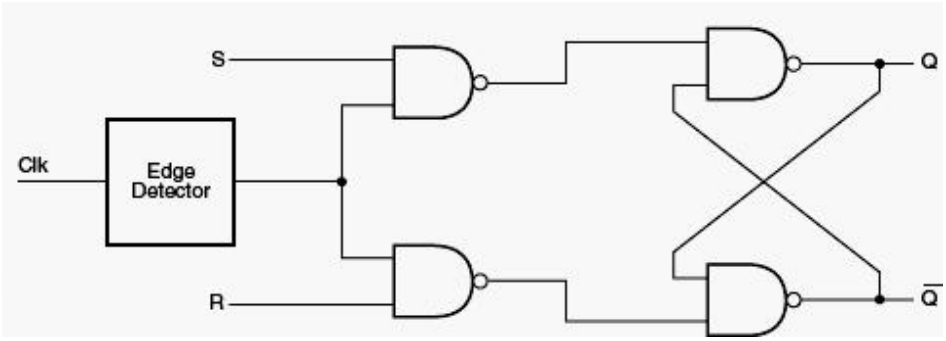
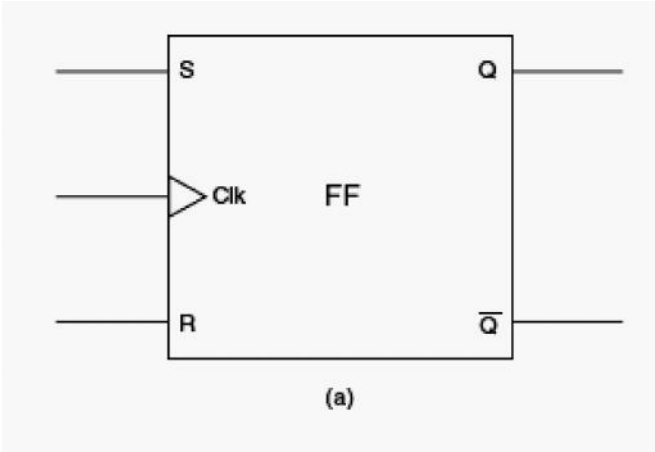
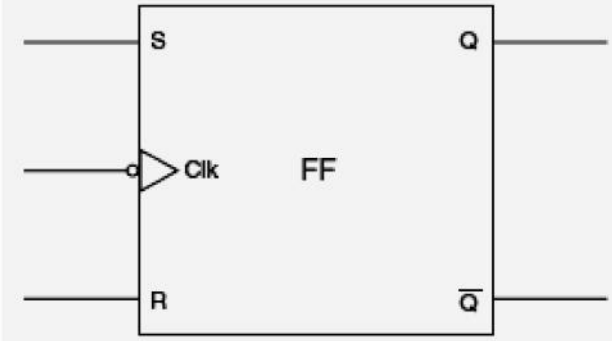
Fundamentals of Digital Computing

Part-4

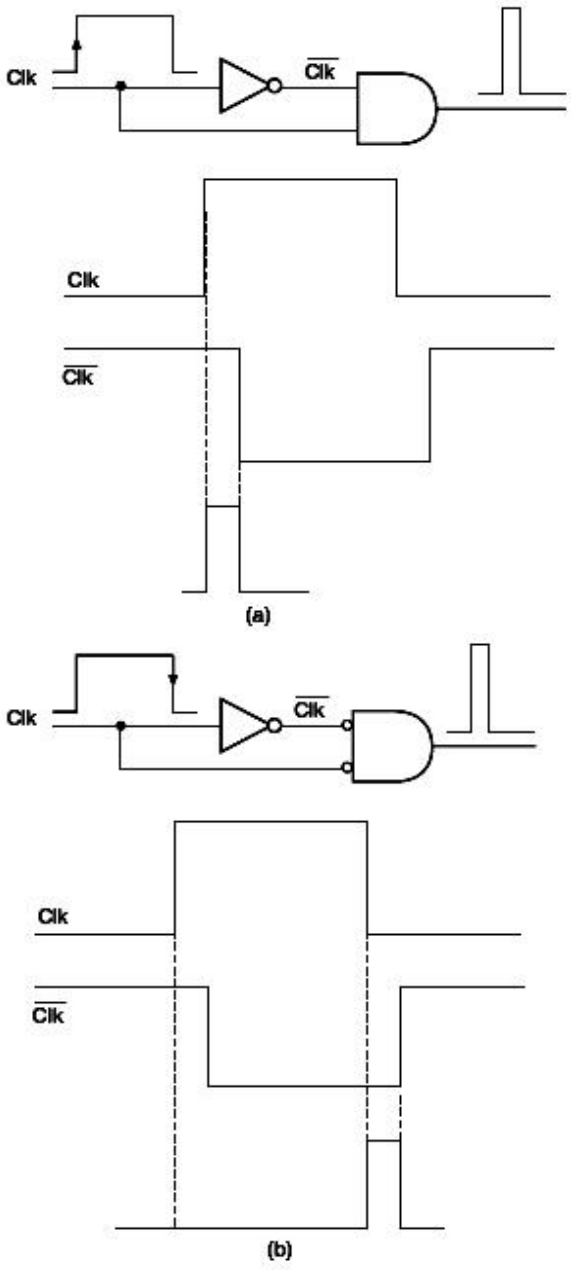

S
Clk
Edge
Detector
R
Q
$\overline{Q}$


S
Clk
FF
R
Q
$\overline{Q}$

(a)


S
Clk
FF
R
Q
$\overline{Q}$

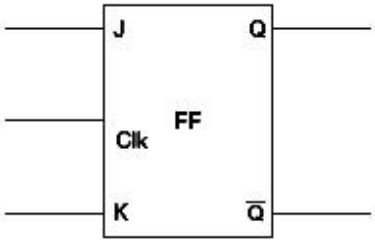
(b)



7.2.2: J-K Flip-Flop

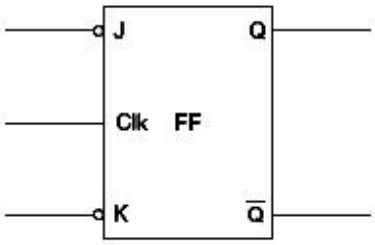
A J-K flip-flop behaves in the same fashion as an R-S flip-flop except for one of the entries in the function table. In the case of an R-S flip-flop, the input combination S = R = 1 (in the case of a flip-flop with active HIGH inputs) and the input combination S = R = 0 (in the case of a flip-flop with active LOW inputs) are prohibited. In the case of a J-K flip-flop with active HIGH inputs, the output of the flip-flop toggles, that is, it goes to the other state, for J = K = 1 . The output toggles for J = K = 0 in the case of the flip-flop having active LOW inputs. Thus, a J-K flip-flop overcomes the problem of a forbidden input combination of the R-S flip-flop. Figures (a) and (b)

respectively show the circuit symbol of level-triggered J-K flip-flops with active HIGH and active LOW inputs, along with their function tables.



Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q}_n$

(a)

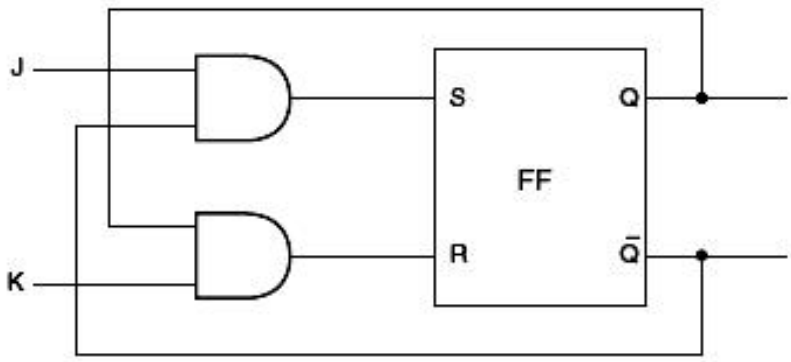


Operation Mode	J	K	Clk	Q_{n+1}
SET	0	1	1	1
RESET	1	0	1	0
NO CHANGE	1	1	1	Q_n
TOGGLE	0	0	1	$\overline{Q}_n$

(b)

Figure shows the realization of a J-K flip-flop with an R-S flip-flop. The characteristic tables for a J-K flip-flop with active HIGH J and K inputs and a J-K flip-flop with active LOW J and K inputs are respectively shown in Figs (a) and (b). The corresponding Karnaugh maps are shown in Fig. (c) for the characteristics table of Fig. (a) and in Fig. (d) for the characteristic table of Fig. (b). The characteristic equations for the Karnaugh maps of Fig (c) and (d) are respectively

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

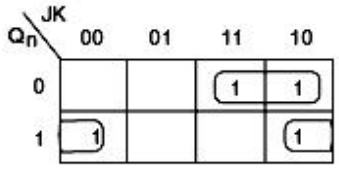


Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

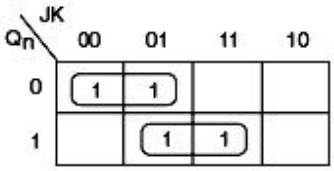
(a)

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

(b)



(c)

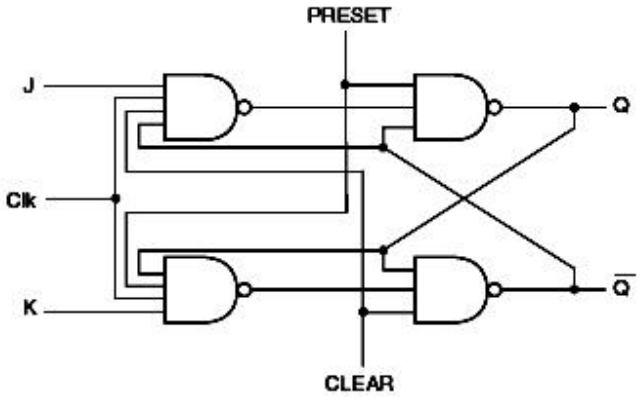


(d)

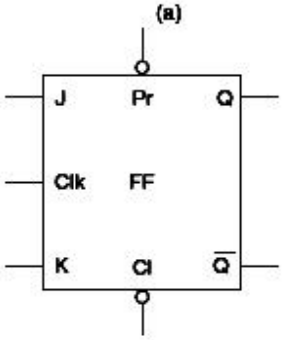
J-K Flip-Flop with PRESET and CLEAR Inputs

It is often necessary to clear a flip-flop to a logic '0' state (Qn = 0) or preset it to a logic '1' state (Qn =1). An example of how this is realized is shown in Fig. The flip-flop is cleared (that is, Qn = 0) whenever the CLEAR input is '0' and the PRESET input is '1'. The flip-flop is preset to the logic '1' state whenever the PRESET input is '0' and the CLEAR input is '1'. Here, the CLEAR and PRESET inputs are active when LOW. Figure shows the circuit symbol of this presettable, clearable, clocked J-K flip-flop. Figure shows the function table of such a flip-flop. It is evident from the function table that, whenever the PRESET input is active, the output goes to the '1' state irrespective of the status of the clock, J and K inputs.

Similarly, when the flip-flop is cleared, that is, the CLEAR input is active, the output goes to the '0' state irrespective of the status of the clock, J and K inputs. In a flip-flop of this type, both PRESET and CLEAR inputs should not be made active at the same time.



(a)



(b)

PR	CL	CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	---	---
1	1	1	0	0	Q_n	$\overline{Q_n}$
1	1	1	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	1	Toggle	
1	1	0	X	X	Q_n	$\overline{Q_n}$

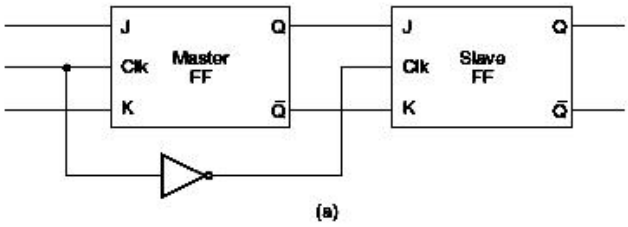
(c)

7.2.3: Master–Slave Flip-Flops

Whenever the width of the pulse clocking the flip-flop is greater than the propagation delay of the flip-flop, the change in state at the output is not reliable. In the case of edge-triggered flip-flops, this pulse width would be the trigger pulse width generated by the edge detector portion of the flip-flop and not the pulse width of the input clock signal. This phenomenon is referred to as the *race problem*. As the propagation delays are normally very small, the

likelihood of the occurrence of a race condition is reasonably high. One way to get over this problem is to use a *master–slave* configuration. Figure (a) shows a master–slave flip-flop constructed with two J-K flip-flops.

The first flip-flop is called the master flip-flop and the second is called the slave. The clock to the slave flip-flop is the complement of the clock to the master flip-flop. When the clock pulse is present, the master flip-flop is enabled while the slave flip-flop is disabled. As a result, the master flip-flop can change state while the slave flip-flop cannot. When the clock goes LOW, the master flip-flop gets disabled while the slave flip-flop is enabled. Therefore, the slave J-K flip-flop changes state as per the logic states at its J and K inputs. The contents of the master flip-flop are therefore transferred to the slave flip-flop, and the master flip-flop, being disabled, can acquire new inputs without affecting the output. As would be clear from the description above, a master– slave flip-flop is a pulse-triggered flip-flop and not an edge-triggered one. Figure (b) shows the truth table of a master–slave J-K flip-flop with active LOW PRESET and CLEAR inputs and active HIGH J and K inputs. The master–slave configuration has become obsolete. The newer IC technologies such as 74LS, 74AS, 74ALS, 74HC and 74HCT do not have master–slave flip-flops in their series.



(a)

PR	CLR	CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	Unstable	
1	1	⎍	0	0	Q_n	$\overline{Q_n}$
1	1	⎍	1	0	1	0
1	1	⎍	0	1	0	1
1	1	⎍	1	1	Toggle	

(b)

7.2.4: Toggle Flip-Flop (*T* Flip-Flop)

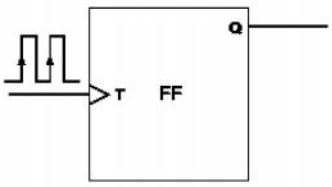
The output of a *toggle flip-flop*, also called a T flip-flop, changes state every time it is triggered at its T input, called the toggle input. That is, the output becomes '1' if it was '0' and '0' if it was '1'. Figures (a) and (b) respectively show the circuit symbols of

positive edge-triggered and negative edge-triggered T flip-flops, along with their function tables.

