UNIT V

SYNCHRONOUS AND ASYNCHRONOUS SEQUENTIAL CIRCUITS

5.1 Introduction

We have already introduced to synchronous sequential circuits. These circuits are further classified depending on the timing of their signals: Synchronous sequential circuits and Asynchronous Sequential Circuits. In synchronous sequential circuits signals can affect the memory elements only at discrete instants of time. In asynchronous sequential circuits change in input signals can affect memory element at any instant of time.

Sl.No	Synchronous sequential circuits	Asynchronous sequential circuits
1.	In synchronous circuits memory elements are clocked flipflops	In asynchronous circuits memory elements are either unclocked flipflops or time delay elements.
2.	The change in input signals can affect memory element upon activation of clock signal	The change in input signals can affect memory element at any instant of time
3.	The maximum operating speed of clock depends on time delays involved	Because of absence of clock asynchronous circuits can operate faster than synchronous circuits
4.	Easier to design	More difficult to design

Table:5.1 Comparison between synchronous sequential circuits and asynchronous sequential circuits

5.2 Clocked sequential circuits

In synchronous or clocked sequential circuits clocked flipflops are used as memory elements which change their individual states in synchronism with the periodic clock signal.
The change in states of flipflop and change in state of the entire circuit occur at the transition of the clock signal.
The states of the output of the flipflop in the sequential circuit give the state of the sequential circuit.

Present state

The status of all state variables at some time t, before the next clock edge represent condition called present state.

Next state

The status of all state variables at some time t+1 represent a condition called next state.
The synchronous or clocked sequential circuits are represented by two models.

Moore model: The output depends only on the present state of the flipflops.
Mealy model: The outputs depends on both the present state of the flipflops and on the inputs

5.2.1 Moore model

When the output of the sequential circuit depends only on the present state of the flipflop the sequential circuit is referred to as Moore model. Let us see one example of Moore model.

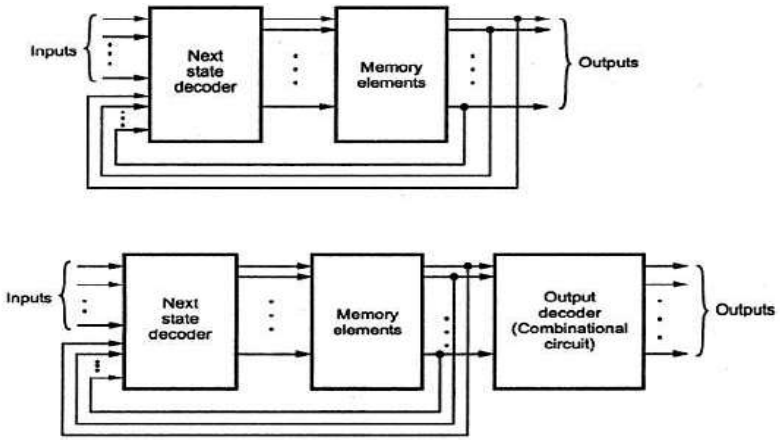


Figure: 5.2 Moore circuit model with an output decoder

5.2.2 Mealy model:

When the output of the sequential circuit depends on both the present state of the flipflops and on the inputs, the sequential circuit is referred to as mealy model. Figure 5.2.shows the sample mealy model.

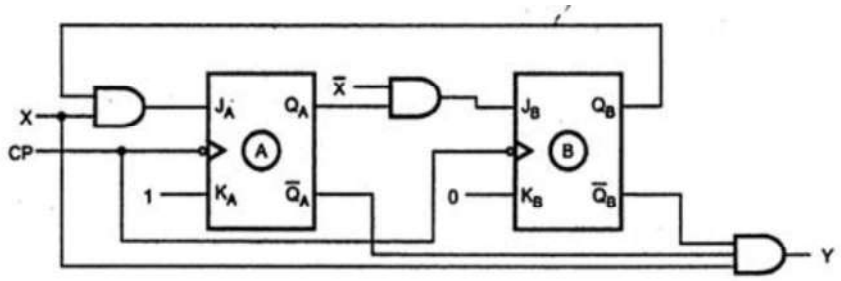


Figure.5.3 Example of mealy model

Looking at Figure: 5.3 we can easily realize that, changes in the input within the clock pluses cannot affect the state of the flip-flop. However, they can affect the output of the circuit. Due to this, if the input variations are not synchronized with the clock, the derived output also not be synchronized with the clock and we get false output (as it is a synchronous sequential

network). The false outputs can be eliminated by allowing input to change only at the active transition of the clock (in our example HIGH –to-LOW).In general form the Mealy model can be represented with in its block schematic as shown as in figure5.3

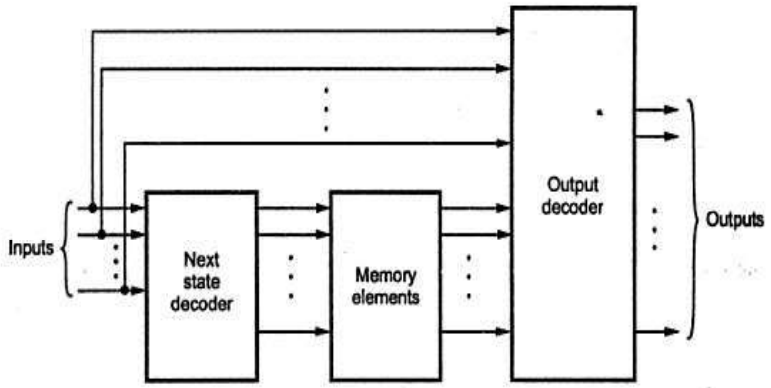


Figure.5.4 Mealy circuit model

5.2.3 Moore Vs Mealy Circuit Models

Sr. No.	Moore model	Mealy model
1.	Its output is a function of present state only.	Its output is a function of present state as well as present input.
2.	Input changes does not affect the output.	Input changes may affect the output of the circuit.
3.	Moore model requires more number of states for implementing same function.	It requires less number of states for implementing same function.

Table. 5.5

5.2.4 Conversion of Models

5.2.4.1Rules to convert Mealy to Moore Model

1. If all the transitions in a Mealy model to a particular state are associated with only one type of output(either 0 or 1) then in corresponding Moore model that output becomes state output. This is illustrated in figure. Here T1,T2 and T3 are the paths leading to state a and corresponding outputs are all zeros. Thus we can say state a output is 0.

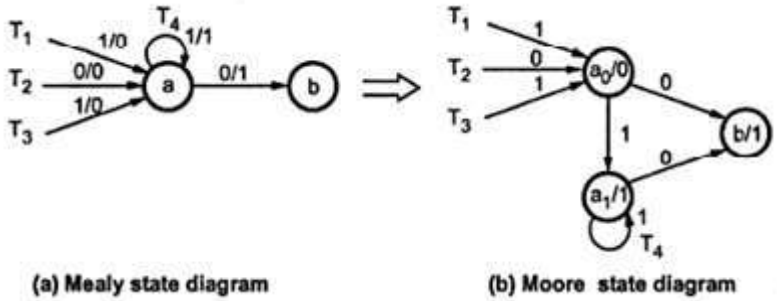


Figure. 5.6 Rule1: Mealy to Moore conversion

When input is 1, state a leads to state b and there is no other input path. Thus we can say state b output is 0(the corresponding output of input path).

2.If the outputs of all transitions in a Mealy model to a particular state are not same we need to insert intermediate state variables. This is illustrated in figure.5.2.5

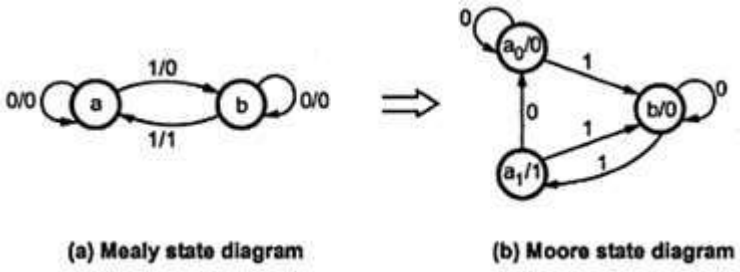


Figure:5.7 Rule 2: Mealy to Moore conversion

5.2.4.2Rules to convert Moore to Mealy Model

1. If the state of transition from two different states of the same input leads to common state then one state can be eliminated. This is illustrated in figure: 5.2.6

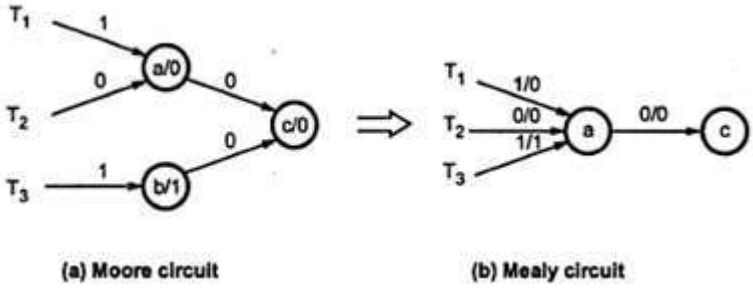


Figure: 5.8

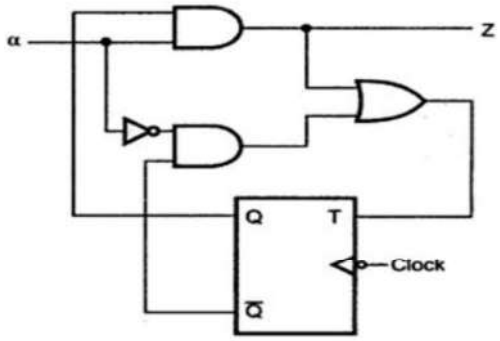
Here, states **a** and **b** leads to common state c when input is 0. Therefore, we can eliminate state**a** or**b.**

5.3 Analysis of clocked sequential circuits

The behaviour of a sequential network is determined from the inputs, the outputs, and the states of its flipflops. Both the outputs and the next state are function of the inputs and the

present state (in case of Moore circuits the outputs are function of only present state). The analysis of sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs and internal states. The success of analysis or design of sequential network depends largely on the aids and systematic techniques such as transition table, state table, state diagrams and state equations used in these processes.

Consider the sequential circuit to be analyzed as shown in figure



Let us see the steps to analyze the given synchronous sequential circuit

1. Determine the flip-flop input equations and the output equations from the sequential circuit.

$$Z = \alpha Q$$

$$T = \alpha Q + \bar{\alpha}\bar{Q}$$

2. Derive the transition equation.

The transition equation for T flip-flop is

$$Q^+ = T \oplus Q$$

$$Q^+ = (\alpha Q + \bar{\alpha}\bar{Q}) \oplus Q$$

3. Plot the next step map for each flip-flop

For Q^+

Q \ α	0	1
0	1	0
1	1	0

$$Q^+ = (\alpha Q + \bar{\alpha}\bar{Q}) \oplus Q$$

4.Plot the transition table

Present State	Next State		Output	
	α = 0	α = 1	α = 0	α = 1
Q	Q^*	Q^+	Z	Z
0	1	0	0	0
1	1	0	0	1

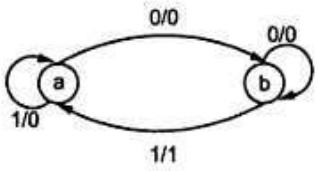
Note: If circuit consist of more than one flipflop we have to combine the map of flipflop to derive the transition table.

5.Draw the state table

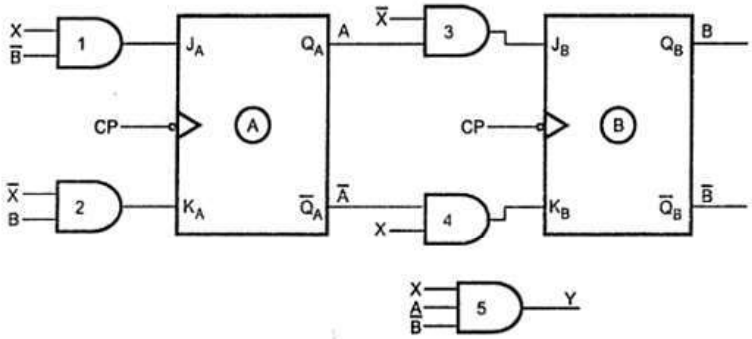
Here new symbols to binary codes are assigned. They are a=0, b=1

Present State	Next State		Output	
	α = 0	α = 1	α = 0	α = 1
a(0)	b	a	0	0
b(1)	b	a	0	1

6. Draw state diagram



Example : Construct the transition table, state table and state diagram for the Mealy sequential circuit given in figure



Solution : 1. Determine the flip-flop input equations and the output equations from the sequential circuit.

$$Y = XA\bar{B}$$

$$J_A = X\bar{B} \qquad K_A = \bar{X}B$$

$$J_B = \bar{X}A \qquad K_B = X\bar{A}$$

2.Derive the transition equations

The transition equations for JK flip-flops can be derived from the characteristic equation of JK flip-flop as follows:

We know that for JK flip-flop,

$$Q^+ = J\bar{Q} + \bar{K}Q$$

$$\therefore A^+ = Q_A^+ = J_A\bar{Q}_A + \bar{K}_A Q_A$$

$$= X\bar{B}\bar{Q}_A + \overline{\bar{X}B}\,Q_A$$

$$= X\bar{B}\bar{A} + (X + \bar{B})A$$

and

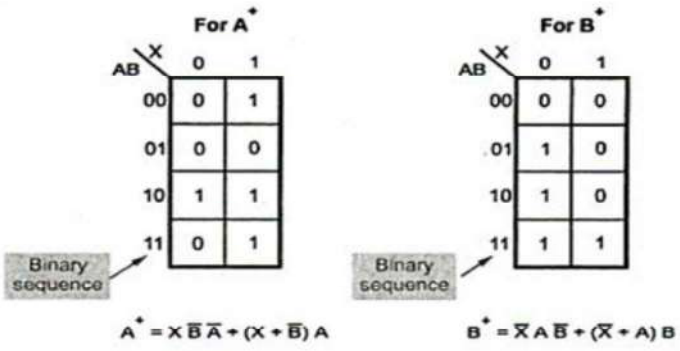
$$B^+ = Q_B^+ = J_B\bar{Q}_B + \bar{K}_B Q_B$$

$$= \bar{X}A\overline{Q_B} + \overline{X\bar{A}}\,Q_B$$

$$= \bar{X}A\bar{B} + (\bar{X} + A)B$$

3.Plot a next-state maps for each flip-flop

The next-state maps are



4.Plot the transition table

The transition table can be formed by combining the above two maps.

Present State A B	Next state		Output	
	X = 0	X = 1	$Y = XA\overline{B}$	
	$A^+ B^+$	$A^+ B^+$	X = 0	X = 1
0 0	0 0	1 0	0	0
0 1	0 1	0 0	0	0
1 0	1 1	1 0	0	1
1 1	0 1	1 1	0	0

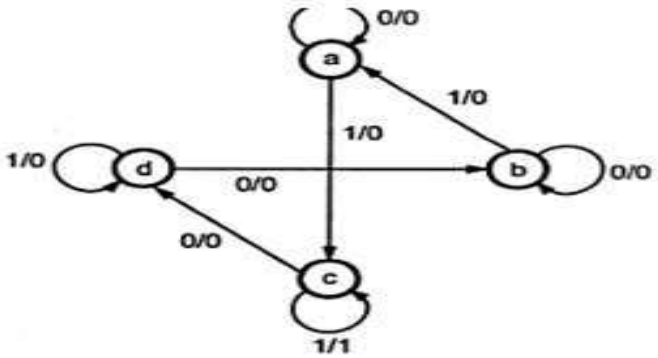
5.Draw the state table

By assigning a = 00, b = 01, c = 10, d = 11 we can write state table from the transition table as shown below.

