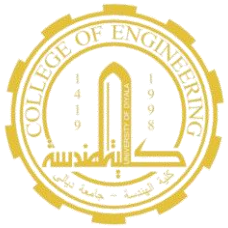
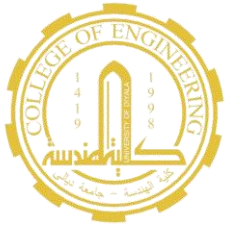
Diyala University
Computer & Software Engineering Department



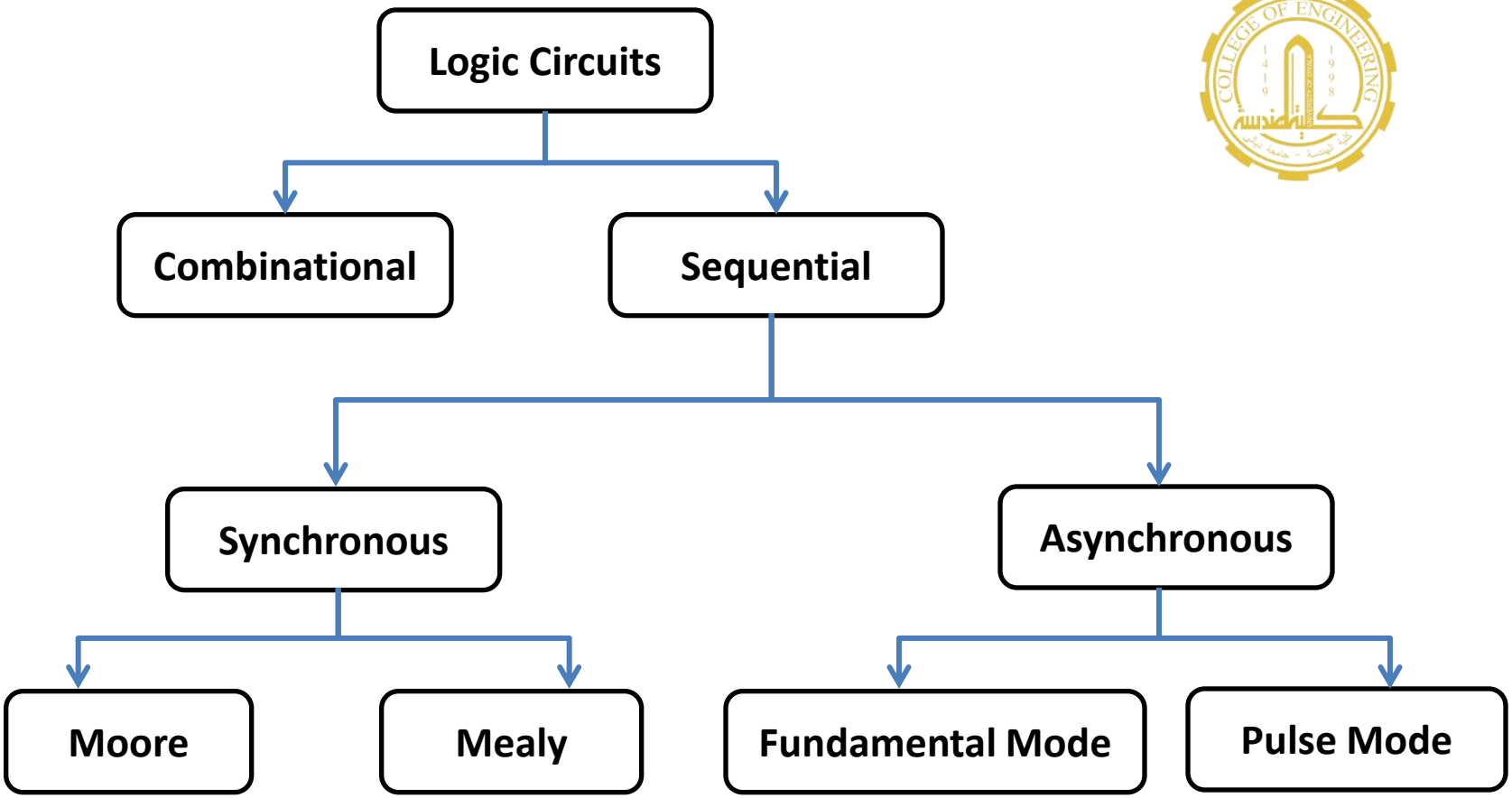
Chapter 4

Sequential Logic Circuits

Types of Logic Circuits



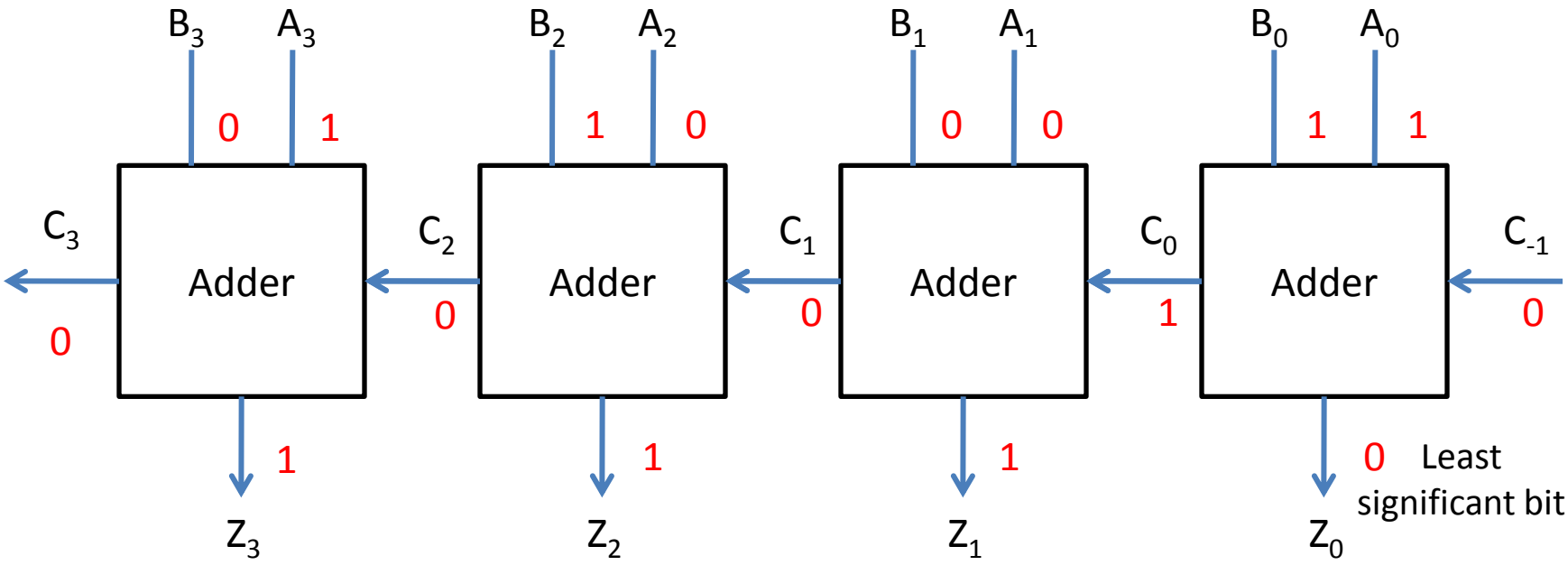
- Combinational logic circuits:
  - **Outputs depend only on its current inputs.**
  - **A combinational circuit may contain an arbitrary number of logic gates and inverters but no feedback loops.**
    - A feedback loop is a connection from the output of one gate to propagate back into the input of that same gate
  - **The function of a combinational circuit represented by a logic diagram is formally described using logic expressions and truth tables.**
- Sequential logic circuits:
  - **Outputs depend not only on the current inputs but also on the past sequences of inputs.**
  - **Sequential logic circuits contain combinational logic in addition to memory elements formed with feedback loops.**
  - **The behavior of sequential circuits is formally described with state transition tables and diagrams.**


Logic Circuits
Combinational
Sequential
Synchronous
Asynchronous
Moore
Mealy
Fundamental Mode
Pulse Mode

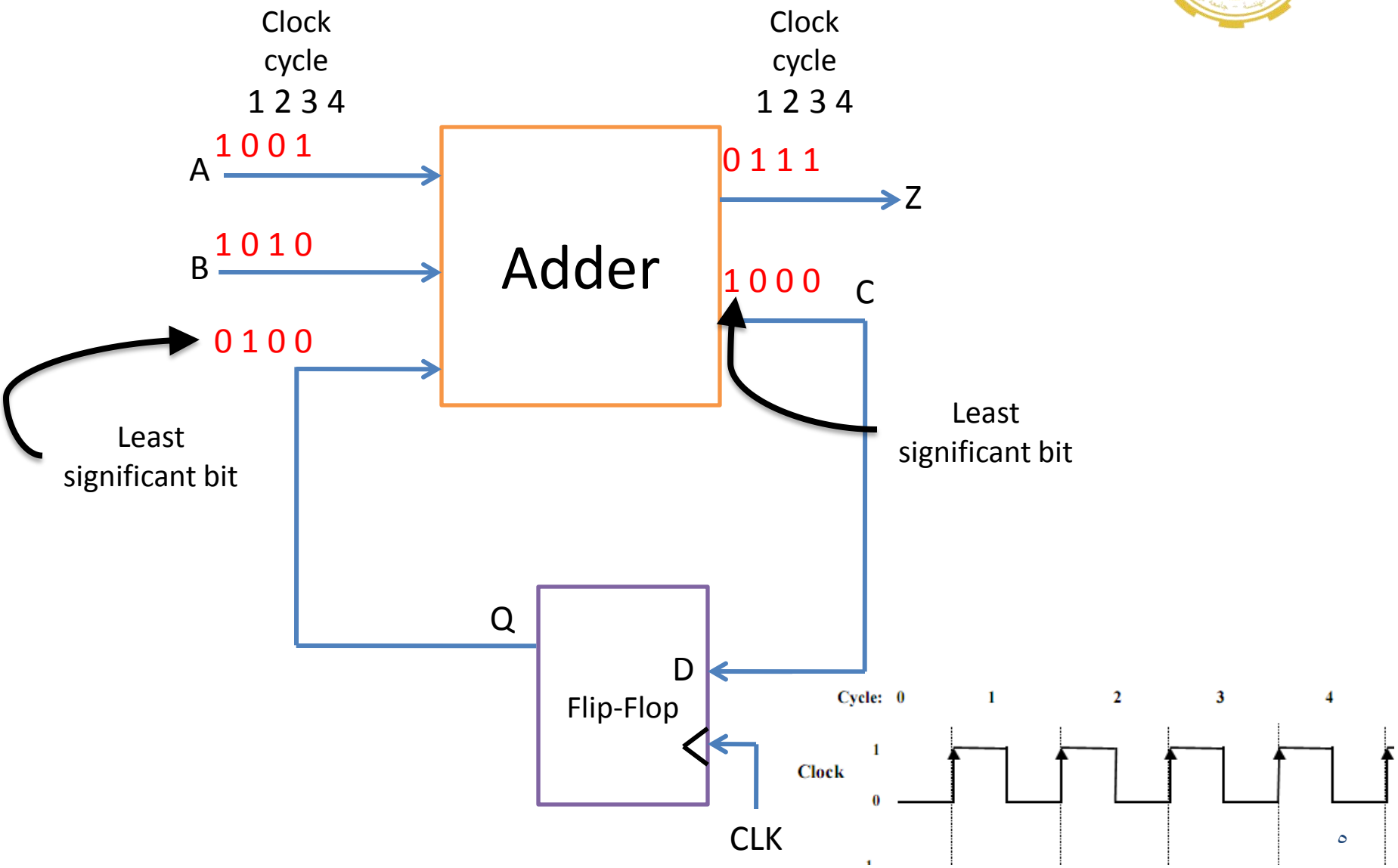
Combinational circuits example:

9+5=14

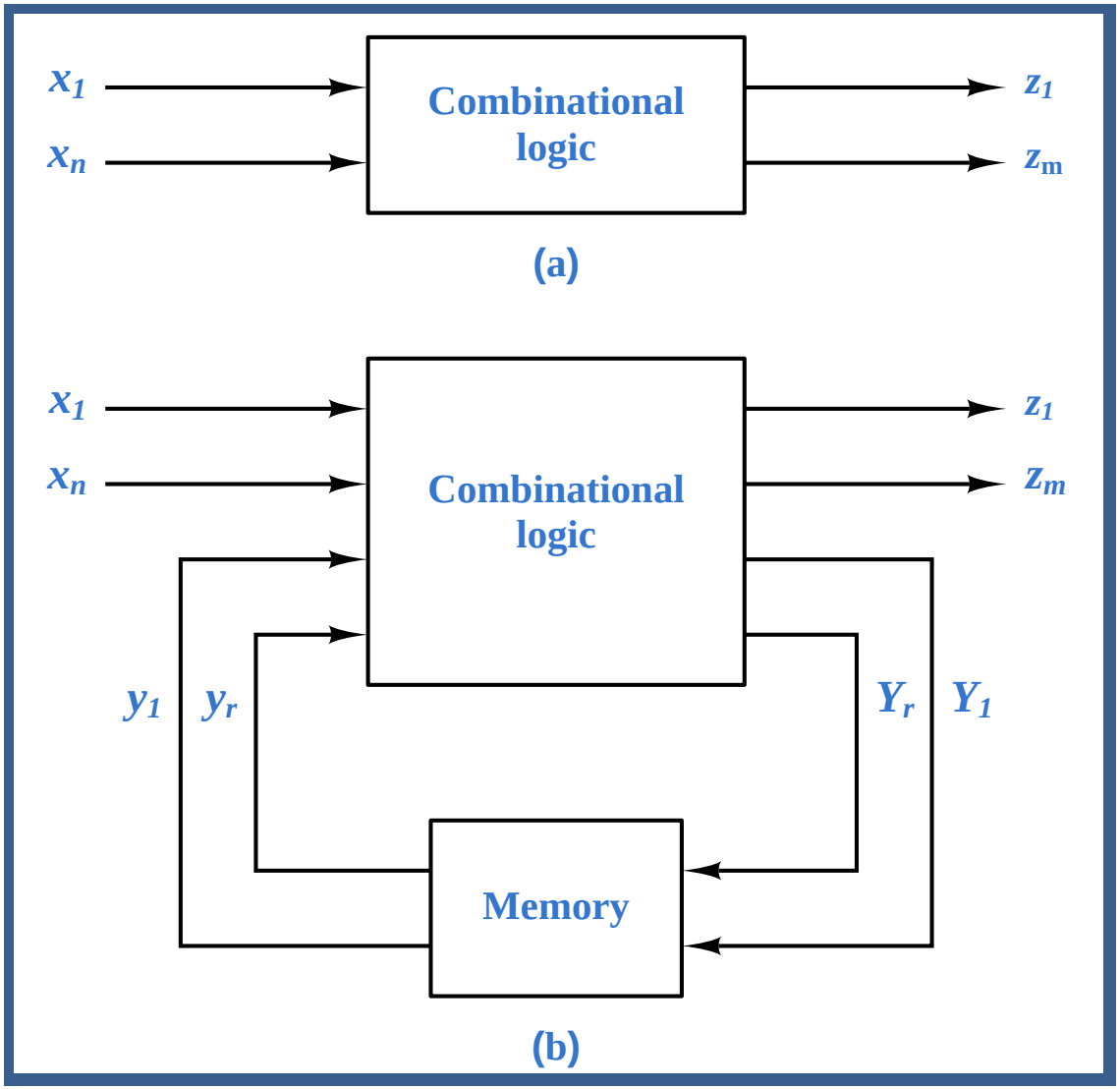
1001+
0101
1110



Sequential circuits example:



The Circuit Model



Truth table for edge triggered flip flop

Inputs			Outputs		Comment
J	K	c	Q	Q’	
0	0	+	Q	Q’	NC
0	1	+	0	1	Reset (store 0)
1	0	+	1	0	Set(store 1)
1	1	+	Q’	Q	Toggle

Inputs			Outputs		Comment
S	R	c	Q	Q’	
0	0	x	Q	Q’	NC
0	1	+	0	1	Reset (store 0)
1	0	+	1	0	Set(store 1)
1	1	+	?	?	Invalid

Truth table for edge triggered flip flop

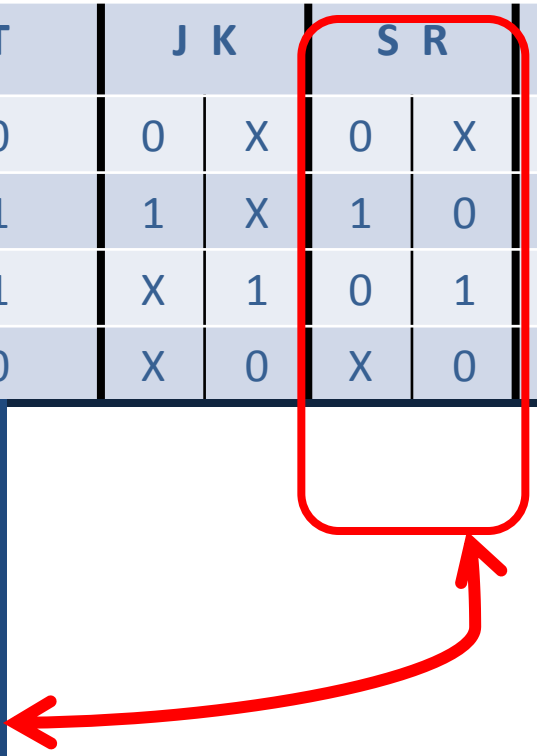
Inputs		Outputs		Comment
D	c	Q	Q’	
1	+	1	0	Set(store 1)
0	+	0	1	Reset (store 0)

Inputs		Outputs		Comment
T	C	Q	Q’	
1	+	Q’	Q	Reset(store 0)
0	+	Q	Q	set (store 1)

Excitation table

State transition		Excitation inputs							
PS (q)	NS (Q)	D	T	J	K	S	R	J	K'
0	0	0	0	0	X	0	X	0	X
0	1	1	1	1	X	1	0	1	X
1	0	0	1	X	1	0	1	X	0
1	1	1	0	X	0	X	0	X	1

present state (PS) output Signal q	External input signals S	R	next state (NS) output signal Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



Synchronous sequential logic circuits

Clocked Synchronous State-Machines

- Such machines have the characteristics:
  - **Sequential circuits designed using flip-flops.**
  - **All flip-flops use a common clock (clocked synchronous).**
  - **A machine using n flip-flops (state memory) has n state variables (the outputs of the flip-flops) and 2^n states.**
  - **In general, the next state and output of the machine both depend on the current state of the machine and on the current input:**

