CSEE 3827: Fundamentals of Computer Systems, Spring 2011

5. Finite State Machine Design

Prof. Martha Kim (martha@cs.columbia.edu)
Web: http://www.cs.columbia.edu/~martha/courses/3827/sp11/

Outline (H&H 3.5)

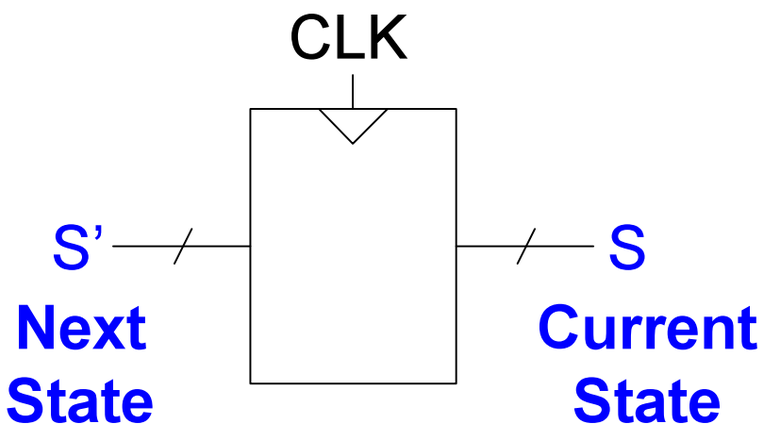
- Finite State Machines
  - Definition
    - Moore
    - Mealy
  - Design procedure
  - Examples

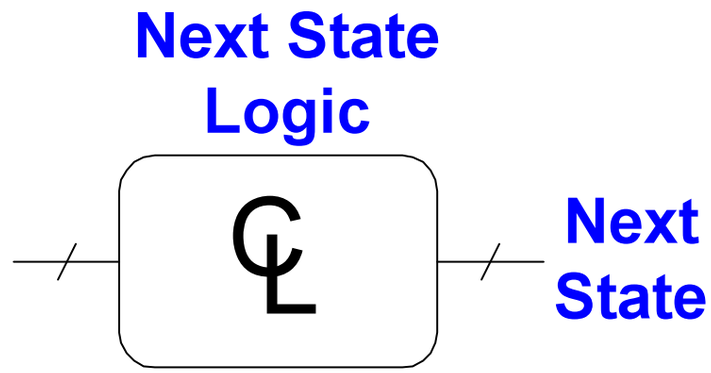
Finite State Machine (FSM)

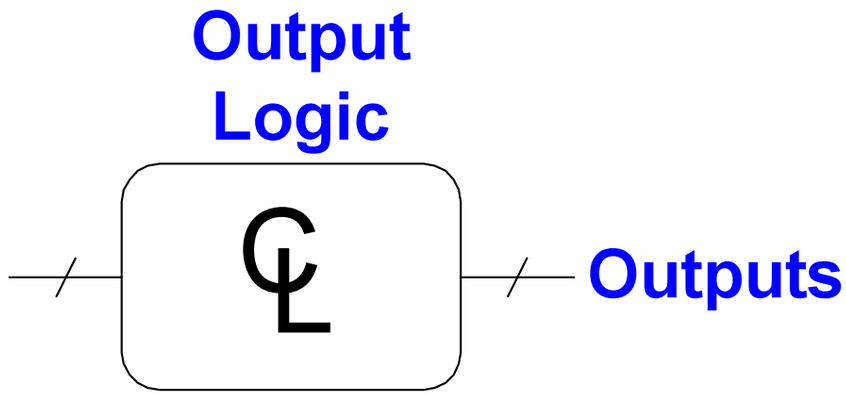
FSM = State register + combinational logic

Stores the current state
and
Loads the next state @ clock edge

Computes the next state
and
Computes the outputs







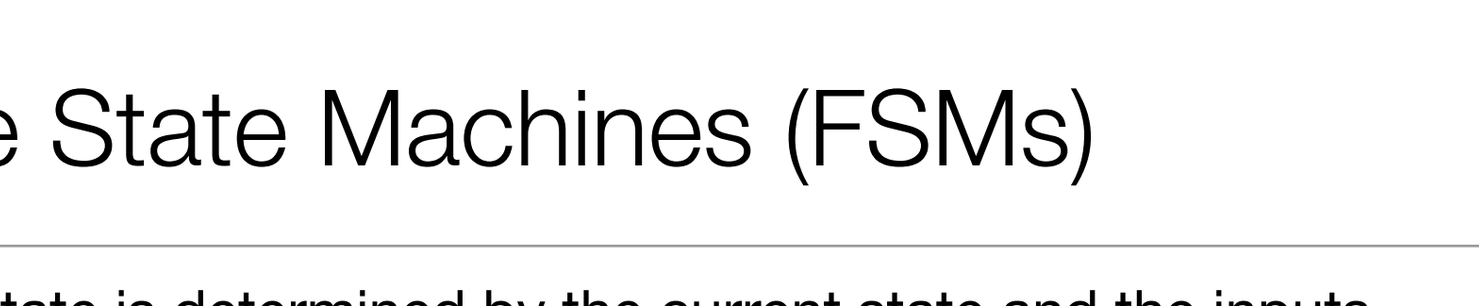
Finite State Machines (FSMs)

- Next state is determined by the current state and the inputs
- Two types of finite state machines differ in the output logic:

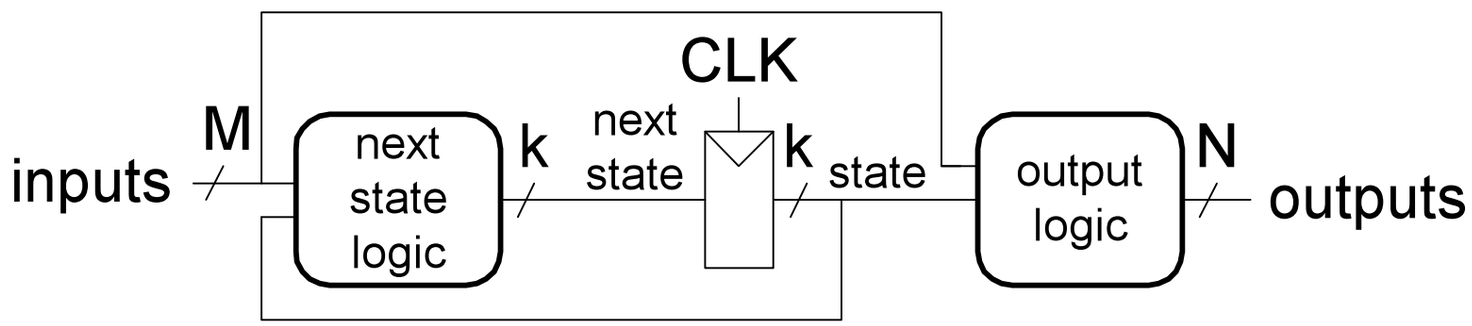
  Moore FSM: outputs depend only on the current state

