```
#|
```



```
|#
```

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

```
; multadd.bm: an iterative multiplier, built from an adder;
;    This is Paillet #7.
;    Note: this is the first multi-input circuit which forced the
;    EQ-LEN hyps. Also, we prove the same Deroulement as Paillet,
;    an induction over the macrocycle would be illegible to specify
;    and let alone prove.
;

;;; CIRCUIT in SUGARED form:
#|
(setq sysd '(sy-MULTADD (Xx Xy)
```

```
(Y1 S mux Ydone Xx Y3)
(YN R 0 Y1)          ; initially called l2 in Paillet, but then renamed to N
(Y3 S dec YN)
(Y4 S dec Y3)
(Y5 S dec Xx)
(Y6 S mux Ydone Y5 Y4)
(Y7 S eq0 Y6)
(Yt S or Y7 Y8)
(Ydone R T Yt)
(Y8 S eq0 Xy)
(Y9 S eq0 Xx)
(Y0 S const 0 Xx)
(Y10 S mux Y9 Y0 Xy)
(Y11 S mux Ydone Y10 Y12)
(Yout R 0 Y11)
(Y12 S plus Y10 Yout)
))

(setq multadd '(  |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; comb_dec.bm: DECrement combinational element (with dec 0 = 0 )
; U7-DONE
```

DEFINITION: $\mathrm{dec}\,(u) = (u - 1)$

```
; Everything below generated by: (bmcomb 'dec '() '(x))
```

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

```
;; A2-Begin-S-DEC
```

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

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

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

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

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

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

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

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

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

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

```
;; A2-End-S-DEC

; eof:comb_dec.bm

; comb_eq0.bm: (slightly abstract) equal to 0 test, could be done with: XOR 1 u
; U7-DONE
```

DEFINITION: $\text{eq0}\,(u) = (u \simeq \mathtt{0})$

```
; Everything below generated by: (bmcomb 'eq0 '() '(x))
```

DEFINITION:
$\text{s-eq0}\,(x)$
$= \ $ **if** $\text{empty}\,(x)$ **then** E
else $\text{a}\,(\text{s-eq0}\,(\text{p}\,(x)),\ \text{eq0}\,(\text{l}\,(x)))$ **endif**

```
;; A2-Begin-S-EQ0
```

THEOREM: a2-empty-s-eq0
$\text{empty}\,(\text{s-eq0}\,(x)) = \text{empty}\,(x)$

THEOREM: a2-e-s-eq0
$(\text{s-eq0}\,(x) = \text{E}) = \text{empty}\,(x)$

Theorem: a2-lp-s-eq0
$\mathrm{len}(\text{s-eq0}(x)) = \mathrm{len}(x)$

Theorem: a2-lpe-s-eq0
$\mathrm{eqlen}(\text{s-eq0}(x),\ x)$

Theorem: a2-ic-s-eq0
$\text{s-eq0}(\mathrm{i}(c\_x,\ x)) = \mathrm{i}(\mathrm{eq0}(c\_x),\ \text{s-eq0}(x))$

Theorem: a2-lc-s-eq0
$(\neg\ \mathrm{empty}(x)) \rightarrow (\mathrm{l}(\text{s-eq0}(x)) = \mathrm{eq0}(\mathrm{l}(x)))$

Theorem: a2-pc-s-eq0
$\mathrm{p}(\text{s-eq0}(x)) = \text{s-eq0}(\mathrm{p}(x))$

Theorem: a2-hc-s-eq0
$(\neg\ \mathrm{empty}(x)) \rightarrow (\mathrm{h}(\text{s-eq0}(x)) = \mathrm{eq0}(\mathrm{h}(x)))$

