# ANALOG AND DIGITAL ELECTRONICS

Course Code: AECB05
Regulation: IARE-18

Prepared by

Ms M. Saritha, Assistant Professor, ECE

INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)
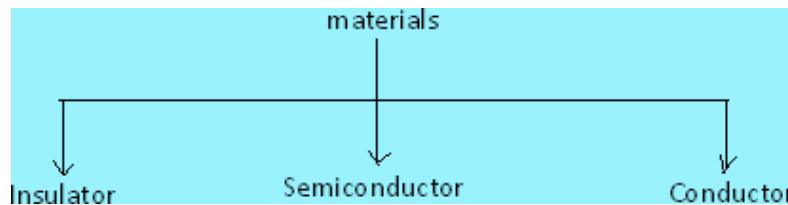
Dundigal, Hyderabad -500043

PN JUNCTION DIODE

INTRODUCTON

Based on the electrical conductivity all the materials in nature are classified as insulators, semiconductors, and conductors.

Insulator: An insulator is a material that offers a very low level (or negligible) of conductivity when voltage is applied. Eg: Paper, Mica, glass, quartz. Typical resistivity level of an insulator is of the order of 10^{10} to 10^{12} Ω-cm. The energy band structure of an insulator is shown in the fig.1.1. Band structure of a material defines the band of energy levels that an electron can occupy. Valance band is the range of electron energy where the electron remain bended too the atom and do not contribute to the electric current. Conduction bend is the range of electron energies higher than valance band where electrons are free to accelerate under the influence of external voltage source resulting in the flow of charge.

The energy band between the valance band and conduction band is called as forbidden band gap. It is the energy required by an electron to move from balance band to conduction band i.e. the energy required for a valance electron to become a free electron.

$$1 \text{ eV} = 1.6 \times 10^{-19} \text{ J}$$

For an insulator, as shown in the fig.1.1 there is a large forbidden band gap of greater than 5Ev. Because of this large gap there a very few electrons in the CB and hence the conductivity of insulator is poor. Even an increase in temperature or applied electric field is insufficient to transfer electrons from VB TO CB.

Insulator   Semiconductor   Conductor
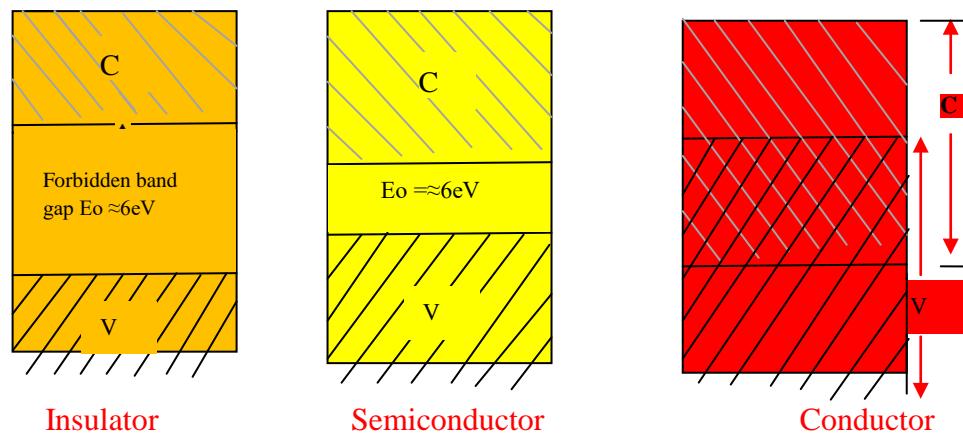
FiG:1.1 Energy band diagrams insulator, semiconductor and conductor

Conductors: A conductor is a material which supports a generous flow of charge when a voltage is applied across its terminals. i.e. it has very high conductivity. Eg: Copper, Aluminum, Silver, Gold. The resistivity of a conductor is in the order of 10^{-4} and 10^{-6} Ω-cm. The Valance and conduction bands overlap (fig1.1) and there is no energy gap for the electrons to move from valance band to conduction band. This implies that there are free electrons in CB even at absolute zero temperature (0K). Therefore at room temperature when electric field is applied large current flows through the conductor.
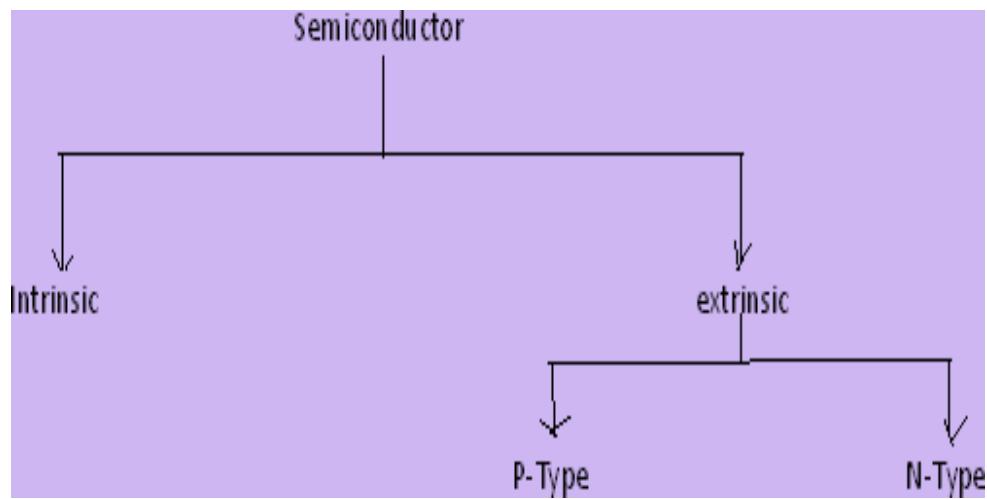
Semiconductor: A semiconductor is a material that has its conductivity somewhere between the insulator and conductor. The resistivity level is in the range of 10 and 10^4 Ω-cm. Two of the most commonly used are Silicon (Si=14 atomic no.) and germanium (Ge=32 atomic no.). Both have 4 valance electrons. The forbidden band gap is in the order of 1eV. For eg., the band gap energy for Si, Ge and GaAs is 1.21, 0.785 and 1.42 eV, respectively at absolute zero temperature (0K). At 0K and at low temperatures, the valance band electrons do not have sufficient energy to move from V to CB. Thus semiconductors act a insulators at 0K. as the temperature increases, a large number of valance electrons acquire sufficient energy to leave the VB, cross the forbidden bandgap and reach CB. These are now free electrons as they can move freely under the influence of electric field. At room temperature there are sufficient electrons in the CB and hence the semiconductor is capable of conducting some current at room temperature.

Inversely related to the conductivity of a material is its resistance to the flow of charge or current. Typical resistivity values for various materials' are given as follows.

Insulator	Semiconductor	Conductor
10^{-6} Ω-cm (Cu)	50Ω-cm (Ge)	10^{12} Ω-cm (mica)
	50×10^3 Ω-cm (Si)	

Typical resistivity values

Semiconductor Types

A pure form of semiconductors is called as intrinsic semiconductor. Conduction in intrinsic sc is either due to thermal excitation or crystal defects. Si and Ge are the two most important semiconductors used. Other examples include Gallium arsenide GaAs, Indium Antimonide (InSb) etc.Let us consider the structure of Si. A Si atomic no. is 14 and it has 4 valance electrons. These 4 electrons are shared by four neighboring atoms in the crystal structure by means of covalent bond. Fig. 1.2a shows the crystal structure of Si at absolute zero temperature (0K). Hence a pure SC acts has poor conductivity (due to lack of free electrons) at low or absolute zero temperature.

At room temperature some of the covalent bonds break up to thermal energy. The valance electrons that jump into conduction band are called as free electrons that are available for conduction.The absence of electrons in covalent bond is represented by a small circle usually referred to as hole which is of positive charge. Even a hole serves as carrier of electricity in a manner similar to that of free electron.
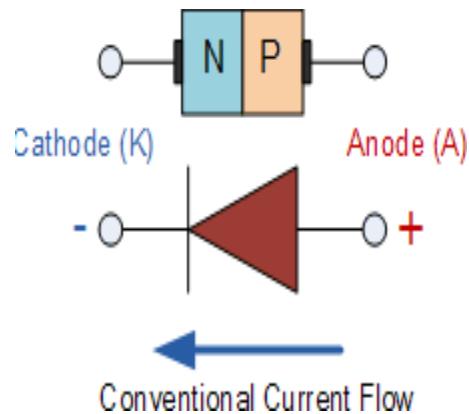
When a bond is in complete so that a hole exists, it is relatively easy for a valance electron in the neighboring atom to leave its covalent bond to fill this hole. An electron moving from a bond to fill a hole moves in a direction opposite to that of the electron. This hole, in its new position may now be filled by an electron from another covalent bond and the hole will correspondingly move one more step in the direction opposite to the motion of electron. Here we have a mechanism for conduction of electricity which does not involve free electrons.

The absence of electrons in covalent bond is represented by a small circle usually referred to as hole which is of positive charge. Even a hole serves as carrier of electricity in a manner similar to that of free electron.

Quantitative Theory Of Pn Junction Diode:
Pn Junction With No Applied Voltage Or Open Circuit Condition:
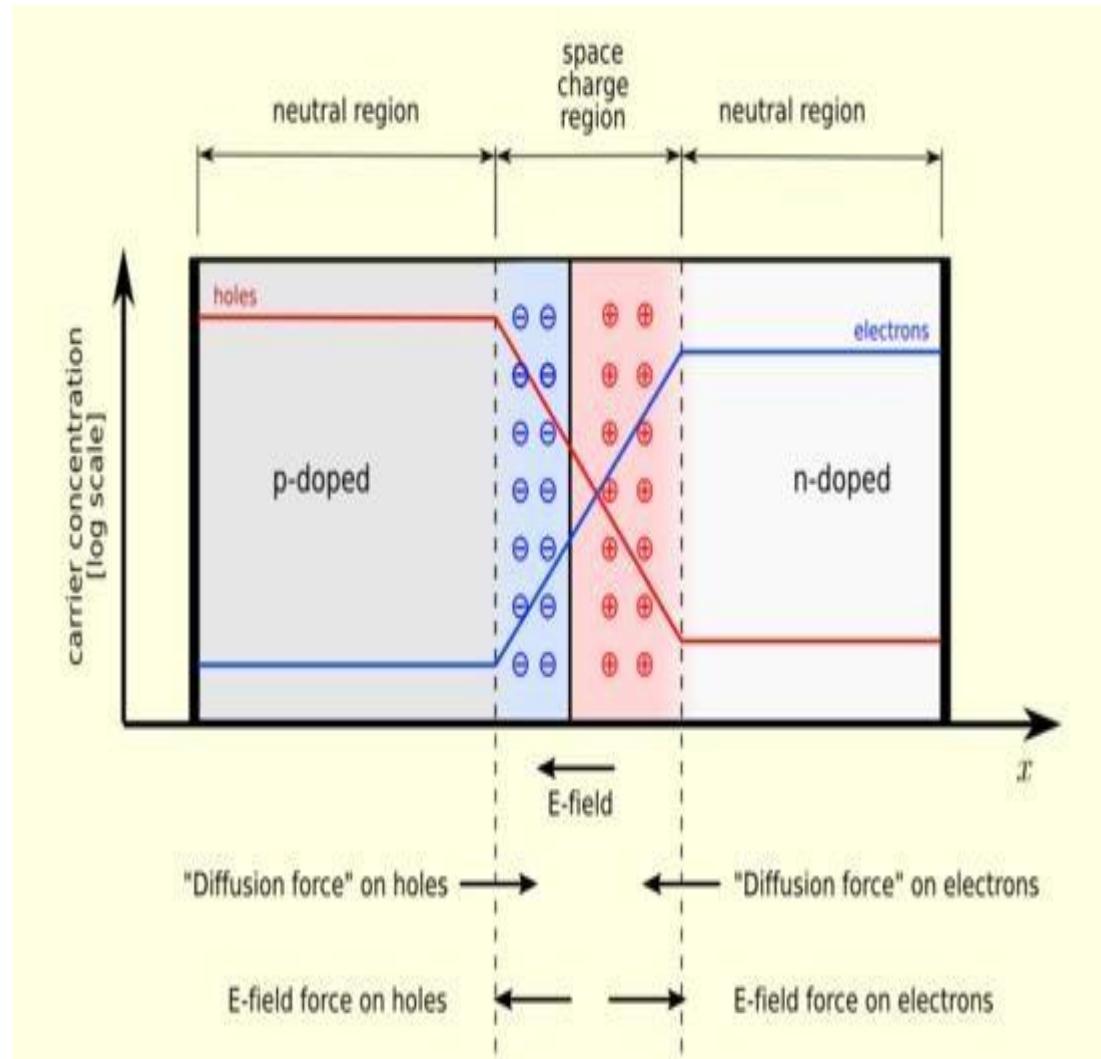
In a piece of sc, if one half is doped by p type impurity and the other half is doped by n type impurity, a PN junction is formed. The plane dividing the two halves or zones is called PN junction. As shown in the fig the n type material has high concentration of free electrons, while p type material has high concentration of holes. Therefore at the junction there is a tendency of free electrons to diffuse over to the P side and the holes to the N side. This process is called diffusion. As the free electrons move across the junction from N type to P type, the donor atoms become positively charged. Hence a positive charge is built on the N-side of the junction. The free electrons that cross the junction uncover the negative acceptor ions by filing the holes. Therefore a negative charge is developed on the p –side of the junction..This net negative charge on the p side prevents further diffusion of electrons into the p side. Similarly the net positive charge on the N side repels the hole crossing from p side to N side. Thus a barrier sis set up near the junction which prevents the further movement of charge carriers i.e. electrons and holes. As a consequence of induced electric field across the depletion layer, an electrostatic potential difference is established between P and N regions, which are called the potential barrier, junction barrier, diffusion potential or contact potential, Vo. The magnitude of the contact potential Vo varies with doping levels and temperature. Vo is 0.3V for Ge and 0.72 V for Si.
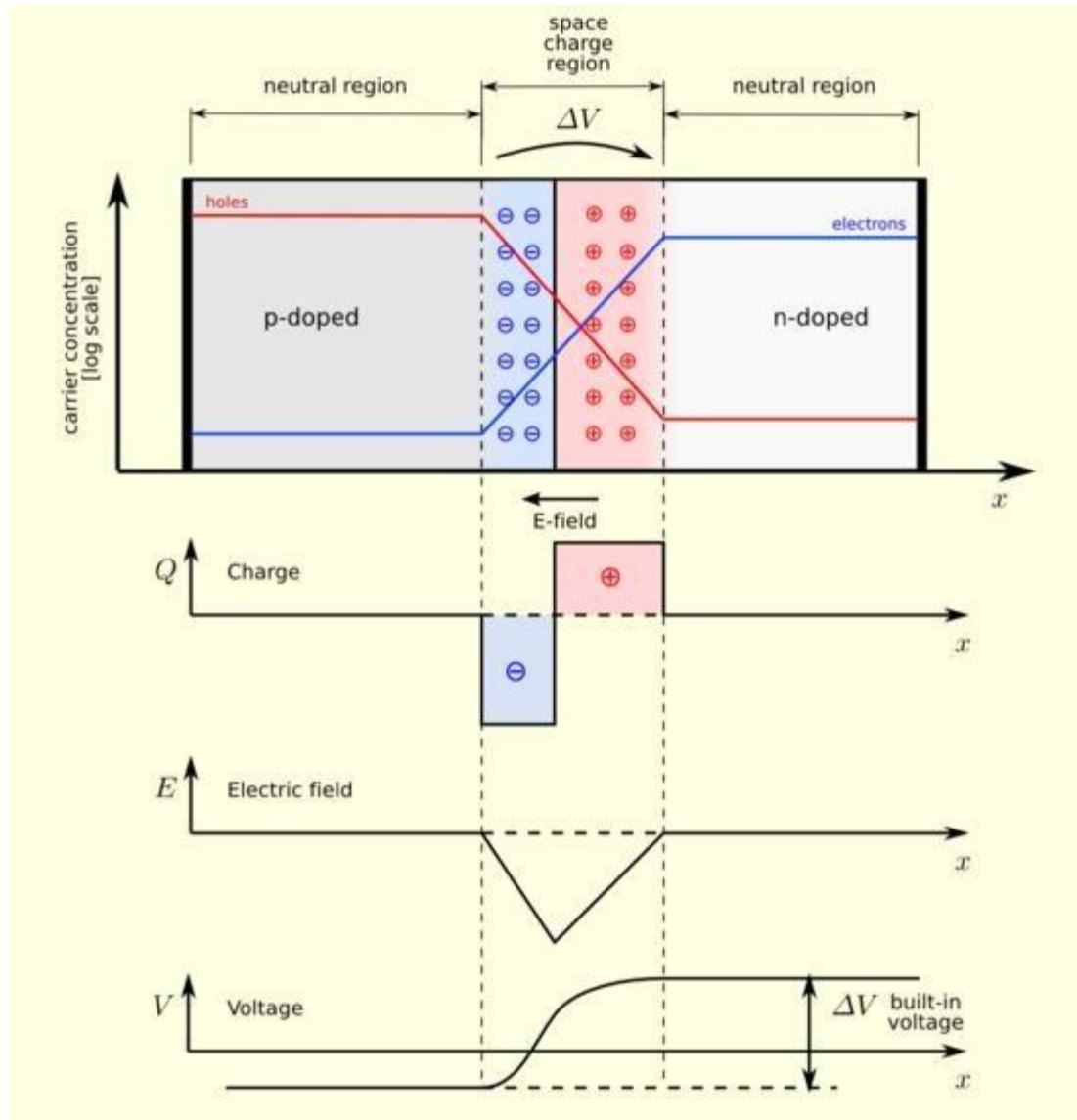

Symbol of PN Junction Diode

The electrostatic field across the junction caused by the positively charged N-Type region tends to drive the holes away from the junction and negatively charged p type regions tend to drive the electrons away from the junction. The majority holes diffusing out of the P region leave behind negatively charged acceptor atoms bound to the lattice, thus exposing a negatives pace charge in a previously neutral region. Similarly electrons diffusing from the N region expose positively ionized donor atoms and a double space charge builds up at the junction.

It is noticed that the space charge layers are of opposite sign to the majority carriers diffusing into them, which tends to reduce the diffusion rate. Thus the double space of the layer causes an electric field to be set up across the junction directed from N to P regions, which is in such a direction to inhibit the diffusion of majority electrons and holes as illustrated in fig 1.7b. The shape of the charge density, ρ, depends upon how diode id doped. Thus the junction region is depleted of mobile charge carriers. Hence it is called depletion layer, space region, and transition region. The depletion region is of the order of 0.5µm thick. There are no mobile carriers in this narrow depletion region. Hence no current flows across the junction and the system is in equilibrium. To the left of this depletion layer, the carrier concentration is p= N_A and to its right it is n= N_D.
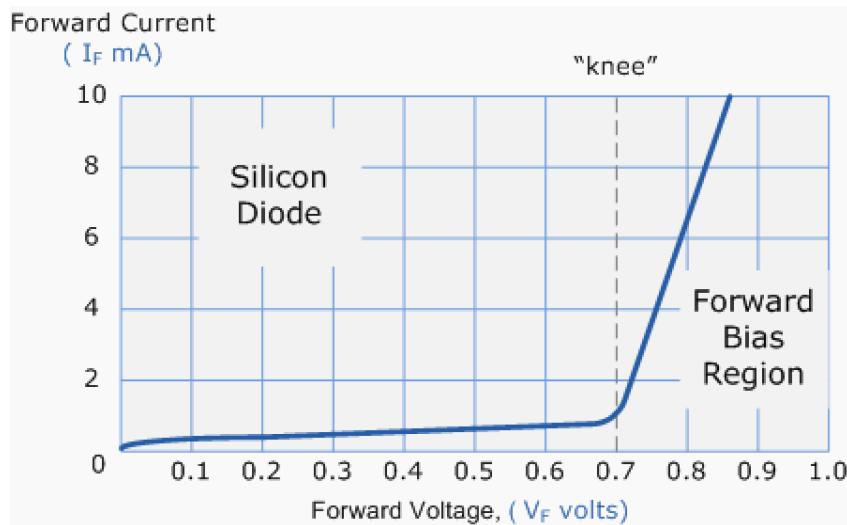
Forward Biased Junction Diode

When a diode is connected in a **Forward Bias** condition, a negative voltage is applied to the N- type material and a positive voltage is applied to the P-type material. If this external voltage becomes greater than the value of the potential barrier, approx. 0.7 volts for silicon and 0.3 volts for germanium, the potential barriers opposition will be overcome and current will start to flow. This is because the negative voltage pushes or repels electrons towards the junction giving them the energy to cross over and combine with the holes being pushed in the opposite direction towards the junction by the positive voltage. This results in a characteristics curve of zero current flowing up to this voltage point,

called the "knee" on the static curves and then a high current flow through the diode with little increase in the external voltage as shown below.
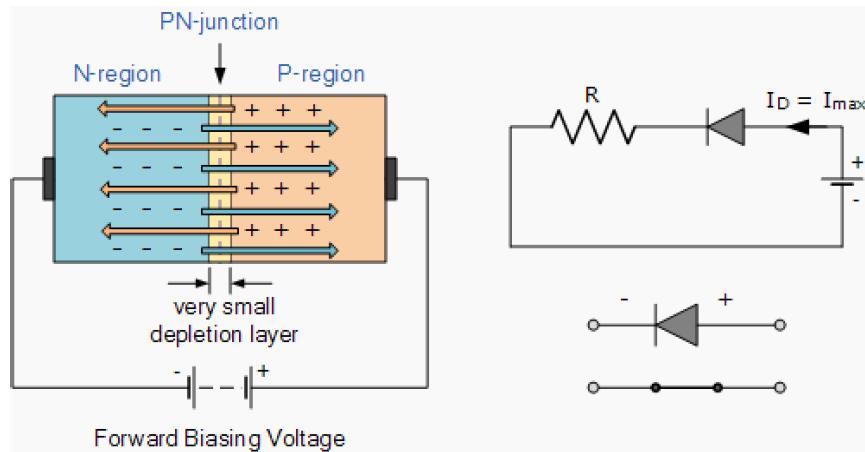
Diode Forward Characteristics

The application of a forward biasing voltage on the junction diode results in the depletion layer becoming very thin and narrow which represents a low impedance path through the junction thereby allowing high currents to flow. The point at which this sudden increase in current takes place is represented on the static I-V characteristics curve above as the "knee" point.

Forward Biased Junction Diode showing a Reduction in the Depletion Layer


Diode Forward Bias
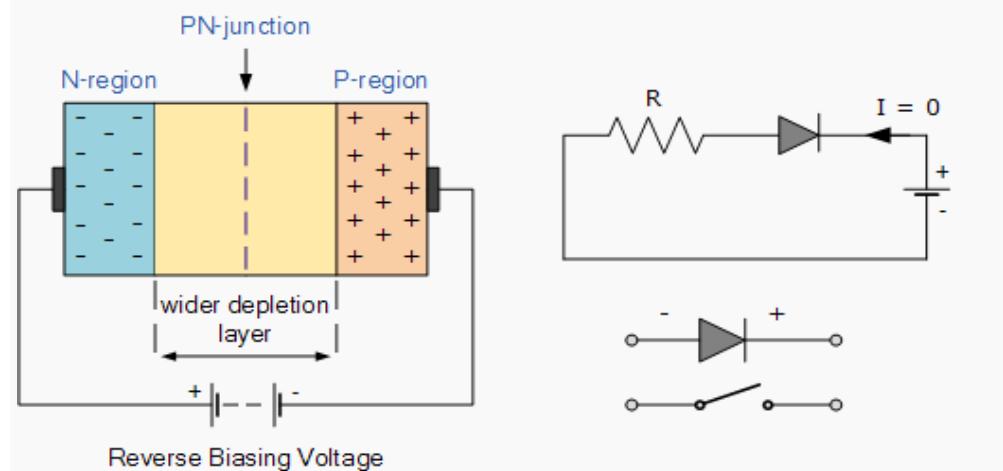
This condition represents the low resistance path through the PN junction allowing very large currents to flow through the diode with only a small increase in bias voltage. The actual potential difference across the junction or diode is kept constant by the action of the depletion layer at approximately 0.3v for germanium and approximately 0.7v for silicon junction diodes. Since the diode can conduct "infinite" current above this knee point as it effectively becomes a short circuit, therefore resistors are used in series with the diode to limit its current flow. Exceeding its maximum forward current specification causes the device to dissipate more power in the form of heat than it was designed for resulting in a very quick failure of the device.

Pn Junction Diode

Reverse Biased Junction Diode

When a diode is connected in a Reverse Bias condition, a positive voltage is applied to the N-type material and a negative voltage is applied to the P-type material. The positive voltage applied to the N- type material attracts electrons towards the positive electrode and away from the junction, while the holes in the P-type end are also attracted away from the junction towards the negative electrode. The net result is that the depletion layer grows wider due to a lack of electrons and holes and presents a high impedance path, almost an insulator. The result is that a high potential barrier is created thus preventing current from flowing through the semiconductor material.


Diode Reverse Bias

This condition represents a high resistance value to the PN junction and practically zero current flows through the junction diode with an increase in bias voltage. However, a very small leakage current does flow through the junction which can be measured in microamperes, (μA). One final point, if the reverse bias voltage Vr applied to the diode is increased to a sufficiently high enough

9
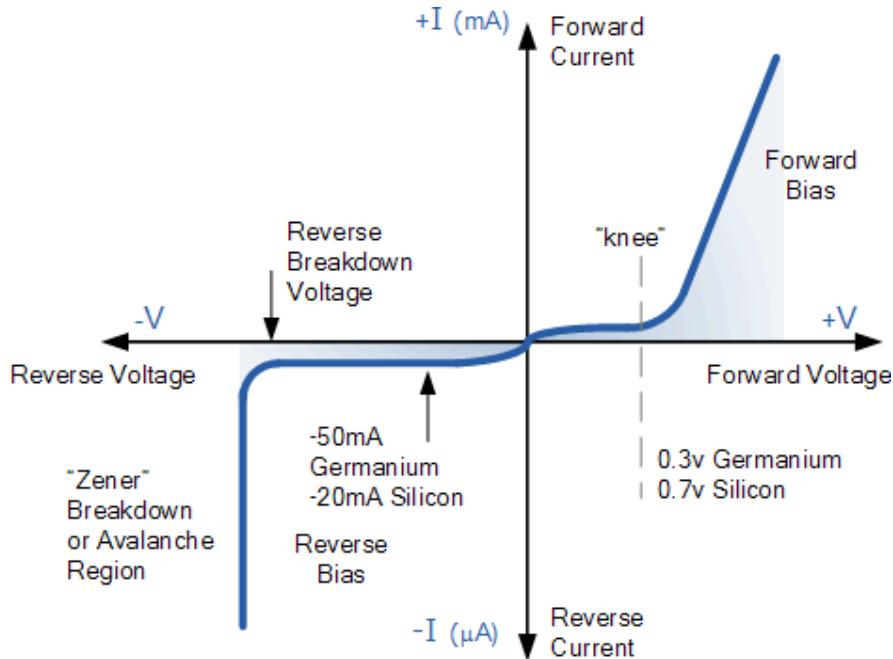
value, it will cause the PN junction to overheat and fail due to the avalanche effect around the junction. This may cause the diode to become shorted and will result in the flow of maximum circuit current, and this shown as a step downward slope in the reverse static characteristics curve below.

Reverse Characteristics Curve for a Junction Diode

Diode Reverse Characteristics

Sometimes this avalanche effect has practical applications in voltage stabilizing circuits where a series limiting resistor is used with the diode to limit this reverse breakdown current to a preset maximum value thereby producing a fixed voltage output across the diode. These types of diodes are commonly known as Zener Diodes
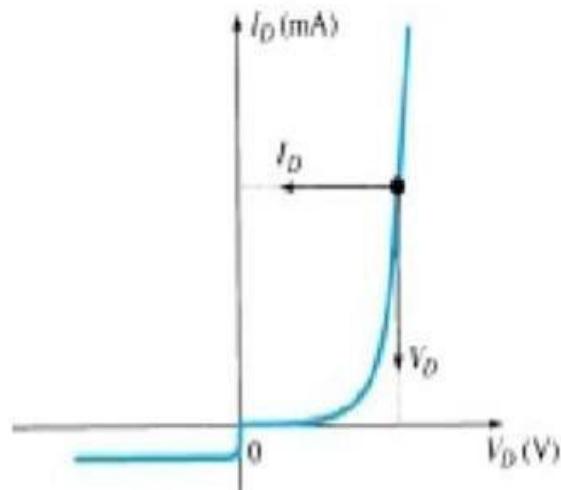
Diode Characteristics

DC or Static Resistance

The application of a dc voltage to a circuit containing a semiconductor diode will result in an operating point on the characteristic curve that will not change with time. The resistance of the diode at the operating point can be found simply by finding the corresponding levels of VD and ID as shown in Fig. 1.12 and applying the following Equation:

$$R_D = \frac{V_D}{I_D}$$

The dc resistance levels at the knee and below will be greater than the resistance levels obtained for the vertical rise section of the characteristics. The resistance levels in the reverse-bias region will naturally be quite high. Since ohmmeters typically employ a relatively constant-current source, the resistance determined will be at a preset current level (typically, a few mill amperes).

Determining the dc resistance of a diode at a particular operating point.

AC or Dynamic Resistance

It is obvious from Eq. 1.3 that the dc resistance of a diode is independent of the shape of the characteristic in the region surrounding the point of interest. If a sinusoidal rather than dc input is applied, the situation will change completely. The varying input will move the instantaneous operating point up and down a region of the characteristics and thus defines a specific change in current and voltage as shown in Fig. 1.13. With no applied varying signal, the point of operation would be the Q- point appearing on Fig. 1.13 determined by the applied dc levels. The designation Q-point is derived from the word quiescent, which means "still or unvarying." A straight-line drawn tangent to the curve through the Q-point as shown in Fig. 1.13 will define a particular change in voltage and current that can be used to determine the ac or dynamic resistance for this region of the diode characteristics. In equation form,

$$r_d = \frac{\Delta V_d}{\Delta I_d}$$
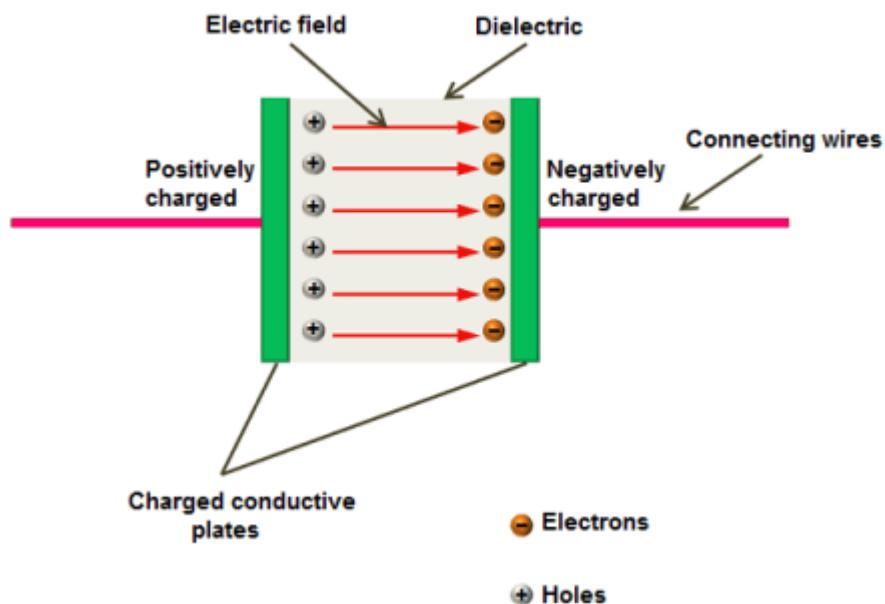
In a p-n junction diode, two types of capacitance take place. They are,

- Transition capacitance (C_T)
- Diffusion capacitance (C_D)

Transition capacitance (C_T)
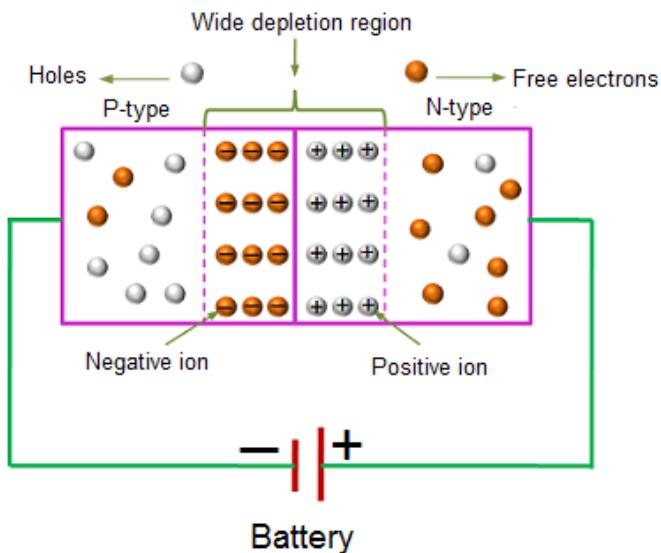
We know that capacitors store electric charge in the form of electric field. This charge storage is done by using two electrically conducting plates (placed close to each other) separated by an insulating material called dielectric.

The conducting plates or electrodes of the capacitor are good conductors of electricity. Therefore, they easily allow electric current through them. On the other hand, dielectric material or medium is poor conductor of electricity. Therefore, it does not allow electric current through it. However, it efficiently allows electric field.

When voltage is applied to the capacitor, charge carriers starts flowing through the conducting wire. When these charge carriers reach the electrodes of the capacitor, they experience a strong opposition from the dielectric or insulating material. As a result, a large number of charge carriers are trapped at the electrodes of the capacitor. These charge carriers cannot move between the plates. However, they exerts electric field between the plates. The charge carriers which are trapped near the dielectric material will stores electric charge. The ability of the material to store electric charge is called capacitance.

13

In a basic capacitor, the capacitance is directly proportional to the size of electrodes or plates and inversely proportional to the distance between two plates. Just like the capacitors, a reverse biased p-n junction diode also stores electric charge at the depletion region. The depletion region is made of immobile positive and negative ions. In a reverse biased p-n junction diode, the p-type and n-type regions have low resistance. Hence, p-type and n-type regions act like the electrodes or conducting plates of the capacitor. The depletion region of the p-n junction diode has high resistance. Hence, the depletion region acts like the dielectric or insulating material. Thus, p-n junction diode can be considered as a parallel plate capacitor. In depletion region, the electric charges (positive and negative ions) do not move from one place to another place. However, they exert electric field or electric force. Therefore, charge is stored at the depletion region in the form of electric field. The ability of a material to store electric charge is called capacitance. Thus, there exists a capacitance at the depletion region.

Reverse bias

The capacitance at the depletion region changes with the change in applied voltage. When reverse bias voltage applied to the p-n junction diode is increased, a large number of holes (majority carriers) from p-side and electrons (majority carriers) from n-side are moved away from the p-n junction. As a result, the width of depletion region increases whereas the size of p-type and n-type regions (plates) decreases. We know that capacitance means the ability to store electric charge. The p-n junction diode with narrow depletion width and large p-type and n-type regions will store large amount of electric charge whereas the p-n junction diode with wide depletion width and small p-type and n-type regions will store only a small amount of electric charge. Therefore, the capacitance of the reverse bias p-n junction diode decreases when voltage increases.

In a forward biased diode, the transition capacitance exist. However, the transition capacitance is very small compared to the diffusion capacitance. Hence, transition capacitance is neglected in forward biased diode. The amount of capacitance changed with increase in voltage is called transition capacitance. The transition capacitance is also known as depletion region capacitance, junction capacitance or barrier capacitance. Transition capacitance is denoted as C_T. The change of capacitance at the depletion region can be defined as the change in electric charge per change in voltage.

$$CT = dQ / dV$$

Where,

CT = Transition capacitance

dQ = Change in electric charge

dV = Change in voltage

The transition capacitance can be mathematically written as,

$C_T = \varepsilon A / W$
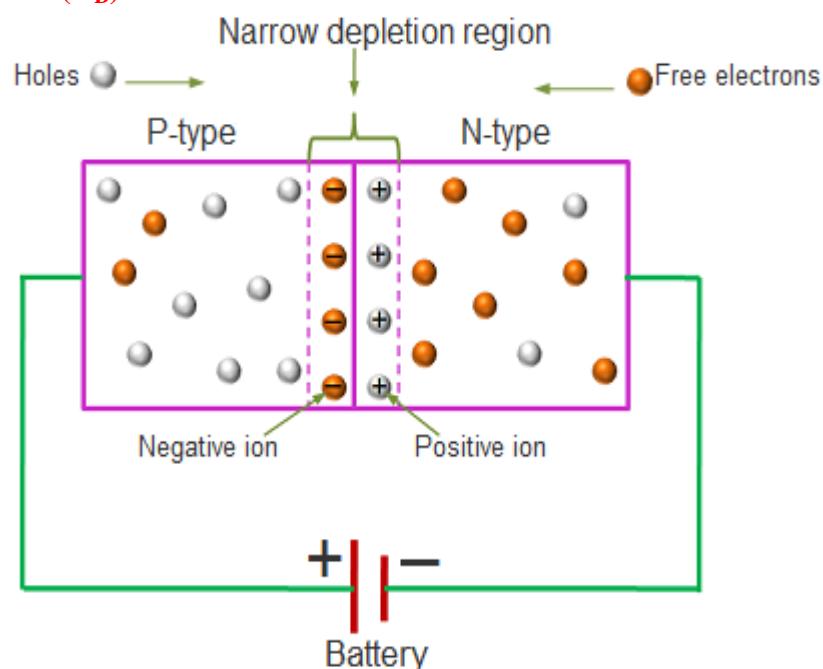
Where,

ε = Permittivity of the semiconductor

A = Area of plates or p-type and n-type regions

W = Width of depletion region

Diffusion capacitance (C_D)

Forward bias

15

The accumulation of holes in the n-region and electrons in the p-region is separated by a very thin depletion region or depletion layer. This depletion region acts like dielectric or insulator of the capacitor and charge stored at both sides of the depletion layer acts like conducting plates of the capacitor.Diffusion capacitance is directly proportional to the electric current or applied voltage. If large electric current flows through the diode, a large amount of charge is accumulated near the depletion layer. As a result, large diffusion capacitance occurs. In the similar way, if small electric current flows through the diode, only a small amount of charge is accumulated near the depletion layer. As a result, small diffusion capacitance occurs. When the width of depletion region decreases, the diffusion capacitance increases. The diffusion capacitance value will be in the range of nano farads (nF) to micro farads (μF).

$$C_T = dQ \,/\, dV$$

Where,

C_T = Transition capacitance

dQ = Change in electric charge

dV = Change in voltage

In forward bias, the diffusion capacitance is the dominant and is given by:

$C_D = dQ \,/\, dV$

Where,

C_D = Diffusion capacitance

dQ = Change in number of minority carriers stored outside the depletion region
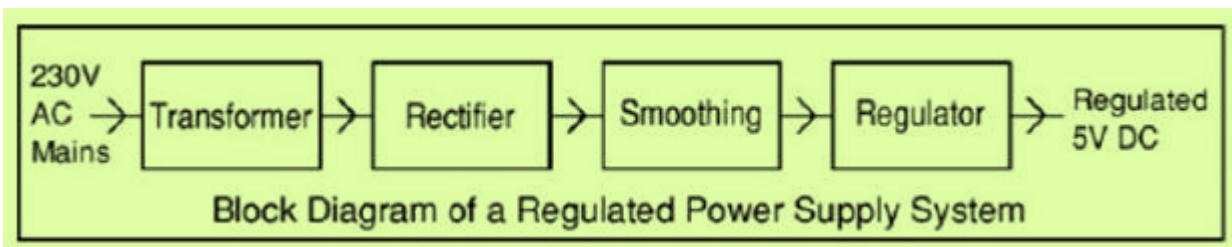
dV = Change in voltage applied across diode

Rectifiers & Filters

Introduction:

For the operation of most of the electronics devices and circuits, a d.c. source is required. So it is advantageous to convert domestic a.c. supply into d.c.voltages. The process of converting a.c. voltage into d.c. voltage is called as rectification. This is achieved with i) Step-down Transformer, ii) Rectifier,

iii) Filter and iv) Voltage regulator circuits.

These elements constitute d.c. regulated power supply shown in the fig 1 below.

230V AC Mains → Transformer → Rectifier → Smoothing → Regulator → Regulated 5V DC

Block Diagram of a Regulated Power Supply System

Block Diagram of regulated D.C Power Supply

16

The block diagram of a regulated D.C. power supply consists of step-down transformer, rectifier, filter, voltage regulator and load. An ideal regulated power supply is an electronics circuit designed to provide a predetermined d.c. voltage Vo which is independent of the load current and variations in the input voltage ad temperature. If the output of a regulator circuit is a AC voltage then it is termed as voltage stabilizer, whereas if the output is a DC voltage then it is termed as voltage regulator.

Rectifier

Any electrical device which offers a low resistance to the current in one direction but a high resistance to the current in the opposite direction is called rectifier. Such a device is capable of converting a sinusoidal input waveform, whose average value is zero, into a unidirectional Waveform, with a non- zero average component. A rectifier is a device, which converts a.c. voltage (bi-directional) to pulsating d.c. voltage (Unidirectional).

Characteristics of a Rectifier Circuit:

Any electrical device which offers a low resistance to the current in one direction but a high resistance to the current in the opposite direction is called rectifier. Such a device is capable of converting a sinusoidal input waveform, whose average value is zero, into a unidirectional waveform, with a non- zero average component.

A rectifier is a device, which converts a.c. voltage (bi-directional) to pulsating d.c..Load currents: They are two types of output current. They are average or d.c. current and RMS currents.

Average or DC current: The average current of a periodic function is defined as the area of one cycle of the curve divided by the base.

It is expressed mathematically as

i)   **Average value/dc value/mean value**= $\dfrac{Area\ over\ one\ period}{Total\ time\ period}$

$$V_{dc} = \frac{1}{T}\int_{0}^{T} V d\,(wt)$$

ii) Effective (or) R.M.S current:

The effective (or) R.M.S. current squared of a periodic function of time is given by the area of one cycle of the curve, which represents the square of the function divided by the base.

$$V_{rms} = \sqrt{\frac{1}{T}\int_0^T V^2 d(wt)}$$

iii) Peak factor:

It is the ratio of peak value to Rms value

$$\text{Peak factor} = \frac{peakvalue}{rmsvalue}$$

Form factor:

It is the ratio of Rms value to average value

$$\text{Form factor} = \frac{Rmsvalue}{averagevalue}$$

iv)Ripple Factor :

It is defined as ration of R.M.S. value of a.c. component to the d.c. component in the output is known as "Ripple Factor".

Any electrical device which offers a low resistance to the current in one direction but a high resistance to the current in the opposite direction is called rectifier. Such a device is capable of converting a sinusoidal input waveform, whose average value is zero, into a unidirectional Waveform, with a non- zero average component. A rectifier is a device, which converts a.c. voltage (bi-directional) to pulsating d.c. voltage (Unidirectional).

Efficiency:

It is the ratio of d.c output power to the a.c. input power. It signifies, how efficiently the rectifier circuit

$$\eta = \frac{o / p \ power}{i / p \ power}$$

converts a.c. power into d.c. power.

v) Peak Inverse Voltage (PIV):

It is defined as the maximum reverse voltage that a diode can withstand without destroying the junction.

vi) Transformer Utilization Factor (UTF):

The d.c. power to be delivered to the load in a rectifier circuit decides the rating of the Transformer used in the circuit. So, transformer utilization factor is defined as

vii) % Regulation:

The variation of the d.c. output voltage as a function of d.c. load current is called regulation. The percentage regulation is defined as

$$\% \ Re \ gulation = \frac{V_{NL} - V_{FL}}{V_{FL}} * 100$$

For an ideal power supply, % Regulation is zero.

Classification Of Rectifiers

Using one or more diodes in the circuit, following rectifier circuits can be designed.

1) Half - Wave Rectifier

2) Full – Wave Rectifier

3) Bridge Rectifier

Half-Wave Rectifier:

A Half – wave rectifier as shown in **fig 1.2** is one, which converts a.c. voltage into a pulsating voltage using only one half cycle of the applied a.c. voltage.
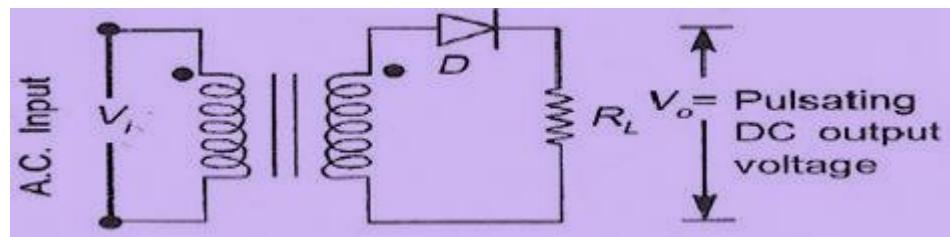

Fig 1.2: Basic structure of Half-Wave Rectifier

The a.c. voltage is applied to the rectifier circuit using step-down transformer-rectifying element i.e., p- n junction diode and the source of a.c. voltage, all connected is series. The a.c. voltage is applied to the rectifier circuit using step-down transformer
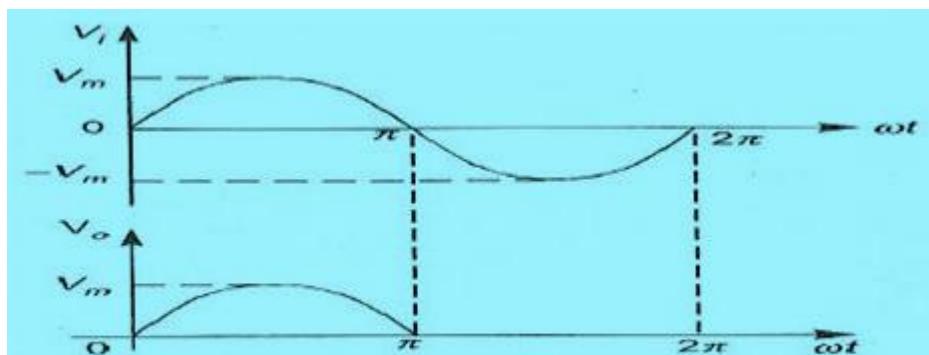
fig 3 Input and output waveforms of a Half wave rectifier

$$V=V_m \sin (wt)$$

The input to the rectifier circuit, Where V_m is the peak value of secondary a.c. voltage.

Operation:

For the positive half-cycle of input a.c. voltage, the diode D is forward biased and hence it conducts. Now a current flows in the circuit and there is a voltage drop across RL. The waveform of the diode current . For the negative half-cycle of input, the diode D is reverse biased and hence it does not Conduct. Now no current flows in the circuit i.e., i=0 and Vo=0. Thus for the negative half- cycle no power is delivered to the load.

Disadvantages Of Half-Wave Rectifier:

1. The ripple factor is high.

2. The efficiency is low.

3. The Transformer Utilization factor is low.

    Because of all these disadvantages, the half-wave rectifier circuit is normally not used as a power rectifier circuit.