If we consider the T input as active when HIGH, the characteristic table of such a flip-flop is shown in Fig. (c). If the T input were active when LOW, then the characteristic table would be as shown in Fig. (d). The Karnaugh maps for the characteristic tables of Figs 10.34(c) and (d) are shown in Figs(e) and (f) respectively. The characteristic equations as written from the Karnaugh maps are as follows:

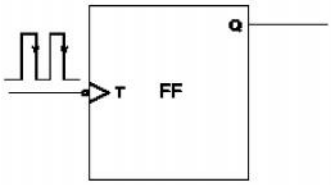
$$Q_{n+1} = T.\overline{Q_n} + \overline{T}.Q_n$$
$$Q_{n+1} = \overline{T.Q_n} + T.Q_n$$

It is obvious from the operational principle of the T flip-flop that the frequency of the signal at the Q output is half the frequency of the signal applied at the T input. A cascaded arrangement of nT flip-flops, where the output of one flip-flop is connected to the T input of the following flip-flop, can be used to divide the input signal frequency by a factor of 2^n. Figure shows a divide-by-16circuit built around a cascaded arrangement of four T flip-flops.



T	Q_n	Q_{n+1}
↑	0	1
↑	1	0

(a)



T	Q_n	Q_{n+1}
↓	0	1
↓	1	0

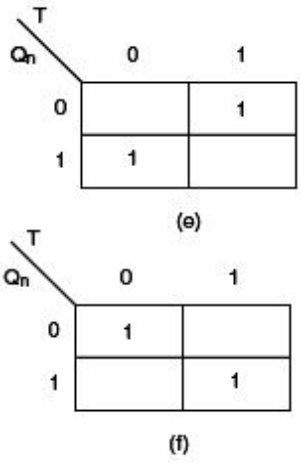
(b)

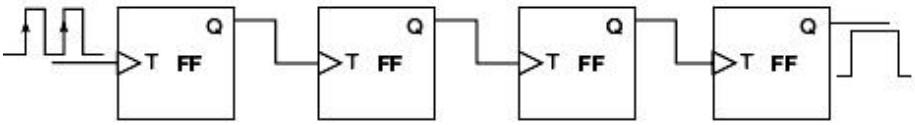
Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

(c)

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

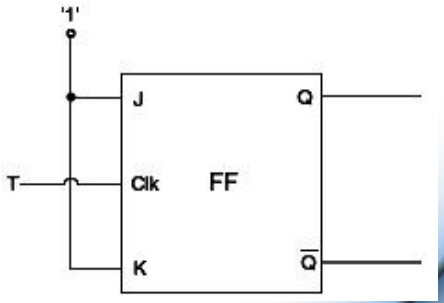
(d)





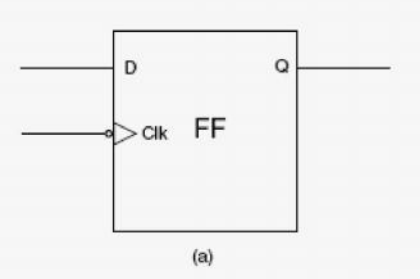
J-K Flip-Flop as a Toggle Flip-Flop

If we recall the function table of a J-K flip-flop, we will see that, when both J and K inputs of the flip-flop are tied to their active level ('1' level if J and K are active when HIGH, and '0' level when J and K are active when LOW), the flip-flop behaves like a toggle flip-flop, with its clock input serving as the T input. In fact, the J-K flip-flop can be used to construct any other flip-flop. That is why it is also sometimes referred to as a universal flip-flop. Figure shows the use of a J-K flip-flop as a T flip-flop.



***7.2.5: D* Flip-Flop**

A D flip-flop, also called a *delay flip-flop*, can be used to provide temporary storage of one bit of information. Figure (a) shows the circuit symbol and function table of a negative edge-triggered D flip-flop. When the clock is active, the data bit (0 or 1) present at the D input is transferred to the output. In the D flip-flop of Fig., the data transfer from D input to Q output occurs on the negative-going (HIGH-to-LOW) transition of the clock input. The D input can acquire new status



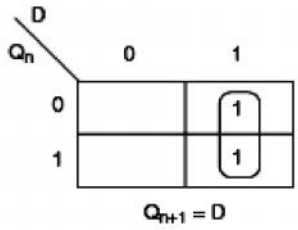
(a)

D	Clk	Q
0	↓	0
1	↓	1

(b)

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

(c)

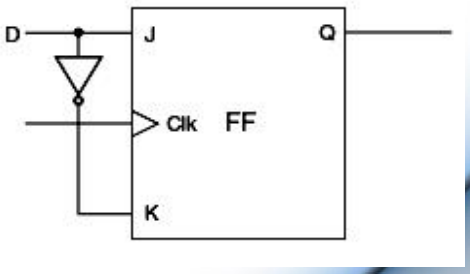


(d)

J-K Flip-Flop as D Flip-Flop

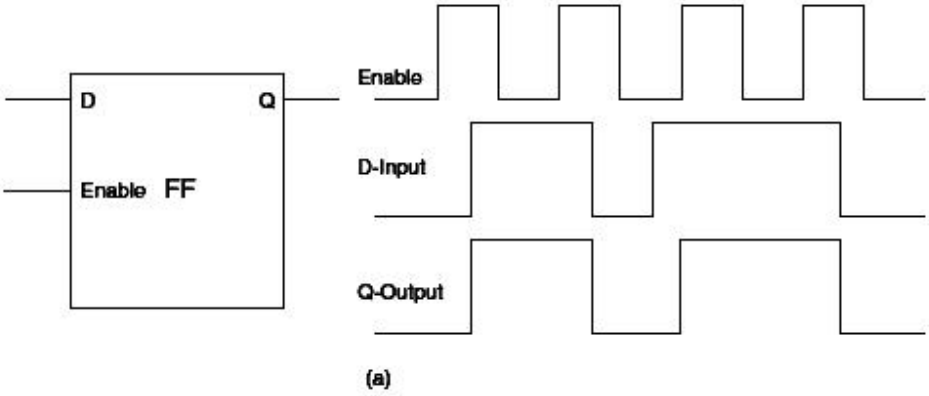
Figure shows how a J-K flip-flop can be used as a D flip-flop. When the D input is a logic '1', the J and K inputs are a logic '1' and '0' respectively. According to the function table of the J-K flip-flop, under these input conditions, the Q output will go to the logic '1' state when clocked.

Also, when the D input is a logic '0', the J and K inputs are a logic '0' and '1' respectively. Again, according to the function table of the J-K flip-flop, under these input conditions, the Q output will go to the logic '0' state when clocked. Thus, in both cases, the D input is passed on to the output when the flip-flop is clocked.

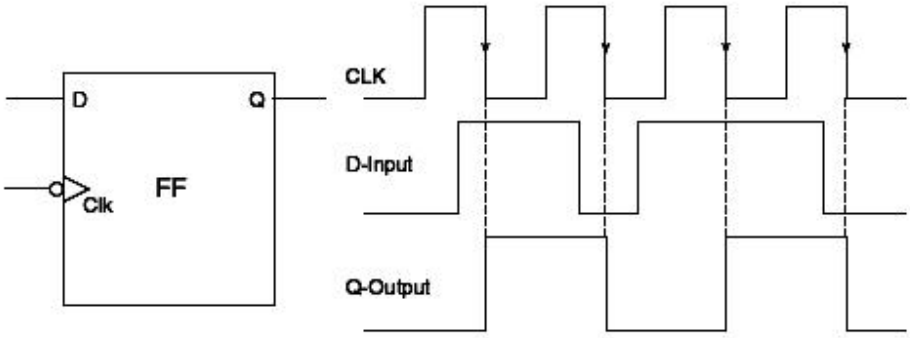


7.2.6 D Latch

In a D latch, the output Q follows the D input as long as the clock input (also called the ENABLE input) is HIGH or LOW, depending upon the clock level to which it responds. When the ENABLE input goes to the inactive level, the output holds on to the logic state it was in just prior to the ENABLE input becoming inactive during the entire time period the ENABLE input is inactive.



(a)



(b)

A D flip-flop should not be confused with a D latch. In a D flip-flop, the data on the D input are transferred to the Q output on the positive- or negative-going transition of the clock signal, depending upon the flip-flop, and this logic state is held at the output until we get the next effective clock transition. The difference between the two is further illustrated in Figs (a) and (b) depicting the functioning of a D latch and a D flip-flop respectively.

3. QUESTIONS:

1. What is Sequential Logic?
2. What is Shift Register?
3. Explain state machine with help of suitable diagram.
4. What is Flip-Flop? Where it is used?
5. Write detail Note on RS Flip-Flop.
6. Explain Level- Triggered and Edged –Triggered Flip-Flops.
7. Explain J-K Flip- Flop with help of suitable diagram.
8. Write short note on Master –Slave Flip-Flop.
9. Explain T Flip-Flop.
10. Explain the J-K Flip-Flop as D Flip-Flop.
11. Write Short note on D Latch.

8

COUNTERS AND REGISTERS

Unit Structure

1. Objectives
2. Introduction
3. Counters
   - 8.2.1: Ripple (Asynchronous) Counter
   - 8.2.2: Binary Ripple Counter
   - 3. : Synchronous Counter
   - 4. : UP/DOWN Counters
   - 5. Presettable Counters
3. Shift Registers
   - 1. : Serial-in to Parallel-out (SIPO) 4-bit Serial-in to Parallel-out Shift Register
   - 2. : 4-bit Serial-in to Serial-out Shift Register
   - 3.: 4-bit Parallel-in to Serial-out Shift Register
   - 8.3.4: 4-bit Parallel-in to Parallel-out Shift Register
   - 8.3.5: Summary of Shift Registers
4. Questions
5. Further Reading

1. OBJECTIVES:

After completing this chapter, you will be able to:

- ❖ Understand the electronics parts like counters & Shift Registers.
- ❖ Understand the structure and working of Asynchronous counters with help of suitable diagrams.
- ❖ Understand the structure and working of Synchronous counters with help of suitable diagrams.
- ❖ Learn the basics of Shift Register of different types with help of appropriate diagrams.

1. INTRODUCTION:

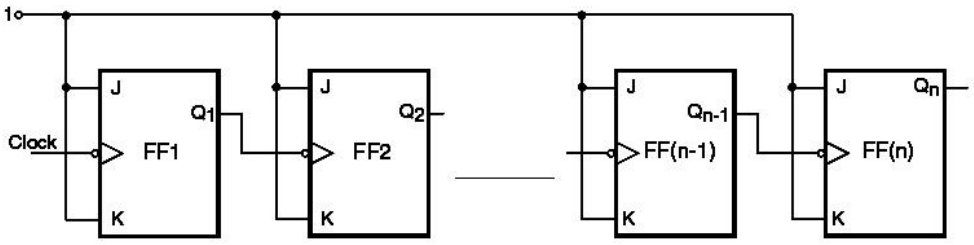
Counters and *registers* belong to the category of MSI sequential logic circuits. They have similar architecture, as both counters and registers comprise a cascaded arrangement of more than one flip-flop with or without combinational logic devices. Both constitute very important building blocks of sequential logic, and different types of counter and register available in integrated circuit (IC) form are used in a wide range of digital systems. While counters are mainly used in counting applications, where they either measure the time interval between two unknown time instants or measure the frequency of a given signal, registers are primarily used for the temporary storage of data present at the output of a digital circuit before they are fed to another digital circuit. We are all familiar with the role of different types of register used inside a microprocessor, and also their use in microprocessor-based applications. Because of the very nature of operation of registers, they form the basis of a very important class of counters called *shift counters*.

2. COUNTERS:

8.2.1: Ripple (Asynchronous) Counter

A *ripple counter* is a cascaded arrangement of flip-flops where the output of one flip-flop drives the clock input of the following flip-flop. The number of flip-flops in the cascaded arrangement depends upon the number of different logic states that it goes through before it repeats the sequence, a parameter known as the modulus of the counter.

In a ripple counter, also called an *asynchronous counter* or a *serial counter*, he clock input is applied only to the first flip-flop, also called the input flip-flop, in the cascaded arrangement. The clock input to any subsequent flip-flop comes from the output of its immediately preceding flip-flop. For instance, the output of the first flip-flop acts as the clock input to the second flip-flop, the output of the second flip-flop feeds the clock input of the third flip-flop and so on. In general, in an arrangement of n flip-flops, the clock input to the nth flip-flop comes from the output of the $(n-1)^{th}$ flip-flop for $n > 1$. Figure shows the generalized block schematic arrangement of an n-bit binary ripple counter.



As a natural consequence of this, not all flip-flops change state at the same time. The second flip-flop can change state only after the output of the first flip-flop has changed its state. That is, the second flip-flop would change state a certain time delay after the occurrence of the input clock pulse owing to the fact that it gets its own clock input from the output of the first flip-flop and not from the input clock. This time delay here equals the sum of propagation delays of two flip-flops, the first and the second flip-flops. In general, the nth flip-flop will change state only after a delay equal to n times the propagation delay of one flip-flop. The term 'ripple counter' comes from the mode in which the clock information ripples through the counter. It is also called an 'asynchronous counter' as different flip-flops comprising the counter do not change state in synchronization with the input clock. In a counter like this, after the occurrence of each clock input pulse, the counter has to wait for a time period equal to the sum of propagation delays of all flip-flops before the next clock pulse can be applied. The propagation delay of each flip-flop, of course, will depend upon the logic family to which it belongs.

Modulus of a Counter

The *modulus* (MOD number) of a counter is the number of different logic states it goes through before it comes back to the initial state to repeat the count sequence. An n-bit counter that counts through all its natural states and does not skip any of the states has a modulus of 2^n . We can see that such counters have a modulus that is an integral power of 2, that is, 2, 4, 8, 16 and so on. These can be modified with the help of additional combinational logic to get a modulus of less than 2^n.

