Digital Logic Design

Hajar Falahati

Department of Computer Engineering
IRAN University of Science and Technology

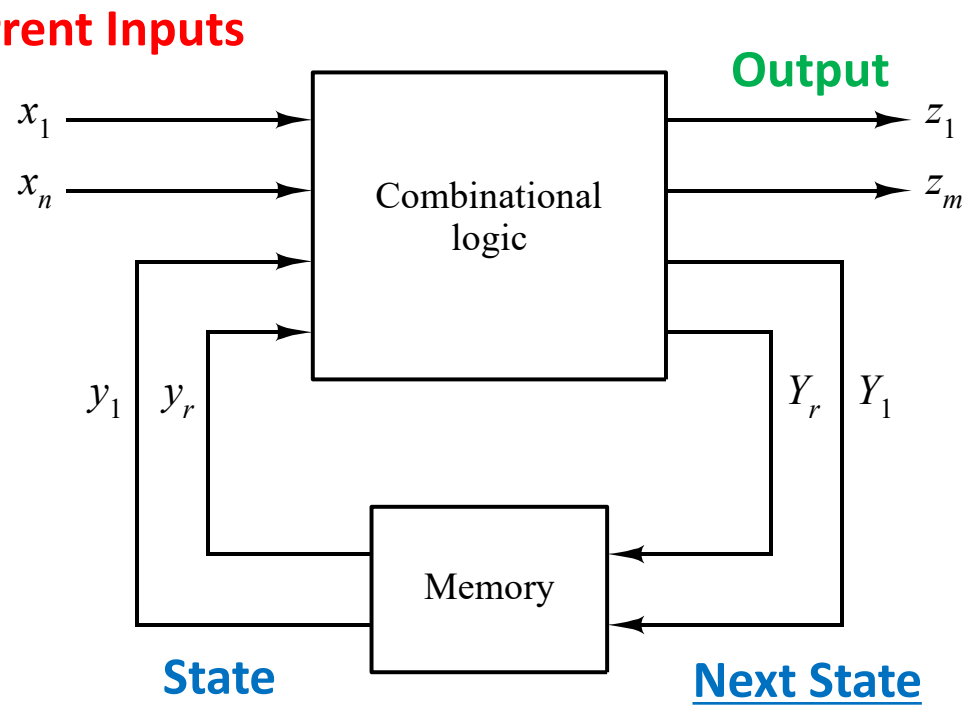
hfalahati@iust.ac.ir

Combinational Logics + Memory

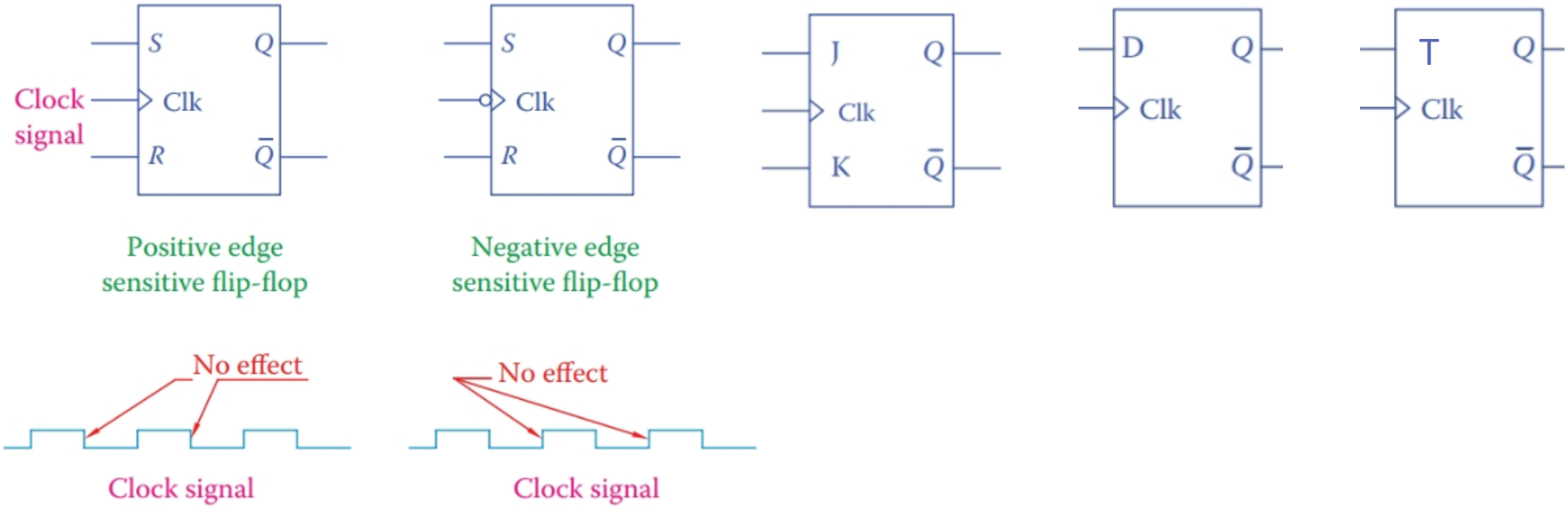
- **Output** of some logics **changes** by
  - **Sequence of inputs**
  - **Time of applying inputs**

- => They **have memory**
- They **contain feedback lines**

- State
  - Produced by previous inputs
  - Information stored in a memory



Flip Flops



R	S	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold the value
0	1	1	0	Set the value
1	0	0	1	Reset the value
1	1	Invalid		Invalid

J	K	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold the value
0	1	0	1	Reset the value
1	0	1	0	Set the value
1	1	$\bar{Q}$	Q	Toggle the value

D	Q	$\bar{Q}$
0	0	1
1	1	0

Delay FF

T	Q	$\bar{Q}$	
0	Q	$\bar{Q}$	Hold the value
1	$\bar{Q}$	Q	Toggle the value

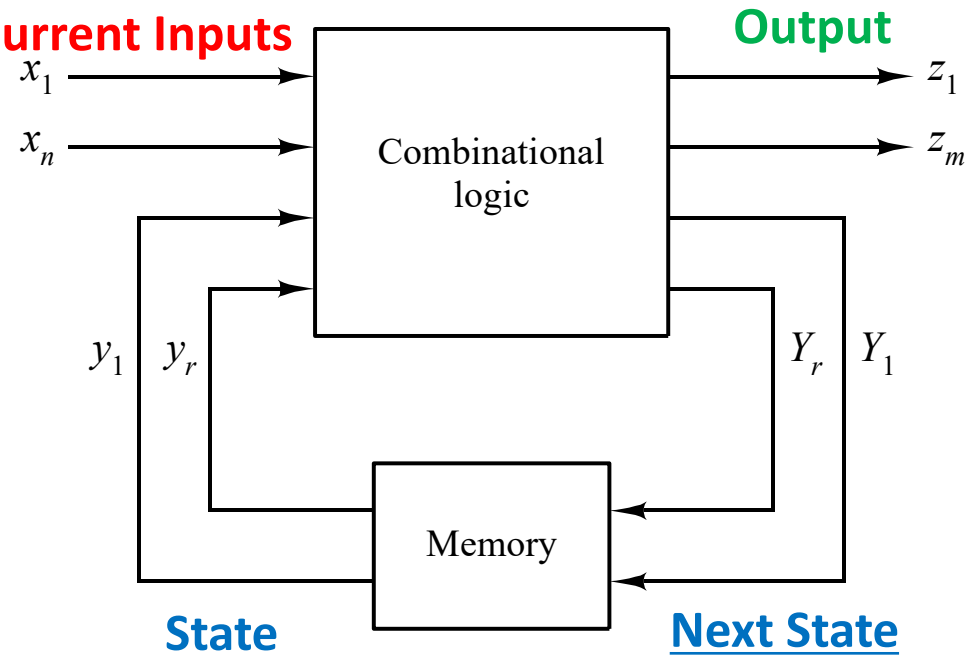
Outline

- Modeling of Synchronous Sequential Circuits
- Analysis of Synchronous Sequential Circuits
- Synthesis of Synchronous Sequential Circuits

Analysis

Analysis

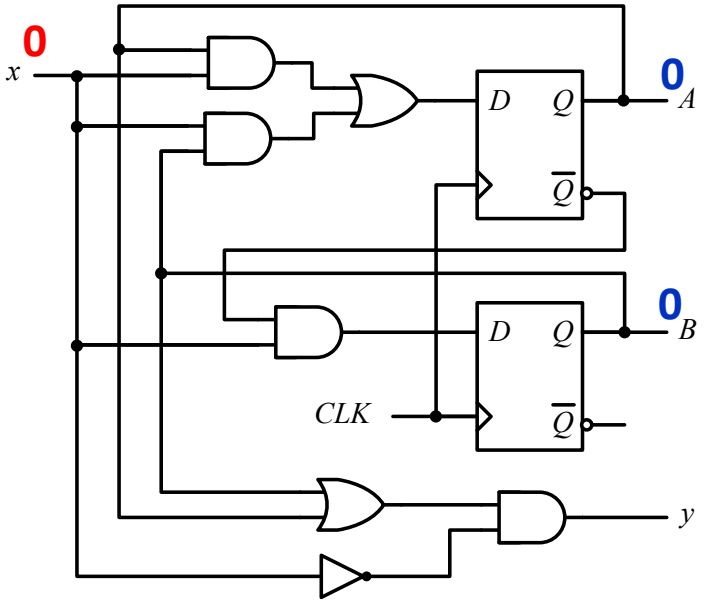
- Determining the **functional** relation between
  - **Outputs**
    - E.g., y(t)
  - **Internal states** (content of FFs)
    - E.g., A(t), B(t)
- Determining the **next state**
  - **Internal states**
    - E.g., A(t+1), B(t+1)



- **State transition**
  - Moving from one memory state to another one
  - Under the control of a set of inputs and at a time determined by an external clock
  - **Next value of states only determine at the clock transition**