Theorem: a2-bc-s-eq0
$\mathrm{b}(\text{s-eq0}(x)) = \text{s-eq0}(\mathrm{b}(x))$

Theorem: a2-bnc-s-eq0
$\mathrm{bn}(n,\ \text{s-eq0}(x)) = \text{s-eq0}(\mathrm{bn}(n,\ x))$

```
;; A2-End-S-EQ0

; eof:comb_eq0.bm

; comb_mux.bm: Mux combinational element, i.e. "if".
; U7-DONE
```

Definition:
mux (*u1*, *u2*, *u3*)
= **if** *u1* **then** *u2*
else *u3* **endif**

```
; everything below generated by: (bmcomb 'mux '() '(x1 x2 x3))
; with the EXCEPTIONS/HAND-MODIFICATIONS given below.
```

Definition:
s-mux (*x1*, *x2*, *x3*)
= **if** empty (*x1*) **then** E
else a (s-mux (p (*x1*), p (*x2*), p (*x3*)), mux (l (*x1*), l (*x2*), l (*x3*))) **endif**

```
; SMUX-is-SIF can make things much simpler on occasions:
```

THEOREM: smux-is-sif
$\text{s-mux}\,(x1,\ x2,\ x3) = \text{s-if}\,(x1,\ x2,\ x3)$

EVENT: Disable smux-is-sif.

```
; We take advantage of SMUX-is-SIF for all inductive proofs. To do so we
; HAND-MODIFY the code generated by Sugar to replace all the hints by
;   - A2-EMPTY, A2-PC replace hint with: ((enable smux-is-sif))
;   - A2-LP, A2-IC, A2-HC, A2-BC: ((enable smux-is-sif) (disable len))
;   - A2-BNC: ((enable smux-is-sif) (disable bn len))

;; A2-Begin-S-MUX
```

THEOREM: a2-empty-s-mux
$\text{empty}\,(\text{s-mux}\,(x1,\ x2,\ x3)) = \text{empty}\,(x1)$

THEOREM: a2-e-s-mux
$(\text{s-mux}\,(x1,\ x2,\ x3) = \text{E}) = \text{empty}\,(x1)$

THEOREM: a2-lp-s-mux
$\text{len}\,(\text{s-mux}\,(x1,\ x2,\ x3)) = \text{len}\,(x1)$

THEOREM: a2-lpe-s-mux
$\text{eqlen}\,(\text{s-mux}\,(x1,\ x2,\ x3),\ x1)$

THEOREM: a2-ic-s-mux
$((\text{len}\,(x1) = \text{len}\,(x2)) \land (\text{len}\,(x2) = \text{len}\,(x3)))$
$\rightarrow \quad (\text{s-mux}\,(\text{i}\,(c\_x1,\ x1),\ \text{i}\,(c\_x2,\ x2),\ \text{i}\,(c\_x3,\ x3))$
$\qquad = \quad \text{i}\,(\text{mux}\,(c\_x1,\ c\_x2,\ c\_x3),\ \text{s-mux}\,(x1,\ x2,\ x3)))$

THEOREM: a2-lc-s-mux
$(\neg\ \text{empty}\,(x1)) \rightarrow (\text{l}\,(\text{s-mux}\,(x1,\ x2,\ x3)) = \text{mux}\,(\text{l}\,(x1),\ \text{l}\,(x2),\ \text{l}\,(x3)))$

THEOREM: a2-pc-s-mux
$\text{p}\,(\text{s-mux}\,(x1,\ x2,\ x3)) = \text{s-mux}\,(\text{p}\,(x1),\ \text{p}\,(x2),\ \text{p}\,(x3))$

THEOREM: a2-hc-s-mux
$((\neg\ \text{empty}\,(x1)) \land ((\text{len}\,(x1) = \text{len}\,(x2)) \land (\text{len}\,(x2) = \text{len}\,(x3))))$
$\rightarrow \quad (\text{h}\,(\text{s-mux}\,(x1,\ x2,\ x3)) = \text{mux}\,(\text{h}\,(x1),\ \text{h}\,(x2),\ \text{h}\,(x3)))$

```
;old:    ((DISABLE MUX S-MUX) (ENABLE H LEN) (INDUCT (S-MUX X1 X2 X3)))
```

