

CSE 140: Components and Design Techniques for Digital Systems

Lecture 8: Sequential Networks and Finite State Machines

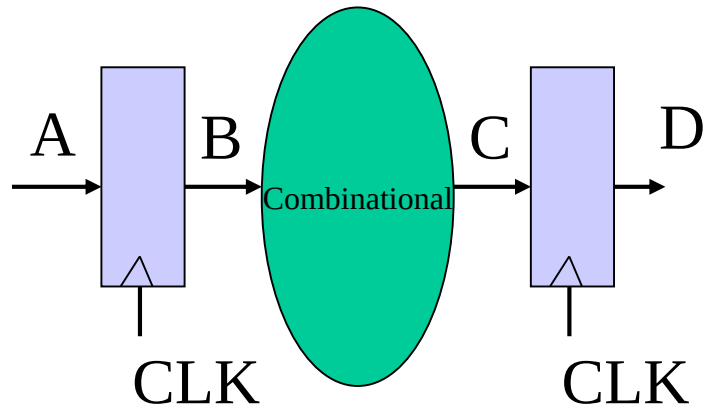
CK Cheng

Dept. of Computer Science and Engineering
University of California, San Diego

Outlines

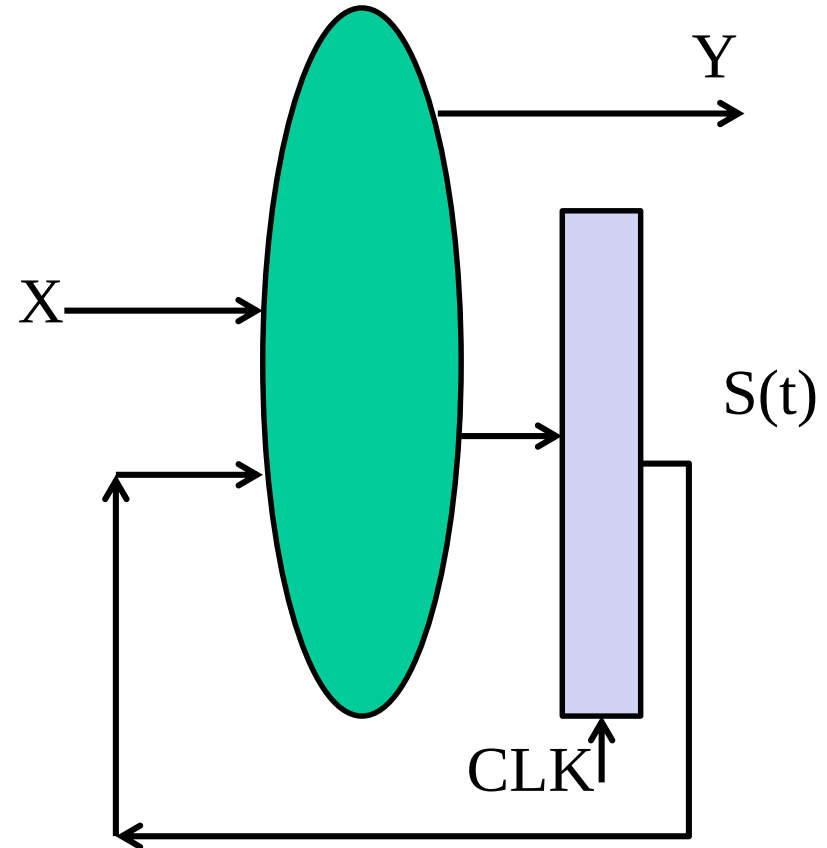
- Specification: Finite State Machine
 - State Table, State Diagram, Behavior
- Implementation
 - Excitation Table
 - Mealy and Moore Machines
 - Examples

Sequential Networks



RTL: Register-Transfer Level
Description

1. Components F-Fs
2. Specification
3. Implementation: Excitation Table



Conceptually, we can align all registers into one single column

Specification

- Combinational Logic
 - Truth Table
 - Boolean Expression
 - Logic Diagram (No feedback loops)
- Sequential Networks:
 - State Diagram, State Assignment, State Table
 - Excitation Table and Characteristic Expression
 - Logic Diagram (FFs and feedback loops)

FINITE STATE MACHINE OVERVIEW USING AN EXAMPLE

Not all sequences are created equal: A coin has two faces. A toss of the coin lands either Head or Tail with equal probability. A series of tossing creates a sequence. Given two patterns, say HHH and HTT, the pattern appears first wins.

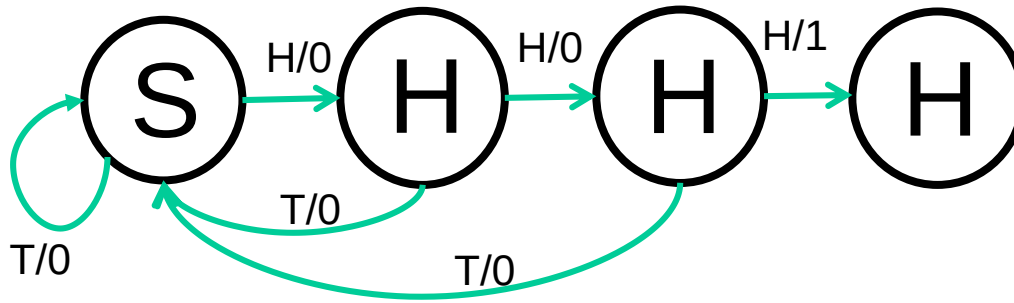
(H) (H) (H) (T) (H) (H)

- A. The two patterns have an equal chance to win
- B. HHH wins more times
- C. HTT wins more times
- D. None of the above



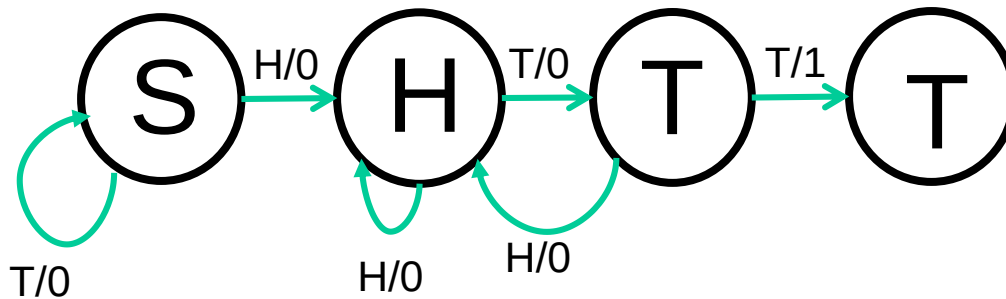
NOT ALL SEQUENCES ARE CREATED EQUAL

State Diagram of HHH



Different diagrams
yield different
expectations

State Diagram of HTT



Implementation: Design Flow

- Input Output Relation
- State Diagram (Transition of states)
 - State minimization (Reduction)
 - Finite state machine partitioning
- State Assignment (Map states into binary code)
 - Binary code, Gray encoding, One hot encoding, Coding optimization
- State Table (Truth table of states)
- Excitation Table (Truth table of FF inputs)
 - K Map, Minimal Expression
 - Logic Diagram

Implementation: Examples

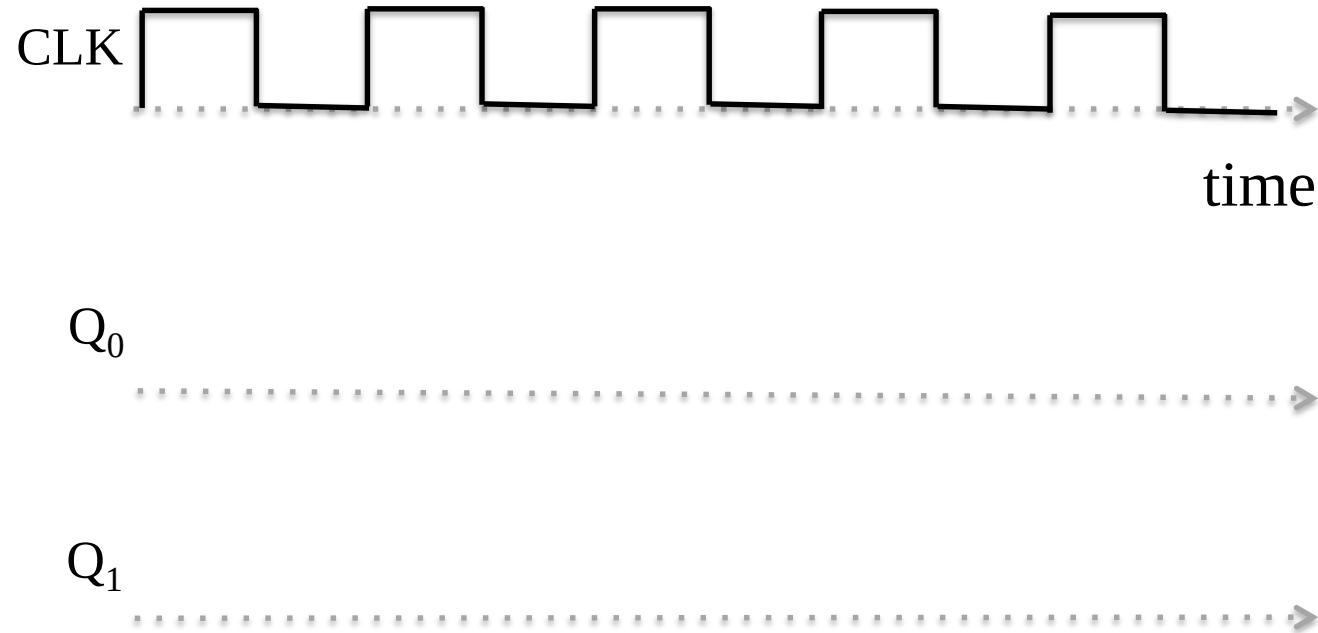
- Example 1: a counter from description, state diagram, state table, netlist, to I/O behavior.
- Two Machines: Mealy and Moore machines
- Example 2: a pattern recognizer from state diagram to netlist to demonstrate the difference of two machines.

Example: A Counter

Symbol/ Circuit



Behavior over time



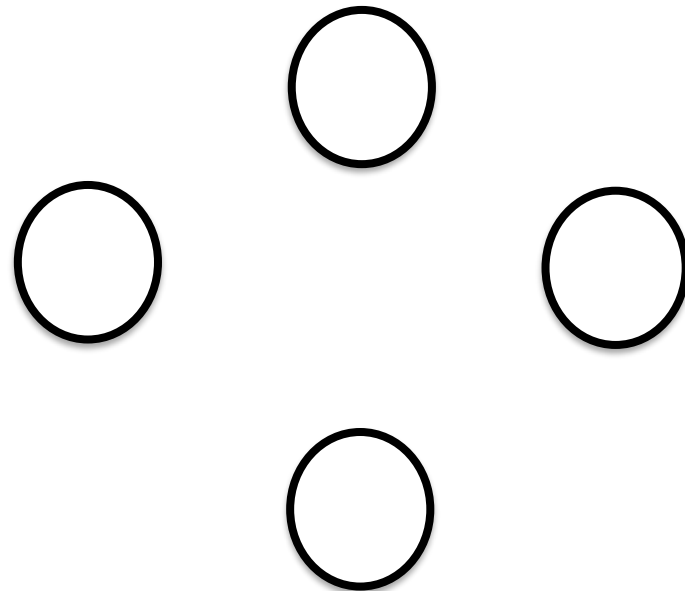
What is the expected output of the counter over time?

