EE1D1: Digital Systems A

BSc. EE, year 1, 2025-2026, lecture 7

Sequential Logic

Computer Engineering Lab

Faculty of Electrical Engineering, Mathematics & Computer Science

TU Delft

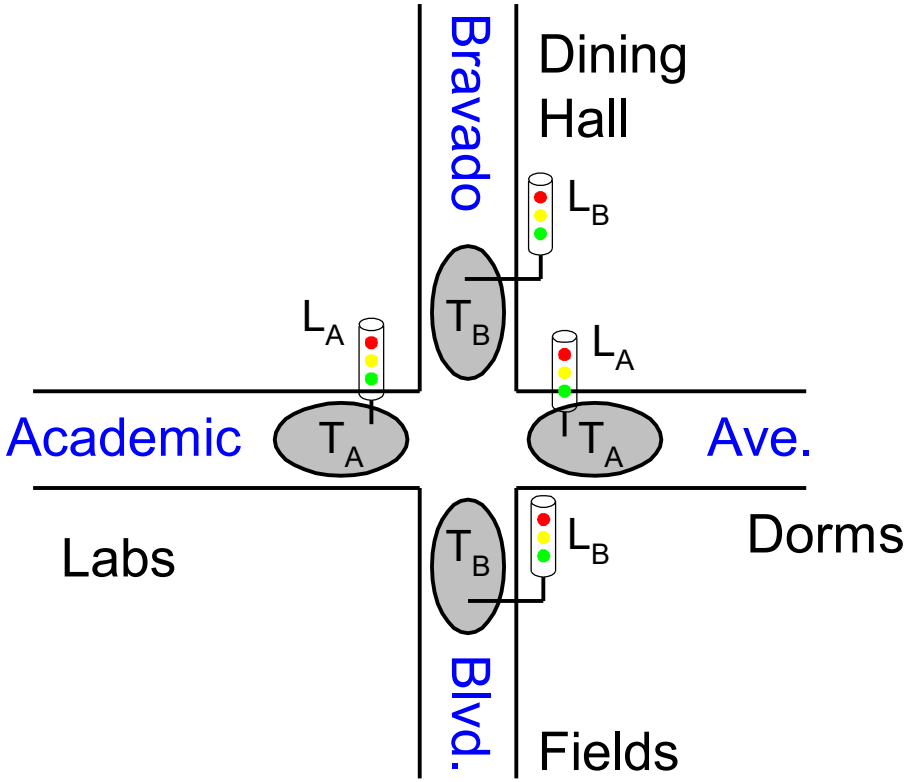
Delft University of Technology

Outline

- **FSM Examples**
  - Traffic light controller
  - Candy dispenser
  - Moore vs Mealy: snail crawler
- **FSM Assignment Courselab**
  - State Table + Karnaugh Maps
  - NAND Gate Implementation
  - NOR Gate Implementation
  - AOI and OAI Gate Implementation
  - Circuit Diagram
  - Simulation

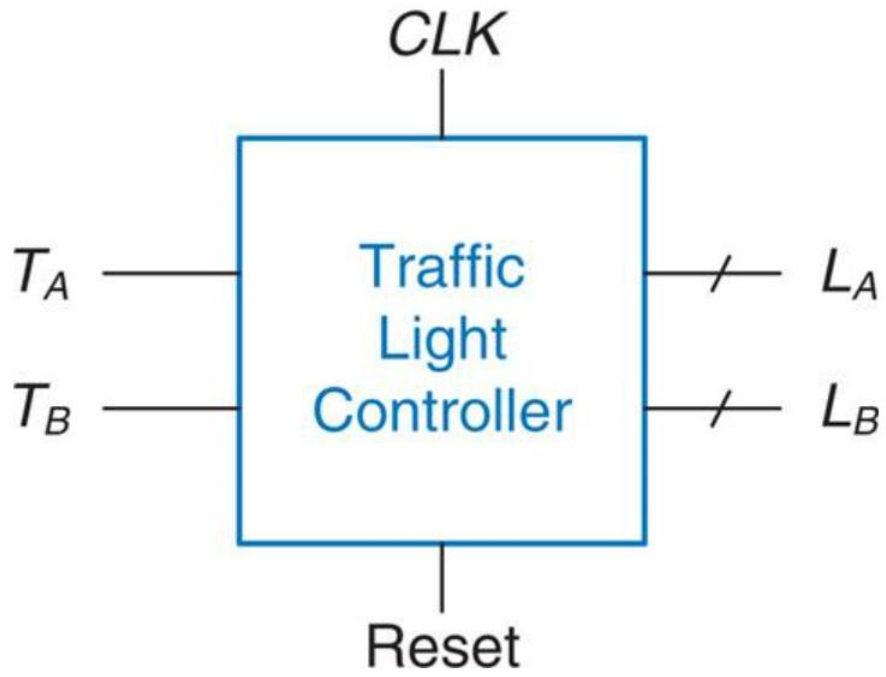
FSM Example: Traffic Light Controller

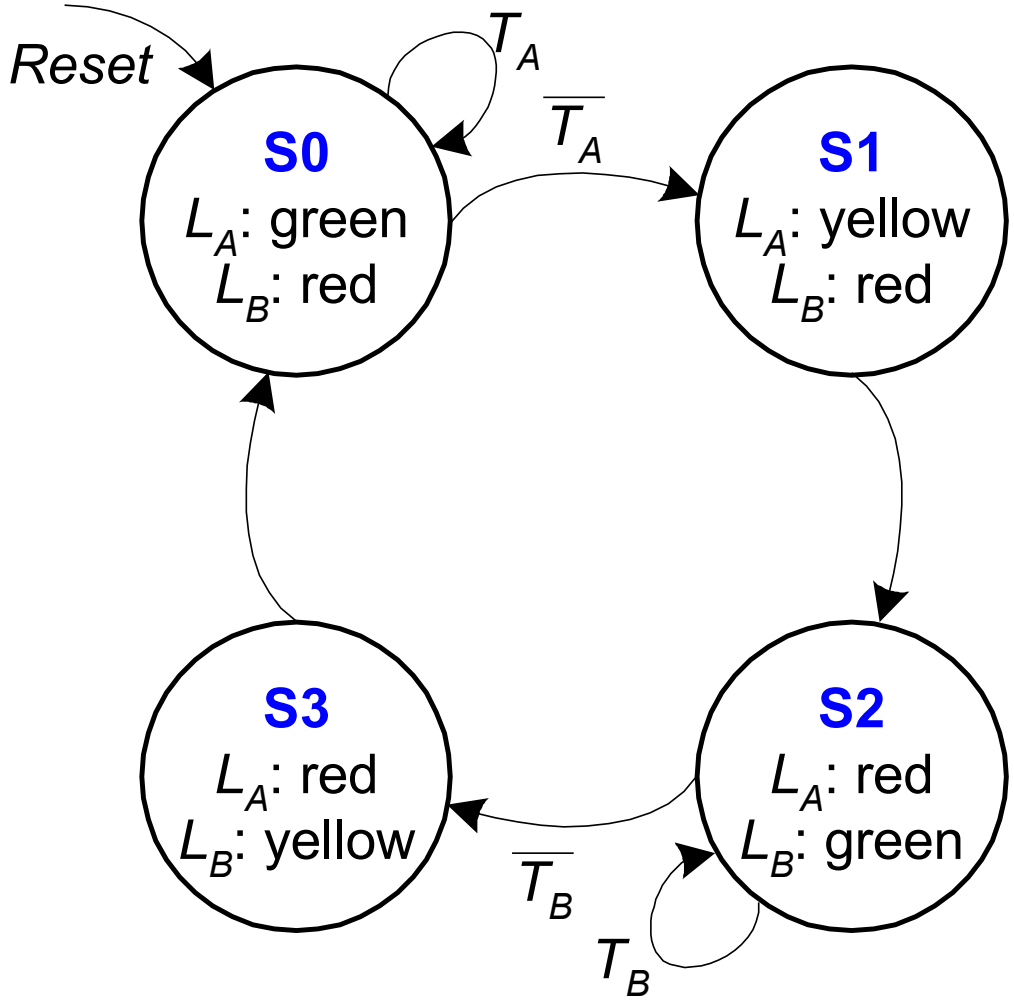
- Traffic light controller at a busy intersection on campus
  - **Traffic sensors:** T_A, T_B (TRUE when there's traffic)
  - **Traffic Lights:** L_A, L_B (Three possible colors Green, Yellow and Red)



