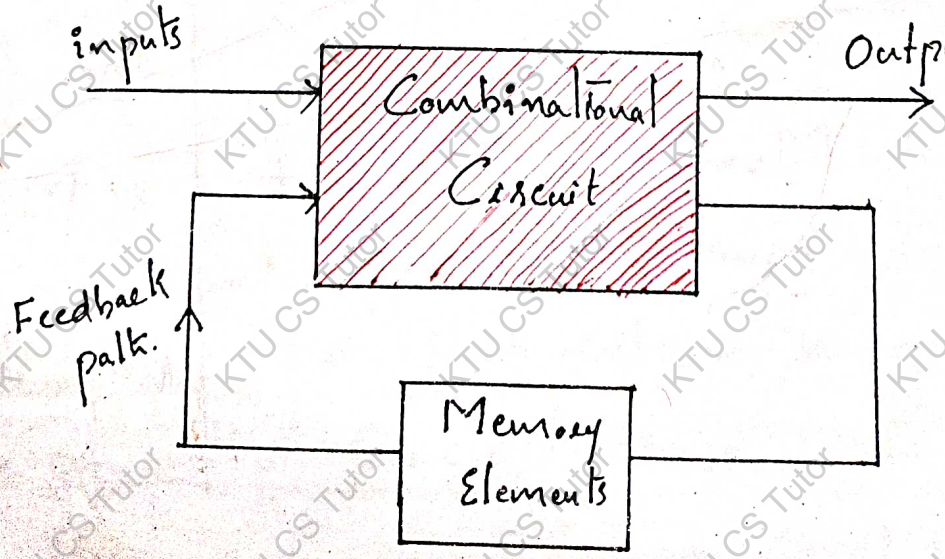
Sequential Circuits

There are many applications in which digital outputs are required to be generated in accordance with the present inputs and previous states of the circuit. This requirement cannot be satisfied using Combinational logic System. There applications require outputs to be generated depend upon the past outputs of these inputs.

The past output is provided by feedback from the outputs back to input. These circuits include the memory elements and are called as "Sequential Circu



It consists of a Combinational Circuit to which storage Elements are connected to form a feedback path.

The storage elements are devices capable of storing binary information.

Combinational Circuits	Sequential Circuits
1. The o/p variables at any instant of time are dependent only on the present input variables.	1. The output variables at any instant of time are dependent not only on the present i/p variables, but also on the previous state, on the past history of the system
2. Memory unit is not required in Combinational Circuits	2. Memory unit is required to store the past history of the i/p variables in Sequential Circuits

3. Combinational Circuits are faster because the delay between the i/p and the o/p is due to propagation delay of Gates only.	3. Sequential Circuits are slower than Combinational Circuits due to propagation delay of flipflops.
4. Combinational Circuits are easy to design.	4. Sequential Circuits are comparatively hard to design.

Classification of Sequential Circuits

The Sequential Circuits may be classified as

(a) Synchronous Sequential Circuits

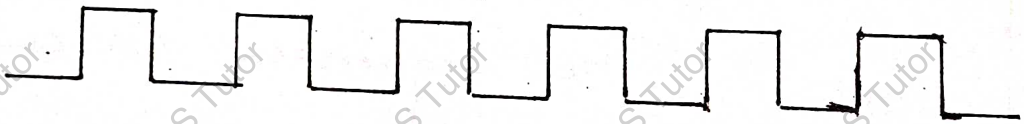
(b) Asynchronous Sequential "

The Sequential Circuits which are controlled by a clock are called Synchronous Sequential Circuits. These circuits will be active only when clock signal is present.

The Sequential Circuits which are not controlled by a clock are called asynchronous Sequential Circuits, ie the Sequential Circuit in which events can take place any time the inputs are applied are called asynchronous Sequential Circuits.

Clock:

Periodically, recurring pulse is called a clock. It is generated by a pulse Generator.



Clock Signal

Synchronous Sequential Circuits	Asynchronous Sequential Circuits
1. In Synchronous circuits memory elements are clocked FFs	1. In asynchronous Circuits, memory elements are either unclocked FF or time delay elements

2. In Synchronous Circuits, the change in input signals can affect memory elements upon activation of clock signal.	2. In asynchronous circ Change in input signals can affect memory elem at any instant of time.
3. The maximum Operating Speed of the clock depends on time-delays involved.	3. Because of the absen of the clock, asynchronou Circuits can operate fast than Synchronous Circuits
4. Easier to design	4. More difficult to des

Latch: (non clocked flip-flops)

The basic memory element is called a Latch.

As the name implies it latches on to a 1 or a 0. They are not dependent on the clock signal for their Operation.

ie a Latch is a Sequential device that checks all its inputs Continuously

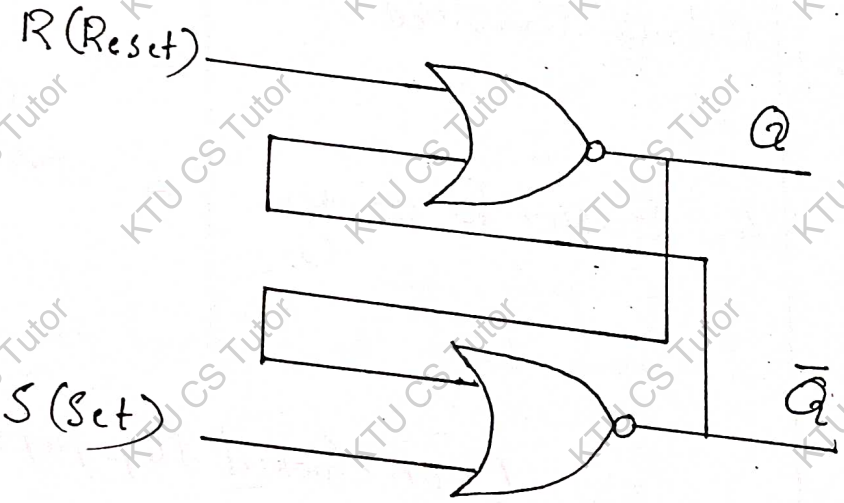
and changes its outputs accordingly at any time independent of a clock signal.

R-S Latch

The simplest latch is the Reset-Set Latch (RS-Latch)

It can be constructed from either two NOR Gates or two NAND Gates.

RS Latch using NOR Gates:



The two NOR Gates are cross coupled so that the output of NOR Gate 1 is connected to one of the inputs of NOR Gate 2 and vice versa. The latch has two Outputs Q & $\bar{Q}$ and two inputs Set (S) & Reset (R).

The state of latch corresponds to the level of Q and $\bar{Q}$.

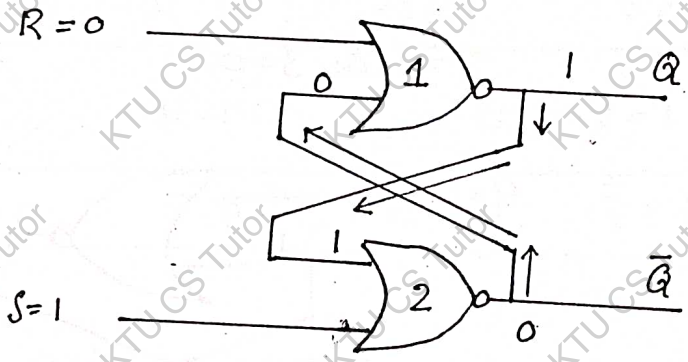
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

From the Truth table of NOR Gate, we can say If any of the inputs is a 1, output is '0'.

