III Semester

ANALOG AND DIGITAL ELECTRONICS			
Course Code	**21CS33**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:2:0	SEE Marks	50
Total Hours of Pedagogy	40 T + 20 P	Total Marks	100
Credits	04	Exam Hours	03

Course Learning Objectives:

CLO 1. Explain the use of photo electronics devices, 555 timer IC, Regulator ICs and uA741

CLO 2. Make use of simplifying techniques in the design of combinational circuits.

CLO 3. Illustrate combinational and sequential digital circuits

CLO 4. Demonstrate the use of flipflops and apply for registers

CLO 5. Design and test counters, Analog-to-Digital and Digital-to-Analog conversion techniques.

Teaching-Learning Process (General Instructions)

These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes.

1. Lecturer method (L) does not mean only traditional lecture method, but different type of teaching methods may be adopted to develop the outcomes.
2. Show Video/animation films to explain functioning of various concepts.
3. Encourage collaborative (Group Learning) Learning in the class.
4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.
5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop thinking skills such as the ability to evaluate, generalize, and analyze information rather than simply recall it.
6. Topics will be introduced in a multiple representation.
7. Show the different ways to solve the same problem and encourage the students to come up with their own creative ways to solve them.
8. Discuss how every concept can be applied to the real world - and when that's possible, it helps improve the students' understanding.

Module-1

BJT Biasing: Fixed bias, Collector to base Bias, voltage divider bias

Operational Amplifier Application Circuits: Peak Detector, Schmitt trigger, Active Filters, Non-Linear Amplifier, Relaxation Oscillator, Current-to-Voltage and Voltage-to-Current Converter, Regulated Power Supply Parameters, adjustable voltage regulator, D to A and A to D converter.

Textbook 1: Part A: Chapter 4 (Sections 4.2, 4.3, 4.4), Chapter 7 (Sections 7.4, 7.6 to 7.11), Chapter 8 (Sections 8.1 and 8.5), Chapter 9.

Laboratory Component:

1. Simulate BJT CE voltage divider biased voltage amplifier using any suitable circuit simulator.
2. Using ua 741 Opamp, design a 1 kHz Relaxation Oscillator with 50% duty cycle
3. Design an astable multivibrator circuit for three cases of duty cycle (50%, <50% and >50%) using NE 555 timer IC.
4. Using ua 741 opamap, design a window comparator for any given UTP and LTP.

Teaching-Learning Process	1. Demonstration of circuits using simulation. 2. Project work: Design a integrated power supply and function generator operating at audio frequency. Sine, square and triangular functions are to be generated. 3. Chalk and Board for numerical

Module-2

Karnaugh maps: minimum forms of switching functions, two and three variable Karnaugh maps, four variable Karnaugh maps, determination of minimum expressions using essential prime implicants, Quine-McClusky Method: determination of prime implicants, the prime implicant chart, Petricks method, simplification of incompletely specified functions, simplification using map-entered variables **Textbook 1: Part B: Chapter 5 (Sections 5.1 to 5.4) Chapter 6 (Sections 6.1 to 6.5)**	
Laboratory Component: 1. Given a 4-variable logic expression, simplify it using appropriate technique and inplement the same using basic gates.	
Teaching-Learning Process	1. Chalk and Board for numerical 2. Laboratory Demonstration
Module-3	
Combinational circuit design and simulation using gates: Review of Combinational circuit design, design of circuits with limited Gate Fan-in, Gate delays and Timing diagrams, Hazards in combinational Logic, simulation and testing of logic circuits Multiplexers, Decoders and Programmable Logic Devices: Multiplexers, three state buffers, decoders and encoders, Programmable Logic devices. **Textbook 1: Part B: Chapter 8, Chapter 9 (Sections 9.1 to 9.6)**	
Laboratory Component: 1. Given a 4-variable logic expression, simplify it using appropriate technique and realize the simplified logic expression using 8:1 multiplexer IC. 2. Design and implement code converter I) Binary to Gray (II) Gray to Binary Code	
Teaching-Learning Process	1. Demonstration using simulator 2. Case study: Applications of Programmable Logic device 3. Chalk and Board for numerical
Module-4	
Introduction to VHDL: VHDL description of combinational circuits, VHDL Models for multiplexers, VHDL Modules. Latches and Flip-Flops: Set Reset Latch, Gated Latches, Edge-Triggered D Flip Flop 3,SR Flip Flop, J K Flip Flop, T Flip Flop. **Textbook 1: Part B: Chapter 10(Sections 10.1 to 10.3), Chapter 11 (Sections 11.1 to 11.7)**	
Laboratory Component: 1. Given a 4-variable logic expression, simplify it using appropriate technique and simulate the same in HDL simulator 2. Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table. And implement the same in HDL.	
Teaching-Learning Process	1. Demonstration using simulator 2. Case study: Arithmetic and Logic unit in VHDL 3. Chalk and Board for numerical
Module-5	
Registers and Counters: Registers and Register Transfers, Parallel Adder with accumulator, shift registers, design of Binary counters, counters for other sequences, counter design using SR and J K Flip Flops.	

Textbook 1: Part B: Chapter 12 (Sections 12.1 to 12.5)

Laboratory Component:

1. Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.
2. Design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and demonstrate on 7-segment display (using IC-7447)

Teaching-Learning Process	1. Demonstration using simulator 2. Project Work: Designing any counter, use LED / Seven-segment display to display the output 3. Chalk and Board for numerical

Course outcome (Course Skill Set)

At the end of the course the student will be able to:

CO 1. Design and analyze application of analog circuits using photo devices, timer IC, power supply and regulator IC and op-amp.

CO 2. Explain the basic principles of A/D and D/A conversion circuits and develop the same.

CO 3. Simplify digital circuits using Karnaugh Map, and Quine-McClusky Methods

CO 4. Explain Gates and flip flops and make us in designing different data processing circuits, registers and counters and compare the types.

CO 5. Develop simple HDL programs

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination (SEE), and a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together

Continuous Internal Evaluation:

Three Unit Tests each of **20 Marks (duration 01 hour**)

1. First test at the end of 5th week of the semester
2. Second test at the end of the 10th week of the semester
3. Third test at the end of the 15th week of the semester

Two assignments each of **10 Marks**

4. First assignment at the end of 4th week of the semester
5. Second assignment at the end of 9th week of the semester

Practical Sessions need to be assessed by appropriate rubrics and viva-voce method. This will contribute to **20 marks**.

- Rubrics for each Experiment taken average for all Lab components – 15 Marks.
- Viva-Voce– 5 Marks (more emphasized on demonstration topics)

The sum of three tests, two assignments, and practical sessions will be out of 100 marks and will be **scaled down to 50 marks**

(to have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course).

CIE methods /question paper has to be designed to attain the different levels of Bloom's taxonomy as per the outcome defined for the course.

Semester End Examination:

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the subject (**duration 03 hours)**

1. The question paper will have ten questions. Each question is set for 20 marks. Marks scored shall be proportionally reduced to 50 marks
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.

The students have to answer 5 full questions, selecting one full question from each module

Suggested Learning Resources:

Textbooks

1. Charles H Roth and Larry L Kinney and Raghunandan G H Analog and Digital Electronics, Cengage Learning,2019

Reference Books

1. Anil K Maini, Varsha Agarwal, Electronic Devices and Circuits, Wiley, 2012.
2. Donald P Leach, Albert Paul Malvino & Goutam Saha, Digital Principles and Applications, 8th Edition, Tata McGraw Hill, 2015.
3. M. Morris Mani, Digital Design, 4th Edition, Pearson Prentice Hall, 2008.
4. David A. Bell, Electronic Devices and Circuits, 5th Edition, Oxford University Press, 2008

Weblinks and Video Lectures (e-Resources):

1. Analog Electronic Circuits: https://nptel.ac.in/courses/108/102/108102112/
2. Digital Electronic Circuits: https://nptel.ac.in/courses/108/105/108105132/
3. Analog Electronics Lab: http://vlabs.iitkgp.ac.in/be/
4. Digital Electronics Lab: http://vlabs.iitkgp.ac.in/dec

Activity Based Learning (Suggested Activities in Class)/ Practical Based learning

1. Real world problem solving - applying the design concepts of oscillator, amplifier, switch, Digital circuits using Opamps, 555 timer, transistor, Digital ICs and design a application like tone generator, temperature sensor, digital clock, dancing lights etc.

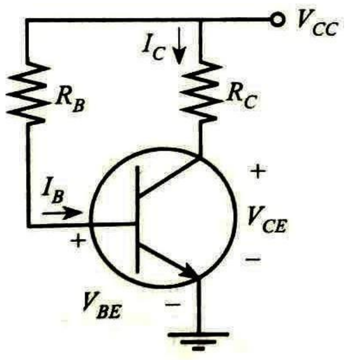
MODULE -1 BJT BIASING

- **Biasing** is a process of connecting DC voltage source to the junction of transistor to make it to operate in the desired region.
- Biasing eliminates the need for separate DC source in the emitter and collector circuit.

Types of Biasing:

(i) Base Bias or Fixed Bias

- The resistance R_B is connected between V_{CC} and base of transistor to establish I_B.
- Because V_{CC} and R_B are fixed, I_B remains fixed hence it is called as fixed bias.



By applying KVL to the base of circuit we get
V_{BE} is 0.7V for silicon and 0.3v for germanium transistor.

$$V_{CC} - I_B R_B - V_{BE} = 0$$

$$I_B R_B + V_{BE} = V_{CC}$$

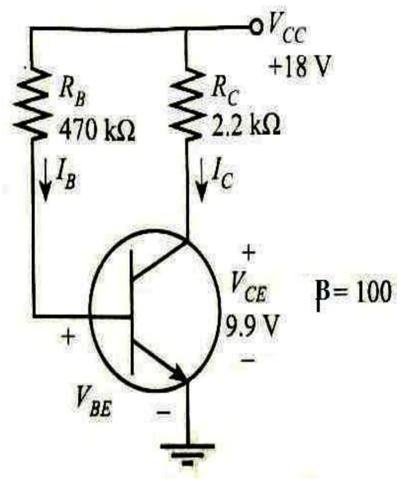
$$\Rightarrow \qquad I_B = \frac{V_{CC} - V_{BE}}{R_B}$$

$$I_C = \beta I_B.$$

$$V_{CC} = V_{CE} + I_C R_C$$
$$V_{CE} = V_{CC} - I_C R_C$$

The collector current is calculated as
By applying KVL to the collector circuit we get

Prob-1) The base bias circuit shown below determine the values of I_C, I_B and V_{CE}.
Soln:



Given: $R_B = 470\ \text{k}\Omega,\ R_C = 2.2\ \text{k}\Omega,\ V_{CC} = 18\ \text{V}$

$$I_B = \frac{V_{CC} - V_{BE}}{R_B} = \frac{18\ \text{V} - 0.7\ \text{V}}{470\ \text{k}\Omega} = 36.8\ \mu\text{A}$$

$$I_C = \beta I_B = 100 \times 36.8\ \mu\text{A} = 3.68\ \text{mA}$$

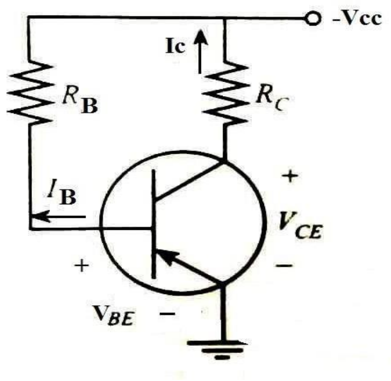
$$V_{CE} = V_{CE} - I_C R_C = 18\ \text{V} - (3.68\ \text{mA} \times 2.2\ \text{k}\Omega) = 9.9\ \text{V}$$

Prob-2) For the above base bias circuit, calculate the maximum and minimum level I_C and V_{CE}. Given β =200 and β=50.

Soln: I_B = 36.8µA

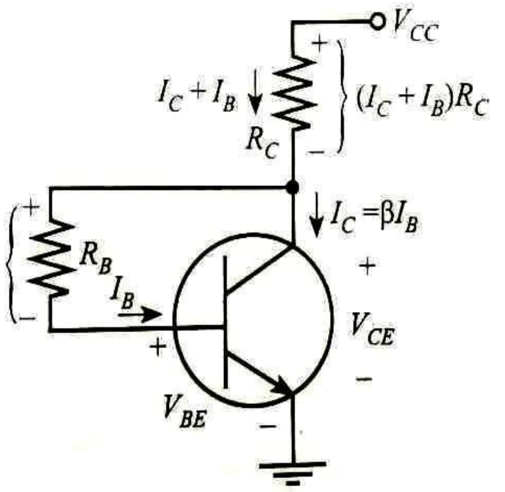
For β =200, I_C=7.36mA and V_{CE}=1.8V and For β =50, I_C=1.84mA and V_{CE}=13.85V

Base Bias using PNP transistor:
The explanation for base biasof PNP is similar to the NPN but the difference is polarities of voltage andcurrents.



Meghana Sambare R

(ii) Collector to Base Bias:



The base resistance is connected between the transistor collector and base terminal
Apply KVL to the outer loop (between collector and emitter)

$$-V_{CE} - R_C(I_C + I_B) + V_{CC} = 0$$

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

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

Also KVL to the loop V_{CE}, $I_B R_B$ and V_{BE}

$$-V_{BE} - I_B R_B + V_{CE} = 0$$

$$V_{CE} = I_B R_B + V_{BE}$$

on equating above equations leads to

$$I_B(R_C + R_B) + I_C R_C = V_{CC} - V_{BE}$$

Substituting $I_C = \beta I_B$ into the above equation we get,

$$I_B(R_C + R_B) + \beta I_B R_C = V_{CC} - V_{BE}$$

This gives

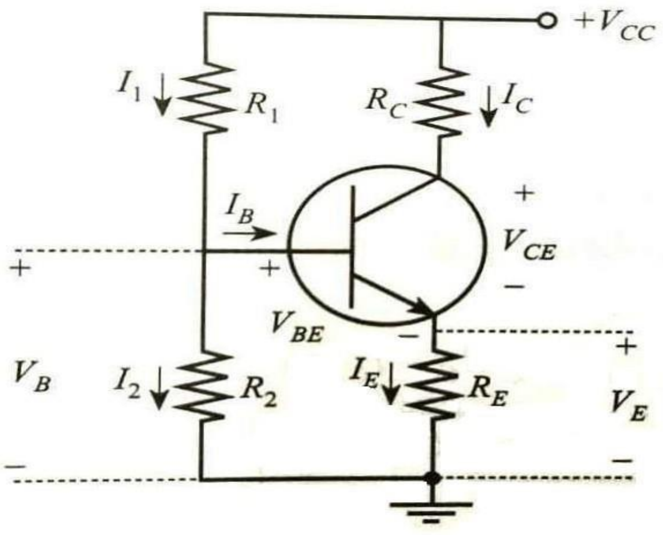
$$I_B = \frac{V_{CC} - V_{BE}}{(\beta + 1)R_C + R_B}$$

$$\therefore I_C = \beta I_B$$

Voltage Divider Bias:

- **It provides more stability when compared to base bias and collector bias.**
- **The resistor R_1 and R_2 form the voltage divider that divides the supply voltage V_{CC} to produce the base voltage V_B.**

- **Voltage divider can be analyzed using Approximation analysis:**



By applying KVL from V_{CC}, R_1, R_2, we get,

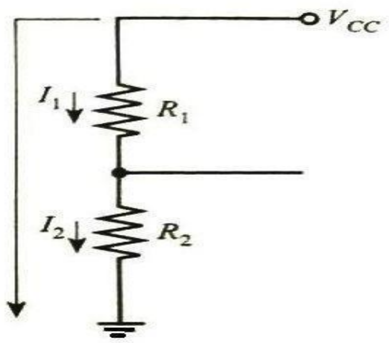
$V_{CC} - I_1 R_1 - I_2 R_2 = 0.$

WKT $\boxed{I_1 \approx I_2}$

$V_{CC} - I_2 R_1 - I_2 R_2 = 0$

$I_2 [R_1 + R_2] = V_{CC}$

$$\boxed{I_2 = \frac{V_{CC}}{R_1 + R_2}}$$



Voltage across R_2 is V_B,

$V_B = I_2 R_2$

$$\boxed{V_B = \frac{V_{CC}}{R_1 + R_2} R_2}$$

Voltage across R_E is V_E,

$$\boxed{V_E = I_E R_E}$$

Applying KVL to base-emitter loop,

$V_B - V_{BE} - V_E = 0$

NOTE : $V_{BE} = V_B - V_E$

$V_E = V_B - V_{BE}$

$I_E R_E = V_B - V_{BE}$

$$\boxed{I_E = \frac{V_B - V_{BE}}{R_E}}$$

Applying KVL to the collector-emitter circuit we have.

$$V_{CC} - I_C R_C - V_{CE} - I_C R_E = 0$$

$$V_{CE} = V_{CC} - I_C R_C - I_C R_E$$

$$\boxed{V_{CE} = V_{CC} - I_C [R_C + R_E]}$$

Problem-3) Determine the values of the resistor R_C and R_E for the circuit given below given that R_1=5K, R_2=1k, V_{BE}=0.7V, V_{CE}=5Vand I_C=2mA and V_{CC}=12V.

Solution:

Applying KVL to the collector circuit we get,

$$V_{CC} - I_C R_C - V_{CE} - I_E R_E = 0$$

$I_C = I_E$=2mA

R_C+R_E =3.5K

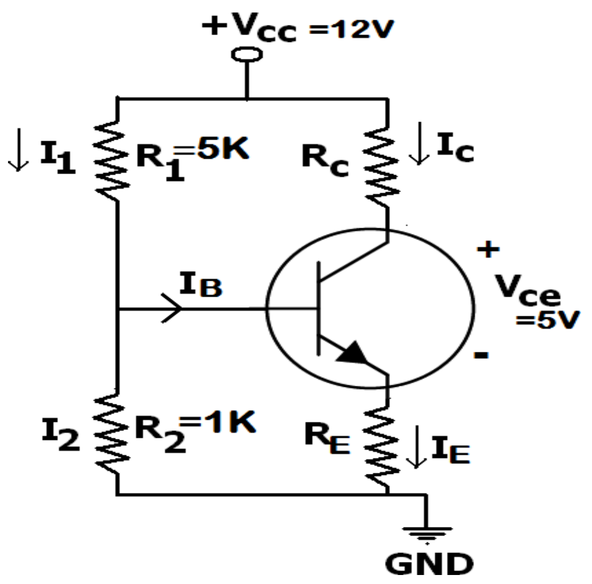
$$\boxed{V_B = \frac{V_{CC}}{R_1 + R_2} R_2}$$

V_B= 2V

$V_E = V_B-V_{BE}$=1.3V

$R_E= V_E*I_E$=2.85K

R_C =3.5K-2.85K=0.65K



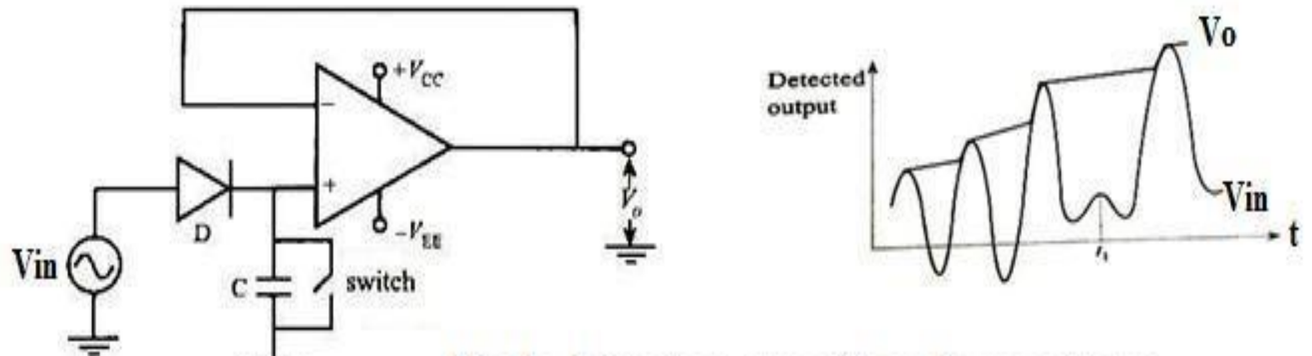
Assignment questions:

1. What is biasing? Explain with neat diagram different types of biasing
2. Explain with neat diagram voltage divider biasing? Mention its advan

Operational Amplifier Application Circuits

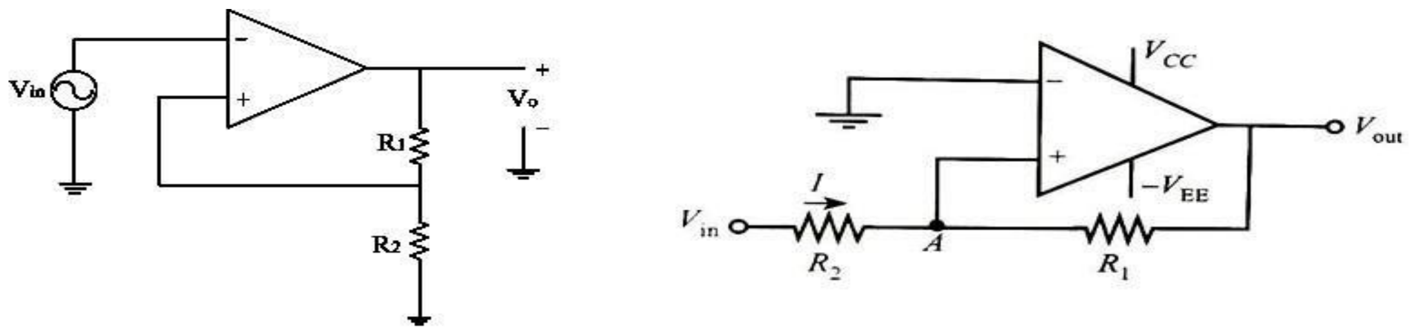
PEAK DETECTOR CIRCUIT:

- Peak detector detects the peak values of the non-sinusoidal wave.
- A Peak detector monitors i/p signal and holds its o/p voltage at the peak level of i/p.
- In the positive half cycle, diode D is forward biased and capacitor C starts charging.
- When input reaches its peak value capacitor gets charged to positive peak value.
- In negative half cycle, as input decreases, diode D is reversed biased and capacitor is isolated and holds the peak value of previous cycle.
- Peak detector finds application in test and measurement instrumentation as well as in amplitude modulation communication.



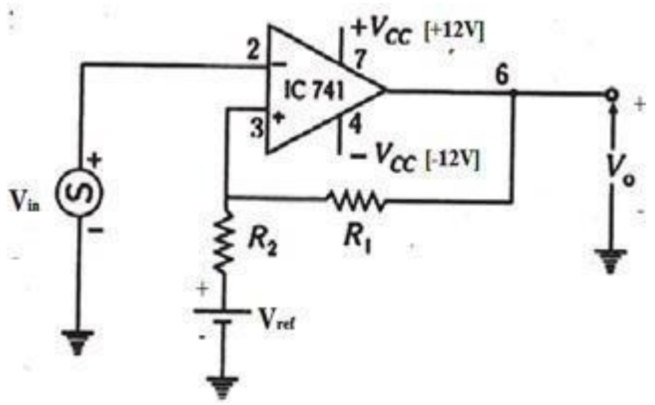
Peak detector circuit and waveform

SCHMITT-TRIGGER



Inverting and non inverting schmitt trigger circuits

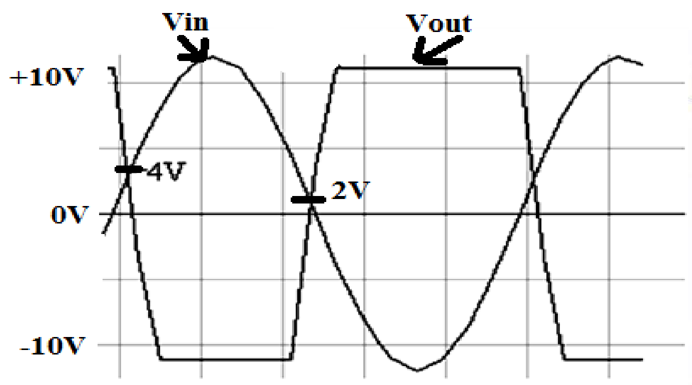
Inverting Schmitt trigger Circuit Diagram and Waveforms:

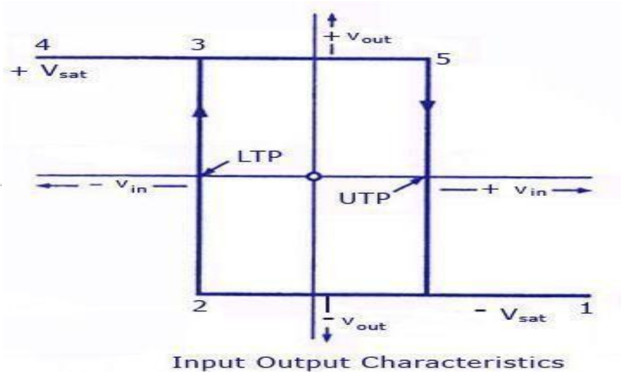


Theory:

- Schmitt Trigger converts an irregular shaped waveform to a square wave or pulse.
- The applied voltage V_{in} triggers (changes) the o\p V_{out} every time when the value of V_{in} exceeds a certain voltage level called the upper threshold voltage [UTP].
- The applied voltage V_{in} also changes the state of the o\p, when the value of V_{in} falls below the certain reference voltage called the lower threshold voltage[LTP].
- The values of LTP &UTP are obtained by using the voltage divider circuit R1 and R2.
- The voltage difference between the upper and lower trigger point is referred to as hysteresis.
- Schmitt trigger devices are typically used for noise immunity and triangle/square wave generators etc
-

Upper and lower trigger points can be written as





$$\text{UTP} = \frac{R_2}{R_1 + R_2} V_{\text{sat}}$$

$$\text{LTP} = \frac{R_2}{R_1 + R_2} (-V_{\text{sat}})$$

$$\begin{aligned} V_{\text{hys}} &= \text{UTP} - \text{LTP} \\ &= \frac{R_2}{R_1 + R_2} V_{\text{sat}} - \frac{R_2}{R_1 + R_2} (-V_{\text{sat}}) \\ &= 2\left(\frac{R_2}{R_1 + R_2} \right) V_{\text{sat}} \\ &= 2\beta V_{\text{sat}} \end{aligned}$$

When the Vo= +Vsat the reference voltage UTP is given by

UTP=[R1*Vref/(R1+R2)] + [R2*Vsat/(R1+R2)]

When the Vo= -Vsat the reference voltage LTP is given by

LTP=[R1*Vref/(R1+R2)] -[R2*Vsat/(R1+R2)]

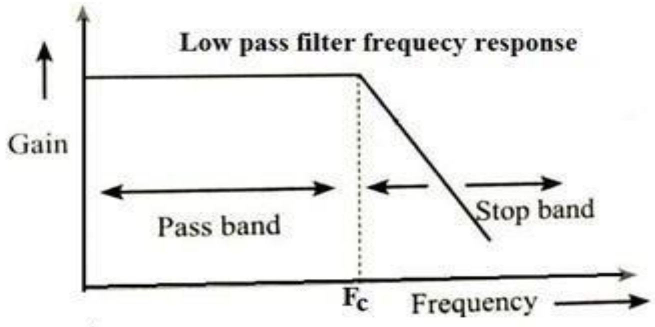
Application of Schmitt trigger:

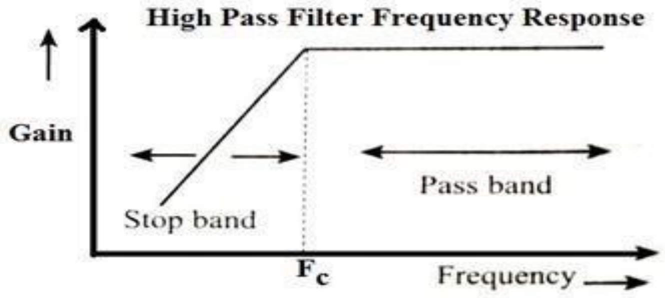
- Digital to analog conversion
- Level detection
-

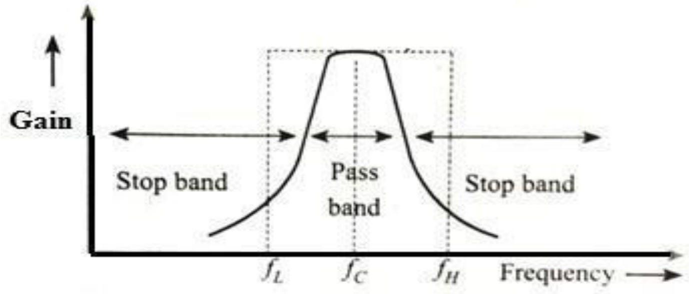
ACTIVE FILTERS:

- ➢ Filters are analog circuit which removes unwanted frequency component from the signal.
- ➢ Two types Passive filters (uses passive components R, C and L) and active filters (uses active components such as amplifier,transistors).
- ➢ The most commonly used filters are
- ❖ **Low Pass Filters:** it is a filter passes low frequency signal and reduces the amplitude of signals with frequency higher than the cut of frequency (F_C).
- ❖ **High Pass Filters:** it is a filter passes high frequency signal and reduces the amplitude of signals with frequency lower than the cut of frequency.
- ❖ **Band Pass Filters:** it is a filter passes frequency within a certain rage and rejects the frequency outside that range. Band pass filter is combination of low and high pass filters.
- ❖ **Band Stop or Band Reject Filters:** it is a filter passes most frequency's but attenuates(reduces) those in a specific range to very low levels. It is the opposite of band pass filter.