FSM Black Box and Finite-State Diagram

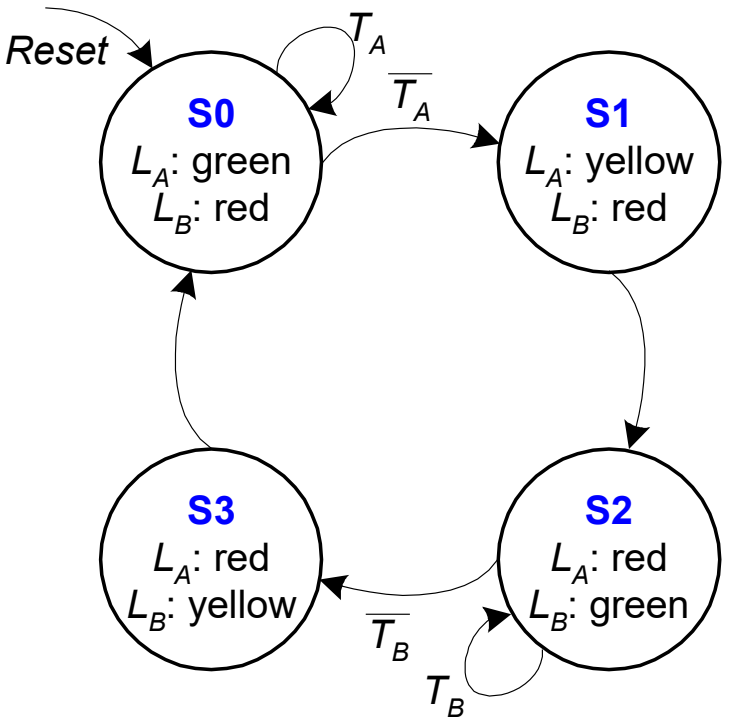
- **Inputs:** *CLK*, *Reset*, T_A, T_B
- **Outputs:** L_A, L_B
- *CLK* period is 5 seconds





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



S : Current State

S': Next State

FSM Encoded State Transition Table

State	Encoding
S0	00
S1	01
S2	10
S3	11

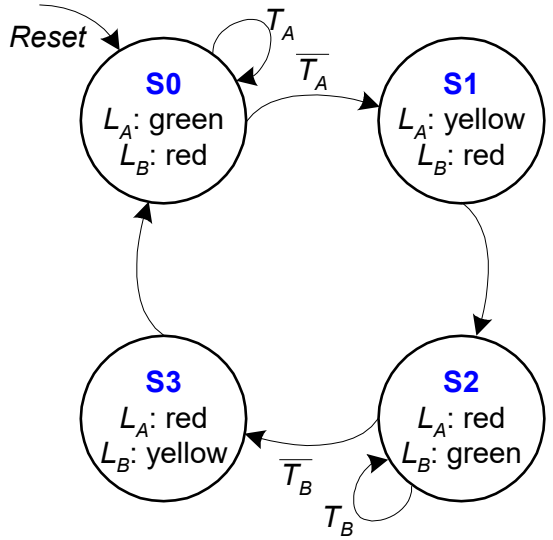
Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

$$S'_1 = \overline{S_1} S_0 + S_1 \overline{S_0}\,\overline{T_B} + S_1 \overline{S_0} T_B = \overline{S_1} S_0 + S_1 \overline{S_0} = S_1 \oplus S_0$$

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

FSM Output Table

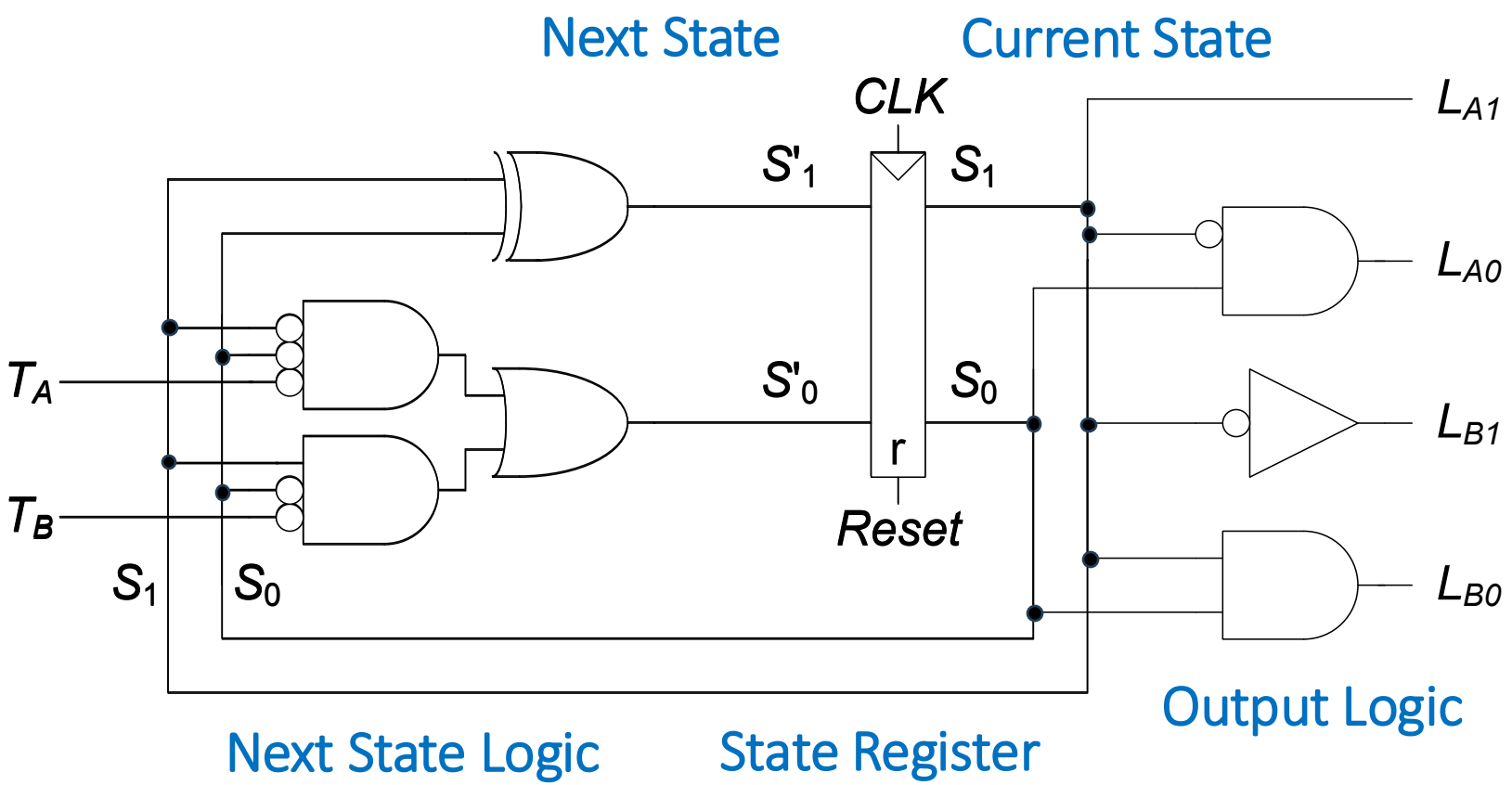
Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1