  Mealy FSM: outputs depend on the current state and the inputs

Moore FSM

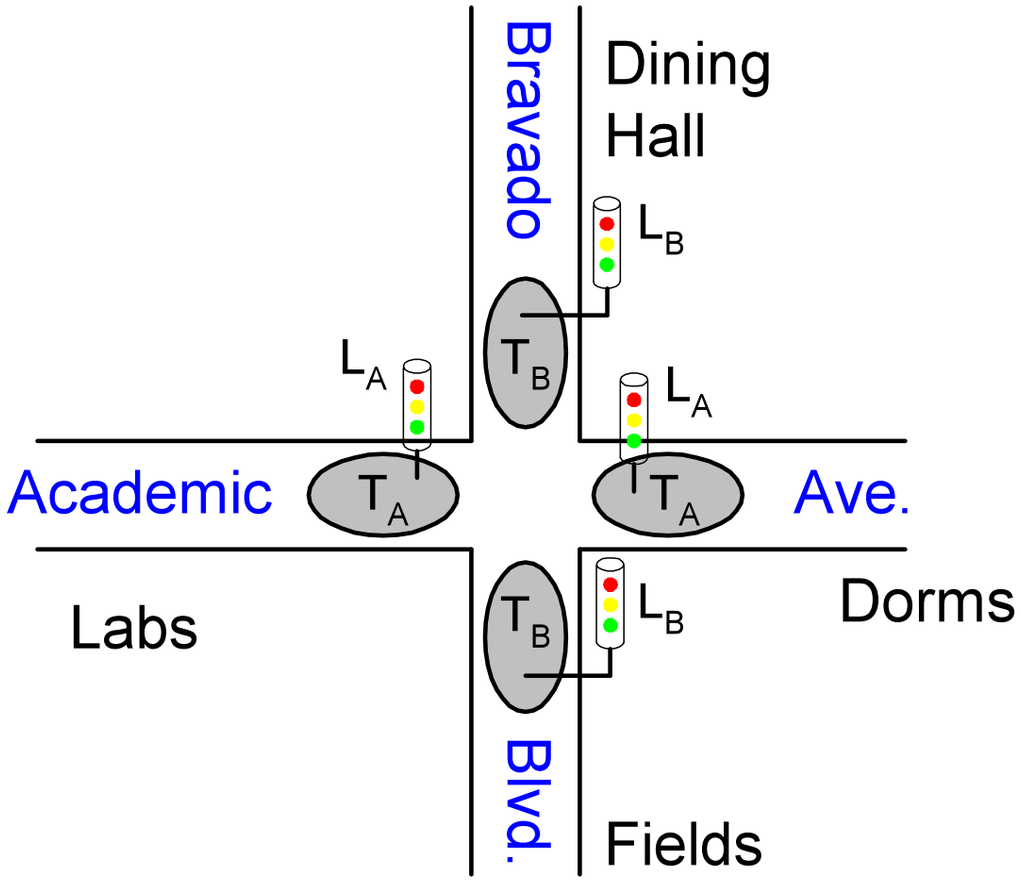


Mealy FSM



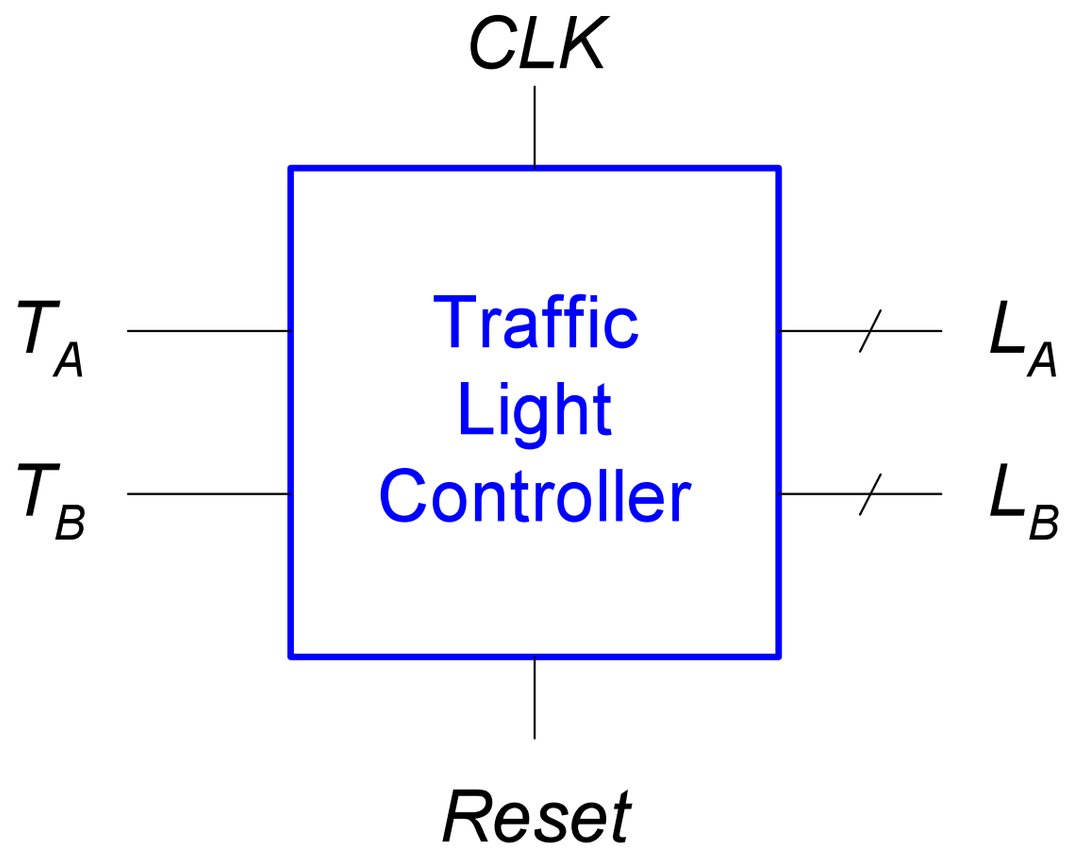
Finite State Machine Example

- Traffic light controller
  - Traffic sensors: TA, TB (TRUE when there is traffic)
  - Lights: LA, LB