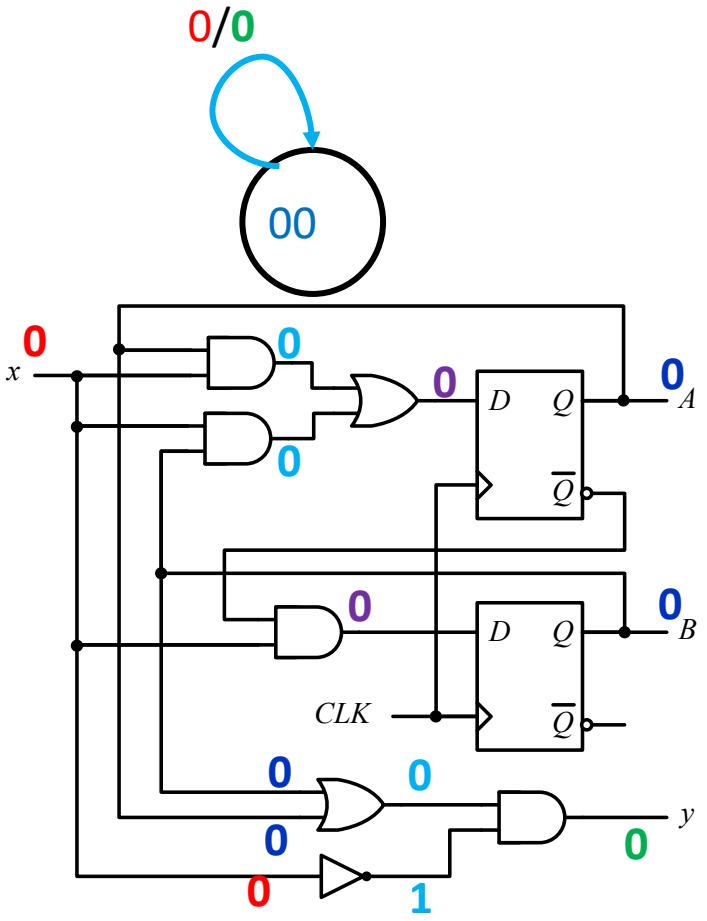
Analysis: Transition Table

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



Analysis: Transition Table

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			



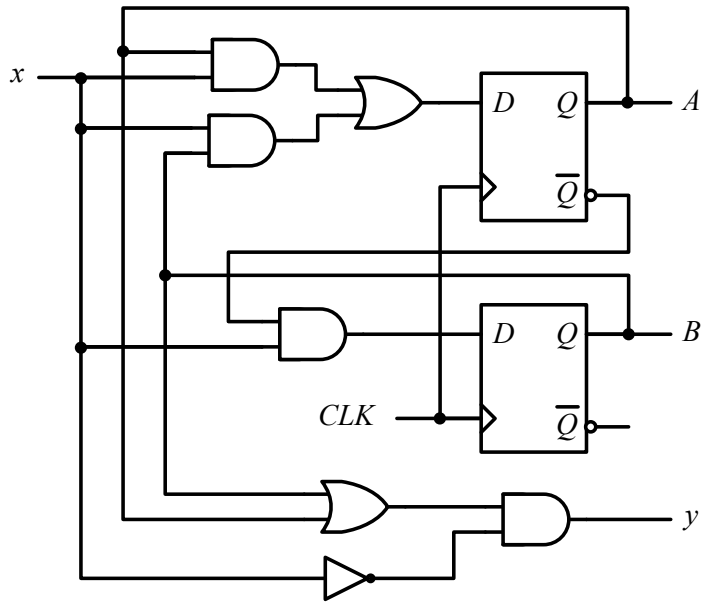
Analysis: Transition Table

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

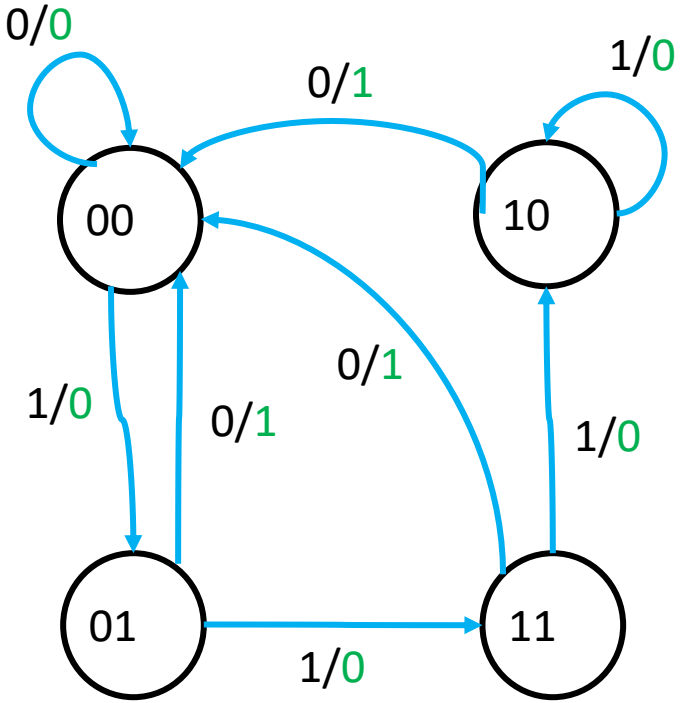
$$A(t+1) = D_A = A(t).x(t) + B(t).x(t)$$

$$B(t+1) = D_B = \overline{A(t)}.x(t)$$

$$y(t) = \overline{x(t)}.(A(t) + B(t))$$



Analysis: State Diagram

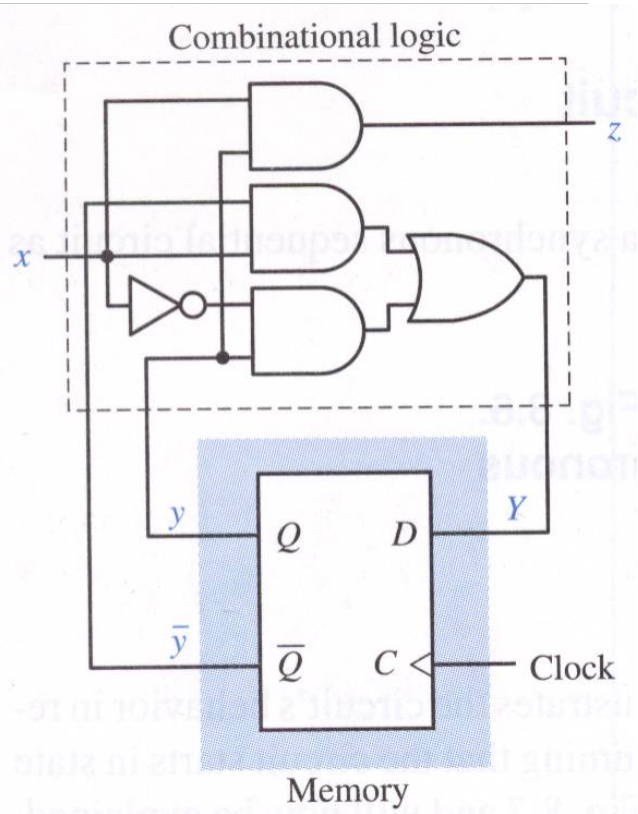


Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Analysis Sample1: Transition Table

$$Y(t+1) = D = \overline{Y(t)}.x(t) + Y(t).\overline{x(t)}$$

$$z(t) = Y(t).x(t)$$

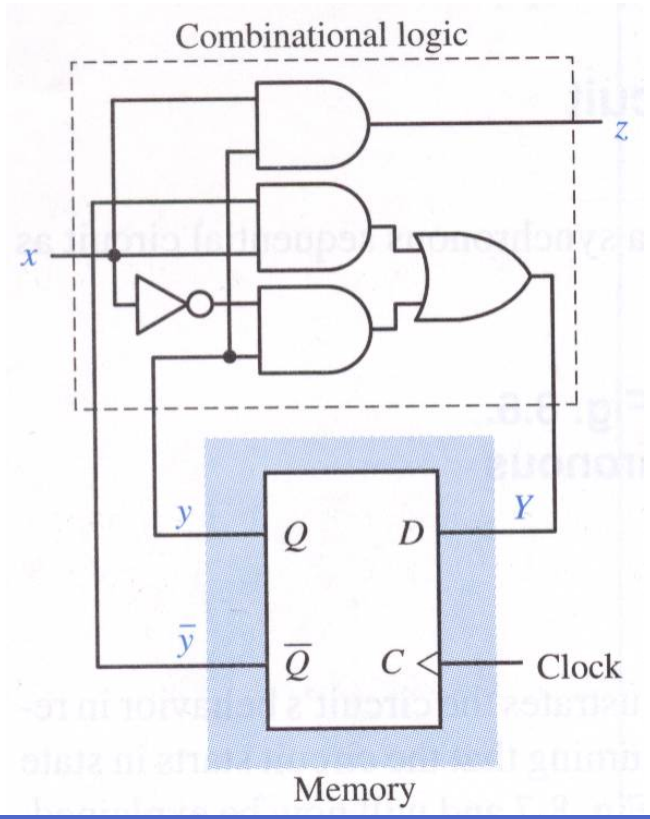


Analysis Sample1: Transition Table

$$Y(t+1) = D = \overline{Y(t)}.x(t) + Y(t).\overline{x(t)}$$

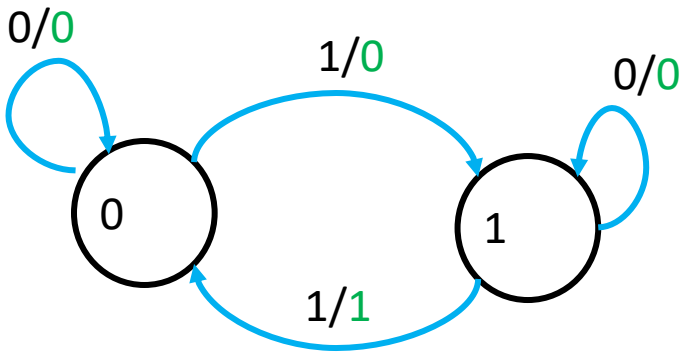
$$z(t) = Y(t).x(t)$$

Present State	Input	Next State	Output
Y	x	Y	z
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



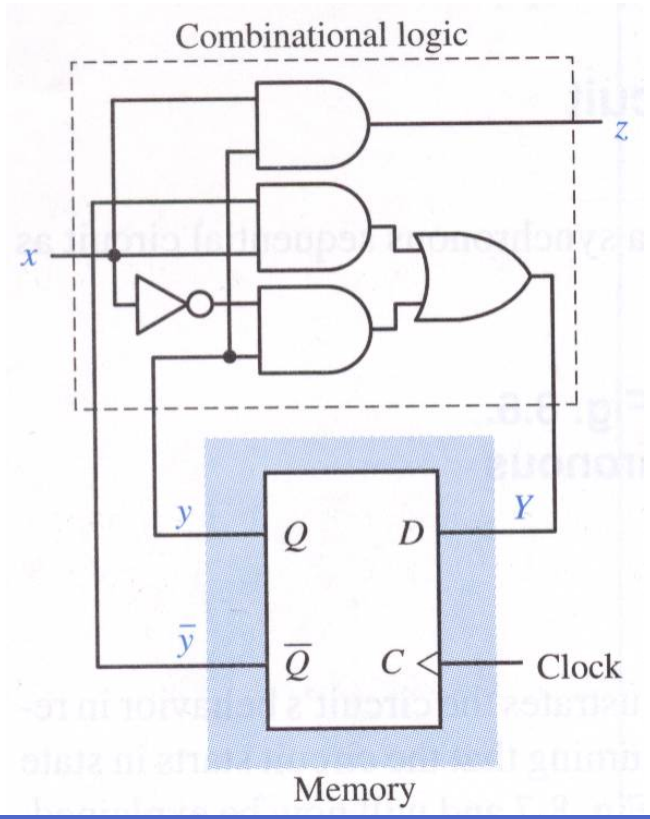
Analysis Sample1: State Diagram

Present State	Input	Next State	Output
Y	x	Y	z
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