THEOREM: a2-bc-s-mux
$((\mathrm{len}(x1) = \mathrm{len}(x2)) \wedge (\mathrm{len}(x2) = \mathrm{len}(x3)))$
$\rightarrow \quad (\mathrm{b}(\text{s-mux}(x1, x2, x3)) = \text{s-mux}(\mathrm{b}(x1), \mathrm{b}(x2), \mathrm{b}(x3)))$

```
;old:    ((DISABLE MUX) (ENABLE B LEN) (INDUCT (S-MUX X1 X2 X3)))
```

THEOREM: a2-bnc-s-mux
$((\mathrm{len}(x1) = \mathrm{len}(x2)) \wedge (\mathrm{len}(x2) = \mathrm{len}(x3)))$
$\rightarrow \quad (\mathrm{bn}(n, \text{s-mux}(x1, x2, x3)) = \text{s-mux}(\mathrm{bn}(n, x1), \mathrm{bn}(n, x2), \mathrm{bn}(n, x3)))$

```
;old: ((DISABLE MUX S-MUX))

;; A2-End-S-MUX

; eof:comb_mux.bm

; comb_plus.bm: Plus combinational element.
; U7-DONE

;   no character function definition since BM already knows about Plus..

; Everything below generated by:     (bmcomb 'plus '() '(x y))
```

DEFINITION:
s-plus (x, y)
$=$ **if** empty (x) **then** E
else a (s-plus (p (x), p (y)), l (x) + l (y)) **endif**

```
;; A2-Begin-S-PLUS
```

THEOREM: a2-empty-s-plus
empty (s-plus (x, y)) = empty (x)

THEOREM: a2-e-s-plus
(s-plus (x, y) = E) = empty (x)

THEOREM: a2-lp-s-plus
len (s-plus (x, y)) = len (x)

THEOREM: a2-lpe-s-plus
eqlen (s-plus (x, y), x)

THEOREM: a2-ic-s-plus
$(\text{len}(x) = \text{len}(y))$
$\rightarrow \quad (\text{s-plus}(\text{i}(c\_x,\ x),\ \text{i}(c\_y,\ y)) = \text{i}(c\_x + c\_y,\ \text{s-plus}(x,\ y)))$

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

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

THEOREM: a2-hc-s-plus
$((\neg\ \text{empty}(x)) \wedge (\text{len}(x) = \text{len}(y)))$
$\rightarrow \quad (\text{h}(\text{s-plus}(x,\ y)) = (\text{h}(x) + \text{h}(y)))$

THEOREM: a2-bc-s-plus
$(\text{len}(x) = \text{len}(y)) \rightarrow (\text{b}(\text{s-plus}(x,\ y)) = \text{s-plus}(\text{b}(x),\ \text{b}(y)))$

THEOREM: a2-bnc-s-plus
$(\text{len}(x) = \text{len}(y)) \rightarrow (\text{bn}(n,\ \text{s-plus}(x,\ y)) = \text{s-plus}(\text{bn}(n,\ x),\ \text{bn}(n,\ y)))$

```
;; A2-End-S-PLUS
```

```
; eof:comb_plus.bm
```

DEFINITION:
topor-sy-multadd (*ln*)
= **if** *ln* = `'y1` **then** `2`
elseif *ln* = `'yn` **then** `0`
elseif *ln* = `'y3` **then** `1`
elseif *ln* = `'y4` **then** `2`
elseif *ln* = `'y5` **then** `1`
elseif *ln* = `'y6` **then** `3`
elseif *ln* = `'y7` **then** `4`
elseif *ln* = `'yt` **then** `5`
elseif *ln* = `'ydone` **then** `0`
elseif *ln* = `'y8` **then** `1`
elseif *ln* = `'y9` **then** `1`
elseif *ln* = `'y0` **then** `1`
elseif *ln* = `'y10` **then** `2`
elseif *ln* = `'y11` **then** `4`
elseif *ln* = `'yout` **then** `0`
elseif *ln* = `'y12` **then** `3`
else `0` **endif**

```
;Parameter found: 0 in: (Y0 S CONST 0 XX)
```

