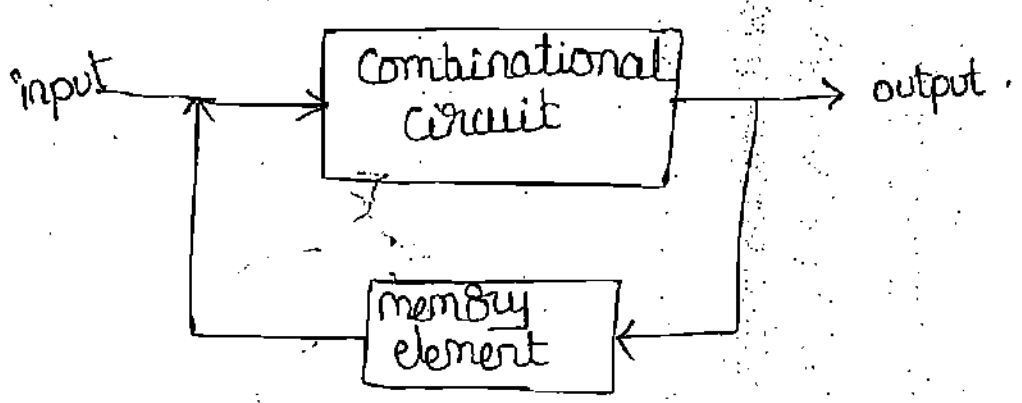
Sequential circuits I.

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-flop is an one-bit

memory device and it can store either 1 & 0.

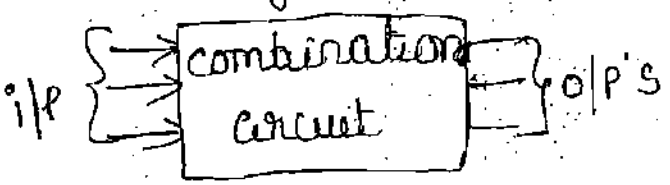
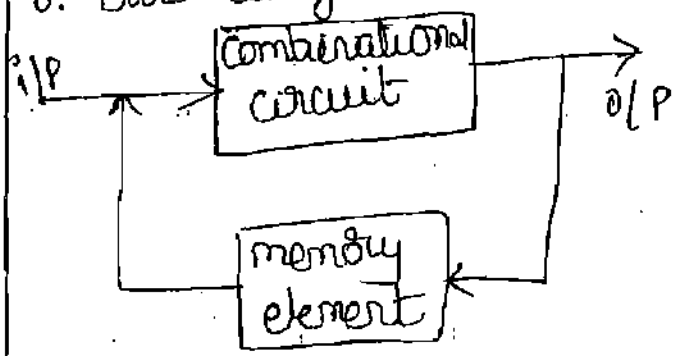
→ latch is a sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

→ 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.

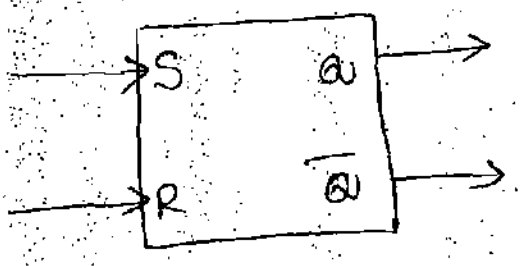
Latch	Flip-flop.
1. A latch is an electronic sequential logic circuit used to store information in an asynchronous arrangement.	1. A flip flop is an electronic sequential logic circuit used to store information in an asynchronous arrangement.
2. one latch can store one bit information, but output state 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.



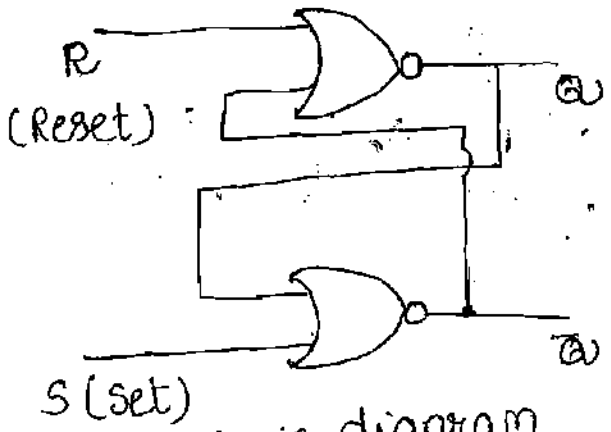
Logic symbol of S-R latch

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.

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

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.

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

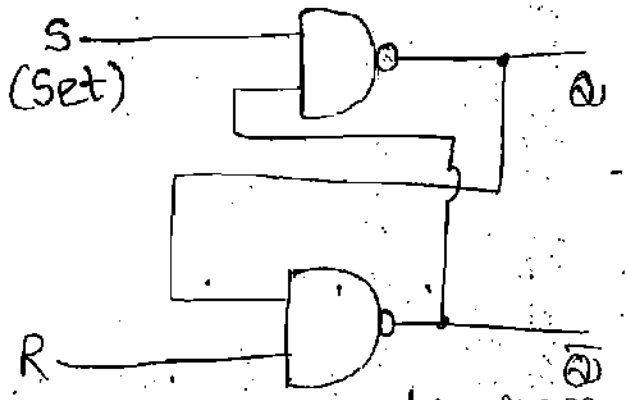
2. SET = 0, RESET = 1, This will always reset Q = 0, where it will remain even after RESET returns to 0.

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 $Q = \bar{Q} = 0$. if the inpu are returned to zero simultaneously, the resulting output state is erratic and unpredictable. This input condition should 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	

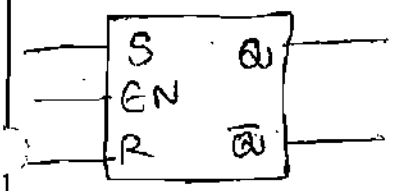
Truth table.

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

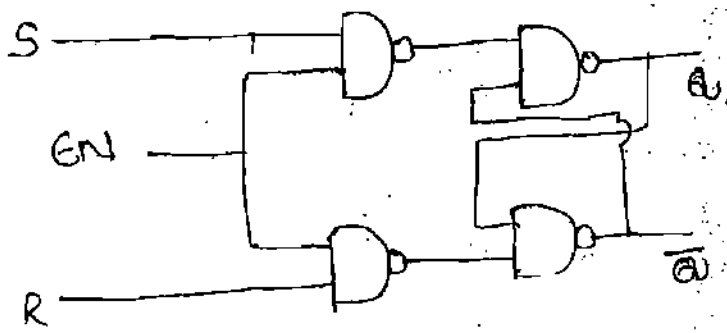
← Simplified truth table.

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.

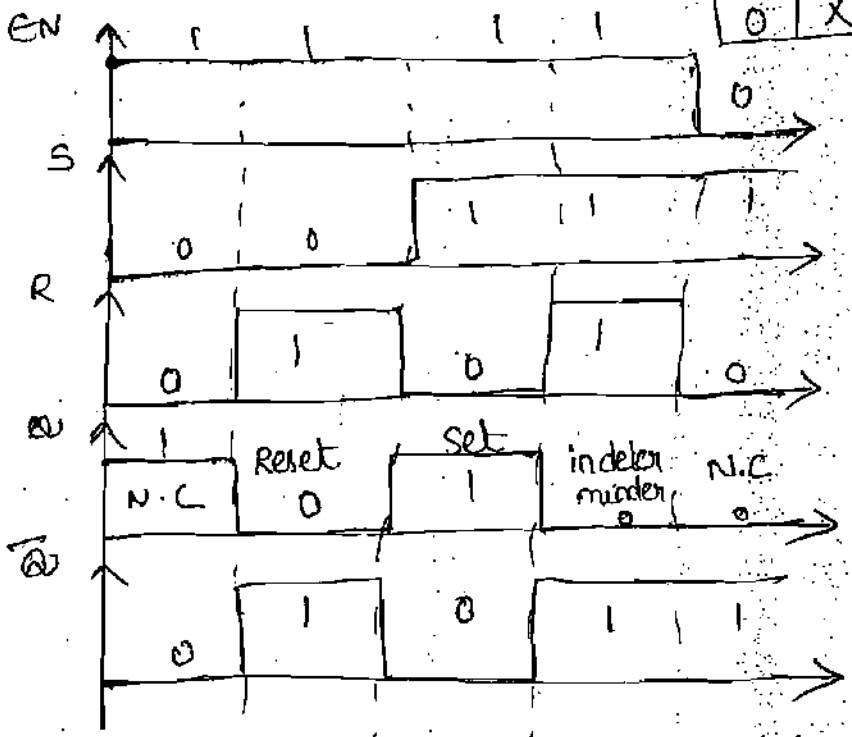


logic symbol



logic diagram.

EN	S	R	Q_n	Q_{n+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	



waveforms for Gated S-R latch.

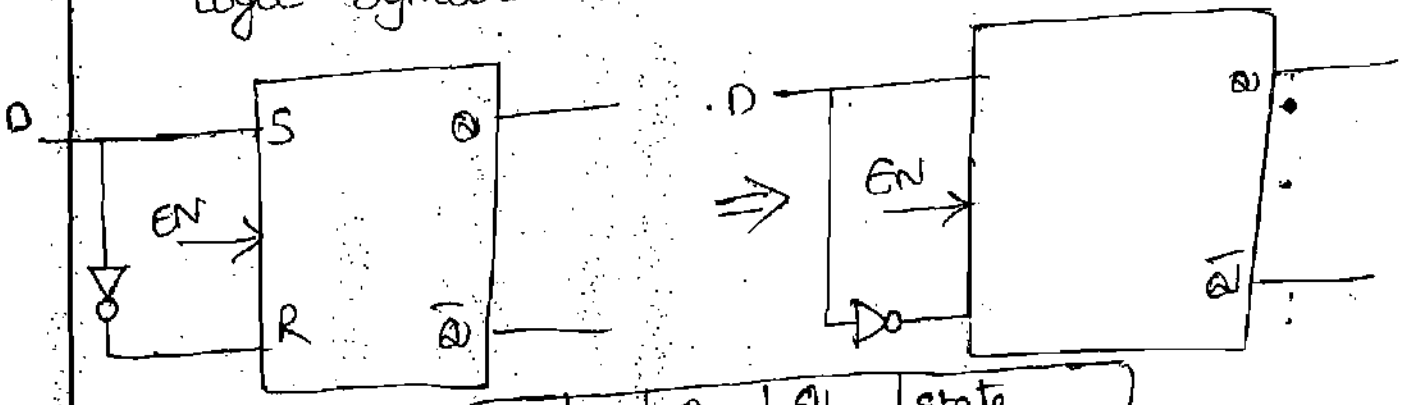
The Gated D-latch, - In many applications, it is not necessary to have separate S and R inputs to a latch. If the input combinations S=R=0 and S=R=1 are never needed, the S and R are always the complement of each other. So, to construct a latch with a single input (S) and obtain the R input by inverting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

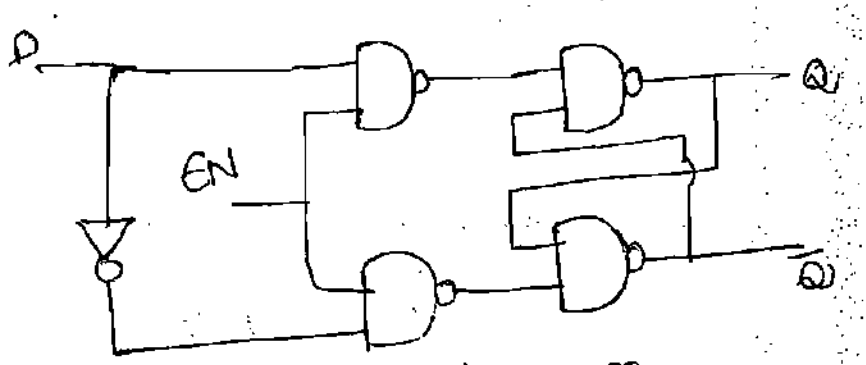
→ The output Q follows the D-input when En is high. So this latch is said to be transparent.

Logic Symbol

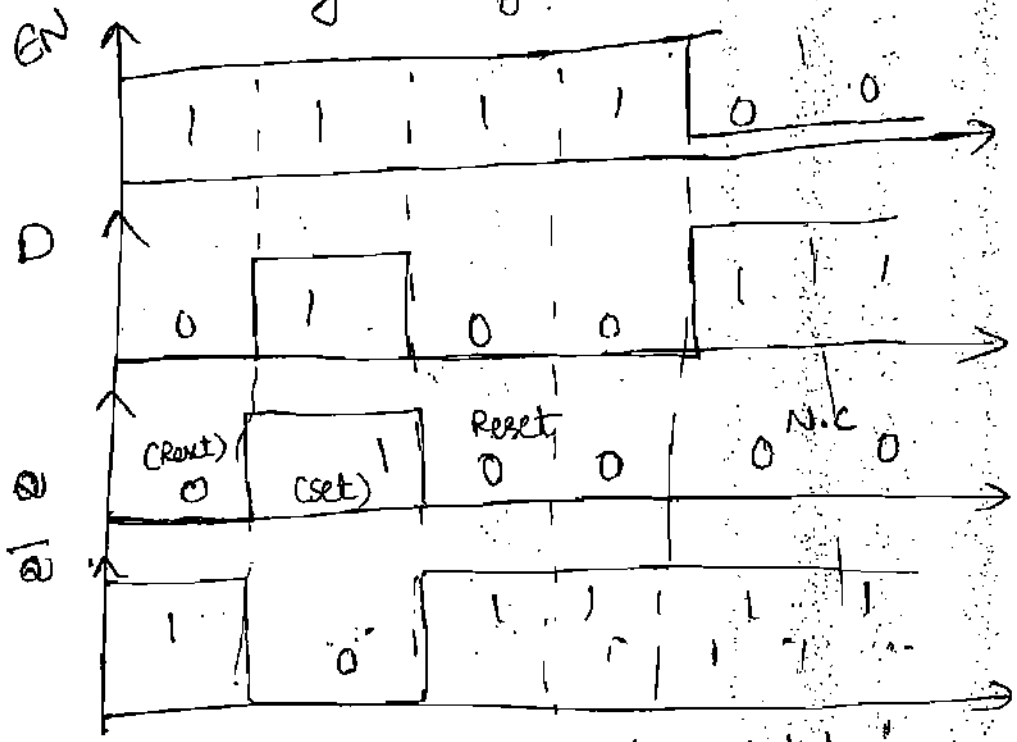


EN	D	Q_n	Q_{n+1}	State
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Set
1	1	1	1	
0	X	0	0	NO change (NC)
0	X	1	1	

Truth table



logic diagram.



waveforms for Gated D-latch.

Flip-flops :-

Types of flip-flops :-

→ S-R flip-flop

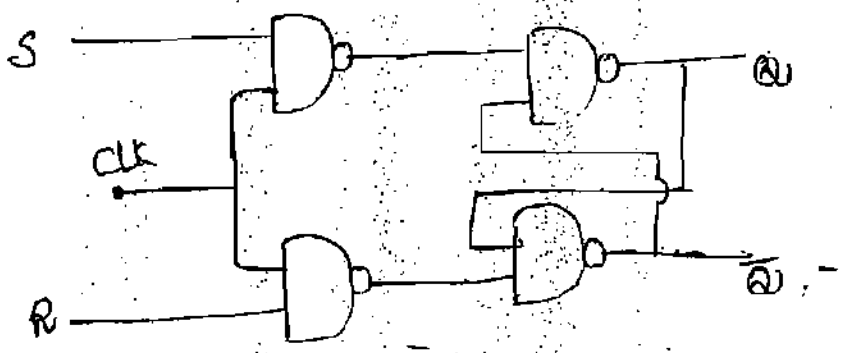
→ D flip-flop

→ J-K flip-flop

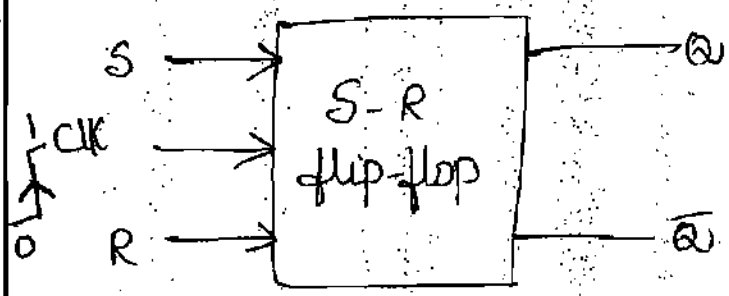
→ T flip-flop.

S-R flip-flop :-

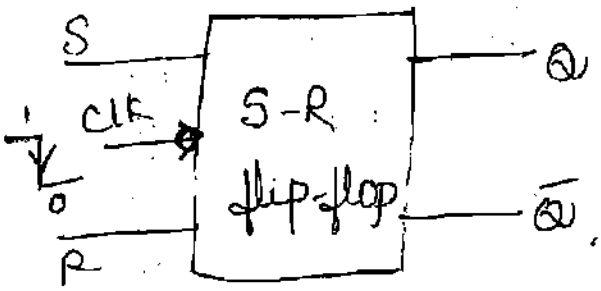
logic diagram



symbol of +ve edge trigger.



symbol of -ve edge trigger

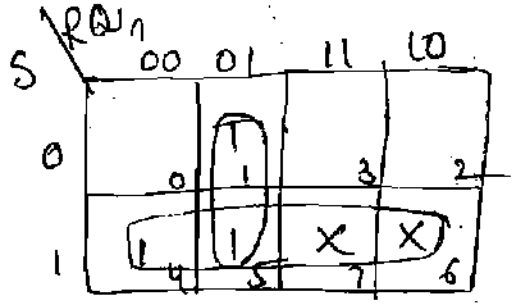


Truth table :-

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.



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

Truth table :-

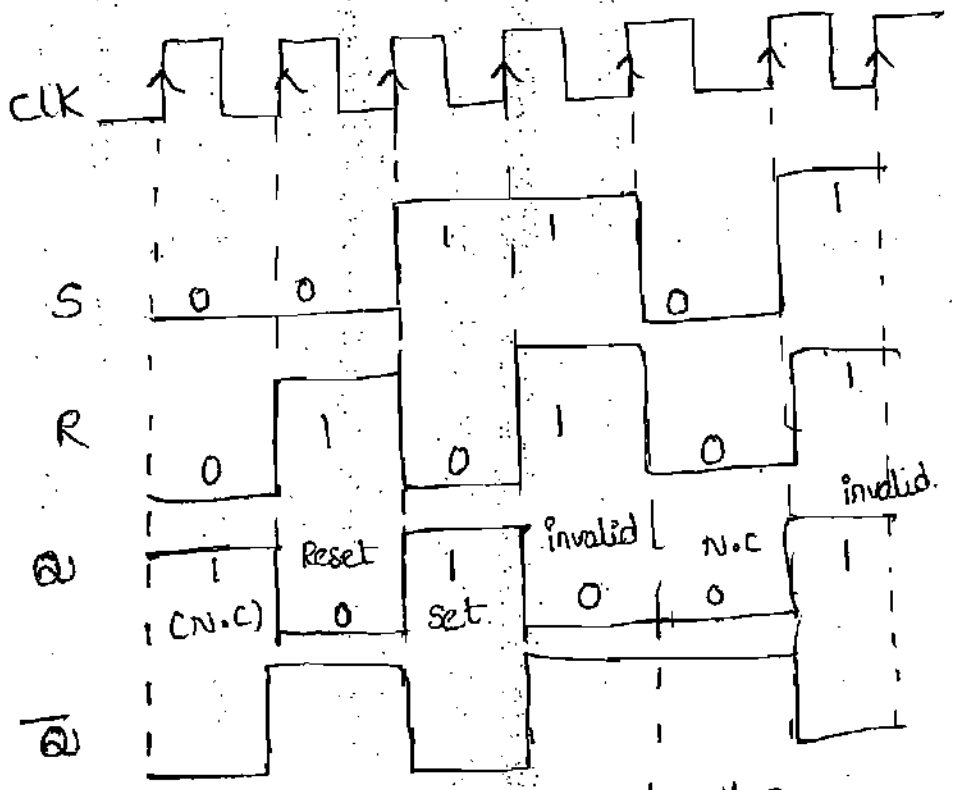
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	X

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	Required R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

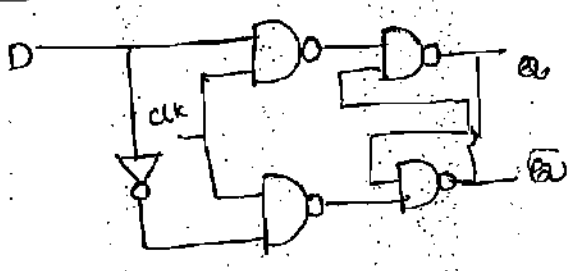
Timing diagram :-



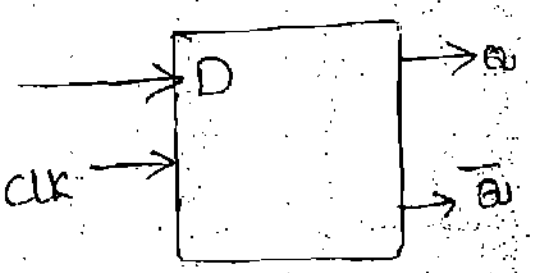
waveforms for S-R flip-flop.

D-flip-flop :-

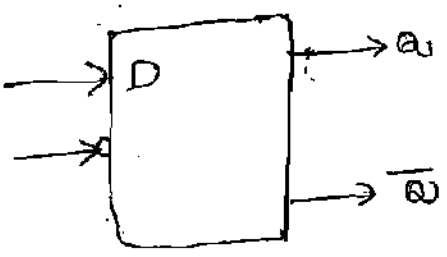
logic diagram :-



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 :-

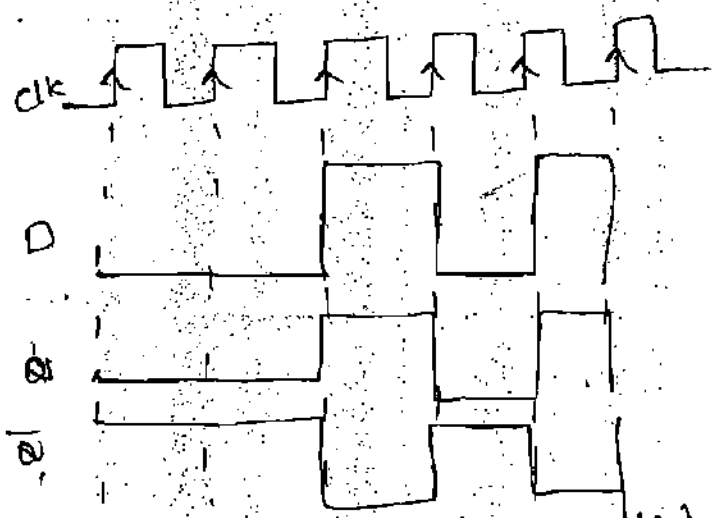
D \ Q_n	0	1
0	0	1
1	1 (2)	1 (3)

$Q_{n+1} = D$.

