CSE 140: Components and Design Techniques for Digital Systems

Lecture 8:

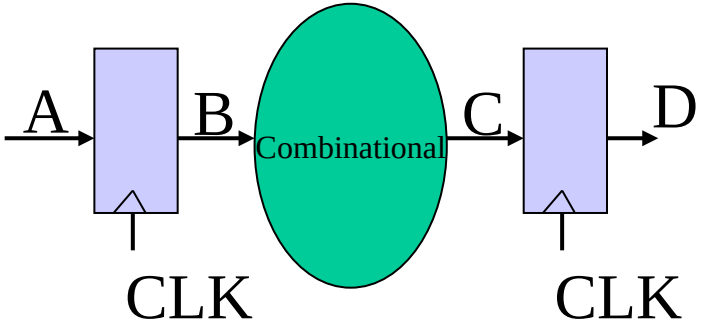
Sequential Networks and Finite State Machines

CK Cheng

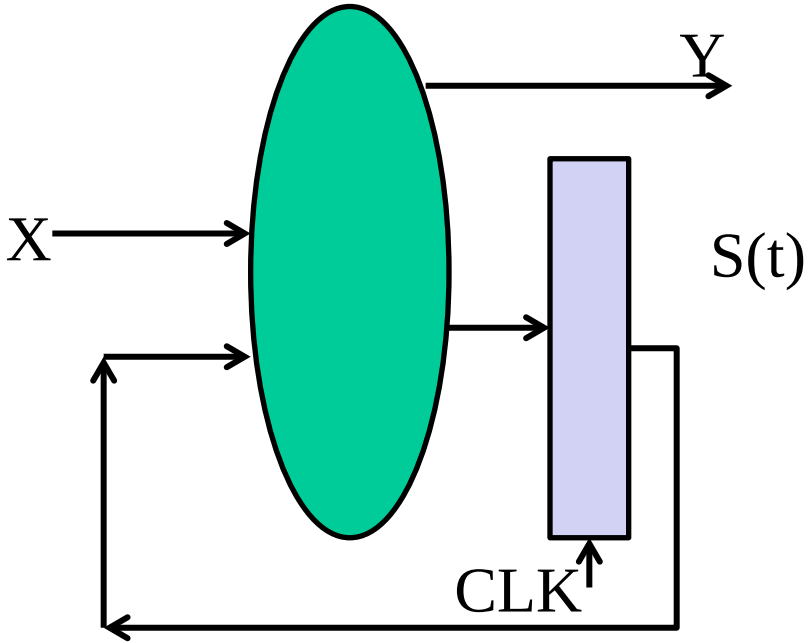
Dept. of Computer Science and Engineering

University of California, San Diego

Sequential Networks



RTL: Register-Transfer Level Description



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

Specification

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

Specification: Finite State Machine

- Input Output Relation
- State Diagram (Transition of States)
- State Table (Truth table of states)
- Excitation Table (Truth table of FF inputs)
- Boolean Expression
- Logic Diagram

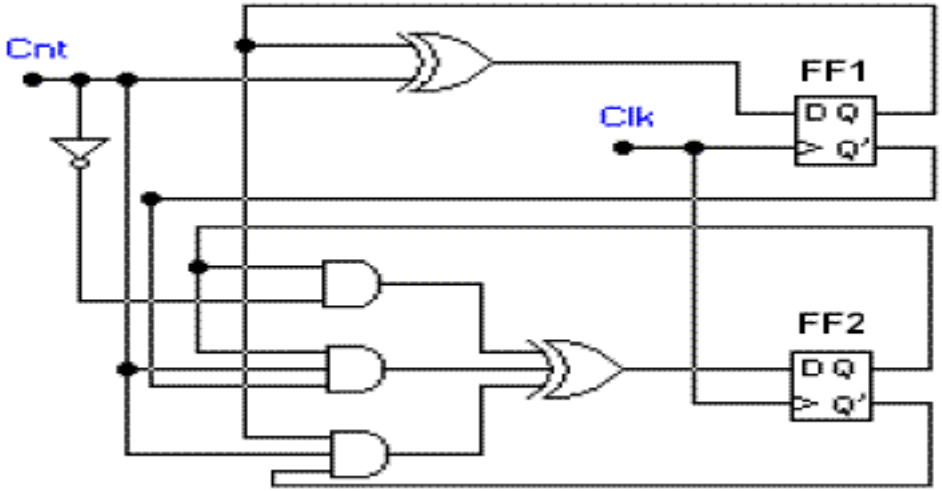
Specification: Examples

- Transition from circuit to finite state machine representation
  - Netlist => State Table => State Diagram => Input Output Relation
- Example 1: a circuit with D Flip Flops
- Example 2: a circuit with other Flip Flops

What we will learn:

1. Given a sequential circuit, describe its behavior over time
2. Given the behavior of a sequential circuit, implement the circuit

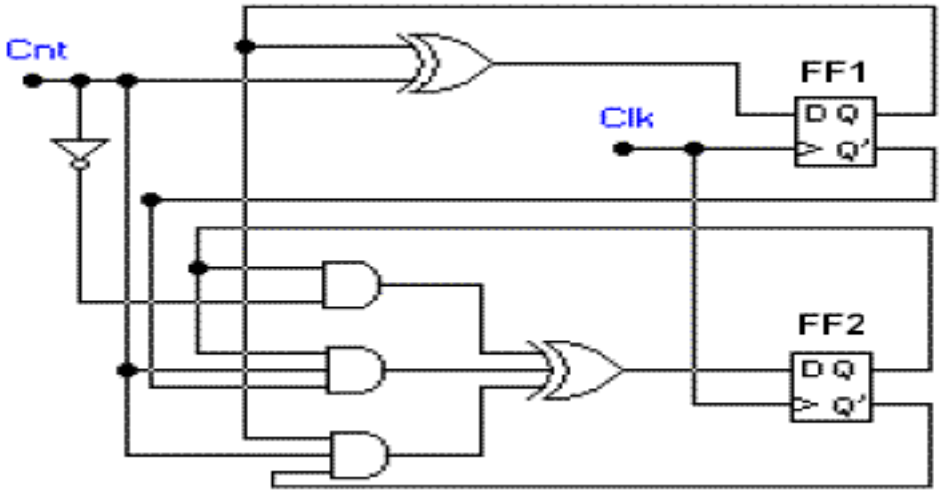
Sequential Circuit: Wall-E



How does Wall-E behave?

What does it mean to describe the behavior of a sequential circuit

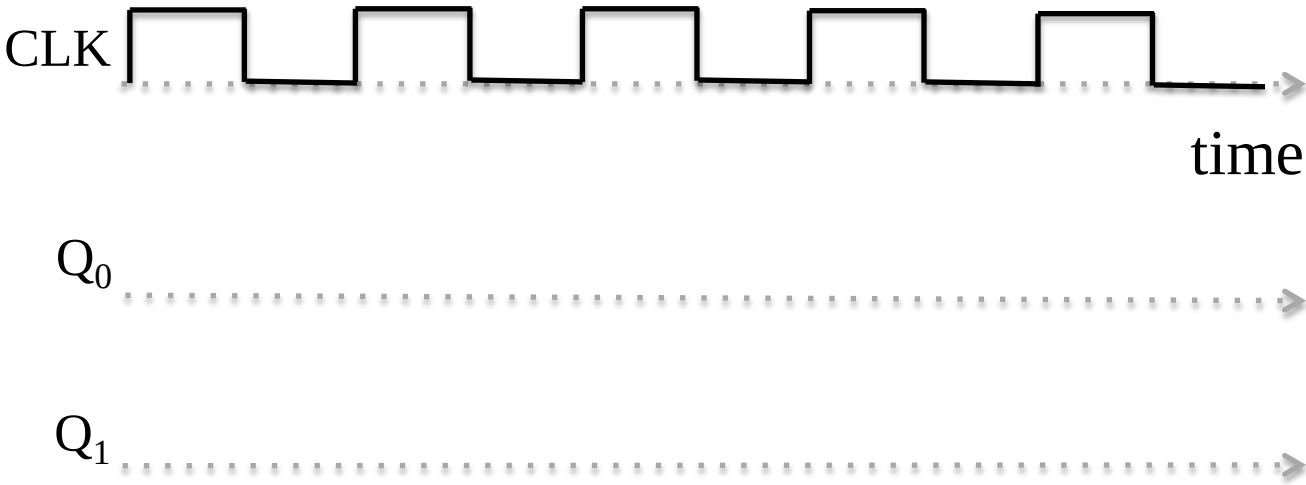
Specify how the output of the circuit changes as a function of inputs and the state of the circuit



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

Symbol/ Circuit

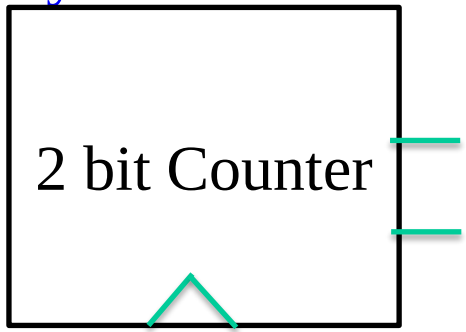
Behavior over time



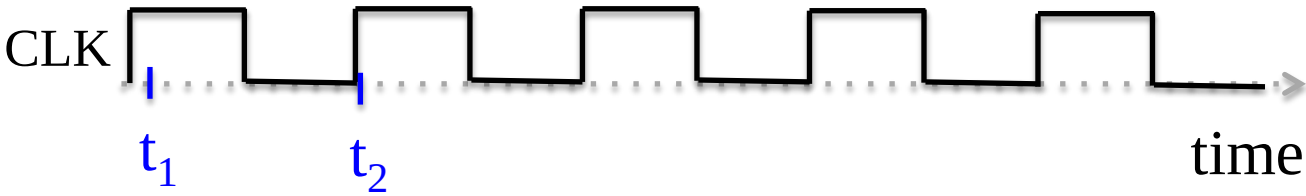
What is the expected output of the counter over time?