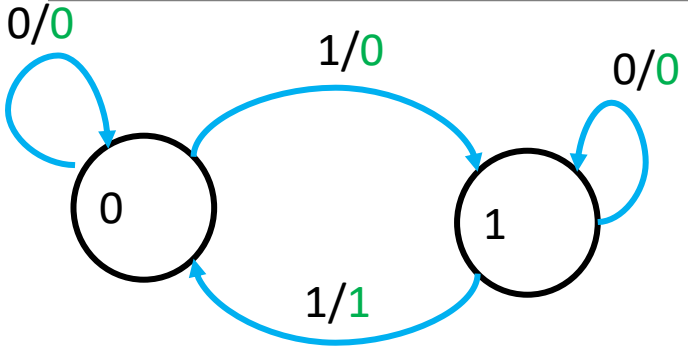


$$Y(t+1) = D = \overline{Y(t)}.x(t) + Y(t).\overline{x(t)}$$

$$z(t) = Y(t).x(t)$$

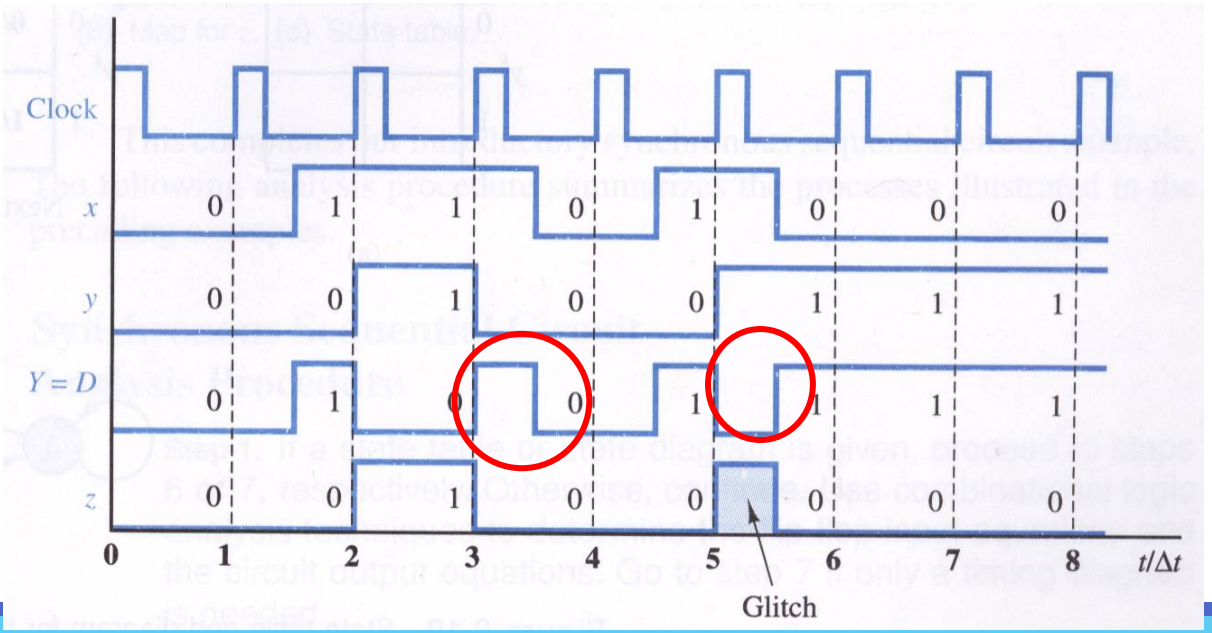


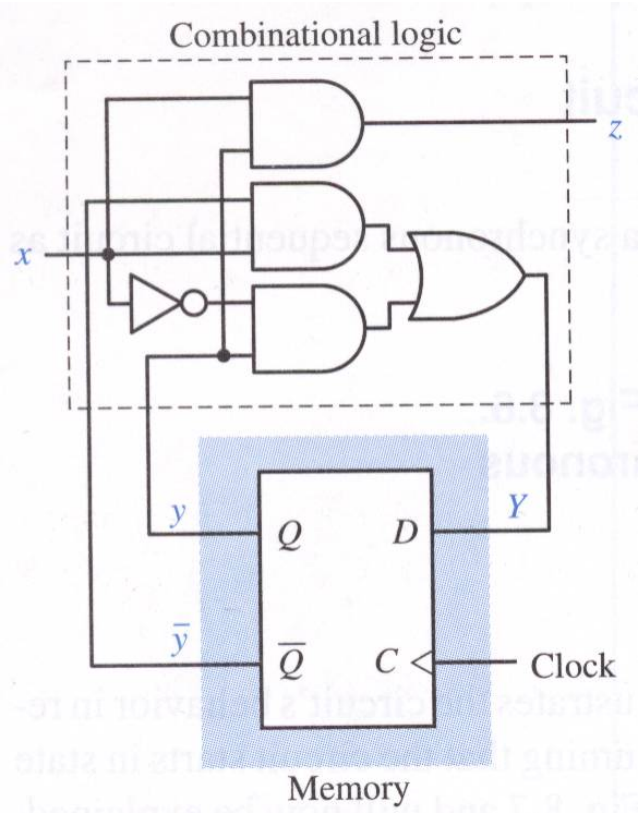
Analysis Sample1: Timing Diagram



$$Y(t+1) = D = \overline{Y(t)}.x(t) + Y(t).\overline{x(t)}$$

$$z(t) = Y(t).x(t)$$

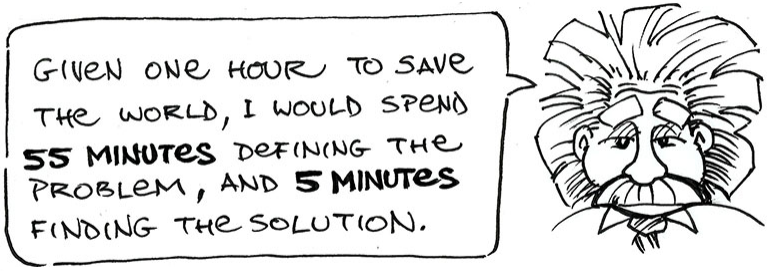


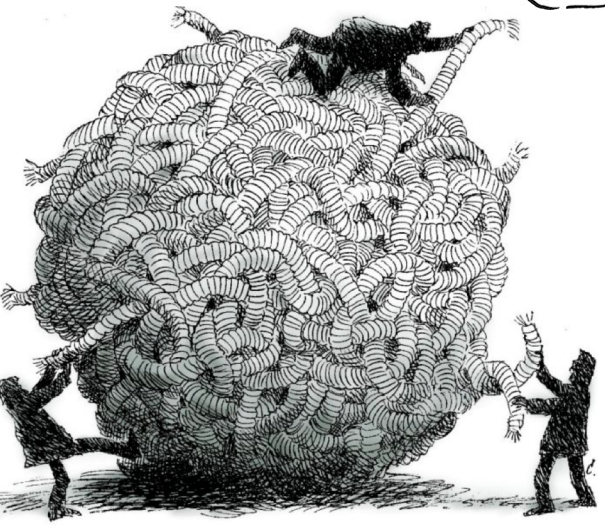


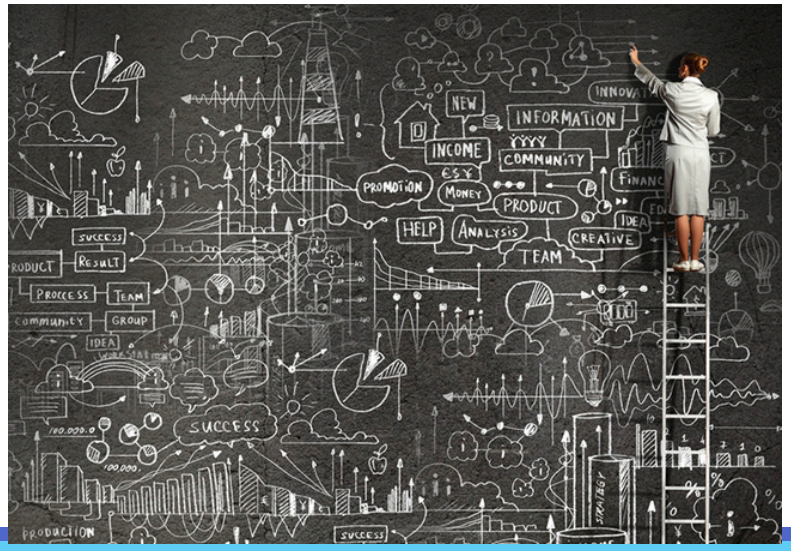
Models

How to Implement Real Designs?

