

#|

Copyright (C) 1994 by Alex Bronstein and Carolyn Talcott. All Rights Reserved.

You may copy and distribute verbatim copies of this Nqthm-1992 event script as you receive it, in any medium, including embedding it verbatim in derivative works, provided that you conspicuously and appropriately publish on each copy a valid copyright notice "Copyright (C) 1994 by Alex Bronstein and Carolyn Talcott. All Rights Reserved."

NO WARRANTY

Alex Bronstein and Carolyn Talcott PROVIDE ABSOLUTELY NO WARRANTY. THE EVENT SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU. SHOULD THE SCRIPT PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

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

|#

EVENT: Start with the library "mlp" using the compiled version.

```
; ppltcpu.bm is our 1st pipelined CPU (on our way to SRC CPU):  
;  
; it's totally trivial and unrealistic: no jumps, and current values in  
; (visible) register(s) can't be USED in instructions! (i.e. NO LOOPS).  
;
```

```
;;; (Sugared) Circuits:
```

#|

```
(setq sy-A '(SY-A (x)  
(Ypc R 0 Ypcn)  
(Ypcn S inc Ypc)
```

```

(Ypr S Up Ypc)
(Yi S Ui Ypr)
(Ye S Ue Yi)
(Yout R 0 Ye)
; and the cork for Ye:
(Yec1 R (UE (UI 0)) Ye)
(Yec R (UE 0) Yec1)
))

```

```

(setq sy-B '(SY-B (x)
(Ypc R 0 Ypcn)
(Ypcn S inc Ypc)
(Ypr S Up Ypc)
(Ypr2 R 0 Ypr)
(Yi S Ui Ypr2)
(Yi2 R 0 Yi)
(Ye S Ue Yi2)
(Yout R 0 Ye)
))

```

```

(setq ppltcpu '( |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_inc.bm: INCrement combinational element
; U7-DONE

```

DEFINITION: $\text{inc}(u) = (1 + u)$

; Everything below generated by: (bmcomb 'inc '() '(x))

DEFINITION:

```

s-inc(x)
=  if empty(x) then E
   else a(s-inc(p(x)), inc(l(x))) endif

```

; A2-Begin-S-INC

THEOREM: a2-empty-s-inc
 $\text{empty}(\text{s-inc}(x)) = \text{empty}(x)$

THEOREM: a2-e-s-inc
 $(\text{s-inc}(x) = E) = \text{empty}(x)$

THEOREM: a2-lp-s-inc
 $\text{len}(\text{s-inc}(x)) = \text{len}(x)$

THEOREM: a2-lpe-s-inc
 $\text{eqlen}(\text{s-inc}(x), x)$

THEOREM: a2-ic-s-inc
 $\text{s-inc}(i(c_x, x)) = i(\text{inc}(c_x), \text{s-inc}(x))$

THEOREM: a2-lc-s-inc
 $(\neg \text{empty}(x)) \rightarrow (l(\text{s-inc}(x)) = \text{inc}(l(x)))$

THEOREM: a2-pc-s-inc
 $p(\text{s-inc}(x)) = \text{s-inc}(p(x))$

THEOREM: a2-hc-s-inc
 $(\neg \text{empty}(x)) \rightarrow (h(\text{s-inc}(x)) = \text{inc}(h(x)))$

THEOREM: a2-bc-s-inc
 $b(\text{s-inc}(x)) = \text{s-inc}(b(x))$

THEOREM: a2-bnc-s-inc
 $\text{bn}(n, \text{s-inc}(x)) = \text{s-inc}(\text{bn}(n, x))$

;; A2-End-S-INC

; eof:comb_inc.bm

; comb_up.bm: Up combinational element (= fun1)
 ; U7-DONE

; arbitrary Char-Fun of arity 1:

EVENT: Introduce the function symbol *up* of one argument.

; Everything below generated by: (bmcomb 'Up '() '(x))

DEFINITION:

$\text{s-up}(x)$
 = **if** $\text{empty}(x)$ **then** E
 else $a(\text{s-up}(p(x)), \text{up}(l(x)))$ **endif**

