Synchronous Systems

Designing digital systems using a clock

Henry Fuchs

Slides adapted from Montek Singh, who adapted them
from Leonard McMillan and from Gary Bishop
Back to McMillan & Chris Terman, MIT 6.004 1999

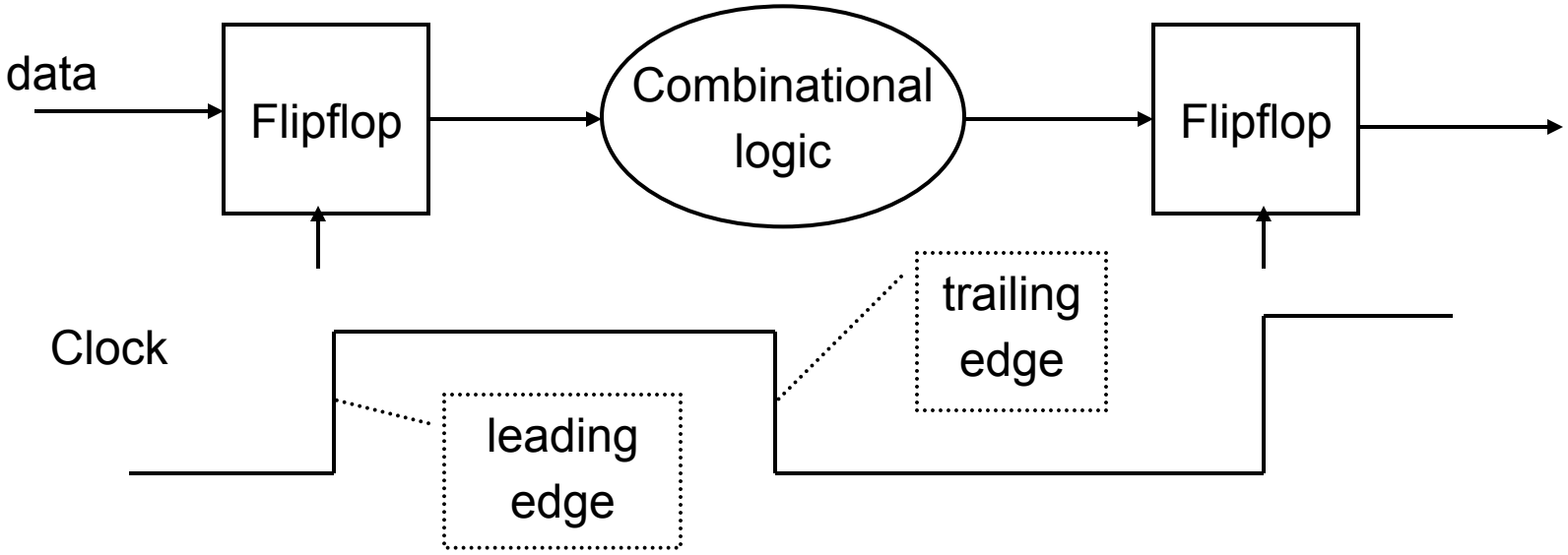
Thursday, April 2, 2015

Lecture 14

Topics

- Synchronous Systems
  - Clocking
- Finite State Machines
  - To control systems over time

Synchronous Systems



On the leading edge of the clock, the input of a flipflop is transferred to the output and held.

We must be sure the output of the combinational logic has *settled* before the next leading clock edge.

Clock period must be "long enough"

Clock Period >= $t_{FF} + t_{logic} + t_{setup}$

Finite State Machines

What is a State Machine?

- Remember automata?