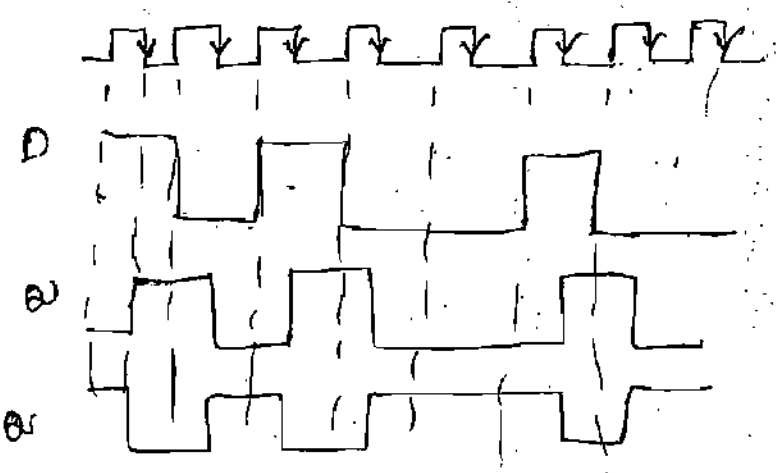
Excitation table :-

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

waveforms



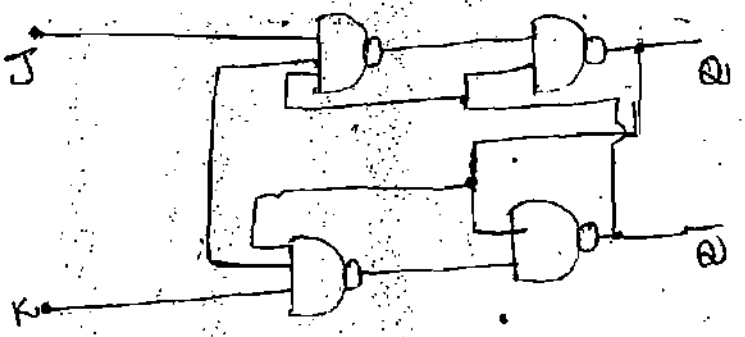
(Positive - edge trigger clk).



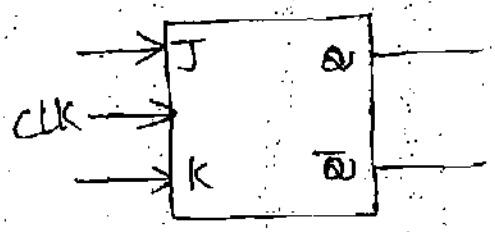
(negative - edge trigger clk).

(J-K - flip flops).

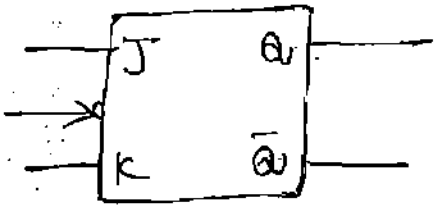
logic diagram



logic symbol.



(positive-edge).



(negative-edge)

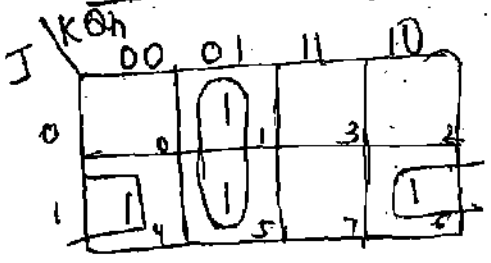
Truth table:-

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.

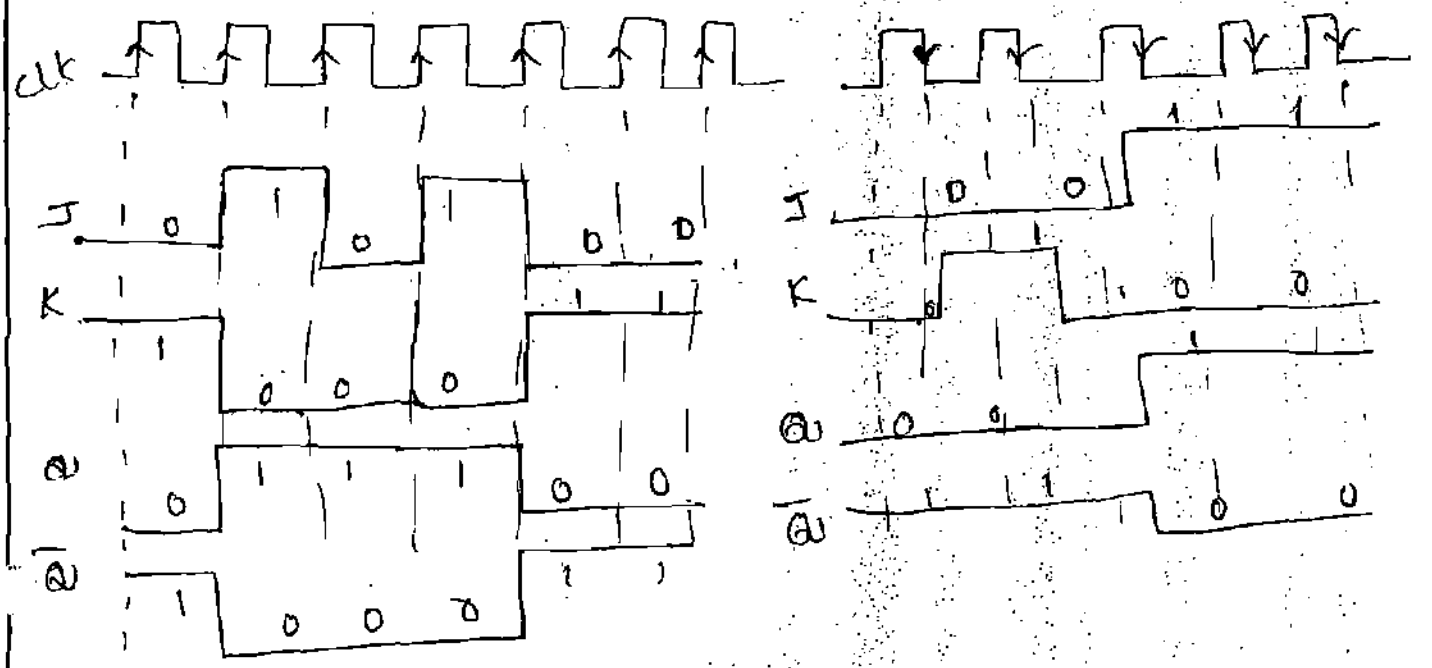


$$Q_{n+1} = \bar{K}Q_n + J\bar{Q}_n$$

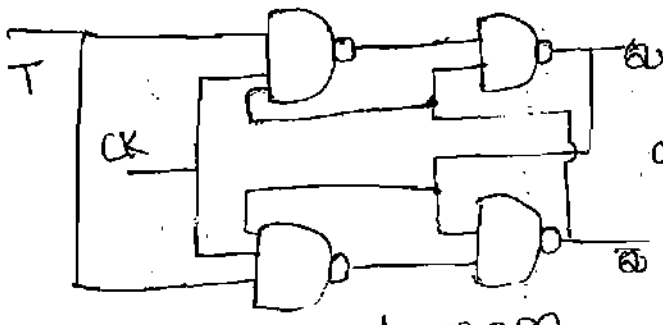
excitation table

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

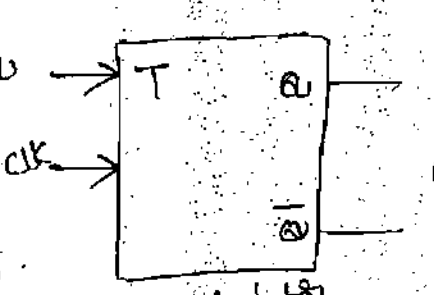
Wave-forms for Positive-edge triggering | negative-edge triggering



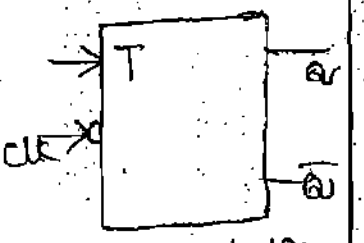
T-flip flop :-



logic diagram.



symbol for (Positive-trigger)



symbol for (negative-trigg)

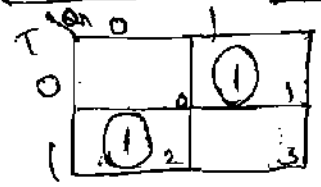
Truth table

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

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	
↑	x	0	0	NO change
↑	x	1	1	

Characteristic equation



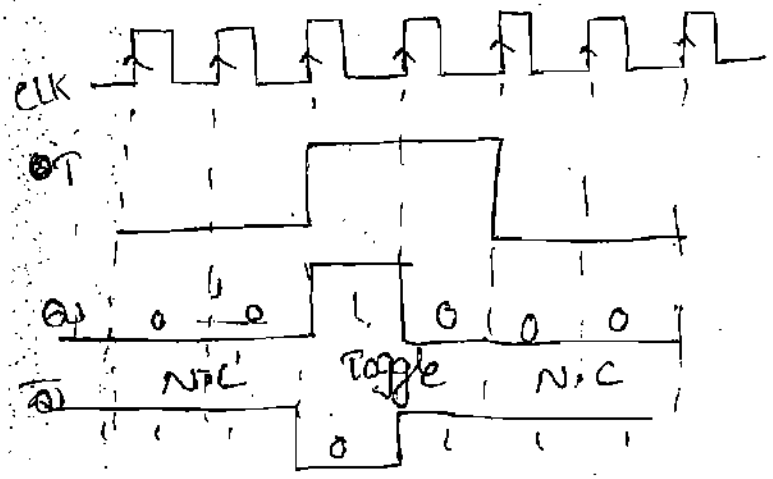
$$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

~~Existion~~

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.

$$t_p >> \tau$$

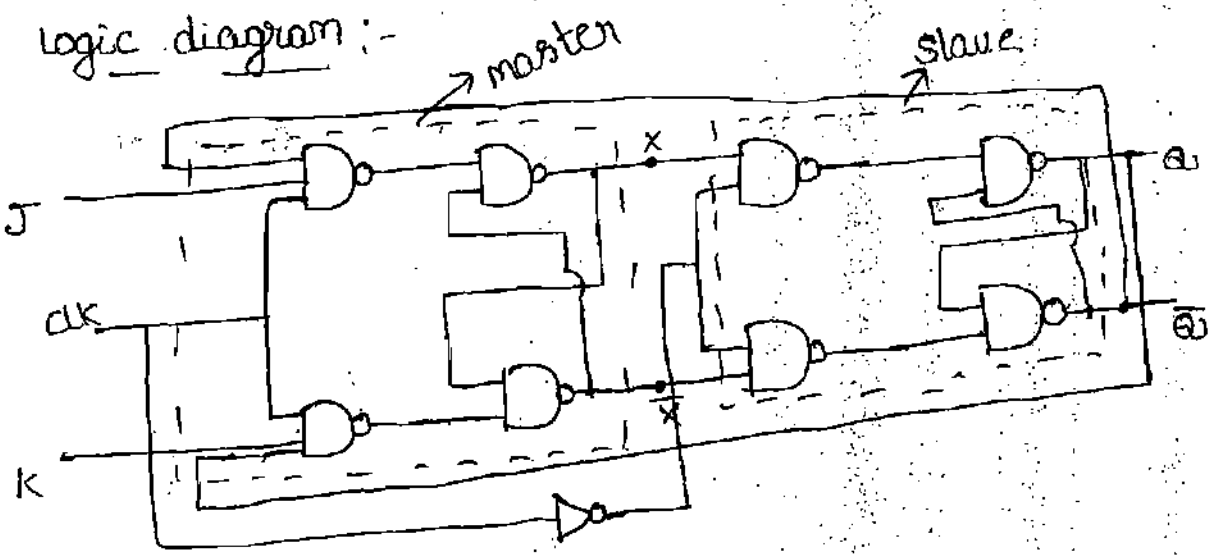
$t_p \rightarrow$ pulse width

$\tau \rightarrow$ propagation delay.

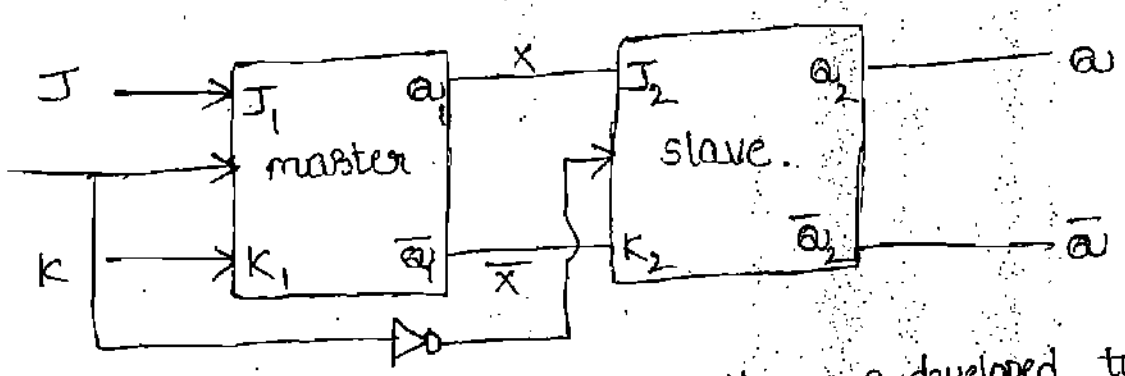
The clock pulse width should be such as to allow only one change to complement the state and not too long to allow many changes resulting in uncertainty about the final state. This is a stringent requirement which cannot be ensured in practice. This problem is eliminated using master-slave flip-flop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

master-slave J-K flip-flop :-

logic diagram :-



Symbol :-



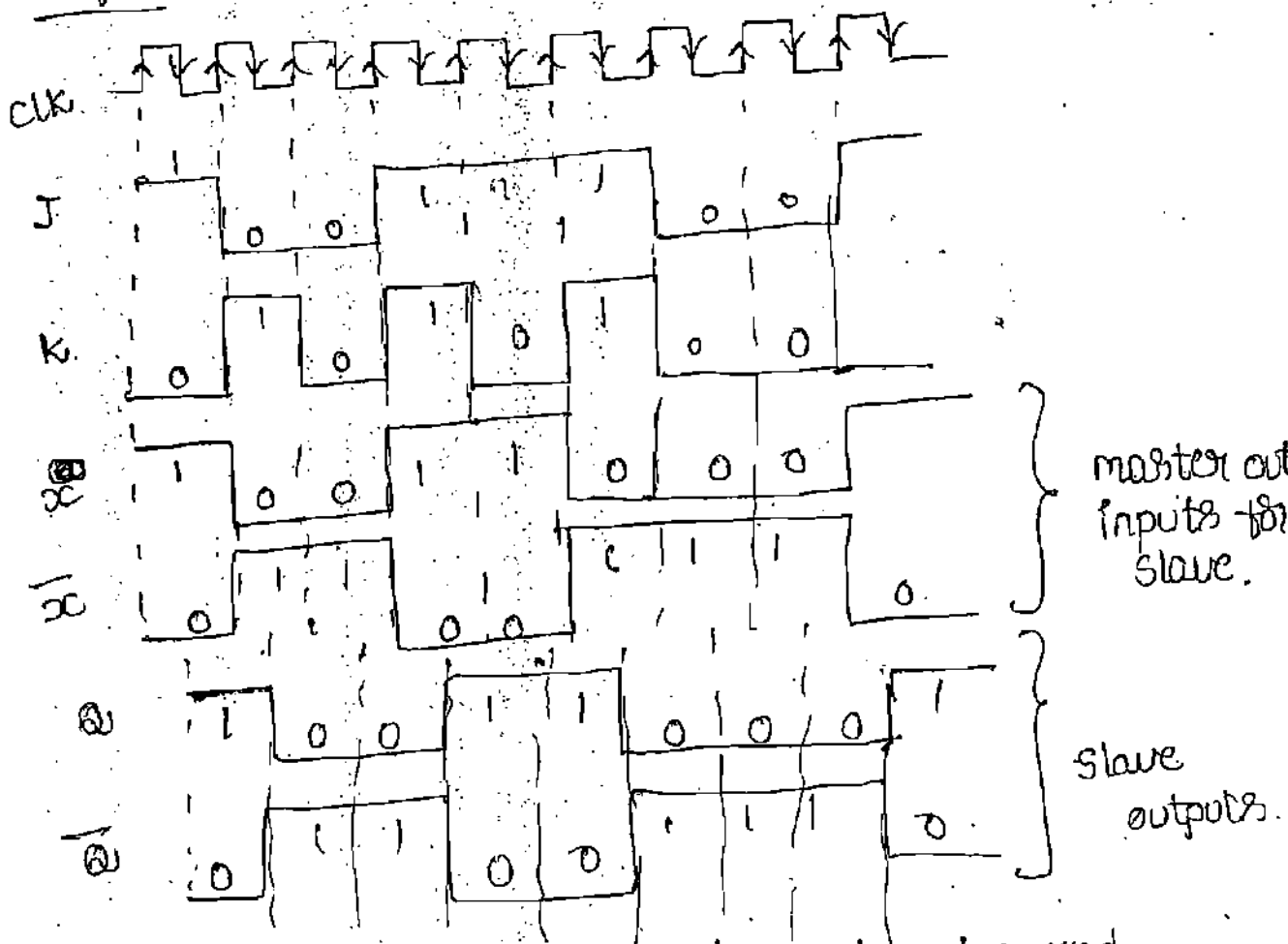
The master-slave flip flop was developed to make the synchronous operation more predictable, that is, to avoid the problems of logic race in clocked flip-flops. This improvement is achieved by introducing a known time delay between the time that the flip-flop responds to a clock pulse and the time the 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 Q and $\bar{Q}$.

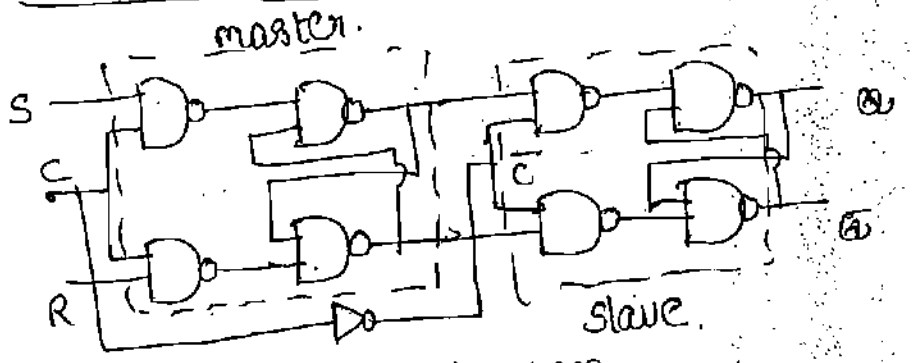
The master-slave flip-flops function very much like the negative-edge triggered flip-flops except for one more disadvantage. The control inputs must be held stable while clk is high, otherwise an unpredictable operation may occur. This problem with the master-slave flip-flop is overcome with an improved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

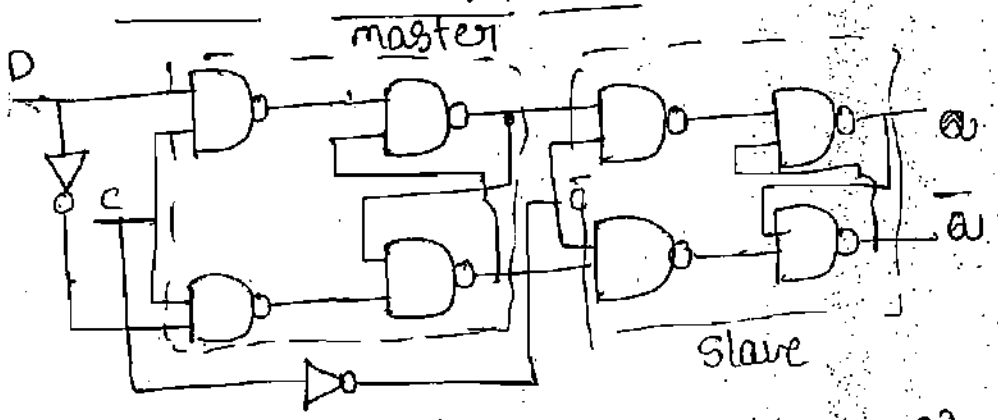


Logic diagram

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

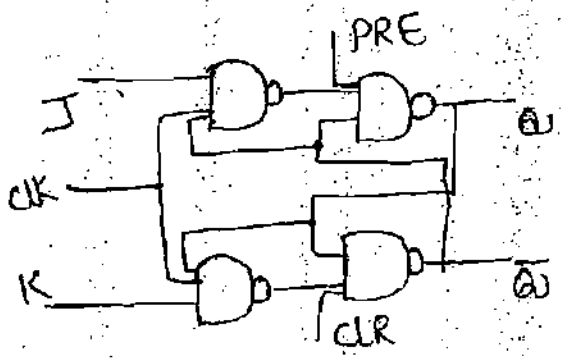
D	clk	Q	State
0	⎍	0	Reset
1	⎍	1	Set

→ Asynchronous inputs (PRESET and CLEAR).

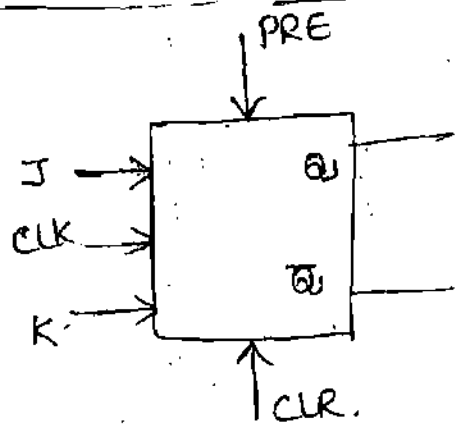
The S-R, D, T and J-K inputs are called synchronous inputs, because their effect on the flip-flop output is synchronized with the clock input.

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)x

→ 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'.

