MADHUBEN & BHANUBHAI PATEL INSTITUTE OF TECHNOLOGY
(A CHARUTAR VIDYAMANDAL INSTITUTION)
PO BOX NO.8, BEYOND GIDC PHASE IV, VITHAL UDYOGNAGAR – 388121, DIST. ANAND, GUJARAT, INDIA.
TELE FAX: +91 2692-230823 EMAIL: PRINCIPAL@MBICT.AC.IN WEB: WWW.MBIT.EDU.IN

Unit 3

Subject: Digital Fundamental

BY: Prof. Tejal Tandel

Sequential switching circuits

Digital logic circuits composed of components such as AND, OR and NOT gates and that do not contain loops are what we refer to as stateless. In other words, the output that the circuit produces only depends on the current inputs, and does not depend on any previous inputs. These types of circuits are also referred to as combinatorial circuits.

However, as we design computing systems, we have a need for our circuits to maintain some record of what has happened before. For example, as you type letters into your favorite editor, you want your editor to remember what you have typed. In your computer, these data are stored in a variety of memories (including your disk drive and your Random Access Memory, or RAM).

In general, a circuit that maintains a memory of the past is referred to as a sequential logic circuit. At the heart of these circuits are logic devices called latches and flip-flops that store a single bit of data until the bit is overwritten at a later (often arbitrary) time.

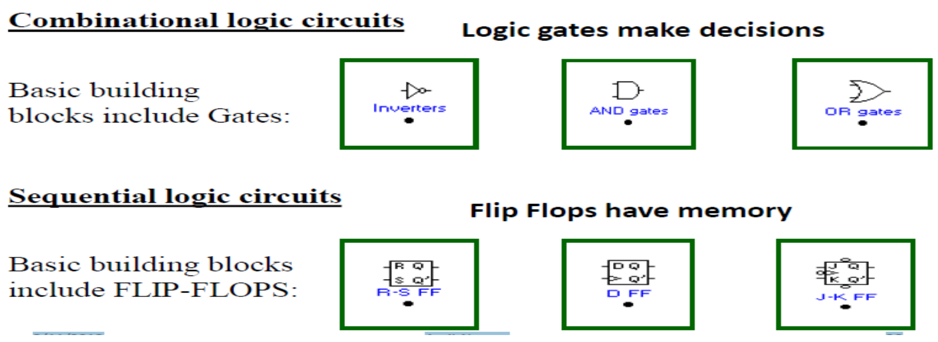
Asynchronous sequential circuit:

A sequential circuit whose behavior depends upon the sequence in which the input signals change is referred to as an ***asynchronous sequential circuit***. The output will be affected whenever the input changes. The commonly used memory elements in these circuits are time-delay devices. There is no need to wait for a clock pulse. Therefore, in general, asynchronous circuits are faster than synchronous sequential circuits. However, in an asynchronous circuit, events are allowed to occur without any synchronization. And in such a case, the system becomes unstable. Since the designs of asynchronous circuits are more tedious and difficult, their uses are rather limited. The memory elements used in sequential circuits are flip-flops which are capable of storing binary information.

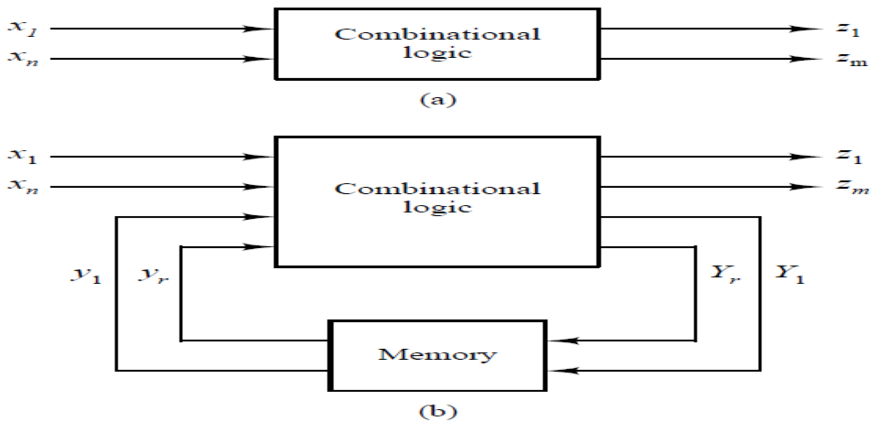
Synchronous sequential circuit:

A sequential circuit whose behavior can be defined from the knowledge of its signal at discrete instants of time is referred to as a ***synchronous sequential circuit***. In these systems, the memory elements are affected only at discrete instants of time. The synchronization is achieved by a timing device known as a system clock, which generates a periodic train of lock pulses. The outputs are affected only with the application of a clock pulse.

Logic circuits are classified into two groups:


Combinational logic circuits
Logic gates make decisions
Basic building blocks include Gates:
Inverters
AND gates
OR gates
Sequential logic circuits
Flip Flops have memory
Basic building blocks include FLIP-FLOPS:
R-S FF
D FF
J-K FF

Combinational Circuits Vs Sequential Circuits



Combinational Circuits	Sequential Circuits
In combinational circuits, the output variables at any instant of time are dependent only on the present input variables.	In sequential circuits, the output variables at any instant of time are dependent not only on the present input variables but also past output variables.
Memory unit is not required in combinational circuits.	Memory unit is required to store the past history.
Combinational circuits are faster because the delay between the input and output is due to propagation delay of gates only.	Sequential circuits are slower than combinational circuits.
Combinational circuits are easy to design	Sequential circuits are comparatively harder to design.

What is exactly Memory?
A memory should support at least three operations:

- It should be able to hold a value
- You should be able to read the value that is saved
- You should be able to change that value

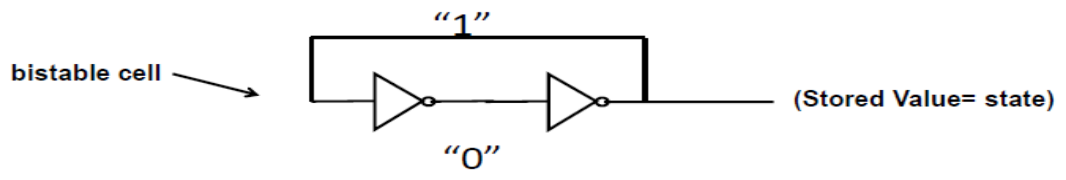
One bit memory
We will start with simplest case, a one bit memory:

- It should be able to hold a single bit, 0 or 1.
- You should be able to read the bit that is saved.
- You should be able to change the bit.

- You can set the bit to 1
- You can reset or clear the bit to 0.

Basic Idea of Storage

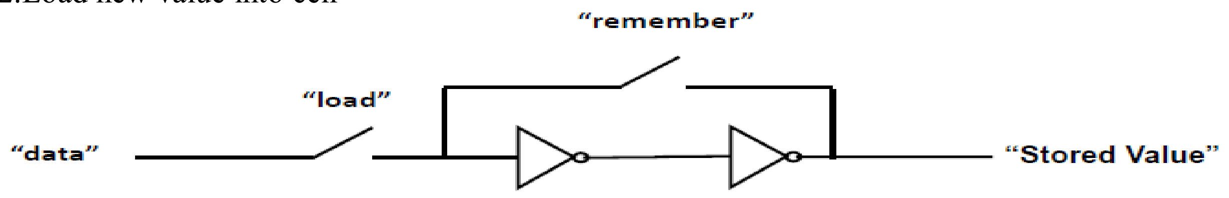
- How can a circuit remember anything, when it's just a bunch of gates that produce outputs according to inputs?
- The idea is to make a loop in a circuit, so outputs are also inputs.



- Two inverters and a feedback loop form a "Static " storage cell

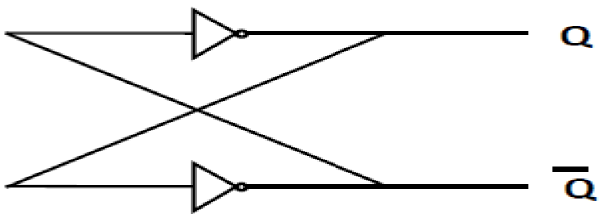
- The cell will hold value as long as it has power applied
- How to get a new value into a storage cell?

1.Selectively break feedback path
2.Load new value into cell

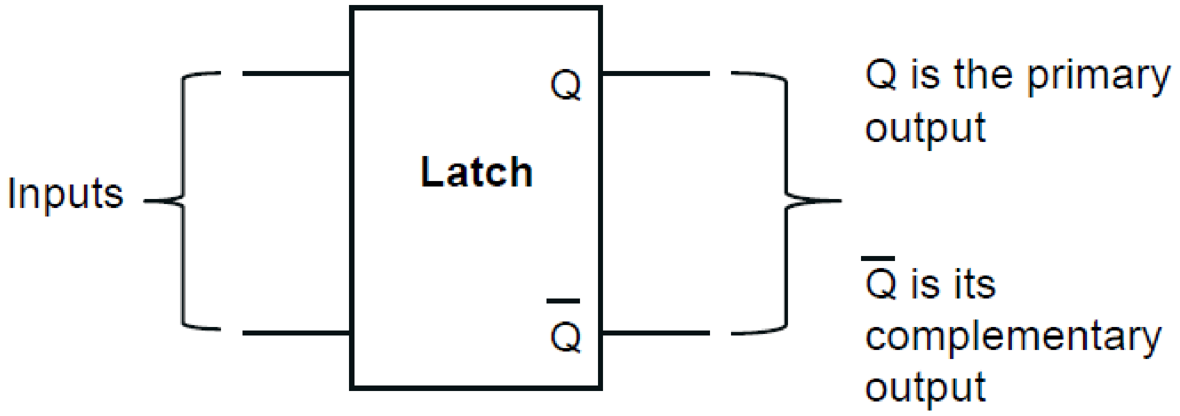


One Bit Memory Cell
The output of each gate is connected to the input of the other gate . This feedback connection is known as "Flip Flop".

