RAMAIAH
SKILL ACADEMY

Finite State Machines

16th -17th December 2024

By: PRIYANKA C
Application Engineer

WWW.MSRUAS.AC.IN

Contents

Introduction to FSM

What , Why and How?

What is FSM?

- “FSM is a mathematical model that is used to design systems that transition between finite states based on inputs”
- It consists of finite number of states, hence the name.
- A set of states, a set of input events, a set of output actions or reactions, and a set of state transitions based on input events forms the fundamental components of an FSM.
- Example : Robot ---Idle, Moving, Cleaning etc

Components of FSM?

- **States** → represents systems different conditions .

  example: Traffic lights can be Red, Green and Yellow
- **Inputs** → External signals that causes transitions between states

  example: sensors or timer
- **Transitions**→ Rules /Conditions for moving between states

  example: After Red, move to Yellow then move to Green
- **Outputs** → Signals generated based on current state

  example: which light is on at the time(Red, Yellow or Green)
- **Clock** → that helps in synchronizing states

Why do we need FSM?

Finite State Machines (FSMs) are critically important because they provide a structured way to model and control the sequential behavior of digital systems.

1. Controls Logic in Sequential Systems

→Essential for managing state-based operations.

→Example: Processor instruction decoding.

2. Simplifies Complex System Design

→Breaks large systems into smaller, manageable states.

→ Example: UART protocol state machine.

3. Critical in VLSI SoC Design

→Drives efficiency, reliability, and modularity.

→Example: Communication Protocols (UART, I2C, SPI).

4. **Deterministic and Predictable Behaviour**

→ Outputs and transitions are well-defined for any given input.

Types of FSM

- **Deterministic FSM**
  - In a deterministic FSM, the next state is uniquely defined by the current state and the input.
  - This means that for any given input and current state, there is only one possible transition to the next state.
  - ***Example: Traffic Light Controller***

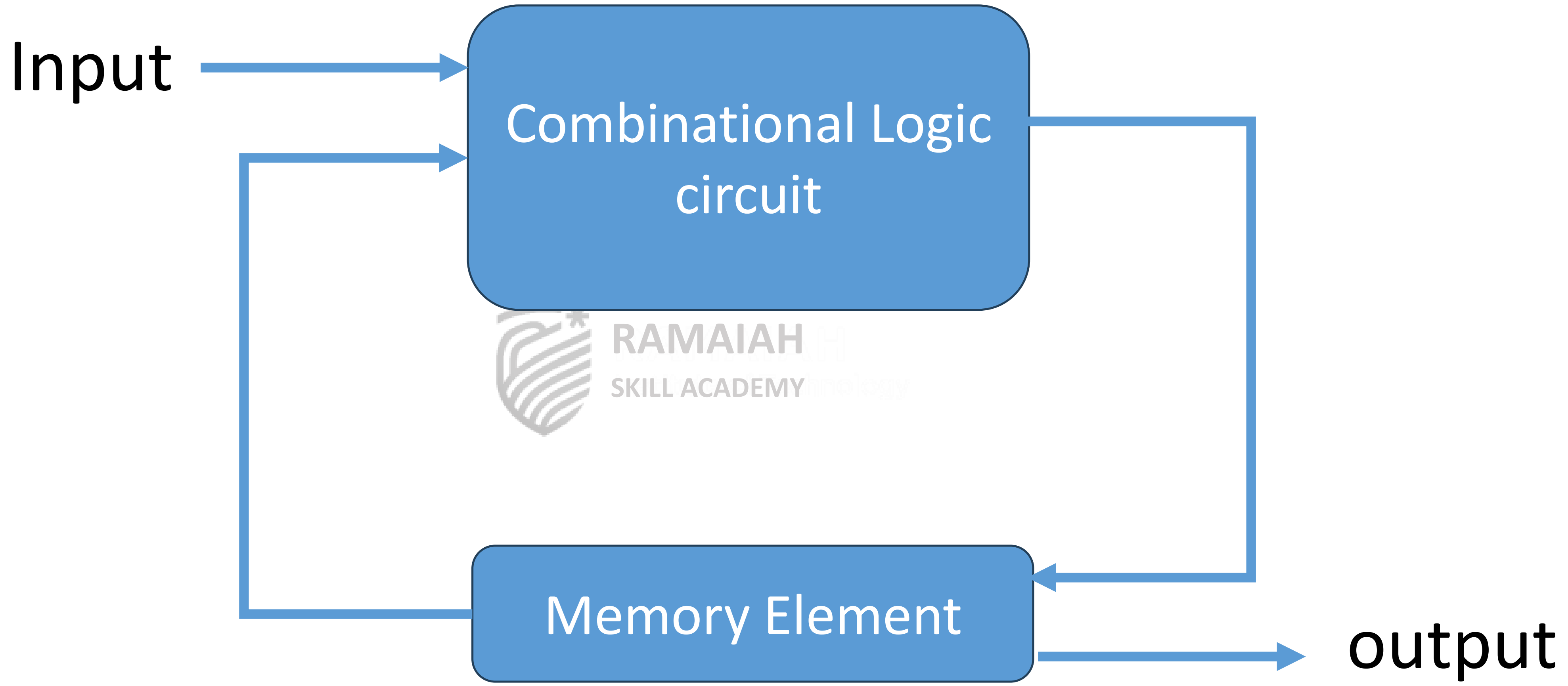
- **Non-Deterministic FSM**
  - Non-deterministic FSMs allow for multiple possible transitions from the same state, based on the same input.
  - This means the machine can simultaneously exist in more than one state.
  - ***Example: Elevator System***

