Flip Flops

A digital computer needs devices which can store information. A flip flop is a binary storage device. It can store binary bit either 0 or 1. It has two stable states HIGH and LOW i.e. 1 and 0. It has the property to remain in one state indefinitely until it is directed by an input signal to switch over to the other state. It is also called bistable multivibrator.

The basic formation of flip flop is to store data. They can be used to keep a record or what value of variable (input, output or intermediate). Flip flop are also used to exercise control over the functionality of a digital circuit i.e. change the operation of a circuit depending on the state of one or more flip flops. These devices are mainly used in situations which require one or more of these three.

Operations, storage and sequencing.

Latch Flip Flop

The R-S (Reset Set) flip flop is the simplest flip flop of all and easiest to understand. It is basically a device which has two outputs one output being the inverse or complement of the other, and two inputs. A pulse on one of the inputs to take on a particular logical state. The outputs will then remain in this state until a similar pulse is applied to the other input. The two inputs are called the Set and Reset input (sometimes called the preset and clear inputs).

Such flip flop can be made simply by cross coupling two inverting gates either NAND or NOR gate could be used Figure 1(a) shows on RS flip flop using NAND gate and Figure 1(b) shows the same circuit using NOR gate.

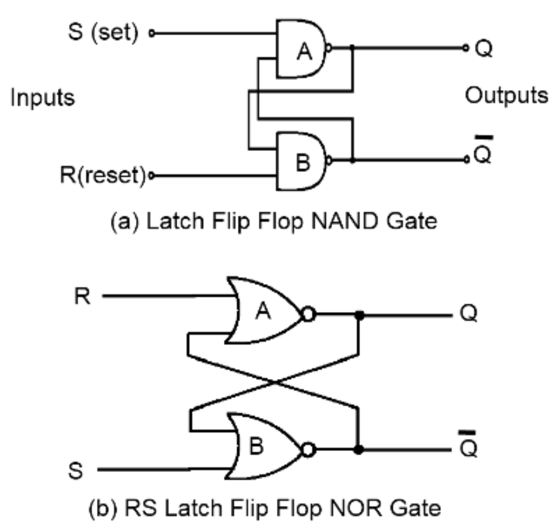


Figure 1: Latch R-S Flip Flop Using NAND and NOR Gates

To describe the circuit of Figure 1(a), assume that initially both R and S are at the logic 1 state and that output is at the logic 0 state.

Now, if Q = 0 and R = 1, then these are the states of inputs of gate B, therefore the outputs of gate B is at 1 (making it the inverse of Q i.e. 0). The output of gate B is connected to an input of gate A so if S = 1, both inputs of gate A are at the logic 1 state. This means that the output of gate A must be 0 (as was originally specified). In other words, the 0 state at Q is continuously disabling gate B so that any change in R has no effect. Also the 1 state at $\overline{Q}$ is continuously enabling gate A so that any change S will be transmitted through to Q. The above conditions constitute one of the stable states of the device referred to as the Reset state since Q = 0.

Now suppose that the R-S flip flop in the Reset state, the S input goes to 0. The output of gate A i.e. Q will go to 1 and with Q = 1 and R = 1, the output of gates B ($\overline{Q}$) will go to 0 with $\overline{Q}$ now 0 gate A is disabled keeping Q at 1. Consequently, when S returns to the 1 state it has no effect on the flip flop whereas a change in R will cause a change in the output of gate B. The above conditions constitute the other stable state of the device, called the Set state since Q = 1. Note that the change of the state of S from 1 to 0 has caused the flip flop to change from the Reset state to the Set state.

There is another input condition which has not yet been considered. That is when both the R and S inputs are taken to the logic state 0. When this happens both Q and $\overline{Q}$will be forced to 1 and will remain so far as long as R and S are kept at 0. However when both inputs return to 1 there is no way of knowing whether the flip flop will latch in the Reset state or the Set state. The condition is said to be indeterminate because of this indeterminate state great care must be taken when using R-S flip flop to ensure that both inputs are not instructed simultaneously.

Table 1: The truth table for the NAND R-S flip flop

Initial Conditions	**Inputs (Pulsed)**		**Final Output**	
Q	**S**	**R**	**Q**	$\overline{Q}$
1	0	0	indeterminate	
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0
0	0	0	indeterminate	
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1

or more simply shown in Table 2

Table 2: Simple NAND R-S Flip Flop Truth Table

S	**R**	**Q**
0	0	indeterminate
0	1	Set (1)
1	0	Reset(0)
1	1	No Change

When NOR gate are used the R and S inputs are transposed compared with the NAND version. Also the stable state when R and S are both 0. A change of state is effected by pulsing the appropriate input to the 1 state. The indeterminate state is now when both R and S are simultaneously at logic 1. Table 3 shows this operation.

Table 3: NOR Gate R-S Flip Flop Truth Table

S	**R**	**Q**
0	0	No Change
0	1	Reset (0)
1	0	Set (1)
1	1	Indeterminate

Clocked RS Flip Flop

The RS latch flip flop required the direct input but no clock. It is very use full to add clock to control precisely the time at which the flip flop changes the state of its output.

In the clocked R-S flip flop the appropriate levels applied to their inputs are blocked till the receipt of a pulse from an other source called clock. The flip flop changes state only when clock pulse is applied depending upon the inputs. The basic circuit is shown in Figure 2. This circuit is formed by adding two AND gates at inputs to the R-S flip flop. In addition to control inputs Set (S) and Reset (R), there is a clock input (C) also.

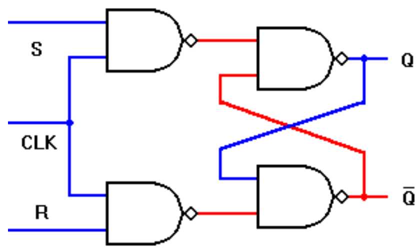


Figure 2: Clocked RS Flip Flop

Table 4: The truth table for the Clocked R-S flip flop

Initial Conditions	Inputs (Pulsed)		Final Output	Comment
Q	**S**	**R**	**Q (t + 1)**	No Change
0	0	0	0	No Change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	indeterminate
1	0	0	1	No Change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	indeterminate

The excitation table for R-S flip flop is very simply derived as given below

Table 5: Excitation table for R-S Flip Flop

S	R	Q
0	0	No Change
0	1	Reset (0)
1	0	Set (1)
1	1	Indeterminate

D Flip Flop

A D type (Data or delay flip flop) has a single data input in addition to the clock input as shown in Figure 3.

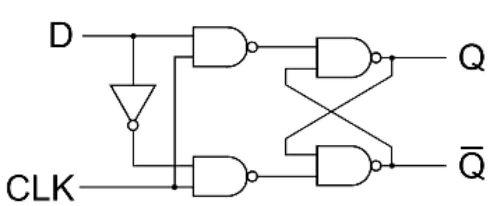


Figure 3: D Flip Flop

Basically, such type of flip flop is a modification of clocked RS flip flop gates from a basic Latch flip flop and NOR gates modify it in to a clock RS flip flop. The D input goes directly to S input and its complement through NOT gate, is applied to the R input.

This kind of flip flop prevents the value of D from reaching the output until a clock pulse occurs. The action of circuit is straight forward as follows.

When the clock is low, both AND gates are disabled, there fore D can change values with out affecting the value of Q. On the other hand, when the clock is high, both AND gates are enabled. In this case, Q is forced equal to D when the clock again goes low, Q retains or stores the last value of D. The truth table for such a flip flop is as given below in table 6.

Table 6: Truth table for D Flip Flop

S	R	Q(t + 1)
0	0	0
0	1	1
1	0	0
1	1	1

The excitation table for D flip flop is very simply derived given as under.

Table 7: Excitation table for D Flip Flop

S	Q
0	0
1	1

JK Flip Flop

One of the most useful and versatile flip flop is the JK flip flop the unique features of a JK flip flop are:

1. If the J and K input are both at 1 and the clock pulse is applied, then the output will change state, regardless of its previous condition.
2. If both J and K inputs are at 0 and the clock pulse is applied there will be no change in the output. There is no indeterminate condition, in the operation of JK flip flop i.e. it has no ambiguous state. The circuit diagram for a JK flip flop is shown in Figure 4.

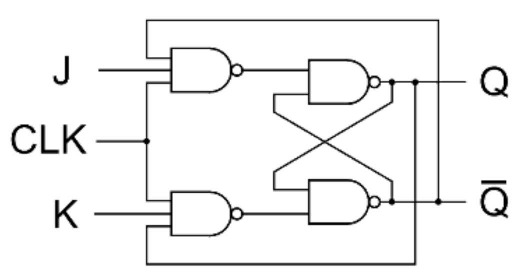


Figure 4: JK Flip Flop

When J = 0 and K = 0

These J and K inputs disable the NAND gates, therefore clock pulse have no effect on the flip flop. In other words, Q returns it last value.

When J = 0 and K = 1,

The upper NAND gate is disabled the lower NAND gate is enabled if Q is 1 therefore, flip flop will be reset (Q = 0 , $\overline{Q}$ =1)if not already in that state.

When J = 1 and K = 0

The lower NAND gate is disabled and the upper NAND gate is enabled if $\overline{Q}$ is at 1, As a result we will be able to set the flip flop (Q = 1, $\overline{Q}$= 0) if not already set

When J = 1 and K = 1

If Q = 0 the lower NAND gate is disabled the upper NAND gate is enabled. This will set the flip flop and hence Q will be 1. On the other hand if Q = 1, the lower NAND gate is enabled and flip flop will be reset and hence Q will be 0. In other words , when J and K are both high, the clock pulses cause the JK flip flop to toggle. Truth table for JK flip flop is shown in table 8.

Table 8: The truth table for the JK flip flop

Initial Conditions	**Inputs (Pulsed)**		**Final Output**
Q	**S**	**R**	**Q (t + 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

The excitation table for JK flip flop is very simply derived as given in table 9.

Table 9: Excitation table for JK Flip Flop

S	**R**	**Q**
0	0	No Change
0	1	0
1	0	1
1	1	Toggle

T Flip Flop

A method of avoiding the indeterminate state found in the working of RS flip flop is to provide only one input (the T input) such, flip flop acts as a toggle switch. Toggle means to change in the previous stage i.e. switch to opposite state. It can be constructed from clocked RS flip flop be incorporating feedback from output to input as shown in Figure 5.

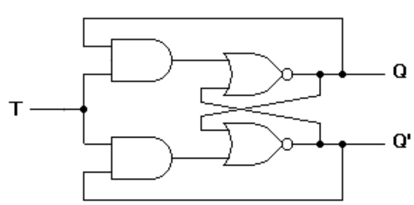


Figure 5: T Flip Flop

Such a flip flop is also called toggle flip flop. In such a flip flop a train of extremely narrow triggers drives the T input each time one of these triggers, the output of the flip flop changes stage. For instance Q equals 0 just before the trigger. Then the upper AND gate is enable and the lower AND gate is disabled. When the trigger arrives, it results in a high S input.

This sets the Q output to 1. When the next trigger appears at the point T, the lower AND gate is enabled and the trigger passes through to the R input this forces the flip flop to reset.

Since each incoming trigger is alternately changed into the set and reset inputs the flip flop toggles. It takes two triggers to produce one cycle of the output waveform. This means the output has half the frequency of the input stated another way, a T flip flop divides the input frequency by two. Thus such a circuit is also called a divide by two circuit.

A disadvantage of the toggle flip flop is that the state of the flip flop after a trigger pulse has been applied is only known if the previous state is known. The truth table for a T flip flop is as given table 7.

Table 7: Truth table for T Flip Flop

Q_n	T	$Q_n + 1$
0	0	0
0	1	1
1	0	1
1	1	0

The excitation table for T flip flop is very simply derived as shown in Table 8.

Table 8: Excitation table for T Flip Flop

T	Q
0	Q_n
1	$\overline{Q}_n$

Generally T flip flop ICs are not available. It can be constructed using JK, RS or D flip flop. Figure 6 shows the relation of T flip flop using JK flip flop.

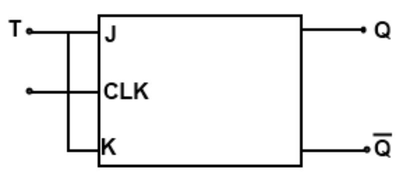


Figure 6: T Flip Flop Using JK Flip Flop

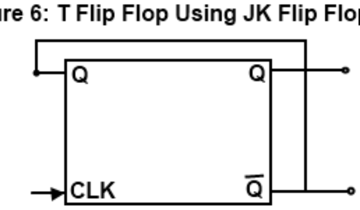


Figure 7: D-type Flip Flop connected as toggle stage

Figure 7: JK & D Flip Flop Connected as T Flip flop

A D-type flip flop may be modified by external connection as a T-type stage as shown in Figure 7. Since the Q logic is used as D-input the opposite of the Q output is transferred into the stage each clock pulse. Thus the stage having Q - 0 transistors $\overline{Q}$= 1, Providing a toggle action, if the stage had Q = 1 the clock pulse would result in Q = 0 being transferred, again providing the toggle operation. The D-type flip flop connected as in Figure 6 will thus operate as a T-type stage, complementing each clock pulse.

Master Slave Flip Flop

Figure 8 shows the schematic diagram of master sloave J-K flip flop

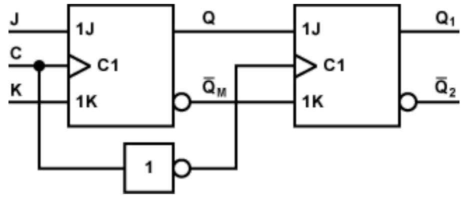


Figure 8: Master Slave JK Flip Flop

Figure 8: Master Slave JK Flip Flop

A master slave flip flop contains two clocked flip flops. The first is called master and the second slave. 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 incative during this period its output remains in the previous state. When clock becomes low the output of the slave flip flop changes because it become active during low clock period. The final output of master slave flip flop is the output of the slave flip flop. So the output of master slave flip flop is available at the end of a clock pulse.

- Clocked or Triggered Flip Flops
- Shift Registers
- Applications of Flip Flops

Category: Digital Electronics

‹ Digital To Analog Converter (DAC)

Up

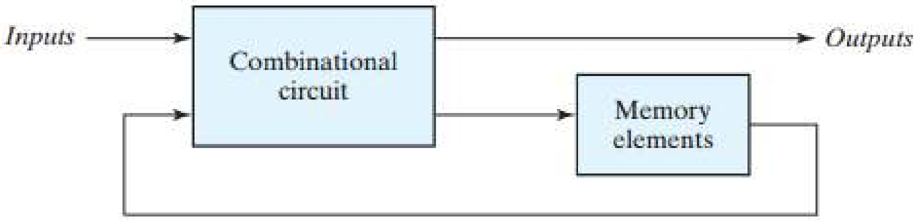
Instruction Set of Intel 8085 Microprocessor ›

Module-3

SEQUENTIAL LOGIC CIRCUITS

Till now we studied the logic circuits whose outputs at any instant of time depend only on the input signals present at that time are known as combinational circuits. Moreover, in a combinational circuit, the output appears immediately for a change in input, except for the propagation delay through circuit gates.

On the other hand, the logic circuits whose outputs at any instant of time depend on the present inputs as well as on the past outputs are called sequential circuits. In sequential circuits, the output signals are fed back to the input side. A block diagram of a sequential circuit is shown in Figure below:-



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. The binary information stored in these elements at any given time defines the ***state*** of the sequential circuit at that time. The sequential circuit receives binary information from external inputs that, together with the present state of the storage elements, determine the binary value of the outputs. These external inputs also determine the condition for changing the state in the storage elements. The block diagram demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements. The next state of the storage elements is also a function of external inputs and the present state. Thus, **a sequential circuit is specified by a time sequence of inputs, outputs, and internal states.**

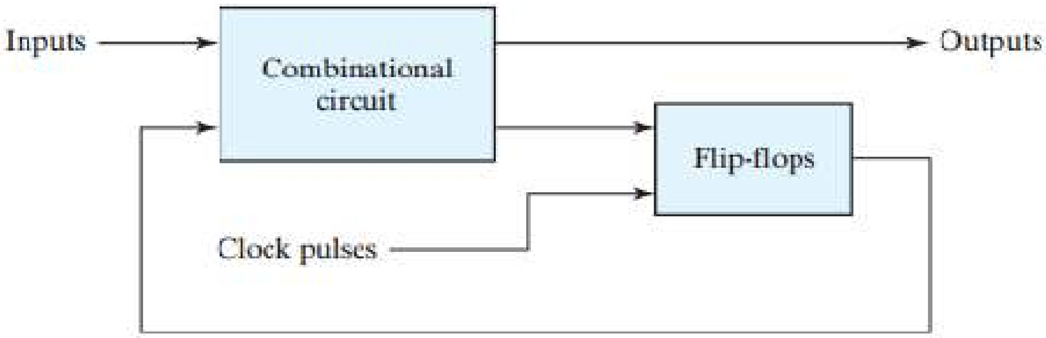
There are two types of sequential circuits, and their classification is a function of the timing of their signals.

Asynchronous sequential circuit:

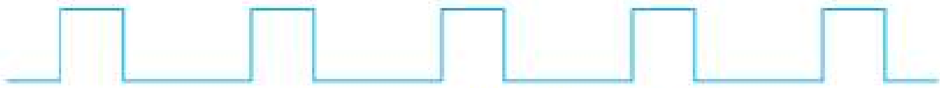
A sequential circuit whose behavior depends upon the sequence in which the input signals change is referred to as an ***asynchronous sequential circuit***. The output will be affected whenever the input changes. The commonly used memory elements in these circuits are time-delay devices. There is no need 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.



(a) Block diagram



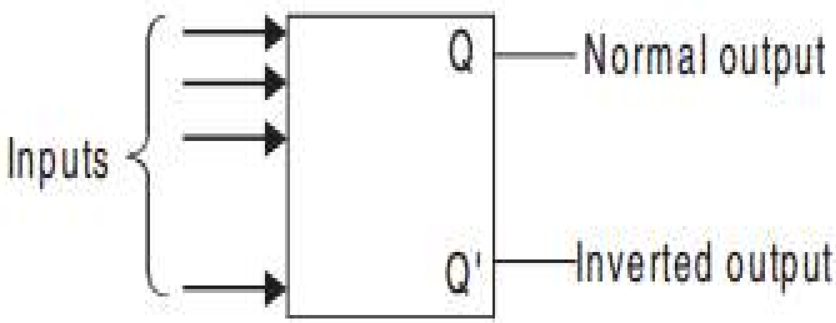
(b) Timing diagram of clock pulses

Synchronous clocked sequential circuit

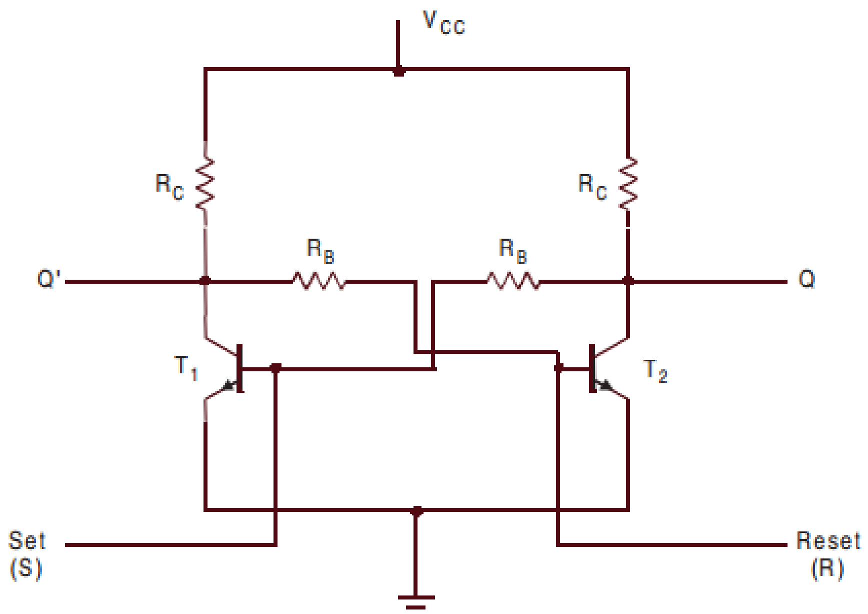
The storage elements (memory) used in clocked sequential circuits are called ***flipflops***

FLIPFLOPS

The basic 1-bit digital memory circuit is known as a flip-flop. It can have only two states, either the 1 state or the 0 state. A flip-flop is also known as a bistable multivibrator. Flip-flops can be obtained by using NAND or NOR gates. The general block diagram representation of a flip-flop is shown in Figure below. It has one or more inputs and two outputs. The two outputs are complementary to each other. If Q is 1 *i.e.*, Set, then Q' is 0; if Q is 0 *i.e.*, Reset, then Q' is 1. That means Q and Q' cannot be at the same state simultaneously. If it happens by any chance, it violates the definition of a flip-flop and hence is called an *undefined* condition. Normally, the state of Q is called the *state* of the flip-flop, whereas the state of Q' is called the *complementary state* of the flip-flop. When the output Q is either 1 or 0, it remains in that state unless one or more inputs are excited to effect a change in the output. Since the output of the flip-flop remains in the same state until the trigger pulse is applied to change the state, it can be regarded as a memory device to store one binary bit. The block diagram of a flip-flop is given below:-



The Bistable multivibrator circuit of a flip-flop is given below:-

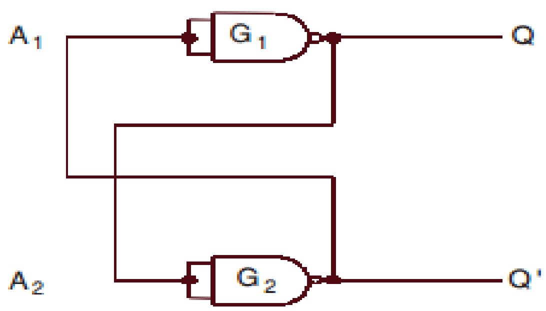


From the circuit shown in above, the multivibrator is basically two cross-coupled inverting amplifiers, consist of two transistors and four resistors. Obviously, if transistor T_1 is initially turned ON (saturated) by applying a positive signal through the Set input at its base, its collector will be at $V_{CE(sat)}$ (0.2 to 0.4 V). The collector of T_1 is connected to the base of T_2, which cannot turn T_2 On. Hence, T_2 remains OFF (cut off). Therefore, the voltage at the collector of T_2 tries to reach V_{CC}. This action only enhances the initial positive signal applied to the base of T_1. Now if the initial signal at the Set input is removed, the circuit will maintain T_1 in the ON state and T_2 in the OFF state indefinitely, *i.e.*, Q = 1 & Q' = 0. In this condition the bistable multivibrator is said to be in the **Set state**. A positive signal applied to the Reset input at the base of T_2 turns it ON. As we have discussed earlier, in the same sequence T_2 turns ON & T_1 turns OFF, resulting in a second stable state *i.e.* Q = 0 & Q' = 1. In this condition the bistable multivibrator is said to be in the ***Reset* state.**

LATCHES

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.