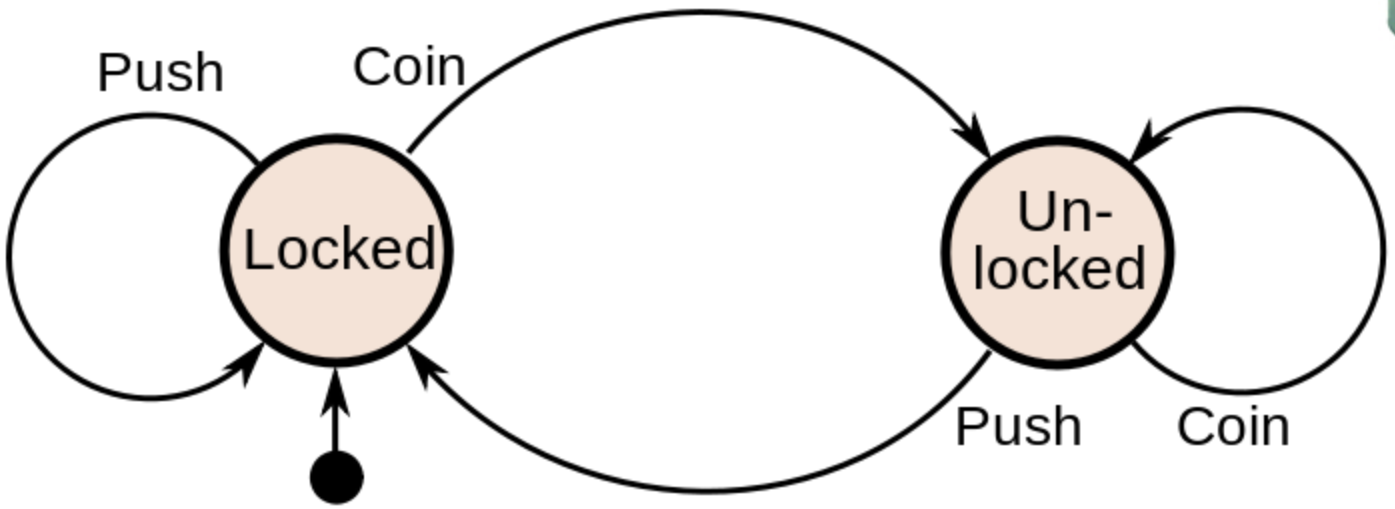
- It is defined by the following:
  - Set of STATES
  - Set of INPUTS
  - Set of OUTPUTS
  - A mapping from (STATES, INPUTS) to …

    … the next STATE and an OUTPUT

- STATE represents memory!

Very Simple Example: A Turnstile

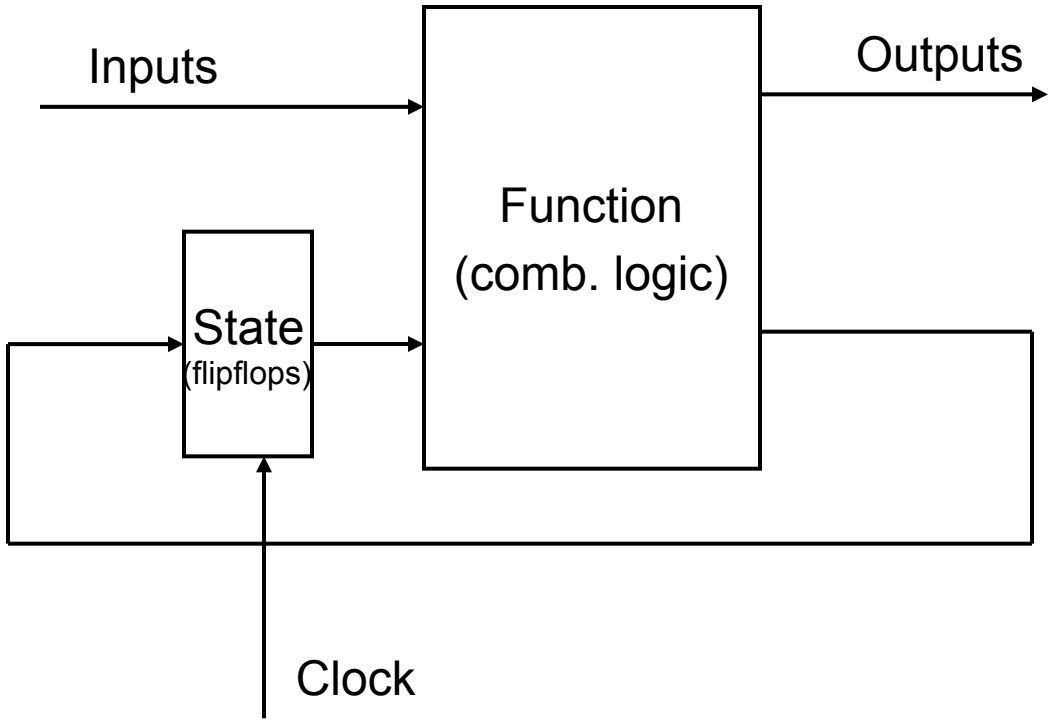
- You have to insert a coin to operate the turnstile
- Simply pushing doesn't work
- 2 states: Locked, Unlocked