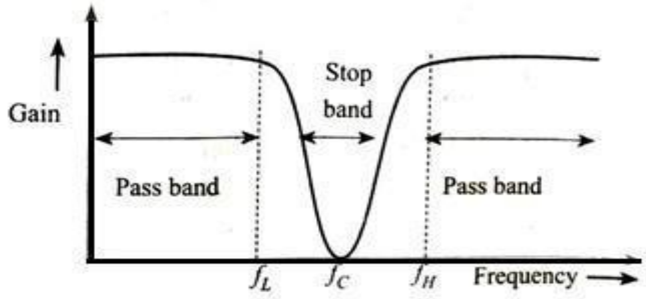
Note: Cut off or corner or break frequency is a boundary in system frequency response at which energy flowing through the system is reduced rather than passing through.







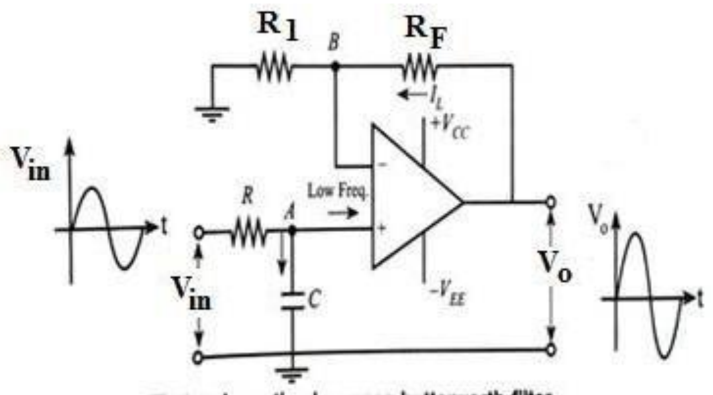
Band pass filter frequency response



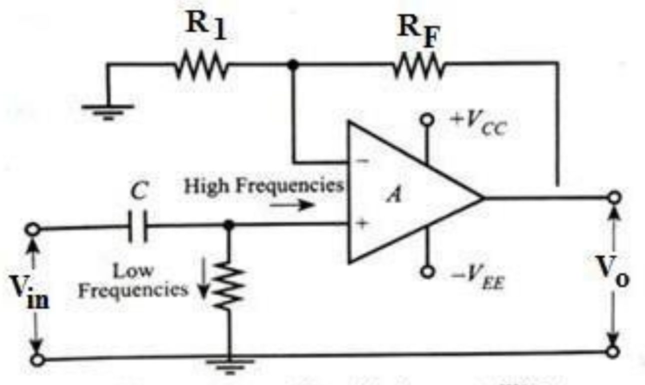
Band reject filter frequency response

First-Order Active low pass Filters

- The simplest low-pass and high-pass active filters are constructed by connecting lag and lead type of R-C sections, respectively, to the non-inverting input of the op-amp wired as a voltage follower.



First order active low pass butterworth filter



First order active high pass filter

- The circuits function as follows. In the case of low-pass circuit at low frequencies, applied input signal appears at the output mostly unattenuated.
- At high frequencies, the output to be near zero.
- The operation of high pass circuit can be explained in similar way.
- The cut-off frequency in both cases is given by Eqn.
- The voltage gain Av is given by Equation.

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

$$A_V = 1 + \frac{R_f}{R_1}$$

Application

- ✓ Used in audio amplifiers
- ✓ Used in speakers to reducing the low or high frequency noise.

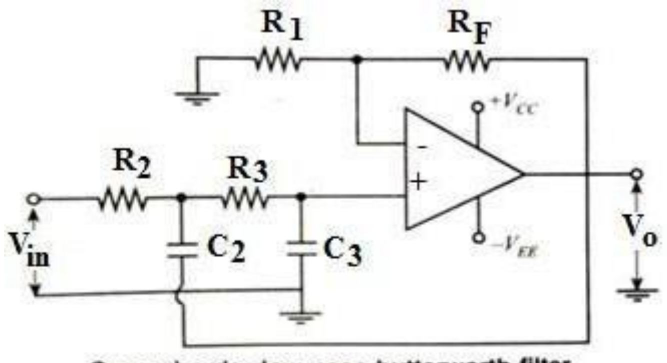
Second-Order Active low pass Filters

- ➢ Second-order filters can be constructed by connecting extra RC-network to the first order filters.
- ➢ The voltage gain Av is given by Equation.

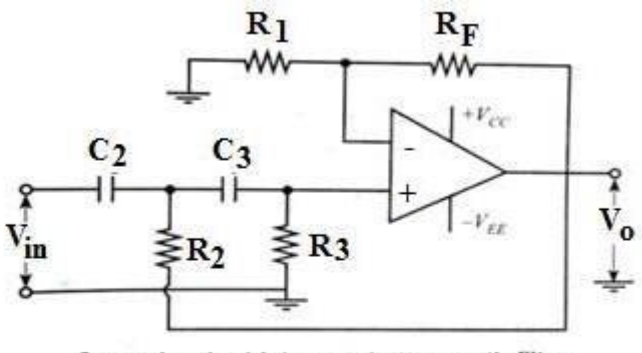
$$A_V = 1 + \frac{R_f}{R_1}$$

- ➢ The cut-off frequency in both cases is given by Eqn.

$$F_C = \frac{1}{2\pi\sqrt{R_2 R_3 C_2 C_3}}$$



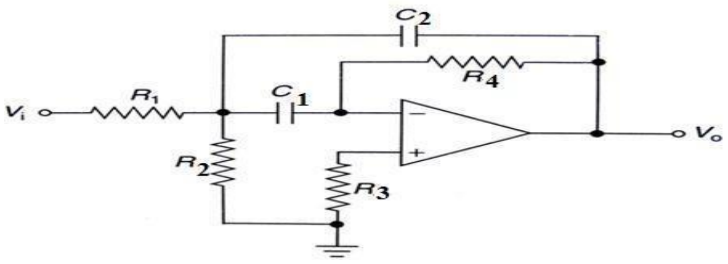
Second order low pass butterworth filter



Second order high pass butter worth filter

Band pass filter:

- ➢ Band pass filter can be formed by cascading the high and low pass filter section in series.
- ➢ These filters are simple to design and offers large bandwidth. Band pass filter is shown below.
- ➢ The cut of frequency is given by



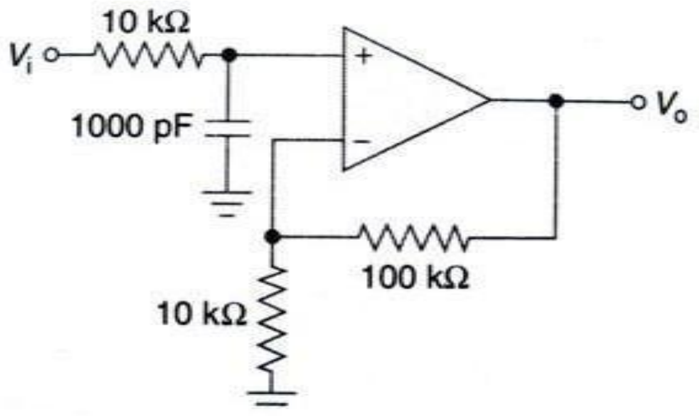
$$F_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

- Q is the quality factor is given by

Prob-5] First order low pass filter shown below determine the cut off frequency and the gain value at 4 times

$$Q = \frac{1}{2}\sqrt{\frac{R_2}{R_1}}$$

cut off frequency and 2 times cut off frequency?



Soln:

F= 1/2πRC = 1/2π x10K x 100x10^{-12}=15.915Khz

Av = 1 + R3/R2 = 1+ 100K/10K = 11

Gain in decibel

Av = 20 log11 = 20.827dB

Gain value at 4 times cut off frequency = 20.827-3x4 =8.827dB

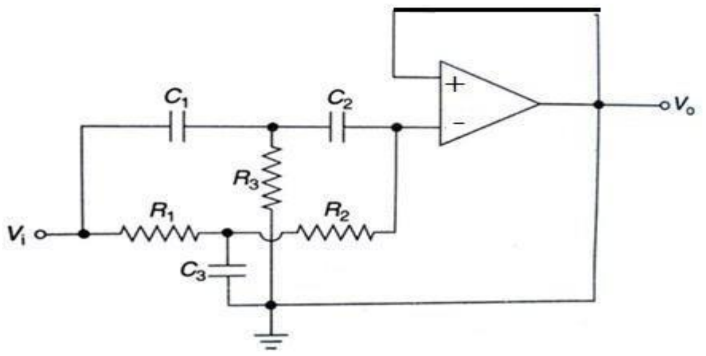
Gain value at 2 times cut off frequency == 20.827-3x2= 14.827dB

Band reject filter:

- These filters are simple to design and have a broad reject frequency range.

- Figure shows the diagram of second order band reject filter.

- It uses twin T network that is connected in series with the non-inverting i/p of the op-amp.



➢ The component value of twin-T network are chosen according to the following equation:

(i) R1 =R2=2 R3 = R/2

(ii) C2 =C3 = 0.5C1

(ii) The cut off frequency is given by F= 1/4πR_3C_3

Problem-1:Design a first order low pass filter with frequency of 2.2KHz and with pass band gain of

2.assume C=0.01µF.

Soln:

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

$$A_V = 1 + \frac{R_f}{R_1}$$

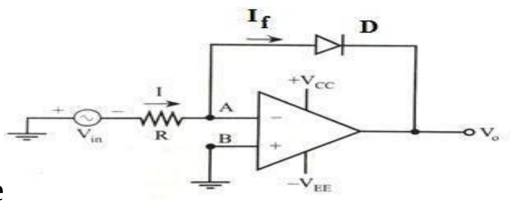
Assignment questions:

1) What is an operational amplifier? Explain with neat diagram application of op-amp.
2) Explain with neat diagram: i) Peak detector circuit ii) Non linear amplifier
3) Explain with neat diagram: i) Current to voltage converter ii) Voltage to current converter and their working.
4) With relevant formulas, neat diagram and waveform explain op - amp Schmitt trigger.
5) With neat figure and relevant wave forms explain the working of relaxation oscillator circuit using op-amp.
6) What is filter? Explain first order low pass and high pass filter with neat diagram
7) With neat diagram and waveform explain Astable multivibrator using 555 timer

Non-linear Amplifier:

❖ A non-linear amplifier circuit gives a non-linear relationship between its input and output signals.
❖ The non-linear amplification can be achieved through by connecting a non-linear device such as PN junction diode in the feedback path.
❖ It is used in AC bridge balance detectors.

$$I = \frac{V_{in}}{R}$$



We have

Let **If** be the current flowing through the diode, the voltage across diode is –Vo
We know diode equation

$$-Vo = \eta V_T \ln \left[\frac{I_f}{I_r}\right]$$

Where: V_T is voltage equivalent temperature.
I_f is diode forward current
$I_r$is diode reverse saturation current

Since current through op-amp is negligible. We get I= I_f

Since I_r, R is constant it can be denoted as Vref

$$Vo = \eta V_T \ln \left[\frac{V_{in}}{V_{ref}}\right]$$

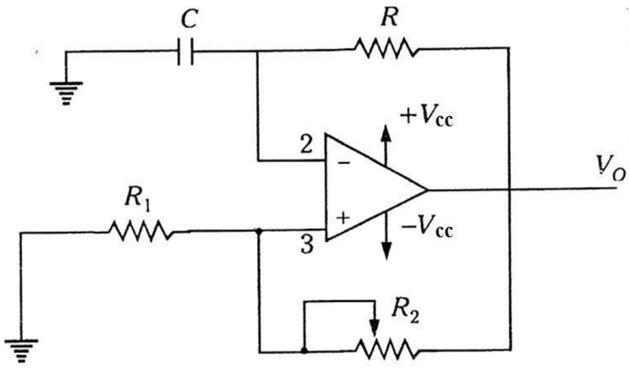
RELAXATION OSCILLATOR

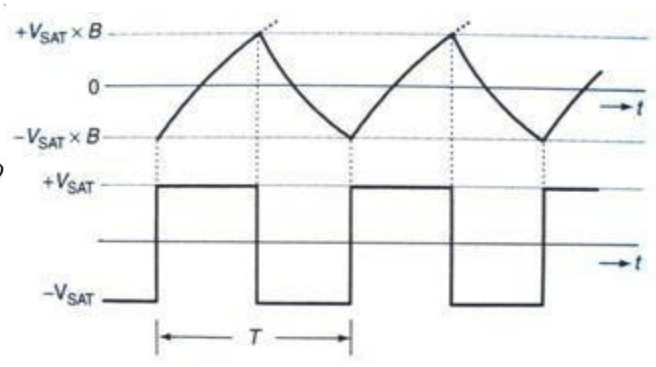
A **relaxation oscillator** is an oscillator ckt that produces a non-sinusoidal output whose time period dependent on the charging time of a capacitor connected as a part of the oscillator ckt. Op-amp is very well suited for constructing of relaxation oscillator ckts that produce a rectangular o/p.

Working: - Let Us assume that the o/p is initially in +ve saturation. As a result voltage at the non-inverting i/p of op-amp is +Vsat*[R1/R1+R2].This forces the o/p to stay in +ve saturation and capacitor C is initially in fully discharge state. Capacitor C starts charging towards +Vsat through R.

The moment the capacitor voltage exceeds the voltage appearing at the non-inverting i/p of op-amp ,the output switches to –Vsat, the voltage appearing at the non-inverting i/p of op-amp also changes to -Vsat*[R1/R1+R2] The capacitor startsdischarging ,it begins to discharge towards –Vsat.

As soon as it becomes more negative than the negative threshold appearing at the non-inverting i/p of op-amp. The output switches back to +Vsat. The cycle repeats thereafter. The output is rectangular wave.





The upper and lower threshold voltage is given by

$$V_{UT} = \frac{R_1}{R_1 + R_2} V_{sat}$$

$$V_{LT} = \frac{R_1}{R_1 + R_2} (-V_{sat})$$

the time taken for one oscillation can be given as,

$$T = 2R_f C\, l\text{n}\left[\frac{+V_{sat} - V_{LT}}{+V_{sat} - V_{UT}}\right]$$

$$f_o = \frac{1}{T}$$

where $f_o \rightarrow$ frequency of oscillation

By substituting the vlaue of upper and lower threshold voltage we get

$$T = 2R_f C\, l\text{n}\left[\frac{+V_{sat} - \left\{(R_1 \times -V_{sat}) / R_1 + R_2\right\}}{+V_{sat} - \left\{(R_1 \times V_{sat}) / R_1 + R_2\right\}}\right]$$

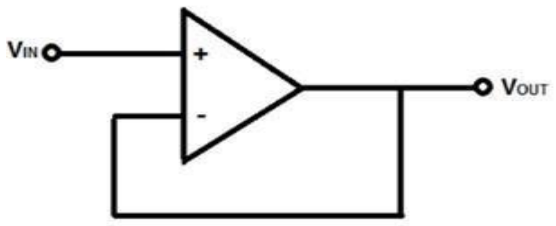
$$\therefore T = 2R_f C\, l\text{n}\left[\frac{+V_{sat}\left[1 + \frac{R_1}{R_1 + R_2}\right]}{+V_{sat}\left[1 - \frac{R_1}{R_1 + R_2}\right]}\right]$$

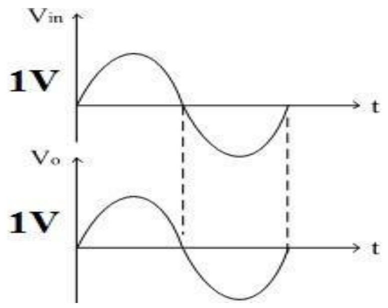
$$\boxed{T = 2R_f C\, l\text{n}\left[\frac{2R_1 + R_2}{R_2}\right]}$$

Above equation gives the total time required for one oscillation.

Meghana Sambare R

Voltage Follower:





$$\frac{V_o}{V_i} = \frac{1V}{1V}$$

$$A_v = 1V$$

- In Voltage follower output voltage is equivalent to the input voltage.
- Op-amp circuit does not provide any amplification.
- Thus, voltage gain is equal to 1.
- The other names of voltage follower are Isolation Amplifier and Unity-Gain Amplifier.

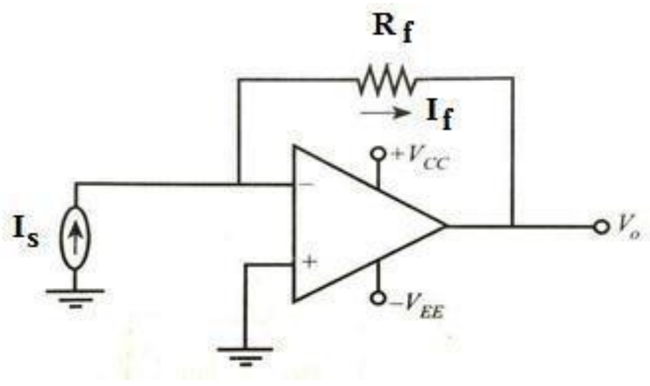
Current to voltage converter (I to V):

- This circuit produces output voltage proportional to the input current.
- The input current I_S is applied to the inverting terminal of op-amp and op-amp output produced is $V_o = R_f \times I_S$
- I to V converter is also called as trans-resistance amplifier.

Voltageto Current converter (V to I):

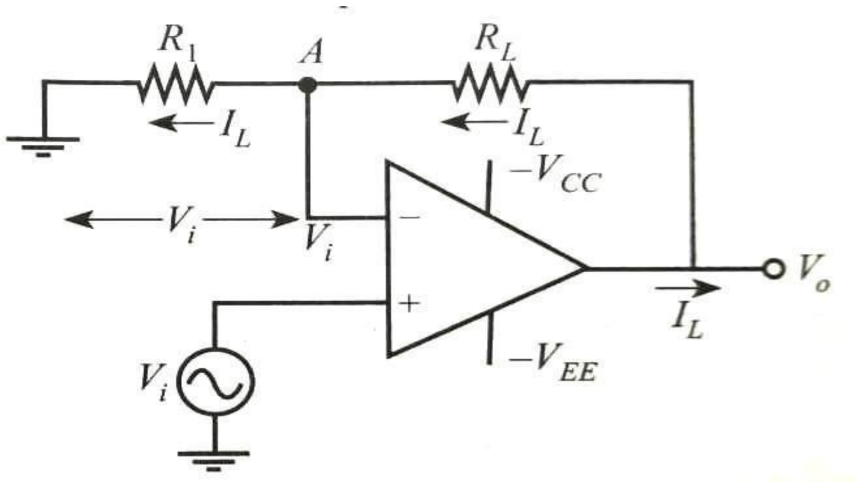
Voltage to Current converter circuit can be two types

(iii) Voltage to Current converter with floating load



Current $I_S = -\frac{V_o}{R_f}$

$V_o = I_S \times R_f$

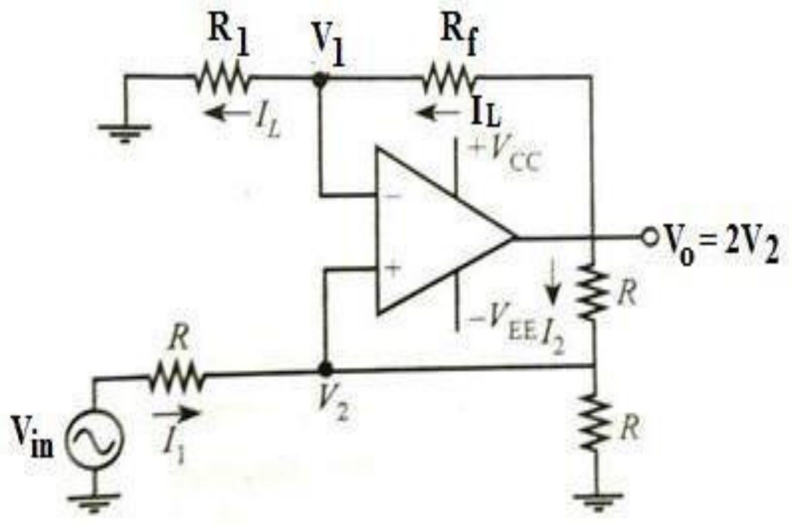


Since voltage at mode A is V_i

$$\therefore V_i = I_L R_1 \qquad \text{or } I_L = V_i / R_1$$

i.e., the input voltage V_i is converted into an output current of which is equal to V_i / R_l.

vi. Voltage to Current converter with grounded load



The current through the load can be calculated as follows.

$$I_1 = \frac{V_{in} - V_2}{R}; \quad I_2 = \frac{V_o - V_2}{R}$$

The load current is given by $I_L = I_1 + I_2$

$$I_L = \frac{V_{in} - V_2}{R} + \frac{V_o - V_2}{R} = \frac{V_{in} + V_o - 2V_2}{R}$$

WKT $\boxed{A_v = 1 + \frac{R}{R} = 2}$ So, $V_o = 2V_2$

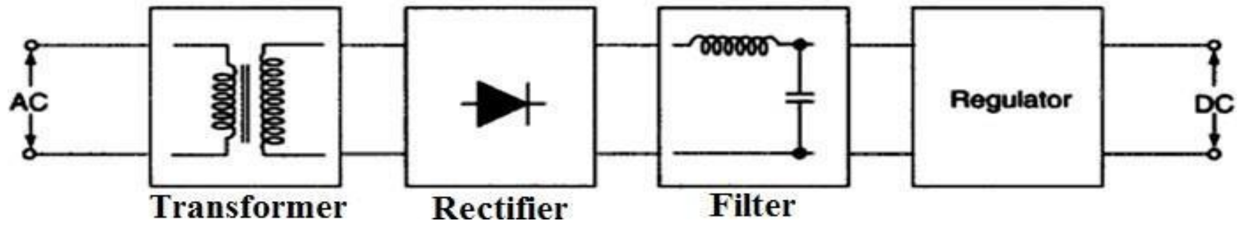
$$\therefore I_L = \frac{V_{in} + 2V_2 - 2V_2}{R}$$

$$\boxed{I_L = \frac{V_{in}}{R}}$$

Thus current I_L is proportional to voltage at the input to applied

Voltage Regulators

Block diagram of regulated power supply:



Mains Transformer: it is required to step up or step down the voltage which is available at its input.

Rectifier circuit: it converts the AC voltage into DC voltage. In practice diode rectifiers are used for example half wave or bridge rectifier.

Mains Transformer: it is required to step up or step down the voltage which is available at its input.

Rectifier circuit: it converts the AC voltage into DC voltage. In practice diode rectifiers are used for example half wave or bridge rectifier.

Filter circuit: rectifier voltage always has some AC content is known as ripple. The filter circuit removes the ripple of rectifier voltage. In order to obtain the pure DC voltage filters circuits are used. Various types of filter such as C-section, L-section and π-section filters are commonly used.

Regulation circuit: it is used to keep the output voltage of a power supply nearly constant under varying input voltage and varying load condition. Commonly used regulator circuits include emitter follower regulator, shunt regulator etc.

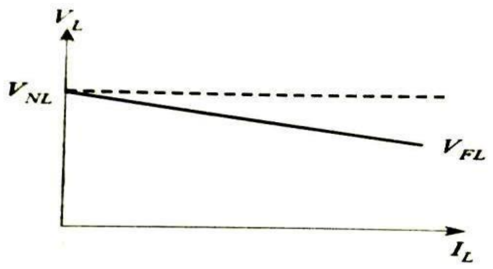
Regulated Power supply parameters:

Line regulation:

Line regulation is defined as the change in regulated o/p voltage for a given change in i/p voltage. Practically it should be small as possible for better performance.

Load Regulation: is defined as change in regulated o/p voltage when the load current is changed from zero(no load) to maximum value(Full load).Practically it should be small as possible for better performance.

$$\text{percentage load regulation} = \left[\frac{V_{NL} - V_{FL}}{V_{FL}}\right] \times 100$$



change in input voltage where the load current and temperature assumed as constant. Practically it should be small as possible for better performance.

Temperature stability factor: temperature stability of power supply is determined by temperature coefficients of various temperature sensitive devices. It is better to choose low temperature coefficients devices to keep output voltage constant.

Ripple rejection factor: The output of the rectifier and filter consists of ripples. Ripple rejection factor shows how effectively the regulator rejects the ripples and attenuates it from input to output. It is expressed in decibel (dB). It is negative value.

$$\text{Ripple rejection in dB} = 20\log\left[\frac{\text{output ripple}}{\text{input ripple}}\right]$$

Note: Rectifier converts AC to DC, rectifier output has some AC content and it is called as ripple.

Prob-1) Determine the o/p ripple of a regulated power supply which provides a ripple rejection of -80dB and aripple voltage in the unregulated i/p were 2V.
Soln:

Ripple rejection in dB is given by

$$20\log\left[\frac{\text{output ripple}}{\text{input ripple}}\right] = -80\text{ dB}$$

$$\therefore \log\left[\frac{\text{output ripple}}{\text{input ripple}}\right] = -4$$

$$\frac{\text{output ripple}}{\text{input ripple}} = \text{Antilog}(-4)$$

$$\frac{\text{output ripple}}{\text{input ripple}} = 10^{-4}$$

$$\therefore \text{output ripple} = 2 \times 10^{-4}\text{ V} = 0.2\text{mV}$$

Prob-2) Two power supplies A and B are available in the market. Power supply A has no load and full load voltages of 40V and 30Vand power supply B has 30V and 28V. Determine which power supply is better.

Soln:
Power supply A: V_{NL}=40V V_{FL}=30V

$$\text{percentage load regulation} = \left[\frac{V_{NL} - V_{FL}}{V_{FL}}\right] \times 100$$

% Load regulation = $[\frac{40-30}{30}]$x 100

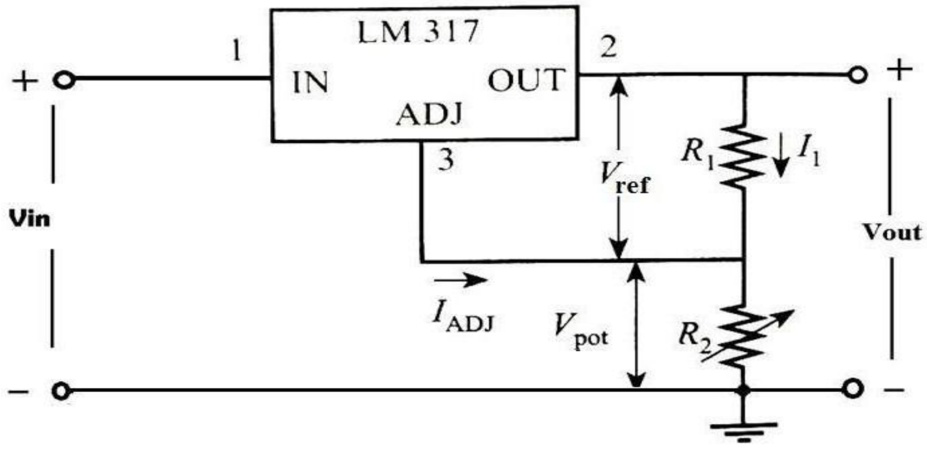
% Load regulation = 33.33%

Power supply B: V_{NL}=30V V_{FL}=28V
% Load regulation = $[\frac{30-28}{28}]$ x 100

% Load regulation = 7.14%

Adjustable voltage regulator:

- Its output voltage can be adjustable over a range.
- There is positive and negative adjustable voltage regulator.
- LM 317 is a positive adjustable voltage regulator and whose output voltage can be adjustable from 1.21V to 57V.
- LM 337 is negative adjustable voltage regulator. Operation of LM 337 is similar to LM 317 but the difference is polarity of regulator voltage.
- The resistor R1 and R2 determines the output voltage Vout.
- The resistor R2 is adjusted to get the output voltage range between 1.21V to 57V.