We consider the fundamental circuit shown in Fig.(last page). It consists of two inverters G_1 and G_2 (NAND gates are used as inverters). The output of G_1 is connected to the input of G_2 (A_2) and the output of G_2 is connected to the input of G_1 (A_1).

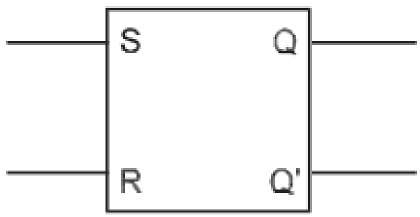
Let us assume the output of G_1 to be Q = 0, which is also the input of G_2 (A_2 = 0). So, the output of G_2 will be Q'= 1, which makes A_1 = 1 and consequently Q = 0 which is according to our assumption. Similarly, we can demonstrate that if Q = 1, then Q' = 0 and this is also consistent with the circuit 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***. In this circuit there is no way to enter the desired digital information to be stored in it. To make that possible we have to modify the circuit by replacing the inverters by NAND gates and then it becomes a flip-flop.

TYPES OF FLIP-FLOPS

There are different types of flip-flops depending on how their inputs and clock pulses cause transition between two states. We will discuss four different types of flip-flops in this chapter, *viz*., S-R, D, J-K, and T. Basically D, J-K, and T are three different modifications of the S-R flip-flop.

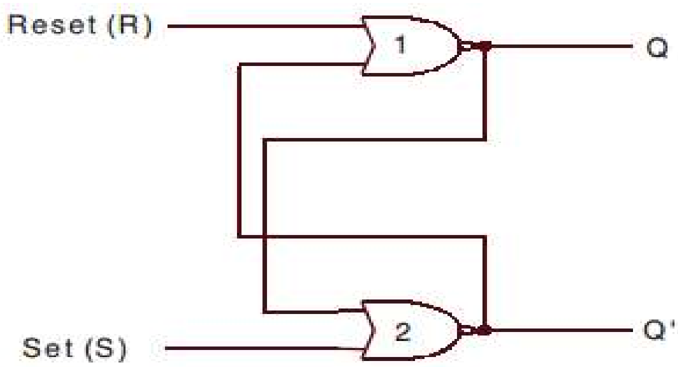
S-R (Set-Reset) Flip-flop

An S-R flip-flop has two inputs named Set (S) and Reset (R), and two outputs Q and Q'. The outputs are complement of each other, *i.e.*, if one of the outputs is 0 then the other should be 1. This can be implemented using NAND or NOR gates. The block diagram of an S-R flip-flop is shown in Figure below:-



S-R Flip-flop Based on NOR Gates

An S-R flip-flop can be constructed with NOR gates at ease by connecting the NOR gates back to back as shown in Figure below. The cross-coupled connections from the output of gate 1 to the input of gate 2 constitute a feedback path. This circuit is not clocked and is classified as an asynchronous sequential circuit. The truth table for the S-R flip-flop based on a NOR gate is shown in the table below



Inputs		Outputs		Action
S	R	Q_{n+1}	Q'_{n+1}	
0	0	Q_n	Q'_n	No change
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Forbidden (Undefined)
0	0	–	–	Indeterminate

To analyze the circuit of S-R Flip-flop Based on NOR Gates, we have to consider the fact that the output of a NOR gate is 0 if any of the inputs are 1, irrespective of the other input. The output is 1 only if all of the inputs are 0. The outputs for all the possible conditions as shown in the above table are described as follows.

Case 1. For S = 0 and R = 0, the flip-flop remains in its present state (Q_n). It means that the next state of the flip-flop does not change, *i.e.*, $Q_{n+1} = 0$ if $Q_n = 0$ and vice versa. First let us assume that $Qn = 1$ and $Q'_n = 0$.Thus the inputs of NOR gate 2 are 1 and 0, and therefore its output $Q'n+1 = 0$. This output $Q'_{n+1} = 0$ is fed back as the input of NOR gate1, thereby producing a 1 at the output, as both of the inputs of NOR gate 1 are 0 and 0; so $Q_{n+1} = 1$ as originally assumed. Now let us assume the opposite case, *i.e.,* $Q_n = 0$ and $Q'_n = 1$. Thus the inputs of NOR gate 1 are 1 and 0, and therefore its output $Q'_{n+1} = 0$. This output $Q_{n+1} = 0 =0$ is fed back as the input of NOR gate 2, thereby producing a 1 at the output, as both of the inputs of NOR gate 2 are 0 and 0; so $Q'_{n+1} = 1$ as originally assumed. Thus we find that the condition S = 0 and R = 0 do not affect the outputs of the flip-flop, which means this is the memory condition of the S-R flip-flop.

Case 2. The second input condition is S = 0 and R = 1. The 1 at R input forces the output of NOR gate 1 to be 0 (*i.e.*, $Q_{n+1} = 0$). Hence both the inputs of NOR gate 2 are 0 and 0 and so its output $Q'_{n+1} = 1$. Thus the condition S = 0 and R = 1 will always reset the flip-flop to 0. Now if the R returns to 0 with S = 0, the flip-flop will remain in the same state.

Case 3. The third input condition is S = 1 and R = 0. The 1 at S input forces the output of NOR gate 2 to be 0 (*i.e.*, $Q'_{n+1} = 0$). Hence both the inputs of NOR gate 1 are 0 and 0 and so its output $Q_{n+1} = 1$. Thus the condition S = 1 and R = 0 will always set the flip-flop to 1. Now if the S returns to 0 with R = 0, the flip-flop will remain in the same state.

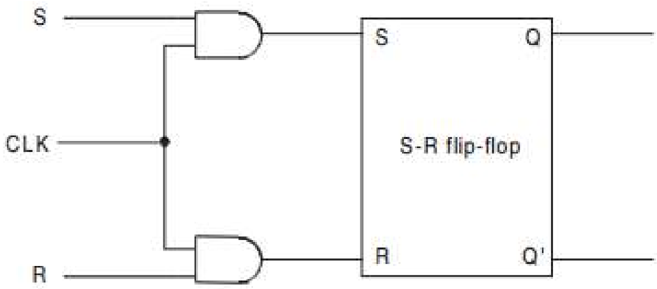
Case 4. The fourth input condition is S = 1 and R = 1. The 1 at R input and 1 at S input forces the output of both NOR gate 1 and NOR gate 2 to be 0. Hence both the outputs of NOR gate 1 and NOR gate 2 are 0 and 0; *i.e.*, $Q_{n+1} = 0$ and $Q'_{n+1} = 0$. Hence this condition S = 1 and R = 1 violates the fact that the outputs of a flip-flop will always be the complement of each other. Since the condition violates the basic definition of flip-flop, it is called the ***undefined*** condition. Generally this condition must be avoided by making sure that 1s are not applied simultaneously to both of the inputs.

Case 5. If case 4 arises at all, then S and R both return to 0 and 0 simultaneously, and then any one of the NOR gates acts faster than the other and assumes the state. For example, if NOR gate 1 is faster than NOR gate 2, then Q_{n+1} will become 1 and this will make $Q'_{n+1} = 0$. Similarly, if NOR gate 2 is faster than NOR gate 1, then Q'_{n+1} will become 1 and this will make $Q_{n+1} = 0$. Hence, this condition is determined by the flip-flop itself. Since this condition cannot be controlled and predicted it is called the ***indeterminate*** condition.

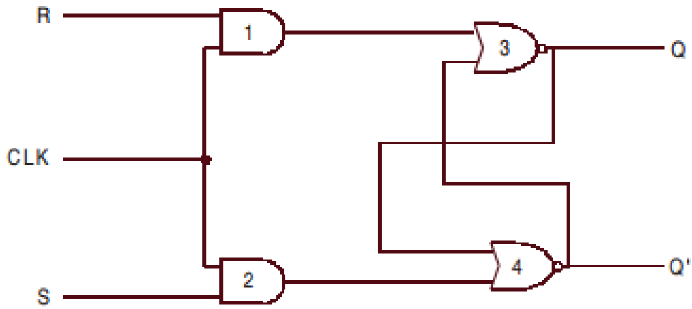
Similarly we can analyze the case of S'-R' Flip-flop Based on NAND Gates (assignment for the students).

CLOCKED S-R FLIP-FLOP

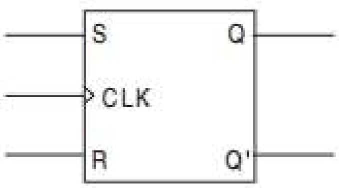
Generally, synchronous circuits change their states only when clock pulses are present. The operation of the basic flip-flop can be modified by including an additional input to control the behavior of the circuit. Such a circuit is shown below:-



The circuit shown above consists of two AND gates. The clock input is connected to both of the AND gates, resulting in LOW outputs when the clock input is LOW. In this situation the changes in S and R inputs will not affect the state (Q) of the flip-flop. On the other hand, if the clock input is HIGH, the changes in S and R will be passed over by the AND gates and they will cause changes in the output (Q) of the flip-flop. This way, any information, either 1 or 0, can be stored in the flip-flop by applying a HIGH clock input and be retained for any desired period of time by applying a LOW at the clock input. This type of flip-flop is called a ***clocked S-R flip-flop***. Such a clocked S-R flip-flop made up of two AND gates and two NOR gates is shown in Figure below:-



The logic symbol of the S-R flip-flop is shown below. It has three inputs: S, R, and CLK. The CLK input is marked with a small triangle. The triangle is a symbol that denotes the fact that the circuit responds to an edge or transition at CLK input.



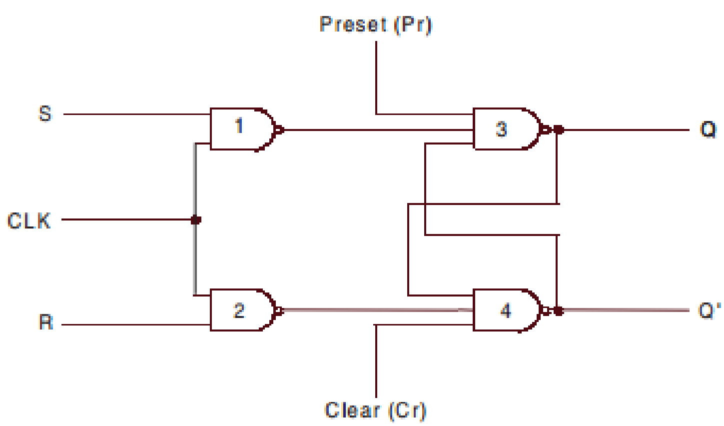
Assuming that the inputs do not change during the presence of the clock pulse, we can express the working of the S-R flip-flop in the form of the truth table shown here. Here, S_n and R_n denote the inputs and Q_n denotes the output during the bit time n. Q_{n+1} denotes the output after the pulse passes *i.e.* in the bit time $n + 1$.

Inputs		*Output*
S_n	R_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	–

Case 1. If $S_n = R_n = 0$, and the clock pulse is not applied, the output of the fl ip-fl op remains in the present state. Even if $S_n = R_n = 0$, and the clock pulse is applied, the output at the end of the clock pulse is the same as the output before the clock pulse, *i.e.*, $Q_{n+1} = Q_n$. The first row of the table indicates that situation. **Case 2.** For $S_n = 0$ and $R_n = 1$, if the clock pulse is applied (*i.e.* CLK = 1), the output of NAND gate 1 becomes 1; whereas the output of NAND gate 2 will be 0. Now a 0 at the input of NAND gate 4 forces the output to be 1 *i.e.* Q' = 1. This 1 goes to the input of NAND gate 3 to make both the inputs of NAND gate 3 as 1, which forces the output of NAND gate 3 to be 0, *i.e.*, Q = 0. **Case 3.** For $S_n = 1$ and $R_n = 0$, if the clock pulse is applied (*i.e.*, CLK = 1), the output of NAND gate 2 becomes 1; whereas the output of NAND gate 1 will be 0. Now a 0 at the input of NAND gate 3 forces the output to be 1, *i.e.*, Q = 1. This 1 goes to the input of NAND gate 4 to make both the inputs of NAND gate 4 as 1, which forces the output of NAND gate 4 to be 0, *i.e.*, Q' = 0. **Case 4.** For $S_n = 1$ and $R_n = 1$, if the clock pulse is applied (*i.e.* CLK = 1), the outputs of both NAND gate 2 and NAND gate 1 becomes 0. Now a 0 at the input of both NAND gate 3 and NAND gate 4 forces the outputs of both the gates to be 1, *i.e.*, Q = 1 and Q' = 1. When the CLK input goes back to 0 (while S and R remain at 1), it is not possible to determine the next state, as it depends on whether the output of gate 1 or gate 2 goes to 1 first.

Preset and Clear

Till now the flip-flops we discussed there when the power is switched on, the state of the circuit is uncertain. It may come to reset (Q = 0) or set (Q = 1) state. But in many applications it is required to initially set or reset the flip-flop., *i.e.*, the initial state of the flip-flop is to be assigned. This is done by using the direct or asynchronous inputs. These inputs are referred to as ***preset*** **(Pr)** and ***clear*** **(Cr)** inputs. These inputs may be applied at any time between clock pulses and is not in synchronism with the clock. Such an S-R flip-flop containing preset and clear inputs is shown in Figure below.



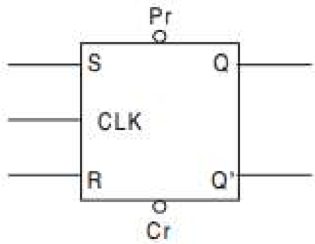
From the above Figure, we see that if Pr = Cr = 1, the circuit operates according to the table of clocked S-R flip-flop as we discussed just before.

If Pr = 1 and Cr = 0, the output of NAND gate 4 is forced to be 1, *i.e.*, Q' = 1 and the flip-flop is reset, overwriting the previous state of the flip-flop.

If Pr = 0 and Cr = 1, the output of NAND gate 3 is forced to be 1, *i.e.*, Q = 1 and the flip-flop is set, overwriting the previous state of the flip-flop. Once the state of the flip-flop is established asynchronously, the inputs Pr and Cr must be connected to logic 1 before the next clock is applied.

The condition Pr = Cr = 0 must not be applied, since this leads to an uncertain state.

The logic symbol of an S-R flip-flop with Pr and Cr inputs is shown in the side. Here, bubbles are used for Pr and Cr inputs, which indicate these are active low inputs, which means that the intended function is performed if the signal applied to Pr and Cr is LOW. The operation of the clocked S-R flip-flop is shown in the table in below. The circuit can be designed such that the asynchronous inputs override the clock, *i.e.*, the circuit can be set or reset even in the presence of the clock pulse.



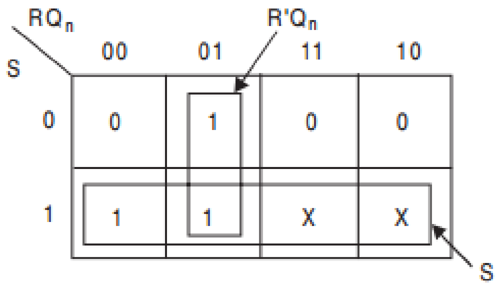
Inputs			*Output*	*Operation*
CLK	*Cr*	*Pr*	*Q*	*performed*
1	1	1	Q_{n+1} (Figure 7.3)	Normal flip-flop
0	1	0	1	Preset
0	0	1	0	Clear
0	0	0	–	Uncertain

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 (Q_{n+1}) depends on the present inputs as well as the present output (Q_n). 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.

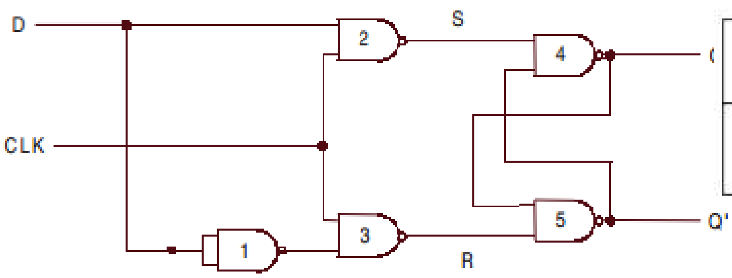
$$\boxed{SR = 0}$$

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

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

CLOCKED D FLIP-FLOP

One way to eliminate the undesirable condition of the indeterminate state in the *SR* latch is to ensure that inputs *S* and *R* are never equal to 1 at the same time. This is done in the *D* latch. The D flip-flop has only one input referred to as the D (**data**) input & two outputs as usual Q and Q'. It transfers the data at the input after the delay of one clock pulse at the output Q. So in some cases the input is referred to as a delay input and the flip-flop gets the name ***delay*** (D) flip-flop. It can be easily constructed from an S-R flip-flop by simply incorporating an inverter between S and R such that the input of the inverter is at the S end & the output of the inverter is at the R end. We can get rid of the undefined condition, *i.e.*, S = R = 1 condition, of the S-R flip-flop in the D flip flop. The D flip-flop is either used as a delay device or as a latch to store one bit of binary information. The truth table of D flip-flop is given in the table below. The structure of the D flip-flop is shown in Figure below, which is being constructed using NAND gates. The same structure can be constructed using only NOR gates.



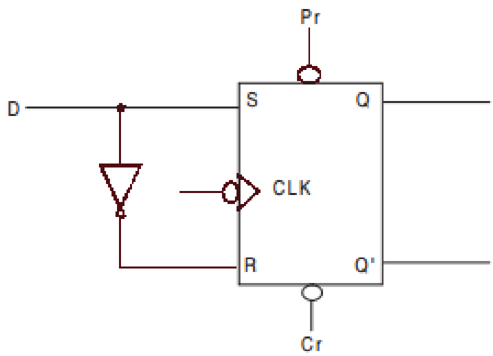
Input D_n	*Output* Q_{n+1}
0	0
1	1

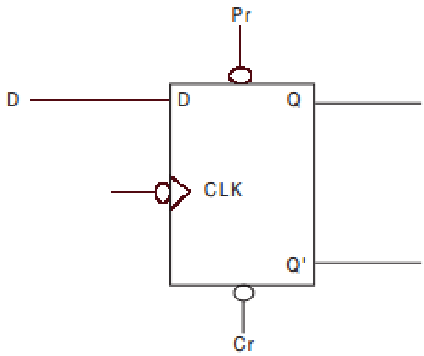
Case 1. If the CLK input is low, the value of the D input has no effect, since the S and R inputs of the basic NAND flip-flop are kept as 1.

Case 2. If the CLK = 1 and D = 1, the NAND gate 1 produces 0, which forces the output of NAND gate 3 as 1. On the other hand, both the inputs of NAND gate 2 are 1, which gives the output of gate 2 as 0. Hence, output

of NAND gate 4 is forced to be 1, *i.e.*, Q = 1, whereas both the inputs of gate 5 are 1 and the output is 0, *i.e.*, Q' = 0. Hence, we find that when D = 1, after one clock pulse passes Q = 1, which means the output follows D.
Case 3. If the CLK = 1, and D = 0, the NAND gate 1 produces 1. Hence both the inputs of NAND gate 3 are 1, which gives the output of gate 3 as 0. On the other hand, D = 0 forces the output of NAND gate 2 to be 1. Hence the output of NAND gate 5 is forced to be 1, *i.e.*, Q' = 1, whereas both the inputs of gate 4 are 1 and the output is 0, *i.e.,* Q = 0. Hence, we find that when D = 0, after one clock pulse passes Q = 0, which means the output again follows D.

A simple way to construct a D flip-flop using an S-R flip-flop is shown in Figure below. The logic symbol of a D flip-flop is shown in Figure below. A D flip-flop is most often used in the construction of sequential circuits like registers.



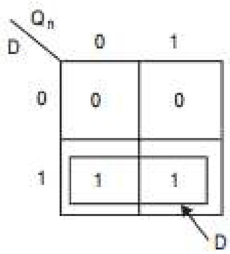


Characteristic Table of a D Flip-flop

As we have already discussed the characteristic equation of an S-R flip-flop, we can similarly find out the characteristic equation of a D flip-flop. The characteristic table of a D flip-flop is given in the table below. From the characteristic table we have to find out the characteristic equation of the D flip-flop.

Flip-flop inputs	*Present output*	*Next output*
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

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



Hence, the characteristic equation of a D flip-flop is

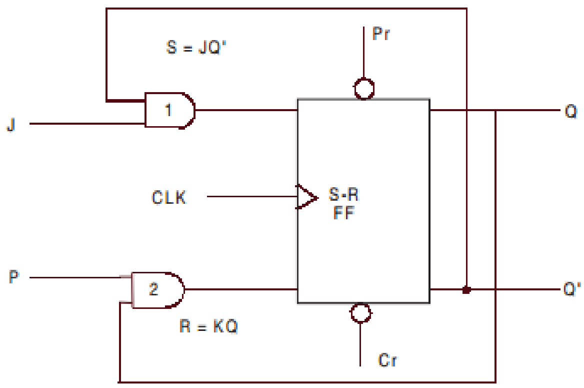
$$\boxed{Q_{n+1} = D}$$

J-K FLIP-FLOP

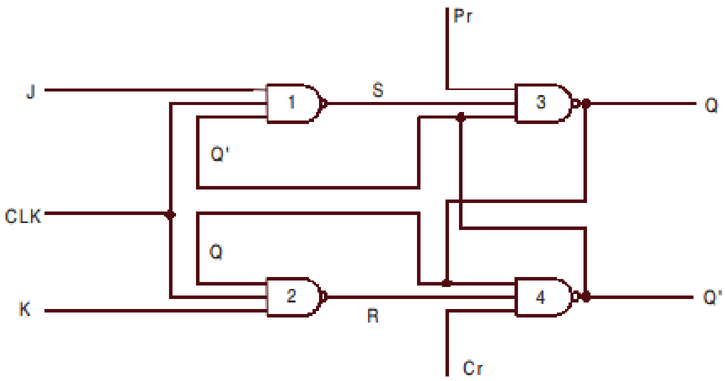
A J-K flip-flop has very similar characteristics to an S-R flip-flop. The only difference is that the undefined condition for an S-R flip-flop, *i.e.*, $S_n = R_n = 1$ condition, is also included in this case. Inputs J and K behave like inputs S and R to set and reset the flip-flop respectively. When 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 below.

