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

; serial.bm: a register, writable in parallel, and readable
; serially. This is Paillet example 3
;
; IMPORTANT NOTE: originally, we proved this multi-circuit WITHOUT
; EQ-LEN hyps, because we got lucky and all the inputs were
; Registered, so it didn’t matter. We now go to the more general
; version for uniformity, even though it will be enormously more
; expensive, and will force EQ-LEN hyps in the correctness thms.
; It might be worth remembering though that for circuits where all
; inputs are immediately Registered, we can do away with the
; EQ-LEN hyp.
;; OTHER IMPORTANT NOTE: all the comments below concerning various 
;; ways of phrasing the hypotheses were written when EMPTY was still 
;; ENABLED.

;;; CIRCUIT in SUGARED form: (after flattening out, yuck...)

#|
(setq sysd '(sy-SERIAL (xC x1 x2 x3)
(YC0 S const 0 xC)
(YM3 S mux xC x3 YC0)
(Y3 R 'a3 YM3)
(YM2 S mux xC x2 Y3)
(Y2 R 'a2 YM2)
(YM1 S mux xC x1 Y2)
(Y1 R 'a1 YM1)
))

(setq serial '( |#
; BM DEFINITIONS and A2 LEMMAS, generated by BMSYSD:
; 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
s-mux (x1, x2, x3) = 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))

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

;old: ((DISABLE MUX B-MUX) (ENABLE B LEN) (INDUCT (S-MUX X1 X2 X3)))
THEOREM: \texttt{a2-bnc-s-mux}
\[ ((\text{len}(x_1) = \text{len}(x_2)) \land (\text{len}(x_2) = \text{len}(x_3))) \rightarrow (\text{bn}(n, s\text{-mux}(x_1, x_2, x_3)) = s\text{-mux}(\text{bn}(n, x_1), \text{bn}(n, x_2), \text{bn}(n, x_3))) \]

; old: \texttt{(DISABLE MUX S-MUX)}

;; A2-End-S-MUX

; eof: comb_mux.bm

DEFINITION:
\texttt{topor-sy-serial}(\texttt{ln}) =
\begin{align*}
\text{if} \texttt{ln} = \texttt{'yc0} & \text{ then } 1 \\
\text{elseif} \texttt{ln} = \texttt{'ym3} & \text{ then } 2 \\
\text{elseif} \texttt{ln} = \texttt{'y3} & \text{ then } 0 \\
\text{elseif} \texttt{ln} = \texttt{'ym2} & \text{ then } 1 \\
\text{elseif} \texttt{ln} = \texttt{'y2} & \text{ then } 0 \\
\text{elseif} \texttt{ln} = \texttt{'ym1} & \text{ then } 1 \\
\text{elseif} \texttt{ln} = \texttt{'y1} & \text{ then } 0 \\
\text{else} & \text{ endif}
\end{align*}

; Parameter found: 0 in: (YC0 S CONST 0 XC)

DEFINITION:
\texttt{sy-serial}(\texttt{ln}, \texttt{xc}, x_1, x_2, x_3) =
\begin{align*}
\text{if} \texttt{ln} = \texttt{'yc0} & \text{ then s-const}(0, xc) \\
\text{elseif} \texttt{ln} = \texttt{'ym3} & \text{ then s-mux}(xc, x_3, \texttt{sy-serial}('yc0, xc, x_1, x_2, x_3)) \\
\text{elseif} \texttt{ln} = \texttt{'y3} & \text{ then if empty}(xc) \text{ then } E \\
\text{else} & \text{ endif}
\end{align*}

\begin{align*}
\text{else} & \text{ i('a3, sy-serial('ym3, p(xc), p(x1), p(x2), p(x3))) endif} \\
\text{elseif} \texttt{ln} = \texttt{'ym2} & \text{ then s-mux}(xc, x_2, \texttt{sy-serial}('y3, xc, x_1, x_2, x_3)) \\
\text{elseif} \texttt{ln} = \texttt{'y2} & \text{ then if empty}(xc) \text{ then } E \\
\text{else} & \text{ endif}
\end{align*}

