ANALOG & DIGITAL ELECTRONICS

LECTURENOTES

B.TECH

(IIYEAR- II SEM)

(2024-25)

Prepared by:

D. SANTHOSH KUMAR, Asst. Prof

Department of Electronics & Communication Engineering

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

(AffiliatedtoJNTUH, Hyderabad, Approved by AICTE - Accredited by NBA& NAAC – 'A' Grade - ISO 9001:2015Certified)

Maisammaguda,Dhulapally (PostVia.Kompally), Secunderabad –500100,Telangana State,India

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

II Year B.Tech EEE-II Sem **L/T/P/C**
3/0/0/3

(R22A0461) ANALOG & DIGITAL ELECTRONICS

OBJECTIVES The main objectives of the course are:

1. Learn the concepts of load line analysis and biasing techniques
2. Learn the concepts of small signal analysis of BJT and FET
3. To understand basic number systems codes and logical gates.
4. To introduce the methods for simplifying Boolean expressions
5. To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits

UNIT-I

BJT Biasing:

Transistor Biasing and Stabilization - Operating point, DC & AC load lines, Biasing - Fixed Bias, Self-Bias, Bias Stability, Bias Compensation using Diode and Transistor amplifying action.

Signal Low Frequency BJT Amplifiers: Transistor Hybrid model, Determination of h-parameters from transistor characteristics, Typical values of h- parameters in CE, CB and CC configurations

UNIT-II

Transistor at High Frequency:

Hybrid π model of Common Emitter transistor model and derivation of Hybrid π model elements.

FET Amplifiers: Analysis of Common Source and Common Drain JFET Amplifiers, Comparison of performance with BJT Amplifiers

UNIT-III

Number System and Boolean Algebra:

Number Systems, Base Conversion Methods, Complements of Numbers, Codes- Binary Codes, Binary Coded Decimal, Unit Distance Code, Digital Logic Gates (AND, NAND, OR, NOR, EX-OR, EX-NOR), Properties of XOR Gates, Universal Gates, Basic Theorems and Properties, Switching Functions, Canonical and Standard Form.

UNIT-IV

Minimization Techniques:

The Karnaugh Map Method, Three variables ,Prime and Essential Implications, Don't Care Map Entries, Using the Maps for Simplifying, Multilevel NAND/NOR realizations.

UNIT-V

Combinational Circuits:

Design procedure – Half adder, Full Adder, Half sub-tractor, Full sub-tractor, Multiplexer/De-multiplexer Sequential circuits: Latches, Flip-Flops-SR, JK, D, T and master slave, characteristic tables and equation.

TEXT BOOKS:

1. "Electronic Devices & Circuits", Special Edition – MRCET, McGraw Hill Publications, 2017.
2. Integrated Electronics Analog Digital Circuits, Jacob Millman and D. Halkias, McGraw Hill.
3. Electronic Devices and Circuits, S.Salivahanan,N.Suresh kumar, McGraw Hill.
4. M. Morris Mano, Digital Design, 3rd Edition, Prentice Hall of India Pvt. Ltd., 2003 /Pearson Education (Singapore) Pvt. Ltd., New Delhi, 2003.
5. Switching and Finite Automata Theory- Zvi Kohavi & Niraj K. Jha, 3rd Edition, Cambridge.

REFERENCE BOOKS:

1. Electronic Devices and Circuits,K.Lal Kishore B.S Publications
2. Electronic Devices and Circuits, G.S.N. Raju, I.K. International Publications, New Delhi, 2006.
3. John F.Wakerly, Digital Design, Fourth Edition, Pearson/PHI, 2006
4. John.M Yarbrough, Digital Logic Applications and Design, Thomson Learning, 2002.
5. Charles H.Roth. Fundamentals of Logic Design, Thomson Learning, 2003.

OUTCOMES:

After completion of the course, the student will be able to:

1. Design the amplifiers with various biasing techniques
2. Design single stage amplifiers using BJT and FET
3. Understand the basic postulates of Boolean algebra and shows the correlation between Boolean expressions
4. Learn the methods for simplifying Boolean expressions
5. Understand the formal procedures for the analysis and design of combinational circuits and sequential circuits

UNIT I:
Biasing and Stabilization:

Biasing and Stabilization: Operating point, the D.C Load line, Fixed bias, Collector to base bias, Self-bias techniques for stabilization, Stabilization factors, (s, s^{I}), Bias Compensation using diode and transistor

TRANSISTOR BIASING AND STABILIZATION

NEED FOR TRANSISTORBIASING

If the o/p signal must be a faithful reproduction of the i/p signal, the transistor must be operated in active region. That means an operating point has to be established in this region . To establish an operating point (proper values of collector current $I_c$and collector to emitter voltage V_{CE}) appropriate supply voltages and resistances must be suitably chosen in the ckt. This process of selecting proper supply voltages and resistance for obtaining desired operating point or Q point is called as biasing and the ckt used for transistor biasing is called as biasingckt.

There are four conditions to be met by a transistor so that it acts as a faithful ampr:

1) Emitter base junction must be forward biased (V_{BE}=0.7Vfor Si, 0.2V for Ge) and collector base junction must be reverse biased for all levels of i/psignal.
2) V_{ce} voltage should not fall below $V_{CE(sat)}$ (0.3V for Si, 0.1V for Ge) for any part of the i/p signal. For V_{CE} less than $V_{CE(sat)}$ the collector base junction is not probably reverse biased.
3) The value of the signal I_c when no signal is applied should be at least equal to the max. collector current t due to signalalone.
4) Max. rating of the transistor $I_{c(max)}$, $V_{CE(max)}$ and $P_{D(max)}$ should not be exceeded at any value of i/p signal.

Consider the fig shown in fig 2.12. If operating point is selected at A, A represents a condition when no bias is applied to the transistor i.e, I_c=0, V_{CE} =0. It does not satisfy the above said conditions necessary for faithful amplification.

Point C is too close to $P_{D(max)}$ curve of the transistor. Therefore the o/p voltage swing in the positive direction is limited.

Point B is located in the middle of active region .It will allow both positive and negative half cycles in the o/p signal. It also provides linear gain and larger possible o/p voltages andcurrents

Hence operating point for a transistor amplifier is selected to be in the middle of active region.

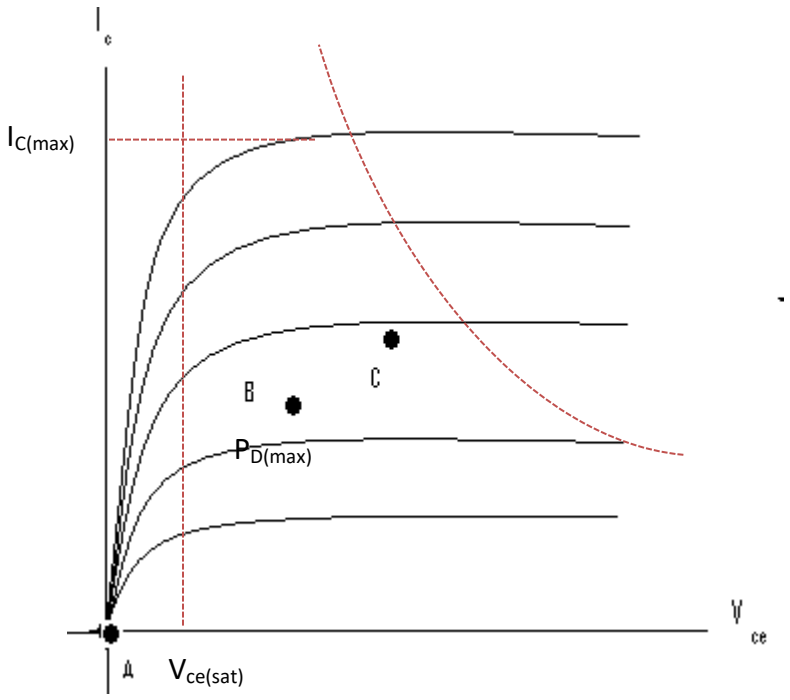


Fig 2.12 CE Output Characteristics

DC LOADLINE

Referring to the biasing circuit of fig 2.13 a, the values of V_{CC} and R_C are fixed and Ic and V_{CE} are dependent on R_B.

Applying Kirchhoff's voltage law to the collector circuit in fig. 2.13, we get

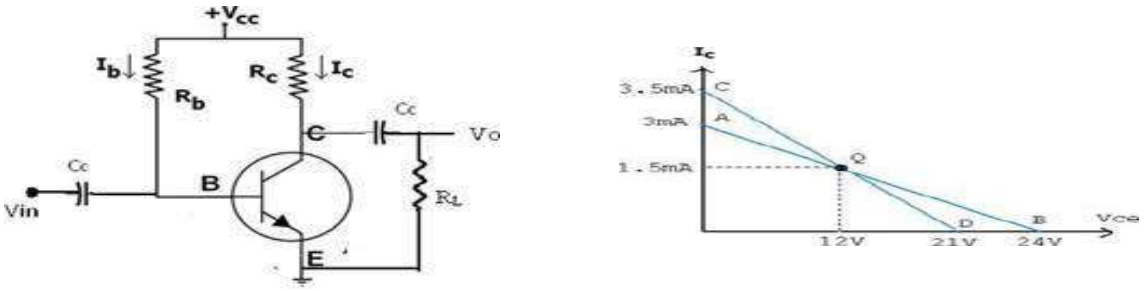


Fig 2.13(a) CE Amplifier Circuit (b) Load line

$$Vcc = IcRc + Vce$$

The straight line represented by AB in fig2.13b is called the dc load line. The coordinates of the end point A are obtained by substituting V_{CE} =0 in the above equation. Then $Ic = \frac{Vcc}{Rc}$. Therefore The coordinates of A are V_{CE} =0 and $Ic = \frac{Vcc}{Rc}$.

The coordinates of B are obtained by substituting Ic=0 in the above equation. Then Vce = Vcc. Therefore the coordinates of B are V_{CE} =Vcc and Ic=0. Thus the dc load line AB can be drawn if the values of Rc and Vcc are known.

As shown in the fig2.13b, the optimum POINT IS LOCATED AT THE MID POINT OF THE MIDWAY BETWEEN a AND b. In order to get faithful amplification, the Q point must be well within the active region of the transistor.

Even though the Q point is fixed properly, it is very important to ensure that the operating point remains stable where it is originally fixed. If the Q point shifts nearer to either A or B, the output voltage and current get clipped, thereby o/p signal is distorted.

In practice, the Q-point tends to shift its position due to any or all of the following three main factors.

1) Reverse saturation current, Ico, which doubles for every 10°C raise intemperature
2) Base emitter Voltage ,VBE, which decreases by 2.5 mV per°C
3) Transistor current gain, h_{FE} or β which increases withtemperature.

If base current I_B is kept constant since I_B is approximately equal to Vcc/RB. If the transistor is replaced by another one of the same type, one cannot ensure that the new transistor will have identical parameters as that of the first one. Parameters such as β vary over a range. This results in the variation of collector current Ic for a given I_B. Hence , in the o/p characteristics, the spacing between the curves might increase or decrease which leads to the shifting of the Q-point to a location which might be completelyunsatisfactory.

AC LOADLINE

After drawing the dc load line, the operating point Q is properly located at the center of the dc load line. This operating point is chosen under zero input signal condition of the circuit. Hence the ac load line should also pas through the operating point Q. The effective ac load resistance R_{ac}, is a combination of R_C parallel toR_L i.e. $R_{ac} = R_L||R_C$. So the slope of the ac load line CQD will be $\left(\frac{-1}{R_{ac}}\right)$. To draw the ac load line, two end points, I.e. $V_{CE(max)}$ and $I_{C(max)}$ when the signal is applied arerequired.

$V_{CE(\max)} = V_{CEQ} + I_{CQ}R_{ac}$, which locates point D on the Vce axis.

$I_{c(\max))} = I_{CQ} + \frac{V_{CEQ}}{R_{ac}}$, which locates the point C on the I_C axis.

By joining points c and D, ac load line CD is constructed. As $R_C > R_{ac}$, The dc load line is less steep than ac load line.

STABILITY FACTOR(S):

The rise of temperature results in increase in the value of transistor gain β and the leakage current Ico. So, I_C also increases which results in a shift in operating point. Therefore, The biasing network should be provided with thermal stability. Maintenance of the operating point is specified by S, which indicates the degree of change in operating point due to change intemperature.

The extent to which I_C is stabilized with varying I_C is measured by a stability factor S

$$S = \frac{\partial I_c}{\partial I_{co}} \approx \frac{dI_c}{dI_{co}} \approx \frac{\Delta I_c}{\Delta I_{co}}, \ \beta \text{ and } I_B \text{ constant}$$

For CEconfiguration $I_C = \beta I_B + (1+\beta) I_{CO}$

Differentiate the above equation w.r.t I_C , We get

$$1 = \beta \frac{dI_B}{dI_C} + (1+\beta) \frac{dI_{co}}{dI_c}$$

$$\therefore \left(1 - \beta \frac{dI_B}{dI_C}\right) = \frac{(\beta+1)}{S}$$

$$\therefore S = \frac{1+\beta}{1-\beta \frac{dI_B}{dI_C}}$$

S should be small to have better thermal stability.

Stability factor S' and S'':

S' is defined as the rate of change of I_C with V_{BE}, keeping I_C and V_{BE} constant.

$$S' = \frac{\partial I_c}{\partial V_{BE}}$$

S'' is defined as the rate of change of I_C with β, keeping I_{CO} and V_{BE}constant.

$$S'' = \frac{\partial I_c}{\partial \beta}$$

METHODS OF TRANSISTORBIASING

1) Fixed bias (basebias)

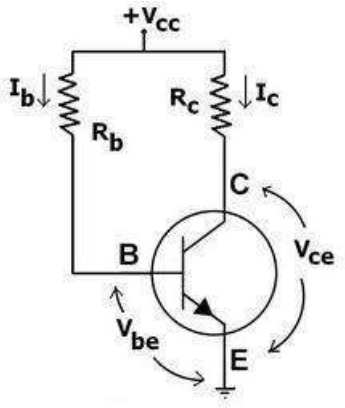


Fig 2.14 Fixed Biasing Circuit

This form of biasing is also called *base bias*. In the fig 4.3 shown, the single power source (for example, battery) is used for both collector and base of a transistor, although separate batteries can also be used.

In the given circuit,

$V_{cc} = I_B R_B + V_{be}$

Therefore, $I_B = (V_{cc} - V_{be})/R_B$

Since the equation is independent of current $I_C R$, $dI_B/dI_C R = 0$ and the stability factor is given by the equation….. reduces to

$S = 1+\beta$

Since β is a large quantity, this is very poor biasing circuit. Therefore in practice the circuit is not used fo biasing.

For a given transistor, V_{be} does not vary significantly during use. As V_{cc} is of fixed value, on selection of R the base current I_B is fixed. Therefore this type is called *fixed bias* type ofcircuit.

Also for given circuit, $V_{cc} = I_C R_C + V_{ce}$

Therefore, $V_{ce} = V_{cc} - I_C R_C$

Merits:

- It is simple to shift the operating point anywhere in the active region bymerely changing the base resistor(R_B).
- A very small number of components arerequired.

Demerits:

- The collector current does not remain constant with variation in temperature orpower supply voltage. Therefore the operating point isunstable.
- Changes in V_{be} will change I_B and thus cause R_E to change. This in turn will alter the gain of thestage.
- When the transistor is replaced with another one, considerable change in the value ofβ can be expected. Due to this change the operating point willshift.

2) EMITTER-FEEDBACKBIAS:

The emitter feedback bias circuit is shown in the fig 2.15. The fixed bias circuit is modified by attaching an external resistor to the emitter. This resistor introduces negative feedback that stabilizes the Q-point. From Kirchhoff's voltage law, the voltage across the base resistor is

$V_{Rb} = V_{CC} - I_e R_e - V_{be}$.

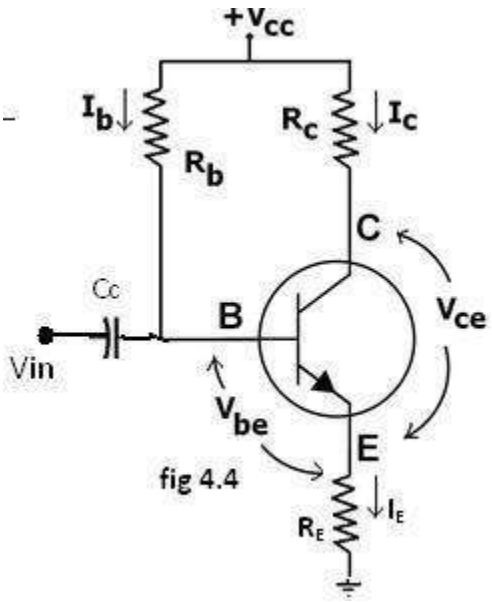


fig 4.4

Fig 2.15 Self Biasing Circuit

From Ohm's law, the base current is

$I_b = V_{Rb} / R_b$.

The way feedback controls the bias point is as follows. If V_{be} is held constant and temperature increases, emitter current increases. However, a larger I_e increases the emitter voltage $V_e = I_e R_e$, which in turn reduces the voltage V_{Rb} across the base resistor. A lower base-resistor voltage drop reduces the base current, which results in less collector current because I_c = ß I_B. Collector current and emitter current are related by $I_c = \alpha I_e$ with $\alpha \approx 1$, so increase in emitter current with temperature is opposed, and operating point is kept stable.

Similarly, if the transistor is replaced by another, there may be a change in I_C (corresponding to change in β-value, for example). By similar process as above, the change is negated and operating point kept stable.

For the given circuit,

$I_B = (V_{CC} - V_{be})/(R_B + (\beta+1)R_E)$.

Merits:

The circuit has the tendency to stabilize operating point against changes in temperature and β-value.

Demerits:

- In this circuit, to keep I_C independent of β the following condition must bemet:

$$I_C = \beta I_B = \frac{\beta(V_{CC} - V_{be})}{R_B + (\beta + 1)R_E} \approx \frac{(V_{CC} - V_{be})}{R_E}$$

which is approximately the case if $(\beta + 1)R_E >> R_B$.

- Asβ-valueisfixedforagiventransistor,thisrelationcanbesatisfiedeitherbykeeping R_E very large, or making R_B verylow.

- If R_E is of large value, high V_{CC} is necessary. This increases cost as well as precautions necessary whilehandling.
- If R_B is low, a separate low voltage supply should be used in the base circuit. Using two supplies of different voltages isimpractical.

- In addition to the above, R_E causes ac feedback which reduces the voltage gain of theamplifier.

3)COLLECTOR TO BASE BIAS OR COLLECTOR FEED-BACKBIAS:

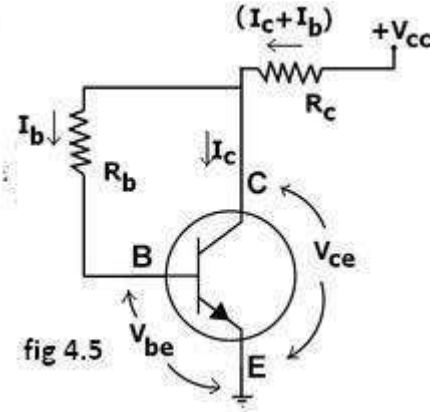


Fig 2.16 Collector to Base Biasing Circuit

This configuration shown in fig 2.16 employs negative feedback to prevent thermal runaway and stabilize the operating point. In this form of biasing, the base resistor $R_B$is connected to the collector instead of connecting it to the DC source V_{cc}. So any thermal runaway will induce a voltage drop across the $R_C$resistor that will throttle the transistor's base current.

From Kirchhoff's voltage law, the voltage V_{R_b}across the base resistor $R_b$is

$$V_{R_b} = V_{cc} - \overbrace{(I_c + I_b)R_c}^{\text{Voltage drop across } R_c} - \overbrace{V_{be}}^{\text{Voltage at base}}.$$

By the Ebers–Moll model, $I_c = \beta I_b$, and so

$$V_{R_b} = V_{cc} - (\overbrace{\beta I_b}^{I_c} + I_b)R_c - V_{be} = V_{cc} - I_b(\beta + 1)R_c - V_{be}.$$

From Ohm's law, the base current $I_b = V_{R_b}/R_b$, andso

$$\overbrace{I_b R_b}^{V_{R_b}} = V_{cc} - I_b(\beta + 1)R_c - V_{be}.$$

Hence, the base current $I_b$is

$$I_b = \frac{V_{cc} - V_{be}}{R_b + (\beta + 1)R_c}$$

If $V_{be}$is held constant and temperature increases, then the collector current $I_c$increases. However, a larger $I_c$causes the voltage drop across resistor $R_c$to increase, which in turn reduces the

voltage V_{R_b}across the base resistor R_b. A lower base-resistor voltage drop reduces the base current I_b, which results in less collector current I_c. Because an increase in collector current with temperature is opposed, the operating point is keptstable.

Merits:

- Circuit stabilizes the operating point against variations in temperature and β(i.e. replacement oftransistor)

Demerits:

- In this circuit, to keep $I_c$independent of β, the following condition must bemet:

$$I_c = \beta I_b = \frac{\beta(V_{cc} - V_{be})}{R_b + R_c + \beta R_c} \approx \frac{(V_{cc} - V_{be})}{R_c}$$

which is the case when

$$\beta R_c \gg R_b.$$

- As β-value is fixed (and generally unknown) for a given transistor, this relation can be satisfied either by keeping $R_c$fairly large or making $R_b$verylow.

- If $R_c$is large, a high $V_{cc}$is necessary, which increases cost as well as precautions necessary whilehandling.
- If $R_b$is low, the reverse bias of the collector–base region is small, which limits the range of collector voltage swing that leaves the transistor in activemode.

- The resistor $R_b$causes an AC feedback, reducing thevoltage gainof the amplifier. This undesirable effect is a trade-off for greaterQ-pointstability.

Usage: The feedback also decreases the input impedance of the amplifier as seen from the base, which can be advantageous. Due to the gain reduction from feedback, this biasing form is used only when the trade-off for stability iswarranted.

4) COLLECTOR –EMITTER FEEDBACKBIAS:

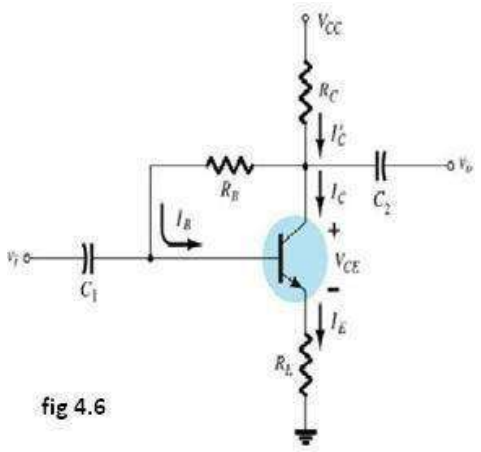


Fig 2.17 Collector-Emitter Biasing Circuit

The above fig 2.17shows the collector –emitter feedback bias circuit that can be obtained by applying both the collector feedback and emitter feedback. Here the collector feedback is provided by connecting a resistance RB from the collector to the base and emitter feedback is provided by connecting an emitter Re from emitter to ground. Both feed backs are used to control collector current and base current IB in the opposite direction to increase the stability as compared to the previous biasingcircuits.

5) VOLTAGE DIVIDER BIAS OR SELF BIAS OR EMITTERBIAS

The voltage divider as shown in the fig 2.18 is formed using external resistors R_1 and R_2. The voltage across R_2 forward biases the emitter junction. By proper selection of resistors R_1 and R_2, the operating point of the transistor can be made independent of β. In this circuit, the voltage divider holds the base voltage fixed independent of base current provided the divider current is large compared to the base current. However, even with a fixed base voltage, collector current varies with temperature (for example) so an emitter resistor is added to stabilize the Q-point, similar to the above circuits with emitter resistor.

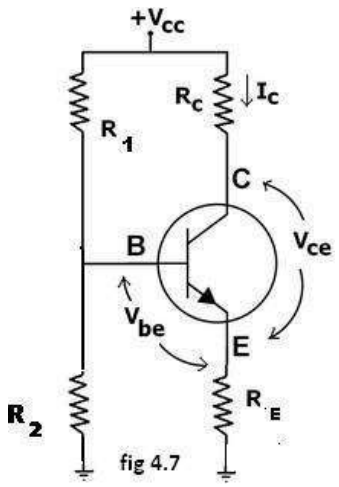


Fig 2.18 Voltage Divider Biasing Circuit

In this circuit the base voltage is given by:

$$V_B = \text{voltageacross } R_2 = V_{cc}\frac{R_2}{(R_1+R_2)} - I_B\frac{R_1R_2}{(R_1+R_2)}$$

$$\approx V_{cc}\frac{R_2}{(R_1+R_2)} \text{provided } I_B << I_2 = V_B/R_2.$$

Also $V_B = V_{be} + I_E R_E$

For the given circuit,

$$I_B = \frac{\frac{V_{CC}}{1+R_1/R_2} - V_{be}}{(\beta+1)R_E + R_1 \parallel R_2}.$$

Let the current in resistor R1 is I1 and this is divided into two parts – current through base and resistor R2. Since the base current is very small so for all practical purpose it is assumed that I1 also flows through R2, so we have

$$I_1 = \frac{V_{CC}}{R_1+R_2}$$

$$V_2 = \frac{V_{CC}}{R_1+R_2}\cdot R_2$$

Applying KVL in the circuit, we have

$$V_2 = V_{BE} + V_E$$

$$V_2 = V_{BE} + I_E R_E$$

$$I_E = \frac{V_2 - V_{BE}}{R_E}$$

$$I_C = \frac{V_2 - V_{BE}}{R_E} \quad \because I_C \cong I_E$$

$$I_C = \frac{\frac{V_{CC}}{R_1 + R_2} \cdot R_2 - V_{BE}}{R_E}$$

It is apparent from above expression that the collector current is independent of ? thus the stability is excellent. In all practical cases the value of VBE is quite small in comparison to the V2, so it can be ignored in the above expression so the collector current is almost independent of the transistor parameters thus this arrangement provides excellent stability.

Again applying KVL in collector circuit, we have

$$V_{CC} = I_C R_C + V_{CE} + I_E R_E$$

$$\because I_C \cong I_E$$

$$\therefore V_{CC} = I_C R_C + V_{CE} + I_C R_E$$

$$V_{CE} = V_{CC} - I_C (R_C + R_E)$$

The resistor RE provides stability to the circuit. If the current through the collector rises, the voltage across the resistor RE also rises. This will cause VCE to increase as the voltage V2 is independent of collector current. This decreases the base current, thus collector current increases to its formervalue.

Stability factor for such circuit arrangement is given by

$$S = \frac{(1+\beta)(R_{eq} + R_E)}{R_{eq} + R_E(1+\beta)}$$

$$R_{eq} = R_1 || R_2$$

$$S = \frac{(1+\beta)\left(1 + \frac{R_{eq}}{R_E}\right)}{\frac{R_{eq}}{R_E} + 1 + \beta}$$

If Req/RE is very small compared to 1, it can be ignored in the above expression thus we have

$$S = \frac{1+\beta}{1+\beta} = 1$$

Which is excellent since it is the smallest possible value for the stability. In actual practice the value of stability factor is around 8-10, since Req/RE cannot be ignored as compared to 1.

Merits:

- Unlike above circuits, only one dc supply isnecessary.
- Operating point is almost independent of βvariation.
- Operating point stabilized against shift intemperature.

Demerits:

- In this circuit, to keep I_C independent of β the following condition must bemet:

$$I_C = \beta I_B = \beta \frac{\frac{V_{CC}}{1+R_1/R_2} - V_{be}}{(\beta+1)R_E + R_1 \parallel R_2} \approx \frac{\frac{V_{CC}}{1+R_1/R_2} - V_{be}}{R_E},$$

which is approximately the caseif $(\beta+1)R_E >> R_1 \parallel R_2$

where R_1 || R_2 denotes the equivalent resistance of R_1 and R_2 connected in parallel.

- As β-value is fixed for a given transistor, this relation can be satisfied either by keeping R_E fairly large, or making R_1||R_2 verylow.

- If R_E is of large value, high V_{CC} is necessary. This increases cost as well as precautions necessary whilehandling.
- If R_1 || R_2 is low, either R_1 is low, or R_2 is low, or both are low. A low R_1 raises V_B closer to V_C, reducing the available swing in collector voltage, and limiting how large $R_C$can be made without driving the transistor out of active mode. A low R_2 lowers V_{be}, reducing the allowed collector current. Lowering both resistor values draws more current from the power supply and lowers the input resistance of the amplifier as seen from thebase.

- AC as well as DC feedback is caused by R_E, which reduces the AC voltage gain of the amplifier. A method to avoid AC feedback while retaining DC feedback is discussedbelow.

Usage: The circuit's stability and merits as above make it widely used for linear circuits.

BIAS COMPENSATION USING DIODE ANDTRANSISTOR

The various biasing circuits considered use some type of negative feedback to stabilize the operation point. Also, diodes, thermistors and sensistors can be used to compensate for variations in current.

DIODE COMPENSATION:

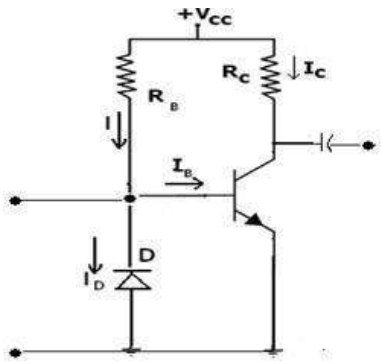


Fig 2.19Diode Compensation Circuit

The following fig 2.19 shows a transistor amplifier with a diode D connected across the base-emitter junction for compensation of change in collector saturation current I_{CO}. The diode is of the same material as the transistor and it is reverse biased by e the emitter-base junction voltage V_{BE}, allowing the diode reverse saturation current I_O to flow through diode D. The base currentI_B=I-I_O.

As long as temperature is constant, diode D operates as a resistor. As the temperature increases, I_{CO} of the transistor increases. Hence, to compensate for this, the base current I_B should be decreased.

The increase in temperature will also cause the leakage current I_O through D to increase and thereby decrease the base current I_B. This is the required action to keep Ic constant.

This type of bias compensation does not need a change in Ic to effect the change in I_C, as both I_O and I_{CO} can track almost equally according to the change intemperature.

THERMISTOR COMPENSATION:

The following fig2.20 a thermistor R_T, having a negative temperature coefficient is connected in parallel with R_2. The resistance of thermistor decreases exponentially with increase of temperature. An increase of temperature will decrease the base voltage V_{BE}, reducing I_B and I_C.