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

Output	Encoding
green	00
yellow	01
red	10

FSM Schematic



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

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

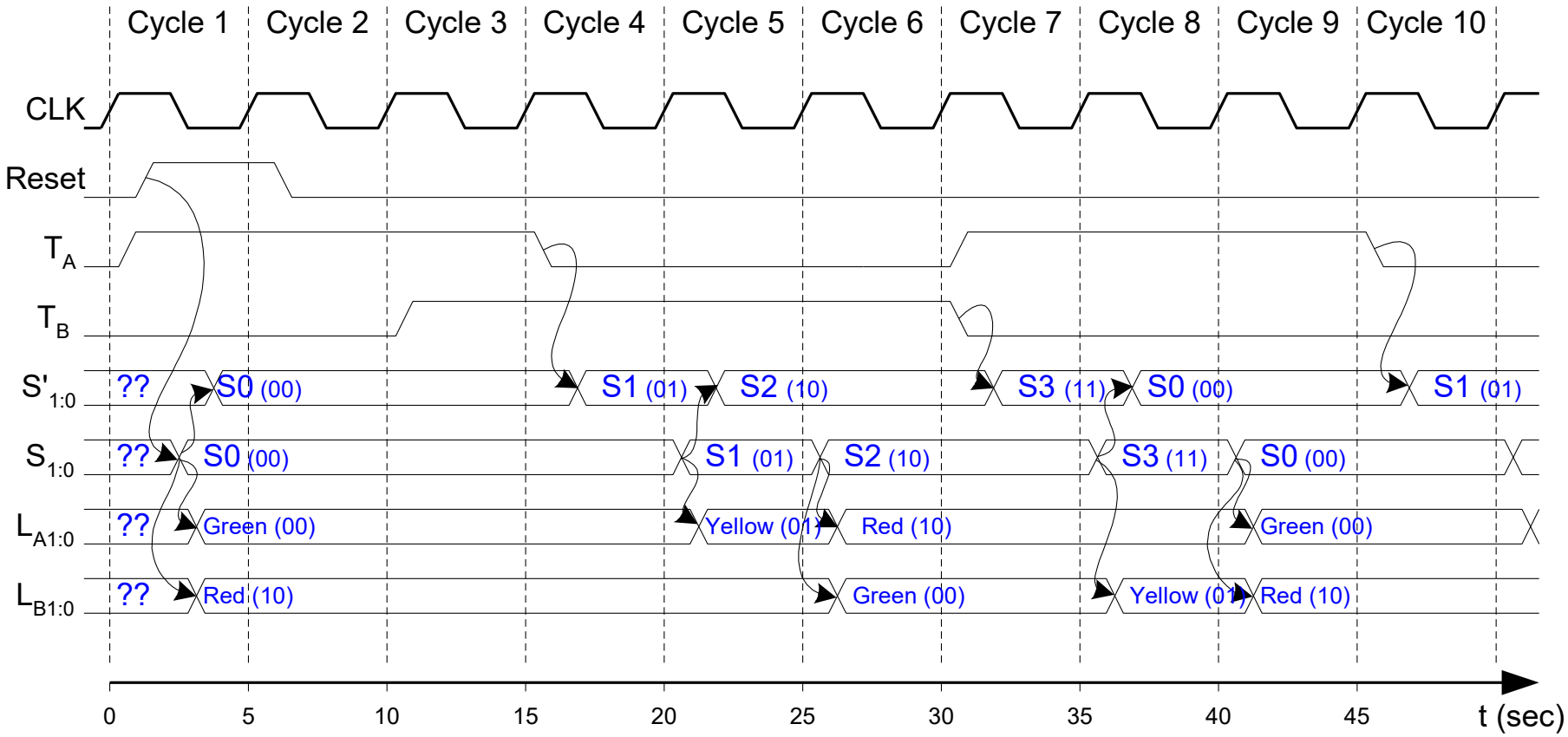
$$L_{A1} = S_1$$

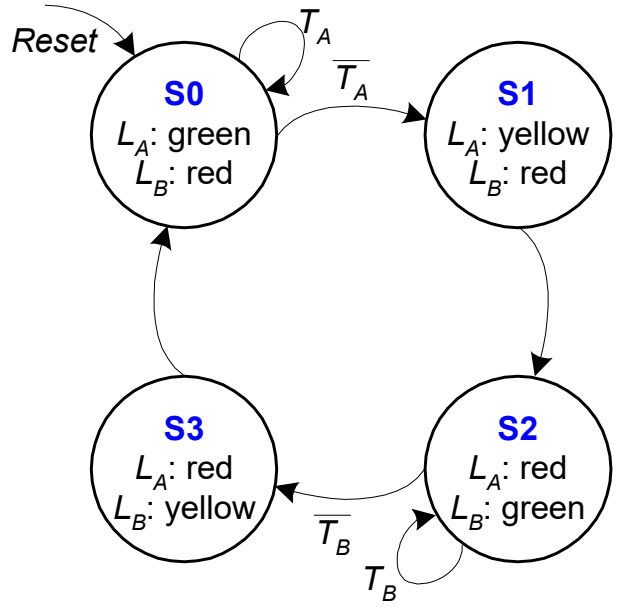
$$L_{A0} = \overline{S}_1 S_0$$

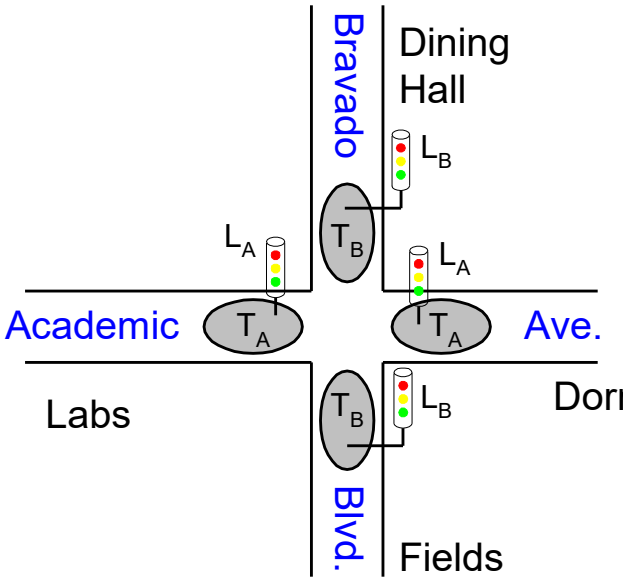
$$L_{B1} = \overline{S}_1$$

$$L_{B0} = S_1 S_0$$

FSM Timing Diagram



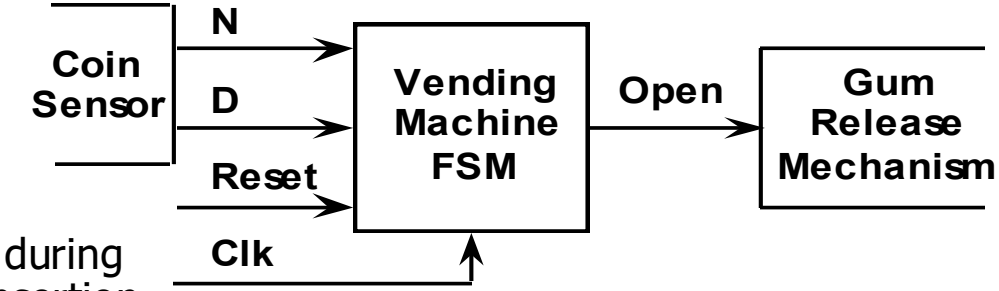




FSM – Example: Candy Dispenser

Problem definition: return gum after at least 15 cents has been inserted, using dimes (10 cent) and nickels (5 cent). No change is returned.

Step 1: Block scheme:



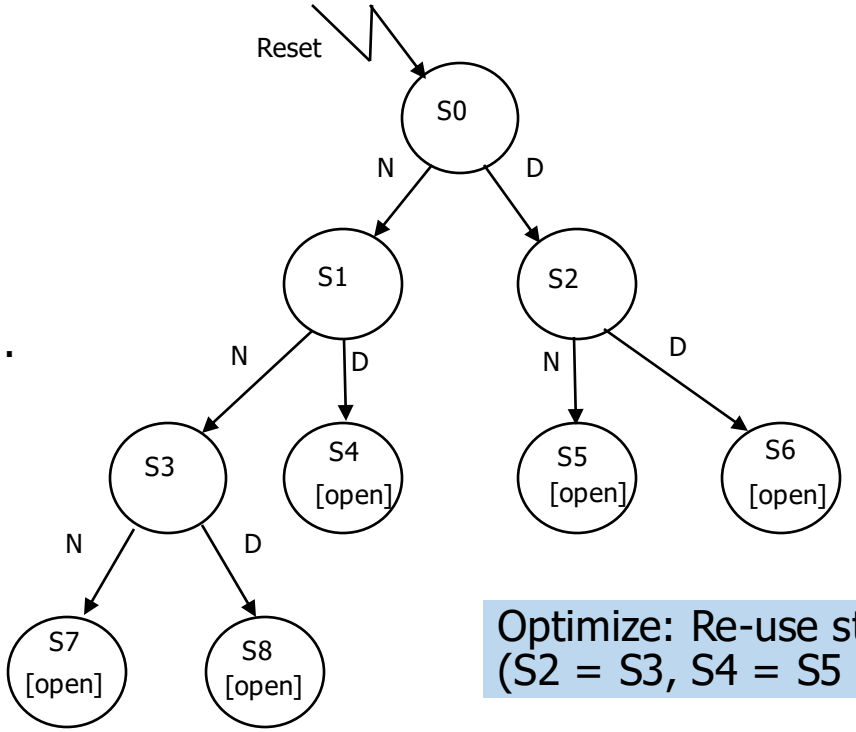
Coin sensor: N=1 or D=1 during 1 clock period after coin insertion

Step 2: State diagram:

States: 0, N, NN, NNN, D, DN, ND, ...

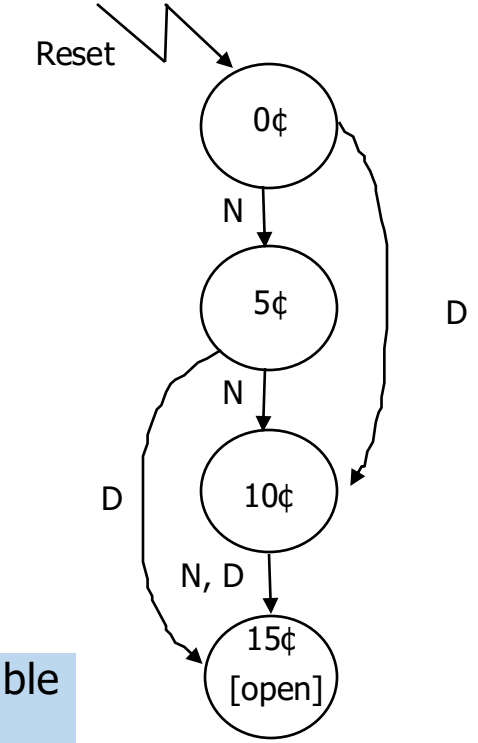
Inputs: N(ickel), D(ime), Reset

Output: open



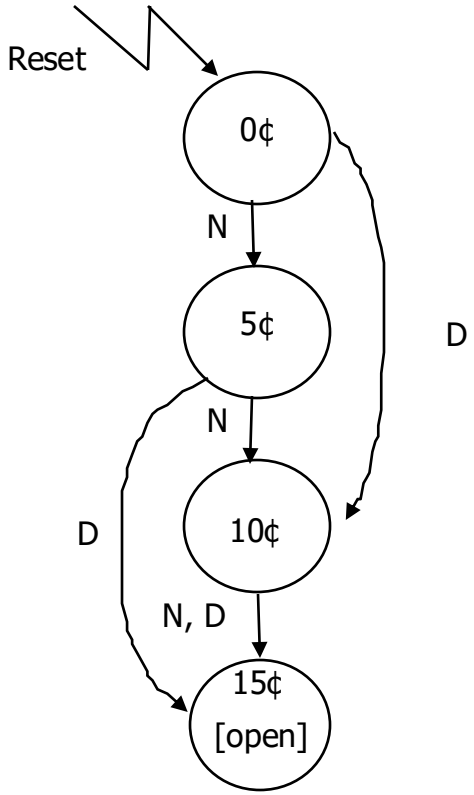
Optimize: Re-use states when possible (S2 = S3, S4 = S5 = ... = S8)

