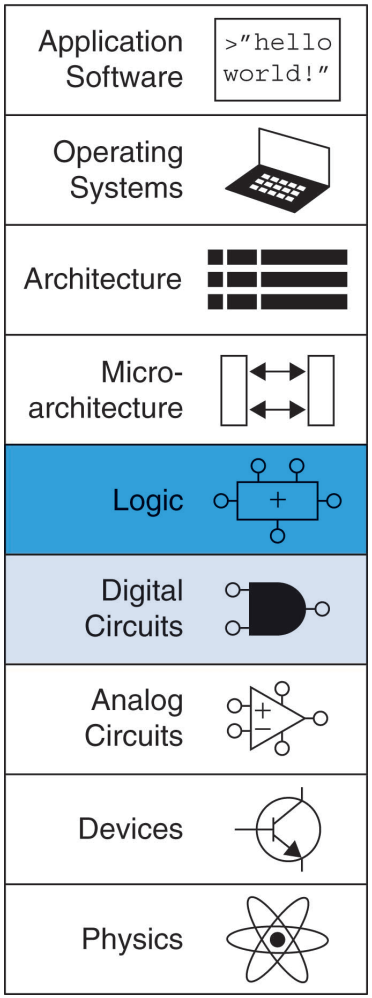
# E85 Digital Design & Computer Engineering

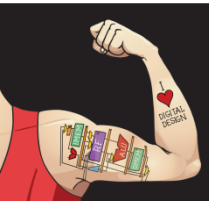

# Lecture 4:
## Finite State Machines

HARVEY MUDD COLLEGE

# Lecture 4

- **Finite State Machines**



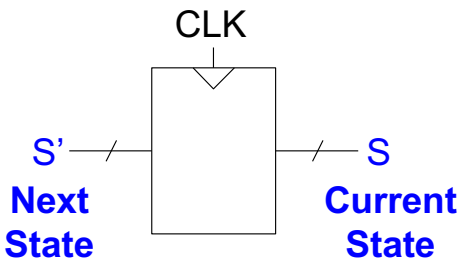



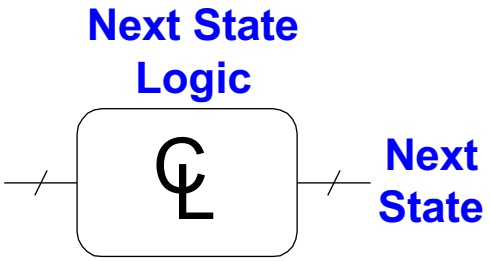
# Synchronous Sequential Logic Design

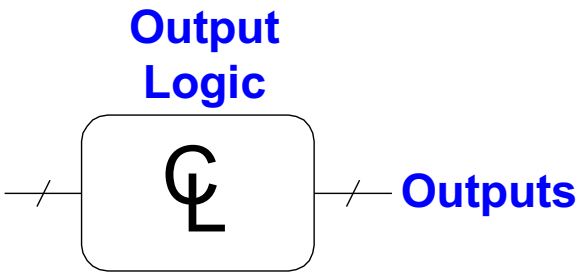
- Breaks cyclic paths by **inserting registers**
- Registers contain **state** of the system
- State changes at clock edge: system **synchronized** to the clock
- **Rules** of synchronous sequential circuit composition:
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit element is a register
  - All registers receive the same clock signal
  - Every cyclic path contains at least one register
- Two common synchronous sequential circuits
  - Finite State Machines (FSMs)
  - Pipelines

# Finite State Machine (FSM)

- Consists of:
  - **State register**
    - Stores current state
    - Loads next state at clock edge
  - **Combinational logic**
    - Computes the next state
    - Computes the outputs



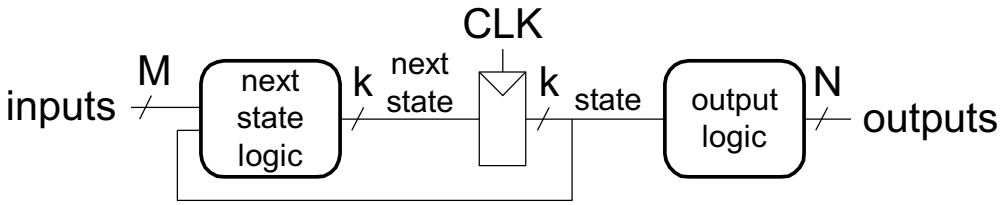



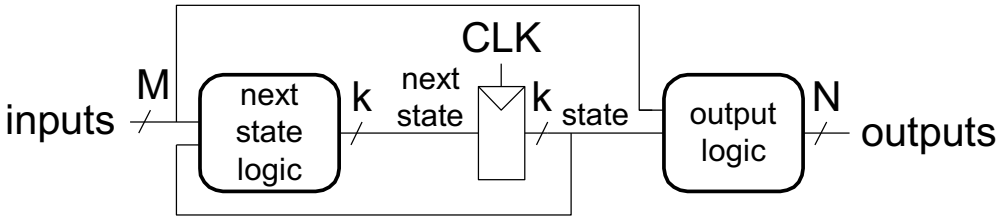
# Finite State Machines (FSMs)

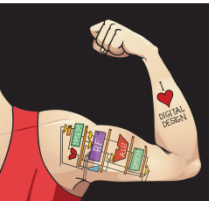
- **Next state** determined by current state and inputs
- Two types of finite state machines differ in **output logic**:
  - **Moore FSM:** outputs depend only on current state
  - **Mealy FSM:** outputs depend on current state *and* inputs