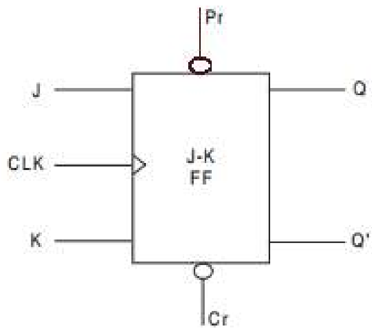
An S-R flip-flop converted into a J-K flip-flop:-



A J-K flip-flop using NAND gates:-



Logic symbol of a J-K flip-flop:-



The TRUTH table for JK flip-flop is:-

Data inputs		*Outputs*		*Inputs to S-R FF*		*Output*
J_n	K_n	Q_n	Q'_n	S_n	R_n	Q_{n+1}
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

Inputs		*Output*
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q'_n

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.

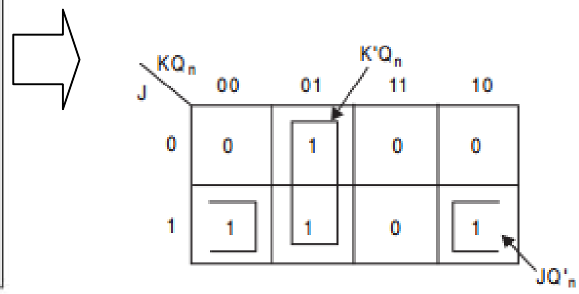
Case 3. When the clock is applied and J = 1 and K = 0 and the previous state of the flip-flop is reset (*i.e.*, $Q_n = 0$ and $Q'_n = 1$), then S = 0 and R = 1. Since S = 0 and R = 1, the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (*i.e.*, $Q_n = 1$ 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.

Characteristic Table of a J-K Flip-flop

As we have already discussed the characteristic equation of an S-R flip-flop, we can similarly find out the characteristic equation of a J-K flip-flop. The characteristic table of a J-K flip-flop is given in the table below. From the characteristic table we have to find out the characteristic equation of the J-K flip-flop.

Flip-flop inputs		*Present output*	*Next output*
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

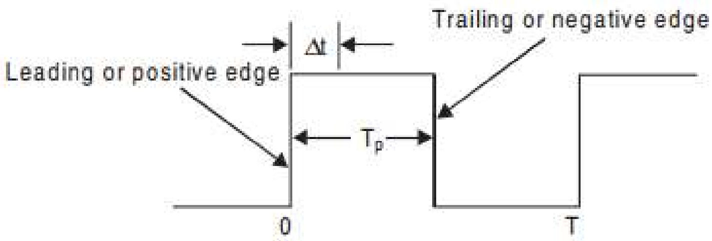


From the Karnaugh map, we obtain $\mathbf{Q_{n+1} = JQ'_n + K'Q_n}$.
Hence, the characteristic equation of a J-K flip-flop is

$$\boxed{Q_{n+1} = JQ'_n + K'Q_n}$$

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.



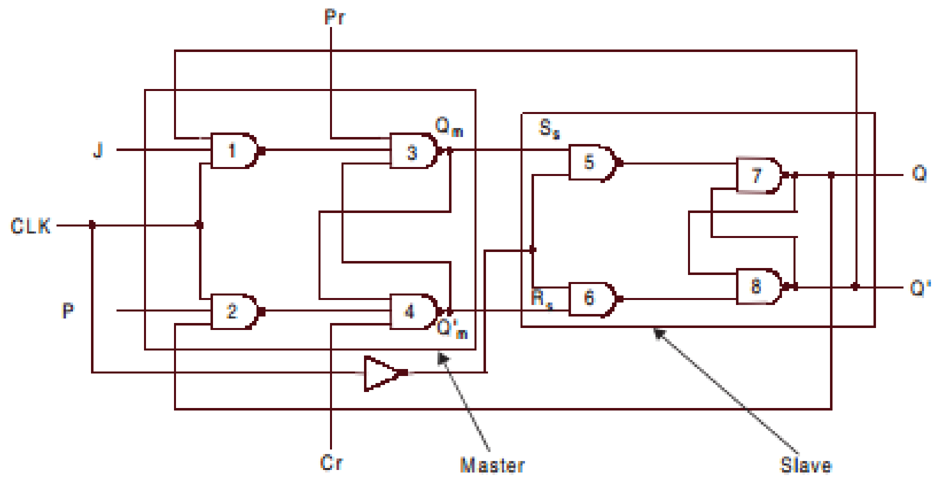
Consider, for example, that the inputs are J = K = 1 and Q = 1, and a pulse as shown above is applied at the clock input. After a time interval Δt equal to the propagation delay through two NAND gates in series, the outputs will change to Q = 0. So now we have J = K = 1 and Q = 0. After another time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

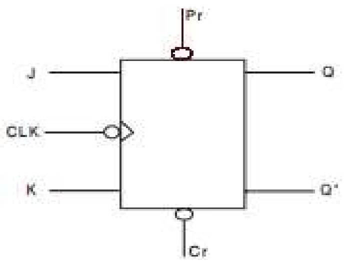
Master-Slave J-K Flip-flop

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) flip-flop.



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 master-slave 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.

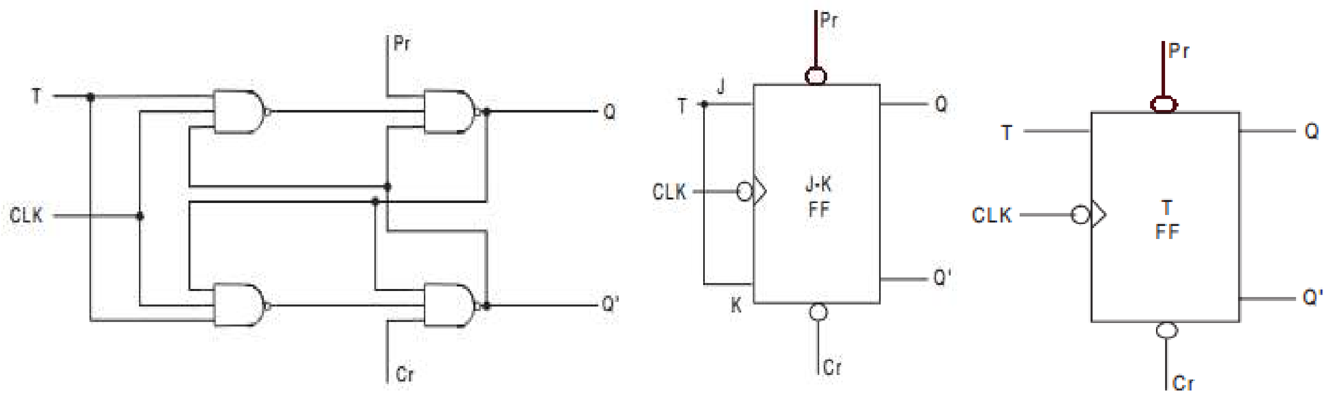
T Flip-flop

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:-

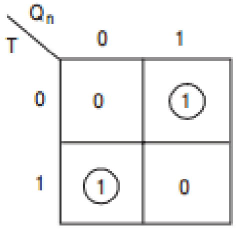


Characteristic Table of a T Flip-flop

As we have already discussed the characteristic equation of a J-K flip-flop, we can similarly find out the characteristic equation of a T flip-flop. The characteristic table of a T flip-flop is given below. From the characteristic table we have to find out the characteristic equation of the T flip-flop.

Flip-flop inputs	*Present output*	*Next output*
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

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



From the Karnaugh map, the Boolean expression of Q_{n+1} is derived as $Q_{n+1} = TQ'_n + T'Q_n$. Hence, the characteristic equation of a T flip-flop is

$$\boxed{Q_{n+1} = TQ'_n + T'Q_n}$$

TRIGGERING OF FLIP-FLOPS

Flip-fl ops 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 flip-flop 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 *edge-triggered* 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:

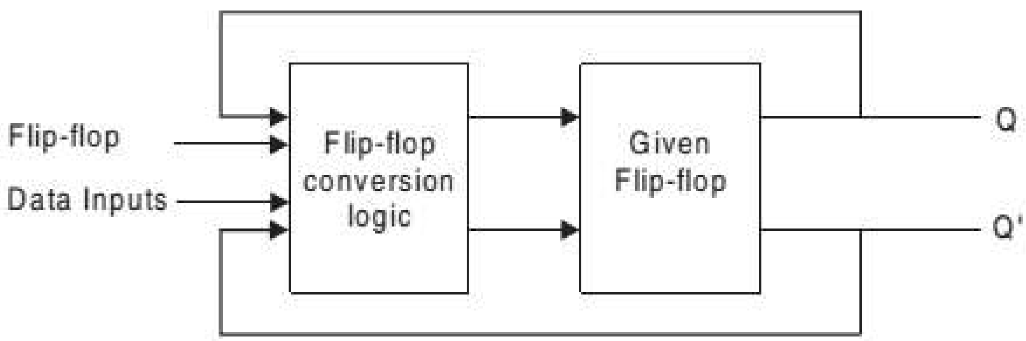
1. S = R = 0 (second row)
2. S = 1, R = 0 (sixth row).

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 flip-flop 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).



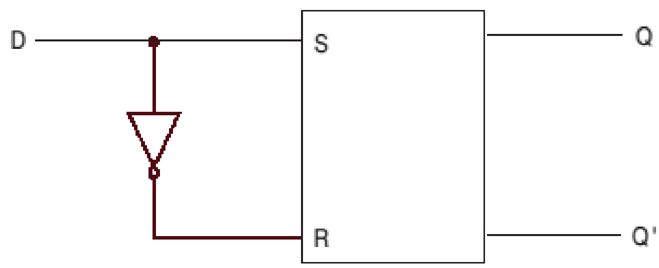
Conversion of an S-R Flip-flop to a D Flip-flop

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:-

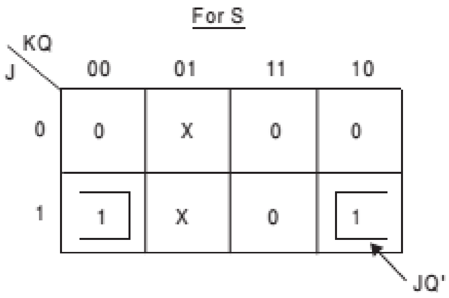


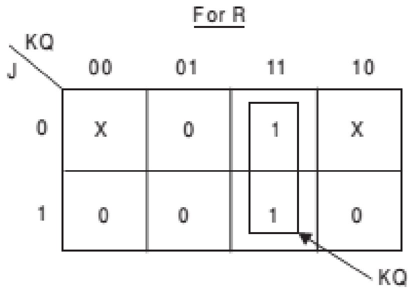
Conversion of an S-R Flip-fl op to a J-K Flip-flop

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:-

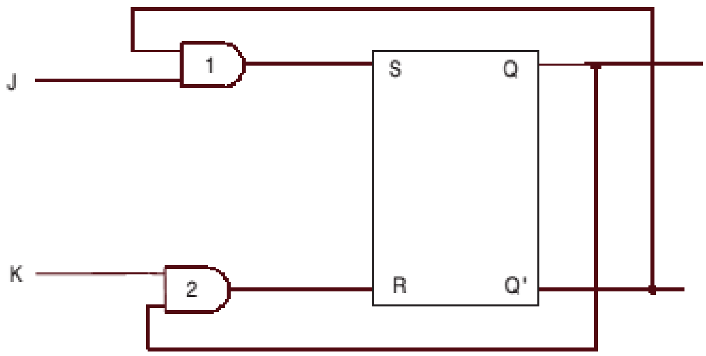




Hence we get the Boolean expression for S and R as

S = JQ'
& R = KQ.

Hence the circuit may be realized as in below:-

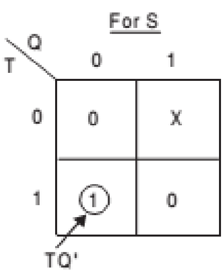


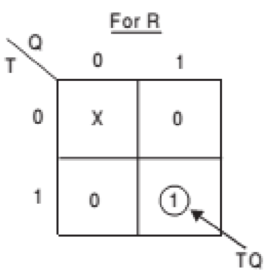
Conversion of an S-R Flip-flop to a T Flip-flop

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

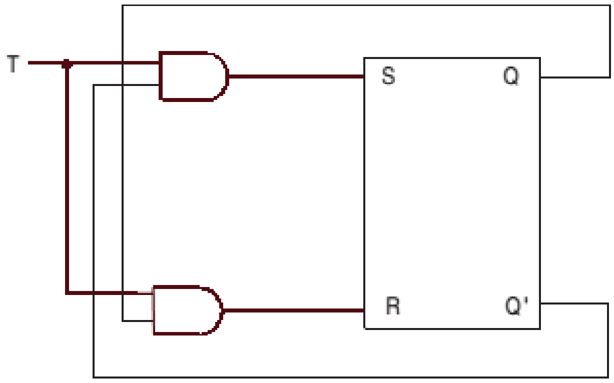
From the above truth table, the Karnaugh map is prepared as shown in Figure below:-





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

Hence the circuit may be realized as in below:-

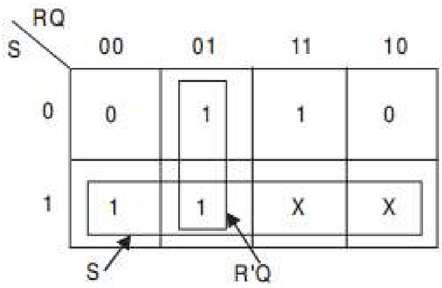


Conversion of a D Flip-flop to an S-R Flip-flop

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

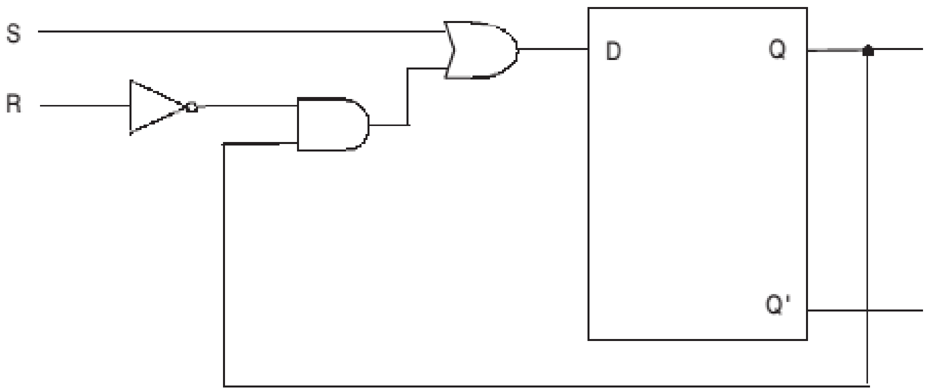
FF data inputs		*Output*	*D FF inputs*
S	*R*	*Q*	*D*
0	0	0	0
0	1	0	0
1	0	0	1
0	1	1	0
0	0	1	1
1	0	1	1

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**

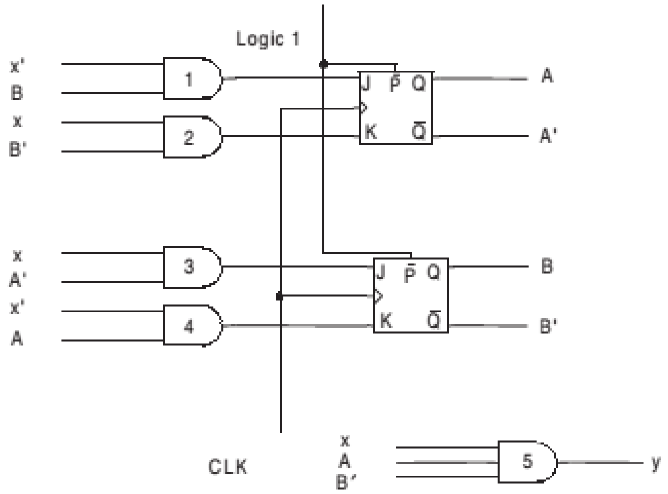
Hence the circuit may be realized as in below:-



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

ANALYSIS OF SEQUENTIAL CIRCUITS

The behavior of a sequential circuit is determined from the inputs, the outputs, and the states of the flip-flops. Both the outputs and the next state are a function of the inputs and the present state. The analysis of sequential circuits consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states. Boolean expressions can be written that describe the behavior of the sequential circuits. We first introduce a specific example of a clocked sequential circuit given below to understand its behavior.



State Table

The time sequence of inputs, outputs and flip-flop states may be enumerated in a state table. The state table for the circuit in Figure above is shown in the table in below. Here in the table there are three sections designated as present state, next state and output. The present state designates the states of the flip-flops before the occurrence of the clock pulse. The next state designates the states of the flip-flops after the application of the clock pulse. The output section shows the values of the output variables during the present state. Again, both the output and the next state sections have two columns, one for $x = 0$ and the other for $x = 1$.

The analysis of the circuit can start from any arbitrary state. In our example, we start the analysis from initial state 00. When the present state is 00, A = 0 and B = 0. From the logic diagram, with $x = 0$, we find both AND gates 1 and 2 produce logic 0 signal and hence the next state remains unchanged. Also, B flip-flop for both AND gates 3 and 4 produce logic 0 signal and hence the next state of B also remains unchanged. Hence, with the clock pulse, flip-flop A and B are both in the memory state, making the next state 00. Similarly, with A = 0 and B = 0, with $x = 1$, we find that gate 1 produces logic 0, whereas gate 2 produces logic 1. Again, with the same condition, gate 3 produces logic 1 whereas gate 4 produces logic 0. Hence, with the clock pulse, flip-flop A is cleared and B is set, making the next state 01. This information is listed in the first row of the state table.

Present state	*Next state*		*Output*	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	*AB*	*AB*	*y*	*y*
00	00	01	0	0
01	11	01	0	0
10	10	00	0	1
11	10	11	0	0

In a similar manner, we can derive the other conditions of the state table also. When the present state is 01, i.e., A = 0 & B = 1. From the logic diagram, with $x = 0$, we find gate 1 produces logic 1 signal and gate 2 produces logic 0. For B flip-flop both gates 3 & 4 produce logic 0 signal & hence the next state of B remains unchanged. Hence, with the clock pulse, flip-flop A is set and B remains in the memory state, making the next state 11. Similarly, with A = 0 and B = 1, with $x = 1$, we find that both gates 1 and 2 produce logic 0. Again, with the same condition, both gates 3 and 4 produce logic 0. Hence, with the clock pulse, both flip-flops A and B remain in the memory state, making the next state 01. This information is listed in the second row of the state table.

When the present state is 10, A = 1 and B = 0. From the logic diagram, with $x = 0$, we find both gates 1 and 2 produce logic 0. For B flip-flop gate 3 produces logic 0 signal but gate 4 produces logic 1. Hence, with the clock pulse, flip-flop A remains in the memory state and B is reset, making the next state 10. Similarly, with A = 1 and B = 0, with $x = 1$, we find that gate 1 produces logic 0, whereas gate 2 produces logic 1. Again, with the same condition, both gates 3 and 4 produce logic 0. Hence, with the clock pulse, A is reset and B remains in the memory state, making the next state 00. This information is listed in the third row of the state table.

Finally when the present state is 11, A = 1 and B = 1. From the logic diagram, with $x = 0$, we find gate 1 produces logic 1 and gate 2 produces logic 0. For B flip-flop gate 3 produces logic 0 signal but gate 4 produces logic 1. Hence, with the clock pulse, flip-flop A remains in the memory state and B is reset, making the next state 10. Similarly, with A = 1 and B = 1, with $x = 1$, we find that both gates 1 and 2 produce logic 0. Again, with the same condition, both gates 3 and 4 produce logic 0. Hence, with the clock pulse, both A and B remain in the memory state, making the next state 11. This information is listed in the last row of the state table.

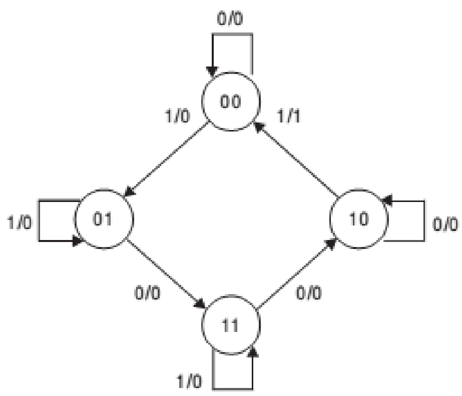
The entries in the output section are easier to derive. In this example, output y = 1 only when $x = 1$, A = 1, and B = 0. Hence the output columns are marked with 0s except when the present state is 10 and input $x = 1$, for which y is marked as 1.

The state table of any sequential circuit is obtained by the same procedure used in the example. In general, a sequential circuit with m flip-flops and n input variables will have $2m$ rows, one for each state. The next state and output sections will have $2n$ columns, one for each input combination.

The external output of a sequential circuit may come from memory elements or logic gates. The output section is only included in the state table if there are outputs from logic gates. Any external output taken directly from a flip-flop is already listed in the present state of the state table.

State Diagram

All the information available in the state table may be represented graphically in the state diagram.