DEFINITION:
sy-multadd (ln, xx, xy)
= **if** ln = `'y1`
then s-mux (sy-multadd (`'ydone`, xx, xy), xx, sy-multadd (`'y3`, xx, xy))
elseif ln = `'yn`
then if empty (xx) **then** E
else i (0, sy-multadd (`'y1`, p (xx), p (xy))) **endif**
elseif ln = `'y3` **then** s-dec (sy-multadd (`'yn`, xx, xy))
elseif ln = `'y4` **then** s-dec (sy-multadd (`'y3`, xx, xy))
elseif ln = `'y5` **then** s-dec (xx)
elseif ln = `'y6`
then s-mux (sy-multadd (`'ydone`, xx, xy),
sy-multadd (`'y5`, xx, xy),
sy-multadd (`'y4`, xx, xy))
elseif ln = `'y7` **then** s-eq0 (sy-multadd (`'y6`, xx, xy))
elseif ln = `'yt`
then s-or (sy-multadd (`'y7`, xx, xy), sy-multadd (`'y8`, xx, xy))
elseif ln = `'ydone`
then if empty (xx) **then** E
else i (**t**, sy-multadd (`'yt`, p (xx), p (xy))) **endif**
elseif ln = `'y8` **then** s-eq0 (xy)
elseif ln = `'y9` **then** s-eq0 (xx)
elseif ln = `'y0` **then** s-const (0, xx)
elseif ln = `'y10`
then s-mux (sy-multadd (`'y9`, xx, xy), sy-multadd (`'y0`, xx, xy), xy)
elseif ln = `'y11`
then s-mux (sy-multadd (`'ydone`, xx, xy),
sy-multadd (`'y10`, xx, xy),
sy-multadd (`'y12`, xx, xy))
elseif ln = `'yout`
then if empty (xx) **then** E
else i (0, sy-multadd (`'y11`, p (xx), p (xy))) **endif**
elseif ln = `'y12`
then s-plus (sy-multadd (`'y10`, xx, xy), sy-multadd (`'yout`, xx, xy))
else sfix (xx) **endif**

```
;; A2-Begin-SY-MULTADD
```

THEOREM: a2-empty-sy-multadd
$(\mathrm{len}(xx) = \mathrm{len}(xy)) \rightarrow (\mathrm{empty}(\text{sy-multadd}(ln, xx, xy)) = \mathrm{empty}(xx))$

THEOREM: a2-e-sy-multadd
$(\mathrm{len}(xx) = \mathrm{len}(xy)) \rightarrow ((\text{sy-multadd}(ln, xx, xy) = \mathrm{E}) = \mathrm{empty}(xx))$

THEOREM: a2-lp-sy-multadd
$(\text{len}(xx) = \text{len}(xy)) \rightarrow (\text{len}(\text{sy-multadd}(ln,\ xx,\ xy)) = \text{len}(xx))$

THEOREM: a2-lpe-sy-multadd
$(\text{len}(xx) = \text{len}(xy)) \rightarrow \text{eqlen}(\text{sy-multadd}(ln,\ xx,\ xy),\ xx)$

THEOREM: a2-pc-sy-multadd
$(\text{len}(xx) = \text{len}(xy))$
$\rightarrow \quad (\text{p}(\text{sy-multadd}(ln,\ xx,\ xy)) = \text{sy-multadd}(ln,\ \text{p}(xx),\ \text{p}(xy)))$

```
;; A2-End-SY-MULTADD

;;; Circuit CORRECTNESS /Paillet:

; first we get the simplified (generalized) sysd:
```

