Synchronous Sequential Circuit Design

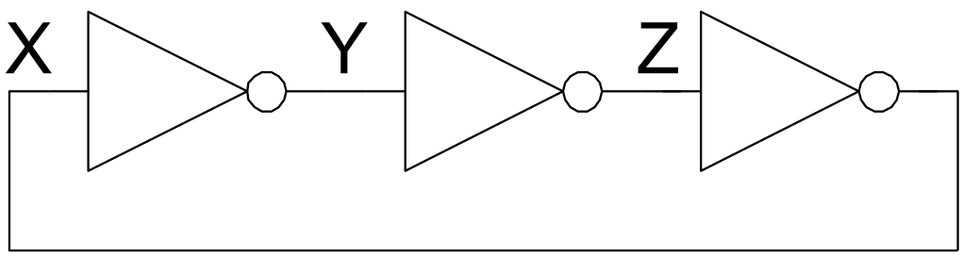
Digital Computer Design

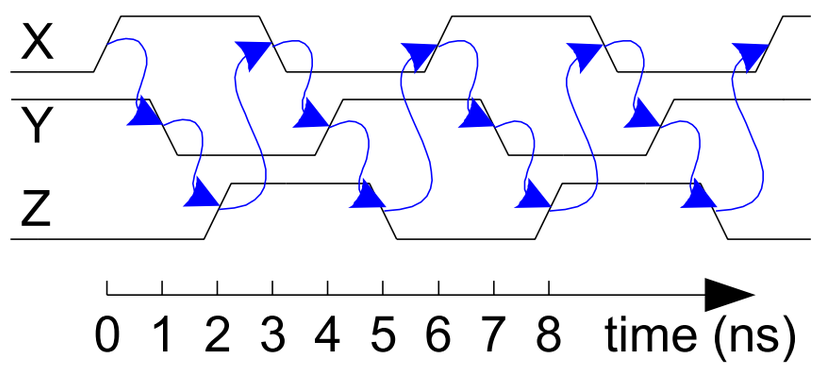
Races and Instability

- **Combinational logic**
  - has **no cyclic paths** and no races
  - If inputs are applied to combinational logic, the outputs will always settle to the correct value within a propagation delay.
- **Sequential circuits with cyclic paths**
  - If there is a cyclic path, **undesirable races** or **unstable behavior** might occur.

Example-1: Asynchronous Sequential Circuit

- A problematic circuit: (no clock)





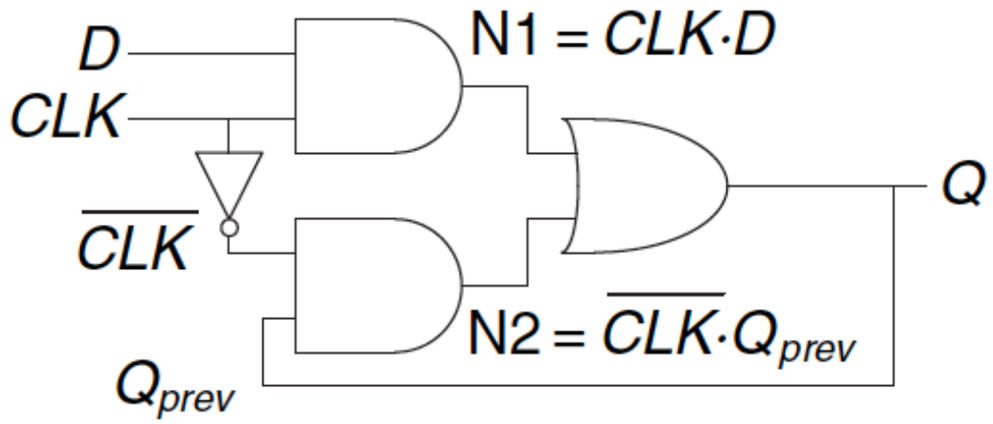
- No inputs and 1-3 outputs
- **Astable circuit**, oscillates
- Period depends on inverter delay
- It has a ***cyclic path***: output fed back to input

The behavior of the asynchronous circuit depends on the gate delays.

Example-2: Asynchronous Sequential Circuit

- A problematic circuit (with clock):

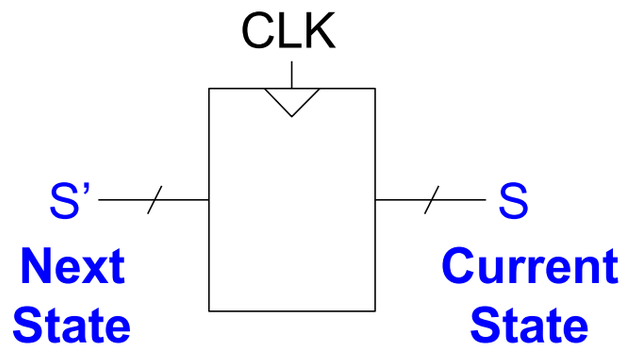
$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



- Suppose CLK=D= 1.
  - Keeps Q=1.
- Bur if the *inverter delay* is **longer than** that of the AND and OR gates:
  - **When CLK becomes 0, Q becomes stuck at 0.**

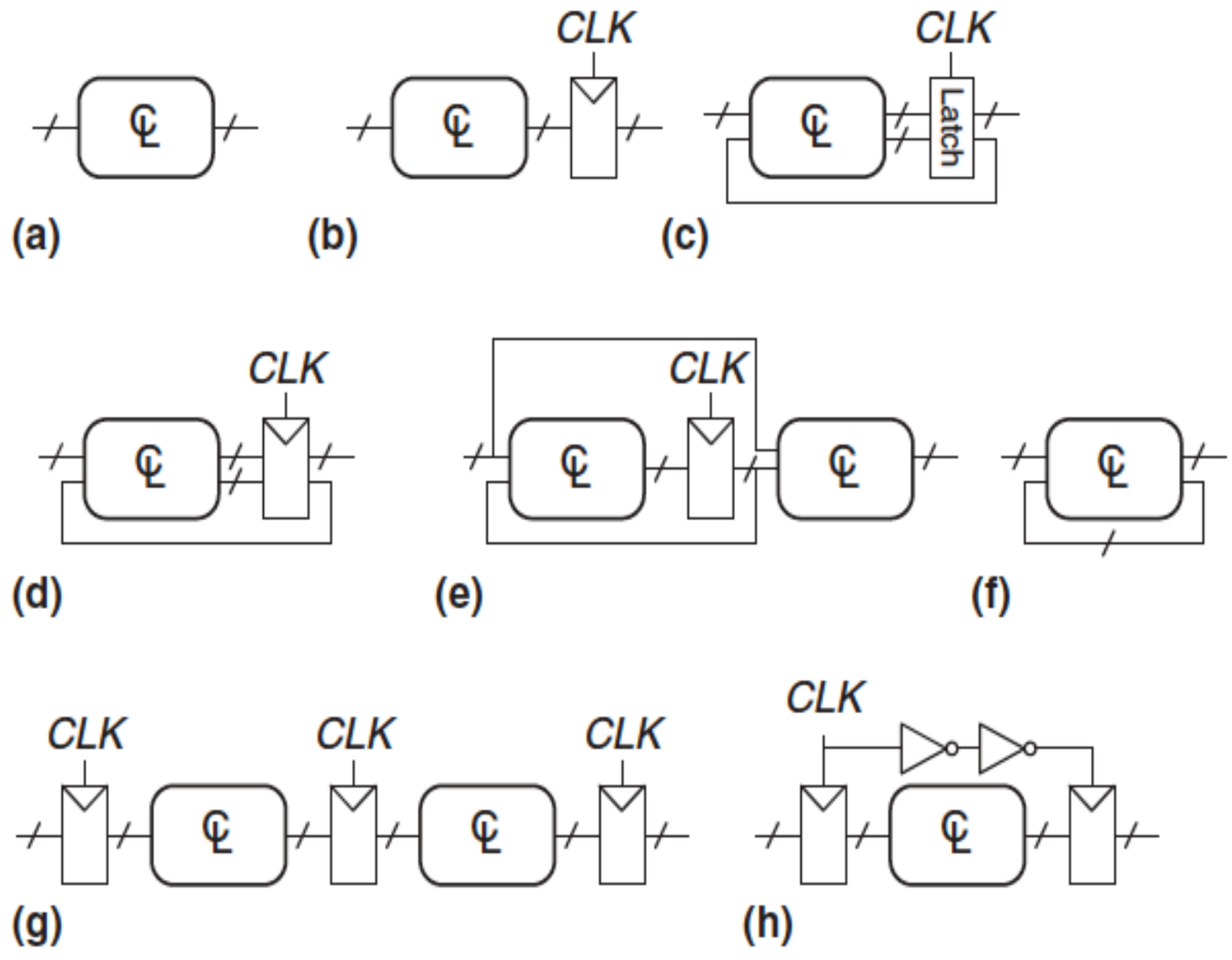
Synchronous Sequential Circuits

- Breaks cyclic paths by inserting registers
  - Registers contain **state** of the system
  - All registers receive the same clock signal
  - Every cyclic path contains at least one register
- State changes at clock edge: system **synchronized** to the clock



A flip-flop is the simplest synchronous sequential circuit.

Circuit Examples



Which of the circuits are synchronous sequential circuits?

