Sequential ckts.

finite State machines:

→ The synchronous (or) clocked sequential ckts are represented by two models

① Moore model

② Mealy model

① **Moore model:** In this model, the o/p depends only on the present state of the flip-flops

② **Mealy model:** In this model, the o/p depends on the present state of the flip-flops and the inputs.

⇒ sequential ckts are also called finite state machine (FSM)

State diagram:

The state diagram (or) state graph is a pictorial representation of the relationships b/w the present state, the next state and the o/p of a sequential ckt.

i.e the state diagram is a pictorial representation of the behaviour of a sequential circuit.

Steps

State — Represented by → a circle (node or vertex)

transition b/w the states — indicated by → Directed lines i.e → connecting the circles

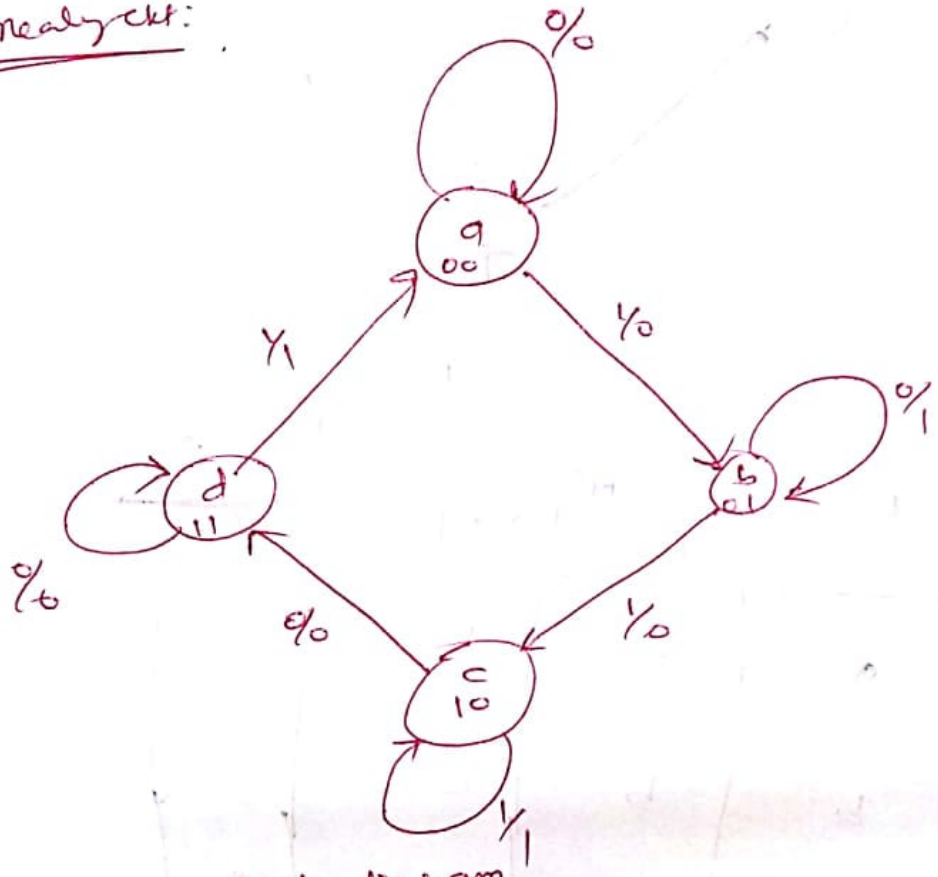
binary number inside the circle — indicated by — state

Directed line labelled with → $x \longrightarrow$ i/p, $y \longrightarrow$ o/p. for ① i/p.

② binary number in the form of $\frac{x}{y}$

Ex:

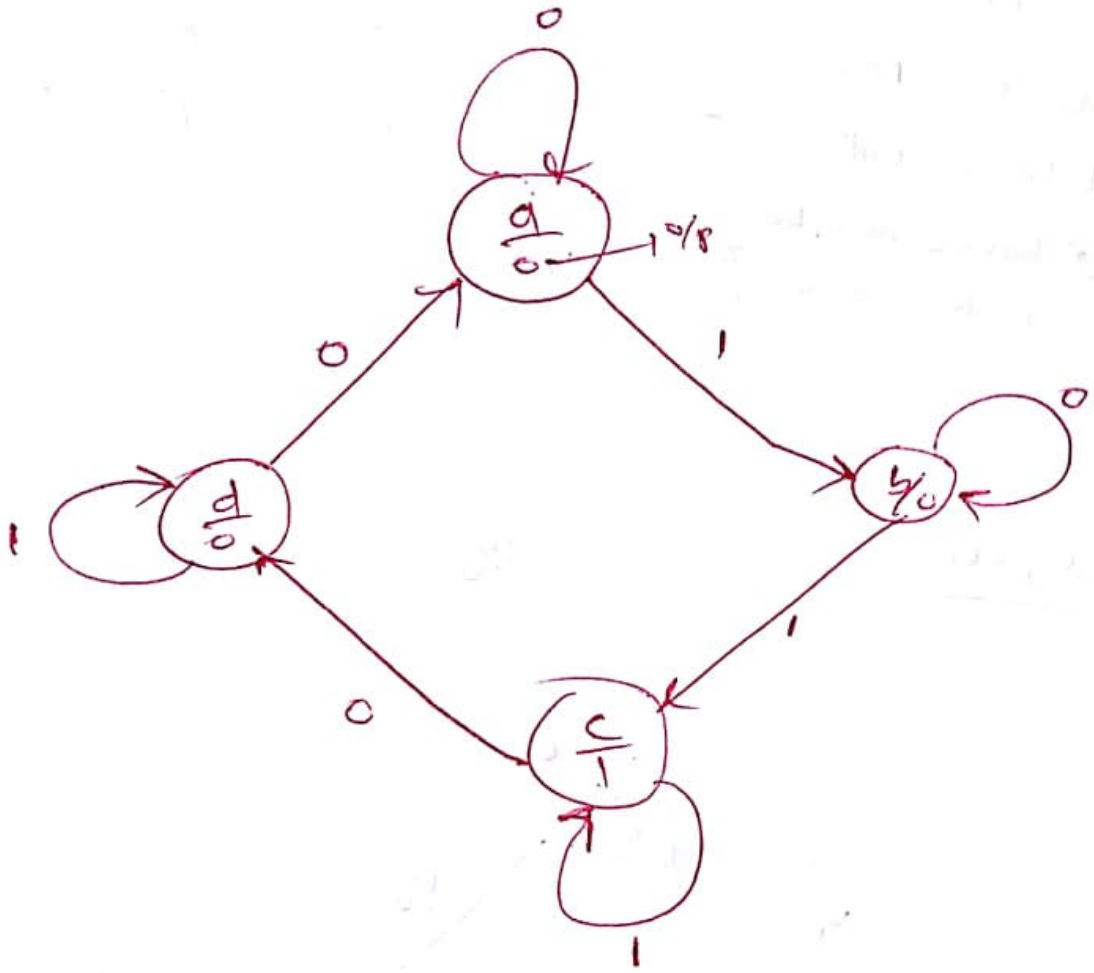
Mealy ckt:



State diagram

PS	NS, o/p		o/p	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	b	c	1	0
c	d	c	0	1
d	d	a	0	1

Moore clk :



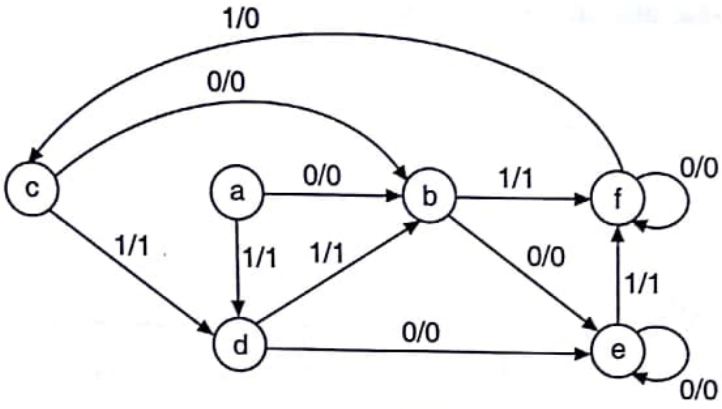
PS	NS		O/P
	x=0	x=1	
a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	1

State reduction: The state reduction technique basically avoids the introduction of redundant states. The reduction in redundant states reduces the number of flip-flops and logic gates, reducing the cost of the final circuit. Two states are said to be equivalent if every possible set of inputs generate exactly the same output and the same next state. When two states are equivalent, one of them can be removed without altering the input-output relationship. Let us illustrate the reduction

technique with an example. Consider the sequential circuit whose state diagram is shown in Figure 6.120a. As shown in the figure, the states are represented by letter symbols instead of their binary values because in state reduction technique, the binary designations of states are not important, but input-output sequences are important. State reduction is done in two steps given as follows.

Step 1. Determine the state table for the given state diagram. Figure 6.120b shows the state table for the given state diagram.

Step 2. Find equivalent states.



(a) State diagram

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	b	d	0	1
b	e	f	0	1
c	b	d	0	1
d	e	b	0	1
e	e	f	0	1
f	f	c	0	0

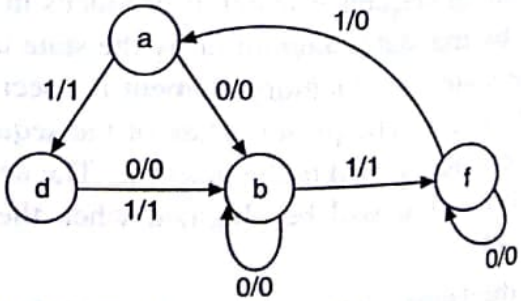
(b) State table

Figure 6.120 Sequential circuit.

As mentioned earlier, in equivalent states, every possible set of inputs generate exactly the same output and the same next state. In the given circuit there are two input combinations X = 0, and X = 1. Looking at the state table for two present states that go to the same next state and have the same output for both input combinations, we can easily find that states a and c are equivalent. Also states b and e are equivalent. This is because, both states a and c go to states b and d and have outputs 0 and 1 for X = 0 and X = 1 respectively. Also states b and e both go to states e and f and have outputs 0 and 1 for X = 0 and X = 1 respectively. Therefore, state c can be removed and replaced by a. Also state e can be removed and replaced by state b. The final reduced state table is shown in Figure 6.121a. The state diagram for the reduced state table consists of only four states and is shown in Figure 6.121b.

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	b	d	0	1
b	b	f	0	1
d	b	b	0	1
f	f	a	0	0

(a) State table



(b) State diagram

Figure 6.121 Reduced state table and state diagram.