Finite State Machines: Describing circuit behavior over time

Symbol/ Circuit

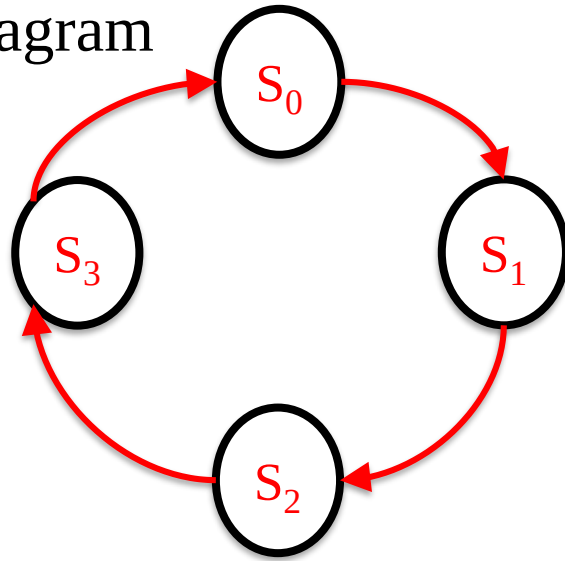


Diagram that depicts behavior over time



Implementing the 2 bit counter

State Diagram



State Table: Symbol

Current state	Next State
S_0	S_1
S_1	S_2
S_2	S_3
S_3	S_0

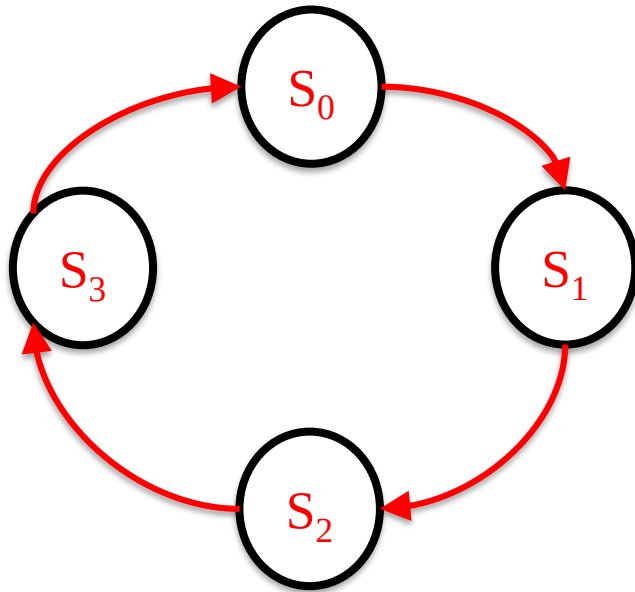
State
Assignment

State	Q_1	Q_0
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$

State Table: Binary

Implementing the 2 bit counter



State Diagram

Current state	Next State
S_0	S_1
S_1	S_2
S_2	S_3
S_3	S_0

$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

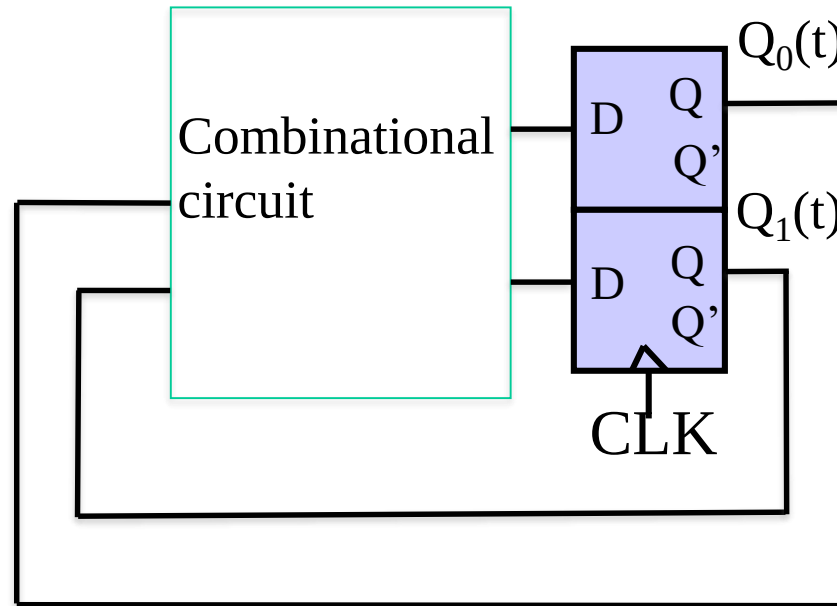
State Table

State Table

$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

$$D_0(t) = Q_0(t)'$$

$$D_1(t) = Q_0(t) Q_1(t)' + Q_0(t)' Q_1(t)$$



Circuit with 2 flip flops

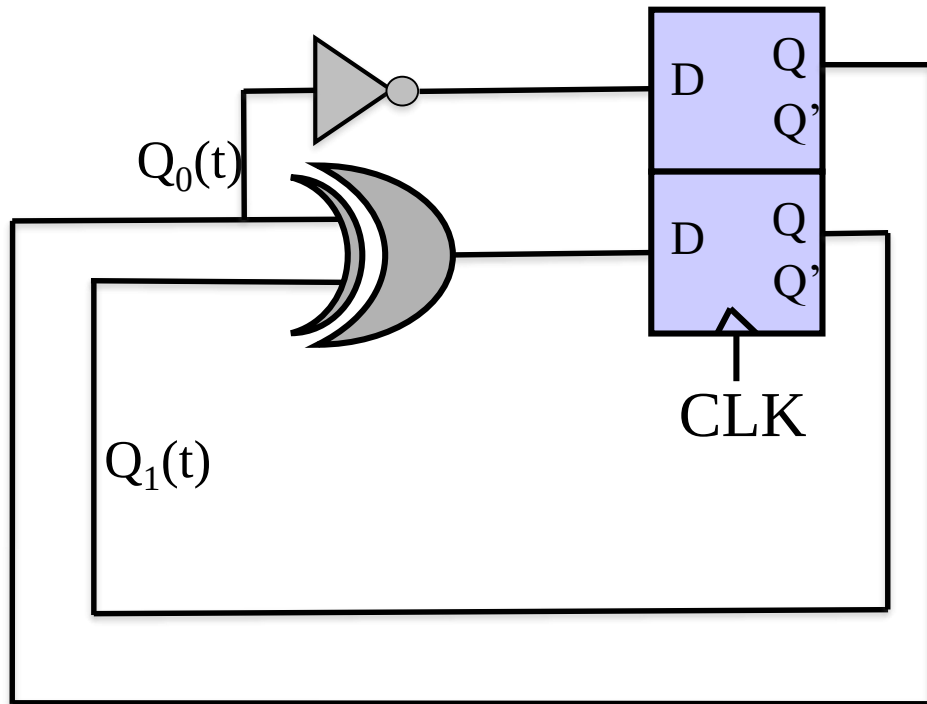
$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

State Table

Truth table→K map→Switching function

$$Q_0(t+1) = Q_0(t)'$$

$$Q_1(t+1) = Q_0(t) Q_1(t)' + Q_0(t)' Q_1(t)$$

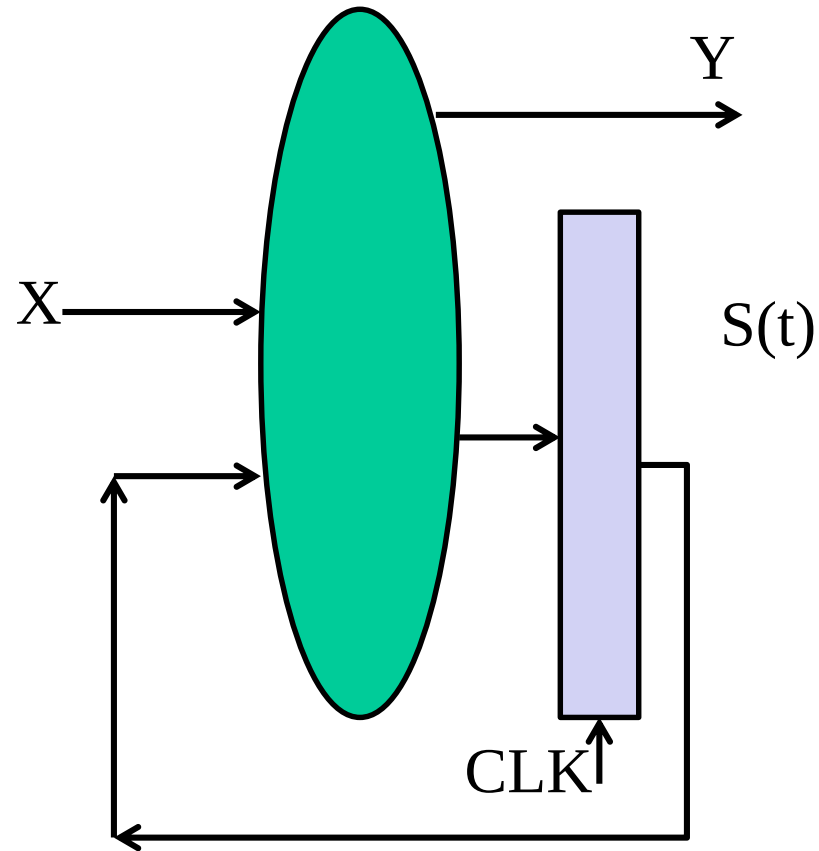


Implementation of 2-bit counter

We store the current state using D-flip flops so that:

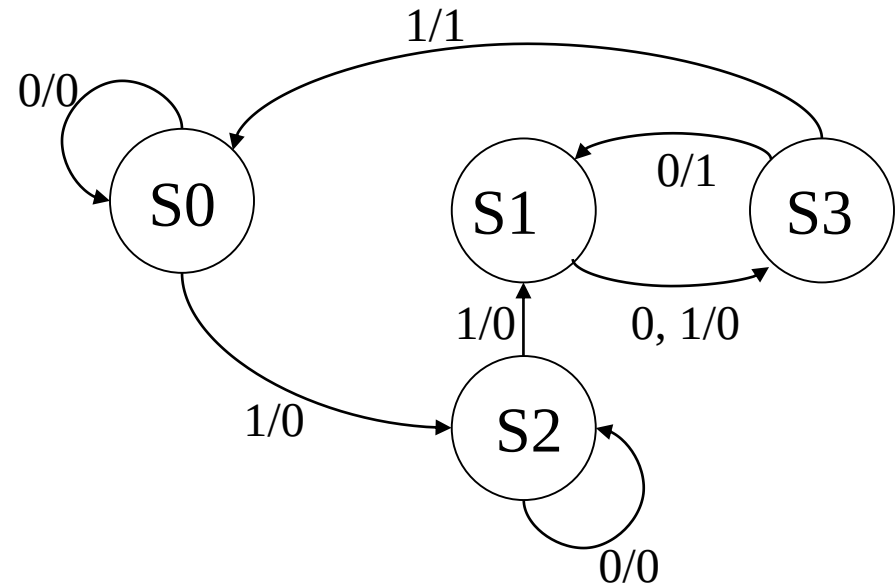
- Inputs to the combinational circuit don't change while the next output is being computed
- The transition to the next state only occurs at the rising edge of the clock

