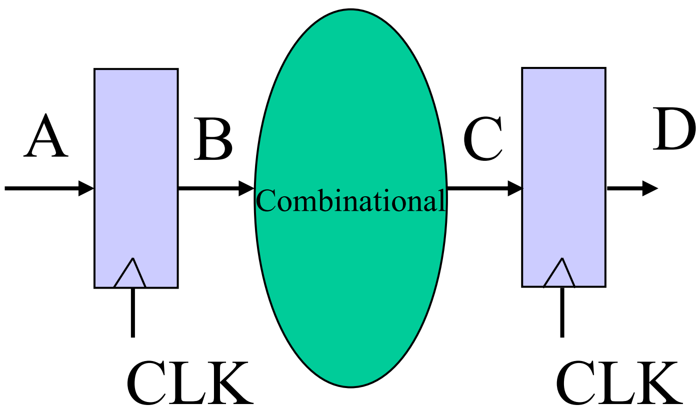
Lecture 2:
Sequential Networks and Finite State Machines

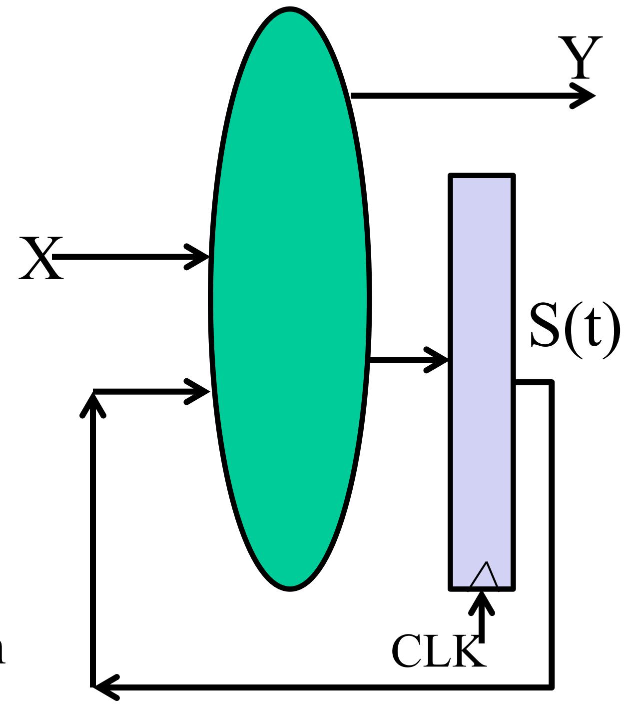
John Eldon
University of California, San Diego

Outline

- 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. Gray: DFFs = "registers"
2. Specification: What does it do?
3. Implementation: Excitation or Transition Table

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)

Implementation: Design Flow

- Input-Output Relation
- State Diagram (Transition of states)
- State Assignment (Map states into binary code)
- State Table (Truth table of states)
- Excitation Table (Truth table of FF inputs)
- Boolean Expression
- Logic Diagram

Implementation: Design Flow

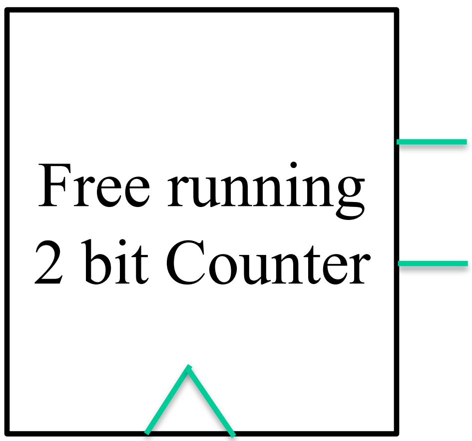
- Input Output Relation
- State Diagram (Transition of states)
  - State minimization (Reduction)
  - Finite state machine partitioning
- State Assignment (Map states into binary code)
  - Binary code, Gray encoding, One hot encoding, Coding optimization
- State Table (Truth table of states)
- Excitation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

Implementation: Examples

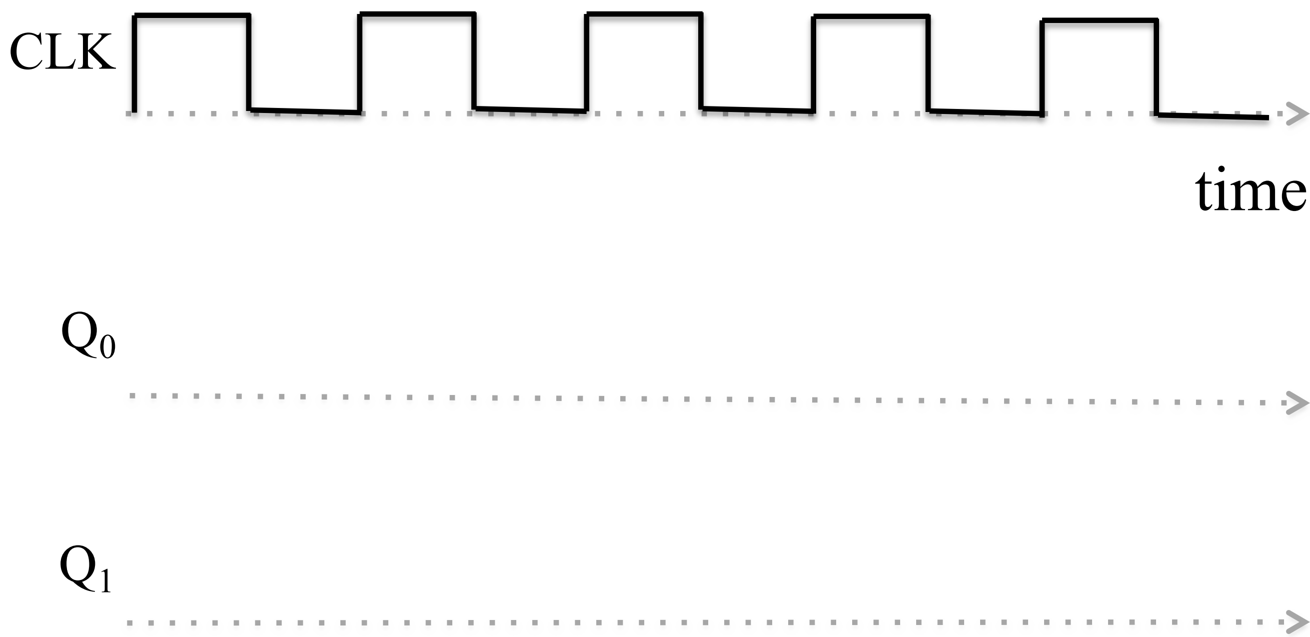
- Example 1: a circuit with D Flip Flops
- Example 2: a circuit with other Flip Flops
- Example 3: analysis of a sequential machine

State: What is it? Why do we need it?

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

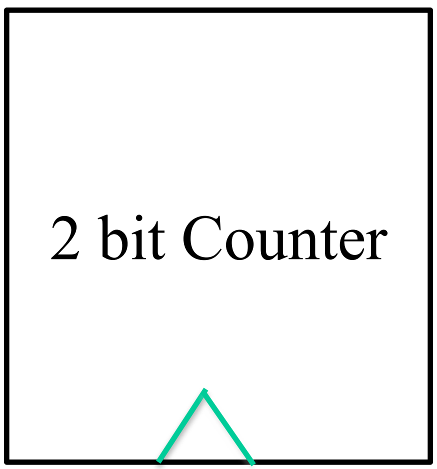
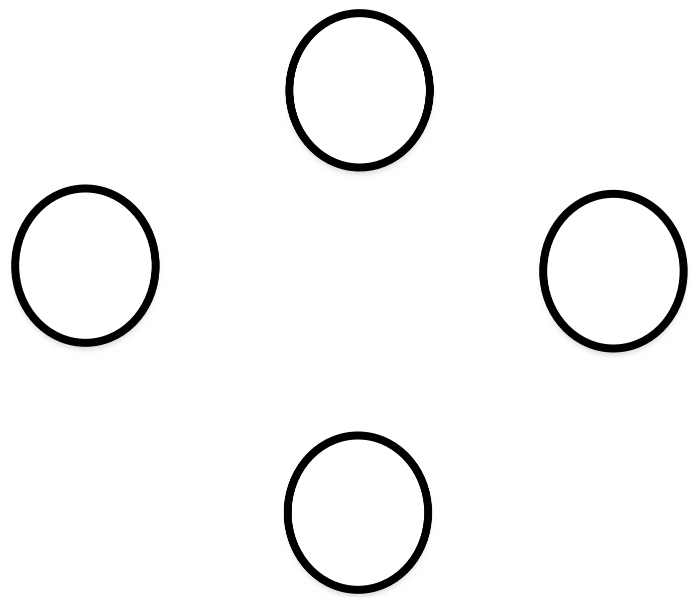
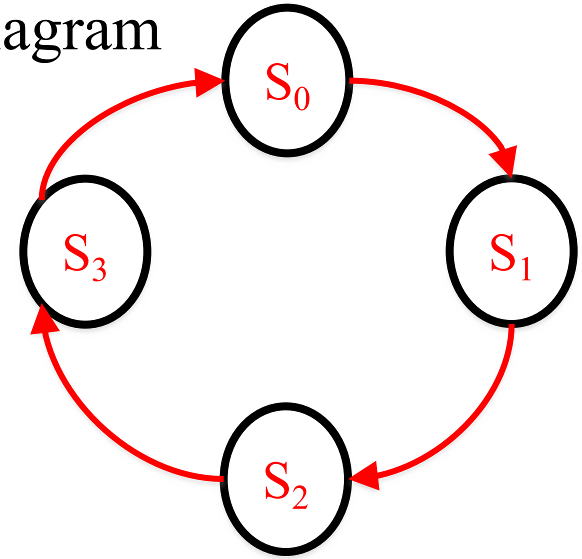