To determine the number of flip-flops required to build a counter having a given modulus, identify the smallest integer m that is either equal to or greater than the desired modulus and is also equal to an integral power of 2. For instance, if the desired modulus is 10, which is the case in a decade counter, the smallest integer greater than or equal to 10 and which is also an integral power of 2 is 16. The number of flip-flops in this case would be 4, as $16 = 2^4$. On the same lines, the number of flip-

flops required to construct counters with MOD numbers of 3, 6, 14, 28 and 63 would be 2, 3, 4, 5 and 6 respectively. In general, the arrangement of a minimum number of N flip-flops can be used to construct any counter with a modulus given by the equation

$$2^{N-1} + 1 \leq \text{modulus} \leq 2^N$$

8.2.2: Binary Ripple Counter

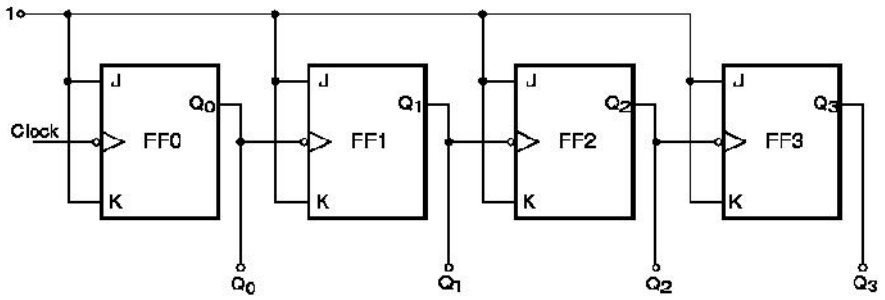
The operation of a binary ripple counter can be best explained with the help of a typical counter of this type. Figure (a) shows a four-bit ripple counter implemented with negative edge-triggered *J-K* flip-flops wired as toggle flip-flops. The output of the first flip-flop feeds the clock input of the second, and the output of the second flip-flop feeds the clock input of the third, the output of which in turn feeds the clock input of the fourth flip-flop. The outputs of the four flip-flops are designated as Q_0 (LSB flip-flop), Q_1, Q_2 and Q_3 (MSB flip-flop). Figure (b) shows the waveforms appearing at Q_0, Q_1, Q_2 and Q_3 outputs as the clock signal goes through successive cycles of trigger pulses. The counter functions as follows.

Let us assume that all the flip-flops are initially cleared to the '0' state. On HIGH-to-LOW transition of the first clock pulse, Q_0 goes from '0' to '1' owing to the toggling action. As the flip-flops used are negative edge-triggered ones, the '0' to '1' transition of Q_0 does not trigger flip-flop FF1. FF1, along with FF2 and FF3, remains in its '0' state. So, on the occurrence of the first negative-going clock transition, $Q_0 = 1$, $Q_1 = 0$, $Q_2 = 0$ and $Q_3 = 0$.

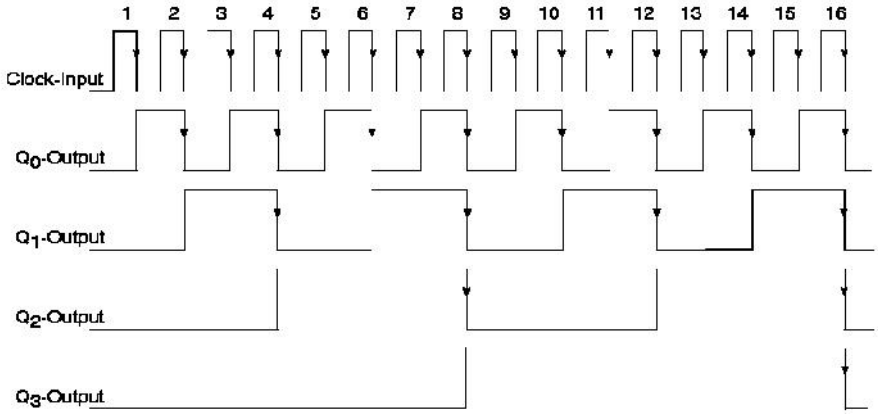
On the HIGH-to-LOW transition of the second clock pulse, Q_0 toggles again. That is, it goes from '1' to '0'. This '1' to '0' transition at the Q_0 output triggers FF1, the output Q_1 of which goes from '0' to '1'. The Q_2 and Q_3 outputs remain unaffected. Therefore, immediately after the occurrence of the second HIGH-to-LOW transition of the clock signal, $Q_0 = 0$, $Q_1 = 1$, $Q_2 = 0$ and $Q_3 = 0$. On similar lines, we can explain the logic status of Q_0, Q_1, Q_2 and Q_3 outputs immediately after subsequent clock transitions. The logic status of outputs for the first 16 relevant (HIGH-to-LOW in the present case) clock signal transitions is summarized in Table.

Thus, we see that the counter goes through 16 distinct states from 0000 to 1111 and then, on the occurrence of the desired transition of the sixteenth clock pulse, it resets to the original state of 0000 from where it had started. In general, if we had N flip-flops, we could count up to 2^N pulses before the

counter resets to the initial state. We can also see from the Q_0 , Q_1 , Q_2 and Q_3 waveforms, as shown



(a)



(b)

Clock signal transition number	Q_0	Q_1	Q_2	Q_3
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	1	1	0	0
After fourth clock transition	0	0	1	0
After fifth clock transition	1	0	1	0
After sixth clock transition	0	1	1	0
After seventh clock transition	1	1	1	0
After eighth clock transition	0	0	0	1
After ninth clock transition	1	0	0	1
After tenth clock transition	0	1	0	1
After eleventh clock transition	1	1	0	1
After twelfth clock transition	0	0	1	1
After thirteenth clock transition	1	0	1	1
After fourteenth clock transition	0	1	1	1
After fifteenth clock transition	1	1	1	1
After sixteenth clock transition	0	0	0	0

in Fig. 11.2(b), that the frequencies of the Q_0, Q_1, Q_2 and Q_3 waveforms are $f/2$, $f/4$, $f/8$ and $f/16$ respectively. Here, f is the frequency of the clock input. This implies that a counter of this type can be used as a divide-by-2^N circuit, where N is the number of flip-flops in the counter chain. In fact, such a counter provides frequency-divided outputs of $f/2^N$, $f/2^{N-1}$, $f/2^{N-2}$, $f/2^{N-3}$, , , $f/2$ at the outputs of the Nth, $(N-1)$th, $(N-2)$th, $(N-3)$th, , first flip-flops. In the case of a four-bit counter of the type shown in Fig. 11.2(a), outputs are available at $f/2$ from the Q_0 output, at $f/4$ from the Q_1 output, at $f/8$ from the Q_2 output and at $f/16$ from the Q_3 output. It may be noted that frequency division is one of the major applications of counters.

Binary Ripple Counters with a Modulus of Less than 2^N

An N-flip-flop binary ripple counter can be modified, as we will see in the following paragraphs, to have any other modulus less than 2^N with the help of simple externally connected combinational logic. We will illustrate this simple concept with the help of an example.

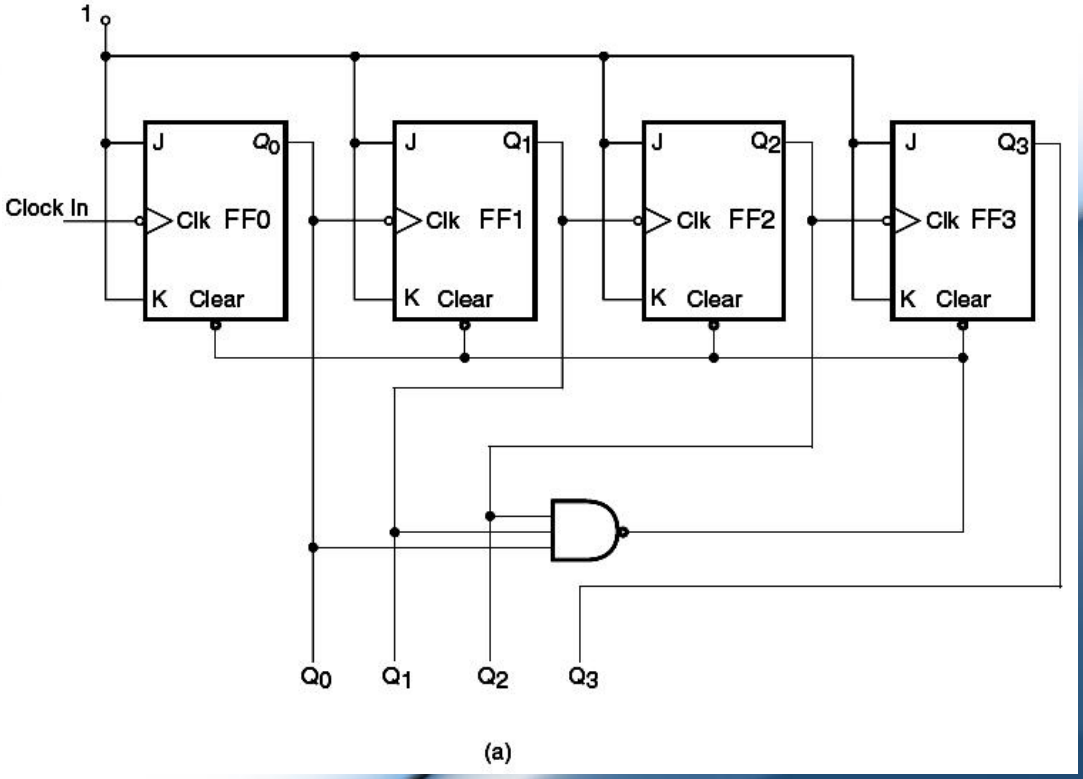
Consider the four-flip-flop binary ripple counter arrangement of Fig. (a). It uses *J-K* flip-flops with an active LOW asynchronous CLEAR input. The NAND gate in the figure has its output connected to the CLEAR inputs of all four flip-flops. The inputs to this three-input NAND gate are from the Q outputs of flip-flops FF0, FF1 and FF2. If we disregard the NAND gate for some time, this counter will go through its natural binary sequence from 0000 to 1111. But that is not to happen in the present arrangement. The counter does start counting from 0000 towards its final count of 1111. The counter keeps counting as long as the asynchronous CLEAR inputs of the different flip-flops are inactive. That is, the NAND gate output is HIGH. This is the case until the counter reaches 0110. With the seventh clock pulse it tends to go to 0111, which makes all NAND gate inputs HIGH, forcing its output to LOW. This HIGH-to-LOW transition at the NAND gate output clears all flip-flop outputs to the logic '0' state, thus disallowing the counter to settle at 0111. From the eighth clock pulse onwards, the counter repeats the sequence. The counter thus always counts from 0000 to 0110 and resets back to 0000. The remaining nine states, which include 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111, are skipped, with the result that we get an MOD-7 counter.

Figure (b) shows the timing waveforms for this counter. By suitably choosing NAND inputs, one can get a counter with any MOD number less than 16. Examination of timing waveforms also

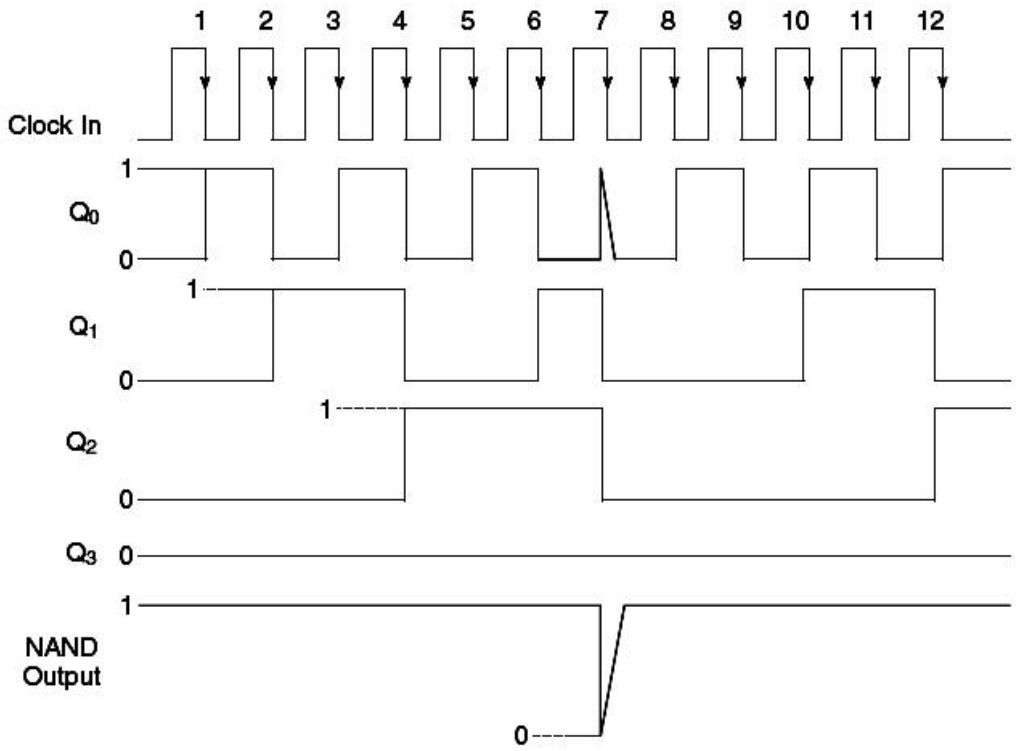
reveals that the frequency of the Q_2 output is one-seventh of the input clock frequency.

The waveform at the Q_2 output is, however, not symmetrical as it would be if the counter were to go through its full binary sequence. The Q_3 output stays in the logic LOW state. It is expected to be so because an MOD-7 counter needs a minimum of three flip-flops. That is why the fourth flip-flop, which was supposed to toggle on the HIGH-to-LOW transition of the eighth clock pulse, and on every successive eighth pulse thereafter, never gets to that stage. The counter is cleared on the seventh clock pulse and every successive seventh clock pulse thereafter.

As another illustration, if the NAND gate used in the counter arrangement of Fig. (a) is a two-input NAND and its inputs are from the Q_1 and Q_3 outputs, the counter will go through 0000 to 1001 and then reset to 0000 again, as, the moment the counter tends to switch from the 1001 to the 1010 state, the NAND gate goes from the '1' to the '0' state, clearing all flip-flops to the '0' state.



(a)



(b)

Steps to be followed to design any binary ripple counter that starts from 0000 and has a modulus of X are summarized as follows:

1. Determine the minimum number of flip-flops N so that $2^N \geq X$ Connect these flip-flops as a binary ripple counter. If $2^N = X$, do not go to steps 2 and 3.

