\land \quad (\text{store-resultp}(\textsc{cc-ls}, \text{update-flags}(\mathit{flags}, \mathit{set\text{-}flags}, \text{v-alu}(\text{c-flag}(\mathit{flags}), a, b, \textsc{op-sub})))\\
&\qquad = \quad (\text{v-to-nat}(a) \not< \text{v-to-nat}(b))))))
\end{aligned}
$$

THEOREM: flags-interpretation-nat-lsr-carry

$$
\begin{aligned}
&(\text{bv2p}(a,\ b) \land (\text{length}(a) \not\simeq 0) \land \text{c-set}(\mathit{set\text{-}flags}))\\
&\to \quad ((\text{store-resultp}(\textsc{cc-cc}, \text{update-flags}(\mathit{flags}, \mathit{set\text{-}flags}, \text{v-alu}(\text{c-flag}(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
&\qquad = \quad ((\text{v-to-nat}(a)\ \mathbf{mod}\ 2) = 0))\\
&\qquad \land \quad (\text{store-resultp}(\textsc{cc-cs}, \text{update-flags}(\mathit{flags}, \mathit{set\text{-}flags}, \text{v-alu}(\text{c-flag}(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
&\qquad\qquad = \quad ((\text{v-to-nat}(a)\ \mathbf{mod}\ 2) \neq 0)))
\end{aligned}
$$

THEOREM: flags-interpretation-nat-lsr-zero

$$
\begin{aligned}
&(\text{bv2p}(a,\ b) \land (\text{length}(a) \not\simeq 0) \land \text{z-set}(\mathit{set\text{-}flags}))\\
&\to \quad ((\text{store-resultp}(\textsc{cc-ne}, \text{update-flags}(\mathit{flags}, \mathit{set\text{-}flags}, \text{v-alu}(\text{c-flag}(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
&\qquad = \quad ((\text{v-to-nat}(a) \div 2) \neq 0))\\
&\qquad \land \quad (\text{store-resultp}(\textsc{cc-eq}, \text{update-flags}(\mathit{flags}, \mathit{set\text{-}flags}, \text{v-alu}(\text{c-flag}(\mathit{flags}),\ a,\ b,\ \textsc{op-lsr})))\\
&\qquad\qquad = \quad ((\text{v-to-nat}(a) \div 2) = 0)))
\end{aligned}
$$

THEOREM: idifference-0

$$(\text{integerp}(a) \land \text{integerp}(b)) \to ((\text{idifference}(a,\ b) = 0) = (a = b))$$

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
$\text{bvp}(x) \rightarrow (\text{v-negp}(x) = (\text{v-to-nat}(x) \not< \exp(2, \text{length}(x) - 1)))$

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
$(1 < (x + x)) = (x \not\simeq 0)$

THEOREM: integer-in-rangep-1
$((l \not\simeq 0) \wedge (l \neq 1)) \rightarrow \text{integer-in-rangep}(1, l)$

THEOREM: lessp-a-plus-a-a
$(a < (a + a)) = (a \not\simeq 0)$

THEOREM: exp-linear-bounds
$(\textbf{if } v < \exp(2,\ n-1) \textbf{ then f}$
$\textbf{else t endif}$
$\land \quad (v < \exp(2,\ n)))$
$\rightarrow \quad (\exp(2,\ n-1) \not< (\exp(2,\ n) - v))$

THEOREM: integer-in-rangep-v-to-int
$(l = \text{length}(a)) \rightarrow \text{integer-in-rangep}(\text{v-to-int}(a),\ l)$

THEOREM: addition-cases-for-integer-in-rangep-lemmas
$(\text{integer-in-rangep}(a,\ l) \land \text{integer-in-rangep}(b,\ l) \land (l \not\simeq 0))$
$\rightarrow \quad (\text{iplus}(a,\ b) < \exp(2,\ l))$

THEOREM: subtraction-cases-for-integer-in-rangep-lemmas
$(\text{integer-in-rangep}(a,\ l) \land \text{integer-in-rangep}(b,\ l) \land (l \not\simeq 0))$
$\rightarrow \quad (\text{iplus}(b,\ \text{ineg}(a)) < \exp(2,\ l))$

THEOREM: exp-2-l-1<=exp-2-l
$(l \not\simeq 0) \rightarrow (\exp(2,\ l-1) < \exp(2,\ l))$

THEOREM: lessp-remainder-theorem
$(n < m) \rightarrow ((n \textbf{ mod } r) < m)$

THEOREM: integer-in-rangep-the-obvious-way
$((l \not\simeq 0) \land \text{integerp}(i) \land \text{integer-in-rangep}(i,\ l))$
$\rightarrow \quad (\text{v-negp}(\text{int-to-v}(i,\ l)) = \text{negativep}(i))$

EVENT: Enable integer-in-rangep-the-obvious-way; name this event ‘integer-in-rangep-the-obvious-way-off’.

