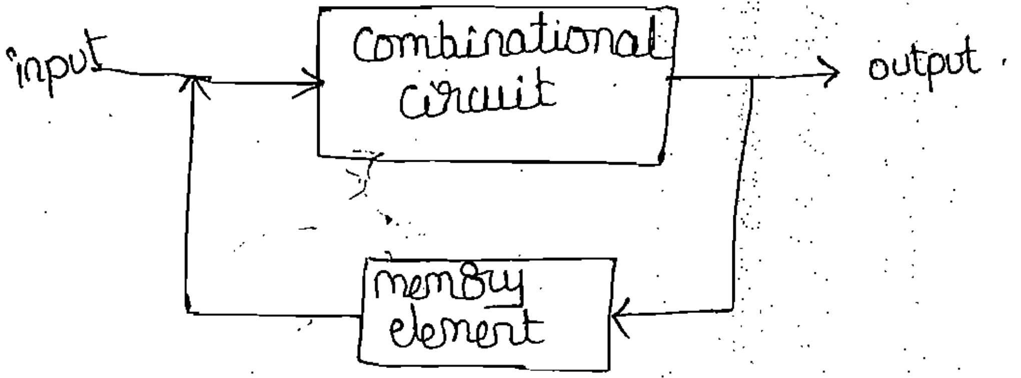
Unit - V
Sequential circuits I.

In sequential logic circuits, the output is a function of the present inputs as well as the past inputs and outputs. sequential circuit include memory element to store the past data. The flip-flop is a basic element of sequential logic circuits.



Block diagram of sequential circuit.

There are two types of sequential circuits.

→ Synchronous circuit :- The sequential circuits which are controlled by a clock are called synchronous sequential circuits. These circuits get actuated only clock signal is present.

→ A synchronous circuit :- The sequential circuits which are not controlled by a clock are called a synchronous sequential circuits, that is the sequential circuits in which events can take place any time the inputs are applied are called A synchronous sequential circuits.

Comparison between Synchronous & Asynchronous sequential ckt

Synchronous sequential circuit	Asynchronous sequential circuit.
1. In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.	1. In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
2. In synchronous circuits, memory elements are clocked FF's.	2. In asynchronous circuits, memory elements are either unclocked FFS or time delay elements.
3. The maximum operating speed of clock depends on time delays involved.	3. Since the clock is not present, asynchronous circuits can operate faster than synchronous circuits.
4. They are easier to design.	4. more difficult to design.

→ latches & Flip flops :-

→ The most important memory element is the flip-flop which is made up of an assembly of logic gates.

→ even though a logic gate by itself has no storage capability, several logic gates can be connected together in ways that permit information to be stored.

→ Flip-flops are the basic building blocks of most sequential circuits. Actually, flip-flops is an one-bit memory device and it can store either 1 & 0.

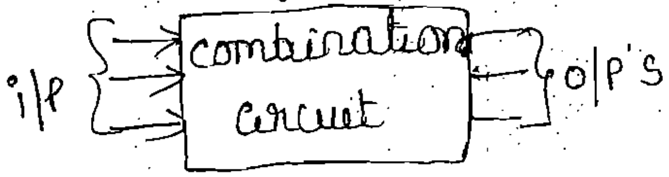
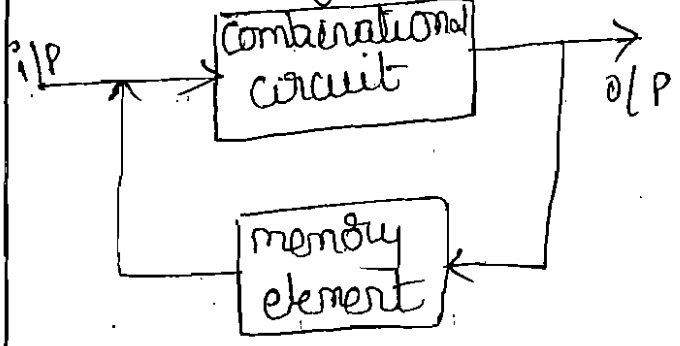
→ latch is a sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

→ It refers to non-clocked flip-flops, because these flip flops 'latch on' to a 1 & a 0 immediately upon receiving the input pulse.

→ Difference between latches & flip flops.

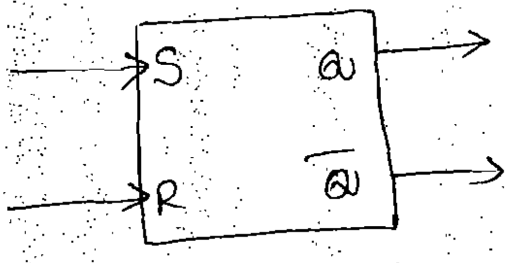
Latch	Flip - flop.
1. A latch is an electronic sequential logic circuit used to store information in an asynchronous arrangement.	1. A flip flop is an electronic sequential logic circuit used to store information in an asynchronous arrangement.
2. one latch can store one bit information, but output state changes only in response to data input.	2. one flip-flop can store one bit-data, but output state changes with clock pulse only.
3. latch is an asynchronous device and it has no clock input.	3. Flip-flop has clock input and its output is synchronised with clock pulse.
4. latch holds a bit value and it remains constant until new inputs force it to change.	4. Flip flops holds a bit value and it remains constant until a clock pulse is received.
5. latches are level -sensitive and the output tracks the input when the level is high. Therefore as long as the level is logic level 1, the output can change if the input changes.	5. Flip flops are edge - sensitive They can store the input only when there is either a rising or falling edge of the clock.

Difference between combinational, sequential circuits.

Combinational circuit	Sequential circuits.
1. The digital logic circuit whose outputs can be determined using the logic function of current state input are combinational logic circuits.	1. The digital logic circuits whose outputs can be determined using the logic function of current state inputs and past state inputs are called as sequential logic circuits.
2. It contains no memory element	2. It contains memory elements.
3. It's behaviour is described by the set of output functions.	3. It's behaviour is described by the set of next state functions and the set of output functions.
4. The combinational logic circuits are independent of the clock.	4. The maximum sequential logic circuits are uses a clock for triggering the flip-flop operation
5. The combinational digital logic circuit don't require any feed back.	5. The sequential digital logic circuits utilize the feedbacks from outputs to inputs.
6. combinational circuits are easy to design.	6. Sequential circuits are complex to design.
7. combinational circuits are faster because the delay between the input and the output is due to propagation delay of gates only.	7. sequential circuits are slower than combinational circuits.
8. Block diagram 	8. Block diagram. 

S-R latch :-

The S-R latch has two inputs, namely SET(S) and RESET(R), and two outputs Q and $\bar{Q}$, where $\bar{Q}$ is the complement of Q.



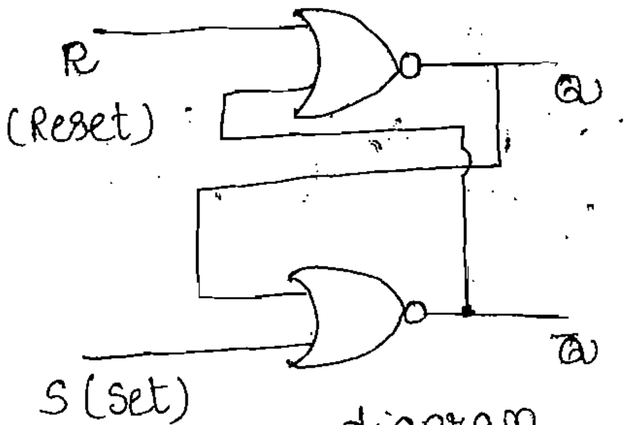
Logic symbol of S-R latch

S-R latch using NOR Gates :- [active-high S-R latch].

The logic diagram of the S-R latch composed of two cross-coupled NOR gates. S and R are two inputs of S-R latch.

→ S stands for set, it means that when S is 1, it stores 1.

→ R stands for Reset, and if R=1, latch Reset and it's output will be 0. This circuit is called as NOR gate latch or S-R latch.



logic diagram.

S	R	Q_n	Q_{n+1}	state
0	0	0	0	NO change
0	0	1	1	
0	1	0	0	ReSET
0	1	1	0	
1	0	0	1	SET
1	0	1	1	
1	1	0	X	indetermined (invalid).
1	1	1	X	

Truth table.

Simplified truth table.

S	R	Q_{n+1}	State
0	0	Q_n	NO change
0	1	0	Reset
1	0	1	Set
1	1	X	invalid state

1. SET=0, RESET=0 : This is the nomal resting state of the NOR latch and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were prior to the occurence of this input condition.

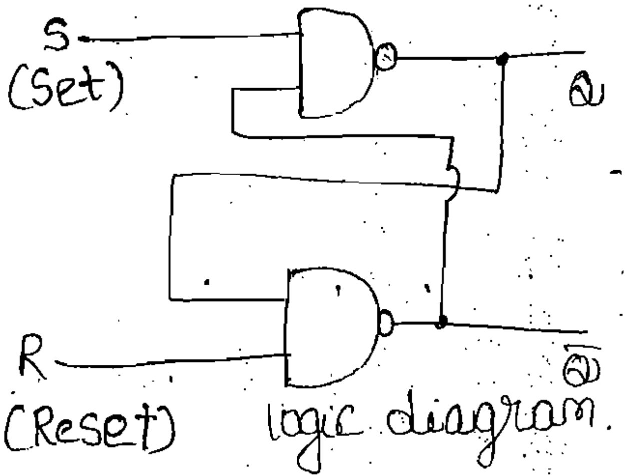
2. SET=0, RESET=1, This will always reset Q=0, where it will remain even after RESET returns to 0.

3. SET=1, RESET=0, This will always set Q=1, where it will remain even after SET returns to 0.

4. SET=1, RESET=1. This condition tries to SET and Reset the latch at the same time, and it produces $Q=\overline{Q}=0$. if the input are returned to zero simultaneoulsly, the resulting output state is erratic and unpredictable. This input condition should so not be used. it is indetermined state or invalid state.

S-R latch using NAND Gates:- (active low S-R latch)

→ The logic diagram of the S-R latch composed of two cross-coupled NAND gates.



logic diagram.

S	R	Q_n	Q_{n+1}	State
0	0	0	X	indetermined (invaild)
0	0	1	X	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No Change
1	1	1	1	

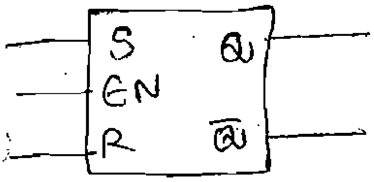
Truthtable.

S	R	Q_{n+1}	State
0	0	X	invaild
0	1	1	Set
1	0	0	Reset
1	1	Q_n	N.C

← Simplifeed truth table.

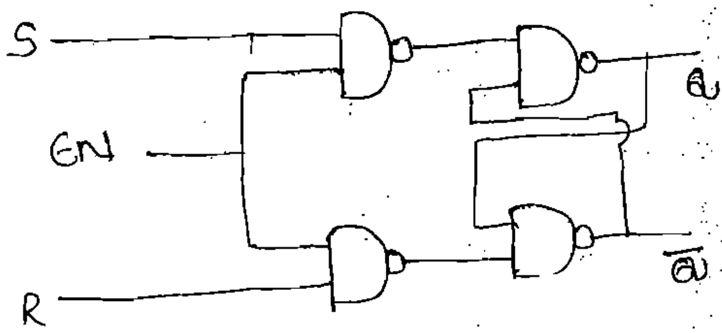
Gated latches :-

The gated S-R latch :- The output can change state any time the input conditions are changed, so they are called Asynchronous latches. A gated S-R latch requires an Enable (EN) input. Its S and R inputs will control the state of latch only when the enable is high. when the enable is low, the inputs become ineffective and no change of state can take place.

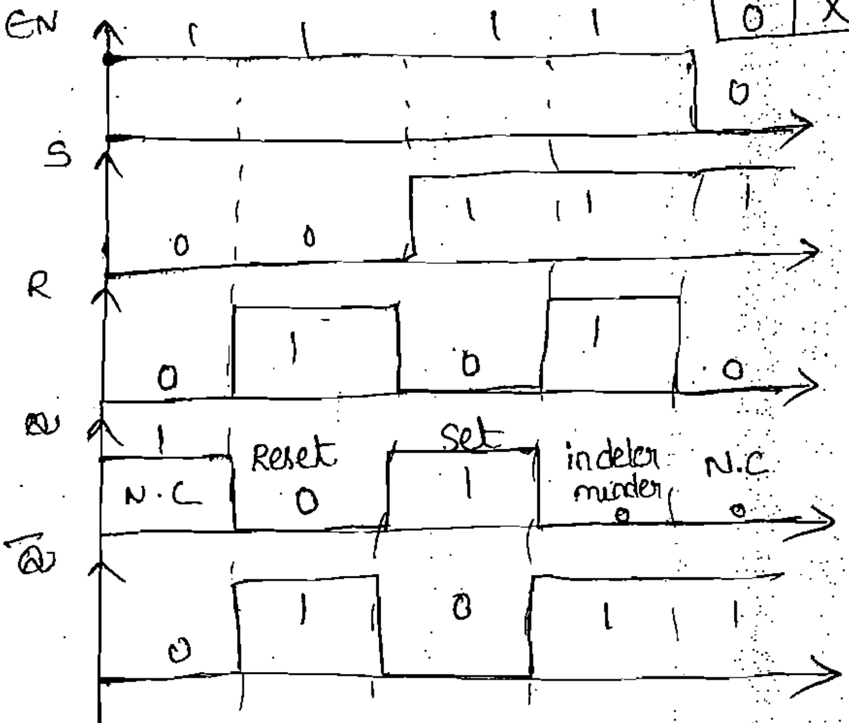


logic symbol.

EN	S	R	Qn	Qn+1	State
1	0	0	0	0	NO change
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	indeterminate (invalid)
1	1	1	1	X	
0	X	X	0	0	NO change.
0	X	X	1	1	



logic diagram.



waveforms for Grated S+R latch.

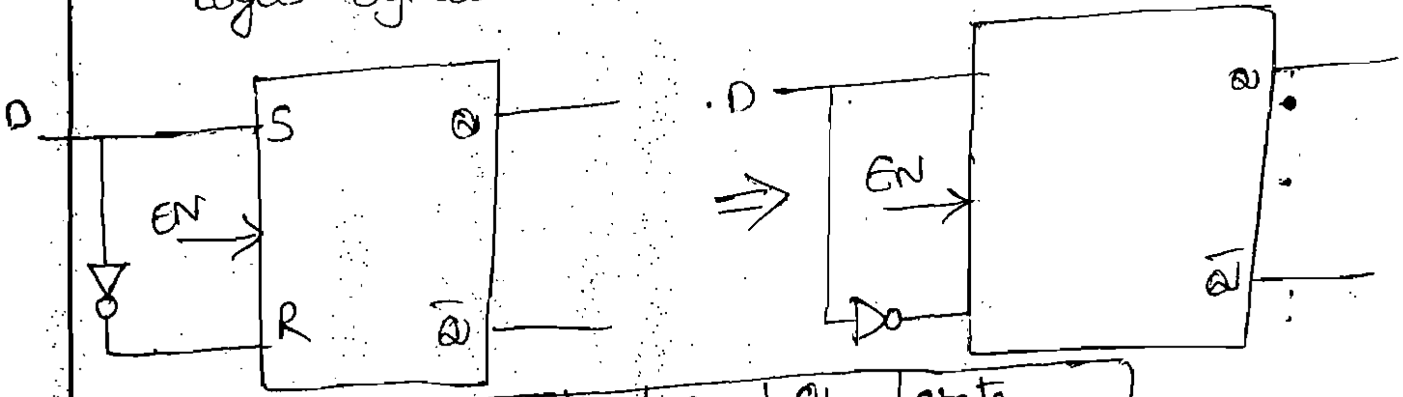
The Grated D-latch; - In many applications, it is not necessary to have separate s and R inputs to a latch. If the input combinations $S=R=0$ and $S=R=1$ are never needed, the S and R are always the complement of each other. So, to construct a latch with a single input (S) and obtain the R input by inverting it. This single input is labelled D (for data). and the device is called a D-latch.

→ when $D=1$, $S=1$ and $R=0$, causing the latch to set when Enabled. when $D=0$, $S=0$ and $R=1$, causing the latch to Reset when enabled. when EN is low, the latch is ineffective, and any change in the value of D input does not affect the output at all.

→ when, EN is high, a low D-input makes Q low, i.e resets the latch and high D input makes a Q high, that is sets the latch.

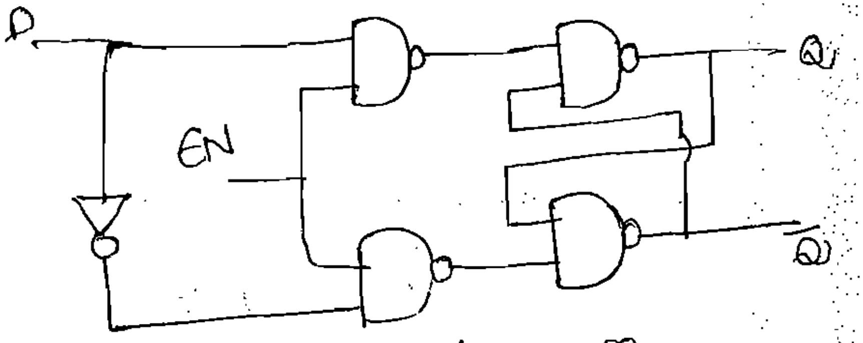
→ The output Q follows the D-input when EN is high. So this latch is said to be transparent.

Logic Symbol

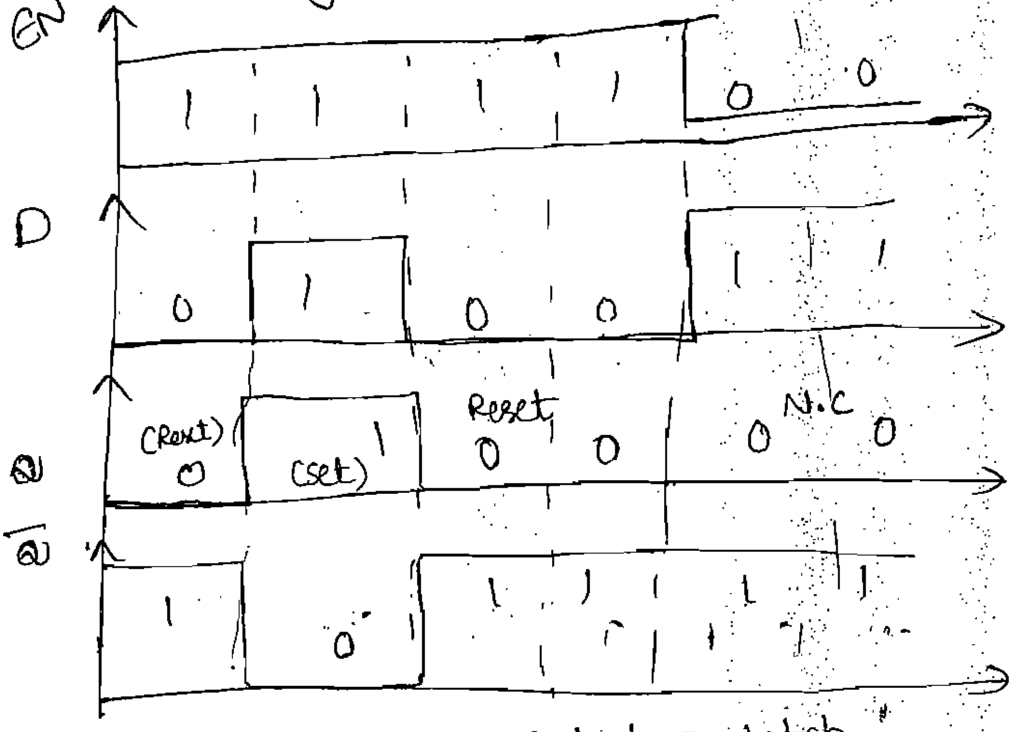


EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	X	0	0	NO change (NC)
0	X	1	1	

Truth table



logic diagram.



waveforms for Gated D-latch.