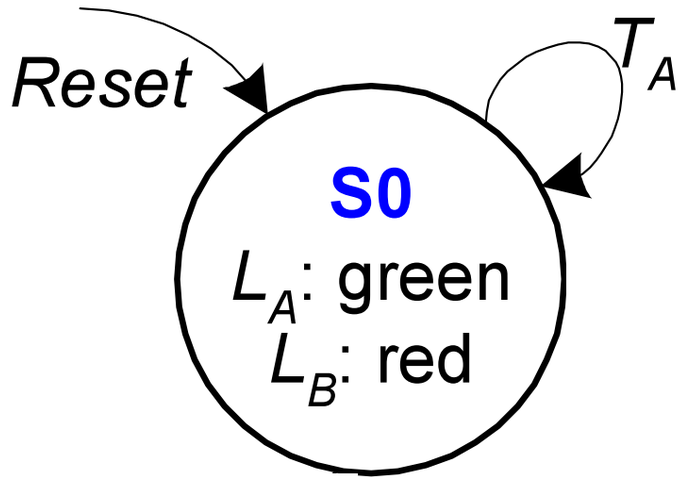
FSM Black Box

- Inputs: CLK, Reset, TA, TB
- Outputs: LA, LB



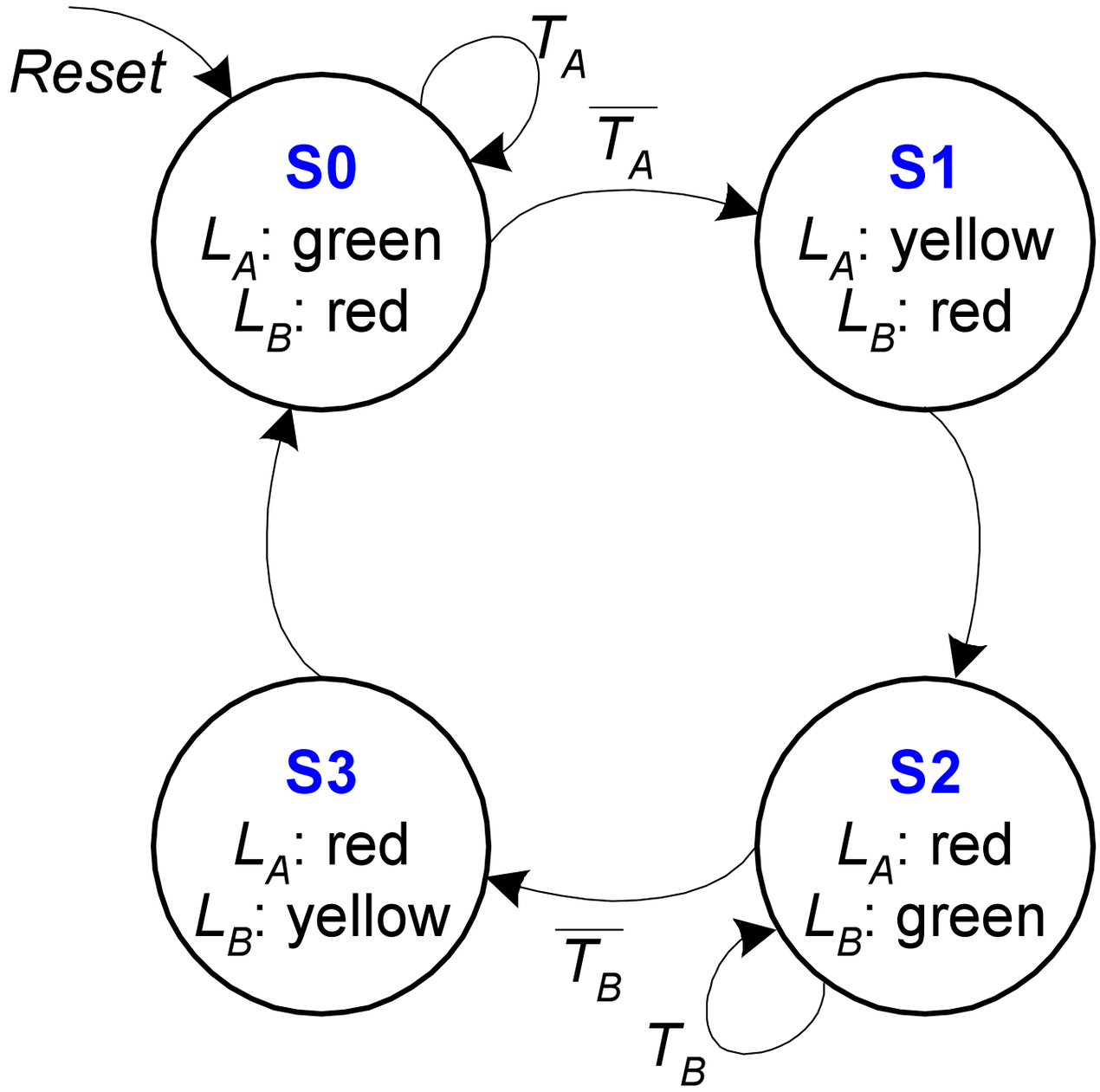
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



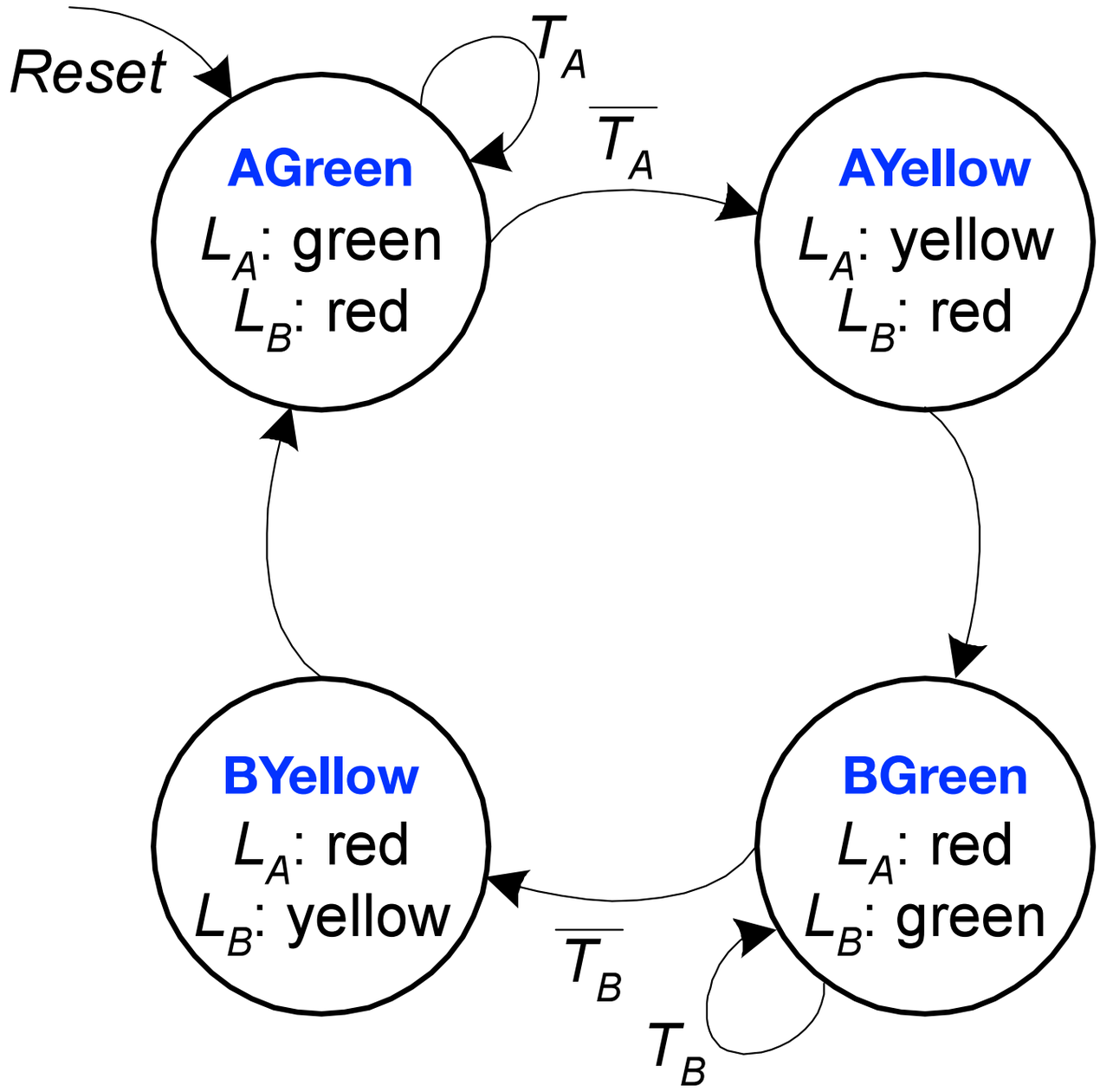
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