Generalized Model of Sequential Circuits



Netlist $\Leftrightarrow$ State Table $\Leftrightarrow$ State Diagram $\Leftrightarrow$ Input Output Relation

PS\Input	X=0	X=1
S0	S0,0	S2,0
S1	S3,0	S3,0
S2	S2,0	S1,0
S3	S1,1	S0,1

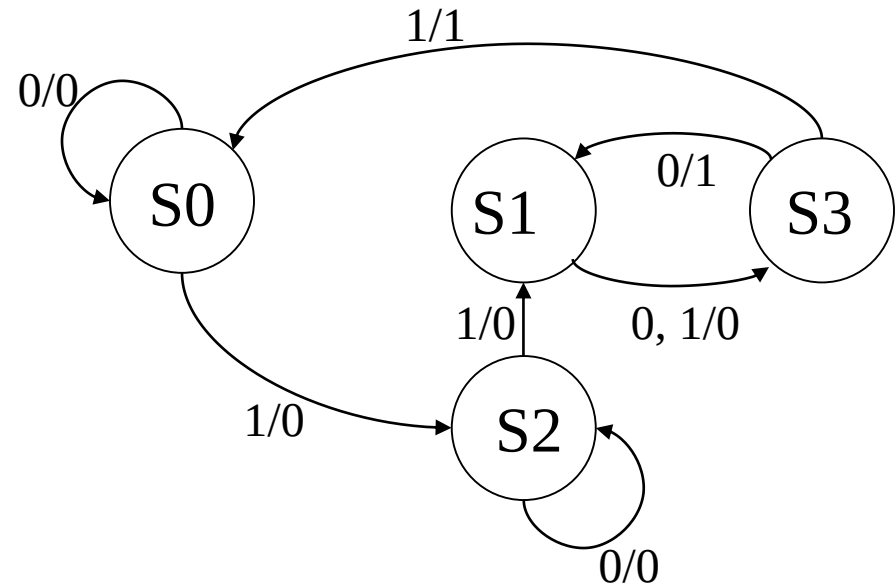


Example: Output sequence

Time	0	1	2	3	4	5
Input	0	1	1	0	1	-
State	S0					
Output						

Netlist $\Leftrightarrow$ State Table $\Leftrightarrow$ State Diagram $\Leftrightarrow$ Input Output Relation

PS\Input	X=0	X=1
S0	S0,0	S2,0
S1	S3,0	S3,0
S2	S2,0	S1,0
S3	S1,1	S0,1



Example: Output sequence

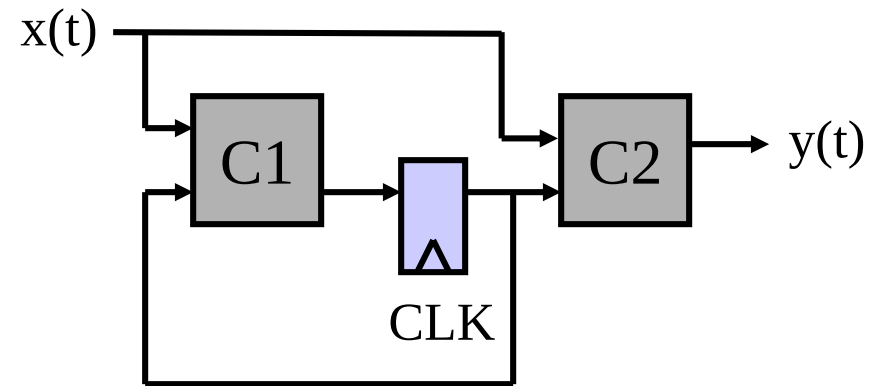
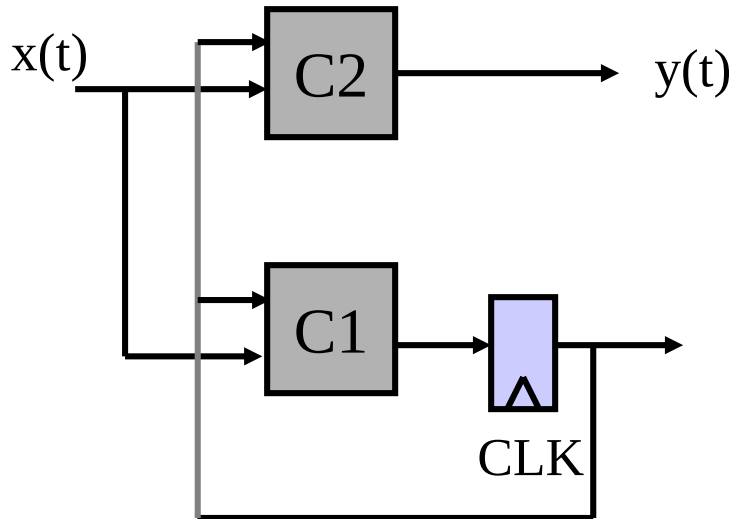
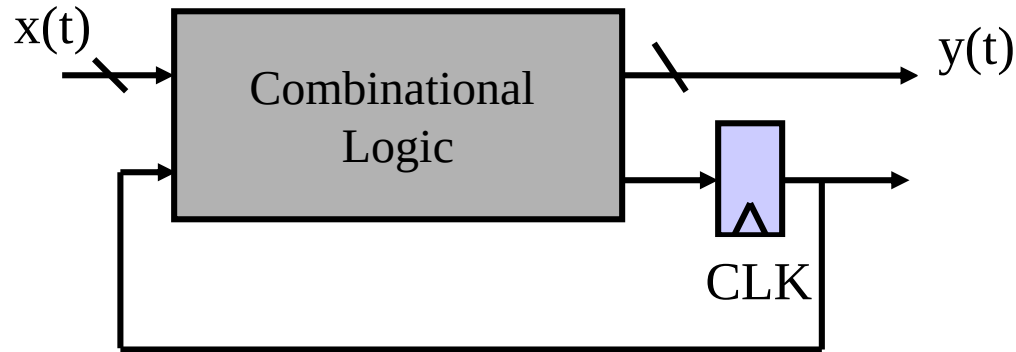
Time	0	1	2	3	4	5
Input	0	1	1	0	1	-
State	S0	S0	S2	S1	S3	S0
Output	0	0	0	0	1	0

Implementation

State Diagram $\Rightarrow$ State Table $\Rightarrow$ Logic Diagram

- Canonical Form: Mealy and Moore Machines
 - Mealy machines: General
 - Moore machines: Output is independent of current input.
- Excitation Table
 - Truth Table of the F-F Inputs
 - Boolean algebra, K-maps for combinational logic
- Examples
- Timing

Canonical Form: Mealy and Moore Machines

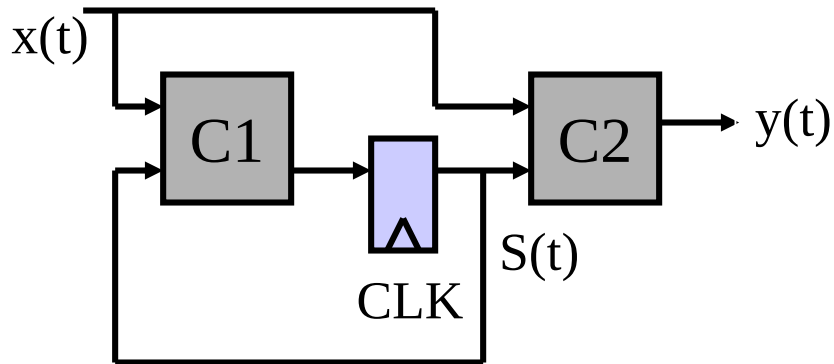


Canonical Form: Mealy and Moore Machines

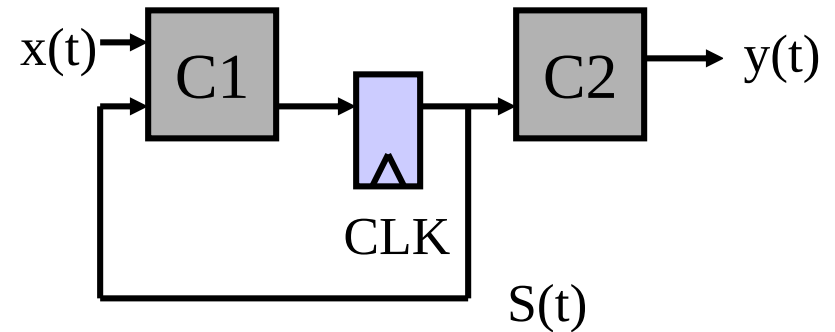
Mealy Machine: $y_i(t) = f_i(X(t), S(t))$

Moore Machine: $y_i(t) = f_i(S(t))$

$$s_i(t+1) = g_i(X(t), S(t))$$



Mealy Machine



Moore Machine

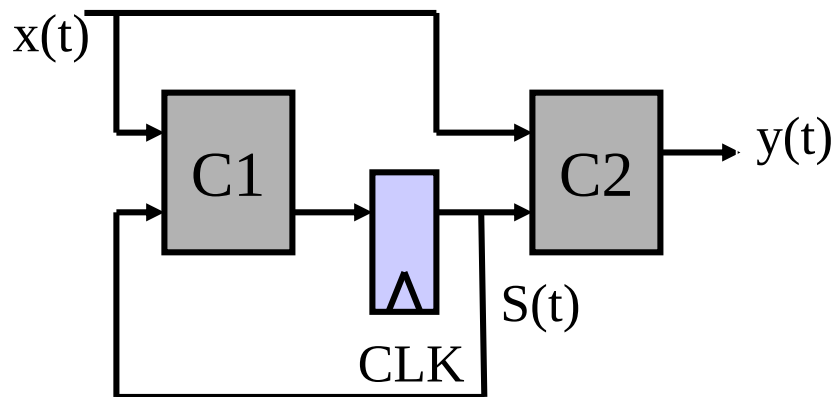
Canonical Form: Mealy and Moore Machines