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

Symbol/ Circuit



Behavior over time



PI Q: At time t_1, what information is needed to produce the output of the counter at the next rising edge of the clock (i.e. t_2)?

A. All the outputs of the counter until t_1
B. The initial output of the counter at time t=0
C. The output of the counter at current time t_1
D. We cannot determine the output of the counter at t_2 prior to t_2

Finite State Machines: Describing circuit behavior over time

Symbol/ Circuit

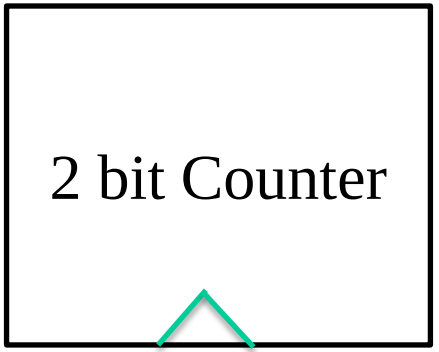
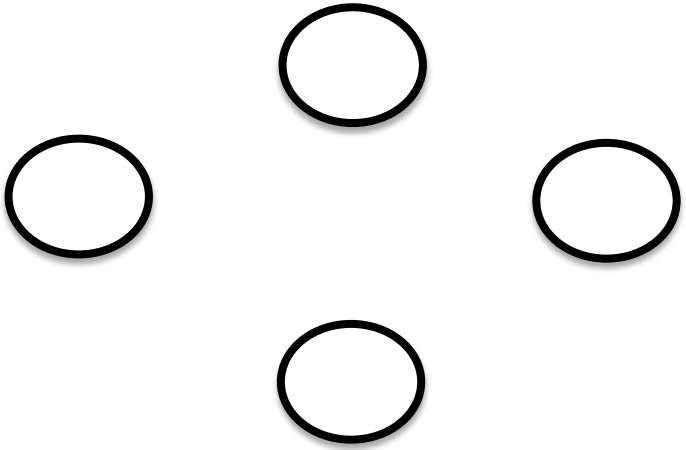
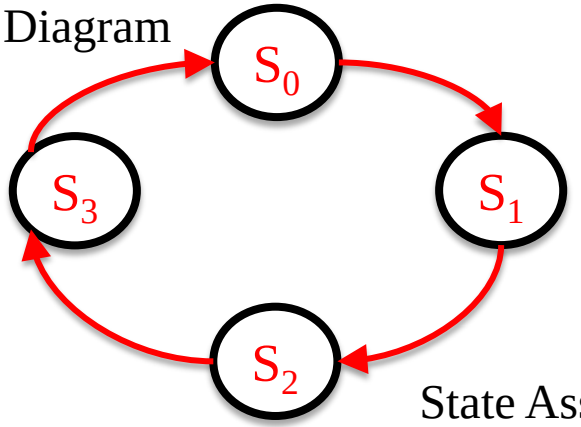


Diagram that depicts behavior over time



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

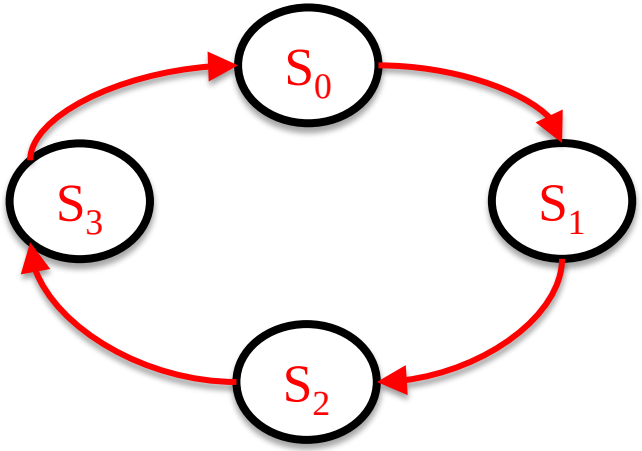
State Assignment

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

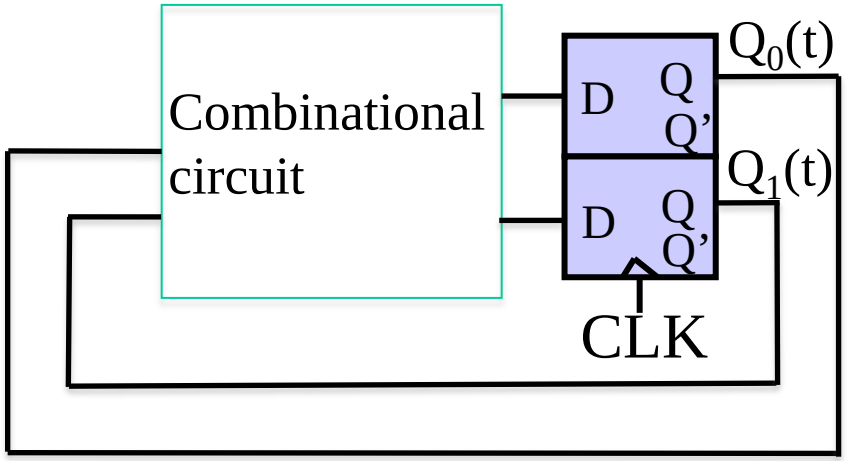
$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

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

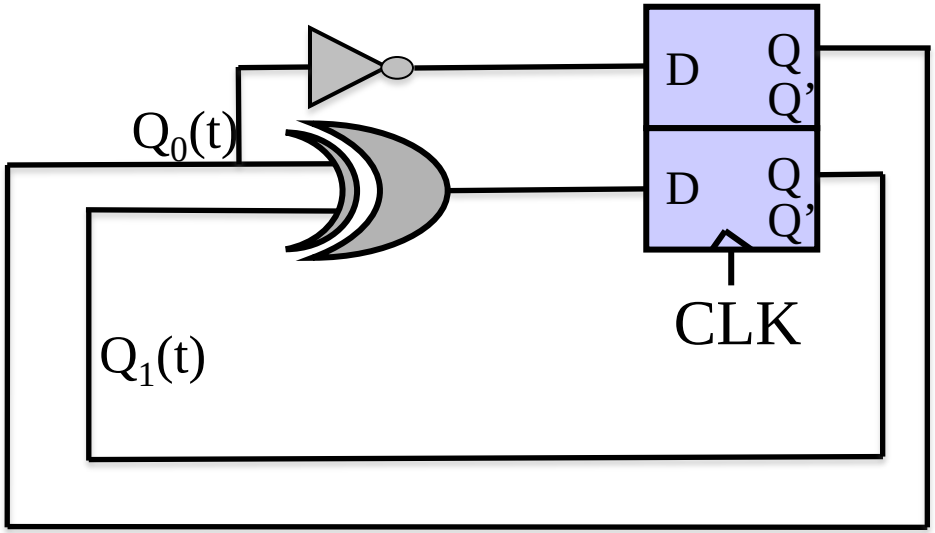
$Q_1(t)$	$Q_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