*Sequential circuits that are not synchronous are **asynchronous**.*

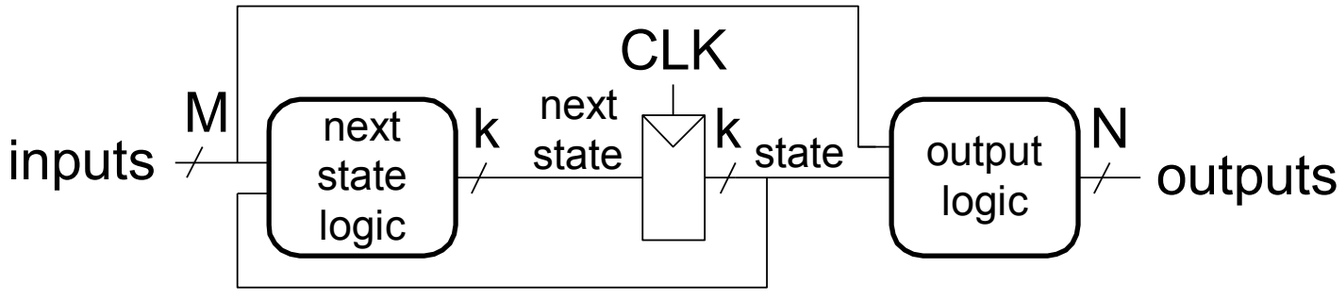
Synchronous Sequential Circuit Design

- Synchronous sequential circuits can be drawn in **FSM form**.
- A FSM consists of two blocks of combinational logic:
  - next state logic
  - output logic