Next state = F(current state, input)

output = G(current state, input)

This type of state machine is called Mealy Machine

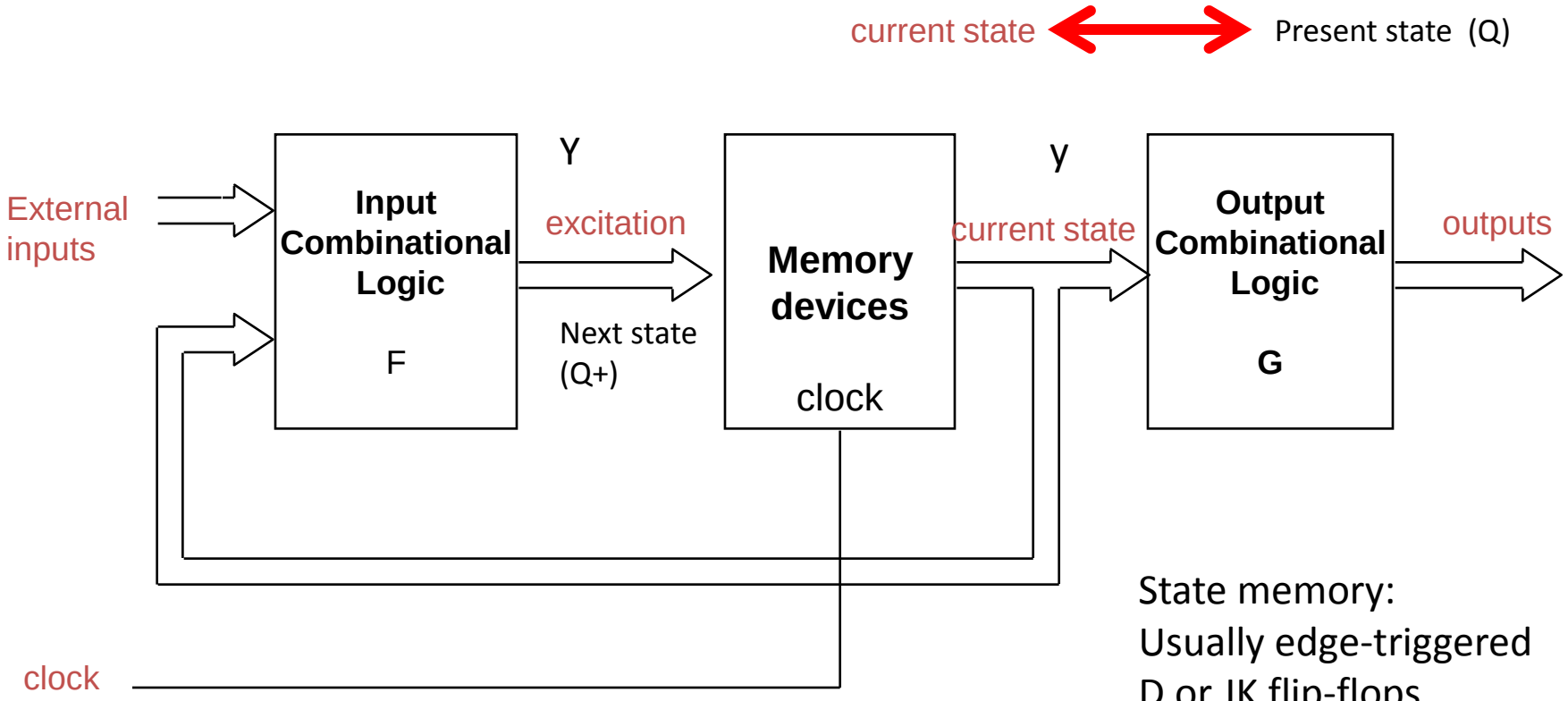
  - **In some cases the next output depends only on the current state and not directly on the current input**

Next state = F(current state, input)

output = G(current state)

Such machines are called Moore machines.

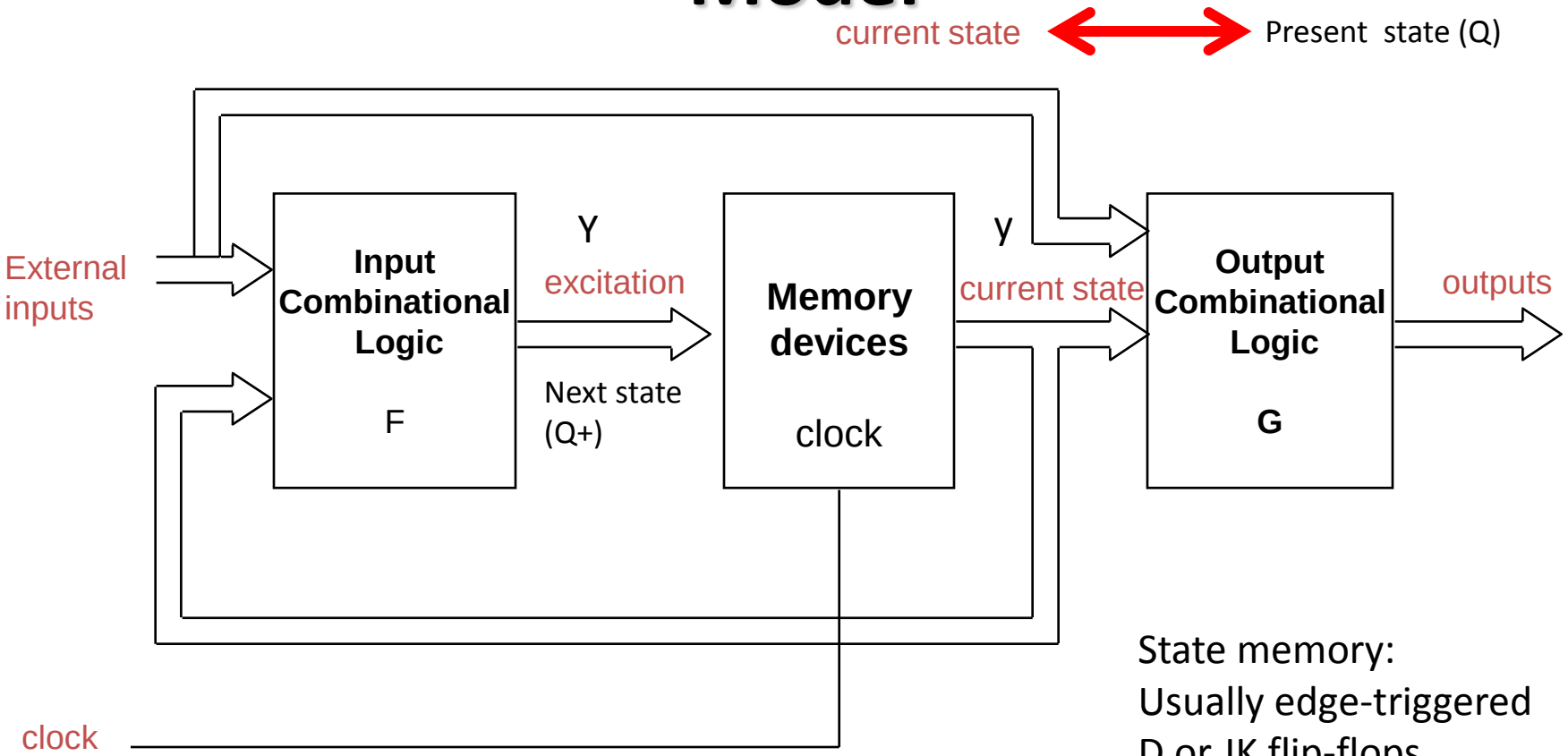
Clocked Synchronous State-Machine Model



State memory:
Usually edge-triggered
D or JK flip-flops

Moore Machine

Clocked Synchronous State-Machine Model



State memory:
Usually edge-triggered
D or JK flip-flops

Mealy machine

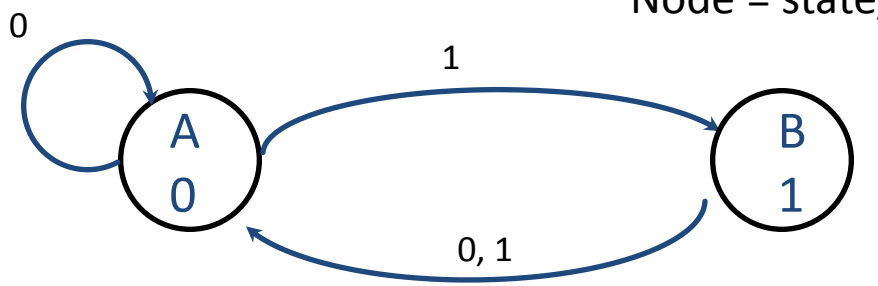
Moore vs. Mealy

Moore	Mealy
Next state = F(current state, input) **Output = G(current state)** **Such machines are called Moore machines.**	**Next state = F(current state, input)** **Output = G(current state, input)** **This type of state machine is called Mealy Machine**
•All state and output transitions occur after falling/rising clock edge (Moore don't have glitch)	**• State transitions occur after falling/rising clock edge (as with Moore machine)** **• Output transitions occur in response to both input and state transition.** **(Mealy may be have glitch)**

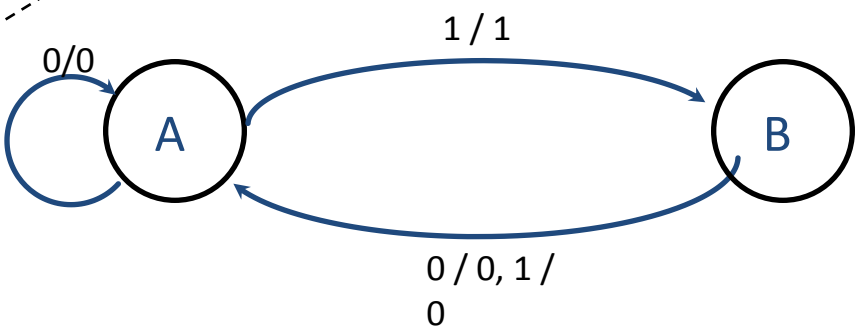
)Moore vs. Mealy (State Diagram

Moore

Format:
Arc = input X
Node = state/output Q

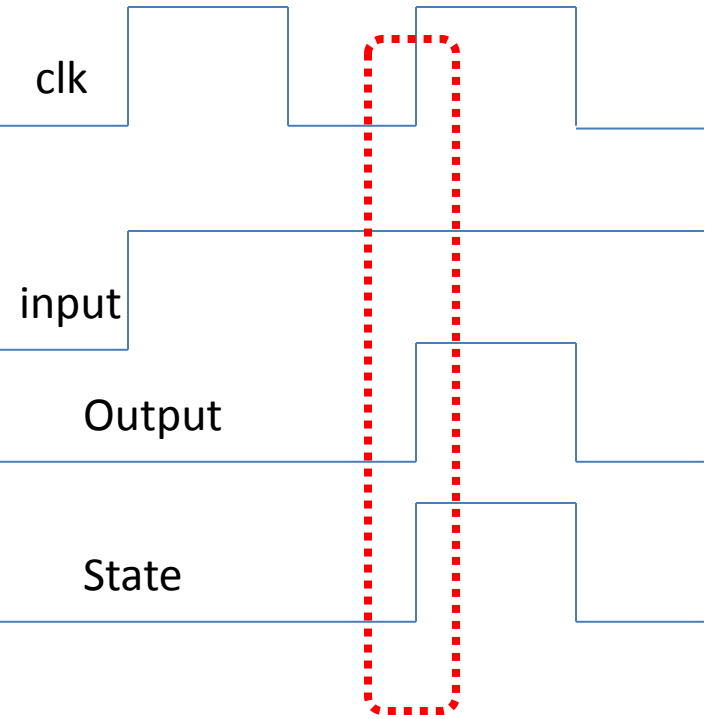


Mealy

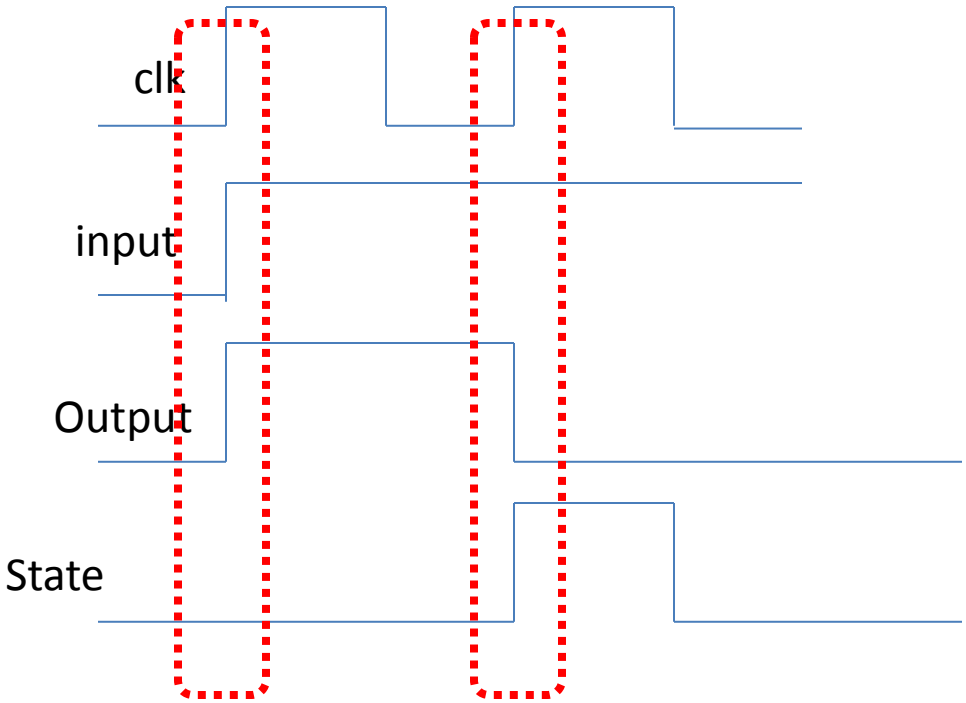


Format:
Arc = input X / mealy output Y
Node = state

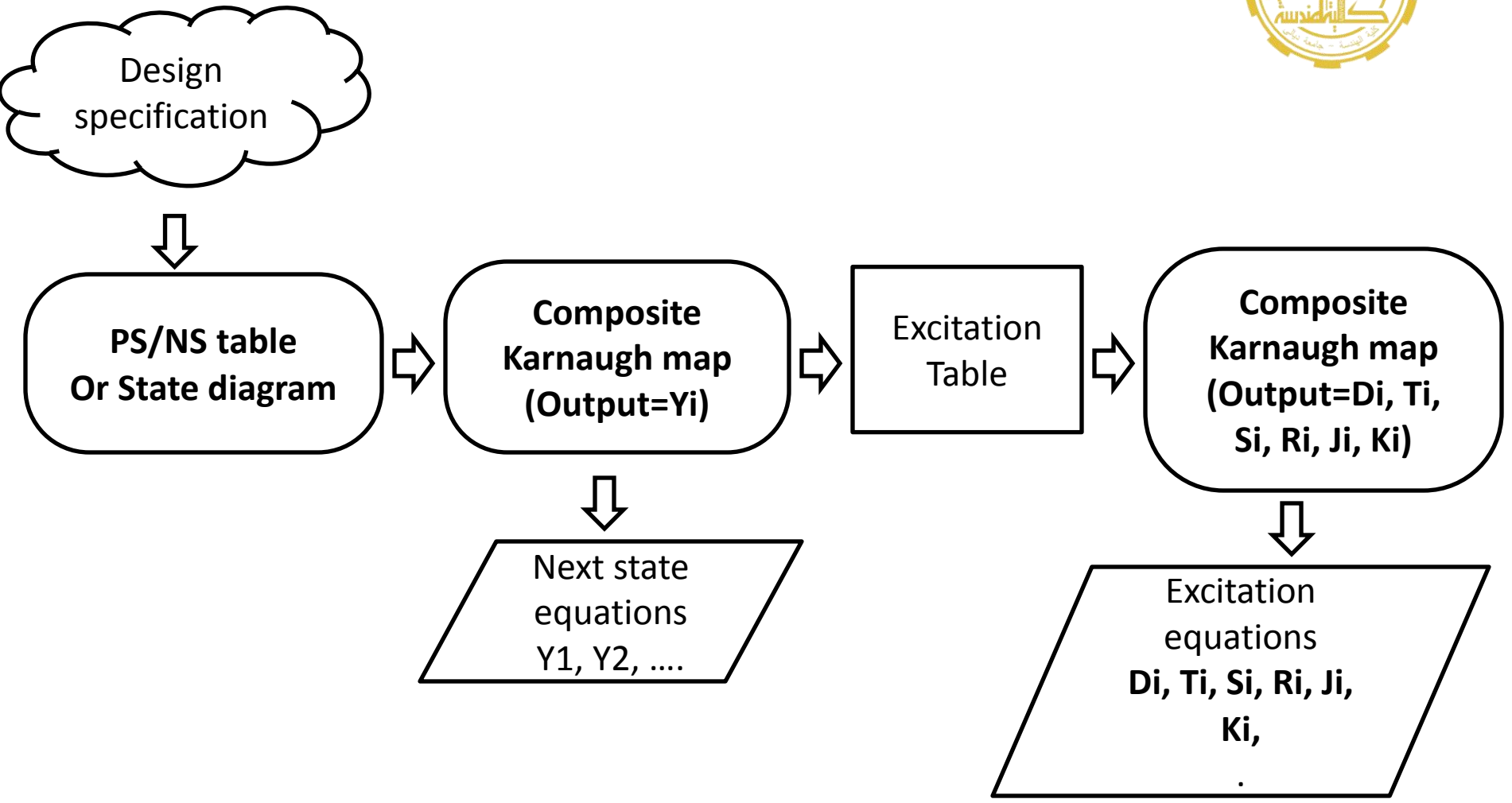
Moore waveform



Mealy waveform



Design procedure



State Machine Design Procedure

Step1: Organize design specifications into a PS/NS table , state diagram, ASM chart, flow map or timing diagram from word description.

Step2: Minimize number of states (optional, can result fewer flip-flops).

Step3: State Assignment: Choose state variables (one variable for each flip-flop) and assign a unique code to each state.

Step 4:Choose flip-flop type (D, J-K, etc.)

- Build excitation table for flip-flop inputs from transition table.
- Derive excitation equations from excitation table.
- Derive output equations from PS/NS table.

Step 5: Draw logic diagram with excitation logic, output logic, and state memory elements.