Types of FSM

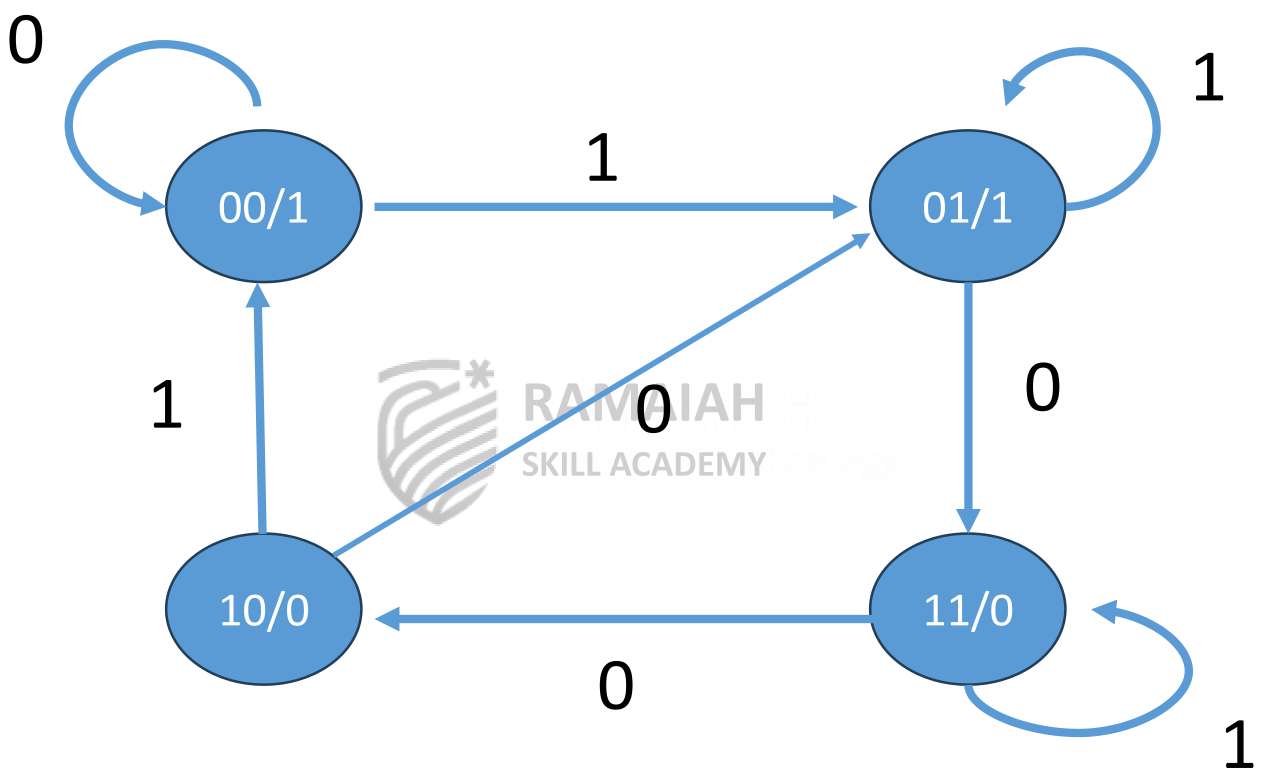
- **Mealy FSM**
  - A Mealy machine produces outputs that depend on both the current state and the current input.
  - The output is generated during a state transition, making it more responsive to changes in the input.
  - ***Example: Temperature control system, coin operated turnstile***

- **Moore FSM**
  - In Moore machine, the outputs depend only on the current state.
  - Regardless of the inputs, the output is fixed for a given state.
  - Transitions between states are still triggered by inputs, but the outputs are tied to the state itself.
  - ***Example: Washing Machine controller***

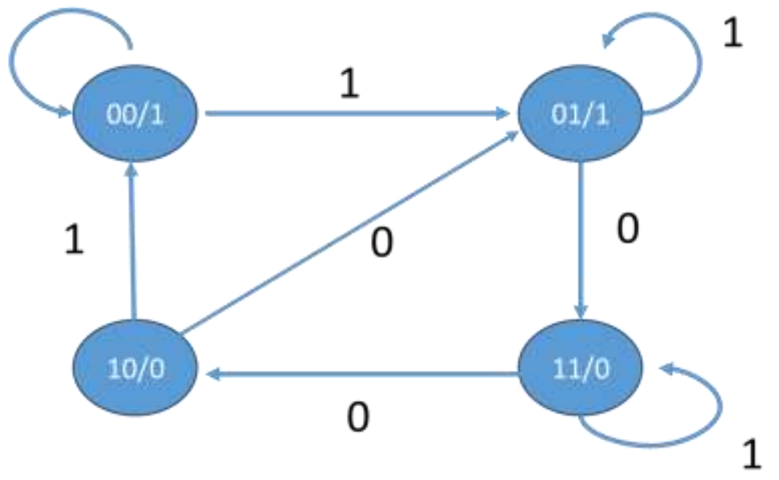
Moore FSM



Moore FSM State Machine



Moore Machine State Table



Input		PS			NS			Output
X		Q1	Q0		Q1	Q0		Y
0		0	0		0	0		1
0		0	1		1	1		0
0		1	0		0	1		1
0		1	1		1	0		0
1		0	0		0	1		1
1		0	1		0	1		1
1		1	0		0	0		1
1		1	1		1	1		0

Moore Machine State Equations

Input		PS			NS			Output
X		Q1	Q0		Q1	Q0		Y
0		0	0		0	0		1
0		0	1		1	1		0
0		1	0		0	1		1
0		1	1		1	0		0
1		0	0		0	1		1
1		0	1		0	1		1
1		1	0		0	0		1
1		1	1		1	1		0