Moore FSM

Moore FSM: outputs depend only on current state

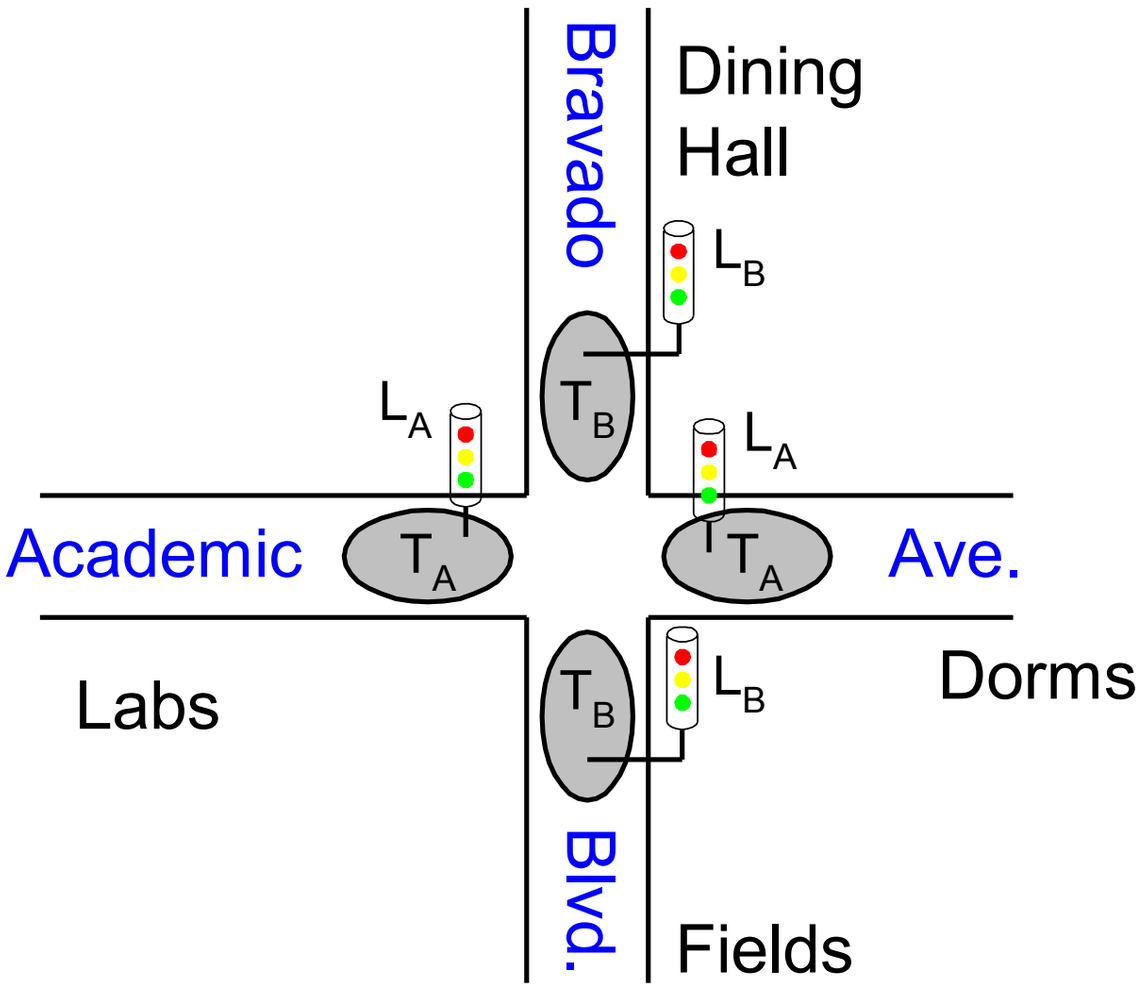
Mealy FSM



Mealy FSM: outputs depend on current state and inputs

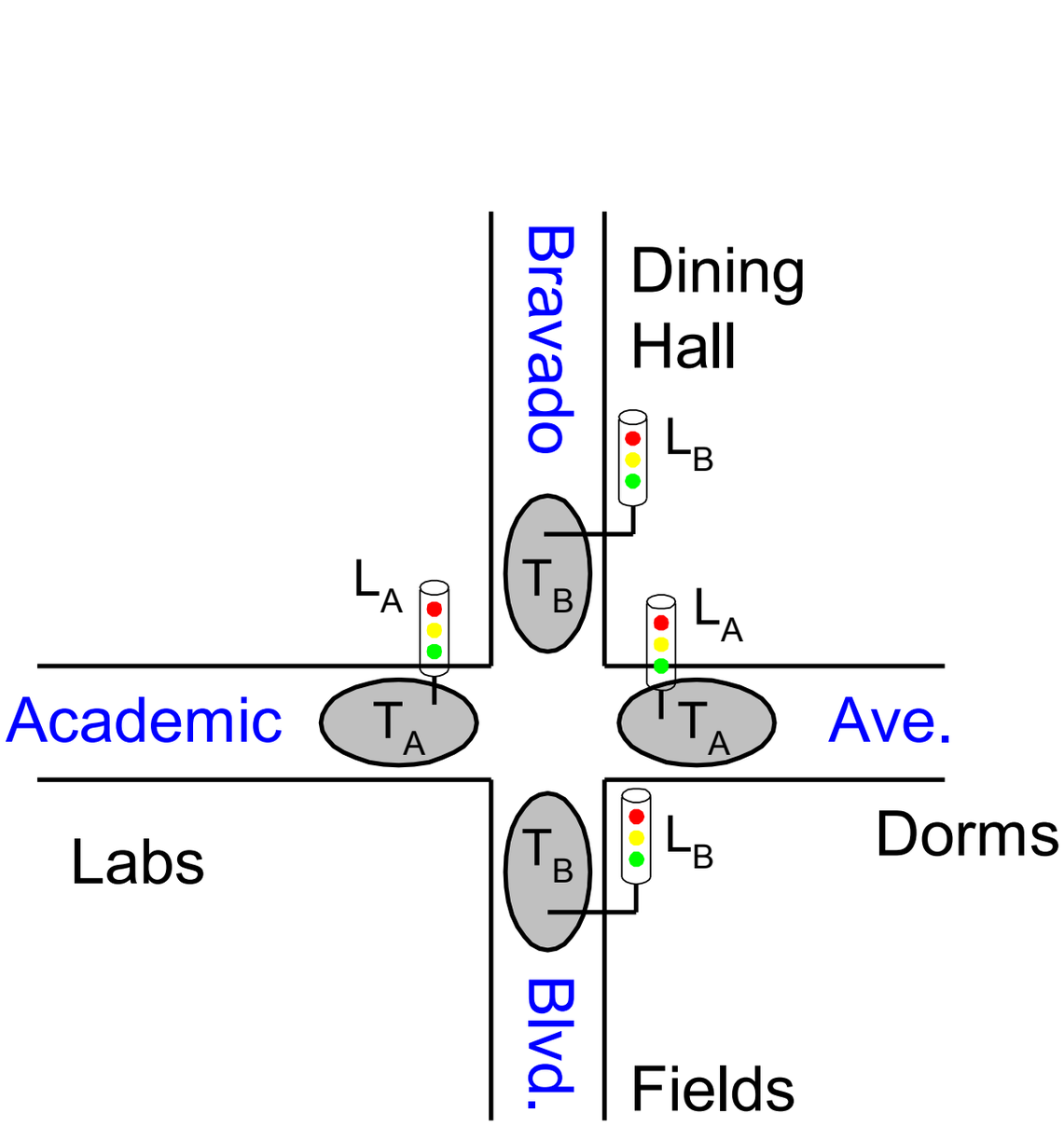
Traffic Light Controller Design

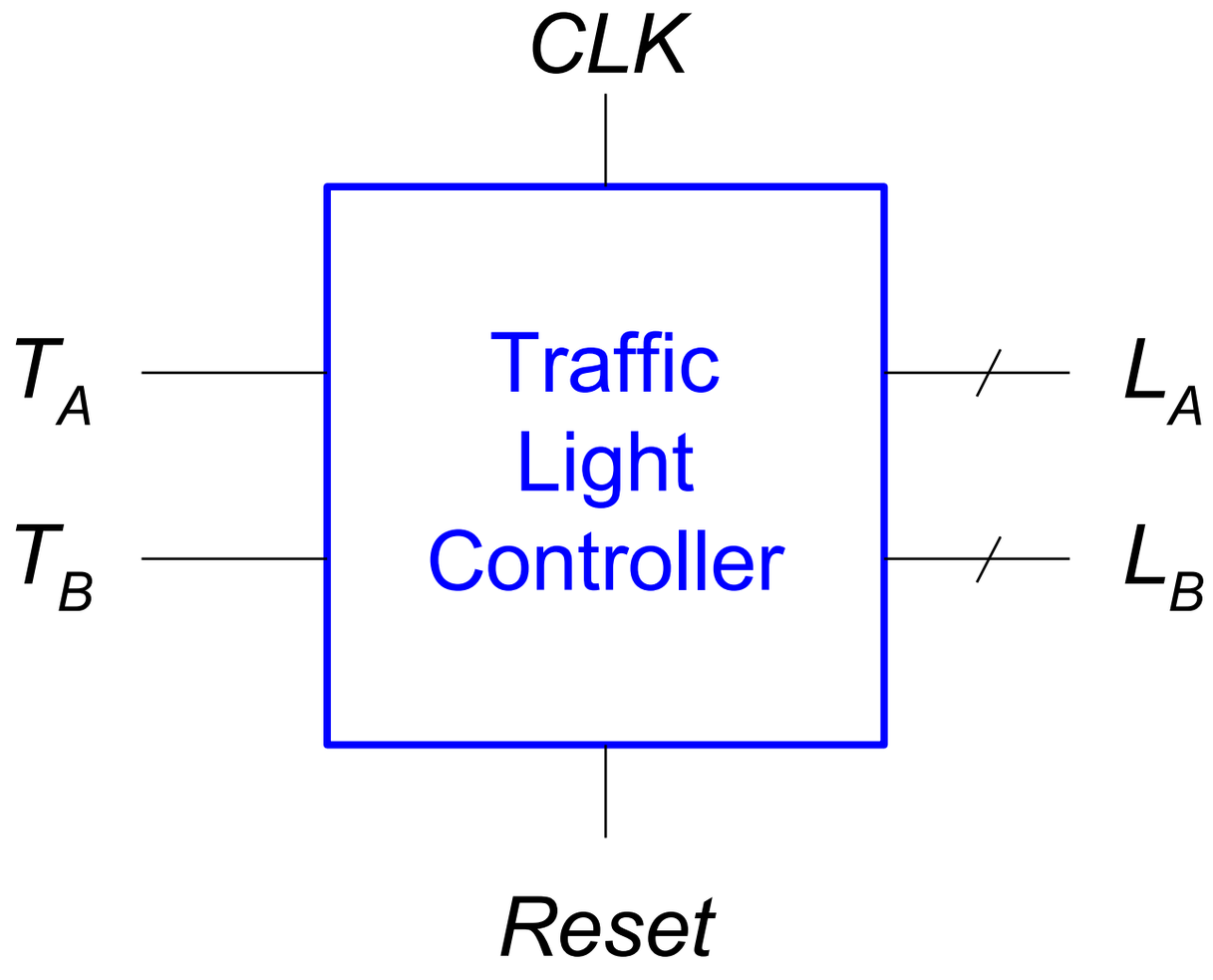
- **Traffic sensors:** T_A, T_B (TRUE when there's traffic)
- **Lights:** L_A, L_B