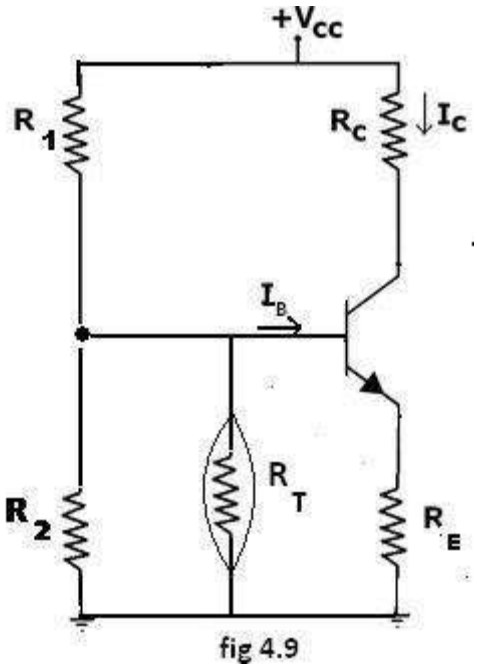


Fig 2.20 Thermistor Compensation

SENSISTOR COMPENSATION:

In the following fig2.21 shown a sensistor Rs having a positive temperature coefficient is connected across R_1 or R_E. Rs increases with temperature. As the temperature increases, the equivalent resistance of the parallel combination of R1 and Rs also increases and hence V_{BE}decreases, reducing I_B and Ic. This reduced Ic compensates for increased Ic caused by the increase in V_{BE}, I_{CO} and β due totemperature.

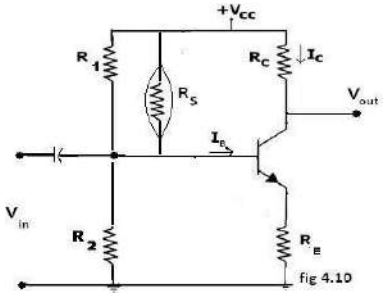


Fig 2.21 Sensistor Compensation

UNIT I-II

Signal Low Frequency BJT Amplifiers:

Transistor Hybrid model, Determination of h parameters from transistor characteristics, Typical values of h- parameters in CE, CB and CC configurations, Analysis of CE, CC, CB Amplifiers and CE Amplifier with emitter resistance, low frequency response of BJT Amplifiers

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**.

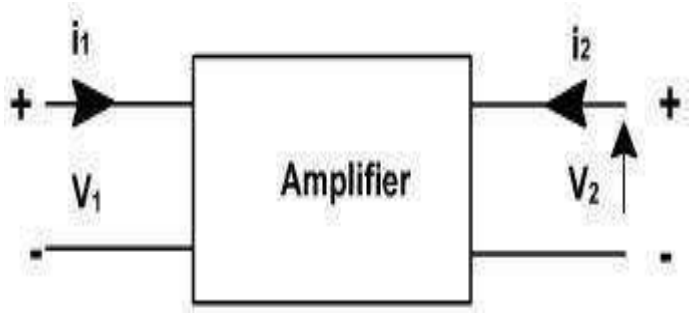


Fig. 3.1: Transistor as a two port Network

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

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

Input impedance with output port open circuited

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

Reverse transfer impedance with input port open circuited

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

Forward transfer impedance with output port open circuited

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

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

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

Reverse transfer admittance with input port short circuited

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

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

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

Reverse voltage transfer ratio with i/p port open circuited

$$h_{21} = \left.\frac{I_2}{I_1}\right|_{V_2=0}$$

Forward voltage transfer ratio with o/p port short circuited

$$h_{22} = \left.\frac{I_2}{V_2}\right|_{I_1=0}$$

output impedence with i/p port open circuited

THE HYBRID MODEL FOR TWO PORT NETWORK:

Based on the definition of hybrid parameters the mathematical model for two pert networks known as h-parameter model can be developed. The hybrid equations can be written as:

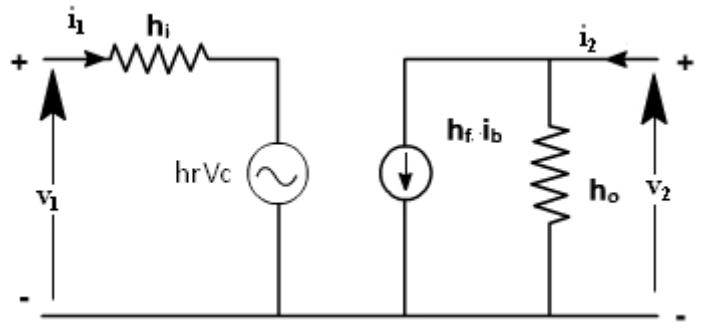
$$V_1 = h_i I_1 + h_r V_2$$

$$I_2 = h_f I_1 + h_o V_2$$

(The following convenient alternative subscript notation is recommended
by the **IEEE Standards**:
***i*=11=input** ***o* = 22 =output**

***f* =21 = forward transfer *r* = 12 = reverse transfer)**

We may now use the four h parameters to construct a mathematical model of the device of Fig.(1). The hybrid circuit for any device indicated in Fig.(2). We can verify that the model of Fig.(2) satisfies above equations by writing Kirchhoff'svoltage and current laws for input and output ports.



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. 3.2.

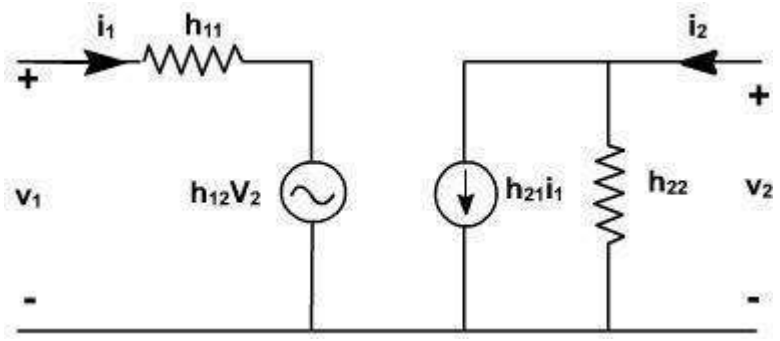


Fig.3.2:-h-parameter equivalent of Transistor

TRANSISTOR HYBRID MODEL:

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

Let us consider CE configuration as show in fig. 3.3. 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 dependent variables.

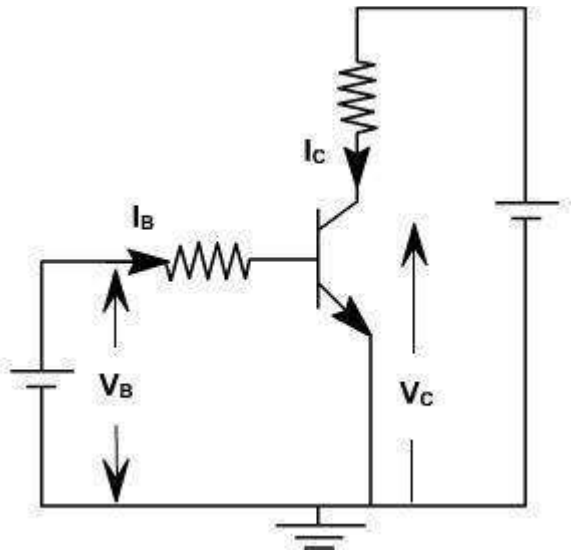


Fig. 3.3 CE Transistor Amplifier

$V_B = f1\ (i_B\ , v_C\ )$

$I_C = f2\ (i_B\ , v_C)$.

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

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

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

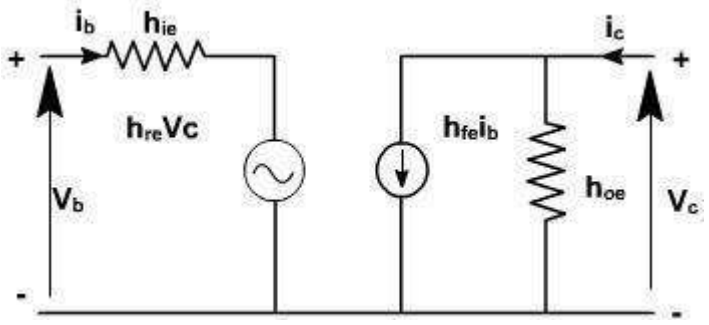


Fig. 3.4:h-parameter model of CE Configuration

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.

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

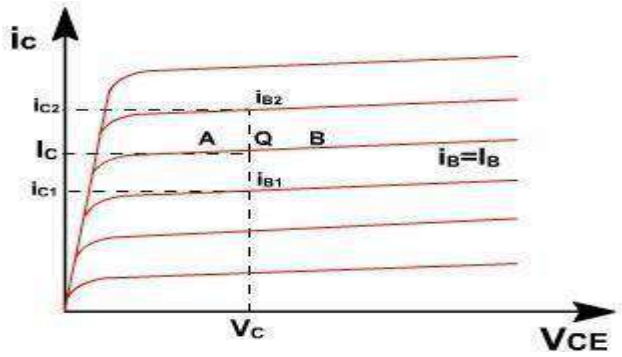


Fig. 3.5Transistor CE Configuration output characteristics

The current increments are taken around the quiescent point Q which corresponds to $i_B = I_B$ and to the collector voltage $V_{CE} = V_C$

$$h_{oe} = \left.\frac{\partial i_C}{\partial V_C}\right|_{i_B}$$

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).

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

h_{ie} is the slope of the appropriate input on fig. 3.6, at the operating point (slope of tangent EF at Q).

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

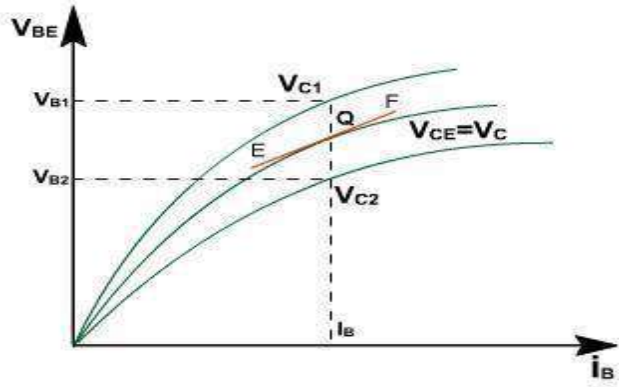


Fig. 3.6 Calculation of h-parameters from output characteristics

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} =25μA/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. 3.7**and to bias the transistor properly.

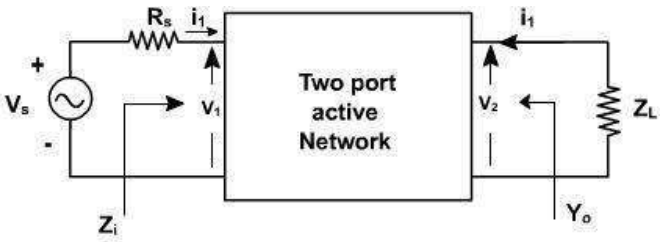


Fig. 3.7 Transistor two port Network

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 impedence, voltage gain, and output impedence.

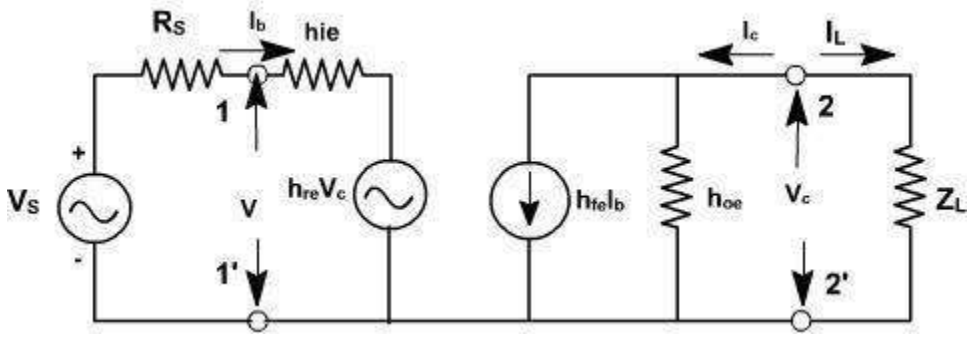


Fig 3.8:h parameter equivalent of Transistor in CE configuration

Current gain:

For the transistor amplifier stage, A_i is defined as the ratio of output to input currents.

$$A_I = \frac{I_L}{I_1} = \frac{-I_2}{I_1}$$

Input impedence:

The impedence looking into the amplifier input terminals (1,1') is the input impedance Z_i

$$Z_i = \frac{V_b}{I_b}$$

$$V_b = h_{ie} I_b + h_{re} V_c$$

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

$$= h_{ie} - \frac{h_{re} I_c Z_L}{I_b}$$

$$\therefore Z_i = h_{ie} + h_{re} A_I Z_L$$

$$= h_{ie} - \frac{h_{re} h_{fe} Z_L}{1 + h_{oe} Z_L}$$

$$\therefore Z_i = h_{ie} - \frac{h_{re} h_{fe}}{Y_L + h_{oe}} \qquad (\text{since } Y_L = \frac{1}{Z_L})$$

Voltage gain:

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

$$A_v = \frac{V_C}{V_b} = -\frac{I_C Z_L}{V_b}$$

$$\therefore A_v = \frac{I_B A_i Z_L}{V_b} = \frac{A_i Z_L}{Z_i}$$

Output Admittance:

$$Y_0 = \left.\frac{I_c}{V_c}\right|_{V_s} = 0$$

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

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

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

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

$$\therefore Y_0 = h_{oe} - \frac{h_{re}\, h_{fe}}{R_s + h_{ie}}$$

Voltage amplification taking into account source impedance (R_S) is given by

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

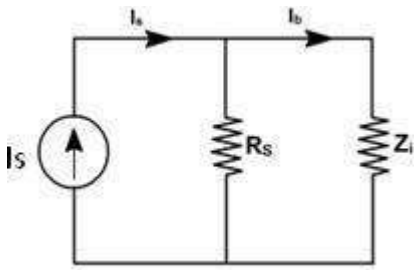
$$= A_V \cdot \frac{Z_i}{Z_i + R_s}$$

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

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

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

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

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

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

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

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. 4hrc in terms of CE parameter can be obtained as follows.

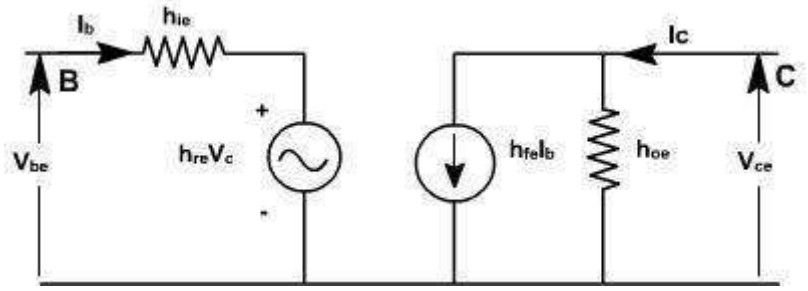


Fig. 3.9 CE h-parameter model

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

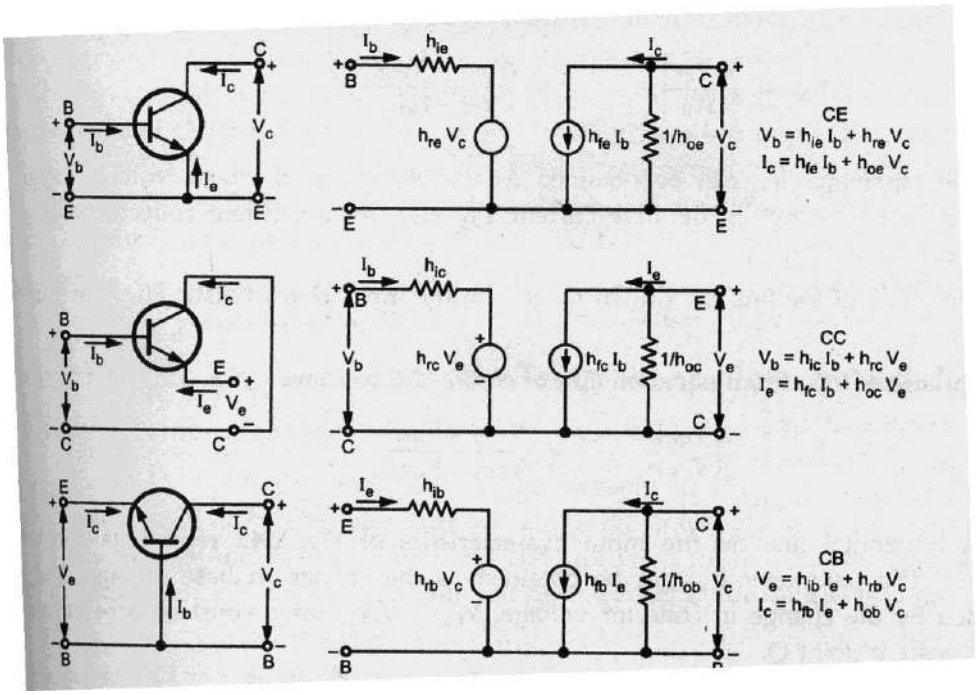


Fig 3.10 hybrid model for transistor in three different configurations

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.

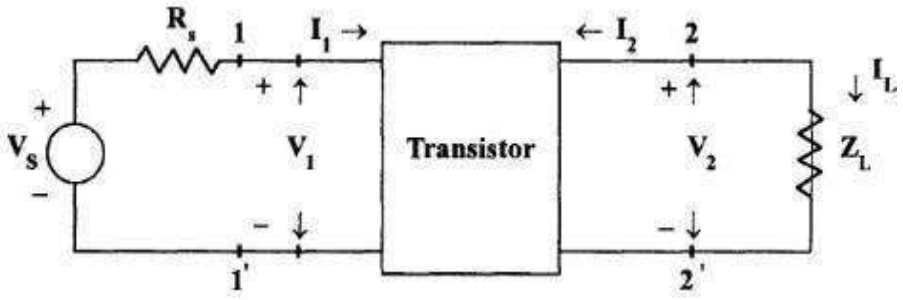


Fig.3.11 Basic Amplifier Circuit

The two port network of Fig. 3.11 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

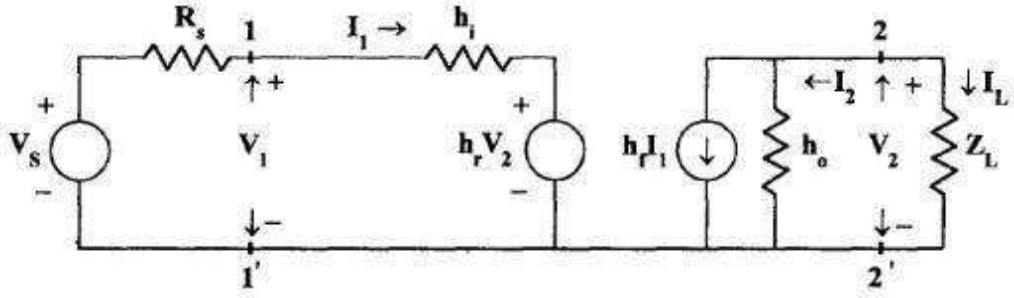


Fig. 3.12 Transistor replaced by its Hybrid Model

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 $V_2 = I_L Z_L = -I_2 Z_L$

$$I_2 = h_f I_1 - I_2 Z_L h_o$$

$I_2 + I_2 Z_L h_o = h_f I_1$

$I_2(1+ Z_L h_o) = h_f I_1$

$A_i = -I_2 / I_1 = - h_f / (1+ Z_L h_o)$

Therefore,

$A_i = - h_f / (1+ Z_L h_o)$

Input Impedence (Z_i)

In the circuitof Fig ,$R_S$isthesignalsourceresistance.Theimpedenceseenwhenlookingintothe amplifier terminals (1,1′) is the amplifier input impedenceZ_i,

$Z_i = V_1 / I_1$

From the input circuit of Fig $V_1 = h_i I_1 + h_r V_2$

$Z_i = (h_i I_1 + h_r V_2) / I_1$

$= h_i + h_r V_2 / I_1$

Substituting

$V_2 = -I_2 Z_L = A_1 I_1 Z_L$

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

$= h_i + h_r A_1 Z_L$

Substituting 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, $I_2 / V_2 = -h_r / (R_S + h_i)$

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

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

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

Voltage Amplification Factor(A_{vs}) taking into account the resistance (R_s) of the source

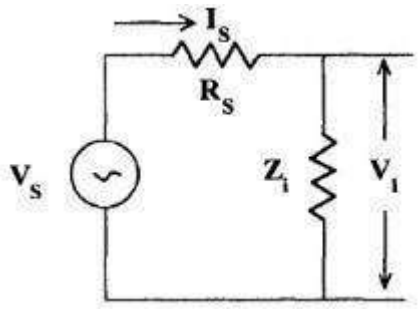


Fig. 3.13 Thevenin's Equivalent Input

Circuit This overall voltage gain A_{vs} is given by

$$A_{vs} = V_2 / V_S = V_2V_1 / V_1V_S = A_v V_1 / V_S$$

From the equivalent input circuit using Thevenin's equivalent for the source shown in Fig. 5.6

$V_1 = V_S Z_i / (Z_{i+} R_S)$

$V_1 / V_S = Z_i / (Z_i + R_S)$

Then, $A_{vs} = A_v Z_i / (Z_i + R_S)$

Substituting $A_v = A_i Z_L / Z_i$

$A_{vs} = A_i Z_L / (Z_i + R_S)$

$A_{vs} = A_i Z_L R_S / (Z_i + R_S) R_S$

$A_{vs} = A_{is} Z_L / R_S$

Current Amplification (A_{is}) taking into account the source Resistance(R_S)

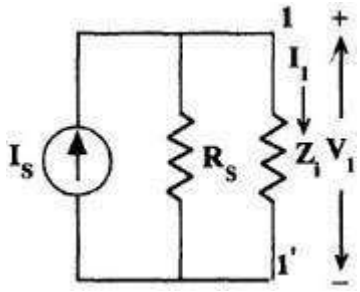


Fig. 3.14 Norton's Equivalent Input Circuit

The modified input circuit using Norton's equivalent circuit for the calculation of A_{is} is shown in Fig. 1.7

Overall Current Gain, $A_{is} = -I_2 / I_S = - I_2 I_1 / I_1 I_S = A_i I_1 / I_S$

FromFig.1.7 $I_1 = I_S R_S / (R_S + Z_i)$

$I_1 / I_S = R_S / (R_S + Z_i)$

andhence, $A_{is} = A_i R_S / (R_S + Z_i)$

Operating Power Gain (A_P)

The operating power gain A_P of the transistor is defined as

$A_P = P_2 / P_1 = -V_2 I_2 / V_1 I_1 = A_v A_i = A_i A_i Z_L / Z_i$

$A_P = A_i^2 (Z_L / Z_i)$

Small Signal analysis of a transistor amplifier

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

UNIT-II
Transistor at high Frequency Response

FREQUENCY RESPONSE OF AMPLIFIERS

For any electronic circuit, the behavior of amplifiers is affected by the frequency of the signal on their input terminal. This characteristic is known as the **frequency response**.

Frequency response is one of the most important property of amplifiers. In the frequency range that amplifiers have been designed for, they must deliver a constant and acceptable level of gain. The frequency response depends directly on the components and the architecture chosen for the design of the amplifier.

Before defining in details the frequency response, we need to present the unit of decibel (dB) and the logarithmic scale related to it. When studying the frequency response, it is indeed more suitable to convert either the power or voltage gain into dB and to represent the frequency scale in a logarithmic (log) scale.

If we consider an amplifier with power gain $\mathbf{A_P}$ and voltage gain $\mathbf{A_V}$, the power and voltage gain in dB are defined by:

$$A_P(dB) = 10\log(A_P)$$
$$A_V(dB) = 20\log(A_V)$$

While the gains in linear scale are always positive ($A_P, A_V \geq 0$), their equivalent in dB can either be positive if an amplification is being realized ($A_P, A_V > 1$) or negative if the input signal is attenuated ($A_P, A_V < 1$).

Often, it is not the gain $\mathbf{A_V(dB)}$ that is investigated but rather a normalized ratio $\mathbf{A_V/A_{V,mid}(dB)=20\log(A_V/A_{V,mid})}$. Where $A_{V,mid}$ is called the **midrange gain** and represents the maximum gain of the amplifier in its frequency working range, for example 20 Hz – 20 kHz for an audio amplifier.

Therefore, when $A_V=A_{V,mid}$, the normalized gain (written indifferently A_V) is $A_V(dB)=0$. This sets a 0 dB reference when the gain is maximum. It is important to note that when the power is divided by two, we observe that $\mathbf{A_P(dB)=10\log(0.5)=-3\ dB}$.

The frequency at which the power drops to 50 % of its midrange value is known as the **cutoff frequency** and noted $\mathbf{f_c}$. Each time that the power is halved, a reduction of 3 dB of the normalized gain is observed. Therefore $\mathbf{A_P=-3\ dB}$ corresponds to $\mathbf{A_{V,mid}/2}$, A_P=-6 dB corresponds to $A_{V,mid}/4$ and so on …

For this same frequency, the voltage (or current) is multiplied by a factor √2=0.7. Halving the voltage signal corresponds to a reduction of 6 dB and follows the same pattern as presented for the power gain.

The most common tool used to represent the frequency response of any system is the **Bode plot**. It consists of the normalized gain A_V(dB) as a function of the frequency in log scale. A simplified Bode graph of an amplifier is shown in the **Figure 1** below:

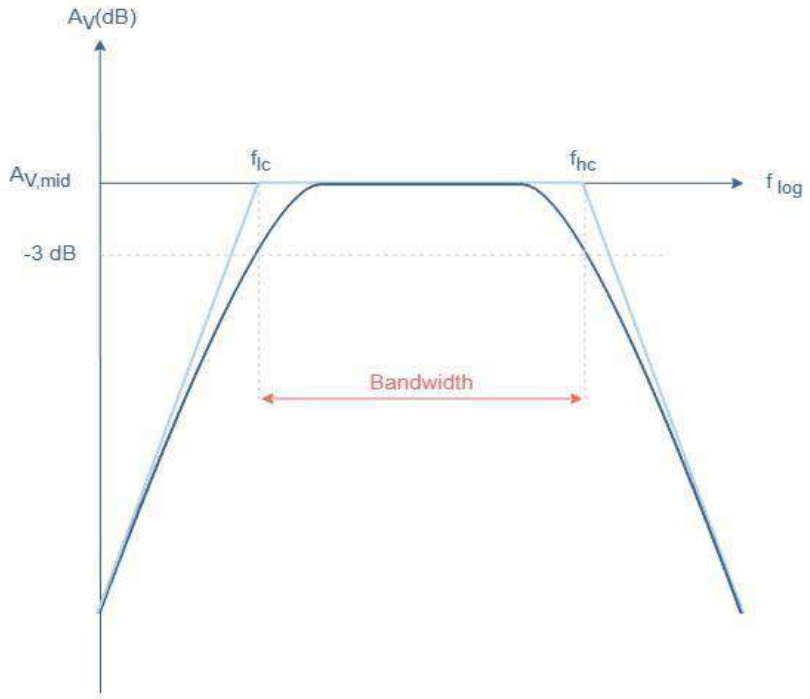


Fig 1 : Typical Bode graph of an amplifier

The light blue curve is called the asymptotic representation while the dark blue curve is the real frequency response of the circuit. In **Figure 1**, two different cutoff frequencies can be distinguished: $\mathbf{f_{lc}}$ for "low cutoff" and $\mathbf{f_{hc}}$ for "high cutoff". The quantity f_{hc}-f_{lc} is called the **bandwidth** and represents the frequency range where the gain is above the -3 dB.

EFFECT OF THE CAPACITORS:

Let us consider a Common Emitter Amplifier (CEA) which configuration is shown in **Figure 2.** The structure around the BJT transistor consists of a voltage divider network (R_1 and R_2), a load (R_L), coupling capacitors (C_1 and C_3) and a bypass capacitor C_2.

As capacitors have a property called **reactance** that is an equivalent of the resistance. The reactance (X_C) of capacitors depends on the frequency and the value of the capacitor, as in the below equation

$$X_C = \frac{1}{2\pi f C}$$

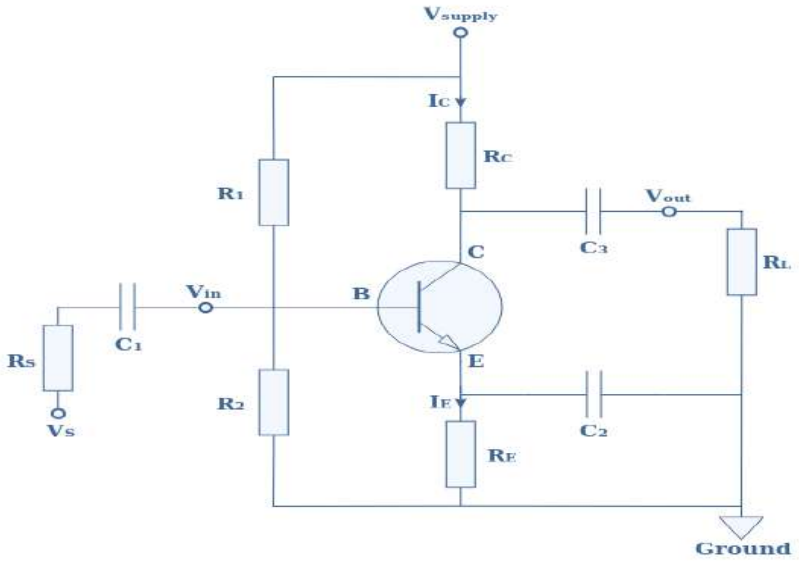


Fig 2: Common Emitter Amplifier

When the frequency is low, X_C tends to be high. Near DC signals, capacitors behave therefore as open circuits. On the other hand, when the frequency increases X_C tends to zero and capacitors act as short circuits.

At low input frequencies, the coupling capacitors will more likely block the signal, since X_{C1} and X_{C3} are higher, more voltage drop will be observed across C_1 and C_3. This results in a lower voltage gain.

At high input frequencies the bypass capacitor C_2 shortens the emitter branch to the ground and the voltage gain of the amplifier is $\mathbf{A_V=(R_C//R_L)/r_e}$ with r_e being the small diode emitter resistance. When the frequencies are lower, the resistance between the emitter and the ground is no longer only r_e but $\mathbf{R_E+r_e}$ and therefore the voltage gain decreases to $\mathbf{A_V=(R_C//R_L)/(R_E+r_e)}$.