- It has two stable states which are known as 1 (HIGH) state and 0 (LOW) state.
- Since flip flop has two stable states, it is called a Binary or Bistable.
- Similarly it stores 1 bit information; either 1 or 0.
- it is a 1 bit memory unit or a 1 bit storage cell.
- Since information is locked or latched, 1 bit memory cell is also known as LATCH.



Latch
Latch are the bi-stable devices which responds to the change of input logic levels as they occur.

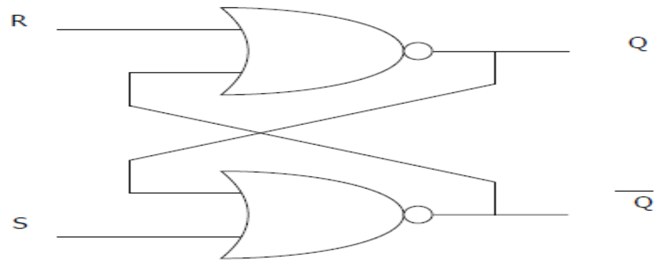


It is said to be in SET state if output Q is High
It is said to be in RESET state if output Q is Low

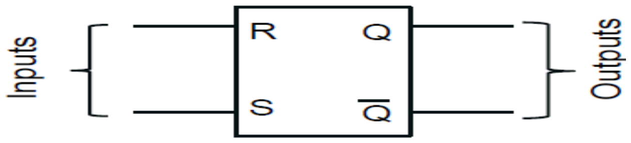
The basic difference between a latch & flip-flop is, Storage elements that operate with signal levels (rather than signal transitions) are referred to as **latches**; those controlled by a clock transition are **flip-flops**. Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices.
The two types of storage elements are related because latches are the basic circuits from which all flip-flops are constructed. Latch is used for certain flipflop which are non-clocked. These flipflop 'latch on' to a 1 or a 0 immediately upon receiving the input pulse called SET or RESET. The latch is sequential device that checks all its inputs continuously and changes its outputs accordingly at any time independent of a clock signal.

Let us assume the output of G1 to be Q = 0, which is also the input of G2 (A2 = 0). So, the output of G2 will be Q'= 1, which makes A1 = 1 and consequently Q = 0 which is according to our assumption. Similarly, we can demonstrate that if Q = 1, then Q' = 0 and this is also consistent with the circuit connections. Hence we see that Q and Q' are always complementary. And if the circuit is in 1 state, it continues to remain in this state and vice versa is also true. Since this information is locked or latched in this circuit, therefore, this circuit is also referred to as a ***latch.***

SR Latch using NOR



Circuit Diagram



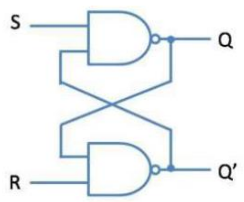
RS Latch using NOR **Logic Table**

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	Indetermine
1	1	1	X	

When the SET input is made HIGH ,Q becomes 1.when the RESET input is made HIGH, Q becomes 0. If both the inputs S and R made LOW ,there is no change in the state of the latch,,if both the inputs are made HIGH ,the output is unpredictable ,(invalid)

Figure shows the logic diagram of a active HIGH S-R latch composed of two cross- couple NOR gates. Note that the output of gate is connected to one of the inputs of the other gate.
The latch works as per the truth table of figure ,where Qn represents the state of the flip-flip before applying inputs and Q n+1 represents the state of the flip fliop after applying inputs.

NAND gate S-R latch (Active LOW)



Logic diagram

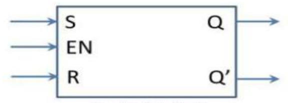
S	R	Q_n	Q_{n+1}	State
0	0	0	X	Indeterminate (Invalid)
0	0	1	X	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	No Change (NC)
1	1	1	1	

The NAND gate is equivalent to an active LOW OR gate, am active LOW S-R latch using OR gates may also be represented.
The operation of this latch is the reverse of the operation of the NOR gate latch.

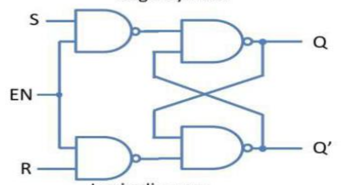
If the 0s are replaced by 1s and 1s by 0s, we get the same truth table as that of NOR gate latch.
The SET and RESET inputs are normally resting in the HIGH state and one of them will be pulsed LOW, whenever we want to change the latch outputs.

GATED LATCHES(CLOCKD FLIP-FLOPS)

S R Flip-flop (Gated S-R Latch)



Logic Symbol

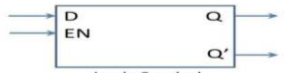


Logic diagram

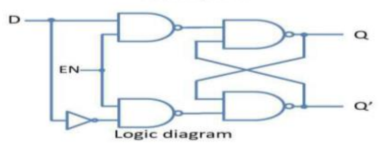
En	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No Change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Indeterminate (Invalid)
1	1	1	1	X	
0	X	X	0	0	No Change (NC)
0	X	X	1	1	

A gated S-R larch requires an ENABLE (EN) input.
Its S and R inputs will control the state of the flip-flop only when the EN is HIGH.
When EN is LOW, the inputs become ineffective and no change of state can take place.
The EN input may be a clock. So, a gated S-R latch is also called a *clocked S-R latch*.
Since this type of flip-flop responds to the changes in inputs only as long as the clock is HIGH, these types of flip-flops are also called *level triggered flip-flops*.

D Flip-flop (Gated D Latch)



Logic Symbol



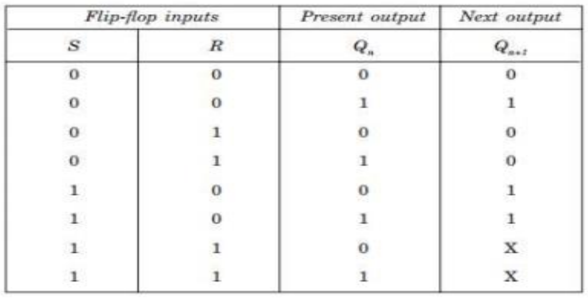
Logic diagram

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	

It differs from the S-R latch in that it has only one input in addition to EN.
When D=1, we have S=1 and R=0, causing the latch to SET when ENABLED.
When D=0, we have 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 flip-flop and a HIGH D input makes Q HIGH, i.e. sets the flip-flop.
In other words, we can say that the output Q follows the D input when EN is HIGH.

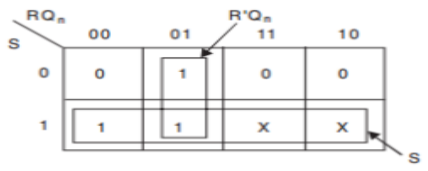
Characteristic Table of an S-R Flip-flop

From the name itself it is very clear that the characteristic table of a flip-flop actually gives us an idea about the character, i.e., the working of the flip-flop. Now, from all our above discussions, we know that the next state flip-flop output (Qn+1) depends on the present inputs as well as the present output (Qn). So in order to know the next state output of a flip-flop, we have to consider the present state output also. The characteristic table of an S-R fl ip-fl op is given in the table below. From the characteristic table we have to find out the characteristic equation of the S-R flip-flop.



Flip-flop inputs		*Present output*	*Next output*
S	*R*	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	X
1	1	1	X

Now we will find out the characteristic equation of the S-R flip-flop from the characteristic table with the help of the Karnaugh map:-



From the Karnaugh map above we find the expression for Q_{n+1} as

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

Along with the above equation we have to consider the fact that S and R cannot be simultaneously 0. In order to take that fact into account we have to incorporate another equation for the S-R flip-flop. The equation is given below.

$$\boxed{SR = 0}$$

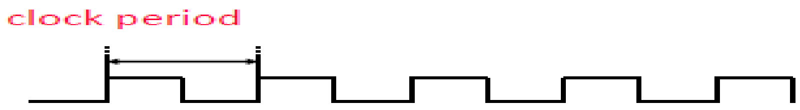
Hence the characteristic equations of an S-R flip-flop are

$$\boxed{\begin{array}{l} Q_{n+1} = S + R'\,Q_n \\ SR = 0 \end{array}}$$

Clock

- clock is a special device that whose output continuously alternates between 0 and 1.

- The time it takes the clock to change from 1 to 0 and back to 1 is called the clock period, or clock cycle time.
- Clocks are often used to synchronize circuits.



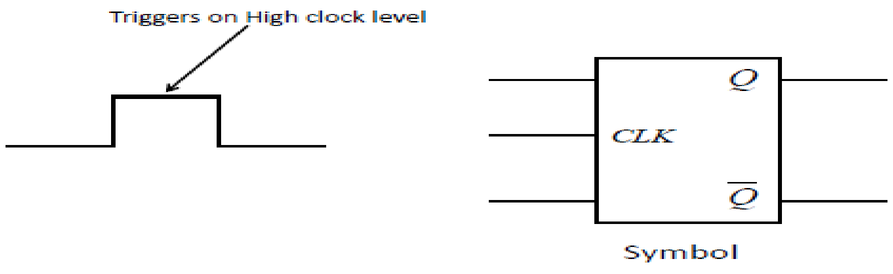
Triggering

- Sequential circuits are dependent on clock pulses applies to their inputs.
- The result of flip-flop responding to a clock input is called clock pulse triggering, of which there are four types. Each type responds to a clock pulse in one of four ways:-

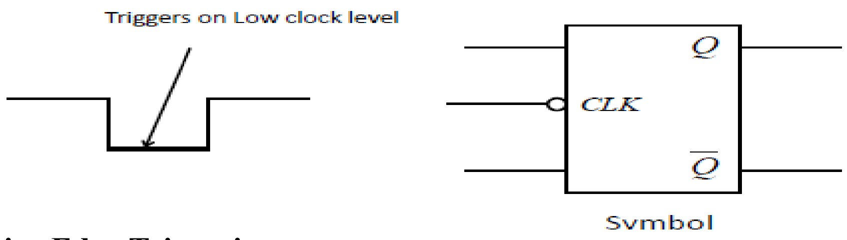
- High level triggering
- Low level triggering
- Positive edge triggering
- Negative edge triggering

High Level Triggering :

- A flip flop who responds to a clock signal during the time at which it is in the logic High state.

