EC 8392 – DIGITAL ELECTRONICS

UNIT – III : SYNCHRONOUS SEQUENTIAL CIRCUITS

CLASSIFICATION OF SYNCHRONOUS SEQUENTIAL CIRCUIT:

In synchronous or clocked sequential circuits, clocked Flip-Flops are used as memory elements, which change their individual states in synchronism with the periodic clock signal. Therefore, the change in states of Flip-Flop and change in state of the entire circuits occur at the transition of the clock signal.

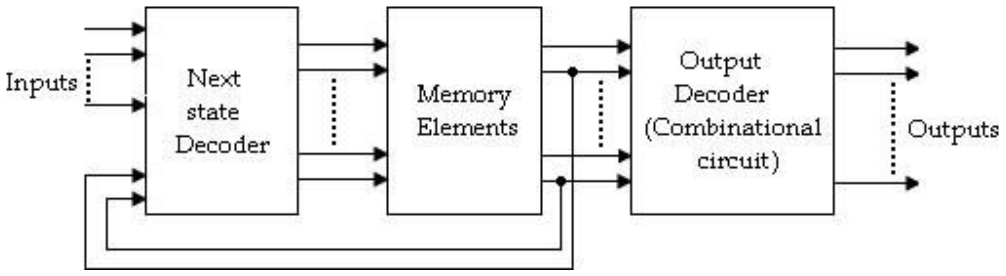
The synchronous or clocked sequential networks are represented by two models.

Moore model: The output depends only on the present state of the Flip-Flops.

Mealy model: The output depends on both the present state of the Flip-Flops and on the inputs.

Moore model:

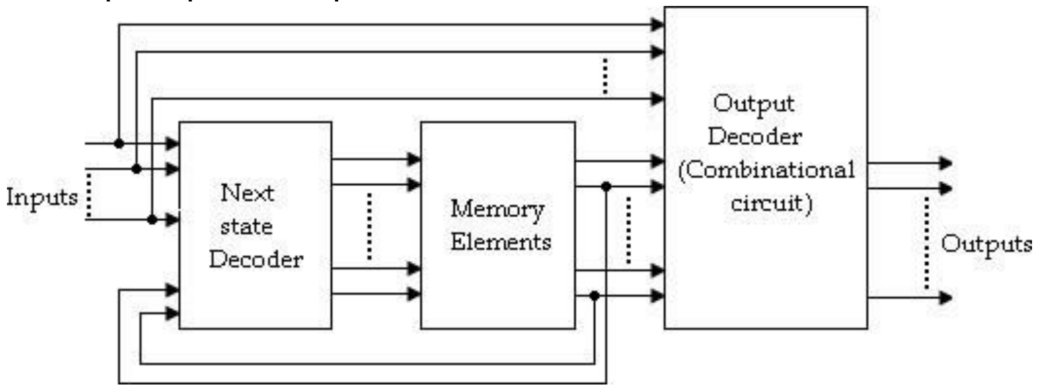
In the Moore model, the outputs are a function of the present state of the Flip- Flops only. The output depends only on present state of Flip-Flops, it appears only after the clock pulse is applied, i.e., it varies in synchronism with the clock input.



Moore model

Mealy model:

In the Mealy model, the outputs are functions of both the present state of the Flip-Flops and inputs.



Mealy model

Difference between Moore and Mealy model

Sl.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	It requires more number of states for implementing same function.	It requires less number of states for implementing same function.

ANALYSIS OF SYNCHRONOUS SEQUENTIAL CIRCUIT:

The behavior of a sequential circuit is determined from the inputs, outputs and the state of its Flip-Flops. The outputs and the next state are both a function of the inputs and the present state. The analysis of a sequential circuit consists of obtaining a table or diagram from the time sequence of inputs, outputs and internal states.

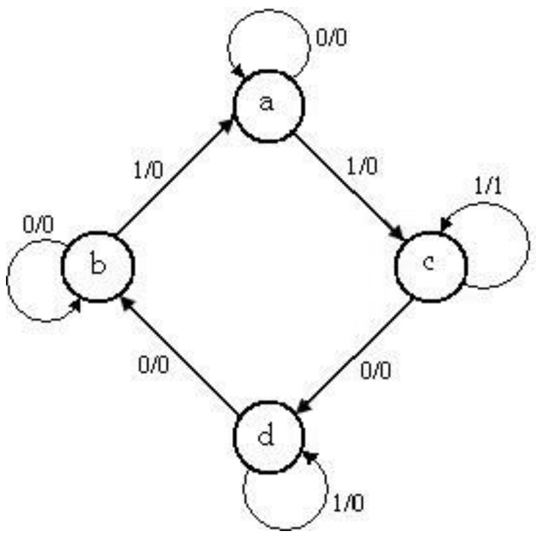
Before going to see the analysis and design examples, we first understand the state diagram, state table.

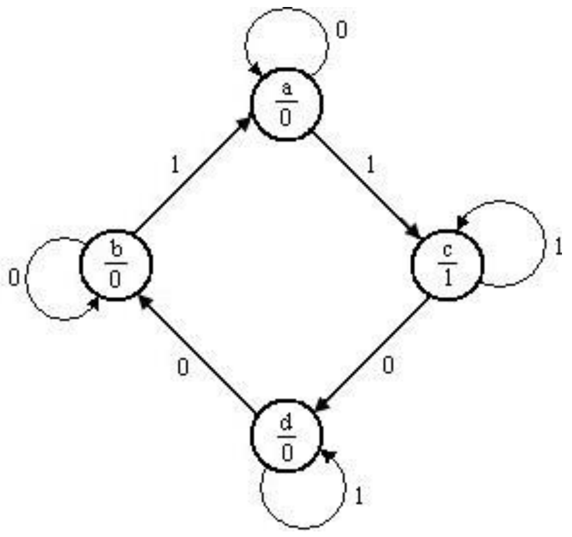
State Diagram

State diagram is a pictorial representation of a behavior of a sequential circuit.

- In the state diagram, a state is represented by a circ e and the transition between states is indicated by directed lines connecting the circles.
- A directed line connecting a circle with circle with itself indicates that next state is same as present state.
- The binary number inside each circle identifies the state represented by the circle.
- The directed lines are labeled with two binary numbers separated by a symbol '/'. The input value that causes the state transition is labeled first and the output value during the present state is labeled after the symbol '/'.

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





State diagram for Mealy circuit **State diagram for Moore circuit**

State Table

State table represents relationship between input, output and Flip-Flop states.

It consists of three sections labeled present state, next state and output.

- The present state designates the state of Flip-Flops before the occurrence of a clock pulse, and the output section gives the values of the output variables during the present state.
- Both the next state and output sections have two columns representing two possible input conditions: X= 0 and X=1.

Present state	**Next state**		**Output**	
	X= 0	**X= 1**	**X= 0**	**X= 1**
AB	AB	AB	Y	Y
a	a	c	0	0

b	b	a	0	0
c	d	c	0	1
d	b	d	0	0

- In case of Moore circuit, the output section has only one column since output does not depend on input.

Present state	**Next state**		**Output**
	X= 0	**X= 1**	**Y**
AB	AB	AB	
a	a	c	0
b	b	a	0
c	d	c	1
d	b	d	0

State Equation

It is an algebraic expression that specifies the condition for a Flip-Flop state transition.

The Flip-Flops may be of any type and the logic diagram may or may not include combinational circuit gates.

ANALYSIS PROCEDURE

The synchronous sequential circuit analysis is summarizes as given below:

1. Assign a state variable to each Flip-Flop in the synchronous sequential circuit.

2. Write the excitation input functions for each Flip-Flop and also write the Moore/ Mealy output equations.
3. Substitute the excitation input functions into the bistable equations for the Flip-Flops to obtain the next state output equations.
4. Obtain the state table and reduced form of the state table.
5. Draw the state diagram by using the second form of the state table.

Analysis of Mealy Model

1.A sequential circuit has two JK Flip-Flops A and B, one input (x) and one output (y). the Flip-Flop input functions are,

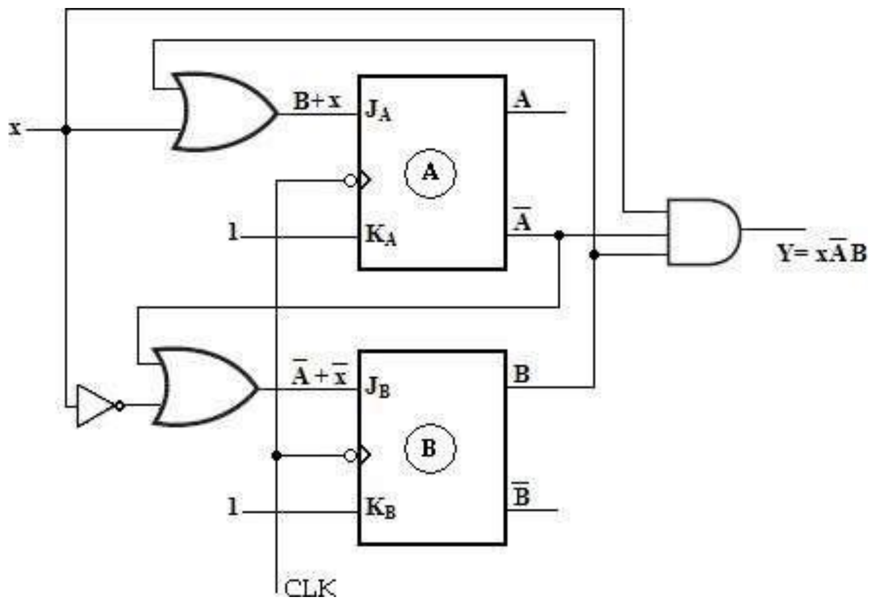
J_A= **B+ x** J_B= **A'+ x'**

K_A= **1** K_B= **1**

and the circuit output function, **Y= xA'B**.

a) Draw the logic diagram of the Mealy circuit,
b) Tabulate the state table,
c) Draw the state diagram.

Soln:



State table:

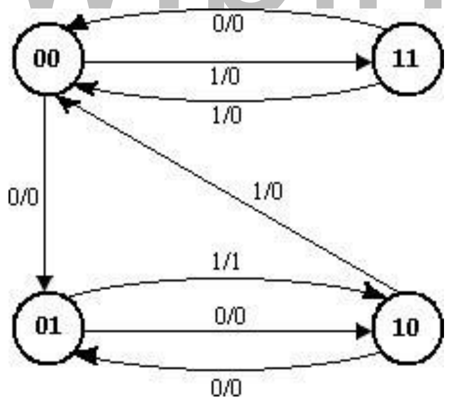
To obtain the next-state values of a sequential circuit with JK Flip-Flops, use the JK Flip-Flop characteristics table

Present state		Input	Flip-Flop Inputs				Next state		Output
A	B	x	J_A= B+ x	K_A= 1	JB= A'+ x'	K_B= 1	A(t+1)	B(t+1)	Y= xA'B
0	0	0	0	1	1	1	0	1	0
0	0	1	1	1	1	1	1	1	0
0	1	0	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0	1	0
1	0	1	1	1	0	1	0	0	0
1	1	0	1	1	1	1	0	0	0
1	1	1	1	1	0	1	0	0	0

Present state		Next state				Output	
		x=0		x=1		x=0	x=1
A	B	A	B	A	B	y	y
0	0	0	1	1	1	0	0
0	1	1	0	1	0	0	1
1	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0

Second form of state table

State Diagram:



State Diagram

2. A sequential circuit with two 'D' Flip-Flops A and B, one input (x) and one output (y). the Flip-Flop input functions are:

   D_A= Ax+ Bx

   D_B= A'x and the circuit output function is,

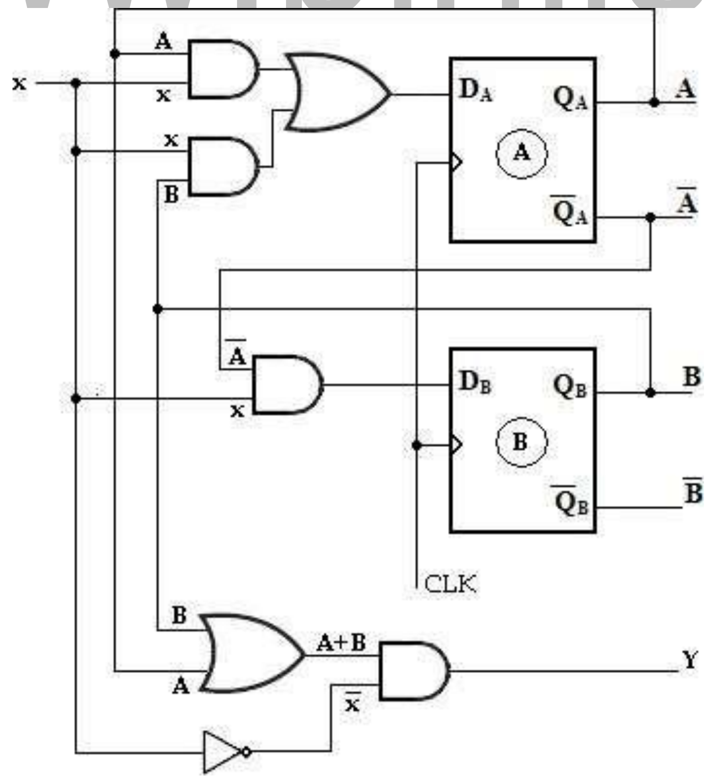
   Y= (A+ B) x'.

   (a) Draw the logic diagram of the circuit,

   (b) Tabulate the state table,

   (c) Draw the state diagram.

Soln:



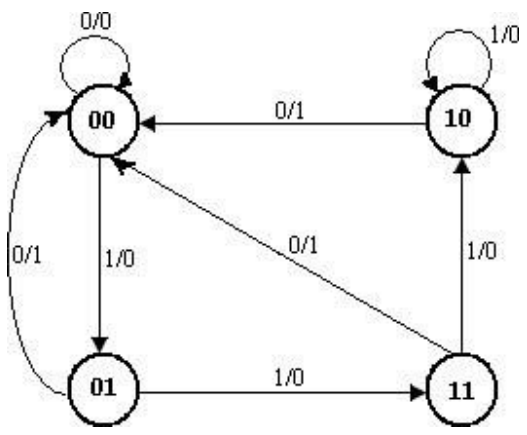
State Table:

Present state		Input	Flip-Flop Inputs		Next state		Output
A	B	x	D_A= Ax+Bx	DB= A'x	A(t+1)	B(t+1)	Y= (A+B)x'
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

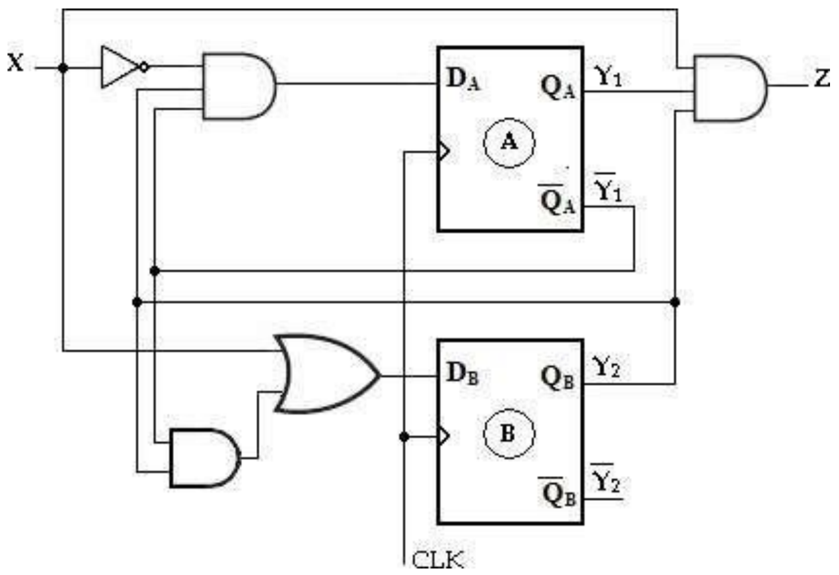
Present state		Next state				Output	
		x = 0		x= 1		x= 0	x= 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Second form of state table

State Diagram:



3. Analyze the synchronous Mealy machine and obtain its state diagram.



Soln:

The given synchronous Mealy machine consists of two D Flip-Flops, one inputs and one output.

The Flip-Flop input functions are,

DA= Y1'Y2X'

D_B= X+ Y_1'Y_2

The circuit output function is, **Z=**

Y_1Y_2X State Table:

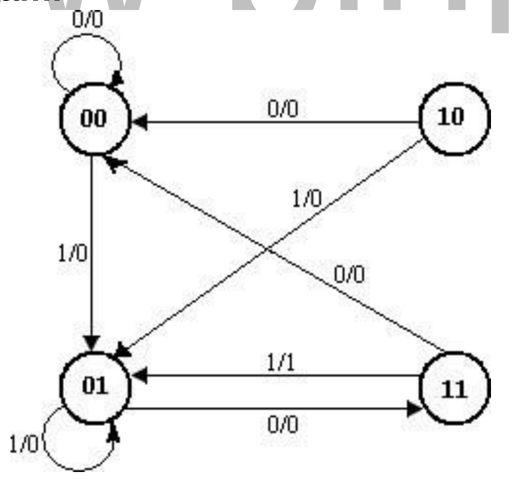
Present state		**Input**	**Flip-Flop Inputs**		**Next state**		**Output**
Y 1	**Y 2**	**X**	**DA= Y1'Y2X'**	**D_B= X+ Y_1'Y_2**	**Y_1 (t+1)**	**Y_2 (t+1)**	**Z= Y_1Y_2X**
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

Present state		**Next state**				**Output**	
		X= 0		**X = 1**		**X= 0**	**X= 1**
Y1	**Y2**	**Y1**	**Y2**	**Y 1**	**Y2**	**Z**	**Z**

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

Second form of state table

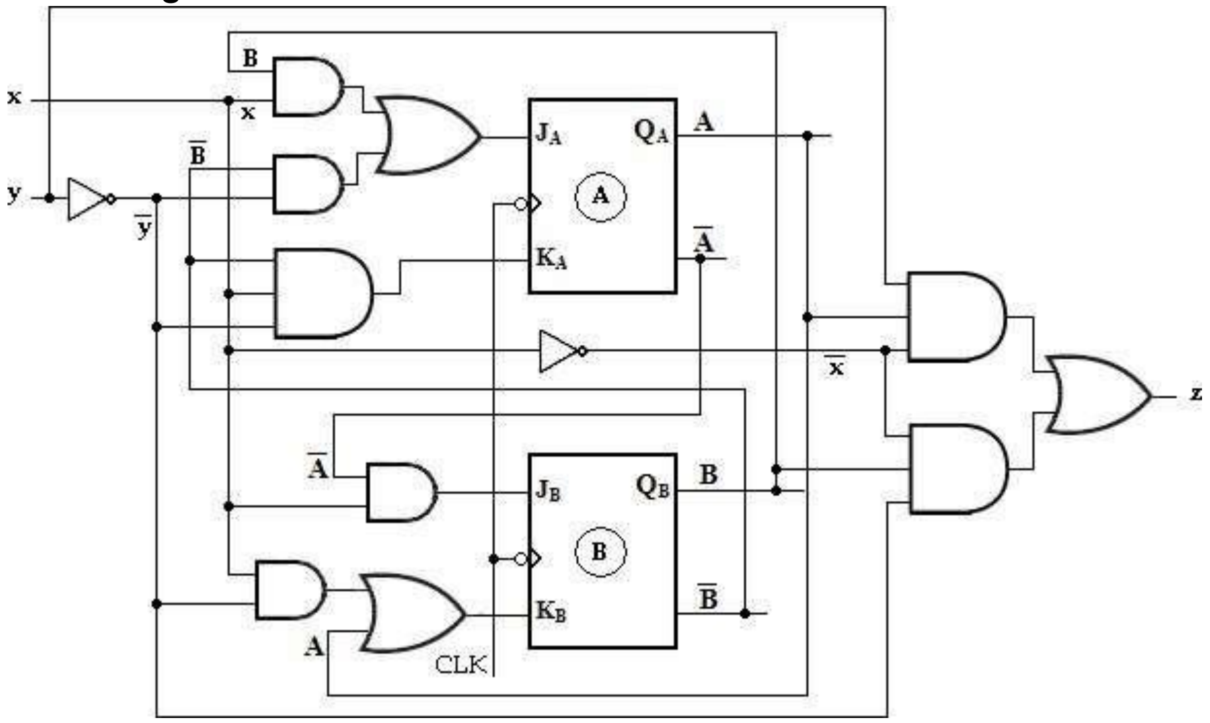
State Diagram:



4. A sequential circuit has two JK Flop-Flops A and B, two inputs x and y and one output z. The Flip-Flop input equation and circuit output equations are **J_A = Bx + B' y' K_A = B' xy' J_B = A' x K_B = A+ xy'** , **z = Ax' y' + Bx' y'**

(a) Draw the logic diagram of the circuit (b) Tabulate the state table. (c) Derive the state equation.

State diagram:

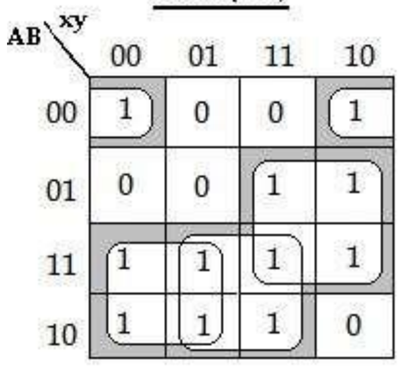


Present state		Input		Flip-Flop Inputs				Next state		Output
A	B	x	y	J_A= Bx+B'y'	K_A= B'xy'	J_B= A'x	K_B= A+xy'	A(t+1)	B(t+1)	z
0	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	0
0	0	1	1	0	0	1	0	0	1	0

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

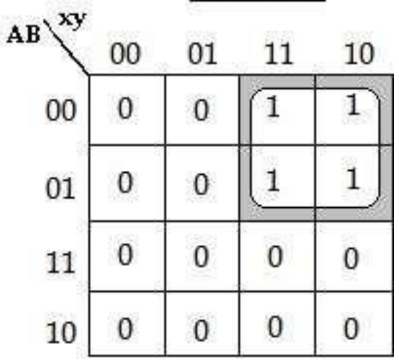
State Equation:

For A(t+1)



A (t+1) = Ax' + Ay + Bx + A'B'y'

For B(t+1)



B (t+1) = A'x

5. A sequential circuit has two JK Flip-Flop A and B. the Flip-Flop input functions are: **J_A= B J_B= x'**

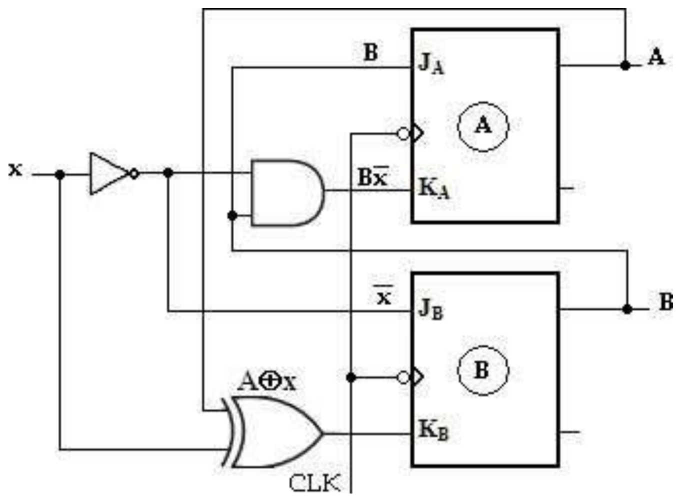
   K_A= Bx' **K_B= A x.**

(a) Draw the logic diagram of the circuit,

(b) Tabulate the state table,

(c) Draw the state diagram.

Logic diagram:



The output function is not given in the problem. The output of the Flip-Flops may be considered as the output of the circuit.

State table:

To obtain the next-state values of a sequential circuit with JK Flip-Flop, use the JK Flip-Flop characteristic table.

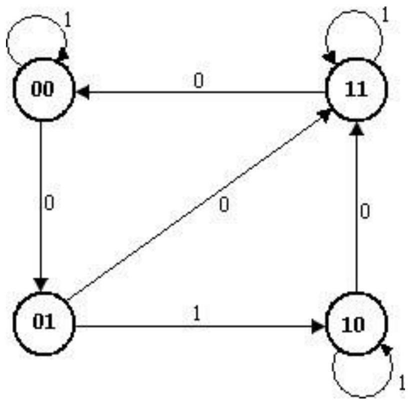
Present state		**Input**	**Flip-Flop Inputs**				**Next state**	
A	**B**	**x**	**J_A= B**	**KA= Bx'**	**JB= x'**	**K_B= A x**	**A(t+1)**	**B(t+1)**
0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0
0	1	0	1	1	1	0	1	1

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

Present state		**Next state**			
		X= 0		**X= 1**	
A	**B**	**A**	**B**	**A**	**B**
0	0	0	1	0	0
0	1	1	1	1	0
1	0	1	1	1	0
1	1	0	0	1	1

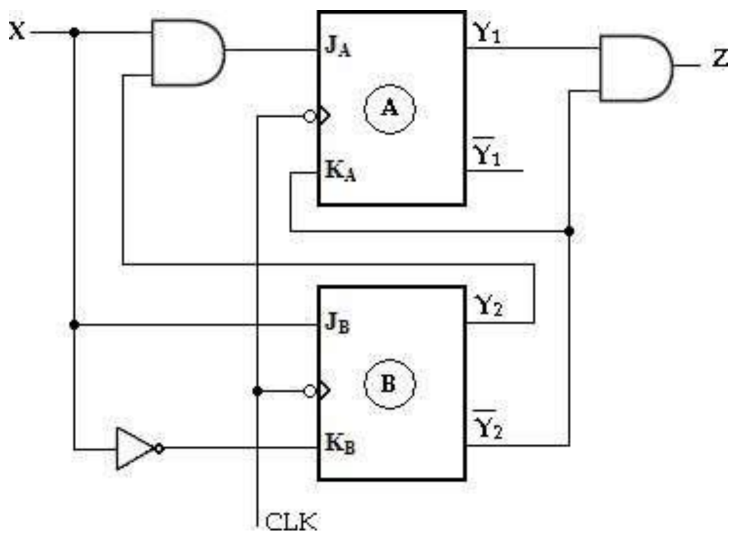
Second form of state table

State Diagram:



Analysis of Moore Model

6. Analyze the synchronous Moore circuit and obtain its state diagram.



Soln:

Using the assigned variable Y_1 and Y_2 for the two JK Flip-Flops, we can write

the four excitation input equations and the Moore output equation as follows:

$J_A = Y_2X$; $K_A = Y_2'$

$J_B = X$; $KB = X'$ and output function, **$Z = Y_1Y_2'$**

State table:

Present state		**Input**	**Flip-Flop Inputs**				**Next state**		**Output**
Y1	**Y2**	**X**	**$J_A = Y_2X$**	**$K_A = Y_2'$**	**$J_B = X$**	**$KB = X'$**	**Y_1 (t+1)**	**Y_2 (t+1)**	**$Z = Y_1Y_2'$**
0	0	0	0	1	0	1	0	0	0
0	0	1	0	1	1	0	0	1	0

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

Present state		**Next state**				**Output**
		X = 0		**X= 1**		**Y**
Y1	**Y2**	**Y1**	**Y2**	**Y1**	**Y2**	
0	0	0	0	0	1	0
0	1	0	0	1	1	0
1	0	0	0	0	1	1
1	1	1	0	1	1	0

Second form of state table

State Diagram:

Here the output depends on the present state only and is independent of the input. The two values inside each circle separated by a slash are for the present state and output.

