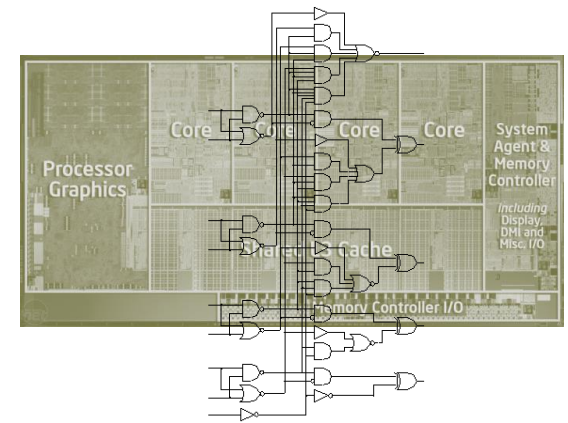




고려대학교
KOREA UNIVERSITY

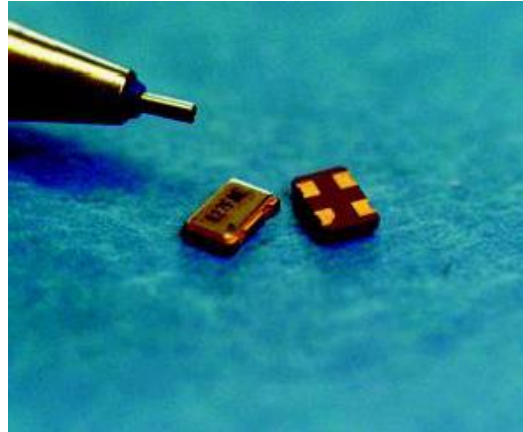


COSE221, COMP211 Logic Design

Lecture 4. Sequential Logic #2

Prof. Taeweon Suh
Computer Science & Engineering
Korea University

Clock Oscillators



32.768kHz Crystals & Oscillators

Surface Mount Ultra Miniature packages for YOUR application

Offered in a variety of package and tolerance options, Abracon's 32.768kHz Precision Crystals & Oscillators are an essential part of virtually all timing applications. Abracon's impressive track record of high reliability products for Automotive Electronics uniquely positions itself to offer low-cost, high quality devices for most Industrial and high volume consumer electronics applications.

FEATURES:

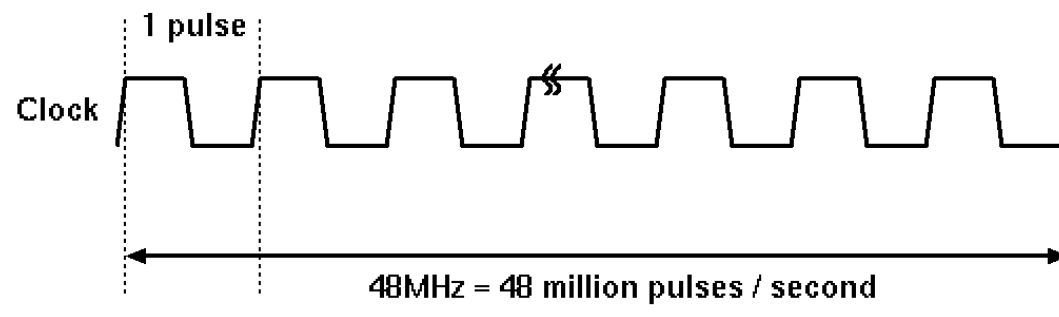
- 32.768kHz reference frequency
- Ultra miniature SMD packages
- Available $\pm 20.0\text{ppm}$ & $\pm 10.0\text{ppm}$ set tolerance
- Operating temperature range of -40°C $+85^{\circ}\text{C}$
- 12.5pf, 9pf, 7pf and 6pf standard crystal plating loads
- Oscillators available 20ppm -40°C $+85^{\circ}\text{C}$ with current consumption as low as 0.8 uA max

Authorized Stocking Distributors:

Mail www.abracon.com for additional listing of authorized distributors.

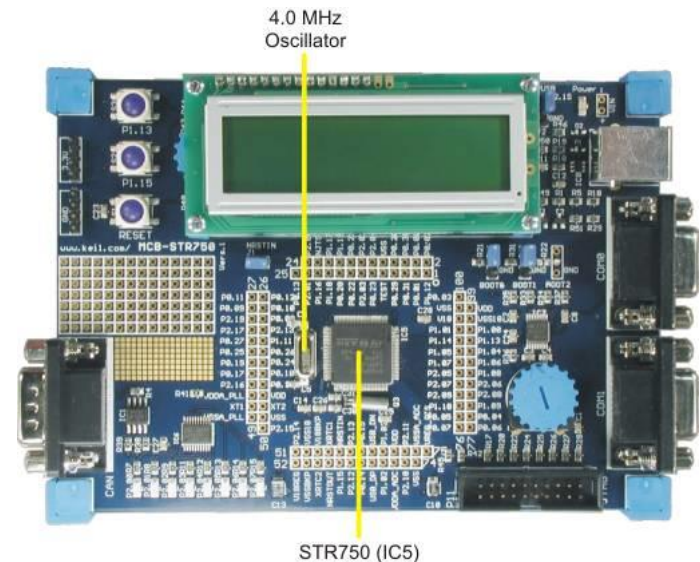
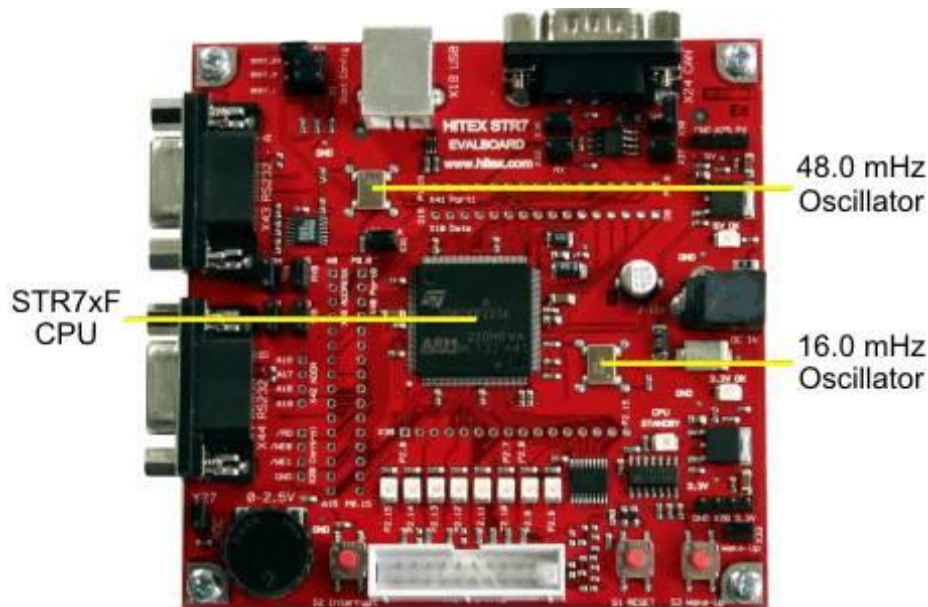
ABRACON CORPORATION
The Power of Linking Together

AB307 2.5 x 1.5 x 0.8 mm
AB309 4.1 x 1.8 x 1.9 mm
AB310 4.8 x 1.8 x 1.2 mm
AB311 4.8 x 2.5 x 0.8 mm
AB313 5.5 x 1.4 x 1.5 mm
AB325 6.5 x 3.8 x 2.5 mm
AB24TR6 8.0 x 1.9 mm
AB206J 8.8 x 2.5 mm