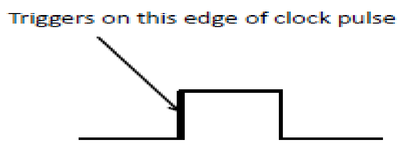


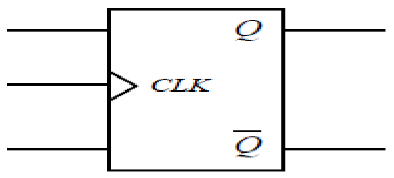
- **Low Level Triggering**
- A flip flop who responds to a clock signal during the time at which it is in the logic Low state.



Positive Edge Triggering

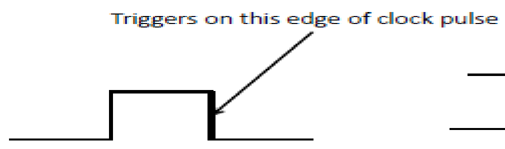
- A flip flop who responds to a clock signal during Low to High transition of clock pulse.

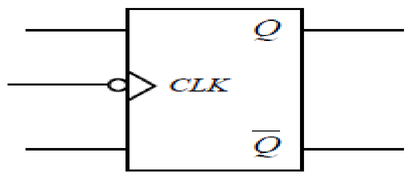




Symbol

- **Negative Edge Triggering**
- A flip flop who responds to a clock signal during High to Low transition of clock pulse.

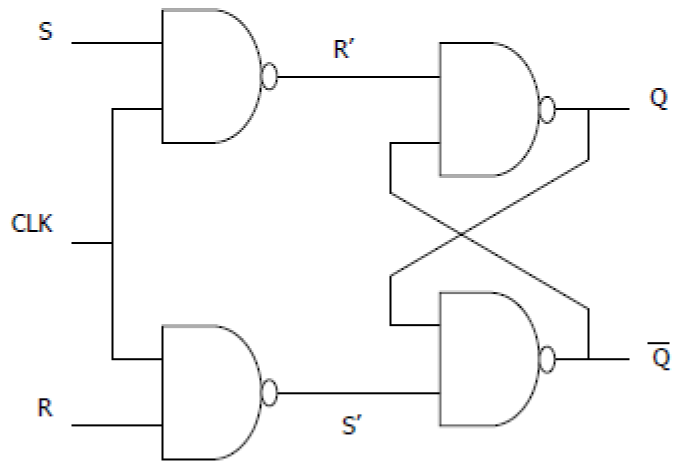


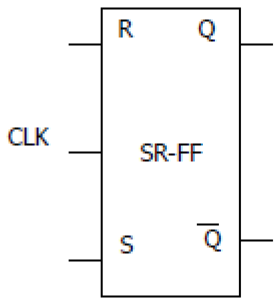


Symbol

Latch vs. Flip-Flop
Latch:
Change stored value under specific status of the control signals.
Transparent for input signals when control signal is on.
May cause combinational feedback loop and extra changes at the Output
Flip-Flop:
Can only change stored value by a momentary switch in value of the control signals Cannot see. the change of its output in the same clock pulse .
Encounter fewer problems than using latches.

Clocked SR Flip Flop





Circuit Diagram

Symbol

Logic Table

CLK	S	R	Q	$\overline{Q}$	State
⊓	0	0	Q	$\overline{Q}$	No Change
⊓	0	1	0	0	Reset
⊓	1	0	0	1	Set
⊓	1	1	X	X	Prohibited
⊔	X	X	Q	$\overline{Q}$	No Change

The S and R inputs of the S-R flipflop are called synchronous control inputs because data on these inputs affect the flipflop's outputs only on the triggering edges of the clock pulse.

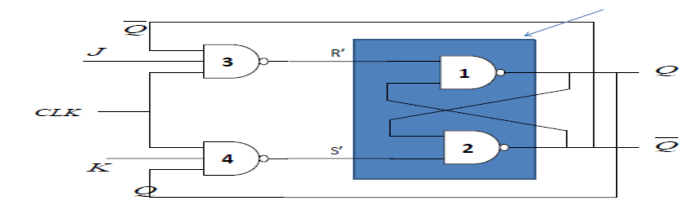
When the SET input is made HIGH ,Q becomes 1 on the positive going edge of the clock pulse and flip flops SET. when the RESET input is made HIGH, Q becomes 0 on the positive going edge of the clock pulse and flip flops REsSET. If both the inputs S and R made LOW ,there is no change in the state of the latch, if both the inputs are made HIGH ,the output is unpredictable ,(invalid).

The truth Table of negative edge –triggered S-R flip flop is the same as that of a positive edge triggered S-R Flip –flop except that the arrows point downwards.

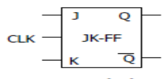
Drawbacks of SR FF

If both inputs are pulled down to logic level 0, both outputs will be at logic level 1. This state should not be allowed to occur in flip-flops.

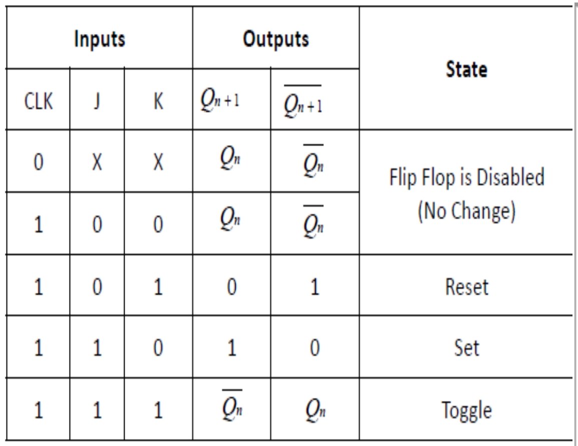
Level Triggered JK Flip Flop



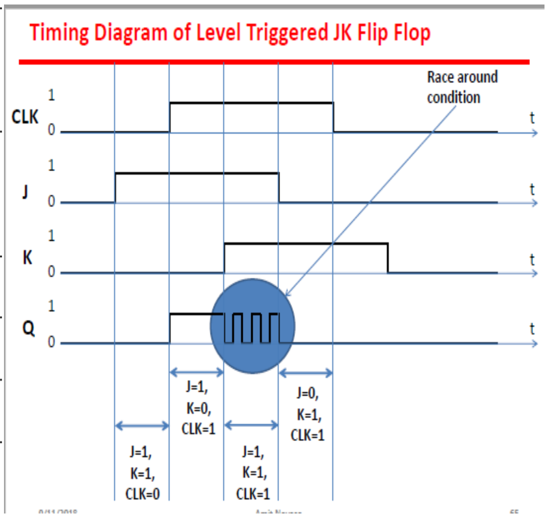
Circuit Diagram of Level Triggered JK Flip Flop



Symbol



Inputs			Outputs		State
CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	
0	X	X	Q_n	$\overline{Q_n}$	Flip Flop is Disabled (No Change)
1	0	0	Q_n	$\overline{Q_n}$	
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	$\overline{Q_n}$	Q_n	Toggle



Truth table

A J-K flip-flop has very similar characteristics to an S-R flip-flop. The only difference is that the undefined condition for an S-R flip-flop, *i.e.*, S*n* = R*n* = 1 condition, is also included in this case. Inputs J and K behave like inputs S and R to set and reset the flip-flop respectively. When J = K = 1, the flip-flop is said to be in a ***toggle state***, which means the output switches to its complementary state every time a clock passes. The data inputs are J and K, which are ANDed with Q' and Q respectively to obtain the inputs for S and R respectively. A J-K flip-flop thus obtained is shown in Figure above . An S-R flip-flop converted into a J-K flip-flop:-

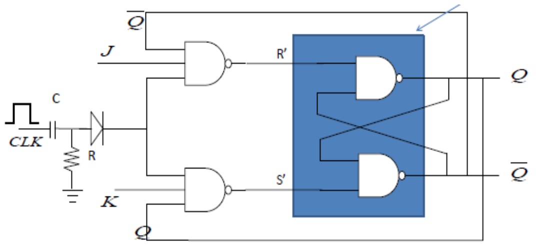
Case 1. When the clock is applied and J = 0, whatever the value of Q'*n* (0 or 1), the output of NAND gate 1 is 1. Similarly, when K = 0, whatever the value of Q*n* (0 or 1), the output of gate 2 is also 1. Therefore, when J = 0 and K = 0, the inputs to the basic flip-flop are S = 1 and R = 1. This condition forces the flip-flop to remain in the same state.

Case 2. When the clock is applied and J = 0 and K = 1 & the previous state of the flip-flop is reset (*i.e.*, Q*n* = 0 and Q'*n* = 1), then S = 1 and R = 1. Since S = 1 and R = 1, the basic flip-flop does not alter the state and remains in the reset state. But if the flip-flop is in set condition (*i.e.*, Q*n* = 1 & Q'*n* = 0), then S = 1 and R = 0. Since S = 1 and R = 0, the basic flip-flop changes its state and resets.

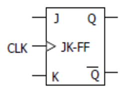
Case 3. When the clock is applied and J = 1 and K = 0 and the previous state of the flip-flop is reset (*i.e.*, Q*n* = 0 and Q'*n* = 1), then S = 0 and R = 1. Since S = 0 and R = 1, the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (*i.e.*, Q*n* = 1 and Q'*n* = 0), then S = 1 and R = 1. Since S = 1 and R = 1, the basic flip-flop does not alter its state and remains in the set state.