1- Determine Input and Output Signals

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



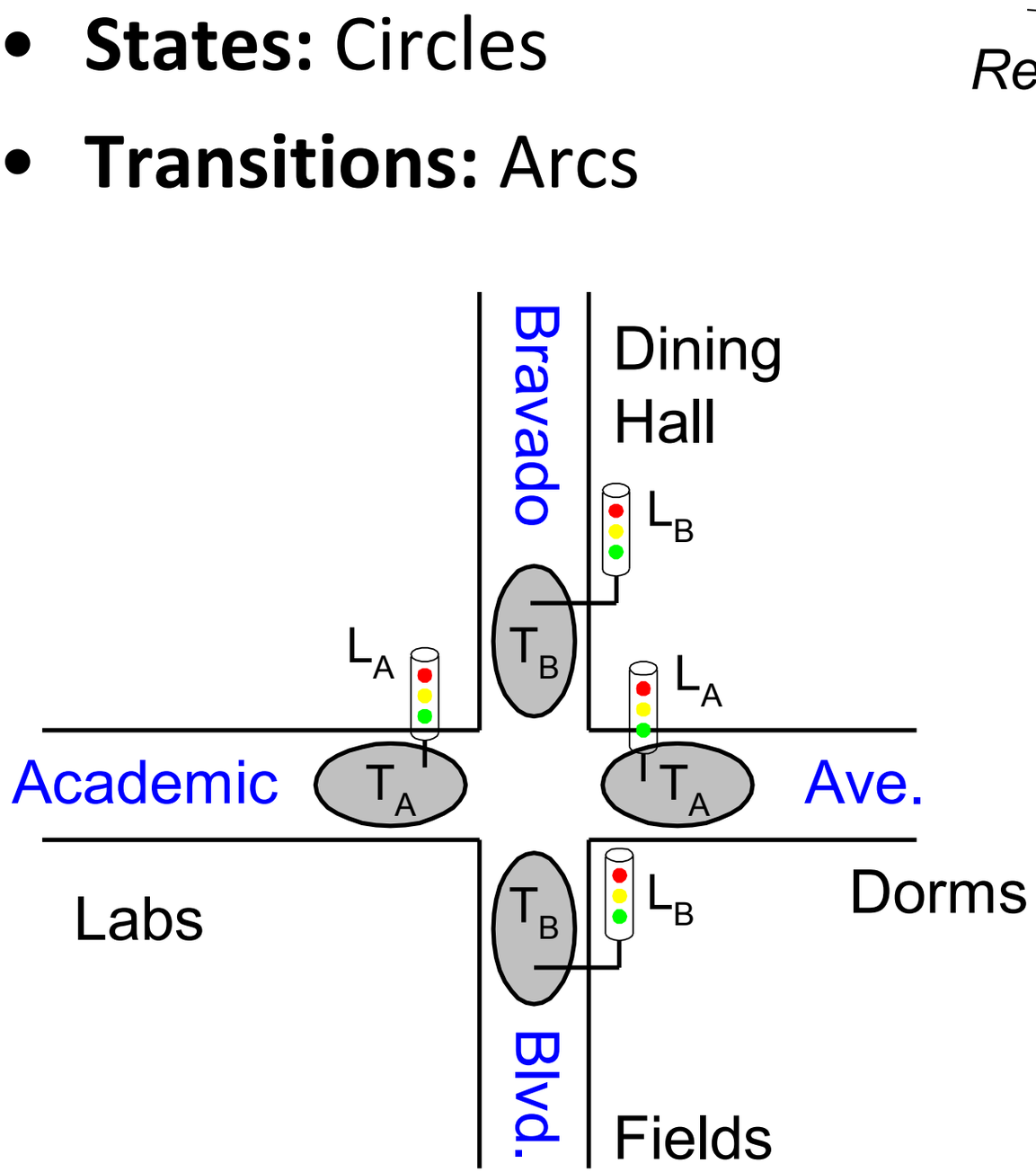


Inputs: *CLK*, *Reset*, T_A, T_B

Outputs: L_A, L_B

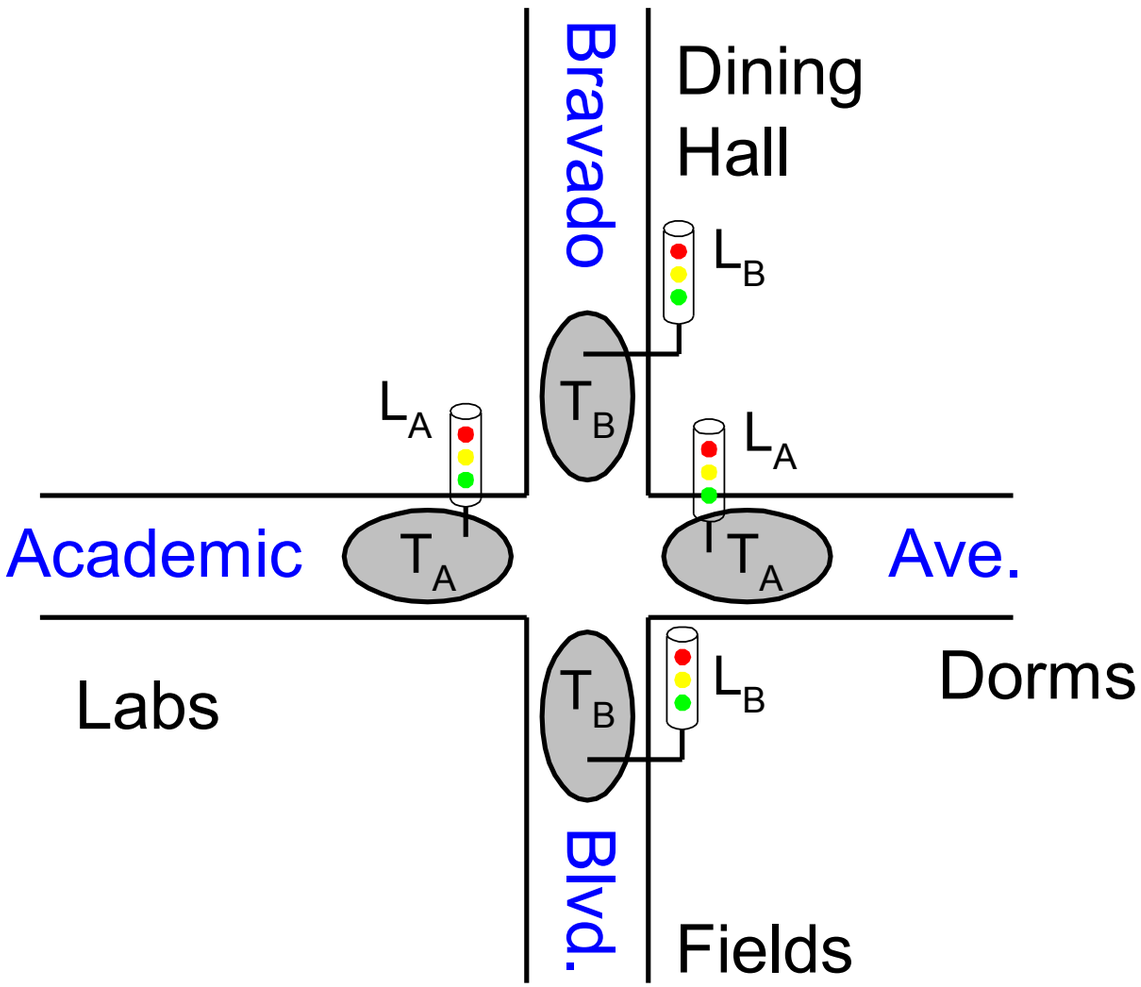
2- Obtain State Transition Diagram

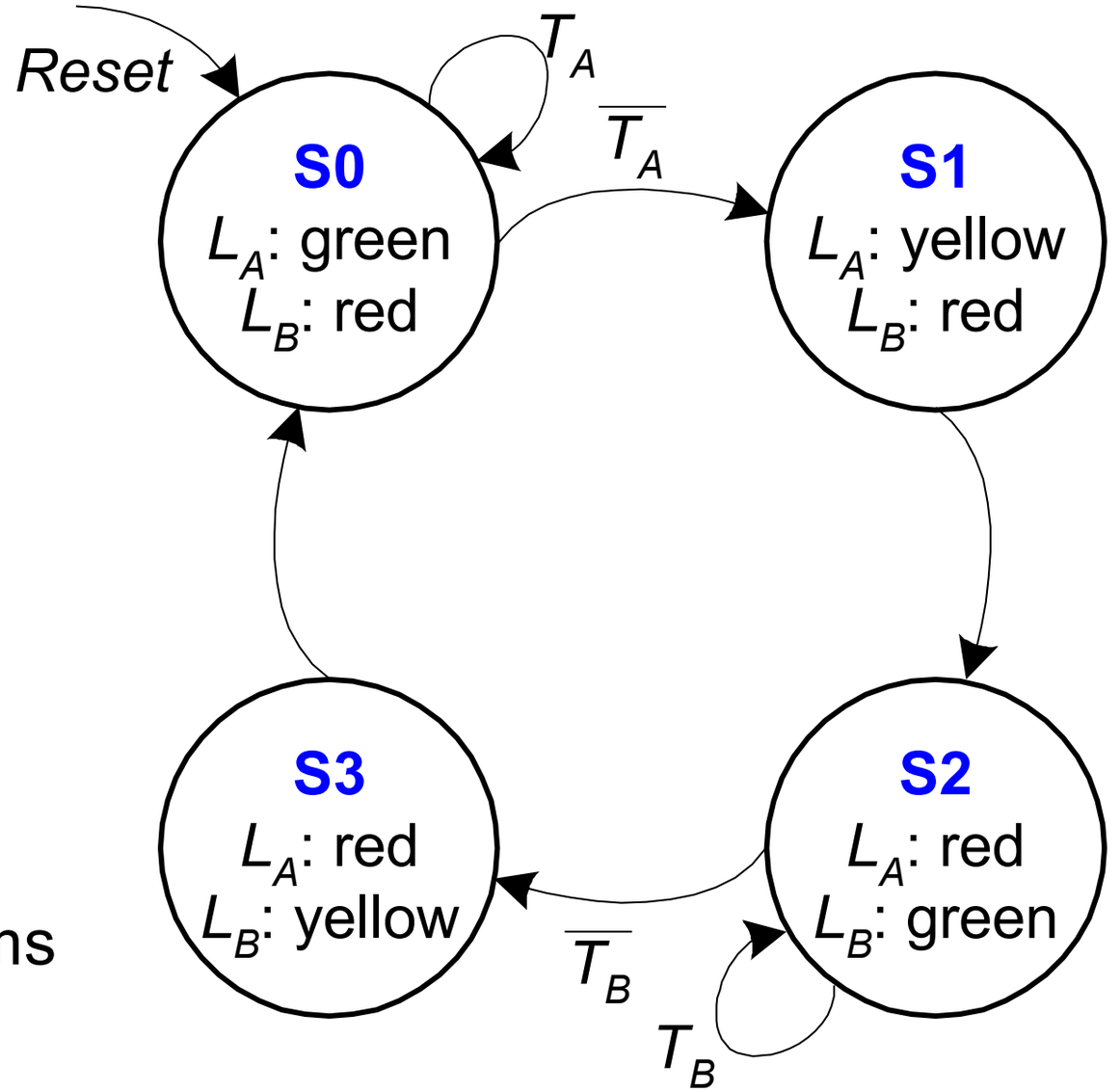
- **Moore FSM:** outputs labeled in each state
- **States:** Circles
- **Transitions:** Arcs



2- Obtain State Transition Diagram

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

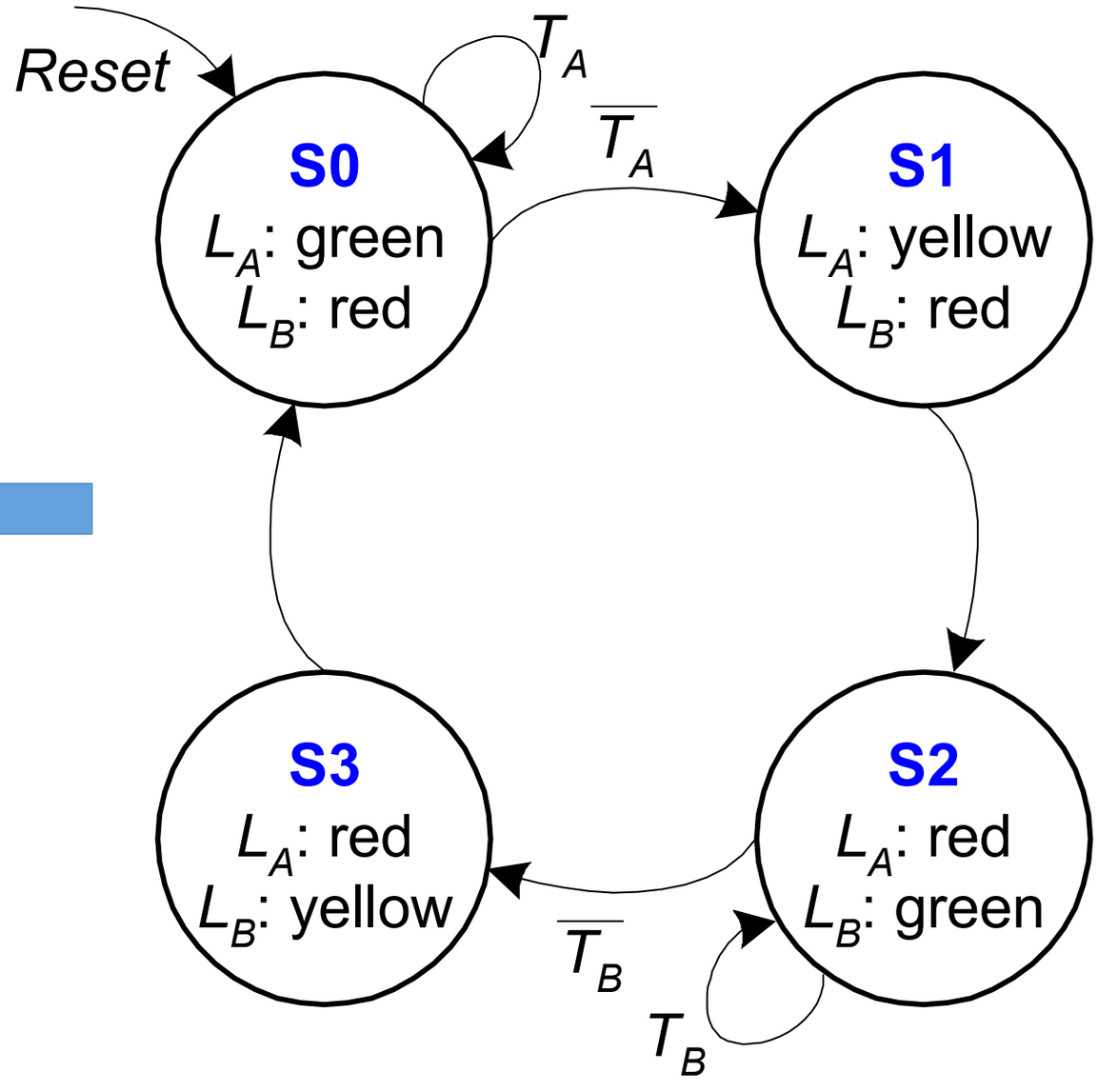




3- From State Transition Diagram to State Transition Table

State Transition Table

Current State	Inputs		Next State
S	T_A	T_B	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