;; A2-Begin-S-UP

```

THEOREM: a2-empty-s-up
empty (s-up (x)) = empty (x)

THEOREM: a2-e-s-up
(s-up (x) = E) = empty (x)

THEOREM: a2-lp-s-up
len (s-up (x)) = len (x)

THEOREM: a2-lpe-s-up
eqlen (s-up (x), x)

THEOREM: a2-ic-s-up
s-up (i (c_x, x)) = i (up (c_x), s-up (x))

THEOREM: a2-lc-s-up
(¬ empty (x)) → (l (s-up (x)) = up (l (x)))

THEOREM: a2-pc-s-up
p (s-up (x)) = s-up (p (x))

THEOREM: a2-hc-s-up
(¬ empty (x)) → (h (s-up (x)) = up (h (x)))

THEOREM: a2-bc-s-up
b (s-up (x)) = s-up (b (x))

THEOREM: a2-bnc-s-up
bn (n, s-up (x)) = s-up (bn (n, x))

;; A2-End-S-UP

; eof:comb_up.bm

; comb_ui.bm: Ui combinational element (= fun1)
; U7-DONE

; arbitrary Char-Fun of arity 1:
EVENT: Introduce the function symbol ui of one argument.

; Everything below generated by: (bmcomb 'Ui '() '(x))

DEFINITION:
s-ui (x)
=  if empty (x) then E
   else a (s-ui (p (x)), ui (l (x))) endif

```

```

;; A2-Begin-S-UI

THEOREM: a2-empty-s-ui
empty (s-ui (x)) = empty (x)

THEOREM: a2-e-s-ui
(s-ui (x) = e) = empty (x)

THEOREM: a2-lp-s-ui
len (s-ui (x)) = len (x)

THEOREM: a2-lpe-s-ui
eq len (s-ui (x), x)

THEOREM: a2-ic-s-ui
s-ui (i (c x, x)) = i (ui (c x), s-ui (x))

THEOREM: a2-lc-s-ui
( $\neg$  empty (x))  $\rightarrow$  (l (s-ui (x)) = ui (l (x)))

THEOREM: a2-pc-s-ui
p (s-ui (x)) = s-ui (p (x))

THEOREM: a2-hc-s-ui
( $\neg$  empty (x))  $\rightarrow$  (h (s-ui (x)) = ui (h (x)))

THEOREM: a2-bc-s-ui
b (s-ui (x)) = s-ui (b (x))

THEOREM: a2-bnc-s-ui
bn (n, s-ui (x)) = s-ui (bn (n, x))

;; A2-End-S-UI

; eof:comb_ui.bm

; comb_ue.bm: Ue combinational element (= fun1)
; U7-DONE

; arbitrary Char-Fun of arity 1:

EVENT: Introduce the function symbol ue of one argument.

; Everything below generated by: (bmcomb 'Ue '() '(x))

```

DEFINITION:
 $\text{s-ue}(x)$
 $= \text{if empty}(x) \text{ then } E$
 $\quad \text{else } a(\text{s-ue}(p(x)), \text{ue}(l(x))) \text{ endif}$
 ;; A2-Begin-S-UE

THEOREM: a2-empty-s-ue
 $\text{empty}(\text{s-ue}(x)) = \text{empty}(x)$

THEOREM: a2-e-s-ue
 $(\text{s-ue}(x) = E) = \text{empty}(x)$

THEOREM: a2-lp-s-ue
 $\text{len}(\text{s-ue}(x)) = \text{len}(x)$

THEOREM: a2-lpe-s-ue
 $\text{eqlen}(\text{s-ue}(x), x)$

THEOREM: a2-ic-s-ue
 $\text{s-ue}(i(c_x, x)) = i(\text{ue}(c_x), \text{s-ue}(x))$

THEOREM: a2-lc-s-ue
 $(\neg \text{empty}(x)) \rightarrow (l(\text{s-ue}(x)) = \text{ue}(l(x)))$

THEOREM: a2-pc-s-ue
 $p(\text{s-ue}(x)) = \text{s-ue}(p(x))$

THEOREM: a2-hc-s-ue
 $(\neg \text{empty}(x)) \rightarrow (h(\text{s-ue}(x)) = \text{ue}(h(x)))$

THEOREM: a2-bc-s-ue
 $b(\text{s-ue}(x)) = \text{s-ue}(b(x))$

THEOREM: a2-bnc-s-ue
 $\text{bn}(n, \text{s-ue}(x)) = \text{s-ue}(\text{bn}(n, x))$

;; A2-End-S-UE

; eof:comb_ue.bm

DEFINITION:

```

topor-sy-a(ln)
=  if ln = 'ypc then 0
    elseif ln = 'ypcn then 1
    elseif ln = 'ypr then 1
    elseif ln = 'yi then 2
    elseif ln = 'ye then 3
    elseif ln = 'yout then 0
    elseif ln = 'yec1 then 0
    elseif ln = 'yec then 0
    else 0 endif