Diagram that depicts behavior over time



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

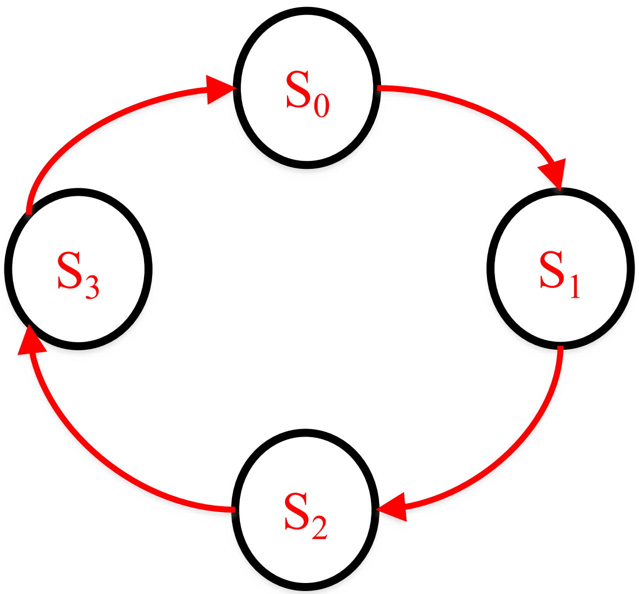
State Assignment

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

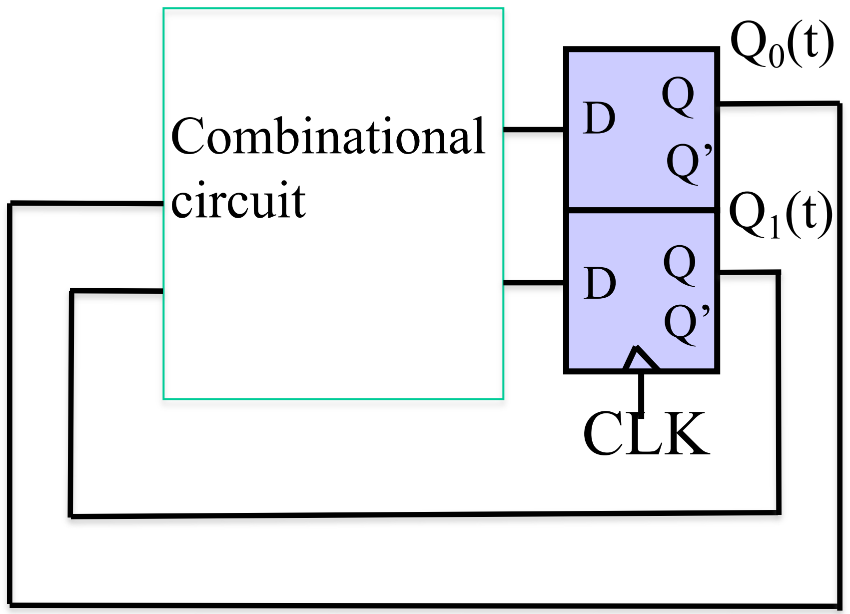
State Table

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

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

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



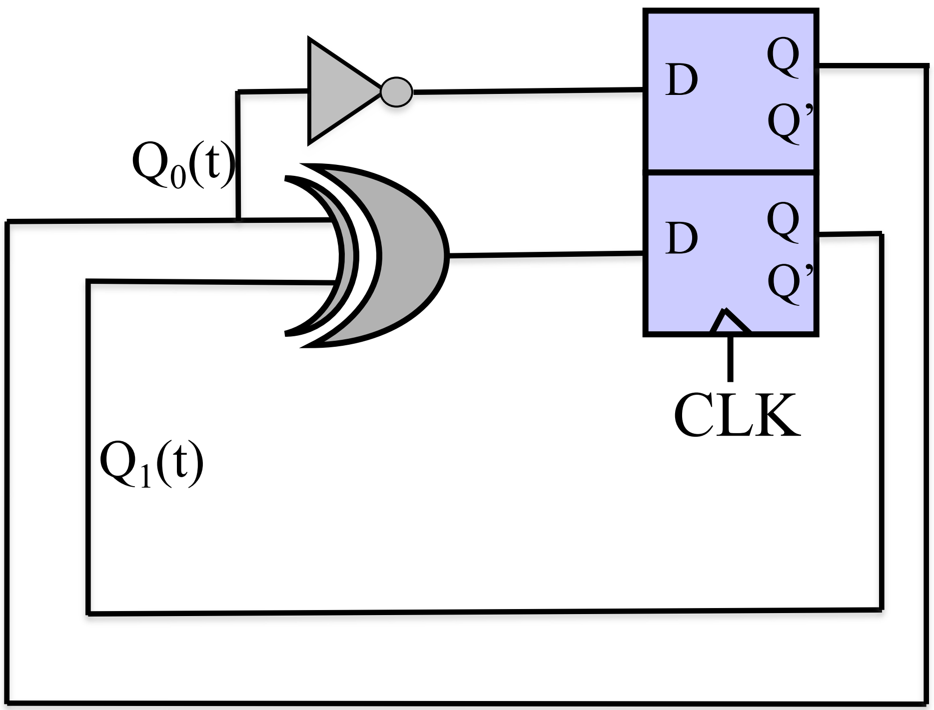
Circuit with 2 flip flops

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

State Table

Truth table→K map→Switching function

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

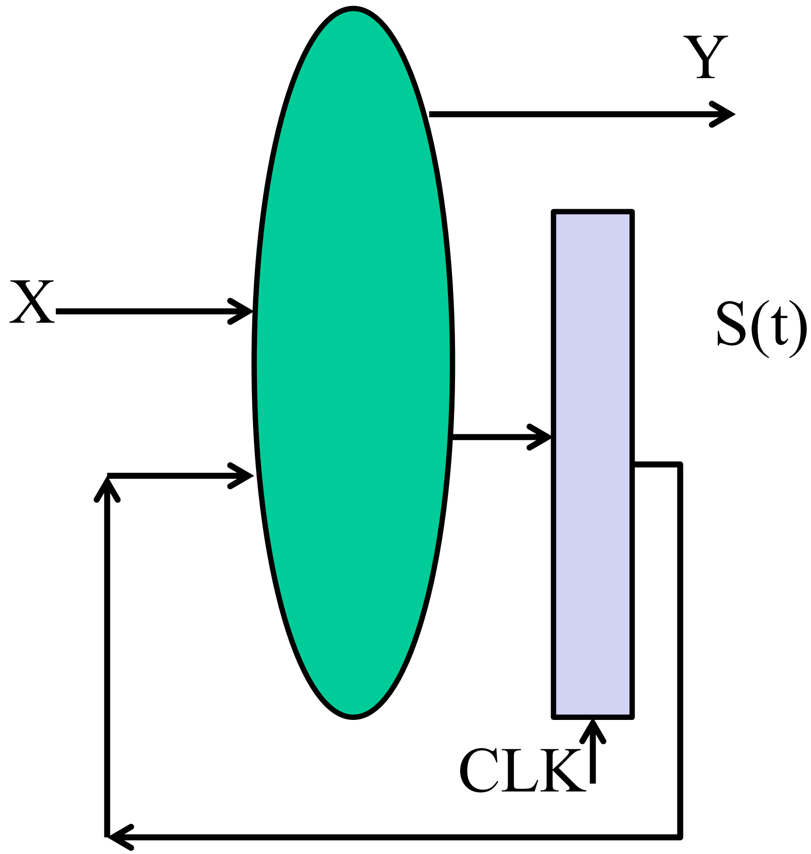


Implementation of 2-bit counter

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

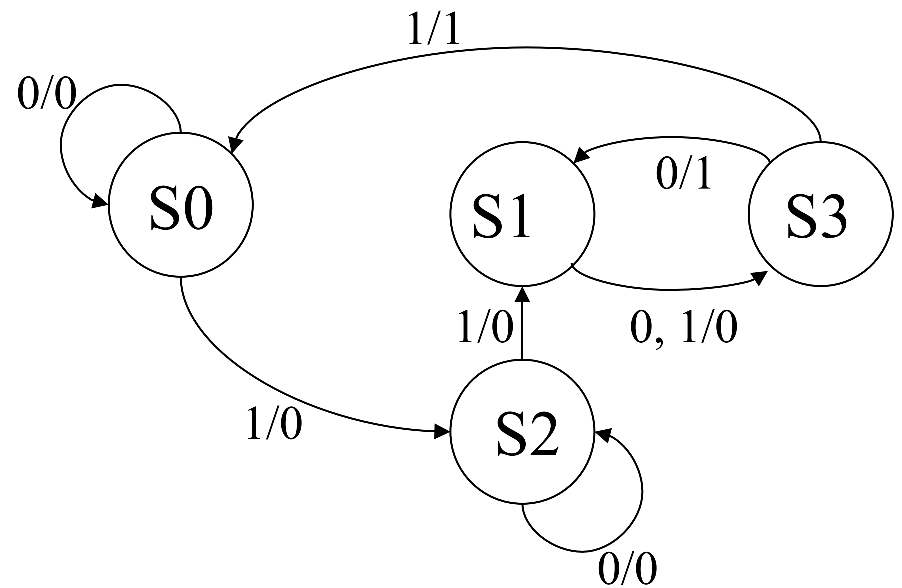
- Inputs to this combinational circuit don't change while the next output is being computed (free-running).
- The transition to the next state occurs only at rising edge of clock.
- Skip the K map & logic optimization and let the synthesizer do it.
- Revisit by hand if not meeting timing.