State assignment: The process of assigning binary values to the states of the sequential machine is known as *state assignment*. The binary values are to be assigned to the states in such a way that

it is possible to implement flip-flop input functions using minimum logic gates. The output values of the physical devices are referred to as state variables.

Rules for state assignment: There are two basic rules for making state assignments.

Rule 1. States having the same NEXT STATE for a given input condition should have assignments which can be grouped into logically adjacent cells in a K-map.

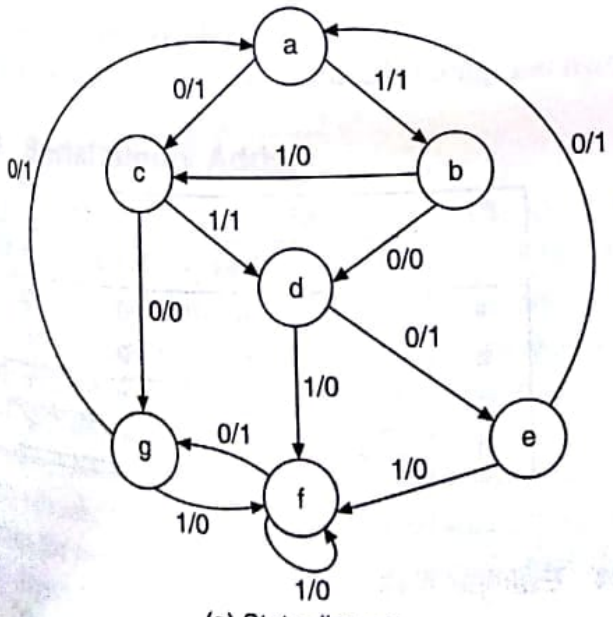
Rule 2. States that are the next states of a single state should have assignments which can be grouped into logically adjacent cells in a K-map.

Transition and output table: The transition and output table can be obtained from the state table by modifying the entries of the state table to correspond to the states of the machine in accordance with the selected state assignment. In this table, the next state and output entries are separated into two sections.

The entries of the next state part of this table define the necessary state transitions of the machine and, thus, specify the next values of the outputs of the FFs used. The next state part of the state table is called the *transition table*. The output part of the table indicates the output of the sequential machine for various input combinations applied to the machine, which is in the present state.

Excitation table: The excitation table of a sequential machine gives information about the excitations or inputs required to be applied to the memory elements in the sequential circuit to bring the sequential machine from the present state to the next state. It also gives information about the outputs of the machine after application of the present inputs. The minimal expressions for the excitations of the FFs and outputs of the machine can be obtained by minimizing the expressions obtained from the excitation table using K-maps. The circuit can be realized using these minimal expressions.

EXAMPLE 6.15 Obtain reduced state table and reduced state diagram for the sequential machine whose state diagram is shown in Figure 6.122a.



(a) State diagram

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	c	b	1	1
b	d	c	0	0
c	(g)e	d	0	1
d	e	f	1	0
e	a	f	1	0
f	(g)e	f	1	0
g	a	f	1	0

(b) State table

Figure 6.122 Example 6.15.

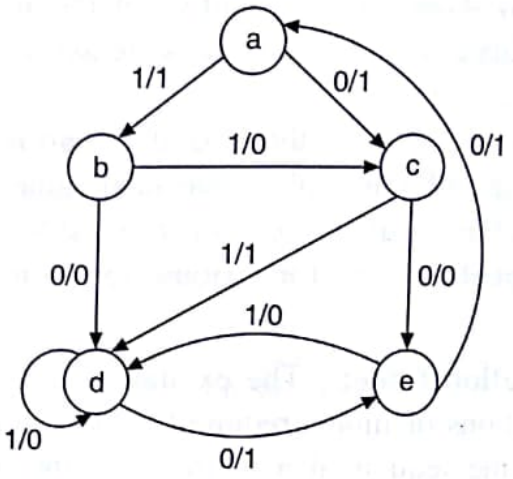
Solution

The state table for the given sequential circuit will be as shown in Figure 6.122b.

From the state table of Figure 6.122b we observe that states g and e are equivalent because they have the same next state and the same output for each one of the inputs. So one of them becomes redundant and can be removed. Let us remove state g. Replacing g by e in the state table we observe that states d and f are equivalent. So one of them becomes redundant and can be removed. Let us remove f and replace f by d. So we are left with five states a, b, c, d, and e. The reduced state table is shown in Figure 6.123a. The reduced state diagram is shown in Figure 6.123b.

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	c	b	1	1
b	d	c	0	0
c	e	d	0	1
d	e	d	1	0
e	a	d	1	0

(a) State table



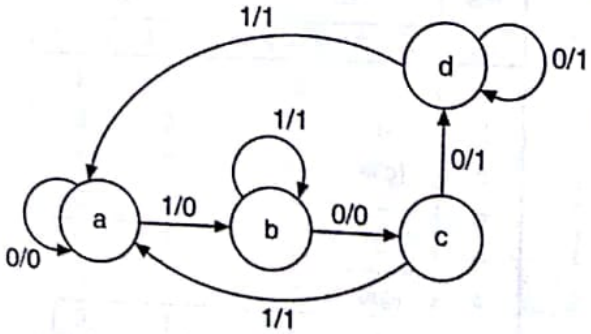
(b) State diagram

Figure 6.123 Example 6.15: Reduced state table and state diagram.

EXAMPLE 6.16 Obtain a reduced state table and reduced state diagram for the sequential machine whose state diagram is shown in Figure 6.124a.

Solution

The state table for the given state diagram is shown in Figure 6.124b. From the state table we observe that states c and d are equivalent. So state d can be removed and d is replaced by c at other places. The reduced state table is shown in Figure 6.125a. The reduced state diagram is shown in Figure 6.125b.



(a) State diagram

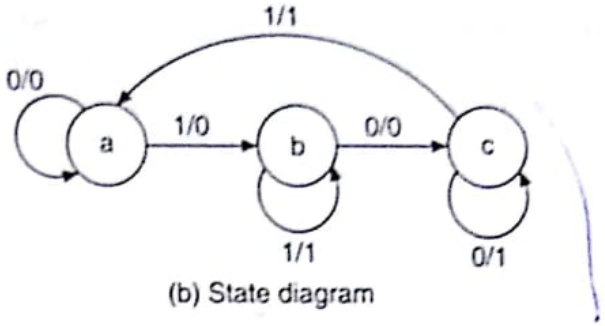
PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	1
c	d	a	1	1
d	d	a	1	1

(b) State table

Figure 6.124 Example 6.16.

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	1
c	c	a	1	1

(a) State table



(b) State diagram

Figure 6.125 Example 6.16: Reduced state table and state diagram.

Design steps: The main steps in the general method for designing sequential circuits using various types of memory elements are as follows:

Step 1. Word statement: State the purpose of the machine in simple, unambiguous words. It should be realizable with a finite number of memory elements.

Step 2. State diagram: Based on the word description of the machine, draw the state diagram which depicts the complete information about it.

Step 3. State table: Write the state table which contains all the information of the state diagram in tabular form.

Step 4. Reduced standard form state table: Remove the redundant states if any in step 2 and write the reduced standard form state table.

Step 5. State assignment and transition table: Assign binary names to the states and write the transition table.

Step 6. Choose type of flip-flops and form the excitation table: Based on the entries in the transition table write the excitation table after choosing the type of FFs.

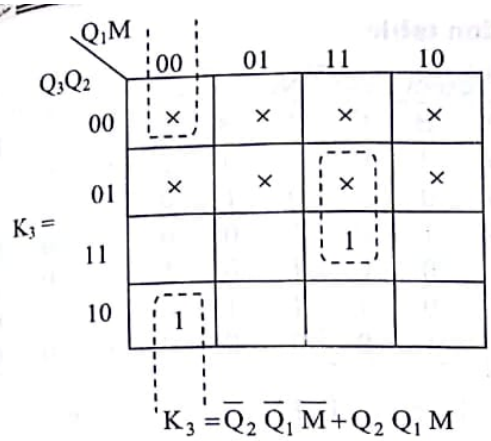
Step 7. K-maps and minimal expressions: Based on the contents of the excitation table draw the K-maps and synthesize the logic functions for each of the excitations as functions of input variables and state variables.

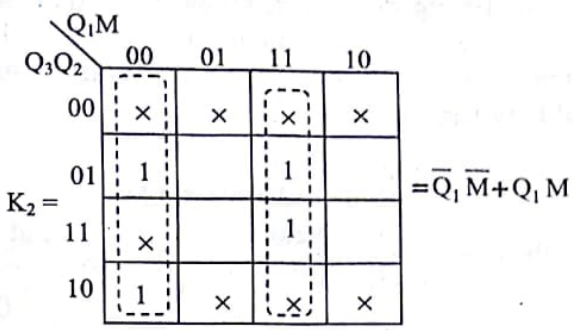
Step 8. Realization: Draw the circuit to realize the minimal expressions obtained above.

6.4.2 Serial Binary Adder

Step 1. Word statement of the problem: The block diagram of a serial binary adder is shown in Figure 6.126. It is a synchronous circuit with two input terminals designated X_1 and X_2 which carry the two binary numbers to be added and one output terminal Z which represents the sum. The inputs and outputs consist of fixed-length sequences of 0s and 1s. The addition is performed serially, i.e. the least significant digits of the numbers X_1 and X_2 arrive at the corresponding input terminals at t_1; a unit time later the next significant digits arrive at the input terminals, and so on. The time interval between the arrivals of two consecutive input digits is determined by the frequency of the circuit's clock. The delay within the combinational circuit is small with respect to the clock frequency and thus the sum digit arrives at the Z terminal immediately following the arrival of the corresponding input digits at the input terminals.

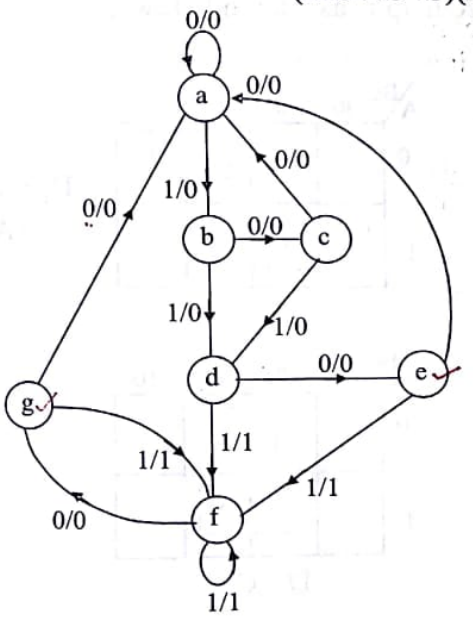
The output of the serial adder z_i at time t_i is a function of the inputs $x_1(t_i)$ and $x_2(t_i)$ at that time t_i and of a carry which had been generated at t_{i-1}. The carry which represents the past history





(41) **Reduce the following state diagram and also write the reduced state table.**

(IES-EC-13)(20M)



There are infinite number of input sequences that can be considered. Consider the input sequence 01010110100 starting from the initial state 'a'.