In the diagram, a state is represented by a circle and the transitions between states are indicated by direct arrows connecting the circles. The binary number inside each circle identifies the state the circle represents. The direct arrows are labeled with two binary numbers separated by a /. The number before the / represents the value of the external input, which causes the state transition, and the number after the / represents the value of the output during the present state. For example, the directed arrow from the state 11 to 10 while $x = 0$ and $y = 0$, and that on the termination of the next clock pulse, the circuit goes to the next state 10. A directed arrow connecting a circle with itself indicates that no change of the state occurs.

There is no difference between a state table and a state diagram except in the manner of representation. The state table is easier to derive from a given logic diagram and the state diagram directly follows the state table. The state diagram gives a pictorial form of the state transitions and hence is easier to interpret.

State Equation

A state equation is an algebraic expression that specifies the conditions for a flip-flop state transition. The left side of the equation denotes the next state of the flip-flop and the right side a Boolean function that specifies the present state conditions that make the next state equal to 1. The state equation is derived directly from a state table. For example, the state equation for flip-flop A can be derived from the table in Figure 7.89. From the next state columns we find that flip-flop A goes to the 1 state four times: when $x = 0$ and AB = 01 or 10 or 11, or when $x = 1$ and AB = 11. This can be expressed algebraically in a state equation as follows:

$$\mathbf{A\ (t + 1) = (A'B + AB' + AB)x' + ABx}$$

Similarly, from the next state columns we find that flip-flop B goes to the 1 state four times: when $x = 0$ and AB = 01 or when $x = 1$ & AB = 00 or 01 or 11. This can be expressed algebraically in a state equation as follows:

$$\mathbf{B\ (t + 1) = A'Bx' + (A'B' + A'B + AB)x}$$

The right-hand side of the state equation is a Boolean function for the present state. When this function is equal to 1, the occurrence of a clock pulse causes flip-flop A or flip-flop B to have a next state of 1. When this function is equal to 0, the occurrence of a clock pulse causes flip-flop A or flip-flop B to have a next state of 0. The LHS of the equation identifies the flip-flop by its letter symbol, followed by the time function designation (t +1), to emphasize that this value is to be reached by the flip-flop one pulse sequence later The state equation for flip-flop A and B are simplified algebraically below. Hence, we get

$$\begin{aligned} A\ (t + 1) &= (A'B + AB' + AB)x' + ABx \\ &= (Bx')A' + AB'x' + AB \\ &= (Bx')A' + (B + B'x')A \\ &= (Bx')A' + (B + x')A \\ &= (Bx')A' + (B'x)A. \end{aligned}$$

If we let $Bx' = J$ and $B'x = K$, we obtain the relationship: $\mathbf{A\ (t + 1) = JA' + KA}$.
which is the characteristic equation of the J-K flip-flop. This relationship between the state equation and the characteristic equation can be justified from inspection of the logic diagram in the figure example of a clocked sequential circuit. In it we find that the J input of flip-flop A is equal to the Boolean function Bx' and the K input is equal to $B'x$.

Similarly, for flip-flop B we get

$$\begin{aligned} B(t+1) &= A'Bx' + (A'B' + A'B + AB)x \\ &= (A'x)B' + A'Bx' + Bx \\ &= (A'x)B' + (x + A'x')B \\ &= (A'x)B' + (x + A')B \\ &= (A'x)B' + (Ax')B. \end{aligned}$$

If we let $A'x = J$ and $Ax' = K$, we obtain the relationship: $\mathbf{B(t + 1) = JB' + KB}$ **,** which is the characteristic equation of the J-K flip-flop. In the diagram in example of a clocked sequential circuit, we find that the J input of flip-flop B is equal to the Boolean function $A'x$ and the K input is equal to Ax'.

DESIGN PROCEDURE OF SEQUENTIAL CIRCUITS

The design of a sequential circuit follows certain steps. The steps may be listed as follows:

1. The word description of a circuit may be given accompanied with a state diagram, or timing diagram, or other pertinent information.
2. Then from the given state diagram the state table has to be prepared.
3. If the state reduction mechanism is possible, then the number of states may be reduced.
4. After state reduction, assign binary values to the states if the states contain letter symbols.
5. Then the number of flip-flops required is to be determined. Each flip-flop is assigned a letter symbol.
6. Then the choice has to be made regarding the type of flip-flop to be used.
7. With the help of a state table and the flip-flop excitation table the circuit excitation and the output tables have to be determined.
8. Then using some simplification technique *e.g.*, a Karnaugh map or some other method, the circuit output functions and the flip-flop input functions have to be determined.
9. Then the logic diagram has to be drawn.

Although certain steps have been specified for designing the sequential circuit, the procedure can be shortened with experience. A sequential circuit is made up of flip-flops and combinational gates. One of the most important parts is the choice of flip-flop. From the excitation table of different flip-flops we see that the J-K flip-flop excitation table contains the maximum number of don’t-care conditions. Hence, for designing any sequential circuit, it will be most simplified if the circuit is designed with, J-K flip-flop.

The number of flip-flops is determined by the number of states. A circuit may have unused binary states if the total number of states is less than 2*m*. The unused states are taken as don’t-care conditions during the design of the combinational part of the circuit.

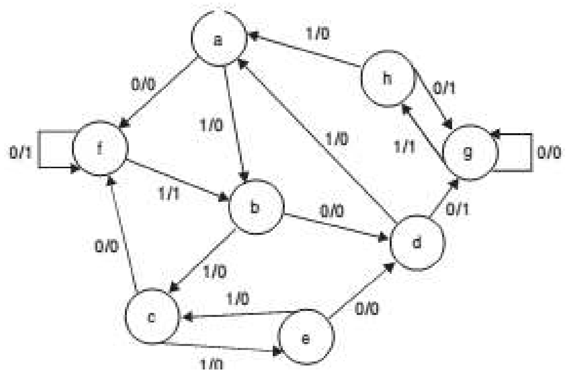
Any design process must consider the problem of minimizing the cost of the final circuit. The most obvious cost reductions are reductions in the number of flip-flops and the number of gates. The reduction of the number of flip-flops in a sequential circuit is referred to as the state reduction. Since *m* flip-flops produce 2*m* states, a reduction in the number of states may (or may not) result in a reduction of the number of flip-flops. State reduction algorithms are concerned with procedures for reducing the number of states in a state table while keeping the external input-output requirements unchanged. An algorithm for the state reduction is given here. If two states in a state table are equivalent, one of them can be removed without altering the input-output relationships.

SEQUENTIAL LOGIC CIRCUITS

Now two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit to the same state or to an equivalent state. We will discuss the state reduction problem with an example in this section later on.

In certain cases the states are specified in letter symbols. In such cases there comes another factor, called state assignment. State assignment procedures are concerned with methods for assigning binary values to states in such a way as to reduce the cost of the combinational circuit that drives the flip-flop. For any problem there may be a number of different state assignments leading to different combinational parts of the sequential circuit. The most common criterion is that the chosen assignment should result in a simple combinational circuit for the flip-flop inputs. However, to date, there are no state assignment procedures that guarantee a minimal-cost combinational circuit.

We now wish to design the clocked sequential circuit whose state diagram is given below:-



The state table for the state diagram shown above is shown in the table in Figure below.

Present state	*Next state* $x = 0$	*Next state* $x = 1$	*Output* $x = 0$	*Output* $x = 1$
a	*f*	*b*	0	0
b	*d*	*c*	0	0
c	*f*	*e*	0	0
d	*g*	*a*	1	0
e	*d*	*c*	0	0
f	*f*	*b*	1	1
g	*g*	*h*	0	1
h	*g*	*a*	1	0

We now look for two equivalent states, & find that *d* & *h* are two such states; they both go to *g* & *a* and have outputs of 1 and 0 for $x = 0$ & $x = 1$, respectively. Therefore, states *d* and *h* are equivalent; one can be removed. Similarly, we find that *b* and *e* are again two such states; they both go to *d* and *c* and have outputs of 0 and 0 for

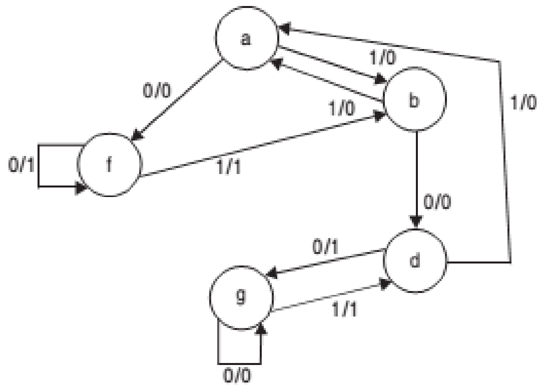
x = 0 and x = 1, respectively. Therefore, states b and e are also equivalent; and one can be removed. The procedure of removing a state and replacing it by its equivalent is demonstrated in the table in Figure below. From the below table we find that present state c now has next states f and b and outputs 0 and 0 for x = 0 and x = 1, respectively. The same next states and outputs appear in the row with present state a. Therefore, states a and c are equivalent; state c can be removed and replaced by a.

Present state	*Next state* $x = 0$	*Next state* $x = 1$	*Output* $x = 0$	*Output* $x = 1$
a	f	b	0	0
b	d	~~e~~ a	0	0
~~c~~	f	~~e~~ b	0	0
d	g	a	1	0
~~e~~	d	c	0	0
f	f	b	1	1
g	g	~~h~~ d	0	1
~~h~~	g	a	1	0

The final reduced state table is shown in below:-

Present state	*Next state* $x = 0$	*Next state* $x = 1$	*Output* $x = 0$	*Output* $x = 1$
a	f	b	0	0
b	d	a	0	0
d	g	a	1	0
f	f	b	1	1
g	g	d	0	1

The state diagram for the reduced state table consists of only five states and is shown in Figure below:-



We now assign the different states the binary values. As we have already discussed, there may be a variety of state assignments. Some of them are shown in the below table. Among them we may choose any of them and accordingly design the circuit.

State	Assignment 1	Assignment 2	Assignment 3	Assignment 4
a	000	001	111	011
b	001	010	001	101
d	010	011	110	111
f	011	100	101	001
g	100	101	010	000

In the table in Figure below, we have used binary assignment 1 to substitute the letter symbols of the five states. It is obvious that a different binary assignment will result in a state table, with completely new binary values for the states while the input-output relationships will remain the same. We will now show the procedure for obtaining the excitation table and the combinational gate structure.

Present state	Next state		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	011	001	0	0
001	010	000	0	0
010	100	000	1	0
011	011	001	1	1
100	100	010	0	1

The derivation of the excitation table is facilitated if we arrange the state table in a different form. This form is shown in the below table, where the present state and the input variables are arranged in the form of a truth table. As we have previously said, we may use any flip-flop, but the simplest form of the circuit is possible with J-K flip-flops. So we now design the circuit using J-K flip-flops.

Present state			Input	Next state			Flip-flop inputs						Output
A	*B*	*C*	*x*	*A*	*B*	*C*	*JA*	*KA*	*JB*	*KB*	*JC*	*KC*	*y*
0	0	0	0	0	1	1	0	X	1	X	1	X	0
0	0	0	1	0	0	1	0	X	0	X	1	X	0
0	0	1	0	0	1	0	0	X	1	X	X	1	0
0	0	1	1	0	0	0	0	X	0	X	X	1	0
0	1	0	0	1	0	0	1	X	X	1	0	X	1
0	1	0	1	0	0	0	0	X	X	1	0	X	0
0	1	1	0	0	1	1	0	X	X	0	X	0	1
0	1	1	1	0	0	1	0	X	X	1	X	0	1
1	0	0	0	1	0	0	X	0	0	X	0	X	0
1	0	0	1	0	1	0	X	1	1	X	0	X	1

There are three unused states in this circuit: binary states 101, 110, and 111. When an input of 0 or 1 is included with these unused states, we obtain six don’t-care terms. These six binary combinations are not listed in the table under the present state or input and are treated as don’t-care terms.

In below figure Karnaugh maps are prepared for JA, KA, JB, KB, JC, and KC.

For JA

AB \ Cx	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	X	X	X	X

For KA

AB \ Cx	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	0	1	X	X

From the Karnaugh maps for JA and KA, we obtain

$$JA = BC'x' \quad \text{and}$$

$$KA = x.$$

For JB

AB \ Cx	00	01	11	10
00	1	0	0	1
01	X	X	X	X
11	X	X	X	X
10	0	1	X	X

For KB

AB \ Cx	00	01	11	10
00	X	X	X	X
01	1	1	1	0
11	X	X	X	X
10	0	X	X	X

The Boolean expressions are derived for JB and KB from the Karnaugh maps as

$$JB = Ax + A'x' \text{ and}$$

$$KB = C' + x.$$

For JC

AB \ Cx	00	01	11	10
00	1	1	X	X
01	0	0	X	X
11	X	X	X	X
10	0	0	X	X

For KC

AB \ Cx	00	01	11	10
00	X	X	1	1
01	X	X	0	0
11	X	X	X	X
10	X	X	X	X

Similarly, the expressions for JC and KC we obtain as

$$JC = A'B' \quad \text{and}$$

$$KC = B'.$$

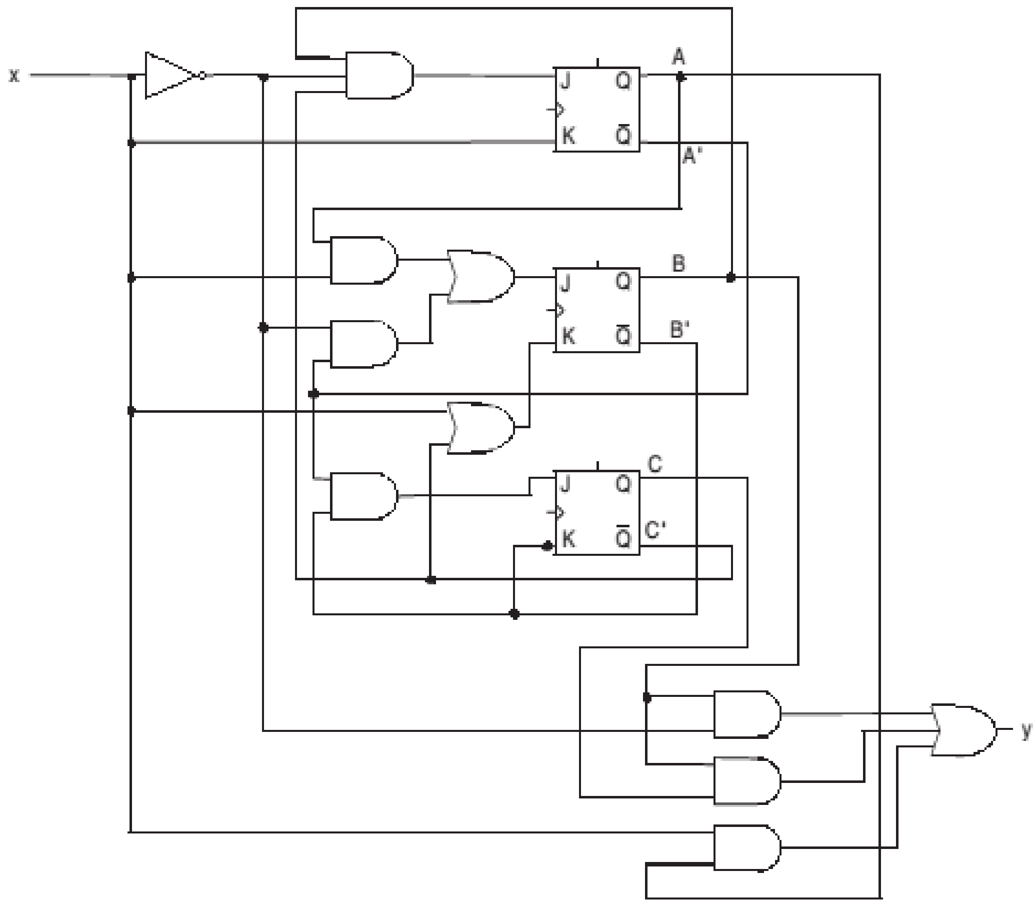
A Karnaugh map has been also prepared below for output *y* and the Boolean expression for *y* is obtained as

Y = B*x*' + BC + A*x*

For Y

AB \ Cx	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	X	X	X	X
10	0	1	X	X

The circuit diagram of the desired sequential logic network is shown in Figure below:-



REGISTERS

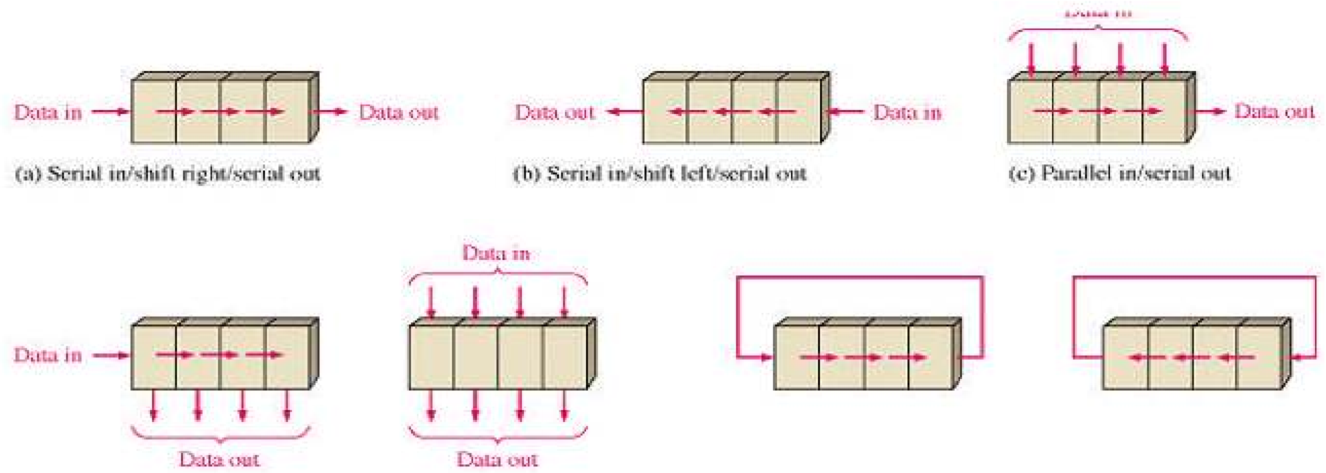
A register is a group of binary storage cells capable of holding binary information. A group of flip-flops constitutes a register, since each flip-flop can work as a binary cell. An *n*-bit register, has n flip-flops and is capable of holding n-bits information. In addition to flip-flops a register can have a combinational part that performs data-processing tasks.

Register:

- A set of n flip-flops
- Each flip-flop stores one bit
- Two basic functions: data storage and data movement.

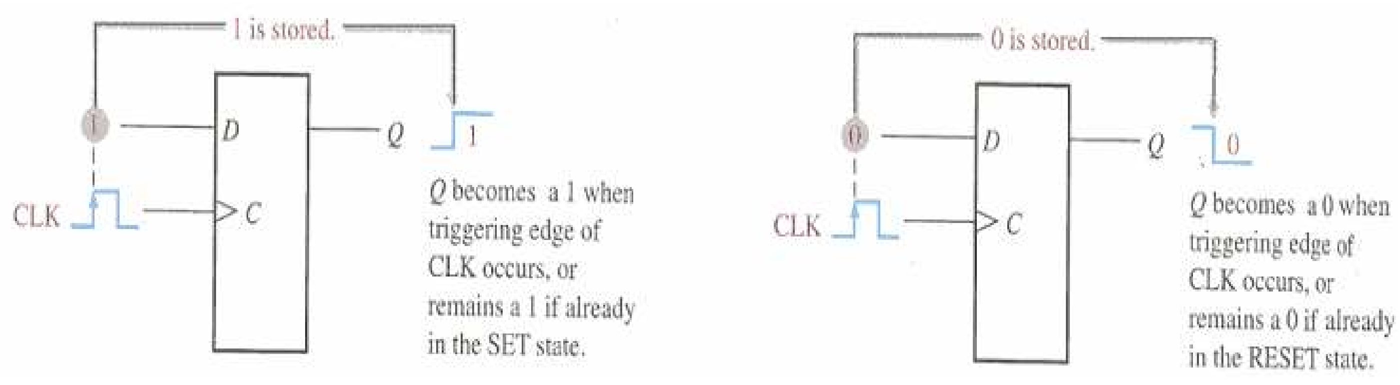
Shift Register: A register that allows each of the flip-flops to pass the stored information to its adjacent neighbor.
Counter: A register that goes through a predetermined sequence of states.

Basic data movement operation in shift registers



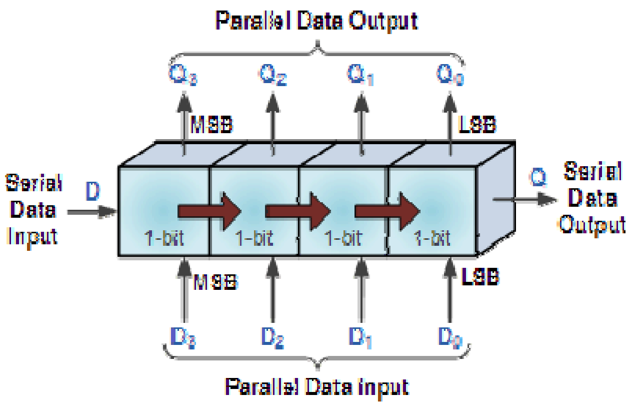
(a) Serial in/shift right/serial out
(b) Serial in/shift left/serial out
(c) Parallel in/serial out
(d) Serial in/parallel out
(e) Parallel in/parallel out
(f) Rotate right
(g) Rotate left

Storage Capacity of a register



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.

The effect of data movement from left to right through a shift register can be presented graphically as:



Shift Register

A shift register is a storage device that used to store binary data. When a number of flip flop are connected in series it is called a register. A single flip flop is supposed to stay in one of the two stable states 1 or 0 or in other words the flip flop contains a number 1 or 0 depending upon the state in which it is. A register will thus contain a series of bits which can be termed as a word or a byte.

If in these registers the connection is done in such a way that the output of one of the flip flop forms in input to other, it is known as a **shift register**. The data in a shift register is moved serially (one bit at a time).

The shift register can be built using RS, JK or D flip-flops various types of shift registers are available some of them are given as under.