- The output voltage is given by

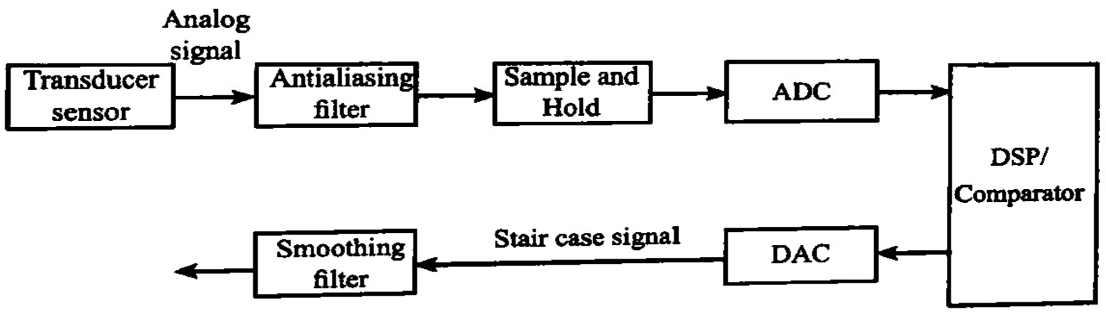
$$V_{out} = V_{ref} * \frac{(R_1+R_2)}{R_1} + I_{Adj} * R_2$$

Advantages:
★ Better load and line regulation
★ Improved performance and overload protection
★ The output voltage can be adjustable from 1.21V to 57V.

Assignment questions:
1. Explain with diagram regulated power supply?
2. Explain the performance parameters of power supply.
3. Explain adjustable voltage regulator with neat diagram? Mention its advantages.

D TO A & A TO D CONVERTERS

The digital system such as computers also need to communicate with physical processes and with people through analog signals. So there is a need of digital to analog converters.



Typical A/D and D/A Converter

Naturally available signal is analog in form, and may be obtained from *sensor or transducer*. This analog signal is band limited by *anti-aliasing filter*. The signal is then *sampled* at a frequency rate, more than twice the maximum frequency of the band limited signal. The sampled signal is held constant by *hold*

circuit while conversion is taking place. The discrete signal from the sample and hold circuit is fed to analog to digital converter (ADC). The ADC gives digital output signal that can be easily processed, stored, or transmitted by digital systems or computer system. The digital signal is converted back to by digital to analog converter (DAC). The output of DAC is usually stair-case waveform, which is passed through smoothing filter to reduce the quantization noise. The diagram shown in the above Figure can be used in the applications such as digital signal processing, digital audio mixing, music and video synthesis, data acquisition, pulse code modulation, and microprocessor instrumentation.

Analog to Digital and Digital to Analog converters:

- Typical Analog to Digital and Digital to Analog converters shown below.
- Analog signal is obtained from trasnducer.
- The high frequency components of analog signal are filtered by anti aliasing filter.
- Output of aliasing filter is fed to the sample and hold circuit.
- The signal is sampled and held constant by hold circuit during conversion.
- The signal from sample and hold circuit is fed to the ADC. The ADC gives a digital output signal.
- The digital signal is converted back to analog signal by DAC.
- The output of DAC is passed through smoothing filter to reduce the noise.

DAC techniques:

DAC converts the digital data into its equialent analog value. Below shows n-bit DAC.

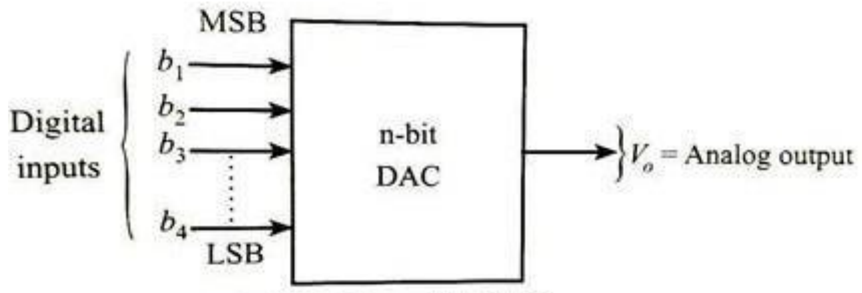


FIGURE : n-bit DAC

If n=4 indicates it is 4 bit DAC

Performance parameters of DAC:

(i) Resolution:

Resolution is the number of various analog output value that is provided by a DAC. For n-bit DAC

$$\textbf{Resolution} = 2^n$$

Resolution is also defined as the ratio of change in output voltage resulting from change of 1 LSB at digital input.

Where**V_oFS**= Full scale output voltage.

If we know resolution we can obtain input and output relation for DAC is

$$\textbf{Vo = Resolution x b}$$

Where b is the decimal Value of Digital input.

(ii) Accuracy:

Comparision of actual output with expected output is called as accuracy and is given by

$$\text{Accuracy} = \frac{V_{oFS}}{(2^n-1)2}$$

(iii) Setting Time:

Setting Time is the time required for a DAC output to settle within ±1/2 LSB of its final value for a

$$\text{Resolution} = \frac{V_{oFS}}{2^n-1}$$

given digital input.

(iv) Stability:

Performance of conveter changes with parameter such as temperature, power supply variation. Performance of conveter must be stable for these changes.

Binary weighted DAC:

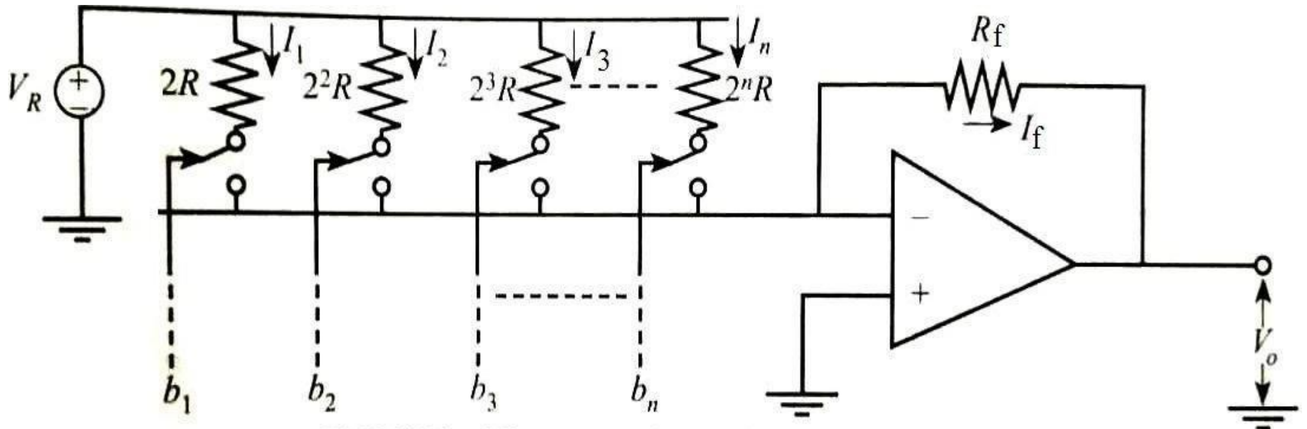


FIGURE : Binary weighted resistor DAC

- Binary weighted resistor DAC uses n-electronic switches controlled by the binary inputs b1, b2, ... bn.
- If the binary input is high (logic 1) then the switch connects the resistance to reference voltage V_R and current flows through the circuit.

$$\textbf{For ON switch current } I = \frac{V_R}{R}$$

- If the binary input is low (logic 0) then the switch disconnects the resistance from reference voltage V_R and no current flows through the circuit.

For OFF switch current, I = 0

$$\begin{aligned} I_f &= I_1 + I_2 + I_3 + \cdots + I_n \\ &= \frac{V_R}{2^1 R} b_1 + \frac{V_R}{2^2 R} b_2 + \frac{V_R}{2^3 R} b_3 + \cdots + \frac{V_R}{2^n R} b_n \\ &= \frac{V_R}{R}\left[b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots + b_n 2^{-n} \right] \end{aligned}$$

The output voltage across R_f is

$$V_o = -I_f R_f$$

Substituting for I_f, $V_o = \frac{-V_R}{R} R_f \left[b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots + b_n 2^{-n} \right]$

If $\boxed{R_f = R}$,

$$V_o = -V_R \left[b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots + b_n 2^{-n} \right]$$

The Total current flows through Rf is written as:

Drawbacks:

1. Each resistor in the network has different value. This makes the circuit more expensive and unattractive.
2. Larger ranges of resistor values are required. Example for 8bit DAC, largest resistor is 128 times the smallest one.

R-2R ladder type DAC:

$$I_1 = V_R / 2R$$

$$I_2 = \frac{V_R / 2}{2R} = \frac{V_R}{4R}$$

$$\text{also } I_3 = \frac{V_R / 4}{2R} = \frac{V_R}{8R}$$

$$\text{and } I_n = \frac{V_R / 2^n - 1}{2R}$$

But WKT

$$V_o = -I_f R_f$$

$$V_o = -R_f (I_1 + I_2 + \cdots + I_n)$$

$$= -R_f \left[\frac{V_R}{2R} b_1 + \frac{V_R}{4R} b_2 + \cdots + \frac{V_R}{2^n R} b_n \right]$$

- R-2R ladder network DAC uses only two values R and 2R.
- Each binary bit connects the switch to either ground or inverting input of op-amp.
- R-2R ladder network DAC shown below

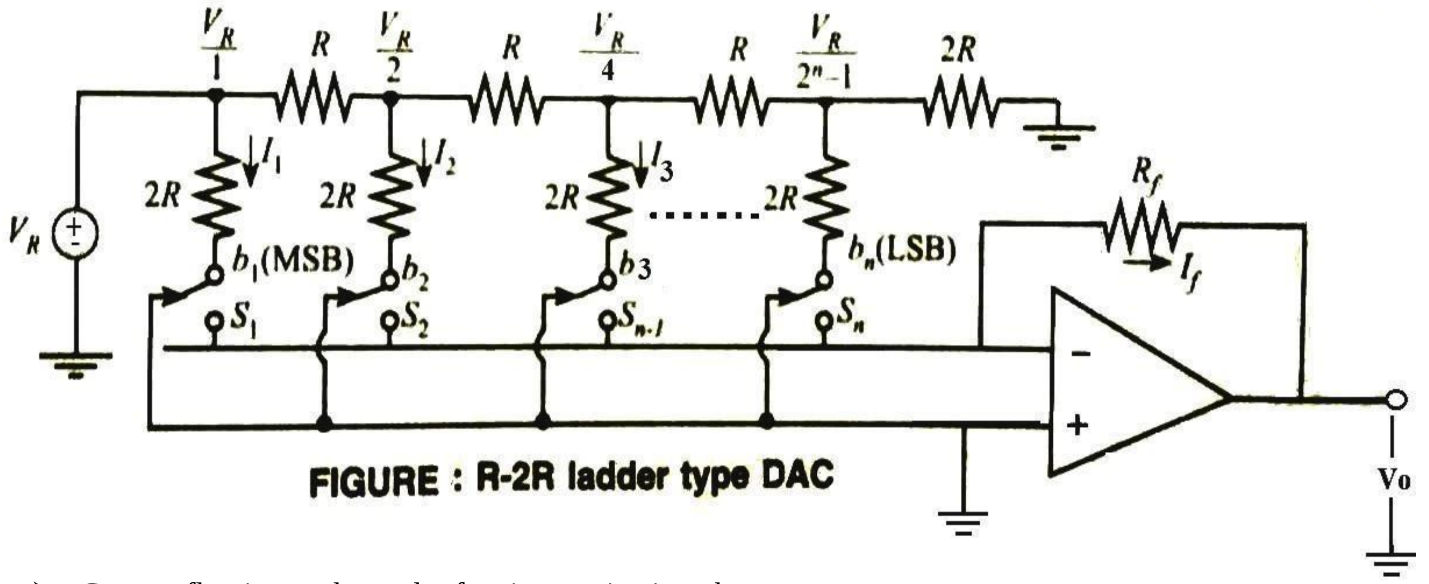


FIGURE : R-2R ladder type DAC

- Current flowing to the each of resistance is given by:

R-2R ladder type DAC:

- R-2R ladder network DAC uses only two values R and 2R.
- Each binary bit connects the switch to either ground or inverting input of op-amp.
- R-2R ladder network DAC shown below

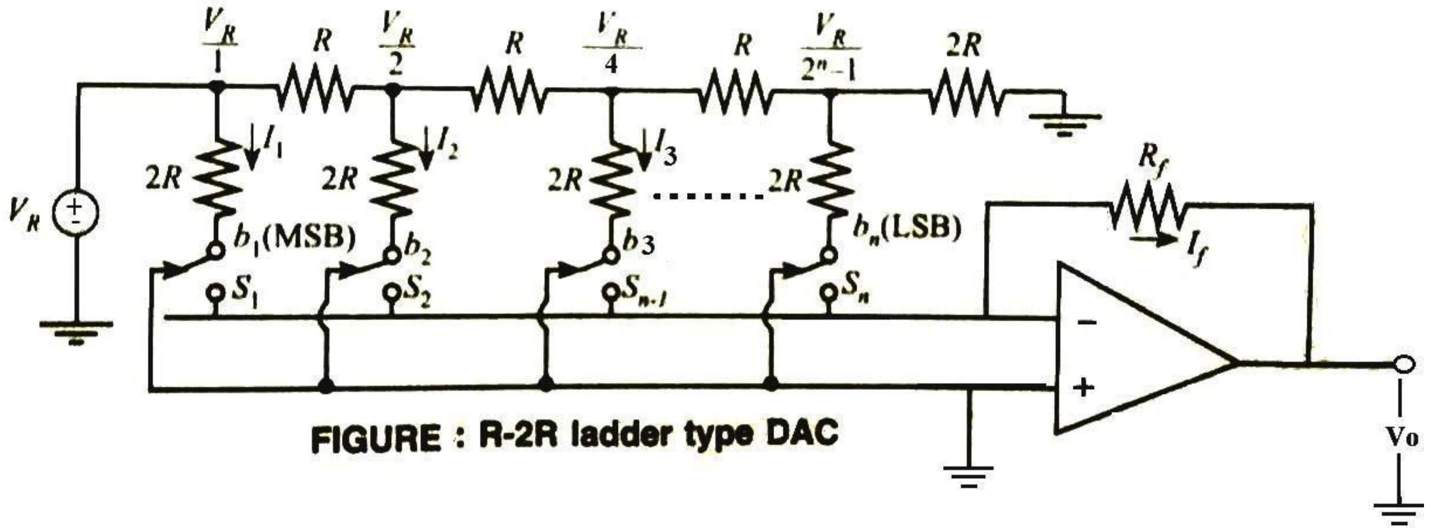


FIGURE : R-2R ladder type DAC

- Current flowing to the each of resistance is given by:

$$I_1 = V_R / 2R$$

$$I_2 = \frac{V_R/2}{2R} = \frac{V_R}{4R}$$

$$\text{also } I_3 = \frac{V_R/4}{2R} = \frac{V_R}{8R}$$

$$\text{and } I_n = \frac{V_R/2^n - 1}{2R}$$

But WKT

$$V_o = -I_f R_f$$

$$V_o = -R_f(I_1 + I_2 + \cdots + I_n)$$

$$= -R_f\left[\frac{V_R}{2R}.b_1 + \frac{V_R}{4R}.b_2 + \cdots + \frac{V_R}{2^n R}.b_n\right]$$

$$= \frac{-V_R}{R} R_f \left(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}\right)$$

When $\boxed{R_f = R}$ V_O can be

$$\boxed{V_o = -V_R\left(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}\right)}$$

$$= \frac{-V_R}{R} R_f \left(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}\right)$$

When $\boxed{R_f = R}$ V_O can be

$$\boxed{V_o = -V_R \left(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_n 2^{-n}\right)}$$

Advantages:

1. It uses only two types of resistors and accurate value of R-2R can be designed.
2. Binary input length can be increased by adding more R-2R value

<u>ADC A ADC Analog to Digital converter</u>

- ❖ It converts analog input voltage to equivalent digital output.
- ❖ ADC is provided with two control pins start and EOC(end of conversion)

ADC techniques:

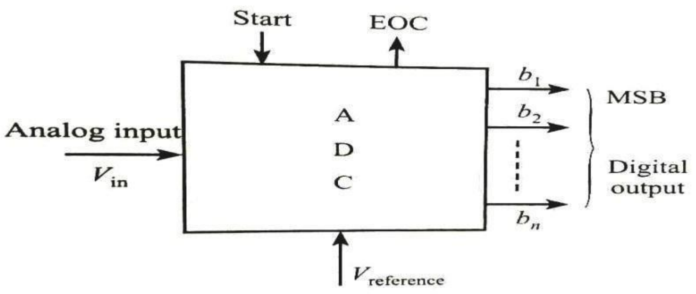


Figure: **Functional diagram of ADC**

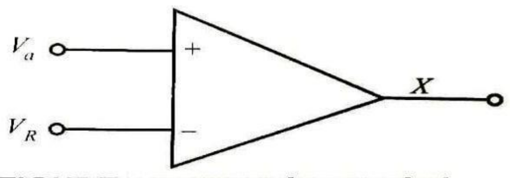


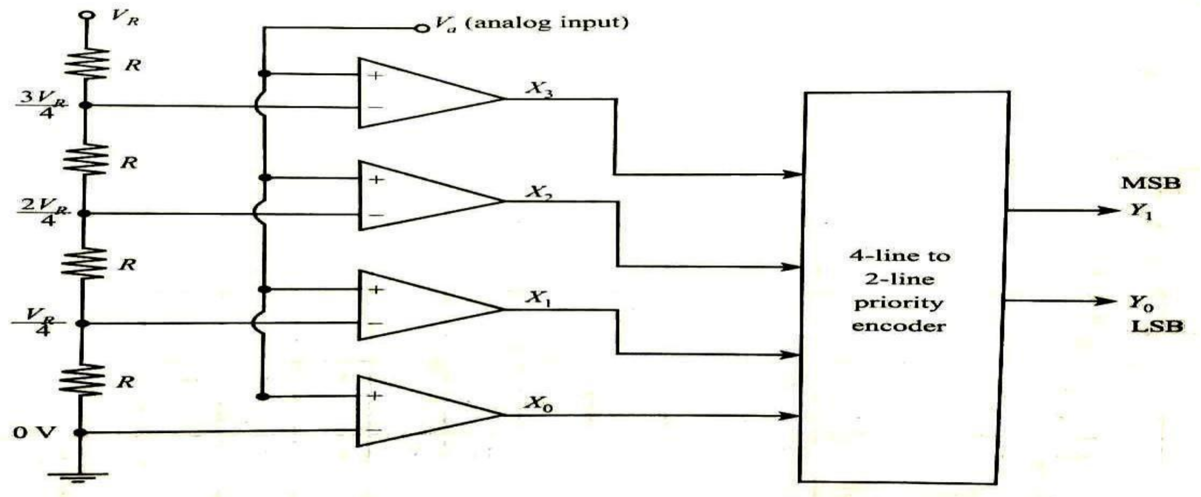
FIGURE : comparator symbol

Input voltage	Logic output (X)
$V_a > V_R$	X = 1
$V_a < V_R$	X = 0
$V_a = V_R$	Previous value

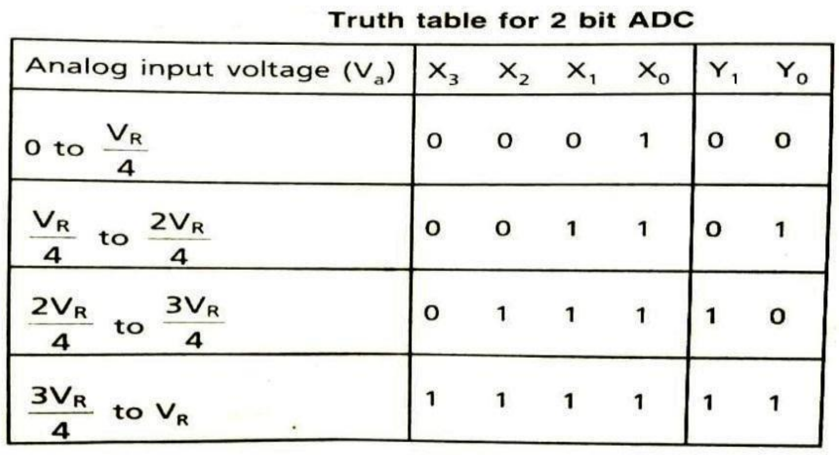
TABLE : Truth table

- Comparator circuit is a two inputs and one output building block that produces low or high output.
- One of the inputs to the comparator is reference voltage and other input is input voltage that needs to be compared with reference voltage.
- If the analog input signal exceeds the reference voltage to any comparator that comparator turns ON.

(i) 2-bit Flash type ADC or simultaneous type ADC



- The reference voltages are 0, $V_R/4$, $V_R/2$ and $3V_R/4$.
- If X_O should be high, the input must be between 0 to $V_R/4$.
- All the comparator output is high when the input is between $3V_R/4$ to V_R
- Priority encoder represents the binary number starting from zero.

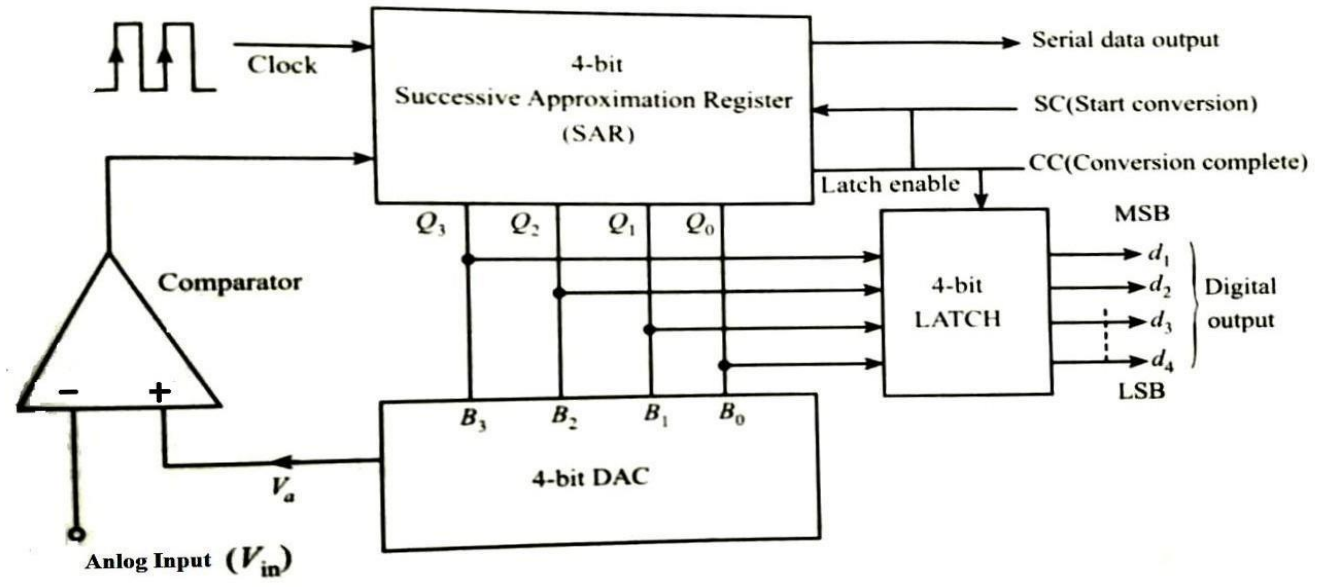


Truth table for 2 bit ADC

Analog input voltage (V_a)	X_3	X_2	X_1	X_0	Y_1	Y_0
0 to $\frac{V_R}{4}$	0	0	0	1	0	0
$\frac{V_R}{4}$ to $\frac{2V_R}{4}$	0	0	1	1	0	1
$\frac{2V_R}{4}$ to $\frac{3V_R}{4}$	0	1	1	1	1	0
$\frac{3V_R}{4}$ to V_R	1	1	1	1	1	1

Successive approximation type ADC:

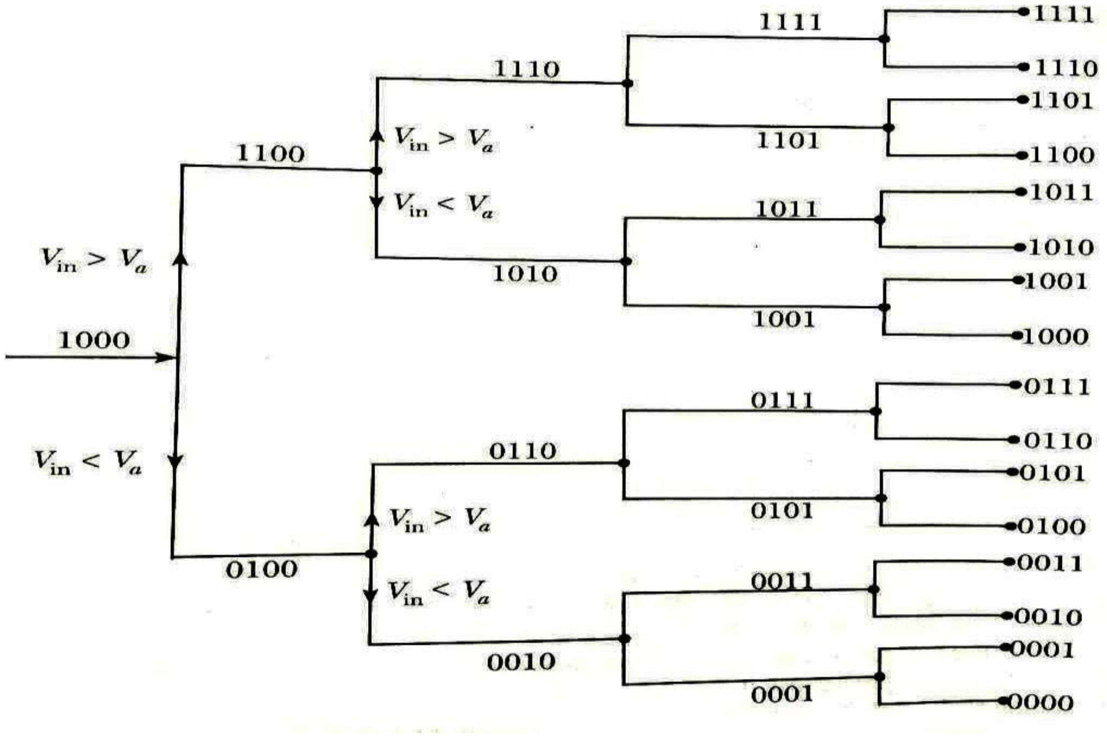
- Successive approximation ADC consists of successive approximation register (SAR), the output of



which is connected to DAC as well as latch.

1. The analog input signal Vin is compared with the analog output signal Va of the DAC.
2. The output of comparator fed back into SAR.
3. The control logic inside SAR adjusts the digital output.
4. The operation is understood with the help of code tree is shown below.
5. The MSB bit is initially set to 1 and remaining three bit set as 000.
6. The digital equivalent voltage Va is compared with the analog input voltage Vin
7. If the analog input voltage is higher than the digital voltage MSB bit is retained as 1 and the second MSB is set to 1.
8. If analog input voltage is lower than digital voltage MSB bit is set 0 and second MSB is set to 1
9. When Vin is equal to Va conversion stops.

For Example: Consider **Vin = 11V >Va = 8V = [1000]**
Since the analog input voltage Vin is higher than the digital voltage Va, the MSB is retained as 1 and the next MSB bit is set to 1 as follows **Va = 12V = [1100]**



Now Vin = 11V <Va = 12V = [1100]
Here now, the analog input voltage Vin is lower than the digital voltage Va. the second MSB is set to 0 and next MSB set to 1 as **Va = 10V = [1010]**

Now again **Vin = 11V >Va = 10V = [1010]**
Again Vin>Va, hence the third MSB is retained to 1 and the last bit is set to 1. The new code word is
Va = 11V = [1011]
Now finally Vin = Va , and the conversion stops.