Sol: Given input sequence 01010110100 starting from the initial state a. Each input of 0 or 1 produces an output of 0 or 1 and causes the circuit to go to the next state.

From the state diagram we obtain the output and state sequence for the given input sequence as follows.

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

To reduce the state diagram we need to construct state table from the given state diagram.

Present state	Next state		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e → f	d	0	1
e	a	f	d	0	1
f	g/e → f		0	1
g/e	a	f	0	1

In the above state table we look for two present states that go to the same next state and have the same output for both input combinations.

States g and e are two such states; they both go to states a and f and have outputs of 0 and 1 for x = 0 and x = 1, respectively.

∴ States g and e are equivalent one can be removed by replacing g as e. Similarly f is replaced by d. These changes shown in reducing state table.

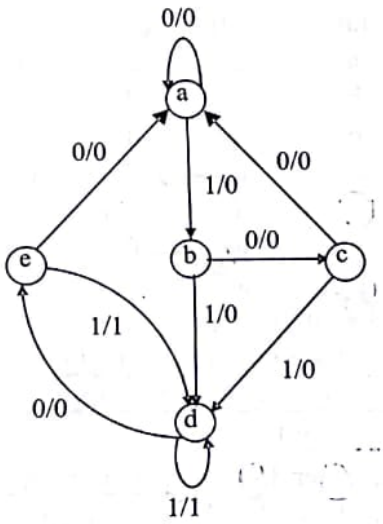
Reducing state table

Present state	Next state		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f d	0	1
e	a	f d	0	1
f	g a	f	0	1
g	a	f	0	1

It can be inferred from above table that both states f & g are identical to states d and e. Hence can be eliminated. So the reduced state table will look like

Reduced state table				
Present state	Next state		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Reduced state diagram:



42. Design a counter that has a repeated sequence of six states as given in the table below: **(IES-EC-13)(20M)**

Count sequence		
A	B	C
0	0	0
0	0	1
0	1	0
1	0	0
1	0	1
1	1	0

Sol: No. of states = $8 = 2^m$

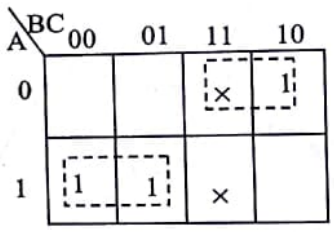
$\therefore$ No .of flip flops (m) = 3

Excitation table:

Present state			Next state			D-flip flop		
A	B	C	A_n	B_n	C_n	D_1	D_2	D_3
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	0	0	0	0	0	0

The choice of D-FF result's in the excitation table shown above. The FF input functions can be simplified using minterms 3 and 7 as don't care conditions. The simplified functions using K-map is as shown below.

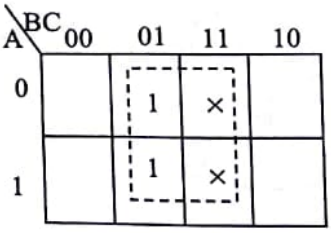
D_1:



$D_1 = \overline{A}B + \overline{B}A$

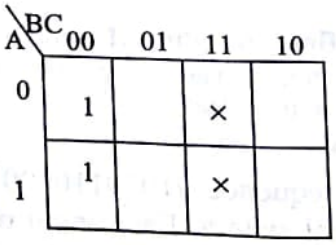
$= A \oplus B$

D_2:



$D_2 = C$

D_3:



$D_3 = \overline{B}\,\overline{C}$

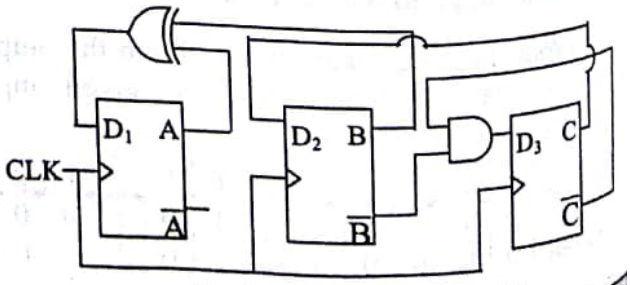


Fig: Logic diagram of counter

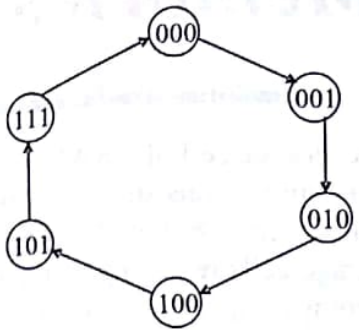
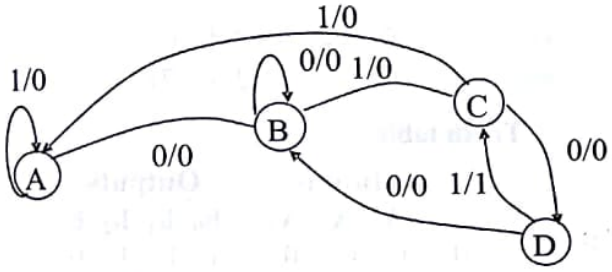


Fig: State diagram of counter

43. Design a two-input, two –output sequence detector which produces an output '1' every time the sequence 0101 is detected and an output '0' otherwise.

(IES-EC-14)(10M)

Sol: State Diagram:



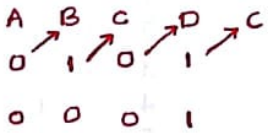
let

A = 00

B = 01

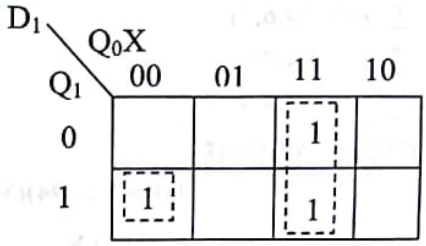
C = 10

D = 11

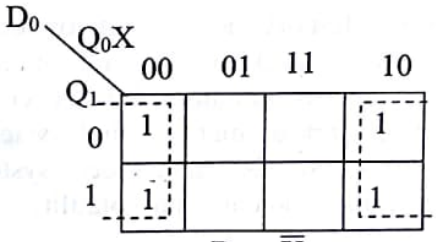


Present state Q	Next state X=0	Next state X=1	O/P X=0	O/P X=1	Z
A	B	A	0	0	0
B	B	C	0	0	0
C	D	A	0	0	0
D	B	C	0	1	1

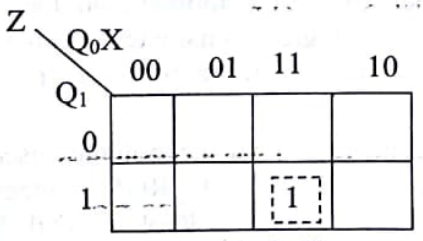
Present state Q_1 Q_0 X	Next state Q_1^+ Q_0^+	D - FF D_1 D_0	output Z
0 0 0	0 1	0 1	0
0 0 1	0 0	0 0	0
0 1 0	0 1	0 1	0
0 1 1	1 0	1 0	0
1 0 0	1 1	1 1	0
1 0 1	0 0	0 0	0
1 1 0	0 1	0 1	0
1 1 1	1 0	1 0	1



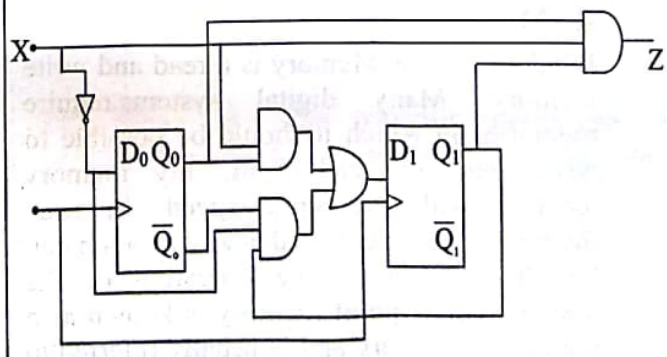
$D_1 = Q_0X + \overline{Q}_0Q_1\overline{X}$