Mealy Machine: $y_i(t) = f_i(X(t), S(t))$

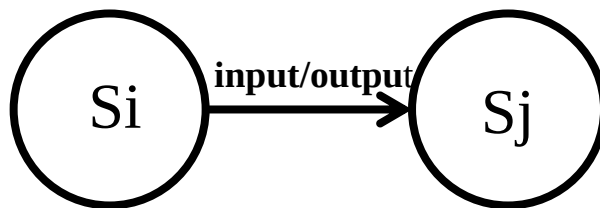
Moore Machine: $y_i(t) = f_i(S(t))$

$s_i(t+1) = g_i(X(t), S(t))$

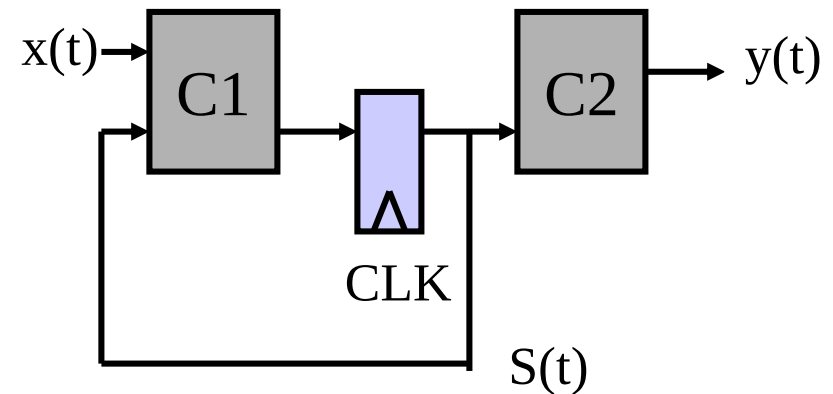
Mealy Machine



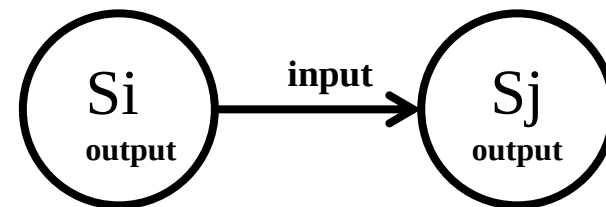
	Input
PS	NS, output



Moore Machine



	Input	
PS	NS	Output



Example: Life on Mars?

Mars rover has a binary input x . When it receives the input sequence $x(t-2, t) = 001$ from its life detection sensors, it means that it has detected life on Mars ☺ and the output $y(t) = 1$, otherwise $y(t) = 0$ (no life on Mars ☹).

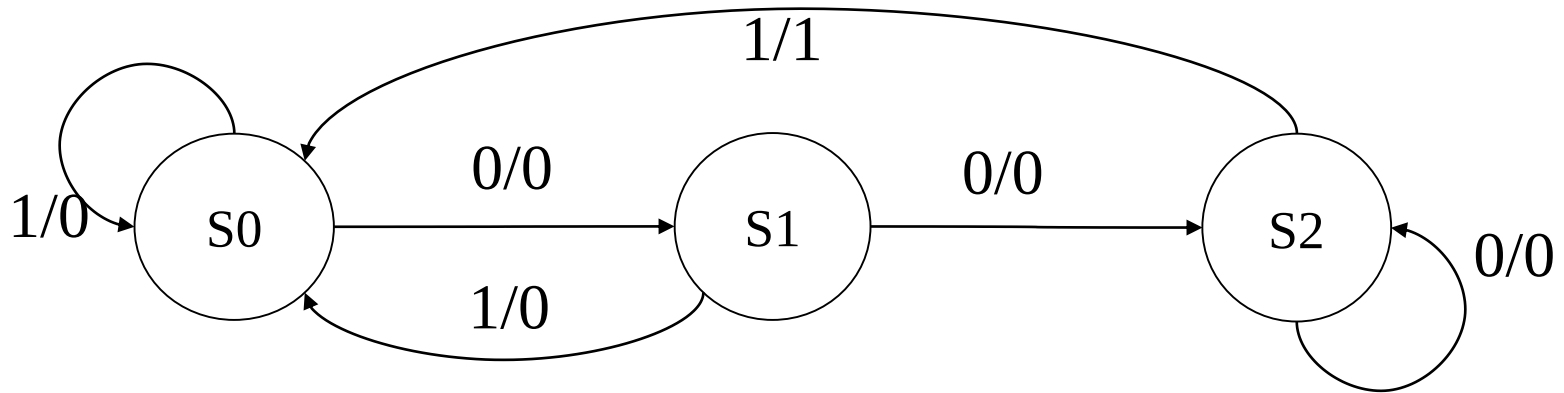
Implement the Life-on-Mars
Pattern Recognizer!



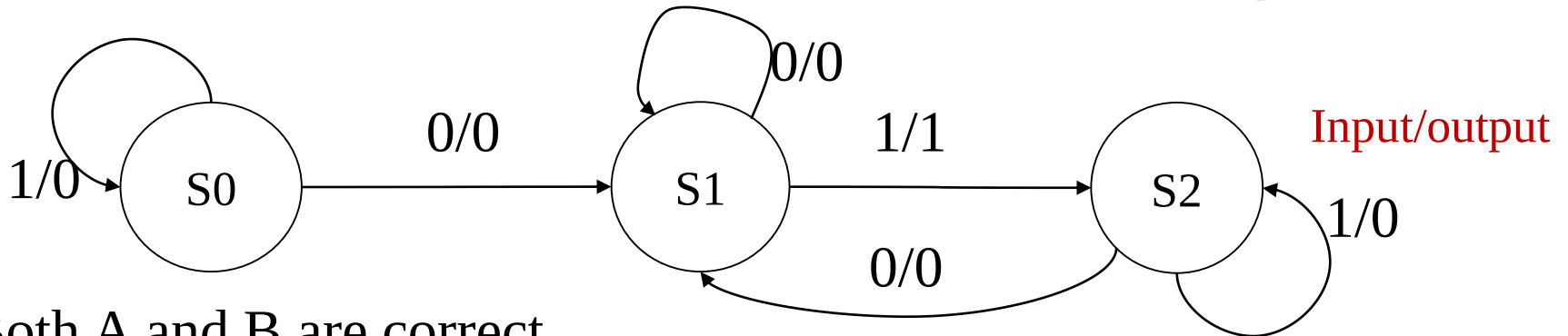
Mars Life Recognizer FSM

Which of the following diagrams is a correct Mealy solution for the 001 pattern recognizer on the Mars rover?

A.



B.

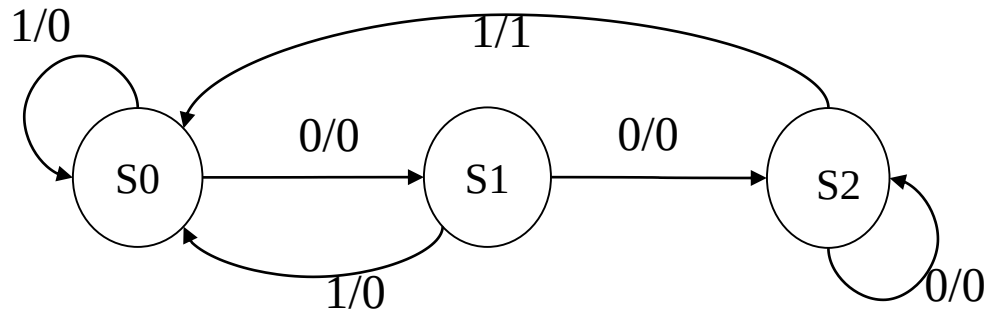


C. Both A and B are correct

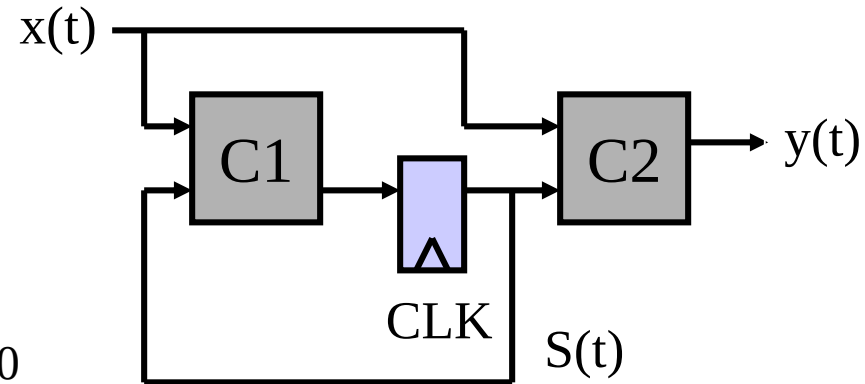
D. None of the above

Mars Life Recognizer FFs

Pattern Recognizer '001'



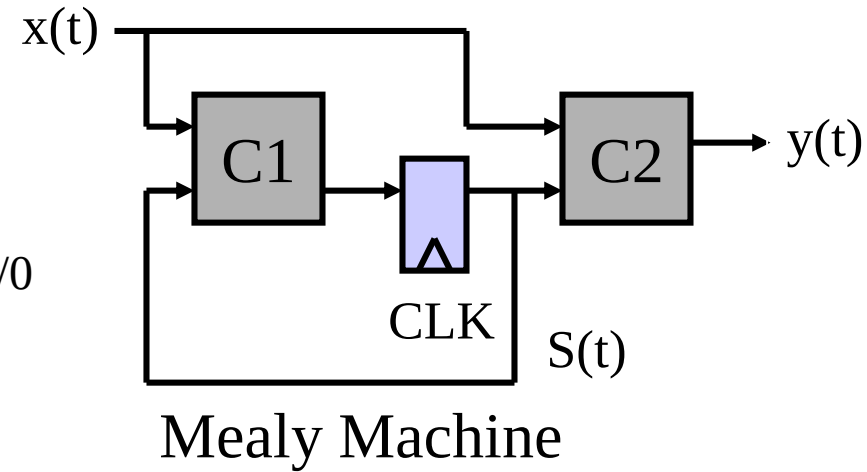
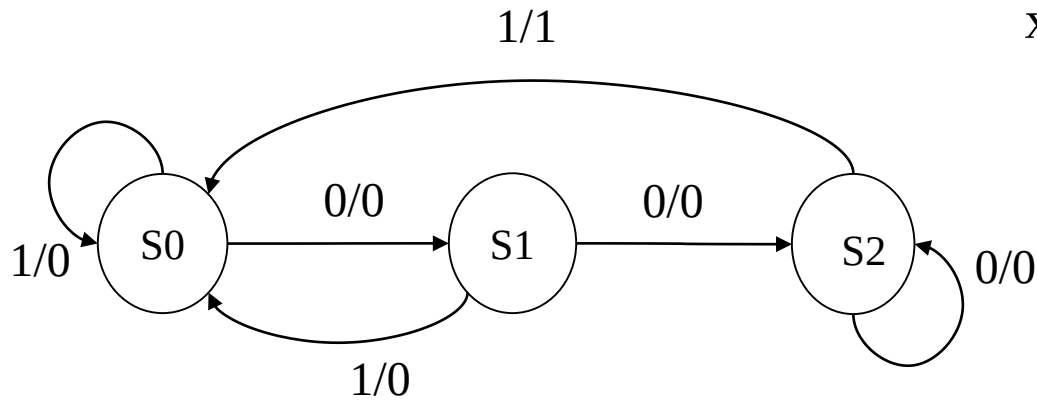
Mealy Machine



What does state table need to show to design controls of C1?