Problem1: the digital input for 4bit DAC is 0111. Calculate its output voltage by assuming V_{oFS} =15V
Soln: Given n=4 ,digital input is 0111=7(decimal value = b) and V_{oFS} =15V

$$\text{Resolution} = \frac{V_{oFS}}{2^n-1}$$

Resolution = 1V/LSB
Vo = resolution x b = 7V

Problem2: For a 8bit DAC having resolution of 22mv/LSB find the full scale output voltage and output voltage if the input is 10000000.
Soln:

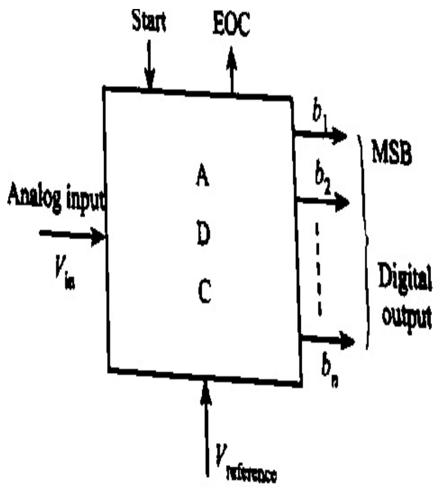
Full scale output voltage $V_{o}FS$ =5.6V and output voltage = 2.8V

Assignment questions:

1. What is DAC? Explain with neat diagram different DAC techniques?
2. What is ADC? Explain with neat diagram different ADC techniques?
3. Explain the performance parameters of a DAC.
4. Mention the disadvantages of binary weighted DAC? Also mention the advantage of R-2R ladder DAC.

A-D CONVERTERS:

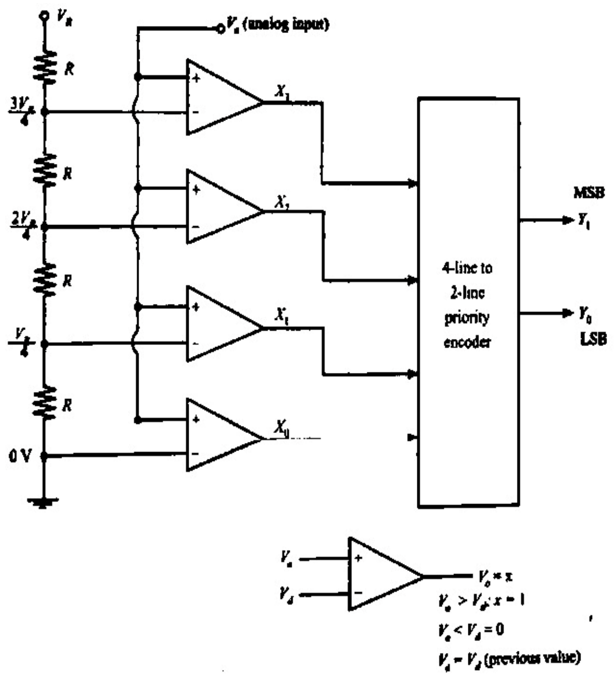
ADC takes the analog signal as input and converts into digital output. The functional diagram of DAC is given below:

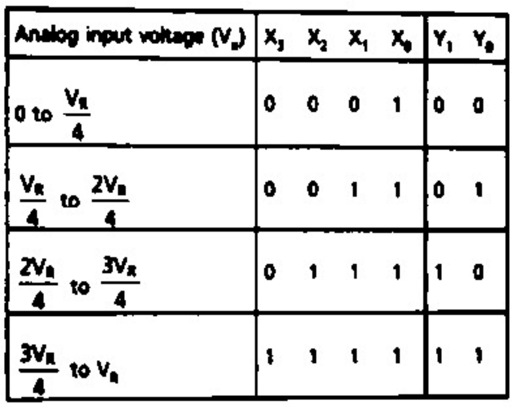


ADC is provided with two control inputs start (input to initiate the conversion) and end of conversion (output to indicate the end of conversion). Direct type ADCs and Integrated type ADCs are the two types of ADCs available.

Flash (Comparator/ Parallel) type ADC: A simple, fast, but most expensive conversion technique.

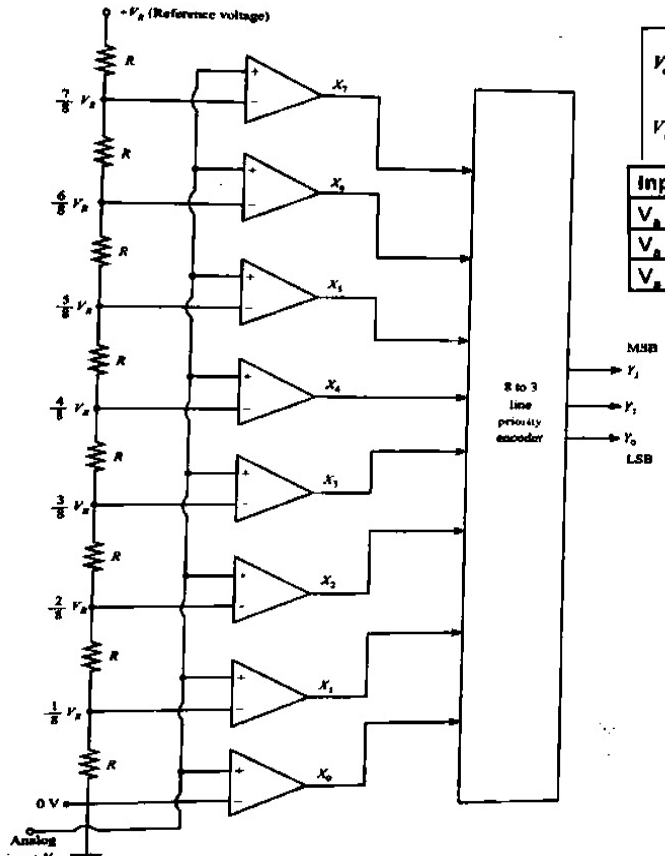
A 2-bit Flash ADC:

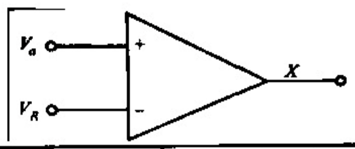


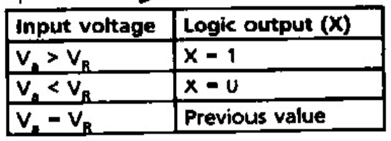


Analog input voltage (V_a)	X_3	X_2	X_1	X_0	Y_1	Y_0
0 to $\frac{V_R}{4}$	0	0	0	1	0	0
$\frac{V_R}{4}$ to $\frac{2V_R}{4}$	0	0	1	1	0	1
$\frac{2V_R}{4}$ to $\frac{3V_R}{4}$	0	1	1	1	1	0
$\frac{3V_R}{4}$ to V_R	1	1	1	1	1	1

A 3-bit Flash ADC:

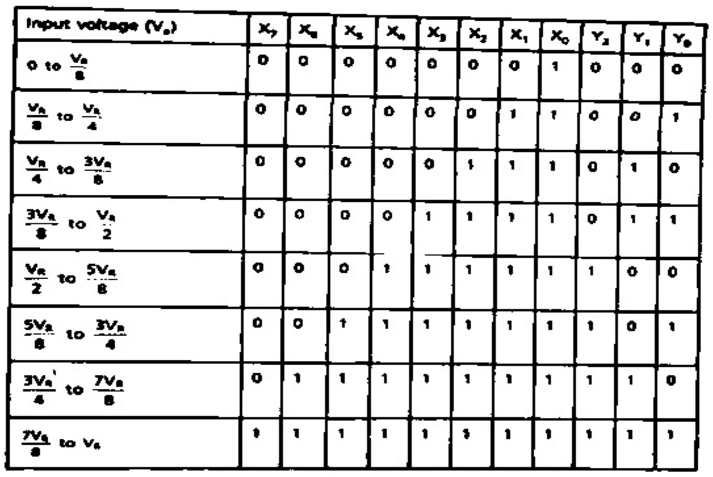






Input voltage	Logic output (X)
$V_a > V_R$	X = 1
$V_a < V_R$	X = 0
$V_a = V_R$	Previous value

The resistive network is to set to equal reference voltages at each node. The comparactor compares set reference value at inverting terminal of Op-Amp with analog output at non-inverting terminal. The truthtable for ADC is given below:



Input voltage (V_a)	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Y_2	Y_1	Y_0
0 to $\frac{V_R}{8}$	0	0	0	0	0	0	0	1	0	0	0
$\frac{V_R}{8}$ to $\frac{V_R}{4}$	0	0	0	0	0	0	1	1	0	0	1
$\frac{V_R}{4}$ to $\frac{3V_R}{8}$	0	0	0	0	0	1	1	1	0	1	0
$\frac{3V_R}{8}$ to $\frac{V_R}{2}$	0	0	0	0	1	1	1	1	0	1	1
$\frac{V_R}{2}$ to $\frac{5V_R}{8}$	0	0	0	1	1	1	1	1	1	0	0
$\frac{5V_R}{8}$ to $\frac{3V_R}{4}$	0	0	1	1	1	1	1	1	1	0	1
$\frac{3V_R}{4}$ to $\frac{7V_R}{8}$	0	1	1	1	1	1	1	1	1	1	0
$\frac{7V_R}{8}$ to V_R	1	1	1	1	1	1	1	1	1	1	1

Advantages:

1. High speed

Disadvantages:

Number of comparators required is almost double for each added bit

Eg.: For 2-bit ADC;

No. of Comparators = 4 (2^2) For 3-bit ADC;

No. of Comparators = 8 (2^3)

Successive Approximation type ADC: The following Figure shows successive approximation ADC

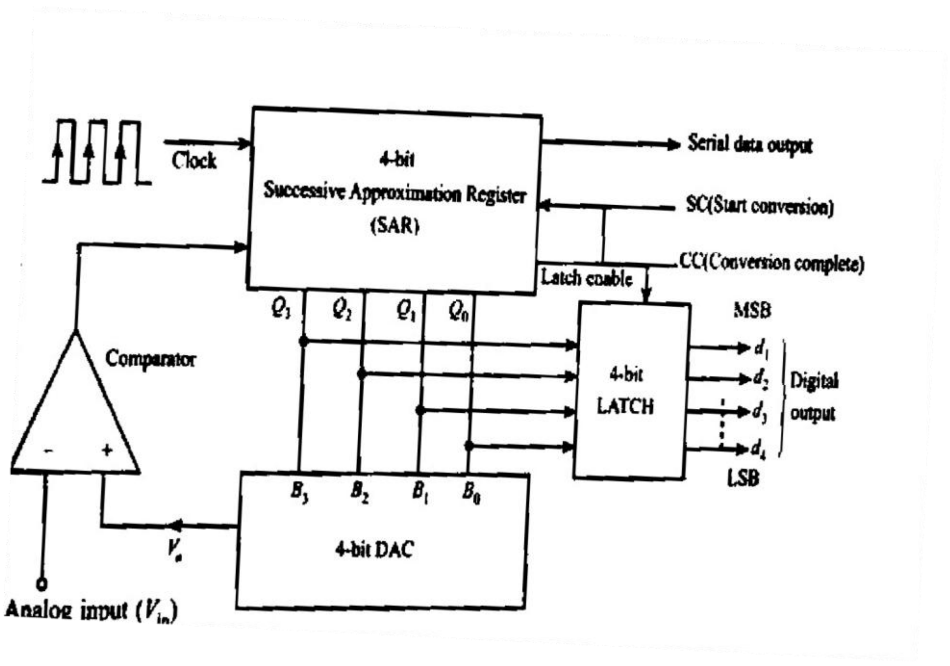
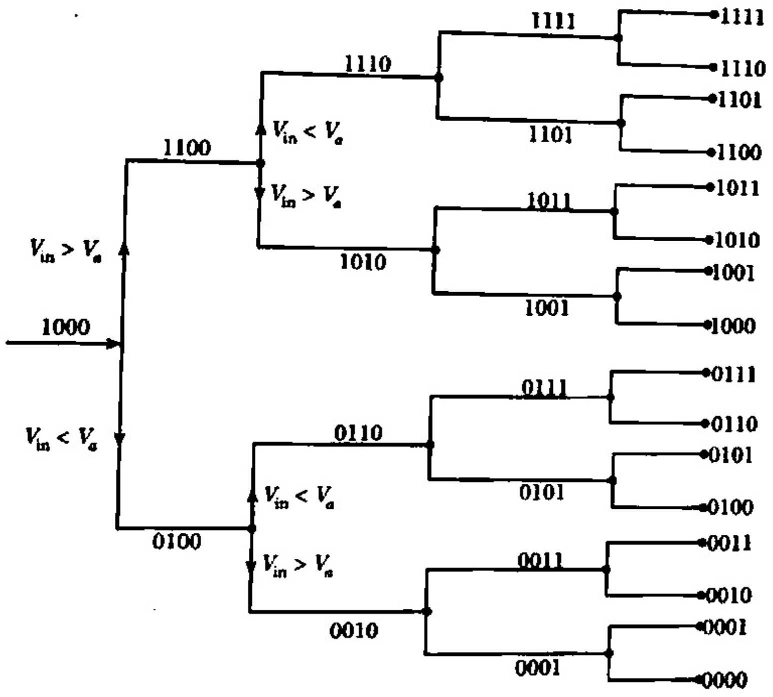


Figure shows a successive approximation register (SAR), the output of which is connected to DAC and output latch circuit. The input signal (Vin) is compared with the analog output signal (Va) of the DAC. Output of the comparator is feedback into SAR. The control logic inside SAR adjusts its digital output; until it is equal to the analog input signal. The operation could be understood by the code tree given below.

At the start of conversion cycle, start conversion terminal is made high. On the first clock pulse, the output of the SAR is made 1000. The DAC produces an analog voltage (Va) proportional to 1000. This analog voltage is compared with input analog signal (Vin). If Vin > Va, the comparator output will be high and SAR keeps Q3 high. On the other hand, if Vin < Va, then the comparator output becomes low and SAR resets Q3 to low. If Vin > Va, SAR follows the upward path in code tree and if Vin < Va, SAR follows downward path.



The conversion time for *n*-bit successive approximation ADC is (*n* + *2*) clock periods.

Advantages:

1. Considerably good speed
2. Good resolution.

ANALOG AND DIGITAL ELECTRONICS LABORATORY

Laboratory Programs:
PART A (Analog Electronic Circuits)

1. Design an astable multivibrator circuit for three cases of duty cycle (50%, <50% and >50%) using NE 555 timer IC. Simulate the same for any one duty cycle.
2. Using µa 741 Op-amp, design a 1 kHz Relaxation Oscillator with 50% duty cycle. And simulate the same.
3. Using µa 741 op-amp, design a window comparator for any given UTP and LTP. And simulate the same.

PART B (Digital Electronic Circuits)

4. Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basicgates. And implement the same in HDL.
5. Given a 4-variable logic expression, simplify it using appropriate technique and realize the simplified logic expression using 8:1 multiplexer IC. And implement the same in HDL.
6. Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table. And implement the same in HDL.
7. Design and implement code converter I)Binary to Gray (II) Gray to Binary Code using basic gates.
8. Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.
9. Design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and demonstrate on 7-segment display (using IC-7447).

Conduct of Practical Examination:

- Experiment distribution
- For laboratories having only one part: Students are allowed to pick one experiment fromthe lot with equal opportunity.
- For laboratories having PART A and PART B: Students are allowed to pick on experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero ofthe changed part only.
- Marks Distribution (Coursed to change in accordance with university regulations)

a) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 =100 Marks
b) For laboratories having PART A and PART B
i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
ii. Part B – Procedure + Execution + Viva = 9+42 + 9 = 60 Marks

Module 1

Experiment: 1

ASTABLE MULTIVIBRATOR

Experiment No.1: Design an astable multivibrator circuit for three cases of duty cycle (50%, <50% and >50%) using NE 555 timer IC. Simulate the same for any one duty cycle.

Description:
Multivibrator is a form of oscillator, which has a non-sinusoidal output. The output waveform is rectangular. The multi vibrators are classified as
i) **Astable or free running multivibrator:** It alternates automatically between two states (low and high for a rectangular output) and remains in each state for a time dependent upon the circuit constants. It is just an oscillator as it requires no external pulse for its operation.

ii) **Monostable or one shot multivibrators:** It has one stable state and one quasi stable. The application of an input pulse triggers the circuit time constants and the output goes to the quazi stable state, after a period of time determined by the time constant, the circuit returns to its initial stable state. The process is repeated upon the application of each trigger pulse.

iii) **Bistable Multivibrators:** It has both stable states. It requires the application of an external triggering pulse to change the output from one state to other. After the output has changed its state, it remains in that state until the application of next trigger pulse. Flip flop is anexample.

Components required: 555 Timer IC, Resistors of 3.3KΩ, 6.8KΩ, Capacitors of C=0.1µF, C=0.01 µF, digital trainer kit(used to give +5v power supply to 555 IC),CRO.

Astable-multivibratorin which ckt is not stable in either state, keeps on switching between 2 states and it does not need any external triggering.It is also called as free running multivibrator.

★ It is used in security alarms, pulse and tone generation.

★ If the T_{on}=T_{off} ,it is called symmetrical Astable –multivibrator

★ if T_{on} not equal to T_{off} ,its an unsymmetrical Astable- multivibrator.

Components required:IC-555, resistor 6.8KΩ, 3.3KΩ, Capacitors 0.1µF (104) ,0.01µF (103), CRO probe and connecting wires.

Circuit diagram and Waveform:

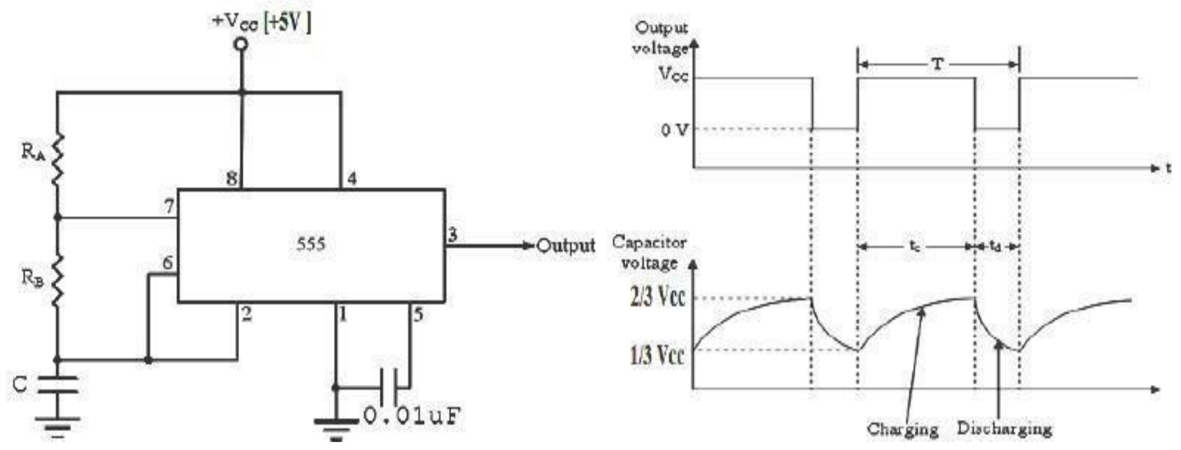


Figure 1: Astable Multivibrator circuit diagram and waveforms

DESIGN:

Given: F=1Khz, D=0.75 and C=0.1µF

T_{ON}=0.693(R_A+ R_B)C

T_{OFF}=0.693(R_B)C

T=T_{ON} + T_{OFF} 0.693(R_A + 2R_B)C…..............(i)

F=1/T ....(ii)

1K=1.44/(R_A + 2R_B)C = R_A + 2R_B =14.4KΩ.................(iii)

D = T_{ON}/(T_{ON} + T_{OFF})➔0.693(R_A+R_B)C

0.693(R_A+2R_B)C

Case 1: 50% duty cycle

Let Frequency =1kHz, T=1ms, C=0.1 µF

T_{ON} = T_{OFF}=0.5ms

For R_A, 0.5ms= 0.693 * R_A *0.1 *10^6

R_A =7.2 kΩ=R_B

Case 2: >50% duty cycle

let Duty cycle be 75%

Let Frequency =1kHz, T=1ms, C=0.1 µF

T_{ON} = 0.75ms

T_{OFF}= 0.25ms

For R_A, 0.75ms= 0.693 * R_A *0.1*10^{-6}

R_A=10 kΩ

ForR_B, 0.25ms= 0.693 * R_A *0.1 *10^{-6}

R_B =3.6 kΩ

Case 3: <50% duty cycle,

let Duty cycle be 25%

Let Frequency =1kHz, T=1ms, C=0.1 µF

T_{ON} = 0.25ms

T_{OFF}= 0.75ms

For R_A, 0.25ms= 0.693 * R_A *0.1*10^{-6}

R_A=3.6 kΩ

ForR_B, 0.75ms= 0.693 * R_A *0.1 *10^{-6}

R_B =10 kΩ

Working:-

- Initially the o\p is high,capacitor C starts charging towards Vcc through R_A and R_B

1. s

soon as the voltage across the capacitor equals 2\3 Vcc, the o\p switches to low state.

2. Now capacitor C discharge through R_B .
3. When voltage across C equals 1\3 Vcc , the o\p goes to high. Then the cycle repeats.
4. The time during which the capacitor charges from 1\3 to 2\3 Vcc is equal to the time the o\p remains highand is given by **$T_c=0.693(R_A+R_B)C$**
5. Similarly the time during which the capacitor discharges from 2\3Vcc to 1\3 Vcc is equal to the time the o\pis low and is given by **$T_d=0.693(R_B)C$**
6. Thus the total time period of the o\p wave form is
   a. **$T=T_c+T_d=0.693(R_A+2R_B)C$** Where R_A and R_B are in ohms and C in Farads.

PROCEDURE:
1. Before making the connections, check the components.
2. Make the connections as shown in figure and switch on the power supply.
3. Observe the capacitor voltage waveform at 6th pin of 555 timer on CRO.
4. Observe the output waveform at 3rd pin of 555 timer on CRO
5. Note down the amplitude levels, time period and hence calculate duty cycle.

Output Waveforms

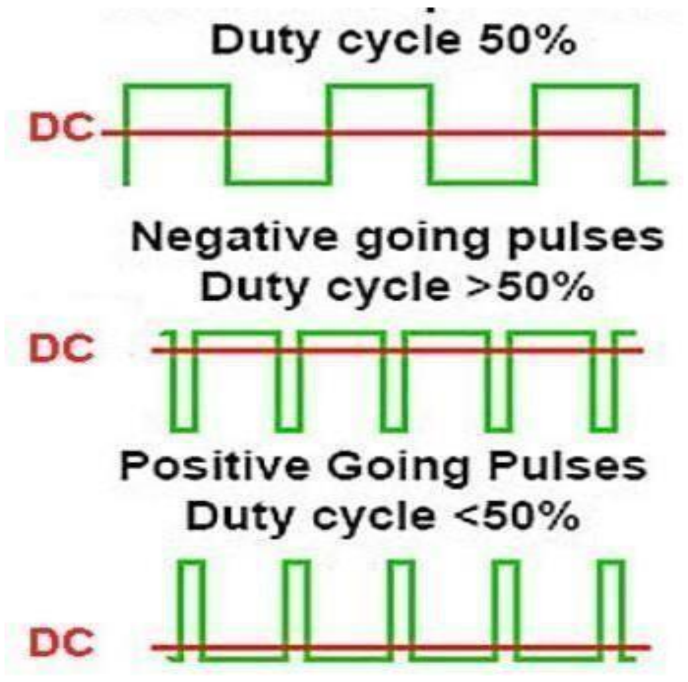


Figure 2: Astablemultivibrator output waveforms

Result:

Note:
Each division in oscilloscope is 0.2
Time=no of div in x-axis x time base
Amplitude= no of div in y-axis x volt/div
Duty cycle= (Ton/Ton +Toff) *100

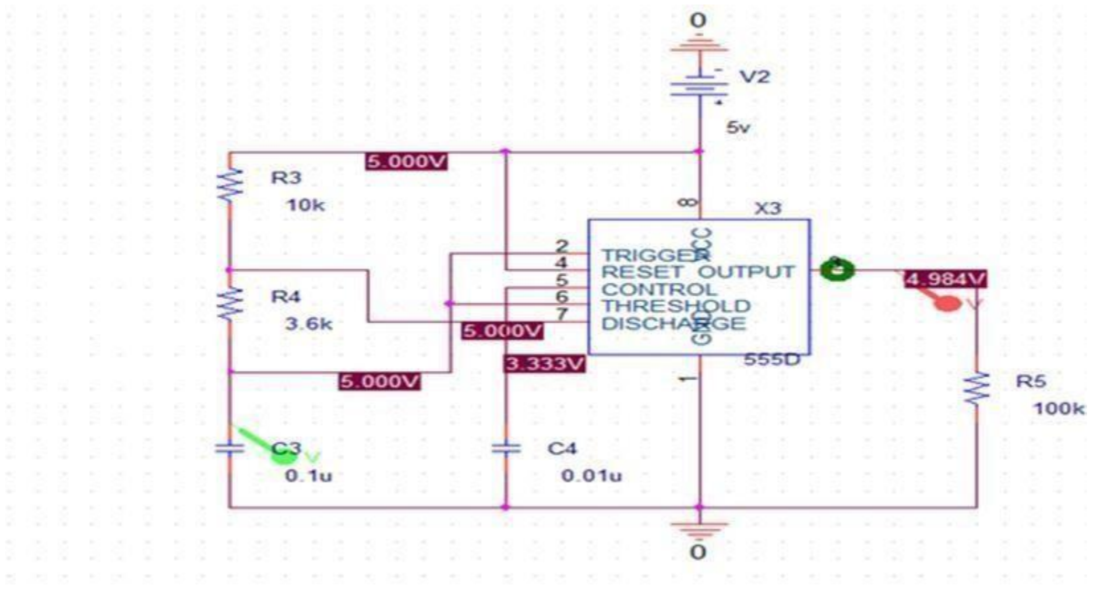
Duty cycle	**Duty cycle**	**Ton**	**Toff**
Theoretical	**50%**	**0.5ms**	**0.5ms**
	75%	**0.75ms**	**0.25ms**
	25%	**0.25ms**	**0.75ms**
Practical	**50%**		
	75%		
	25%		

SIMULATION using VHDL:

Simulation:

Circuit diagram: Astable multivibrator for duty cycle >50%

Figure 3: astablemultivibrator

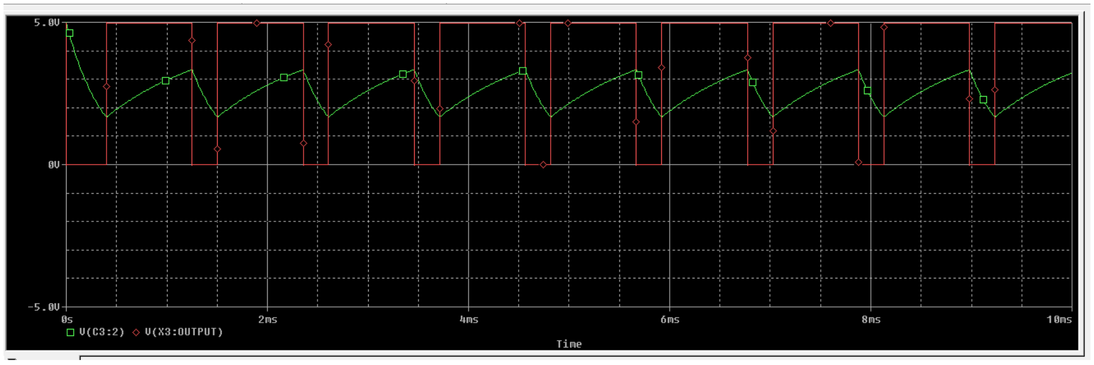


Output waveform:

Type of analysis: TIME DOMAIN (TRANSIENT)

Run to time: 10m

Step size: 0.01m



Experiment: 2

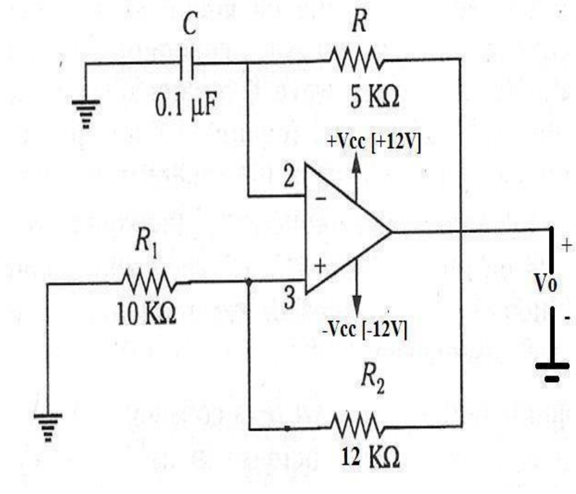
OP-AMP AS A RELAXATION OSCILLATOR

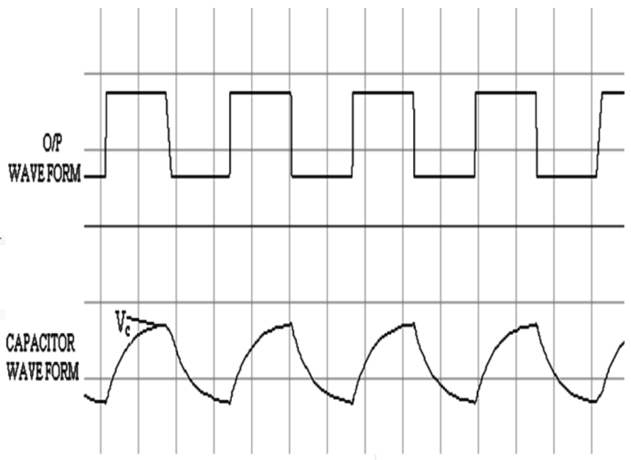
Using ua 741 Op-amp, design a 1 kHz Relaxation Oscillator with 50% duty cycle and simulate the same.