Moore FSM



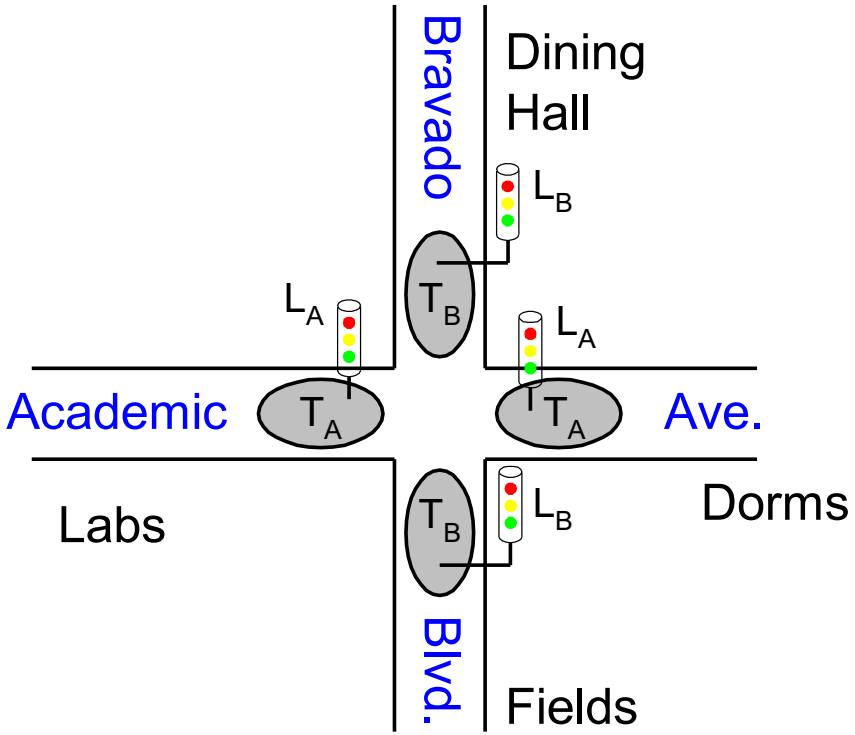

Mealy FSM



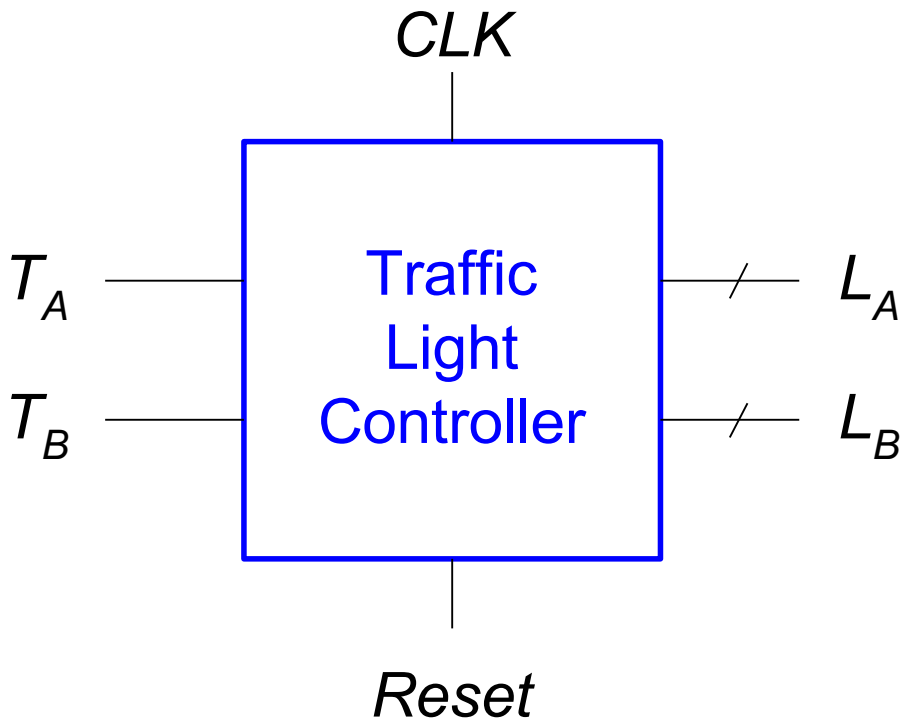
# FSM Example

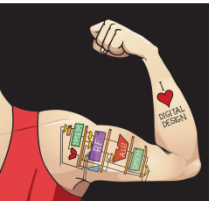
- Traffic light controller
  - Traffic sensors: $T_A$, $T_B$ (TRUE when there's traffic)
  - Lights: $L_A$, $L_B$



# FSM Black Box

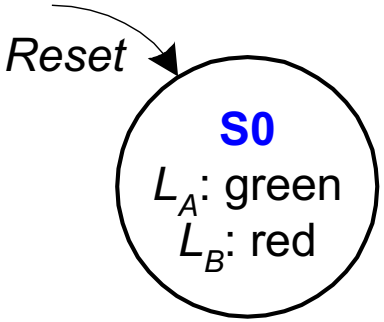
- Inputs: *CLK*, *Reset*, $T_A$, $T_B$
- Outputs: $L_A$, $L_B$

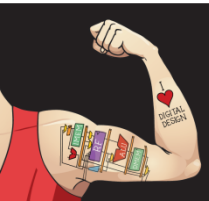


# FSM State Transition Diagram

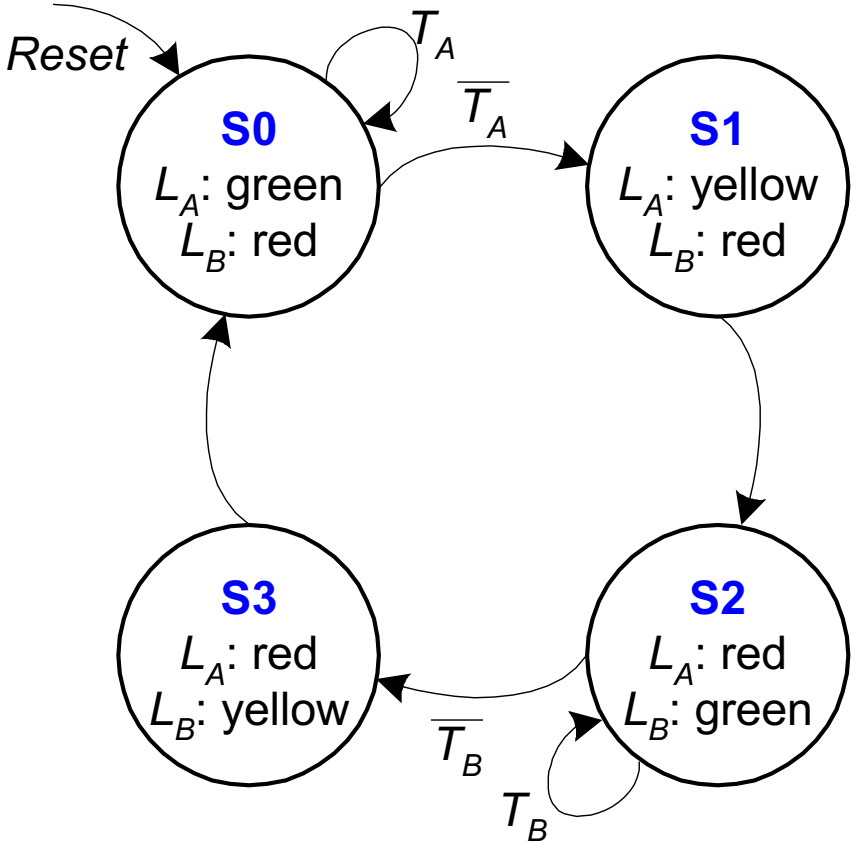
- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs

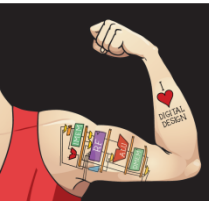


# FSM State Transition Diagram

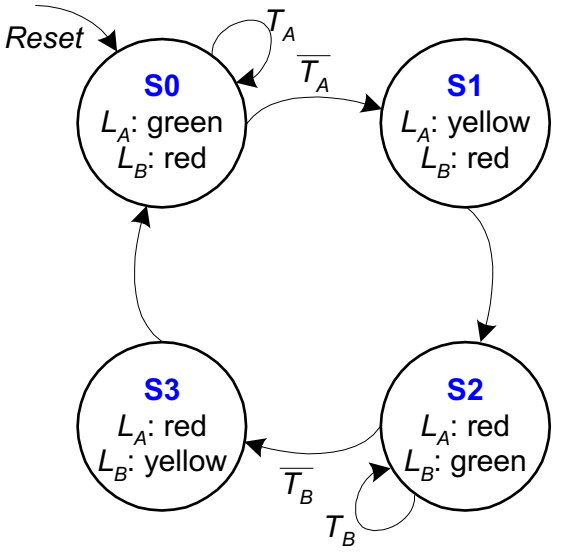
- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs

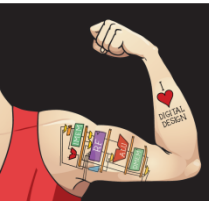


# FSM State Transition Table

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

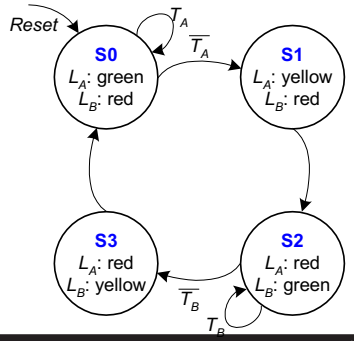


# FSM Encoded State Transition Table

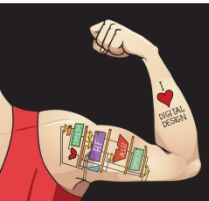
| Current State | | Inputs | | Next State | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $T_A$ | $T_B$ | $S'_1$ | $S'_0$ |
| 0 | 0 | 0 | X | | |
| 0 | 0 | 1 | X | | |
| 0 | 1 | X | X | | |
| 1 | 0 | X | 0 | | |
| 1 | 0 | X | 1 | | |
| 1 | 1 | X | X | | |

| State | Encoding |
|---|---|
| S0 | 00 |
| S1 | 01 |
| S2 | 10 |
| S3 | 11 |




$S'_1 =$

$S'_0 =$



# FSM Output Table

| Current State | | Outputs | | | |
|---|---|---|---|---|---|
| $S_1$ | $S_0$ | $L_{A1}$ | $L_{A0}$ | $L_{B1}$ | $L_{B0}$ |
| 0 | 0 | | | | |
| 0 | 1 | | | | |
| 1 | 0 | | | | |
| 1 | 1 | | | | |