Generalized Model of Sequential Circuits



Netlist ⇔ State Table ⇔ State Diagram ⇔ 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

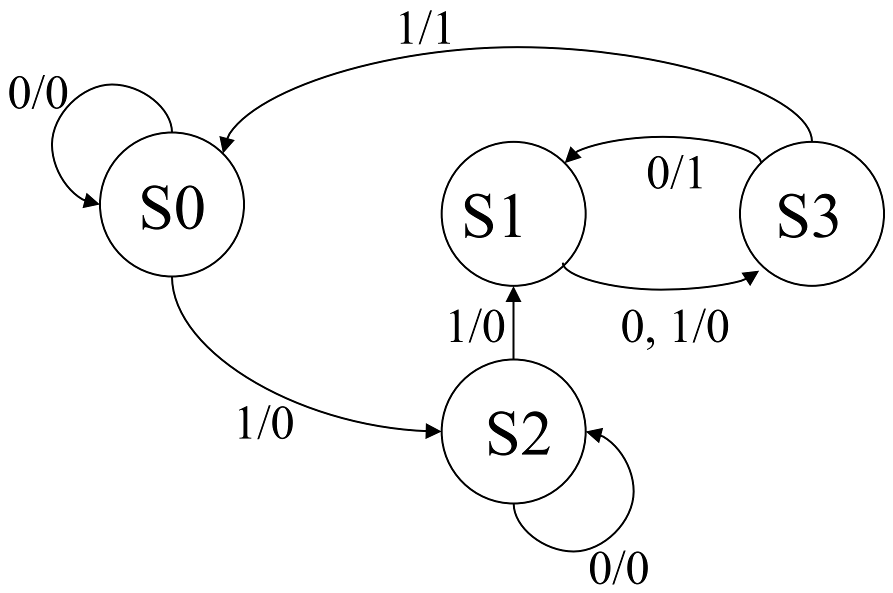


Example: Output sequence

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

Netlist ⇔ State Table ⇔ State Diagram ⇔ 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

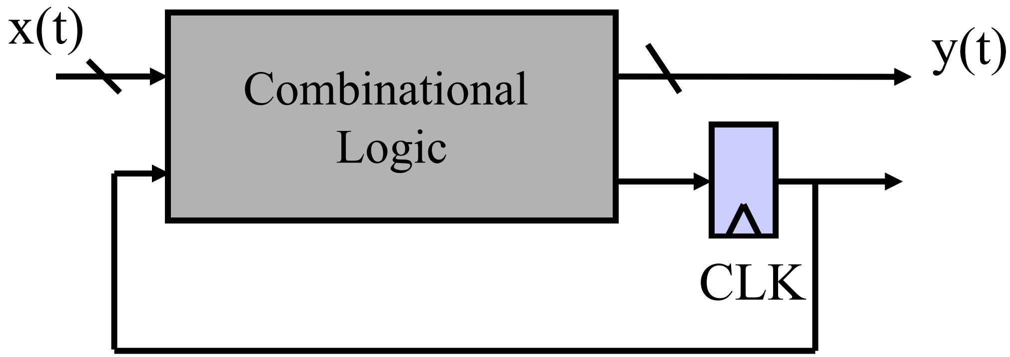
Time	0	1	2	3	4	5
Input	0	1	1	0	1	-
State	S0	S0	S2	S1	S3	S0
Output	0	0	0	0	1	0

Implementation

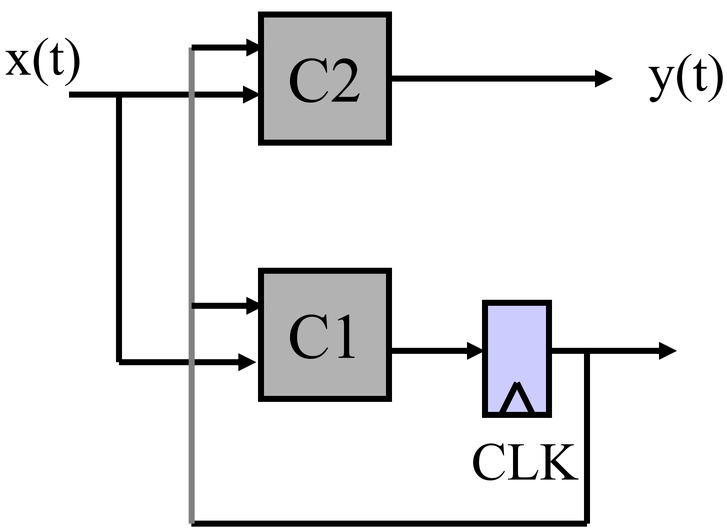
State Diagram => State Table => Logic Diagram

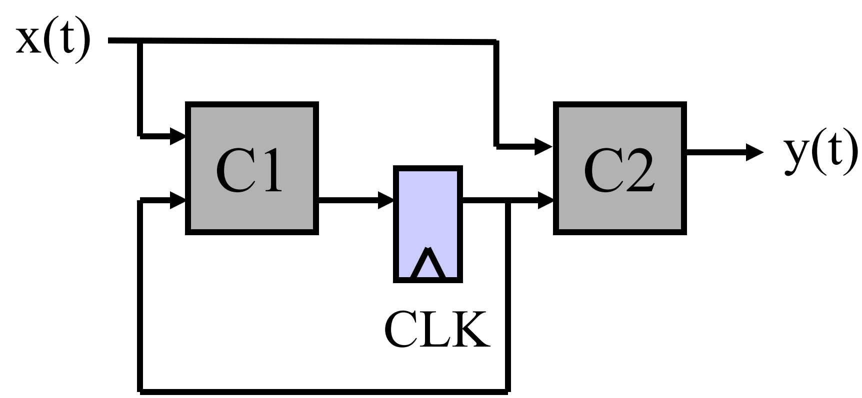
- Canonical Form: Mealy and Moore Machines
  - Mealy machines: General
  - Moore machines: Output is independent of current input (subset of Mealy).
- Excitation Table
  - Truth Table of the F-F Inputs
  - Boolean algebra, K-maps for combinational logic
- Examples
- Timing

Canonical Form: Mealy and Moore Machines



e.g., Traffic Light Controller:
C1 = brains
C2 maps state to lights themselves

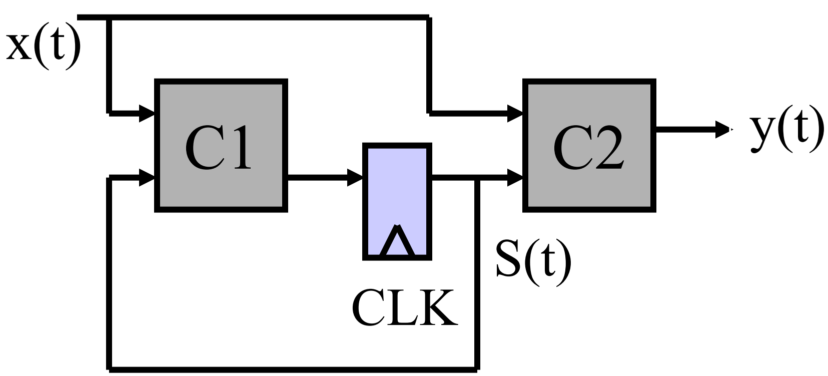




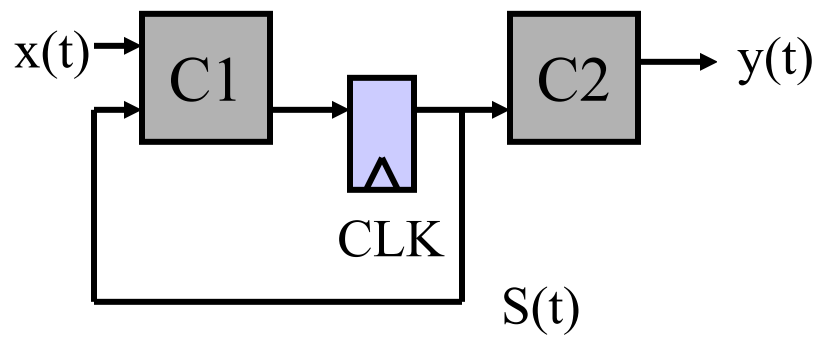
Canonical Form: Mealy and Moore Machines

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

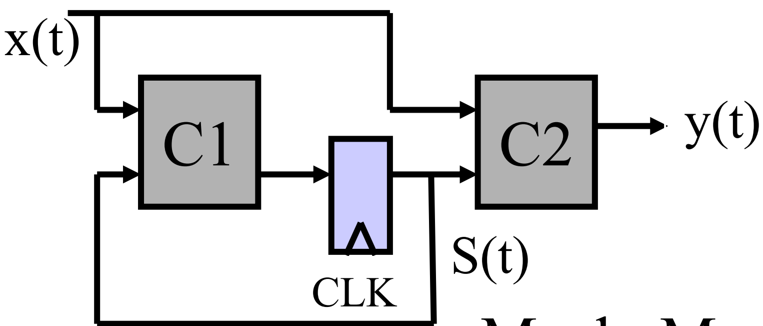
often C2=wire →
output is synchronous w/ CLK

Canonical Form: Mealy and Moore Machines

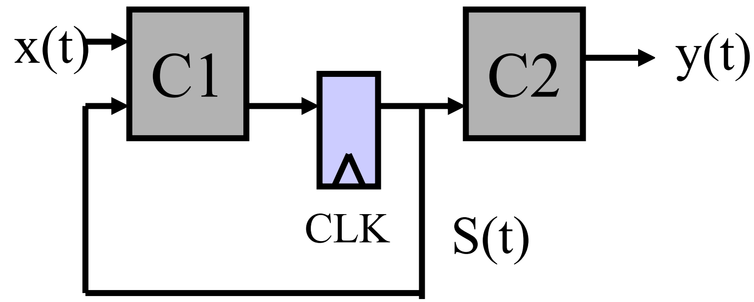
Mealy Machine: $y_i(t) = f_i(X(t), S(t))$

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

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



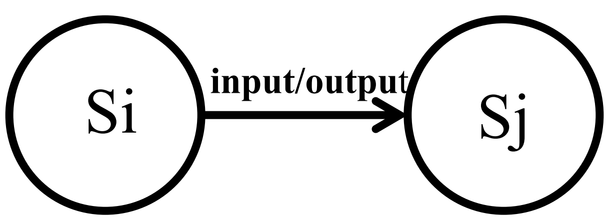
Mealy Machine

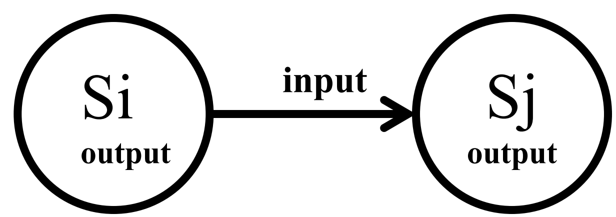


Moore Machine

	Input
PS	NS, output

	Input	
PS	NS	Output





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 ☹).