Operation of RS Latch

using Nor gate active high RS latch is constructed

Case(i) R=0 & S=1



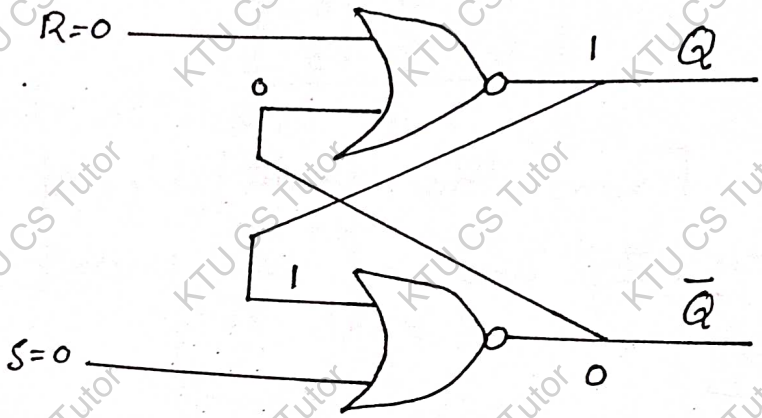
In this Case, S input of NOR Gate 2 is at logic 1., hence its Output ($\bar{Q}$) goes to logic 0, which indicates both inputs of NOR Gate 1 at logic 0. So its Output (Q) is at logic 1.

The Cycle Continues & we get Q=1 & $\bar{Q}$=0.

Case (ii) Remove S i/p

ie make S=0

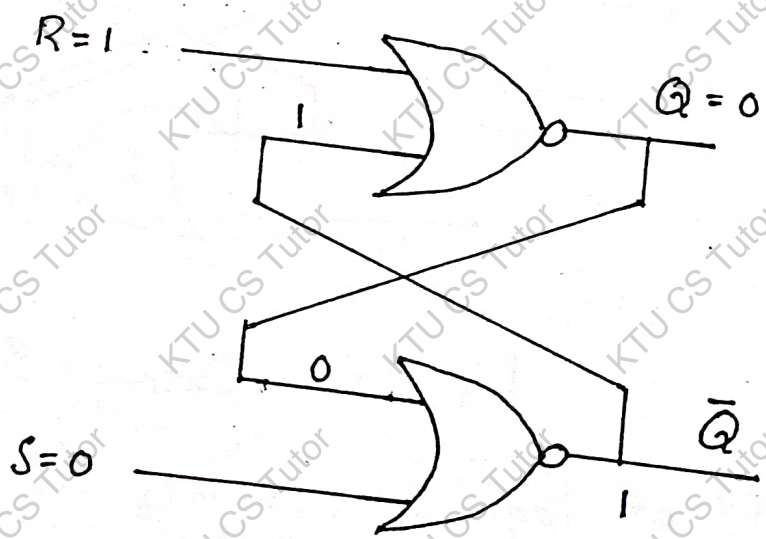
ie S=0 & R=0



Output Remains Same ie $Q=1$, $\bar{Q}=0$

Case (iii)

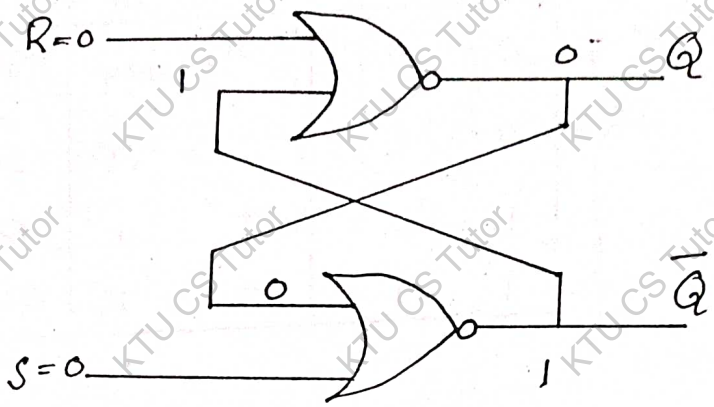
$R=1$ & $S=0$



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

Case (iv) Remove R ie R=0

S=0 & R=0



Output remains Same ie Q=0 & $\bar{Q}$=1

S=1, R=0, Q=1, $\bar{Q}$=0

S=0, R=0, Q=1, $\bar{Q}$=0 Memory

S=0, R=1, Q=0, $\bar{Q}$=1

S=0, R=0, Q=0, $\bar{Q}$=1 Memory

Case (V) S=1, R=1

When both inputs are at logic 1 the outputs of both NOR Gates goes to logic 0. ie Q=0 & $\bar{Q}$=0.

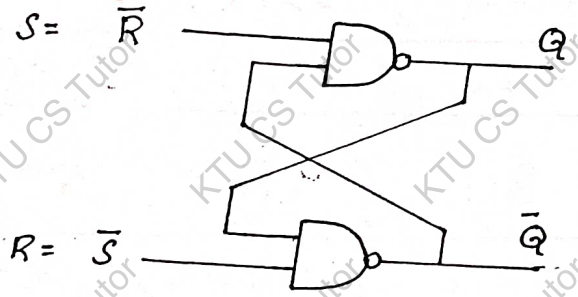
$Q \neq \bar{Q}$ for any Condition.

So this Condition is an Invalid Condition.

Input		Output		State
R	S	Q	$\bar{Q}$	
0	0	Memory		Memory → No change in previous state
0	1	1	0	set
1	0	0	1	Reset
1	1	x	x	Invalid

R-S Latch using NAND Gates or $\bar{R}\,\bar{S}$ Latch

active low



Truth Table

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

Note:- For a NAND Gate if one or more i/p is zero, o/p is 1

Case 1,

S=1, R=0, Q=0, $\bar{Q}$=1

S=1, R=1, Q=0, $\bar{Q}$=1

Memory

Case 2,

S=0, R=1, Q=1, $\bar{Q}$=0

S=1, R=1, Q=1, $\bar{Q}$=0

Case 3, S=0, R=0

Q=1, $\bar{Q}$=1 → Not possible

Truth Table

S	R	Q	$\bar{Q}$
0	0	Invalid	
0	1	1	0
1	0	0	1
1	1	Memory	

Flip flops are the ~~most~~ basic building blocks of sequential circuits
It is called ~~as~~ as binary or one bit memory.
It have generally 2 o/ps Q and $\bar{Q}$
The state of ff ~~normally~~ always refers to the state of the normal o/p Q.