- A **relaxation oscillator**is an oscillator ckt that produces a non-sinusoidal output whose time period dependent on the charging time of a capacitor connected as a part of the oscillatorckt.
- Op-amp is very well suited for constructing of relaxation oscillatorckts that produce a rectangularo/p.

Components required: Op-amp µA-741, resistor 10KΩ, 12KΩ, 5KΩ, Capacitors 0.1µF (104), CRO probe and connecting wires.

Circuit Diagram and Waveforms:





DESIGN :

Design for a frequency of 1KHz (implies $T = \frac{1}{f} = \frac{1}{10^3} = 10^{-3} = 1ms$)

The period of the output rectangular wave is given as $T = 2RC\ln\left(\frac{1+\beta}{1-\beta}\right)$ -------(1)

Where, $\beta = \frac{R_1}{R_1 + R_2}$ is the feedback fraction

Let if R_2=1.16 R_1, then T = 2RC ----------(2)

Use R_2=1.16 R_1, for equation (2) to be applied.

Let R_1 = 10kΩ, then R_2 = 11.6kΩ

Choose next a value of C and then calculate value of R from equation (2).

Let C=0.1µF (i.e., 10^{-7}), then $R = \frac{T}{2C} = \frac{10^{-3}}{2\times10^{-7}} = 5K\Omega$

Working: -

- Let Us assume that the o/p is initially in +ve saturation. As a result voltage at the non-inverting i/p of op- amp is +Vsat*[R1/R1+R2].This forces the o/p to stay in +ve saturation and capacitor C is initially in fully dischargestate.
- Capacitor C starts charging towards +Vsat throughR.
- The moment the capacitor voltage exceeds the voltage appearing at the non-inverting i/p of op-amp ,the output switches to –Vsat, the voltage appearing at the non-inverting i/p of op-amp also changes to - Vsat*[R1/R1+R2]
- The capacitor starts discharging,it begins to discharge towards–Vsat.
- As soon as it becomes more negative than the negative threshold appearing at the non-inverting i/p of op-amp. The output switches back to +Vsat.
- The cycle repeats thereafter. The output is rectangularwave.
- Theexpressionfortimeperiodofoutputwaveformcanbederivedfromexponentialchargingand discharging process and is given by :

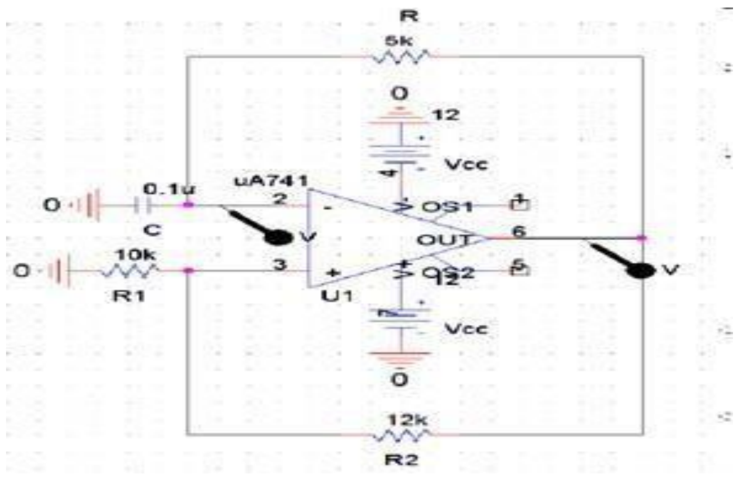
$$T = 2RC \ln\left(\frac{1+\beta}{1-\beta}\right)$$

PROCEDURE :

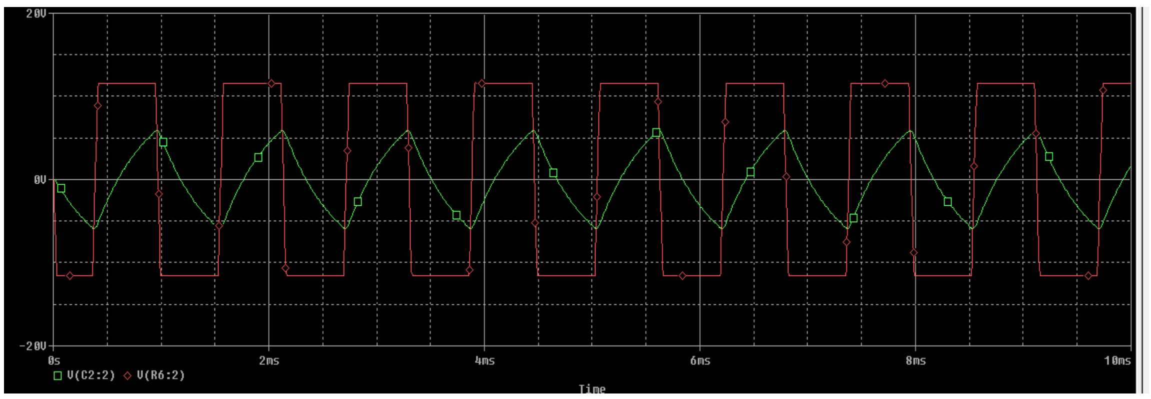
1. Before making the connections check all thecomponents.
2. Make the connections as shown in figure and switch on the powersupply.
3. Observe the voltage waveform across the capacitor onCRO.
4. Also observe the output waveform on CRO. Measure its amplitude andfrequency.

Result: Verified the O/P Waveform and calculated the practical oscillatorfrequency.
Period of the outputwaveform=____________ms, Frequencyf=___Hz

Design and implement a rectangular waveform generator (Op-Amp relaxation oscillator) using simulation package and demonstrate the changes in frequency when all resistor values are doubled.



TYPE OF ANALYSIS : TIME DOMAIN
RUN TO TIME : 20ms



Meghana Sambare R

Experiment: 3

Window Comparator for Any Given UTP And LTP

Experiment No.3: Using ua 741 Op-amp, design window comparator for any given UTP and LTP. And simulate the same.

Description:

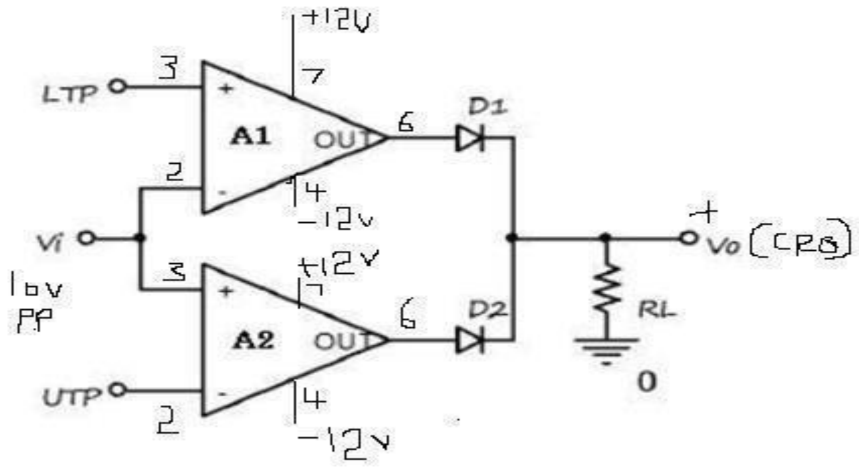
A **Window Comparator** is basically the inverting and the non-inverting comparators combined into a single comparator stage. The window comparator detects input voltage levels that are within a specific band or window of voltages, instead of indicating whether a voltage is greater or less than some preset or fixed voltage referencepoint.

This time, instead of having just one reference voltage value, a window comparator will have two reference voltages implemented by a pair of voltage comparators. One which triggers an op-amp comparator on detection of some upper voltage threshold, $V_{REF(UPPER)}$ and one which triggers an op-amp comparator on detection of a lower voltage threshold level, $V_{REF(LOWER)}$. Then the voltage levels between these two upper and lower reference voltages is called the "window", hence its name.

Components Required:

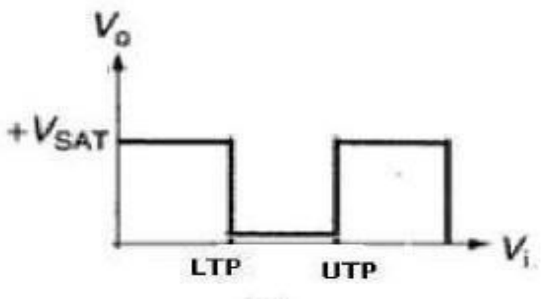
Two Op amp IC µA 741, Two diode 1N4007, Resistor of 1KΩ, DC regulated power Supply, trainer kit (+12v & -12v is given to Op amp from this), Signal generator, CRO.

Circuit:



Circuit Diagram for Window comparator

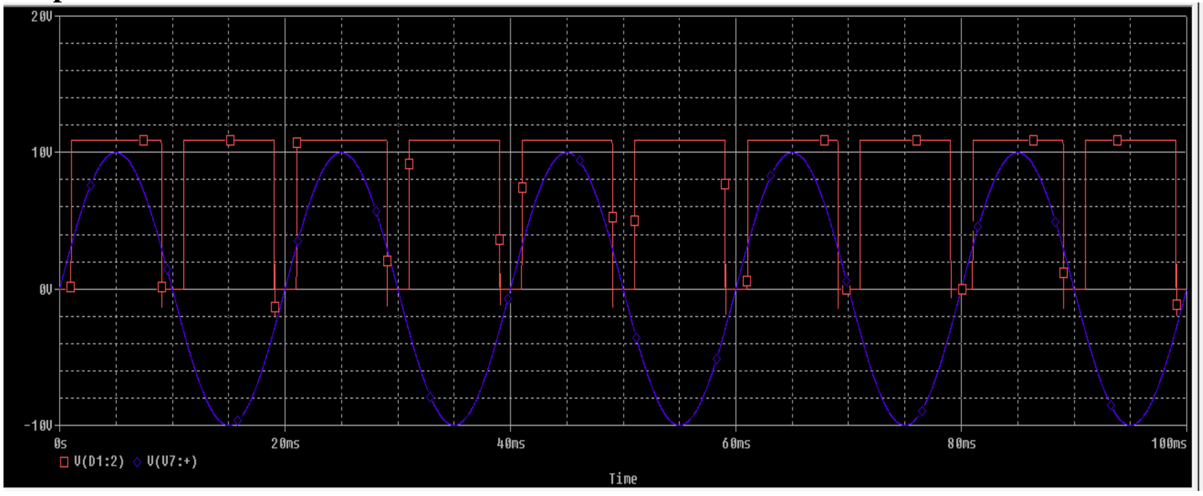
Output waveform



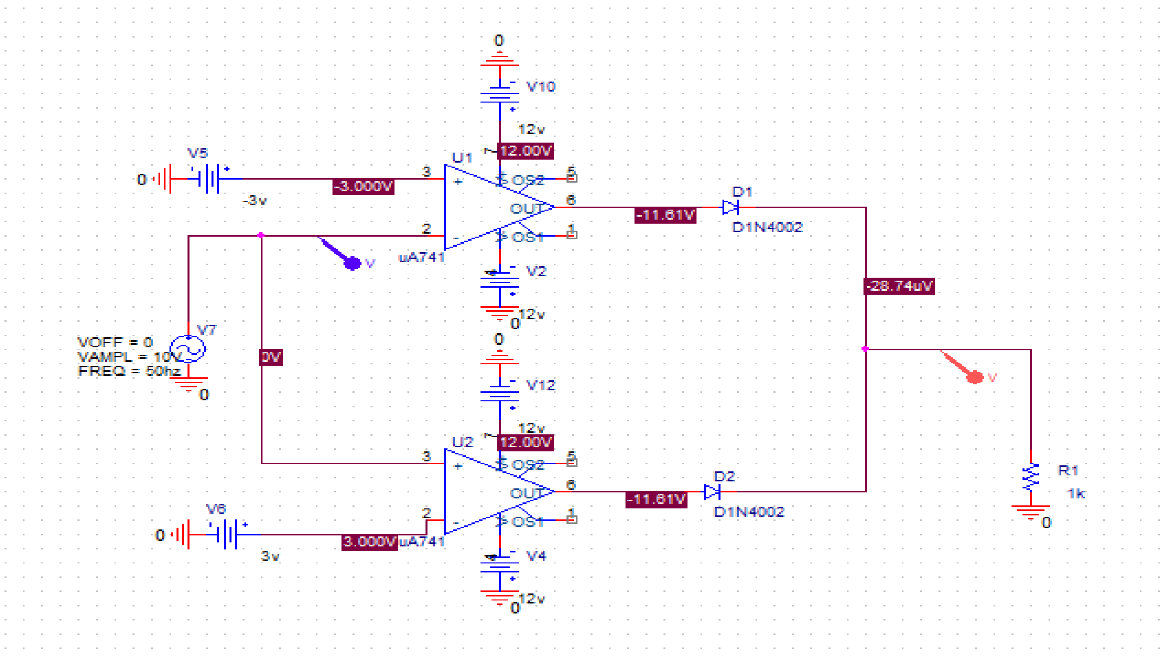
Simulation:
Components to be placed in the schematic:
Two Op amp IC µA 741, Two diode 1N4007, Resistor of 1KΩ, DC regulated power Supply, trainer kit (+12v & -12v is given to Op amp from this).

UTP =3V, LTP = -3V

Output waveform:



Type of analysis: TIME DOMAIN (TRANSIENT)
Run to time: 100m
step size: 0.01m



Module -2 **Karnaugh map**

Minimum Forms of Switching Functions:

Find a minimum sum of product expression for

$$F(a,b,c) = \Sigma m(0,1,2,5,6,7)$$

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$

a'b'(c'+c) + bc'(a'+a) + ac(b'+b)

we know that (a'+a)=1

F = a'b' + bc' + ac

OR

$$F = a'b'c' + a'b'c + a'bc' + ab'c + abc' + abc$$

= a'c'(b'+b) + b'c (a'+a) **ab(c'+c)**

we know that (a'+a)=1

F = a'c' + b'c + ab

Find minimum product of sum expression for the below

$$(A + B' + C + D')(A + B' + C' + D')(A + B' + C' + D)(A' + B' + C' + D)(A + B + C' + D)(A' + B + C' + D)$$

(A + B'+D')(C+C') **(B'+C'+D)(A+A')** **(B+C'+D)(A+A')**

We konw that (a+a')=1

$$\mathbf{F} = (A + B' + D')\ (B' + C' + D)\ (B + C' + D)$$

$$\mathbf{F} = (A + B' + D')\ (C' + D)(B'+B)$$

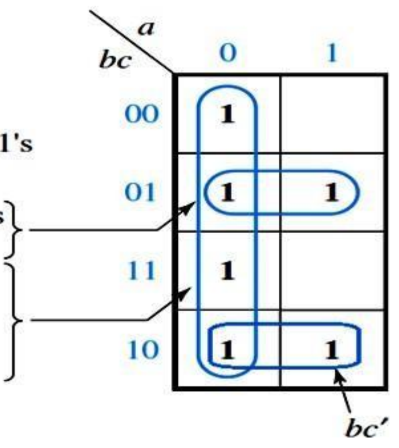
$$\mathbf{F} = (A + B' + D')(C' + D)$$

NOTE: If a function is given in algebraic form, each product term can be plotted directly as a group of 1‟s on the map for example.

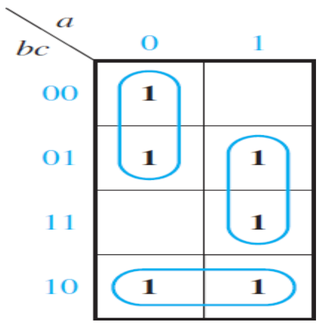
$$f(a,b,c) = bc' + b'c + a'$$

we would plot the map as follows:

1. The term bc' is 1 when $bc = 10$, so we place 1's in both squares of the $bc = 10$ row of the map.
2. The term $b'c$ is 1 when $bc = 01$, so we place 1's in both squares of the $bc = 01$ row of the map.
3. The term a' is 1 when $a = 0$, so we place 1's in all the squares of the $a = 0$ column of the map.



Problem- F (A,B,C) = $\bar{A}\,\bar{B} + B\,\bar{C} + A\,C$



TRUTH TABLE TO KARNAUGH MAP (K-MAP)

- K-MAP is the graphical representation of truth table, used to minimize the boolean expression.
- Using K-MAP expression with 2 – 4 variables are easily minimized**.**

HOW TO CONVERT TRUTH TABLE TO K-MAP

Two-Variables Maps

Consider the truth table given below

A	B	Y	Position in Decimal
0	0	0	0
0	1	0	1
1	0	**1** ➡ AB	2
1	1	**1** ➡ AB	3

Solution:

Constructing a Karnaugh Map

i) Note the variables and complements (A,B, $\bar{A}$ $\bar{B}$)

ii) The vertical column has $\bar{A}$ followed by A and horizontal row has $\bar{B}$ followed by B.

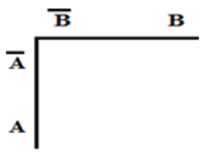


Fig - a

iii) In terms of decimal equivalence each position of Karnaugh map can be drawn in fig (b).

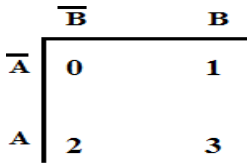


Fig - b

iv) The first output 1 appears for input A=1 and B=0. The fundamental product for this input condition

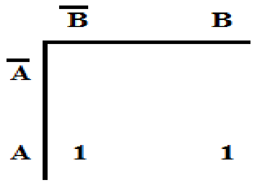


Fig - c

is $A\bar{B}$ .we can also observe for input A=1 and B=1, the output is 1. The fundamental product for this input condition is AB. Enter this fundamental product on the Karnaugh map as shown in fig (c).

v) The final step enter "0‟s in the remaining spaces of Karnaugh map as shown in fig (d).

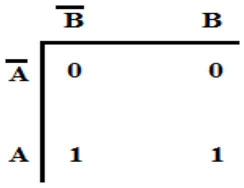


Fig - d

vi) The above table can be written using minterms as Y=∑m (2, 3).

Three variables maps

Draw the K- map for logic equation Y=F(A,B,C)=∑m(2,6,7)

Solution:-

A	**B**	**C**	**Y**
0	0	0	0
0	0	1	0
0	1	0	**1**
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	**1**
1	1	1	**1**

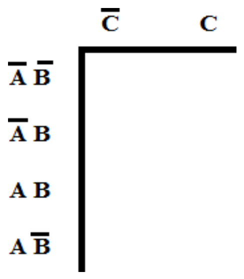


Fig: (a)

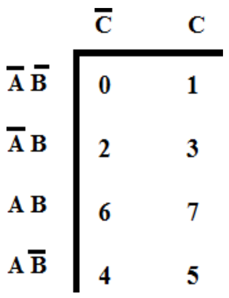


Fig: (b)

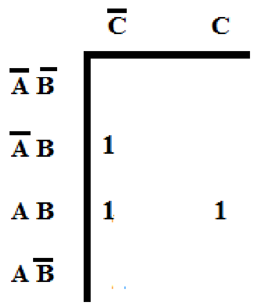


Fig: (c)

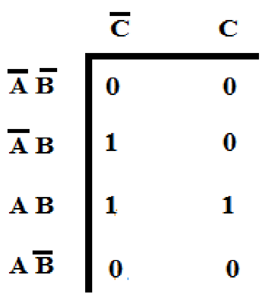


Fig: (d)

- ✓ First draw the blank map (fig.a). The vertical columns are labeled $\overline{A}\overline{B}$, $\overline{A}B$, $A\overline{B}$, AB . With this order only one variable changes from complemented to un-complemented form (or vice versa) as move downwards.
- ✓ In terms of decimal equivalence of each position the Karnaugh map is shown in fig (b).
- ✓ Now in the Table, output 1 appears for ABC inputs 010,110 and 111. The fundamental products for these input conditions are $\overline{A}B\overline{C}$,$AB\overline{C}$,$ABC$. Enter 1s for these products on the Karnaugh map as shown infig (c).
- ✓ The final step is to enter 0s in the remaining spaces as shown in fig (d).

Four-Variable Maps

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

- ✓ Draw a blank map (Fig.a). The vertical row is $\overline{A}\overline{B}$, $\overline{A}B$, AB, $A\overline{B}$horizontal row is $\overline{C}\overline{D}$,$\overline{C}D$, CD, $C\overline{D}$.
- ✓ In terms of decimal equivalence of each position the Karnaugh map is shown in fig.b.
- ✓ From the table, we can observe output 1s appearing for ABCD inputs of 0001, 0110, 0111 and 1110. The fundamental products for these input conditions are $\overline{A}\overline{B}\overline{C}D$, $\overline{A}BC\overline{D}$, $\overline{A}BCD$, $ABC\overline{D}$.
- ✓ Fig.c shows entering 1‟s on the Karnaugh map.
- ✓ The final step is filling 0s in the map as shown in fig.C.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$				
$\bar{A}B$				
AB				
$A\bar{B}$				

Fig (a)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

Fig (b)

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0	0
$\bar{A}B$	0	0	1	1
AB	0	0	0	1
$A\bar{B}$	0	0	0	0

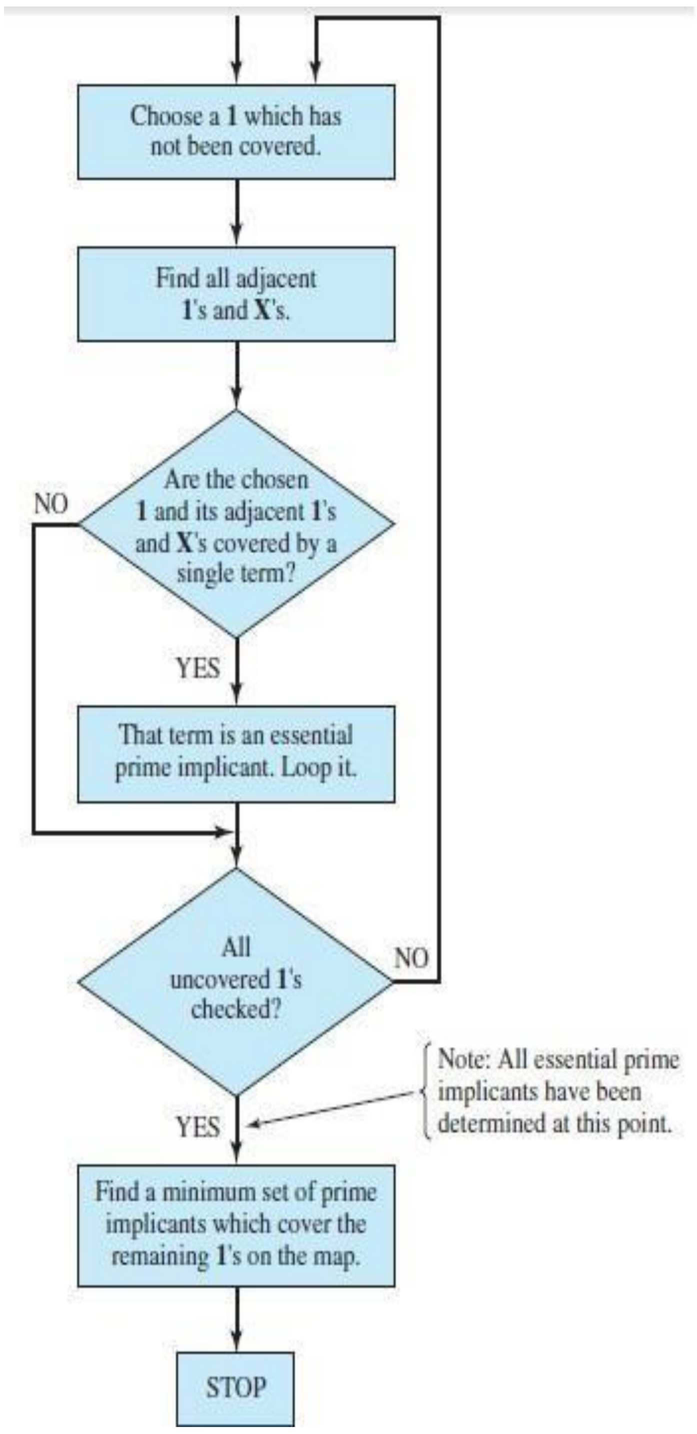
Fig (c)

DETERMINATION OF MINIMUM EXPRESSIONS USING ESSENTIAL PRIME IMPLICANTS:

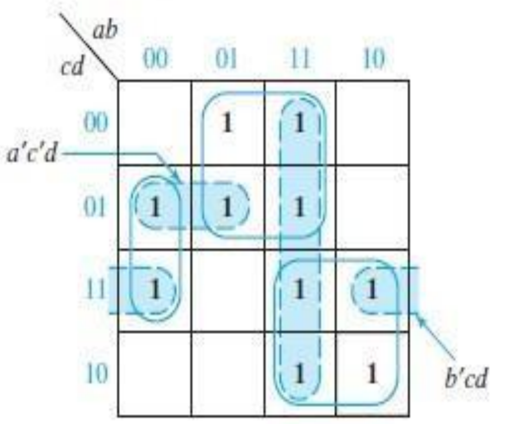
Any single 1 or any group of 1's which can be combined together on a map of the function F represents a product term which is called an implicant of F. Several implicants of F MAY BE POSSIBLE. A product term implicant is called a *prime implicant* if it cannot be combined with another term to eliminate a variable.

The following Figure shows the flowchart for determining a Minimum Sum of Products using a KarnaughMap with an Example.

1. Choose a minterm (a 1) which has not yet been covered.
2. Find all 1's and X's adjacent to that minterm (Check the n adjacent squares on an n-variable map.
3. If a single term covers the minterm and all of the adjacent 1's and X's, then that term is an essential prime implicant, so select that term. (Note that don't-care terms are treated like 1's in steps 2 and 3 but not in step 1.)
4. Repeat steps 1, 2, and 3 until all essential prime implicants have been chosen.
5. Find a minimum set of prime implicants which cover the remaining 1's on the map. (If there is more than one such set, choose a set with a minimum number of literals.)


Choose a 1 which has not been covered.
Find all adjacent 1's and X's.
Are the chosen 1 and its adjacent 1's and X's covered by a single term?
NO
YES
That term is an essential prime implicant. Loop it.
All uncovered 1's checked?
NO
YES
Note: All essential prime implicants have been determined at this point.
Find a minimum set of prime implicants which cover the remaining 1's on the map.
STOP

Minimum solution: $F = a'b'd + bc' + ac$
All prime implicants: $a'b'd$, bc', ac, $a'c'd$, ab, $b'cd$



DETERMINATION OF PRIME IMPLICANTS:

- ❖ In order to apply the Quine-McCluskey method to determine a minimum sum-of-productsexpression for a function, the function must be given as a sum of minterms.
- ❖ In the first part of the Quine-McCluskey method, all of the prime implicants of a function are systematically formed by combining minterms.