7. A sequential circuit has two T Flip-Flop A and B. The Flip-Flop input functions are:

   T_A= Bx **T_B= x**

   y= AB

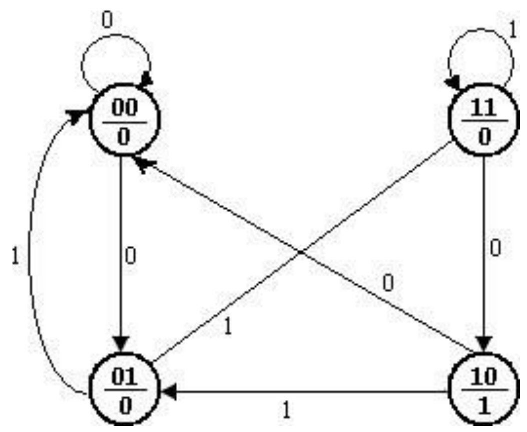
   (a) Draw the logic diagram of the circuit,

   (b) Tabulate the state table,

   (c) Draw the state diagram.

Soln:

Logic diagram:



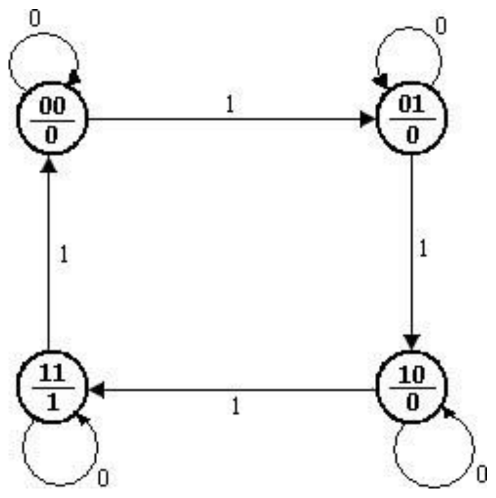
State table

Present state		Input	Flip-Flop Inputs		Next state		Output
A	B	x	T_A= Bx	T_B= x	A (t+1)	B (t+1)	y= AB
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	0	0
1	0	0	0	0	1	0	0
1	0	1	0	1	1	1	0
1	1	0	0	0	1	1	1
1	1	1	1	1	0	0	1

Present state		Next state				Output	
		x = 0		x= 1		x= 0	x= 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0
1	1	1	1	0	0	1	1

Second form of state table

State Diagram:



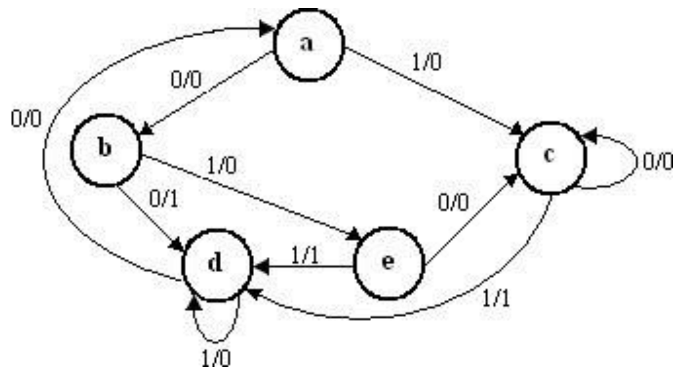
STATE REDUCTION/ MINIMIZATION

The state reduction is used to avoid the redundant states in the sequential circuits. The reduction in redundant states reduces the number of required Flip- Flops 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.

Since 'n' Flip-Flops produced 2^n state, a reduction in the number of states may result in a reduction in the number of Flip-Flops.

The need for state reduction or state minimization is explained with one example.



State diagram

Step 1: Determine the state table for given 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	1
d	a	d	0	0
e	c	d	0	1

State table

Step 2: Find equivalent states

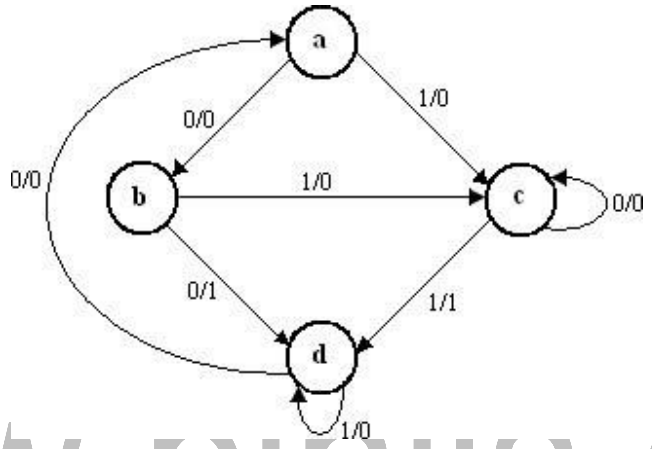
From the above state table **c** and **e** generate exactly same next state and same output for every possible set of inputs. The state **c** and **e** go to next states **c** and **d** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **e** can be removed and replaced by **c**. The final reduced state table is shown below.

Present state	**Next state**		**Output**	
	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

Reduced state table

The state diagram for the reduced table consists of only four states and is shown below.



Reduced state diagram

1.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	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

Soln: From the above state table **e** and **g** generate exactly same next state and same output for every possible set of inputs. The state **e** and **g** go to next states **a** and **f** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **g** can be removed and replaced by **e**.

The reduced state table-1 is shown below.

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

Reduced state table-1

Now states d and f are equivalent. Both states go to the same next state (e, f) and have same output (0, 1). Therefore one state can be removed; **f** is replaced by **d.**

The final reduced state table-2 is shown below.

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	**d**	0	0
d	e	**d**	0	1
e	a	**d**	0	1

Reduced state table-

2Thus 7 states are reduced into 5 states.

2.Determine a minimal state table equivalent furnished below

Present state	Next state	
	X=0	X=1
1	1, 0	1, 0
2	1, 1	6, 1
3	4, 0	5, 0
4	1, 1	7, 0
5	2, 0	3, 0
6	4, 0	5, 0
7	2, 0	3, 0

Soln:

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
1	1	1	0	0
2	1	6	1	1

3	4	5	0	0
4	1	7	1	0
5	2	3	0	0
6	4	5	0	0
7	2	3	0	0

From the above state table, **5** and **7** generate exactly same next state and same output for every possible set of inputs. The state **5** and **7** go to next states **2** and **3** and have outputs 0 and 0 for x=0 and x=1 respectively. Therefore state **7** can be removed and replaced by **5**.

Similarly, **3** and **6** generate exactly same next state and same output for every possible set of inputs. The state **3** and **6** go to next states **4** and **5** and have outputs 0 and 0 for x=0 and x=1 respectively. Therefore state **6** can be removed and replaced by **3**.

The final reduced state table is shown below.

Present state	**Next state**		**Output**	
	X= 0	**X= 1**	**X= 0**	**X= 1**
1	1	1	0	0
2	1	3	1	1
3	4	5	0	0
4	1	5	1	0
5	2	3	0	0

Reduced state table Thus 7 states are reduced into 5 states.

3.Minimize the following state table.

Present state	Next state	
	X=0	X=1
A	D, 0	C, 1
B	E, 1	A, 1
C	H, 1	D, 1
D	D, 0	C, 1
E	B, 0	G, 1
F	H, 1	D, 1
G	A, 0	F, 1
H	C, 0	A, 1
I	G, 1	H,1

Soln:

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
A	D	C	0	1
B	E	A	1	1
C	H	D	1	1
D	D	C	0	1
E	B	G	0	1
F	H	D	1	1
G	A	F	0	1

H	C	A	0	1
I	G	H	1	1

From the above state table, **A** and **D** generate exactly same next state and same output for every possible set of inputs. The state **A** and **D** go to next states **D** and **C** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **D** can be removed and replaced by **A**. Similarly, **C** and **F** generate exactly same next state and same output for every possible set of inputs. The state **C** and **F** go to next states **H** and **D** and have outputs 1 and 1 for x=0 and x=1 respectively. Therefore state **F** can be removed and replaced by **C**.

The reduced state table-1 is shown below.

Present state	**Next state**		**Output**	
	X=0	**X=1**	**X=0**	**X=1**
A	A	C	0	1
B	E	A	1	1
C	H	A	1	1
E	B	G	0	1
G	A	C	0	1
H	C	A	0	1
I	G	H	1	1

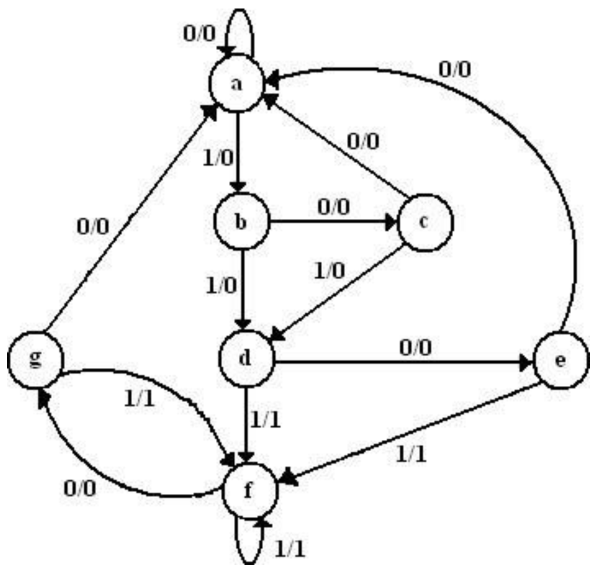
Reduced state table-1

From the above reduced state table-1, **A** and **G** generate exactly same next state and same output for every possible set of inputs. The state **A** and **G** go to next states **A** and **C** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **G** can be removed and replaced by **A**. The final reduced state table-2 is shown below.

Present state	**Next state**		**Output**	
	X= 0	**X= 1**	**X= 0**	**X= 1**
A	A	C	0	1
B	E	A	1	1
C	H	A	1	1
E	B	A	0	1
H	C	A	0	1
I	A	H	1	1

Reduced state table-

2Thus 9 states are reduced into 6 states. 4.Reduce the following state diagram.



Soln:

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

State table

From the above state table **e** and **g** generate exactly same next state and same output for every possible set of inputs. The state **e** and **g** go to next states **a** and **f** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **g** can be removed and replaced by **e**. The reduced state table-1 is shown below.

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

Reduced state table-1

Now states d and f are equivalent. Both states go to the same next state (e, f) and have same output (0, 1). Therefore one state can be removed; **f** is replaced by **d.**

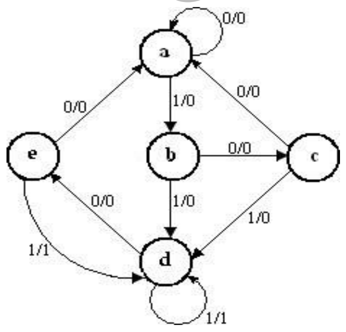
The final reduced state table-2 is shown below.

Present state	**Next state**		**Output**	
	X=0	**X=1**	**X=0**	**X=1**
a	a	b	0	0
b	c	d	0	0
c	a	**d**	0	0
d	e	**d**	0	1
e	a	**d**	0	1

Reduced state table-2

Thus 7 states are reduced into 5 states.

The state diagram for the reduced state table-2 is,

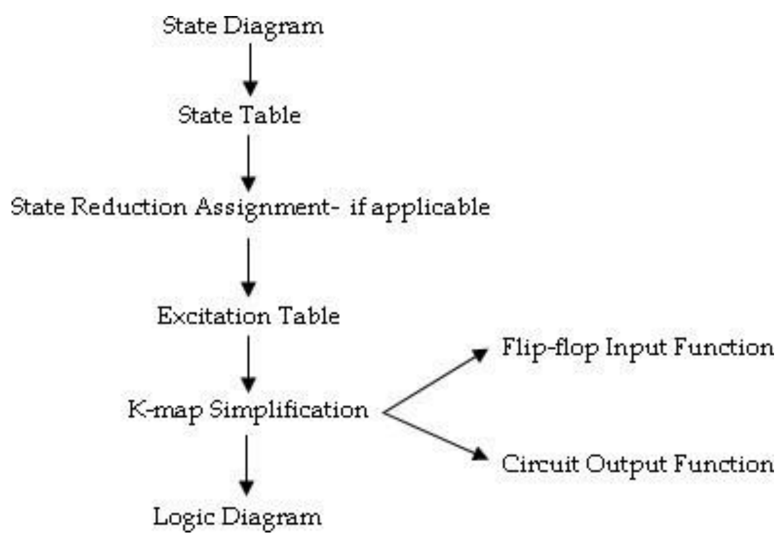


DESIGN OF SYNCHRONOUS SEQUENTIAL CIRCUITS:

A synchronous sequential circuit is made up of number of Flip-Flops and combinational gates. The design of circuit consists of choosing the Flip-Flops and then finding a combinational gate structure together with the Flip-Flops. The number of Flip-Flops is determined from the number of states needed in the circuit. The combinational circuit is derived from the state table.

Design procedure:

1. The given problem is determined with a state diagram.
2. From the state diagram, obtain the state table.
3. The number of states may be reduced by state reduction methods (if applicable).
4. Assign binary values to each state (Binary Assignment) if the state table contains letter symbols.
5. Determine the number of Flip-Flops and assign a letter symbol (A, B, C,…) to each.
6. Choose the type of Flip-Flop (SR, JK, D, T) to be used.
7. From the state table, circuit excitation and output tables.
8. Using K-map or any other simplification method, derive the circuit output functions and the Flip-Flop input functions.
9. Draw the logic diagram.



The type of Flip-Flop to be used may be included in the design specifications or may depend what is available to the designer. Many digital systems are constructed with JK Flip-Flops because they are the most versatile available. The selection of inputs is given as follows.

Flip-Flop	Application
JK	General Applications
D	Applications requiring transfer of data (Ex: Shift Registers)
T	Application involving complementation (Ex: Binary Counters)

Excitation Tables:

Before going to the design examples for the clocked synchronous sequential circuits we revise Flip-Flop excitation tables.

Present State	Next State	Inputs	
Qn	**Qn+1**	**S**	**R**
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Excitation table for SR Flip-Flop

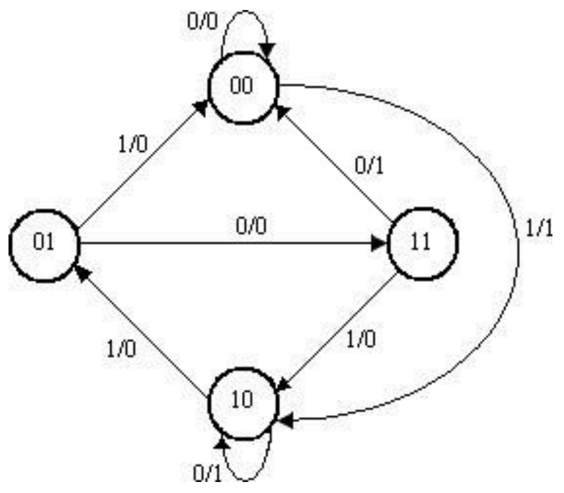
Present State	Next State	Inputs	
Qn	Qn+1	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Present State	Next State	Input
Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Present State	Next State	Input
Qn	Qn+1	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table for D Flip-Flop

1.A sequential circuit has one input and one output. The state diagram is shown below. Design the sequential circuit with a) D-Flip-Flops, b) T Flip-Flops, c) RS Flip-Flops and d) JK Flip-Flops.



Solution:

State Table:

The state table for the state diagram is,

Present state		**Next state**		**Output**	
		X= 0	**X= 1**	**X= 0**	**X= 1**
A	B	AB	AB	Y	Y
0	0	00	10	0	1
0	1	11	00	0	0
1	0	10	01	1	0
1	1	00	10	1	0

State reduction:

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 2 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:**

Excitation table:

Using the excitation table for T Flip-Flop, we can determine the excitation table for the
given circuit as,

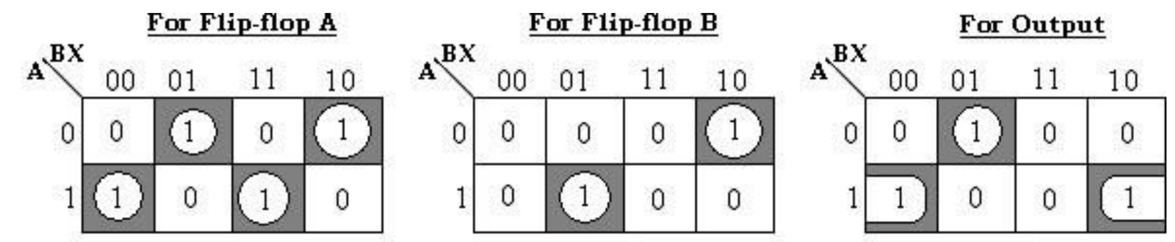
Present State	**Next State**	**Input**
Qn	**Qn+1**	**D**
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table for D Flip-Flop

Present state		Input	Next state		Flip-Flop Inputs		Output
A	B	X	A	B	DA	DB	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

Circuit excitation table

K-map Simplification:

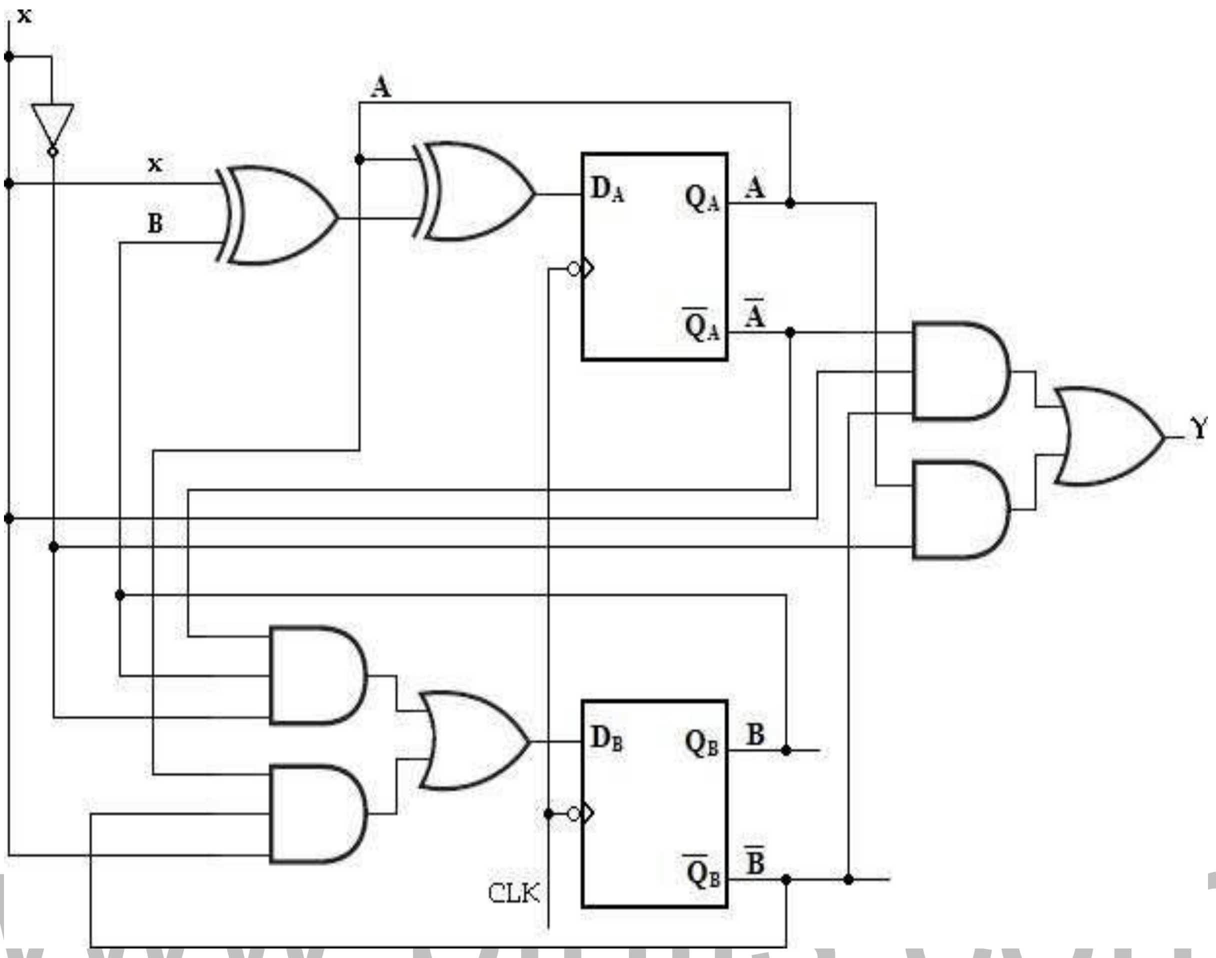


$D_A = A'B'X + A'BX' + ABX + AB'X'$
$= A \oplus (B \oplus x)$

$D_B = A'BX' + AB'X$

$Y = A'B'X + AX'$

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



Logic diagram of given sequential circuit using D

ii) Design using T Flip-Flops:

Using the n determine the

excitation

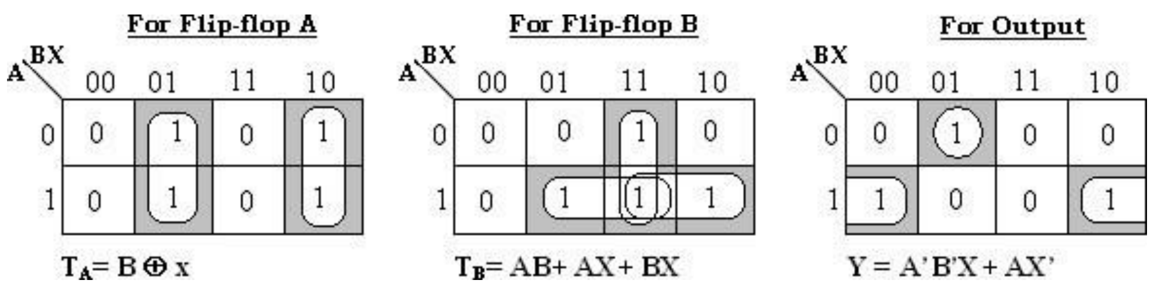
Present State	Next State	Input
Qn	**Qn+1**	**T**
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table for T Flip-Flop

Present state		Input	Next state		Flip-Flop Inputs		Output
A	B	X	A	B	TA	TB	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

Circuit excitation table

K-map Simplification:



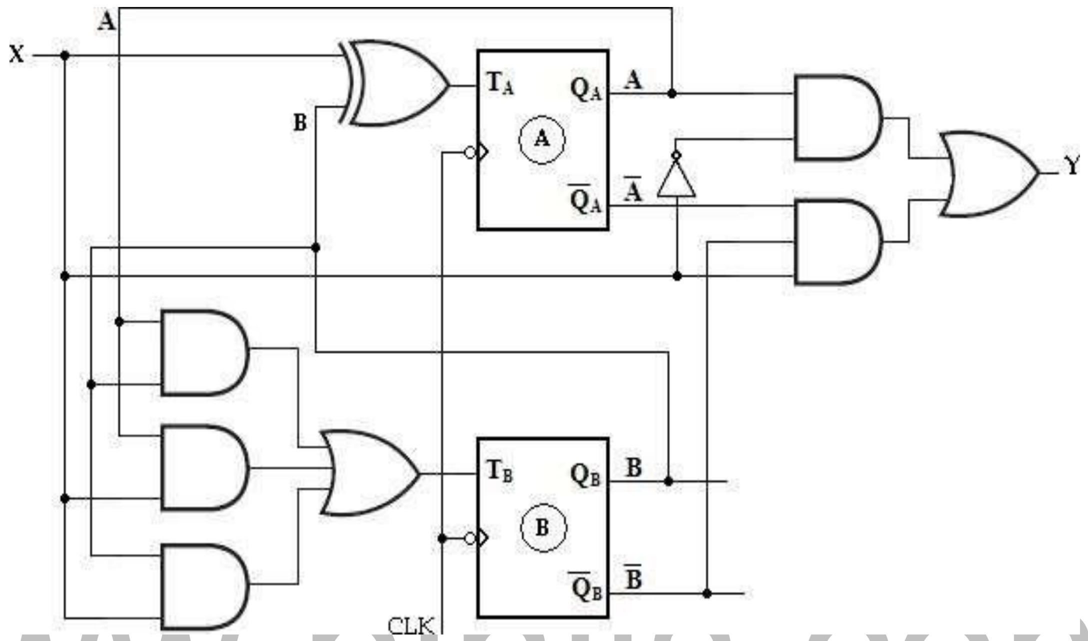
Therefore, input functions for,

T_A= B x and

T_B= AB+ AX+ BX

Circuit output function, **Y = XA'B'+ X'A**

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



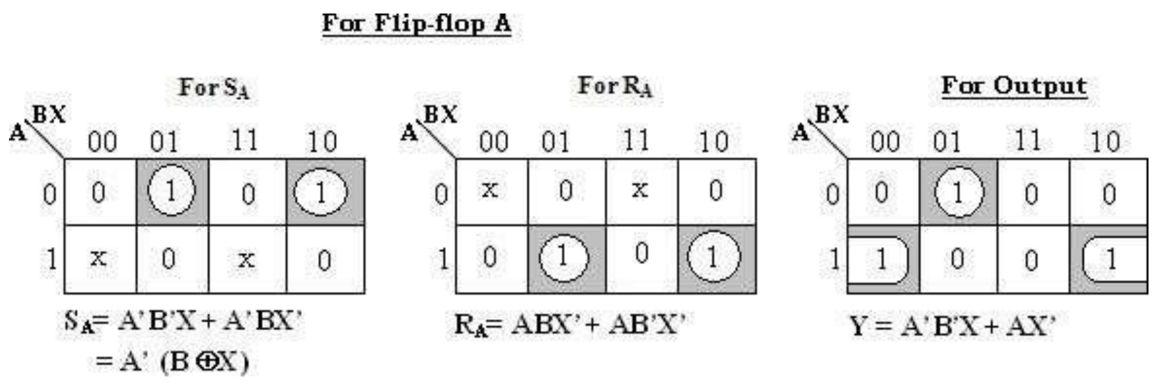
Present State	Next State	Inputs	
Qn	Qn+1	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Excitation table for SR Flip-Flop

Present state		Input	Next state		lip-Flop nputs				Output
A	B	X	A	B	SA	RA	SB	RB	Y
0	0	0	0	0	0	x	0	x	0
0	0	1	1	0	1	0	0	x	1
0	1	0	1	1	1	0	x	0	0
0	1	1	0	0	0	x	0	1	0
1	0	0	1	0	x	0	0	x	1
1	0	1	0	1	0	1	1	0	0
1	1	0	0	0	0	1	0	1	1
1	1	1	1	0	x	0	0	1	0