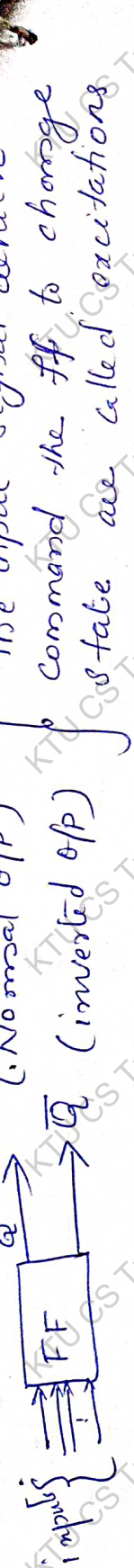
Flip-Flops

The Latch is an asynchronous transparent Sequential Circuit. ie any change in the input of Latch is transmitted immediately to the output at Q and $\bar{Q}$.

The Operation of the latch can be modified by providing an additional Control input that determines when the state of the Circuit is to be changed. This additional Control i/p is called <u>clock or clock pulse</u>.

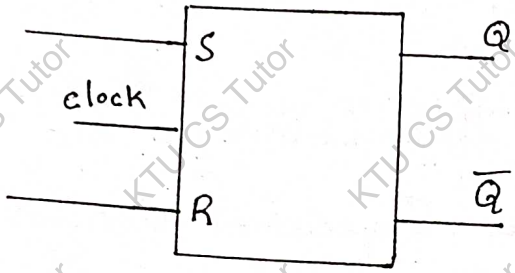
With this clock, the latch is called Flip flop

There are many types of flipfl

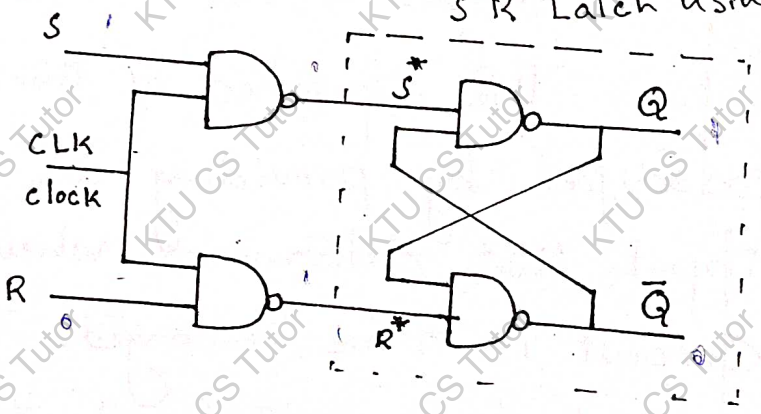


1. S R Flip flop
2. J K "
3. D "
4. T "

S R Flip Flop



S R Latch using NAND Gates



An Inverter ckt is Connected at the i/p Stage of S R latch to make it an S R Latch Using NOR Gates.

A S-R f.f requires a clock. Its S & R i/ps will control the state of the ff only when the clk is high. Otherwise the i/ps become ineffective & no change in state take place.

Case 1 : $S=1, R=0$

When $S=1$ & $R=0$, $S^* = 0$ & $R^* = 1$

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

$S=0, R=0, S^*=1, R^*=1$	Memory

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

Case 2 :

When $S=0, R=1, S^*=1$ & $R^*=0$

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

$S=0, R=0, S^*=1, R^*=1$	Memory

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

Truth Table

S	R	Q	$\bar{Q}$
0	0	Memory	(previous state)
0	1	0	1
1	0	1	0
1	1	Invalid	

Case 3 : $S=1, R=1$

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

Not - possible.

D- Flip Flop

A flip-flop whose output follows the data input when the clock is active

D → For Delay / Data

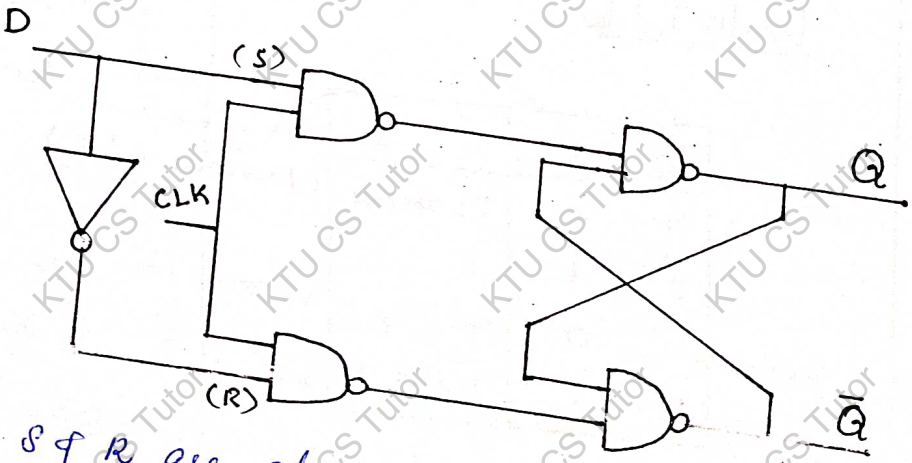
∴ it is called as a Delay Flip Flop / Data Flip Flop

Note: In a SR FF inorder store a 1/0 we have to do two things

eg: For storing a 1, S=1 & R=0

" 0, S=0 & R=1

Both can be done in single step just by connecting an Inverter between the S & R terminals



When S & R are always complement to each other then we construct a latch with a single input s and R is obtained by inverting it. This single i/p is labbled D and the device is called d latch.

D input goes directly to the S input and its complement is applied to the R-input through a NOT Gate.

∴ Only two conditions exist

When D=1, S=1 & R=0

D=0, S=0 & R=1

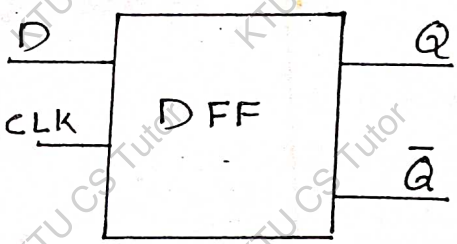
∴ during the occurance of clock pulse

if D=1, the Q output is set

D=0, o/p is reset.

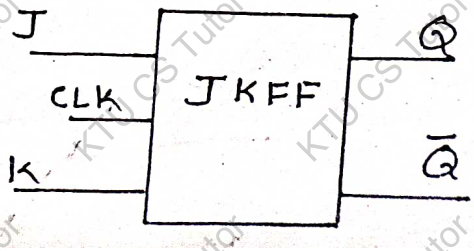
Truth Table

D	Q
0	0
1	1

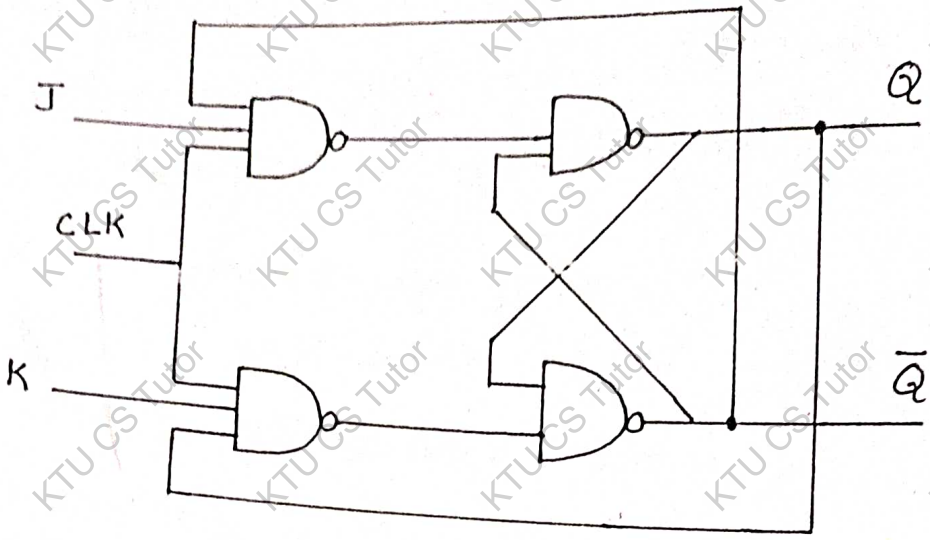


J-K Flip Flop

A J-K Flip Flop is a refinement of the S-R Flipflop. In J-K flipflop, the unpredictable state in the S-R flipflop is defined.



