UNIT - 3

1. Flip-Flops
2. Gated Latches
3. Masterslave - Flip-Flops
4. Edge triggering
5. T-Flip Flops
6. Registers and shift registers
7. Registers and shift registers
8. Counters
9. Decoders
10. Multiplexers
11. Programable logic devices (PLDs)
12. Programable array logic (PAL)
13. Complex Programable logic devices (CPLDs)
14. Field programable gate array (FPGA)
15. Sequencial circuits
16. Up-down counters
17. Timing Diagrams
18. The finite state machine model
19. Synthesis of finite state machines

Flip-Flop :

→ A Flip-flop is a logical circuit used in digital electronics with two stable states to store binary data.

→ Flip-Flops along with latches are used for data communication and performing different operations inside the systems.

→ Both are used as data storage elements.

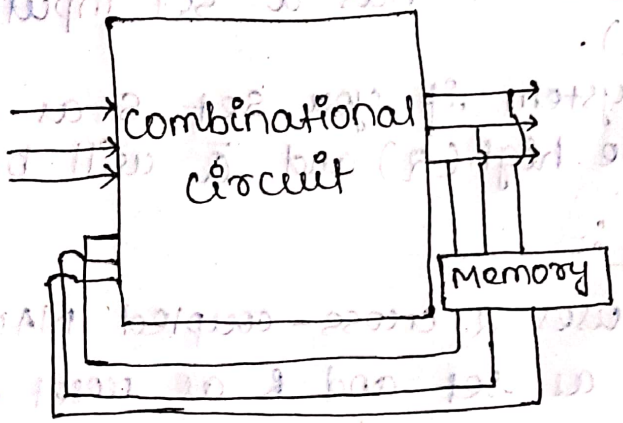
→ These works as gating and clocking mechanisms.

A sequencial circuit that consists of two stable states which is used to store binary data that can be changed by applying varying inputs.

Definition of Sequencial Circuit :-

→ In a sequencial circuit the present output depend on the present input as well as past output or outputs.

→ In means the sequencial circuit keeps track of previous output and helps to generate present output.



→ Flip-Flops are edge triggered and Latches are leve triggered.

→ Both work as temporary data storage elements.

→ Both can be implemented using various digital logic gates like - AND, OR, NOT, NAND or NOR gate

→ Latches can work as data storage element, control circuit, flip-flop circuit, state machines and memory elements.

→ There are two types of latches available -
  i) S-R Latch (Set, Reset Latch)
  ii) D Latch (Data Latch)

→ Flip-Flops only changes states when a control signal goes from high to low or low to high. But in Latches output can change as soon as the input changes.

→ Flip-Flop is a synchronous circuit uses a clock.

→ It can be also said as gated of clocked Latch.

→ Flip-Flops can be of different types -
  i) S-R Flip-Flop
  ii) J-K Flip-Flop
  iii) D Flip-Flop
  iv) T Flip-Flop

i) S-R Flip-Flop :-

→ It is the most common type of flip-flop.

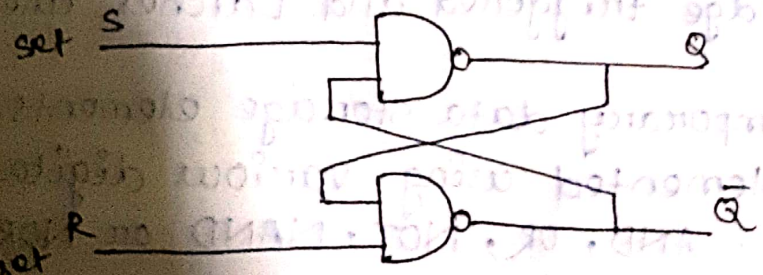
→ This flip-flop circuit has a set input(S) and reset input (R).

→ Here in this system if you set S as active the output would be high(Q) and $\bar{Q}$ will be low.

* NAND S-R Latch :-

→ NAND SR Latch uses 2 cross-coupled NAND gates.

→ uses 2 inputs S as set and R as reset.



Truth Table :-

S	R	Q(n+1)
0	0	Invalid
0	1	1
1	0	0
1	1	Hold (P

check

S=0, R=0

S 0

R 0

S=0, R=1

S 0

R 1

S=1, R=0

S 1

R 0

S=1, R=1

suppose

$\bar{Q}=0$

Q = 1

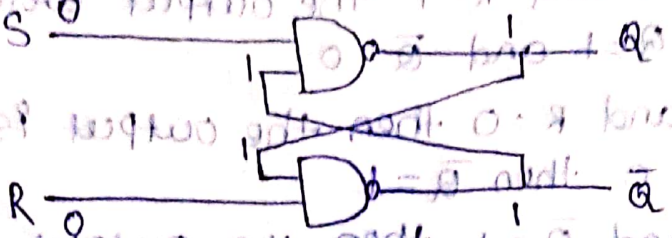
Truth Table :

S	R	Q(n+1)
0	0	Invalid
0	1	1
1	0	0
1	1	Hold (previous output)

[$\because \overline{1 \cdot Q} = \bar{1} + \bar{Q}$ so $0 + Q = Q$
$\overline{Q \cdot 1} = \bar{Q} + \bar{1} = \bar{Q} + 0 = \bar{Q}$.]

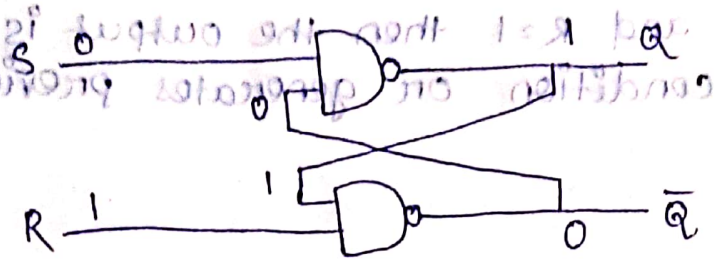
Check

S = 0, R = 0



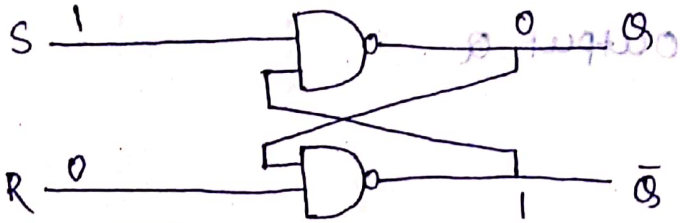
As $Q = \bar{Q}$, so output is invalid

S = 0, R = 1



As Q = 1, so output is 1

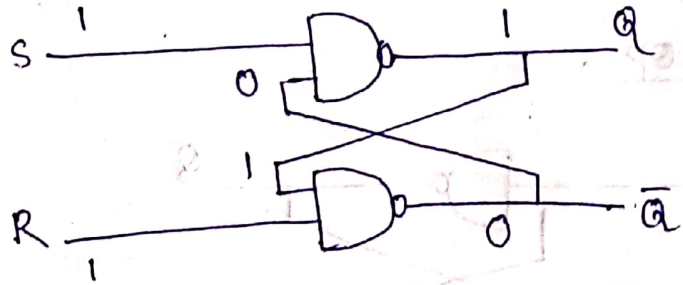
S = 1, R = 0



As Q = 0, so output is 0.

S = 1, R = 1

suppose
$\bar{Q} = 0$
$Q = 1$



As current output = previous output

So, output is Hold.

→ In NAND S-R latch if we input S=0 and R=0 the output will be invalid. Because if one of the inputs to NAND gate is 0 then the output will be
→ Hence in S=0 case the result Q=1 and in R=0 case the result $\bar{Q}=1$. But $Q=\bar{Q}=1$ is impossible.

→ If we input S=0 and R=1 then the output will be 1.
As if we input S=0 the value of Q=1.
So if we input Q=1 and R=1 the output will be which is valid as Q=1 and $\bar{Q}=0$

→ If we input S=1 and R=0 then the output is 0.
As if we input R=0 then $\bar{Q}=1$
If we input S=1 and $\bar{Q}=1$ then the output is 0, which is valid as Q=0 and $\bar{Q}=1$.

→ If we input S=1 and R=1 then the output is considered as hold condition or generates previous output as -

for S=1 $\overline{1 \cdot \bar{Q}}$

$= \bar{1} + \bar{\bar{Q}}$

$= 0 + Q = Q$

= previous output Q

for R=1 $\overline{Q \cdot 1}$

$= \bar{Q} + \bar{1}$

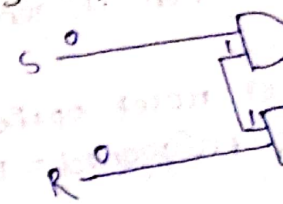
$= \bar{Q} + 0$

= $\bar{Q}$ (previous output $\bar{Q}$)

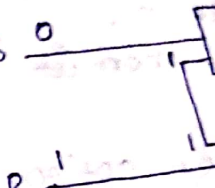
Truth Table:

clock	S	R	Q
Not triggered	X	X	Q
Triggered	0	0	Hold
Triggered	0	1	1
Triggered	1	0	0
Triggered	1	1	

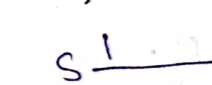
If S=0, R=0

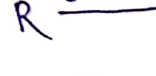


If S=0, R=1

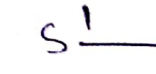


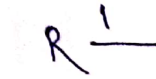
If S=1, R=0



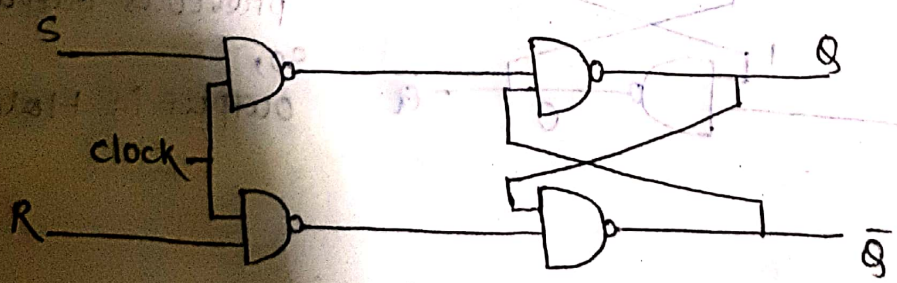


If S=1, R=1





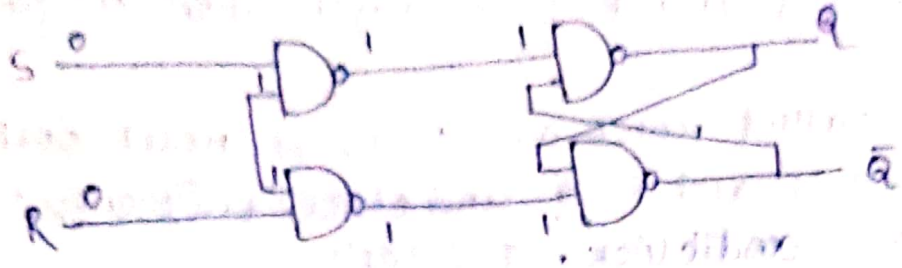
NAND SR Flip-Flop :-



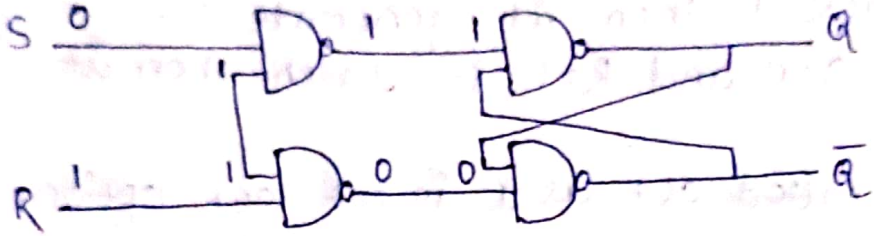
Truth Table :

Clock	S	R	Q(n+1)
Non triggered	x	x	Qn
Triggered	0	0	Hold (previous output)
Triggered	0	1	0 (Reset)
Triggered	1	0	1 (set)
Triggered	1	1	Invalid

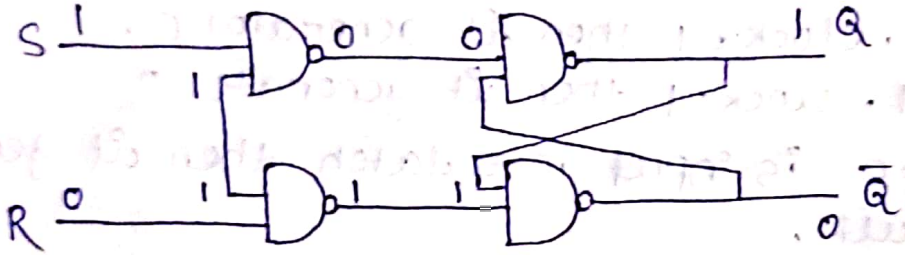
If S=0, R=0



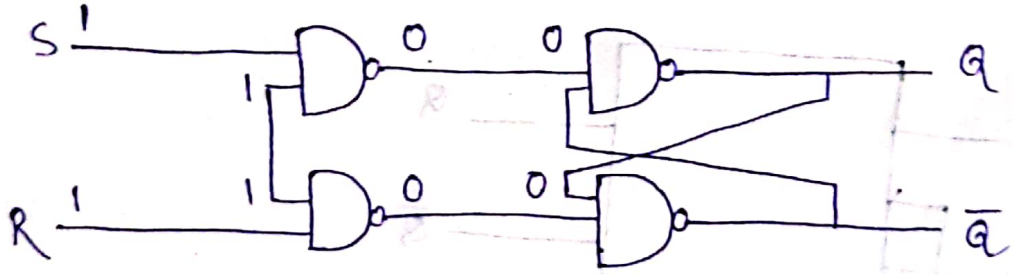
If S=0, R=1



If S=1, R=0



If S=1, R=1



→ In NAND SR Flip-Flop if we input clock triggered and S=0, R=0 then the output will be hold.
As if inputs S=1 and R=1 in latch and the output of which is hold.