Circuit excitation

K-map



For Flip-flop B

For S_B

A \ BX	00	01	11	10
0	0	0	0	x
1	0	1	0	0

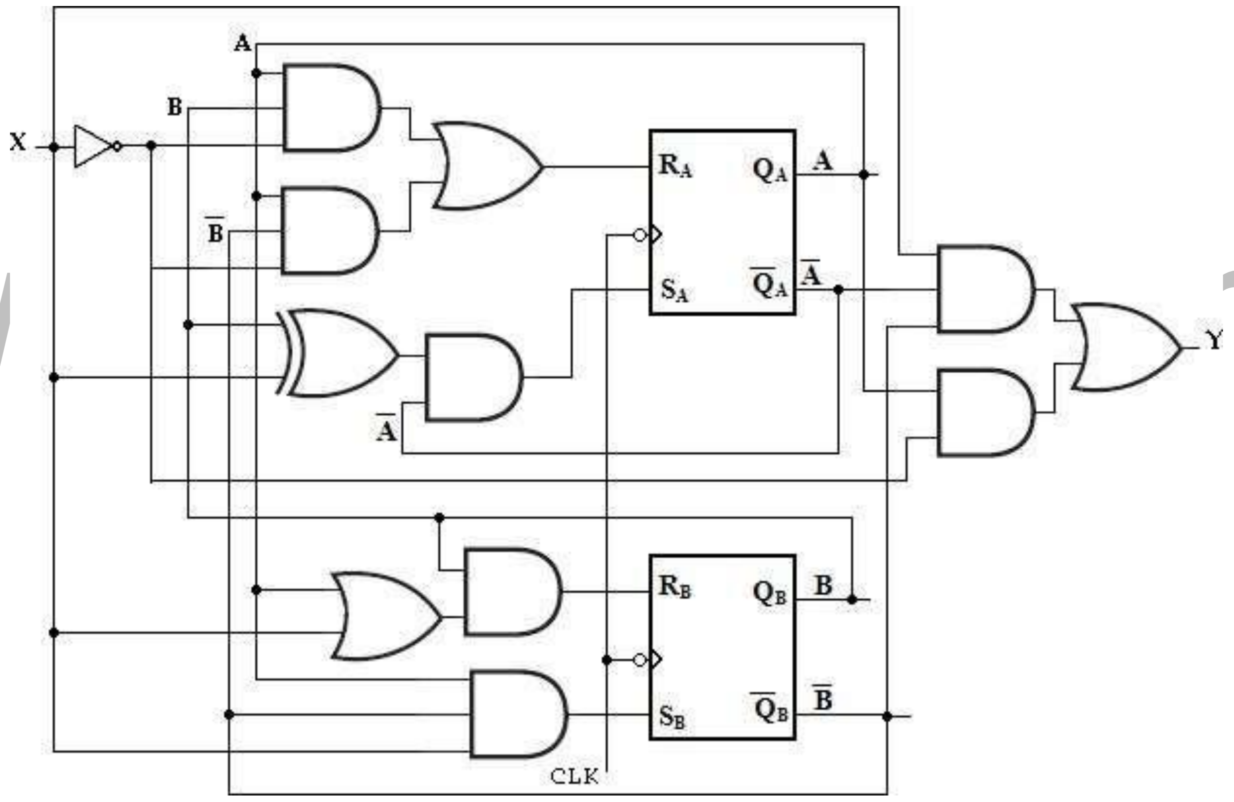
$S_B = AB'X$

For R_B

A \ BX	00	01	11	10
0	x	x	1	0
1	x	0	1	1

$R_B = AB + BX$

With these Flip-Flop input functions and circuit output function we can draw



iii Design using JK Flip-

Using the excitation table for JK Flip-Flop, we can determine the excitation table for the given circuit as,

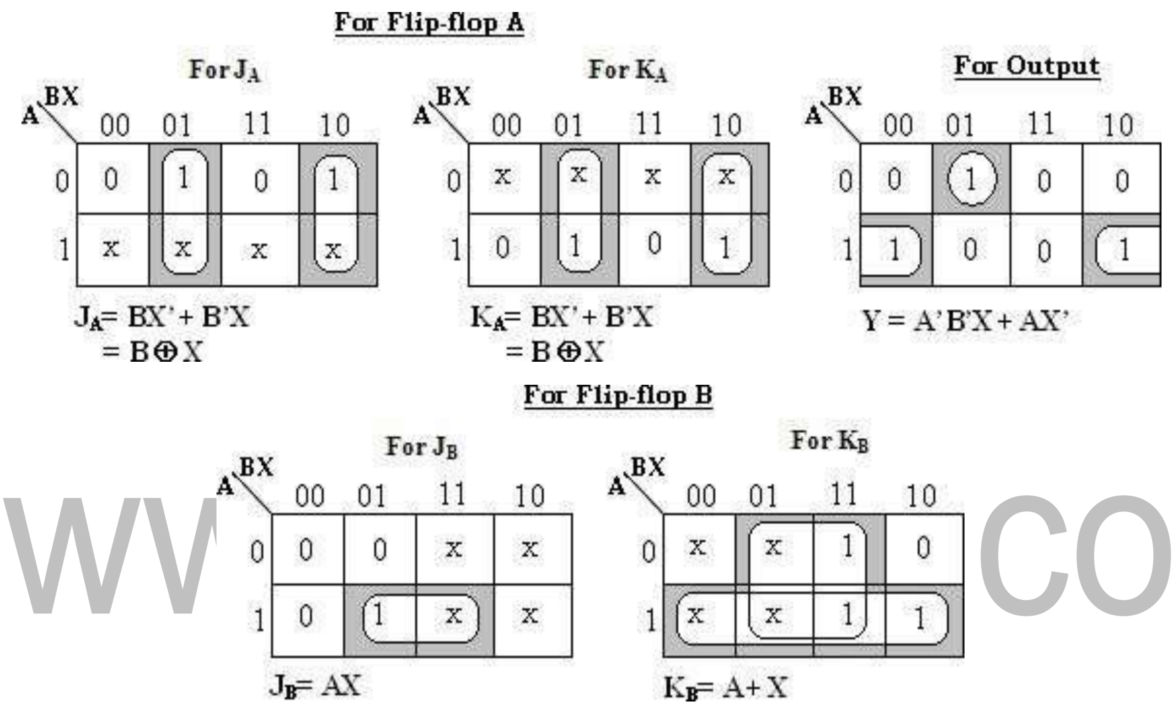
Present State	Next State	Inputs	
Qn	Qn+1	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Present state		Input	Next state		Flip-Flop Inputs				Output
A	B	X	A	B	JA	KA	JB	KB	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

Circuit excitation table

K-map Simplification:

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



The input functions

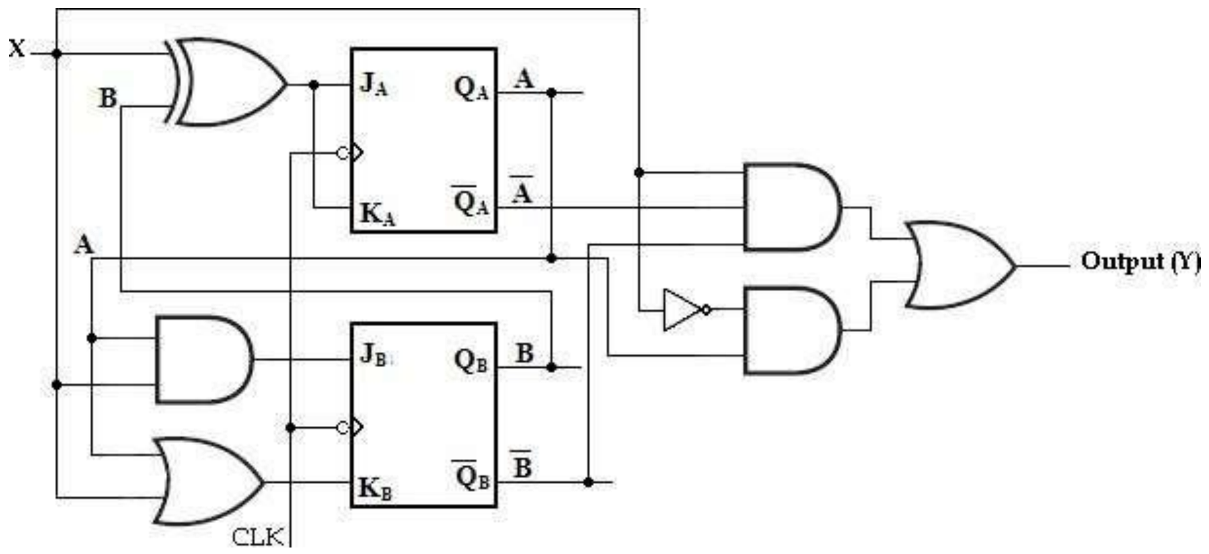
J_A= BX'+ B'X

J_B=

K_A= BX'+ B'X

K_B= A+

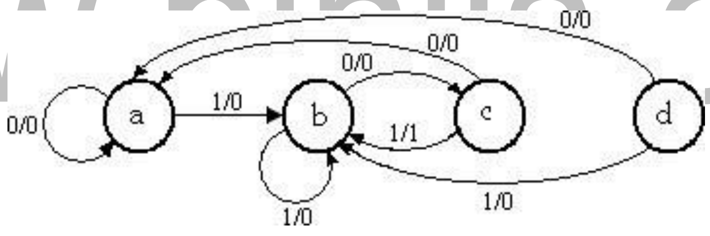
Circuit output function, **Y= AX'+**



Logic diagram of given sequential circuit using JK Flip-Flop

2. Design a clocked sequential machine using JK Flip-Flops for the state diagram

shown in the figure. Use state reduction if possible. Make proper state



Soln:

State

Present state	**Next state**		**Output**	
	X= 0	**X= 1**	**X= 0**	**X= 1**
a	a	b	0	0
b	c	b	0	0
c	a	b	0	1
d	a	b	0	0

From the above state table **a** and **d** generate exactly same next state and same output for every possible set of inputs. The state **a** and **d** go to next states **a** and **b** and have outputs 0 and 0 for x=0 and x=1 respectively. Therefore state **d** can be removed and replaced by **a**. The final reduced state table is shown below.

Present state	**Next state**		**Output**	
	X=0	**X=1**	**X=0**	**X=1**
a	a	b	0	0
b	c	b	0	0
c	a	b	0	1

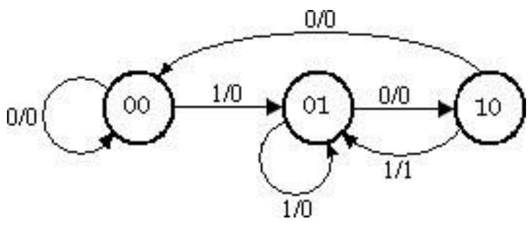
Reduced State table

Binary Assignment:

Now each state is assigned with binary values. Since there are three states, number of Flip-Flops required is two and 2 binary numbers are assigned to the states.

a= 00; b= 0; and c= 10

The reduced state diagram is drawn as,



Reduced State Diagram

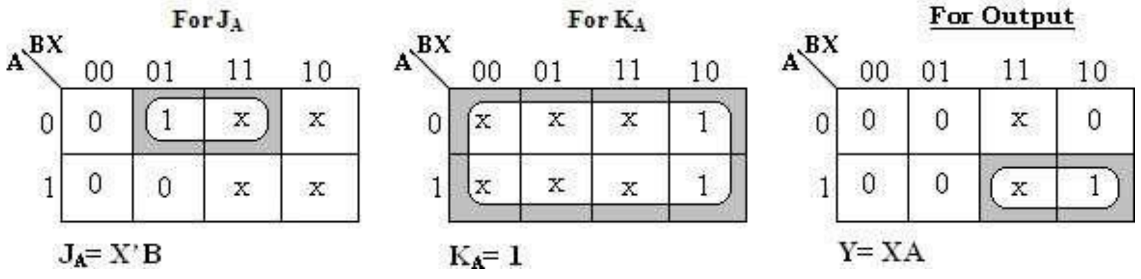
Excitation Table:

Present State	Next State	Inputs	
Qn	Qn+1	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Input	Present state		Next state		Flip-Flop nputs				Output
X	A	B	A	B	JA	KA	JB	KB	Y
0	0	0	0	0	0	x	0	x	0
1	0	0	0	1	0	x	1	x	0
0	0	1	1	0	1	x	x	1	0
1	0	1	0	1	0	x	x	0	0
0	1	0	0	0	x	1	0	x	0
1	1	0	0	1	x	1	1	x	1
0	1	1	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x

K-map Simplification:

For Flip-flop A


For J_A
For K_A
For Output

For J_A

A \ BX	00	01	11	10
0	0	1	x	x
1	0	0	x	x

$J_A = X'B$

For K_A

A \ BX	00	01	11	10
0	x	x	x	1
1	x	x	x	1

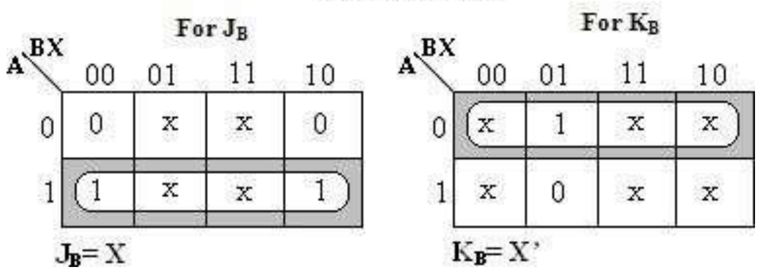
$K_A = 1$

For Output

A \ BX	00	01	11	10
0	0	0	x	0
1	0	0	x	1

$Y = XA$

For Flip-flop B


For J_B
For K_B

For J_B

A \ BX	00	01	11	10
0	0	x	x	0
1	1	x	x	1

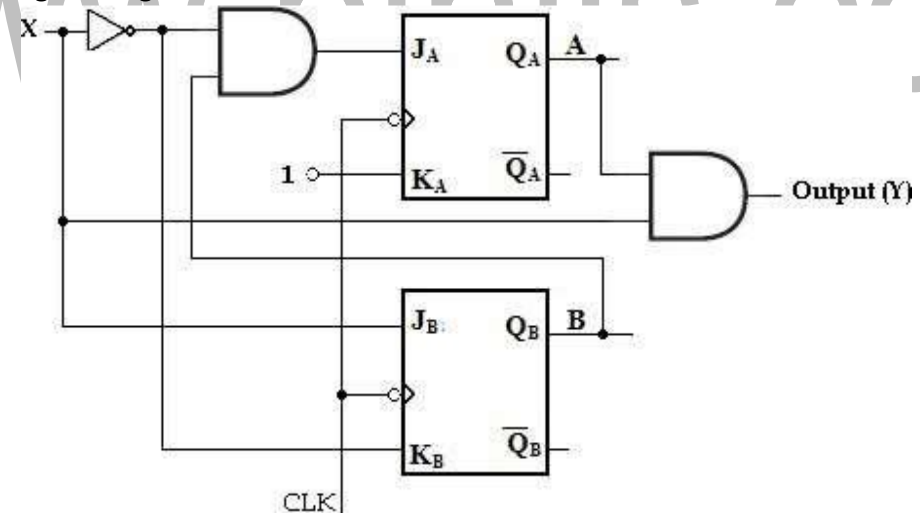
$J_B = X$

For K_B

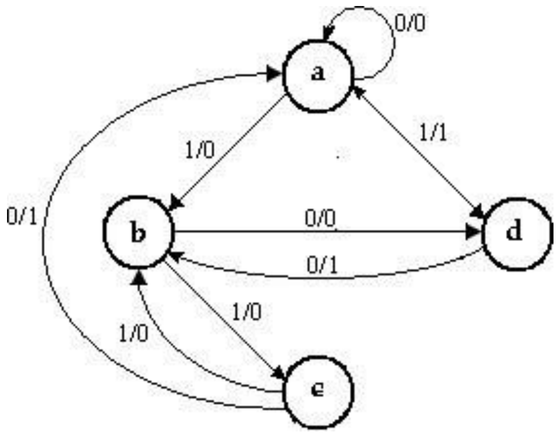
A \ BX	00	01	11	10
0	x	1	x	x
1	x	0	x	x

$K_B = X'$

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



3. 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.



Soln:

State Table:

State table for the given state diagram is,

Present state	**Next state**		**Output**	
	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 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.

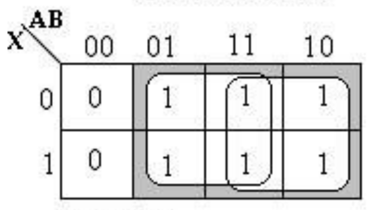
State Assignment:

Use straight binary assignments as a= 00, b= 01, c= 10 and d= 11, the transition table is,

Input	Present state		Next state		Flip-Flop Inputs		Output
X	A	B	A	B	TA	TB	Y
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

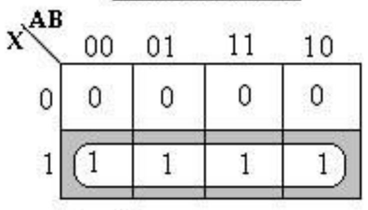
K-map simplification:

For Flip-flop A



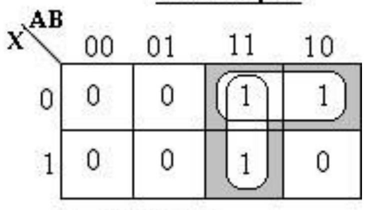
T_A= A+B

For Flip-flop B



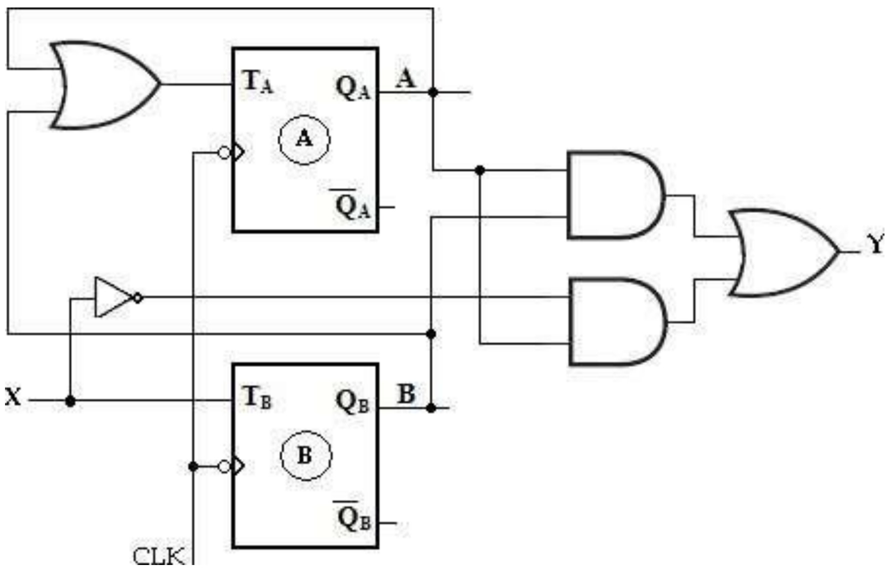
T_B= X

For Output



Z= AB+ X'A

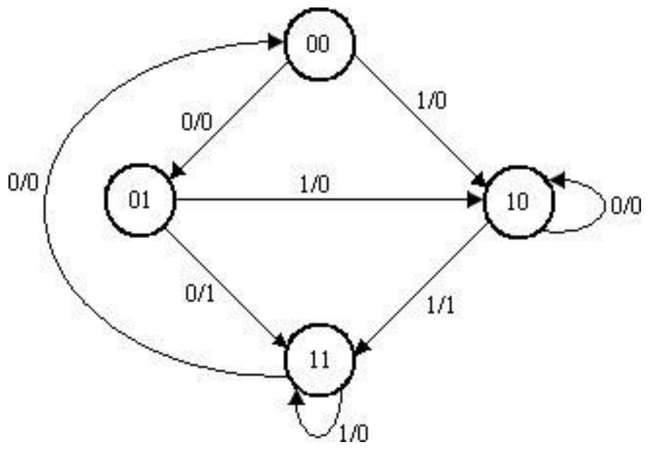
Logic Diagram:



STATE ASSIGNMENT:

In sequential circuits, the behavior of the circuit is defined in terms of its inputs, present states, next states and outputs. To generate 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 functions, it is necessary to represent states in the state diagram using binary values instead of alphabets. This procedure is known as ***state assignment***.



Reduced state diagram with binary states

Rules for state assignments

There are two basic rules for making state assignments.

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.

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 K-map.

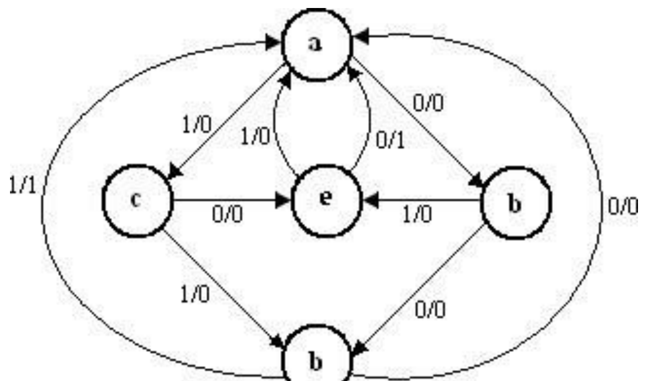
Present state	**Next state**		**Output**	
	X=0	**X=1**	**X=0**	**X=1**
00	01	10	0	0
01	11	10	1	0
10	10	11	0	1

11	00	11	0	0

State table with assignment states

State Assignment Problem:

1. Design a sequential circuit for a state diagram shown below. Use state assignment rules for assigning states and compare the required combinational circuit with random state assignment.



Using random state assignment we assign,

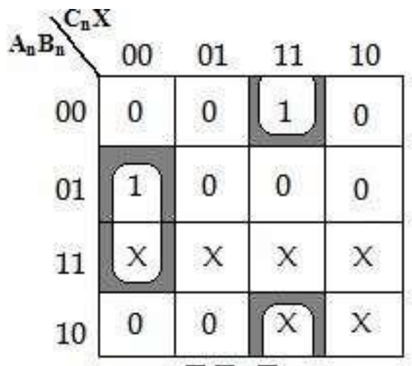
a= 000, b= 001, c= 010, d= 011 and e= 100.

The excitation table with these assignments is given as,

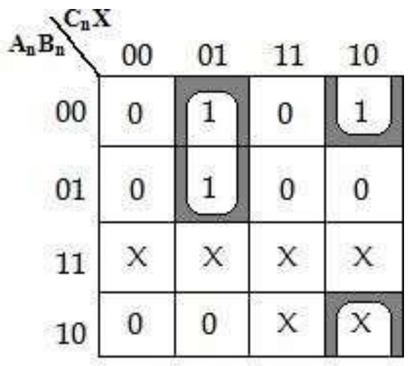
Present te st a			Input	Next stat e			Outpu t
An	B n	C n	X	An+1	Bn+1	Cn+1	Z
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	0

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

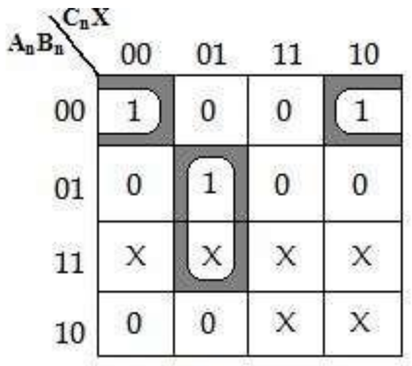
K-map Simplification:



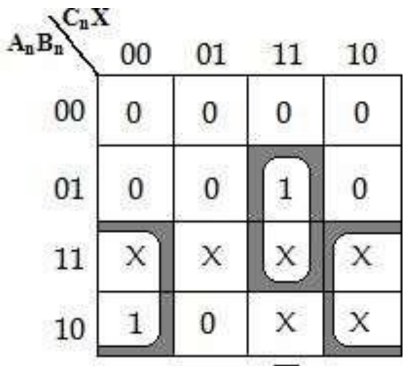
$D_A = B_n\overline{C_n}\,\overline{X} + \overline{B_n}C_nX$



$D_B = \overline{A_n}\,\overline{C_n}X + \overline{B_n}C_n\overline{X}$



$D_C = \overline{A_n}\,\overline{B_n}\,\overline{X} + B_n\overline{C_n}X$



$Z = B_nC_nX + A_n\overline{X}$

The random assignments require:

7 three input AND

functions 1 two input AND

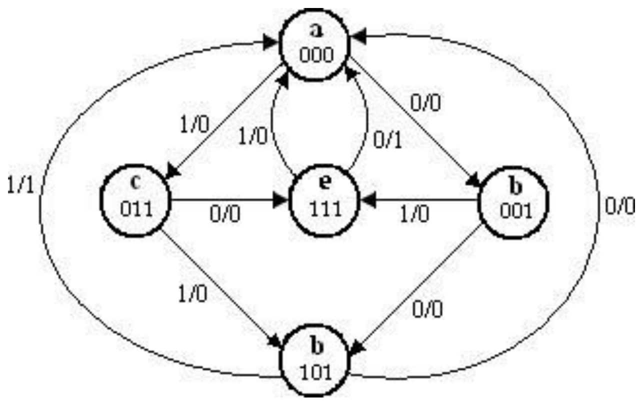
function

4 two input OR functions

----

12 gates with 31 inputs

Now, we will apply the state assignment rules and compare the results.



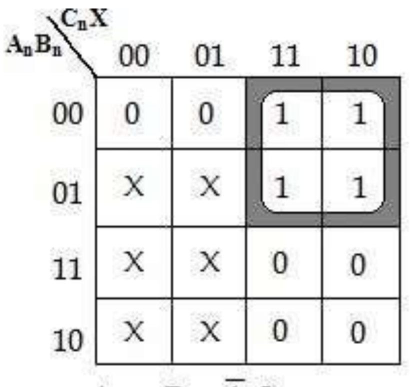
State diagram after applying Rules 1 and 2

Rule 1 says that: e and d must be adjacent, and b and c must be adjacent. Rule 2 says that: e and d must be adjacent, and b and c must be adjacent.