Step 3: Simplification:



FSM – Example: Candy Dispenser

Step 3: Simplification:



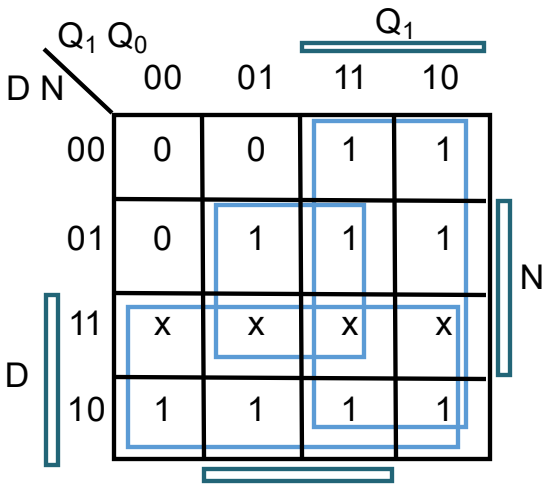
Step 4: State encoding:

Step 5: State table:

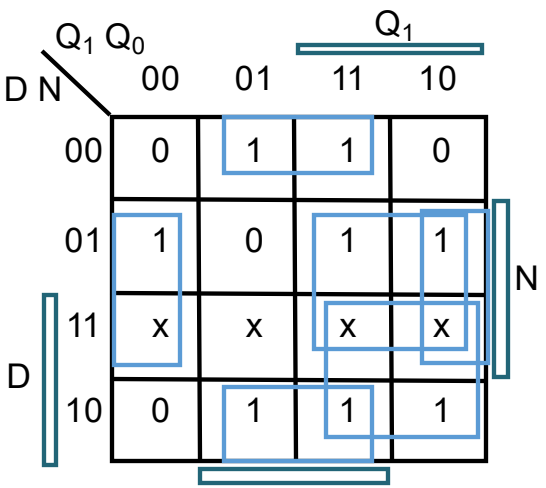
Current State	Q_1	Q_0	Inputs D	Inputs N	Next State	D_1	D_0	Output Open
0¢	0	0	0	0	0¢	0	0	0
			0	1	5¢	0	1	0
			1	0	10¢	1	0	0
			1	1	X	X	X	X
5¢	0	1	0	0	5¢	0	1	0
			0	1	10¢	1	0	0
			1	0	15¢	1	1	0
			1	1	X	X	X	X
10¢	1	0	0	0	10¢	1	0	0
			0	1	15¢	1	1	0
			1	0	15¢	1	1	0
			1	1	X	X	X	X
15¢	1	1	0	0	15¢	1	1	1
			0	1	15¢	1	1	1
			1	0	15¢	1	1	1
			1	1	X	X	X	X

FSM – Example: Candy dispenser

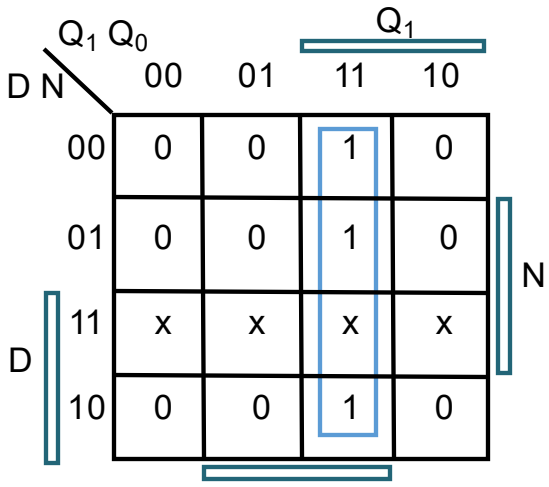
Step 6: Implementation:



K-map for D_1



K-map for D_0

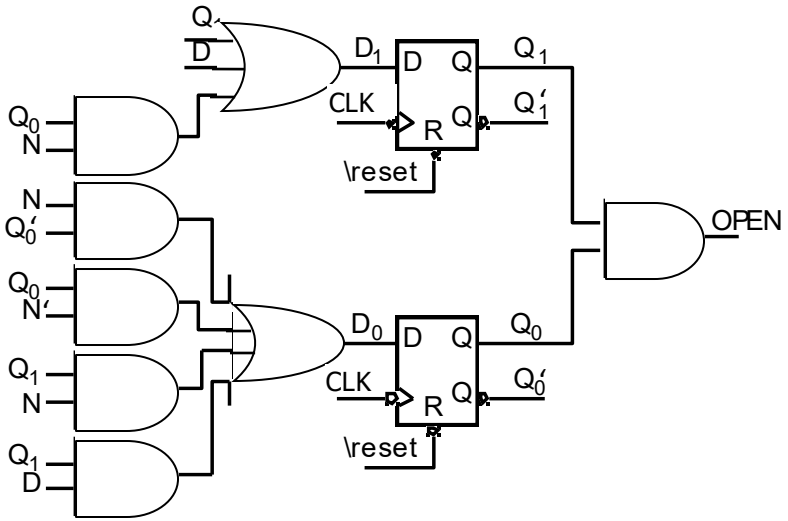


K-map for OPEN

$D_1 = Q_1 + D + Q_0\, N$

$D_0 = N\, Q_0' + Q_0\, N' + Q_1\, N + Q_1\, D$

$OPEN = Q_1\, Q_0$

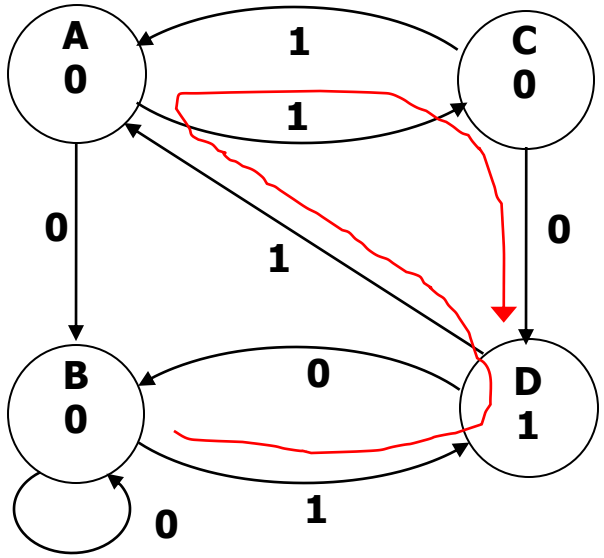


FSM – Question

Given the adjacent state diagram for an FSM with 1 input and 1 output.

Which input sequence (input value per clock period, last input value on the right) will give a 1 at the end?

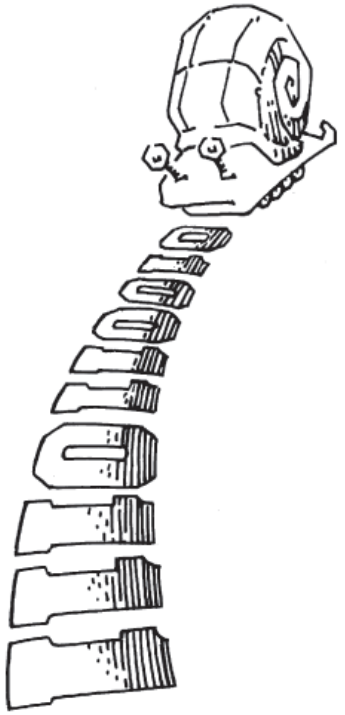
a. 0011
b. 0111
c. 1100
d. 1110



From B to D when input is 1
From D to A when input is 1
From A to C when input is 1
From C to D when input is 0
=> answer d

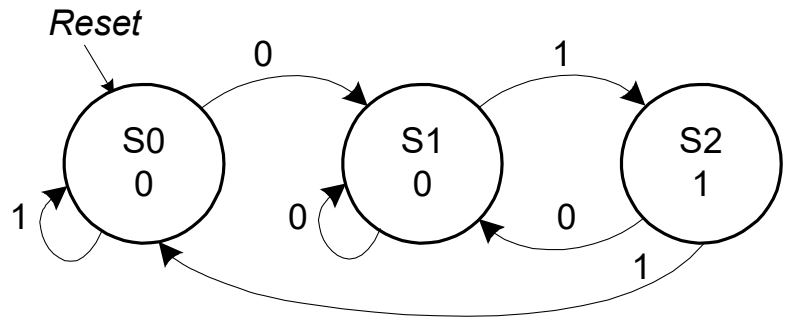
Moore vs. Mealy FSM

Alyssa P. Hacker has a snail that crawls down a paper tape with 1's and 0's on it. The snail smiles whenever the last two digits it has crawled over are 01. Design **Moore** and **Mealy** FSMs of the snail's brain.