4- State Encoding

- The state transition diagram is abstract in that it uses states labeled S0, S1, S2, S3.
  - To build a real circuit, the states and outputs must be assigned **binary encodings**.
- Each state is encoded with **two bits**: $S_{1:0}$

State	Encoding
S0	00
S1	01
S2	10
S3	11

5- Encoded State Transition Table

State Transition Table

Current State	Inputs		Next State
S	T_A	T_B	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

State	Encoding
S0	00
S1	01
S2	10
S3	11

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	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

6- Register (Flip-Flop) Input Equations (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

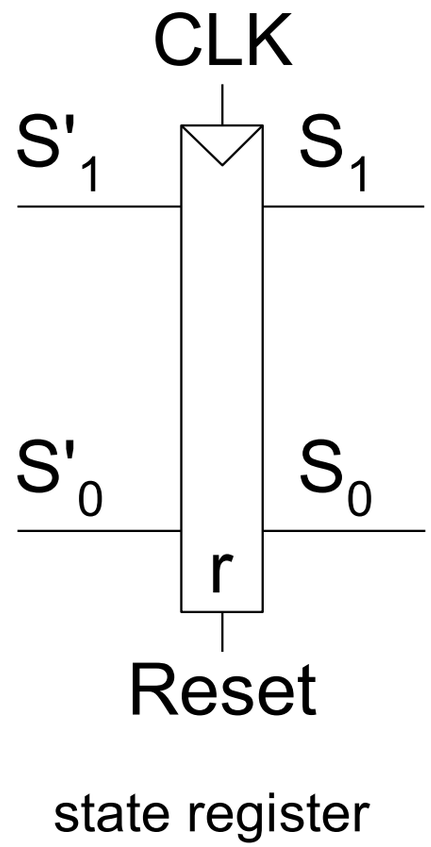
Flip-Flop (register) Input Equations:

$$S'_1 = (\overline{S}_1 \cdot S_0) + (S_1 \cdot \overline{S}_0 \cdot \overline{T}_B) + (S_1 \cdot \overline{S}_0 \cdot T_B)$$

$$S'_0 = (\overline{S}_1 \cdot \overline{S}_0 \cdot \overline{T}_A) + (S_1 \cdot \overline{S}_0 \cdot \overline{T}_B)$$

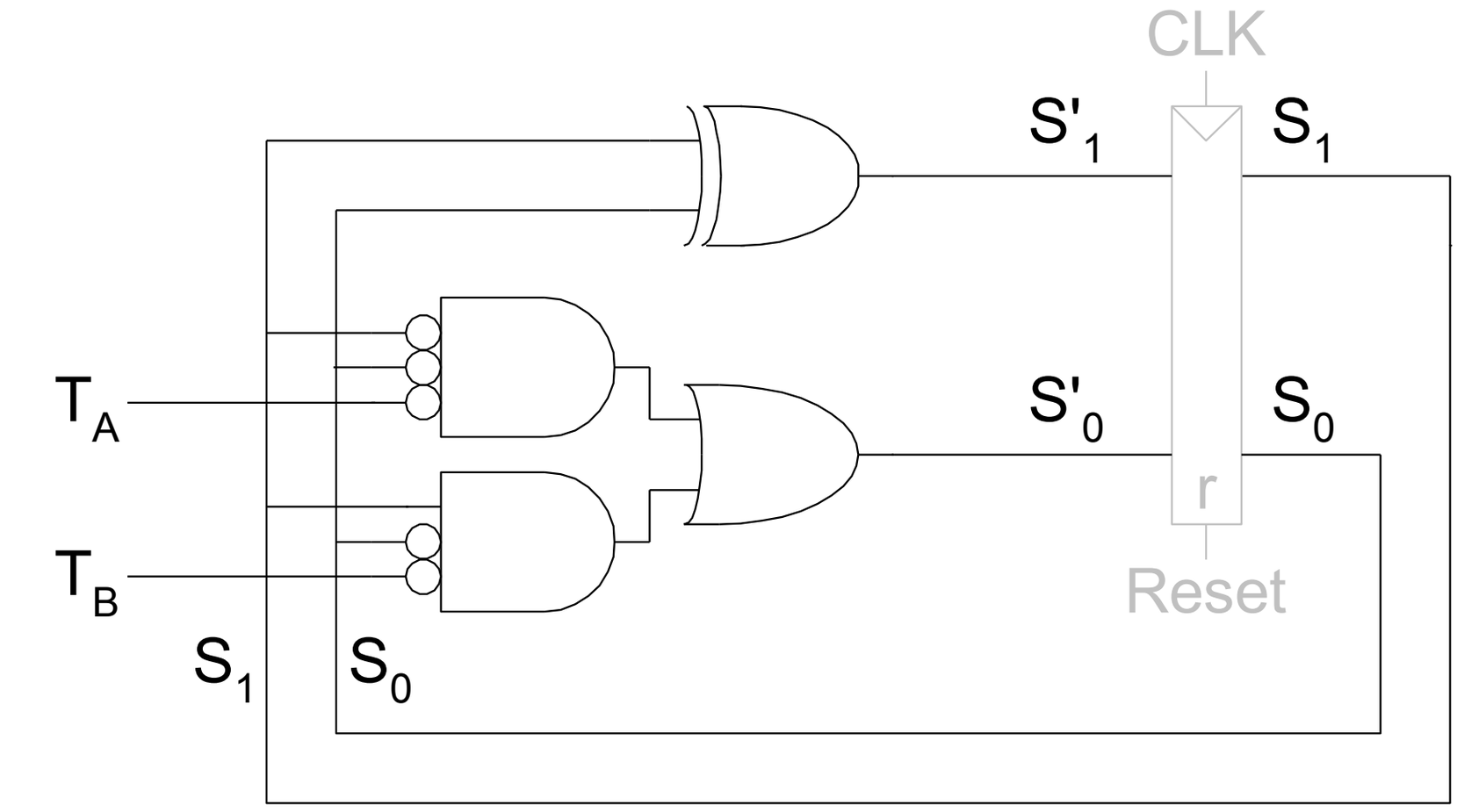
The equations can also be simplified using Karnaugh maps: $S'_1 = S_1 \text{ xor } S_0$

Circuit Schematic: Next State Logic



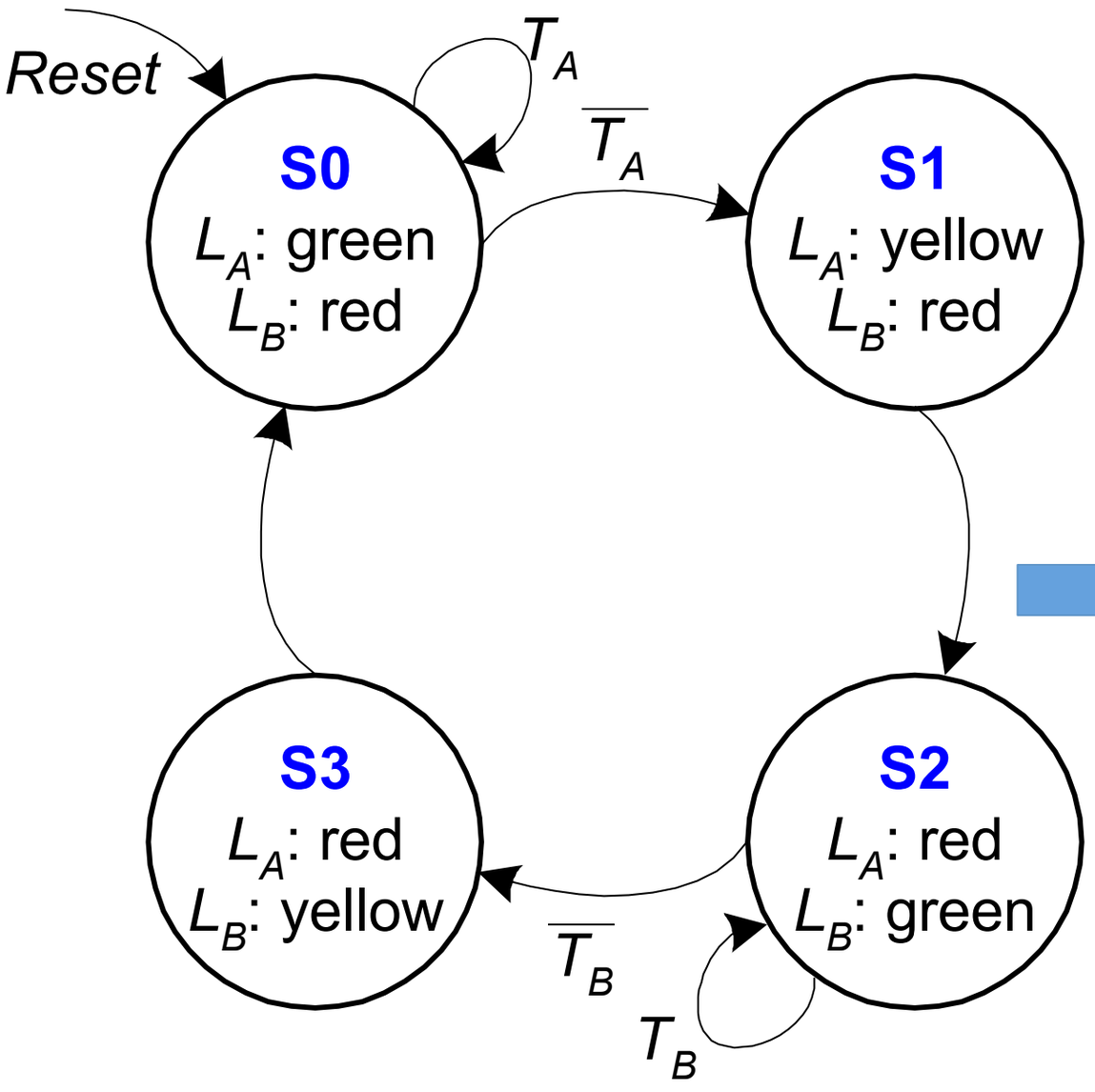
state register

Circuit Schematic: Next State Logic



inputs next state logic state register

7- Output Table and Output Equations (Output Logic)