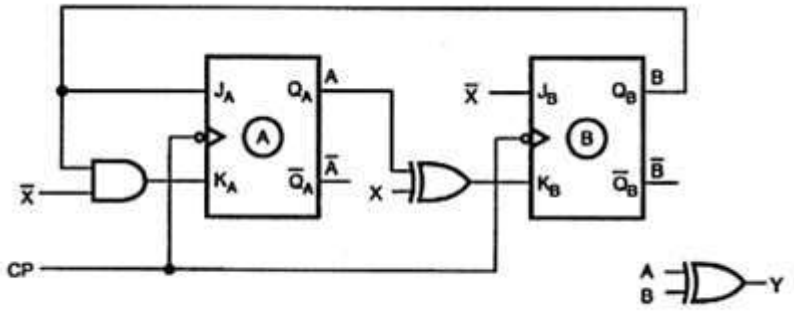
Present State A B	Next state		Output	
	X = 0	X = 1	Y	
	$A^+ B^+$	$A^+ B^+$	X = 0	X = 1
a (00)	a	c	0	0
b (01)	b	a	0	0
c (10)	d	c	0	1
d (11)	b	d	0	0

6.Draw the state diagram

From the state table we can draw the state diagram



Excersice : Derive the transition table, state table and state diagram for Moore sequential circuit shown infigure



5.4 Design of Clocked Sequential Circuits

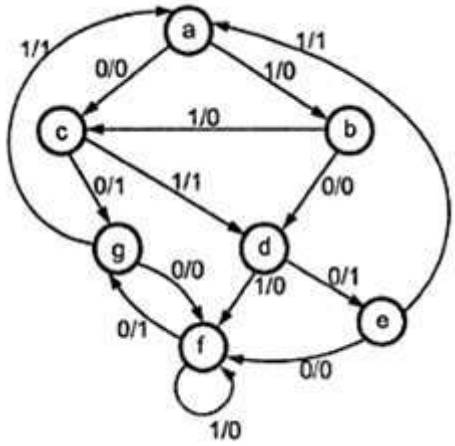
Here, we will summarize the recommended steps for the design of a clocked sequential circuits.

1. It is necessary to first obtain the state table for the given circuit information such a state diagram, a timing-diagram, or other pertinent information.
2. The number of states may be reduced by state reduction technique if the sequential circuit can be categorized by input-output relationships independent of the number of states.
3. Assign binary values to each state in the state table.
4. Determine the number of flip-flops needed and assign a lettert symbol to each.
5. Choose the type of flip-flop to be used.
6. From the state table, derive the circuit exctation and output tables.
7. Using the K-map or any other simplification method, derive the circuit output functions and flip-flop input functions.
8. Draw the logic diagram.

5.5 State Reduction

The state reduction technique basically avoids the introduction of redundant states. The reduction in redundant states reduce the number of required flipflops and logic gates, reducing the cost of the final circuit. The two states are said to be redundant or equivalent, if every possible set of inputs generate exactly same output and same next state. When two states are equivalent, one of them can be removed without altering the input-output relationship.

Example : Desigh a clocked sequential circuit for state diagram shown in the Fig 9.44



Solution : The state table for the sequential circuit will be as shown in Table 9.18 (a).

Present state	Next state		Output (Z)	
	X = 0	X = 1	X = 0	X = 1
a	c	b	0	0
b	(d) f	c	0	0
c	(g) e	(d) f	1	1
d	e	f	1	0
e	f	a	0	1
f	(g) e	f	1	0
g	f	a	0	1

As states e and g are eqivalent, we eliminate the state g as shown in the sate table. After replacing g by e, we can notice that the states d and f are eqivalent. Thus one of them say f can be eliminated. Then the reduced state table is as shown in Table 9.18(b)

Present state	Next state		Output (Z)	
	X = 0	X = 1	X = 0	X = 1
a	c	b	0	0
b	f	c	0	0
c	e	f	1	1
e	f	a	0	1
f	e	f	1	0

No each state is assigned with binary values. Since there are five states, number of flip-flops required is 3 and 3-bit binary numbers are assigned to the states as shown below.

a = 000, b = 001, c = 010, e = 011, f = 100

If D flip-flops are used in design, the excitation Table 9.18 (c) is as given below.

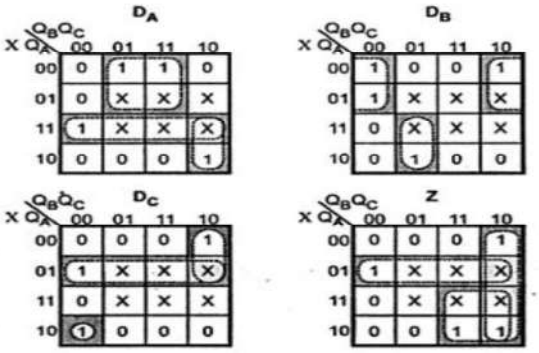
Input	Present state			Next state			Output
X	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	Z
0	0	0	0	0	1	0	0
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	1	0	0	0
0	1	0	0	0	1	1	1
1	0	0	0	0	0	1	0
1	0	0	1	0	1	0	0
1	0	1	0	1	0	0	1
1	0	1	1	0	0	0	1
1	1	0	0	1	0	0	0

The flip-flop inputs DA, DB and DC are not included in the excitation table as they equal to the next state.

$$D_A = Q_{A+1}, \quad D_B = Q_{B+1} \quad \text{and} \quad D_C = Q_{C+1}$$

Mapping for D_A, D_B, D_C and Z (Assuming X, don't care, care condition for unused states).

K-map simplification



Hence the design equations are :

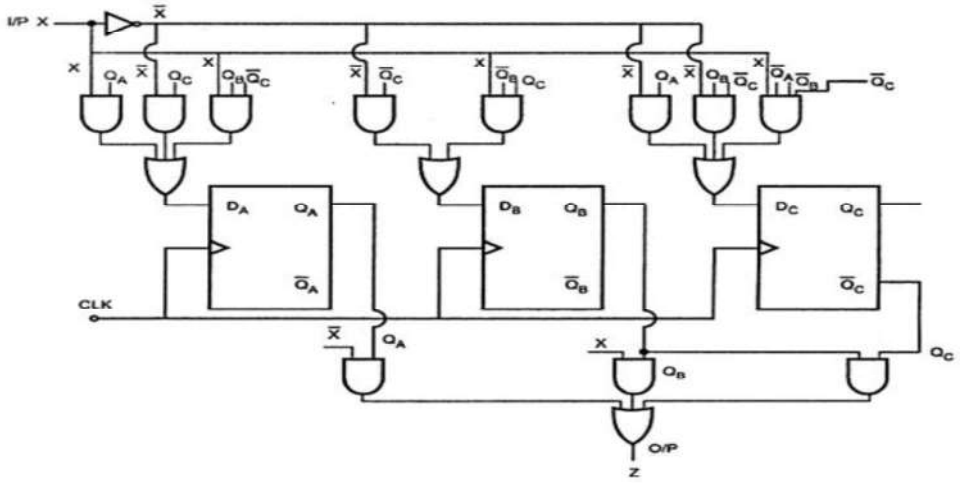
$$D_A = X Q_A + \overline{X} Q_C + X Q_B \overline{Q}_C$$

$$D_B = \overline{X}\,\overline{Q}_C + X \overline{Q}_B Q_C$$

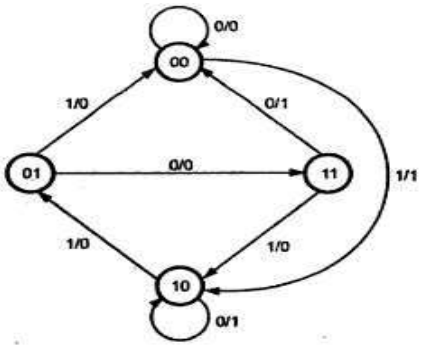
$$D_C = \overline{X} Q_A + \overline{X} Q_B \overline{Q}_C + X \overline{Q}_A \overline{Q}_B \overline{Q}_C$$

$$Z = \overline{X} Q_A + X Q_B + Q_B \overline{Q}_C$$

The circit diagram is as shown in Fig. 9.44 (b)



Example: A sequential circuit has one input and one output. The state diagram is shown in figure. Design the sequential circuit with a)D flipflops b) T flipflops c) RS flipflops d) JK flipflops



Solution: The state table for the state diagram

Present state		Next state		Output	
		X = 0	X = 1	X = 0	X = 1
A	B	AB	AB	Y	Y
0	0	0 0	1 0	0	1
0	1	1 1	0 0	0	0
1	0	1 0	0 1	1	0
1	1	0 0	1 0	1	0

As seen from the state table there is no equivalent states. Therefore, no reduction in the state diagram. The state table shows that circuit goes through four states, therefore we require

flip-flops (number of states = 2^m, where m = number of flip-flops). Since two flip-flops are required first is denoted as **A** and second is denoted as **B**.

i) Design using D flip-flops

As mentioned earlier, for D flip-flops next states are nothing but the new present states. Thus, we can directly use next states to determine the flip-flop input with the help of K-map simplification.

K-map simplification

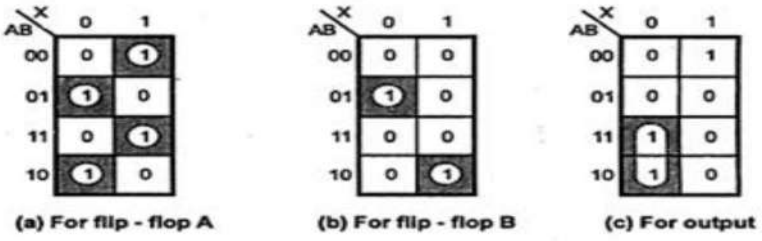


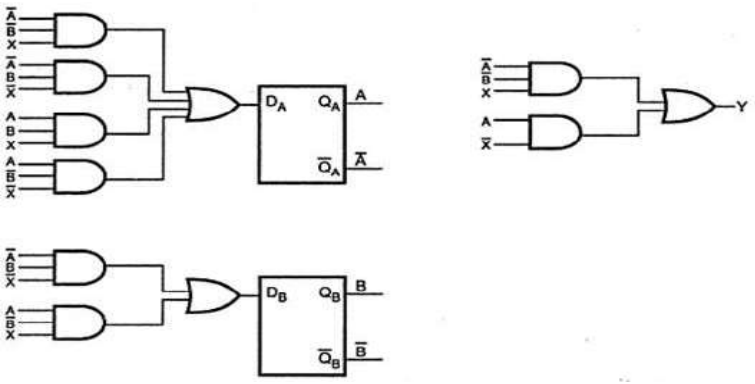
Fig. 9.15

$$D_A = \bar{A}\bar{B}X + \bar{A}B\bar{X} + ABX + A\bar{B}\bar{X} \quad \text{and}$$

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

$$Y = \bar{A}\bar{B}X + A\bar{X}$$

With these flip-flop input functions and circuit output function we can draw the logic diagram as follows.



ii) Design using T flip-flops

a) Write the excitation table for T flipflop

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

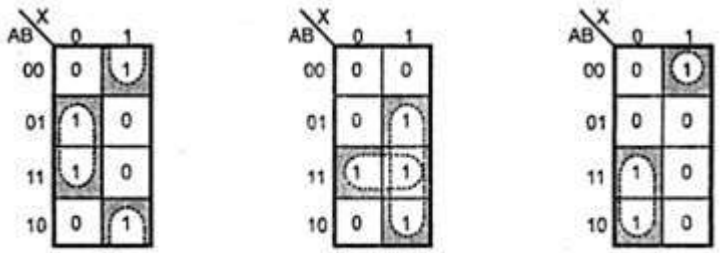
b) Determine the excitation table for the given circuit

Present state		Input	Next state		Flip-flop inputs		Output
A	B	X	A	B	T_A	T_B	Y
0	0	0	0	0	0	0	0
0	0	1	1	0	1	0	1
0	1	0	1	1	1	0	0
0	1	1	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	1	1	1	0	0	1	0

In the first row of circuit excitation table shows that there is no change in the state for both flip-flops. The transition from 0 -> 0 for T flip-flop requires input T to be at logic 0. The second row shows the flip-flop A has transition 0 -> 1. It requires the input TA to be at logic 1. Similarly, we can find inputs for each flip-flop for each row in the table by referring present state, next state and excitation table.

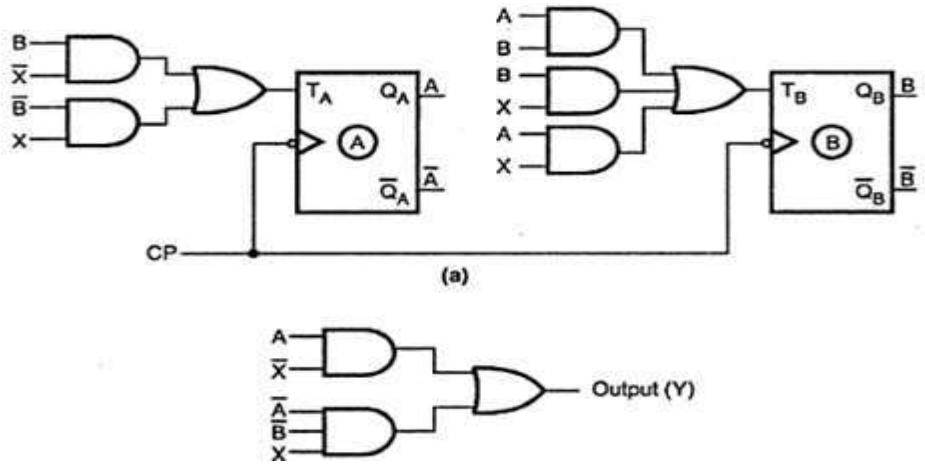
Let us use K-map simplification to determine the flip-flop input functions and circuit output functions.

K-map simplification



(a) For flip-flop A (b) For flip-flop B (c) For output

With these flip-flop input functions and circuit output function we can draw the logic diagram as follows.



iii) Design using RS flip-flops

a) Write the excitation table for RS flipflop

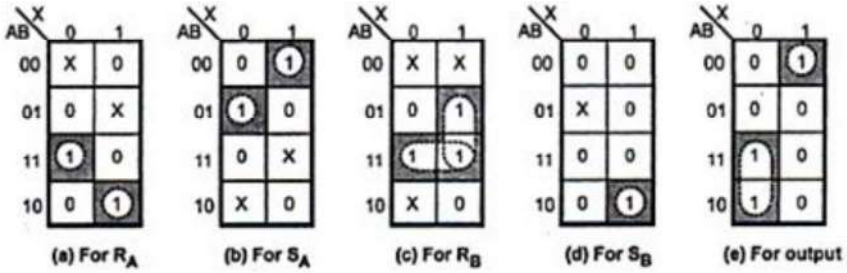
Q_n	Q_{n+1}	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

b) Determine the excitation table for the given circuit

Present state		Input	Next state		Flip-flop inputs				Output
A	B	X	A	B	R_A	S_A	R_B	S_B	Y
0	0	0	0	0	X	0	X	0	0
0	0	1	1	0	0	1	X	0	1
0	1	0	1	1	0	1	0	X	0
0	1	1	0	0	X	0	1	0	0
1	0	0	1	0	0	X	X	0	1
1	0	1	0	1	1	0	0	1	0
1	1	0	0	0	1	0	1	0	1
1	1	1	1	0	0	X	1	0	0

The first row of circuit excitation table shows that there is no change in the state for both flip-flops. The transition from 0 -> 0 for RS flip-flop requires inputs R and S to be X and 0, respectively. Similarly, we can determine the inputs for each flip-flop for each row in the table by referring present state, next state and excitation table. Let us use K-map simplification to determine the flip-flop input functions and circuit output functions.