2.Identify the flip-flops that will be in the logic HIGH state at the count whose decimal equivalent is X. Choose a NAND gate with the number of inputs equal to the number of flip-flops that would be in the logic HIGH state. As an example, if the objective were to design an MOD-12 counter, then, in the corresponding count, that is, 1100, two flip-flops would be in the logic HIGH state. The desired NAND gate would therefore be a two-input gate.

3.Connect the Q outputs of the identified flip-flops to the inputs of the NAND gate and the NAND gate output to asynchronous clear inputs of all flip-flops.

8.2.3: Synchronous Counter

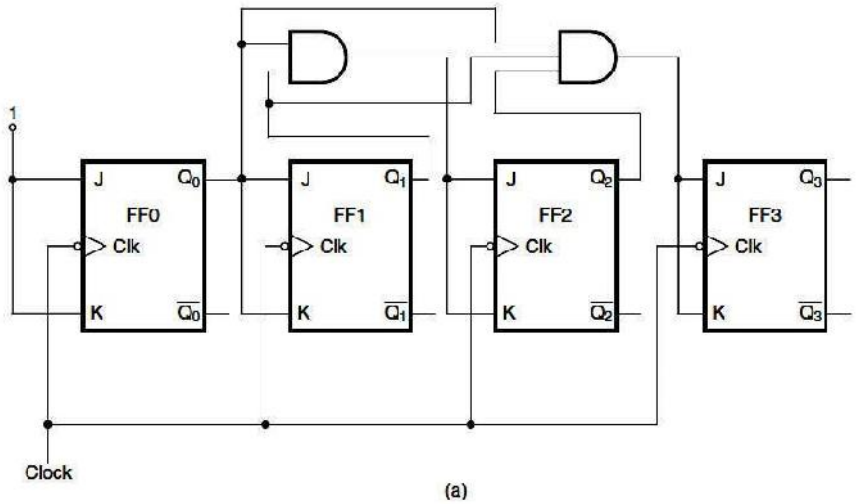
In a *synchronous counter*, also known as a *parallel counter*, all the flip-flops in the counter change state at the same time in synchronism with the input clock signal. The clock signal in this case is simultaneously applied to the clock inputs of all the flip-flops. The delay involved in this case is equal to the propagation

delay of one flip-flop only, irrespective of the number of flip-flops used to construct the counter. In other words, the delay is independent of the size of the counter.

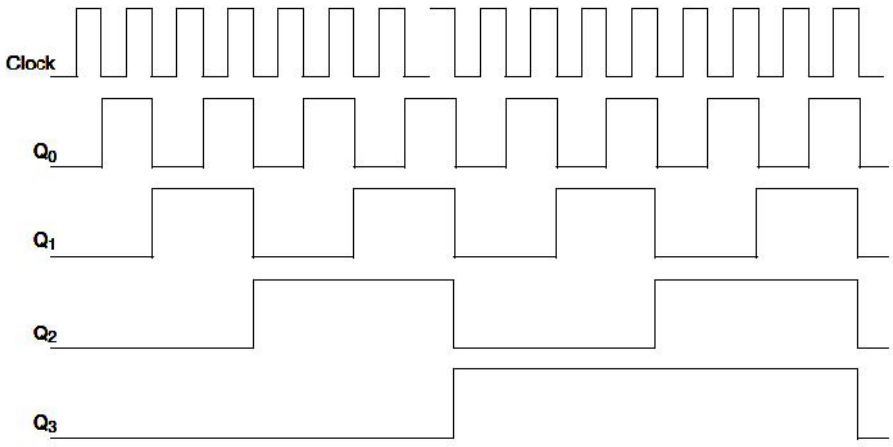
Ripple counters discussed thus far in this chapter are asynchronous in nature as the different flipflops comprising the counter are not clocked simultaneously and in synchronism with the clock pulses. The total propagation delay in such a counter, as explained earlier, is equal to the sum of propagation delays due to different flip-flops. The propagation delay becomes prohibitively large in a ripple counter with a large count. On the other hand, in a synchronous counter, all flip-flops in the counter are clocked simultaneously in synchronism with the clock, and as a consequence all flip-flops change state at the same time. The propagation delay in this case is independent of the number of flip-flops used.

Since the different flip-flops in a synchronous counter are clocked at the same time, there needs to be additional logic circuitry to ensure that the various flip-flops toggle at the right time. For instance, if we look at the count sequence of a four-bit binary counter shown in Table, we find that flip-flop FF0 toggles with every clock pulse, flip-flop FF1 toggles only when the output of FF0 is in the '1' state, flip-flop FF2 toggles only with those clock pulses when the outputs of FF0 and FF1 are both in the logic '1' state and flip-flop FF3 toggles only with those clock pulses when Q0,Q1 and Q2 are all in the logic '1' state. Such logic can be easily implemented with AND gates. Figure (a) shows the schematic arrangement of a four-bit synchronous counter. The timing waveforms are shown in Fig. (b). The diagram is self-explanatory. As an example, ICs 74162 and 74163 are four-bit synchronous counters, with the former being a decade counter and the latter a binary counter.

A synchronous counter that counts in the reverse or downward sequence can be constructed in a similar manner by using complementary outputs of the flip-flops to drive the J and K inputs of the following flip-flops. Refer to the reverse or downward count sequence as given in Table. As is evident from the table, FF0 toggles with every clock pulse, FF1 toggles only when Q0 is logic '0', FF2 toggles only when both Q0 and Q1 are in the logic '0' state and FF3 toggles only when Q0, Q1 and Q2 are in the logic '0' state. Referring to the four-bit synchronous UP counter of Fig. (a), if the J and K inputs of flip-flop FF1 are fed from the Q0 output instead of the Q0 output, the inputs to the two-input AND gate are and instead of Q0 and Q1, and the inputs to the three-input AND gate are, and instead of Q0, Q1 and Q2, we get a counter that counts in reverse order. In that case it becomes a four-bit synchronous DOWN counter.



(a)



(b)

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	1	1	1	1	9	0	1	1	1
2	1	1	1	0	10	0	1	1	0
3	1	1	0	1	11	0	1	0	1
4	1	1	0	0	12	0	1	0	0
5	1	0	1	1	13	0	0	1	1
6	1	0	1	0	14	0	0	1	0
7	1	0	0	1	15	0	0	0	1

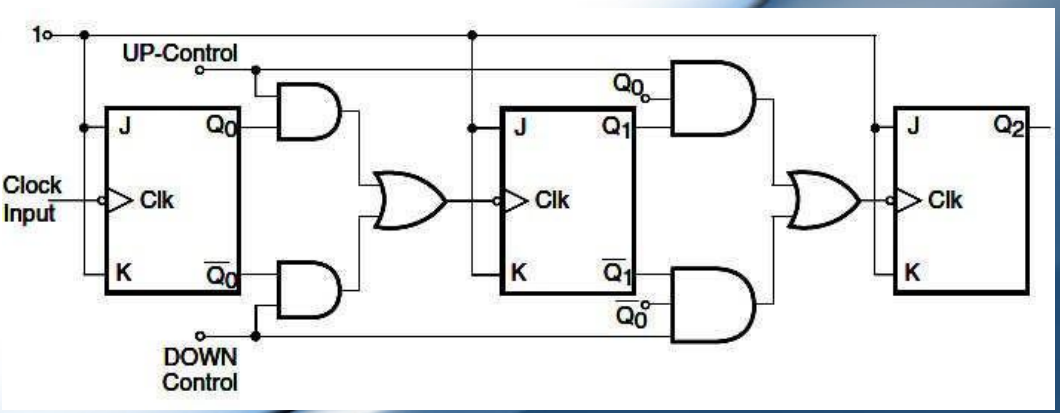
8.2.4: UP/DOWN Counters

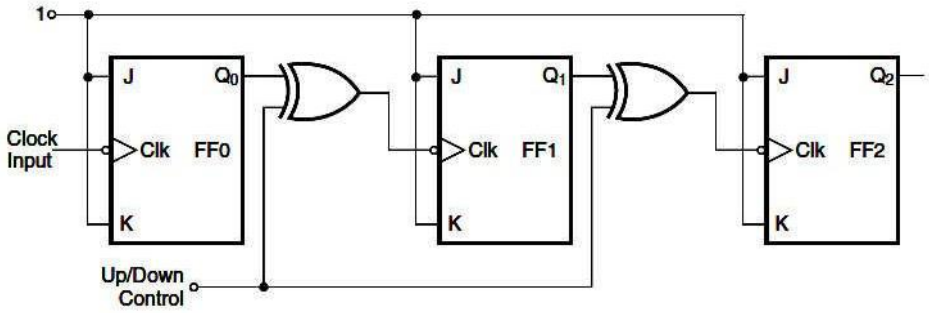
Counters are also available in integrated circuit form as UP/DOWN counters, which can be made to operate as either UP or DOWN counters. As outlined in Section 11.5, an UP counter is one that counts upwards or in the forward direction by one LSB

every time it is clocked. A four-bit binary UP counter will count as 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 0000, 0001 and so on. A DOWN counter counts in the reverse direction or downwards by one LSB every time it is clocked. The four-bit binary DOWN counter will count as 0000, 1111, 1110, 1101, 1100, 1011, 1010, 1001, 1000, 0111, 0110, 0101, 0100, 0011, 0010, 0001, 0000, 1111 and so on.

Some counter ICs have separate clock inputs for UP and DOWN counts, while others have a single clock input and an UP/DOWN control pin. The logic status of this control pin decides the counting mode. As an example, ICs 74190 and 74191 are four-bit UP/DOWN counters in the TTL family with a single clock input and an UP/DOWN control pin. While IC 74190 is a BCD decade counter, IC 74191 is a binary counter. Also, ICs 74192 and 74193 are four-bit UP/DOWN counters in the TTL family, with separate clock input terminals for UP and DOWN counts. While IC 74192 is a BCD decade counter, IC 74193 is a binary counter.

Figure shows a three-bit binary UP/DOWN counter. This is only one possible logic arrangement. As we can see, the counter counts upwards when UP control is logic '1' and DOWN control is logic '0'. In this case the clock input of each flip-flop other than the LSB flip-flop is fed from the normal output of the immediately preceding flip-flop. The counter counts downwards when the UP control input is logic '0' and DOWN control is logic '1'. In this case, the clock input of each flip-flop other than the LSB flip-flop is fed from the complemented output of the immediately preceding flip-flop. Figure shows another possible configuration for a three-bit binary ripple UP/DOWN counter. It has a common control input. When this input is in logic '1' state the counter counts downwards, and when it is in logic '0' state it counts upwards.





8.2.5: Presettable Counters

Presettable counters are those that can be preset to any starting count either asynchronously (independently of the clock signal) or synchronously (with the active transition of the clock signal). The presetting operation is achieved with the help of PRESET and CLEAR (or MASTER RESET) inputs available on the flip-flops. The presetting operation is also known as the 'preloading' or simply the 'loading' operation.

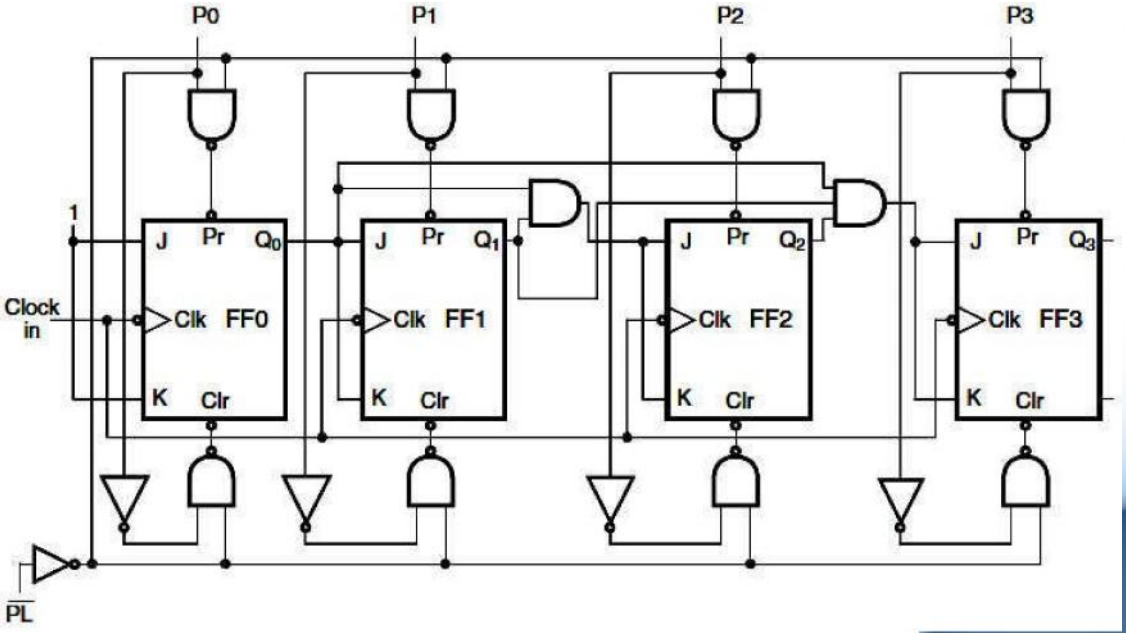
Presettable counters can be UP counters, DOWN counters or UP/DOWN counters. Additional inputs/outputs available on a presettable UP/DOWN counter usually include PRESET inputs, from where any desired count can be loaded, parallel load (*PL*) inputs, which when active allow the PRESET inputs to be loaded onto the counter outputs, and terminal count (*TC*) outputs, which become active when the counter reaches the terminal count.