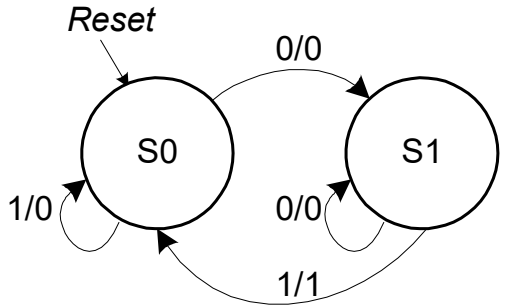


State Transition Diagrams

Moore FSM

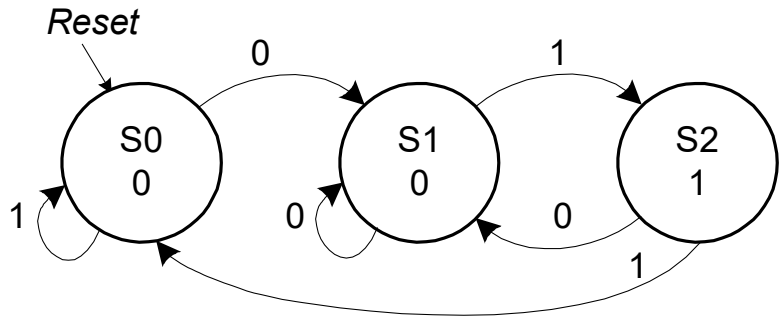


Mealy FSM



Moore FSM State Transition Table

Moore FSM



Current State		Inputs	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
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0