K-map simplification



Therefore input function for

$$R_A = AB\bar{X} + A\bar{B}X$$

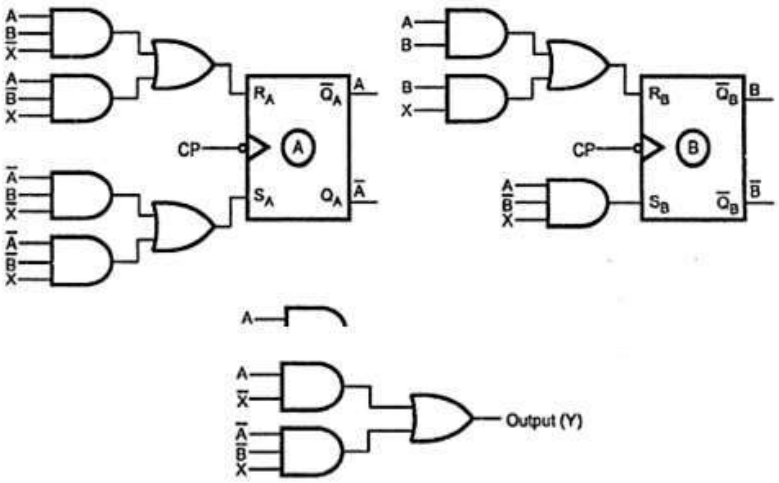
$$S_A = \bar{A}B\bar{X} + \bar{A}\bar{B}X$$

$$R_B = AB + BX$$

$$S_B = A\bar{B}X \text{ and,}$$

$$\text{Circuit output function} = A\bar{X} + \bar{A}\bar{B}X$$

With these flip-flop input functions and circuit output function we can draw the logic diagram as follows.



iv) Design using JK Flip-flops

a) Write the excitation table for JK flipflop

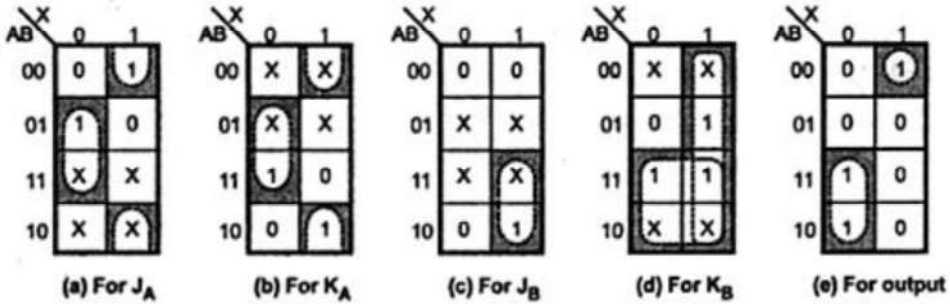
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

b) Determine the excitation table for the given circuit

Present state		Input	Next state		Flip-flop inputs				Output
A	B	X	A	B	J_A	K_A	J_B	K_B	Y
0	0	0	0	0	0	X	0	X	0
0	0	1	1	0	1	X	0	X	1
0	1	0	1	1	1	X	X	0	0
0	1	1	0	0	0	X	X	1	0
1	0	0	1	0	X	0	0	X	1
1	0	1	0	1	X	1	1	X	0
1	1	0	0	0	X	1	X	1	1
1	1	1	1	0	X	0	X	1	0

The first row of circuit excitation table shows that there is no change in the state for both flip-flops. The transition from 0 -> 0 for JK flip-flop requires inputs J and K to be 0 and X, respectively. Similarly, we can determine the inputs for each flip-flop for each row in the table referring present state, next state and excitation table. Let us use K-map.
Simplification to determine the flip-flop input functions and circuit output functions.

K-map simplification



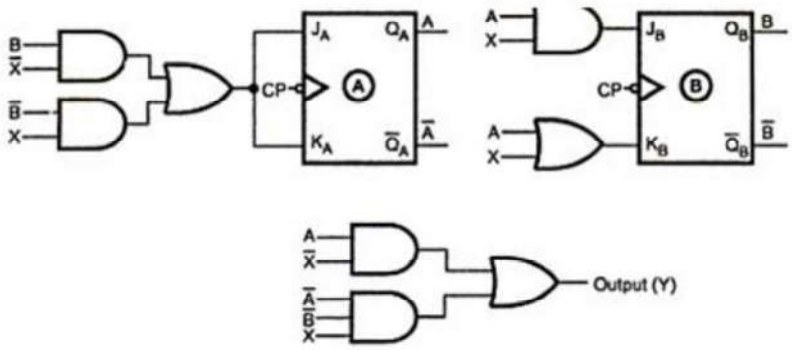
Therefore, input function for $J_A = B\bar{X} + \bar{B}X$

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

$J_B = AX$

$K_B = A + X$

Circuit output function $= A\bar{X} + \bar{A}\,\bar{B}\,X$



5.6 State Assignment

In sequential circuits is defined in terms of its inputs, present states, next state and outputs. To generate desired next state at particular present state and inputs, it is necessary to have specific flipflop inputs. These flipflop inputs are described by a set of boolean functions called flipflop input functions. To determine the flipflop input functions, it is necessary to represent states in the state diagram using binary values instead of alphabets. This procedure is known as stste assignments. We must assign binary values to the states in such a way that it is possible to implement flipflop input functions using minimum logic gates.

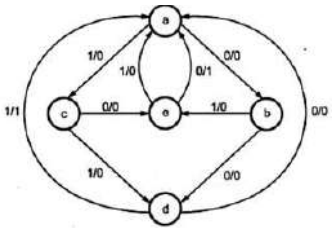
5.6.1 Rules for state assignments

There are two basic rules for making state assignments.

Rule 1: State having the same NEXT STATES for a given input condition should have assignments which can be grouped into logically adjacent cells in a Kmap.

Rule 2: States that are the NEXT STATES of a single state should have assignment which can be grouped into logically adjacent cells in a Kmap.

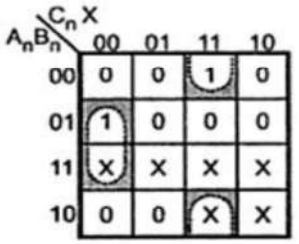
Example: 5.6.1 Design a sequential circuit for a state diagram shown in figure. Use state assignment rules for assigning states and compare the required combinational circuit with random state assignment.



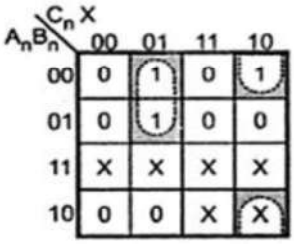
Using random state assignment we assign.
a = 000, b = 001, c = 010, d = 011 and e = 100. The excitation table with these assignments is as given inTable

Present state			Input	Next state			Output
A_n	B_n	C_n	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
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

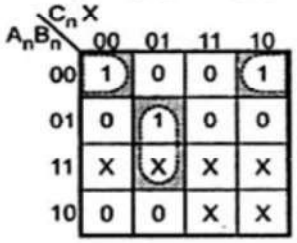
K-map simplification



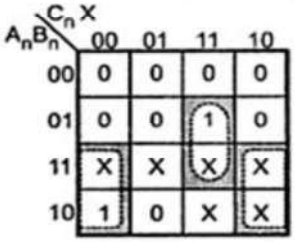
$D_A = B_n \overline{C}_n \overline{X} + \overline{B}_n C_n X$



$D_B = \overline{A}_n \overline{C}_n X + \overline{B}_n C_n \overline{X}$



$D_C = \overline{A}_n \overline{B}_n \overline{X} + B_n \overline{C}_n X$



$Z = B_n C_n X + A_n \overline{X}$

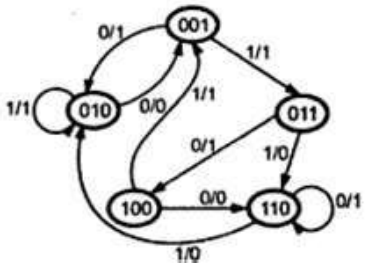
5.7 Design with unused state

There are occasions when a sequential circuit may use less than the available this maximum number of states. We can consider the unused states as don't care conditions and can be used to simplify the input functions to flip-flops.

Let us consider one example, First we will design the given sequential circuit without using unused states and then we will design the given sequential circuit using unused states.

Example:5.7.1 Design the sequential circuit for the state diagram shown in figure. Use JK flip-flops

Solution: The state table for given state diagram is as follows



Present State			Input	Next State			Flip-flop inputs						Output
A	B	C	X	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C	Y
0	0	1	0	0	1	0	0	X	1	X	X	1	1
0	0	1	1	0	1	1	0	X	1	X	X	0	1
0	1	0	0	0	0	1	0	X	X	1	1	X	0
0	1	0	1	0	1	0	0	X	X	0	0	X	1
0	1	1	0	1	0	0	1	X	X	1	X	1	1
0	1	1	1	1	1	0	1	X	X	0	X	1	0
1	0	0	0	1	1	0	X	0	1	X	0	X	0
1	0	0	1	0	0	1	X	1	0	X	1	X	1
1	1	0	0	1	1	0	X	0	X	0	0	X	1
1	1	0	1	0	1	0	X	1	X	0	0	X	0

For J_A

AB \ CX	00	01	11	10
00			0	0
01	0	0	1	1
11	X	X		
10	X	X		

For K_A

AB \ CX	00	01	11	10
00			1	1
01	X	X	X	X
11	X	X		
10	1	0		

For J_B

AB \ CX	00	01	11	10
00			X	X
01	X	X	X	X
11	0	1		
10	0	1		

For K_B

AB \ CX	00	01	11	10
00			X	X
01	1	0	0	1
11	0	0		
10	X	X		

For J_C

AB \ CX	00	01	11	10
00			X	X
01	1	0	X	X
11	0	0		
10	0	1		

For K_C

AB \ CX	00	01	11	10
00			1	0
01	X	X	1	1
11	X	X		
10	X	X		

For Output

AB \ CX	00	01	11	10
00			1	1
01	0	1	0	1
11	1	0		
10	0	1		

Therefore, input functions for

$$J_A = \overline{A}BC$$

$$K_A = A\overline{CX} + \overline{A}C$$

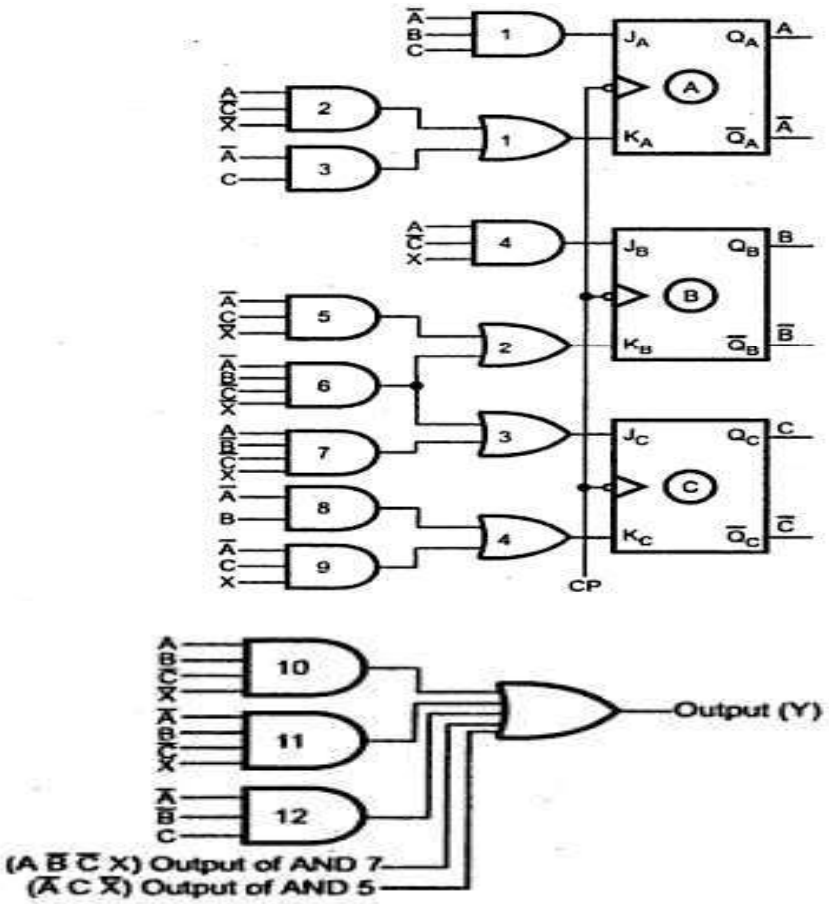
$$J_B = A\overline{CX}$$

$$K_B = \overline{A}B\overline{C}\,\overline{X} + \overline{A}C\overline{X}$$

$$J_C = \overline{A}B\overline{C}\,\overline{X} + A\overline{B}\,\overline{C}X$$

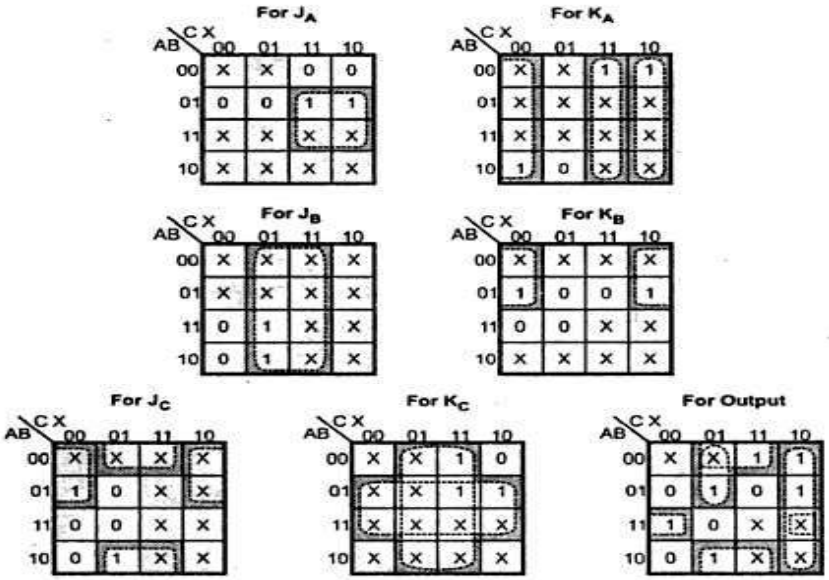
$$K_C = \overline{A}B + \overline{A}CX \text{ and}$$

Circuit output function, $Y = AB\,\overline{C}\,\overline{X} + \overline{A}\,B\overline{C} + A\overline{B}\,\overline{C}X + \overline{A}\,\overline{B}C + \overline{A}C\overline{X}$



Let us see the circuit design with the use of unused states. These unused states 000,101 and 111 are considered as a don't cares and are used to simplify the kmaps as follows:

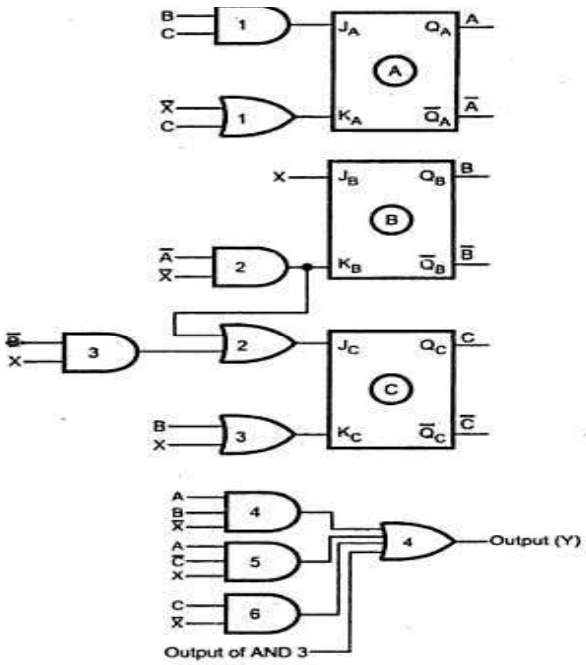
K-map simplification



Therefore, input functions for

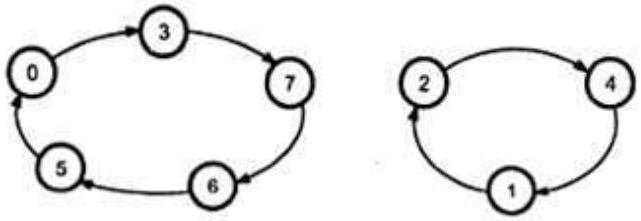
$$J_A = BC$$
$$K_A = \overline{X} + C$$
$$J_B = X$$
$$K_B = \overline{A}\,\overline{X}$$
$$J_C = \overline{A}\,\overline{X} + \overline{B}X \quad \text{and}$$
$$K_C = B + X$$

The circuit output function $Y = AB\overline{X} + A\overline{C}X + \overline{B}X + C\overline{X}$



5.7.1 Lockout Conditions

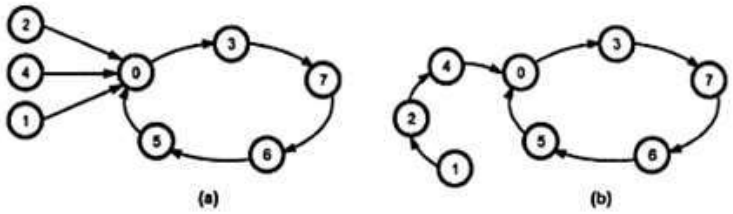
In a counter if the next state of some unused state is again an unused state and if by chance the counter happens to find itself in the unused states and never arrived at a used state then the counter is said to be in lockout conditions. This is illustrated in the Fig. 9.37



(a) Desired sequence

(b) Unused state forming lockout

The circuit that goes in lockout condition is called bushless circuit. Toi make sure that the coiunter will come to the initial state from any unused state, the additional logic circuit is necessary. To ensure that the lockout does not occur, the counter should be designmed by forcing the next state to be the initial state from the unused states as shown in figure.

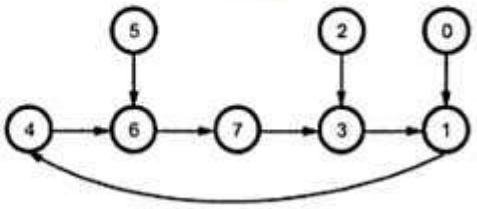


For example, as shown in Fig, actually it is not necessary to force all unused states into initial state. Forcing any one state is sufficient. Because, if counter initially goes to unused state which is not forced, it will go to another unused state. This will continue until it reaches the forced unused state. Once forced unused state is reached next state is used state, and circuit is lock free circuit. This is illustrated in Fig. 9.38(b)

Examle :*Design a synchronous counter for*

$4 \rightarrow 6 \rightarrow 7 \rightarrow 3 \rightarrow 1 \rightarrow 4 \ldots$

Avoid lockout condition. Use JK type design.



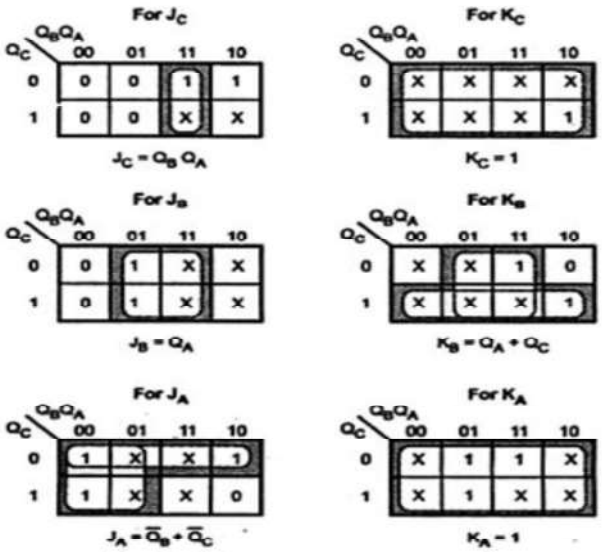
Solution : State diagram

Here, states 5,2 and 0 are forced to go into 6,3 and 1 state, respectively to avoid lockout condition.

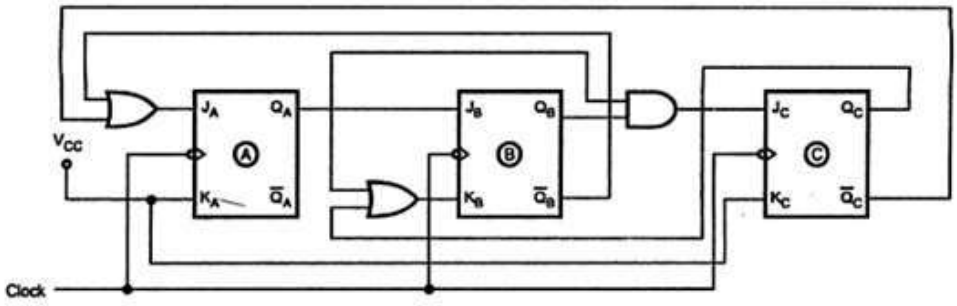
Excitation table

Present states			Next states			Flip-flop inputs					
A	B	C	A_{+1}	B_{+1}	C_{+1}	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

K-map simplification



Logic diagram



Example : Design a synchronous conter with states 0,1,2,3,0,1……..using JK FFs.

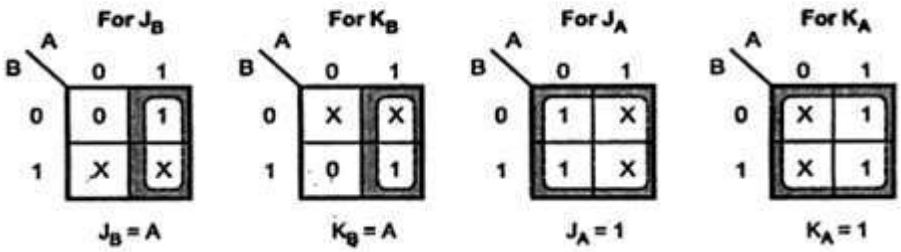
Solution:

Transition table

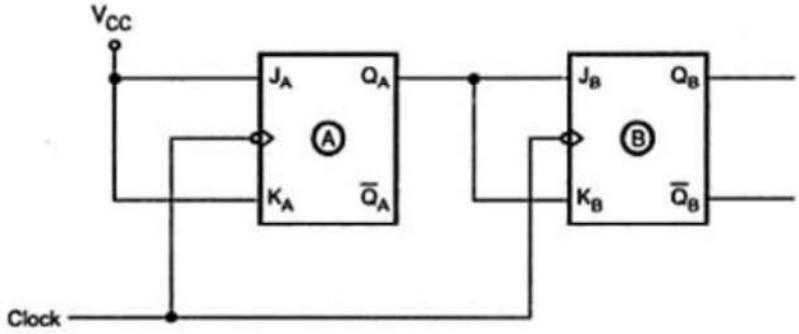
Present state		Next state		Flip-flop inputs			
Q_B	Q_A	Q_{B+1}	Q_{A+1}	J_B	K_B	J_A	K_A
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

Note : Refer excitation table (Table 6.7 (b)) for JK flip-flop.

K-map simplification



Logic diagram



Exercise problems

1. Using JK design a synchronous counter that has the following sequence:---0-2-5-6-0--- undesired states 1,3,4,7 must always go to 0 on the next clock pulse.
2. Design the circuit to generate the sequence:
   0---2---5---4---7---3.
3. Design sequence generator to generate sequence1,9,2,7,3,6 using JK flipflop.

5.8 Serial Adder

Serial adder is a circuit in which bits are added a pair at a time. When speed is not of great importance, it is a cost-effective option to use serial adder.

Mealy-type FSM for serial Adder

The Fig. 9.23 shows the block diagram of the serial adder. It includes three shift registers that are used to hold number A, number B and sum of numbers A and B. Shift registers A and B have parallel-load capability. Thus, we can load values of A and B into these registers. At each clock cycle, a pair of bits is added by the sequential circuit used as adder, and at the end of the cycle the resulting sum bit is shifted into the C register. Therefore, n clock cycles are required to add n-bit number. After n clock cycles we get the output sum at the parallel-out of shift register C.

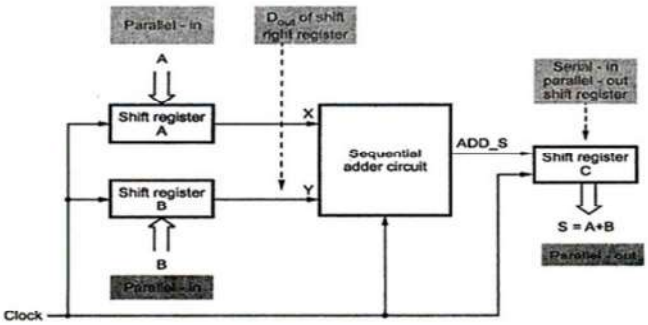
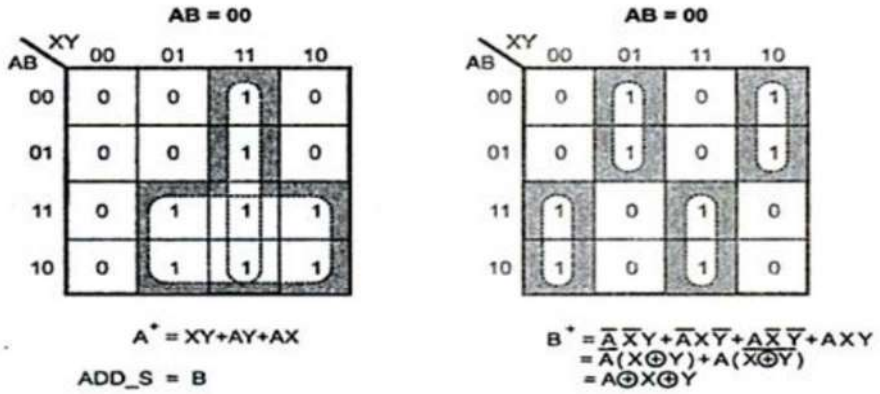


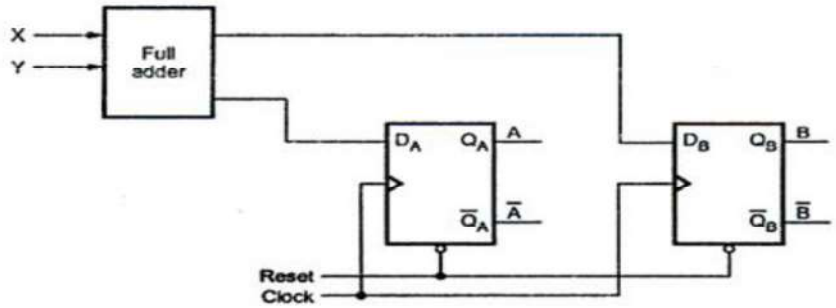
Figure 5.9:block diagram of serial adder

Present state AB	Next state XY = 00	00	10	11	Output ADD_S
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

K-map simplification for deriving next-state and output expressions is as follows.



Here, also expressions for A and B correspond to the carry and sum expressions in the full-adder circuit. The Fig. 9.30 shows the implementation of sequential adder circuit.



5.9 Algorithm State Machines(ASM)

The main components of digital system are: Control logic and data path. The data processing path commonly known as a datapath, manipulates data in registers according to systems requirements. The control logic initiates a sequence of commands to the data path. It uses status conditions from the datapath as decision variables for determining the sequence of control signals.

The control sequence and data path tasks of a digital system are specified by means of a hardware algorithm. A hardware algorithm is a step by step procedure to implement the desired task with selected circuit components. We know that a flow chart is an algorithm. A special flowchart that has been developed specifically to define digital hardware algorithms is called Algorithmic State Machine.

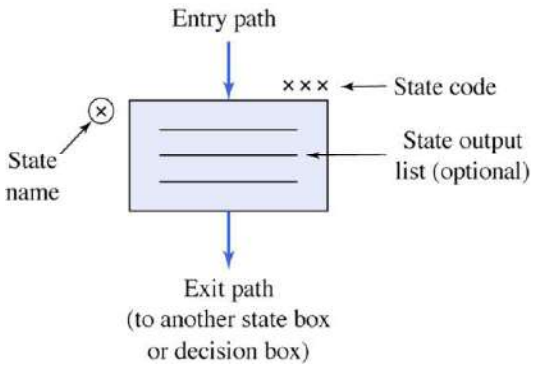
ASM stands for 'Algorithm State Machine 'or simply state machine is the another name given to sequential network is used to control a digital system which carries out a step by a step -by -step procedure .It should be noted that ASM charts represent physical hardware and offers several advantages.

1. Operation of a digital system can be easily understood by inspection of the ASM chart.
2. ASM charts represent physical hardware.
3. The ASM chart is equivalent to a state graph, and it leads directly to a hardware realization.
4. ASM charts can be described the operation of both combinational and sequential circuits.
5. ASM charts are easier to understand and can be converted several equivalent form.
6. The ASM chart may be equivalently expressed as a state and output table.

5.9.1 Component of an ASM Chart

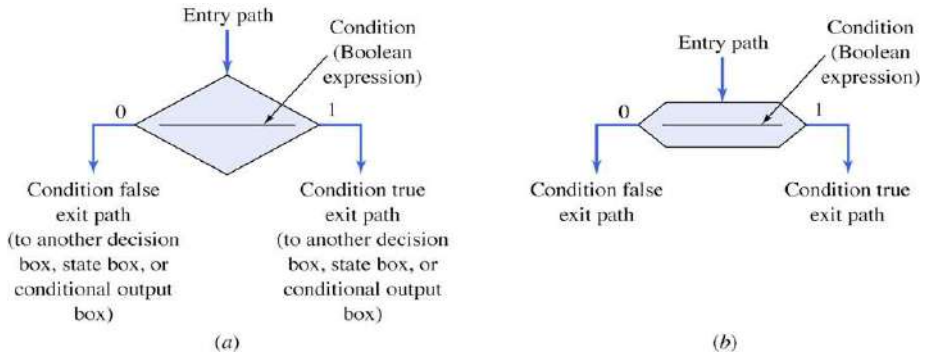
5.9.1.1State Box*:*

The state of the system is represented by a state box .It is a rectangular box .At the top left hand corner the name of state is shown ,which at the top right hand corner the state assignment is given .Within the state box ,the output signals are listed .



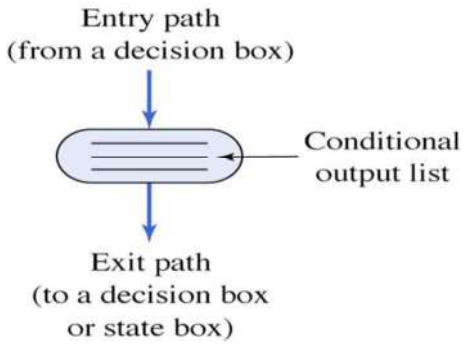
5.9.1.2Decision box:

It a diamond –shaped box with true false branches .Boolean condition is placed in the box and the decision is made from the value of one or more input signals .The decision box must follow and be associated with a state.