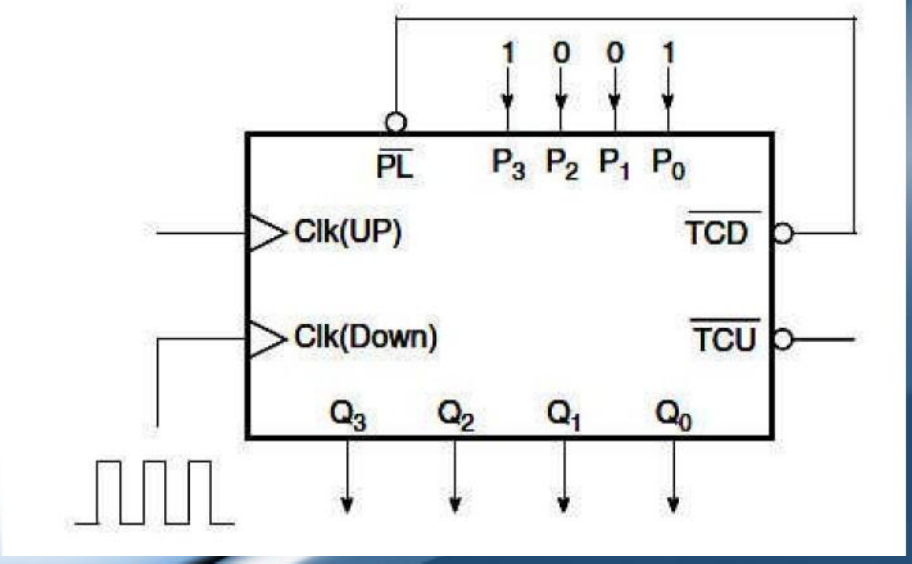
Figure shows the logic diagram of a four-bit presettable synchronous UP counter. The data available on P_3 , P_2 , P_1 and P_0 inputs are loaded onto the counter when the parallel load) input goes LOW.

When the input goes LOW, one of the inputs of all NAND gates, including the four NAND gates connected to the PRESET inputs and the four NAND gates connected to the CLEAR inputs, goes to the logic '1' state. What reaches the PRESET inputs of FF3, FF2, FF1 and FF0 is , , and respectively, and what reaches their CLEAR inputs is P_3 , P_2 , P_1 and P_0 respectively. Since PRESET and CLEAR are active LOW inputs, the counter flip-flops FF3, FF2, FF1 and FF0 will respectively be loaded with P_3 , P_2 , P_1 and P_0 . For example, if P_3= 1, the PRESET and CLEAR inputs of FF3 will be in the '0' and '1' logic states respectively. This implies that the Q_3 output will go to the logic '1' state. Thus, FF3 has been loaded with P_3 . Similarly, if P_3 = 0, the PRESET and CLEAR inputs of flip-flop FF3 will be in the '1' and '0' states respectively. The flip-flop output (Q_3 output) will

be cleared to the '0' state. Again, the flip-flop is loaded with P_3 logic status when the input becomes active.

Counter ICs 74190, 74191, 74192 and 74193 are asynchronously presettable synchronous UP/DOWN counters. Many synchronous counters use synchronous presetting whereby the counter is preset or loaded with the data on the active transition of the same clock signal that is used for counting. Presettable counters also have terminal count ( ) outputs, which allow them to be cascaded together to get counters with higher MOD numbers. In the cascade arrangement, the terminal count output of the lower-order counter feeds the clock input of the next higher-order counter.





8.3 SHIFT REGISTERS:

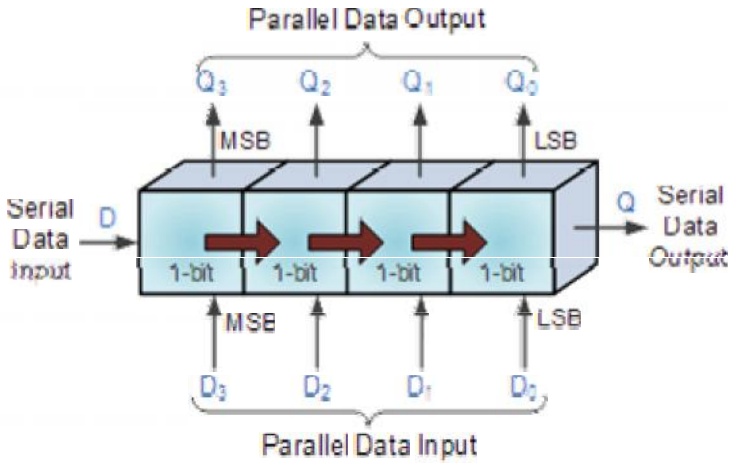
The **Shift Register** is another type of sequential logic circuit that is used for the storage or transfer of data in the form of binary numbers and then "shifts" the data out once every clock cycle, hence the name "shift register". It basically consists of several single bit "D-Type Data Latches", one for each bit (0 or 1) connected together in a serial or daisy-chain arrangement so that the output from one data latch becomes the input of the next latch and so on. The data bits may be fed in or out of the register serially, i.e. one after the other from either the left or the right direction, or in parallel, i.e. all together. The number of individual data latches required to make up a single **Shift Register** is determined by the number of bits to be stored with the most common being 8-bits wide, i.e. eight individual data latches.

The Shift Register is used for data storage or data movement and are used in calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The individual data latches that make up a single shift register are all driven by a common clock (**Clk**) signal making them synchronous devices. Shift register IC's are generally provided with a *clear* or *reset* connection so that they can be "SET" or "RESET" as required.

Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

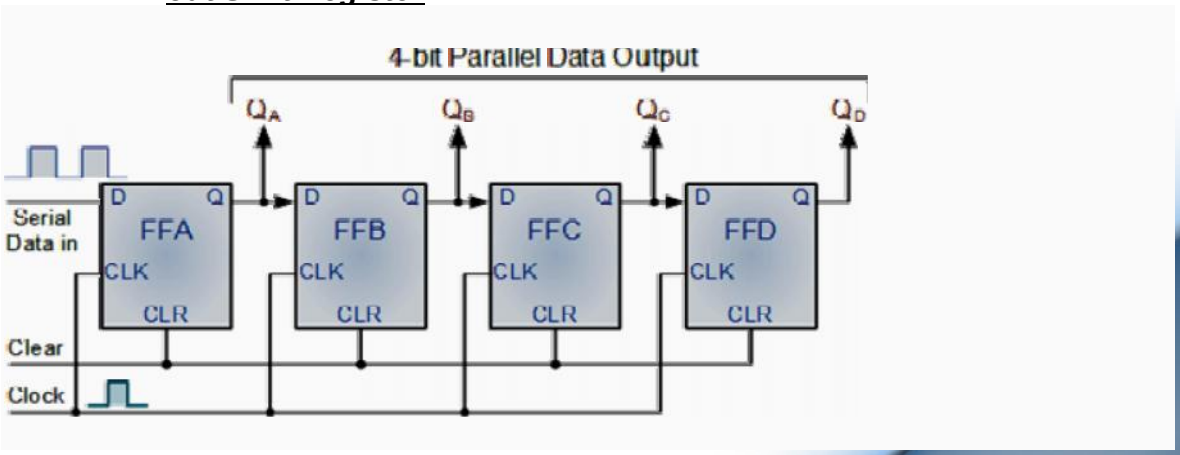
- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available in parallel form.
- Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- Parallel-in to Parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

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



Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

8.3.1: Serial-in to Parallel-out (SIPO) 4-bit Serial-in to Parallel-out Shift Register



The operation is as follows. Lets assume that all the flip-flops (FFA to FFD) have just been RESET (CLEAR input) and that all the outputs Q_A to Q_D are at logic level "0" i.e, no parallel data output. If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0". Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.

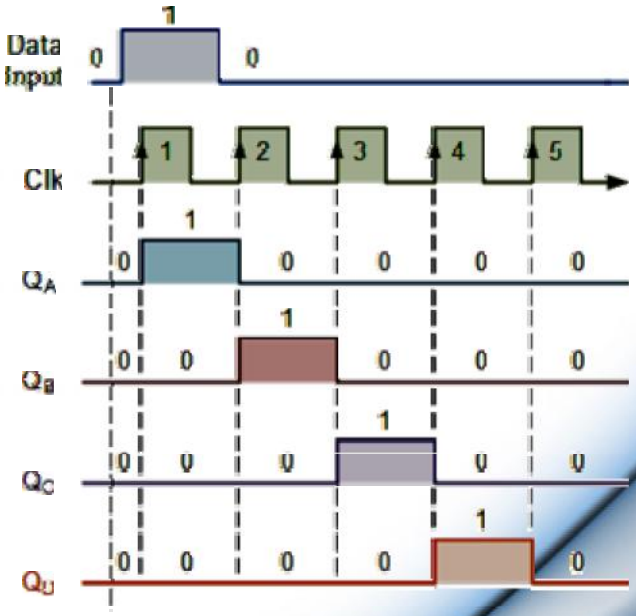
The second clock pulse will change the output of FFA to logic "0" and the output of FFB and Q_B HIGH to logic "1" as its input D has the logic "1" level on it from Q_A. The logic "1" has now moved or been "shifted" one place along the register to the right as

it is now at Q_A. When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of Q_A to Q_D. Then the data has been converted from a serial data input signal to a parallel data output. The truth table and following waveforms show the propagation of the logic "1" through the register from left to right as follows.

Basic Movement of Data through a Shift Register

Clock Pulse No	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0



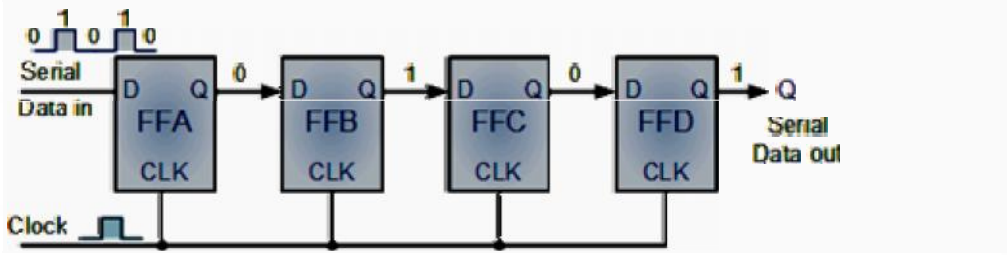
Note that after the fourth clock pulse has ended the 4-bits of data (0-0-0-1) are stored in the register and will remain there provided clocking of the register has stopped. In practice the input data to the register may consist of various combinations of logic "1" and "0". Commonly available SIPO IC's include the standard 8-bit 74LS164 or the 74LS594.

Serial-in to Serial-out (SISO)

This **shift register** is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs Q_A to Q_D, this time the data is allowed to flow straight through the register and out of the other end. Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name **Serial-in to Serial-Out Shift Register** or **SISO**.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk). The logic circuit diagram below shows a generalized serial-in serial-out shift register.

8.3.2: 4-bit Serial-in to Serial-out Shift Register



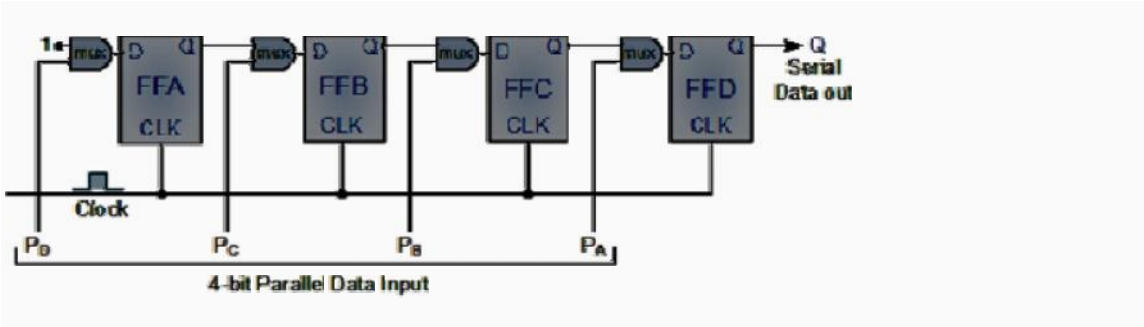
You may think what's the point of a SISO shift register if the output data is exactly the same as the input data. Well this type of **Shift Register** also acts as a temporary storage device or as a time delay device for the data, with the amount of time delay being controlled by the number of stages in the register, 4, 8, 16 etc or by varying the application of the clock pulses. Commonly available IC's include the 74HC595 8-bit Serial-in/Serial-out Shift Register all with 3-state outputs.

Parallel-in to Serial-out (PISO)

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format i.e. all the data bits enter their inputs simultaneously, to the parallel input pins P_A to P_D of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at P_A to P_D. This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this system a clock pulse is not required to parallel load the register as it is

already present, but four clock pulses are required to unload the data.

8.3.3: 4-bit Parallel-in to Serial-out Shift Register

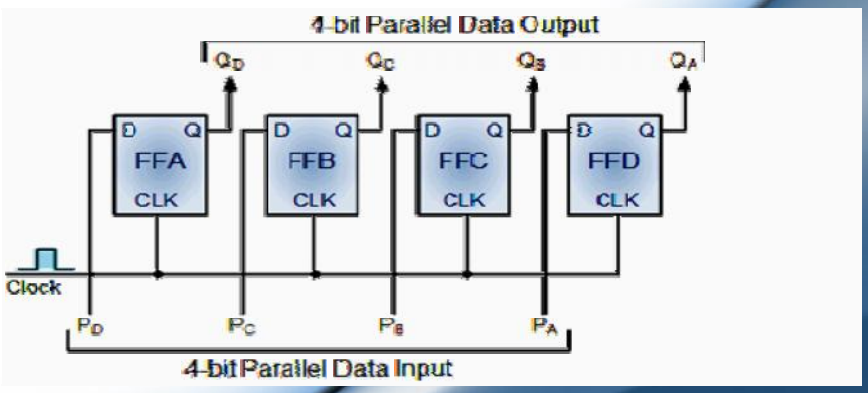


As this type of shift register converts parallel data, such as an 8-bit data word into serial format, it can be used to multiplex many different input lines into a single serial DATA stream which can be sent directly to a computer or transmitted over a communications line. Commonly available IC's include the 74HC166 8-bit Parallel-in/Serial-out Shift Registers.

Parallel-in to Parallel-out (PIPO)

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins P_A to P_D and then transferred together directly to their respective output pins Q_A to Q_A by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

8.3.4: 4-bit Parallel-in to Parallel-out Shift Register



The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input

(PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

8.3.5:Summary of Shift Registers

- Then to summarise.
- A simple **Shift Register** can be made using only D-type flip-Flops, one flip-Flop for each data bit.
- The output from each flip-Flop is connected to the D input of the flip-flop at its right.
- Shift registers hold the data in their memory which is moved or "shifted" to their required positions on each clock pulse.
- Each clock pulse shifts the contents of the register one bit position to either the left or the right.
- The data bits can be loaded one bit at a time in a series input (SI) configuration or be loaded simultaneously in a parallel configuration (PI).
- Data may be removed from the register one bit at a time for a series output (SO) or removed all at the same time from a parallel output (PO).
- One application of shift registers is converting between serial and parallel data.
- Shift registers are identified as SIPO, SISO, PISO, PIPO, and universal shift registers.