Two minterms will combine if they differ in one variable

show both the binary notation and its algebraic equivalent.

$$AB'CD' + AB'CD = AB'C$$

$$\underbrace{1\ 0\ 1}_{X}\ \underbrace{0}_{Y} + \underbrace{1\ 0\ 1}_{X}\ \underbrace{1}_{Y'} = \underbrace{1\ 0\ 1}_{X}\ - \quad \text{(the dash indicates a missing variable)}$$

$$A'BC'D + A'BCD' \quad \text{(will not combine)}$$

$$0\ 1\ 0\ 1 + 0\ 1\ 1\ 0 \quad \text{(will not combine)}$$

In order to find all of the prime implicants, all possible pairs of minterms should be compared and combined whenever possible. To reduce the required number of comparisons, the binary minterms are sorted into groups according to the number of 1’s in each term.

Now, function; *f(a, b, c, d) = ∑m (0, 1, 2, 5, 6, 7, 8, 9, 10, 14)* can be represented by following list of minterms:

group 0	0	0000
group 1	1	0001
	2	0010
	8	1000
group 2	5	0101
	6	0110
	9	1001
	10	1010
group 3	7	0111
	14	1110

- ❖ In this list, the term in group 0 has zero 1’s, the terms in group 1 have one 1, those in group 2 have two 1’s, and thosein group 3 have three 1’s.
- ❖ Two terms can be combined if they differ in exactly one variable. Only terms in adjacent groups must be compared.
- ❖ First, we will compare the term in group 0 with all of the terms in group 1.Terms 0000 and 0001 can be combined toeliminate the fourth variable, which yields 000– ($a'b'c'$).
- ❖ Similarly, 0 and 2 combine to form 00–0 ($a'b'd'$), and 0 and 8 combine to form –000 ($b'c'd'$). The resulting terms are listed in Column II of the following Table.
- ❖ Whenever two terms combine, the corresponding decimal numbers differ by a power of 2 (1, 2, 4,8, etc.).
- ❖ Since the comparison of group 0 with groups 2 and 3 is unnecessary, we proceed to compare termsin groups 1 and 2. Comparing term 1 with all terms in group 2, we find that it combines with 5 and9 but not with 6 or 10. Similarly, term 2 combines only with 6 and 10, and term 8 onlywith 9 and 10. The resulting terms are listed in Column 2.

- ❖ Each time a term is combined with another term, it is checked off. Also note that, a term may be used more than once. Even though two terms have already been combined with other terms, they still must be compared and combined if possible.
- ❖ At this stage, we may generate redundant terms, but these redundant terms will be eliminated later.

We finish with Column 1 by comparing terms in groups 2 and 3. New terms are formed by combining terms 5 and 7, 6 and 7, 6 and 14, and 10 and 14.

a	b	c	d	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Column 1 abcd	Column 2 abcd	Column 3 abcd

1. Note that the terms in Column 2 have been divided into groups. In order to combine two terms, theterms must have the same variables, and the terms must differ in exactly one of these variables. Thus, it is necessary only to compare terms which have dashes (missing variables) in correspondingplaces and which differ by exactly one in the number of 1's.
2. Terms in the first group in Column 2 need only be compared with terms in the second group whichhave dashes in the same places. Term 000– (0, 1) combines only with term 100– (8, 9) to yield – 00– ($b'c'$).

3. The resulting term is listed in Column 3 along with the designation 0, 1, 8, 9 to indicate that it wasformed by combining minterms 0, 1, 8, and 9.

4. Term (0, 2) combines only with (8, 10), and term (0, 8) combines with both (1, 9) and (2, 10).
5. Again, the terms which have been combined are checked off. Comparing terms from the second and third groups in Column 2, we find that (2,6) combines with (10, 14), and (2, 10) combines with(6,14).
6. Note that there are three pairs of duplicate terms in Column 3. These duplicate terms were formedin each case by combining the same set of four minterms in a different order.
7. After deleting the duplicate terms, we compare terms from the two groups in Column 3. Because no further combination is possible, the process terminates.

8. In general, we would keep comparing terms and forming new groups of terms and new columns until no more terms could be combined. The terms which have not been checked off because they cannot be combined with other terms are called *prime implicants*. Because every minterm has been included in at least one of the prime implicants, the function is equal to the sum of its prime implicants. In this example we have;

$$f = \underset{(1,5)}{a'c'd} + \underset{(5,7)}{a'bd} + \underset{(6,7)}{a'bc} + \underset{(0,1,8,9)}{b'c'} + \underset{(0,2,8,10)}{b'd'} + \underset{(2,6,10,14)}{cd'}$$

Definition:

Given a function F of n variables, a product term P is an ***implicant*** of F iff for every combinationof values of the n variables for which P = 1, F is also equal to 1.

A ***prime implicant*** of a function F is a product term implicant which is no longer an implicant ifany literal is deleted from it.

Consider an Example:

$$F(a, b, c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac$$

In the above function, the implicant $a'b'c'$ is not a prime implicant because a can be eliminated, andthe resulting term $b'c'$ is still an implicant of F. The implicants $b'\ c'$ and are prime implicants because if we delete a literal from either term, the term will no longer be an implicant of F. The Quine-McCluskey method, as previously illustrated, finds all of the product term implicants of a function. The implicants which are nonprime are checked off in the process of combining terms, so that theremaining terms are prime implicants. Any nonprime term in a sum-of-products expression can thus be replaced with a prime implicant, which reduces the number of literals and simplifies the expression.

THE PRIME IMPLICANT CHART:

The second part of the Quine-McCluskey method employs a prime implicant chart to select a minimum set of prime implicants. The minterms of the function are listed across the top of the chart, and the prime implicants are listed down the side. A prime implicant is equal to a sum of minterms, and the prime implicant is said to cover these minterms. If a prime implicant covers a given minterm, an ***X*** is placed at the intersection of the corresponding row and column. The following Table shows the prime implicant. All of the prime implicants (terms which have notbeen checked off in the above Table) are listed on the left.

-	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9) $(b' c')$										
(0, 2, 8, 10) $(b'd')$										
(2, 6, 10, 14) (cd')										
(1, 5) $(a'c'd)$										
(5, 7) $(a' bd)$										
(6, 7) $(a'bc)$										

1. In the first row, X's are placed in columns 0, 1, 8, and 9, because prime implicant $b'c'$ was formedfrom the sum of minterms 0, 1, 8, and 9. Similarly, the all other X's are placed.
2. If a minterm is covered by only one prime implicant, then that prime implicant is called an *essentialprime implicant* and must be included in the minimum sum of products. Essential prime implicantsare easy to find using the prime implicant chart. If a given column contains only one$X$, then the corresponding row is an essential prime implicant. In the above Table, columns 9 and 14 each contain one X, so prime implicants $b'c'$ and cd' are essential.
3. Each time a prime implicant is selected for inclusion in the minimum sum, the corresponding rowshould be crossed out. After doing this, the columns which correspond to all minterms covered by that prime implicant should also be crossed out.
4. A minimum set of prime implicants must now be chosen to cover the remaining columns. In this example, the resulting minimum sum of products is –

$$f = b'c' + cd' + a'bd$$

Example: *Solve using QM method: F= ∑m (0, 1, 2, 5, 6, 7).Solution:*

a	b	c	F
1	0	0	
1	0	1	
1	1	0	
1	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

<u>Column 1</u> abc	<u>Column 2</u> abc

The following Table shows the resulting prime implicants chart:

-	0	1	2	5	6	7

Therefore, ***F =***

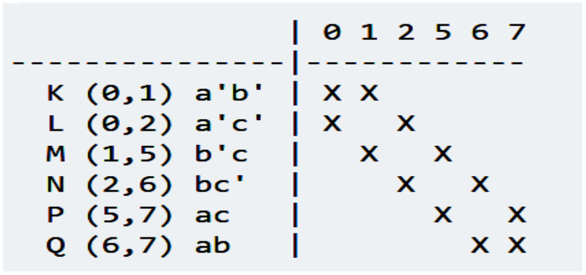
<u>NOTE:</u> A prime implicant chart which has two or more *X*'s in every column is called a *cyclic primeimplicant chart.*

Petrick Method:

- Petrick‟s method is a technique for determining all minimum sum-of-products solutions from a prime implicant chart.
- Petrick's method is very tedious for large charts, but it is easy to implement on a computer.

$$F = \Sigma\, m(0, 1, 2, 5, 6, 7)$$

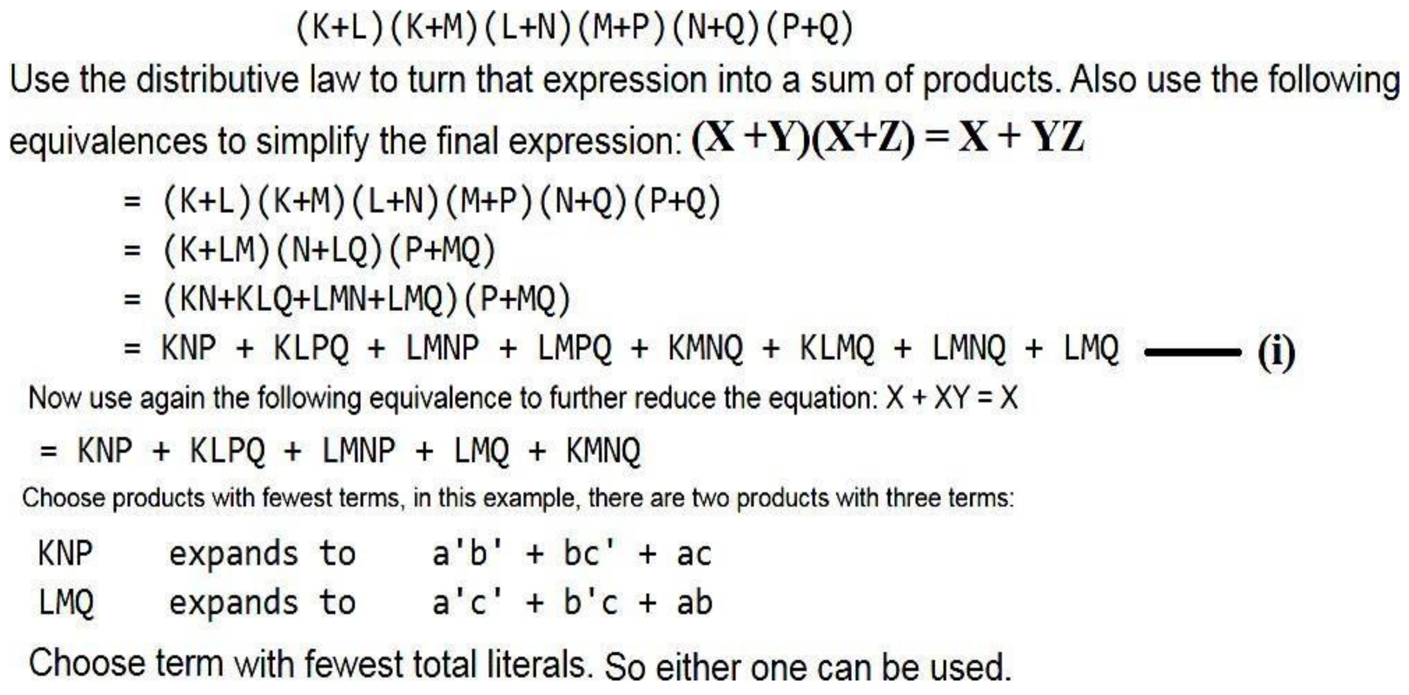
0	000	0	000 ✓	0, 1	00–
1	001	1	001 ✓	0, 2	0–0
2	010	2	010 ✓	1, 5	–01
5	101	5	101 ✓	2, 6	–10
6	110	6	110 ✓	5, 7	1–1
7	111	7	111 ✓	6, 7	11–



	0	1	2	5	6	7
K (0,1) a'b'	X	X				
L (0,2) a'c'	X		X			
M (1,5) b'c		X		X		
N (2,6) bc'			X		X	
P (5,7) ac				X		X
Q (6,7) ab					X	X

- We will label the rows of the table *K*, *L*, *M*, etc. We will form a logic function Y, which is true when all of the minterms in the chart have been covered.
- We must choose row *K* or *L* in order to cover minterm 0. Therefore, the expression (*P*1 + *P*2).
- In order to cover minterm 1, we must choose row *K* or *M*; therefore, (*P*1+*P*3).
- In order to cover minterm 2, (*L*+*N*).
- Similarly, in order to cover minterms 5, 6, and 7, the expressions (*M*+*P*), (*N*+*Q*) and (*P*+*Q*).

Note: in equation (i) we have
LMPQ + KLMQ + LMNQ + LMQ



(K+L)(K+M)(L+N)(M+P)(N+Q)(P+Q)

Use the distributive law to turn that expression into a sum of products. Also use the following equivalences to simplify the final expression: **(X + Y)(X+Z) = X + YZ**

= (K+L)(K+M)(L+N)(M+P)(N+Q)(P+Q)
= (K+LM)(N+LQ)(P+MQ)
= (KN+KLQ+LMN+LMQ)(P+MQ)
= KNP + KLPQ + LMNP + LMPQ + KMNQ + KLMQ + LMNQ + LMQ —— **(i)**

Now use again the following equivalence to further reduce the equation: X + XY = X

= KNP + KLPQ + LMNP + LMQ + KMNQ

Choose products with fewest terms, in this example, there are two products with three terms:

KNP	expands to	a'b' + bc' + ac
LMQ	expands to	a'c' + b'c + ab

Choose term with fewest total literals. So either one can be used.

WE KNOW THAT X + XY = X(1+Y) =X ......BCOZ [Y+1=1]
LMPQ + KLMQ + LMQ(1+N)
LMPQ + KLMQ+LMQ = LMPQ + LMQ (1+K) = LMPQ +
LMQLMQ(1+P) = LMQ

So that in equation LMPQ + KLMQ + LMNQ + LMQ these terms can be written as LMQ

Note: instead of label K,L,M.......... we can choose P1,P2,P3 and so on

Simplification of Incompletely Specified Functions:

Prob-3]Simplify F(A,B,C,D)=∑m(1,3,6,7,8,9,10,12,14,15) + dc (11,13)using QM method

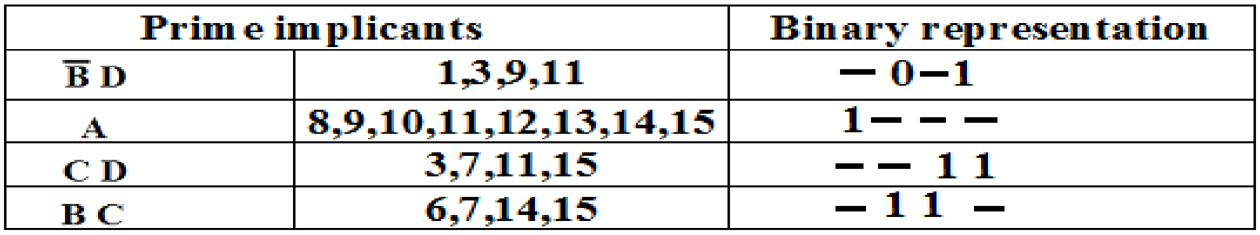
Stage - 1	
Minterm	**Binary representation**
1	**0001**
3	**0011**
6	**0110**
7	**0111**
8	**1000**
9	**1001**
10	**1010**
12	**1100**
14	**1110**
15	**1111**
11	**1011**
13	**1101**

Stage - 2	
Minterm	**Binary representation**
1	**0001** √
8	**1000** √
3	**0011** √
6	**0110** √
9	**1001** √
10	**1010** √
12	**1100** √
7	**0111** √
11	**1011** √
13	**1101** √
14	**1110** √
15	**1111** √

Stage 3	
Minterm	**Binary representation**
1,3	00 -1 √
1,9	-0 0 1 √
8,9	100- √
8,10	10-0 √
8,12	1-00 √
3,7	0-11 √
3,11	-011 √
6,7	011- √
6,14	-110 √
9,11	10-1 √
9,13	1-01 √
10,11	101- √
10,14	1-10 √
12,13	110- √
12,14	11-0 √
7,15	-111 √
11,15	1-11 √
13,15	11-1 √
14,15	111- √

Minterms	**Binary representation**
1,3,9,11	-0-1
8,9,10,11	10- - √
8,9,12,13	1-0- √
8,9,12,14	1--0 √
3,7,11,15	- - 11
6,7,14,15	-11-
9,11,13,15	1- - 1 √
10,11,14,15	1- 1- √
12,13,14,15	1 1 - - √

Minterms	**Binary representation**
8,9,10,11,12,13,14,15	1 - - -



Prime implicants		**Binary representation**
$\bar{B}D$	1,3,9,11	−0−1
A	8,9,10,11,12,13,14,15	1− − −
C D	3,7,11,15	− − 1 1
B C	6,7,14,15	− 1 1 −

Prime implicants	m1	m3	m6	m7	m8	m9	m10	d11	m12	d13	m14	m15
1,3,9,11	ʘ	▪				▪		▪				
8,9,10,11,12,13,14,15					ʘ	▪	▪	▪	▪	▪	▪	▪
3,7,11,15		▪		▪				▪				▪
6,7,14,15			ʘ	▪							▪	▪

$$Y = \overline{B}D + A + BC$$

Note: In the determination of prime implicants consider all don‟t care‟s as minterms. For finding essential prime implicants don‟t care‟s are not necessary to cover.

Disadvantages:-

- This method is more tedious and people don‟t prefer it for simplification problem with smaller number of variables.

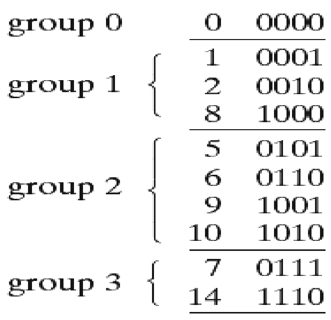
Advantage:-

- For simplification problems with large number of variables Quine-McClusky method can offer solution and Karnaugh map does not.

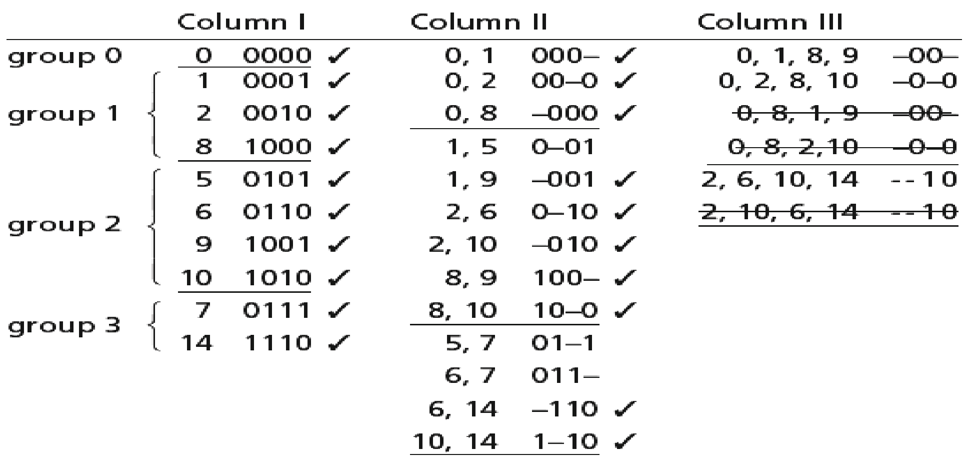
$$f(a, b, c, d) = \Sigma\, m(0, 1, 2, 5, 6, 7, 8, 9, 10, 14)$$

Problem-1

Solution:



group 0	0	0000
group 1	1	0001
	2	0010
	8	1000
group 2	5	0101
	6	0110
	9	1001
	10	1010
group 3	7	0111
	14	1110



	Column I		Column II		Column III	
group 0	0	0000 ✓	0, 1	000– ✓	0, 1, 8, 9	–00–
group 1	1	0001 ✓	0, 2	00–0 ✓	0, 2, 8, 10	–0–0
	2	0010 ✓	0, 8	–000 ✓	~~0, 8, 1, 9~~	~~–00–~~
	8	1000 ✓	1, 5	0–01	~~0, 8, 2,10~~	~~–0–0~~
group 2	5	0101 ✓	1, 9	–001 ✓	2, 6, 10, 14	- - 10
	6	0110 ✓	2, 6	0–10 ✓	~~2, 10, 6, 14~~	~~- - 10~~
	9	1001 ✓	2, 10	–010 ✓		
	10	1010 ✓	8, 9	100– ✓		
group 3	7	0111 ✓	8, 10	10–0 ✓		
	14	1110 ✓	5, 7	01–1		
			6, 7	011–		
			6, 14	–110 ✓		
			10, 14	1–10 ✓		

		0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	$b'c'$	×	×					×	⊗		
(0, 2, 8, 10)	$b'd'$	×		×				×		×	
(2, 6, 10, 14)	cd'			×		×				×	⊗
(1, 5)	$a'c'd$		×		×						
(5, 7)	$a'bd$				⊗		×				
(6, 7)	$a'bc$					×	×				

$$f = b'c' + cd' + a'bd$$

Problem-2

$$F(A, B, C, D) = \Sigma\, m(2, 3, 7, 9, 11, 13) + \Sigma\, d(1, 10, 15)$$

(the terms following d are don't-care terms)

The don't-care terms are treated like required minterms when finding the prime implicants:

1	0001	✓
2	0010	✓
3	0011	✓
9	1001	✓
10	1010	✓
7	0111	✓
11	1011	✓
13	1101	✓
15	1111	✓

(1, 3)	00–1	✓
(1, 9)	–001	✓
(2, 3)	001–	✓
(2, 10)	–010	✓
(3, 7)	0–11	✓
(3, 11)	–011	✓
(9, 11)	10–1	✓
(9, 13)	1–01	✓
(10, 11)	101–	✓
(7, 15)	–111	✓
(11, 15)	1–11	✓
(13, 15)	11–1	✓

(1, 3, 9, 11)	–0–1
(2, 3, 10,11)	–01–
(3, 7, 11, 15)	- - 1 1
(9, 11, 13, 15)	1 - - 1

The don't-care columns are omitted when forming the prime implicant chart:

	2	3	7	9	11	13
(1, 3, 9, 11)		*		*	*	
*(2, 3, 10, 11)	⊗	*			*	
*(3, 7, 11, 15)		*	⊗		*	
*(9, 11, 13, 15)				*	*	⊗

$$F = B'C + CD + AD$$

Problem-3

Find a minimum sum-of-products expression for the following function:

$$f(A,B,C,D,E) = \Sigma\, m(0,2,3,5,7,9,11,13,14,16,18,24,26,28,30)$$

0	00000	✓
2	00010	✓
16	10000	✓
3	00011	✓
5	00101	✓
9	01001	✓
18	10010	✓
24	11000	✓
7	00111	✓
11	01011	✓
13	01101	✓
14	01110	✓
26	11010	✓
28	11100	✓
30	11110	✓

0, 2	000–0	✓
0, 16	–0000	✓
2, 3	0001–	
2, 18	–0010	✓
16, 18	100–0	✓
16, 24	1–000	✓
3, 7	00–11	
3, 11	0–011	
5, 7	001–1	
5, 13	0–101	
9, 11	010–1	
9, 13	01–01	
18, 26	1–010	✓
24, 26	110–0	✓
24, 28	11–00	✓
14, 30	–1110	
26, 30	11–10	✓
28, 30	111–0	✓

0, 2, 16, 18	–00–0
16, 18, 24, 26	1–0–0
24, 26, 28, 30	11--0

	0	2	3	5	7	9	11	13	14	16	18	24	26	28	30
(0, 2, 16, 18)	⊗	×								×	×				
(16, 18, 24, 26)										×	×	×	×		
(24, 26, 28, 30)												×	×	⊗	X
(2, 3)		×	×												
(3, 7)			×		×										
(3, 11)			×				⊗								
(5, 7)				×	⊗										
(5, 13)				×				×							
(9, 11)						×	×								
(9, 13)						⊗		×							
(14, 30)									⊗						×

$$f = B'C'E' + ABE' + A'C'DE + A'B'CE + A'BD'E + BCDE'$$

or

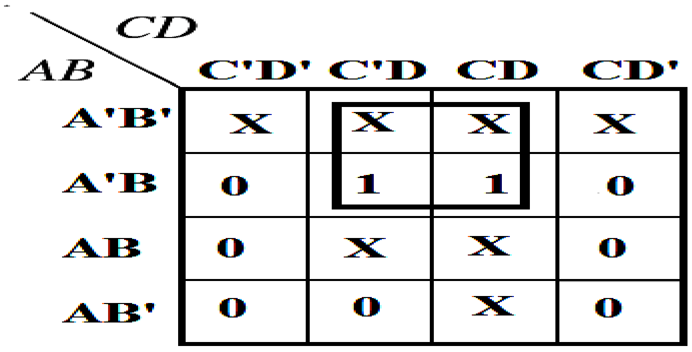
$$f = BCDE' + B'C'E' + ABE' + A'B'DE + A'CD'E + A'BC'E$$

Simplification Using Map entered Variables:

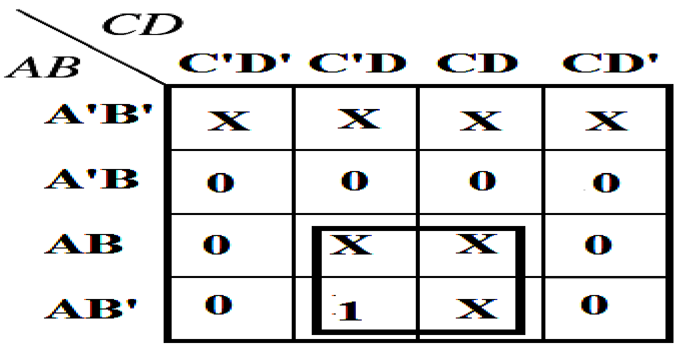
- By using map-entered variables, Karnaugh map techniques can be extended to simplify functions with more than five variables.
- sum-of-products expression for *F* of the form

$$F = MS0 + P1MS1 + P2MS2 + \cdots$$

Where,
*MS*0 is the minimum sum obtained by setting *P*1 = *P*2 =· · · =0.
*MS*1 is the minimum sum obtained by setting *P*1 =1, *P2* =0, and replacing all 1"s on the map with don"t-cares.



$E = 1,\ F = 0$
$MS_1 = A'D$
(c)



$E = 0,\ F = 1$
$MS_2 = AD$
(d)

S2 is the minimum sum obtained by setting P1 =0, P2 = 1 and replacing all 1‟s on the map with don‟t-cares.

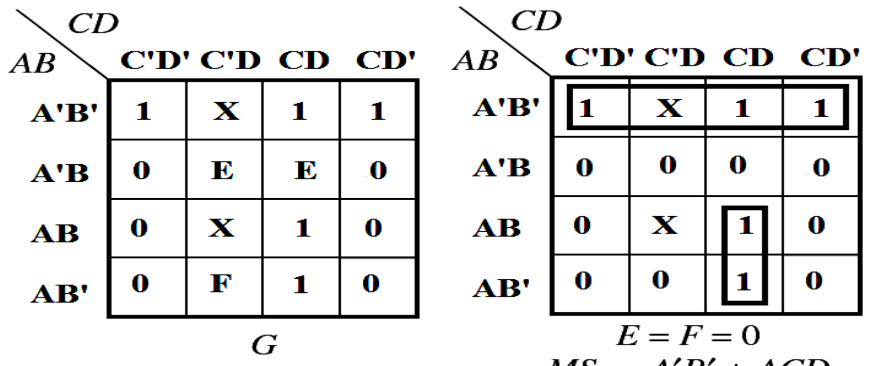
- ❖ Figure below shows a four-variable map with two additional variables entered in the squares in the map.

$$G = A'B' + ACD + EA'D + FAD$$

- ❖ The resulting expression is a minimum sum of products for *G*:

Problem-1: $G(A, B, C, D, E, F) = m_0 + m_2 + m_3 + Em_5 + Em_7 + Fm_9 + m_{11} + m_{15}$ (+ don't-care terms **1, 10, 13**)

Soln:



$E = F = 0$

$MS_0 = A'B' + ACD$

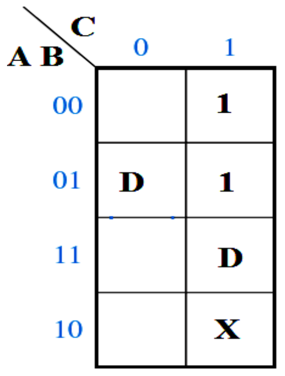
(a) **(b)**

- ❖ Now use a three-variable map to simplify the function

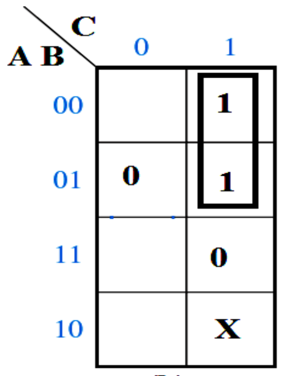
Problem-2: Use 3-variable k-map to simplify the below function:

$F(A, B, C, D) = A'B'C + A'BC + A'BC'D + ABCD + (AB'C)$

where the A B' C is a don't-care term.