5.9.1.3Conditional output box:

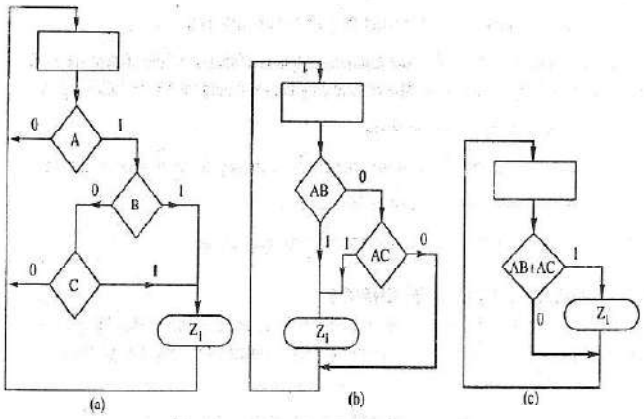
A condition output box is shown in Fig. is a rectangular box with curved ends .It contain conditional output list .The conditional output depends on both the state of the system and the inputs .Therefore the conditional output signals are sometimes known as Mealy output .A condition output must follow a decision box.



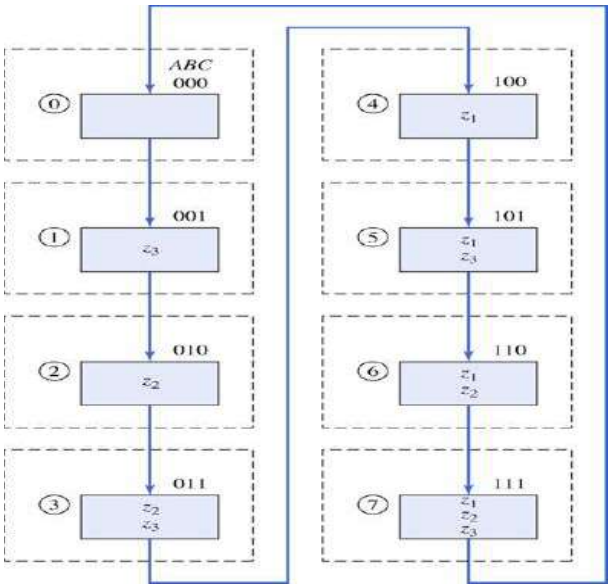
5.9.2 Equivalent ASM charts

ASM charts are not unique, it may have more than one equivalent form Fig. shown three equivalent ASM charts for combinational network Z=A(B+C).

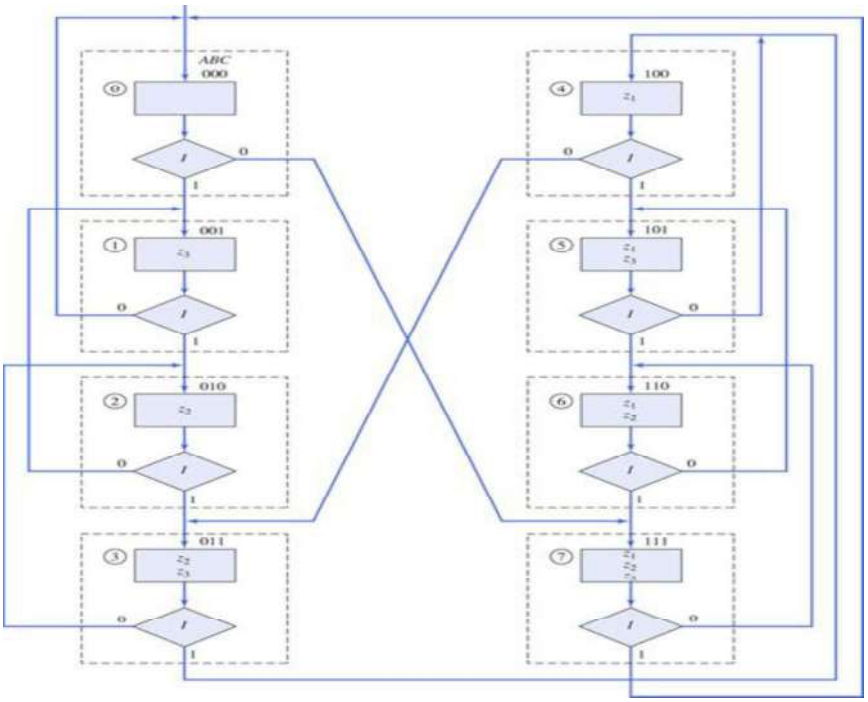
Fig. Equivalent ASM charts for Z=A(B+C)



5.9.3 ASM chart for mod-8 binary counter



5.9.4 ASM chart for mod-8 binary up/down counter



5.9.5Conversion of State Diagram to An ASM Chart

5.9.5.1 Mealy Machine.

In case of Mealy machine, output is a function of both present state and input. For construction of ASM chart from Mealy state diagram, we should follow the following steps.

1. Represent each state by state box.
2. Put input in decision box after each state box.
3. The Mealy output appears in conditional output boxes since they depend on both the state and input.
4. Mealy circuit output written only when it is equal to '1' i.e. true.
5. Depending on value of input connect the path to next state box.

Example

Convert the state diagram of Figure below to ASM chart.

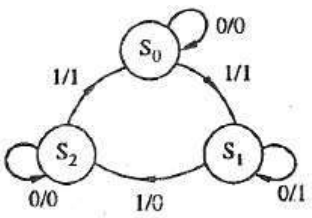


Figure: state diagram

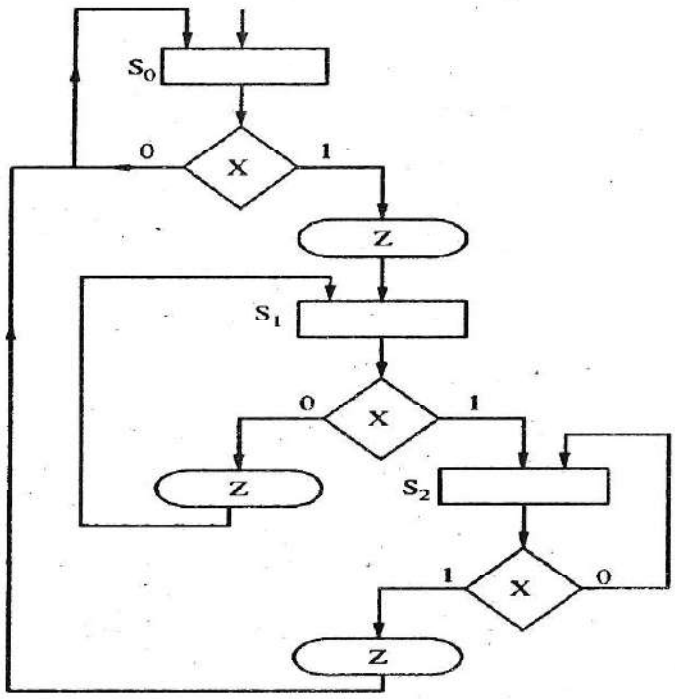


Figure: ASM chart

5.9.5.2 Moore Machine

In case of Moore machine, output is a function of the present state only. For construction of ASM chart from Moore state diagram, we should follow the following steps

1. Represent each state by state box.
2. The Moore output is placed in the state boxes since they do not depend on the input.
3. After each state box put the input in decision box.
4. Depending on value of input connect the path to next state box.

Example

Convert the state diagram of Fig. below to ASM chart.

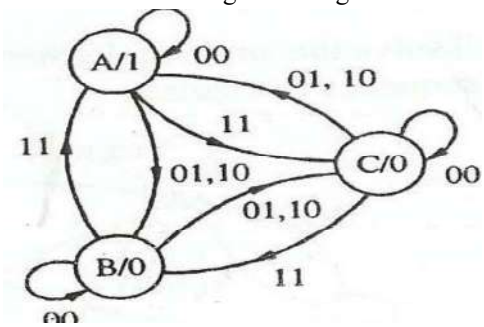


Figure: state diagram

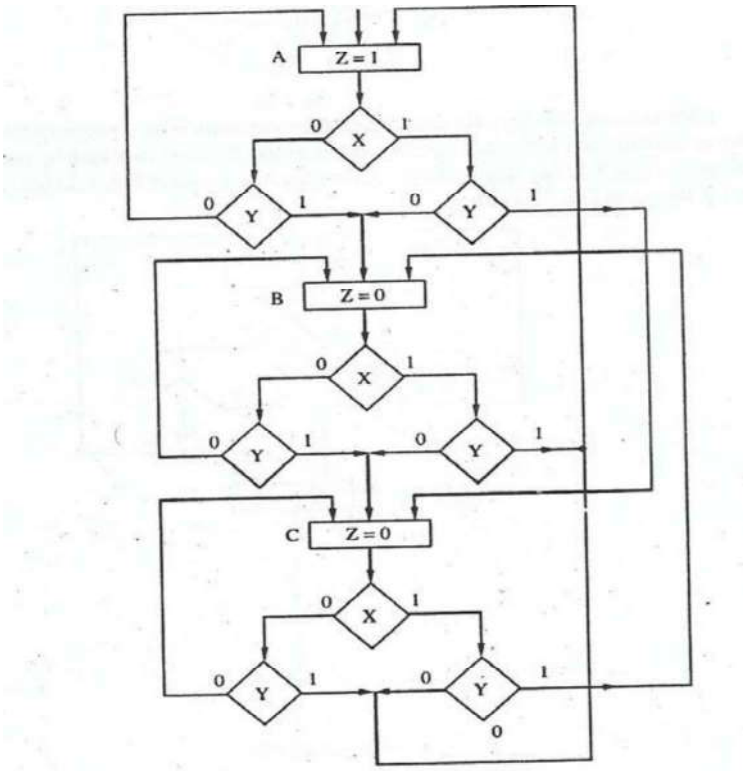


Figure: ASM chart

5.10 Asynchronous Sequential Circuits

5.10.1 Introduction

We have seen in synchronous sequential circuits memory elements are clocked flip-flops. In asynchronous sequential circuits memory elements are either un clocked flip-flops or time delay elements. Therefore in asynchronous sequential circuits change in input signals can affect memory element at any instant of time. In synchronous circuits the designer has to consider the time delays involved to determine the maximum operating speed of the clock. In asynchronous circuits clock is absent and state change occurs according to delay times of the logic. Due to this asynchronous circuits are more difficult to design. However because of absence of clock asynchronous circuits are faster than synchronous circuits.

5.10.2 Types of asynchronous circuits

It consists of a combinational circuit and delay elements connected to form feedback loops. There are n input variables, m output variables, and k internal states. The delay elements provide a short term memory for the sequential circuit. The present state and next state variables in asynchronous sequential circuits are called secondary and excitation variables respectively. When an input variable changes in value the secondary variables ie,. y1,y2,…,yr do not change instantaneously. Certain amount of time is required for the input signal to propagate from the

input terminals through the combinational circuits and the delay elements. The combinational circuit generates Y excitation variables which gives the next state of the circuit. The excitation variables are propagated through delay elements to become the new present state for the secondary variables, ie., y1,y2,…yr. In the steady state condition excitation and secondary variables are same, but during transition they are different.

According to how input variables are to be considered there are two types of asynchronous circuits

Fundamental mode circuits

Pulse mode circuits

5.10.2.1 Fundamental mode circuits

It assumes that:

Input changes should be spaced in time by at least, the time needed for the circuit to settle into a stable state following an input change. The input variables should change only when the circuit is stable.

Only one input variable can change at a given instant of time

Inputs are levels and not pulses

Delay lines are used as memory elements

5.10.2.2 Pulse mode circuit

It assumes that:

The input variables are pulses instead of levels

The width of the pulses is long enough for the circuit to respond to the input

The pulse width must not be so long that it is still present after the new state is reached

Pulses should not occur simultaneously on two or more input lines

Flip-flops are commonly used as a memory elements

Memory element transitions are initiated only by input pulses

Input variables are used only in the un complemented or the complemented forms but not both.

Analysis of pulse mode asynchronous circuits

In the Analysis of pulse mode asynchronous sequential circuits, circuits respond immediately to pulse on their inputs rather than waiting for clock signal, as in synchronous sequential circuits. The pulse mode circuits assume that pulses do not occur simultaneously on two or more input lines means that a circuit with n input lines has only n+ 1 input condition rather than 2^n as is the case for synchronous circuits. They also assume that a state transition can occur only if an input pulse occurs. Hence the memory elements of the circuit respond only when an input pulse arrives.

Design of pulse mode circuit

The design of pulse mode circuits is similar to the design of synchronous circuits. However when designing pulse mode circuits remember that no clock pulse is present inputs occur on only one line at a time and only un complemented forms of input signals may be used.

The absence of a clock pulse indicates that latch or flip-flop triggering must be accomplished by utilizing the pulses on the input signals and therefore all circuit timing information must be obtained from the input pulses. Hence the input pulses not only provide

input information but also assume the functions performed by the clock pulse in synchronous circuits.

The steps involved in the design of pulse mode asynchronous sequential circuits are

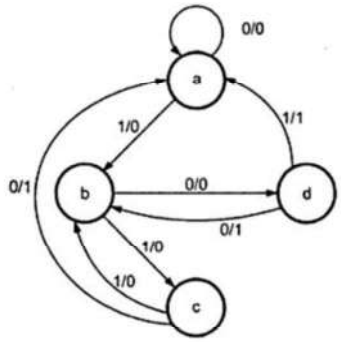
Define states and draw a state diagram and/or state table of the circuit.
Minimize the state table.
Do state assignment.
Choose the type of latch or flip-flop to be used and determine excitation equations.
Construct excitation table for the circuit.
Determine the output equation and the flip-flop input equation using kmap simplifications.
Draw the logic diagram.

5.11 Analysis of fundamental mode sequential circuits:

Fundamental mode asynchronous sequential circuit analysis requires careful attention because these circuits utilize unclocked memory and level inputs. The procedure to analyze these circuits is as follows:

Determine the next state and output equation from the given sequential circuits
Construct the state table
Construct the transition table
Construct output map

Example: Design a clocked sequential machine using T flip-flops for the following state diagram. Use state reduction if possible. Also use straight binary state assignment.



Solution: State table for the given state diagram shown in table

Present State	Next state		Output (Z)	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	d	c	0	0
c	a	b	1	0
d	b	a	1	1

Even though states a and c are having same next states for input X=0 and X=1, as the outputs are not same state reduction is not possible.

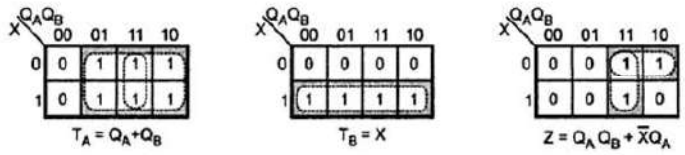
Using straight binary assignments as a=00, b=01, c=10 and d=11, the transition table(Excitation table) is shown in table.