Simple example 1

The specification :

An idle system is activated when an input, A is given. Then, an output**,** B **is** produced after two interval time or cycles later Next the system will be back to produced after two interval time or cycles later. Next, the system will be back to the idle state, waiting for the next triggering input A **.**

Step 1: Understand the specs

- Get a sample input/ output relationship

E:001001110

Z:000010010

- Draw a simple block diagram

Step 2: Draw state diagram

State table or state assignment

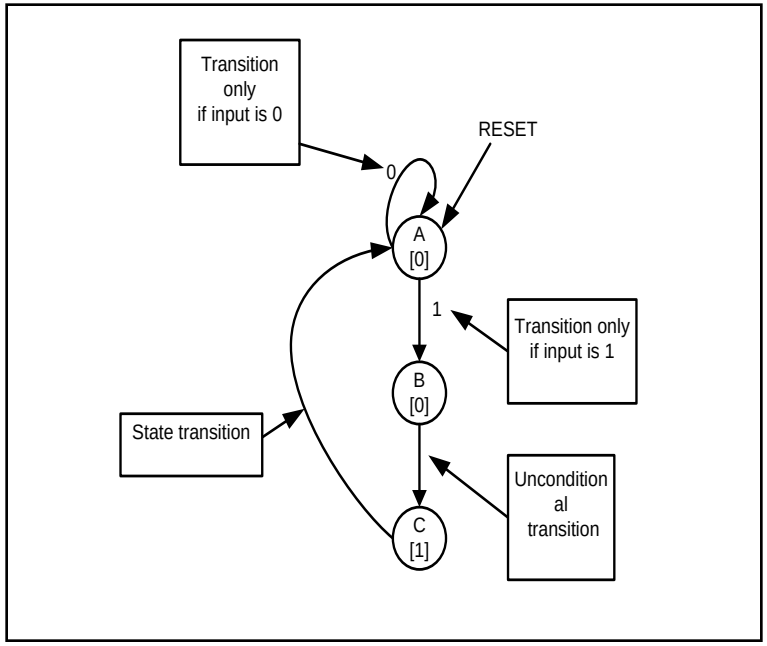
E

S	0	1	Z
A	A	B	0
B	C	C	0
C	A	A	1

E

y2	y1	0	1	Z
0	0	00	01	0
0	1	10	10	0
1	0	00	00	1

State transition diagram 'STD'



Next state &output equation

Present state			Next state		Output
(y2)	(y1)	E	y2+ (Y2)	y1+ (Y1)	Z
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	1	0	0
0	1	1	1	0	
1	0	0	0	0	1
1	0	1	0	0	
1	1	0	X	X	X
1	1	1	X	X	

y2y1 / E

0	1	X	0
0	1	X	0

Y2=y1

y2y1 / E

0	0	X	0
1	0	X	0

$Y1 = y2\ \overline{y1}\ \overline{E}$

y1 / y2

0	0
1	X

OUT=y2=Z

Another method and continue step:

Step 3: get symbolic state table

Present state	Input	Next state	Output
	E		Z
S0	0	S0	0
	1	S1	
S1	0	S2	0
	1	S2	
S2	0	S0	1
	1	S0	

Step 4 : state assignment

Present state	Input	Next state	Output
	E		Z
S0=00	0	00	0
	1	01	
S1=01	0	10	0
	1	10	
S2=10	0	00	1
	1	00	
S3=11	0	11	x
	1	11	

Next state &output equation

Present state		Input	Next state		Output
y2	y1	E	Y2	Y1	Z
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	1	0	0
0	1	1	1	0	
1	0	0	0	0	1
1	0	1	0	0	
1	1	0	X	X	x
1	1	1	X	X	

y2y1 / E

0	1	X	0
0	1	X	0

Y2=y1

y2y1 / E

0	0	X	0
1	0	X	0

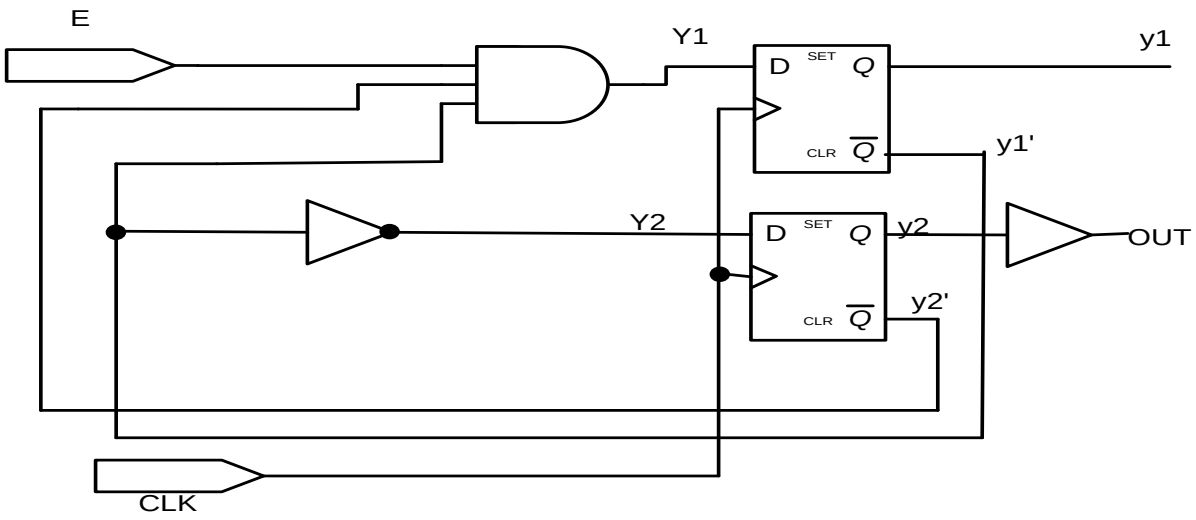
Y1= y2’ y1’ E

y1 / y2

0	0
1	X

Z=y2

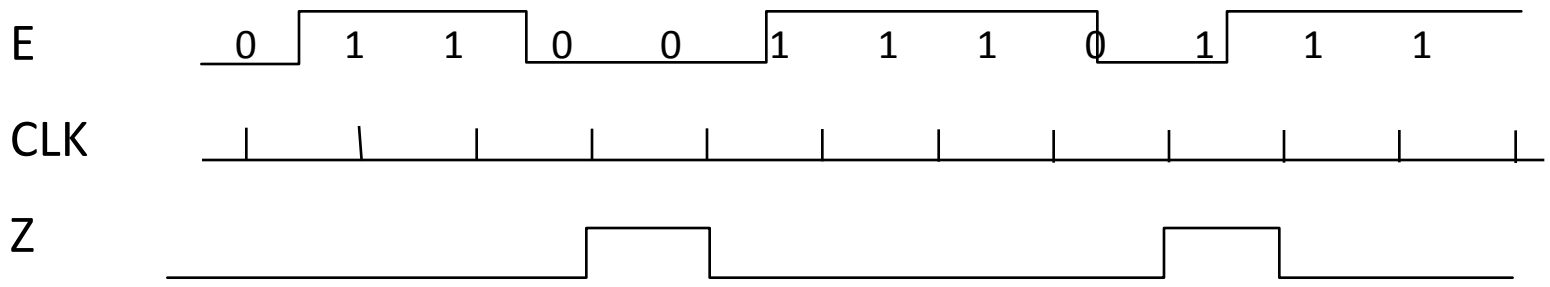
Circuit diagram



Timing diagram : in class

Example 2 State Machine Design :110 Detector " MOORE"

- Word description (110 input sequence detector):
  - **Design a state machine with input" E" and output "Z".**
  - **Z should be 1 whenever the sequence 1 1 0 has been detected on "E" on the last 3 consecutive rising clock edges (or ticks).**
  - **Otherwise, Z = 0**
  - **Note: this is a Moore machine, that is the output, Z, depends only on inputs at previous clocks rising edges , not on the current input.**
- Timing diagram interpretation of word description (only rising clock edges are shown):



State Machine Design Example 2: 110 Detector

Step1: Choosing States

- Possible states (What does the state machine need to remember?):
  - **Initial : power up, no clocks yet** **Z = 0**
  - **A : first 1 not found** **Z = 0**
  - **B : first 1 found** **Z = 0**
  - **C : at least 2 consecutive 1s found** **Z = 0**
  - **D : found 1 1 0** **Z = 1**

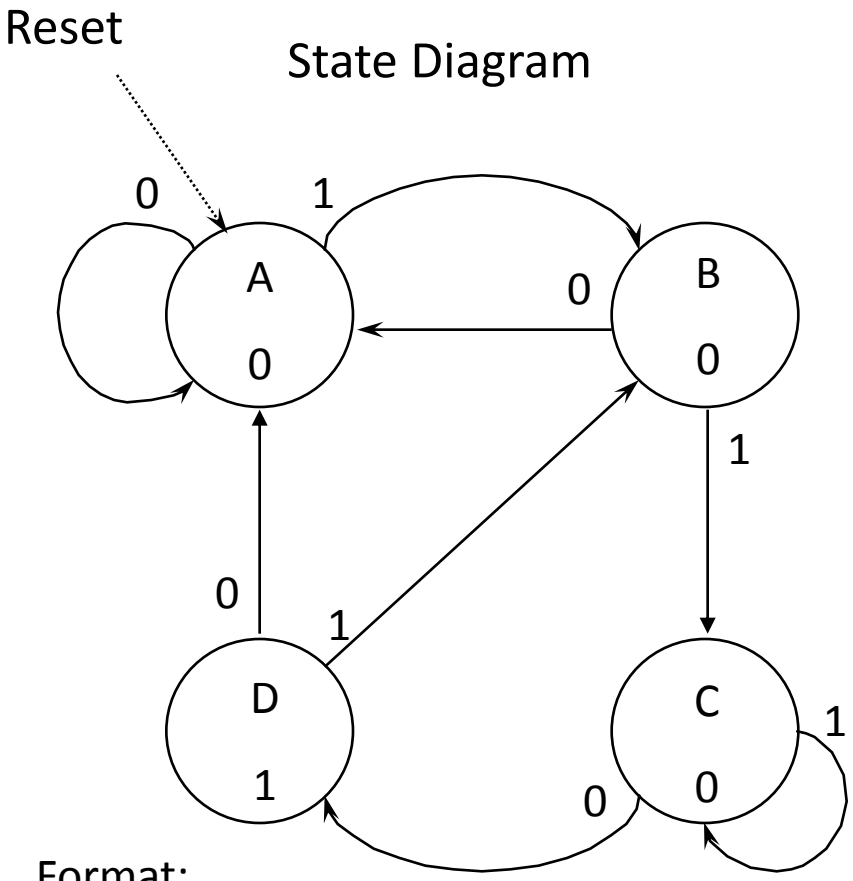
- Are all the states needed?
  - **Notice: Initial is equivalent to A**
  - **We can drop the state Initial and replace it with state A**

State Machine Design Example 2: 110 Detector

Step 1: State/Output Table and Diagram

State Table

S	E = 0	E = 1	Z
A	A	B	0
B	A	C	0
C	D	C	0
D	A	B	1
	S*		



Format:
Arc: input E
Node: state/output Z

State Machine Design Example 2: 110 Detector

Step 3: State Assignment

- Choose state variable assignments:
  - Initial state all 0s
  - Q2 = last E, so Q2* = E
  - minimize number of transitions

Q1	Q2	S	E = 0	E = 1	Z
0	0	A	A	B	0
0	1	B	A	C	0
1	1	C	D	C	0
1	0	D	A	B	1
			S*		

State Machine Design Example 2: 110 Detector

Step 4: Transition/Output Table

- Step 4: Build transition/output table from state/output table by substituting state variable combinations instead of state names.

Q1	Q2	E = 0	E = 1	Z
0	0	00	01	0
0	1	00	11	0
1	1	10	11	0
1	0	00	01	1
Q1* Q2*				
=D1 D2 ← Step 6				

Q1 Q2	E = 0	E = 1	Z
S0	S0	S1	0
S1	S0	S3	0
S3	S2	S3	0
S2	S0	S1	1
Q1* Q2*			
=D1 D2 ← Step 6			

- Step 4: Choose D Flip-Flops , so Q*= D
- Step 4: Excitation table:
  - **Same as Transition/output table with Q1*=D1, Q2*=D2**

Symbolic state table

Present state	Input	Next States	Output
	E		Z
S0	0	S0	0
	1	S1	
S1	0	S0	0
	1	S3	
S2	0	S0	0
	1	S1	
S3	0	S2	1
	1	S3	

Encoded State table

Present state		Input	Next States		Output
y2	y1	E	Y2	Y1	Z
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	0	0	0
0	1	1	1	1	
1	0	0	0	0	0
1	0	1	0	1	
1	1	0	1	0	1
1	1	1	1	1	

State Machine Design Example 2: 110 Detector
Steps 4: Excitation/Output Equations

E \ y2y1			
0	0	0	1
0	1	0	1

$Y2 = y2\ y1' + y1\ y2'\ E$

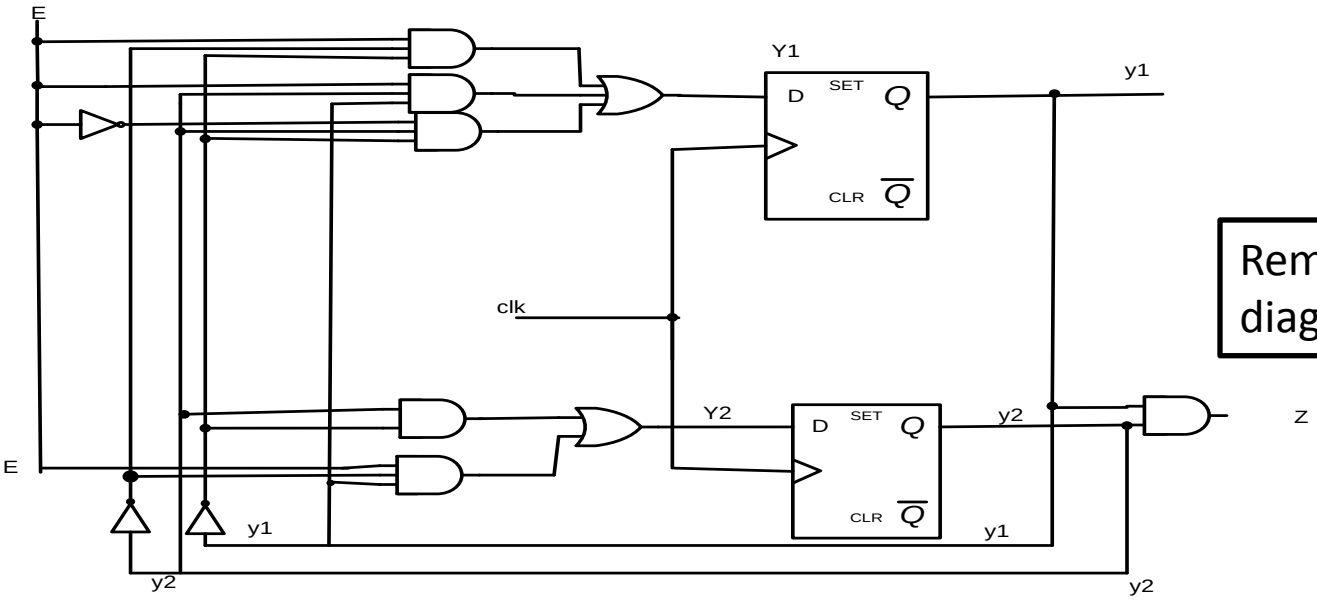
E \ y2y1			
0	0	0	1
1	0	1	0

$Y1 = y2'\ y1'\ E + y2\ y1'\ E' + y2\ y1\ E$

y2 \ y1	
0	0
0	1

$Z = y2y1$

Step 5: Logic Diagram 'MOORE'

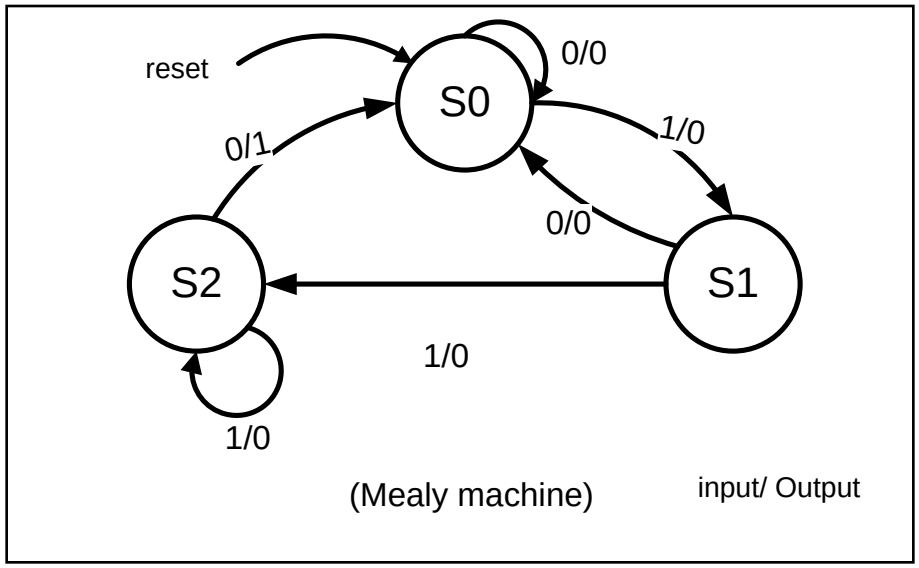


Remember the time diagram

P = Preset
C = Clear
Both active low

Example 2 State Machine Design :110 Detector " MEALY "

From specification example 2: understand the problem,
S0 means 0 bit found, S1=1 bit found, S2=bits found .
If all the third bit is detected (110) completed while S2 reset go to S0 while at the same time outputting a 1



Present state	Input	Next States	Output
	E		Z
S0	0	S0	0
	1	S1	0
S1	0	S0	0
	1	S2	0
S2	0	S0	1
	1	S2	0

State assignment table

Present state		Input	Next States		Output
y2	y1	E	Y2	Y1	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	X	X	X
1	1	1	X	X	X

HOME WORK : circuit diagram

y2y1 / E

0	0	X	0
0	1	X	1

$Y2 = y2\,E + y1\,E$

y2y1 / E

0	0	X	0
1	0	X	0

$Y1 = \overline{y2}\ \overline{y1}\ E$

y2y1 / E

0	0	X	1
0	0	X	0

$Z = y2\ E$

Example: “Moore”

Design sequence logic circuit has one input, E , and one output, Z. All changes in the circuit occur on the positive edge of a clock signal. The output Z is equal to 1 if during two immediately preceding clock cycles the input E was equal to 1. Otherwise, the value of Z is equal to 0.