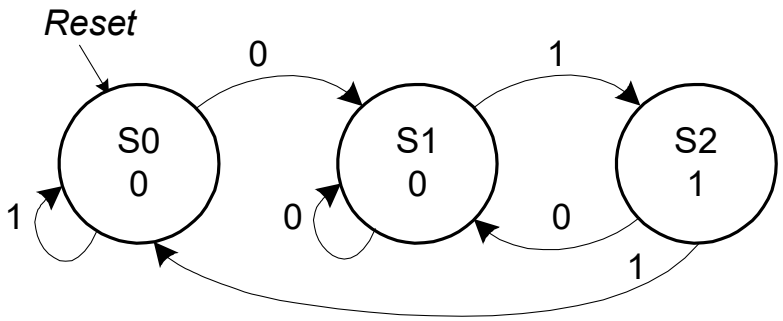
State	Encoding
S0	00
S1	01
S2	10

$$S_1' = S_0 A$$
$$S_0' = \overline{A}$$

Using don't care conditions
Not shown in the table

Moore FSM Output Table

Moore FSM



State	Encoding
S0	00
S1	01
S2	10

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

$$Y = S_1$$

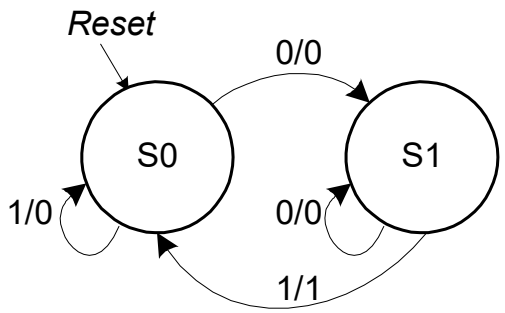
Mealy State Transition & Output Table

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

State	Encoding
S0	0
S1	1

$$S_0' = \overline{A}$$

$$Y = S_0 A$$



Moore FSM Schematic

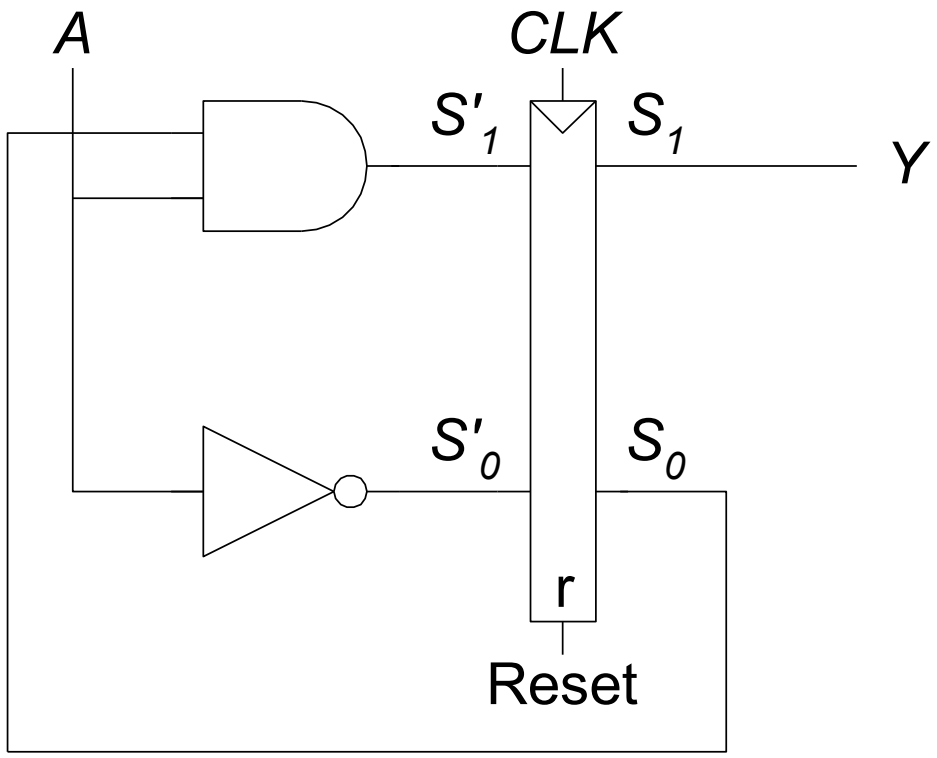
Next State Equations

$S_1' = S_0 A$

$S_0' = \overline{A}$

Output Equation

$Y = S_1$



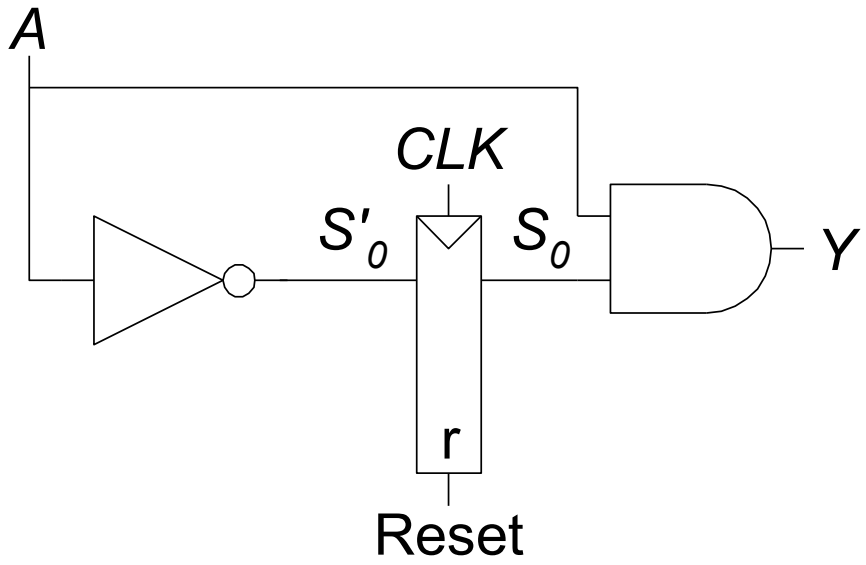
Mealy FSM Schematic

Next State Equation

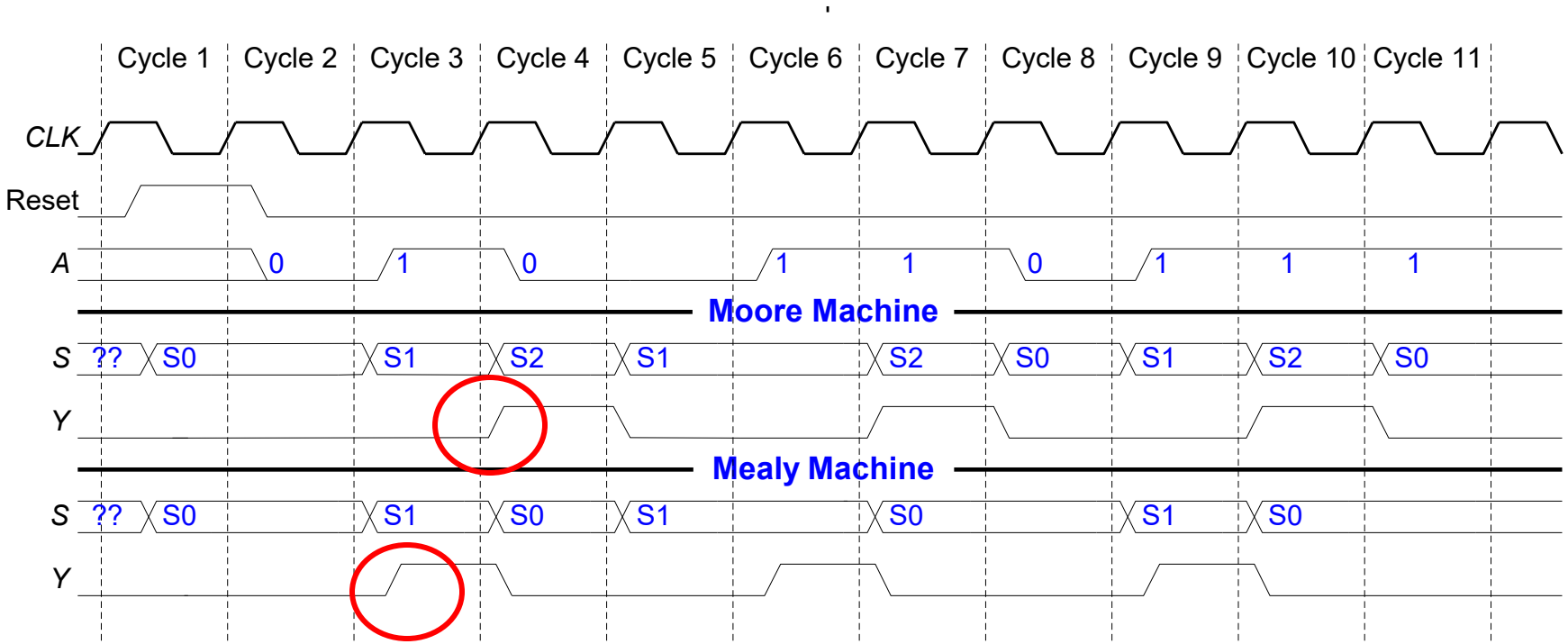
$S_0' = \overline{A}$

Output Equation

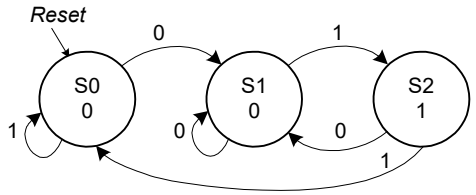
$Y = S_0 A$



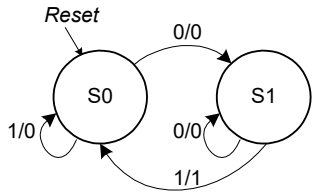
Moore and Mealy Timing Diagram



Moore FSM



Mealy FSM

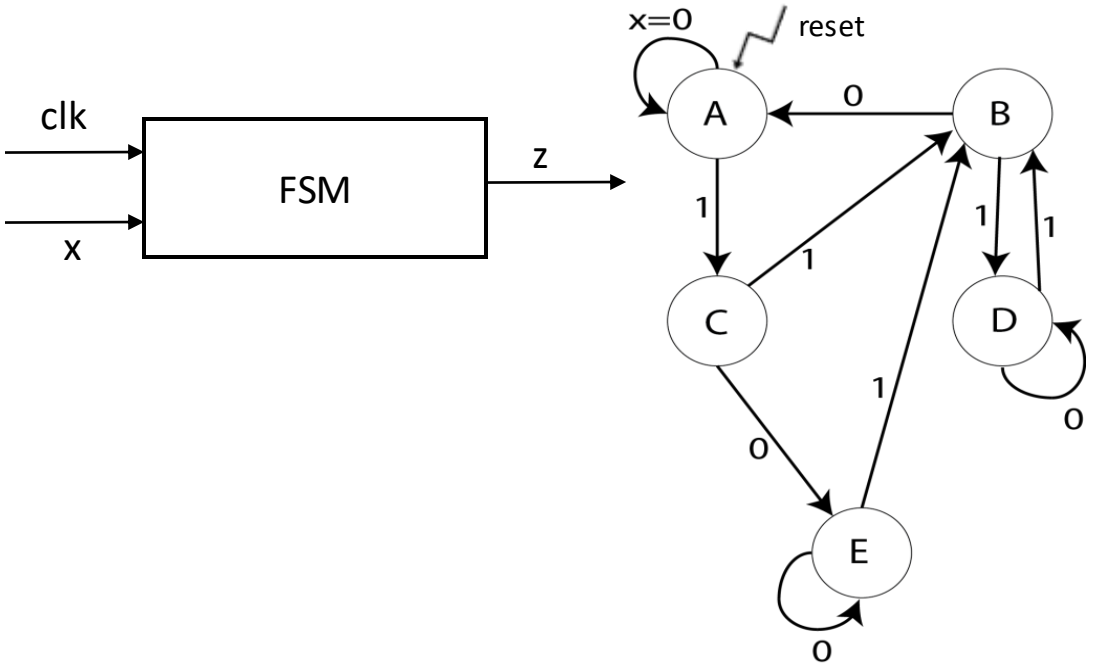


Mealy FSM: asserts Y **immediately** when input pattern 01 is detected

Moore FSM: asserts Y one cycle **after** input pattern 01 is detected

FSM Assignment similar to Courselab Assignment 3B

Given the following state diagram of a Moore machine called "system":



Herein, the five states are coded as follows:

	State Encoding			output
	y2	y1	y0	**z**
A	0	0	0	1
B	1	1	1	0
C	0	1	0	1
D	1	0	1	1
E	1	0	0	0

with state A as the initial state.

The **inputs** of the systems are x ∈ {0,1}, reset ∈ {0,1} and clk ∈ {0,1}.

The **current** state of the system is described by Y = (*y2,y1,y0*) with y*i* ∈ {0,1}

The **output** of the system is described by z ∈ {0,1}.

The functionality of the system is as follows. If reset = 1, the system returns to the initial state "A". If reset = 0, the system operates according to input x and synchronously on the rising clock edge.

Assignment:

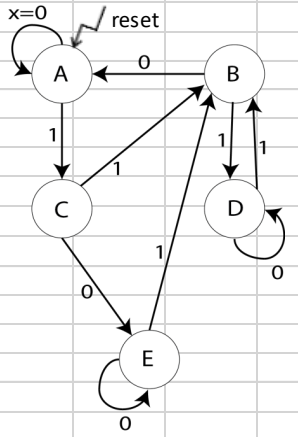
1. Write SystemVerilog description that models the system described above, using only D-flipflops and the gates mentioned at point 2.
2. Implement z using only NOR-gates, y2 using NAND gates, y1 using an AOI gate, and y0 using an OAI gate. All NAND and NOR gates must have 2 or 4 inputs.
3. Write a test bench that verifies your own SystemVerilog model by starting with a reset and then going through all state transitions at least once. (Don't forget the clock signal clk).

State Table + Kargaugh Maps

FSM example assignment

State Table

Current state				input	Next state				output
Symbol	y2	y1	y0	x	Symbol	y2+	y1+	y0+	z
A	0	0	0	0					
	0	0	0	1					
X	0	0	1	0					
	0	0	1	1					
C	0	1	0	0					
	0	1	0	1					
B	0	1	1	0					
	0	1	1	1					
E	1	0	0	0					
	1	0	0	1					
D	1	0	1	0					
	1	0	1	1					
X	1	1	0	0					
	1	1	0	1					
X	1	1	1	0					
	1	1	1	1					



State Encoding				output
	y2	y1	y0	z
A	0	0	0	1
B	1	1	1	0
C	0	1	0	1
D	1	0	1	1
E	1	0	0	0

Karnaugh Maps

y2+ [1]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

y1+ [2]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

y0+ [3]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

z [4]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

Inverted Karnaugh Maps

y2+' [5]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

y1+' [6]

y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

y0+' [7]

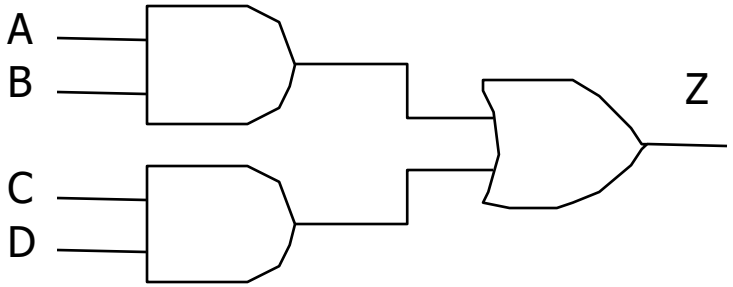
y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

z' [8]

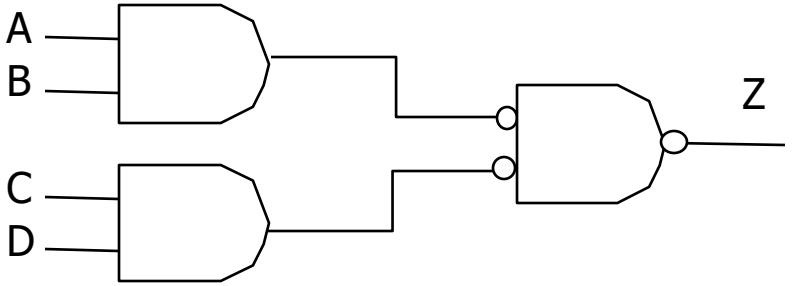
y0x \ y2y1	00	01	11	10
00				
01				
11				
10				