Logic Diagram



Note: The 1st stage which is a 2-input NAND Gates of SR Flip flop is replaced by 3-ip NAND Gates.

Case 1: J=1, K=0, Q=1, $\bar{Q}$=0

J=0, K=0, Q=1, $\bar{Q}$=0

Case 2: J=0, K=1, Q=0, $\bar{Q}$=1

J=0, K=0, Q=0, $\bar{Q}$=1

Case 3: J=1, K=1, Q changes from

1 → 0 → 1 → 0 → 1 . . .

$\bar{Q}$ → 0 → 1 → 0 → 0 . . .

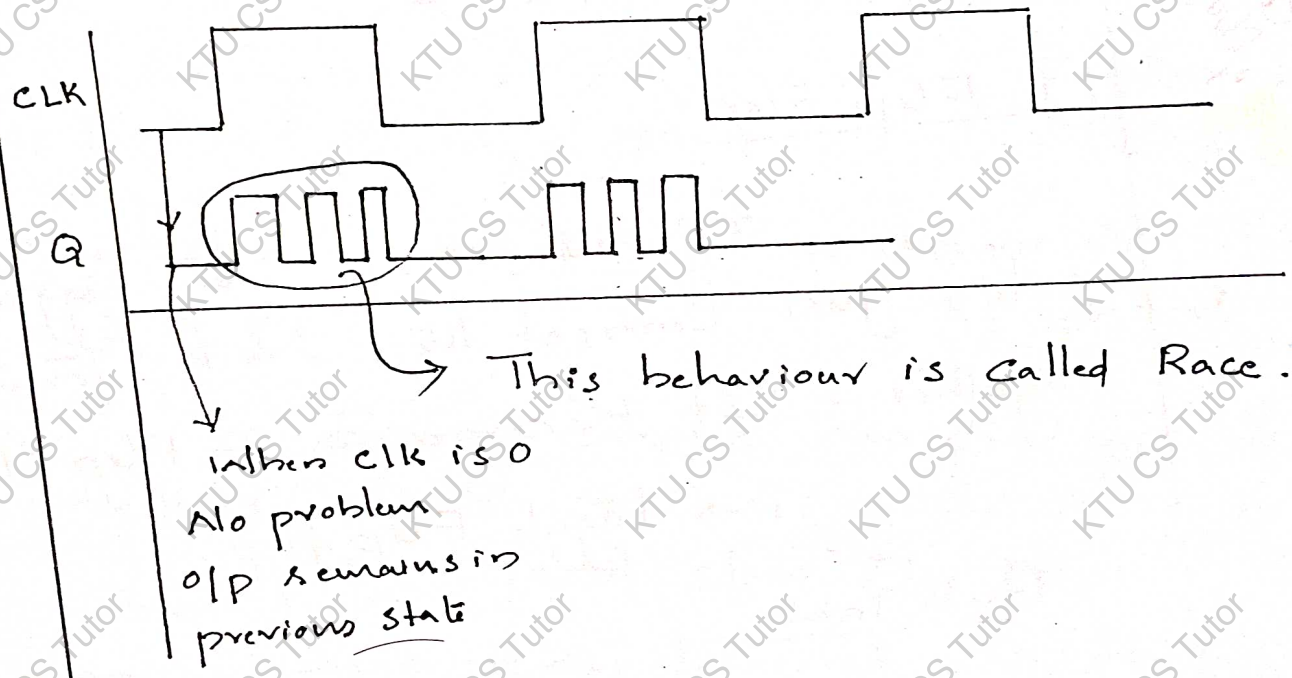
ie O/p changes its state continously when the clock is high.

This process of alternating binary states continuously is called Toggling.

Race around Condition

Race around Condition or Simply Racing is the major limitation of a J-K FF.

Racing means the FF keeps on Toggling more than once in a single clock period. So it leads to an unpredictable operation



Note: Toggling is a Controlled operation But Racing is an uncontrolled operation.

So if we can control this phenomena (racing), ie if it toggles only once in every clock cycle, then it is useful.

Truth Table

J	K	Q	$\bar{Q}$
0	0	Memory	
0	1	0	1
1	0	1	0
1	1	Toggles	

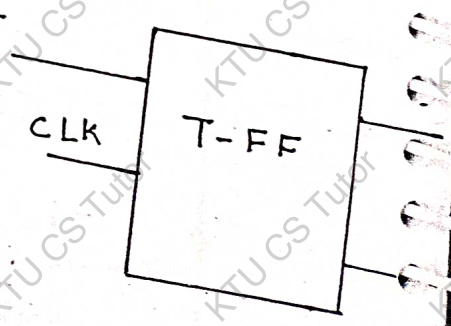
T Flip Flop

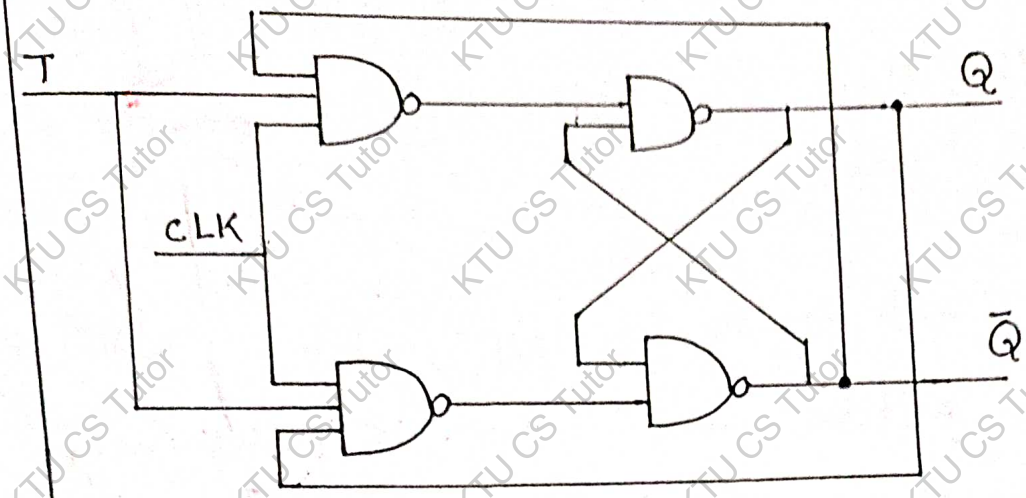
If we need the toggling action only, just connect the J & K terminals together. It becomes a T-Flip flop.