DEFINITION:
```
sy-m2 (ln, xx, xy)
=   if ln = 'yn
    then if empty (xx)  then E
         else i (0,
                 s-mux (sy-m2 ('ydone, p (xx), p (xy)),
                        p (xx),
                        s-dec (sy-m2 ('yn, p (xx), p (xy))))) endif
    elseif ln = 'ydone
    then if empty (xx)  then E
         else i (t,
                 s-or (s-eq0 (s-mux (sy-m2 ('ydone, p (xx), p (xy)),
                                     s-dec (p (xx)),
                                     s-dec (s-dec (sy-m2 ('yn, p (xx), p (xy)))))),
                       s-eq0 (p (xy)))) endif
    elseif ln = 'yout
    then if empty (xx)  then E
         else i (0,
                 s-mux (sy-m2 ('ydone, p (xx), p (xy)),
                        s-mux (s-eq0 (p (xx)), s-const (0, p (xx)), p (xy)),
                        s-plus (s-mux (s-eq0 (p (xx)),
                                       s-const (0, p (xx)),
                                       p (xy)),
                                sy-m2 ('yout, p (xx), p (xy))))) endif
    else sfix (xx) endif
```

```
; M2 is just a GENERALIZED sysd, and our A2 lemmas should still be
; true. The following were (Sugar) generated by:
```

```
; (vp (bma2sysd-aux 'sy-M2 'sy-M2 '(Xx Xy)
;                     '(DEC OR EQ0 CONST MUX PLUS)))
; and then A2-PC was taken out by hand, preemptively.

;; A2-Begin-SY-M2
```

THEOREM: a2-empty-sy-m2
(len (xx) = len (xy)) → (empty (sy-m2 (ln, xx, xy)) = empty (xx))

THEOREM: a2-e-sy-m2
(len (xx) = len (xy)) → ((sy-m2 (ln, xx, xy) = E) = empty (xx))

THEOREM: a2-lp-sy-m2
(len (xx) = len (xy)) → (len (sy-m2 (ln, xx, xy)) = len (xx))

THEOREM: a2-lpe-sy-m2
(len (xx) = len (xy)) → eqlen (sy-m2 (ln, xx, xy), xx)

```
;(PROVE-LEMMA A2-PC-SY-M2 (REWRITE)
;    (IMPLIES (EQUAL (LEN XX) (LEN XY))
;             (EQUAL (P (SY-M2 LN XX XY)) (SY-M2 LN (P XX) (P XY))))
;    ((DISABLE LEN S-DEC S-OR S-EQ0 S-CONST S-MUX S-PLUS A2-IC-S-DEC
;              A2-IC-S-OR A2-IC-S-EQ0 A2-IC-S-CONST A2-IC-S-MUX
;              A2-IC-S-PLUS)
;     (ENABLE STR-ADD1-LEN-P)))

;; A2-End-SY-M2

; This KEY equality lemma is proved immediately, 2 cases
; on empty Xx:
```

THEOREM: multadd-is-m2
(sy-multadd (`'yout`, xx, xy) = sy-m2 (`'yout`, xx, xy))
∧ (sy-multadd (`'ydone`, xx, xy) = sy-m2 (`'ydone`, xx, xy))
∧ (sy-multadd (`'yn`, xx, xy) = sy-m2 (`'yn`, xx, xy))

```
; Note: simplified sysd was obtained w/ same hint as above minus
; induction and M2 expansions, and replacing sy-multadd by sy-m2
; by hand, IN:
;(dcl dummy ())
;(prove-lemma d1 ()
;(implies (not (empty Xx))
;  (equal (sy-multadd 'Ydone Xx Xy) (dummy)))
```

```
;)


; The FUNDAMENTAL invariant property for the circuit is:
```

