EE 421 / CS 425
Digital System Design
Fall 2025 Lecture 8

Shahid Masud

State Machines

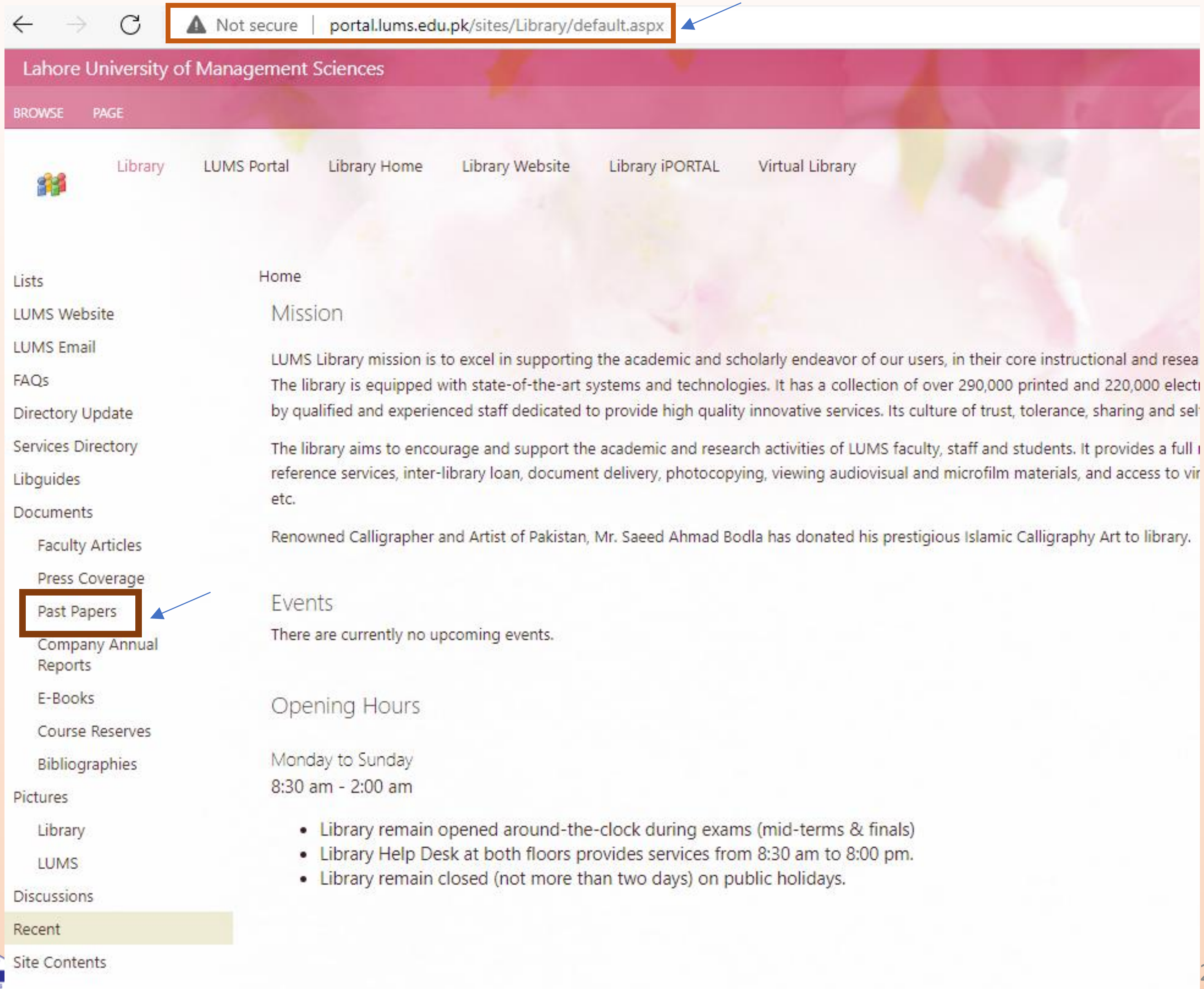
Vending Machine Example

Topics

ANNOUNCEMENT
QUIZ 2
Next Week

- State Diagrams
- Introduction to State Machines
- Moore State Machine and Mealy State Machine
- State Tables Description of State Machines
- Design Example of a State Machine (Vending Machine)
- Sequential State Machine Circuit Design
- One Hot Encoding and ASMD

Practice Questions from Past Papers



Check folders:

SSE → EE 421

and

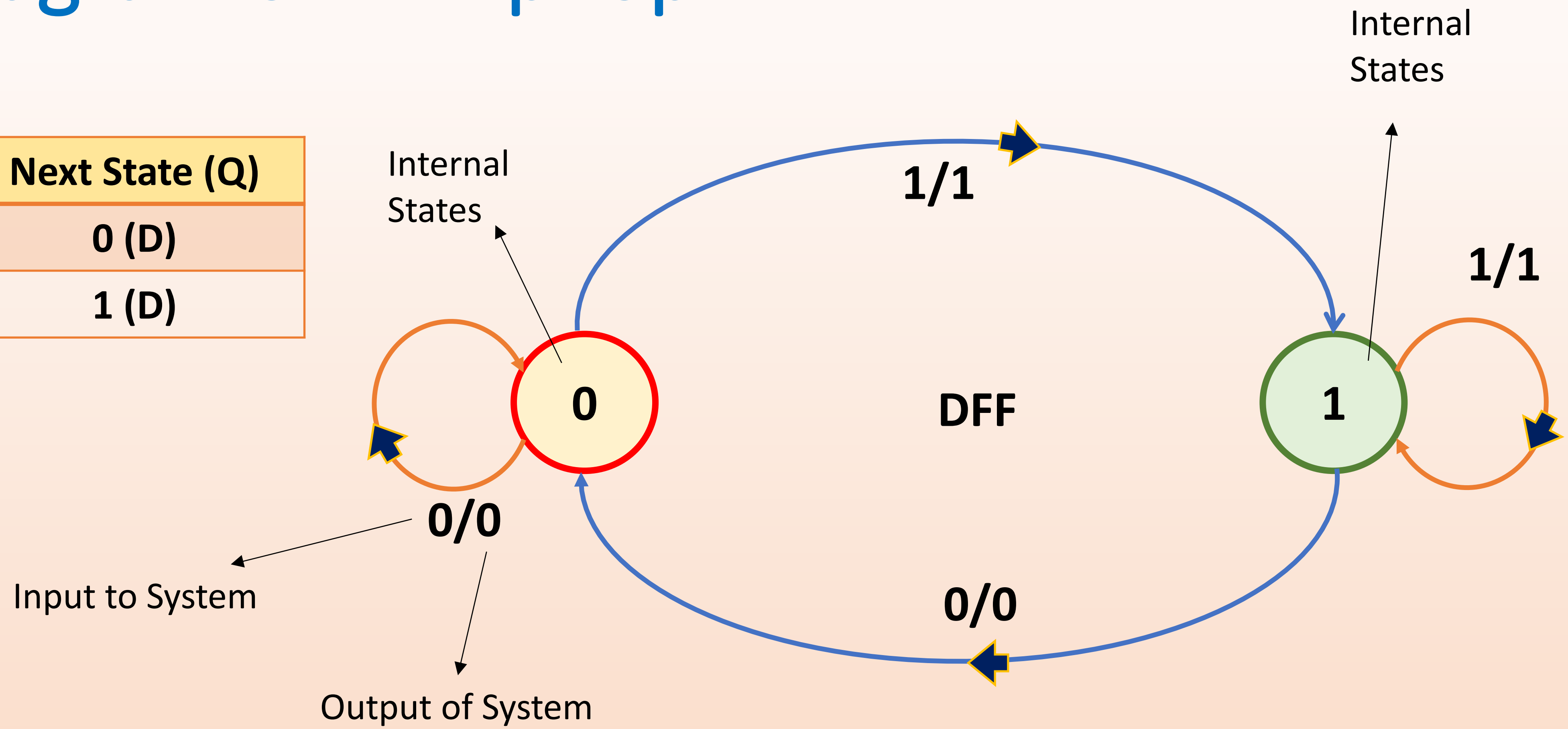
CS → CMPE 424

and

CS → CS 424

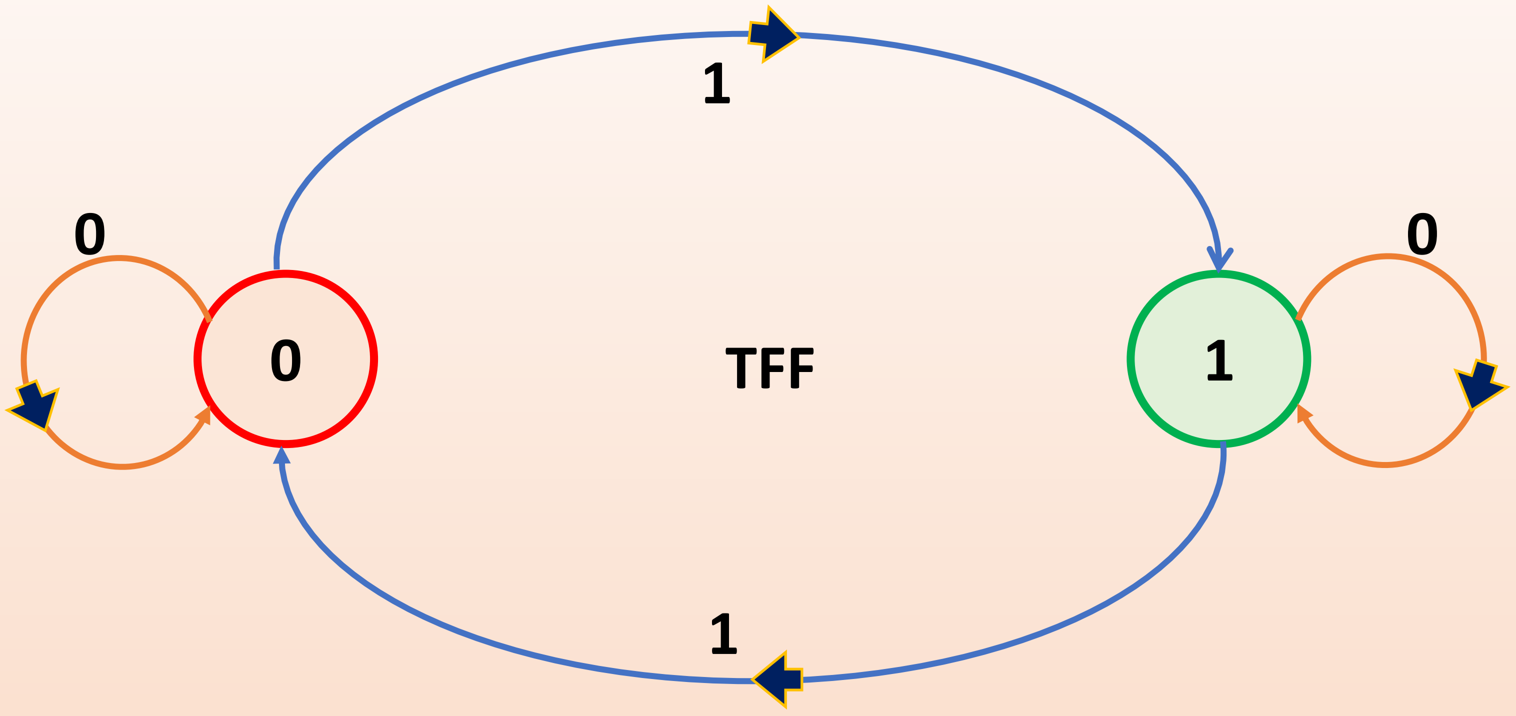
State Diagram of D Flipflop

Input (D)	Next State (Q)
0	0 (D)
1	1 (D)



State Diagram of T Flipflop

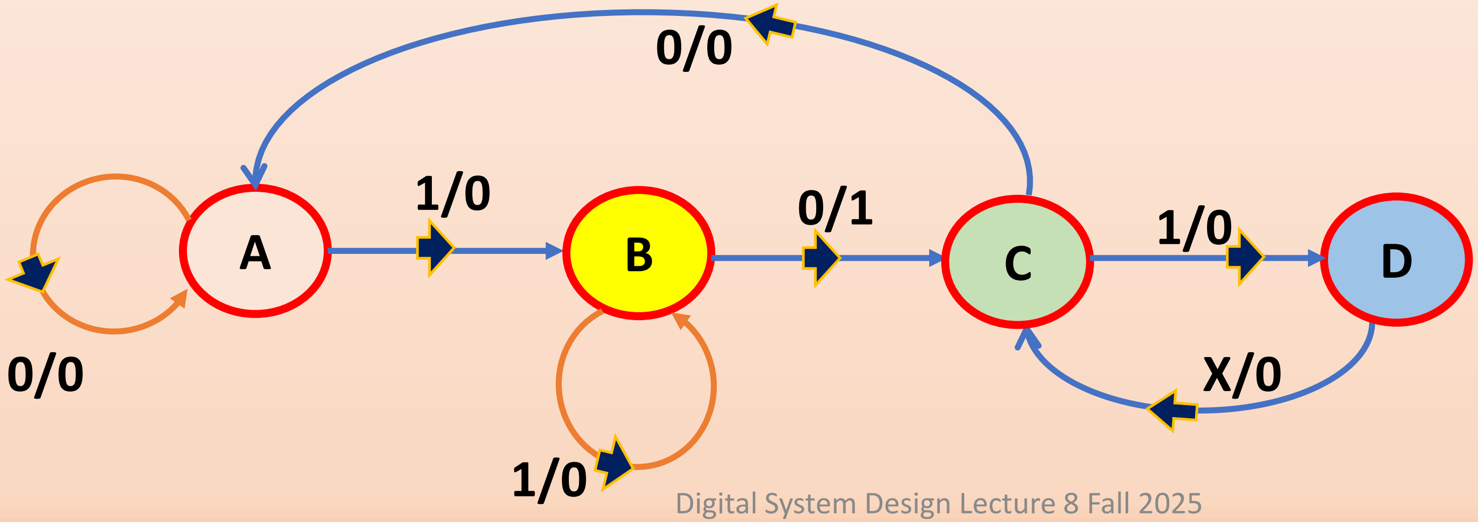
Input (T)	Next State (Q)
0	Q
1	Q'



Linking State Table with State Diagram

Complete State Description Including Inputs, Present State, Outputs and Next State

Q (t)	Q (t+1)		Z (Output)	
	Input X=0	Input X=1	Input X=0	Input X=1
A	A	B	0	0
B	C	B	1	0
C	A	D	0	0
D	C	C	0	0