THEOREM: integer-in-rangep-iplus
$(\text{integer-in-rangep}(a,\ l) \land \text{integer-in-rangep}(b,\ l) \land (l \not\simeq 0))$
$\rightarrow \quad (\text{integer-in-rangep}(\text{iplus}(a,\ b),\ l)$
$\qquad = \quad (\text{v-negp}(\text{int-to-v}(\text{iplus}(a,\ b),\ l)) = \text{negativep}(\text{iplus}(a,\ b))))$

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

THEOREM: integer-in-rangep-iplus\$commuted
$(\text{integer-in-rangep}(a,\ l) \land \text{integer-in-rangep}(b,\ l) \land (l \not\simeq 0))$
$\rightarrow \quad (\text{negativep}(\text{iplus}(a,\ b))$
$\qquad = \quad (\text{integer-in-rangep}(\text{iplus}(a,\ b),\ l)$
$\qquad\qquad \leftrightarrow \quad \text{v-negp}(\text{int-to-v}(\text{iplus}(a,\ b),\ l))))$

THEOREM: integer-in-rangep-iplus-carry
$(\text{integer-in-rangep}\,(a,\ l) \land \text{integer-in-rangep}\,(b,\ l) \land (l \not\simeq \mathtt{0}))$
$\rightarrow\ \ (\text{integer-in-rangep}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)),\ l)$
$\quad = \ \ (\text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)),\ l))$
$\qquad = \ \ \text{negativep}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)))))$

EVENT: Enable integer-in-rangep-iplus-carry; name this event 'integer-in-rangep-iplus-carry-off'.

THEOREM: integer-in-rangep-iplus-carry$commuted
$(\text{integer-in-rangep}\,(a,\ l) \land \text{integer-in-rangep}\,(b,\ l) \land (l \not\simeq \mathtt{0}))$
$\rightarrow\ \ (\text{negativep}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)))$
$\quad = \ \ (\text{integer-in-rangep}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)),\ l)$
$\qquad \leftrightarrow \ \ \text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(\mathtt{1},\ \text{iplus}\,(a,\ b)),\ l))))$

THEOREM: not-integerp-minus-zero
$\text{integerp}\,(-\ x) = (x \not\simeq \mathtt{0})$

THEOREM: integer-in-rangep-iplus-ineg
$(\text{integer-in-rangep}\,(a,\ l) \land \text{integer-in-rangep}\,(b,\ l) \land (l \not\simeq \mathtt{0}))$
$\rightarrow\ \ (\text{integer-in-rangep}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)),\ l)$
$\quad = \ \ (\text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)),\ l))$
$\qquad = \ \ \text{negativep}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)))))$

EVENT: Enable integer-in-rangep-iplus-ineg; name this event 'integer-in-rangep-iplus-ineg-off'.

THEOREM: integer-in-rangep-iplus-ineg$commuted
$(\text{integer-in-rangep}\,(a,\ l) \land \text{integer-in-rangep}\,(b,\ l) \land (l \not\simeq \mathtt{0}))$
$\rightarrow\ \ (\text{negativep}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)))$
$\quad = \ \ (\text{integer-in-rangep}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)),\ l)$
$\qquad \leftrightarrow \ \ \text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(b,\ \text{ineg}\,(a)),\ l))))$

THEOREM: integer-in-rangep-iplus-ineg-carry
$(\text{integer-in-rangep}\,(a,\ l)$
$\land\ \ \text{integer-in-rangep}\,(b,\ l)$
$\land\ \ (l \not\simeq \mathtt{0})$
$\land\ \ (l \neq \mathtt{1}))$
$\rightarrow\ \ (\text{integer-in-rangep}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\mathtt{1},\ a))),\ l)$
$\quad = \ \ (\text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\mathtt{1},\ a))),\ l))$
$\qquad = \ \ \text{negativep}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\mathtt{1},\ a))))))$

EVENT: Enable integer-in-rangep-iplus-ineg-carry; name this event 'integer-in-rangep-iplus-ineg-carry-off'.

THEOREM: integer-in-rangep-iplus-ineg-carry$commuted

$$
\begin{array}{l}
(\text{integer-in-rangep}\,(a,\ l) \\
\ \ \land\ \ \text{integer-in-rangep}\,(b,\ l) \\
\ \ \land\ \ (l \not\simeq \texttt{0}) \\
\ \ \land\ \ (l \neq \texttt{1})) \\
\to\ \ (\text{negativep}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\texttt{1},\ a)))) \\
\qquad = \ \ (\text{v-negp}\,(\text{int-to-v}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\texttt{1},\ a))),\ l)) \\
\qquad\qquad \leftrightarrow\ \ \text{integer-in-rangep}\,(\text{iplus}\,(b,\ \text{ineg}\,(\text{iplus}\,(\texttt{1},\ a))),\ l)))
\end{array}
$$

THEOREM: lessp-1-exp-2

$(l \not\simeq \texttt{0}) \to (\texttt{1} < \exp(\texttt{2},\ l))$

THEOREM: v-nzerop-int-to-v

$$
\begin{array}{l}
(\text{integer-in-rangep}\,(n,\ l) \land (l \not\simeq \texttt{0})) \\
\to\ \ (\text{v-nzerop}\,(\text{int-to-v}\,(n,\ l)) = (\neg\ \text{izerop}\,(n)))
\end{array}
$$

THEOREM: izerop-inc-v-nzerop

$$
\begin{array}{l}
(\text{integer-in-rangep}\,(a,\ l) \land (l \not\simeq \texttt{0})) \\
\to\ \ (\text{v-nzerop}\,(\text{int-to-v}\,(\text{iplus}\,(\texttt{1},\ a),\ l)) = (\neg\ \text{izerop}\,(\text{iplus}\,(\texttt{1},\ a))))
\end{array}
$$