Case 4. When the clock is applied and J = 1 and K = 1 and the previous state of the flip-flop is reset (*i.e.*, Q*n* = 0 and Q'*n* = 1), then S = 0 and R = 1. Since S = 0 and R = 1, the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (*i.e.*, Q*n* = 1 and Q'*n* = 0), then S = 1 and R = 0. Since S = 1 and R = 0, the basic flip-flop changes its state and goes to the reset state. So we find that for J = 1 and K = 1, the flip-flop toggles its state from *set* to *reset* and vice versa. Toggle means to switch to the opposite state.

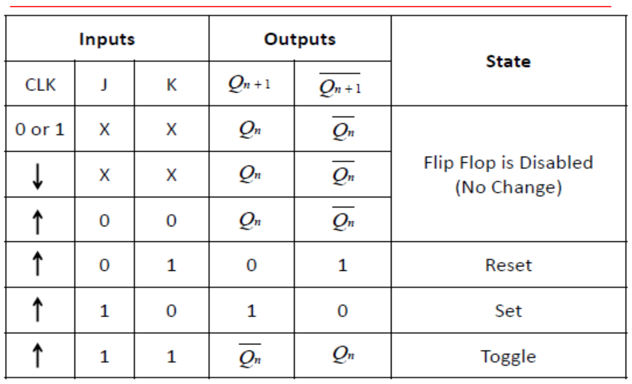
Edge Triggered JK Flip Flop



Circuit Diagram of Edge Triggered JK Flip Flop



Symbol



Inputs			Outputs		State
CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	
0 or 1	X	X	Q_n	$\overline{Q_n}$	Flip Flop is Disabled (No Change)
↓	X	X	Q_n	$\overline{Q_n}$	
↑	0	0	Q_n	$\overline{Q_n}$	
↑	0	1	0	1	Reset
↑	1	0	1	0	Set
↑	1	1	$\overline{Q_n}$	Q_n	Toggle

How to Avoid Race Around Condition in JK Flip Flop using Edge Triggered JK Flip Flops?

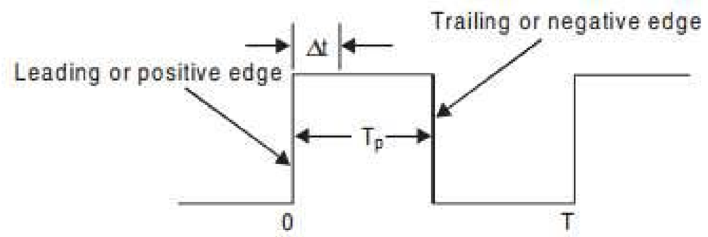
- For the racing around to take place, it is necessary to have the enable input high along with JK=1.
- As the enable input remains high for a long time in a JK Flip Flop, the problem of multiple toggling arises.
- But in edge triggered JK Flip Flop, the positive clock pulse is present only for a very short time.
- Hence by the time changed outputs return back to the inputs of NAND gates 3 and 4, the clock pulse has died down to zero. Hence the multiple toggling can not take place.
- Thus the edge triggering avoids the race around condition.

Synchronous Sequential Circuits Vs Asynchronous Sequential Circuits

Synchronous Seq Circuits	Asynchronous Seq Circuits
In Synchronous circuits, memory elements are clocked FFs.	In Asynchronous circuits, memory elements are either unclocked FFs or time delay elements
In Synchronous circuits, the change in input signals can affect memory elements upon activation of clock signals.	In Asynchronous circuits, the change in input signals can affect memory elements at any instant of time.
The maximum operating speed of the clock depends on time delays involved	Because of the absence of the clock, asynchronous circuits can operate faster than synchronous circuits
Easier to design	More difficult to design

Race-around Condition of a J-K Flip-flop

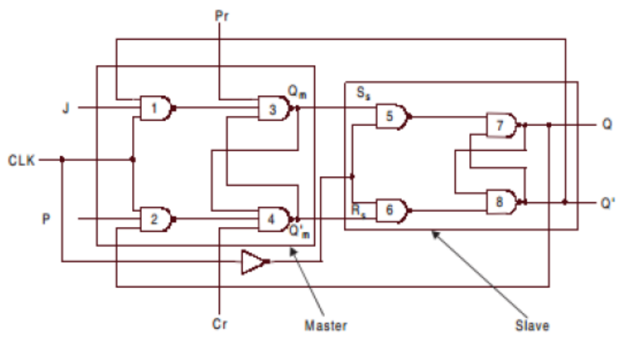
The inherent difficulty of an S-R flip-flop (*i.e.*, S = R = 1) is eliminated by using the feedback connections from the outputs to the inputs of gate 1 and gate 2 as discussed in JK flip-flop. Truth tables JK flip-flop were formed with the assumption that the inputs do not change during the clock pulse (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 below is applied at the clock input.



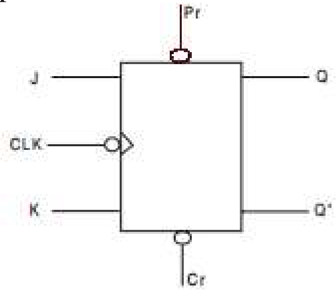
Consider, for example, that the inputs are J = K = 1 and Q = 1, and a pulse as shown above is applied at the clock input. After a time interval Δt equal to the propagation delay through two NAND gates in series, the outputs will change to Q = 0. So now we have J = K = 1 and Q = 0. After another time interval of Δt 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 $tp < \Delta t <$ 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.

Master-Slave J-K Flip-flop

A master-slave (M-S) flip-flop is shown in Figure below. Basically, a master-slave flip-flop is a system of two flip-flops—one being designated as *master* and the other is the *slave*. From the figure below we see that a clock pulse is applied to the master and the inverted form of the same clock pulse is applied to the slave. When CLK = 1, the first flip-flop (*i.e.*, the master) is enabled and the outputs Q*m* and Q'*m* respond to the inputs J and K according to the table shown in Figure 7.13. At this time the second flip-flop (*i.e.*, the slave) is disabled because the CLK is LOW to the second flip-flop. Similarly, when CLK becomes LOW, the master becomes disabled and the slave becomes active, since now the CLK to it is HIGH. Therefore, the outputs Q and Q' follow the outputs Q*m* and Q'*m* respectively. Since the second flip-flop just follows the first one, it is referred to as a slave and the first one is called the master. Hence, the configuration is referred to as a master-slave (M-S) flipflop.



In this type of circuit configuration the inputs to the gates 5 and 6 do not change at the time of application of the clock pulse. Hence the race-around condition does not exist. The state of the master-slave flip-flop, shown in above Figure, changes at the negative transition (trailing edge) of the clock pulse. Hence, it becomes negative triggering a master-slave flip-flop. This can be changed to a positive edge triggering flip-flop by adding two inverters to the system—one before the clock pulse is applied to the master and an additional one in between the master and the slave. The logic symbol of a negative edge master-slave is shown in Figure below. The system of master-slave flip-flops is not restricted to J-K masterslave only. There may be an S-R master-slave or a D master-slave, etc., in all of them the slave is an S-R flip-flop, whereas the master changes to J-K or S-R or D flip-flops..

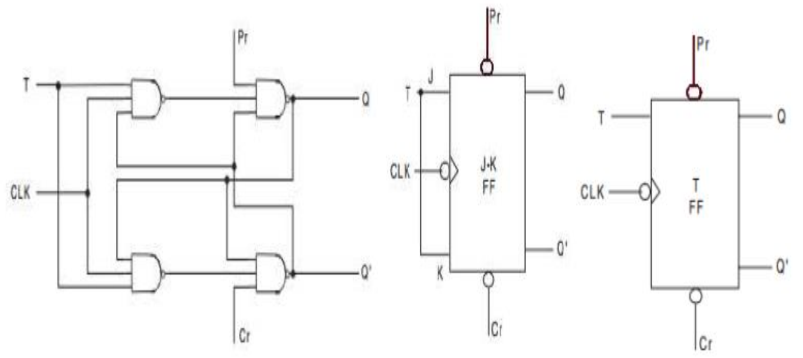


T Flip-flop

With a slight modification of a J-K flip-flop, we can construct a new flip-flop called a T flip-flop. If the two inputs J and K of a J-K flip-flop are tied together it is referred to as a T flip-flop. Hence, a T flip-flop has only one input T and two outputs Q and Q'. The name T flip-flop actually indicates the fact that the flip-flop has the ability to toggle. It has actually only two states—***toggle state*** **and** ***memory state***. Since there are only two states, a T flip-flop is a very good option to use in counter design and in sequential circuits design where switching an operation is required. The truth table of a T flip-flop is given below:-

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

If the T input is in 0 state (*i.e.*, J = K = 0) prior to a clock pulse, the Q output will not change with the clock pulse. On the other hand, if the T input is in 1 state (*i.e.*, J = K = 1) prior to a clock pulse, the Q output will change to Q' with the clock pulse. In other words, we may say that, if T = 1 and the device is clocked, then the output toggles its state. The truth table shows that when T = 0, then Q*n*+1 = Q*n*, *i.e.*, the next state is the same as the present state and no change occurs. When T = 1, then Q*n*+1 = Q'*n*, *i.e.*, the state of the flip-flop is complemented. The circuit diagram of a T flip-flop and the block diagram of the T flip-flop is shown below:-



TRIGGERING OF FLIP-FLOPS

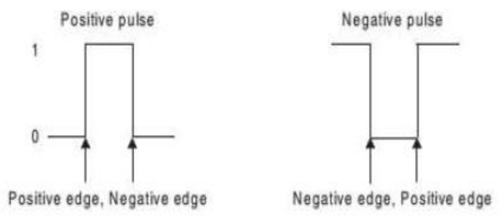
Flip-flops are synchronous sequential circuits. This type of circuit works with the application of a synchronization mechanism, which is termed as a *clock*. Based on the specific interval or point in the clock during or at which triggering of the flip-flop takes place, it can be classified into two different types—***level triggering*** **and** ***edge triggering***. A clock pulse starts from an initial value of 0, goes momentarily to 1, and after a short interval, returns to the initial value.