Flip-flops :-

Types of flip-flops :-

→ S-R flip-flop

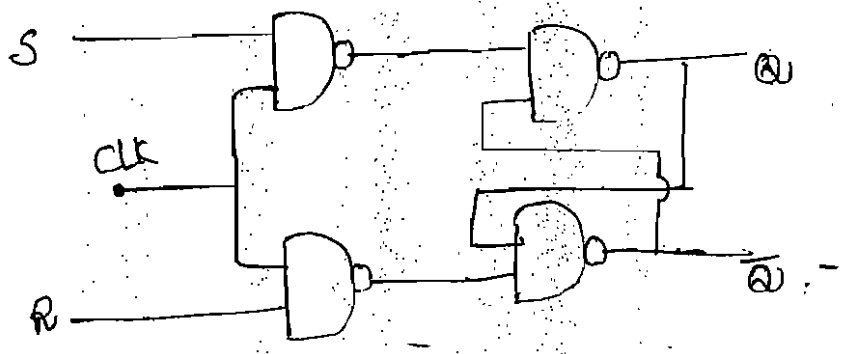
→ D flip-flop

→ J-K flip-flop

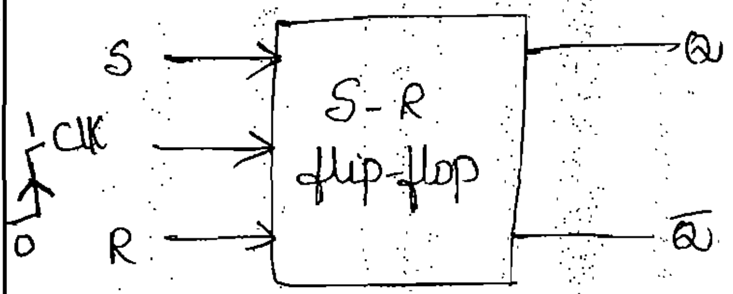
→ T flip-flop.

S-R flip-flop:-

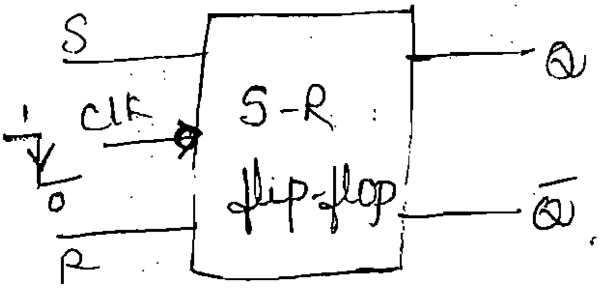
logic diagram.



symbol of +ve edge trigger.



symbol of - ve edge trigger



Truth table:-

Clk	S	R	Q_n	Q_{n+1}	State
↑	0	0	0	0	NO change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	X	invalid State
↑	1	1	1	X	
0	X	X	0	0	NO change
0	X	X	1	1	

Characteristic equation:- The characteristic equation of a flip-flop is the equation expressing the next state of a flip-flop in terms of its present state and present excitations. To obtain the characteristic equation of a.

flip-flop write the excitation requirements of the flip-flop, draw a K-map for the next state of the flip-flop in terms of its present state and inputs and simplify it.

S \ RQn	00	01	11	10
0		1		
1	1	1	X	X

$Q_{n+1} = S + \bar{R}Q_n$.

Truth table :-

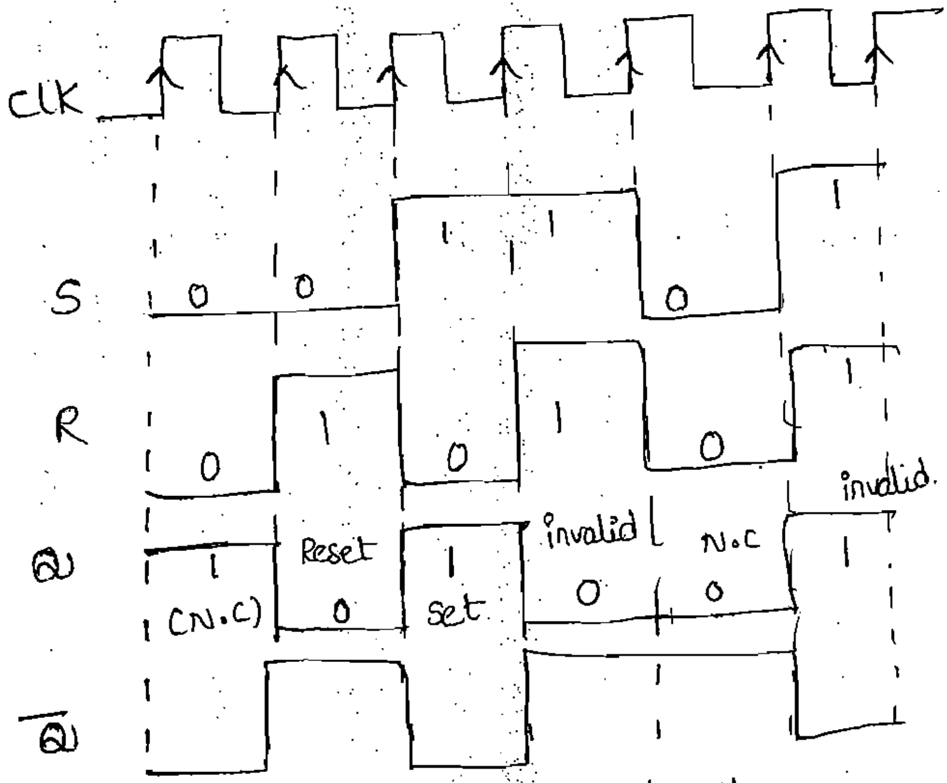
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	X

Excitation table :- A table which lists the present state, the next state and the excitations of a flip-flop is called the excitation table.

→ A table which indicates the excitations required to take the flip-flop from the present state to the next state.

Present State Q_n	next State Q_{n+1}	Required S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

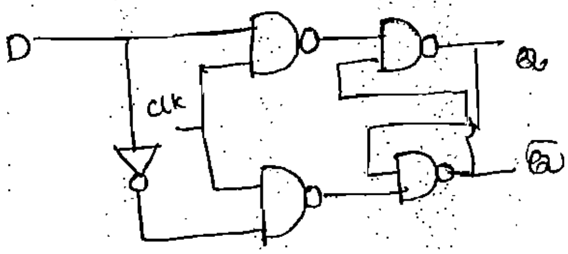
Timing diagram :-



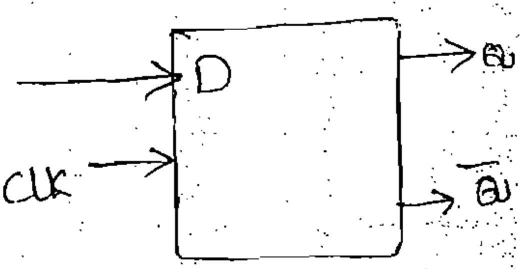
waveforms for S-R flip-flop.

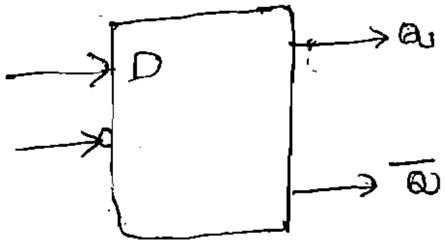
D- flip -flop :-

logic diagram :-



Symbol of +ve edge trigger   Symbol of -ve trigger





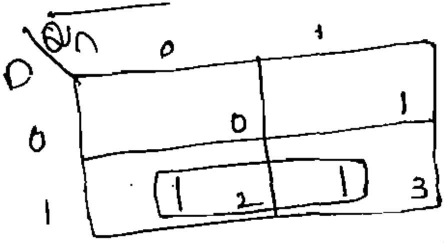
Truth table :-

D	Q_{n+1}
0	0
1	1

characteristic Table.

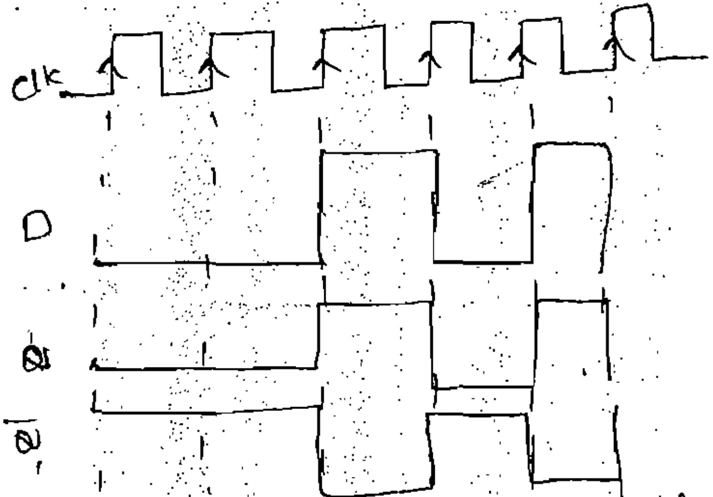
Clk	D	Q_n	Q_{n+1}	State
↑	0	0	0	Reset
↑	0	1	0	
↑	1	0	1	Set
↑	1	1	1	
↑	X	0	0	NO change
↑	X	1	1	

characteristic equation :-



$Q_{n+1} = D.$

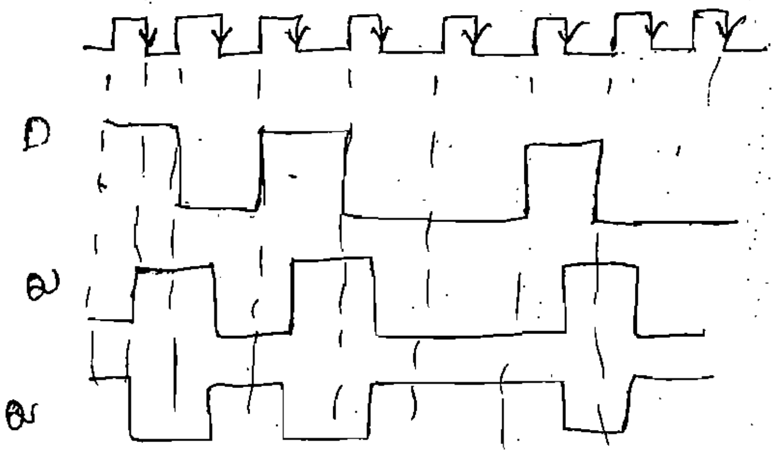
waveforms



(Positive-edge trigger clk).

Excitation table :-

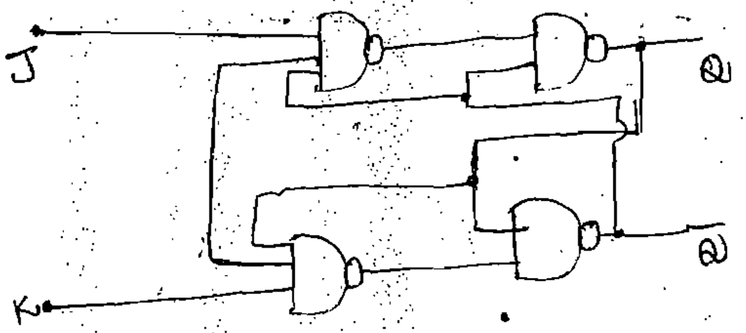
Present state Q_n	Next state Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1



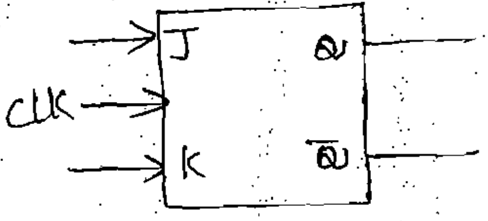
(negative-edge trigger clk).

(J-K - flip flops).

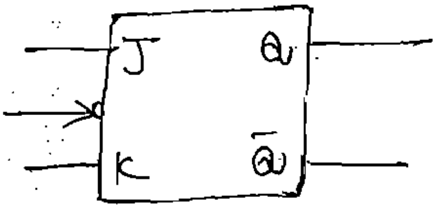
logic diagram



logic symbol.



(positive - edge)



(negative - edge)

Truth table:-

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

characteristic table

clk	J	K	Q_n	Q_{n+1}	State
↑	0	0	0	0	No change
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	1	Toggle
↑	1	1	1	0	
0	X	X	0	0	No change
0	X	X	1	1	

Characteristic equation

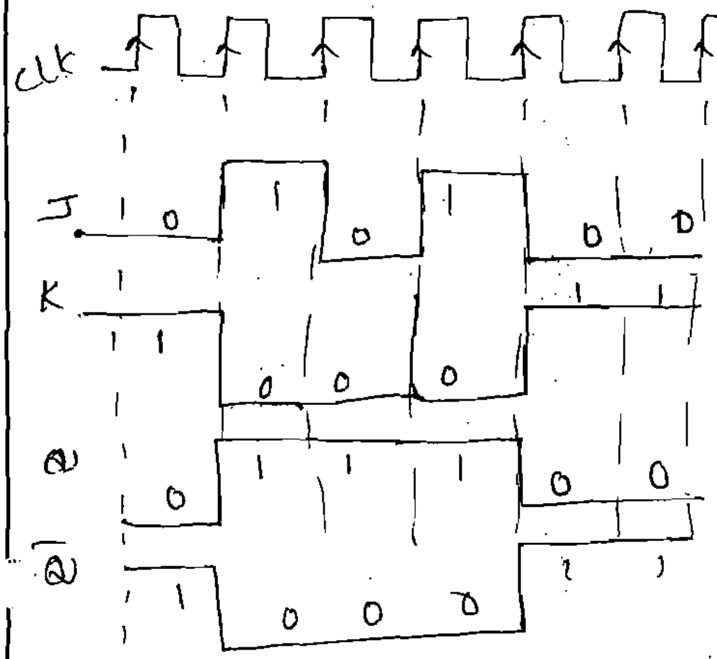
J \ KQ_n	00	01	11	10
0	0	1 1	3	2
1	1 4	1 5	7	1 6

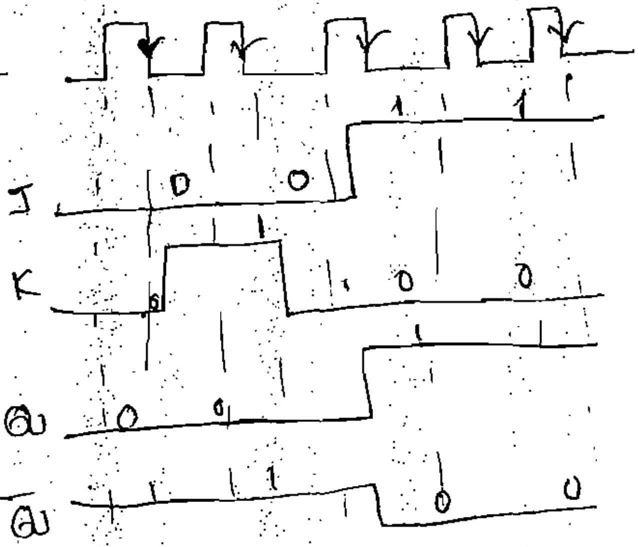
$Q_{n+1} = \overline{K}Q_n + J\overline{Q_n}$

excitation table

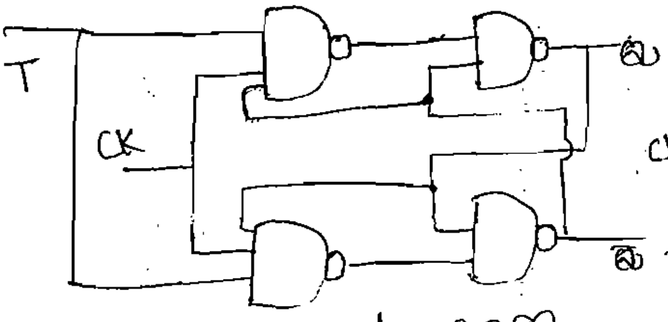
Present state	next state	inputs J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

wave-forms for Positive-edge triggering | negative-edge triggering

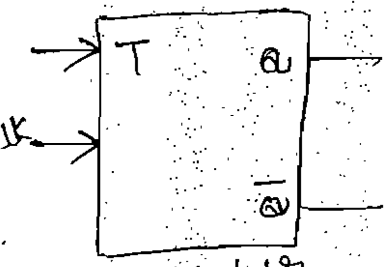




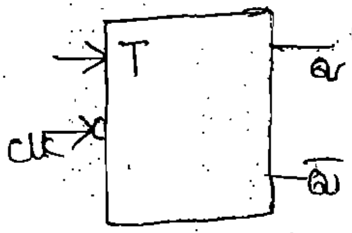
T-flip flop ;-



logic diagram.



symbol for (Positive-trigger)



symbol for (negative-trigger)

Truth table

T	Q_{n+1}
0	Q_n
1	0

characteristic table

CLK	T	Q_n	Q_{n+1}	State
↑	0	0	0	NO change
↑	0	1	1	
↑	1	0	1	Toggle
↑	1	1	0	
$\cancel{\uparrow}$	X	0	0	NO change
$\cancel{\uparrow}$	X	1	1	

characteristic equation

T \ Q_n	0	1
0	0	1 1
1	1 2	3

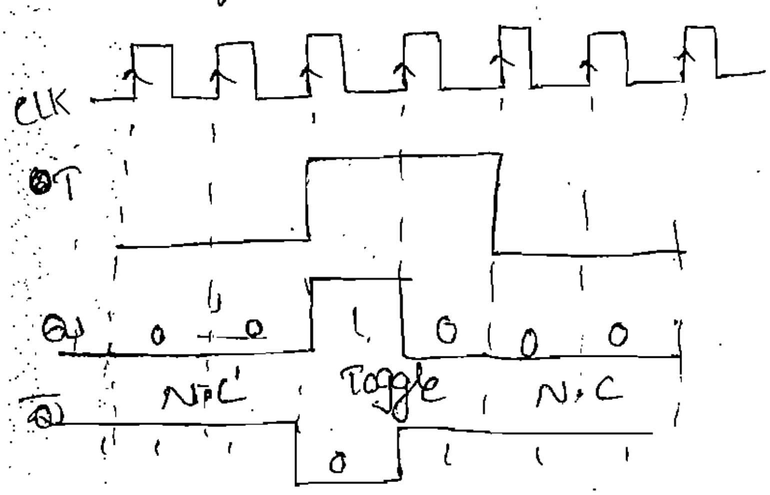
$$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n.$$

~~Exsiton~~

Excitation table :-

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

waveforms .



Race - Around condition :-

If the width of the clock pulse tp is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0, 0 to 1 and so on, and at the end of the clock pulse, its state will be uncertain. This phenomenon is called the race around condition. The outputs Q and $\bar{Q}$ will change on their own if the clock pulse width tp is too long compared with the propagation delay τ of each NAND Gate .

$$t_p >> \tau$$

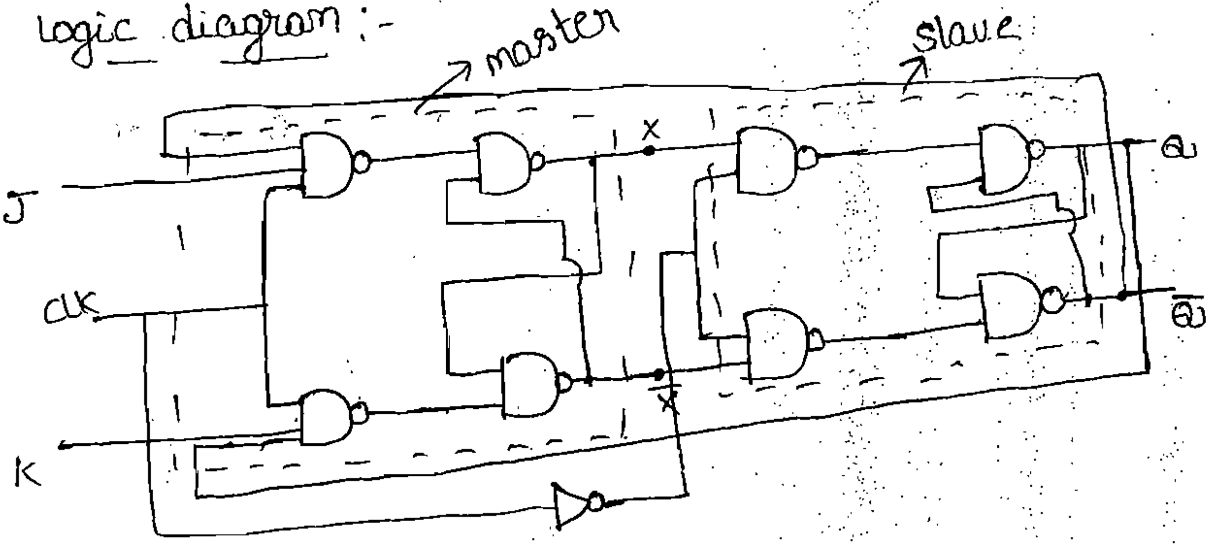
t_p → pulse width.

τ → propagation delay.