State Table

Truth table→K map→Switching function

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

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

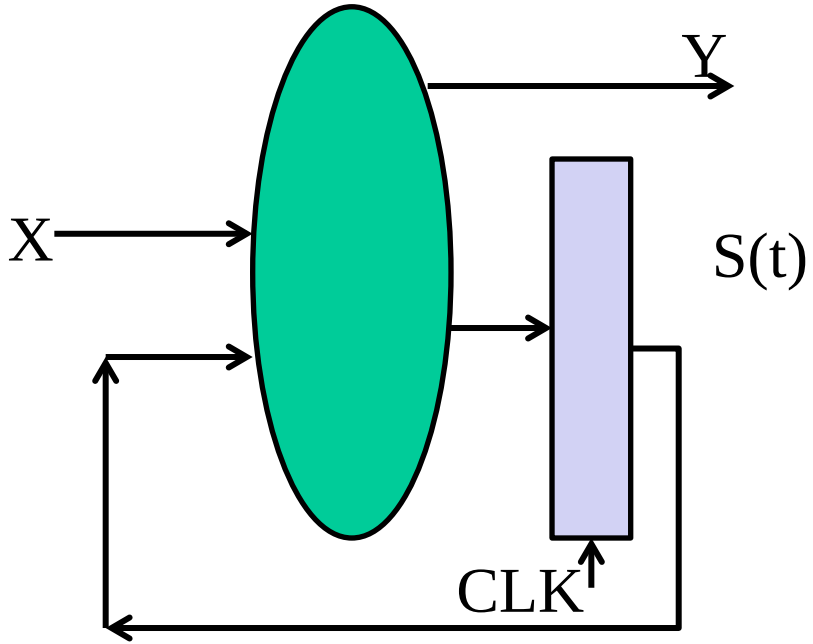


Implementation of 2-bit counter

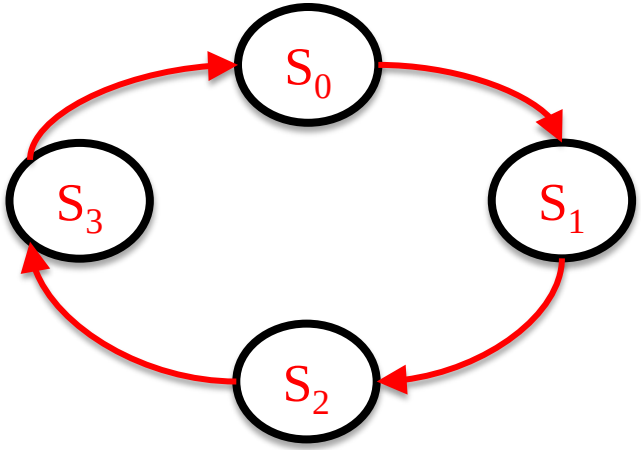
We store the current state using D-flip flops so that:

- 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



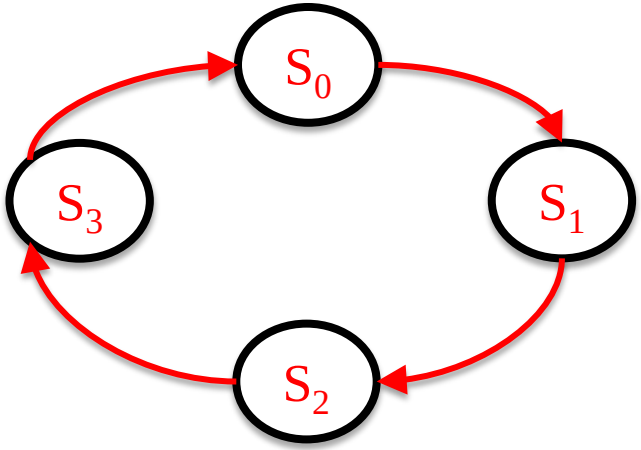
Let’s implement our free running 2-bit counter using T-flip flops



State Table

PS	Next state
S_0	S_1
S_1	S_2
S_2	S_3
S_3	S_0

Let’s implement our free running 2-bit counter using T-flip flops



State Table

S_0	S_1
S_1	S_2
S_2	S_3
S_3	S_0

State Table with Assigned Encoding

Current	Next
0 0	01
0 1	10
1 0	11
1 1	00

Let’s implement our free running 2-bit counter using T-flip flops

Excitation table

id	$Q_1(t)$	$Q_0(t)$	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0			0	1
1	0	1			1	0
2	1	0			1	1
3	1	1			0	0

Let’s implement our free running 2-bit counter using T-flip flops

Excitation table

id	$Q_1(t)$	$Q_0(t)$	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	0	1	0	1
1	0	1	1	1	1	0
2	1	0	0	1	1	1
3	1	1	1	1	0	0

Let’s implement our free running 2-bit counter using T-flip flops

Excitation table

id	$Q_1(t)$	$Q_0(t)$	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	0	1	0	1
1	0	1	1	1	1	0
2	1	0	0	1	1	1
3	1	1	1	1	0	0

$T_0(t) =$

$T_1(t) =$

$Q_0(t+1) = T_0(t)\, Q'_0(t) + T'_0(t) Q_0(t)$

$Q_1(t+1) = T_1(t)\, Q'_1(t) + T'_1(t) Q_1(t)$

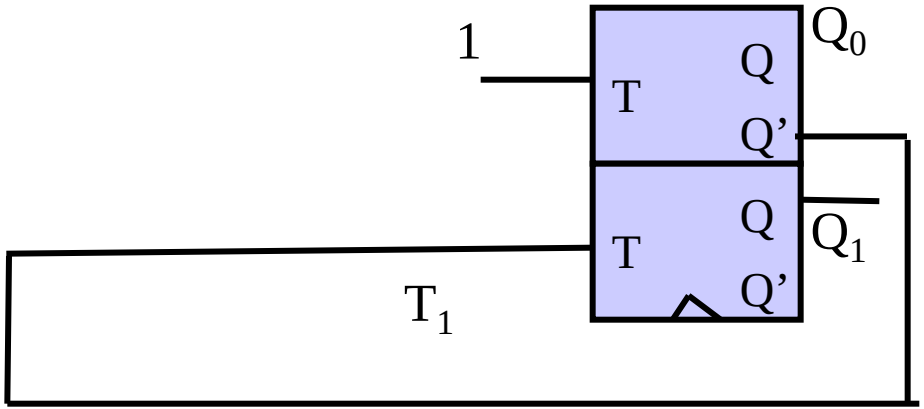
Let’s implement our free running 2-bit counter using T-flip flops

Excitation table

id	$Q_1(t)$	$Q_0(t)$	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$
0	0	0	0	1	0	1
1	0	1	1	1	1	0
2	1	0	0	1	1	1
3	1	1	1	1	0	0

$$T_0(t) = 1$$
$$T_1(t) = Q_0(t)$$

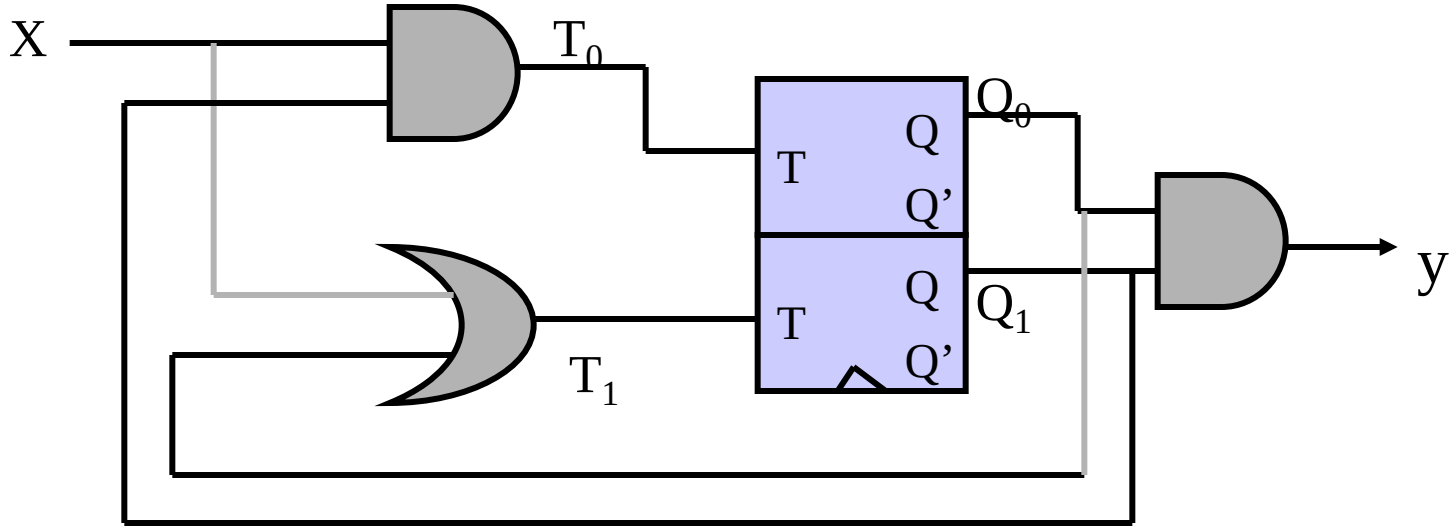
Free running counter with T flip flops



$T_0(t) = 1$

$T_1(t) = Q_0(t)$

Example 3 Circuit with T Flip-Flops



$$y(t) = Q_1(t)Q_0(t)$$
$$T_0(t) = x(t)\, Q_1(t)$$
$$T_1(t) = x(t) + Q_0(t)$$

Logic Diagram => Excitation Table => State Table

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

$T_0(t) = x(t)\, Q_1(t)$

$T_1(t) = x(t) + Q_0(t)$

$Q_0(t+1) = T_0(t)\, Q'_0(t) + T'_0(t)Q_0(t)$

$Q_1(t+1) = T_1(t)\, Q'_1(t) + T'_1(t)Q_1(t)$

Excitation Table:

Truth table of the F-F inputs

id	$Q_1(t)$	$Q_0(t)$	x	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$	y
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0
2	0	1	0	1	0	1	1	0
3	0	1	1	1	0	1	1	0
4	1	0	0	0	0	1	0	0
5	1	0	1	1	1	0	1	0
6	1	1	0	1	0	0	1	1
7	1	1	1	1	1	0	0	1

Excitation Table: iClicker

In excitation table, the inputs of the flip flops are used to produce

A. The present state

B. The next state

Excitation Table =>State Table => State Diagram

id	$Q_1(t)$	$Q_0(t)$	x	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$	y
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0
2	0	1	0	1	0	1	1	0
3	0	1	1	1	0	1	1	0
4	1	0	0	0	0	1	0	0
5	1	0	1	1	1	0	1	0
6	1	1	0	1	0	0	1	1
7	1	1	1	1	1	0	0	1

State Assignment

S0 00

S1 01

S2 10

S3 11

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

Excitation Table =>State Table => State Diagram

id	$Q_1(t)$	$Q_0(t)$	x	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$	y
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0
2	0	1	0	1	0	1	1	0
3	0	1	1	1	0	1	1	0
4	1	0	0	0	0	1	0	0
5	1	0	1	1	1	0	1	0
6	1	1	0	1	0	0	1	1
7	1	1	1	1	1	0	0	1

State Assignment

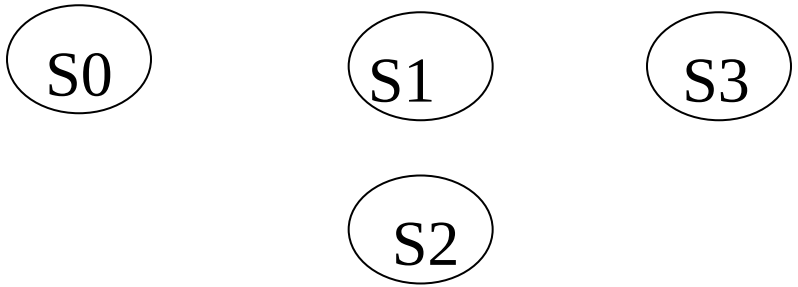
S0 00

S1 01

S2 10

S3 11

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



Excitation Table =>State Table => State Diagram

id	$Q_1(t)$	$Q_0(t)$	x	$T_1(t)$	$T_0(t)$	$Q_1(t+1)$	$Q_0(t+1)$	y
0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	0
2	0	1	0	1	0	1	1	0
3	0	1	1	1	0	1	1	0
4	1	0	0	0	0	1	0	0
5	1	0	1	1	1	0	1	0
6	1	1	0	1	0	0	1	1
7	1	1	1	1	1	0	0	1

State Assignment

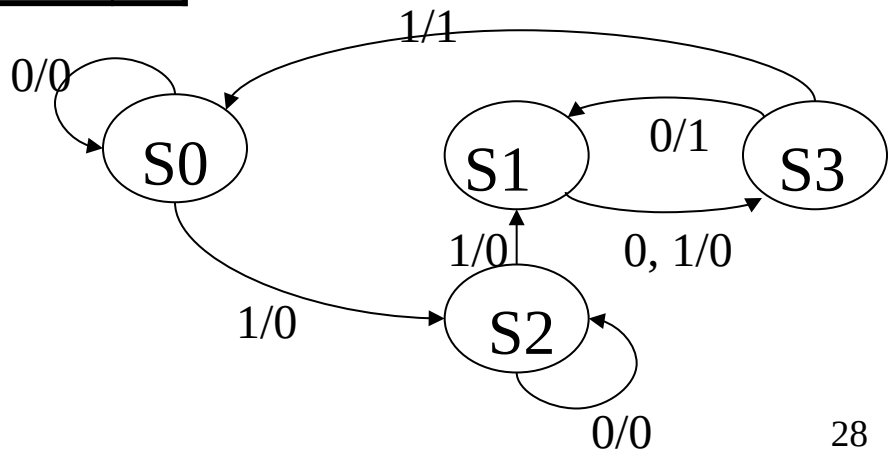
S0 00

S1 01

S2 10

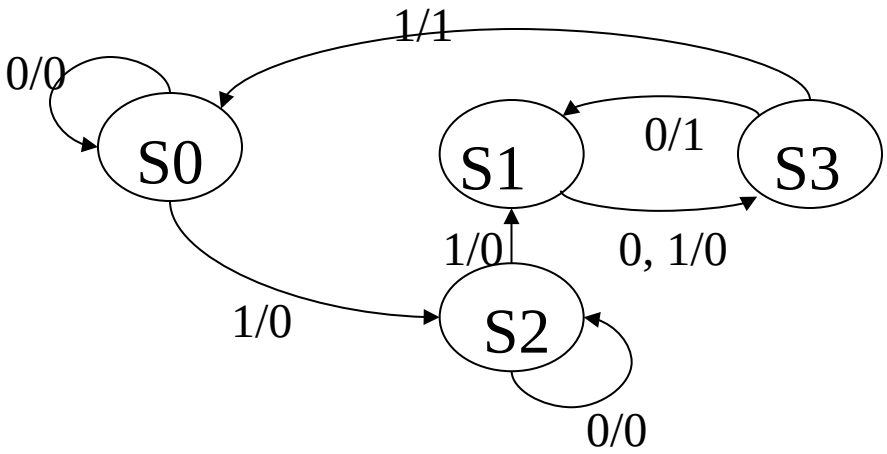
S3 11

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



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	S1,0	S0,1

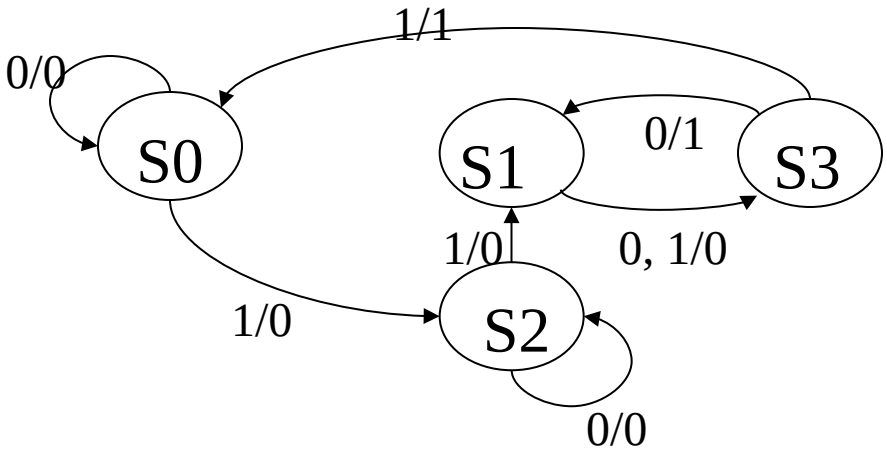


Example: Output sequence

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	S1,0	S0,1



Example: Output sequence

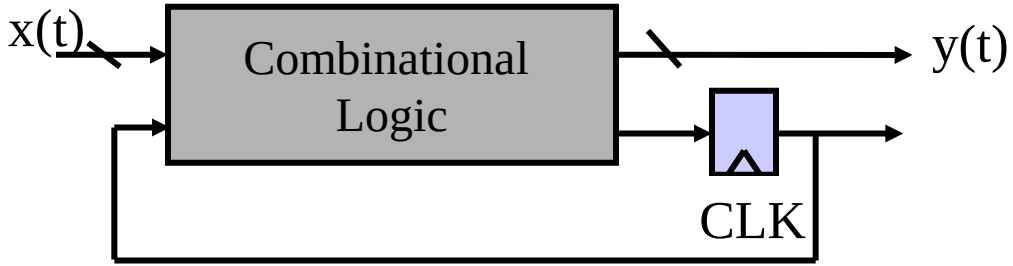
Time	0	1	2	3	4	5
Input	0	1	1	0	1	-
State	S0	S0	S2	S1	S3	S0
Output	0	0	0	0	1	0