Step 1: understand the question

Thus, the circuit detects if two or more consecutive 1s occur on its input E. Circuits that detect the occurrence of a particular pattern on its inputs are referred to as sequence detectors.

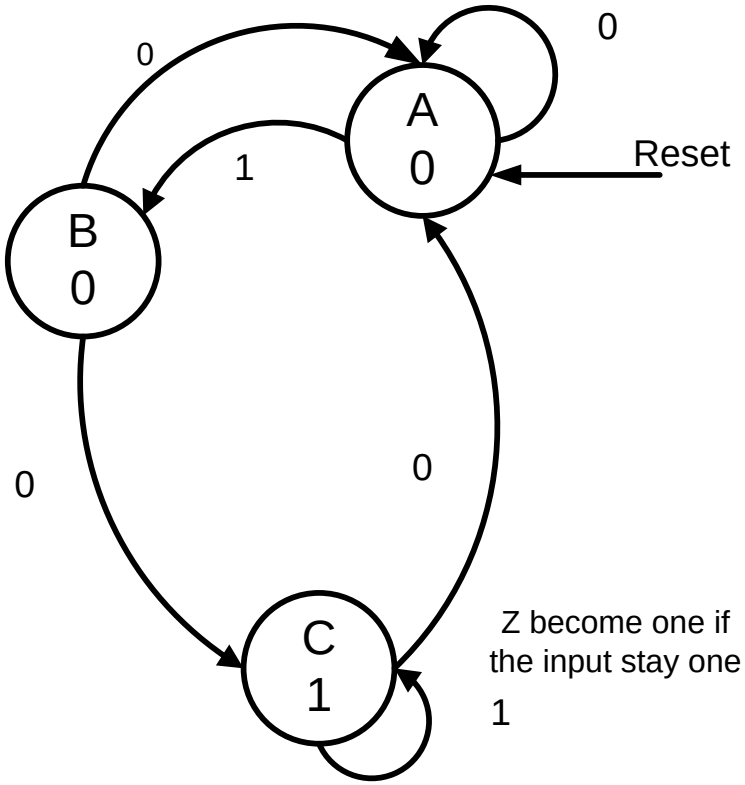
Step 2: State Diagram

Clock cycle: 1 2 3 4 5 6 7 8 9 10 11

E : 0 1 0 1 1 0 1 1 1 0 1

Z : 0 0 0 0 0 1 0 0 1 1 0

	E		
S	0	1	Z
A	A	B	0
B	A	C	0
C	A	C	1



State assignment table

Present state	Input	Next States	Output
	E		Z
S0	0	S0	0
	1	S1	
S1	0	S0	0
	1	S2	
S2	0	S0	1
	1	S2	
S3	0	S3	X
	1	S3	

Present state		Input	Next States		Output
y2	y1	E	Y2	Y1	Z
0	0	0	0	0	0
0	0	1	0	1	
0	1	0	0	0	0
0	1	1	1	0	
1	0	0	0	0	1
1	0	1	1	0	
1	1	0	X	X	X
1	1	1	X	X	

Output equation

E \ y2y1	00	01	11	10
0	0	0	X	0
1	0	0	X	0

Y1=y2'y1' E

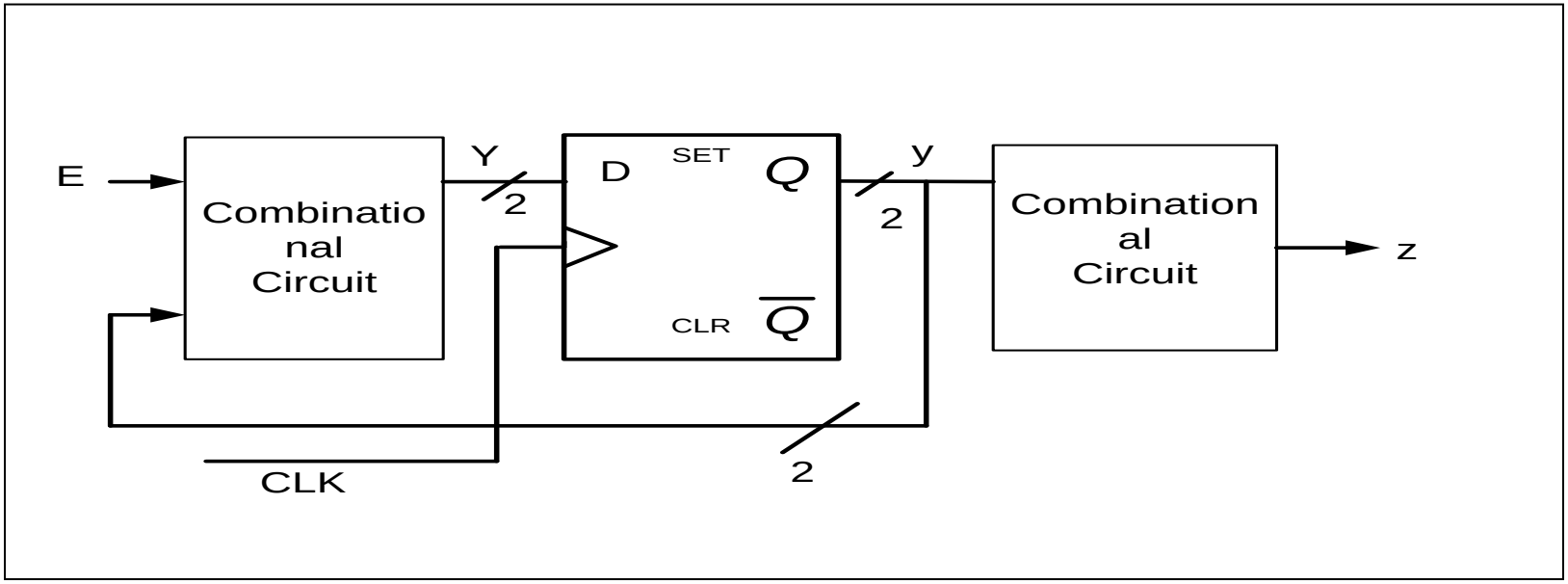
E \ y2y1	00	01	11	10
0	0	0	X	0
0	1	1	X	1

Y2= E(y1+y2)

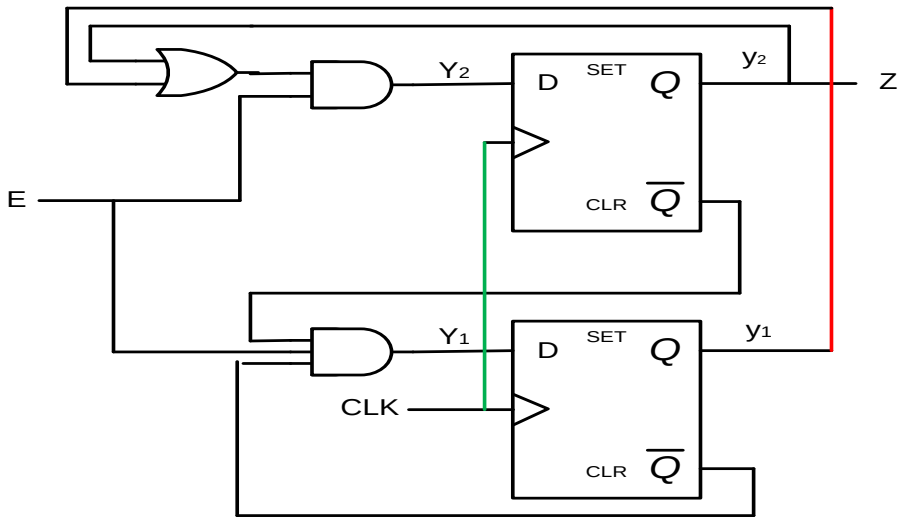
y2 \ y1	0	1
0	0	X
1	0	X

Z=y2

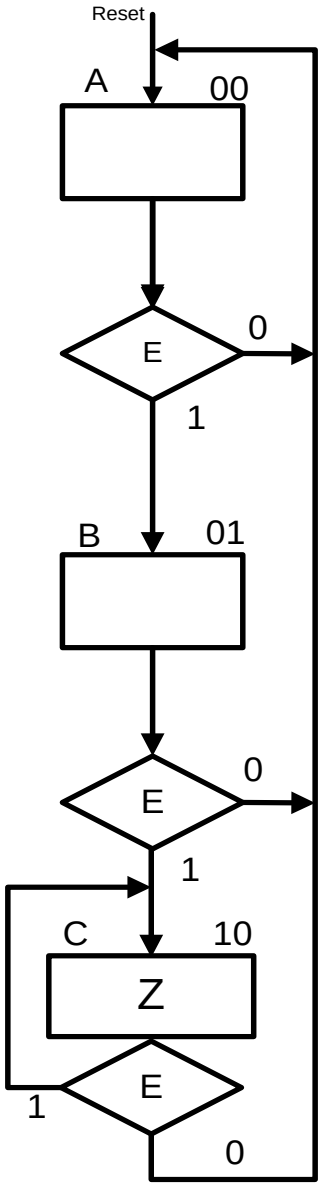
Function block diagram



Logic circuit diagram



Flow chart :

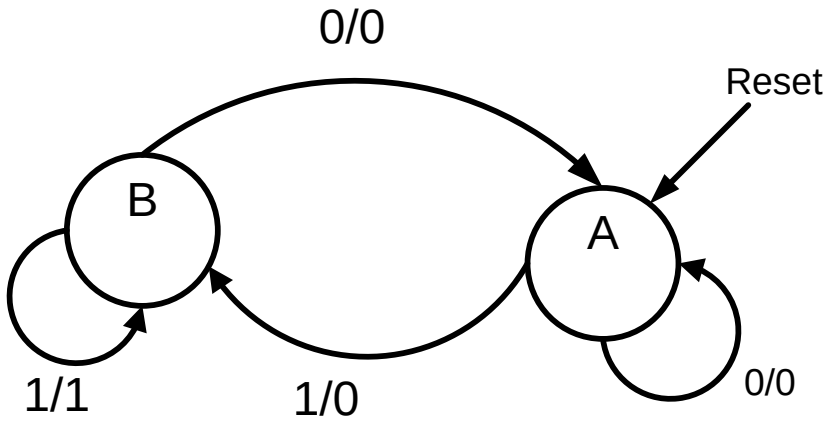


Same example : Mealy

Clock cycle:	1	2	3	4	5	6	7	8	9	10	11
E :	0	1	0	1	1	0	1	1	1	0	1
Z :	0	0	0	0	1	0	0	1	1	0	0

Moore

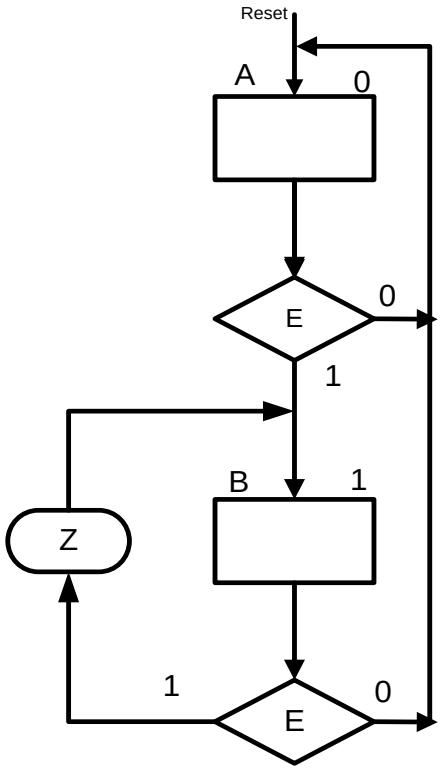
Clock cycle:	1	2	3	4	5	6	7	8	9	10	11
E :	0	1	0	1	1	0	1	1	1	0	1
Z :	0	0	0	0	0	1	0	0	1	1	0



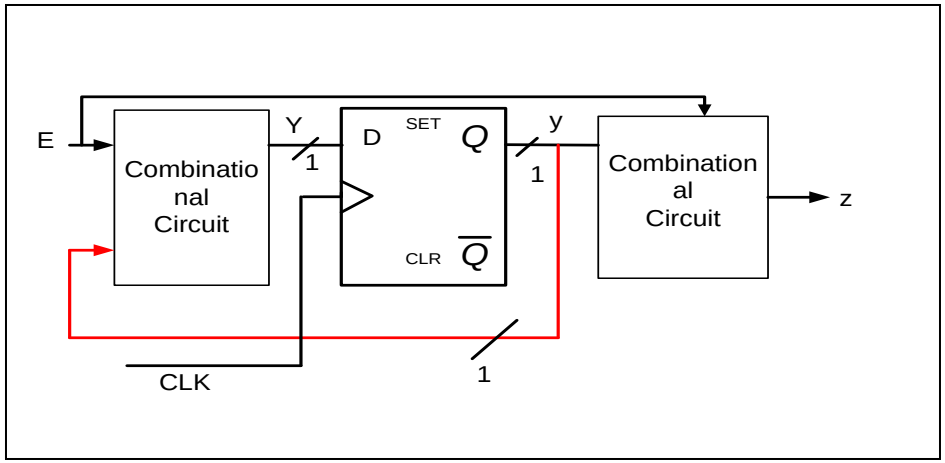
S(y)	E(Y) 0	1	Z
A	A	B	00
B	A	B	01

Y=E
Z=E y

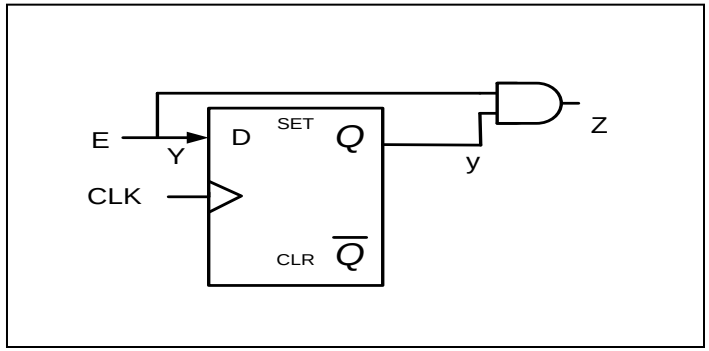
Flow chart :



Function block diagram



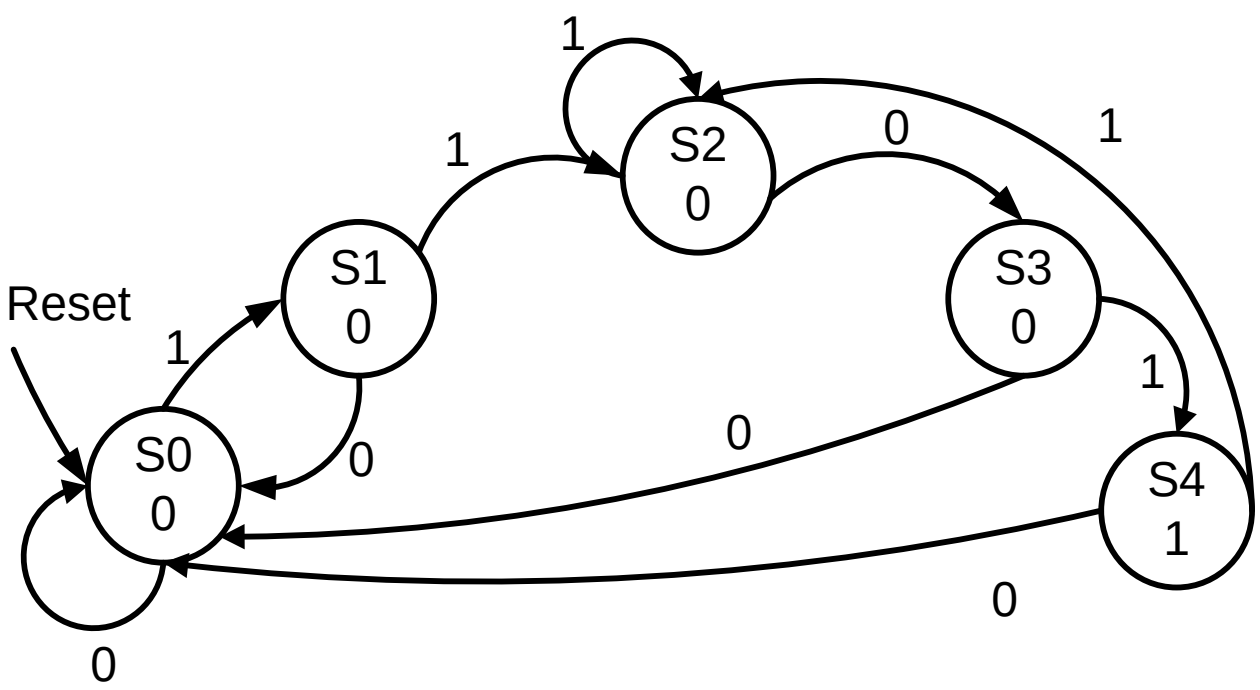
Logic circuit diagram



Example: Robotic Smile

Designer owns a pet robotic snail with an FSM brain. The snail crawls from left to right along a paper tape containing a sequence of 1's and 0's. On each clock cycle, the snail crawls to the next bit. The snail smiles when the last four bits that it has crawled over are, from left to right, 1011. Design the FSM to computer when the snail should smile.
The input A is the bit underneath the snail's antennae. The output Y is true when the snail smiles. Compare Moore and Mealy state machine designs. Sketch a timing diagram for each machine showing the input, states and output as your snail crawls along the sequence 111011010.