From wikipedia

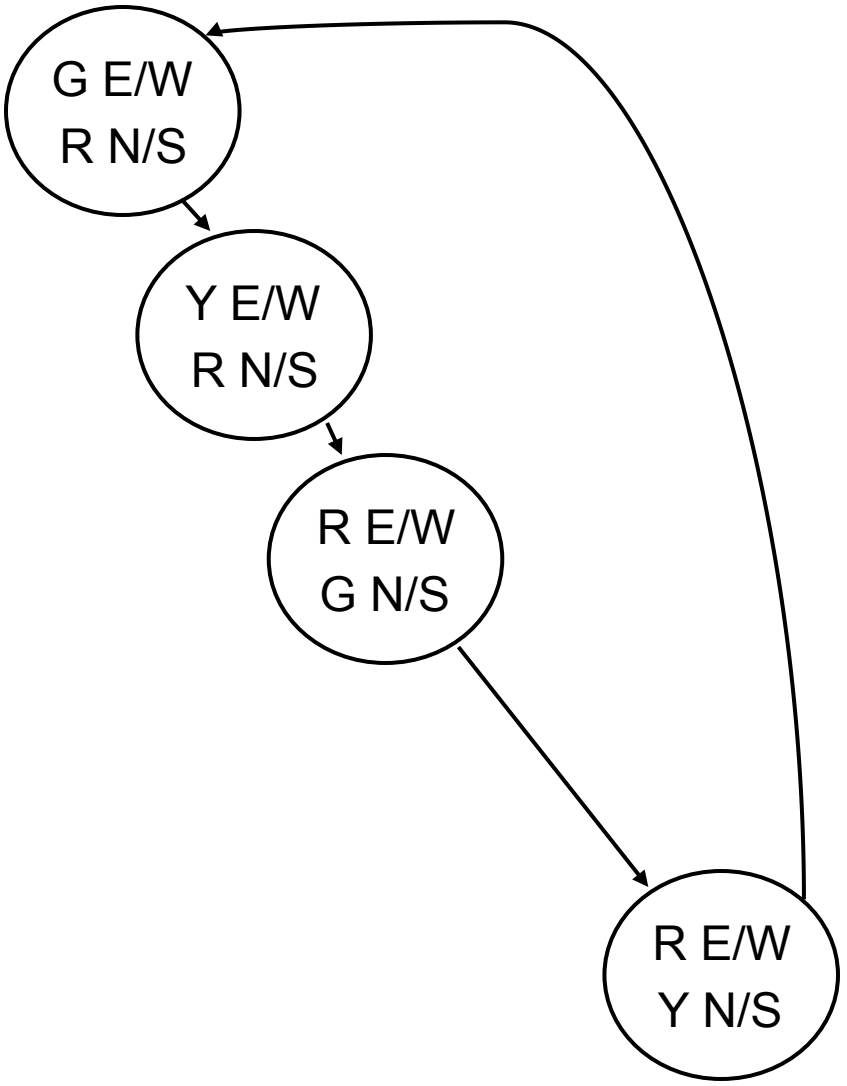
Implementing an FSM

- Use flipflops to represent state (i.e., memory)
- Use combinational logic to compute:
  - output as a function of inputs + state
  - next state as a function of inputs + state