Full Wave Rectifier:

A full-wave rectifier converts an ac voltage into a pulsating dc voltage using both half cycles of the applied ac voltage. In order to rectify both the half cycles of ac input, two diodes are used in this circuit. The diodes feed a common load RL with the help of a center-tap transformer. A center-tap transformer is the one, which produces two sinusoidal waveforms of same magnitude and frequency but out of phase with respect to the ground in the secondary winding of the transformer. The full wave rectifier is shown in the **fig 4** below

fig 4   Full-Wave Rectifier.
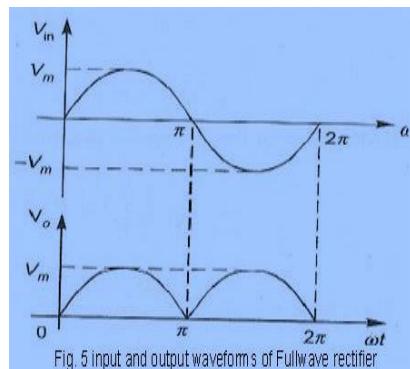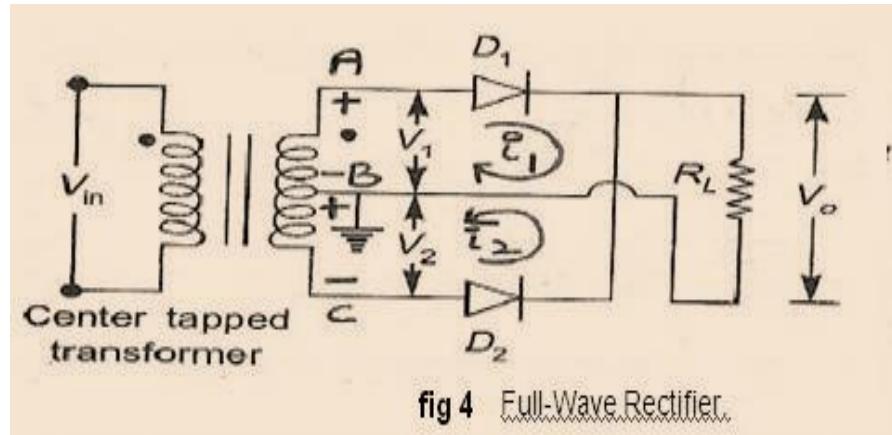
Fig. 5 input and output waveforms of Fullwave rectifier

21

During positive half of the input signal, anode of diode D1 becomes positive and at the same time the anode of diode D2 becomes negative. Hence D1 conducts and D2 does not conduct. The load current flows through D1 and the voltage drop across RL will be equal to the input voltage. During the negative half cycle of the input, the anode of D1 becomes negative and the anode of D2 becomes positive. Hence, D1 does not conduct and D2 conducts. The load current flows through D2 and the voltage drop across RL will be equal to the input voltage. It is noted that the load current flows in the both the half cycles of ac voltage and in the same direction through the load resistance.

Average voltage

$$V_{dc} = I_{dc} . R_L . = \frac{2 I_m}{\pi} . R_L \quad \text{We know } I_m = \frac{V_m}{R_S + R_f + R_L}$$

$$\therefore V_{dc} = \frac{2.V_m R_L}{\pi (R_S + R_f + R_L)}$$

$$If (R_s + R_f) << R_L$$

$$V_{dc} = \frac{2V_m}{\pi} = 0.637 V_m.$$

Average Current

$$\frac{1}{2\pi} \int_0^{2\pi} i d\theta = \frac{1}{2\pi} \int_0^{2\pi} I_m \sin \theta dt$$

$$I_{dc} \qquad = \frac{I_m}{2\pi} \left[\int_0^{\pi} \sin \theta d\theta - \int_{\pi}^{2\pi} \sin \theta d\theta \right]$$

$$= \frac{I_m}{2\pi} [(-2)(-2)]$$

$$= \frac{I_m}{2\pi} .4 = \frac{2 I_m}{\pi} = 0.637 I_m .$$

$$\boxed{I_{dc} \qquad = 0.637 I_m.}$$

$$\therefore I_{DC} \text{ FWR} \quad = 2 I_{DC} \text{ HWR.}$$

Ripple Factor:

$$\gamma = \sqrt{\left(\frac{I_{rms}}{I_{dc}}\right)^2 - 1}$$

for FWR,

$$I_{rms} = \frac{I_m}{\sqrt{2}} \quad \& \quad I_{DC} = \frac{2 I_m}{\pi}$$

$$\therefore \gamma_{FWR} = \sqrt{\left(\frac{I_m}{\sqrt{2}} \Big/ \frac{2 I_m}{\pi}\right)^2 - 1}$$

$$= \sqrt{\left(\frac{\pi}{2\sqrt{2}}\right)^2 - 1}$$

$$= \sqrt{\left(\frac{3.1416}{2 \times 1.414}\right)^2 - 1} = 0.483$$

Efficiency :

$$\eta = \frac{P_{dc}}{P_{ac}} \times 100\%$$

For FWR, $\quad p_{dc} = I_{dc}^2 . R_L = \left(\frac{2}{\pi} . I_m\right)^2 . R_L$

$P_{ac} = I_{rms}^2 (R_f + R_S + R_L)$

$$\left(\frac{I_m}{\sqrt{2}}\right)^2 (R_f + R_s + R_L)$$

$$\eta = \frac{\dfrac{I_m^2 \, 4}{\pi^2} . R_L}{\dfrac{I^2 m^2}{2} . (R_f + R_s + R_L)}$$

$If (R_f + R_s) \ll R_L$

$$\eta = \frac{4}{\pi^2} . \frac{2}{1} = \frac{8}{\pi^2} = 0.812 = 81.2\%$$

Transformer Utilization Factor (TUF):

The d.c. power to be delivered to the load in a rectifier circuit decides the rating of the transformer used in the circuit. So, transformer utilization factor is defined as

a) $\text{TUF (Secondary)} = \dfrac{P_{dc} \ delivered \ to \ load}{AC \ power \ rating \ of \ transformer \ secondary}$

b) Since both the windings are used $\text{TUF}_{FWR} = 2 \ \text{TUF}_{HWR}$
$$= \quad 2 \times 0.287 = 0.574$$

c) $\text{TUF primary} = \text{Rated efficiency} = \dfrac{P_{dc}}{P_{ac}} \times 100 = 81.2\%$

d) $\text{Average} = \dfrac{0.812 + 0.574}{2} = 0.693$
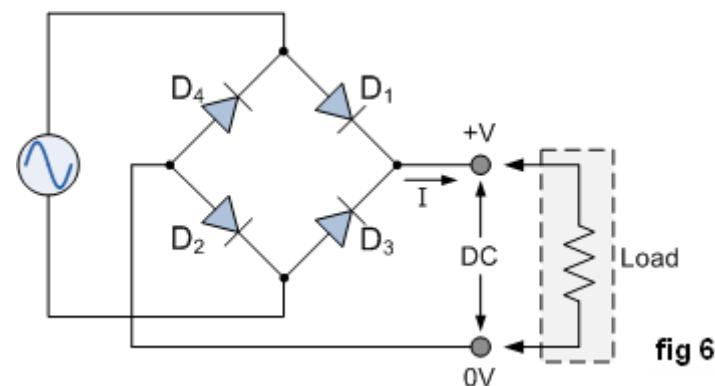
Peak Inverse Voltage (PIV):

It is defined as the maximum reverse voltage that a diode can withstand without destroying the junction. The peak inverse voltage across a diode is the peak of the negative half- cycle. For half-wave rectifier.

Bridge Rectifier.
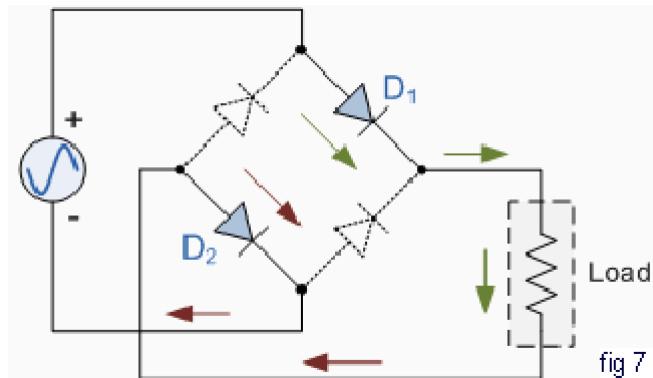
Another type of circuit that produces the same output waveform as the full wave rectifier circuit above, is that of the **Full Wave Bridge Rectifier**. This type of single phase rectifier uses four individual rectifying diodes connected in a closed loop "bridge" configuration to produce the desired output. The main advantage of this bridge circuit is that it does not require a special centre tapped transformer, thereby reducing its size and cost. The single secondary winding is connected to one side of the diode bridge network and the load to the other side as shown below.

### <u>The Diode Bridge Rectifier</u>

fig 6
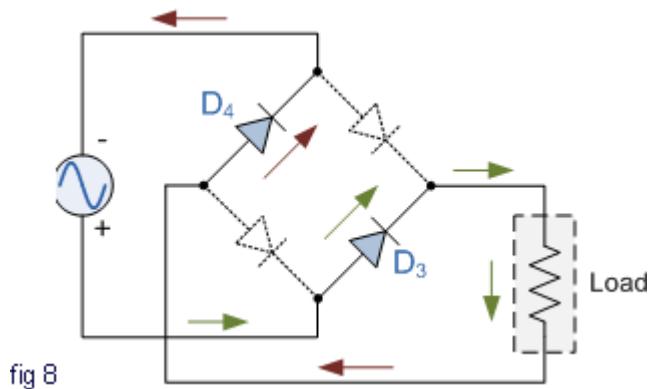
The four diodes labelled D_1 to D_4 are arranged in "series pairs" with only two diodes conducting current during each half cycle. During the positive half cycle of the supply, diodes D1 and D2 conduct in series while diodes D3 and D4 are reverse biased and the current flows through the load as shown below (fig 7).

The Positive Half-cycle

fig 7

The Negative Half-cycle

During the negative half cycle of the supply, diodes D3 and D4 conduct in series (fig 8), but diodes D1 and D2 switch "OFF" as they are now reverse biased. The current flowing through the load is the same direction as before.

fig 8

As the current flowing through the load is unidirectional, so the voltage developed across the load is also unidirectional the same as for the previous two diode full-wave rectifier, therefore the average voltage across the load is $0.637V_{max}$. However in reality, during each half cycle the current flows through two diodes instead of just one so the amplitude of the output voltage is two voltage drops (2 x 0.7 = 1.4V) less than the input V_{MAX} amplitude. The ripple frequency is

25

Therefore, the following expressions are same as that of full wave rectifier.

a) Average current $I_{dc} = \dfrac{2I_m}{\pi}$

b) RMS current $I_{rms} = \dfrac{I_m}{\sqrt{2}}$

c) DC output voltage (no.load) V_{DC} $\dfrac{2V_m}{\pi}$

d) Ripple factor $\gamma = 0.482$

e) Rectification efficiency $= \eta = 0.812$

f) DC output voltage full load.

$$= V_{DCFL} = \dfrac{2V_m}{\pi} - I_{dc}(R_S + 2R_f); \qquad \text{i.e., less by one diode loss.}$$

TUF of both primary & secondary are 0.812 therefore TUF overall is 0.812 (better than FWR with 0.693)

now twice the supply frequency (e.g. 100Hz for a 50Hz supply)

Comparison:				
Sl No.	Parameter	HWR	FWR	BR
1	No. of diodes	1	2	4
2	PIV of diodes	V_m	$2V_m$	V_m
3	Secondary voltage (rms)	V	V-0-V	V
4	DC output voltage at no load	$\dfrac{V_m}{\pi} = 0.318\,V_m$	$\dfrac{2V_m}{\pi} = 0.636\,V_m$	$\dfrac{2V_m}{\pi} = 0.636\,V_m$
5	Ripple factor γ	1.21	0.482	0.482
6	Ripple frequency	f	2f	2f
7	Rectification efficiency η	0.406	0.812	0.812
8	TUF	0.287	0.693	0.812

Filters

The output of a rectifier contains dc component as well as ac component. Filters are used to minimize the undesirable ac i.e., ripple leaving only the dc component to appear at the output.

Some important filters are:

1. Inductor filter

2. Capacitor filter

3. LC or L section filter

4. CLC or Π-type filter

26

Capacitor Filter:

The output of a rectifier contains dc component as well as ac component. Filters are used to minimize the undesirable ac i.e., ripple leaving only the dc component to appear at the output.
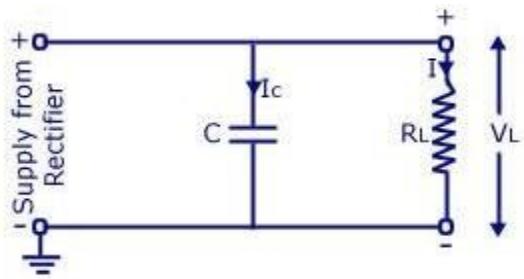
This is the most simple form of the **filter circuit** and in this arrangement a high value capacitor C is placed directly across the output terminals, as shown in figure. During the conduction period it gets charged and stores up energy to it during non-conduction period. Through this process, the time duration during which Ft is to be noted here that the capacitor C gets charged to the peak because there is no resistance (except the negligible forward resistance of diode) in the charging path. But the discharging time is quite large (roughly 100 times more than the charging time depending upon the value of RL) because it discharges through load resistance RL.

The function of the capacitor filter may be viewed in terms of impedances. The large value capacitor C offers a low impedance shunt path to the ac components or ripples but offers high impedance to the dc component. Thus ripples get bypassed through capacitor C and only dc component flows through the load resistance RL Capacitor filter is very popular because of its low cost, small size, light weight and good characteristics.

During the conduction period it gets charged and stores up energy to it during non-conduction period. Through this process, the time duration during which Ft is to be noted here that the capacitor C gets charged to the peak because there is no resistance (except the negligible forward resistance of diode) in the charging path. But the discharging time is quite large (roughly 100 times more than the charging time depending upon the value of RL) because it discharges through load resistance RL.

For a fixed-value filter capacitance larger the load resistance RL larger will be the discharge time constant CRL and therefore, lower the ripples and more the output voltage. On the other hand lower the load resistance (or more the load current), lower will be the output voltage.
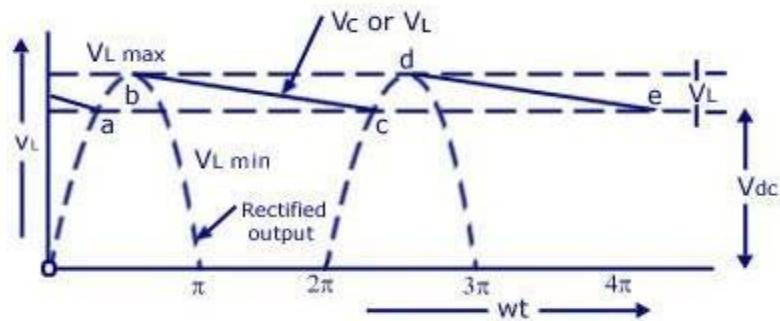
The large value capacitor C offers a low impedance shunt path to the ac components or ripples but offers high impedance to the dc component. Thus ripples get bypassed through capacitor C and only dc component flows through the load resistance RL Capacitor filter is very popular because of its low cost, small size, light weight and good characteristics.
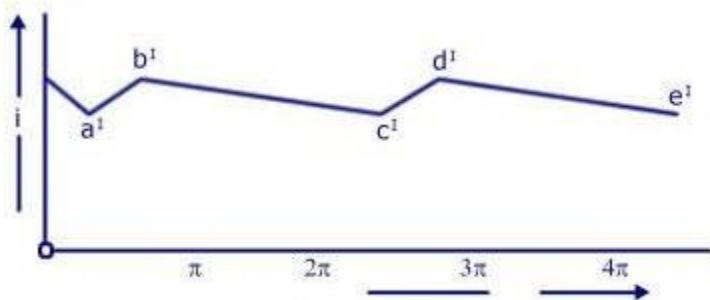
Circuit Diagram

Input voltage Waveform to Rectifier
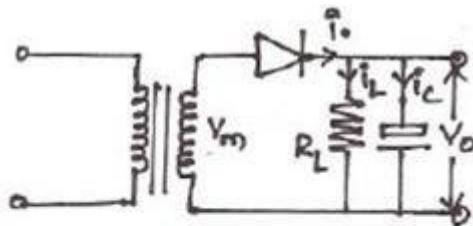
Rectified and filtered Output Voltage Waveform

Load Current Waveform
Half-wave Rectifier With Shunt Capacitor Filter

28
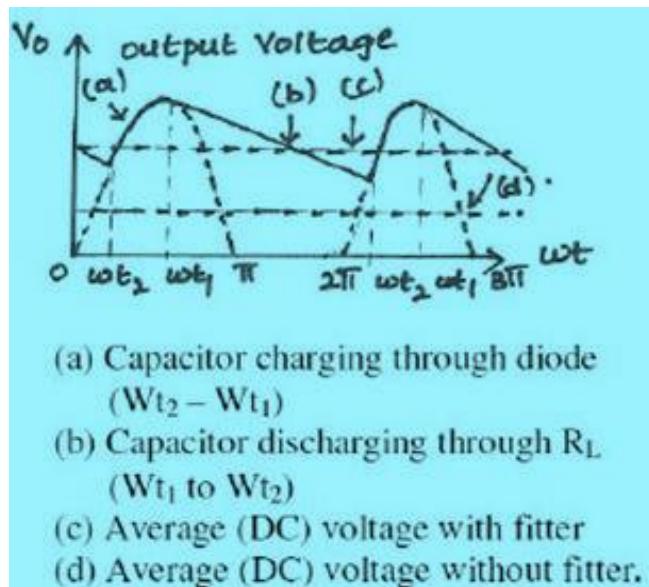
Capacitor Filter With Hwr

Cut In angle – wt_2

Cut out angle = Wt_1
$Wt_1 = \pi - \tan^{-1} WCR_L$

(a) Capacitor charging through diode
$(Wt_2 - Wt_1)$
(b) Capacitor discharging through R_L
$(Wt_1$ to $Wt_2)$
(c) Average (DC) voltage with fitter
(d) Average (DC) voltage without fitter.
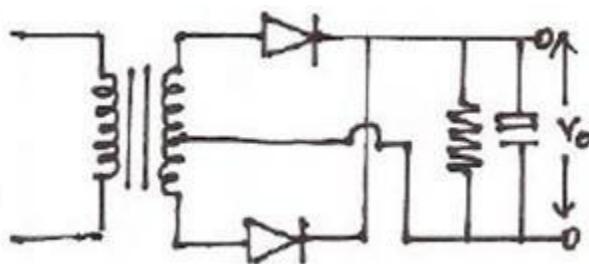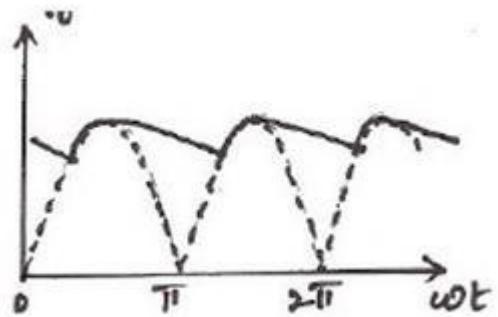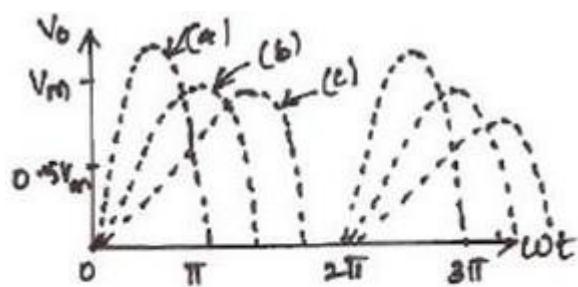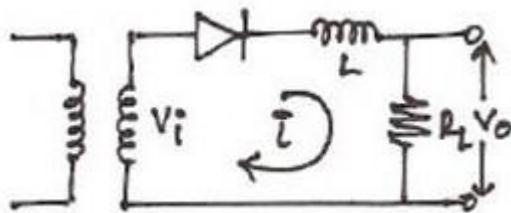
Capacitor Filter With Fwr

Ripple factor $r = \dfrac{1}{4\sqrt{3}\,fCR_L}$
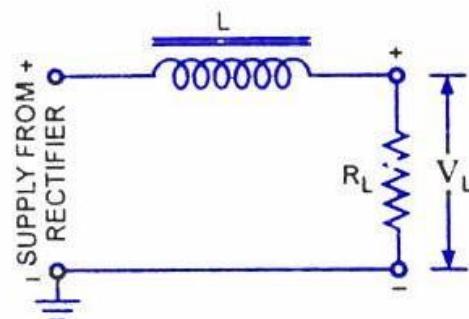
Ripple freq $_{FWR}$ = 2 ripple freq $_{HWR}$.
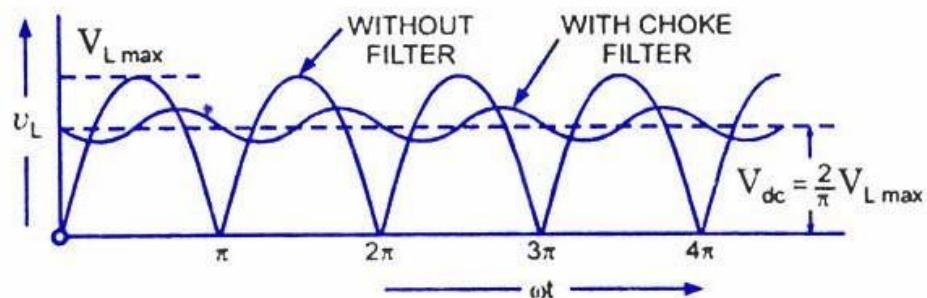
(a) $\dfrac{W_L}{R_L} = 0$ (b) $\dfrac{W_L}{R_L} = 1$ (c) $\dfrac{W_L}{R_L} = 5$.

1. For a fixed-value filter capacitance larger the load resistance RL larger will be the discharge time constant CRL and therefore, lower the ripples and more the output voltage. On the other hand lower the load resistance (or more the load current), lower will be the output voltage.

2. Similarly smaller the filter capacitor, the less charge it can hold and more it will discharge. Thus the peak-to-peak value of the ripple will increase, and the average dc level will decrease. Larger the filter capacitor, the more charge it can hold and the less it will discharge. Hence the peak-to-peak value of the ripple will be less, and the average dc level will increase. But, the maximum value of the capacitance that can be employed is limited by another factor. The larger the capacitance value, the greater is the current required to charge the capacitor to a given voltage. The maximum current that can be handled by a diode is limited by the figure quoted by the manufacturer. Thus the maximum value of the capacitance, that can be used in the shunt filter capacitor is limited.

Circuit Diagram

Output Voltage Waveforms
Full-Wave Rectifier With Series Inductor Filter

Series Inductor Filter.

In this arrangement a high value inductor or choke L is connected in series with the rectifier element and the load, as illustrated in figure. The filtering action of an inductor filter depends upon its property of opposing any change in the current flowing through it. When the output current of the rectifier
increases above a certain value, energy is stored in it in the form of magnetic field and this energy is given up when the output current falls below the average value. Thus by placing a choke coil in series with the rectifier output and load, any sudden change in current that might have occurred in the circuit without an inductor is smoothed out by the presence of the inductor L.
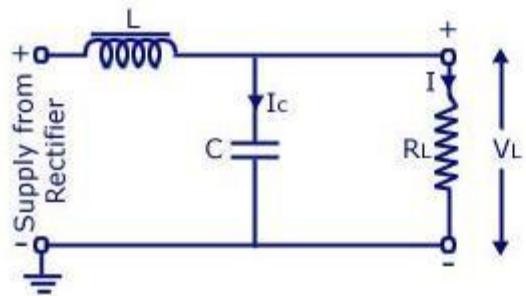
The function of the inductor filter may be viewed in terms of impedances. The choke offers high impedance to the ac components but offers almost zero resistance to the desired dc components. Thus ripples are removed to a large extent. Nature of the output voltage without filter and with choke filter is shown in figure. For dc (zero frequency), the choke resistance R_c in series with the load resistance R_L forms a voltage divider and dc voltage across the load is given as where V_{dc} is dc voltage output from a full-wave rectifier. Usually choke coil resistance Rc, is much small than R_L and, therefore, almost entire of the dc voltage is available across the load resistance R_L. Since the reactance of inductor increases with the increase in frequency, better filtering of the higher harmonic components takes place, so effect of third and higher harmonic voltages can be neglected. As obvious from equation , if choke coil resistance R_c is negligible in comparison to load resistance R_L, then the entire dc component of rectifier output is available across 2 R_L and is equal to — $V_{L\ max}$. The ac voltage partly drops across X_L and partly over R_L.

L-Section Filter:
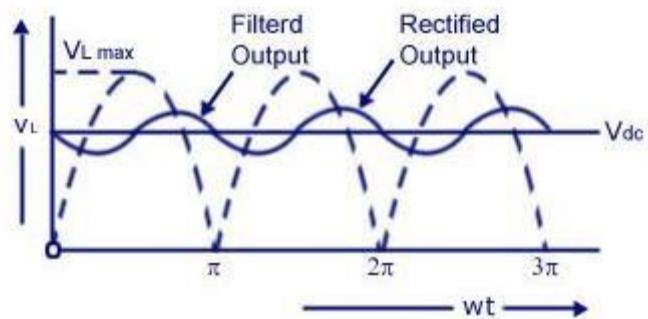
A simple series inductor reduces both the peakand effective values of the output current and output voltage. On the other hand a simple shunt capacitor filter reduces the ripple voltage but increases the diode current. The diode may get damaged due to large current and at the same time it causes greater heating of supply transformer resulting in reduced efficiency
In an inductor filter, ripple factor increases with the increase in load resistance RL while in a capacitor filter it varies inversely with load resistance RL.
From economical point of view also, neither series inductor nor shunt capacitor type filters are suitable.

Circuit Digram

Rectified and Filtered Output Voltage Waveform
Full-wave Rectifier With Choke-Input Filter

BIPOLAR JUNCTION TRANSISTOR

Introduction

A bipolar junction transistor (BJT) is a three terminal device in which operation depends on the interaction of both majority and minority carriers and hence the name bipolar. The BJT is analogues to vacuum triode and is comparatively smaller in size. It is used as amplifier and oscillator circuits, and as a switch in digital circuits. It has wide applications in computers, satellites and other modern communication systems.

Construction Of BJT And Its Symbols

The Bipolar Transistor basic construction consists of two PN-junctions producing three connecting terminals with each terminal being given a name to identify it from the other two. These three terminals are known and labelled as the Emitter (E), the Base (B) and the Collector (C) respectively. There are two basic types of bipolar transistor construction, PNP and NPN, which basically describes the physical arrangement of the P-type and N-type semiconductor materials from which they are made.

Transistors are three terminal active devices made from different semiconductor materials that can act as either an insulator or a conductor by the application of a small signal voltage. The transistor's ability to change between these two states enables it to have two basic functions: "switching" (digital electronics) or "amplification" (analogue electronics). Then bipolar transistors have the ability to operate within three different regions:

1. Active Region - the transistor operates as an amplifier and $Ic = \beta.Ib$
2. Saturation - the transistor is "fully-ON" operating as a switch and $Ic = I(saturation)$
3. Cut-off - the transistor is "fully-OFF" operating as a switch and $Ic = 0$

Bipolar Transistors are current regulating devices that control the amount of current flowing through them in proportion to the amount of biasing voltage applied to their base terminal acting like a current-controlled switch. The principle of operation of the two transistor types PNP and NPN, is exactly the same the only difference being in their biasing and the polarity of the power supply for each type(fig

Bipolar Transistor Construction

PNP Transistor

NPN Transistor

Emitter

Collector

Emitter

Collector

P   N   P

N   P   N

Base

a). Construction

Base

E   C   E   C

B

b). Two-diode Analogy

B

V_{CE}

V_{CE}

E I_E I_C C   E I_E I_C C

V_{EB} I_B V_{CB} V_{EB} I_B V_{CB}

B

B
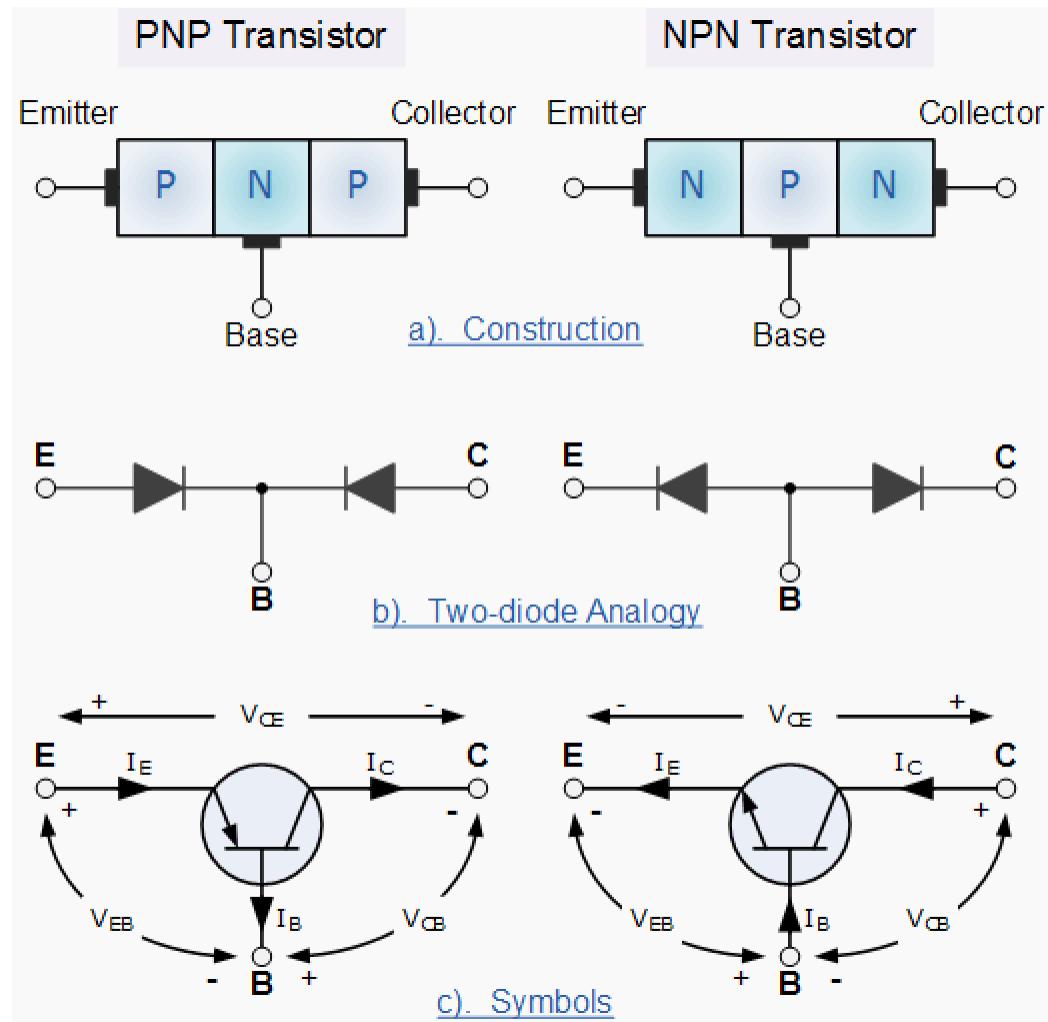
c). Symbols

Fig 3.1 Bipolar Junction Transistor Symbol

The construction and circuit symbols for both the PNP and NPN bipolar transistor are given above with the arrow in the circuit symbol always showing the direction of "conventional current flow" between the base terminal and its emitter terminal. The direction of the arrow always points from the positive P-type region to the negative N-type region for both transistor types, exactly the same as for the standard diode symbol.

Transistor Current Components:
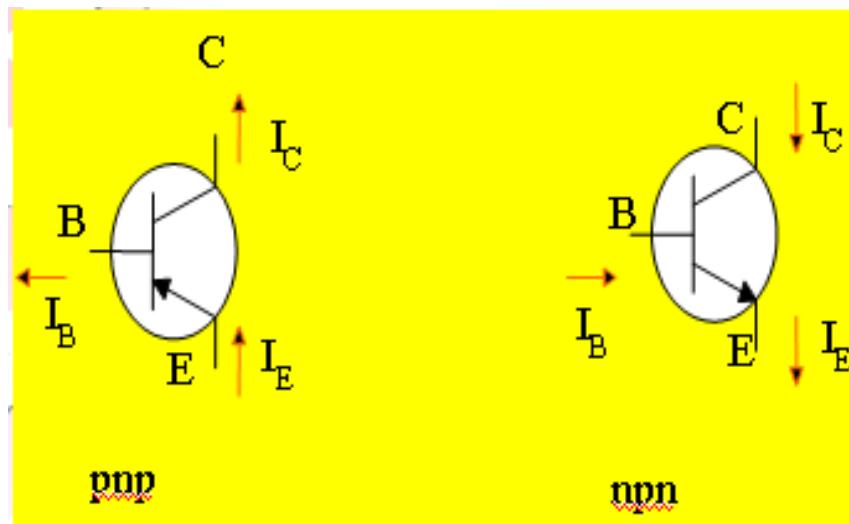
Fig 3.2 Bipolar Junction Transistor Current Components

The above fig 3.2 shows the various current components, which flow across the forward biased emitter junction and reverse- biased collector junction. The emitter current I_E consists of hole current I_{PE} (holes crossing from emitter into base) and electron current I_{nE} (electrons crossing from base into emitter).The ratio of hole to electron currents, I_{pE} $_/$ I_{nE} , crossing the emitter junction is proportional to the ratio of the conductivity of the p material to that of the n material. In a transistor, the doping of that of the emitter is made much larger than the doping of the base. This feature ensures (in p-n-p transistor) that the emitter current consists an almost entirely of holes. Such a situation is desired since the current which results from electrons crossing the emitter junction from base to emitter do not contribute carriers, which can reach the collector.

Not all the holes crossing the emitter junction J_E reach the the collector junction J_C
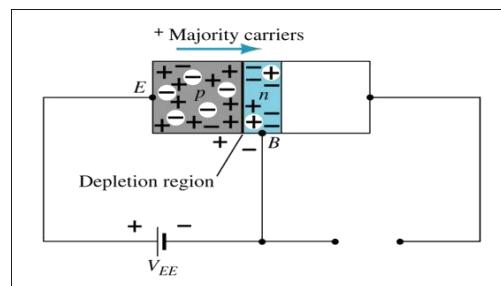
Because some of them combine with the electrons in n-type base. If I_{pC} is hole current at junction J_C there must be a bulk recombination current (I_{PE}- I_{pC}) leaving the base.

Actually, electrons enter the base region through the base lead to supply those charges, which have been lost by recombination with the holes injected in to the base across J_E. If the emitter were open circuited so that $I_E=0$ then I_{pC} would be zero. Under these circumstances, the base and collector current I_C would equal the reverse saturation current I_{CO}. If $I_E \neq 0$ then
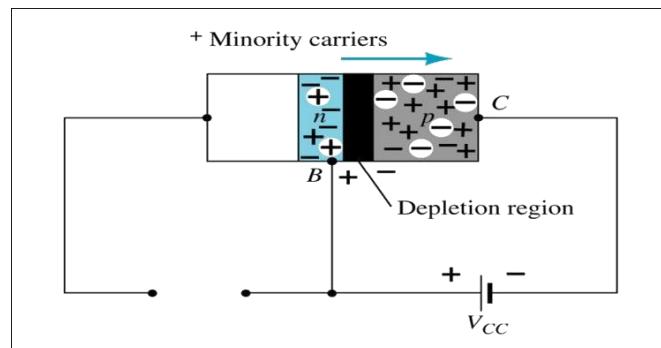
IC= ICO- IpC

For a p-n-p transistor, I_{CO} consists of holes moving across J_C from left to right (base to collector) and electrons crossing J_C in opposite direction. Assumed referenced direction for I_{CO} i.e. from right to left, then for a p-n-p transistor, I_{CO} is negative. For an n-p-n transistor, I_{CO} is positive. The basic operation will be described using the pnp transistor. The operation of the pnp transistor is exactly the same if the roles played by the electron and hole are interchanged.
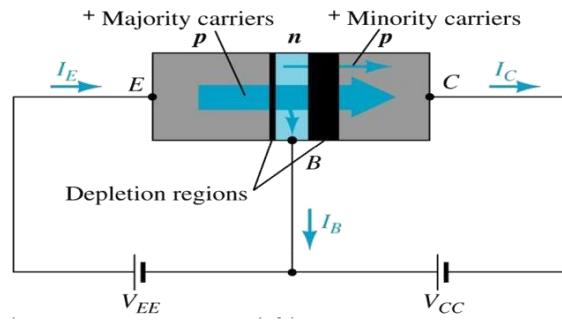
One p-n junction of a transistor is reverse-biased, whereas the other is forward-biased.

3.3a Forward-biased junction of a pnp transistor

3.3b Reverse-biased junction of a pnp transistor

Both biasing potentials have been applied to a pnp transistor and resulting majority and minority carrier flows indicated.

Majority carriers (+) will diffuse across the forward-biased p-n junction into the n-type material.

A very small number of carriers (+) will through n-type material to the base terminal. Resulting IB is typically in order of microamperes.

The large number of majority carriers will diffuse across the reverse-biased junction into the p-type material connected to the collector terminal

Applying KCL to the transistor :

$$I_E = I_C + I_B$$

Bipolar Transistor Configurations

As the **Bipolar Transistor** is a three terminal device, there are basically three possible ways to connect it within an electronic circuit with one terminal being common to both the input and output. Each method of connection responding differently to its input signal within a circuit as the static characteristics of the transistor vary with each circuit arrangement.

1. Common Base Configuration - has Voltage Gain but no Current Gain.
2. Common Emitter Configuration - has both Current and Voltage Gain.
3. Common Collector Configuration - has Current Gain but no Voltage Gain.

Common-Base Configuration

Common-base terminology is derived from the fact that the : base is common to both input and output of t configuration. base is usually the terminal closest to or at ground potential. Majority carriers can cross the reverse-biased junction because the injected majority carriers will appear as minority carriers in the n-type material. All current directions will refer to conventional (hole) flow and the arrows in all electronic symbols have a direction defined by this convention.

Note that the applied biasing (voltage sources) are such as to establish current in the direction indicated for each branch.
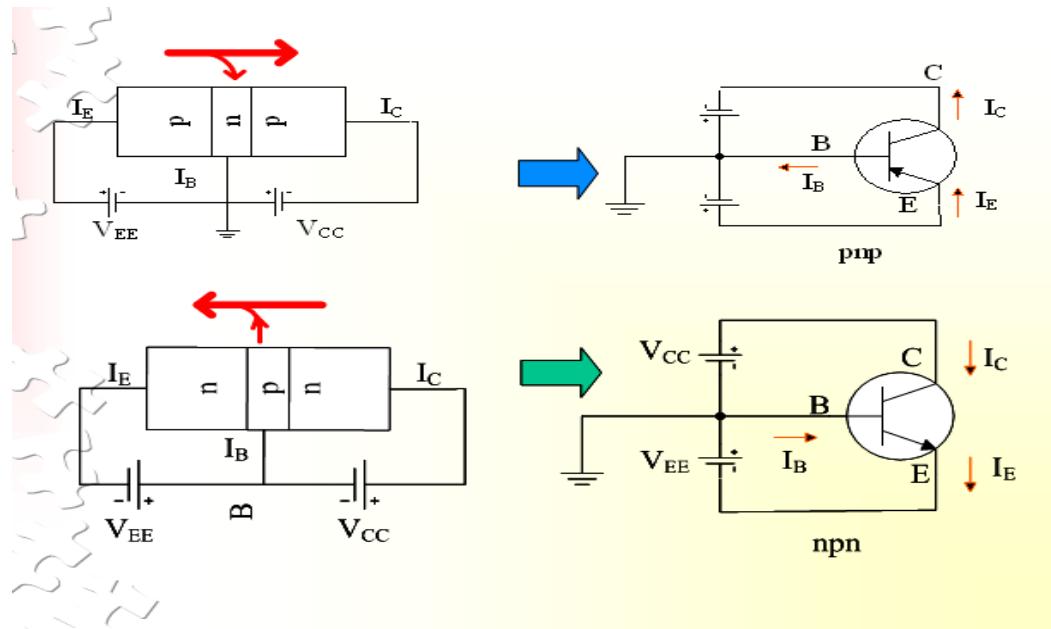

Fig 3.4 CB Configuration

To describe the behavior of common-base amplifiers requires two set of characteristics:

1. Input or driving point characteristics.

2. Output or collector characteristics

The output characteristics has 3 basic regions:

*   Active region –defined by the biasing arrangements

*   Cutoff region – region where the collector current is 0A

- Saturation region- region of the characteristics to the left of $V_{CB} = 0V$
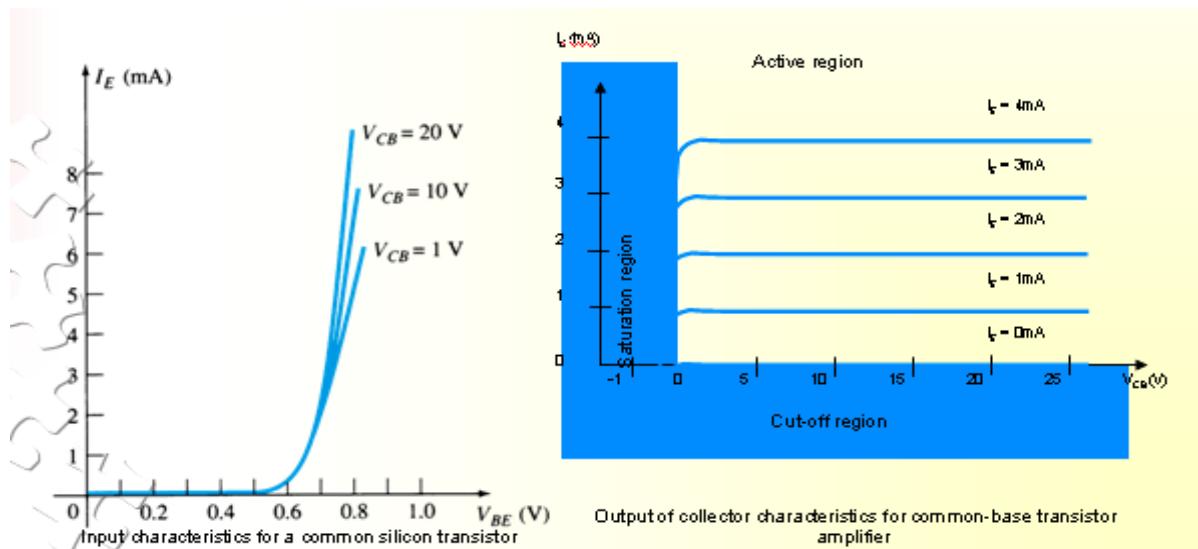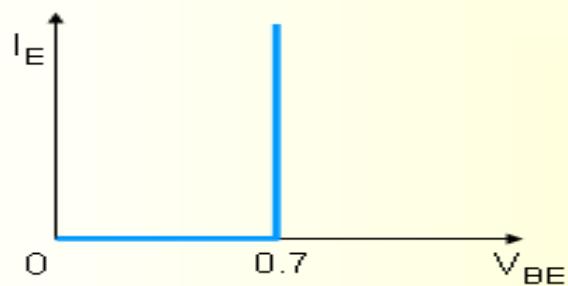
Fig 3.5 CB Input-Output Characteristics

Active region	Saturation region	Cut-off region
•IE increased, Ic increased •BE junction forward bias and CB junction reverse bias •Refer to the graf, Ic ≈ IE •Ic not depends on VCB •Suitable region for the transistor working as amplifier	• BE and CB junction is forward bias • Small changes in VCB will cause big different to Ic • The allocation for this region is to the left of VCB= 0 V.	• Region below the line of IE=0 A • BE and CB is reverse bias • no current flow at collector, only leakage current

The curves (output characteristics) clearly indicate that a first approximation to the relationship between IE and IC in the active region is given by

$I_C \approx IE$

Once a transistor is in the 'on' state, the base-emitter voltage will be assumed to be $V_{BE} = 0.7V$

40

In the dc mode the level of I_C and I_E due to the majority carriers are related by a quantity called alpha

$\square = \alpha_{dc}$

$I_C = \square I_E + I_{CBO}$

It can then be summarize to $I_C = \square I_E$ (ignore I_{CBO} due to small value)

For ac situations where the point of operation moves on the characteristics curve, an ac alpha defined by α_{ac}

Alpha a common base current gain factor that shows the efficiency by calculating the current percent from current flow from emitter to collector. The value of $\square$ is typical from $0.9 \sim 0.998$.

Biasing: Proper biasing CB configuration in active region by approximation $I_C \square I_E$ ($I_B \square 0$ uA)
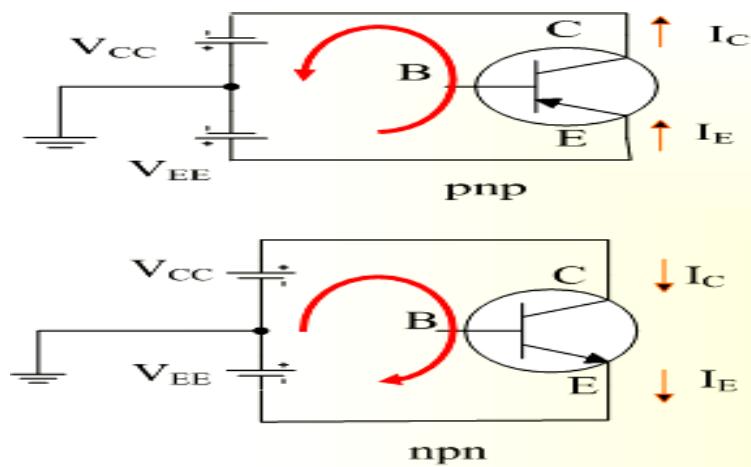
Fig 3.6 CE Configuration

41

Transistor As An Amplifier
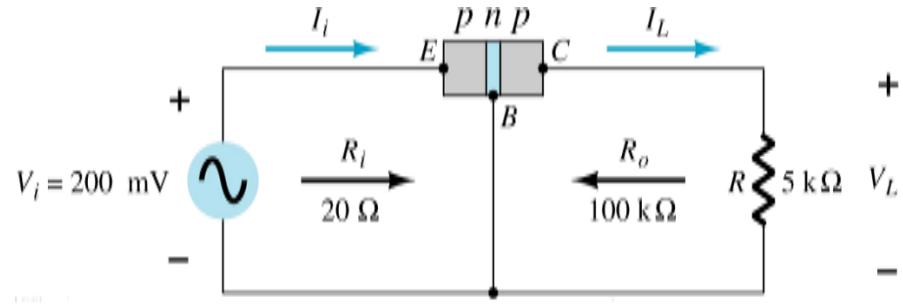
Fig 3.7 Basic Transistor Amplifier Circuit

Common-Emitter Configuration

It is called common-emitter configuration since : emitter is common or reference to both input and output terminals.emitter is usually the terminal closest to or at ground potential. Almost amplifier design is using connection of CE due to the high gain for current and voltage. Two set of characteristics are necessary to describe the behavior for CE ;input (base terminal) and output (collector terminal) parameters.