- Identify the problem
- Model the problem

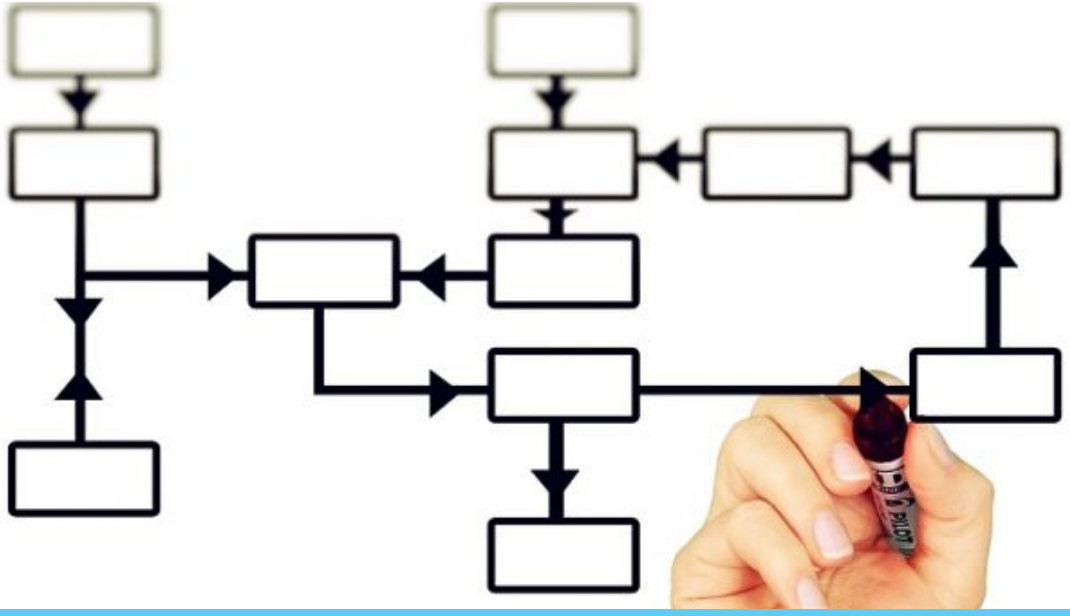


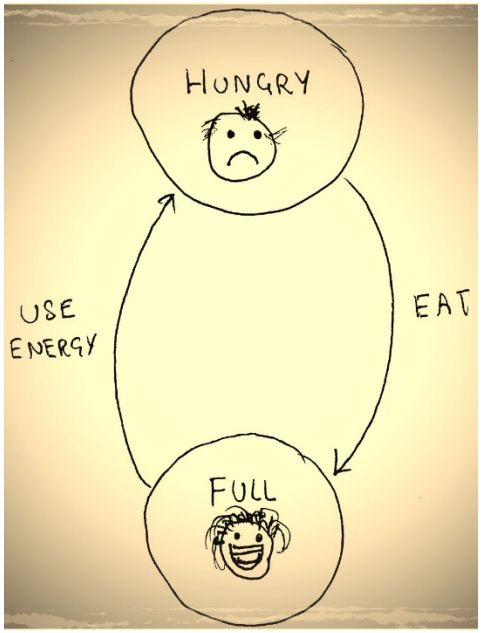




Modeling Types

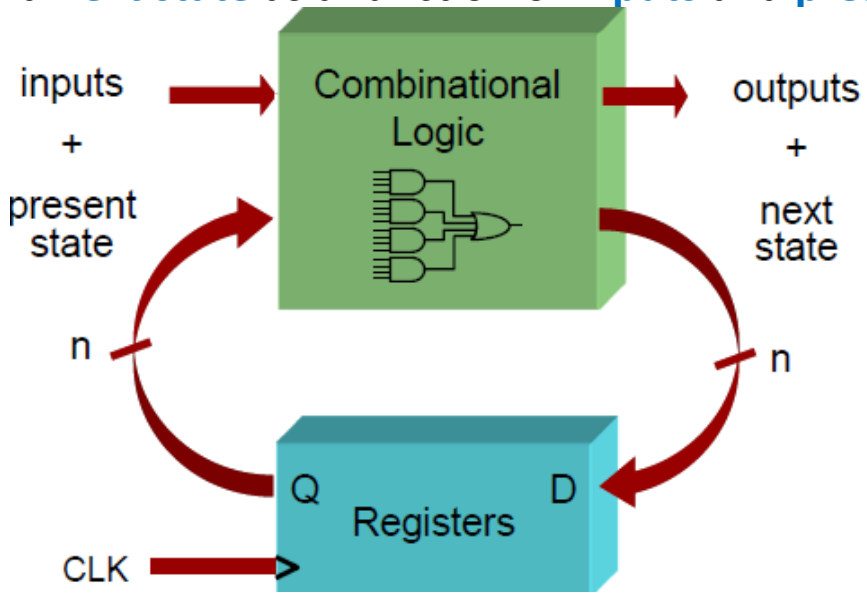
- Finite State Machine
  - FSM
- Algorithmic State Machine
  - ASM





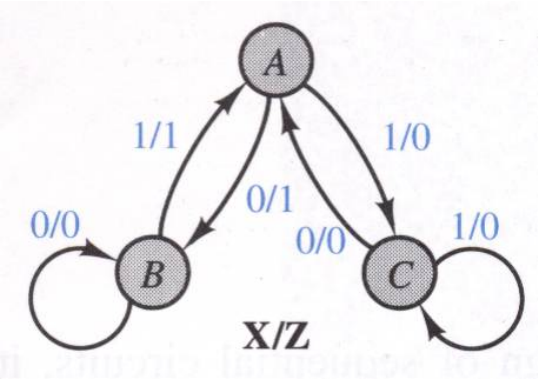
Finite State Machine (FSM)

- A system that visits a **finite** number of **logically distinct states**
- A useful **abstraction** for **sequential circuits** with **centralized states of operation**
- At each **clock edge**, combinational logic computes
  - **Outputs** and **next state** as a function of **inputs** and **present state**



FSM Representation

- State diagram & state table
  - Simple digital systems
  - A few number of inputs and outputs

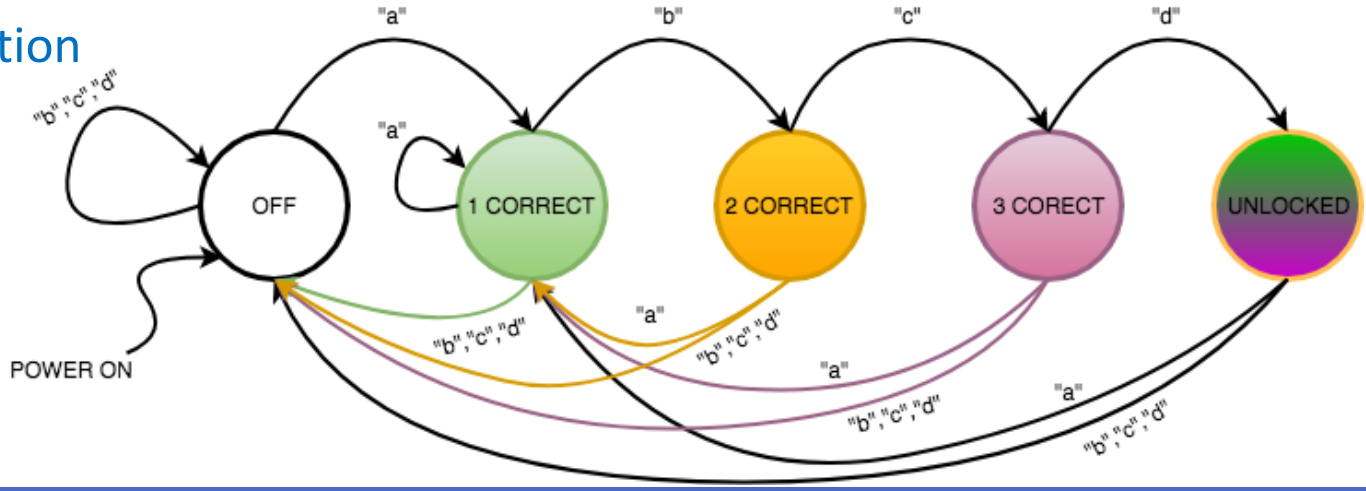


Present state	Input x = 0	Input x = 1
A	B/1	C/0
B	B/0	A/1
C	A/0	C/0

Next state/output

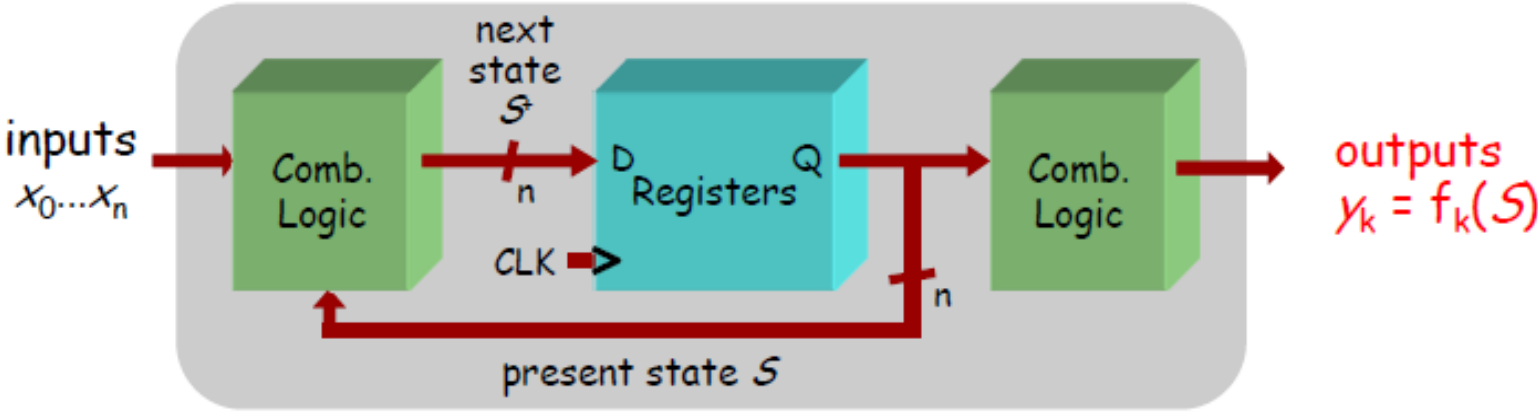
State Diagram

- A set of states
  - Nodes in a graph
- A set of inputs and outputs
  - Edges in a graph
- A set of state transition function
  - Edges in a graph
- An output function