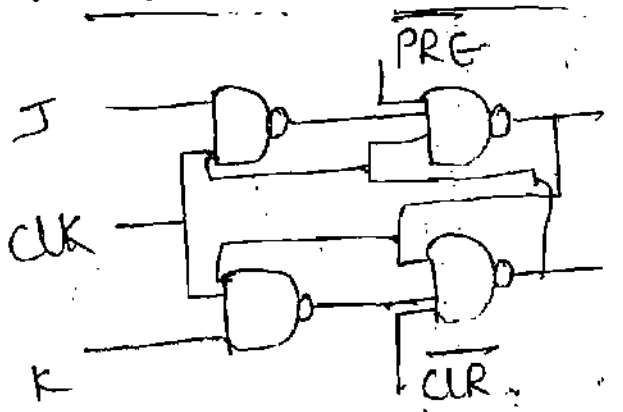
→ when PRE = 1, CLR = 0 then Asynchronous input PRESET is active the flip flop output is '1'.

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

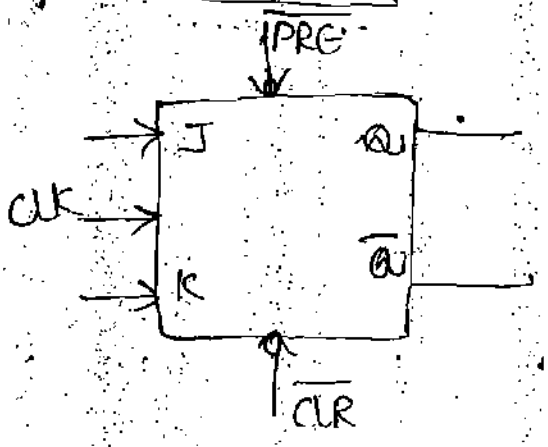
DC SET (PRE)	DC RESET (CLR)	FF response
0	0	clock operation
0	1	Q = 0
1	0	Q = 1
1	1	Not used.

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 A synchronous 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

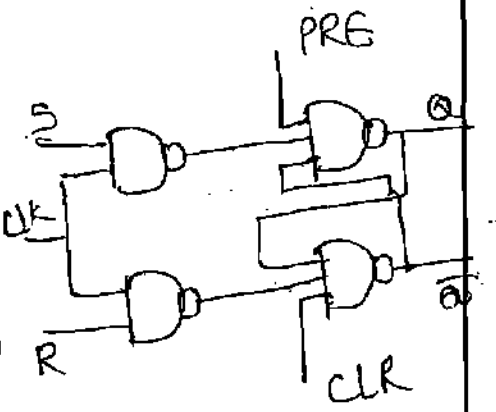
Truth table for J-K flip-flop (both Asynchronous & synchronous inputs).

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	↑	1	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
0	0	↑	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.	

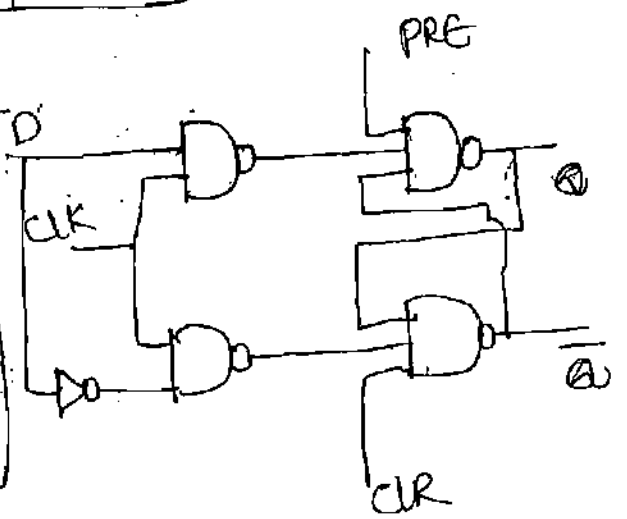


logic diagram

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 flip flop

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

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

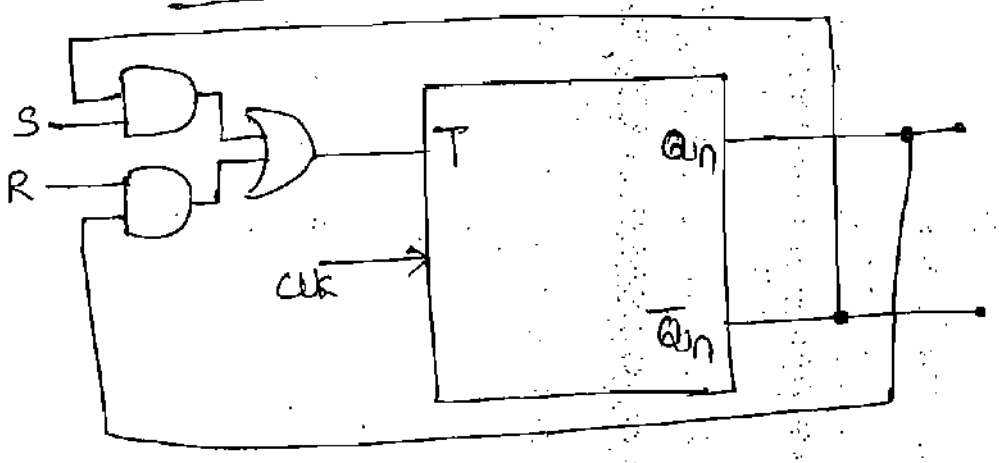
S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	X	X
1	1	1	X	X

K-map for T

S \ RQ_n	00	01	11	10
0	0 (0)	0 (1)	0 (3)	0 (2)
1	1 (4)	0 (5)	X (7)	X (6)

$T = S.\overline{Q_n} + R.Q_n$.

Logic diagram



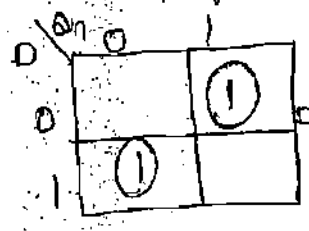
→ Conversion of T-flip-flop to D-flip-flop.

Available → T-flipflop → excitation table

Required → D-flipflop → characteristic table.

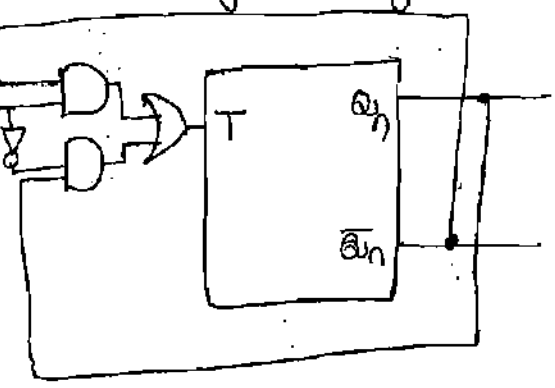
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

K-map for T



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

Logic diagram



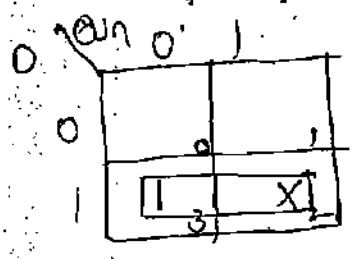
→ Construct a D-flip flop ~~by~~ using S-R flip flop.

Available → S-R flipflop → excitation table.

Required → D flip flop → characteristic table.

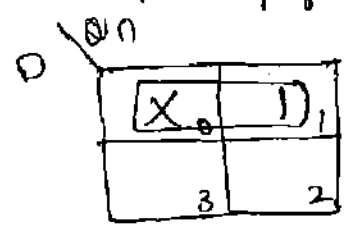
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

K-map for S



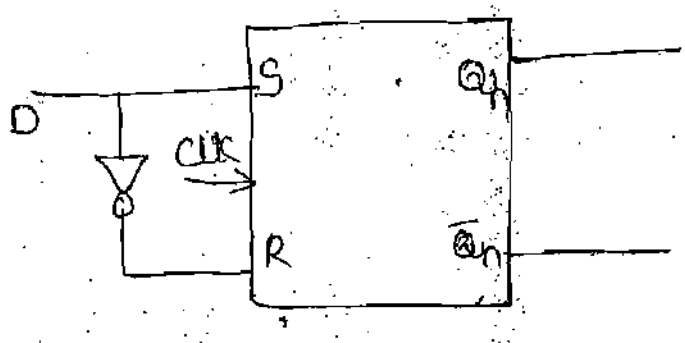
$S = D$

K-map for R



$R = \bar{D}$

logic diagram.



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

Available → 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 \ $K Q_n$	00	01	11	10
0	0	X	0	0
1	1	X	0	1

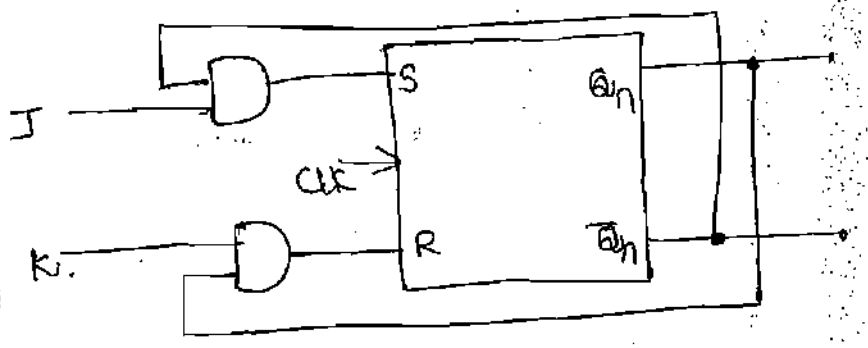
$S = J\bar{Q}_n$

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

K-map for J

T \ Q_n	0	1
0	0	X
1	1	X

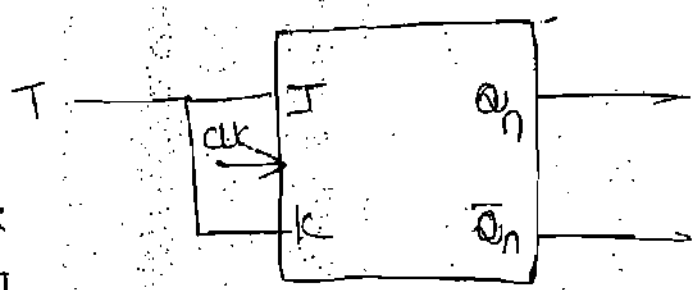
$J = T$.

K-map for K

T \ Q_n	0	1
0	X	0
1	X	1

$K = T$

logic-diagram

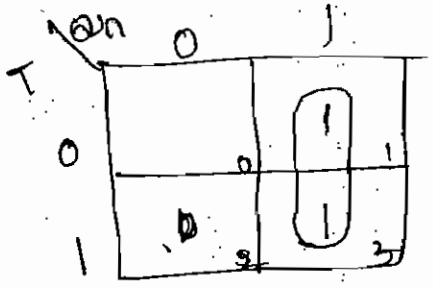


→ conversion of D-flip flop to T flip-flop.

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

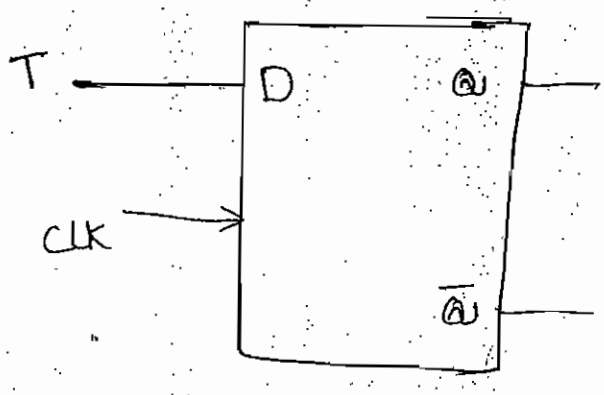
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

K-map for D



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 or synchronous counters. Asynchronous counters are also called ripple counters.

→ In asynchronous counters flip flops 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 FF's 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 A synchronous counters.

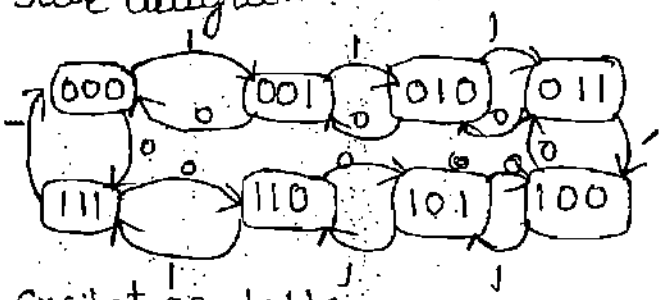
Design steps of synchronous counters :-

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

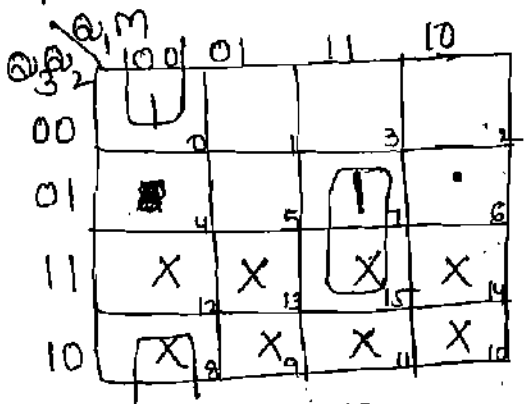
m=1 up counting

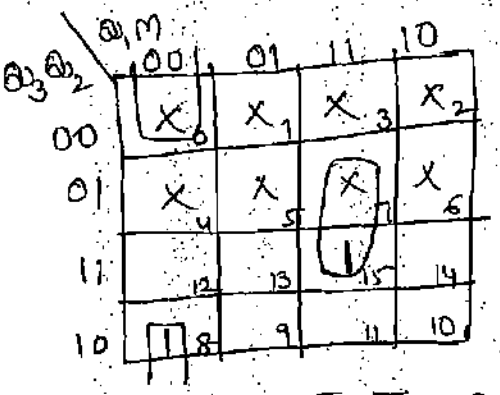
Step 3 :- Excitation table

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	1

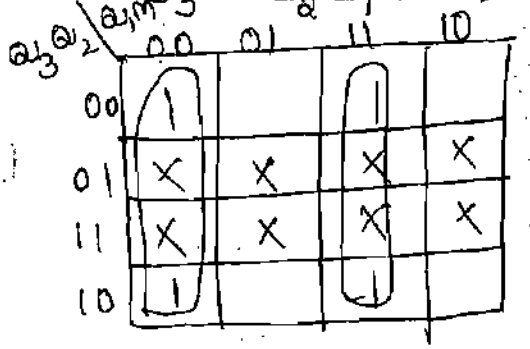
Step 4:- obtain the minimal expression.

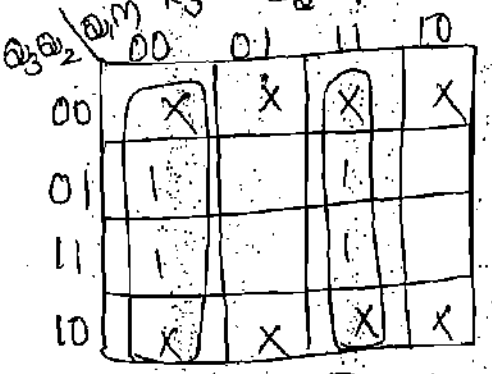




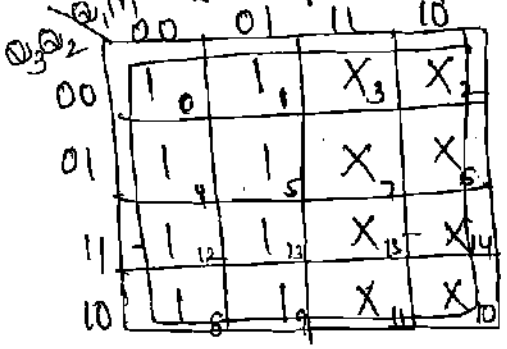
$K_3 = \overline{Q_2}\,\overline{Q_1}\,\overline{M} + Q_2 Q_1 M$

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



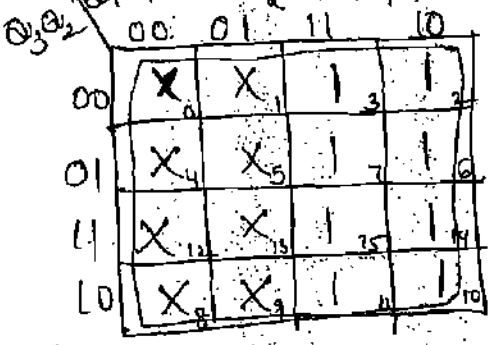


$J_2 = \overline{Q_1}\,\overline{M} + Q_1 M$



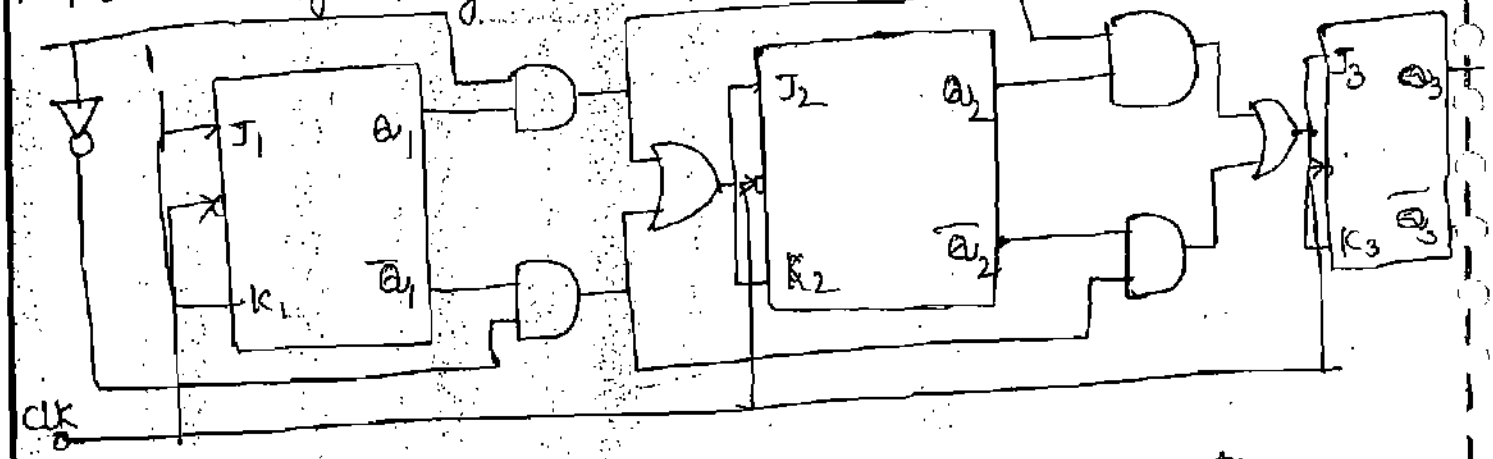
$J_1 = 1$

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



$K_1 = 1$

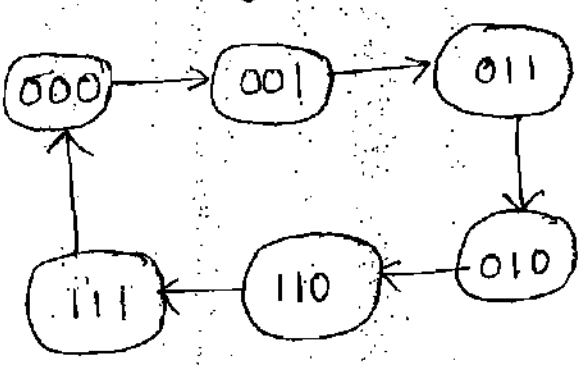
Step 5 :- The logic diagram.



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

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

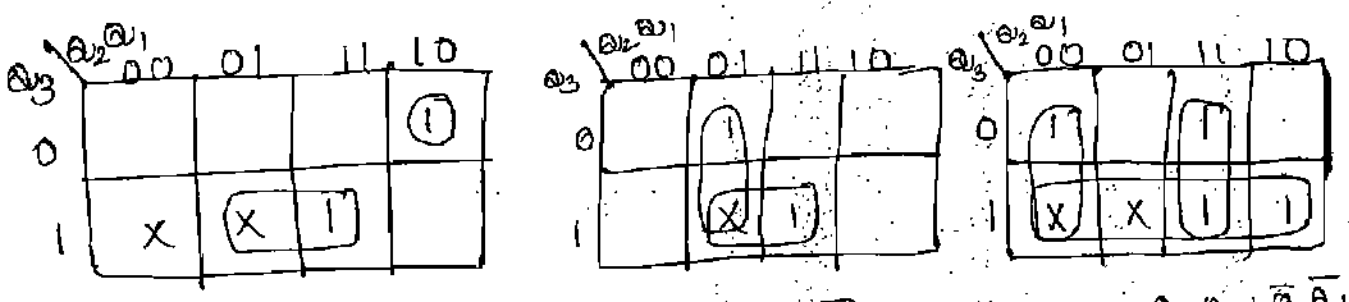
Step 2 :- state diagram



Step 3 :-

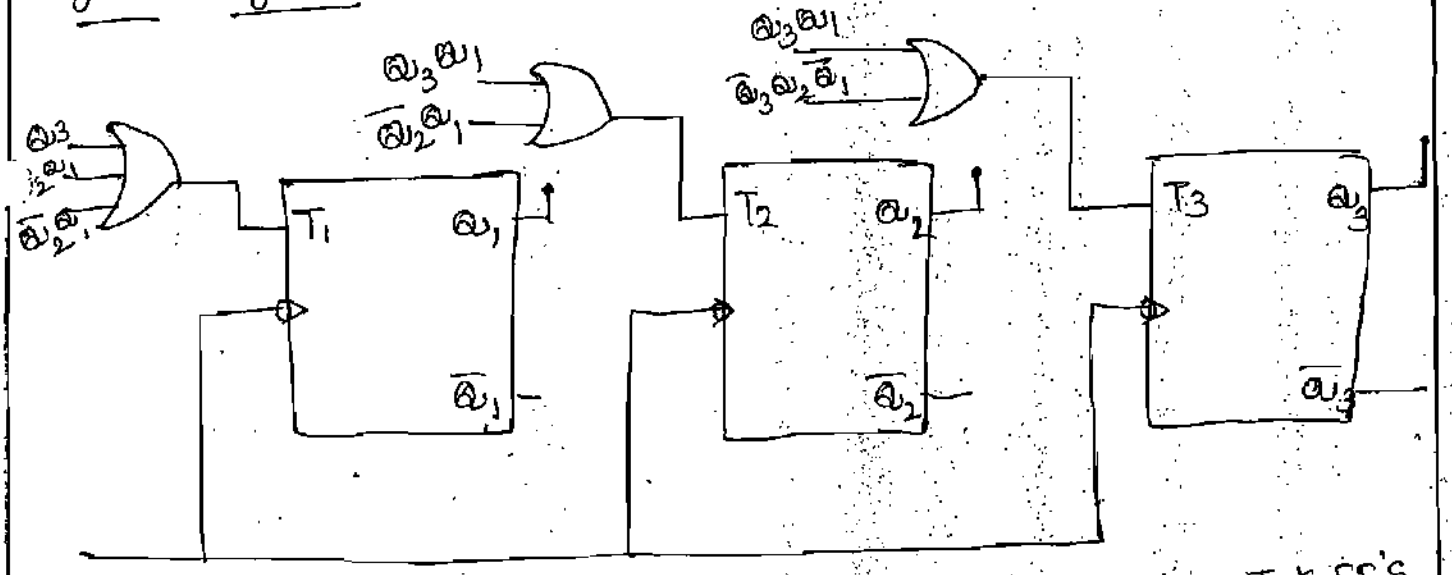
Present state			next state			Required excitations		
Q_3	Q_2	Q_1	Q_3	Q_2	Q_1	T_3	T_2	T_1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Step 4 :- minimal expressions :-



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

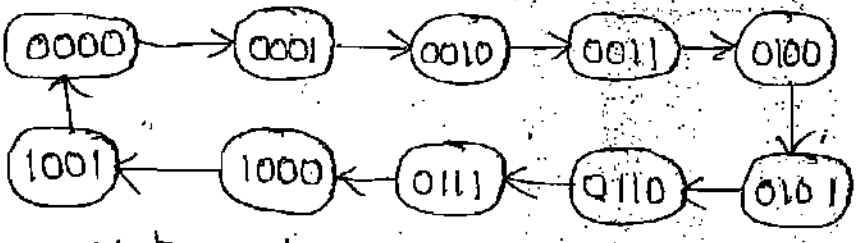
logic diagram :-



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

Step 1 :- 4 FF's

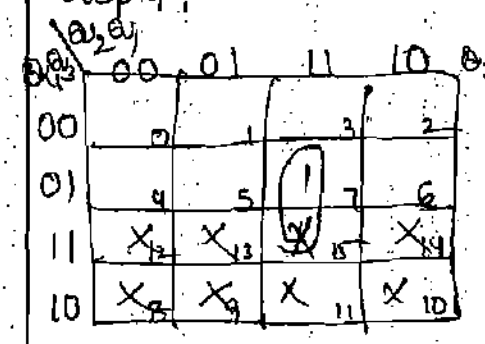
Step 2 :- State diagram.