 Proper Biasing common-emitter configuration in active region

(a) npn transistor configuration

$$I_E = I_C + I_B$$

(b) pnp transistor configuration

Fig 4.7 : Common-emitter configuration
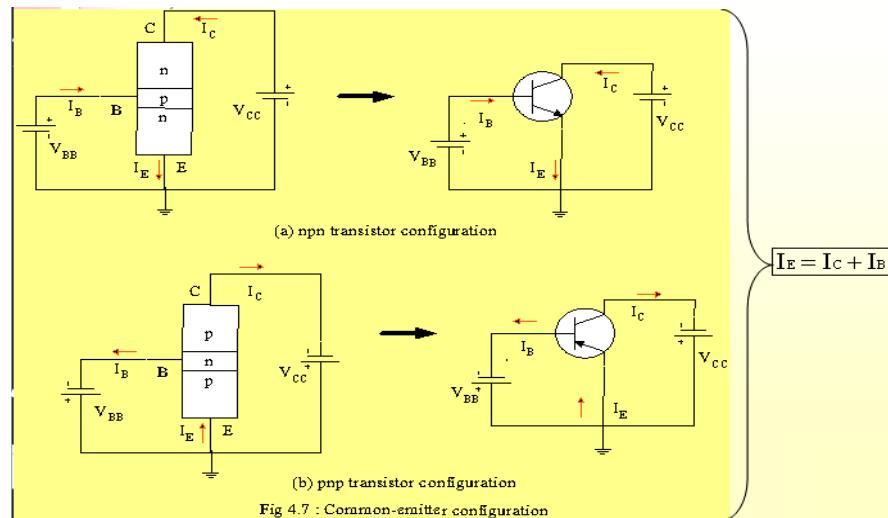
Fig 3.8 CE Configuration

I_B is microamperes compared to miliamperes of I_C.
I_B will flow when $V_{BE} > 0.7V$ for silicon and 0.3V for germanium Before this value I_B is very small and no I_B.
Base-emitter junction is forward bias Increasing V_{CE} will reduce I_B for different values.
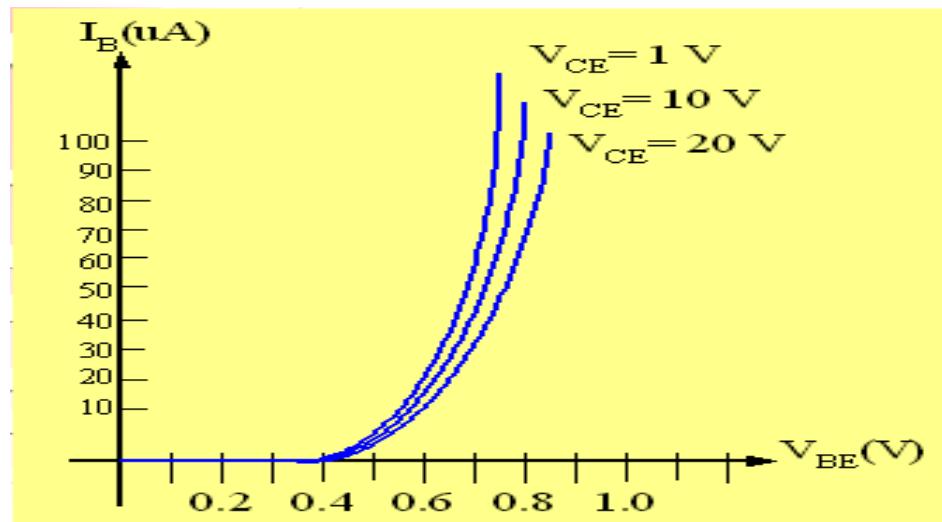
Fig 3.9a Input characteristics for common-emitter npn transistor
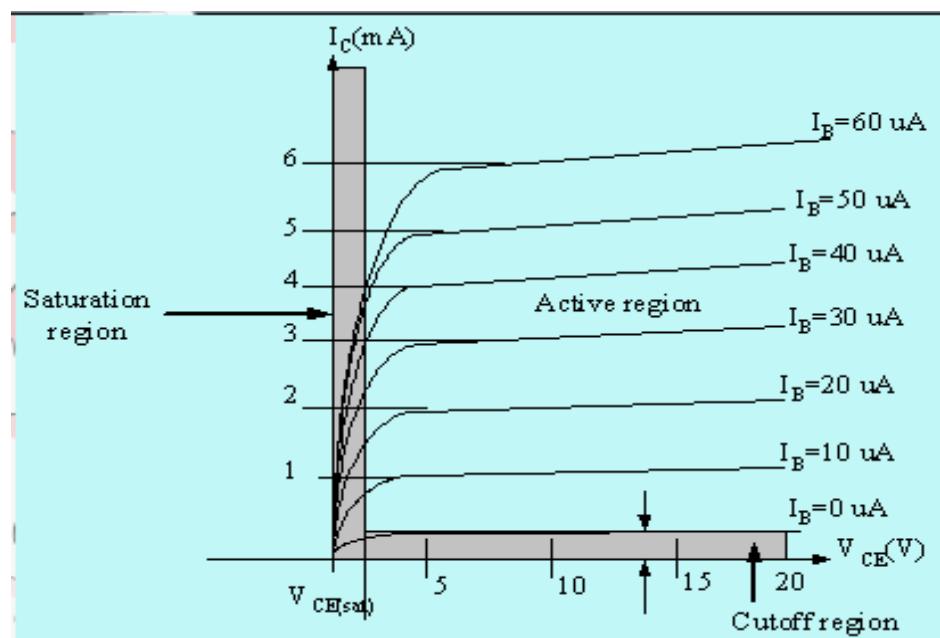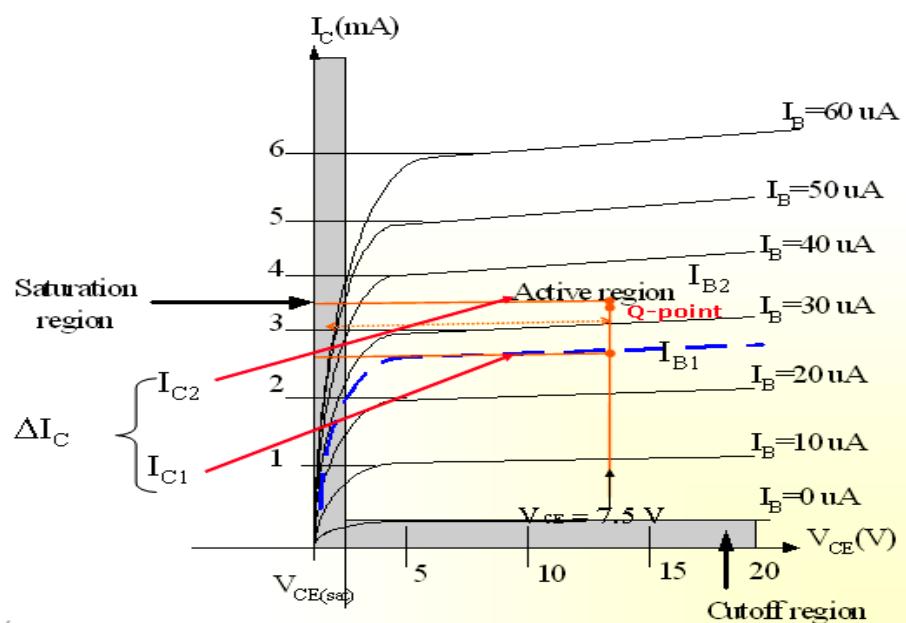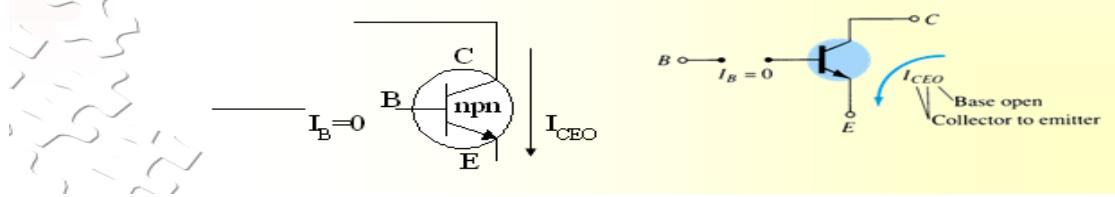
Fig 3.9b Output characteristics for common-emitter npn transistor

For small V_{CE} ($V_{CE} < V_{CESAT}$, I_C increase linearly with increasing of V_{CE} $V_{CE} > V_{CESAT}$ I_C not totally depends on V_{CE} □ constant I_C

I_B(uA) is very small compare to I_C (mA). Small increase in I_B cause big increase in I_C I_B=0 A □ I_{CEO} occur.

Noticing the value when I_C=0A. There is still some value of current flows.

Active region	Saturation region	Cut-off region
• B-E junction is forward bias • C-B junction is reverse bias • can be employed for voltage, current and power amplification	• B-E and C-B junction is forward bias, thus the values of I_B and I_C is too big. • The value of V_{CE} is so small. • Suitable region when the transistor as a logic switch. • NOT and avoid this region when the transistor as an amplifier.	• region below I_B=0µA is to be avoided if an undistorted o/p signal is required • B-E junction and C-B junction is reverse bias • I_B=0, I_C not zero, during this condition I_C=I_{CEO} where is this current flow when B-E is reverse bias.

44

Relationship analysis between α and β

CASE 1

$$I_E = I_C + I_B \tag{1}$$

subtitute equ. $I_C = \beta I_B$ into (1) we get

$$I_E = (\beta + 1)I_B$$

CASE 2

known : $\alpha = \dfrac{I_C}{I_E} \Rightarrow I_E = \dfrac{I_C}{\alpha} \tag{2}$
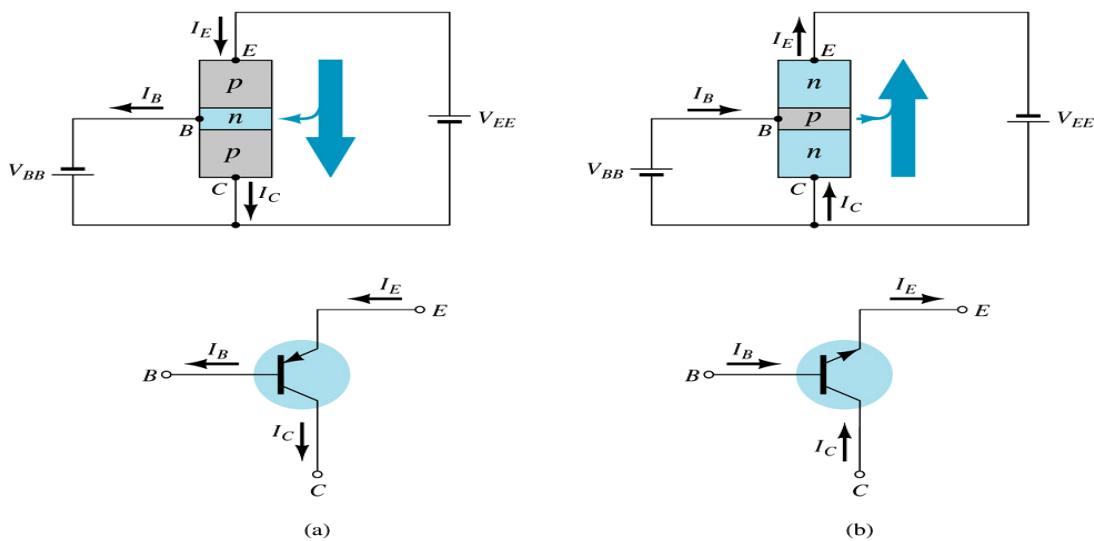
known : $\beta = \dfrac{I_C}{I_B} \Rightarrow I_B = \dfrac{I_C}{\beta} \tag{3}$

subtitute (2) and (3) into (1) we get,
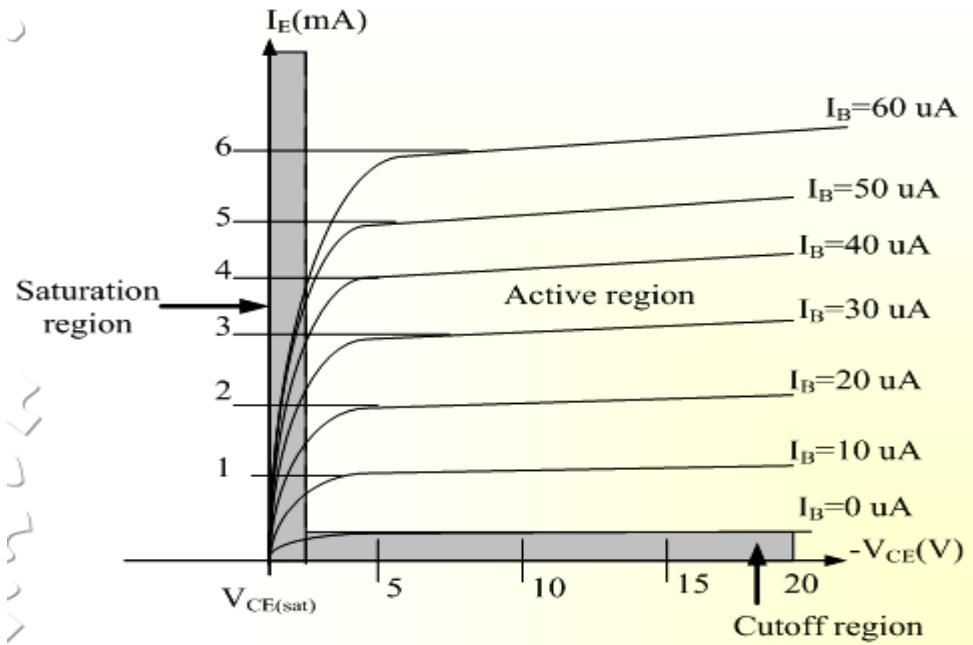
$$\alpha = \dfrac{\beta}{\beta + 1} \qquad \text{and} \qquad \beta = \dfrac{\alpha}{1 - \alpha}$$

Common – Collector Configuration

Also called emitter-follower (EF). It is called common-emitter configuration since both the signal source and the load share the collector terminal as a common connection point.The output voltage is obtained at emitter terminal. The input characteristic of common-collector configuration is similar with common-emitter. configuration.Common-collector circuit configuration is provided with the load resistor connected from emitter to ground. It is used primarily for impedance- matching purpose since it has high input impedance and low output impedance.

(a)   (b)
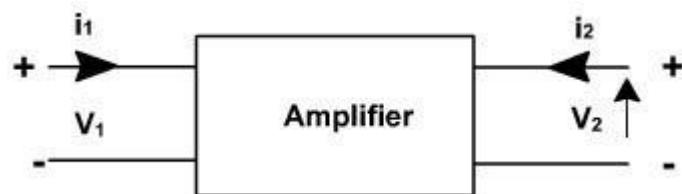
For the common-collector configuration, the output characteristics are a plot of I_E vs V_{CE} for

BJT Hybrid Model

Small signal low frequency transistor Models:

All the transistor amplifiers are two port networks having two voltages and two currents. The positive directions of voltages and currents are shown in **fig. 1**.

A two-port network is represented by four external variables: voltage V_1 and current I_1 at the input port, and voltage V_2 and current I_2 at the output port, so that the two-port network can be treated as a black box modeled by the relationships between the four variables, V_1, V_2, I_1, I_2 . Out of four variables two can be selected as are independent variables and two are dependent variables. The dependent variables can be expressed interns of independent variables. This leads to various two port parameters out of which the following three are important:

1. Impedance parameters (z-parameters)
2. Admittance parameters (y-parameters)

3. Hybrid parameters (h-parameters)

z-parameters

A two-port network can be described by z-parameters as

$V_1 = Z_{11}I_1 + Z_{12}I_2$

$V_2 = Z_{21}I_1 + Z_{22}I_2$

In matrix form, the above equation can be rewritten as

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

Where

$z_{11} = \left. \dfrac{V_1}{I_1} \right|_{I_2=0}$

$z_{12} = \left. \dfrac{V_1}{I_2} \right|_{I_1=0}$

Input impedance with output port open circuited

$z_{21} = \left. \dfrac{V_2}{I_1} \right|_{I_2=0}$

Reverse transfer impedance with input port open circuited

Forward transfer impedance with output port open circuited

$z_{22} = \left. \dfrac{V_2}{I_2} \right|_{I_1=0}$

Output impedance with input port open circuited

Y-parameters

A two-port network can be described by Y-parameters as

$I_1 = Y_{11}V_1 + Y_{12}V_2$

$I_2 = Y_{21}V_1 + Y_{22}V_2$

In matrix form, the above equation can be rewritten as

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

$$y_{11} = \left. \frac{I_1}{V_1} \right|_{V_2=0}$$

Input admittance with output port short circuited

$$y_{12} = \left. \frac{I_1}{V_2} \right|_{V_1=0}$$

$$y_{21} = \left. \frac{I_2}{V_1} \right|_{V_2=0}$$

Reverse transfer admittance with input port short circuited
Forward transfer admittance with output port short circuited

$$y_{22} = \left. \frac{I_2}{V_2} \right|_{V_1=0}$$

Output admittance with input port short circuited

Hybrid parameters (h-parameters)
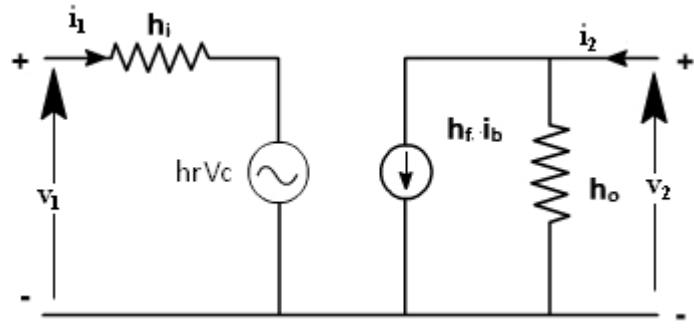
If the input current I1 and output voltage V2 are taken as independent variables, the dependent variables V1 and I2 can be written as

$$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$$

Where h_{11}, h_{12}, h_{21}, h_{22} are called as hybrid parameters.

$$h_{11} = \left. \frac{V_1}{I_1} \right|_{V_2=0}$$

Input impedence with o/p port short circuited

If these parameters are specified for a particular configuration, then suffixes e,b or c are also included, e.g. h_{fe} ,h $_{ib}$ are h parameters of common emitter and common collector amplifiers

Using two equations the generalized model of the amplifier can be drawn as shown in fig. 2.
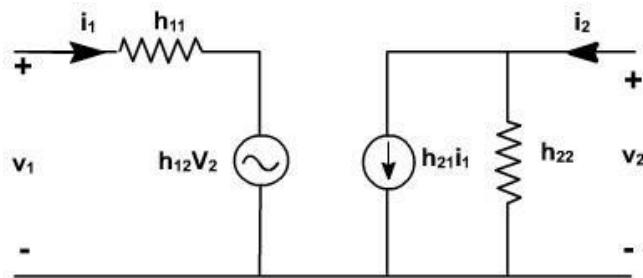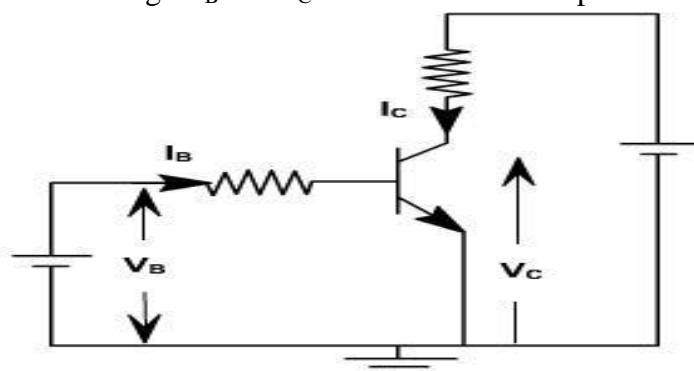
Fig. 2

Transistor Hybrid Model:

The hybrid model for a transistor amplifier can be derived as follow:

Let us consider CE configuration as show in fig. 3. The variables, i_B, i_C ,v_C, and v_B represent total instantaneous currents and voltages i_B and v_C can be taken as independent variables and v_B, I_C as

$V_B = f1$ (i_B
,v_C) $I_C = f2$
(i_B ,v_C).

Using Taylor 's series expression, and neglecting higher order terms we obtain.

$$\Delta v_B = \left.\frac{\partial f_1}{\partial i_B}\right|_{v_C} \Delta i_B + \left.\frac{\partial f_1}{\partial v_C}\right|_{i_B} \Delta v_C$$

$$\Delta i_C = \left.\frac{\partial f_2}{\partial i_B}\right|_{v_C} \Delta i_B + \left.\frac{\partial f_2}{\partial v_C}\right|_{i_B} \Delta v_C$$

The partial derivatives are taken keeping the collector voltage or base current constant. The Δv_B, Δv_C, Δi_B, Δi_C represent the small signal (incremental) base and collector current and voltage and can be represented as v_B, i_C, i_B, v_C
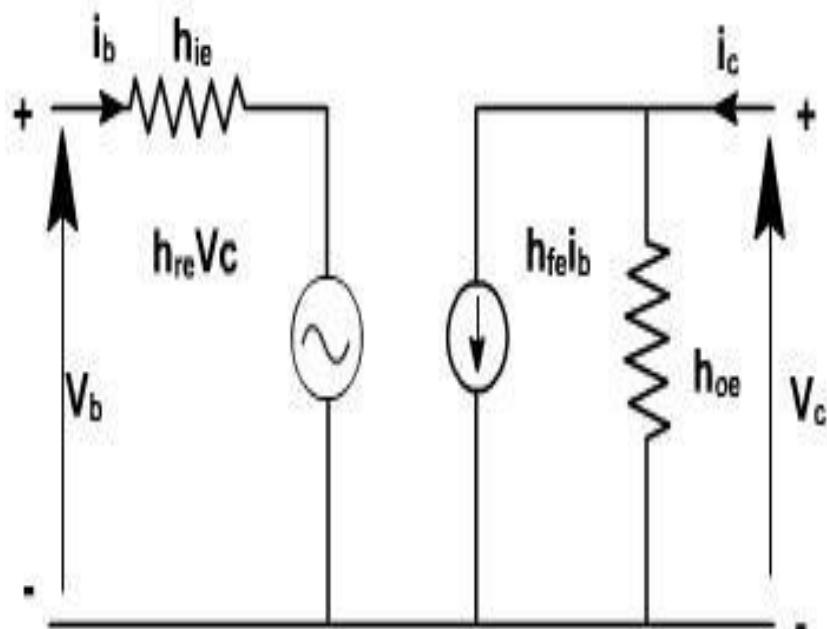
$$\therefore v_b = h_{ie} i_B + h_{re} v_C$$
$$i_C = h_{fe} i_B + h_{oe} v_b$$

where

$$h_{ie} = \left.\frac{\partial f_1}{\partial i_B}\right|_{v_C} = \left.\frac{\partial v_B}{\partial i_B}\right|_{v_C} ; \qquad h_{re} = \left.\frac{\partial f_1}{\partial v_C}\right|_{i_B} = \left.\frac{\partial v_B}{\partial v_C}\right|_{i_B}$$

$$h_{fe} = \left.\frac{\partial f_2}{\partial i_B}\right|_{v_C} = \left.\frac{\partial i_C}{\partial i_B}\right|_{v_C} ; \qquad h_{oe} = \left.\frac{\partial f_2}{\partial v_C}\right|_{i_B} = \left.\frac{\partial v_B}{\partial v_C}\right|_{i_B}$$

To determine the four h-parameters of transistor amplifier, input and output characteristic are used. Input characteristic depicts the relationship between input voltage and input current with output voltage as parameter. The output characteristic depicts the relationship between output voltage and output current with input current as parameter. Fig. 5, shows the output characteristics of CE amplifier.

$$h_{fe} = \frac{\partial i_C}{\partial i_B}\bigg|_{V_C} = \frac{i_{C2} - i_{C1}}{i_{b2} - i_{b1}}$$
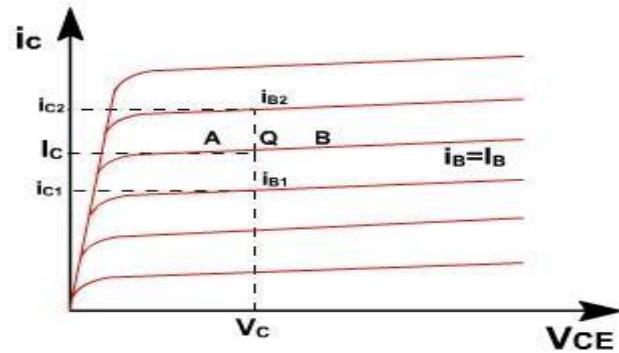
Fig. 5

The current increments are taken around the quiescent point Q which corresponds to $i_B = I_B$ and to the collector voltage $V_{CE} = V_C$
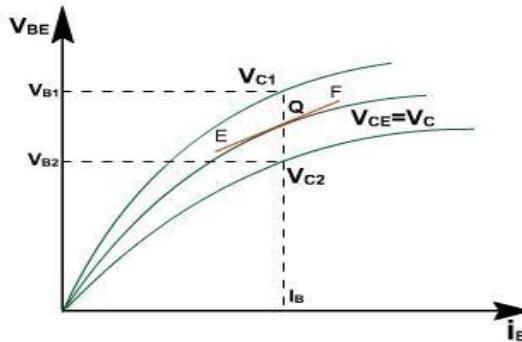
$$h_{oe} = \frac{\partial i_C}{\partial V_C}\bigg|_{i_B}$$

The value of h_{oe} at the quiescent operating point is given by the slope of the output characteristic at the operating point (i.e. slope of tangent AB)

.

$$h_{ie} = \frac{\partial V_B}{\partial i_B} \approx \frac{\Delta V_B}{\Delta i_B}\bigg|_{V_C}$$

h_{ie} is the slope of the appropriate input on fig. 6, at the operating point (slope of tangent EF at Q).

$$h_{re} = \frac{\partial V_B}{\partial V_C} = \frac{\Delta V_B}{\Delta V_C}\bigg|_{I_B} = \frac{V_{B2} - V_{B1}}{V_{C2} - V_{C1}}$$

51

A vertical line on the input characteristic represents constant base current. The parameter hre can be obtained from the ratio (V_{B2}– V_{B1}) and (V_{C2}– V_{C1}) for at Q.

Typical CE h-parametersof transistor 2N1573 are given below:

h_{ie} = 1000 ohm.
h_{re} = 2.5 * 10 –
4
h_{fe} = 50
h_{oe} = 25 □ □A / V
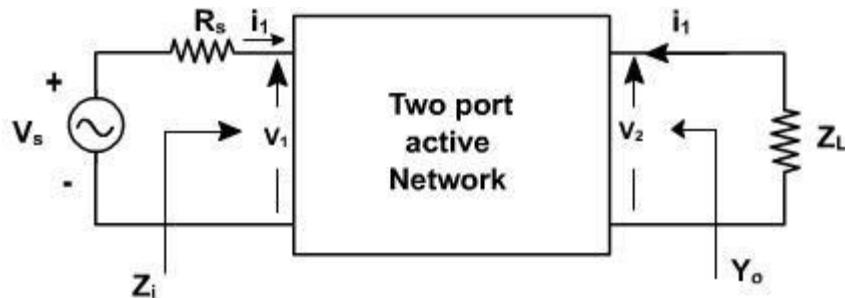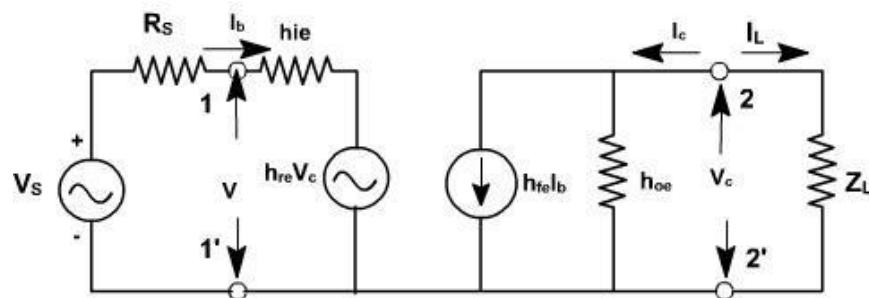
Analysis Of A Transistor Amplifier Using H-Parameters:

To form a transistor amplifier it is only necessary to connect an external load and signal source as indicated in **fig. 1** and to bias the transistor properly.

Consider the two-port network of CE amplifier. R_S is the source resistance and Z_L is the load impedence h-parameters are assumed to be constant over the operating range. The ac equivalent circuit is shown in fig. 2. (Phasor notations are used assuming sinusoidal voltage input). The quantities of interest are the current gain, input impedance, voltage gain, and output impedance.

Voltage gain:

The ratio of output voltage to input voltage gives the gain of the transistors.

$$A_v = \frac{V_c}{V_b} = -\frac{I_c Z_L}{V_b}$$

$$\therefore A_v = \frac{I_b A_i Z_L}{V_b} =$$

$$Y_0 = \frac{I_c}{V_c}\bigg|_{V_s} = 0$$

Output Admitta

$$I_c = h_{fe}I_b + h_{oe} V_c$$

$$\frac{I_c}{V_c} = h_{fe} \frac{I_b}{V_c} + h_{oe}$$

when $V_s = 0$, $R_s \cdot I_b + h_{ie} \cdot I_b + h_{re} V_c = 0$.

$$\frac{I_b}{V_c} = -\frac{h_{re}}{R_s + h_{ie}}$$

$$\therefore Y_0 = h_{oe} - \frac{h_{re}\, h_{fe}}{R_s + h_{ie}}$$
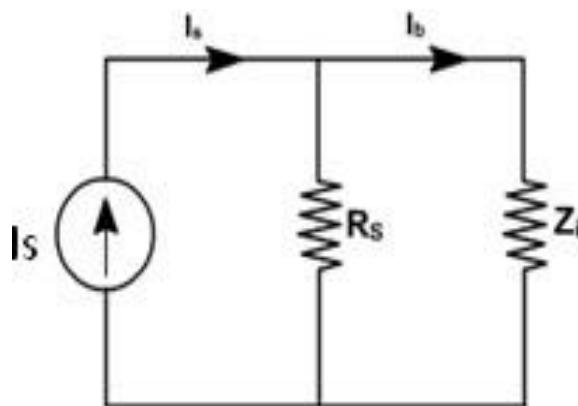
Voltage amplification taking into account source impedance (R_S) is given by

$$A_{vs} = \frac{V_c}{V_s} = \frac{V_c}{V_b} * \frac{V_b}{V_s} \qquad \left(V_b = \frac{V_s}{R_s + Z_i} * Z_i\right)$$

$$= A_v \cdot \frac{Z_i}{Z_i + R_s}$$

$$= \frac{A_i Z_L}{Z_i + R_s}$$

It is defined as

A_v is the voltage gain for an ideal voltage source $(R_v = 0)$.

Consider input source to be a current source I_S in parallel with a resistance R_S as shown in <u>fig. 3</u>.
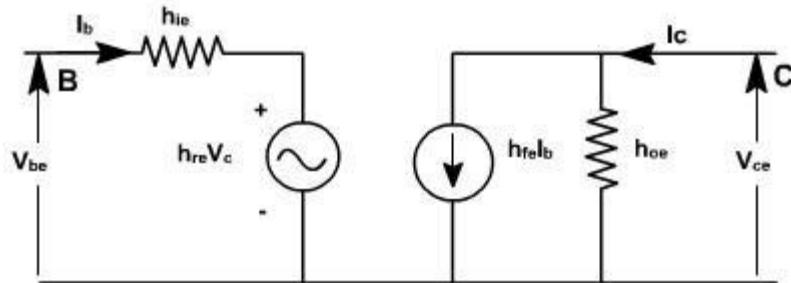
In this case, overall current gain A_{IS} is defined as

$$A_{I_s} = \frac{I_L}{I_s}$$

$$= -\frac{I_c}{I_s}$$

$$= -\frac{I_c}{I_b} * \frac{I_b}{I_s} \qquad \left(I_b = \frac{I_s * R_s}{R_s + Z_i} \right)$$

$$= A_I \cdot \frac{R_s}{R_s + Z_i}$$

$$\text{If } R_s \to \infty, \qquad A_{I_s} \to A_I$$

h-parameters

To analyze multistage amplifier the h-parameters of the transistor used are obtained from manufacture
data sheet. The manufacture data sheet usually provides h-parameter in CE configuration. These parameters may be converted into CC and CB values. For example fig. 4 hrc in terms of CE parameter can be obtained as follows.

For CE transistor
configuaration Vbe =
hie Ib + hre Vce
Ic = h fe Ib + hoe Vce
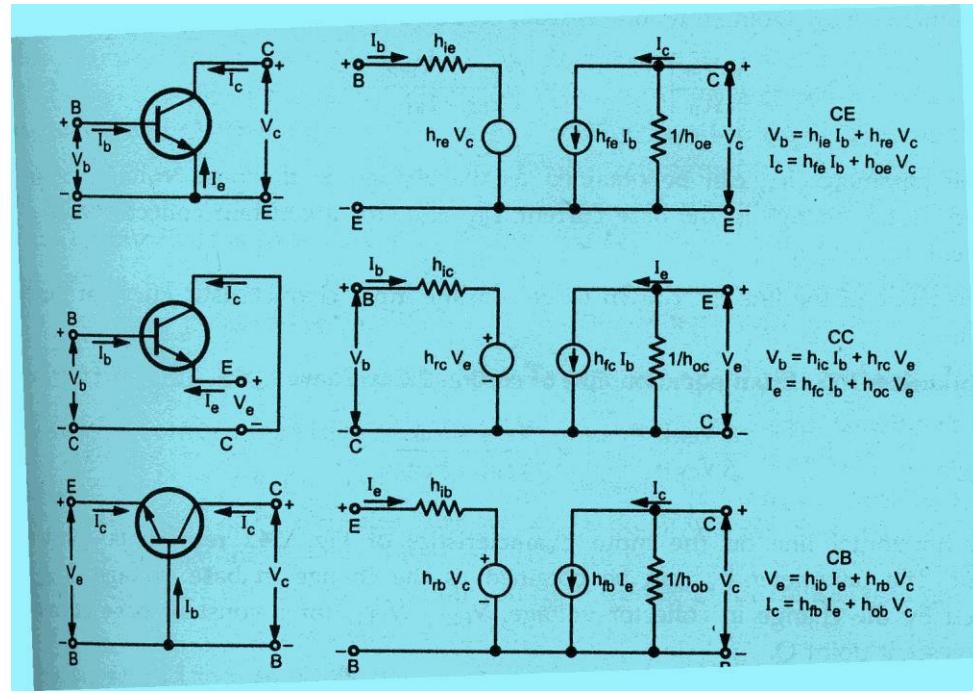
The circuit can be redrawn like CC transistor configuration as
shown in fig. 5. Vbc = hie Ib + hrc Vec

Ic = hfe Ib + hoe Vec

54
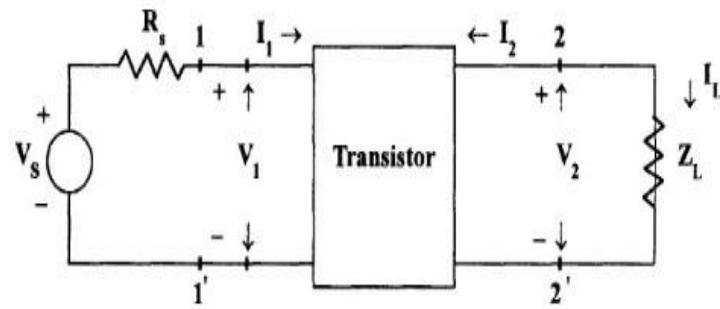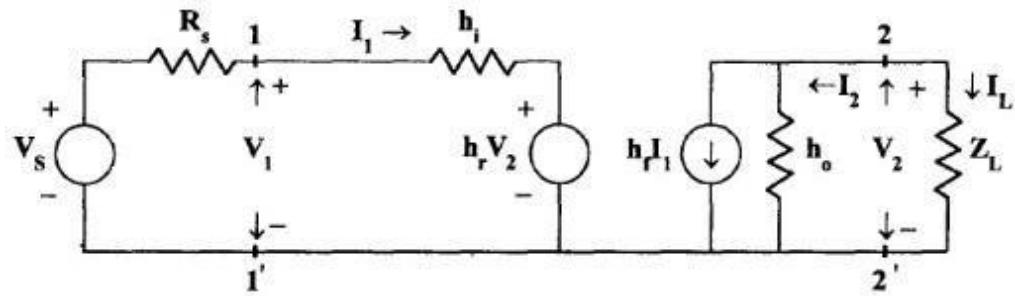
Typical h-parameter values for a transistor

Parameter	CE	CC	CB
h_i	1100 Ω	1100 Ω	22 Ω
h_r	2.5×10^{-4}	1	3×10^{-4}
h_f	50	-51	-0.98
h_o	25 μA/V	25 μA/V	0.49 μA/V

Analysis of a Transistor amplifier circuit using h-parameters

A transistor amplifier can be constructed by connecting an external load and signal source and biasing the transistor properly.

The two port network of Fig. 1.4 represents a transistor in any one of its configuration. It is assumed that h-parameters remain constant over the operating range.The input is sinusoidal and I_1,V- 1,I_2 and V_2 are phase quantities

Current Gain or Current Amplification (A_i)

For transistor amplifier the current gain A_i is defined as the ratio of output current to input current,i.e,
$A_i = I_L / I_1 = -I_2 / I_1$

From the circuit of Fig

$I_2 = h_f I_1 + h_o V_2$
Substituting $V_2 = I_L Z_L = -$
$I_2 Z_L$
$I_2 = h_f I_1 - I_2 Z_L h_o$
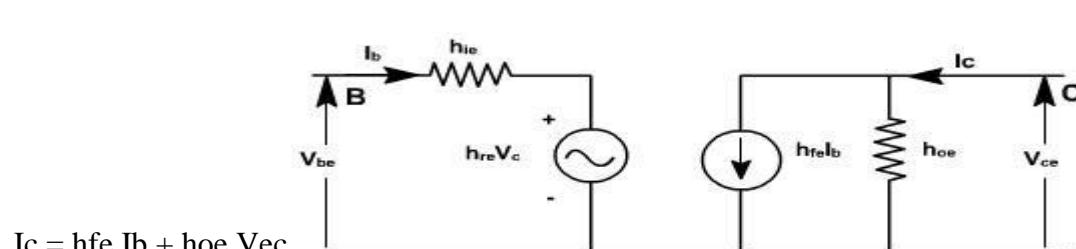
For CE transistor configuaration
$Vbe = hie\ Ib + hre\ Vce$
$Ic = h\ fe\ Ib + hoe\ Vce$

The circuit can be redrawn like CC transistor configuration as shown in fig. 5.
$Vbc = hie\ Ib + hrc\ Vec$

$Ic = hfe\ Ib + hoe\ Vec$

$I_2 +$

$I_2 Z_L$

$h_o = h_f$

$I_1 \, I_2 ($

$1 + Z_L$

$h_o) =$

$h_f \, I_1$

$A_i = -I_2 \, / \, I_1 = - \, h_f \, / \, (\, 1+$

$Z_L \, h_o)$ Therefore,

$A_i = - \, h_f \, / \, (\, 1 + Z_L$

$h_o)$

Input Impedence (Z_i)

In the circuit of Fig   , R_S is the signal source resistance .The impedence seen when looking into the amplifier terminals $(1,1^{'})$ is the amplifier input impedence Z_i,

$Z_i = V_1 \, / \, I_1$

From the input circuit of Fig V_1

$= h_i \, I_1 + h_r V_2 \; Z_i = (\, h_i \, I_1 + h_r V_2) \, /$

I_1

$= h_i + h_r$

$V_2 \, / \, I_1$

Substitutin

g

$V_2 = -I_2 \, Z_L = A_1 I_1 Z_L$

$Z_i = h_i + h_r \, A_1 I_1 Z_L \, / \, I_1$

$= h_i$

$+ \, h_r$

$A_1 Z_L$

Substit

uting

for A_i

$Z_i = h_i - h_f \, h_r \, Z_L \, / \, (1 + h_o Z_L)$

$= h_i - h_f \, h_r \, Z_L \, /$

$Z_L (1/Z_L + h_o)$

Taking the Load admittance

as $Y_L = 1 / \, Z_L \; Z_i = h_i - h_f \, h_r \, /$

$(Y_L + h_o)$

Voltage Gain or Voltage Gain Amplification Factor(A_v)

The ratio of output voltage V_2 to input voltage V_1 give the voltage gain of the transistor i.e, $A_v = V_2 / V_1$

Substituting

$V_2 = -I_2 Z_L = A_1 I_1 Z_L$

$A_v = A_1 I_1 Z_L / V_1 = A_i Z_L / Z_i$

Output Admittance (Y_o)

Y_o is obtained by setting V_S to zero, Z_L to infinity and by driving the output terminals from a generator V_2. If the current V_2 is I_2 then $Y_o = I_2/V_2$ with $V_S = 0$ and $R_L = \infty$.

From the circuit of fig

$I_2 = h_f I_1 + h_o V_2$

Dividing by V_2,

$I_2 / V_2 = h_f I_1/V_2 + h_o$

With $V_2 = 0$, by KVL in input circuit,

$R_S I_1 + h_i I_1 + h_r V_2 = 0$ $(R_S + h_i)$

$I_1 + h_r V_2 = 0$

Hence, $\qquad I_2 / V_2 = -h_r/ (R_S + h_i)$

$= h_f (-h_r/(R_S + h_i) + h_o$ $Y_o = h_o - h_f h_r/(R_S + h_i)$

The output admittance is a function of source resistance. If the source impedence is resistive then Y_o is real.

Small Signal analysis of a transistor amplifier

$A_i = - h_f / (1+ Z_L h_o)$	$A_v = A_i Z_L / Z_i$
$Z_i = h_i + h_r A_1 Z_L = h_i - h_f h_r / (Y_L + h_o)$	$A_{vs} = A_v Z_i / (Z_i + R_S) = A_i Z_L / (Z_i + R_S)$ $= A_{is} Z_L / R_S$
$Y_o = h_o - h_f h_r/(R_S + h_i) = 1/ Z_o$	$A_{is} = A_i R_S / (R_S + Z_i) = A_{vs} = A_{is}$ R_S/ Z_L

MODULE-III
NUMBER SYSTEMS

Binary

The ancient Indian writer Pingala developed advanced mathematical concepts for describing prosody, and in doing so presented the first known description of a binary numeral system. A full set of 8 trigrams and 64 hexagrams, analogous to the 3-bit and 6-bit binary numerals, were known to the ancient Chinese in the classic text I Ching. An arrangement of the hexagrams of the I Ching, ordered according to the values of the corresponding binary numbers (from 0 to 63), and a method for generating the same, was developed by the Chinese scholar and philosopher Shao Yong in the 11th century.

In 1854, British mathematician George Boole published a landmark paper detailing an algebraic system of logic that would become known as Boolean algebra. His logical calculus was to become instrumental in the design of digital electronic circuitry. In 1937, Claude Shannon produced his master's thesis at MIT that implemented Boolean algebra and binary arithmetic using electronic relays and switches for the first time in history. Entitled *A* Symbolic Analysis of Relay and Switching Circuits, Shannon's thesis essentially founded practical digital circuit design.

Binary codes

Binary codes are codes which are represented in binary system with modification from the original ones.

- Weighted Binary codes
- Non Weighted Codes

Weighted binary codes are those which obey the positional weighting principles, each position of the number represents a specific weight. The binary counting sequence is an example.

Decimal	BCD 8421	Excess-3	84-2-1	2421	5211	Bi-Quinary 5043210		5	0	4	3	2	1	0
0	0000	0011	0000	0000	0000	0100001	0		X					X
1	0001	0100	0111	0001	0001	0100010	1		X				X	
2	0010	0101	0110	0010	0011	0100100	2		X			X		
3	0011	0110	0101	0011	0101	0101000	3		X		X			
4	0100	0111	0100	0100	0111	0110000	4		X	X				
5	0101	1000	1011	1011	1000	1000001	5	X						X
6	0110	1001	1010	1100	1010	1000010	6	X					X	
7	0111	1010	1001	1101	1100	1000100	7	X				X		
8	1000	1011	1000	1110	1110	1001000	8	X			X			
9	1001	1111	1111	1111	1111	1010000	9	X		X				

Reflective Code

A code is said to be reflective when code for 9 is complement for the code for 0, and so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

Sequential Codes

A code is said to be sequential when two subsequent codes, seen as numbers in binary Representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

Non weighted codes

Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value. Ex: Excess-3 code

Excess-3 Code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3).

Gray Code

  The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. The gray code is a reflective digital code which has the special property that any two subsequent numbers codes differ by only one bit. This is also called a unitdistance code. In digital Gray code has got a special place.

Binary to Gray Conversion

- Gray Code MSB is binary code MSB.
- Gray Code MSB-1 is the XOR of binary code MSB and MSB-1.
- MSB-2 bit of gray code is XOR of MSB-1 and MSB-2 bit of binary code.
- MSB-N bit of gray code is XOR of MSB-N-1 and MSB-N bit of binary code.

A code is said to be sequential when two subsequent codes, seen as numbers in binary Representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

Decimal Number	Binary Code	Gray Code	Decimal Number	Binary Code	Gray Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Error detection codes

1) Parity bits

A **parity bit** is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

2) Checksums

A checksum of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a one's-complement prior to transmission to detect errors resulting in all-zero messages. Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers. The sum may be negated by means of a one's-complement prior to transmission to detect errors resulting in all-zero messages. Checksum schemes include parity bits and check digits.

Error detection codes

3) Parity bits

A **parity bit** is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

4) Checksums

A **checksum** of a message is a modular arithmetic sum of message code words of a fixed word length (e.g., byte values). The sum may be negated by means of a one's-complement prior to transmission to detect errors resulting in all-zero messages. Checksum schemes include parity bits, check digits, and longitudinal redundancy checks. Some checksum schemes, such as the Luhn algorithm and the Verhoeff algorithm, are specifically designed to detect errors commonly introduced by humans in writing down or remembering identification numbers.

5) Cyclic redundancy checks (CRCs)

A **cyclic redundancy check (CRC)** is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a so-called *generator polynomial*, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result.Cyclic codes have favorable properties in that they are well suited for detecting burst errors. CRCs are particularly easy to implement in hardware, and are therefore commonly used in digital networks and storage devices such as hard disk drives.Even parity is a special case of a cyclic redundancy check, where the single-bit CRC is generated by the divisor $x+1$.

Number Base Conversions

The ancient Indian writer Pingala developed advanced mathematical concepts for describing prosody, and in doing so presented the first known description of a binary numeral system. A full set of 8 trigrams and 64 hexagrams, analogous to the 3-bit and 6-bit binary numerals, were known to the ancient Chinese in the classic text I Ching. An arrangement of the hexagrams of the I Ching, ordered according to the values of the corresponding binary numbers (from 0 to 63), and a method for generating the same, was developed by the Chinese scholar and philosopher Shao Yong in the 11th century.

In 1937, Claude Shannon produced his master's thesis at MIT that implemented Boolean algebra and binary arithmetic using electronic relays and switches for the first time in history. Entitled *A Symbolic Analysis of Relay and Switching Circuits*, Shannon's thesis essentially founded practical digital circuit design.

Any number in one base system can be converted into another base system Types

1) decimal to any base

2) Any base to decimal

3) Any base to Any base

Decimal number: $123.45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$

Base b number: $N = a_{q-1}b^{q-1} + \cdots + a_0 b^0 + \cdots + a_{-p}b^{-p}$
$b > 1$, $0 <= a_i <= b-1$
Integer part: $a_{q-1}a_{q-2} \cdots a_0$
Fractional part: $a_{-1}a_{-2} \cdots a_{-p}$
Most significant digit: a_{q-1} . . .
Least significant digit: a_{-p}

Binary number ($b=2$): $1101.01 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}$

Representing number N in base b: $(N)_b$

Complement of digit a: $a' = (b-1)-a$
Decimal system: 9's complement of $3 = 9-3 = 6$
Binary system: 1's complement of $1 = 1-1 = 0$

Fractional number:

$$(N)_{b_1} = a_{-1}b_2^{-1} + a_{-2}b_2^{-2} + \cdots + a_{-p}b_2^{-p}$$

$$b_2 \cdot (N)_{b_1} = a_{-1} + a_{-2}b_2^{-1} + \cdots + a_{-p}b_2^{-p+1}$$

Example: Convert $(0.3125)_{10}$ to base 8
$0.3125 \cdot 8 = 2.5000$ hence $a_{-1} = 2$
$0.5000 \cdot 8 = 4.0000$ hence $a_{-2} = 4$

Thus, $(0.3125)_{10} = (0.24)_8$

A **parity bit** is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) in the outcome is even or odd. It is a very simple scheme that can be used to detect single or any other odd number (i.e., three, five, etc.) of errors in the output. An even number of flipped bits will make the parity bit appear correct even though the data is erroneous.