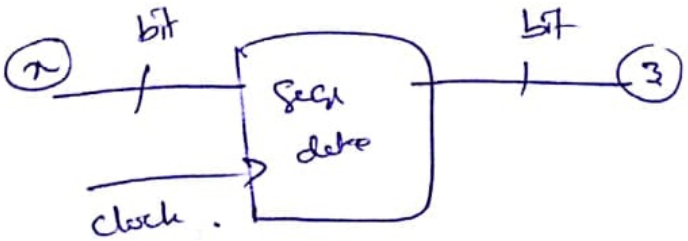


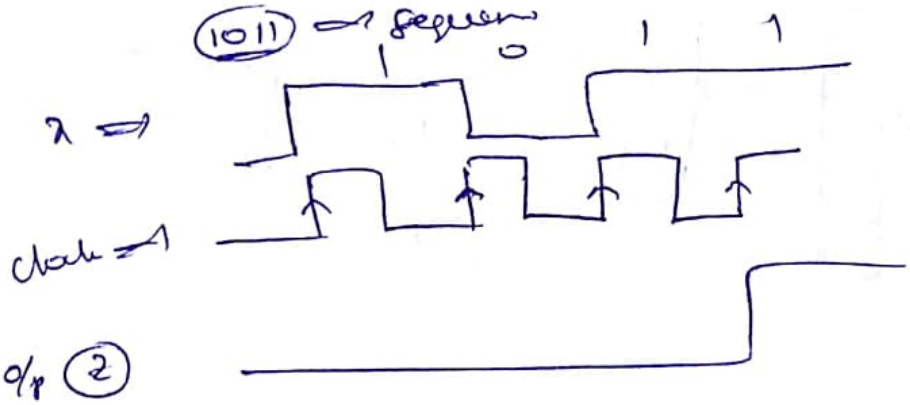
$D_0 = \overline{X}$

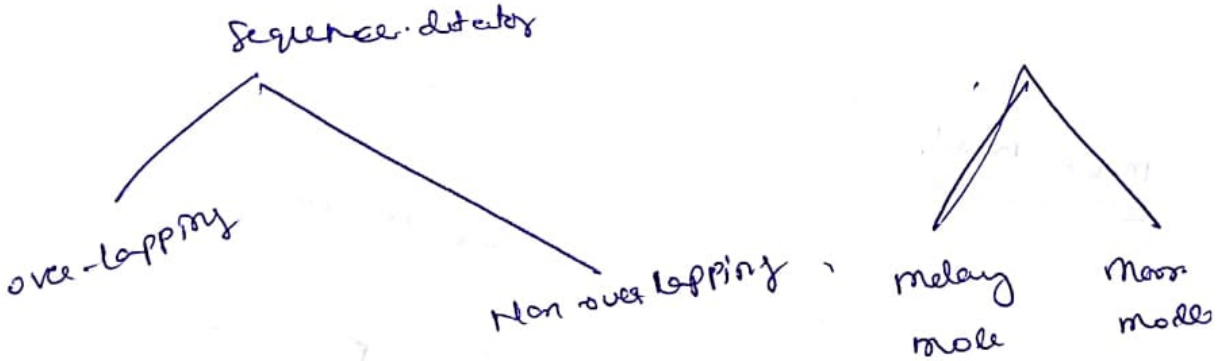


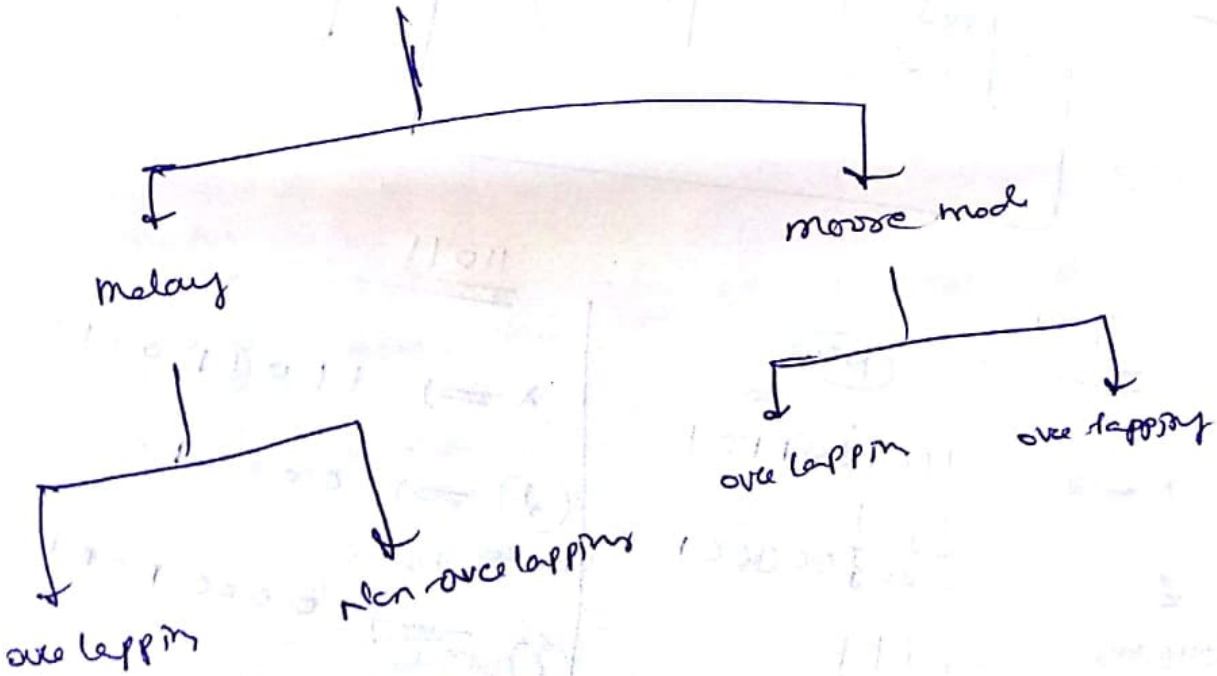
$Z = Q_0Q_1X$



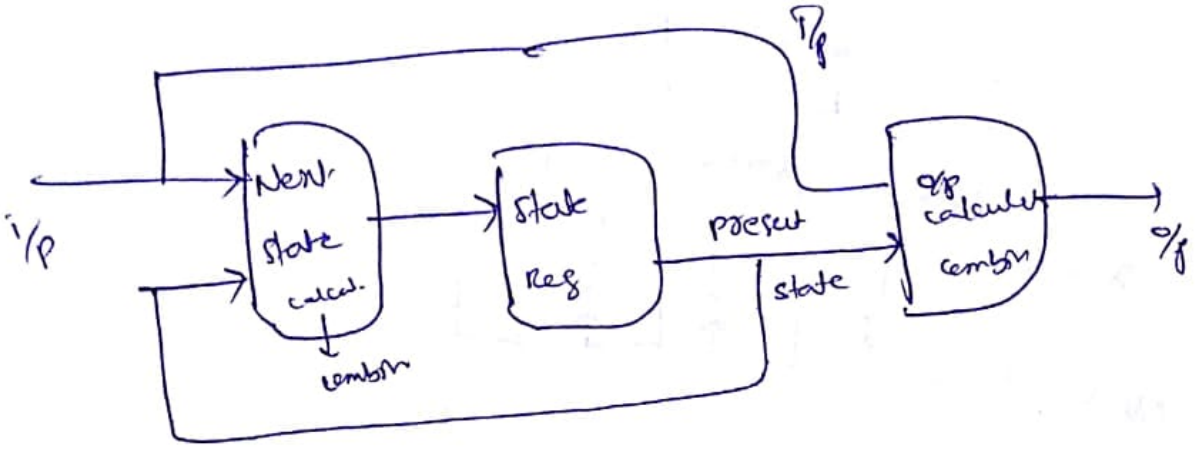




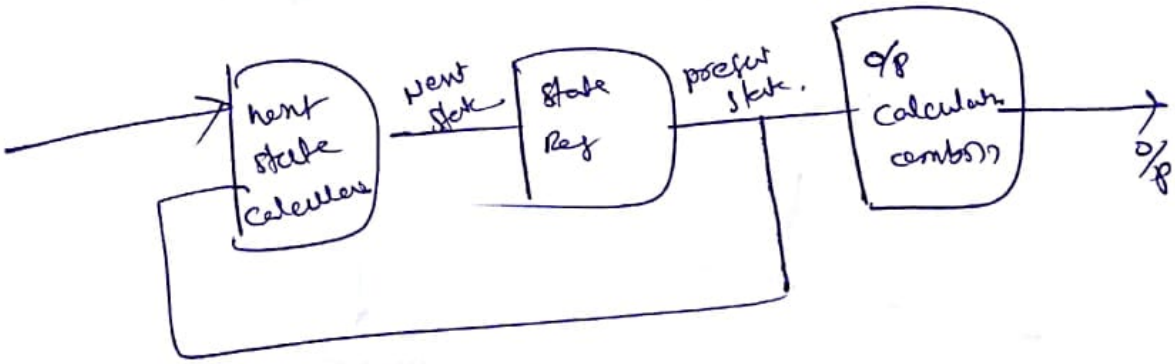




Mealy model



moore model:



1101

NAND

x ⟹ 1 1 0 1 1 0 1 1 0 1

0 0 0 1 0 0 0 1 0 0 1

Z (Non-overlapping)

2

1 bit OL

11011

x ⟹ 1 1 0 1 1 0 1 1 0 1 1

2 ⟹ 0 0 0 0 1 0 0 0 0 0 1

Non-overlapping

⟹ 0 0 0 0 1 0 0 1 0 0 1

2

2 bit OL

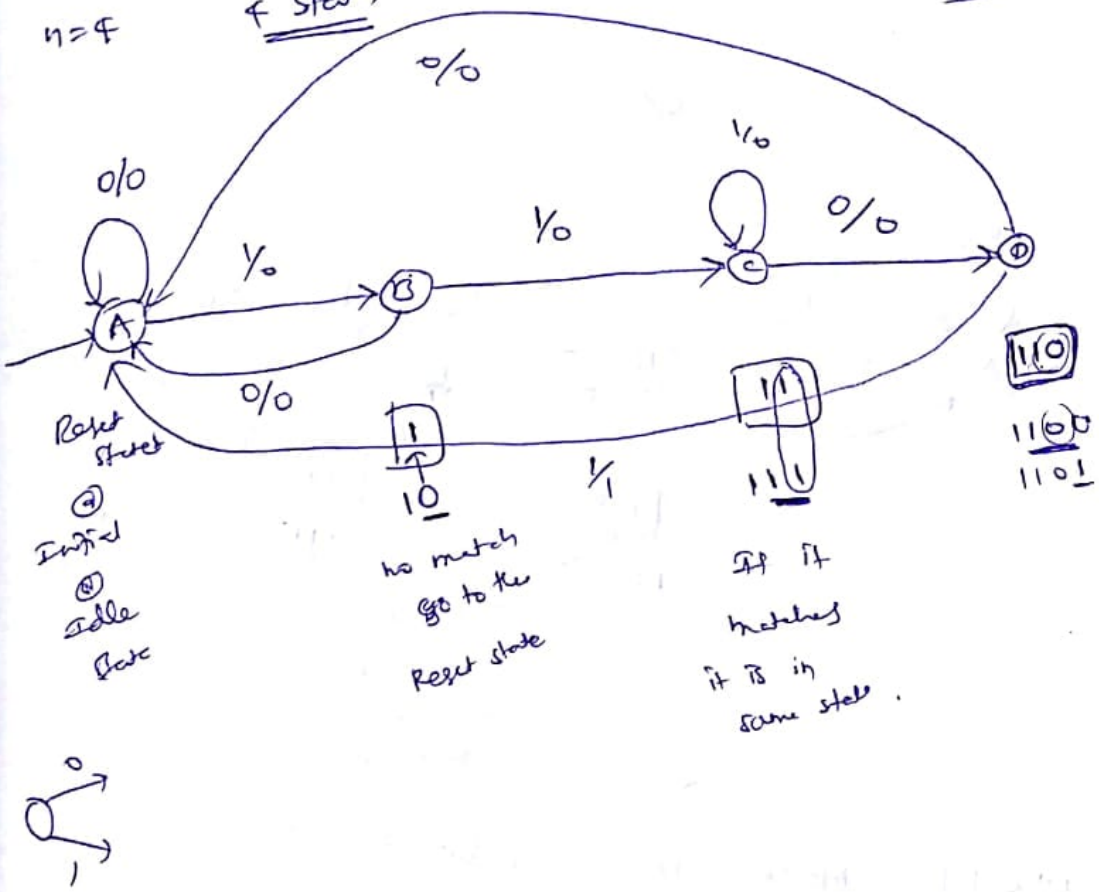
1 1 0 1 1 0 1 1 0 1 1

Mealy model: n bit ⇒ (n) states

more mode ⇒ (n+1) states

1101 ⇒ Mealy model (Non-OL) ⇒ (2) ⇒ 1101 / bit ⇒ Square detector

n=4 ⇒ 4 States



Mealy Model

01. for non-OL, move the last bit to reset state

02. for 1-bit OL, compare the last bit to 1-bit state

03. for 2-bit OL, compare the last bit to 2-bit state.