Level Triggering of Flip-flops

If a flip-flop gets enabled when a clock pulse goes HIGH and remains enabled throughout the duration of the clock pulse remaining HIGH, the flip-flop is said to be a *level triggered flip-flop*. If the flip-flop changes its state when the clock pulse is positive, it is termed as a *positive level triggered flip-flop*. On the other hand, if a NOT gate is introduced in the clock input terminal of the flip-flop, then the flip-flop changes its state when the clock pulse is negative, it is termed as a *negative level triggered flip-flop.* The main drawback of level triggering is that, as long as the clock pulse is active, the flip-flop changes its state more than once or many times for the change in inputs. If the inputs do not change during one clock pulse, then the output remains stable. On the other hand, if the frequency of the input change is higher than the input clock frequency, the output of the flipflop undergoes multiple changes as long as the clock remains active. This can be overcome by using either master-slave flip-flops or the edge-triggered flip-flop.

Edge-triggering of Flip-flops

A clock pulse goes from 0 to 1 and then returns from 1 to 0. The Figure below shows the two transitions and they are defined as the *positive edge* (0 to 1 transition) and the *negative edge* (1 to 0 transition). The term *edgetriggered* means that the flip-flop changes its state only at either the positive or negative edge of the clock pulse.

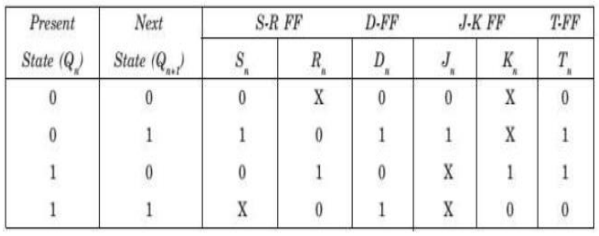


EXCITATION TABLE OF A FLIP-FLOP

The truth table of a flip-flop is also referred to as the characteristic table of a flip-flop, since this table refers to the operational characteristics of the flip-flop. But in designing sequential circuits, we often face situations where the present state(PS) & the next state(NS) of the flip-flop is specified, and we have to find out the input conditions that must prevail for the desired output condition. By present and next states we mean to say the conditions before and after the clock pulse respectively. For example, the output of an S-R flip-flop before the clock pulse is Q*n* = 1 and it is desired that the output does not change when the clock pulse is applied. Now from the characteristic table of an S-R flip-flop, we obtain the following conditions:

1. S = R = 0 (second row)
2. S = 1, R = 0 (sixth row).

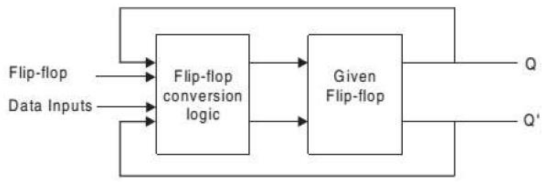
We come to the conclusion from the above conditions that the R input must be 0, whereas the S input may be 0 or 1 (*i.e.*, don't-care). Similarly, for all possible situations, the input conditions can be found out. A tabulation of these conditions is known as an *excitation table*. The table below gives the excitation table for S-R, D, J-K, & T flip-flops. These conditions are derived from the corresponding characteristic tables of the flip-flops.



Present	Next	S-R FF		D-FF	J-K FF		T-FF
State (Q_n)	State (Q_{n+1})	S_n	R_n	D_n	J_n	K_n	T_n
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	0	0	1	0	X	1	1
1	1	X	0	1	X	0	0

INTERCONVERSION OF FLIP-FLOPS

In many applications, we are being given a type of flip-flop, whereas we may require some other type. In such cases we may have to convert the given flip-flop to our required flip-flop. Now we may follow a general model for such conversions of flip-flops. The model is shown in below From the model we see that it is required to design the conversion logic for converting new input definitions into input codes that will cause the given flipflop to work like the desired flip-flop. To design the conversion logic we need to combine the excitation table for both flip-flops and make a truth table with data input(s) and Q as the inputs and the input(s) of the given flip-flop as the output(s).

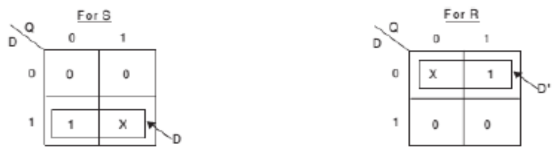


Conversion of an S-R Flip-flop to a D Flip-flop

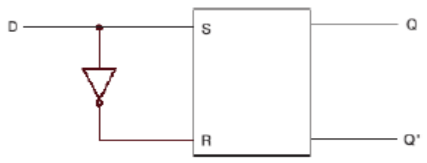
The excitation tables of S-R and D flip-flops are given below from which we make the truth table given

FF data inputs	*Output*	*S-R FF inputs*	
D	*Q*	*S*	*R*
0	0	0	X
1	0	1	0
0	1	0	1
1	1	X	0

From the above table, we make the Karnaugh maps for inputs S and R as shown in Figure below:-



Simplifying with the help of the Karnaugh maps, we obtain S = D and R = D'. Hence the circuit may be designed as in Figure below:-

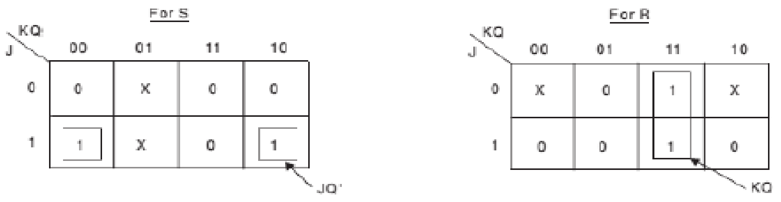


Conversion of an S-R Flip-flop to a J-K Flip-flop

The excitation tables of S-R and J-K flip-flops, as we studied before, from which we make the truth table given in below

FF data inputs		*Output*	*S-R FF inputs*	
J	*K*	*Q*	*S*	*R*
0	0	0	0	X
0	1	0	0	X
1	0	0	1	0
1	1	0	1	0
0	1	1	0	1
1	1	1	0	1
0	0	1	X	0
1	0	1	X	0

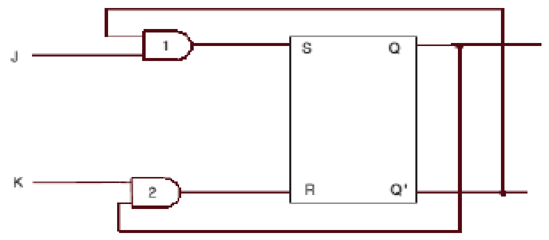
From the above truth table, the Karnaugh map is prepared as shown in Figure below:-



Hence we get the Boolean expression for S and R as

$$\boxed{\begin{array}{l} S = JQ' \\ \& \; R = KQ. \end{array}}$$

Hence the circuit may be realized as in below:-

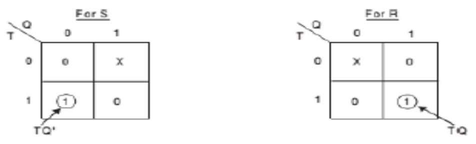


Conversion of an S-R Flip-flop to a T Flip-flop

The excitation tables of S-R and T flip-flops, as we studied before, from which we make the truth table given in below:-

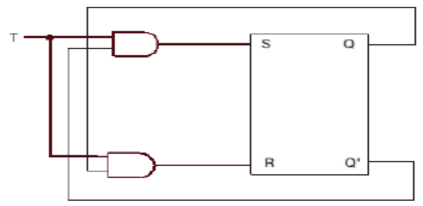
FF data inputs	*Output*	*S-R FF inputs*	
T	*Q*	*S*	*R*
0	0	0	X
1	0	1	0
1	1	0	1
0	1	X	0

From the above truth table, the Karnaugh map is prepared as shown in Figure below:-



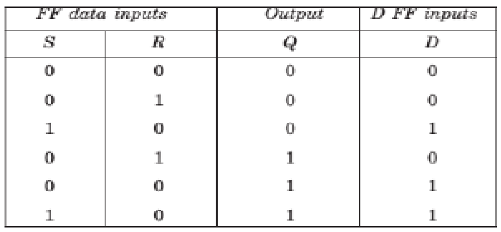
Hence we get the Boolean expression for S and R as:- $\boxed{S = TQ' \text{ and } R = TQ}$

Hence the circuit may be realized as in below:-



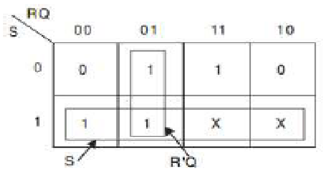
Conversion of a D Flip-flop to an S-R Flip-flop

The excitation tables of S-R and D flip-flops, as we studied before, from which we make the truth table given in below:-



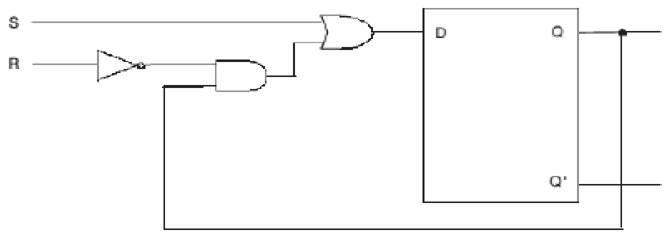
FF data inputs		*Output*	*D FF inputs*
S	*R*	*Q*	*D*
0	0	0	0
0	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1
1	0	1	1

From the above truth table, the Karnaugh map is prepared as shown in Figure below:-



Hence we get the Boolean expression for S and R as:- **D = S + R'Q**

Hence the circuit may be realized as in below:-



Similar procedure is applied for all type of Flip-Flop conversion and is left as an assignment for the student.