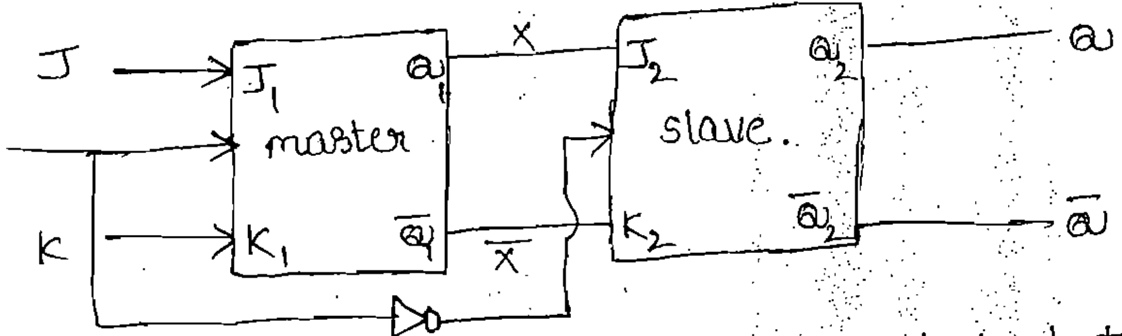
The clock pulse width should be such as to allow only one change to complement the state and not too long to allow many changes resulting in uncertainty about the final state. This is a stringent requirement which cannot be ensured in practice. This problem is eliminated using master - slave flip-flop or edge trigged flip-flop.

master - slave J-K flip - flop :-

logic diagram :-



Symbol ;-



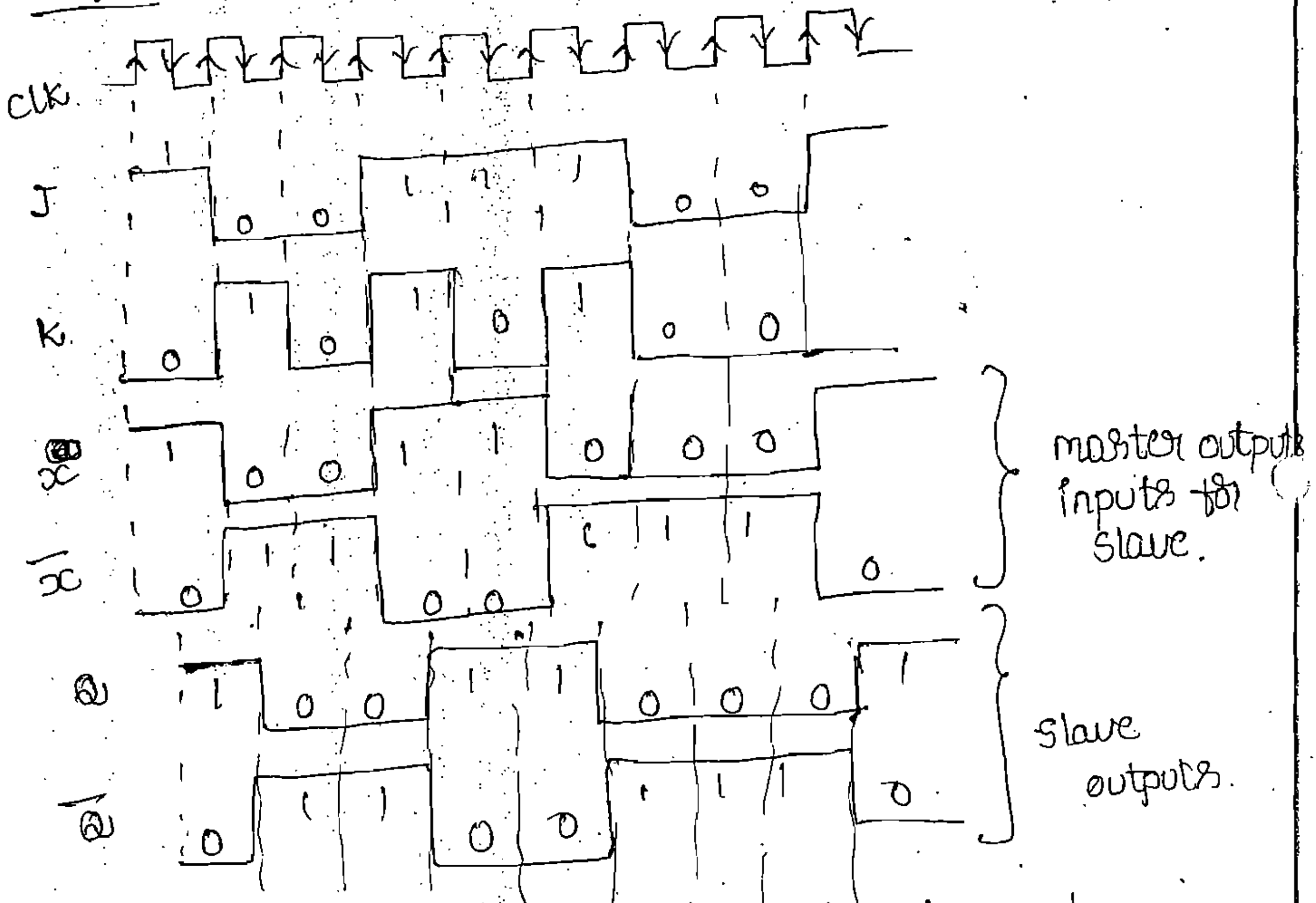
The master - slave flip flop was developed to make the synchronous operation more predictable, that is, to avoid the problems of logic race in clocked flip-flops. This improvement is achieved by introducing a known time delay between the time that the flip-flop responds to a clock pulse and the time the response appears at its output. A master - slave flip-flop is also called a pulse - triggered flip-flop because the length of the time required for its output to change state equals the width of one clock pulse.

In master - slave J-K flip - flop actually contains two flip-flops - a master flip - flop and a slave flip - flop. The control inputs are applied to the master flip - flop and master output is given as input to the

Slave flip-flop. on the rising edge of the clock pulse, the levels on the control inputs are used to determine the output of the master on the falling edge of the clock pulse, the state of the master is transferred to the slave, whose outputs are equal to the master.

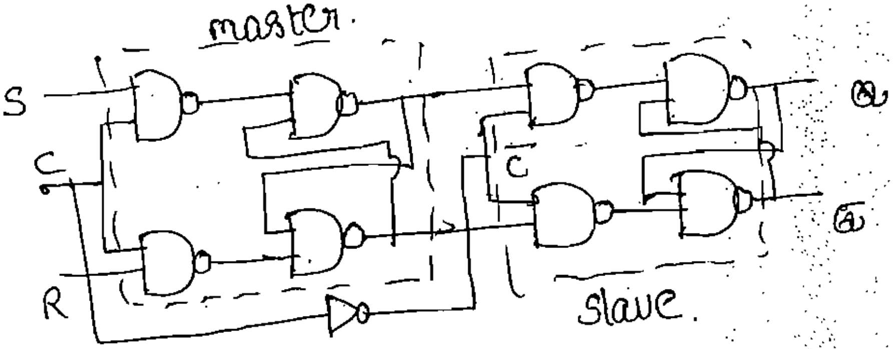
The master-slave flip-flops function very much like the negative-edge triggered flip-flops except for one more disadvantage The control inputs must be held stable while clk is high, otherwise an unpredictable operation may occur. This problem with the master-slave flip-flop is overcome with an improved master-slave version called the master-slave with data lock-out

waveforms :-



in slave accept the negative edge triggered signal only. master accept the positive-edge triggered signal only.

master- slave S-R flip flop :-

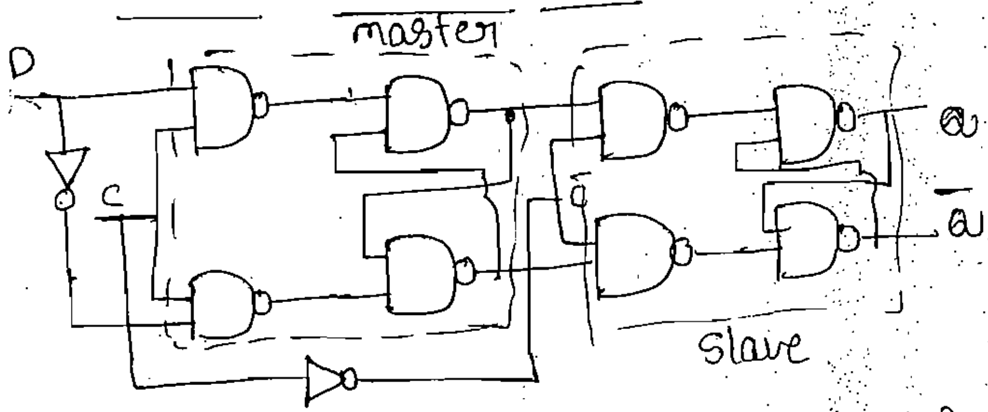


Logic diagram

S	R	clk	Q	State
0	0	⎍	Q_0	N.C
0	1	⎍	0	Reset
1	0	⎍	1	set
1	1	⎍	?	invalid

Truth table

master - slave D flip flop ;-



Truth table

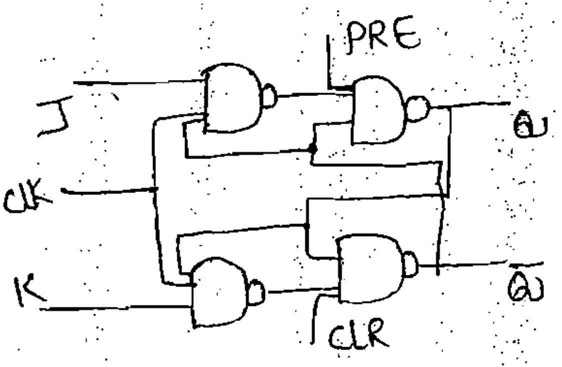
D	clk	Q	State
0	⎍	0	Reset
1	⎍	1	set

→ Asynchronous inputs (PRESET and CLEAR).

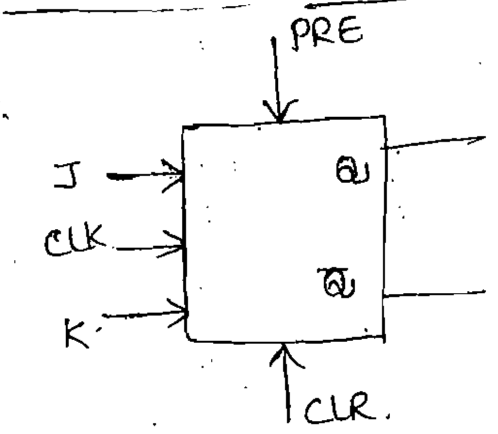
The S-R, D, T and J-K inputs are called synchronous inputs, because their effect on the flip-flop output is synchronized with the clock input.

most IC flip-flops also have one or more asynchronous inputs. These Asynchronous inputs affect the flip-flop output independently of the synchronous inputs and the clock input These asynchronous inputs can be used to SET the flip-flop to the 1 state or RESET the flip-flop to the 0 state at any time regardless of the conditions at the other inputs. They are normally labelled PRESET or direct SET or DC SET, and CLEAR or direct RESET or DC CLEAR.

J-K flip-flop with Active-high Asynchronous inputs :-



Logic diagram.



→ When PRE = 0, CLR = 0 then DC SET = 1 and DC CLEAR = 1, The Asynchronous inputs are inactive and the flip-flop responds freely to J, K and clk inputs in the normal way. In other words, the clocked operation can takes place.

→ when PRE = 0, CLR = 1 then DC SET = 0 and DC CLEAR = 1
The)×

→ when PRE = 0, CLR = 0 then Asynchronous inputs are inactive and the flip-flop responds freely to J, K and clk inputs.

→ when PRE = 0, CLR = 1 then Asynchronous input clear is active then flip flop output is '0'.

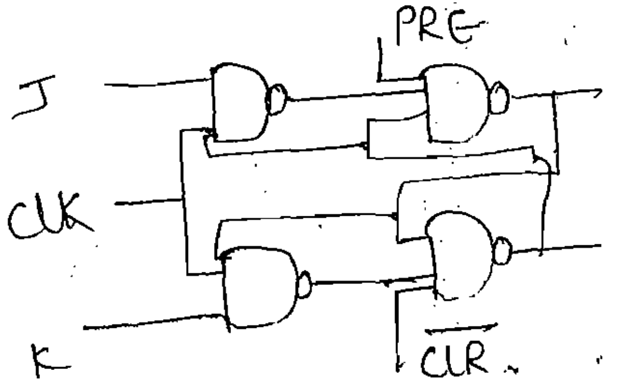
→ when PRE = 1, CLR = 0 then Asynchronous input PRESET is active the flip flop output is '1'.

→ when PRE = '1', CLR = '1'. This condition should not be used since it can result in an invalid state.

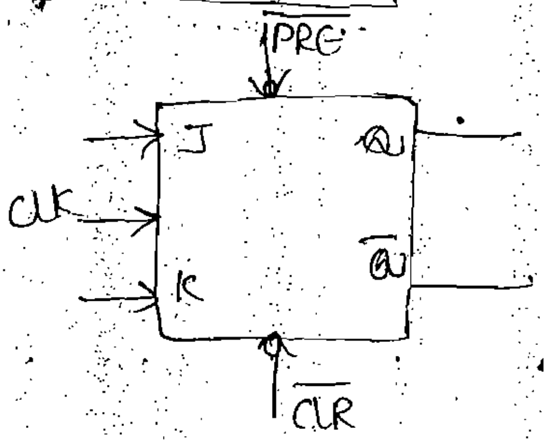
DC SET (PRE)	DC RESET (CLR)	FF response
0	0	clock operation
0	1	Q = 0
1	0	Q = 1
1	1	Not used.

Truth table

J-K flip-flop with Active-low Asynchronous inputs



logic diagram



- → when $\overline{PRE}$ =0 and $\overline{CLR}$ =0, This condition should not be used. since it can result in an invalid State.
- → when $\overline{PRE}$ =0 and $\overline{CLR}$ = 1, then Asynchronous input $\overline{PRESET}$ is active then output is '1'.
- → when $\overline{PRE}$ = 1 and $\overline{CLR}$ = 0, then Asynchronous input $\overline{CLEAR}$ is active then output is '0'.
- → when $\overline{PRE}$ =1 and $\overline{CLR}$ = 1, Then Asynchronous inputs are inactive ~~the~~ and the flip responds freely to J, K and CLK inputs.

DC SET ($\overline{PRE}$)	DC RESET ($\overline{CLR}$)	FF response
0	0	not used
0	1	Q = 1
1	0	Q = 0
1	1	clock operation

Truth table.

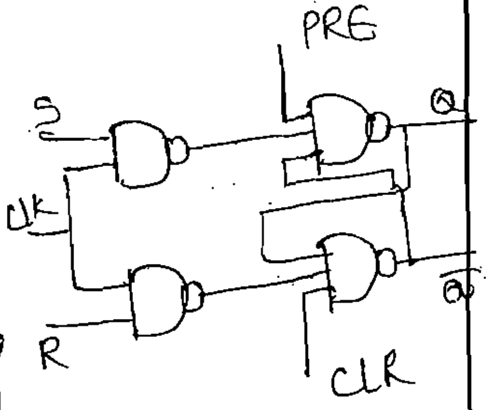
Truth table for J-K flip-flop (both Asynchronous & synchronous inputs).

PRE	CLR	CLK	J	K	Q	$\overline{Q}$
1	1	X	X	X	invalid	
1	0	X	X	X	1	0
0	1	X	X	X	0	1
0	0	↑	0	0	0	1
0	0	↑	0	1	0	1
0	0	↑	ϕ	0	1	0
0	0	↑	1	1	0	1

X - don't care means either '0' or '1'.

Similarly S-R flip-flop

PRE	CLR	CLK	S	R	Q	$\overline{Q}$
1	1	X	X	X	invalid	
1	0	X	X	X	1	0
0	1	X	X	X	0	1
D	D	↑	0	0	0	1
0	0	↑	0	1	0	1
0	0	↑	1	0	1	0
0	0	↑	1	1	invalid	
0	0	0	X	X	N.C.	

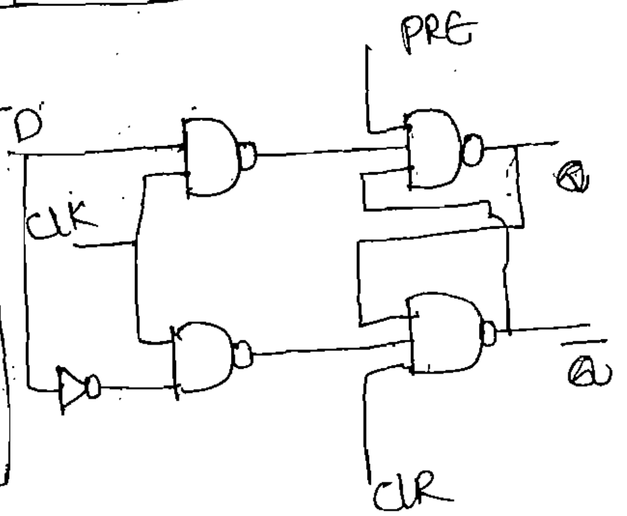


logic diagram

Similarly for D - flip-flop.

PRE	CLR	CLK	D	Q	$\overline{Q}$
1	1	X	X	invalid	
1	0	X	X	1	0
0	1	X	X	0	1
0	0	↑	0	0	1
0	0	↑	1	1	0

Truth table



logic diagram

Flip-flop conversions:-

Step 1:- obtain the characteristic table of required flip-flop.

Step 2:- And also obtain the Excitation table of available flip-flop.

Step 3:- obtain the Expressions for the inputs of the existing flip-flop in terms of the inputs of the required flip-flop and the present state variables of the existing flip-flop and implement them.

Conversion of T-flip-flop to S-R flipflop

Available flip-flop → T-flip flop (Excitation table)

Required flip-flop → S-R flip-flop (characteristic table)

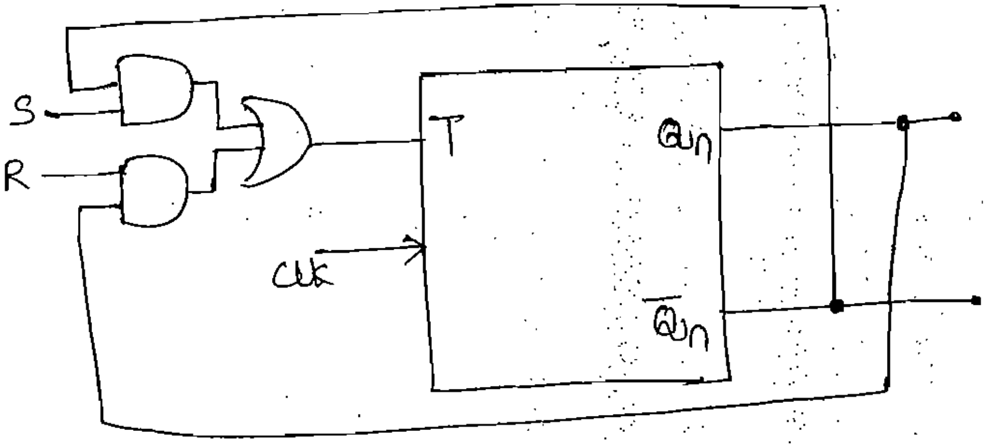
S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

K-map for T

S \ RQ_n	00	01	11	10
0	0_0	0_1	1_3	0_2
1	1_4	0_5	X_7	X_6

$T = S.\bar{Q}_n + R.Q_n.$

logic diagram



→ conversion of T-flip-flop to D-flip-flop.

Available → T-flipflop → excitation table

Required → D-flipflop → characteristic table

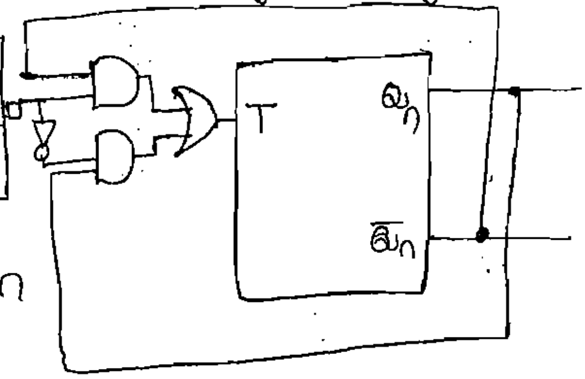
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

K-map for T

D \ Q_n	0	1
0		1
1	1	

$T = D\bar{Q}_n + \bar{D}Q_n$

logic diagram



→ construct a D-flip flop ~~SR~~ using S-R flip flop.

Avaliable → S-R flipflop → excitation table.

Requried → D flip flop → characteristic table.