T → toggle Flip flop.

Truth Table

T	Q
0	Memory
1	Toggles





Case 1: When T=0, J=0, K=0 leads to Previous state. Memory Operation

T=1, J=1, K=1 leads to Toggling Op

How do we avoid the Race around condition in a J K FF?

Racing can be avoided by two methods

(i) Master Slave Flip flops

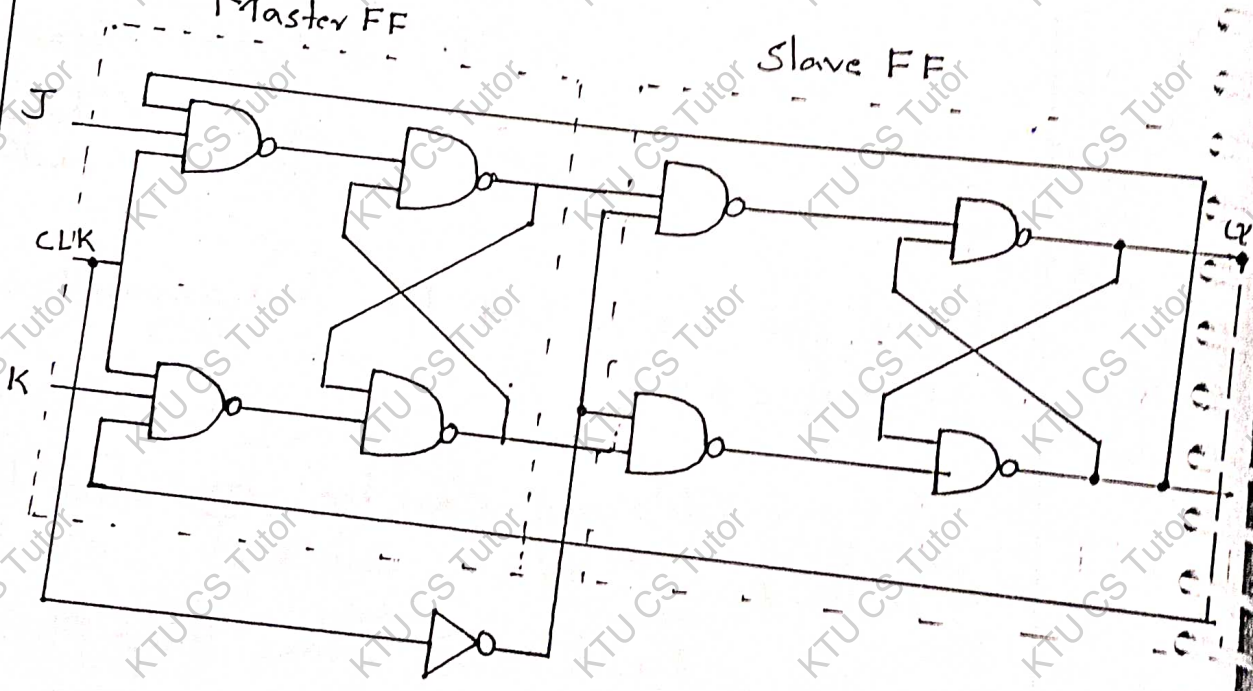
(ii) Edge Triggered Flip flops.

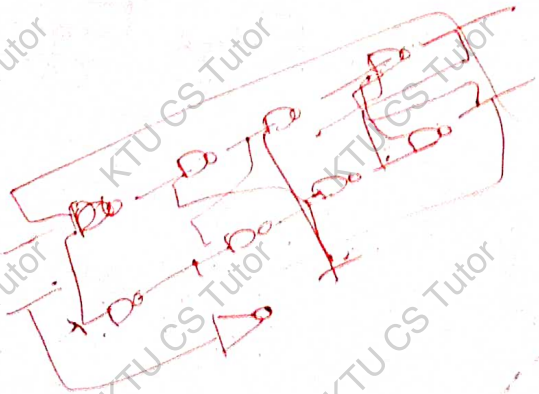
Master - Slave JK Flip flop

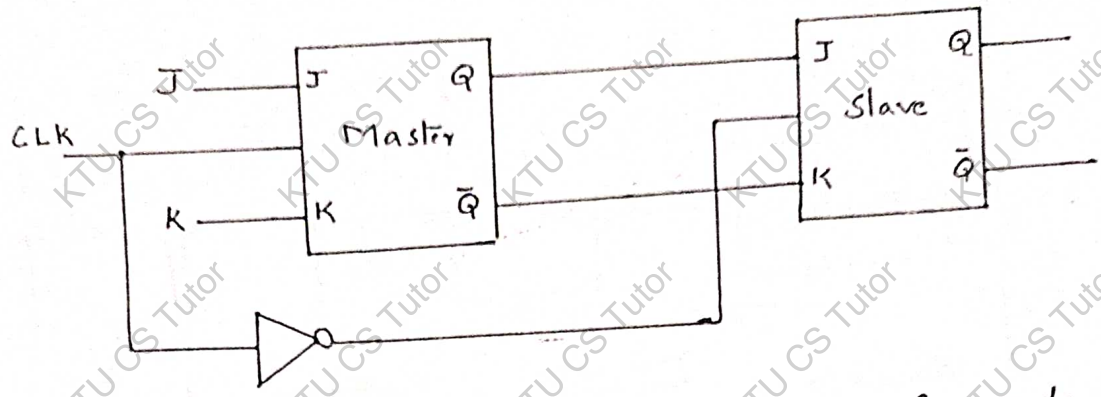
It is constructed from two flip flops. One flip flop serves as a master and the other as a slave.

If clock pulse is high, the master FF will be active and when clock pulse is low, the slave FF will be active.

Logic Diagram

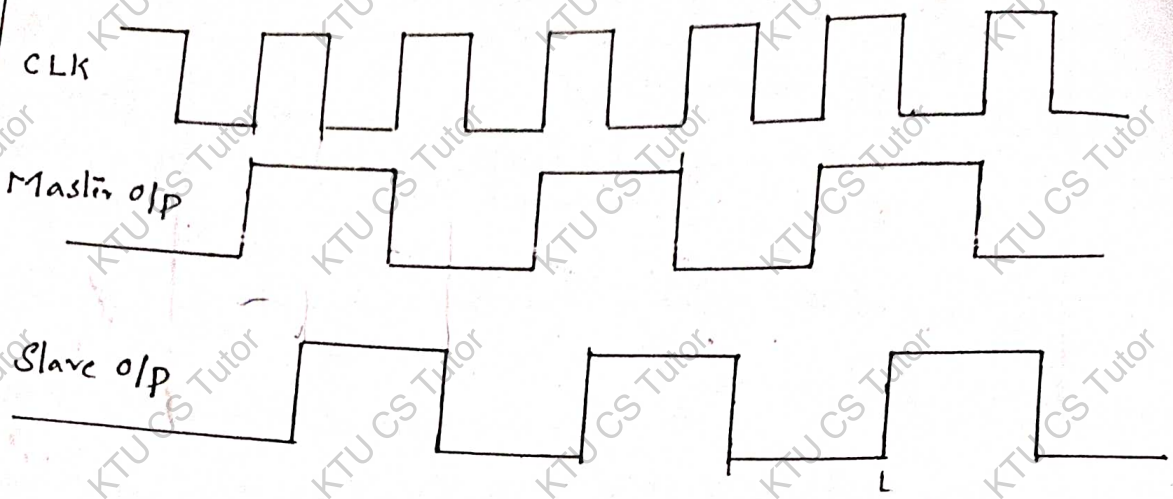






A master-slave FF is a cascade of two clocked JK Flip flops with feedback from output of the second to the i/p of the first. When the clock is high, the master is active. The output of the master is set or reset according to the state of the input. As the slave is inactive during this period its output remains in the previous state.