NAND Gate Implementation

- Derive sum-of-products to find NAND implementation



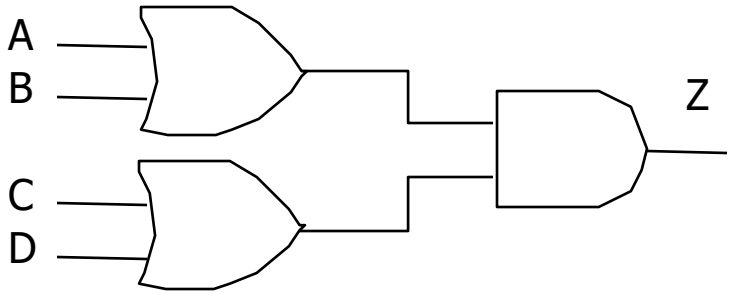
AB+CD



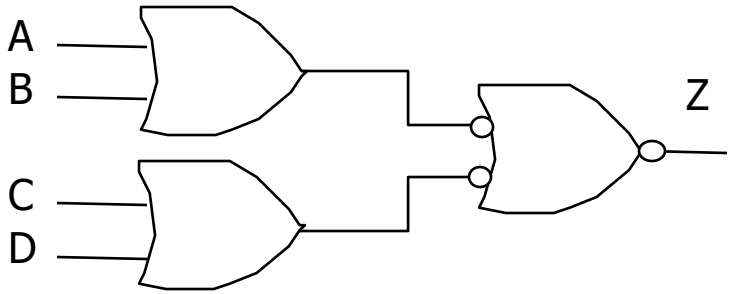
(AB+CD)”
((AB)’ • (CD)’)’

NOR Gate Implementation

- Derive product-of-sums to find NOR implementation



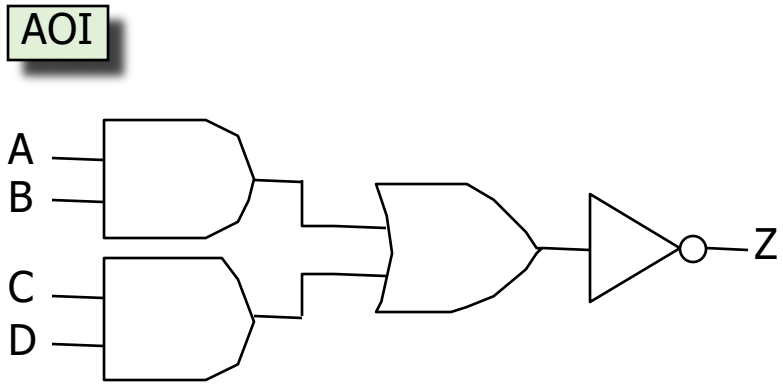
(A+B)(C+D)



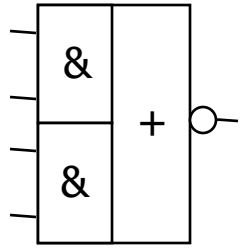
((A+B)(C+D))″ =
((A+B)′ + (C+D)′)′

AOI Gate Implementation

- Derive sum-of-products for inverted output to find AOI implementation

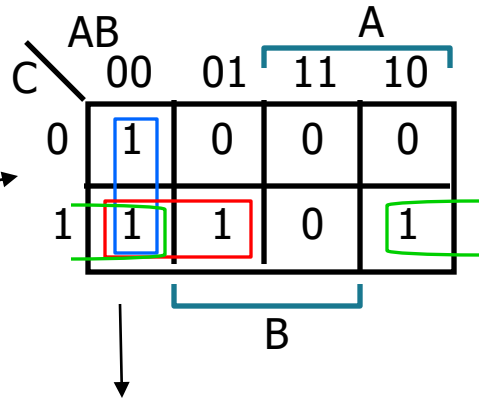


2x2 AOI Schematic Symbol

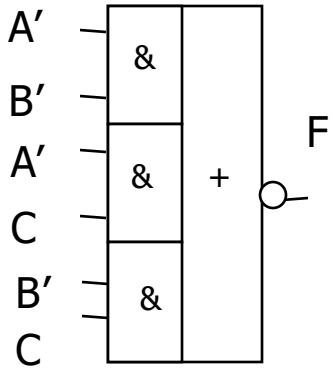


$F = \sum m(2,4,6,7)$

$\Rightarrow F' = \sum m(0,1,3,5)$

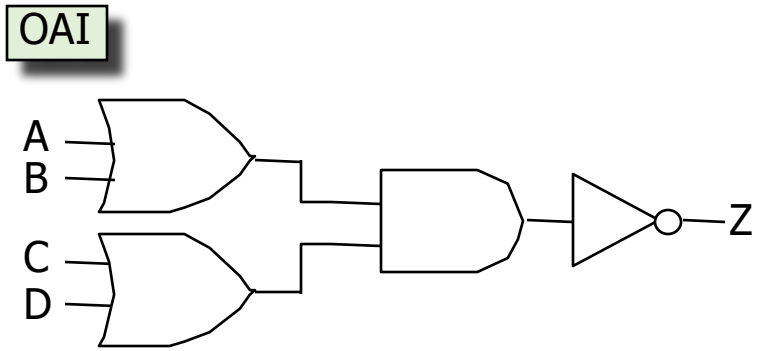


$F' = A'B' + A'C + B'C$

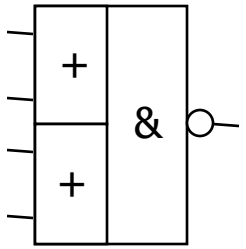


OAI Gate Implementation

- Derive product-of-sums for inverted output to find OAI implementation

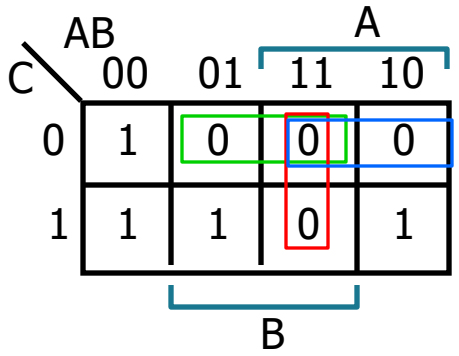


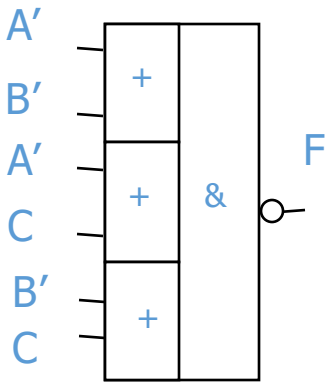
2x2 OAI Schematic Symbol



$F = \prod M(0,1,3,5)$

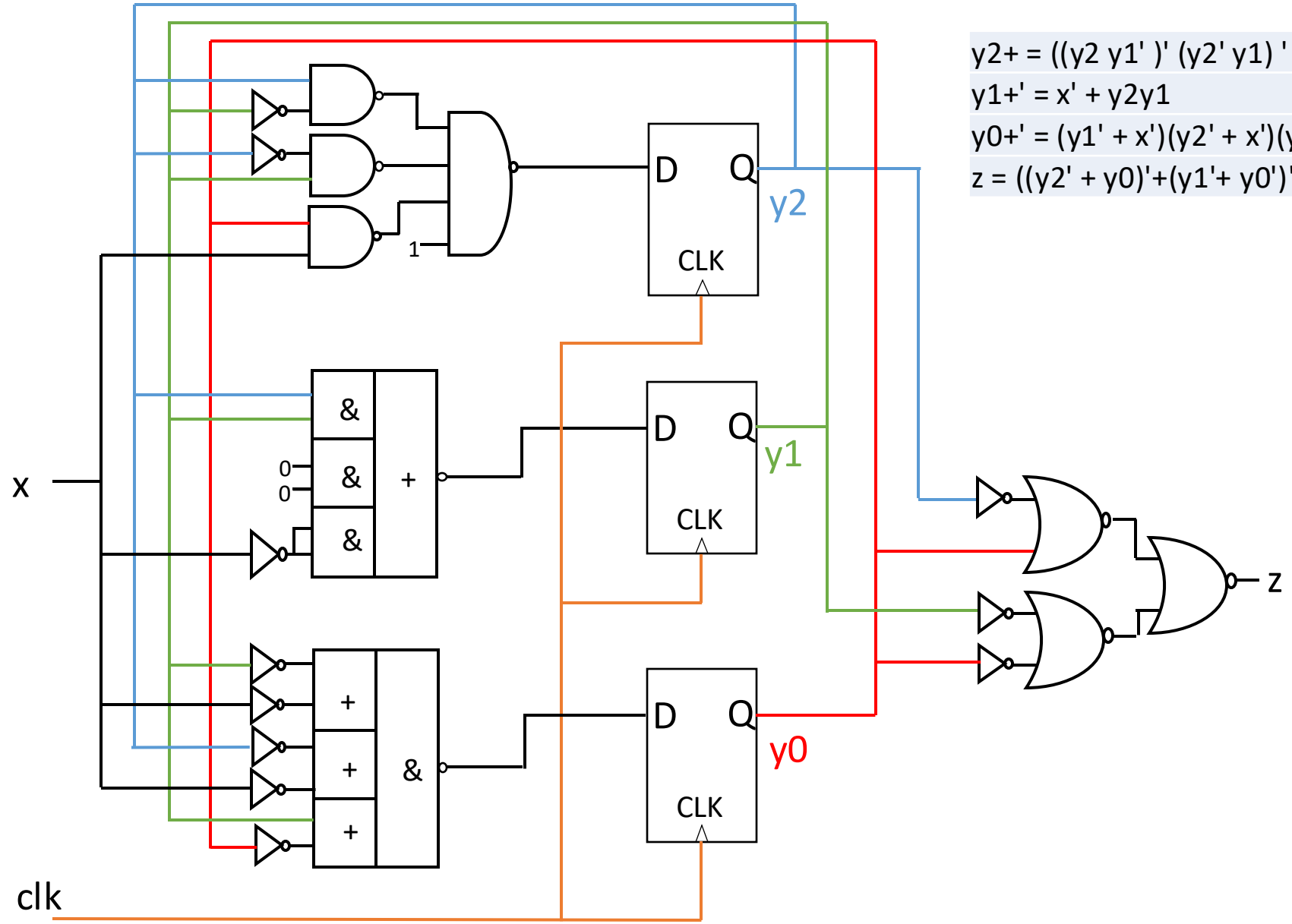
$\Rightarrow \quad F' = \prod M(2,4,6,7)$





