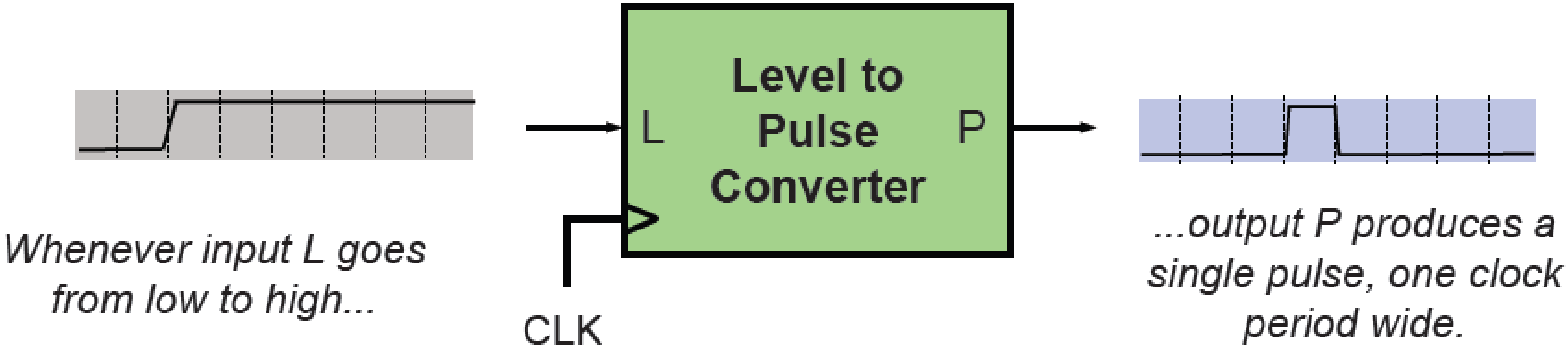
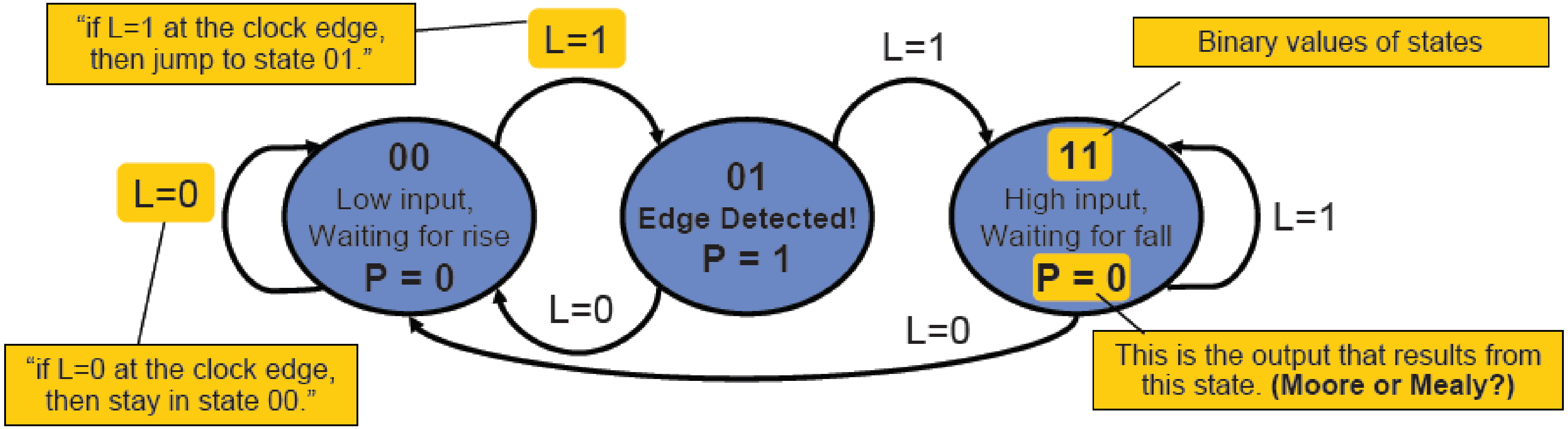
- A level-to-pulse converter produces a single-cycle pulse each time its input goes high.
- In other words, it's a synchronous rising-edge detector.
- Sample uses:
  - □ Buttons and switches pressed by humans for arbitrary periods of time
  - □ Single-cycle enable signals for counters



Whenever input L goes from low to high...