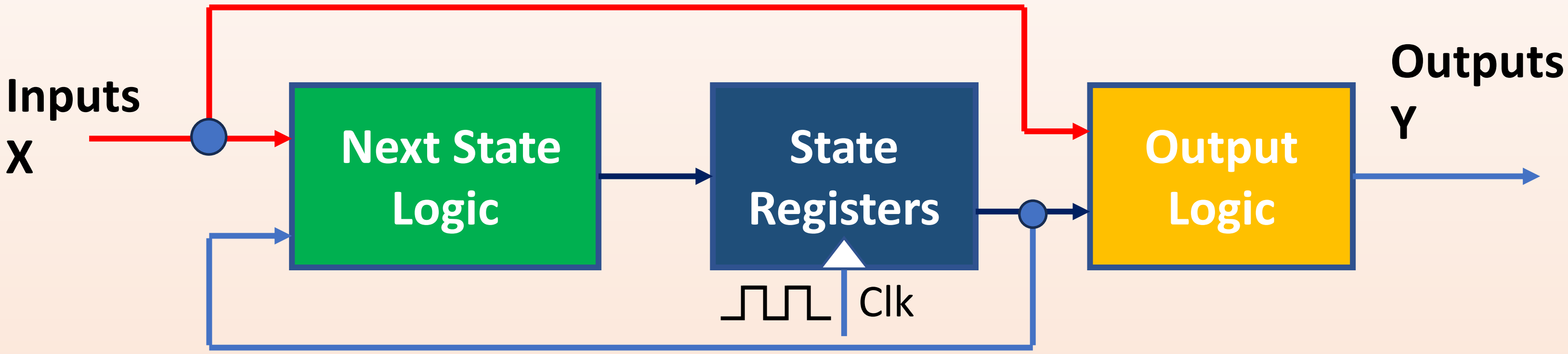


Types of State Machines

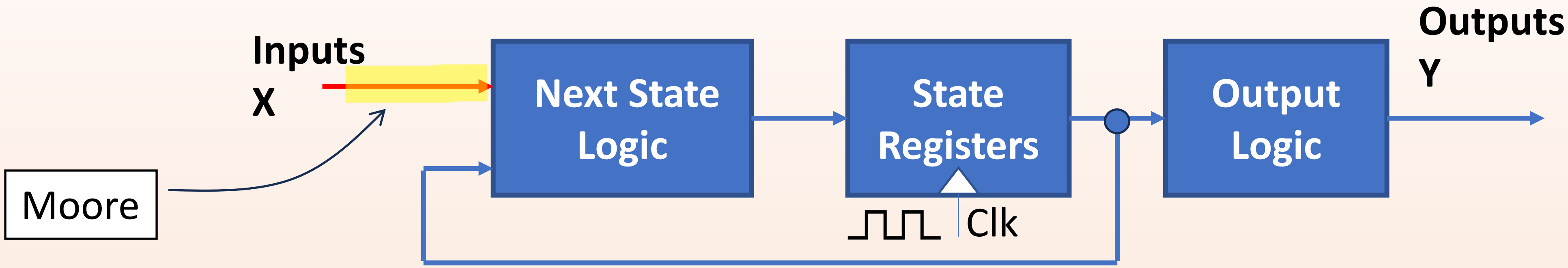
- **Mealy Machine**
  - Output depends upon Internal State plus External Inputs
  - Output can change at any time and not necessarily after a Clocked event

- **Moore Machine**
  - Output depends upon External Inputs and Current Internal State
  - Output is Synchronized with the Change in Internal States

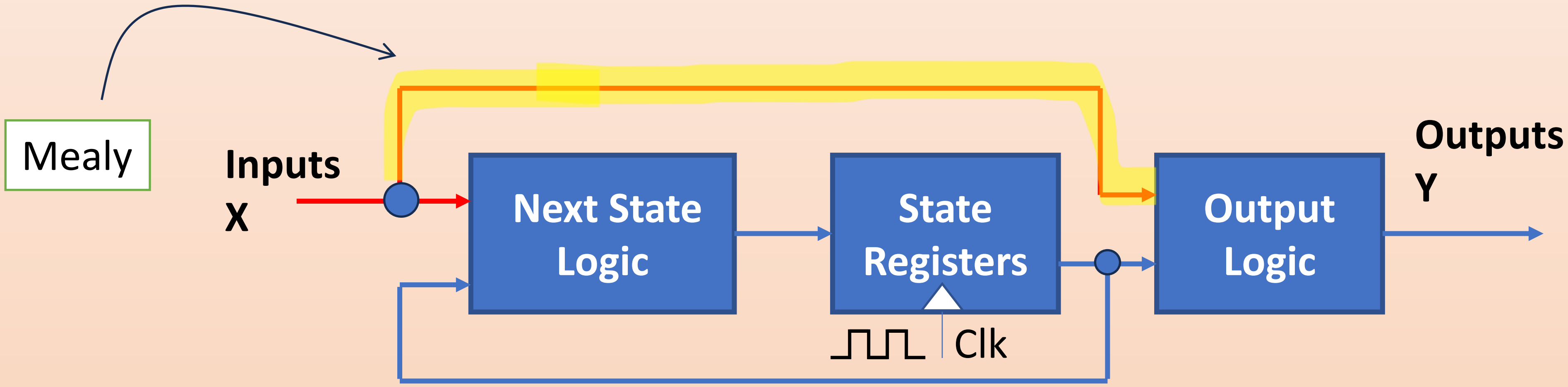
Mealy State Machine Block Diagram



State Machine Block Diagrams - Comparison



Side by side



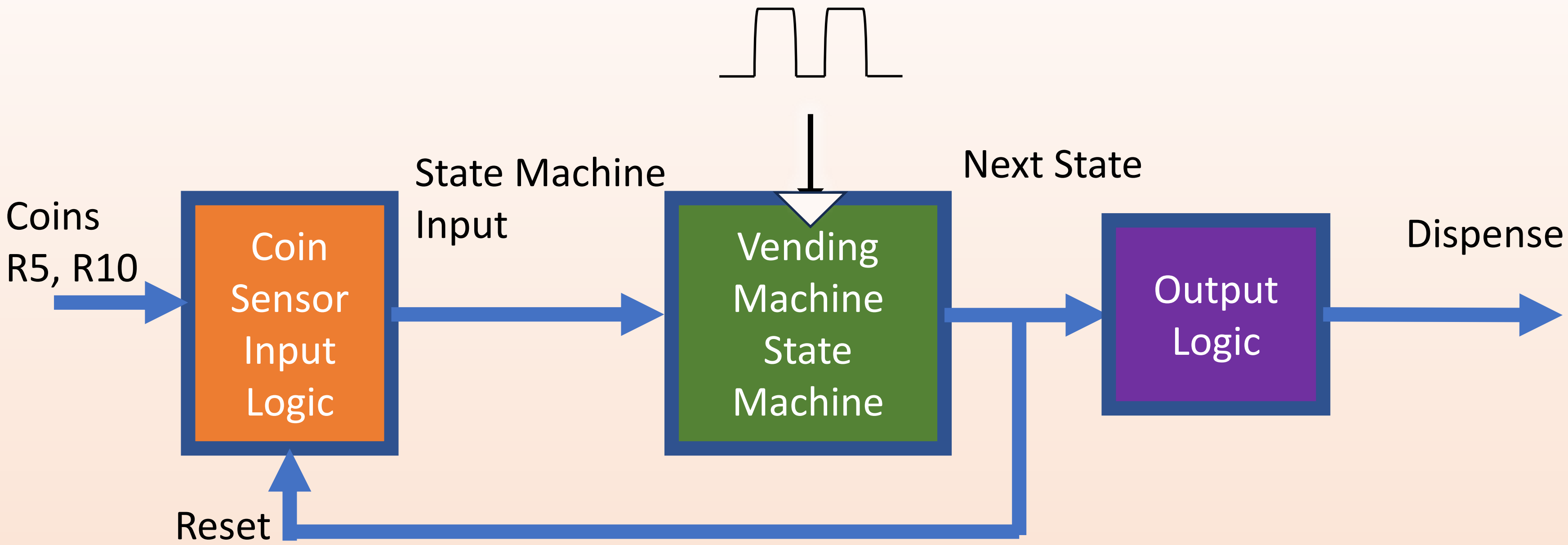
State Machine Based Synchronous Design

- Understand the Problem – desired sequence based on inputs and present state
- Develop abstract representation of FSM – A state diagram or a state table that shows all possible states and transitions
- Perform state minimization – to achieve efficient implementation
- Perform state assignment
- Choose appropriate flipflop for storage elements (eg. D flipflops)
- Use K-maps to determine characteristic equations for Next State
- After K-maps based minimization; draw complete logic circuit using combinational and sequential elements

Example of a State Machine Based Design – Coke (Next Cola 👍) Vending Machine

- Machine dispenses a can of coke for Rs. 15/-
- You can provide coins of either Rs. 5/- or Rs. 10/-
- The Machine does not provide any change back
- The Machine is ‘Reset’ after the can has been dispensed