D	Q_n	Q_{n+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 for S

D \ Q_n	0	1
0	0	1
1	1 3	X 2

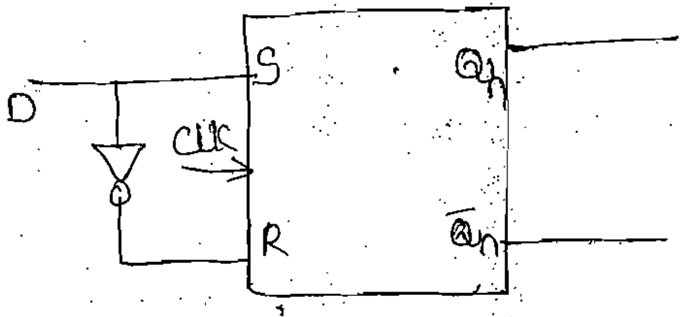
$S = D$

K-map for R

D \ Q_n	0	1
0	X 0	1 1
1	3	2

$R = \bar{D}$

logic diagram.



→ Relize the S-R flip-flop by using J-K flip flop.

Availiable → S-R → excitation table

Required → J-K → characteristic table

J	K	Q_n	Q_{n+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

K-map for S

J \ KQ_n	00	01	11	10
0	0_0	X_1	0_3	0_2
1	1_4	X_5	0_7	1_6

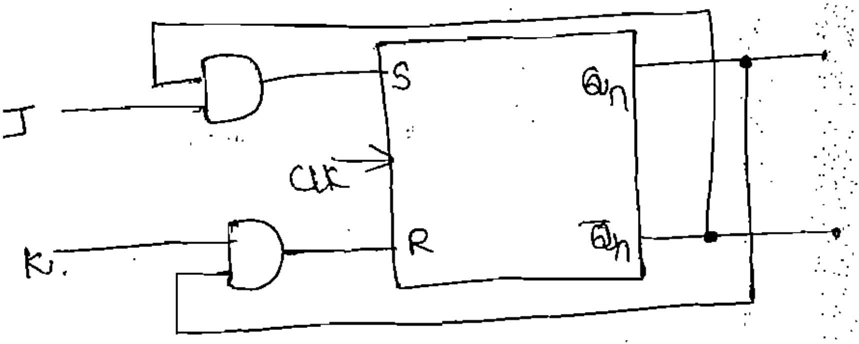
$S = J\bar{Q}_n$

K-map for R

X	0	1	X
0	0	1	0

$R = KQ_n$

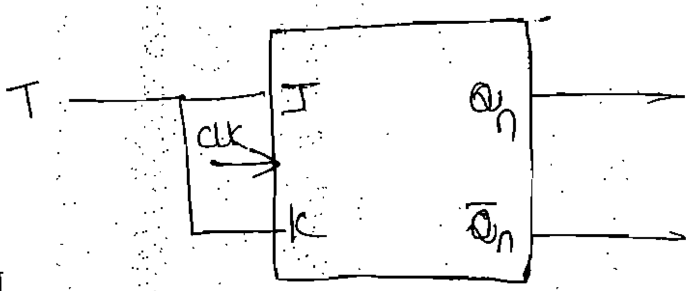
logic diagram:-



⇒ convert J-K flip-flop to T flip-flop

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

logic-diagram



K-map for J

T \ Q_n	0	1
0	0	X
1	1_2	X_3

$J = T.$

K-map for k

T \ Q_n	0	1
0	X	0
1	X	1

$K = T$

→ conversion of D-flip flop to T flip-flop.

* Available → D-flip flop → excitation table
* Available → T-flip flop → characteristic table.

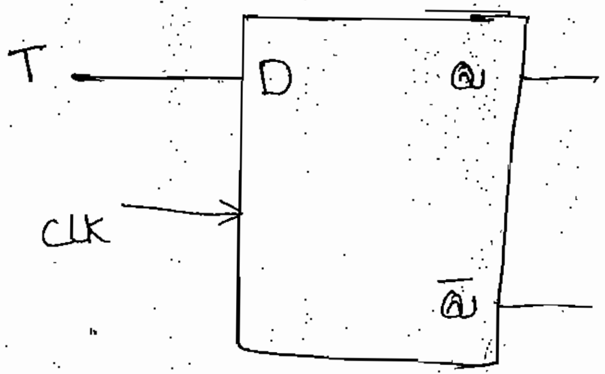
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

K-map for D.

T \ Q_n	0	1
0		1
1	0	1

D = T.

logic diagram



Counters :- A digital counter is a set of flip-flops whose states change in response to pulses applied at the input to the counter.

→ The name itself it indicates, a counter is used to count pulses.

→ counters may be asynchronous counters & synchronous counters. Asynchronous counters are also called ripple counters.

→ In asynchronous counters Flipflops are not triggered simultaneously. The clock does not directly control the time at which every stage changes state.

→ In synchronous counters are clocked such that each FF in the counter is triggered at the same time.

comparison of synchronous and Asynchronous counters

Asynchronous counters	synchronous counters.
1. In this type of counter FF's are connected in such a way that the output of first FF drives the clock for the second FF, the output of the second FF to the third FF.	1. In this type of counter there is no connection between the output of first FF and clock input of next FF and so on.
2. All the FF's are not clocked simultaneously.	2. All the FFS are clocked simultaneously.
3. Design and implement is very simple even for more number of states.	3. Design and implementation becomes tedious and complex as the number of states increases.
4. main drawback of these counters is their low speed as the clock is propagated through a number of FFS before it reaches the last FF.	4. since clock is applied to all the FF's simultaneously the total propagation delay is equal to the propagation delay of only one FF. Hence they are faster.

Synchronous counters :-

→ synchronous counters have the advantages of high speed and less severe decoding problems but the disadvantage of having more circuitry than that of Asynchronous counters.

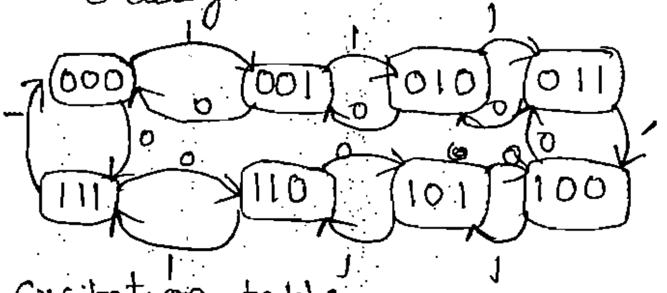
Design steps of synchronous counters ;-

1. number of flip-flops
2. state diagram
3. choice of flip-flops and excitation table.
4. minimal Expression for excitations
5. logic diagram.

→ Design of a synchronous 3-bit up-down counter using J-K FF'S

Step 1 ;- A 3-bit counter requires 3-FFS. It has 8 states. (000 - - - . 111) and all the states are valid.

Step 2 ;- state diagram.



m=0 down counting
m=1 up counting

Step 3 ;- Excitation table.

mode (m)	Present State Q_3	Q_2	Q_1	Next state Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	1	1	1	1	X	1	X	1	X
0	0	0	1	0	0	0	0	X	0	X	X	1
0	0	1	0	0	0	1	0	X	X	1	1	X
0	0	1	1	0	1	0	0	X	X	0	X	1
0	1	0	0	0	1	1	X	1	1	X	1	X
0	1	0	1	1	0	0	X	0	0	X	X	1
0	1	1	0	1	0	1	X	0	X	1	1	X
0	1	1	1	1	1	0	X	0	X	0	X	1

mode(m)	PS $Q_3Q_2Q_1$	N.S $Q_3\ Q_2\ Q_1$	J_3	K_3	J_2	K_2	J_1	K_1
1	0 0 0	0 0 1	0	X	0	X	1	X
1	0 0 1	0 1 0	0	X	1	X	X	1
1	0 1 0	0 1 1	0	X	X	0	1	X
1	0 1 1	1 0 0	1	X	X	1	X	1
1	1 0 0	1 0 1	X	0	0	X	1	X
1	1 0 1	1 1 0	X	0	1	X	X	1
1	1 1 0	1 1 1	X	0	X	0	1	X
1	1 1 1	0 0 0	X	1	X	1	X	

Step 4 :- obtain the minimal expression.

$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	1_0	$_1$	$_3$	$_2$
01	$_4$	$_5$	1_7	$_6$
11	X_{12}	X_{13}	X_{15}	X_{14}
10	X_8	X_9	X_{11}	X_{10}

$J_3 = \overline{Q_2}\,\overline{Q_1}\,\overline{M} + Q_2Q_1M.$

$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	X_0	X_1	X_3	X_2
01	X_4	X_5	X_7	X_6
11	$_{12}$	$_{13}$	1_{15}	$_{14}$
10	1_8	$_9$	$_{11}$	$_{10}$

$K_3 = \overline{Q_2}\,\overline{Q_1}\,\overline{M} + Q_2Q_1M.$

$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	1		1	
01	X	X	X	X
11	X	X	X	X
10	1		1	

$J_2 = \overline{Q_1}\,\overline{M} + Q_1M$

$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	X	X	X	X
01	1		1	
11	1		1	
10	X	X	X	X

$K_2 = \overline{Q_1}\,\overline{M} + Q_1M$

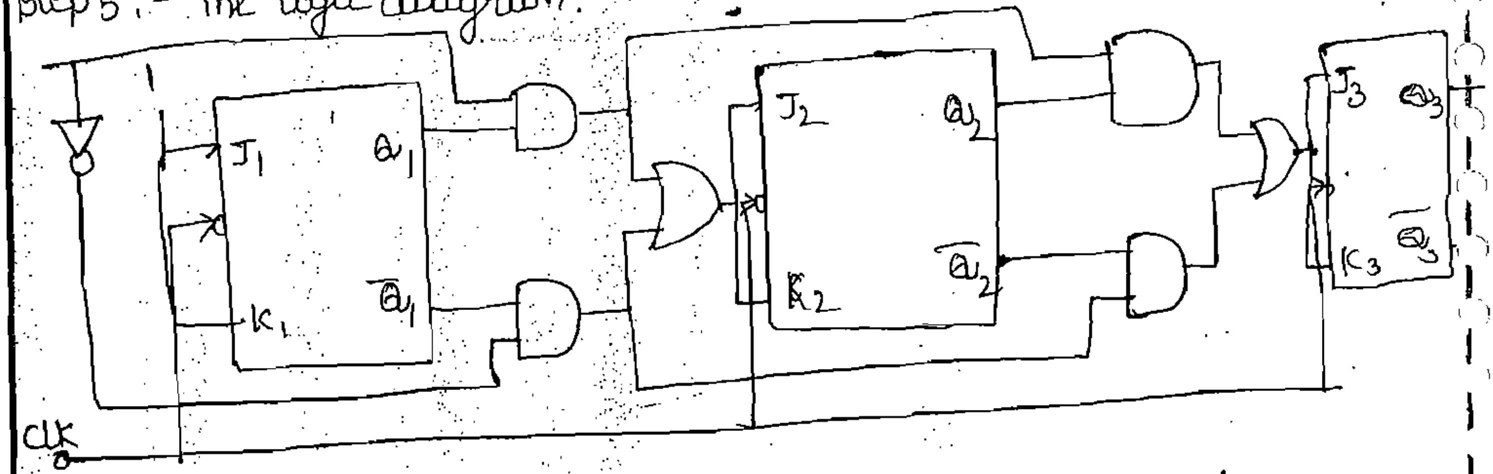
$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	1_0	1_1	X_3	X_2
01	1_4	1_5	X_7	X_6
11	1_{12}	1_{13}	X_{15}	X_{14}
10	1_8	1_9	X_{11}	X_{10}

$J_1 = 1$

$Q_3Q_2 \backslash Q_1M$	00	01	11	10
00	X_0	X_1	1_3	1_2
01	X_4	X_5	1_7	1_6
11	X_{12}	X_{13}	1_{15}	1_{14}
10	X_8	X_9	1_{11}	1_{10}

$K_1 = 1.$

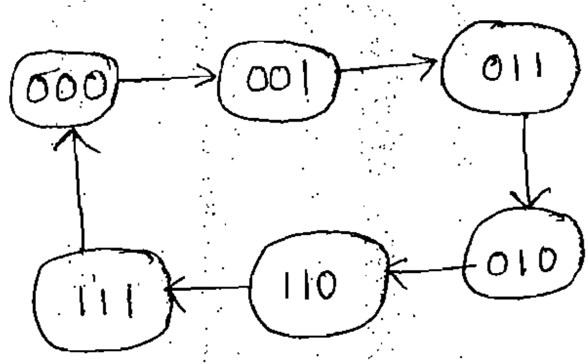
Step 5 :- The logic diagram.



→ Design of a synchronous modulo-6 Gray code counter.

Step 1 :- number of flip flops - 3 FF'S

Step 2 :- state diagram.



Step 3 :-

Present state			next state			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Step 4 :- minimal expressions :-

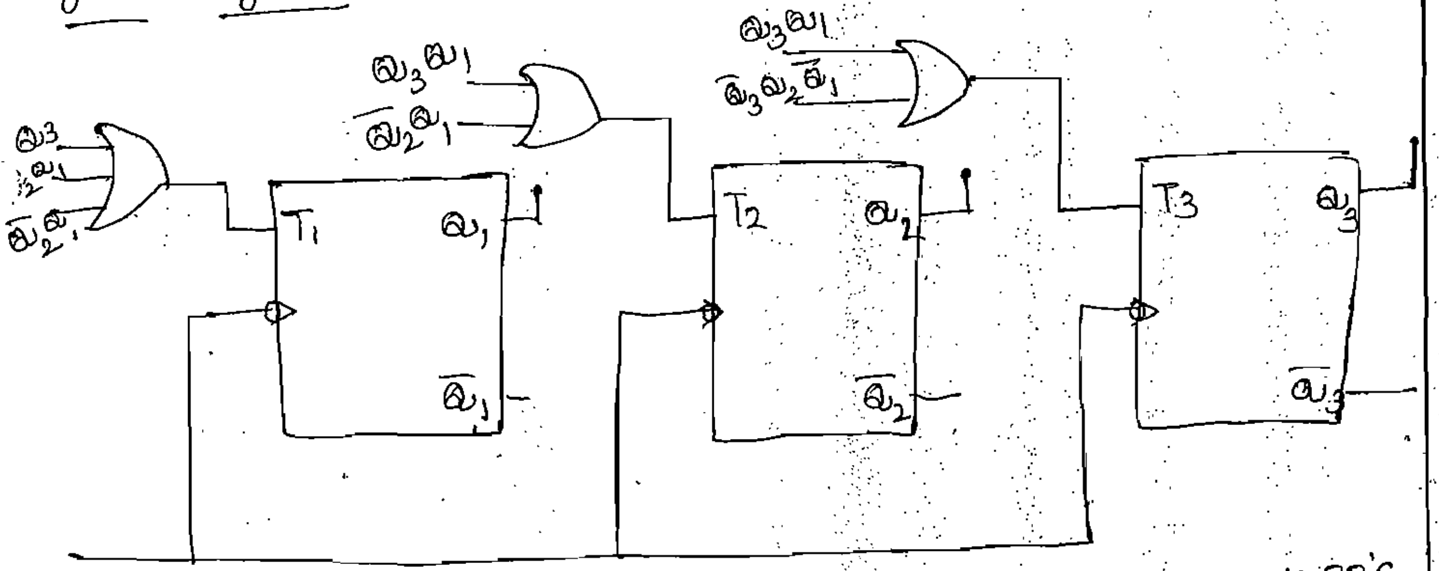
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0				1
1	X	X	1	

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0		1		
1		X	1	

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	1		1	
1	X	X	1	1

$T_3 = Q_3Q_1 + \bar{Q}_3Q_2\bar{Q}_1$ $T_2 = Q_3Q_1 + \bar{Q}_2Q_1$ $T_1 = Q_3 + Q_2Q_1 + \bar{Q}_2\bar{Q}_1$

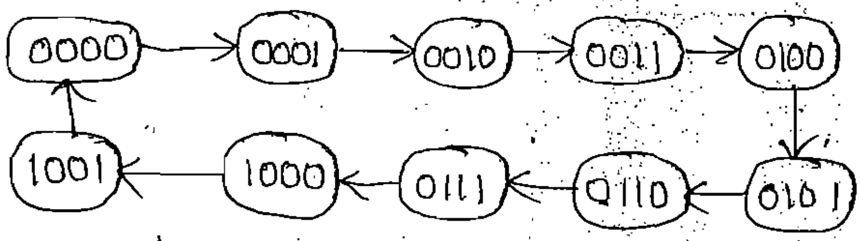
logic diagram :-



→ Design a synchronous (~~mod-9~~) BCD counter using J-K FF'S.

Step 1 :- 4 FF'S

Step 2 :- State diagram.



step 3 :- excitation table.

Present state				next state				J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
Q_4	Q_3	Q_2	Q_1	Q_4	Q_3	Q_2	Q_1								
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1

Step 4 :-

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	0	1	3	2
01	4	5	1 7	6
11	X 12	X 13	X 15	X 14
10	X 8	X 9	X 11	X 10

$J_4 = Q_3Q_2Q_1$

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10		1	X	X

$K_4 = Q_1$

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00			1	
01	X	X	X	X
4	X	X	X	X
10			X	X

$J_3 = Q_2Q_1$

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00		1	X	X
01		1	X	X
11	X	X	X	X
10			X	X

$J_2 = \overline{Q_4}Q_1$

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	X	X	1	
01	X	X	1	
11	X	X	X	X
10	X	X	X	X

$K_2 = Q_1$

$Q_4Q_3 \backslash Q_2Q_1$	00	01	11	10
00	X	X	X	X
01			1	
11	X	X	X	X
10	X	X	X	X

$K_3 = Q_2Q_1$

Step 4 :- The minimal expression.

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X_0	X_1	X_3	X_2
1	4	X_5	1_7	6

$K_3 = Q_1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	1	X
1	X	X		

$K_2 = \overline{Q_3}$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	1	X
1	X	X	X	X

$J_3 = 1$

$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	X	X
1		X	X	1

$J_1 = Q_2$

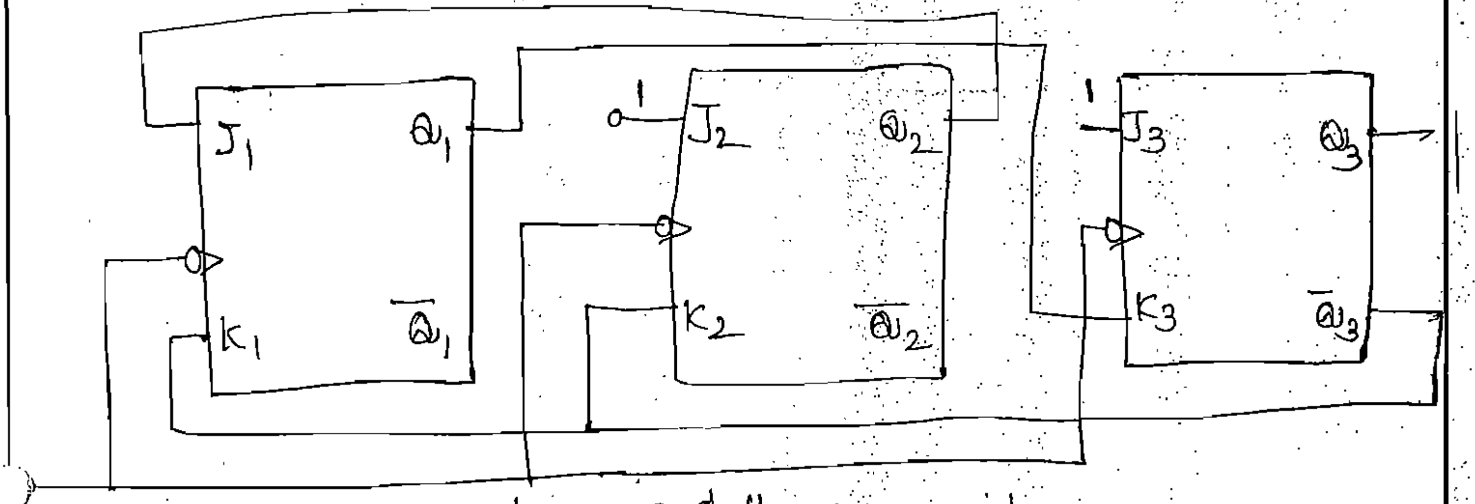
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	1	X
1	X	X		X

$K_1 = \overline{Q_3}$

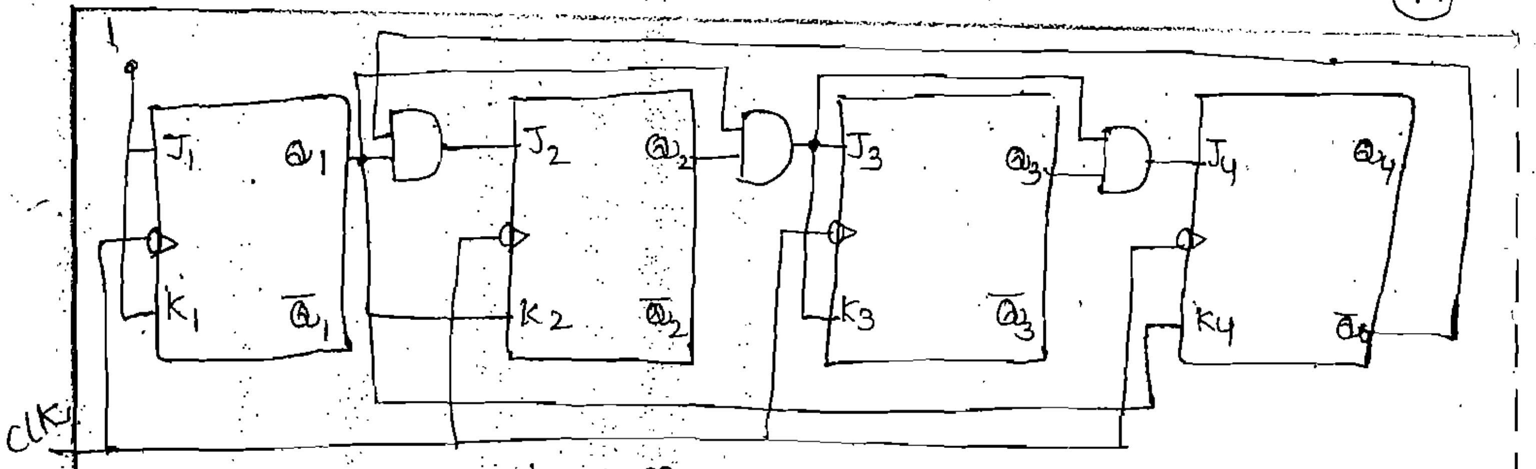
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	X	X	X	X
1	1	X	X	X

$J_2 = 1$

step 5 :- logic diagram



logic diagram of the J-K counter.