Applications of Flip Flops

- ✓ Elimination of Keyboard Debounce

✓As a memory element

✓In various types of registers

✓In counters

✓As delay element

✓Parallel Data storage

✓Serial Data Storage

✓Serial to Parallel Conversion

✓Parallel to Serial Conversion

✓Frequency Division

Shift registers

Introduction

Shift registers are a type of sequential logic circuit, mainly for storage of digital data. They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop. Most of the registers possess no characteristic internal sequence of states. All flip-flop is driven by a common clock, and all are set or reset simultaneously. In these few lectures, the basic types of shift registers are studied, such as Serial In - Serial Out, Serial In - Parallel Out, Parallel In – Serial Out, Parallel In - Parallel Out, and bidirectional shift registers. A special form of counter - the shift register counter, is also introduced.

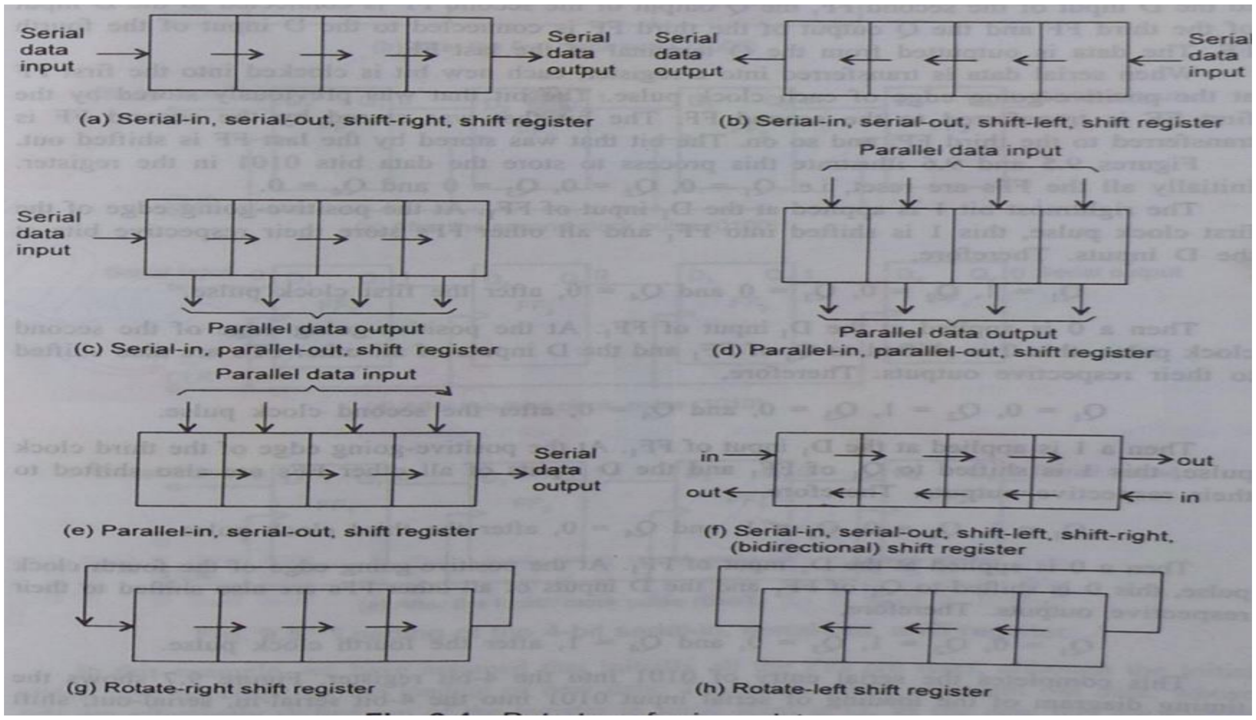
Register:

A set of n flip-flops

Each flip-flop stores one bit

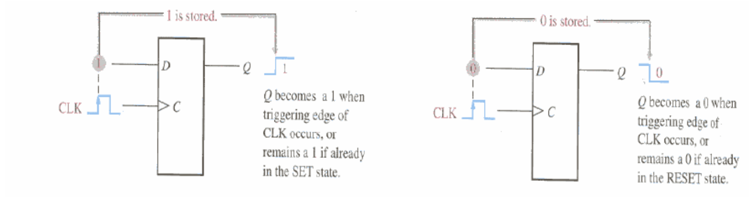
Two basic functions: data storage (Figure 1.2) and data movement (Figure 1.1). Shift Register:

A register that allows each of the flip-flops to pass the stored information to its adjacent neighbour ?? Figure 1.1 shows the basic data movement in shift registers.


Serial data input
Serial data output
(a) Serial-in, serial-out, shift-right, shift register
Serial data output
Serial data input
(b) Serial-in, serial-out, shift-left, shift register
Parallel data input
Parallel data output
(d) Parallel-in, parallel-out, shift register
Serial data input
Parallel data output
(c) Serial-in, parallel-out, shift register
Parallel data input
Serial data output
(e) Parallel-in, serial-out, shift register
in
out
out
in
(f) Serial-in, serial-out, shift-left, shift-right, (bidirectional) shift register
(g) Rotate-right shift register
(h) Rotate-left shift register

Storage Capacity:

The storage capacity of a register is the total number of bits (1 or 0) of digital data it can retain. Each stage (flip flop) in a shift register represents one bit of storage capacity. Therefore the number of stages in a register determines its storage capacity.


1 is stored.
D
C
Q
CLK
1
Q becomes a 1 when triggering edge of CLK occurs, or remains a 1 if already in the SET state.
0 is stored.
D
C
Q
CLK
0
Q becomes a 0 when triggering edge of CLK occurs, or remains a 0 if already in the RESET state.

Serial In - Serial Out Shift Registers

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

2.1 Example: Basic four-bit shift register

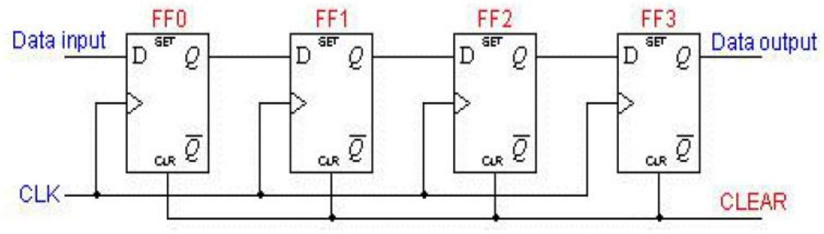


Figure 2.1

A basic four-bit shift register can be constructed using four D flip-flops, as shown in Figure 2.1.

The operation of the circuit is as follows.

The register is first cleared, forcing all four outputs to zero.

The input data is then applied sequentially to the D input of the first flip-flop on the left (FF0).

During each clock pulse, one bit is transmitted from left to right.

Assume a data word to be 1001.

The least significant bit of the data has to be shifted through the register from FF0 to FF3.

In order to get the data out of the register, they must be shifted out serially. This can be done destructively or non-destructively. For destructive readout, the original data is lost and at the end of the read cycle, all flip-flops are reset to zero.

FF0	FF1	FF2	FF3	
0	0	0	0	1001

Figure 2.2 illustrates entry of the four bits 1010 into the register. Figure 2.3 shows the four bits (1010) being serially shifted out of the register and replaced by all zeros. Figure

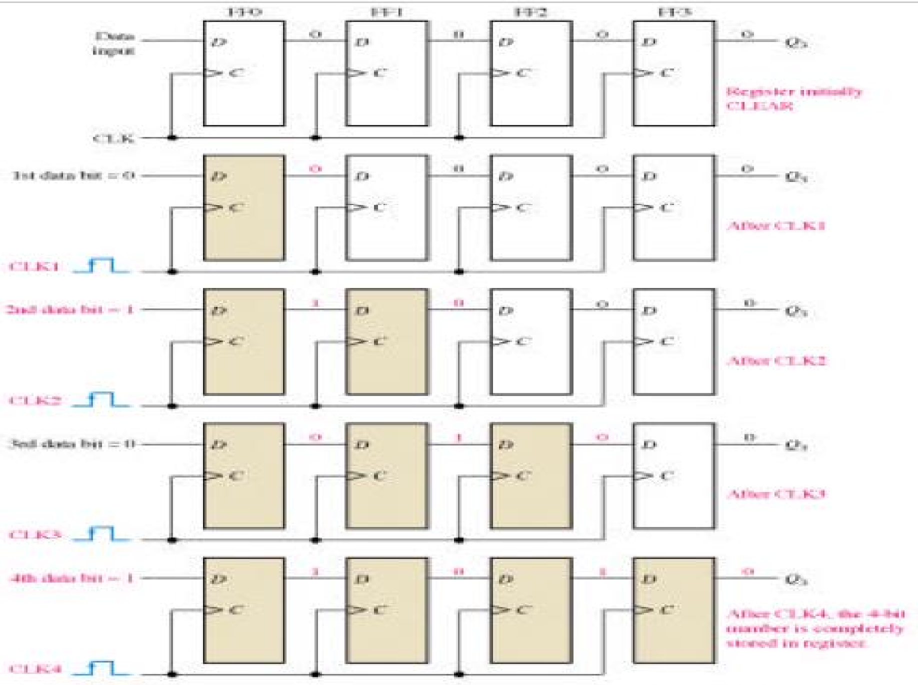
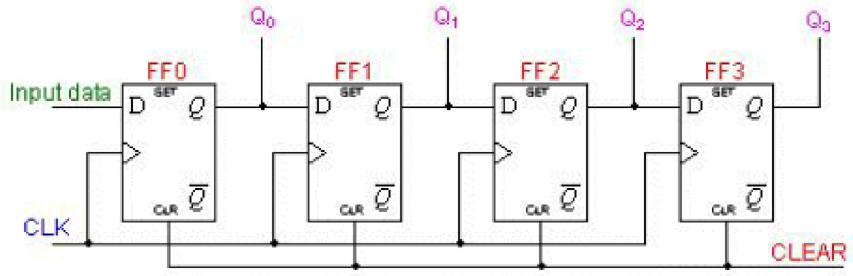


Figure 2.2: Four bits (1010) being entered serially into the register.