Block Diagram of Vending Machine Example



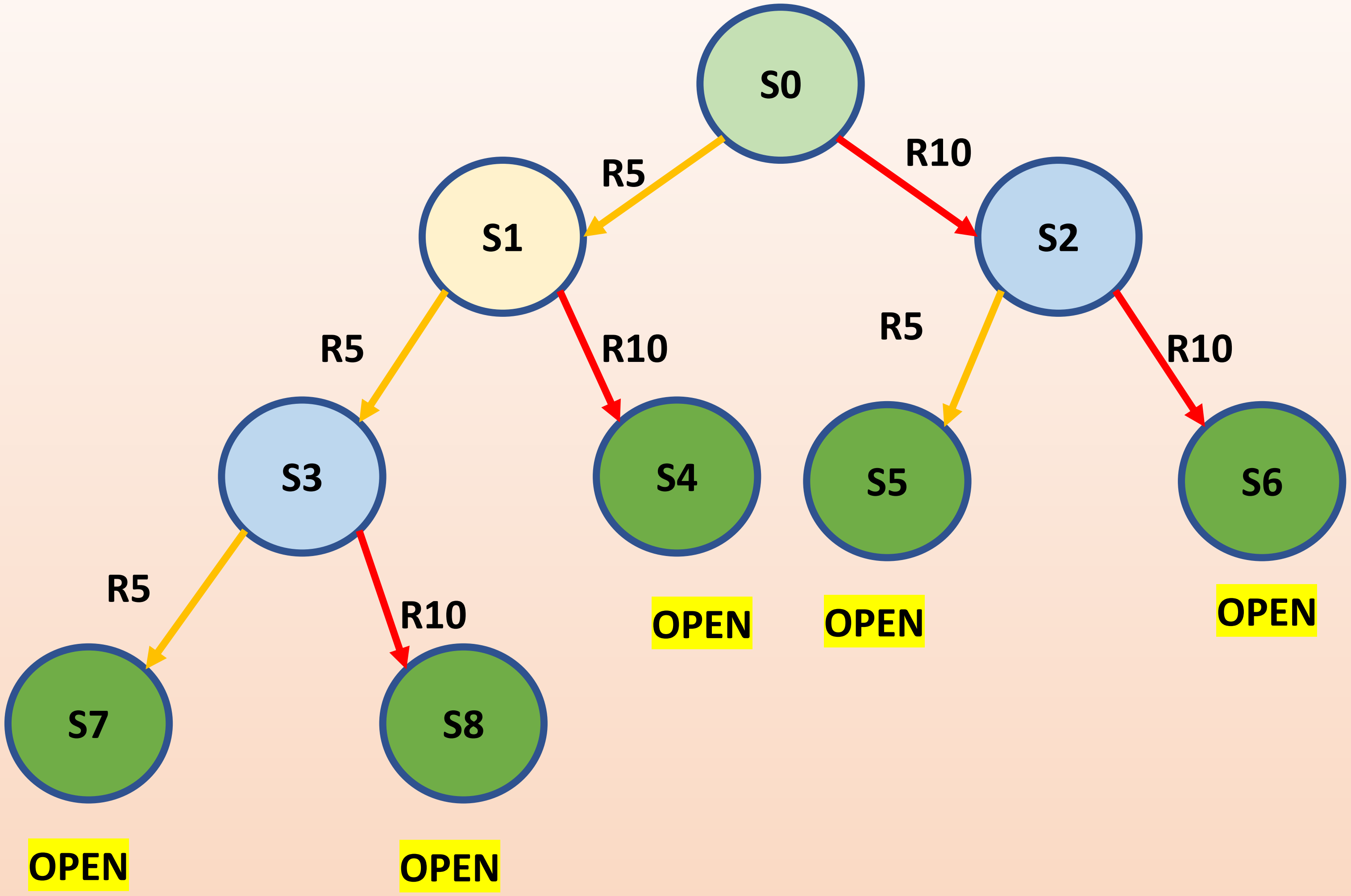
Is Input Directly Connected to Output Logic?

Question: Which type is this? Moore or Mealy??

Elaborate State Machine in text description

- Enumerate all possible inputs and outputs
- Objective: Insert sufficient coins to release a can of Coke
- Either Insert R5 + R5 + R5 in sequence
- OR Insert R5 + R10 in sequence
- OR Insert R10 + R5 in sequence
- OR Insert R10 + R10 in sequence
- OR Insert R5 + R5 + R10 in sequence

Make a State Diagram Representation of State Machine (Moore Machine)

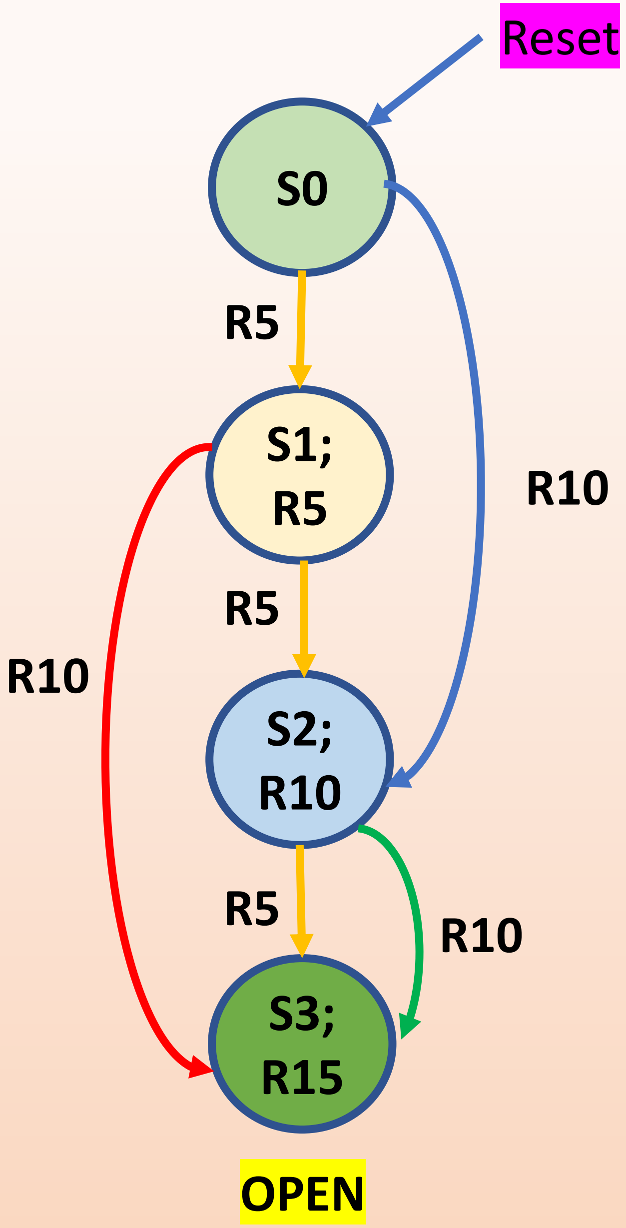


Simplifying the State Machine

Identify Similar States (same present state and next state)

State Minimization by Observation

- **Reset brings to State S0**
- **State S1 represents R5 received so far, one possible path**
- **State S2 represents R10 received so far, two possible paths;**
- **State S3 represents R15 received so far, three possible paths**
- **S4, S5, S6, S7 have identical behaviour so they can be Combined into one state**



