

14. Tabulate the PLA programmable table for the four Boolean functions listed below.

$$A(x,y,z) = \sum m(0,1,2,4,6)$$

$$B(x,y,z) = \sum m(0,2,6,7)$$

$$C(x,y,z) = \sum m(3,6)$$

$$D(x,y,z) = \sum m(1,3,5,7)$$

15. Tabulate the PLA programming table for the four Boolean functions listed below. Minimize the number of product terms.

$$A(x,y,z) = \sum (1,2,4,6)$$

$$B(x,y,z) = \sum (0,1,6,7)$$

$$C(x,y,z) = \sum (2,6)$$

$$D(x,y,z) = \sum (1,2,3,5,7)$$

16. Derive the PLA programming table for the combinational circuit that squares a 3-bit number. Minimize the number of product terms.
17. List the PLA programming table for the BCD to excess 3 code converter whose Boolean functions are simplified.
18. List the PAL programming table for the BCD to excess 3 code converter whose Boolean functions are simplified.
19. The following is a truth table of a 3-input, 4-output combinational circuit. Tabulate the PAL programming table for the circuit and mark the fuse map in a PAL diagram.

Inputs			Outputs			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

Chapter 6

Synchronous Sequential Logic Circuit

6.1 Introduction

A block diagram of a sequential circuit is shown in Fig. 6.1. It consists of a combinational circuit to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information. This binary information stored in these elements at any given time define the state of the sequential circuit at that time.

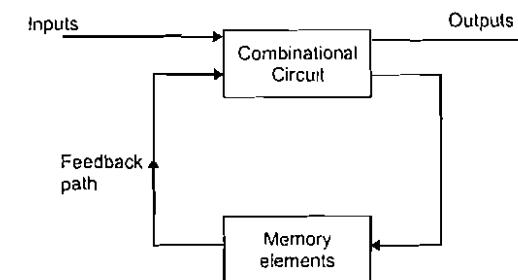


Fig 6.1 Block diagram of sequential circuits

The sequential circuit receives the binary information from external inputs. These inputs and the present state of memory elements determine the binary value of the outputs of the circuit. They also determine the condition for changing the state in memory elements. Thus, the next state of the memory elements is also a function of the external inputs and the present state.

6.1.1 Mealy Model Sequential Circuit

Fig 6.2 shows the clocked synchronous sequential Mealy machine. The output of mealy machine is the function of present inputs and present state (Flip flop outputs). If X is input, Q_n is the present state and the next state is $Q_{(n+1)}$, the output of Mealy function (Z) is given below.

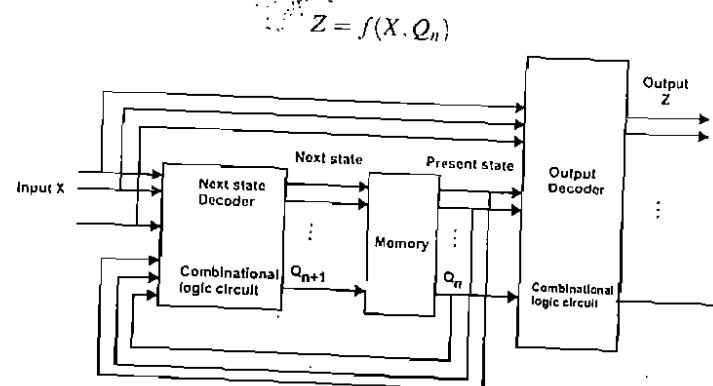


Fig 6.2 Mealy model sequential circuit

The output of memory element is connected to the input of output decoder and next state decoder circuit. The output of memory element is considered as present state.

6.1.2 Moore Model Sequential Circuit

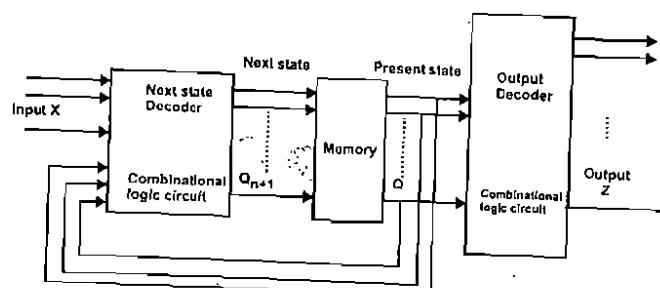


Fig 6.3 Moore model sequential circuit

Fig.6.3 shows the block diagram of a Moore machine. The output of Moore machine depends only on the present state. So the output of Moore machine is a function of its present state (Q_n). If the input is X , the next state is $Q_{(n+1)}$ and the present state is Q_n . The output of Moore machine is represented mathematically by

$$Z = f(Q_n)$$

The differences between the Moore machine and Mealy machine are tabulated as follows

S.No.	Moore machine	Mealy machine
1.	The output of this machine is the function of the present state only.	Its output is function of present input as well as present state.
2.	Input changes do not affect the output	Input changes may affect the output of the circuit
3.	It requires more number of states for implementing same function	It requires less number of states for implementing same function

6.2 Analysis and Synthesis of Synchronous Sequential Circuits

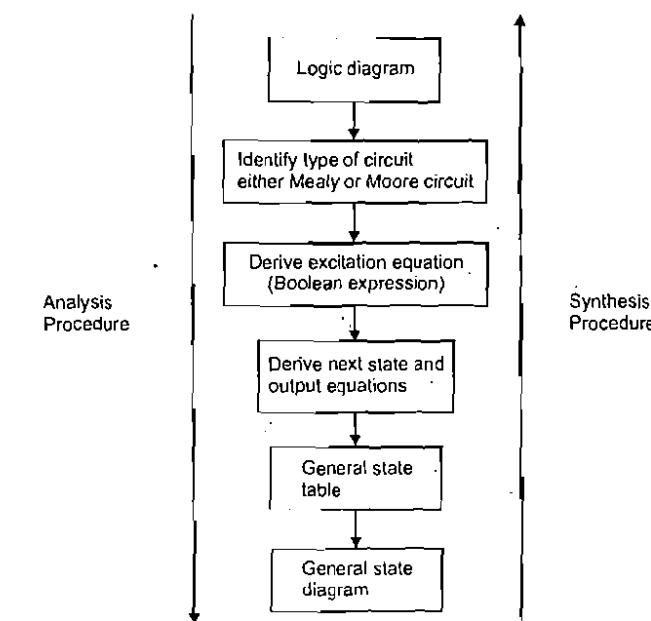


Fig 6.4 Flow chart

6.4 Digital Electronics

The behaviour of sequential circuit can be determined from the inputs, outputs and state of its flip flops. The outputs and next state are both a function of its inputs and the present state. The analysis of a sequential circuit consists of obtaining a state table or state diagram for the time sequence of inputs, outputs and internal states. The analysis of the clocked sequential circuits can be done by following the procedure as shown in Fig.6.4. The reverse process of analysis is known as synthesis of clocked sequential logic circuit.

For the analysis of sequential circuit, we start with the logic diagram. The excitation equation or Boolean expression of each flip-flop is derived from this logic diagram. Then, to obtain the next state equation, we insert the excitation equations into the characteristic equations. The output equations can be derived from the schematic. We can generate the state table using output and next state equations.

6.2.1 Analysis of Example Sequential Logic Circuit

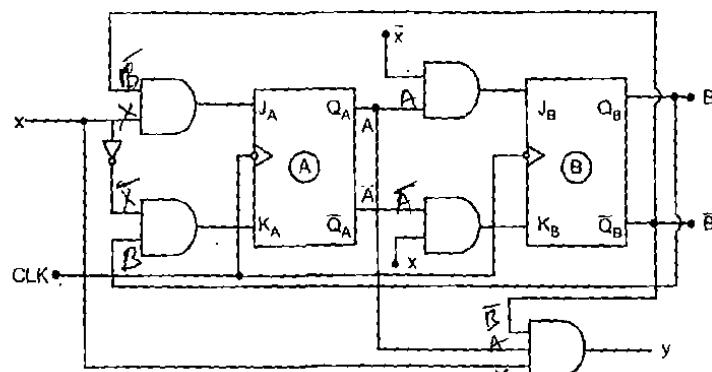


Fig. 6.5 Example of sequential logic circuit

Fig. 6.5 shows a clocked sequential circuit. It has one input variable x , one output variable y and two clocked JK flip flops. The flip flops are labelled as A and B and their outputs are labelled as A and \bar{A} , B and \bar{B} respectively.

Step 1 : Type of circuit

The output(y) of given logic circuit (Fig.6.5) depends on present input and also on present state (Flip flop outputs) of flip flops, so that the given sequential logic circuit is Mealy sequential machine.

Step 2 : Excitation equations

The excitation equations or Boolean expressions of flip flops A and B are obtained. The equations will be in the form of present states A and B and external

input x , since here are two JK flip flops which have output A and B . Therefore the excitation equation (equation formed for flip flop input)

$$\begin{aligned}\text{For Flip flop - } A & \quad J_A = x\bar{B} \\ & \quad K_A = \bar{x}B \\ \text{For flip flop - } B & \quad J_B = \bar{x}A \\ & \quad K_B = x\bar{A}\end{aligned}$$

Step 3 : Next state equations The state equations can be derived directly from the logic diagram. Looking at Fig.6.5 we can see that the signal for J input of the flip flop A is generated by the function $\bar{B}x$ and the signal for input K by the function $\bar{B}x$. Substituting $J = \bar{B}x$ and $K = B\bar{x}$ into a JK flip flop characteristic equation given by

$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

State equation for flip flop A

$$\begin{aligned}A_{n+1} &= (\bar{B}x)\bar{Q}_n + (B\bar{x})Q_n \quad \text{where } Q_n = A \quad J \\ &= \bar{B}x\bar{A} + \bar{B}\bar{x}A \\ &= \bar{A}\bar{B}x + A(\bar{B}\bar{x}) \\ &= \bar{A}\bar{B}x + A(\bar{B} + x) \\ &= \bar{A}\bar{B}x + A\bar{B} + Ax \\ &= \bar{A}\bar{B} + x(A + \bar{A}\bar{B}) \\ &= AB + x(A + \bar{B}) \quad \because (A + \bar{A}\bar{B} = A + \bar{B}) \\ A_{n+1} &= A\bar{B} + Ax + \bar{B}x\end{aligned}$$

State equation for flip flop B

Similarly, we can find the state equation for flip flop B . $J = \bar{A}x$ and $K = \bar{A}x$. Therefore the state equation of flip flop B is given as

$$\begin{aligned}B_{n+1} &= A\bar{x}\bar{B} + (\bar{A}x)B \\ &= A\bar{x}\bar{B} + (A + \bar{x})B \\ &= A\bar{x}\bar{B} + AB + B\bar{x} \\ &= \bar{x}(A\bar{B} + B) + AB \\ &= \bar{x}(A + B) + AB \\ B_{n+1} &= A\bar{x} + B\bar{x} + AB\end{aligned}$$

Output equation

The given sequential circuit has output y . The output equation can be found from the Fig.6.5 which is derived using three input AND gate

$$y = A\bar{B}x$$

Step 4: State table

Table 6.1 is the state table for the given sequential logic circuit. It represents the relationship between input, output and flip flop states. It consists of three columns: present state, next state and output

Present state: It specifies the state of the flip flop before occurrence of a clock pulse.

Next state: It is the state of flip flop after the application of a clock pulse.

Output: This section gives the value of the output variables during the present state. Both next state and output section have two columns representing two possible input conditions $x = 0$ and $x = 1$.

Table 6.1

Present state	Next state		Output	
	AB	AB	y x=0	y x=1
AB	$x=0$	$x=1$	$x=0$	$x=1$
00	00	10	0	0
01	01	00	0	0
10	11	10	0	1
11	01	11	0	1

We can derive the state table as follows

- (i) If present state $AB = 00, x = 0$

When a present state is 00 i.e. $A = 0$ and $B = 0$ and input $x = 0$, the next state is obtained by using next state equation

Next state for flip flop A

$$\begin{aligned} A_{n+1} &= A\bar{B} + Ax + \bar{B}x \\ &= 01 + 0.0 + 1.0 \\ &= 0 \end{aligned}$$

Next state for flip flop B

$$\begin{aligned} B_{n+1} &= A\bar{x} + B\bar{x} + AB \\ &= 0.1 + 0.1 + 0.0 \\ &= 0 \end{aligned}$$

Next state for this case $AB = 00$

- (ii) If present state $AB = 00, x = 1$

Next state for flip-flop A

$$\begin{aligned} A_{n+1} &= A\bar{B} + Ax + \bar{B}x \\ &= 0.1 + 0.1 + 1.1 \\ &= 1 \end{aligned}$$

Next state flip flop B

$$\begin{aligned} B_{n+1} &= A\bar{x} + B\bar{x} + AB \\ &= 0.0 + 0.0 + 0.0 \\ &= 0 \end{aligned}$$

Next state for this case $AB = 10$

Similarly we can obtain next state for all these different cases as shown in the table.

- (iii) Determine the entries in the output section. For this, we have to examine AND gate for all possible present states and input.

$$\begin{aligned} (a) \quad \text{If a present state } AB = 00, x = 0 \\ \text{output } y &= A\bar{B}x \\ &= 0.10 \\ &\boxed{y = 0} \end{aligned}$$

$$\begin{aligned} (b) \quad \text{If a present state } AB = 00, x = 1 \\ \text{output } y &= A\bar{B}x \\ &= 0.11 \\ &\boxed{y = 0} \end{aligned}$$

6.8 Digital Electronics

Thus, the state table of any sequential circuit can be obtained by the same procedure used in the above example. This example contains 2 flip flops and one input, and one output, producing four rows, two columns in the next state and output sections. In general, a sequential circuit with m flip-flops and n -input variables produces 2^m rows and one for each state and 2^n columns, one for each input combination in the next state and output sections of the state table.

Step 5 State diagram

State diagram is a graphical representation of a state table. Fig.6.6 shows the state diagram for sequential circuit. Here each state is represented by a circle, and transition between states is indicated by directed lines connecting the circles. The binary number inside each circle identifies the state represented by the circle. The directed lines are labelled with two binary numbers separated by a symbol '/' (slash). The input value that causes the state transition is labelled first and output value is next.

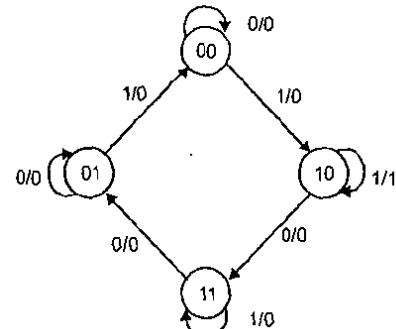


Fig 6.6 State diagram of Fig.6.5

Example 6.1 Derive the state table and state diagram for the sequential circuit shown in Fig.6.7(a).

Solution

Step 1: Type of circuit

The output y of given sequential circuit (Fig.6.7) depends on the present input and also present state (flip flop output) of flip flops, so the given sequential logic circuit is Mealy sequential machine.

Step 2: Excitation equation

For flip flop A

$$J_A = x + B$$

$$K_A = A$$

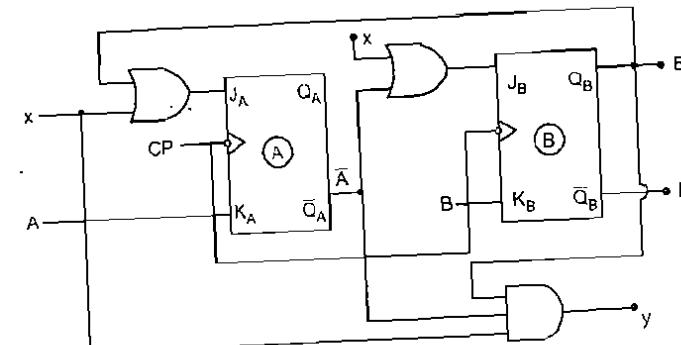


Fig 6.7

For flip flop B

$$J_B = \bar{A} + x$$

$$K_B = B$$

Step 3

We know that characteristic equation of JK flip flop

$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

State equation for flip flop A

$$\begin{aligned}
 A_{n+1} &= (x + B)\bar{Q}_n + A Q_n && (\text{where } Q_n = A \text{ for flip flop A}) \\
 &= (x + B)\bar{A} + \bar{A}A \\
 &= \bar{A}x + \bar{A}B + 0 \\
 A_{n+1} &= \bar{A}x + \bar{A}B
 \end{aligned}$$

State equation for flip flop B

$$\begin{aligned}
 B_{n+1} &= (\bar{A} + x)\bar{Q}_n + \bar{B}Q_n && (\text{where } Q_n = B \text{ for flip flop B}) \\
 &= (\bar{A} + x)\bar{B} + \bar{B}B \\
 B_{n+1} &= \bar{A}\bar{B} + \bar{B}x
 \end{aligned}$$

Output equation

$$y = \bar{A}Bx$$

Step 4 : State table

Present state	Next state		Output	
	AB	\bar{AB}	y	
AB	$x = 0$	$x = 1$	$x = 0$	$x = 1$
00	00	11	0	0
01	10	10	0	1
10	01	00	0	0
11	00	00	0	0

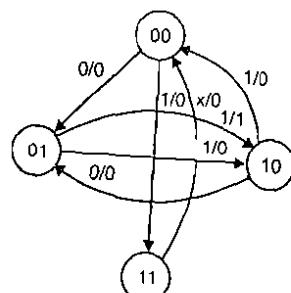
Step 5 : State diagram

Fig 6.8

Example 6.2 Derive the state table and state diagram for the sequential circuit shown in Fig 6.9

 Solution**Step 1 : Type of circuit**

The output of a given circuit (See Fig.6.9) depends on present input and also on present states, so the given sequential logic circuit is Mealy machines

Step 2: Excitation Equation

For flip flop A

$$D_A = Ax + Bx$$

For Flip flop B

$$D_B = \bar{A}x$$

For Output

$$y = AB + \bar{x}$$

Step 3:

We know that characteristic equation of D flip flop (next state depends on input D)

$$A_{n+1} = D_A$$

$$A_{n+1} = Ax + Bx$$

$$B_{n+1} = D_B$$

$$B_{n+1} = \bar{A}x$$

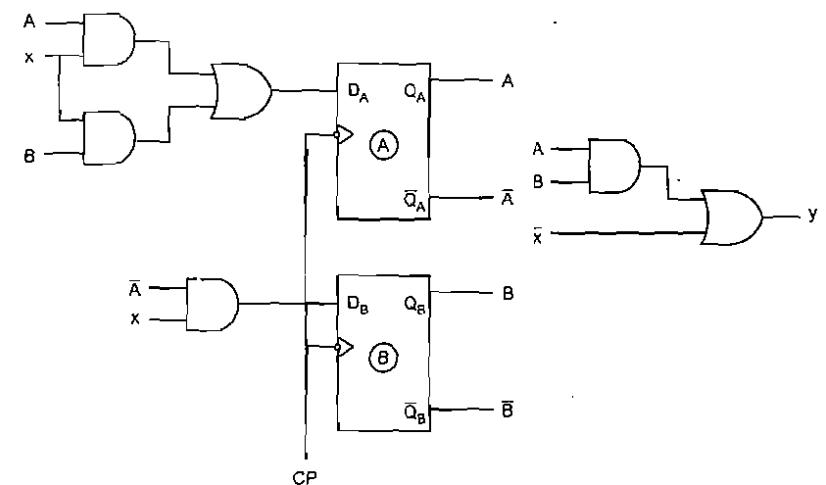
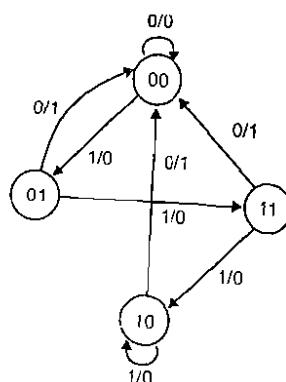


Fig 6.9

Step 4: State table

The state table contains four rows and three columns. The next state and output have two sub columns.

Present state	Next state		Output	
	AB	\bar{AB}	y	
AB	$x = 0$	$x = 1$	$x = 0$	$x = 1$
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

Step 5: State Diagram

Example 6.3 Derive the state table and state diagram for sequential circuit shown in Fig.6.10

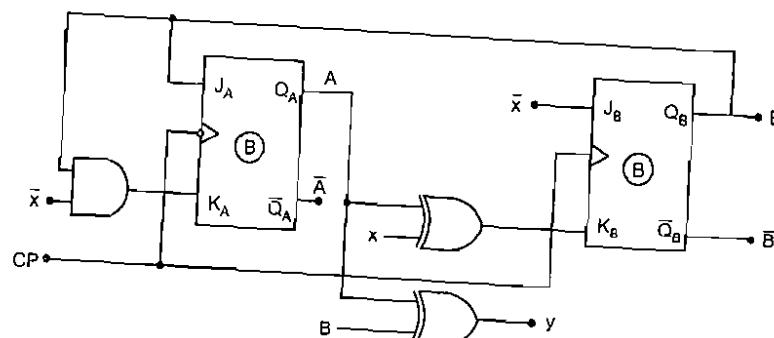


Fig 6.10

Solution**Step 1 : Type of circuit**

The output y of the sequential circuit depends on present state only, so the given logic circuit is the Moore type circuit.

Step 2: Excitation equations

For flip flop A

$$J_A = B$$

$$K_A = B\bar{x}$$

For flip flop B

$$J_B = \bar{x}$$

$$K_B = A \oplus x$$

$$y = A \oplus B$$

Step 3 :

We know that characteristics equation of JK flip flop

$$A_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

$$\begin{aligned} \text{State equation for flip flop } A : A_{n+1} &= B\bar{A} + (\bar{B}\bar{x})A \\ &= B\bar{A} + A(\bar{B} + x) \\ &= B\bar{A} + A\bar{B} + x \end{aligned} \quad (\because Q_n = A)$$

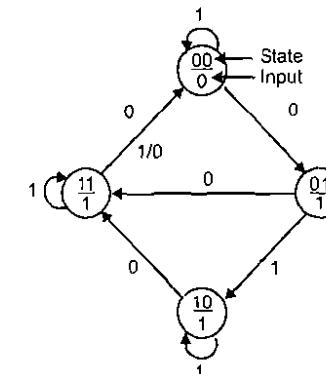
$$A_{n+1} = (A \oplus B) + x$$

$$\begin{aligned} \text{State equation for flip flop } B : B_{n+1} &= \bar{x}\bar{B} + (\bar{A} \oplus \bar{x})B \\ &= \bar{x}\bar{B} + (Ax + \bar{A}\bar{x})B \end{aligned}$$

$$B_{n+1} = \bar{x}\bar{B} + Ax\bar{B} + \bar{A}\bar{x}B$$

Step 4: State table

Present state	Next state		Output
	$x = 0$	$x = 1$	
AB	$\bar{A}\bar{B}$	$\bar{A}\bar{B}$	-
00	01	00	0
01	11	10	1
10	11	10	1
11	00	11	0

State diagram

Note: The state diagram for Moore machine is different from Mealy machine. Here each circle is coded with state binary number/output.

6.3 State Reduction

Any logic design process must consider the problem of minimizing the cost of the final circuit. One way to reduce the cost is by reducing the number of flip flops, i.e. by reducing the number of states. The state reduction technique basically avoids the introduction of redundant equivalent states. The reduction of redundant states reduces the number of flip flops and logic gates required, thus reducing the cost of the final circuit. Two states are said to be redundant or equivalent, if every possible set of inputs generate exactly the same outputs and the same next states. When two states are equivalent, one of them can be removed without altering input-output relationship. Let us consider the state diagram shown in Fig. 6.11. The states are denoted by letter symbols instead of their binary values because in state reduction technique internal states are also important, but input output sequences are more important. The procedure contains two steps.

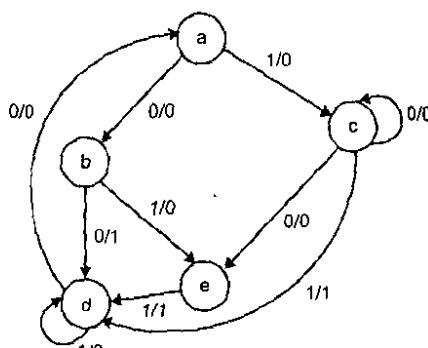


Fig. 6.11

Step 1: Finding the state table for the given state diagram

First the given state diagram is converted into a state table. Fig. 6.11 shows the example of state diagram.

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

Both are equivalent states because of state c and e having same next state and same output

Step 2: Finding equivalent states

The two present states go to the same next state and have the same output for both the input combinations. We can easily find this from the state table. States

c and e are equivalent. This is because both c and e states go to states c and d for outputs of 0 and 1 for $x = 0, x = 1$ respectively. Therefore, the state e can be removed and replaced by c. The final reduced table and state diagram are given in the table 6.2 and Fig. 6.12. The second row have e state for the input $x = 1$, it is replaced by c because the states c and e are equivalent.

Table 6.2 Reduced state table

Present state	Next state		Output	
	AB	AB	$y = 0$	$y = 1$
AB	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	b	c	0	0
b	d	c	1	0
c	c	d	0	1
d	a	d	0	0

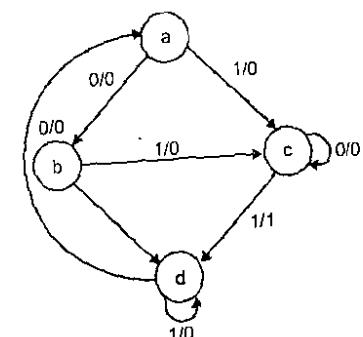


Fig. 6.12 Reduced state diagram

Example 6.4 Obtain the reduced state table and reduced state diagram for a sequential circuit whose state diagram is shown in Fig. 6.13.

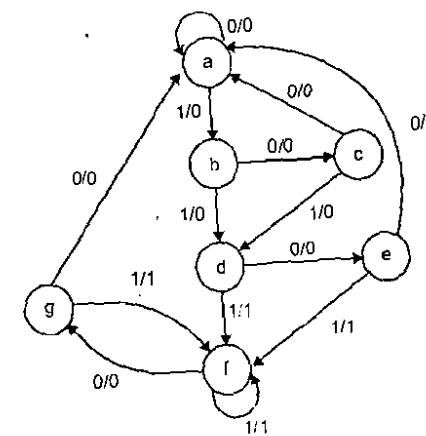


Fig. 6.13

Solution

The given diagram has seven states, one input and one output. As per the step 1, the given state diagram is converted to a state table.