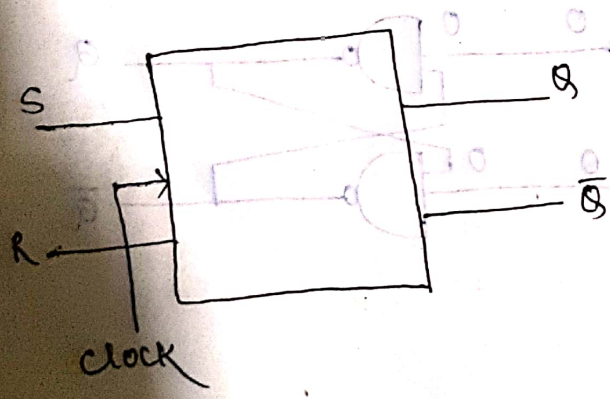
→ If we input S=0 and R=1 and clock=triggered=1 then the output will be 0 (Reset).
As if S=0, clock=1 then it generate 1.
If R=1, clock=1 then it generate 0.
If we input S=1 and R=0 in latch then it generates 0.
The state is called reset as 1 is at reset option.

→ If we input S=1 and R=0 and clock=triggered=1 then the output will be 1 (Set).
As if R=0, clock=1 then it generate 1.
If S=1, clock=1 then it generate 0.
If we input S=0 and R=1 in latch then it generates 1.
The state is called set as 1 is at set option.

→ If we input S=1 & R=1 and clock=triggered=1 then the output will be invalid.
As if S=1, clock=1 then it generates 0.
if R=1, clock=1 then it generates 0

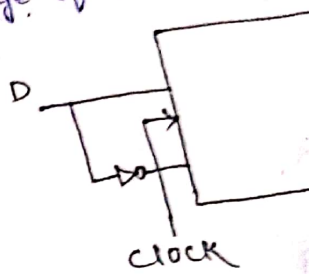
If S=0, R=0 is input into latch then it generate invalid result.

Block diagram of SR Flip-Flop :-

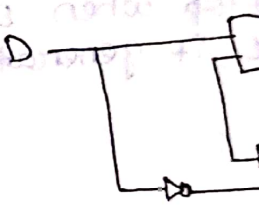


D Latch :-

→ D Latches are called
→ These are implemen
i) D: Data or
ii) Clock pulse
→ The output of the terminal as long
→ When the clock sig latch is stored edge of the clock



circuit diagram

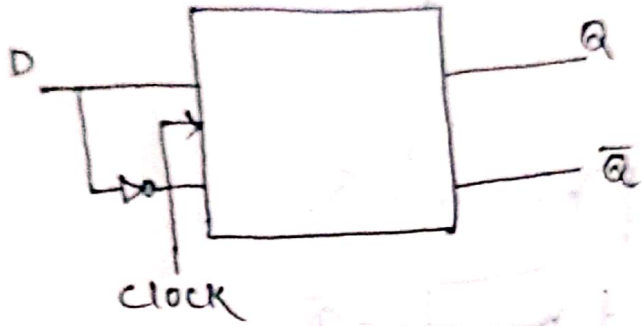


Truth Table

Clock	D
Not triggered	0
Triggered	1
Triggered	1

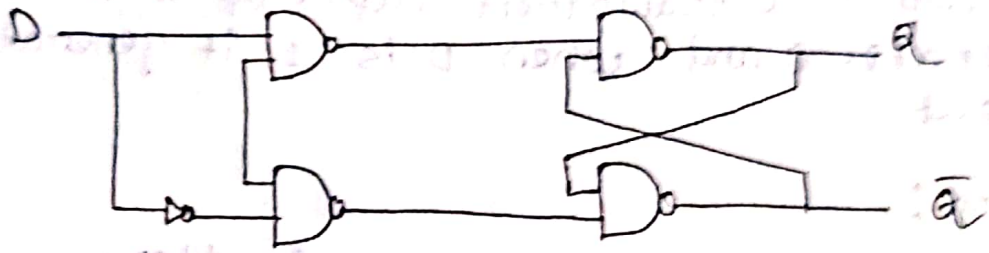
D Latch:

→ D latches are called as transparent latch.
→ These are implemented using 2 inputs.
  i) D: Data or Delay
  ii) clock pulse
→ The output of the latch follows the input at D terminal as long as the clock signal is high.
→ When the clock signal goes low the output of the latch is stored and held until the next rising edge of the clock.



Exp 10

Circuit diagram of NAND D-Flip-Flop:

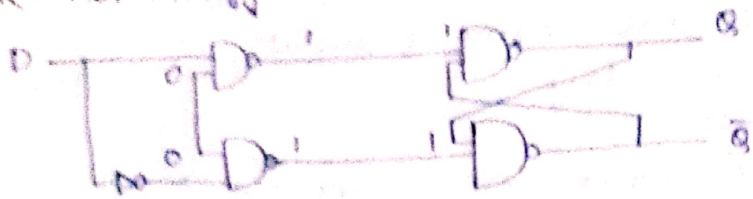


Truth Table

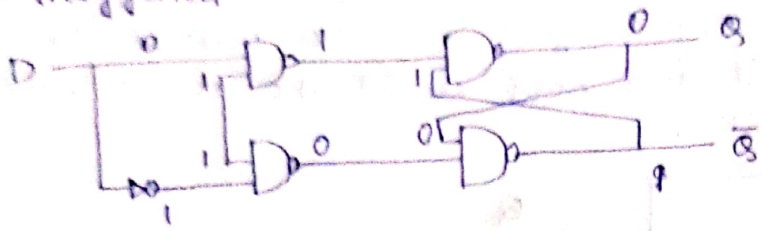
clock	D	Qn	Q(n+1)
Not triggered 0	X	Qn (Hold)	Prev. output
Triggered 1	0		0
Triggered 1	1		1

check:

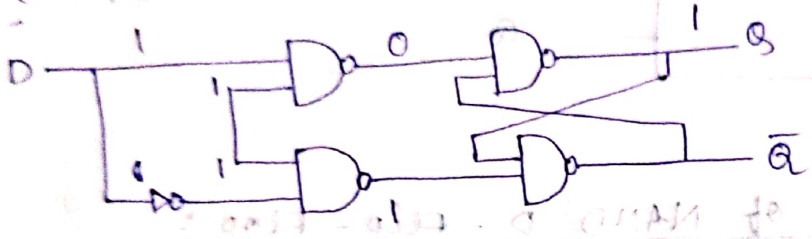
clock not triggered.



clock triggered. , D=0



clock = 1 , D = 1



→ As D Flip-Flop is a transparent flip-flop when D is 0 it generates 0 and when D is 1 it generates 1 as output.

check

1. Q = 0
$\bar{Q}$ = 1
output = 1
Q = 1

2. Q
$\bar{Q}$

JK Flip-Flop :

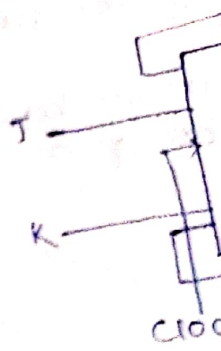
→ JK Flip-Flop is most widely used flip-flop.

→ It solves the drawback of SR flip-flop of giving invalid result for (1,1) using the advanced feature of SR flip-Flop.

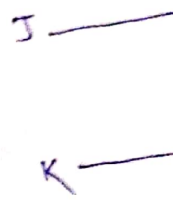
→ It generates the outputs Q and $\bar{Q}$.

→ It uses the SR flip-flop as its technology and uses one more feature of using the outputs Q & $\bar{Q}$ as the feedback.

Block diagram



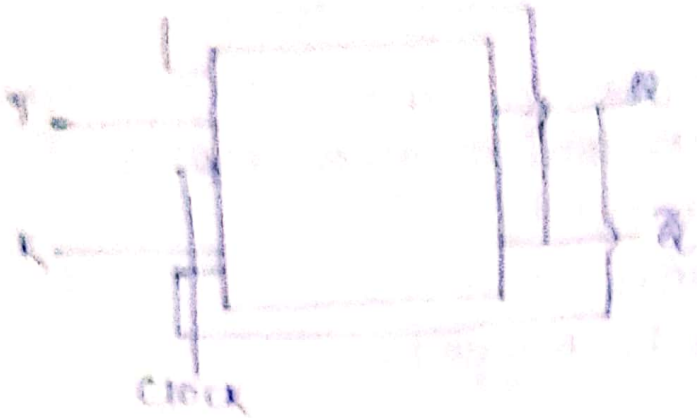
circuit diagr



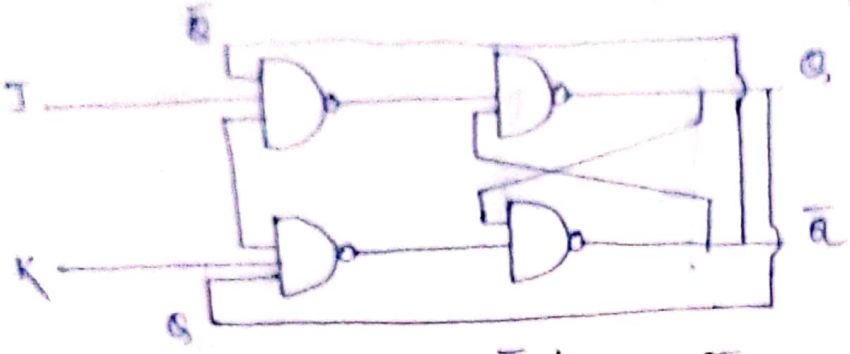
Truth T

clock
Not triggered

Block diagram of NAND JK Flip Flop:



Circuit diagram of JK flip flop:



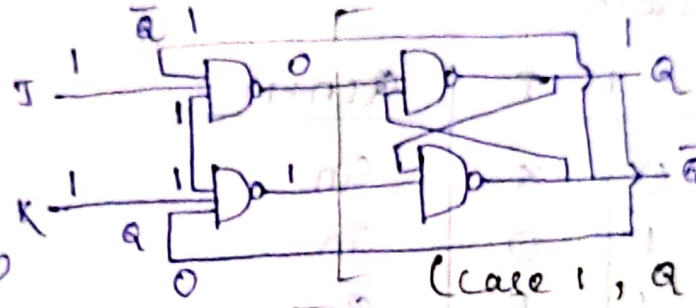
Check

1. $Q = 0$. 2. $Q = 1$

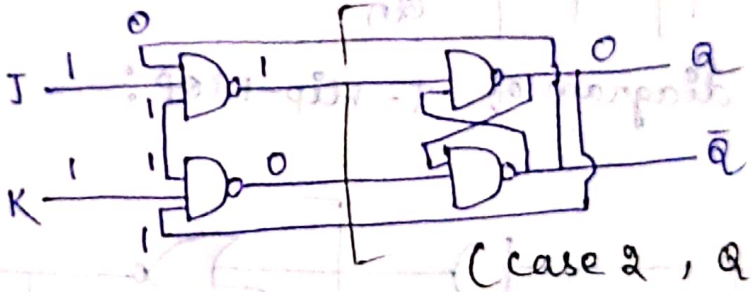
$\bar{Q} = 1$ $\bar{Q} = 0$

Output = 1 Output = 0

$Q = 1$ $Q = 0$



(case 1, Q = 0)



(case 2, Q = 1)

Truth Table

Clock		J	K	Q(n+1)
Not triggered	0	X	X	Qn
	1	0	0	Hold
	1	0	1	0
	1	1	0	1
	1	1	1	Toggle

→ In case-1 when we input $Q=0$ and $\bar{Q}=1$ we h... to input (0,1) in latch which generates resu... $Q=1$.

→ In case-2 when we input $Q=1$ and $\bar{Q}=0$ we have input (1,0) in latch which generates $Q=0$.

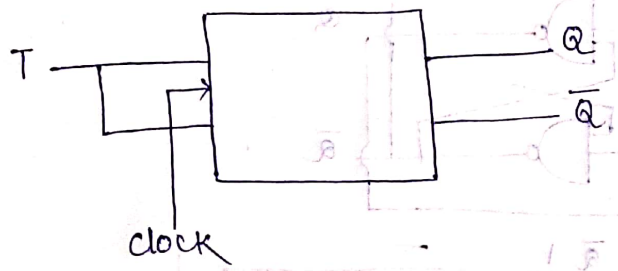
As $Q=0$ provides $Q_{n+1}=1$

$Q=1$ provides $Q_{n+1}=0$

It means the result is toggled.

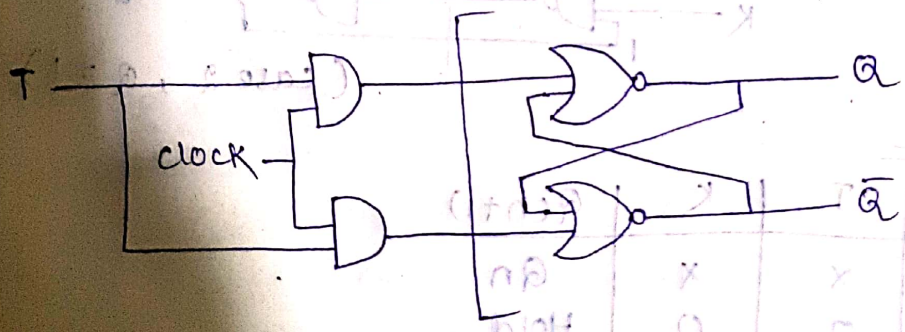
T Flip-Flop :-

The block diagram of T-Flip-Flop -



clock	T	Q(n+1)
Not triggered 0	X	Qn
Triggered 1	0	Qn
Triggered 1	1	$\bar{Q}n$

Circuit diagram of T-Flip-Flop :-



→ It uses JK Flip
→ When JK in JK
→ When the clock be generated.
→ When the clock the results.