R1011L
M----L



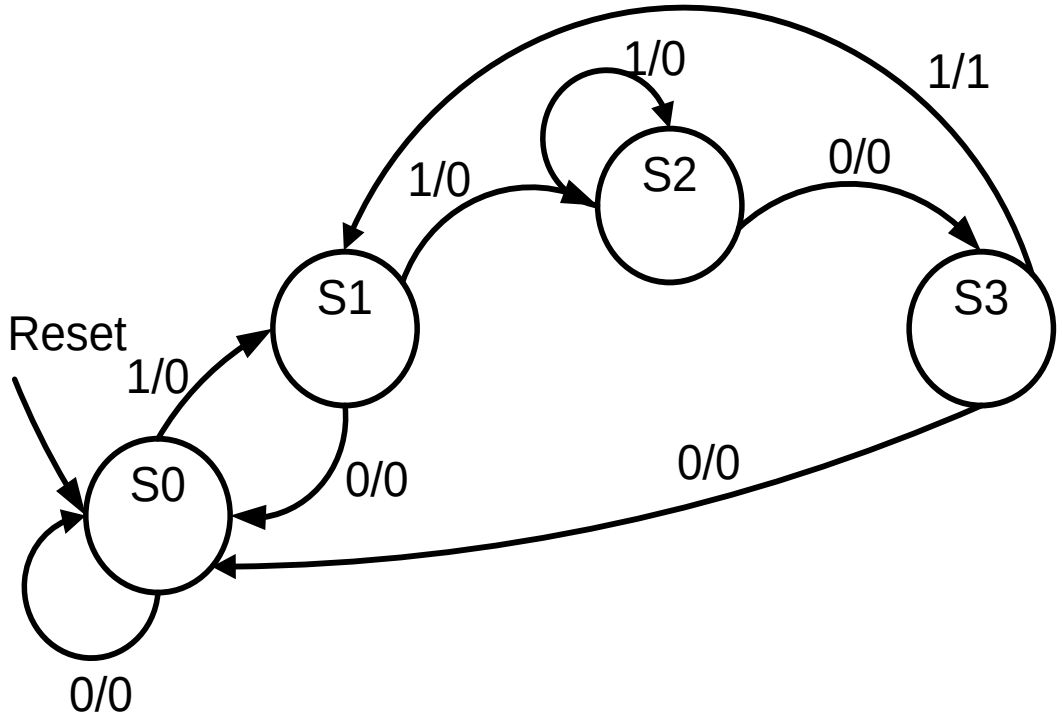
Moore machine

Present y	Input (E)	NEXT STATE (Y)	OUTPUT (Z)
S0	0	S0	0
S0	1	S1	
S1	0	S0	0
S1	1	S2	
S2	0	S3	0
S2	1	S2	
S3	0	S0	0
S3	1	S4	
S4	0	S0	1
S4	1	S2	

Present			Input	NEXT STATE			output
y2	y1	y0	E	Y2	Y1	Y0	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	
0	0	1	0	0	0	0	0
0	0	1	1	0	1	0	
0	1	0	0	0	1	1	0
0	1	0	1	0	1	0	
0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	
1	0	0	0	0	0	0	1
1	0	0	1	0	1	0	

Homework : output equations and logic circuit and timing waveform

Mealy Machine



Present state		Input	Next state		Output
y2	y1	A	Y2	Y1	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	1

Homework : output equations and logic circuit and timing waveform

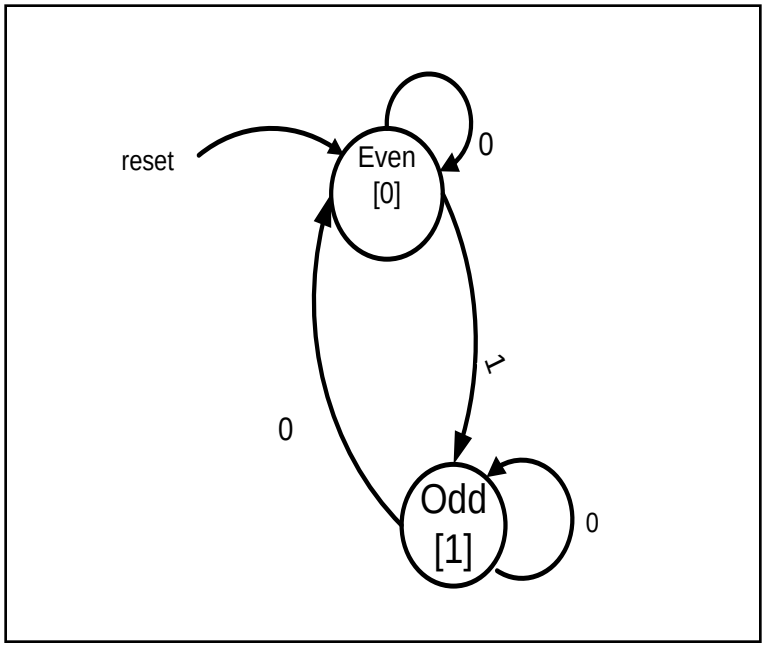
Example 3: odd parity checker

Introduction:

UART (universal asynchronous receiver transmitter) is an example. Recovery from the error is usually done by retransmitting the data that use in telecommunication. Consider an *even parity scheme* using nine bit codewords. The code comprises 8 data bits followed by a parity bit. The following examples would make the parity scheme clear:

1. The parity of the data 1111 1110 is odd since there are 7 numbers of '1' bits in the data. The parity bit will be 1, giving the code word 1111 1110 1.
2. The parity of the data 1111 1111 is even as there are 8 numbers of '1' bits . The parity bit is 0, giving the code word 1111 1111 0.
3. The parity of the data 0000 0000 is even (zero being an even number). The parity bit is 0, giving the code word 0000 0000 0.
4. A null or non-existent bit stream also has zero '1' bits and, therefore, it would get the parity bit 0 in an even parity scheme.

The specification: Assert output whenever input bit stream has odd # of 1's.
Step 1: understand the specs.
Get sample input/output relationship. In class
Step 2 : draw state diagram, step 3: symbolic state table,



Present (y)state	Next state(Y)		Out (Z)
	E=0	E=1	
Even	Even	odd	0
Odd	odd	Even	1

Present state	Input	Next state	Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Step 4 :Encoded state transition table &Step 5 : output equations

Present state	Input	Next state	Output
y	E	Y	Z
0	0	0	0
0	1	1	
1	0	1	1
1	1	0	

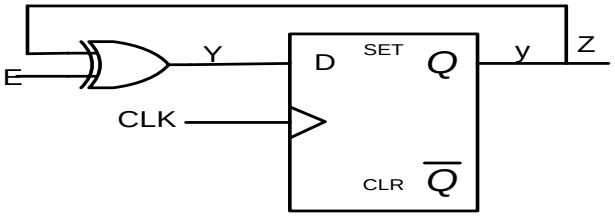
Present (y)state	Next state(Y)		Out (Z)
	E=0	E=1	
0	0	1	0
1	1	0	1

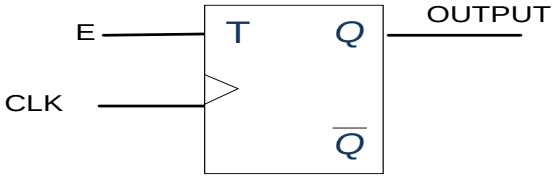
Y=y ^ E

y \ E	0	1
0	0	1
1	1	0

Z=y=OUT

y	
0	
1	





Timing in class

Latch/Flip-Flop Characteristic Equations

Device	Characteristic Equations
S-R latch	Q+ = S+R'.Q
D latch	Q+ = D
Edge-triggered D flip-flop	Q+ = D
Master/Slave S-R flip-flop	Q+ = S+R'.Q
Master/Slave J-K flip flop	Q+ = J.Q' + K'.Q
Edge Triggered J-K flip-flop	Q+ = J.Q' + K'.Q
T flip-flop	Q+ = Q'
T flip-flop with enable	Q+ = EN.Q' + EN'.Q

State Machine Analysis Procedure

Step1: Assign s state variable to each flip-flop in the synchronous sequential logic circuit.

Step2: Analyze the combinational logic to determine flip-flop input (excitation) equations: Di = Fi (Q, inputs)

Also write the Moore and/or Mealy output equations.

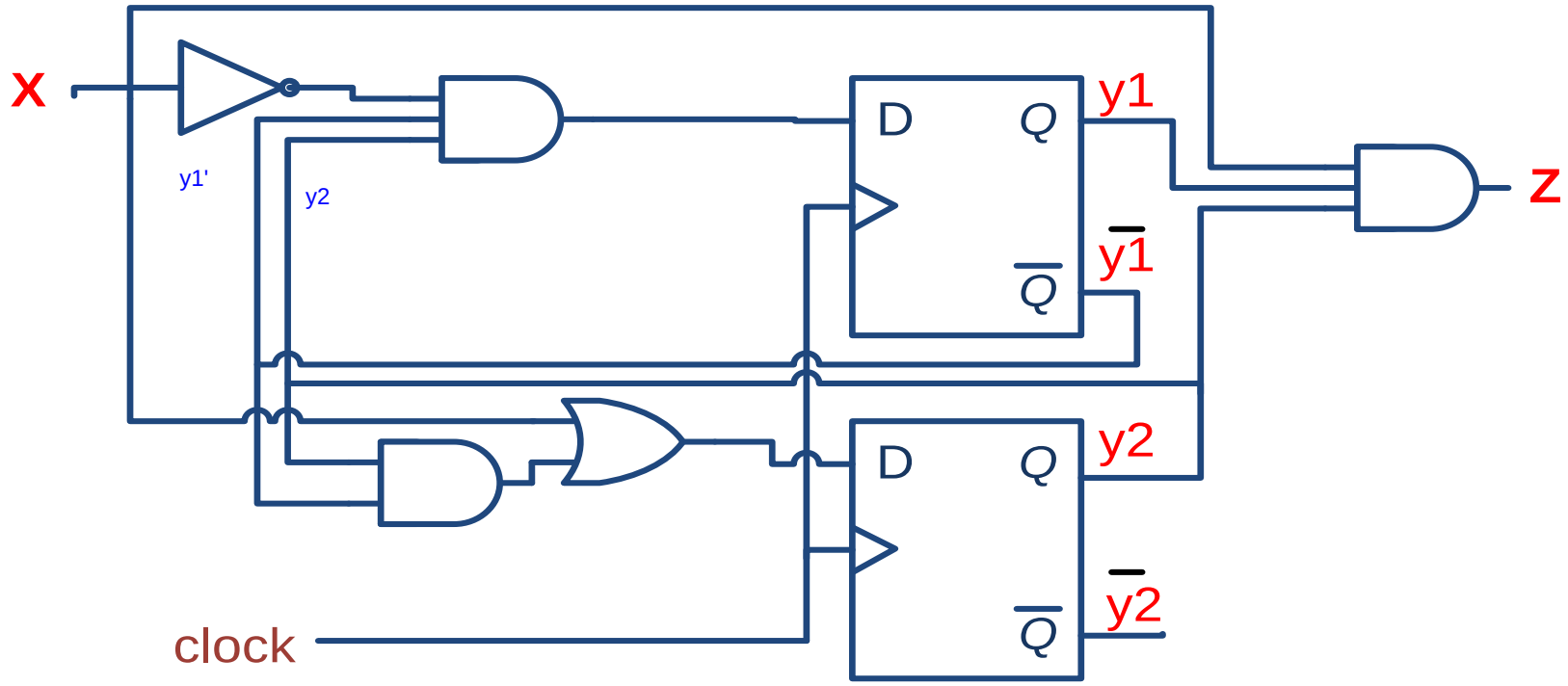
Step3: Substitute excitation equations into flip-flop characteristic equations, to obtain the next state output equations.

Step 4: Obtain a composite Karnaugh map using the next state output equations and Moore and/or Mealy output equations

Step 5: Use the composite Karnaugh map to obtain a PS/NS table, state diagram, ASM chart, flow map or timing diagram to show the behavior of the circuit.

Example 9-2/p515

- Analyze the synchronous Mealy machine in Figure below to obtain its state diagram.



Solution:-

$$D1 = \overline{y1} \cdot y2 \cdot \overline{X}$$

$$D2 = X + \overline{y1} \cdot y2$$

$$Z = y1 \cdot y2 \cdot X$$

Y1Y2,Ps Z=

y1y2 \ X	0	1
00	00,0	01,0
01	11,0	01,0
11	00,0	01,1
10	00,0	01,0

composite Karnaugh map

Y1=

y1y2 \ X	0	1
00	0	0
01	1	0
11	0	0
10	0	0

Y2=

y1y2 \ X	0	1
00	0	1
01	1	1
11	0	1
10	0	1

Ps Z=

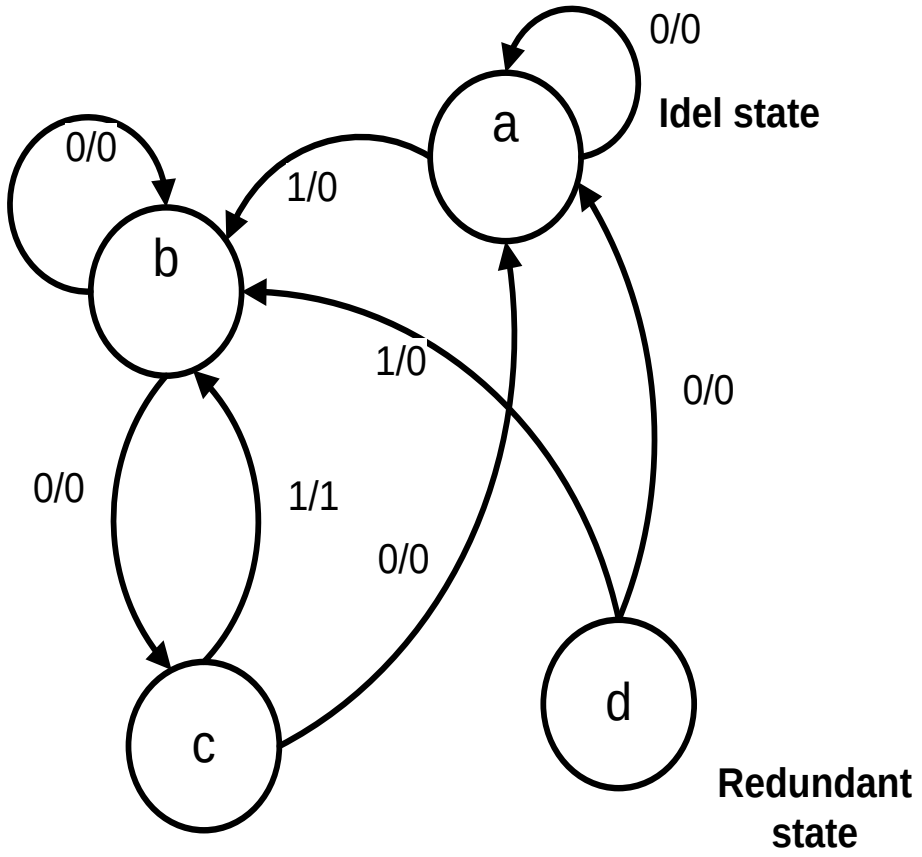
y1y2 \ X	0	1
00	0	0
01	0	0
11	0	1
10	0	0

Q=D→Y=D

$$Y1 = \overline{y1} \cdot y2 \cdot \overline{X}$$

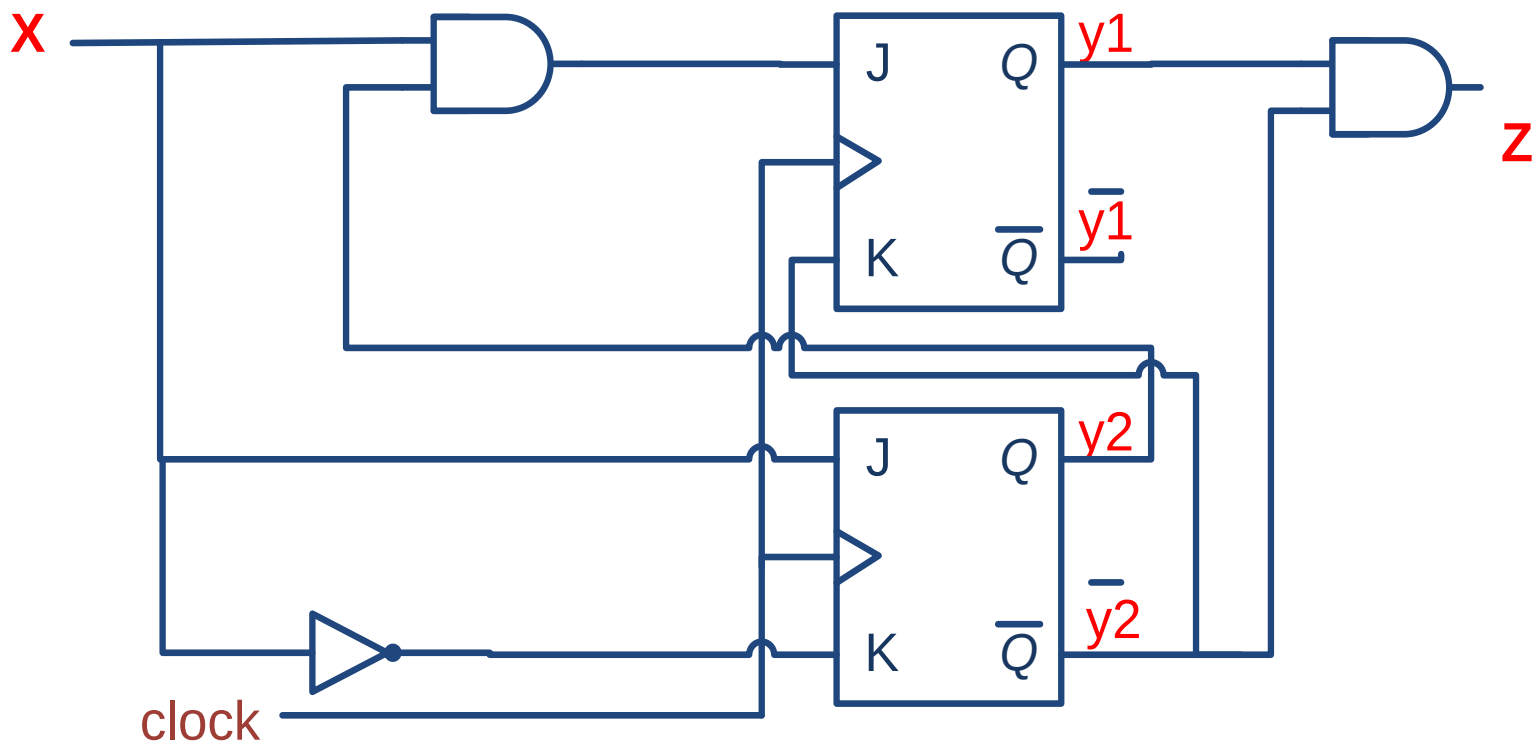
$$Y2 = X + \overline{y1} \cdot y2$$

Y1Y2,Ps Z=	X=0	X=1
a 00	a,0	b,0
b 01	c,0	b,0
c 11	a,0	b,1
d 10	a,0	b,0



Example 9-3/p517

- Analyze the synchronous Moore machine in Figure below to obtain its state diagram.