State table

Table 6.4(a)

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	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Both are equivalent states because of state e and g having same next state and same output

From the above state table, it is clear that states e and g are equivalent. So the state g is replaced by state e. The reduced state table is shown in Ex.6.4

Reduced state Table

Table 6.4(b)

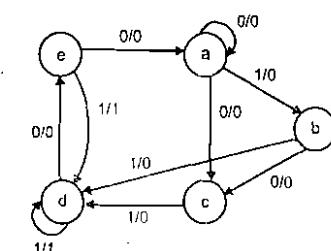
Present state	Next state		Output	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

From the above reduced table, states d and f are equivalent, hence 'f' can be replaced by d and it can be removed. Then finally the reduced state table is shown in Table 6.4(c)

Final reduced table

The state diagram of the reduced state Table is shown in Fig.6.4(b).

Present state	Next state		Output	
	AB	AB	$x=0$	$x=1$
a	$x=0$	$x=1$	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1



6.4 State Assignment

In sequential circuits we know that the behaviour of the circuit is defined in terms of its inputs, present state, next state and outputs. To generate the desired next state at particular present state and inputs, it is necessary to have specific flip flop inputs. These flip flop inputs are described by a set of Boolean functions called flip flop input functions. To determine the flip flop input functions, it is necessary to represent states in the state diagram using binary values instead of alphabets. This procedure is known as state assignment. The following rules are used in state assignment.

Rule 1. States having the same next states for a given input condition should have assignments which can be grouped into logically adjacent cells in a K-map. (Fig.6.15)

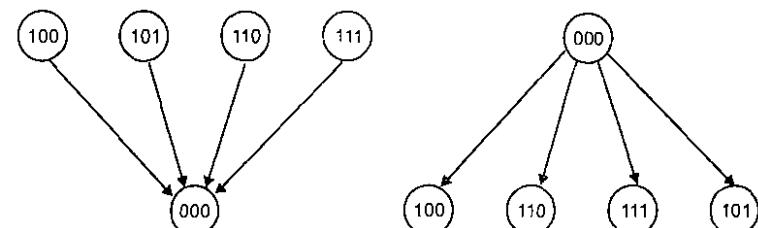


Fig.6.15

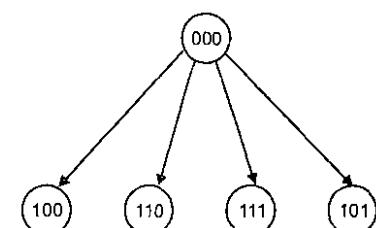


Fig.6.16

Rule 2. States having different next states should have assignment which can be grouped into logically adjacent cells in K-map. (Fig.6.16)

Example 6.5 Design a sequential circuit using D flip flop for a state diagram given below. Use state assignment rules for assigning states and compare the required combinational circuit with random state

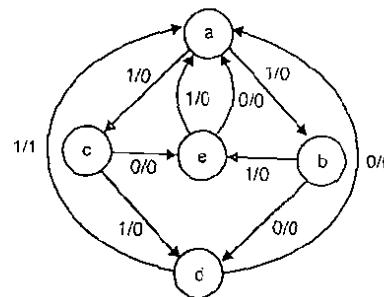


Fig 6.17

Solution

The states are a, b, c, d and e . Each state is randomly assigned.

$a = 000, b = 001, c = 010, d = 011, e = 100$. The remaining combinations are considered as don't care conditions.

Excitation table

Present state			Input	Next State			Output
A	B	C	X	A_{n+1}	B_{n+1}	C_{n+1}	Z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	1	0	1	1	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	1
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

Present state		Input	Next State		Output
1	0	1	X	X	X
1	0	1	X	X	X
1	1	0	X	X	X
1	1	0	X	X	X
1	1	1	X	X	X
1	1	1	X	X	X

K-map simplification

The D flip flop input is equal to next state and the flip flop expression is obtained directly.

Expression for flip flop input D_A

AB	Cx	00	01	11	10
00	0	1	3	1	2
01	4	5	7	6	
11	12	13	15	14	
10	8	9	11	10	x

$$D_A = \bar{B}C\bar{C}x + \bar{B}Cx$$

Expression for flip flop input D_B

AB	Cx	00	01	11	10
00	0	1	3	2	1
01	4	5	7	6	
11	12	13	15	14	x
10	8	9	11	x	x

$$D_B = \bar{A}C\bar{x} + \bar{B}Cx$$

Expression for flip flop input D_C

AB	Cx	00	01	11	10
00	0	1	3	2	1
01	4	5	7	6	
11	12	13	15	14	x
10	8	9	11	x	x

$$D_C = \bar{A}\bar{B}\bar{x} + B\bar{C}x$$

Expression for flip flop input D_D

AB	Cx	00	01	11	10
00	0	1	3	2	
01	4	5	7	6	
11	12	13	15	14	x
10	8	9	11	x	x

$$Z = BCx + A\bar{x}$$

The random assignment requires

7 three input AND gates

1 two input AND gates

4 two input OR gates

Total 12 gates with 34 inputs and 3 flip flops are required to construct the sequential logic circuit. Now we apply state assignment rules, then follow the above steps.

From Rule 1, The states e and d must be adjacent

From Rule 2, states b and c must be adjacent. We form the adjacent cells in the 3 variable K-map

A\B	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Excitation table

Present state	Input	Next State	Output				
A	B	C	X	A_{n+1}	B_{n+1}	C_{n+1}	Z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	1	0
0	0	1	0	1	0	1	0
0	0	1	1	1	1	1	0
0	1	0	0	X	X	X	X
0	1	0	1	X	X	X	X
0	1	1	0	1	1	1	0
0	1	1	1	1	0	1	0
1	0	0	0	X	X	X	X
1	0	0	1	X	X	X	X
1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

K-Map simplification

Expression for flip flop input D_A

AB\Cx	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$D_A = AC$$

Expression for flip flop input D_B

AB\Cx	00	01	11	10
00	0	1	3	2
01	4	5	1	6
11	12	13	15	14
10	8	9	11	10

$$D_B = \bar{A}Bx + A\bar{B}x$$

Expression for flip flop input D_C

AB\Cx	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$D_C = A$$

Expression for flip flop input D_D

AB\Cx	00	01	11	10
00	0	1	3	2
01	4	5	1	6
11	12	13	x	14
10	8	x	9	10

$$D_D = ABx + A\bar{B}x$$

Under the state assignment rules, we require

4 three input AND gates

1 two input AND gate

2 two input OR gates

A total of 7 gates with 18 inputs and 3 flip flops are required to construct the sequential logic circuit based on the state assignment rules.

6.5 Design Procedure

The following steps are followed to design the clocked sequential logic circuit.

- Obtain the state table from the given circuit information such as a state diagram, a timing diagram or description.
- The number of states may be reduced by state reduction technique.
- Assign binary values to each state in the state table.
- Determine the number of flip flops required and assign a letter symbol to each flip flop.
- Choose the flip flop type to be used according to the application.
- Derive the excitation table from the reduced state table.
- Derive the expression for flip flop inputs and outputs using k-map simplification (The present state and inputs are considered for k-map simplification) and draw logic circuit using flip flops and gates

6.6 Synthesis of Clocked Sequential Logic Circuits

Synthesis means that, it is the reverse process of analysing a sequential logic circuit. In this synthesis, we get a logic circuit from the information of state diagram, word description etc. The detailed steps are given in the example. Now we will see the detailed description of each step.

A state diagram is obtained from the word description, timing diagram or other pertinent information. From this state diagram, we can form a state table.

The reduction of number of states and binary value assignment to each state gives the reduction in combinational circuit requirement. The number of flip flops required to design any sequential logic circuit depends on the number of states.

Example 6.6 A sequential circuit has one input and one output and its state diagram is shown in Fig.6.18(a). Design the sequential circuit using D flip flop

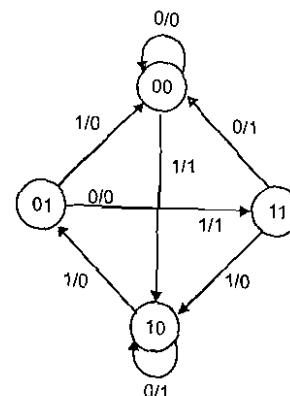


Fig 6.18(a)

Solution

The given state diagram consists of four states. It has one input (x) and one output (y). The state table for the given state diagram is shown in Table 6.6(a). It is clear that there are no equivalent states. Therefore, there is no reduction in the state diagram. As the state diagram contains 4-states, it requires 2 flip-flops which are named as A and B .

Table 6.6 (a)

Present state	Next state	Output	
		$x = 0$	$x = 1$
AB	AB	$x = 0$	$x = 1$
00	00	10	0
01	11	00	0
10	10	01	1
11	00	10	1

Design using D-flip flop

For the design of circuit using D flip flop (or any flip flop), we need the excitation table. Table Ex 6.6(b) shows the excitation table of D flip flop from which we can develop excitation table for the required circuit as shown in table 6.6(c).

Table 6.6 (b) Excitation table for D -flip flop

Present state	Next state	Flip flop input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table

Present state	Input	Next state	Flip flop input	Output			
A	B	x	A	B	D_A	D_B	y
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	1	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

The flip-flop input function and the circuit output function are obtained by using K-map simplification.

Input equation (or) function for flip flop $A(D_A)$

Bx	00	01	11	10
A	0	1	3	2
0	4	5	7	6
1	1	1	1	1

$$\begin{aligned}D_A &= \bar{A}\bar{B}x + \bar{A}B\bar{x} + A\bar{B}\bar{x} + ABx \\&= \bar{A}(\bar{B}x + B\bar{x}) + A(\bar{B}\bar{x} + Bx)\end{aligned}$$

Let us consider $z = \bar{B}x + B\bar{x}$, then $\bar{B}\bar{x} + Bx = z$. Simplify the above equation

$$\begin{aligned}D_A &= \bar{A}z + Az \\&= A\bar{z} + z\end{aligned}$$

Substitute $z = \bar{B}x + B\bar{x} = B \oplus x$ in the above equation

$$D_A = A \oplus B \oplus x$$

Input equation for flip flop B (D_B)

	$\bar{B}x$	00	01	11	10
0		0	1	3	2
1		4	5	7	6

$$D_B = \bar{A}\bar{B}x + A\bar{B}x$$

Output function y

	$\bar{B}x$	00	01	11	10
0		0	1	3	2
1		4	1	5	6

$$y = A\bar{x} + \bar{A}\bar{B}x$$

The input equation for flip flop and output equation are simulated as follows

$$DA = A \oplus B \oplus x$$

$$DB = \bar{A}\bar{B}x + A\bar{B}x$$

$$y = A\bar{x} + \bar{A}\bar{B}x$$

A sequential circuit using D flip flop is obtained by using the above equations as shown in fig 6.18(b).

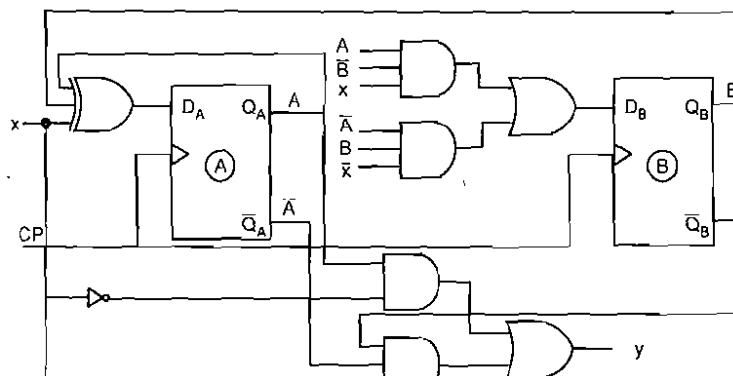


Fig 6.18(b)

6.7 Sequence Generator

A sequential circuit which generates a prescribed sequence of bits, synchronous with the clock, is referred to as a sequence generator. We can construct sequence generators by two ways

1. Sequence generators using counters
2. Sequence generators using shift registers

6.7.1 Sequence Generator using Counters

Fig.6.19 shows the block diagram of a sequence generator using counters. It contains two stages

1. counter, and

2. next state decoder.

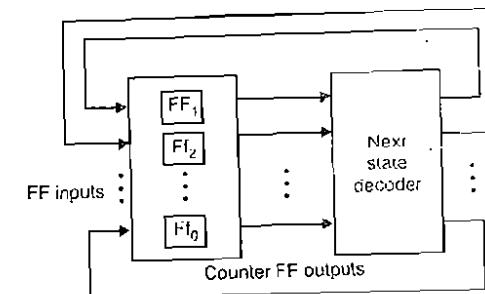


Fig 6.19

Design Procedure

Step 1 : Determine the number of flip-flops required

The number of flip-flops required to generate a particular sequence can be determined as follows.

- (a) Find the number of 1's in the sequence.
- (b) Find the number of 0's in the sequence.
- (c) Take the maximum value from both. If 'n' is the required number of flip-flops, choose minimum value of 'n' to satisfy the following condition.

$$\max(0's, 1's) \leq 2^{n-1}$$

Step 2 : State assignment

Once the number of flip-flops is decided, we have to assign unique states corresponding to each bit in the given sequence such that the flip-flop representing least significant bit generates the given sequence (the output of the flip-flop which represents the least significant bit is used to represent the given sequence)

Step 3 : Draw the state diagram from the above state assignment and obtain the excitation table from the state diagram.

Step 4 : Find the Boolean expression for each flip-flop input by using k-map and draw the logic diagram for this Boolean expression

Example 6.7 Find the number of flip-flops required to generate the sequence 10110110.

Solution

In the given sequence, the number of 0's are 3 and number of 1's are 5.

$$\begin{aligned} \max(3, 5) &\leq 2^{n-1} \\ 5 &\leq 2^{n-1} \\ n = 4 \end{aligned}$$

■

Example 6.8 Design a sequence generator using JK flip-flop to generate the sequence 1101011.

Solution

Step 1: Number of flip-flops required

Number of 0's in the sequence = 2

Number of 1's in the sequence = 5

$$\text{Hence } \max(2, 5) \leq 2^{n-1}$$

$$5 \leq 2^{n-1}$$

$$n = 4$$

We need four flip-flops named as A, B, C and D. The desired sequence is generated by the D flip-flops output

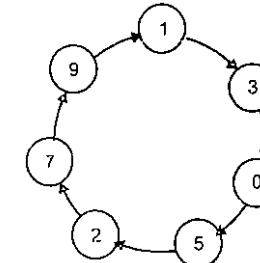
Step 2: State assignment

Decimal equivalent	A	B	C	D
1	0	0	0	1
3	0	0	1	1
0	0	0	0	0
5	0	1	0	1
2	0	0	1	0
7	0	1	1	1
9	1	0	0	1

Given sequence,
first enter
this column

Assign binary value based
on non-repeated states

Step 3: State diagram



Excitation table

Present state								Next state				Flip-flop inputs					
Q_A	Q_B	Q_C	Q_D	Q_A	Q_B	Q_C	Q_D	J_A	K_A	J_B	K_B	J_C	K_C	J_D	K_D		
0	0	0	0	0	1	0	1	0	X	1	X	X	1	1	X		
0	0	0	1	0	0	1	1	0	X	0	X	1	X	X	0		
0	0	1	0	0	1	1	1	0	X	1	X	X	0	1	X		
0	0	1	1	0	0	0	0	0	0	X	0	X	X	1	X	1	
0	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
0	1	0	1	0	0	1	0	0	X	X	1	1	X	X	1	X	
0	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
0	1	1	1	1	0	0	1	1	X	X	1	X	1	X	0		
1	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	0	0	1	0	0	0	1	1	X	1	0	X	0	X	X	0	
1	0	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	0	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	1	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	1	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	1	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	
1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	

Note : The unused states 4, 6, 8, 10, 11, 12, 13, 14, and 15 are considered as X

K-map simplification

$Q_A Q_B$		For J_A			
Q_A	Q_B	00	01	11	10
0	0	x			
0	1	x	1	x	
1	1	x	x	x	x
1	0	x	x	x	x

$Q_A Q_B$		For K_A			
Q_A	Q_B	00	01	11	10
0	0	x	x	x	x
0	1	x	x	x	x
1	1	x	x	x	x
1	0	x	x	x	x

$Q_A Q_B$		For J_B			
Q_A	Q_B	00	01	11	10
0	0	1			
0	1	x	x	x	x
1	1	x	x	x	x
1	0	x	x	x	x

		For K_B				For J_C				For K_C							
		Q _A Q _B	00	01	11	10	Q _A Q _B	00	01	11	10	Q _A Q _B	00	01	11	10	
		00	x	x	x	x	00	x	1	x	x	00	x	x	1	x	
		01	x	x	x	x	01	x	1	x	x	01	x	x	1	x	
		11	x	x	x	x	11	x	x	x	x	11	x	x	x	x	
		10	x	x	x	x	10	x		x	x	10	x	x	x	x	

		For J_D				For K_D					
		Q _A Q _B	00	01	11	10	Q _A Q _B	00	01	11	10
		00	1	x	x	1	00	1	x	1	x
		01	x	x	x	x	01	x	1	x	x
		11	x	x	x	x	11	x	x	x	x
		10	x	x	x	x	10	x	x	x	x

Logic diagram

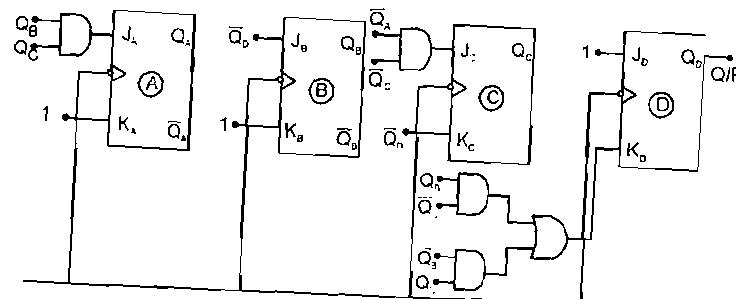
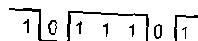


Fig 6.20

Example 6.9 Design a pulse train generator for the waveform shown below.



Solution

Step 1 : The pulse is repeated for every 4-bit sequence 0111. Therefore the required number of flip flop is determined as follows.

- (i) number of 0's = 1
- (ii) number of 1's ≈ 3

$$\text{Hence } \max(1, 3) \leq 2^{n-1}$$

$$3 \leq 2^{n-1}$$

$$n = 3$$

We need three flip-flop named as A, B and C . The desired sequence is generated by the C flip-flop.

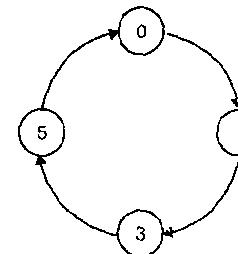
Step 2 : State assignment

Decimal equivalent	A	B	C
0	0	0	0
1	0	0	1
3	0	1	1
5	1	0	1

Given sequence

Note : The unused states are 2, 4, 6 and 7. Consider the don't care (X) for these states in the K-map simplification.

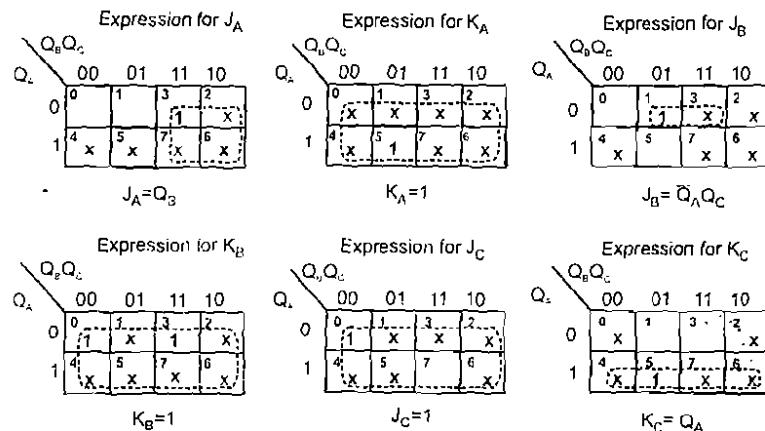
Step 3 : State diagram



Excitation table

Present state			Next state			Flip-flop inputs					
Q _A	Q _B	Q _C	Q _A	Q _B	Q _C	J ₃	K ₁	J ₂	K _B	J _C	K _C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	1	1	0	1	1	X	X	1	X	0
1	0	1	0	0	0	X	1	0	X	X	1

Note : Unused states 2, 4, 6 and 7 are considered as X

K-map simplification

This minimal expression form is K-map simplification

$$\begin{array}{lll} J_A = Q_B & J_B = \bar{Q}_A Q_C & J_C = 1 \\ K_A = 1 & K_B = 1 & K_C = Q_A \end{array}$$

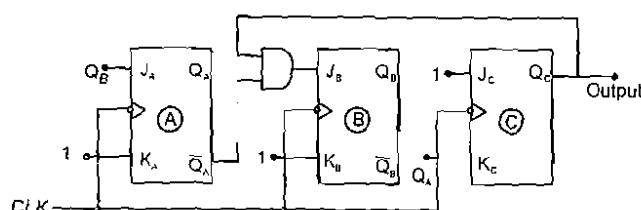
Logic diagram

Fig 6.21

6.7.2 Sequence Generator using Shift Registers

This is another method for designing sequence generator. In this method shift registers with next state decoder logic are used. Fig.6.22 shows the block diagram of sequence generator using shift registers.

From this Fig.6.22, we see that the output of next state decoder is a function of Q_A, Q_B, \dots, Q_n . The next state decoder is a logic circuit which decodes the output of shift register and generates input to get desired sequence from flip flop A (least significant bit).

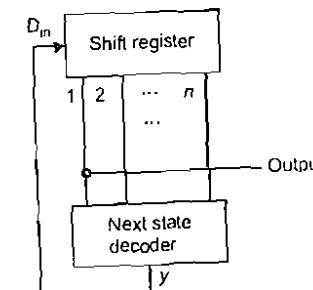


Fig 6.22 Block diagram of a sequence generator using shift registers

Example 6.10 Design a sequence generator to generate the sequence 1101011 by shift register method.

In this approach, the minimum number of flip-flops n , required to generate a sequence of length N is given by

$$N \leq 2^{n-1}$$

In this example $N = 7$ and therefore, the minimum value of n , which may generate the sequence is

$$7 \leq 2^n - 1$$

$$n = 3$$

With the three flip-flops, the sequence generation is shown in Table 6.7. The state diagram is shown in Fig.6.23

Table 6.7 State assignment

Flip-flop outputs			D_{in}	States
Q_A	Q_B	Q_C		
1	0	0	1	4
1	0	1	0	5
0	0	0	1	0
1	1	0	0	6
0	1	0	1	2
1	1	1	1	7

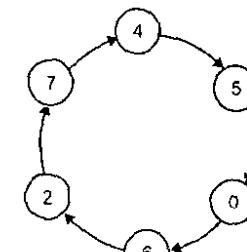


Fig 6.23

K-map simplification for D_{in}

		$Q_A Q_C$	00	01	11	10	
		Q_A	0	0	1	3	2
Q_B	0	0	1	x	x	1	
	1	4	5	7	1	6	

$$D_{in} = \bar{Q}_A + \bar{Q}_B \bar{Q}_C + Q_B Q_C \\ = \bar{Q}_A + Q_B \oplus Q_C$$

The logic diagram is shown in Fig.6.24. The initial state is 100. So the input for D_A , D_B and D_C are 100 respectively.

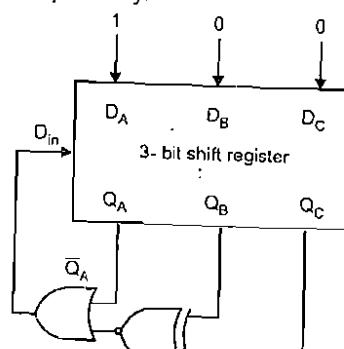


Fig 6.24

6.8 Sequence Detector

A sequence detector is a sequential logic circuit that can be used to detect whether a given sequence of bits has been received or not. We can draw the state diagram when we know the sequence and then follow the steps to design sequential logic circuit to obtain the sequence detector sequential logic circuit.

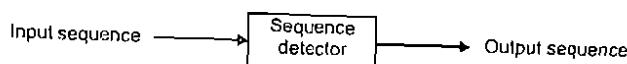


Fig 6.25

Generally sequence detector produces an output = 1, whenever it detects the desired input sequence and '0' for other cases. There are two types of detectors:

1. A detector which detects overlapping input sequence, and
2. A detector which detects non-overlapping input sequence.

Example 6.11 Design a sequence detector which detects the sequence 100011.

Solution

In general, the number of states in the state diagram is equal to the number of bits in the sequence. Once the number of states is known, one has to draw the directed lines with inputs and outputs as weighed between the two states. Let us start to draw state diagram, assuming initial state is A.

State A: In this state, the detector may receive either an input 0 or 1. Based on these inputs, a sequence detector is either in same state or move on to the next state as shown in Fig.6.26.

When input is 1, we have detected first bit in the sequence, hence we have to go to next state (B) to detect the next bit in the sequence

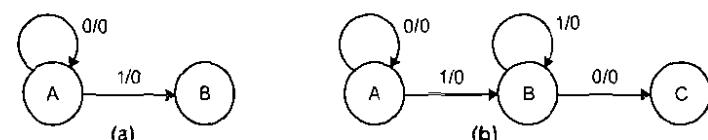


Fig 6.26

When input is 0, we have to remain in state A, because bit '0' is not the first bit in the sequence

State B: When input is 0, we have detected the second bit in the sequence. Hence we have to go to Next state (C) to detect the next bit in the sequence [See Fig.6.26(b)]

When input is 1, we have to remain in the state B, because 1 which we have detected may start the sequence output which is still zero for both cases

State C: When input is 0, we have detected the third bit in the sequence, hence we have to go next state (D) to detect the next bit in the sequence

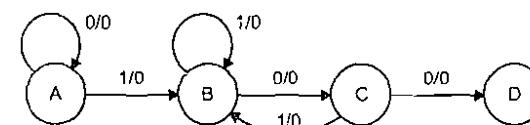


Fig 6.26(c)

When input is 1, we have to go to state B, because 1 which we have detected may not be in the sequence to be detected but it may be the start/bit of the sequence, hence we can move to state B. The output is still zero (See Fig.6.26(c))

State D to State F: As explained for state A, state B and state C, if the desired bit is detected, we have to go for the next state otherwise we have to go to the previous state from where we can continue the desired sequence. When complete sequence is detected, we have to make output 1 and go to the initial state. The complete state diagram is shown in Fig.6.26(d).