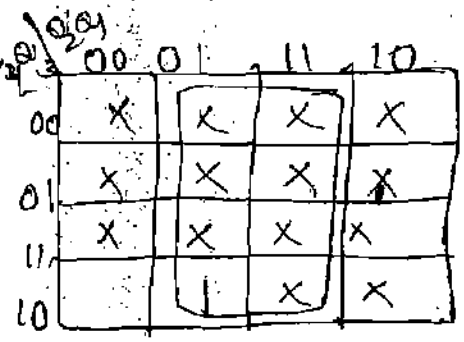
Step 3 :- excitation table.

Present state Q_4 Q_3 Q_2 Q_1	next state Q_4 Q_3 Q_2 Q_1	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_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

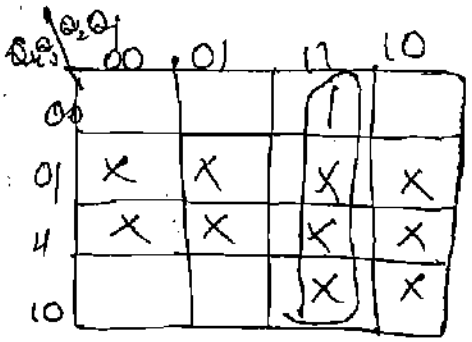
Step 4 :-



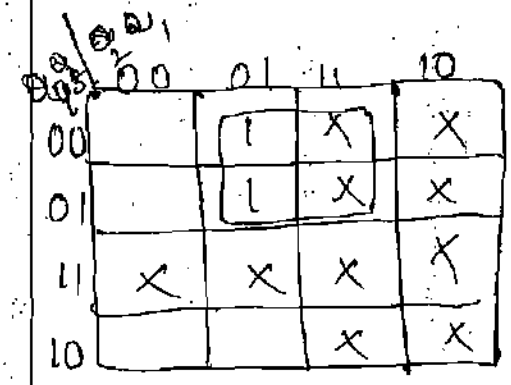
$J_4 = Q_3 Q_2 Q_1$



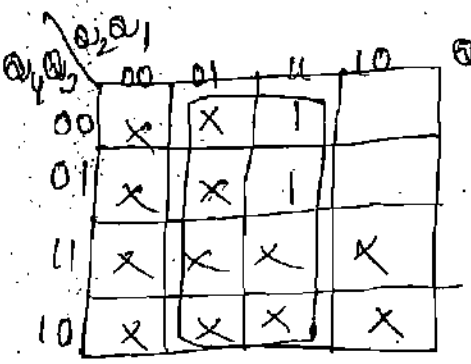
$K_4 = Q_1$



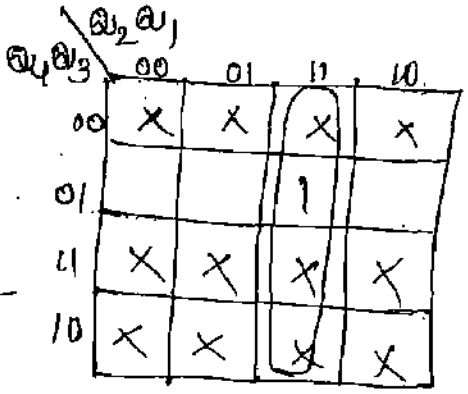
$J_3 = Q_2 Q_1$



$J_2 = \overline{Q_4} Q_1$



$K_2 = Q_1$



$K_3 = Q_2 Q_1$

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

$J_1 = Q_2$

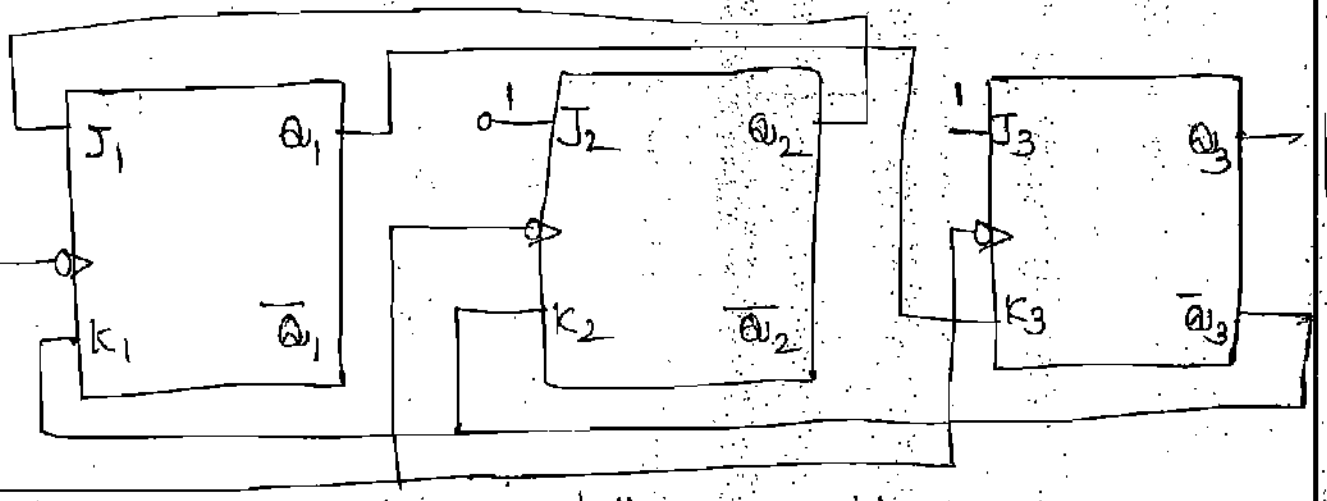
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	x	x	1	x
1	x	x		x

$K_1 = \overline{Q_3}$

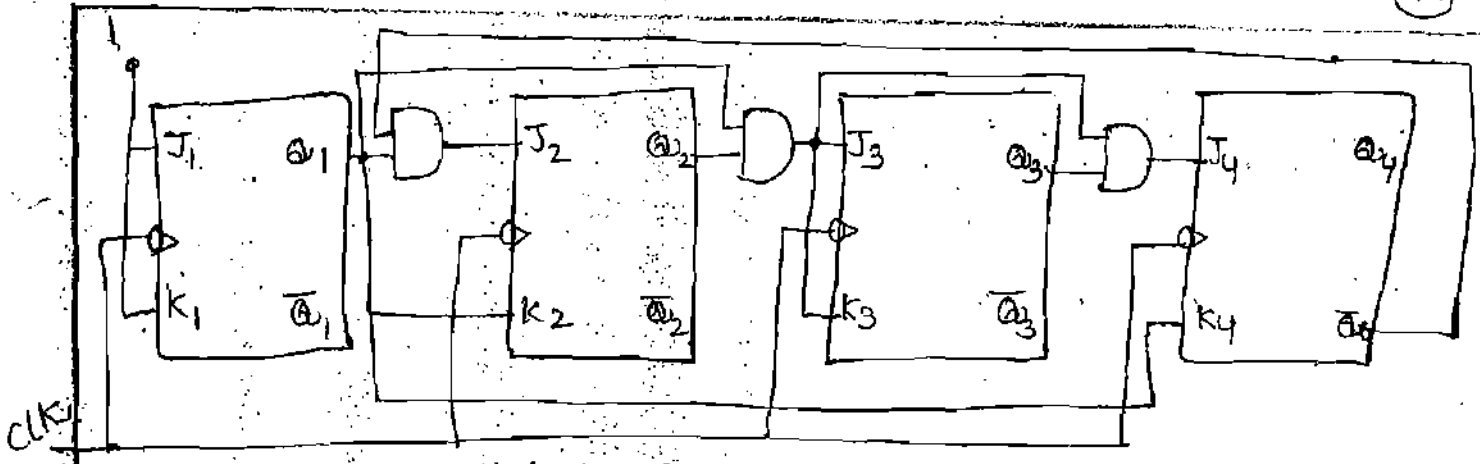
$Q_3 \backslash Q_2Q_1$	00	01	11	10
0	x	x	x	x
1	1	x	x	x

$J_2 = 1$

Step 5 :- logic diagram



logic diagram of the J-K counter.

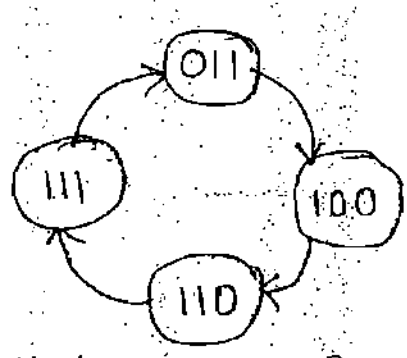


logic diagram

→ 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)

Step 1:- number of flip-flops - 3 flip-flops.

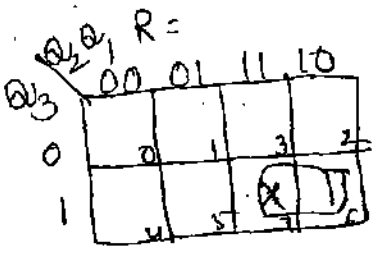
Step 2:- State diagram.



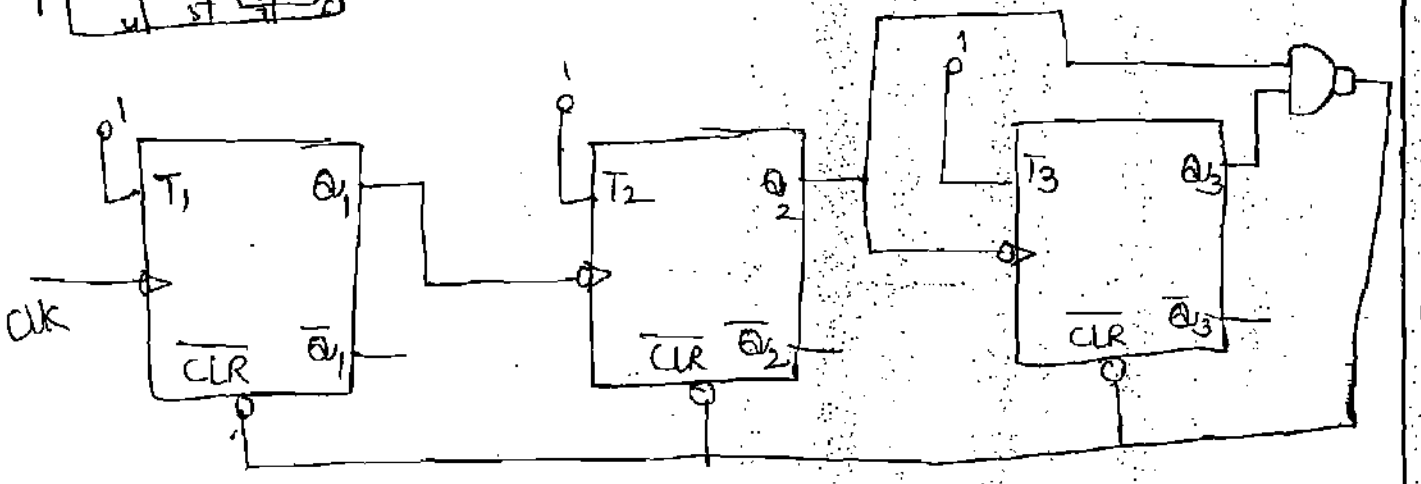
State diagram

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



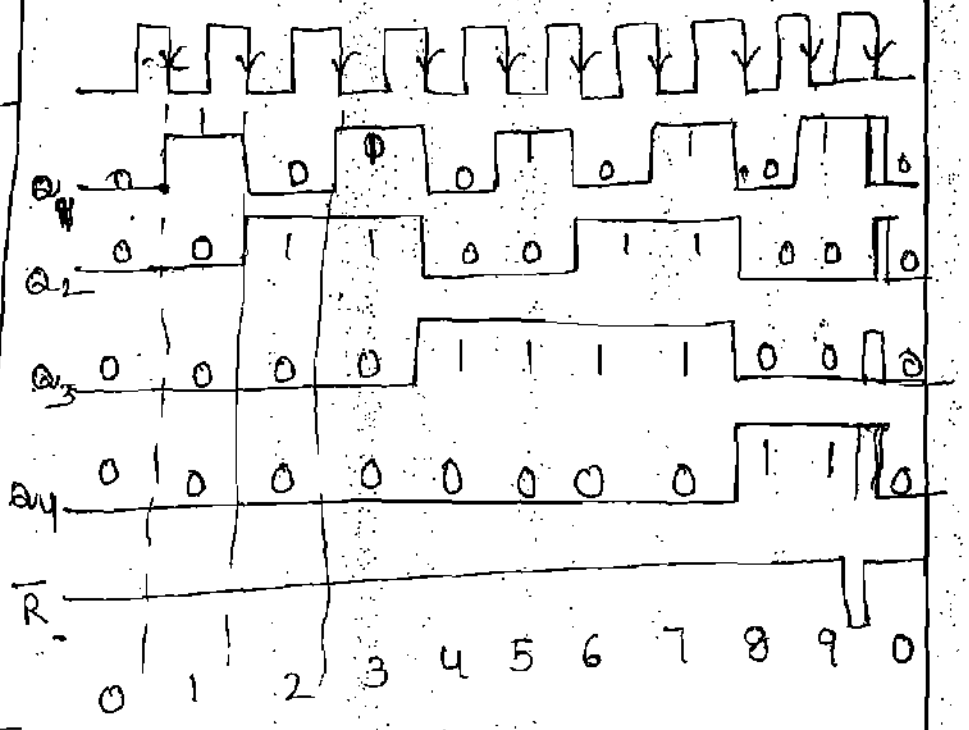
$R = Q_3Q_2$



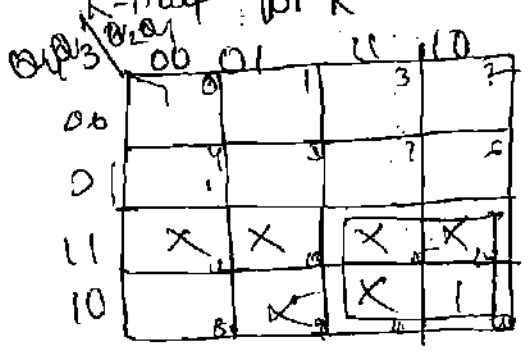
logic diagram

mod -10 Asynchronous counter :-

R	After Pulses	Count 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



R = 0 for 0000 to 1001
R = 1 for 1010
R = X for 1011 to 1111

$R = Q_4Q_2$

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 $N \le 2^n$ is $n = 3$: three FFs can have eight possible states, out of which only six are utilized and remaining two states 110 and 111.

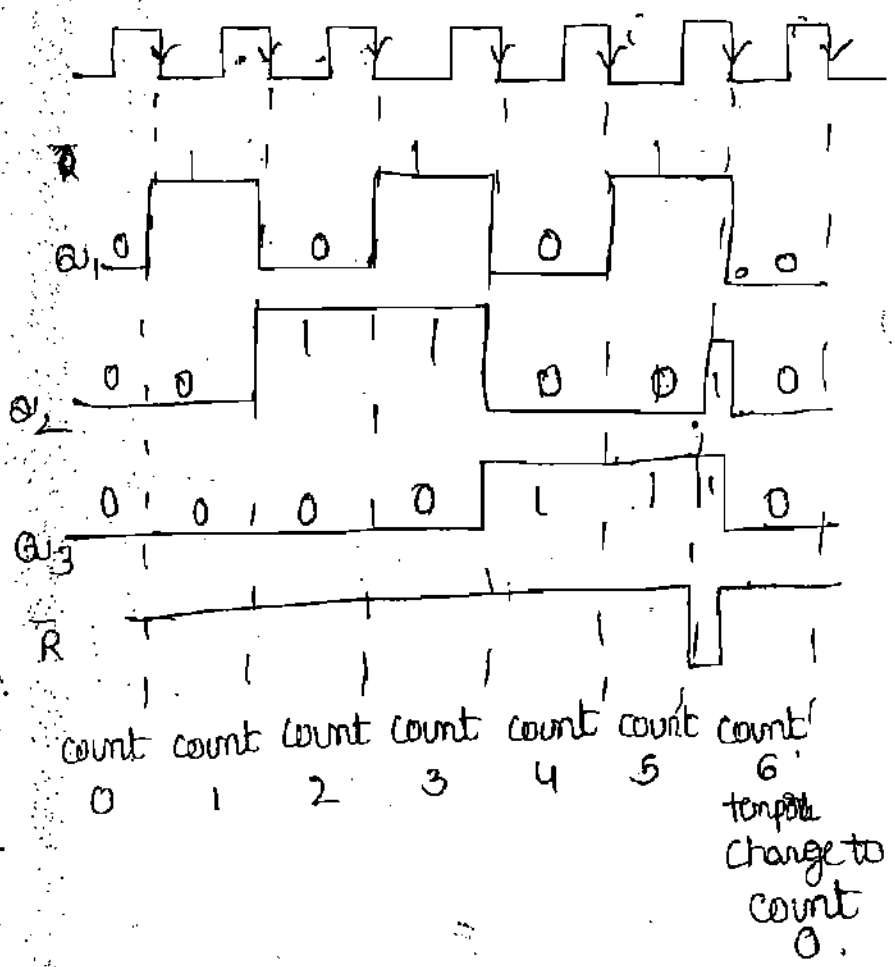
→ For the design, To write a truth table with the present state outputs Q_3, Q_2 and Q_1 as the variables, and reset R as the output.

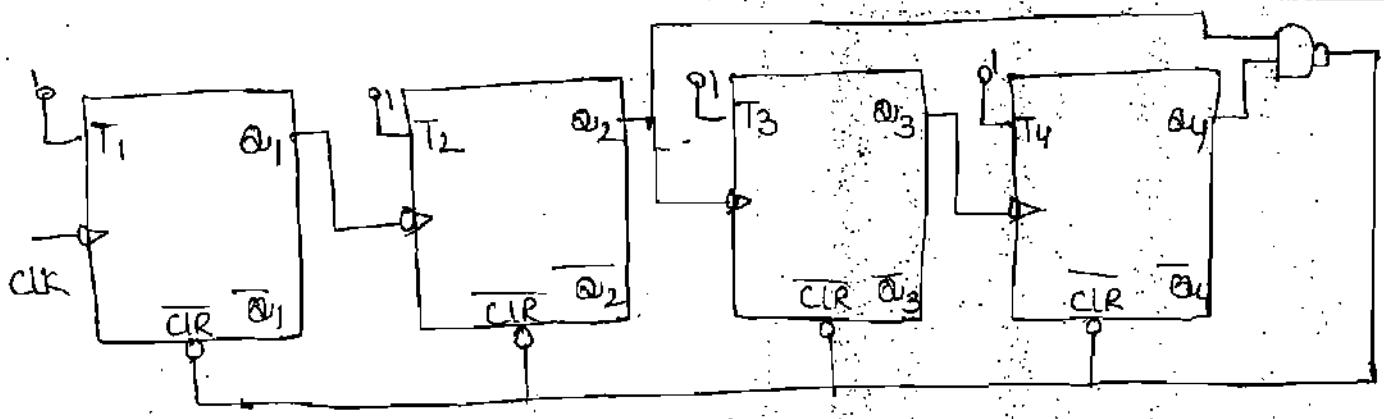
$R = 0$ for 000 to 101, $R = 1$ for 110, and $R = X$ for 111.

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	1	0	1
	↓	↓	↓	
	0	0	0	
	1	1	1	X
7	0	0	1	1

Timing diagram.





logic - diagram.

Asynchronous mod - 10 counter using T Flip-flops.