---

* Think if Binary Phase Shift Keying. To send a 0, don't invert the sinusoidal carrier. To send a 1, invert the sinusoidal carrier (phase shift 180 degrees) for the duration of the 1. Narrow-band: symbol (data bit) duration >> carrier cycle time. Ultrawideband (UWB) -- modulation frequency might be only about 1/3 of carrier frequency.

Implement the Life-on-Mars Pattern Recognizer!

The WAR of the WORLDS

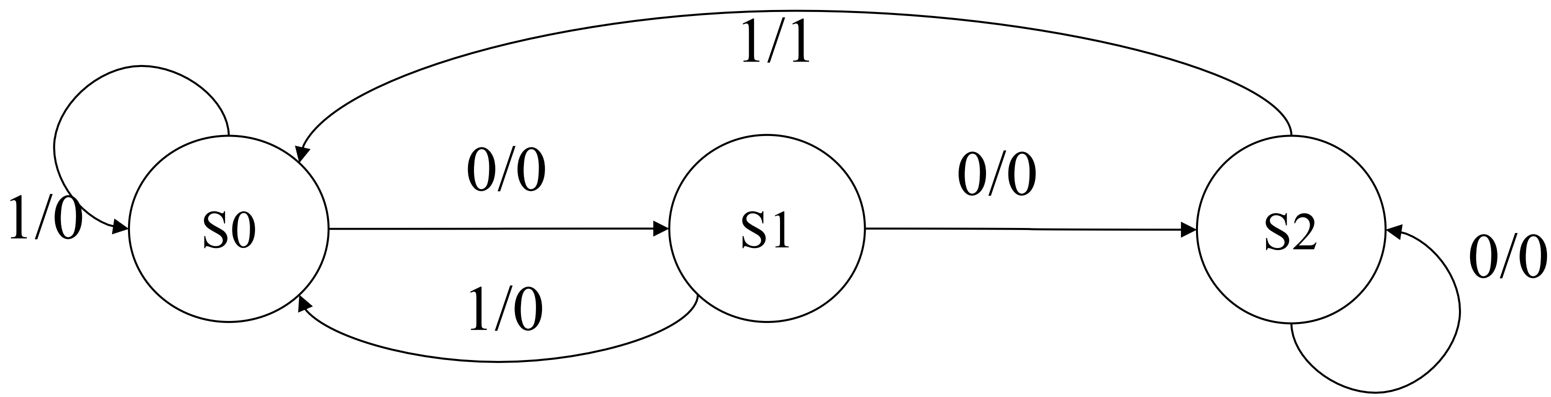
By H. G. Wells

Author of "Under the Knife," "The Time Machine," etc.

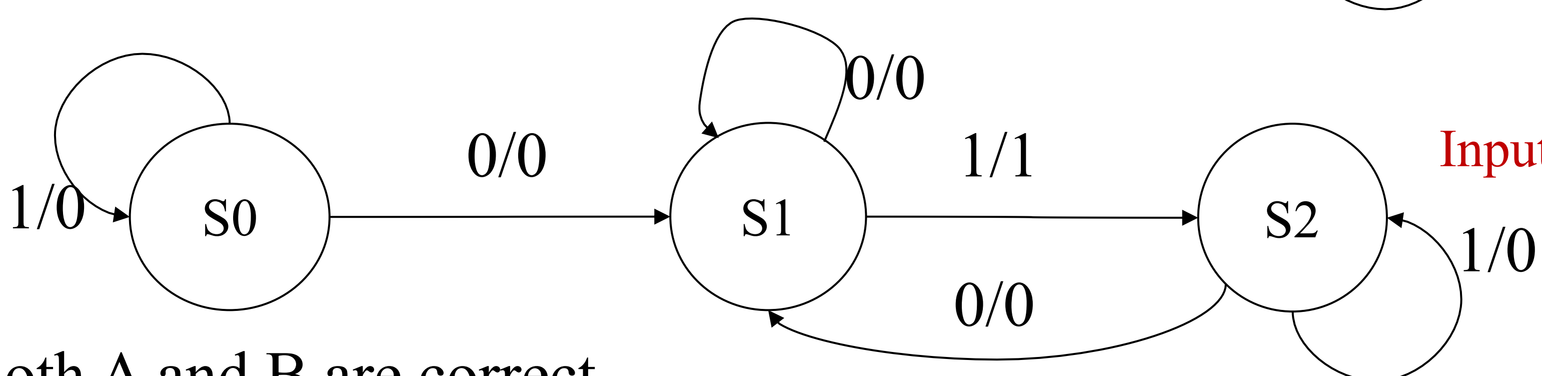
Mars Life Recognizer FSM

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

A.



B.



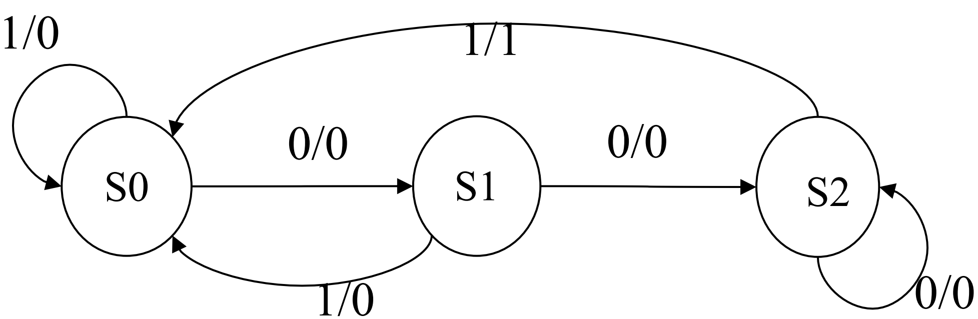
Input/output

C. Both A and B are correct

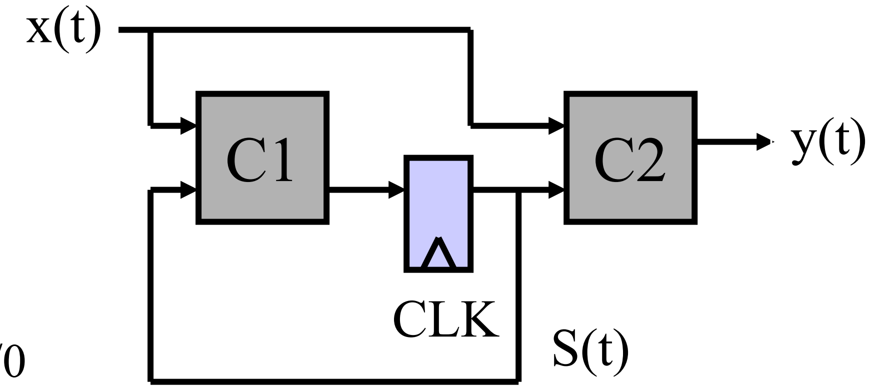
D. None of the above

Mars Life Recognizer FFs

Pattern Recognizer '001'



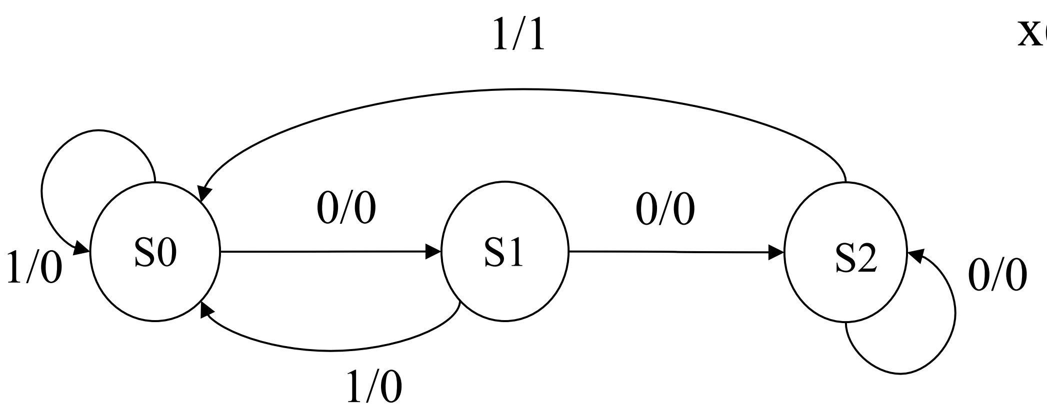
Mealy Machine

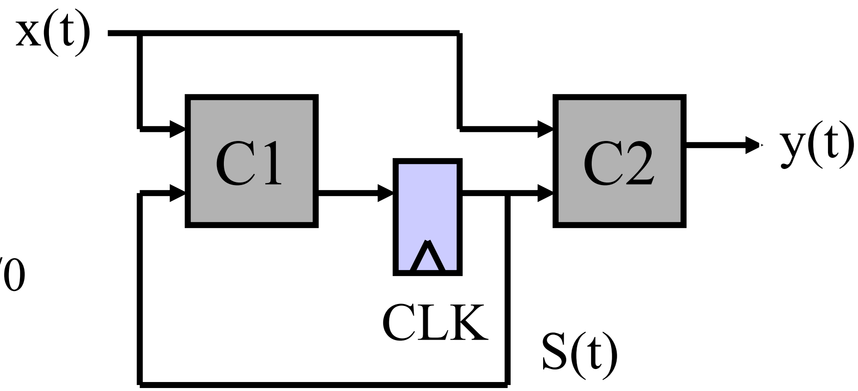


What does state table need to show to design controls of C1?