THEOREM: v-nzerop-int-to-v-0

$\text{izerop}\,(z) \to (\neg\ \text{v-nzerop}\,(\text{int-to-v}\,(z,\ l)))$

THEOREM: integer-in-rangep-0

$\text{izerop}\,(a) \to \text{integer-in-rangep}\,(a,\ l)$

THEOREM: not-v-negp-nat-to-v-0

$(a \simeq \texttt{0}) \to (\neg\ \text{v-negp}\,(\text{nat-to-v}\,(a,\ l)))$

THEOREM: length-int-to-v

$\text{length}\,(\text{int-to-v}\,(i,\ l)) = \text{fix}\,(l)$

THEOREM: integer-in-rangep–1

$(l \not\simeq \texttt{0}) \to \text{integer-in-rangep}\,(\texttt{-1},\ l)$

THEOREM: flags-interpretation-int-move

$$
\begin{array}{l}
\text{bvp}\,(a) \\
\to\ \ ((\text{v-set}\,(\mathit{set\text{-}flags}) \\
\qquad \to\ \ ((\text{store-resultp}\,(\textsc{cc-vc}, \\
\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags}, \\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-move}))) \\
\qquad\qquad = \ \ \text{integer-in-rangep}\,(\text{v-to-int}\,(a),\ \text{length}\,(a))) \\
\qquad\quad \land\ \ (\text{store-resultp}\,(\textsc{cc-vc}, \\
\qquad\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags}, \\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},
\end{array}
$$

$$
\begin{array}{l}
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\qquad\quad \textsc{op-m15})))\\
\qquad\quad = \quad \text{integer-in-rangep}\,(\text{v-to-int}\,(a),\ \text{length}\,(a)))\\
\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-vs},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\qquad\quad \textsc{op-move})))\\
\qquad\quad = \quad (\neg\ \text{integer-in-rangep}\,(\text{v-to-int}\,(a),\ \text{length}\,(a))))\\
\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-vs},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\qquad\quad \textsc{op-m15})))\\
\qquad\quad = \quad (\neg\ \text{integer-in-rangep}\,(\text{v-to-int}\,(a),\ \text{length}\,(a))))))\\
\wedge \quad (\text{z-set}\,(\mathit{set\text{-}flags})\\
\quad \rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\quad a,\\
\qquad\qquad\qquad\qquad\qquad\quad b,\\
\qquad\qquad\qquad\qquad\qquad\quad \textsc{op-move})))\\
\qquad\quad = \quad (\text{v-to-int}\,(a) \neq \mathtt{0}))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-eq},\\
\qquad\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad\qquad \textsc{op-move})))\\
\qquad\qquad = \quad (\text{v-to-int}\,(a) = \mathtt{0}))\\
\qquad \wedge \quad (\text{store-resultp}\,(\textsc{cc-ne},\\
\qquad\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad\qquad b,
\end{array}
$$

OP-M15)))
= (v-to-int (a) $\neq$ 0))
$\wedge$ (store-resultp (CC-EQ,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-M15)))
= izerop (v-to-int (a)))))
$\wedge$ (n-set (*set-flags*)
$\rightarrow$ ((store-resultp (CC-PL,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-MOVE)))
= ($\neg$ negativep (v-to-int (a))))
$\wedge$ (store-resultp (CC-MI,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-MOVE)))
= negativep (v-to-int (a)))
$\wedge$ (store-resultp (CC-PL,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-M15)))
= ($\neg$ negativep (v-to-int (a))))
$\wedge$ (store-resultp (CC-MI,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-M15)))
= negativep (v-to-int (a)))))
$\wedge$ ((v-set (*set-flags*) $\wedge$ n-set (*set-flags*))

$$
\begin{array}{l}
\quad\rightarrow\ ((\text{store-resultp}\,(\textsc{cc-ge},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-move})))\\
\qquad\quad = \ (\neg\ \text{negativep}\,(\text{v-to-int}\,(a))))\\
\qquad\land\ (\text{store-resultp}\,(\textsc{cc-lt},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-move})))\\
\qquad\qquad = \ \text{negativep}\,(\text{v-to-int}\,(a)))\\
\qquad\land\ (\text{store-resultp}\,(\textsc{cc-ge},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-m}15)))\\
\qquad\qquad = \ (\neg\ \text{negativep}\,(\text{v-to-int}\,(a))))\\
\qquad\land\ (\text{store-resultp}\,(\textsc{cc-lt},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-m}15)))\\
\qquad\qquad = \ \text{negativep}\,(\text{v-to-int}\,(a)))))\\
\land\ ((\text{v-set}\,(\mathit{set\text{-}flags}) \land \text{n-set}\,(\mathit{set\text{-}flags}) \land \text{z-set}\,(\mathit{set\text{-}flags}))\\
\quad\rightarrow\ ((\text{store-resultp}\,(\textsc{cc-gt},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\\
\qquad\qquad\qquad\qquad\qquad a,\\
\qquad\qquad\qquad\qquad\qquad b,\\
\qquad\qquad\qquad\qquad\qquad \textsc{op-move})))\\
\qquad\quad = \ \text{ilessp}\,(\mathtt{0},\ \text{v-to-int}\,(a)))\\
\qquad\land\ (\text{store-resultp}\,(\textsc{cc-le},\\
\qquad\qquad\qquad\text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\mathit{set\text{-}flags},
\end{array}
$$

```
                                       v-alu (c-flag (flags),
                                              a,
                                              b,
                                              OP-MOVE)))
                = ileq (v-to-int (a), 0))
            ∧ (store-resultp (CC-GT,
                              update-flags (flags,
                                            set-flags,
                                            v-alu (c-flag (flags),
                                                   a,
                                                   b,
                                                   OP-M15)))
                = ilessp (0, v-to-int (a)))
            ∧ (store-resultp (CC-LE,
                              update-flags (flags,
                                            set-flags,
                                            v-alu (c-flag (flags),
                                                   a,
                                                   b,
                                                   OP-M15)))
                = ileq (v-to-int (a), 0)))))
```