Truth Table :-

clock	T
Not triggered 0	X
	0
	0
	1
	1

→ It generates
→ It generates 1 in case

Excitation Ta

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

The equati

Q_{n+1} =

Applicati

1. Counter
2. Freque
3. Shift
4. Storag
5. Bounce
6. Data
7. Data
8. Late
9. Prim

→ It uses JK flip-flop with XOR latch.
→ When JK in JK flip-flop then it becomes T flip-flop.
→ When the clock is not triggered previous output will be generated.
→ When the clock is triggered to 1 then it will toggle the results.

Truth Table :-

clock	T	Qn	Qn+1
Not triggered 0	x	Qn	previous output
	0	0	0
	0	1	1
	1	0	1
	1	1	0

→ It generates XOR results.
→ It generates 0 in case of similar inputs & generates 1 in case of dissimilar inputs.

Excitation Table :-

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

The equation of T-flip-flop

$Q_{n+1} = T \oplus Q_n$

Application of Flip-Flop :-

1. Counters
2. Frequency dividers
3. Shift registers
4. Storage registers
5. Bounce elimination switch
6. Data storage
7. Data transfere
8. Latch
9. Primary Memory

→ Master Slave Flip-Flop :

Race around condition in JK Flip Flop :

→ → In JK flip flop if J=K=1 and clock=1 then the output Q is toggled, we have experienced.

→ If such condition arises for longer period of time as long as the clock is high, or 1, then this is the output of the flip flop unstable are uncertain.

→ This problem is called as race around condition in JK Flip Flop.

→ This problem can be avoided by ensuring that the clock input can be 1 only for a short period of time.

→ This leads to use of master slave flip flop.

Architecture of Master Slave Flip-Flop :

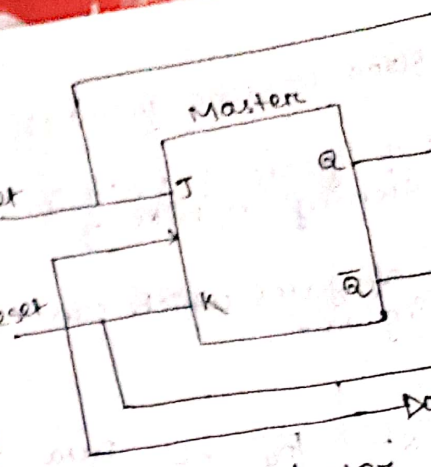
→ The Master Slave JK flipflop is basically a combination of 2 JK flip flops connected together in a series configuration.

→ One flip-flop work as Master and other as slave.

→ The output from the master flip flop is connected to the 2 inputs of the slave flip-flop where the output is feedback to the inputs of the master flip-flop.

→ In addition to these two flip flops the circuit also includes an inverter. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip flop.

In other words if CP (clock pulse) = 0 for a master flip-flop then CP=1 for a slave flip-flop and if CP=1 for a master flip-flop then it is set to 0 for slave flip-flop.



Working of Master

i) When clock pulse the JK inputs ma system. It is isc information is p the slave and ou

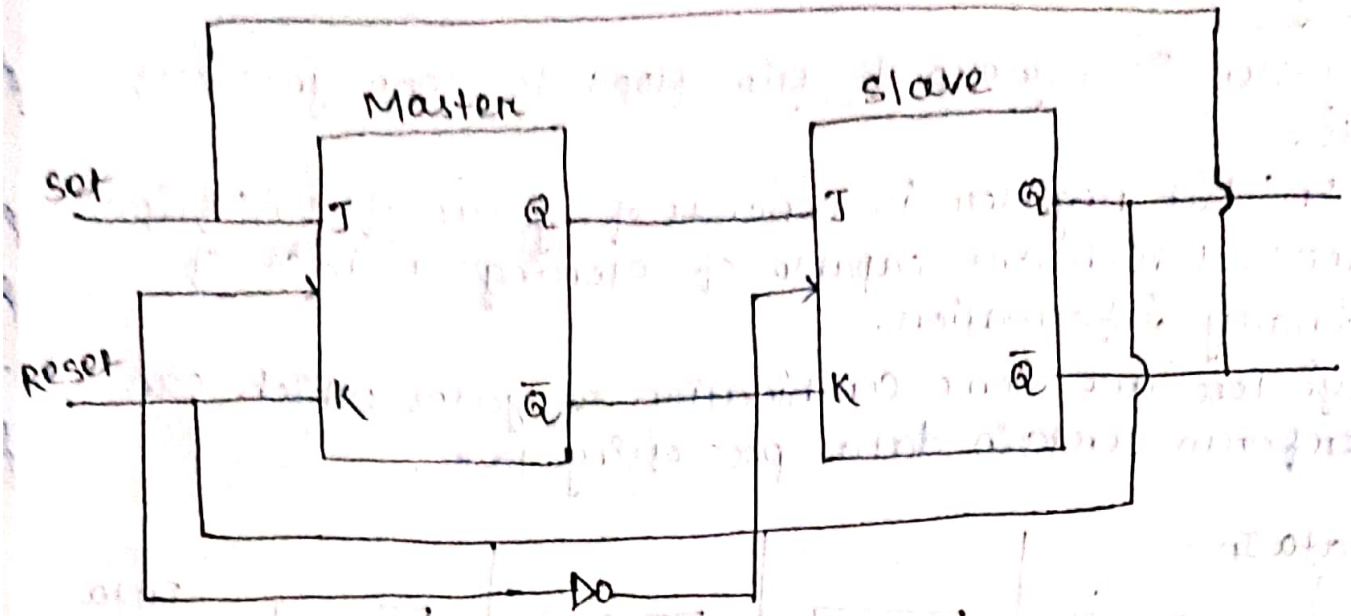
ii) Firstly the mast and the slave fl so the master re

iii) If J=0 and K= goes to the K i the slave to re

iv) If J=1 and K goes to the J master.

v) If J=1 and K= of the clock negative tr

vi) If I=0 a remains



Working of Master Slave :-

i) When clock pulse go to 1 the slave is isolated, the JK inputs may affect the state of the system. It is isolated until CP goes to 0, information is passed from the master, flip flop to the slave and output is obtained.

ii) Firstly the master flip-flop is positive level triggered and the slave flip flop is negative level triggered so the master responds before the slave.

iii) If J=0 and K=1, the high $\bar{Q}$ output of master goes to the K input of the slave and the clock forces the slave to reset. Hence, the slave copies the Master.

iv) If J=1 and K=0 the high Q output of the master goes to the J input of the slave and the negative transition of the clock sets the slave, copying the master.

v) If J=1 and K=1 it toggles on the positive transition of the clock and hence the slave toggles on the negative transition of the clock.

vi) If I=0 and K=0 the flip flop is disables and Q remains unchanged.

Register :

→ Register is a group of flip flops to store group of bits.

→ A 'n' bit register is consists of group of 'n' flip flops which are capable of storing n bits of binary information.

→ Register are some combination of gates which can perform certain data processing task.

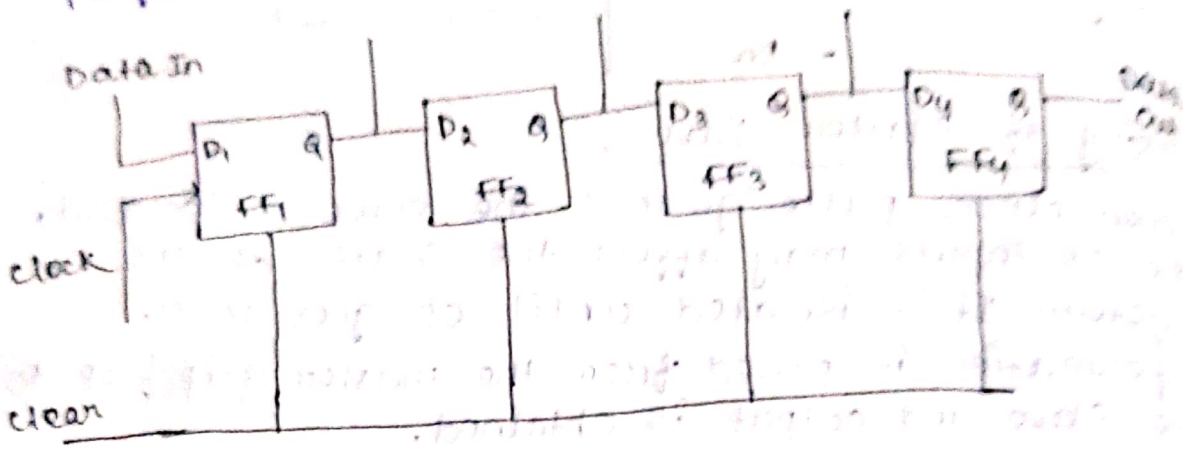


Diagram of Register

→ In the above diagram a register is connected with type construction.

→ A common clock input triggers to all flip-flops on the positive edge of the clock pulse and the binary data is available in 4 inputs - D_1, D_2, D_3, D_4

→ The clear signal connected to all the flip-flops. When the signal value goes to 0 then all the flip-flops reset to 0 synchronously.

→ The clear input is useful for buffer clearing the register.

→ The load signals and resends are used for helping the flip-flop to load the binary information.

Shift register

→ It a register

→ It permits

→ In a shift

→ Two methods

i) Serial

ii) Paralle

i) In Serial

ii) In paralle

Shift

basing

on shift

i) Serial

ii) Serial

iii) Parall

iv) Parall

→ The bits

→ In bidi

→ This is

Shift Register :

- If a register is capable of shifting binary information either to the left or right is called as shift register.
- It permits the stored data to move from a particular location to some other location within the digital system.
- In a shift register the flips are connected in such a way that the bits of the binary number entered into the shift register.
- Two methods of shifting binary register are :
  - i) Serial shifting
  - ii) Parallel shifting

i) In Serial shifting method it shifts 1 bit at a serial manner begning with MSB or LSB.

ii) In parallel shifting operation all data gets shifted simultaneously during a single clock pulse.

Shift registers are classified into 4 types basing upon how binary information is entered, or shifted.

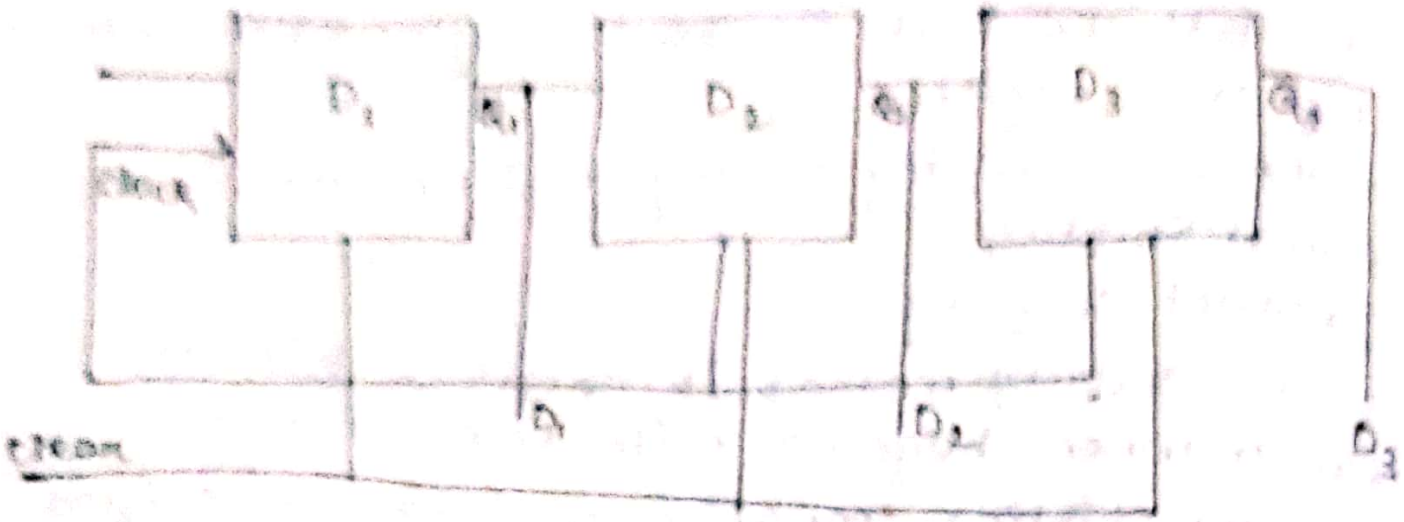
- i) Serial In Parallel OUT. (SIPO)
- ii) Serial In serial Out. (SISO)
- iii) Parallel In Parallel Out. (PIPO)
- iv) Parallel In Serial Out. (PISO)

- The bits are shifted on flip-flops with the occurrence of clock-pulse either in the right direction or left direction which is known as right shift or left shift respectively.
- In bidirectional shift register the data can be shifted from left to right as well as in reverse direction using the mode control.
- This is also known as universal shift register.