FSM State Transition Diagram

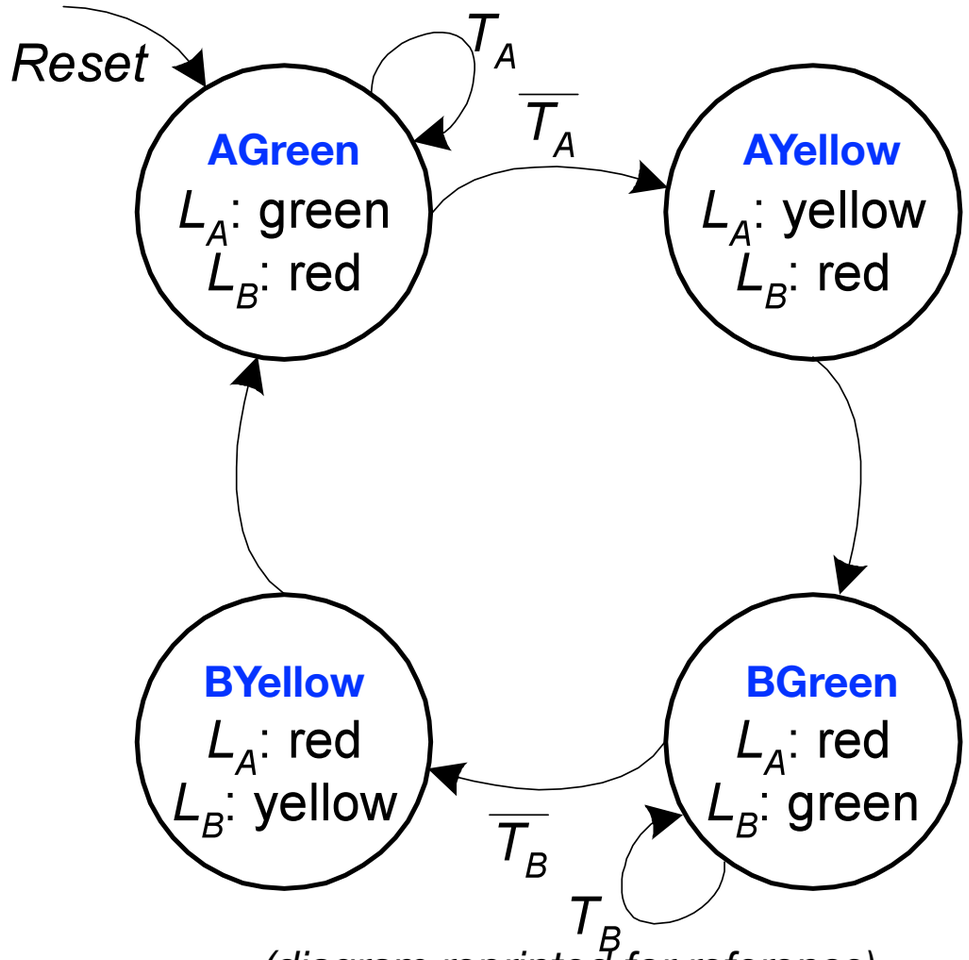
- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



FSM State Transition Table

- State transitions (from diagram) can be rewritten in a state transition table

  (S = current state, S' = next state)



(diagram reprinted for reference)

Current State	Inputs		Next State
S	**TA**	**TB**	**S'**
AGreen	0	X	AYellow
AGreen	1	X	AGreen
AYellow	X	X	BGreen
BGreen	X	0	BYellow
BGreen	X	1	BGreen
BYellow	X	X	AGreen

FSM Encoded State Transition Table

- After selecting a state encoding the symbolic states in the transition table can be annotated with actual state / next state bits

State	Encoding S1	Encoding S0
AGreen	0	0
AYellow	0	1
BGreen	1	0
BYellow	1	1

Current State S	Encoded Current State S1	Encoded Current State S0	Inputs TA	Inputs TB	Next State S’	Encoded Next State S1’	Encoded Next State S0’
AGreen	0	0	0	X	AYellow	0	1
AGreen	0	0	1	X	AGreen	0	0
AYellow	0	1	X	X	BGreen	1	0
BGreen	1	0	X	0	BYellow	1	1
BGreen	1	0	X	1	BGreen	1	0
BYellow	1	1	X	X	AGreen	0	0

- One can then compute the next state logic

S1’= S1 XOR S0
S0’ = `S1`S0`TA + S1`S0`TB

FSM Output Table

- FSM output logic is computed in much the same manner as the next state logic

- Because this is a Moore machine, outputs are a function of the current state only (were it a Mealy output would be a function of current state + inputs)

output encoding

Output	Encoding	
Green	0	0
Yellow	0	1
Red	1	0

output truth table

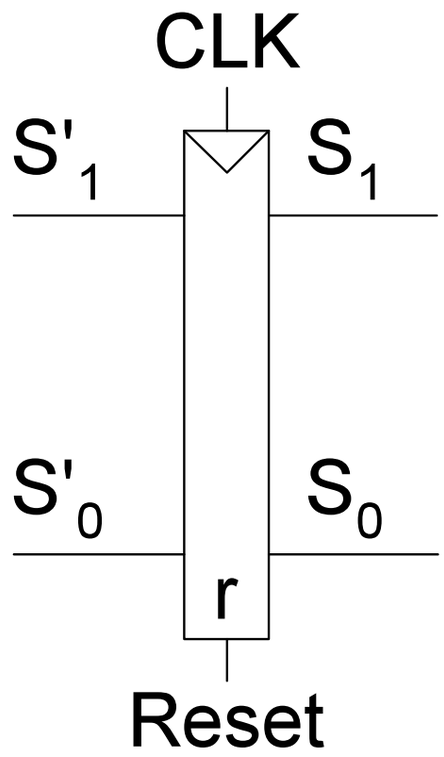
	State		LA		LB	
State	**S1**	**S0**	**LA1**	**LA0**	**LB1**	**LB0**
AGreen	0	0	0	0	1	0
AYellow	0	1	0	1	1	0
BGreen	1	0	1	0	0	0
BYellow	1	1	1	0	0	1