1. Shift Left Register
2. Shift Right Register
3. Shift Around Register
4. Bi-directional Shift Register

There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic types of registers:-

1. Serial in/Serial out (SISO)
2. Serial in/Parallel out (SIPO)
3. Parallel in/Serial out (PISO)
4. Parallel in/Parallel out (PIPO)

SERIAL-IN--SERIAL-OUT SHIFT REGISTER

From the name itself it is obvious that this type of register accepts data serially, *i.e.*, one bit at a time at the single input line. The output is also obtained on a single output line in a serial fashion. The data within the register may be shifted from left to right using *shift-left* register, or may be shifted from right to left using *shift-right* register.

Shift-right Register

A shift-right register can be constructed with either J-K or D flip-flops as shown in Figure 8.3. A J-K flip-flop based shift register requires connection of both J and K inputs. Input data are connected to the J and K inputs of the left most (lowest order) flip-fl op. To input a 0, one should apply a 0 at the J input, *i.e.*, J = 0 and K = 1 and vice versa. With the application of a clock pulse the data will be shifted by one bit to the right.

In the shift register using D flip-flop, D input of the left most flip-flop is used as a serial input line. To input 0, one should apply 0 at the D input and vice versa.

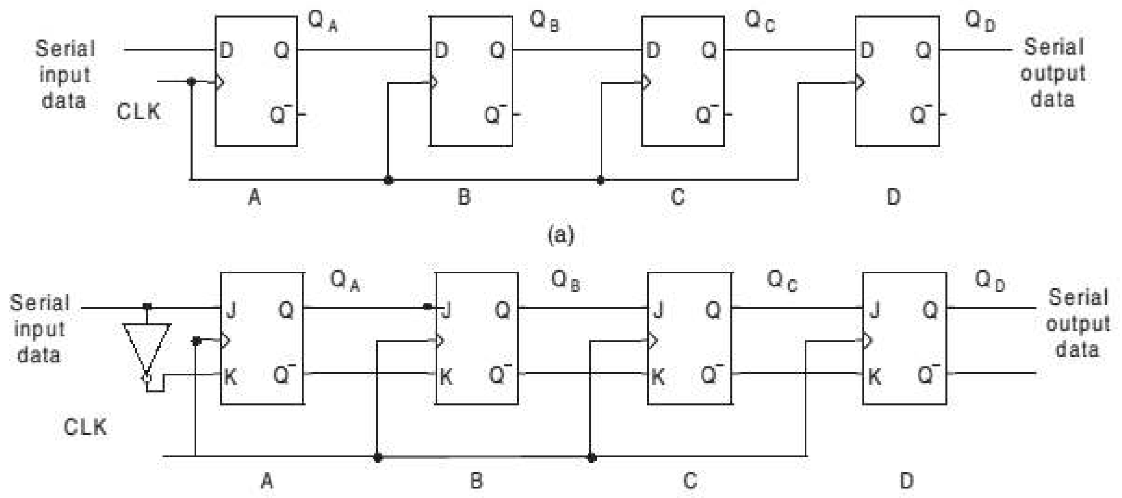


Figure of Shift-right register (a) using D flip-flops, (b) using J-K flip-flops.

The clock pulse is applied to all the flip-flops simultaneously. When the clock pulse is applied, each flip-flop is either set or reset according to the data available at that point of time at the respective inputs of the individual flip-flops. Hence the input data bit at the serial input line is entered into flip-flop A by the first clock pulse. At the same time, the data of stage A is shifted into stage B and so on to the following stages. For each clock pulse, data stored in the register is shifted to the right by one stage. New data is entered into stage A, whereas the data present in stage D are shifted out (to the right).

Operation of the Shift-right Register:-

Timing pulse	Q_A	Q_B	Q_C	Q_D	*Serial output at* Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	1	0	0	0	0
After 2nd clock pulse	1	1	0	0	0
After 3rd clock pulse	0	1	1	0	0
After 4th clock pulse	1	0	1	1	1

For example, consider that all the stages are reset and a logical input 1011 is applied at the serial input line connected to stage A. The data after four clock pulses is shown in above Table.

Let us now illustrate the entry of the 4-bit number 1011 into the register, beginning with the right-most bit. A 1 is applied at the serial input line, making D = 1. As the first clock pulse is applied, flip-flop A is SET, thus storing the 1. Next, a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 1 for flip-flop B also, because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is "shifted" to the flip-flop A and the 1 in the flip-flop A is "shifted" to flip-flop B. The 0 in the binary number is now applied at the serial input line, and the third clock pulse is now applied. This 0 is entered in flip-flop A and the 1 stored in flip-flop A is now "shifted" to flip-flop B and the 1 stored in flip-flop B is now "shifted" to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 0 stored in flip-flop A is now "shifted" to flip-flop B and the 1 stored in flip-flop B is now "shifted" to flip-flop C and the 1 stored in flip-flop C is now "shifted" to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now completed.

From the third column of above Table we can get the serial output of the data that is being entered in the register. We find that after the first, second, and the third clock pulses the output at the serial output line *i.e.*, Q_D is 0. After the fourth clock pulse the output at the serial output line is 1. If we want to get the total data that we have entered in the register in a serial manner from Q_D, then we have to apply another three clock pulses. After the fifth clock pulse we will gate another 1 at Q_D. After the sixth clock pulse the output at Q_D will be 0 and after the seventh clock pulse the output at Q_D will be 1. In this process of the fifth, sixth, and the seventh clock pulses if no data is being supplied at the serial input line then the A, B, and C flip-flops will again be RESET with output 0.

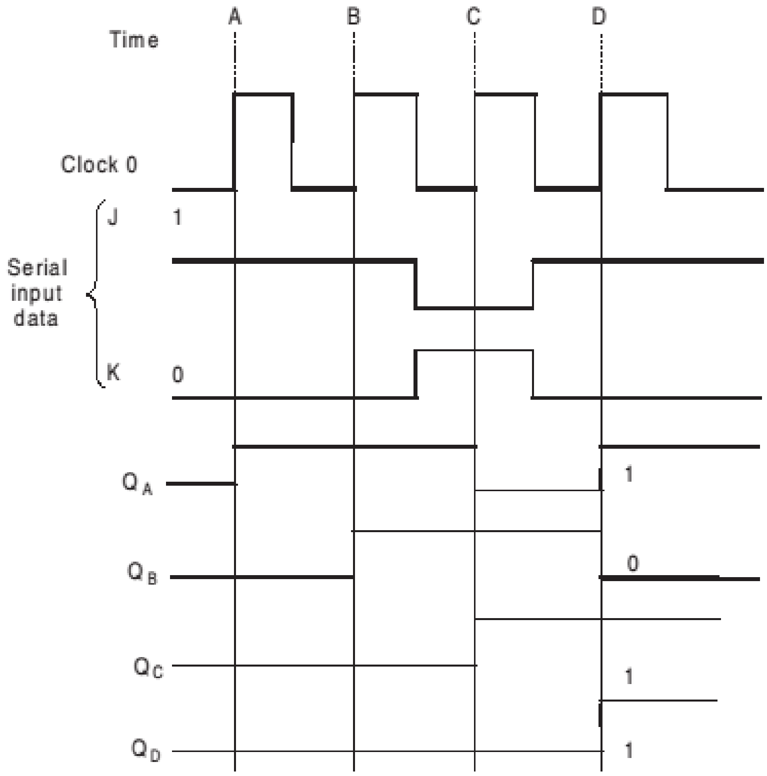


Figure:-Waveforms of 4-bit serial input shift-right register.

The waveforms shown above illustrate the entry of a 4-bit number 1011. For a J-K flip-flop, the data bit to be shifted into the flip-flop must be present at the J and K inputs when the clock transitions from low to high occur. Since the data bit is either 1 or 0, there can be two different cases:

1. To shift a 1 into the flip-flop, J = 1 and K = 0,
2. To shift a 0 into the flip-flop, J = 0 and K = 1.

At time A: All the flip-flops are reset. At the serial data input line a 1 is given and with the first clock pulse this 1 is shifted at Q_A making $Q_A = 1$. At the same time the 0 in Q_A is shifted to Q_B, and the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time A are $Q_AQ_BQ_CQ_D = 1000$.

2.

At time B: The flip-flop A contains 1, & all other flip-flop contains 0. Now, again, 1 is given at the serial data input line. With the 2nd clock pulse this 1 is shifted to Q_A. The 1 in Q_A is shifted to Q_B & the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time B are $Q_AQ_BQ_CQ_D = 1100$.

3.

At time C: The flip-flop A & B contain 1, & all other flip-flops contain 0. Now a 0 is given at the serial data input line. With the 3rd clock pulse this 0 is shifted to Q_A. The 1 in Q_A is shifted to Q_B & the 1 in Q_B is shifted to Q_C & the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are $Q_AQ_BQ_CQ_D = 0110$.

4.

At time D: The flip-flop B and flip-flop C contain 1, and all other flip-flops contain 0. Now another 1 is given at the serial data input line. With the fourth clock pulse this 1 is shifted to Q_A. The 0 in Q_A is shifted to Q_B and the 1 in Q_B is shifted to Q_C and the 1 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are $Q_AQ_BQ_CQ_D = 1011$. To summarize, we have shifted 4 data bits in a serial manner into four flip-flops. These 4 data bits could represent a 4-bit binary number 1011, assuming that we began shifting with the LSB first. Notice that the LSB is in D and the MSB is in A. These four flip-flops could be defined as a 4-bit shift register.

Shift-left Register

A shift-left register can also be constructed with either J-K or D flip-flops as shown in Figure below. Let us now illustrate the entry of the 4-bit number 1110 into the register, beginning with the right-most bit. A 0 is applied at the serial input line, making D = 0. As the first clock pulse is applied, flip-fl op A is RESET, thus storing the 0. Next a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 0 for flip-flop B, because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is "shifted" to the flip-flop A and the 0 in the flip-flop A is "shifted" to flip-flop B. The 1 in the binary number is now applied at the serial input line, and the third clock pulse is now applied. This 1 is entered in flip-flop A and the 1 stored in flip-flop A is now "shifted" to flip-flop B and the 0 stored in flip-flop B is now "shifted" to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 1 stored in flip-flop A is now "shifted" to flip-flop B and the 1 stored in flip-flop B is now "shifted" to flip-flop C and the 0 stored in flip-flop C is now "shifted" to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now completed.

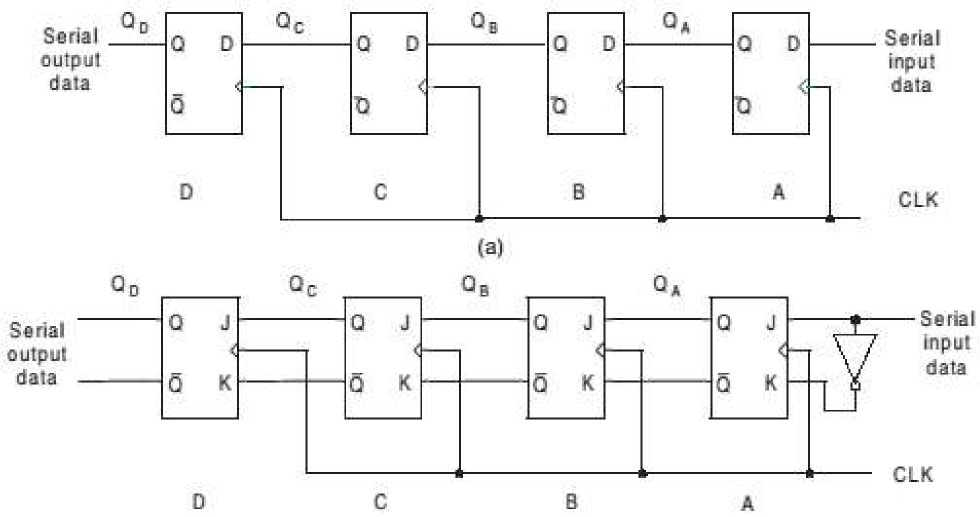


Figure:- Shift-left register (*a*) using D flip-flops, (*b*) using J-K flip-flops.

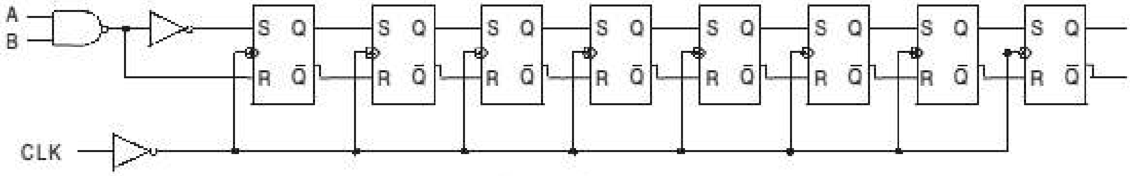
Timing pulse	Q_D	Q_C	Q_B	Q_A	*Serial output at* Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	0	0	0	0	0
After 2nd clock pulse	0	0	0	1	0
After 3rd clock pulse	0	0	1	1	0
After 4th clock pulse	0	1	1	1	0

8-bit Serial-in–Serial-out Shift Register

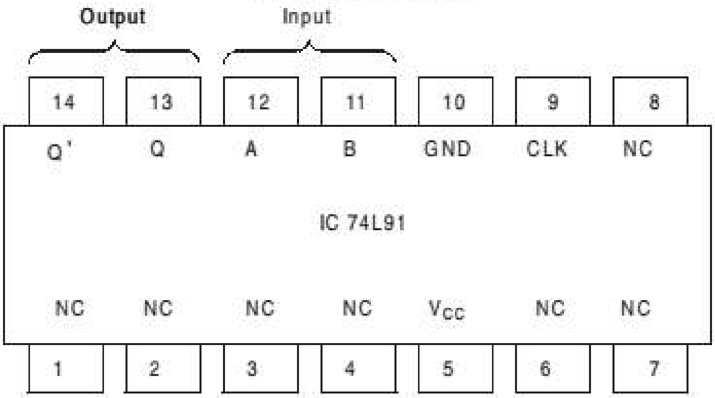
The pinout and logic diagram of IC 74L91 is shown in Figure below. This IC is actually an example of an 8-bit serial-in–serial-out shift register. There are eight S-R flip-flops connected to provide a serial input as well as a serial output. The clock input at each flip-flop is negative edge-triggered. However, the applied clock signal is passed through an inverter. Hence the data will be shifted on the positive edges of the input clock pulses.

An inverter is connected in between R and S on the first flip-flop. This means that this circuit functions as a D-type flip-flop. So the input to the register is a single liner on which the data can be shifted into the register appears serially. The data input is applied at either A (pin 12) or B (pin 11). The data level at A (or B) is complemented by the NAND gate and then applied to the R input of the first flip-flop. The same data level is complemented by the NAND gate and then again complemented by the inverter before it appears at the S input. So, a 0 at input A will *reset* the first flip-flop (in other words this 0 is shifted into the first flip-flop) on a positive clock transition.

The NAND gate with A and B inputs provide a gating function for the input data stream if required, if gating is not required, simply connect pins 11 and 12 together and apply the input data stream to this connection.



(a) Logic diagram.



(b) Pinout diagram of IC 74L91.

SERIAL-IN–PARALLEL-OUT REGISTER

In this type of register, the data is shifted in serially, but shifted out in parallel. To obtain the output data in parallel, it is required that all the output bits are available at the same time. This can be accomplished by connecting the output of each flip-flop to an output pin. Once the data is stored in the flip-flop the bits are available simultaneously. The basic configuration of a serial-in–parallel-out shift register is shown in below.

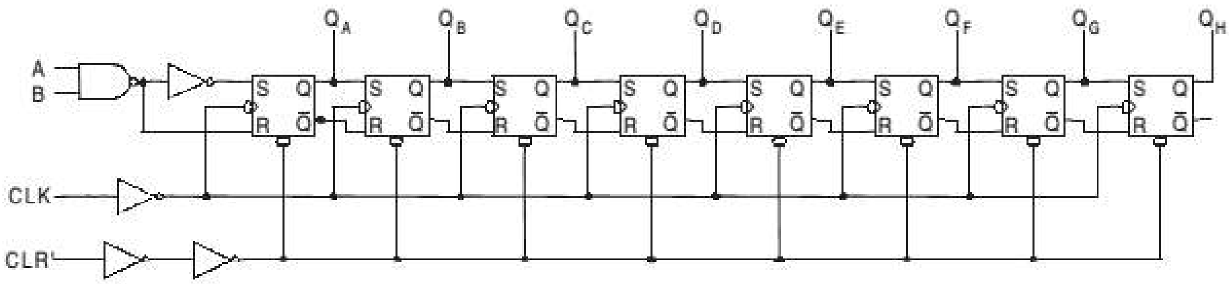
8-bit Serial-in–Parallel-out Shift Register

The pinout and logic diagram of IC 74164 is shown in Figure below. IC 74164 is an example of an 8-bit SIPO shift register. There are eight S-R flip-flops, which are all sensitive to negative clock transitions. The logic diagram in Figure below is almost the same as shown in SISO with only two exceptions: (1) each flip-flop has an asynchronous CLEAR input; and (2) the true side of each flip-flop is available as an output—thus all 8 bits of any number stored in the register are available simultaneously as an output (this is a parallel data output).

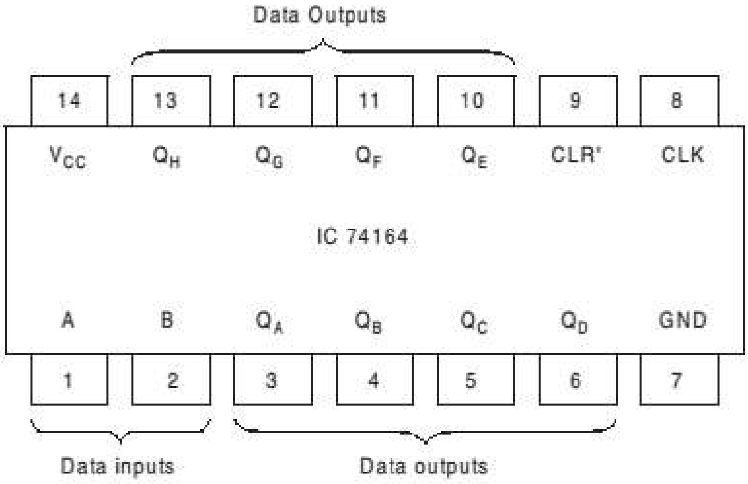
Hence, a low level at the CLR input to the chip (pin 9) is applied through an amplifier and will reset every flip-flop. As long as the CLR input to the chip is LOW, the flip-flop outputs will all remain low. It means that, in effect, the register will contain all zeros.

Shifting of data into the register in a serial fashion is exactly the same as the IC 74L91. Data at the serial input may be changed while the clock is either low or high, but the usual hold and setup times must be observed. The data sheet for this device gives hold time as 0.0 ns and setup time as 30 ns.

Now we try to analyze the gated serial inputs A and B. Suppose that the serial data is connected to B; then A can be used as a control line. Here's how it works:



(a) Logic diagram.



(b) Pinout diagram of IC 74164.

A is held high: The NAND gate is enabled and the serial input data passes through the NAND gate inverted. The input data is shifted serially into the register.
A is held low: The NAND gate output is forced high, the input data steam is inhibited, and the next clock pulse will shift a 0 into the first flip-flop. Each succeeding positive clock pulse will shift another 0 into the register. After eight clock pulses, the register will be full of zeros.

PARALLEL-IN–SERIAL-OUT REGISTER

In the preceding two cases the data was shifted into the registers in a serial manner. Here we develop an idea for the parallel entry of data into the register. Here the data bits are entered into the flip-flops simultaneously, rather than a bit-by-bit basis.

A 4-bit parallel-in–serial-out register is illustrated in Figure below. A, B, C, and D are the four parallel data input lines and $SHIFT / \overline{LOAD}$ $(SH / \overline{LD})$ is a control input that allows the four bits of data at A, B, C, and D inputs to enter into the register in parallel or shift the data in serial. When $SHIFT / \overline{LOAD}$ is HIGH, AND gates G_1, G_3 & G_5 are enabled, allowing the data bits to shift right from one stage to the next. When $SHIFT / \overline{LOAD}$ is LOW, AND gates G_2, G_4, and G_6 are enabled, allowing the data bits at the parallel inputs. When a clock pulse is applied, the flip-flops with D = 1 will be set and the flip-flops with D = 0 will be reset, thereby storing all the four bits simultaneously. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which of the AND gates are enabled by the level on the $SHIFT / \overline{LOAD}$ input.