Decimal to Binary

Example: Convert $(432.354)_{10}$ to binary

Q_i	r_i
216	$0 = a_0$
108	$0 = a_1$
54	$0 = a_2$
27	$0 = a_3$
13	$1 = a_4$
6	$1 = a_5$
3	$0 = a_6$
1	$1 = a_7$
	$1 = a_8$

0.354 · 2 = 0.708 hence $a_{-1} = 0$
0.708 · 2 = 1.416 hence $a_{-2} = 1$
0.416 · 2 = 0.832 hence $a_{-3} = 0$
0.832 · 2 = 1.664 hence $a_{-4} = 1$
0.664 · 2 = 1.328 hence $a_{-5} = 1$
0.328 · 2 = 0.656 hence $a_{-6} = 0$
. $a_{-7} = 1$
                                     etc.

Thus, $(432.354)_{10} = (110110000.0101101...)_2$

Octal To Binary

Example: Convert $(123.4)_8$ to binary
$$(123.4)_8 = (001\ 010\ 011.100)_2$$

Example: Convert $(1010110.0101)_2$ to octal
$$(1010110.0101)_2 = (001\ 010\ 110.010\ 100)_2 = (126.24)_8$$

Error Detection and Correction Codes:
- No communication channel or storage device is completely error-free

- As the number of bits per area or the transmission rate increases, more errors occur. • Impossible to detect or correct 100% of the errors

Hamming Codes

1. One of the most effective codes for error-recovery

2. Used in situations where random errors are likely to occur

3. Error detection and correction increases in proportion to the number of parity
   bits (error- checking bits) added to the end of the information bits
   code word = information bits + parity bits

   Hamming distance: the number of bit positions in which two code words differ.
   >   10001001
   >   10110001

   Minimum Hamming distance or D(min) : determines its error detecting and correcting
   capability.

4. Hamming codes can always detect D(min) – 1 errors, but can only correct half of those
   errors.

```
EX.  Data          Parity Code
     Bits          Bit    Word
     00            0      000
     01            1      011
     10            1      101
     11            0      110
                                    000* 100
                                    001   101*
                                    010   110*
                          |         011* 111
```

5. Single parity bit can only detect error, not correct it
6. Error-correcting codes require more than a single
   paritybit EX. 0 0 0 0 0
   0 1 0 1 1

   1 0 1 1 0
   1 1 1 0 1

Minimum Hamming distance = 3

Can detect up to 2 errors and correct 1 error Cyclic Redundancy Check

1. Let the information byte F = 1001011

2. The sender and receiver agree on an arbitrary binary pattern P. Let P = 1011.

3. Shift F to the left by 1 less than the number of bits in P. Now, F = 1001011000.

4. Let F be the dividend and P be the divisor. Perform "modulo 2 division".

5. After performing the division, we ignore the quotient. We got 100 for the remainder, which becomes the actual CRC checksum.

6. Add the remainder to F, giving the message M: 1001011 + 100 =

M is decoded and checked by the message receiver using the reverse process.

```
            1010100
1011 |  1001011100
        1011
        001001
          1001
          0010
          001011
            1011
            0000          ← Remainder
```

1001011100 = M

Boolean Algebra And Theorems

- Fundamental postulates of Boolean algebra
- Basic theorems and properties
- Switching functions
- Canonical and Standard forms
- Algebraic simplification digital logic gates, properties of XOR gates
- Universal gates
- Multilevel NAND/NOR realizations

Boolean algebra: Boolean algebra, like any other deductive mathematical system, may be defined with asset of elements, a set of operators, and a number of unproved axioms or postulates. A *set* of elements is any collection of objects having a common property. If **S** is a set and *x* and *y* are certain objects, then **x Î D**enotes that *x* is a member of the set **S**, and *y* ï**S**

denotes that **y** is not an element of **S**. A set with adenumerable number of elements is specified by braces: **A** = {1,2,3,4}, *i.e.* the elements of set **A** are thenumbers 1, 2, 3, and 4. A *binary operator* defined on a set S of elements is a rule that assigns to each pair ofelements from S a unique element from S._ Example: In *a*b=c*, we say that * is a binary operator if it specifies a rule for finding *c* from the pair (*a,b*)and also if *a*, *b*, *c* Î S.

CLOSURE: The Boolean system is *closed* with respect to a binary operator if for every pair of Boolean values,it produces a Boolean result. For example, logical AND is closed in the Boolean system because it accepts only Boolean operands and produces only Boolean results.
A set *S* is closed with respect to a binary operator if, for every pair of elements of *S*, the binary operator specifies a rule for obtaining a unique element of *S*.
For example, the set of natural numbers N = {1, 2, 3, 4, … 9} is closed with respect to the binary operator plus (+) by the rule of arithmetic addition, since for any *a*, *b* Î N we obtain a unique *c* Î N by the operation *a + b* = c.

ASSOCIATIVE LAW:

A binary operator * on a set *S* is said to be associative whenever (*x * y*) * *z* = *x* * (*y * z*) for all *x*, *y*, *z* Î S, forall Boolean values x, y and z.

COMMUTATIVE LAW:

A binary operator * on a set *S* is said to be commutative whenever x * *y* = *y* * *x* for all *x*, *y*, *z* ϵ S
IDENTITY ELEMENT:
A set *S* is said to have an identity element with respect to a binary operation * on S if
there exists an element *e* ϵ S with the property *e* * *x* = *x* * *e* = *x* for every *x* ϵ S

BASIC IDENTITIES OF BOOLEAN ALGEBRA

- *Postulate 1(Definition)*: A Boolean algebra is a closed algebraic system containing a set *K* of two or more elements and the two operators · and + which refer to logical AND and logical OR •*x + 0 = x*

- *x · 0 = 0*

- *x + 1 = 1*

- *x · 1 = 1*

- *x + x = x*

- *x · x = x*

- *x + x' = x*

- *x · x' = 0*

- *x + y = y + x*

- $x \cdot x = x$

- $x + x' = x$

- $x \cdot x' = 0$

- $xy = yx$

- $x + (y + z) = (x + y) + z$

- $x (yz) = (xy) z$

- $x (y + z) = xy + xz$

- $x + yz = (x + y)(x + z)$

- $(x + y)' = x' y'$

- $(xy)' = x' + y'$

- $(x')' = x$

- $x (y + z) = xy + xz$

- $x + yz = (x + y)(x + z)$

- $x \cdot 0 = 0$

- $x + 1 = 1$

- $x \cdot 1 = 1$

-

DeMorgan's Theorem

(a) $(a + b)' = a'b'$

(b) $(ab)' = a' + b'$
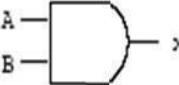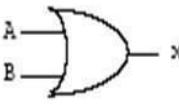
Generalized DeMorgan's

Theorem   (a) $(a + b + ... z)' = a'b' ... z'$
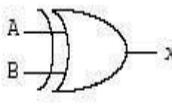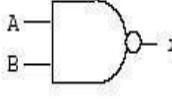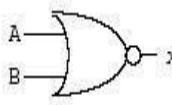
(b) $(a.b ... z)' = a' + b' + ... z'$

Logic Gates

- **Formal logic:** In formal logic, a statement (proposition) is a declarative sentence that is either true(1) or false (0). It is easier to communicate with computers using formal logic.

- **Boolean variable:** Takes only two values – either true (1) or false (0). They are used as basic units of formal logic.

- **Boolean algebra:** Deals with binary variables and logic operations operating on those variables.

- **Logic diagram:** Composed of graphic symbols for logic gates. A simple circuit sketch that represents inputs and outputs of Boolean functions.

Name	Graphic symbol	Algebraic function	Truth table	
Inverter	A —▷o— x	$x = A'$	A|x 0|1 1|0	
AND	A, B —[ ]— x	$x = AB$	A B|x 0 0|0 0 1|0 1 0|0 1 1|1	True if both are true.
OR	A, B —[ ]— x	$x = A + B$	A B|x 0 0|0 0 1|1 1 0|1 1 1|1	True if either one is true.

- Other common gates include:

Name	Graphic symbol	Algebraic function	Truth table	
Exclusive-OR (XOR)	A, B — x	$x = A \oplus B$ $= A'B + AB'$	A B|x 0 0|0 0 1|1 1 0|1 1 1|0	Parity check: True if only one is true.
NAND	A, B —o x	$x = (AB)'$	A B|x 0 0|1 0 1|1 1 0|1 1 1|0	Inversion of AND.
NOR	A, B —o x	$x = \overline{A + B}$	A B|x 0 0|1 0 1|0 1 0|0 1 1|0	Inversion of OR.

Minimization of switching functions is to obtain logic circuits with least circuit complexity. This goal is very difficult since how a minimal function relates to the implementation technology is important. For example, If we are building a logic circuit that uses discreet logic made of small scale Integration ICs(SSIs) like 7400 series, in which basic building block are constructed and are available for use. The goal of minimization would be to reduce the number of ICs and not the logic gates. For example, If we require two 6 and gates and 5 Or gates, we would require 2 AND ICs (each has 4 AND gates) and one OR IC. (4 gates). On the other hand if the same logic could be implemented with only 10 nand gates, we require only 3 ICs. Similarly when we design logic on Programmable device, we may implement the design with certain number of gates and remaining gates may not be used.

Multilevel Implementation Using Nand And Nor Gate:

Two-level implementation means that any path from input to output contains maximum two gates hence the name two-level for the two levels of gates.

Implementing Two-Level logic using NOR gate requires the Boolean expression to be in Product of Sum (POS) form.

In Product of Sum form, 1ˢᵗ level of the gate is OR gate and 2ⁿᵈ level of the gate is AND gate.

To implement a Boolean function using NOR gate, there are basically three step;
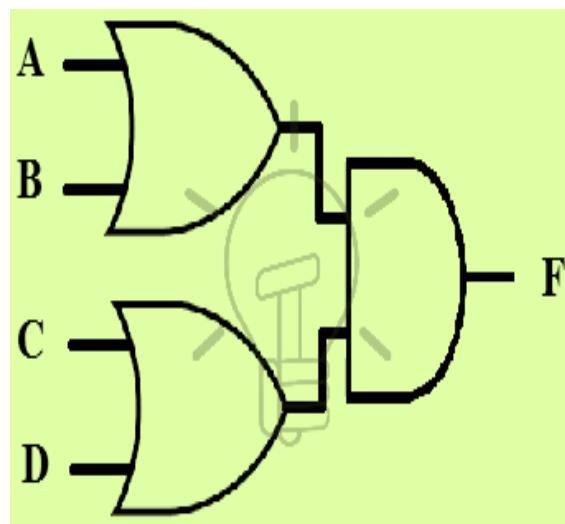
Product of Sum Form

First, you need to have a simplified Product of Sum expression for the function you need to implement.

Simplified Product of Sum expression can be made using Karnaugh Map (K-map) by combining the '0's and then inverting the output function.
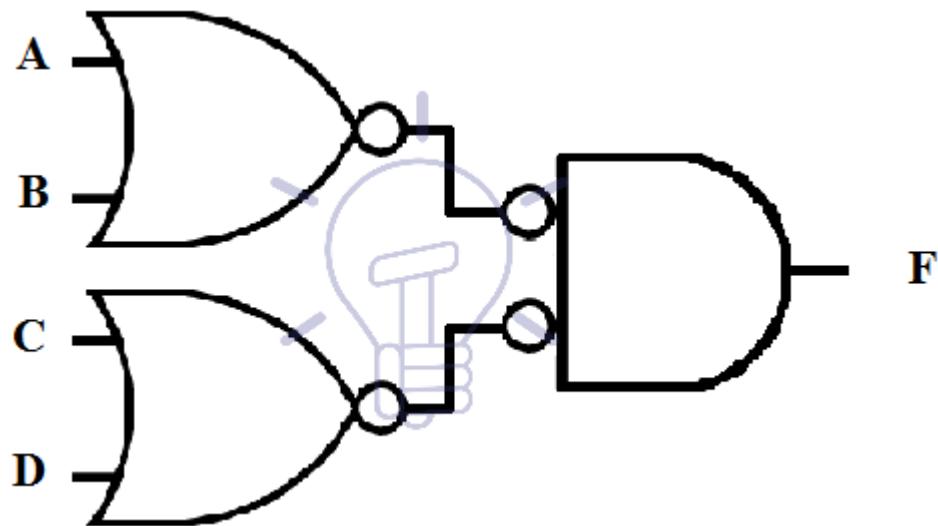
F = (A + B) (C + D)

Draw its schematic using AND-OR NOT gates as shown in the figure given below.

Mixed Notation
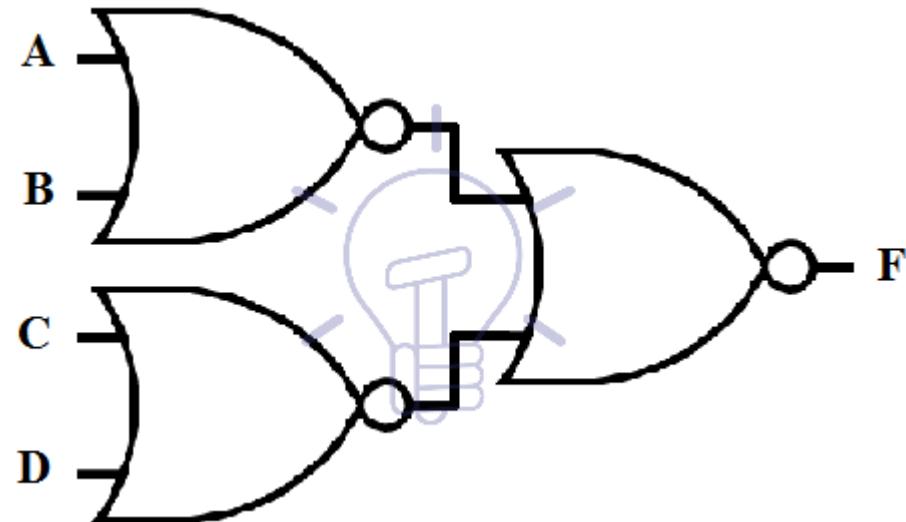
Next step is to draw the above-mentioned schematic using OR-Invert and Invert-AND gates. OR-Invert should replace OR gates and invert-AND replaces AND gates. This schematic is said to be in mixed notation and its schematic is given below.

A bubble means complement. Two bubbles along a line mean double complementation and they cancel each other. However, a single bubble along a line should be compensated by inserting an Inverter in that line or if it is an input line then you can also feed a complemented input if available.
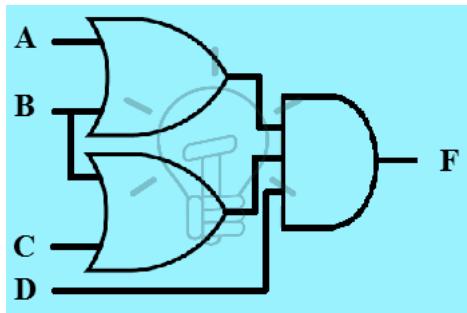
NOR Gate Conversion

The last step is to redraw the whole schematic replacing OR-Invert and Invert-AND gate symbol by NOR gate symbol because OR-Invert and Invert-AND are equivalent to NOR gate. The final schematic is shown in the figure given below.
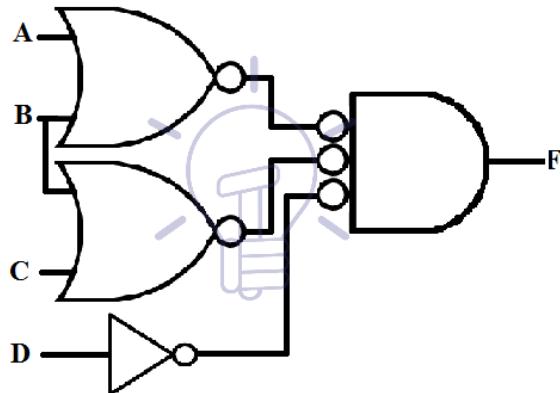
F = (A + B) (B + C) D

This function is in simplified Product of Sum form. First, we need to draw its OR-AND schematic.
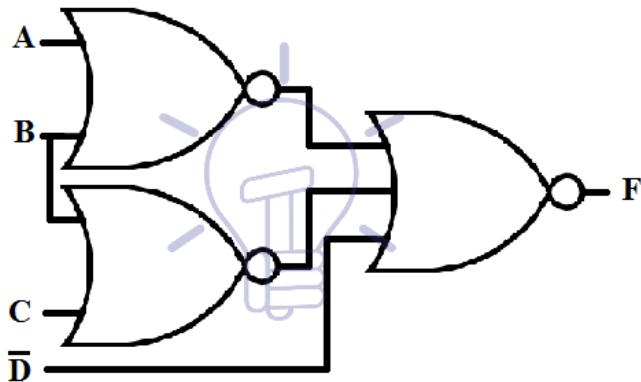
Now we convert the above-given schematic into mixed notation by converting OR gate into OR-INVERT and AND gate into INVERT-AND.

Input line D to the input of AND gate has a single bubble.To compensate this bubble we need to either insert an inverter in this line or complement the input D if available.

Now replace every OR-Invert and Invert-AND with NOR gate as shown in the figure given below.
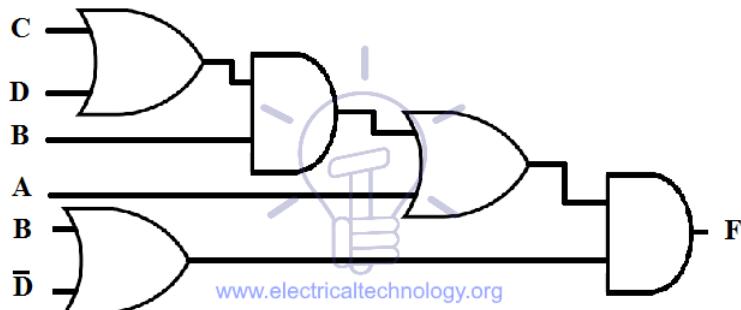
Multi level implementation using NOR Gate

Schematic having more than two levels of gates is known as a multi-level schematic.We can implement multi-level POS expression using NOR gate. The conversion of multi-level expression into NOR gate has the same method as two-level implementation. The multi-level expression can be converted into two-level expression but for the sake of realization, we will implement a multi-level expression. Suppose a 4-level function:

F = (A + B (C + D)) (B + D')
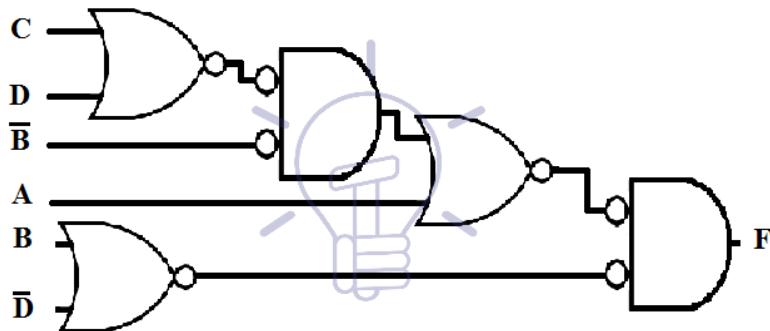
First, we will draw its schematic using AND, OR, NOT gates.
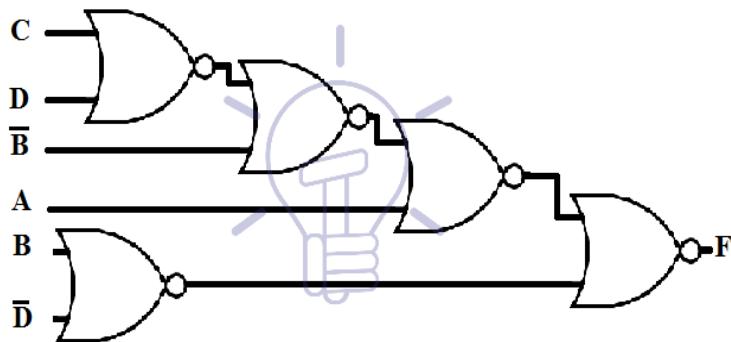
Multi-level NOR Gate Implementation

Notice the OR-AND pattern like two-level implementation. It can be easily converted since the bubble cancels each other.

Now we will convert it into mixed notation for NOR.

The two bubbles along a single line cancel each other. However, there is a single bubble at the 2ndlevel gate's input. so we will complement the input B to compensate the bubble. Now redraw the whole schematic replacing OR-Invert and Invert-AND with NOR gate symbol as shown in the figure below.
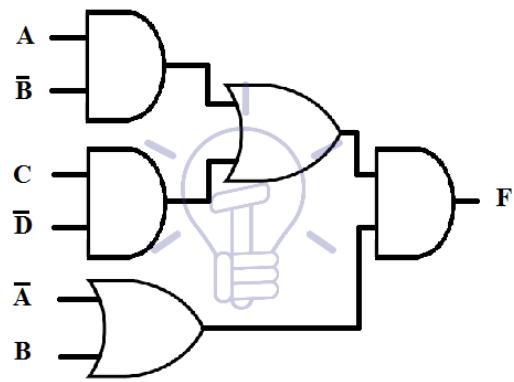
3-Level Implementation & Example using NOR Gate

A 3-level implementation using NOR gate's Example is given below;
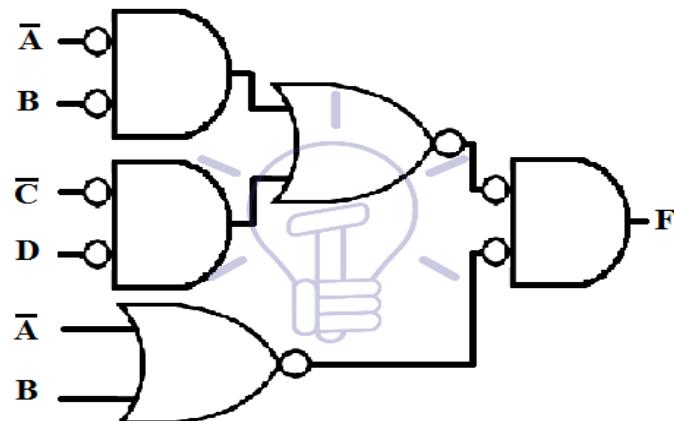
F = (AB' + CD') (A' + B)

First, we will draw its schematic using AND,OR,NOT gates as given in the figure below.
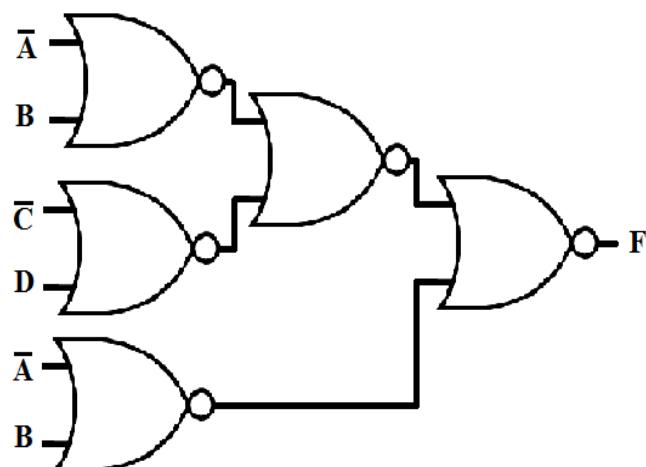
3-Level NOR Gate Implementation

Now we will convert it into mixed notation for NOR.

The single bubbles at the input line of all first level gates need an inverter or the inputs to be complimented. The two bubbles along the same line cancel each other. Now that all the bubbles have been accounted for, we will redraw this schematic by replacing OR-Invert and Invert-AND with NOR gates as shown in the figure below.

74
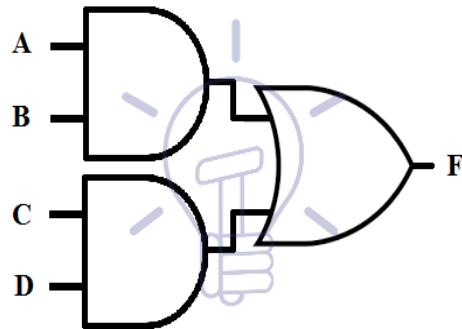
NAND Gate:

NAND Gate is a universal logic gate which means any Boolean logic can be implemented using NAND gate including individual logic gates. In other words, any kind of Boolean function can be implemented using only NAND gates.

NAND gate is commercially used because it allows the access to wired logic which is a logic function formed by connecting the outputs of NAND gates. Wired logic does not consist of a physical gate but the wires behave as a logic function. The other reason for commercial usage of NAND gate is that it can be easily fabricated and has a low fabrication cost. It also shrinks the schematic by decreasing the number of gates, which results in small size and as mall delay, fast speed and Low power consumption.

As we know a typical Boolean function implementation consists of AND, OR and NOT gates. To implement a whole Boolean function using NAND gate first, we need to convert these gates into NAND gate.
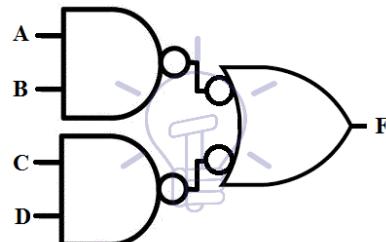
Related Articles:

Two-Level Implementation

Mixed Notation

2nd step is to convert the AND-OR schematic into mixed notation. In mixed notation for NAND gate, AND gate is converted into AND-invert and OR gate is converted into INVERT-OR. Mixed notation design for the above function is given below.

Mixed Notation for NAND Gate

MODULE-IV
MINIMIZATION OF BOOLEAN FUNCTIONS

Introduction

Minimization of switching functions is to obtain logic circuits with least circuit complexity. This goal is very difficult since how a minimal function relates to the implementation technology is important. For example, If we are building a logic circuit that uses discrete logic made of small scale Integration ICs(SSIs) like 7400 series, in which basic building block are constructed and are available for use. The goal of minimization would be to reduce the number of ICs and not the logic gates. For example, If we require two 6 and gates and 5 Or gates,we would require 2 AND ICs(each has 4 AND gates) and one OR IC. (4 gates). On the other hand if the same logic could be implemented with only 10 nand gates, we require only 3 ICs. Similarly when we design logic on Programmable device, we may implement the design with certain number of gates and remaining gates may not be used.

 Whatever may be the criteria of minimization we would be guided by the following:

- Boolean algebra helps us simplify expressions and circuits
- Karnaugh Map: A graphical technique for simplifying a Boolean expression into either form:
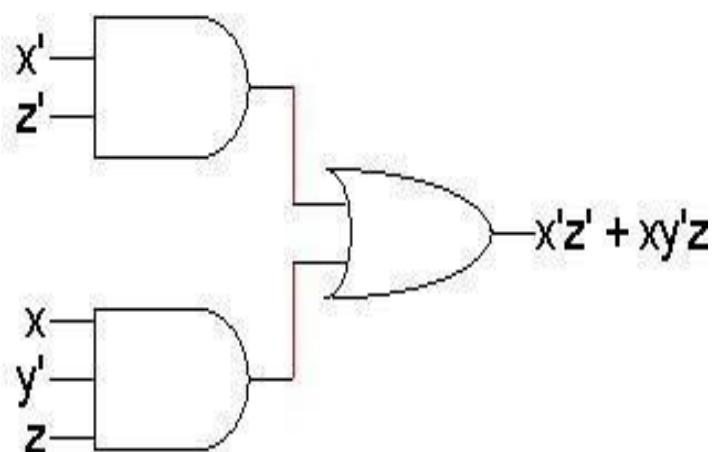
    minimal sum of products
    (MSP) o   minimal
    product of sums (MPS)
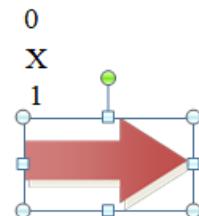
- Goal of the simplification.

    Circuit-wise, this leads to a *minimal* two-levelimplementation
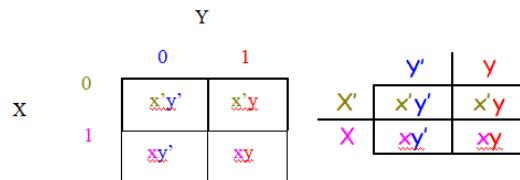
- A two-variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map

x	Y	Minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	Xy
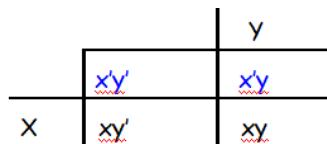
	0	1
	x'y'	x'y
	xy'	xy

- Now we can easily see which minterms contain common literals
  - Minterms on the left and right sides contain y' and y respectively
  - Minterms in the top and bottom rows contain x' and x respectively

	Y=0	Y=1
X=0	x'y'	x'y
X=1	xy'	xy

	y'	y
X'	x'y'	x'y
X	xy'	xy

K- map Simplification

- Imagine a two-variable sum of minterms x'y' + x'y
- Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x'

	y'	y
	x'y'	x'y
X	xy'	xy

- What happens if you simplify this expression using Boolean algebra?
- x'y' + x'y= x'(y' + y) [Distributive]

  = x'+1   [y + y' = 1]

  = x'   [x + 1 = x]

	Y	
	0	1
X 0	x'y'	x'y
1	xy'	xy

	y'	y
X'	x'y'	x'y
X	xy'	xy

K- map Simplification

- Imagine a two-variable sum of minterms x'y' + x'y

- Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x'

	y	
	x'y'	x'y
X	xy'	xy

- What happens if you simplify this expression using Boolean algebra?

- x'y' + x'y = x'(y' + y) [Distributive]

   = x'+1 [y + y' = 1]

   = x'   [x + 1 = x]

A Three-Variable Karnaugh Ma

- For a three-variable expression with inputs x, y, z, the arrangement of minterms is more tricky:

		YZ		
	00	01	11	10
X 0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

		YZ		
	00	01	11	10
X 0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

- Another way to label the K-map (use whichever you like):

		Y		
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

		Y		
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

- With this ordering, any group of 2, 4 or 8 adjacent squares on the map contains common literals that can be factored out

		Y		
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

$$x'y'z + x'yz$$
$$= x'z(y' + y)$$
$$= x'z \cdot 1$$
$$= x'z$$

- "Adjacency" includes wrapping around the left and right sides:

		Y		
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		
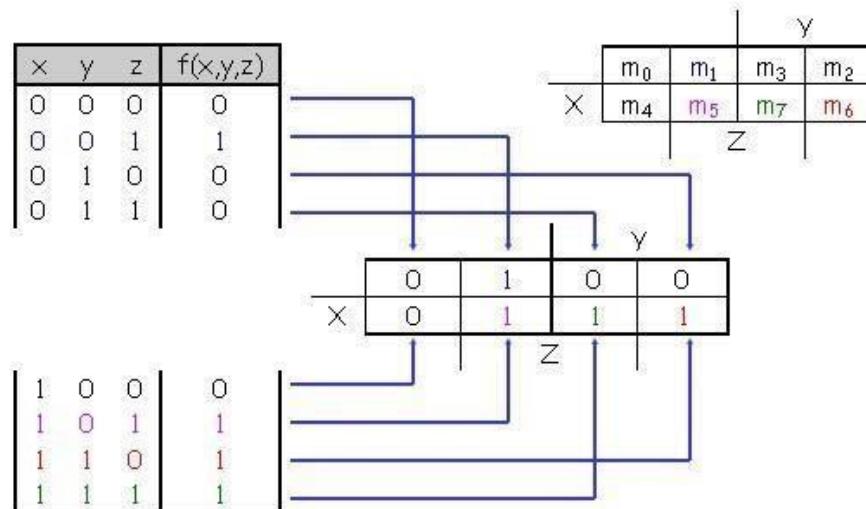
$$x'y'z' + xy'z' + x'yz' + xyz'$$
$$= z'(x'y' + xy' + x'y + xy)$$
$$= z'(y'(x' + x) + y(x' + x))$$
$$= z'(y'+y)$$
$$= z'$$

- We'll use this property of adjacent squares to do our simplifications.
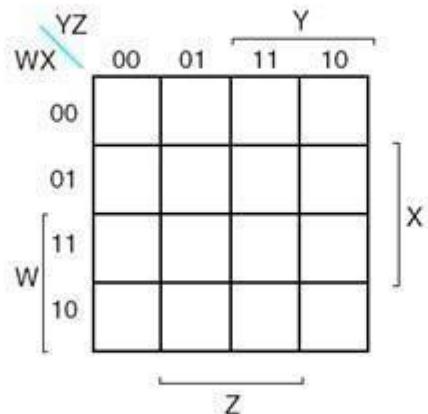
K-Maps From Truth Tables

- We can fill in the K-map directly from a truth table
  - The output in row *i* of the table goes into square m_i of the K-map
  - Remember that the rightmost columns of the K-map are "switched"

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

		Y		
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

		Y		
	0	1	0	0
X	0	1	1	1
		Z		
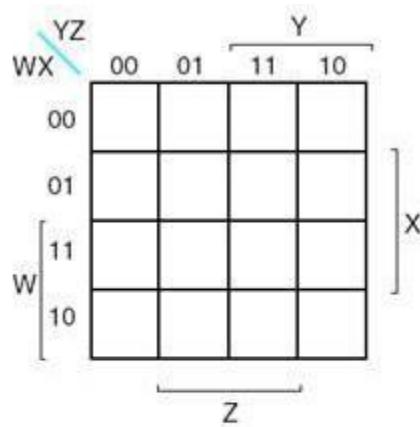
Four-Variable K-Maps – F(W,X,Y,Z)

- We can do four-variable expressions too!
  - The minterms in the third and fourth columns, *and* in the third and fourth rows, are switched around.
  - Again, this ensures that adjacent squares have common literals
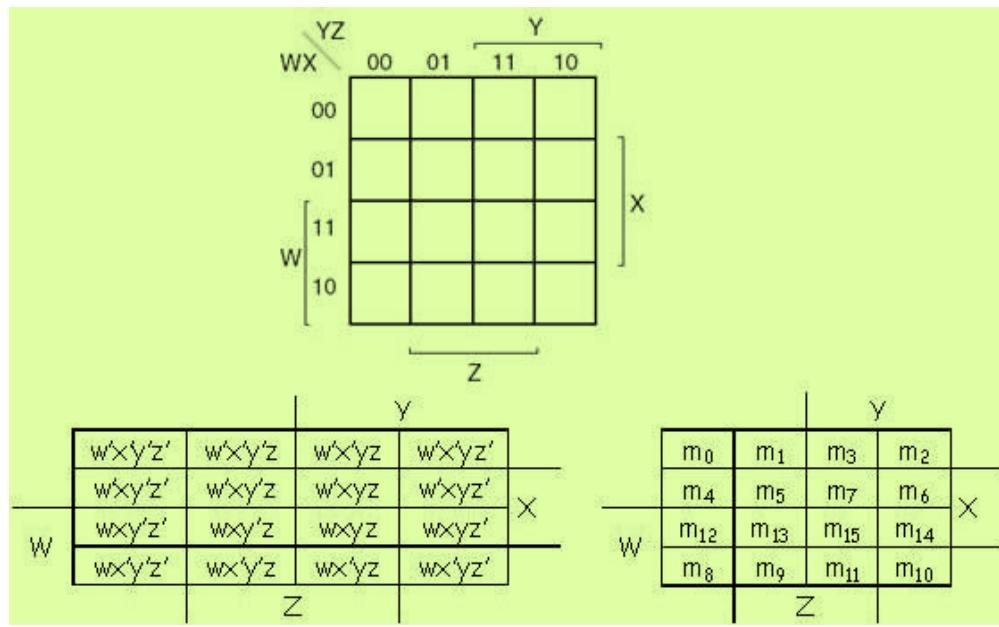
- Grouping minterms is similar to the three-variable case, but:
  - You can have rectangular groups of 1, 2, 4, 8 or 16 minterms
  - You can wrap around *all four* sides
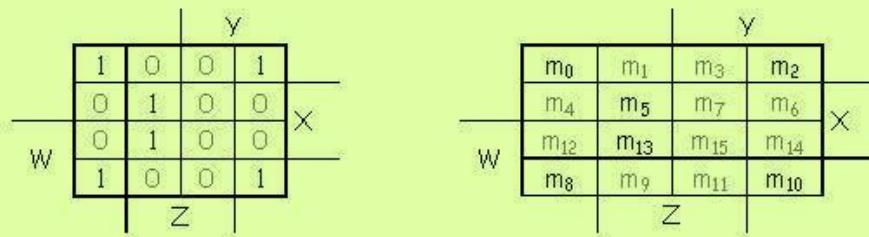
$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

YZ / WX

WX \ YZ	00	01	11	10
00				
01				
11				
10				

X, W, Z, Y labels

w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
w'xy'z'	w'xy'z	w'xyz	w'xyz'
wxy'z'	wxy'z	wxyz	wxyz'
wx'y'z'	wx'y'z	wx'yz	wx'yz'
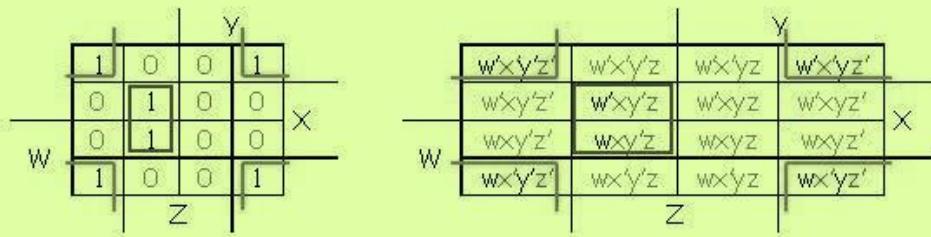
m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

- The expression is already a sum of minterms, so here's the K-map:

1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

- We can make the following groups, resulting in the MSP $x'z' + xy'z$

1	0	0	1
0	1	0	0
0	1	0	0
1	0	0	1

w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
w'xy'z'	w'xy'z	w'xyz	w'xyz'
wxy'z'	wxy'z	wxyz	wxyz'
wx'y'z'	wx'y'z	wx'yz	wx'yz'

Simplify m0+m2+m5+m8+m10+m13
PoS Optimization

- Maxterms are grouped to find minimal PoS expression

x \ yz	00	01	11	10
0	x+y+z	x+y+z'	x+y'+z'	x+y'+z
1	x'+y+z	x'+y+z'	x'+y'+z'	x'+y'+z

- $F(W,X,Y,Z) = \prod M(0,1,2,4,5)$

	00	01	11	10
0	x +y+z	x+y+z'	x+y'+z'	x+y'+z
1	x' +y+z	x'+y+z'	x'+y'+z'	x'+y'+z

yz / x

$F(W,X,Y,Z) = Y \cdot (X + Z)$

	00	01	11	10
0	0	0	1	0
1	0	0	1	1

yz / x

PoS Optimization from SoP

$F(W,X,Y,Z) = \Sigma m(0,1,2,5,8,9,10)$
$\qquad = \prod M(3,4,6,7,11,12,13,14,15)$

YZ / WX

WX\YZ	00	01	11	10
00			0	
01	0		0	0
11	0	0	0	0
10			0	
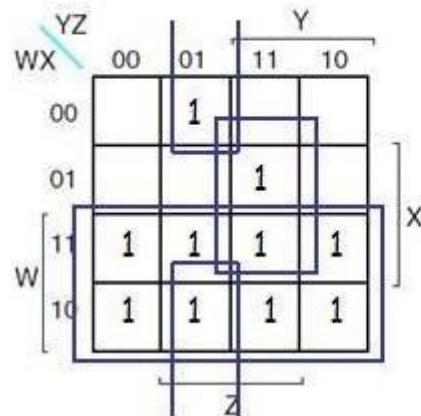
$F(W,X,Y,Z) = (W' + X')(Y' + Z')(X' + Z)$

Or,

$F(W,X,Y,Z) = X'Y' + X'Z' + W'Y'Z$

Which one is the minimal one?

SoP Optimization from PoS

$$F(W,X,Y,Z) = \prod M(0,2,3,4,5,6)$$
$$= \Sigma m(1,7,8,9,10,11,12,13,14,15)$$

$$F(W,X,Y,Z) = W + XYZ + X'Y'Z$$

Don't care

- You don't always need all 2^n input combinations in an n-variable function
  - If you can guarantee that certain input combinations never occur
  - If some outputs aren't used in the rest of the circuit
- We mark don't-care outputs in truth tables and K-maps with Xs.

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	X
1	1	1	1

- Within a K-map, each X can be considered as either 0 or 1. You should pick the interpretation that allows for the most simplification.

- Find a MSP for

$$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15), \, d(w,x,y,z) = \Sigma m(7,10,13)$$

This notation means that input combinations $wxyz = 0111, 1010$ and 1101 (corresponding to minterms m_7, m_{10} and m_{13}) are unused.

5-Variable K-Map

In above boolean table, from 0 to 15, A is 0 and from 16 to 31, A is 1. A 5-variable K-Map is drawn as below.
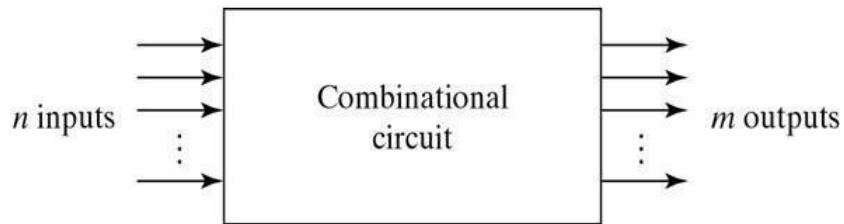
Again, as we did with 3-variable & 4-variable K-Map, carefully note the numbering of each cell. Now, we have two squares and we can loop octets, quads and pairs between these two squares. What we need to do is to visualize second square on first square and figure out adjacent cells. Let's understand how to simplify 5-variables K-Map by taking couple of examples.

Design Of Combinational Circuits
Combinational Logic

- Logic circuits for digital systems may be combinational or sequential.
- A combinational circuit consists of input variables, logic gates, and output variables.

For n input variables,there are 2^n possible combinations of binary input variables .For each possible input Combination ,there is one and only one possible output combination.A combinational circuit can be described by m Boolean functions one for each output variables.Usually the input s comes from flip-flops and outputs goto flip-flops.

Design Procedure:

1. The problem is stated
2. The number of available input variables and required output variables is determined.
3. The input and output variables are assigned lettersymbols.
4. The truth table that defines the required relationship between inputs and outputs is derived.
5. The simplified Boolean function for each output is obtained.
6. The logic diagram is drawn.

Adders:
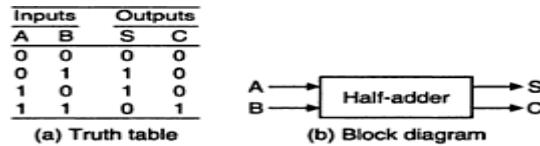
Digital computers perform variety of information processing tasks,the one is arithmetic operations.And the most basic arithmetic operation is the addition of two binary digits.i.e, 4 basic possible operations are:

$$0+0=0,0+1=1,1+0=1,1+1=10$$

The first three operations produce a sum whose length is one digit, but when augends and addend bits are equal to 1,the binary sum consists of two digits.The higher significant bit of this result is

called a carry.A combinational circuit that performs the addition of two bits is called a half- adder. One that performs the addition of 3 bits (two significant bits & previous carry) is called a full adder.& 2 half adder can employ as a full-adder.

The Half Adder: A Half Adder is a combinational circuit with two binary inputs (augends and addend bits and two binary outputs (sum and carry bits.) It adds the two inputs (A and B) and produces the sum (S) and the carry (C) bits. It is an arithmetic operation of addition of two single bit words.

(a) Truth table   (b) Block diagram
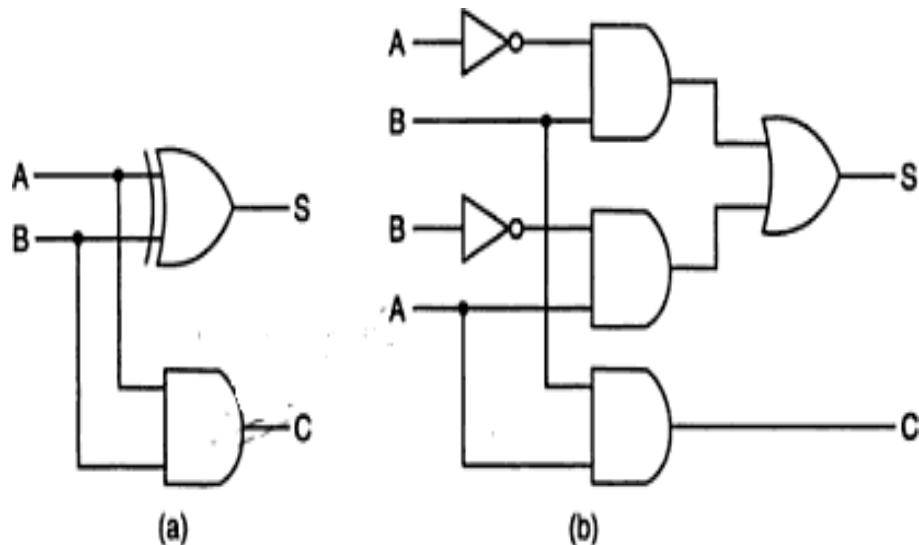
The Sum(S) bit and the carry (C) bit, according to the rules of binary addition, the sum (S) is the X-OR of A and B (It represents the LSB of the sum). Therefore,

$$S=A'□+□B'$$

The carry (C) is the AND of A and B (it is 0 unless both the inputs are 1).Therefore,

$$C=AB$$
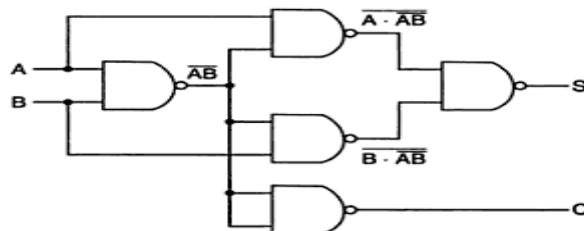
A half-adder can be realized by using one X-OR gate and one AND gate a

(a)   (b)

Logic diagrams of half-adder

NAND Logic:
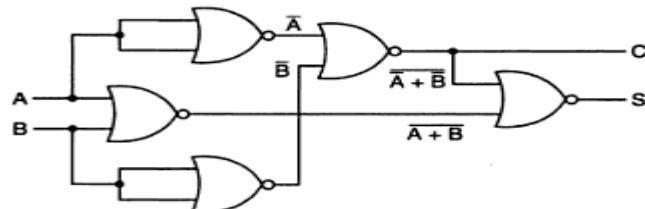
$$S = A\overline{B} + \overline{A}B = A\overline{B} + A\overline{A} + \overline{A}B + B\overline{B}$$
$$= A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B})$$
$$= A \cdot \overline{AB} + B \cdot \overline{AB}$$
$$= \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$
$$C = AB = \overline{\overline{AB}}$$

Logic diagram of a half-adder using only 2-input NAND gates.

NOR Logic:

$$S = A\overline{B} + \overline{A}B = A\overline{B} + A\overline{A} + \overline{A}B + B\overline{B}$$
$$= A(\overline{A} + \overline{B}) + B(\overline{A} + \overline{B})$$

$$= (A + B)(\overline{A} + \overline{B})$$
$$= \overline{\overline{A + B} + \overline{\overline{A} + \overline{B}}}$$
$$C = AB = \overline{\overline{AB}} = \overline{\overline{A} + \overline{B}}$$
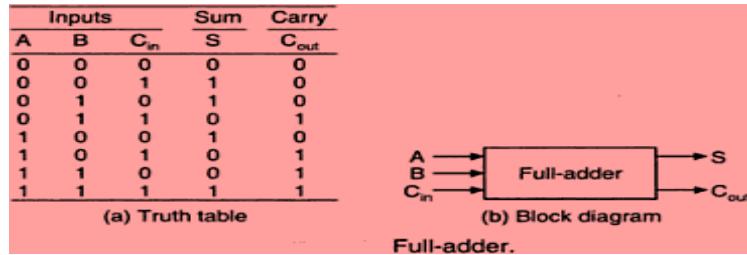
Logic diagram of a half-adder using only 2-input NOR gates.