Solution:-

$$Y = Q = q \cdot \overline{K} + \overline{q} \cdot J$$

$$J1 = y2 \cdot X$$

$$K1 = \overline{y2}$$

$$Y1 = y1 \cdot \overline{K1} + \overline{y1} \cdot J1 = y1 \cdot y2 + \overline{y1} \cdot y2 \cdot X$$

$$J2 = X$$

$$K2 = \overline{X}$$

$$Y2 = y2 \cdot \overline{K2} + \overline{y2} \cdot J2 = y2 \cdot X + \overline{y2} \cdot X = X$$

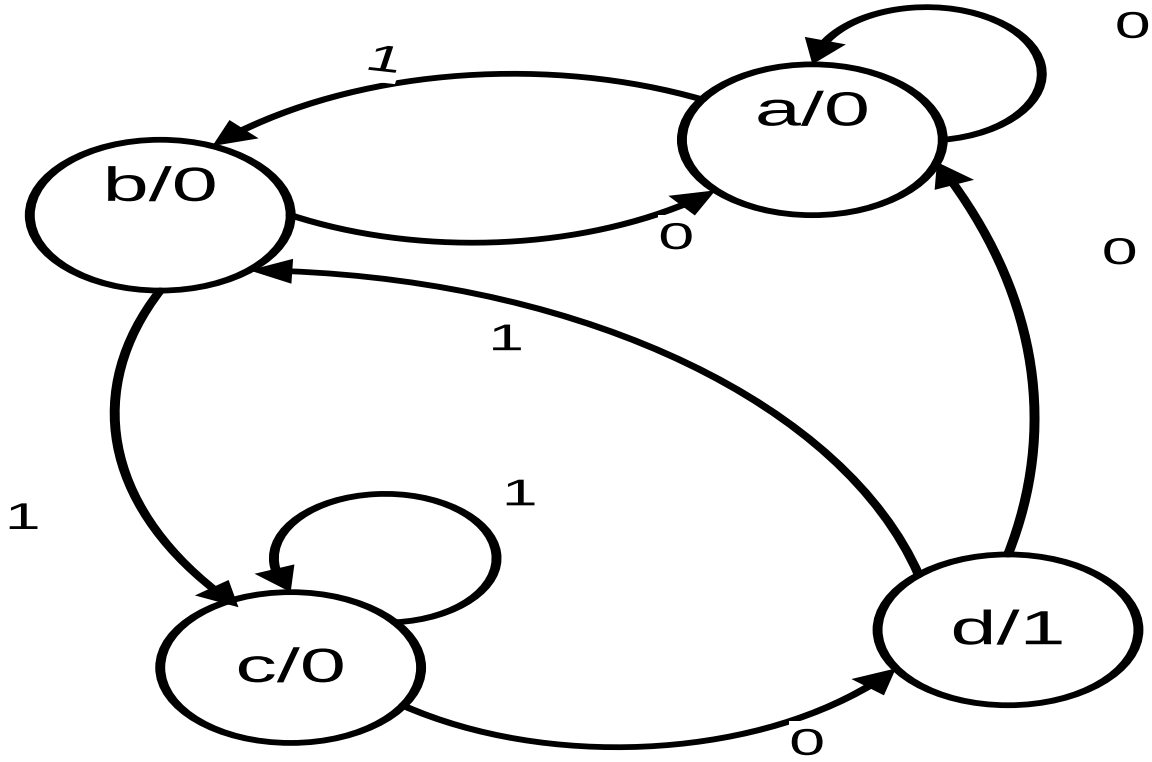
$$Z = y1 \cdot \overline{y2}$$

Example 9-3

Y1Y2= y1y2 \ X	0	1	Ps Z=
00	00	01	0
01	00	11	0
11	10	11	0
10	00	01	1

Y1Y2= y1y2 \ X	0	1	Ps Z=
a 00	**a**	**b**	0
b 01	**a**	**c**	0
c 11	**d**	**c**	0
d 10	**a**	**b**	1

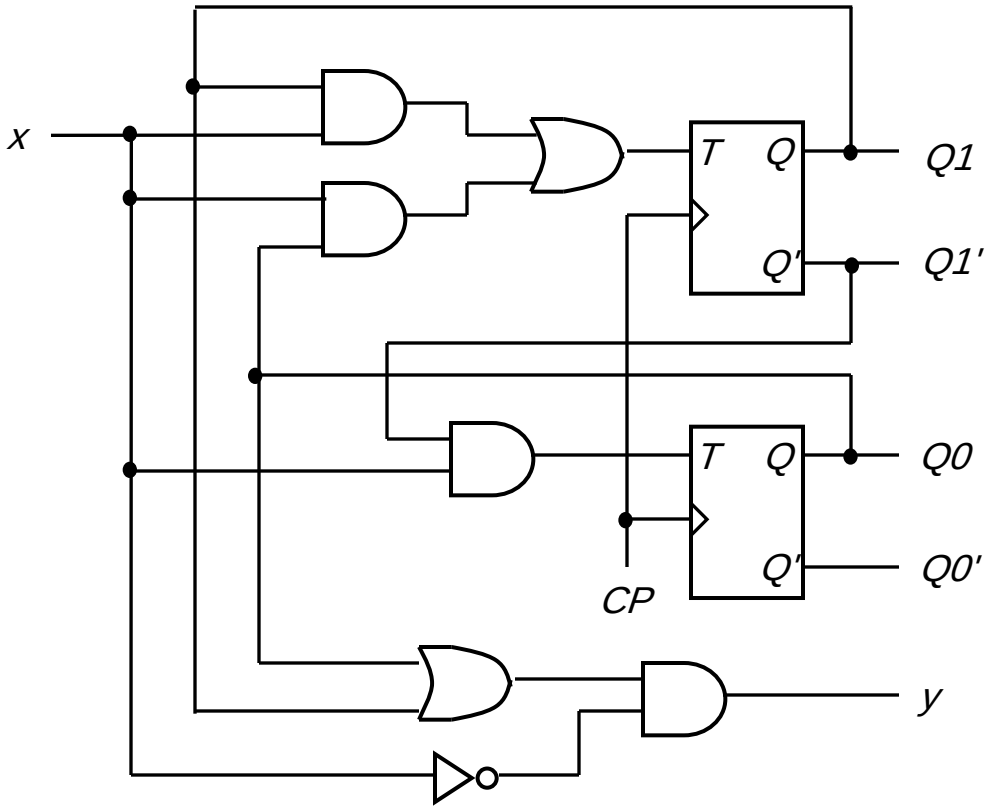
Example 9-3



Input=X
/Output=/PS Z

State Machine Analysis Example

Analyze the state machine:



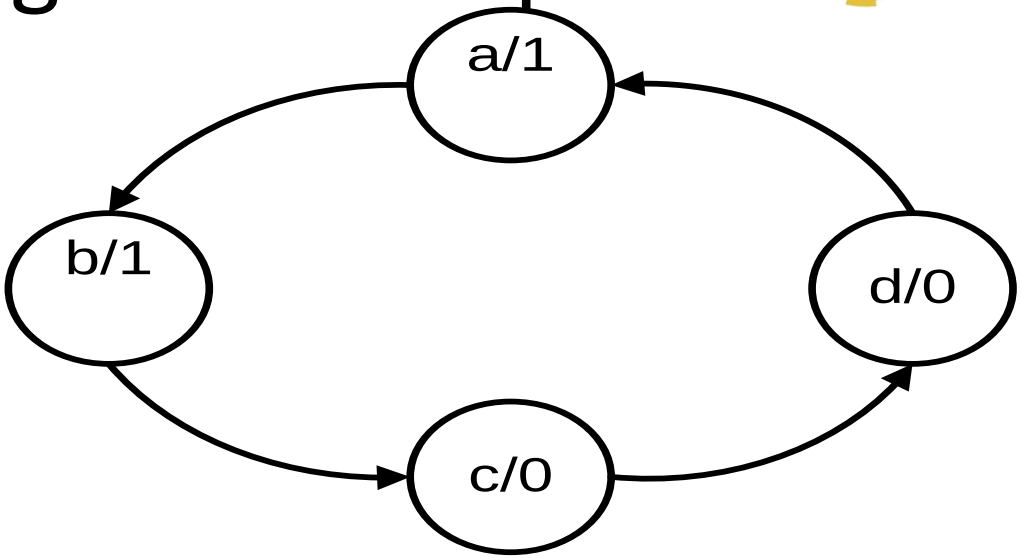
This is a Mealy Machine since output = G(current state, input)

Timing diagram description

$$T_Z = 4 * T_{CK}$$

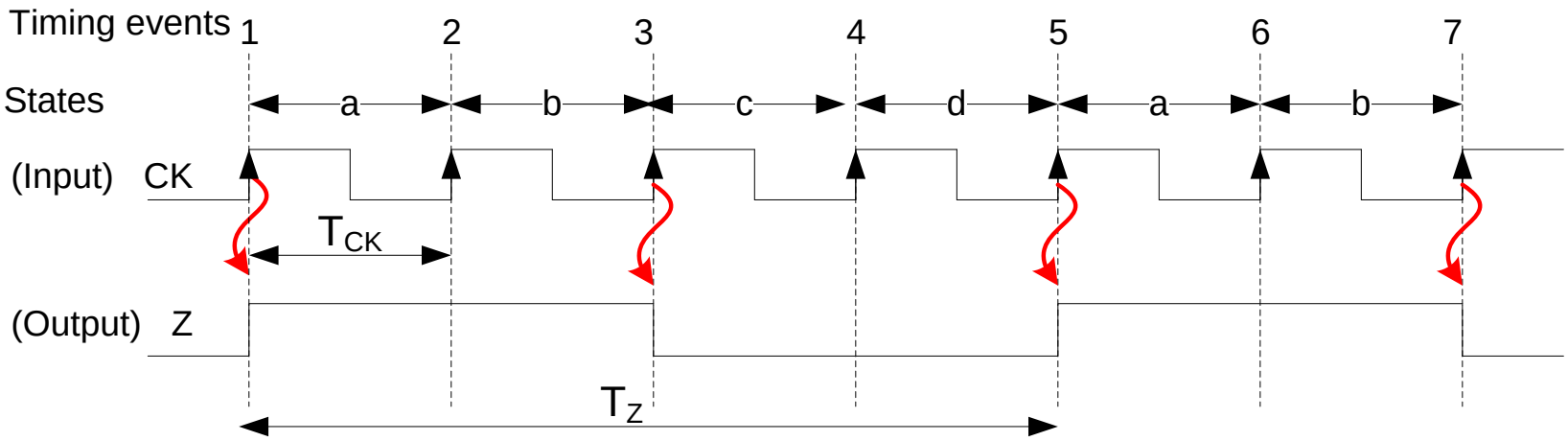
$$F = \frac{1}{T}$$

$$F_Z = \frac{1}{4} * F_{CK}$$



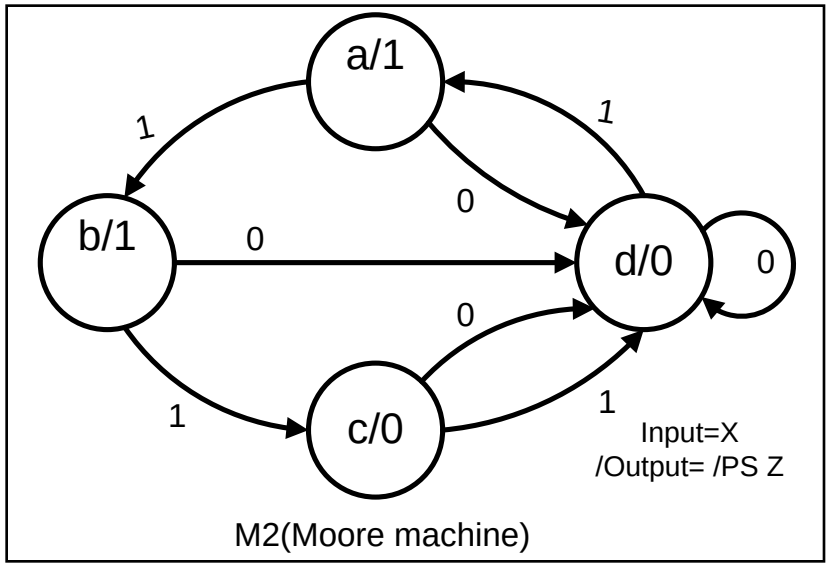
M1(Moore machine)

Frequency Divider

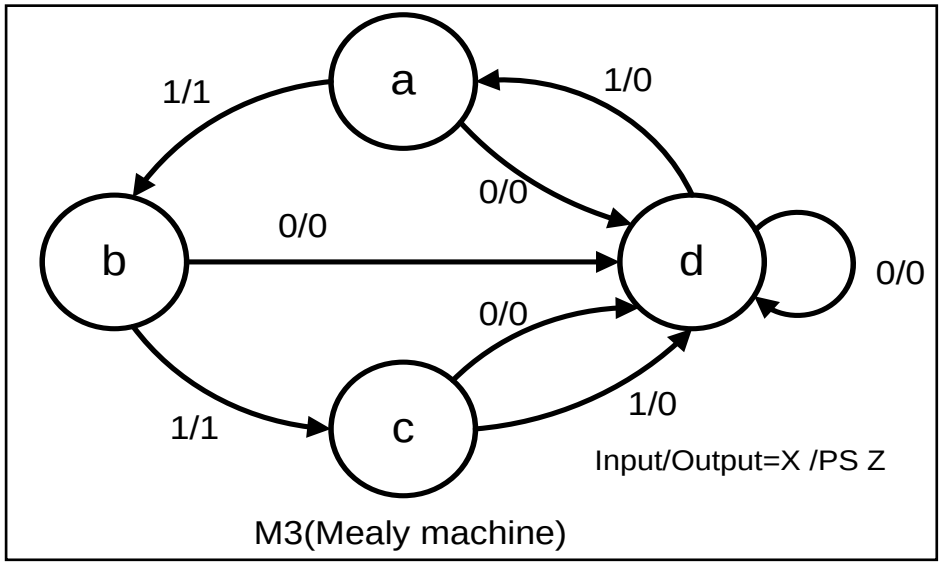


Example 9-4/P521

For both synchronous machines M2 and M3 we have one external input X. When the external input is 1, each provides an output signal Z is ¼ the frequency of the system clock signal, just like M1 in the timing diagram. When x is 0 on the next clock timing event, both M2 and M3 go to state d (an idle or reset state). both M2 and M3 stay in the idle state until X is 1 on the next clock timing event, allowing both machines to move to state a.



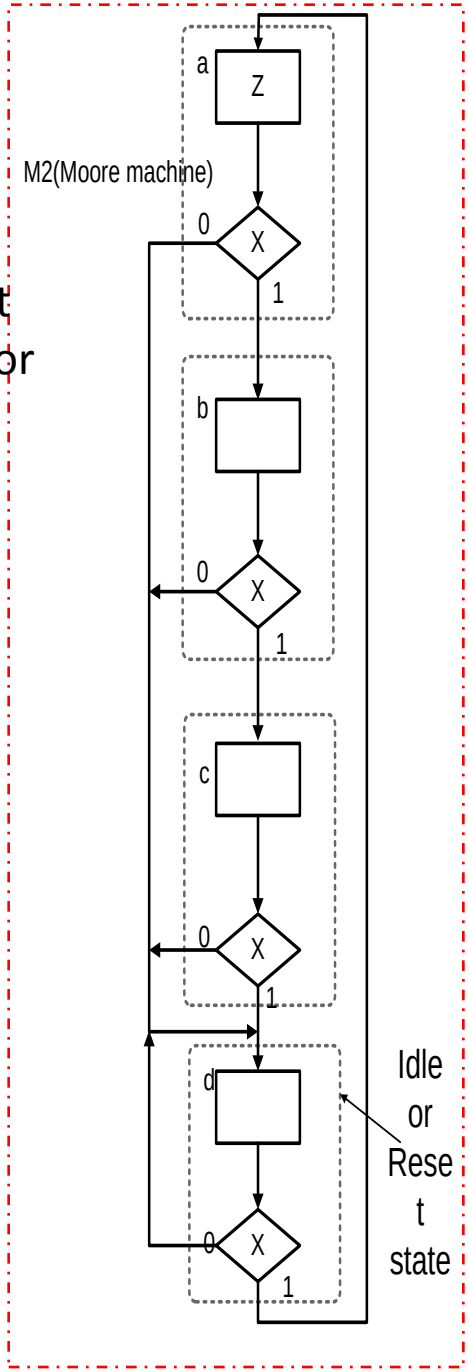
M2(Moore machine)

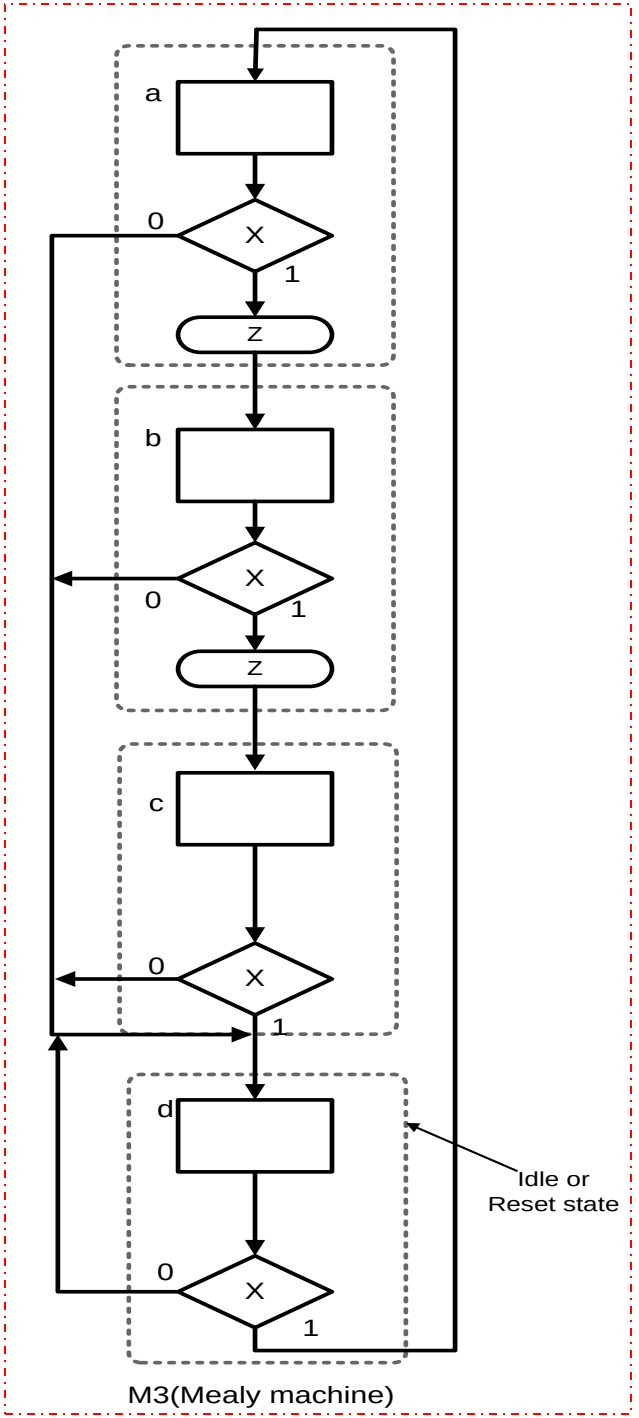


M3(Mealy machine)

Example 9-4

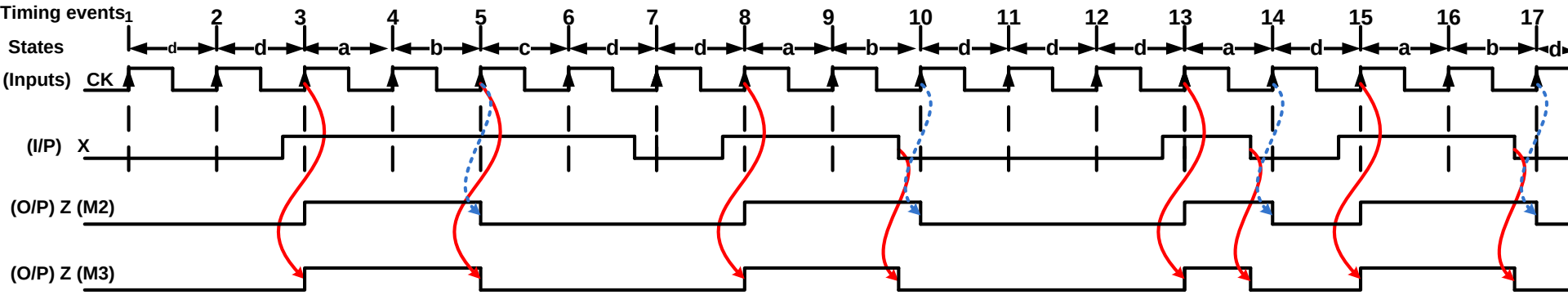
(a-) Draw an equivalent ASM chart for a Moore and Mealy machine for M2 and M3.