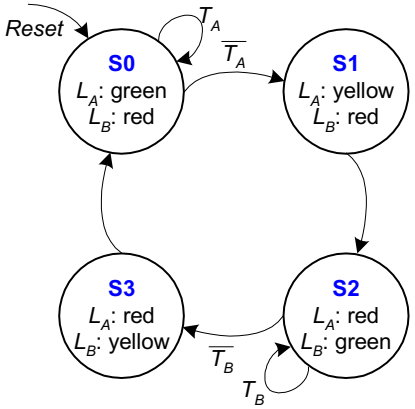
$L_{A1} =$

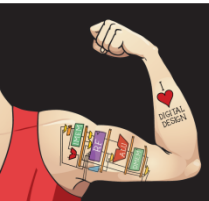
$L_{A0} =$

$L_{B1} =$

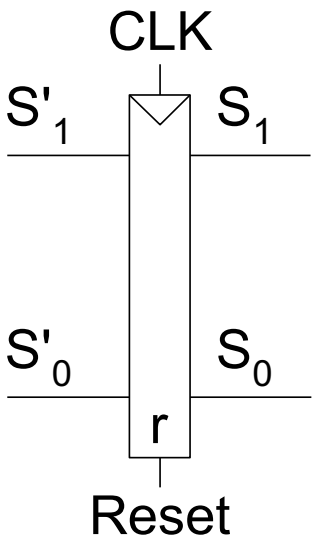
$L_{B0} =$

| Output | Encoding |
|---|---|
| green | 00 |
| yellow | 01 |
| red | 10 |

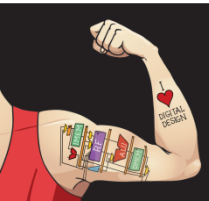


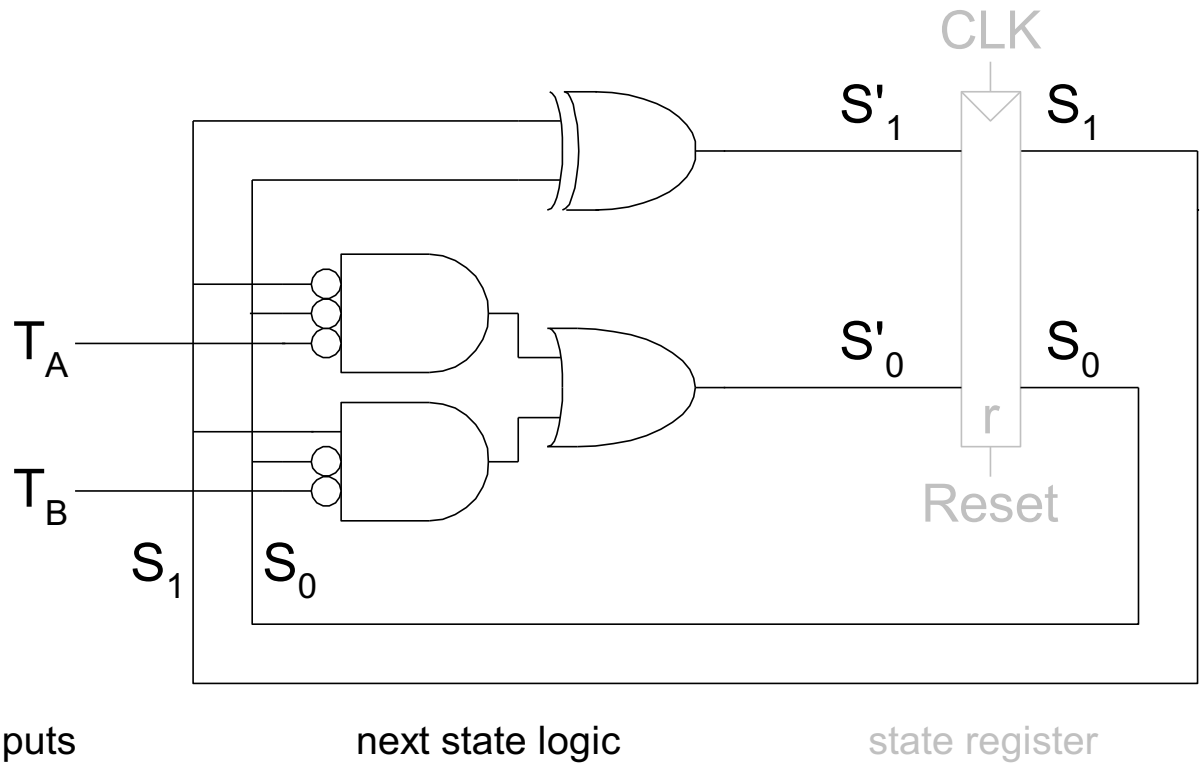
# FSM Schematic: State Register



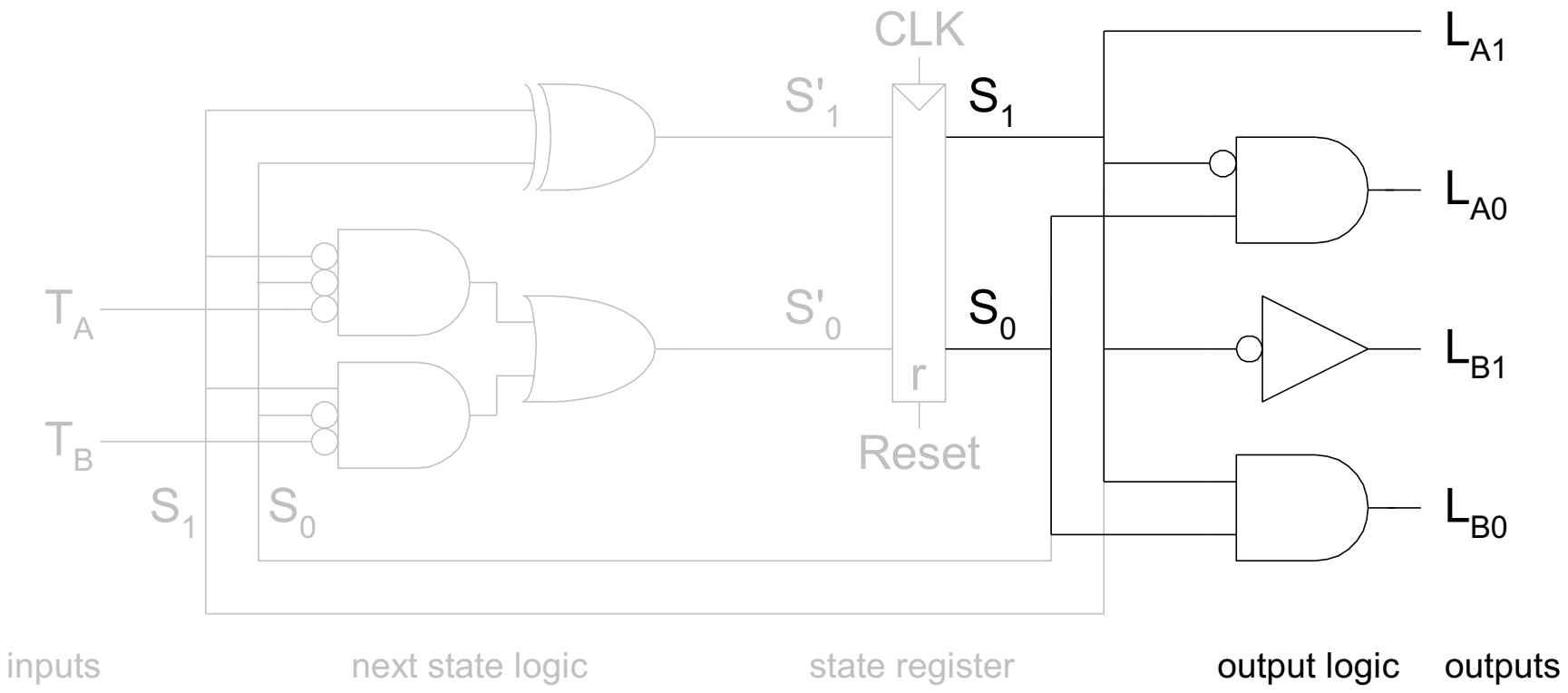

state register



# FSM Schematic: Next State Logic

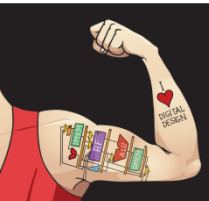


# FSM Schematic: Output Logic



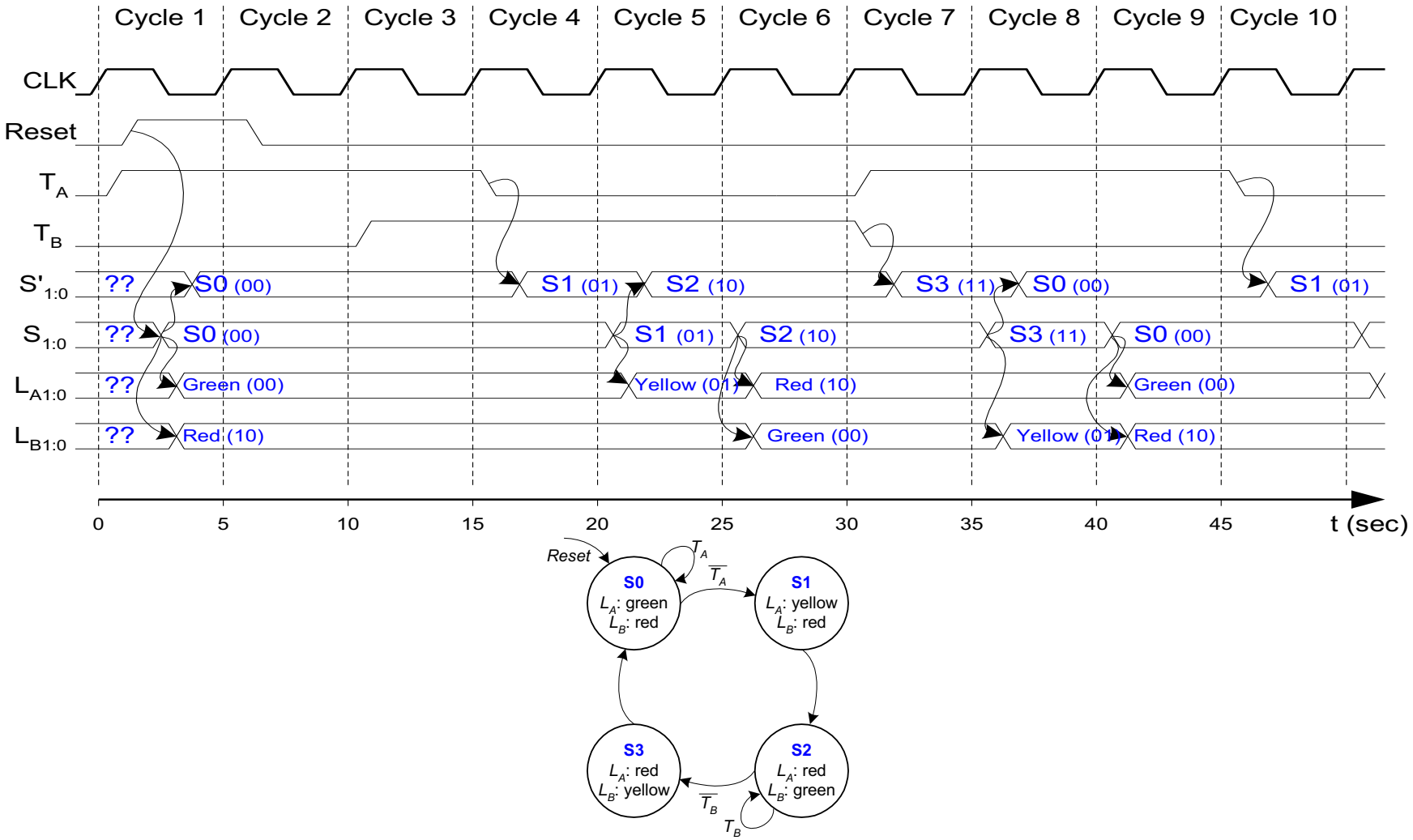
# SystemVerilog Description

```
module trafficFSM(input  logic       clk, reset,
                  input  logic       ta, tb,
                  output logic [1:0] la, lb);
  logic [1:0] state, nextstate, sb;
  logic       tab, tbb, p1, p2;

  // state register
  flopr #(2) statereg(clk, reset, nextstate, state);

  // next state logic
  not n0(sb[0], s[0]);
  not n1(sb[1], s[1]);
  not n2(tab, ta);