9

COMPUTER ORGANISATION

Unit Structure

1. Objectives
2. Computers
   9.1.1: Functional Units
   9.1.2: Control Unit (CU)
   9.1.3: Memory System in a Computer
   9.1.4: Secondary Storage
   9.1.5: Input Output Devices
2. Questions
3. Further Reading

1. OBJECTIVES:

After completing this chapter, you will be able to:

- Learn the basics about computers.
- Understand the structure & working of computer.
- Understand of different parts of computers and their work.
- Learn about the memory organization within computers.

2. COMPUTERS:

A computer as shown in Fig. performs basically five major operations or functions irrespective of their size and make. These are 1) it accepts data or instructions by way of input, 2) it stores data, 3) it can process data as required by the user, 4) it gives results in the form of output, and 5) it controls all operations inside a computer. We discuss below each of these operations.

1. Input: This is the process of entering data and programs in to the computer system. You should know that computer is an electronic machine like any other machine which takes as inputs raw data and performs some processing giving out processed data. Therefore, the input unit takes data from us to the computer in an organized manner for processing.

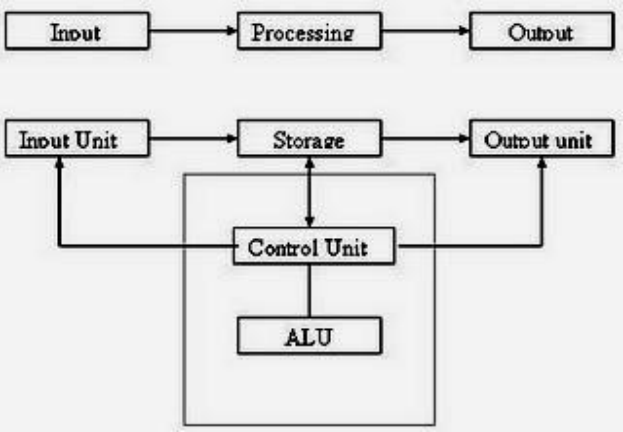


Fig. Basic computer Operations

2.Storage: The process of saving data and instructions permanently is known as storage. Data has to be fed into the system before the actual processing starts. It is because the processing speed of Central Processing Unit (CPU) is so fast that the data has to be provided to CPU with the same speed. Therefore the data is first stored in the storage unit for faster access and processing. This storage unit or the primary storage of the computer system is designed to do the above functionality. It provides space for storing data and instructions.

The storage unit performs the following major functions:

- All data and instructions are stored here before and after processing.
- Intermediate results of processing are also stored here.

3.Processing: The task of performing operations like arithmetic and logical operations is called processing. The Central Processing Unit (CPU) takes data and instructions from the storage unit and makes all sorts of calculations based on the instructions given and the type of data provided. It is then sent back to the storage unit.

4.Output: This is the process of producing results from the data for getting useful information. Similarly the output produced by the computer after processing must also be kept somewhere inside the computer before being given to you in human readable form. Again the output is also stored inside the computer for further processing.

5.Control: The manner how instructions are executed and the above operations are performed. Controlling of all operations like input, processing and output are performed by control unit. It takes care of step by step processing of all operations inside the computer.

9.1.1: FUNCTIONAL UNITS

In order to carry out the operations mentioned in the previous section the computer allocates the task between its various functional units. The computer system is divided into three separate units for its operation. They are 1) arithmetic logical unit, 2) control unit, and 3) central processing unit.

Arithmetic Logical Unit (ALU)

After you enter data through the input device it is stored in the primary storage unit. The actual processing of the data and instruction are performed by Arithmetic Logical Unit. The major operations performed by the ALU are addition, subtraction, multiplication, division, logic and comparison. Data is transferred to ALU from storage unit when required. After processing the output is returned back to storage unit for further processing or getting stored.

2. :Control Unit (CU)

The next component of computer is the Control Unit, which acts like the supervisor seeing that things are done in proper fashion. The control unit determines the sequence in which computer programs and instructions are executed. Things like processing of programs stored in the main memory, interpretation of the instructions and issuing of signals for other units of the computer to execute them. It also acts as a switch board operator when several users access the computer simultaneously. Thereby it coordinates the activities of computer's peripheral equipment as they perform the input and output. Therefore it is the manager of all operations mentioned in the previous section.

Central Processing Unit (CPU)

The ALU and the CU of a computer system are jointly known as the central processing unit. You may call CPU as the brain of any computer system. It is just like brain that takes all major decisions, makes all sorts of calculations and directs different parts of the computer functions by activating and controlling the operations.

3. : MEMORY SYSTEM IN A COMPUTER

There are two kinds of computer memory: *primary* and *secondary*. Primary memory is accessible directly by the processing unit. RAM is an example of primary memory. As soon as the computer is switched off the contents of the primary memory is lost. You can store and retrieve data much faster with primary

memory compared to secondary memory. Secondary memory such as floppy disks, magnetic disk, etc., is located outside the computer. Primary memory is more expensive than secondary memory. Because of this the size of primary memory is less than that of secondary memory. We will discuss about secondary memory later on.

Computer memory is used to store two things: i) instructions to execute a program and ii) data. When the computer is doing any job, the data that have to be processed are stored in the primary memory. This data may come from an input device like keyboard or from a secondary storage device like a floppy disk.

As program or the set of instructions is kept in primary memory, the computer is able to follow instantly the set of instructions. For example, when you book ticket from railway reservation counter, the computer has to follow the same steps: take the request, check the availability of seats, calculate fare, wait for money to be paid, store the reservation and get the ticket printed out. The programme containing these steps is kept in memory of the computer and is followed for each request.

But inside the computer, the steps followed are quite different from what we see on the monitor or screen. In computer's memory both programs and data are stored in the binary form. You have already been introduced with decimal number system, that is the numbers 1 to 9 and 0. The binary system has only two values 0 and 1. These are called *bits*. As human beings we all understand decimal system but the computer can only understand binary system. It is because a large number of integrated circuits inside the computer can be considered as switches, which can be made ON, or OFF. If a switch is ON it is considered 1 and if it is OFF it is 0. A number of switches in different states will give you a message like this: 110101....10. So the computer takes input in the form of 0 and 1 and gives output in the form 0 and 1 only. Is it not absurd if the computer gives outputs as 0's & 1's only? But you do not have to worry about. Every number in binary system can be converted to decimal system and vice versa; for example, 1010 meaning decimal 10. Therefore it is the computer that takes information or data in decimal form from you, convert it in to binary form, process it producing output in binary form and again convert the output to decimal form.

The primary memory as you know in the computer is in the form of IC's (Integrated Circuits). These circuits are called Random Access Memory (RAM). Each of RAM's locations stores one *byte* of information. (One *byte* is equal to *8 bits*). A bit is an acronym for *binary digit*, which stands for one binary piece of information. This can be either 0 or 1. You will know more about RAM later. The

Primary or internal storage section is made up of several small storage locations (ICs) called cells. Each of these cells can store a fixed number of bits called *word length***.**

Each cell has a unique number assigned to it called the address of the cell and it is used to identify the cells. The address starts at 0 and goes up to (N-1). You should know that the memory is like a large cabinet containing as many drawers as there are addresses on memory. Each drawer contains a word and the address is written on outside of the drawer.

Capacity of Primary Memory

You know that each cell of memory contains one character or 1 byte of data. So the capacity is defined in terms of byte or words. Thus 64 kilobyte (KB) memory is capable of storing 64 ´ 1024 = 32,768 bytes. (1 kilobyte is 1024 bytes). A memory size ranges from few kilobytes in small systems to several thousand kilobytes in large mainframe and super computer. In your personal computer you will find memory capacity in the range of 64 KB, 4 MB, 8 MB and even 16 MB (MB = Million bytes).

The following terms related to memory of a computer are discussed below:

1. **Random Access Memory (RAM):** The primary storage is referred to as random access memory (RAM) because it is possible to randomly select and use any location of the memory directly store and retrieve data. It takes same time to any address of the memory as the first address. It is also called read/write memory. The storage of data and instructions inside the primary storage is temporary. It disappears from RAM as soon as the power to the computer is switched off. The memories, which loose their content on failure of power supply, are known as **volatile** memories .So now we can say that RAM is volatile memory.

2. **Read Only Memory (ROM):** There is another memory in computer, which is called Read Only Memory (ROM). Again it is the ICs inside the PC that form the ROM. The storage of program and data in the ROM is permanent. The ROM stores some standard processing programs supplied by the manufacturers to operate the personal computer. The ROM can only be read by the CPU but it cannot be changed. The basic input/output program is stored in the ROM that examines and initializes various equipment attached to the PC when the switch is made ON. The memories, which do not loose their content on failure of power supply, are known as **non-volatile** memories. ROM is non-volatile memory.

3. **PROM** There is another type of primary memory in computer, which is called Programmable Read Only Memory (PROM). You know that it is not possible to modify or erase programs stored in ROM, but it is possible for you to store your program in PROM chip. Once the programmes are written it cannot be changed and remain intact even if power is switched off. Therefore programs or instructions written in PROM or ROM cannot be erased or changed.

4. **EPROM:** This stands for Erasable Programmable Read Only Memory, which over come the problem of PROM & ROM. EPROM chip can be programmed time and again by erasing the information stored earlier in it. Information stored in EPROM exposing the chip for some time ultraviolet light and it erases chip is reprogrammed using a special programming facility. When the EPROM is in use information can only be read.

5. **Cache Memory:** The speed of CPU is extremely high compared to the access time of main memory. Therefore the performance of CPU decreases due to the slow speed of main memory. To decrease the mismatch in operating speed, a small memory chip is attached between CPU and Main memory whose access time is very close to the processing speed of CPU. It is called CACHE memory. CACHE memories are accessed much faster than conventional RAM. It is used to store programs or data currently being executed or temporary data frequently used by the CPU. So each memory makes main memory to be faster and larger than it really is. It is also very expensive to have bigger size of cache memory and its size is normally kept small.

6. **Registers:** The CPU processes data and instructions with high speed, there is also movement of data between various units of computer. It is necessary to transfer the processed data with high speed. So the computer uses a number of special memory units called *registers.* They are not part of the main memory but they store data or information temporarily and pass it on as directed by the control unit.

9.1.4:SECONDARY STORAGE

1. You are now clear that the operating speed of primary memory or main memory should be as fast as possible to cope up with the CPU speed. These high-speed storage devices are very expensive and hence the cost per bit of storage is also very high. Again the storage capacity of the main memory is also very limited. Often it is necessary to store hundreds of millions of bytes of data for the CPU to process. Therefore additional memory is required in all the computer systems. This memory is called *auxiliary memory* or *secondary storage*.

2. In this type of memory the cost per bit of storage is low. However, the operating speed is slower than that of the primary storage. Huge volume of data are stored here on permanent basis and transferred to the primary storage as and when required. Most widely used secondary storage devices are *magnetic tapes* and *magnetic disk*.

3. **Magnetic Tape:** Magnetic tapes are used for large computers like mainframe computers where large volume of data is stored for a longer time. In PC also you can use tapes in the form of cassettes. The cost of storing data in tapes is inexpensive. Tapes consist of magnetic materials that store data permanently. It can be 12.5 mm to 25 mm wide plastic film-type and 500 meter to 1200 meter long which is coated with magnetic material. The deck is connected to the central processor and information is fed into or read from the tape through the processor. It similar to cassette tape recorder.

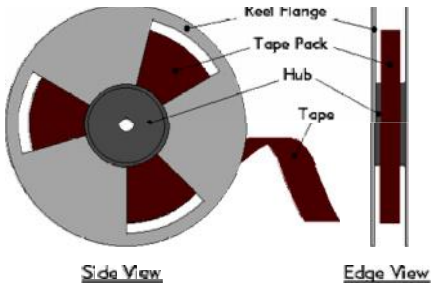


Fig: Magnetic Tape

1. **Advantages of Magnetic Tape:**

- **Compact:** A 10-inch diameter reel of tape is 2400 feet long and is able to hold 800, 1600 or 6250 characters in each inch of its length. The maximum capacity of such tape is 180 million characters. Thus data are stored much more compactly on tape.
- **Economical**: The cost of storing characters is very less as compared to other storage devices.
- **Fast**: Copying of data is easier and fast.
- **Long term Storage and Re-usability**: Magnetic tapes can be used for long term storage and a tape can be used repeatedly with out loss of data.

2. **Magnetic Disk:** You might have seen the gramophone record, which is circular like a disk and coated with magnetic material. Magnetic disks used in computer are made on the same principle. It rotates with very high speed inside the computer

drive. Data is stored on both the surface of the disk. Magnetic disks are most popular for *direct access* storage device. Each disk consists of a number of invisible *concentric circles* called *tracks*. Information is recorded on tracks of a disk surface in the form of tiny magnetic spots. The presence of a magnetic spot represents *one bit* and its absence represents zero bit. The information stored in a disk can be read many times without affecting the stored data. So the reading operation is non-destructive. But if you want to write a new data, then the existing data is erased from the disk and new data is recorded.

3. **Floppy Disk:** It is similar to magnetic disk discussed above. They are 5.25 inch or 3.5 inch in diameter. They come in single or double density and recorded on one or both surface of the diskette. The capacity of a 5.25-inch floppy is 1.2 mega bytes whereas for 3.5 inch floppy it is 1.44 mega bytes. It is cheaper than any other storage devices and is portable. The floppy is a low cost device particularly suitable for personal computer system. A floppy disk drive reads and writes data to a small, circular piece of metal-coated plastic similar to audio cassette tape.