The Full Adder:

A Full-adder is a combinational circuit that adds two bits and a carry and outputs a sum bit and a carry bit. To add two binary numbers, each having two or more bits, the LSBs can be added by using a half-adder. The carry resulted from the addition of the LSBs is carried over to the next significant column and added to the two bits in that column. So, in the second and higher columns, the two data bits of that column and the carry bit generated from the addition in the previous column need to be added.

The full-adder adds the bits A and B and the carry from the previous column called the carry-in C_{in} and outputs the sum bit S and the carry bit called the carry-out C_{out} . The variable S gives the value of the least significant bit of the sum. The variable C_{out} gives the output carry.The

eight rows under the input variables designate all possible combinations of 1s and 0s that these variables may have. The 1s and 0s for the output variables are determined from the arithmetic sum of the input bits. When all the bits are 0s , the output is 0. The S output is equal to 1 when only 1 input is equal to 1 or when all the inputs are equal to 1. The C_{out} has a carry of 1 if two or three inputs are equal to 1.

Inputs			Sum	Carry
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) Truth table   (b) Block diagram

Full-adder.
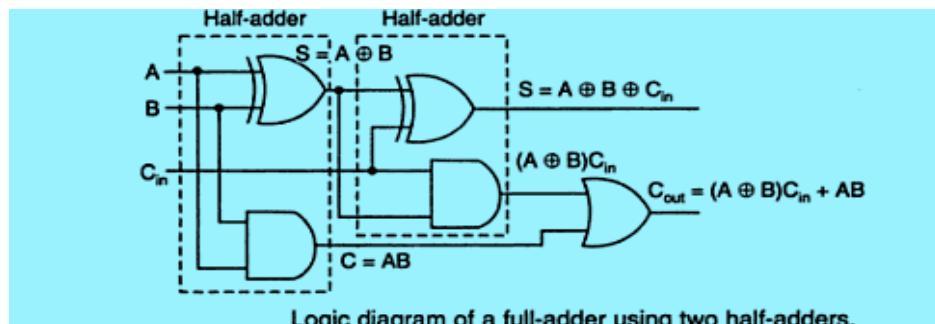
From the truth table, a circuit that will produce the correct sum and carry bits in response to every possible combination of A,B and C_{in} is described by

$$S \square \overline{AB}\overline{C}in \square A\overline{B}\overline{Cin} \square AB\overline{C}in \square$$
$$ABCin\ Cout \square A\underline{B}Cin \square AB\underline{Cin} \square$$
$$ABCin \square ABCin \qquad —$$
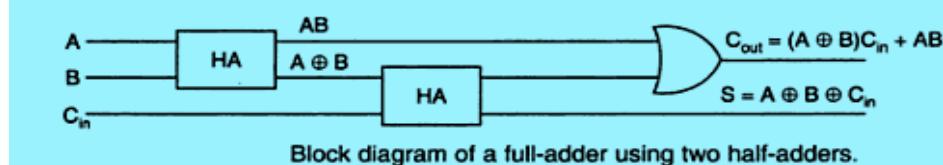
and

$$S \square A \square B \square Cin : Cout \square ACin \square BCin \square AB$$

The sum term of the full-adder is the X-OR of A,B, and C_{in}, i.e, the sum bit the modulo sum of the data bits in that column and the carry from the previous column. The logic diagram of the full-adder using two X-OR gates and two AND gates (i.e, Two half adders) and one OR gate is

Logic diagram of a full-adder using two half-adders.

The block diagram of a full-adder using two half-adders is

Block diagram of a full-adder using two half-adders.

Even though a full-adder can be constructed using two half-adders, the disadvantage is that the bits must propagate through several gates in accession, which makes the total propagation delay greater than that of the full-adder circuit using AOI logic.

The Full-adder neither can also be realized using universal logic, i.e., either only NAND gates or only NOR gates as
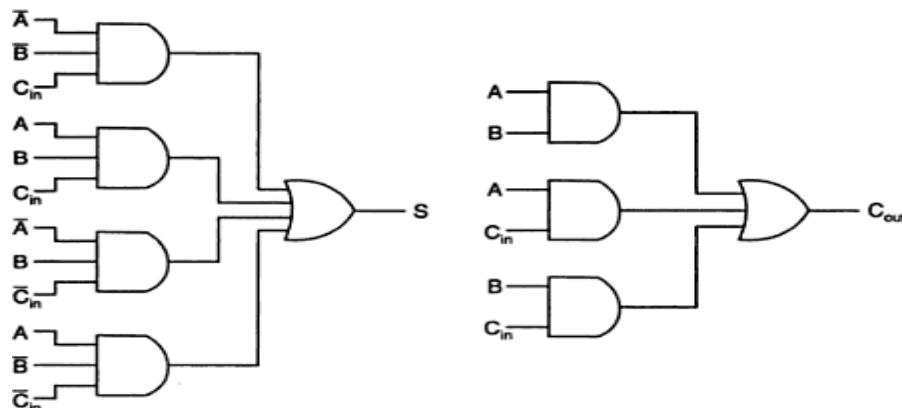
$$A \oplus B = \overline{\overline{A \cdot \overline{AB}} \cdot \overline{B \cdot \overline{AB}}}$$

Then
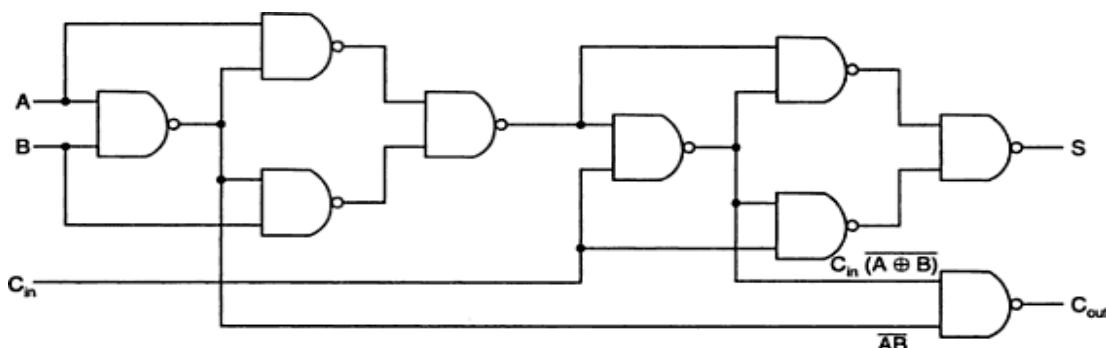
$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) \cdot \overline{(A \oplus B)C_{in}}} \cdot \overline{C_{in} \cdot \overline{(A \oplus B)C_{in}}}}$$

NAND Logic:

$$C_{out} = C_{in}(A \oplus B) + AB = \overline{\overline{C_{in}(A \oplus B)} \cdot \overline{AB}}$$

Sum and carry bits of a full-adder using AOI logic.

Logic diagram of a full-adder using only 2-input NAND gates.

89
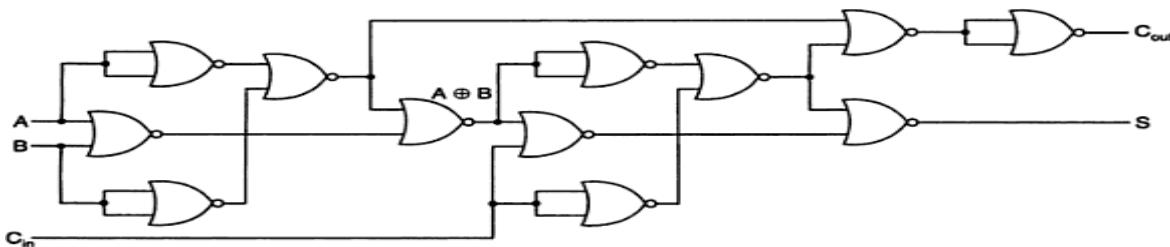
NOR Logic:

Then

$$A \oplus B = \overline{\overline{(A + B)} + \overline{A} + \overline{B}}$$

$$S = A \oplus B \oplus C_{in} = \overline{\overline{(A \oplus B) + C_{in}} + \overline{\overline{(A \oplus B)} + \overline{C_{in}}}}$$

$$C_{out} = AB + C_{in}(A \oplus B) = \overline{\overline{A} + \overline{B} + \overline{C_{in} + A \oplus B}}$$

Logic diagram of a full-adder using only 2-input NOR gates.

Subtractors:

The subtraction of two binary numbers may be accomplished by taking the complement of the subtrahend and adding it to the minuend. By this, the subtraction operation becomes an addition operation and instead of having a separate circuit for subtraction, the adder itself can be used to perform subtraction. This results in reduction of hardware. In subtraction, each subtrahend bit of the number is subtracted from its corresponding significant minuend bit to form a difference bit. If the minuend bit is smaller than the subtrahend bit, a 1 is borrowed from the next significant position., that has been borrowed must be conveyed to the next higher pair of bits by means of a signal coming out (output) of a given stage and going into (input) the next higher stage.

The Half-Subtractor:

A Half-subtractor is a combinational circuit that subtracts one bit from the other and produces the difference. It also has an output to specify if a 1 has been borrowed. . It is used to subtract the LSB of the subtrahend from the LSB of the minuend when one binary number is subtracted from the other.

A Half-subtractor is a combinational circuit with two inputs A and B and two outputs d and b. d indicates the difference and b is the output signal generated that informs the next stage that a 1 has been borrowed. When a bit B is subtracted from another bit A, a difference bit (d) and a borrow bit (b) result according to the rules given as

Inputs		Outputs	
A	B	d	b
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	1

(a) Truth table

(b) Block diagram
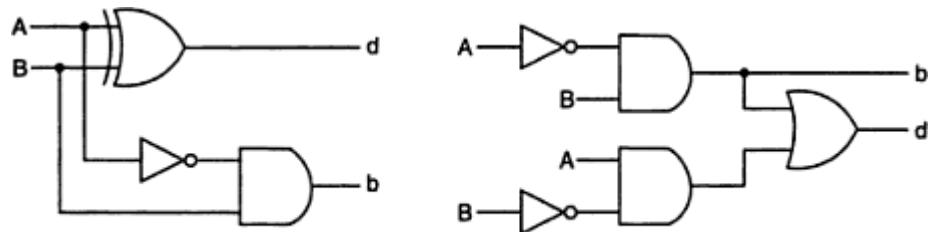
Half-subtractor.

The output borrow b is a 0 as long as A≥B. It is a 1 for A=0 and B=1. The d output is the result of the arithmetic operation2b+A-B.

A circuit that produces the correct difference and borrow bits in response to every possible combination of the two 1-bit numbers is , therefore ,

$$d=A\square+\square A \oplus B \text{ and } b=\square B$$

That is, the difference bit is obtained by X-OR ing the two inputs, and the borrow bit is obtained by ANDing the complement of the minuend with the subtrahend.Note that logic for this exactly the same as the logic for output S in the half-adder.
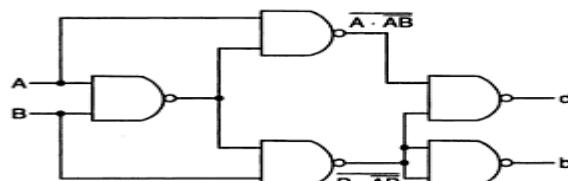
Logic diagrams of a half-subtractor.

A half-substractor can also be realized using universal logic either using only NAND gates or using NOR gates as:

NAND Logic:

$$d = A \oplus B = \overline{A \cdot \overline{AB} \cdot B \cdot \overline{AB}}$$

$$b = \overline{A}B = B(\overline{A} + \overline{B}) = B(\overline{AB}) = \overline{\overline{B \cdot \overline{AB}}}$$
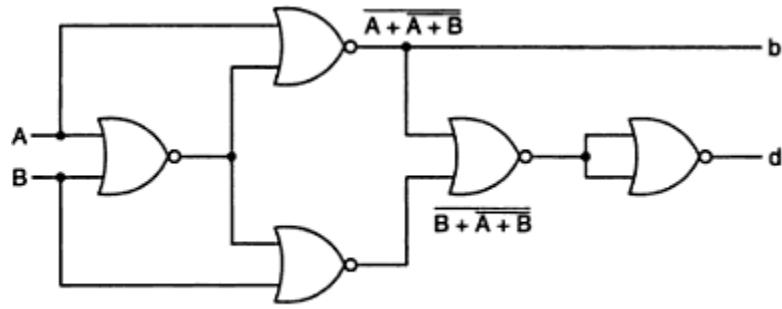
Logic diagram of a half-subtractor using only 2-input NAND gates.

NOR Logic:

$$d = A \oplus B = A\overline{B} + \overline{A}B = A\overline{B} + B\overline{B} + \overline{A}B + A\overline{A}$$

$$= \overline{B}(A + B) + \overline{A}(A + B) = \overline{\overline{B + A} + \overline{B} + A} + \overline{A + B}$$

$$d = \overline{A}B = \overline{A}(A + B) = \overline{\overline{A}(A + B)} = \overline{A + (\overline{A + B})}$$
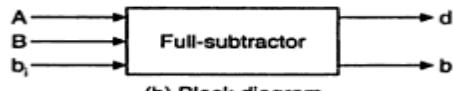
Logic diagram of a half-subtractor using only 2-input NOR gates.

The Full-Subtractor:

The half-subtractor can be only for LSB subtraction. IF there is a borrow during the subtraction of the LSBs, it affects the subtraction in the next higher column; the subtrahend bit is subtracted from the minuend bit, considering the borrow from that column used for the subtraction in the preceding column. Such a subtraction is performed by a full-subtractor. It subtracts one bit (B) from another bit (A) , when already there is a borrow b_i from this column for the subtraction in the preceding column, and outputs the difference bit (d) and the borrow bit(b) required from the next d and b. The two outputs present the difference and output borrow. The 1s and 0s for the output variables are determined from the subtraction of $A-B-b_i$.

Inputs			Difference	Borrow
A	B	b_i	d	b
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(a) Truth table

(b) Block diagram

Full-subtractor.
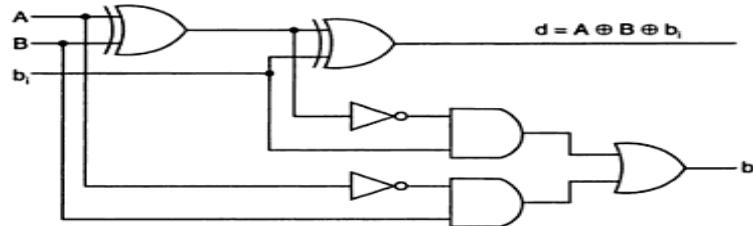
From the truth table, a circuit that will produce the correct difference and borrow bits in response to every possiblecombinations of A,B and b_i is

$$d = \overline{A}\overline{B}b_i + \overline{A}B\,\overline{b}_i + A\overline{B}\,\overline{b}_i + ABb_i$$
$$= b_i(AB + \overline{A}\overline{B}) + \overline{b}_i(A\overline{B} + \overline{A}B)$$
$$= b_i(\overline{A \oplus B}) + \overline{b}_i(A \oplus B) = A \oplus B \oplus b_i$$

and

$$b = \overline{A}\overline{B}b_i + \overline{A}B\,\overline{b}_i + \overline{A}Bb_i + ABb_i = \overline{A}B(b_i + \overline{b}_i) + (AB + \overline{A}\overline{B})b_i$$
$$= \overline{A}B + (\overline{A \oplus B})b_i$$

A full-subtractor can be realized using X-OR gates and AOI gates as
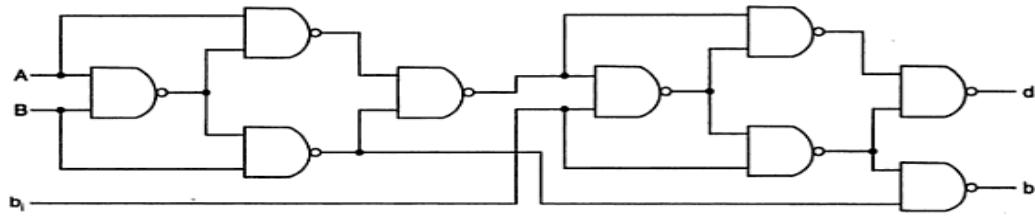
Logic diagram of a full-subtractor.

The full subtractor can also be realized using universal logic either using only NAND gates or using NOR gates as:

NAND Logic:

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}} = \overline{\overline{(A \oplus B)\overline{(A \oplus B)b_i}} \cdot \overline{b_i\overline{(A \oplus B)b_i}}}$$

$$b = \overline{A}B + b_i\overline{(A \oplus B)} = \overline{\overline{\overline{A}B + b_i\overline{(A \oplus B)}}}$$

$$= \overline{\overline{\overline{A}B} \cdot \overline{b_i(A \oplus B)}} = \overline{\overline{B(\overline{A} + \overline{B})} \cdot \overline{b_i(\overline{b_i} + (A \oplus B))}}$$

$$= \overline{\overline{B \cdot \overline{AB}} \cdot \overline{b_i[\overline{b_i} \cdot (A \oplus B)]}}$$
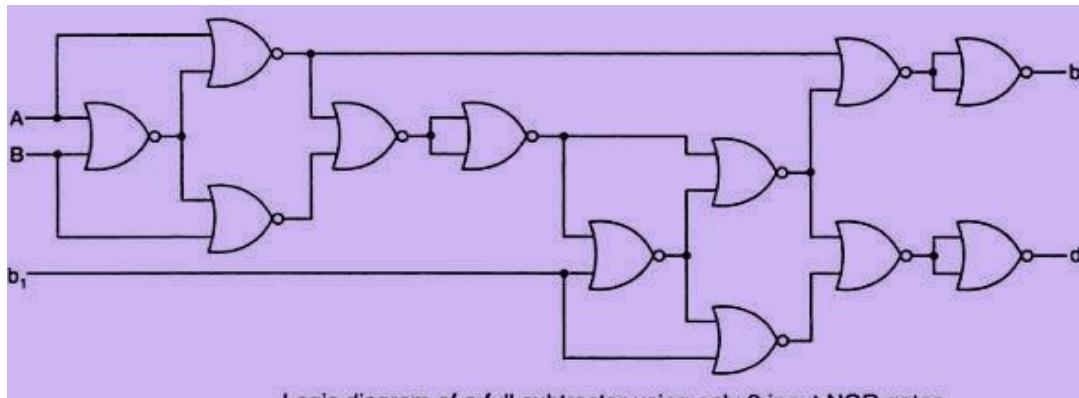
Logic diagram of a full-subtractor using only 2-input NAND gates.

NOR Logic:

$$d = A \oplus B \oplus b_i = \overline{\overline{(A \oplus B) \oplus b_i}}$$

$$= \overline{(A \oplus B)b_i + \overline{(A \oplus B)}\overline{b_i}}$$

$$= \overline{[(A \oplus B) + \overline{(A \oplus B)}\overline{b_i}][b_i + \overline{(A \oplus B)}\overline{b_i}]}$$
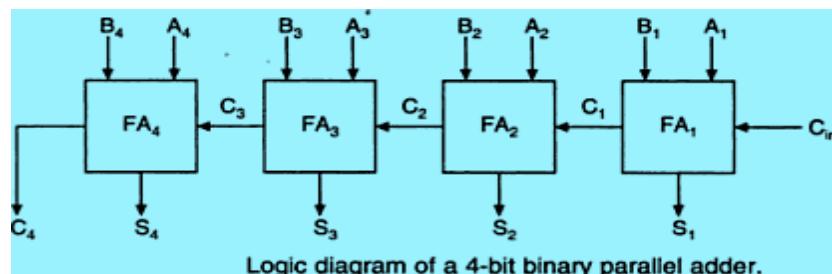
$$= \overline{\overline{(A \oplus B) + \overline{(A \oplus B)} + b_i} + \overline{b_i + \overline{(A \oplus B)} + b_i}}$$

$$= \overline{\overline{\overline{(A \oplus B) + \overline{(A \oplus B)} + b_i}} + \overline{\overline{b_i + \overline{(A \oplus B)} + b_i}}}$$

$$b = \overline{A}B + b_i(\overline{A \oplus B})$$

$$= \overline{A}(A + B) + (\overline{A \oplus B})[(A \oplus B) + b_i]$$

$$= \overline{\overline{A + \overline{(A + B)}} + \overline{(A \oplus B) + \overline{(A \oplus B)} + b_i}}$$

Logic diagram of a full subtractor using only 2-input NOR gates.

Binary Parallel Adder:

A binary parallel adder is a digital circuit that adds two binary numbers in parallel form and produces the arithmetic sum of those numbers in parallel form. It consists of full adders connected in a chain , with the output carry from each full-adder connected to the input carry of the next full-adder in the chain.

The interconnection of four full-adder (FA) circuits to provide a 4-bit parallel adder. The augends bits of A and addend bits of B are designated by subscript numbers from right to left, with subscript 1 denoting the lower –order bit. The carries are connected in a chain through the full-adders. The input carry to the adder is C_{in} and the output carry is C4. The S output generates the required sum bits. When the 4-bit full-adder circuit is enclosed within an IC package, it has four terminals for the augends bits, four terminals for the addend bits, four terminals for the sum bits, and two terminals for the input and output carries. AN n-bit parallel adder requires n-full adders. It can be constructed from 4-bit, 2-bit and 1-bit full adder ICs by cascading several packages. The output carry from one package must be connected to the input carry of the one with the next higher –order bits. The 4-bit full adder is a typical example of an MSI function.

Logic diagram of a 4-bit binary parallel adder.

Ripple carry adder:
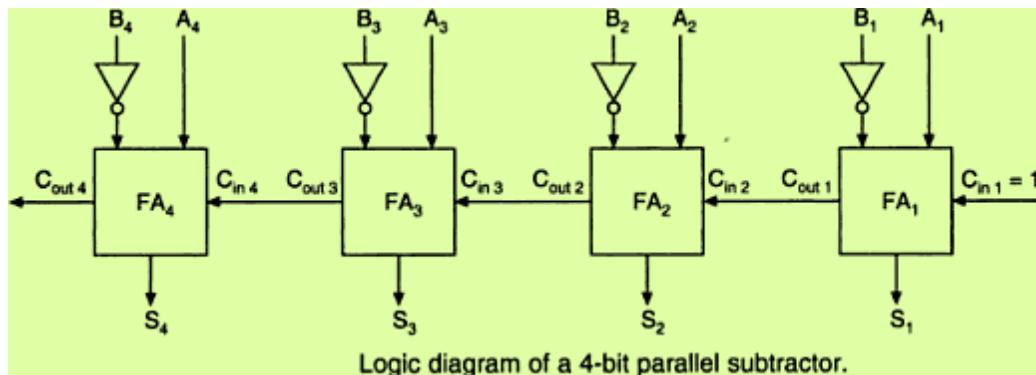
In the parallel adder, the carry –out of each stage is connected to the carry-in of the next stage. The sum and carry-out bits of any stage cannot be produced, until sometime after the carry-in of that stage occurs. This is due to the propagation delays in the logic circuitry,

94

which lead to a time delay in the addition process. The carry propagation delay for each full- adder is the time between the application of the carry-in and the occurrence of the carry-out.

The 4-bit parallel adder, the sum (S1) and carry-out (C1) bits given by FA1 are not valid, until after the propagation delay of FA1. Similarly, the sum S2 and carry-out (C2) bits given by FA2 are not valid until after the cumulative propagation delay of two full adders (FA1 and FA2) , and so on. At each stage ,the sum bit is not valid until after the carry bits in all the preceding stages are valid. Carry bits must propagate or ripple through all stages before the most significant sum bit is valid. Thus, the total sum (the parallel output) is not valid until after the cumulative delay of all the adders.

The parallel adder in which the carry-out of each full-adder is the carry-in to the next most significant adder is called a ripple carry adder.. The greater the number of bits that a ripple carry adder must add, the greater the time required for it to perform a valid addition. If two numbers are added such that no carries occur between stages, then the add time is simply the propagation time through a single full-adder.

4- Bit Parallel Subtractor:

Logic diagram of a 4-bit parallel subtractor.
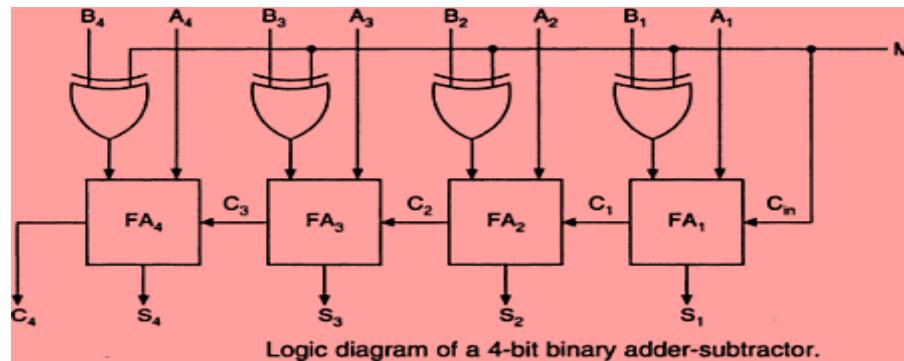
The subtraction of binary numbers can be carried out most conveniently by means of complements , the subtraction A-B can be done by taking the 2's complement of B and adding it to A . The 2's complement can be obtained by taking the 1's complement and adding 1 tothe least significant pair of bits. The 1's complement can be implemented with inverters as

Binary-Adder Subtractor:

A 4-bit adder-subtractor, the addition and subtraction operations are combined into one circuit with one common binary adder. This is done by including an X-OR gate with each full- adder. The mode input M controls the operation. When M=0, the circuit is an adder, and when M=1, the circuit becomes a subtractor. Each X-OR gate receives input M and one of the inputs of B. When M=0, $B \oplus 0 = B$ .The full-adder receives the value of B , the input carry is 0

and the circuit performs A+B. when $B \oplus 1 = B'$ and $C_1=1$. The B inputs are complemented and a 1 is through the input carry. The circuit performs the operation A plus the 2's complement of B.

Logic diagram of a 4-bit binary adder-subtractor.
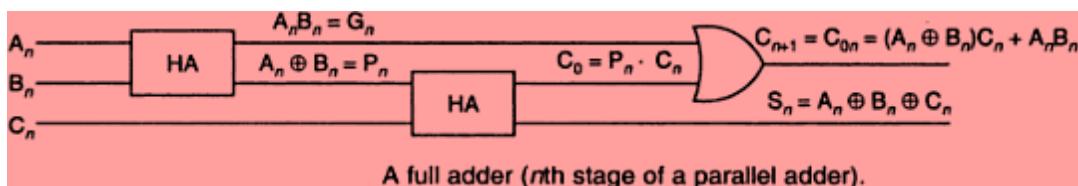
The Look-Ahead –Carry Adder:

In parallel-adder,the speed with which an addition can be performed is governed by the time required for the carries to propagate or ripple through all of the stages of the adder. The look-ahead carry adder speeds up the process by eliminating this ripple carry delay. It examines all the input bits simultaneously and also generates the carry-in bits for all the stages simultaneously.

The method of speeding up the addition process is based on the two additional functions of the full-adder, called the carry generate and carry propagate functions.

Consider one full adder stage; say the nth stage of a parallel adder as shown in fig. we know that is made by two half adders and that the half adder contains an X-OR gate to produce the sum and an AND gate to produce the carry. If both the bits A_n and B_n are 1s, a carry has to be generated in this stage regardless of whether the input carry C_{in} is a 0 or a 1. This is called generated carry, expressed as $G_n = A_n.B_n$ which has to appear at the output through the OR gate as shown in fig.

A full adder (nth stage of a parallel adder).

There is another possibility of producing a carry out. X-OR gate inside the half-aDder at the input produces an intermediary sum bit- call it P_n –which is expressed as

$$P_n = A_n \oplus B_n$$

sum bit and $S_n = P_n \oplus C_n$ where $P_n = A_n \oplus B_n$ and output carry $C_0 = P_n.C_n = (A_n \oplus B_n) C_n$ which becomes carry for the (n+1) th stage.

Consider the case of both Pn and Cn being 1. The input carry Cn has to be propagated to the output only if Pn is 1. If Pn is 0, even if Cn is 1, the and gate in the second half-adder will inhibit Cn . the carry out of the nth stage is 1 when either Gn=1 or Pn.Cn =1 or both Gn and Pn.Cn are equal to 1.

For the final sum and carry outputs of the nth stage, we get the following Boolean expressions.

$$S_n = P_n \oplus C_n \text{ where } P_n = A_n \oplus B_n$$
$$C_{on} = C_{n+1} = G_n + P_n C_n \text{ where } G_n = A_n \cdot B_n$$

Observe the recursive nature of the expression for the output carry at the nth stage which becomes the input carry for the (n+1)st stage .it is possible to express the output carry of a higher significant stage is the carry-out of the previous stage.

Based on these , the expression for the carry-outs of various full adders are as follows,

$$C_1 = G_0 + P_0 \cdot C_0$$
$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$
$$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$
$$C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$
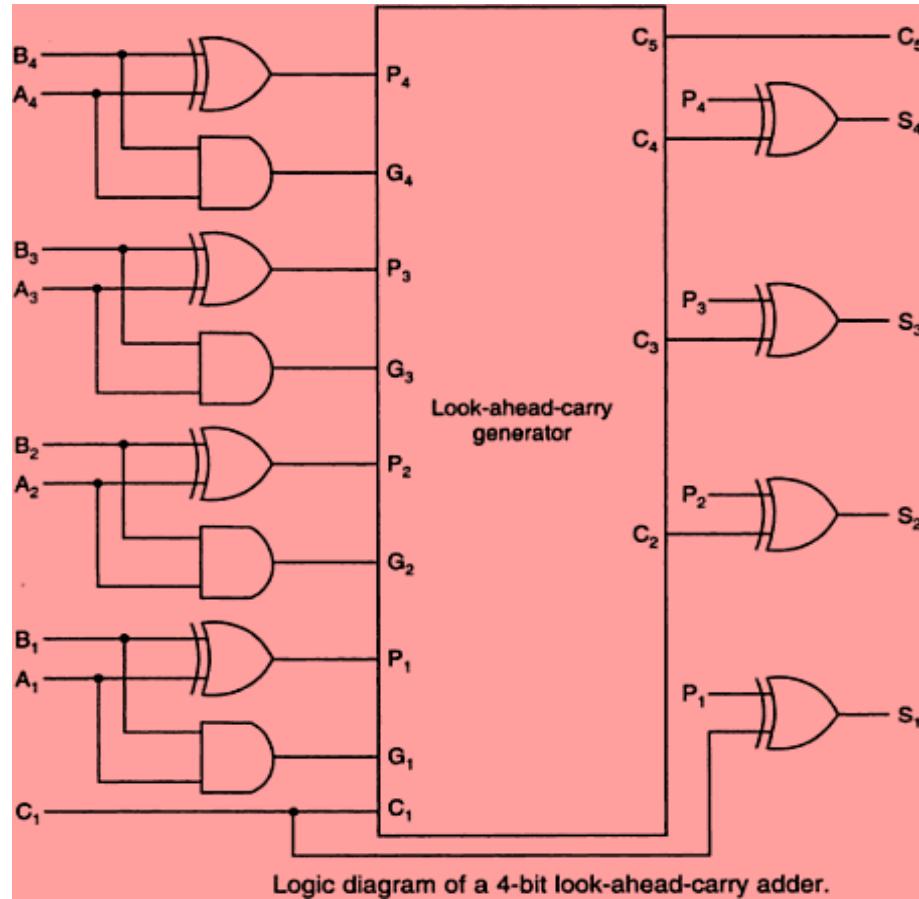
The general expression for *n* stages designated as 0 through (*n* − 1) would be

$$C_n = G_{n-1} + P_{n-1} \cdot C_{n-1} = G_{n-1} + P_{n-1} \cdot G_{n-2} + P_{n-1} \cdot P_{n-2} \cdot G_{n-3} + ... + P_{n-1} \cdot ... P_0 \cdot C_0$$
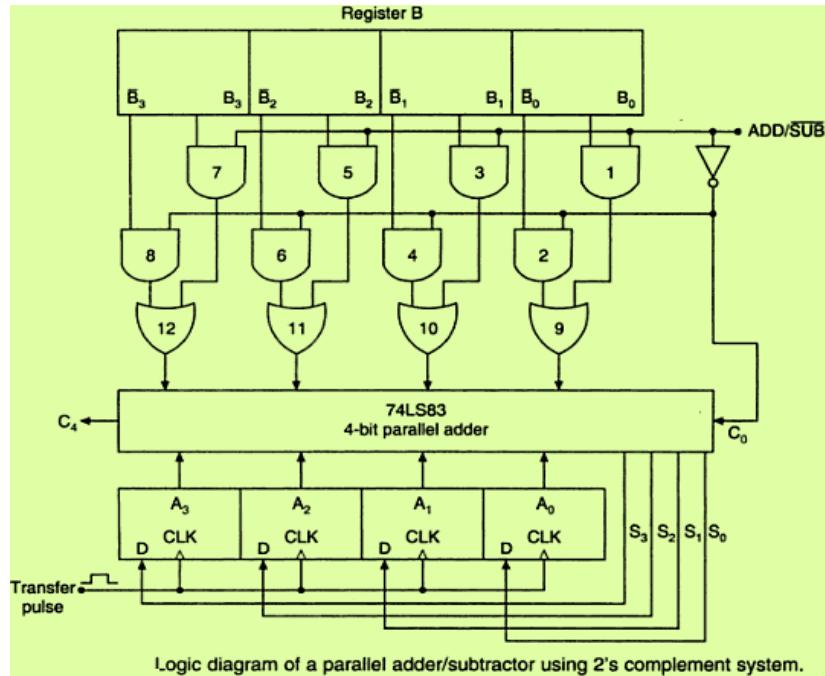
Observe that the final output carry is expressed as a function of the input variables in SOP form. Which is two level AND-OR or equivalent NAND-NAND form. Observe that the full look-ahead scheme requires the use of OR gate with (n+1) inputs and AND gates with number of inputs varying from 2 to (n+1).

Logic diagram of a 4-bit look-ahead-carry adder.

2's complement Addition and Subtraction using Parallel Adders:

Most modern computers use the 2's complement system to represent negative numbers and to perform subtraction operations of signed numbers can be performed using only the addition operation ,if we use the 2's complement form to represent negative numbers.

The circuit shown can perform both addition and subtraction in the 2's complement. This adder/subtractor circuit is controlled by the control signal ADD/SUB'. When the ADD/SUB' level is HIGH, the circuit performs the addition of the numbers stored in registers A and B. When the ADD/Sub' level is LOW, the circuit subtract the number in register B from the number in register

Logic diagram of a parallel adder/subtractor using 2's complement system.
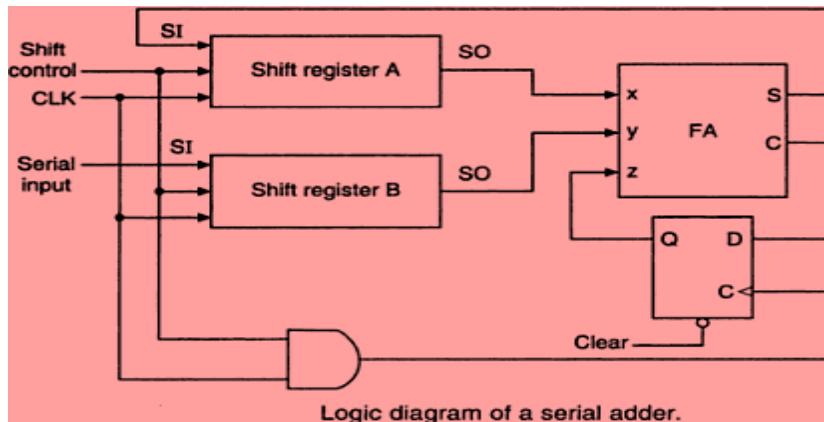
Serial Adder:

A serial adder is used to add binary numbers in serial form. The two binary numbers to be added serially are stored in two shift registers A and B. Bits are added one pair at a time through a single full adder (FA) circuit as shown. The carry out of the full-adder is transferred to a D flip-flop. The output of this flip-flop is then used as the carry input for the next pair of significant bits. The sum bit from the S output of the full-adder could be transferred to a third shift register. By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both augend and the sum bits. The serial input register B can be used to transfer a new binary number while the addend bits are shifted out during the addition.

The operation of the serial adder is:

Initially register A holds the augend, register B holds the addend and the carry flip-flop is cleared to 0. The outputs (SO) of A and B provide a pair of significant bits for the full-adder at x and y. The shift control enables both registers and carry flip-flop , so, at the clock pulse both registers are shifted once to the right, the sum bit from S enters the left most flip-flop of A , and the output carry is transferred into flip-flop Q . The shift control enables the registers for a number of clock pulses equal to the number of bits of the registers. For each succeeding clock pulse a new sum bit is transferred to A, a new carry is transferred to Q, and both registers are shifted once to the right. This process continues until the shift control is disabled. Thus the addition is

accomplished by passing each pair of bits together with the previous carry through a single full adder circuit and transferring the sum, one bit at a time, into register A.

Initially, register A and the carry flip-flop are cleared to 0 and then the first number is added from B. While B is shifted through the full adder, a second number is transferred to it through its serial input. The second number is then added to the content of register A while a third number is transferred serially into register B. This can be repeated to form the addition of two, three, or more numbers and accumulate their sum in register A.

Logic diagram of a serial adder.

Difference between Serial and Parallel Adders:

The parallel adder registers with parallel load, whereas the serial adder uses shift registers. The number of full adder circuits in the parallel adder is equal to the number of bits in the binary numbers, whereas the serial adder requires only one full adder circuit and a carry flip- flop. Excluding the registers, the parallel adder is a combinational circuit, whereas the serial adder is a sequential circuit. The sequential circuit in the serial adder consists of a full-adder and a flip-flop that stores the output carry.

BCD Adder:
The BCD addition process:
- Add the 4-bit BCD code groups for each decimal digit position using ordinary binary addition.

- For those positions where the sum is 9 or less, the sum is in proper BCD form and no correction is needed.

- When the sum of two digits is greater than 9, a correction of 0110 should be added to that sum, to produce the proper BCD result. This will produce a carry to be added to the next decimalposition.

A BCD adder circuit must be able to operate in accordance with the above steps. In other words, the circuit must be able to do the following:

1. Add two 4-bit BCD code groups, using straight binaryaddition.

100

2. Determine, if the sum of this addition is greater than 1101 (decimal 9); if it is , add 0110 (decimal 6) to this sum and generate a carry to the next decimal position.

The first requirement is easily met by using a 4- bit binary parallel adder such as the 74LS83 IC .For example , if the two BCD code groups $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are applied to a 4-bit parallel adder, the adder will output $S_4S_3S_2S_1S_0$, where S_4 is actually C_4 , the carry –out of the MSB bits.

The sum outputs $S_4S_3S_2S_1S_0$ can range anywhere from 00000 to 10010 9when both the BCD code groups are 1001=9). The circuitry for a BCD adder must include the logic needed to detect whenever the sum is greater than 01001, so that the correction can be added in. Those cases , where the sum is greater than 1001 are listed as:

S_4	S_3	S_2	S_1	S_0	Decimal number
0	1	0	1	0	10
0	1	0	1	1	11
0	1	1	0	0	12
0	1	1	0	1	13
0	1	1	1	0	14
0	1	1	1	1	15
1	0	0	0	0	16
1	0	0	0	1	17
1	0	0	1	0	18
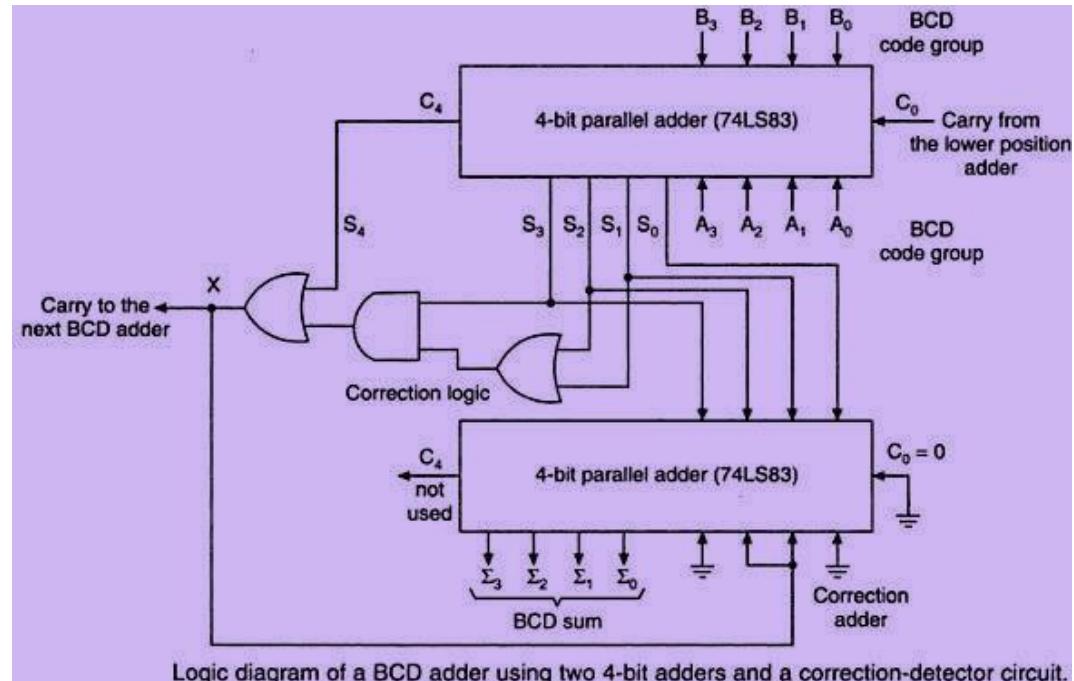
Let us define a logic output X that will go HIGH only when the sum is greater than 01001 (i.e, for the cases in table). If examine these cases ,see that X will be HIGH for either of the following conditions:

1. Whenever $S_4 =1$ (sum greater than 15)

2. Whenever $S_3 =1$ and either S_2 or S_1 or both are 1 (sum 10 to

15) This condition can be expressed as

$$X=S_4+S_3(S_2+S_1)$$

Whenever X=1, it is necessary to add the correction factor 0110 to the sum bits, and to generate a carry. The circuit consists of three basic parts. The two BCD code groups $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are added together in the upper 4-bit adder, to produce the sum $S_4S_3S_2S_1S_0$. The logic gates shown implement the expression for X. The lower 4-bit adder will add the correction 0110 to the sum bits, only when X=1, producing the final BCD sum output represented by $\Sigma_3\Sigma_2\Sigma_1\Sigma_0$. The X is also the carry-out that is produced when the sum is greater than 01001. When X=0, there is no carry and no addition of 0110. In such cases, $\Sigma_3\Sigma_2\Sigma_1\Sigma_0= S_3S_2S_1S_0$.

Two or more BCD adders can be connected in cascade when two or more digit decimal numbers are to be added. The carry-out of the first BCD adder is connected as the carry-in of the second BCD adder, the carry-out of the second BCD adder is connected as the carry-in of the third BCD adder and so on.

Logic diagram of a BCD adder using two 4-bit adders and a correction-detector circuit.

EXCESS-3 (XS-3) ADDER:

To perform Excess-3 additions,
1. Add two xs-3 code groups
2. If carry=1, add 0011(3) to the sum of those two codegroups

   If carry =0, subtract 0011(3) i.e., add 1101 (13 in decimal) to the sum of those two code groups.

   Ex: Add 9 and 5

1100		9 in Xs-3
	+1000	5 in xs-3
	——— — — ———	
1	0100	there is a carry
+0011	0011	add 3 to each group
----------	----------	
0100	0111	14 in xs-3
(1)	(4)	

EX:

```
(b)    0 1 1
      +0 1 1
      -------
       1 1 0
      +1 1 0
      -------
Ignore carry 1 1 0 1
              (7)
```

adder. If the carry is a 0, then 1101(3) is added to the sum bits (This is equivalent to subtracting 0011(3) from the sum bits. The correct sum in xs-3 is obtained

Excess-3 (XS-3) Subtractor:

To perform Excess-3 subtraction,
1. Complement the subtrahend
2. Add the complemented subtrahend to the minuend.
3. If carry =1, result is positive. Add 3 and end around carry to the result . If carry=0, the result is negative. Subtract 3, i.e, and take the 1's complement of the result.

```
Ex:    Perform 9-4
        1100          9 in xs-3
       +1000          Complement of 4 n Xs-3
       --------
(1)     0100          There is a carry
       +0011          Add 0011(3)
     ------------
        0111
           1          End around carry
     ------------
        1000          5 in xs-3
```

The minuend and the 1's complement of the subtrahend in xs-3 are added in the upper 4- bit parallel adder. If the carry-out from the upper adder is a 0, then 1101 is added to the sum bits of the upper adder in the lower adder and the sum bits of the lower adder are complemented to get the result. If the carry-out from the upper adder is a 1, then 3=0011 is added to the sum bits of the lower adder and the sum bits of the lower adder give the result

Code Converters:

A conversion circuit must be inserted between the two systems if each uses different codes for the same information. Thus a code converter is a logic circuit whose inputs are bit patterns representing numbers (or character) in one cod and whose outputs are the corresponding representation in a different code. Code converters are usually multiple output circuits.
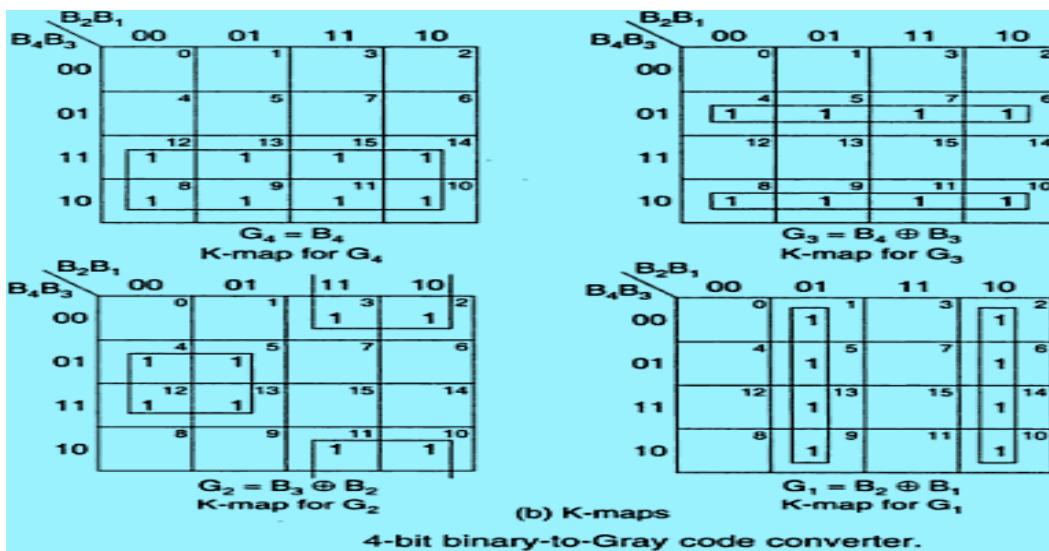
To convert from binary code A to binary code B, the input lines must supply the bit combination of elements as specified by code A and the output lines must generate the corresponding bit combination of code B. A combinational circuit performs this transformation by means of logic gates.