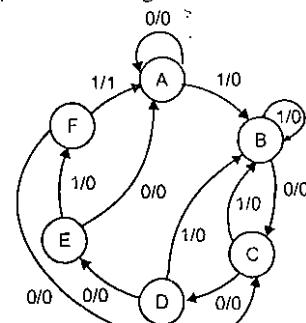


Fig 6.26(d)

State assignment

Assume that for state assignment, we need 3 flip flops to construct the sequence detector circuit. The sequence detector has a total of 6 states. Two flip flops are enough for less than or equal to 4 states. Hence $3(2^3 = 8)$ flip flops are required to construct the sequence detector. We choose JK flip flop and the flip flops are labeled as A, B and C, assuming state assignments as A = 000, B = 001, C = 010, D = 011, E = 100 and F = 101.

Excitation table

We can easily write the excitation table from the state diagram.

Input	Present state			Next state			Output	Flip flop inputs						
	X	A	B	C	A_{n+1}	B_{n+1}	C_{n+1}	J_A	K_A	J_B	K_B	J_C	K_C	
0	0	0	0	0	0	0	0	0	X	0	X	0	X	
0	0	0	1	0	1	0	0	0	0	X	1	X	X	1
0	0	1	0	0	1	1	0	0	X	X	0	1	X	
0	0	1	1	1	0	0	0	0	1	X	X	1	X	1
0	1	0	0	0	0	0	0	0	X	1	0	X	0	X
0	1	0	1	1	1	0	0	0	X	1	1	X	X	1
1	0	0	0	0	0	1	0	0	X	0	X	1	X	
1	0	0	1	0	0	0	1	0	0	X	0	X	X	0
1	0	1	0	0	0	1	0	0	0	X	X	1	X	0
1	0	1	1	0	0	1	0	0	0	X	X	1	X	0
1	1	0	0	1	0	1	0	0	X	0	0	X	1	X
1	1	0	1	0	0	0	1	1	X	1	0	X	X	1

K-map simplification

Expression for flip flop input J_A

AX	BC			
	00	01	11	10
00	0	1	3	2
01	4	x	7	x
11	x	13	15	14
10	6	9	11	10

$$J_A = \bar{X} BC$$

Expression for flip flop input J_B

AX	BC			
	00	01	11	10
00	0	1	3	2
01	4	5	7	x
11	x	13	15	14
10	8	9	11	x

$$J_B = \bar{X} C$$

Expression for flip flop input J_C

AX	BC			
	00	01	11	10
00	0	1	3	2
01	4	5	7	x
11	x	13	15	x
10	8	9	11	x

$$J_C = X + B$$

Expression for flip flop input K_A

AB	Cx			
	00	01	11	10
00	x	x	x	x
01	1	1	x	x
11		1	x	
10	x	x	x	x

$$K_A = X + C$$

Expression for flip flop input K_B

AB	Cx			
	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	x	x	x	x
10	x	x	x	x

$$K_B = \bar{X} + C$$

Expression for flip flop input K_C

XA	BC			
	00	01	11	10
00	x	1	3	2
01	x	5	7	x
11	x	13	15	x
10	x	9	11	x

$$K_C = \bar{X} + B$$

Expression for Output Y

AX	BC			
	00	01	11	10
00	0	1	3	2
01	4	5	7	x
11	x	13	15	x
10	6	9	11	10

$$Y = XAC$$

6.36 Digital Electronics

Logic diagram for sequence detector

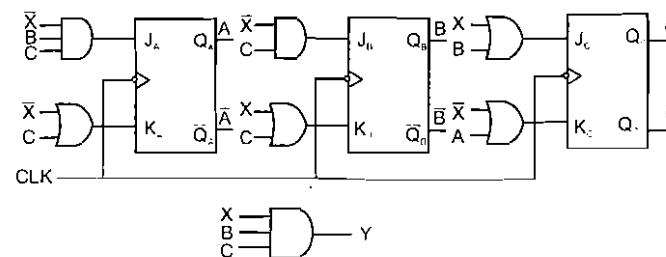


Fig 6.27 Logic diagram for sequence detector

Example 6.12 Design a sequence detector to detect the sequence 101 from 10101.

Solution

We have only 3 states because we have to detect the sequence 101 from the given number 10101. This circuit is allowed repetition.

Initially, we assume that the circuit is in its reset state, state a . With a 1 coming in as first bit in the valid sequence, it will go from state a to state b with an output as 0 because we have not yet detected all the bits in the sequence. When input is 0, we detect second valid bit in the sequence so it will go from state b to state c , otherwise state b remains same. When the input is 1, we detect third bit in the sequence, which will go from state c to state b with the output as 1 because we are yet to detect all bits in the sequence. If the input is 0, it will go from state c to state a with output 0.

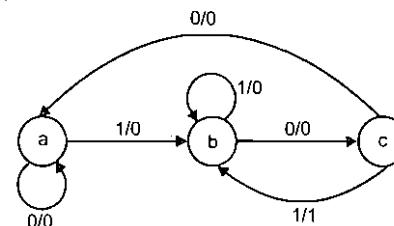


Fig 6.28 State diagram

The binary values are assigned to state a , b and c . Only two flip flops are enough ($2^2 = 4$) to design the sequence detector sequential logic circuit, by using T flip flops.

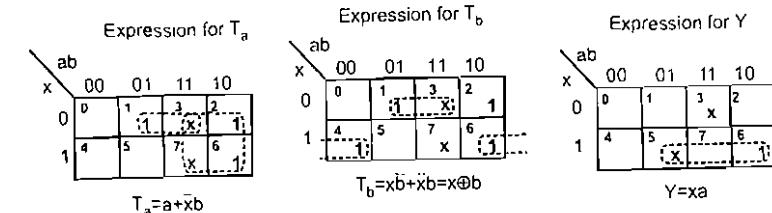
Assume the state assignment

$$a = 00, \quad b = 01, \quad c = 10$$

Consider the input is X and the output is Y .

Input	Present state		Next state		Flip flop Inputs		Output
X	a	b	a_{n+1}	b_{n+1}	T_a	T_b	Y
0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	0
0	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0
1	0	1	0	1	0	0	0
1	1	0	0	1	1	1	1
1	1	1	X	X	X	X	X

K-map simplification



Logic diagram for sequence detector

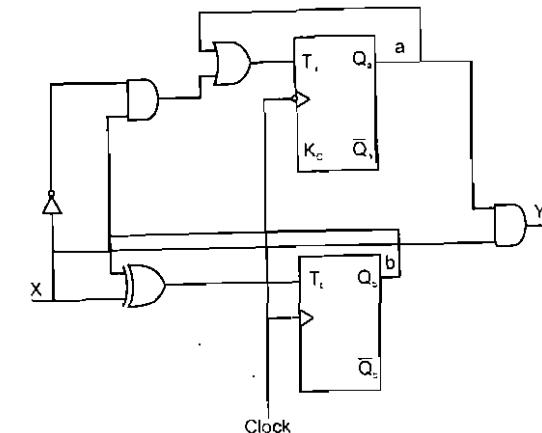


Fig 6.29 Logic diagram for sequence detector

Example 6.13 A sequential circuit with two D flip flops A and B and input X and output Y is specified by the following next state and output equations

$$A(t+I) = AX + BX$$

$$B(t+I) = A'X$$

$$Y = (A+B)X'$$

(a) Draw the logic diagram of the circuit

(b) Derive the state table

(c) Derive the state diagram

Solution

(a)

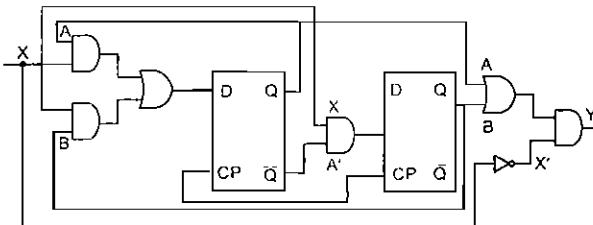


Fig 6.30

(b) **State Table**

Present state	Input	Next state	Flip flop Inputs		Output		
A	B	X	A^+	B^+	D_A	D_B	
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	1	1	1	1	1	0
1	0	1	1	0	1	0	0
1	1	1	1	0	1	0	0

(c) **State diagram**

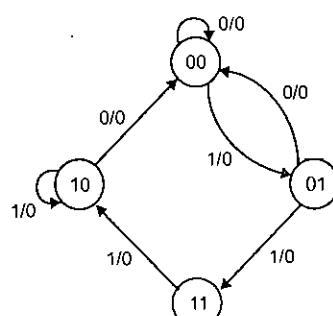


Fig 6.31

Example 6.14 Reduce the number of states in the following state table and tabulate the reduced state table.

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

Solution

Two states are said to be equivalent if their next states and outputs are equal.

Compare state 'a' with other states. The next state of 'a' and 'f' are equal for the inputs 0 and 1 but their outputs are not equal. Hence $a \neq f$. Similarly comparing other states it is found that $b \equiv e$ as their next state and outputs are equal and $d = h$. Hence the state table can be reduced as shown. The row e is removed from the table and if any previous row containing e , it is replaced by b and the rows are compared. If any state has the same next state and outputs replace one row, continue this process to obtain reduced state table.

[The strike out lines are kept to illustrate reduction process]

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

After the first comparison, we find that $a \equiv c$ hence the reduced state table is as shown below.

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

Example 6.15 Design a sequential circuit with two D flip flops, A and B, and one input x . When $x = 0$, the state of the circuit remains the same. When $x = 1$, the circuit passes through the state transitions from 00 to 01 to 11 to 10 and back to 00 and repeats.

Solution

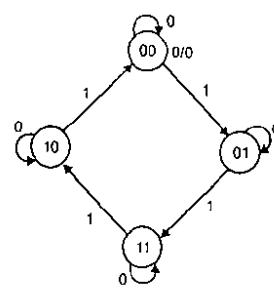


Fig 6.32

State table

Present state	Input	Next state		Output	
		A^+	B^+	D_A	D_B
0 0	0	0	0	0	0
0 0	1	0	1	0	1
0 1	0	0	1	0	1
0 1	1	1	1	1	1
1 0	0	1	0	1	0
1 0	1	0	0	0	0
1 1	0	1	1	1	1
1 1	1	1	0	1	0

K-map simplification

Expression for flip flop input D_A

A	Bx			
	00	01	11	10
0	0	1	0	2
1	4	5	7	6

$$D_A = \bar{A}\bar{x} + Bx$$

Expression for flip flop input D_B

A	Bx			
	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$$D_B = \bar{A}\bar{x} + Bx$$

Circuit diagram

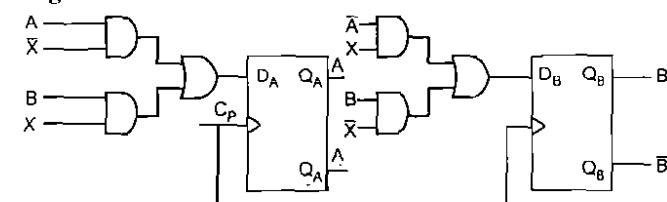


Fig 6.33

Example 6.16 Design a sequential circuit that three flip flops A, B, C, one input x and one output y . The state diagram is shown in the fig 6.34. The circuit is to be designed by treating the unused states as don't care conditions. Use JK flip flops in the design.

Solution

State diagram

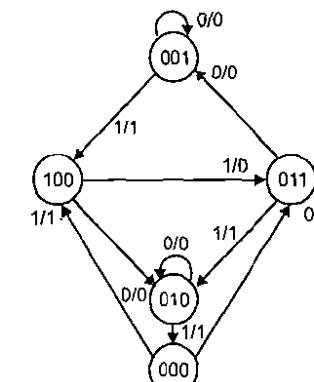


Fig 6.34

State table for the circuit

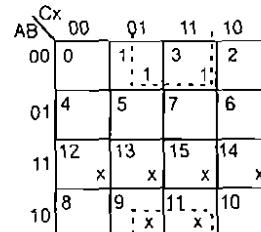
Present State	Next state		Output y	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A 0 0	$A^+ 0$	$A^+ 1$	$B^+ 0$	$B^+ 1$
A 0 1	0	1	0	1
A 1 0	0	1	0	1
A 1 1	0	1	0	1
B 0 0	$B^+ 0$	$B^+ 1$	$C^+ 0$	$C^+ 1$
B 0 1	0	1	0	1
B 1 0	0	1	0	1
B 1 1	0	1	0	1
C 0 0	$C^+ 0$	$C^+ 1$	$A^+ 0$	$A^+ 1$
C 0 1	0	1	0	1
C 1 0	0	1	0	1
C 1 1	0	1	0	1

Excitation table for the circuit

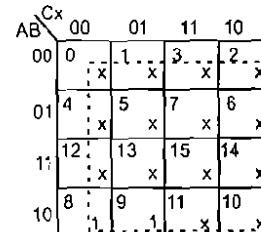
Present state	Input	Next state	Flip flop Inputs						Output	
			A	B	C	J _A	K _A	J _B	K _B	
0 0 0	0	0 1 1	0	X	1	X	1	X	0	0
0 0 0	1	1 0 0	1	X	0	X	0	X	1	1
0 0 1	0	0 0 1	0	X	0	X	X	0	0	0
0 0 1	1	1 0 0	1	X	0	X	X	1	1	1
0 1 0	0	0 1 0	0	X	X	0	0	X	0	0
0 1 0	1	0 0 0	0	X	X	1	0	X	1	1
0 1 1	0	0 0 1	0	X	X	1	X	0	0	0
0 1 1	1	1 0 1	0	X	X	1	X	1	0	1
1 0 0	0	0 1 0	0	1	0	X	1	X	0	0
1 0 0	1	0 1 1	0	1	X	1	X	1	X	0

K-map simplification

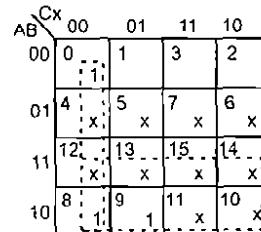
Note : minterms 10, 11, 12, 13, 14 and 15 are don't care



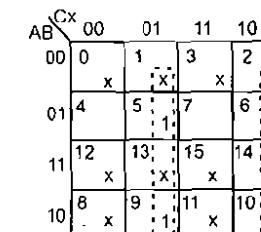
$$J_A = \bar{B}x$$



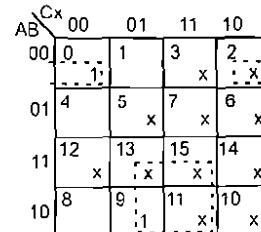
$$K_A = 1 \text{ (since all the cells are grouped)}$$



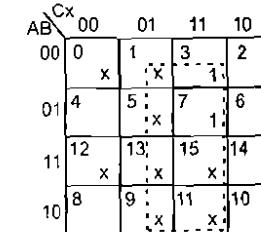
$$J_B = \bar{C} \bar{x} + A$$



$$K_B = \bar{C}x + Cx = C \oplus x$$



$$J_C = A \bar{B} \bar{x} + Ax$$



$$K_C = X$$

AB	Cx			
	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$$Y = Ax$$

Circuit Diagram

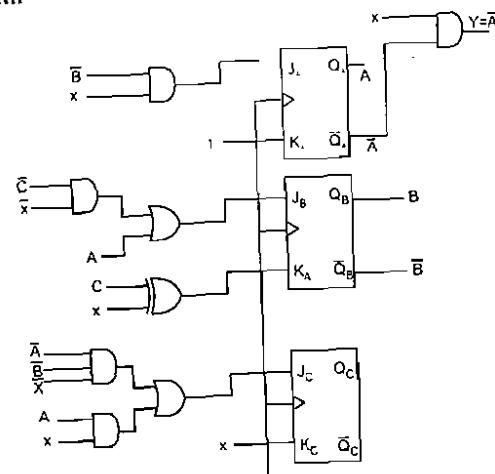


Fig 6.35

Example 6.17 Derive the state table and state diagram for the sequential circuit shown Fig 6.36.

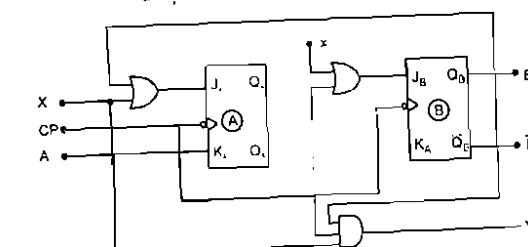


Fig 6.36

Solution

The given circuit has two flip flops, A and B, one input X and one input Y. So, it produces a state table with 4 rows as shown below

Present State		Next state		Output	
A	B	$\bar{A}\bar{B}$	AB	y	y
0	0	01	11	0	1
0	1	10	10	0	1
1	0	01	00	0	0
1	1	00	00	0	0

The state diagram for the sequential circuit is shown in Fig 6.37.
State diagram

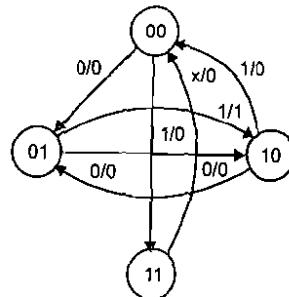


Fig 6.37

Example 6.18 A sequential circuit with two D flip flops, A and B; two inputs x and y; and one output z, is specified by the following next state and output equations:

$$A(t+1) = \bar{y}y + xA$$

$$B(t+1) = \bar{x}B + xA$$

$$z = B$$

- Draw the logic diagram of the circuit
- List the state table for sequential circuit
- Draw the corresponding state diagram

Solution

(a) **Logic diagram of the sequential logic circuit**

A sequential logic circuit is drawn using given equations as shown in Fig.6.38.

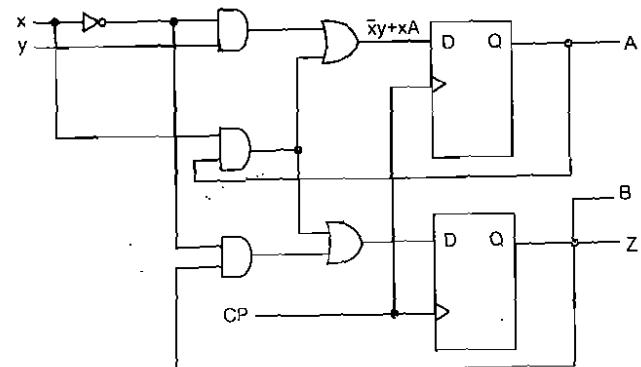


Fig 6.38

(b) **State table**

Present State		Inputs		Next state		Output
A	B	x	y	A	B	z
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	1
0	1	0	1	1	1	1
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	0	0
1	0	0	1	1	0	0
1	0	1	0	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

(c) **State diagram**

The state diagram is drawn as shown in Fig.6.39 with the help of the above table

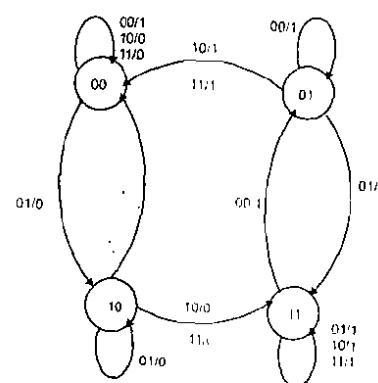


Fig 6.39

Example 6.19 Design a sequential circuit with two flip-flops A and B, and one input x . When $x = 0$, the state of the circuit remains same. When $x = 1$, the circuit goes through the state transitions from 00 to 01 to 10 to 11 to 10 back to 00, and repeats.

Solution

State table

Present state		Input	Next state		FF inputs	
A	B	x	A	B	D_A	D_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	1	0
1	0	1	0	1	0	1
1	1	0	1	0	1	0
1	1	1	1	1	1	1

Note : the FF inputs of the D FF is same as the next state

K-map simplification

Expression for flip flop input D_A

Bx

	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$$D_A = Ax + Bx$$

Expression for flip flop input D_B

Bx

	00	01	11	10
0	0	1	3	2
1	4	5	7	6

$$D_B = \bar{A}x + B\bar{x}$$

Logic Diagram

By using the above Boolean Expression, the diagram is constructed as shown in Fig.6.40.

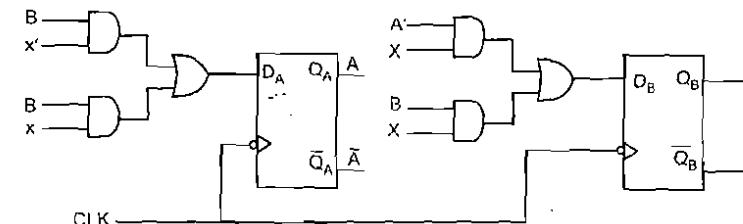


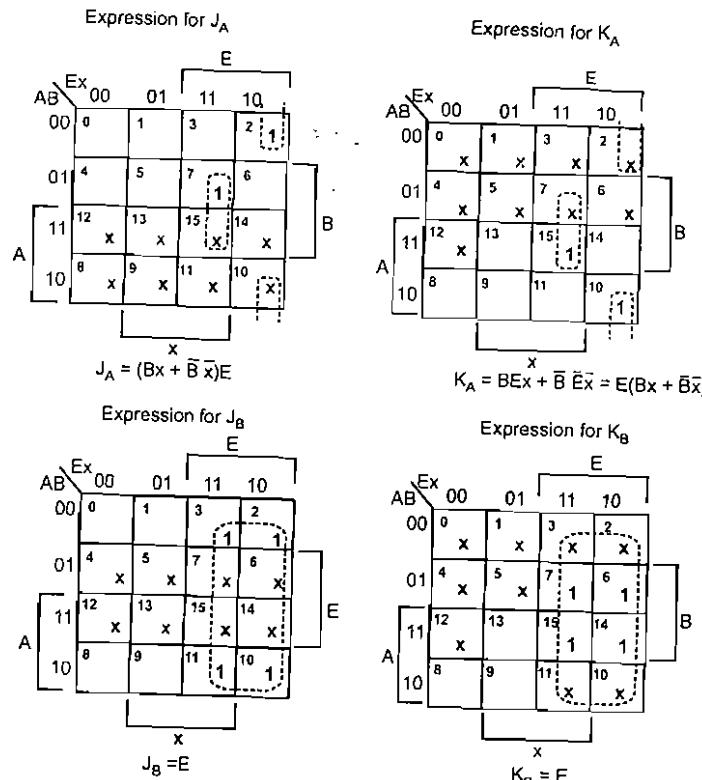
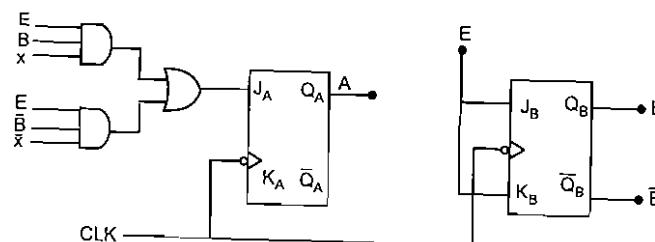
Fig 6.40

Example 6.20 Design a sequential circuit with two JK flip-flops A and B and two inputs E and x . If $E = 0$, the circuit remains in the same state regardless of the value of x . When $E = 1$ and $x = 1$, the circuit goes through the state transitions from 00 to 01 to 10 to 11 back to 00, and repeats. When $E = 1$ and $x = 0$, the circuit goes through the state transitions from 00 to 11 to 10 to 01 back to 00, and repeats.

Solution

The excitation table is derived directly from the given specification.

Present state		Input	Next state		Flip flop Inputs				
A	B	E	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	0	1	0	0	0	X	0	X
0	0	1	0	1	1	1	X	1	X
0	0	1	1	0	1	0	X	1	X
0	1	0	0	1	0	1	0	X	X
0	1	0	1	0	1	0	X	X	0
0	1	1	0	0	0	0	0	X	X
0	1	1	1	1	0	1	X	X	1
1	0	0	0	1	0	0	X	0	0
1	0	0	1	1	0	0	X	0	0
1	0	1	0	0	1	1	X	1	X
1	0	1	1	1	1	1	X	0	1
1	1	0	0	1	0	0	X	0	X
1	1	0	1	1	1	1	X	0	X
1	1	1	0	1	0	0	X	0	X
1	1	1	1	0	0	0	X	1	X

K-map simplification**Logic diagram****Short Questions and Answer**

1. *Mention the steps involved in the analysis of a sequential circuit.*

The analysis of a sequential circuit consists of:

- Obtaining a table or a diagram for the time sequence of inputs, outputs and internal states.
- Writing Boolean expressions, which include the necessary time sequence, either directly or indirectly.

2. *What is a state-equation?*

A state-equation (also called transition equation) specifies the next state as a function of the present state and inputs.

3. *What is a state-table?*

A state-table contains data such as the time sequence of inputs, outputs and flip-flop states. The table consists of four sections labelled present state, input, next state and output state.

4. *What is a state diagram?*

A state diagram is a graphical representation of the information available in a state table. In the diagram, a state is represented by a circle and the transitions between states are indicated by directed lines connecting the circles.

5. *What are the informations obtained from a state diagram?*

The informations obtained are as follows :

The state of the flip-flops are identified by the binary number inside the circle.

A directed line connecting a circle which indicates that change of state occurs.

6. *What are input and output equations?*

The part of the circuit that generates the inputs to flipflops is described algebraically by a set of Boolean functions called input equations.

7. *How are the next-state values of a sequential circuit that uses JK or T type flip-flops derived?*

The next-state values of a sequential circuit that uses JK or T type flip-flops are derived using the following procedure:

- Determine the flip flop input equations in terms of the present state and input variables.

- (b) List the binary values of each input equation.
- (c) Use the corresponding flip flop characteristic table to determine the next state value in the state table.

8. How are the next-state values obtained from the characteristic equation?

The next state values can be obtained by evaluating the state equations from the characteristic equation. This is done by the following procedure:

- (a) Determine the flip-flop input equations in terms of the present state and input variables.
- (b) Substitute the input equations into the flip-flop characteristic equation to obtain the state equations.
- (c) Use the corresponding state equations to determine the next state value in the state table.

9. What are the two models of sequential circuits?

The two models of sequential circuits are:

- (a) Mealy model, and
- (b) Moore model.

10. Compare Mealy and Moore machine.

Mealy machine	Moore machine
i. The output is a function of both the present state and input.	i. The output is a function of the present state only.
ii. The outputs may change if the inputs change during the clock cycle.	ii. The outputs are synchronized with the clock, because they depend only on flip-flop outputs that are synchronized with the clock.

11. What is the use of initial statement?

The initial statement is used for generating input signals to simulate a design. In simulating a sequential circuit, it is necessary to generate a clock source for triggering the flip-flops.