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





Mealy Machine

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

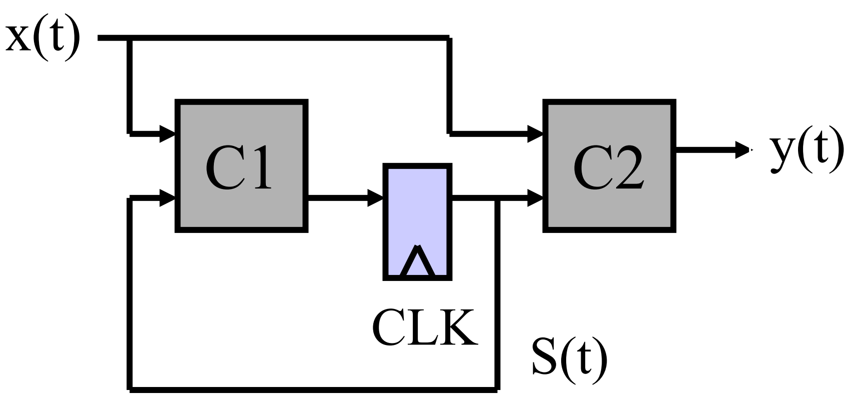
State Assignment
S0: 00
S1: 01
S2: 10

S(t)\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

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

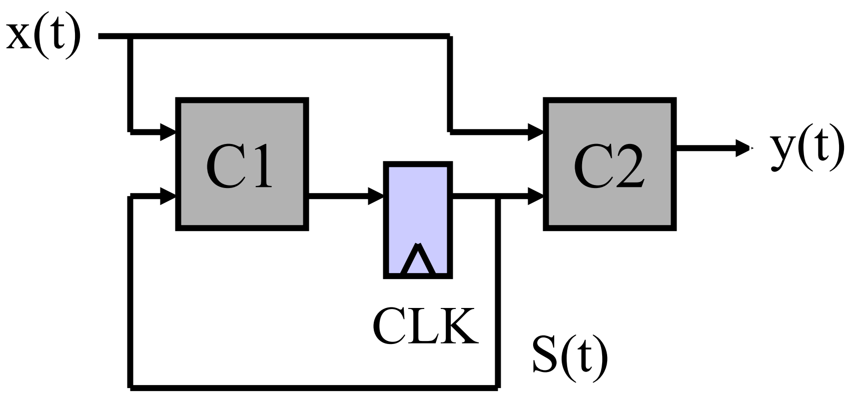


Mealy Machine

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			
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_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			
7	111			

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

$D_1(t)$:

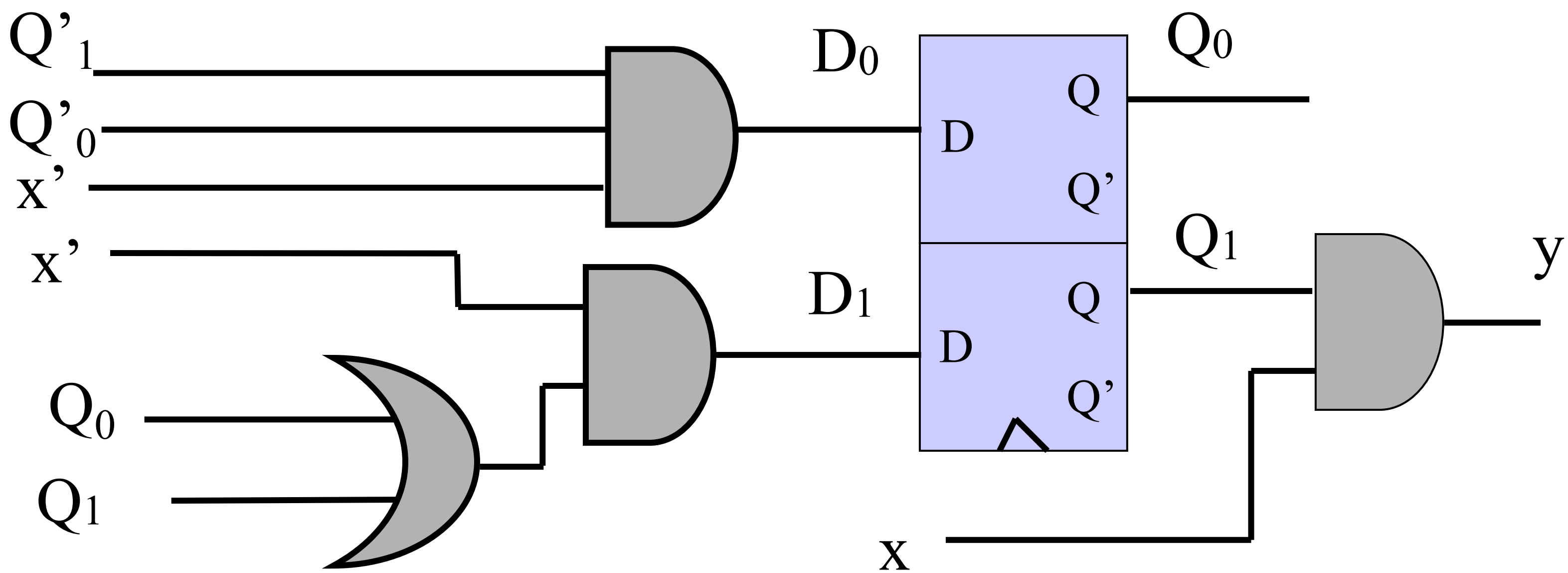
		Q_0	Q_0	
	0: 0	2: 1	6: X	4: 1
$x(t)$	1: 0	3: 0	7: X	5: 0
			Q_1	Q_1

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

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

$y = Q_1x$

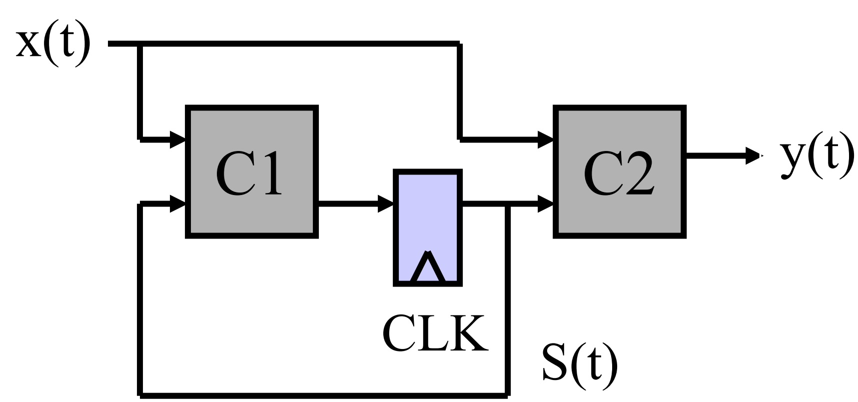
State Diagram => State Table => Excitation Table => Circuit