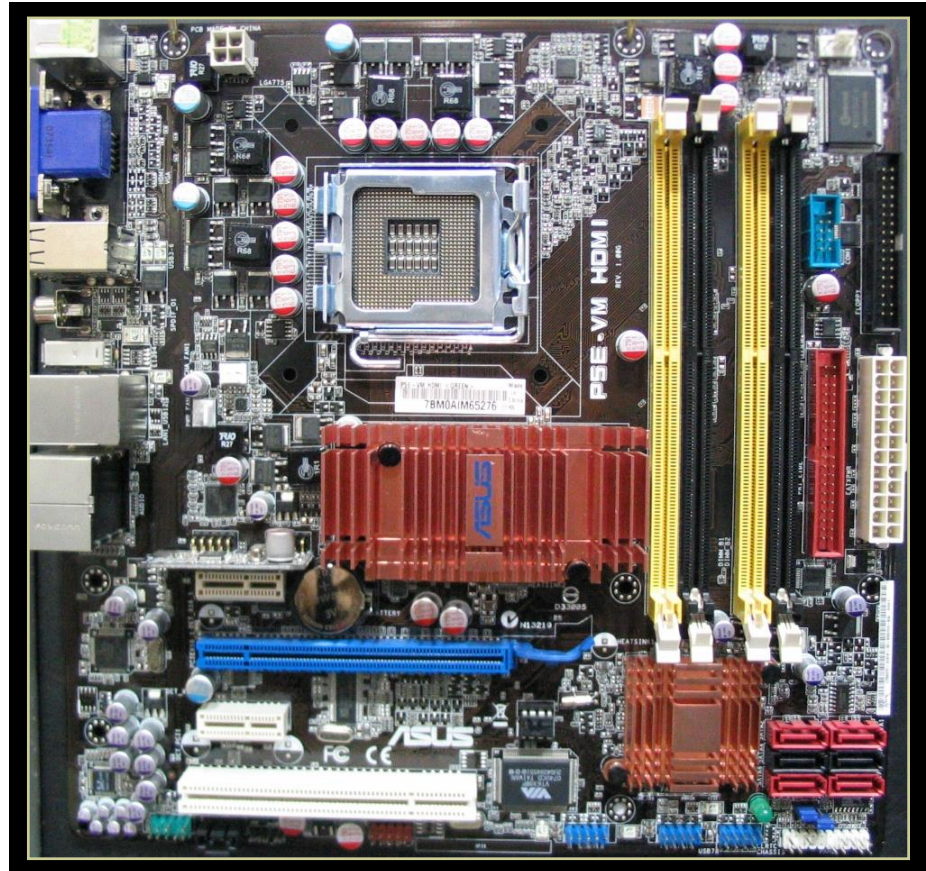


Clock Oscillators in Digital Systems

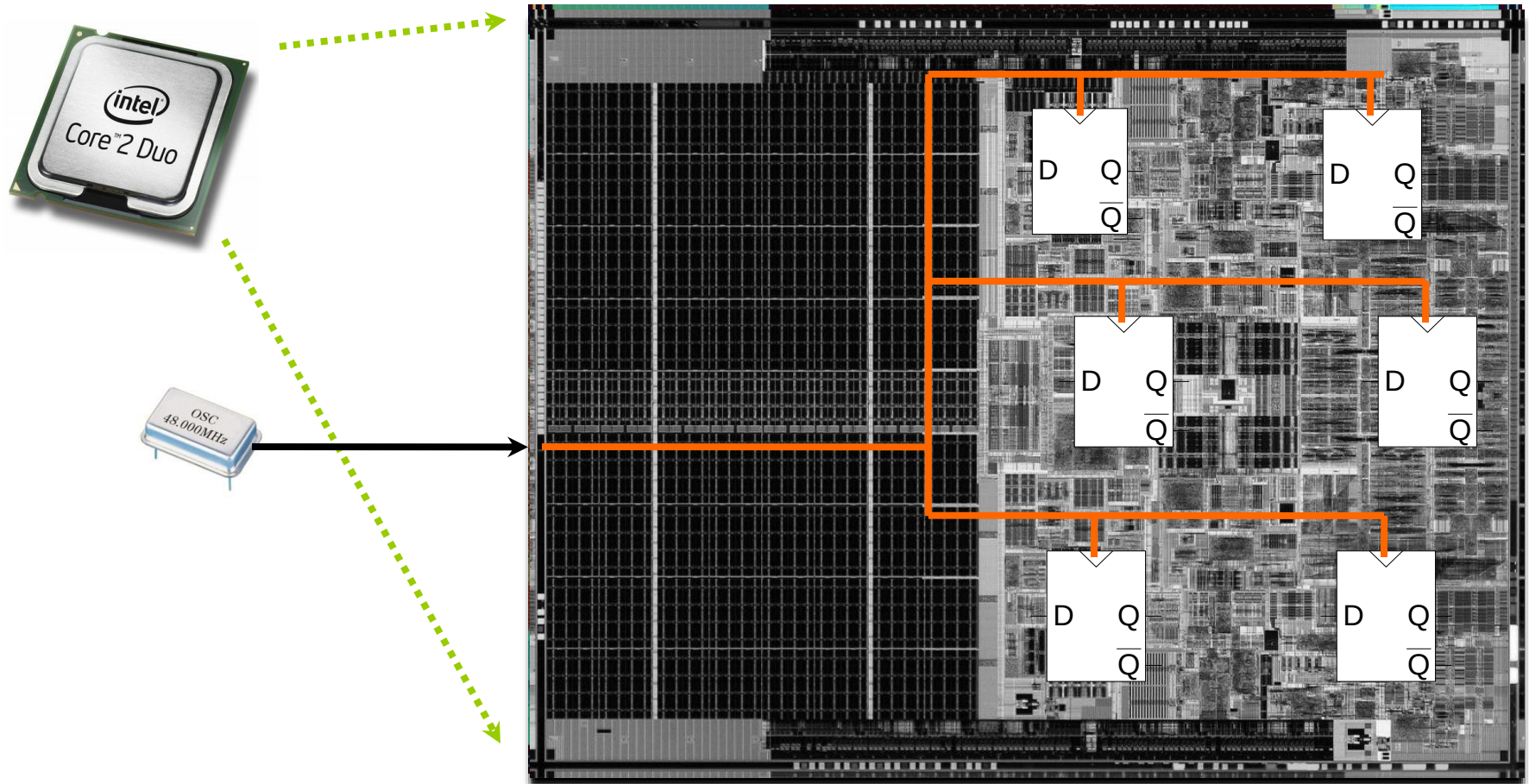
- Virtually **all digital systems** are essentially operating synchronous to the clock



Where are Clock Oscillators?



Clock in Digital Circuit



Synchronous Sequential Logic

- Output of sequential logic is determined not only by current inputs but also by state stored in registers
- When sequential logic is working (updated) at the event (e.g., rising or falling edge) of clock source, we say that the circuit is synchronous to the clock
 - In other words, if the state is updated at the event of clock source, the circuit is **synchronous sequential logic**
- Virtually all digital systems are essentially synchronous to the clock
 - Virtually all digital systems are synchronous sequential logic

Synchronous Sequential Logic

- Synchronous sequential logic 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**
 - will talk in depth about pipelining in computer architecture course next semester

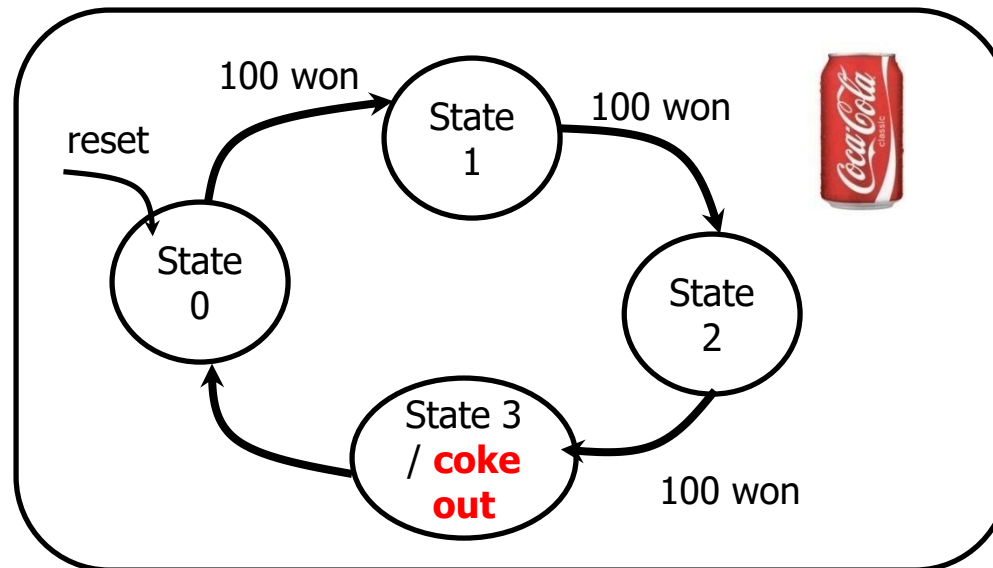
Finite State Machine (FSM)

- Finite state machine (FSM) is composed of 2 components: registers and combinational logic
 - Register represents one of the **finite** number of states
 - ***K-bit* register** can represent **one of a finite number (2^K) of unique states**
 - **An initial state (in register)** is assigned based on **reset** input at the (rising or falling) edge of clock
 - The **next state** may change depending on the current state as the next input comes in
 - Based on the current state (and input), **output** is determined via combinational logic