Types of Shift Registers:

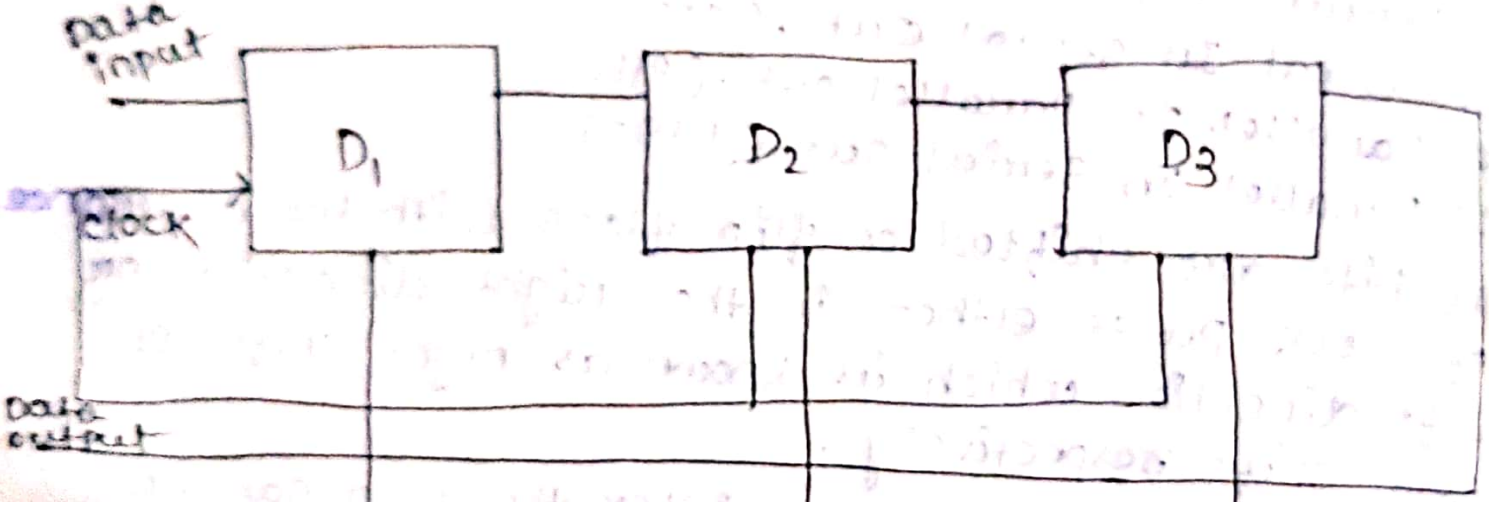
i) Serial In Parallel Out (SIPO):

In this type of register the data are serially (bit by bit or byte) inputted through the input line & produces the stored data through parallel lines.



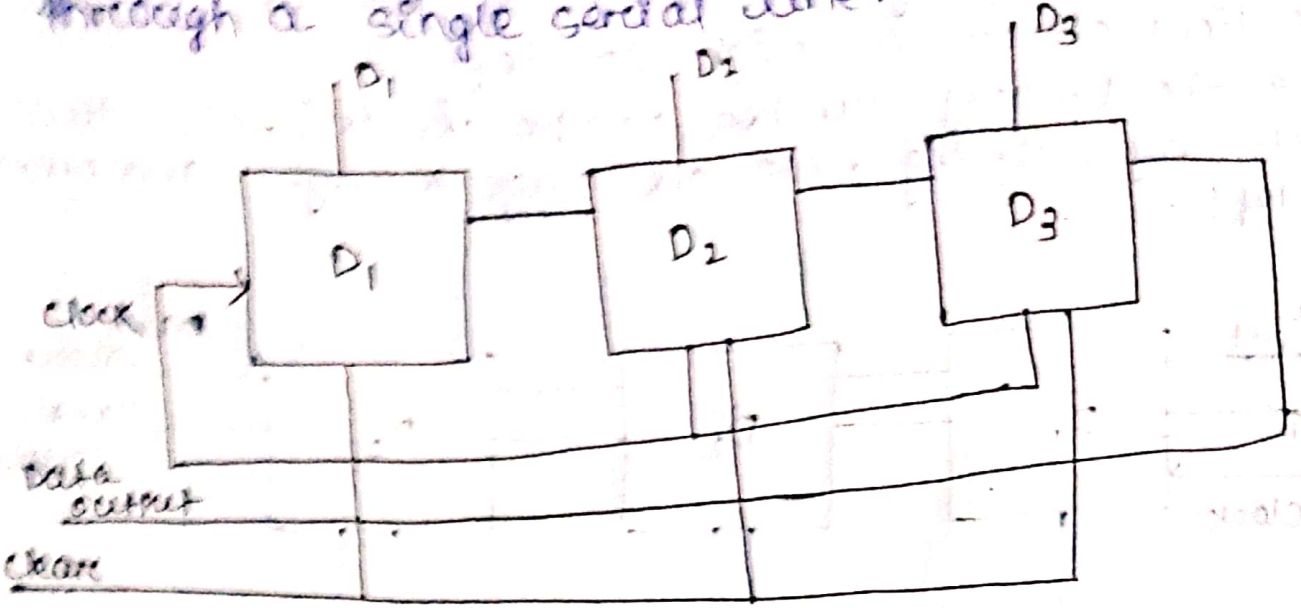
ii) Serial In Serial Out (SISO):

In SISO shift register the data are input through a single line and outputs can be collected through another single line.



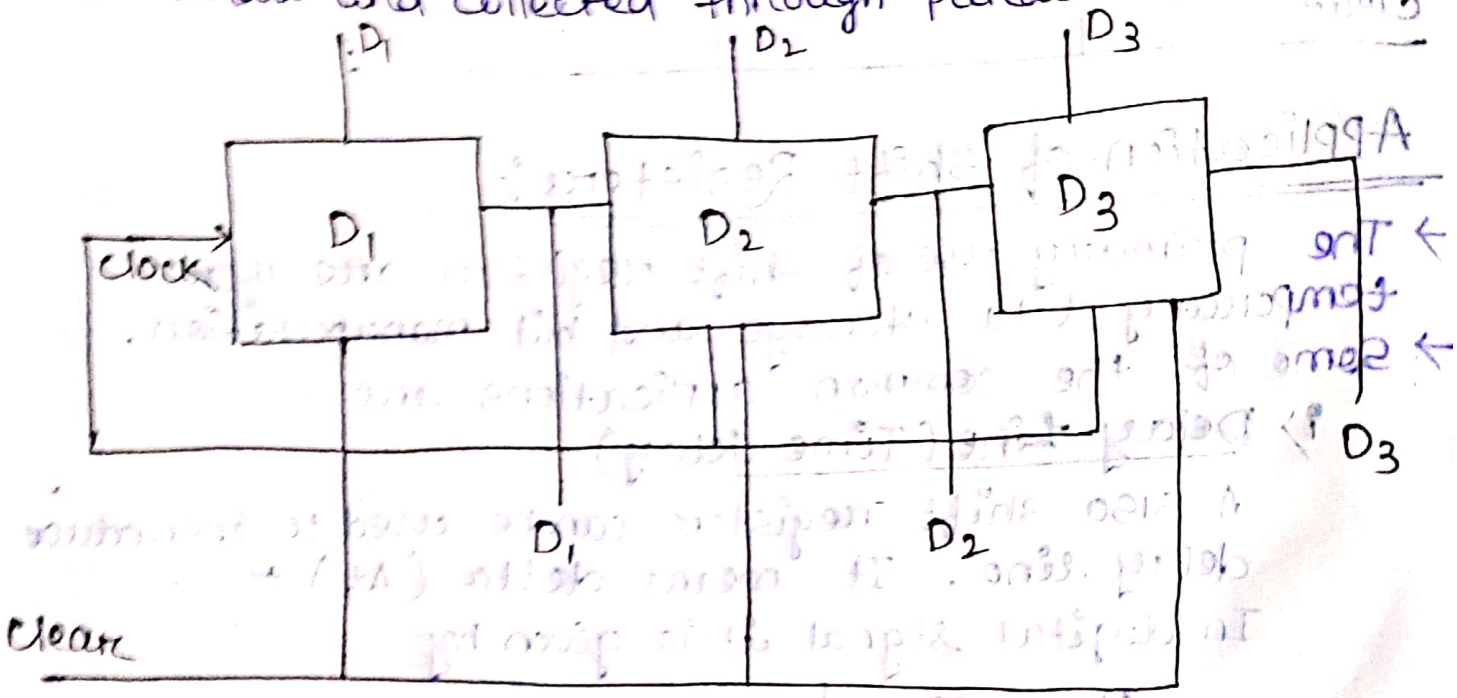
iii) Parallel In Serial Out (PISO) :-

In this shift register the data entered through parallel lines and stored data are collected through a single serial line.



iv) Parallel In parallel Out (PIPO) :-

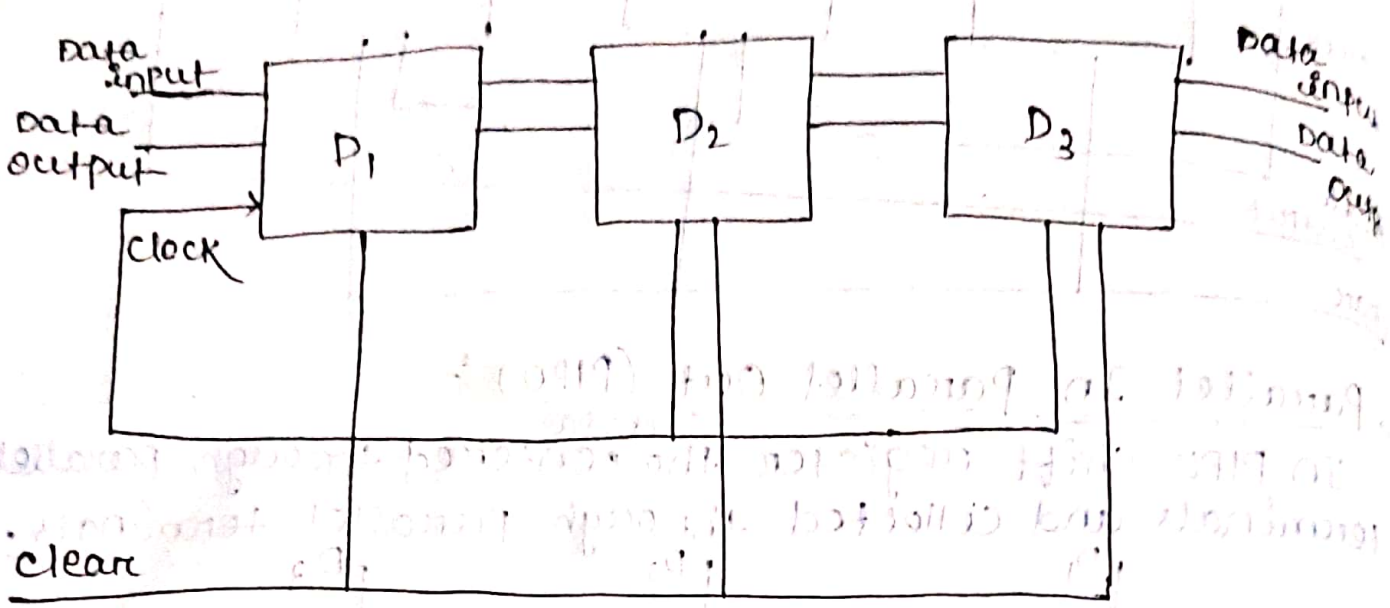
In PIPO shift register the data entered through parallel terminals and collected through parallel terminals.



Bidirectional shift registers :

→ A bidirectional shift register is something on which the shifting of data can be possible in both directions either in right or left.

→ Here the binary number can be devided into two parts by shifting into one stage to right and other to left.



Application of Shift Registers :-

→ The primary use of shift registers are for temporary data storage and bit manipulation.

→ Some of the common applications are -

1) **Delay Line (Time delay) -**

A SISO shift register can be used to introduce delay line. It means delta (Δt) t.

In digital signal it is given by

$$\Delta t = N \times \frac{1}{fc}$$

N = No. of signals

fc = clock frequency

ii) Serial to Parallel converter :

The data in serial form can be converted to parallel form by using serial in parallel out shift register.

iii) Parallel to serial converter :

The data in parallel form can be converted into serial form by using parallel in serial out shift register.

iv) Ring Counter :

If the serial output Qn of a shift register is connected back to the serial input. Then the injected pulse will keep circulating. This circuit is called ring counter.

v) Twisted Ring Counter :

In a shift register if the new complement is connected back to the serial input. Then the resulting circuit is known as twisted ring counter.

vi) Sequence counter :

If a circuit generates a prescribed sequence of bits in synchronous clock, then it is called sequence generater / counter.

Counter :

→ It is a sequencial circuit consisting of a set of flip-flops connected in a suitable manner to count a sequence of input pulse.