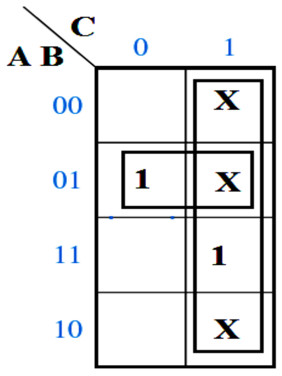


(a)



(b)

MSo = A' C when D =0

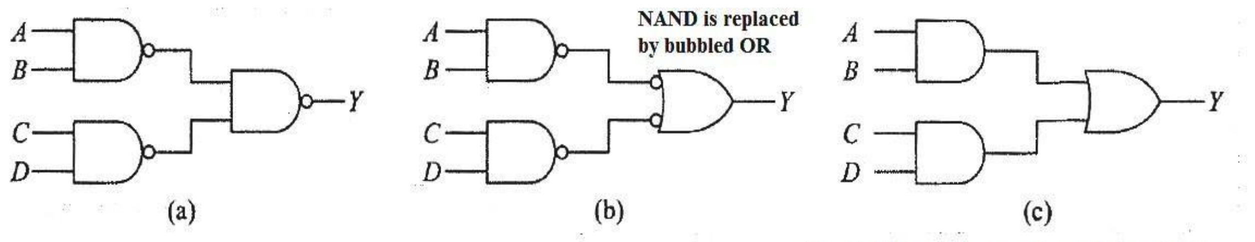


(c)

MS $_1$ = C+ A'B when D=1 & minterm as X

❖ The resulting expression is a minimum sum of products for *F*:

$$F = A'C + D(C + A'B) = A'C + CD + A'BD$$



Two bubbles on the same line called as double inversion. each double inversion cancell out fig (c)

NOTE:

Prove that NAND-NAND network is equal to AND-OR network

- ❖ **SOP expression can be represented by either using AND-OR or NAND-NAND network**
- ❖ **POS expression can be represented by either using OR-AND or NOR-NOR network**

Assignment questions:

1. Using K-map to simplify the Boolean expression and give the implementation of the same using NOR gates only. F(A,B,C,D) = ΠM (0,1,2,4,5,12,14) +dc(8,10)

   Soln:

$$(\overline{A} + \overline{D})(\overline{C} + \overline{D})(A + \overline{B} + \overline{C})$$

2. Simplify the following Boolean expression using Quine McClusky method
   F(A,B,C,D) = ∑m (1,2,8,9,10,12,13,14)
   Soln:

$$\overline{B}\,\overline{C}\,D + \overline{B}\,C\,\overline{D} + A\,\overline{C} + A\,\overline{D}$$

3. For the below expression find all the prime implicant using QM method
   F(A,B,C,D)= ∑m(1,5,7,9,11,12,14,15)

4. A digital system is to be designed in which the months of the year is given as input in four bit form. The month January is represented as „0000‟ and so on. The output of the system should be „1‟ corresponding to the input of the month containing 31 days or otherwise it is „0‟. Consider the excess numbers in the input beyond „1011‟ as don‟t care conditions. For this system of four variables (A,B,C,D), find the following:
   i) Boolean expression in Σm and πM form
   ii) Write the truth table
   iii) Using K-map, simplify the Boolean expression of canonical min term form
   iv) Implement the simplified equation using NAND-NAND gates.
   v) Simplify the same using QM method.

Soln:

$$Y = \overline{A}\,\overline{D} + A D + B C$$

5. A digital system is to be designed in which the months of the year is given as input in four bit form. The month January is represented as „0000" and so on. The output of the system should be „1" corresponding to the input of the month containing 31 days or otherwise it is „0". Consider the excess numbers in the input beyond „1011" as don"t care conditions. For this system of four variables (A,B,C,D), find the following:

i) Boolean expression in Σm and πM form
ii) Write the truth table
iii) Using K-map, simplify the Boolean expression of canonical min term form
iv) Implement the simplified equation using NAND-NAND gates.
v) Simplify the same using QM method.

Soln: $Y = \overline{A}\,\overline{D} + A D + B C$

6. Using K-map to simplify the Boolean expression and give the logic circuit.
F(A,B,C,D) = ∑m (0,3,4,5,6,7,11,15) +dc(2,8,9,10,12,13)
Soln: $Y = \overline{A} B + C D + \overline{B}\,\overline{D}$

7. Reduce the following function using QM method
F(A,B,C,D) = ∑m (1,4,,6,8,9,10,11,12,13) +dc(3,15)
Soln: $Y = A \overline{B} + A \overline{C} + \overline{A} B \overline{D} + \overline{B} D$

8. Using K-map to simplify the Boolean expression and give the logic circuit.
F(A,B,C,D) = ΠM (0,1,2,4,5,12,14) +dc(8,9,11,12,13,15)
Soln: $Y = C (B + D)$

9. Give the logic circuit simplified logic circuit and equation using QM method for the following Boolean expression F(A,B,C,D) = ∑m (0,3,5,6,7,11,14)
Soln: $Y = \overline{B} C D + \overline{A} B D + B C \overline{D} + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

10. Using K-Map technique simplify F(A,B,C,D)=Σm(0,2,3,5,6,7,8,9)+dc(10,11,12,13,14,15) draw the SOPcircuit.
Soln: $Y = A + B D + \overline{B}\,\overline{D} + \overline{A} C$

11. Using Quine McClusky method simplify f(A,B,C,D)=Σm(1,3,6,7,10,12,13,14,15)
Soln: $Y = B C + A B + \overline{A} C D + A C \overline{D}$

12. Find the minimum sum of product expression using QM method for the function

F(A,B,C,D)= ∑m(1,3,4,5,6,7,10,12,13)+ ∑dc(2,9,15)

13. What is petrick method? Find all the prime implicants of the following expression and then find the all minimum sum of product expression using petricks method

F(A,B,C,D)= ∑m(9,12,13,15)+ ∑dc(1,4,5,7,8,11,14)

14. Using the Method of Map entered variables, use four variable map to find minimum sum of product expression for

(i) F(A,B,C,D,E)= ∑m(0,4,5,7,9)+ ∑dc(6,11) + E(m1+m15), where m represents minterm.
(ii) Z(A,B,C,D,E,F,G)= ∑m(0,3,13,15)+ ∑dc(1,2,7,9,14) +E(6,8)+F(12)+G(5)
(iii) F(A,B,C,D,E)= ∑m(0,4,6,13,14)+ ∑dc(2,9) + E(1,12)

15. Plot the following function on K map

(i) F(A,B,C,D) = BD' + B' CD + ABC + ABC' D + B' D'

(ii) F(A,B,C) = A'B' + BC' + AC

(iii) F(A,B,C) = B'C + BC' + A'

16. circuitby using NAND gates only.
F(A,B,C,D)= ∑m(7,9,10,11,12,13,14,14)

17. Find the minimal SOP of the following Boolean function using K-
Maps(i) F(A,B,C,D)= ∑m(6,7,9,10,13)+ ∑dc(1,4,5,11)
(ii) F(A,B,C,D)= ΠM(1,2,3,4,9,10)+ Πdc(0,1,4,15)

18. Simplify the given Boolean function by using Kmap method in POS form. Realize logic circuit by usingNAND gates only.
F(A,B,C,D)= ∑m(0,1,2,3,4,5,7)

PART – B

Experiment: 4

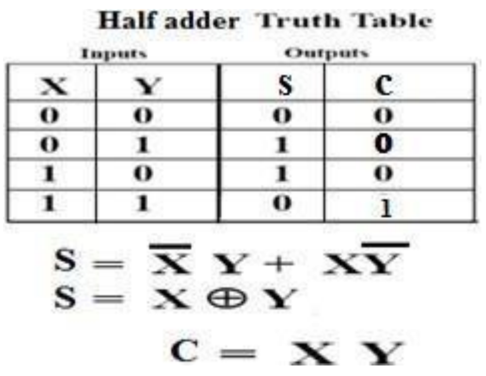
ADDER and SUBTRACTOR

Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates. And implement the same in HDL.

Components required: IC-7404, IC-7408, IC-7432, IC-7411, and Patch Chords.

Half Adder

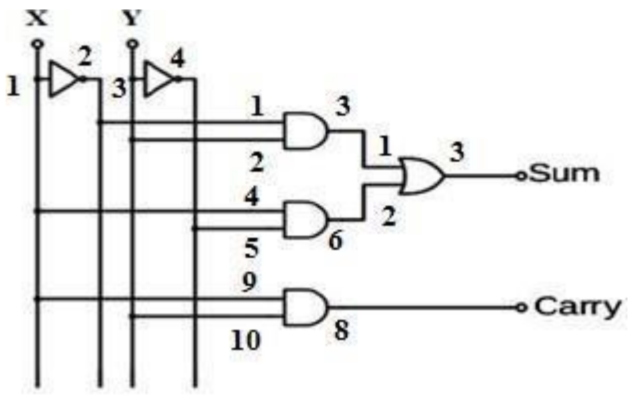
- Combinational circuit is a circuit in which we combine the different gates in thecircuit
- Half adder is a combinational logic circuit with two inputs and twooutputs.
- The half adder circuit isdesigned to add two single bit binary numbers X andY.
- This circuit has two outputs **carry** and **sum**.



Half adder Truth Table

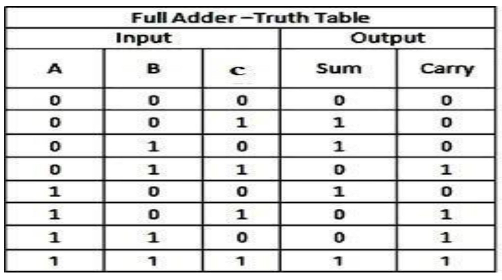
Inputs		Outputs	
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \overline{X}Y + X\overline{Y}$$
$$S = X \oplus Y$$
$$C = XY$$



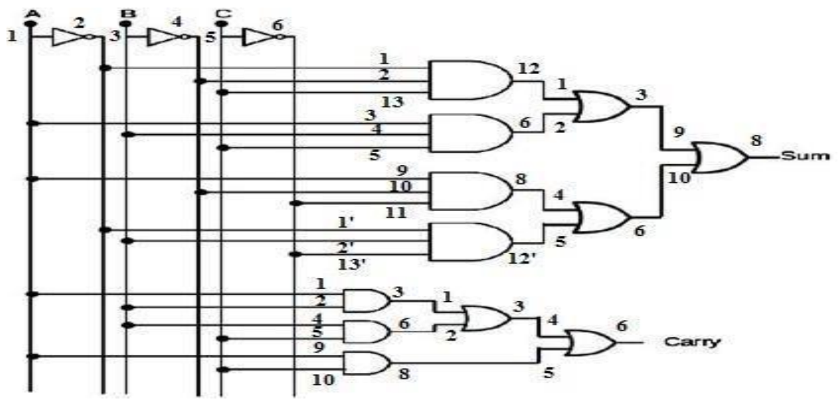
Full Adder

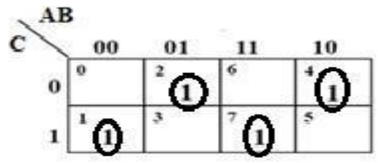
- Full adder is developed to overcome the drawback of Half Addercircuit.
- It can add two one-bit numbers A and B, and carry C.
- It consists of ***three inputs and two outputs***, two inputs are the bits to be added, the third input represents the carry form the previousposition.
- The Sum is equal to 1 when only one input is equal to 1 or when all three inputs are equal to1 andthe output has a carry 1 if two or three inputs are equal to1.
- The Karnaugh maps and the simplified expression are shown in the followingfigures:



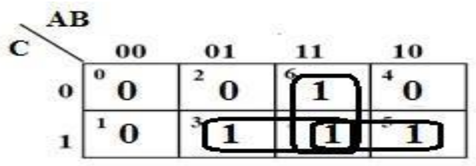
Full Adder –Truth Table

Input			Output	
A	B	c	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1





$$Sum = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$



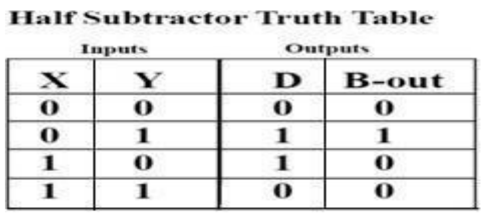
$$Carry = A\,B + A\,C + B\,C$$

Half Subtractor

- Subtracting a single-bit binary value Y from anther X(i.e. X -Y) produces a difference bit D and borrowout bitB-out.

Full Subtractor:

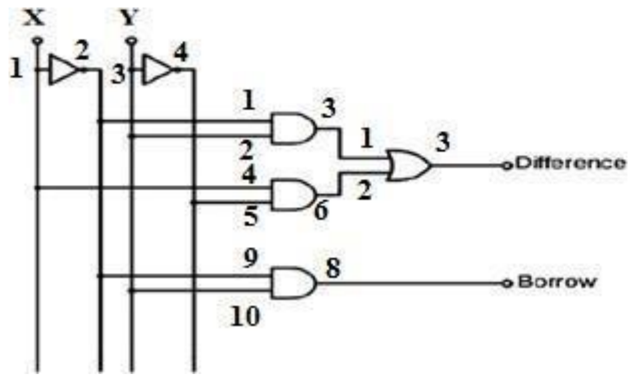
- The disadvantage of a half subtractor is overcome by fullsubtractor.



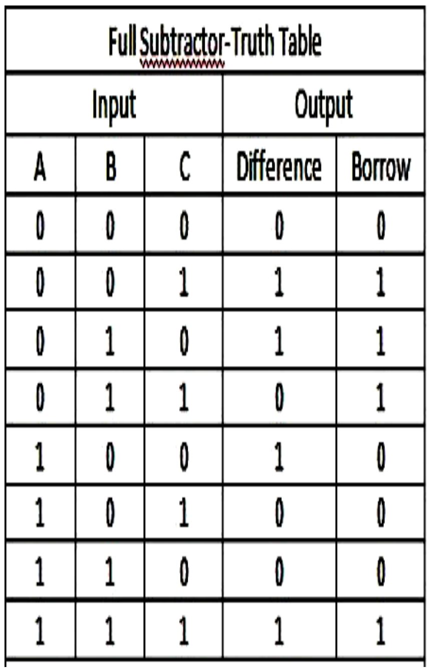
Half Subtractor Truth Table

Inputs		Outputs	
X	Y	D	B-out
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$D = \bar{X}\,Y + X\,\bar{Y}$$
$$D = X \oplus Y$$
$$B\text{-}out = \bar{X}\,Y$$

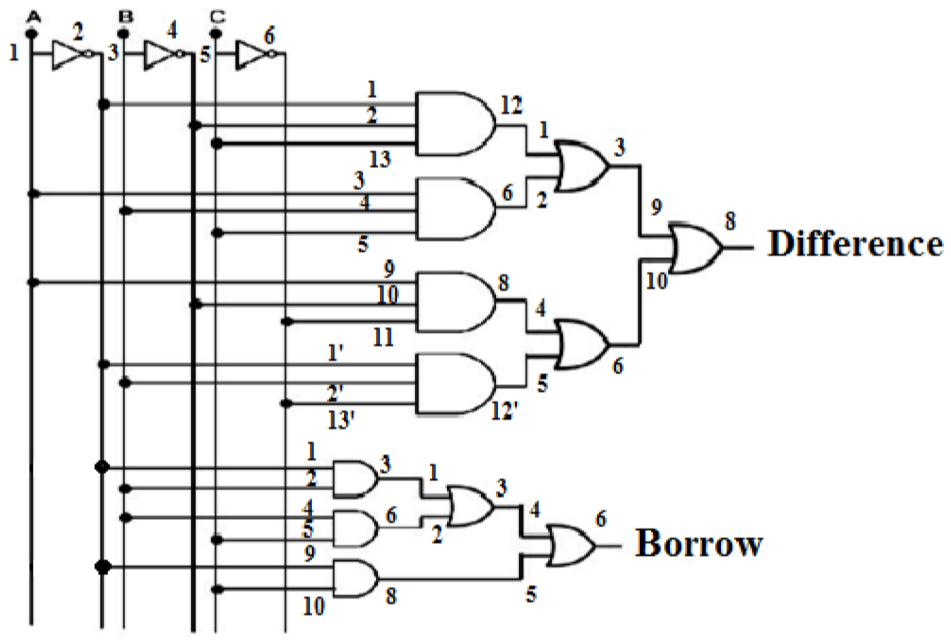


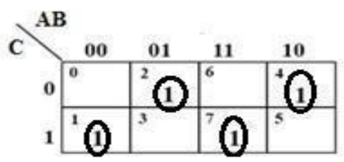
- The full subtractor is acombinational circuit with three inputs A, B, C and two outputs Difference and Borrow. A is the 'minuend', B is subtrahend', C is the 'borrow' produced by the previous stage, Difference is the difference output and Borrow isthe borrowoutput.



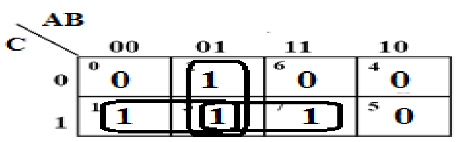
Full Subtractor-Truth Table

Input			Output	
A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1





$$Difference = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$



$$Borrow = \bar{A}\,C + \bar{A}\,B + B\,C$$

Meghana Sambare R

Simulation using VHDL:

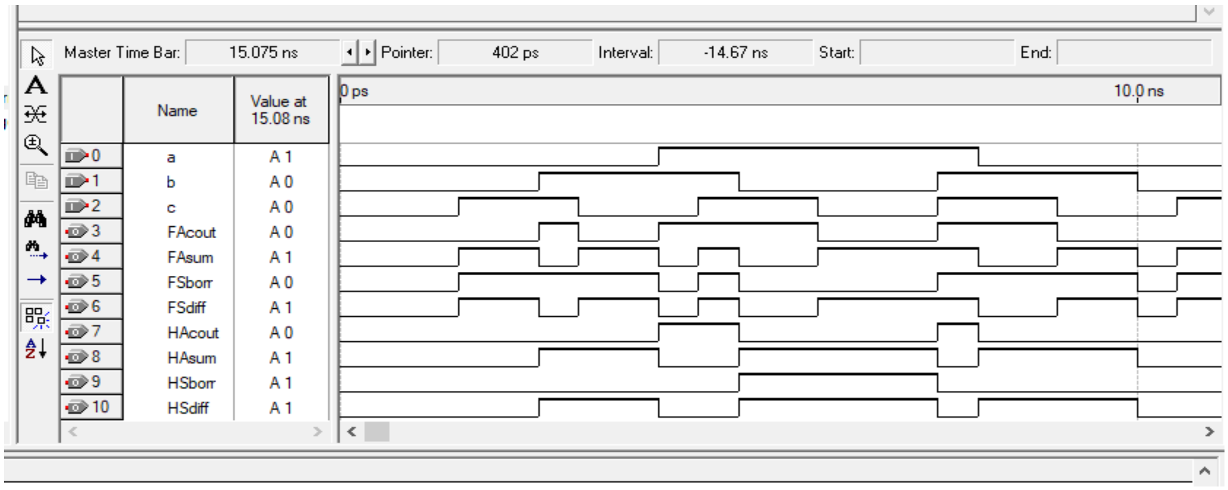
Simulation: VHDL code for adder, subtractor

```
library ieee;
use ieee.std_logic_1164.all;

entity adder is
   port(a,b,c: in std_logic;
          HAsum, HAcout, FAsum, FAcout, HSdiff, HSborr, FSdiff, FSborr: out std_logic); end
adder;

architecture dataflow of adder is
begin
HAsum<= a xor b;
   HAcout<= a and b;
   FAsum<= a xor b xor c;
   FAcout<= ((a and b)or(b and c) or(a and c));
   HSdiff<= a xor b;
   HSborr<= (a and (not b));
   FSdiff<= a xor b xor c;
   FSborr<= ((b xor c) and (not a)) or (b and c);
   end dataflow;
```

Waveform



Master Time Bar: 15.075 ns   Pointer: 402 ps   Interval: -14.67 ns   Start: End:

	Name	Value at 15.08 ns
0	a	A 1
1	b	A 0
2	c	A 0
3	FAcout	A 0
4	FAsum	A 1
5	FSborr	A 0
6	FSdiff	A 1
7	HAcout	A 0
8	HAsum	A 1
9	HSborr	A 1
10	HSdiff	A 1

Note: File name, project name, entity name should be same

MODULE 3 **Combinational circuit design and simulation using gates Multiplexers, Decoders and Programmable Logic Devices**

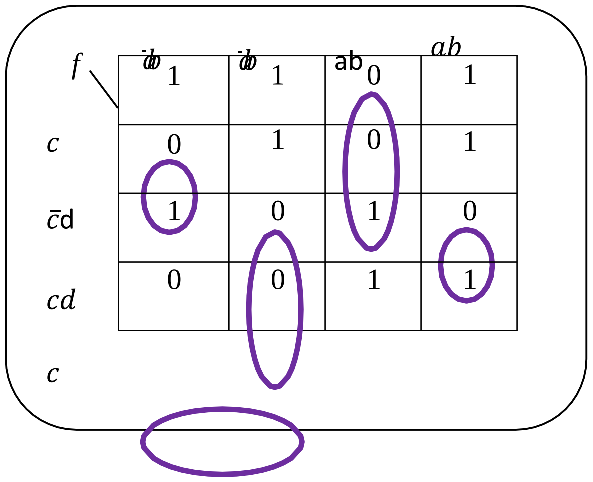
REVIEW OF COMBINATIONAL CIRCUIT DESIGN:

1. Steps involved in the design of a combinational switching circuit:
2. Set up a truth table which specifies the output(s) as a function of the input variables. If a given combination of values for the input variables can never occur at the circuit inputs, the correspondingoutput values are don't-cares.
3. Derive simplified algebraic expressions for the output functions using Karnaugh Maps, or Quine-McCluskey method, or any other similar procedure. The resulting algebraic expressions are then manipulated into the proper form, depending on the type of gates to be used in realizing the circuit.
4. When a circuit has two or more outputs, common terms in the output functions can often be used to reduce the total number of gates or gate inputs.
5. Minimum two-level AND-OR, or NAND-NAND circuits can be realized using the minimum sum- of-products. Minimum two-level OR-AND, or NOR-NOR circuits can be realized using the minimum product-of-sums.

DESIGN OF CIRCUITS WITH LIMITED GATE FAN-IN:

In practical logic design problems, the maximum number of inputs on each gate (or the fan-in) is limited. Depending on the type of gates used, this limit may be two, three, four, eight, or some other number. If a two-level realization of a circuit requires more gate inputs than allowed, factoring the logic expression to obtain a multi-level realization is necessary.

Example: *Realize* $(a, b, c, d) = \Sigma m(0, 3, 4, 5, 8, 9, 10, 14, 15)$ *using three input NOR gates.Solution:*



The product-of-sum equation is:

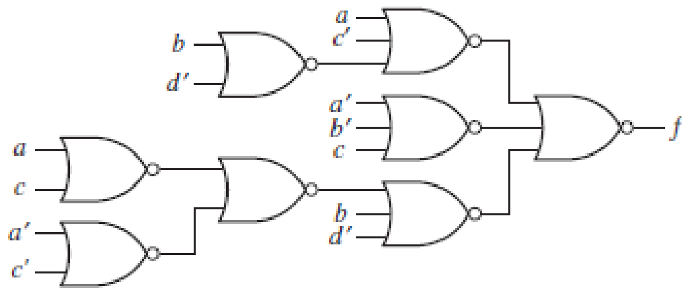
$f = (a' + b' + c)(a + b' + c')(a + c' + d)(a + b + c + d')(a' + b + c' + d')$

As can be seen from the preceding expression, a two-level realization requires three-three input gates, twofour-input gates and one five-input gate. The expression for f' is factored to reduce the maximum number of gate inputs to three and, then, it is complemented.

Or $f' = abc' + a'bc + a'cd' + a'b'c'd + ab'cd$

i.e., $f' = abc' + a'(b + d') + b'd(a'c' + ac)$

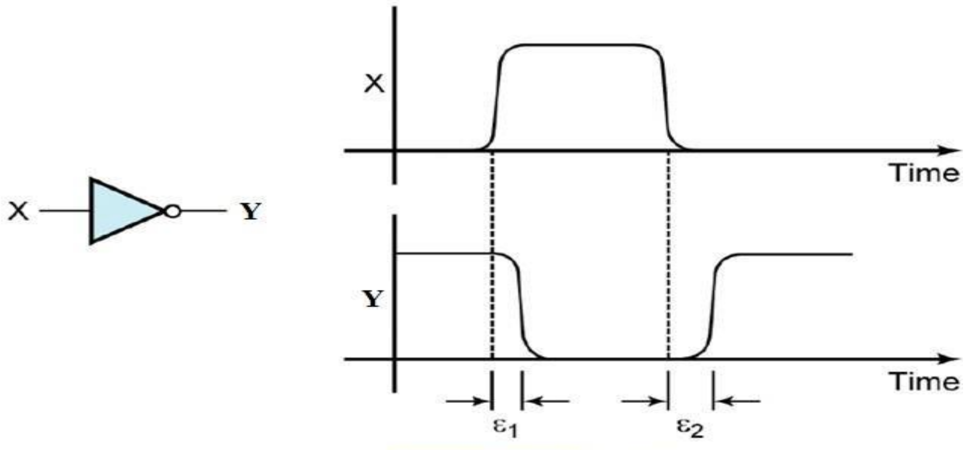
Or $f = [(a' + b' + c)][(a + c') + (b'd)][(b + d') + (a + c)(a' + c')]$



❖ Combinational logic circuit is a circuit in which we combine the different logic gates in the circuit.

Gate Delays and Timing Diagrams

- When the input to a logic gate is changed, the output will not change instantaneously.
- The transistors or switching elements within the gate take a finite time to react to a change in input, so that gate output is delayed with respect to the input change.
- Such a delay is called as propagation delay **(ε)** of the logic circuit.
- Figure below shows possible input and output waveforms for an inverter.



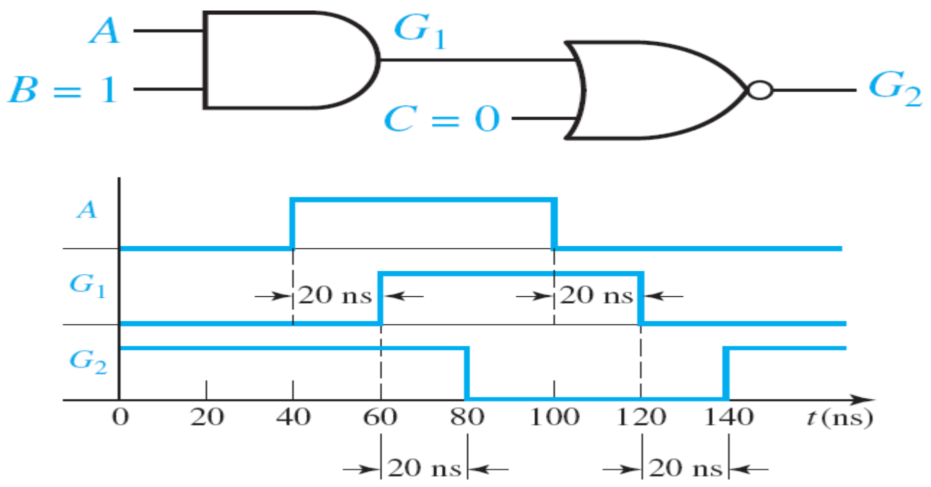
Propagation Delay in an Inverter

- In practice, the propagation delay for a 0 to 1 output change may be different than the delay for a 1 to 0 change.
- Propagation delays for integrated circuit gates are in few nanoseconds and in many cases these delays

can be neglected.