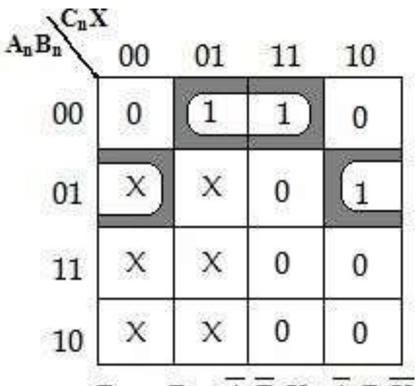
Applying Rule 1, Rule 2 to the state diagram we get the state assignment as,

Present state			**Input**	**Next state**			**Output**
An	**Bn**	**Cn**	**X**	**An+1**	**Bn+1**	**Cn+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

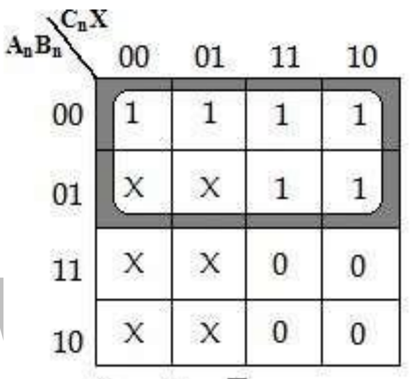
K-map Simplification:



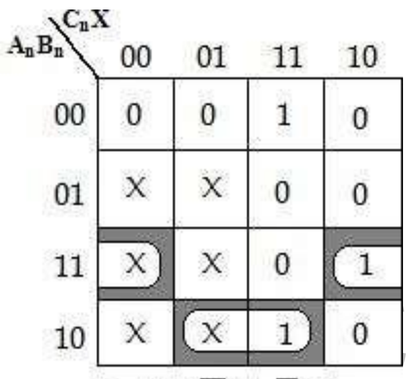
$A_{n+1} = D_A = \overline{A}_n C_n$



$B_{n+1} = D_B = \overline{A}_n \overline{B}_n X + \overline{A}_n B_n \overline{X}$



$C_{n+1} = D_C = \overline{A}_n$



$Z = A_n B_n \overline{X} + A_n \overline{B}_n X$

The state assignments using Rule 1 and 2

require: 4 three input AND functions

1 two input AND function

2 two input OR functions

---

7 gates with 18 inputs

Thus by simply applying Rules 1 and 2 good results have been achieved.

EC 8392 – DIGITAL ELECTRONICS

UNIT – III : SYNCHRONOUS SEQUENTIAL CIRCUITS

SYNCHRONOUS COUNTERS

Flip-Flops can be connected together to perform counting operations. Such a group of Flip- Flops is a **counter**. The number of Flip-Flops used and the way in which they are connected determine the number of states (called the modulus) and also the specific sequence of states that the counter goes through during each complete cycle.

Counters are classified into two broad categories according to the way they are clocked:

- Asynchronous counters,
- Synchronous counters.

In asynchronous (ripple) counters, the first Flip-Flop is clocked by the external clock pulse and then each successive Flip-Flop is clocked by the output of the preceding Flip-Flop.

In synchronous counters, the clock input is connected to all of the Flip-Flops so that they are clocked simultaneously. Within each of these two categories, counters are classified primarily by the type of sequence, the number of states, or the number of Flip-Flops in the counter.

The term 'synchronous' refers to events that have a fixed time relationship with each other. In synchronous counter, the clock pulses are applied to all Flip- Flops simultaneously. Hence there is minimum propagation delay.

Table : 3.3 – Difference between Asynchronous and synchronous Counter

S.No	**Asynchronous (ripple) counter**	**Synchronous counter**
1	All the Flip-Flops are not clocked simultaneously.	All the Flip-Flops are clocked simultaneously.
2	The delay times of all Flip- Flops are added. Therefore there is considerable propagation delay.	There is minimum propagation delay.
3	Speed of operation is low	Speed of operation is high.
4	Logic circuit is very simple even for more number of states.	Design involves complex logic circuit as number of state increases.
5	Minimum numbers of logic devices are needed.	The number of logic devices is more than ripple counters.
6	Cheaper than synchronous counters.	Costlier than ripple counters.

Bit Synchronous Binary Counter

In this counter the clock signal is connected in parallel to clock inputs of both the Flip-Flops (FF_0 and FF_1). The output of FF_0 is connected to J_1 and K_1 inputs of the second Flip-Flop (FF_1).

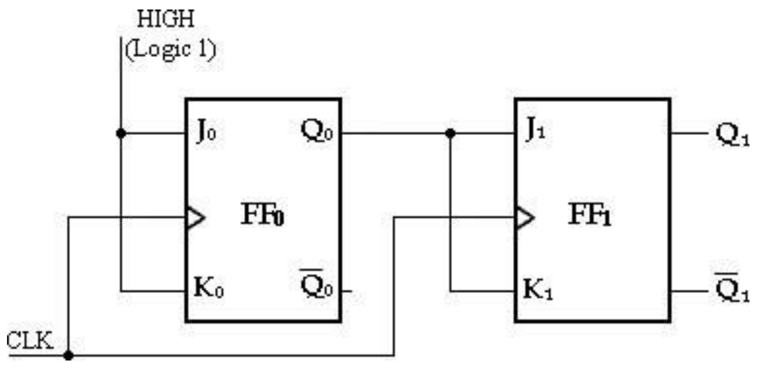


Fig : 3.34 - 2-Bit Synchronous Binary Counter

Assume that the counter is initially in the binary 0 state: i.e., both Flip-Flops are RESET. When the positive edge of the first clock pulse is applied, FF_0 will toggle because $J_0= k_0= 1$, whereas FF_1 output will remain 0 because $J_1= k_1= 0$. After the first clock pulse $Q_0=1$ and $Q_1=0$.

When the leading edge of CLK2 occurs, FF_0 will toggle and Q_0 will go LOW. Since FF_1 has a HIGH ($Q_0 = 1$) on its J_1 and K_1 inputs at the triggering edge of this clock pulse, the Flip-Flop toggles and Q_1 goes HIGH. Thus, after CLK2,

$Q_0 = 0$ and $Q_1 = 1$.

When the leading edge of CLK3 occurs, FF_0 again toggles to the SET state ($Q_0 = 1$), and FF_1 remains SET ($Q_1 = 1$) because its J_1 and K_1 inputs are both LOW ($Q_0 = 0$).

After this triggering edge, $Q_0 = 1$ and $Q_1 = 1$.

Finally, at the leading edge of CLK4, Q_0 and Q_1 go LOW because they both have a toggle condition on their J_1 and K_1 inputs. The counter has now recycled to its original state, $Q_0 = Q_1 = 0$.

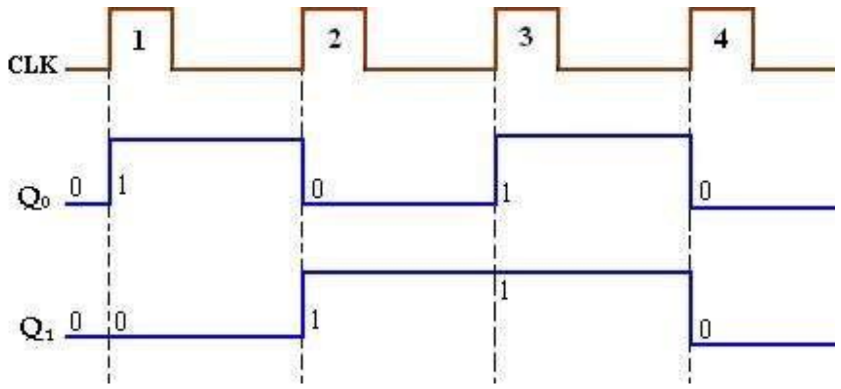


Fig : 3.35 - Timing diagram

3- Bit Synchronous Binary Counter

A 3 bit synchronous binary counter is constructed with three JK Flip-Flops and an AND gate. The output of FF$_0$ (Q$_0$) changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state. To produce this operation, FF$_0$ must be held in the toggle mode by constant HIGH, on its J$_0$ and K$_0$ inputs.

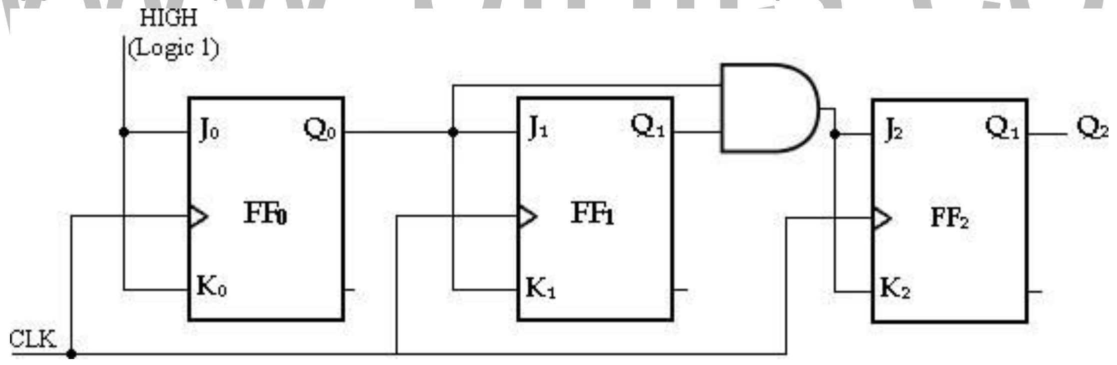


Fig : 3.36 - 3-Bit Synchronous Binary Counter

The output of FF$_1$ (Q$_1$) goes to the opposite state following each time Q$_0$= 1.

This change occurs at CLK2, CLK4, CLK6, and CLK8. The CLK8 pulse causes the counter to recycle. To produce this operation, Q$_0$ is connected to the J$_1$ and K$_1$ inputs of FF_1. When Q_0= 1 and a clock pulse occurs, FF_1 is in the toggle mode and therefore changes state. When Q_0= 0, FF_1 is in the no-change mode and remains in its present state.

The output of FF_2 (Q_2) changes state both times; it is preceded by the unique condition in which both Q_0 and Q_1 are HIGH. This condition is detected by the AND gate and applied to the J_2 and K_2 inputs of FF_3. Whenever both outputs Q_0= Q_1= 1, the output of the AND gate makes the J_2= K_2= 1 and FF_2 toggles on the following clock pulse. Otherwise, the J_2 and K_2 inputs of FF2 are held LOW by the AND gate output, FF_2 does not change state.

CLOCK Pulse	Q2	Q1	Q0
Initiall y 1	0 0	0 0	0 1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

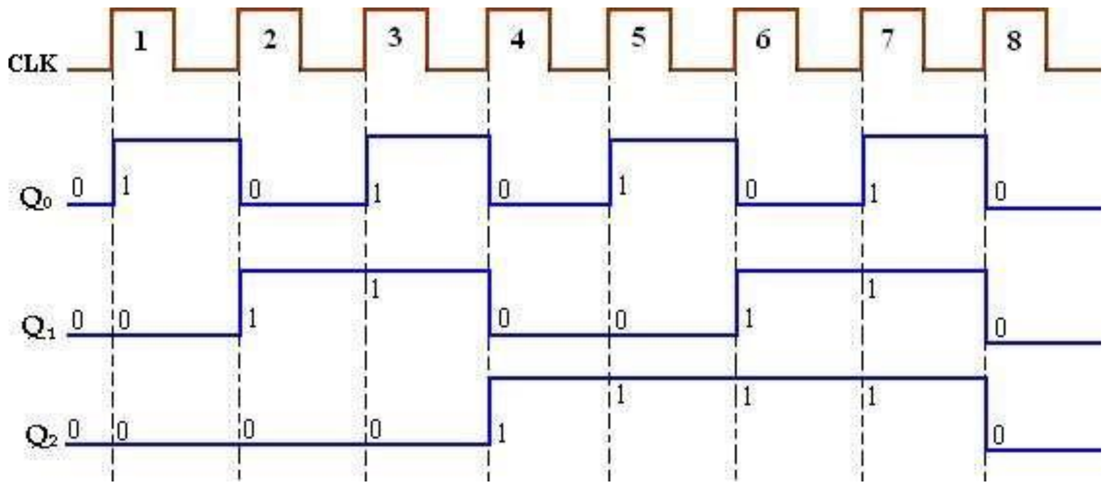


Fig : 3.37 - Timing diagram

4- Bit Synchronous Binary Counter

This particular counter is implemented with negative edge-triggered Flip- Flops. The reasoning behind the J and K input control for the first three Flip- Flops is the same as previously discussed for the 3-bit counter. For the fourth stage, the Flip- Flop has to change the state when $Q_0= Q_1= Q_2= 1$. This condition is decoded by AND gate G_3.

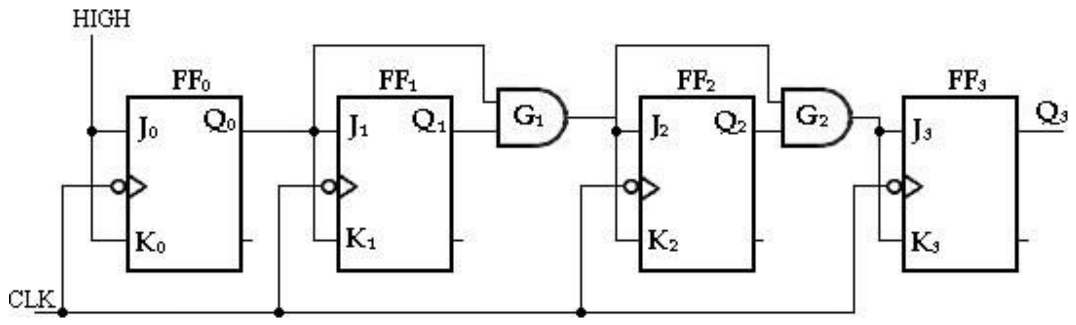


Fig : 3.38 - 4-Bit Synchronous Binary Counter

Therefore, when $Q_0= Q_1= Q_2= 1$, Flip-Flop FF_3 toggles and for all other times it is in a no-change condition. Points where the AND gate outputs are HIGH are indicated by the shaded areas.

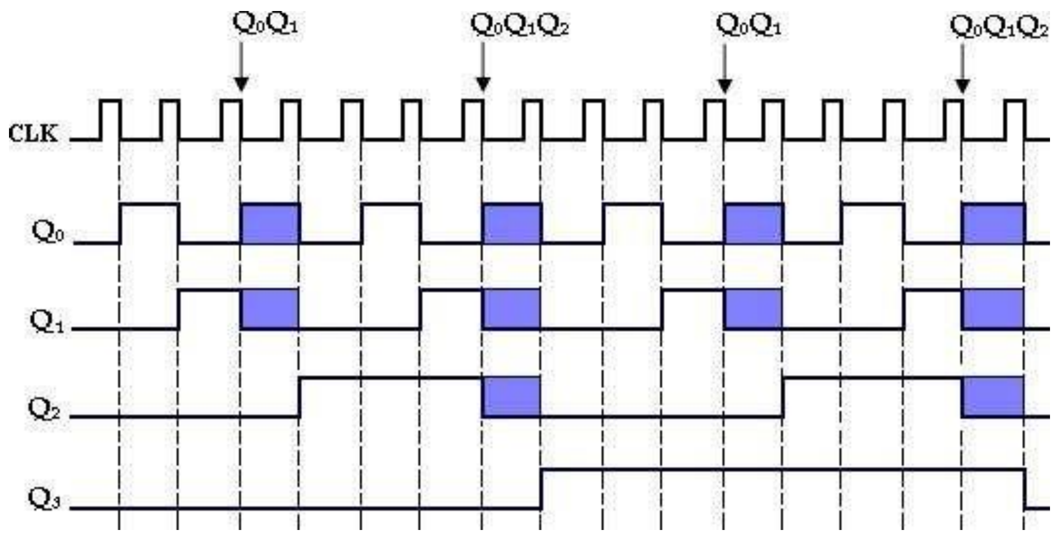
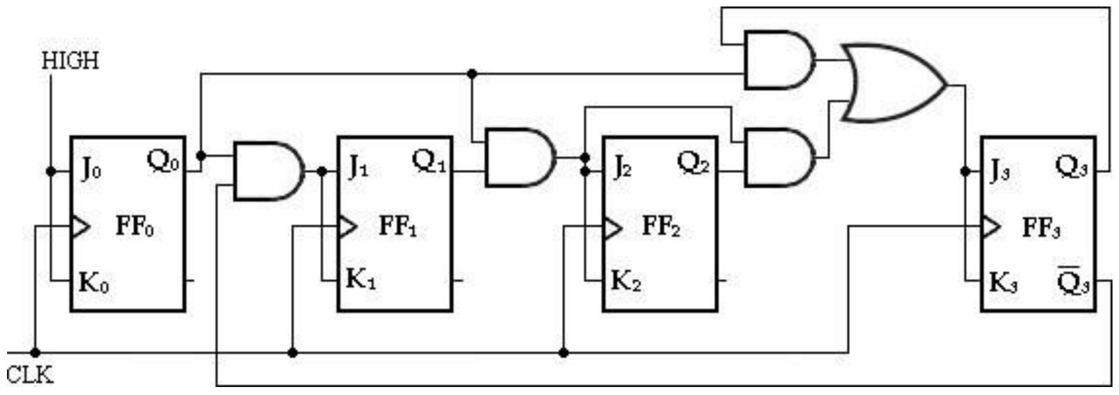


Fig : 3.39 - Timing state

4-Bit Synchronous Decade Counter: (BCD Counter):

BCD decade counter has a sequence from 0000 to 1001 (9). After 1001 it must recycle back to 0000 state. This counter requires four Flip-Flops and AND/OR logic as shown below.



CLOCK Pulse	Q3	Q2	Q1	Q0
Initiall	0	0	0	0
y 1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10(recycles)	0	0	0	0

xFirst, notice that FF_0 (Q_0) toggles on each clock pulse, so the logic equation for its J_0 and K_0 inputs is

$$J_0 = K_0 = 1$$

This equation is implemented by connecting J_0 and K_0 to a constant HIGH level. xNext, notice from table, that FF_1 (Q_1) changes on the next clock pulse each time Q0 = 1 and Q3 = 0, so the logic equation for the J_1 and K_1 inputs is

$$J_1 = K_1 = Q_0Q_3'$$

This equation is implemented by ANDing Q_0 and Q_3 and connecting the gate output to the J_1 and K_1 inputs of FF_l.

xFlip-Flop 2 (Q_2) changes on the next clock pulse each time both $Q_0 = Q_1 = 1$. This requires an input logic equation as follows:

$$J2 = K2 = Q0Q1$$

This equation is implemented by ANDing Q_0 and Q_1 and connecting the gate output to the J_2 and K_2 inputs of FF_3.

xFinally, FF_3 (Q_3) changes to the opposite state on the next clock pulse each time

$Q_0 = 1$, $Q_1 = 1$, and $Q_2 = 1$ (state 7), or when $Q_0 = 1$ and $Q_1 = 1$ (state 9). The equation for this is as follows:

$$J3= K3= Q0Q1Q2+ Q0Q3$$

This function is implemented with the AND/OR logic connected to the J_3 and K_3 inputs of FF_3.

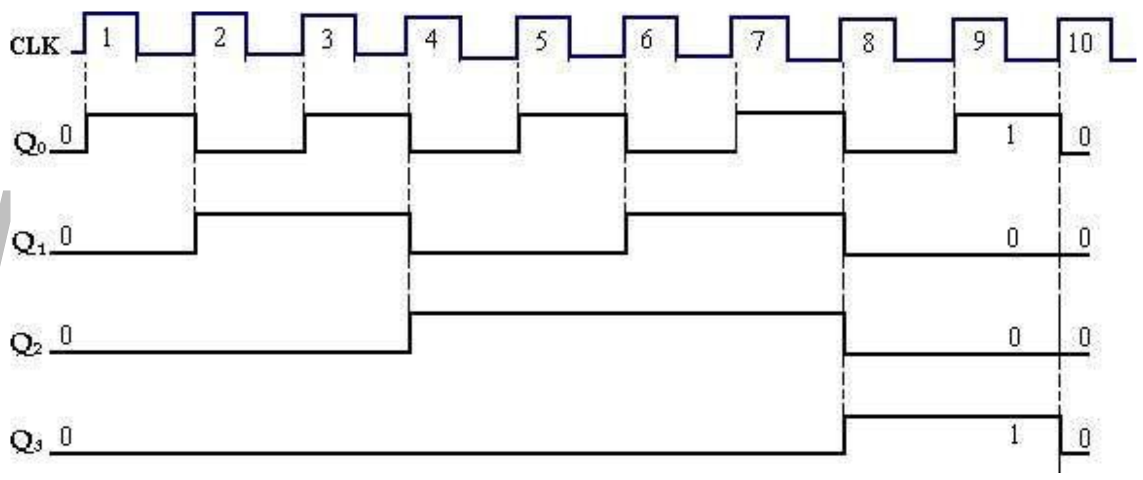


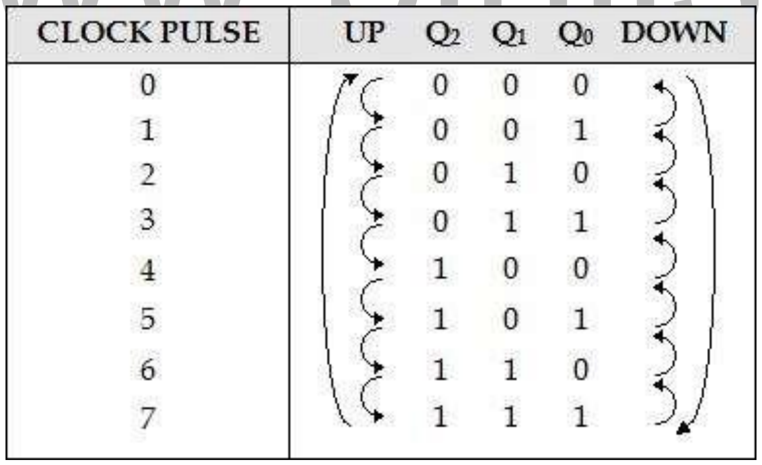
Fig : 3.40 - Timing diagram

Synchronous UP/DOWN Counter

An up/down counter is a bidirectional counter, capable of progressing in either direction through a certain sequence. A 3-bit binary counter that advances upward through its sequence (0, 1, 2, 3, 4, 5, 6, 7) and then can be reversed so that it goes through the sequence in the opposite direction (7, 6, 5, 4, 3, 2, 1,0) is an illustration of up/down sequential operation.

The complete up/down sequence for a 3-bit binary counter is shown in table below. The arrows indicate the state-to-state movement of the counter for both its UP and its DOWN modes of operation. An examination of Q_0 for both the up and down sequences shows that FF_0 toggles on each clock pulse. Thus, the J_0 and K_0 inputs of
FF_0 are,

$J_0 = K_0 = 1$



CLOCK PULSE	UP	Q_2	Q_1	Q_0	DOWN
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

To form a synchronous UP/DOWN counter, the control input (UP/DOWN) is used to allow either the normal output or the inverted output of one Flip-Flop to the J and K inputs of the next Flip-Flop. When UP/DOWN= 1, the MOD 8 counter will count from 000 to 111 and UP/DOWN= 0, it will count from 111 to 000.

When UP/DOWN= 1, it will enable AND gates 1 and 3 and disable AND gates 2 and 4. This allows the Q0 and Q1 outputs through the AND gates to the J and K inputs of the following Flip-Flops, so the counter counts up as pulses are applied. When UP/DOWN= 0, the reverse action takes place.

$$J_1 = K_1 = (Q_0.UP) + (Q_0'.DOWN)$$

$$J_2 = K_2 = (Q_0.\ Q_1.UP) + (Q_0'.Q_1'.DOWN)$$

Fig : 3.41 - 3-bit UP/DOWN Synchronous Counter

MODULUS-N-COUNTERS

The counter with 'n' Flip-Flops has maximum MOD number 2^n. Find the number of Flip-Flops (n) required for the desired MOD number (N) using the equation,

$2n \geq N$

(i) For example, a 3 bit binary counter is a MOD 8 counter. The basic counter can be modified to produce MOD numbers less than 2^n by allowing the counter to skin those are normally part of counting sequence.

$n = 3$ $N = 8$ $2^n =$

$2^3 = 8 = N$

(ii) MOD 5 Counter:

$2^n = N$

$2^n = 5$

$2^2 = 4$ less than N.

$2^3 = 8 > N(5)$ Therefore, 3 Flip-Flops are required.

(iii) MOD 10 Counter:

$2^n = N = 10$ $2^3 = 8$ less than N.

$2^4 = 16 > N(10)$. To construct any MOD-N counter, the following methods can be used.

1. Find the number of Flip-Flops (n) required for the desired MOD number (N) using the equation,

   $$2^n \geq N.$$
2. Connect all the Flip-Flops as a required counter.
3. Find the binary number for N.
4. Connect all Flip-Flop outputs for which Q= 1 when the count is N, as inputs to NAND gate.
5. Connect the NAND gate output to the CLR input of each Flip-Flop.

When the counter reaches Nth state, the output of the NAND gate goes LOW, resetting all Flip-Flops to 0. Therefore the counter counts from 0 through N-1.

For example, MOD-10 counter reaches state 10 (1010). i.e., $Q_3Q_2Q_1Q_0 = 1\ 0\ 1\ 0$. The outputs Q_3 and Q_1 are connected to the NAND gate and the output of the NAND gate goes LOW and resetting all Flip-Flops to zero. Therefore MOD-10 counter counts from 0000 to 1001. And then recycles to the zero value.

The MOD-10 counter circuit is shown below.

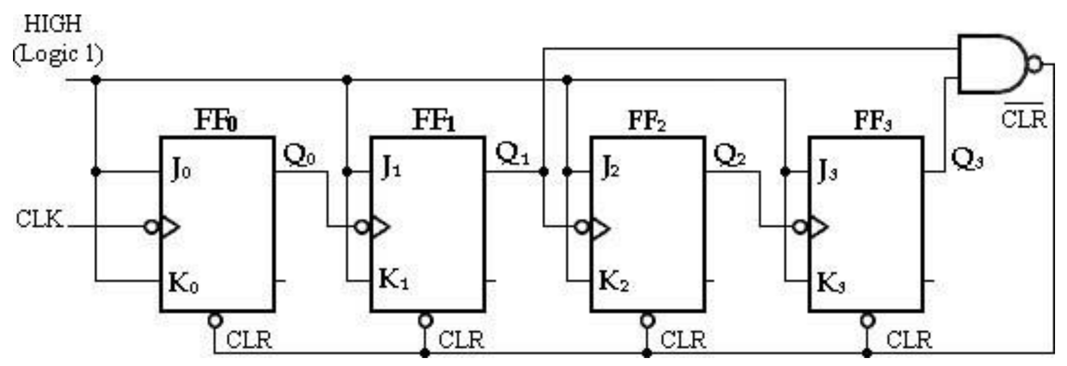


Fig : 3.42 - MOD-10 (Decade)

EC 8392 – DIGITAL ELECTRONICS

UNIT – III : SYNCHRONOUS SEQUENTIAL CIRCUITS

INTRODUCTION

In ***combinational logic circuits***, the outputs at any instant of time depend only on the input signals present at that time. For a change in input, the output occurs immediately.

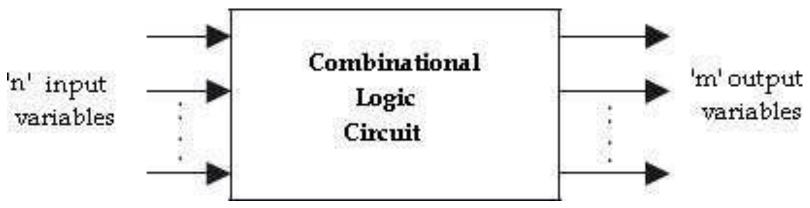


Fig : 3.1 - Combinational Circuit- Block Diagram

In ***sequential logic circuits***, it consists of combinational circuits to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information either 1 or 0.