FSM Quick Example

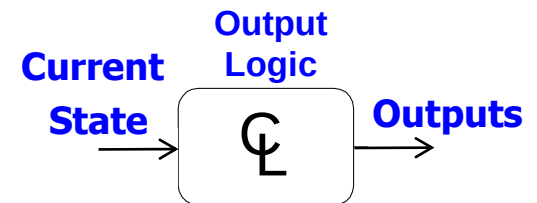
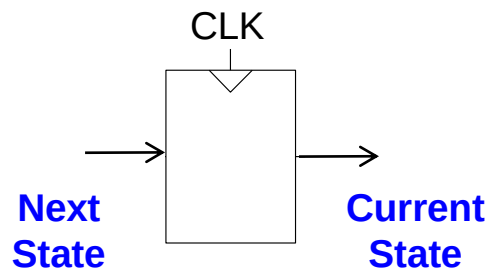
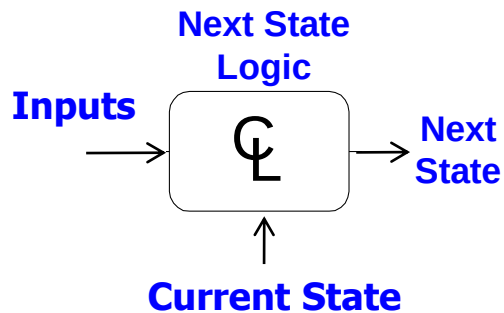
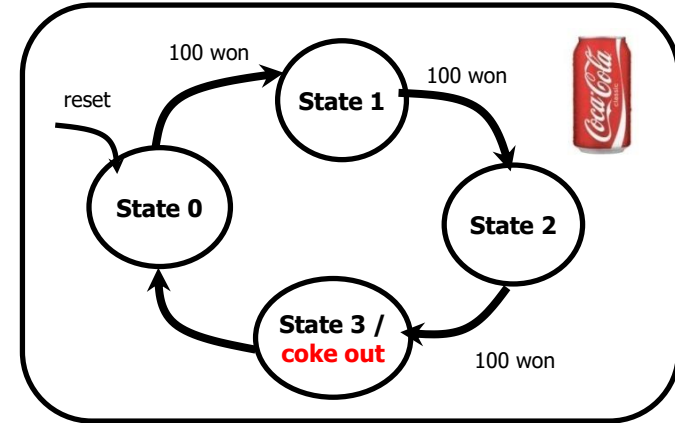
- Vending machine

- You are asked to design a vending machine to sell cokes.
 - Suppose that a coke costs 300 won
 - The machine takes only 100 won coins
- How would you design a logic with inputs and output?



Finite State Machine (FSM)

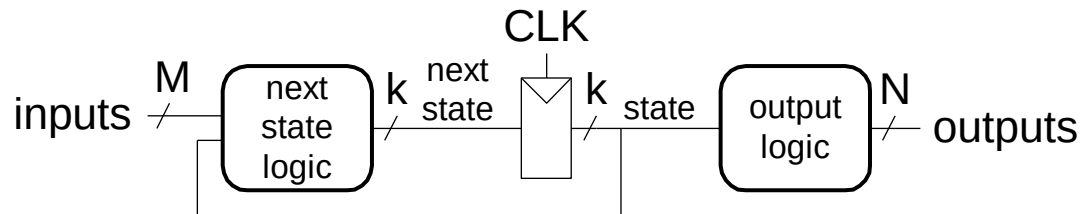
- FSM is composed of
 - State register**
 - Stores the current state
 - Loads the next state at the clock edge
 - Combinational logic**
 - Computes the **next state** based on current state and input
 - Computes the **outputs** based on current state (and input)



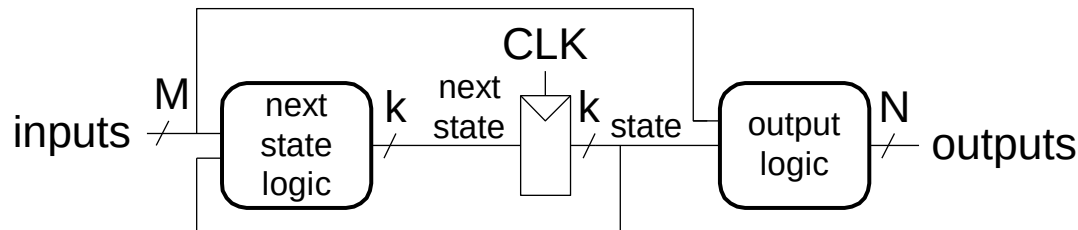
Finite State Machines (FSMs)

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

Moore FSM



Mealy FSM

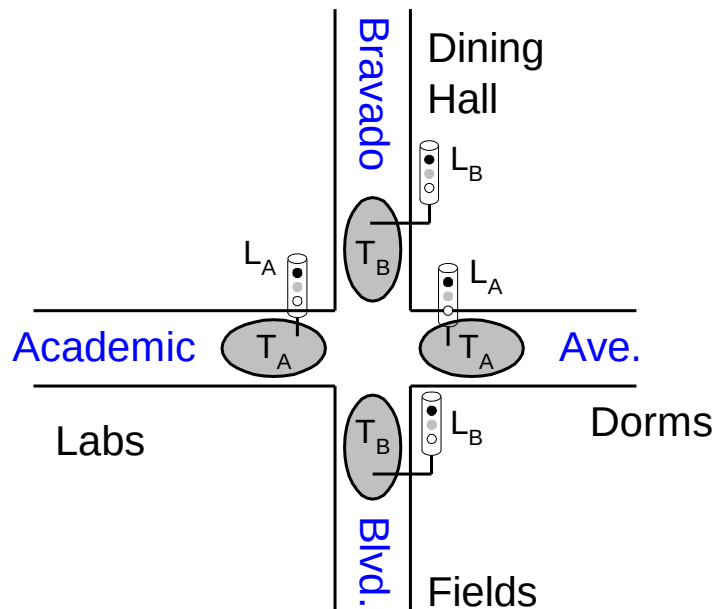


Moore and Mealy

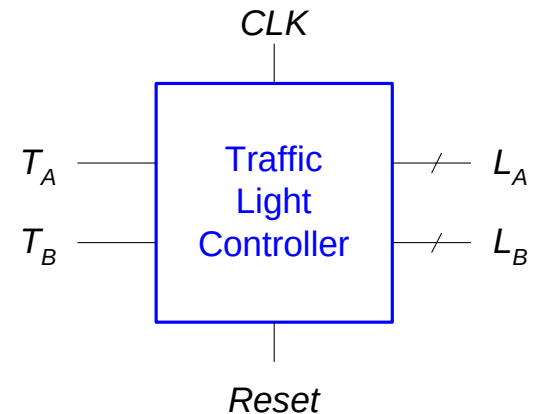
- Edward F. Moore, 1925 - 2003
 - Together with Mealy, developed automata theory, the mathematical underpinnings of state machines, at Bell Labs.
 - Not to be confused with Intel founder Gordon Moore
 - Published a seminal article, Gedanken-experiments on Sequential Machines in 1956
- George H. Mealy
 - Published "A Method of Synthesizing Sequential Circuits" in 1955
 - Wrote the first Bell Labs operating system for the IBM 704 computer

Finite State Machine Example

- Let's design a simplified traffic light controller
 - Traffic sensors (sensing human traffic): T_A , T_B
 - Each sensor becomes TRUE if students are present
 - Each sensor becomes FALSE if students are NOT present (i.e., the street is empty)
 - Lights: L_A , L_B
 - Each light receives digital inputs specifying whether it should be green, yellow, or red