\begin{align*}
\text{else} & \text{ i('a2, sy-serial('ym2, p(xc), p(x1), p(x2), p(x3))) endif} \\
\text{elseif} \texttt{ln} = \texttt{'ym1} & \text{ then s-mux}(xc, x_1, \texttt{sy-serial}('y2, xc, x_1, x_2, x_3)) \\
\text{elseif} \texttt{ln} = \texttt{'y1} & \text{ then if empty}(xc) \text{ then } E \\
\text{else} & \text{ endif}
\end{align*}

\begin{align*}
\text{else} & \text{ sfix}(xc) \text{ endif}
\end{align*}

;; A2-Begin-SY-SERIAL
Theorem: a2-empty-sy-serial
((\text{len}(xc) = \text{len}(x1)) \land (\text{len}(x1) = \text{len}(x2)) \land (\text{len}(x2) = \text{len}(x3)))
\rightarrow (\text{empty}(\text{sy-serial}(ln, xc, x1, x2, x3)) = \text{empty}(xc))

Theorem: a2-e-sy-serial
((\text{len}(xc) = \text{len}(x1)) \land (\text{len}(x1) = \text{len}(x2)) \land (\text{len}(x2) = \text{len}(x3)))
\rightarrow ((\text{sy-serial}(ln, xc, x1, x2, x3) = e) = \text{empty}(xc))

Theorem: a2-lp-sy-serial
((\text{len}(xc) = \text{len}(x1)) \land (\text{len}(x1) = \text{len}(x2)) \land (\text{len}(x2) = \text{len}(x3)))
\rightarrow (\text{len}(\text{sy-serial}(ln, xc, x1, x2, x3))) = \text{len}(xc)

Theorem: a2-lpe-sy-serial
((\text{len}(xc) = \text{len}(x1)) \land (\text{len}(x1) = \text{len}(x2)) \land (\text{len}(x2) = \text{len}(x3)))
\rightarrow \text{eqlen}(\text{sy-serial}(ln, xc, x1, x2, x3), xc)

Theorem: a2-pc-sy-serial
((\text{len}(xc) = \text{len}(x1)) \land (\text{len}(x1) = \text{len}(x2)) \land (\text{len}(x2) = \text{len}(x3)))
\rightarrow \text{p}(\text{sy-serial}(ln, x1, x2, x3))

;; A2-End-SY-Serial

;; Circuit CORRECTNESS /Paillet:

; SPECIFICATION:

; Here we interpret Paillet as talking about last-chars implicitly

Definition:
serial-spec-l(xc, x1, x2, x3)
= \text{if} \text{l}(p(xc)) \text{ then } \text{l}(p(x1))
    \text{ elseif} \text{l}(p(p(xc))) \text{ then } \text{l}(p(p(x2)))
    \text{ elseif} \text{l}(p(p(p(xc)))) \text{ then } \text{l}(p(p(p(x3))))
    \text{ else } 0 \text{ endif}

; Here we interpret Paillet as really talking about streams (and
; correct for the missing initial values):

Definition:
serial-spec(xc, x1, x2, x3)
= \text{i}(\text{\textquoteleft}a1,\text{\textquoteleft}s-if}(p(xc),
\[
p (x_1), \\
i (a_2), \\
s - i f (p (p (x_c))), \\
p (p (x_2)), \\
i (a_3), \\
s - i f (p (p (p (x_c))), p (p (p (x_3))), s - c o n s t (0, p (p (p (x_c)))))) \\
\]

; CORRECTNESS:

; note: we don't need EQ-LEN hyp here, although it was tried and 
; didn't hurt.

THEOREM: serial-correct-1
\[
\neg \text{empty} (x_c) \\
\land \neg \text{empty} (p (x_c)) \\
\land \neg \text{empty} (p (p (x_c))) \\
\land \neg \text{empty} (p (p (p (x_c)))) \\
\rightarrow (s y - s e r i a l ('y_1, x_c, x_1, x_2, x_3)) = s e r i a l - s p e c l (x_c, x_1, x_2, x_3) \\
\]

; Note: we shouldn't need the EQ-LEN hyp here, since it's just an unfolding.