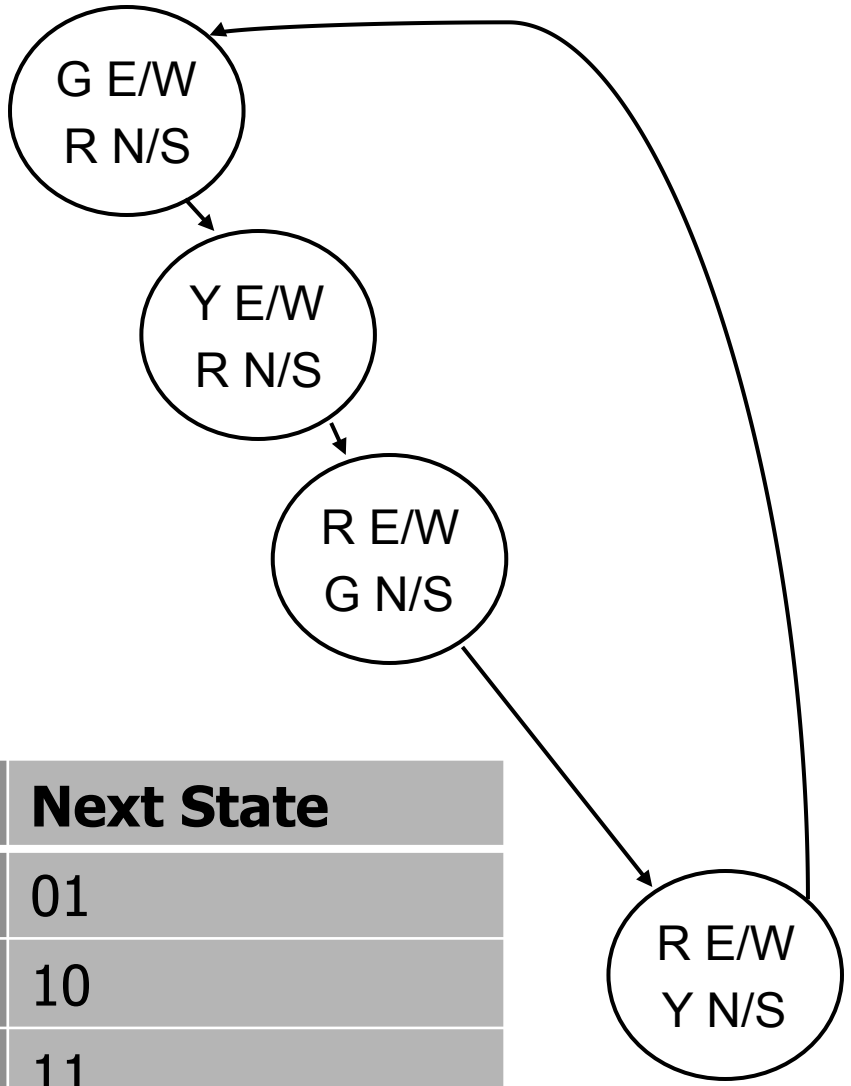
Example: Traffic Light Controller

- 4 states
  - each corresponding to a circle
  - represent states using 2 bits
    - 00, 01, 10, 11
- 4 bits of output
  - 2 bits for east/west light
  - 2 bits for north/south light
    - Red (00)
    - Yellow (01)
    - Green (10)



Example: Traffic Light Controller

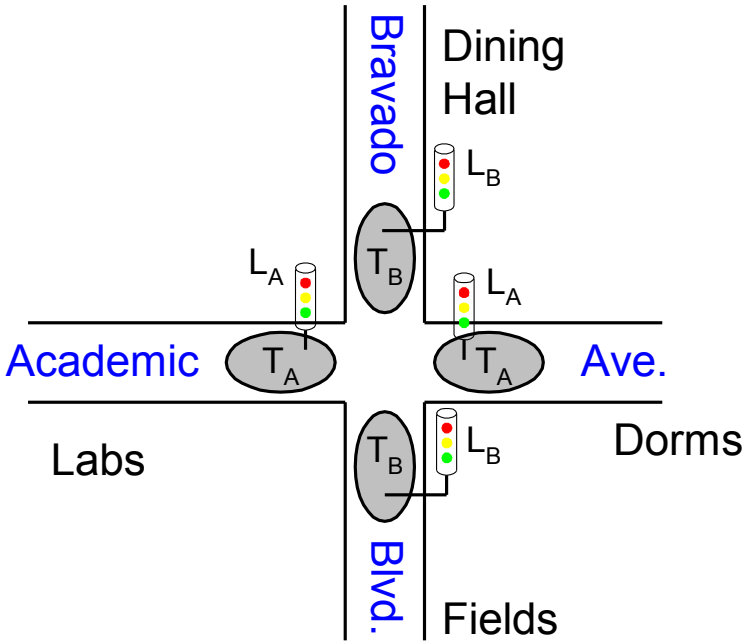
- 4 states
  - 00, 01, 10, 11
  - need 2 flipflops to represent state
- 4 bits of output
  - R (00), Y (01), G (10)
- Truth table:
  - this defines the combinational logic part of the FSM