X \ Q1Q0	00	01	11	10
0	0	1	1	0
1	0	0	1	0

$Q1' = X' Q0 + Q1 Q0$

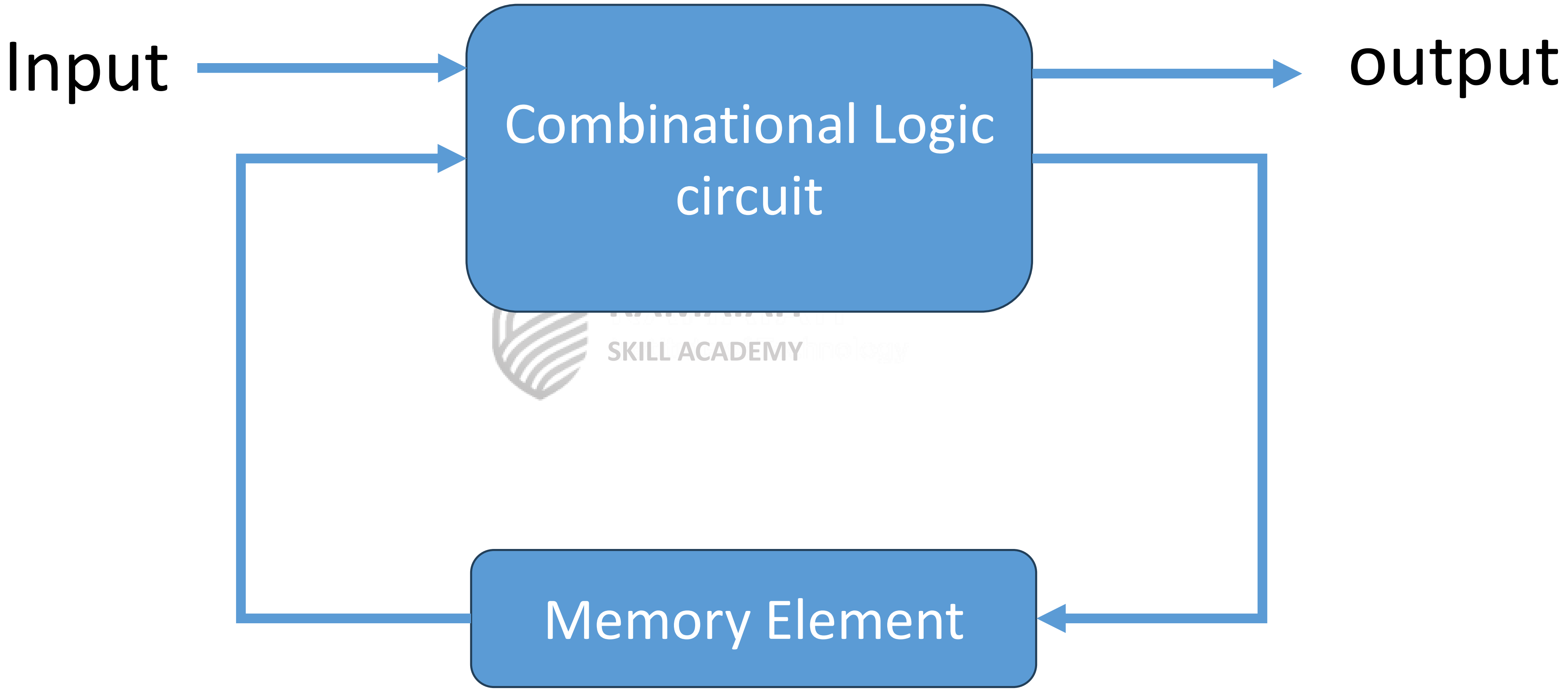
X \ Q1Q0	00	01	11	10
0	1	0	0	1
1	1	1	0	1

$Y = Q0' + X Q1'$

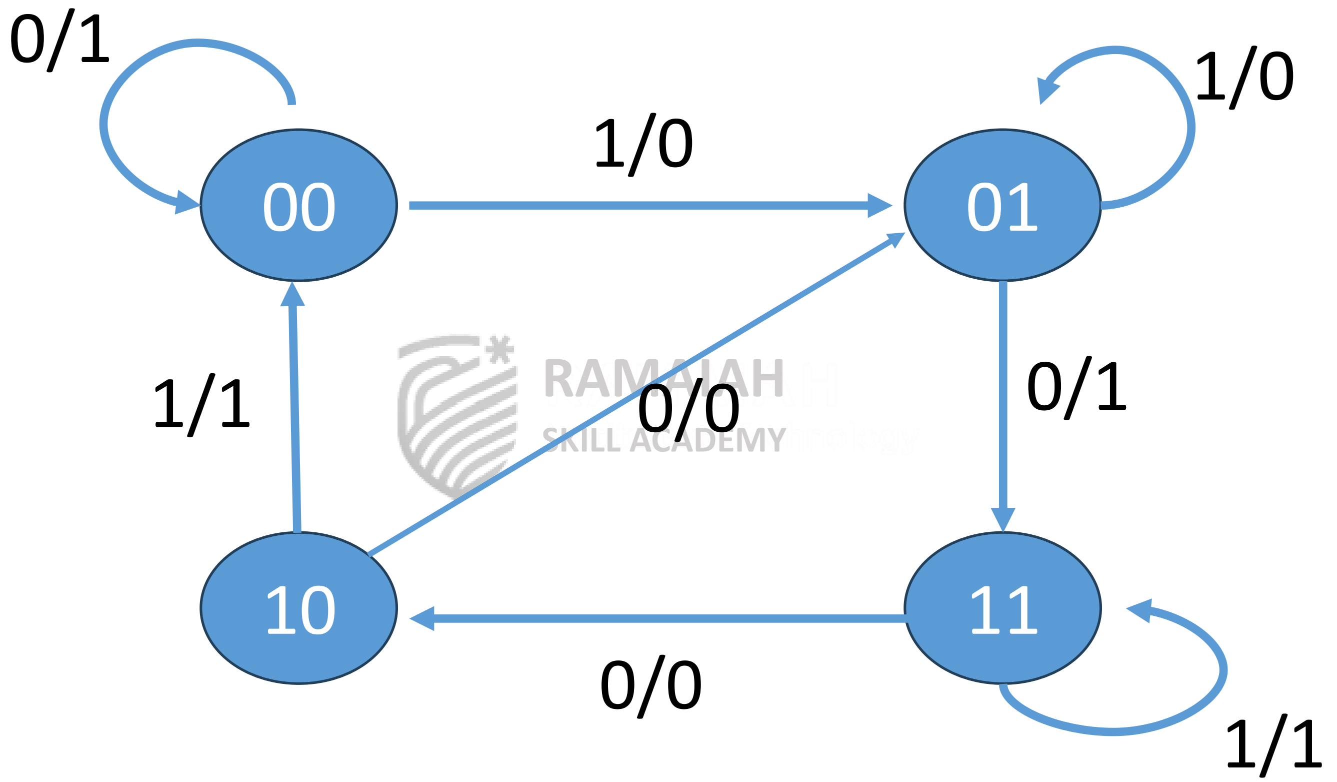
X \ Q1Q0	00	01	11	10
0	0	1	0	1
1	1	1	1	0