F' = (A′ + B′) (A′ + C) (B' + C)

Circuit Diagram



y2+ = ((y2 y1')' (y2' y1) ' (y0x)')'

y1+' = x' + y2y1

y0+' = (y1' + x')(y2' + x')(y1 + y0')

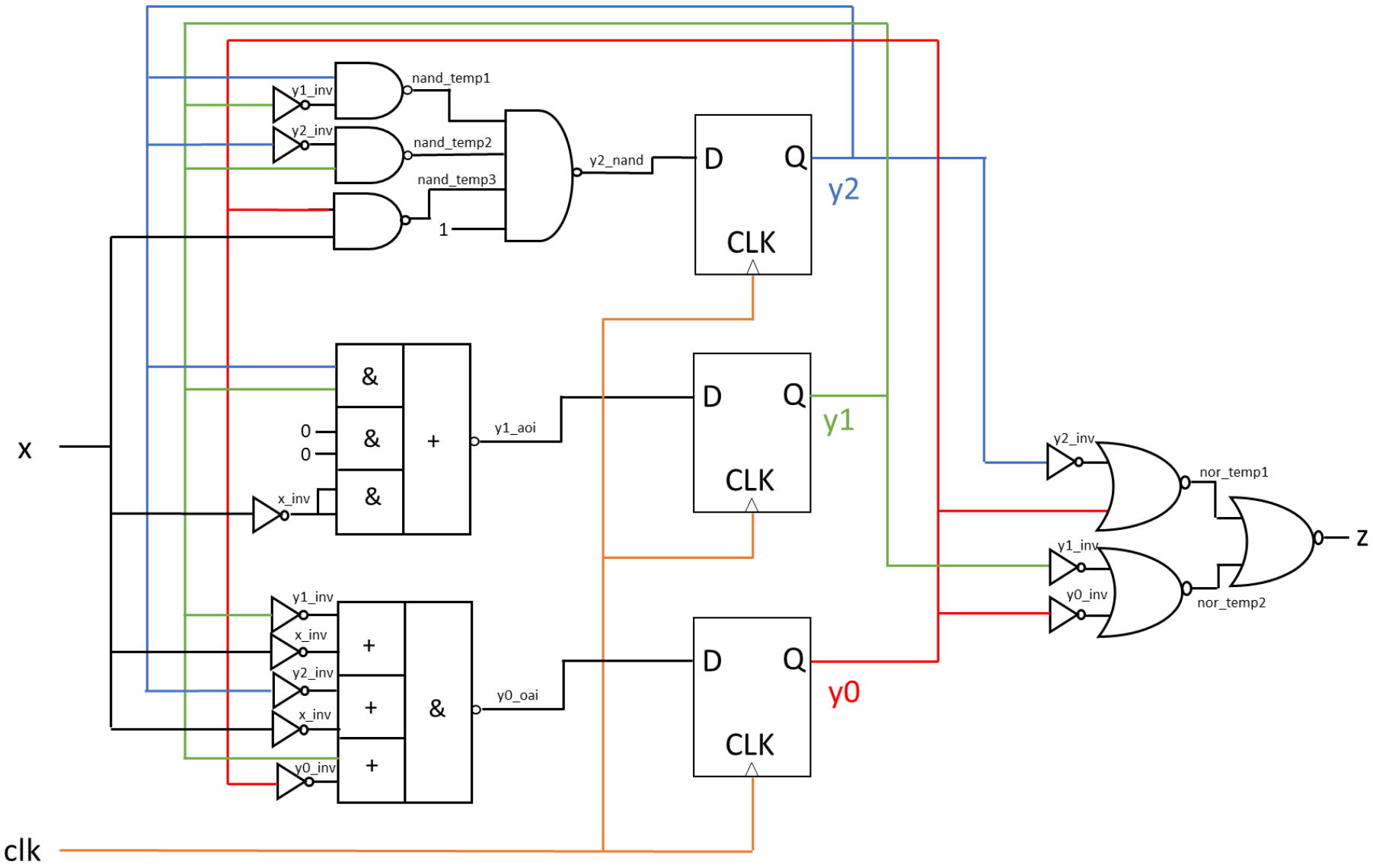
z = ((y2' + y0)'+(y1'+ y0')')'

SystemVerilog Implementation

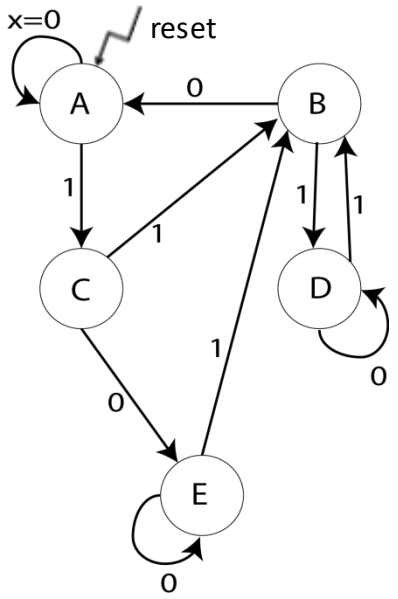
- Library with components containing AOI gates etc.
- Need to create port maps to implement the circuit

- You start with a file where you only have to add the port maps and signal declarations

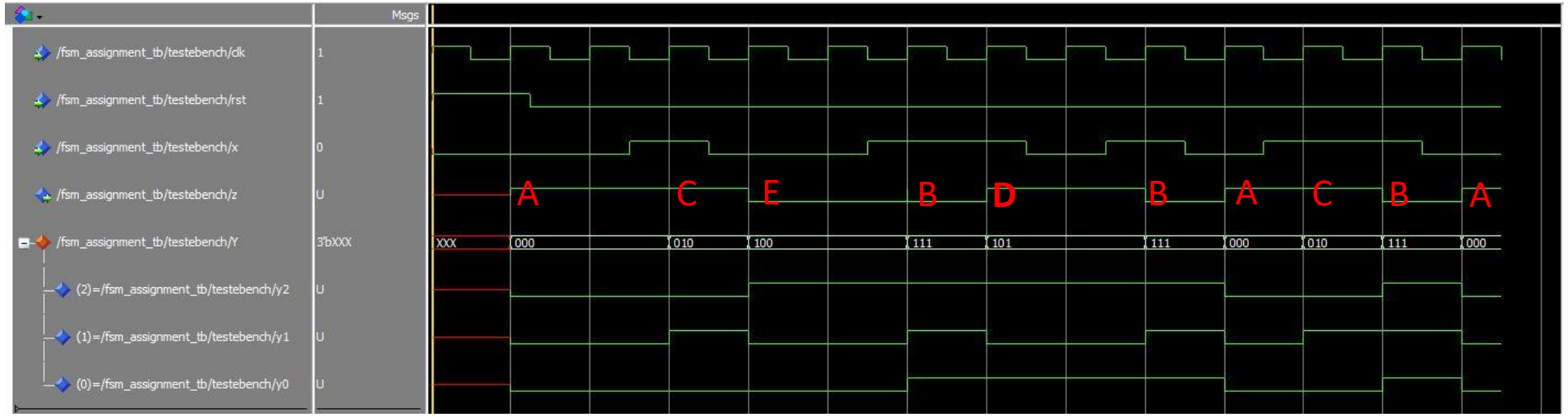
Circuit Diagram (with Signal names)



Test Bench Results



State Encoding				output
	y2	y1	y0	z
A	0	0	0	1
B	1	1	1	0
C	0	1	0	1
D	1	0	1	1
E	1	0	0	0



Summary

- Design Finite State Machines for traffic light and candy dispenser.
- Moore vs. Mealy machine
- FSM Assignment Courselab

To Do List

- Reading Material book "Digital Design":
  - Section 3.4

- Assignments for this lecture:
  - Gated Practice Assignment Lecture 7

Thank you