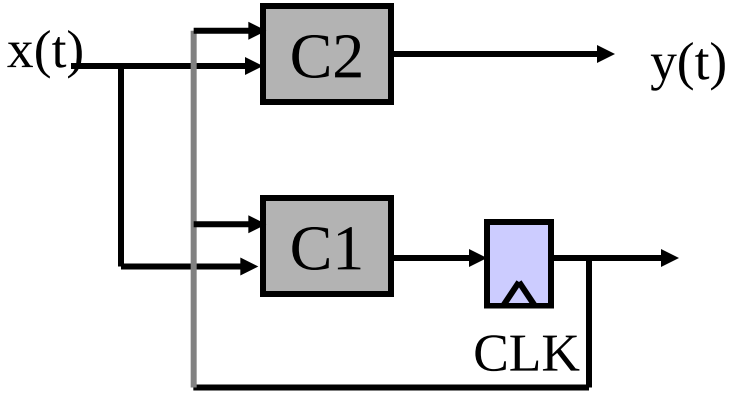
Implementation

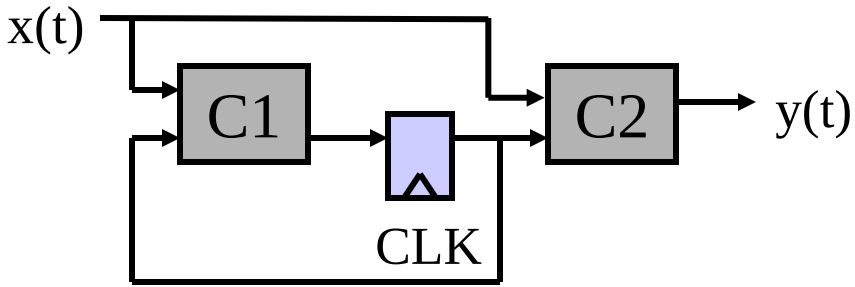
State Diagram => State Table => Logic Diagram

- Canonical Form: Mealy and Moore Machines
- Excitation Table
  - Truth Table of the F-F Inputs
  - Boolean algebra, K-maps for combinational logic
- Examples
- Timing

Canonical Form: Mealy and Moore Machines



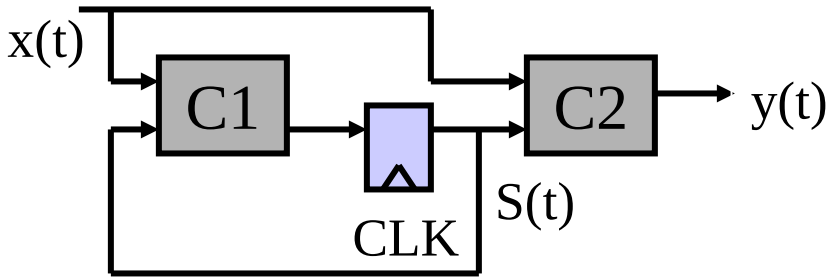




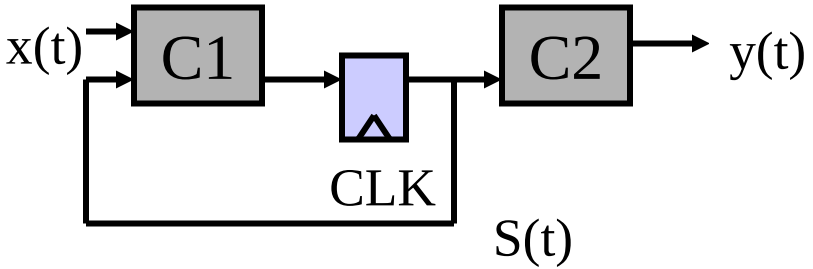
Canonical Form: Mealy and Moore Machines

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

Life on Mars?

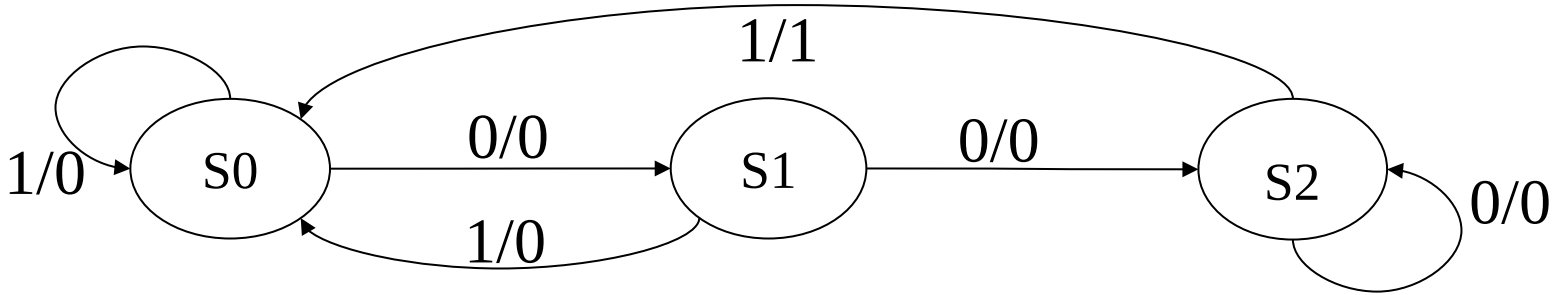
Mars rover has a binary input x. When it receives the input sequence x(t-2, t) = 001 from its life detection sensors, it means that the it has detected life on Mars ☺ and the output y(t) = 1, otherwise y(t) = 0 (no life on Mars ☹).

Implement the Life-on-Mars Pattern Recognizer!

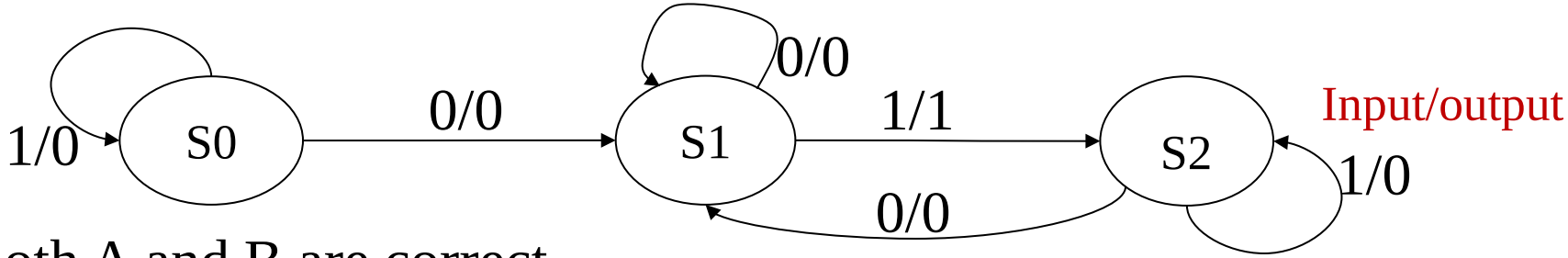
Mars Life Recognizer FSM

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

A.



B.

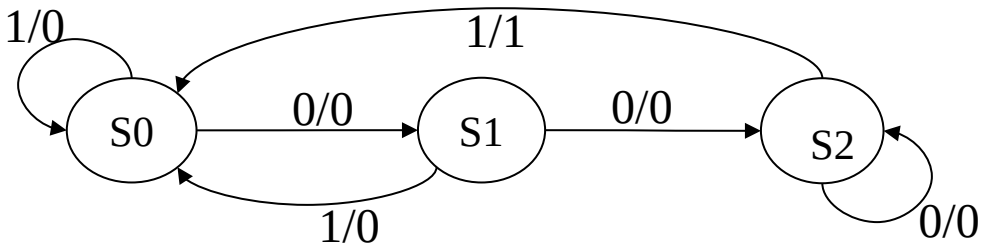


C. Both A and B are correct

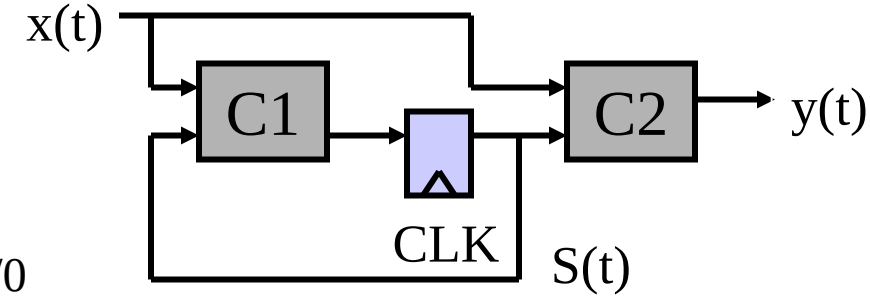
D. None of the above

Mars Life Recognizer FFs

Pattern Recognizer ‘001’