12. How can the always statement be controlled?

The always statement could be controlled by delays that wait for a certain time or by certain conditions to become true or by events to occur.

13. What is a sensitivity list?

A sensitivity list specifies the events that must occur to initiate the execution of the procedural statements in the always block. Statements within the block execute sequentially and the execution suspends after the last statement has been executed.

14. What are the two kinds of procedural assignments?

- (a) The two kinds of procedural assignments are:

Blocking assignments

Blocking assignment statements are executed sequentially in the order, they are listed in a sequential block.

Blocking assignments use the symbol ($=$) as the assignment operator. Example for procedural blocking assignments:

$$B = A$$

$$C = B + 1$$

(b) Non-blocking assignments:

It evaluates the expressions on the right hand side, but do not make the assignment to the left hand side, until all expressions are evaluated.

It uses the (\leftarrow) as the operator.

Example for non-blocking assignments: $B \leftarrow A$

$$C \leftarrow B + 1$$

15. How can we determine the behavior of a clocked sequential circuit?

The behavior of a clocked sequential circuit could be determined from the inputs, outputs and the state of its flip-flops.

16. What are clocked sequential circuits?

Synchronous sequential circuits that use clock pulses in the inputs of storage elements are called clocked sequential circuits.

17. How can we describe the structure of a sequential circuit?

The sequential circuit is made up of flip-flops and gates and so its structure can be described by a combination of data flow and behavioral statements.

Short Answer Questions

1. Differentiate synchronous and asynchronous sequential logic circuits.
2. What are the classification of sequential machine?
3. Define Mealy and Moore Machines.
4. Define state assignment.
5. When are two states said to be equivalent states?

Review Questions

1. What are the steps for the design of an asynchronous sequential circuit?
2. What is the significance of state assignment?
3. List the different techniques used for state assignment.
4. Write a short note on
 - (a) Shared row state assignment.
 - (b) One hot state assignment.

Exercises

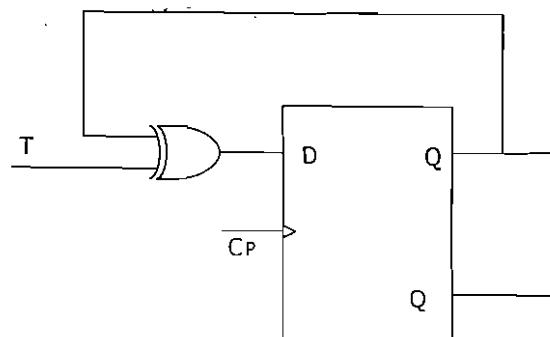
1. A sequential circuit with two D flip flops, A and B, two inputs x and y, and one input z is specified by the following next state and output equations.

$$A(t+1) = x'y + xA$$

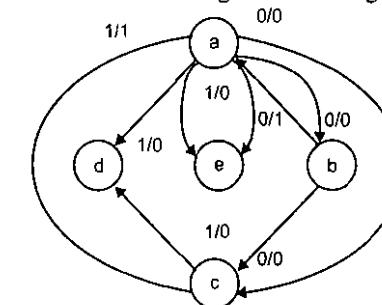
$$B(t+1) = x'B + xA$$

$$Z = B$$

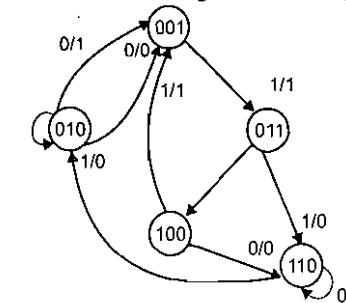
- (a) Draw the logic diagram of the circuit
- (b) Derive the state table.



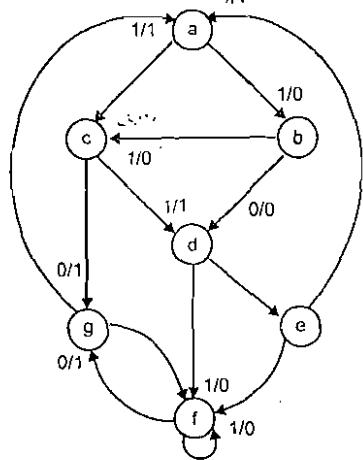
2. A sequential circuit has one flip flop Q, two inputs x and y; and one output S. It consists of a full-adder circuit connected to a D flip flop as shown in following fig.1. Derive the state table and state diagram of the sequential circuit.
3. Analyze the circuit in the following Fig.2 and prove that it is equivalent to a T flip flop.
4. A sequential circuit has two JK flip flops, one input x and one output y. Following is the logic diagram of the circuit. Derive the state table and state diagram 7. Design a sequential circuit for the given state diagram by using JK flip flop of the circuit.
5. Design a sequential circuit with two JK flip flops, A and B and two inputs E and X. If E = 0, the circuit remains in the same state regardless of the value of x. When E = 1 and X = 1, the circuit goes through the state transitions from 00 to 01 to 10 to 11 back to 00 and repeats. When E = 1 and X = 0, the circuit goes through the state transitions from 00 to 11 to 10 to 01 back to 00 and repeats.
6. Design a sequential circuit for the given state diagram.



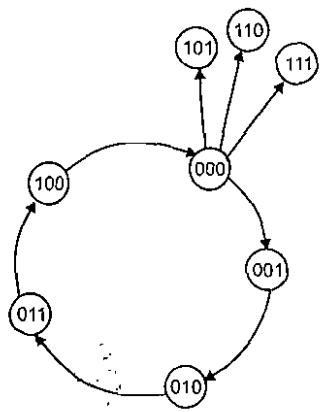
7. Design a sequential circuit for the given state diagram in Fig. P5.



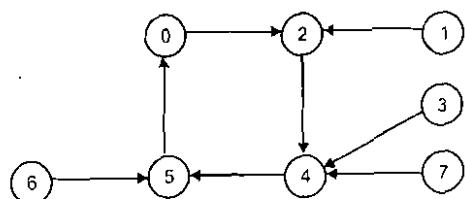
8. Design a clocked sequential circuit for the given state diagram in Fig. P6.



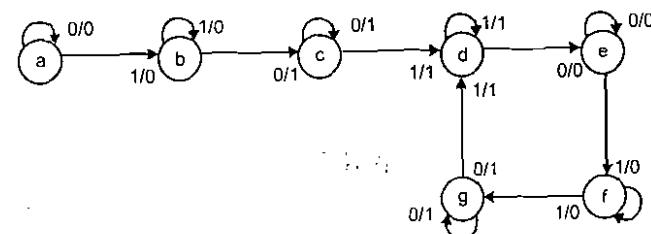
9. Design a sequential circuit for the following state diagram.



10. Design a sequential circuit for the given state diagram using JK flip flop.



11. Obtain the state table for the given state diagram and reduce it.



MODULE-V:

Sequential Logic Circuits - II

Sequential Circuit Design

Steps in the design process for sequential circuits

State Diagrams and State Tables

Examples

Steps in Design of a Sequential Circuit

1. **Specification** – A description of the sequential circuit. Should include a detailing of the inputs, the outputs, and the operation. Possibly assumes that you have knowledge of digital system basics.
2. **Formulation**: Generate a state diagram and/or a state table from the statement of the problem.
3. **State Assignment**: From a state table assign binary codes to the states.
4. **Flip-flop Input Equation Generation**: Select the type of flip-flop for the circuit and generate the needed input for the required state transitions
5. **Output Equation Generation**: Derive output logic equations for generation of the output from the inputs and current state.
6. **Optimization**: Optimize the input and output equations. Today, CAD systems are typically used for this in real systems.
7. **Technology Mapping**: Generate a logic diagram of the circuit using ANDs, ORs, Inverters, and F/Fs.
8. **Verification**: Use a HDL to verify the design.

Sequential machines are typically classified as either a Mealy machine or a Moore machine implementation.

Moore machine: The outputs of the circuit depend only upon the current state of the circuit.

Mealy machine: The outputs of the circuit depend upon both the current state of the circuit and the inputs.

An example to go through the steps

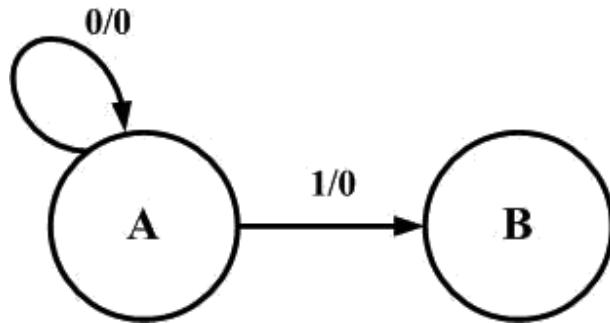
The specification: The circuit will have one input, X, and one output, Z. The output Z will be 0 except when the input sequence 1101 are the last 4 inputs received on X. In that case it will be a 1

Generation of a state diagram

Create states and meaning for them.

State A – the last input was a 0 and previous inputs unknown. Can also be the reset state. **State B** – the last input was a 1 and the previous input was a 0. The start of a new sequence possibly.

Capture this in a state diagram



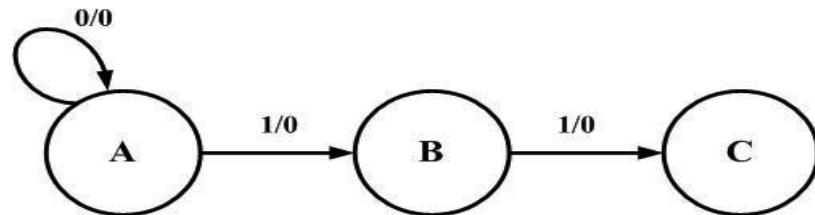
- Capture this in a state diagram

Circles represent the states

Lines and arcs represent the transition between states.

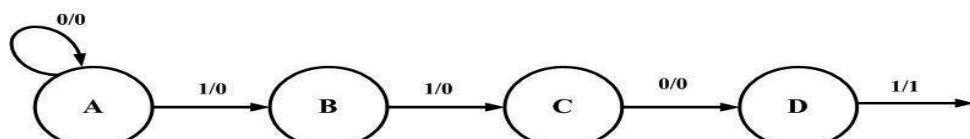
The notation Input/output on the line or arc specifies the input that causes this transition and the output for this change of state.

Add a state C – Have detected the input sequence 11 which is the start of the sequence



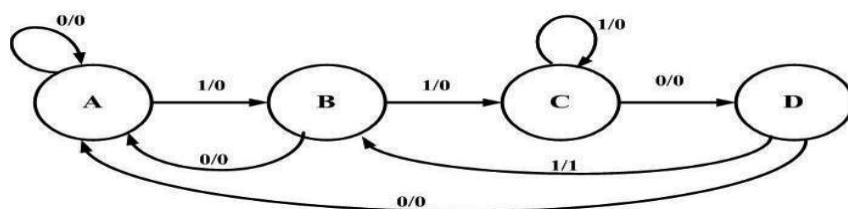
- Add a state D

State D – have detected the 3rd input in the start of a sequence, a 0, now having 110. From State D, if the next input is a 1 the sequence has been detected and a 1 is output.



- The previous diagram was incomplete.

- In each state the next input could be a 0 or a 1. This must be included

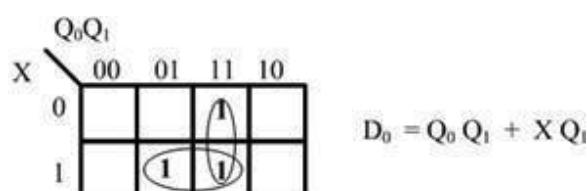


The state table**This can be done directly from the state diagram**

Prresent State	Next State		Output	
	X=0	X=1	X=0	X=1
A	A	B	0	0
B	A	C	0	0
C	D	C	0	0
D	A	B	0	1

Now need to do a state assignment**Will select a gray encoding****For this state A will be encoded 00, state B 01, state C 11 and state D 10**

Prpresent State	Next State		Output	
	X=0	X=1	X=0	X=1
00	00	01	0	0
01	00	11	0	0
11	10	11	0	0
10	00	01	0	1

Flip-flop input equations**Generate the equations for the flip-flop inputs Generate the D_0 equation****Generate the D_1 equation**

X	Q ₀ Q ₁	00	01	11	10	
0						D ₁ = X
1		1	1	1	1	

The output equation

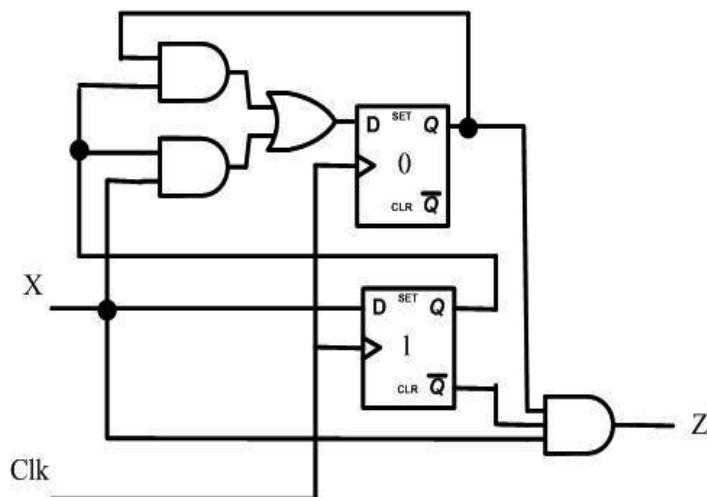
The next step is to generate the equation for the output Z and what is needed to generate it.

Create a K-map from the truth table.

X	Q ₀ Q ₁	00	01	11	10	
0						Z = X Q ₀ \bar{Q}_1
1					1	

Now map to a circuit

The circuit has 2 D type F/Fs



Shift registers:

In digital circuits, a shift register is a cascade of flip-flops sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it, shifting in the data present at its input and shifting out the last bit in the array, at each transition of the clock input. More generally, a shift register may be multidimensional, such that its "data in" and stage outputs are themselves bit arrays: this is implemented simply by running several shift registers of the same bit-length in parallel.

Shift registers can have both parallel and serial inputs and outputs. These are often configured as serial-in, parallel-out (SIPO) or as parallel-in, serial-out (PISO). There are also types that have both serial and parallel input and types with serial and parallel output. There are also bi-directional shift registers which allow shifting in both directions: L→R or R→L. The serial input and last output of a shift register can also be connected to create a circular shift register

Shift registers are a type of logic circuits closely related to counters. They are basically for the storage and transfer of digital data.

Buffer register:

The buffer register is the simple set of registers. It is simply stores the binary word. The buffer may be controlled buffer. Most of the buffer registers used D Flip-flops.

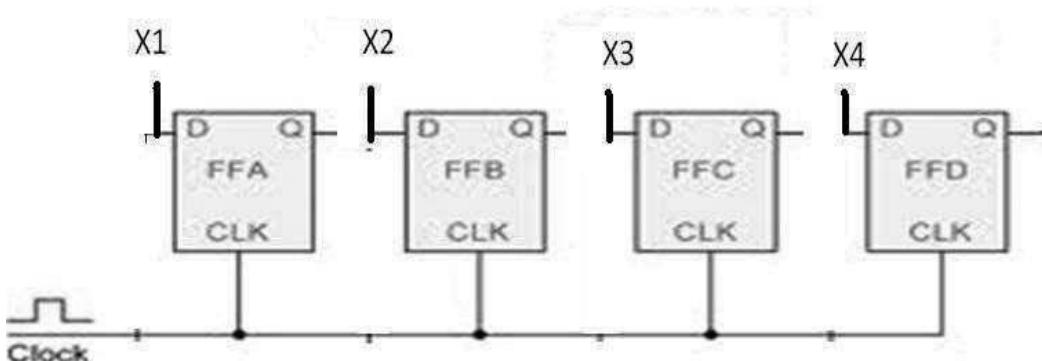


Figure: logic diagram of 4-bit buffer register

The figure shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals. On the application of clock pulse, the output word becomes the same as the word applied at the terminals. i.e., the input word is loaded into the register by the application of clock pulse.

When the positive clock edge arrives, the stored word becomes:

$$Q_4 Q_3 Q_2 Q_1 = X_4 X_3 X_2 X_1$$

$$Q = X$$

Controlled buffer register:

If goes LOW, all the FFs are RESET and the output becomes, Q=0000.

When is HIGH, the register is ready for action. LOAD is the control input.

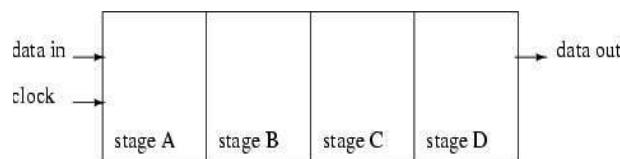
When LOAD is HIGH, the data bits X can reach the D inputs of FF's.

$$Q_4 Q_3 Q_2 Q_1 = X_4 X_3 X_2 X_1$$

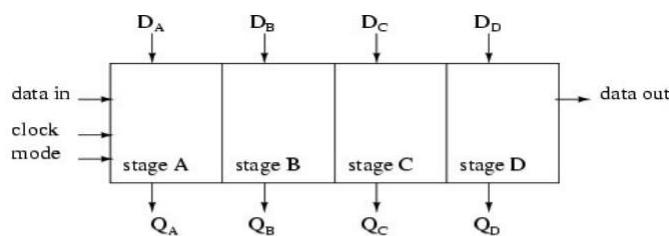
$$Q = X$$

When load is low, the X bits cannot reach the FF's.

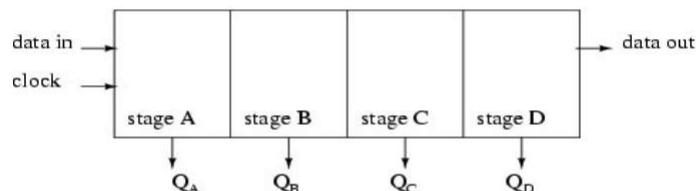
Data transmission in shift registers:



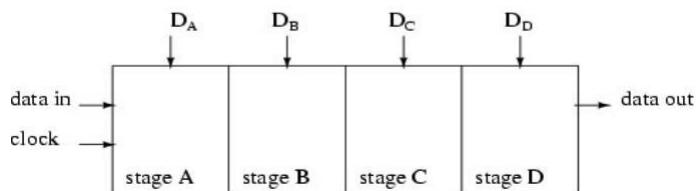
Serial-in, serial-out shift register with 4-stages



Parallel-in, parallel-out shift register with 4-stages



Serial-in, parallel-out shift register with 4-stages



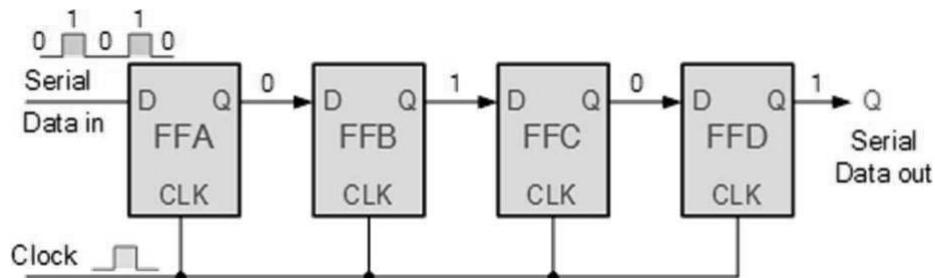
Parallel-in, serial-out shift register with 4-stages

A number of ff's connected together such that data may be shifted into and shifted out of them is called shift register. data may be shifted into or out of the register in serial form or in parallel form. There are four basic types of shift registers.

1. Serial in, serial out, shift right, shift registers
2. Serial in, serial out, shift left, shift registers
3. Parallel in, serial out shift registers
4. Parallel in, parallel out shift registers

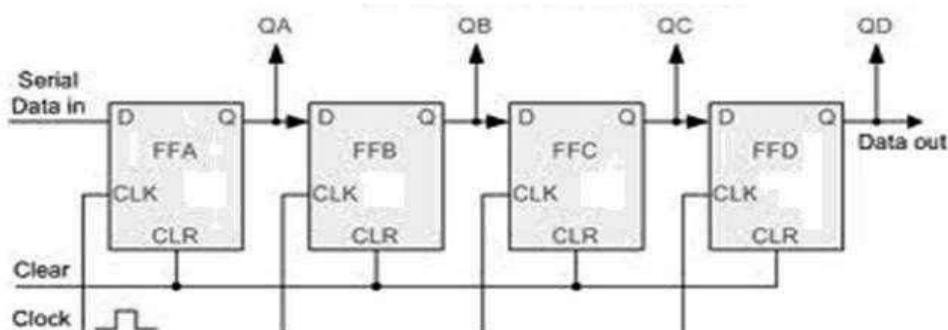
Serial IN, serial OUT, shift right, shift left register:

The logic diagram of 4-bit serial in serial out, right shift register with four stages. The register can store four bits of data. Serial data is applied at the input D of the first FF. the Q output of the first FF is connected to the D input of another FF. the data is outputted from the Q terminal of the last FF.



When serial data is transferred into a register, each new bit is clocked into the first FF at the positive going edge of each clock pulse. The bit that was previously stored by the first FF is transferred to the second FF. the bit that was stored by the Second FF is transferred to the third FF.

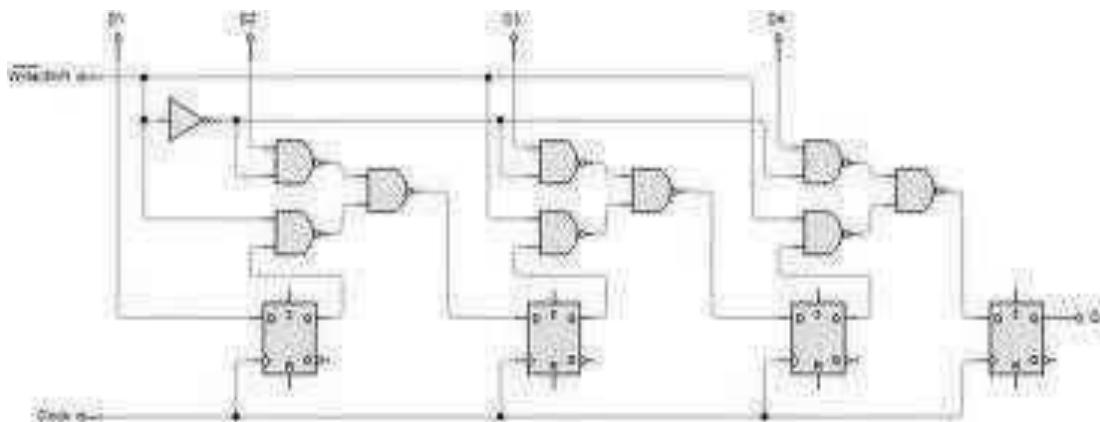
Serial-in, parallel-out, shift register:



In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form.

Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously, rather than on a bit-by-bit basis with the serial output. The serial-in, parallel out, shift register can be used as serial-in, serial out, shift register if the output is taken from the Q terminal of the last FF.

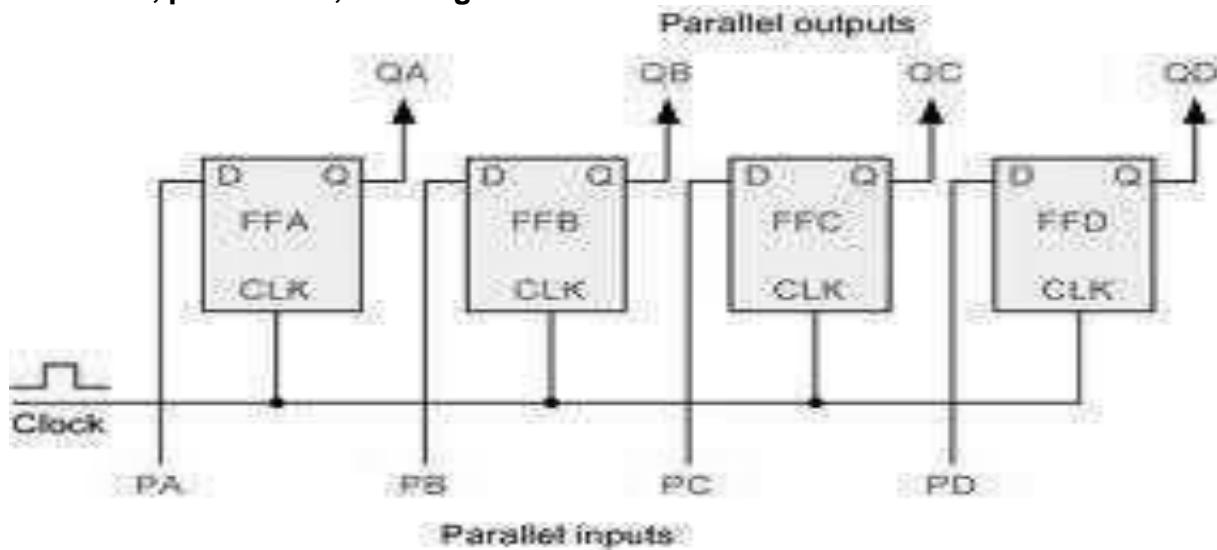
Parallel-in, serial-out, shift register:



For a parallel-in, serial out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by-bit basis on one line as with serial data bits are transferred out of the register serially. On a bit-by-bit basis over a single line.

There are four data lines A,B,C,D through which the data is entered into the register in parallel form. The signal shift/ load allows the data to be entered in parallel form into the register and the data is shifted out serially from terminal Q4

Parallel-in, parallel-out, shift register



In a parallel-in, parallel-out shift register, the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form. Data is applied to the D input terminals of the FF's. When a clock pulse is applied, at the positive going edge of the pulse, the D inputs are shifted into the Q outputs of the FFs. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

Bidirectional shift register:

A bidirectional shift register is one which the data bits can be shifted from left to right or from right to left. A fig shows the logic diagram of a 4-bit serial-in, serial out, bidirectional shift register. Right/left is the mode signal, when right /left is a 1, the logic circuit works as a shift-register.the bidirectional operation is achieved by using the mode signal and two NAND gates and one OR gate for each stage.