There is another type of capacitors that affect the frequency response of the amplifier and is not represented in **Figure 2**. They are known as **internal transistor capacitors** and represented in **Figure 3** below :

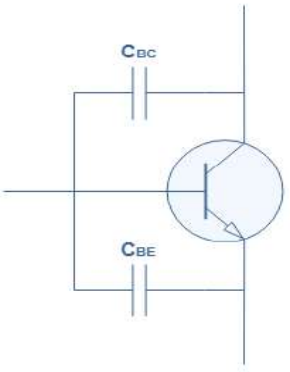


Fig 3: Internal transistor capacitors

Whereas the coupling and bypass capacitors act as **high-pass filter** (they block low frequencies), these internal capacitors behave differently. Indeed, if the frequency is low, C_{BC} and C_{BE} act as an open circuit and the transistor is not affected at all. However, if the frequency increases, more signal passes through them instead of going in the base branch of the transistor, therefore decreasing the voltage gain. The cutoff frequency of a RC filter:

$$f_c = \frac{1}{2\pi RC}$$

ANALYSIS AT LOW FREQUENCY

First of all we consider the input high-pass filter $R_{in}C_1$. Where R_{in} is the total input impedance of the amplifier which can be expressed as:

$$R_{in}=R_S+(R_1//R_2//\beta R_E)$$

The low cutoff frequency of the input will therefore be:

$$f_{cl,in}=1/(2\pi R_{in}C_1)$$

The same procedure can be done for the output where the output resistance is

$$R_{out}=R_C//R_L$$

The low cutoff frequency of the output filter is:

$$f_{cl,out}=1/(2\pi R_{out}C_3)$$

Finally, for the bypass capacitor, the resistance formula is more complex and given by

$$R_{bypass}=R_E//((r_e+(R_S//\beta R_E)/\beta))$$

The low cutoff frequency of the bypass structure is thus:

$$f_{cl,bypass}=1/(2\pi R_{bypass}C_2)$$

One last thing we need to understand before plotting the Bode graph is about the slope out of the midrange values. The decrease of $A_{V,mid}$ with the frequency is called **roll-off** and its value for each simple RC filter is -20 dB/decade (dB/dec). This value means for high-pass filters (resp. low-pass filters) that each time the frequency is divided by 10 (resp. multiplied by 10), a decrease of -20 dB is observed for the gain of the amplifier.

When multiple filters are blocking the same range of frequencies, the roll-off is enhanced. In our example three filters are simultaneously blocking the frequencies below 35 Hz, the roll-off is therefore 3*(-20 dB/dec)=-60 dB/dec.

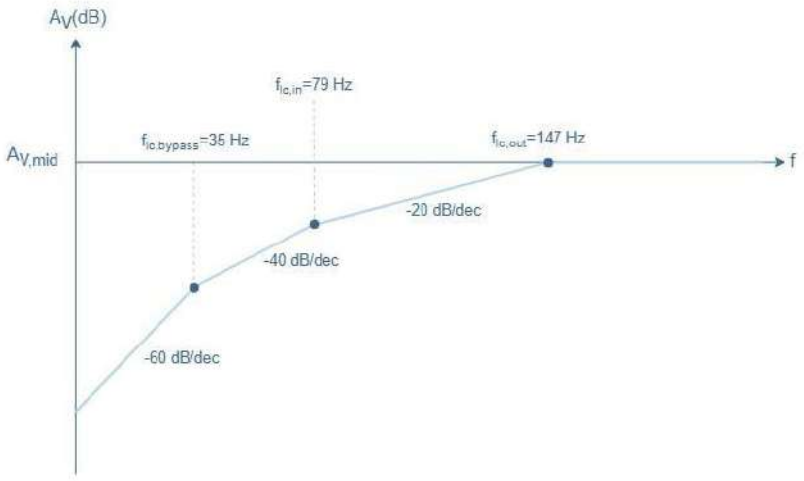


Fig 4 : Low frequency response of the CEA

ANALYSIS AT HIGH FREQUENCY

As stated previously, it is the internal transistor capacitors that will limit the gain at high frequencies acting as low-pass filters. It can be shown that the equivalent circuit of **Figure 2** at high frequency can be drawn such as presented in **Figure 5** :

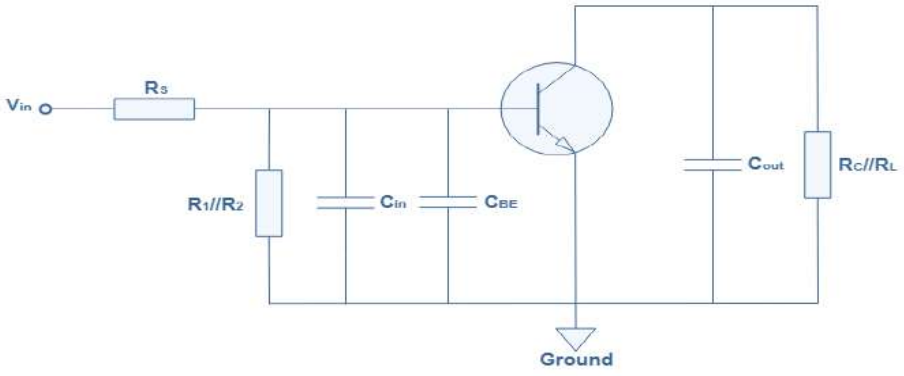


Fig 5 : Equivalent CEA at high frequency

We can note that the coupling capacitors are not represented since they behave as short circuits at high frequencies. Moreover, the emitter branch is shorten to the ground for the same reason applying to the bypass capacitor.

The internal capacitor C_{BC} is converted via **Miller's theorem** into the equivalent C_{in} and C_{out} capacitors. Moreover, this theorem states that

$$C_{in}=C_{BC}(A_{V,mid}+1) \text{ and}$$

$$C_{out}=C_{BC}(A_{V,mid}+1)/A_{V,mid}.$$

The total input capacitance of this circuit is

$$C_{IN}=C_{BE}+C_{in} ;$$

The total input resistance is

$$R_{IN}=R_S//R_1//R_2//\beta r_e.$$

The numerical application to our example gives

$A_{V,mid}=(R_C//R_L)/r_e=108$, $\mathbf{C_{IN}=575\ pF}$ and $\mathbf{R_{IN}=409\ \Omega.}$

The high cutoff frequency of the input is therefore

$$f_{hc,in}=1/(2\pi R_{IN}C_{IN})=677\ kHz.$$

From the output point of view, the high cutoff frequency is simply given by the filter $(R_C//R_L)C_{out}$ with $C_{out}=5.3\ pF$: $f_{hc,out}=1/(2\pi(R_C//R_L)C_{out})=1.1\ MHz$.

The information given here is summarized in a Bode plot representing the high frequency response of the CEA in asymptotic representation:

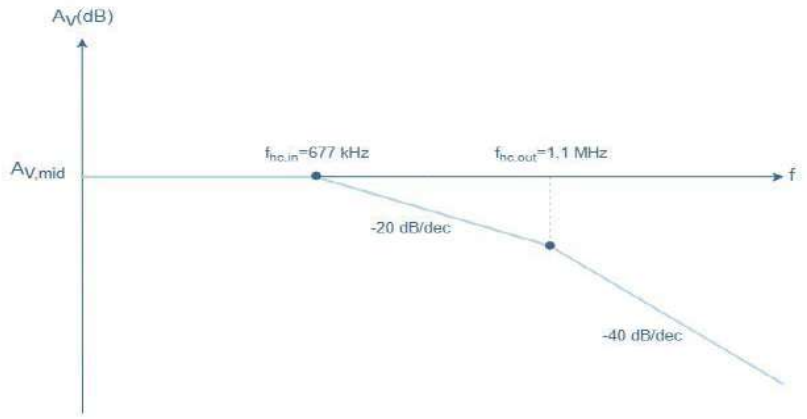


Fig 6: High frequency response of the CEA

By merging the two Bode graphs obtained for the low and high frequency responses in **Figure 4 and 6**, we can now plot the overall frequency response of the CEA configuration

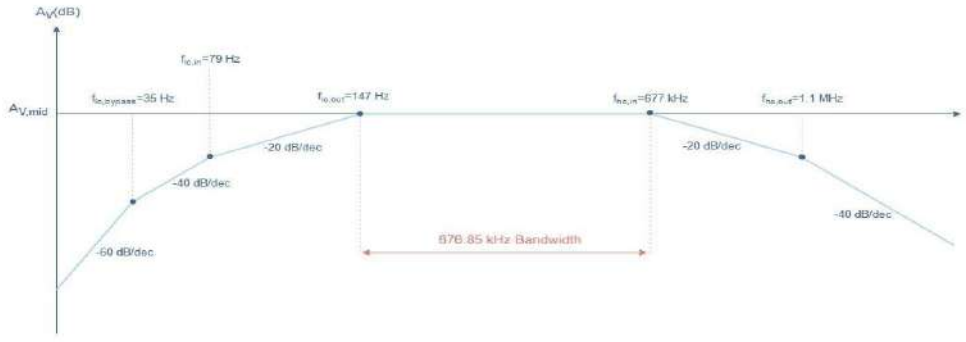


Fig 7: Total frequency response of the CEA

Hybrid-pi (π) common emitter transistor model

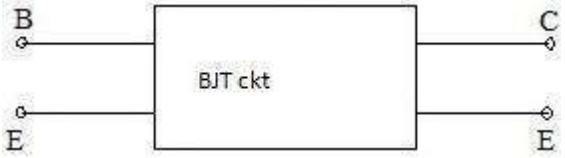
For *amplifier circuits Common Emitter configuration is preferred* Because for Common Collector (hrc < 1). For Common Collector Configuration, voltage gain Av < 1. So even by cascading you can't increase voltage gain. For Common Base, current gain hib < 1. So overall voltage gain is < 1. But for Common Emitter, hre» 1. Therefore Voltage gain can be increased by cascading Common Emitter stage. So Common Emitter configuration is widely used.

Under reverse bias condition the capacitance at the junction is called transition or space charge capacitance. Under forward bias condition the capacitance is called diffusion or storage capacitance. At high frequencies, BJT cannot be analysed by h-parameters.

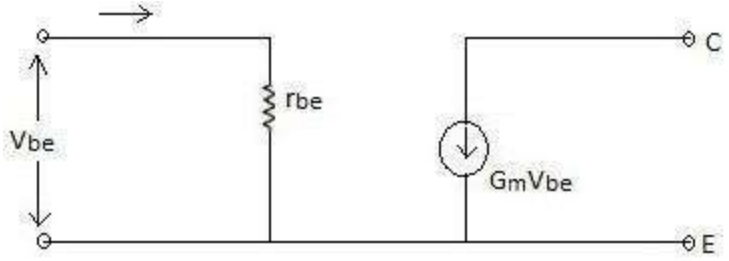
Giacolleto model - hybrid π equivalent circuit

Desirable features of hybrid π equivalent circuit are:
(1) The value of components in the equivalent circuit are independent of frequencies.
(2) The values of all the resistive components in the equivalent circuit can be determined from the known or Specified values of h-parameters at low frequencies.
(3) The results obtained by using this equivalent circuit agrees with the experimental result.

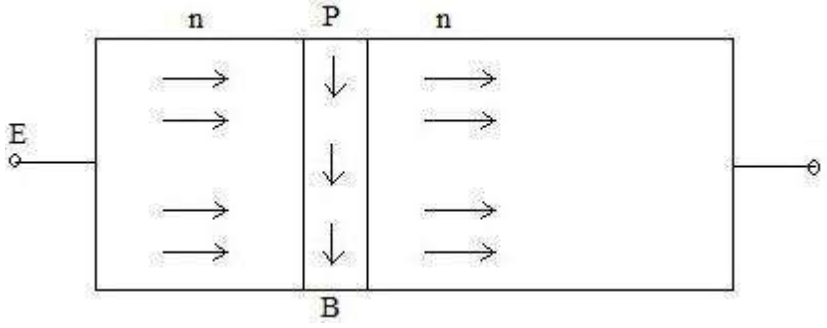
The components of the equivalent circuit exist in the form of π hence the name.

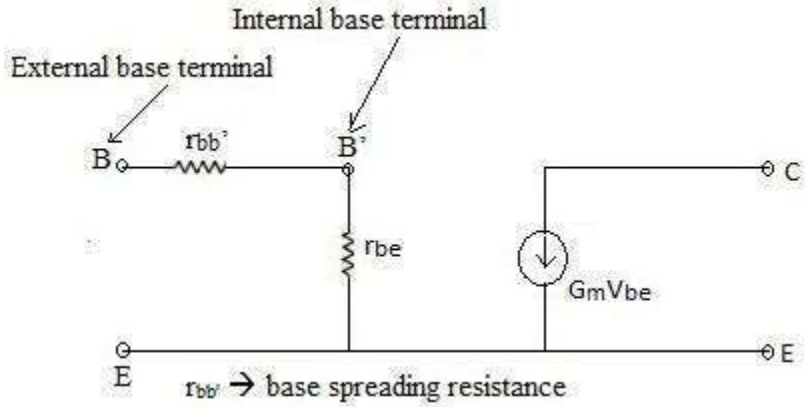


For small signal behaviour the transistor at its input port behaves as a resistor.

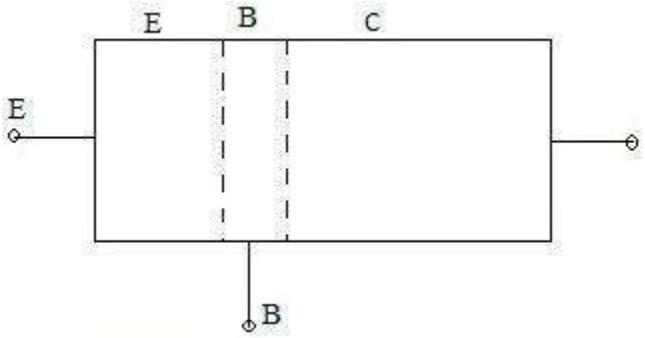


The output port is a dependent current source.

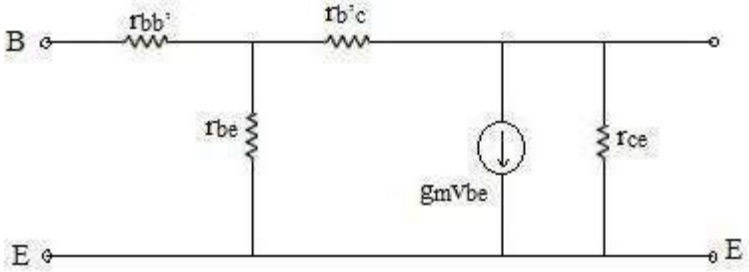




Because the base (B) is lightly doped all the depletion region lies entirely in the Base region. So, when the collector voltage is increased the depletion region in the base increases.



rce --> This resistance is added to compensate for the change in IC due to change in VCE.



The *Hybrid-π* or *Giacoletto Model* for the Common Emitter amplifier circuit (single stage) is as shown :

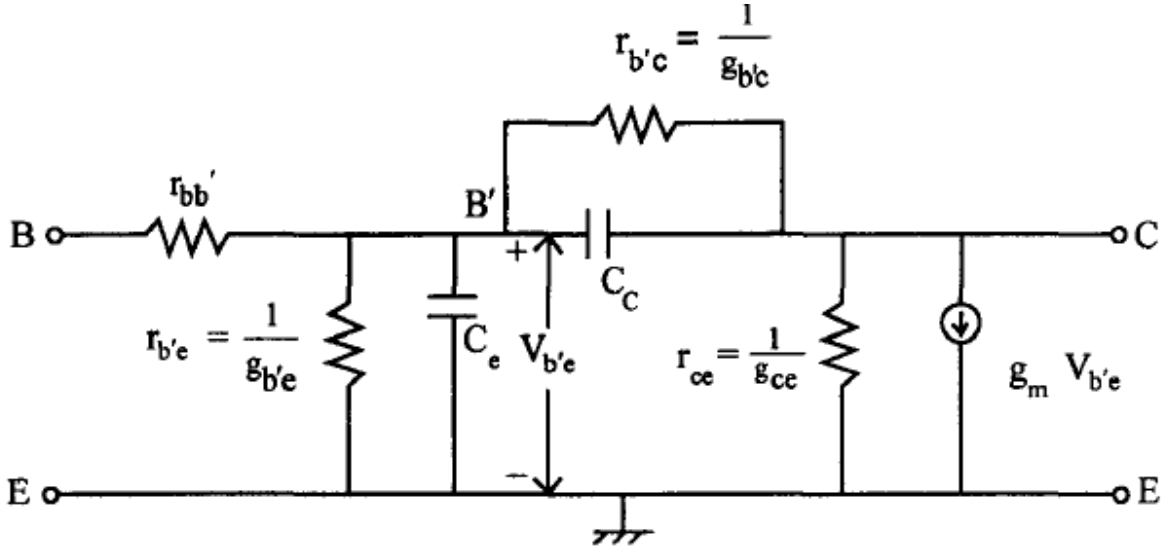


Fig 8 : Hybrid-π CE BJT Model

Analysis of this circuit gives satisfactory results at *all* frequencies not only at *high frequencies* but also at *low frequencies.* All the parameters are assumed to be independent of frequency.

Circuit Components

B' is the internal node of the base of the Transconductance amplifier. It is not physically accessible.
The base spreading resistance rb'b is represented as a *lumped* parameter between base B and internal node B'. ($g_m V_{b'e}$) is a current generator. Vb'e is the input voltage across the emitter junction. If $V_{b'e}$ increases, more carriers are injected into the base of the transistor. So the increase in the number of carriers is α $V_{b'e}$). This results in small signal current (since we are taking into account changes in $V_{b'e}$). This effect is represented by the current generator $g_m V_{b'e}$. This represents the current that results because of changes in $V_{b'e}$ when C is shorted to E.

When the number of carriers injected into the base increase, base recombination also increases. So this effect is taken care of by$g_{b'e}$. As recombination increases, base current increases. Minority carrier storage in the *base is* represented by c_e the diffusion capacitance.

According to Early Effect, the change in voltage between Collector and Emitter changes the base width. So base width will be modulated according to the voltage between Collector and Emitter. When base width changes, the minority carrier concentration in base changes. Hence the current which is proportional to carrier concentration also changes. So I_E changes and hence I_C changes. This feedback effect [I_E on input side, I_C on output side] is taken into account by connecting g_{bF_c} between B', and C. The conductance between Collector and Base isg_{ce}. C_c represents the collector junction barrier capacitance.

The High frequency model parameters of a BJT in terms of low frequency hybrid parameters is given below

- Tran conductance $g_m = I_C/V_T$
- Internal Base node to emitter resistance $r_{bF_e} = h_{fe}/ g_m = (h_{fe}*V_T)/ I_C$
- Internal Base node to collector resistance r_{bF_c} = (h_{re}* r_{bF_e}) / (1-h_{re}) assuming hre << 1 it reduces to r_{bF_c} = (h_{re}* r_{bF_e})
- Base spreading resistance $r_{bbF} = h_{ie} - r_{bF_e}$ = h_{ie} – (h_{fe}* Vt)/ Ic
- Collector to emitter resistance r_{ce} = 1 / (h_{oe} – (1+ h_{fe})/ r_{bF_c})

Variation of Hybrid Parameters with $|I_C|$, $|V_{CE}|$and T

1) **Transconductance Amplifier or Mutual Conductance (gm):**

$g_m = I_C/V_T$

g_m is α I_C

V_T = T/ll,600

Therefore g_m α 1/T

g_m is independent of V_{CE}
Since in the active region of the transconductance, Ic is independent of V_{CE}

UNIT II-II

FET Amplifiers: Analysis of Common Source and Common Drain JFET Amplifiers, Comparison of performance with BJT Amplifiers.

BIASING FET:-

For the proper functioning of a linear FET amplifier, it is necessary to maintain the operating point Q stable in the central portion of the pinch off region The Q point should be independent of device parameter variations and ambient temperature variations

This can be achieved by suitably selecting the gate to source voltage VGS and drain current ID which is referred to as biasing

JFET biasing circuits are very similar to BJT biasing circuits The main difference between JFET circuits and BJT circuits is the operation of the active components themselves

There are mainly two types of Biasing circuits

1) Self bias
2) Voltage divider-bias.

SELFBIAS

Self bias is a JFET biasing circuit that uses a source resistor to help reverse bias the JFET gate. A self bias circuit is shown in the fig. Self bias is the most common type of JFET bias. This JFET must be operated such that gate source junction is always reverse biased. This condition requires a negative VGS for an N channel JFET and a positive VGS for P channel JFET. This can be achieved using the self bias arrangement as shown in Fig. The gate resistor RG doesn't affect the bias because it has essentially no voltage drop across it, and: the gate remains at 0V .RG is necessary only to isolate an ac signal from ground in amplifier applications. The voltage drop across resistor RS makes gate source junction reverse biased.

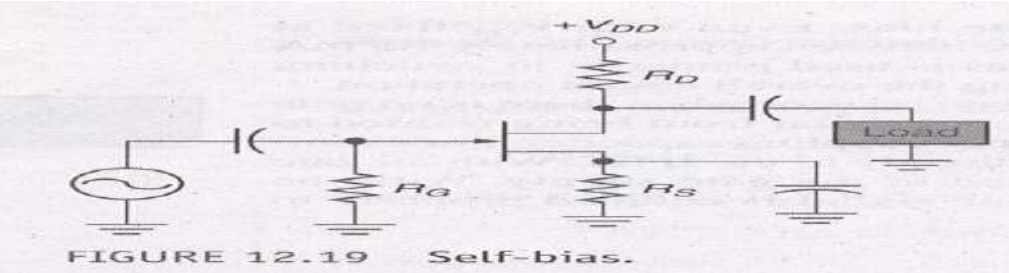


FIGURE 12.19 Self-bias.

For the dc analysis coupling capacitors are open circuits.

For the N channel FET in Fig (a)

IS produces a voltage drop across RS and makes the source positive w.r.t ground. In any JFET circuit all the source current passes through the device to the drain circuit .This is due to the fact that there is no significant gate current.

We can define source current as IS = ID

(VG =0 because there is no gate current flowing in RG So VG across RG is zero)

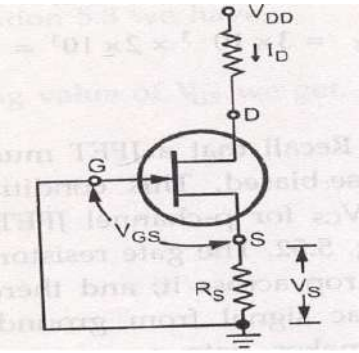
VG =0 then VS= ISRS =ID RS

VGS = VG-VS =0-ID RS=- ID RS

DC analysis of self Bias:-

In the following DC analysis, the N channel J FET shown in the fig. is used for illustration.

For DC analysis we can replace coupling capacitors by open circuits and we can also replace the resistor RG by a short circuit equivalent.:. IG = 0.The relation between ID and VGS is given by



$$Id=Idss[1-\frac{Vgs}{Vp}]^2$$

VGS for N channel JFET is =-idRs

Substuting this value in the above equation

$$Id=Idss[1-\frac{(-IdRs)}{Vp}]^2$$

$$Id=Idss[1+\frac{(IdRs)}{Vp}]^2$$

For the N-chanel FET in the above figure

Is produces a voltage drop across Rs and makes the source positive w.r.t ground in any JFET circuit all the source current passes through the device to drain circuit this is due to the fact that there is no significant gate current. Therefore we can define source current as Is=Id and Vg=0 then

Vs= Is Rs =IdRs

Vgs=Vg-Vs=0-IdRs=-IdRs

Drawing the self bias line:-

Typical transfer characteristics for a self biased JFET are shown in the fig.

The maximum drain current is 5mA and the gate source cut off voltage is -3V. This means the gate voltage has to be between 0 and -3V.

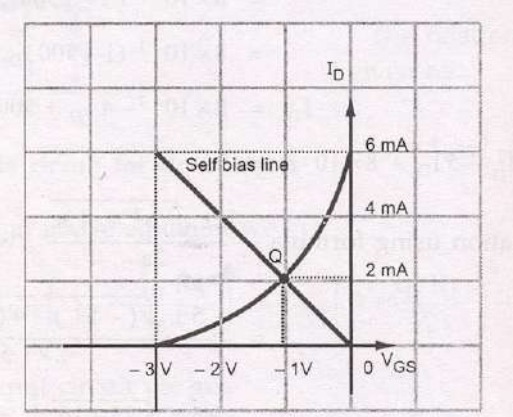


Fig. 5.55 Transfer characteristics

Now using the equation VGS = -IDRS and assuming RS of any suitable value we can draw the self bias line.

Let us assume RS = 500Ω

With this Rs , we can plot two points corresponding to ID = 0 and Id = IDSS

for ID =0

VGS = -ID RS

VGS = 0X (500.Ω) = 0V

So the first point is (0 ,0)

(I_d, VGS)

For ID= IDSS=5mA

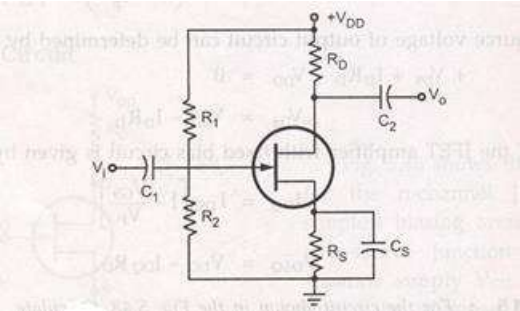
VGS = (-5mA) (500 Ω) = -3V

So the 2nd Point will be (5mA,-3V)

By plotting these two points, we can draw the straight line through the points. This line will intersect the transconductance curve and it is known as self bias line.The intersection point gives the operating point of the self bias JFET for the circuit.

At Q point , the ID is slightly > than 2mA and VGS is slightly > -1V. The Q point for the self bias JFET depends on the value of Rs.If Rs is large, Q point far down on the transconductance curve ,ID is small, when Rs is small Q point is far up on the curve , ID is large.

VOLTAGE DIVIDERBIAS:-



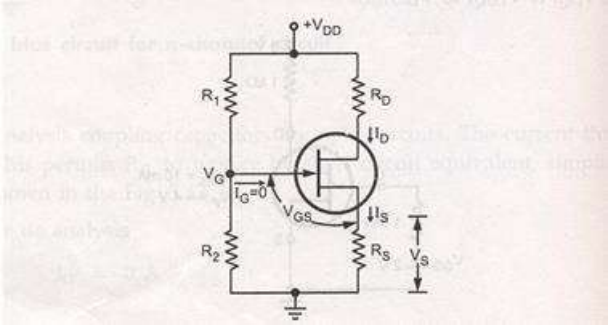
The fig. shows N channel JFET with voltage divider bias. The voltage at the source of JFET must be more positive than the voltage at the gate in order to keep the gate to source junction reverse biased. The source voltage is

VS = IDRS

The gate voltage is set by resistors R1 and R2 as expressed by the following equation using the voltage divider formula.

$$Vg=\left(\frac{R2}{(R1+R2)}\right)Vdd$$

For dc analysis



Applying KVL to the input circuit

VG-VGS-VS =0

:: VGS = VG-Vs=VG-ISRS

VGS=VG-IDRS :: IS =ID

Applying KVL to the input circuit we get

VDS+IDRD+VS-VDD =0

::VDS = VDD-IDRD-IDRS

VDS = VDD-ID (RD +RS)

The Q point of a JFET amplifier , using the voltage divider bias is

IDQ = IDSS[1-VGS/VP]2

VDSQ = VDD-ID (RD+RS)

COMPARISON OF MOSFET WITH JFET

a. In enhancement and depletion types of MOSFET, the transverse electricfield induced across an insulating layer deposited on the semiconductor material controls the conductivity of thechannel.

b. In the JFET the transverse electric field across the reverse biased PN junction controls the conductivity of thechannel.

c. The gate leakage current in a MOSFET is of the order of 10^{-12}A. Hence the input resistance of a MOSFET is very high in the order of 10^{10} to 10^{15} Ω. The gate leakage current of a JFET is of the order of 10^{-9}A., and its input resistance is of the order of10^{8}Ω.

d. TheoutputcharacteristicsoftheJFETareflatterthanthoseoftheMOSFET,andhencethe drain resistance of a JFET (0.1 to 1MΩ) is much higher than that of a MOSFET (1 to50kΩ).

e. JFETs are operated only in the depletion mode. The depletion type MOSFET may be operated in both depletion and enhancementmode.

f. Comparing to JFET, MOSFETs are easier tofabricate.

g. Special digital CMOS circuits are available which involve near zero power dissipation and very low voltage and current requirements. This makes them suitable for portable systems.

FET AMPLIFIERS

INTRODUCTION

Field Effect Transistor (FET) amplifiers provide an excellent voltage gain and high input impedence. Because of high input impedence and other characteristics of JFETs they are preferred over BJTs for certain types of applications.

There are 3 basic FET circuit configurations:

i) Common Source

ii)Common Drain

iii)Common Gain

Similar to BJT CE,CC and CB circuits, only difference is in BJT large output collector current is controlledbysmallinputbasecurrentwhereasFETcontrolsoutputcurrentbymeansofsmallinput voltage. In both the cases output current is controlledvariable.

FET amplifier circuits use voltage controlled nature of the JFET. In Pinch off region, I_D depends only on V_{GS}.

Common Source (CS)Amplifier

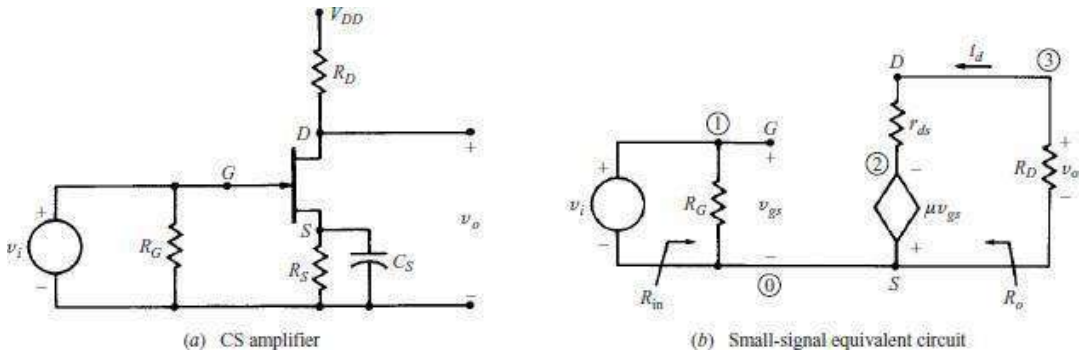


Fig. 5.1 (a) CS Amplifier (b) Small-signal equivalent circuit

A simple Common Source amplifier is shown in Fig. 5.1(a) and associated small signal equivalent circuit using voltage-source model of FET is shown in Fig. 5.1(b)

Voltage Gain

Source resistance (R_S) is used to set the Q-Point but is bypassed by C_S for mid-frequency operation.