When clock becomes low, the output of the slave flipflop changes because it becomes active during low clock period. The final output of the master slave flip-flop is the output of the slave flip-flop. So the output of the master slave flip-flop is available at the end of a clock pulse.



Note: Inside a clock pulse the output toggles only once.

Thus racing is avoided.

(Truth Table Same as that of JK FF)

Triggering of Flipflops

The triggering method can be classified into two different types.

(1) Level Triggering (2) Edge Triggering

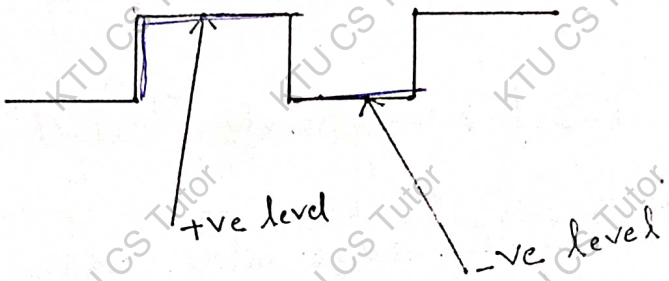
Level Triggered FFs

A FF which is sensitive to the level of a clock pulse

Level triggering can be of two types

+ve level triggering ie transition from 0 to 1

-ve level triggering ie transition from 1 to 0



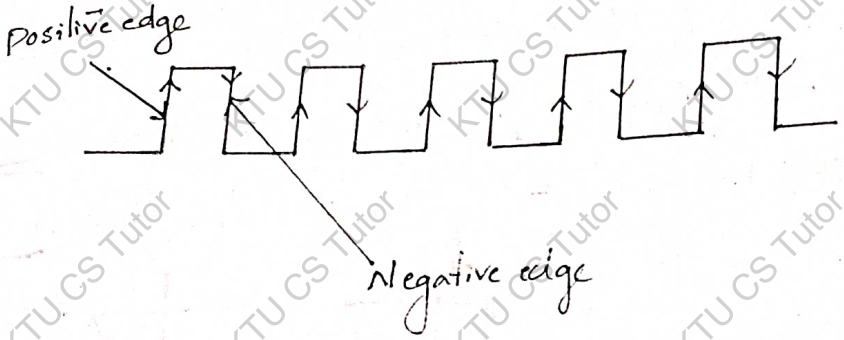
If the FF changes its states when clock is +ve level, it is termed as +ve level triggering

If the FF changes its states when the clock is negative, it is termed as -ve level triggering.

Edge Triggered Flip Flop

Edge:→ HIGH to LOW (negative edge)

LOW to HIGH (positive edge)

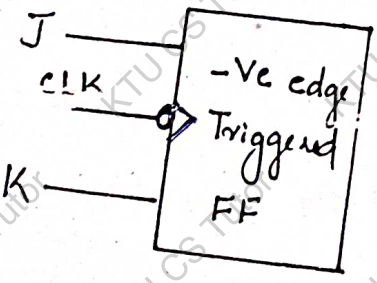


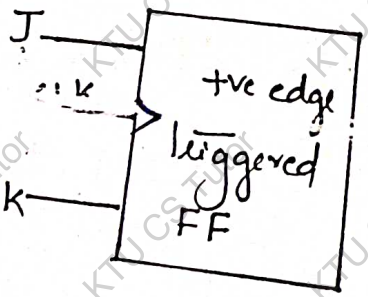
Positive edge also called as rising or leading edge.

Negative edge also called as falling or trailing edge.

Note: The term edge triggering means that the flip flop changes its state at the +ve ed. or at the -ve edge.

Representation of edge triggered FF





Note: In -ve edge triggered FF clk is represented with a Bubble

Note:

-Ve edge triggered FF has the same characteristics of a Master slave FF.

Note:

Since Edge triggered FFs changes its state only during edges, it rectifies the problem of racing

Excitation Tables:

Excitation table shows, the i/p for the required flipflop, for a particular present state and next state.

Let Q_n be the present state and Q_{n+1} be next state

S-R Flipflop

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

The required i/p condition for each of the four transition.
We need a table that list the required inputs for a given change of state.
Such list is called an excitation table.

J-K Flip flop

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

D Flip flop

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

T Flip flop

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

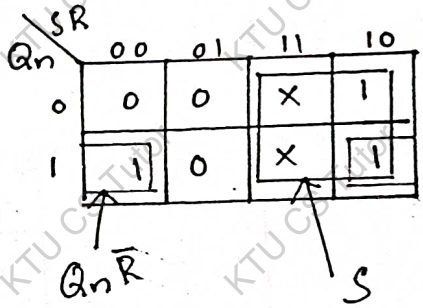
Characteristic Equation of flip flops

Characteristic Equation shows the next state, for a particular Combination of present state and i/p to the flip flops.

S-R Flip flop

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

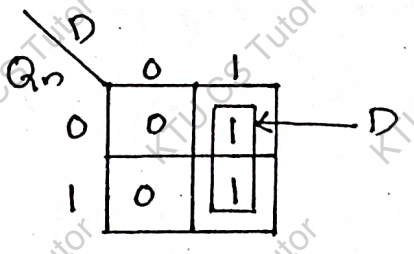
Invalid



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

D-Flip flop

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

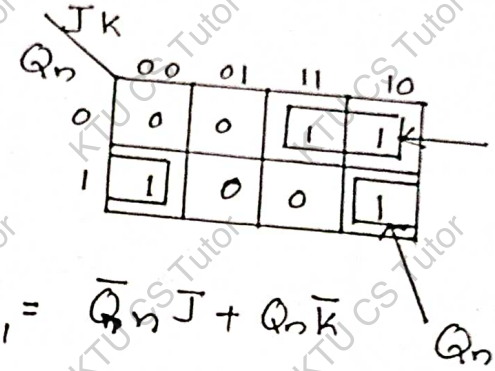


$Q_{n+1} = D$

J-K Flip flop

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

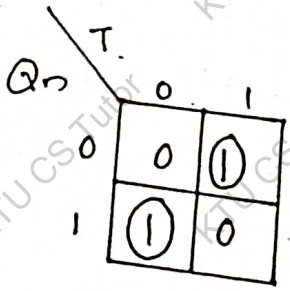
Toggles



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

T- Flip flop

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



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

$= Q_n \oplus T$

Conversion of Flipflops

One type of flipflop is converted to another type by using the following steps.