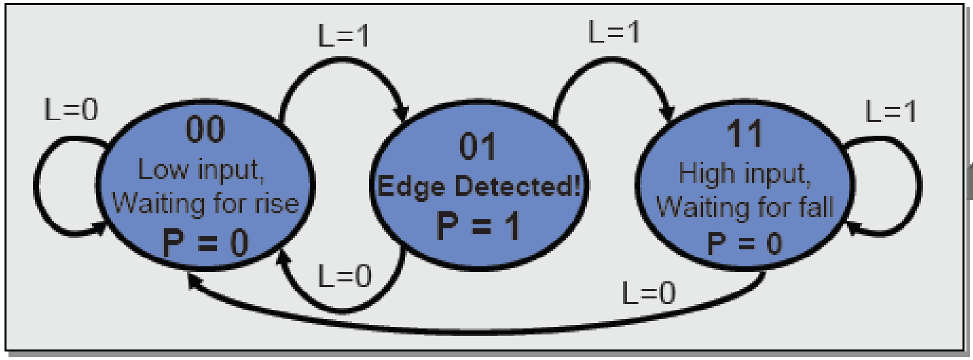
...output P produces a single pulse, one clock period wide.

- **State transition diagram** is a useful FSM representation and design aid


"if L=1 at the clock edge, then jump to state 01."
L=1
L=1
Binary values of states
L=0
00
Low input, Waiting for rise
P = 0
01
Edge Detected!
P = 1
11
High input, Waiting for fall
P = 0
L=1
L=0
L=0
"if L=0 at the clock edge, then stay in state 00."
This is the output that results from this state. (Moore or Mealy?)

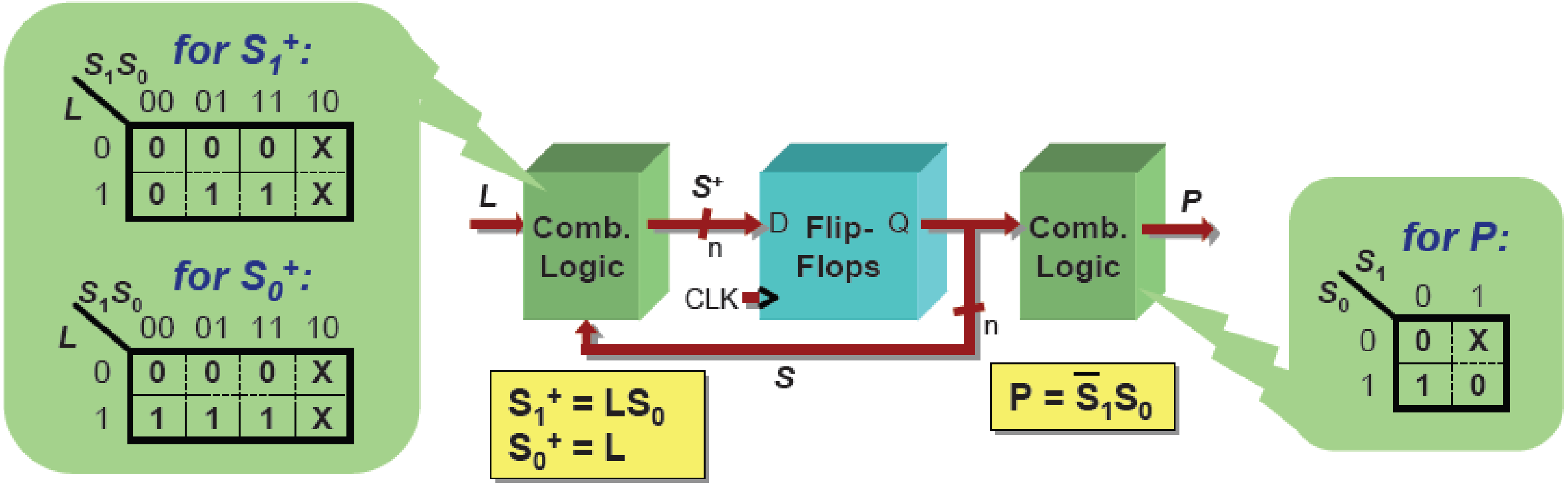
Logic Derivation for a Moore FSM

- Transition diagram is readily converted to a state transition table (just a truth table)