→ It is classified into 4 categories.

i) Asynchronous or Ripple counter
ii) Synchronous counter
iii) Modular counter
iv) Single or multimode counter

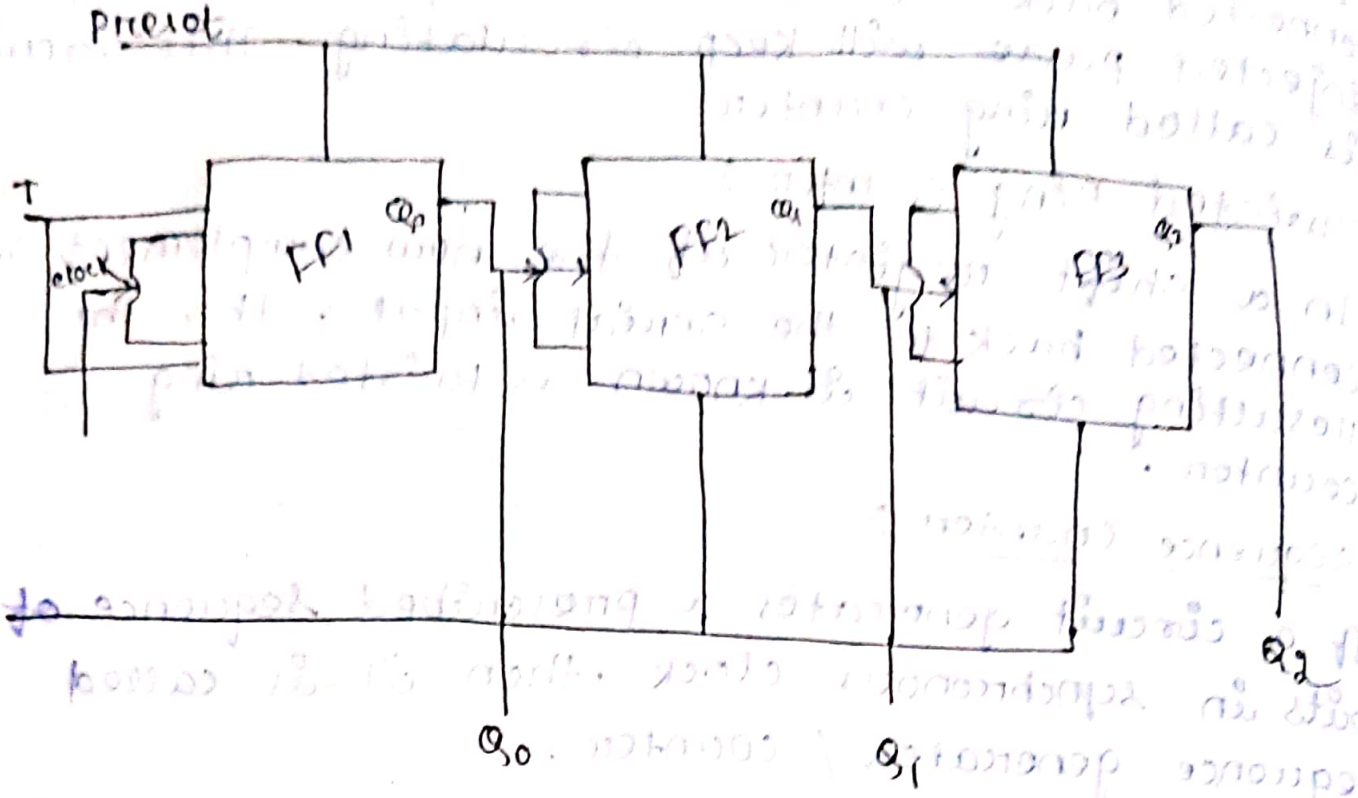
9) Asynchronous or Ripple Counter :

→ It can be constructed by using minimum number of hardware so that each flip-flop is triggered by the output from the previous flip-flop.

→ It consists of a series of connections of complementing flip-flops the output of each flip flop connected to the clock input to the next higher flip-flop.

→ The flip-flop holding the least significant bit (LSB) receives the incoming count pulse.

→ A complementary flip-flop can be obtained from a T flip-flop.



→ Asynchronous counters can be designed either using T or JK flip-flops.

→ As the asynchronous counter doesnot depends on the same clock signal, the output of one flip-flop will be sent as input clock to the next flip-flop.

→ It provides separate outputs from each flip-flop.

→ As the output of one flip-flop is used as the input of the next, hence it triggers

next flip-flop.

→ The asynchronous counter can be called as the ripple counter.

ii) Synchronous Counter :

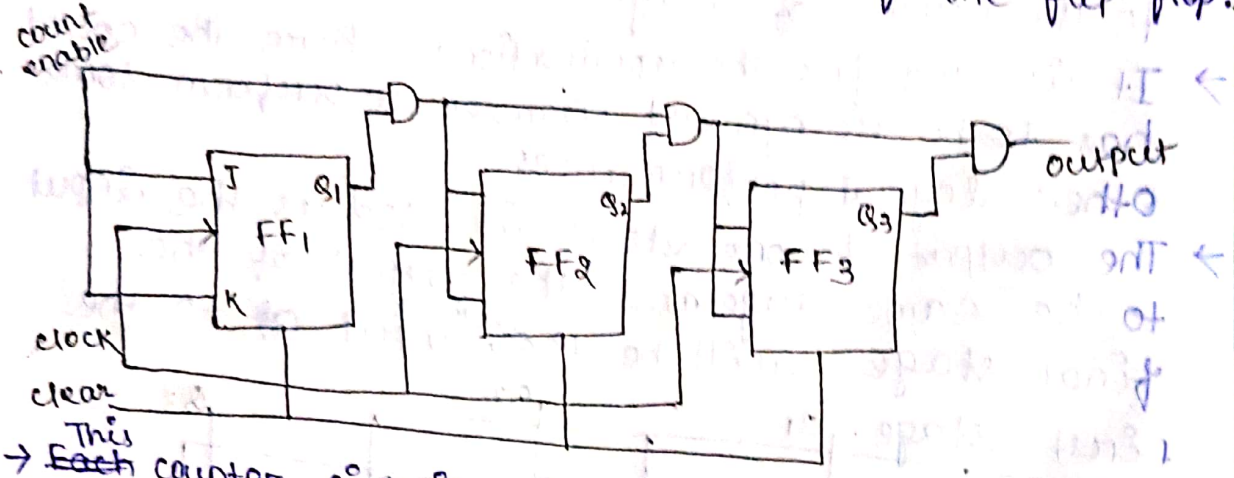
→ In this case the counter refers to something where one flip-flop will co-ordinate with the other flip-flop based on time.

→ Here the same clock-pulse provided as difference to all the flip-flops.

→ The counter that uses the clock signal for transection from one state to another is known as synchronous counters.

→ Here all the flip-flops are triggered by the same clock signal.

→ Here all the flip-flops are connected and the output state of the previous flip-flop determines the change in the current state of the flip-flop.



→ This counter circuit will usually employ flip-flops with complementing capability.

→ Both T and JK flip-flop designs have the same property.

→ In case of JK, if J=K=1 and the clock goes through a possible transection then the complemented value will be the outcome.

→ The output of the flip-flop doesn't change if J=K=0.

→ The counter may be controlled with an enable input that turns the counter on or off without removing the clock signal from the flip-flop.

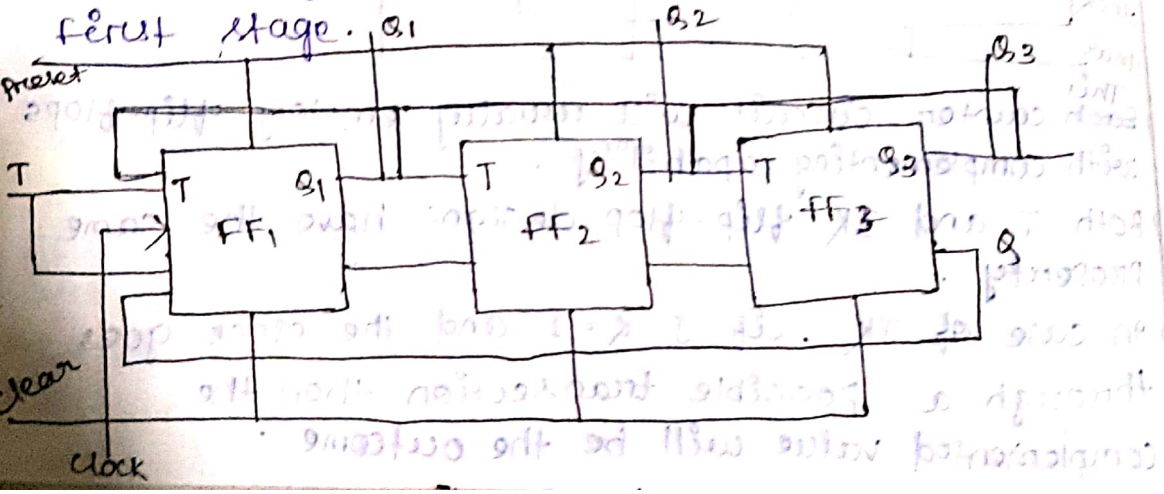
→ The chain of end gates generates the required logic i.e. the J and K input and output carry is extended to the next stage or next flip-flop.

iii) Modular counter or ring counter :-

→ It is the simplest example of shift registers used for ring counter which only contains a single logical value 0 or 1. This value circulates the total cycle length as usual to the number of stages.

→ It is used for the applications where the count has to be recognised inorder to perform some other logical performances.

→ The output of one stage is used as the input to the same stage and the output of the final stage will be the feedback input to the first stage.

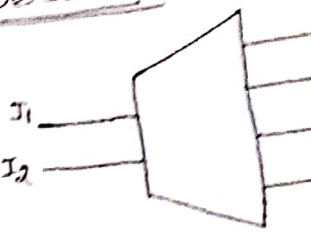


Decoder :-

→ The decoder decodes the on active output signal.

→ A decoder is a circuit bit binary code into each output line will the possible combinations

2 to 4 Decoder :-

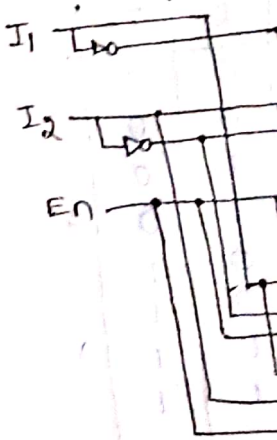


In a 2 to 4 decoder signals and 4 control

Truth Table :-

I_1	I_2	D_0	D_1	D_2
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	0	0

Circuit Diagram :-

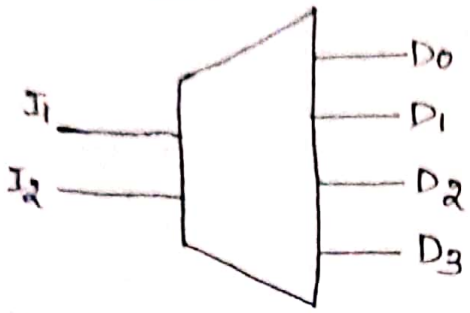


Decoder :

→ The decoder decodes the coded input signals into an active output signal.

→ A Decoder is a logical circuit which convert 'n' bit binary code into 2^n output lines such that each output line will be activated for one of the possible combination of input values.

2 to 4 Decoder :



(Block diagram of 2 to 4 Decoder)

In a 2 to 4 decoder there will be 2 control input signals and 4 control output signals.

Truth Table :

I_1	I_2	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

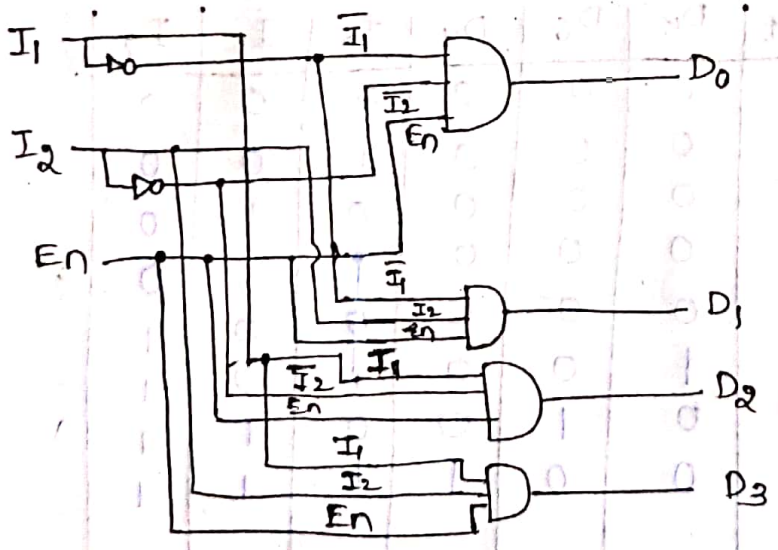
$D_0 = \overline{I_1}\,\overline{I_2}$

$D_1 = \overline{I_1} I_2$

$D_2 = I_1 \overline{I_2}$

$D_3 = I_1 I_2$

Circuit Diagram :-

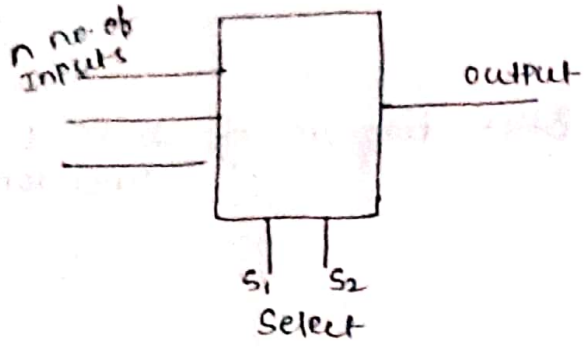


Encoder :-

→ An encoder is a digital circuit which performs the inverse operation of a Decoder, so that opposite of decoding process is called encoding.

→ An Encoder converts the active input signals into a coded output signals.

Block Diagram



→ It has n input control lines.

→ It has 2 select input signal.

→ Only one output will be generated from the n input signal but at the same moment only one input signal is activated at a time.

→ Hence n bit data will be encoded into one output.

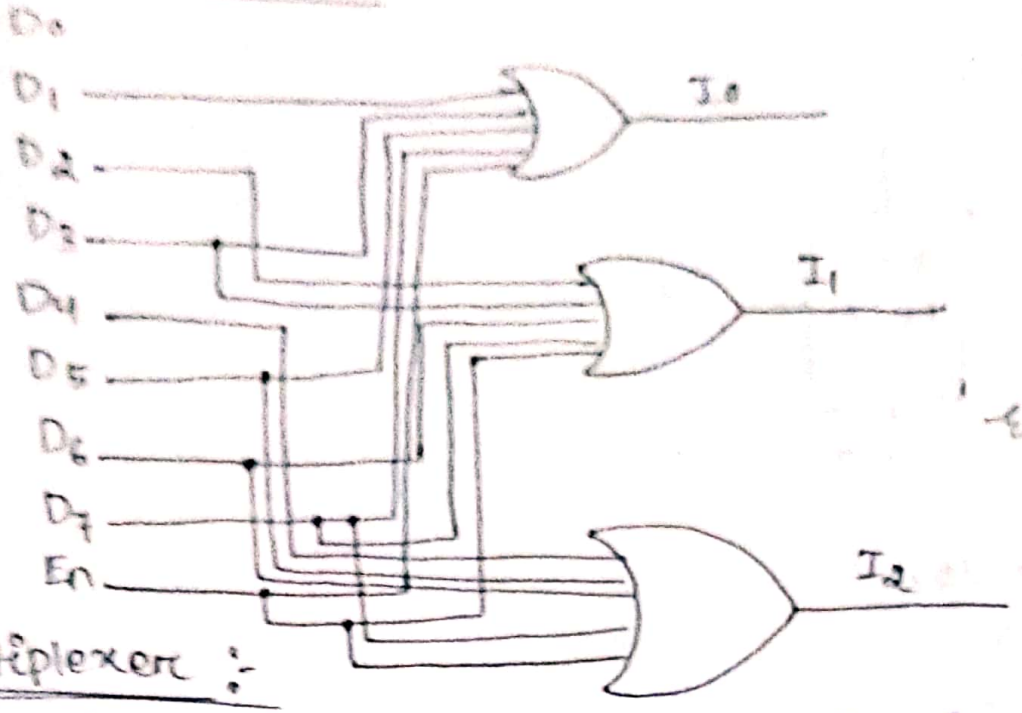
An Octal to Binary encoder :-

An octal to binary encoder accepts 8 inputs and produces 3 bit output code corresponding to the active signals.