then 1 bit to 1-bit state and so on

more model

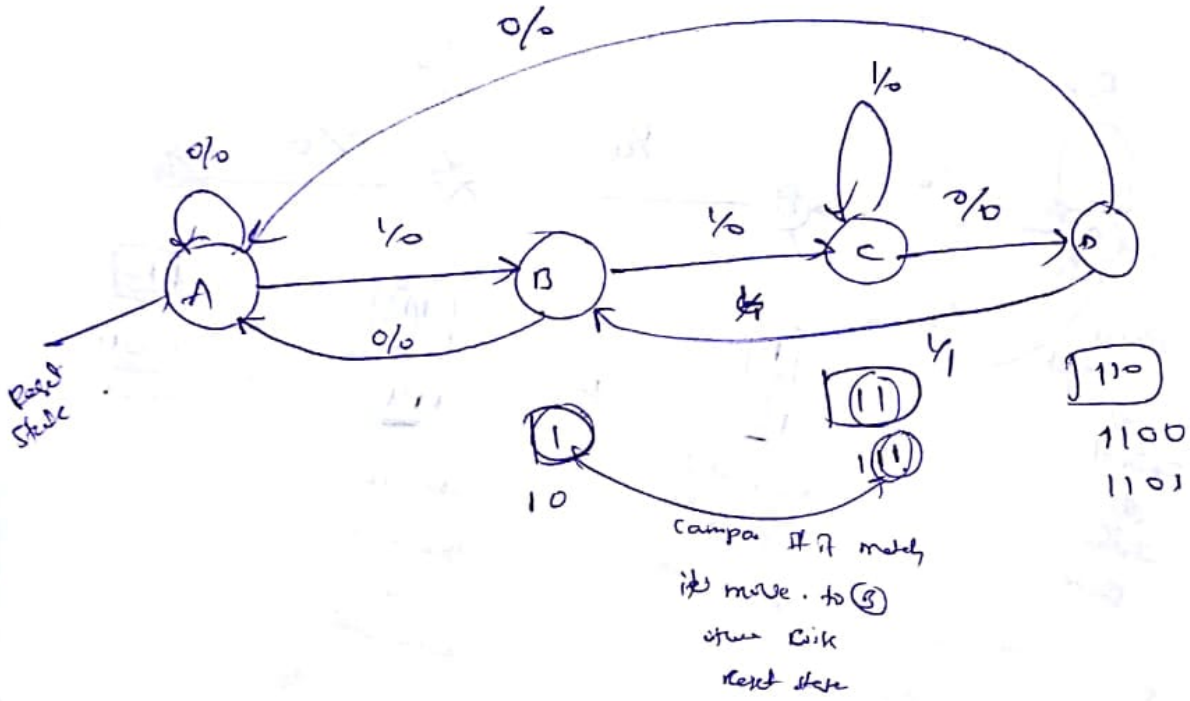
01. for non-OL, compare the last bit to 1-bit state

02. for 1-bit OL, compare the last 2-bits to 2 bit state then 1-bit to 1-bit state and so on

03. for 2-bit OL, compare the last 3-bits to 3-bit state then 2 bits to 2 bits state and so on

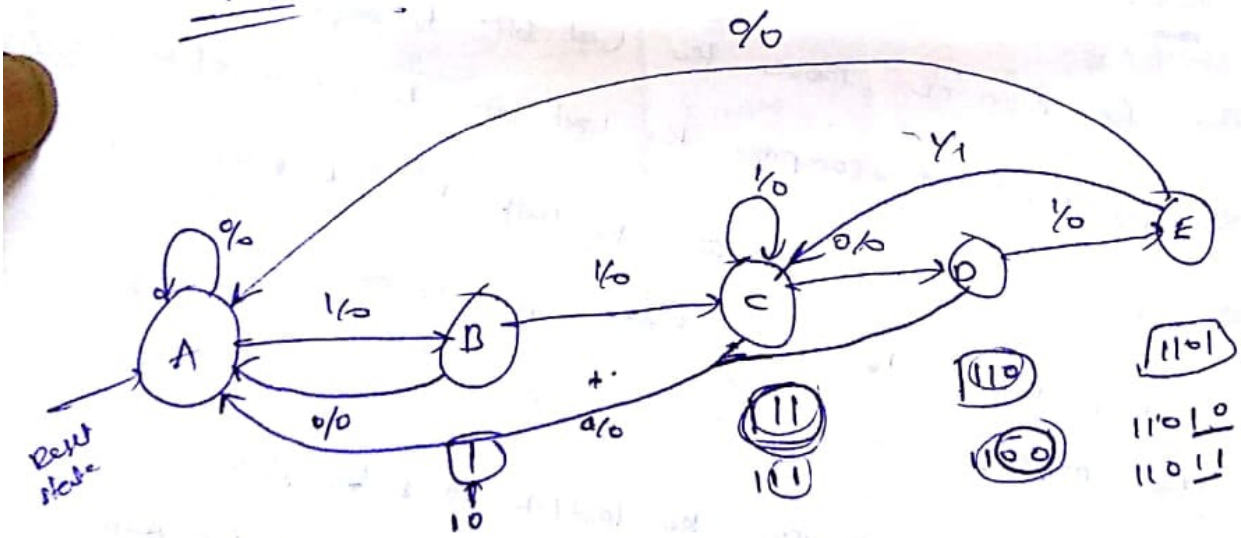
1101 mealy model 1-bit over lap

1101011 n=4 4 states



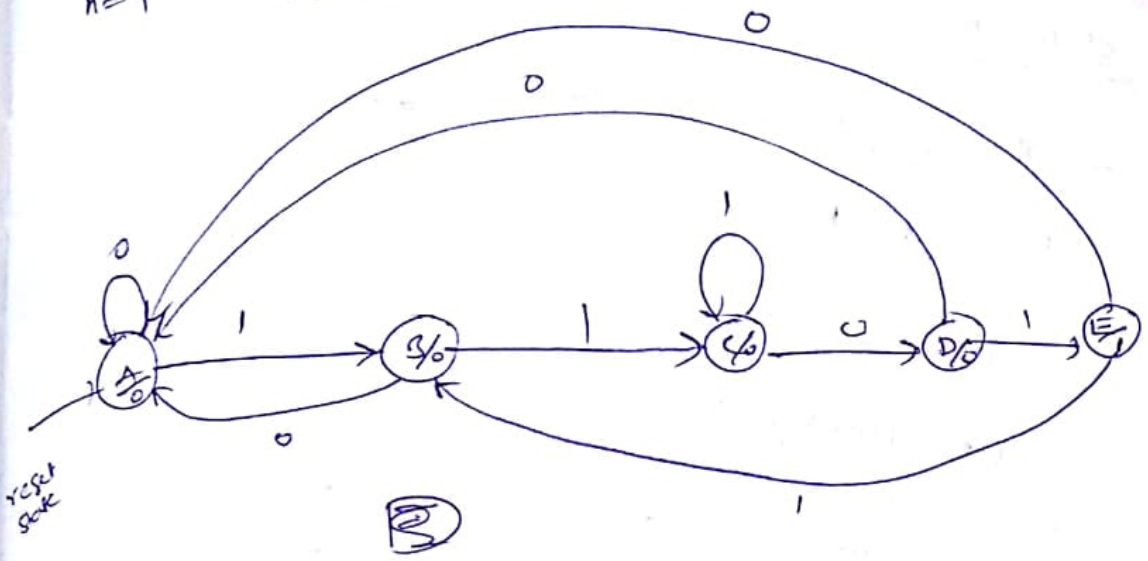
11011 Mealy model 2-bit over lapping

n=5 5 states



Ex: 1101 Moore model. Non-OL

n=4 n+1 = states ⇒ (5) states

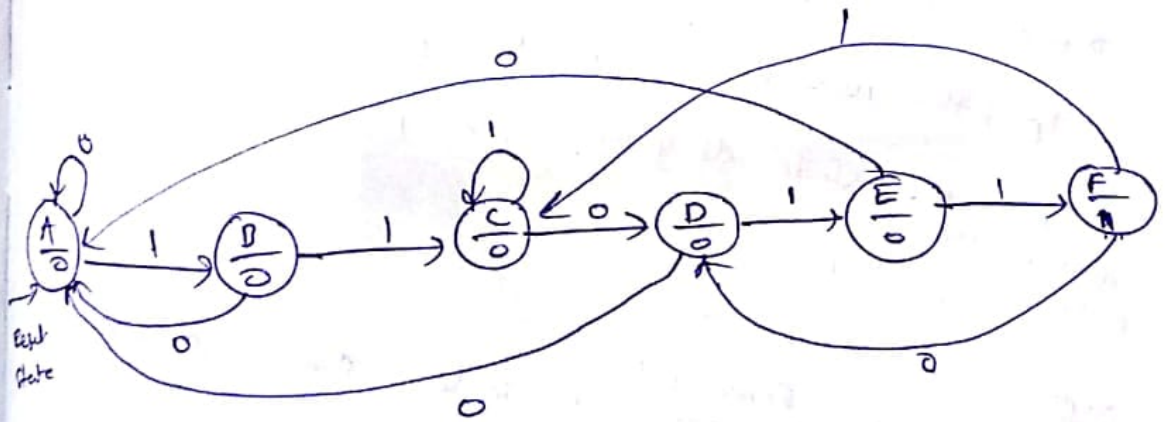


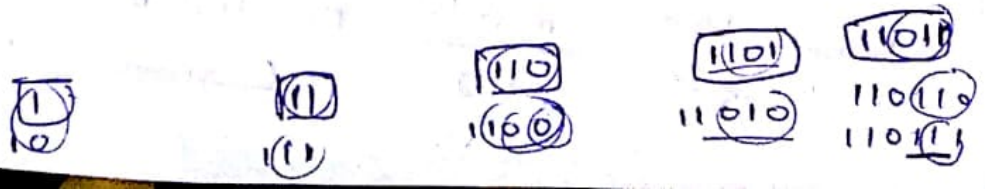
(2)