The information stored in the memory elements at any given time defines the present state of the sequential circuit. The present state and the external circuit determine the output and the next state of sequential circuits.

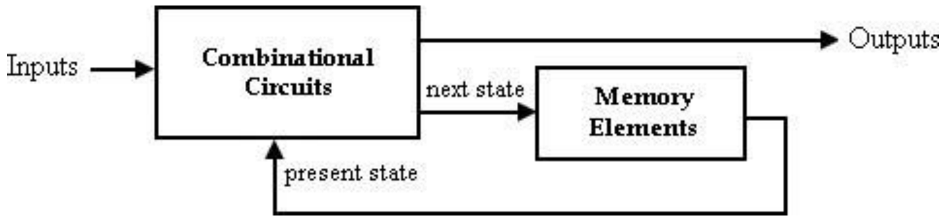


Fig : 3.2 - Sequential Circuit- Block Diagram

Thus in sequential circuits, the output variables depend not only on the present input variables but also on the past history of input variables.

The rotary channel selected knob on an old-fashioned TV is like a combinational. Its output selects a channel based only on its current input – the position of the knob. The channel-up and channel-down push buttons on a TV is like a sequential circuit. The channel selection depends on the past sequence of up/down pushes.

Table: 3.1 - The comparison between combinational and sequential circuits

S.No	Combinational logic	Sequential logic
1	The output variable, at all times depends on the combination of input variables.	The output variable depends not only on the present input but also depend upon the past history of inputs.
2	Memory unit is not required	Memory unit is required to store the past history of input variables.
3	Faster in speed	Slower than combinational circuits.
4	Easy to design	Comparatively harder to design.
5	Eg. Parallel adder	Eg. Serial adder

Classification of Logic Circuits

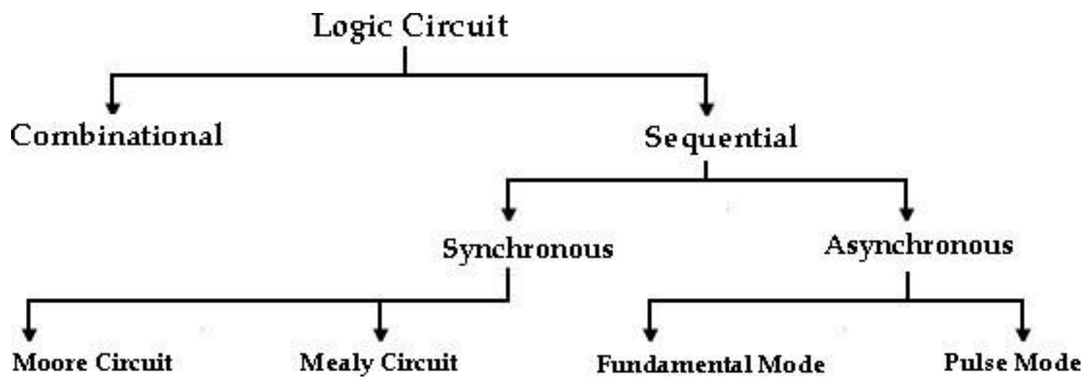


Fig : 3.3 – Classification of sequential Circuits

The sequential circuits can be classified depending on the timing of their signals:

- Synchronous sequential circuits
- 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. The memory elements used in both circuits are Flip-Flops, which are capable of storing 1- bit information.

Table: 3.2 - The comparison between Synchronous and asynchronous sequential circuits

S.No	Synchronous sequential circuits	Asynchronous sequential circuits
1	Memory elements are clocked Flip-Flops	Memory elements are either unclocked Flip-Flops 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 the absence of clock, it can operate faster than synchronous circuits.
4	Easier to design	More difficult to design

LATCHES:

Latches and Flip-Flops are the basic building blocks of the most sequential circuits. Latches are used for a sequential device that checks all of its inputs continuously and changes its outputs accordingly at any time independent of clocking signal. Enable signal is provided with the latch. When enable signal is active output changes occur as the input changes. But when enable signal is not activated input changes do not affect the output.

Flip-Flop is used for a sequential device that normally samples its inputs and changes its outputs only at times determined by clocking signal.

SR Latch:

The simplest type of latch is the set-reset (SR) latch. It can be constructed from either two NOR gates or two NAND gates.

SR latch using NOR gates:

The two NOR gates are cross-coupled so that the output of NOR gate 1 is connected to one of the inputs of NOR gate 2 and vice versa. The latch has two outputs Q and Q' and two inputs, set and reset.

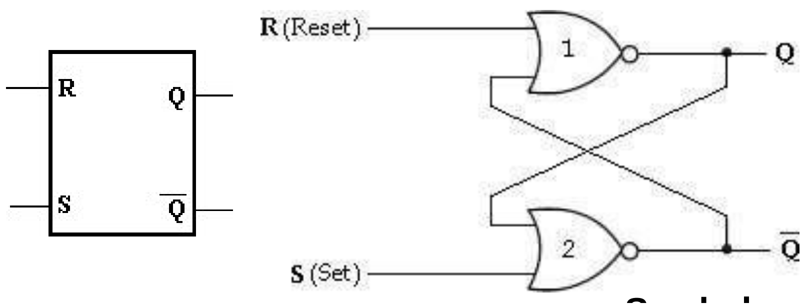


Fig : 3.4 - SR latch using NOR gates

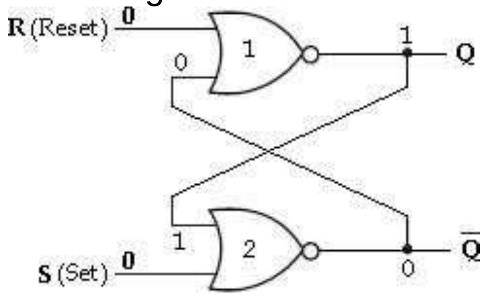
Before going to analyse the SR latch, we recall that a logic 1 at any input of a NOR gate forces its output to a logic 0. Let us understand the operation of this circuit for various input/ output possibilities.

Case 1: S= 0 and R= 0

Initially, Q= 1 and Q'= 0

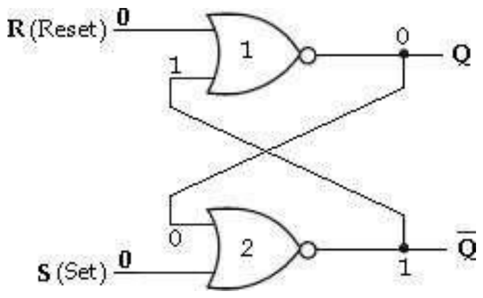
Let us assume that initially Q=1 and Q'=0. With Q'=0, both inputs to NOR gate 1 are at logic 0. So, its output, Q is at logic 1. With Q=1, one input of NOR gate 2 is at logic

1.Hence its output, Q' is at logic 0. This shows that when S and R both are low, the output does not change.



Initially, Q= 0 and Q'= 1

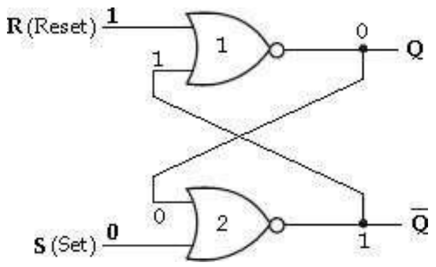
With Q'=1, one input of NOR gate 1 is at logic 1, hence its output, Q is at logic 0.With Q=0, both inputs to NOR gate 2 are at logic 0. So, its output Q' is at logic 1. In this case also there is no change in the output state.



Case 2: S= 0 and R= 1

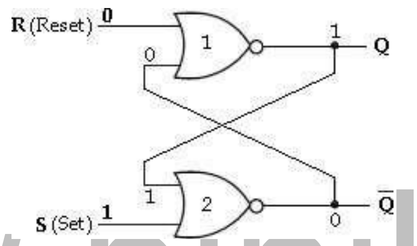
In this case, R input of the NOR gate 1 is at logic 1, hence its output, Q is at logic 0.

Both inputs to NOR gate 2 are now at logic 0. So that its output, Q' is at logic 1.



Case 3: S= 1 and R= 0

In this case, S input of the NOR gate 2 is at logic 1, hence its output, Q is at logic 0. Both inputs to NOR gate 1 are now at logic 0. So that its output, Q is at logic 1.



Case 4: S= 1 and R= 1

When R and S both are at logic 1, they force the outputs of both NOR gates to the low state, i.e., (Q=0 and Q'=0). So, we call this an indeterminate or prohibited state, and represent this condition in the truth table as an asterisk (*). This condition also violates the basic definition of a latch that requires Q to be complement of Q'.

Thus in normal operation this condition must be avoided by making sure that 1's are not applied to both the inputs simultaneously.

We can summarize the operation of SR latch as follows:

xWhen S= 0 and R= 0, the output, Q_{n+1} remains in its present state, Q_n. xWhen S= 0 and R= 1, the latch is reset to 0. xWhen S= 1 and R= 0, the latch is set to 1.

xWhen S= 1 and R= 1, the output of both gates will produce 0. i.e., **Qn+1= Qn+1'= 0.**

The truth table of NOR based SR latch is shown below.

S	R	Qn	Qn+1	State
0 0	0 0	0 1	0 1	No Change (NC)
0 0	1 1	0 1	0 0	Reset
1 1	0 0	0 1	1 1	Set
1 1	1 1	0 1	x x	Indeterminate *

SR latch using NAND gates:

The SR latch can also be implemented using NAND gates. The inputs of this Latch are S and R. To understand how this circuit functions, recall that a low on any input to a NAND gate forces its output high.

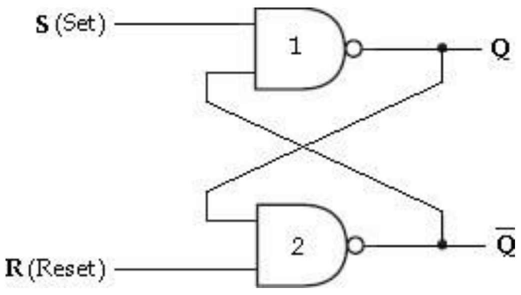


Fig : 3.5 - SR latch using NAND gates

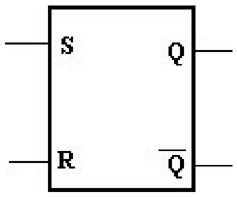


Fig : 3.6 - Logic Symbol

We can summarize the operation of SR latch as follows:

xWhen S= 0 and R= 0, the output of both gates will produce 0. i.e., **Qn+1= Qn+1'= 1.**

xWhen S= 0 and R= 1, the latch is reset to 0. xWhen S= 1 and R= 0, the latch is set to 1. xWhen S= 1 and R= 1, the output, Q_{n+1} remains in its present state, Q_n.

The truth table of NAND based SR latch is shown below.

S	**R**	**Qn**	**Qn+1**	**State**
0 0	0 0	0 1	x x	Indeterminate *
0 0	1 1	0 1	1 1	Set
1 1	0 0	0 1	0 0	Reset

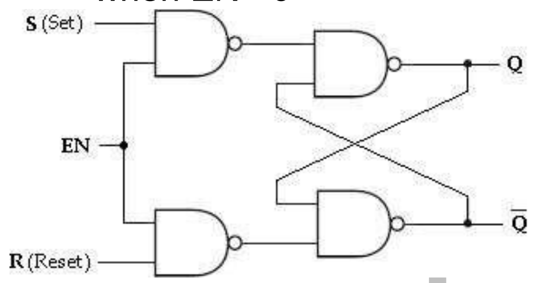
1 1	1 1	0 1	0 1	No Change (NC)

Gated SR Latch:

In the SR latch, the output changes occur immediately after the input changes i.e, the latch is sensitive to its S and R inputs all the time.

A latch that is sensitive to the inputs only when an enable input is active. Such a latch with enable input is known as gated SR latch.

xThe circuit behaves like SR latch when EN= 1. It retains its previous state when EN= 0



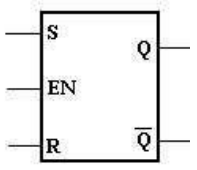


Fig : 3.6 - SR Latch with enable input using NAND gates Fig : 3.7 - Logic Symbol The truth table of gated SR latch is show below.

EN	S	R	Qn	Qn+1	State
1 1	0 0	0 0	0 1	0 1	No Change (NC)
1 1	0 0	1 1	0 1	0 0	Reset

1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	x x	Indeterminate
1	1	1	1		*
0	x x	x x	0	0	No Change (NC)
0			1	1	

When S is HIGH and R is LOW, a HIGH on the EN input sets the latch. When S is LOW and R is HIGH, a HIGH on the EN input resets the latch.

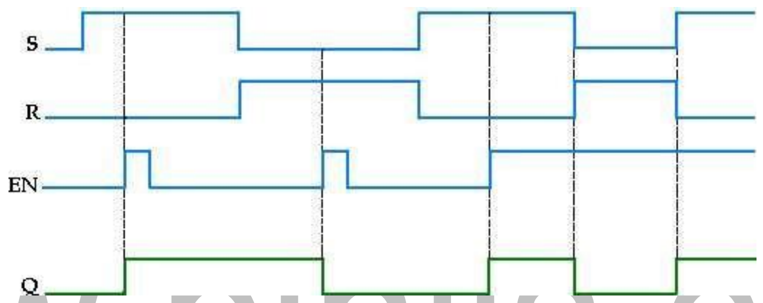
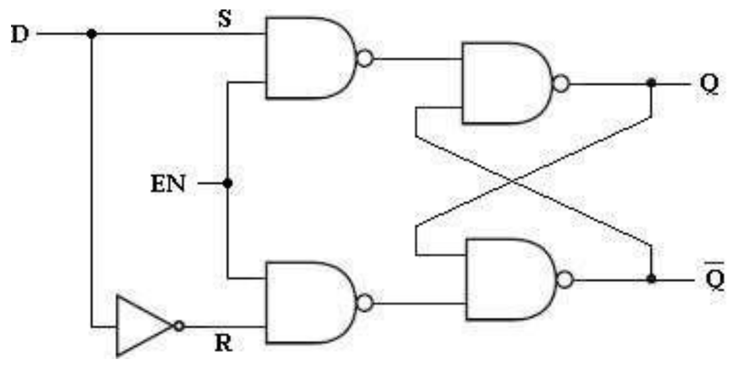


Fig : 3.8 – Timing diagram

D Latch

In SR latch, when both inputs are same (00 or 11), the output either does not change or it is invalid. In many practical applications, these input conditions are not required. These input conditions can be avoided by making them complement of each other. This modified SR latch is known as **D latch.**



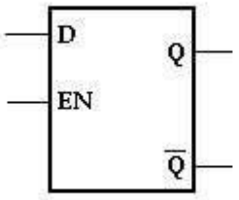


Fig : 3.9 - D Latch Fig : 3.10 - Logic Symbol

As shown in the figure, D input goes directly to the S input, and its complement is applied to the R input. Therefore, only two input conditions exists, either S=0 and R=1 or S=1 and R=0. The truth table for D latch is shown below.

EN	**D**	**Qn**	**Qn+1**	**State**
1	0	x	0	Reset
1	1	x	1	Set
0	x	x	Qn	No Change (NC)

As shown in the truth table, the Q output follows the D input. For this reason, D latch is called ***transparent latch***.

When D is HIGH and EN is HIGH. Q goes HIGH. When D is LOW and EN is HIGH, Q goes LOW. When EN is LOW, the state of the latch is not affected by the D input.

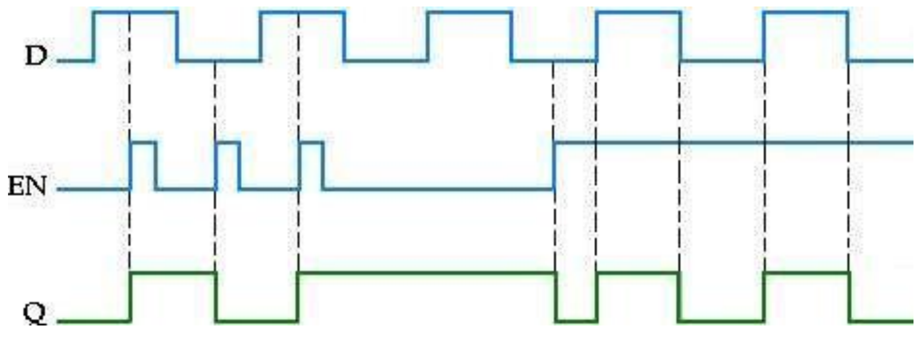


Fig : 3.11 – Timing diagram

EC 8392 – DIGITAL ELECTRONICS

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

ANALYSIS EXAMPLE

Asynchronous sequential circuits can be constructed with the use of SR latches with or without external feedback paths. Of course, there is always a feedback loop within the latch itself. The analysis of a circuit with latches will be demonstrated by means of a specific example from which it will be possible to generalize the procedural steps necessary to analyze other, similar circuits.

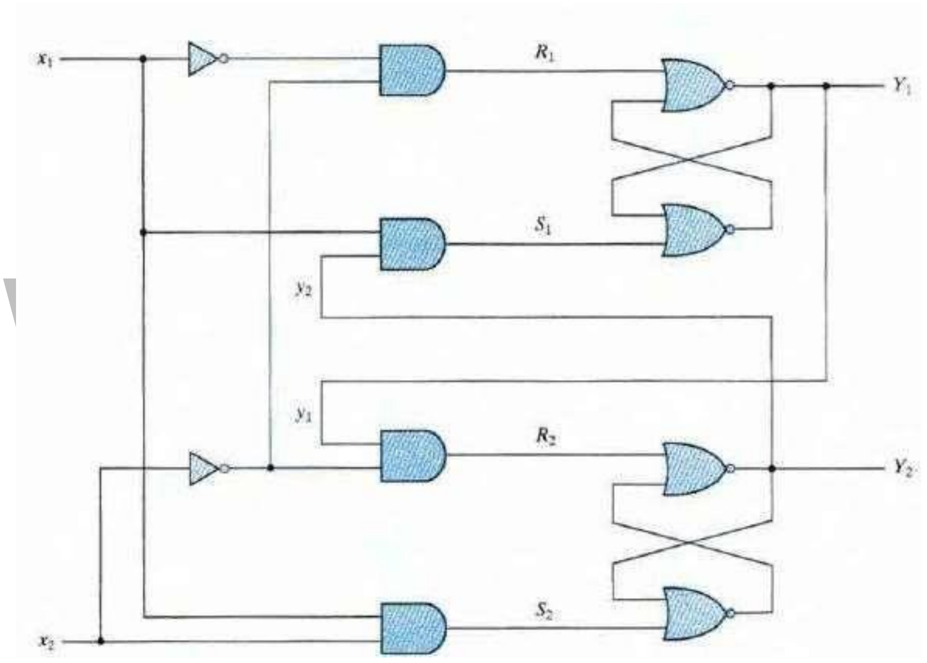


Fig: 4.8 - Examples of a circuit with SR latches

Image source from Digital Design by Moris Mano (Page No. 428)

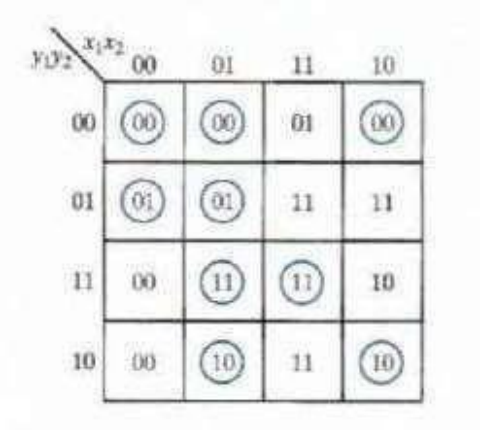
There are two inputs, x_1 and x_2, and two external feedback loops giving rise to the secondary variables, y_1 and y_2. Note that this circuit resembles a conventional sequential circuit with latches behaving like flip-flops without clock pulses. The analysis of the circuit requires that we first obtain the Boolean function for the S and R inputs in each latch:

$$S_1 = x_1 y_2 \qquad S_2 = x_1 x_2$$
$$R_1 = x_1' x_2' \qquad R_2 = x_2' y_1$$

We then check whether the condition SR = 0 is satisfied to ensure proper operation of the circuit:

$$S_1 R_1 = x_1 y_2 x_1' x_2' = 0$$
$$S_2 R_2 = x_1 x_2 x_2' y_1 = 0$$

Fig: 4.9 - Transition table



y_1y_2 \ x_1x_2	00	01	11	10
00	00	00	01	00
01	01	01	11	11
11	00	11	11	10
10	00	10	11	10

Image source from Digital Design by Moris Mano (Page No. 429)

The procedure for analyzing an asynchronous sequential circuit with SR latches can be summarized as follows:

1. Label each latch output with Y_i and its external feedback path (if any) with Yi for i

   = 1,

   2, ... k.
2. Derive the Boolean functions for the S_i and R_i inputs in each latch.
3. Check whether SR = 0 for each NOR latch or whether S'R' = 0 for each NAND latch. If either of these conditions is not satisfied, there is a possibility that the circuit may not operate properly.
4. Evaluate Y = S + R'y for each NOR latch or Y = S' + Ry for each NAND latch.
5. Consruct a map, with the y's representing the rows and the x inputs representing the colwnns.
6. Plot the value of $Y= Y_1Y_2 \ldots Y_K$ in the map.
7. Circle all stable states such that Y = y. The resulting map is then the transition table.

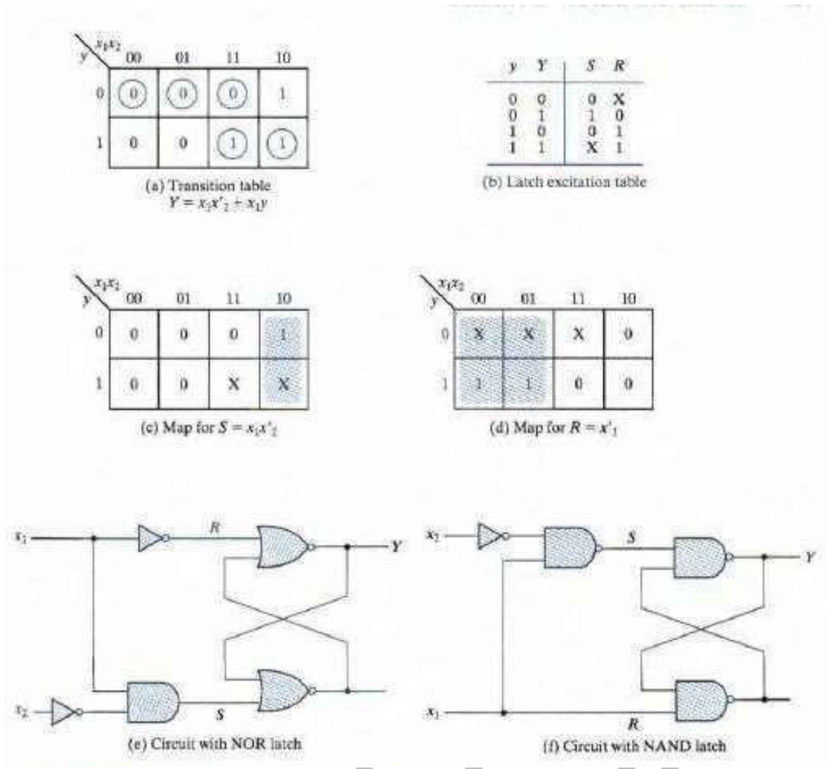

(a) Transition table
$Y = x_1x'_2 + x_1y$
(b) Latch excitation table
(c) Map for $S = x_1x'_2$
(d) Map for $R = x'_1$
(e) Circuit with NOR latch
(f) Circuit with NAND latch

Fig: 4.10 - Derivation of a Latch Circuit fromTransition table

Image source from Digital Design by Moris Mano (Page No. 431)

DESIGN PROCEDURE

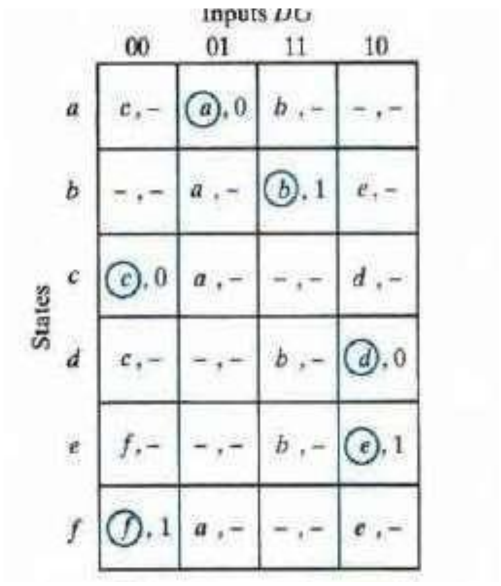
The design of an asynchronous sequential circuit starts from the statement of the problem and culminates in a logic diagram. There are a number of design steps that must be carried out in order to minimize the complexity of the circuit and to produce a stable circuit without critical races. Briefly, the design steps are as follows: A primitive flow table is obtained from the design specifications. The flow table is then reduced to a minimum number of states. Next, the states are given a binary assignment from which we obtain the transition table. Finally, from the transition table, we derive the logic diagram as a combinational circuit with feedback or as a circuit with SR latches.

DESIGN EXAMPLE

It is necessary to design a gated latch circuit with two inputs G (gate) and D (data) and one output Q. Binary information present at the D input is transferred to the Q output when G is equal to 1. The Q output will follow the D input as long as G = 1. When G goes to 0, the information that was present at the D input at the time the transition occurred is retained at the Q output. The gated latch is a memory element that accepts the value of D when G = 1 and retains this value after G goes to 0. Once G = 0, a change in D does not change the value of the output Q.

Primitive Flow Table

A primitive flow table is a flow table with only one stable total state in each row. Remember that a total stale consists of the internal state combined with the input. The derivation of the primitive flow table can be facilitated if we first form a table with all possible total states in the system.



		Inputs DG			
		00	01	11	10
States	a	c, –	ⓐ, 0	b, –	–, –
	b	–, –	a, –	ⓑ, 1	e, –
	c	ⓒ, 0	a, –	–, –	d, –
	d	c, –	–, –	b, –	ⓓ, 0
	e	f, –	–, –	b, –	ⓔ, 1
	f	ⓕ, 1	a, –	–, –	e, –

Fig: 4.10 - Primitive flow table

Image source from Digital Design by Moris Mano (Page No. 434)

The primitive flow table for the gated latch has one row for each state and one column for each input combination. First, we fill in one square in each row belonging to the stable state in that row. These entries are determined from Table. For example, Slate Q is stable and the output is 0 when the input is 01. This information is entered into the flow table in the first row and second column. Similarly, the other five stable stales together with their output are entered into the corresponding input columns. Next, we note that since both inputs are not allowed to change simultaneously, we can enter dash marks in each row that differs in two or more variables from the input variables associated with the stable state. For example, the first row in the flow table shows a stable state with an input of 01. Since only one input can change at any given time, it can change to 00 or 11, but not to 10. Therefore, we enter two dashes in the 10 column of row a. This will eventually result in a don't-care condition for the next state and output in this square. Following the same procedure, we fill in a second square in each row of the primitive flow table.