red light

- Compute output bits as function of state bits

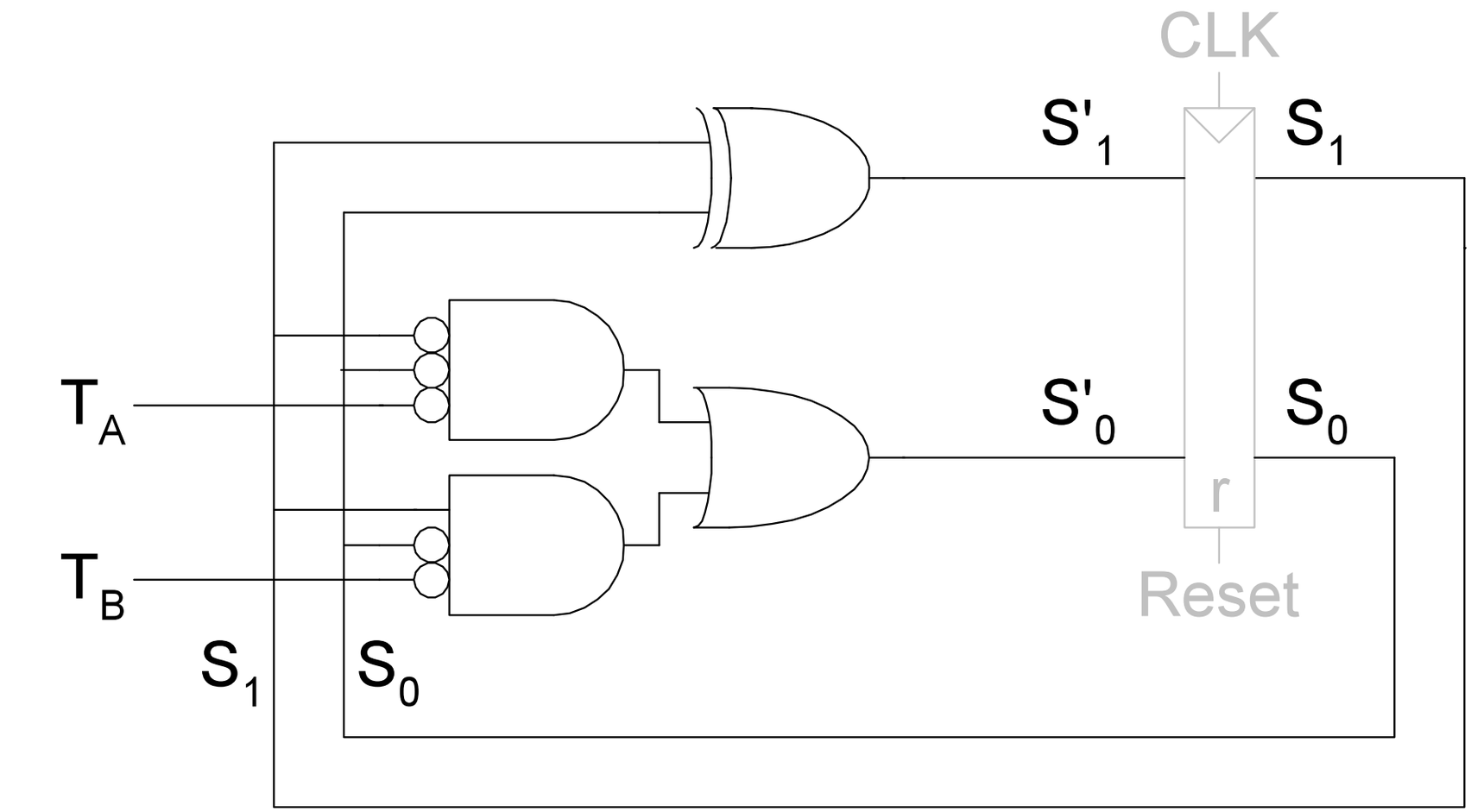
LA1 = S1; LA0 = S1`S0
LB1 = `S1; LB0 = S1S0

FSM Schematic: State Register



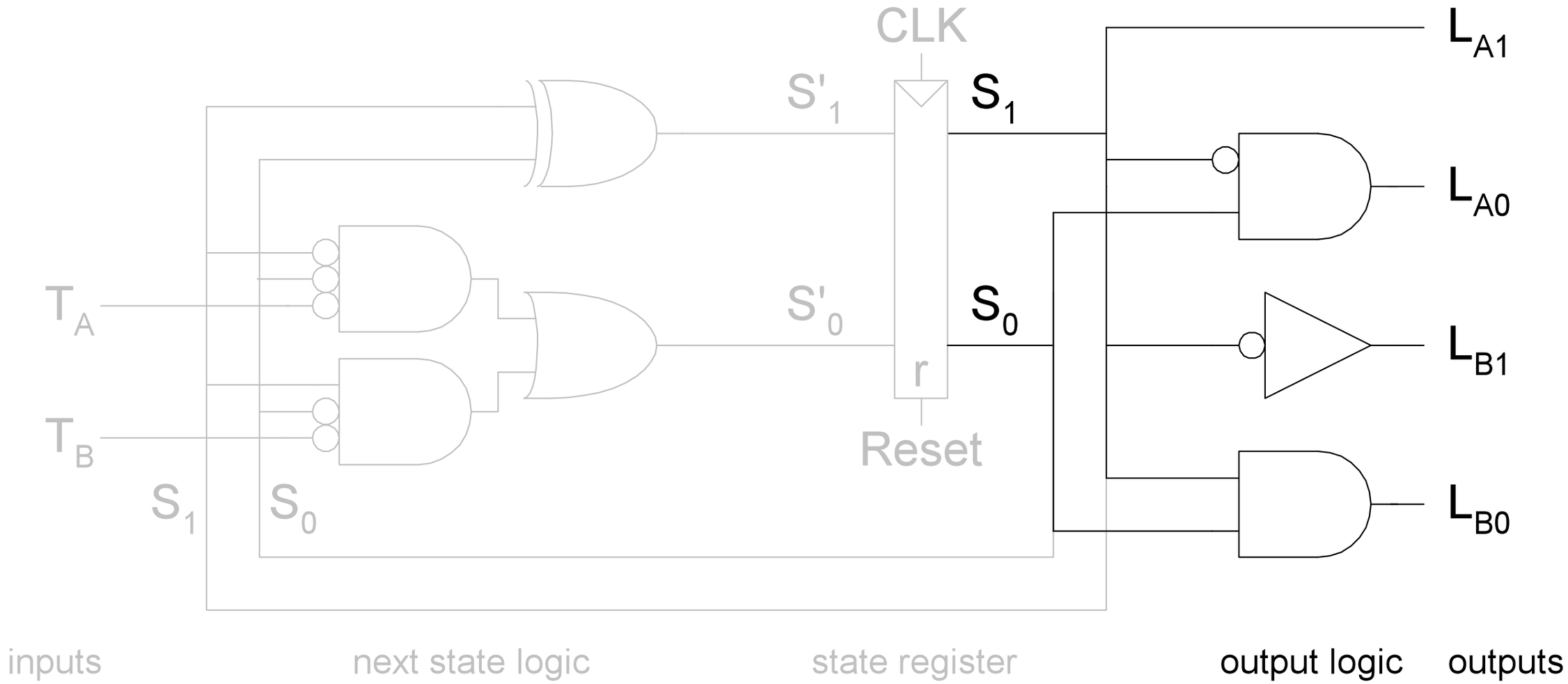
state register

FSM Schematic: Next State Logic



inputs next state logic state register

FSM Schematic: Output Logic



What does the Reset signal do to this machine?