THEOREM: serial-correct
\[
\neg \text{empty} (p (p (p (x_c)))) \\
\rightarrow (s y - s e r i a l ('y_1, x_c, x_1, x_2, x_3)) = s e r i a l - s p e c (x_c, x_1, x_2, x_3) \\
\]

; NOTE that above we have a choice of how we phrase the hypothesis:
; 1: (and (not (empty xC)) (not (empty (p xC)))
; (not (empty (p (p xC)))) (not (empty (p (p (p xC))))))
; is highly redundant but says everything needed and so solves
; in 1 step.
; 2: (not (empty (p (p xC)))) concise, \( \rightarrow \) many cases (but
; LESS time!)
; 3: (not (empty (Pn 3 xC))) concise, \( \rightarrow \) same # cases as 2, but
; more time.
; Rewrite lemmas such as:
;(prove-lemma not-empty-Pn (rewrite)
;(equal (not (empty (Pn n x))))
; (if (zerop n)
; (not (empty x))
; (and (not (empty x))
; (not (empty (Pn (sub1 n) (P x))))))
;)
; although true, have no effect on the hypothesis expansion,
; unfortunately..

; Another property listed as "correctess" in Paillet is:
; Note that here we have translated P . . into L P . ., because
; if we try to understand this last Paillet property as speaking of
; streams, then the Hypothesis: P3 C = 1 and P2 C = P C = 0
; doesn't make any sense!!!
; in fact he acknowledges that "these computations are supposed to
; be made in a temporal interval corresponding to one cycle, but
; this interval is not indicated in the calculus to avoid too much
; notation". Formally of course, we don't have that luxury...

; Again, we don't need the EQ-LEN hyp, although when we tested it,
; it threw BM into a loop, until we DISABLED LEN; this trick might
; carry over!!

**THEOREM**: serial-correct-specialcase-l

\[
((l(p(xc)) = f) \\
\land (l(p(p(xc))) = f) \\
\land (l(p(p(p(xc)))) = t) \\
\land (\neg \text{empty}(xc)) \\
\land (\neg \text{empty}(p(xc))) \\
\land (\neg \text{empty}(p(p(xc)))) \\
\land (\neg \text{empty}(p(p(p(xc)))))) \\
\rightarrow (l(sy-serial('y1, xc, x1, x2, x3)) = l(p(p(xc)))) \\
\land (l(p(sy-serial('y1, xc, x1, x2, x3))) = l(p(p(xc)))) \\
\land (l(p(p(sy-serial('y1, xc, x1, x2, x3))) = l(p(p(p(xc))))))
\]

; Note above that using the (redundant) hypothesis:
; (not (empty xC)) (not (empty (p xC))) (not (empty (p (p xC))))
; (not (empty (p (p xC))))
; makes the proof instantaneous, since otherwise BM goes through
; eliminations to realize the "equal" hyps imply it.

; eof: serial.bm
;})
Index

a, 2
a2-bc-s-mux, 3
a2-bnc-s-mux, 4
a2-e-s-mux, 3
a2-e-sy-serial, 5
a2-empty-s-mux, 3
a2-empty-sy-serial, 5
a2-hc-s-mux, 3
a2-ie-s-mux, 3
a2-ic-s-mux, 3
a2-lp-s-mux, 3
a2-lp-sy-serial, 5
a2-lpe-s-mux, 3
a2-lpe-sy-serial, 5
a2-pc-s-mux, 3
a2-pc-sy-serial, 5
b, 3
bn, 4
e, 2–5
empty, 2–7
eqlen, 3, 5
h, 3
i, 3, 4, 6
l, 2, 3, 5–7
len, 3–5
mux, 2, 3
p, 2–7
s-const, 4, 6
s-if, 2, 6
s-mux, 2–4
serial-correct, 6
serial-correct-l, 6
serial-correct-specialcase-l, 7
serial-spec, 5, 6
serial-spec-l, 5, 6
sfix, 4
smux-is-sif, 2
sy-serial, 4–7
topor-sy-serial, 4