Output	Encoding
green	00
yellow	01
red	10

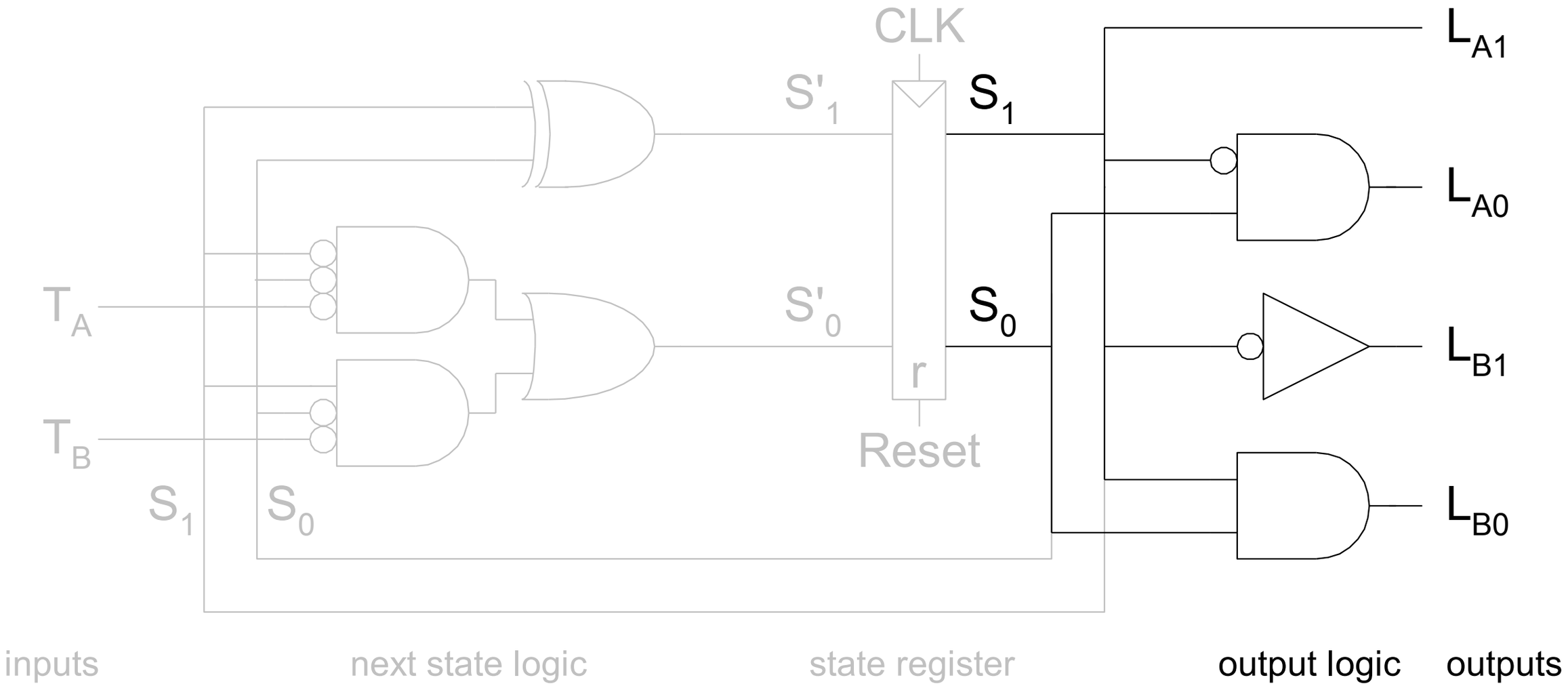
Output Table

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

Output Equations:

$$L_{A1} = S_1$$
$$L_{A0} = \overline{S_1} \cdot S_0$$
$$L_{B1} = \overline{S_1}$$
$$L_{B0} = S_1 \cdot S_0$$

Circuit Schematic: Output Logic



One-Hot State Encoding

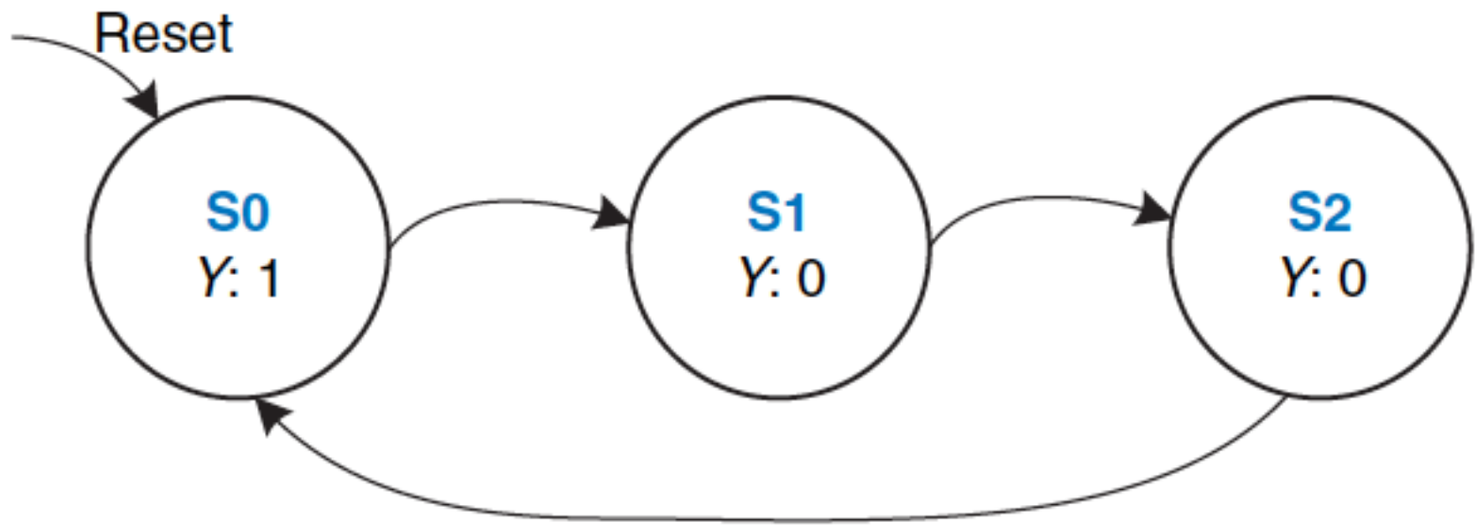
- **Binary encoding**:
  - i.e., for four states, 00, 01, 10, 11
  - **K states** only needs **$\log_2 K$** bits of state
- **One-hot encoding**
  - One state bit per state
  - **K states** only needs **K** bits of state
  - Only one state bit HIGH at once
  - i.e., for 4 states, 0001, 0010, 0100, 1000
  - Requires more flip-flops
  - Often next state and output logic is simpler

State	Binary Encoding
S0	00
S1	01
S2	10

State	One Hot Encoding		
	S_2	S_1	S_0
S0	0	0	1
S1	0	1	0
S2	1	0	0

A divide-by-N Counter Design With One Hot State Encoding

- The output Y is HIGH for one clock cycle out of every N.



Divide-by-3 counter

State Transition Table

Current State	Next State
S0	S1
S1	S2
S2	S0

Output Table

Current State	Output
S0	1
S1	0
S2	0

State Table, Output Table, State Encoding

State	One Hot Encoding		
	S_2	S_1	S_0
S0	0	0	1
S1	0	1	0
S2	1	0	0

State Encoding

Current State			Next State		
S_2	S_1	S_0	S'_2	S'_1	S'_0
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

Encoded State Table

Current State			Output
S_2	S_1	S_0	Y
0	0	1	1
0	1	0	0
1	0	0	0

Encoded Output Table

Register Input Equations and Output Equations

Current State			Next State		
S_2	S_1	S_0	S'_2	S'_1	S'_0
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	0	0	1