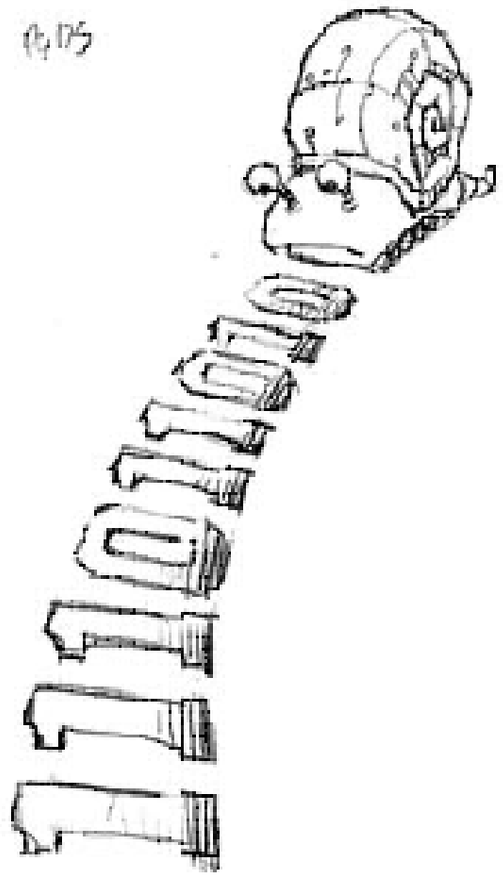
S = 00 = AGreen = A gets green light, B gets red light

FSM State Encoding

- Binary encoding: i.e., for four states, 00, 01, 10, 11
- One-hot encoding
    - One state bit per state
    - Only one state bit is HIGH at once
    - I.e., for four states, 0001, 0010, 0100, 1000
    - Requires more flip-flops
    - Often next state and output logic is simpler
- Sometimes a semantically meaningful encoding makes the most sense (e.g., a faucet controller with a volume state and a temperature state)

Moore v. Mealy FSM

Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last four digits it has crawled over are 1101. Design Moore and Mealy FSMs of the snail's brain.

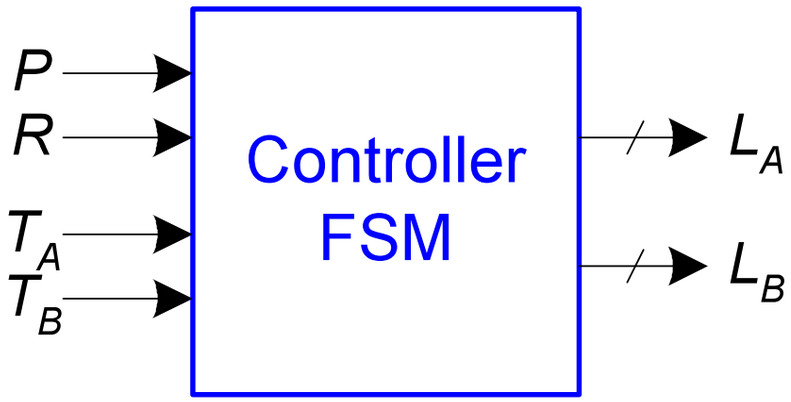


Factoring State Machines

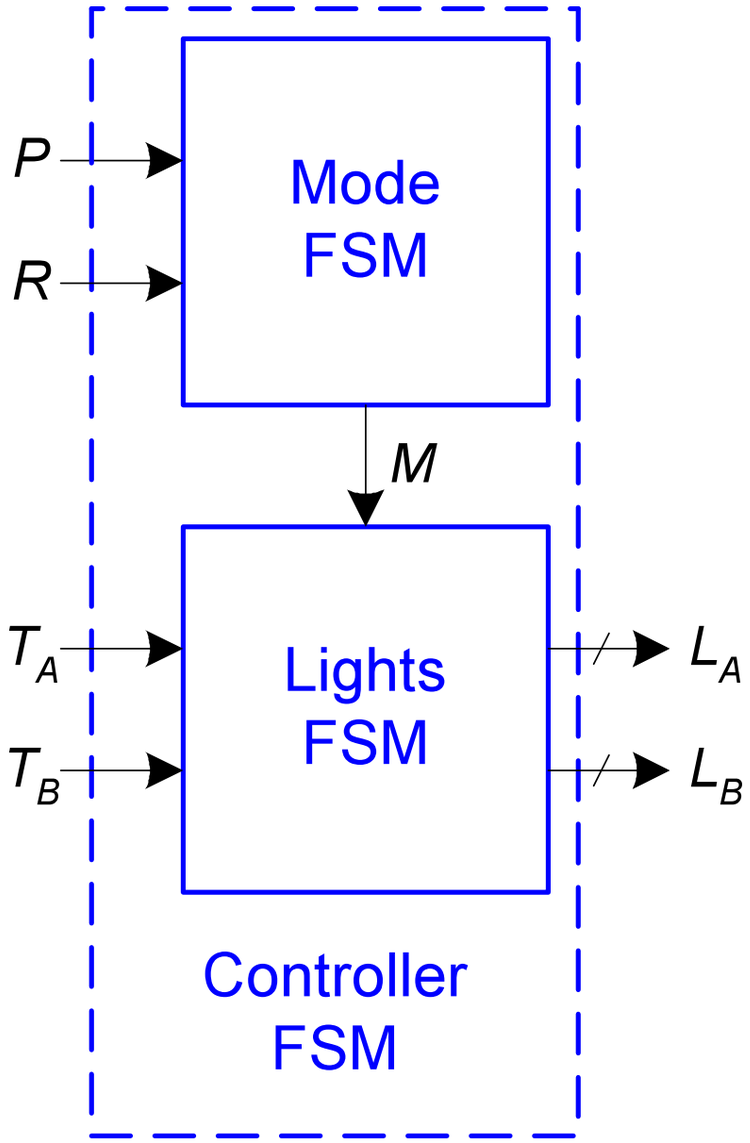
- Break complex FSMs into smaller interacting FSMs
- Example: Modify the traffic light controller to have a Parade Mode.
  - The FSM receives two more inputs: P, R
  - When P = 1, it enters Parade Mode and the Bravado Blvd. light stays green.
  - When R = 1, it leaves Parade Mode