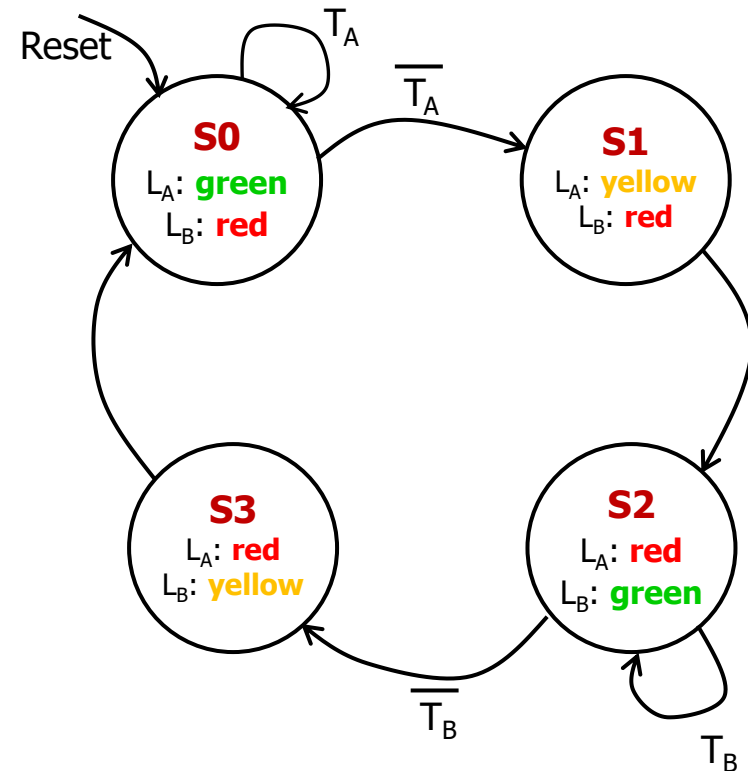
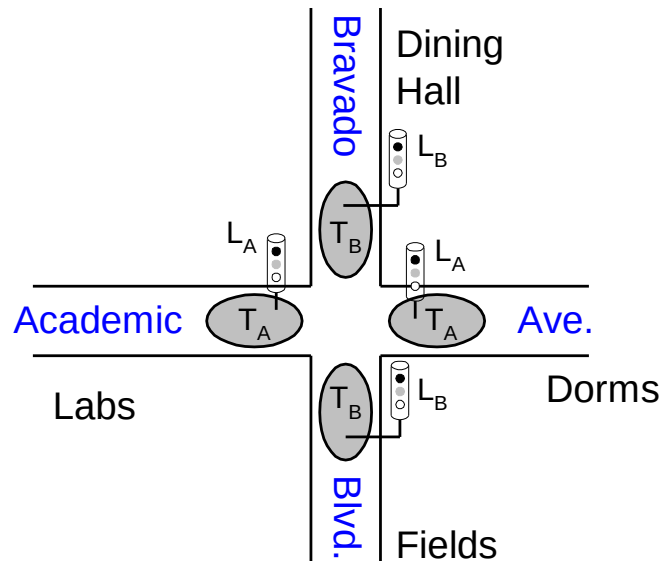


Inputs: clk , $Reset$, T_A , T_B
Outputs: L_A , L_B

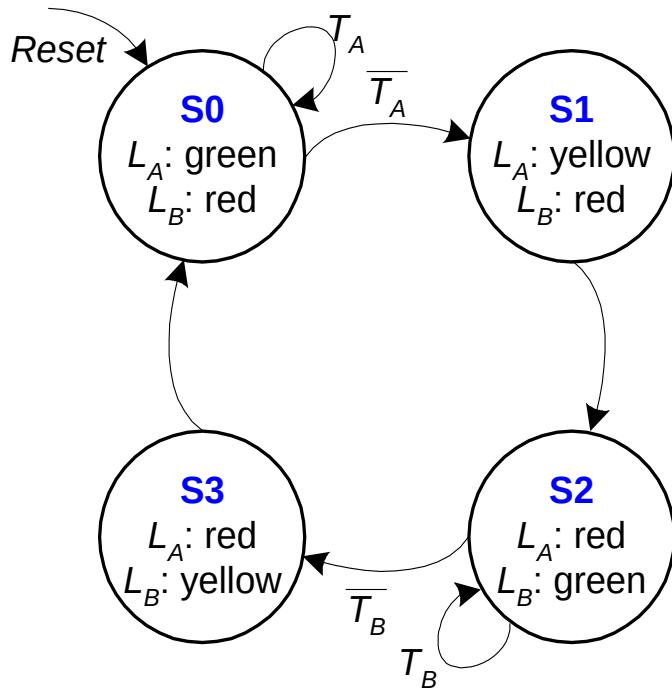


FSM State Transition Diagram

- Moore FSM
 - Circles represent states
 - Arcs represent transitions between states
 - Outputs are labeled in each state



FSM 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

FSM Encoded State Transition Table

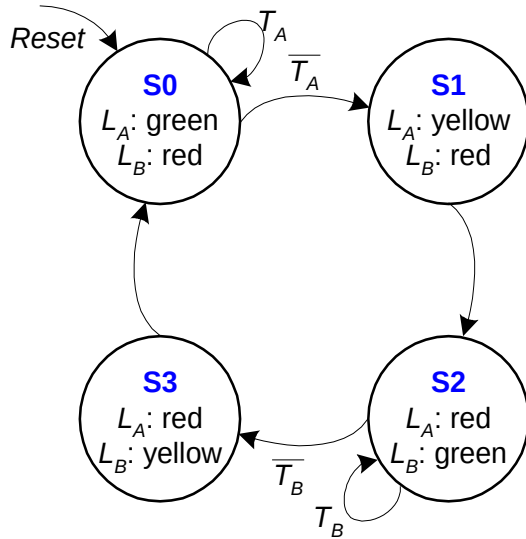
State	Encoding
S0	00
S1	01
S2	10
S3	11

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

$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S_1} \overline{S_0} \overline{T_A} + S_1 \overline{S_0} \overline{T_B}$$

FSM 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	Encoding
green	00
yellow	01
red	10