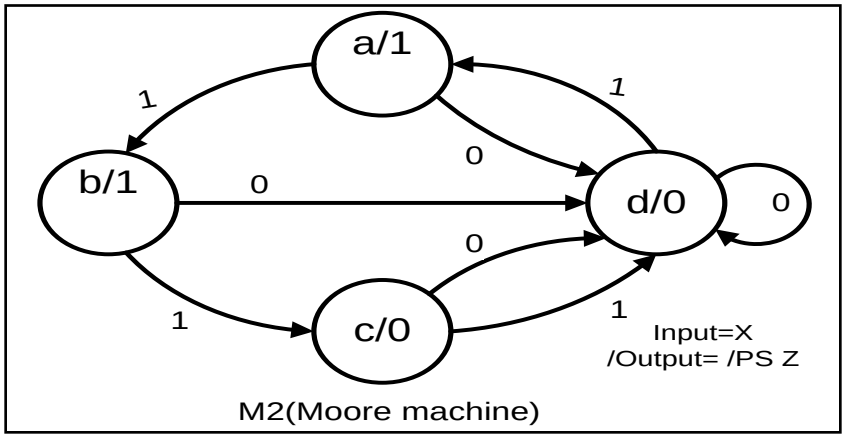


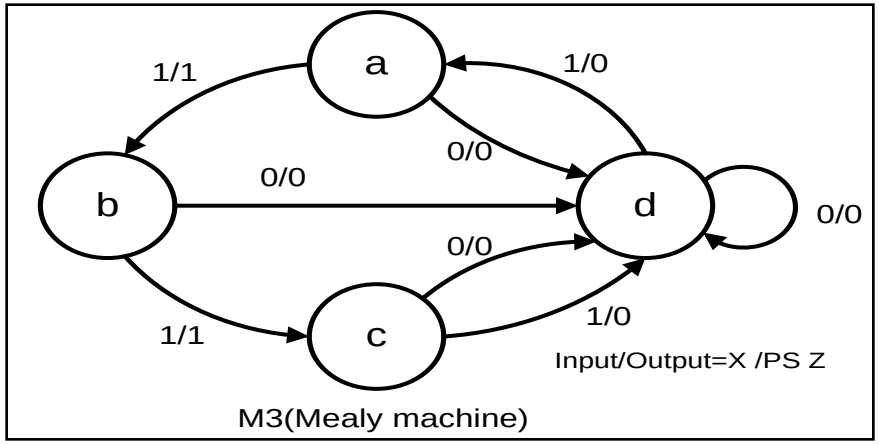


Example 9-4

(b-) Draw a timing diagram for both M2 and M3 machine for the input sequence 00111101100010110 where the MSB in the binary number represent the first value of X in the sequence. Show the change in the asynchronous input X occurring approximately ¼ of a cycle prior to the next clock timing event. Assume that output Z is initially o and the machine is in the idle state .

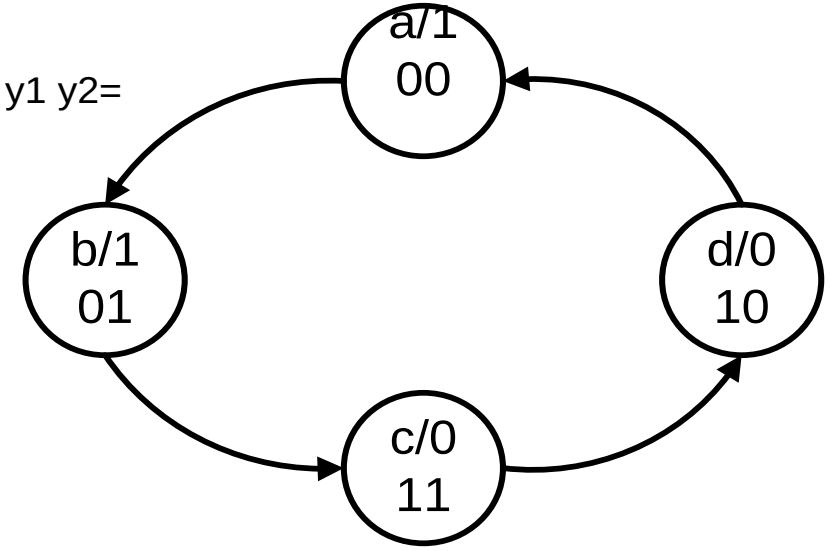






D Flip-Flop Design procedure

- 1-General Method (composite Karnaugh map.)
- 2- Set or Hold Method



M1(Moore machine)

Y1Y2=	X	Ps Z=
y1y2		
a 00	b	1
b 01	c	1
c 11	d	0
d 10	a	0

1-General Method (composite Karnaugh map.)

$$Y1 = y2$$

$$Y2 = \bar{y}1$$

$$Z = \bar{y}1$$

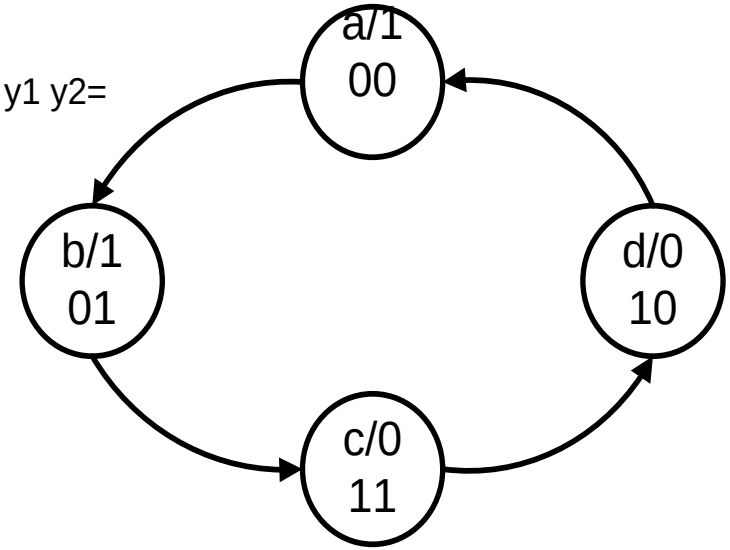
Y1Y2=

y1y2 \ X	Ps Z=
a 00	b 1
b 01	c 1
c 11	d 0
d 10	a 0

Y1Y2=

y1y2 \ X	Ps Z=
a 00	01 1
b 01	11 1
c 11	10 0
d 10	00 0

2- Set or Hold Method



M1(Moore machine)

$$Y1 = \bar{y}1.y2 + y1.y2$$

Y1=1 when y1$\bar{y2}$=1 (Set b→c)

Y1=1 when y1y2=1 (Hold c→d)

$$Y2 = \bar{y}1.\bar{y}2 + \bar{y}1.y2$$

Y2=1 when y1$\bar{y2}$=$\bar{1}$ (Set a→b)

Y2=1 when y1$\bar{y2}$=1 (Hold b→c)

$$Z = \bar{y}1.\bar{y}2 + \bar{y}1.y2$$

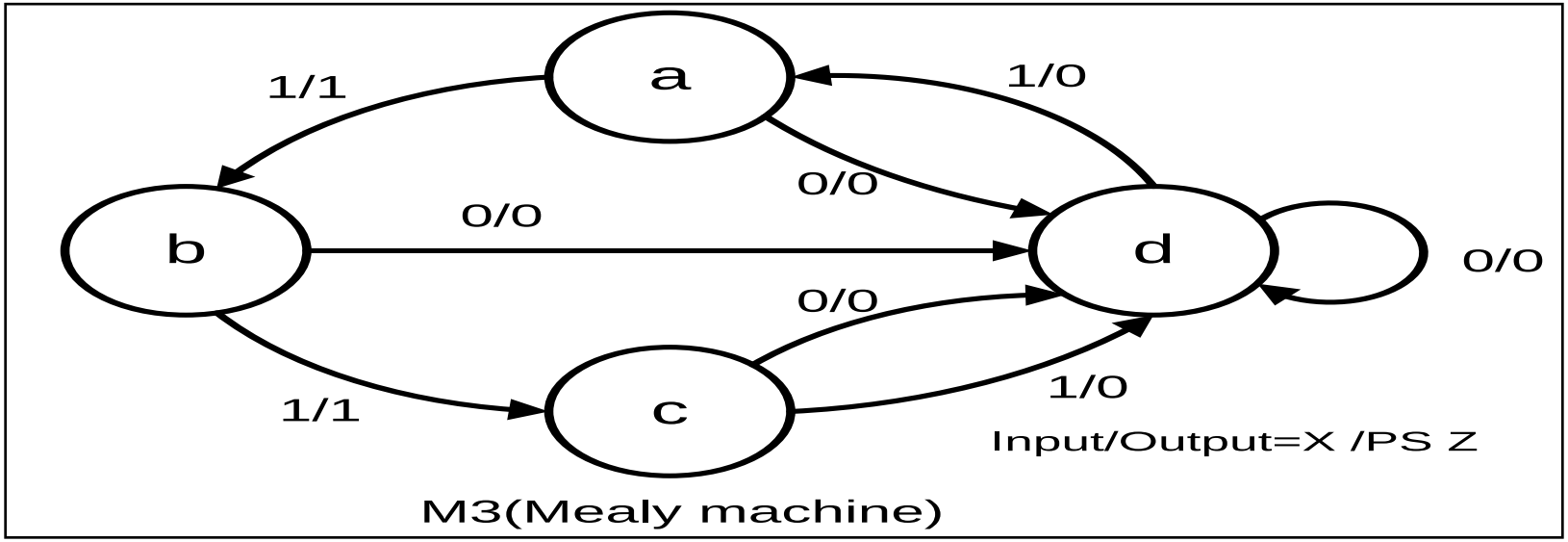
Example 9-7/P531

Obtain the synchronous circuit equation for M3. Use the following state assignments y1y2=00 for d, 01 for a, 11 for b 10 for c).

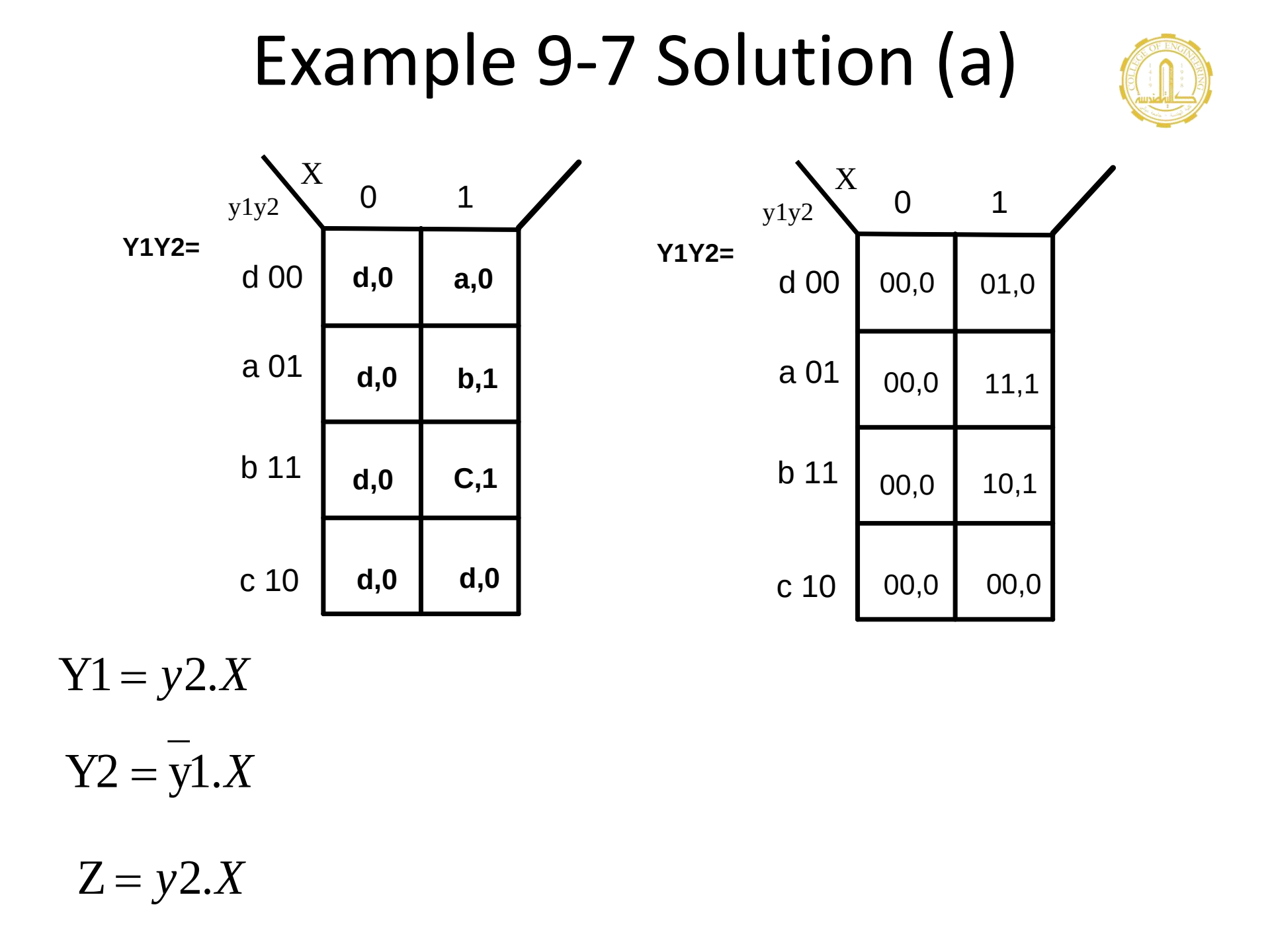
(a) Using a composite Karnaugh map.

(b) Using Set or Hold method.

(c) Use the reduced equation to obtain a circuit diagram using positive edge-triggered D flip-flops.



Example 9-7 Solution (a)



Y1Y2=

y1y2 \ X	0	1
d 00	d,0	a,0
a 01	d,0	b,1
b 11	d,0	C,1
c 10	d,0	d,0

Y1Y2=

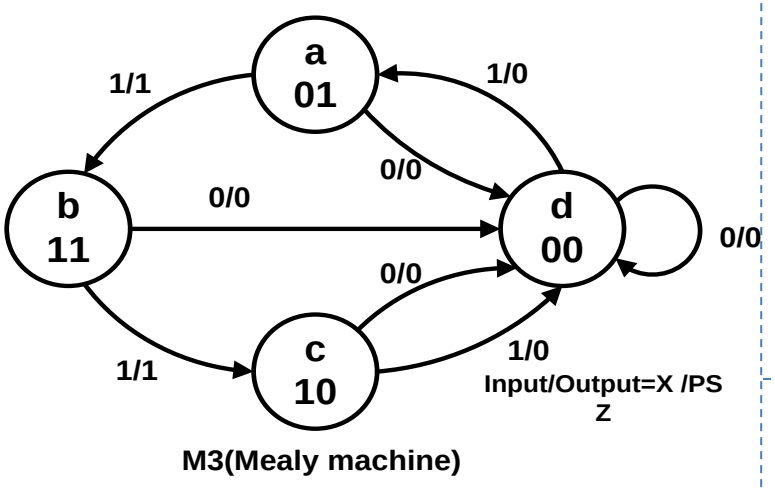
y1y2 \ X	0	1
d 00	00,0	01,0
a 01	00,0	11,1
b 11	00,0	10,1
c 10	00,0	00,0

$$Y1 = y2.X$$

$$Y2 = \bar{y}1.X$$

$$Z = y2.X$$

Example 9-7 Solution (b)



M3(Mealy machine)

$$Y1 = \bar{y}1.y2.X$$ Y1=1 when $y1\bar{y}2=1$ AND X=1 (Set a→b)

$+$

$$y1.y2.X$$ Y1=1 when y1y2=1 AND X=1 (Hold b→c)

$$Y2 = \bar{y}1.\bar{y}2.X$$ Y2=1 when $y1\bar{y}2\bar{=}1$ AND X=1 (Set d→a)