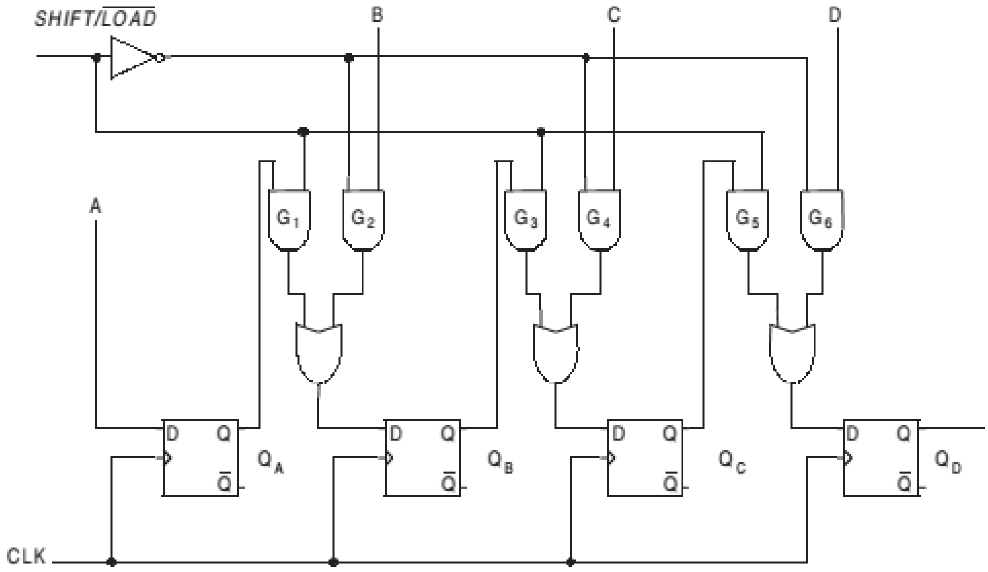
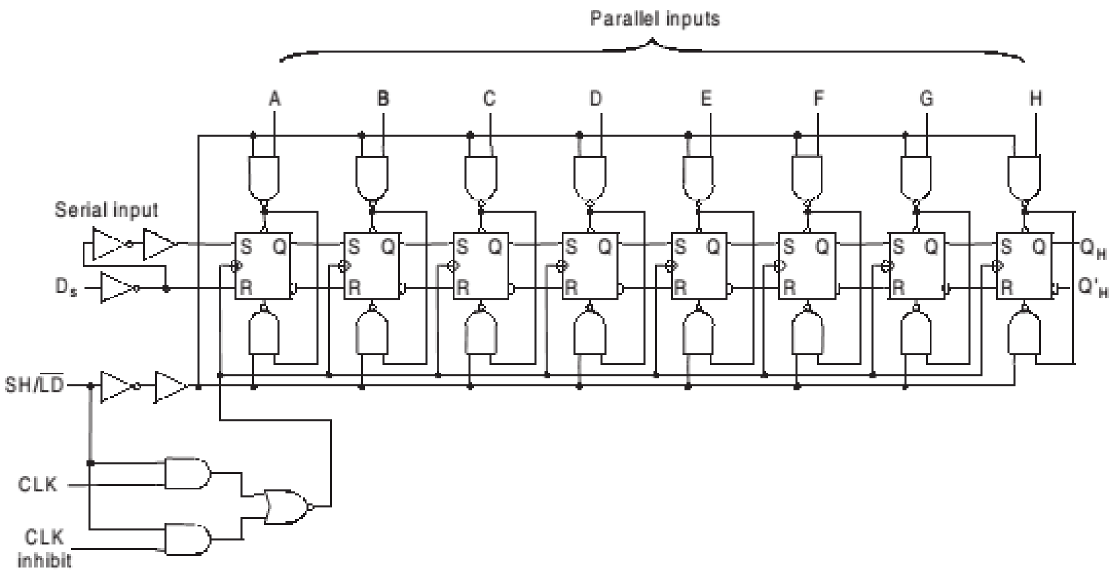


Figure:- A 4-bit parallel-in–serial-out shift register.

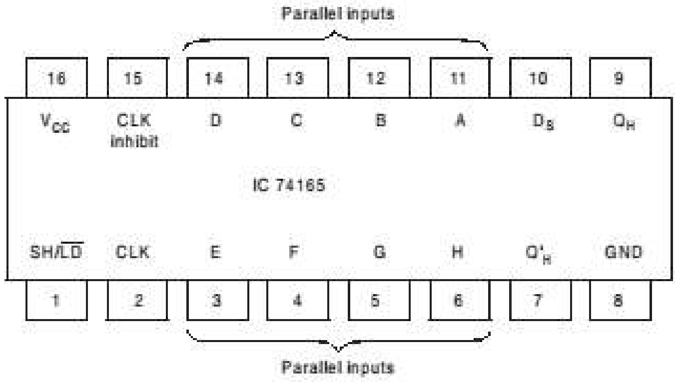
8-bit Parallel-in–Serial-out Shift Register

The pinout and logic diagram of IC 74165 is shown in Figure below. IC 74165 is an example of an 8-bit serial/parallel-in and serial-out shift register. The data can be loaded into the register in parallel and shifted out serially at QH using either of two clocks (CLK or CLK inhibit). It also contains a serial input, DS through which the data can be serially shifted in.

When the input *SHIFT* / $\overline{LOAD}$ $(SH / \overline{LD})$ is LOW, it enables all the NAND gates for parallel loading. When an input data bit is a 0, the fl ip-fl op is asynchronously RESET by a LOW output of the lower NAND gate. Similarly, when the input data bit is a 1, the flip-flop is asynchronously SET by a LOW output of the upper NAND gate. The clock is inhibited during parallel loading operation. A HIGH on the *SHIFT* / $\overline{LOAD}$ input enables the clock causing the data in the register to shift right. With the low to high transitions of either clock, the serial input data (DS) are shifted into the 8-bit register.



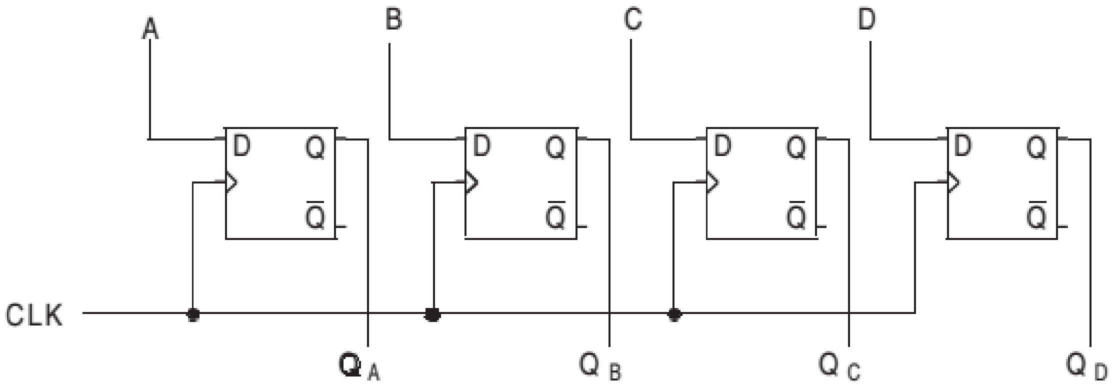
(a) Logic diagram.



(b) Pinout diagram of IC 74165.

PARALLEL-IN–PARALLEL-OUT REGISTER

The parallel input of data has already been discussed in the preceding section of parallel-in–serial-out shift register. Also, in this type of register there is no interconnection between the flip-flops since no serial shifting is required. Hence, the moment the parallel entry of the data is accomplished the data will be available at the parallel outputs of the register. A simple parallel-in–parallel out shift register is shown in Figure below.



Here the parallel inputs to be applied at A, B, C, and D inputs are directly connected to the D inputs of the respective fl ip-fl ops. On applying the clock transitions, these inputs are entered into the register and are immediately available at the outputs Q_1, Q_2, Q_3, and Q_4.

UNIVERSAL REGISTER

A register that is capable of transferring data in only one direction is called a '***unidirectional shift register***' whereas the register that is capable of transferring data in both left and right direction is called a '***bidirectional shift register***'. Now if the register has both the shift-right and shift-left capabilities, along with the necessary input and output terminals for parallel transfer, then it is called a ***shift register* with *parallel load*** or '***universal shift register***'.

The most general shift register has all the capabilities listed below. Others may have only some of these functions, with at least one shift operation.

1) A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift-right.
2) A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.

3) A parallel-load control to enable a parallel transfer and the *n* input lines associated with the parallel transfer.
4) *n* parallel output lines.
5) A *clear* control to clear the register to 0.
6) A *CLK* input for clock pulses to synchronize all operations.
7) A control state that leaves the information in the register unchanged even though clock pulses are continuously applied.

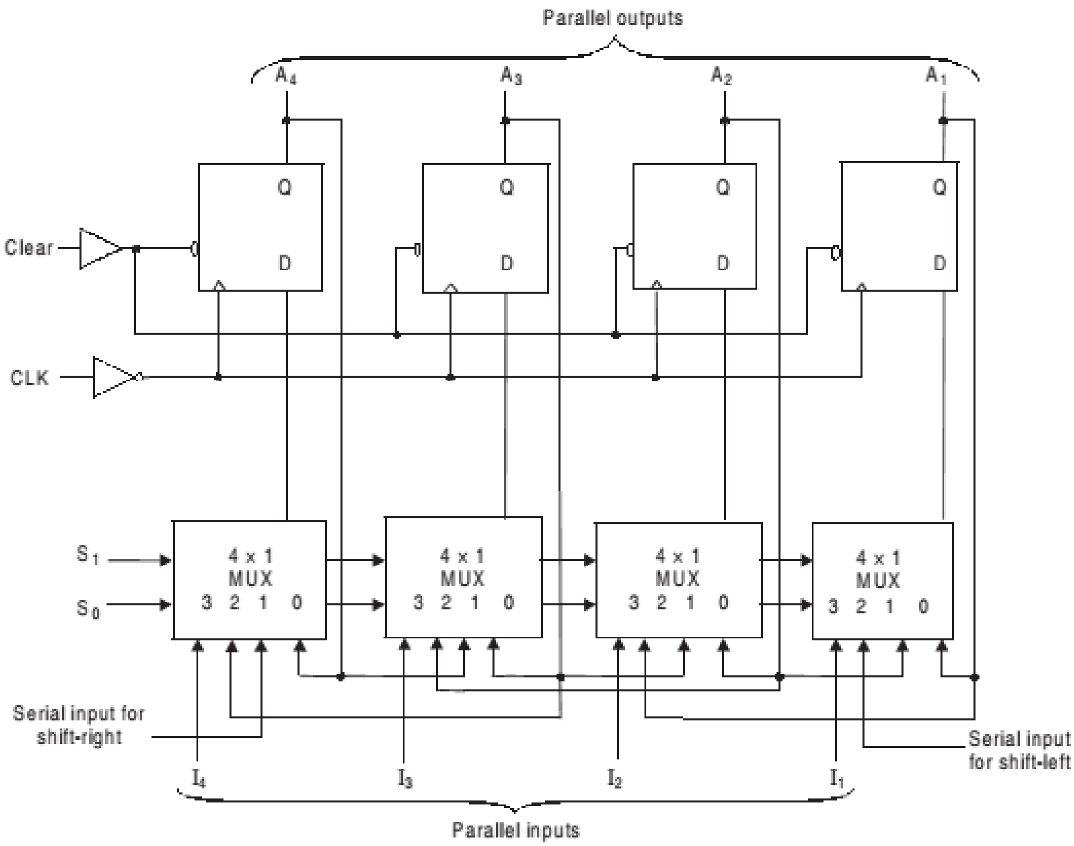


Figure:- 4-bit universal shift register.

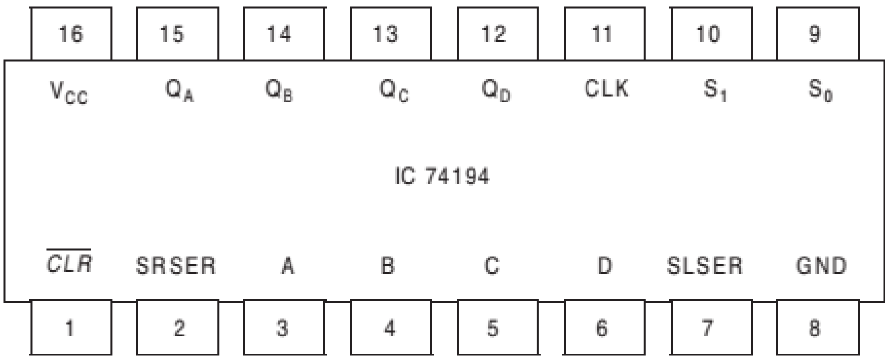
The diagram of a shift-register with all the capabilities listed above is shown in Figure above. This is similar to IC type 74194. Though it consists of four D flip-flops, S-R flip-flops can also be used with an inverter inserted between the S and R terminals. The four multiplexers drawn are also part of the register. The four multiplexers have two common selection lines S_1 and S_0. When $S_1S_0 = 00$, the input 0 is selected for each of the multiplexers. Similarly, when $S_1S_0 = 01$, the input 1, when $S_1S_0 = 10$, the input 2 and for $S_1S_0 = 11$, the input 3, is selected for each of the multiplexers.

The S1 and S0 inputs control the mode of operation of the register as specified in the entries of functions in the below Table. When $S_1S_0 = 00$, the present value of the register is applied to the D inputs of the flip-flops. Hence this condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock pulse transition transfers into each flip-flop the binary value held previously & no change of state occurs. When $S_1S_0 = 01$, terminals 1 of each of the multiplexer inputs have a path to the D inputs of each of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop A_4. Similarly, with S1S0 = 10, a shift-left operation results, with the other serial input going into flip-flop A_1. Finally, when S1S0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock pulse.

Mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift-right
1	0	Shift-left
1	1	Parallel load

Table:- Function table for the universal register

A universal register is a general-purpose register capable of performing three operations: shift-right, shift-left, and parallel load. Not all shift registers available in MSI circuits have all these capabilities. The particular application dictates the choice of one MSI circuit over another. As we have already mentioned IC 74194 is a 4-bit bidirectional shift register with parallel load. The pinout diagram of IC 74194 is shown in Figure below:-



The parallel loading of data is accomplished with a positive transition of the clock and by applying the four bits of data to the parallel inputs and a HIGH to the S_1 and S_0 inputs. Similarly, shift-right is accomplished synchronously with the positive edge of the clock when S_0 is HIGH and S_1 is LOW. In this mode the serial data is entered at the shift right serial input. In the same manner, when S_0 is LOW and S_1 is HIGH, data bits shift left synchronously with the clock pulse and new data is entered at the shift-left serial input.

SHIFT REGISTER COUNTERS

Shift registers may be arranged to form different types of counters. These shift registers use *feedback*, where the output of the last flip-flop in the shift register is fed back to the first flip-flop. Based on the type of this feedback connection, the shift register counters are classified as (*i*) ring counter and (*ii*) twisted ring or Johnson or Shift counter.

Ring Counter

It is possible to devise a counter-like circuit in which each flip-flop reaches the state Q = 1 for exactly one count, while for all other counts Q = 0. Then Q indicates directly an occurrence of the corresponding count. Actually, since this does not represent binary numbers, it is better to say that the outputs of the flip-flops represent a code. Such a circuit is shown in Figure below, which is known as a ***ring counter***. The Q output of the last stage in the shift register is fed back as the input to the first stage, which creates a ring-like structure. Hence a ring *counter* is a circular shift register with only one flip-flop being set at any particular time and all others being cleared. The single bit is shifted from one flip-flop to the other to produce the sequence of timing signals. Such encoding where there is a single 1 and the rest of the code variables are 0, is called a *one-hot code*.

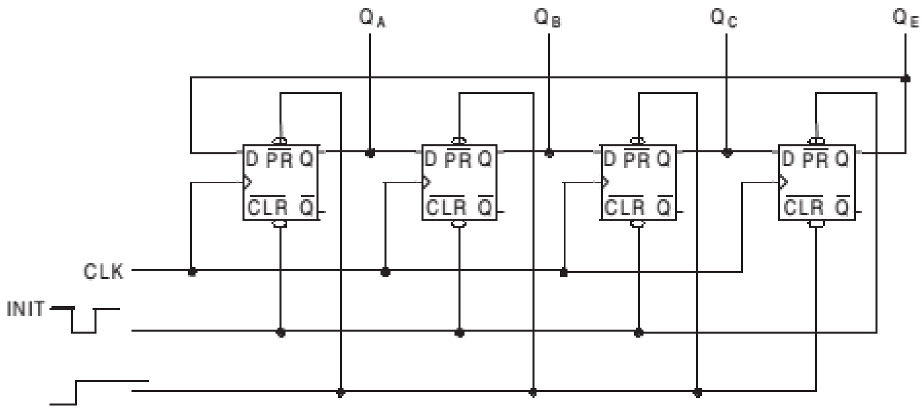


Figure:- A 4-bit ring counter using D flip-flops

The circuit shown in Figure above consists of four flip-flops and their outputs are Q_A,Q_B, Q_C, and Q_E respectively. The PRESET input of the last flip-fl op and the CLEAR inputs of the other three flip-flops are connected together. Now, by applying a LOW pulse at this line, the last flip-flop is SET and all the others are RESET, *i.e.*, $Q_AQ_BQ_CQ_E$ = 0001. Hence, from the circuit it is clear that D_A = 1, D_B = 0, D_C = 0, and D_E = 0. Therefore, when a clock pulse is applied, the 1st flip-flop is set to 1, while the other three flip-flops are reset to 0 *i.e.*, the output of the ring counter is $Q_AQ_BQ_CQ_E$ = 1000. Similarly, when the 2nd clock pulse is applied, the 1 in the first flip-flop is shifted to the second flip-flop & the output of the counter becomes $Q_AQ_BQ_CQ_E$ = 0100; on occurrence of the 3rd clock pulse, the output will be $Q_AQ_BQ_CQ_E$ = 0010; on occurrence of the fourth clock pulse the output becomes $Q_AQ_BQ_CQ_E$ = 0001, *i.e.*, the initial state. Thus, the 1 is shifted around the register as long as the clock pulses are applied. The truth table that describes the operation of the above 4-bit ring counter is shown in Table below:-

INIT	*CLK*	Q_A	Q_B	Q_C	Q_E
L	X	0	0	0	1
H	↑	1	0	0	0
H	↑	0	1	0	0
H	↑	0	0	1	0
H	↑	0	0	0	1

Johnson Counter

A *k*-bit ring counter circulates a single bit among the flip-flops to provide *k* distinguishable states. The number of sates can be doubled if the shift register is connected as a *switch-tail* ring counter. A switch-tail ring counter is a circular shift register with the complement of the last flip-flop being connected to the input of the first flip-flop. Figure below shows such a type of shift register. The circular connection is made from the complement of the rightmost flip-flop to the input of the leftmost flip-flop. The register shifts its contents once to the right with every clock pulse, and at the same time, the complement value of the E flip-flop is transferred into the A flip-flop. Starting from a cleared state, the switch-tail ring counter goes through a sequence of eight states as listed in Table below. In general a *k*-bit switch-tail counter will go through 2k states. Starting with all 0s each shift operation inserts 1s from the left until the register is filled with all 1s. In the following sequences, 0s are inserted from the left until the register is again filled with all 0s.

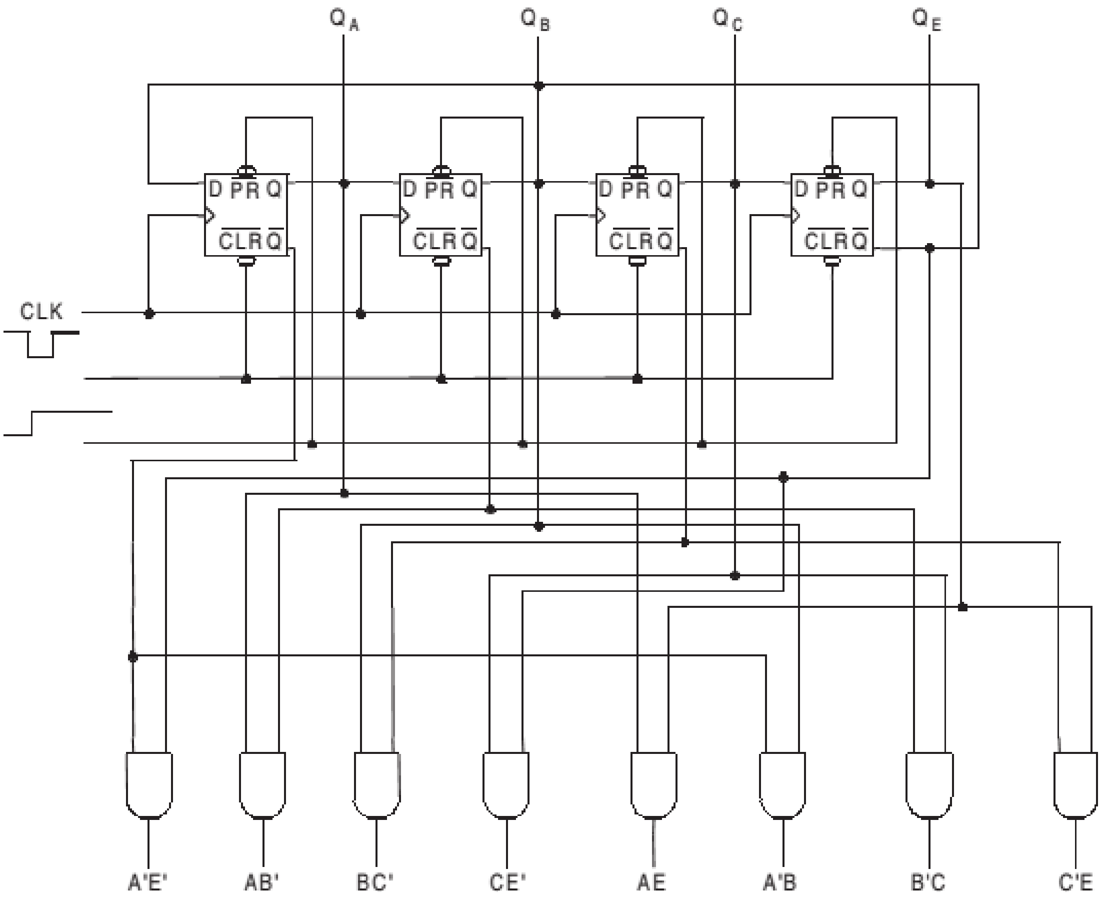


Figure:- A 4-bit Johnson counter using D flip-flops and decoding gates.

A *Johnson* or *moebius counter* is a switch-tail ring counter with 2k decoding gates to provide outputs for 2*k* timing signals. The decoding gates are also shown in Figure above. Since each gate is enabled during one particular state sequence, the outputs of the gates generate eight timing sequences in succession.

The decoding of a *k*-bit switch-tail ring counter to obtain 2*k* timing sequences follows a regular pattern. The all-0s state is decoded by taking the complement of the two extreme flip-flop outputs. The all-1s state is decoded by taking the normal outputs of the two extreme flip-flops. All other states are decoded from an adjacent 1, 0 or 0, 1 pattern in the sequence.
For example, sequence 6 has an adjacent 0 and 1 pattern in flip-flops A and B. the decoded output is then obtained by taking the complement A and the normal of B, or the A′B.

Sequence number	Flip-flop outputs			
	A	B	C	E
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

Table :- Count sequence of a 4-bit Johnson counter

One disadvantage of the circuit in Figure 8.16 is that, if it finds itself in an unused state, it will persist in moving from one invalid state to another and never find its way to a valid state. The difficulty can be corrected by modifying the circuit to avoid this undesirable condition. One correcting procedure is to disconnect the output from flip-flop B that goes to the D input of flip-flop C, and instead enable the input of flip-flop C by the function:

DC = (A + C)B, where DC is the flip-flop input function for the D input of the flip-flop C.