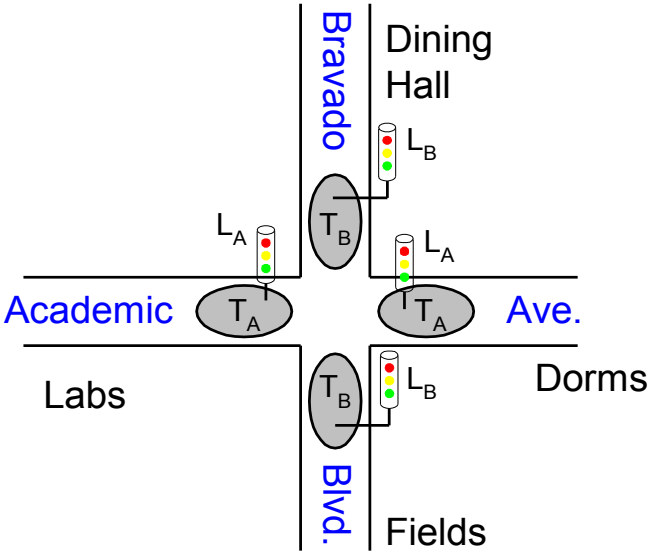
Input (State)	Output	Next State
00	10 00	01
01	01 00	10
10	00 10	11
11	00 01	00

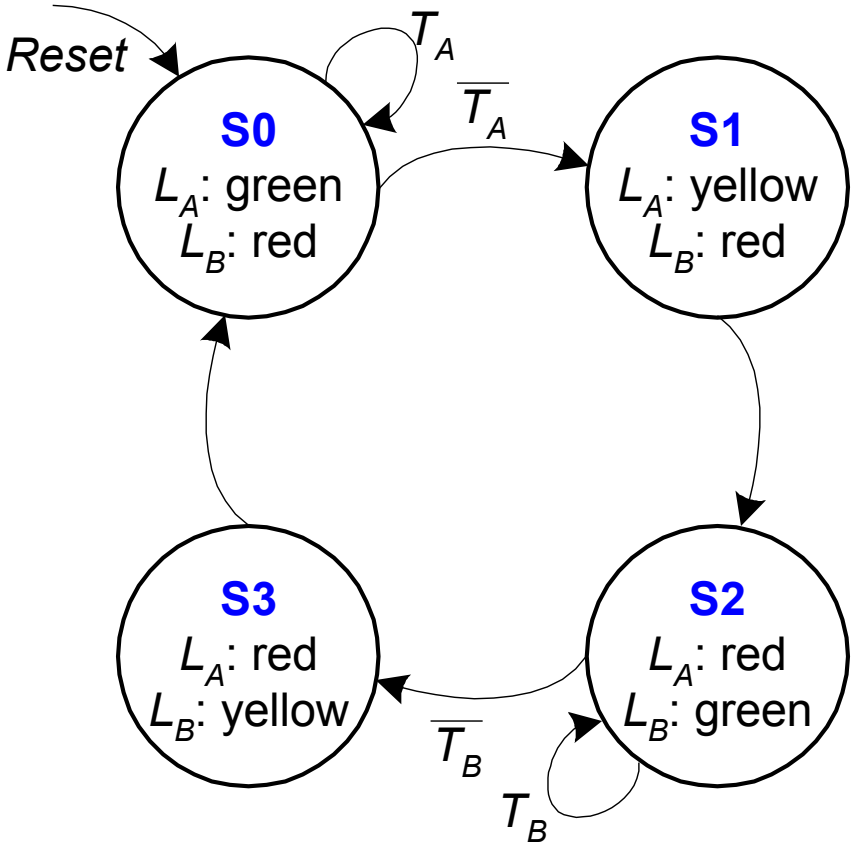
More Complex FSM

- When *reset*, L_A is green and L_B is red
- As long as traffic on Academic (T_A high), keep L_A green
- When T_A goes low, sequence to traffic on Bravado
- Follow same algorithm for Bravado
- Let's say clock is 5 secs. (time for yellow light)