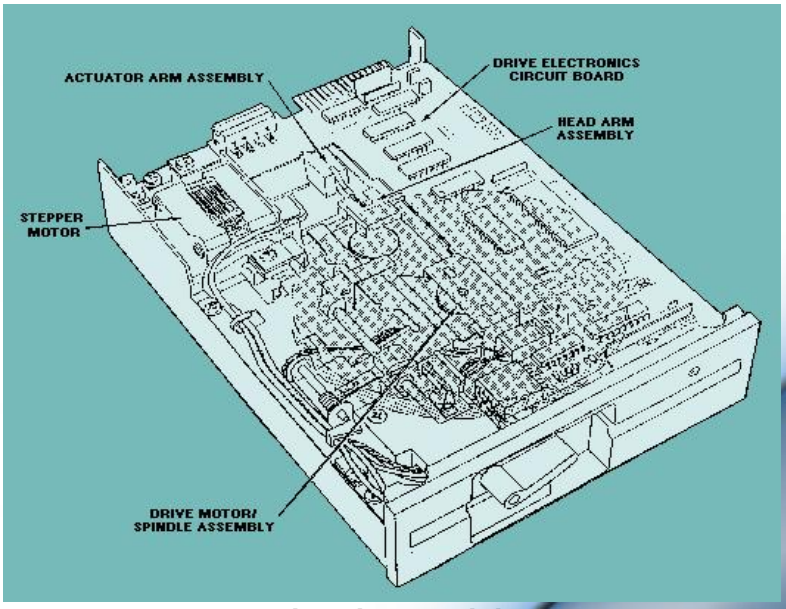


Fig: Floppy Disk

2. **Optical Disk:**

With every new application and software there is greater demand for memory capacity. It is the necessity to store large volume of data that has led to the development of optical disk storage medium. Optical disks can be divided into the following categories:

1. *Compact Disk/ Read Only Memory (CD-ROM)*: CD-ROM disks are made of reflective metals. CD-ROM is written during the process of manufacturing by high power *laser beam.* Here the storage density is very high, storage cost is very low and access time is relatively fast. Each disk is approximately 4 1/2 inches in diameter and can hold over 600 MB of data. As the CD-ROM can be *read only* we cannot write or make changes into the data contained in it.

2. *Write Once, Read Many (WORM)*: The inconvenience that we can not write any thing in to a CD-ROM is avoided in WORM. A WORM allows the user to write data permanently on to the disk. Once the data is written it can never be erased without physically damaging the disk. Here data can be recorded from keyboard, video scanner, OCR equipment and other devices. The advantage of WORM is that it can store vast amount of data amounting to gigabytes (10^9 bytes). Any document in a WORM can be accessed very fast, say less than 30 seconds.

3. *Erasable Optical Disk*: These are optical disks where data can be written, erased and re-written. This also applies a laser beam to write and re-write the data. These disks may be used as alternatives to traditional disks. Erasable optical disks are based on a technology known as *magnetic optical* (MO). To write a data bit on to the erasable optical disk the MO drive's laser beam heats a tiny, precisely defined point on the disk's surface and magnetises it.

9.1.5: INPUT OUTPUT DEVICES

A computer is only useful when it is able to communicate with the external environment. When you work with the computer you feed your data and instructions through some devices to the computer. These devices are called Input devices. Similarly computer after processing, gives output through other devices called output devices.

For a particular application one form of device is more desirable compared to others. We will discuss various types of I/O devices that are used for different types of applications. They are also known as peripheral devices because they surround the CPU and make a communication between computer and the outer world.

1 Input Devices

Input devices are necessary to convert our information or data in to a form which can be understood by the computer. A good input device should provide timely, accurate and useful data to the main memory of the computer for processing followings are the most useful input devices.

1. **Keyboard: -** This is the standard input device attached to all computers. The layout of keyboard is just like the traditional typewriter of the type QWERTY. It also contains some extra command keys and function keys. It contains a total of 101 to 104 keys. A typical keyboard used in a computer is shown in Fig.. You have to press correct combination of keys to input data. The computer can recognise the electrical signals corresponding to the correct key combination and processing is done accordingly.

Fig: Keyboard

2. **Mouse:** - Mouse is an input device shown in Fig. 2.7 that is used with your personal computer. It rolls on a small ball and has two or three buttons on the top. When you roll the mouse across a flat surface the screen censors the mouse in the direction of mouse movement. The cursor moves very fast with mouse giving you more freedom to work in any direction. It is easier and faster to move through a mouse.

Fig: Mouse

3. **Scanner:** The keyboard can input only text through keys provided in it. If we want to input a picture the keyboard cannot do that. Scanner is an optical device that can input any graphical matter and display it back. The common optical scanner devices are Magnetic Ink Character Recognition (MICR), Optical Mark Reader (OMR) and Optical Character Reader (OCR).

- **Magnetic Ink Character Recognition (MICR):** - This is widely used by banks to process large volumes of cheques and drafts. Cheques are put inside the MICR. As they enter the reading unit the cheques pass through the magnetic field which causes the read head to recognise the character of the cheques.

- **Optical Mark Reader (OMR):** This technique is used when students have appeared in objective type tests and they had to mark their answer by darkening a square or circular space by pencil. These answer sheets are directly fed to a computer for grading where OMR is used.

- **Optical Character Recognition (OCR):** - This technique unites the direct reading of any printed character. Suppose you have a set of hand written characters on a piece of paper. You put it inside the scanner of the computer. This pattern is compared with a site of patterns stored inside the computer. Whichever pattern is matched is called a character read. Patterns that cannot be identified are rejected. OCRs are expensive though better the MICR.

Output Devices

1. **Visual Display Unit:** The most popular input/output device is the Visual Display Unit (VDU). It is also called the monitor. A Keyboard is used to input data and Monitor is used to display the input data and to receive massages from the computer. A monitor has its own box which is separated from the main computer system and is connected to the computer by cable. In some systems it is compact with the system unit. It can be *color* or *monochrome*.

2. **Terminals:** It is a very popular interactive input-output unit. It can be divided into two types: hard copy terminals and *soft copy* terminals. A *hard copy* terminal provides a printout on paper whereas soft copy terminals provide visual copy on monitor. A terminal when connected to a CPU sends instructions directly to the computer. Terminals are also classified as dumb terminals or intelligent terminals depending upon the work situation.

3. **Printer:** It is an important output device which can be used to get a printed copy of the processed text or result on paper. There are different types of printers that are designed for different types of applications. Depending on their speed and approach of printing, printers are classified as *impact* and *non-impact* printers. Impact printers use the familiar typewriter approach of hammering a typeface against the paper and inked ribbon. *Dot-matrix printers* are of this type. Non-impact printers

do not hit or impact a ribbon to print. They use electro-static chemicals and ink-jet technologies. *Laser printers* and *Ink-jet printers* are of this type. This type of printers can produce color printing and elaborate graphics.

10

OPERATING SYSTEMS

Unit Structure

1. Objectives
2. Introduction
   10.1.1: Types of OS
3. Windows Operating System
4. Linux Operating System
5. Some Linux Commands
6. Questions
7. Further Reading

1. : OBJECTIVES

After completing this chapter, you will be able to:

Understand the Operating System and its structure.

Different types of Operating Systems, their applications, use & history.

Interact with LINUX OS using several commands

2. INTRODUCTION:

An **operating system** (**OS**) is a set of programs that manage computer hardware resources and provide common services for application software. The operating system is the most important type of system software in a computer system. A user cannot run an application program on the computer without an operating system, unless the application program is self booting.
Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between application programs and the computer hardware, although the application code is usually executed directly by the hardware and will frequently call the OS or be interrupted by

it. Operating systems are found on almost any device that contains a computer — from cellular phones and video game consoles to supercomputers and web servers.

Examples of popular modern operating systems include Android, iOS, Linux, Mac OS X, all of which have their roots in Unix, and Microsoft Windows.

10.1.1: Types of OS:

Real-time

A real-time operating system is a multitasking operating system that aims at executing real-time applications. Real-time operating systems often use specialized scheduling algorithms so that they can achieve a deterministic nature of behavior. The main objective of real-time operating systems is their quick and predictable response to events. They have an event-driven or time-sharing design and often aspects of both. An event-driven system switches between tasks based on their priorities or external events while time-sharing operating systems switch tasks based on clock interrupts.

Multi-user vs. Single-user

A multi-user operating system allows multiple users to access a computer system concurrently. Time-sharing system can be classified as multi-user systems as they enable a multiple user access to a computer through the sharing of time. Single-user operating systems, as opposed to a multi-user operating system, are usable by a single user at a time. Being able to have multiple accounts on a Windows operating system does not make it a multi-user system. Rather, only the network administrator is the real user. But for a Unix-like operating system, it is possible for two users to login at a time and this capability of the OS makes it a multi-user operating system.

Multi-tasking vs. Single-tasking

When only a single program is allowed to run at a time, the system is grouped under a single-tasking system. However, when the operating system allows the execution of multiple tasks at one time, it is classified as a multi-tasking operating system. Multi-tasking can be of two types: pre-emptive or co-operative. In pre-emptive multitasking, the operating system slices the CPU time and dedicates one slot to each of the programs. Unix-like operating systems such as Solaris and Linux support pre-emptive multitasking, as does AmigaOS. Cooperative multitasking is achieved by relying on each process to give time to the other processes in a defined manner. MS Windows prior to Windows 2000 and Mac OS prior to OS X used to support cooperative multitasking.

Distributed

A distributed operating system manages a group of independent computers and makes them appear to be a single computer. The development of networked computers that could be linked and communicate with each other gave rise to distributed computing. Distributed computations are carried out on more than one machine. When computers in a group work in cooperation, they make a distributed system.

Embedded

Embedded operating systems are designed to be used in embedded computer systems. They are designed to operate on small machines like PDAs with less autonomy. They are able to operate with a limited number of resources. They are very compact and extremely efficient by design. Windows CE and Minix 3 are some examples of embedded operating systems.

10.2 WINDOWS OPERATING SYSTEM:

Introduction to Windows 7:

Windows 7 is an **operating system**, developed by the global giant Microsoft, which was released for public in October 2009. An **operating system** can be understood as a software program designed to facilitate the communication between computer hardware and software. Without an operating system, a computer is useless. **Windows 7** is more simpler and easier to use compared to its predecessor, Windows Vista. Windows Vista cannot be considered as a very successful launch of Microsoft. Windows 7 has a 64-bit along with the availability of 32-bit support which enables the users to use almost all the latest PCs. Be it desktops, laptops, notebooks, or anything,

Windows 7 supports them all.

The purpose behind the launching of Windows 7:

Windows 7 was launched with many new and advanced features beneficial to the users. The main aim was to cope up with the limitations present in the previous versions that were highly criticised. Windows Vista consisted of a big range of user friendly features but unfortunately, the system failed due to ever increasing complaints and negative reviews coming from the press. Contrary to this, Windows 7 was developed with a focus on rectifying the mistakes and providing noticeable upgrade to the product line of Windows.

This product of Microsoft has launched **six different editions**. They are listed as under:

- Starter
- Home Premium
- Professional
- Ultimate
- OEM
- Enterprise

System requirements for installing Windows 7:

The installation of Windows 7 operating system needs certain bare minimum requirements.

- RAM (Random Access Memory) ranging from 1GB to 2GB is enough here.
- Free Hard Disk space ranging from 16GB to 20GB is needed.
- DirectX 9 graphics device with WDDM 1.0 or higher drive is sufficient in both cases.

Various available options for the installation of Windows 7:

Microsoft has taken all the necessary steps to ensure that this version is developed in a manner that it is accepted by all. Hence, keeping this in mind, its installation process is made extremely simple.

Users wanting to go for this product, have the option of just upgrading directly from Windows Vista or even XP. Secondly they can opt for a clean install option available to them. And for the fresh users, they can directly buy new computers having this operating system already installed.

Most Striking features of Windows 7:

The most attractive feature of Windows 7 can be its stability and reliability. It's advanced graphical features, Aeropeek, and new taskbar, which is covered by **Windows Shell**. This taskbar has been given the name '**Superbar**' by its developers. This new Taskbar has been praised by the users so far. Although a bit difficult to get a grasp of, these new features have proved to be a hit among the users. The best improvement by Microsoft here is the replacement of Quick Launch Toolbar with pinning applications. This innovation has resulted in more easy to see icons. They are an absolute delight to use.

Another interesting feature added here is that of **Aero Snap**. This feature enables the user to maximize the window as soon as the edge of the screen is dragged. Just a little move by the user, and the window restores its previous state.

Users having the leisure of touch screen monitors, can also enjoy another interesting feature of **native touch**.

The developers have taken full care to see that performance of Windows 7 is much better compared to vista and this can be proved with the introduction of innumerable new features including the following: support for virtual hard disks, Handwriting recognition, improved presentation on multi-core processors, improved boot performance, DirectAccess and kernel improvements, etc.

Windows components like **Internet Explorer** and **Windows Media Player 12** are included in this product of Microsoft. The new look that the developers have given to Windows Media Player here is uncomparable. It is more stylish and sleek than ever, making it more enjoyable.

For Game lovers also, windows 7 has proved to be better than its previous edition, Windows Vista. Many popular games like Internet Spades, Internet Backgammon, etc. which had disappeared from Vista have been restored in Windows 7.

By the lauch of **Windows 7**, Microsoft has successfully tried to regain its lost popularity to a considerable extent.

10.3 LINUX OPERATING SYSTEM:

Linux is, in simplest terms, an operating system. It is the software on a computer that enables applications and the computer operator to access the devices on the computer to perform desired functions. The operating system (OS) relays instructions from an application to, for instance, the computer's processor. The processor performs the instructed task, then sends the results back to the application via the operating system.

Explained in these terms, Linux is very similar to other operating systems, such as Windows and OS X.

But something sets Linux apart from these operating systems. The Linux operating system represented a $25 billion ecosystem in 2008. Since its inception in 1991, Linux has grown to become a force in computing, powering everything from the New York Stock Exchange to mobile phones to supercomputers to consumer devices.

As an open operating system, Linux is developed collaboratively, meaning no one company is solely responsible for its development or ongoing support. Companies participating in the

Linux economy share research and development costs with their partners and competitors. This spreading of development burden amongst individuals and companies has resulted in a large and efficient ecosystem and unheralded software innovation.

Over 1,000 developers, from at least 100 different companies, contribute to every kernel release. In the past two years alone, over 3,200 developers from 200 companies have contributed to the kernel--which is just one small piece of a Linux distribution.

This article will explore the various components of the Linux operating system, how they are created and work together, the communities of Linux, and Linux's incredible impact on the IT ecosystem.

Where is Linux?

One of the most noted properties of Linux is where it can be used. Windows and OS X are predominantly found on personal computing devices such as desktop and laptop computers. Other operating systems, such as Symbian, are found on small devices such as phones and PDAs, while mainframes and supercomputers found in major academic and corporate labs use specialized operating systems such as AS/400 and the Cray OS.

Linux, which began its existence as a server OS and Has become useful as a desktop OS, can also be used on all of these devices. ‚ÄúFrom wristwatches to supercomputers,,Äù is the popular description of Linux' capabilities.