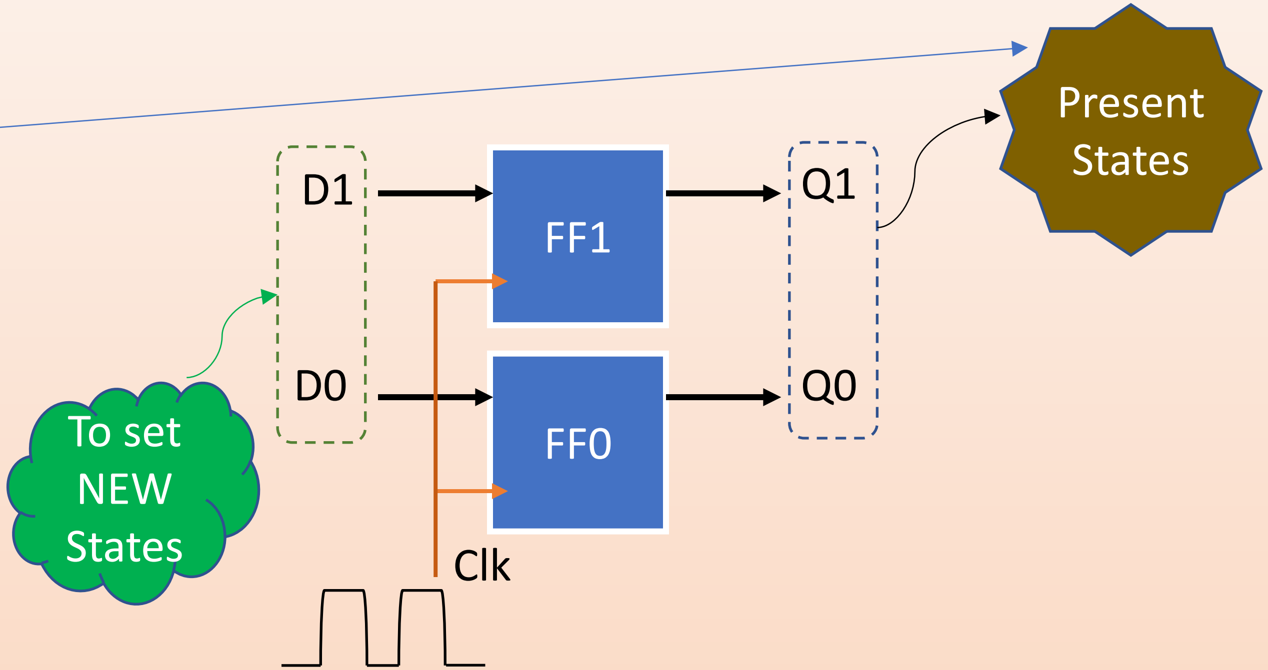
State Transition Table Describing Vending Machine

Q1	Q0	Present State	Inputs		Next State	Output OPEN
			R10	R5		
0	0	S0; R0	0	0	S0; R0	0
			0	1	S1; R5	0
			1	0	S2; R10	0
			1	1	Not Allowed	X
0	1	S1; R5	0	0	S1; R5	0
			0	1	S2; R10	0
			1	0	S3; R15	0
			1	1	Not Allowed	X
1	0	S2; R10	0	0	S2; R10	0
			0	1	S3; R15	0
			1	0	S3; R20	0
			1	1	Not Allowed	X
1	1	S3; R15	0	0	S3; R15	1
			0	1	S3; R15	1
			1	0	S3; R15	1
			1	1	Not Allowed	X

State Mapping to Flipflops

- In the reduced State Diagram, there are 4 states
- We can distinctly represent these states using two Flipflops
- Assign states as follows:

States	Code	Q1	Q0
S0	00	0	0
S1	01	0	1
S2	10	1	0
S3	11	1	1



Present State to Next State Table using DFF

Present State		Inputs		Next State		Output OPEN
Q1	Q0	R10	R5	D1	D0	
0	0 (S0)	0	0	0	0 (S0)	0
		0	1	0	1 (S1)	0
		1	0	1	0 (S2)	0
		1	1	X	X	X
0	1 (S1)	0	0	0	1 (S1)	0
		0	1	1	0 (S2)	0
		1	0	1	1 (S3)	0
		1	1	X	X	X
1	0 (S2)	0	0	1	0 (S2)	0
		0	1	1	1 (S3)	0
		1	0	1	1 (S3)	0
		1	1	X	X	X
1	1 (S3)	0	0	1	1 (S3)	1
		0	1	1	1 (S3)	1
		1	0	1	1 (S3)	1
		1	1	X	X	X

Characteristic Equation for D0 using K-Map

R10,R5\Q1Q0	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	X	X	X	X
10	0	1	1	1

R5'.Q0

R5.Q0'

R10.Q1

Q1Q0

$$D0 = (R5'.Q0) + (R5.Q0') + (R10.Q1) + (Q1.Q0)$$

Characteristic Equation for D1 using K-Map

R10,R5\Q1Q0	00	01	11	10
00	0	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	1	1

Q1

R10

R5.Q0

D1 = Q1 + R10 + (R5.Q0)

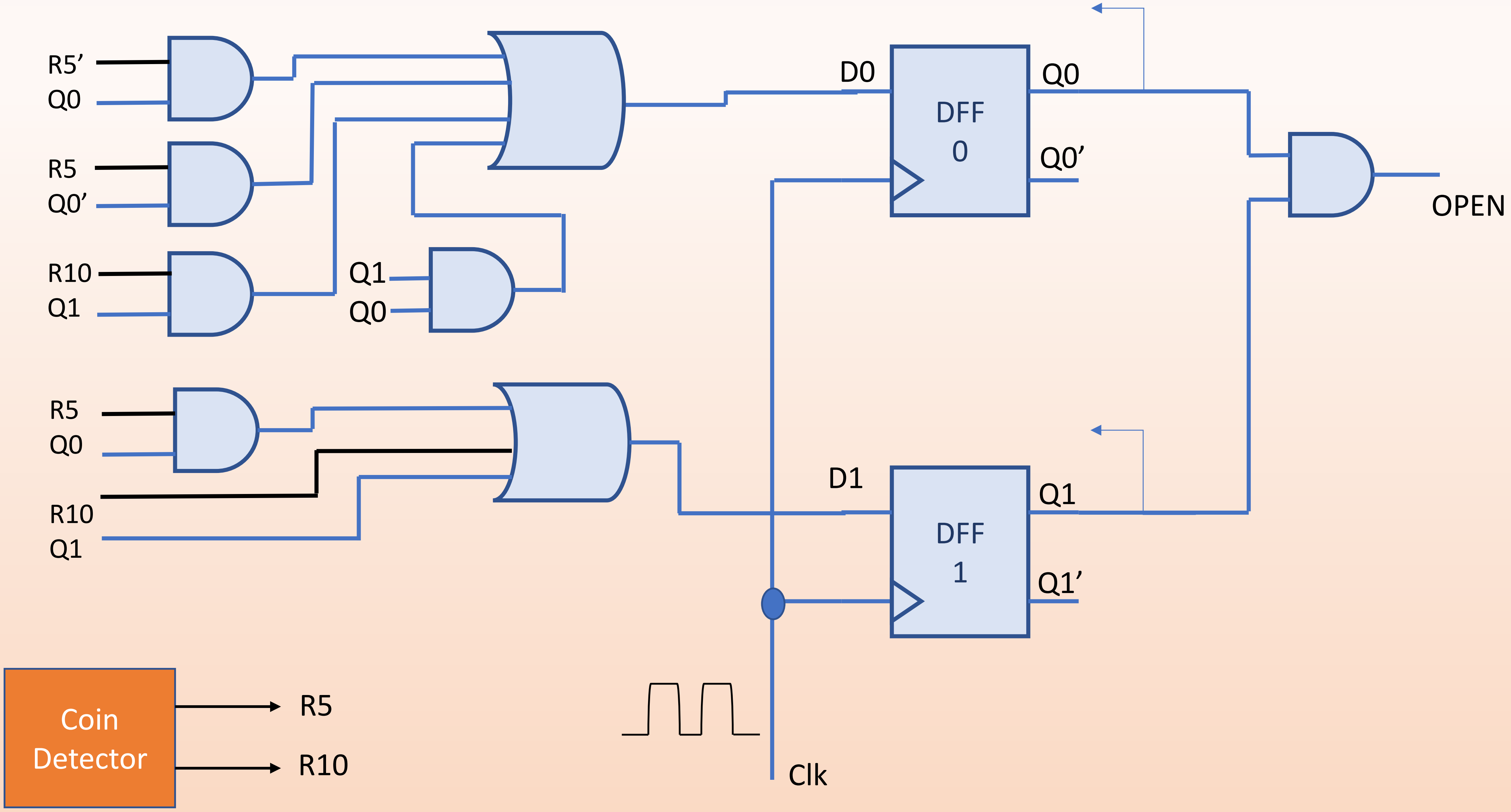
Characteristic Equation for OPEN using K-Map