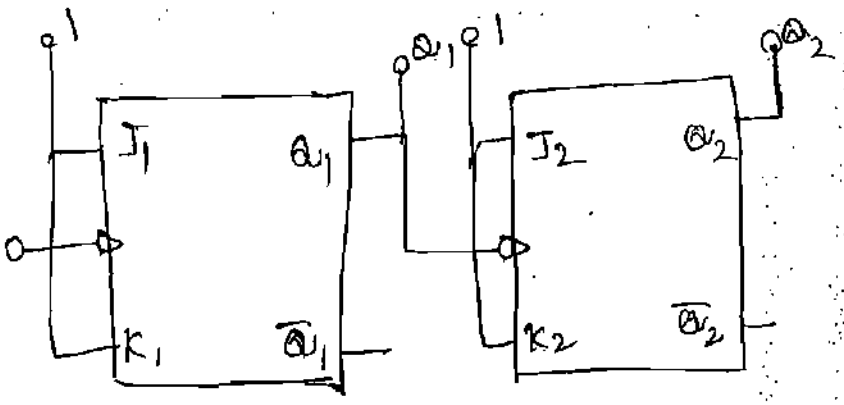
→ Two - bit ripple up - counter using negative edge - triggered flip-flops:

→ 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, FF_1 toggles at the negative-going edge of this pulse, therefore, Q_1 goes from low to high. This becomes a positive - going signal at the clock input of FF_2.

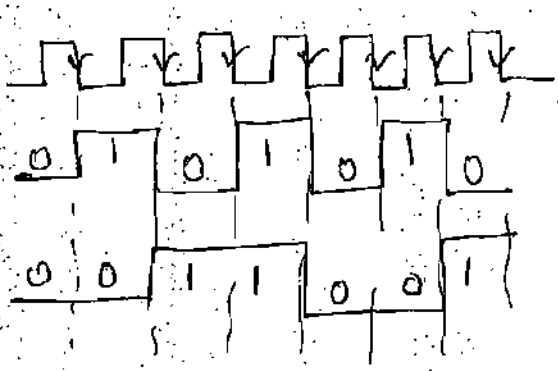
→ So FF_2 is not affected, and hence, the state of the counter after one clock pulse is $Q_1 = 1$ and $Q_2 = 0$.

→ So next clock pulse FF_1 is change to 1 to 0 then negative going edge of this pulse FF_2 is change to 0 to 1. $Q_1 = 0$ and $Q_2 = 1$

→ So next clock pulse FF_1 is change to 0 to 1 then positive going edge of this pulse FF_2 is no change $Q_1 = 1$, $Q_2 = 1$.

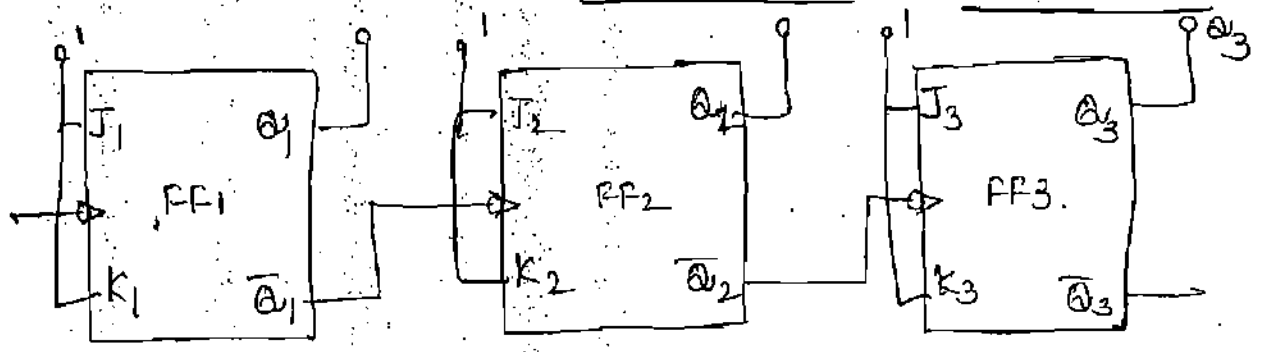


Logic diagram.



Timing diagram.

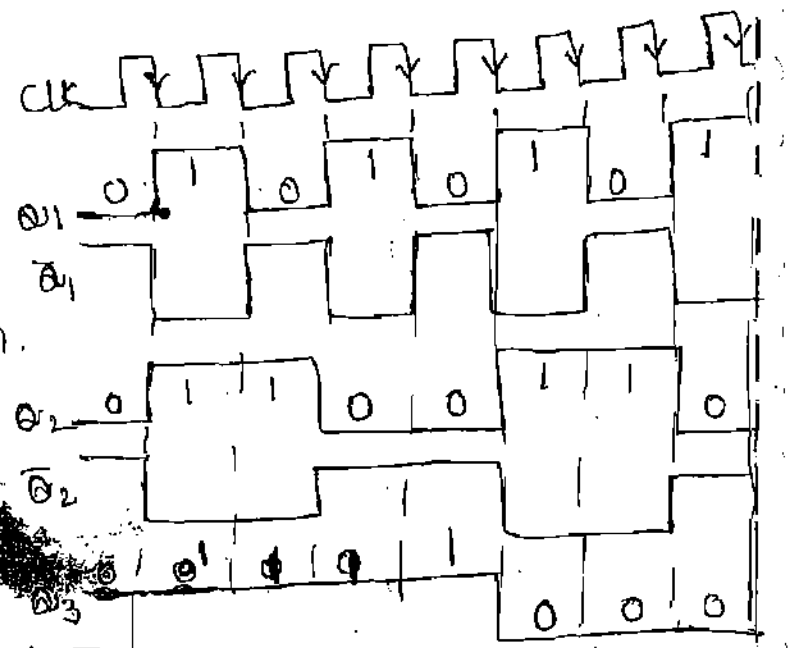
3-bit down-counter using negative-edge trigged J-K flip-flops.



Logic diagram.

A 3-bit down counter counts in order 0,7,6,5,4,3,2,1,0,1 ...

For down counting $\overline{Q}_1$ to FF_1 is connected to the clock of FF_2. the $\overline{Q}_2$ to FF_2 is connected to the clock of FF3. output taken from Q_1, Q_2 and Q_3.



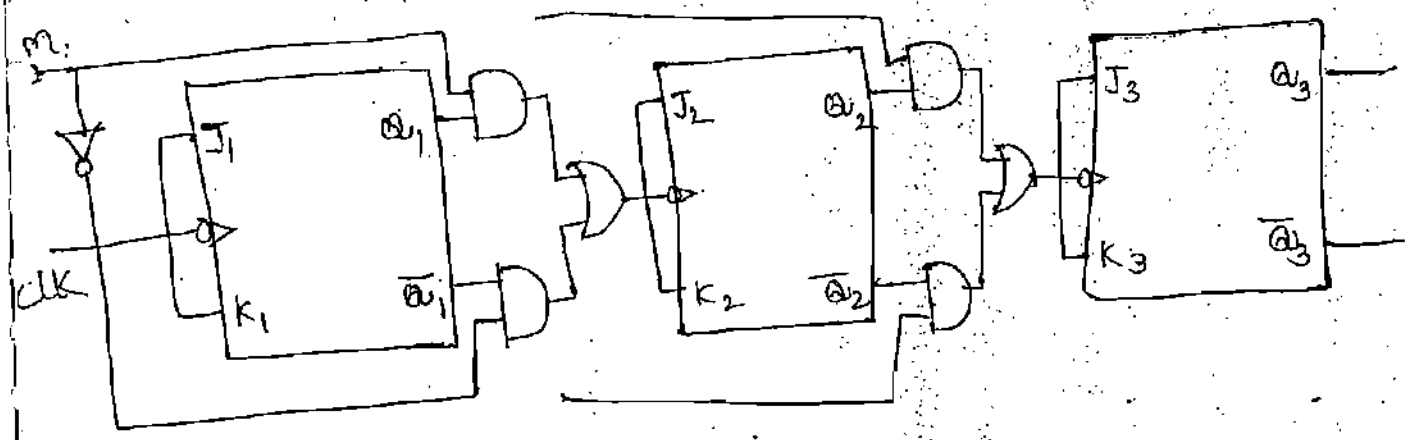
3-bit updown counter using negative edge-triggered flip flops :-

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 FF_2 and for down-counting $\overline{Q}_1$ is transmitted to clock FF_2.

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



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.

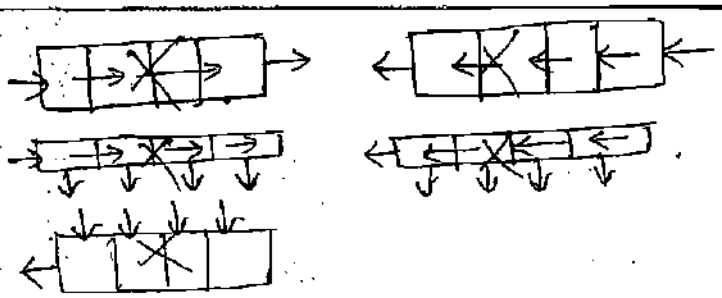
→ where as a counter has a specified sequence of states.

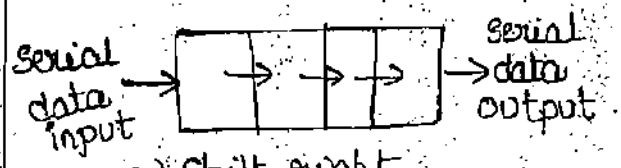
Data - Transmission in shift registers :-

→ 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.

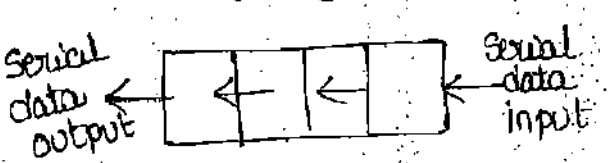
→ Data may be shifted into or out of the register either in serial form or in parallel form. So, there are four types of shift-registers. They are

1. Serial - in, Serial - out
2. Serial - in, parallel - out
3. parallel - in, serial - out
4. parallel - in, parallel - out



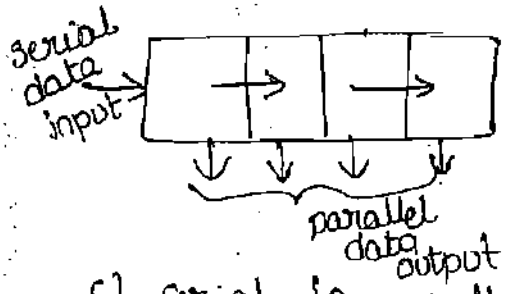


a) Shift-right

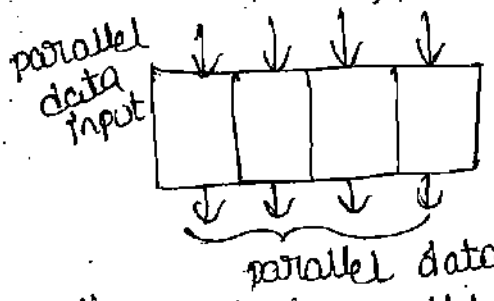


b) Shift - left.

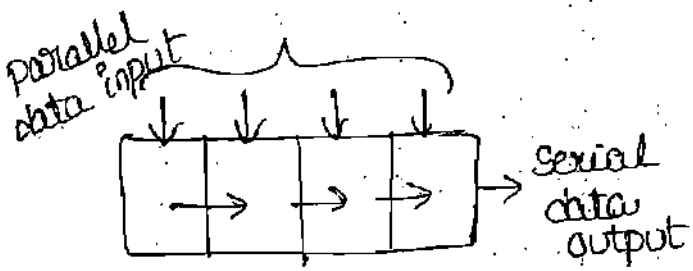
Serial - in, serial - out



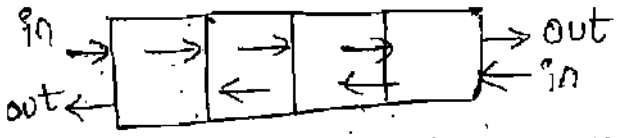
c) Serial - in, parallel out Shift regis



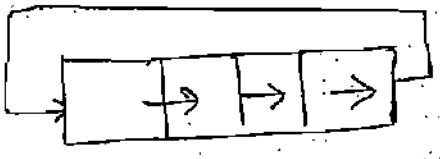
d) parallel - in, parallel - out, shift regis.



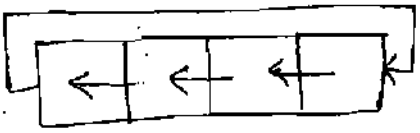
e) parallel - in, serial output



f) bi-directional shift register.

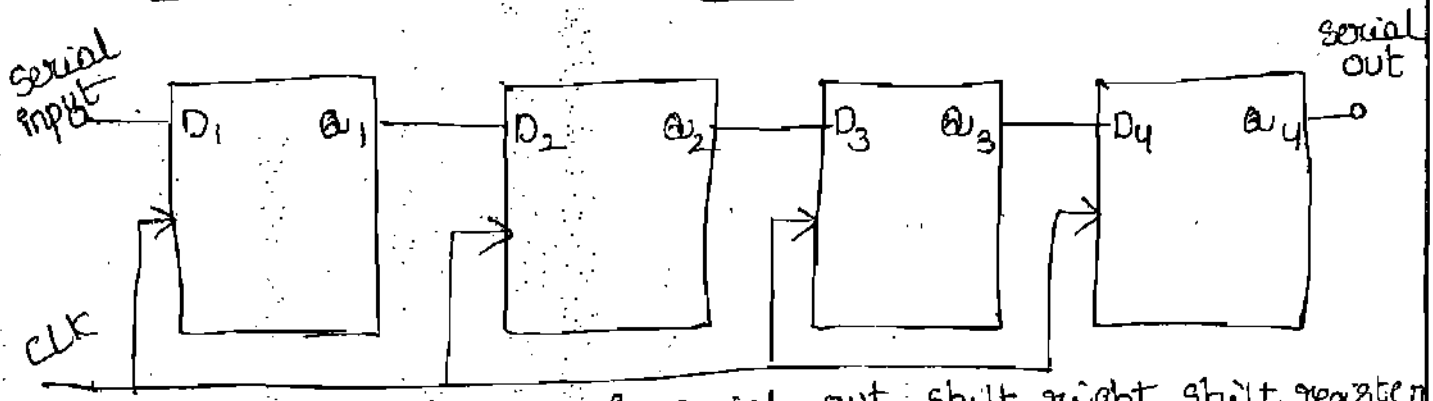


g). Rotate - right shift register

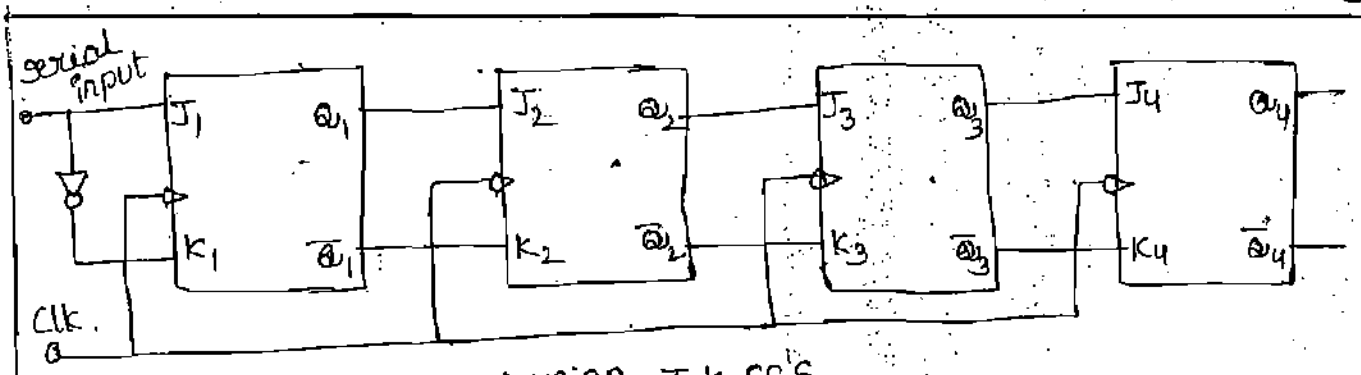


h) Rotate - left shift register.

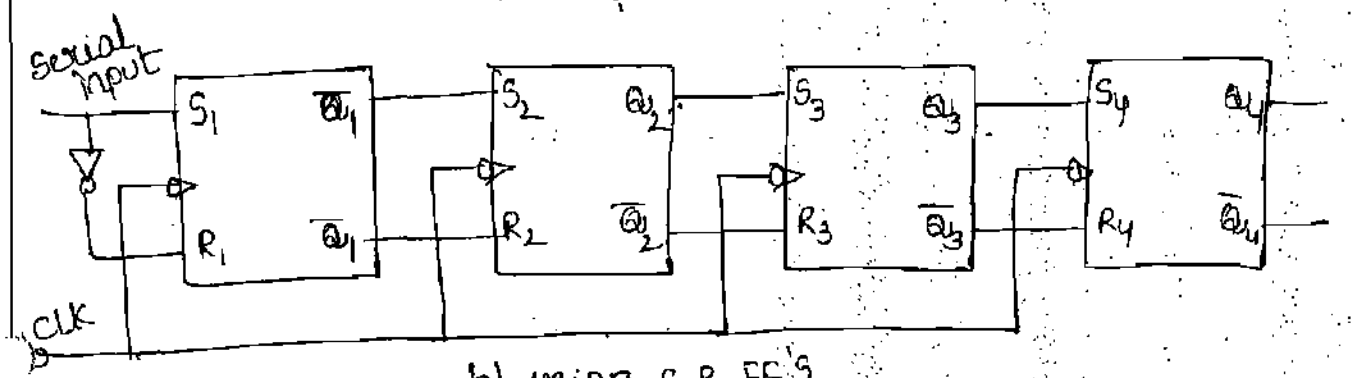
→ Serial - in, serial - out Shift register :-



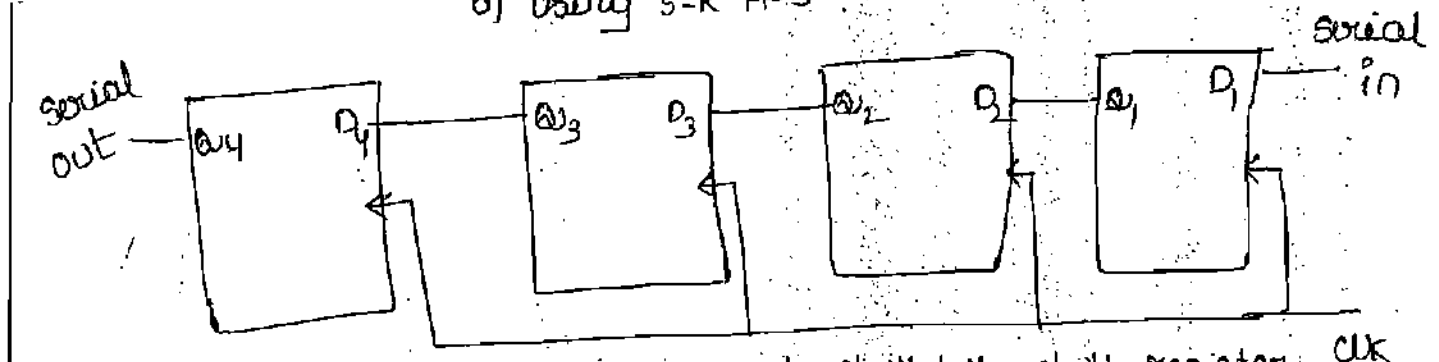
4-bit Serial - in, Serial - out, shift right shift register



a) using J-K FF'S.

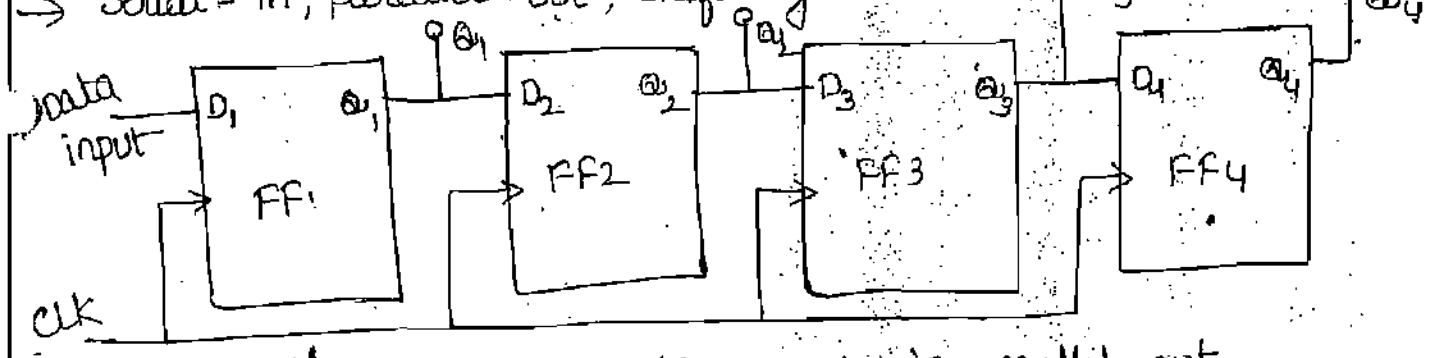


b) using S-R FF'S.

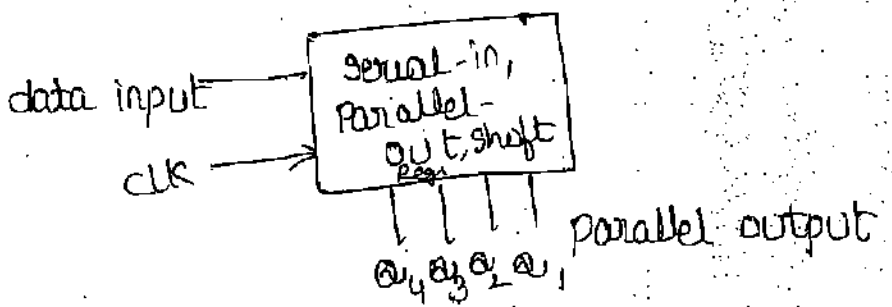


A 4-bit serial-in, serial-out, shift left shift register.