THEOREM: flags-interpretation-int-inc-overflow

```
(bvp (a) ∧ (length (a) ≄ 0) ∧ v-set (set-flags))
→ ((store-resultp (CC-VC,
                   update-flags (flags,
                                 set-flags,
                                 v-alu (c-flag (flags), a, b, OP-INC)))
    = integer-in-rangep (iplus (v-to-int (a), 1), length (a)))
   ∧ (store-resultp (CC-VS,
                     update-flags (flags,
                                   set-flags,
                                   v-alu (c-flag (flags), a, b, OP-INC)))
      = (¬ integer-in-rangep (iplus (v-to-int (a), 1),
                              length (a)))))
```

THEOREM: flags-interpretation-int-inc-zero

```
(bvp (a) ∧ (length (a) ≠ 0) ∧ z-set (set-flags))
→ ((store-resultp (CC-NE,
                   update-flags (flags,
                                 set-flags,
                                 v-alu (c-flag (flags), a, b, OP-INC)))
    = (iplus (v-to-int (a), 1) ≠ 0))
   ∧ (store-resultp (CC-EQ,
```

update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
= (iplus (v-to-int (a), 1) = 0)))

THEOREM: flags-interpretation-int-inc-negative
(bvp (a)
∧ (length (a) $\neq$ 0)
∧ (length (a) $\neq$ 1)
∧ v-set (*set-flags*)
∧ n-set (*set-flags*))
→ ((store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
= ($\neg$ negativep (iplus (v-to-int (a), 1))))
∧ (store-resultp (CC-LT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
= negativep (iplus (v-to-int (a), 1))))

THEOREM: flags-interpretation-int-inc-gt-0
(bvp (a)
∧ (length (a) $\neq$ 0)
∧ (length (a) $\neq$ 1)
∧ n-set (*set-flags*)
∧ v-set (*set-flags*)
∧ z-set (*set-flags*))
→ ((store-resultp (CC-GT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
= ilessp (0, iplus (1, v-to-int (a))))
∧ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-INC)))
= ileq (iplus (1, v-to-int (a)), 0)))

THEOREM: flags-interpretation-int-addc-overflow
(bv2p (a, b) ∧ (length (a) $\not\simeq$ 0) ∧ v-set (*set-flags*))
→ ((store-resultp (CC-VC,
update-flags (*flags*,
set-flags,

```
                          v-alu (c-flag (flags), a, b, OP-ADDC)))
    =  integer-in-rangep (iplus (b-to-nat (c-flag (flags)),
                                 iplus (v-to-int (a), v-to-int (b))),
                          length (a)))
   ∧  (store-resultp (CC-VS,
                      update-flags (flags,
                                    set-flags,
                                    v-alu (c-flag (flags), a, b, OP-ADDC)))
       =  (¬ integer-in-rangep (iplus (b-to-nat (c-flag (flags)),
                                       iplus (v-to-int (a),
                                              v-to-int (b))),
                                length (a)))))
```

THEOREM: flags-interpretation-int-addc-negative

```
 (bv2p (a, b)
  ∧  (length (a) ≠ 0)
  ∧  (length (a) ≠ 1)
  ∧  v-set (set-flags)
  ∧  n-set (set-flags))
 →  ((store-resultp (CC-LT,
                     update-flags (flags,
                                   set-flags,
                                   v-alu (c-flag (flags), a, b, OP-ADDC)))
      =  negativep (iplus (b-to-nat (c-flag (flags)),
                           iplus (v-to-int (a), v-to-int (b)))))
     ∧  (store-resultp (CC-GE,
                        update-flags (flags,
                                      set-flags,
                                      v-alu (c-flag (flags), a, b, OP-ADDC)))
         =  (¬ negativep (iplus (b-to-nat (c-flag (flags)),
                                 iplus (v-to-int (a), v-to-int (b)))))))
```

THEOREM: flags-interpretation-int-addc-gt-0

```
 (bv2p (a, b)
  ∧  (length (a) ≠ 0)
  ∧  (length (a) ≠ 1)
  ∧  n-set (set-flags)
  ∧  v-set (set-flags)
  ∧  z-set (set-flags))
 →  ((store-resultp (CC-GT,
                     update-flags (flags,
                                   set-flags,
                                   v-alu (c-flag (flags), a, b, OP-ADDC)))
      =  ilessp (0,
```

iplus (b-to-nat (c-flag (*flags*)),
iplus (v-to-int (*a*), v-to-int (*b*)))))
∧ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ADDC)))
= ileq (iplus (b-to-nat (c-flag (*flags*)),
iplus (v-to-int (*a*), v-to-int (*b*))),
0)))

THEOREM: flags-interpretation-int-add-overflow
(bv2p (*a*, *b*) ∧ (length (*a*) ≠ 0) ∧ v-set (*set-flags*))
→ ((store-resultp (CC-VC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ADD)))
= integer-in-rangep (iplus (v-to-int (*a*), v-to-int (*b*)), length (*a*)))
∧ (store-resultp (CC-VS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ADD)))
= (¬ integer-in-rangep (iplus (v-to-int (*a*), v-to-int (*b*)),
length (*a*)))))

THEOREM: flags-interpretation-int-add-negative
(bv2p (*a*, *b*) ∧ (length (*a*) ≠ 0) ∧ v-set (*set-flags*) ∧ n-set (*set-flags*))
→ ((store-resultp (CC-LT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ADD)))
= negativep (iplus (v-to-int (*a*), v-to-int (*b*))))
∧ (store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ADD)))
= (¬ negativep (iplus (v-to-int (*a*), v-to-int (*b*))))))