- A. next state $S(t+1)$ vs. input $x(t)$, and present state $S(t)$
- B. output $y(t)$ vs. input $x(t)$, and present state $S(t)$
- C. output $y(t)$ vs. present state $S(t)$
- D. None of the above

State Diagram => State Table with State Assignment



$S(t) \backslash x$	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1

State Assignment

S0: 00

S1: 01

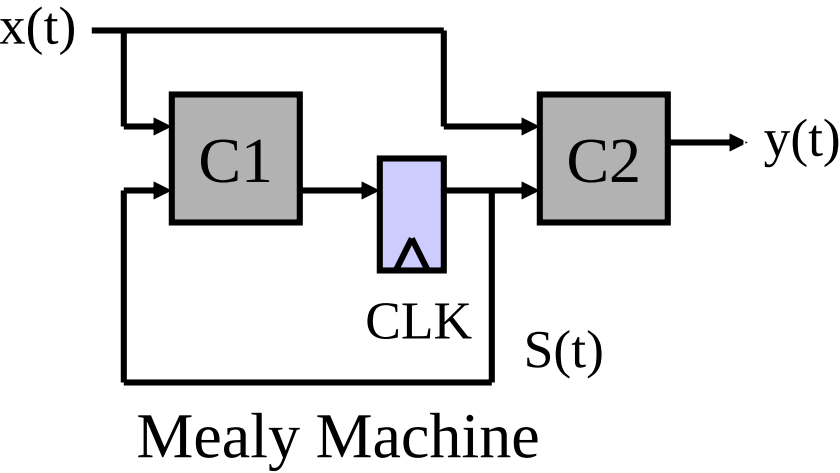
S2: 10

$S(t) \backslash x$	0	1
00	01,0	00,0
01	10,0	00,0
10	10,0	00,1

$Q_1(t+1)Q_0(t+1), y$

State Diagram => State Table => Excitation Table => Circuit

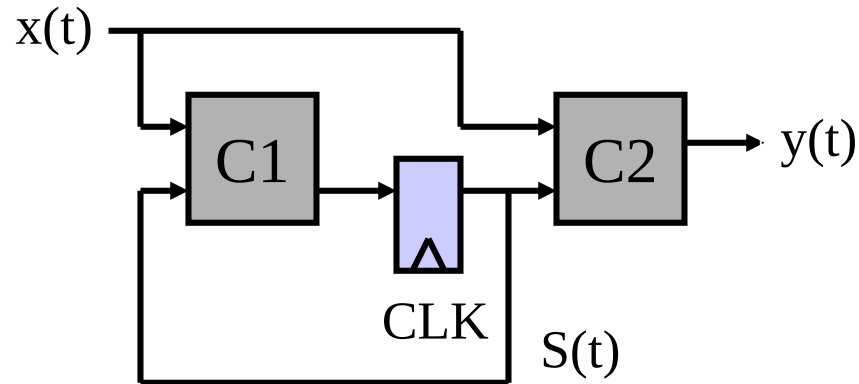
$Q_1(t) \ Q_0(t) \backslash x$	0	1
00	01,0	00,0
01	10,0	00,0
10	10,0	00,1



id	$Q_1 Q_0 x$	D_1	D_0	y
0	000	0	1	0
1	001	0	0	0
2	010	1	0	0
3	011	0	0	0
4	100	1	0	0
5	101	0	0	1
6	110			
7	111			

State Diagram => **State Table** => **Excitation Table** => Circuit

$Q_1(t) Q_0(t) \backslash x$	0	1
00	01,0	00,0
01	10,0	00,0
10	10,0	00,1



Mealy Machine

id	$Q_1 Q_0 x$	D_1	D_0	y
0	000	0	1	0
1	001	0	0	0
2	010	1	0	0
3	011	0	0	0
4	100	1	0	0
5	101	0	0	1
6	110			
7	111			

iClicker: What to fill in rows 6 and 7 of excitation table?

- A. All 0s
- B. All 1s
- C. All Don't Cares

State Diagram => State Table => Excitation Table => **Circuit**

id	Q_1Q_0x	D_1	D_0	y
0	000	0	1	0
1	001	0	0	0
2	010	1	0	0
3	011	0	0	0
4	100	1	0	0
5	101	0	0	1
6	110	X	X	X
7	111	X	X	X

$D_1(t)$:

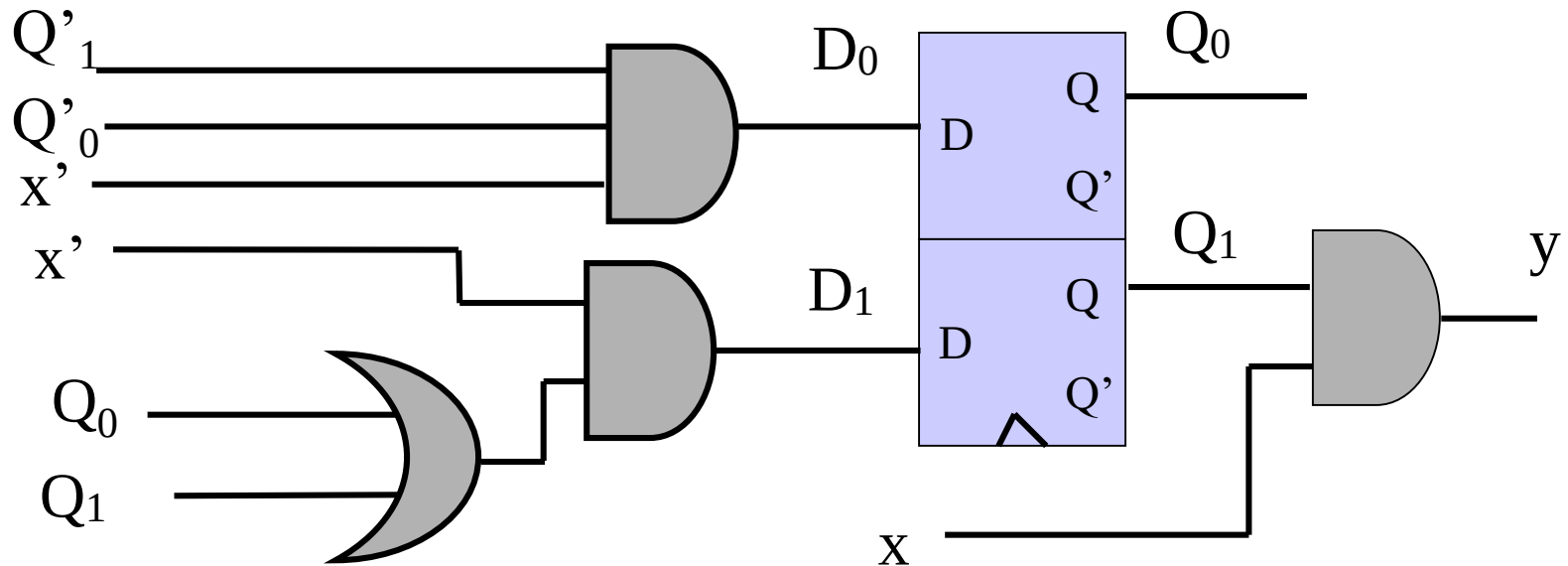
		Q_0	
$x(t)$	⁰ 0	² 1	⁶ X
	¹ 0	³ 0	⁷ X
		Q_1	
		⁴ 1	⁵ 0

$$D_1(t) = x'Q_0 + x'Q_1$$

$$D_0(t) = Q_1'Q_0'x'$$

$$y = Q_1x$$

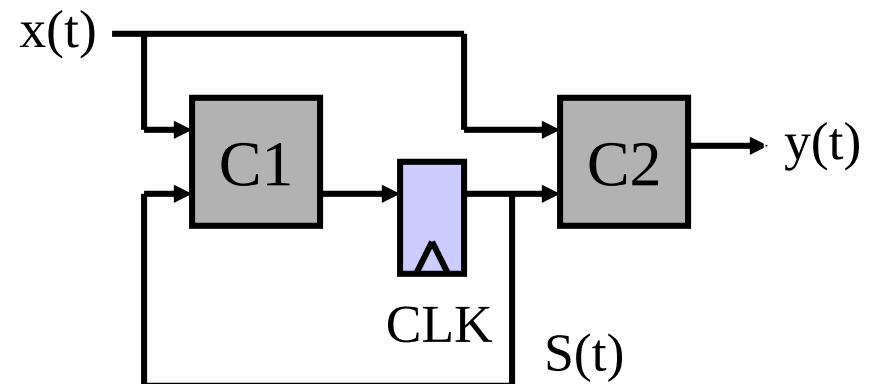
State Diagram => State Table => Excitation Table => **Circuit**