11011 Moore model 2-bit OL

n=5 n+1 = (6) 1100(11)011

2-bit OL

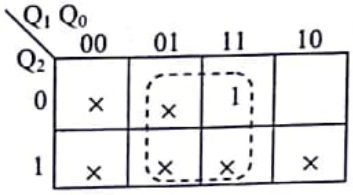




$K_2 = \sum m(4) + \sum d(0, 1, 2, 3, 5, 6, 7) = 1$

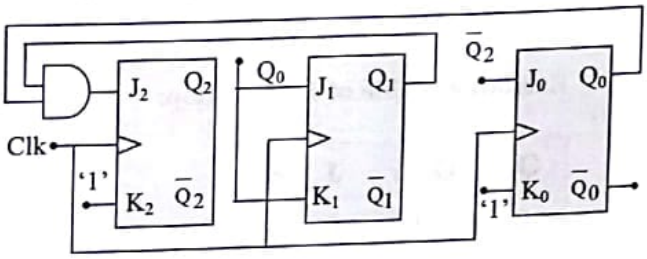
$K_2 = 1$

$K_1 = \sum m(3) + \sum d(0, 1, 4, 5, 6, 7)$

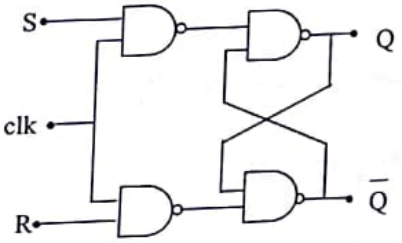


$K_1 = Q_0$

$K_0 = \sum m(1, 3) + \sum d(0, 2, 4, 5, 6, 7) = 1$



17. Synthesize an SR clocked flip flop using basic gates give input truth table. **(IES-EC-92)(8M)**



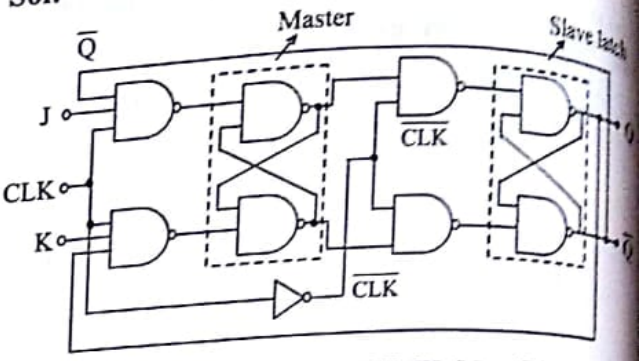
Sol:

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	×
1	1	1	×

S	R	Q_{n+1}
0	0	NC
0	1	Reset
1	0	Set
1	1	Indeterminate

18. Convert a JK flip flop in to a JK Master - slave flip flop. **(IES-EC-92&97)(8M)**

Sol:



Master slave JK FLIP-FLOP: The race around problem is eliminated in master slave FLIP FLOP, because while the clock drives the master. The inverted clock drives the slave

19. Explain about Ripple counters. **(IES-EC-94)(7M)**

Sol: Ripple counters: Asynchronous counters are also called ripple counters. It is easier to design and requires the least amount of hardware. In ripple counters the flip flops with in the counter are not made to change the states at exactly the same time. This is because the flip flops are not triggered simultaneously. The transition of the first stage ripples through to the last stage, so it is called ripple counter.

20. Design a synchronous sequential circuit, using JK flip flops, with one input line, x and one output line, Z. The circuit is to recognize the occurrence of the sequence 1111 in the input string. Overlapping occurrences are also to be recognized for example.

If x = 00110 1111 010 111110 .....

The Z = 00 000 0001 000 000110 ..... **(IES-EC-94)(20M)**

Sol: The sequence in the input string is 1111

Let ABCD are 4 states.

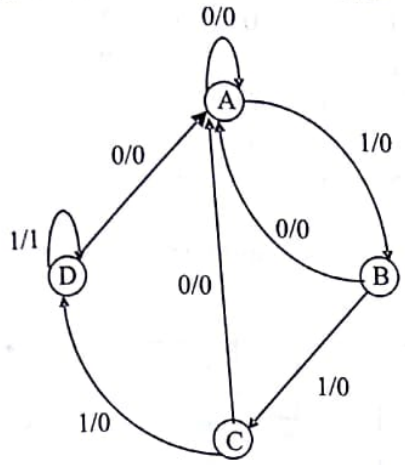
A = Initial state

B = 1__

C = 11__

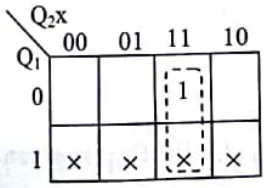
D = 111__

State diagram of sequential circuit



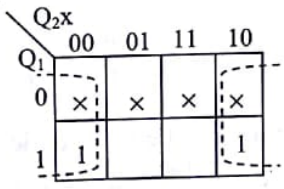
Present state			Next state		Output	Flip Flop inputs			
Q_1	Q_2	x	Q_1	Q_2	Z	J_1	K_1	J_2	K_2
0	0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	0	×	1	×
0	1	0	0	0	0	0	×	×	1
0	1	1	1	0	0	1	×	×	1
1	0	0	0	0	0	×	1	0	×
1	0	1	1	1	0	×	0	1	×
1	1	0	0	0	0	×	1	×	1
1	1	1	1	1	1	×	0	×	0

J_1 :



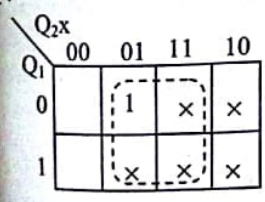
$J_1 = Q_2x$

K_1 :



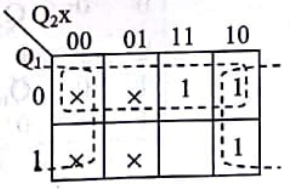
$K_1 = \overline{x}$

J_2 :



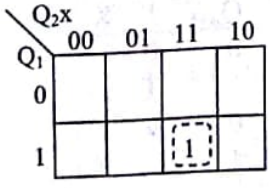
$J_2 = x$

K_2 :

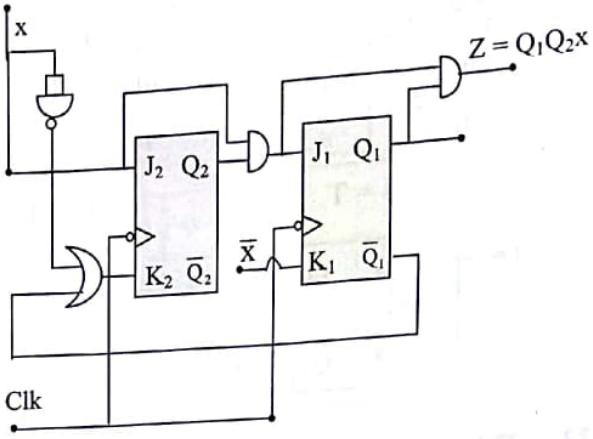


$K_2 = \overline{x} + \overline{Q}_1$

Z :



$Z = Q_1Q_2x$



21. How can JK flip flop be used as
i) D -flip flop and
ii) T – flip flop? Justify your answer with the help of truth tables?
(IES-EC-96)(10M)

Sol: i) Conversion of JK flip flop to D flip flop:
(Refer to Q no: 30)

ii) JK flip flop to T flip flop:-

T flip flop Excitation table:-

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

T	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

Conversion table for JK flip-flop to T flip-flop:

T	Q_n	Q_{n+1}	J	K
0	0	0	0	×
0	1	1	×	0
1	0	1	1	×
1	1	0	×	1

T	J	K	Q_{n+1}
0	0	0	0
1	1	1	$\overline{Q}_n$

04. Design a sequence detector which produces an output of 1 every time the sequence 0 1 0 1 is detected and zero at all other times.

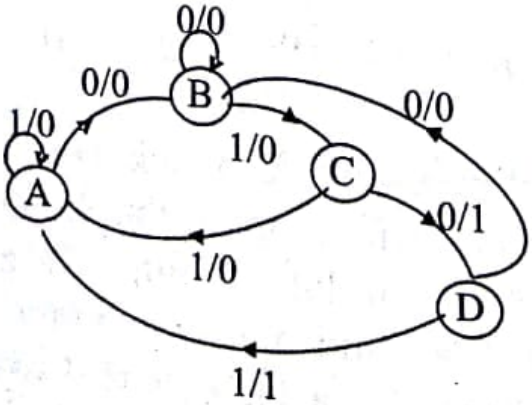
(IES-EC-83)(10M)

Sol: Sequence Detector:

The sequence detector is a clocked synchronous state machine. The state of a sequential circuit is a result of all previous input and determines the circuit's output and future behavior. This is why sequential circuits are often referred to as state machines.

Most sequential circuits use of clock signal to control when the circuit change states. The inputs of the circuit along with the circuit's current state provide the information to determine the circuits next state. The clock signal then controls the passing of this information to the state memory. The output depends only on the circuits state, this is known as a more machine.