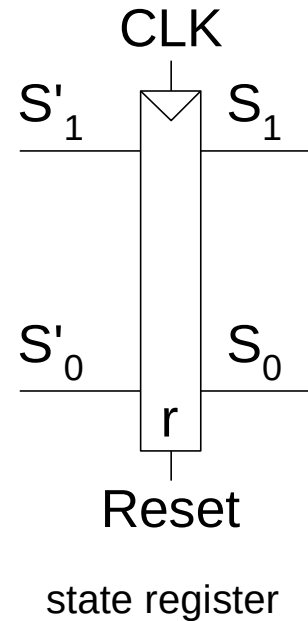
$$L_{A1} = S_1$$

$$L_{B1} = \overline{S_1}$$

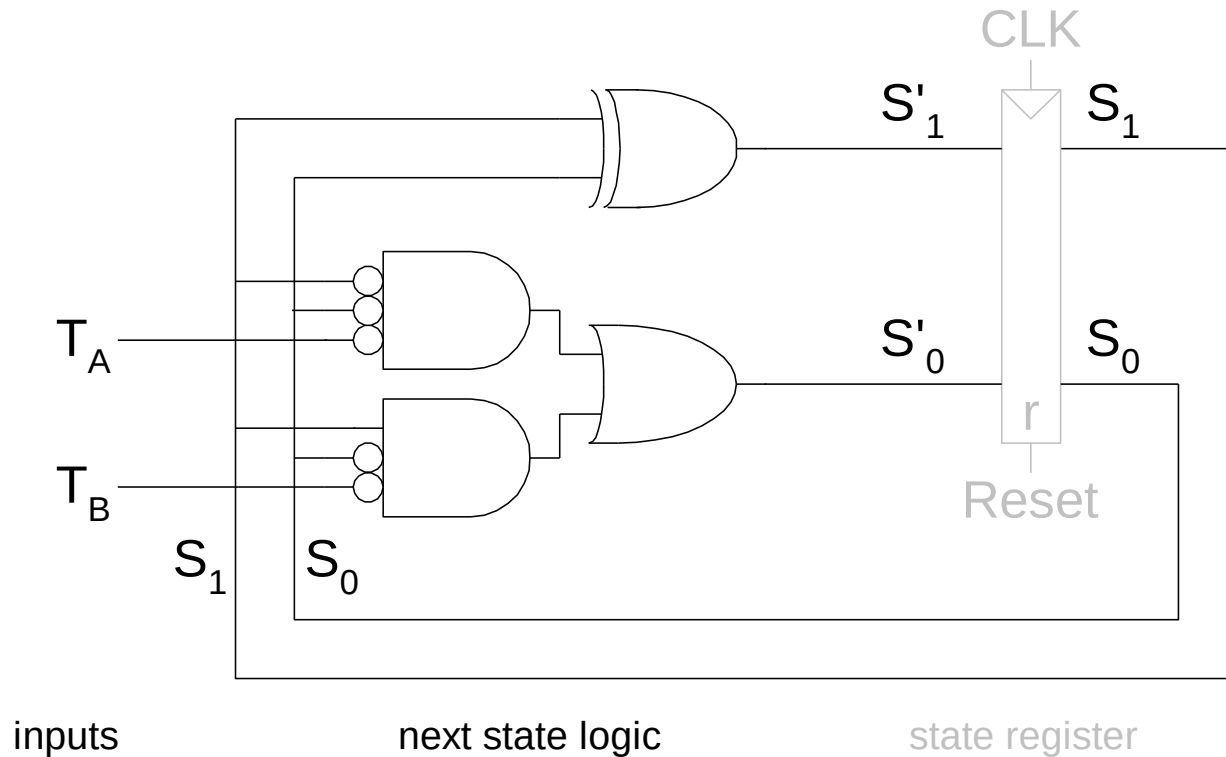
$$L_{A0} = \overline{S_1}S_0$$

$$L_{B0} = S_1S_0$$

FSM Schematic: State Register



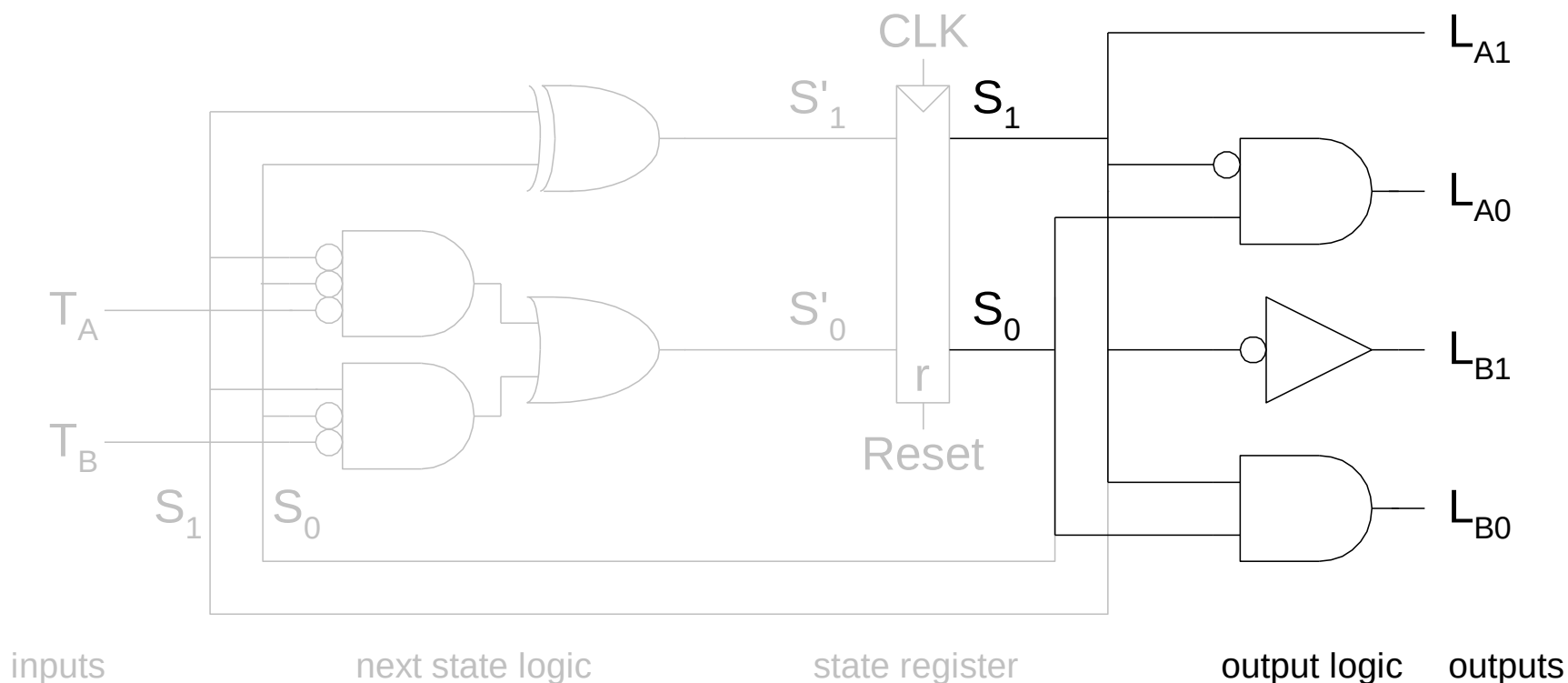
FSM Schematic: Next State Logic



$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S_1} \overline{S_0} \overline{T_A} + S_1 \overline{S_0} \overline{T_B}$$

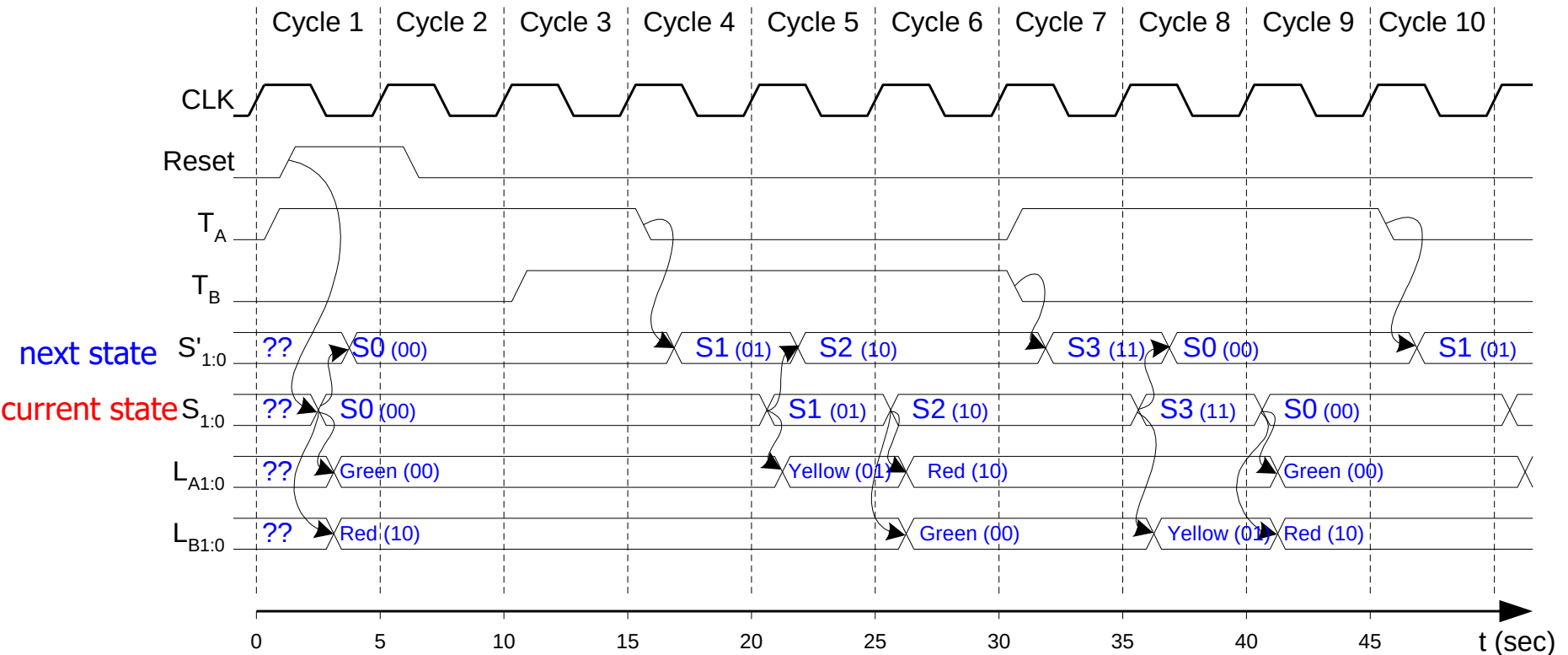
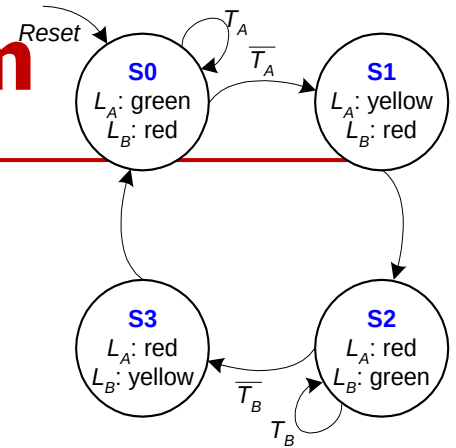
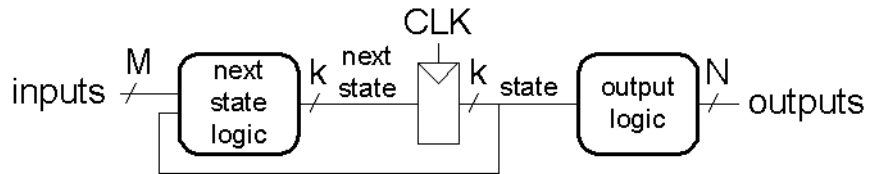
FSM Schematic: Output Logic



$$\begin{aligned} L_{A1} &= \overline{S_1} \\ L_{A0} &= \overline{S_1} S_0 \\ L_{B1} &= S_1 \\ L_{B0} &= S_1 S_0 \end{aligned}$$