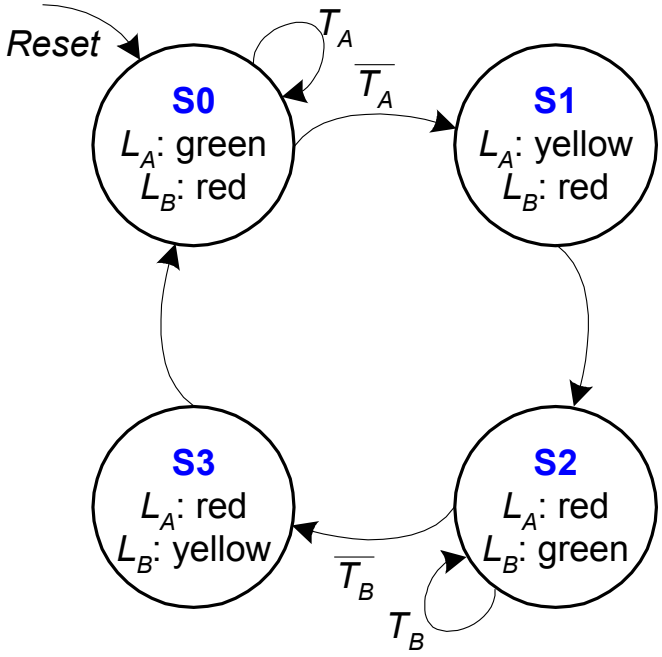
State Transition Diagram

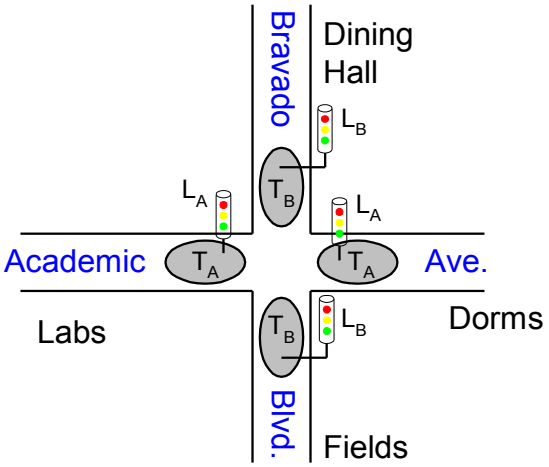




State Transition Table

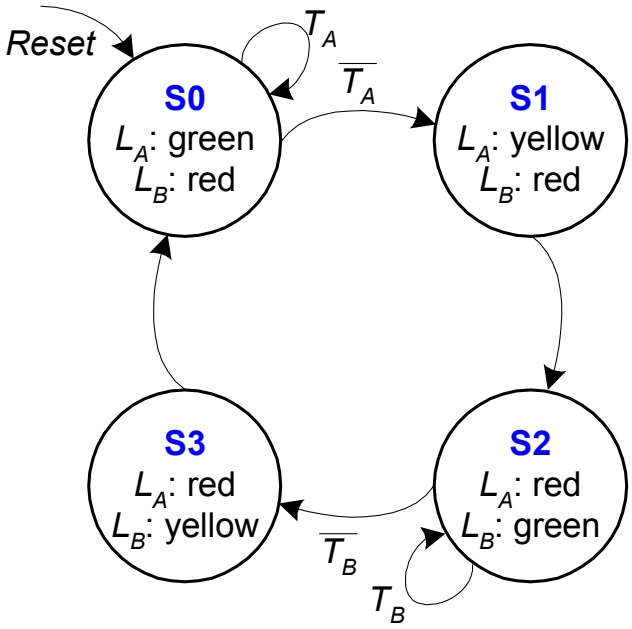
Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0