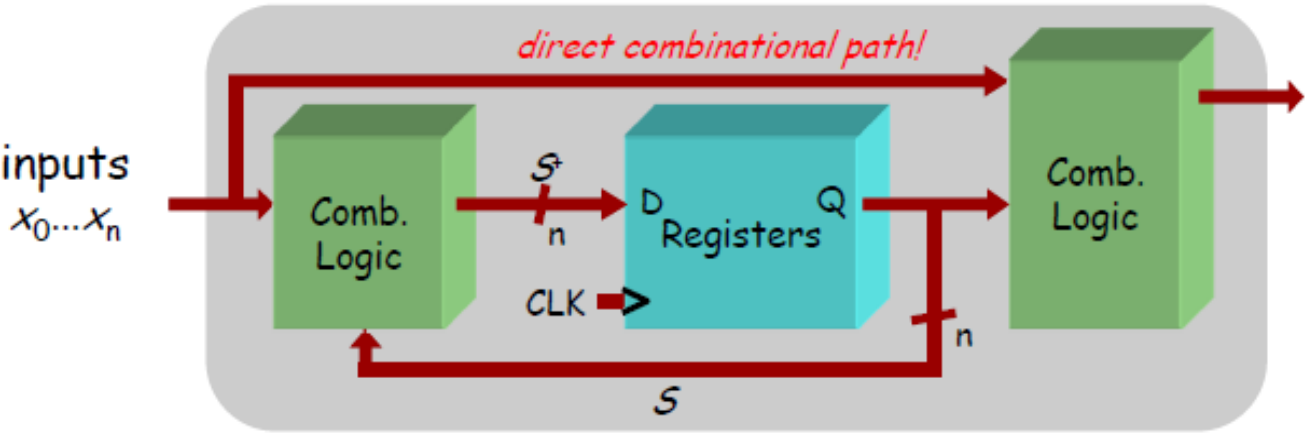


FSM Types

Moore FSM

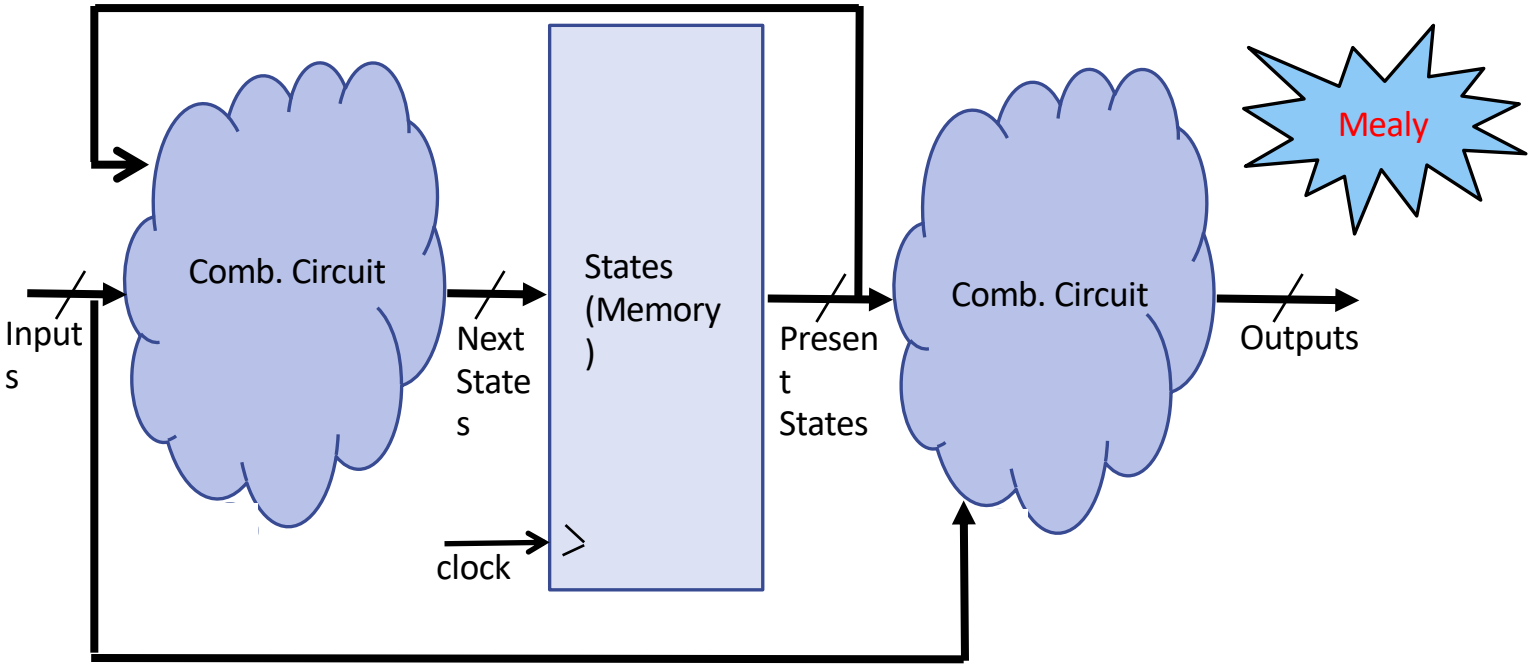


$S' = g\,(S, X_0 \ldots X_n)$

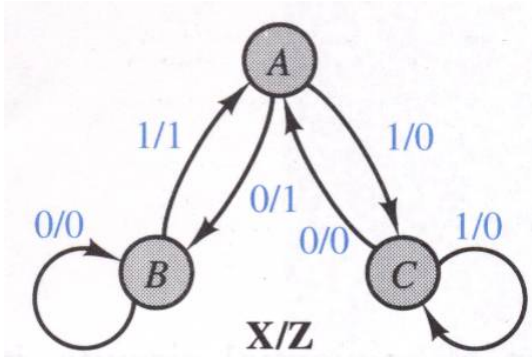


Mealy FSM

Moore and Mealy Machines



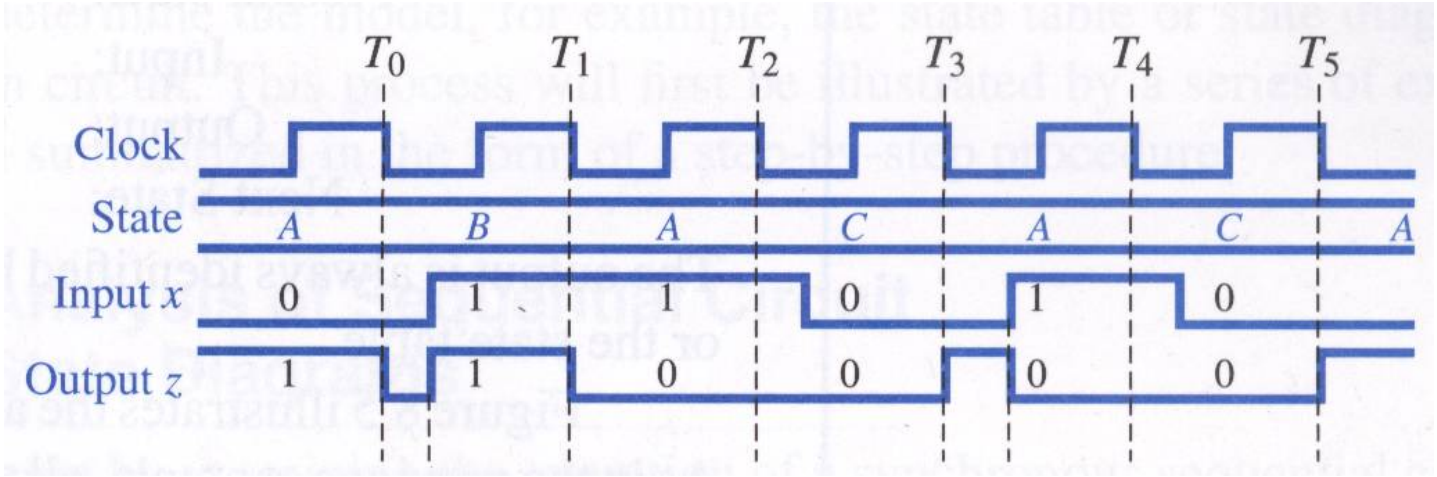
Mealy FSM: State Diagram & State Table



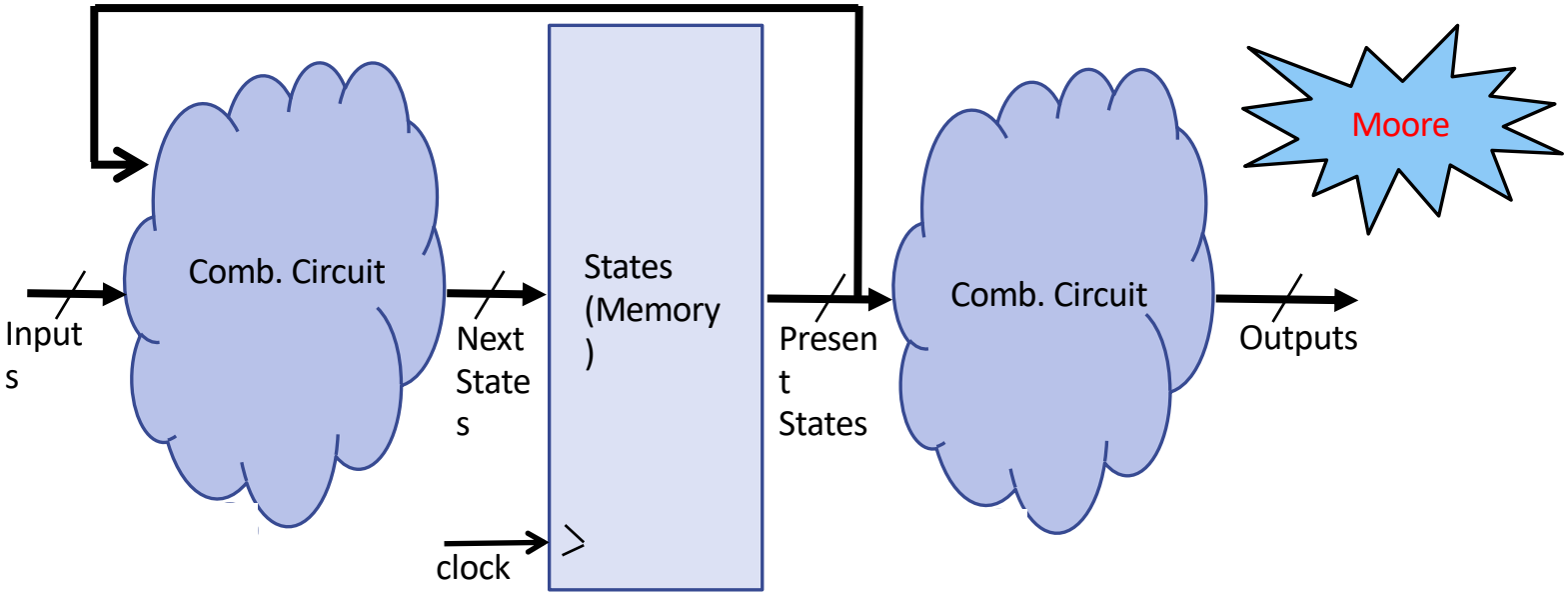
Present state	Input x = 0	Input x = 1
A	B/1	C/0
B	B/0	A/1
C	A/0	C/0

Next state/output

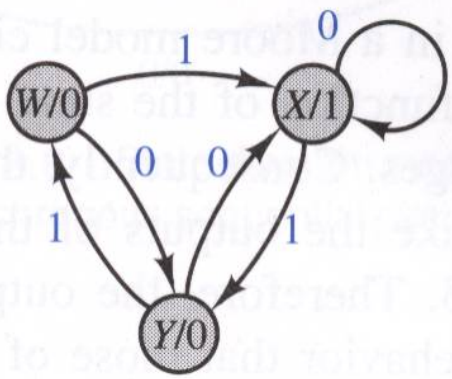
Time:	0	1	2	3	4	5
Present state:	A	B	A	C	A	C
Input:	0	1	1	0	1	0
Output:	1	1	0	0	0	0
Next state:	B	A	C	A	C	A



Moore and Mealy Machines

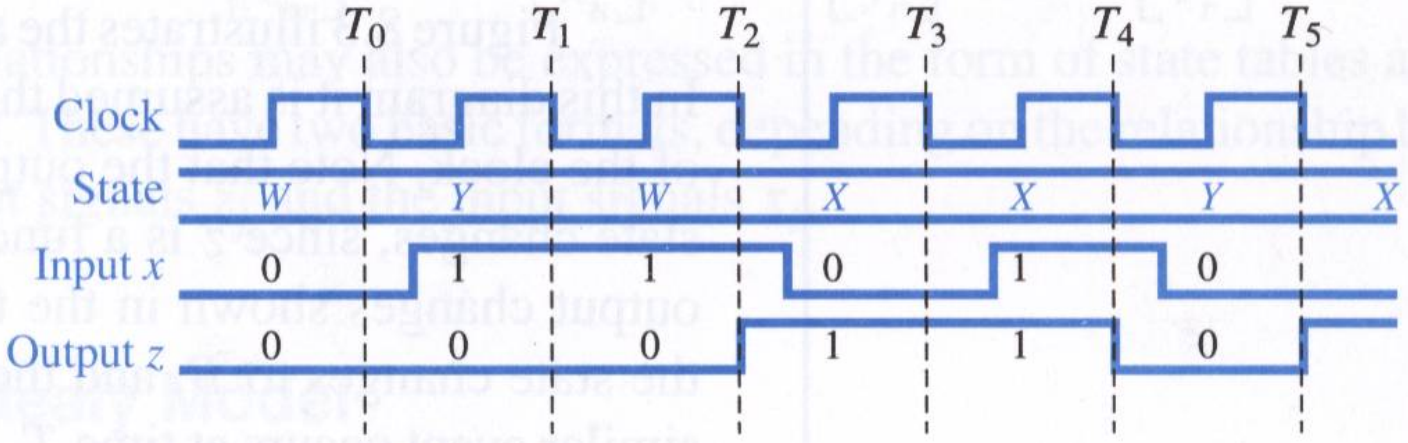


Moore FSM: State Diagram & State Table



Present state	Input x = 0	Input x = 1	Outputs
W	Y	X	0
X	X	Y	1
Y	X	W	0

Time:	0	1	2	3	4	5
Present state:	W	Y	W	X	X	Y
Input:	0	1	1	0	1	0
Output:	0	0	0	1	1	0
Next State:	Y	W	X	X	Y	X