R10,R5\Q1Q0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	X	X	X	X
10	0	0	1	0

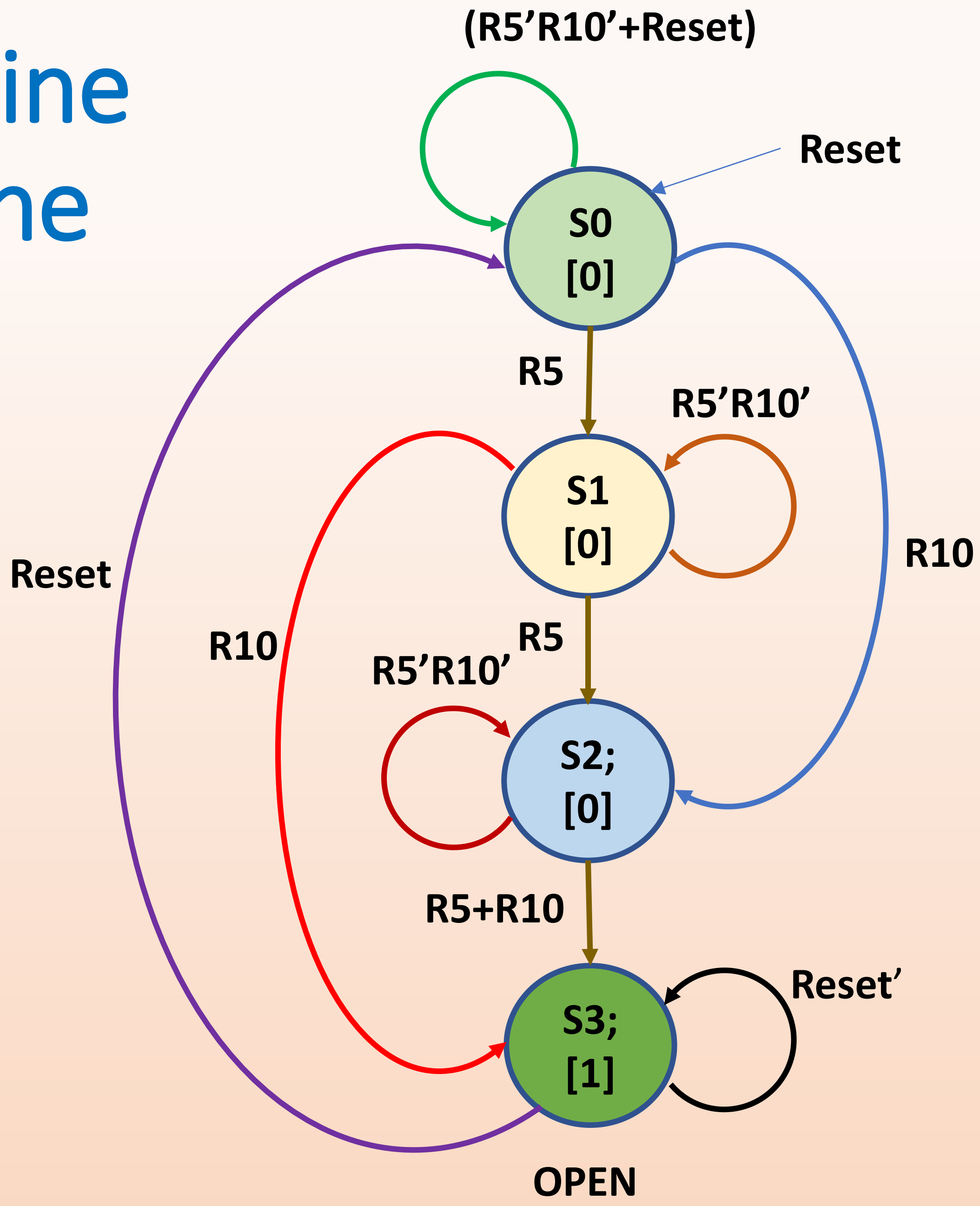
Q1Q0

OPEN = Q1Q0

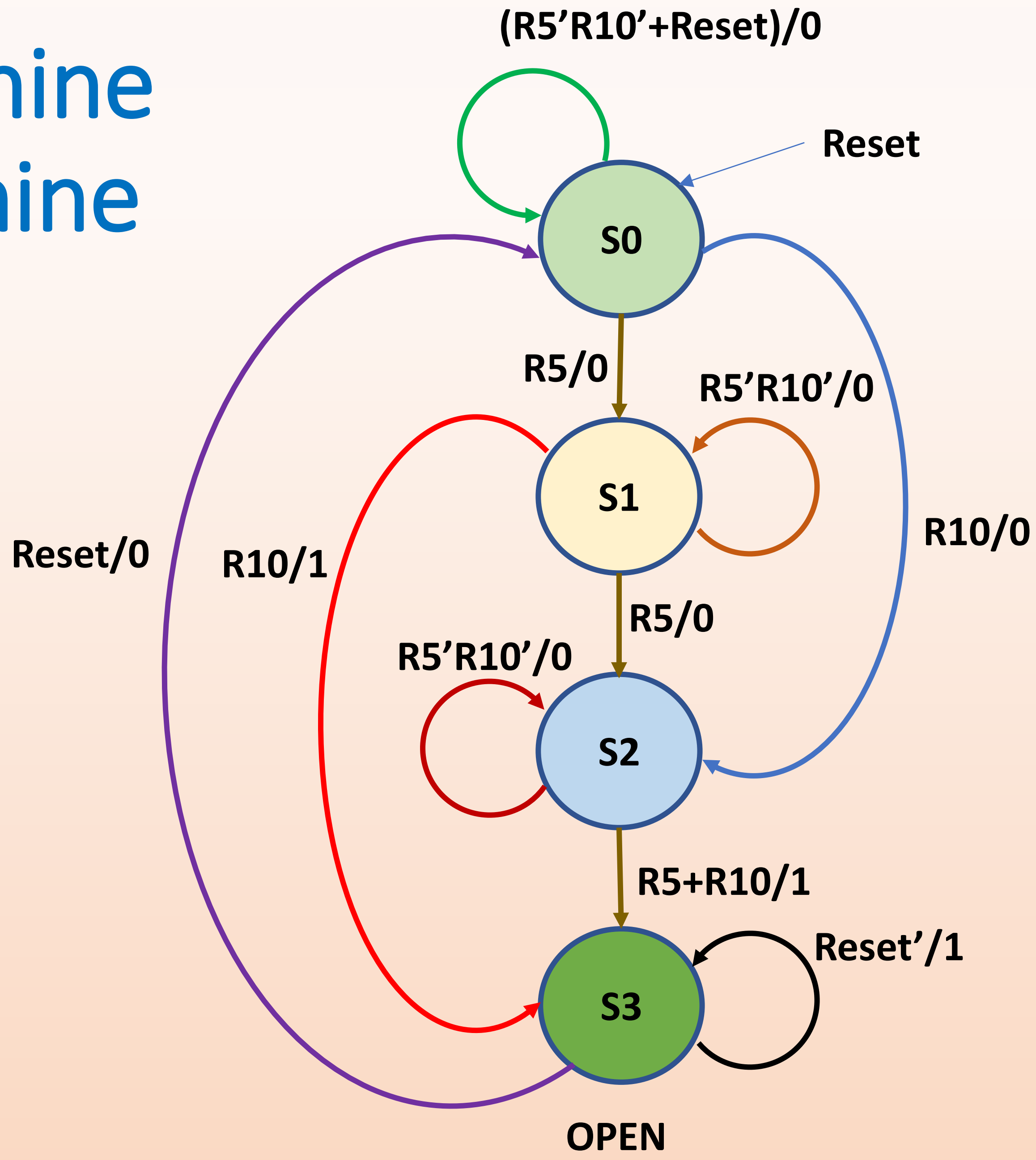
Vending Machine Circuit using DFF



Moore State Machine for Vending Machine



Mealy State Machine for Vending Machine



Some Comparison Moore vs Mealy

- Mealy machine requires fewer states to reach output in comparison with Moore machine
- Mealy machine is more susceptible to glitches
- Explicit output values are shown in Mealy machine associated with each transition
- Output changes after state is changed in Moore machine
- Output in Moore machine depends upon state only; inputs can steer the output towards a particular state that affects output
- Output depends upon present state and the present value at the input; thus, output can change immediately with the change in input, independent of synchronous clock.