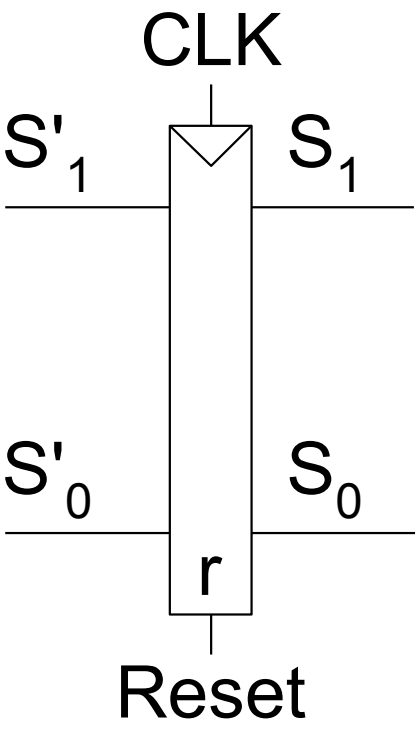
Encoded State Transition Table

State	Encoding
S0	00
S1	01
S2	10
S3	11



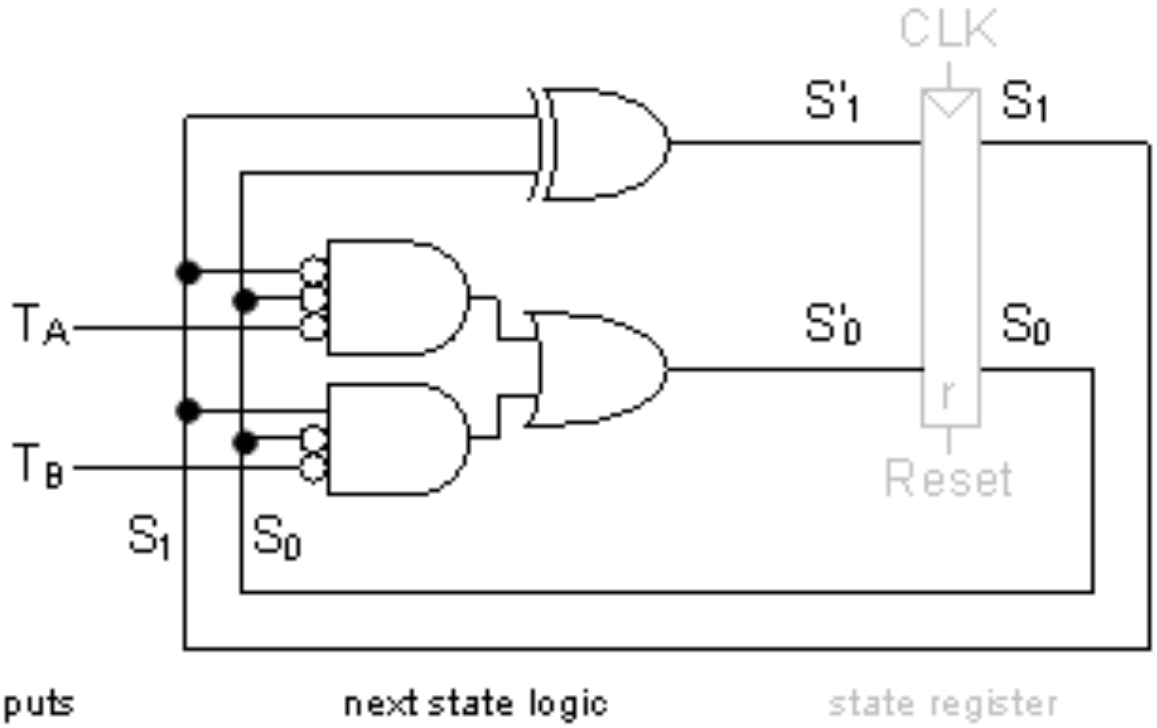
Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