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

$D_0(t) = Q'_1 Q'_0 x'$

$y = Q_1 x$

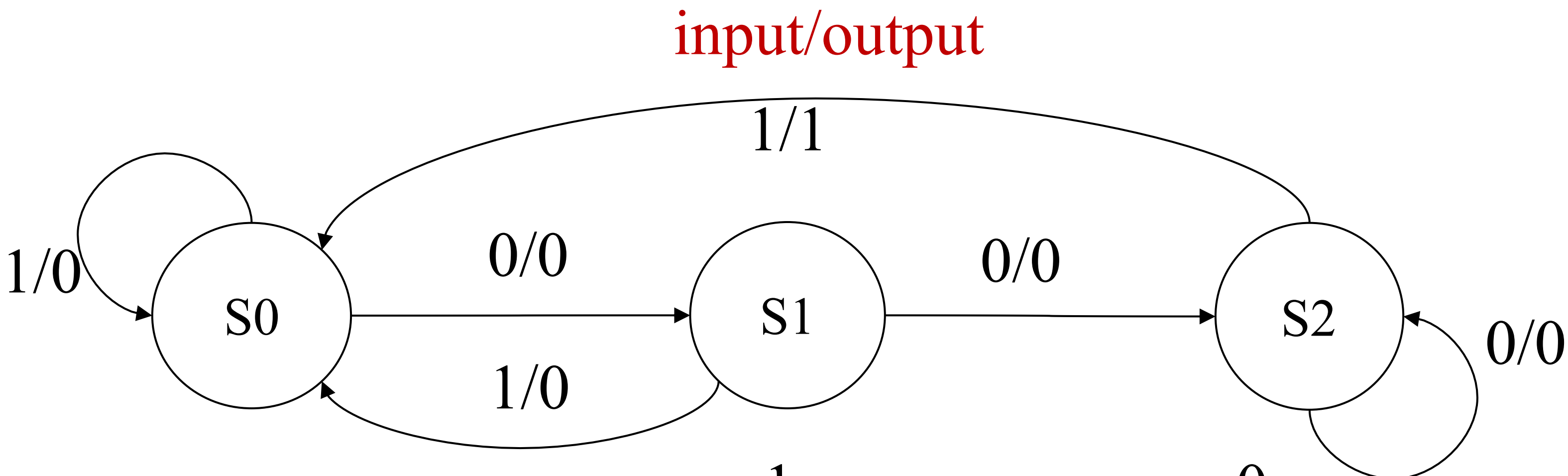


Mealy Machine

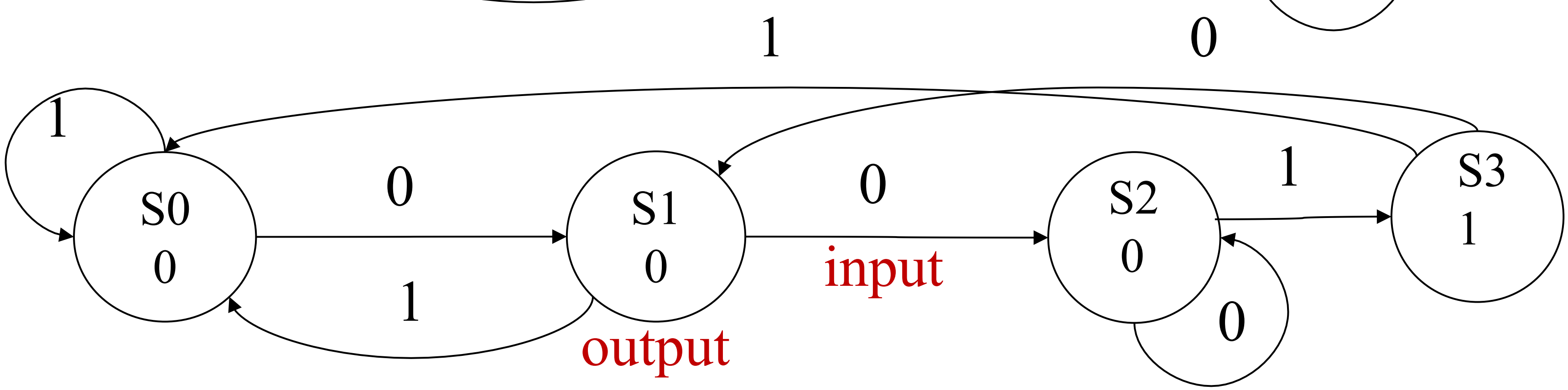
Moore FSM for the Mars Life Recognizer

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

A.



B.

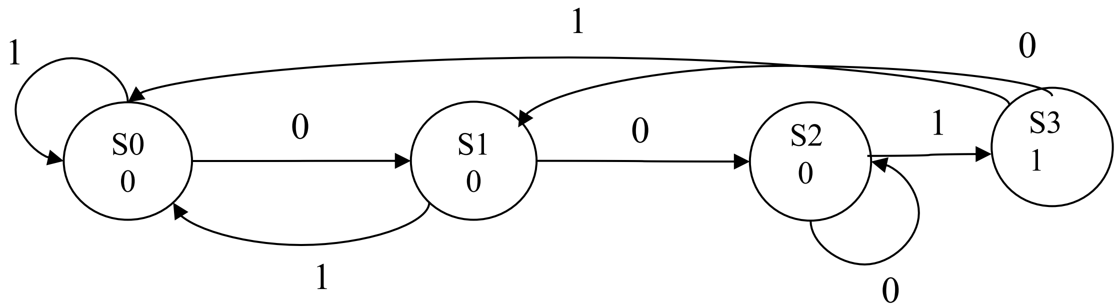


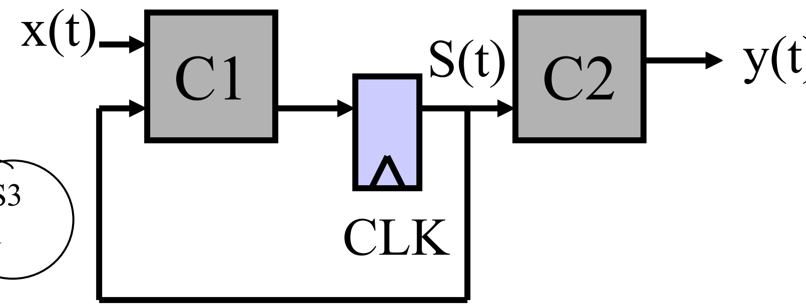
C. Both A and B are correct

D. None of the above

Moore Mars Life Recognizer: FF Input Specs

Pattern Recognizer ‘001’





Moore Machine

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))

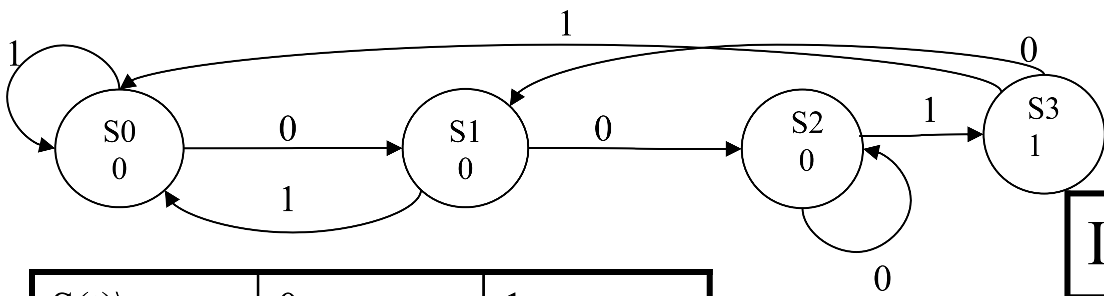
B. (current input, current state vs. current output y(t))

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_1Q_0\x	0	1
00	01,0	00,0
01	10,0	00,0
10	10,0	11,0
11	01,1	00,1

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

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

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

$D_1(t)$:

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

$D_0(t)$:

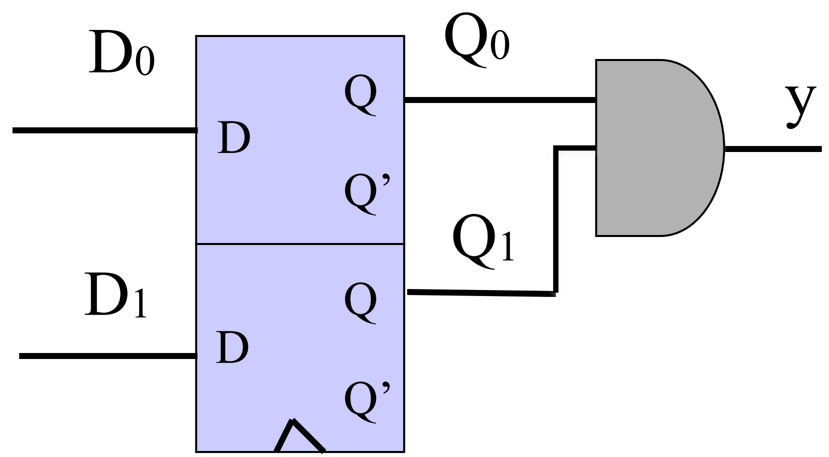
		Q_0		
	0: 1	2: 0	6: 1	4: 0
x(t)	1: 0	3: 0	7: 0	5: 1
			Q_1	

y(t):

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

Mars Life Recognizer Circuit Implementation

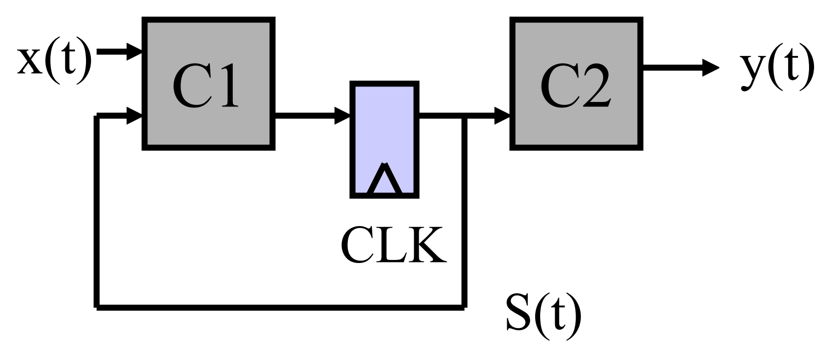
State Diagram => State Table => Excitation Table => Circuit



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

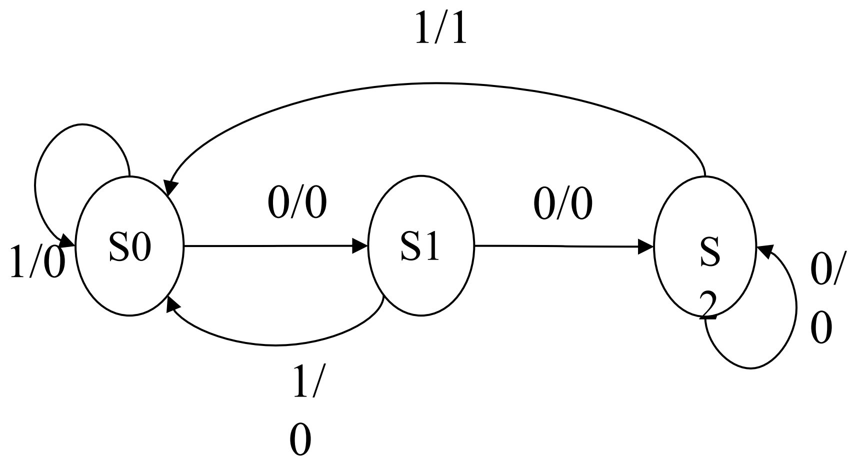
$D_0(t) = Q_1(t)'Q_0(t)'x(t)' + Q_1(t)Q_0(t)\, x(t)' + Q_1(t)Q_0(t)'\, x(t)$