FSM Timing Diagram

Moore FSM



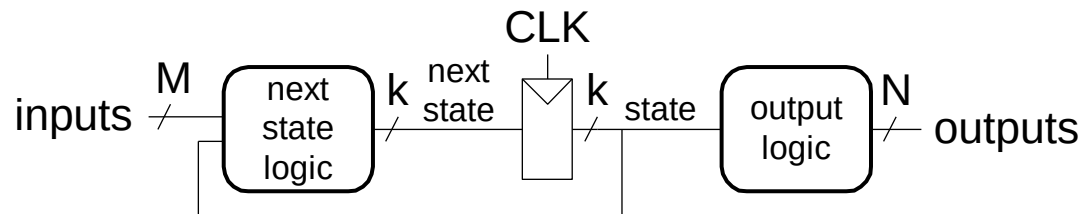
FSM State Encoding

- In the previous example, the state and output encodings were selected arbitrarily
 - Different choice would have resulted in a different circuit
- Commonly used encoding methods
 - **Binary encoding**
 - Each state is represented as a binary number
 - For example, to represent four states, we need 2 bits (00, 01, 10, 11)
 - **One-hot encoding**
 - A separate bit is used for each state
 - Only one bit is HIGH at once (one-hot)
 - For example, to represent four states, we need 4 bits (0001, 0010, 0100, 1000)
 - So, it requires more flip-flops
 - But, it often results in simpler next state and output logic

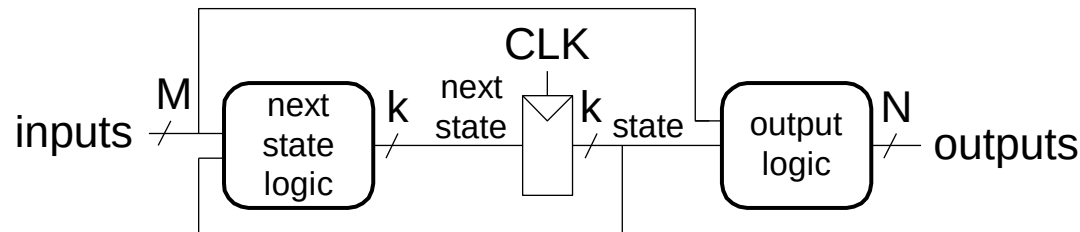
Moore vs. Mealy FSM

- Two types of FSMs 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

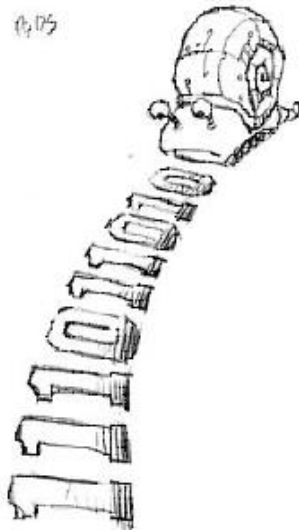


Mealy FSM



Snail Example

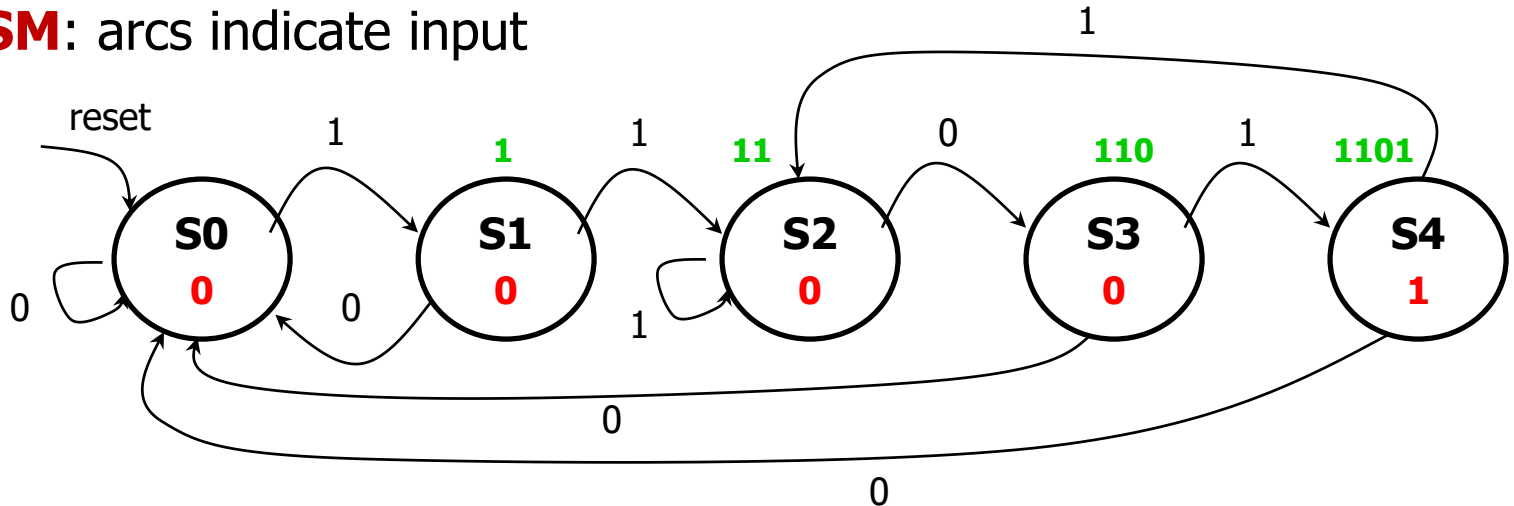
- There is a snail
 - The snail crawls down a paper tape with 1's and 0's on it
 - The snail smiles whenever the last four numbers it has crawled over are **1101**
- Design Moore and Mealy FSMs of the snail's brain



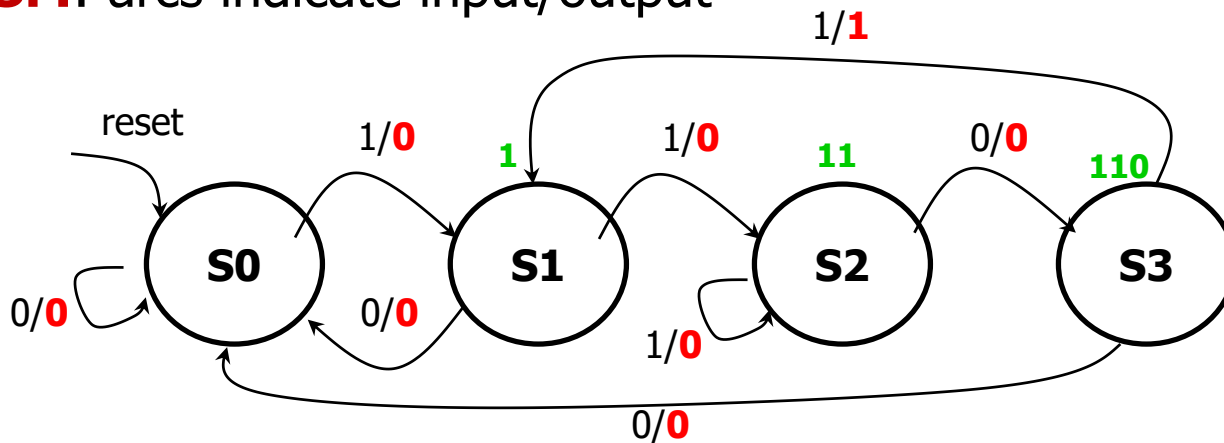
State Transition Diagrams

(1101)