THEOREM: flags-interpretation-int-add-gt-0
(bv2p (*a*, *b*)
∧ (length (*a*) ≠ 0)
∧ (length (*a*) ≠ 1)
∧ n-set (*set-flags*)
∧ v-set (*set-flags*)
∧ z-set (*set-flags*))
→ ((store-resultp (CC-GT,

$$
\begin{array}{l}
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-add})))\\
\quad = \quad \text{ilessp}\,(\mathtt{0},\ \text{iplus}\,(\text{v-to-int}\,(a),\ \text{v-to-int}\,(b))))\\
\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-le},\\
\qquad\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-add})))\\
\qquad = \quad \text{ileq}\,(\text{iplus}\,(\text{v-to-int}\,(a),\ \text{v-to-int}\,(b)),\ \mathtt{0})))
\end{array}
$$

Theorem: int-neg-range-implications

$$
\begin{array}{l}
((\text{negativep}\,(\text{v-to-int}\,(a))\\
\ \wedge \quad (\neg\ \text{integer-in-rangep}\,(\text{negative-guts}\,(\text{v-to-int}\,(a)),\ \text{length}\,(a))))\\
\ \to \quad \text{v-negp}\,(\text{int-to-v}\,(\text{negative-guts}\,(\text{v-to-int}\,(a)),\ \text{length}\,(a))))\\
\wedge \quad (((\neg\ \text{negativep}\,(\text{v-to-int}\,(a)))\\
\qquad \wedge \quad (\neg\ \text{integer-in-rangep}\,(-\ \text{v-to-int}\,(a),\ \text{length}\,(a))))\\
\qquad \to \quad (\neg\ \text{v-negp}\,(\text{int-to-v}\,(-\ \text{v-to-int}\,(a),\ \text{length}\,(a)))))\\
\wedge \quad (((\neg\ \text{negativep}\,(\text{v-to-int}\,(a)))\\
\qquad \wedge \quad (\neg\ \text{integer-in-rangep}\,(-\ \text{v-to-int}\,(a),\ \text{length}\,(a))))\\
\qquad \to \quad \text{v-nzerop}\,(\text{int-to-v}\,(-\ \text{v-to-int}\,(a),\ \text{length}\,(a))))
\end{array}
$$

Theorem: flags-interpretation-int-neg-overflow

$$
\begin{array}{l}
(\text{bvp}\,(a)\ \wedge\ (\text{length}\,(a)\ \not\simeq\ \mathtt{0})\ \wedge\ \text{v-set}\,(\mathit{set\text{-}flags}))\\
\to \quad ((\text{store-resultp}\,(\textsc{cc-vc},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-neg})))\\
\quad = \quad \text{integer-in-rangep}\,(\text{ineg}\,(\text{v-to-int}\,(a)),\ \text{length}\,(a)))\\
\quad \wedge \quad (\text{store-resultp}\,(\textsc{cc-vs},\\
\qquad\qquad\qquad\quad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad\quad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-neg})))\\
\qquad = \quad (\neg\ \text{integer-in-rangep}\,(\text{ineg}\,(\text{v-to-int}\,(a)),\ \text{length}\,(a)))))
\end{array}
$$

Theorem: flags-interpretation-int-neg-negative

$$
\begin{array}{l}
(\text{bvp}\,(a)\\
\ \wedge \quad (\text{length}\,(a)\ \neq\ \mathtt{0})\\
\ \wedge \quad (\text{length}\,(a)\ \neq\ \mathtt{1})\\
\ \wedge \quad \text{v-set}\,(\mathit{set\text{-}flags})\\
\ \wedge \quad \text{n-set}\,(\mathit{set\text{-}flags}))\\
\to \quad ((\text{store-resultp}\,(\textsc{cc-lt},\\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags},\\
\qquad\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},\\
\qquad\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-neg})))\\
\quad = \quad \text{negativep}\,(\text{ineg}\,(\text{v-to-int}\,(a))))
\end{array}
$$

$\land$ (store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-NEG)))
= ($\neg$ negativep (ineg (v-to-int (a))))))

THEOREM: flags-interpretation-int-neg-zero
(bvp (a) $\land$ (length (a) $\neq$ 0) $\land$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-NE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-NEG)))
= (ineg (v-to-int (a)) $\neq$ 0))
$\land$ (store-resultp (CC-EQ,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-NEG)))
= (ineg (v-to-int (a)) = 0)))

THEOREM: flags-interpretation-int-neg-gt-0
(bvp (a)
$\land$ (length (a) $\neq$ 0)
$\land$ (length (a) $\neq$ 1)
$\land$ n-set (*set-flags*)
$\land$ v-set (*set-flags*)
$\land$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-GT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-NEG)))
= ilessp (0, ineg (v-to-int (a))))
$\land$ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-NEG)))
= ileq (ineg (v-to-int (a)), 0)))

THEOREM: int-dec-rangep-lemmas
(($\neg$ integer-in-rangep (v-to-int (a) $-$ 1, length (a)))
$\rightarrow$ v-nzerop (int-to-v (v-to-int (a) $-$ 1, length (a))))
$\land$ (((length (a) $\neq$ 1)
$\land$ ($\neg$ integer-in-rangep ($-$ (1 + negative-guts (v-to-int (a))),
length (a))))
$\rightarrow$ v-nzerop (int-to-v ($-$ (1 + negative-guts (v-to-int (a))),
length (a))))