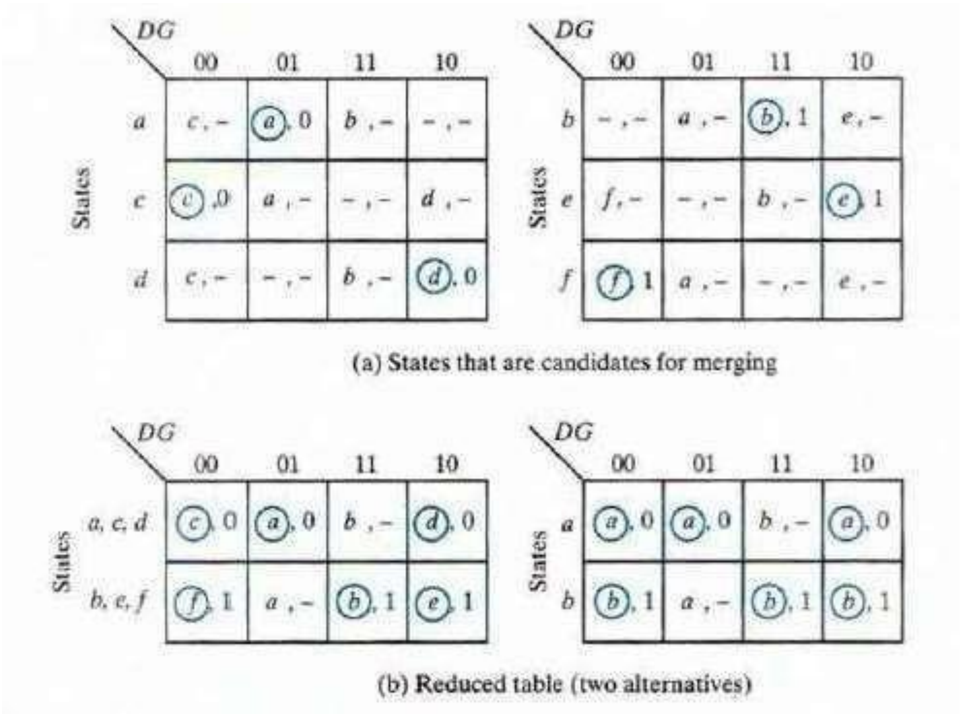
Next, it is necessary to find values for two more squares in each row. The comments listed in Table may help in deriving the necessary information. For example, state c is associated with input 00 and is reached after a change in input from state a or d. Therefore, an unstable state c is shown in column 00 and rows a and d in the flow table. The output is marked with a dash to indicate a don't-care condition. The interpretation of this situation is that if the circuit is in stable state a and the input changes from 01 to 00, the circuit first goes to an unstable next state c. which changes the present-slate value from a to c, causing a transition to the third row and first column of the table. The unstable state values for the other square s are determined in a similar manner. All outputs associated with unstable states are marked with a dash to indicate don't -care conditions.

Reduction of the Primitive flow table

The primitive flow table has only one stable state in each row. The table can be reduced to a smaller number of rows if two or more stable states are placed in the same row. The grouping of stable states from separate rows into one common row is called merging. Merging a number of stable states in the same row means that the binary state variable ultimately assigned to the merged row will not change when the input variable changes. This is because, in a primitive flow table, the stale variable changes every time the input changes, but in a reduced flow table, a change of input will not cause a change in the state variable if the next stable state is in the same row.

Two or more rows in the primitive flow table can be merged into one row if there are non conflicting stales and outputs in each of the columns. Whenever one state symbol and don't -care entries are encountered in the same column, the state is listed in the merged row.

Moreover, if the stale is circled in one of the rows, it is also circled in the merged row. The output value is included with each stable slate in the merged row. Because the merged stales have the same output, the state cannot be distinguished on the basis of the output.



States \ DG	00	01	11	10
a	c, –	a, 0	b, –	–, –
c	c, 0	a, –	–, –	d, –
d	c, –	–, –	b, –	d, 0

States \ DG	00	01	11	10
b	–, –	a, –	b, 1	e, –
e	f, –	–, –	b, –	e, 1
f	f, 1	a, –	–, –	e, –

(a) States that are candidates for merging

States \ DG	00	01	11	10
a, c, d	c, 0	a, 0	b, –	d, 0
b, e, f	f, 1	a, –	b, 1	e, 1

States \ DG	00	01	11	10
a	a, 0	a, 0	b, –	a, 0
b	b, 1	a, –	b, 1	b, 1

(b) Reduced table (two alternatives)

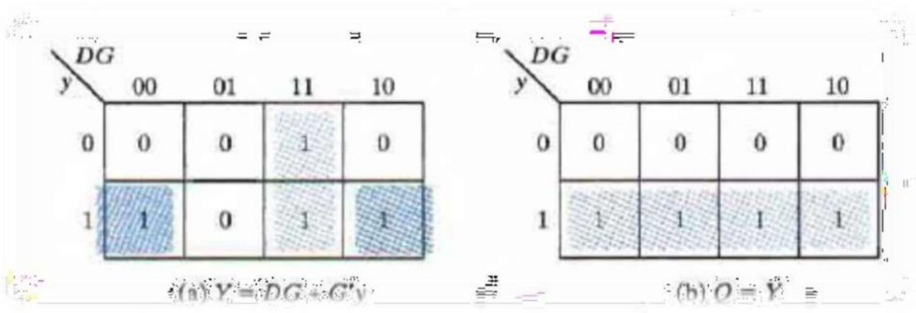
Fig: 4.11 - Reduction of primitive flow tables

Image source from Digital Design by Moris Mano (Page No. 436)

Transition table and Logic Diagram

In order to obtain the circuit described by the reduced flow table, it is necessary to assign a distinct binary value to each state. This assignment converts the flow table into a transition table. In the general case, a binary state assignment must be made to ensure that the circuit will be free of critical races. Fortunately, there can be no critical races in a two-row flow table; therefore, we can finish the design of the gated latch prior to studying that section.Assigning 0 to state a and 1 to state b in the reduced flow table of Fig. (b), we obtain the transition table of Fig. (a). The transition table is, in effect, a map for the excitation variable Y. The simplified Boolean function for Y is then obtained from the map as

Y = DC + C'y



y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(a) $Y = DG + G'y$

y \ DG	00	01	11	10
0	0	0	0	0
1	1	1	1	1

(b) $Q = Y$

Fig: 4.12 - Transition table and output map for a gated latch Image source from Digital Design by Moris Mano (Page No. 437)

Fig. Transition Table and output amp for gated latch

Fig : 4.13 : Gated-latch logic diagram

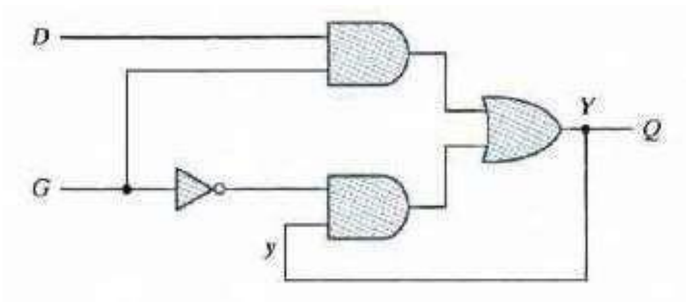


Image source from Digital Design by Moris Mano (Page No. 437)

Assigning outputs to unstable states

The procedure for making the assignment to outputs associated with unstable states can be summarized as follows:

1. Assign a 0 to an output variable associated with an unstable state which is a transient state between two stab le states that have a 0 in the corresponding output variable.
2. Assign a 1 to an output variable associated with an unstable state which is a transient state between two stable states that have a 1 in the corresponding output variable.
3. Assign a don' t-care condition to an output variable associated with an unstable state which is a transient state between two stable states that have different value s (0 and 1, or 1 and 0) in the corresponding output variable .

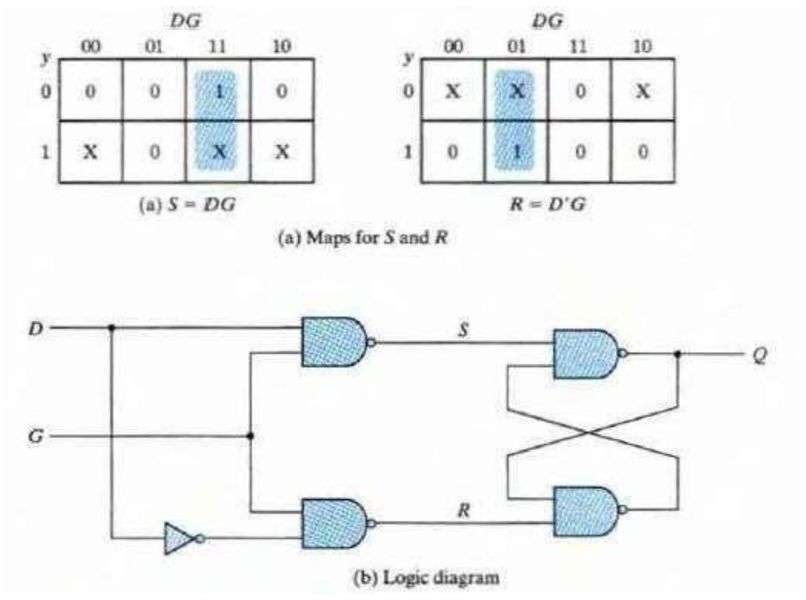
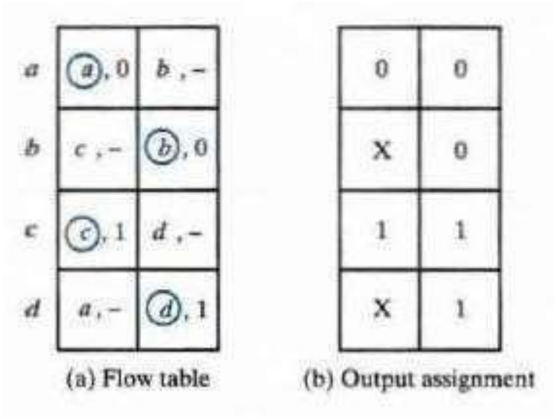


Fig 4.14 – Circuit with SR Latches

Image source from Digital Design by Moris Mano (Page No. 438)



a	ⓐ, 0	b , –
b	c , –	ⓑ, 0
c	ⓒ, 1	d , –
d	a , –	ⓓ, 1

(a) Flow table

0	0
X	0
1	1
X	1

(b) Output assignment

Fig 4.15 – Assigning output values to unstable states

Image source from Digital Design by Moris Mano (Page No. 438)

EC 8392 – DIGITAL ELECTRONICS

UNIT – III : SYNCHRONOUS SEQUENTIAL CIRCUITS

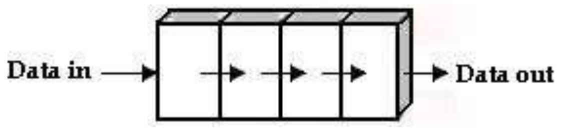
SHIFT REGISTERS:

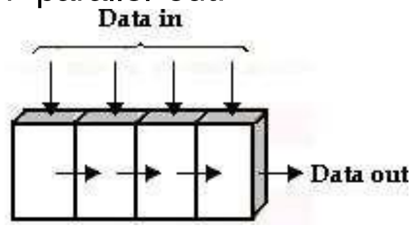
A register is simply a group of Flip-Flops that can be used to store a binary number. There must be one Flip-Flop for each bit in the binary number. For instance, a register used to store an 8-bit binary number must have 8 Flip-Flops.

The Flip-Flops must be connected such that the binary number can be entered (shifted) into the register and possibly shifted out. A group of Flip-Flops connected to provide either or both of these functions is called a ***shift register***.

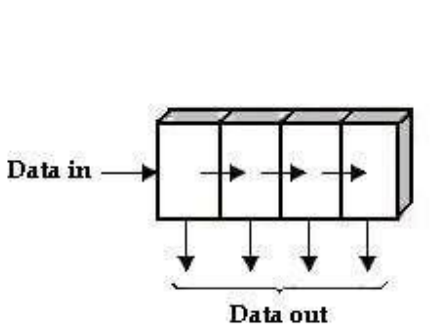
The bits in a binary number (data) can be removed from one place to another in either of two ways. The first method involves shifting the data one bit at a time in a serial fashion, beginning with either the most significant bit (MSB) or the least significant bit (LSB). This technique is referred to as *serial shifting*. The second method involves shifting all the data bits simultaneously and is referred to as *parallel shifting*. There are two ways to shift into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic register types—

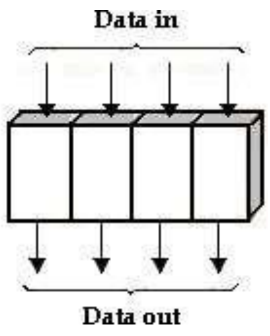
i. Serial in- serial out,
ii. Serial in- parallel out,
iii. Parallel in- serial out, iv. Parallel in- parallel out.





(i) Serial in- serial out (iii) Parallel in- serial out





(iii) Serial in- parallel out (iv) Parallel in- parallel out

Fig : 3.43 – Various types of Shift Registers

Serial-In Serial-Out Shift Register:

The serial in/serial out shift register accepts data serially, i.e., one bit at a time on a single line. It produces the stored information on its output also in serial form.

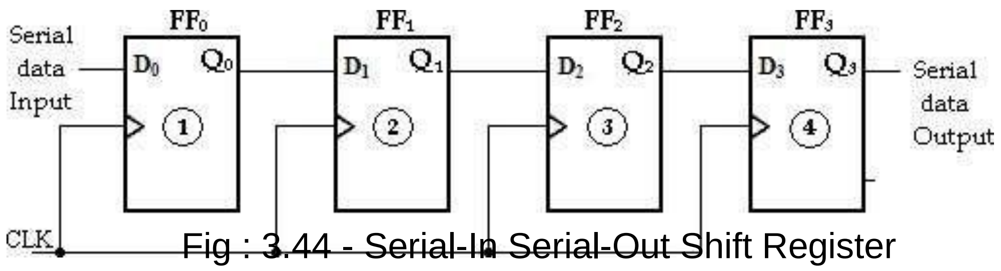


Fig : 3.44 - Serial-In Serial-Out Shift Register

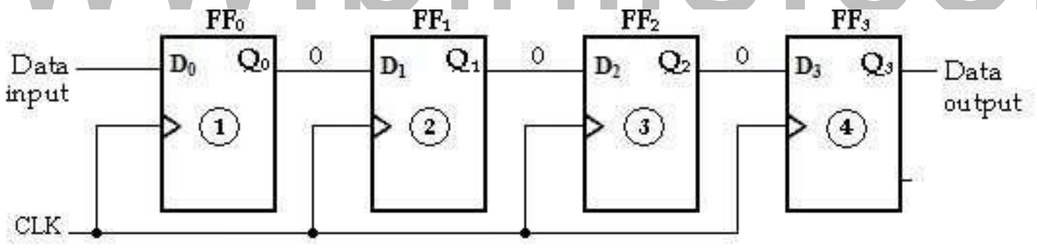
The entry of the four bits 1010 into the register is illustrated below, beginning with the right-most bit. The register is initially clear. The 0 is put onto the data input line, making D=0 for FF_0. When the first clock pulse is applied, FF_0 is reset, thus storing the 0.

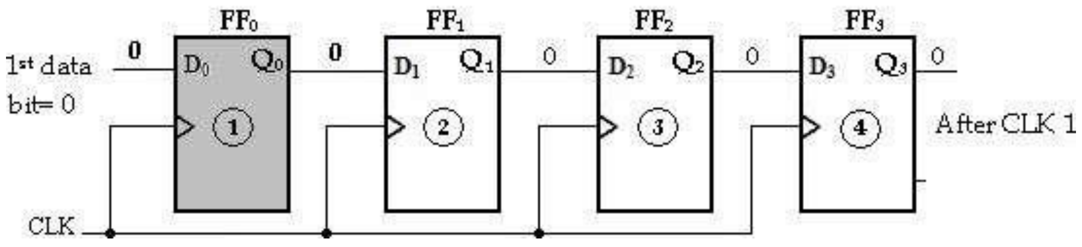
Next the second bit, which is a 1, is applied to the data input, making D=1 for FF_0 and D=0 for FF_1 because the D input of FF_1 is connected to the Q_0 output. When the second clock pulse occurs, the 1 on the data input is shifted into FF0, causing FF0 to set; and the 0 that was in FF0 is shifted into FFl.

The third bit, a 0, is now put onto the data-input line, and a clock pulse is applied. The 0 is entered into FF_0, the 1 stored in FF_0 is shifted into FF_l, and the 0 stored in FF_1 is shifted into FF_2.

The last bit, a 1, is now applied to the data input, and a clock pulse is applied.

This time the 1 is entered into FF0, the 0 stored in FF0 is shifted into FFl, the 1 stored in FF1 is shifted into FF2, and the 0 stored in FF2 is shifted into FF3. This completes the serial entry of the four bits into the shift register, where they can be stored for any length of time as long as the Flip-Flops have dc power.





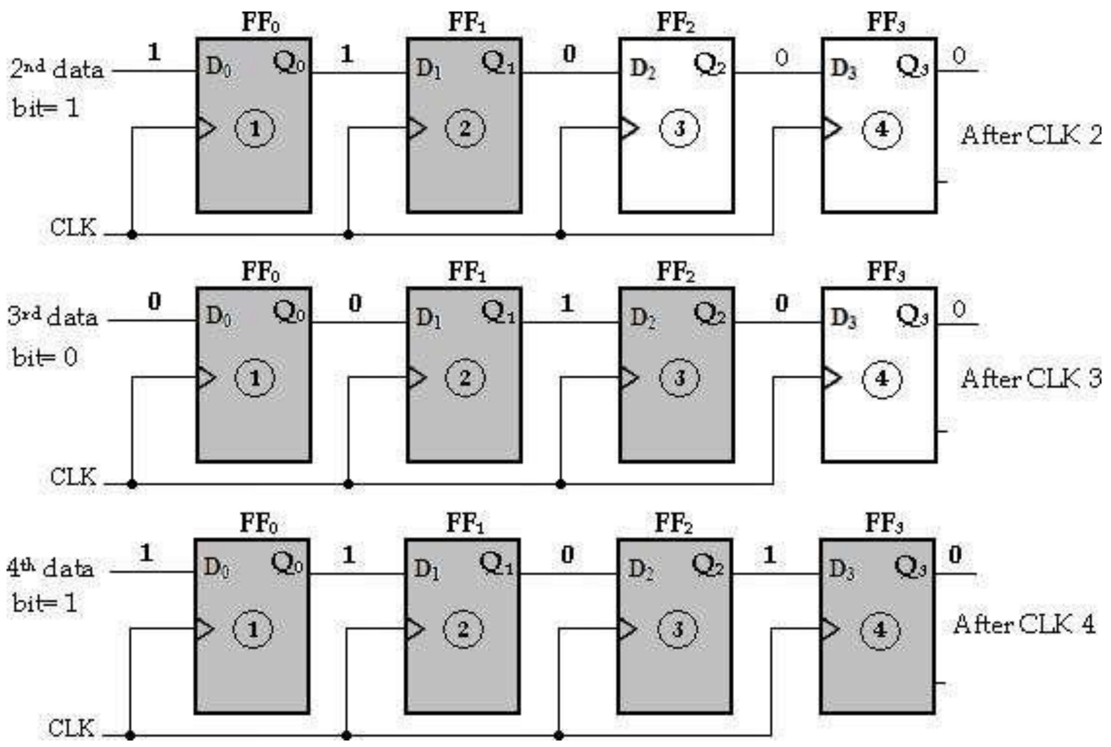


Fig : 3.45 - Four bits (1010) being entered serially into the register

To get the data out of the register, the bits must be shifted out serially and taken off the Q3 output. After CLK4, the right-most bit, 0, appears on the Q3 output.

When clock pulse CLK5 is applied, the second bit appears on the Q3 output. Clock pulse CLK6 shifts the third bit to the output, and CLK7 shifts the fourth bit to the output. While the original four bits are being shifted out, more bits can be shifted in. All zeros are shown being shifted out, more bits can be shifted in.

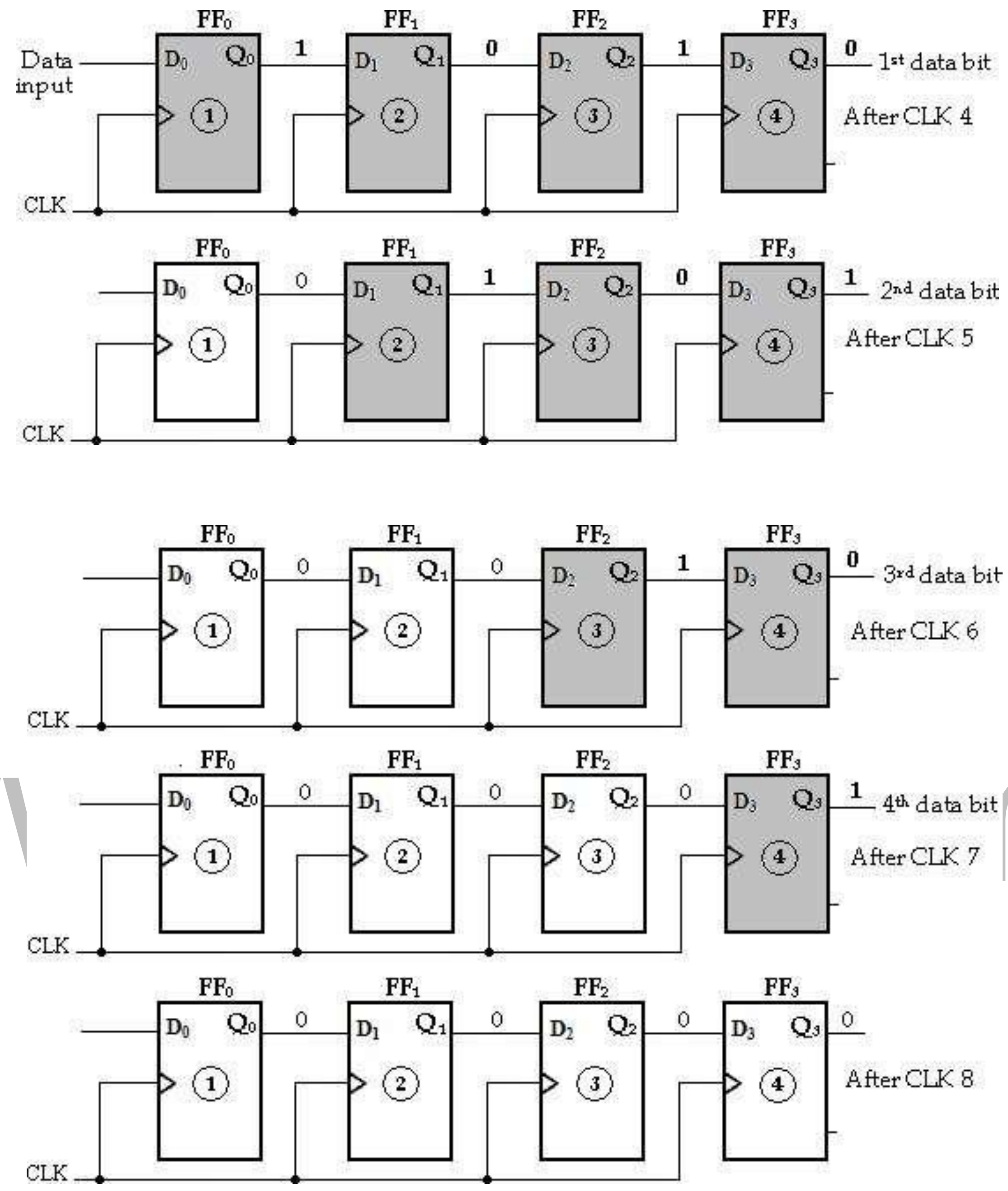


Fig : 3.46 - Four bits (1010) being entered serially-shifted out of the register and replaced by all zeros

Serial-In Parallel-Out Shift Register:

In this shift register, data bits are entered into the register in the same as serial-in serial-out shift register. But the output is taken in parallel. Once the data are stored, each bit appears on its respective output line and all bits are available simultaneously instead of on a bit-by-bit.