From the small signal equivalent circuit ,the output voltage

$$V_O = -R_D \mu V_{gs}(R_D + r_d)$$

Where $V_{gs} = V_i$, the input voltage,

Hence, the voltage gain,

$$A_V = V_O / V_i = -R_D \mu (R_D + r_d)$$

Input Impedence

From Fig. 5.1(b) Input Impedence is

$$Z_i = R_G$$

For voltage divider bias as in CE Amplifiers of BJT

$$R_G = R_1 \| R_2$$

Output Impedance

Output impedance is the impedance measured at the output terminals with the input voltage $V_I = 0$

From the Fig. 5.1(b) when the input voltage $V_i = 0$, $V_{gs} = 0$ and hence

$$\mu V_{gs} = 0$$

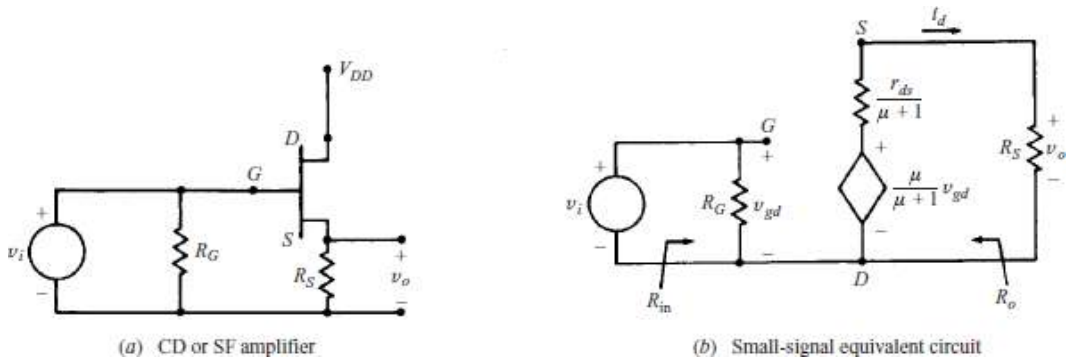
The equivalent circuit for calculating output impedence is given in Fig. 5.2.

Output impedence $Z_o = r_d \| R_D$

Normally r_d will be far greater than R_D . Hence $Z_o \approx R_D$

Common DrainAmplifier

A simple common drain amplifier is shown in Fig. 5.2(a) and associated small signal equivalent circuit using the voltage source model of FET is shown in Fig. 5.2(b).Since voltage V_{gd} is more easily determined than V_{gs}, the voltage source in the output circuit is expressed in terms of V_{gs} and Thevenin's theorem.



(a) CD or SF amplifier (b) Small-signal equivalent circuit

Fig. 5.2 (a)CD Amplifier (b)Small-signal equivalent circuit

Voltage Gain

The output voltage,

$$V_O = R_S \mu V_{gd} / (\mu + 1) R_S + r_d$$

Where $V_{gd} = V_i$ the input voltage.

Hence, the voltage gain,

$$A_v = V_O / V_i = R_S \mu / (\mu + 1) R_S + r_d$$

Input Impedence

From Fig. 5.2(b), Input Impedence $Z_i = R_G$

Output Impedence

From Fig. 5.2(b), Output impedence measured at the output terminals with input voltage $V_i = 0$ can be calculated from the following equivalent circuit.

As $V_i = 0$: $V_{gd} = 0$: $\mu v_{gd} / (\mu + 1) = 0$

Output Impedence

$$Z_O = r_d / (\mu + 1) \parallel R_S$$

When $\mu \gg 1$

$$Z_O = (r_d / \mu) \parallel R_S = (1/g_m) \parallel R_S$$

BIASING FET

For the proper functioning of a linear FET amplifier, it is necessary to maintain the operating point Q stable in the central portion of the pinch off region The Q point should be independent of device parameter variations and ambient temperature variations

This can be achieved by suitably selecting the gate to source voltage VGS and drain current ID which is referred to as biasing

JFET biasing circuits are very similar to BJT biasing circuitsThe main difference between JFET circuits and BJT circuits is the operation of the active components themselves

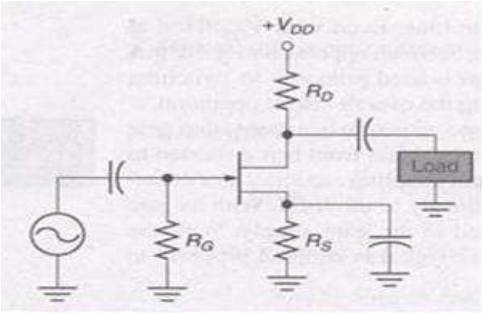
There are mainly two types of Biasing circuits

1. Selfbias
2. Voltage dividerbias.

5.13.1. SELF BIAS:-

Self bias is a JFET biasing circuit that uses a source resistor to help reverse bias the JFET gate.

A self bias circuit is shown in the fig 5.3



Self bias is the most common type of JFET bias.
This JFET must be operated such that gate source junction is always reverse biased.
This condition requires a negative VGS for an N channel JFET and a positive VGS for P channel JFET.
This can be achieved using the self bias arrangement as shown in Fig 5.3.
The gate resistor RG doesn't affect the bias because it has essentially no voltage drop across it, and :
the gate remains at 0V .RG is necessary only to isolate an ac signal from ground in amplifier applications.
The voltage drop across resistor RS makes gate source junction reverse biased.

DC analysis of self Bias:-

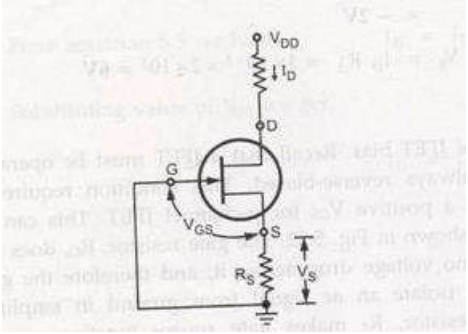
In the following DC analysis , the N channel J FET shown in the fig5.4. is used for illustration.

ForDCanalysiswecanreplacecouplingcapacitorsbyopencircuitsand we canalsoreplace the resistor RG by a short circuitequivalent.

:. IG = 0

The relation between ID and VGS is given by

$$Id=Idss[1-\frac{Vgs}{Vp}]^2$$



VGS for N channel JFET is =-id Rs

Substuting this value in the above equation

$$Id=Idss[1-\frac{(-IdRs)}{Vp}]^2$$

$$Id=Idss[1+\frac{(IdRs)}{Vp}]^2$$

For the N-chanel FET in the above figure

Is produces a voltage drop across Rs and makes the source positive w.r.t ground

in any JFET circuit all the source current passes through the device to drain circuit this is due to the fact that there is no significant gate current

therefore we can define source current as Is=Id and Vg=0 then

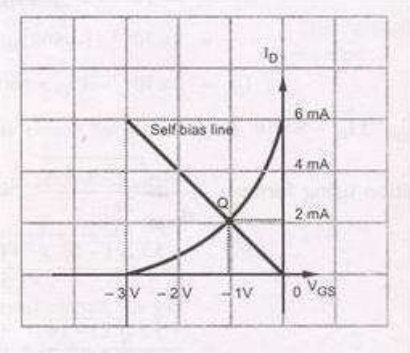
Vs= Is Rs =IdRs

Vgs=Vg-Vs=0-IdRs=-IdRs

Drawing the self bias line:-

Typical transfer characteristics for a self biased JFET are shown in the figure 5.5 below:

The maximum drain current is 6mA and the gate source cut off voltage is -3V. This means the gate voltage has to be between 0 and -3V.



Now using the equation VGS = -IDRS and assuming RS of any suitable value we can draw the self bias line.

Let us assume RS = 500Ω

With this Rs , we can plot two points corresponding to ID = 0 and Id =IDSS

for ID =0

VGS = -ID RS

VGS = 0X (500.Ω) = 0V

So the first point is (0 ,0)

(I_d, VGS)

For ID= IDSS=6mA

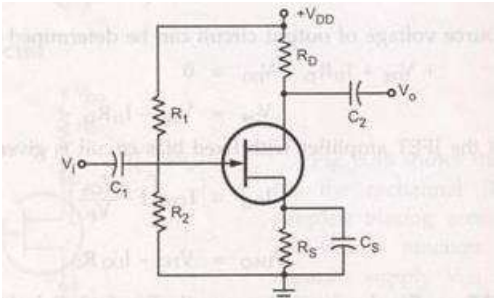
VGS = (-6mA) (500 Ω) = -3V

So the 2nd Point will be (6mA,-3V)

By plotting these two points, we can draw the straight line through the points. This line will intersect the transconductance curve and it is known as self bias line. The intersection point gives the operating point of the self bias JFET for the circuit.

At Q point , the ID is slightly > than 2mA and VGS is slightly > -1V. The Q point for the self bias JFET depends on the value of Rs.If Rs is large, Q point far down on the transconductance curve ,ID is small, when Rs is small Q point is far up on the curve , ID is large.

5.13.2 VOLTAGE DIVIDER BIAS:-



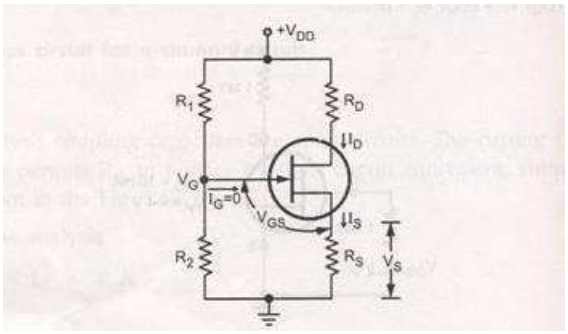
The fig5.6 shows N channel JFET with voltage divider bias. The voltage at the source of JFET must be more positive than the voltage at the gate in order to keep the gate to source junction reverse biased. The source voltage is

VS = IDRS

The gate voltage is set by resistors R1 and R2 as expressed by the following equation using the voltage divider formula.

Vg= $Vdd\left(\frac{R2}{(R1+R2)}\right)$

For dc analysis fig 5.5



Applying KVL to the input circuit

VG-VGS-VS =0

:: VGS = VG-Vs=VG-ISRS

VGS=VG-IDRS :: IS =ID

Applying KVL to the input circuit we get

VDS+IDRD+VS-VDD =0

::VDS = VDD-IDRD-IDRS

VDS = VDD-ID (RD +RS)

The Q point of a JFET amplifier , using the voltage divider bias is

IDQ = IDSS[1-VGS/VP]2

VDSQ = VDD-ID (RD+RS)

UNIT III

Number System and Boolean Algebra

If base or radix of a number system is 'r', then the numbers present in that number system are ranging from zero to r-1. The total numbers present in that number system is 'r'. So, we will get various number systems, by choosing the values of radix as greater than or equal to two.

In this chapter, let us discuss about the **popular number systems** and how to represent a number in the respective number system. The following number systems are the most commonly used.

- Decimal Number system
- Binary Number system
- Octal Number system
- Hexadecimal Number system

Decimal Number System

The **base** or radix of Decimal number system is **10**. So, the numbers ranging from 0 to 9 are used in this number system. The part of the number that lies to the left of the **decimal point**is known as integer part. Similarly, the part of the number that lies to the right of the decimal point is known as fractional part.

In this number system, the successive positions to the left of the decimal point having weights of 10^0, 10^1, 10^2, 10^3 and so on. Similarly, the successive positions to the right of the decimal point having weights of 10^{-1}, 10^{-2}, 10^{-3} and so on. That means, each position has specific weight, which is **power of base 10**

Example

Consider the **decimal number 1358.246**. Integer part of this number is 1358 and fractional part of this number is 0.246. The digits 8, 5, 3 and 1 have weights of 100, 101, 10^2 and 10^3respectively. Similarly, the digits 2, 4 and 6 have weights of 10^{-1}, 10^{-2} and 10^{-3} respectively.

Mathematically, we can write it as

$1358.246 = (1 \times 10^3) + (3 \times 10^2) + (5 \times 10^1) + (8 \times 10^0) + (2 \times 10^{-1}) + (4 \times 10^{-2}) + (6 \times 10^{-3})$

After simplifying the right hand side terms, we will get the decimal number, which is on left hand side.

Binary Number System

All digital circuits and systems use this binary number system. The **base** or radix of this number system is **2**. So, the numbers 0 and 1 are used in this number system.

The part of the number, which lies to the left of the **binary point** is known as integer part. Similarly, the part of the number, which lies to the right of the binary point is known as fractional part.

In this number system, the successive positions to the left of the binary point having weights of 2^0, 2^1, 2^2, 2^3 and so on. Similarly, the successive positions to the right of the binary point having weights of 2^{-1}, 2^{-2}, 2^{-3} and so on. That means, each position has specific weight, which is **power of base 2**.

Example

Consider the **binary number 1101.011**. Integer part of this number is 1101 and fractional part of this number is 0.011. The digits 1, 0, 1 and 1 of integer part have weights of 2^0, 2^1, 2^2, 2^3respectively. Similarly, the digits 0, 1 and 1 of fractional part have weights of 2^{-1}, 2^{-2}, 2^{-3} respectively.

Mathematically, we can write it as

$1101.011 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (0 \times 2^{-1}) +(1 \times 2^{-2}) + (1 \times 2^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of binary number on left hand side.

Octal Number System

The **base** or radix of octal number system is **8**. So, the numbers ranging from 0 to 7 are used in this number system. The part of the number that lies to the left of the **octal point** is known as integer part. Similarly, the part of the number that lies to the right of the octal point is known as fractional part.

In this number system, the successive positions to the left of the octal point having weights of 8^0, 8^1, 8^2, 8^3 and so on. Similarly, the successive positions to the right of the octal point having weights of 8^{-1}, 8^{-2}, 8^{-3} and so on. That means, each position has specific weight, which is **power of base 8**.

Example

Consider the **octal number 1457.236**. Integer part of this number is 1457 and fractional part of this number is 0.236. The digits 7, 5, 4 and 1 have weights of 8^0, 8^1, 8^2 and 8^3respectively. Similarly, the digits 2, 3 and 6 have weights of 8^{-1}, 8^{-2}, 8^{-3} respectively.

Mathematically, we can write it as

$1457.236 = (1 \times 8^3) + (4 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) +(3 \times 8^{-2}) + (6 \times 8^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of octal number on left hand side.

Hexadecimal Number System

The **base** or radix of Hexa-decimal number system is **16**. So, the numbers ranging from 0 to 9 and the letters from A to F are used in this number system. The decimal equivalent of Hexa-decimal digits from A to F are 10 to 15.

The part of the number, which lies to the left of the **hexadecimal point** is known as integer part. Similarly, the part of the number, which lies to the right of the Hexa-decimal point is known as fractional part.

In this number system, the successive positions to the left of the Hexa-decimal point having weights of 16^0, 16^1, 16^2, 16^3 and so on. Similarly, the successive positions to the right of the Hexa-decimal point having weights of 16^{-1}, 16^{-2}, 16^{-3} and so on. That means, each position has specific weight, which is **power of base 16**.

Example

Consider the **Hexa-decimal number 1A05.2C4**. Integer part of this number is 1A05 and fractional part of this number is 0.2C4. The digits 5, 0, A and 1 have weights of 16^0, 16^1, 16^2 and 16^3respectively. Similarly, the digits 2, C and 4 have weights of 16^{-1}, 16^{-2} and 16^{-3} respectively.

Mathematically, we can write it as

$1A05.2C4 = (1 \times 16^3) + (10 \times 16^2) + (0 \times 16^1) + (5 \times 16^0) + (2 \times 16^{-1}) + (12 \times 16^{-2}) + (4 \times 16^{-3})$

After simplifying the right hand side terms, we will get a decimal number, which is an equivalent of Hexa-decimal number on left hand side.

In previous chapter, we have seen the four prominent number systems. In this chapter, let us convert the numbers from one number system to the other in order to find the equivalent value.

Decimal Number to other Bases Conversion

If the decimal number contains both integer part and fractional part, then convert both the parts of decimal number into other base individually. Follow these steps for converting the decimal number into its equivalent number of any base 'r'.

- Do **division** of integer part of decimal number and **successive quotients** with base 'r' and note down the remainders till the quotient is zero. Consider the remainders in reverse order to get the integer part of equivalent number of base 'r'. That means, first and last remainders denote the least significant digit and most significant digit respectively.
- Do **multiplication** of fractional part of decimal number and **successive fractions** with base 'r' and note down the carry till the result is zero or the desired number of equivalent digits is obtained. Consider the normal sequence of carry in order to get the fractional part of equivalent number of base 'r'.

Decimal to Binary Conversion

The following two types of operations take place, while converting decimal number into its equivalent binary number.

- Division of integer part and successive quotients with base 2.
- Multiplication of fractional part and successive fractions with base 2.

Example

Consider the **decimal number 58.25**. Here, the integer part is 58 and fractional part is 0.25.

Step 1 − Division of 58 and successive quotients with base 2.

Operation	Quotient	Remainder
58/2	29	**0 (LSB)**
29/2	14	**1**
14/2	7	**0**
7/2	3	**1**
3/2	1	**1**
1/2	0	**1(MSB)**

$$\Rightarrow (58)_{10} = (111010)_2$$

Therefore, the **integer part** of equivalent binary number is **111010**.

Step 2 − Multiplication of 0.25 and successive fractions with base 2.

Operation	Result	Carry
0.25 x 2	0.5	0
0.5 x 2	1.0	1
-	0.0	-

$$\Rightarrow (.25)_{10} = (.01)_2$$

Therefore, the **fractional part** of equivalent binary number is **.01**

$$\Rightarrow (58.25)_{10} = (111010.01)_2$$

Therefore, the **binary equivalent** of decimal number 58.25 is 111010.01.

Decimal to Octal Conversion

The following two types of operations take place, while converting decimal number into its equivalent octal number.

- Division of integer part and successive quotients with base 8.
- Multiplication of fractional part and successive fractions with base 8.

Example

Consider the **decimal number 58.25**. Here, the integer part is 58 and fractional part is 0.25.

Step 1 − Division of 58 and successive quotients with base 8.

Operation	Quotient	Remainder
58/8	7	**2**
7/8	0	**7**

$\Rightarrow (58)_{10} = (72)_8$

Therefore, the **integer part** of equivalent octal number is **72**.

Step 2 − Multiplication of 0.25 and successive fractions with base 8.

Operation	Result	Carry
0.25 x 8	2.00	2
-	0.00	-

$\Rightarrow (.25)_{10} = (.2)_8$

Therefore, the **fractional part** of equivalent octal number is .2

$\Rightarrow (58.25)_{10} = (72.2)_8$

Therefore, the **octal equivalent** of decimal number 58.25 is 72.2.

Decimal to Hexa-Decimal Conversion

The following two types of operations take place, while converting decimal number into its equivalent hexa-decimal number.

- Division of integer part and successive quotients with base 16.

- Multiplication of fractional part and successive fractions with base 16.

Example

Consider the **decimal number 58.25**. Here, the integer part is 58 and decimal part is 0.25.

Step 1 − Division of 58 and successive quotients with base 16.

Operation	Quotient	Remainder
58/16	3	10=A
3/16	0	3

$\Rightarrow (58)_{10} = (3A)_{16}$

Therefore, the **integer part** of equivalent Hexa-decimal number is 3A.

Step 2 − Multiplication of 0.25 and successive fractions with base 16.

Operation	Result	Carry
0.25 x 16	4.00	4
-	0.00	-

$\Rightarrow (.25)_{10} = (.4)_{16}$

Therefore, the **fractional part** of equivalent Hexa-decimal number is .4.

$$\mathbf{\Rightarrow (58.25)_{10} = (3A.4)_{16}}$$

Therefore, the **Hexa-decimal equivalent** of decimal number 58.25 is 3A.4.

Binary Number to other Bases Conversion

The process of converting a number from binary to decimal is different to the process of converting a binary number to other bases. Now, let us discuss about the conversion of a binary number to decimal, octal and Hexa-decimal number systems one by one.

Binary to Decimal Conversion

For converting a binary number into its equivalent decimal number, first multiply the bits of binary number with the respective positional weights and then add all those products.

Example

Consider the **binary number 1101.11**.

Mathematically, we can write it as

$(1101.11)_2 = (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2})$

$\Rightarrow (1101.11)_2 = 8 + 4 + 0 + 1 + 0.5 + 0.25 = 13.75$

$\Rightarrow (1101.11)_2 = (13.75)_{10}$

Therefore, the **decimal equivalent** of binary number 1101.11 is 13.75.

Binary to Octal Conversion

We know that the bases of binary and octal number systems are 2 and 8 respectively. Three bits of binary number is equivalent to one octal digit, since $2^3 = 8$.

Follow these two steps for converting a binary number into its equivalent octal number.

- Start from the binary point and make the groups of 3 bits on both sides of binary point. If one or two bits are less while making the group of 3 bits, then include required number of zeros on extreme sides.
- Write the octal digits corresponding to each group of 3 bits.

Example

Consider the **binary number 101110.01101**.

Step 1 − Make the groups of 3 bits on both sides of binary point.

101 110.011 01

Here, on right side of binary point, the last group is having only 2 bits. So, include one zero on extreme side in order to make it as group of 3 bits.

$\Rightarrow$ 101 110.011 010

Step 2 − Write the octal digits corresponding to each group of 3 bits.

$\Rightarrow (101\ 110.011\ 010)_2 = (56.32)_8$

Therefore, the **octal equivalent** of binary number 101110.01101 is 56.32.

Binary to Hexa-Decimal Conversion

We know that the bases of binary and Hexa-decimal number systems are 2 and 16 respectively. Four bits of binary number is equivalent to one Hexa-decimal digit, since $2^4 = 16$.

Follow these two steps for converting a binary number into its equivalent Hexa-decimal number.

- Start from the binary point and make the groups of 4 bits on both sides of binary point. If some bits are less while making the group of 4 bits, then include required number of zeros on extreme sides.
- Write the Hexa-decimal digits corresponding to each group of 4 bits.

Example

Consider the **binary number 101110.01101**

Step 1 − Make the groups of 4 bits on both sides of binary point.

10 1110.0110 1

Here, the first group is having only 2 bits. So, include two zeros on extreme side in order to make it as group of 4 bits. Similarly, include three zeros on extreme side in order to make the last group also as group of 4 bits.

⇒ 0010 1110.0110 1000

Step 2 − Write the Hexa-decimal digits corresponding to each group of 4 bits.

$$\Rightarrow (0010\ 1110.0110\ 1000)_2 = (2E.68)_{16}$$

Therefore, the **Hexa-decimal equivalent** of binary number 101110.01101 is (2E.68).

Octal Number to other Bases Conversion

The process of converting a number from octal to decimal is different to the process of converting an octal number to other bases. Now, let us discuss about the conversion of an octal number to decimal, binary and Hexa-decimal number systems one by one.

Octal to Decimal Conversion

For converting an octal number into its equivalent decimal number, first multiply the digits of octal number with the respective positional weights and then add all those products.

Example

Consider the **octal number 145.23**.

Mathematically, we can write it as

$(145.23)_8 = (1 \times 8^2) + (4 \times 8^1) + (5 \times 8^0) + (2 \times 8^{-1}) + (3 \times 8^{-2})$

$\Rightarrow (145.23)_8 = 64 + 32 + 5 + 0.25 + 0.05 = 101.3$

$\Rightarrow (145.23)_8 = (101.3)_{10}$

Therefore, the **decimal equivalent** of octal number 145.23 is 101.3.

Octal to Binary Conversion

The process of converting an octal number to an equivalent binary number is just opposite to that of binary to octal conversion. By representing each octal digit with 3 bits, we will get the equivalent binary number.

Example

Consider the **octal number 145.23**.

Represent each octal digit with 3 bits.

$$(145.23)_8 = (001\ 100\ 101.010\ 011)_2$$

The value doesn't change by removing the zeros, which are on the extreme side.

$$\Rightarrow (145.23)_8 = (1100101.010011)_2$$

Therefore, the **binary equivalent** of octal number 145.23 is 1100101.010011.

Octal to Hexa-Decimal Conversion

Follow these two steps for converting an octal number into its equivalent Hexa-decimal number.

- Convert octal number into its equivalent binary number.
- Convert the above binary number into its equivalent Hexa-decimal number.

Example

Consider the **octal number 145.23**

In previous example, we got the binary equivalent of octal number 145.23 as 1100101.010011.

By following the procedure of binary to Hexa-decimal conversion, we will get

$$(1100101.010011)_2 = (65.4C)16$$

$$\Rightarrow (145.23)_8 = (65.4C)_{16}$$

Therefore, the **Hexa-decimal equivalent** of octal number 145.23 is 65.4*C*.

Hexa-Decimal Number to other Bases Conversion

The process of converting a number from Hexa-decimal to decimal is different to the process of converting Hexa-decimal number into other bases. Now, let us discuss about the conversion of Hexa-decimal number to decimal, binary and octal number systems one by one.

Hexa-Decimal to Decimal Conversion

For converting Hexa-decimal number into its equivalent decimal number, first multiply the digits of Hexa-decimal number with the respective positional weights and then add all those products.

Example

Consider the **Hexa-decimal number 1A5.2**

Mathematically, we can write it as

$$(1A5.2)_{16} = (1 \times 16^2) + (10 \times 16^1) + (5 \times 16^0) + (2 \times 16^{-1})$$

$$\Rightarrow (1A5.2)_{16} = 256 + 160 + 5 + 0.125 = 421.125$$

$$\Rightarrow (1A5.2)_{16} = (421.125)_{10}$$

Therefore, the **decimal equivalent** of Hexa-decimal number 1A5.2 is 421.125.

Hexa-Decimal to Binary Conversion

The process of converting Hexa-decimal number into its equivalent binary number is just opposite to that of binary to Hexa-decimal conversion. By representing each Hexa-decimal digit with 4 bits, we will get the equivalent binary number.

Example

Consider the **Hexa-decimal number 65.4C**

Represent each Hexa-decimal digit with 4 bits.

$$(65.4C)_6 = (0110\ 0101.0100\ 1100)_2$$

The value doesn't change by removing the zeros, which are at two extreme sides.

$\Rightarrow (65.4C)_{16} = (1100101.010011)_2$

Therefore, the **binary equivalent** of Hexa-decimal number 65.4C is 1100101.010011.

Hexa-Decimal to Octal Conversion

Follow these two steps for converting Hexa-decimal number into its equivalent octal number.

- Convert Hexa-decimal number into its equivalent binary number.
- Convert the above binary number into its equivalent octal number.

Example

Consider the **Hexa-decimal number 65.4C**

In previous example, we got the binary equivalent of Hexa-decimal number 65.4C as 1100101.010011.

By following the procedure of binary to octal conversion, we will get

$$(1100101.010011)_2 = (145.23)_8$$

$$\Rightarrow (65.4C)_{16} = (145.23)_8$$

Therefore, the **octal equivalent** of Hexa-decimal number 65.4*C*is 145.23.

We can make the binary numbers into the following two groups − **Unsigned numbers** and **Signed numbers**.

Unsigned Numbers

Unsigned numbers contain only magnitude of the number. They don't have any sign. That means all unsigned binary numbers are positive. As in decimal number system, the placing of positive sign in front of the number is optional for representing positive numbers. Therefore, all positive numbers including zero can be treated as unsigned numbers if positive sign is not assigned in front of the number.

Signed Numbers

Signed numbers contain both sign and magnitude of the number. Generally, the sign is placed in front of number. So, we have to consider the positive sign for positive numbers and negative sign for negative numbers. Therefore, all numbers can be treated as signed numbers if the corresponding sign is assigned in front of the number.

If sign bit is zero, which indicates the binary number is positive. Similarly, if sign bit is one, which indicates the binary number is negative.

Representation of Un-Signed Binary Numbers

The bits present in the un-signed binary number holds the **magnitude** of a number. That means, if the un-signed binary number contains **'N'** bits, then all **N** bits represent the magnitude of the number, since it doesn't have any sign bit.

Example

Consider the **decimal number 108**. The binary equivalent of this number is **1101100**. This is the representation of unsigned binary number.

$$(108)_{10} = (1101100)_2$$

It is having 7 bits. These 7 bits represent the magnitude of the number 108.

Representation of Signed Binary Numbers

The Most Significant Bit (MSB) of signed binary numbers is used to indicate the sign of the numbers. Hence, it is also called as **sign bit**. The positive sign is represented by placing '0' in the sign bit. Similarly, the negative sign is represented by placing '1' in the sign bit.

If the signed binary number contains 'N' bits, then (N-1) bits only represent the magnitude of the number since one bit (MSB) is reserved for representing sign of the number.

There are three **types of representations** for signed binary numbers

- Sign-Magnitude form
- 1's complement form
- 2's complement form

Representation of a positive number in all these 3 forms is same. But, only the representation of negative number will differ in each form.

Example

Consider the **positive decimal number +108**. The binary equivalent of magnitude of this number is 1101100. These 7 bits represent the magnitude of the number 108. Since it is positive number, consider the sign bit as zero, which is placed on left most side of magnitude.

$$(+108)_{10} = (01101100)_2$$

Therefore, the **signed binary representation** of positive decimal number +108 is $\mathbf{01101100}$. So, the same representation is valid in sign-magnitude form, 1's complement form and 2's complement form for positive decimal number +108.

Sign-Magnitude form

In sign-magnitude form, the MSB is used for representing **sign**of the number and the remaining bits represent the **magnitude**of the number. So, just include sign bit at the left most side of unsigned binary number. This representation is similar to the signed decimal numbers representation.

Example

Consider the **negative decimal number -108**. The magnitude of this number is 108. We know the unsigned binary representation of 108 is 1101100. It is having 7 bits. All these bits represent the magnitude.

Since the given number is negative, consider the sign bit as one, which is placed on left most side of magnitude.

$$(-108)_{10} = (11101100)_2$$

Therefore, the sign-magnitude representation of -108 is **11101100**.