A HIGH on the right/left control input enables the AND gates G1, G2, G3 and G4 and disables the AND gates G5,G6,G7 and G8, and the state of Q output of each FF is passed through the gate to the D input of the following FF. when a clock pulse occurs, the data bits are then effectively shifted one place to the right. A LOW on the right/left control inputs enables the AND gates G5, G6, G7 and G8 and disables the And gates G1, G2, G3 and G4 and the Q output of each FF is passed to the D input of the preceding FF. when a clock pulse occurs, the data bits are then effectively shifted one place to the left. Hence, the circuit works as a bidirectional shift register

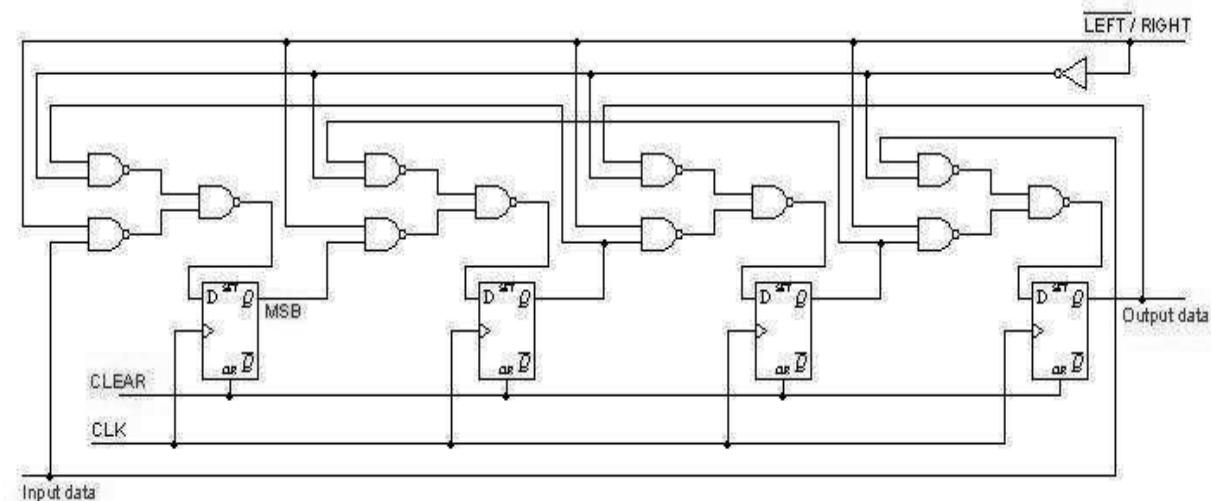


Figure: logic diagram of a 4-bit bidirectional shift register

Universal shift register:

A register is capable of shifting in one direction only is a unidirectional shift register. One that can shift both directions is a bidirectional shift register. If the register has both shifts and parallel load capabilities, it is referred to as a universal shift registers. Universal shift register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be in serial form or I parallel form. The most general shift register has the following capabilities.

1. A clear control to clear the register to 0
2. A clock input to synchronize the operations
3. A shift-right control to enable the shift-right operation and serial input and output lines associated with the shift-right

4. A shift-left control to enable the shift-left operation and serial input and output lines associated with the shift-left
5. A parallel loads control to enable a parallel transfer and the n input lines associated with the parallel transfer
6. N parallel output lines
7. A control state that leaves the information in the register unchanged in the presence of the clock.

A universal shift register can be realized using multiplexers. The below fig shows the logic diagram of a 4-bit universal shift register that has all capabilities. It consists of 4 D flip-flops and four multiplexers. The four multiplexers have two common selection inputs s_1 and s_0 . Input 0 in each multiplexer is selected when $S1S0=00$, input 1 is selected when $S1S0=01$ and input 2 is selected when $S1S0=10$ and input 4 is selected when $S1S0=11$. The selection inputs control the mode of operation of the register according to the functions entries. When $S1S0=0$, the present value of the register is applied to the D inputs of flip-flops. The condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs. When $S1S0=01$, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flop. This causes a shift-right operation, with serial input transferred into flip-flop A4. When $S1S0=10$, a shift left operation results with the other serial input going into flip-flop A1. Finally when $S1S0=11$, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock cycle

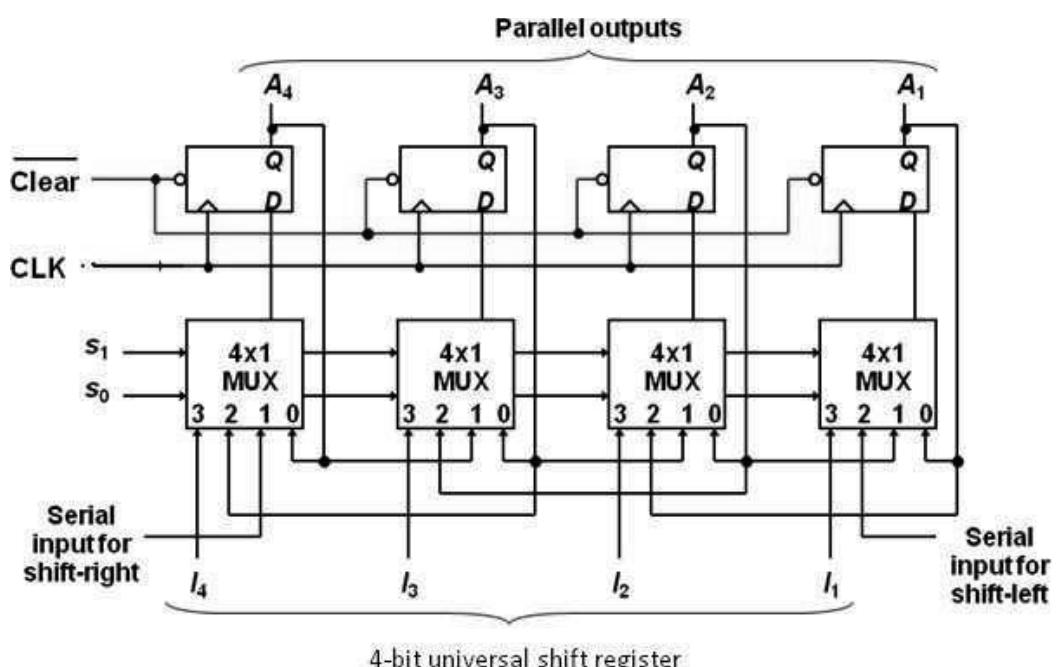


Figure: logic diagram 4-bit universal shift register

Function table for the register

mode control		
S0	S1	register operation
0	0	No change
0	1	Shift Right
1	0	Shift left
1	1	Parallel load

Counters:

Counter is a device which stores (and sometimes displays) the number of times particular event or process has occurred, often in relationship to a clock signal. A Digital counter is a set of flip flops whose state change in response to pulses applied at the input to the counter. Counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters

In electronics counters can be implemented quite easily using register-type circuits such as the flip-flops and a wide variety of classifications exist:

Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops

Synchronous counter – all state bits change under control of a

singleclock Decade counter – counts through ten states per stage

Up/down counter – counts both up and down, under command of a control input

Ring counter – formed by a shift register with feedback connection in a ring

Johnson counter – a twisted ring counter

Cascaded counter

Modulus counter.

Each is useful for different applications. Usually, counter circuits are digital in nature, and count in natural binary. Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters.

Occasionally there are advantages to using a counting sequence other than the natural binary sequence such as the binary coded decimal counter, a linear feed-back shift register counter, or a gray-code counter.

Counters are useful for digital clocks and timers, and in oven timers, VCR clocks, etc.

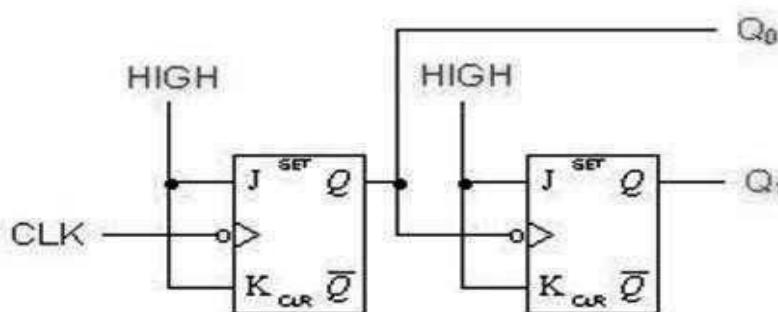
Asynchronous counters:

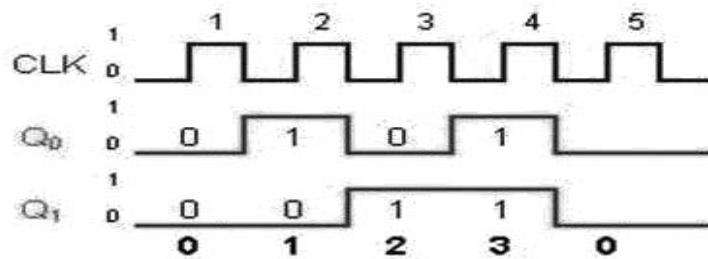
An asynchronous (ripple) counter is a single JK-type flip-flop, with its J (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), one will get another 1 bit counter that counts half as fast. Putting them together yields a two-bit counter:

Two-bit ripple up-counter using negative edge triggered flip flop:

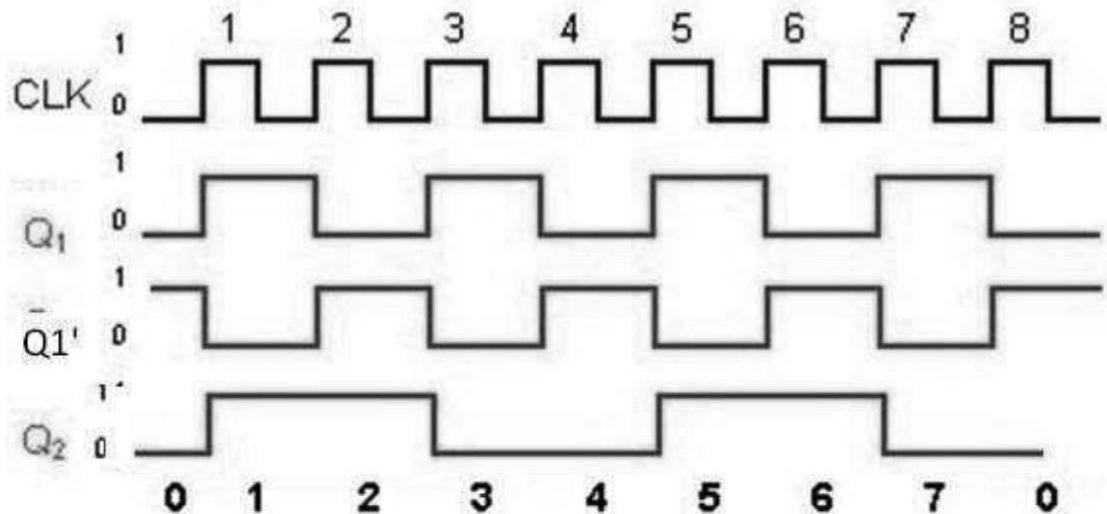
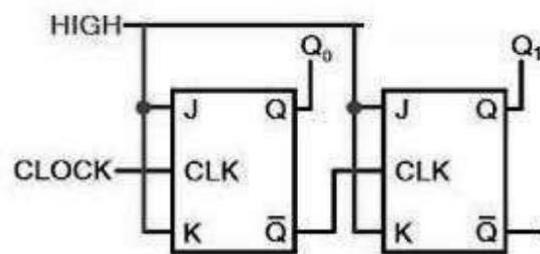
Two bit ripple counter used two flip-flops. There are four possible states from 2 – bit up-counting i.e. 00, 01, 10 and 11.

- The counter is initially assumed to be at a state 00 where the outputs of the two flip-flops are noted as Q_1Q_0 . Where Q_1 forms the MSB and Q_0 forms the LSB.
- For the negative edge of the first clock pulse, output of the first flip-flop FF_1 toggles its state. Thus Q_1 remains at 0 and Q_0 toggles to 1 and the counter state are now read as 01.
- During the next negative edge of the input clock pulse FF_1 toggles and $Q_0 = 0$. The output Q_0 being a clock signal for the second flip-flop FF_2 and the present transition acts as a negative edge for FF_2 thus toggles its state $Q_1 = 1$. The counter state is now read as 10.
- For the next negative edge of the input clock to FF_1 output Q_0 toggles to 1. But this transition from 0 to 1 being a positive edge for FF_2 output Q_1 remains at 1. The counter state is now read as 11.
- For the next negative edge of the input clock, Q_0 toggles to 0. This transition from 1 to 0 acts as a negative edge clock for FF_2 and its output Q_1 toggles to 0. Thus the starting state 00 is attained. Figure shown below





Two-bit ripple down-counter using negative edge triggered flip flop:



A 2-bit down-counter counts in the order 0,3,2,1,0,1.....,i.e, 00,11,10,01,00,11etc. the above fig. shows ripple down counter, using negative edge triggered J-K FFs and its timing diagram.

For down counting, Q_1' of FF1 is connected to the clock of Ff2. Let initially all the FF1 toggles, so, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.

The negative-going signal at Q_1' is applied to the clock input of FF2, toggles FF2 and, therefore, Q2 goes from a 0 to a 1. So, after one clock pulse $Q_2=1$ and $Q_1=1$, i.e., the state of the counter is 11.

At the negative-going edge of the second clock pulse, Q1 changes from a 1 to a 0 and Q_1' from a 0 to a 1.

This positive-going signal at Q_1' does not affect FF2 and, therefore, Q2 remains at a 1. Hence, the state of the counter after second clock pulse is 10.

At the negative going edge of the third clock pulse, FF1 toggles. So Q1, goes from a 0 to a 1 and Q_1' from 1 to 0. This negative going signal at Q_1' toggles FF2 and, so, Q2 changes from 1 to 0, hence, the state of the counter after the third clock pulse is 01.

At the negative going edge of the fourth clock pulse, FF1 toggles. So Q1, goes from a 1 to a 0 and Q_1' from 0 to 1. This positive going signal at Q_1' does not affect FF2 and, so, Q2 remains at 0, hence, the state of the counter after the fourth clock pulse is 00.

Two-bit ripple up-down counter using negative edge triggered flip flop:

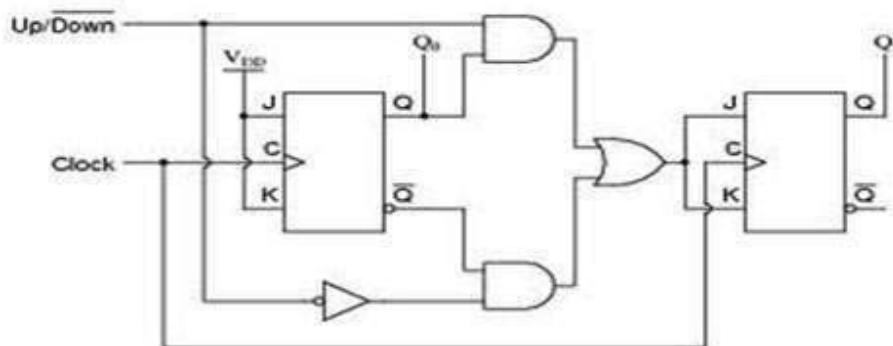


Figure: asynchronous 2-bit ripple up-down counter using negative edge triggered flip flop:

As the name indicates an up-down counter is a counter which can count both in upward and downward directions. An up-down counter is also called a forward/backward counter or a bidirectional counter. So, a control signal or a mode signal M is required to choose the direction of count. When $M=1$ for up counting, Q_1 is transmitted to clock of FF2 and when $M=0$ for down counting, Q_1' is transmitted to clock of FF2. This is achieved by using two AND gates and one OR gate. The external clock signal is applied to FF1.

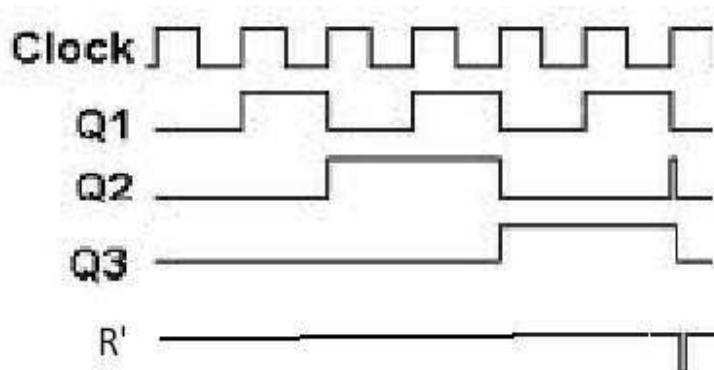
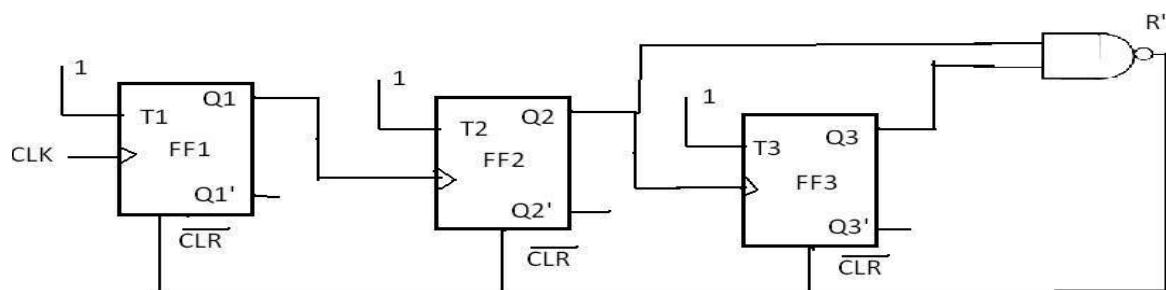
$$\text{Clock signal to FF2} = (Q_1 \cdot \text{Up}) + (Q_1' \cdot \text{Down}) = Q_1 m + Q_1' M'$$

Design of Asynchronous counters:

To design a asynchronous counter, first we write the sequence, then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R and R' using K-Map or any other method. Provide a feedback such that R and R' resets all the FF's after the desired count

Design of a Mod-6 asynchronous counter using T FFs:

A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided. It is —divide by-6-counter, in the sense that it divides the input clock frequency by 6. It requires three FFs, because the smallest value of n satisfying the condition $N \leq 2^n$ is $n=3$; three FFs can have 8 possible states, out of which only six are utilized and the remaining two states 110 and 111, are invalid. If initially the counter is in 000 state, then after the sixth clock pulse, it goes to 001, after the second clock pulse, it goes to 010, and so on.



After sixth clock pulse it goes to 000. For the design, write the truth table with present state outputs Q3, Q2 and Q1 as the variables, and reset R as the output and obtain an expression for R in terms of Q3, Q2, and Q1 that decides the feedback into be provided. From the truth table, $R = Q_3 Q_2 Q_1'$. For active-low Reset, R' is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.

$$R=0 \text{ for } 000 \text{ to } 101, R=1 \text{ for } 110, \text{ and } R=X \text{ for } 111$$

Therefore,

$$R = Q_3 Q_2 Q_1' + Q_3 Q_2 Q_1 = Q_3 Q_2$$

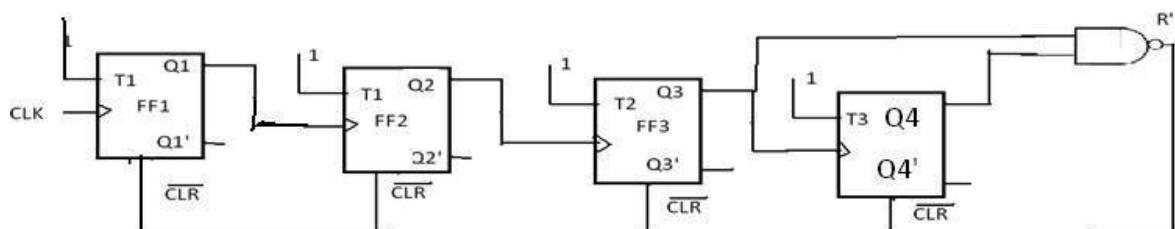
The logic diagram and timing diagram of Mod-6 counter is shown in the above fig.

The truth table is as shown in below.

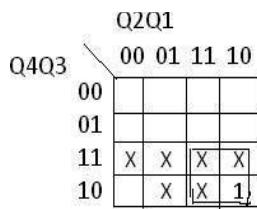
After pulses	States			
	Q3	Q2	Q1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	↓	↓	↓	
0	0	0	0	0
7	0	0	0	0

Design of a mod-10 asynchronous counter using T-flip-flops:

A mod-10 counter is a decade counter. It is also called a BCD counter or a divide-by-10 counter. It requires four flip-flops (condition $10 \leq 2^n$ is $n=4$). So, there are 16 possible states, out of which ten are valid and remaining six are invalid. The counter has ten stable states, 0000 through 1001, i.e., it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001. When the tenth clock pulse is applied, the counter goes to state 1010 temporarily, but because of the feedback provided, it resets to initial state 0000. So, there will be a glitch in the waveform of Q2. The state 1010 is a temporary state for which the reset signal R=1, R=0 for 0000 to 1001, and R=C for 1011 to 1111.



The count table and the K-Map for reset are shown in fig. from the K-Map $R=Q_4Q_2$. So, feedback is provided from second and fourth FFs. For active-HIGH reset, Q_4Q_2 is applied to the clear terminal. For active-LOW reset 4 2 is connected to all flip-flops.

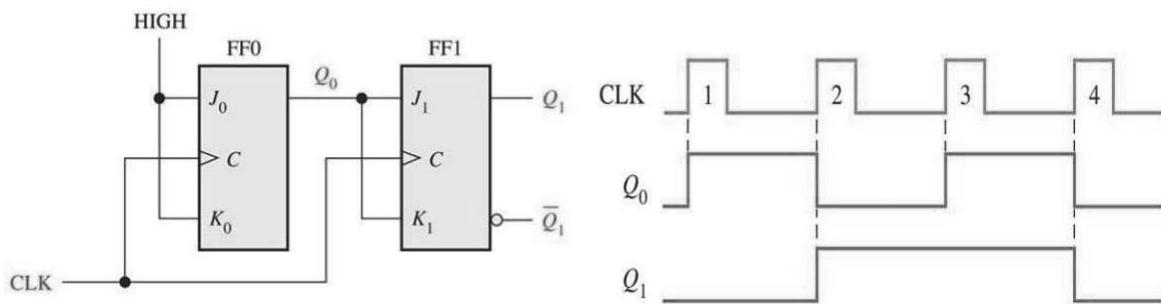


After pulses	Count			
	Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	0	1	0	1
10	0	0	0	0

Synchronous counters:

Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. If the clock frequency is very high, the asynchronous counter may skip some of the states. This problem is overcome in synchronous counters or parallel counters. Synchronous counters are counters in which all the flip flops are triggered simultaneously by the clock pulses. Synchronous counters have a common clock pulse applied simultaneously to all flip-flops.

A 2-Bit Synchronous Binary Counter



Design of synchronous counters:

For a systematic design of synchronous counters, the following procedure is used.

Step 1: State Diagram: draw the state diagram showing all the possible states. State diagram, which also be called nth transition diagrams, is a graphical means of depicting the sequence of states through which the counter progresses.

Step 2: number of flip-flops: based on the description of the problem, determine the required number n of the flip-flops - the smallest value of n is such that the number of states $N \leq 2^n$ --- and the desired counting sequence.

Step 3: choice of flip-flops excitation table: select the type of flip-flop to be used and write the excitation table. An excitation table is a table that lists the present state (ps), the next state(ns) and required excitations.

Step4: minimal expressions for excitations: obtain the minimal expressions for the excitations of the FF using K-maps drawn for the excitation of the flip-flops in terms of the present states and inputs.

Step5: logic diagram: draw a logic diagram based on the minimal expressions

Design of a synchronous 3-bit up-down counter using JK flip-flops:

Step1: determine the number of flip-flops required. A 3-bit counter requires three FFs. It has 8 states (000,001,010,011,101,110,111) and all the states are valid. Hence no don't cares. For selecting up and down modes, a control or mode signal M is required. When the mode signal M=1 and counts down when M=0. The clock signal is applied to all the FFs simultaneously.

Step2: draw the state diagrams: the state diagram of the 3-bit up-down counter is drawn as

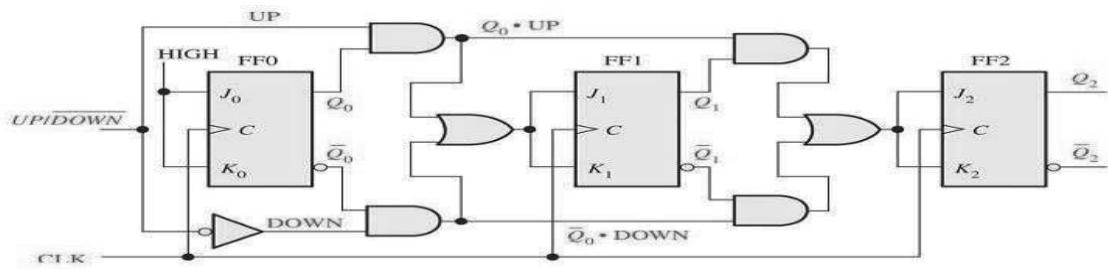
Step3: select the type of flip flop and draw the excitation table: JK flip-flops are selected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in fig.

PS				mode	NS			required excitations						
Q3	Q2	Q1	M		Q3	Q2	Q1	J3	K3	J2	K2	J1	K1	
0	0	0	0		1	1	1	1	x	1	x	1	x	
0	0	0	1		0	0	1	0	x	0	x	1	x	
0	0	1	0		0	0	0	0	x	0	x	x	1	
0	0	1	1		0	1	0	0	x	1	x	x	1	
0	1	0	0		0	0	1	0	x	x	1	1	x	
0	1	0	1		0	1	1	0	x	x	0	1	x	
0	1	1	0		0	1	0	0	x	x	0	x	1	
0	1	1	1		1	0	0	1	x	x	1	x	1	
1	0	0	0		0	1	1	x	1	1	x	1	x	
1	0	0	1		1	0	1	x	0	0	x	1	x	
1	0	1	0		1	0	0	x	0	0	x	x	1	
1	0	1	1		1	1	0	x	0	1	x	x	1	
1	1	0	0		1	0	1	x	0	x	1	1	x	
1	1	0	1		1	1	1	x	0	x	0	1	x	
1	1	1	0		1	1	0	x	0	x	0	x	1	
1	1	1	1		0	0	0	x	1	x	1	x	1	

Step4: obtain the minimal expressions: From the excitation table we can conclude that J1=1 and K1=1, because all the entries for J1 and K1 are either X or 1. The K-maps for J3, K3, J2 and K2 based on the excitation table and the minimal expression obtained from them are shown in fig.