$$D_1(t) = x'Q_0 + x'Q_1$$

$$D_0(t) = Q'_1Q'_0x'$$

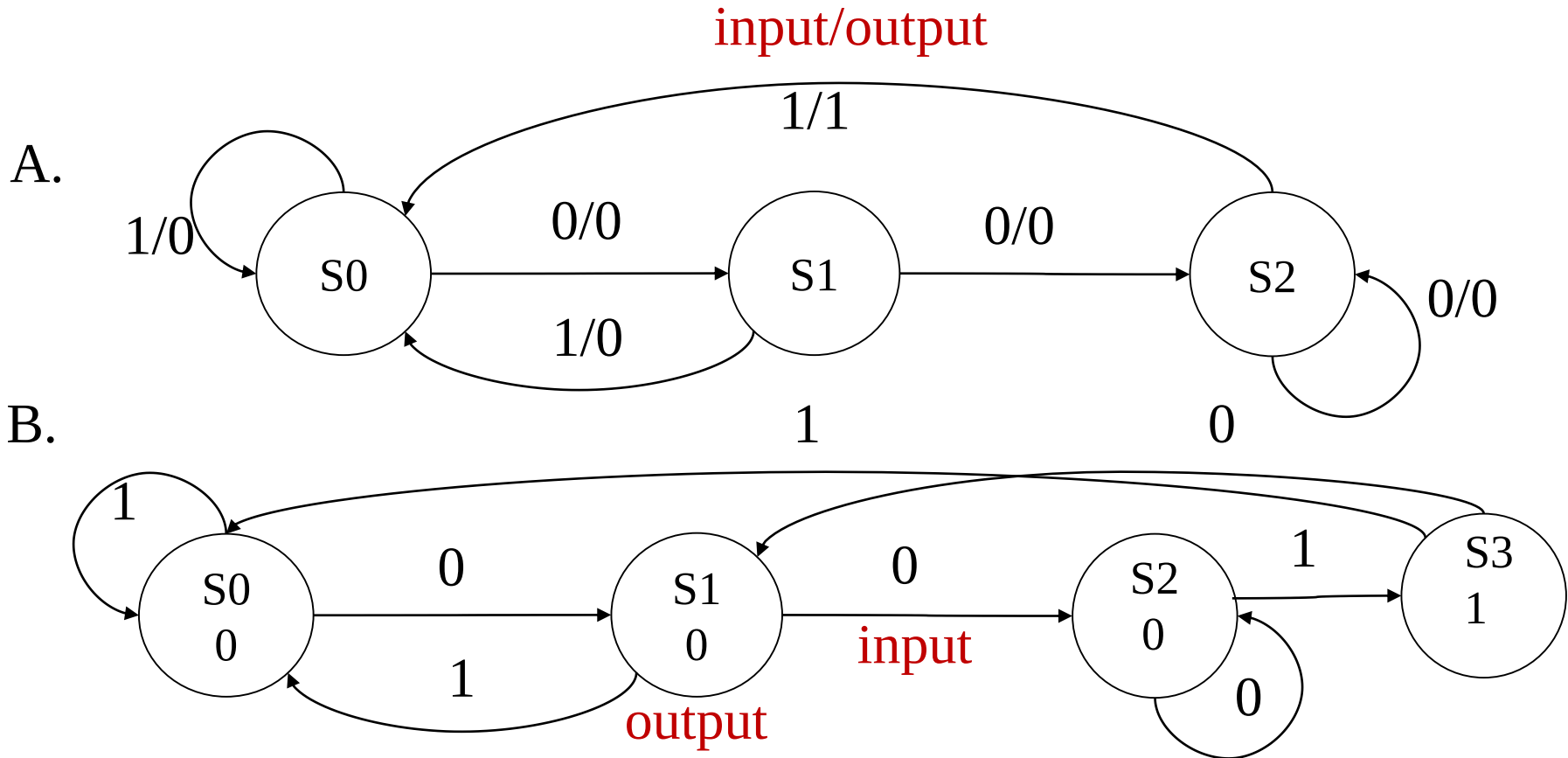
$$y = Q_1x$$



Mealy Machine

Moore FSM for the Mars Life Recognizer

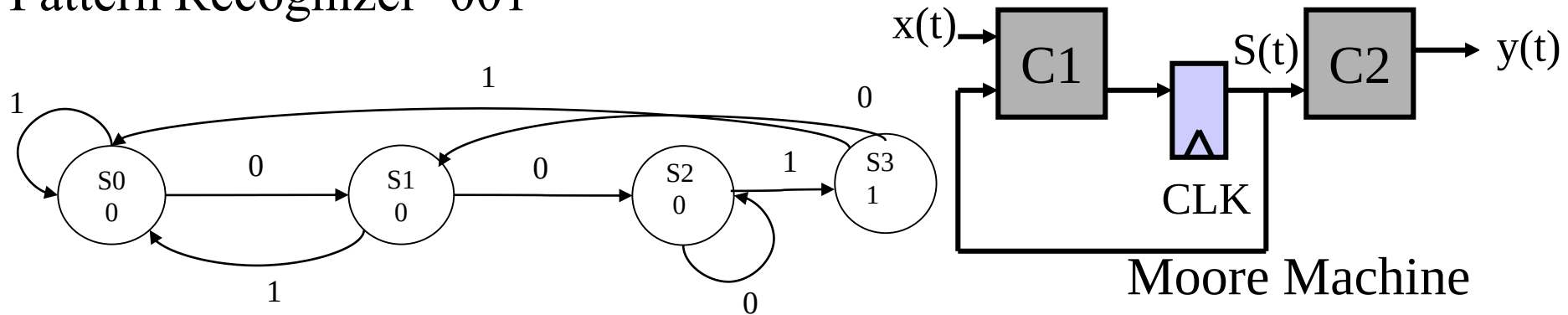
Which of the following diagrams is a correct Moore solution to the '001' pattern recognizer?



- C. Both A and B are correct
- D. None of the above

Moore Mars Life Recognizer: FF Input Specs

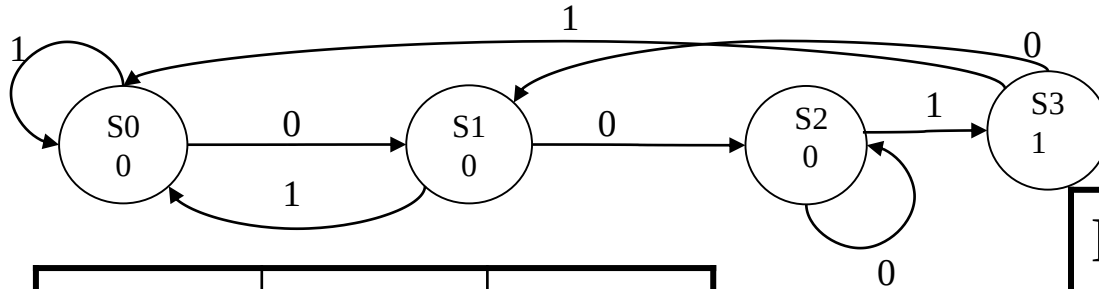
Pattern Recognizer '001'



What does state table need to show to design controls of $C2$?

- A.(current input $x(t)$, current state $S(t)$ vs. next state, $S(t+1)$)
- B.(current input, current state vs. current output $y(t)$)
- C.(current state vs. current output $y(t)$ and next state)
- D.(current state vs. current output $y(t)$)
- E.None of the above

Moore Mars Life Recognizer: State Table



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

Q ₁ Q ₀ \x	0	1
00	01,0	00,0
01	10,0	00,0
10	10,0	11,0
11	01,1	00,1

ID	Q ₁ Q ₀ x	D ₁	D ₀	y
0	000	0	1	0
1	001	0	0	0
2	010	1	0	0
3	011	0	0	0
4	100	1	0	0
5	101	1	1	0
6	110	0	1	1
7	111	0	0	1

Q₁(t+1)Q₀(t+1), y

Mars Life Recognizer: Circuit Design

id	Q_1Q_0x	D_1	D_0	y
0	000	0	1	0
1	001	0	0	0
2	010	1	0	0
3	011	0	0	0
4	100	1	0	0
5	101	1	1	0
6	110	0	1	1
7	111	0	0	1

$D_1(t)$:

$x(t)$		Q_0			
		0	1	0	1
		1	0	0	1
		Q_1			

$D_0(t)$:

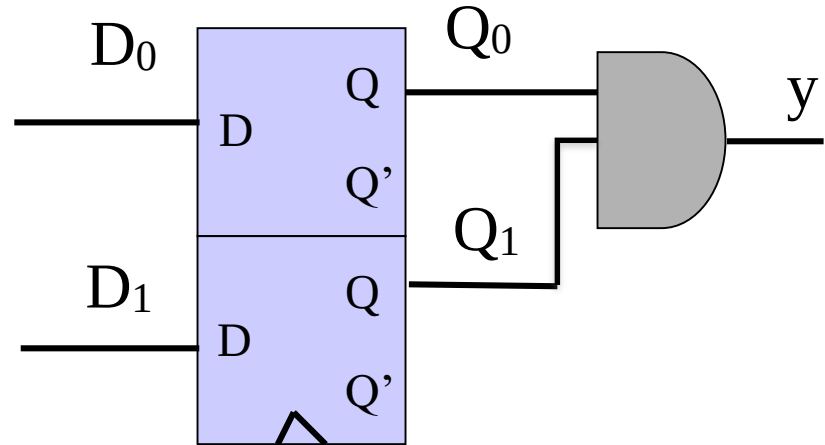
$x(t)$		Q_0			
		1	0	1	0
		0	0	0	1
		Q_1			

$y(t)$:

$x(t)$		Q_0			
		0	0	1	0
		0	0	1	0
		Q_1			

Mars Life Recognizer Circuit Implementation

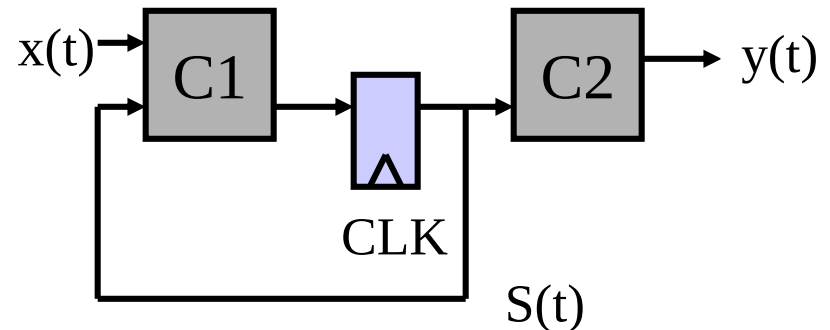
State Diagram => State Table => Excitation Table => **Circuit**



$$D_1(t) = Q_1(t)Q_0(t)' + Q_1(t)'Q_0(t)x(t)$$

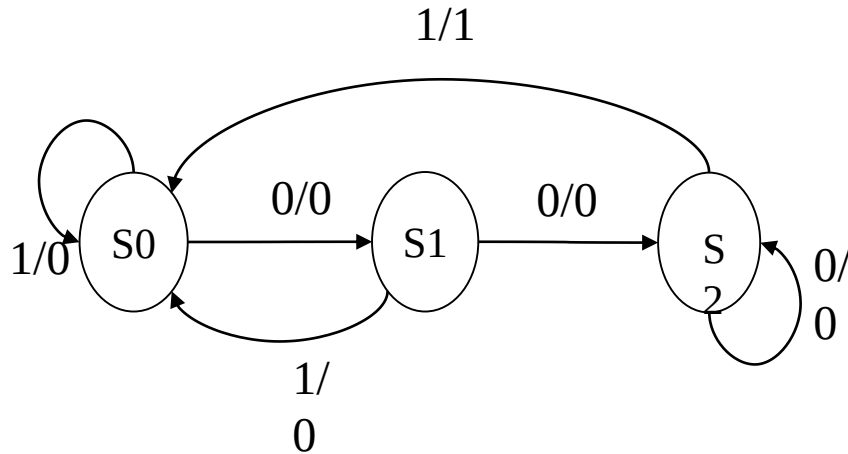
$$D_0(t) = Q_1(t)'Q_0(t)'x(t)' + Q_1(t)Q_0(t)x(t)' + Q_1(t)Q_0(t)'x(t)$$

$$y(t) = Q_1(t)Q_0(t)$$

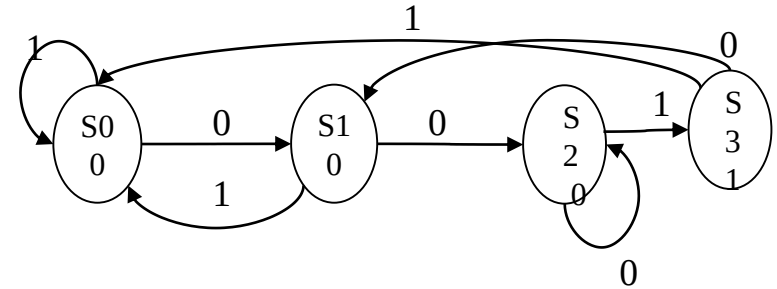


Moore Machine

Conversion from Mealy to Moore Machine



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

Conversion from Mealy to Moore Machine

S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1

S(t)\x	0	1	y
S0	S1	S0	0
S1	S2	S0	0
S2	S2	S3	0
S3			

Algorithm

1. Identify distinct (NS, y) pair
2. Replace each distinct (NS, y) pair with distinct new states
3. Insert rows of present state = new states