→ Serial-in, parallel-out, shift register

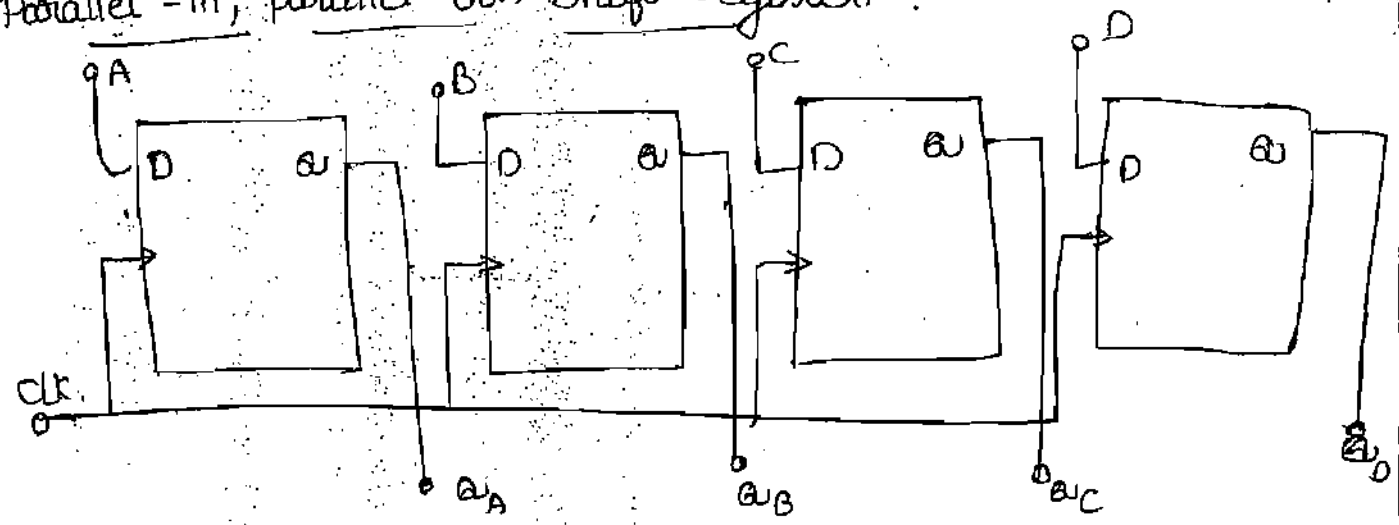


logic diagram for serial-in, parallel-out



logic symbol.

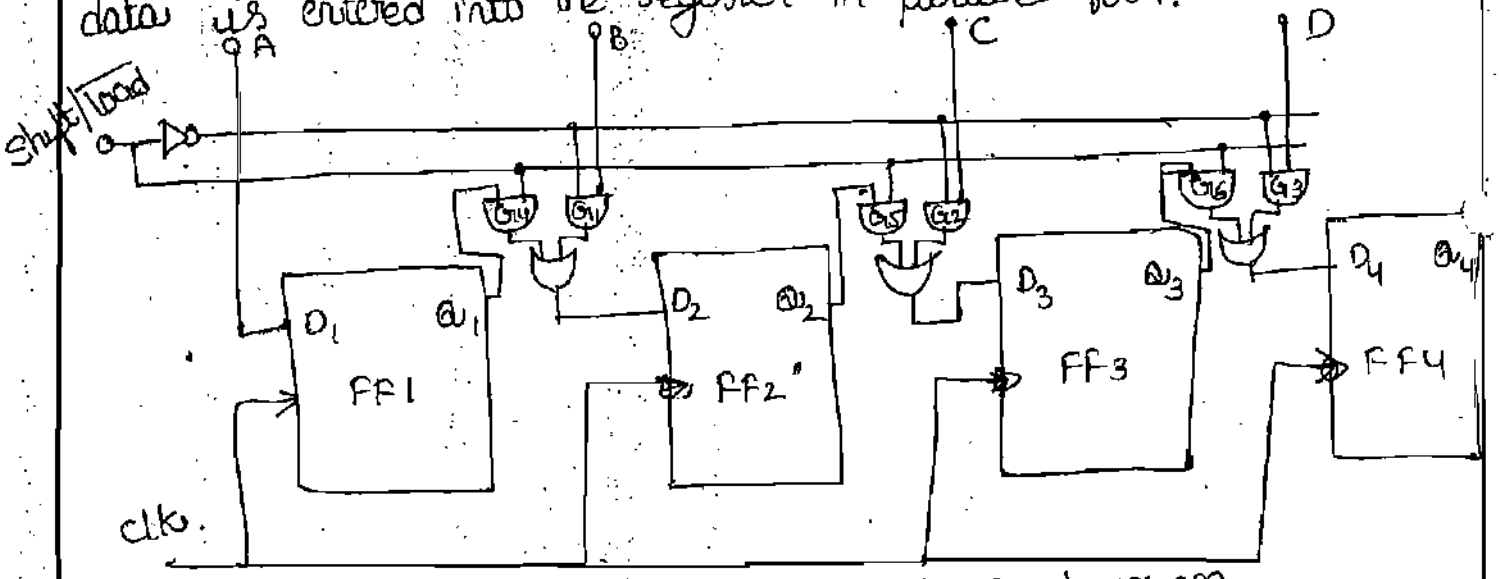
→ Parallel-in, parallel-out shift-register:-



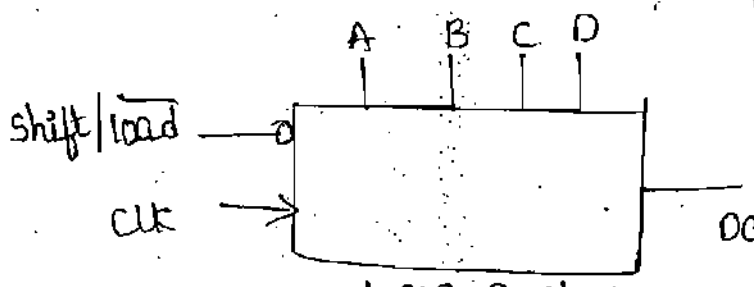
Logic diagram of a 4-bit parallel-in, parallel-out

→ 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 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.

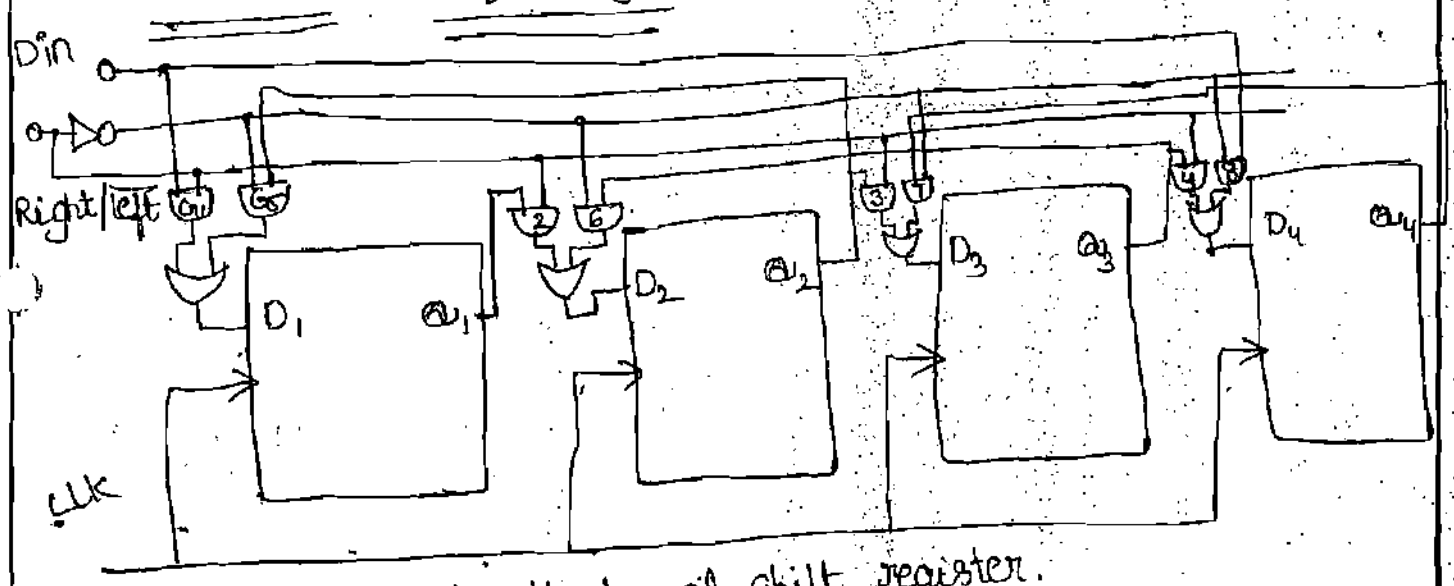
→ when shift / $\overline{Load}$ line is high, gates G_1, G_2 and G_3 are disabled, but G_4, G_5, G_6 are enabled allowing the data bits to shift right from one-stage to the next.

→ 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.

→ when Right / $\overline{\text{left}}$ is a 1, the logic circuit works as a shift - right shift register.

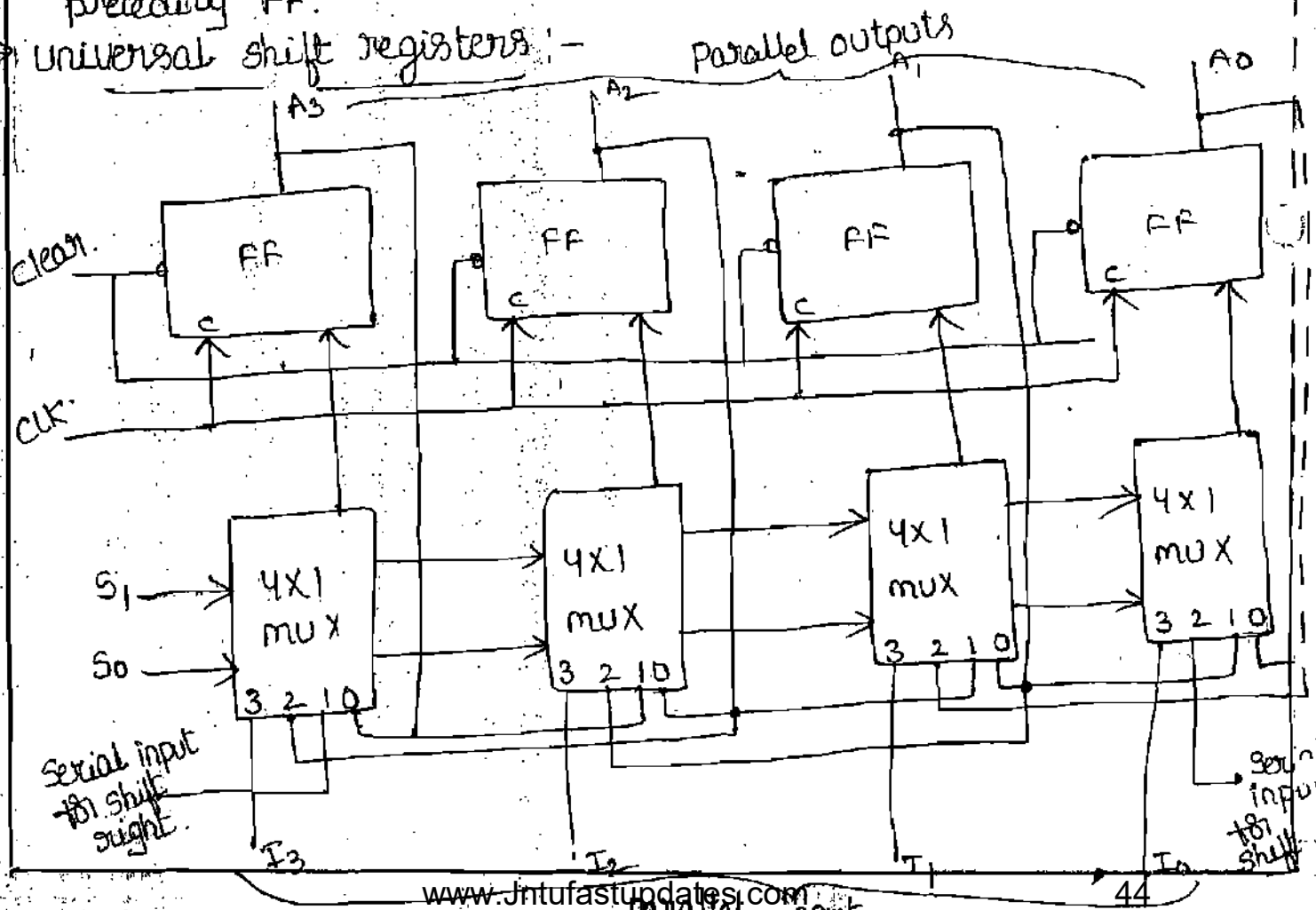
→ when Right / $\overline{\text{left}}$ is a 0, the logic circuit works as a shift - left shift register.

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

→ when mode signal Right / $\overline{\text{left}}$ is a '1' the AND Gates. G_1, G_2, G_3 and G_4 are enabled and disables 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 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.

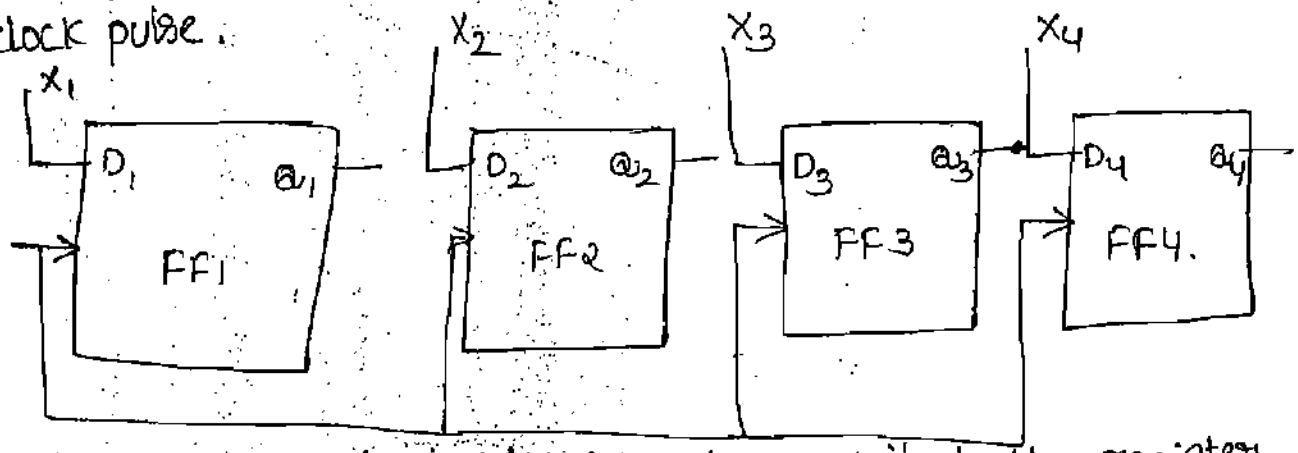
→ Finally $S_1S_0=11$ the binary information on the parallel

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.

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

$$Q = X.$$

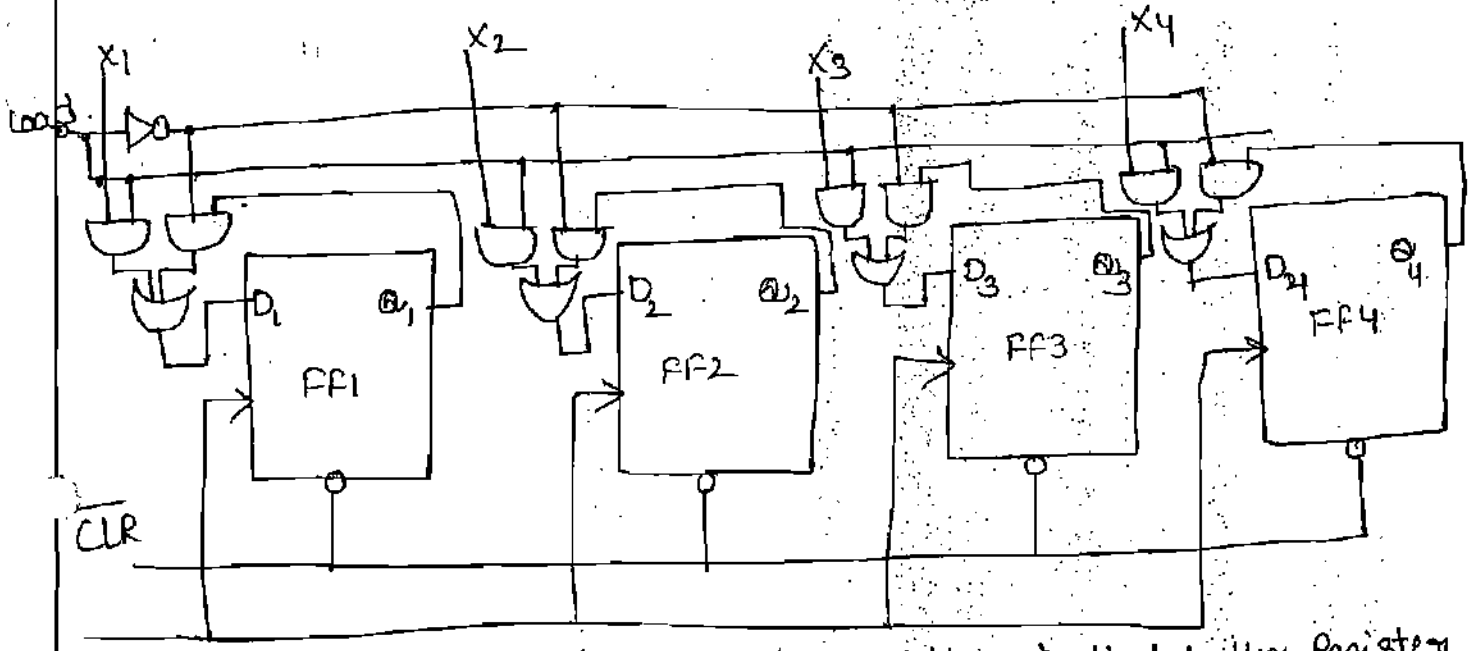
Controlled buffer Register :-

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 registers can be built by adding more FFs.

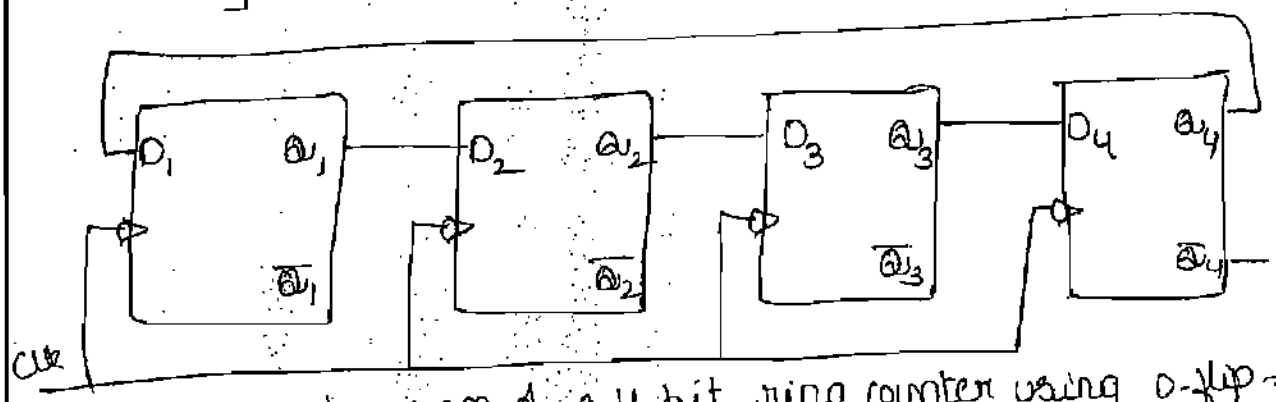
→ Shift Register counters :-

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

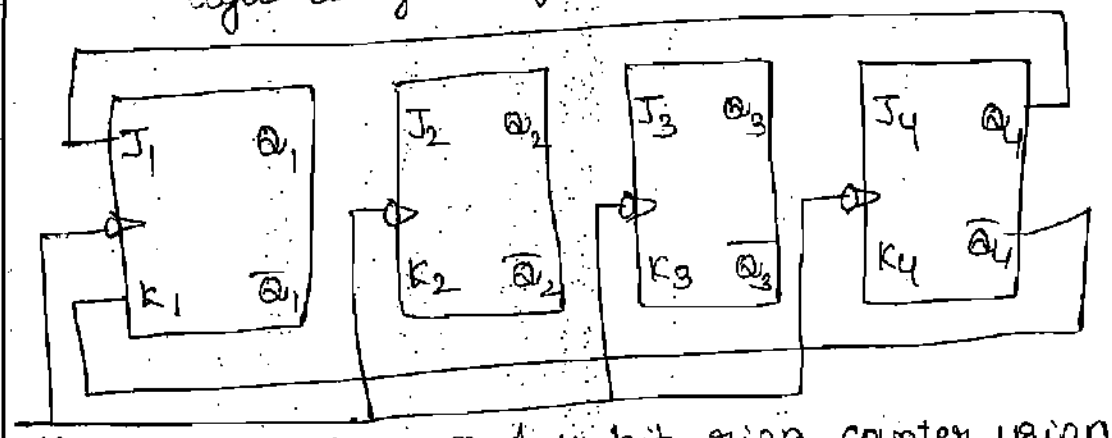
counter is the ring counter (simple ring counter)
twisted ring counter (Johnson counter or the switch-tail

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 rind and, therefore, the name ring counter..



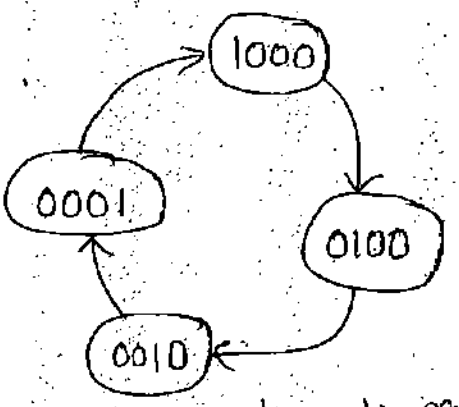
logic diagram of a 4-bit ring counter using D-flip-flops.