Truth Table :-

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	I_0	I_1	I_2
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	1	1	1	1

Circuit Diagram :



Multiplexer :-

→ A Multiplexer is a combinational circuit that has 2^n input lines and a single output line.

→ It is a multi input and single output digital circuit.

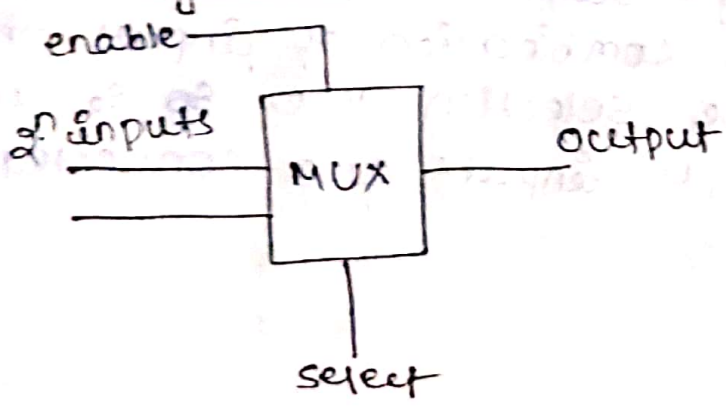
→ The binary information is received from the input lines and directed to the output lines.

→ It is also known as MUX.

→ There are several types of multiplexers like -

- 2 x 1 Mux
- 8 x 1 MUX
- 16 x 1 MUX

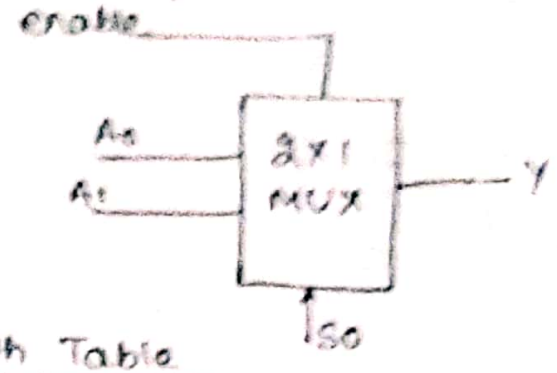
Block Diagram



e.g. 2x1 Multiplexer :

In a 2x1 MUX, there are only two inputs A_0, A_1, one selection line S_0 and a single output Y.

Block Diagram



Truth Table

Input	output
S_0	Y
0	A_0
1	A_1

Logical Expression $Y = \bar{S_0} A_0 + S_0 A_1$

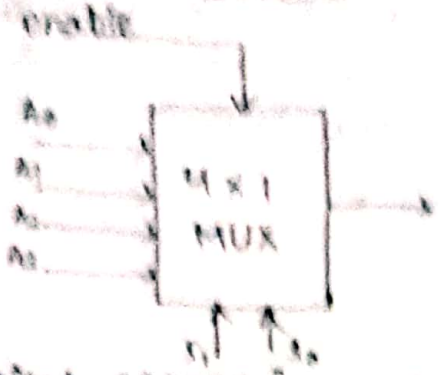
Circuit Diagram :-

4x1 Multiplexer

→ In a 4x1 multiplexer there is a total of 4 inputs A_0, A_1, A_2, A_3, two selection lines S_0 and S_1 and a single output Y.

→ On the basis of the combination of inputs that are present at the selection lines S_0 and S_1, one of these 4 inputs are connected to the output.

Block Diagram :

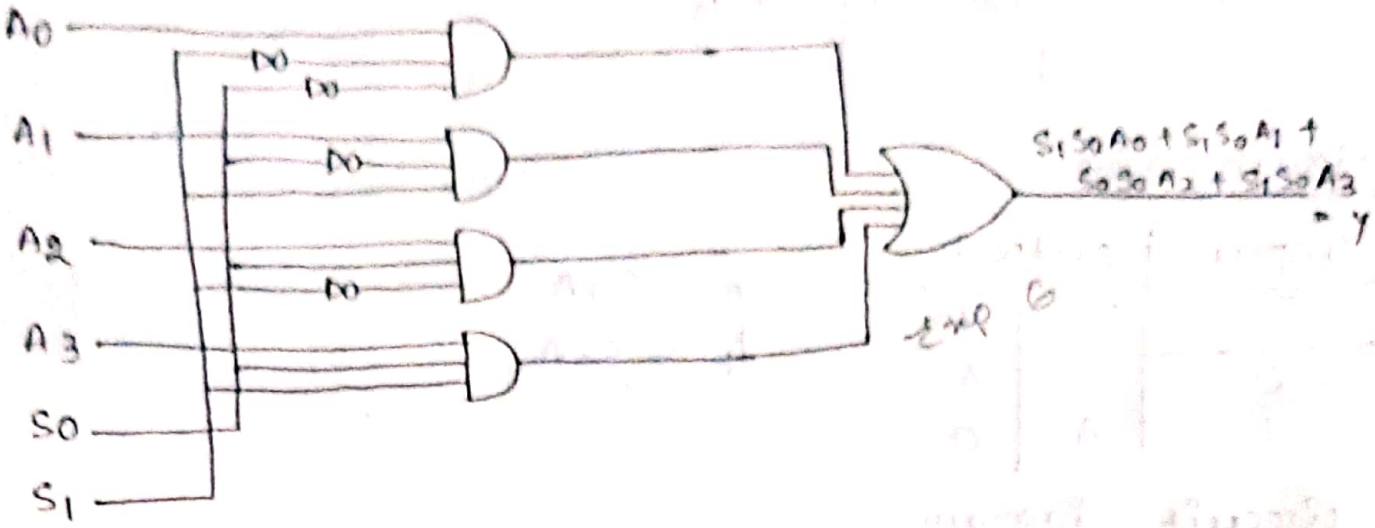


Truth Table

Inputs		output
S_0	S_1	Y
0	0	A_0
0	1	A_1
1	0	A_2
1	1	A_3

Logical expression = $S_1S_0A_0 + S_1S_0A_1 + S_1S_0A_2 + S_1S_0A_3$

Circuit Diagram :



Demultiplexer :

→ Demultiplexer also known as DeMUX. It receives one input and generates many outputs.

→ In multiplexer, several inputs are converted to single output whereas DeMUX performs its reverse action.

→ By applying the control signals the single output is scattered into multiple outputs.

→ In DeMUX there are 'n' selection lines and 2^n output lines.

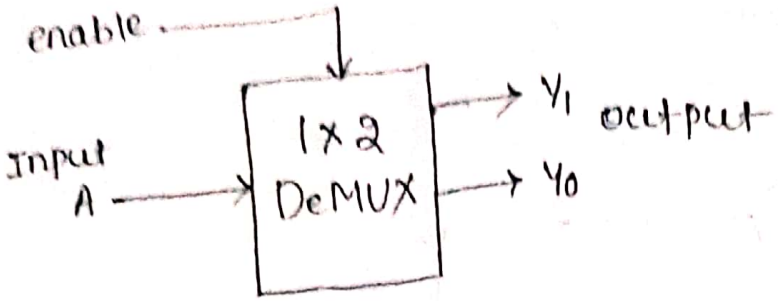
→ DeMUX are of different types:-

1X2 DeMUX
1X4 DeMUX
1X8 DeMUX
1X16 DeMUX

e.g→ 1 X 2 DeMUX

→ In 1x2 DeMUX there are two outputs Y_0 and Y_1 one selection line S_0, a single input i.e. A
→ On the basis of selection value the input will connected to one of the outputs.

Block diagram



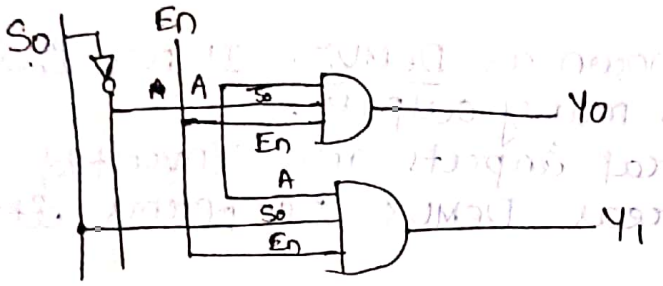
Truth Table :-

Input	output	
S_0	Y_1	Y_0
0	0	A
1	A	0

$Y_0 = \overline{S_0} A$

$Y_1 = S_0 A$

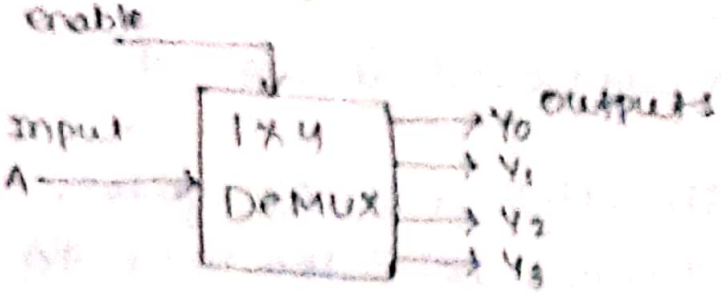
circuit diagram



1 X 4 DeMUX :-

→ In 1X4 DeMUX there are 4 outputs Y_0, Y_1, Y_2, Y_3 two selection lines S_0, S_1 and one input i.e. A.
→ On the bases of combination of input which are present at the selection lines S_0 and S_1 the input will be connected to one of the outputs.

Block Diagram :-



Truth Table :-

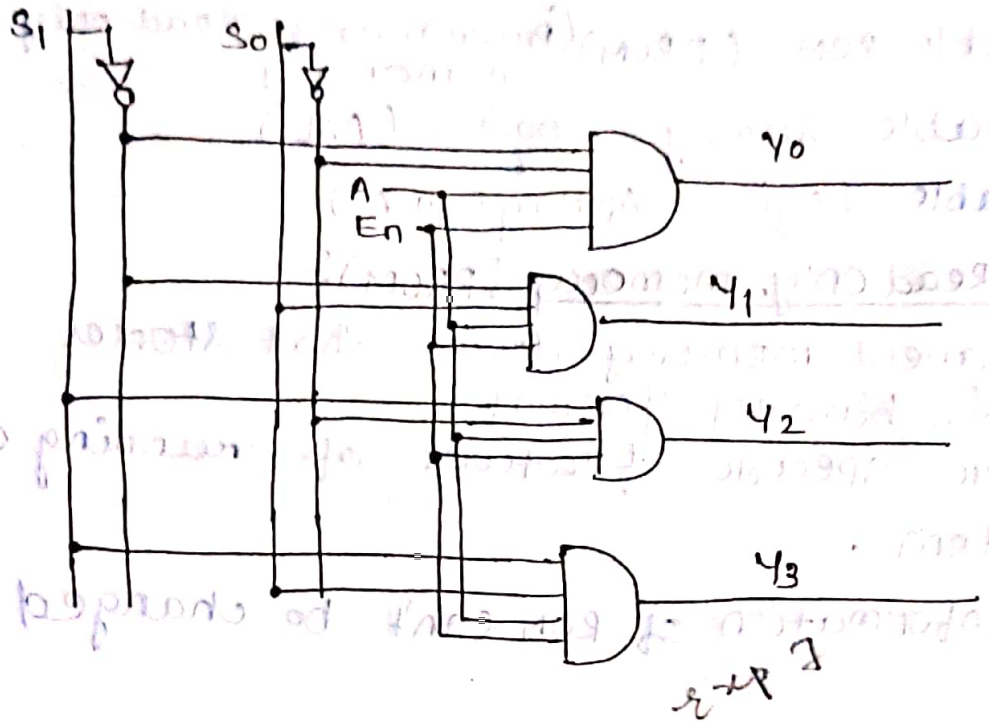
Inputs		outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	A
0	1	0	0	A	0
1	0	0	A	0	0
1	1	A	0	0	0

$Y_0 = \overline{S_1}\,\overline{S_0}\,A$

$Y_1 = \overline{S_1}\,S_0\,A$

$Y_2 = S_1\,\overline{S_0}\,A$

$Y_3 = S_1 S_0 A$



Programmable logic

→ PLDs are integrated circuits connected inside the computer systems.
→ It contains an array of AND gates and OR gates.
→ The method of entering information into the PLDs to make it workable is called as programming.
→ These integrated circuits or devices are electrically programed in order to implement boolean functions based on requirements.
→ Here programming refers to hardware programming rather software programming.
→ The three categories of PLDs with programmable features are :-
  i) Programmable ROM (PROM) [Programmable Read only memory]
  ii) Programmable Array logic (PAL)
  iii) Programmable logic Array (PLA)

i) Programable Read Only memory (PROM):-

→ ROM is a permanent memory device that stores information in binary format.
→ It is used for specific features of running a digital system.
→ The stored information of ROM can't be changed further.
→ If we need to change then it must have programmable feature. i.e. PROM
→ The PROM programmers only can perform necessary change to the binary information present on it.
→ PROM is a programmable logic device which has a fixed AND Array & programmable OR array.