Encoded State Table

Flip-Flop (register) Input Equations: $S'_2=S_1$

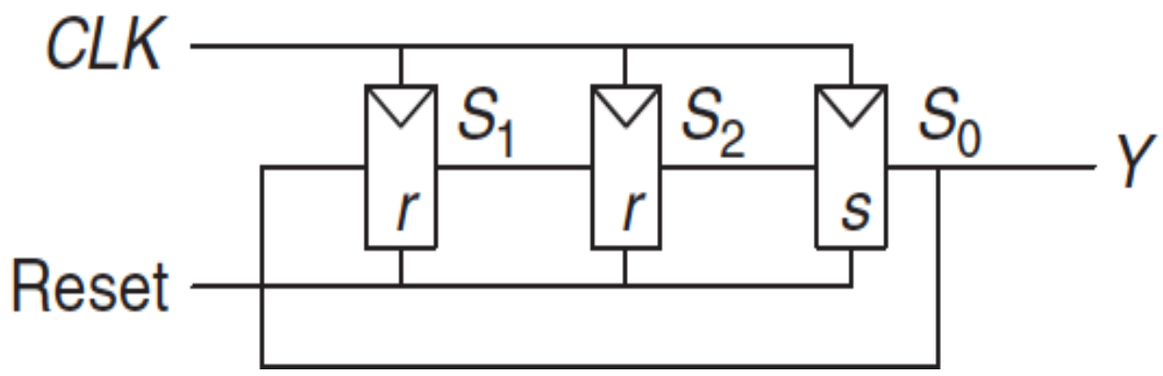
$S'_1=S_0$

$S'_0=S$

Current State			Output
S_2	S_1	S_0	Y
0	0	1	1
0	1	0	0
1	0	0	0

Output Table

Output Equations: $Y=S_0$



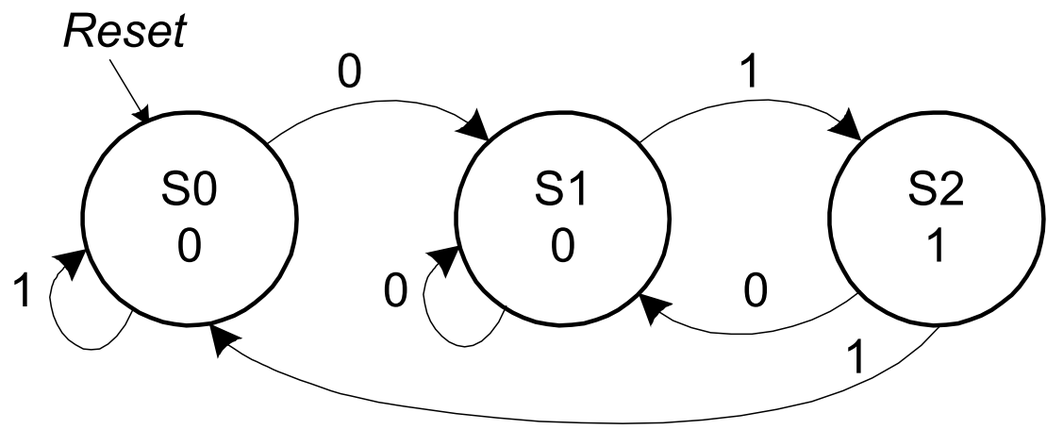
It is easy to derive boolean equations with One-Hot encoding.

Pattern Recognizer Design

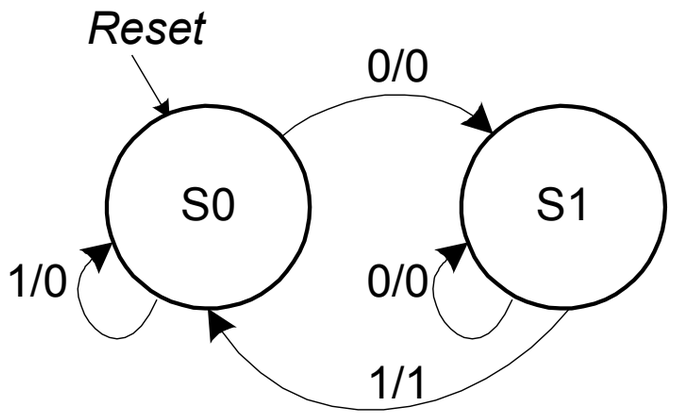
- A synchronous sequential circuit that recognizes the occurrence of **01** by observing a given sequence.
  - 0010101000101010001010010001
  - Outputs 1 when the patter is found.
- **Design Steps:**
  1. Determine Input and Output Signals
  2. Draw State Transition Diagram
  3. State Transition Table
  4. Encoded State Transition Table
  5. Register Input Equations
  6. Encoded Outputs
  7. Output Equations
  8. Draw Schematic

State Transition Diagrams

Moore FSM

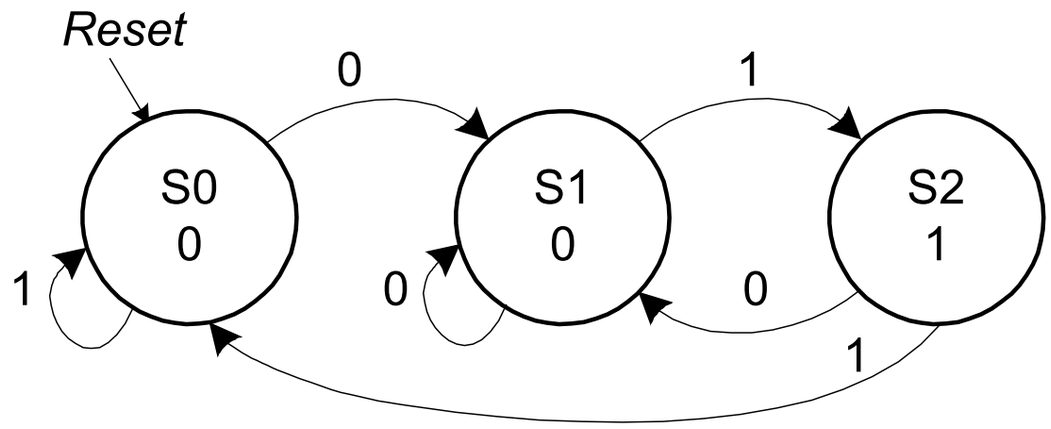


Mealy FSM



Moore State Transition and Output Tables

Moore FSM

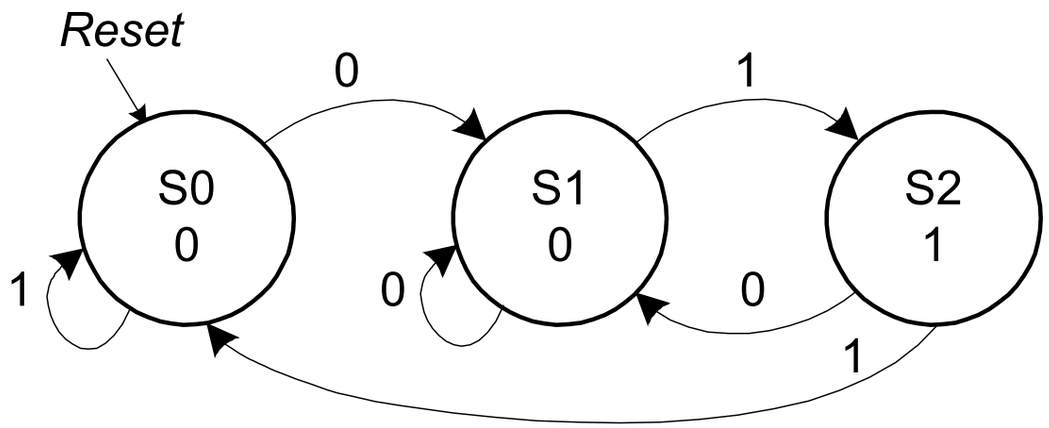


Current State S	Input A	Next State S’
S0	0	S1
S0	1	S0
S1	0	S1
S1	1	S2
S2	0	S1
S2	1	S0

Current State S	Output Y
S0	0
S1	0
S2	1

State Transition and Output Tables with State Encoding

Moore FSM

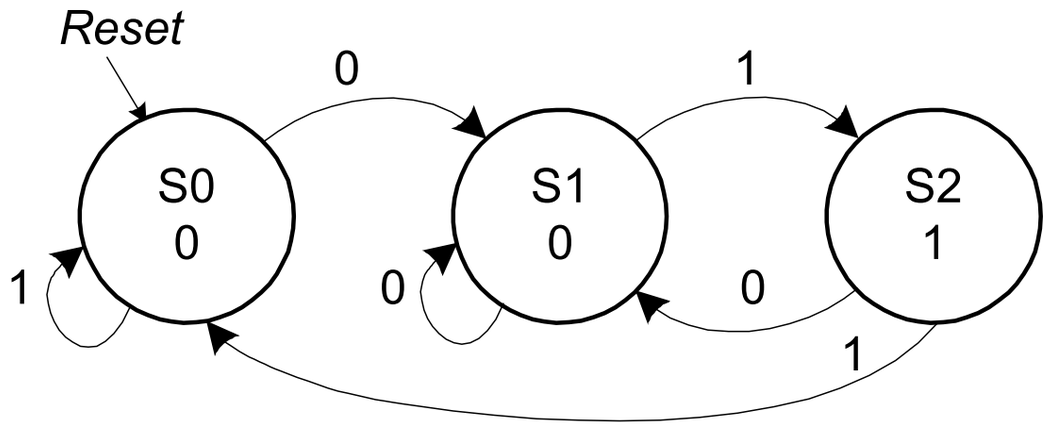


Current State	Output
S_1	S_0	Y
0	0	0
0	1	0
1	0	1

Current State		Input	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
1	0	1	1	0
1	0	0	0	1
1	1	1	0	0