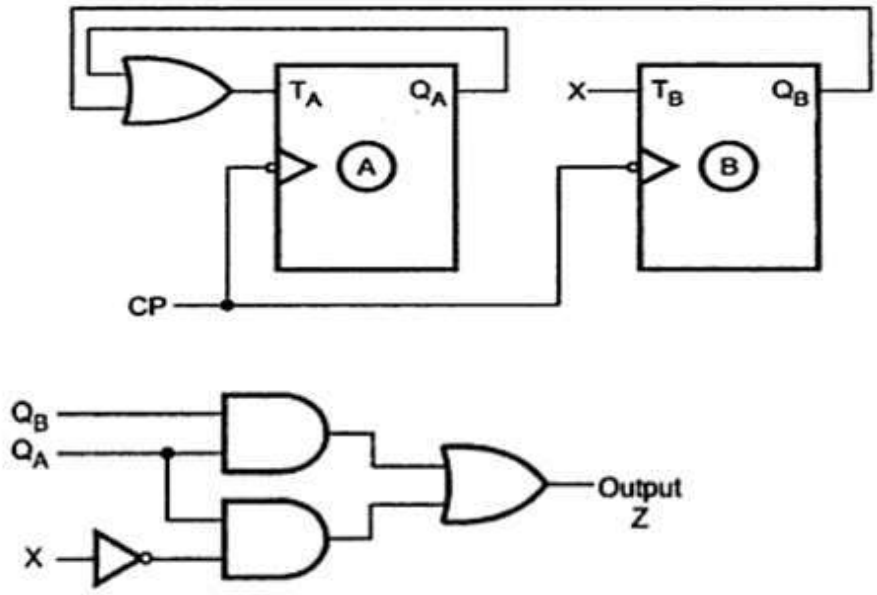
Input	Present state		Next state		Flip-flop input		Output
X	Q_A	Q_B	Q_{A+1}	Q_{B+1}	T_A	T_B	Z
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	1	0	1
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	0
1	0	1	1	0	1	1	0
1	1	0	0	1	1	1	0
1	1	1	0	0	1	1	1

The flip-flop inputs and the circuit outputs are

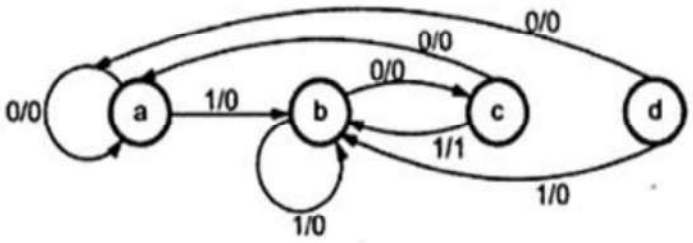
$$T_A = f(Q_A, Q_B, X)$$
$$T_B = f(Q_A, Q_B, X)$$
$$Z = f(Q_A, Q_B, X)$$

K-map Simplification





Example: Design a clocked sequential machine using JK flip-flops for the state diagram shown in figure. Use state reduction if possible. Make proper state assignment.



State table

State	Output	
	X = 0	X = 1
a	a, 0	b, 0
b	c, 0	b, 0
c	a, 0	b, 1
d	a, 0	b, 0

In the above table a, d have the same output hence we can reduce it to single.

Reduced state table

State	Output	
	X = 0	X = 1
a	a, 0	b, 0
b	c, 0	b, 0
c	a, 0	b, 1

Now each state is assigned with binary values. Since there are three states, number of flip-flops required is two and 2 two binary numbers are assigned to the states as shown below:

a =00, b=01 and c=10

With above state assignment, the excitation table can be given as shown below

Input	Present State		Next State		Flip-flop inputs				Output
X	Q_B	Q_A	Q_{B+1}	Q_{A+1}	J_B	K_B	J_A	K_A	Z
0	0	0	0	0	0	X	0	X	0
0	0	1	1	0	1	X	X	1	0
0	1	0	0	0	X	1	0	X	0
1	0	0	0	1	0	X	1	X	0
1	0	1	0	1	0	X	X	0	0
1	1	0	0	1	X	1	1	X	1

K-map Simplification

For J_B

Q_BQ_A / X	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$J_B = \overline{X}\,Q_A$

For K_B

Q_BQ_A / X	00	01	11	10
0	X	X	X	1
1	X	X	X	1

$K_B = 1$

For J_A

Q_BQ_A / X	00	01	11	10
0	0	X	X	0
1	1	X	X	1

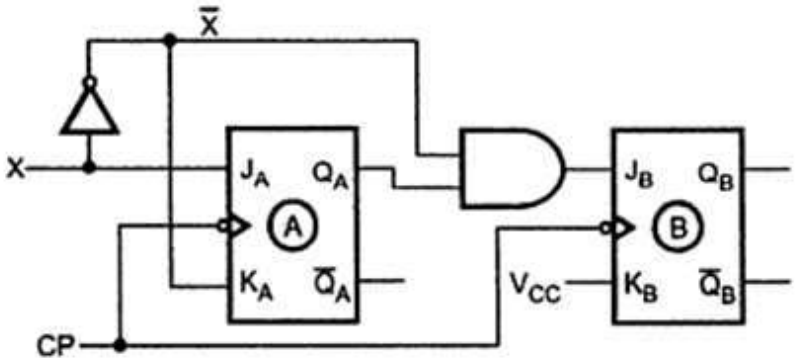
$J_A = X$

For K_A

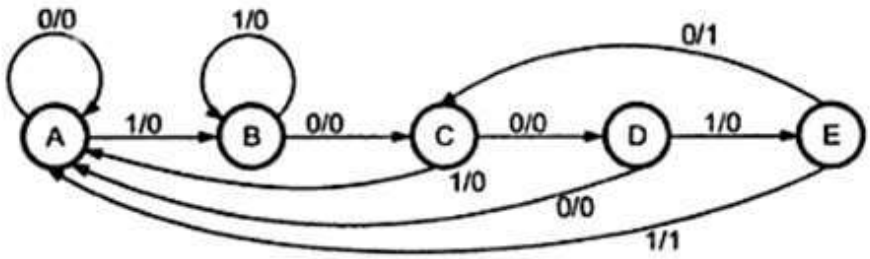
Q_BQ_A / X	00	01	11	10
0	X	1	X	X
1	X	0	X	X

$K_A = \overline{X}$

Logic diagram



Example :Design the sequential circuit using D flip-flops for the state diagram given in figure



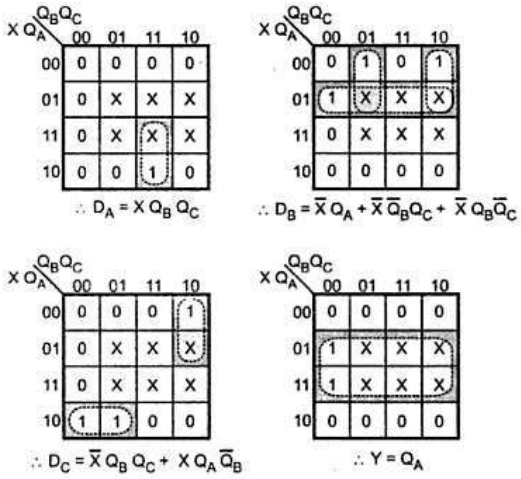
Solution : State assignments are A = 000, B = 001, C = 010, D = 011, E = 100. Three D flips-flops are required.

X	Present state			Next state			Output
	Q_A	Q_B	Q_C	Q_{A+1}	Q_{B+1}	Q_{C+1}	Y
0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0
0	0	1	1	0	0	0	0
0	1	0	0	0	1	0	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	1	0
1	0	1	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	0	0	1

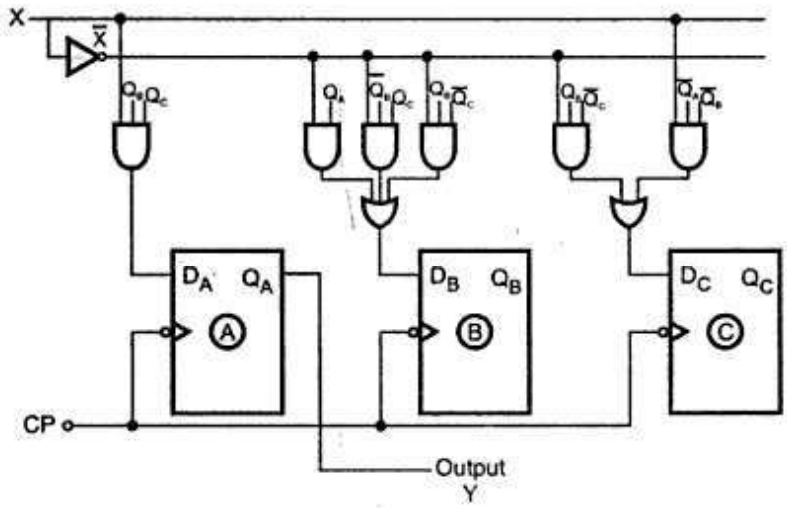
The flip-flop inputs D_A, D_B and D_C are not included in the excitation table as they equal to the next state.

$$D_A = Q_{A+1}, \quad D_B = Q_{B+1}, \quad \text{and} \quad D_C = Q_{C+1}$$

Kmap simplifications

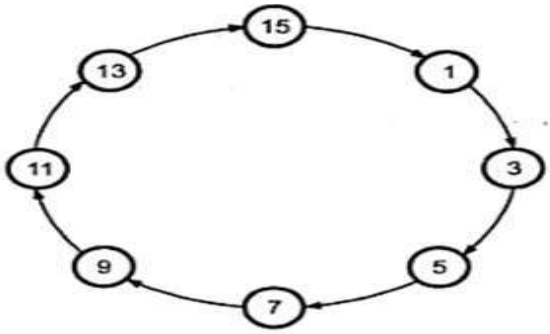


The logic diagram is as shown below :



Example: Design and implement 4 bit binary counter (using D flip-flops) which counts all possible odd numbers only.

Solution: The state diagram for given problem is as shown in the figure



Excitation table

Present state				Next state			
Q_A	Q_B	Q_C	Q_D	Q_{A+}	Q_{B+}	Q_{C+}	Q_{D+}
0	0	0	0	×	×	×	×
0	0	0	1	0	0	1	1
0	0	1	0	×	×	×	×
0	0	1	1	0	1	0	1
0	1	0	0	×	×	×	×
0	1	0	1	0	1	1	1
0	1	1	0	×	×	×	×
0	1	1	1	1	0	0	1
1	0	0	0	×	×	×	×
1	0	0	1	1	0	1	1
1	0	1	0	×	×	×	×
1	0	1	1	1	1	0	1
1	1	0	0	×	×	×	×
1	1	0	1	1	1	1	1
1	1	1	0	×	×	×	×
1	1	1	1	0	0	0	1

K-map simplification

For D_A

$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	X	0	0	X
01	X	0	1	X
11	X	1	0	X
10	X	1	1	X

$D_A = Q_A \overline{Q}_C + Q_A \overline{Q}_B + \overline{Q}_A Q_B Q_C$

For D_B

$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	X	0	1	X
01	X	1	0	X
11	X	1	0	X
10	X	0	1	X

$D_B = Q_B \overline{Q}_C + \overline{Q}_B Q_C$

For D_C

$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	X	1	0	X
01	X	1	0	X
11	X	1	0	X
10	X	1	0	X

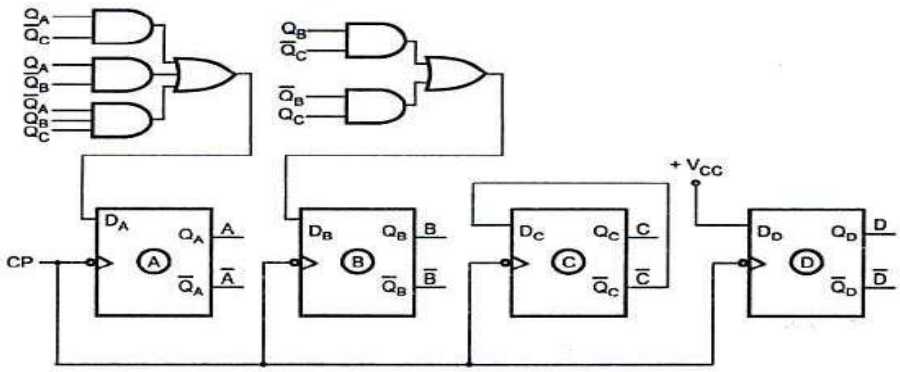
$D_C = \overline{Q}_C$

For D_D

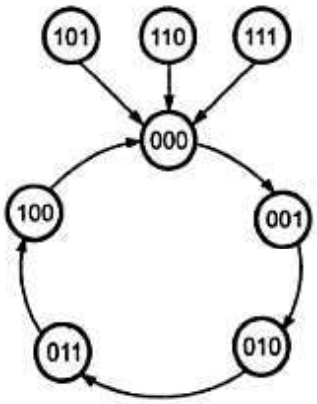
$Q_A Q_B$ \ $Q_C Q_D$	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	1	1	X
10	X	1	1	X

$D_D = 1$

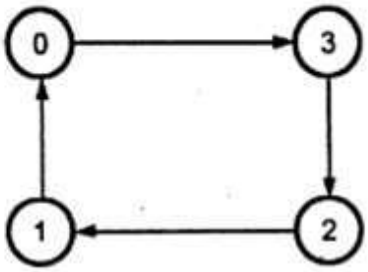
Logic diagram



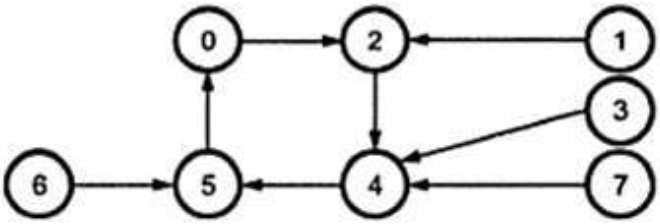
1. *Draw and explain the block diagram of Moore model.*
2. *Draw and explain the block diagram of Mealy model.*
3. *Compare Moore and Mealy models.*
4. *Give recommended steps for the design of a clocked Synchronous Sequential Networks.*
5. *What is state diagram ?*
6. *What is state table ?*
7. *Explain the process of state reduction with the help of suitable example.*
8. *State the rules for state assignments.*
9. *Design a sequential circuit for the following state diagram.*



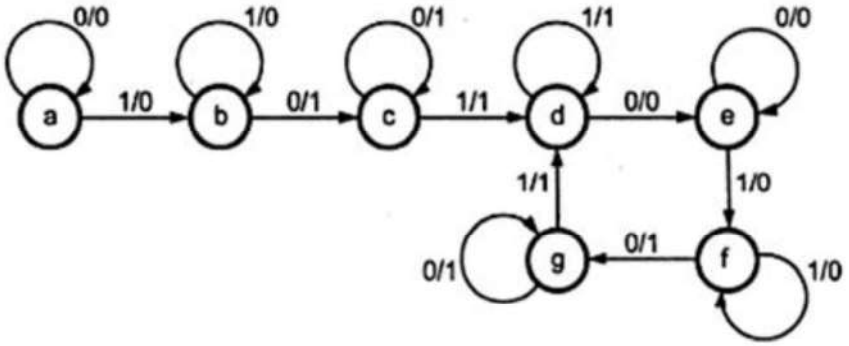
Design the sequential circuit for the given state diagram using T flip-flop.



Design the sequential circuit for the given state diagram using D flip-flop.



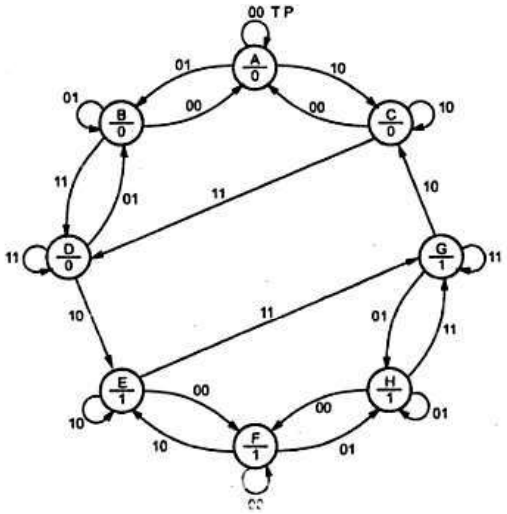
Obtain the state table for the given state diagram. Use state reduction technique.