Johnson counters can be constructed for any number of timing sequences. The number of flip-flops needed is one-half the number of timing signals. The number of decoding gates is equal to the number of timing sequences and only 2-input gates are employed. Ring counter does not require any decoding gates, since in ring counter only one flip-flop will be in the set condition at any time.

Asynchronous and Synchronous Shift Registers

Asynchronous circuits changes state each time the input changes the state, while synchronous circuit changes state only when triggered by a momentary change in the input signal. This momentary change is called triggering.

Shift registers are made of flip flops and their operation depends upon the state at the flip flops. Flip flops changes their states due to triggering when flip flop change their state on the base of input pulse then it is called Edge triggering. In edge triggering flip flop change its state on the basses of Leading edge or trailing edge. When flip flop works on the bases of change in DC level, that is called Asynchronous Triggering. And the shift registers work on this principle is called Asynchronous shift registers. On the other hand, shift registers changes their state only when triggered by clock pulse are called Synchronous shift registers these type of shift registers usually used in counters.

Counters

Counting is frequently required in digital computers and other digital systems to record the number of events occurring in a specified interval of time. Normally an electronic counter is used for counting the number of pulses coming at the input line in a specified time period. The counter must possess memory since it has to remember its past states. As with other sequential logic circuits counters can be synchronous or asynchronous.

As the name suggests, it is a circuit which counts. The main purpose of the counter is to record the number of occurrence of some input. There are many types of counter both binary and decimal. Commonly used counters are

1. Binary Ripple Counter
2. Ring Counter
3. BCD Counter
4. Decade counter
5. Up down Counter
6. Frequency Counter

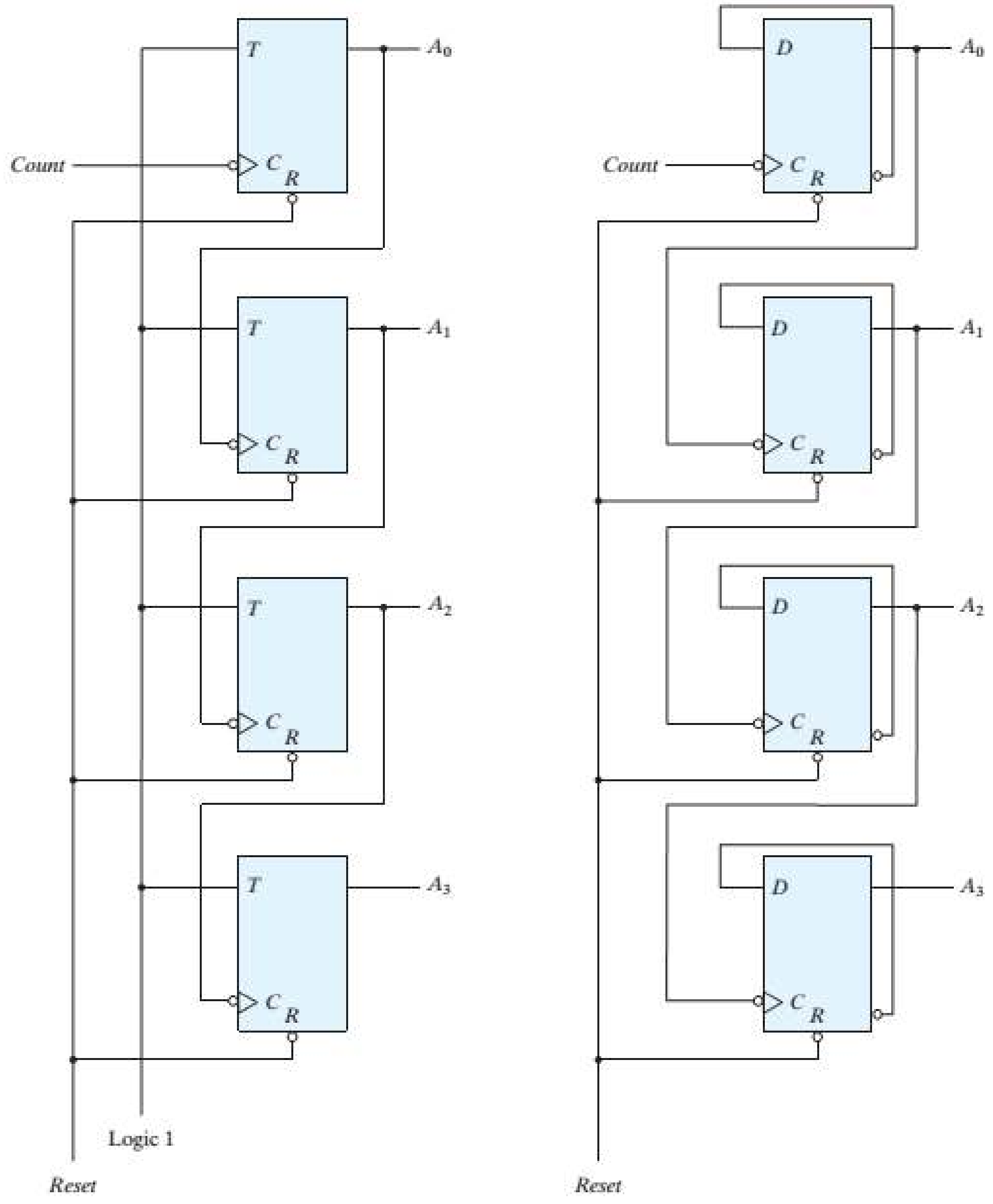
Ripple Counter

A counter that follows the binary number sequence is called a binary counter. An n -bit binary counter consists of n flip-flops and can count in binary from 0 through 2n - 1. Counters are available in two categories: ripple counters and synchronous counters. In a ripple counter, a flip-flop output transition serves as a source for triggering other flip-flops. In other words, the C input of some or all flip-flops are triggered, not by the common clock pulses, but rather by the transition that occurs in other flip-flop outputs. In a synchronous counter, the C inputs of all flip-flops receive the common clock.

Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the *C* input of the next higher order flip-flop. The flip-flop holding the least significant bit receives the incoming count pulses. A complementing flip-flop can be obtained from a *JK* flip-flop with the *J* and *K* inputs tied together or from a *T* flip-flop. A third possibility is to use a *D* flip-flop with the complement output connected to the *D* input. In this way, the *D* input is always the complement of the present state, and the next clock pulse will cause the flip-flop to complement. The logic diagram of two 4-bit binary ripple counters is shown in Fig. below . The output of each flip-flop is connected to the *C* input of the next flip-flop in sequence. The flip-flop holding the least significant bit receives the incoming count pulses. The *T* inputs of all the flip-flops in (a) are connected to a permanent logic 1, making each flip-flop complement if the signal in its *C* input goes through a negative transition. The bubble in front of the dynamic indicator symbol next to *C* indicates that the flip-flops respond to the negative-edge transition of the input. The negative transition occurs when the output of the previous flip-flop to which *C* is connected goes from 1 to 0.

The count starts with binary 0 and increments by 1 with each count pulse input. After the count of 15, the counter goes back to 0 to repeat the count. The least significant bit,A_0, is complemented with each count pulse input. Every time that A_0 goes from 1 to 0, it complementsA_1. Every time thatA_1 goes from 1 to 0, it complements A_2. Every time that A_2 goes from 1 to 0, it complements A_3, and so on for any other higher order bits of a ripple counter. For example, consider the transition from count 0011 to 0100. A_0 is complemented with the count pulse. Since A_0 goes from 1 to 0, it triggers A_1 and complements it. As a result, A_1 goes from 1 to 0, which in turn complements A_2, changing it from 0 to 1. A_2 does not trigger A_3, because A_2 produces a positive transition and the flip-flop responds only to negative transitions. Thus, the count from 0011 to 0100 is achieved by changing the bits one at a time, so the count goes from 0011 to 0010, then to 0000, and finally to 0100. The flip-flops change one at a time in succession, and the signal propagates through the counter in a ripple fashion from one stage to the next.



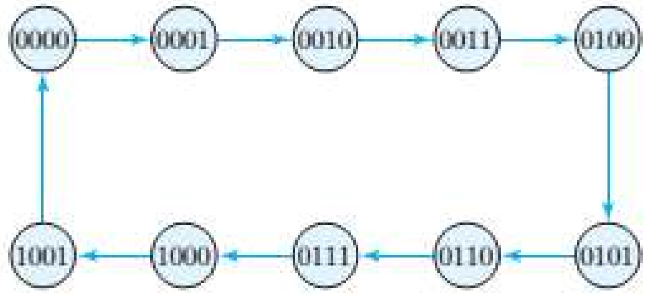
(a) With *T* flip-flops (b) With *D* flip-flops

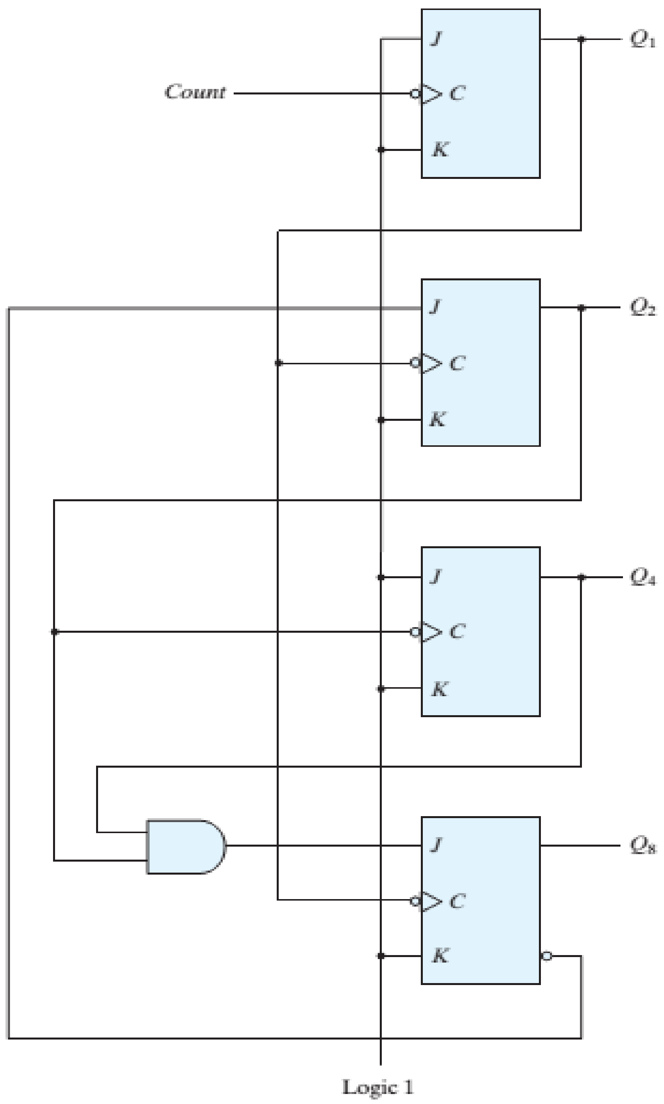
A binary counter with a reverse count is called a *binary countdown counter*. In a countdown counter, the binary count is decremented by 1 with every input count pulse. The count of a four-bit countdown counter starts from binary 15 and continues to binary counts 14, 13, 12, . . . , 0 and then back to 15. A list of the count sequence of a binary countdown counter shows that the least significant bit is complemented with every count pulse. Any other bit in the sequence is complemented if its previous least significant bit goes from 0 to 1. Therefore, the diagram of a binary countdown counter looks the same as the binary ripple counter in Fig. above , provided that all flip-flops trigger on the positive edge of the clock. (The bubble in the *C* inputs must be absent.) If negative-edge-triggered flip-flops are used, then the *C* input of each flip-flop must be connected to the complemented output of the previous flip-flop. Then, when the true output goes from 0 to 1, the complement will go from 1 to 0 and complement the next flip-flop as required.

BCD Ripple Counter

A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9.Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits. The sequence of states in a decimal counter is dictated by the binary code used to represent a decimal digit. If BCD is used, the sequence of states is as shown in the state diagram in the side Fig.:-

(Fig:- State diagram for a decimal BCD counter)





A decimal counter is similar to a binary counter, except that the state after 1001 (the code for decimal digit 9) is 0000 (the code for decimal digit 0).

The logic diagram of a BCD ripple counter using *JK* flip-flops is shown in Figure below. The four outputs are designated by the letter symbol *Q*, with a numeric subscript equal to the binary weight of the corresponding bit in the BCD code. Note that the output of Q_1 is applied to the *C* inputs of both Q_2 and Q_8 and the output of Q_2 is applied to the *C* input of Q_4. The *J* and *K* inputs are connected either to a permanent 1 signal or to outputs of other flip-flops.

A ripple counter is an asynchronous sequential circuit. Signals that affect the flip-flop transition depend on the way they change from 1 to 0. The operation of the counter can be explained by a list of conditions for flip-flop transitions. These conditions are derived from the logic diagram and from knowledge of how a *JK* flip-flop operates. Remember that when the *C* input goes from 1 to 0, the flip-flop is set if $J = 1$, is cleared if $K = 1$, is complemented if $J = K = 1$, and is left unchanged if $J = K = 0$.

To verify that these conditions result in the sequence required by a BCD ripple counter, it is necessary to verify that the flip-flop transitions indeed follow a sequence of states as specified by the state diagram as mentioned above .Q_1 changes state after each clock pulse.Q_2 complements every time Q_1 goes from 1 to 0, as long as $Q_8 = 0$. When Q_8 becomes 1, Q_2 remains at 0. Q_4 complements every time Q_2 goes from 1 to 0. Q_8 remains at 0 as long as Q_2 or Q_4 is 0. When both Q_2 and Q_4 become 1, Q_8 complements when Q_1 goes from 1 to 0. Q_8 is cleared on the next transition of Q_1.

The BCD counter of Fig. above is a *decade* counter, since it counts from 0 to 9. To count in decimal from 0 to 99, we need a two-decade counter. To count from 0 to 999, we need a three-decade counter. Multiple decade counters can be constructed by connecting BCD counters in cascade, one for each decade. A three-decade counter is shown in Fig. below:-

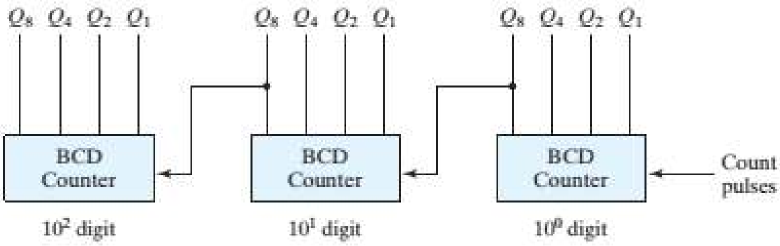


Fig: - Block diagram of a three-decade decimal BCD counter

The inputs to the second and third decades come from Q_8 of the previous decade. When Q_8 in one decade goes from 1 to 0, it triggers the count for the next higher order decade while its own decade goes from 9 to 0.

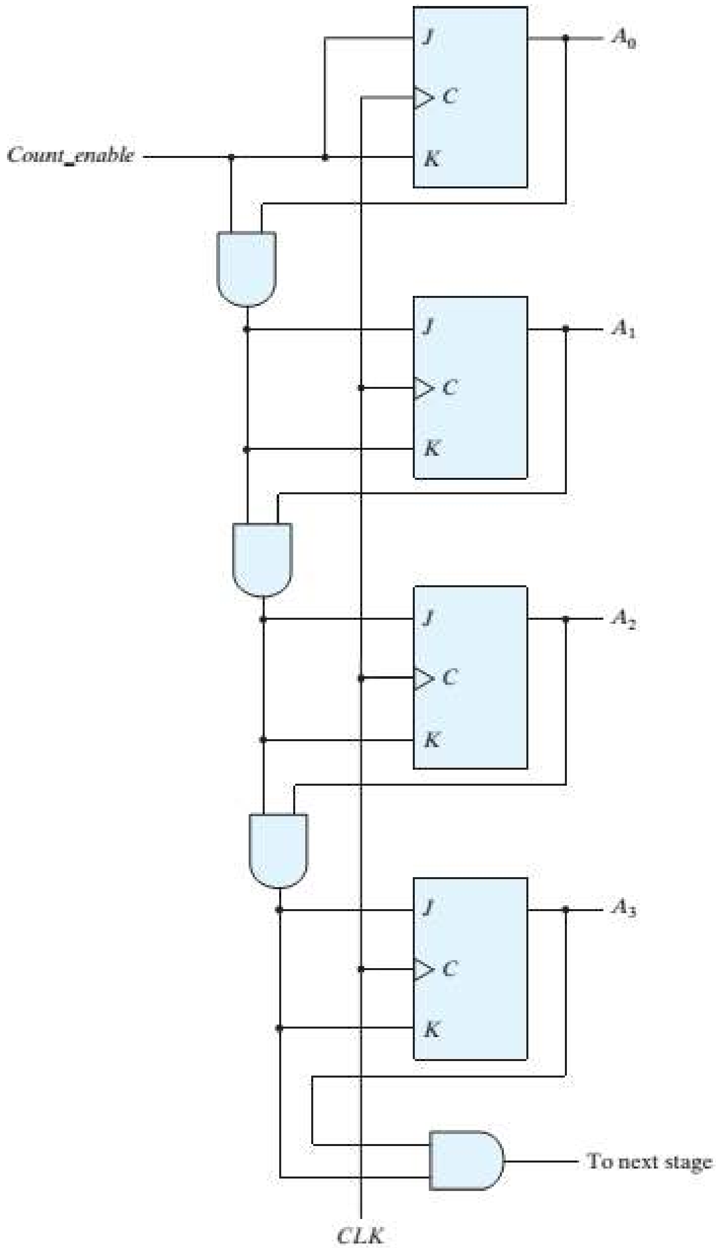
Synchronous counters

Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops. A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter. The decision whether a flip-flop is to be complemented is determined from the values of the data inputs, such as *T* or *J* and *K* at the time of the clock edge. If $T = 0$ or $J = K = 0$, the flip-flop does not change state. If $T = 1$ or $J = K = 1$, the flip-flop complements. Here we present some typical synchronous counters and explain their operation.

Binary Counter

The design of a synchronous binary counter is so simple that there is no need to go through a sequential logic design process. In a synchronous binary counter, the flip-flop in the least significant position is complemented with every pulse. *A flip-flop in any other position is complemented when all the bits in the lower significant positions are equal to 1.* For example, if the present state of a four-bit counter is$A_3A_2A_1A_0$ = 0011, the next count is 0100.$A0$ is always complemented.$A1$ is complemented because the present state of $A0$ = 1.A_2 is complemented because the present state of A_1A_0 = 11. However, A_3 is not complemented, because the present state of $A_2A_1A_0$ = 011, which does not give an all-1's condition.

Synchronous binary counters have a regular pattern and can be constructed with complementing flip□flops and gates. The regular pattern can be seen from the 4□bit counter depicted in Fig. below.



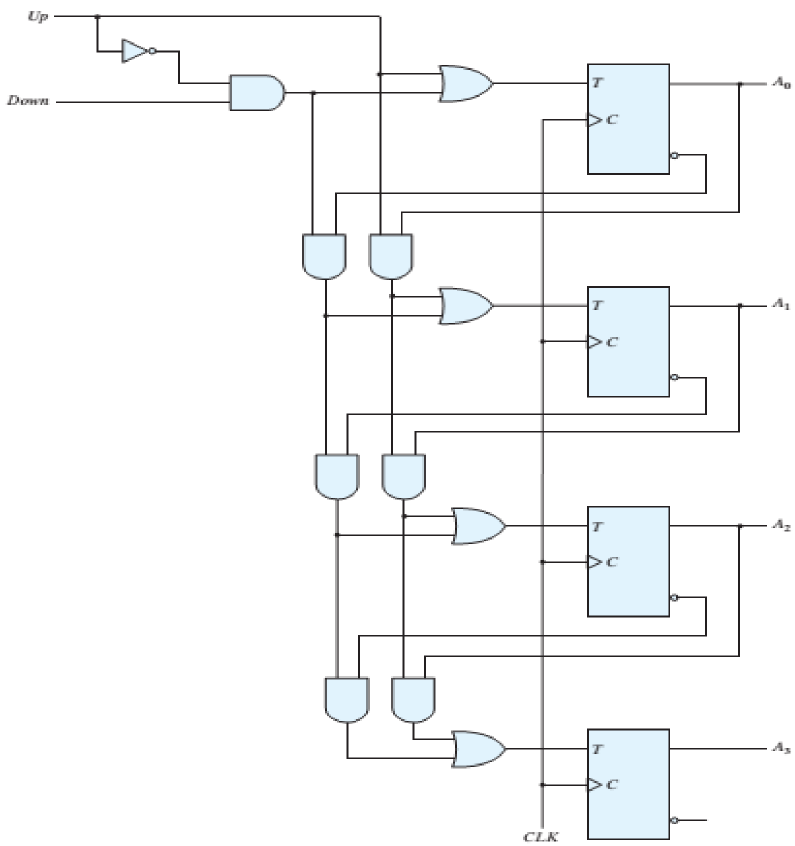
(**Figure: -** Four□bit synchronous binary counter)

The *C* inputs of all flip□flops are connected to a common clock. The counter is enabled by *Count enable*. If the enable input is 0, all *J* and *K* inputs are equal to 0 and the clock does not change the state of the counter. The first stage, *A*0, has its *J* and *K* equal to 1 if the counter is enabled. The other *J* and *K* inputs are equal to 1 if all previous least significant stages are equal to 1 and the count is enabled. The chain of AND gates generates the required logic for the *J* and *K* inputs in each stage. The counter can be extended to any number of stages, with each stage having an additional flip□flop and an AND gate that gives an output of 1 if all previous flip□flop outputs are 1.

Note that the flip□flops trigger on the positive edge of the clock. The polarity of the clock is not essential here, but it is with the ripple counter. The synchronous counter can be triggered with either the positive or the negative clock edge. The complementing flip□flops in a binary counter can be of either the *JK* type, the *T* type, or the *D* type with XOR gates.