3.0 Serial In - Parallel Out Shift Registers

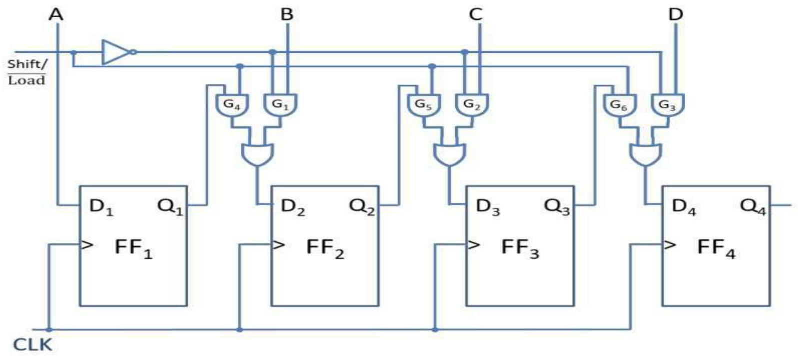
For this kind of register, data bits are entered serially in the same manner as discussed in the last section. The difference is the way in which the data bits are taken out of the register. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously. A construction of a four-bit serial in - parallel out register is shown below



In the table below, we can see how the four-bit binary number 1001 is shifted to the Q outputs of the register.

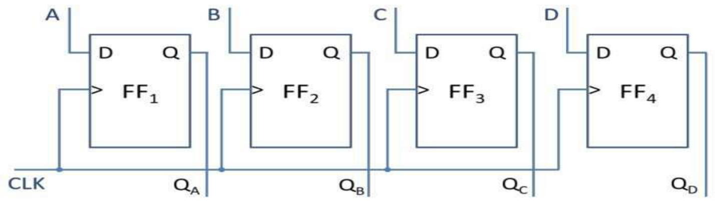
Clear	FF0	FF1	FF2	FF3
1001	0	0	0	0
	1	0	0	0
	0	1	0	0
	0	0	1	0
	1	0	0	1

Parallel-in, Serial-out, Shift register



- There are four data lines A, B, C, and D through which the data is entered into the register in parallel form.
- The signal Shift/$\overline{LOAD}$ allows (a) the data to be entered in parallel form into the register and (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 gates G_4, G_5, and 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, whereas 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 a clock pulse is applied, these data bits are shifted t the Q output terminals of the FFs and, therefore, data is inputted in one step.
  The OR gate allows wither the normal shifting operation or the parallel data entry depending on which NAD gates are enabled by the level on the Shift/$\overline{LOAD}$ input.

Parallel-in, Parallel-out, Shift register



- In a parallel-in, parallel-out, shift register the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form.
- Data is applied to the D input terminals of the FFs.
- When a clock pulse is applied, at the positive-going edge of that pulse, the D inputs are shifted into the Q outputs of the FFs.

  The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

Ring counter

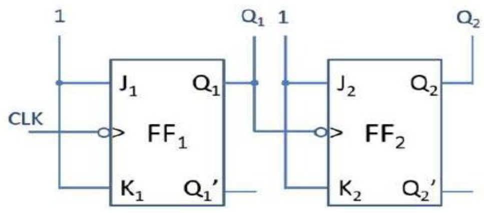
- This is the simplest shift register counter. The basic ring counter using D FFs is shown in figure.
- The FFs are arranged as in a normal shift register, i.e. Q output of each stage is connected to the D input of the next stage, but the Q output of the last FF is connected back to the D input of the first FF such that the array of FFs is arranged in a ring, and therefore, the name *ring counter.*

Asynchronous Vs Synchronous Counter

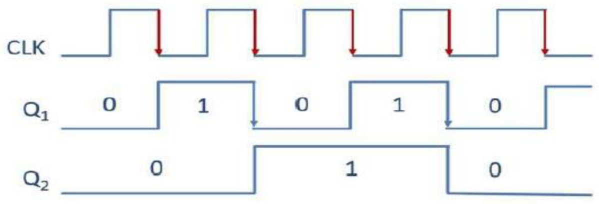
Asynchronous Counter	Synchronous Counter
✓ Output of the preceding Flip Flop is connected to clock of the next Flip Flop.	✓ There is no connection between output of preceding Flip Flop and Clock of next one.
✓ All the Flip Flops are not clocked simultaneously.	✓ All the Flip Flops receive clock signal simultaneously.
✓ Logic circuit is simple.	✓ With increase in number of states, the logic circuit becomes complicated.

Asynchronous Counter	Synchronous Counter
✓ P.D. = n X (t_d) where n is number of Flip Flops and t_d is propagation delay of flip flop.	✓ P.D. = $(t_d)_{FF} + (t_d)_{Gate}$,It is much shorter than that of asynchronous counter.
✓ Frequency of operation is low because of the long propagation delay	✓ Frequency of operation is high due to shorter propagation delay.

2-bit ripple up-counter using negative edge triggered FF

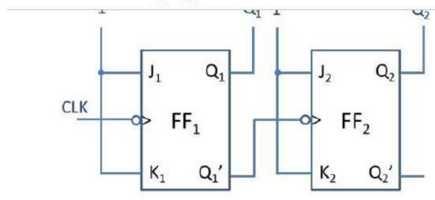


CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↓	0	0	0	1
↓	0	1	1	0
↓	1	0	1	1
↓	1	1	0	0

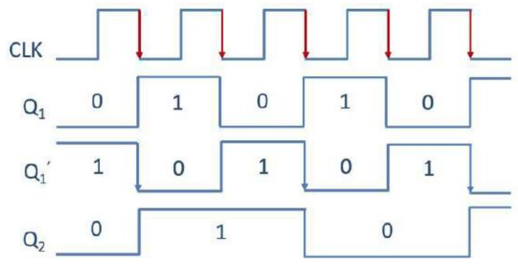


- The 2-bit up-counter counts in the order 0, 1, 2, 3, 0, … etc..
- The counter is initially reset to 00.
- When the first clock pulse is applied, FF_1 toggles at the negative edge of this pulse, therefore, Q_1 goes from LOW to HIGH.
- This becomes a positive edge at the clock input of FF_2. So FF_2 is not affected, and hence the state of the counter after one clock pulse is $Q_1 = 1$ and $Q_2 = 0$, i.e. 01.
- At the negative edge of the second clock pulse, FF_1 toggles.
- So Q_1 changes from HIGH to LOW and this negative edge clock applied to CLK of FF_2 activates FF_2, and hence, Q_2 goes from LOW to HIGH. Therefore, $Q_1 = 0$ and $Q_2 = 1$, i.e. 10 is the state of the counter after the second clock pulse.
- At the negative edge of the third clock pulse, FF_1 toggles.
- So Q_1 changes from a 0 to a 1. This becomes a positive edge to FF_2, hence FF_2 is not affected. Therefore, $Q_2 = 1$ and $Q_1 = 1$, i.e. 11 is the state of the counter after the third clock pulse.
- At the negative edge of the fourth clock pulse, FF_1 toggles.
- So, Q_1 changes from a 1 to a 0. This negative edge at Q_1 toggles FF_2, hence Q_2 also changes from a 1 to a 0. Therefore, $Q_2 = 0$ and $Q_1 = 0$, i.e. 00 is the state of the counter after the fourth clock pulse.
- So, it acts as a mod-4 counter with Q_1 as the LSB and Q_2 as the MSB.

2-bit ripple down-counter using negative edge triggered FF



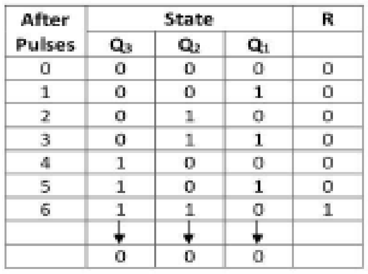
CLK	Present State		Next State	
	Q_1	Q_0	Q_1	Q_0
↓	0	0	1	1
↓	1	1	1	0
↓	1	0	0	1
↓	0	1	0	0



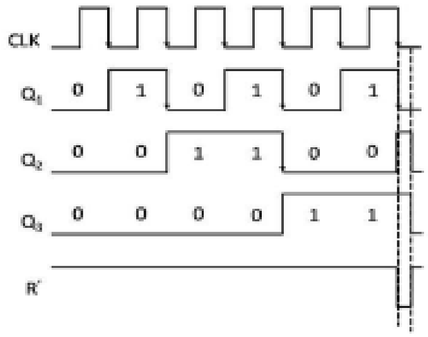
- A 2-bit down-counter counts in the order 0, 3, 2, 1, 0, ...etc.
- For down counting, Q_1' of FF_1 is connected to the clock of FF_2. Let initially all the FFs be reset, i.e. 00.
  - At the negative edge of the first clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
  - This negative edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 0 to a 1. So, after one clock pulse $Q_2 = 1$ and $Q_1 = 1$, i.e. the state of the counter is 11.
  - At the negative edge of the second clock pulse, Q_1 changes from a 1 to a 0 and Q_1' from a 0 to a 1.
  - This positive edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 1. Hence, the state of the counter after second clock pulse is 10.
  - At the negative edge of third clock pulse, FF_1 toggles. So, Q_1 goes from a 0 to a 1 and Q_1' goes from a 1 to a 0.
  - This negative edge at Q_1' applied to the clock input of FF_2, toggles FF_2 and, therefore, Q_2 goes from a 1 to a 0. Hence, the state of the counter is 01.
  - At the negative edge of fourth clock pulse, FF_1 toggles. So, Q_1 goes from a 1 to a 0 and Q_1' goes from a 0 to a 1.
  - This positive edge at Q_1' does not affect FF_2 and, therefore Q_2 remains at a 0. Hence, the state of the counter after fourth clock pulse is 00.