  not n3(tbb, tb);
  xor x1(nextstate[1], state[0], state[1]);
  and a1(p1, sb[0], sb[1], tab);
  and a2(p2, sb[0], s[1], tbb);
  or  o1(nextstate[0], p1, p2);

  // output logic
  buf b1(la[1], s[1]);
  and a3(la[0], s[0], sb[1]);
  inv n4(lb[1], s[1]);
  and a4(lb[0], s[0], s[1]);
endmodule
```



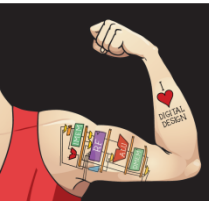
- Multi-input XOR: Odd parity

# FSM Timing Diagram



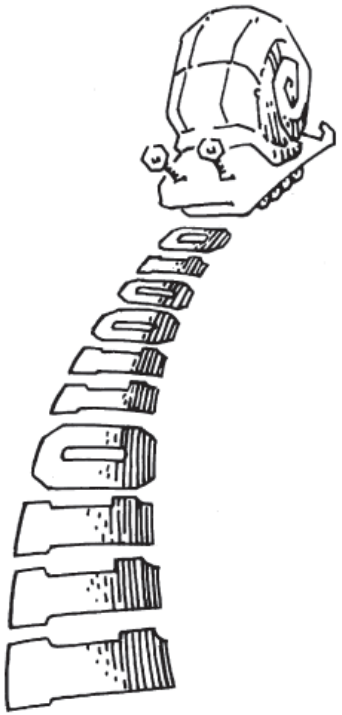
# FSM State Encoding

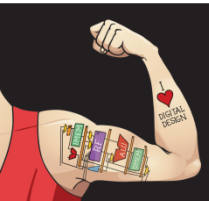
- **Binary** encoding:
  - i.e., for four states, 00, 01, 10, 11
- **One-hot** encoding
  - One state bit per state
  - Only one state bit HIGH at once
  - i.e., for 4 states, 0001, 0010, 0100, 1000
  - Requires more flip-flops
  - Often next state and output logic is simpler



# Moore vs. Mealy FSM

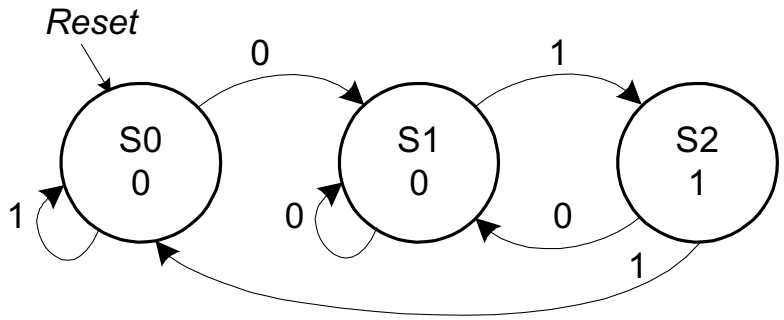
- Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last two digits it has crawled over are 01. Design Moore and Mealy FSMs of the snail's brain.