Up–Down Binary Counter

A synchronous countdown binary counter goes through the binary states in reverse order, from 1111 down to 0000 and back to 1111 to repeat the count. It is possible to design a countdown counter in the usual manner, but the result is predictable by inspection of the downward binary count. The bit in the least significant position is complemented with each pulse. *A bit in any other position is complemented if all lower significant bits are equal to 0.* For example, the next state after the present state of 0100 is 0011. The least significant bit is always complemented. The second significant bit is complemented because the first bit is 0. The third significant bit is complemented because the first two bits are equal to 0. But the fourth bit does not change, because not all lower significant bits are equal to 0.A countdown binary counter can be constructed as shown in previous Fig., except that the inputs to the AND gates must come from the complemented outputs, instead of the normal outputs, of the previous flip□flops. The two operations can be combined in one circuit to form a counter capable of counting either up or down. The circuit of a 4bit up–down binary counter using *T* flip□flops is shown in Fig. below.



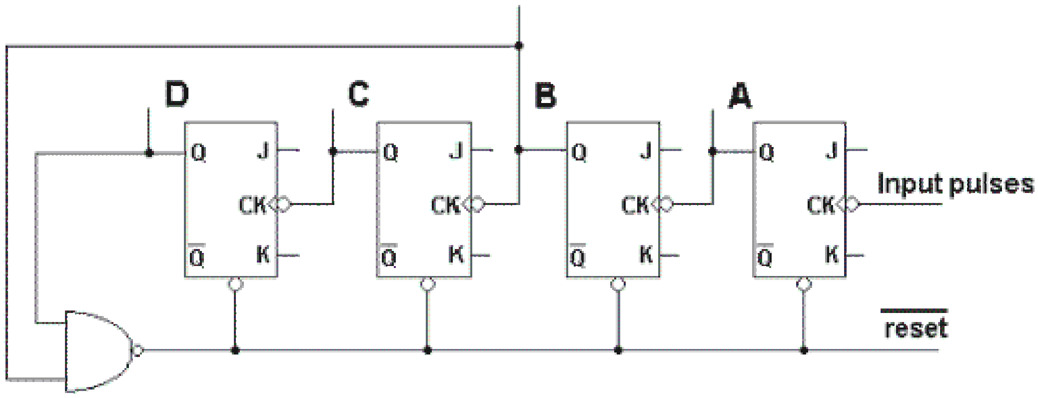
It has an up control input and a down control input. When the up input is 1, the circuit counts up, since the *T* inputs receive their signals from the values of the previous normal outputs of the flip‐flops. When the down input is 1 and the up input is 0, the circuit counts down, since the complemented outputs of the previous flip‐flops are applied to the *T* inputs. When the up and down inputs are both 0, the circuit does not change state and remains

Decade Counter

A decade counter is the one which goes through 10 unique combinations of outputs and then resets as the clock proceeds. We may use some sort of a feedback in a 4-bit binary counter to skip any six of the sixteen possible output states from 0000 to 1111 to get to a decade counter. A decade counter does not necessarily count from 0000 to 1001 it could count as 0000,0001, 0010, 1000, 1001, 1010, 1011, 1110, 1111, 0000, 0001 and so on.

Figure below shows a decade counter having a binary count that is always equivalent to the input pulse count. The circuit is essentially, a ripple counter which count up to 16. We desire however, a circuit operation in which the count advance from 0 to 9 and then reset to 0 for a new cycle. This reset is a accomplished at the desired count as follows.

1. With counter REST count = 0000 the counter is ready to stage counter cycle.
2. Input pulses advance counter in binary sequence up to count of a (count = 1001)
3. The next count pulse advance the count to 10 count = 1010. A logic NAND gate decodes the count of 10 providing a level change at that time to trigger the one shot unit which then resets all counter stages. Thus, the pulse after the counter is at count = 9, effectively results in the counter going to count = 0.



(**Figure:** Decade Counter)

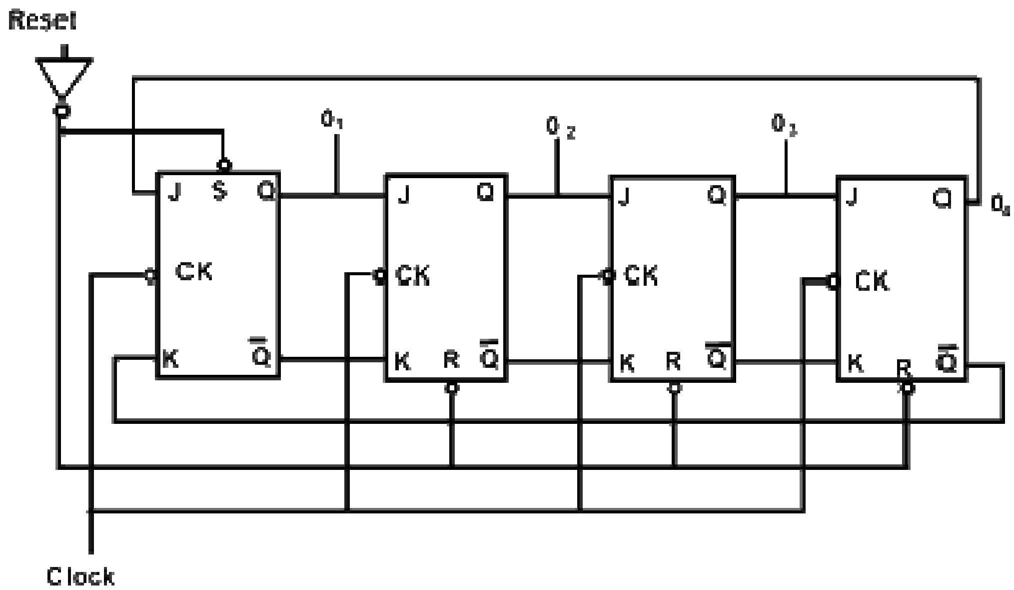
Table below provides a count table showing the binary count equivalent to the decimal count of input pulses. The table also shows that the count goes momentarily count from nine (1001) to ten (1010) before resetting to zero(0000). The NAND gate provides an output of 1 until the count reach ten. The count of ten is decoded (or sensed in this case) by using logic inputs that are all 1 at the count of ten. When the count becomes ten the NAND gate output goes to logical 0, providing a 1 to 0 logic change to trigger the one shot unit, which then provides a short pulse to reset all counter stages.

The Q signal is used since it is normally high and goes low during the one shot timing period the flip flop in this circuit being reset by a low signal level (active low clearing). The one shot pulse need only be long enough so that slowest counter stage resets. Actually, at this time only the $2^1$and 2^3 stage need be reset, but all stages are reset to insure that a new cycle at the count 0000.

Table : Decade Counter Truth Table				
Input Pulses	**D**	**C**	**B**	**A**
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
10	1	0	1	0
0	0	0	0	0

Ring Counter

The ring counter is the simplest example of a shift register. The simplest counter is called a Ring counter. The ring counter contains only one logical 1 or 0 which it circulates. The total cycle length is equal to the number of stages. The ring counter is useful in applications where count has to be recognized in order to perform some other logical operation. Since only one output is ever at logic 1 at given time extra logic gates are not required to decode the counts and the flip flop outputs may be used directly to perform the required operation.



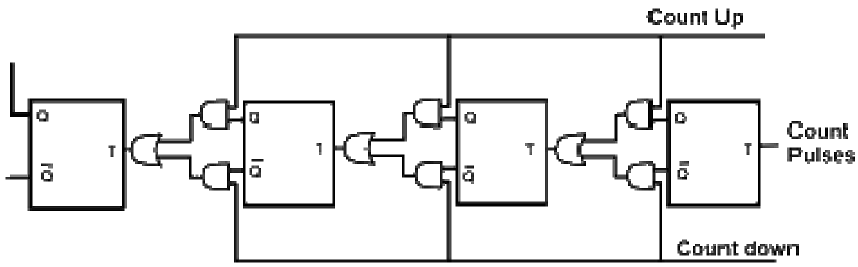
(**Figure:** Simple Ring Counter)

Note that in the above diagram the Reset will reset Q_2, Q_3 and Q_4 but will put Q_1 to a logic 1 state. This 1 will circulate when clock pulses are applied.

Table: Ring Counter Truth Table				
Clock	**0_1**	**0_2**	**0_3**	**0_4**
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0

Up-Down Counter

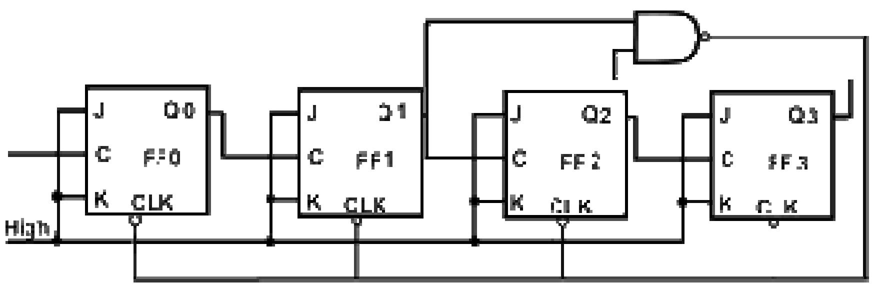
An up down counter is a bi-directional counter and it can be made to count upwards as well as downwards. In other words an up down counter is one which can provide both count up and down counts operations in a single unit. In the previous section it was seen that if triggering pulses are obtained from $\bar{Q}$ output the counter is a count up and if the triggering pulses are obtained from $\bar{Q}$ outputs, the counter is a countdown. Figure below gives an up down counter. When the count up signal is high the AND gate connecting Q output and count up signal gives and output 1 which passes through the OR gate to trigger the next flip flop. This results in the count up operation. Similarly a signal from countdown line will result the circuit to act as a down counter.



(Figure: Up Down Counter)

BCD Counter

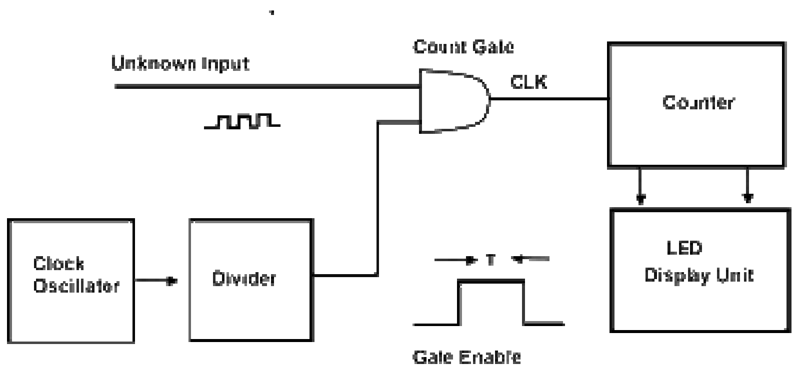
It is a special case of a decade counter in which the counter counts 0000 to 1001 and then resets. The output weights of the flip flops in these counters are in accordance with 8421 code. For instance, at the end of seventh clock pulse, the output sequence will be 0111 (Decimal equivalent of 0111 as per 8421 code is 7). These counters will thus be different from other decade counters that provide the same count by using some kind of forced feedback to skip some of the natural binary counts Figure below shows a counter of the BCD type.



(Figure: BCD Counter)

Frequency Counter

Frequency counter is a digital device which can be used to measure the frequency of the periodic waveforms. The block diagram of frequency counter is shown in Figure below.



(Figure: Frequency Counter)

A signal having time period t applied at one of the input terminal of AND gate. While a unknown signal is also applied at the other input terminal of the AND gate. So, it is used as a clock for counter indicates the frequency of the unknown signal in respect to this time period. The time interval of the counter may be called contents. Let us suppose that time period of gate signal is one second and unknown signal is a square wave of 250 Hertz. In this condition counter counts 250 at the end of one second. This will be frequency of unknown signal.

Shift Registers

A register is a device which is used to store information. Flip flops are often used to make a register. Each flip flop can store 1-bit of information and therefore for storing a n-bit word n-flip-flops are required in the register for example a computer employing 16-bit word length requires 16 flip-flops to hold the number before it is manipulated. The input to a register or output from it may be either in serial or parallel form depending upon the requirement.

Shift Register

A shift register is a storage device that used to store binary data. When a number of flip flop are connected in series it is called a register. A single flip flop is supposed to stay in one of the two stable states 1 or 0 or in other words the flip flop contains a number 1 or 0 depending upon the state in which it is. A register will thus contain a series of bits which can be termed as a word or a byte.

If in these registers the connection is done in such a way that the output of one of the flip flop forms in input to other, it is known as a shift register. The data in a shift register is moved serially (one bit at a time).

The shift register can be built using RS, JK or D flip-flops various types of shift registers are available some of them are given as under.

1. Shift Left Register
2. Shift Right Register
3. Shift Around Register
4. Bi-directional Shift Register

Shift Left Register

A four stage shift-left register is shown in figure 1. The individual stages are JK flip-flops. Notice that the date input consists of opposite binary signals, the reference data signal going to the J input and the opposite data signal going to the K input. For the D-type stage the single data input line is connected as the D-input.

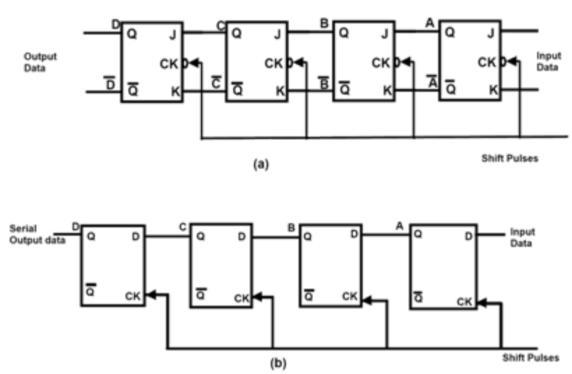


Figure 1: Shift Registers (a) JK (b) D-type

The shift pulse is applied to each stage operating each simultaneously. When the shift pulse occurs the data input is shifted in to that stage. Each stage is set or reset corresponding to the input data at the time the shift pulse occurs. Thus the input data bit is shift in to stage A by the first shift pulse. At the same time the data of stage A is shifted into the stage B and so on for the following stages. At each shift pulse data stored in the register stages shifts left by one stage. New data shifted into stage A, whereas the data present in stage D is shifted out to the left for use by some other shift register or computer unit.

For example consider starting with all stages reset all Q-outputs to logical 0 and applying steady logical 1 input as data input stage A. Table 1 shows the data in each stage after each of four shift pulses. Notice table 2 how the logical 1 input first shifts into stage A and then left to stage D after four shift pulses.

As another example consider shifting alternate 0 and 1 data into stage A starting with all stages 1. Table 2 shows the data in each stage after each of four shift pulses.

Finally as a third example of shift register operation. Consider starting with the count in step 4 of table 2 and applying four more shift pulses with placing a steady logical 0 input as data input to stage A table 3 show this operation.

Table 1				
OPERATION OF SHIFT-LEFT REGISTER				
Shift Pulse	**D**	**C**	**B**	**A**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1

Table 2				
OPERATION OF SHIFT-LEFT REGISTER				
Shift Pulse	**D**	**C**	**B**	**A**
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
3	1	0	1	0
4	0	1	0	1

Table 3				
OPERATION OF SHIFT-LEFT REGISTER				
Shift Pulse	**D**	**C**	**B**	**A**
0	0	1	1	1
1	1	1	1	0
2	0	1	0	0
3	1	0	0	0
4	0	1	0	0

Consider the data in stage A as least significant (2^o) bits (LSB) and those in stage D as most significant bits MSB shift-left register operation provides data starting with MSB bit. A few points should be made clear in register operation.

1. The number of shift pulses be the same as the number of shift in register stages.Consider the data in stage A as least significant (2o) bits (LSB) and those in stage D as most significant bits MSB shift-left register operation provides data starting with MSB bit. A few points should be made clear in register operation.
2. Changes in the shift stages take place simultaneously but only the shift pulse occurs.
3. Data shift into a register stage depends only on what logic levels were present at input terminals J and K at the time the shift pulse occurred. Changes that then take place resulting from data shifted will not affect the next stage until the next shift pulse occurs.

Shift Right Register

Sometimes it is necessary to shift the least significant digit first, as when addition is to be carried out serially. In that case a shift right register is used as in Figure 2 input data is applied to stage D and shifted right. The shift operation is the same as discussed in Shift Left Register except that data transfers to the right. Table 4 shows the action of shifting all logical 1 inputs into an initially reset shift register.

In addition to shifting data register, data into a register data is also of a register. Table 5 shows register operation for an initial value of 1101. Notice that the output from stage A contains the binary number each bit (starting initially with LSB) appearing at the output of each shift step. In the present example it was assumed that logical 0 was shifted as input data so that after four shift pulses have occurred the data has passed through the register and the stages are left reset after the fourth shift pulse.

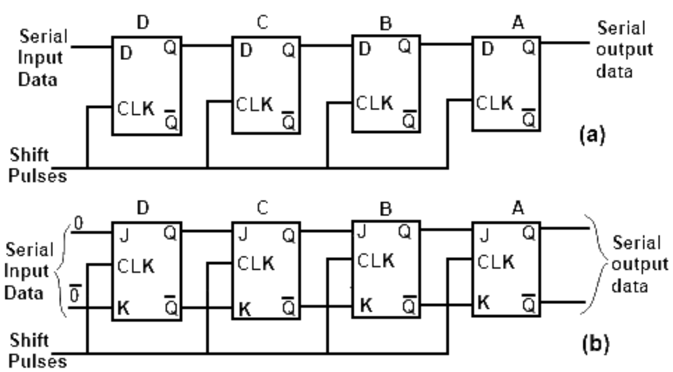


Figure 2: Shift Right Register (a) D-type (b) JK

Table 4				
SHIFT RIGHT OPERATION				
Shift Pulse	D	C	B	A
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0

Table 4				
3	1	1	1	0
4	1	1	1	1

Table 5				
SHIFTED OUT OF SHIFT RIGHT REGISTER				
Shift Pulse	**D**	**C**	**B**	**A**
0	1	1	0	1
1	0	1	1	0
2	0	0	1	1
3	0	0	0	1
4	0	0	0	0

Shift Around Register

When it is required to shift data out of a register with out losing the initial data a shift around register can be used. Figure 3 shows the JK stages in a shift right, shift around register connection. All that was needed was connection of the input of stage A as into the stage D. Then as four shift pulses move the binary data into stage A, the data being shift out of stage A is shift into stage D and returns into the register.

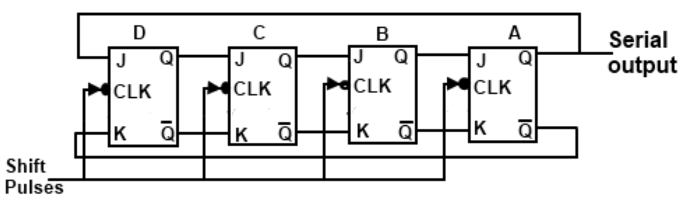


Figure 3: Shift Right Around Register

Table 6

Table 6				
AROUND ACTION WITH SHIFT RIGHT REGISTER				
Shift Pulse	**D**	**C**	**B**	**A**
0	1	1	0	1
1	1	1	1	0
2	0	1	1	1
3	1	0	1	1
4	1	1	0	1

Table 6 shows the result of shifting the binary number 1101 through (and around) the shift register.

Notice that after four shift pulses have occurred the initial value is again in the shift register. To see how any action has taken place other than just shifting the number around there register consider tow shift register stages as in Figure 4 each register, shows in block form, is a four stage shift right register. Externally connecting the A and output of register 1 back to the data input of the same register results in it acting as a shift around register. The logic signal appearing at output A and is also shifted into register 2. Table 7 shows the operation of starting with 1101 in register 1 and 000 in register 2 if the shift around of register 1 were not used and data input were left uncommented (logical 0) then after four shift pulses the data originally in register 1 would be in register 2, with register 1 then reset.

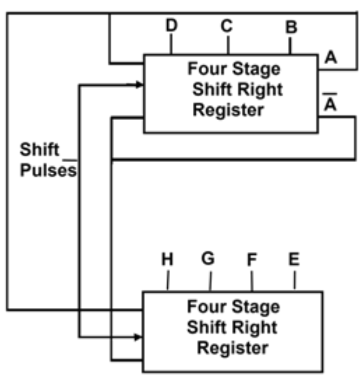


Figure 4: Two Shift Right Registers

Table 7								
OPERATION OF SHIFT REGISTERS OF FIGURE 4								
Shift Pulse	**D**	**C**	**B**	**A**	**H**	**G**	**F**	**E**
0	1	1	0	1	0	0	0	0
1	1	1	1	0	1	0	0	0
2	0	1	1	1	0	1	0	0
3	1	0	1	1	1	0	1	0
4	1	1	0	1	1	1	0	1

Bidirectional Shift Register

A bidirectional shift register is one which can do both the shift left and shift right operations. The arrangement is shown in Figure 5 there are two separate sets of flip-flops. The following steps are controlled by a clock sequentially. The clock and timing arrangements have not been shown in the figure. The lower register is the one in which the data being shifted right or left. The upper register is being used as a temporary storage. The steps are as follows.

1. The contents of the lower register are gated up directly to the upper register, which is assumed to have been cleared previously. This is a parallel transfer of data and is achieved by the first pulse or gate up pulse applied as the gate up terminals.
2. All the lower registers are reset i.e. set =0 by giving a pulse at the reset terminals.
3. The contents of the upper register are gate down to the lower register either one position to the right or to the left as desired. This is again a parallel transfer of data.
4. The upper register is reset for the next shift operation.

Asynchronous And Synchronous Shift Registers

Asynchronous circuits changes state each time the input changes the state, while synchronous circuit changes state only when triggered by a momentary change in the input signal. This momentary change is called triggering.

Shift registers are made of flip flops their operation depends upon the state at the flip flop and their operation depends upon the state at the flip flops. Flip flops changes their states due to triggering when flip flop change their state on the base of input pulse then it is called Edge triggering. In edge triggering flip flop change its state on the basses of Leading edge or trailing edge. When flip flop works on the bases of change in DC level, that is called Asynchronous Triggering. And the shift registers work on this principle are called Asynchronous shift registers. On the other hand, shift registers changes their state only when triggered by clock pulse are called Synchronous shift registers these type of shift registers usually used in counters.

Category: Digital Electronics

‹ Sequential Logic Circuits

Up

Speed of Microprocessor ›