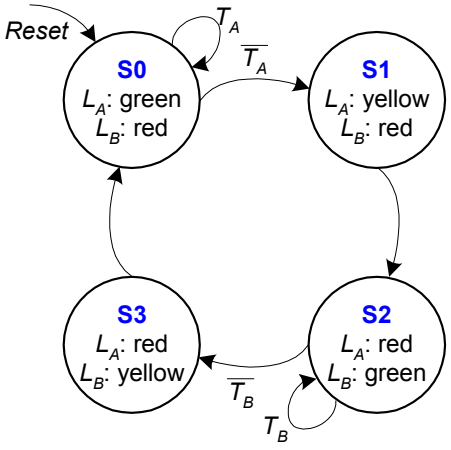
State Register



state register

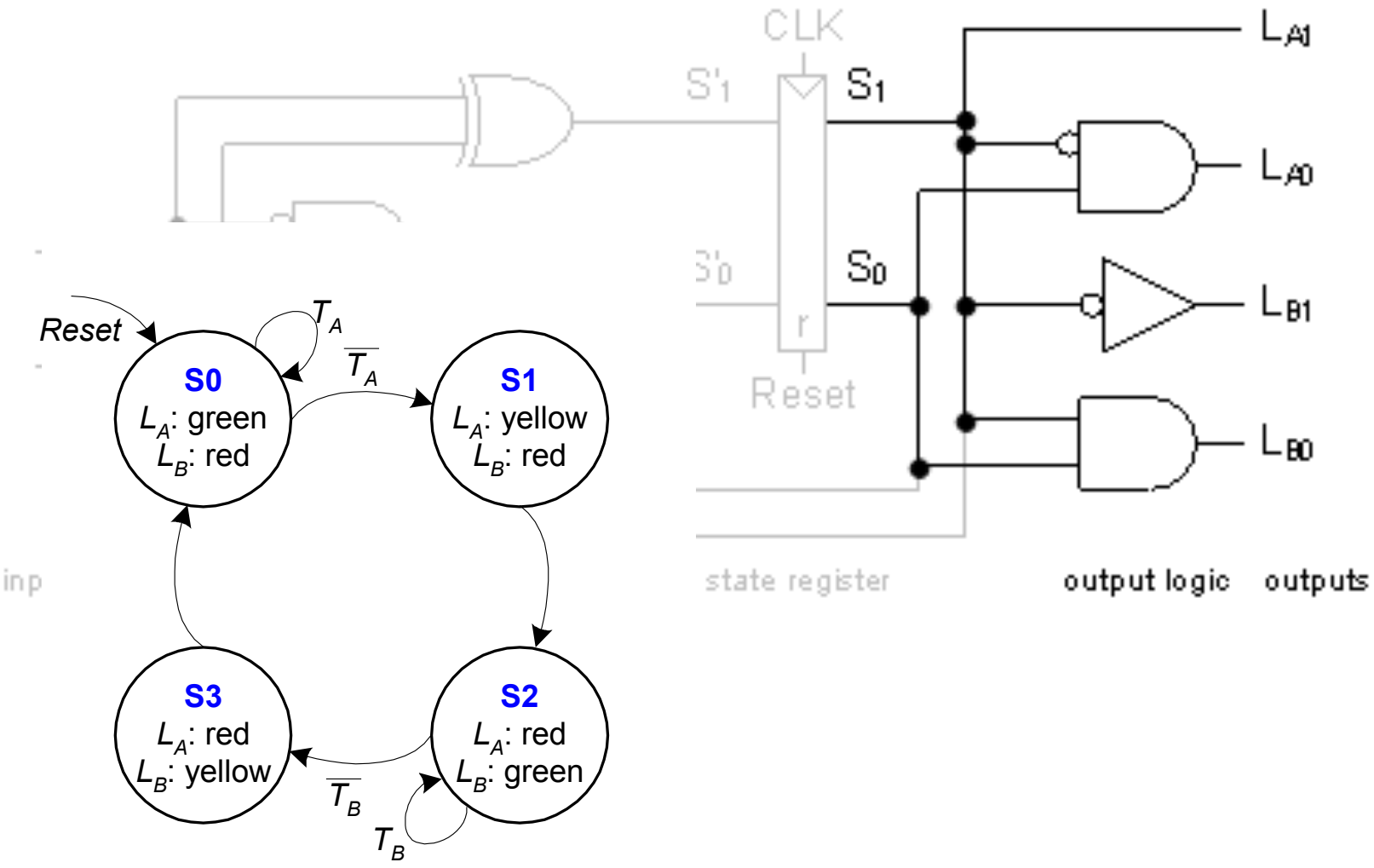
Next State Logic





Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

Output Logic



Summary

- Synchronous Systems
  - Run by the clock (edge)
- Finite State Machines:
  - Implement using flipflops for memory, and combinational logic to compute output, next state