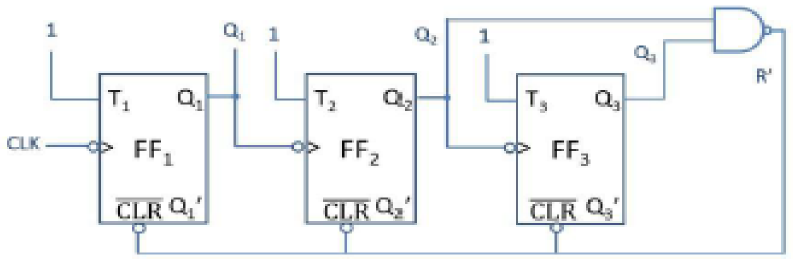
Modulo-6 asynchronous counter

- A mod-6 counter has six stable states 000, 001, 010, 011, 100, 101.
- It is also known as "Divide by 6" counter and it requires 3 Flip-flops for designing.
- At the 6th clock pulse, it will again reset to 000 via feedback circuit.
- Reset signal R = 1 at time of 110, R = 0 for 000 to 101 and R = X for invalid states i.e. 111.
- Therefore, $R = Q_3Q_2Q_1' + Q_3Q_2Q_1 = Q_3Q_2$.



After	State			R
Pulses	Q_3	Q_2	Q_1	
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	

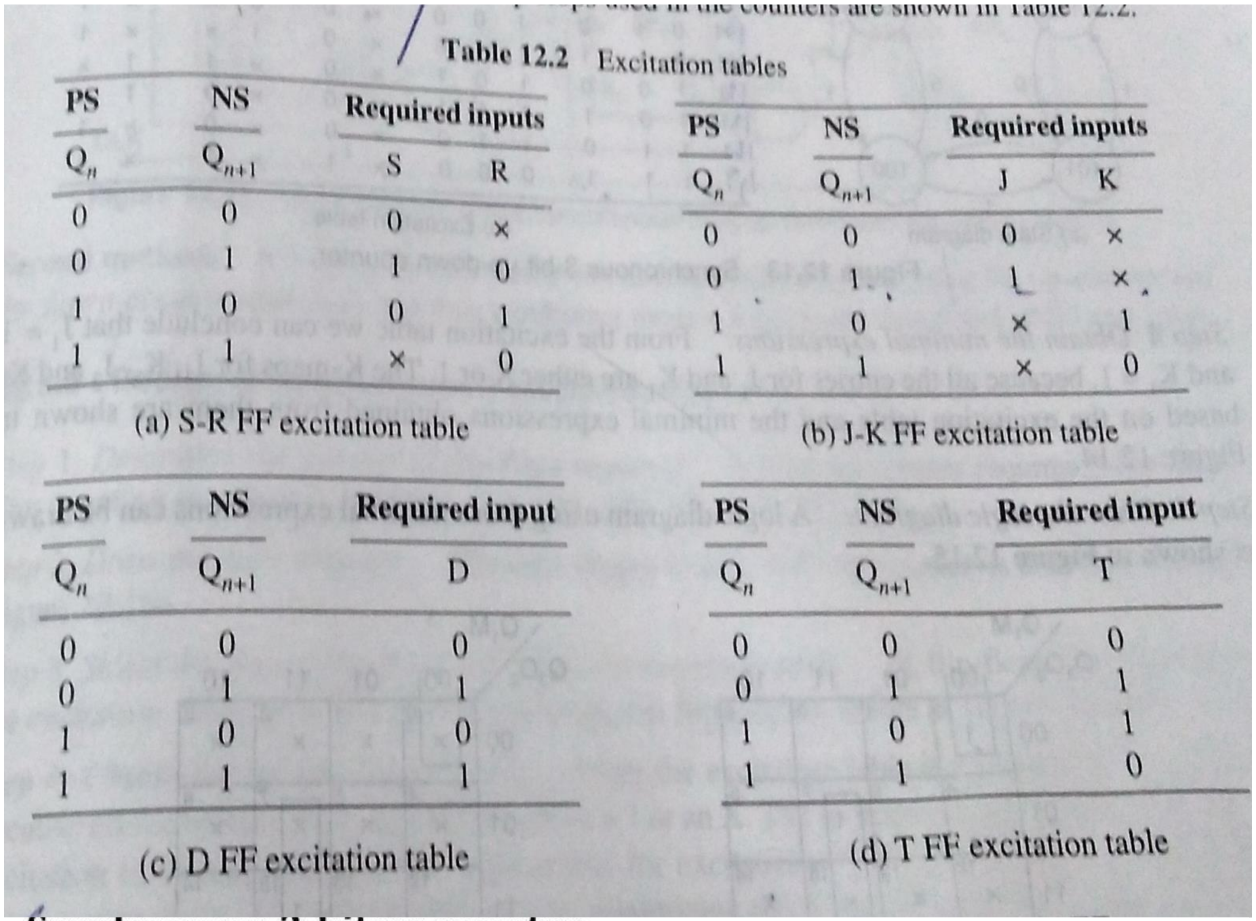




Synchronous counter designing

- Step 1. Number of flip-flops:
  Based on the description of the problem, determine the required number n of the FFs - the smallest value of n is such that the number of states $N \leq 2^n$ and the desired counting sequence.
- Step 2. State diagram:
  Draw the state diagram showing all the possible states.
- Step 3. Choice of flip-flops and excitation table:
  Select the type of flip-flops to be used and write the excitation table.
  An excitation table is a table that lists the present state (PS), the next state (NS) and the required excitations.
- Step 4. Minimal expressions for excitations:
  Obtain the minimal expressions for the excitations of the FFs using K-maps for the excitations of the flip-flops in terms of the present states and inputs.
- Step 5. Logic Diagram:
  Draw the logic diagram based on the minimal expressions.

Flip-Flop Excitation Tables



used in the counters are shown in Table 12.2.

Table 12.2 Excitation tables

PS	NS	Required inputs	
Q_n	Q_{n+1}	S	R
0	0	0	×
0	1	1	0
1	0	0	1
1	1	×	0

(a) S-R FF excitation table

PS	NS	Required inputs	
Q_n	Q_{n+1}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

(b) J-K FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

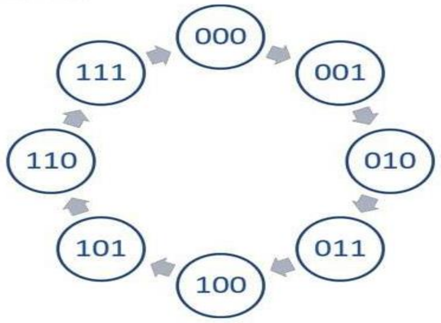
(c) D FF excitation table

PS	NS	Required input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(d) T FF excitation table

Synchronous 3-bit up counter

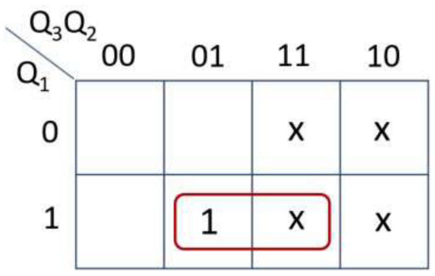
- Step 1. Number of flip-flops:
  A 3-bit up-counter requires 3 flip-flops. The counting sequence is 000, 001, 010, 011, 100, 101, 110, 111, 000 …
- Step 2. Draw the state diagram:



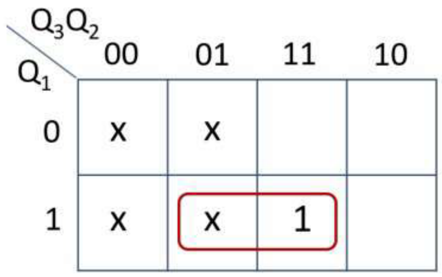
- Step 3. Select the type of flip-flops and draw the excitation table:
  JK flip-flops are selected and the excitation table of a 3-bit up-counter using J-K flip-flops is drawn as shown below.

Present State			Next State			Required Excitation					
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	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

- Step 4. Obtain the minimal expressions
  From excitation table, $J_1 = K_1 = 1$.
  K – Maps for excitations J_3, K_3, J_2 and K_2 and their minimized form are as follows:



$J_3 = Q_2Q_1$



$K_3 = Q_2Q_1$

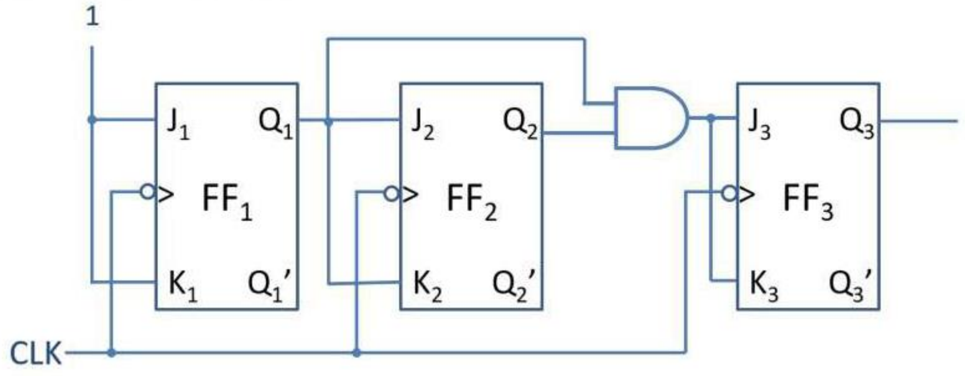
$Q_1 \backslash Q_3Q_2$	00	01	11	10
0		x	x	
1	1	x	x	1

$J_2 = Q_1$

$Q_1 \backslash Q_3Q_2$	00	01	11	10
0	x			x
1	x	1	1	x

$K_2 = Q_1$

- Step 5. Draw the logic diagram



1
J_1
Q_1
FF_1
K_1
Q_1'
J_2
Q_2
FF_2
K_2
Q_2'
J_3
Q_3
FF_3
K_3
Q_3'
CLK