For example, a binary –to-gray code converter has four binary input lines B4, B3,B2,B1 and four gray code output lines G4,G3,G2,G1. When the input is 0010, for instance, the output should be 0011 and so forth. To design a code converter, we use a code table treating it as a truth table to express each output as a Boolean algebraic function of all the inputs.

In this example, of binary –to-gray code conversion, we can treat the binary to the gray code table as four truth tables to derive expressions for G4, G3, G2, and G1. Each of these four expressions would, in general, contain all the four input variables B4, B3,B2,and B1. Thus,this code converter is actually equivalent to four logic circuits, one for each of the truth tables.

The logic expression derived for the code converter can be simplified using the usual techniques, including ‗don't cares' if present. Even if the input is an unweighted code, the same cell numbering method which we used earlier can be used, but the cell numbers --must correspond to the input combinations as if they were an 8-4-2-1 weighted code.

Design of a 4-bit binary to gray code converter:

(b) K-maps

4-bit binary-to-Gray code converter.

Design of a 4-bit gray to Binary code converter:
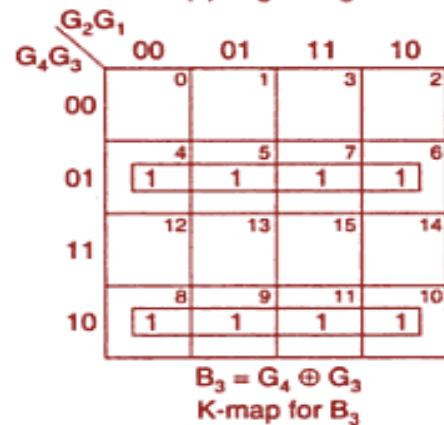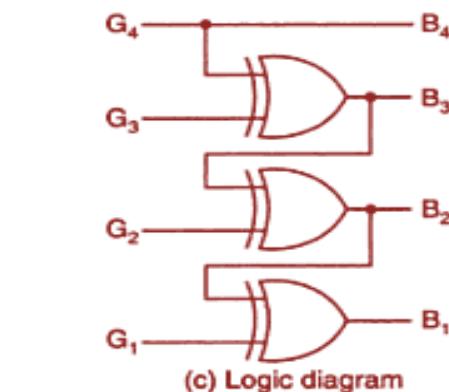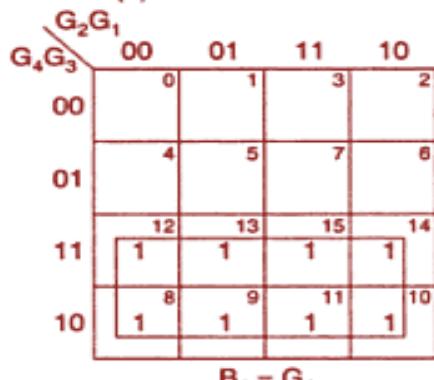
$B_4 = \Sigma\, m(12, 13, 15, 14, 10, 11, 9, 8) = \Sigma\, m(8, 9, 10, 11, 12, 13, 14, 15)$
$B_3 = \Sigma\, m(6, 7, 5, 4, 10, 11, 9, 8) = \Sigma\, m(4, 5, 6, 7, 8, 9, 10, 11)$
$B_2 = \Sigma\, m(3, 2, 5, 4, 15, 14, 9, 8) = \Sigma\, m(2, 3, 4, 5, 8, 9, 14, 15)$
$B_1 = \Sigma\, m(1, 2, 7, 4, 13, 14, 11, 8) = \Sigma\, m(1, 2, 4, 7, 8, 11, 13, 14)$
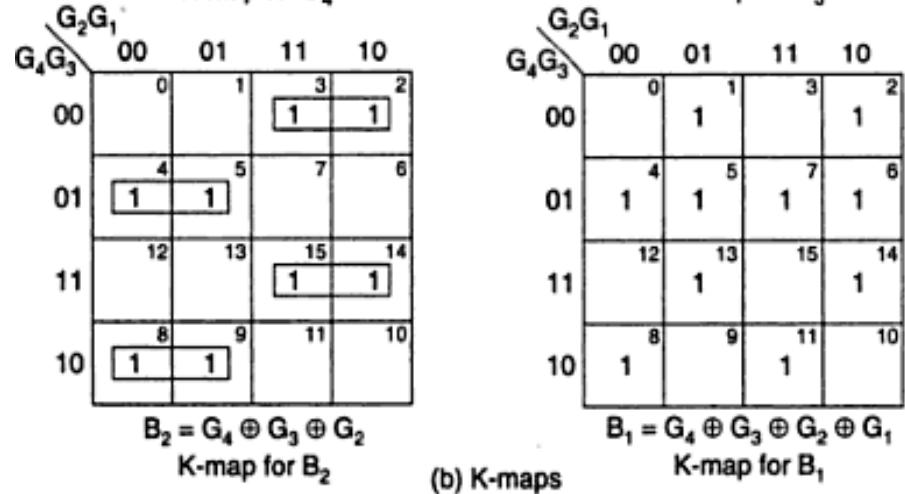
$B_4 = G_4$
$B_3 = \bar{G}_4 G_3 + G_4 \bar{G}_3 = G_4 \oplus G_3$
$B_2 = \bar{G}_4 G_3 \bar{G}_2 + \bar{G}_4 \bar{G}_3 G_2 + G_4 \bar{G}_3 \bar{G}_2 + G_4 G_3 G_2$
$\quad = \bar{G}_4(G_3 \oplus G_2) + G_4(\overline{G_3 \oplus G_2}) = G_4 \oplus G_3 \oplus G_2 = B_3 \oplus G_2$
$B_1 = \bar{G}_4 \bar{G}_3 \bar{G}_2 G_1 + \bar{G}_4 \bar{G}_3 G_2 \bar{G}_1 + \bar{G}_4 G_3 G_2 G_1 + \bar{G}_4 G_3 \bar{G}_2 \bar{G}_1 + G_4 G_3 \bar{G}_2 G_1$
$\qquad\qquad\qquad\qquad\qquad + G_4 G_3 G_2 \bar{G}_1 + G_4 \bar{G}_3 G_2 G_1 + G_4 \bar{G}_3 \bar{G}_2 \bar{G}_1$

$$= \bar{G}_4 \bar{G}_3 (G_2 \oplus G_1) + G_4 G_3 (G_2 \oplus G_1) + \bar{G}_4 G_3 (\overline{G_2 \oplus G_1}) + G_4 \bar{G}_3 (\overline{G_2 \oplus G_1})$$
$$= (G_2 \oplus G_1)(\overline{G_4 \oplus G_3}) + (\overline{G_2 \oplus G_1})(G_4 \oplus G_3)$$
$$= G_4 \oplus G_3 \oplus G_2 \oplus G_1$$

4-bit Gray				4-bit binary			
G_4	G_3	G_2	G_1	B_4	B_3	B_2	B_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

(a) Conversion table

(c) Logic diagram

$B_4 = G_4$
K-map for B_4

$B_3 = G_4 \oplus G_3$
K-map for B_3

$$B_2 = G_4 \oplus G_3 \oplus G_2$$
K-map for B_2

(b) K-maps

$$B_1 = G_4 \oplus G_3 \oplus G_2 \oplus G_1$$
K-map for B_1

4-bit Gray-to-binary code converter.

Design of a 4-bit BCD to XS-3 code converter:

8421 code				XS-3 code			
B_4	B_3	B_2	B_1	X_4	X_3	X_2	X_1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

(a) Conversion table
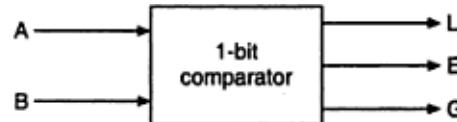
$X_4 = \Sigma\, m(5, 6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)$
$X_3 = \Sigma\, m(1, 2, 3, 4, 9) + d(10, 11, 12, 13, 14, 15)$
$X_2 = \Sigma\, m(0, 3, 4, 7, 8) + d(10, 11, 12, 13, 14, 15)$
$X_1 = \Sigma\, m(0, 2, 4, 6, 8) + d(10, 11, 12, 13, 14, 15)$

The minimal expressions are

$X_4 = B_4 + B_3 B_2 + B_3 B_1$
$X_3 = B_3 \bar{B}_2 \bar{B}_1 + \bar{B}_3 B_1 + \bar{B}_3 B_2$
$X_2 = \bar{B}_2 \bar{B}_1 + B_2 B_1$
$X_1 = \bar{B}_1$

(b) Minimal expressions

4-bit BCD-to-XS-3 code converter

Comparators:

$$EQUALITY = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

Block diagram of a 1-bit comparator.

The logic for a 1-bit magnitude comparator: Let the 1-bit numbers be $A = A_0$ and $B = B_0$.
If $A_0 = 1$ and $B_0 = 0$, then $A > B$.
Therefore,

$$A > B: G = A_0 \overline{B}_0$$

If $A_0 = 0$ and $B_0 = 1$, then $A < B$.
Therefore,

$$A < B: L = \overline{A}_0 B_0$$
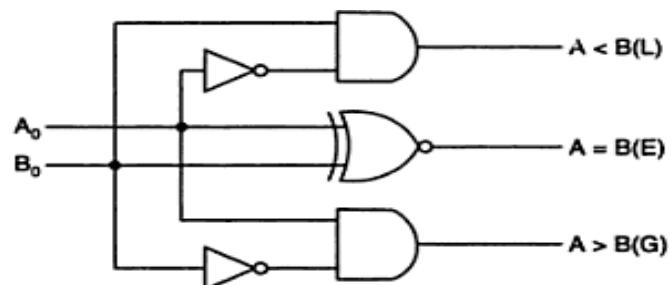
If A_0 and B_0 coincide, i.e. $A_0 = B_0 = 0$ or if $A_0 = B_0 = 1$, then $A = B$.
Therefore,

$$A = B : E = A_0 \odot B_0$$

A_0	B_0	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

(a) Truth table

(b) Logic diagram

1-bit comparator.

1. Magnitude Comparator:

1- bit Magnitude Comparator:

The logic for a 2-bit magnitude comparator: Let the two 2-bit numbers be $A = A_1 A_0$ and $B = B_1 B_0$.

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$. So the logic expression for $A > B$ is
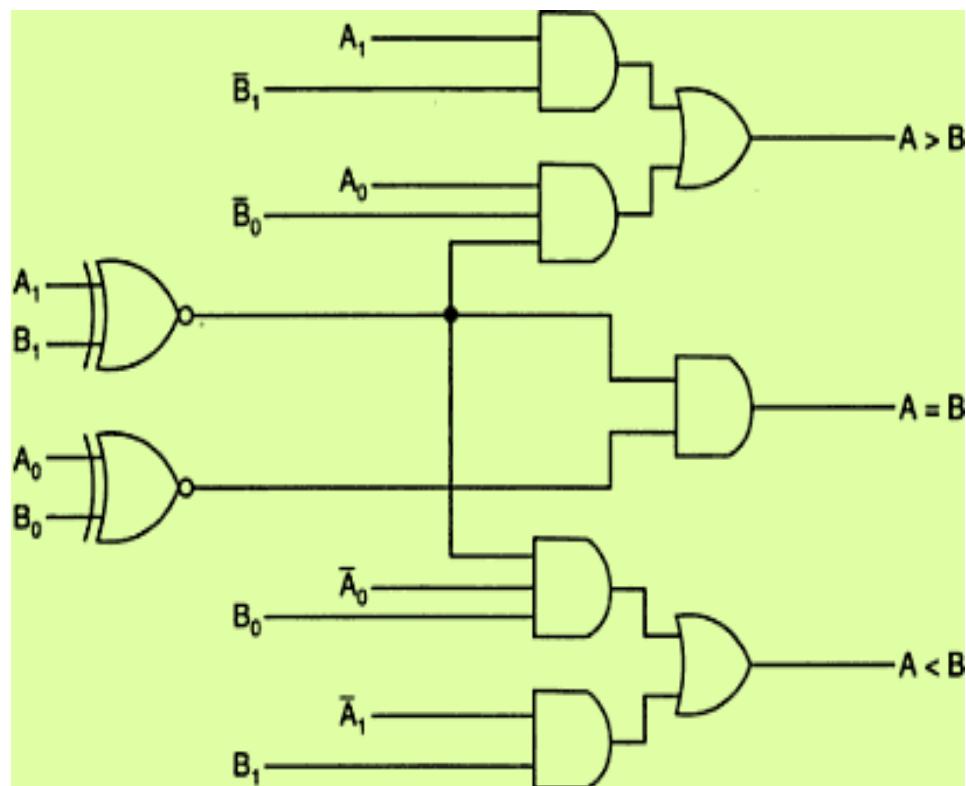
$$A > B : G = A_1 \bar{B}_1 + (A_1 \odot B_1)A_0 \bar{B}_0$$

1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If A_1 and B_1 coincide and $A_0 = 0$ and $B_0 = 1$, then $A < B$. So the expression for $A < B$ is

$$A < B : L = \bar{A}_1 B_1 + (A_1 \odot B_1)\bar{A}_0 B_0$$

If A_1 and B_1 coincide and if A_0 and B_0 coincide then $A = B$. So the expression for $A = B$ is

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$

Logic diagram of a 2-bit magnitude comparator.

4- Bit Magnitude Comparator:

The logic for a 4-bit magnitude comparator: Let the two 4-bit numbers be $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$.

1. If $A_3 = 1$ and $B_3 = 0$, then $A > B$. Or
2. If A_3 and B_3 coincide, and if $A_2 = 1$ and $B_2 = 0$, then $A > B$. Or
3. If A_3 and B_3 coincide, and if A_2 and B_2 coincide, and if $A_1 = 1$ and $B_1 = 0$, then $A > B$. Or
4. If A_3 and B_3 coincide, and if A_2 and B_2 coincide, and if A_1 and B_1 coincide, and if $A_0 = 1$ and $B_0 = 0$, then $A > B$.

From these statements, we see that the logic expression for $A > B$ can be written as

$$(A > B) = A_3\bar{B}_3 + (A_3 \odot B_3)A_2\bar{B}_2 + (A_3 \odot B_3)(A_2 \odot B_2)A_1\bar{B}_1$$
$$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)A_0\bar{B}_0$$
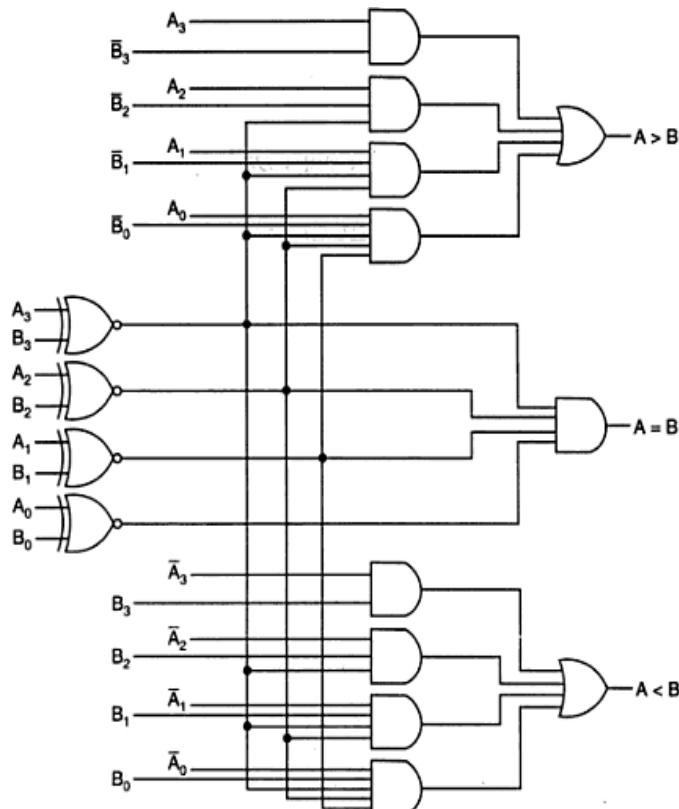
Similarly, the logic expression for $A < B$ can be written as

$$A < B = \bar{A}_3B_3 + (A_3 \odot B_3)\bar{A}_2B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1B_1$$
$$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\bar{A}_0B_0$$
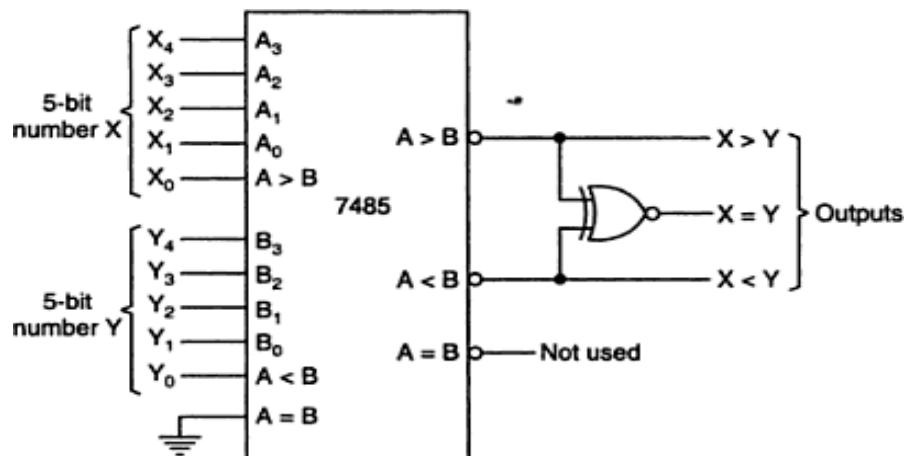
If A_3 and B_3 coincide and if A_2 and B_2 coincide and if A_1 and B_1 coincide and if A_0 and B_0 coincide, then $A = B$.

So the expression for $A = B$ can be written as
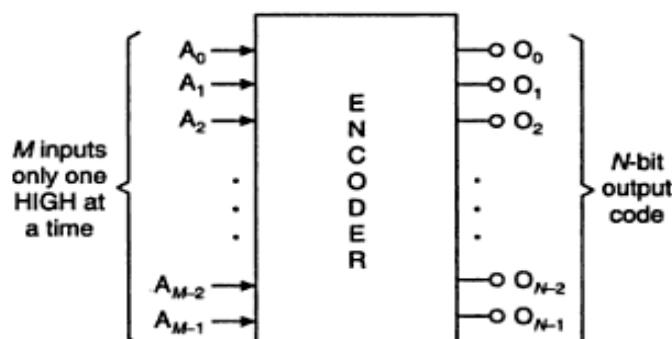
$$(A = B) = (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

Encoders:

Use of 7485 as a 5-bit comparator.
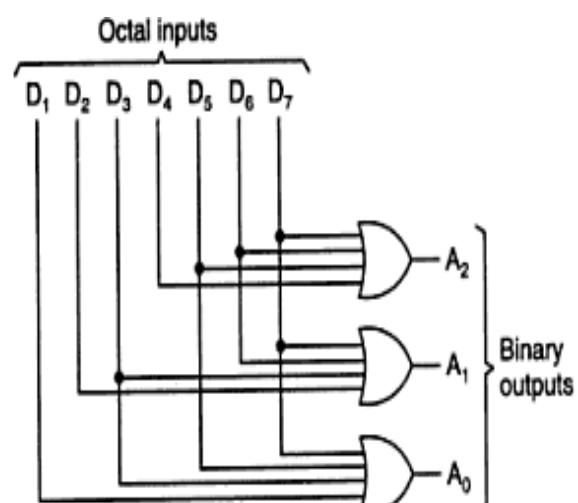
Block diagram of encoder.

Octal to Binary Encoder:

Octal digits		Binary		
		A_2	A_1	A_0
D_0	0	0	0	0
D_1	1	0	0	1
D_2	2	0	1	0
D_3	3	0	1	1
D_4	4	1	0	0
D_5	5	1	0	1
D_6	6	1	1	0
D_7	7	1	1	1
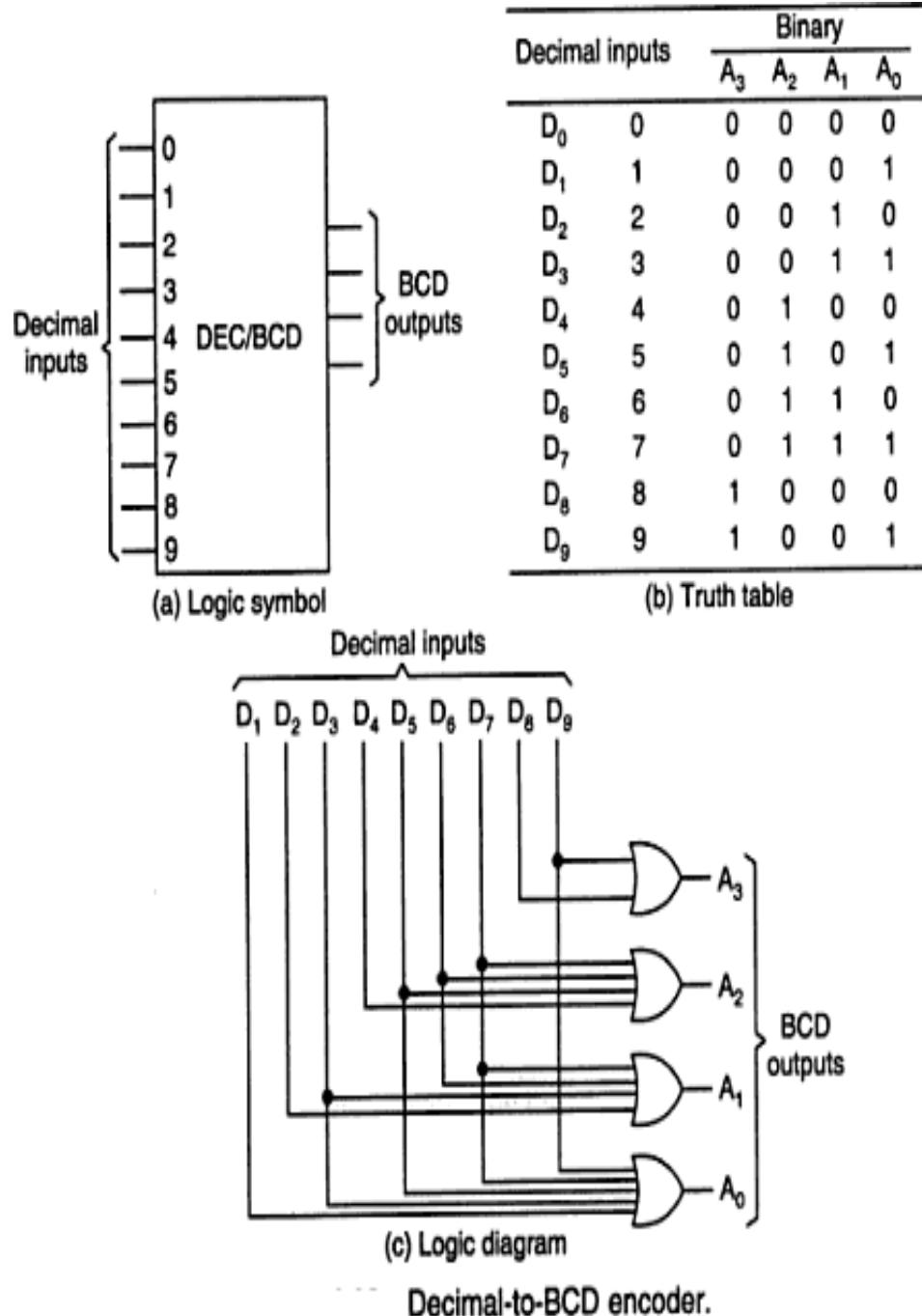
(a) Truth table

(b) Logic diagram

Octal-to-binary encoder.

Decimal to BCD Encoder:

Decimal inputs		Binary			
		A_3	A_2	A_1	A_0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

(a) Logic symbol   (b) Truth table

(c) Logic diagram

Decimal-to-BCD encoder.
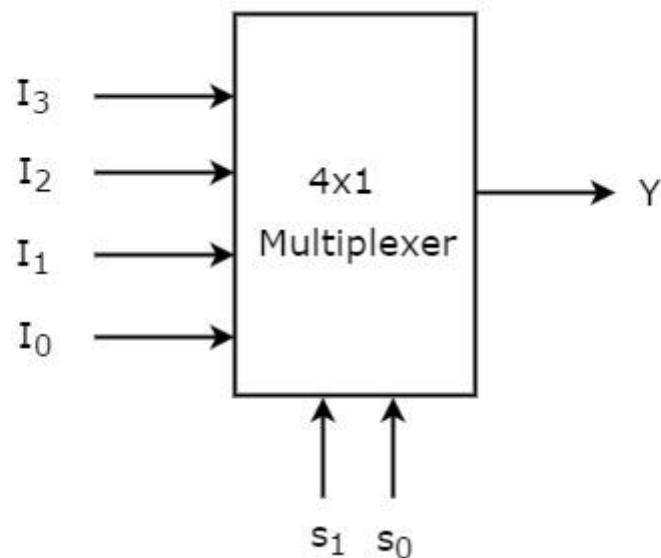
Multiplexer is a combinational circuit that has maximum of 2^n data inputs, 'n' selection lines and single output line. One of these data inputs will be connected to the output based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination will select only one data input. Multiplexer is also called as **Mux**.

4x1 Multiplexer

4x1 Multiplexer has four data inputs I_3, I_2, I_1 & I_0, two selection lines s_1 & s_0 and one output Y. The **block diagram** of 4x1 Multiplexer is shown in the following figure.
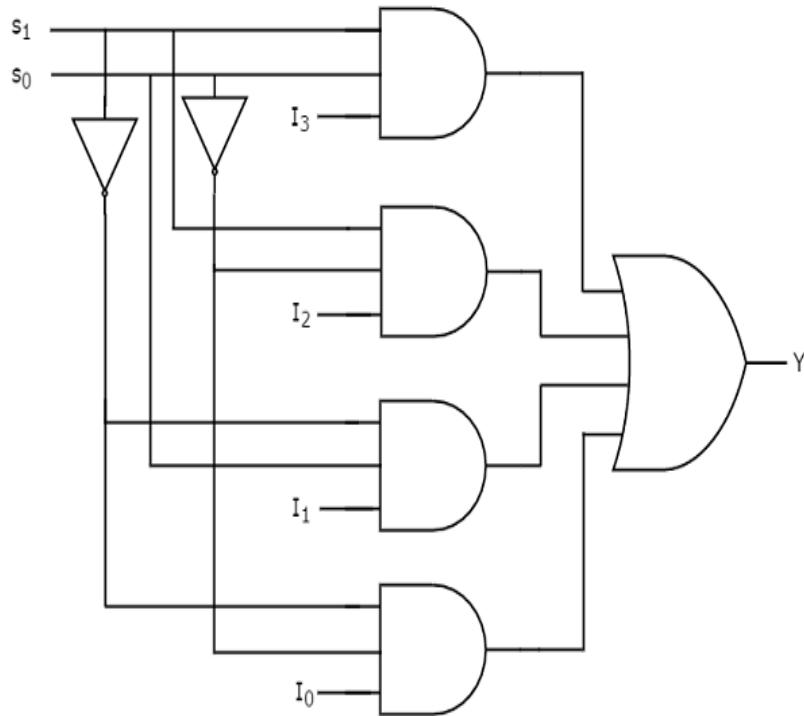
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

Selection Lines		Output
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

From Truth table, we can directly write the **Boolean function** for output, Y as

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3 Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

We can implement this Boolean function using Inverters, AND gates & OR gate. The **circuit diagram** of 4x1 multiplexer is shown in the following figure.

We can easily understand the operation of the above circuit. Similarly, you can implement 8x1 Multiplexer and 16x1 multiplexer by following the same procedure.

Implementation of Higher-order Multiplexers.

Now, let us implement the following two higher-order Multiplexers using lower-order Multiplexers.

- 8x1 Multiplexer
- 16x1 Multiplexer

8x1 Multiplexer:
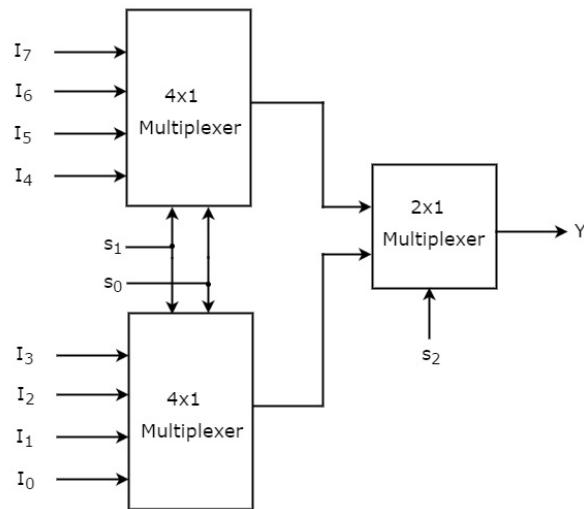
In this section, let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer. We know that 4x1 Multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output.

So, we require two **4x1 Multiplexers** in first stage in order to get the 8 data inputs. Since, each 4x1 Multiplexer produces one output, we require a **2x1 Multiplexer** in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 8x1 Multiplexer has eight data inputs I_7 to I_0, three selection lines s_2, s_1 & s0 and one output Y. The **Truth table** of 8x1 Multiplexer is shown below.

Selection Inputs			Output
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

We can implement 8x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The **block diagram** of 8x1 Multiplexer is shown in the following figure.

The same **selection lines, s_1 & s_0** are applied to both 4x1 Multiplexers. The data inputs of upper 4x1 Multiplexer are I_7 to I_4 and the data inputs of lower 4x1 Multiplexer are I_3 to I_0. Therefore, each 4x1 Multiplexer produces an output based on the values of selection lines, s_1 & s_0.

The outputs of first stage 4x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other **selection line, s_2** is applied to 2x1 Multiplexer.

- If s_2 is zero, then the output of 2x1 Multiplexer will be one of the 4 inputs I_3 to I_0 based on the values of selection lines s_1 & s_0.

- If s_2 is one, then the output of 2x1 Multiplexer will be one of the 4 inputs I_7 to I_4 based on the values of selection lines s_1 & s_0.

Therefore, the overall combination of two 4x1 Multiplexers and one 2x1 Multiplexer performs as one 8x1 Multiplexer.

16x1 Multiplexer:
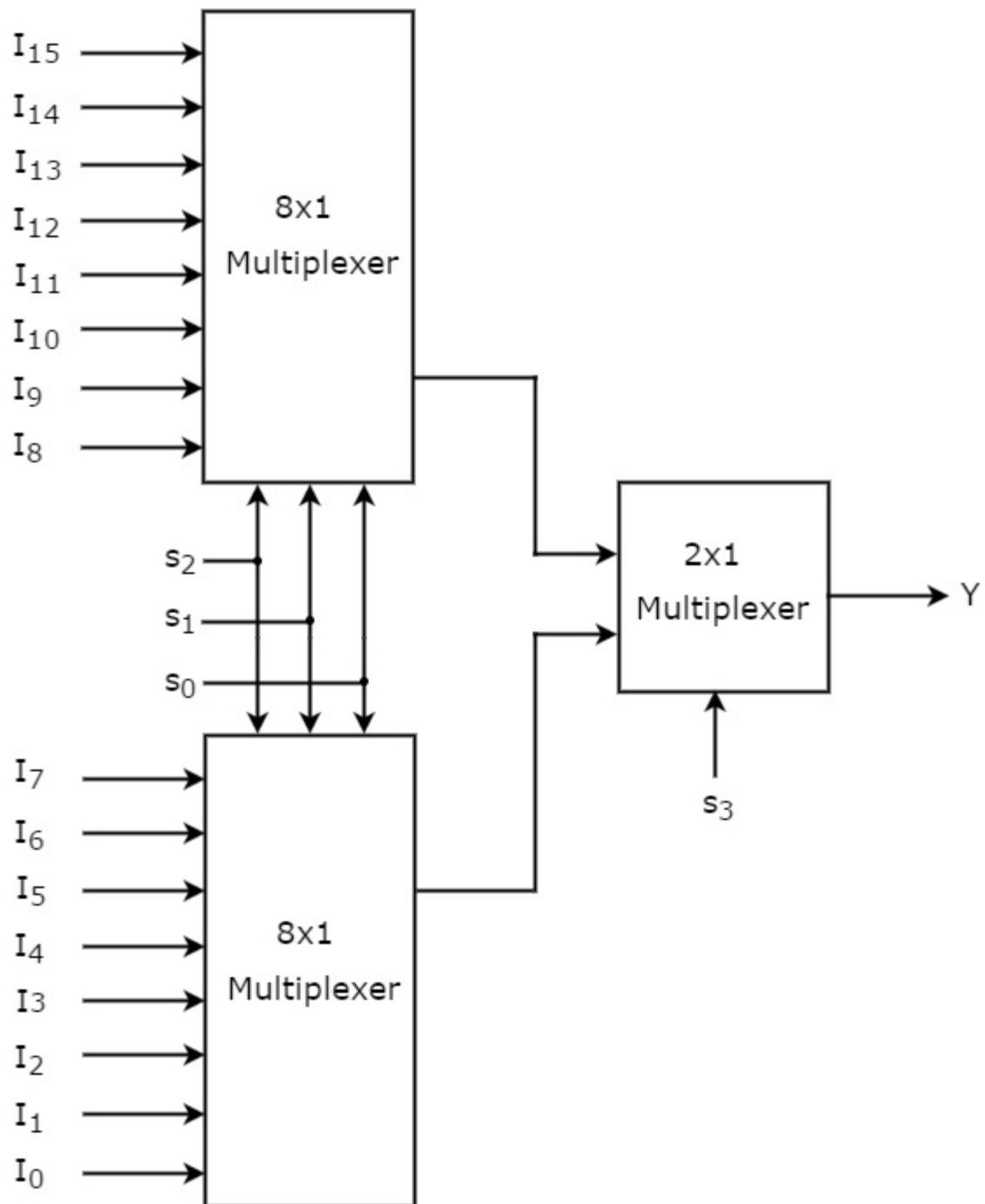
In this section, let us implement 16x1 Multiplexer using 8x1 Multiplexers and 2x1 Multiplexer. We know that 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output. Whereas, 16x1 Multiplexer has 16 data inputs, 4 selection lines and one output.

So, we require two **8x1 Multiplexers** in first stage in order to get the 16 data inputs. Since, each 8x1 Multiplexer produces one output, we require a 2x1 Multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output.

Let the 16x1 Multiplexer has sixteen data inputs I_{15} to I_0, four selection lines s_3 to s_0 and one output Y. The **Truth table** of 16x1 Multiplexer is shown below.

Selection Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}
1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

We can implement 16x1 Multiplexer using lower order Multiplexers easily by considering the above Truth table. The **block diagram** of 16x1 Multiplexer is shown in the following figure.

The **same selection lines, s_2, s_1 & s_0** are applied to both 8x1 Multiplexers. The data inputs of upper 8x1 Multiplexer are I_{15} to I_8 and the data inputs of lower 8x1 Multiplexer are I_7 to I_0. Therefore, each 8x1 Multiplexer produces an output based on the values of selection lines, s_2, s_1 & s_0.

The outputs of first stage 8x1 Multiplexers are applied as inputs of 2x1 Multiplexer that is present in second stage. The other **selection line, s_3** is applied to 2x1 Multiplexer.

- If s_3 is zero, then the output of 2x1 Multiplexer will be one of the 8 inputs Is_7 to I_0 based on the values of selection lines s_2, s_1 & s_0.

- If s_3 is one, then the output of 2x1 Multiplexer will be one of the 8 inputs I_{15} to I_8 based on the values of selection lines s_2, s_1 & s_0.

Therefore, the overall combination of two 8x1 Multiplexers and one 2x1 Multiplexer performs as one 16x1 Multiplexer.

Applications of Multiplexer:

Multiplexer are used in various fields where multiple data need to be transmitted using a single line. Following are some of the applications of multiplexers -

1. **Communication system** – Communication system is a set of system that enable communication like transmission system, relay and tributary station, and communication network. The efficiency of communication system can be increased considerably using multiplexer. Multiplexer allow the process of transmitting different type of data such as audio, video at the same time using a single transmission line.
2. **Telephone network –** In telephone network, multiple audio signals are integrated on a single line for transmission with the help of multiplexers. In this way, multiple audio signals can be isolated and eventually, the desire audio signals reach the intended recipients.
3. **Computer memory** - Multiplexers are used to implement huge amount of memory into the computer, at the same time reduces the number of copper lines required to connect the memory to other parts of the computer circuit.
4. **Transmission from the computer system of a satellite** – Multiplexer can be used for the transmission of data signals from the computer system of a satellite or spacecraft to the ground system using the GPS (Global Positioning System) satellites.
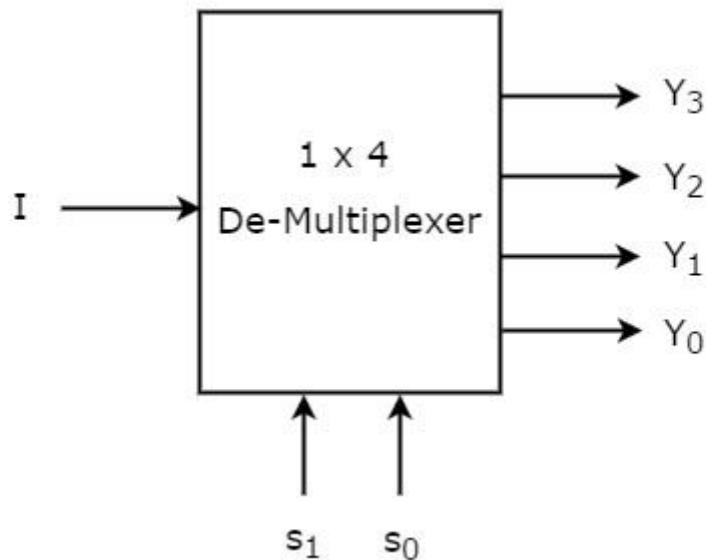
De-Multiplexer

De-Multiplexer is a combinational circuit that performs the reverse operation of Multiplexer. It has single input, 'n' selection lines and maximum of 2^n outputs. The input will be connected to one of these outputs based on the values of selection lines.

Since there are 'n' selection lines, there will be 2^n possible combinations of zeros and ones. So, each combination can select only one output. De-Multiplexer is also called as **De-Mux**.

1x4 De-Multiplexer

1x4 De-Multiplexer has one input I, two selection lines, s_1 & s_0 and four outputs Y_3, Y_2, Y_1 & Y_0. The **block diagram** of 1x4 De-Multiplexer is shown in the following figure.

The single input 'I' will be connected to one of the four outputs, Y_3 to Y_0 based on the values of selection lines s_1 & s0. The **Truth table** of 1x4 De-Multiplexer is shown below.

From the above Truth table, we can directly write the **Boolean functions** for each output as

Selection Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0
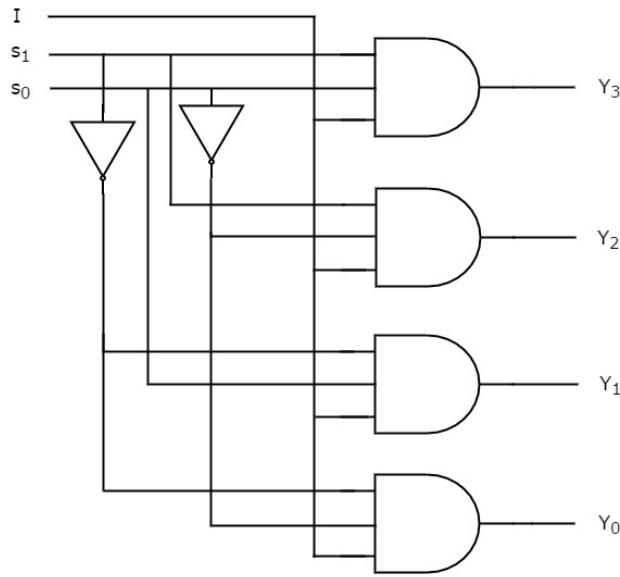
$$Y3=s1s0I$$
$$Y2=s1s0I'$$
$$Y1=s1's0I$$
$$Y0=s1's0I'$$

We can implement these Boolean functions using Inverters & 3-input AND gates. The **circuit diagram** of 1x4 De-Multiplexer is shown in the following figure.

We can easily understand the operation of the above circuit. Similarly, you can implement 1x8 De-Multiplexer and 1x16 De-Multiplexer by following the same procedure.

Implementation of Higher-order De-Multiplexers

Now, let us implement the following two higher-order De-Multiplexers using lower-order De-Multiplexers.

- 1x8 De-Multiplexer
- 1x16 De-Multiplexer

1x8 De-Multiplexer
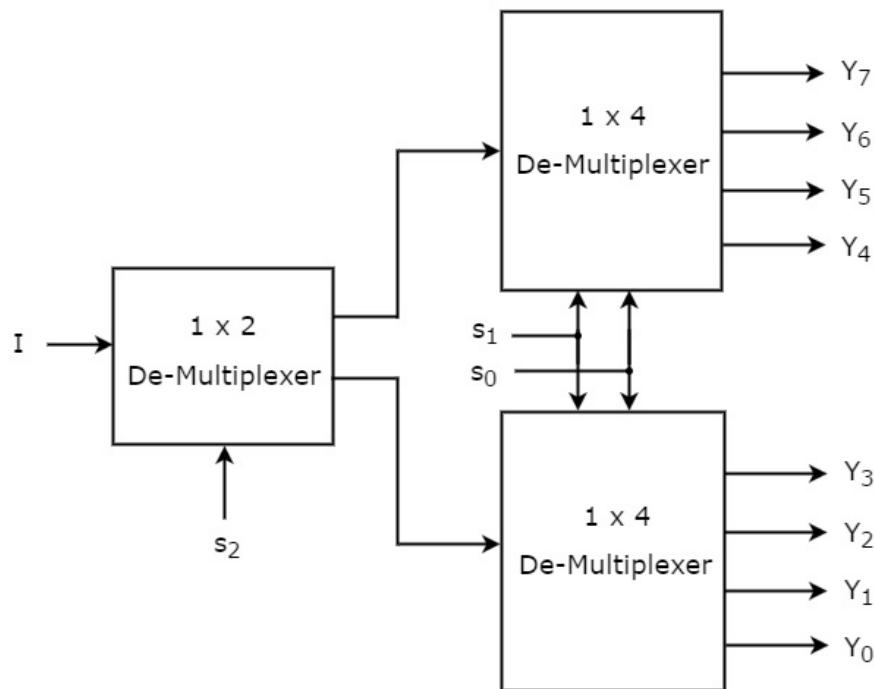
In this section, let us implement 1x8 De-Multiplexer using 1x4 De-Multiplexers and 1x2 De-Multiplexer. We know that 1x4 De-Multiplexer has single input, two selection lines and four outputs. Whereas, 1x8 De-Multiplexer has single input, three selection lines and eight outputs.

So, we require two **1x4 De-Multiplexers** in second stage in order to get the final eight outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage. Input of this 1x2 De-Multiplexer will be the overall input of 1x8 De-Multiplexer.

Let the 1x8 De-Multiplexer has one input I, three selection lines s_2, s_1 & s_0 and outputs Y_7 to Y_0. The **Truth table** of 1x8 De-Multiplexer is shown below.

Selection Inputs			Outputs							
s_2	s_1	s_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

We can implement 1x8 De-Multiplexer using lower order Multiplexers easily by considering the above Truth table. The **block diagram** of 1x8 De-Multiplexer is shown in the following figure.

The common **selection lines, s_1 & s_0** are applied to both 1x4 De-Multiplexers. The outputs of upper 1x4 De-Multiplexer are Y_7 to Y_4 and the outputs of lower 1x4 De-Multiplexer are Y_3 to Y_0.The other **selection line, s_2** is applied to 1x2 De-Multiplexer. If s_2 is zero, then one of the four outputs of lower 1x4 De-Multiplexer will be equal to input, I based on the values of selection
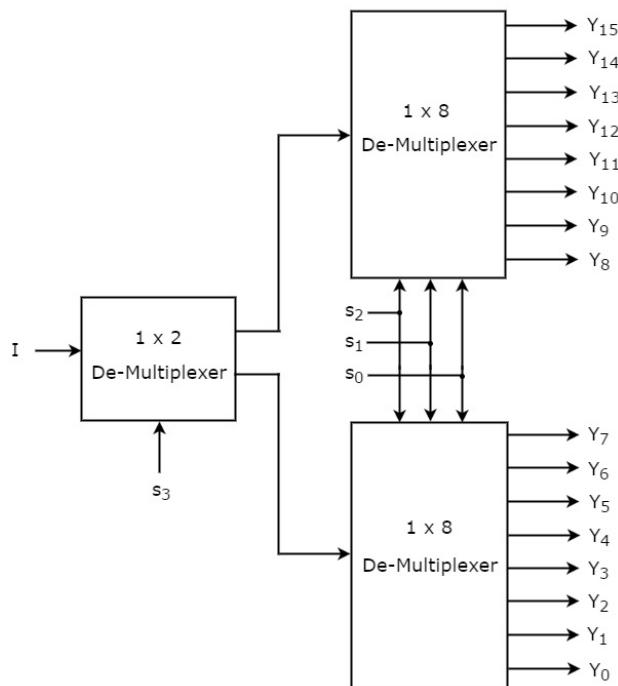
lines s_1 & s_0. Similarly, if s_2 is one, then one of the four outputs of upper 1x4 DeMultiplexer will be equal to input, I based on the values of selection lines s_1 & s_0.

1x16 De-Multiplexer

In this section, let us implement 1x16 De-Multiplexer using 1x8 De-Multiplexers and 1x2 De-Multiplexer. We know that 1x8 De-Multiplexer has single input, three selection lines and eight outputs. Whereas, 1x16 De-Multiplexer has single input, four selection lines and sixteen outputs.

So, we require two **1x8 De-Multiplexers** in second stage in order to get the final sixteen outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage. Input of this 1x2 De-Multiplexer will be the overall input of **1x16 De-Multiplexer.**

Let the 1x16 De-Multiplexer has one input I, four selection lines s_3, s_2, s_1 & s_0 and outputs Y_{15} to Y_0. The **block diagram** of 1x16 De-Multiplexer using lower order Multiplexers is shown in the following figure.

The common **selection lines s_2, s_1 & s_0** are applied to both 1x8 De-Multiplexers. The outputs of upper 1x8 De-Multiplexer are Y_{15} to Y_8 and the outputs of lower 1x8 DeMultiplexer are Y_7 to Y_0. The other **selection line, s_3** is applied to 1x2 De-Multiplexer. If s_3 is zero, then one of the eight outputs of lower 1x8 De-Multiplexer will be equal to input, I based on the values of selection lines s_2, s_1 & s_0. Similarly, if s3 is one, then one of the 8 outputs of upper 1x8 De-Multiplexer will be equal to input, I based on the values of selection lines s_2, s_1 & s_0

Applications of Demultiplexer:

1. Demultiplexer is used to connect a single source to multiple destinations. The main application area of demultiplexer is communication system where multiplexer are used. Most of the communication system are bidirectional i.e. they function in both ways (transmitting and receiving signals). Hence, for most of the applications, the multiplexer and demultiplexer work in sync. Demultiplexer is also used for reconstruction of parallel data and ALU circuits.

2. Communication System - Communication system use multiplexer to carry multiple data like audio, video and other form of data using a single line for transmission. This process makes the transmission easier. The demultiplexer receives the output signals of the multiplexer and converts them back to the original form of the data at the receiving end. The multiplexer and demultiplexer work together to carry out the process of transmission and reception of data in communication system.

3. ALU (Arithmetic Logic Unit) – In an ALU circuit, the output of ALU can be stored in multiple registers or storage units with the help of demultiplexer. The output of ALU is fed as the data input to the demultiplexer. Each output of demultiplexer is connected to multiple register which can be stored in the registers.