```

DEFINITION:

```

sy-a(ln, x)
=  if ln = 'ypc
    then if empty(x) then E
        else i(0, sy-a('ypcn, p(x))) endif
    elseif ln = 'ypcn then s-inc(sy-a('ypc, x))
    elseif ln = 'ypr then s-up(sy-a('ypc, x))
    elseif ln = 'yi then s-ui(sy-a('ypr, x))
    elseif ln = 'ye then s-ue(sy-a('yi, x))
    elseif ln = 'yout
    then if empty(x) then E
        else i(0, sy-a('ye, p(x))) endif
    elseif ln = 'yec1
    then if empty(x) then E
        else i(ue(ui(0)), sy-a('ye, p(x))) endif
    elseif ln = 'yec
    then if empty(x) then E
        else i(ue(0), sy-a('yec1, p(x))) endif
    else sfix(x) endif

```

;; A2-Begin-SY-A

THEOREM: a2-empty-sy-a

empty(sy-a(*ln*, *x*)) = empty(*x*)

THEOREM: a2-e-sy-a

(sy-a(*ln*, *x*) = E) = empty(*x*)

THEOREM: a2-lp-sy-a

len(sy-a(*ln*, *x*)) = len(*x*)

THEOREM: a2-lpe-sy-a

eqlen(sy-a(*ln*, *x*), *x*)

THEOREM: a2-pc-sy-a
 $p(\text{sy-a}(ln, x)) = \text{sy-a}(ln, p(x))$
 ;; A2-End-SY-A

DEFINITION:
 topor-sy-b(ln)
 = if $ln = \text{'ypc}$ then 0
 elseif $ln = \text{'ypcn}$ then 1
 elseif $ln = \text{'ypr}$ then 1
 elseif $ln = \text{'ypr2}$ then 0
 elseif $ln = \text{'yi}$ then 1
 elseif $ln = \text{'yi2}$ then 0
 elseif $ln = \text{'ye}$ then 1
 elseif $ln = \text{'yout}$ then 0
 else 0 endif

DEFINITION:
 sy-b(ln, x)
 = if $ln = \text{'ypc}$
 then if empty(x) then E
 else i(0, sy-b('ypcn, p(x))) endif
 elseif $ln = \text{'ypcn}$ then s-inc(sy-b('ypc, x))
 elseif $ln = \text{'ypr}$ then s-up(sy-b('ypc, x))
 elseif $ln = \text{'ypr2}$
 then if empty(x) then E
 else i(0, sy-b('ypr, p(x))) endif
 elseif $ln = \text{'yi}$ then s-ui(sy-b('ypr2, x))
 elseif $ln = \text{'yi2}$
 then if empty(x) then E
 else i(0, sy-b('yi, p(x))) endif
 elseif $ln = \text{'ye}$ then s-ue(sy-b('yi2, x))
 elseif $ln = \text{'yout}$
 then if empty(x) then E
 else i(0, sy-b('ye, p(x))) endif
 else sfix(x) endif
 ;; A2-Begin-SY-B

THEOREM: a2-empty-sy-b
 $\text{empty}(\text{sy-b}(ln, x)) = \text{empty}(x)$

THEOREM: a2-e-sy-b
 $(\text{sy-b}(ln, x) = E) = \text{empty}(x)$

THEOREM: a2-lp-sy-b
 $\text{len}(\text{sy-b}(ln, x)) = \text{len}(x)$

THEOREM: a2-lpe-sy-b
 $\text{eqlen}(\text{sy-b}(ln, x), x)$

THEOREM: a2-pc-sy-b
 $p(\text{sy-b}(ln, x)) = \text{sy-b}(ln, p(x))$

; ; A2-End-SY-B

```
; SOME ANIMATION:
; (setq x5T (A (A (A (A (e) T) T) T) T) T))
;(sy-A 'Yout x5t)
; (A (A (A (A (E) O)
;      '(UE (UI (UP 0))))
;      '(UE (UI (UP 1))))
;      '(UE (UI (UP 2))))
;      '(UE (UI (UP 3))))
;(sy-A 'Ye x5t)
; (A (A (A (A (E) '(UE (UI (UP 0))))
;      '(UE (UI (UP 1))))
;      '(UE (UI (UP 2))))
;      '(UE (UI (UP 3))))
;      '(UE (UI (UP 4))))
;(sy-B 'Yout x5t)
; (A (A (A (A (E) O)
;      '(UE O))
;      '(UE (UI O)))
;      '(UE (UI (UP 0))))
;      '(UE (UI (UP 1))))
;(sy-B 'Ye x5t)
; (A (A (A (A (E) '(UE O))
;      '(UE (UI O)))
;      '(UE (UI (UP 0))))
;      '(UE (UI (UP 1))))
;      '(UE (UI (UP 2))))
;*
; THIS SHOWS that Yout is not PPL but YE is. Proof of PPL YE w/ cork:
; (UE O) (UE (UI O))
; NOTE: I should really find a way to prove such a thing without going
; back to the circuit and altering def... (w/ DECORK & CORK?)

; This takes care of the PC loop:
```