$Q0' = Q1' + X Q0$

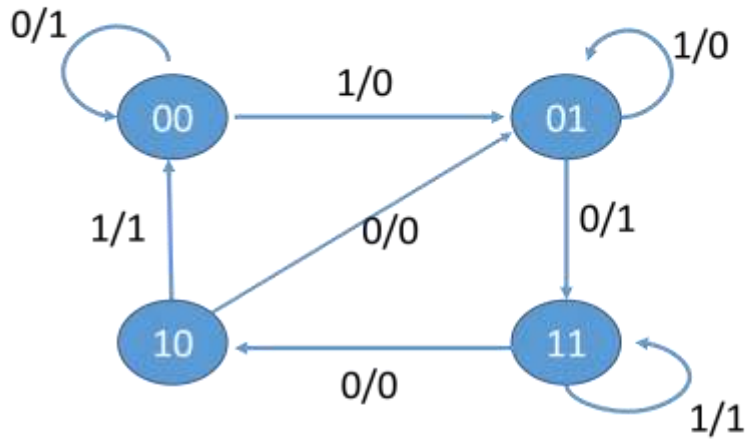
Mealy FSM



Mealy FSM State Machine



Mealy Machine State Table



Input		PS			NS			Output
X		Q1	Q0		Q1	Q0		Y
0		0	0		0	0		1
0		0	1		1	1		1
0		1	0		0	1		0
0		1	1		1	0		0
1		0	0		0	1		0
1		0	1		0	1		0
1		1	0		0	0		1
1		1	1		1	1		1

Mealy Machine State Equations

Input		PS			NS			Out put
X		Q1	Q0		Q1	Q0		Y
0		0	0		0	0		1
0		0	1		1	1		1
0		1	0		0	1		1
0		1	1		1	0		0
1		0	0		0	1		0
1		0	1		0	1		0
1		1	0		0	0		1
1		1	1		1	1		1

X \ Q1Q0	00	01	11	10
0	0	1	1	0
1	0	0	1	0

Q1' =

X \ Q1Q0	00	01	11	10
0	0	1	0	1
1	1	1	1	0

Q0' =

X \ Q1Q0	00	01	11	10
0	1	0	0	1
1	1	1	0	1

Y =

Differences between Mealy and Moore Machine

Mealy Machine :

- Output depends on both the current state and input.
- Outputs can change immediately with input changes.
- State transitions are determined by inputs and state.
- Typically more complex due to dependency on both state and input for output.
- Can respond faster because the output is based on input, reducing delays.
- Requires additional logic for output generation based on inputs and states.

Moore Machine :

- Output depends only on the current state.
- Outputs change only on state transitions
- State transitions are determined only by the current state.
- Simpler because outputs depend only on the state.
- May have slower response time since the output only changes on state transitions.
- Easier to implement as outputs are directly linked to the state.

Applications of FSM

1. **Processors:** Instruction decoders, ALU controllers, pipeline stages.

2. **Controllers:** Memory controllers, DMA controllers, clock controllers.

3. **Communication Protocols:** UART, AXI, SPI, I2C controllers.

4. **Error Control:** CRCs, ECC circuits, and state-based error recovery mechanisms.

5. **Interface Logic:** Handshaking protocols in bus systems.

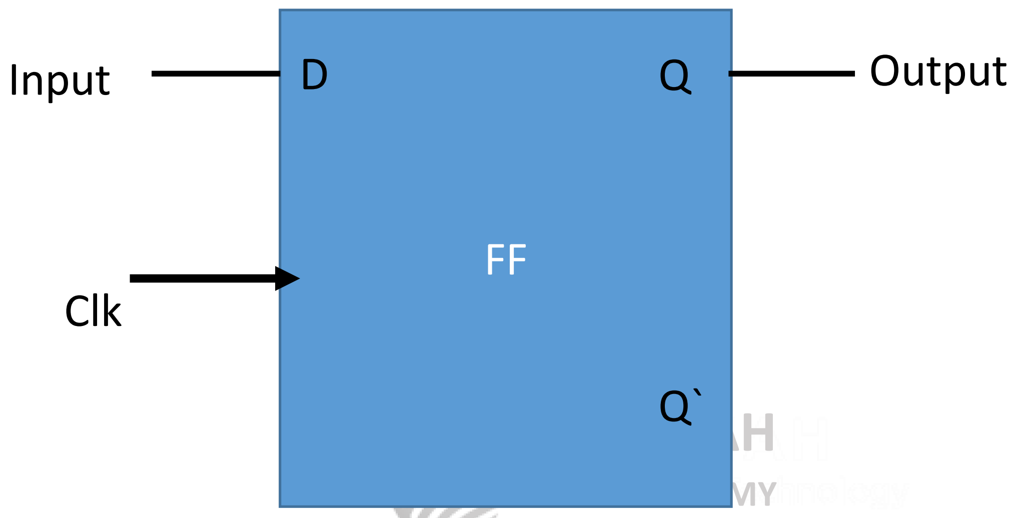
6. **State Machines in FPGAs:** Implementing sequential circuits on FPGAs using FSMs.