Conversion from Mealy to Moore Machine

Mealy

S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1

S(t)\x	0	1	y
S0	S1	S0	0
S1	S2	S0	0
S2	S2	S3	0
S3			

Moore

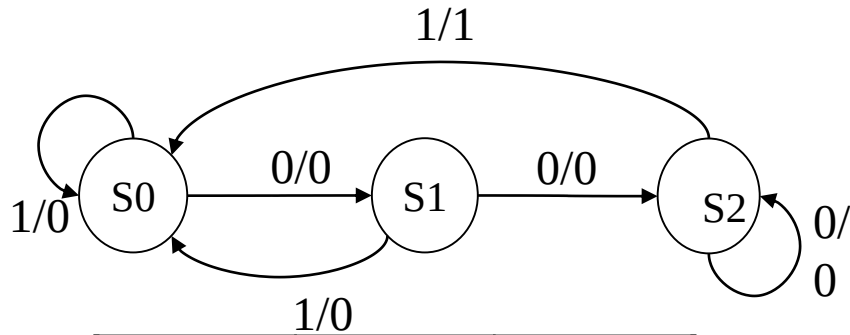
1. Find distinct NS, y
2. Add new states to represent distinct NS, y

iClicker

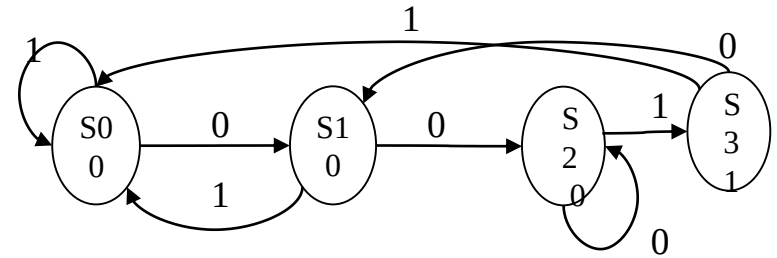
For the above Moore machine, what are the next states with respect to present state S3?

- A. S2, S3, 1
- B. S2, S0, 1
- C. S1, S0, 1
- D. S1, S0, 0
- E. None of the above.

Conversion from Mealy to Moore Machine



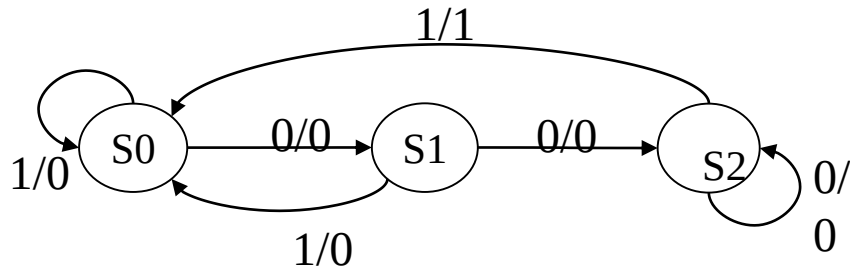
S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1



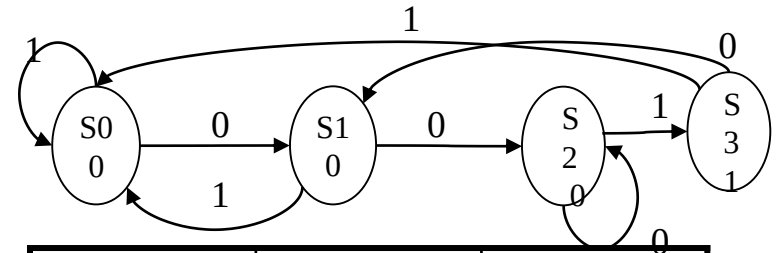
S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

Time	0	1	2	3	4	5	6	7	8
x	0	1	0	0	1	1	0	0	1
S _{mealy}	S0	S1	S0	S1	S2	S0	S0	S1	S2
y _{mealy}									
S _{moore}	S0								
y _{moore}									

Conversion from Mealy to Moore Machine



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1



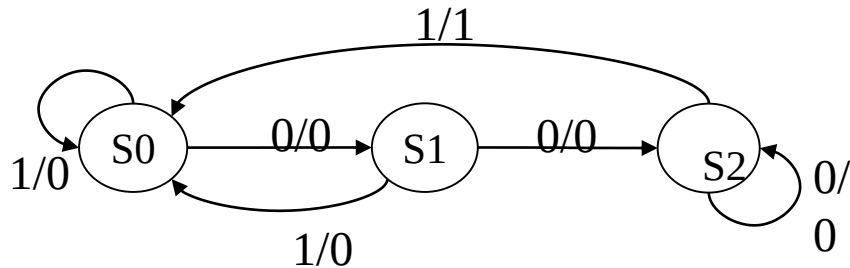
S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

Time	0	1	2	3	4	5	6	7	8
x	0	1	0	0	1	1	0	0	1
S _{mealy}	S0	S1	S0	S1	S2	S0	S0	S1	S2
y _{mealy}									
S _{moore}	S0								
y _{moore}									

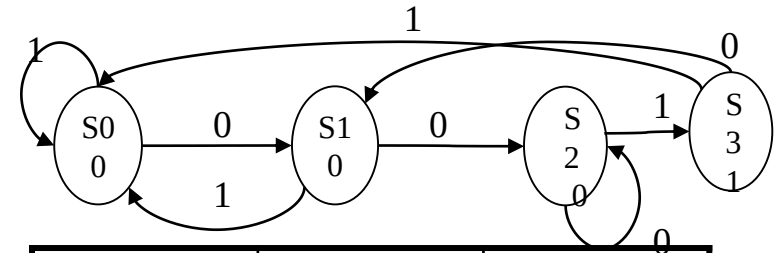
iClicker S_{moore}[0-5]

- A. S0,S1,S0,S1,S2,S3
- B. S0,S1,S0,S1,S2,S0
- C. S3,S1,S0,S1,S2,S3
- D. S3,S1,S0,S1,S2,S0
- E. None of the above

Conversion from Mealy to Moore Machine



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1



S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

Time	0	1	2	3	4	5	6	7	8
x	0	1	0	0	1	1	0	0	1
S _{mealy}	S0	S1	S0	S1	S2	S0	S0	S1	S2
y _{mealy}									
S _{moore}	S0								
y _{moore}									

iClicker y_{moore}[0-5]

A. 0,0,0,0,1,0

B. 0,0,0,0,0,1

C. 0,1,0,0,0,0

D. 0,0,0,0,0,0,

E. None of the above

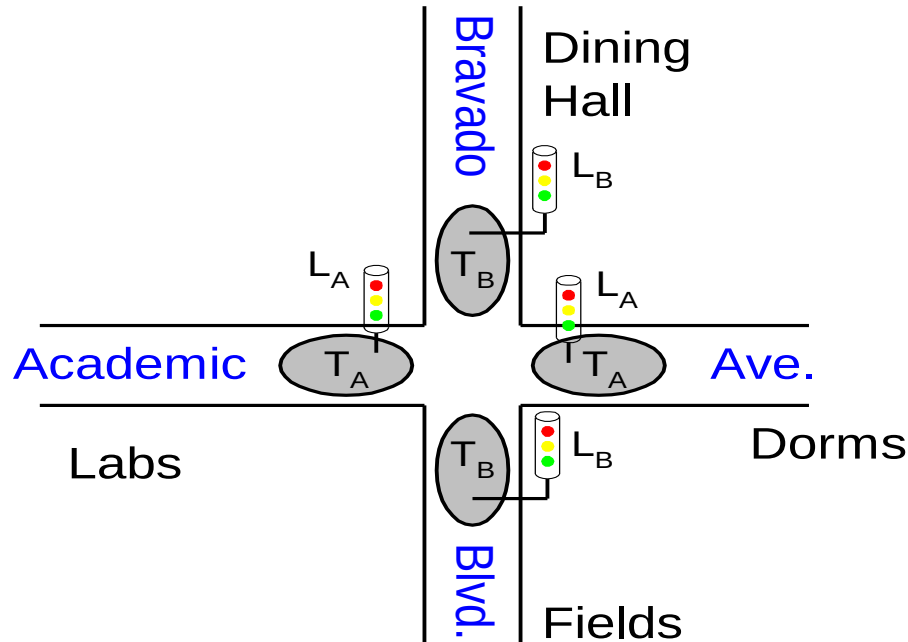
Conversion from Mealy to Moore Machine

Algorithm

1. Identify distinct (NS, y) pair
2. Replace each distinct (NS, y) pair with distinct **new states**
3. Insert rows of present state = **new states**
4. Append each present state with its output y

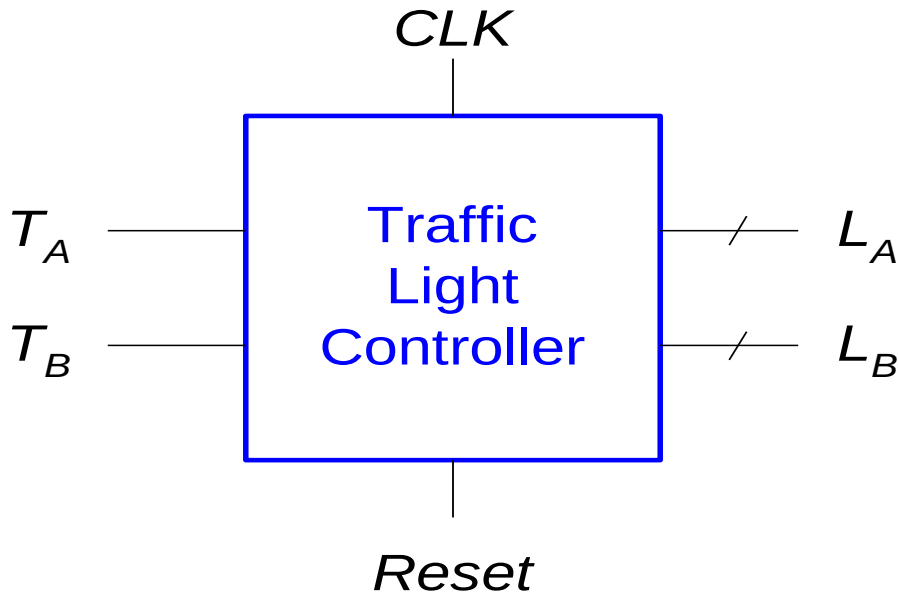
Finite State Machine Example

- Traffic light controller
 - Traffic sensors: T_A , T_B (TRUE when there's traffic)
 - Lights: L_A , L_B