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

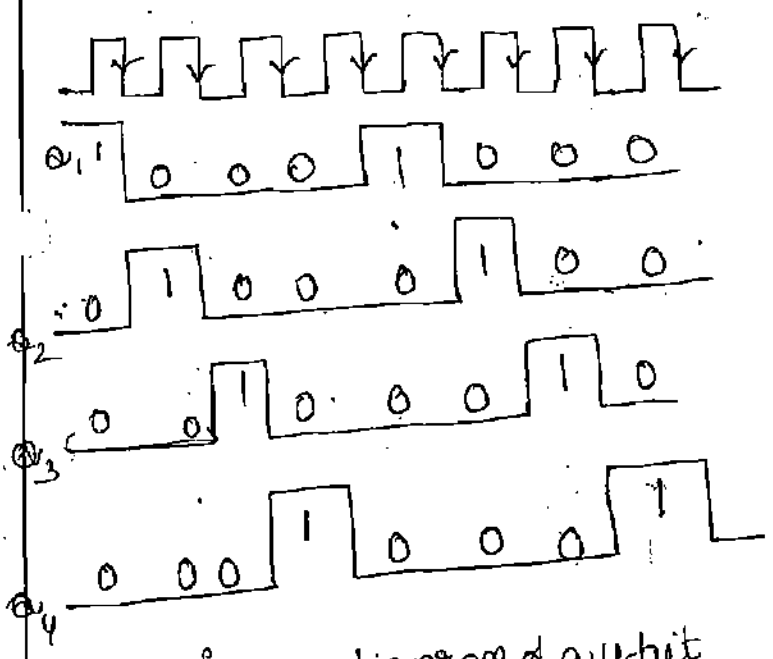
In most instances, only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially, the first FF is present to a 1. So, the initial state is 1000, that is $Q_1 = 1$, $Q_2 = 0$, $Q_3 = 0$

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.



State - diagram.

~~Twisted Ring Counter~~ :-



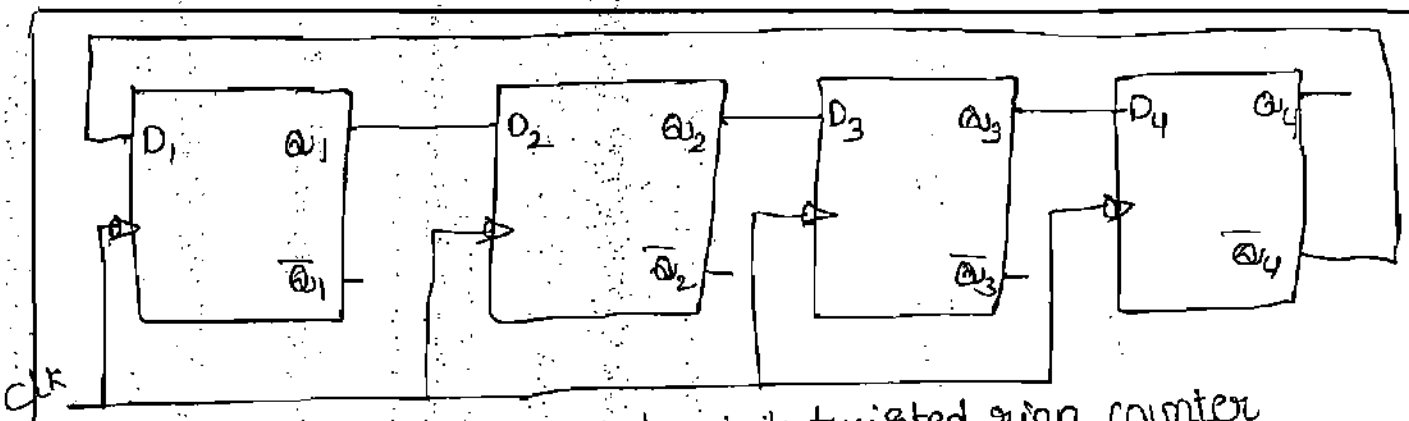
Timing diagram of a 4-bit ring counter.

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.

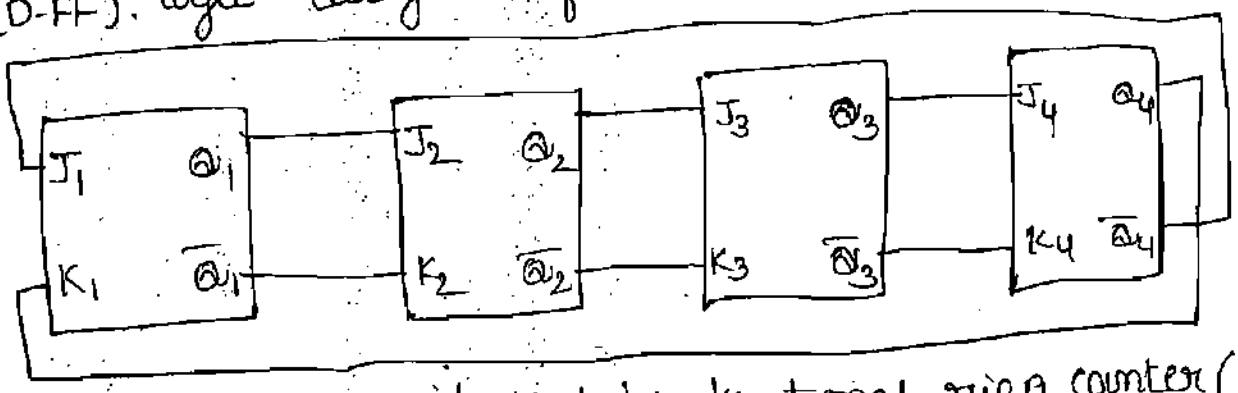
Sequence table.

Twisted Ring counter :-

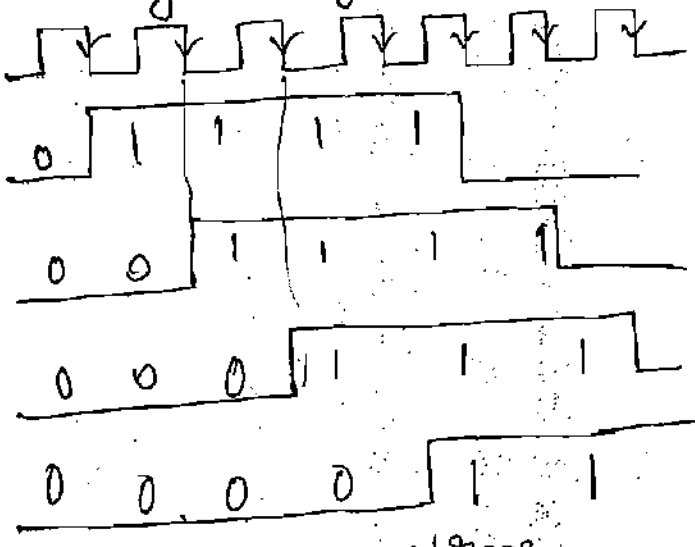
The counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. The Q output of each stage is connected to the D input of the next stage, but the $\bar{Q}$ output of the last stage is connected to the D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



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



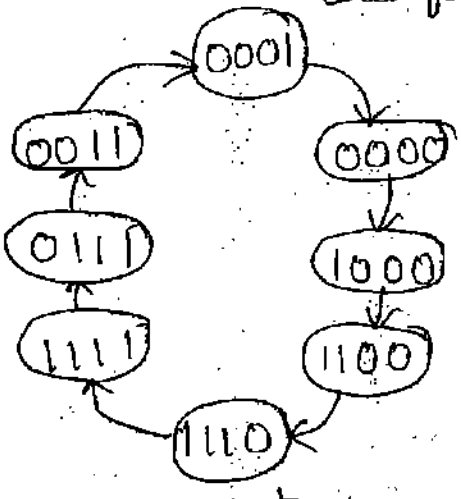
logic - diagram of 4-bit bi-directional ring counter (J-K FF)



waveforms

Q_1	Q_2	Q_3	Q_4	After clk pulse
0	0	0	0	0
1	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

Let 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 :-

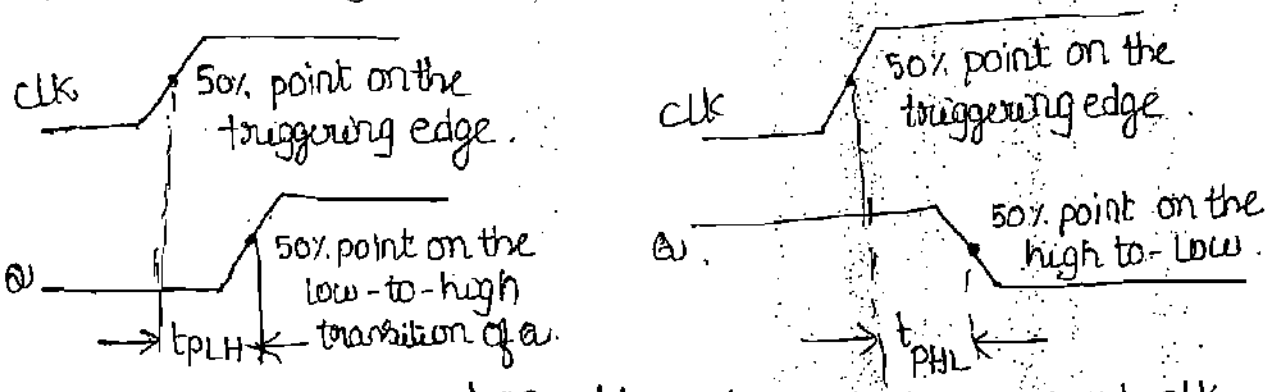
1. Time delays
2. serial / parallel data conversion
3. Ring counters
4. universal Asynchronous receiver transmitter.

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.

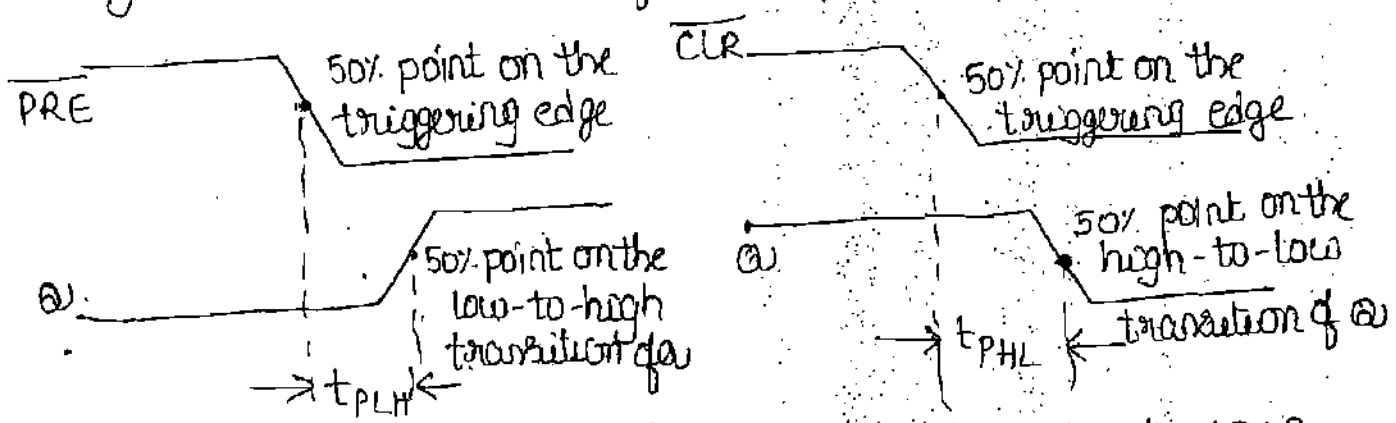
→ propagation delay t_{PHL} measured from the triggering of the clock pulse to the high-to-low transistion of the output.



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



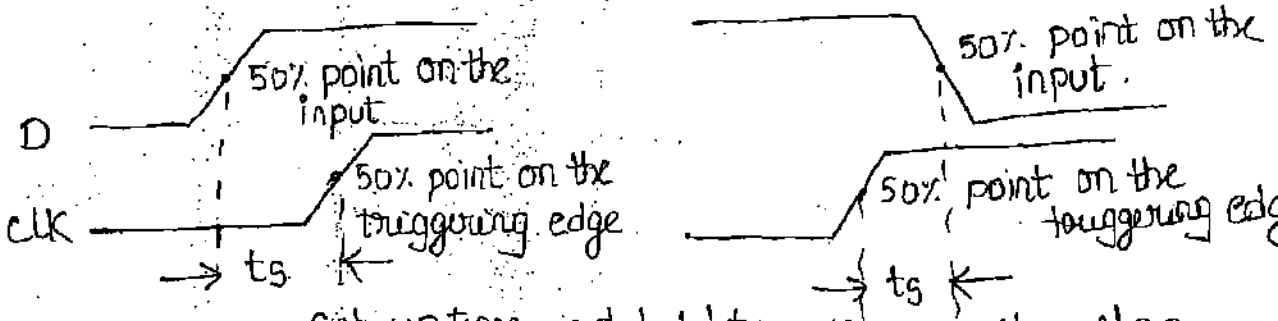
propagation delays t_{PLH}, t_{PHL} w.r.t. PRESET and CLEAR

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

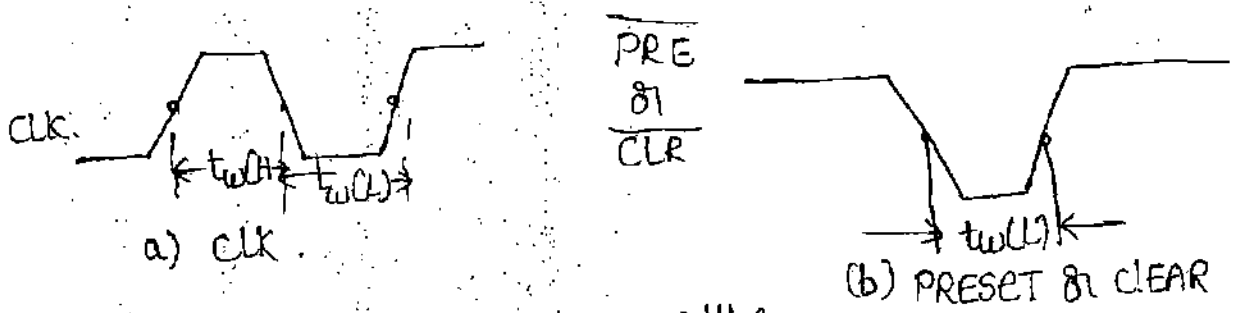
hold time :- The hold time (t_h) is the minimum time for which the control signals need to be maintained constant at the input terminals of the flip flop, after the arrival of the triggering edge of the clock pulse.

maximum clock frequency :- The maximum clock frequency (f_{max}) 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 f_{max} 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 $t_w(H)$ and the minimum low time $t_w(L)$ are specified and for asynchronous inputs.



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



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.

$$P_{TOT} = N \cdot V_{CC} \cdot I_{CC}$$

$$= (N \cdot P) \text{ mw}$$

MODULE-IV:

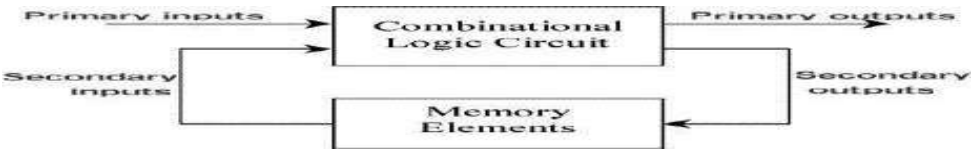
Sequential Logic Circuits - I

Sequential circuits

Classification of sequential circuits: Sequential circuits may be classified as two types.

1. **Synchronous sequential circuits**
2. **Asynchronous sequential circuits**

Combinational logic refers to circuits whose output is strictly depended on the present value of the inputs. As soon as inputs are changed, the information about the previous inputs is lost, that is, combinational logics circuits have no memory. Although every digital system is likely to have combinational circuits, most systems encountered in practice also include memory elements, which require that the system be described in terms of sequential logic. Circuits whose output depends not only on the present input value but also the past input value are known as sequential logic circuits. The mathematical model of a sequential circuit is usually referred to as a sequential machine.



Comparison between combinational and sequential circuits

Combinational circuit	Sequential circuit
1. In combinational circuits, the output variables at any instant of time are dependent only on the present input variables	1. in sequential circuits the output variables at any instant of time are dependent not only on the present input variables, but also on the present state
2.memory unit is not requires in combinational circuit	2.memory unit is required to store the past history of the input variables
3. these circuits are faster because the delay between the i/p and o/p due to propagation delay of gates only	3. sequential circuits are slower than combinational circuits
4. easy to design	4. comparatively hard to design

Level mode and pulse mode asynchronous sequential circuits:

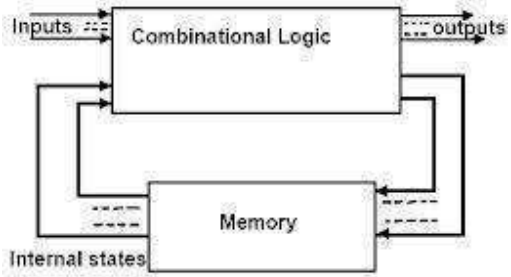


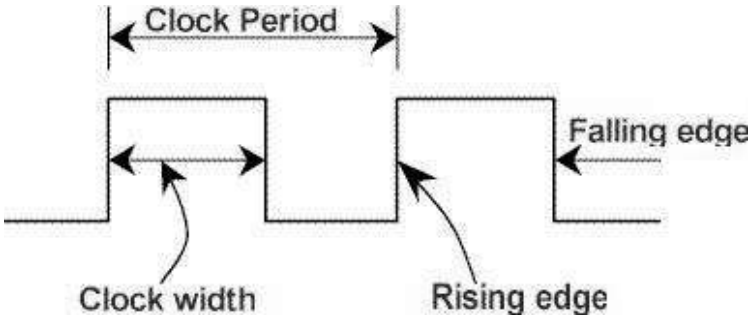
Figure 1: Asynchronous Sequential Circuit

Fig shows a block diagram of an asynchronous sequential circuit. It consists of a combinational circuit and delay elements connected to form the feedbackloops. The present state and next state variables in asynchronous sequential circuits called secondary variables and excitation variables respectively..

There are two types of asynchronous circuits: fundamental mode circuits and pulse mode circuits.

Synchronous and Asynchronous Operation:

Sequential circuits are divided into two main types: synchronous and asynchronous. Their classification depends on the timing of their signals.Synchronous sequential circuits change their states and output values at discrete instants of time, which are specified by the rising and falling edge of a free-running clock signal. The clock signal is generally some form of square wave as shown in Figure below.



From the diagram you can see that the clock period is the time between successive transitions in the same direction, that is, between two rising or two falling edges. State transitions in synchronous sequential circuits are made to take place at times when the clock is making a transition from 0 to 1 (rising edge) or from 1 to 0 (falling edge). Between successive clock pulses there is no change in the information stored in memory.

The reciprocal of the clock period is referred to as the clock frequency. The clock width is defined as the time during which the value of the clock signal is equal to 1. The ratio of the clock width and clock period is referred to as the duty cycle. A clock signal is said to

be active high if the state changes occur at the clock's rising edge or during the clock width. Otherwise, the clock is said to be active low. Synchronous sequential circuits are also known as clocked sequential circuits.

The memory elements used in synchronous sequential circuits are usually flip-flops. These circuits are binary cells capable of storing one bit of information. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the bit stored in it. Binary information can enter a flip-flop in a variety of ways, a fact which give rise to the different types of flip-flops. For information on the different types of basic flip-flop circuits and their logical properties, see the previous tutorial on flip-flops.

In asynchronous sequential circuits, the transition from one state to another is initiated by the change in the primary inputs; there is no external synchronization. The memory commonly used in asynchronous sequential circuits are time-delayed devices, usually implemented by feedback among logic gates. Thus, asynchronous sequential circuits may be regarded as combinational circuits with feedback. Because of the feedback among logic gates, asynchronous sequential circuits may, at times, become unstable due to transient conditions. The instability problem imposes many difficulties on the designer. Hence, they are not as commonly used as synchronous systems.

Fundamental Mode Circuits assumes that:

1. **The input variables change only when the circuit is stable**
2. **Only one input variable can change at a given time**
3. **Inputs are levels are not pulses**

A pulse mode circuit assumes that:

1. **The input variables are pulses instead of levels**
2. **The width of the pulses is long enough for the circuit to respond to the input**
3. **The pulse width must not be so long that is still present after the new state is reached.**

Latches and flip-flops

Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs as long as the enable signal is asserted. In other words, when they are enabled, their content changes immediately when their inputs change. Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes.

There are basically four main types of latches and flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations. In this chapter, we

will look at the operations of the various latches and flip-flops.the flip-flops has two outputs, labeled Q and Q‘. the Q output is the normal output of the flip flop and Q‘ is the inverted output.

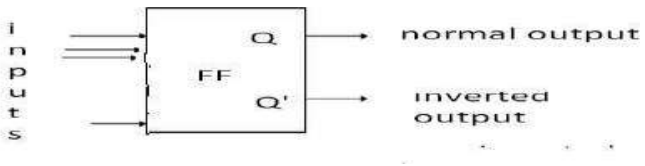


Figure: basic symbol of flipflop

A latch may be an active-high input latch or an active –LOW input latch.active –HIGH means that the SET and RESET inputs are normally resting in the low state and one of them will be pulsed high whenever we want to change latch outputs.

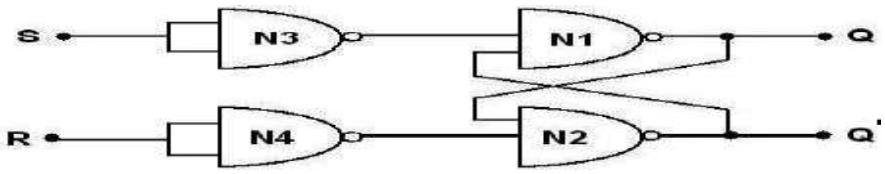
SR latch:

The latch has two outputs Q and Q‘. When the circuit is switched on the latch may enter into any state. If Q=1, then Q‘=0, which is called SET state. If Q=0, then Q‘=1, which is called RESET state. Whether the latch is in SET state or RESET state, it will continue to remain in the same state, as long as the power is not switched off. But the latch is not an useful circuit, since there is no way of entering the desired input. It is the fundamental building block in constructing flip-flops, as explained in the following sections

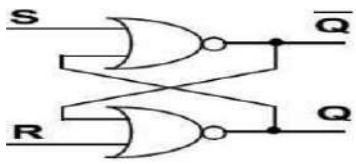
NAND latch

NAND latch is the fundamental building block in constructing a flip-flop. It has the property of holding on to any previous output, as long as it is not disturbed.

The opration of NAND latch is the reverse of the operation of NOR latch.if 0‘s are replaced by 1‘s and 1‘s are replaced by 0‘s we get the same truth table as that of the NOR latch shown



NOR latch



S	R	Q	$\overline{Q}$	Function
0	0	Q^+	$\overline{Q}^+$	Storage State
0	1	0	1	Reset
1	0	1	0	Set
1	1	0-?	0-?	Indeterminate State

The analysis of the operation of the active-HIGHNOR latch can be summarized as follows.