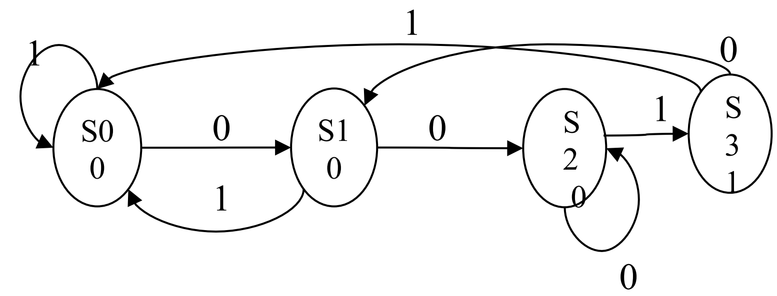
$y(t) = Q_1(t)Q_0(t)$



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**

Moore

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

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

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

A.S2, S3, 1

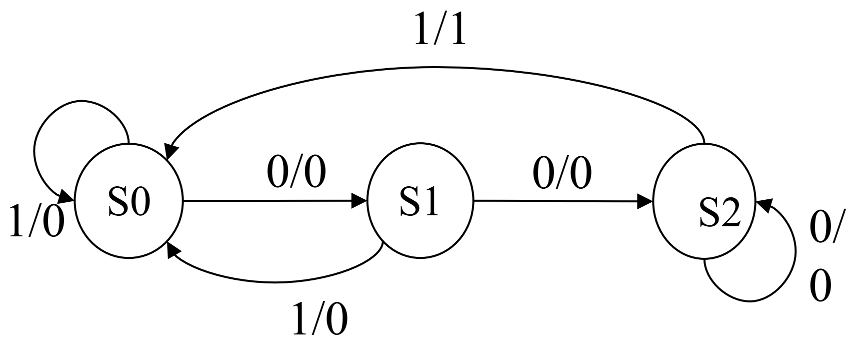
B.S2, S0, 1

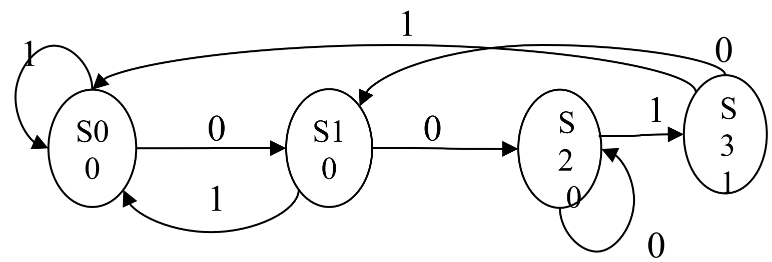
C.S1, S0, 1

D.S1, S0. 0

E.None of the above.

Conversion from Mealy to Moore Machine



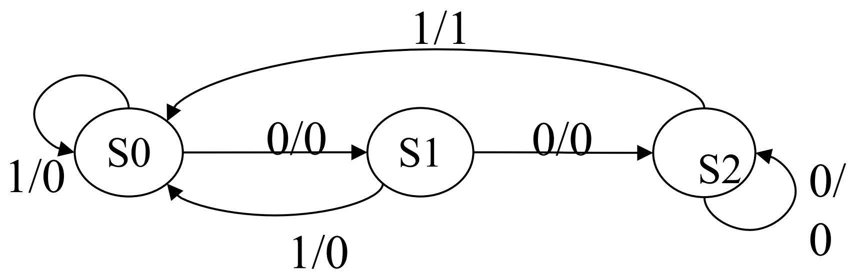


S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S0,1

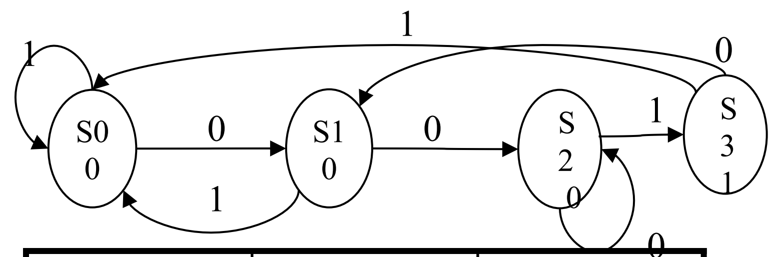
S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

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



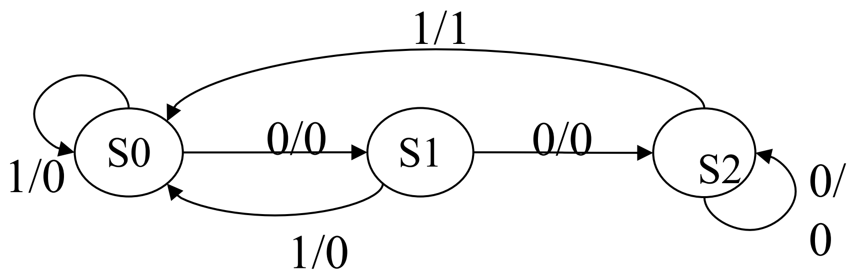
S(t)\x	0	1
S0	S1,0	S0,0
S1	S2,0	S0,0
S2	S2,0	S3,0
S3	S1,1	S0,1