	00	01	11	10
Q3Q2	1		1	
Q1M	X	X	X	X
	X	X	X	X

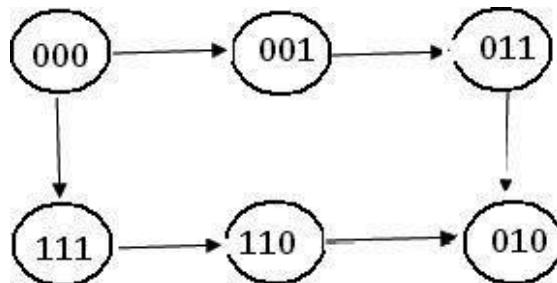
Step5: draw the logic diagram: a logic diagram using those minimal expressions can be drawn as shown in fig.



Design of a synchronous modulo-6 gray cod counter:

Step 1: the number of flip-flops: we know that the counting sequence for a modulo-6 gray code counter is 000, 001, 011, 010, 110, and 111. It requires $n=3$ FFs ($N \leq 2^3$, i.e., $6 \leq 2^3$). 3 FFs can have 8 states. So the remaining two states 101 and 100 are invalid. The entries for excitation corresponding to invalid states are don't cares.

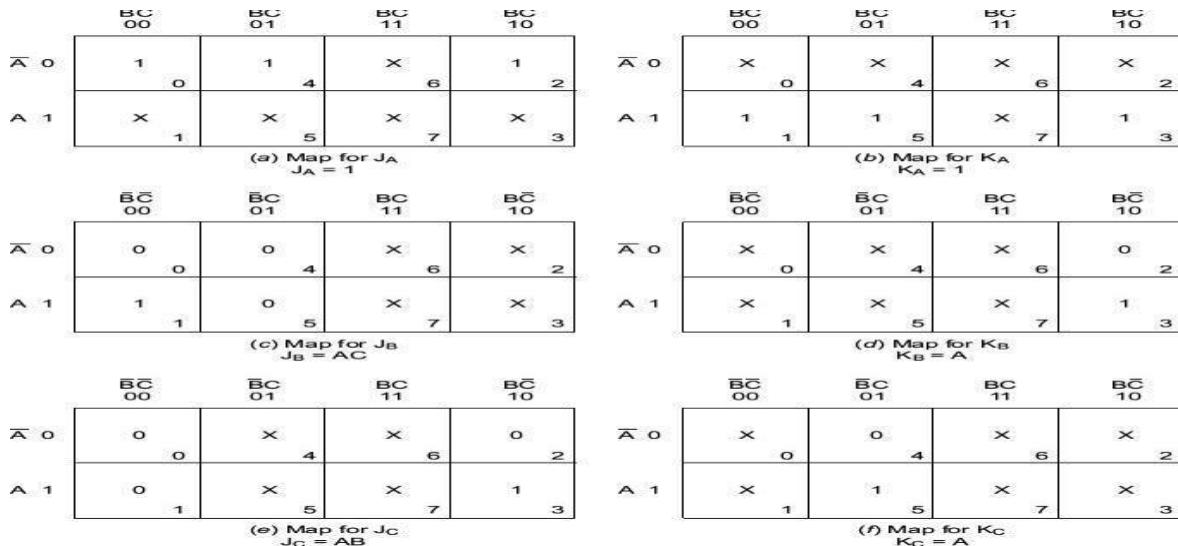
Step2: the state diagram: the state diagram of the mod-6 gray code converter is drawn as shown in fig.



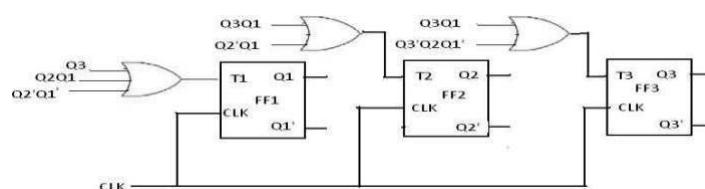
Step3: type of flip-flop and the excitation table: T flip-flops are selected and the excitation table of the mod-6 gray code counter using T-flip-flops is written as shown in fig.

PS			NS			required excitations		
Q3	Q2	Q1	Q3	Q2	Q1	T3	T2	T1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Step4: The minimal expressions: the K-maps for excitations of FFs T3,T2, and T1 in terms of outputs of FFs Q3,Q2, and Q1, their minimization and the minimal expressions for excitations obtained from them are shown if fig



Step5: the logic diagram: the logic diagram based on those minimal expressions is drawn as shown in fig.



Design of a synchronous BCD Up-Down counter using FFs:

Step1: the number of flip-flops: a BCD counter is a mod-10 counter has 10 states (0000 through 1001) and so it requires $n=4$ FFs ($N \leq 2^n$, i.e., $10 \leq 2^4$). 4 FFs can have 16 states. So out of 16 states, six states (1010 through 1111) are invalid. For selecting up and down mode, a control or mode signal M is required. , it counts up when $M=1$ and counts down when $M=0$. The clock signal is applied to all FFs.

Step2: the state diagram: The state diagram of the mod-10 up-down counter is drawn as shown in fig.

Step3: types of flip-flops and excitation table: T flip-flops are selected and the excitation table of the modulo-10 up down counter using T flip-flops is drawn as shown in fig.

The remaining minterms are don't cares ($\sum d(20,21,22,23,24,25,26,37,28,29,30,31)$) from the excitation table we can see that $T1=1$ and the expression for $T4, T3, T2$ are as follows.

$$T4 = \sum m(0, 15, 16, 19) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$

$$T3 = \sum m(7, 15, 16, 8) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$

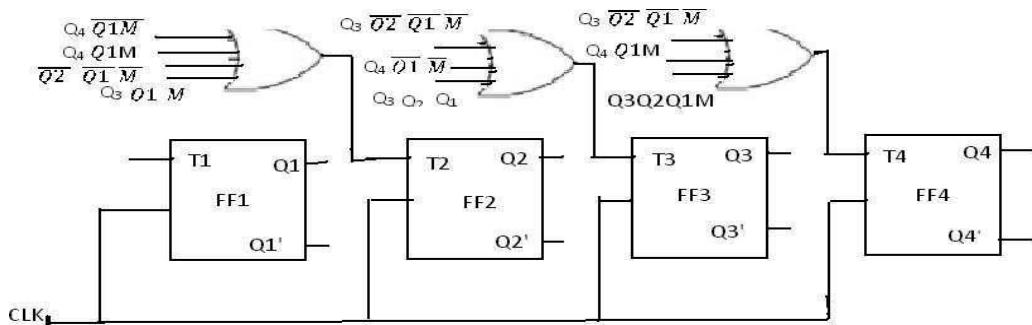
$$T2 = \sum m(3, 4, 7, 8, 11, 12, 15, 16) + d(20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31)$$

PS					NS					required excitations							
				mode	Q4	Q3	Q2	Q1	M	Q4	Q3	Q2	Q1	T4	T3	T2	T1
0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	1	1
0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	1	1	1
0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1
0	1	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1
0	1	1	0	0	0	1	0	0	1	0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	1	1	1	1	0	0	0	0	0	0	1
0	1	1	1	1	1	0	0	0	0	1	0	1	1	1	1	1	1
1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	1

Step4: The minimal expression: since there are 4 state variables and a mode signal, we require 5 variable kmaps. 20 conditions of $Q_4 Q_3 Q_2 Q_1 M$ are valid and the remaining 12 combinations are invalid. So the entries for excitations corresponding to those invalid combinations are don't cares. Minimizing K-maps for T2 we get

$$T_2 = Q_4 Q_1' M + Q_4' Q_1 M + Q_2 Q_1' M' + Q_3 Q_1' M'$$

Step5: the logic diagram: the logic diagram based on the above equation is shown in fig.



Shift register counters:

One of the applications of shift register is that they can be arranged to form several types of counters. The most widely used shift register counter is ring counter as well as the twisted ring counter.

Ring counter: this is the simplest shift register counter. The basic ring counter using D flip-flops is shown in fig. the realization of this counter using JK FFs. The Q output of each stage is connected to the D flip-flop connected back to the ring counter.

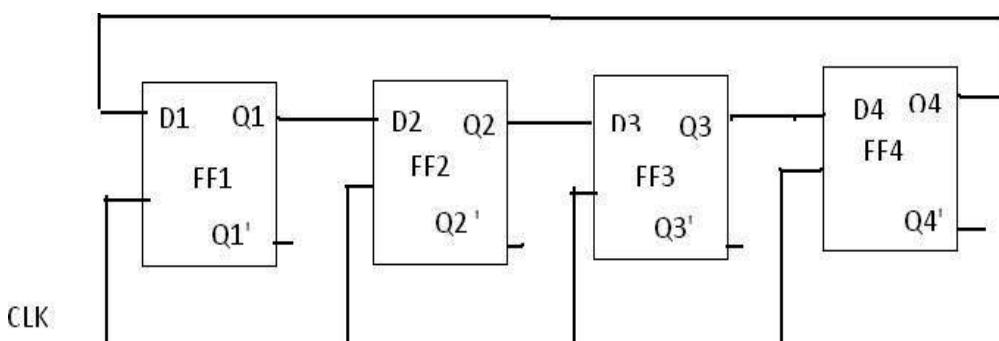


FIGURE: logic diagram of 4-bit ring counter using D flip-flops

Only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially the first FF is present to a 1. So, the initial state is 1000, i.e., $Q_1=1, Q_2=0, Q_3=0, Q_4=0$. After each clock pulse, the contents of the register are shifted to the right by one bit and Q_4 is shifted back to Q_1 . The sequence repeats after four clock pulses. The number

of distinct states in the ring counter, i.e., the mod of the ring counter is equal to number of FFs used in the counter. An n-bit ring counter can count only n bits, whereas n-bit ripple counter can count 2^n bits. So, the ring counter is uneconomical compared to a ripple counter but has advantage of requiring no decoder, since we can read the count by simply noting which FF is set. Since it is entirely a synchronous operation and requires no gates external FFs, it has the further advantage of being very fast.

Timing diagram:

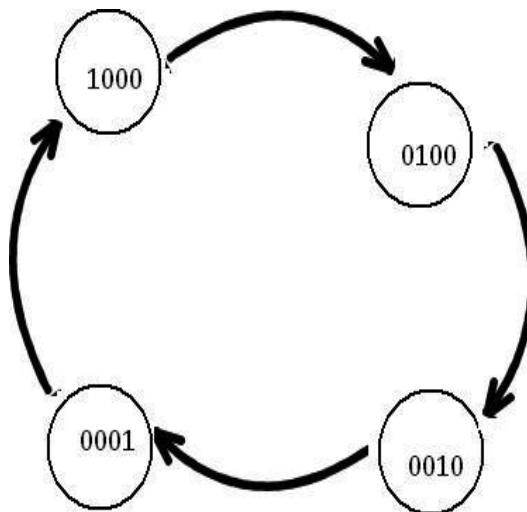
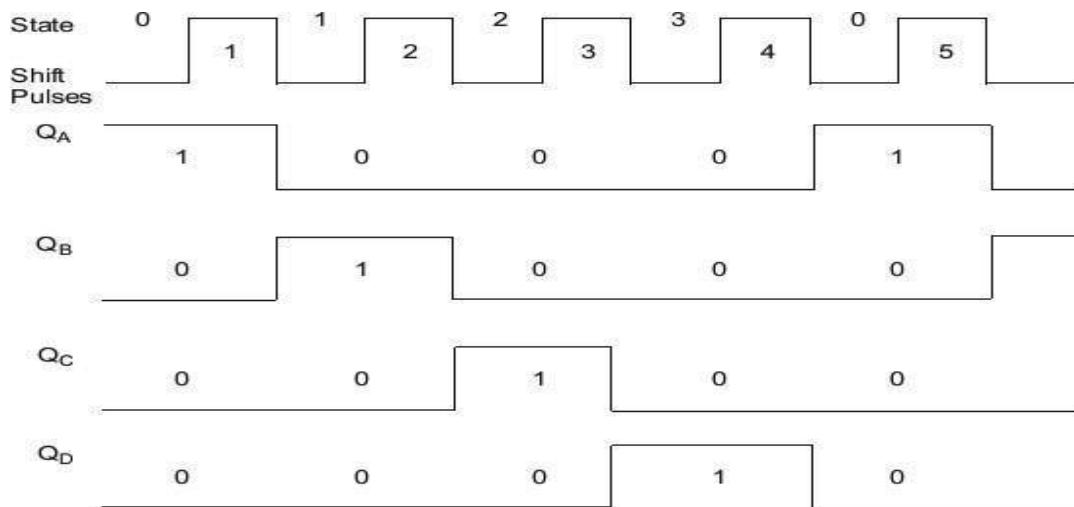


Figure: state diagram

Twisted Ring counter (Johnson counter):

This counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. the Q output of each is connected to the D input of the next stage, but the Q' output of the last stage is connected to the D input of the first stage, therefore, the name twisted ring counter. This feedback arrangement produces a unique sequence of states.

The logic diagram of a 4-bit Johnson counter using D FF is shown in fig. the realization of the same using J-K FFs is shown in fig.. The state diagram and the sequence table are shown in figure. The timing diagram of a Johnson counter is shown in figure.

Let initially all the FFs be reset, i.e., the state of the counter be 0000. After each clock pulse, the level of Q1 is shifted to Q2, the level of Q2to Q3, Q3 to Q4 and the level of Q4' to Q1 and the sequences given in fig.

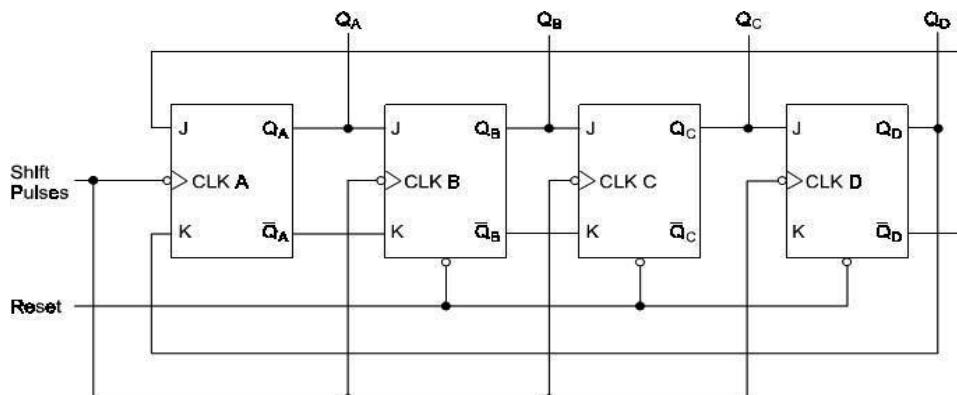


Figure: Johnson counter with JK flip-flops

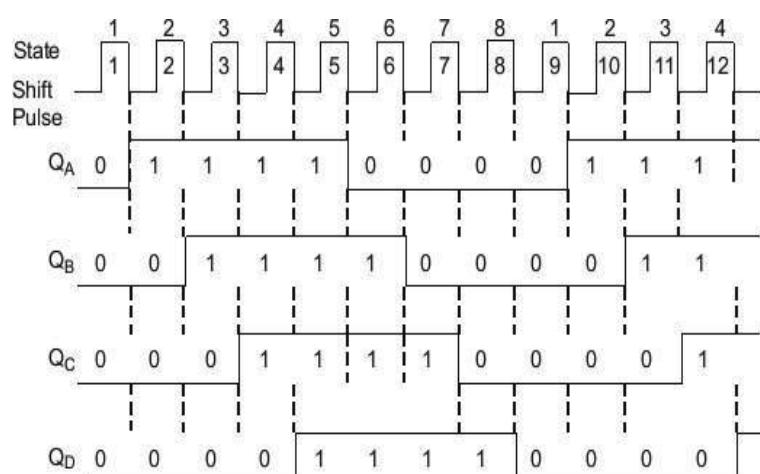
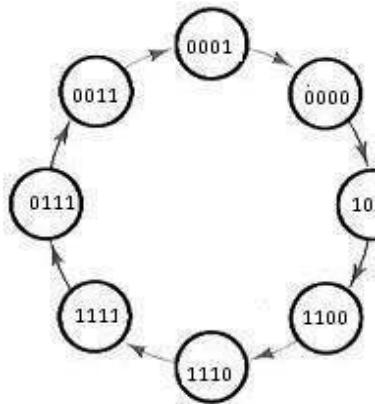


Figure: timing diagram

State diagram:

					<u>after clock pulse</u>
Q1	Q2	Q3	Q4		
0	0	0	0	0	0
1	0	0	0	1	1
1	1	0	0	2	2
1	1	1	0	3	3
1	1	1	1	4	4
0	1	1	1	5	5
0	0	1	1	6	6
0	0	0	1	7	7
0	0	0	0	8	8
1	0	0	0	9	9

Excitation table**Synthesis of sequential circuits:**

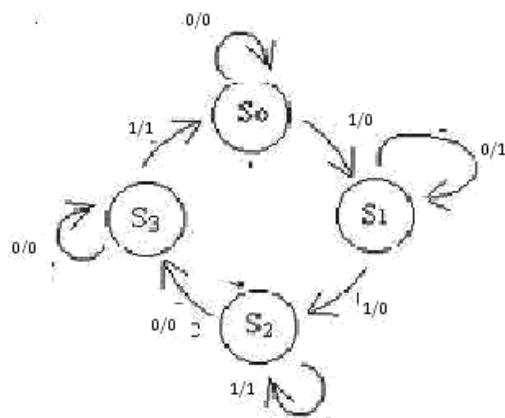
The synchronous or clocked sequential circuits are represented by two models.

1. Moore circuit: in this model, the output depends only on the present state of the flip-flops
2. Meelay circuit: in this model, the output depends on both present state of the flip-flop. And the inputs.

Sequential circuits are also called finite state machines (FSMs). This name is due to the fact that the functional behavior of these circuits can be represented using a finite number of states.

State diagram: the state diagram or state graph is a pictorial representation of the relationships between the present state, the input, the next state, and the output of a sequential circuit. The state diagram is a pictorial representation of the behavior of a sequential circuit.

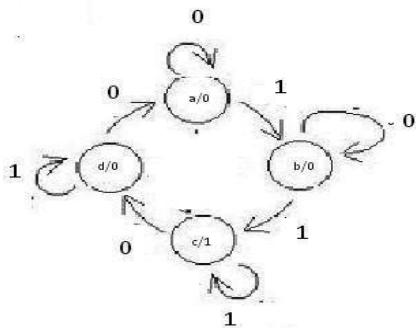
The state represented by a circle also called the node or vertex and the transition between states is indicated by directed lines connecting circle. a directed line connecting a circle with itself indicates that the next state is the same as the present state. The binary number inside each circle identifies the state represented by the circle. The direct lines are labeled with two binary numbers separated by a symbol. The input value is applied during the present state is labeled after the symbol.

**Fig :a) state diagram (meelay circuit)**

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

fig: b) state table

In case of moore circuit ,the directed lines are labeled with only one binary number representing the input that causes the state transition. The output is indicated with in the circle below the present state, because the output depends only on the present state and not on the input.



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

Fig: a) state diagram (moore circuit)**fig:b) state table**

Serial binary adder:

Step1: word statement of the problem: the block diagram of a serial binary adder is shown in fig. 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 0s and 1s.the output of the serial 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-1} and of carry which had been generated at t_{i-1} .

1. The carry which represent the past history of the serial adder may be a 0 or 1. The circuit has two states. If one state indicates that carry from the previous addition is a 0, the other state indicates that the carry from the previous addition is a 1

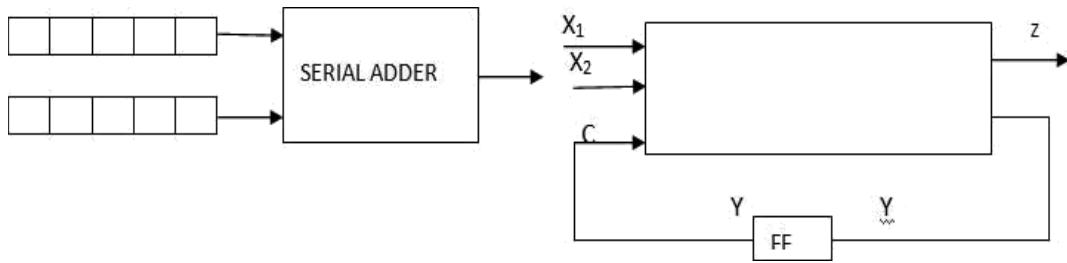


Figure: block diagram of serial binary adder

Step2 and 3: state diagram and state table: let a designate the state of the serial adder at t_i if a carry 0 was generated at t_{i-1} , and let b designate the state of the serial adder at t_i if carry 1 was generated at t_{i-1} . the state of the adder at that time when the present inputs are applied is referred to as the present state(PS) and the state to which the adder goes as a result of the new carry value is referred to as next state(NS).

The behavior of serial adder may be described by the state diagram and state table.



Figures: serial adder state diagram and state table

If the machine is in state B, i.e., carry from the previous addition is a 1, inputs $X_1=0$ and $X_2=1$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=0$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=1$ gives sum, 1 and carry 0. So the machine remains in state B and outputs a 1. Inputs $X_1=0$ and $X_2=0$ gives sum, 1 and carry 0. So the machine goes to state A and outputs a 1. The state table also gives the same information.

Setp4: reduced standard from state table: the machine is already in this form. So no need to do anything

Step5: state assignment and transition and output table:

The states, $A=0$ and $B=1$ have already been assigned. So, the transition and output table is as shown.

PS		NS		O/P			
0	0	1	0	1	0	0	1
0	1	0	1	0	1	0	1
0	0	0	0	1	0	1	1
1	0	1	1	1	1	0	1

STEP6: choose type of FF and excitation table: to write table, select the memory element the excitation table is as shown in fig.

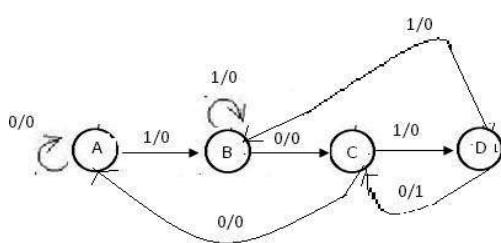
PS	I/P	NS	I/P-FF	O/P
Y	x1	x2	D	Z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sequence detector:

Step1: 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.

Step2 and 3: state diagram and state table: the state diagram and the state table of the sequence detector. At the time t_1 , the machine is assumed to be in the initial state designed 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 0, the machine does not start the detection process because the first bit in the desired sequence is a 1. If the input bit is a 1 the detection process starts.



PS	NS,Z
X=0	X=1
A	A,0
B	C,0
C	A,0
D	C,1

Figure: state diagram and state table of sequence detector

So, the machine goes to state B and outputs a 0. While in state B, the machinery may receive 0 or 1 bit. If the bit is 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 0.. if the bit is a 1, the two bits become 11 and this is not a part of the valid sequence

Step4: reduced standard form state table: the machine is already in this form.
So no need to do anything.

Step5: 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. Let $A=00$, $B=01$, $C=10$ and $D=11$ be the state assignment.

PS(y ₁ y ₂)	NS(Y ₁ Y 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	1 0	1 1	0

Step6: choose type of flip-flops and form the excitation table: select the D flip-flops as memory elements and draw the excitation table.

PS y ₁	y ₂	I/P X	NS Y ₁	INPUTS			O/P Z
				Y ₂	D ₁	D ₂	
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

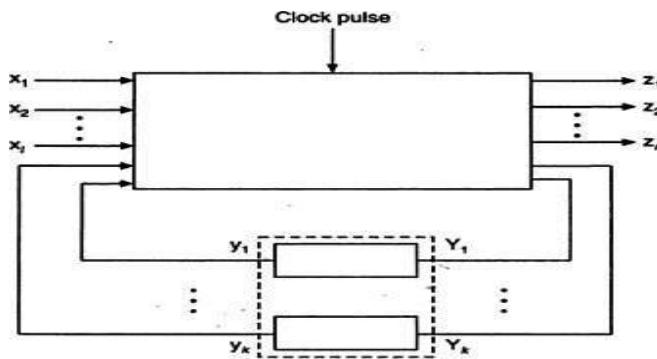
Step7: K-maps and minimal functions: based on the contents of the excitation table , draw the k-map and simplify them to obtain the minimal expressions for D₁ and D₂ in terms of y₁, y₂ and x as shown in fig. The expression for z (z=y₁,y₂) can be obtained directly from table

Step8: implementation: the logic diagram based on these minimal expressions

Finite State Machine:

Finite state machine can be defined as a type of machine whose past histories can affect its future behavior in a finite number of ways. To clarify, consider for example of binary full adder. Its output depends on the present input and the carry generated from the previous input. It may have a large number of previous input histories but they can be divided into two types: (i) Input

The most general model of a sequential circuit has inputs, outputs and internal states. A sequential circuit is referred to as a finite state machine (FSM). A finite state machine is abstract model that describes the synchronous sequential machine. The fig. shows the block diagram of a finite state model. X_1, X_2, \dots, X_l are inputs. Z_1, Z_2, \dots, Z_m are outputs. Y_1, Y_2, \dots, Y_k are state variables, and Y_1, Y_2, \dots, Y_k represent the next state.



Capabilities and limitations of finite-state machine

Let a finite state machine have n states. Let a long sequence of input be given to the machine. The machine will progress starting from its beginning state to the next states according to the state transitions. However, after some time the input string may be longer than n , the number of states. As there are only n states in the machine, it must come to a state it was previously been in and from this phase if the input remains the same the machine will function in a periodically repeating fashion. From here a conclusion that for a n state machine the output will become periodic after a number of clock pulses less than equal to n can be drawn. States are memory elements. As for a finite state machine the number of states is finite, so finite number of memory elements are required to design a finite state machine.

Limitations:

- Periodic sequence and limitations of finite states:** with n -state machines, we can generate periodic sequences of n states are smaller than n states. For example, in a 6-state machine, we can have a maximum periodic sequence as 0,1,2,3,4,5,0,1....
- No infinite sequence:** consider an infinite sequence such that the output is 1 when and only when the number of inputs received so far is equal to $P(P+1)/2$ for $P=1,2,3,\dots$, i.e., the desired input-output sequence has the following form:

Input: x
Output: 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1

Such an infinite sequence cannot be produced by a finite state machine.

3. Limited memory: the finite state machine has a limited memory and due to limited memory it cannot produce certain outputs. Consider a binary multiplier circuit for multiplying two arbitrarily large binary numbers. The memory is not sufficient to store arbitrarily large partial products resulted during multiplication.

Finite state machines are two types. They differ in the way the output is generated they are:

1. Mealy type model: in this model, the output is a function of the present state and the present input.
2. Moore type model: in this model, the output is a function of the present state only.