1. **SET=0, RESET=0: this is normal resting state of the NOR latch and it has no effect on the output state. Q and Q‘ will remain in whatever stste they were prior to the occurrence of this input condition.**
2. **SET=1, RESET=0: this will always set Q=1, where it will remain even after SET returns to 0**
3. **SET=0, RESET=1: this will always reset Q=0, where it will remain even after RESET 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=Q‘=0. If the inputs are returned to zero simultaneously, the resulting output stste is erratic and unpredictable. This input condition should not be used.**

The SET and RESET inputs are normally in the LOW state and one of them will be pulsed HIGH. Whenever we want to change the latch outputs..

RS Flip-flop:

The basic flip-flop is a one bit memory cell that gives the fundamental idea of memory device. It constructed using two NAND gates. The two NAND gates N1 andN2 are connected such that, output of N1 is connected to input of N2 and output of N2 to input of N1. These form the feedback path the inputs are S and R, and outputs are Q and Q‘. The logic diagram and the block diagram of R-S flip-flop with clocked input

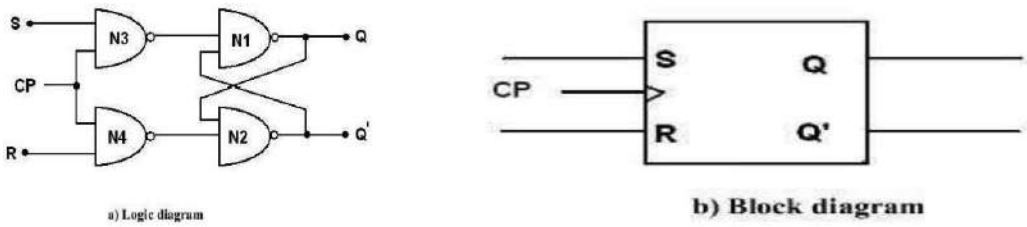


Figure: RS Flip-flop

The flip-flop can be made to respond only during the occurrence of clock pulse by adding two NAND gates to the input latch. So synchronization is achieved. i.e., flip-flops are allowed to change their states only at particular instant of time. The clock pulses are generated by a clock pulse generator. The flip-flops are affected only with the arrival of clock pulse.

Operation:

1. When CP=0 the output of N3 and N4 are 1 regardless of the value of S and R. This is given as input to N1 and N2. This makes the previous value of Q and Q‘unchanged.

2. When CP=1 the information at S and R inputs are allowed to reach the latch and change of state in flip-flop takes place.

3. CP=1, S=1, R=0 gives the SET state i.e., Q=1, Q‘=0.

4. CP=1, S=0, R=1 gives the RESET state i.e., Q=0, Q'=1.

5. CP=1, S=0, R=0 does not affect the state of flip-flop.

6. CP=1, S=1, R=1 is not allowed, because it is not able to determine the next state. This condition is said to be a —race condition‖.

In the logic symbol CP input is marked with a triangle. It indicates the circuit responds to an input change from 0 to 1. The characteristic table gives the operation conditions of flip-flop. Q(t) is the present state maintained in the flip-flop at time _t'. Q(t+1) is the state after the occurrence of clock pulse.

Truth table

S	R	$Q_{(t+1)}$	Comments
0	0	Q_t	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	*	Not allowed

Edge triggered RS flip-flop:

Some flip-flops have an RC circuit at the input next to the clock pulse. By the design of the circuit the R-C time constant is much smaller than the width of the clock pulse. So the output changes will occur only at specific level of clock pulse. The capacitor gets fully charged when clock pulse goes from low to high. This change produces a narrow positive spike. Later at the trailing edge it produces narrow negative spike. This operation is called edge triggering, as the flip-flop responds only at the changing state of clock pulse. If output transition occurs at rising edge of clock pulse (0□1), it is called positively edge triggering. If it occurs at trailing edge (1□ 0) it is called negative edge triggering. Figure shows the logic and block diagram.

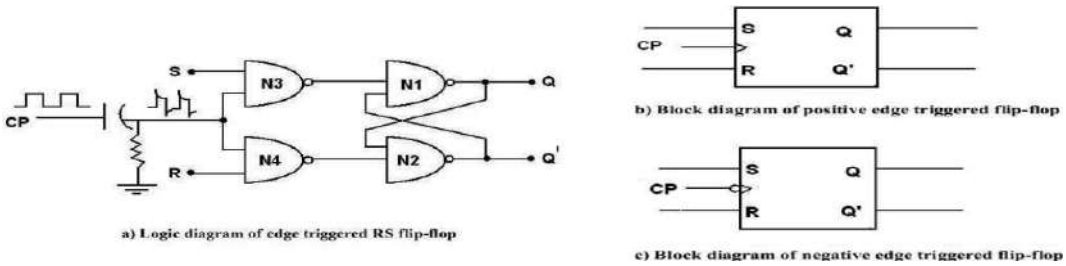
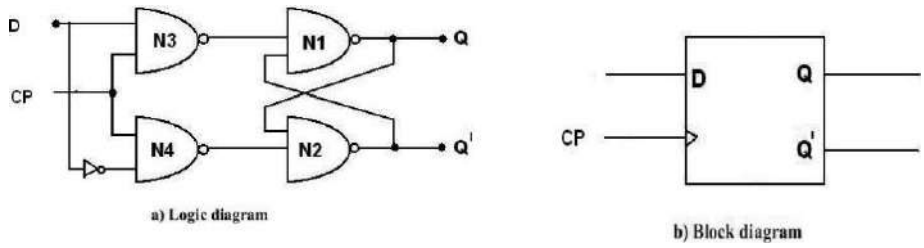


Figure: Edge triggered RS flip-flop

D flip-flop:

The D flip-flop is the modified form of R-S flip-flop. R-S flip-flop is converted to D flip-flop by adding an inverter between S and R and only one input D is taken instead of S and R. So one input is D and complement of D is given as another input. The logic diagram and the block diagram of D flip-flop with clocked input

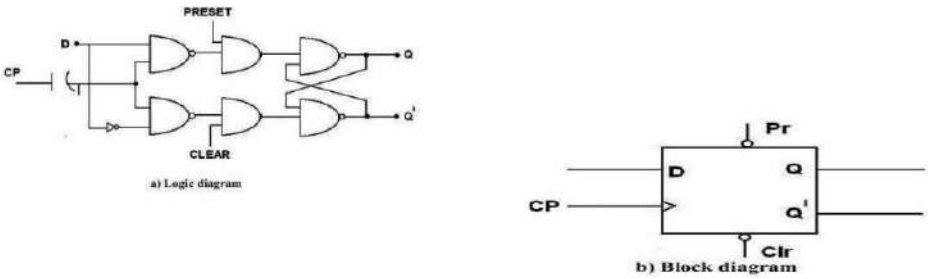


When the clock is low both the NAND gates (N1 and N2) are disabled and Q retains its last value. When clock is high both the gates are enabled and the input value at D is transferred to its output Q. D flip-flop is also called ―Data flip-flop‖.

Truth table

CP	D	Q
0	x	Previous state
1	0	0
1	1	1

Edge Triggered D Flip-flop:



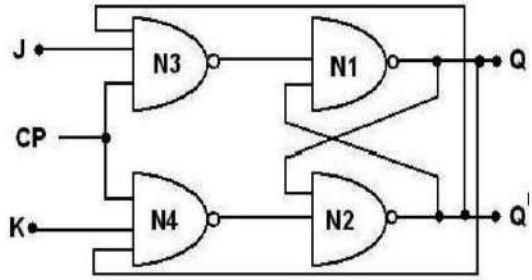
Truth table

PRESET	CLEAR	CP	D	Q
0	0	X	X	*(forbidden)
0	1	X	X	1
1	0	X	X	0
1	0	0	X	NC
1	1	1	X	NC
1	1	↓	X	NC
1	1	↑	0	0
1	1	↑	1	1

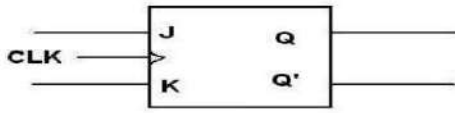
Figure: truth table, block diagram, logic diagram of edge triggered

flip-flop JK flip-flop (edge triggered JK flip-flop)

The race condition in RS flip-flop, when R=S=1 is eliminated in J-K flip-flop. There is a feedback from the output to the inputs. Figure 3.4 represents one way of building a JK flip-flop.



a) Logic diagram



b) Block diagram

Truth table

J	K	$Q_{(t+1)}$	Comments
0	0	Q_t	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	Q'_t	Complement/ toggle.

Figure: JK flip-flop

The J and K are called control inputs, because they determine what the flip-flop does when a positive clock edge arrives.

Operation:

1. When J=0, K=0 then both N3 and N4 will produce high output and the previous value of Q and Q‘ retained as it is.

2. When J=0, K=1, N3 will get an output as 1 and output of N4 depends on the value of Q. The final output is Q=0, Q‘=1 i.e., reset state

3. When J=1, K=0 the output of N4 is 1 and N3 depends on the value of Q‘. The final output is Q=1 and Q‘=0 i.e., set state

4. When J=1, K=1 it is possible to set (or) reset the flip-flop depending on the current state of output. If Q=1, Q‘=0 then N4 passes ‘0‘to N2 which produces Q‘=1, Q=0 which is reset state. When J=1, K=1, Q changes to the complement of the last state. The flip-flop is said to be in the toggle state.

The characteristic equation of the JK flip-flop is:

$$Q_{next} = J\overline{Q} + \overline{K}Q$$

JK flip-flop operation[28]

Characteristic table				Excitation table				
J	K	Q_{next}	Comment	Q	Q_{next}	J	K	Comment
0	0	Q	hold state	0	0	0	X	No change
0	1	0	reset	0	1	1	X	Set
1	0	1	set	1	0	X	1	Reset
1	1	Q	toggle	1	1	X	0	No change

T flip-flop:

If the T input is high, the T flip-flop changes state ("toggles") whenever the clock input is strobed. If the T input is low, the flip-flop holds the previous value. This behavior is described by the characteristic equation

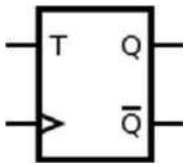


Figure : symbol for T flip flop

$Q_{next} = T \oplus Q = T\overline{Q} + \overline{T}Q$ **(expanding the XOR operator**

When T is held high, the toggle flip-flop divides the clock frequency by two; that is, if clock frequency is 4 MHz, the output frequency obtained from the flip-flop will be 2 MHz This "divide by" feature has application in various types of digital counters. A T flip-flop can also be built using a JK flip-flop (J & K pins are connected together and act as T) or D flip-flop (T input and $P_{revious}$ is connected to the D input through an XOR gate).

T flip-flop operation[28]

Characteristic table				Excitation table			
T	Q	Q_{next}	Comment	Q	Q_{next}	T	Comment
0	0	0	hold state (no clk)	0	0	0	No change
0	1	1	hold state (no clk)	1	1	0	No change
1	0	1	toggle	0	1	1	Complement
1	1	0	toggle	1	0	1	Complement

Flip flop operating characteristics:

The operation characteristics specify the performance, operating requirements, and operating limitations of the circuits. The operation characteristics mentions here apply to all flip-flops regardless of the particular form of the circuit.

Propagation Delay Time: is the interval of time required after an input signal has been applied for the resulting output change to occur.

Set-up Time: is the minimum interval required for the logic levels to be maintained constantly on the inputs (J and K, or S and R, or D) prior to the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip-flop.

Hold Time: is the minimum interval required for the logic levels to remain on the inputs after the triggering edge of the clock pulse in order for the levels to be reliably clocked into the flip-flop.

Maximum Clock Frequency: is the highest rate that a flip-flop can be reliably triggered. Power Dissipation: is the total power consumption of the device. It is equal to product of supply voltage (Vcc) and the current (Icc).

$P=V_{cc}.I_{cc}$

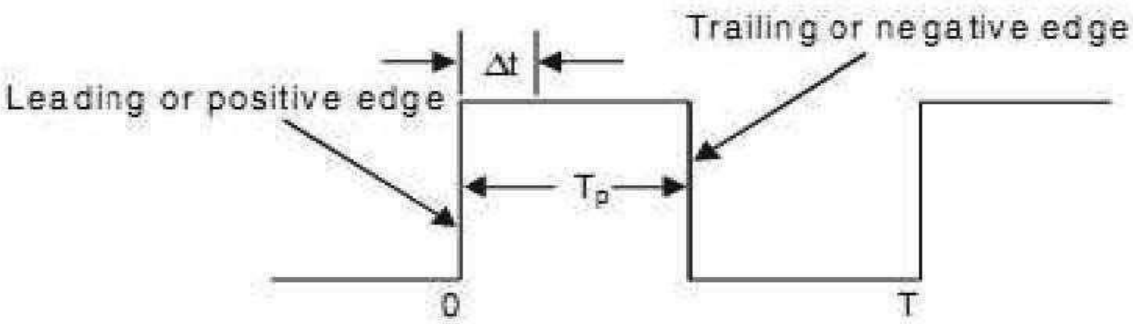
The power dissipation of a flip flop is usually in mW.

Pulse Widths: are the minimum pulse widths specified by the manufacturer for the Clock, SET and CLEAR inputs.

Clock transition times: for reliable triggering, the clock waveform transition 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 triggering erratically or not trigger at all.

Race around Condition

The inherent difficulty of an S-R flip-flop (i.e., S = R = 1) is eliminated by using the feedback connections from the outputs to the inputs of gate 1 and gate 2 as shown in Figure. Truth tables in figure were formed with the assumption that the inputs do not change during the clock pulse (CLK = 1). But the consideration is not true because of the feedback connections



Consider, for example, that the inputs are J = K = 1 and Q = 1, and a pulse as shown in Figure is applied at the clock input.

After a time interval t equal to the propagation delay through two NAND gates in series, the outputs will change to Q = 0. So now we have J = K = 1 and Q = 0.

After another time interval of t the output will change back to Q = 1. Hence, we conclude that for the time duration of tP of the clock pulse, the output will oscillate between 0 and 1. Hence, at the end of the clock pulse, the value of the output is not certain. This situation is referred to as a race-around condition.

Generally, the propagation delay of TTL gates is of the order of nanoseconds. So if the clock pulse is of the order of microseconds, then the output will change thousands of times within the clock pulse.

This race-around condition can be avoided if tp< t < T. Due to the small propagation delay of the ICs it may be difficult to satisfy the above condition.

A more practical way to avoid the problem is to use the master-slave (M-S) configuration as discussed below.

Applications of flip-flops:

Frequency Division: When a pulse waveform is applied to the clock input of a J-K flip-flop that is connected to toggle, the Q output is a square wave with half the frequency of the clock input. If more flip-flops are connected together as shown in the figure below, further division of the clock frequency can be achieved

. Parallel data storage: a group of flip-flops is called register. To store data of N bits, N flip-flops are required. Since the data is available in parallel form. When a clock pulse is applied to all flip-flops simultaneously, these bits will transfer will be transferred to the Q outputs of the flip flops.

Serial data storage: to store data of N bits available in serial form, N number of D-flip-flops is connected in cascade. The clock signal is connected to all the flip-flops. The serial data is applied to the D input terminal of the first flip-flop.

Transfer of data: data stored in flip-flops may be transferred out in a serial fashion, i.e., bit-by-bit from the output of one flip-flops or may be transferred out in parallel form.

Excitation Tables:

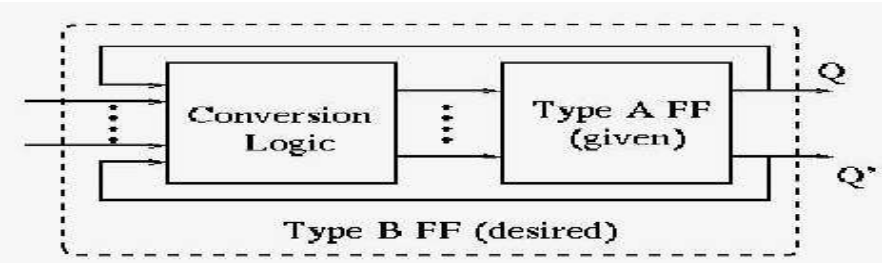
Previous State -> Present State	D
0 -> 0	0
0 -> 1	1
1 -> 0	0
1 -> 1	1

Previous State -> Present State	J	K
0 -> 0	0	X
0 -> 1	1	X
1 -> 0	X	1
1 -> 1	X	0

Previous State -> Present State	S	R
0 -> 0	0	X
0 -> 1	1	0
1 -> 0	0	1
1 -> 1	X	0

Previous State -> Present State	T
0 -> 0	0
0 -> 1	1
1 -> 0	1
1 -> 1	0

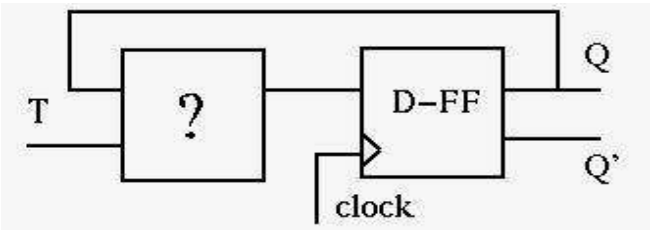
Conversions of flip-flops:



The key here is to use the excitation table, which shows the necessary triggering signal (S,R,J,K, D and T) for a desired flip-flop state transition :

Q_t	Q_{t+1}	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

Convert a D-FF to a T-FF:



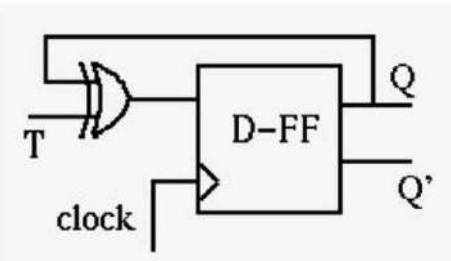
We need to design the circuit to generate the triggering signal D as a function of T and Q: . Consider the excitation table:

$D = f(T, Q).$

Q_t	Q_{t+1}	T	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1

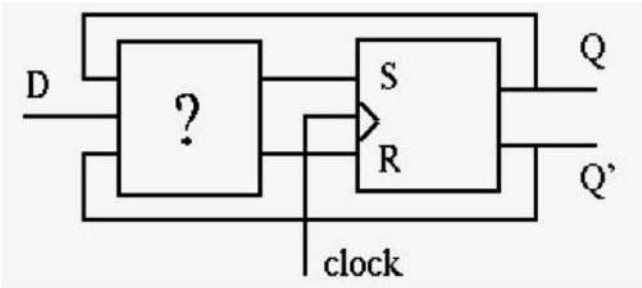
Treating as a function of and current FF state , we have

$$D = T'Q + TQ' = T \oplus Q$$



Convert a RS-FF to a D-FF:

We need to design the circuit to generate the triggering signals S and R as functions of and consider the excitation table:



Q_t	Q_{t+1}	D	S	R
0	0	0	0	x
0	1	1	1	0
1	0	0	0	1
1	1	1	x	0

The desired signal and can be obtained as functions of and current FF state from the Karnaugh maps:

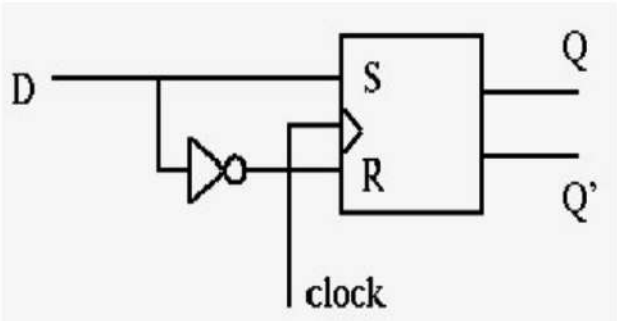
D \ Q	0	1
0	0	0
1	1	X

S=D

D \ Q	0	1
0	X	1
1	0	0

R=D'

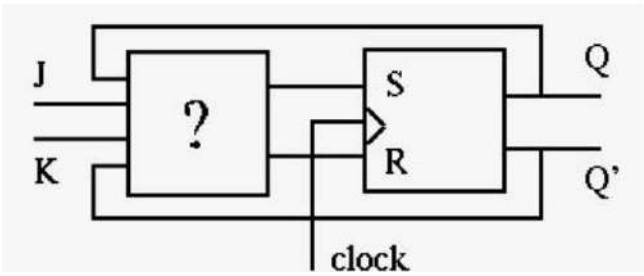
$S = D, \quad R = D'$



Convert a RS-FF to a JK-FF:

We need to design the circuit to generate the triggering signals S and R as functions of, J, K.

Consider the excitation table: The desired signal and as functions of, and current FF state can be obtained from the Karnaugh maps:



Q_t	Q_{t+1}	J	K	S	R
0	0	0	x	0	x
0	1	1	x	1	0
1	0	x	1	0	1
1	1	x	0	x	0

K-maps:

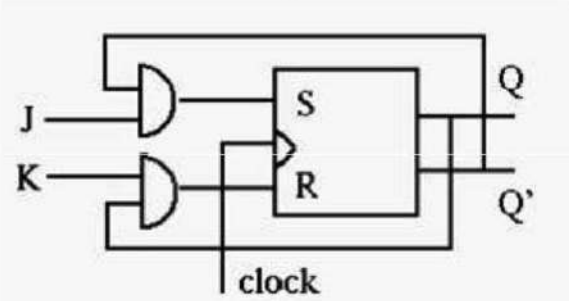
K \ QJ	00	01	11	10
0	0	1	X	X
1	0	1	0	0

S=Q'J

K \ QJ	00	01	11	10
0	X	0	0	0
1	X	0	1	1

R=QK

$$S = Q'J, \quad R = QK$$



The Master-Slave JK Flip-flop:

The Master-Slave Flip-Flop is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse. The outputs from Q and Q from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip-flop being connected to the two inputs of the "Slave" flip-flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip-flop as shown below.

The input signals J and K are connected to the gated "master" SR flip-flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1". As the clock input of the "slave" flip-flop is the inverse (complement) of the "master" clock input, the "slave" SR flip-flop does not toggle. The outputs from the "master" flip-flop are only "seen" by the gated "slave" flip-flop when the clock input goes "LOW" to logic level "0". When the clock is "LOW", the outputs from the "master" flip-flop are latched and any additional changes to its inputs are ignored. The gated "slave" flip-flop now responds to the state of its inputs passed over by the "master" section. Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip-flop are fed through to the gated inputs of the "slave" flip-flop and on the "High-to-Low" transition the same inputs are reflected on the output of the "slave" making this type of flip-flop edge or pulse-triggered. Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the Master-Slave JK Flip-flop is a "Synchronous" device as it only passes data with the timing of the clock signal.