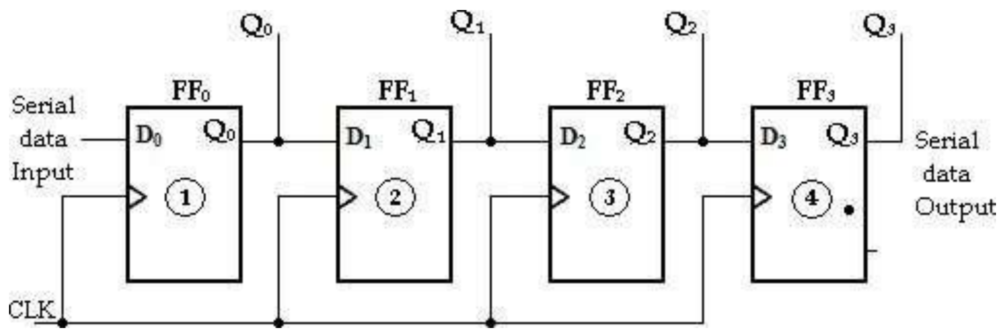
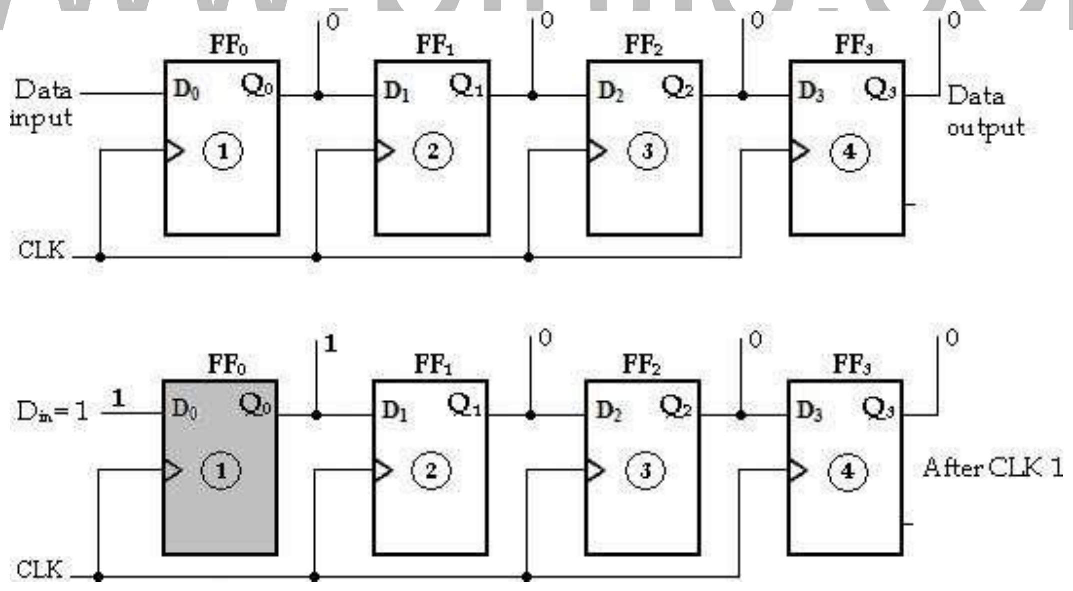


Fig : 3.47 - Serial-In parallel-Out Shift Register



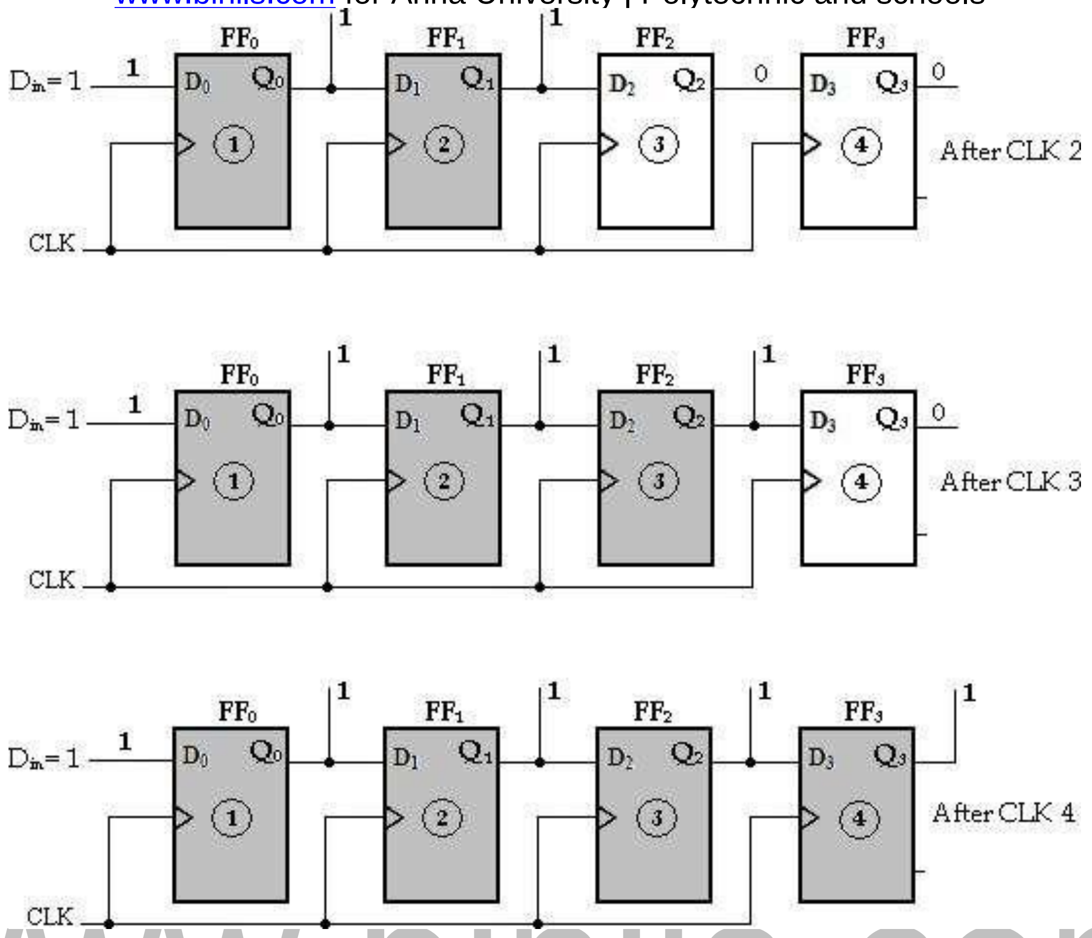


Fig : 3.47 - Four bits (1111) being serially entered into the register

Parallel-In Serial-Out Shift Register:

In this type, the bits are entered in parallel i.e., simultaneously into their respective stages on parallel lines.

A 4-bit parallel-in serial-out shift register is illustrated below. There are four data input lines, X_0, X_1, X_2 and X_3 for entering data in parallel into the register.

SHIFT/ LOAD input is the control input, which allows four bits of data to **load** in parallel into the register.

When SHIFT/LOAD is LOW, gates G_1, G_2, G_3 and G_4 are enabled, allowing each data bit to be applied to the D input of its respective Flip-Flop. When a clock pulse is applied, the Flip-Flops with D = 1 will **set** and those with D = 0 will **reset,** thereby storing all four bits simultaneously.

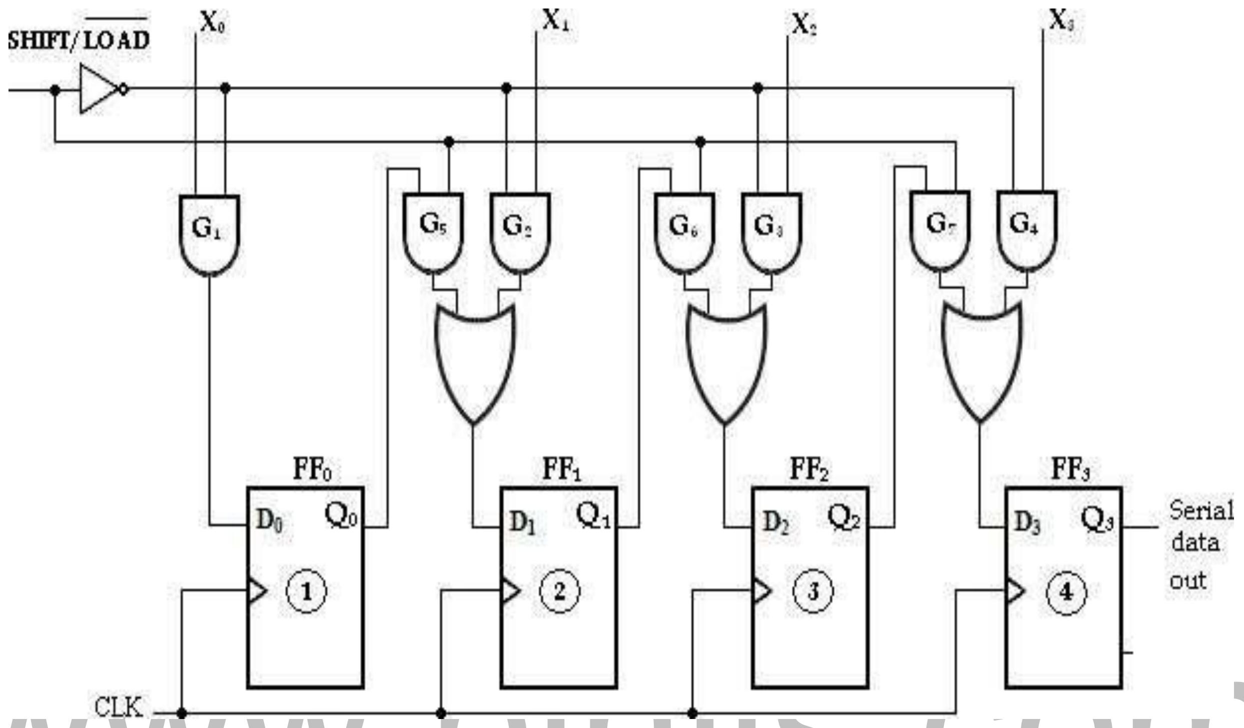


Fig : 3.48 - Parallel-In Serial-Out Shift

When SHIFT/LOAD is HIGH, gates G_1, G_2, G_3 and G_4 are disabled and gates G_5,

G_6 and G_7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data- entry operation, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

Parallel-In Parallel-Out Shift Register:

In this type, there is simultaneous entry of all data bits and the bits appear on

parallel outputs simultaneously.

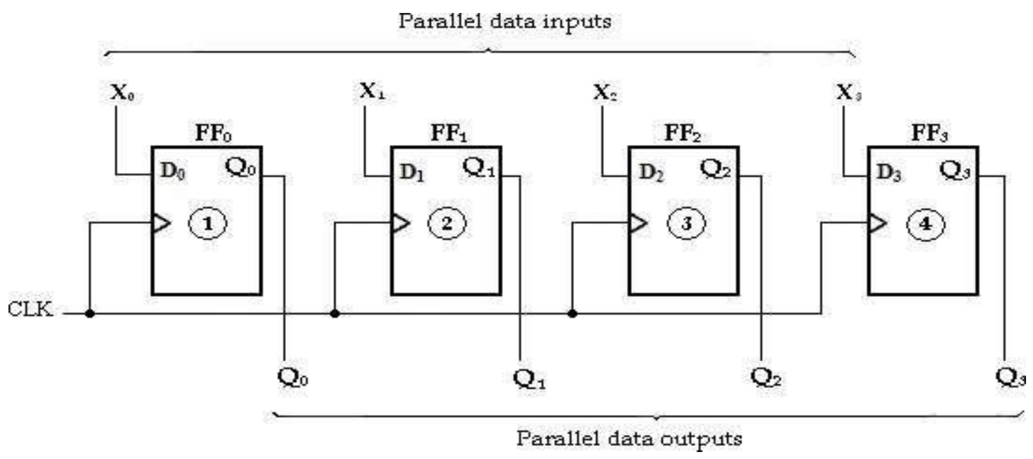


Fig : 3.49 - Parallel-In Parallel-Out Shift Register

UNIVERSAL SHIFT REGISTERS

If the register has shift and parallel load capabilities, then it is called a shift register with parallel load or ***universal shift register***. Shift register can be used for converting serial data to parallel data, and vice-versa. If a parallel load capability is added to a shift register, the data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

The functions of universal shift register are:

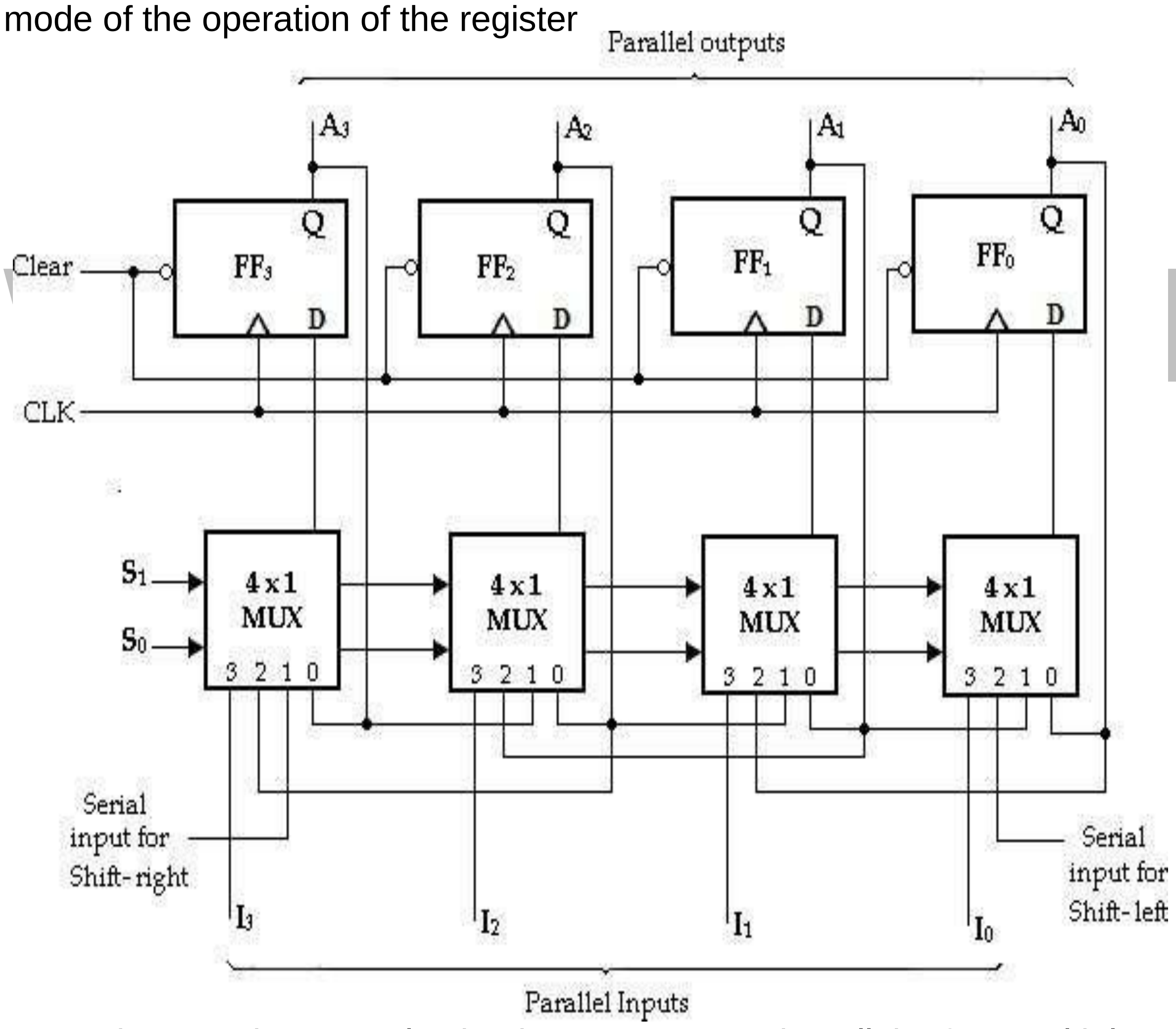
1. A clear control to clear the register to 0.
2. A clock input to synchronize the operations.
3. A shift-right control to enable the shift right operation and the serial input and output lines associated with the shift right.
4. A shift-left control to enable the shift left operation and the serial input and output lines associated with the shift left.
5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
6. ‘n’ parallel output lines.

7. A control line that leaves the information in the register unchanged even though the clock pulses re continuously applied.

It consists of four D-Flip-Flops and four 4 input multiplexers (MUX). S_0

selection inputs are used to select one of the four inputs of each multiplexer.

The input 0 in each MUX is selected when S_1S_0= 00 and input 1 is selected when S_1S_0= 01. Similarly inputs 2 and 3 are selected when S_1S_0= 10 and S_1S_0= 11 respectively. The inputs S1 and S0 control the mode of the operation of the register



and S_1 are the two selection inputs connected to all the four multiplexers. These two

Fig : 3.50 - 4-Bit Universal Shift Register

When S_1S_0= 00, the present value of the register is applied to the D-inputs of the Flip-Flops. This is done by connecting the output of each Flip-Flop to the 0 input of the respective multiplexer. The next clock pulse transfers into each Flip-Flop, the binary value is held previously, and hence no change of state occurs.

When S_1S_0= 01, terminal 1 of the multiplexer inputs has a path to the D inputs of the Flip-Flops. This causes a shift-right operation with the lefter serial input transferred into Flip-Flop FF_3.

When S_1S_0= 10, a shift-left operation results with the right serial input going into Flip-Flop FF_1.

Finally when S_1S_0= 11, the binary information on the parallel input lines (I_1, I_2, I_3 and I_4) are transferred into the register simultaneously during the next clock pulse.

The function table of bi-directional shift register with parallel inputs and parallel outputs is shown below.

Mode Control		**Operation**
S 1	**S 0**	
0	0	No change
0	1	Shift-right
1	0	Shift-left
1	1	Parallel load

BI-DIRECTION SHIFT REGISTERS:

A bidirectional shift register is one in which the data can be shifted either left or right. It can be implemented by using gating logic that enables the transfer of a data bit from one stage to the next stage to the right or to the left depending on the level of a control line.

A 4-bit bidirectional shift register is shown below. A HIGH on the RIGHT/LEFT control input allows data bits inside the register to be shifted to the right, and a LOW enables data bits inside the register to be shifted to the left.

When the RIGHT/LEFT control input is **HIGH**, gates G_1, G_2, G_3 and G_4 are enabled, and the state of the Q output of each Flip-Flop is passed through to the D input of the following Flip-Flop. When a clock pulse occurs, the data bits are shifted one place to the right.

When the RIGHT/LEFT control input is **LOW**, gates G_5, G_6, G_7 and G_8 are enabled, and the Q output of each Flip-Flop is passed through to the D input of the preceding Flip-Flop. When a clock pulse occurs, the data bits are then shifted one place to the left.

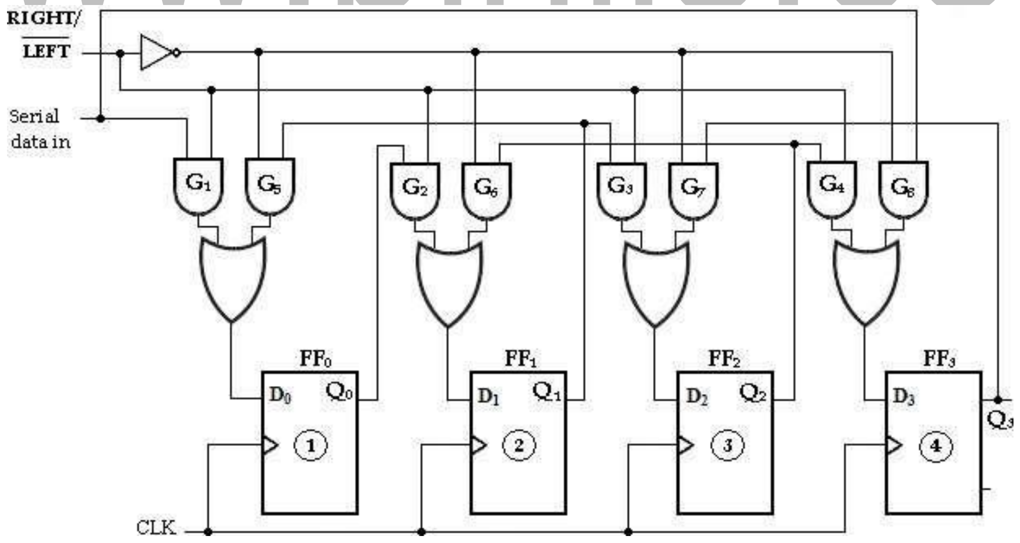


Fig : 3.51 - 4-bit bi-directional shift register

EC 8392 – DIGITAL ELECTRONICS

UNIT – III : SYNCHRONOUS SEQUENTIAL CIRCUITS

TRIGGERING OF FLIP-FLOPS

The state of a Flip-Flop is switched by a momentary change in the input signal. This momentary change is called a trigger and the transition it causes is said to trigger the Flip-Flop. Clocked Flip-Flops are triggered by pulses. A clock pulse starts from an initial value of 0, goes momentarily to 1and after a short time, returns to its initial 0 value.

Latches are controlled by enable signal, and they are level triggered, either positive level triggered or negative level triggered. The output is free to change according to the S and R input values, when active level is maintained at the enable input.

Flip-Flops are different from latches. Flip-Flops are pulse or clock edge triggered instead of level triggered.

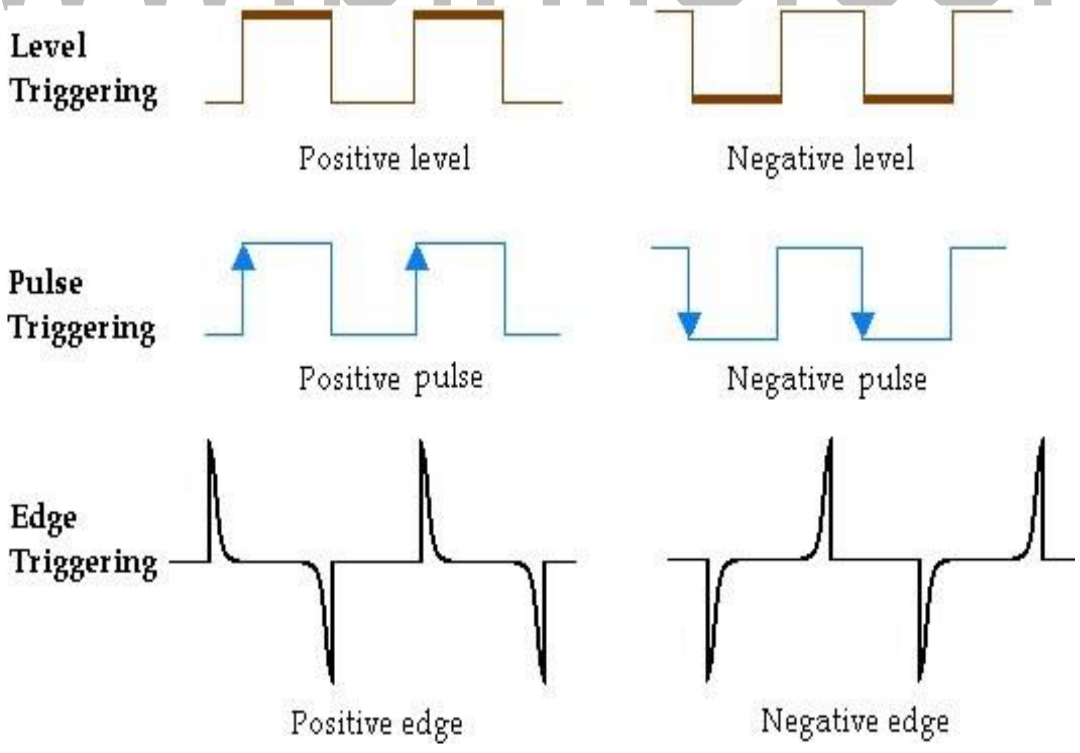


Fig : 3.12 – Various types of triggering

EDGE TRIGGERED FLIP-FLOPS

Flip-Flops are synchronous bistable devices (has two outputs Q and Q’). In this case, the term synchronous means that the output changes state only at a specified point on the triggering input called the clock (CLK), i.e., changes in the output occur in synchronization with the clock.

An ***edge-triggered Flip-Flop*** changes state either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse and is sensitive to its inputs only at this transition of the clock. The different types of edge-triggered Flip- Flops are—xS-R Flip-Flop, xJ-K Flip-Flop, xD Flip-Flop, xT Flip- Flop.

Although the S-R Flip-Flop is not available in IC form, it is the basis for the D and J-K Flip-Flops. Each type can be either positive edge-triggered (no bubble at C input) or negative edge-triggered (bubble at C input). The key to identifying an edge- triggered Flip-Flop by its logic symbol is the small triangle inside the block at the clock (C) input. This triangle is called the **dynamic input indicator**.

S-R Flip-Flop

The S and R inputs of the S-R Flip-Flop are called ***synchronous*** inputs because data on these inputs are transferred to the Flip-Flop's output only on the triggering edge of the clock pulse. The circuit is similar to SR latch except enable signal is replaced by clock pulse (CLK). On the positive edge of the clock pulse, the circuit responds to the S and R inputs.

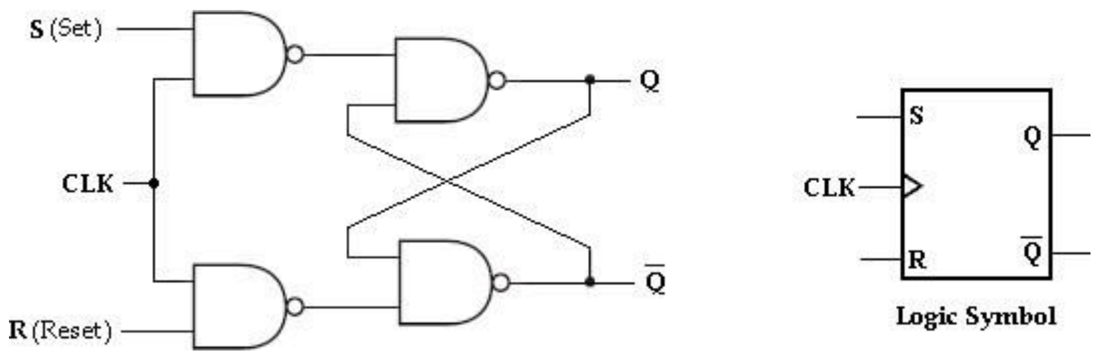


Fig : 3.13 - SR Flip-Flop

When S is HIGH and R is LOW, the Q output goes HIGH on the triggering edge of the clock pulse, and the Flip-Flop is SET. When S is LOW and R is HIGH, the Q output goes LOW on the triggering edge of the clock pulse, and the Flip- Flop is RESET. When both S and R are LOW, the output does not change from its prior state.

An invalid condition exists when both S and R are HIGH.

CLK	S	R	Qn	Qn+1	State
1	0	0	0	0	No Change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	x x	Indeterminate *
1	1	1	1		
0	x	x	0	0	No Change (NC)
0	x	x	1	1	

Truth table for SR Flip-Flop

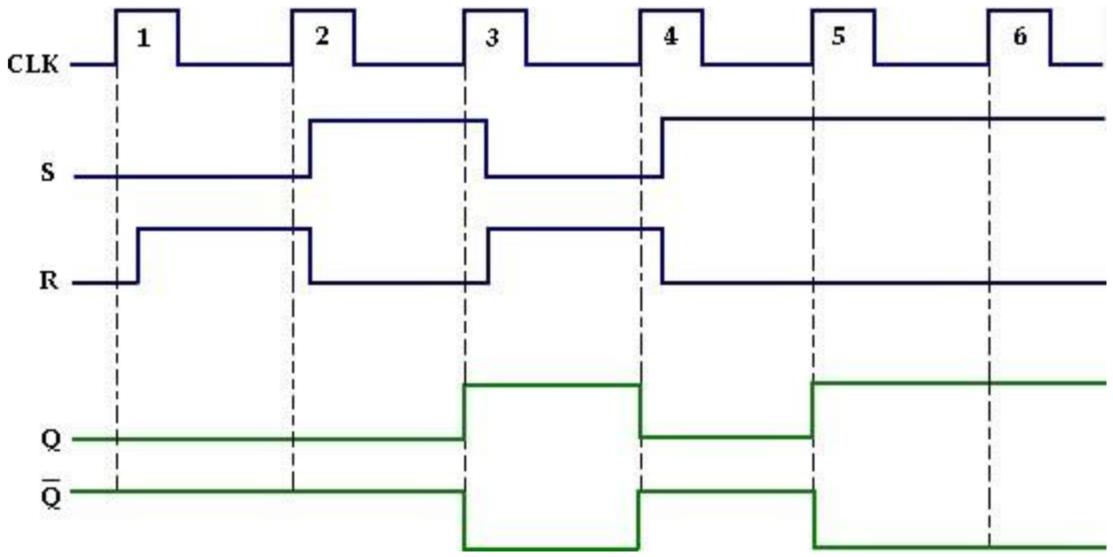


Fig : 3.14 - Input and output waveforms of SR Flip-Flop

J-K Flip-Flop:

JK means Jack Kilby, Texas Instrument (TI) Engineer, who invented IC in 1958. JK Flip-Flop has two inputs J(set) and K(reset). A JK Flip-Flop can be obtained from the clocked SR Flip-Flop by augmenting two AND gates as shown below.