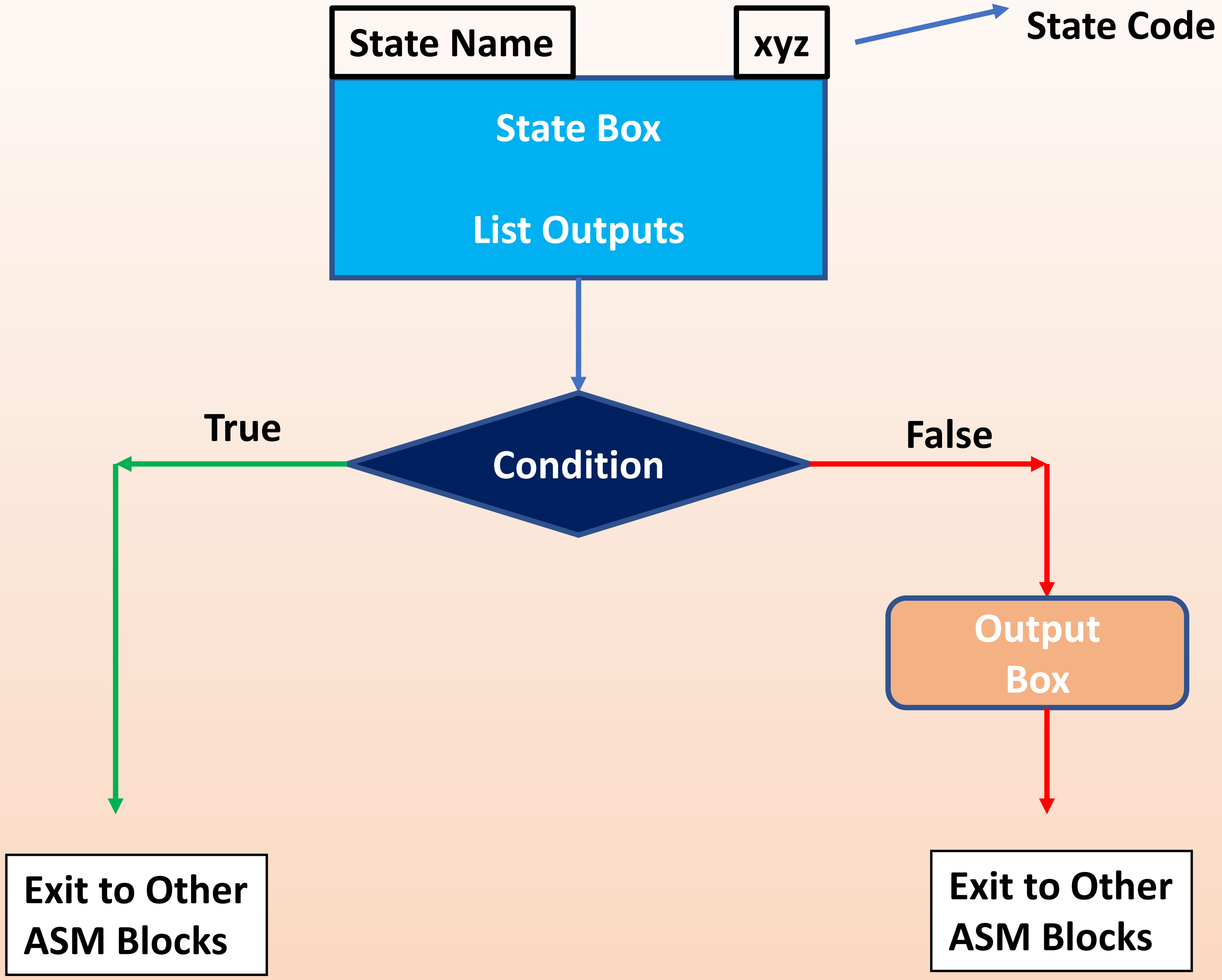
One Hot Encoding – one FF for each state

Present State				Inputs		Next State				Output OPEN
Q3	Q2	Q1	Q0	R10	R5	D3	D2	D1	D0	Y
0	0	0	1	0	0	0	0	0	1	
				0	1	0	0	1	0	
				1	0	0	1	0	0	
				1	1	X	X	X	X	
0	0	1	0	0	0	0	0	1	0	
				0	1	0	1	0	0	
				1	0	1	0	0	0	1
				1	1	X	X	X	X	
0	1	0	0	0	0	0	1	0	0	
				0	1	1	0	0	0	1
				1	0	1	0	0	0	1
				1	1	X	X	X	X	
1	0	0	0	X	X	1	0	0	0	1

D3 is directly the Output and Its State

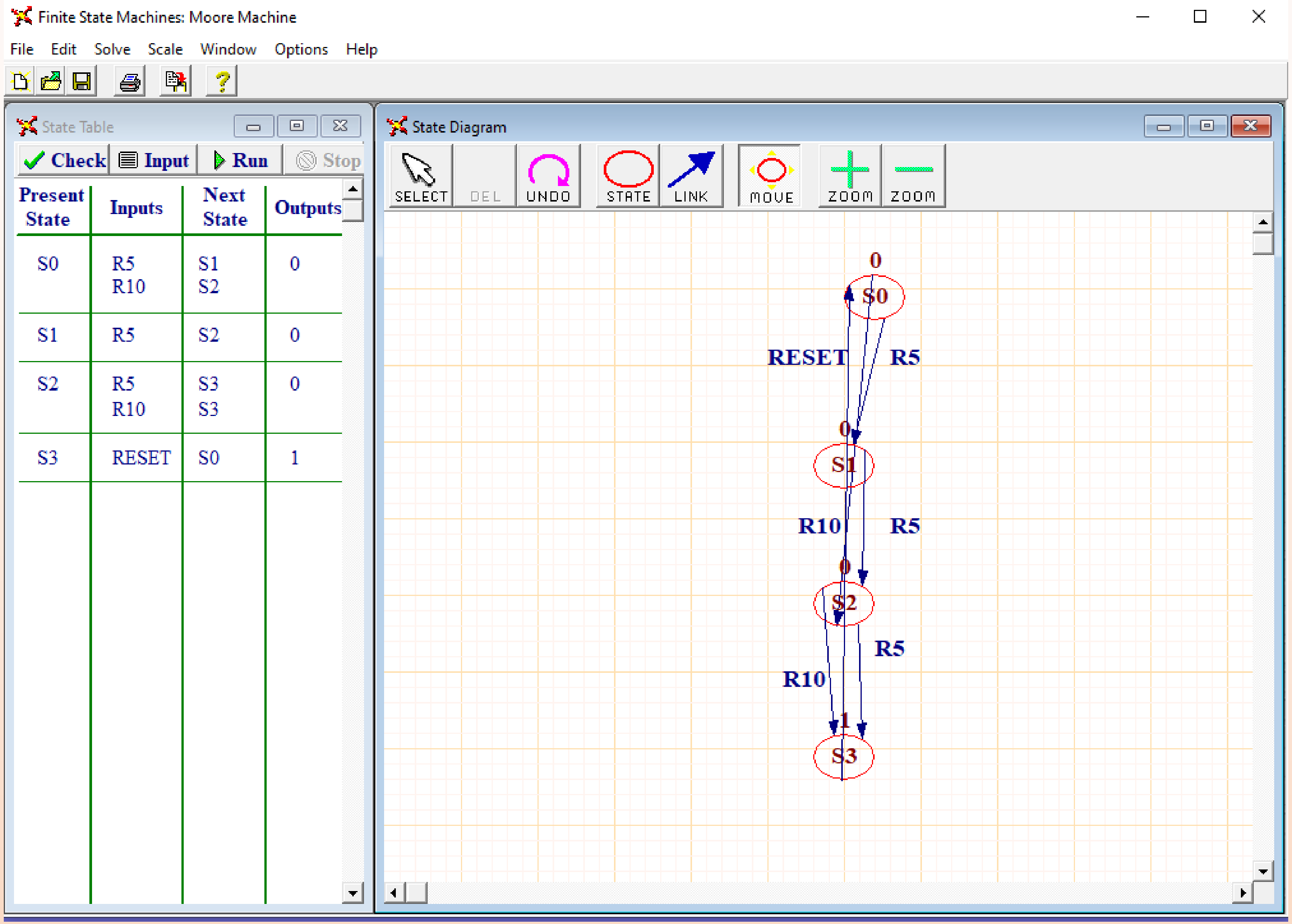
The Design Becomes Simpler
Less Combinational Logic – At the Expense of Extra DFF