Mealy Machine



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

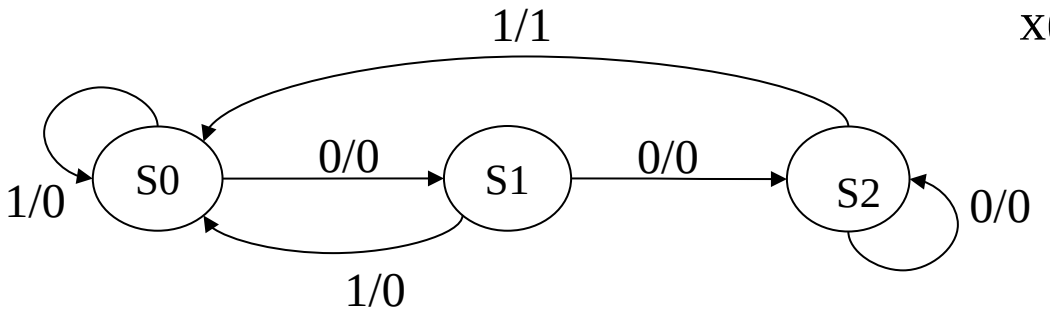
A. next state S(t+1) vs. input x(t), and present state S(t)

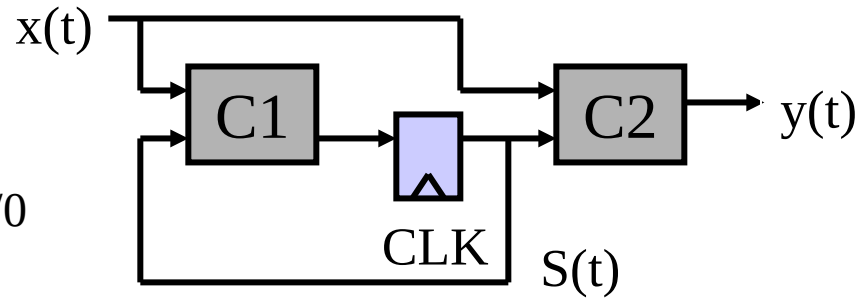
B. output y(t) vs. input x(t), and present state S(t)

C. output y(t) vs. present state S(t)

D. None of the above

State Diagram => State Table with State Assignment





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

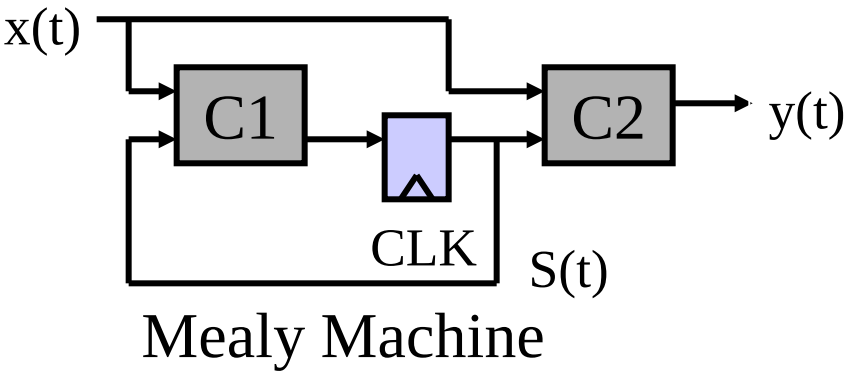
S2: 10

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

$Q_1(t)\ Q_0(t)\backslash x$	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	X	X	X
7	111	X	X	X

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

$D_1(t)$:

x(t) \ Q_0	$\overline{Q_0}$	Q_0
$\overline{Q_1}$	0 (0)	1 (2)
Q_1	0 (1)	0 (3)

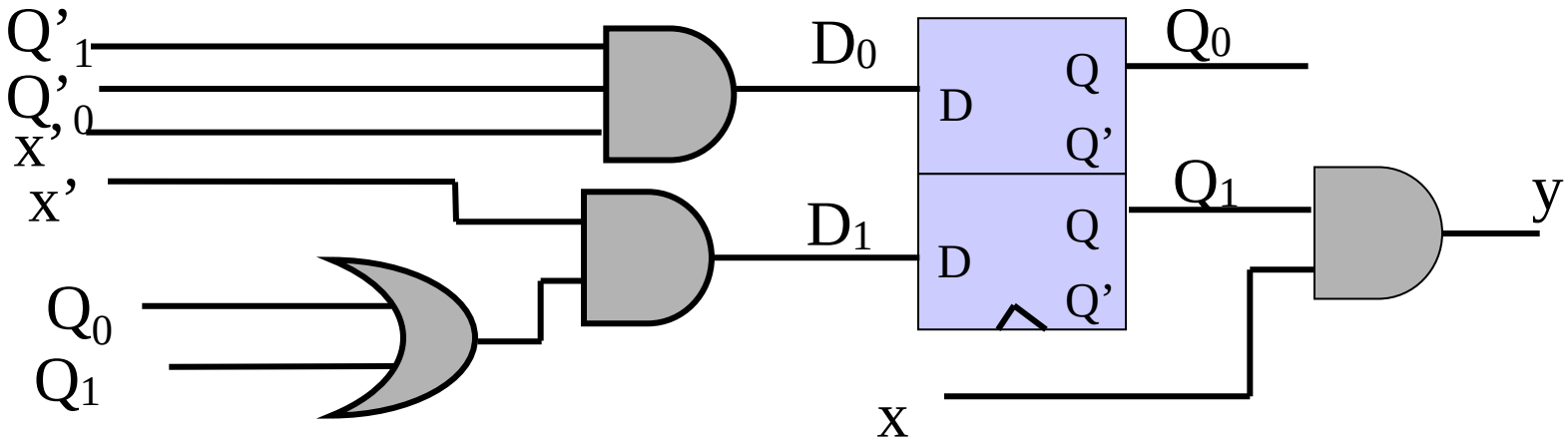
x(t) \ Q_0	$\overline{Q_0}$	Q_0
$\overline{Q_1}$	X (6)	1 (4)
Q_1	X (7)	0 (5)

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

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

$y = Q_1x$

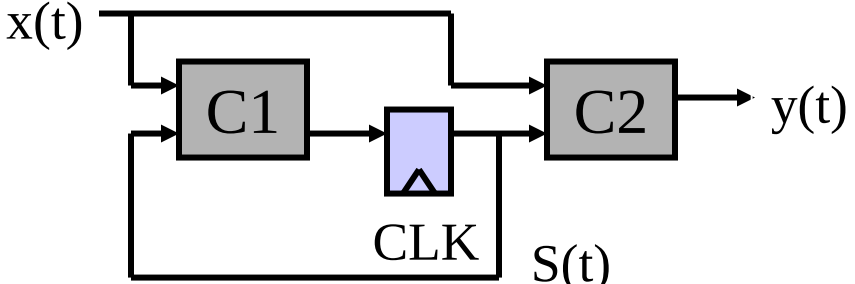
State Diagram => State Table => Excitation Table => Circuit