Example: Design a T flipflop from logic gates

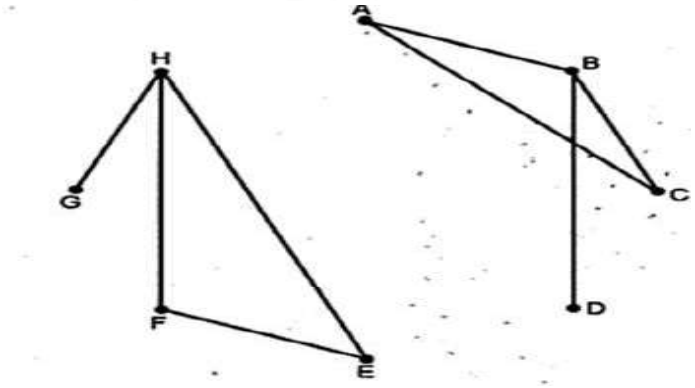
Solution : The T flip-flop has one excitation input and one clock input. But here we use anothe input P that will function as a clock. The flip-flop will change state if T = 1 and when he clock(P) changes from 1 to 0. Under all other input conditions, output Q will remain constant. We assume that T and P do not change simultaneously.

The state diagram for above problem state is as shown in figure



Draw the primitive flow table

Present state	Next state, Output Z for AB inputs			
	00	01	11	10
A	(A), 0	B, –	–, –	C, –
B	A, –	(B), 0	D, –	–, –
C	A, –	–, –	D, –	(C), 0
D	–, –	B, –	(D), 0	E, –
E	F, –	–, –	G, –	(E), 1
F	(F), 1	H, –	–, –	E, –
G	–, –	H, –	(G), 1	C, –
H	F, –	(H), 1	G, –	–, –



The merger graph shown in figure gives the four compatible pairs as a set of maximal compatibles

(A, B, C) → S_0
D → S_1
(E, F, H) → S_2
G → S_3

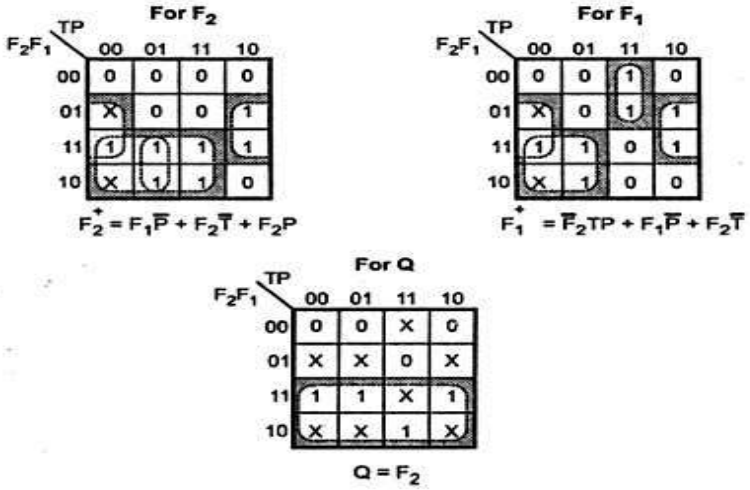
This set of maximum compatibles covers all of the original states resulting in the reduced flow table as shown in the figure

Present state	Next state, Output Q for T P inputs			
	00	01	11	10
S_0	(S_0), 0	(S_0), 0	S_1, –	(S_0), 0
S_1	–, –	S_0, –	(S_1), 0	S_2, –
S_2	(S_2), 1	(S_2), 1	S_3, –	(S_2), 1
S_3	–, –	S_2, –	(S_3), 1	S_0, –

By making state assignment as S0=00, S1=01, S2=11 and S3=10 we can avoid all the races. Substituting state assigned values to state we get transition table as shown in figure.

Present state		Next state, Output Q for T P inputs			
		Next state			
F_2	F_1	00	01	11	10
0	0	00,0	00,0	01,–	00,0
0	1	–,–	00,–	01,0	11,–
1	1	11,1	11,1	10,–	11,1
1	0	–,–	11,–	10,1	00,–

K-map simplification



5.12Design of fundamental mode sequential circuits

The steps involved in designing of asynchronous sequential circuit.

Construction of a primitive flow table from the problem statement. An intermediate step may include the development of a state diagram.

Primitive flow table is reduced by eliminating redundant states by using state reduction technique.

State assignment is made.

The primitive flow table is realized using appropriate logic elements.

5.13Race Free State Assignment

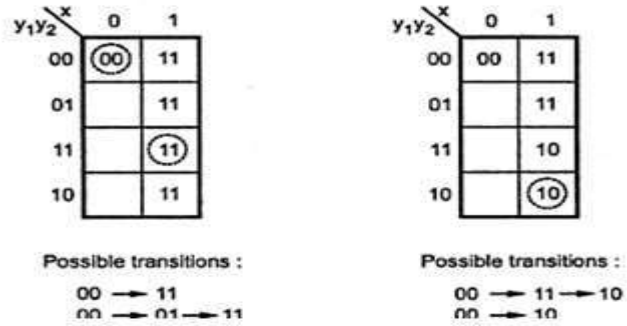
The state assignment step in asynchronous circuits is essentially the same as it is for synchronous circuits, except for one difference. In synchronous circuits, the state assignments are made with the objective of circuit reduction. In asynchronous circuits, the objective of state assignment is to avoid cirtical races.

5.13.1Races and cycles

When two or more binary state variables change their values in response to change in an input variable, race condion occurs in an asynchronous sequential circuit. In case of unequal delays, a race condition may cause the state variables to change in an unpreditable manner. For example, if there is a change in two state variables due to change in input variuable such that both change from 00 to 11. In this situation the difference in delays may cause the first variable to change faster than the second resulting the state variables to change in sequence from 00 to 11 and then to 11. On the other hand, if the second variables changes faster than the first, the state variables changes from 00 to 01 and then to 11. If the state that the circuit reaches does not depend on the order in which the state variable changes, the race condition is not harmful and then it is called

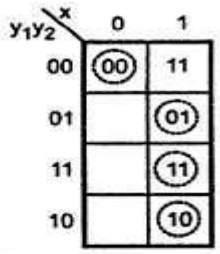
5.13.1.1Noncritical Races

Fig. 11.12 illustrates noncritical areas. It shows transition tables in which X is a input variable and Y_1 Y_2 are the state variables. Consider a circuit is a stable state Y_1Y_2X =000 and there is a change in input from 0 to 1. With this change in the input there are three possibilities that the state variables may change. They can either change simultaneously from 00 to 11, or they may change in sequence from 00 to 01 and then to 11, or they may change in sequence from 00 to 10 and then to 11. In all cases final stable state is 11, which results in a noncritical race condition. In Fig. 11.12 (b) final stable state is Y_1Y_2X =101.



5.13.1.2Critical Races

Fig. 11.13 illustrates critical race. Consider a circuit is in a stable Y_1Y_2X = 000 and there is a change in input from 0 to 1. If state variables change simultaneously, the final stable state is Y_1Y_2X = 111. If Y_2 changes to 1 before Y_1 because of unequal propagation delay, then the circuit goes to the stable state 011 and remain there. On the other hand, if Y_1 changes faster than Y_2, then the circuit goes to the stable state 101 and remain there. Hence, the race is critical because the circuit goes to different stable states depending on the order in which the state variables change.



Possible transitions :

00 → 11
00 → 01
00 → 10

5.13.1.3Cycles

A cycle occurs when an asynchronous circuit makes a transition through a series of unstable states. When a state assighment is made so that it introduces cycles, care must be taken to ensure that each cycle terminates on a stable state. If a cycle does not contain a stable state, the circuit will go from one unstable state to another, until the inputs are changed. Obviously such a situation must always be avoided when designing asynchronous circuits.

Two techinques are commonly used for making a critical-race free state assignment.

1. Shared row state assignment
2. One hot state assignment.

Shared Row State Assignment

Races can be avoided by making a proper binary assignment to the state variables. Here, the state variables are assigned with binary numbers in such a way that only one state variable can change at any one time when a transition occurs. To accomplish this, it is necessary that states between which transition occur be given adjacent assignments. Two binary values are said to be adjacent if they differ in only one variable. For example, 110 and 111 are adjacent because they differ only in the third bit.

Fig. 5.11shows the transition diagram. The transition diagram shows that there is transition from state a to state b and transition from state a to state c. The state a is assigned binary value 00 and state c is asigned binary value 11. This assigment will cause a critical race during the transition from a to c because there are two changes in the binary table variables. A race free assignment can be obtained by introducing addition binary state say d with binary value 10, which is adjacent to both a and c. Fig. 5.11shows the modified transition diagram. As shown in the Fig. 5.11the transition from a to c will go through d. This causes the binary variables to change from 00 -> 10 - > 11 which satisify the condition that only one binary variable changes during each state transition, thus acvoiding the critical race.

This technique is called shared row state assignment because in this technique extra state, i.e. extra row is introduced in a flow table. This extra state is shared between two stable states.

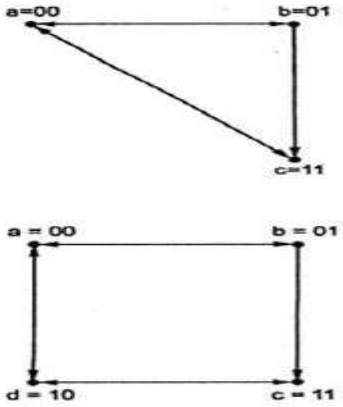


Figure 5.11

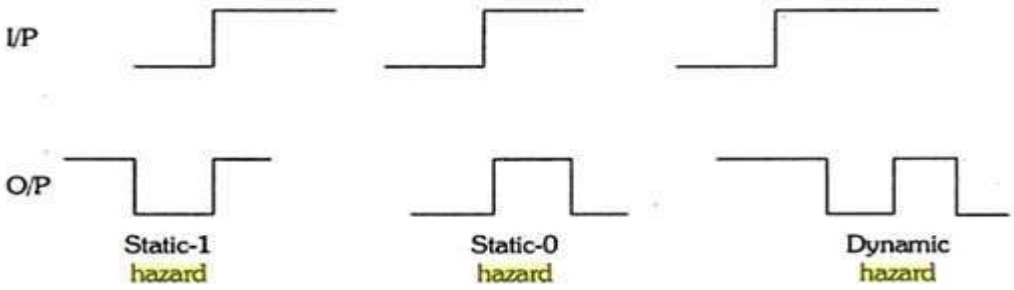
3.15.3 Hazards

In analysis of combinational logic circuits, we ignore the circuit delay and predict only the steady state behaviour. That is, the circuit output is afunction of its inputs under assumption that the inputs have been stable for a long time, relative to the delays of the elements of the circuit.

With the effect of the circuit delays, the transient behaviour of the combinational circuit may vary from the predicted steady state analysis. With this, the circuit may produce a short pulse of a output, also called of "glitch", at that time, when the steady state analysis predits that there is no change in the output.

A "Hazard" is a unwanted switching transient i.e. spike or glich, that occurs due to unequal path or unequal propagatiomn delay through a combinational circuit. There are two types of hazards, namely (1) static hazard (2) Dynamic hazard. These two hazards occur in a combinational circuits, similarly "essential hazard" a third type, occur in the sequential circuits.

The Fig. 3.232 shows the Hazards.



5.14.1Static hazard. Static hazard is a condition which results in asingle momentary in correct output due to change in a input vvariable when the output is expected to remain in the same state. There are two types, static-1 hazard and static-0 hazard.

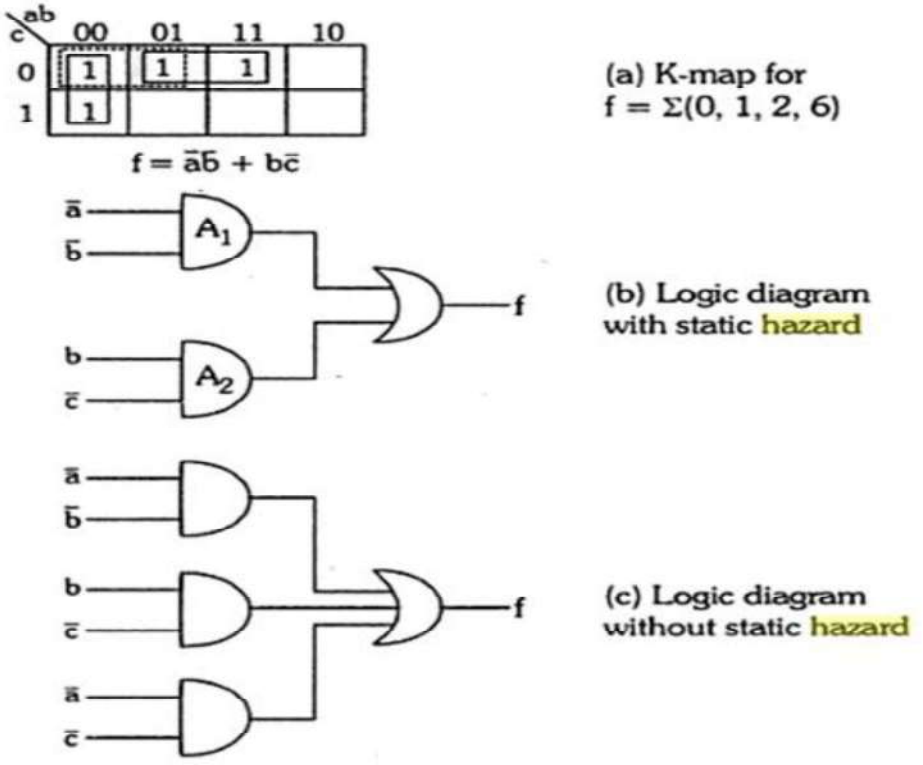
Static-1 hazard: If the output momentarily goes to state '0' when the output is expected to remain in state '1', as per the steady state analysis.

Static-0 hazard: If the output momentarily goes to state '1' when the output is expected to remain in state '0' as per the steady state analysis.

From below diagram, when abc = 000, the output f =1 due to 1 output from the AND Gate 1(A-). Now, if the input b is changes from 0 to 1, causes abc = 000 to 010, the output f should be in 1 due to 1 at the output of Lower AND gate (A2). Due to change in the value of input variable, the output f is supported to remain in HIGH state. But, due to unequal propagation delay, if the A_1 gate output changes to 0 shortly before A_2 gate output becomes '1', then during this short period, the output f = 0 momentarily. This situation is called static hazard.

A static hazard can be removed by covering the adjacent cells with a redundant grouping that overloops both grouping. For the above circuit, static hazard can be eliminated by covering adjacent cells (000, 010 shown in dotted lines). This gives a term $\bar{a}c$, which overloops both ab and bcgroupings.