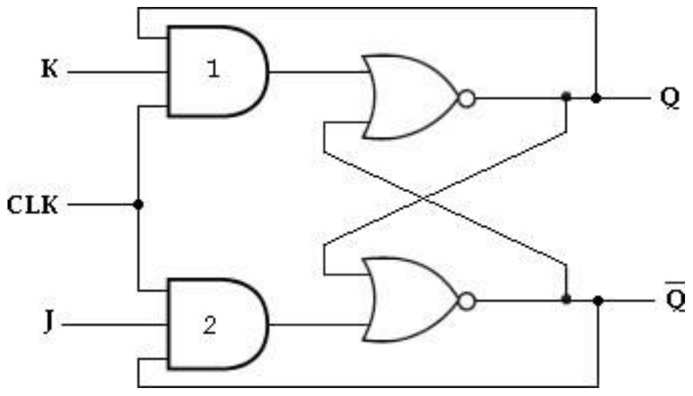


Fig : 3.15 - JK Flip Flop

The data input J and the output Q' are applied o the first AND gate and its output (JQ') is applied to the S input of SR Flip-Flop. Similarly, the data input K and the output Q are applied to the second AND gate and its output (KQ) is applied to the R input of SR Flip-Flop.

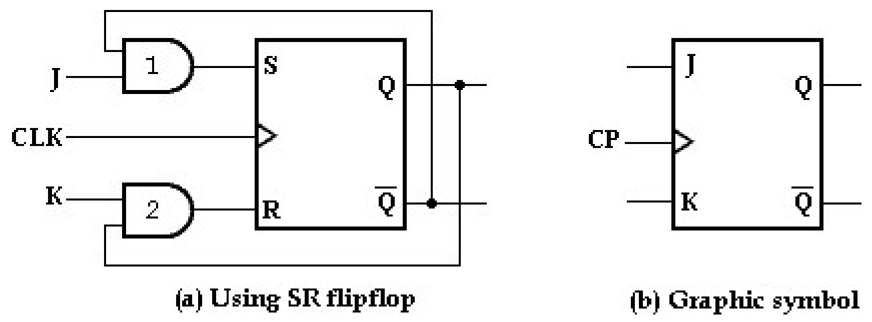


Fig : 3.16 – JK Flipflop using SR Flipflop

J= K= 0

When J=K= 0, both AND gates are disabled. Therefore clock pulse have no effect, hence the Flip-Flop output is same as the previous output.

J= 0, K= 1

When J= 0 and K= 1, AND gate 1 is disabled i.e., S= 0 and R= 1. This condition will reset the Flip-Flop to 0.

J= 1, K= 0

When J= 1 and K= 0, AND gate 2 is disabled i.e., S= 1 and R= 0. Therefore the Flip-Flop will set on the application of a clock pulse.

J= K= 0

When J=K= 1, it is possible to set or reset the Flip-Flop. If Q is High, AND gate 2 passes on a reset pulse to the next clock. When Q is low, AND gate 1 passes on a set pulse to the next clock. Eitherway, Q changes to the complement of the last state i.e., toggle. Toggle means to switch to the opposite state.

The truth table of JK Flip-Flop is given below.

CLK	**Inputs**		**Output**	**State**
	J	**K**	**Qn+1**	
1	0	0	Qn	No Change
1	0	1	0	Reset

1	1	0	1	Set
1	1	1	Q_n'	Toggle

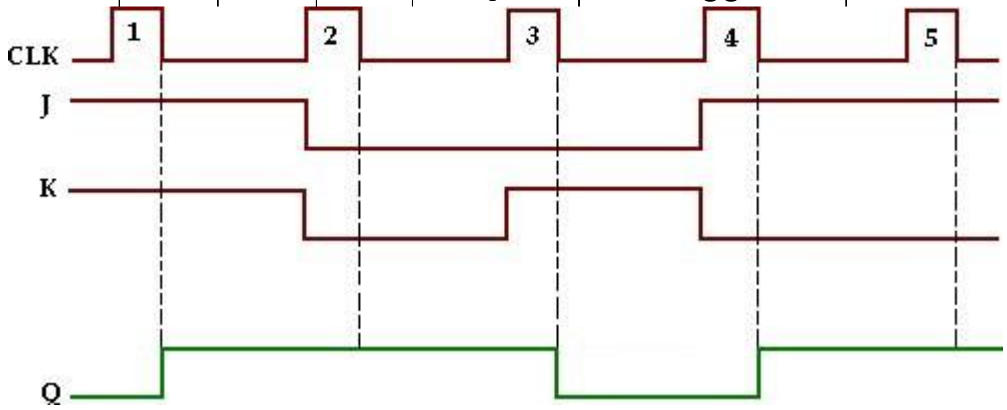


Fig : 3.17 - Input and output waveforms of JK Flip-Flop

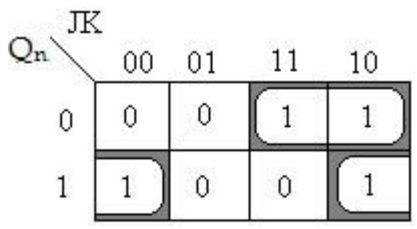
Characteristic table and Characteristic equation:

The characteristic table for JK Flip-Flop is shown in the table below. From the table, K-map for the next state transition (Q_{n+1}) can be drawn and the simplified logic expression which represents the characteristic equation of JK Flip-Flop can be found.

Qn	**J**	**K**	**Qn+1**
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Characteristic table

K-map Simplification:

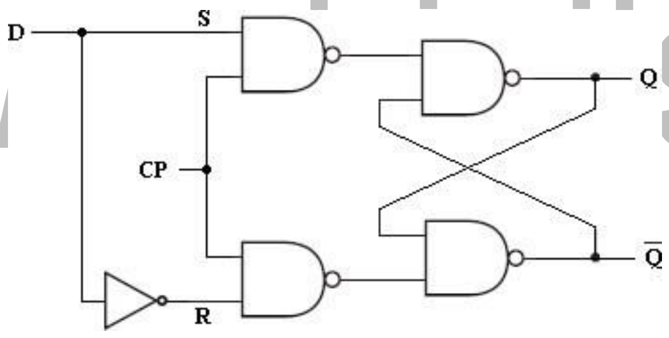


Characteristic equation: $\mathbf{Q_{n+1} = JQ' + K'Q}$.

D Flip-Flop:

Like in D latch, in D Flip-Flop the basic SR Flip-Flop is used with complemented inputs. The D Flip-Flop is similar to D-latch except clock pulse is used instead of enable input.

Fig : 3.18 - D Flip-Flop



To eliminate the undesirable condition of the indeterminate state in the RS Flip-Flop is to ensure that inputs S and R are never equal to 1 at the same time. This is done by D Flip-Flop. The D (***delay***) Flip-Flop has one input called delay input and clock pulse input. The D Flip-Flop using SR Flip-Flop is shown below.

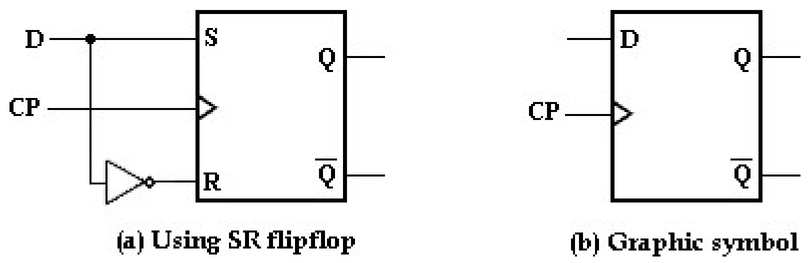


Fig : 3.19 – D Flipflop using SR Flipflop The truth table of D Flip-Flop is given below.

Clock	D	Qn+1	State
1	0	0	Reset
1	1	1	Set
0	x	Qn	No Change

Truth table for D Flip-Flop

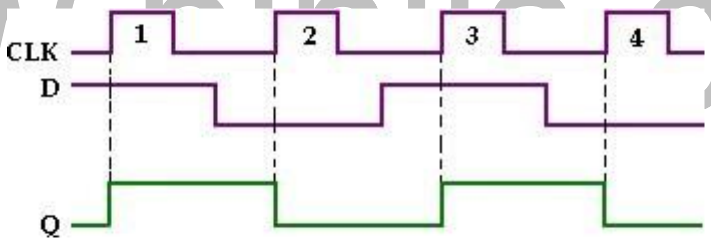


Fig : 3.20 - Input and output waveforms of clocked D Flip-Flop

Looking at the truth table for D Flip-Flop we can realize that Q_{n+1} function follows the D input at the positive going edges of the clock pulses.

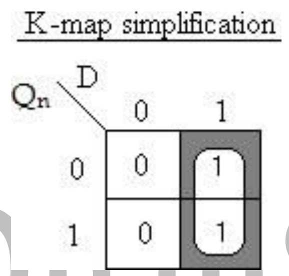
Characteristic table and Characteristic equation:

The characteristic table for D Flip-Flop shows that the next state of the Flip- Flop is independent of the present state since Q_{n+1} is equal to D. This means that an input pulse will transfer the value of input D into the output of the Flip-

Flop independent of the value of the output before the pulse was applied. The characteristic equation is derived from K-map.

Qn	D	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

Characteristic table

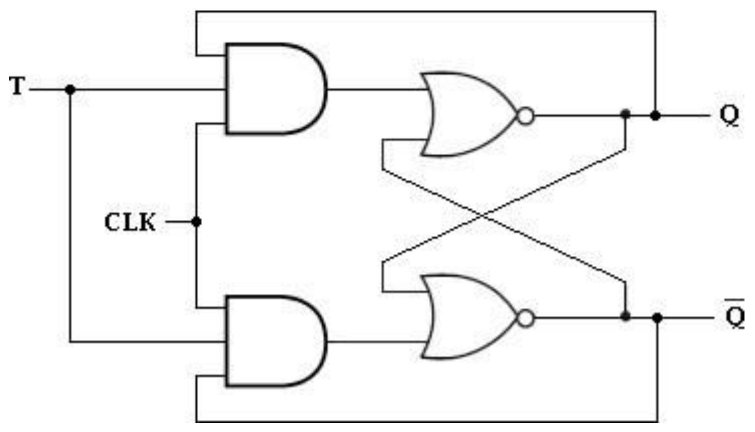


Characteristic equation: **Q_{n+1}=**

D. T Flip-Flop

The T (***Toggle***) Flip-Flop is a modification of the JK Flip-Flop. It is obtained from JK Flip-Flop by connecting both inputs J and K together, i.e., single input.

Regardless of the present state, the Flip-Flop complements its output when the clock pulse occurs while input T= 1.



When T= 0, Q_{n+1}= Q_n, ie., the next state is the sameas the present state and no change occurs. When T= 1, Qn+1= Qn’,ie., the next state is the complement of the present state.

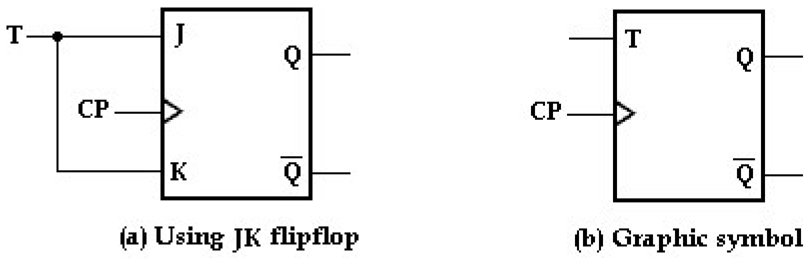


Fig : 3.22 – T flipflop using JK Flipflop The truth table of T Flip-Flop is given below.

T	Qn+1	State
0	Qn	No Change
1	Qn’	Toggle

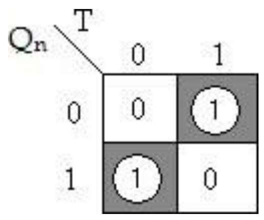
Truth table for T Flip-Flop

Characteristic table and Characteristic equation:

The characteristic table for T Flip-Flop is shown below and characteristic equation is derived using K-map.

Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

K-map Simplification:



Characteristic equation: $\mathbf{Q_{n+1} = TQ_n' + T'Q_n}$.

Master-Slave JK Flip-Flop

A master-slave Flip-Flop is constructed using two separate JK Flip-Flops. The first Flip-Flop is called the master. It is driven by the positive edge of the clock pulse. The second Flip-Flop is called the slave. It is driven by the negative edge of the clock pulse. The logic diagram of a master-slave JK Flip-Flop is shown below.

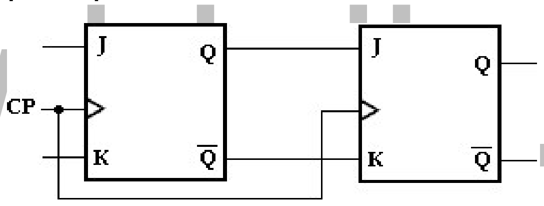


Fig : 3.32 - Logic diagram

When the clock pulse has a positive edge, the master acts according to its J- K inputs, but the slave does not respond, since it requires a negative edge at the clock input.

When the clock input has a negative edge, the slave Flip-Flop copies the master outputs. But the master does not respond since it requires a positive edge at its clock input. The clocked master-slave J-K Flip-Flop using NAND gates is shown below.

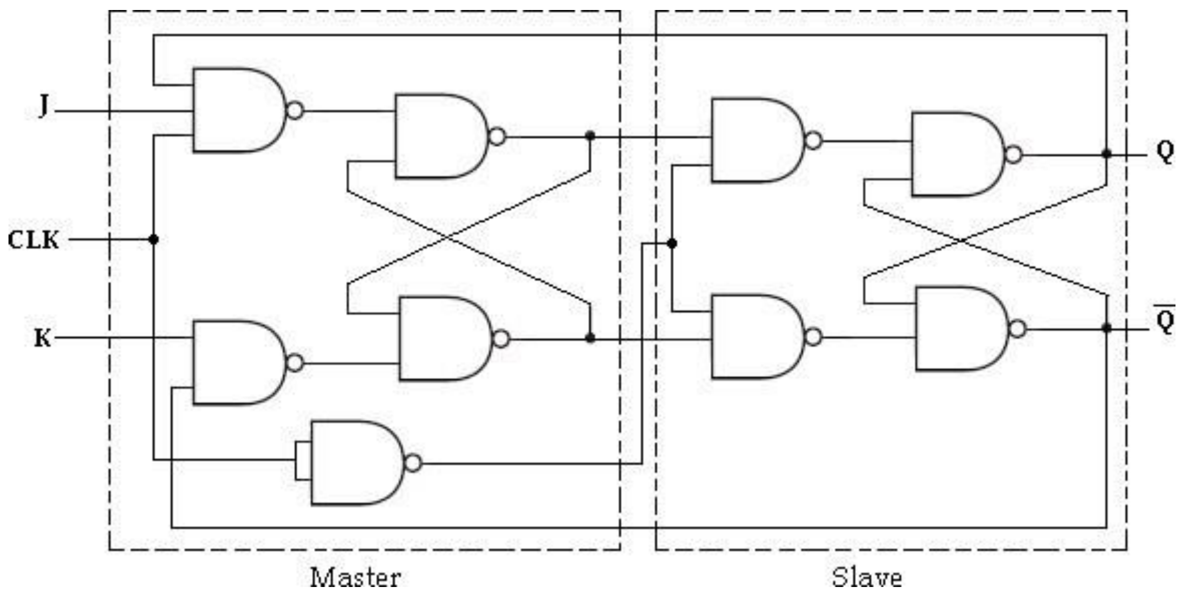


Fig : 3.33 - Master-Slave JK Flip-Flop

APPLICATION TABLE (OR) EXCITATION TABLE:

The ***characteristic table*** is useful for **analysis** and for defining the operation of the Flip-Flop. It specifies the next state (Q_{n+1}) when the inputs and present state are known.

The ***excitation or application table*** is useful for **design** process. It is used to find the Flip-Flop input conditions that will cause the required transition, when the present state (Q_n) and the next state (Q_{n+1}) are known.

SR Flip-Flop:

Present State	Inputs		Next State
Qn	S	R	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	x

Characteristic Table

Present State	Next State	Inputs		Inputs	
Qn	Qn+1	S	R	S	R
0	0	0	0	0	x
0	0	0	1		
0	1	1	0	1	0
1	0	0	1	0	1
1	1	0	0	x	0
1	1	1	0		

Modified Table

Present State	Next State	Inputs	
Qn	Qn+1	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Excitation Table

The above table presents the excitation table for SR Flip-Flop. It consists of present state (Q_n), next state (Q_{n+1}) and a column for each input to show how the required transition is achieved.

There are 4 possible transitions from present state to next state. The required Input conditions for each of the four transitions are derived from the information available in the characteristic table. The symbol 'x' denotes the don't care condition, it does not matter whether the input is 0 or 1.

JK Flip-Flop:

Present State	Inputs		Next State
Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Characteristic Table

Present State	Next State	Inputs		Inputs	
Qn	Qn+1	J	K	J	K
0	0	0	0	0	x
0	0	0	1		
0	1	1	0	1	x
0	1	1	1		
1	0	0	1	x	1
1	0	1	1		
1	1	0	0	x	0
1	1	1	0		

Modified Table

Present State	Next State	Inputs	
Qn	**Qn+1**	**J**	**K**
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Excitation Table

D Flip-Flop

Present State	Input	Next State
Qn	**D**	**Qn+1**
0	0	0
0	1	1
1	0	0
1	1	1

Characteristic Table

Present State	Next State	Input
Qn	**Qn+1**	**D**
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

T Flip-Flop

Present State	Input	Next State
Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table

Present State	Next State	Input
Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

Modified Table

REALIZATION OF ONE FLIP-FLOP USING OTHER FLIP-FLOPS

It is possible to convert one Flip-Flop into another Flip-Flop with some additional gates or simply doing some extra connection. The realization of one Flip- Flop using other Flip-Flops is implemented by the use of characteristic tables and excitation tables. Let us see few conversions among Flip-Flops.

SR Flip-Flop to D Flip-
Flop SR Flip-Flop to JK
Flip-Flop SR Flip-Flop to
T Flip-Flop JK Flip-Flop
to T Flip-Flop JK Flip-
Flop to D Flip-Flop D
Flip-Flop to T Flip-Flop
T Flip-Flop to D Flip-Flop

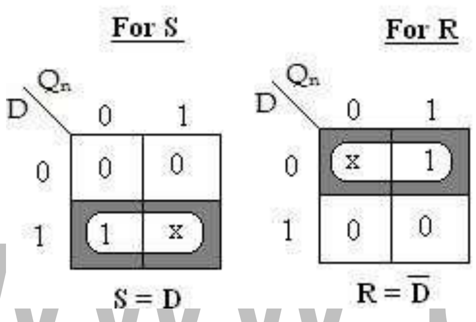
SR Flip-Flop to D Flip-Flop:

1. Write the characteristic table for required Flip-Flop (D Flip-Flop). x Write the excitation table for given Flip-Flop (SR Flip-Flop).
2. Determine the expression for the given Flip-Flop inputs (S and R) by using K- map.
3. Draw the Flip-Flop conversion logic diagram to obtain the required Flip- Flop (D Flip-Flop) by using the above obtained expression.

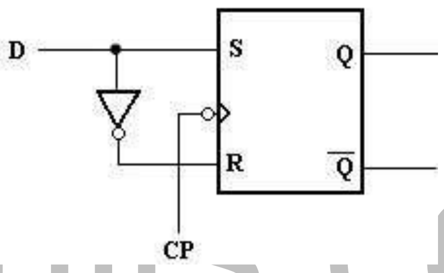
The excitation table for the above conversion is

Required Flip-Flop (D)			Given Flip-Flop (SR)	
Input	Present state	Next state	Flip-Flop Inputs	
D	Qn	Qn+1	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

K-map simplification



Logic diagram

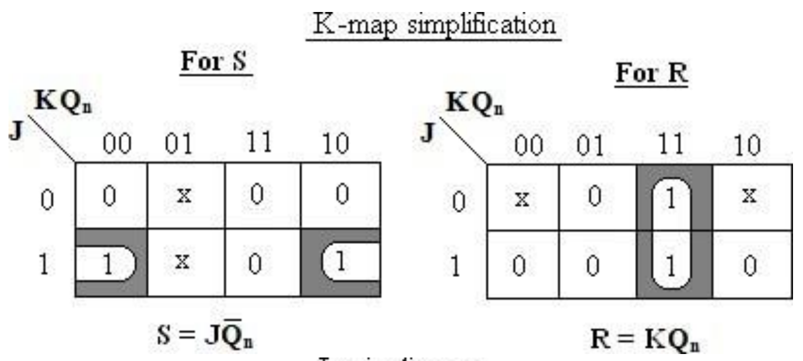


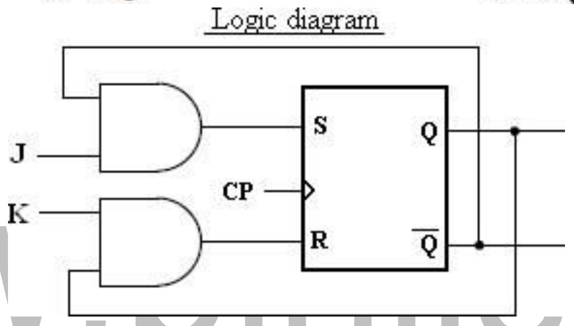
SR Flip-Flop to JK Flip-Flop

The excitation table for the above conversion is,

Inputs		Present state	Next state	Flip-Flop Input	
J	K	Qn	Qn+1	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0

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

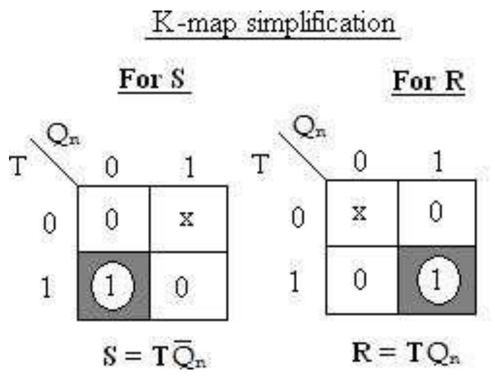


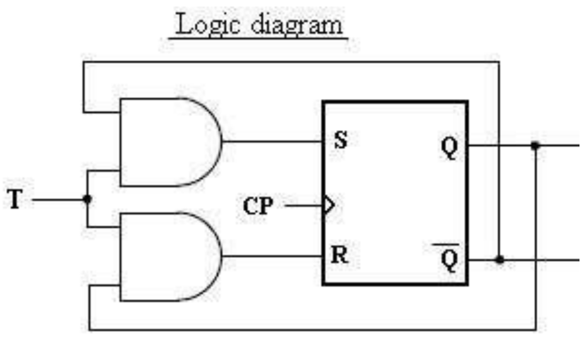


SR Flip-Flop to T Flip-Flop

The excitation table for the above conversion is

Input	Present state	Next state	Flip-Flop Inputs	
T	Qn	Qn+1	S	R
0	0	0	0	x
0	1	1	x	0
1	0	1	1	0
1	1	0	0	1

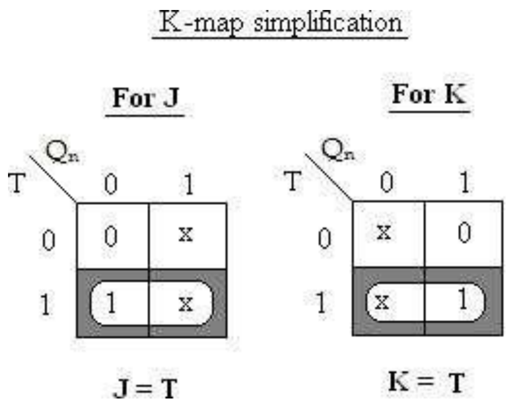


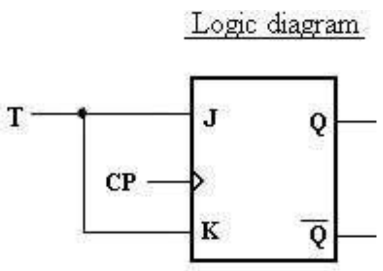


JK Flip-Flop to T Flip-Flop

The excitation table for the above conversion is

Input	Present state	Next state	Flip-Flop Inputs	
T	Qn	Qn+1	J	K
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1

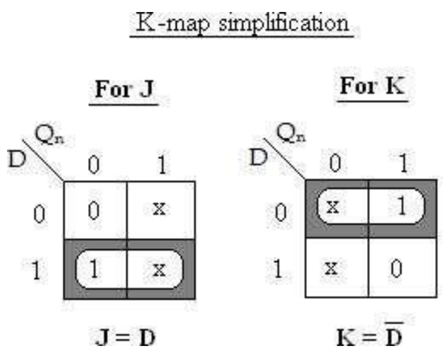


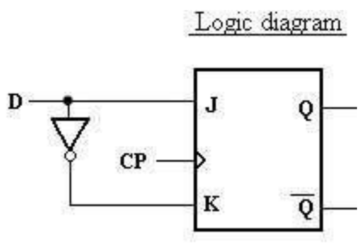


JK Flip-Flop to D Flip-Flop

The excitation table for the above conversion is

Input	Present state	Next state	Flip-Flop Inputs	
D	Qn	Qn+1	J	K
0	0	0	0	x
0	1	0	x	1
1	0	1	1	x
1	1	1	x	0



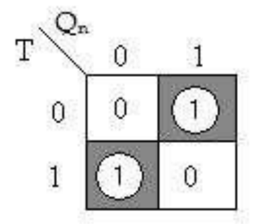


D Flip-Flop to T Flip-Flop

The excitation table for the above conversion is

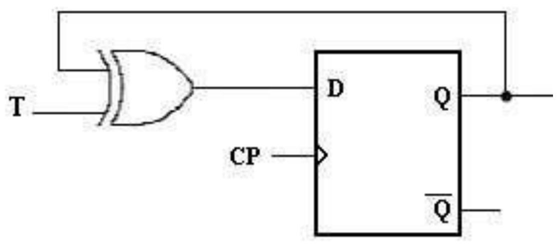
Inp ut	Present state	Next state	Flip-Flop Input
T	Qn	Qn+1	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

K-map simplification



$$D = \overline{T}Q_n + T\overline{Q}_n$$
$$= T \oplus Q_n$$

Logic diagram

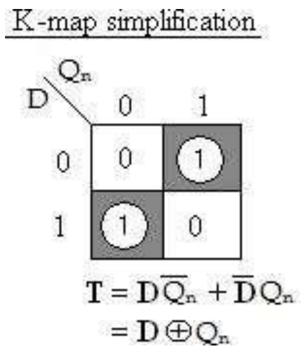


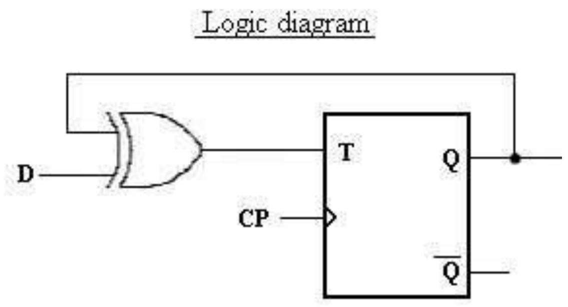
T Flip-Flop to D Flip-Flop

The excitation table for the above conversion is

Inpu t	Present state	Next state	Flip-Flop Input
D	Qn	Qn+1	T

1 1	0 1	1 1	1 0


K-map simplification
Qn
D
0
1
0
1
0
1
1
0
T = D$\overline{Q}_n$ + $\overline{D}$Qn
= D ⊕ Qn


Logic diagram
D
T
Q
CP
$\overline{Q}$