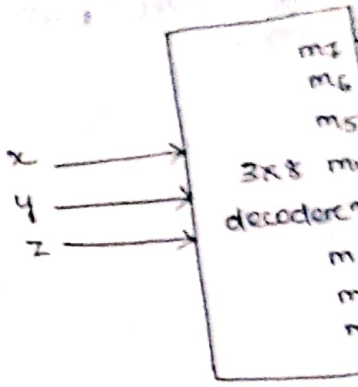
Block diagram

input

→ with n inputs we can using 2^n AND gates.
→ The product terms can be
decoder.
→ which generates n min
→ But the inputs of OR
those are received from
→ The outputs from PROM
sum of mean terms & SOP

e.g. A(x, y, z) = Σm(5, 6,
B(x, y, z) = Σm(3,

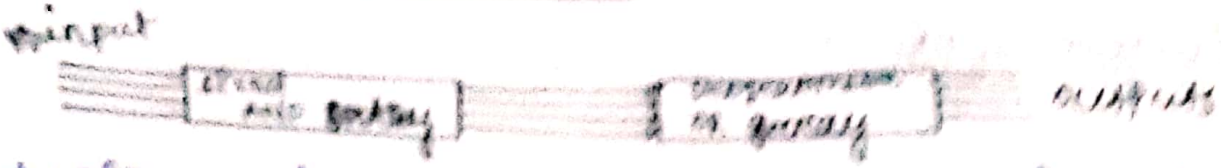
For this we can use a
it



ii) Programmable Arch

→ PAL is a Pregram
Programmable AND
→ The advantage of
only the product
instead of generat

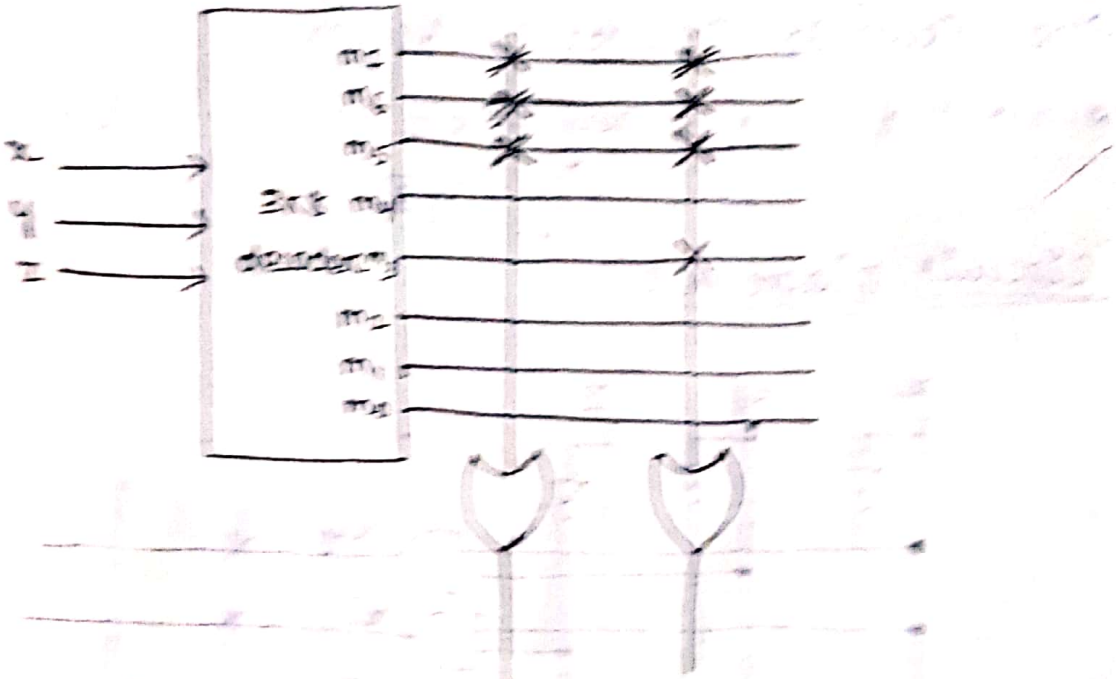
Block diagram of AND PLA :-



→ With n inputs we can generate 2^n product terms using 2^n AND gates.
→ The product terms can be implemented using n x 2^n decoder.
→ which generates n min terms.
→ But the inputs of OR gates can be programmed these are received from the AND gates.
→ The outputs from PLA will be the sum of some of min terms (SOP)

e.g. $A(x, y, z) = \Sigma m(5, 6, 7)$
$B(x, y, z) = \Sigma m(3, 5, 6, 7)$

For this one can use a 3x8 decoder for solving it.



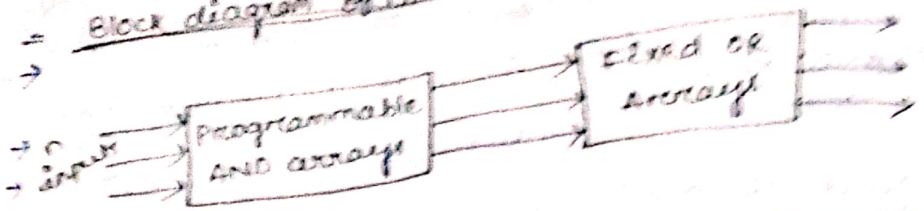
ii) Programmable Array Logic (PAL) :-

→ PAL is a programmable logic device which has programmable AND array and fixed OR array.
→ The advantage of it is that we can generate only the required number of boolean functions required by programming all the min terms using the

- Programmable AND

= Block diagram of PAL :

→



→ As the inputs of AND gates are programmable each AND gate has both normal and complemented inputs of variables.

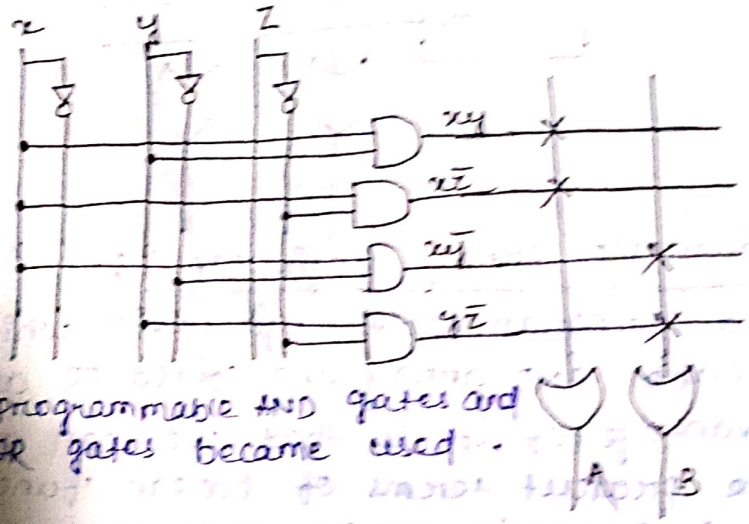
→ So based on the requirement we can program all of these inputs and generate the product terms.

→ The number of inputs to OR gates are of fixed type and outputs of PAL will be in the sum of products form.

e.g → $A = A = xy + x\bar{z}$

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

Circuit Diagram :-



Here 4 programmable AND gates and fixed OR gates became used.

- PLA is a programmable which both programmable OR array.

→ It has the same block diagram of PLA :

→ Inputs of AND gates are programmable each AND gate has both normal and complemented variables.

→ So based on the requirement we can program the product terms in AND gates.

→ As the inputs of OR gates are programmable we can program the outputs of OR gates since all outputs are available to each OR gate.

→ The outputs of PLA are of product terms.

e.g → $A = x\bar{y} + xz$

$B = x\bar{y} + yz + x\bar{z}$

Circuit Diagram :-

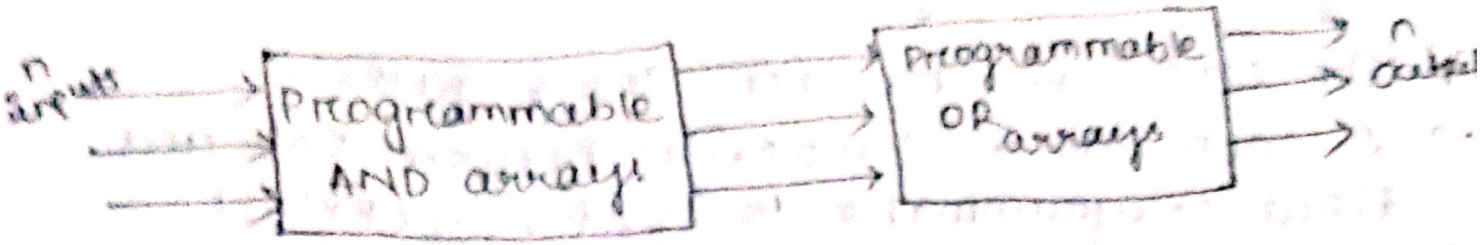
Here x programmable AND gates and programmable OR gates became used.

iii) Programmable Logic Array (PLA):

→ PLA is a programmable logic device in which both programmable AND array and programmable OR arrays are used.

→ It is the most flexible PLD.

Block diagram of PLA:



→ Inputs of AND gates are programmable hence AND gates has both normal and complemented inputs of variables.

→ So based on the requirements we can generate only the product terms required by using the AND gates.

→ As the inputs of OR gates are also programmable we can program any number of product terms. since all outputs of AND gates are applied as inputs to each OR gate.

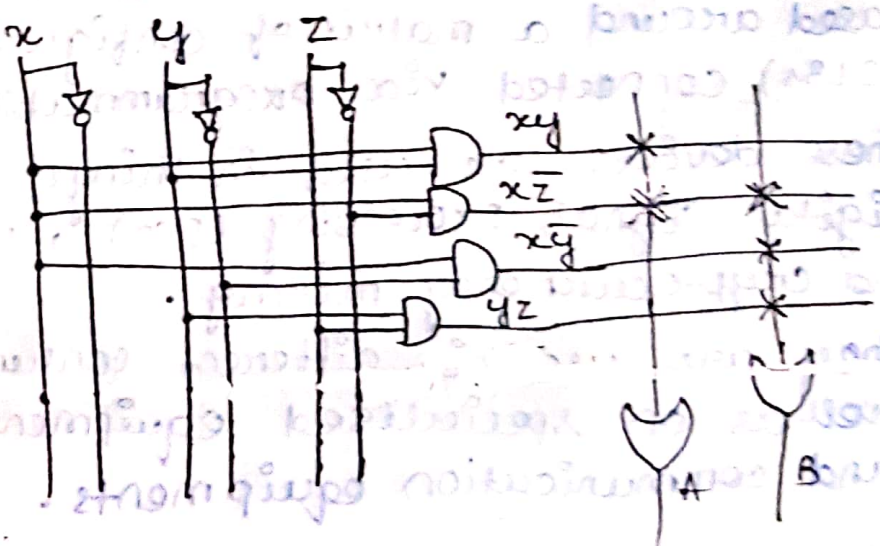
→ The outputs of PLA will be in the form of sum of product terms.

e.g → $A = xy + x\bar{z}$

$B = x\bar{y} + yz + x\bar{z}$

Circuit Diagram:

Here 4 programmable AND gates and 2 programmable OR gates became used.



Complex Programmable Logic Devices (CPLD):

→ Complex programmable logic device is an innovative product compared to earlier PLA and PALs.

→ The earlier logic device were not all programmable hence the logic was built by combining multiple logical chips together.

→ The CPLDs are based on sea of gates.

→ A CPLD has a complexity between PAL and field programmable gate Array (FPGA)

→ CPLD has features of both CPAL and FPGA.

→ The major difference betⁿ CPLD and FPGA is that, FPGA can be based on look up tables and CPLD is developed using sea of gates.

→ The common feature of both are that they are based on large number of gates and flexible provision for constructing logical devices.

Field Programmable Gate Array (FPGA):-

→ FPGA is a configurable integrated circuit which allows us to implement a wide range of ~~costum~~ custom digital circuit.

→ FPGAs are the semiconductor devices data that are based around a matrix of configurable logic blocks. (CLBs) connected via programmable interconnects.

→ These devices are used in things ~~used in~~ like digital signal processing (DSP), machine learning and cryptocurrency mining.

→ They are used in different consumer electronics as well as on specialised equipments like- satellites and communication equipments.

→ The FPGA devices are imp circuits using the verilog l language (VHDL).

Hardware Requirements for

i) Lattice ICE 40 development
ii) Bread board.
iii) Push Buttons
iv) Jumper wires
v) USB extension cable

Look up Tables:-

→ Look up table is used for di

→ It is an array that rep a simpler array indexing

→ The addressing of the com array with a value v

→ The addressing is genera K.

→ When the address is re slot K is fetched.

→ Look up tables are al ~~at~~ input values by m from the array, in it may include poin offsets or labels) to

→ FPGA also make exte hardware implemented Preprogrammable hardwo

→ The FPGA devices are implemented in digital circuits using the verilog hardware description language (VHDL).