Mathematical representation of synchronous sequential machine:

The relation between the present state $S(t)$, present input $X(t)$, and next state $s(t+1)$ can be given as

$$S(t+1) = f\{S(t), X(t)\}$$

The value of output $Z(t)$ can be given as

$$Z(t) = g\{S(t), X(t)\} \text{ for mealy model}$$

$$Z(t) = G\{S(t)\} \text{ for Moore model}$$

Because, in a mealy machine, the output depends on the present state and input, whereas in a Moore machine, the output depends only on the present state.

Comparison between the Moore machine and mealy machine:

Moore machine	mealy machine
1. its output is a function of present state only $Z(t) = g\{S(t)\}$	1. its output is a function of present state as well as present input $Z(t) = g\{S(t), X(t)\}$
2. input changes do not affect the output	2. input changes may affect the output of the circuit
3. it requires more number of states for implementing same function	3. it requires less number of states for implementing same function

Mealy model:

When the output of the sequential circuit depends on both the present state of the flip-flops and on the inputs, the sequential circuit is referred to as mealy circuit or mealy machine.

The fig. shows the logic diagram of the mealy model. Notice that the output depends upon the present state as well as the present inputs. We can easily realize that changes in the input during the clock pulse cannot affect the state of the flip-flop. They can affect the output of the circuit. If the input variations are not synchronized with a clock, the derived output will also not be synchronized with the clock and we get false output. The false outputs can be eliminated by allowing input to change only at the active transition of the clock.

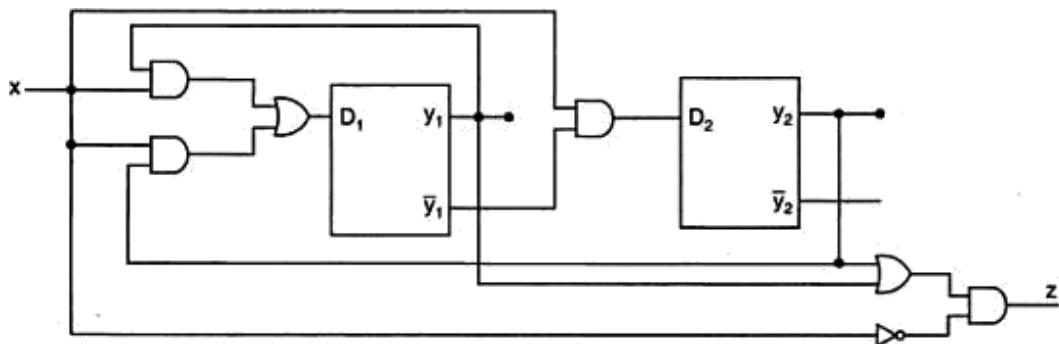


Fig: logic diagram of a mealy model

The behavior of a clocked sequential circuit can be described algebraically by means of state equations. A state equation specifies the next state as a function of the present state and inputs. The mealy model shown in fig. consists of two D flip-flops, an input x and an output z . since the D input of a flip-flop determines the value of the next state, the state equations for the model can be written as

$$Y_1(t+1) = y_1(t)x(t) + y_2(t)x(t)$$

$$Y_2(t+1) = 1(t)x(t)$$

And the output equation is

$$Z(t) = \{y_1(t) + y_2(t)\} X'(t)$$

Where $y(t+1)$ is the next state of the flip-flop one clock edge later, $x(t)$ is the present input, and $z(t)$ is the present output. If $y_1(t+1)$ are represented by $y_1(t)$ and $y_2(t)$, in more compact form, the equations are

$$Y_1(t+1) = y_1 = y_1x + y_2x$$

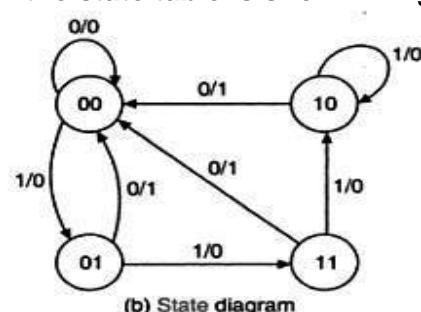
$$Y_2(t+1) = y_2 = y_1'x$$

$$Z = (y_1 + y_2)x'$$

The stable table of the mealy model based on the above state equations and output equation is shown in fig. the state diagram based on the state table is shown in fig.

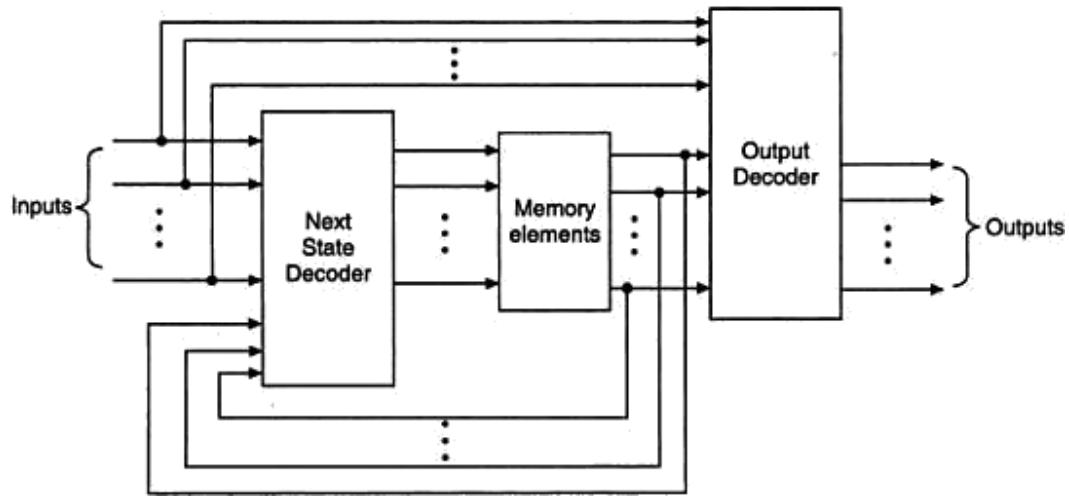
PS	NS		O/P	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
y_1	y_2	y_1	y_2	z
0	0	0 0	0 1	0 0
0	1	0 0	1 1	1 0
1	0	0 0	1 0	1 0
1	1	0 0	1 0	1 0

(a) State table



(b) State diagram

In general form, the mealy circuit can be represented with its block schematic as shown in below fig.



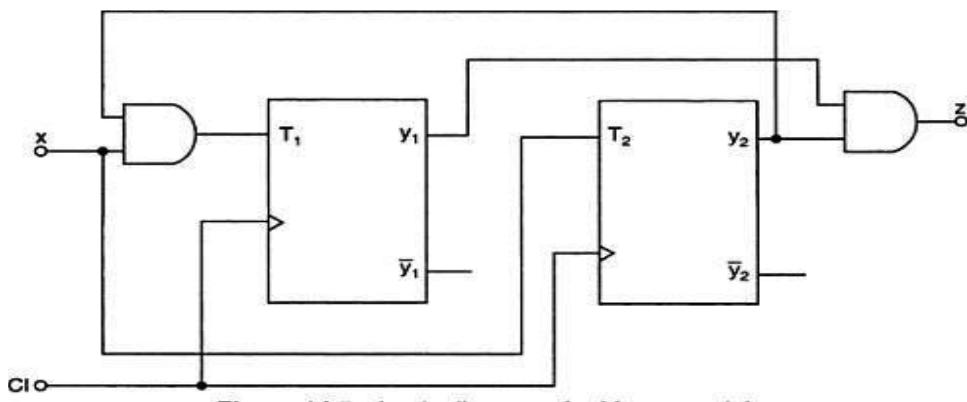
Moore model: when the output of the sequential circuit depends up only on the present state of the flip-flop, the sequential circuit is referred as to as the Moore circuit or the Moore machine.

Notice that the output depend only on the present state. It does not depend upon the input at all. The input is used only to determine the inputs of flip-flops. It is not used to determine the output. The circuit shown has two T flip-flops, one input x , and one output z . it can be described algebraically by two input equations an output equation.

$$T_1 = y_2 x$$

$$T_2 = x$$

$$Z = y_1 y_2$$



Characteristic equation of a T-flip-flop is $Q(t+1) = TQ' + T'Q$

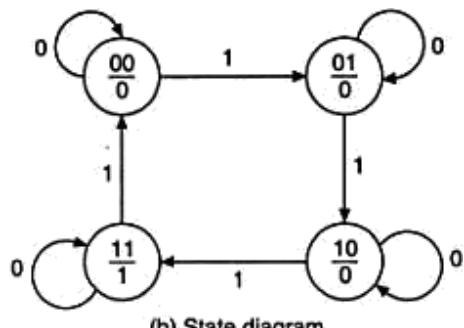
The values for the next state can be derived from the state equations by substituting T_1 and T_2 in the characteristic equation yielding

$$\begin{aligned}
 Y_1(t+1) &= Y_1 = (y_2 x) \oplus = (2)y_1 + (y_2 x) 1 \\
 &= y_1 2 + y_1 + 1y_2 x \\
 &= y_2 (t+1) = x \oplus y_2 = x 2 + y_2
 \end{aligned}$$

The state table of the Moore model based on the above state equations and output equation is shown in fig.

PS	NS				O/P	
	x = 0		x = 1			
y_1	y_2	Y_1	Y_2	Y_1	Y_2	z
0	0	0	0	0	1	0
0	1	0	1	1	0	0
1	0	1	0	1	1	0
1	1	1	1	0	0	1

(a) State table



(b) State diagram

In general form , the Moore circuit can be represented with its block schematic as shown in below fig.

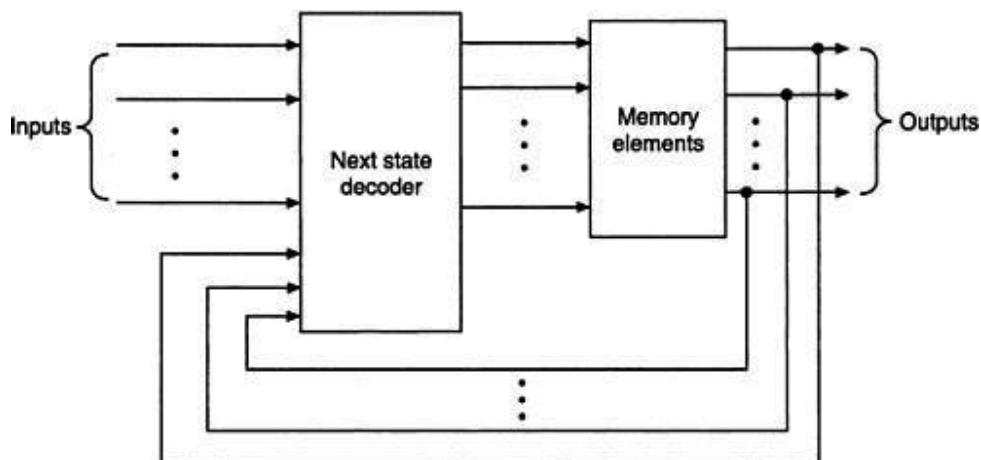


Figure: moore circuit model:

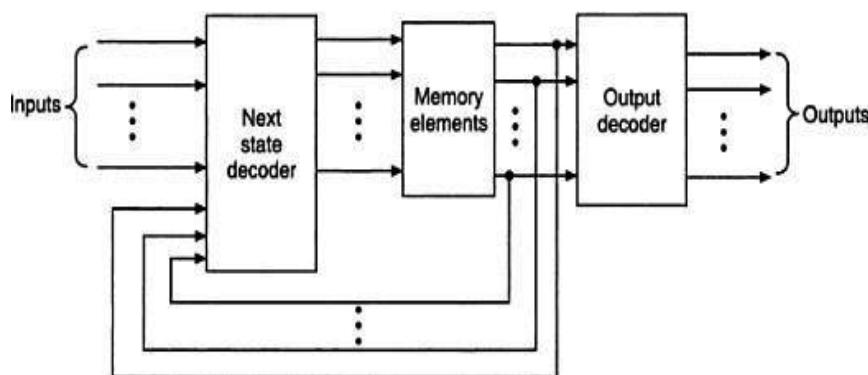
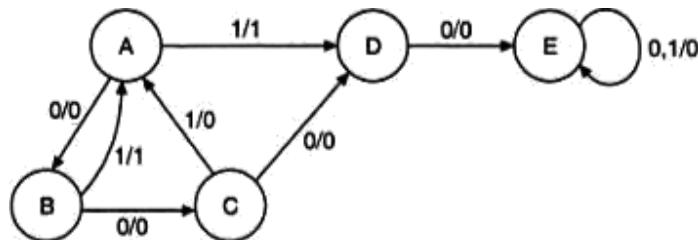


Figure: moore circuit model with an output decoder

Important definitions and theorems:

A). Finite state machine-definitions:

Consider the state diagram of a finite state machine shown in fig. it is five-state machine with one input variable and one output variable.



Successor: looking at the state diagram when present state is A and input is 1, the next state is D. this condition is specified as D is the successor of A. similarly we can say that A is the 1 successor of B, and C,D is the 11 successor of B and C, C is the 00 successor of A and D, D is the 000 successor of A,E, is the 10 successor of A or 0000 successor of A and so on.

Terminal state: looking at the state diagram , we observe that no such input sequence exists which can take the sequential machine out of state E and thus state E is said to be a terminal state.

Strongly-connected machine: in sequential machines many times certain subsets of states may not be reachable from other subsets of states. Even if the machine does not contain any terminal state. If for every pair of states s_i, s_j , of a sequential machine there exists an input sequence which takes the machine M from s_i to s_j , then the sequential machine is said to be strongly connected.

B). state equivalence and machine minimization:

In realizing the logic diagram from a stat table or state diagram many times we come across redundant states. Redundant states are states whose functions can be accomplished by other states. The elimination of redundant states reduces the total number of states of the machines which in turn results in reduction of the number of flip-flops and logic gates, reducing the cost of the final circuit.

Two states are said to be equivalent. When two states are equivalent, one of them can be removed without altering the input output relationship.

State equivalence theorem: it states that two states s_1 , and s_2 are equivalent if for every possible input sequence applied. The machine goes to the same next state and generates the same output. That is

If $S_1(t+1)=S_2(t+1)$ and $Z_1=Z_2$, then $S_1=S_2$

C). distinguishable states and distinguishing sequences:

Two states s_a , and s_b of a sequential machine are distinguishable, if and only if there exists at least one finite input sequence which when applied to the sequential machine causes different outputs sequences depending on weather s_a or s_b is the initial state.

Consider states A and B in the state table, when input $X=0$, their outputs are 0 and 1 respectively and therefore, states A and B are called 1-distinguishable. Now consider states A and E . the output sequence is as follows.

$X=0$ A C,0 and E D, 0 ; outputs are the same



$C \rightarrow E, 0$ and $D \rightarrow b, 1$; outputs are different

Here the outputs are different after 2-state transition and hence states A and E are 2-distinguishable. Again consider states A and C. the output sequence is as follows:

$X=0 \quad A \rightarrow C, 0$ and $C \rightarrow E, 0$; outputs are the same

$C \rightarrow E, 0$ and $E \rightarrow D, 0$; outputs are the

same E \rightarrow $D, 0$ – and D $\rightarrow B, 1$; outputs are

different

Here the outputs are different after 3-transition and hence states A and B are 3-distinguishable. the concept of K-distinguishable leads directly to the definition of K-equivalence. States that are not K-distinguishable are said to be K-equivalent.

Truth table for Distungishable states:

PS	NS,Z	
	<u>X=0</u>	<u>X=1</u>
A	C,0	F,0
B	D,1	F,0
C	E,0	B,0
D	B,1	E,0
E	D,0	B,0
F	D,1	B,0

Merger Chart Methods:

Merger graphs:

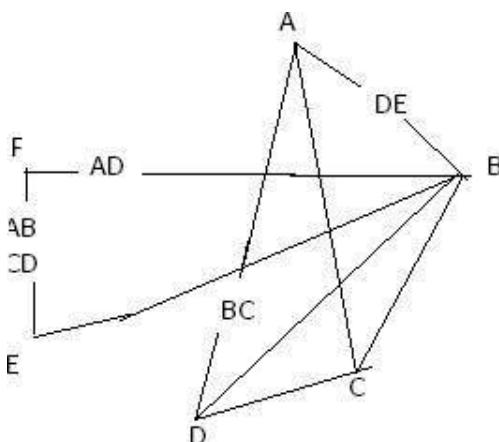
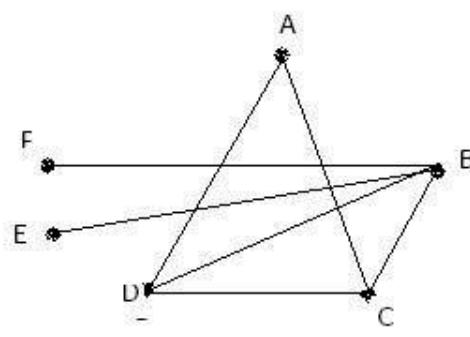
The merger graph is a state reducing tool used to reduce states in the incompletely specified machine. The merger graph is defined as follows.

1. Each state in the state table is represented by a vertex in the merger graph. So it contains the same number of vertices as the state table contains states.
2. Each compatible state pair is indicated by an unbroken line draw between the two state vertices
3. Every potentially compatible state pair with non-conflicting outputs but with different next states is connected by a broken line. The implied states are written in the line break between the two potentially compatible states.
4. If two states are incompatible no connecting line is drawn.

Consider a state table of an incompletely specified machine shown in fig. the corresponding merger graph shown in fig.

State table:

PS	NS,Z			
	I1	I2	I3	I4
A	...	E,1	B,1
B	...	D,1	...	F,1
C	F,1
D	C,1	...
E	C,0	...	A,0	F,1
F	D,0	A,1	B,0	...

**a) Merger graph****b) simplified merger graph**

States A and B have non-conflicting outputs, but the successor under input I₂ are compatible only if implied states D and E are compatible. So, draw a broken line from A to B with DE written in between states A and C are compatible because the next states and output entries of states A and C are not conflicting. Therefore, a line is drawn between nodes A and C. states A and D have non-conflicting outputs but the successor under input I₃ are B and C. hence join A and D by a broken line with BC entered In between.

Two states are said to be incompatible if no line is drawn between them. If implied states are incompatible, they are crossed and the corresponding line is ignored. Like, implied states D and E are incompatible, so states A and B are also incompatible. Next, it is necessary to check whether the incompatibility of A and B does not invalidate any other broken line. Observe that states E and F also become incompatible because the implied pair AB is incompatible. The broken lines which remain in the graph after all the implied pairs have been verified to be compatible are regarded as complete lines. After checking all possibilities of incompatibility, the merger graph gives the following seven compatible pairs.

$$(A, C) (A, D) (B, C) (B, D) (C, D) (B, E) (B, F)$$

These compatible pairs are further checked for further compatibility. For example, pairs (B,C)(B,D)(C,D) are compatible. So (B, C, D) is also compatible. Also pairs (A,c)(A,D)(C,D) are compatible. So (A,C,D) is also compatible. . In this way the entire set of compatibles of sequential machine can be generated from its compatible pairs. To find the minimal set of compatibles for state reduction, it is useful to find what are called the maximal compatibles. A set of compatibles state pairs is said to be maximal, if it is not completely covered by any other set of compatible state pairs. The maximum compatible can be found by looking at the merger graph for polygons which are not contained within any higher order complete polygons. For example only triangles (A, C,D) and (B,C,D) are of higher order. The set of maximal compatibles for this sequential machine given as

$$(A, C, D) (B, C, D) (B, E) (B, F)$$

Example:

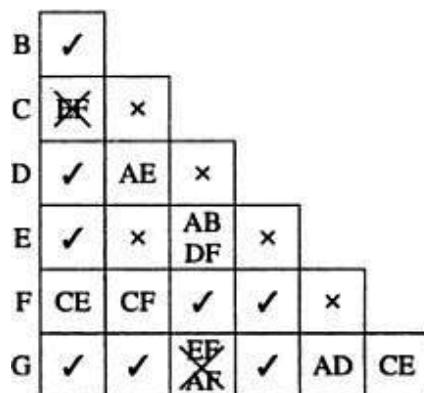
Draw the merger graph and obtain the set of maximal compatibles for the incompletely specified sequential machine whose state table is given in Table 7.24.

Table 7.24 Example 7.9: State table

PS	NS, Z	
	I ₁	I ₂
A	E, 0	B, 0
B	F, 0	A, 0
C	E, -	C, 0
D	F, 1	D, 0
E	C, 1	C, 0
F	D, -	B, 0

mark \times in the corresponding cell. For example, states B and C are incompatible because their outputs are conflicting and hence the cell corresponding to them contains a cross mark \times . Similarly states B, E; D, E; E, F are incompatible. Hence put a \times mark in the corresponding cells. On the other hand, states A and B are compatible and hence the cell corresponding to them contains the check mark \checkmark . Similarly, cells corresponding to states A, D; A, E; A, G; B, G; C, F; D, F ; D, G are also compatible. So a check mark is put in those cells also. The implied pairs or pairs corresponding to the state pair are written within the cell as shown in Table 7.26. For example, states A and C are compatible only when implied states E and F are compatible. Therefore, EF is written in the cell corresponding to states A and C. States C and E are compatible only when implied states A and B, and D and F are compatible. So AB and DF are written in the cell corresponding to states C and E. In a similar way, the entire merger table is written. Now it is necessary to check whether the implied pairs are compatible or not by observing the merger table. The implied states are incompatible if the corresponding cell contains a \times . For example, implied pair E, F is incompatible because cell EF contains a \times . Similarly, implied pairs EF, AF are incompatible because EF contains a \times . It is indicated by a \times .

PS	NS, Z			
	00	01	11	10
A	E, 0	-	-	-
B	-	F, 1	E, 1	A, 1
C	F, 0	-	A, 0	F, 1
D	-	-	A, 1	-
E	-	C, 0	B, 0	D, 1
F	C, 0	C, 1	-	-
G	E, 0	-	-	A, 1

Figure: state table

State Minimization: Completely Specified Machines

Two states, s_i and s_j of machine M are distinguishable if and only if there exists a finite input sequence which when applied to M causes different output sequences depending on whether M started in s_i or s_j .

Such a sequence is called a distinguishing sequence for (s_i, s_j) .

If there exists a distinguishing sequence of length k for (s_i, s_j) , they are said to be k-distinguishable.

EXAMPLE:

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

- states A and B are 1-distinguishable, since a 1 input applied to A yields an output 1, versus an output 0 from B.
- states A and E are 3-distinguishable, since input sequence 111 applied to A yields output 100, versus an output 101 from E.
- States s_i and s_j ($s_i \sim s_j$) are said to be equivalent iff no distinguishing sequence exists for (s_i, s_j) .
- If $s_i \sim s_j$ and $s_j \sim s_k$, then $s_i \sim s_k$. So state equivalence is an equivalence relation (i.e. it is a reflexive, symmetric and transitive relation).
- An equivalence relation partitions the elements of a set into equivalence classes.
- Property: If $s_i \sim s_j$, their corresponding X-successors, for all inputs X, are also equivalent.
- Procedure: Group states of M so that two states are in the same group iff they are equivalent (forms a partition of the states).

Completely Specified Machines

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

P_i : partition using distinguishing sequences of length i.

Partition: Distinguishing Sequence:

$P_0 = (ABCDEF)$

$P_1 = (A C E)(BD F)$

$P_2 = (A C E)(B D)(F)$

$P_3 = (A C)(E)(B D)(F)$

$x = 1$

$x = 1; x = 1$

$x = 1; x = 1; x = 1$

- All states equivalent to each other form an equivalence class. These may be combined into one state in the reduced (quotient) machine.
- Start an initial partition of a single block. Iteratively refine this partition by separating the 1-distinguishable states, 2-distinguishable states and so on.
- To obtain P_{k+1} , for each block B_i of P_k , create one block of states that are not 1-distinguishable within B_i , and create different blocks of states that are 1-distinguishable within B_i .

Theorem: The equivalence partition is unique.

Theorem: If two states, s_i and s_j , of machine M are distinguishable, then they are $(n-1)$ -distinguishable, where n is the number of states in M.

Definition: Two machines, M_1 and M_2 , are equivalent ($M_1 \sim M_2$) if, for every state in M_1 there is a corresponding equivalent state in M_2 and vice versa.

Theorem. For every machine M there is a minimum machine $M_{\text{red}} \sim M$. M_{red} is unique up to isomorphism.

Completely Specified Machines

- Reduced machine obtained from previous example:

$$\begin{aligned} P_4 &= (A C)(E)(B D)(F) \\ &= \alpha \beta \gamma \delta \end{aligned}$$

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

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

State Minimization of CSMs: Complexity

Algorithm DFA \sim DFA_{min}

Input: A finite automaton $M = (Q, \Sigma, q_0, F)$ with no unreachable states.

Output: A minimum finite automaton $M' = (Q', \Sigma, q'_0, F')$.

Method:

1. $t := 2$; $Q_0 := \{ \text{undefined} \}$; $Q_1 := F$; $Q_2 := Q \setminus F$.

2. while there is $0 < i < t$, a with $(q_i, a) \in Q_i$, for all $j < t$

do (a) Choose such an i , a , and j with $(q_i, a) \in Q_j$.

(b) $Q_{t+1} := \{ q | (q, a) \in Q_j \}; Q_i := Q_i \setminus Q_{t+1}$;

$t := t + 1$.

end.

3. (* Denote $[q]$ the equivalence class of state q , and $\{Q_i\}$ the set of all equivalence classes. *)

$q'_0 := [q_0]$.

' ($[q], a) := [(q, a)]$ for all $q \in Q, a \in \Sigma$.

Standard implementation: $O(kn^2)$, where $n = |Q|$ and $k = |\Sigma|$

Modification of the body of the while loop:

1. Choose such an i, a , and choose $j_1, j_2 < t$ with $(q_i, a) \in Q_{j_1}$ and $(q_i, a) \in Q_{j_2}$.

2. If $| \{ q | (q, a) \in Q_{j_1} \} | > | \{ q | (q, a) \in Q_{j_2} \} |$

```

then  $Q_{t+1} := \{q \mid Q_i \mid (q,a) Q_j\}$  else  $Q_{t+1} := \{q\}$ 
 $Q_i \mid (q,a) Q_j$  } fl;
 $Q_i := Q_i \setminus Q_{t+1}$ ;
 $t := t + 1$ .

```

(i.e. put smallest set in $t + 1$)

Note: $|Q_{t+1}| \leq \frac{1}{2}|Q_i|$. Therefore, for all $q \in Q$, the name of the class which contains a given state q changes at most $\log(n)$ times.