THEOREM: multadd-correct-m2-inv

$$
\begin{aligned}
&((xa \in \mathbf{N})\\
&\ \ \wedge\ \ (xb \in \mathbf{N})\\
&\ \ \wedge\ \ (k \in \mathbf{N})\\
&\ \ \wedge\ \ (k < (1 + (1 + xa)))\\
&\ \ \wedge\ \ (\mathtt{1} < k)\\
&\ \ \wedge\ \ (xa \not\simeq \mathtt{0})\\
&\ \ \wedge\ \ (xb \not\simeq \mathtt{0}))\\
&\rightarrow\ \ ((\mathrm{l}\,(\text{sy-m2}\,(\texttt{'yout},\ \text{s-constl}\,(xa,\ k),\ \text{s-constl}\,(xb,\ k)))\\
&\qquad =\ \ (xb * (k - 1)))\\
&\qquad \wedge\ \ (\mathrm{l}\,(\text{sy-m2}\,(\texttt{'ydone},\ \text{s-constl}\,(xa,\ k),\ \text{s-constl}\,(xb,\ k)))\\
&\qquad\qquad =\ \ (xa = (k - 1)))\\
&\qquad \wedge\ \ (\mathrm{l}\,(\text{sy-m2}\,(\texttt{'yn},\ \text{s-constl}\,(xa,\ k),\ \text{s-constl}\,(xb,\ k)))\\
&\qquad\qquad =\ \ (xa - ((k - 1) - 1))))
\end{aligned}
$$

```
; Now it's just a matter of instantiating it to the right
; Deroulement, and moving to MULTADD.

; NOTE: in a first draft of this proof, we first went to the
; identical theorem in terms of M2, and then passed over trivially.
; Attempting direct results about SY-MULTADD during experimentation
; is asking for trouble...

; NOTE: factoring out below Xx and Xy in the hyps as is done below
; yields a true,provable, but UNUSABLE rewrite thm because Xa and
; Xb are FREE. It looks cuter though, so we may want it that way
; in FINAL thms.
```

THEOREM: multadd-correct

$$
\begin{aligned}
&((xa \in \mathbf{N})\\
&\ \ \wedge\ \ (xb \in \mathbf{N})\\
&\ \ \wedge\ \ (xa \not\simeq \mathtt{0})\\
&\ \ \wedge\ \ (xb \not\simeq \mathtt{0})\\
&\ \ \wedge\ \ (xx = \text{s-constl}\,(xa,\ 1 + xa))\\
&\ \ \wedge\ \ (xy = \text{s-constl}\,(xb,\ 1 + xa)))\\
&\rightarrow\ \ ((\mathrm{l}\,(\text{sy-multadd}\,(\texttt{'yout},\ xx,\ xy)) = (xb * xa))\\
&\qquad \wedge\ \ (\mathrm{l}\,(\text{sy-multadd}\,(\texttt{'ydone},\ xx,\ xy)) = \mathbf{t}))
\end{aligned}
$$

```
; And of treating the case where either input is 0 separately:
```

THEOREM: multadd-correct-case-0

$$
\begin{array}{l}
((xa \in \mathbf{N}) \\
\ \wedge\quad (xb \in \mathbf{N}) \\
\ \wedge\quad ((xa \simeq \mathtt{0}) \vee (xb \simeq \mathtt{0})) \\
\ \wedge\quad (xx = \text{s-constl}\,(xa,\, \mathtt{2})) \\
\ \wedge\quad (xy = \text{s-constl}\,(xb,\, \mathtt{2}))) \\
\rightarrow\quad ((\text{l}\,(\text{sy-multadd}\,(\mathtt{'yout},\, xx,\, xy)) = (xb * xa)) \\
\qquad\quad \wedge\quad (\text{l}\,(\text{sy-multadd}\,(\mathtt{'ydone},\, xx,\, xy)) = \mathbf{t}))
\end{array}
$$

```
; eof: multadd.bm
;))
```

Index