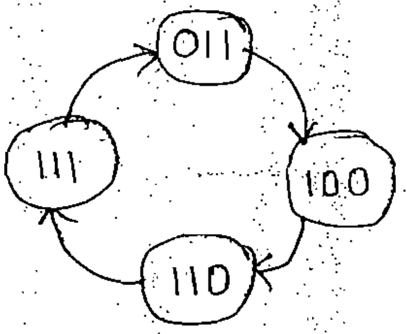


Logic diagram.

→ Design a J-K counter that goes through states 3,4,6,7 and 3 . . . is the counter self-starting (take a remaining states are invalid)

Step1 :- number of flip-flops - 3 flip-flops.

step2 :- State diagram.



State-diagram

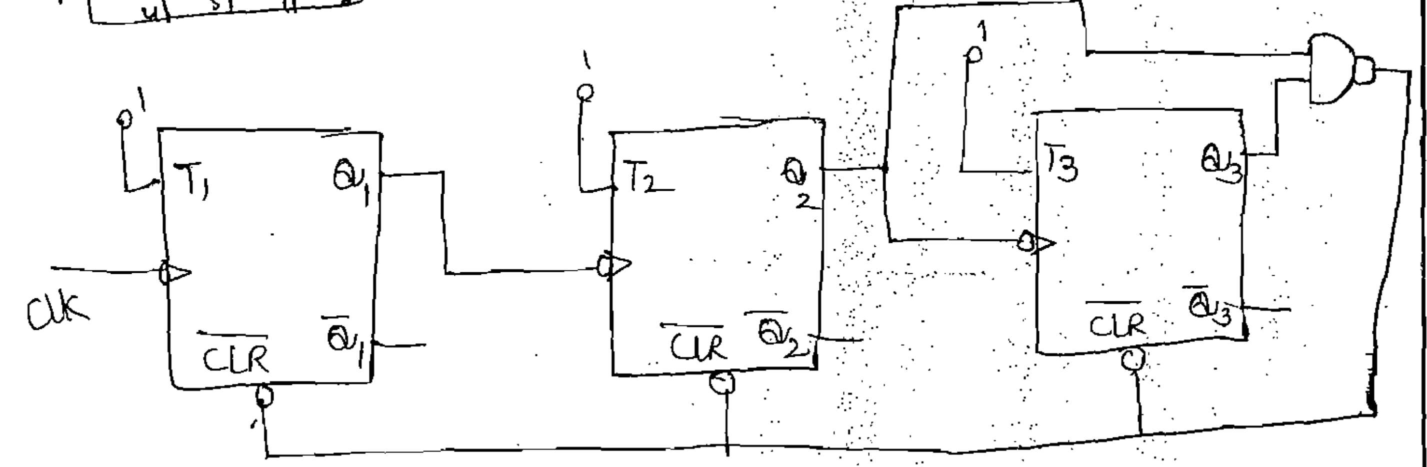
Step 3 :- excitation table

Present state			next state			Required inputs					
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	J_3	K_3	J_2	K_2	J_1	K_1
0	1	1	1	0	0	X	X	X	1	X	1
1	0	0	1	1	0	X	0	1	X	0	X
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Q_2Q_1, R =

Q_3 \ Q_2Q_1	00	01	11	10
0				
1			X	1

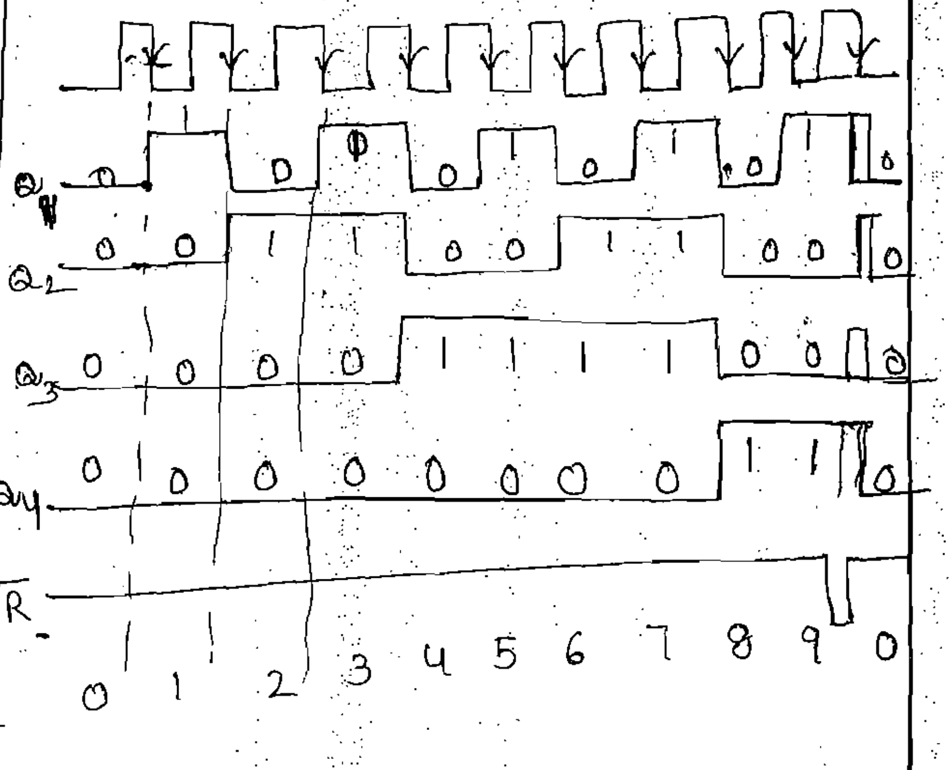
$R = Q_3 Q_2$



logic diagram.

mod-10 Asynchronous counter :-

R	After pulses	Q_4	Q_3	Q_2	Q_1
0	0	0	0	0	0
0	1	0	0	0	1
0	2	0	0	1	0
0	3	0	0	1	1
0	4	0	1	0	0
0	5	0	1	0	1
0	6	0	1	1	0
0	7	0	1	1	1
0	8	1	0	0	0
0	9	1	0	0	1
1	10	0	0	0	0



K-map for R

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00				
01				
11	X	X	X	X
10		X	X	1

R = 0 for 0000 to 1001

R = 1 for 1010

R = X for 1011 to 1111

$R = Q_4 Q_2$

A synchronous counters :-

→ Design a mod-6 Asynchronous counter using T FF's

→ A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. when six pulse is applied, the counter temporarily goes to 110 state but immediately reset to 000.

→ It requires three FF's, because the smallest value of n satisfying the condition $N \leq 2^n$ is $n = 3$. ∴ three FFS can have eight possible states, out of which only six are utilized and remaining two states 110 and 111.

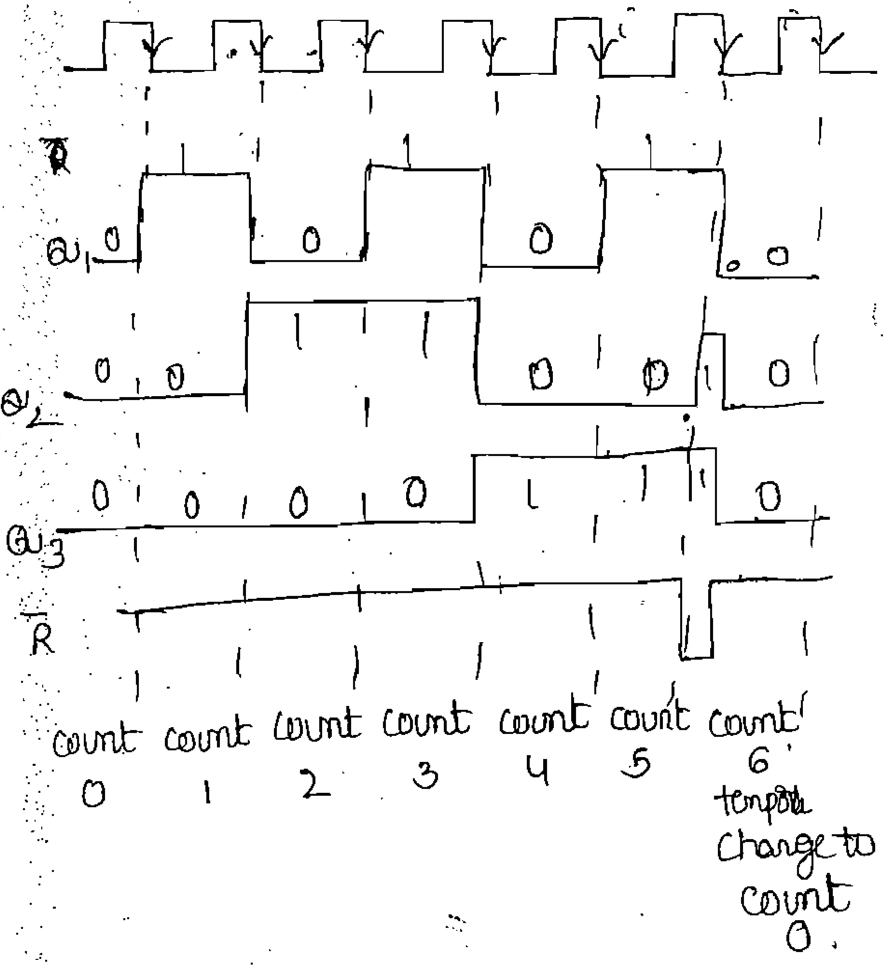
→ For the design, To write a truth table with the present state outputs Q_3, Q_2 and Q_1 as the variables, and Reset R as the output.

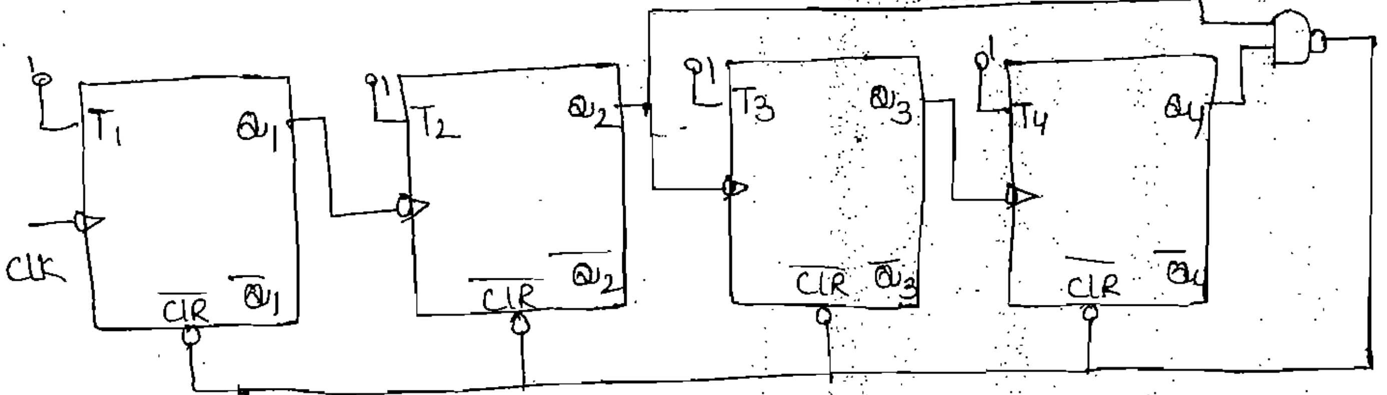
R = 0 for 000 to 101, R = 1 for 110, and R = X for ~~xx~~ 111.

Table for R

After pulses	State Q_3	Q_2	Q_1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1 ↓ 0	1 ↓ 0	0 ↓ 0	1
	1	1	1	X
7	0	0	1	1

Timing diagram.





logic- diagram.

Asynchronous mod - 10 counter using T Flip-flops.

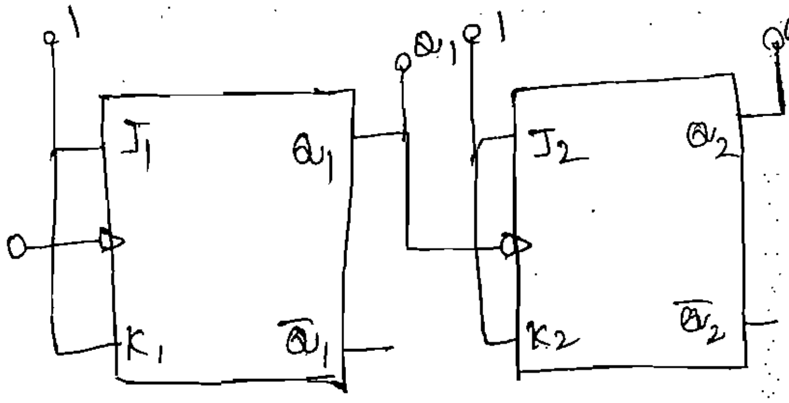
-> Two - bit ripple up- counter using negative edge - triggered flip-flops:-

-> The 2-bit up - counter counts in the order 0 1, 2, 3, 0, 1 . . . i.e 00, 01, 10, 11, 00, 01 . . etc. A 2-bit ripple up-counter, using negative edge - triggered J-K FFs, The counter initially reset to 00 when first clock pulse is applied, FF₁ toggles at the negative-going edge of this pulse, therefore, Q₁ goes from low to high. This becomes a positive - going signal at the clock input of FF2.

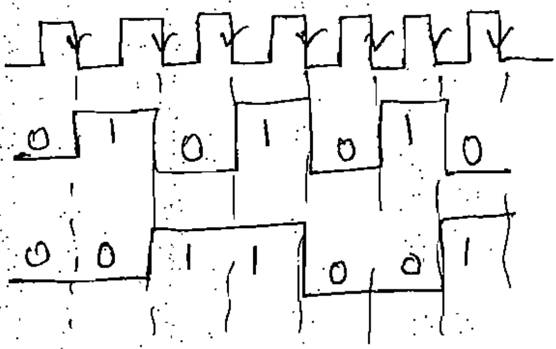
-> So FF₂ is not affected, and hence, the state of the counter after one clock pulse is Q₁ = 1 and Q₂ = 0.

-> So next clock pulse. FF1 is change to 1 to 0. then negative going edge of this pulse FF2 is change to 0 to 1. Q₁ = 0 and Q₂ = 1

-> So next clock pulse FF₁ is change to 0 to 1. then positive going edge of this pulse FF₂ is no change Q₁ = 1, Q₂ = 1.

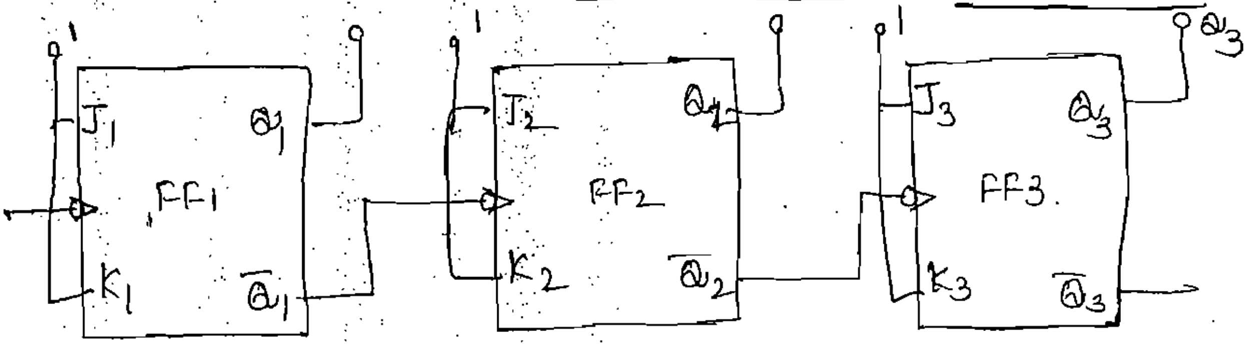


logic, diagram.



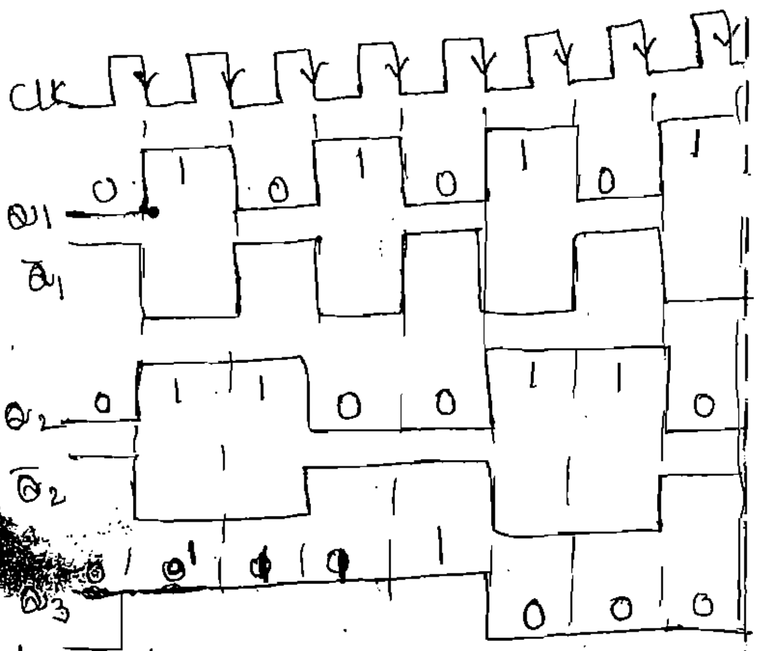
Timing diagram.

3-bit down-counter using negative-edge trigged J-K flip-flops.



logic diagram.

A 3-bit down counter counts in order 0, 7, 6, 5, 4, 3, 2, 1, 0, 1 ... For down counting $\bar{Q}_1$ to FF1 is connected to the clock of FF2. the $\bar{Q}_2$ to FF2 is connected to the clock of FF3. output taken from Q_1, Q_2 and Q_3.