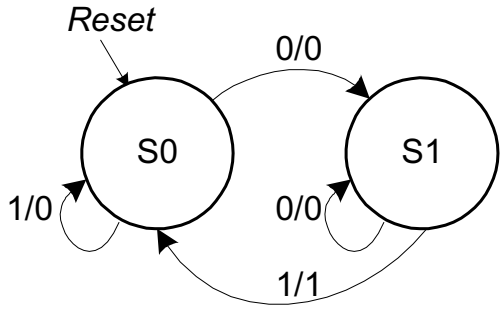


# State Transition Diagrams

## Moore FSM




## Mealy FSM



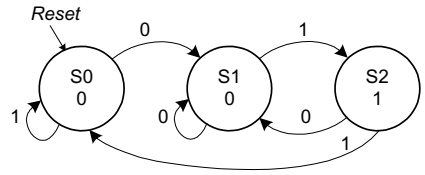

Mealy FSM: arcs indicate input/output

# Moore FSM State Transition Table

| Current State $S_1$ | Current State $S_0$ | Inputs $A$ | Next State $S'_1$ | Next State $S'_0$ |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |

| State | Encoding |
|---|---|
| S0 | 00 |
| S1 | 01 |
| S2 | 10 |

Moore FSM




$S_1' =$

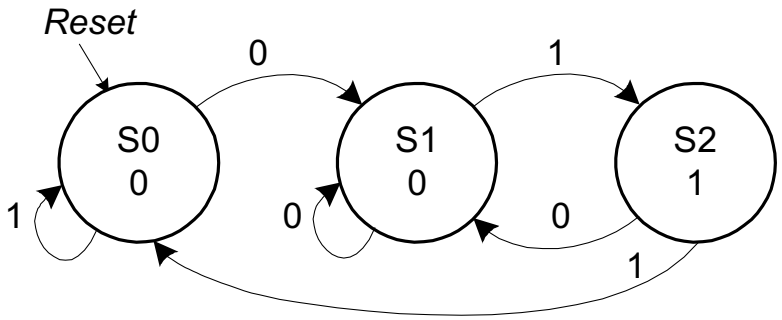
$S_0' =$

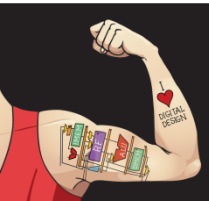
# Moore FSM Output Table

| Current State $S_1$ | Current State $S_0$ | Output $Y$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |

$Y =$

**Moore FSM**



# Mealy FSM State Transition & Output Table

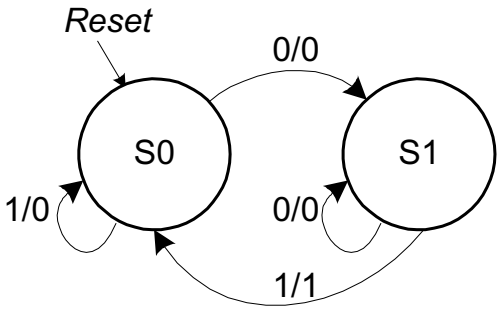
| Current State | Input | Next State | Output |
|---|---|---|---|
| $S_0$ | $A$ | $S'_0$ | $Y$ |
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

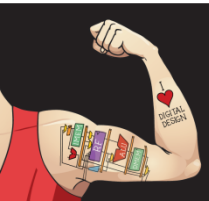
| State | Encoding |
|---|---|
| S0 | 0 |
| S1 | 1 |