4. Serial to parallel converter - A serial to parallel converter is used for reconstructing parallel data from incoming serial data stream. In this technique, serial data from the incoming serial data stream is given as data input to the demultiplexer at the regular intervals. A counter is attaching to the control input of the demultiplexer. This counter directs the data signal to the output of the demultiplexer where these data signals are stored. When all data signals have been stored, the output of the demultiplexer can be retrieved and read out in parallel.

Hazards In Combinational Logic Circuits:

A hazard, if exists, in a digital circuit causes a temporary fluctuation in output of the circuit. In other words, a hazard in a digital circuit is a temporary disturbance in ideal operation of the circuit which if given some time, gets resolved itself. These disturbances or fluctuations occur when different paths from the input to output have different delays and due to this fact, changes in input variables do not change the output instantly but do appear at output after a small delay caused by the circuit building elements, i.e., logic gates.
There are three different kinds of hazards found in digital circuits
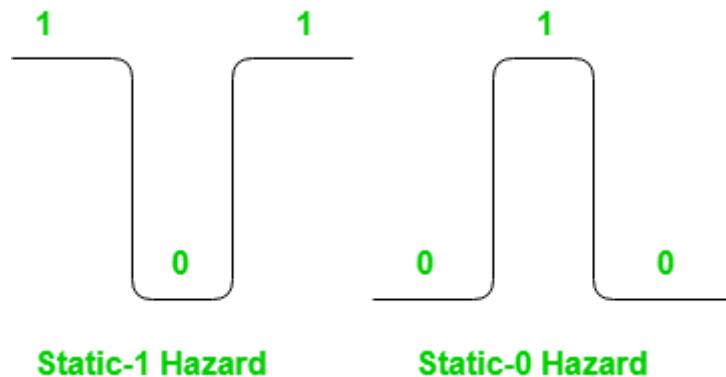
1. Static hazard
2. Dynamic hazard
3. Functional hazard

We will discuss only static hazards here to understand it completely.

Formally, a static hazard takes place when change in an input causes the output to change momentarily before stabilizing to its correct value. Based on what is the correct value, there are two types of static hazards, as shown below in the image:

- Static-1 Hazard: If the output is currently at logic state 1 and after the input changes its state, the output momentarily changes to 0 before settling on 1, then it is a Static-1 hazard.
- Static-0 Hazard: If the output is currently at logic state 0 and after the input changes its state, the output momentarily changes to 1 before settling on 0, then it is a Static-0 hazard.

Static-1 Hazard   **Static-0 Hazard**
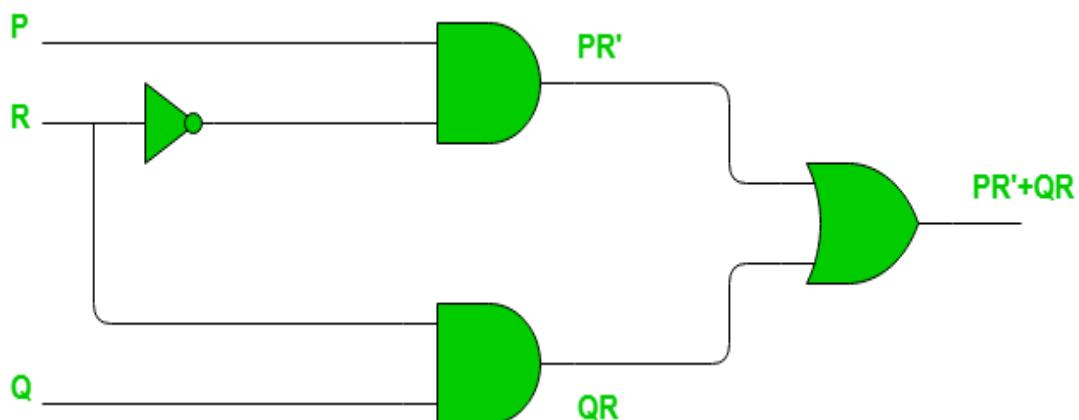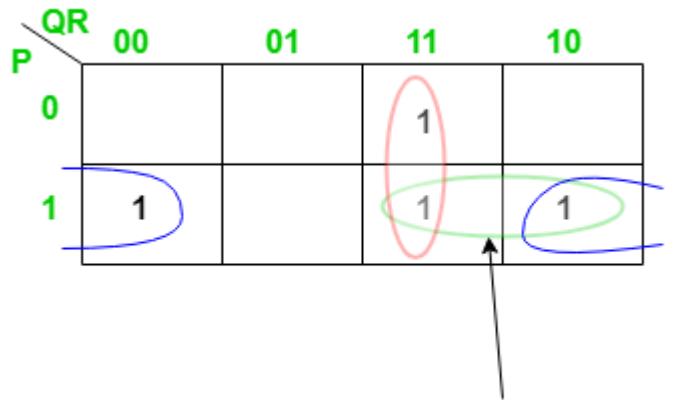
Detection of Static hazards using K-map:

Lets consider static-1 hazard first. To detect a static-1 hazard for a digital circuit following steps are used:

- **Step-1:** Write down the output of the digital circuit, say **Y**.
- **Step-2:** Draw the K-map for this function **Y** and note all adjacent 1's.
- **Step-3:** If there exists any pair of cells with 1's which do not occur to be in the same group (i.e. prime implicant), it indicates the presence of a static-1 hazard. Each such pair is a static-1 hazard. Lets have an example:

Example – Consider the circuit shown below.
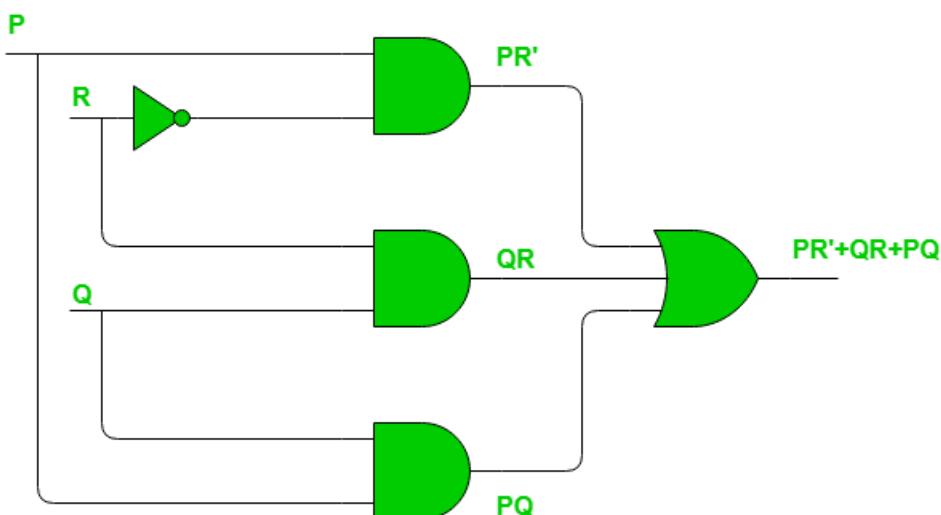
The pair of 1's encircled as green are not part of the grouping/pairing provided by the output of this Boolean function. This will cause a static-1 hazard in this circuit.

Removalofstatic-1hazard:

Once detected, a static-1 hazard can be easily removed by introducing some more terms (logic gates) to the function (circuit). The most common idea is to add the missing group in the existing Boolean function, as adding this term would not affect the function by any mean but it will remove the hazard. Since in above example the pair of 1's encircled with blue color causes the static-1 hazard, we just add this as a prime implicant to the existing function as follows:

Note that there is no difference in number of minterms of this function. The reason is that the static-1 hazards are based on how we group 1's (or 0's for static-0 hazard) for a given set of 1's in K-map. Thus it does not make any difference in number of 1's in K-map. The circuit would look like as shown below with the change made for removal of static-1 hazard.

Similarly for **Static-0 Hazards** we need to consider 0's instead of 1's and if any adjacent 0's in K-map are not grouped into same group that may cause a static-0 hazard. The method to detect and resolve the static-0 hazard is completely same as the one we followed for static-1 hazard except that instead of SOP, POS will be used as we are dealing with 0's in this case.
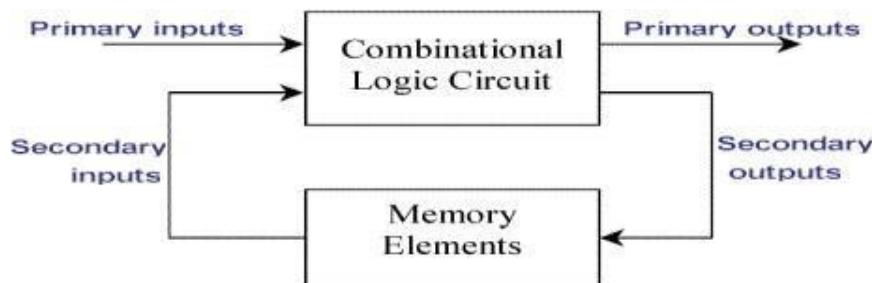
SEQUENTIAL CIRCUITS &FUNDAMENTALS

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

1. Synchronous sequentialcircuits
2. Asynchronous sequentialcircuits

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

Comparison between combinational and sequential circuits

Combinational circuit	Sequential circuit
1. In combinational circuits, the output variables at any instant of time are dependent only on the present input Variables	1. in sequential circuits the output variables at any instant of time are dependent not only on the present input variables, but also on the present state
2.memory unit is not requires in combinational circuit	2.memory unit is required to store the past history of the input variables

3. these circuits are faster because the delay between the i/p and o/p	3. sequential circuits are slower than combinational circuits due to propagation delay of gates only
4. easy to design	4. comparatively hard to design

Level mode and pulse mode asynchronous sequential circuits:
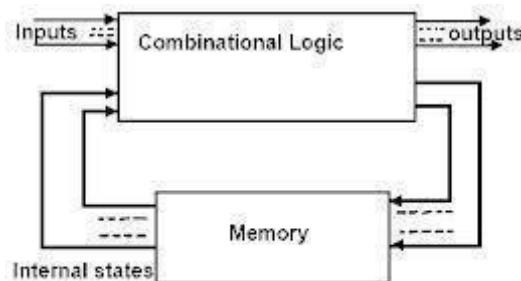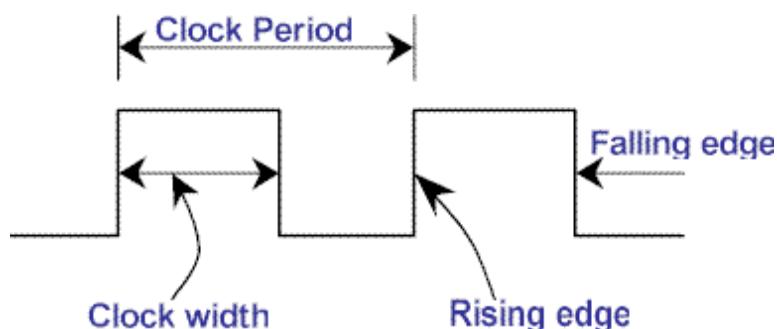
Figure 1: Asynchronous Sequential Circuit

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

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

Synchronous and Asynchronous Operation:

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

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

The reciprocal of the clock period is referred to as the **clock frequency**. The **clock width** is defined as the time during which the value of the clock signal is equal to 1. The ratio of the clock width and clock period is referred to as the duty cycle. A clock signal is said to be **active high** if the state changes occur at the clock's rising edge or during the clock width. Otherwise, the clock is said to be **active low**. Synchronous sequential circuits are also known as **clocked sequential circuits**.

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

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

Fundamental Mode Circuits assumes that:

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

A pulse mode circuit assumes that:

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

Latches and flip-flops

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

There are basically four main types of latches and flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state.

For each type, there are also different variations that enhance their operations. In this chapter, we will look at the operations of the various latches and flip-flops.the flip-flops has two outputs, labeled Q and Q'. the Q output is the normal output of the flip flop and Q' is the inverted output.
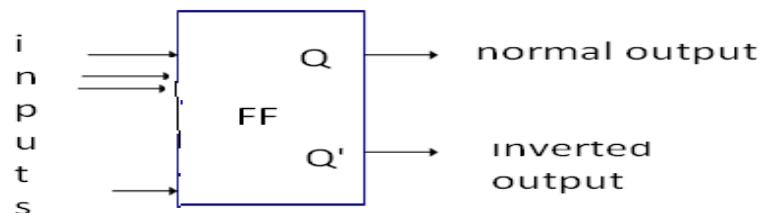
Figure: basic symbol of flipflop

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

SR latch:

S	R	Q	$\overline{Q}$	Function
0	0	Q^+	$\overline{Q}^+$	Storage State
0	1	0	1	Reset
1	0	1	0	Set
1	1	0-?	0-?	Indeterminate State
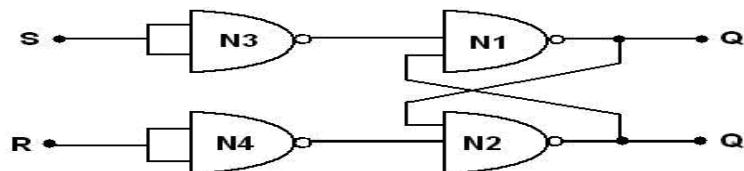
The latch has two outputs Q and Q'. When the circuit is switched on the latch may enter into any state. If Q=1, then Q'=0, which is called SET state. If Q=0, then Q'=1, which is called

RESET state. Whether the latch is in SET state or RESET state, it will continue to remain in the same state, as long as the power is not switched off. But the latch is not an useful circuit, since there is no way of entering the desired input. It is the fundamental building block in constructing flip-flops, as explained in the following sections
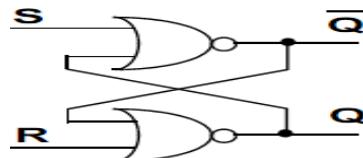
NAND latch

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

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


NOR latch

The analysis of the operation of the active-HIGH NOR latch can be summarized as follows.

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

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

RS Flip-flop:

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

a) Logic diagram
b) Block diagram
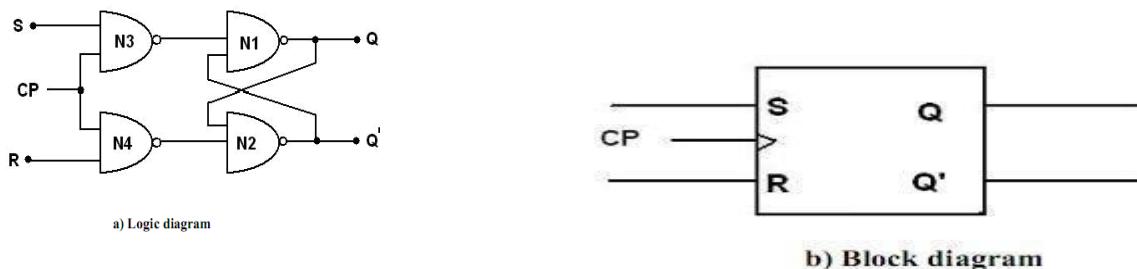
Figure: RS Flip-flop

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

Operation:

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

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

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

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

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

6. CP=1, S=1, R=1 is not allowed, because it is not able to determine the next state. This condit ion is said to be a —race condition.
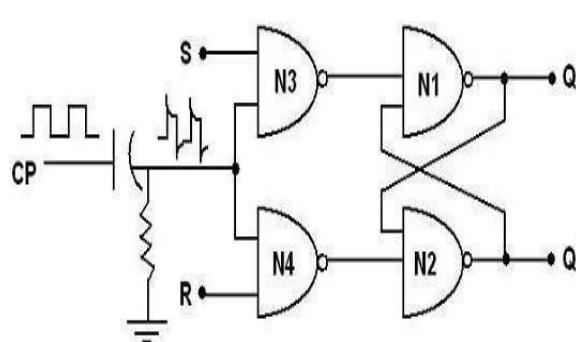
In the logic symbol CP input is marked with a triangle. It indicates the circuit responds to an input change from 0 to 1. The characteristic table gives the operation conditions of flip-flop. Q(t) is the present state maintained in the flip-flop at time _t'. Q(t+1) is the state after the occurrence of clock pulse.

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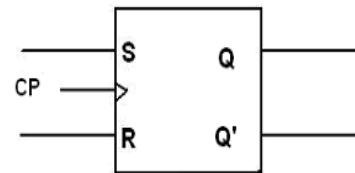
S	R	$Q_{(t+1)}$	Comments
0	0	Q_t	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	*	Not allowed

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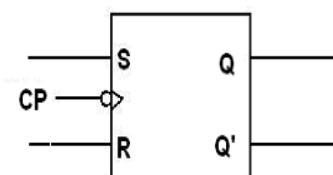
Some flip-flops have an RC circuit at the input next to the clock pulse. By the design of the circuit the R-C time constant is much smaller than the width of the clock pulse. So the output changes will occur only at specific level of clock pulse. The capacitor gets fully charged when clock pulse goes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

a) Logic diagram of edge triggered RS flip-flop

b) Block diagram of positive edge triggered flip-flop
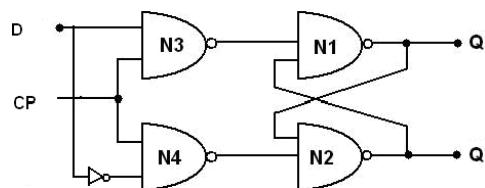
c) Block diagram of negative edge triggered flip-flop

135
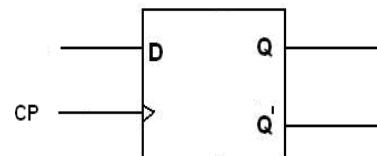
Figure: Edge triggered RS flip-flo

D flip-flop:

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


a) Logic diagram
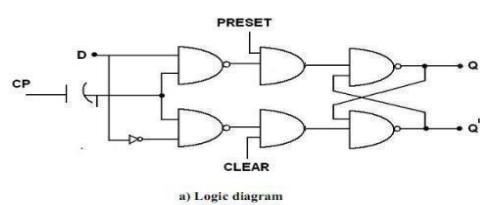
b) Block diagram

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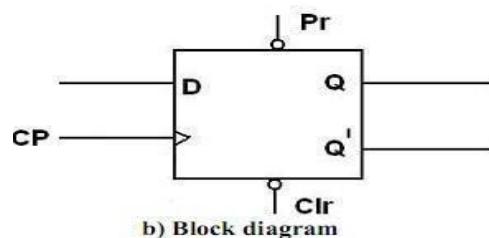
Truth table

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

Edge Triggered D Flip-flop:

a) Logic diagram
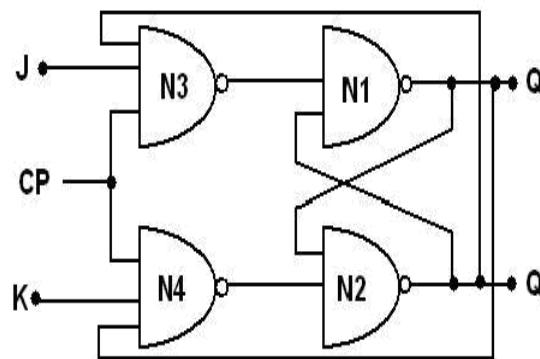
b) Block diagram

Truth table

PRESET	CLEAR	CP	D	Q
0	0	X	X	*(forbidden)
0	1	X	X	1
1	0	X	X	0
1	0	0	X	NC
1	1	1	X	NC
1	1	↓	X	NC
1	1	↑	0	0
1	1	↑	1	1
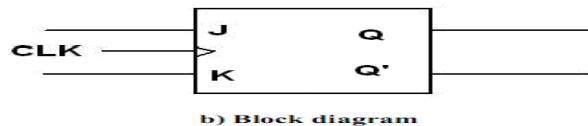
Figure: truth table, block diagram, logic diagram of edge triggered flip-flop
JK flip-flop (edge triggered JK flip-flop)

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

a) Logic diagram

Figure: JK flip-flop

b) Block diagram

Truth table

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

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

Operation:

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

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

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

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

The characteristic equation of the JK flip-flop is:

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

JK flip-flop operation								
Characteristic table				**Excitation table**				
J	**K**	**Q_{next}**	**Comment**	**Q**	**Q_{next}**	**J**	**K**	**Comment**
0	0	Q	hold state	0	0	0	X	No change
0	1	0	reset	0	1	1	X	Set
1	0	1	set	1	0	X	1	Reset
1	1	Q	toggle	1	1	X	0	No change

T flip-flop:

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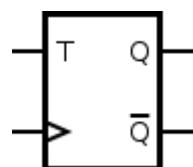
Figure : symbol for T flip flop

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

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

T flip-flop operation[28]							
Characteristic table				**Excitation table**			
T	Q	Q_{next}	Comment	Q	Q_{next}	T	Comment
0	0	0	hold state (no clk)	0	0	0	No change
0	1	1	hold state (no clk)	1	1	0	No change
1	0	1	Toggle	0	1	1	Complement
1	1	0	Toggle	1	0	1	Complement

Flip flop operating characteristics:

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

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

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

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

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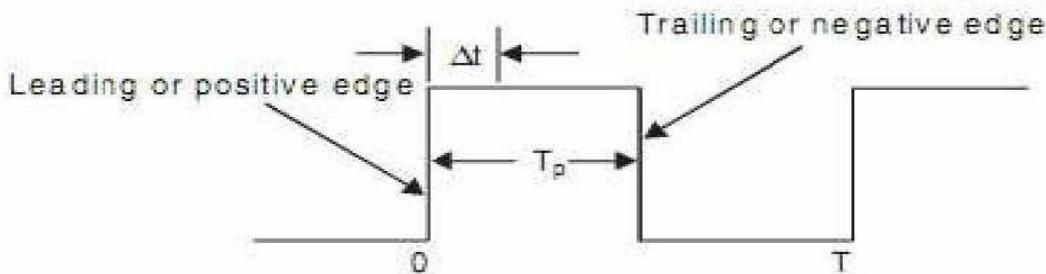
Dissipation: is the total power consumption of the device. It is equal to product of supply voltage (Vcc) and the current (Icc).

$$P = V_{cc} \cdot I_{cc}$$

The power dissipation of a flip flop is usually in mW

Clock transition times: for reliable triggering, the clock waveform transition times should be kept very short. If the clock signal takes too long to make the transitions from one level to other, the flip flop may either triggering erratically or not trigger at all. Race around Condition

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

- Consider, for example, that the inputs are $J = K = 1$ and $Q = 1$, and a pulse as shown in Figure is applied at the clock input.
- After a time interval t equal to the propagation delay through two NAND gates in series, the outputs will change to $Q = 0$. So now we have $J = K = 1$ and $Q = 0$.
- After another time interval of t the output will change back to $Q = 1$. Hence, we conclude that for the time duration of tP of the clock pulse, the output will oscillate between 0 and 1. Hence, at the end of the clock pulse, the value of the output is not certain. This situation is referred to as a race-around condition.
- Generally, the propagation delay of TTL gates is of the order of nanoseconds. So if the clock pulse is of the order of microseconds, then the output will change thousands of times within the clockpulse.
- This race-around condition can be avoided if tp< t < T. Due to the small propagation delay of the ICs it may be difficult to satisfy the above condition.
- A more practical way to avoid the problem is to use the master-slave (M-S) configuration as discussed below.

. Excitation Tables:

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

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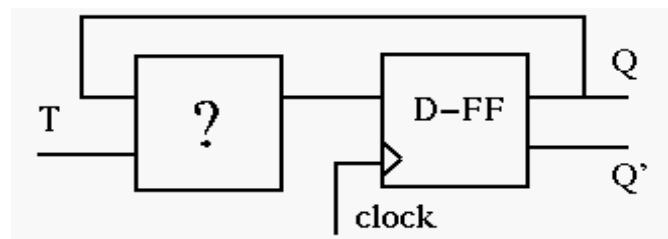
Previous State -> Present State	S	R
0 -> 0	0	X
0 -> 1	1	0
1 -> 0	0	1
1 -> 1	X	0

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

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

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

Convert a D-FF to a T-FF:
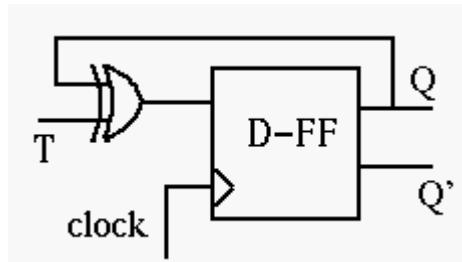
We need to design the circuit to generate the triggering signal D as a function of T and Q:

$$D = f(T, Q).$$

. Consider the excitation table:

Q_t	Q_{t+1}	T	D
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	1
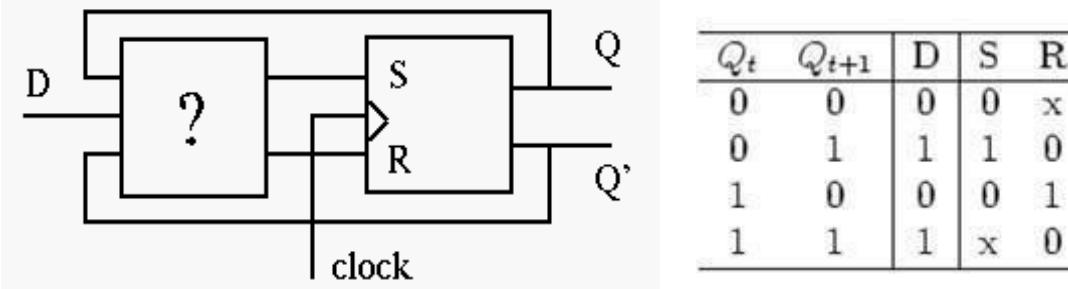
Treating as a function of and current FF state, we have
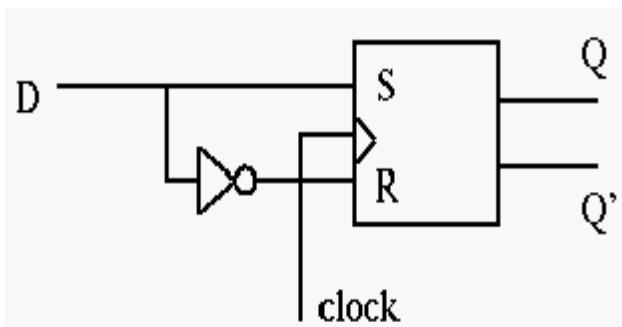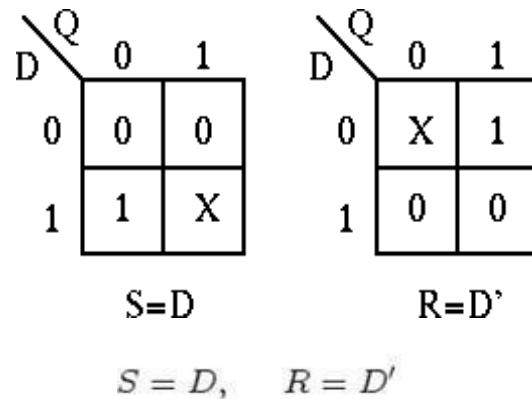
$$D = T'Q + TQ' = T \oplus Q$$

Convert a RS-FF to a D-FF:

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

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

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


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

Convert a RS-FF to a JK-FF:

We need to design the circuit to generate the triggering signals S and R as functions of, J,K.Consider the excitation table: The desired signal and   as functions of,   and current FF state can be obtained from the Karnaugh maps:

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

K-maps:

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

S=Q'J
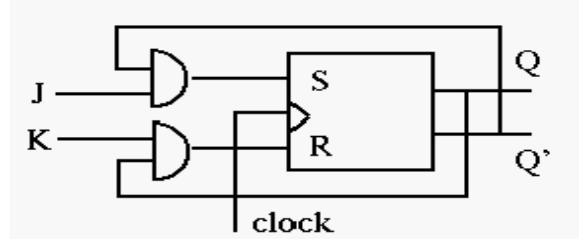
K\QJ	00	01	11	10
0	X	0	0	0
1	X	0	1	1

R=QK

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

Sequential Circuit Design

1. Specification – A description of the sequential circuit. Should include a detailing of the inputs, the outputs, and the operation. Possibly assumes that you have knowledge of digital system basics.
2. Formulation: Generate a state diagram and/or a state table from the statement of the problem.
3. State Assignment: From a state table assign binary codes to the states.
4. Flip-flop Input Equation Generation: Select the type of flip-flop for the circuit and generate the needed input for the required statetransitions
5. Output Equation Generation: Derive output logic equations for generation of the output from the inputs and current state.
6. Optimization: Optimize the input and output equations. Today, CAD systems are typically

used for this in real systems.

7. Technology Mapping: Generate a logic diagram of the circuit using ANDs, ORs, Inverters, and F/Fs.
8. Verification: Use a HDL to verify the design.

Shift registers:

In digital circuits, a **shift register** is a cascade of flip-flops sharing the same clock, in which the output of each flip-flop is connected to the "data" input of the next flip-flop in the chain, resulting in a circuit that shifts by one position the "bit array" stored in it, *shifting in* the data present at its input and *shifting out* the last bit in the array, at each transition of the clock input. More generally, a **shift register** may be multidimensional, such that its "data in" and stage outputs are themselves bit arrays: this is implemented simply by running several shift registers of the same bit-length in parallel.

Shift registers can have both parallel and serial inputs and outputs. These are often configured as **serial-in, parallel-out** (SIPO) or as **parallel-in, serial-out** (PISO). There are also types that have both serial and parallel input and types with serial and parallel output. There are also **bi- directional** shift registers which allow shifting in both directions: L→R or R→L. The serial input and last output of a shift register can also
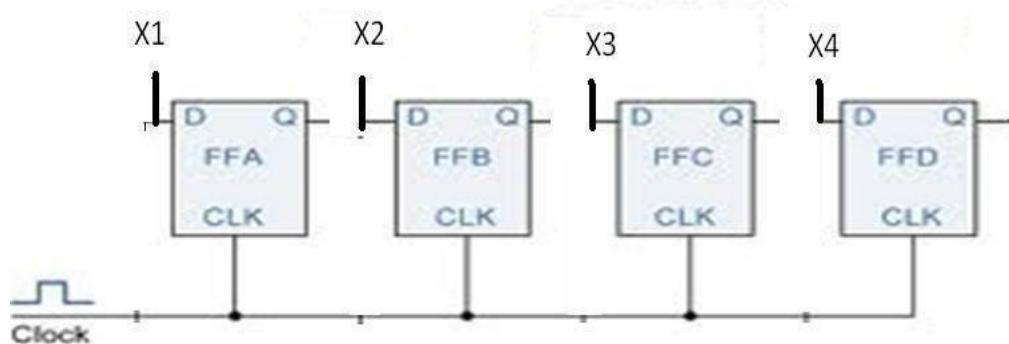
Figure: logic diagram of 4-bit buffer register

The figure shows a 4-bit buffer register. The binary word to be stored is applied to the data terminals. On the application of clock pulse, the output word becomes the same as the word applied at the terminals. i.e., the input word is loaded into the register by the application of clock pulse.

When the positive clock edge arrives, the stored word becomes:

$$Q4Q3Q2Q1=X4X3X2X1$$
$$Q=X$$

Controlled buffer register:

If ☐☐☐goes LOW, all the FFs are RESET and the output becomes, Q=0000.
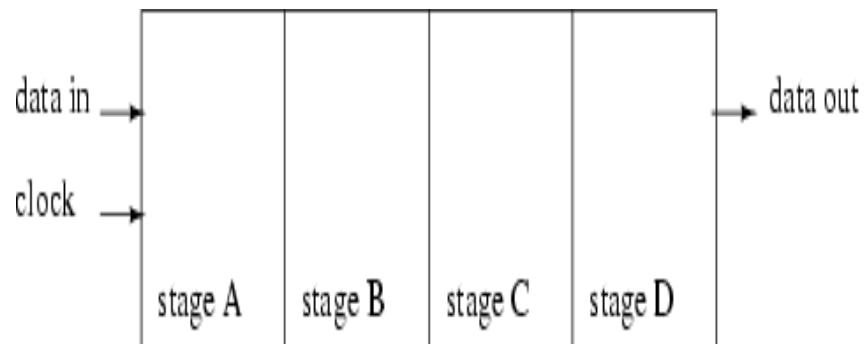
When ☐☐☐ is HIGH, the register is ready for action. LOAD is the control input. When LOAD is HIGH, the data bits X can reach the D inputs of FF's.

Q4Q3Q2Q1=X4X3X2X1
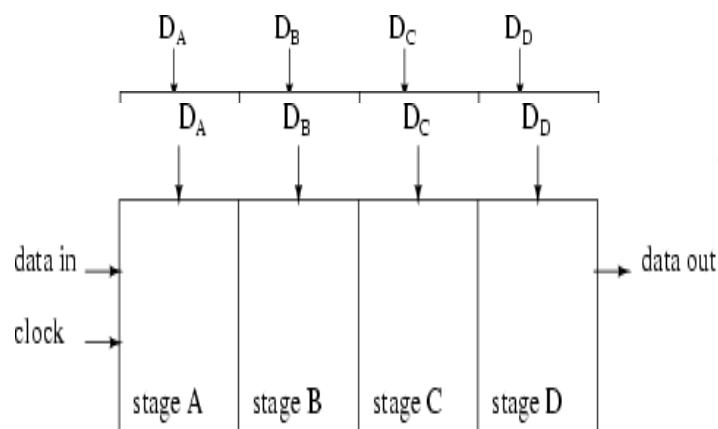
Q=X

When load is low, the X bits cannot reach the FF's.

Data transmission in shift registers:

Serial-in, serial-out shift register with 4-stages
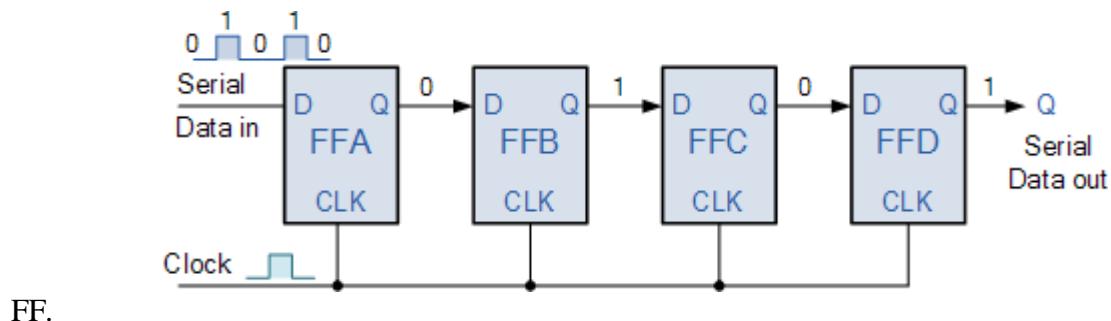
Parallel-in, serial-out shift register with 4-stages

A number of ff's connected together such that data may be shifted into and shifted out of them is called shift register. data may be shifted into or out of the register in serial form or in parallel form. There are four basic types of shift registers.

1. Serial in, serial out, shift right, shift registers
2. Serial in, serial out, shift left, shift registers
3. Parallel in, serial out shift registers
4. Parallel in, parallel out shift registers
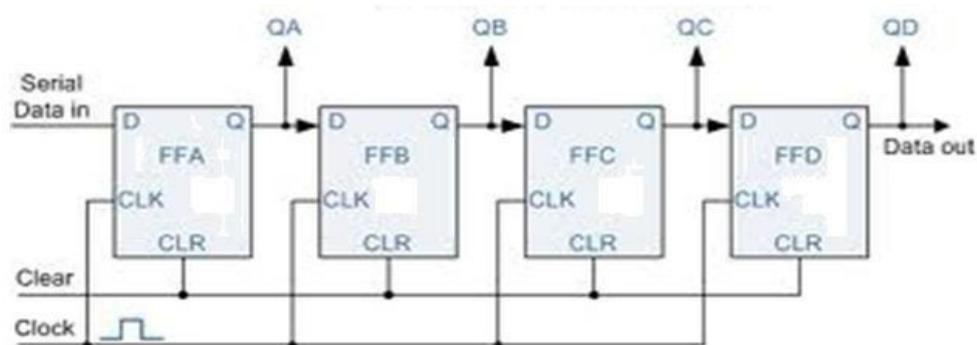
Serial IN, serial OUT, shift right, shift left register:

The logic diagram of 4-bit serial in serial out, right shift register with four stages. The register can store four bits of data. Serial data is applied at the input D of the first FF. the Q output of the first FF is connected to the D input of another FF. the data is outputted from the Q terminal of the last

FF.

When serial data is transferred into a register, each new bit is clocked into the first FF at the positive going edge of each clock pulse. The bit that was previously stored by the first FF is transferred to the second FF. the bit that was stored by the Second FF is transferred to the third FF.

Serial-in, parallel-out, shift register:
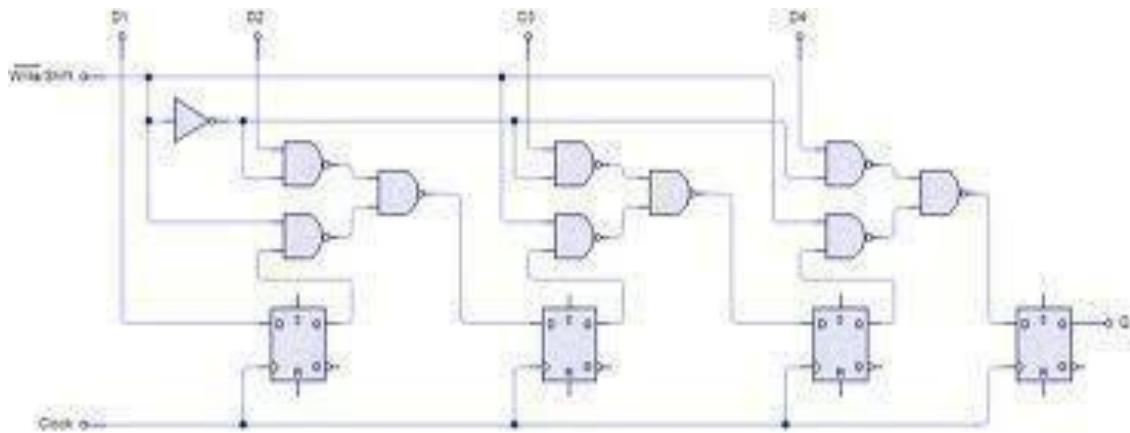
In this type of register, the data bits are entered into the register serially, but the data stored in the

register is shifted out in parallel form.

Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously, rather than on a bit-by-bit basis with the serial output. The serial-in, parallel out, shift register can be used as serial-in, serial out, shift register if the output is taken from the Q terminal of the last FF.
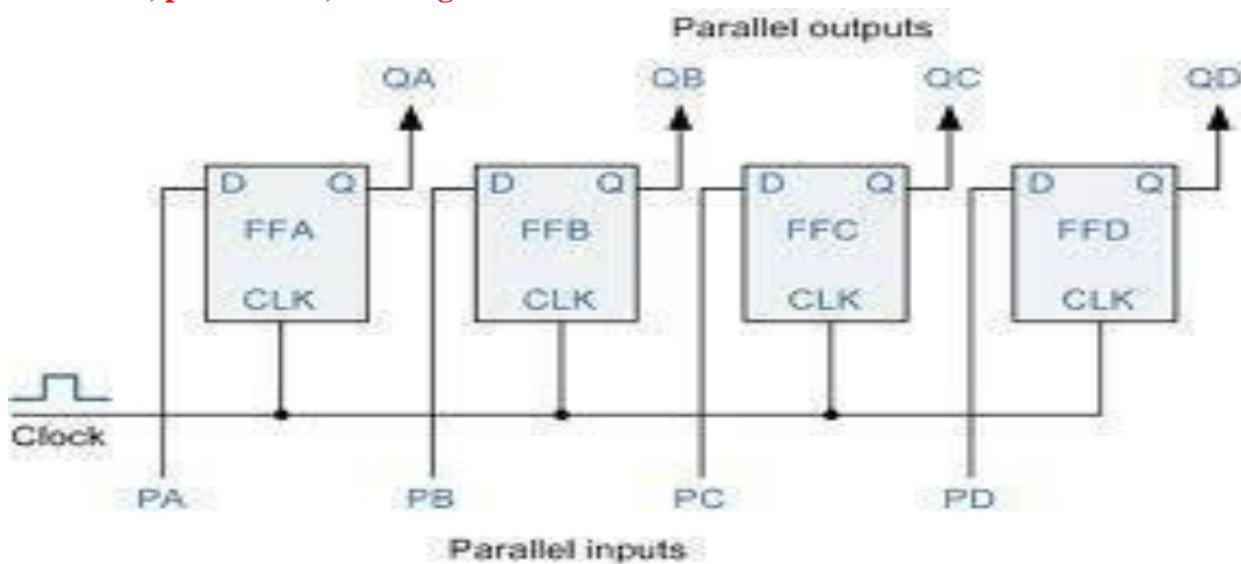
Parallel-in, serial-out, shift register:

For a parallel-in, serial out, shift register, the data bits are entered simultaneously into their respective stages on parallel lines, rather than on a bit-by-bit basis on one line as with serial data bits are transferred out of the register serially. On a bit-by-bit basis over a single line.

There are four data lines A,B,C,D through which the data is entered into the register in parallel form. The signal shift/ load allows the data to be entered in parallel form into the register and the data is shifted out serially from terminalQ4

Parallel-in, parallel-out, shift register
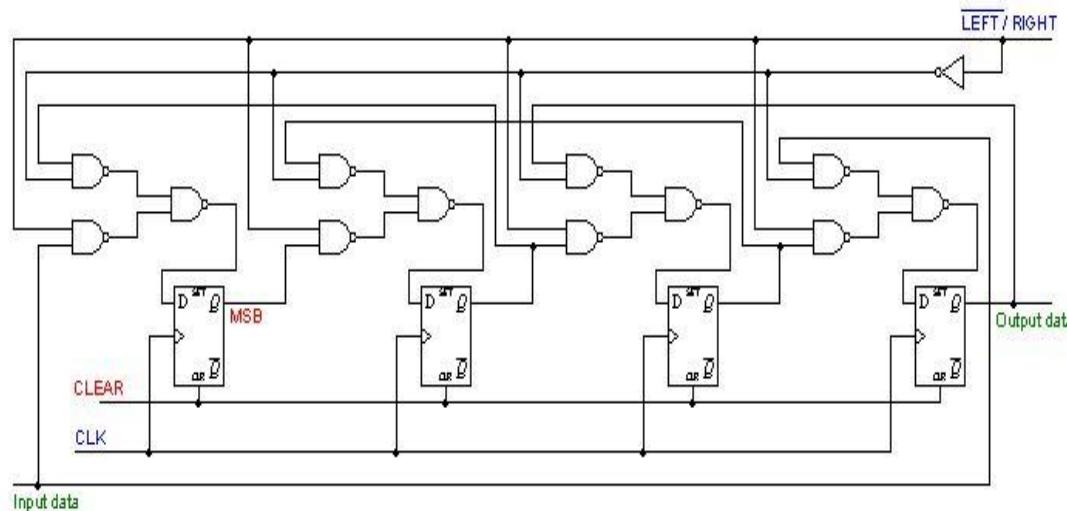
In a parallel-in, parallel-out shift register, the data is entered into the register in parallel form, and also the data is taken out of the register in parallel form. Data is applied to the D input terminals of the FF's. When a clock pulse is applied, at the positive going edge of the pulse, the D inputs are shifted into the Q outputs of the FFs. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form.

Bidirectional shift register:

A bidirectional shift register is one which the data bits can be shifted from left to right or from right to left. A fig shows the logic diagram of a 4-bit serial-in, serial out, bidirectional shift register. Right/left is the mode signal, when right /left is a 1, the logic circuit works as a shift-register.the bidirectional operation is achieved by using the mode signal and two NAND gates and one OR gate for each stage.

A HIGH on the right/left control input enables the AND gates G1, G2, G3 and G4 and disables the AND gates G5,G6,G7 and G8, and the state of Q output of each FF is passed through the gate to the D input of the following FF. when a clock pulse occurs, the data bits are then effectively shifted one place to the right. A LOW on the right/left control inputs enables the AND gates G5, G6, G7 and G8 and disables the And gates G1, G2, G3 and G4 and the Q output of each FF is passed to the D input of the preceding FF. when a clock pulse occurs, the data bits are then

effectively shifted one place to the left. Hence, the circuit works as a bidirectional shift register

Figure: logic diagram of a 4-bit bidirectional shift register

Universal shift register:

A register is capable of shifting in one direction only is a unidirectional shift register. One that can shift both directions is a bidirectional shift register. If the register has both shifts and parallel load capabilities, it is referred to as a universal shift registers. Universal shift register is a bidirectional register, whose input can be either in serial form or in parallel form and whose output also can be in serial form or I parallel form.

 The most general shift register has the following capabilities.

149

1. A clear control to clear the register to 0
2. A clock input to synchronize theoperations
3. A shift-right control to enable the shift-right operation and serial input and output lines associated with the shift-right. A shift-left control to enable the shift-left operation and serial input and outputlines associated with theshift-left
4. A parallel loads control to enable a parallel transfer and the n input lines associated with the parallel transfer
5. N parallel output lines
6. A control state that leaves the information in the register unchanged in the presence of the clock.

A universal shift register can be realized using multiplexers. The below fig shows the logic diagram of a 4-bit universal shift register that has all capabilities. It consists of 4 D flip-flops and four multiplexers. The four multiplexers have two common selection inputs s1 and s0. Input 0 in each multiplexer is selected when S1S0=00, input 1 is selected when S1S0=01 and input 2 is selected when S1S0=10 and input 4 is selected when S1S0=11. The selection inputs control the mode of operation of the register according to the functions entries. When S1S0=0, the present value of the register is applied to the D inputs of flip-flops. The condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock edge transfers into each flip-flop the binary value it held previously, and no change of state occurs. When S1S0=01, terminal 1 of the multiplexer inputs have a path to the D inputs of the flip-flop. This causes a shift-right operation, with serial input transferred into flip-flopA4. When S1S0=10, a shift left operation results with the other serial input going into flip-flop A1. Finally when S1S0=11, the binary information on the parallel input lines is transferred into the register simultaneously during the

4-bit universal shift register

next clock cycle
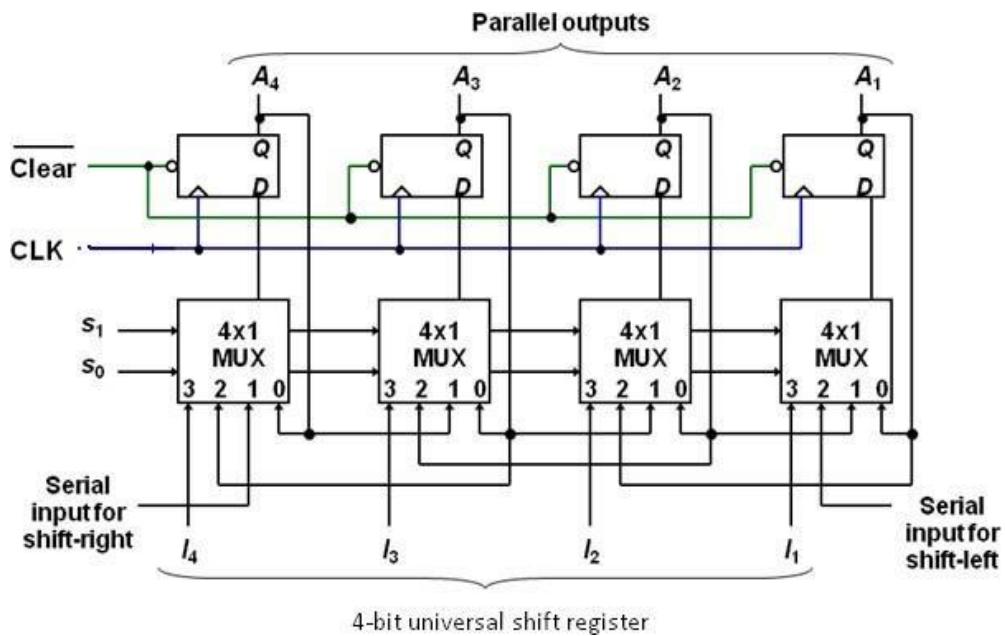
Figure: logic diagram 4-bit universal shift register

Function table for theregister

mode control		
S0	**S1**	**register operation**
0	0	No change
0	1	Shift Right
1	0	Shift left
1	1	Parallel load

Counters:

Counter is a device which stores (and sometimes displays) the number of times particular event or process has occurred, often in relationship to a clock signal. A Digital counter is a set of flip flops whose state change in response to pulses applied at the input to the counter. Counters may be asynchronous counters or synchronous counters. Asynchronous counters are also called ripple counters
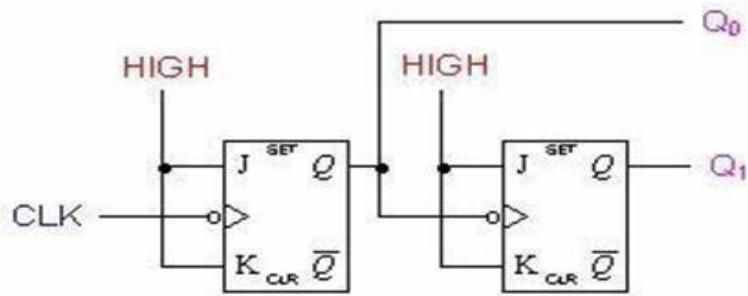
In electronics counters can be implemented quite easily using register-type circuits such as the flip-flops and a wide variety of classifications exist:

- Asynchronous (ripple) counter – changing state bits are used as clocks to subsequent state flip-flops
- Synchronous counter – all state bits change under control of a single clock
- Decade counter – counts through ten states perstage
- Up/down counter – counts both up and down, under command of a control input
- Ring counter – formed by a shift register with feedback connection in a ring

Each is useful for different applications. Usually, counter circuits are digital in nature, and count in natural binary Many types of counter circuits are available as digital building blocks, for example a number of chips in the 4000 series implement different counters.