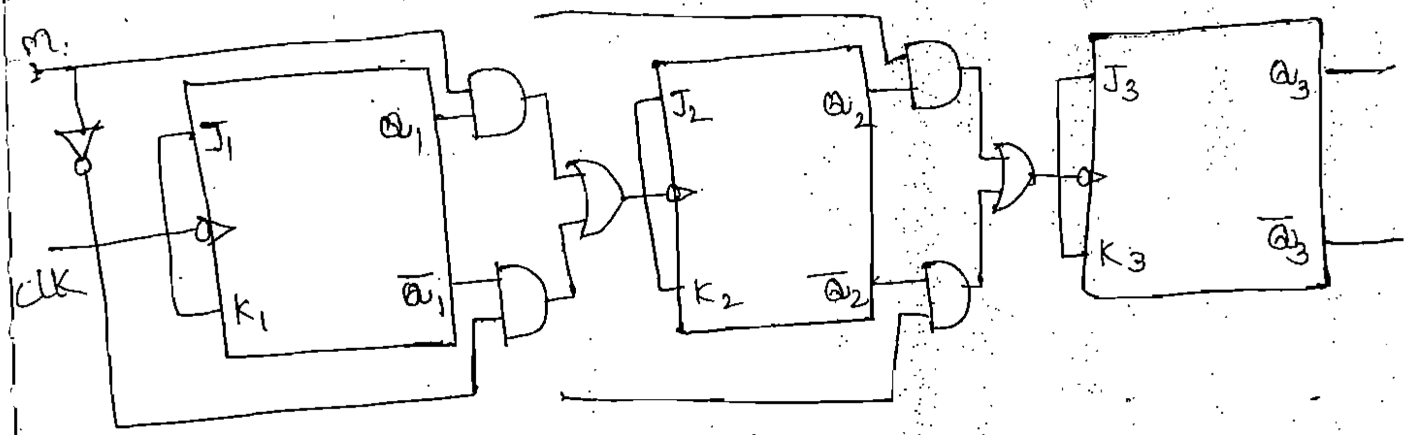
3-bit updown counter using negative edge-triggered flip-flops :-

As the name indicates an up-down counter is a counter which can count both in upward and downward directions An up-down counter is also called a forward / backward counter or a bi-directional counter.

→ so control signal M or mode signal is required to chose the direction of count.

→ when m=1 for up counting, m=0 for down counting for up-counting Q_1 is transmitted to clock FF2 and for down-counting $\bar{Q}_1$ is transmitted to clock FF2.

→ This is achieved by using two AND gates and one OR Grate.



Shift registers :-

→ As a flip-flop can store only one bit of data, a '0' or a '1', it is referred to as a single-bit register. when more bits of data are to be stored, a number of a number of FF's are used.

→ A register is a set of FF's used to store binary data. The storage capacity of a register is the number of bits of digital data it can retain

→ shift-register are a type of logic circuits closely related to counters.

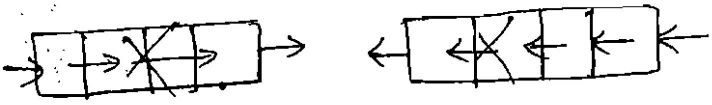
→ They are used basically for the storage and transfer of digital data. The basic difference between a shift register and counter is that a shift register has no specified sequence of states except in certain very specialized applications.

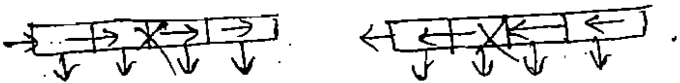
→ where as a counter has a specified sequence of states.

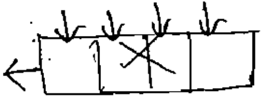
Data - Transmission in shift registers :-

→ A number of flip-flop's connected together such that data may be shifted into and shifted out of them is called a shift-register.

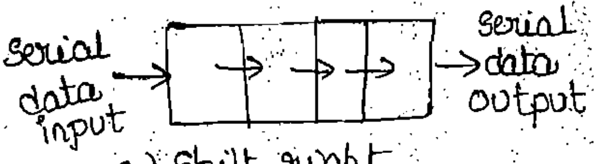
→ Data may be shifted into or out of the register either in serial form or in parallel form. So, there are four types of shift-registers. They are

1. Serial - in, Serial - out 

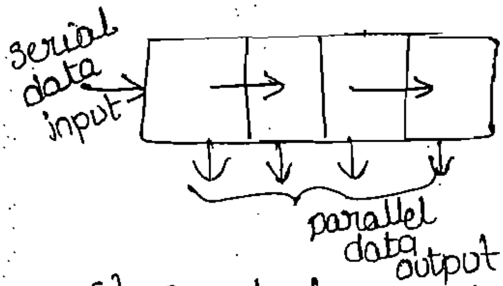
2. Serial - in, parallel - out 

3. parallel - in, serial - out 

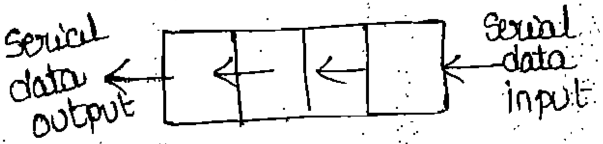
4. parallel - in, parallel - out



a) Shift-right

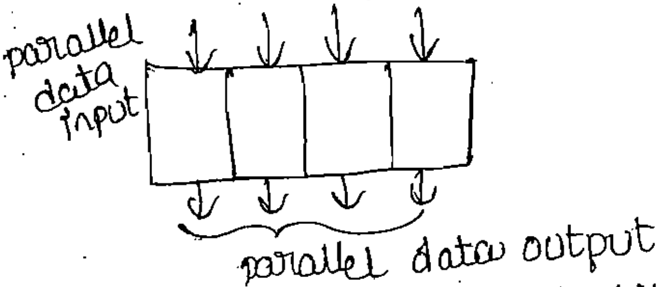


c) Serial - in, parallel out shift regi

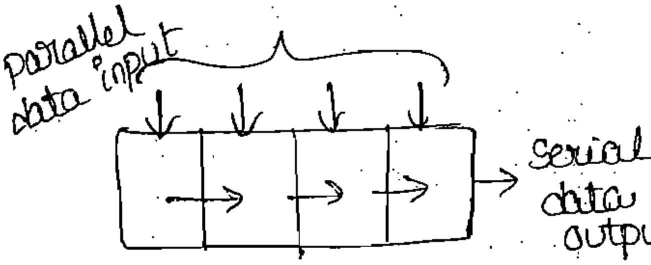


b) Shift - left.

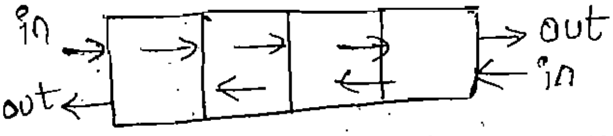
Serial - in, serial - out



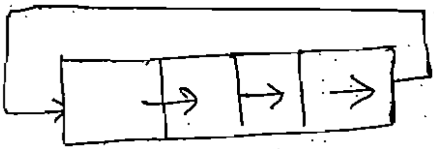
d) parallel - in, parallel - out, shift regis.



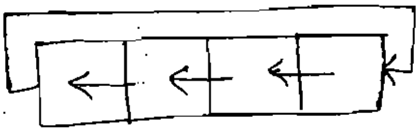
e) parallel - in, serial output.



f) bi-directional shift register.

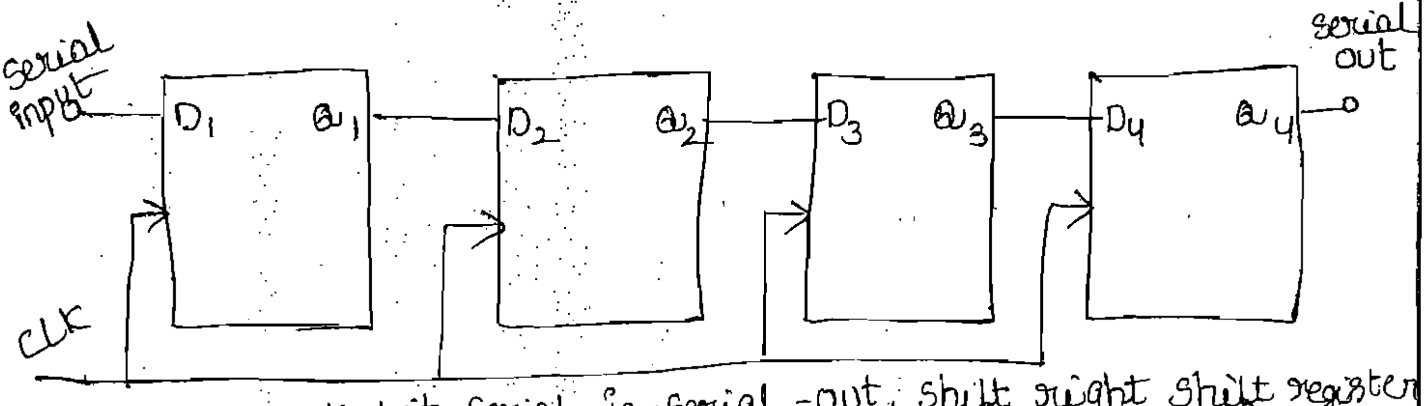


g) Rotate - right shift register

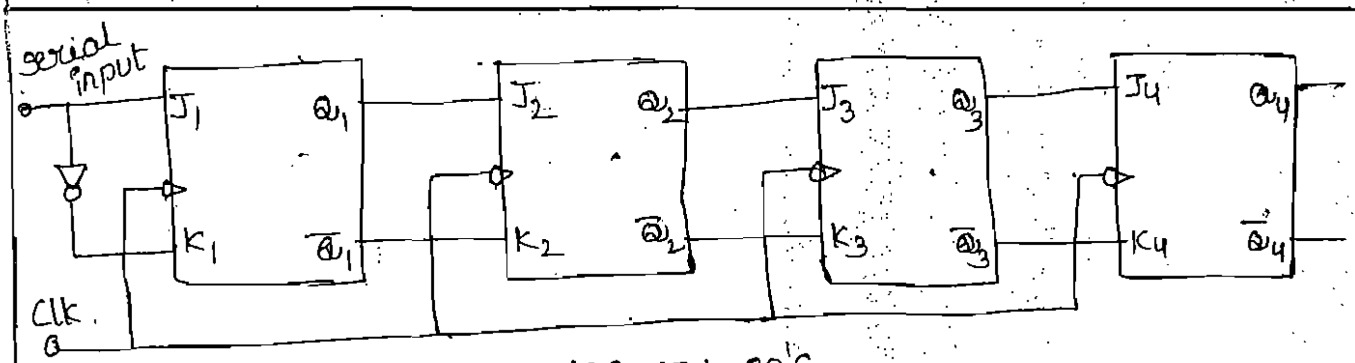


h) Rotate - left shift register.

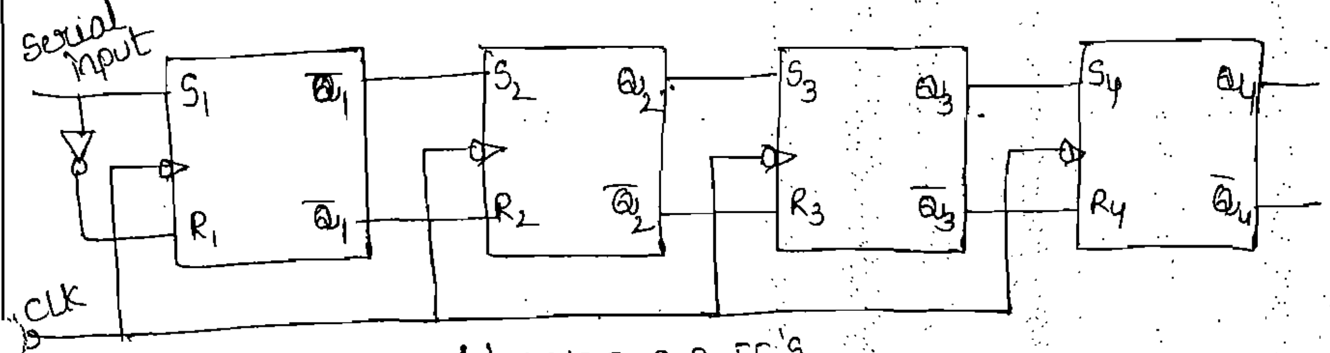
→ Serial - in, serial - out shift register :-



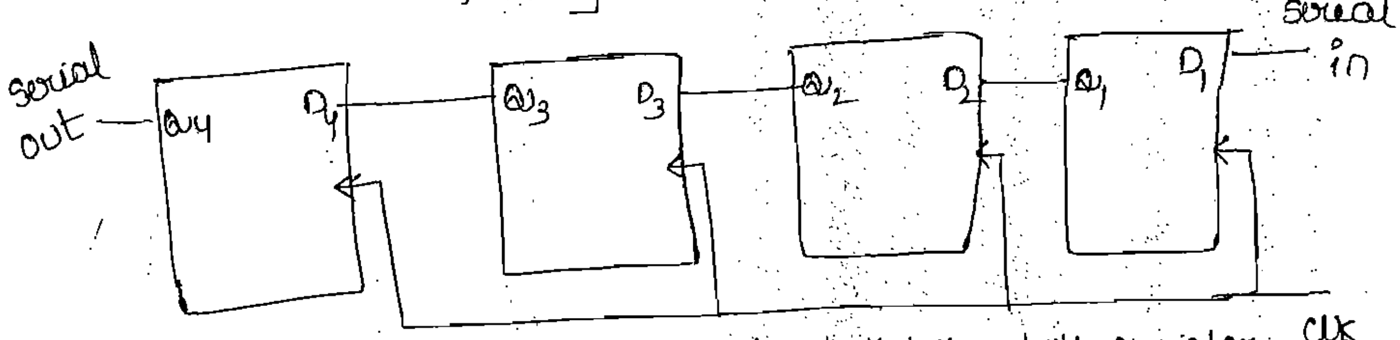
4-bit Serial - in, Serial - out, shift right shift register



a) using J-K FF'S.

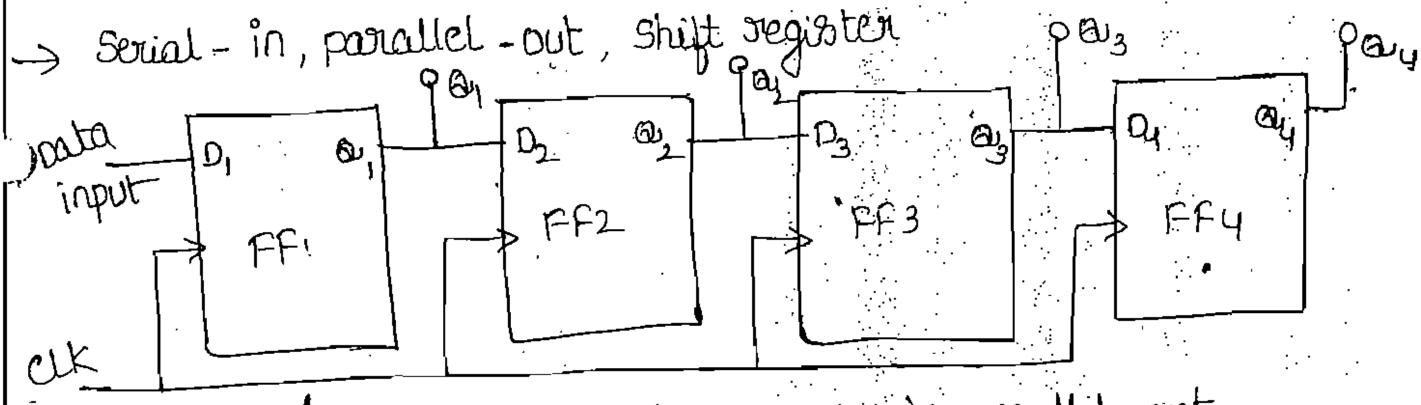


b) using S-R FF'S

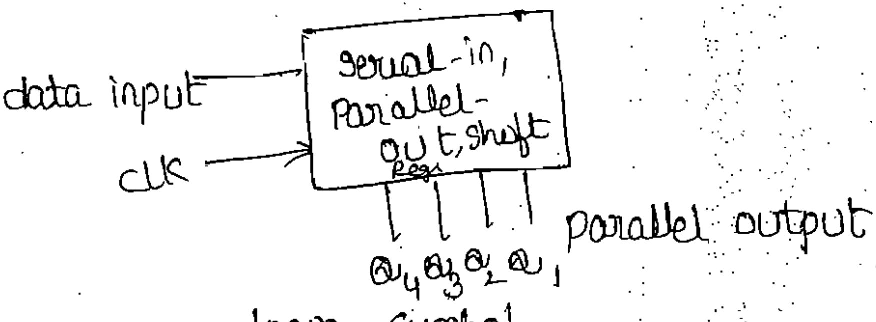


A 4-bit serial-in, serial-out, shift left shift register.

→ Serial-in, parallel-out, shift register

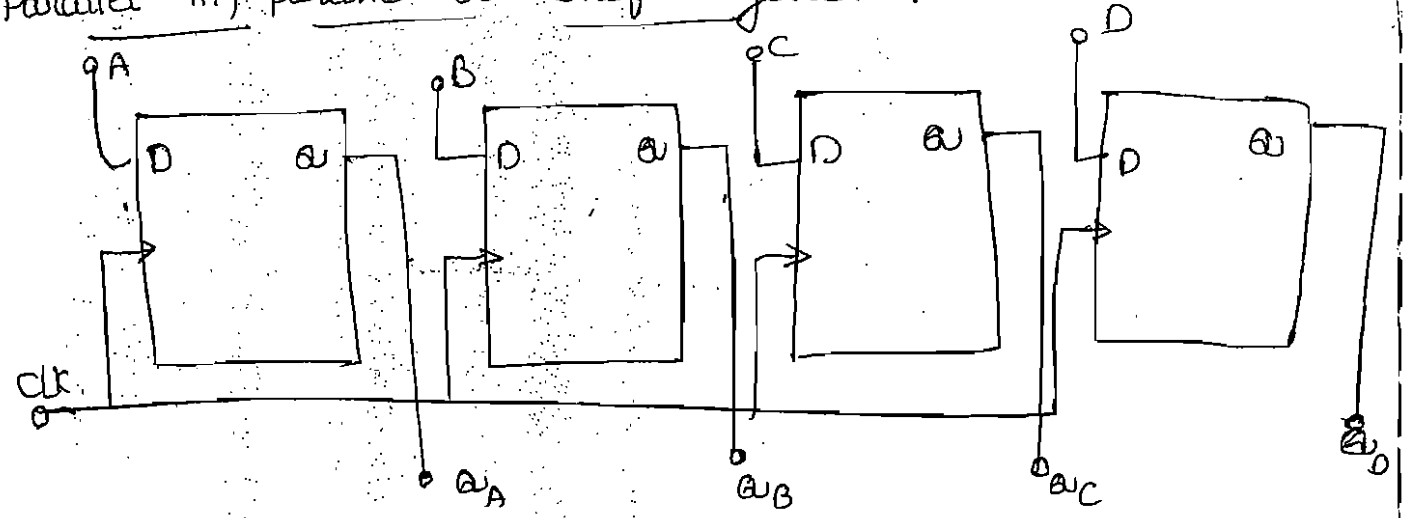


logic diagram for serial-in, parallel-out



logic symbol.