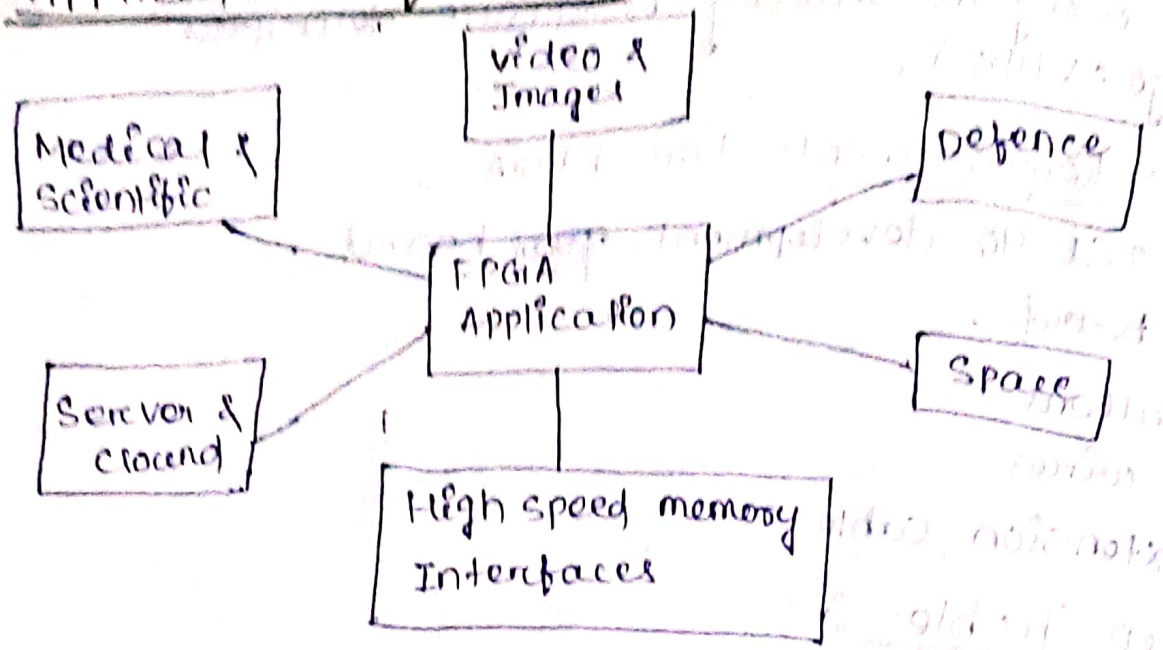
Hardware Requirements for FPGA :

i) Lattice ICE 40 development good board.
ii) Bread board.
iii) Push Buttons
iv) Jumper wires
v) USB extension cable

Look up Tables :-

→ Look up table is used for direct addressing of information.
→ It is an array that replaces run time computation with a simpler array indexing operation.
→ The addressing of the components are maintained in the array with a value v and key K.
→ The addressing is generated in the form of slots as K.
→ When the address is retrieved as the value v then the slot K is fetched.
→ Look up tables are also used extensively to validate input values by matching a list of valid items from the array, in some programming languages it may include pointer functions ~~of objects~~ (or offsets or labels) to process the matching input.
→ FPGA also make extensive use of reconfigurable hardware implemented look up tables to provide programmable hardware functionality.

Applications of FPGA :-



→ The FPGA design and programming develops integrated circuits to integrate memories, logic gates and other processing elements.

→ The devices can be popular due to its speed, flexibility and memory space saving.

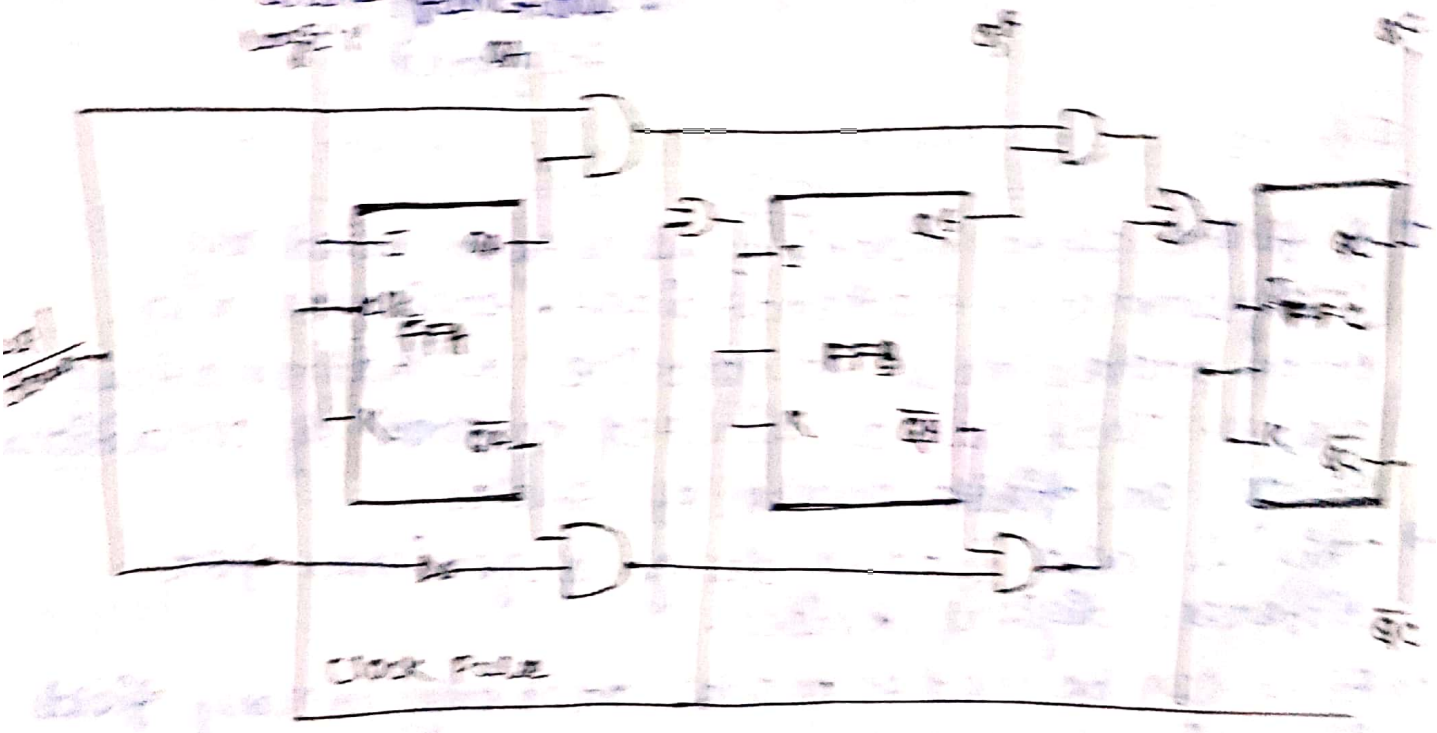
→ FPGA contains IP (Intellectual Property) to perform wide variety of functions like roll of micro-processors, filters, phase clocked ~~table~~ loops and many more other functions.

FPGA Designed Tools :-

1. Xilinx Vivado
2. Xilinx ISE
3. Intel (Altera) Quartus
4. Modelsim
5. VHDL
6. Verilog

Advantages :-

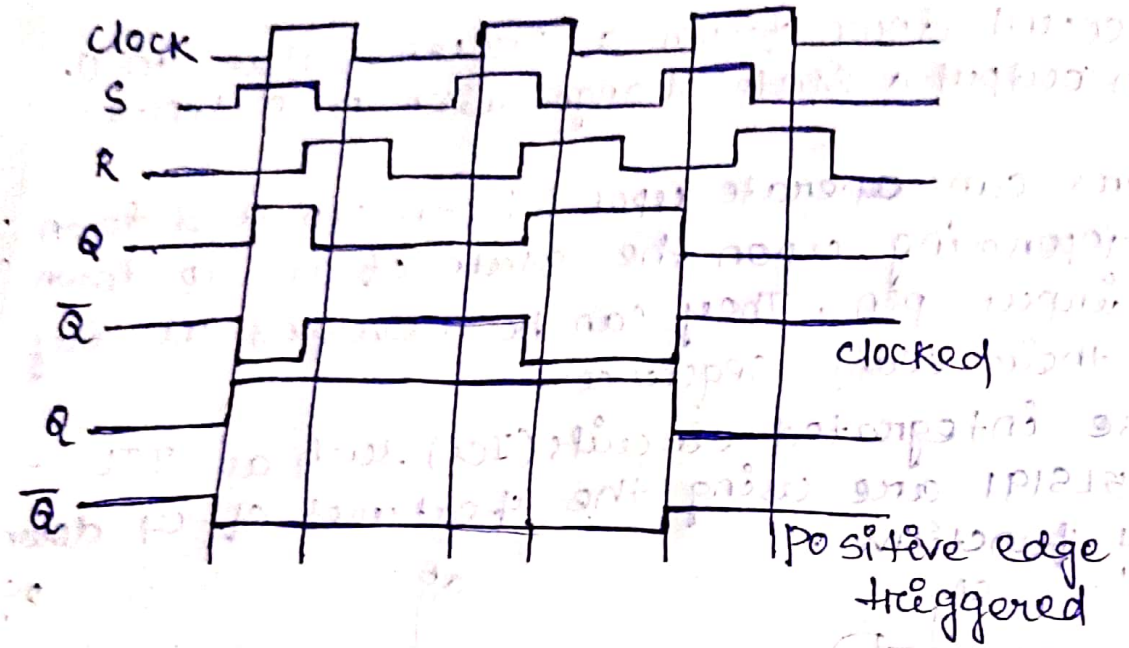
→ FPGA as a application specific integrated circuit has the ability to easily change the functionality of a digital device acc. to feedback after a product has been designed.



Timing Diagram :-

→ Any device that need to communicate with other over serial communication method use these.
→ When number of flip-flops are connected togather there is the use of clock signal at the very beginning of information
→ This specifies rise and fall of signaling of

Timing Diagram of a set / preset flip flop :-



Finite State Machine (Finite automate) :-
(FSM)

→ A finite state machine is a term used by programmers, mathematicians, engineers and other professionals to describe a mathematical model. That has a limited number of conditi States or finite number of states.
→ This is also used to design algorithms for different digital devices.
→ FSMs can be used to model problems in many fiel including mathematics, artificial intelligence, gam linguistics, vending machines, traffic lights, contr in CPU, analysis of protocols, recognition of spee language processing mechanisms, mechaniques of parking meters, Pop machines, automated gas pu etc.

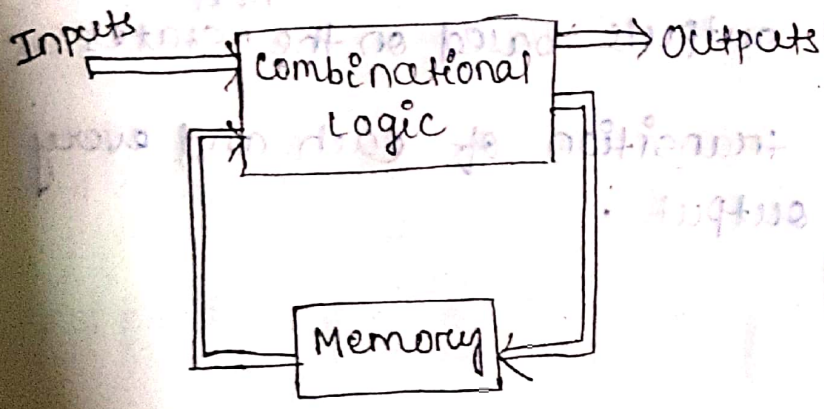
→ A practical example of it is a set of buttons on a video game controller that are connected to a specific set of actions within the game.
→ When a user inputs hitting certain buttons the system knows to implement the actions that correcresponds and allow the game to be played.
→ FSM is nothing but a digital circuit which have finite number of states.
→ Like a counter of 3 bit count there will be 8 possible number of states like 000, 001, 010, ---, 111 (3x8 counter). which are a finite number of states.
→ Actions those are carried out in a FSM are listed below -

i) A set of possible input events
ii) A set of possible output events correspond to the input.
iii) A set of expected states the system can exhibit.
~~iv) A finite state~~

→ A FSM is of two types :-
i) Mealy State Machine
ii) Moore state Machine

Mealy State Machine :-

→ A mealy state machine is a FSM if outputs depend on both present inputs and present states.
→



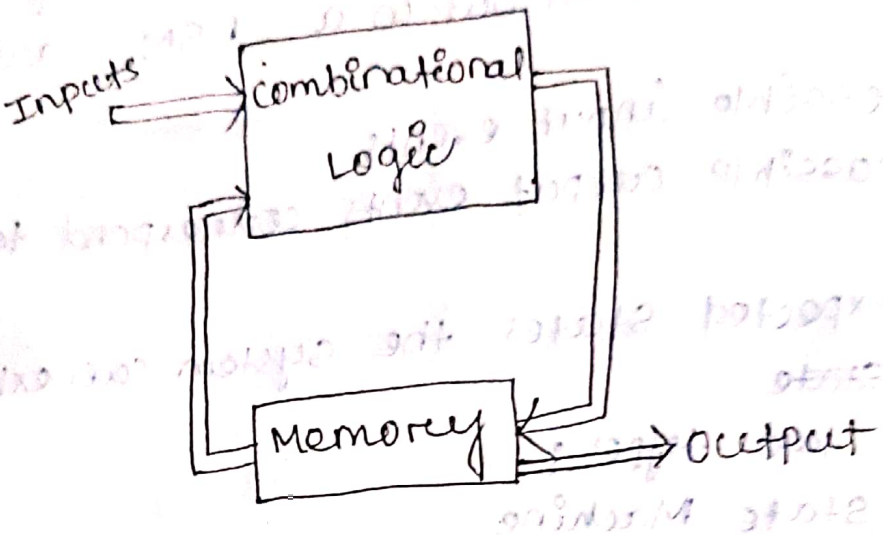
→ This machine have 2 parts -
i) Combinational logic ii) Memory

→ The memory is useful for previous parts of outputs (present states) as inputs combinational logic.
→ Hence based upon the present input and present states the mealy machines produces outputs.

→ **Moore State Machine :-**

→ A FSM is known as moore state machine if outputs depend only on present states.

Block Diagram



→ There are two parts available in moore state machine.
  i) Combinational logic ii) Memory
→ Here the present inputs and the present states determine the next state.
→ Hence it produces outputs based on the next states only.
→ It depends on the transition of each and every state to produce output.

e.g→