Moore FSM: arcs indicate input

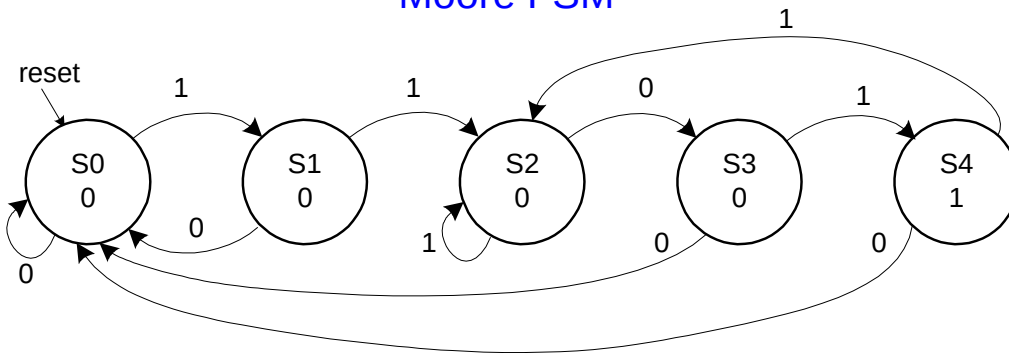


Mealy FSM: arcs indicate input/output



Moore FSM State Transition Table

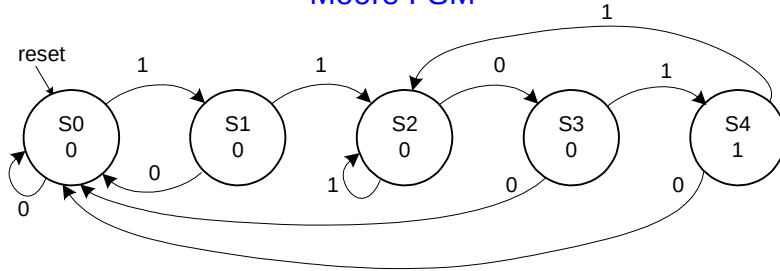
Moore FSM



Current State	Inputs	Next State
<i>S</i>	<i>A</i>	<i>S'</i>
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S2
S2	0	S3
S2	1	S2
S3	0	S0
S3	1	S4
S4	0	S0
S4	1	S2

Moore FSM State Transition Table

Moore FSM



State	Encoding
S0	000
S1	001
S2	010
S3	011
S4	100

$$S'_2 = S_1 S_0 A$$

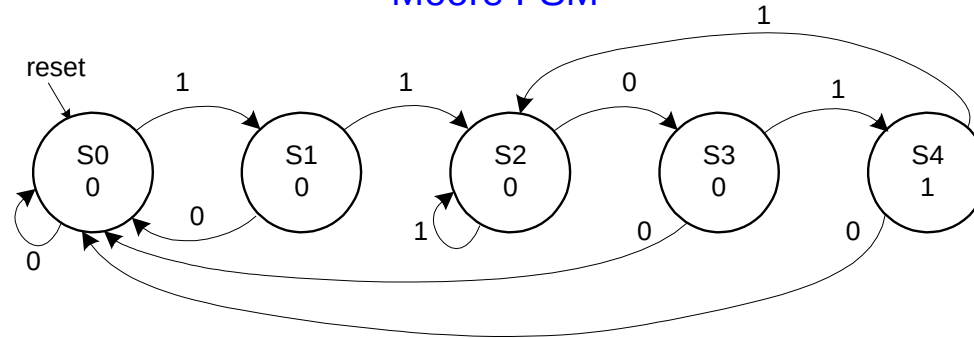
$$S'_1 = \overline{S_1} S_0 A + S_1 \overline{S_0} + S_2 A$$

$$S'_0 = \overline{S_2} \overline{S_1} \overline{S_0} A + S_1 \overline{S_0} \overline{A}$$

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

Moore FSM Output Table

Moore FSM



	Current State			Output
	S_2	S_1	S_0	Y
S0	0	0	0	0
S1	0	0	1	0
S2	0	1	0	0
S3	0	1	1	0
S4	1	0	0	1

$$Y = S_2$$

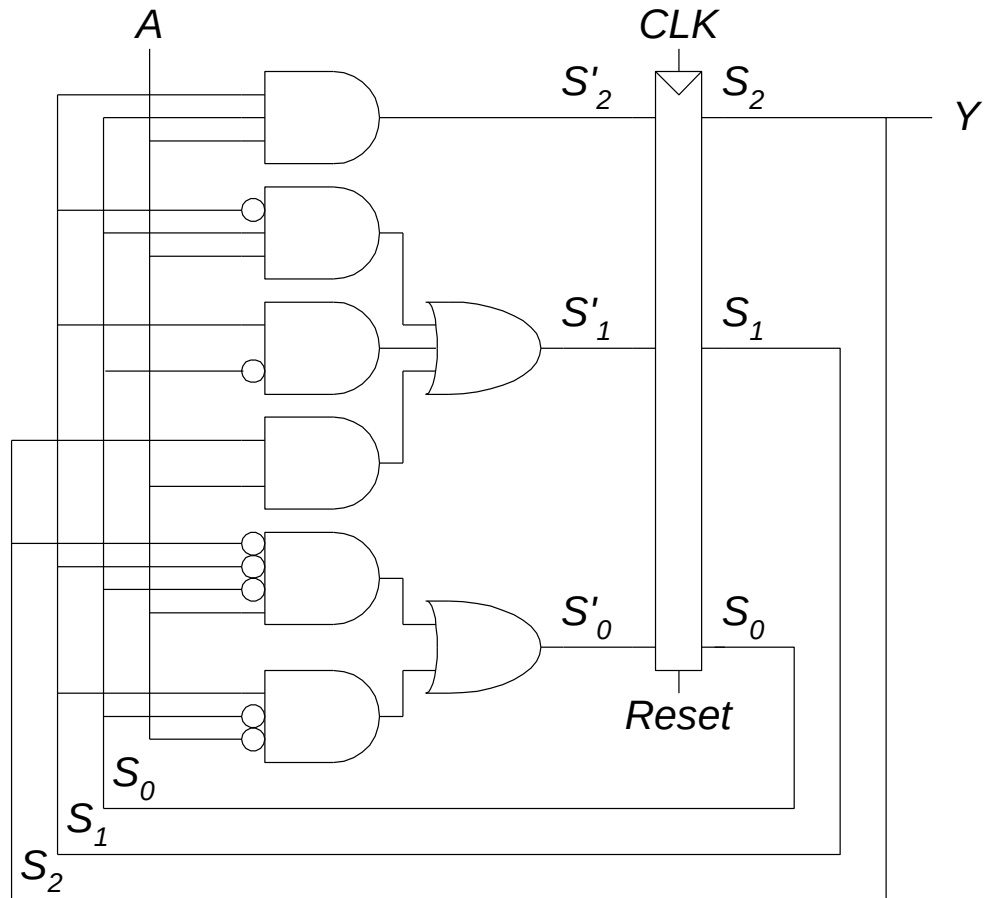
Moore FSM Schematic

$$S'_2 = S_1 S_0 A$$

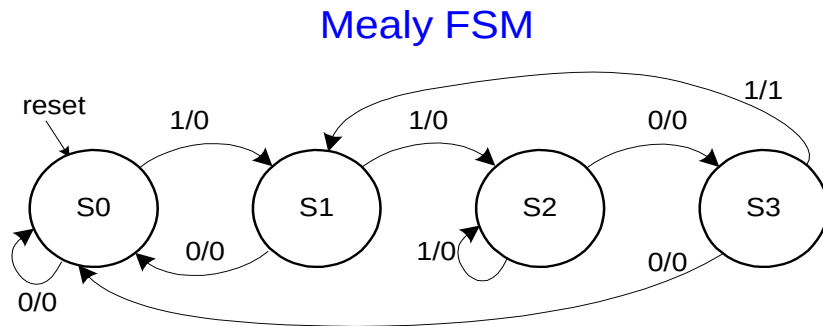
$$S'_1 = \overline{S_1} S_0 A + S_1 \overline{S_0} + S_2 A$$

$$S'_0 = \overline{S_2} \overline{S_1} \overline{S_0} A + S_1 \overline{S_0} \overline{A}$$

$$Y = S_2$$



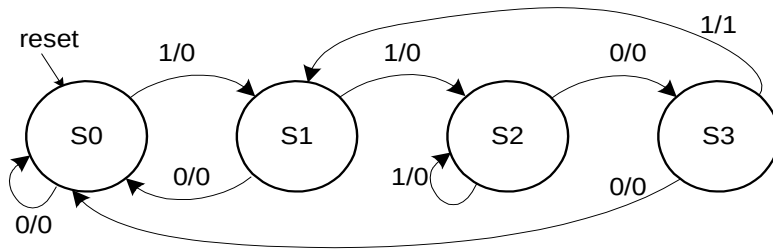
Mealy FSM State Transition and Output Table



Current State	Inputs	Next State
<i>S</i>	<i>A</i>	<i>S'</i>
S0	0	S0
S0	1	S1
S1	0	S0
S1	1	S2
S2	0	S3
S2	1	S2
S3	0	S0
S3	1	S1