Register Input Equations and Output Equations

Moore FSM



Current State	Output
S_1	S_0	Y
0	0	0
0	1	0
1	0	1

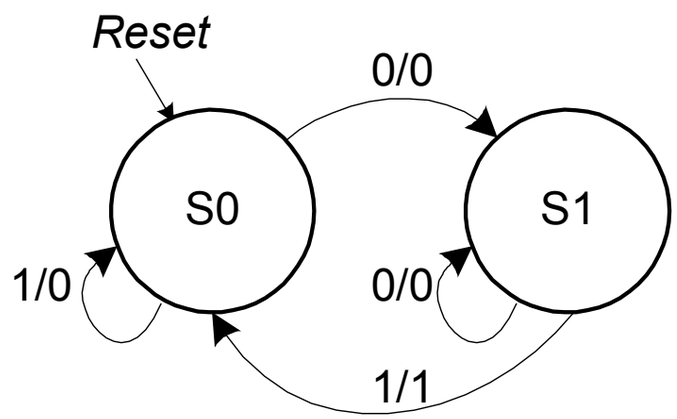
$$Y = S_1$$

Current State		Input	Next State	
S_1	S_0	A	S'_1	S'_0
0	0	0	0	1
0	0	1	0	0
0	1	0	0	1
1	0	1	1	0
1	0	0	0	1
1	1	1	0	0

$$S'_1 = S_0 A$$
$$S'_0 = \overline{A}$$

Mealy State Transition and Output Tables

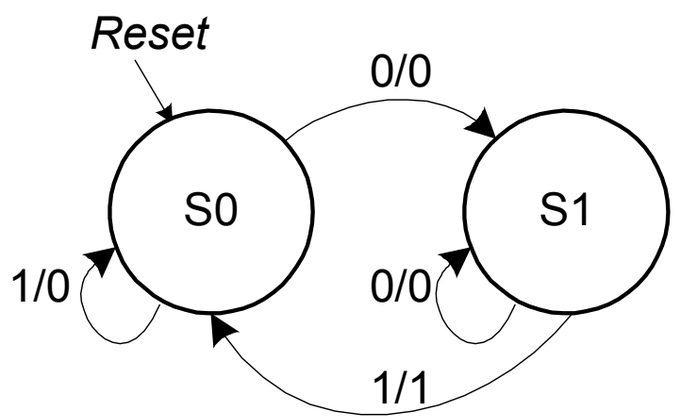
Mealy FSM



Current State S	Input A	Next State S’	Output Y
S0	0	S1	0
S0	1	S0	0
S1	0	S1	0
S1	1	S2	1

State Transition and Output Tables with State Encodings

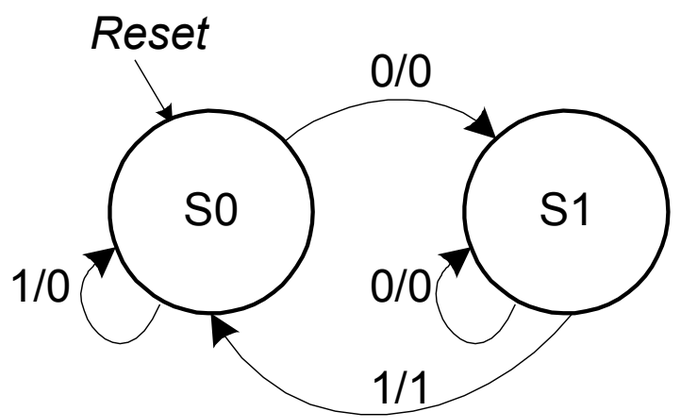
Mealy FSM



Current State S	Input A	Next State S’	Output Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

Register Input Equations and Output Equations

Mealy FSM

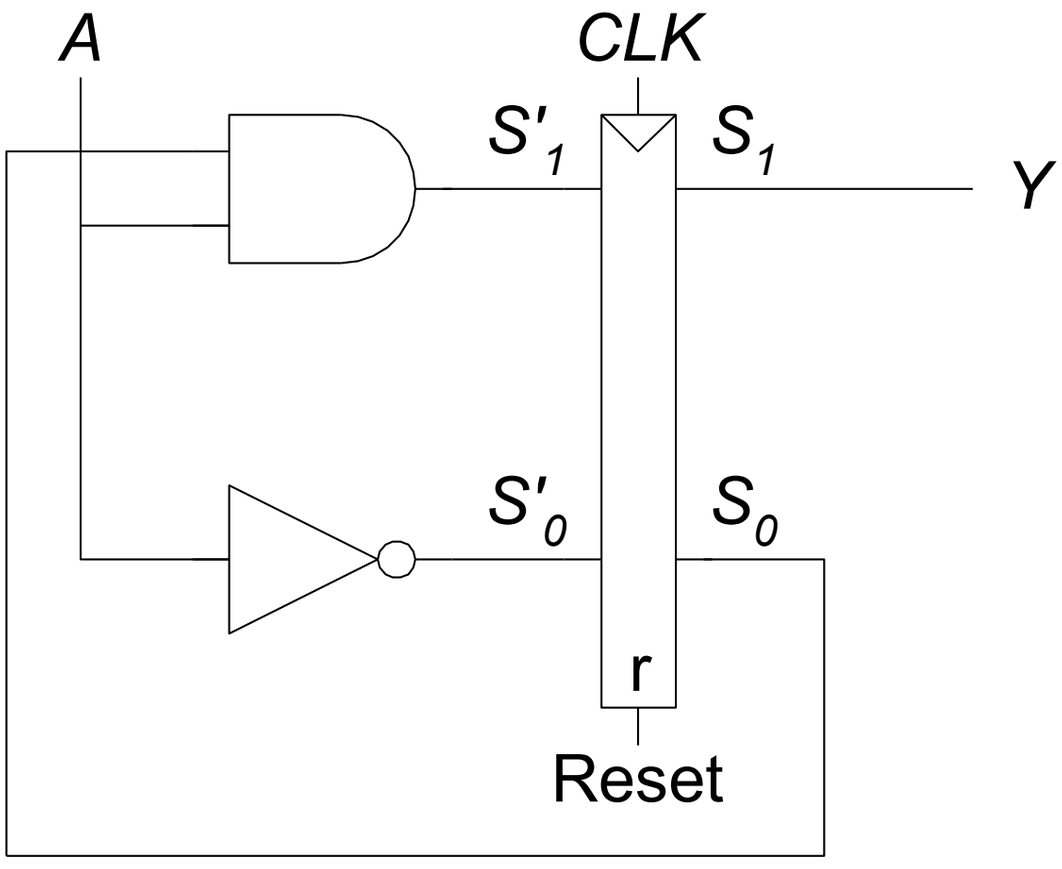


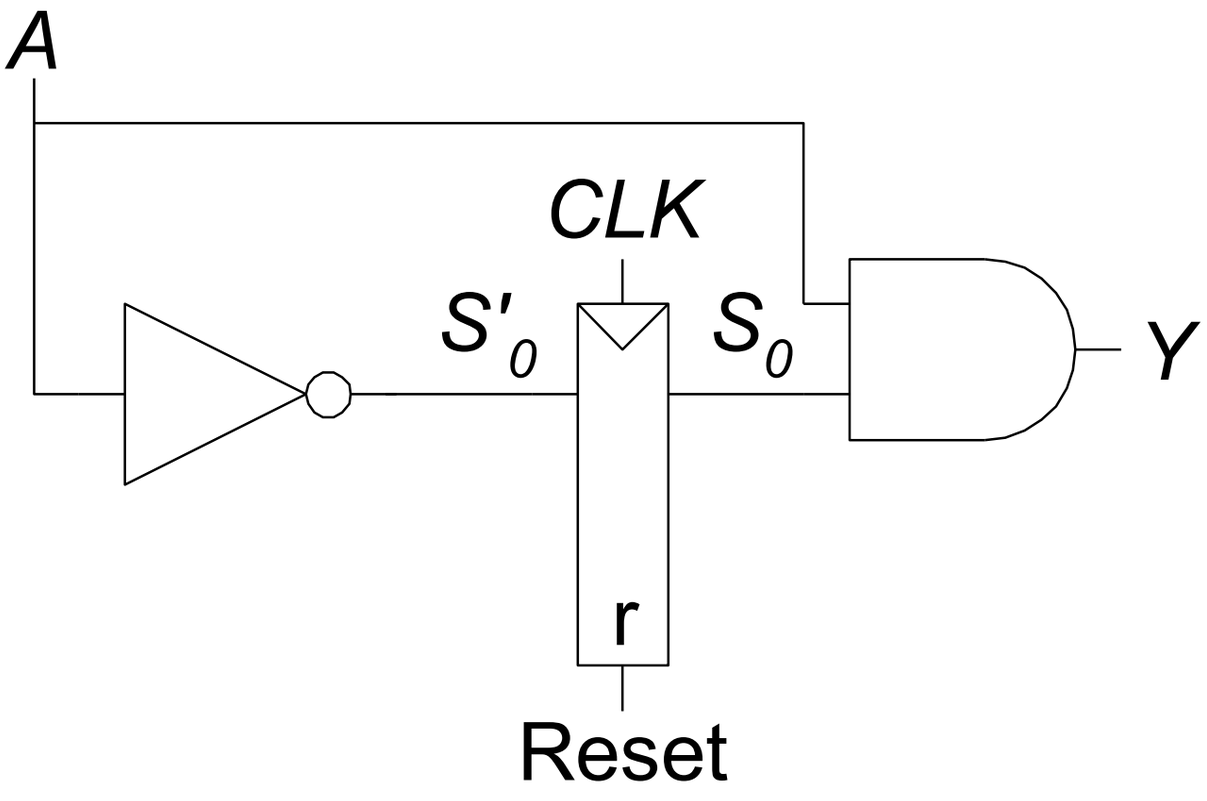
Current State S	Input A	Next State S’	Output Y
0	0	1	0
0	1	0	0
1	0	1	0
1	1	0	1

$$S'_0 = \overline{A}$$

$$Y = S_0 A$$

Circuit Schematic





Moore Machine

Mealy Machine

Further Reading

- These slides are sufficient to understand the design of the synchronous sequential circuits.
- You can read **Chapter 3** of your book
  – **Sections 3.2, 3.3, 3.5**