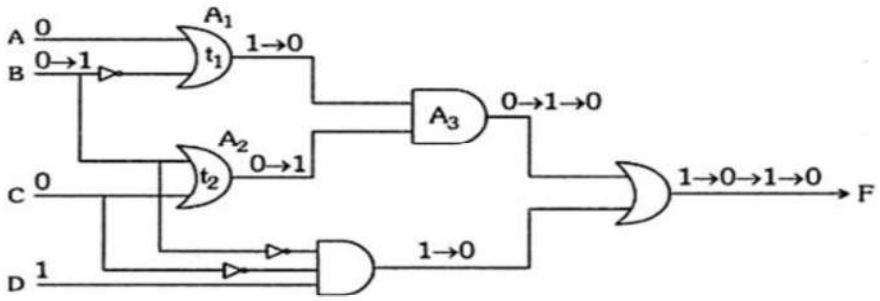
Now, if the input (abc) changes 000 to 010, the output will remain at '1' state, because the output of new AND gate is high for both input combinations. Here, though, there is a change in propagation delay, since the lower AND gate output is 1, the output of the circuit is 1 only, without any hazard.



Note: While designing a static hazard free circuit, all adjacent input combinations, having some output occur with some subcube of the corresponding function i.e, every pair of adjacent 1 cells and every pair of adjacent '0' cells in the K-map of a swithching function should be covered by at least one subcube.

5.14.2 Dynamic hazard.

When the output is supported to change from 0 to 1 or from 1 to 0, the circuit may go three or more transients and produce more than one spike or glitch. Such multiple glitch situation is called of "dynamic hazards". Dynamic hazards can be eliminated by some method of used in static hazards. Here multiple outputb transitions can occur bif there are multiple paths with different delays from the changing inputs to the changing output.



That has three different paths from input A to the oputput F. One of the path goes through OR(A_1) gate with delay "t_1"(slow) and other path is through OR gate (A2) with propagation delay 't_2 '(slower). If the input is (ABCD) = 0001 then the output is 1. If the change in the input (i.e. B is from 0 tio 1), then the input is 0101, the O/P AND (A_3) gate is changes from 0 to 1 and then, 0, because the difference in propagation delay of two OR gate (A_1 and A_2). Finally the O/P is changed from 1 to 0, with the glitch change.

Note: (1)Dynamic hazads can be eliminated properly by designing a two-level AND-OR (or) OR-AND logic circuits.

(2)The variables and its complements are not connected to the some first-level gate.

5.14.3Essential Hazrds:

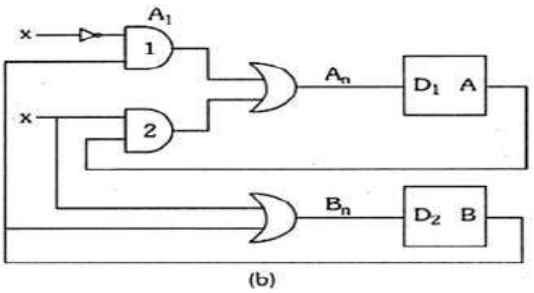
The essential hazard is atype of hazard that exists only in asynchronous sequential circuits with two or more feedback. Generally, it occur in toggling type circuits. It is an operational error generally caused by an excessive delay to a feedback variable in response to an input change, leadge a transition into an unwanted state. This type of hazard cannot be eliminated by adding any redundant gates, but they can be eliminated by adjusting the amount of delay in the affected path. So, the core must be taken while designing the feedback path delay, so that the delay is long enough compared to the delay of the other signals that orginate from the input terminals.

Elimination of Essential Hazards.

Consider the figure PS and NS table and itys corresponding sequential circuit with DFF.

PS	NS(A_nB_n)	
AB	x = 0	x = 1
00	00	01
01	11	01
11	11	11

(a)



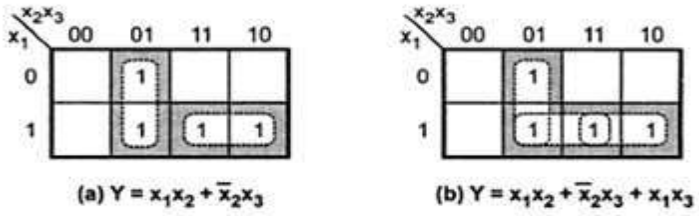
(b)

In the above circuit, the A_n depend on **x** and B_n depends on x. Assume, the delay associated with NOT gate is very large compared to the combined propagation delay of FFS and Gates. If the PS of the circuit is 00, where x changes from '0 to 1', then B_n becomes 1 and there by A becomes 1. This B is feedback to the input of the flight FF through AND gate 1. If the change in input is not propagated through NOT gate, then An becomes 1 and there by A also becomes 1. Hence the NS = A_nB_n = 11 instead of 01 of given in the table. This is called of "Essential Hazard", which occuring due to the result of the input change and B is changing faster.

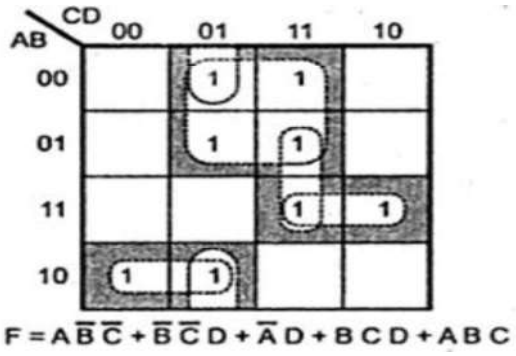
This essential hazard can be eliminated by (1) Reducing the delay of NOT gate or (2) Increasing the delay in the feedback path of B to AND gate 1.

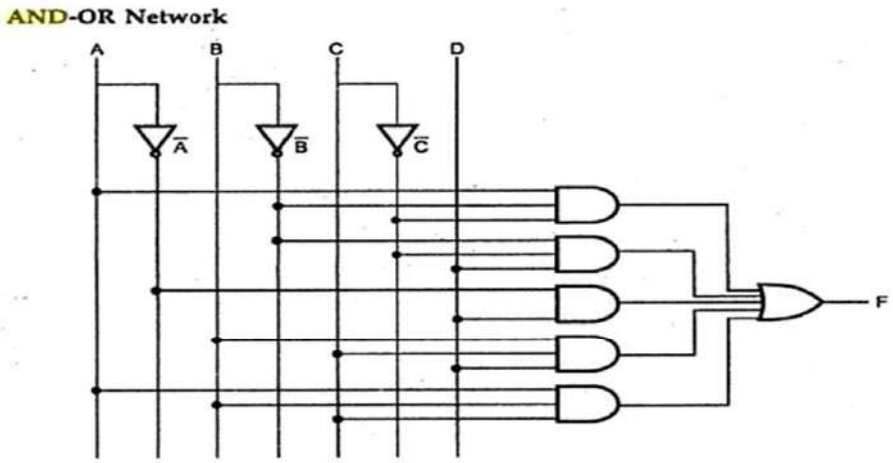
5.14.4 Eliminating a Hazard

The hazard exists because of the change of input results in a differenrt product terms covering two midterms or different sum terms covering two maxterms. Whenever the circuit move from one product term to another or move one sum term to another, there is a possibility of a momentary interval when neither term is equal to 1, giving rise to an undesirable 0 output. Hazards can be eliminated by enclosing two minterms or maxterms inquestion. For Example, it the circuit has minterms $X_1X_2 + X_2X_3$, then these two minterms must be enclosed by introducing another minterm X_1X_3.



Example: Implement the switching function F=∑(1,3,5,7,8,9,14,15) by a static hazard free two level AND-OR gate network.





5.15 Design of Combinational and Sequential circuits using VERILOG.

5.15.1Operators in Verilog HDL

Verilog HDL includes following kinds of operators

Boolean logical

Unary reduction logical

Bitwise logical

Relational

Binary arithmetic

Unary arithmetic

Other Boolean logical operators

Logical operators operate on logical operands and return a logical value,ie.,TRUE(1) or FALSE(0). Used typically in if and while statements. Do not confuse logical operators with the bitwise boolean operators. For example ! is a logicl NOT and ~ is a bitwise NOT. The first negates,eg.,!(5==6) is TRUE. The second complements the bits .eg.,~{1,0,1,1} is 0100.

Operator	Name
!	Logical negation
&&	Logical AND
||	logical OR

Unary reduction logical operators

Unary reduction logical operators operate on a single operand. They produce a single bit result from applying the operator to all of the bits of the operand. FOR EXAMPLE in statement B=&A if A=1101, THEN B=(1&1&0&1)=0

Operator	Name
&	AND reduction
|	OR reduction
^	XOR reduction

~&	NAND reduction
~|	NOR reduction
~^	XNOR reduction
Operator	Operation
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~&	Bitwise NAND
~|	Bitwise NOR
~	Bitwise negation
~^ or ^~	Equivalence bitwise NOT XOR

Bitwise logical operators

Bitwise operators operate on the bits of the operand or operands. The result of A&B is the AND of each corresponding bit of A with B. For example if A=1011 and B=0101 then C=0001. Except for bitwise negation these operators operate on a two operands.

Relational operator

Relational operators compare two operands and return a logical value ie.,TRUE(1) or FALSE(0). For example if A=0100 and B=0100 then statement if (A==B) results True(1) If any bit is unknown the relation is ambiguous and the result is unknown(X).

Opertor	**Operation**
>	Greater than
>=	Greater than or equal
<	less than
<=	less than or equal
==	Logical equality
!=	Logical inequality

Binary arithmetic operators

Binary arithmetic operators operate on two operands. Register and net ie.,wire,operands are treated as unsigned. However real and imaginary operands may be signed. If any bit of an operand is unknown (x) then the result is unknown.

Operator	**Operation**	**Comments**
+	Addition	
-	Subtraction	
*	Multiplication	
/	Division	Divide by zero produces an (x)
%	Modulus	

Unary arithmetic operators

Operator	**Operation**	**Comments**

-	Unary minus	Changes sign of its operand

Other operators

The conditional operator operates much like in the language C

Operator	Operation
===	Case equality
!==	Case inequality
{,}	
<<	Shift left
>>	Shift right
?:	Conditional

Other precedence

The precedence of operators is shown in table. The top of the table is the highest precedence and bottom is the lowest. Operators on the same line have the same precedence and associate left to right in an expression.

Type	Operators
Unary operators	! & ~& | ~| ^ ~^ + - (Highest precedence)
Multiplying operators	* / %
Sign operators	+ -
Relational operators	<<>> <<= >>= == != === ~==
Logical operators	& ~& ^ ~^ | ~| && ||
Conditional operators	?:(lowest precedence)

5.15.1Structure of the data flow description

Data flow describes how the circuit signals flow from the inputs to the outputs. There are some concurrent statements which allow to describe the circuit in terms of operations on signals and flow of signals in the circuit. When such concurrent statements are used in a program, the style is called a dataflow design.

Verilog HDL code for full adder in data flow description

```
modulefull_add(A,B,Cin,Cout,Sum);
inputA,B,Cin;
outputSum,Cout;
```

```
        assign Sum=(A^B)^Cin;
        assignCout=(A&B)|(Cin&A)|(Cin?&B);
endmodule
```

Verilog HDL code for multiplexer in data flow description

```
module mux 2*1(D0,D1,S,Enbar,Y);
input D0,D1,S,Enbar;
outputY;
wire I1,I2,I3,I4;
        assign #10Y=I3|I4;
        assign #10 I3=D0&I1&I2;
        assign #10 I4=D1&S&I2;
        assign #10 I1=~S;
        assign #10 I2=~Enbar;
endmodule
```

5.15.2Structure of the behavioral description

It is sometimes possible to directly describe the behavior or the functionality of a circuit. Such a modeling style is called behavioral modeling which is very similar in syntax and semantics to that of a high level programming language.

Verilog HDL code for full adder in behavioral description

```
modulefull_add(A,B,Cin,Cout,Sum);
inputA,B,Cin;
outputSum,Cout;
regSum,Cout;
always @(A,B,Cin)
        begin
         Sum=(A^B)^Cin;
        Cout=(A&B)|(Cin&A)|(Cin?&B);
        end
endmodule
```

Verilog HDL code for full subtractor inbehavioral description

```
modulefull_sub(A,B,Bin,Bout,D);
inputA,B,Bin;
outputD,Bout;
regD,Bout;
always @(A,B,Bin)
        begin
```

```
            D=A^B^Cin;
            Bout=(~A&B)|(~Bin&A)|(Bin^B);
           end
endmodule
```

Verilog HDL code for half subtractor inbehavioral description

```
modulehalf_sub(A,B,Bout,D);
input A,B;
outputD,Bout;
regD,Bout;
always @(A,B)
           begin
            D=A^B;
            Bout=(~A)&B;
           end
endmodule
```

Verilog HDL code for 2*1 multiplexer in behavioral description

```
module mux 2*1(D0,D1,S,Enbar,Y);
input D0,D1,S,Enbar;
outputY;
reg Y;
always @(S,D0,D1,Enbar)
           begin
           if(Enbar= =1)
                      Y=1'bz;
           else
                      begin
                      if(S)
                                 Y=D1;
           else
                                 Y=D0;
                                 end
           end
endmodule
```

Verilog HDL code for JK flipflop inbehavioral description

```
module JK_FF(J,K,CLK,Q,Q');
input J,K;
input CLK;
output    Q,Q';
reg Q;
```

```
assign Q'=~Q;
always @ (posedge CLK)
        case ({J,K})
        2'b00:Q<= Q;
        2'b01:Q<= 1'b0;
        2'b10:Q<= 1'b1;
        2'b11:Q<= ~Q;
        endcase
endmodule
```

Verilog HDL code for 4bit synchronous counter with parallel load

```
module  bin_ctr(Count,Load,I,CLK,CLR,A,CO);
input Count, Load;
input CLK,CLR;
input [3:0]I;
output[3:0]A;
output CO;
reg [3:0]A;
assign CO=Count &~Load & (A= = 4'b1111);
always @ (posedge CLK or negedge CLR)
        if (~CLR)A   <=4'b0000;
else if (Load)A <=I;
else if (Count)A <=A+1'b1;
else A<= A;
endmodule
```

5.15.3Gate level/ Structural description

In structural design verilog uses components or gates to model the system.

Verilog HDL code in structural description of a half adder

```
modulehalf_add(A,B,Sum,Cout);
input A,B;
output Sum, Cout;
        xor X1(Sum,A,B);
        and A1 (Cout,A,B);
endmodule
```

Verilog HDL code in structural description of a 2*4 decoder with enable input

```
module  decoder2*4(A,B,En,Y);
input A,B;
input En;
```

```
output [3:0] Y;
wireAbar,Bbar;
        not (Abar,A);
        not (Bbar,B);
        and (Y0,Abar,Bbar,En);
        and (Y1,Abar,B,En);
        and (Y2,A,Bbar,En);
        and (Y3,A,B,En);
endmodule
```

Verilog HDL code in structural description of a full adder with two half adder

```
modulefull_add(A,B,Cin,Sum,Cout);
inputA,B,Cin;
outputSum,Cout;
wire S0,C0,C1;
full adder
        HA H1(A,B,S0,C0);
        HA H2(S0,Cin,Sum,C1);
        Or (Cout,C0,C1);
Endmodule
```

5.16Two marks questions with answers

1. **What are secondary variables?**
   Present state variables in asynchronous sequential circuits
2. **What are excitation variables?**
   Next state variables in asynchronous sequential circuits
3. **What is fundamental mode sequential circuit?**
   -input variables changes if the circuit is stable
   -inputs are levels, not pulses
   -only one input can change at a given time
4. **What is pulse mode circuit?**
   -inputs are pulses
   -width of pulses are long for circuit to respond to the input
   -pulse width must not be so long that it is still present after the new state is reached
5. **What are the significance of state assignment?**
   In synchronous circuits-state assignments are made with the objective of circuit Reduction Asynchronous circuits-its objective is to avoid critical races

6. **When does race condition occur?**
   -two or more binary state variables change their value in response to the change in i/p Variable