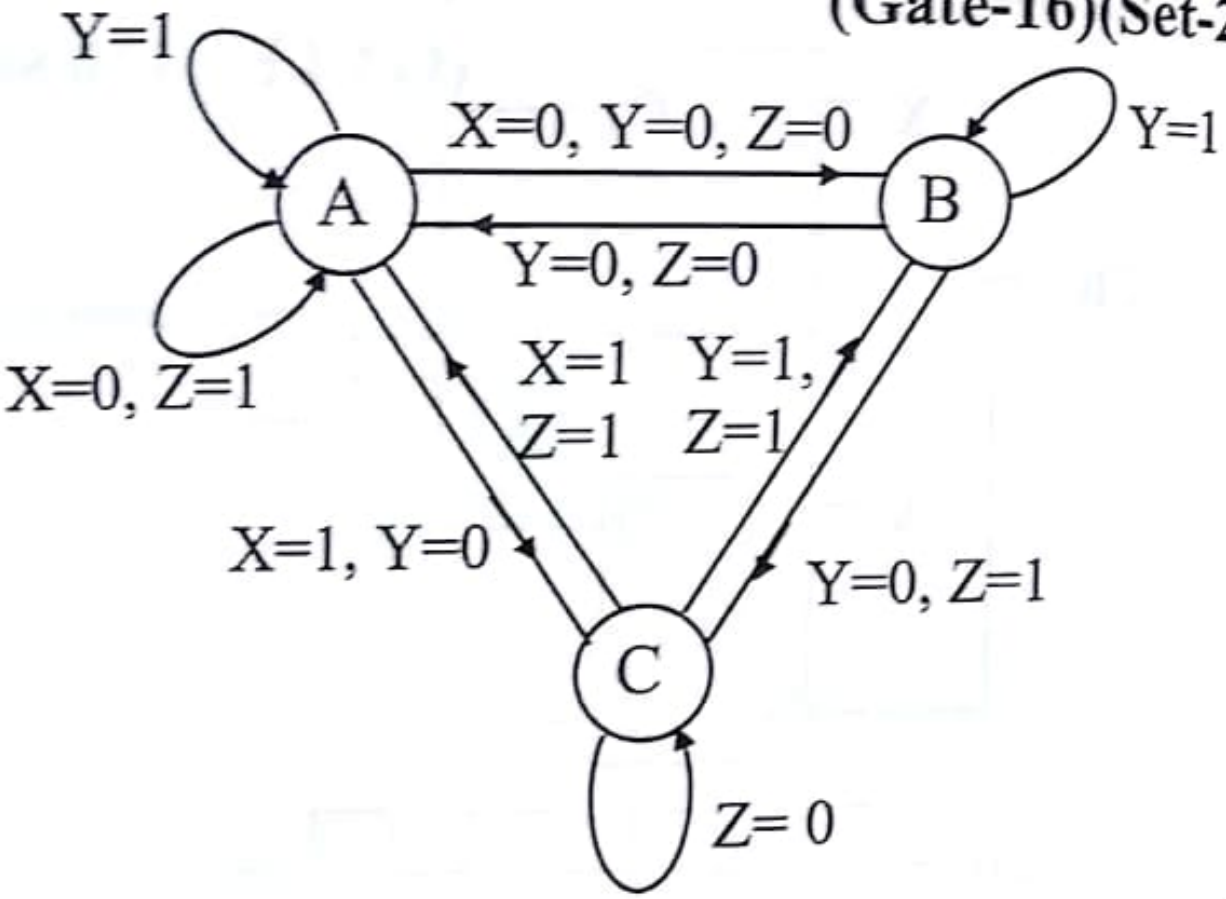
State diagram for 4 – bit sequence 0101:



05. Design a

22. The state transition diagram for a finite state machine with states A, B and C, and binary input X, Y and Z, is shown in the figure.

(Gate-16)(Set-2)



Which one of the following statements is correct?

(a) Transitions from State A are ambiguously defined

(b) Transition from State B are ambiguously defined

(c) Transitions from State C are ambiguously defined

(d) All of the state transitions are defined unambiguously.

considered as logic 1 thus 'set' , 'Reset' inputs are always logic '1'. To give input of '0', +5V should be replaced by 0V.

22. Ans: (c)

Sol: In state (C), when XYZ = 111, then Ambiguity occurs

Because, from state (C)

$\Rightarrow$ When X = 1, Z = 1

$\Rightarrow$ N.S is (A)

When Y = 1, Z = 1 $\Rightarrow$ N.S is (B)

23. Ans: (d)

Sol: At 6th Clk pulse (i.e. at 6ns)

$\Rightarrow Q_2Q_1Q_0 = 110 \Rightarrow$ It makes NAND gate output '0' at 8ns due to its delay. By that time counter receives 7th, 8th Clk pulse and counts 111, 000. Thus it is a Mod - 8 counter

24. Ans: 10

Sol:

Clk	A	B	C	D
0	1	1	0	1
1	0	1	1	0
2	0	0	1	1
3	1	0	0	1
4	0	1	0	0
5	0	0	1	0
6	0	0	0	1
7	1	0	0	0
8	1	1	0	0
9	1	1	1	0
10	1	1	1	1 $\Rightarrow$ required state

25. Ans: (d)

Sol: If $X_{IN} = 0$

Clk	$D_A = Q_A \oplus Q_B$	D_B=1	Q_A	Q_B
0	-	-	0	0
1	0	1	0	1
2	1	1	1	1
3	0	1	0	1

Thus number of possible states are two

If $X_{IN} = 1 \Rightarrow D_B = \overline{Q_A}.1 = \overline{Q_A}$

Clk	$D_A = Q_A \oplus Q_B$	$D_B = \overline{Q_A}$	Q_A	Q_B
0			0	0
1	0	1	0	1
2	1	1	1	1
3	0	0	0	0

Thus number of possible states are three

26. Ans: 4

Sol: Given input sequence

1 0 1 0 1 1 0 1 0 0 1 1 0 1

1 2 3 4

Number of times out will be 1 is 4

Conventional Solutions

01.

Sol:

CLK				FF1	FF0	
	M	Q_1	Q_0	T_1	T_0	
	0	0	0	1	1	M = 0 i.e. Down counter
1	0	1	1	0	1	
2	0	1	0	1	1	
3	0	0	1	0	1	
4	0	0	0			
	1	0	0	0	1	M = 1 i.e. Up counter
1	1	0	1	1	1	
2	1	1	0	0	1	
3	1	1	1	1	1	
4	0	0	0			

$T_1(M, Q_1, Q_o) = \sum m(0,2,5,7)$ ..... (1)

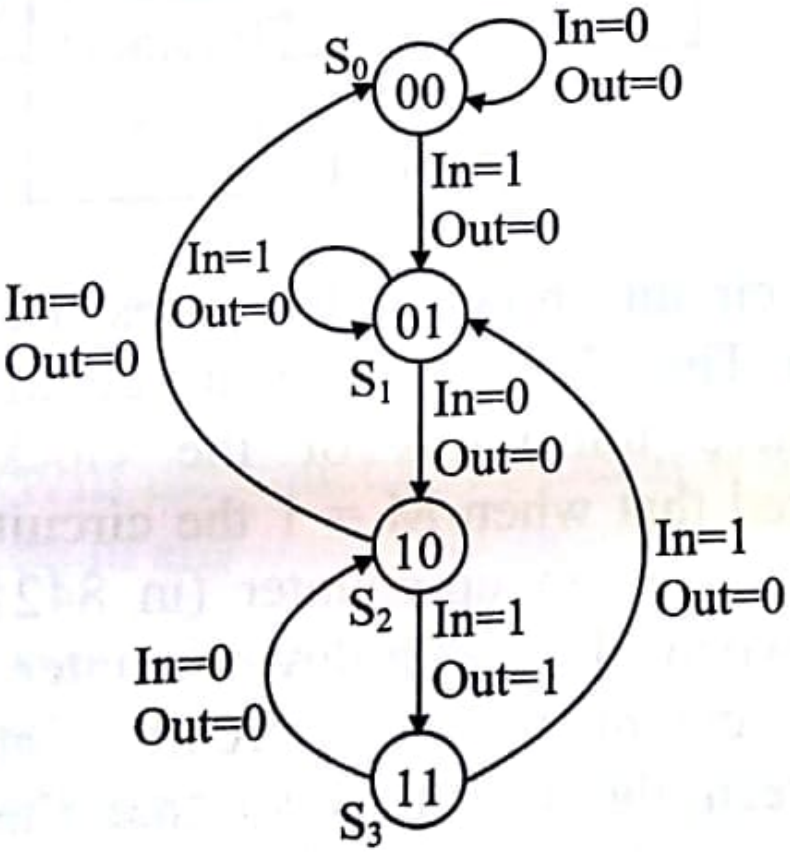
$= \overline{M}\,\overline{Q_0} + MQ_o = M \odot Q_0$

$T_0(M, Q_1, Q_o) = \sum m(0,1,2,3,4,5,6,7)$ ..... (2)

$= 1$

	$Q_1Q_o = S_1S_o$			
	I_o	I_1	I_2	I_3
	00	01	10	11
$\overline{M} = 0$	0	1	2	3
$M = 1$	4	5	6	7
	$\overline{M}$	M	$\overline{M}$	M

26. The state diagram of a finite state machine (FSM) designed to detect an overlapping sequence of three bits is shown in the figure. The FSM has an input 'In' and an output 'out'. The initial state of the FSM is S_0.



If the input sequence is 10101101001101, starting with the left most bit, then the number of times 'Out' will be 1 is _____

(Gate-17) (Set 2)

two states each. Instead of a we will use a_0 and a_1 to denote the fact that the carry is 0 and the sum is either 0 or 1 respectively. When the state is splitted, we have to direct transition associated with output 0 to state 0 and transition associated with output 1 to state 1. So state a with output 1 is directed to state a_1 and state a without 0 is directed to state a_0. Figure 6.130b shows the state table for the Moore type serial adder. Once, the Moore type state diagram and state table are drawn the Moore circuit can be designed following the normal procedure.

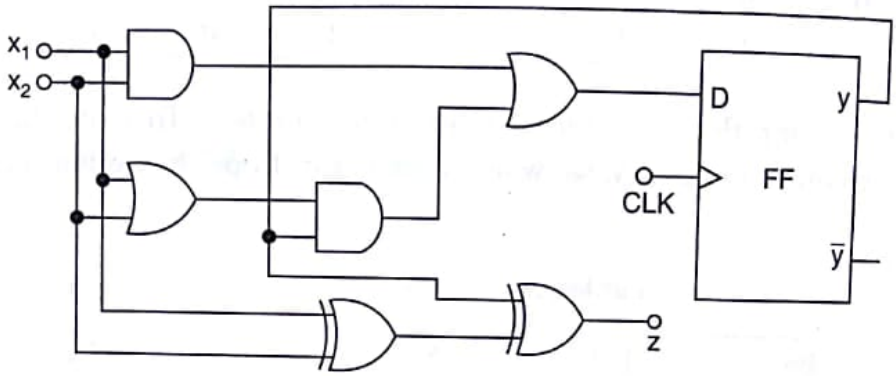
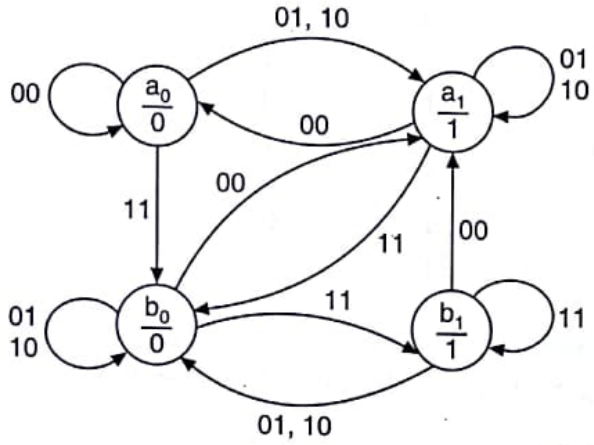


Figure 6.129 Logic diagram of the serial binary adder.