Design Flow of FSM

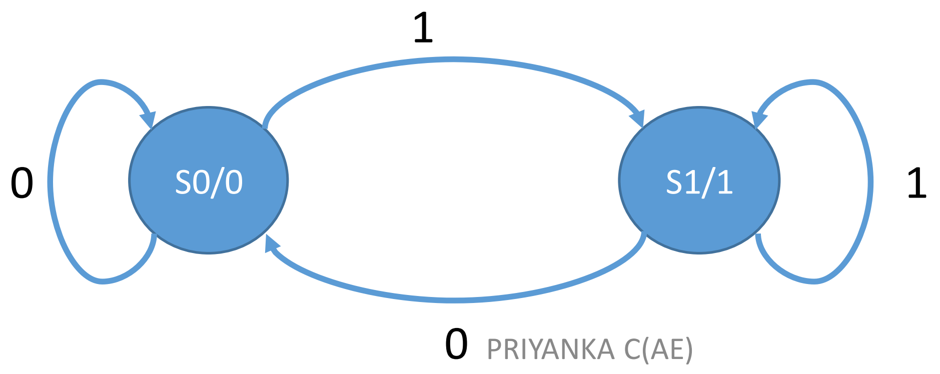
- *Define system requirements -- states and Transitions*
- *Create state transition table or diagram*
- *Choose the state encoding –one-hot, binary, gray*
- *Initialize the FSM*
- *Process input events*
- *Check the current state and input event*
- *Execute transition*
- *Update current state*