THEOREM: eq-pc
 $\text{sy-b}('ypc, x) = \text{sy-a}('ypc, x)$

THEOREM: eq-a-b
 $\text{sy-b}('ye, x) = \text{sy-a}('yec, x)$

; EQ-A-B2: this phrasing REMOVES the need for fiddling with the circuit:
; i.e. has the cork explicitly in thm.

THEOREM: eq-a-b2
 $\text{sy-b}('ye, x)$
 $= \text{if empty}(x) \text{ then } E$
 $\quad \text{else } i(\text{ue}(0),$
 $\quad \quad \text{if empty}(p(x)) \text{ then } E$
 $\quad \quad \text{else } i(\text{ue}(\text{ui}(0)), \text{sy-a}('ye, p(p(x)))) \text{ endif) endif}$

; EQ-A-B3: this is a weaker but more legible version of the explicitly
; corked thm.

THEOREM: eq-a-b3
 $(\neg \text{empty}(p(x)))$
 $\rightarrow (\text{sy-b}('ye, x) = i(\text{ue}(0), i(\text{ue}(\text{ui}(0)), \text{sy-a}('ye, p(p(x)))))$

; Leaving the circuit alone AND WITHOUT EXPLICITING the cork, we get:

THEOREM: eq-a-b4
 $(\neg \text{empty}(p(x))) \rightarrow (b(b(\text{sy-b}('ye, x))) = \text{sy-a}('ye, p(p(x))))$

; eof: ppltcpu.bm
;))

Index

a, 2–4, 6
a2-bc-s-inc, 3
a2-bc-s-ue, 6
a2-bc-s-ui, 5
a2-bc-s-up, 4
a2-bnc-s-inc, 3
a2-bnc-s-ue, 6
a2-bnc-s-ui, 5
a2-bnc-s-up, 4
a2-e-s-inc, 2
a2-e-s-ue, 6
a2-e-s-ui, 5
a2-e-s-up, 4
a2-e-sy-a, 7
a2-e-sy-b, 8
a2-empty-s-inc, 2
a2-empty-s-ue, 6
a2-empty-s-ui, 5
a2-empty-s-up, 4
a2-empty-sy-a, 7
a2-empty-sy-b, 8
a2-hc-s-inc, 3
a2-hc-s-ue, 6
a2-hc-s-ui, 5
a2-hc-s-up, 4
a2-ic-s-inc, 3
a2-ic-s-ue, 6
a2-ic-s-ui, 5
a2-ic-s-up, 4
a2-lc-s-inc, 3
a2-lc-s-ue, 6
a2-lc-s-ui, 5
a2-lc-s-up, 4
a2-lp-s-inc, 3
a2-lp-s-ue, 6
a2-lp-s-ui, 5
a2-lp-s-up, 4
a2-lp-sy-a, 7
a2-lp-sy-b, 9
a2-lpe-s-inc, 3
a2-lpe-s-ue, 6
a2-lpe-s-ui, 5
a2-lpe-s-up, 4
a2-lpe-sy-a, 7
a2-lpe-sy-b, 9
a2-pc-s-inc, 3
a2-pc-s-ue, 6
a2-pc-s-ui, 5
a2-pc-s-up, 4
a2-pc-sy-a, 8
a2-pc-sy-b, 9

b, 3–6, 10
bn, 3–6

e, 2–8, 10
empty, 2–8, 10
eq-a-b, 10
eq-a-b2, 10
eq-a-b3, 10
eq-a-b4, 10
eq-pc, 10
eq-len, 3–7, 9

h, 3–6

i, 3–8, 10
inc, 2, 3

l, 2–6
len, 3–7, 9

p, 2–10

s-inc, 2, 3, 7, 8
s-ue, 6–8
s-ui, 4, 5, 7, 8
s-up, 3, 4, 7, 8
sfix, 7, 8
sy-a, 7, 8, 10
sy-b, 8–10

topor-sy-a, 7

topor-sy-b, 8

ue, 5–7, 10

ui, 4, 5, 7, 10

up, 3, 4