**Mealy FSM**

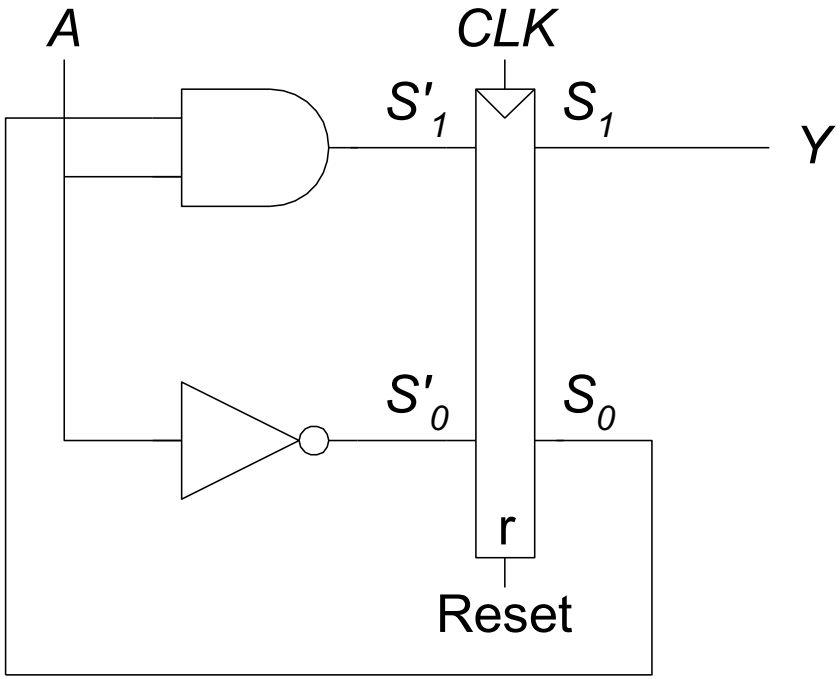
$S_0' =$

$Y =$

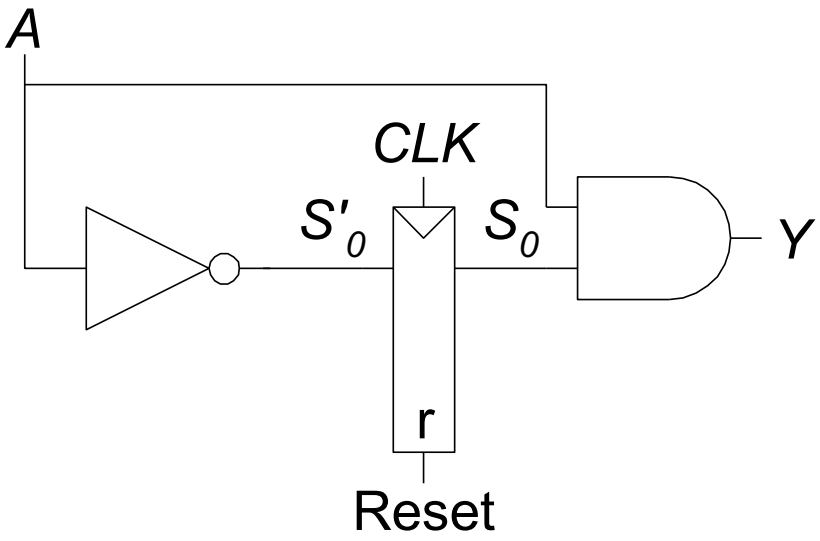


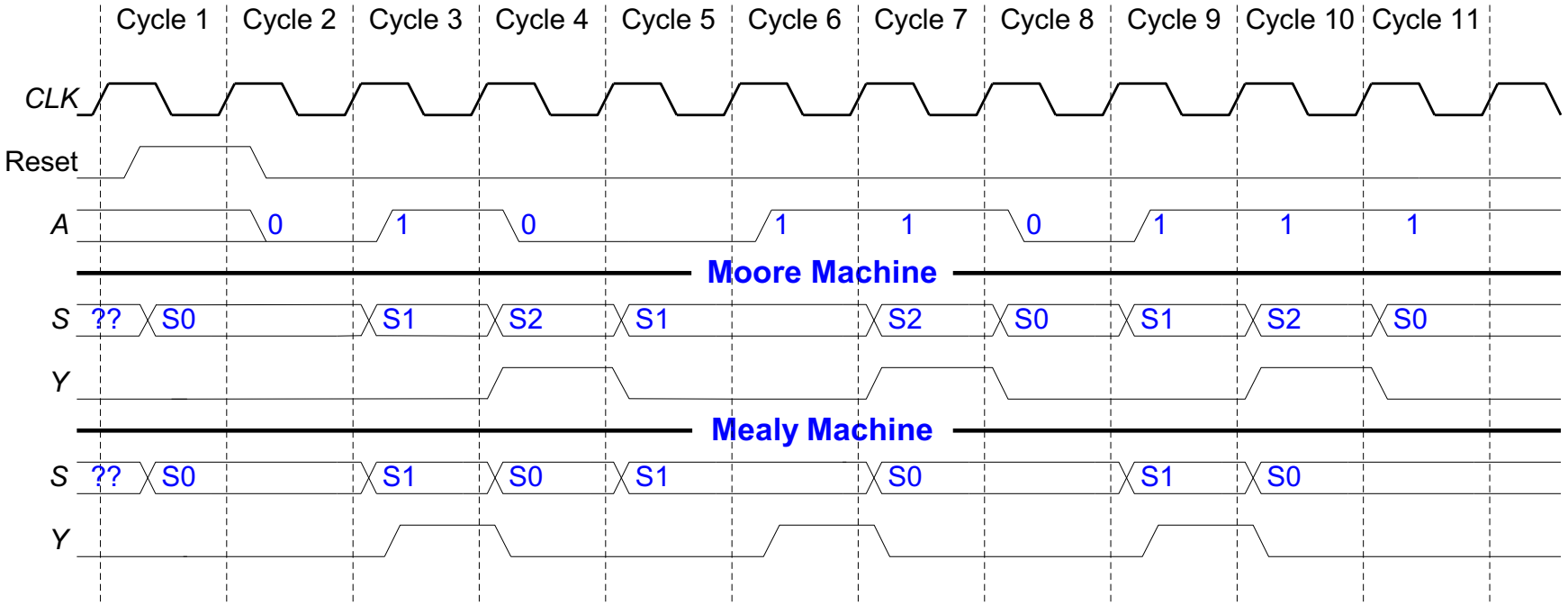
# Moore




# Mealy

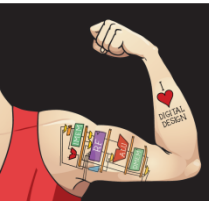


# Moore & Mealy Timing Diagram



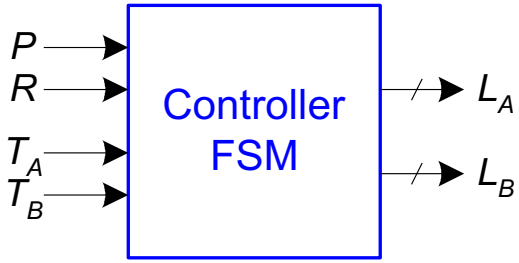
# Factoring State Machines

- Break complex FSMs into smaller interacting FSMs
- Example: Modify traffic light controller to have Parade Mode.
  - Two more inputs: *P*, *R*
  - When ***P* = 1**, enter Parade Mode & Bravado Blvd light stays green
  - When ***R* = 1**, leave Parade Mode