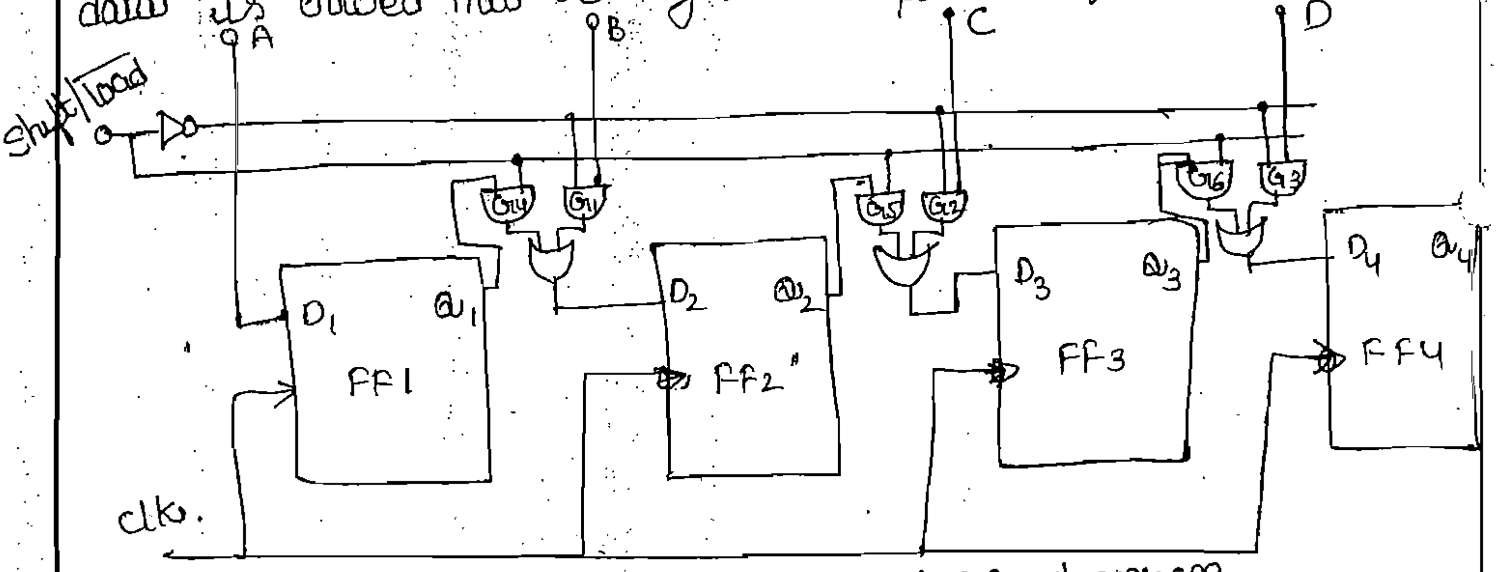
→ Parallel -in, parallel - out, Shift - register :-



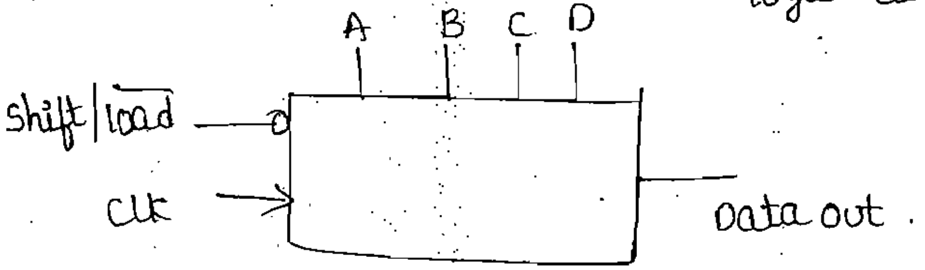
logic diagram of a 4-bit parallel- in, parallel -out

→ Parallel - in, Serial - out shift - register :-

For a parallel -in, serial- out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by- bit basis over a single line. a 4-bit parallel -in, serial-out, shift register using D-FFS. There are four data lines A, B, C and D through which the data is entered into the register in parallel form.



logic diagram



logic symbol

The signal ~~a~~ shift | $\overline{LOAD}$ allows (a) the data to be entered into the register in parallel form. (b) the data to be shifted out serially from terminal Q_4.

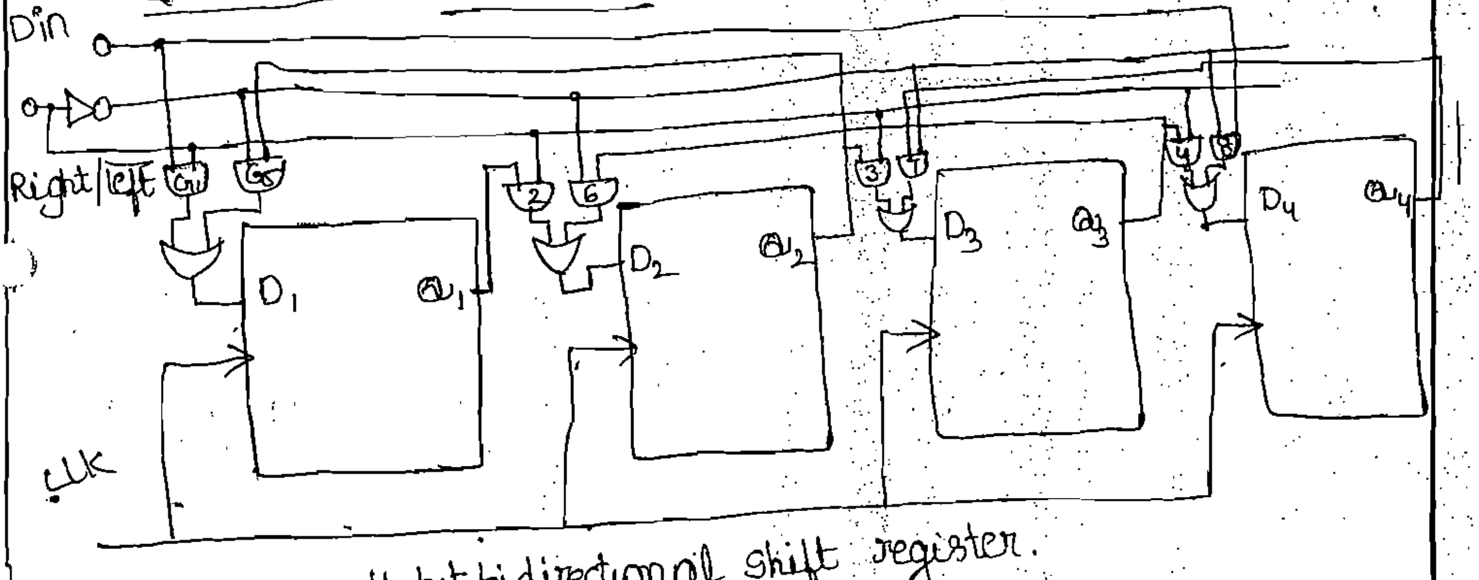
→ when Shift | $\overline{Load}$ line is high, gates G_1, G_2 and G_3 are disabled, but G_4, G_5, G_6 are enabled allowing the data bits to shift-right from one-stage to the next.

→ when Shift | $\overline{Load}$ line is low, gates G_4, G_5 and G_6 are disabled, where as gates G_1, G_2 and G_3 are enabled allowing the data input to appear at the D inputs of the respective FFs.

→ when clock pulse is applied, these data bits are shifted to the Q output terminals of the FFS and, therefore, data is inputted in one step.

→ The OR Gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on the shift | $\overline{Load}$ input.

→ Bi-directional shift register :-



4-bit bi-directional shift register.

→ A bi-directional shift register is one in which the data bits can be shifted from left to Right or from Right to left.

→ in above figure 4-bit serial-in, serial-out, bidirectional (shift left, shift right) shift reighter.

-> whent Right | $\overline{\text{left}}$ is a 1, the logic circuit works as a shift-right shift register.

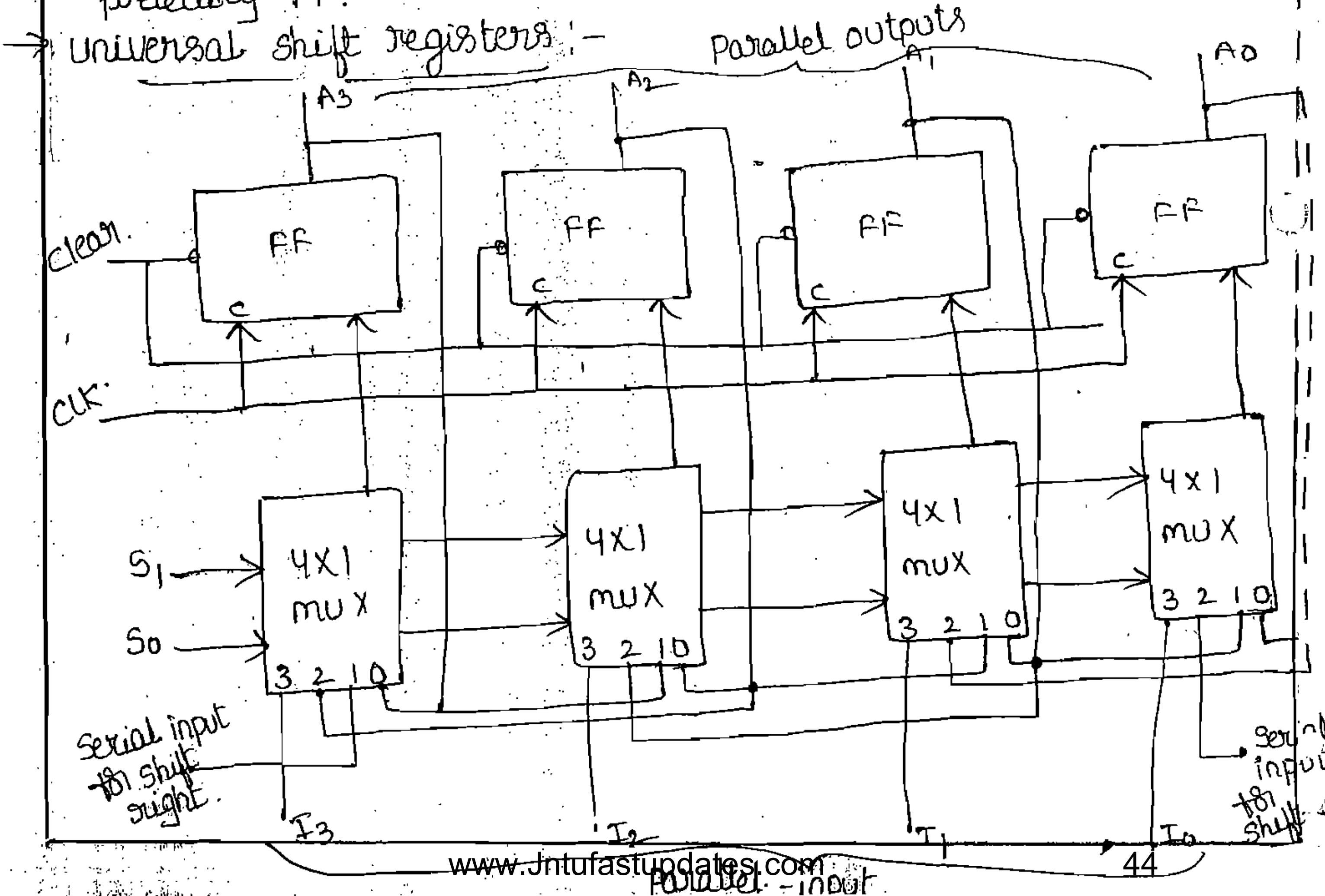
-> when Right | $\overline{\text{left}}$ is a 0, the logic circuit works as a shift-left shift register.

-> The bi-directional operation is achieved by using the mode signal and two AND gates and one OR Gate for each stage.

-> whent mode signal Right | $\overline{\text{left}}$ is a '1' the AND Grates. G_1, G_2, G_3 and G_4 ~~and~~ are enabled and disables the AND Grates G_5, G_6, G_7 and G_8 and the state of a output of each FF is passed through the gate to the D input of the following FF.

-> when mode signal Right | $\overline{\text{left}}$ is a '0' the AND Gates. G_1, G_2, G_3 and G_4 are disabled, and enabled the AND gates G_5, G_6, G_7 and G_8 and the state of a output of each FF is passed through the gate to the D input of the preceding FF.

-> universal shift registers :-



→ A register capable of shifting in one direction only is a uni-directional shift register. one that can shift in both directional is a bi-directional shift register.

→ If the register has both shifts and parallel load capabilities, it is referred to as a universal shift register.

→ A universal shift-register has both shifts ~~as~~ it means whose input can be either in serial form or in parallel form and whose output also can be either in serial form or in parallel form.

→ A universal shift register can be realized using multiplexers. it consists of four D-flip flops and four multiplexers.

→ The four multiplexers have two common selection inputs S_1 and S_0. Input 0 in each multiplexer is selected when $S_1S_0=00$ Input 1 is selected when $S_1S_0=01$, and input 2 is selected when $S_1S_0=10$ and input 3 is selected when $S_1S_0=11$.

mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	parallel load.

→ when $S_1S_0=0$, the present value of the register is applied to the D-input of flip-flops. This condition forms a path from the output of each flip-flop into the input of the same flip-flop.

→ when $S_1S_0=01$, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop A4.

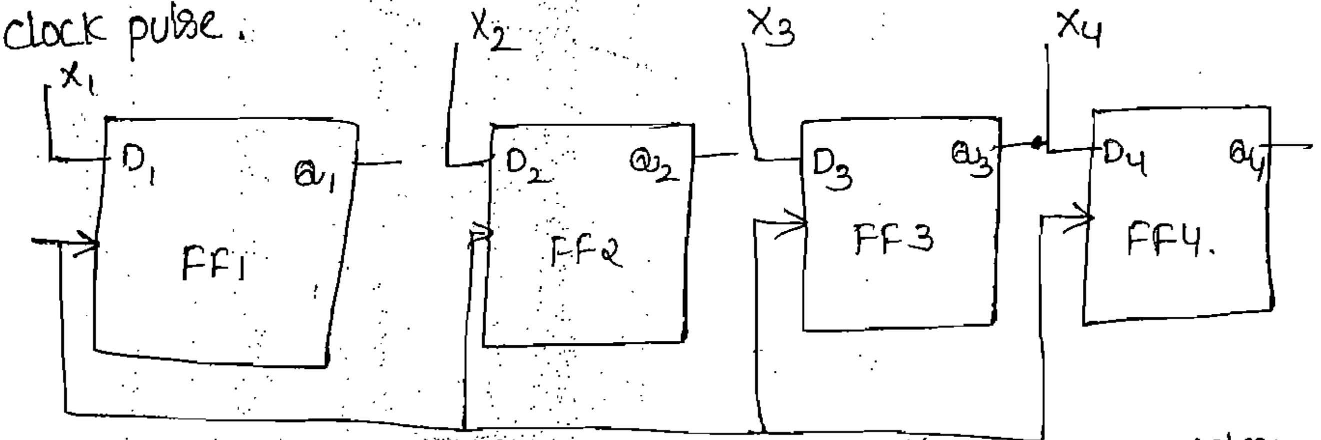
→ when $S_1S_0=10$, a shift-left operation results with the other serial input going into flip-flop A1.

→ Finally $S_1S_0=11$ the binary information on the parallel input lines is transferred into the register simultaneously during the next clock edge.

Buffer Register :-

Some registers do nothing more than storing a binary word. The buffer register is the simplest of registers. It simply stores the binary word. The buffer may be a controlled buffer. most of the buffer registers use D-flip flops.

The binary word to be stored is applied to the data terminals. on the application of clock pulse, the output word becomes the same as the word applied at the input terminals the input word is loaded into the register by the application of clock pulse.



logic diagram of a 4-bit buffer register.

$Q_4 Q_3 Q_2 Q_1 = X_4 X_3 X_2 X_1$

$Q = X$

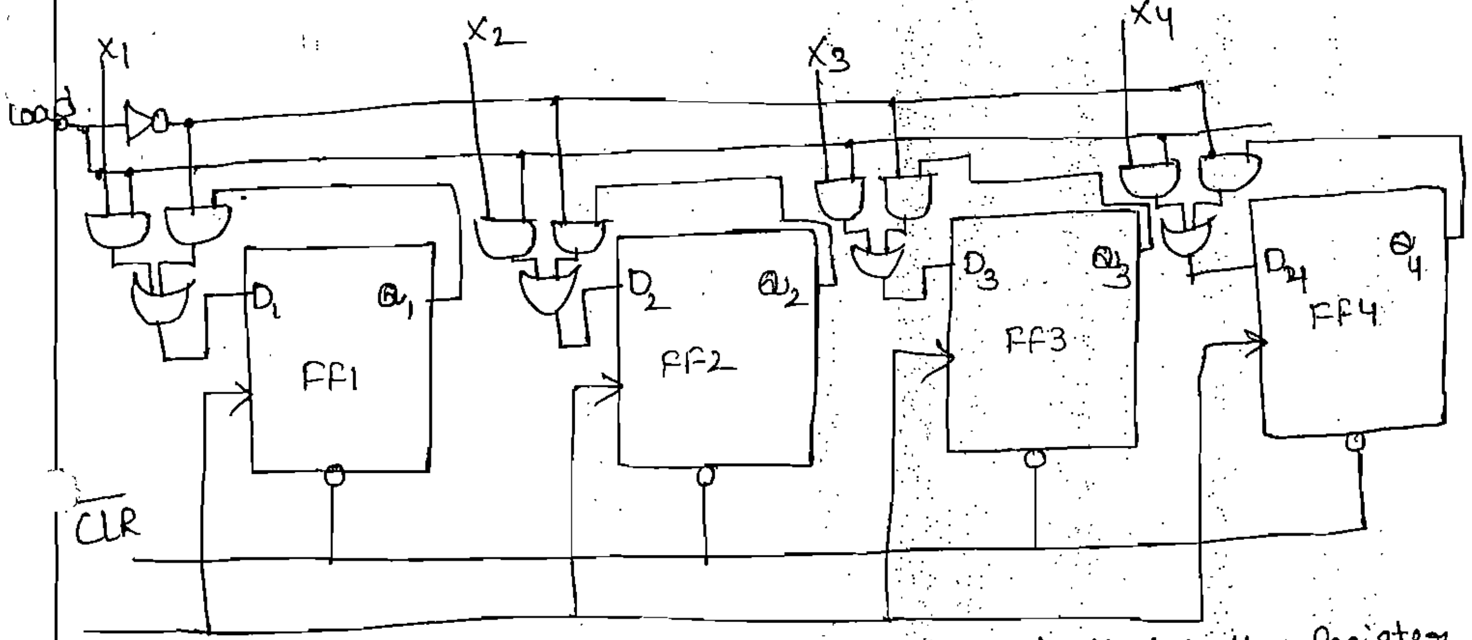
Controlled buffer Register :-

In Buffer register is too primitive to be of any use. it needs some control over the x bits, that is some way of holding them off until we are ready to store them.

→ If $\overline{CLR}$ goes low, all the FFs are RESET and the output becomes, $Q = 0000$.

→ when $\overline{CLR}$ goes high, the register is ready for action.

Load is the control input. when Load is high, the data bits X can reach the D inputs of FF'S. At the positive - going edge of the next clock pulse, the register is loaded.



logic diagram of a 4-bit controlled buffer Register.

when Load is low, the X bits cannot reach the FF'S. At the same time, the inverted signal $\overline{Load}$ is high. This forces each flip- flop output to feed back to its data input. Therefore, data is circulated or retained as each clock pulse arrives. In other words, the contents of the register remain unchanged in spite of the clock pulses.

→ longer buffer registeris can be built by adding more FFS.

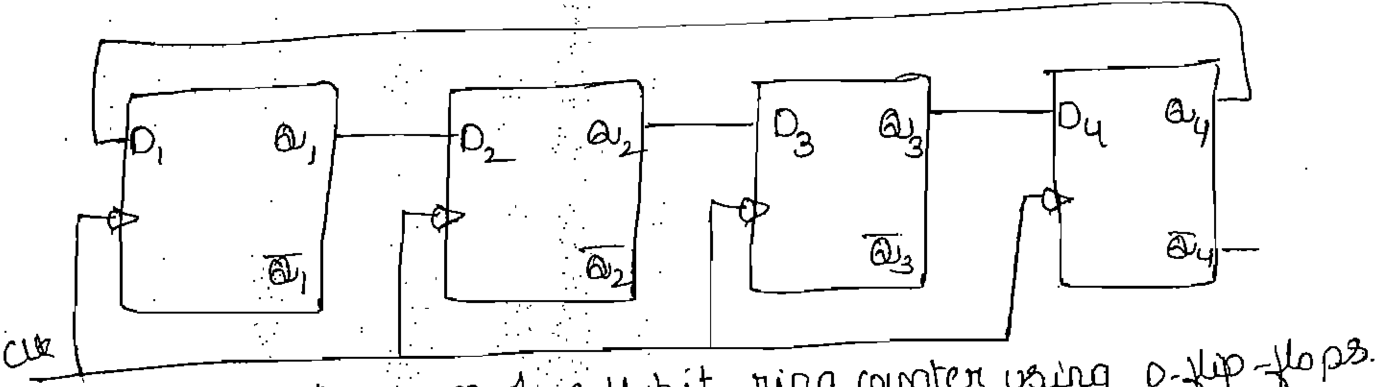
→ Shifto Register counters :-

Shift register counters are obtained from serial-in Serial-out Shift-registers by providing feedback from the output of the last FF to the input of the first FF. These devices are called counters because they exhibit a specified sequence of states. The most widely used shift register

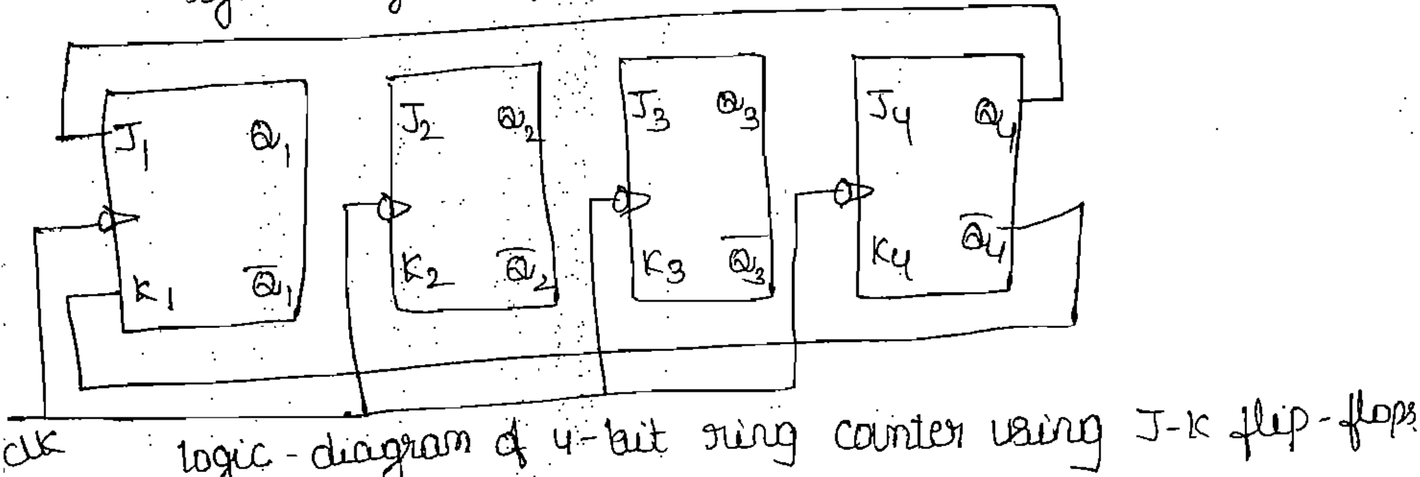
counter is the ring counter [simple ring counter)
twisted ring counter [Johnson counter or the switch-tail

Ring counter :-

This is the simplest shift register counter. The basic ring counter using D-FFS. The realization of this counter using J-K FFs is shown below figure. Its flip-flops are arranged as in a normal shift register, that is the Q output of each state is connected to the D input of the next stage, but the Q output of the last FF is connected back to the D-input of the first FF such that the array of FFS is arranged in ring and, therefore, the name ring counter.



logic diagram of a 4-bit ring counter using D-flip-flops.