Current State		In	Next State		Out
S_1	S_0	L	S_1^+	S_0^+	P
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	0

- Combinational logic may be derived by Karnaugh maps



for S_1^+*:*

$L \backslash S_1S_0$	00	01	11	10
0	0	0	0	X
1	0	1	1	X

for S_0^+*:*

$L \backslash S_1S_0$	00	01	11	10
0	0	0	0	X
1	1	1	1	X

$S_1^+ = LS_0$

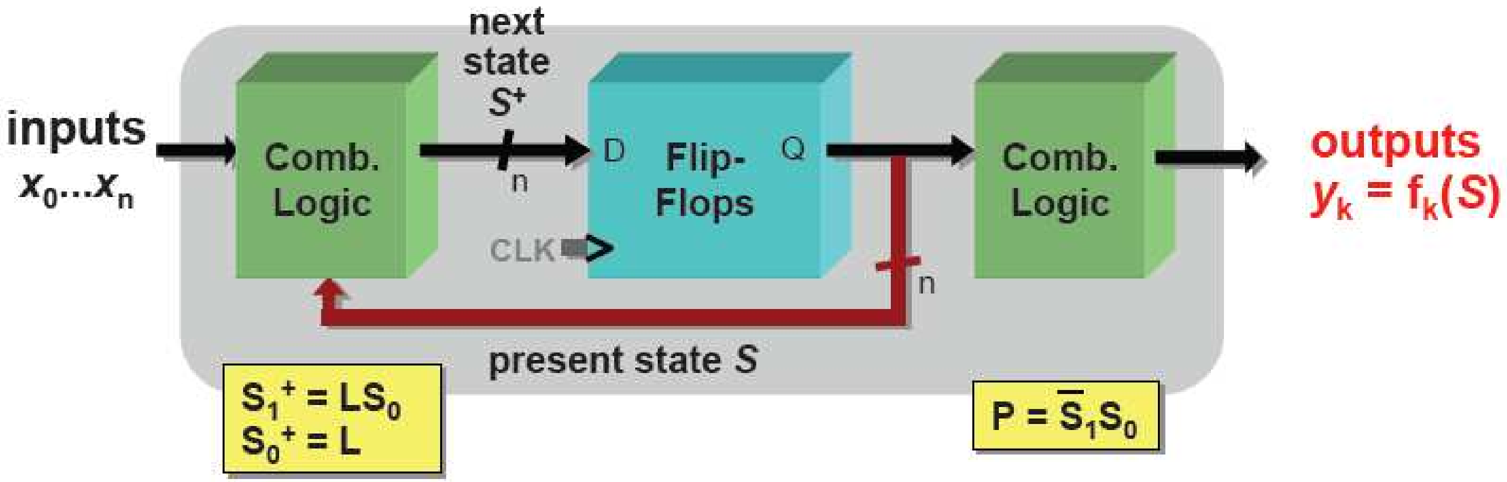
$S_0^+ = L$

$P = \overline{S_1}S_0$

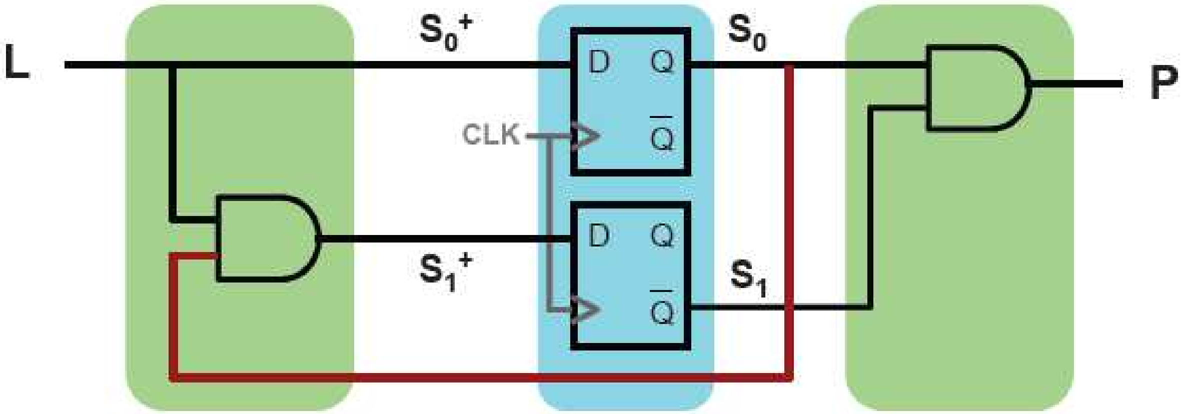
for P:

$S_0 \backslash S_1$	0	1
0	0	X
1	1	0

Moore Implementation

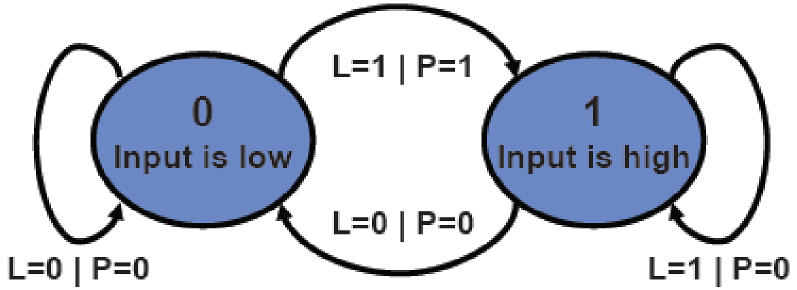


Moore FSM circuit implementation of level-to-pulse converter:



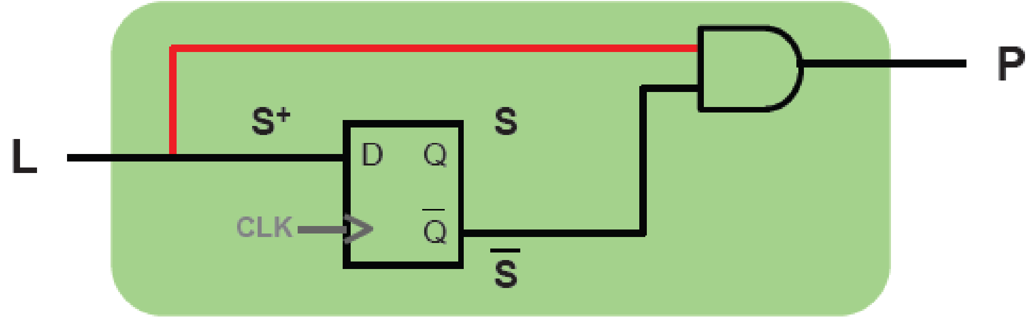
Note: 3 states, so $2^2 = 4>3$ i.e. 2 memory bits needed, with one unused state

Mealy FSM Design



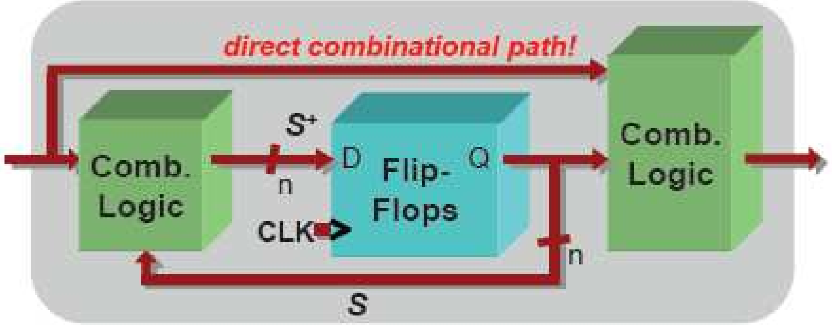
Pres. State	In	Next State	Out
S	*L*	*S^+*	*P*
0	0	0	0
0	1	1	1
1	0	0	0
1	1	1	0

Mealy FSM circuit implementation of level-to-pulse converter:

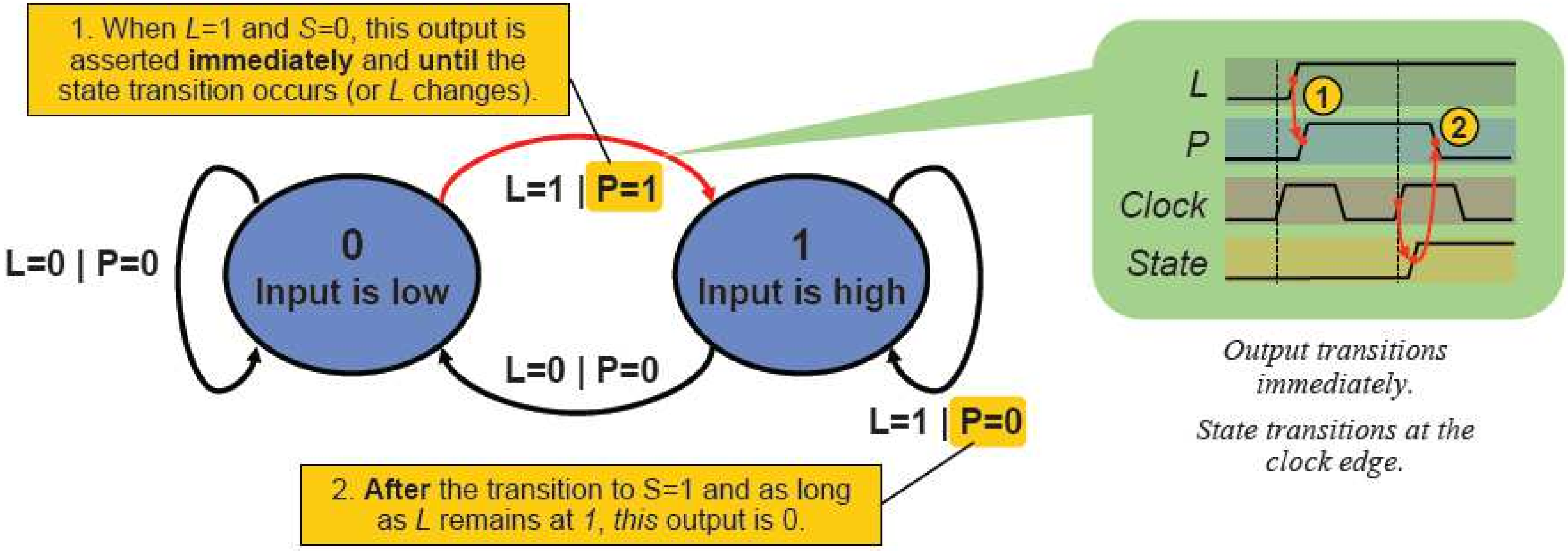


- FSM’s state simply remembers the previous value of L
- Circuit benefits from the Mealy FSM’s implicit single-cycle assertion of outputs during state transitions

Mealy FSM implementation



- Since outputs are determined by state *and* inputs, Mealy FSMs may need fewer states than Moore FSM implementations



1. When *L*=1 and *S*=0, this output is asserted **immediately** and **until** the state transition occurs (or *L* changes).

2. **After** the transition to S=1 and as long as *L* remains at *1*, *this* output is 0.

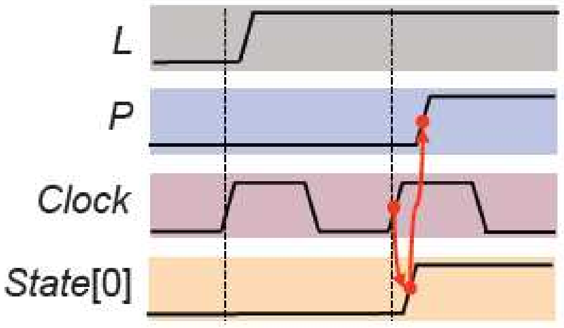
Output transitions immediately.

State transitions at the clock edge.

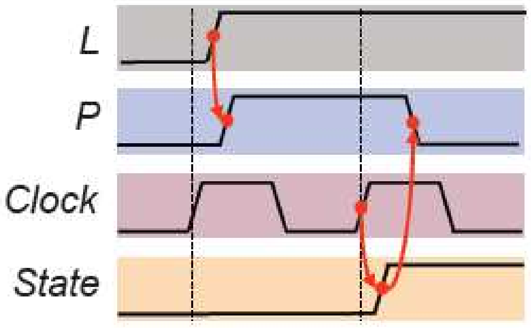
Mealy/Moore FSM Trade-Offs

- Remember that the difference is in the output:
  - Moore outputs are based on state only
  - Mealy outputs are based on state *and input*
  - Therefore, Mealy outputs generally occur one cycle earlier than a Moore:

Moore: delayed assertion of P



Mealy: immediate assertion of P



- Compared to a Moore FSM, a Mealy FSM might...
  - Be more difficult to conceptualize and design
  - Have fewer states