$\land$ $(((\neg$ negativep (iplus (`-1`, v-to-int (a)))))
  $\land$ v-nzerop (int-to-v (iplus (`-1`, v-to-int (a)), length (a))))
  $\rightarrow$ (iplus (`-1`, v-to-int (a)) $\neq$ `0`))
$\land$ (((length (a) $\not\simeq$ `0`)
  $\land$ ($\neg$ negativep (iplus (`-1`, v-to-int (a))))
  $\land$ ($\neg$ v-nzerop (int-to-v (iplus (`-1`, v-to-int (a)), length (a)))))
  $\rightarrow$ (iplus (`-1`, v-to-int (a)) = `0`))

Theorem: flags-interpretation-int-dec-overflow
(bvp (a) $\land$ (length (a) $\not\simeq$ `0`) $\land$ v-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-VC,
    update-flags (*flags*,
      set-flags,
      v-alu (c-flag (*flags*), a, b, OP-DEC)))
  = integer-in-rangep (idifference (v-to-int (a), `1`), length (a)))
  $\land$ (store-resultp (CC-VS,
    update-flags (*flags*,
      set-flags,
      v-alu (c-flag (*flags*), a, b, OP-DEC)))
    = ($\neg$ integer-in-rangep (idifference (v-to-int (a), `1`),
      length (a))))))

Theorem: flags-interpretation-int-dec-negative
(bvp (a) $\land$ (length (a) $\neq$ `0`) $\land$ v-set (*set-flags*) $\land$ n-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-LT,
    update-flags (*flags*,
      set-flags,
      v-alu (c-flag (*flags*), a, b, OP-DEC)))
  = negativep (idifference (v-to-int (a), `1`)))
  $\land$ (store-resultp (CC-GE,
    update-flags (*flags*,
      set-flags,
      v-alu (c-flag (*flags*), a, b, OP-DEC)))
    = ($\neg$ negativep (idifference (v-to-int (a), `1`)))))

Theorem: flags-interpretation-int-dec-zero
(bvp (a) $\land$ (length (a) $\neq$ `0`) $\land$ (length (a) $\neq$ `1`) $\land$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-NE,
    update-flags (*flags*,
      set-flags,
      v-alu (c-flag (*flags*), a, b, OP-DEC)))
  = (idifference (v-to-int (a), `1`) $\neq$ `0`))
  $\land$ (store-resultp (CC-EQ,
    update-flags (*flags*,
      set-flags,

```
                                        v-alu (c-flag (flags), a, b, OP-DEC)))
              =    (idifference (v-to-int (a), 1) = 0)))
```

THEOREM: flags-interpretation-int-dec-gt-0

```
 (bvp (a)
  ∧    (length (a) ≠ 0)
  ∧    (length (a) ≠ 1)
  ∧    n-set (set-flags)
  ∧    v-set (set-flags)
  ∧    z-set (set-flags))
 →    ((store-resultp (CC-GT,
                       update-flags (flags,
                                     set-flags,
                                     v-alu (c-flag (flags), a, b, OP-DEC)))
          =    ilessp (0, idifference (v-to-int (a), 1)))
          ∧    (store-resultp (CC-LE,
                               update-flags (flags,
                                             set-flags,
                                             v-alu (c-flag (flags), a, b, OP-DEC)))
              =    ileq (idifference (v-to-int (a), 1), 0)))
```

THEOREM: flags-interpretation-int-subb-overflow

```
 (bv2p (a, b) ∧ (length (a) ≄ 0) ∧ v-set (set-flags))
 →    ((store-resultp (CC-VC,
                       update-flags (flags,
                                     set-flags,
                                     v-alu (c-flag (flags), a, b, OP-SUBB)))
          =    integer-in-rangep (idifference (v-to-int (b),
                                               iplus (v-to-int (a),
                                                      b-to-nat (c-flag (flags)))),
                                  length (a)))
          ∧    (store-resultp (CC-VS,
                               update-flags (flags,
                                             set-flags,
                                             v-alu (c-flag (flags), a, b, OP-SUBB)))
              =    (¬ integer-in-rangep (idifference (v-to-int (b),
                                                       iplus (v-to-int (a),
                                                              b-to-nat (c-flag (flags)))),
                                         length (a)))))
```

THEOREM: flags-interpretation-int-subb-negative

```
 (bv2p (a, b)
  ∧    (length (a) ≠ 0)
  ∧    (length (a) ≠ 1)
  ∧    v-set (set-flags)
```

$\wedge$ n-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-LT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
$=$ negativep (idifference (v-to-int (b),
iplus (v-to-int (a),
b-to-nat (c-flag (*flags*))))))
$\wedge$ (store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
$=$ ($\neg$ negativep (idifference (v-to-int (b),
iplus (v-to-int (a),
b-to-nat (c-flag (*flags*))))))))

THEOREM: flags-interpretation-int-subb-gt-0
(bv2p (a, b)
$\wedge$ (length (a) $\neq$ 0)
$\wedge$ (length (a) $\neq$ 1)
$\wedge$ n-set (*set-flags*)
$\wedge$ v-set (*set-flags*)
$\wedge$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-GT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
$=$ ilessp (0,
idifference (v-to-int (b),
iplus (v-to-int (a), b-to-nat (c-flag (*flags*))))))
$\wedge$ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUBB)))
$=$ ileq (idifference (v-to-int (b),
iplus (v-to-int (a),
b-to-nat (c-flag (*flags*)))),
0)))

THEOREM: flags-interpretation-int-sub-overflow
(bv2p (a, b) $\wedge$ (length (a) $\not\simeq$ 0) $\wedge$ v-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-VC,
update-flags (*flags*,
set-flags,

v-alu (c-flag (*flags*), a, b, OP-SUB)))
= integer-in-rangep (idifference (v-to-int (b), v-to-int (a)),
length (a)))
∧ (store-resultp (CC-VS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= (¬ integer-in-rangep (idifference (v-to-int (b),
v-to-int (a)),
length (a)))))