1's complement form

The 1's complement of a number is obtained by **complementing all the bits** of signed binary number. So, 1's complement of positive number gives a negative number. Similarly, 1's complement of negative number gives a positive number.

That means, if you perform two times 1's complement of a binary number including sign bit, then you will get the original signed binary number.

Example

Consider the **negative decimal number -108**. The magnitude of this number is 108. We know the signed binary representation of 108 is 01101100.

It is having 8 bits. The MSB of this number is zero, which indicates positive number. Complement of zero is one and vice-versa. So, replace zeros by ones and ones by zeros in order to get the negative number.

$$(-108)_{10} = (10010011)_2$$

Therefore, the **1's complement of $(108)_{10}$** is **$(10010011)_2$**.

2's complement form

The 2's complement of a binary number is obtained by **adding one to the 1's complement** of signed binary number. So, 2's complement of positive number gives a negative number. Similarly, 2's complement of negative number gives a positive number.

That means, if you perform two times 2's complement of a binary number including sign bit, then you will get the original signed binary number.

Example

Consider the **negative decimal number -108**.

We know the 1's complement of $(\mathbf{108})_{\mathbf{10}}$ is $(\mathbf{10010011})_{\mathbf{2}}$

2's compliment of $(108)_{10}$ = 1's compliment of $(108)_{10}$ + 1.

= 10010011 + 1

= 10010100

Therefore, the **2's complement of $(108)_{10}$** is **$(10010100)_2$**.

In this chapter, let us discuss about the basic arithmetic operations, which can be performed on any two signed binary numbers using 2's complement method. The **basic arithmetic operations** are addition and subtraction.

Addition of two Signed Binary Numbers

Consider the two signed binary numbers A & B, which are represented in 2's complement form. We can perform the **addition** of these two numbers, which is similar to the addition of two unsigned binary numbers. But, if the resultant sum contains carry out from sign bit, then discard (ignore) it in order to get the correct value.

If resultant sum is positive, you can find the magnitude of it directly. But, if the resultant sum is negative, then take 2's complement of it in order to get the magnitude.

Example 1

Let us perform the **addition** of two decimal numbers **+7 and +4** using 2's complement method.

The **2's complement** representations of +7 and +4 with 5 bits each are shown below.

$$(+7)_{10} = (00111)_2$$

$$(+4)_{10} = (00100)_2$$

The addition of these two numbers is

$$(+7)_{10} + (+4)_{10} = (00111)_2 + (00100)_2$$

$$\Rightarrow (+7)_{10} + (+4)_{10} = (01011)_2.$$

The resultant sum contains 5 bits. So, there is no carry out from sign bit. The sign bit '0' indicates that the resultant sum is **positive**. So, the magnitude of sum is 11 in decimal number system. Therefore, addition of two positive numbers will give another positive number.

Example 2

Let us perform the **addition** of two decimal numbers **-7** and **-4**using 2's complement method.

The **2's complement** representation of -7 and -4 with 5 bits each are shown below.

$$(-7)_{10} = (11001)_2$$

$$(-4)_{10} = (11100)_2$$

The addition of these two numbers is

$$(-7)_{10} + (-4)_{10} = (11001)_2 + (11100)_2$$

$$\Rightarrow (-7)_{10} + (-4)_{10} = (110101)_2.$$

The resultant sum contains 6 bits. In this case, carry is obtained from sign bit. So, we can remove it

Resultant sum after removing carry is $(-7)_{10} + (-4)_{10} = \mathbf{(10101)_2}$.

The sign bit '1' indicates that the resultant sum is **negative**. So, by taking 2's complement of it we will get the magnitude of resultant sum as 11 in decimal number system. Therefore, addition of two negative numbers will give another negative number.

Subtraction of two Signed Binary Numbers

Consider the two signed binary numbers A & B, which are represented in 2's complement form. We know that 2's complement of positive number gives a negative number. So, whenever we have to subtract a number B from number A, then take 2's complement of B and add it to A. So, **mathematically**we can write it as

A - B = A + ***(2's complement of B)***

Similarly, if we have to subtract the number A from number B, then take 2's complement of A and add it to B. So, **mathematically** we can write it as

B - A = B + ***(2's complement of A)***

So, the subtraction of two signed binary numbers is similar to the addition of two signed binary numbers. But, we have to take 2's complement of the number, which is supposed to be subtracted. This is the **advantage** of 2's complement technique. Follow, the same rules of addition of two signed binary numbers.

Example 3

Let us perform the **subtraction** of two decimal numbers **+7 and +4** using 2's complement method.

The subtraction of these two numbers is

$$(+7)_{10} - (+4)_{10} = (+7)_{10} + (-4)_{10}.$$

The **2's complement** representation of +7 and -4 with 5 bits each are shown below.

$$(+7)_{10} = (00111)_2$$

$$(+4)_{10} = (11100)_2$$

$\Rightarrow (+7)_{10} + (+4)_{10} = (00111)_2 + (11100)_2 = (00011)_2$

Here, the carry obtained from sign bit. So, we can remove it. The resultant sum after removing carry is

$(+7)_{10} + (+4)_{10} = \mathbf{(00011)_2}$

The sign bit '0' indicates that the resultant sum is **positive**. So, the magnitude of it is 3 in decimal number system. Therefore, subtraction of two decimal numbers +7 and +4 is +3.

Example 4

Let us perform the **subtraction of** two decimal numbers **+4** and **+7** using 2's complement method.

The subtraction of these two numbers is

$$(+4)_{10} - (+7)_{10} = (+4)_{10} + (-7)_{10}.$$

The **2's complement** representation of +4 and -7 with 5 bits each are shown below.

$$(+4)_{10} = (00100)_2$$

$$(-7)_{10} = (11001)_2$$

$\Rightarrow (+4)_{10} + (-7)_{10} = (00100)_2 + (11001)_2 = (11101)_2$

Here, carry is not obtained from sign bit. The sign bit '1' indicates that the resultant sum is **negative**. So, by taking 2's complement of it we will get the magnitude of resultant sum as 3 in decimal number system. Therefore, subtraction of two decimal numbers +4 and +7 is -3.

In the coding, when numbers or letters are represented by a specific group of symbols, it is said to be that number or letter is being encoded. The group of symbols is called as **code**. The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as **binary code**.

Binary codes can be classified into two types.

- Weighted codes
- Unweighted codes

If the code has positional weights, then it is said to be **weighted code**. Otherwise, it is an unweighted code. Weighted codes can be further classified as positively weighted codes and negatively weighted codes.

Binary Codes for Decimal digits

The following table shows the various binary codes for decimal digits 0 to 9.

Decimal Digit	8421 Code	2421 Code	84-2-1 Code	Excess 3 Code
0	0000	0000	0000	0011
1	0001	0001	0111	0100
2	0010	0010	0110	0101
3	0011	0011	0101	0110
4	0100	0100	0100	0111
5	0101	1011	1011	1000
6	0110	1100	1010	1001
7	0111	1101	1001	1010
8	1000	1110	1000	1011
9	1001	1111	1111	1100

We have 10 digits in decimal number system. To represent these 10 digits in binary, we require minimum of 4 bits. But, with 4 bits there will be 16 unique combinations of zeros and ones. Since, we have only 10 decimal digits, the other 6 combinations of zeros and ones are not required.

8 4 2 1 code

- The weights of this code are 8, 4, 2 and 1.
- This code has all positive weights. So, it is a **positively weighted code**.
- This code is also called as **natural BCD** (Binary Coded Decimal) **code**.

Example

Let us find the BCD equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the BCD (8421) codes of 7, 8 and 6 are 0111, 1000 and 0110 respectively.

$$\therefore (786)_{10} = (011110000110)_{BCD}$$

There are 12 bits in BCD representation, since each BCD code of decimal digit has 4 bits.

2 4 2 1 code

- The weights of this code are 2, 4, 2 and 1.
- This code has all positive weights. So, it is a **positively weighted code**.
- It is an **unnatural BCD** code. Sum of weights of unnatural BCD codes is equal to 9.
- It is a **self-complementing** code. Self-complementing codes provide the 9's complement of a decimal number, just by interchanging 1's and 0's in its equivalent 2421 representation.

Example

Let us find the 2421 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 2421 codes of 7, 8 and 6 are 1101, 1110 and 1100 respectively.

Therefore, the 2421 equivalent of the decimal number 786 is **110111101100**.

8 4 -2 -1 code

- The weights of this code are 8, 4, -2 and -1.
- This code has negative weights along with positive weights. So, it is a **negatively weighted code**.
- It is an **unnatural BCD** code.
- It is a **self-complementing** code.

Example

Let us find the 8 4-2-1 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the 8 4 -2 -1 codes of 7, 8 and 6 are 1001, 1000 and 1010 respectively.

Therefore, the 8 4 -2 -1 equivalent of the decimal number 786 is **100110001010**.

Excess 3 code

- This code doesn't have any weights. So, it is an **un-weighted code**.
- We will get the Excess 3 code of a decimal number by adding three (0011) to the binary equivalent of that decimal number. Hence, it is called as Excess 3 code.
- It is a **self-complementing** code.

Example

Let us find the Excess 3 equivalent of the decimal number 786. This number has 3 decimal digits 7, 8 and 6. From the table, we can write the Excess 3 codes of 7, 8 and 6 are 1010, 1011 and 1001 respectively.

Therefore, the Excess 3 equivalent of the decimal number 786 is **101010111001**

Gray Code

The following table shows the 4-bit Gray codes corresponding to each 4-bit binary code.

Decimal Number	Binary Code	Gray Code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101

7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

- This code doesn't have any weights. So, it is an **un-weighted code**.
- In the above table, the successive Gray codes are differed in one bit position only. Hence, this code is called as **unit distance** code.

Binary code to Gray Code Conversion

Follow these steps for converting a binary code into its equivalent Gray code.

- Consider the given binary code and place a zero to the left of MSB.
- Compare the successive two bits starting from zero. If the 2 bits are same, then the output is zero. Otherwise, output is one.
- Repeat the above step till the LSB of Gray code is obtained.

Example

From the table, we know that the Gray code corresponding to binary code 1000 is 1100. Now, let us verify it by using the above procedure.

Given, binary code is 1000.

Step 1 − By placing zero to the left of MSB, the binary code will be 01000.

Step 2 − By comparing successive two bits of new binary code, we will get the gray code as **1100**.

Boolean Algebra is an algebra, which deals with binary numbers & binary variables. Hence, it is also called as Binary Algebra or logical Algebra. A mathematician, named George Boole had developed this algebra in 1854. The variables used in this algebra are also called as Boolean variables.

The range of voltages corresponding to Logic 'High' is represented with '1' and the range of voltages corresponding to logic 'Low' is represented with '0'.

Postulates and Basic Laws of Boolean Algebra

In this section, let us discuss about the Boolean postulates and basic laws that are used in Boolean algebra. These are useful in minimizing Boolean functions.

Boolean Postulates

Consider the binary numbers 0 and 1, Boolean variable (x) and its complement (x'). Either the Boolean variable or complement of it is known as **literal**. The four possible **logical OR** operations among these literals and binary numbers are shown below.

$$x + 0 = x$$

$$x + 1 = 1$$

$$x + x = x$$

$$x + x' = 1$$

Similarly, the four possible **logical AND** operations among those literals and binary numbers are shown below.

$$x.1 = x$$

$$x.0 = 0$$

$$x.x = x$$

$$x.x' = 0$$

These are the simple Boolean postulates. We can verify these postulates easily, by substituting the Boolean variable with '0' or '1'.

Note− The complement of complement of any Boolean variable is equal to the variable itself. i.e., $(x')'=x$.

Basic Laws of Boolean Algebra

Following are the three basic laws of Boolean Algebra.

- Commutative law
- Associative law
- Distributive law

Commutative Law

If any logical operation of two Boolean variables give the same result irrespective of the order of those two variables, then that logical operation is said to be **Commutative**. The logical OR & logical AND operations of two Boolean variables x & y are shown below

$$x + y = y + x$$

$$x.y = y.x$$

The symbol '+' indicates logical OR operation. Similarly, the symbol '.' indicates logical AND operation and it is optional to represent. Commutative law obeys for logical OR & logical AND operations.

Associative Law

If a logical operation of any two Boolean variables is performed first and then the same operation is performed with the remaining variable gives the same result, then that logical operation is said to be **Associative**. The logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

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

$$x.(y.z) = (x.y).z$$

Associative law obeys for logical OR & logical AND operations.

Distributive Law

If any logical operation can be distributed to all the terms present in the Boolean function, then that logical operation is said to be **Distributive**. The distribution of logical OR & logical AND operations of three Boolean variables x, y & z are shown below.

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

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

Distributive law obeys for logical OR and logical AND operations.

These are the Basic laws of Boolean algebra. We can verify these laws easily, by substituting the Boolean variables with '0' or '1'.

Theorems of Boolean Algebra

The following two theorems are used in Boolean algebra.

- Duality theorem
- DeMorgan's theorem

Duality Theorem

This theorem states that the **dual** of the Boolean function is obtained by interchanging the logical AND operator with logical OR operator and zeros with ones. For every Boolean function, there will be a corresponding Dual function.

Let us make the Boolean equations (relations) that we discussed in the section of Boolean postulates and basic laws into two groups. The following table shows these two groups.

Group1	Group2
x + 0 = x	x.1 = x
x + 1 = 1	x.0 = 0
x + x = x	x.x = x
x + x' = 1	x.x' = 0
x + y = y + x	x.y = y.x
x + (y + z) = (x + y) + z	x.(y.z) = (x.y).z
x.(y + z) = x.y + x.z	x + (y.z) = (x + y).(x + z)

In each row, there are two Boolean equations and they are dual to each other. We can verify all these Boolean equations of Group1 and Group2 by using duality theorem.

DeMorgan's Theorem

This theorem is useful in finding the **complement of Boolean function**. It states that the complement of logical OR of at least two Boolean variables is equal to the logical AND of each complemented variable.

DeMorgan's theorem with 2 Boolean variables x and y can be represented as

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

The dual of the above Boolean function is

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

Therefore, the complement of logical AND of two Boolean variables is equal to the logical OR of each complemented variable. Similarly, we can apply DeMorgan's theorem for more than 2 Boolean variables also.

Simplification of Boolean Functions

Till now, we discussed the postulates, basic laws and theorems of Boolean algebra. Now, let us simplify some Boolean functions.

Example 1

Let us **simplify** the Boolean function, f = p'qr + pq'r + pqr' + pqr

We can simplify this function in two methods.

Method 1

Given Boolean function, f = p'qr + pq'r + pqr' +pqr.

Step 1 − In first and second terms r is common and in third and fourth terms pq is common. So, take the common terms by using **Distributive law**.

$$\Rightarrow f = (p'q + pq')r + pq(r' + r)$$

Step 2 − The terms present in first parenthesis can be simplified to Ex-OR operation. The terms present in second parenthesis can be simplified to '1' using **Boolean postulate**

$$\Rightarrow f = (p \oplus q)r + pq(1)$$

Step 3 − The first term can't be simplified further. But, the second term can be simplified to pq using **Boolean postulate**.

$$\Rightarrow f = (p \oplus q)r + pq$$

Therefore, the simplified Boolean function is **$f = (p \oplus q)r + pq$**

Method 2

Given Boolean function, f = p'qr + pq'r + pqr' + pqr.

Step 1 − Use the **Boolean postulate**, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.

$$\Rightarrow f = p'qr + pq'r + pqr' + pqr + pqr + pqr$$

Step 2 − Use **Distributive law** for 1st and 4th terms, 2nd and 5th terms, 3rd and 6th terms.

$$\Rightarrow f = qr(p' + p) + pr(q' + q) + pq(r' + r)$$

Step 3 − Use **Boolean postulate**, x + x' = 1 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = qr(1) + pr(1) + pq(1)$$

Step 4 − Use **Boolean postulate**, x.1 = x for simplifying the above three terms.

$$\Rightarrow f = qr + pr + pq$$

$$\Rightarrow f = pq + qr + pr$$

Therefore, the simplified Boolean function is **f = pq + qr + pr**.

So, we got two different Boolean functions after simplifying the given Boolean function in each method. Functionally, those two Boolean functions are same. So, based on the requirement, we can choose one of those two Boolean functions.

Example 2

Let us find the **complement** of the Boolean function, f = p'q + pq'.

The complement of Boolean function is f' = (p'q + pq')'.

Step 1 − Use DeMorgan's theorem, (x + y)' = x'.y'.

$$\Rightarrow f' = (p'q)'.(pq')'$$

Step 2 − Use DeMorgan's theorem, (x.y)' = x' + y'

$$\Rightarrow f' = \{(p')' + q'\}.\{p' + (q')'\}$$

Step3 − Use the Boolean postulate, (x')'=x.

$$\Rightarrow f' = \{p + q'\}.\{p' + q\}$$

$$\Rightarrow f' = pp' + pq + p'q' + qq'$$

Step 4 − Use the Boolean postulate, xx'=0.

$$\Rightarrow f = 0 + pq + p'q' + 0$$

$$\Rightarrow f = pq + p'q'$$

Therefore, the **complement** of Boolean function, p'q + pq' is **pq + p'q'**.

We will get four Boolean product terms by combining two variables x and y with logical AND operation. These Boolean product terms are called as **min terms** or **standard product terms**. The min terms are x'y', x'y, xy' and xy.

Similarly, we will get four Boolean sum terms by combining two variables x and y with logical OR operation. These Boolean sum terms are called as **Max terms** or **standard sum terms**. The Max terms are x + y, x + y', x' + y and x' + y'.

The following table shows the representation of min terms and MAX terms for 2 variables.

x	y	Min terms	Max terms
0	0	$m_0=x'y'$	$M_0=x + y$
0	1	$m_1=x'y$	$M_1=x + y'$
1	0	$m_2=xy'$	$M_2=x' + y$
1	1	$m_3=xy$	$M_3=x' + y'$

If the binary variable is '0', then it is represented as complement of variable in min term and as the variable itself in Max term. Similarly, if the binary variable is '1', then it is represented as complement of variable in Max term and as the variable itself in min term.

From the above table, we can easily notice that min terms and Max terms are complement of each other. If there are 'n' Boolean variables, then there will be 2^n min terms and 2^n Max terms.

Canonical SoP and PoS forms

A truth table consists of a set of inputs and output(s). If there are 'n' input variables, then there will be 2^n possible combinations with zeros and ones. So the value of each output variable depends on the combination of input variables. So, each output variable will have '1' for some combination of input variables and '0' for some other combination of input variables.

Therefore, we can express each output variable in following two ways.

- Canonical SoP form
- Canonical PoS form

Canonical SoP form

Canonical SoP form means Canonical Sum of Products form. In this form, each product term contains all literals. So, these product terms are nothing but the min terms. Hence, canonical SoP form is also called as **sum of min terms** form.

First, identify the min terms for which, the output variable is one and then do the logical OR of those min terms in order to get the Boolean expression (function) corresponding to that output variable. This Boolean function will be in the form of sum of min terms.

Follow the same procedure for other output variables also, if there is more than one output variable.

Example

Consider the following **truth table**.

Inputs		**Output**	
p	**q**	**r**	**f**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Here, the output (f) is '1' for four combinations of inputs. The corresponding min terms are p'qr, pq'r, pqr', pqr. By doing logical OR of these four min terms, we will get the Boolean function of output (f).

Therefore, the Boolean function of output is, f = p'qr + pq'r + pqr' + pqr. This is the **canonical SoP form** of output, f. We can also represent this function in following two notations.

$$f=m3+m5+m6+m7f=m3+m5+m6+m7$$

$$f=\sum m(3,5,6,7)f=\sum m(3,5,6,7)$$

In one equation, we represented the function as sum of respective min terms. In other equation, we used the symbol for summation of those min terms.

Canonical PoS form

Canonical PoS form means Canonical Product of Sums form. In this form, each sum term contains all literals. So, these sum terms are nothing but the Max terms. Hence, canonical PoS form is also called as **product of Max terms** form.

First, identify the Max terms for which, the output variable is zero and then do the logical AND of those Max terms in order to get the Boolean expression (function) corresponding to that output variable. This Boolean function will be in the form of product of Max terms.

Follow the same procedure for other output variables also, if there is more than one output variable.

Example

Consider the same truth table of previous example. Here, the output (f) is '0' for four combinations of inputs. The corresponding Max terms are p + q + r, p + q + r', p + q' + r, p' + q + r. By doing logical AND of these four Max terms, we will get the Boolean function of output (f).

Therefore, the Boolean function of output is, f = (p + q + r).(p + q + r').(p + q' + r).(p' + q + r). This is the **canonical PoS form**of output, f. We can also represent this function in following two notations.

$$f=M0.M1.M2.M4f=M0.M1.M2.M4$$

$$f=\prod M(0,1,2,4)f=\prod M(0,1,2,4)$$

In one equation, we represented the function as product of respective Max terms. In other equation, we used the symbol for multiplication of those Max terms.

The Boolean function, f = (p + q + r).(p + q + r').(p + q' + r).(p' + q + r) is the dual of the Boolean function, f = p'qr + pq'r + pqr' + pqr.

Therefore, both canonical SoP and canonical PoS forms are **Dual**to each other. Functionally, these two forms are same. Based on the requirement, we can use one of these two forms.

Standard SoP and PoS forms

We discussed two canonical forms of representing the Boolean output(s). Similarly, there are two standard forms of representing the Boolean output(s). These are the simplified version of canonical forms.

- Standard SoP form
- Standard PoS form

We will discuss about Logic gates in later chapters. The main **advantage** of standard forms is that the number of inputs applied to logic gates can be minimized. Sometimes, there will be reduction in the total number of logic gates required.

Standard SoP form

Standard SoP form means **Standard Sum of Products** form. In this form, each product term need not contain all literals. So, the product terms may or may not be the min terms. Therefore, the Standard SoP form is the simplified form of canonical SoP form.

We will get Standard SoP form of output variable in two steps.

- Get the canonical SoP form of output variable
- Simplify the above Boolean function, which is in canonical SoP form.

Follow the same procedure for other output variables also, if there is more than one output variable. Sometimes, it may not possible to simplify the canonical SoP form. In that case, both canonical and standard SoP forms are same.

Example

Convert the following Boolean function into Standard SoP form.

$$f = p'qr + pq'r + pqr' + pqr$$

The given Boolean function is in canonical SoP form. Now, we have to simplify this Boolean function in order to get standard SoP form.

Step 1 − Use the **Boolean postulate**, x + x = x. That means, the Logical OR operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the last term pqr two more times.

$$\Rightarrow f = p'qr + pq'r + pqr' + pqr + pqr + pqr$$

Step 2 − Use **Distributive law** for 1^{st} and 4^{th} terms, 2^{nd} and 5^{th} terms, 3^{rd} and 6^{th} terms.

$$\Rightarrow f = qr(p' + p) + pr(q' + q) + pq(r' + r)$$

Step 3 − Use **Boolean postulate**, x + x' = 1 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = qr(1) + pr(1) + pq(1)$$

Step 4 − Use **Boolean postulate**, x.1 = x for simplifying above three terms.

$$\Rightarrow f = qr + pr + pq$$

$$\Rightarrow f = pq + qr + pr$$

This is the simplified Boolean function. Therefore, the **standard SoP form** corresponding to given canonical SoP form is **f = pq + qr + pr**

Standard PoS form

Standard PoS form means **Standard Product of Sums** form. In this form, each sum term need not contain all literals. So, the sum terms may or may not be the Max terms. Therefore, the Standard PoS form is the simplified form of canonical PoS form.

We will get Standard PoS form of output variable in two steps.

- Get the canonical PoS form of output variable
- Simplify the above Boolean function, which is in canonical PoS form.

Follow the same procedure for other output variables also, if there is more than one output variable. Sometimes, it may not possible to simplify the canonical PoS form. In that case, both canonical and standard PoS forms are same.

Example

Convert the following Boolean function into Standard PoS form.

$$f = (p + q + r).(p + q + r').(p + q' + r).(p' + q + r)$$

The given Boolean function is in canonical PoS form. Now, we have to simplify this Boolean function in order to get standard PoS form.

Step 1 − Use the **Boolean postulate**, x.x = x. That means, the Logical AND operation with any Boolean variable 'n' times will be equal to the same variable. So, we can write the first term p+q+r two more times.

$$\Rightarrow f = (p + q + r).(p + q + r).(p + q + r).(p + q + r').(p +q' + r).(p' + q + r)$$

Step 2 − Use **Distributive law,** x + (y.z) = (x + y).(x + z) for 1^{st} and 4^{th} parenthesis, 2^{nd} and 5^{th} parenthesis, 3^{rd} and 6^{th}parenthesis.

$$\Rightarrow f = (p + q + rr').(p + r + qq').(q + r + pp')$$

Step 3 − Use **Boolean postulate**, x.x'=0 for simplifying the terms present in each parenthesis.

$$\Rightarrow f = (p + q + 0).(p + r + 0).(q + r + 0)$$

Step 4 − Use **Boolean postulate**, x + 0 = x for simplifying the terms present in each parenthesis

$$\Rightarrow f = (p + q).(p + r).(q + r)$$

$$\Rightarrow f = (p + q).(q + r).(p + r)$$

This is the simplified Boolean function. Therefore, the **standard PoS form** corresponding to given canonical PoS form is **f = (p + q).(q + r).(p + r)**. This is the **dual** of the Boolean function, f = pq + qr + pr.

Therefore, both Standard SoP and Standard PoS forms are Dual to each other.

Digital electronic circuits operate with voltages of **two logic levels**namely Logic Low and Logic High. The range of voltages corresponding to Logic Low is represented with '0'. Similarly, the range of voltages corresponding to Logic High is represented with '1'.

The basic digital electronic circuit that has one or more inputs and single output is known as **Logic gate**. Hence, the Logic gates are the building blocks of any digital system. We can classify these Logic gates into the following three categories.

- Basic gates
- Universal gates
- Special gates

Now, let us discuss about the Logic gates come under each category one by one.

Basic Gates

In earlier chapters, we learnt that the Boolean functions can be represented either in sum of products form or in product of sums form based on the requirement. So, we can implement these Boolean functions by using basic gates. The basic gates are AND, OR & NOT gates.

AND gate

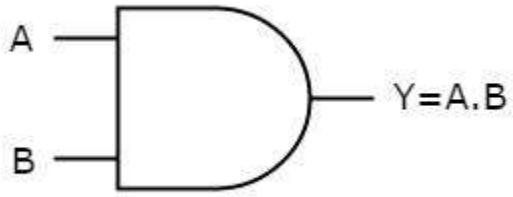
An AND gate is a digital circuit that has two or more inputs and produces an output, which is the **logical AND** of all those inputs. It is optional to represent the **Logical AND** with the symbol '.'.

The following table shows the **truth table** of 2-input AND gate.

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Here A, B are the inputs and Y is the output of two input AND gate. If both inputs are '1', then only the output, Y is '1'. For remaining combinations of inputs, the output, Y is '0'.

The following figure shows the **symbol** of an AND gate, which is having two inputs A, B and one output, Y.



This AND gate produces an output (Y), which is the **logical AND**of two inputs A, B. Similarly, if there are 'n' inputs, then the AND gate produces an output, which is the logical AND of all those inputs. That means, the output of AND gate will be '1', when all the inputs are '1'.

OR gate

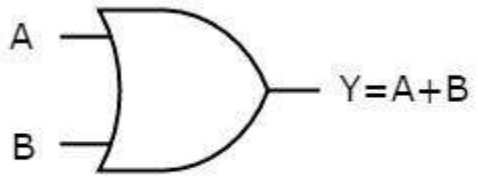
An OR gate is a digital circuit that has two or more inputs and produces an output, which is the logical OR of all those inputs. This **logical OR** is represented with the symbol '+'.

The following table shows the **truth table** of 2-input OR gate.

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Here A, B are the inputs and Y is the output of two input OR gate. If both inputs are '0', then only the output, Y is '0'. For remaining combinations of inputs, the output, Y is '1'.

The following figure shows the **symbol** of an OR gate, which is having two inputs A, B and one output, Y.



This OR gate produces an output (Y), which is the **logical OR** of two inputs A, B. Similarly, if there are 'n' inputs, then the OR gate produces an output, which is the logical OR of all those inputs. That means, the output of an OR gate will be '1', when at least one of those inputs is '1'.

NOT gate

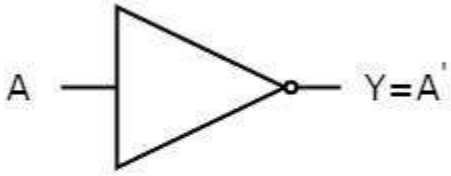
A NOT gate is a digital circuit that has single input and single output. The output of NOT gate is the **logical inversion** of input. Hence, the NOT gate is also called as inverter.

The following table shows the **truth table** of NOT gate.

A	Y = A'
0	1
1	0

Here A and Y are the input and output of NOT gate respectively. If the input, A is '0', then the output, Y is '1'. Similarly, if the input, A is '1', then the output, Y is '0'.

The following figure shows the **symbol** of NOT gate, which is having one input, A and one output, Y.



This NOT gate produces an output (Y), which is the **complement** of input, A.

Universal gates

NAND & NOR gates are called as **universal gates**. Because we can implement any Boolean function, which is in sum of products form by using NAND gates alone. Similarly, we can implement any Boolean function, which is in product of sums form by using NOR gates alone.

NAND gate

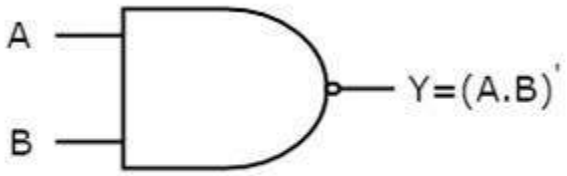
NAND gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical AND** of all those inputs.

The following table shows the **truth table** of 2-input NAND gate.

A	B	Y = (A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input NAND gate. When both inputs are '1', the output, Y is '0'. If at least one of the input is zero, then the output, Y is '1'. This is just opposite to that of two input AND gate operation.

The following image shows the **symbol** of NAND gate, which is having two inputs A, B and one output, Y.



NAND gate operation is same as that of AND gate followed by an inverter. That's why the NAND gate symbol is represented like that.

NOR gate

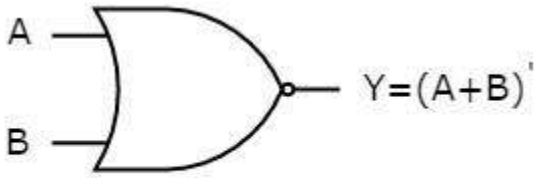
NOR gate is a digital circuit that has two or more inputs and produces an output, which is the **inversion of logical OR** of all those inputs.