FSM for Sequential Circuits

D Flip-flop

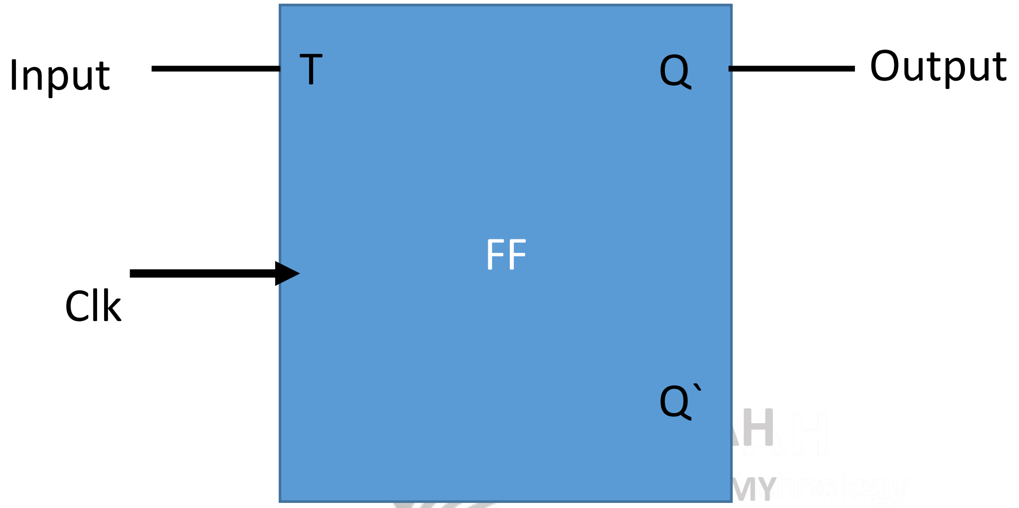


D	Q	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

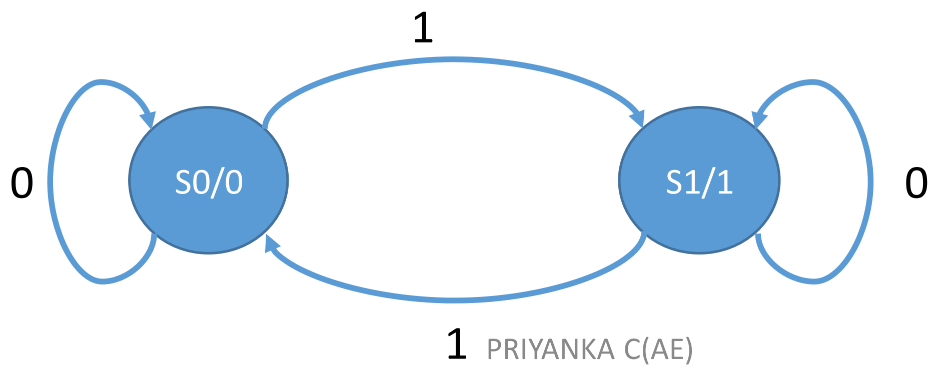


FSM for Sequential Circuits

T Flip-flop

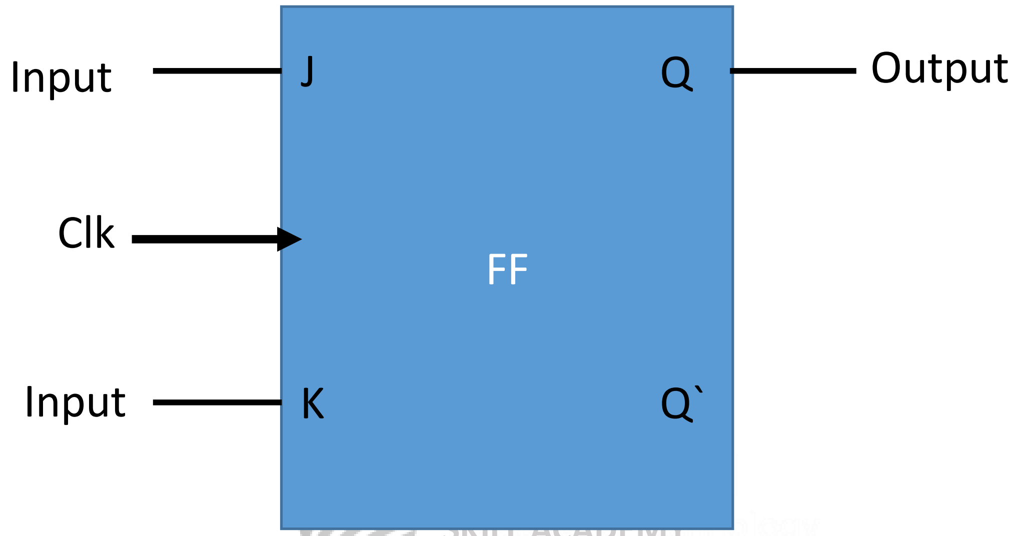


T	Q	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

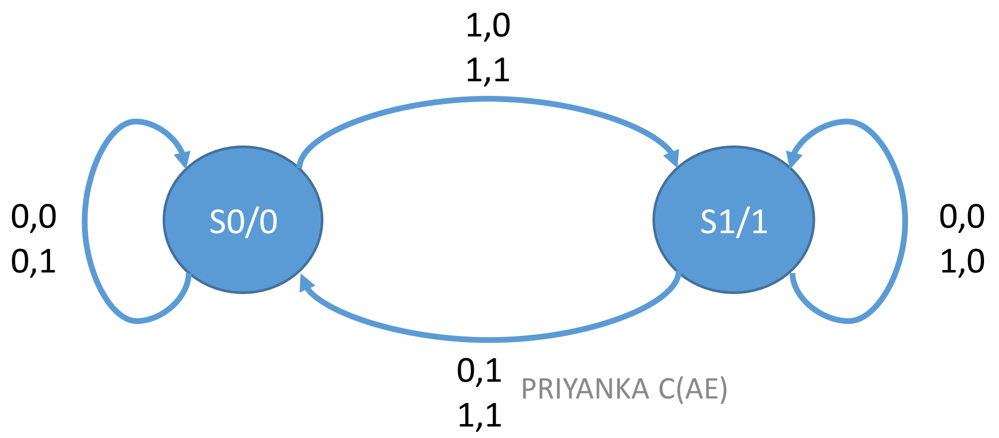


FSM for Sequential Circuits

JK Flip-flop



J	K	Q	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

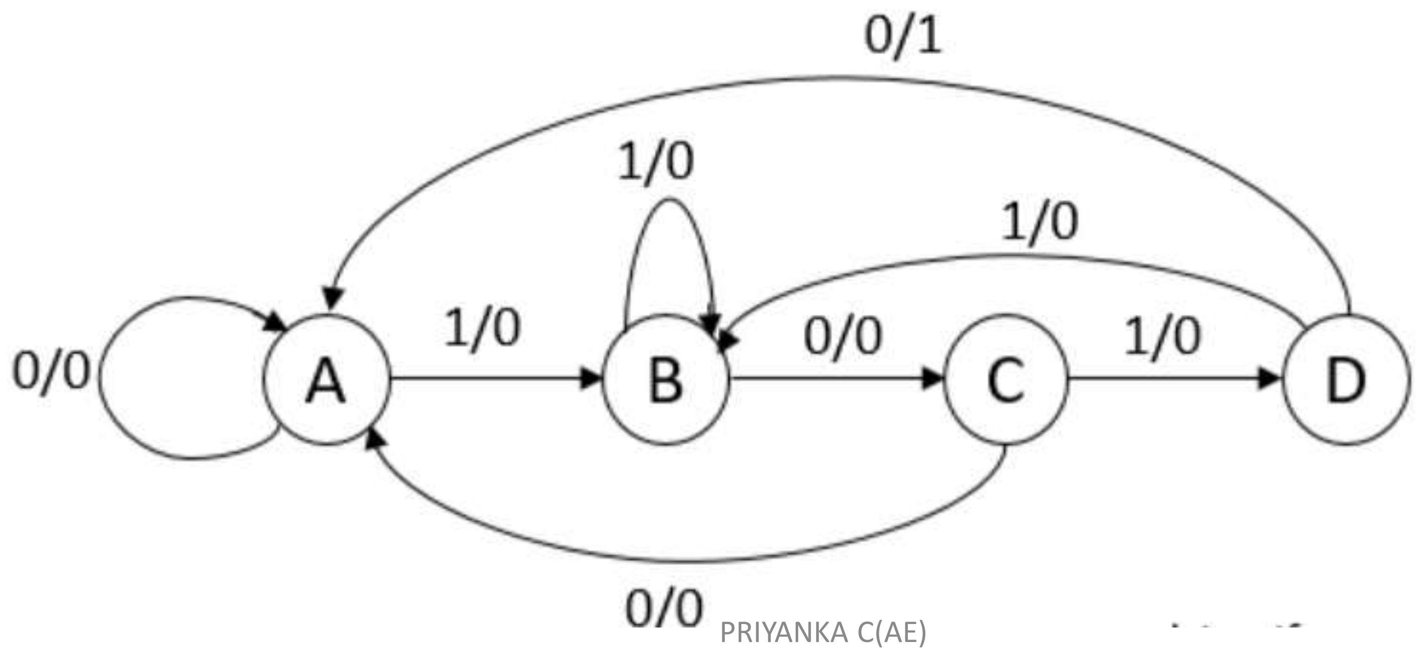


Sequence Detector - 1010

Mealy Machine

Non-Overlapping Sequence Detector

----- once the sequence is started, the detector resets and starts from beginning to find new sequence