THEOREM: flags-interpretation-int-sub-negative
(bv2p (a, b)
∧ (length (a) $\neq$ 0)
∧ (length (a) $\neq$ 1)
∧ v-set (*set-flags*)
∧ n-set (*set-flags*))
→ ((store-resultp (CC-LT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= negativep (idifference (v-to-int (b), v-to-int (a))))
∧ (store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= (¬ negativep (idifference (v-to-int (b), v-to-int (a))))))

THEOREM: flags-interpretation-int-sub-gt-0
(bv2p (a, b)
∧ (length (a) $\neq$ 0)
∧ (length (a) $\neq$ 1)
∧ n-set (*set-flags*)
∧ v-set (*set-flags*)
∧ z-set (*set-flags*))
→ ((store-resultp (CC-GT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= ilessp (0, idifference (v-to-int (b), v-to-int (a))))
∧ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-SUB)))
= ileq (idifference (v-to-int (b), v-to-int (a)), 0)))

THEOREM: equal-idiv-2-0
$(\text{idiv}\,(i,\ 2) = 0) = (\text{izerop}\,(i) \lor (i = 1))$

THEOREM: integer-in-rangep-idiv
$\text{integer-in-rangep}\,(i,\ l) \rightarrow \text{integer-in-rangep}\,(\text{idiv}\,(i,\ 2),\ l)$

THEOREM: flags-interpretation-int-asr-zero
$$
\begin{array}{l}
(\text{bvp}\,(a) \land (\text{length}\,(a) \not\simeq 0) \land \text{z-set}\,(\mathit{set\text{-}flags})) \\
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-ne}, \\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags}, \\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags}, \\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-asr}))) \\
\qquad = \quad (\text{idiv}\,(\text{v-to-int}\,(a),\ 2) \neq 0)) \\
\qquad \land \quad (\text{store-resultp}\,(\textsc{cc-eq}, \\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags}, \\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags}, \\
\qquad\qquad\qquad\qquad \text{v-alu}\,(\text{c-flag}\,(\mathit{flags}),\ a,\ b,\ \textsc{op-asr}))) \\
\qquad\quad = \quad (\text{idiv}\,(\text{v-to-int}\,(a),\ 2) = 0)))
\end{array}
$$

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

THEOREM: remainder-difference-plus-x-x-plus-y-y
$(((x + x) - (y + y))\ \textbf{mod}\ 2) = 0$

THEOREM: remainder-difference-plus-x-x-add1-plus-y-y
$((1 + (y + y)) < (x + x))$
$\rightarrow \quad ((((x + x) - (1 + (y + y)))\ \textbf{mod}\ 2) = 1)$

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
$$
\begin{array}{l}
(\text{bvp}\,(a) \land (\text{length}\,(a) \not\simeq 0) \land \text{c-set}\,(\mathit{set\text{-}flags})) \\
\rightarrow \quad ((\text{store-resultp}\,(\textsc{cc-cc}, \\
\qquad\qquad\qquad \text{update-flags}\,(\mathit{flags}, \\
\qquad\qquad\qquad\qquad \mathit{set\text{-}flags},
\end{array}
$$

v-alu (c-flag (*flags*), *a*, *b*, OP-ASR)))
= (imod (v-to-int (*a*), 2) = 0))
∧ (store-resultp (CC-CS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ASR)))
= (imod (v-to-int (*a*), 2) ≠ 0)))

THEOREM: flags-interpretation-int-asr-negative
(bvp (*a*) ∧ (length (*a*) $\not\simeq$ 0))
→ ((n-set (*set-flags*)
→ ((store-resultp (CC-PL,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), *a*, *b*, OP-ASR)))
= (¬ negativep (idiv (v-to-int (*a*), 2))))
∧ (store-resultp (CC-MI,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-ASR)))
= negativep (idiv (v-to-int (*a*), 2)))))
∧ ((n-set (*set-flags*) ∧ v-set (*set-flags*))
→ ((store-resultp (CC-GE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-ASR)))
= (¬ negativep (idiv (v-to-int (*a*), 2))))
∧ (store-resultp (CC-LT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*),
a,
b,
OP-ASR)))
= negativep (idiv (v-to-int (*a*), 2))))))

THEOREM: flags-interpretation-int-asr-gt-0
(bvp (*a*)

$\wedge$ (length $(a) \not\simeq 0$)
$\wedge$ v-set (*set-flags*)
$\wedge$ n-set (*set-flags*)
$\wedge$ z-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-GT,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ASR)))
= ilessp (0, idiv (v-to-int (a), 2)))
$\wedge$ (store-resultp (CC-LE,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ASR)))
= ileq (idiv (v-to-int (a), 2), 0)))

THEOREM: flags-interpretation-int-asr-overflow
(bvp (a) $\wedge$ v-set (*set-flags*))
$\rightarrow$ ((store-resultp (CC-VC,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ASR)))
= integer-in-rangep (idiv (v-to-int (a), 2), length (a)))
$\wedge$ (store-resultp (CC-VS,
update-flags (*flags*,
set-flags,
v-alu (c-flag (*flags*), a, b, OP-ASR)))
= ($\neg$ integer-in-rangep (idiv (v-to-int (a), 2), length (a)))))

DEFINITION:
v-neg (a) = v-subtracter-output (**f**, a, nat-to-v (0, length (a)))