Occasionally there are advantages to using a counting sequence other than the natural binary sequence such as the binary coded decimal counter, a linear feed-back shift register counter, or a gray-code counter.

Counters are useful for digital clocks and timers, and in oven timers, VCR clocks, etc.

Asynchronous counters:

An asynchronous (ripple) counter is a single JK-type flip-flop, with its J (data) input fed from its own inverted output. This circuit can store one bit, and hence can count from zero to one before it overflows (starts over from 0). This counter will increment once for every clock cycle and takes two clock cycles to overflow, so every cycle it will alternate between a transition from 0 to 1 and a transition from 1 to 0. Notice that this creates a new clock with a 50% duty cycle at exactly half the frequency of the input clock. If this output is then used as the clock signal for a similarly arranged D flip-flop (remembering to invert the output to the input), one will get another 1 bit counter that counts half as fast. Putting them together yields a two-bit counter:
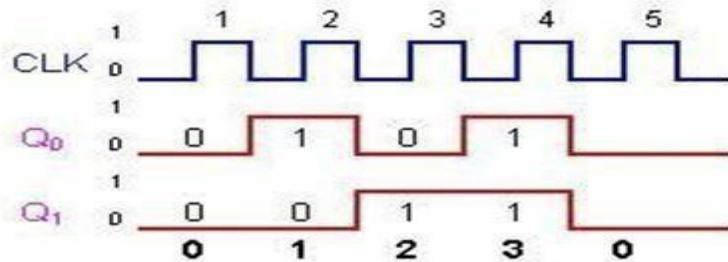
Two-bit ripple up-counter using negative edge triggered flip flop:

Two bit ripple counter used two flip-flops. There are four possible states from 2 – bit up- counting I.e. 00, 01, 10 and 11.
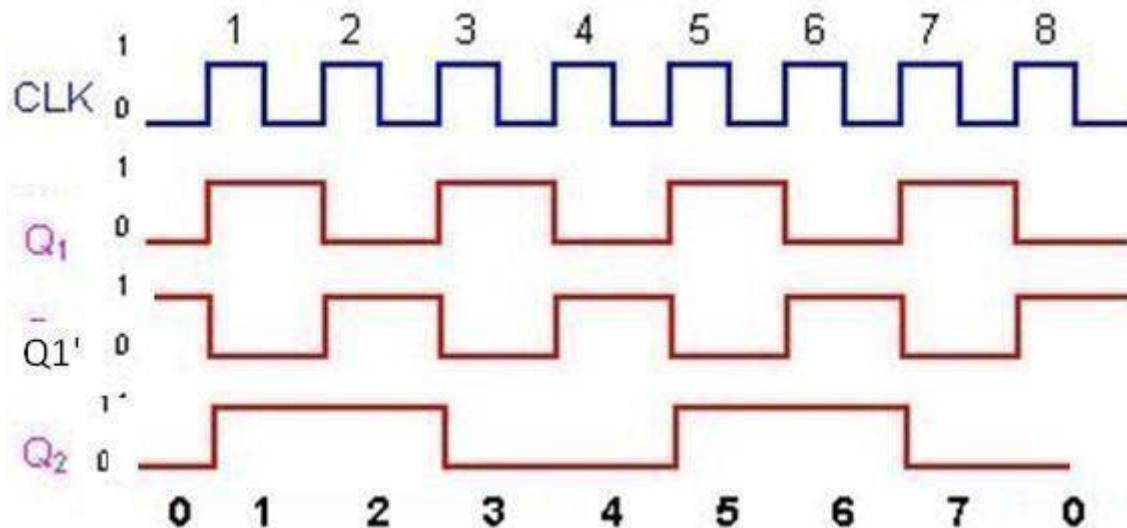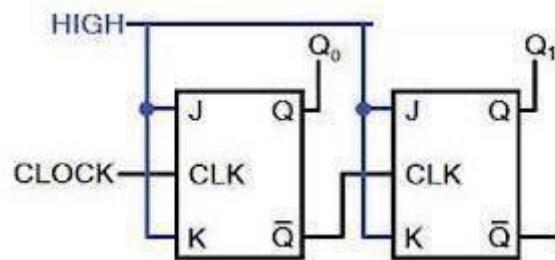
· The counter is initially assumed to be at a state 00 where the outputs of the tow flip-flops are noted as Q_1Q_0. Where Q_1 forms the MSB and Q_0 forms the LSB.

·

· For the negative edge of the first clock pulse, output of the first flip-flop FF1 toggles its state. Thus Q_1 remains at 0 and Q_0 toggles to 1 and the counter state are now read as 01.

During the next negative edge of the input clock pulse FF1 toggles and $Q_0 = 0$. The output Q_0 being a clock signal for the second flip-flop FF2 and the present transition acts as a negative edge for FF2 thus toggles its state $Q_1 = 1$. The counter state is now read as 10.

· For the next negative edge of the input clock to FF1 output Q_0 toggles to 1. But this transition from 0 to 1 being a positive edge for FF2 output Q_1 remains at 1. The counter state is now read as 11.

· For the next negative edge of the input clock, Q_0 toggles to 0. This transition from 1 to 0 acts as a negative edge clock for FF2 and its output Q_1 toggles to 0. Thus the starting state 00 is attained. Figure shown below

Two-bit ripple down-counter using negative edge triggered flip flop:

A 2-bit down-counter counts in the order 0,3,2,1,0,1…….,i.e, 00,11,10,01,00,11 …..,etc. the above fig. shows ripple down counter, using negative edge triggered J-K FFs and its timing diagram.

- For down counting, Q1' of FF1 is connected to the clock of Ff2. Let initially all the FF1 toggles, so, Q1 goes from a 0 to a 1 and Q1' goes from a 1 to a 0.

- The negative-going signal at Q1' is applied to the clock input of FF2, toggles Ff2 and, therefore, Q2 goes from a 0 to a 1.so, after one clock pulse Q2=1 and Q1=1, I.e., the state of the counter is 11.
- At the negative-going edge of the second clock pulse, Q1 changes from a 1 to a 0 and Q1' from a 0 to a 1.
- This positive-going signal at Q1' does not affect FF2 and, therefore, Q2 remains at a 1. Hence , the state of the counter after second clock pulse is 10
- At the negative going edge of the third clock pulse, FF1 toggles. So Q1, goes from a 0 to a 1 and Q1' from 1 to 0. This negative going signal at Q1' toggles FF2 and, so, Q2 changes from 1 to 0, hence, the state of the counter after the third clock pulse is 01.
- At the negative going edge of the fourth clock pulse, FF1 toggles. So Q1, goes from a 1 to a 0 and Q1' from 0 to 1. . This positive going signal at Q1' does not affect FF2 and, so, Q2 remains at 0, hence, the state of the counter after the fourth clock pulse is 00.

Two-bit ripple up-down counter using negative edge triggered flip flop:
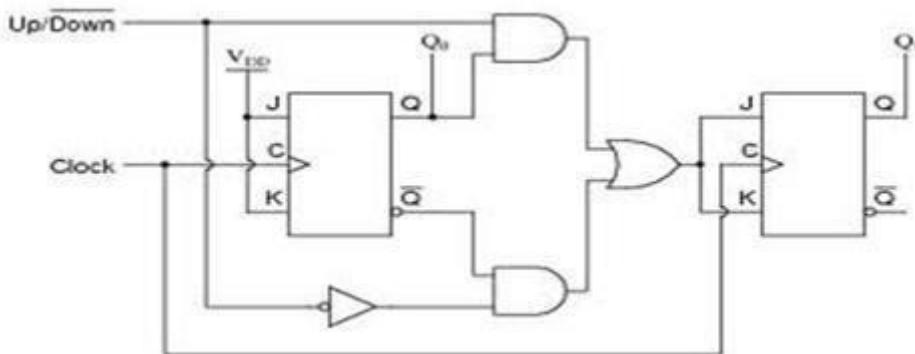
Figure: asynchronous 2-bit ripple up-down counter using negative edge triggered flip flop

- As the name indicates an up-down counter is a counter which can count both in upward and downward directions. An up-down counter is also called a forward/backward counter or a bidirectional counter. So, a control signal or a mode signal M is required to choose the direction of count. When M=1 for up counting, Q1 is transmitted to clock of FF2 and when M=0 for down counting, Q1' is transmitted to clock of FF2. This is achieved by using two AND gates and one OR gates. The external clock signal is applied to FF1.
- Clock signal to FF2= (Q1.Up)+(Q1'. Down)=Q1m+Q1'M'
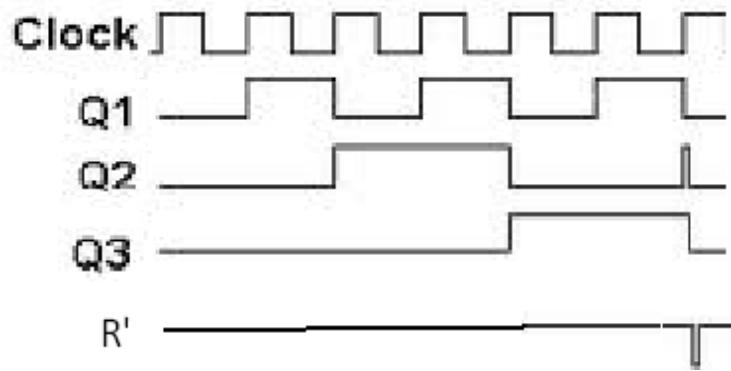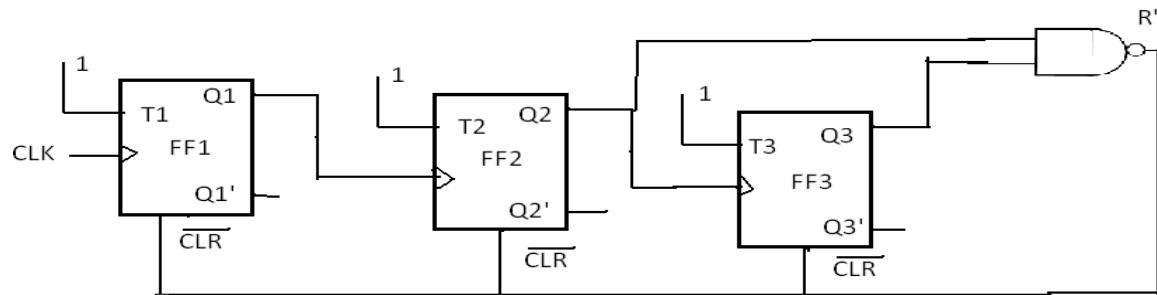
Design of Asynchronous counters:

To design a asynchronous counter, first we write the sequence , then tabulate the values of reset signal R for various states of the counter and obtain the minimal expression for R and R' using K-Map or any other method. Provide a feedback such that R and R' resets all the FF's after

the desired count

Design of a Mod-6 asynchronous counter using T FFs:

A mod-6 counter has six stable states 000, 001, 010, 011, 100, and 101. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided. it is –divide by-6-counter‖, in the sense that it divides the input clock frequency by 6.it requires three FFs, because the smallest value of n satisfying the condition $N \leq 2^n$ is n=3; three FFs can have 8 possible states, out of which only six are utilized and the remaining two states 110and 111, are invalid. If initially the counter is in 000 state, then after the sixth clock pulse, it goes to 001, after the second clock pulse, it goes to 010, and so on.

After sixth clock pulse it goes to 000. For the design, write the truth table with present state outputs Q3, Q2 and Q1 as the variables, and reset R as the output and obtain an expression for R in terms of Q3, Q2, and Q1that decides the feedback into be provided. From the truth table, R=Q3Q2. For active-low Reset, R' is used. The reset pulse is of very short duration, of the order of nanoseconds and it is equal to the propagation delay time of the NAND gate used. The expression for R can also be determined as follows.
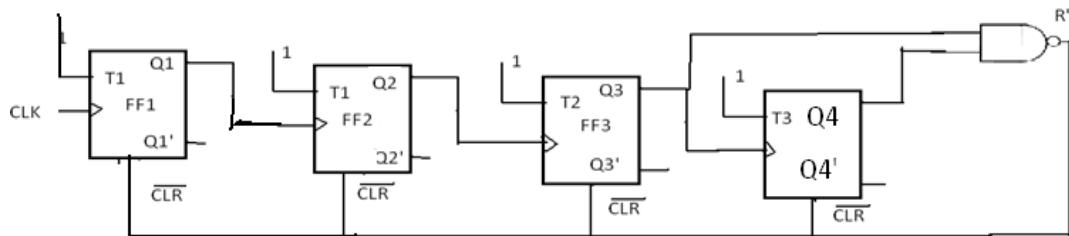
R=0 for 000 to 101, R=1 for 110, and R=X=for111

Therefore, R=Q3Q2Q1'+Q3Q2Q1=Q3Q2

The logic diagram and timing diagram of Mod-6 counter is shown in the above fig.

The truth table is as shown in below.

After pulses	States			
	Q3	Q2	Q1	R
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
	0↓	0↓	0↓	0
7	0	0	0	0

Design of a mod-10 asynchronous counter using T-flip-flops:

A mod-10 counter is a decade counter. It also called a BCD counter or a divide-by-10 counter. It requires four flip-flops (condition $10 \leq 2^n$ is n=4). So, there are 16 possible states, out of which ten are valid and remaining six are invalid. The counter has ten stable state, 0000 through 1001, i.e., it counts from 0 to 9. The initial state is 0000 and after nine clock pulses it goes to 1001. When the tenth clock pulse is applied, the counter goes to state 1010 temporarily, but because of the feedback provided, it resets to initial state 0000. So, there will be a glitch in the waveform of Q2. The state 1010 is a temporary state for which the reset signal R=1, R=0 for 0000 to 1001, and R=C for 1011 to 1111.
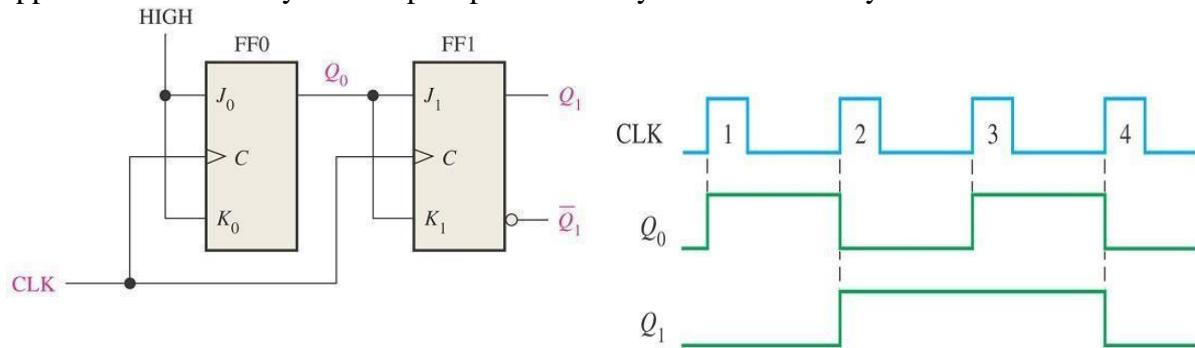
The count table and the K-Map for reset are shown in fig. from the K-Map R=Q4Q2. So, feedback is provided from second and fourth FFs. For active –HIGH reset, Q4Q2 is applied to the clear terminal. For active-LOW reset ☐4☐2 is connected ☐☐☐ isof all Flip=flops.

Q2Q1

Q4Q3	00	01	11	10
00				
01				
11	X	X	X	X
10		X	X	1

After pulses	Count			
	Q4	Q3	Q2	Q1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	0	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	0	1	0	1
10	0	0	0	0

Synchronous counters:

Asynchronous counters are serial counters. They are slow because each FF can change state only if all the preceding FFs have changed their state. if the clock frequency is very high, the asynchronous counter may skip some of the states. This problem is overcome in synchronous counters or parallel counters. Synchronous counters are counters in which all the flip flops are triggered simultaneously by the clock pulses Synchronous counters have a common clock pulse applied simultaneously to all flip-flops. A 2-Bit Synchronous Binary Counter

Design of synchronous counters:
For a systematic design of synchronous counters. The following procedure is used.

Step 1: State Diagram: draw the state diagram showing all the possible states state diagram which also be called nth transition diagrams, is a graphical means of depicting the sequence of states through which the counter progresses.

Step2: number of flip-flops: based on the description of the problem, determine the required number n of the flip-flops- the smallest value of n is such that the number of states $N \leq 2^n$--- and the desired counting sequence.

Step3: choice of flip-flops excitation table: select the type of flip-flop to be used and write the excitation table. An excitation table is a table that lists the present state (ps) , the next state(ns) and

required excitations.

Step4: minimal expressions for excitations: obtain the minimal expressions for the excitations of the FF using K-maps drawn for the excitation of the flip-flops in terms of the present states and inputs.

Step5: logic diagram: draw a logic diagram based on the minimal expressions

Design of a synchronous 3-bit up-down counter using JK flip-flops:

Step1: determine the number of flip-flops required. A 3-bit counter requires three FFs. It has 8 states (000,001,010,011,101,110,111) and all the states are valid. Hence no don't cares. For selecting up and down modes, a control or mode signal M is required. When the mode signal M=1 and counts down when M=0. The clock signal is applied to all the FFs simultaneously.

Step2: draw the state diagrams: the state diagram of the 3-bit up-down counter is drawn as
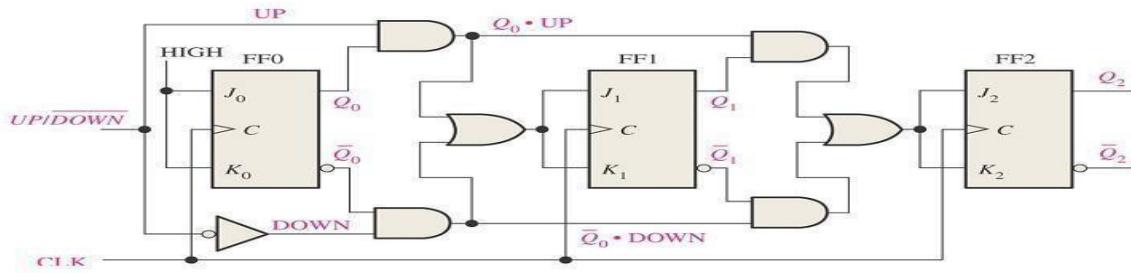
Step3: select the type of flip flop and draw the excitation table: JK flip-flops are selected and the excitation table of a 3-bit up-down counter using JK flip-flops is drawn as shown in fig.

PS			mode	NS			required excitations					
Q3	Q2	Q1	M	Q3	Q2	Q1	J3	K3	J2	K2	J1	K1
0	0	0	0	1	1	1	1	x	1	x	1	x
0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	0	0	0	x	0	x	x	1
0	0	1	1	0	1	0	0	x	1	x	x	1
0	1	0	0	0	0	1	0	x	x	1	1	x
0	1	0	1	0	1	1	0	x	x	0	1	x
0	1	1	0	0	1	0	0	x	x	0	x	1
0	1	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	1	1	x	1	1	x	1	x
1	0	0	1	1	0	1	x	0	0	x	1	x
1	0	1	0	1	0	0	x	0	0	x	x	1
1	0	1	1	1	1	0	x	0	1	x	x	1
1	1	0	0	1	0	1	x	0	x	1	1	x
1	1	0	1	1	1	1	x	0	x	0	1	x
1	1	1	0	1	1	0	x	0	x	0	x	1
1	1	1	1	0	0	0	x	1	x	1	x	1

Step4: obtain the minimal expressions: From the excitation table we can conclude that J1=1 and K1=1, because all the entries for J1 and K1 are either X or 1. The K-maps for J3, K3,J2 and K2 based on the excitation table and the minimal expression obtained from them are shown in fig.

```
        00   01   11    10
Q3Q2\Q1M
       | 1 |
             |   | 1 |
       | X | X | X | X |
       | X | X | X | X |
```
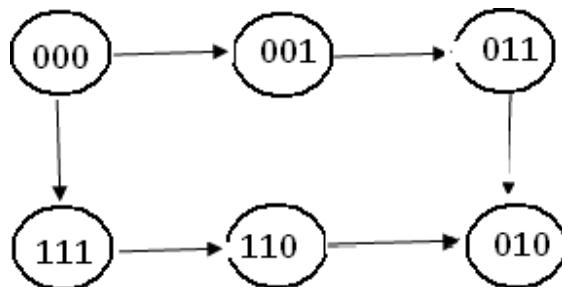
Step5: draw the logic diagram: a logic diagram using those minimal expressions can be drawn as shown in fig.

Design of a synchronous modulo-6 gray cod counter:

Step 1: the number of flip-flops: we know that the counting sequence for a modulo-6 gray code counter is 000, 001, 011, 010, 110, and 111. It requires n=3FFs ($N \leq 2^n$, i.e., $6 \leq 2^3$). 3 FFs can have 8 states. So the remaining two states 101 and 100 are invalid. The entries for excitation corresponding to invalid states are don't cares.
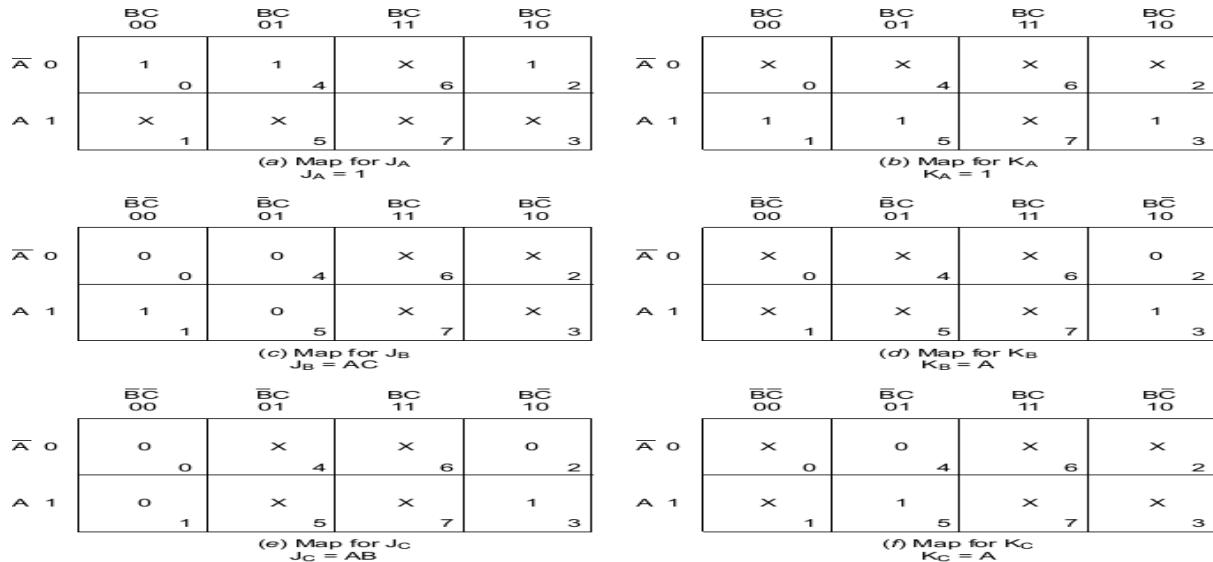
Step2: the state diagram: the state diagram of the mod-6 gray code converter is drawn as shown in fig.
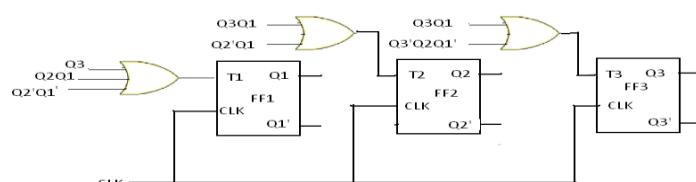
Step3: type of flip-flop and the excitation table: T flip-flops are selected and the excitation table of the mod-6 gray code counter using T-flip-flops is written as shown in fig.

PS			NS			required excitations		
Q3	Q2	Q1	Q3	Q2	Q1	T3	T2	T1
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	0	1
0	1	0	1	1	0	1	0	0
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

Step4: The minimal expressions: the K-maps for excitations of FFs T3,T2,and T1 in terms of outputs of FFs Q3,Q2, and Q1, their minimization and the minimal expressions for excitations obtained from them are shown if fig



Step5: the logic diagram: the logic diagram based on those minimal expressions is drawn as shown in fig.

Design of a synchronous BCD Up-Down counter using FFs:

Step1: the number of flip-flops: a BCD counter is a mod-10 counter has 10 states (0000 through 1001) and so it requires n=4FFs($N \leq 2^n$, i.e., $10 \leq 2^4$). 4 FFS can have 16 states. So out of 16 states, six states (1010 through 1111) are invalid. For selecting up and down mode, a control or mode signal M is required. , it counts up when M=1 and counts down when M=0. The clock signal is applied to all FFs.

**Step2:** the state diagram: The state diagram of the mod-10 up-down counter is drawn as shown in fig.

Step3: types of flip-flops and excitation table: T flip-flops are selected and the excitation table of the modulo-10 up down counter using T flip-flops is drawn as shown in fig.

The remaining minterms are don't cares($\sum d(20,21,22,23,24,25,26,37,28,29,30,31)$) from the excitation table we can see that T1=1 and the expression for T4,T3,T2 are as follows.

$T4=\sum m(0,15,16,19)+d(20,21,22,23,24,25,26,27,28,29,30,31)$
$T3=\sum m(7,15,16,8)+d(20,21,22,23,24,25,26,27,28,29,30,31)$
$T2=\sum m(3,4,7,8,11,12,15,16)+d(20,21,22,23,24,25,26,27,28,29,30,31)$
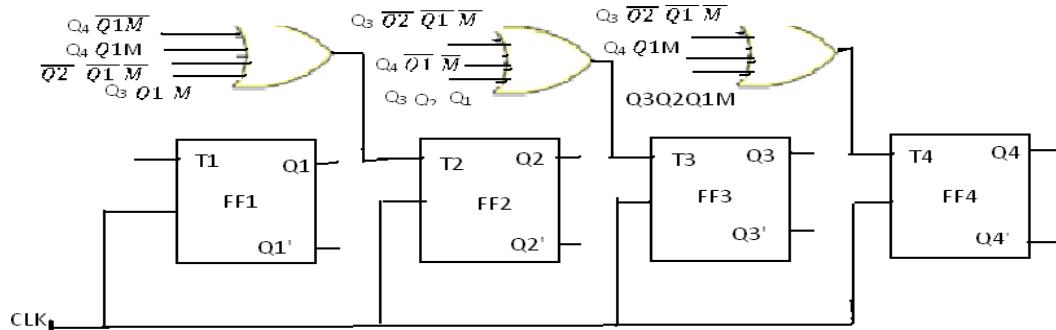
PS				mode	NS				required excitations			
Q4	Q3	Q2	Q1	M	Q4	Q3	Q2	Q1	T4	T3	T2	T1
0	0	0	0	0	1	0	0	1	1	0	0	1
0	0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	0	0	1	0	0	1	1
0	0	1	0	1	0	0	1	1	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	1
0	0	1	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	0	0	1	1	0	1	1	1
0	1	0	0	1	0	1	0	1	0	0	0	1
0	1	0	1	0	0	1	0	0	0	0	0	1
0	1	0	1	1	0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	0	1	0	0	1	1
0	1	1	0	1	0	1	1	1	0	0	0	1
0	1	1	1	0	0	1	1	0	0	0	0	1
0	1	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	1	0	0	1

Step4: The minimal expression: since there are 4 state variables and a mode signal, we require 5 variable kmaps. 20 conditions of Q4Q3Q2Q1M are valid and the remaining 12 combinations are invalid. So the entries for excitations corresponding to those invalid combinations are don't cares. Minimizing K-maps for T2 we get

$$T2= Q4Q1'M+Q4'Q1M+Q2Q1'M'+Q3Q1'M'$$

Step5: the logic diagram: the logic diagram based on the above equation is shown in fig.

Shift register counters:

One of the applications of shift register is that they can be arranged to form several types of counters. The most widely used shift register counter is ring counter as well as the twisted ring counter.

Ring counter: this is the simplest shift register counter. The basic ring counter using D flip- flops is shown in fig. the realization of this counter using JK FFs. The Q output of each stage is connected to the D flip-flop connected back to the ring counter.
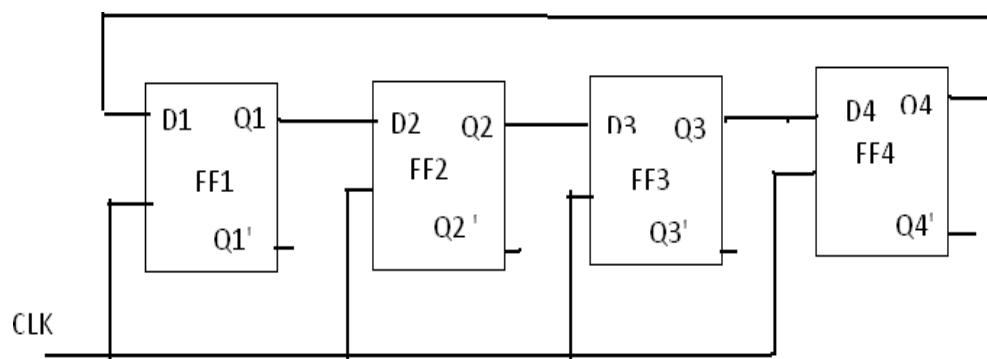
fig: logic diagram of 4-bit ring counter using D flip-flops

Only a single 1 is in the register and is made to circulate around the register as long as clock pulses are applied. Initially the first FF is present to a 1. So, the initial state is 1000, i.e., Q1=1, Q2=0,Q3=0,Q4=0. After each clock pulse, the contents of the register are shifted to the right by one bit and Q4 is shifted back to Q1. The sequence repeats after four clock pulses. The number

of distinct states in the ring counter, i.e., the mod of the ring counter is equal to number of FFs used in the counter. An n-bit ring counter can count only n bits, where as n-bit ripple counter can count 2^n bits. So, the ring counter is uneconomical compared to a ripple counter but has advantage of requiring no decoder, since we can read the count by simply noting which FF is set. Since it is entirely a synchronous operation and requires no gates external FFs, it has the further advantage of being very fast.

Timing diagram:
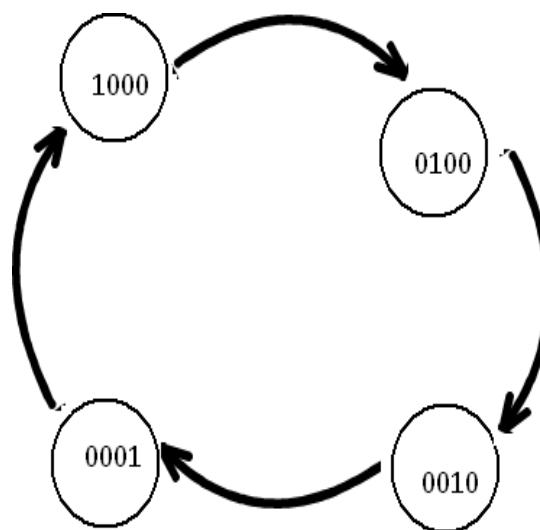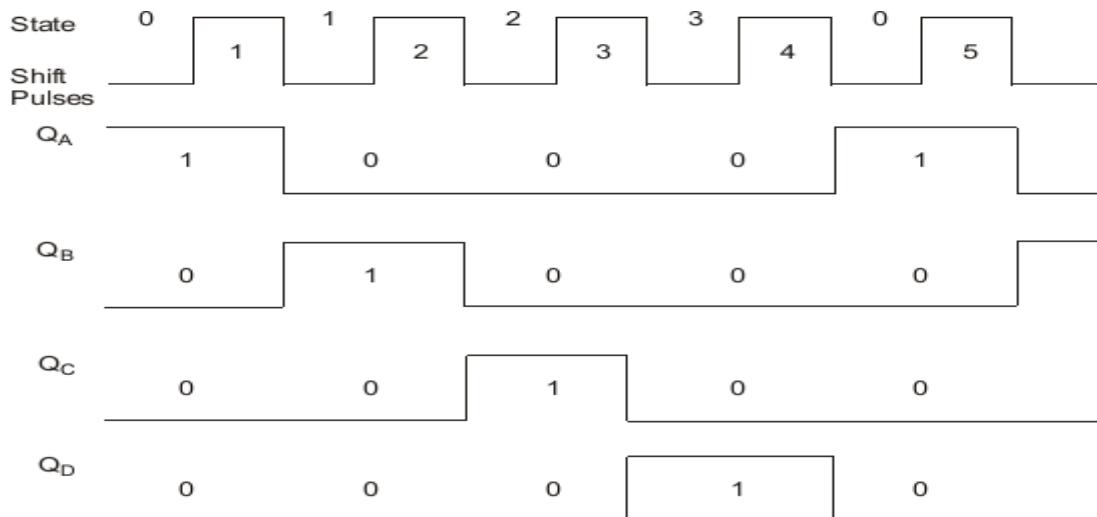
Figure: state diagram

Twisted Ring counter (Johnson counter):

This counter is obtained from a serial-in, serial-out shift register by providing feedback from the inverted output of the last FF to the D input of the first FF. the Q output of each is connected to the D input of the next stage, but the Q' output of the last stage is connected to the D input of the first stage, therefore, the name twisted ring counter. This feedback arrangement produces a unique sequence of states.

The logic diagram of a 4-bit Johnson counter using D FF is shown in fig. the realization of the same using J-K FFs is shown in fig.. The state diagram and the sequence table are shown in figure. The timing diagram of a Johnson counter is shown infigure.

Let initially all the FFs be reset, i.e., the state of the counter be 0000. After each clock pulse, the level of Q1 is shifted to Q2, the level of Q2to Q3, Q3 to Q4 and the level of Q4'to Q1 and the sequences given in fig.
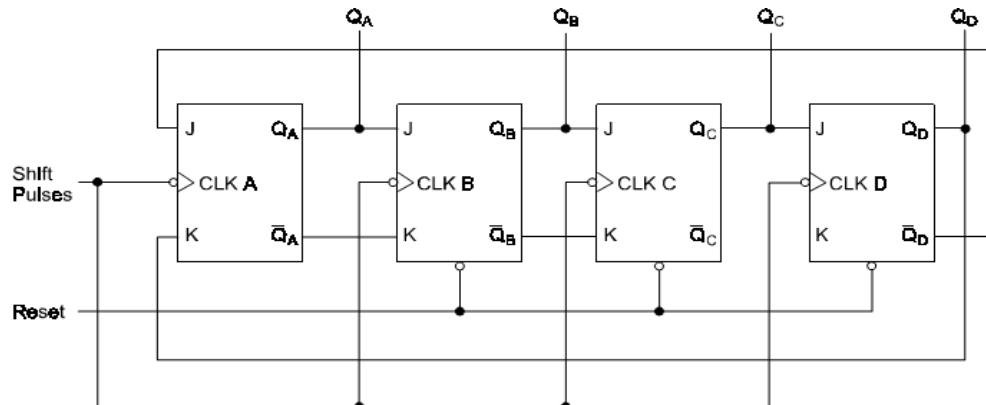

Figure: Johnson counter with JK flip-flops
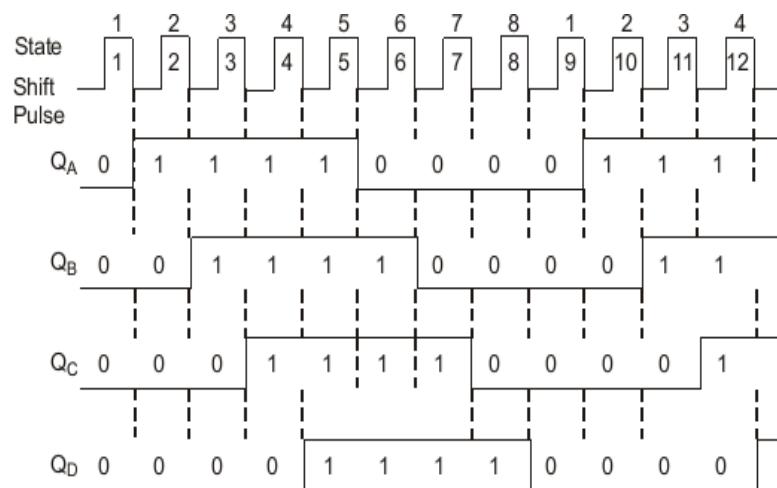
Figure: timing diagram

168

Statediagram:

	Q1	Q2	Q3	Q4	after clock pulse
	0	0	0	0	0
	1	0	0	0	1
	1	1	0	0	2
	1	1	1	0	3
	1	1	1	1	4
	0	1	1	1	5
	0	0	1	1	6
	0	0	0	1	7
	0	0	0	0	8
	1	0	0	0	9

Excitation table

Synthesis of sequential circuits:

The synchronous or clocked sequential circuits are represented by two models.
1. Moore circuit: in this model, the output depends only on the present state of the flip-flops
2. Meelay circuit: in this model, the output depends on both present state of the flip-flop. And the inputs.

Sequential circuits are also called finite state machines (FSMs). This name is due to the fast that the functional behavior of these circuits can be represented using a finite number of states.

State diagram: the state diagram or state graph is a pictorial representation of the relationships between the present state, the input, the next state, and the output of a sequential circuit. The state diagram is a pictorial representation of the behavior of a sequential circuit.

The state represented by a circle also called the node or vertex and the transition between states is indicated by directed lines connecting circle. a directed line connecting a circle with itself indicates that the next state is the same as the present state. The binary number inside each circle identifies the state represented by the circle. The direct lines are labeled with two binary numbers separated by a symbol. The input value is applied during the present state is labeled after the symbol.

	NS,O/P	
	INPUT X	
PS	X=0	X=1
a	a,0	b,0
b	b,1	c,0
c	d,0	c,1
d	d,0	a,1
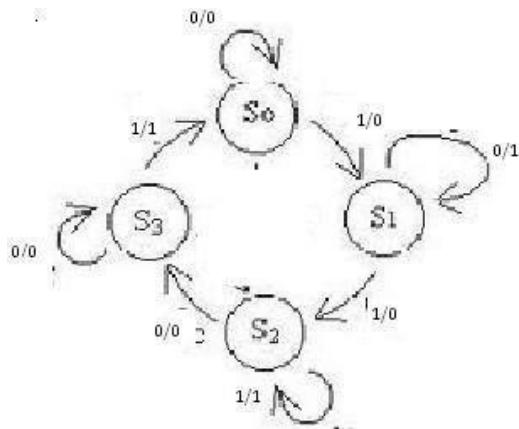
fig :a) state diagram (meelay circuit)   fig: b) state table

In case of moore circuit ,the directed lines are labeled with only one binary number representing the input that causes the state transition. The output is indicated with in the circle below the present state, because the output depends only on the present state and not on the input.


	NS		
	INPUT X		
PS	X=0	X=1	O/P
a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	0
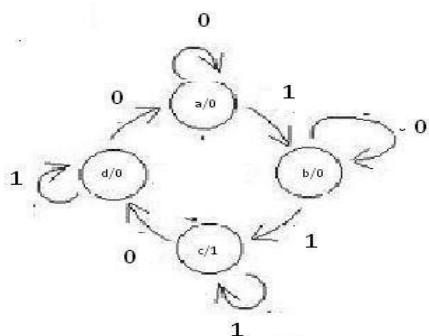
fig: a)   state diagram (moore circuit)   fig:b) state table

Serial binary adder:

Step1: word statement of the problem: the block diagram of a serial binary adder is shown in fig. it is a synchronous circuit with two input terminals designated X1and X2 which carry the two binary numbers to be added and one output terminal Z which represents the sum. The inputs and outputs consist of fixed-length sequences 0s and 1s.the output of the serial Z_i at time $t_i$is a function of the inputs $X_1(t_i)$ and $X_2(t_i)$ at that time t_{i-1} and of carry which had been generated att_i-

1. The carry which represent the past history of the serial adder may be a 0 or 1. The circuit has two states. If one state indicates that carry from the previous addition is a 0, the other state indicates that the carry from the previous addition is a 1
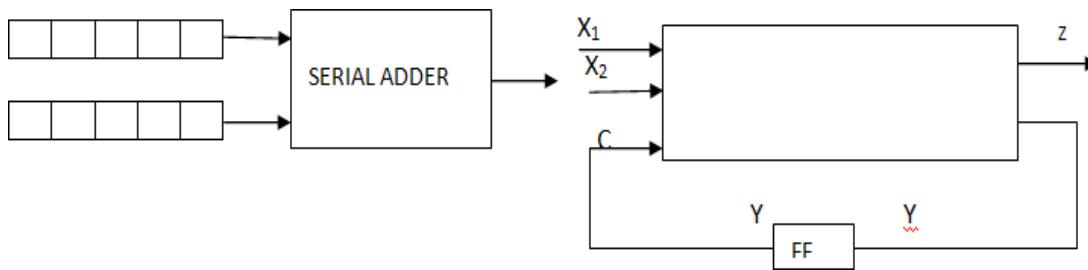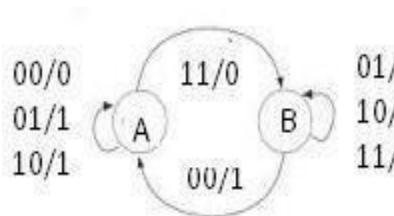
Figure: block diagram of serial binary adder

Step2 and 3: state diagram and state table: let a designate the state of the serial adder at t_i if a carry 0 was generated at t_{i-1}, and let b designate the state of the serial adder at t_i if carry 1 was generated at t_{i-1} .the state of the adder at that time when the present inputs are applied is referred to as the present state(PS) and the state to which the adder goes as a result of the new carry value is referred to as next state(NS).

The behavior of serial adder may be described by the state diagram and state table.

PS	NS ,O/P			
	X1 X2			
	0 0	0 1	1 0	1 1
A	A,0	B,0	B,1	B,0
B	A,1	B,0	B,0	B,1

Figures: serial adder state diagram and state table

If the machine is in state B, i.e., carry from the previous addition is a 1, inputs $X_1=0$ and $X_2=1$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=0$ gives sum, 0 and carry 1. So the machine remains in state B and outputs a 0. Inputs $X_1=1$ and $X_2=1$ gives sum, 1 and carry 0. So the machine remains in state B and outputs a 1. Inputs $X_1=0$ and $X_2=0$ gives sum, 1 and carry 0. So the machine goes to state A and outputs a 1. The state table also gives the same information.

Setp4: reduced standard from state table: the machine is already in this form. So no need to do anything

Step5: state assignment and transition and output table:

The states, A=0 and B=1 have already been assigned. So, the transition and output table is as shown.

PS	NS				O/P			
	0	**0**	**1**	**1**	**0**	**0**	**1**	**1**
	0	**1**	**0**	**1**	**0**	**1**	**0**	**1**
0	0	0	0	1	0	1	1	1
1	0	1	1	1	1	0	0	1

STEP6: choose type of FF and excitation table: to write table, select the memory elementthe excitation table is as shown in fig.
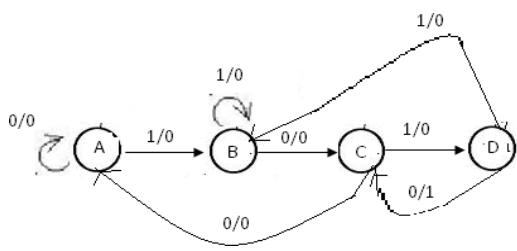
PS	I/P		NS	I/P-FF	O/P
y	x1	x2	Y	D	Z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Sequence detector:

Step1: word statement of the problem: a sequence detector is a sequential machine which produces an output 1 every time the desired sequence is detected and an output 0 at all other times

Suppose we want to design a sequence detector to detect the sequence 1010 and say that overlapping is permitted i.e., for example, if the input sequence is 01101010 the corresponding output sequence is 00000101.

Step2 and 3: state diagram and state table: the state diagram and the state table of the sequence detector. At the time t1, the machine is assumed to be in the initial state designed arbitrarily as A. while in this state, the machine can receive first bit input, either a 0 o r a 1. If the input bit is 0, the machine does not start the detection process because the first bit in the desired sequence is a 1. If the input bit is a 1 the detection process starts.

PS	NS,Z	
	X=0	X=1
A	A,0	B,0
B	C,0	B,0
C	A,0	D,0
D	C,1	B,0

Figure: state diagram and state table of sequence detector

So, the machine goes to state B and outputs a 0. While in state B, the machinery may receive 0 or 1 bit. If the bit is 0, the machine goes to the next state, say state c, because the previous two bits are 10 which are a part of the valid sequence, and outputs 0.. if the bit is a 1, the two bits become 11 and this not a part of the valid sequence

Step4: reduced standard form state table: the machine is already in this form. So no need to do anything.

Step5: state assignment and transition and output table: there are four states therefore two states variables are required. Two state variables can have a maximum of four states, so, all states are utilized and thus there are no invalid states. Hence, there are no don't cares. Let a=00, B=01, C=10 and D=11 be the state assignment.

PS(y_1y_2)	NS(Y_1Y_2) X=0		X=1		O/P(z) X=0	X=1
A= 0 0	0	0	0	1	0	0
B=0 1	1	0	0	1	0	0
C=1 0	0	0	1	1	0	0
D=1 1	1	1	0	1	1	0

Step6: choose type of flip-flops and form the excitation table: select the D flip-flops as memory elements and draw the excitation table.

PS y_1	Y_2	I/P X	NS Y_1	Y_2	INPUTS - FFS D_1	D_2	O/P Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0

Step7: K-maps and minimal functions: based on the contents of the excitation table , draw the k-map and simplify them to obtain the minimal expressions for D1 and D2 in terms of y1, y2 and x as shown in fig. The expression for z (z=y1,y2) can be obtained directly from table

Step8: implementation: The logic diagram based on these minimal