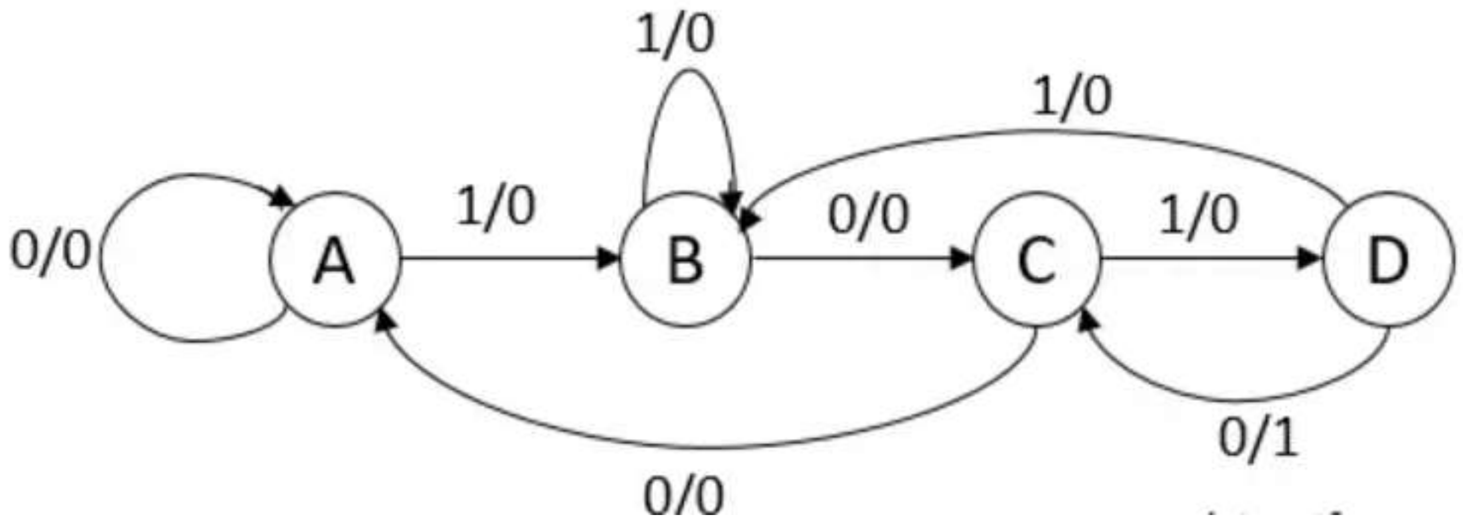


Sequence Detector – 1010

Mealy Machine

Overlapping Sequence Detector

----- Last few bits of the sequence can be the start of another sequence

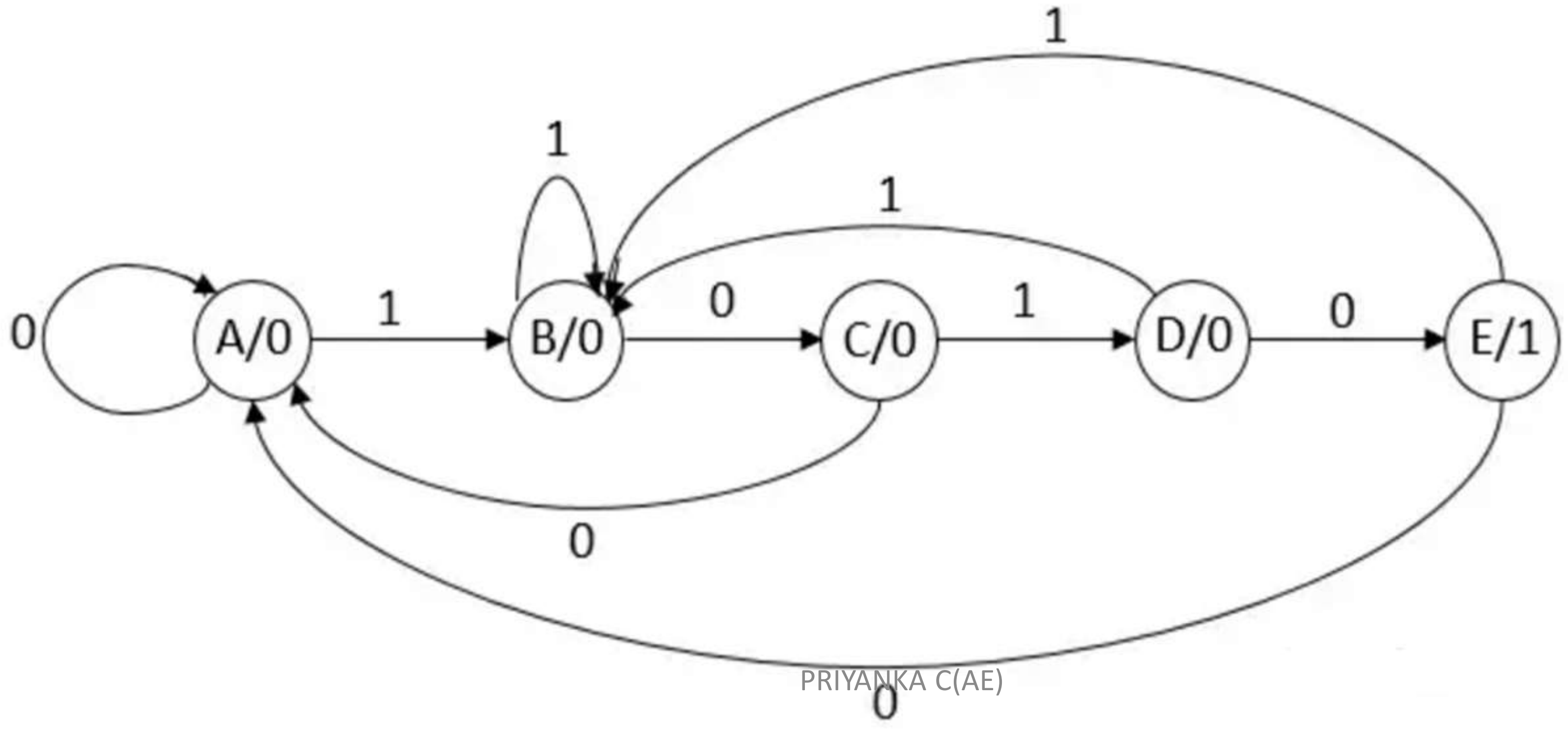


Sequence Detector – 1010

Moore Machine

Non-Overlapping Sequence Detector

----- once the sequence is started, the detector resets and starts from beginning to find new sequence

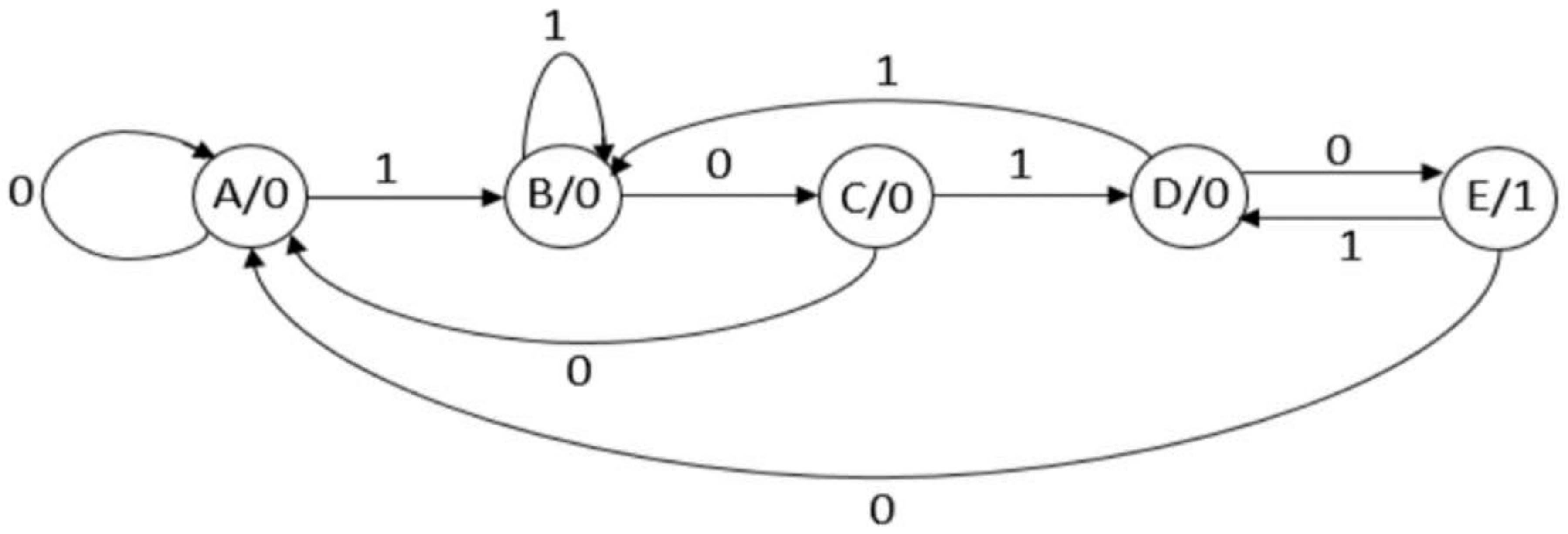


Sequence Detector – 1010

Moore Machine

Overlapping Sequence Detector

----- Last few bits of the sequence can be the start of another sequence



Assignment 1

1. Derive the state diagram for an FSM that has an input w and an output z. The machine has to generate z = 1 when the previous four values of w were 1001 or 1111; otherwise, z = 0. Overlapping input patterns are allowed.

   An example of the desired behavior is

   w: 010111100110011111

   z: 000000100100010011

   Write a Verilog RTL code and verify the same.

Assignment 2

2. A sequential circuit has two inputs, w1 and w2, and an output, z. Its function is to compare the input sequences on the two inputs. If w 1=w2 during any four consecutive clock cycles, the circuit produces z = 1 otherwise, z = 0.

    For example
    w1:0110111000110
    w2:1110101000111
    z: 0000100001110

Write a Verilog RTL code and verify the same.

Assignment 3

3. Design the FSM controller for the traffic lights at an intersection (North/ South (NS) vs. East/West (EW) with green and red lights only.

The rule:

(a) if no car detected, stay the same state,

(b) if cars are detected in the direction with red light (independent of whether cars are detected in the direction with green light), switch state.

Design the circuit using JK-FF.

Write a Verilog RTL code and verify the same.

Assignment 4

4. A serial adder receives two operands A = an-1,..., a,... a0 and B = bn-1,..., bi,... bo as two sequences of bits (i = 0, 1,..., n-1) and adds them one bit at a time to generate the sequence of bits s of the sum as the output.

Implement this serial adder as a finite state machine.

Design the circuit using D-FF.

Write a Verilog RTL code and verify the same.

RAMAIAH
SKILL ACADEMY

Thank You