The following table shows the **truth table** of 2-input NOR gate

A	B	Y = (A+B)'
0	0	1
0	1	0
1	0	0
1	1	0

Here A, B are the inputs and Y is the output. If both inputs are '0', then the output, Y is '1'. If at least one of the input is '1', then the output, Y is '0'. This is just opposite to that of two input OR gate operation.

The following figure shows the **symbol** of NOR gate, which is having two inputs A, B and one output, Y.



NOR gate operation is same as that of OR gate followed by an inverter. That's why the NOR gate symbol is represented like that.

Special Gates

Ex-OR & Ex-NOR gates are called as special gates. Because, these two gates are special cases of OR & NOR gates.

Ex-OR gate

The full form of Ex-OR gate is **Exclusive-OR** gate. Its function is same as that of OR gate except for some cases, when the inputs having even number of ones.

The following table shows the **truth table** of 2-input Ex-OR gate.

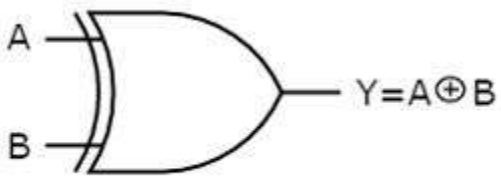
A	B	Y = A⊕B
0	0	0

0	1	1
1	0	1
1	1	0

Here A, B are the inputs and Y is the output of two input Ex-OR gate. The truth table of Ex-OR gate is same as that of OR gate for first three rows. The only modification is in the fourth row. That means, the output (Y) is zero instead of one, when both the inputs are one, since the inputs having even number of ones.

Therefore, the output of Ex-OR gate is '1', when only one of the two inputs is '1'. And it is zero, when both inputs are same.

Below figure shows the **symbol** of Ex-OR gate, which is having two inputs A, B and one output, Y.



Ex-OR gate operation is similar to that of OR gate, except for few combination(s) of inputs. That's why the Ex-OR gate symbol is represented like that. The output of Ex-OR gate is '1', when odd number of ones present at the inputs. Hence, the output of Ex-OR gate is also called as an **odd function**.

Ex-NOR gate

The full form of Ex-NOR gate is **Exclusive-NOR** gate. Its function is same as that of NOR gate except for some cases, when the inputs having even number of ones.

The following table shows the **truth table** of 2-input Ex-NOR gate.

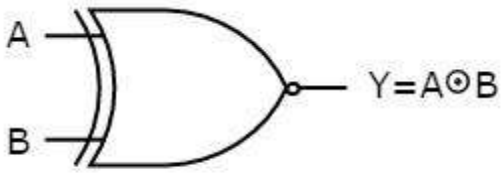
A	B	Y = A⊙B
0	0	1
0	1	0
1	0	0

1	1	1

Here A, B are the inputs and Y is the output. The truth table of Ex-NOR gate is same as that of NOR gate for first three rows. The only modification is in the fourth row. That means, the output is one instead of zero, when both the inputs are one.

Therefore, the output of Ex-NOR gate is '1', when both inputs are same. And it is zero, when both the inputs are different.

The following figure shows the **symbol** of Ex-NOR gate, which is having two inputs A, B and one output, Y.



Ex-NOR gate operation is similar to that of NOR gate, except for few combination(s) of inputs. That's why the Ex-NOR gate symbol is represented like that. The output of Ex-NOR gate is '1', when even number of ones present at the inputs. Hence, the output of Ex-NOR gate is also called as an **even function**.

From the above truth tables of Ex-OR & Ex-NOR logic gates, we can easily notice that the Ex-NOR operation is just the logical inversion of Ex-OR operation.

The maximum number of levels that are present between inputs and output is two in **two level logic**. That means, irrespective of total number of logic gates, the maximum number of Logic gates that are present (cascaded) between any input and output is two in two level logic. Here, the outputs of first level Logic gates are connected as inputs of second level Logic gate(s).

Consider the four Logic gates AND, OR, NAND & NOR. Since, there are 4 Logic gates, we will get 16 possible ways of realizing two level logic. Those are AND-AND, AND-OR, ANDNAND, AND-NOR, OR-AND, OR-OR, OR-NAND, OR-NOR, NAND-AND, NAND-OR, NANDNAND, NAND-NOR, NOR-AND, NOR-OR, NOR-NAND, NOR-NOR.

These two level logic realizations can be classified into the following two categories.

- Degenerative form
- Non-degenerative form

Degenerative Form

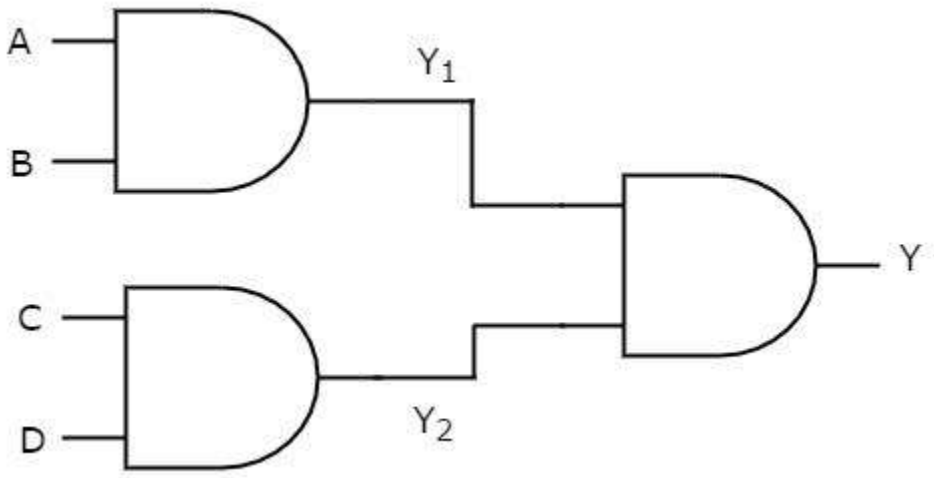
If the output of two level logic realization can be obtained by using single Logic gate, then it is called as **degenerative form**. Obviously, the number of inputs of single Logic gate increases. Due to this, the fan-in of Logic gate increases. This is an advantage of degenerative form.

Only **6 combinations** of two level logic realizations out of 16 combinations come under degenerative form. Those are AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-NOR, NORNAND.

In this section, let us discuss some realizations. Assume, A, B, C & D are the inputs and Y is the output in each logic realization.

AND-AND Logic

In this logic realization, AND gates are present in both levels. Below figure shows an example for **AND-AND logic** realization.



We will get the outputs of first level logic gates as Y1=ABY1=AB and Y2=CDY2=CD

These outputs, Y1Y1 and Y2Y2 are applied as inputs of AND gate that is present in second level. So, the output of this AND gate is

$$Y=Y1Y2Y=Y1Y2$$

Substitute Y1Y1 and Y2Y2 values in the above equation.

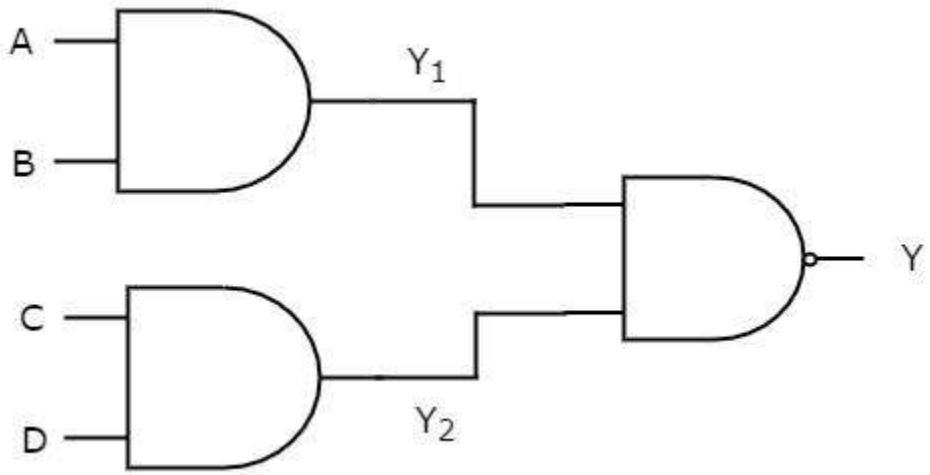
$$Y=(AB)(CD)Y=(AB)(CD)$$

$\Rightarrow Y=ABCD \Rightarrow Y=ABCD$

Therefore, the output of this AND-AND logic realization is **ABCD**. This Boolean function can be implemented by using a 4 input AND gate. Hence, it is **degenerative form**.

AND-NAND Logic

In this logic realization, AND gates are present in first level and NAND gate(s) are present in second level. The following figure shows an example for **AND-NAND logic** realization.



Previously, we got the outputs of first level logic gates as Y1=ABY1=AB and Y2=CDY2=CD

These outputs,Y1Y1 and Y2Y2 are applied as inputs of NAND gate that is present in second level. So, the output of this NAND gate is

$$Y=(Y1Y2)'Y=(Y1Y2)'$$

Substitute Y1Y1 and Y2Y2 values in the above equation.

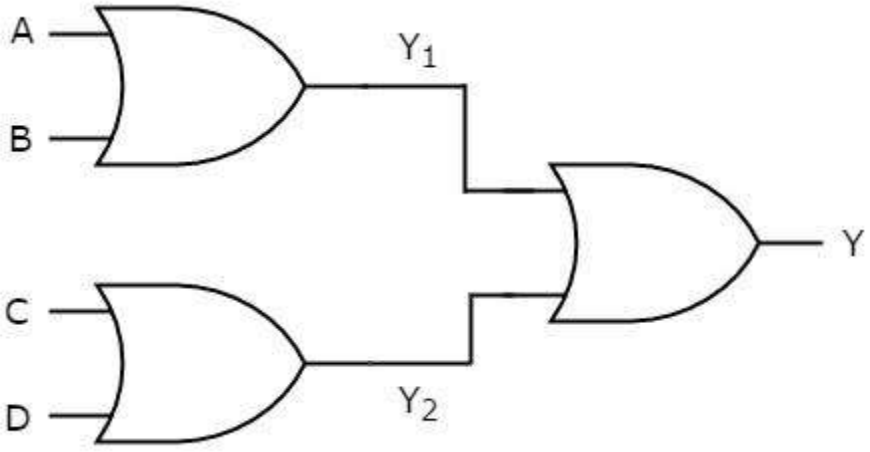
$$Y=((AB)(CD))'Y=((AB)(CD))'$$

$\Rightarrow Y=(ABCD)'\Rightarrow Y=(ABCD)'$

Therefore, the output of this AND-NAND logic realization is (ABCD)'(ABCD)'. This Boolean function can be implemented by using a 4 input NAND gate. Hence, it is **degenerative form**.

OR-OR Logic

In this logic realization, OR gates are present in both levels. The following figure shows an example for **OR-OR logic** realization.



We will get the outputs of first level logic gates as Y1=A+BY1=A+Band Y2=C+DY2=C+D.

These outputs, Y1Y1 and Y2Y2 are applied as inputs of OR gate that is present in second level. So, the output of this OR gate is

$$Y=Y1+Y2Y=Y1+Y2$$

Substitute Y1Y1 and Y2Y2 values in the above equation.

$$Y=(A+B)+(C+D)Y=(A+B)+(C+D)$$

$$\Rightarrow Y=A+B+C+D\Rightarrow Y=A+B+C+D$$

Therefore, the output of this OR-OR logic realization is **A+B+C+D**. This Boolean function can be implemented by using a 4 input OR gate. Hence, it is **degenerative form**.

Similarly, you can verify whether the remaining realizations belong to this category or not.

Non-degenerative Form

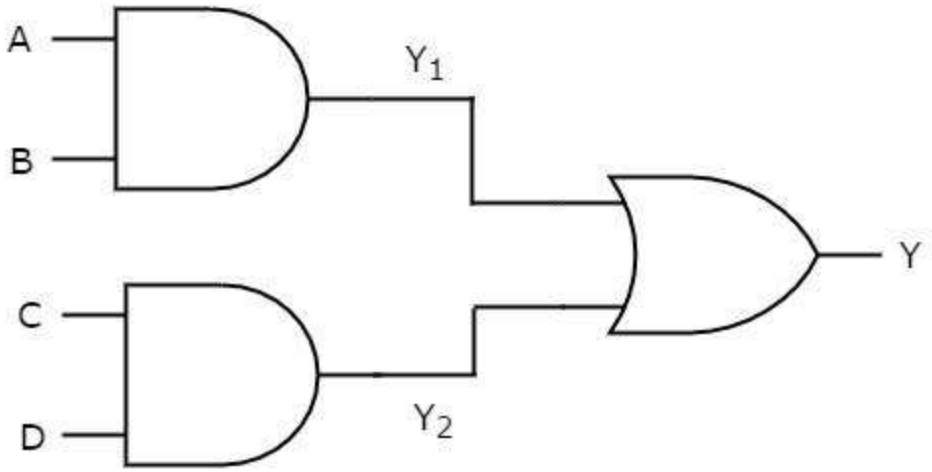
If the output of two level logic realization can't be obtained by using single logic gate, then it is called as **non-degenerative form**.

The remaining **10 combinations** of two level logic realizations come under nondegenerative form. Those are AND-OR, AND-NOR, OR-AND, OR-NAND, NAND-AND, NANDOR, NAND-NAND, NOR-AND, NOR-OR, NOR-NOR.

Now, let us discuss some realizations. Assume, A, B, C & D are the inputs and Y is the output in each logic realization.

AND-OR Logic

In this logic realization, AND gates are present in first level and OR gate(s) are present in second level. Below figure shows an example for **AND-OR logic** realization.



Previously, we got the outputs of first level logic gates as Y1=ABY1=AB and Y2=CDY2=CD.

These outputs, Y1 and Y2 are applied as inputs of OR gate that is present in second level. So, the output of this OR gate is

Y=Y1+Y2Y=Y1+Y2

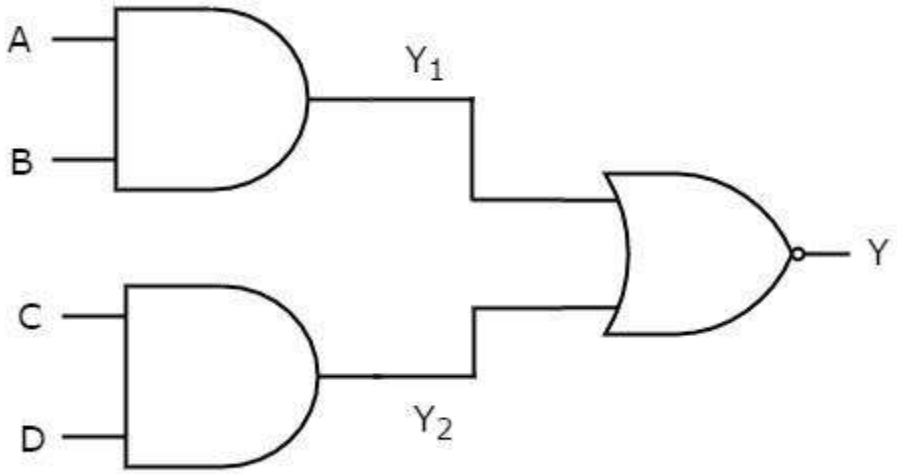
Substitute Y1Y1 and Y2Y2 values in the above equation

Y=AB+CDY=AB+CD

Therefore, the output of this AND-OR logic realization is **AB+CD**. This Boolean function is in **Sum of Products** form. Since, we can't implement it by using single logic gate, this AND-OR logic realization is a **non-degenerative form**.

AND-NOR Logic

In this logic realization, AND gates are present in first level and NOR gate(s) are present in second level. The following figure shows an example for **AND-NOR logic** realization.



We know the outputs of first level logic gates as Y1=ABY1=AB and Y2=CDY2=CD

These outputs, Y1 and Y2 are applied as inputs of NOR gate that is present in second level. So, the output of this NOR gate is

Y=(Y1+Y2)'Y=(Y1+Y2)'

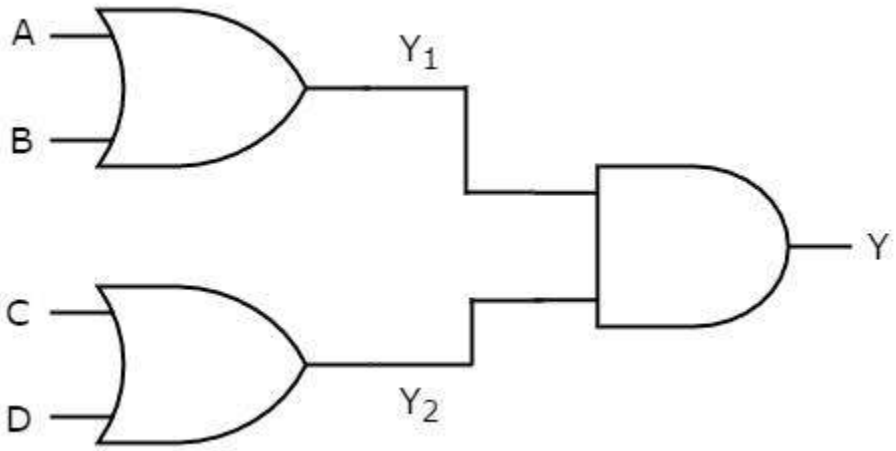
Substitute Y1Y1 and Y2Y2 values in the above equation.

Y=(AB+CD)'Y=(AB+CD)'

Therefore, the output of this AND-NOR logic realization is (AB+CD)'(AB+CD)'. This Boolean function is in **AND-OR-Invert** form. Since, we can't implement it by using single logic gate, this AND-NOR logic realization is a **non-degenerative form**

OR-AND Logic

In this logic realization, OR gates are present in first level & AND gate(s) are present in second level. The following figure shows an example for **OR-AND logic** realization.



Previously, we got the outputs of first level logic gates as Y1=A+BY1=A+B and Y2=C+DY2=C+D.
These outputs, Y1Y1 and Y2Y2 are applied as inputs of AND gate that is present in second level. So, the output of this AND gate is

$$Y=Y1Y2Y=Y1Y2$$

Substitute Y1Y1 and Y2Y2 values in the above equation.

$$Y=(A+B)(C+D)Y=(A+B)(C+D)$$

Therefore, the output of this OR-AND logic realization is **(A + B) (C + D)**. This Boolean function is in **Product of Sums** form. Since, we can't implement it by using single logic gate, this OR-AND logic realization is a **non-degenerative form**.

Similarly, you can verify whether the remaining realizations belong to this category or not.

UNIT-IV

Minimization Techniques

In previous chapters, we have simplified the Boolean functions using Boolean postulates and theorems. It is a time consuming process and we have to re-write the simplified expressions after each step.

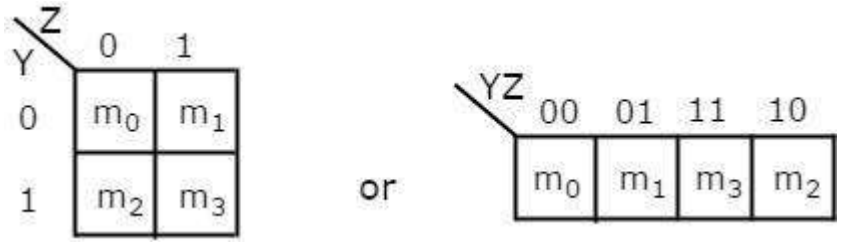
To overcome this difficulty, Karnaugh introduced a method for simplification of Boolean functions in an easy way. This method is known as Karnaugh map method or K-map method. It is a graphical method, which consists of 2^n cells for 'n' variables. The adjacent cells are differed only in single bit position.

K-Maps for 2 to 5 Variables

K-Map method is most suitable for minimizing Boolean functions of 2 variables to 5 variables. Now, let us discuss about the K-Maps for 2 to 5 variables one by one.

2 Variable K-Map

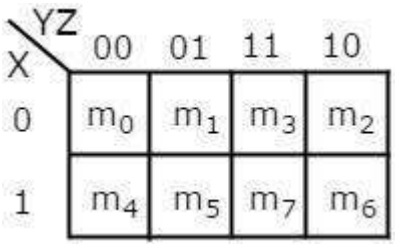
The number of cells in 2 variable K-map is four, since the number of variables is two. The following figure shows **2 variable K-Map**.



- There is only one possibility of grouping 4 adjacent min terms.
- The possible combinations of grouping 2 adjacent min terms are {(m_0, m_1), (m_2, m_3), (m_0, m_2) and (m_1, m_3)}.

3 Variable K-Map

The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows **3 variable K-Map**.



- There is only one possibility of grouping 8 adjacent min terms.
- The possible combinations of grouping 4 adjacent min terms are {(m_0, m_1, m_3, m_2), (m_4, m_5, m_7, m_6), (m_0, m_1, m_4, m_5), (m_1, m_3, m_5, m_7), (m_3, m_2, m_7, m_6) and (m_2, m_0, m_6, m_4)}.
- The possible combinations of grouping 2 adjacent min terms are {(m_0, m_1), (m_1, m_3), (m_3, m_2), (m_2, m_0), (m_4, m_5), (m_5, m_7), (m_7, m_6), (m_6, m_4), (m_0, m_4), (m_1, m_5), (m_3, m_7) and (m_2, m_6)}.
- If x=0, then 3 variable K-map becomes 2 variable K-map.

4 Variable K-Map

The number of cells in 4 variable K-map is sixteen, since the number of variables is four. The following figure shows **4 variable K-Map**.

WX \ YZ	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

- There is only one possibility of grouping 16 adjacent min terms.
- Let R_1, R_2, R_3 and R_4 represents the min terms of first row, second row, third row and fourth row respectively. Similarly, C_1, C_2, C_3 and C_4 represents the min terms of first column, second column, third column and fourth column respectively. The possible combinations of grouping 8 adjacent min terms are {(R_1, R_2), (R_2, R_3), (R_3, R_4), (R_4, R_1), (C_1, C_2), (C_2, C_3), (C_3, C_4), (C_4, C_1)}.
- If w=0, then 4 variable K-map becomes 3 variable K-map.

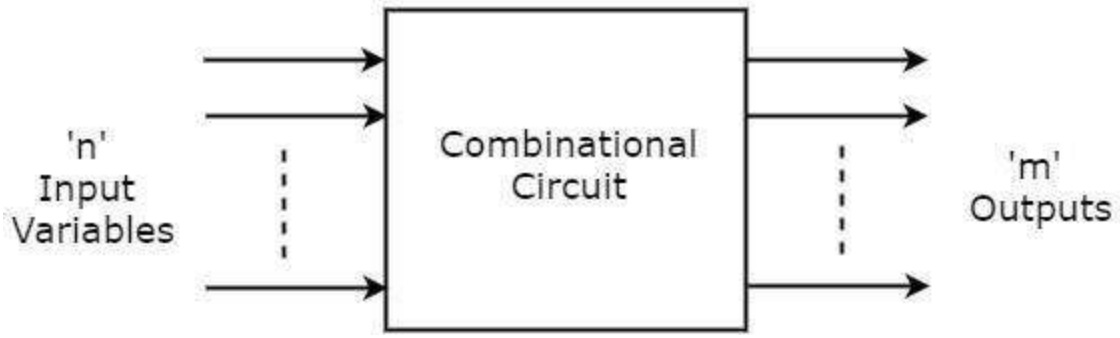
5 Variable K-Map

The number of cells in 5 variable K-map is thirty-two, since the number of variables is 5. The following figure shows **5 variable K-Map**.

UNIT 5

Combinational Circuits

Combinational circuits consist of Logic gates. These circuits operate with binary values. The output(s) of combinational circuit depends on the combination of present inputs. The following figure shows the **block diagram** of combinational circuit.



This combinational circuit has 'n' input variables and 'm' outputs. Each combination of input variables will affect the output(s).

Design procedure of Combinational circuits

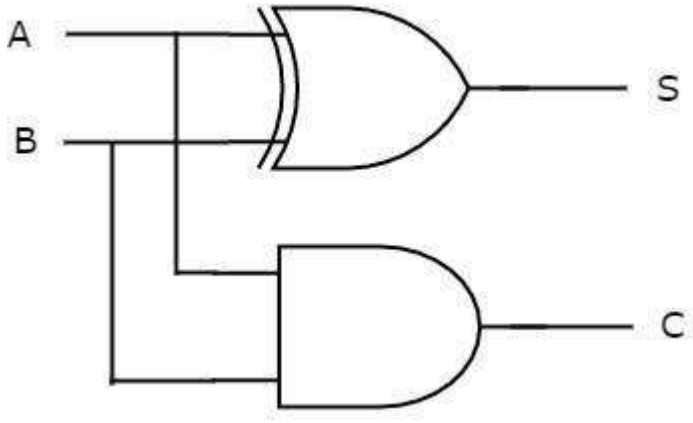
- Find the required number of input variables and outputs from given specifications.
- Formulate the **Truth table**. If there are 'n' input variables, then there will be 2n possible combinations. For each combination of input, find the output values.
- Find the **Boolean expressions** for each output. If necessary, simplify those expressions.
- Implement the above Boolean expressions corresponding to each output by using **Logic gates**.
- In this chapter, let us discuss about the basic arithmetic circuits like Binary adder and Binary subtractor. These circuits can be operated with binary values 0 and 1.
- Binary Adder
- The most basic arithmetic operation is addition. The circuit, which performs the addition of two binary numbers is known as **Binary adder**. First, let us implement an adder, which performs the addition of two bits.

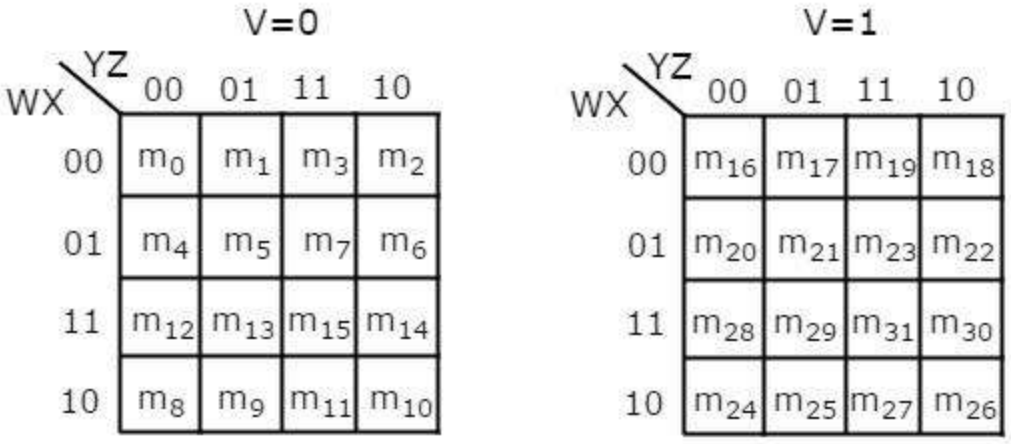
Half Adder

- Half adder is a combinational circuit, which performs the addition of two binary numbers A and B are of **single bit**. It produces two outputs sum, S & carry, C.
- The **Truth table** of Half adder is shown below.

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- When we do the addition of two bits, the resultant sum can have the values ranging from 0 to 2 in decimal. We can represent the decimal digits 0 and 1 with single bit in binary. But, we can't represent decimal digit 2 with single bit in binary. So, we require two bits for representing it in binary.
- Let, sum, S is the Least significant bit and carry, C is the Most significant bit of the resultant sum. For first three combinations of inputs, carry, C is zero and the value of S will be either zero or one based on the **number of ones** present at the inputs. But, for last combination of inputs, carry, C is one and sum, S is zero, since the resultant sum is two.
- From Truth table, we can directly write the **Boolean functions**for each output as
    - $S=A\oplus BS=A\oplus B$
- $C=ABC=AB$
- We can implement the above functions with 2-input Ex-OR gate & 2-input AND gate. The **circuit diagram** of Half adder is shown in the following figure.





- There is only one possibility of grouping 32 adjacent min terms.
- There are two possibilities of grouping 16 adjacent min terms. i.e., grouping of min terms from m_0 to m_{15} and m_{16} to m_{31}.
- If v=0, then 5 variable K-map becomes 4 variable K-map.

In the above all K-maps, we used exclusively the min terms notation. Similarly, you can use exclusively the Max terms notation.

Minimization of Boolean Functions using K-Maps

If we consider the combination of inputs for which the Boolean function is '1', then we will get the Boolean function, which is in **standard sum of products** form after simplifying the K-map.

Similarly, if we consider the combination of inputs for which the Boolean function is '0', then we will get the Boolean function, which is in **standard product of sums** form after simplifying the K-map.

Follow these **rules for simplifying K-maps** in order to get standard sum of products form.

- Select the respective K-map based on the number of variables present in the Boolean function.
- If the Boolean function is given as sum of min terms form, then place the ones at respective min term cells in the K-map. If the Boolean function is given as sum of products form, then place the ones in all possible cells of K-map for which the given product terms are valid.
- Check for the possibilities of grouping maximum number of adjacent ones. It should be powers of two. Start from highest power of two and upto least power of two. Highest power is equal to the number of variables considered in K-map and least power is zero.
- Each grouping will give either a literal or one product term. It is known as **prime implicant**. The prime implicant is said to be **essential prime implicant**, if atleast single '1' is not covered with any other groupings but only that grouping covers.

- Note down all the prime implicants and essential prime implicants. The simplified Boolean function contains all essential prime implicants and only the required prime implicants.

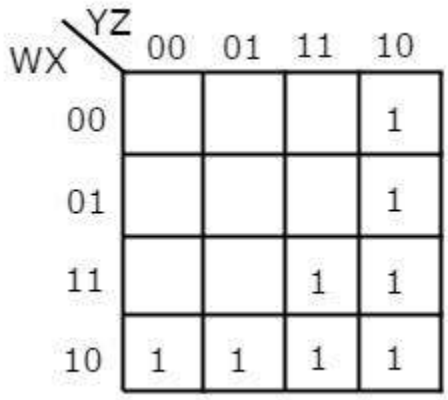
Note 1 − If outputs are not defined for some combination of inputs, then those output values will be represented with **don't care symbol 'x'**. That means, we can consider them as either '0' or '1'.

Note 2 − If don't care terms also present, then place don't cares 'x' in the respective cells of K-map. Consider only the don't cares 'x' that are helpful for grouping maximum number of adjacent ones. In those cases, treat the don't care value as '1'.

