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:
$\text{s-inc}\,(x)$
$= \ \ \textbf{if}\ \text{empty}\,(x)\ \ \textbf{then}\ \textsc{e}$
$\quad \textbf{else}\ \text{a}\,(\text{s-inc}\,(\text{p}\,(x)), \text{inc}\,(\text{l}\,(x)))\ \textbf{endif}$

```
;; A2-Begin-S-INC
```

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

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

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
$\text{s-inc}(\text{i}(c\_x,\ x)) = \text{i}(\text{inc}(c\_x),\ \text{s-inc}(x))$

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

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

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

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

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 $\text{a}(\text{s-up}(\text{p}(x)),\ \text{up}(\text{l}(x)))$ **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
($\neg$ empty (x)) $\rightarrow$ (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
($\neg$ empty (x)) $\rightarrow$ (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
$\text{empty}\,(\text{s-ui}\,(x)) = \text{empty}\,(x)$

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

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

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

THEOREM: a2-ic-s-ui
$\text{s-ui}\,(\text{i}\,(c\_x,\ x)) = \text{i}\,(\text{ui}\,(c\_x),\ \text{s-ui}\,(x))$

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

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

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

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

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

```
;; 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:
s-ue (x)
$=$ **if** empty (x) **then** E
else a (s-ue (p (x)), ue (l (x))) **endif**

```
;; A2-Begin-S-UE
```

THEOREM: a2-empty-s-ue
empty (s-ue (x)) $=$ empty (x)

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

THEOREM: a2-lp-s-ue
len (s-ue (x)) $=$ len (x)

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

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

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

THEOREM: a2-pc-s-ue
p (s-ue (x)) $=$ s-ue (p (x))

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

THEOREM: a2-bc-s-ue
b (s-ue (x)) $=$ s-ue (b (x))

THEOREM: a2-bnc-s-ue
bn (n, s-ue (x)) $=$ s-ue (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 (sy-a (ln, x)) = sy-a (ln, p (x))

```
;; A2-End-SY-A
```

DEFINITION:
topor-sy-b (ln)
= **if** ln = `'ypc` **then** `0`
elseif ln = `'ypcn` **then** `1`
elseif ln = `'ypr` **then** `1`
elseif ln = `'ypr2` **then** `0`
elseif ln = `'yi` **then** `1`
elseif ln = `'yi2` **then** `0`
elseif ln = `'ye` **then** `1`
elseif ln = `'yout` **then** `0`
else `0` **endif**

DEFINITION:
sy-b (ln, x)
= **if** ln = `'ypc`
then if empty (x) **then** E
else i (`0`, sy-b (`'ypcn`, p (x))) **endif**
elseif ln = `'ypcn` **then** s-inc (sy-b (`'ypc`, x))
elseif ln = `'ypr` **then** s-up (sy-b (`'ypc`, x))
elseif ln = `'ypr2`
then if empty (x) **then** E
else i (`0`, sy-b (`'ypr`, p (x))) **endif**
elseif ln = `'yi` **then** s-ui (sy-b (`'ypr2`, x))
elseif ln = `'yi2`
then if empty (x) **then** E
else i (`0`, sy-b (`'yi`, p (x))) **endif**
elseif ln = `'ye` **then** s-ue (sy-b (`'yi2`, x))
elseif ln = `'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
empty (sy-b (ln, x)) = empty (x)

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

```
;; A2-End-SY-B

; SOME ANIMATION:
; (setq x5T (A (A (A (A (A (e) T) T) T) T) T))
;*(sy-A 'Yout x5t)
; (A (A (A (A (A (E) 0)
;       '(UE (UI (UP 0))))
;           '(UE (UI (UP 1))))
;        '(UE (UI (UP 2))))
;     '(UE (UI (UP 3))))
;*(sy-A 'Ye x5t)
; (A (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 (A (E) 0)
;       '(UE 0))
;    '(UE (UI 0)))
;        '(UE (UI (UP 0))))
;     '(UE (UI (UP 1))))
;*(sy-B 'Ye x5t)
; (A (A (A (A (A (E) '(UE 0))
;       '(UE (UI 0)))
;           '(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 0) (UE (UI 0))
; 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
sy-b ('ypc, x) = sy-a ('ypc, x)

THEOREM: eq-a-b
sy-b ('ye, x) = sy-a ('yec, x)

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

THEOREM: eq-a-b2
sy-b ('ye, x)
= **if** empty (x) **then** E
  else i (ue (0),
    if empty (p (x)) **then** E
    else i (ue (ui (0)), sy-a ('ye, p (p (x)))) **endif**) **endif**

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

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

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

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

```
; eof: ppltcpu.bm
;))
```

Index