Step 1: Write the truth table of required flipf and excitation table of Given flip fl

2: Combine the above truth table and excitation table to form Conversion ta

3: Using k-maps, Simplify the logic expression for excitation inputs of Given flip flop.

4: Draw a Circuit for the desired FF using flip flop Conversion logic an the given flip flop.

Convert S-R FF to J-K FF

Write all possible Combinations of Jk (required FF) & find the next sta

Truth Table of required FF

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

For this present and Next state find the input to S-R FF (available FF)

Excitation Table of SR FF

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

S

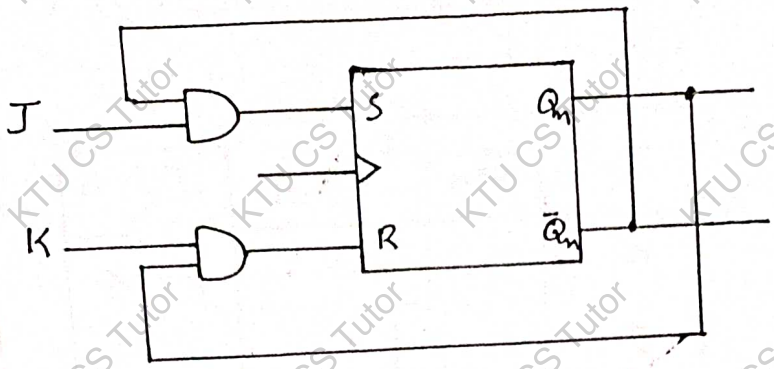
J \ KQn	00	01	11	10
0	0	x	0	0
1	1	x	0	1

$S = J\bar{Q}$

R

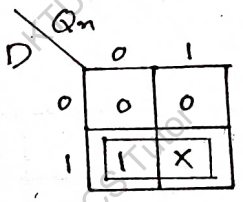
J \ KQn	00	01	11	10
0	x	0	1	x
1	0	0	1	0

$J = kQ_n$

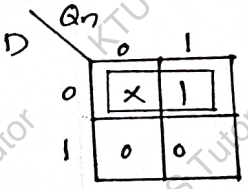


Convert S-R FF to D-FF

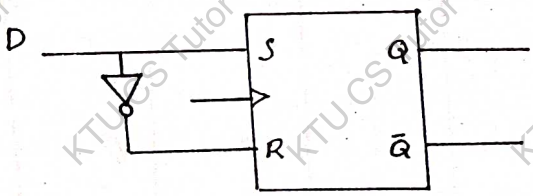
Input	Present state	Next state	Flipflop i/p	
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0



$S = D$

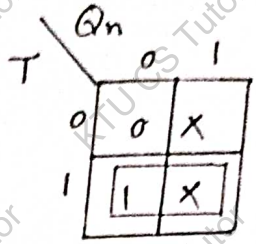


$R = \bar{D}$

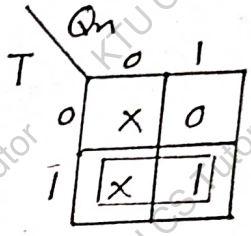


Convert J-K FF to T FF

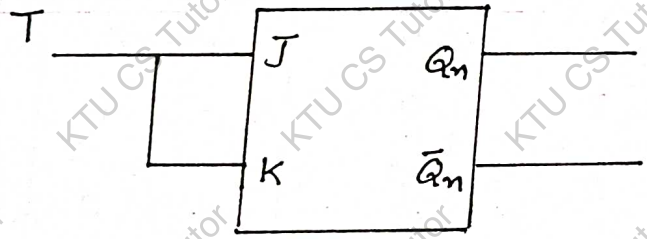
T	Q_n	Q_{n+1}	J	K
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1



J = T

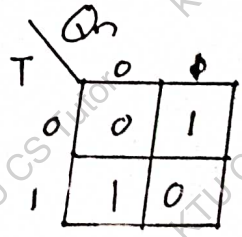


K = T

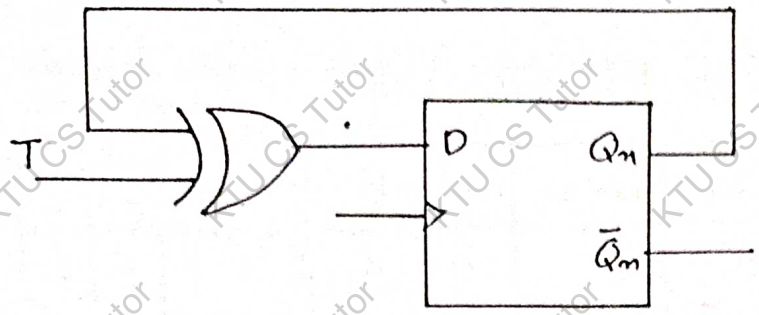


Convert D FF to T FF.

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



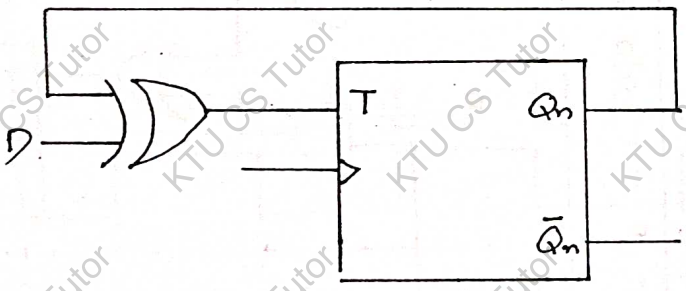
$D = T \oplus Q_n$



Convert T FF to D FF

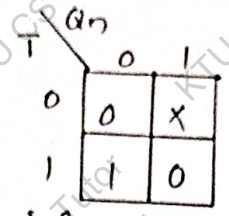
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

$T = D \oplus Q_n$

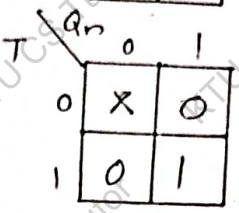


Convert S-R FF to T FF

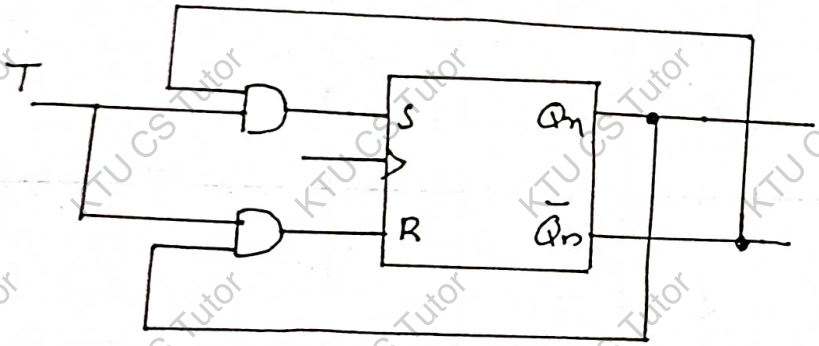
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1