Example

Let us **simplify** the following Boolean function, **f(W, X, Y, Z)= WX'Y' + WY + W'YZ'** using K-map.

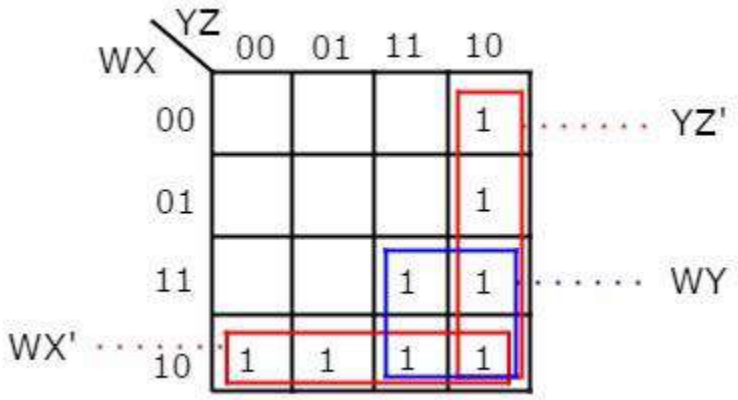
The given Boolean function is in sum of products form. It is having 4 variables W, X, Y & Z. So, we require **4 variable K-map**. The **4 variable K-map** with ones corresponding to the given product terms is shown in the following figure.



Here, 1s are placed in the following cells of K-map.

- The cells, which are common to the intersection of Row 4 and columns 1 & 2 are corresponding to the product term, **WX'Y'**.
- The cells, which are common to the intersection of Rows 3 & 4 and columns 3 & 4 are corresponding to the product term, **WY**.
- The cells, which are common to the intersection of Rows 1 & 2 and column 4 are corresponding to the product term, **W'YZ'**.

There are no possibilities of grouping either 16 adjacent ones or 8 adjacent ones. There are three possibilities of grouping 4 adjacent ones. After these three groupings, there is no single one left as ungrouped. So, we no need to check for grouping of 2 adjacent ones. The **4 variable K-map** with these three **groupings** is shown in the following figure.



Here, we got three prime implicants WX', WY & YZ'. All these prime implicants are **essential** because of following reasons.

- Two ones **(m_8 & m_9)** of fourth row grouping are not covered by any other groupings. Only fourth row grouping covers those two ones.
- Single one **(m_{15})** of square shape grouping is not covered by any other groupings. Only the square shape grouping covers that one.
- Two ones **(m_2 & m_6)** of fourth column grouping are not covered by any other groupings. Only fourth column grouping covers those two ones.

Therefore, the **simplified Boolean function** is

f = WX' + WY + YZ'

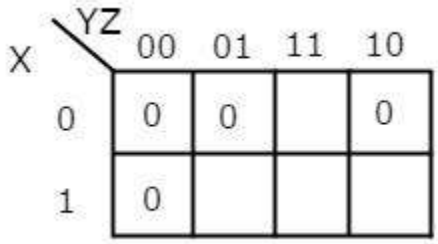
Follow these **rules for simplifying K-maps** in order to get standard product of sums form.

- Select the respective K-map based on the number of variables present in the Boolean function.
- If the Boolean function is given as product of Max terms form, then place the zeroes at respective Max term cells in the K-map. If the Boolean function is given as product of sums form, then place the zeroes in all possible cells of K-map for which the given sum terms are valid.
- Check for the possibilities of grouping maximum number of adjacent zeroes. It should be powers of two. Start from highest power of two and upto least power of two. Highest power is equal to the number of variables considered in K-map and least power is zero.
- Each grouping will give either a literal or one sum term. It is known as **prime implicant**. The prime implicant is said to be **essential prime implicant**, if atleast single '0' is not covered with any other groupings but only that grouping covers.
- Note down all the prime implicants and essential prime implicants. The simplified Boolean function contains all essential prime implicants and only the required prime implicants.

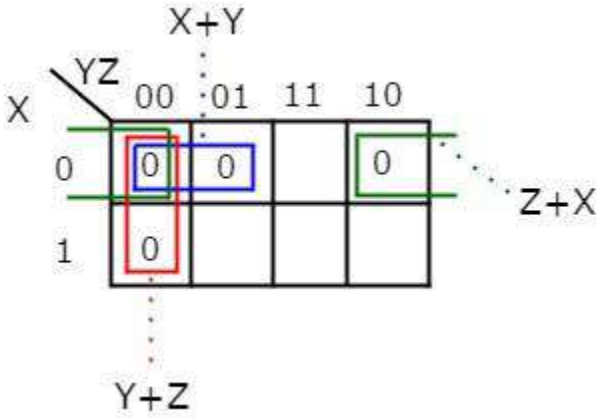
Note − If don't care terms also present, then place don't cares 'x' in the respective cells of K-map. Consider only the don't cares 'x' that are helpful for grouping maximum number of adjacent zeroes. In those cases, treat the don't care value as '0'.

Example

Let us **simplify** the following Boolean function, f(X,Y,Z)=∏M(0,1,2,4)f(X,Y,Z)=∏M(0,1,2,4) using K-map. The given Boolean function is in product of Max terms form. It is having 3 variables X, Y & Z. So, we require 3 variable K-map. The given Max terms are M_0, M_1, M_2 & M_4. The 3 **variable K-map** with zeroes corresponding to the given Max terms is shown in the following figure.



There are no possibilities of grouping either 8 adjacent zeroes or 4 adjacent zeroes. There are three possibilities of grouping 2 adjacent zeroes. After these three groupings, there is no single zero left as ungrouped. The **3 variable K-map** with these three **groupings** is shown in the following figure.



Here, we got three prime implicants X + Y, Y + Z & Z + X. All these prime implicants are **essential** because one zero in each grouping is not covered by any other groupings except with their individual groupings.

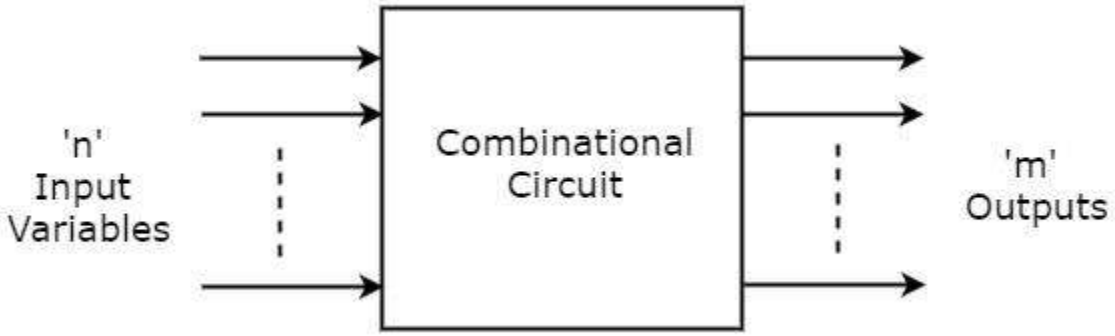
Therefore, the **simplified Boolean function** is

$$\mathbf{f = (X + Y).(Y + Z).(Z + X)}$$

In this way, we can easily simplify the Boolean functions up to 5 variables using K-map method. For more than 5 variables, it is difficult to simplify the functions using K-Maps. Because, the number of **cells** in K-map gets **doubled** by including a new variable.

Combinational Circuits

Combinational circuits consist of Logic gates. These circuits operate with binary values. The output(s) of combinational circuit depends on the combination of present inputs. The following figure shows the **block diagram** of combinational circuit.



This combinational circuit has 'n' input variables and 'm' outputs. Each combination of input variables will affect the output(s).

Design procedure of Combinational circuits

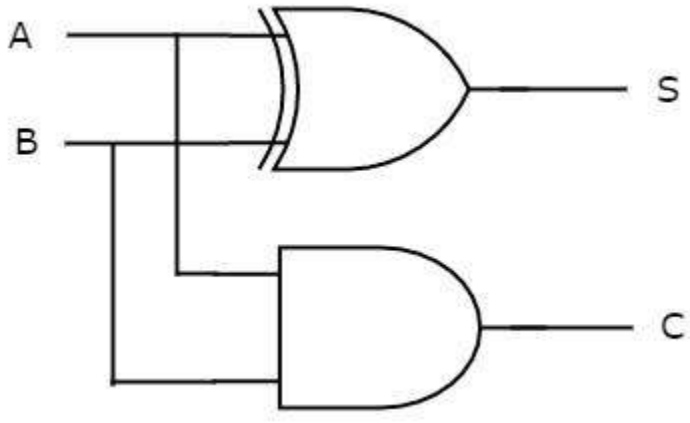
- Find the required number of input variables and outputs from given specifications.
- Formulate the **Truth table**. If there are 'n' input variables, then there will be 2n possible combinations. For each combination of input, find the output values.
- Find the **Boolean expressions** for each output. If necessary, simplify those expressions.
- Implement the above Boolean expressions corresponding to each output by using **Logic gates**.
- In this chapter, let us discuss about the basic arithmetic circuits like Binary adder and Binary subtractor. These circuits can be operated with binary values 0 and 1.
- Binary Adder
- The most basic arithmetic operation is addition. The circuit, which performs the addition of two binary numbers is known as **Binary adder**. First, let us implement an adder, which performs the addition of two bits.

Half Adder

- Half adder is a combinational circuit, which performs the addition of two binary numbers A and B are of **single bit**. It produces two outputs sum, S & carry, C.
- The **Truth table** of Half adder is shown below.

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- When we do the addition of two bits, the resultant sum can have the values ranging from 0 to 2 in decimal. We can represent the decimal digits 0 and 1 with single bit in binary. But, we can't represent decimal digit 2 with single bit in binary. So, we require two bits for representing it in binary.
- Let, sum, S is the Least significant bit and carry, C is the Most significant bit of the resultant sum. For first three combinations of inputs, carry, C is zero and the value of S will be either zero or one based on the **number of ones** present at the inputs. But, for last combination of inputs, carry, C is one and sum, S is zero, since the resultant sum is two.
- From Truth table, we can directly write the **Boolean functions**for each output as
    - $S=A \oplus BS=A \oplus B$
- $C=ABC=AB$
- We can implement the above functions with 2-input Ex-OR gate & 2-input AND gate. The **circuit diagram** of Half adder is shown in the following figure.



- In the above circuit, a two input Ex-OR gate & two input AND gate produces sum, S & carry, C respectively. Therefore, Half-adder performs the addition of two bits.

Full Adder

- Full adder is a combinational circuit, which performs the **addition of three bits** A, B and C_{in}. Where, A & B are the two parallel significant bits and C_{in} is the carry bit, which is generated from previous stage. This Full adder also produces two outputs sum, S & carry, C_{out}, which are similar to Half adder.
- The **Truth table** of Full adder is shown below.

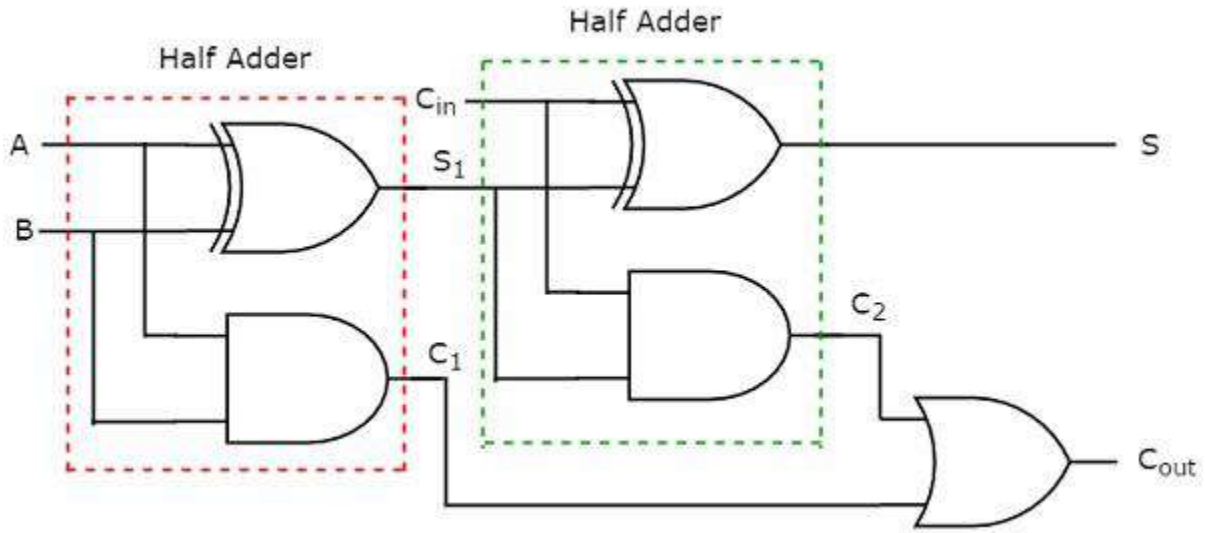
Inputs			**Outputs**	
A	**B**	**C_{in}**	**C_{out}**	**S**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

- When we do the addition of three bits, the resultant sum can have the values ranging from 0 to 3 in decimal. We can represent the decimal digits 0 and 1 with single bit in binary. But, we can't represent the decimal digits 2 and 3 with single bit in binary. So, we require two bits for representing those two decimal digits in binary.
- Let, sum, S is the Least significant bit and carry, C_{out} is the Most significant bit of resultant sum. It is easy to fill the values of outputs for all combinations of inputs in the truth table. Just count

the **number of ones** present at the inputs and write the equivalent binary number at outputs. If C_{in} is equal to zero, then Full adder truth table is same as that of Half adder truth table.

- We will get the following **Boolean functions** for each output after simplification.
    - S=A⊕B⊕CinS=A⊕B⊕Cin

- cout=AB+(A⊕B)cincout=AB+(A⊕B)cin
- The sum, S is equal to one, when odd number of ones present at the inputs. We know that Ex-OR gate produces an output, which is an odd function. So, we can use either two 2input Ex-OR gates or one 3-input Ex-OR gate in order to produce sum, S. We can implement carry, C_{out} using two 2-input AND gates & one OR gate. The **circuit diagram** of Full adder is shown in the following figure.

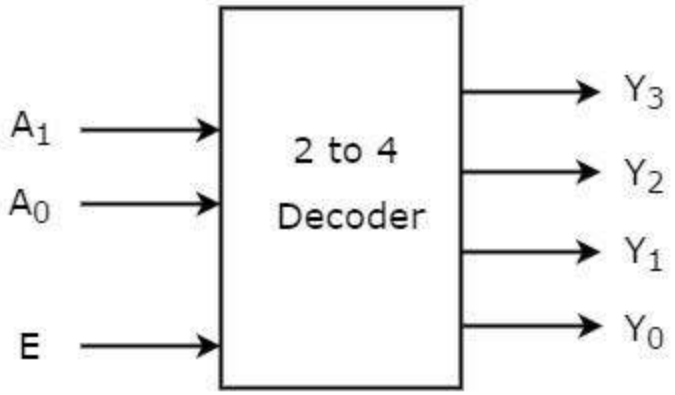


- This adder is called as **Full adder** because for implementing one Full adder, we require two Half adders and one OR gate. If C_{in} is zero, then Full adder becomes Half adder. We can verify it easily from the above circuit diagram or from the Boolean functions of outputs of Full adder.

Decoder is a combinational circuit that has 'n' input lines and maximum of 2^n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The outputs of the decoder are nothing but the **min terms**of 'n' input variables (lines), when it is enabled.

2 to 4 Decoder

Let 2 to 4 Decoder has two inputs A_1 & A_0 and four outputs Y_3, Y_2, Y_1 & Y_0. The **block diagram** of 2 to 4 decoder is shown in the following figure.



One of these four outputs will be '1' for each combination of inputs when enable, E is '1'. The **Truth table** of 2 to 4 decoder is shown below.

Enable	**Inputs**		**Outputs**			
E	**A_1**	**A_0**	**Y_3**	**Y_2**	**Y_1**	**Y_0**
0	x	x	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

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

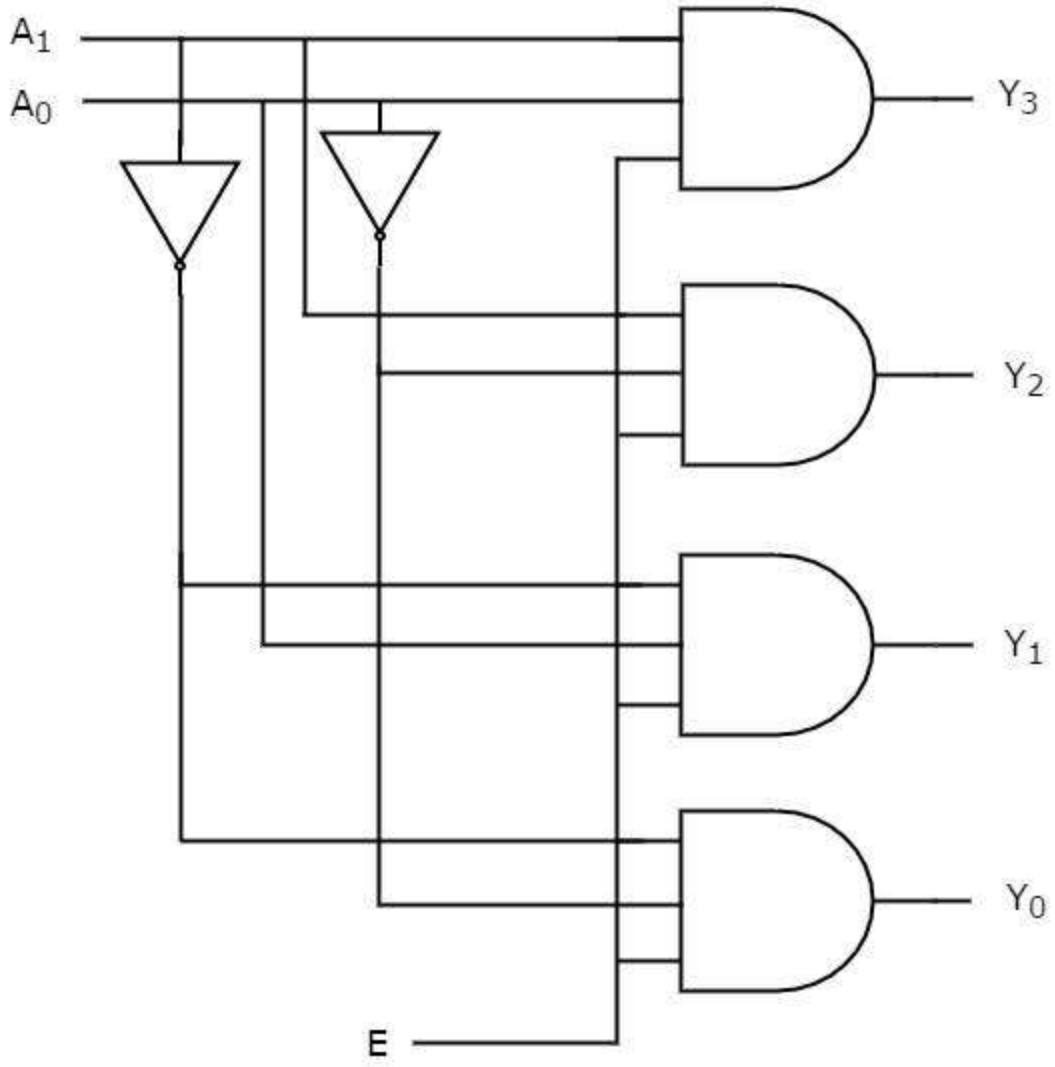
Y3=E.A1.A0Y3=E.A1.A0

Y2=E.A1.A0'Y2=E.A1.A0'

Y1=E.A1'.A0Y1=E.A1'.A0

Y0=E.A1'.A0'Y0=E.A1'.A0'

Each output is having one product term. So, there are four product terms in total. We can implement these four product terms by using four AND gates having three inputs each & two inverters. The **circuit diagram** of 2 to 4 decoder is shown in the following figure.



Therefore, the outputs of 2 to 4 decoder are nothing but the **min terms** of two input variables A_1 & A_0, when enable, E is equal to one. If enable, E is zero, then all the outputs of decoder will be equal to zero.

Similarly, 3 to 8 decoder produces eight min terms of three input variables A_2, A_1 & A_0 and 4 to 16 decoder produces sixteen min terms of four input variables A_3, A_2, A_1 & A_0.

Implementation of Higher-order Decoders

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

- 3 to 8 decoder
- 4 to 16 decoder

3 to 8 Decoder

In this section, let us implement **3 to 8 decoder using 2 to 4 decoders**. We know that 2 to 4 Decoder has two inputs, A_1 & $A_0$and four outputs, Y_3 to Y_0. Whereas, 3 to 8 Decoder has three inputs A_2, A_1 & A_0 and eight outputs, Y_7 to Y_0.

We can find the number of lower order decoders required for implementing higher order decoder using the following formula.

Requirednumberoflowerorderdecoders=m2m1Requirednumberoflowerorderdecoders=m2m1

Where,

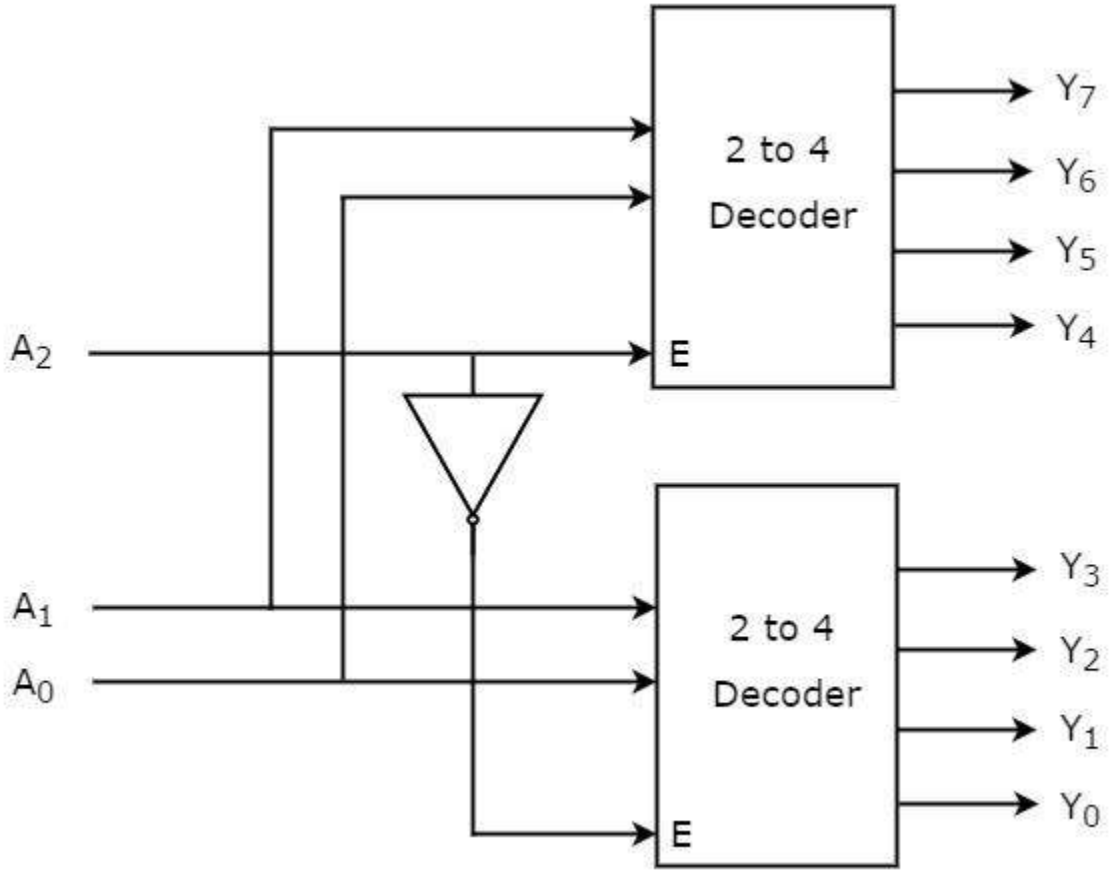
m1m1 is the number of outputs of lower order decoder.

m2m2 is the number of outputs of higher order decoder.

Here, m1m1 = 4 and m2m2 = 8. Substitute, these two values in the above formula.

Requirednumberof2to4decoders=84=2Requirednumberof2to4decoders=84=2

Therefore, we require two 2 to 4 decoders for implementing one 3 to 8 decoder. The **block diagram** of 3 to 8 decoder using 2 to 4 decoders is shown in the following figure.



The parallel inputs A_1 & A_0 are applied to each 2 to 4 decoder. The complement of input A_2 is connected to Enable, E of lower 2 to 4 decoder in order to get the outputs, Y_3 to Y_0. These are

the **lower four min terms**. The input, A_2 is directly connected to Enable, E of upper 2 to 4 decoder in order to get the outputs, $Y_7$to Y_4. These are the **higher four min terms**.

4 to 16 Decoder

In this section, let us implement **4 to 16 decoder using 3 to 8 decoders**. We know that 3 to 8 Decoder has three inputs A_2, A_1& A_0 and eight outputs, Y_7 to Y_0. Whereas, 4 to 16 Decoder has four inputs A_3, A_2, A_1 & A_0 and sixteen outputs, Y_{15} to Y_0

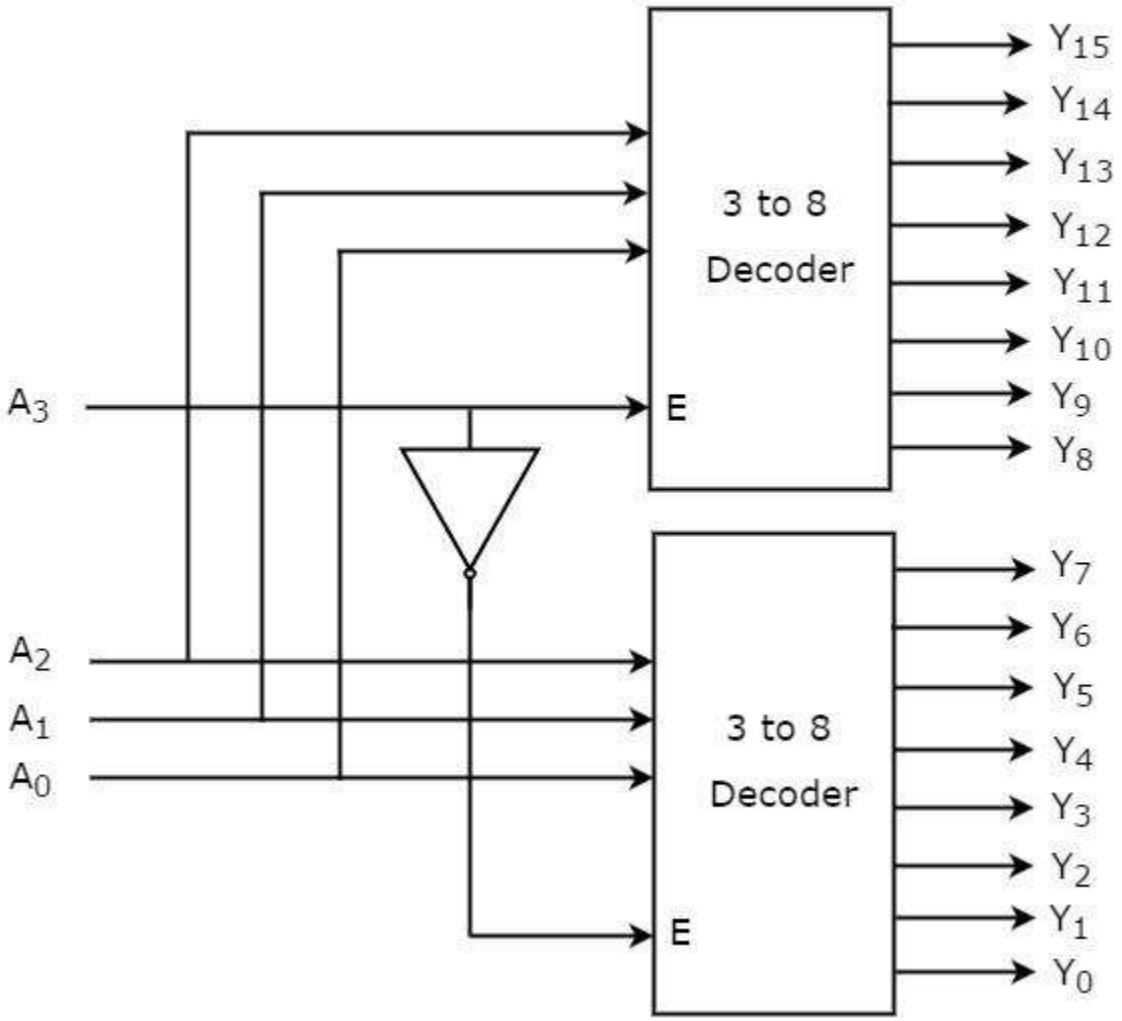
We know the following formula for finding the number of lower order decoders required.

Requirednumberoflowerorderdecoders=m2m1Requirednumberoflowerorderdecoders=m2m1

Substitute, m1m1 = 8 and m2m2 = 16 in the above formula.

Requirednumberof3to8decoders=168=2Requirednumberof3to8decoders=168=2

Therefore, we require two 3 to 8 decoders for implementing one 4 to 16 decoder. The **block diagram** of 4 to 16 decoder using 3 to 8 decoders is shown in the following figure.

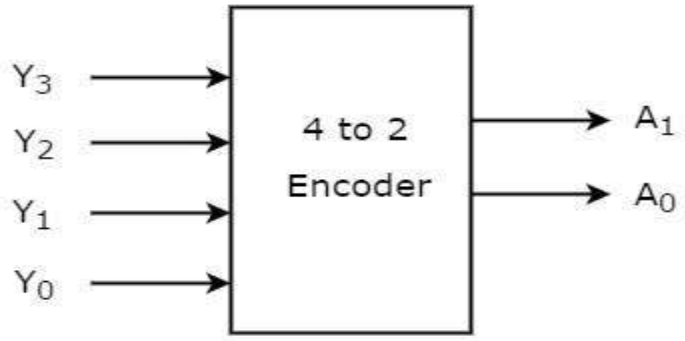


The parallel inputs A_2, A_1 & A_0 are applied to each 3 to 8 decoder. The complement of input, A3 is connected to Enable, E of lower 3 to 8 decoder in order to get the outputs, Y_7 to Y_0. These are the **lower eight min terms**. The input, A_3 is directly connected to Enable, E of upper 3 to 8 decoder in order to get the outputs, Y_{15} to Y_8. These are the **higher eight min terms**.