THEOREM: bv-cvzbv
bv (cvzbv (c, v, bv)) = bv

THEOREM: c-cvzbv
c (cvzbv (c, v, bv)) = c

THEOREM: v-cvzbv
v (cvzbv (c, v, bv)) = v

THEOREM: zb-cvzbv
zb (cvzbv (c, v, bv)) = v-zerop (bv)

THEOREM: bv-v-alu
bv (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-subtracter-output (c, a, b)
elseif op = list (**t**, **t**, **t**, **f**) **then** v-subtracter-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)
 $\lor$ (op = OP-XOR)
 $\lor$ (op = OP-OR)
 $\lor$ (op = OP-AND)
 $\lor$ (op = OP-NOT)
 $\lor$ (op = OP-M15))
 $\rightarrow$ (c (v-alu (c, a, b, op)) = **f**))
$\land$ ((listp (a)
 $\land$ ((op = OP-ROR) $\lor$ (op = OP-ASR) $\lor$ (op = OP-LSR)))
 $\rightarrow$ (c (v-alu (c, a, b, op)) = nth (0, a)))

THEOREM: v-v-alu
((op = OP-MOVE)
 $\lor$ (op = OP-ROR)
 $\lor$ (op = OP-ASR)
 $\lor$ (op = OP-ASR)
 $\lor$ (op = OP-LSR)
 $\lor$ (op = OP-XOR)
 $\lor$ (op = OP-OR)
 $\lor$ (op = OP-AND)
 $\lor$ (op = OP-NOT)
 $\lor$ (op = OP-M15))
$\rightarrow$ (v (v-alu (c, a, b, op)) = **f**)

THEOREM: v-alu-correct-nat-rewrite
bv2p (a, b) $\rightarrow$ (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
$(\text{bv2p}(a, b) \land (\text{length}(a) \neq 0))$
$\rightarrow \quad (\text{v-alu}(c, a, b, op) = \text{v-alu-int}(c, a, b, op))$

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
$\text{nat-to-v}(n \ \mathbf{mod}\ \exp(2, l), l) = \text{nat-to-v}(n, l)$

THEOREM: nat-to-v-of-v-to-nat*
$(\text{bvp}(v) \land (l = \text{length}(v))) \rightarrow (\text{nat-to-v}(\text{v-to-nat}(v), l) = v)$

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
$(\text{bv2p}(a, b) \land (\text{length}(a) = 32))$
$\rightarrow \quad (\text{bv}(\text{v-alu}(c, a, b, op))$
$= \quad \mathbf{if}\ op = \text{OP-INC}\ \mathbf{then}\ \text{nat-to-v}(1 + \text{v-to-nat}(a), 32)$
$\mathbf{elseif}\ op = \text{OP-ADDC}$
$\mathbf{then}\ \text{nat-to-v}(\text{b-to-nat}(c) + \text{v-to-nat}(a) + \text{v-to-nat}(b), 32)$
$\mathbf{elseif}\ op = \text{OP-ADD}$
$\mathbf{then}\ \text{nat-to-v}(\text{v-to-nat}(a) + \text{v-to-nat}(b), 32)$
$\mathbf{elseif}\ op = \text{OP-DEC}$
$\mathbf{then}\ \text{nat-to-v}((\text{v-to-nat}(a) + \exp(2, 32)) - 1, 32)$
$\mathbf{elseif}\ op = \text{OP-SUBB}$
$\mathbf{then}\ \text{nat-to-v}((\text{v-to-nat}(b) + \exp(2, 32))$
$\quad - \quad (\text{v-to-nat}(a) + \text{b-to-nat}(c)),$
$\quad 32)$
$\mathbf{elseif}\ op = \text{OP-SUB}$
$\mathbf{then}\ \text{nat-to-v}((\text{v-to-nat}(b) + \exp(2, 32)) - \text{v-to-nat}(a),$
$\quad 32)$
$\mathbf{elseif}\ op = \text{OP-LSR}\ \mathbf{then}\ \text{nat-to-v}(\text{v-to-nat}(a) \div 2, 32)$
$\mathbf{else}\ \text{bv}(\text{v-alu}(c, a, b, op))\ \mathbf{endif})$

THEOREM: fix-int-int-to-v
fix-int (v-to-int (v)) = v-to-int (v)

THEOREM: int-to-v-of-v-to-int
((l = length (v)) ∧ bvp (v)) → (int-to-v (v-to-int (v), l) = v)

THEOREM: ineg-as-idifference
idifference (0, i) = ineg (i)

THEOREM: bv-v-alu-as-integer
(bv2p (a, b) ∧ (length (a) = 32))
→ (bv (v-alu (c, a, b, op))
  = **if** op = OP-INC **then** int-to-v (iplus (1, v-to-int (a)), 32)
  elseif op = OP-ADDC
  then int-to-v (iplus (b-to-nat (c),
      iplus (v-to-int (a), v-to-int (b))),
    32)
  elseif op = OP-ADD
  then int-to-v (iplus (v-to-int (a), v-to-int (b)), 32)
  elseif op = OP-NEG **then** int-to-v (ineg (v-to-int (a)), 32)
  elseif op = OP-DEC
  then int-to-v (idifference (v-to-int (a), 1), 32)
  elseif op = OP-SUBB
  then int-to-v (idifference (v-to-int (b),
      iplus (v-to-int (a), b-to-nat (c))),
    32)
  elseif op = OP-SUB
  then int-to-v (idifference (v-to-int (b), v-to-int (a)), 32)
  elseif op = OP-ASR **then** int-to-v (idiv (v-to-int (a), 2), 32)
  else bv (v-alu (c, a, b, op)) **endif**)

EVENT: Make the library `"fm9001-replay"` and compile it.

Index