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

$D_0(t) = Q'_1Q'_0\,x'$

$y = Q_1x$

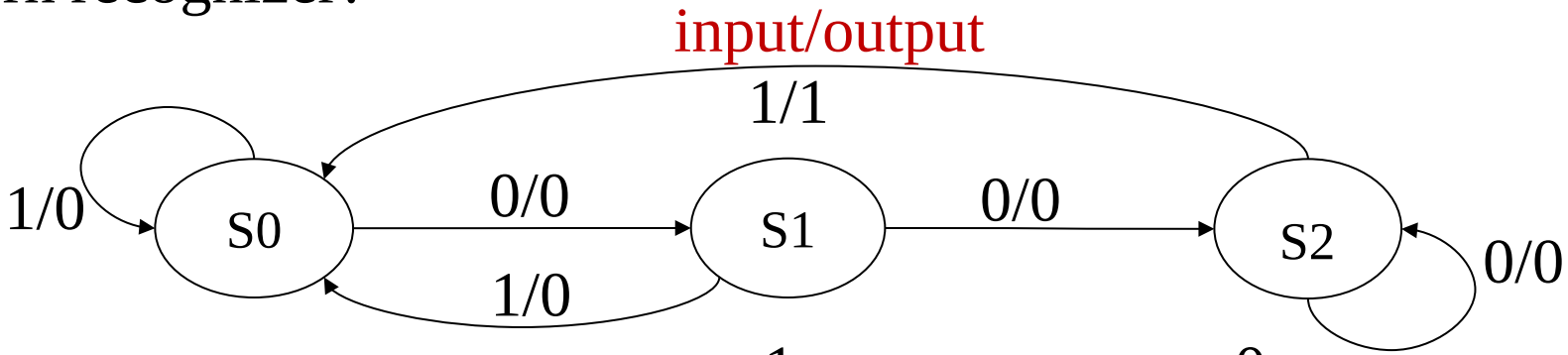


Mealy Machine

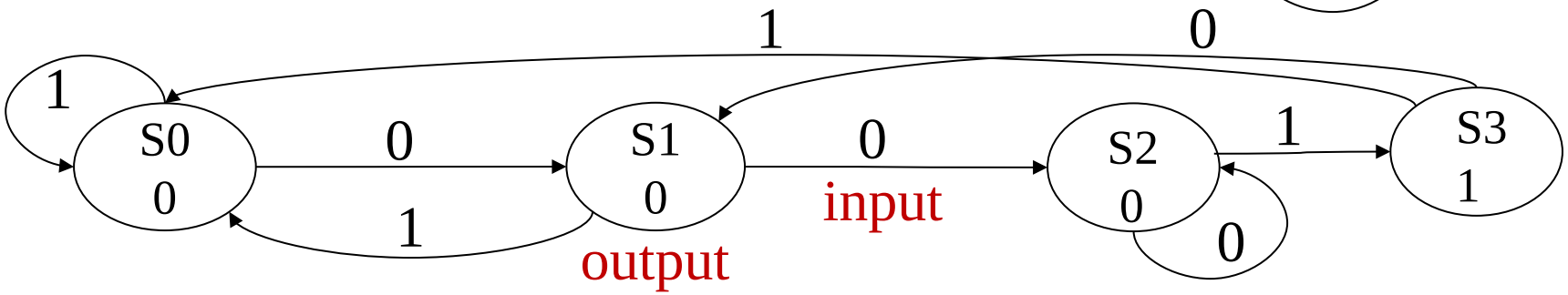
Moore FSM for the Mars Life Recognizer

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

A.



B.

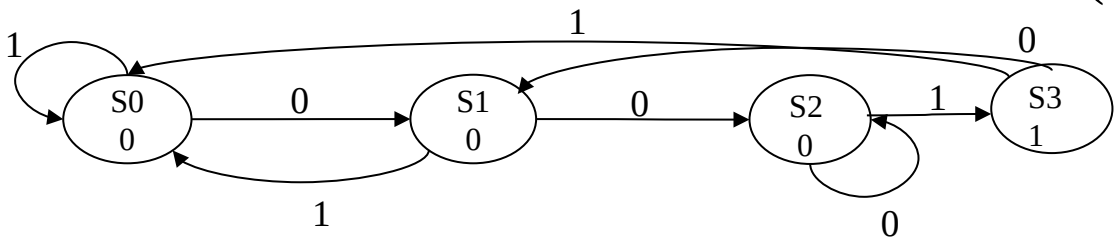


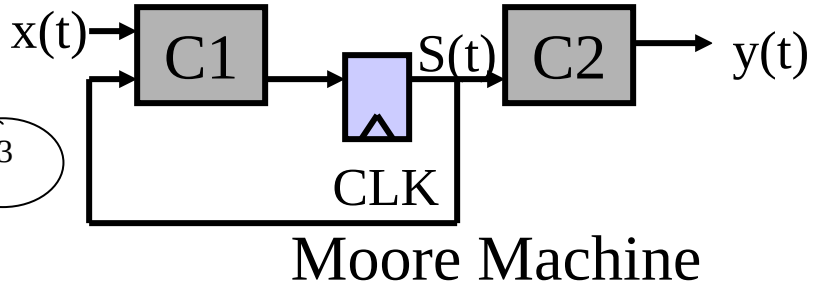
C. Both A and B are correct

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))

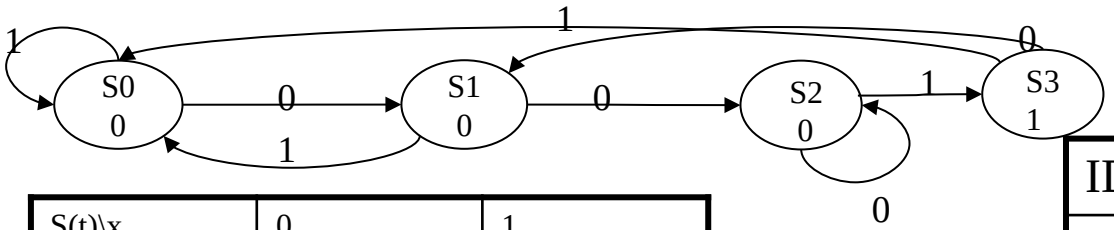
B. (current input, current state vs. current output y(t))

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_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

$D_1(t)$:

x(t) \ Q_1 \ Q_0	$\overline{Q_0}$		Q_0	
	0 0	2 1	6 0	4 1
$\overline{Q_1}$	1 0	3 0	7 0	5 1
Q_1	1 0	3 0	7 0	5 1

$D_0(t)$:

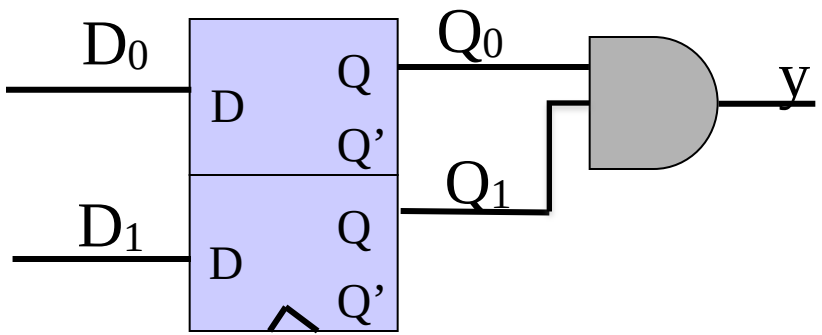
x(t) \ Q_1 \ Q_0	$\overline{Q_0}$		Q_0	
	0 1	2 0	6 1	4 0
$\overline{Q_1}$	1 0	3 0	7 0	5 1
Q_1	1 0	3 0	7 0	5 1

$y(t)$:

x(t) \ Q_1 \ Q_0	$\overline{Q_0}$		Q_0	
	0 0	2 0	6 1	4 0
$\overline{Q_1}$	1 0	3 0	7 1	5 0
Q_1	1 0	3 0	7 1	5 0