Parade FSM

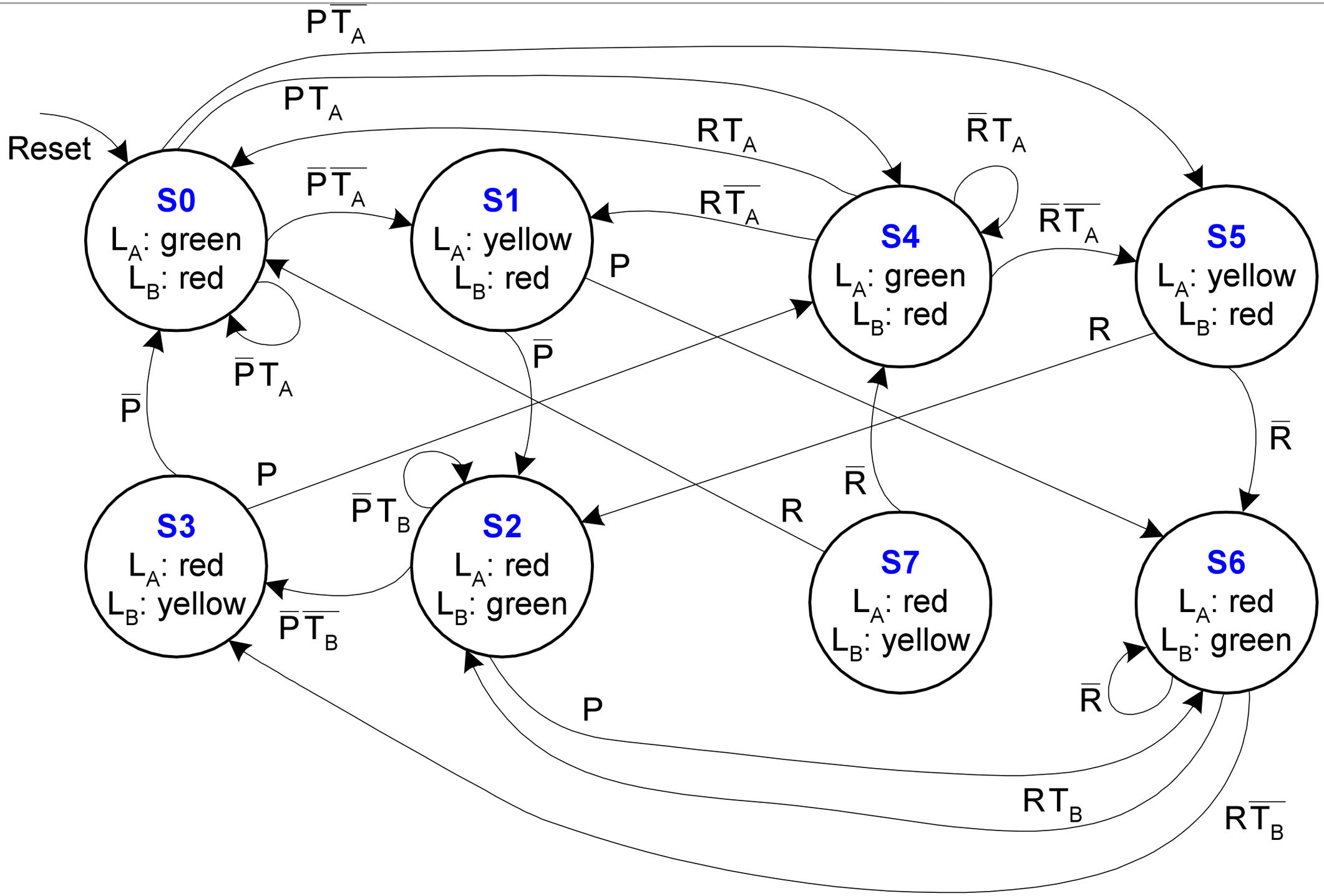
- Unfactored FSM



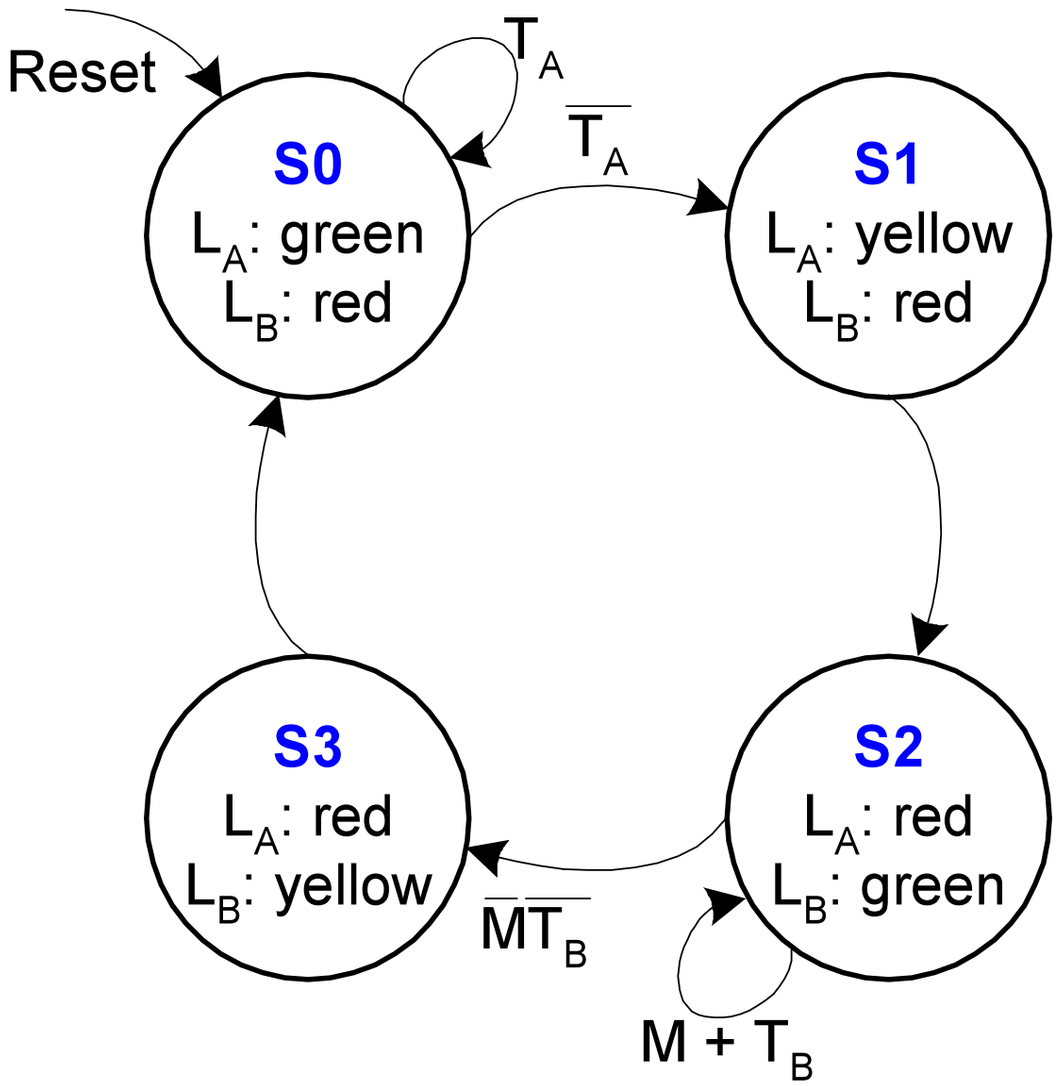
- Factored FSM



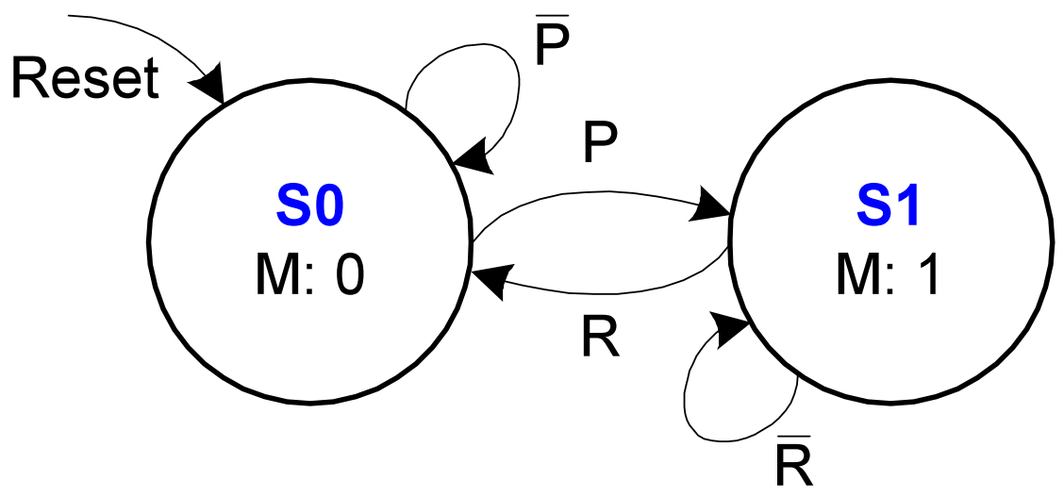
Unfactored FSM State Transition Diagram



Factored FSM State Transition Diagram



Lights FSM



Mode FSM

FSM Design Procedure

- Identify the inputs and outputs
- Sketch a state transition diagram
- Write a state transition table
- Select state encodings
- For a Moore machine:

  Rewrite the state transition table with the selected state encodings

  Write the output table

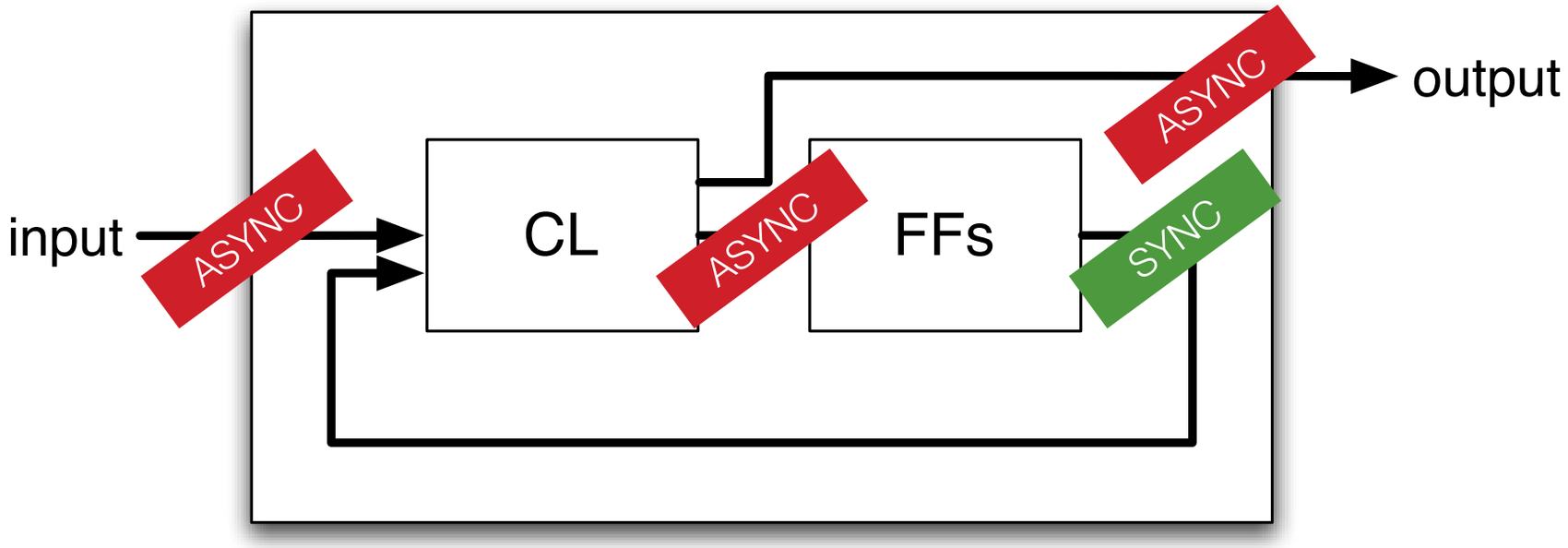
- For a Mealy machine:

  Rewrite the combined state transition and output table with the selected state encodings

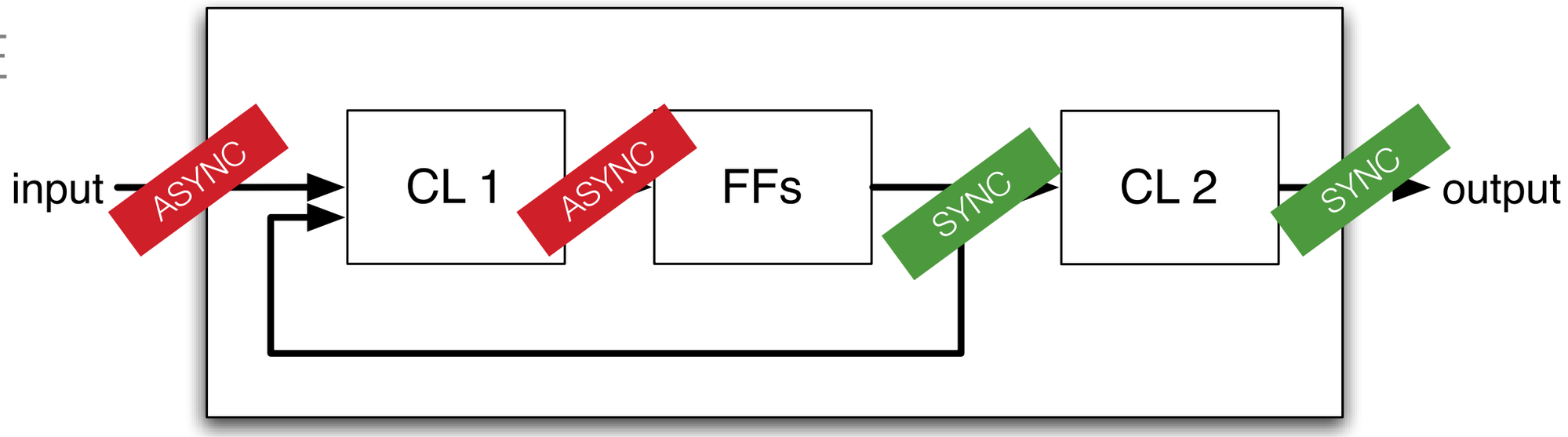
- Write Boolean equations for the next state and output logic
- Sketch the circuit schematic

FSM timing characteristics

MEALY



MOORE



Advanced FSM design and implementation

Unused states: *extra state encodings (e.g., using 3 FFs to represent 6 states leaves 2 unused states) can be treated as "don't care" values and used to simplify the combinational logic*

State minimization: *two states are equivalent if they transition to the same or equivalent states on the same inputs (while producing the same outputs in the case of a Mealy machine)*