(a) State diagram

PS	NS / Input				O/P (sum)
	00	01	10	11	
a_0	a_0	a_1	a_1	b_0	0
a_1	a_0	a_1	a_1	b_0	1
b_0	a_1	b_0	b_0	b_1	0
b_1	a_1	b_0	b_0	b_1	1

(b) State table

Figure 6.130 Moore type serial adder.

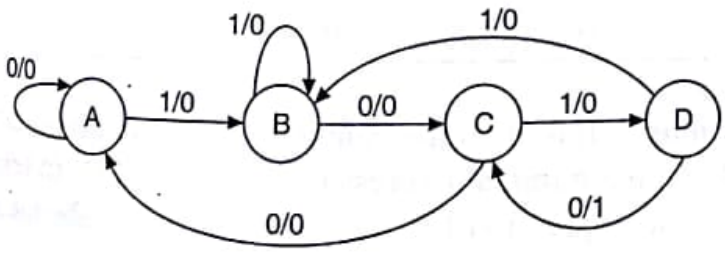
6.4.3 The Sequence Detector

Step 1. *Word statement of the problem:* A sequence detector is a sequential machine which produces an output 1 every time the desired sequence is detected and an output 0 at all other times.

Suppose we want to design a sequence detector to detect the sequence 1010 and say that overlapping is permitted, i.e. for example, if the input sequence is 01101010 the corresponding output sequence is 00000101. We can start the synthesis procedure by constructing the state diagram of the machine.

Steps 2 *and* 3. *State diagram and state table:* The state diagram and the state table of the sequence detector are shown in Figure 6.131. At time t_1, the machine is assumed to be in the

initial state designated arbitrarily as A. While in this state, the machine can receive first bit input, either a 0 or a 1. If the input bit is a 0, the machine does not start the detection process because the first bit in the desired sequence is a 1. So, it remains in the same state and outputs a 0. Hence, an arc labelled 0/0 starting and terminating at A is drawn. If the input bit is a 1, the detection process starts. So, the machine goes to state B and outputs a 0. Hence, an arc labelled 1/0 starting at A and terminating at B is drawn. While in state B, the machine may receive a 0 or a 1 bit. If the bit is a 0, the machine goes to the next state, say state C, because the previous two bits are 10 which are a part of the valid sequence, and outputs a 0. So, draw an arc labelled 0/0 from B to C. If the bit is a 1, the two bits become 11 and this is not a part of the valid sequence. Since overlapping is permitted, the second 1 may be used to start the sequence, so, the machine remains in state B only and outputs a 0. So, an arc labelled 1/0 starting and terminating at B is drawn. While in state C, the machine may receive a 0 or a 1 bit. If it receives a 0, the last three bits received will be 100 and this is not a part of the valid sequence and also none of the last two bits can be used to start the new sequence. So, the machine goes to state A to restart the detection process and outputs a 0. Hence, an arc labelled 0/0 starting at C and terminating at A is drawn. If it receives a 1 bit, the last three bits will be 101 which are a part of the valid sequence. So, the machine goes to the next state, say state D, and outputs a 0. So, an arc labelled 1/0 is drawn from C to D. While in state D, the machine may receive a 0 or a 1. If it receives a 0, the last four bits become 1010 which is a valid sequence and the machine outputs a 1. Since overlapping is permitted, the machine can utilize the last two bits 10 to get another 1010 sequence. So, the machine goes to state C (if overlapping is not permitted or the machine has to restart after outputting a 1, the machine goes to state A). So, an arc labelled 0/1 is drawn from D to C. If it receives a 1 bit, the last four bits received will be 1011 which is not a valid sequence. So, the machine outputs a 0. Since the fourth bit 1 can become the starting bit for the valid sequence, the machine goes to state B. So, an arc labelled 1/0 is drawn from D to B.



(a) State diagram

PS	NS, z	
	x = 0	x = 1
A	A, 0	B, 0
B	C, 0	B, 0
C	A, 0	D, 0
D	C, 1	B, 0

(b) State table

Figure 6.131 Sequence (1010) detector.

Step 4. *Reduced standard form state table:* The machine is already in this form. So no need to do anything.

Step 5. *State assignment and transition and output table:* There are four states; therefore, two state variables are required. Two state variables can have a maximum of four states. So, all states are utilized and thus there are no invalid states. Hence, there are no don't cares. Assign the states arbitrarily. Let A → 00, B → 01, C → 10, and D → 11 be the state assignment. With this assignment draw the transition and output Table 6.16.

Table 6.16 Transition and output table

PS (y_1y_2)	NS (Y_1Y_2)		O/P (z)	
	x = 0	x = 1	x = 0	x = 1
A → 0 0	0 0	0 1	0	0
B → 0 1	1 0	0 1	0	0
C → 1 0	0 0	1 1	0	0
D → 1 1	1 0	0 1	1	0

Step 6. Choose type of Flip-flops and form the excitation table: Select the D flip-flops as memory elements and draw the excitation Table 6.17.

Table 6.17 Excitation table

PS		I/P	NS		I/P to FFs		O/P
y_1	y_2	x	Y_1	Y_2	D_1	D_2	z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0

Step 7. K-maps and minimal expressions: Based on the contents of the excitation table, draw the K-maps and simplify them to obtain the minimal expressions for D_1 and D_2 in terms of y_1, y_2, and x as shown in Figure 6.132. The expression for $z(z = y_1, y_2)$ can be obtained directly from Table 6.16.

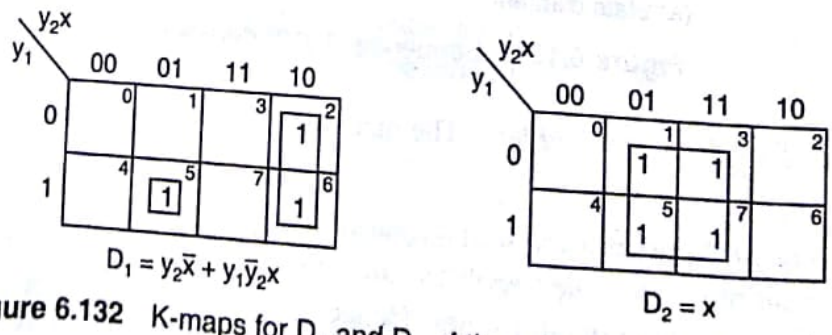


Figure 6.132 K-maps for D_1 and D_2 of the sequence (1010) detector.

Step 8. Implementation: The logic diagram based on these minimal expressions is shown in Figure 6.133.

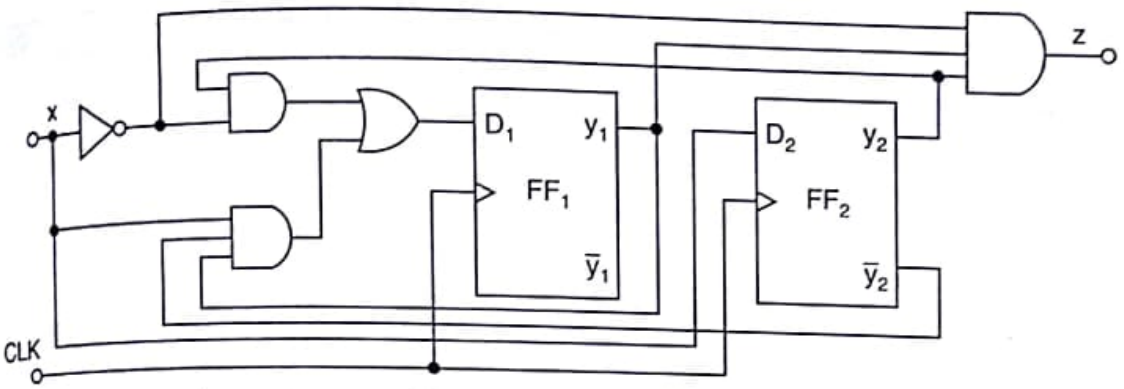
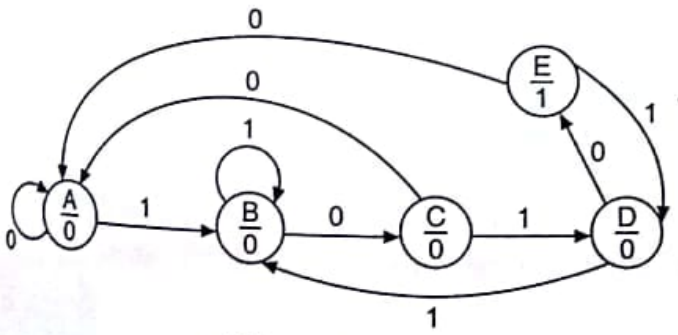


Figure 6.133 Logic diagram of the sequence (1010) detector using D flip-flops.

Moore type circuit: For the design of Moore type of circuit, the same steps are to be followed. The state diagram and the state table of a Moore type sequence detector to detect the sequence 1010 are shown in Figure 6.134. In the Moore type state diagram the outputs are written inside the circle below the state name. The state diagram is drawn in the normal way. The machine will be in state D if the last three bits are 101. If the next bit is a 0, the last four bits will be 1010 which is a valid sequence. So the machine outputs a 1, but to utilize overlapping it cannot go to state C because the output at C is 0. Instead it goes to a new state E where output is equal to 1. While at E, if the next bit is a 0 the last five bits become 10100. So the machine goes to state A to restart the detection. If the next bit is a 1, the last five bits become 10101. So to utilize the last three bits, i.e. 101 it goes to state D. Once, the Moore type state diagram and state table are drawn, the Moore circuit can be designed following the normal procedure.



(a) State diagram

PS	NS		O/P
	X = 0	X = 1	
A	A	B	0
B	C	B	0
C	A	D	0
D	E	B	0
E	A	D	1

(b) State table

Figure 6.134 Moore circuit.

EXAMPLE 6.17 Draw the state diagram and the state table for a Moore type sequence detector to detect the sequence 110.

Solution

The Moore type state diagram and state table of sequence detector to detect the sequence 110 are shown in Figure 6.135. The state diagram is drawn in the normal way. The machine is in state C when the last two bits received are 11. If the next bit is a 0, the last three bits become 110 which is a valid sequence, hence it outputs a 1, but the machine cannot go to state A to restart the detection process because state A outputs a 0. So the machine goes to a new state D and outputs a 1. While at D if the next bit received is a 0, the last four bits will be 1100. So the machine goes to state A to restart the process. If the bit received is a 1, the last four bits will be 1101. So the machine goes to state B to utilize the last bit.