Mars Life Recognizer Circuit Implementation

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

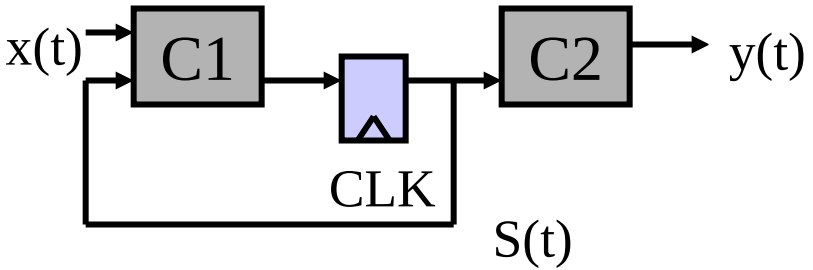


$D_1(t)= Q_1(t)Q_0(t)'+Q_1(t)'Q_0(t)\ x(t)$

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

$\quad Q_1(t)Q_0(t)\ x(t)'+Q_1(t)Q_0(t)'\ x(t)$

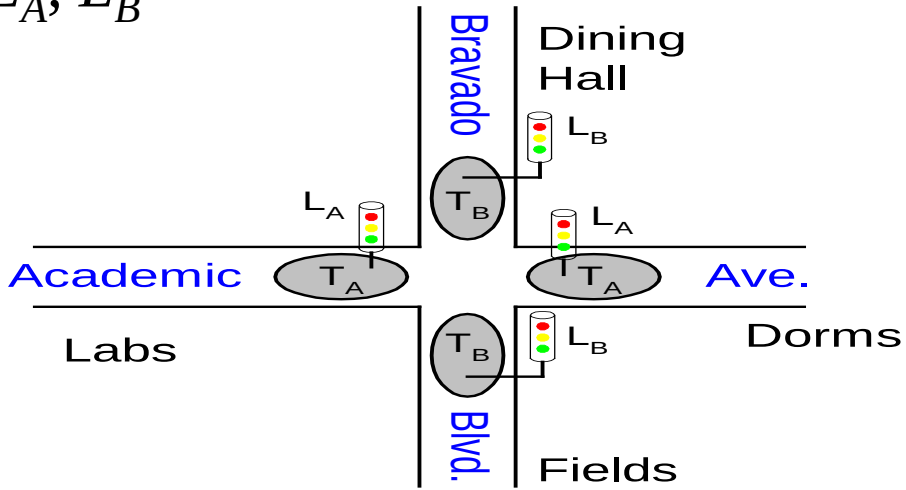
$y(t)= Q_1(t)Q_0(t)$



Moore Machine

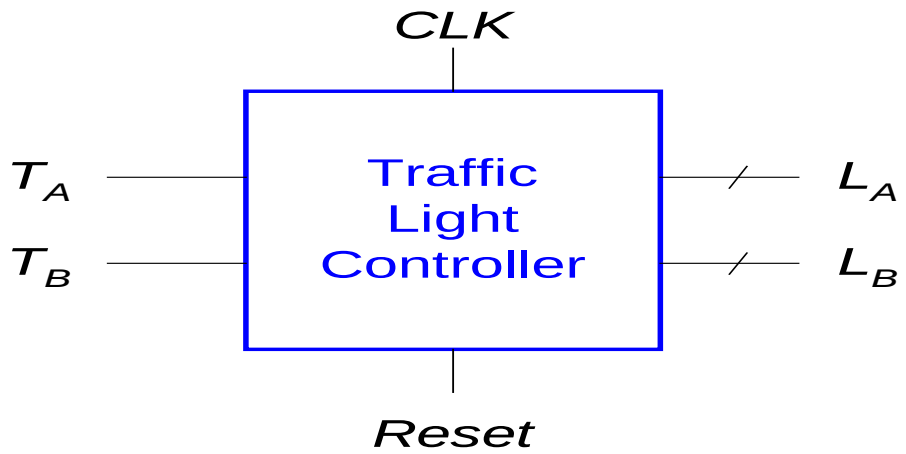
Finite State Machine Example

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



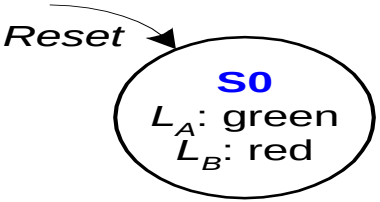
FSM Black Box

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



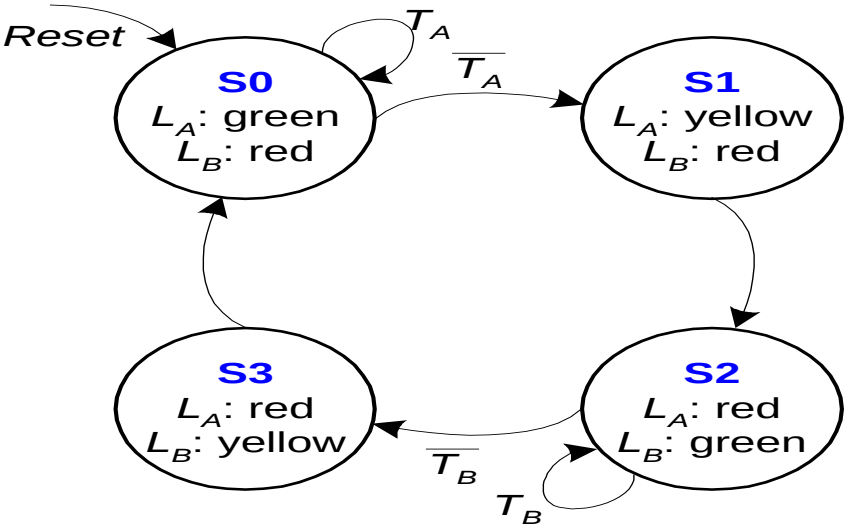
FSM State Transition Diagram

- 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

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

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

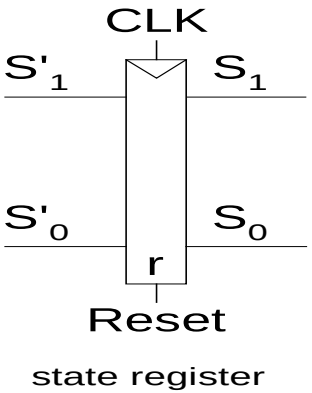
FSM Output Table

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

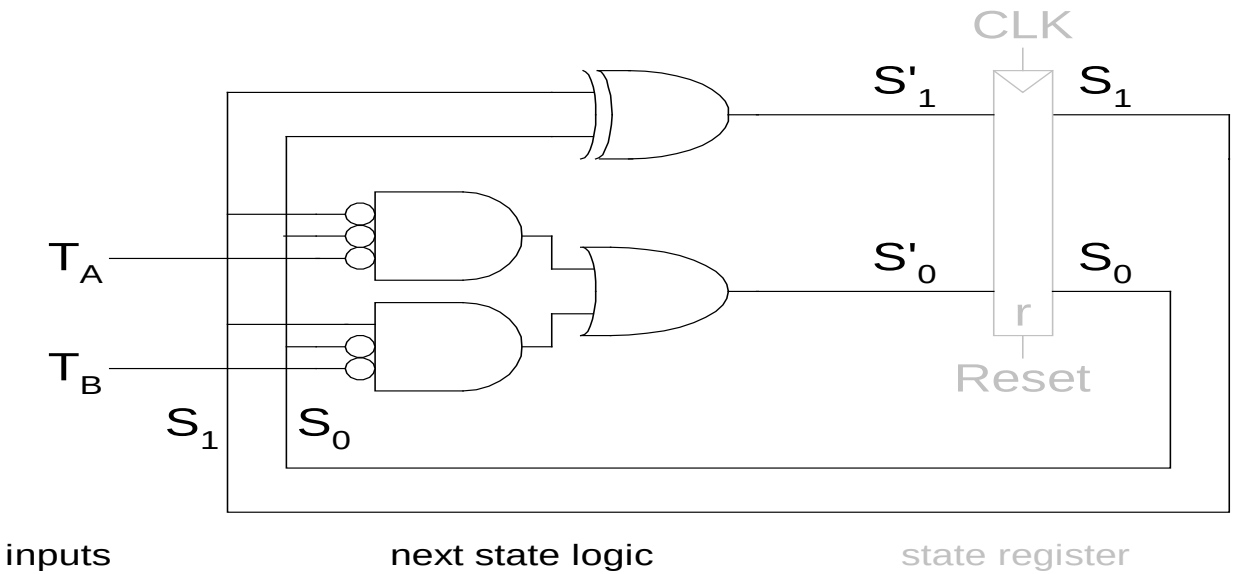
Output	Encoding
green	00
yellow	01
red	10

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

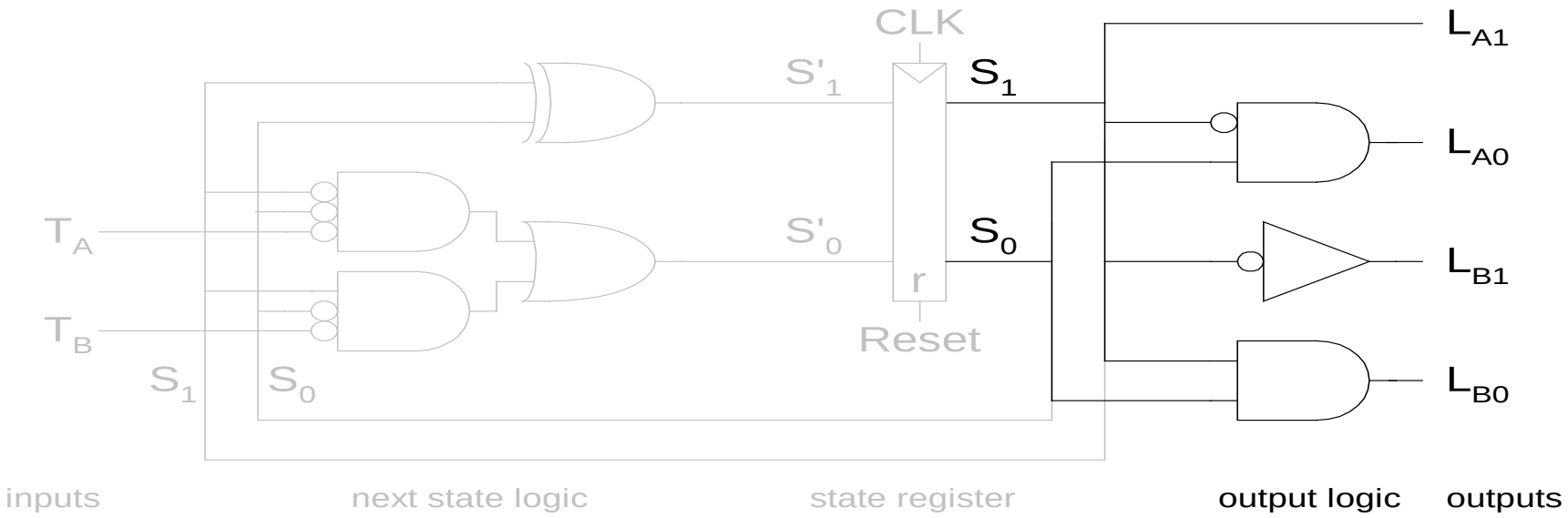
FSM Schematic: State Register



Logic Diagram



FSM Schematic: Output Logic



Summary: Implementation

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