An abbreviated list of some of the popular electronic devices Linux is used on today includes:

Dell Inspiron Mini 9 and 12

Garmin Nuvi 860, 880, and 5000

Google Android Dev Phone 1

HP Mini 1000

Lenovo IdeaPad S9

Motorola MotoRokr EM35 Phone

One Laptop Per Child XO2

Sony Bravia Television

Sony Reader

TiVo Digital Video Recorder

Volvo In-Car Navigation System

Yamaha Motif Keyboard

These are just the most recent examples of Linux-based devices available to consumers worldwide. This actual number of items that use Linux numbers in the thousands. The Linux Foundation is building a centralized database that will list all currently offered Linux-based products, as well as archive those devices that pioneered Linux-based electronics.

Linux Distributions:

Well-known Linux distributions include:

- Arch Linux, a minimalist distribution maintained by a volunteer community and primarily based on binary packages in the tar.gz and tar.xz format.
- Debian, a non-commercial distribution maintained by a volunteer developer community with a strong commitment to free software principles

- Knoppix, the first Live CD distribution to run completely from removable media without installation to a hard disk, derived from Debian
- Linux Mint Debian Edition (LMDE) is based directly on Debian's *testing* distribution.
- Ubuntu, a popular desktop and server distribution derived from Debian, maintained by Canonical Ltd.
- BackTrack, based off the Ubuntu Operating System. Used for digital forensics and penetration testing.
- Kubuntu, the KDE version of Ubuntu.
- Linux Mint, a distribution based on and compatible with Ubuntu.
- Xubuntu is the Xfce version of Ubuntu.
- Fedora, a community distribution sponsored by Red Hat
- Red Hat Enterprise Linux, which is a derivative of Fedora, maintained and commercially supported by Red Hat.
- CentOS, a distribution derived from the same sources used by Red Hat, maintained by a dedicated volunteer community of developers with both 100% Red Hat-compatible versions and an upgraded version that is not always 100% upstream compatible
- Oracle Enterprise Linux, which is a derivative of Red Hat Enterprise Linux, maintained and commercially supported by Oracle.
- Mandriva, a Red Hat derivative popular in France and Brazil, today maintained by the French company of the same name.
- Manthiran Linux is a popular linux distro introduced to this world by Quara Foundation.
- PCLinuxOS, a derivative of Mandriva, grew from a group of packages into a community-spawned desktop distribution.
- Gentoo, a distribution targeted at power users, known for its FreeBSD Ports-like automated system for compiling applications from source code
- openSUSE a community distribution mainly sponsored by Novell.
- SUSE Linux Enterprise, derived from openSUSE, maintained and commercially supported by Novell.
- Slackware, one of the first Linux distributions, founded in 1993, and since then actively maintained by Patrick J. Volkerding.
- Damn Small Linux, "DSL" is a Biz-card Desktop OS

Advantages of Linux Operating System:

Low cost:

There is no need to spend time and huge amount money to obtain licenses since Linux and much of it's software come with the GNU General Public License. There is no need to worry about any software's that you use in Linux.

Stability:

Linux has high stability compared with other operating systems. There is no need to reboot the Linux system to maintain performance levels. Rarely it freeze up or slow down. It has a continuous up-times of hundreds of days or more.

Performance:

Linux provides high performance on various networks. It has the ability to handle large numbers of users simultaneously.

Networking:

Linux provides a strong support for network functionality; client and server systems can be easily set up on any computer running Linux. It can perform tasks like network backup more faster than other operating systems.

Flexibility:

Linux is very flexible. Linux can be used for high performance server applications, desktop applications, and embedded systems. You can install only the needed components for a particular use. You can also restrict the use of specific computers.

Compatibility:

It runs all common Unix software packages and can process all common file formats.

Wider Choice:

There is a large number of Linux distributions which gives you a wider choice. Each organization develop and support different distribution. You can pick the one you like best; the core function's are the same.

Fast and easy installation:

Linux distributions come with user-friendly installation.

Better use of hard disk:

Linux uses its resources well enough even when the hard disk is almost full.

Multitasking:

Linux is a multitasking operating system. It can handle many things at the same time.

Security:

Linux is one of the most secure operating systems. File ownership and permissions make linux more secure.

Open source:

Linux is an Open source operating systems. You can easily get the source code for linux and edit it to develop your personal operating system.

Today, Linux is widely used for both basic home and office uses. It is the main operating system used for high performance business and in web servers. Linux has made a high impact in this world.

Comparison of Windows and Linux:

Both Linux and Windows are operating systems. An operating system is the most important program that runs on a computer. Every general-purpose computer must have an operating system to run other programs. Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the display screen, keeping track of files and directories on the disk, and controlling peripheral devices such as disk drives and printers.

1. Reduces the risk of carpal tunnel syndrome

When linux is properly installed, there no longer a need to use the mouse. Chances of you using a mouse is close to zero.

2. Use the extra cash for rewards

Linux is 100% free while Windows Vista Ultimate costs $398.99 at the time of writing. Companies that pay a licensing annually could have used the money for other things like buying an additional server to reduce the load or even give a bigger bonus to its loyal employees.

3. Formats are free, freedom is preserved

Linux file formats can be accessed in a variety of ways because they are free. Windows on the other hand makes you lock your own data in secret formats that can only be accessed with tools leased to you at the vendor's price. "What we will get with Microsoft is a three-year lease on a health record we need to keep for 100 years"

4. Zero risk in violating license agreements

Linux is open source so you are unlikely to violate any license agreement. All the software is happily yours. With MS Windows you likely already violate all kinds of licenses and you could be pronounced a computer pirate if only a smart lawyer was after you. The worldwide PC software piracy rate for 2004 is at 35%. Which means that 3 out of 10 people are likely to get into real trouble.

5. Transparent vs Proprietary

MS Windows is based on DOS, Linux is based on UNIX. MS Windows Graphical User Interface (GUI) is based on Microsoft-own marketing-driven specifications. Linux GUI is based on industry-standard network-transparent X-Windows.

6. Better network, processing capabilities

Linux beats Windows hands down on network features, as a development platform, in data processing capabilities, and as a scientific workstation. MS Windows desktop has a more polished appearance, simple general business applications, and many more games for kids (less intellectual games compared to linux's).

7. Customizable

Linux is customizable in a way that Windows is not. For example,NASlite is a version of Linux that runs off a single floppy disk and converts an old computer into a file server. This ultra small edition of Linux is capable of networking, file sharing and being a web server.

8. Flexibility

Windows must boot from a primary partition. Linux can boot from either a primary partition or a logical partition inside an extended partition. Windows must boot from the first hard disk. Linux can boot from any hard disk in the computer.

9. Mobility

Windows allows programs to store user information (files and settings) anywhere. This makes it impossibly hard to backup user data files and settings and to switch to a new computer. In contrast, Linux stores all user data in the home directory making it much easier to migrate from an old computer to a new one. If home directories are segregated in their own partition, you can even upgrade from one version of Linux to another without having to migrate user data and settings.

10. Proven Security

Why isn't Linux affected by viruses? Simply because its code has been open source for more than a decade, tested by people all around the world, and not by a single development team like in the case of Windows. This leads to a lightning fast finding and fixing for exploitable holes in Linux. So that proves Linux as having an extremely enhanced security and lesser chances of exploits compared to Windows.

10.4 SOME LINUX COMMANDS:

mkdir - make directories

Usage

mkdir [OPTION] DIRECTORY

Options

Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

-m, mode=MODE set permission mode (as in chmod), not rwxrwxrwx - umask

-p, parents no error if existing, make parent directories as needed

-v, verbose print a message for each created directory

-help display this help and exit

-version output version information and exit

cd - change directories

Use cd to change directories. Type cd followed by the name of a directory to access that directory.Keep in mind that you are always in a directory and can navigate to directories hierarchically above or below.

mv- change the name of a directory

Type mv followed by the current name of a directory and the new name of the directory.

Ex: mv testdir newnamedir

pwd - print working directory

will show you the full path to the directory you are currently in. This is very handy to use, especially when performing some of the other commands on this page

rmdir - Remove an existing directory

rm -r

Removes directories and files within the directories recursively.

chown - change file owner and group

Usage

chown [OPTION] OWNER[:[GROUP]] FILE

chown [OPTION] :GROUP FILE

chown [OPTION] --reference=RFILE FILE

Options

Change the owner and/or group of each FILE to OWNER and/or GROUP. With --reference, change the owner and group of each FILE to those of RFILE.

-c, changes like verbose but report only when a change is made

-dereference affect the referent of each symbolic link, rather than the symbolic link itself

-h, no-dereference affect each symbolic link instead of any referenced file (useful only on systems that can change the ownership of a symlink)

-from=CURRENT_OWNER:CURRENT_GROUP

change the owner and/or group of each file only if its current owner and/or group match those specified here. Either may be omitted, in which case a match is not required for the omitted attribute.

-no-preserve-root do not treat '/' specially (the default)

-preserve-root fail to operate recursively on '/'

-f, -silent, -quiet suppress most error messages

-reference=RFILE use RFILE's owner and group rather than the specifying OWNER:GROUP values

-R, -recursive operate on files and directories recursively

-v, -verbose output a diagnostic for every file processed

The following options modify how a hierarchy is traversed when the -R option is also specified. If more than one is specified, only the final one takes effect.

-H if a command line argument is a symbolic link to a directory, traverse it

-L traverse every symbolic link to a directory encountered

-P do not traverse any symbolic links (default)

chmod - change file access permissions

Usage

chmod [-r] permissions filenames

r Change the permission on files that are in the subdirectories of the directory that you are currently in permission Specifies the rights that are being granted. Below is the different rights that you can grant in an alpha numeric format.filenames File or directory that you are associating the rights with Permissions

u - User who owns the file.

g - Group that owns the file.

o - Other.

a - All.

r - Read the file.

w - Write or edit the file.

x - Execute or run the file as a program.

Numeric Permissions:

CHMOD can also to attributed by using Numeric Permissions:

400 read by owner

040 read by group

004 read by anybody (other)

200 write by owner

020 write by group

002 write by anybody

100 execute by owner

010 execute by group

001 execute by anybody

ls - Short listing of directory contents

-a list hidden files

-d list the name of the current directory

-F show directories with a trailing '/'
executable files with a trailing '*'

-g show group ownership of file in long listing

-i print the inode number of each file

-l long listing giving details about files and directories

-R list all subdirectories encountered

-t sort by time modified instead of name

cp - Copy files

cp myfile yourfile

Copy the files "myfile" to the file "yourfile" in the current working directory. This command will create the file "yourfile" if it doesn't exist. It will normally overwrite it without warning if it exists.

cp -i myfile yourfile

With the "-i" option, if the file "yourfile" exists, you will be prompted before it is overwritten.

cp -i /data/myfile

Copy the file "/data/myfile" to the current working directory and name it "myfile". Prompt before overwriting the file.

cp -dpr srcdir destdir

Copy all files from the directory "srcdir" to the directory "destdir" preserving links (-poption), file attributes (-p option), and copy recursively (-r option). With these options, a directory and all it contents can be copied to another dir

ln - Creates a symbolic link to a file.

ln -s test symlink

Creates a symbolic link named symlink that points to the file test Typing "ls -i test symlink" will show the two files are different with different inodes. Typing "ls -l test symlink" will show that symlink points to the file test.

locate - A fast database driven file locator.

slocate -u

This command builds the slocate database. It will take several minutes to complete this command.This command must be used before searching for files, however cron runs this command periodically on most systems.locate whereis Lists all files whose names contain the string "whereis". directory.

more - Allows file contents or piped output to be sent to the screen one page at a time

less - Opposite of the more command

cat - Sends file contents to standard output. This is a way to list the contents of short files to the screen. It works well with piping.

whereis - Report all known instances of a command

wc - Print byte, word, and line counts

bg

bg jobs Places the current job (or, by using the alternative form, the specified jobs) in the background, suspending its execution so that a new user prompt appears immediately. Use the jobs command to discover the identities of background jobs.

cal month year - Prints a calendar for the specified month of the specified year.

cat files - Prints the contents of the specified files.

clear - Clears the terminal screen.

cmp file1 file2 - Compares two files, reporting all discrepancies. Similar to the diff command, though the output format differs.

diff file1 file2 - Compares two files, reporting all discrepancies. Similar to the cmp command, though the output format differs.

dmesg - Prints the messages resulting from the most recent system boot.

fg

fg jobs - Brings the current job (or the specified jobs) to the foreground.

file files - Determines and prints a description of the type of each specified file.

find path -name pattern -print

Searches the specified path for files with names matching the specified pattern (usually enclosed in single quotes) and prints their names. The findcommand has many other arguments and functions; see the online documentation.

finger users - Prints descriptions of the specified users.

free - Displays the amount of used and free system memory.

ftp hostname

Opens an FTP connection to the specified host, allowing files to be transferred. The FTP program provides subcommands for accomplishing file transfers; see the online documentation.

head files - Prints the first several lines of each specified file.

ispell files - Checks the spelling of the contents of the specified files.

kill process_ids

kill - signal process_ids

kill -l

Kills the specified processes, sends the specified processes the specified signal (given as a number or name), or prints a list of available signals.

killall program

killall - signal program

Kills all processes that are instances of the specified program or sends the specified signal to all processes that are instances of the specified program.

mail - Launches a simple mail client that permits sending and receiving email messages.

man title

man section title - Prints the specified man page.

ping host - Sends an echo request via TCP/IP to the specified host. A response confirms that the host is operational.

reboot - Reboots the system (requires root privileges).

shutdown minutes

shutdown -r minutes

Shuts down the system after the specified number of minutes elapses (requires root privileges). The -r option causes the system to be rebooted once it has shut down.

sleep time - Causes the command interpreter to pause for the specified number of seconds.

sort files - Sorts the specified files. The command has many useful arguments; see the online documentation.

split file - Splits a file into several smaller files. The command has many arguments; see the online documentation

sync - Completes all pending input/output operations (requires root privileges).

telnet host - Opens a login session on the specified host.

top - Prints a display of system processes that's continually updated until the user presses the q key.

traceroute host - Uses echo requests to determine and print a network path to the host.

uptime - Prints the system uptime.

w - Prints the current system users.

wall - Prints a message to each user except those who've disabled message reception. Type **Ctrl-D** to end the message

.

Thank You