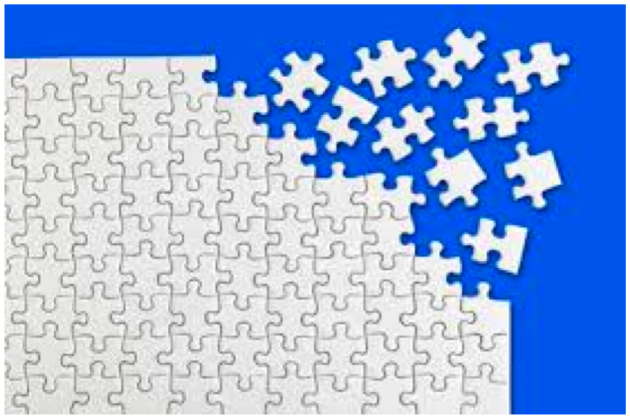


Vending Machine

- The computer department need a new soda machine
- We decided to ask computer students to design a controller for the new vending machine

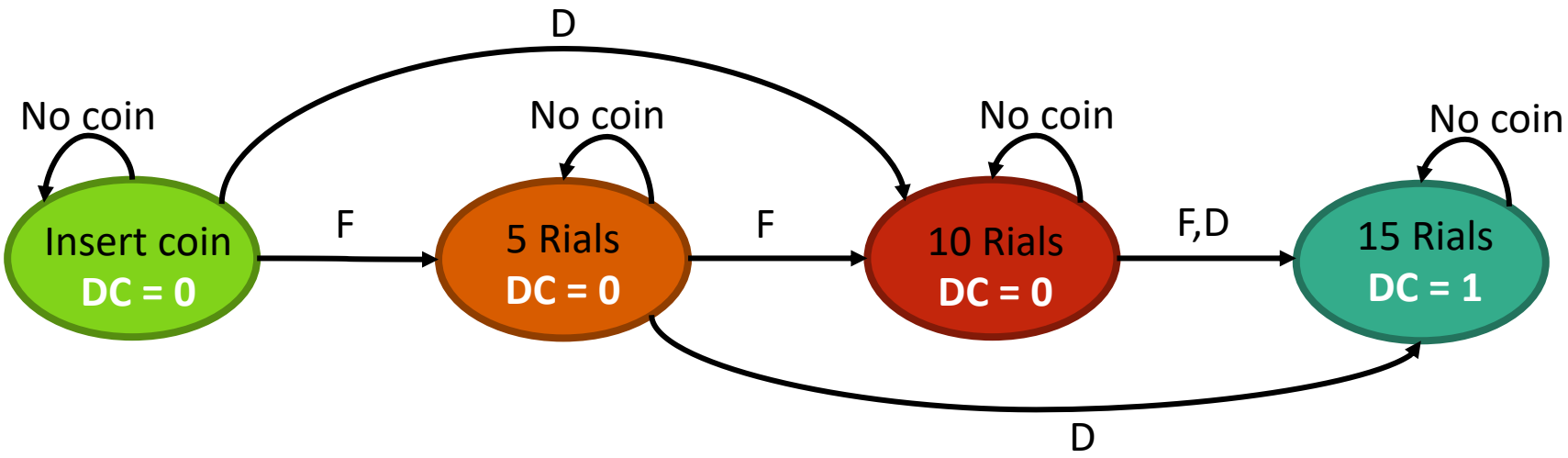
Vending Machine: Characteristics

- Characteristics
  - All selections cost **15** Rial
  - Machine does not return changes!
- Input
  - **D**: dim inserted (10 Rial!)
  - **F**: five inserted (5 Rial!)
- Output
  - **DC**: dispense can



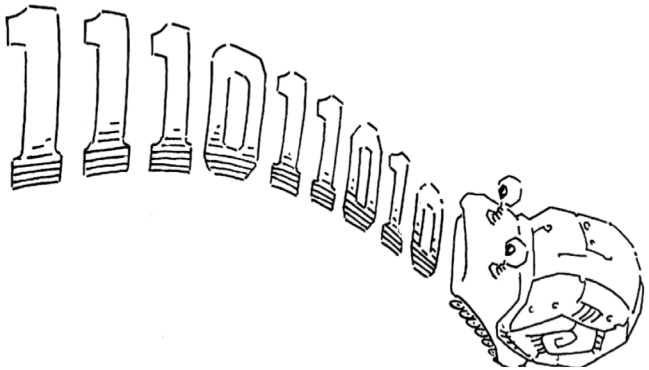
Vending Machine: FSM

- **Insert coin**
  - Default state
  - No money has been inserted
- **5 Rials**
  - A 5 Rial coin has been inserted
- **10 Rials**
  - A 10 Rial coin has been inserted
- **15 Rials**
  - Total money has been reached to 15 Rial
  - Done!



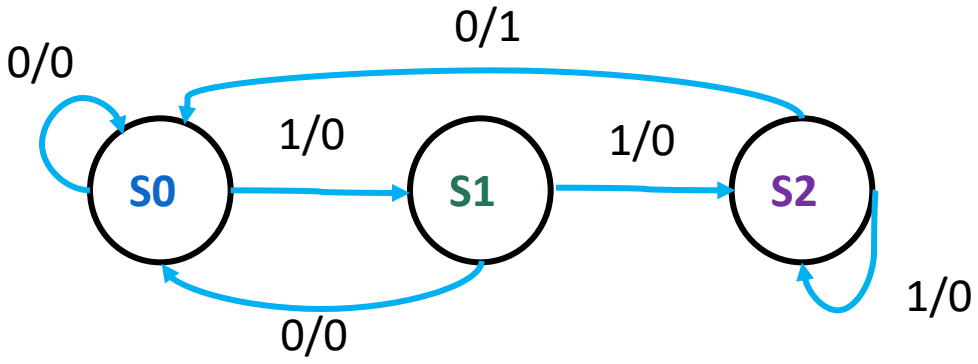
Pattern Recognition

- Recognize a specific bit pattern (*110*) in a bitstream
  - Mealy
  - Moore



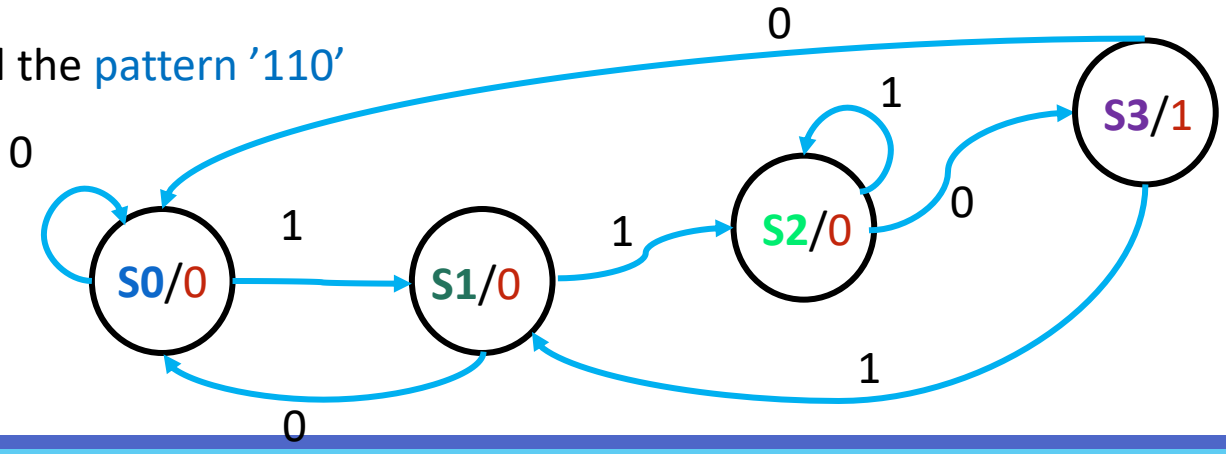
Pattern Recognition (110): Mealy

- State **S0**
  - We have not recognized any useful pattern
- State **S1**
  - We have recognized the pattern ’1’
- State **S2**:
  - We have recognized the pattern ’11’
  - **Output**: recognizing an input bit ’0’ in state S2



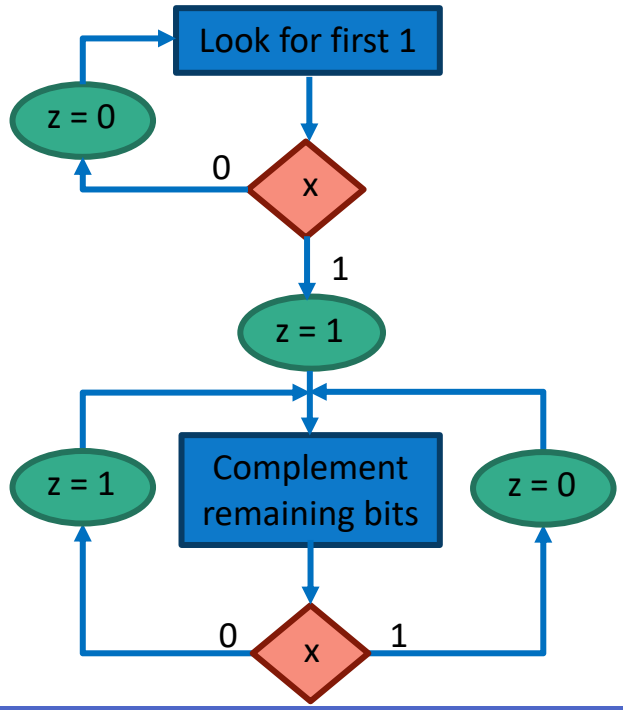
Pattern Recognition (110): Moore

- State **S0**
  - We have not recognized any useful pattern
- State **S1**
  - We have recognized the pattern ’1’
- State **S2**:
  - We have recognized the pattern ’11’
- State **S3**:
  - We have recognized the pattern ’110’
  - Output becomes 1