Algorithmic State Machine Description - ASMD



Topics

- (Leftover) Example of Vending Machine
- State Minimization
- Row Matching Method
- Using Implication Charts
- **QUIZ 2 in lecture #10**



Trying FSM implementation with WinlogiLab

Advantages of Minimum States

- Reduces number of logic gates and flipflops required for the implementation of state machine
- With fewer states, there are more don't care conditions into the next-state and output logic equations, making the implementation simpler.
- Simpler and less logic means shorter critical paths and higher achievable clock rate
- Fewer components also means shorter design time and lower manufacturing cost

State Minimization and Reduction

- **State Reduction identifies and combines states that have equivalent behaviour**
- **Two states have equivalent behaviour iff:**
  - **for all input combinations, their outputs are the same, and**
  - **they change to the same or equivalent next state**

State Reduction Algorithms

Begin with the symbolic state transition table.

We group together states that have the same state outputs (Moore Machine) or Transition outputs (Mealy Machine). These are potentially equivalent, since states cannot be equivalent if their outputs differ.

Next, examine the transitions to see if they go to the same next state for every input combination. If they do, the states are equivalent and can be combined into a renamed new state.

Then change all transitions into the newly combined states.

Repeat the process until no additional states can be combined.

Row Matching method for State Reduction

Use Case of a Four-Bit Sequence Detector

Specifications:

The machine has a single input X and a single output Z

The output is asserted after each 4-bit input sequence if the sequence contains binary Strings of “0110” or “1010”

The machine returns to “Reset” State after each 4-bit sequence

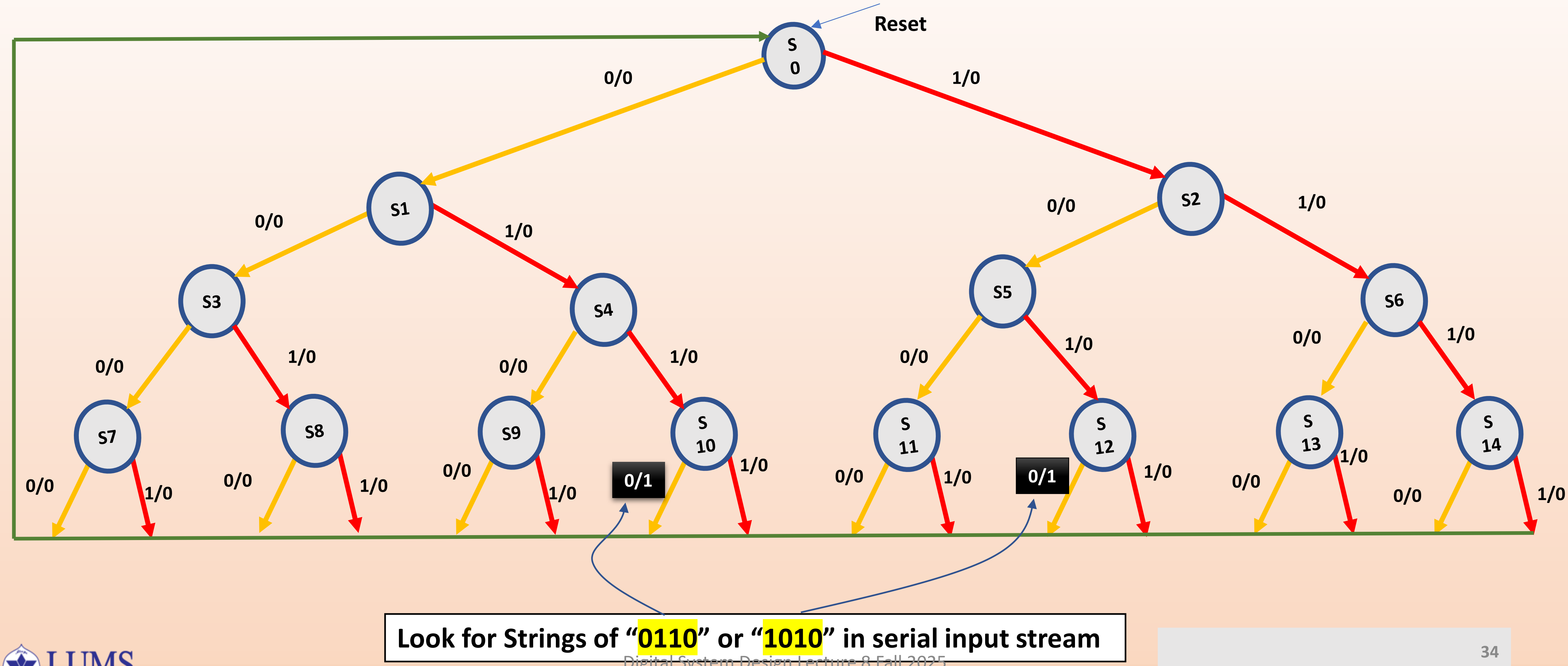
Assumptions:

Mealy Implementation

Example:

Input	X= 0010	0110	1100	1010	0011	…….	…….
Output	Z = 0000	0001	0000	0001	0000 ……	……	

Mealy State Machine for 4-Bit Sequence Detector



Initial State Transition Table

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011	S10	S0	S0	1	0
100	S11	S0	S0	0	0
101	S12	S0	S0	1	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0

Examine table to find rows with Identical next state and output values

Eg. We can combine S10 and S12; Both have same next state and same output

Merge S10 and S12 into one state “S10A” and make new table

Iteration:
States S7, S8, S9 and S11, S13, S14 have same Next states and same outputs; Hence these can be combined

Input Sequence	Present State	Next State		Output	
		When X=0	When X=1	When X=0	When X=1
Reset	S0	S1	S2	0	0
0	S1	S3	S4	0	0
1	S2	S5	S7	0	0
00	S3	S7	S8	0	0
01	S4	S9	S10	0	0
10	S5	S11	S12	0	0
11	S6	S13	S14	0	0
000	S7	S0	S0	0	0
001	S8	S0	S0	0	0
010	S9	S0	S0	0	0
011 or 101	S10A	S0	S0	1	0
100	S11	S0	S0	0	0
110	S13	S0	S0	0	0
111	S14	S0	S0	0	0