An **Encoder** is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^ninput lines with 'n' bits. It is optional to represent the enable signal in encoders.

4 to 2 Encoder

Let 4 to 2 Encoder has four inputs Y_3, Y_2, Y_1 & Y_0 and two outputs A_1 & A_0. The **block diagram** of 4 to 2 Encoder is shown in the following figure.



At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output. The **Truth table** of 4 to 2 encoder is shown below.

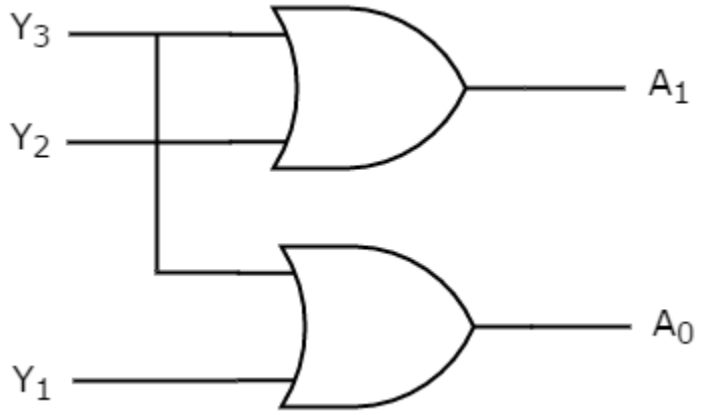
Inputs				Outputs	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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

$$A1=Y3+Y2A1=Y3+Y2$$

$$A0=Y3+Y1A0=Y3+Y1$$

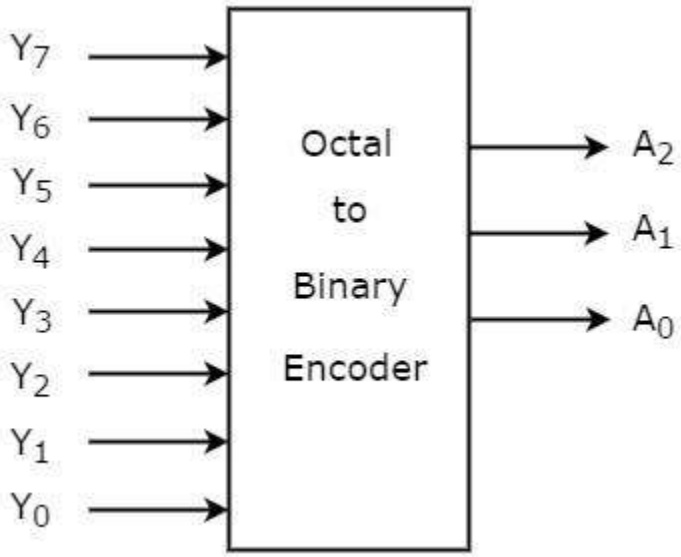
We can implement the above two Boolean functions by using two input OR gates. The **circuit diagram** of 4 to 2 encoder is shown in the following figure.



The above circuit diagram contains two OR gates. These OR gates encode the four inputs with two bits

Octal to Binary Encoder

Octal to binary Encoder has eight inputs, Y_7 to Y_0 and three outputs A_2, A_1 & A_0. Octal to binary encoder is nothing but 8 to 3 encoder. The **block diagram** of octal to binary Encoder is shown in the following figure.



At any time, only one of these eight inputs can be '1' in order to get the respective binary code. The **Truth table** of octal to binary encoder is shown below.

Inputs								Outputs		
Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

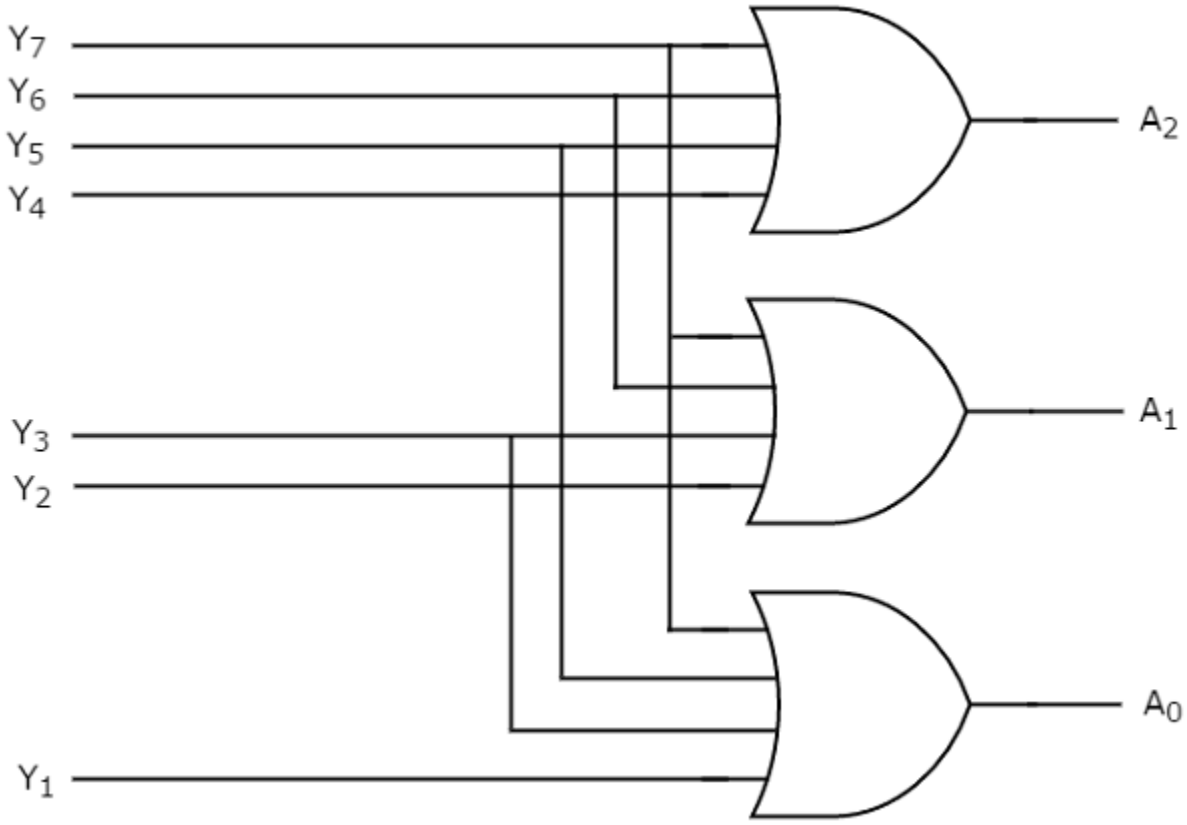
From Truth table, we can write the **Boolean functions** for each output as

$$A2=Y7+Y6+Y5+Y4A2=Y7+Y6+Y5+Y4$$

$$A1=Y7+Y6+Y3+Y2A1=Y7+Y6+Y3+Y2$$

$$A0=Y7+Y5+Y3+Y1A0=Y7+Y5+Y3+Y1$$

We can implement the above Boolean functions by using four input OR gates. The **circuit diagram** of octal to binary encoder is shown in the following figure.



The above circuit diagram contains three 4-input OR gates. These OR gates encode the eight inputs with three bits.

Drawbacks of Encoder

Following are the drawbacks of normal encoder.

- There is an ambiguity, when all outputs of encoder are equal to zero. Because, it could be the code corresponding to the inputs, when only least significant input is one or when all inputs are zero.
- If more than one input is active High, then the encoder produces an output, which may not be the correct code. For **example**, if both Y_3 and Y_6 are '1', then the encoder produces 111 at the output. This is neither equivalent code corresponding to Y_3, when it is '1' nor the equivalent code corresponding to Y_6, when it is '1'.

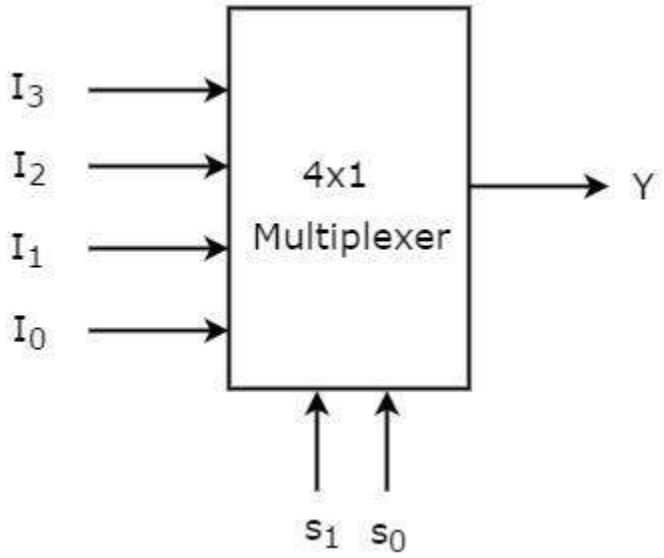
Multiplexer

is a combinational circuit that has maximum of 2^ndata 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.



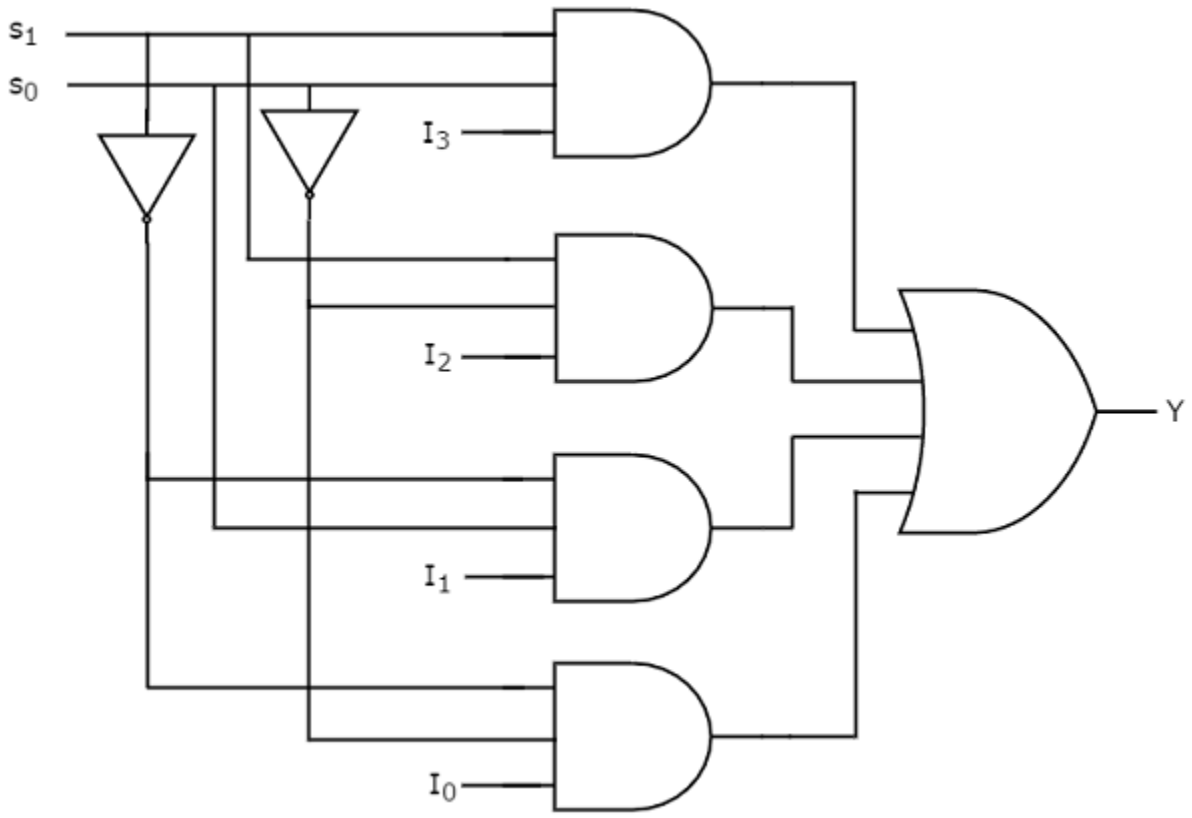
One of these 4 inputs will be connected to the output based on the combination of inputs present at these two selection lines. **Truth table** of 4x1 Multiplexer is shown below.

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

$$Y=S1'S0'I0+S1'S0I1+S1S0'I2+S1S0I3Y=S1'S0'I0+S1'S0I1+S1S0'I2+S1S0I3$$

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

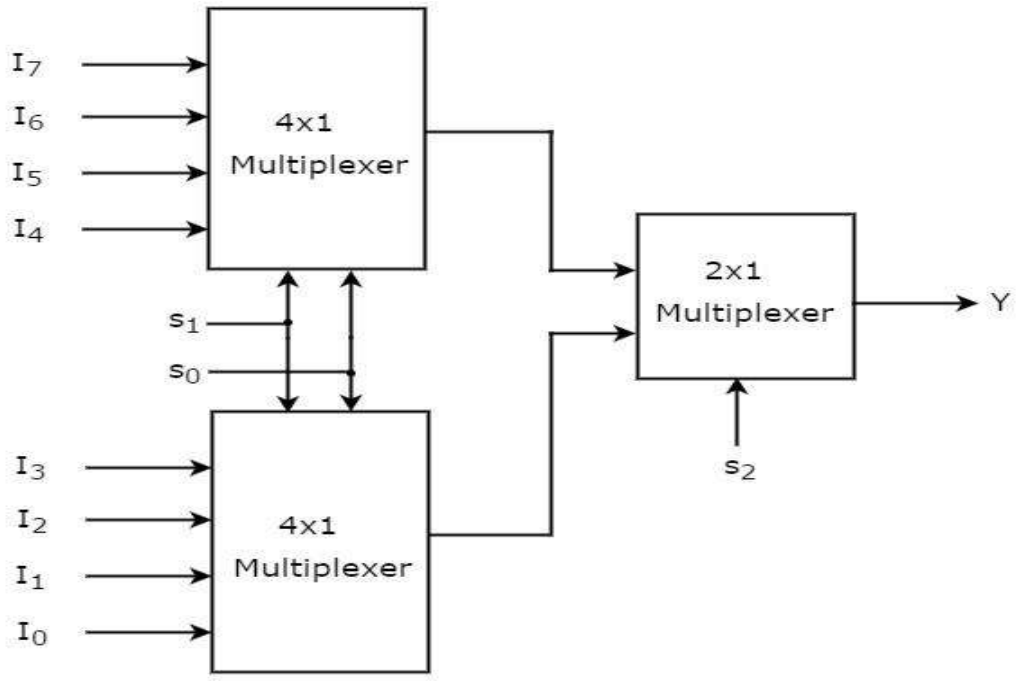
In this section, let us implement 8x1 Multiplexer using 4x1 Multiplexers and 2x1 Multiplexer. We know that 4x1 Multiplexer has 4 data inputs, 2 selection lines and one output. Whereas, 8x1 Multiplexer has 8 data inputs, 3 selection lines and one output.

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.

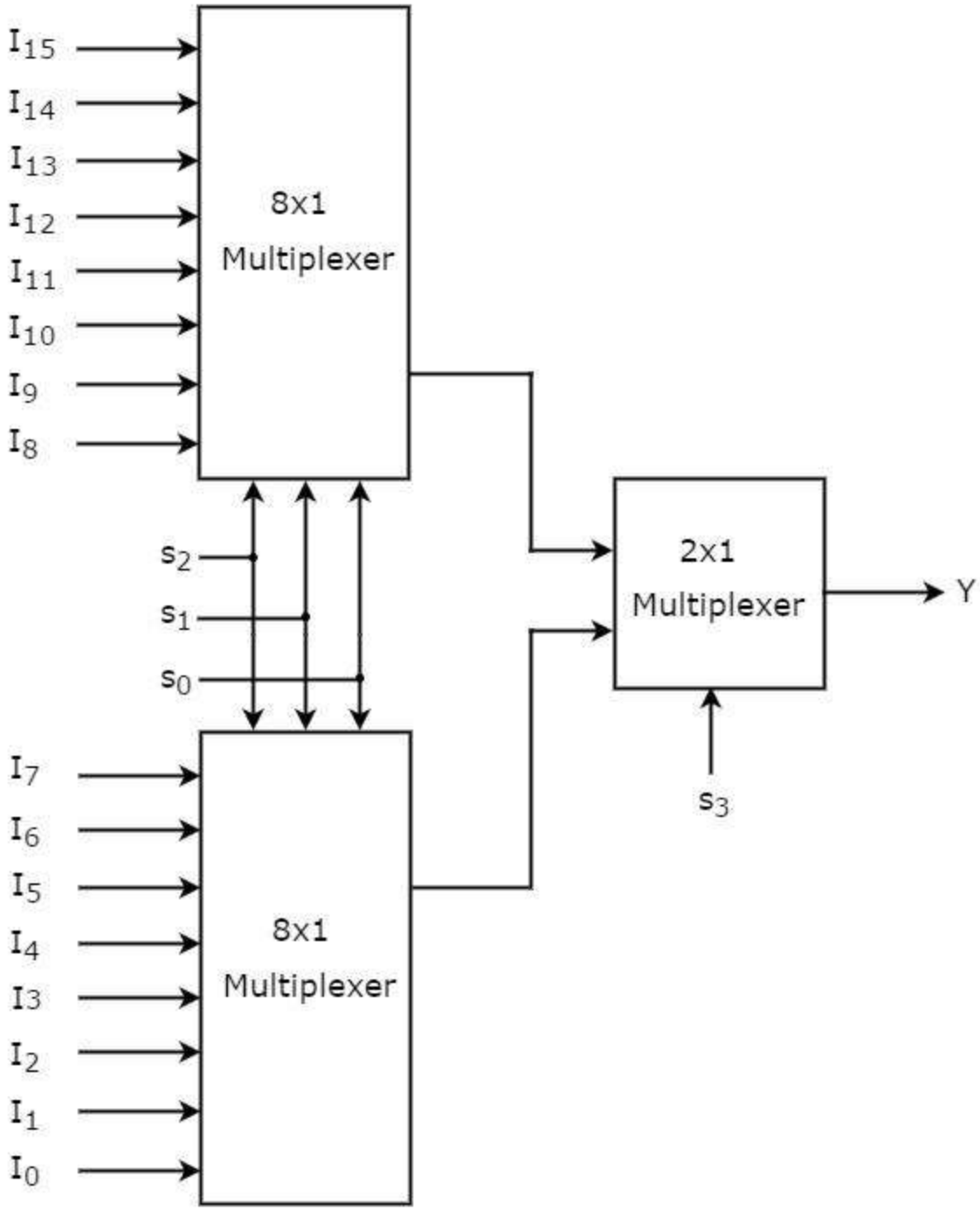
So, we require two **8x1 Multiplexers** in first stage in order to get the 16 data inputs. Since, each 8x1 Multiplexer produces one output, we require a 2x1 Multiplexer in second stage by considering the outputs of first stage as inputs and to produce the final output.

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.

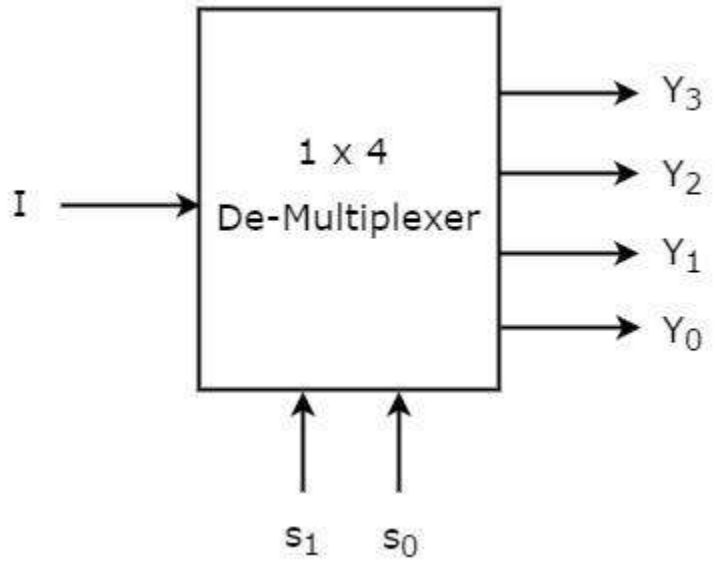
De-Multiplexer

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.

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

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

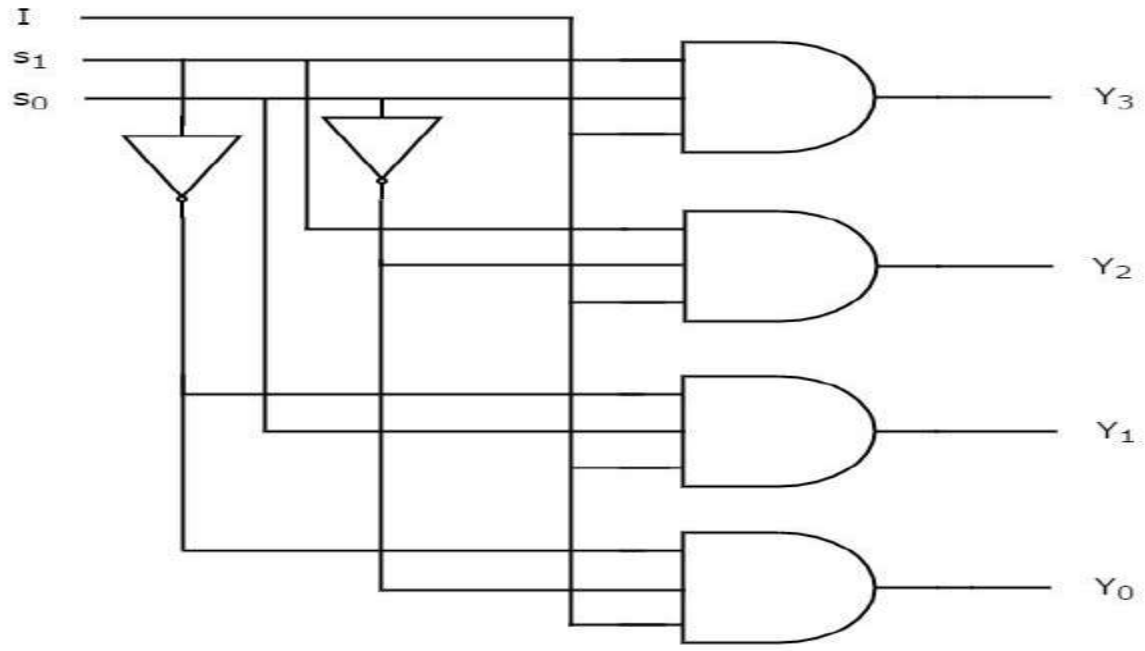
Y3=s1s0IY3=s1s0I

Y2=s1s0'IY2=s1s0'I

Y1=s1's0IY1=s1's0I

Y0=s1's0'IY0=s1's0'I

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.

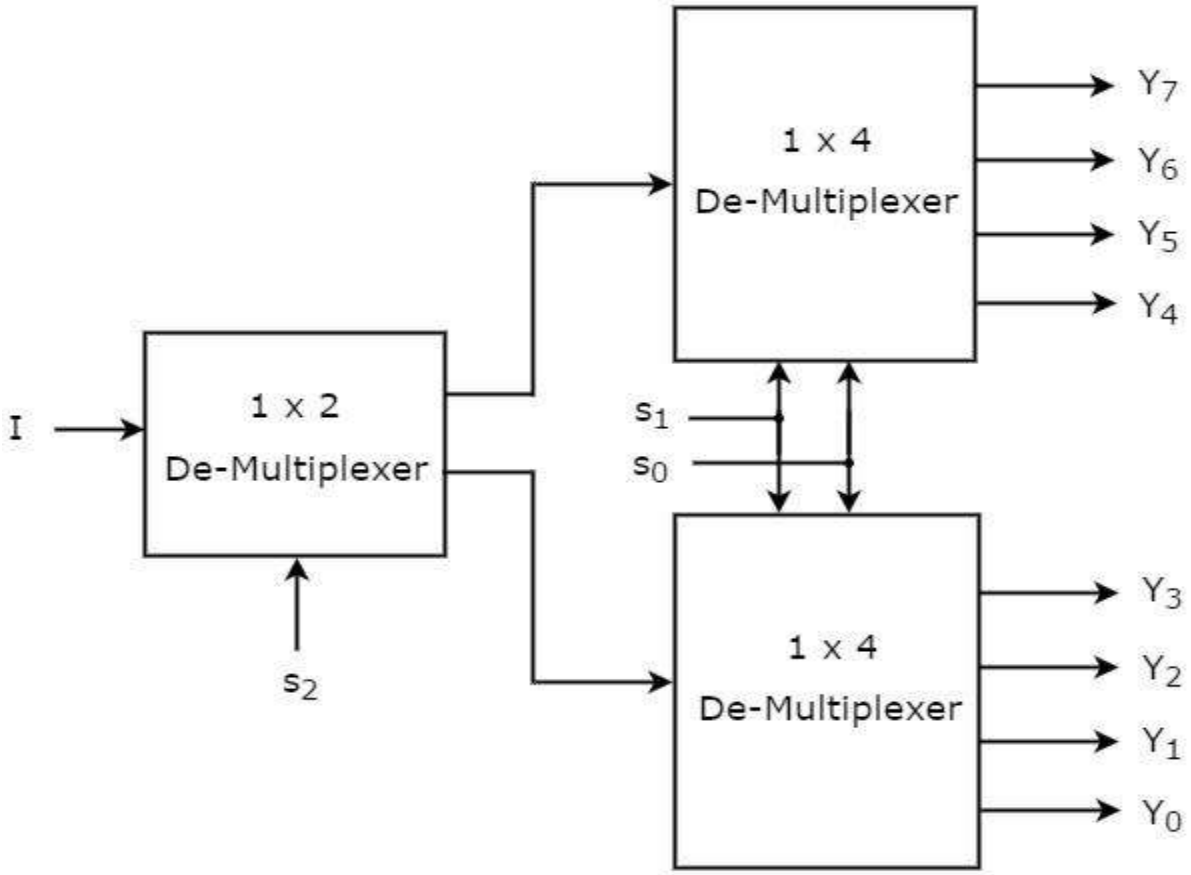
So, we require two **1x4 De-Multiplexers** in second stage in order to get the final eight outputs. Since, the number of inputs in second stage is two, we require **1x2 DeMultiplexer** in first stage so that the outputs of first stage will be the inputs of second stage. Input of this 1x2 De-Multiplexer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

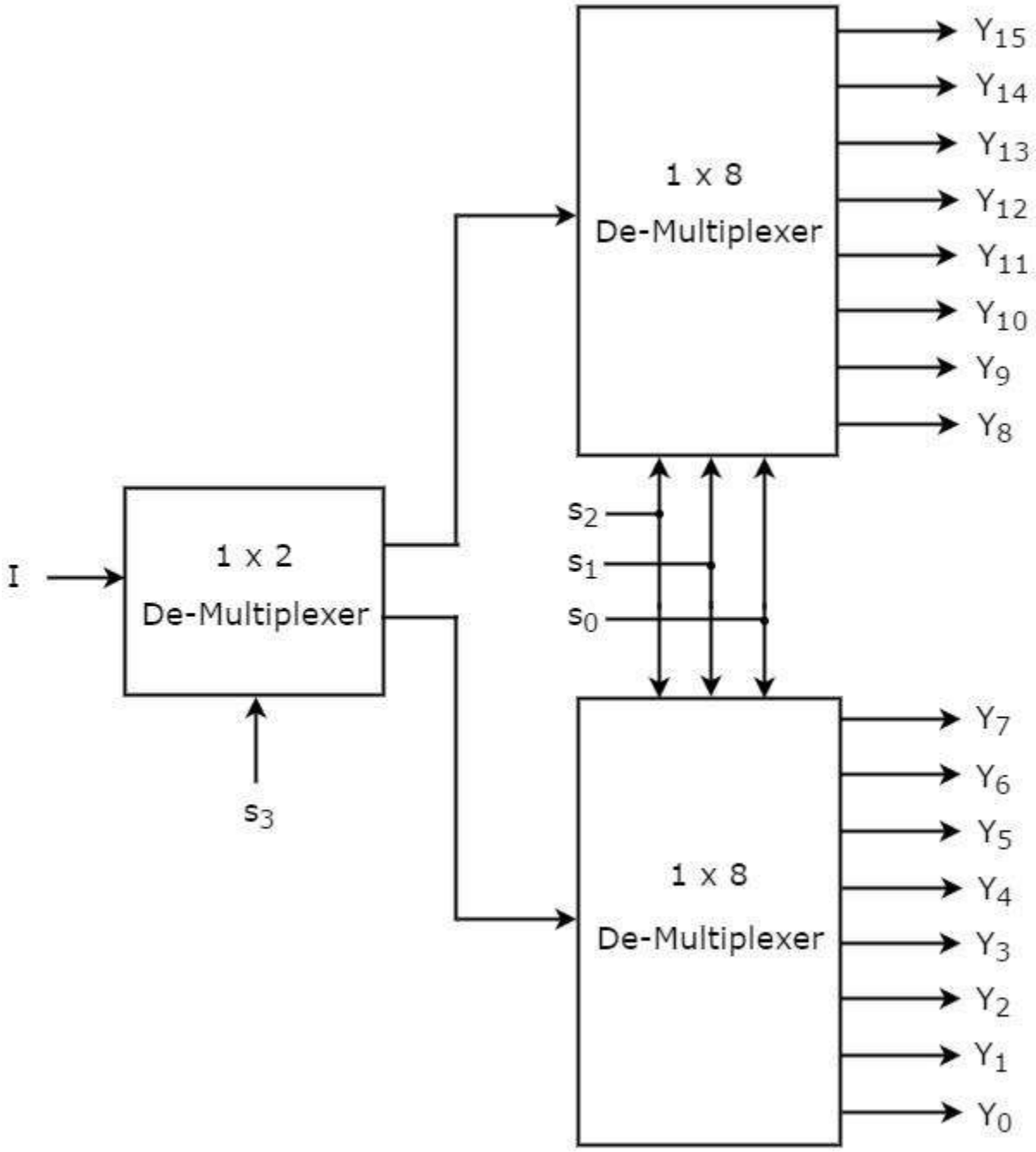
1x16 De-Multiplexer

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.