Goal: Develop an implementation such that all computations can be assigned to transitions containing a state for which the name of the corresponding class is changed.

Suitable data structures achieve an $O(kn \log n)$ implementation.

State Minimization:

Incompletely Specified Machines

Statement of the problem: given an incompletely specified machine M , find a machine M' such that:

- on any input sequence, M' produces the same outputs as M , whenever M is specified.
- there does not exist a machine M'' with fewer states than M' which has the same property

Machine M:

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, -	s3, 0
s3	s3, 1	s2, 0

Attempt to reduce this case to usual state minimization of completely specified machines.

Brute Force Method: Force the don't cares to all their possible values and choose the smallest of the completely specified machines so obtained.

In this example, it means to state minimize two completely specified machines obtained from M , by setting the don't care to either 0 and 1.

Suppose that the - is set to be a 0.

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, 0	s3, 0
s3	s3, 1	s2, 0

States s_1 and s_2 are equivalent if s_3 and s_2 are equivalent, but s_3 and s_2 assert different outputs under input 0, so s_1 and s_2 are not equivalent.

States s_1 and s_3 are not equivalent either.

So this completely specified machine cannot be reduced further (3 states is the minimum).

Suppose that the - is set to be a 1.

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, -	s3, 0
s3	s3, 1	s2, 0

States s1 is incompatible with both s2 and s3.

States s3 and s2 are equivalent.

So number of states is reduced from 3 to 2.

Machine M''_{red} :

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

Can this always be done?

Machine M:

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, -	s1, 0
s3	s1, 1	s2, 0

Machine M₂:

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, 0	s1, 0
s3	s1, 1	s2, 0

Machine M₃:

PS	NS, z	
	x=0	x=1
s1	s3, 0	s2, 0
s2	s2, 1	s1, 0
s3	s1, 1	s2, 0

Machine M₂ and M₃ are formed by filling in the unspecified entry in M with 0 and 1, respectively.

Both machines M_2 and M_3 cannot be reduced.

Conclusion?: M cannot be minimized further!

But is it a correct conclusion?

Note: that we want to merge two states when, for any input sequence, they generate the same output sequence, but only where both outputs are specified.

Definition: A set of states is compatible if they agree on the outputs where they are all specified.

Machine M'' :

PS	NS, z	
	x=0	x=1
s1	s3,0	s2,0
s2	s2,-	s1,0
s3	s1,1	s2,0

In this case we have two compatible sets: $A = (s1, s2)$ and $B = (s3, s2)$. A reduced machine M_{red} can be built as follows.

Machine M_{red}

PS	NS, z			
	I1	I2	I3	I4
A	A,1	A,0		
B	B,0	A,0		
s1	s3,0	s1,-	-	-
s2	s6,-	s2,0	s1,-	-
s3	-,1	-,-	s4,0	-
s4	s1,0	-,-	-	s5,1
s5	-,-	s5,-	s2,1	s1,1
s6	-,-	s2,1	s6,-	s4,1

A set of compatibles that cover all states is: $(s3s6), (s4s6), (s1s6), (s4s5), (s2s5)$.

But $(s3s6)$ requires $(s4s6)$,

$(s4s6)$ requires $(s4s5)$, $(s4s5)$ requires $(s1s5)$,

$(s1s6)$ requires $(s1s2)$, $(s1s2)$ requires $(s3s6)$,

$(s2s5)$ requires $(s1s2)$.

So, this selection of compatibles requires too many other compatibles...

PS	NS, z			
	I1	I2	I3	I4
s1	s3,0	s1,-	-	-
s2	s6,-	s2,0	s1,-	-
s3	-,1	-,-	s4,0	-
s4	s1,0	-,-	-	s5,1
s5	-,-	s5,-	s2,1	s1,1
s6	-,-	s2,1	s6,-	s4,1

Another set of compatibles that covers all states is $(s1s2s5), (s3s6), (s4s5)$. But $(s1s2s5)$ requires $(s3s6)$ $(s3s6)$ requires $(s4s6)$ $(s4s6)$ requires $(s4s5)$ $(s4s5)$ requires $(s1s5)$.

So must select also $(s4s6)$ and $(s1s5)$.

Selection of minimum set is a binate covering problem

When a next state is unspecified, the future behavior of the machine is unpredictable. This suggests the definition of admissible input sequence.

Definition. An input sequence is admissible, for a starting state of a machine if no unspecified next state is encountered, except possibly at the final step.

Definition. State s_i of machine M_1 is said to cover, or contain, state s_j of M_2 provided

1. every input sequence admissible to s_j is also admissible to s_i , and
2. its application to both M_1 and M_2 (initially in s_i and s_j , respectively) results in identical output sequences whenever the outputs of M_2 are specified.

Definition. Machine M_1 is said to cover machine M_2 if for every state s_j in M_2 , there is a corresponding state s_i in M_1 such that s_i covers s_j .

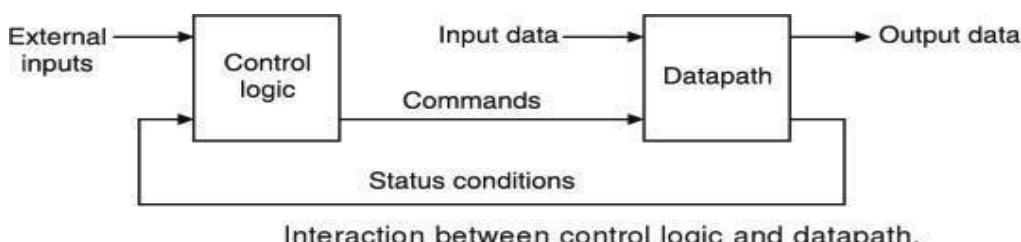
Algorithmic State Machines:

The binary information stored in the digital system can be classified as either data or control information.

The data information is manipulated by performing arithmetic, logic, shift and other data processing tasks.

The control information provides the command signals that controls the various operations on the data in order to accomplish the desired data processing task.

Design a digital system we have to design two subsystems data path subsystem and control subsystem.



ASM CHART:

A special flow chart that has been developed specifically to define digital hardware algorithms is called ASM chart.

A hardware algorithm is a step by step procedure to implement the desire task.

Difference b/n conventional flow chart and ASM chart:

conventional flow chart describes the sequence of procedural steps and decision paths for an algorithm without concern for their time relationship

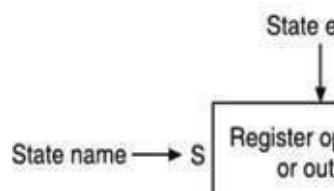
An ASM chart describes the sequence of events as well as the timing relationship b/n the states of sequential controller and the events that occur while going from one state to the next

1. State box: A state of a clocked sequential circuit is represented by a rectangle called *state box*. It is equivalent to a node in the state diagram or a row in the state table. The name of the state is written to the left of the box. The binary code assigned to the state is indicated outside on the top right-side of the box. A list of unconditional outputs if any associated with the state are written within the box.

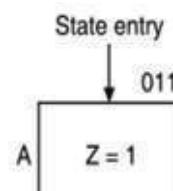
2. Decision box: The decision box or condition box is represented by a diamond-shaped symbol with one input and two or more output paths. The output branches are true and false branches. The decision box describes the effect of an input on the control subsystem. A Boolean variable or input or expression written inside the diamond indicates a condition which is evaluated to determine which branch to take.

ASM consists of

1. State box
2. Decision box
3. Conditional box State box

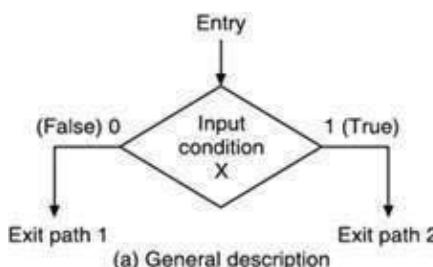


(a) General description

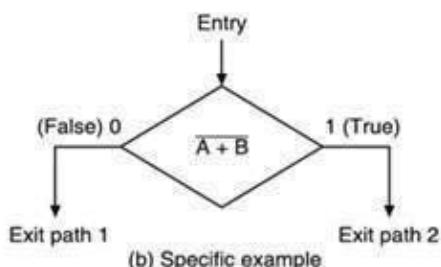


(b) Specific example

Decision box



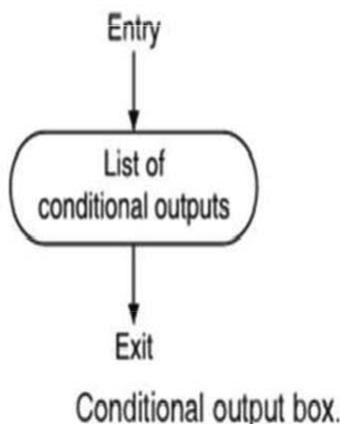
(a) General description



(b) Specific example

Decision box.

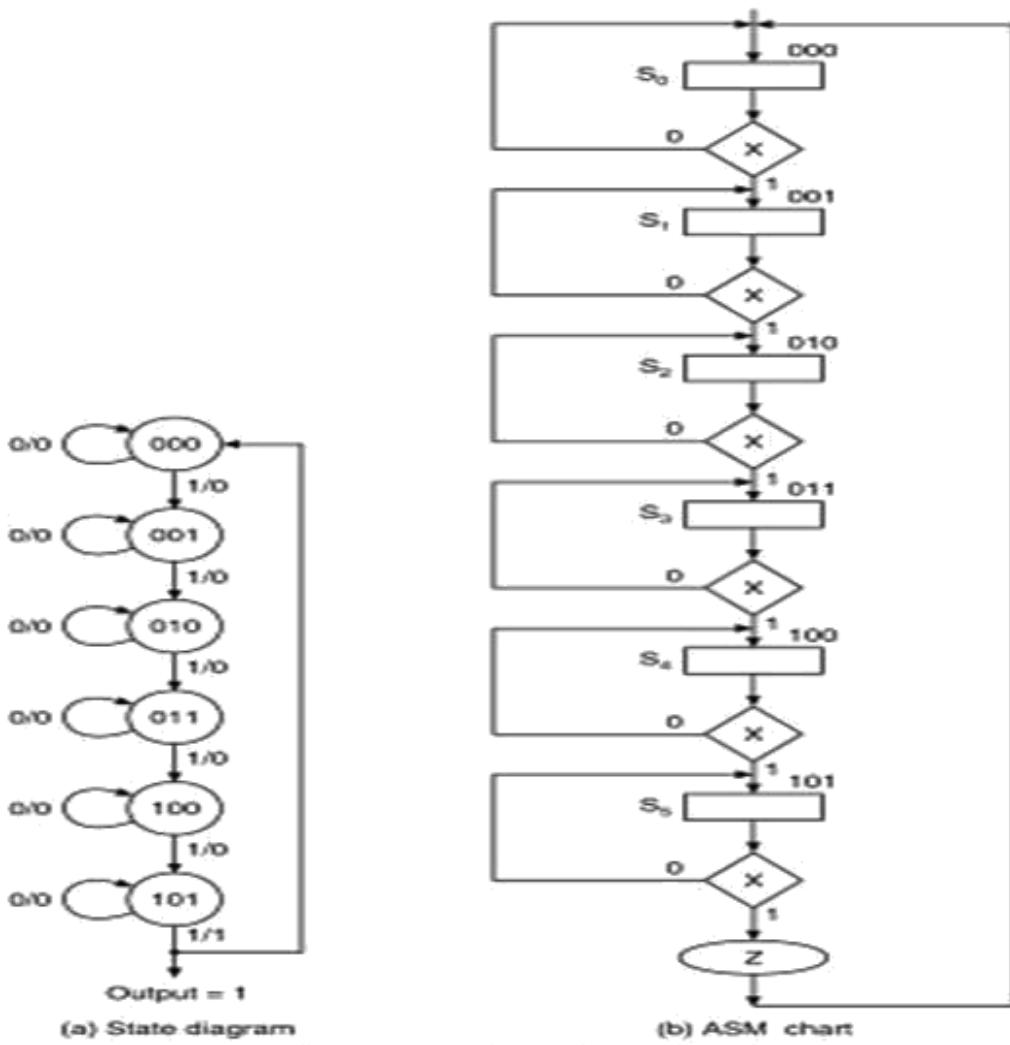
3. Conditional output box: The conditional output box is represented by a rectangle with rounded corners or by an oval with one input line and one output line. The outputs that depend on both the state of the system and the inputs are indicated inside the box.



Conditional output box.

SALIENT FEATURES OF ASM CHARTS

1. An ASM chart describes the sequence of events as well as the timing relationship between the states of a sequential controller and the events that occur while going from one state to the next.
2. An ASM chart contains one or more interconnected ASM blocks.
3. Each ASM block contains exactly one state box together with the decision boxes and conditional output boxes associated with that state.
4. Every block in an ASM chart specifies the operations that are to be performed during one common clock pulse.
5. An ASM block has exactly one entrance path and one or more exit paths represented by the structure of the decision boxes.
6. A path through an ASM block from entrance to exit is referred to as a link path.
7. The operations specified within the state and conditional output boxes in the block are performed in the datapath subsystem.
8. Internal feedback within an ASM block is not permitted. Even so, following a decision box or conditional output boxes, the machine may reenter the same state.
9. Each block in the ASM chart describes the state of the system during one clock pulse interval. When a digital system enters the state associated with a given ASM block, the outputs indicated within the state box become true. The conditions associated with the decision boxes are evaluated to determine which path or paths to be followed to enter the next ASM block.

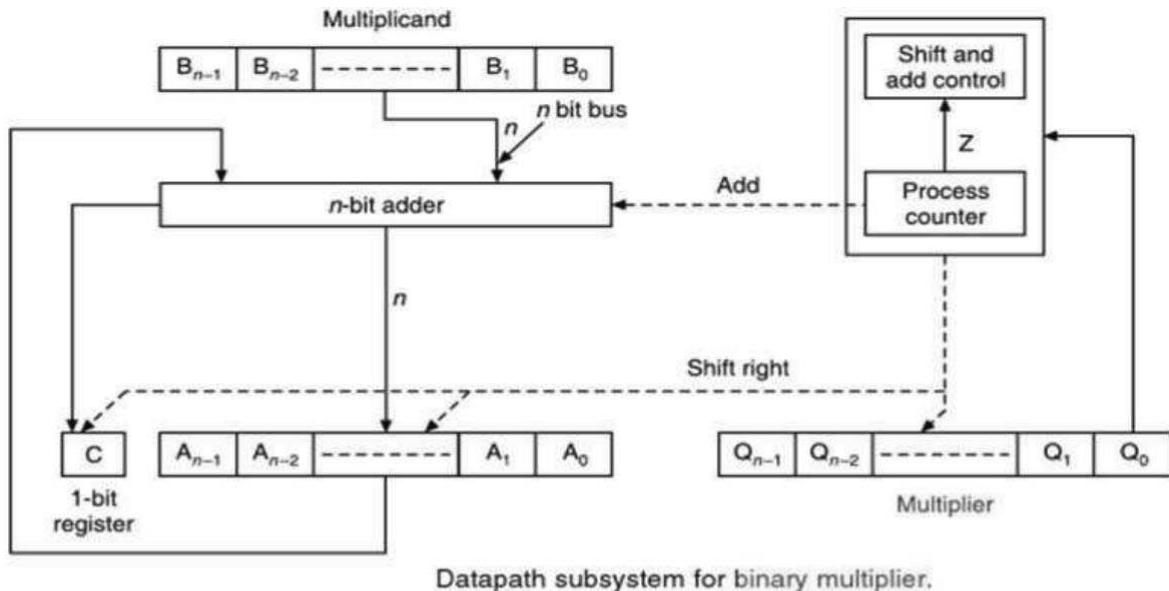


State diagram and ASM chart for mod-6 counter.

BINARY MULTIPLIER

$$\begin{array}{r}
 1101 \\
 1010 \\
 \hline
 0000 \\
 1101 \\
 0000 \\
 \hline
 1101 \\
 \hline
 10000010
 \end{array}
 \quad \begin{array}{l}
 \leftarrow 13_{10} \dots \text{Multiplicand} \\
 \leftarrow 10_{10} \dots \text{Multiplier} \\
 \leftarrow \text{Partial product 1} \\
 \leftarrow \text{Partial product 2} \\
 \leftarrow \text{Partial product 3} \\
 \leftarrow \text{Partial product 4} \\
 \leftarrow 130_{10} \dots \text{Product}
 \end{array}$$

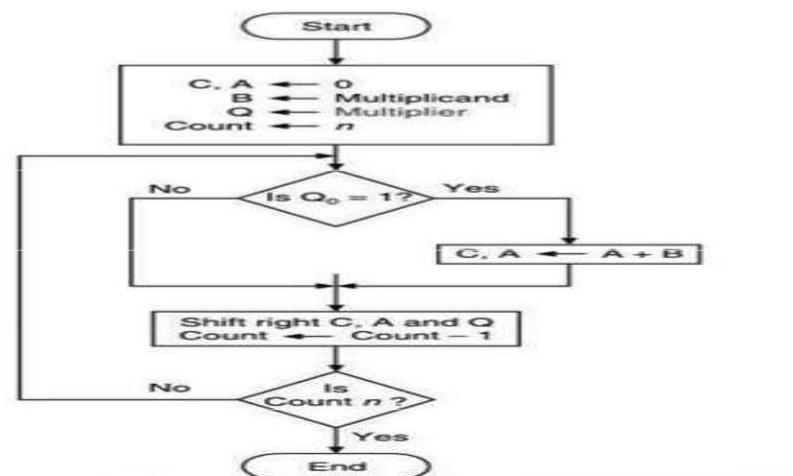
Data path subsystem for binary multiplier



Datapath subsystem for binary multiplier.

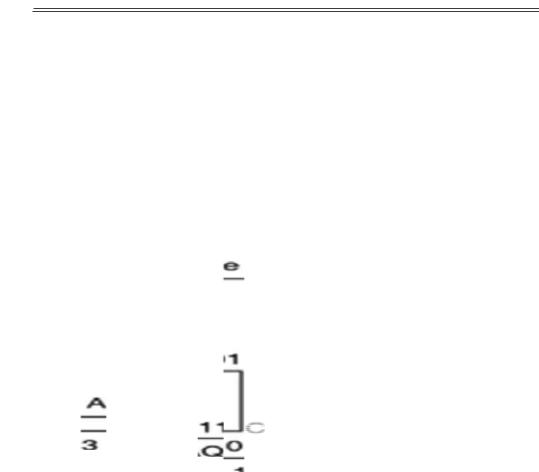
Multiplication Operation Steps

1. Bit 0 of multiplier operand (Q_0 of Q register) is checked.
2. If bit 0 (Q_0) is one then multiplicand and partial product are added and all bits of C, A and Q registers are shifted to the right one bit, so that the C bit goes into A_{n-1} , A_0 goes into Q_{n-1} , and Q_0 is lost. If bit 0 (Q_0) is 0, then no addition is performed, only shift operation is carried out.
3. Steps 1 and 2 are repeated n times to get the desired result in the A and Q registers.



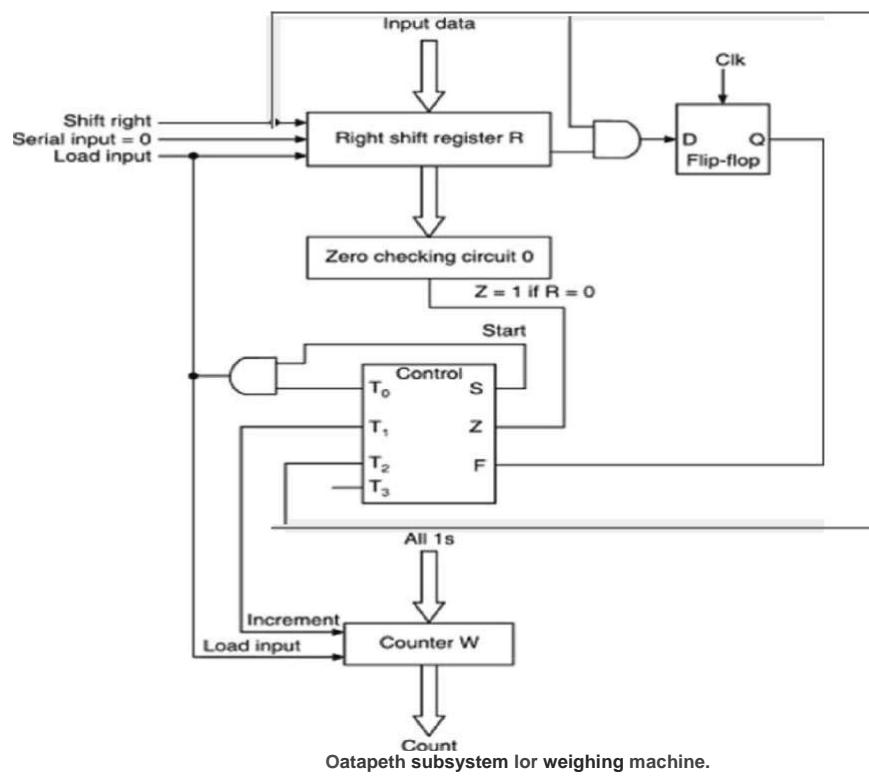
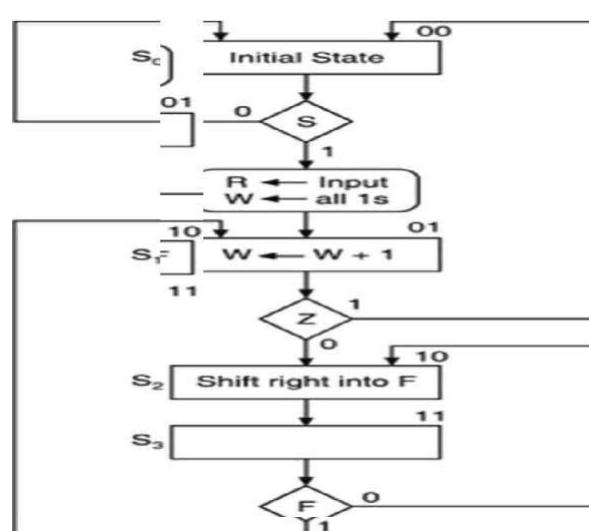
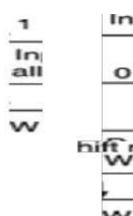
B	C	A	Q	Components	Count P
1101	0	0 000	1010	$B \leftarrow \text{Multiplicand}$ $Q \leftarrow \text{Multiplier}$ $A \leftarrow 0, C \leftarrow 0, P \leftarrow n$	
1101	0	0 000	1010		1
	0	0000	0	101	$C A Q \text{ shifted right}$
				0 1 0 i 21	
J101	0	0110	1010	$Q_0 = 1, A \leftarrow A + B$ $C A P \leftarrow \dots, P \leftarrow \text{J right}$ "0. $C A Q \text{ shifted right}$	001 (1)
b	0 0	1 i u	i b i	"0. $C A Q \text{ shifted right}$	
1 1 0 1 1	0 0	0 0 0	1 0 1	$P \leftarrow P - 1$	
0	1 0 0	0 0	0 1 0	$C A @ .shifted right$	

flowchart for multiplication in a computer.



ASM FOR WEIGHING MACHINE

In the original form for tabular minimization of Boolean expressions, we have to arrange bits in increasing order of their weights. This is only one of the many situations when we have to examine the 1's of a given binary word. The weight of a binary number is defined as the number of 1's in it. The weight of the binary number 1101 is 3.

put
1s

6r / S . Ln*ti atJy lite weighing stack ine i s in state Sq. one weighing process starts wific suu-t (S) signal becomes 1 . Whi le in state fi if *> i s 1 , the clock pulse cauws three jobs to be done sinTu ltaneously."

1. Bi n ay n u m ber is loader4 i nEo xug isCer R.
2. W regJ ster is set to all G s.
3. The xtachine is transFered to sta re S , .

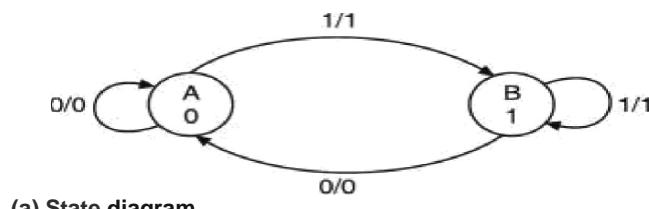
State S₁: While in state S₁, the clock pulse causes two jobs to be done simultaneously:
· CCountcr W in income rated by' 1{in thJz First councJ, all I s b+>c:Om &J1 US).

2. IT Z i s 0. the in ash ine go• s la Lhc scale fi : i l Z i s l . tMe machinc go•es to state fi .

crore \$ 2• In this state. regiMer R ix lifted righthyl hitin thaE L SC y <ws intt F and M \$ B is
1naded with 0

fiim Sy: In iii is state, rhe value of F is chicken. Jf iz is 0, the machine is rransf'ered to
the state S,, othew'ise the machine i3 transferred to state S,. Thus. when F - I. W is
incremented.

All the operations occur in coincidence with the clock pulse while in the corresponding state.
Also nokcc <w the rugi star k should eventually contain all 0s when the last I is stiir in into .

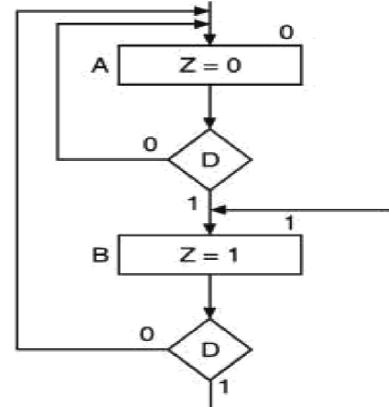


(a) State diagram

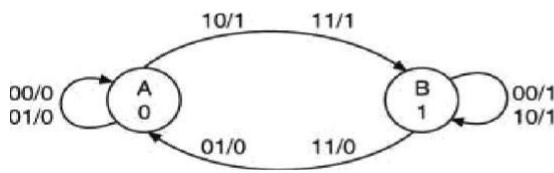
PS	NS, O/P	
	Input 0	
D - D	D - 1	
A	A, D	B, 1
B	A, D	B, 1

(D)

State table



(c) ASM chart



(a)

State diagram

PS	NS, OGP			
	Input J-K	D1	10	11
OD	OD	D1	10	11
A	A, 0	A, 0	B, 1	B, 1
B	B, 1	A, 0	B, 1	A, 0

(b) state table

(c) ADM chart