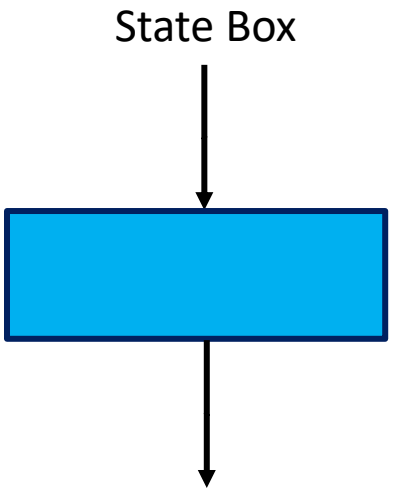
Algorithmic State Machine (ASM)

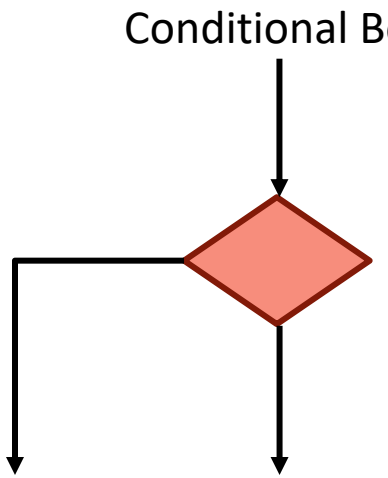
- A **systematic** way to **design complex** digital systems
  - Complex digital systems
  - Large number of inputs and outputs
- **Describes** the **sequence of events**
- **Describes timing relationship** between the states
- **Behavioral model**
- **Basic Idea**
  - Flowchart

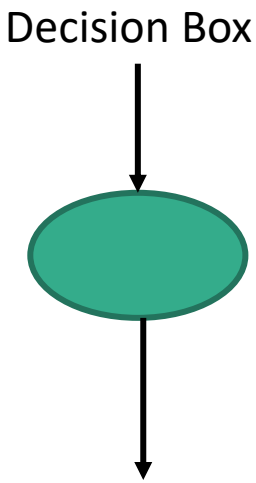


ASM chart Elements

- ASM chart elements
  - State box
  - Decision box
  - Conditional box

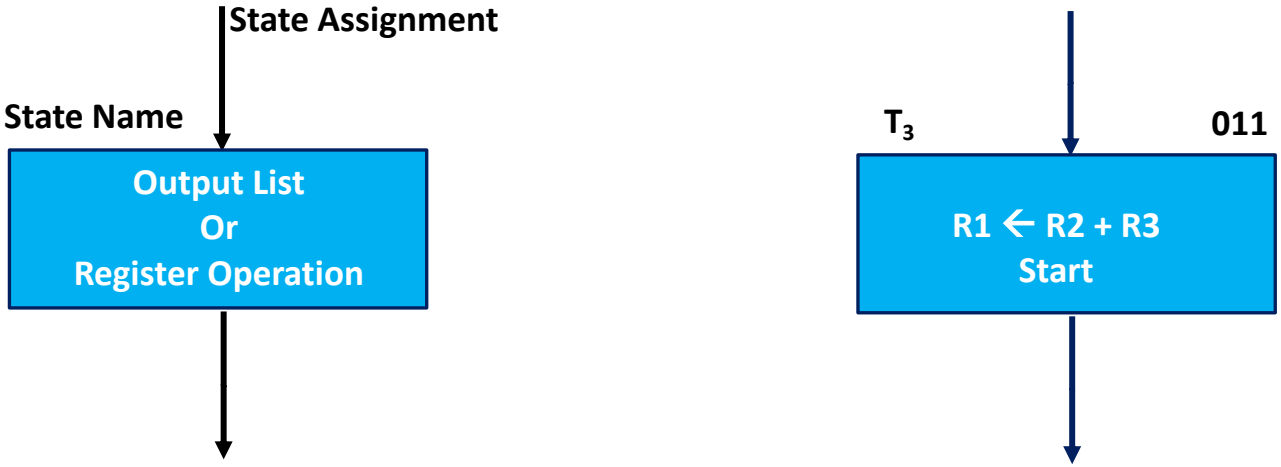






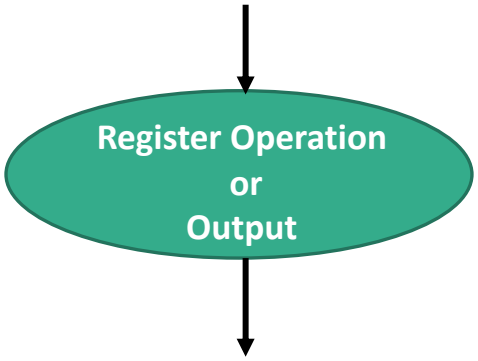
State Box

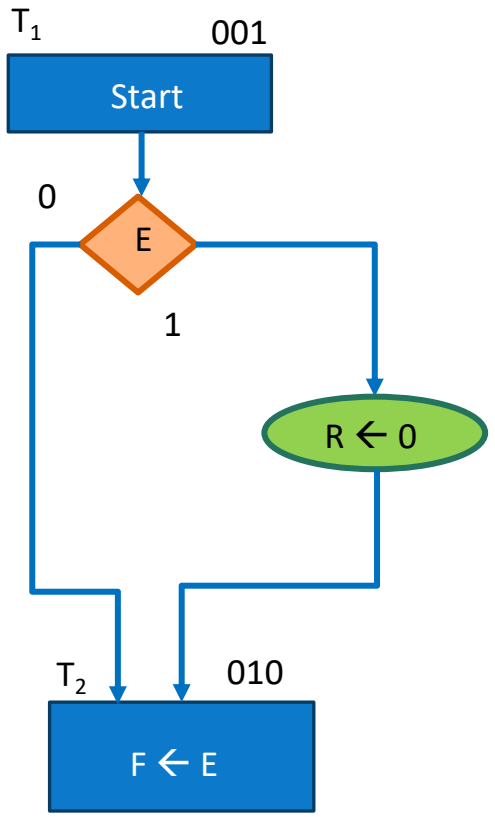
- Represents the state of the system
- Register Transfers
- Takes one cycle to be executed
  - R1 <- R2 + R3



Conditional Box

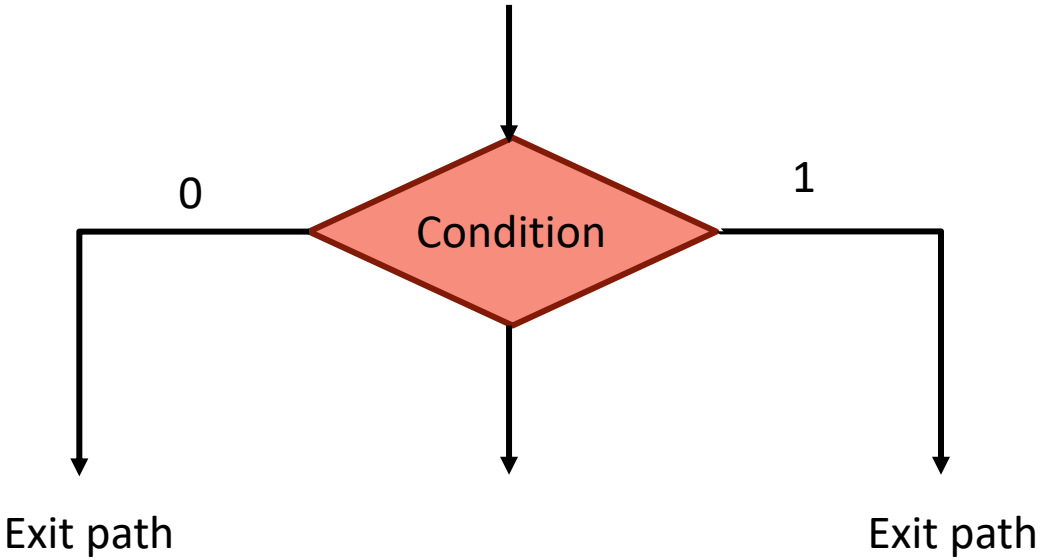
- Register Transfers
- Contain conditional output list
  - Depends on both the state of the system and the inputs
  - A.k.a., mealy output
  - A condition output must follow a decision box
- Takes one cycle to be executed
  - R1 <- R2 + R3





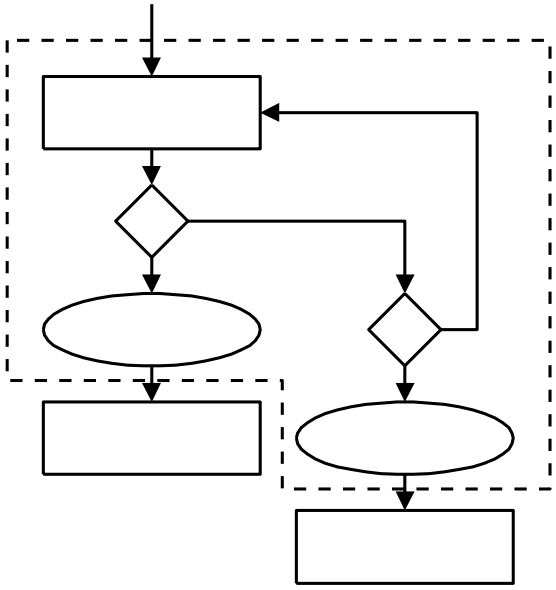
Decision Box

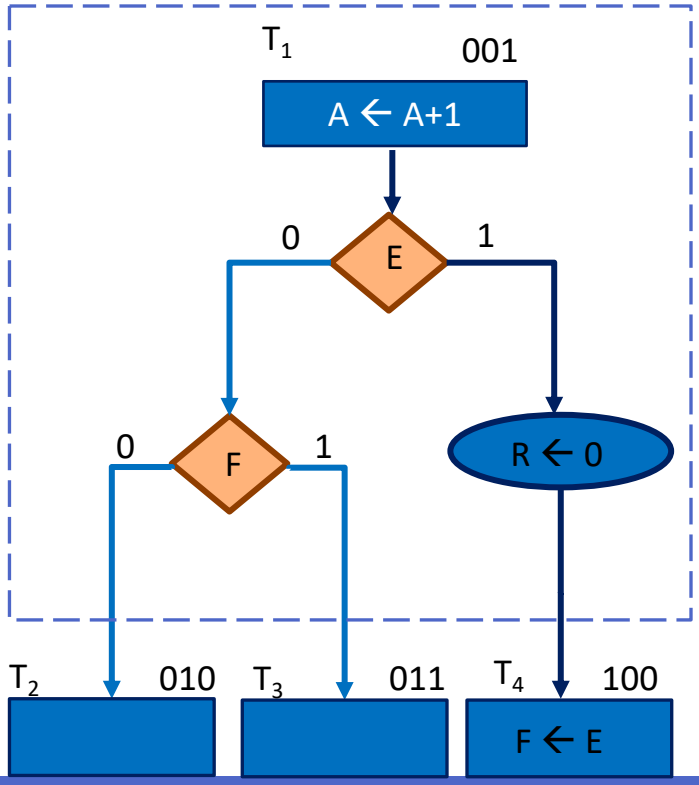
- Binary expressions
  - ~R1[7]