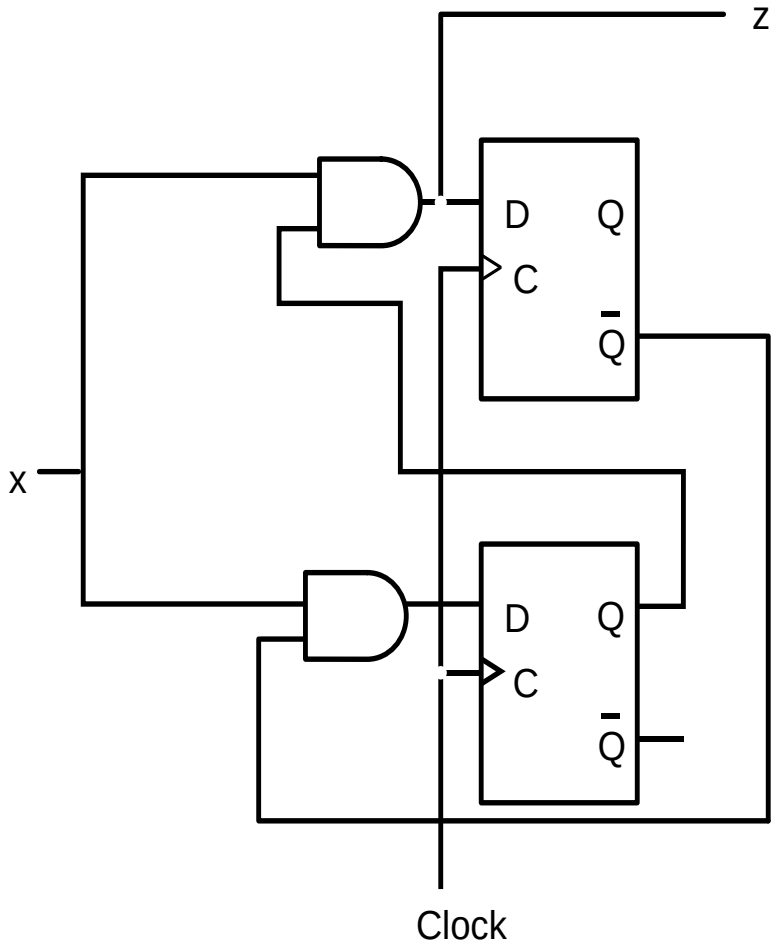
$+$

$$\bar{y}1.y2.X$$ Y2=1 when $\bar{y}1y2=1$ AND X=1 (Hold a→b)

$$Z = y1.y2.X$$

$$+ \bar{y}1.y2.X$$

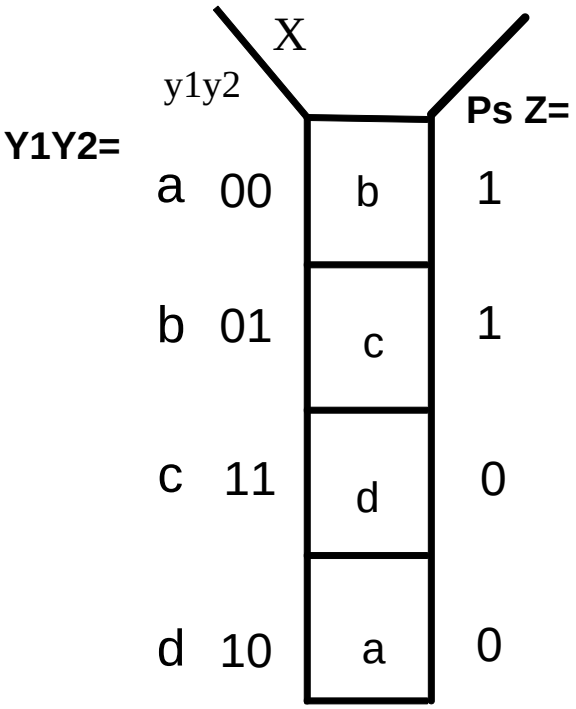
Example 9-7 Solution (c)

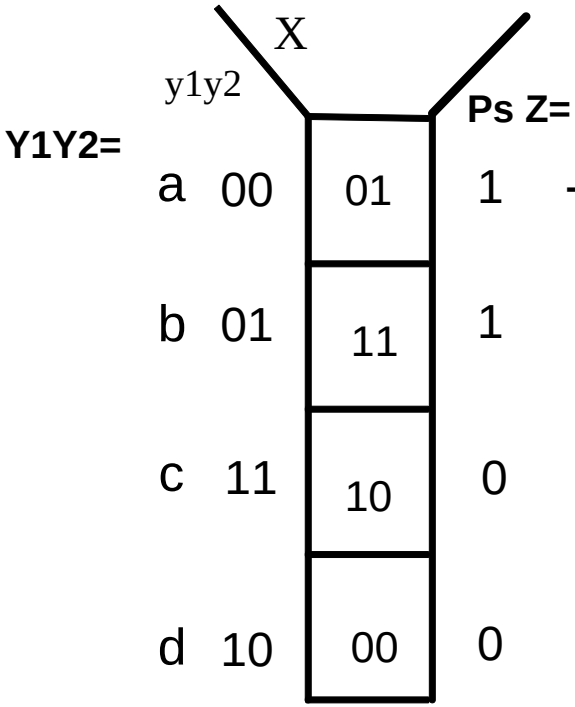


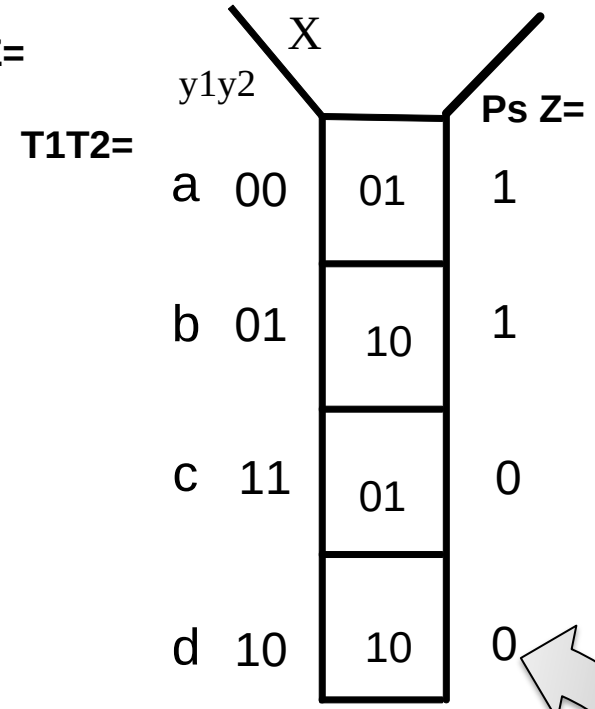
$$D1 = Y1 = y2.X$$

$$D2 = Y2 = \bar{y}1.X$$

$$Z = y2.X$$







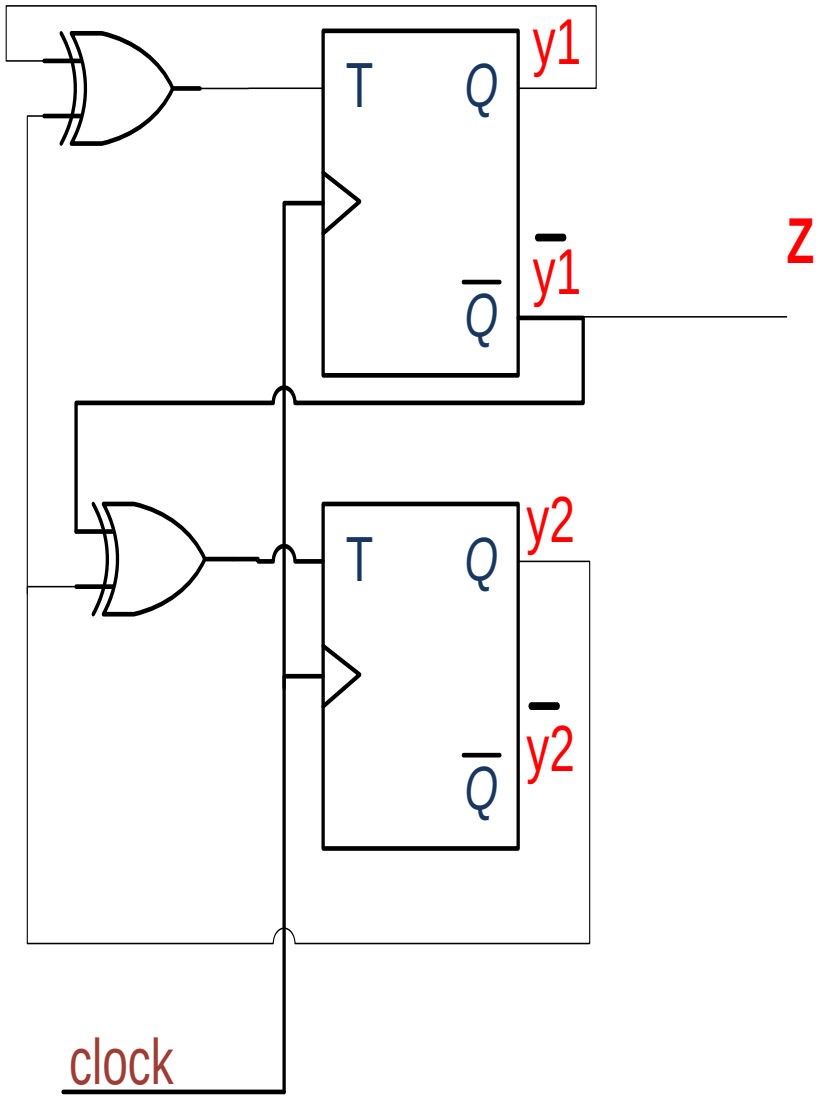
State transition		
PS (q)	NS (Q)	T
0	0	0
0	1	1
1	0	1
1	1	0

T Flip-Flop Design procedure

$$T1 = y1 \oplus y2$$

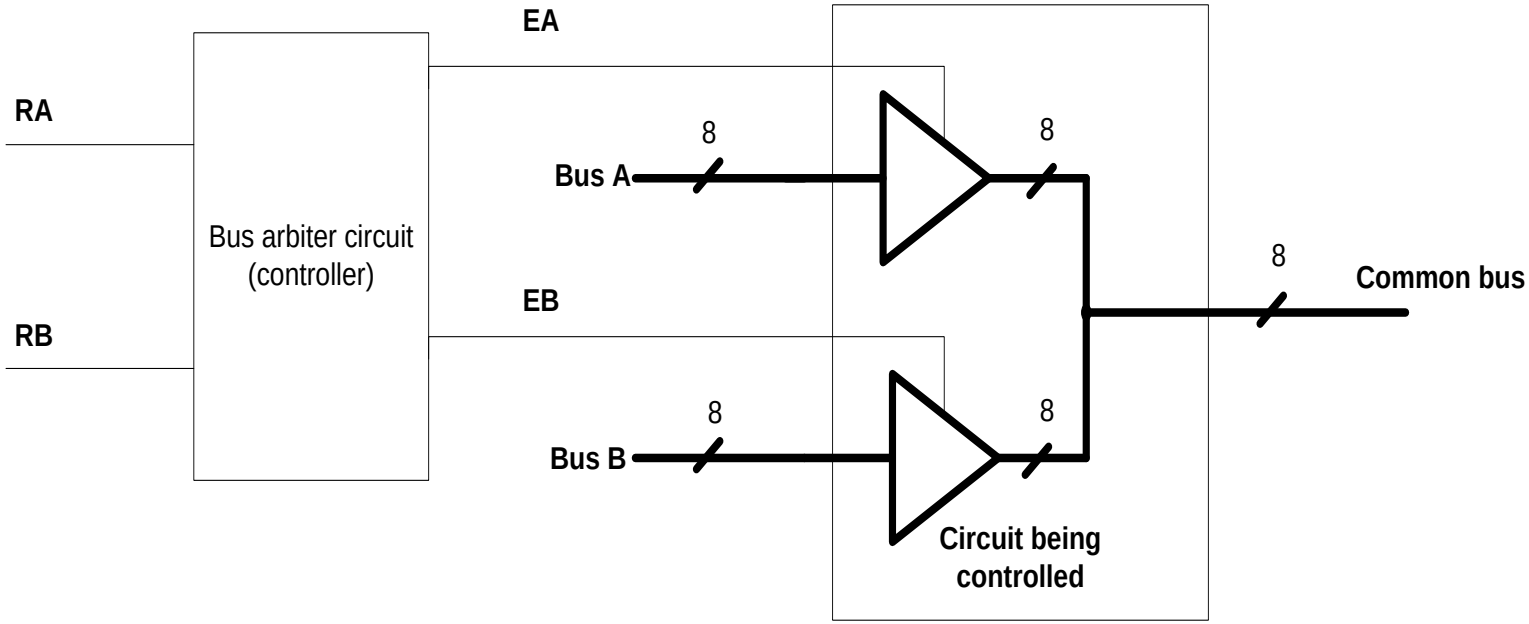
$$T2 = y1 \otimes y2 = \overline{y1} \oplus y2$$

$$Z = \overline{y1}$$



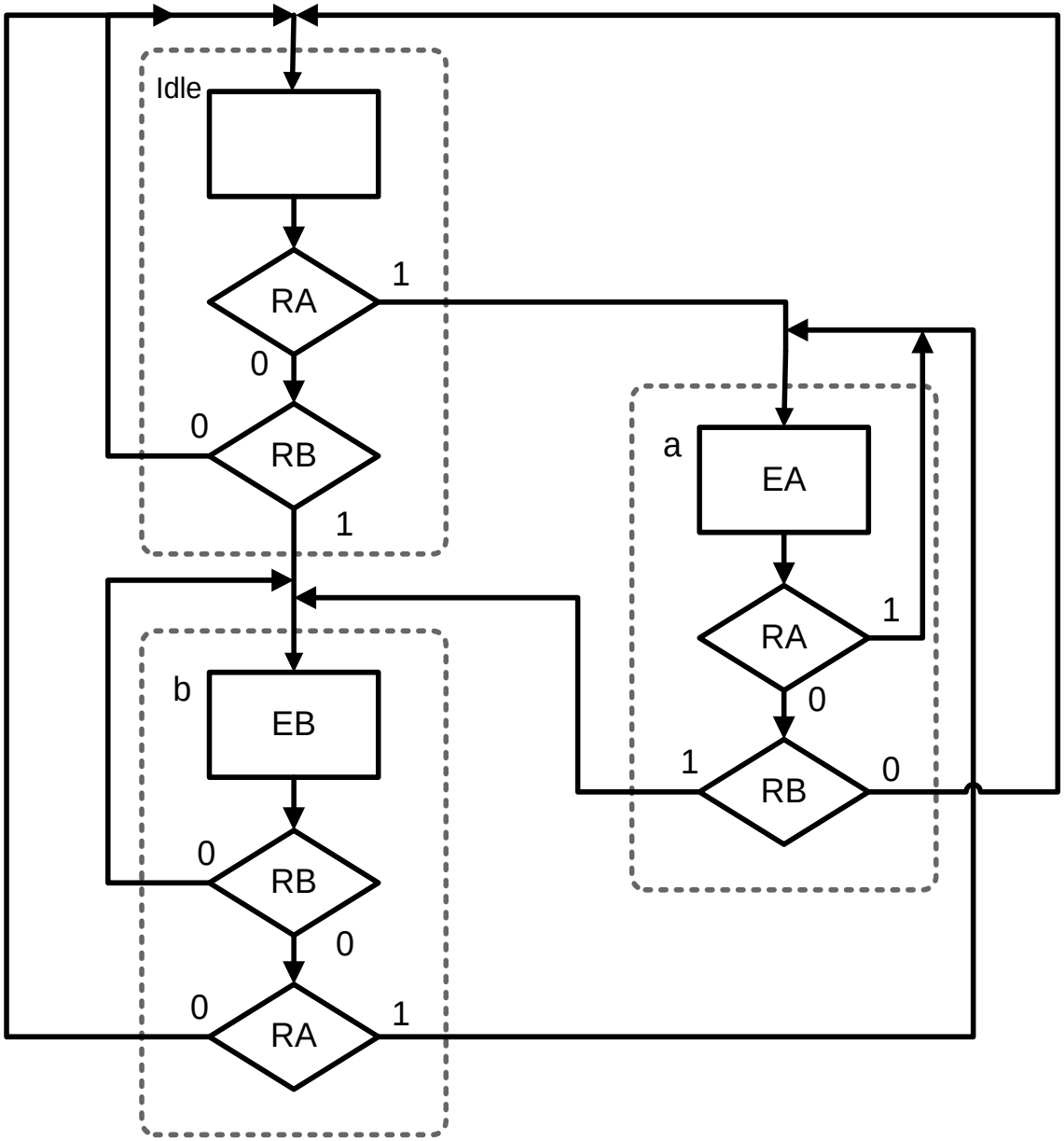
Example 9-9/p539

- RA RB represent bus request A and request B (input signal).
- EA EB represent bus enable A and enable B (output signal).



Example 9-9/p539 Cont.

- Design bus arbiter circuit so the it behaves as following manner:
- 1-For the inputs RA RB=00 the circuit either goes to ‘idle’ state (outputs EA EB=00 disabling both bus A and B buffers) , or remains in the ‘idle’ state.
- 2- When inputs RA RB change to 10 or 11 the circuit goes to ‘a’ state (outputs EA EB=10 enabling bus A and disabling bus B buffers).
- 3- When inputs RA RB change to 01 in the idle state, the circuit goes to ‘b’ state (outputs EA EB=01 disabling bus A and enabling bus B buffers)
- 4-To go from state ‘a’ to state ‘b’ requires inputs RA RB=01
- 5- To go from state ‘b’ to state ‘a’ requires inputs RA RB=10


Idle
RA
1
0
RB
0
1
a
EA
RA
1
0
RB
1
0
b
EB
RB
0
0
RA
0
1

PS	NS Inputs RA RB				PS outputs EA EB
	00	01	11	10	
Idle	Idle	b	a	a	00
a	Idle	b	a	a	10
b	Idle	b	b	a	01
c	Idle	-	-	-	- -

Y1Y2,Ps Z=

State transition			
PS (q)	NS (Q)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

y1y2 \ RA RB	00	01	11	10	PS EA EB
00	00	11	01	01	00
01	00	11	01	01	10
11	00	11	11	01	01
10	00	XX	XX	XX	XX

$$EA = \overline{y1}.y2$$

$$EB = y1$$

J1J2,Ps Z=

y1y2 \ RA RB	00	01	11	10
00	00	11	01	01
01	0X	1X	0X	0X
11	XX	XX	XX	XX
10	X0	XX	XX	XX

$$J1 = \overline{RA} \cdot RB$$

$$J2 = RA + RB$$

K1K2,Ps Z=

y1y2 \ RA RB	00	01	11	10
00	XX	XX	XX	XX
01	X1	X0	X0	X0
11	11	00	00	10
10	1X	XX	XX	XX

$$K1 = \overline{RB}$$

$$K2 = \overline{RA} \cdot \overline{RB}$$

$$J1 = \overline{RA} \cdot RB$$

$$J2 = RA + RB$$

$$K1 = \overline{RB}$$

$$K2 = \overline{RA} \cdot \overline{RB}$$

$$EA = \overline{y1}.y2$$

$$EB = y1$$