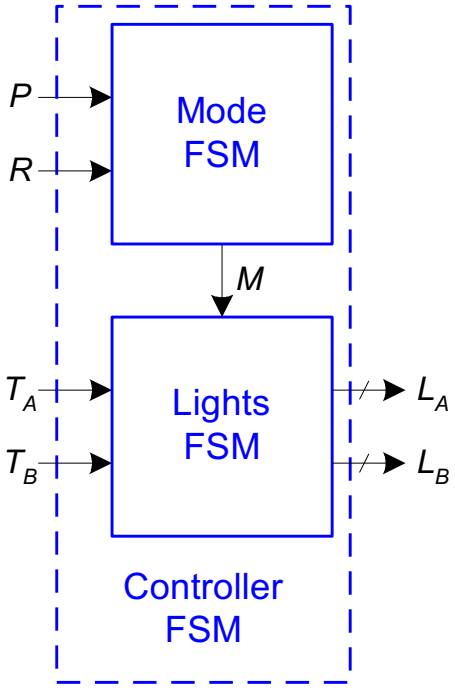


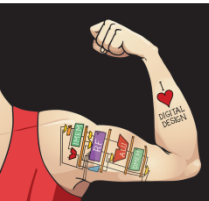
# Parade FSM

## Unfactored FSM

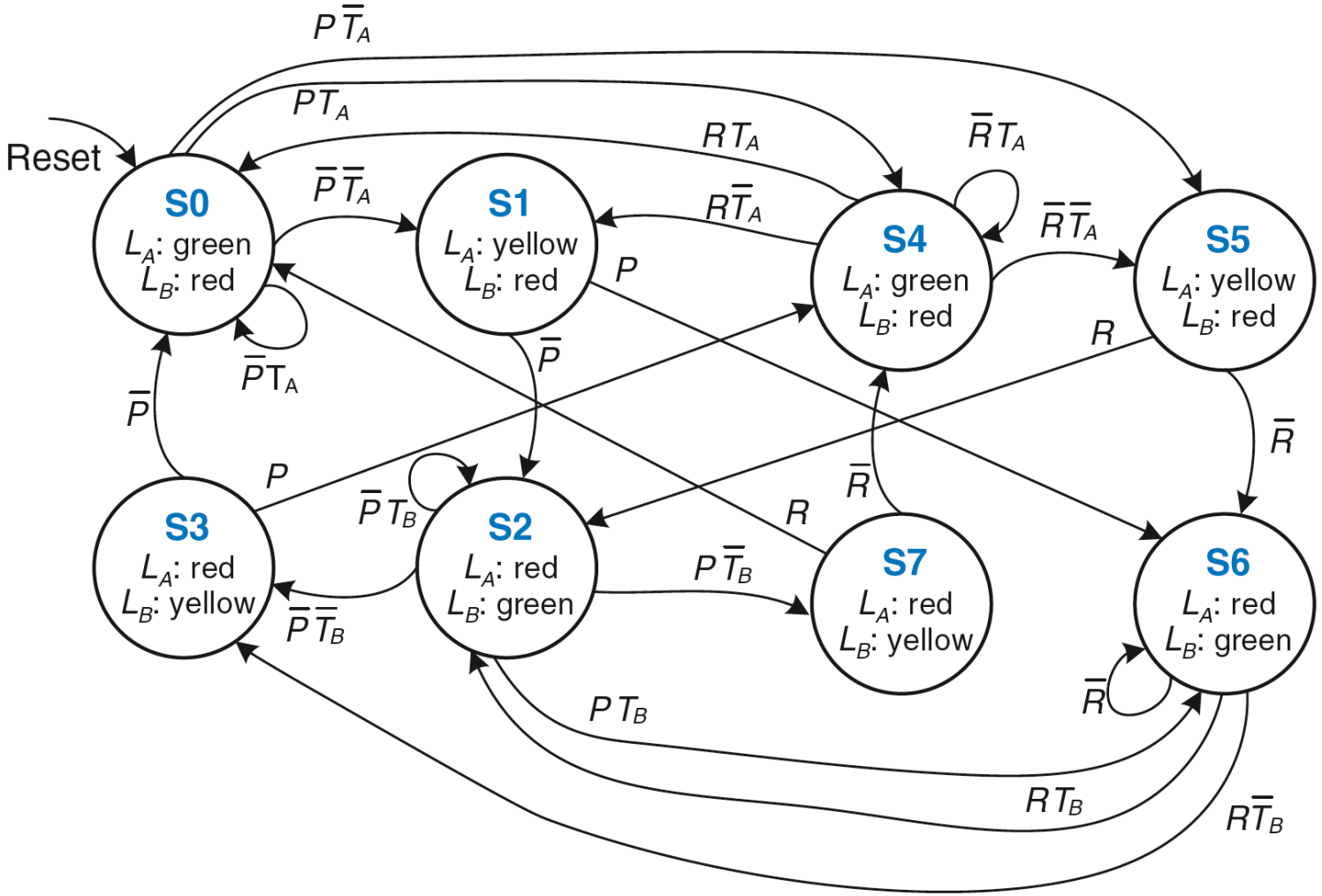


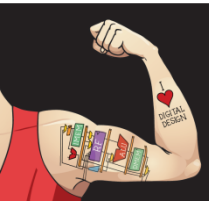

## Factored FSM



# Unfactored FSM

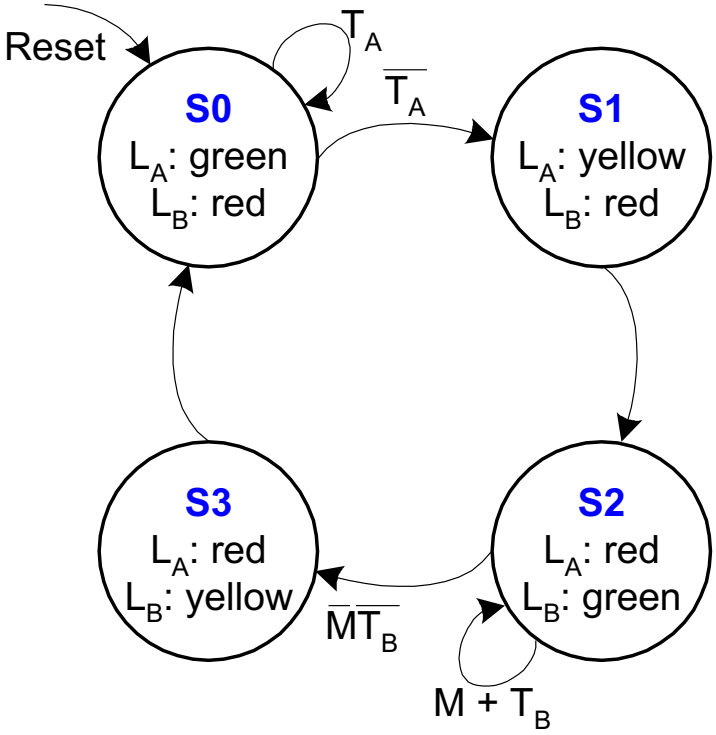


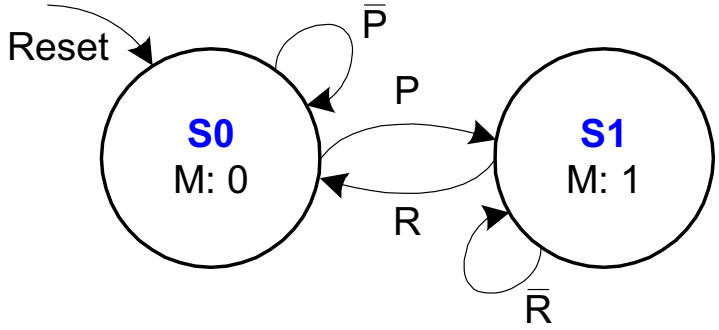


ELSEVIER

# Factored FSM




Lights FSM




Mode FSM

# FSM Design Procedure

1. Identify inputs and outputs
2. Sketch state transition diagram
3. Write state transition table
4. Select state encodings
5. For Moore machine:
   a. Rewrite state transition table with state encodings
   b. Write output table

5. For a Mealy machine:
   Rewrite combined state transition and output table with state encodings
6. Write Boolean equations for next state and output logic
7. Sketch the circuit schematic