ASM Block

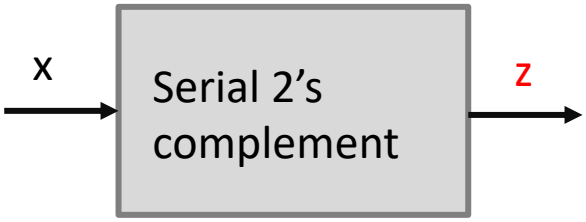
- Represents **what happens** in the system during one clock cycle
- Includes
  - Only **one state box**
  - All other boxes (decision and conditional boxes)
    - Connected to the exit path of the state box





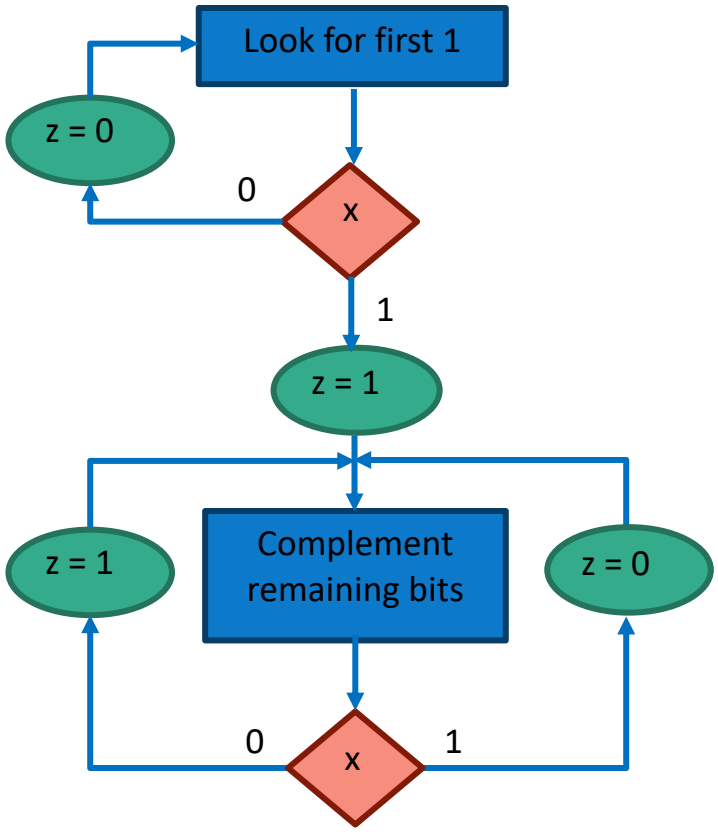
Sample Design 1

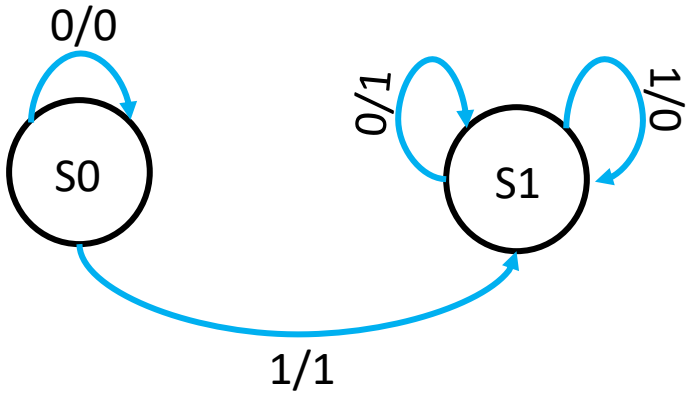
- Design a control unit for a serial 2's complement



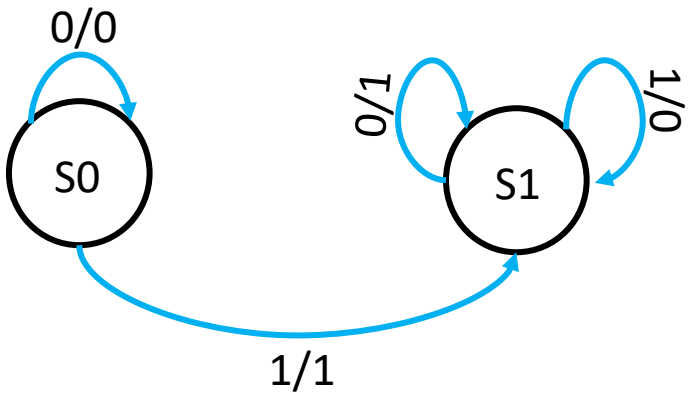
t	x	z
0	0	0
1	0	0
2	1	1
3	1	0

Sample Design 1: Modeling





Sample Design 1: Excitation Table

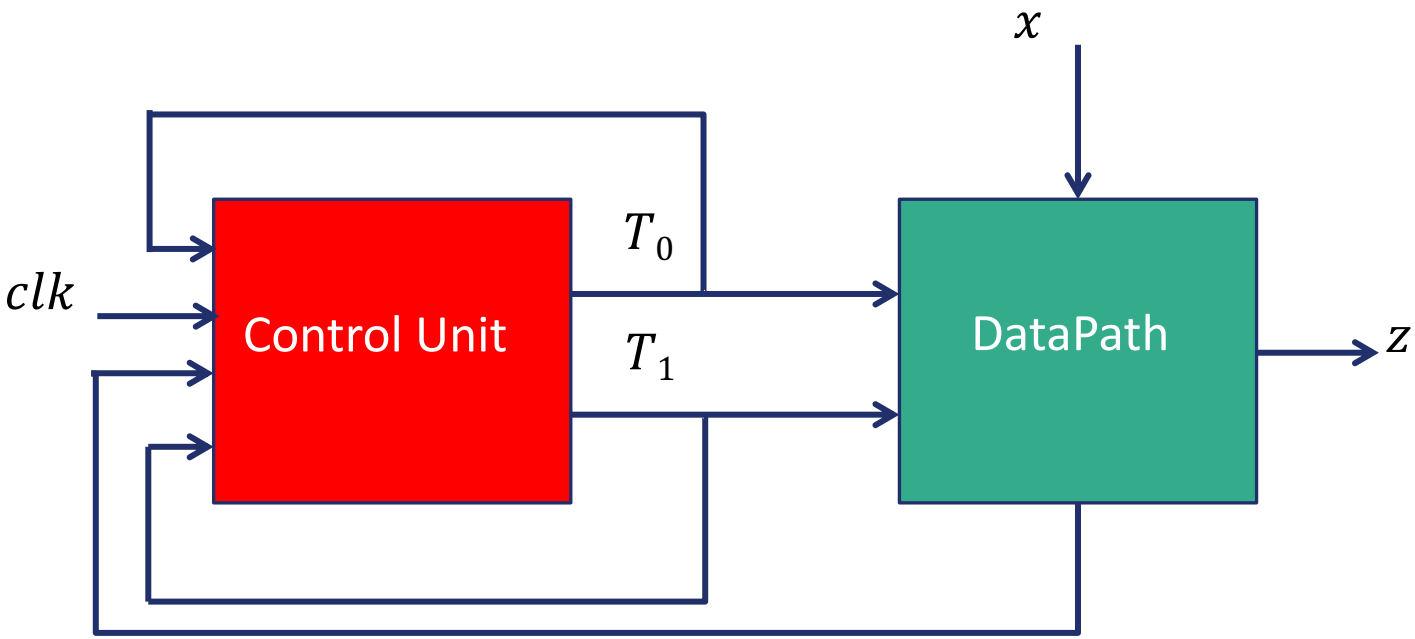


State	Input (x)	Next State	Output (z)	D
A(0)	0	A	0	0
A	1	B	1	1
B(1)	0	B	1	1
B	1	B	0	1

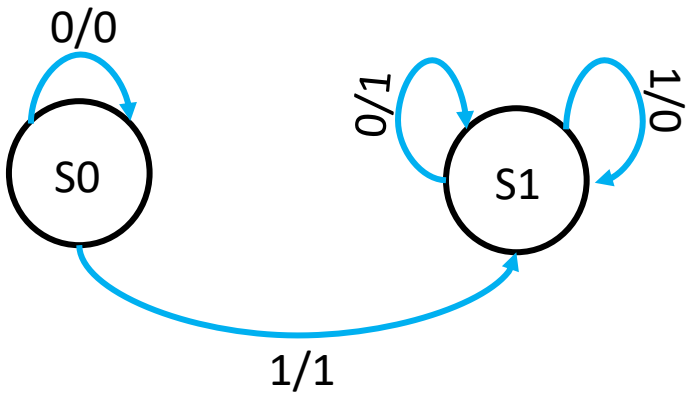
$z = Ax+Bx'$

$D=Ax+B$

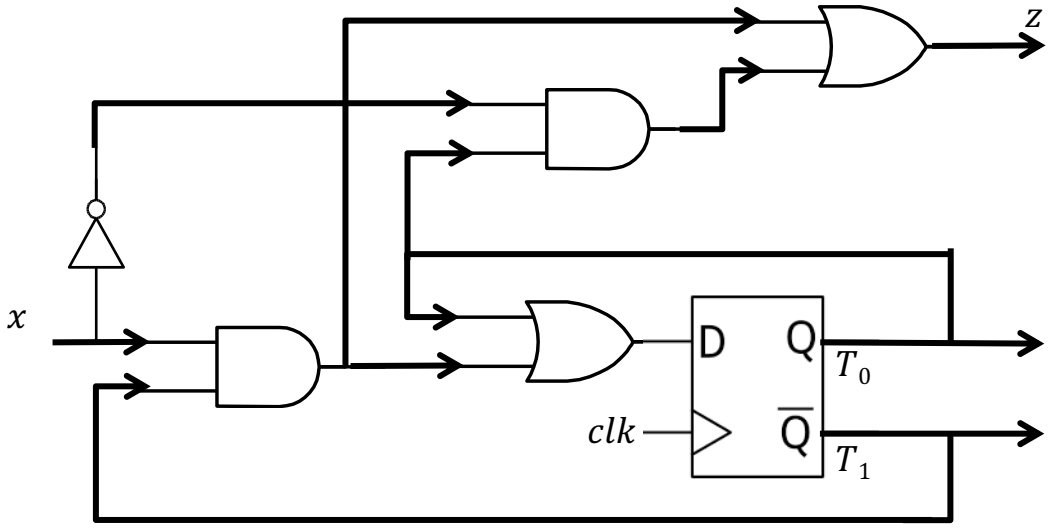
Sample Design 1: Block Diagram



Sample Design 1: Design



State	Input (x)	Next State	Output (z)	D
A(0)	0	A	0	0
A	1	B	1	1
B(1)	0	B	1	1
B	1	B	0	1



z = Ax+Bx'

D = Ax+B

Moore Vs. Mealy

- Mealy
  - Has richer description
  - Usually requires smaller number of states
  - Smaller circuit area
  - Computes outputs as soon as inputs change
  - Not synchronized with the clock
  - Responds one clock cycle sooner than equivalent Moore FSM
  - May have glitch

- Moore
  - Synchronize with clock
  - No glitch, while in Mealy, you may have glitch

Thank You