$S = T\bar{Q}$

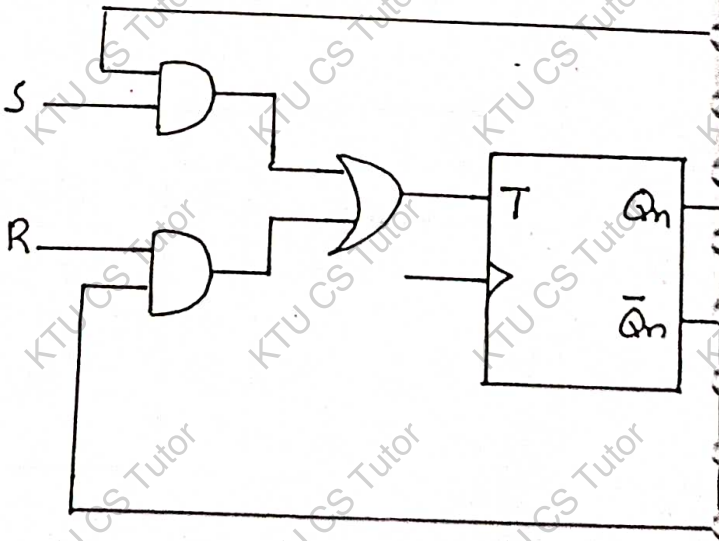


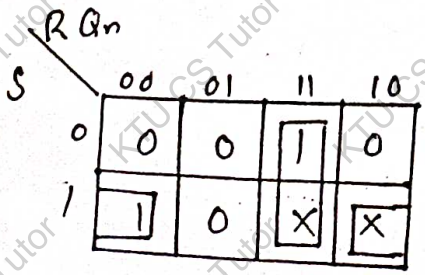
$R = TQ$



Convert T FF to S R FF

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





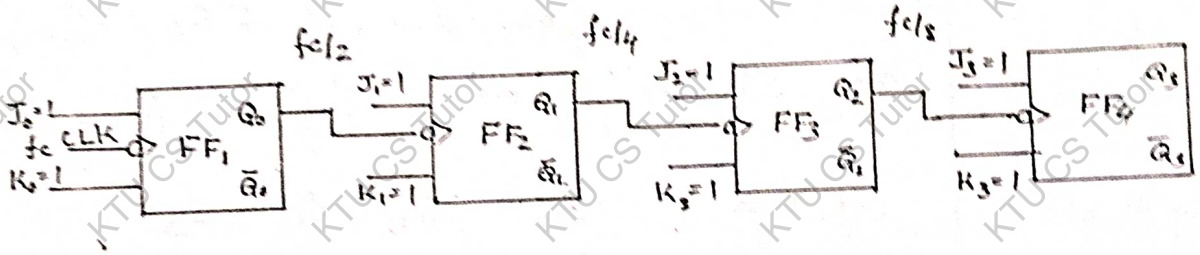
$T = S\bar{Q}_n + RQ_n$

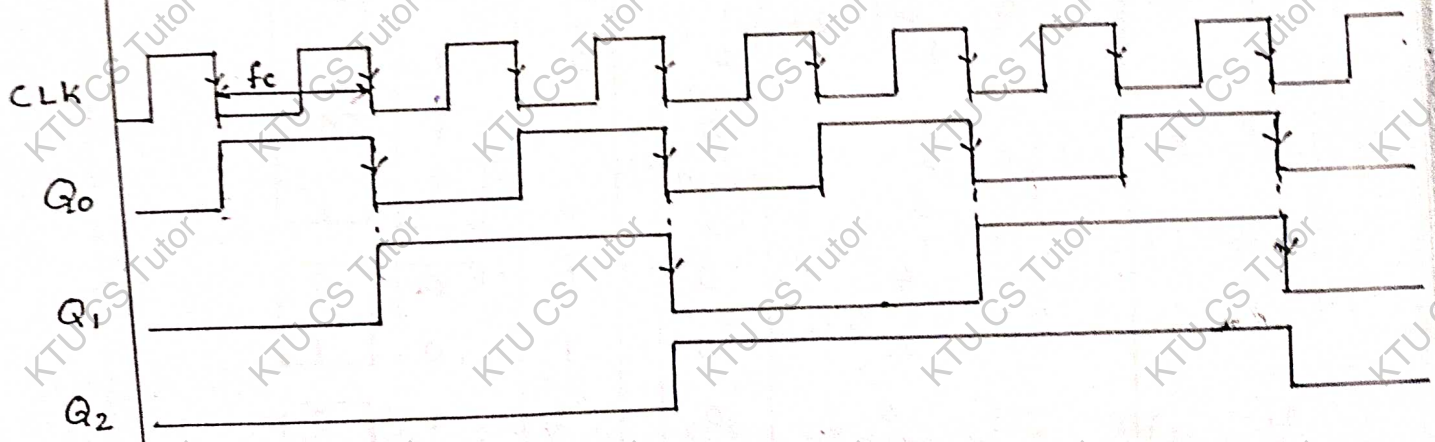
Application of Flip flops

1. Frequency division :

Flip flops may be used to divide the input signal frequency by an number. A single FF may be used to divide the i/p clock frequency by 2. Two FFs may be used divide the i/p frequency by 4.

In General, N flip-flops may be used, to divide the i/p frequency by 2^N.





Let frequency of clock signal be f_c

Then frequency of Q_0 is $f_c/2$

" Q_1 $f_c/4$

" Q_2 $f_c/8$

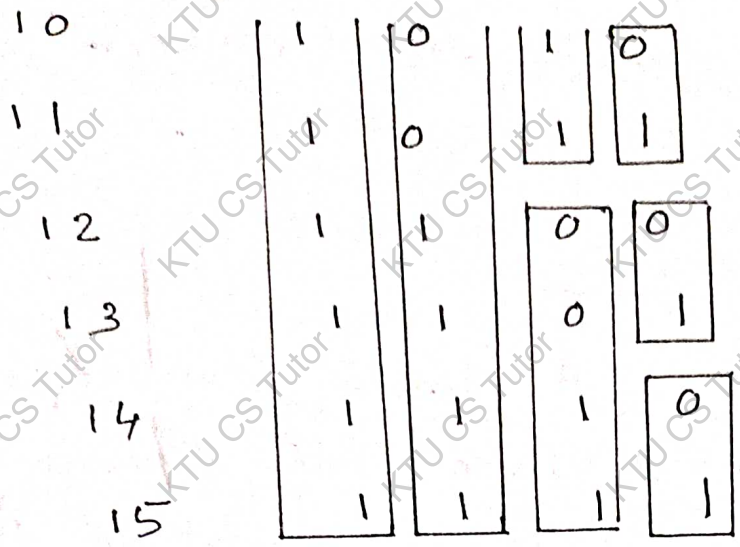
Note:-

If there are 'N' Flipflops, it divides the clock frequency by a factor of 2^N.

2. Counting:

Consider 4 FFs Connected in Seri as shown above [fig in freq division appli

CLK	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



Flip flop ○ FF_1 changes state on every negat
transition of clock pu

So it goes like 0→1→0→1→

FF_2 changes state after 2 clock
pulse

So it goes like 00→11→00→11

FF_3 changes state after 4 clock pu

So it goes like 0000→1111→
→1111-

FF_4 changes state after 8 clock pu

00000000 11111111→
00000000 11111111-

So By just checking the o/p of each FFs we can count the number of clock pulses. So this ckt can be used