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}									

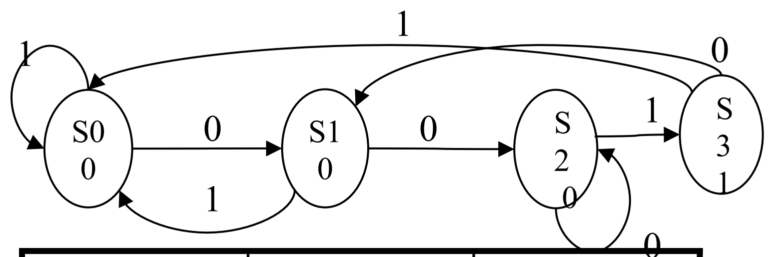
iClicker $S_{moore}[0\text{-}5]$

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}									

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

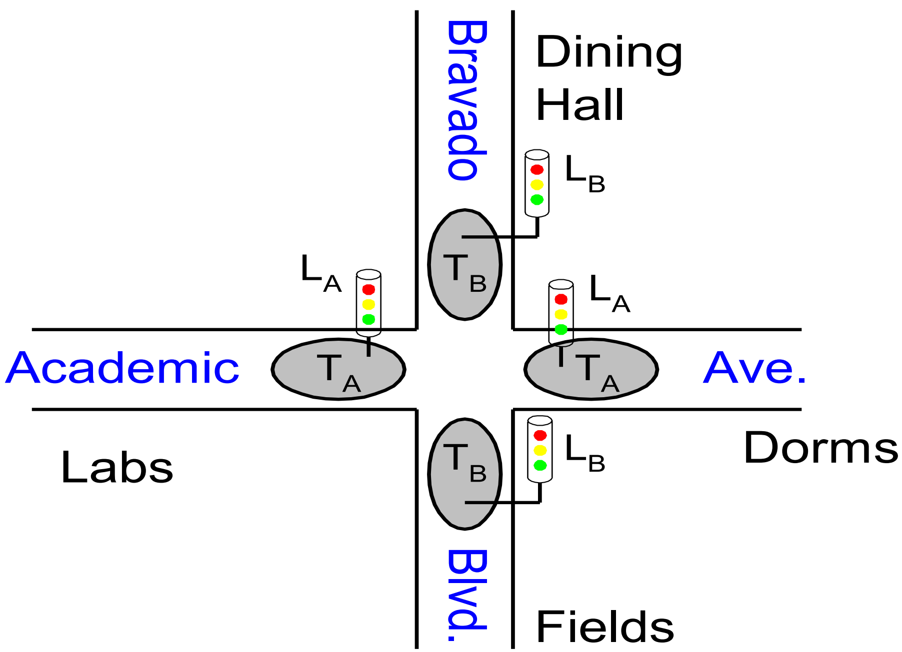
Conversion from Mealy to Moore Machine

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

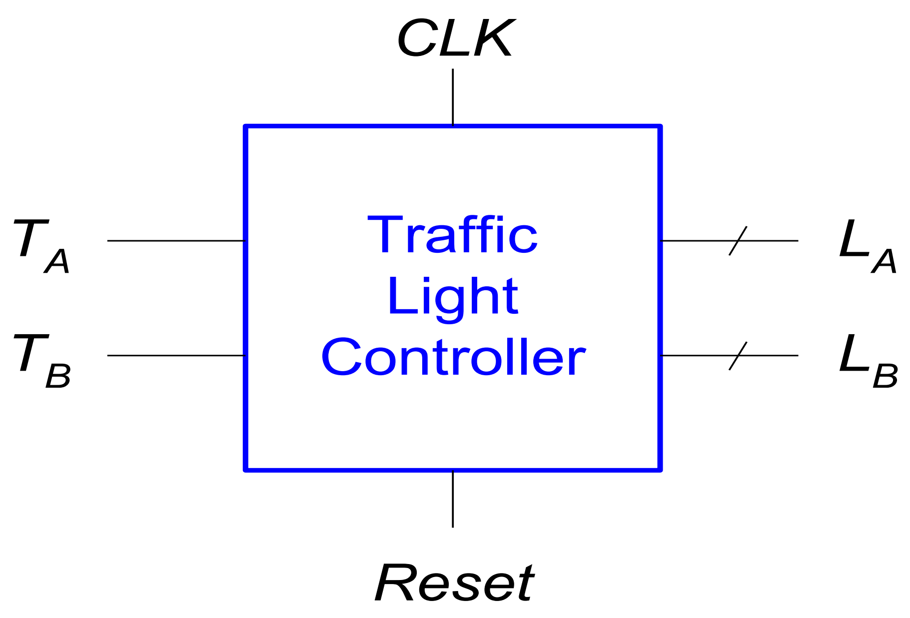
Finite State Machine Example

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



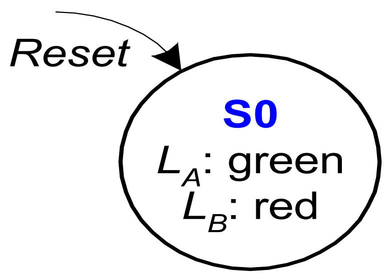
FSM Black Box

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



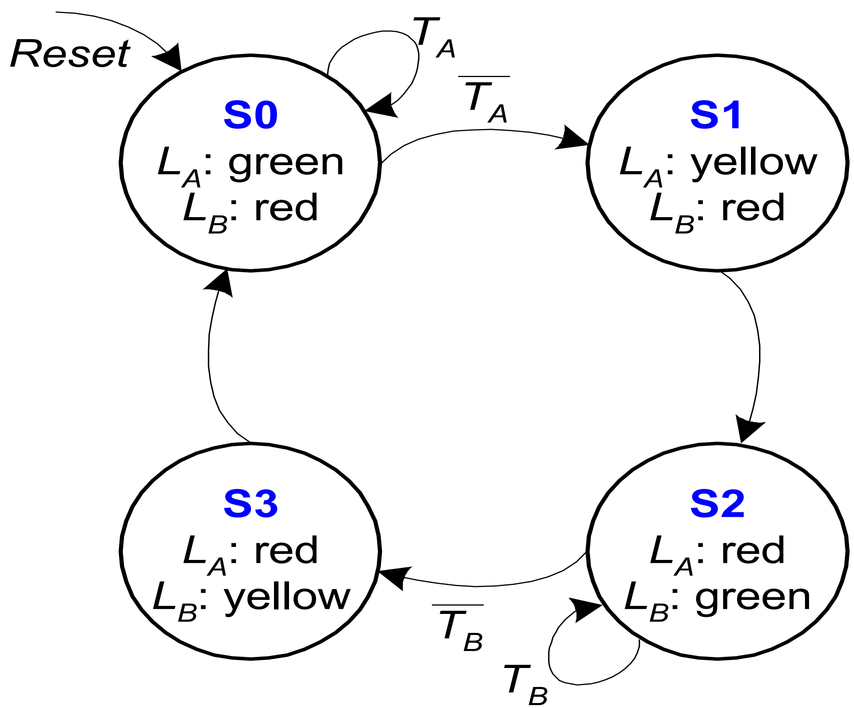
FSM State Transition Diagram

- 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	
$Q_1(\mathrm{t})$	$Q_0(\mathrm{t})$	T_A	T_B	$Q_1(\mathrm{t}+1)$	$Q_0(\mathrm{t}+1)$
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

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

$$Q_0(t+1) = Q'_1(\mathrm{t})Q'_0(\mathrm{t})T'_A + Q_1(\mathrm{t})Q'_0(\mathrm{t})T'_B$$

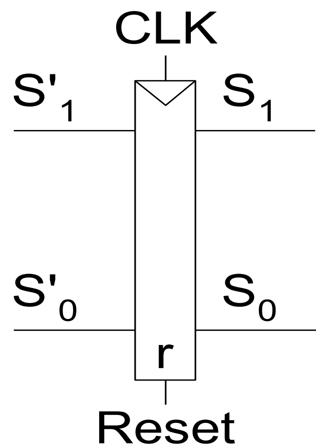
FSM Output Table

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

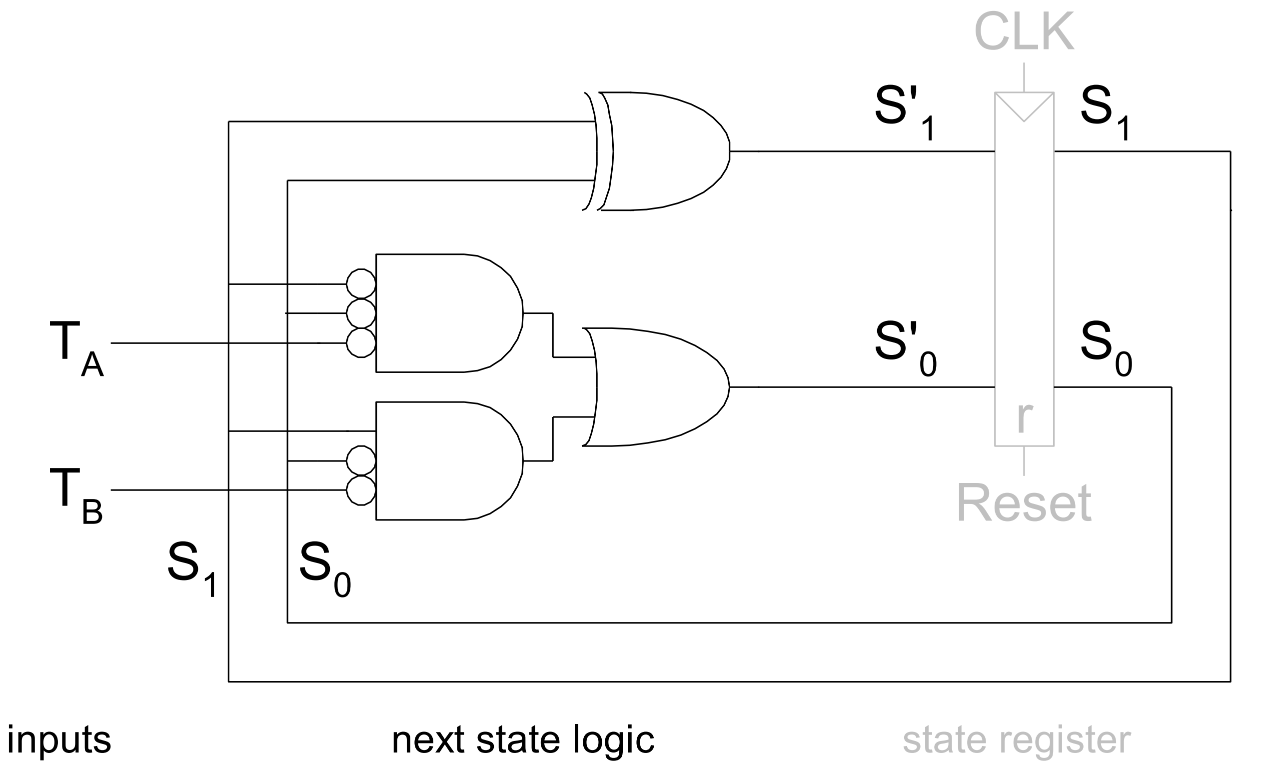
$$L_{A1} = Q_1$$
$$L_{A0} = Q'_1 Q_0$$
$$L_{B1} = Q'_1$$
$$L_{B0} = Q_1 Q_0$$

FSM Schematic: State Register

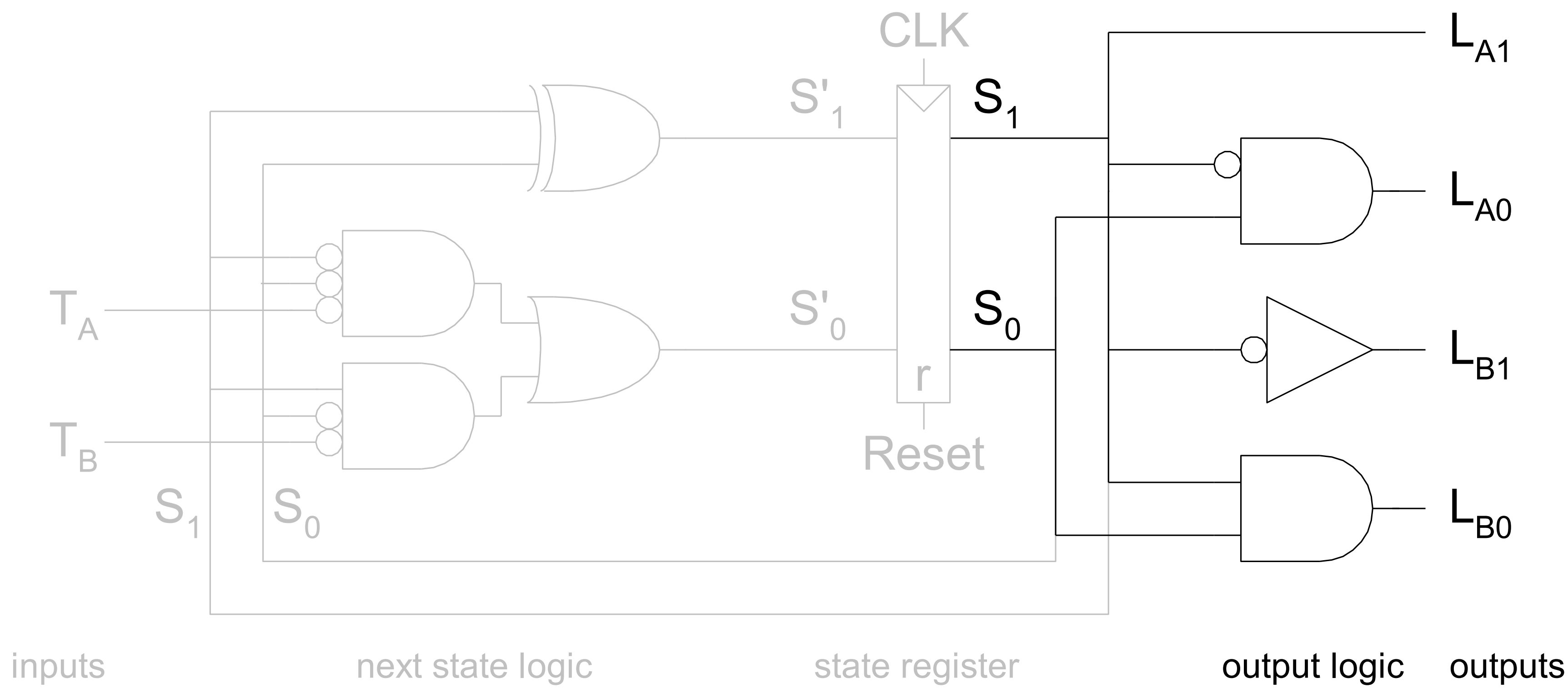


state register

Logic Diagram



FSM Schematic: Output Logic



Summary: Implementation

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