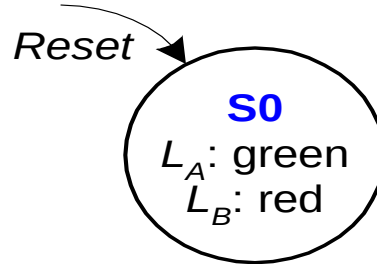
FSM Black Box

- Inputs: CLK , $Reset$, T_A , T_B
- Outputs: L_A , L_B



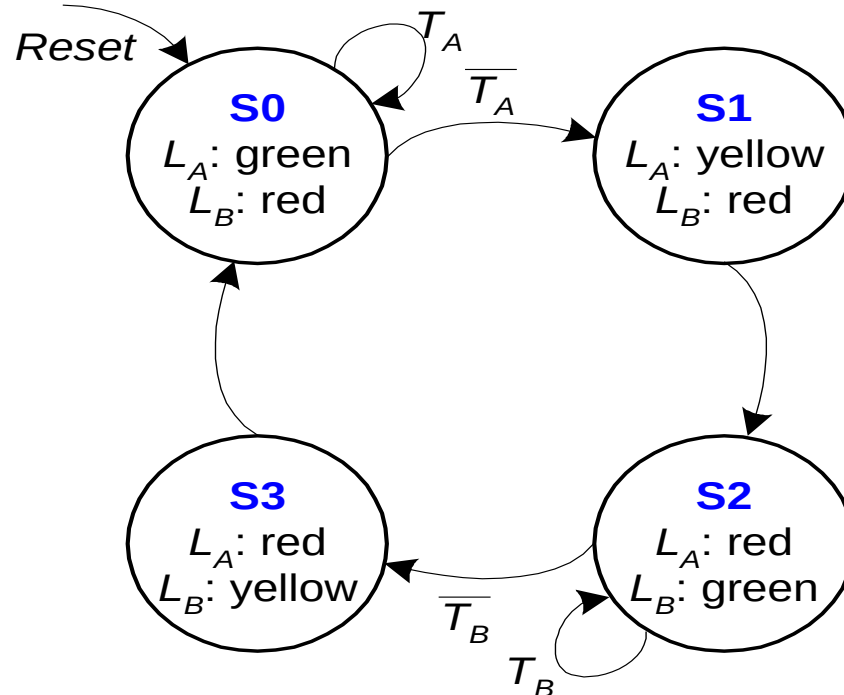
FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



FSM State Transition Diagram

- Moore FSM: outputs labeled in each state
- States: Circles
- Transitions: Arcs



FSM State Transition Table

PS	Inputs		NS
	T_A	T_B	
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

State Transition Table

PS		Inputs		NS	
$Q_1(t)$	$Q_0(t)$	T_A	T_B	$Q_1(t+1)$	$Q_0(t+1)$
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

State	Encoding
S0	00
S1	01
S2	10
S3	11

$$Q_1(t+1) = Q_1(t) \oplus Q_0(t)$$

$$Q_0(t+1) = Q_1'(t)Q_0'(t)T_A' + Q_1(t)Q_0'(t)T_B'$$

FSM Output Table

PS		Outputs			
Q_1	Q_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

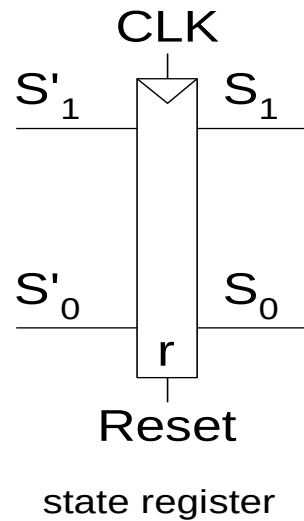
$$L_{A1} = Q_1$$

$$L_{A0} = Q_1' Q_0$$

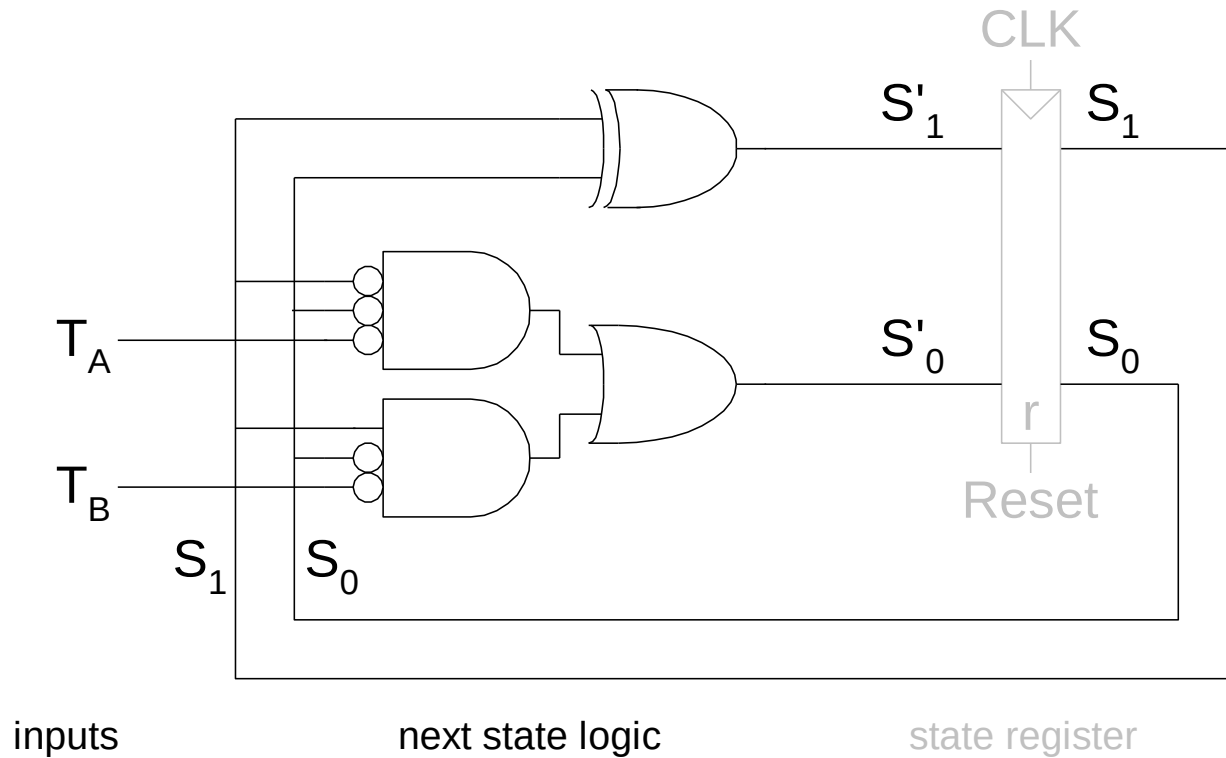
$$L_{B1} = Q_1'$$

$$L_{B0} = Q_1 Q_0$$

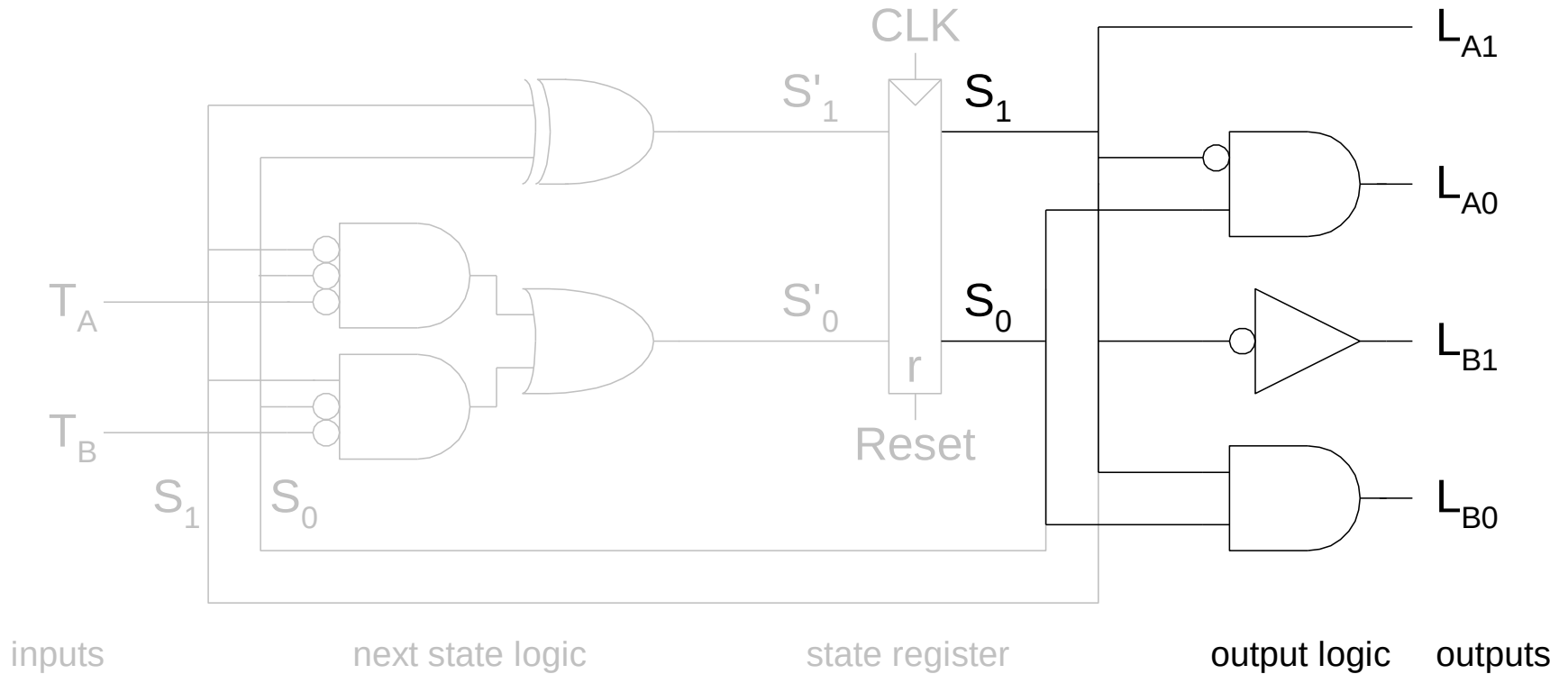
FSM Schematic: State Register



Logic Diagram



FSM Schematic: Output Logic



Summary: Implementation

- Set up canonical form
 - Mealy or Moore machine
- Identify the next states
 - state diagram $\Leftrightarrow$ state table
 - state assignment
- Derive excitation table
 - Inputs of flip flops
- Design the combinational logic
 - don't care set utilization