Mealy FSM State Transition and Output Table

Mealy FSM



State	Encoding
S0	00
S1	01
S2	10
S3	11

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

$$S'_1 = S_1 \overline{S_0} + \overline{S_1} S_0 A$$

$$S'_0 = \overline{S_1} \overline{S_0} A + \overline{S_1} S_0 \overline{A} + S_1 S_0 A$$

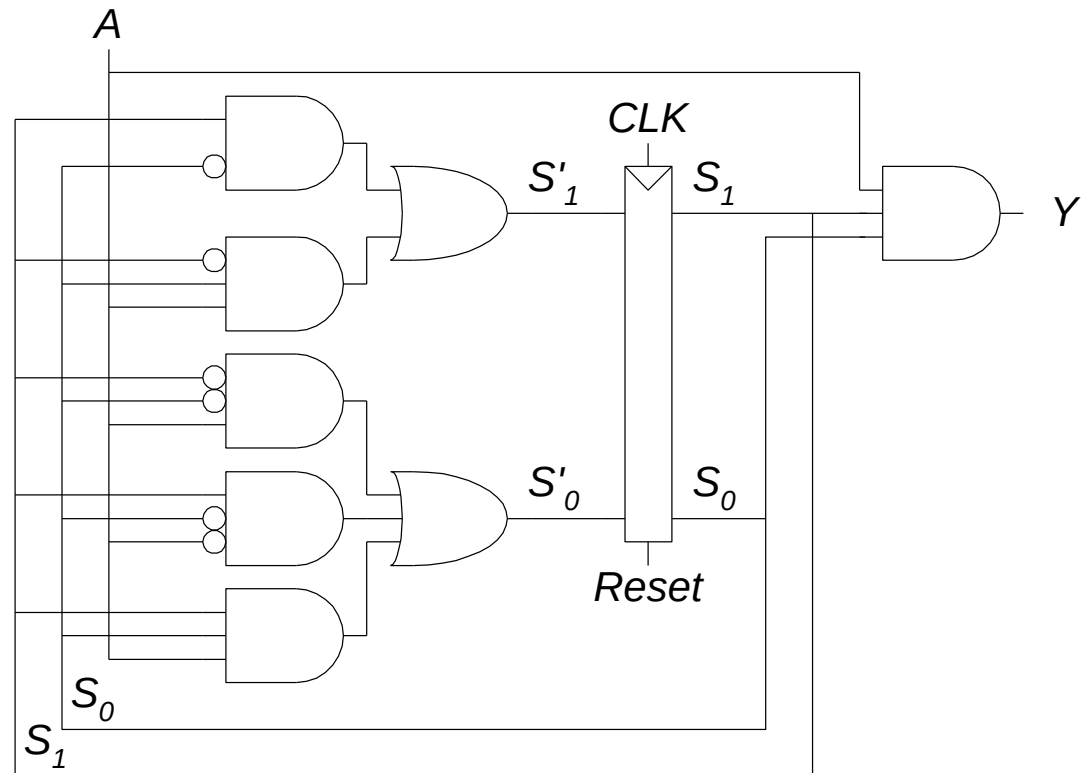
$$Y = S_1 S_0 A$$

Mealy FSM Schematic

$$S'_1 = S_1 \overline{S_0} + \overline{S_1} S_0 A$$

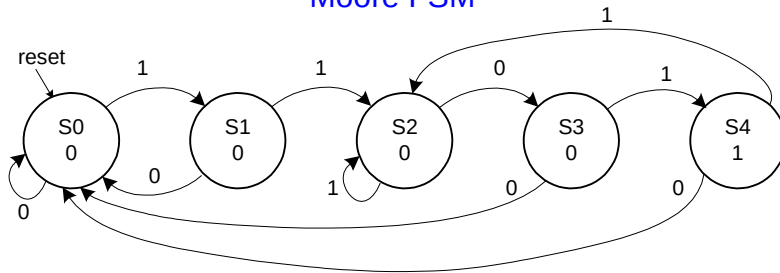
$$S'_0 = \overline{S_1} S_0 A + S_1 \overline{S_0} \overline{A} + S_1 S_0 A$$

$$Y = S_1 S_0 A$$

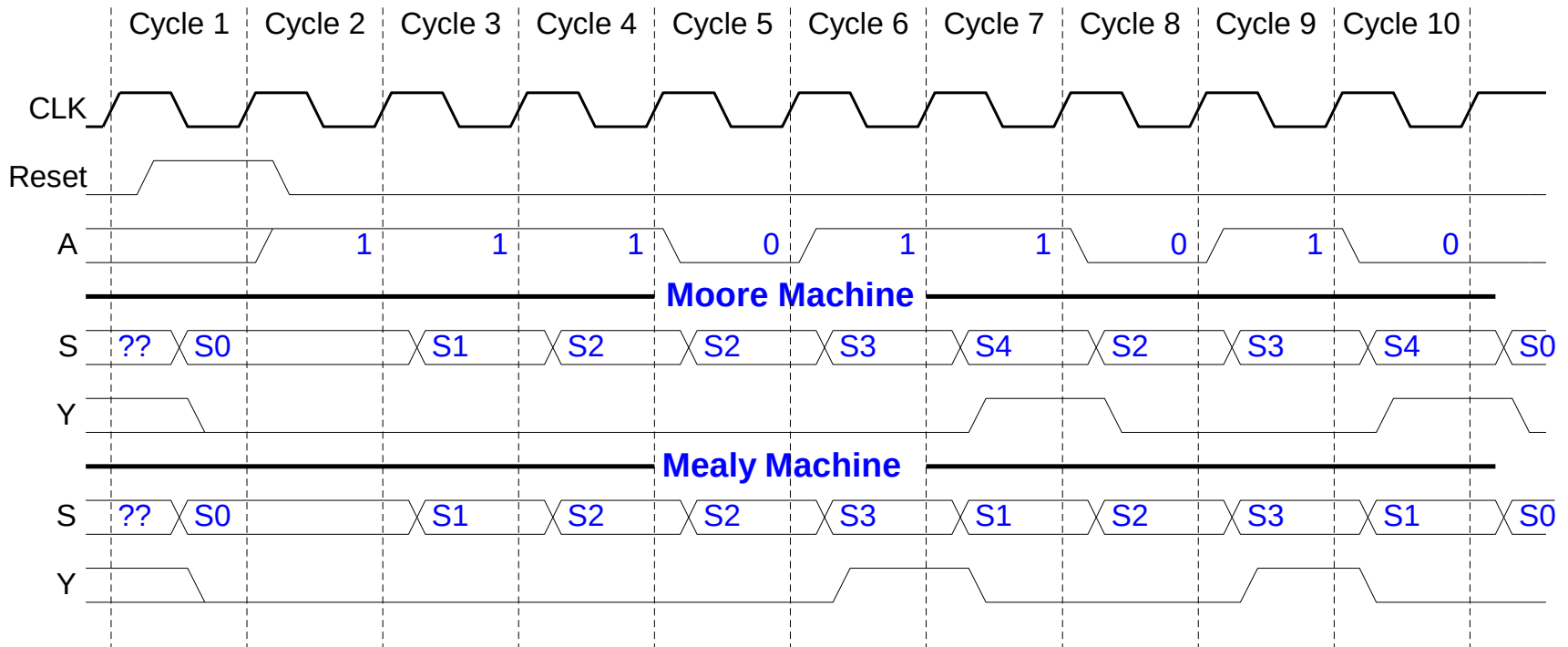
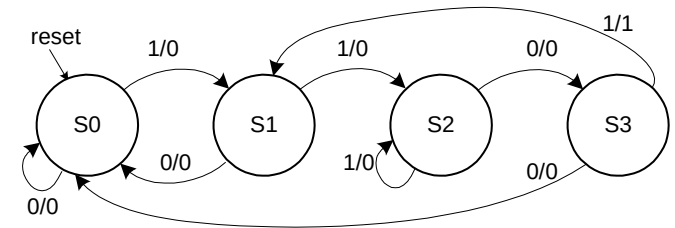


Moore and Mealy Timing Diagram

Moore FSM



Mealy FSM



Difference between Moore and Mealy

- A Moore machine typically has more states than a Mealy machine for a given problem
- A Mealy machine's output rises a cycle sooner because it responds to the input rather than waiting for the state change
- When choosing your FSM design style, consider when you want your outputs to respond

FSM Design Procedure

- Identify 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 state encodings
 - Write the output table
- For a Mealy machine
 - Rewrite the combined state transition table and output table with the state encodings
- Write Boolean equations for the next state and output logic
- Sketch the circuit schematic