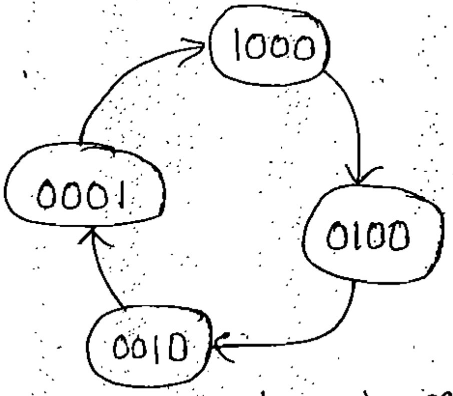


logic-diagram of 4-bit ring counter using J-K flip-flops

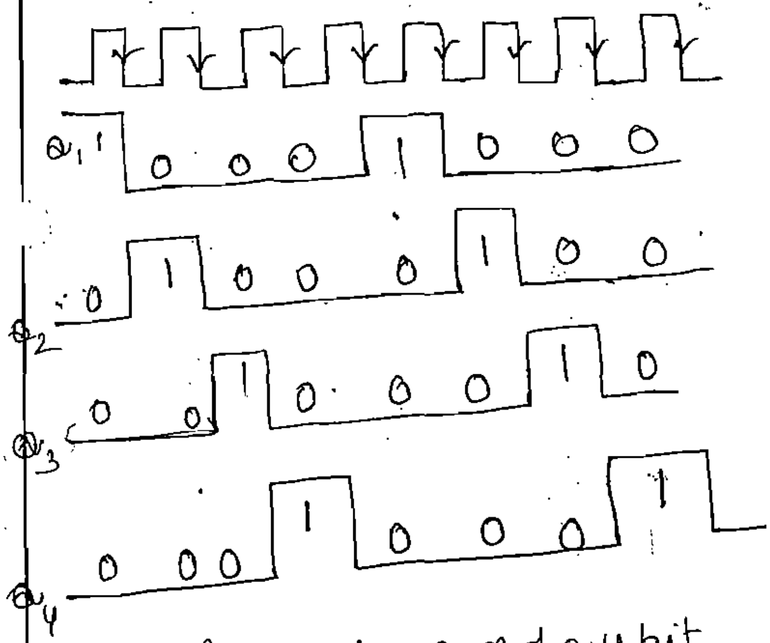
In most instances, only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially, the first FF is present to a 1. so, the initial state is 1000, that is $Q_1=1$, $Q_2=0$, $Q_3=0$

and $Q_4=0$. After each clock pulse, the contents of the register are shifted to the right by one bit and Q_4 is shifted back to Q_1. The sequence repeats after four clock pulses. The number of distinct states in the ring counter.

~~Twisted Ring counter~~ :-



State - diagram.



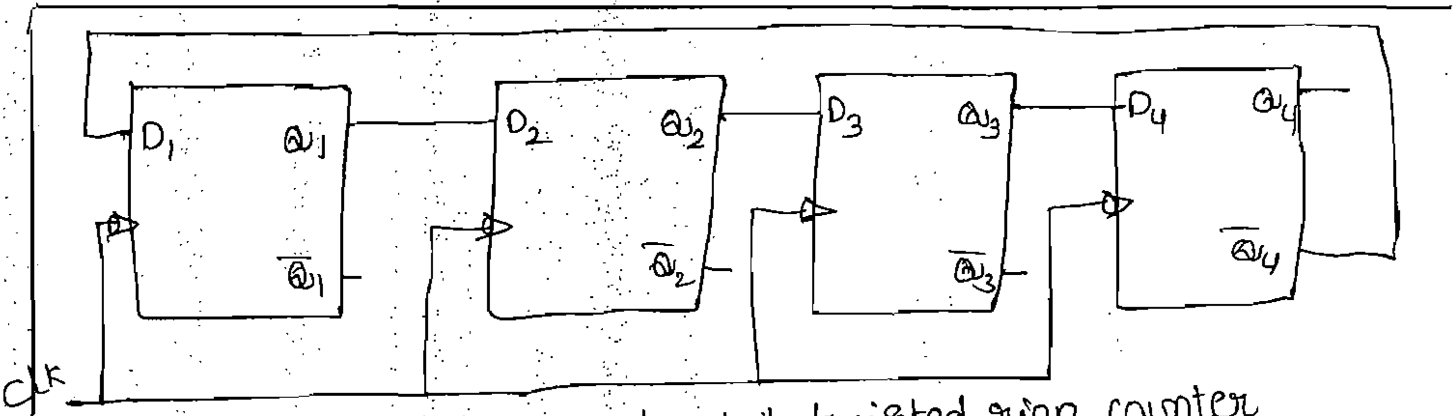
Timing diagram of a 4-bit ring counter.

Q_1	Q_2	Q_3	Q_4	After clk pul
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7.

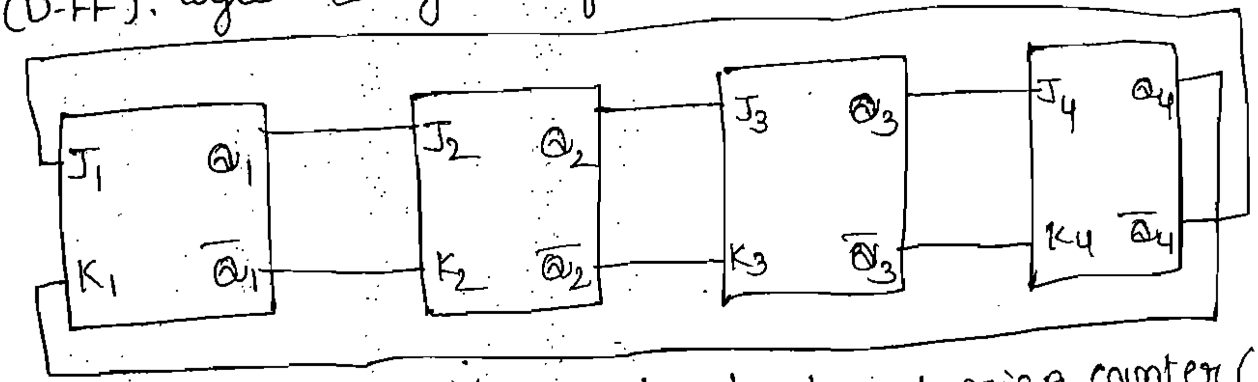
Sequence table.

Twisted Ring counter :-

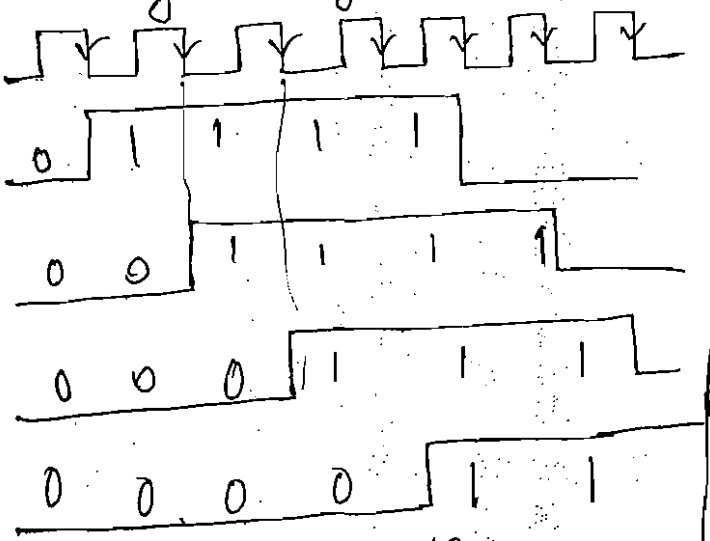
The counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. The Q output of each stage is connected to the D input of the next stage, but the $\bar{Q}$ output of the last stage is connected to the D input of first stage, therefore, the name twisted ring counter.



(D-FF). logic-diagram of 4-bit twisted ring counter



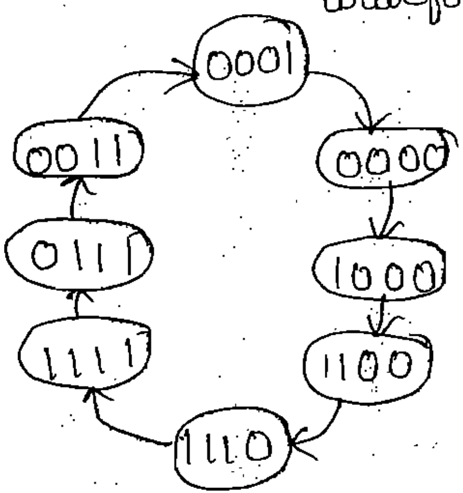
logic-diagram of 4-bit bi-directional ring counter (J-K FF)



waveforms

Q_1	Q_2	Q_3	Q_4	After clk pulse
0	0	0	0	0
Φ	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9.

sequence table.



state diagram

'et initally all the FFS be reset, that is the state of the counter be 0000. After each clock pulse, the level of Q_1 is shifted to Q_2, the level of Q_2 to Q_3, Q_3 to Q_4 and the level of $\overline{Q_4}$ to Q_1 and the sequence is repeated after every eight clock pulses.

→ An n FF Johnson counter can have 2n unique states and can count up to 2n pulses. So, it is a mod-2n counter.

Applications of flip flop :-

1. parallel data storage
2. serial data storage
3. Transfer of data.
4. serial - to - parallel conversion
5. parallel - to - serial conversion
6. counting
7. frequency division.

Applications of shift register :-

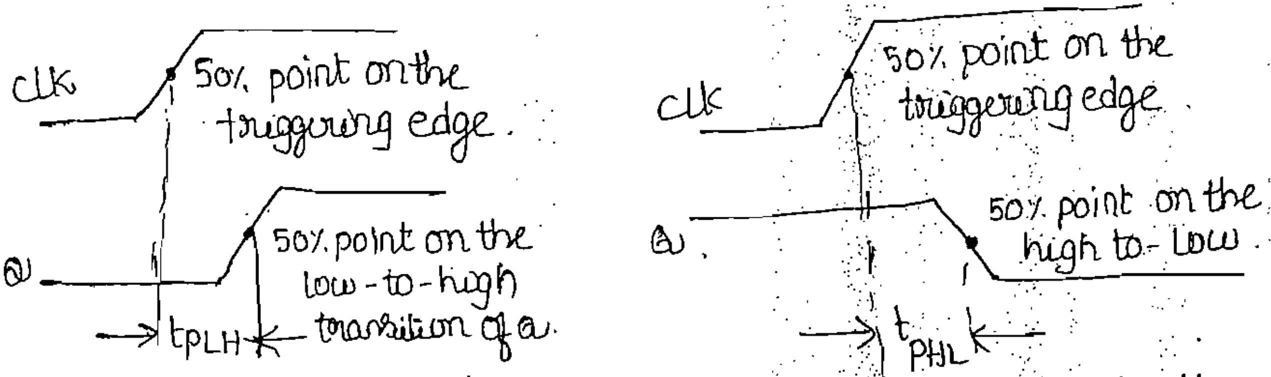
1. Time delays
2. serial / parallel data conversion
3. Ring counters
4. universal Asynchronous receiver transmitter.

Flip-flop operating characteristics :-

propagation delay time :- The output of a flip-flop will not change state immediately after the application of the clock signal or a synchronous inputs. The time interval between the time of application of the triggering edge or Asynchronous inputs and the time at which the output actually makes a transition is called the "propagation delay" time of the flip-flop. It is usually in the range of a few ns to 1µs.

→ propagation delay t_{PLH} measured from the triggering of the clock pulse to the low-to-high transition of the output.

→ propagation delay t_{PHL} measured from the triggering of the clock pulse to the high-to-low transistion of the output.

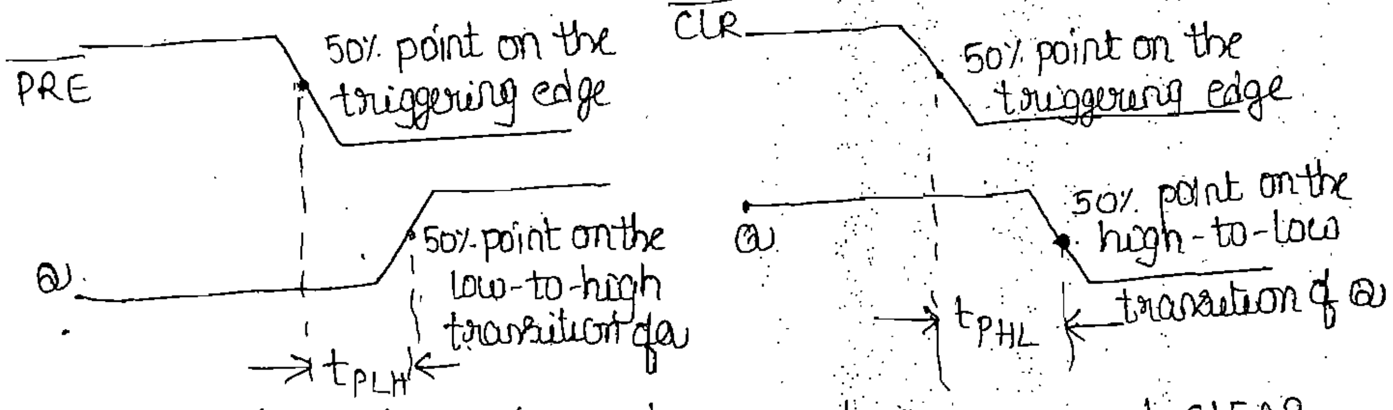


CLK
50% point on the triggering edge.
Q
50% point on the low-to-high transition of Q.
t_{PLH}
CLK
50% point on the triggering edge
Q
50% point on the high to-Low
t_{PHL}

propagation delays t_{PLH} and t_{PHL} w.r.t clk.

→ Propagation delay t_{PLH} measured from the PRESET input to the low-to-high transition of the output

→ propagation delay t_{PHL} measured from the CLEAR input to the high-to-low transition of the output



PRE
50% point on the triggering edge
Q
50% point on the low-to-high transition of Q
t_{PLH}
$\overline{CLR}$
50% point on the triggering edge
Q
50% point on the high-to-low transition of Q
t_{PHL}

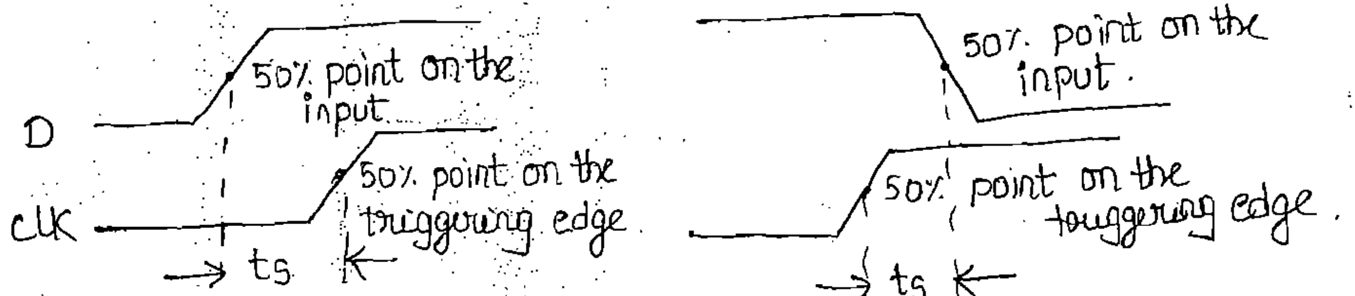
propagation delays t_{PLH}, t_{PHL} w.r.t PRESET and CLEAR

Set-up-time :- The set-up-time (ts) is the minimum time for which the control levels need to the maintained constant on the input terminals of the flip-flop, prior to the arrival of the triggering edge of the clock pulse

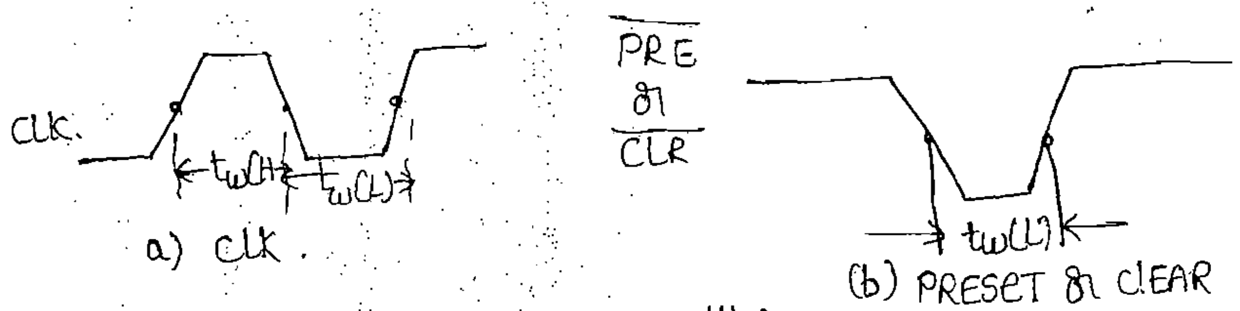
hold time :- The hold time (t_h) is the minimum time for which the control signals need to be maintained constant at the input terminals of the flip-flop, after the arrival of the triggering edge of the clock pulse.

maximum clock frequency :- The maximum clock frequency (fmax) is the highest frequency at which a flip flop can be reliably triggered. If the clock frequency is above this maximum, the flip-flop would be unable to respond quickely enough and its operation will be unreliable. The fmax limit will vary from one flip flop to another.

Pulse widths :- The manufacturer usually specifies the minimum pulse widths for the clock and asynchronous inputs. For the clock signal, the minimum High time tw(H) and the minimum Low time tw(L) are specified and for asynchronous inputs.


D
CLK
50% point on the input
50% point on the triggering edge
ts
50% point on the input.
50% point on the triggering edge.
ts

Set-up time and hold time for a D-flip-flop.



a) CLK.

(b) PRESET or CLEAR

minimum pulse widths.

Clock transition times:- For reliable triggering, the clock waveform transition times (rise and fall times) should be kept very short. If the clock signal takes too long to make the transitions from one level to other, the flip-flop may either trigger erratically or not trigger at all.

power dissipation:- The power dissipation of a flip-flop is the total power consumption of the device. It is

$$P = V_{CC} \cdot I_{CC}$$

V_{CC} = supply voltage
I_{CC} = current

The power dissipation of a flip-flop is usually in mw.

If a digital system has N-flip flops and if each flip-flop dissipates P mw of power, the total power requirements.

$$P_{TOT} = N \cdot V_{CC} \cdot I_{CC}$$
$$= (N \cdot P)\ mw$$