- Timing diagrams show various signals in the circuit as a function of time or Timing diagrams show various input and output waveform of a logic circuit.

Problem-1] Draw the timing diagram for the circuit shown below assume that each gate has a propagation delay of 20ns.
Soln:



Timing Diagram for AND-NOR Circuit

- The output of gate **G1** changes 20 ns after **A** changes, and the output of gate **G2** changes 20 ns after **G1** changes.

Hazards in Combinational Logic

- When the input to a combinational circuit changes, unwanted switching transients may appear in the output.
- These transients occur when different paths from input to output have different propagation delays.
- Unwanted glitches due to finite propagation delay of logic circuit called as hazard.
- Additional gates in logic circuit preventing hazard called as hazard cover.

Types of Hazard:

Static-1 Hazard:-

- In response to any single input change and for some combination of propagation delays, a circuit output may momentarily go to 0 when it should remain a constant 1, we say that the circuit has a static 1-hazard.
- A condition Y = A+A` should always generate 1 at the output, i.e. static-1.
- But the NOT gate output takes finite time to become 1 (transition from 0 to 1).
- Thus for the OR gate there are two zeros appearing at its input for that small duration, resulting a 0 at its output.
- The width of this zero is in nanosecond and is called a glitch. This may cause malfunctioning of circuit.

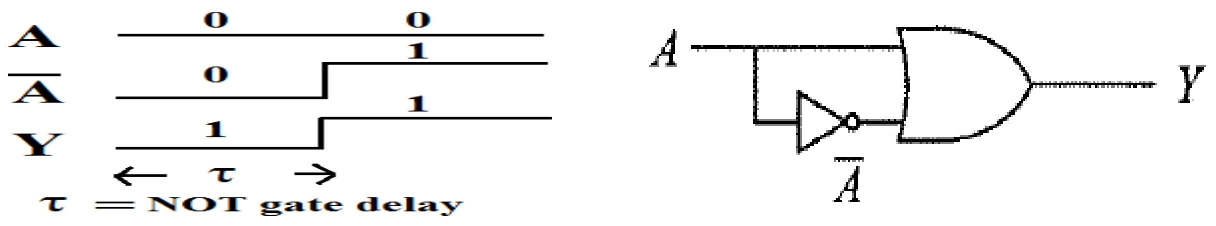
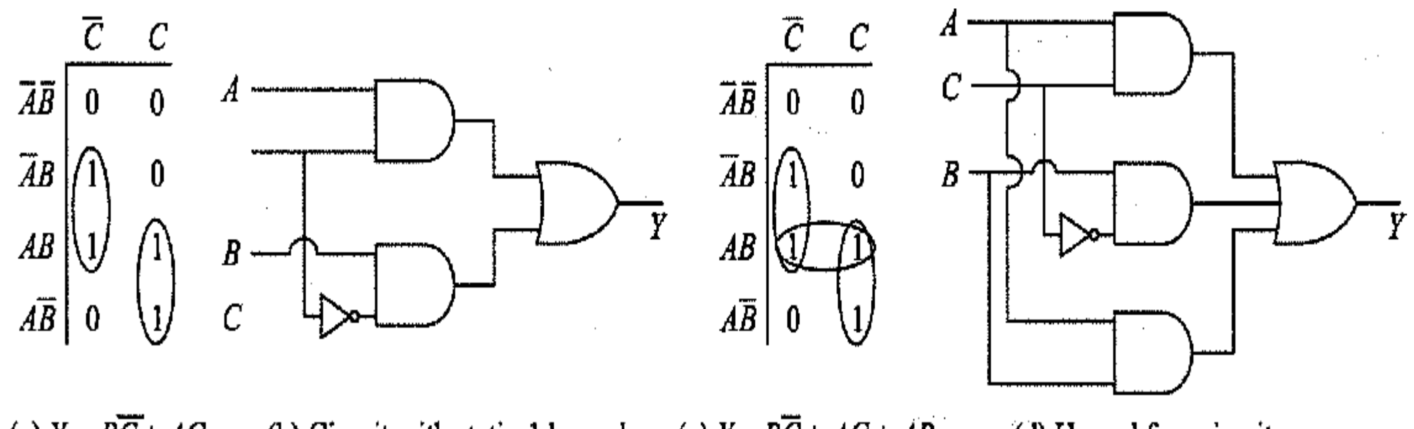


Fig: Static-1 Hazard

- To illustrate Static-1 hazard, let consider the following K-Map(Fig.a) , which is represented by $Y = B\overline{C} + AC$.
- The corresponding circuit is shown in fig.b. Consider, for this circuit input B=1 and A=1 and then C = 0. The output shows glitch.
- Consider another grouping for the same map in Fig.c. This includes one additional term AB and now output $Y = B\overline{C} + AC + AB$.
- The corresponding circuit diagram is shown in Fig.d. This circuit though require more hardware than previous one and,it is hazard free. The additional term AB ensures Y=1.

Fig: Static-1 hazard and its covers



(a) $Y = B\overline{C} + AC$ (b) Circuit with static-1 hazard (c) $Y = B\overline{C} + AC + AB$ (d) Hazard free circuit

Static-0 Hazard:-

- In response to any single input change and for some combination of propagation delays, circuit output may momentarily go to 1 when it should remain a 0, we say that the circuit has a static 0-hazard.
- An A.A’ condition should always generate 0 at the output, i.e. static-0.

- But the NOT gate output takes finite time to become 0. Thus for final AND gate there are two ones appearing at its input for a small duration resulting a 1 at its output .
- This Y=1 occurs for a very small duration (few nanosecond) but may cause malfunctioning of circuit.

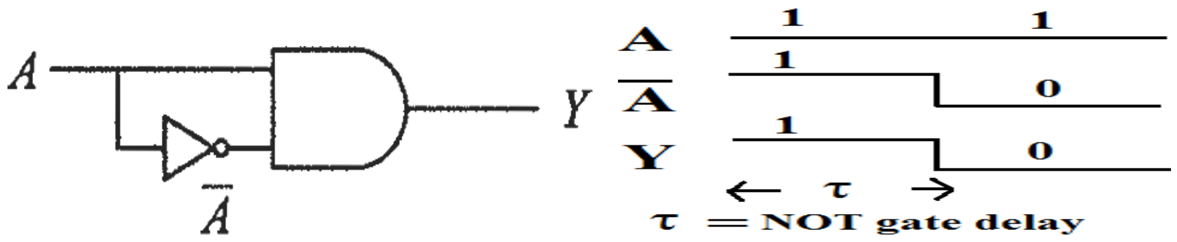


Fig: Static-0 Hazard

How to prevent Static-0 Hazard:-

- In the given K-Map, Group 0s such that a POS form results. Fig-a shows the minimal cover in POS form that gives Y= (B+C)(A+C') and corresponding circuit in Fig-b.
- To prevent this we add one additional group, i.e. one more sum term (A+B) as shown in Fig-c and the corresponding circuit is shown in Fig-d. The additional term (A+B) ensures Y=0

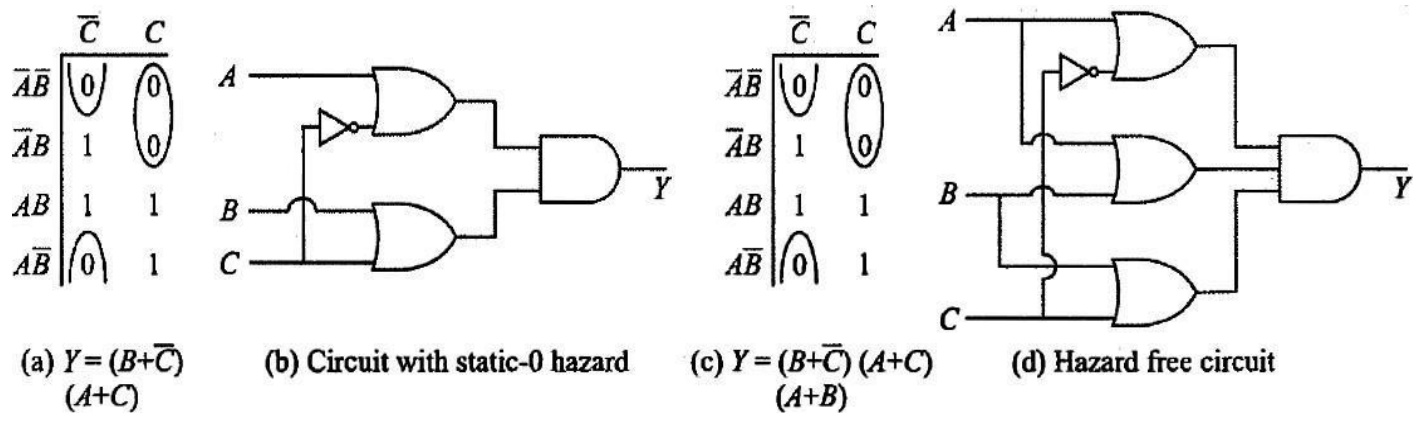
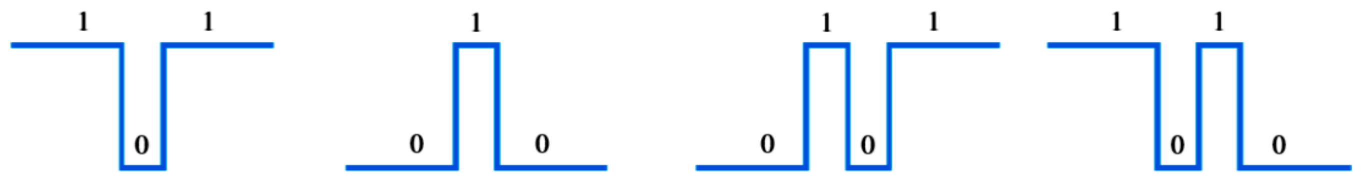


Fig: Static-0 hazard and its cover

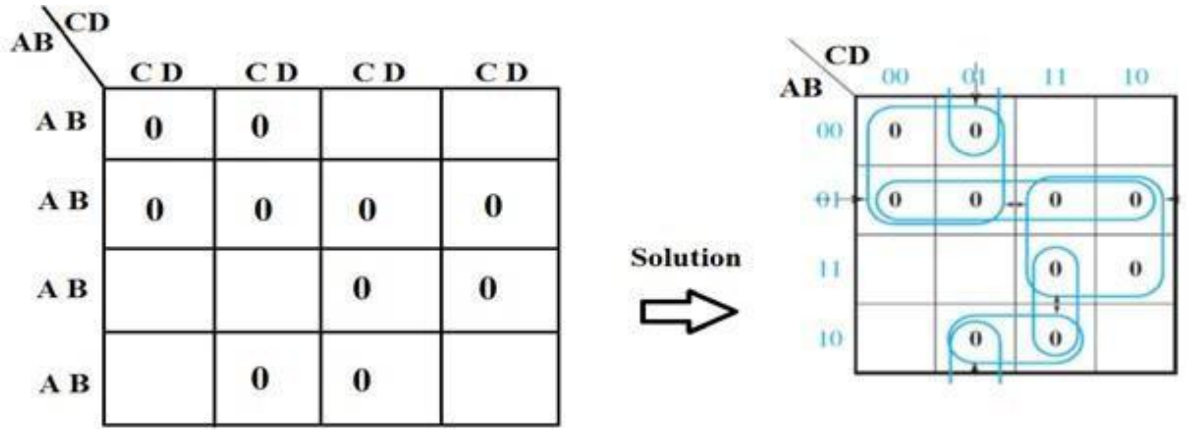
Dynamic Hazard:-

- Dynamic hazard occurs when circuit output makes multiple transitions before it settles to a final value.
- In response to any single input change and for some combination of propagation delays, a circuit output is supposed to change from 0 to 1 (or 1 to 0), the output may change three or more times, we say that the circuit has a dynamic hazard.
- The output of logic equation in dynamic hazard degenerates into Y=A+A'.A or Y=(A+A').A

Types of Hazards



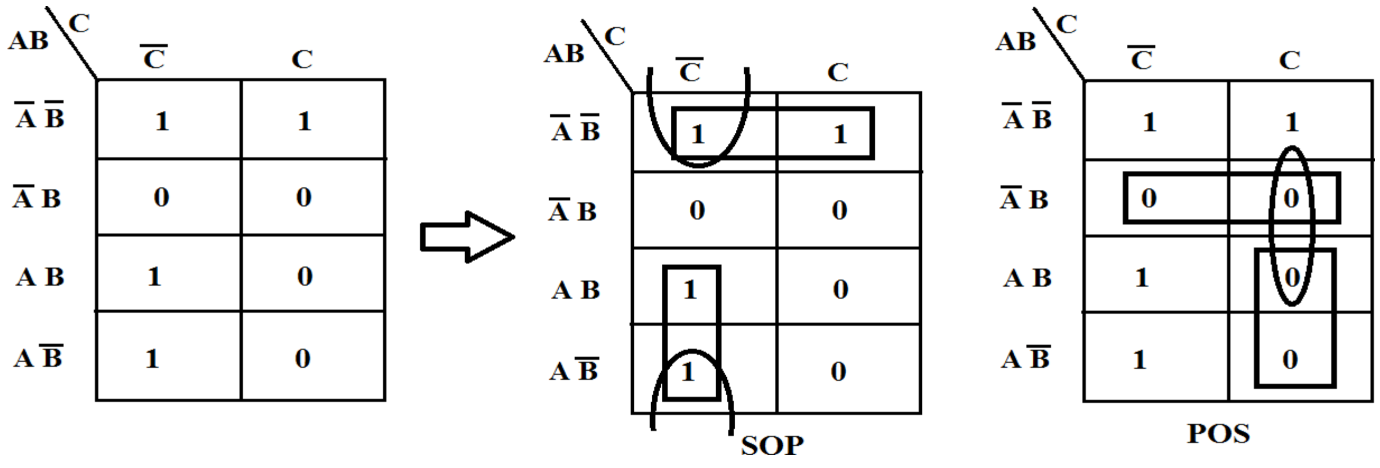
(a) Static 1-hazard (b) Static 0-hazard (c) Dynamic hazards



$$F = (A + C)(A' + D')(B' + C' + D)(C + D')(A + B' + D)(A' + B' + C')$$

Problem-2] For a given K map give the static -0 hazard free circuit and POS expression?

Problem-3] For the following K-map given below give SOP &POS form that do not shows static 0 or static 1 hazard .



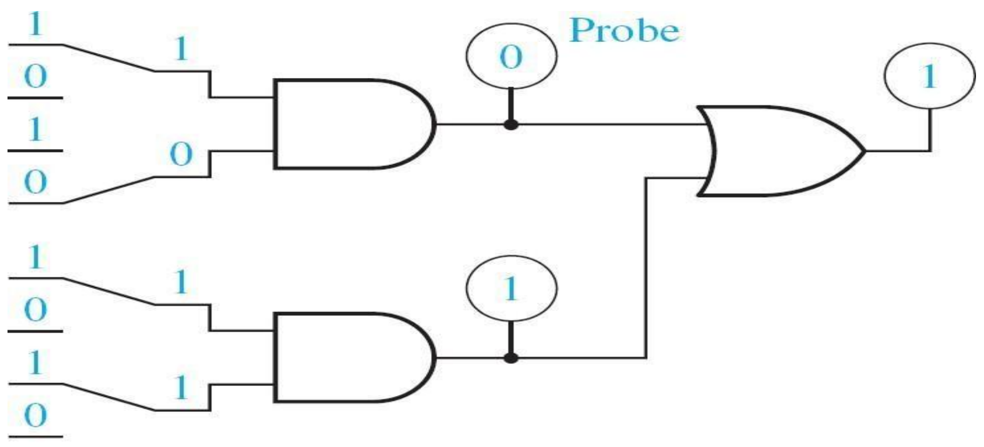
$$\text{(SOP) } Y = \overline{A}\,\overline{B} + A\overline{C} + \overline{B}\,\overline{C} \;,\quad \text{(POS)} Y = (A + \overline{B})(\overline{A} + \overline{C})(\overline{B} + \overline{C})$$

Simulation and Testing of Logic Circuits:

- ❖ An important part of the logic design process is verifying the final design is correct and debugging the design if necessary.
- ❖ Logic circuits may be tested by actually building them or by simulating them on a computer.
- ❖ To use a computer program for simulating logic circuits, you must first specify the circuit components and connections; then, specify the circuit inputs; and, finally, observe the circuit outputs.
- ❖ A simple simulator for combinational logic works as follows:

  1. The circuit inputs are applied to the first set of gates in the circuit, and the outputs of those gates are calculated.
  2. If the input to any gate has changed, then the output of that gate is calculated. The outputs of the gates which changed are fed into the next level of gate inputs
  3. Step 2 is repeated until no more changes in gate inputs occur. The circuit is then in a steady-state condition and the outputs may be read.
  4. Steps 1 through 3 are repeated every time a circuit input changes.

- ❖ The two logic values, 0 and 1, are not sufficient for simulating logic circuits. At times, the value of a gate input or output may be unknown, and we will represent this unknown value by X.
- ❖ Figure below shows a typical simulation screen on a personal computer. The switches are set to 0 or 1 for each input. The probes indicate the value of each gate output.



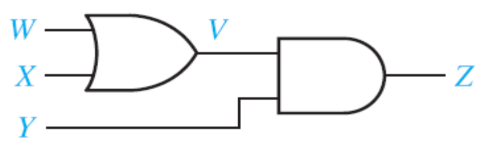
Simulation screen showing switches

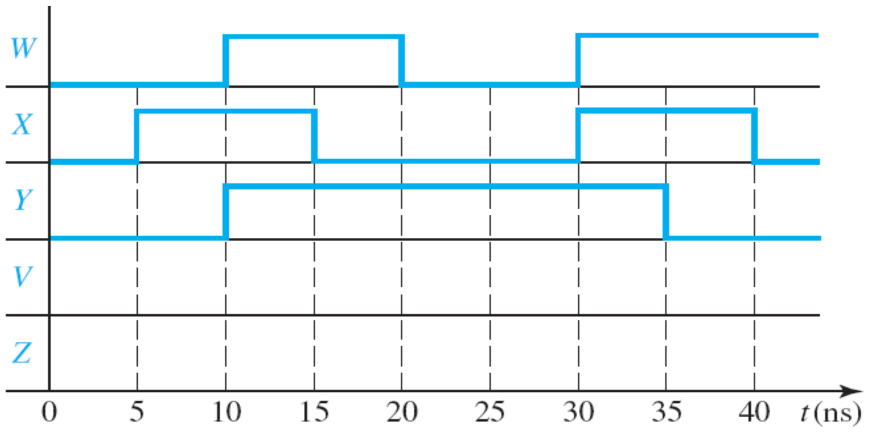
- If a circuit output is wrong for some set of input values, this may be due to several possible causes:
  1. Incorrect design
  2. Gates connected wrong
  3. Wrong input signals to the circuit
  4. Defective gates
  5. Defective connecting wires

Assignments:

1. Consider the following logic function. *F*(*A*, *B*, *C*, *D*)_ *m*(0, 4, 5, 10, 11, 13, 14, 15)
   (a) Find two different minimum circuits which implement *F* using AND & OR gates. Identify two hazards in each circuit.
   (b) Find an AND-OR circuit for *F* which has no hazards.
   (c) Find an OR-AND circuit for *F* which has no hazards.
2. What is hazard? Explain different types of hazard with an example.
3. What is Combinational circuit? Explain with example simulation and testing of logic circuit with an example?
   Explain with an example propagation delay and timing diagram.

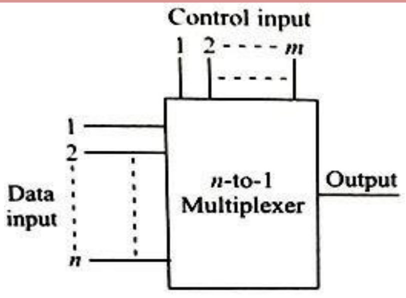
4. Complete the timing diagram for the circuit. Assume that both gates have propagation delay of 5 ns.





Multiplexers:

- A multiplexer is a circuit with **many inputs but only one output by applying control signals we can steer any input to the output.**
- A multiplexer (or data selector, abbreviated as MUX) has a group of data inputs and a group of control inputs. The control inputs are used to select one of the data inputs and connect it to the output terminal.
- The block diagram of multiplexer has '**n**' input signals, '**m**' control signals and**'1'** o/p signal.

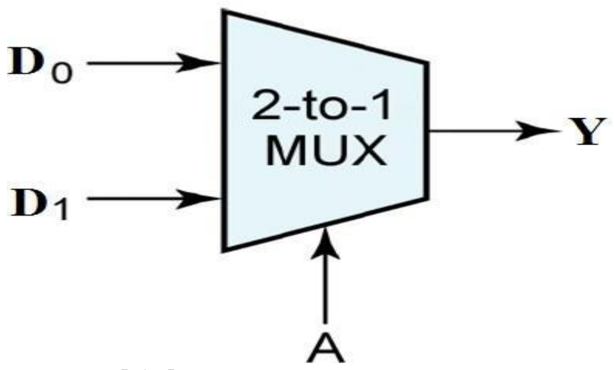


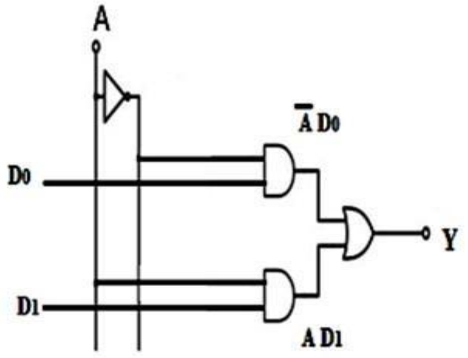
2 to 1 Multiplexer:

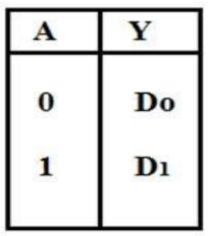
Multiplexer block diagram2 to 1 multiplexer logic circuit and its truth table

- The circuit diagram of a 2 to 1 Multiplexer is shown above with its truth table, depending on control signal ‘**A’** one of the two inputs (**D_0** and **D_1**) is steered (selected) to output ‘**Y’.**
- Logic equation for this circuit is given by

$$Y = \overline{A}\, D_0 + A D_1$$





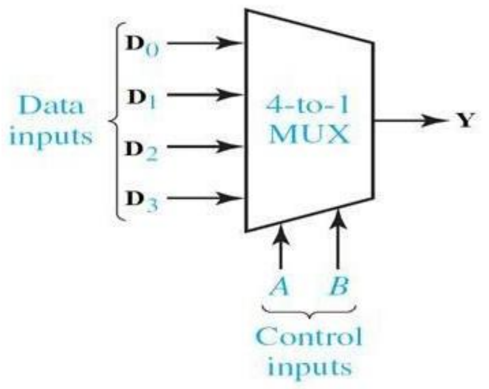


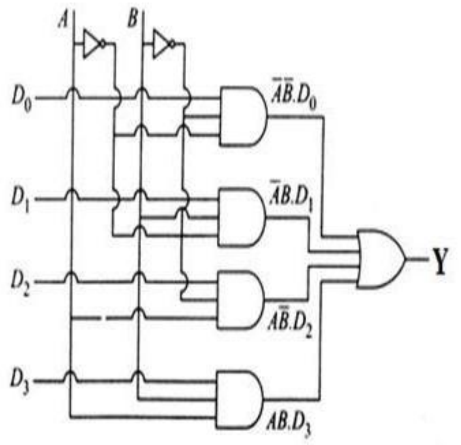
A	Y
0	D_0
1	D_1

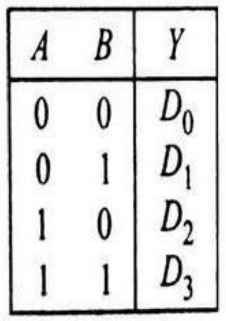
4 to 1 Multiplexers:

- The circuit diagram of a 4 to 1 Multiplexer is shown below with its truth table, depending on control signal **A, B** one of the four inputs (**D_0**to**D_3**) is steered(selected) to output **Y.**

- Logic equation for this circuit is given by







A	B	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

4 to 1 multiplexer logic circuit and its truth table

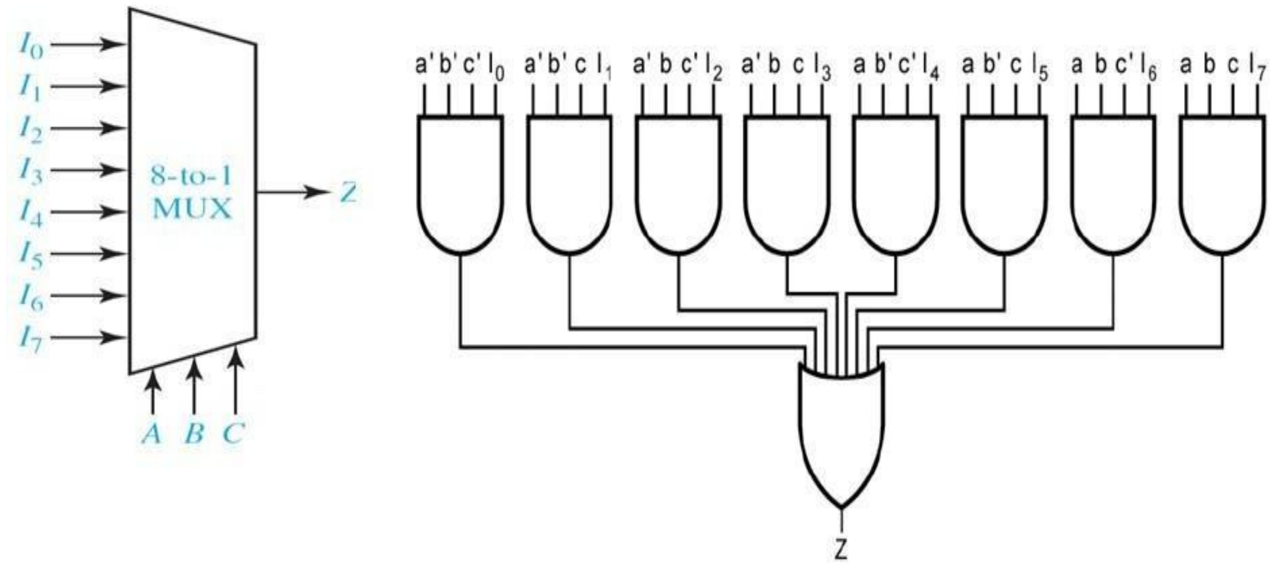
- Logic equation for this circuit is given by

$$Y = \overline{A}\,\overline{B}\,D_0 + \overline{A}\,B\,D_1 + A\,\overline{B}\,D_2 + A\,B\,D_3$$

- For instance if A=0, B=0, **D_0 is steered to the output.**
  $Y = 0'.0'.D_0+0'.0\ .D_1 + 0.0'.D_2 + 0.0\ .D_3$
  $Y = 1.\ 1.D_0+ 0'.0.\ D_1+ 0.0'.D_2+ 0.0.D_3$
  $Y = D_0$

8 to 1 Multiplexers:

- The circuit diagram of 8 to 1 Multiplexer is shown below with its truth table, depending on control signal **ABC** one of the eight inputs (**I_0** to **I_7**) is steered (selected) to output **Z.**
- **For instance, when ABC= 000, Z = I_0**
-

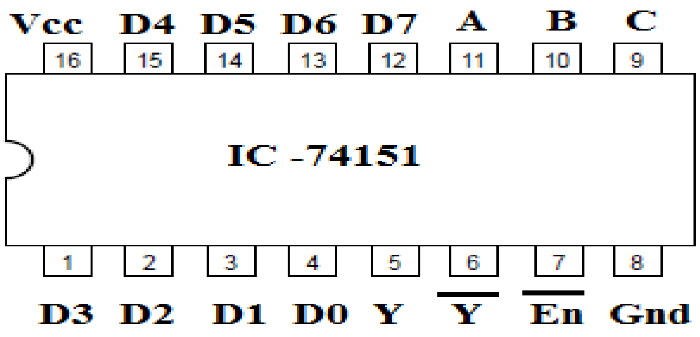


- Logic equation for this circuit is given by

$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

- IC 74151 is commercial available for 8:1 MUX

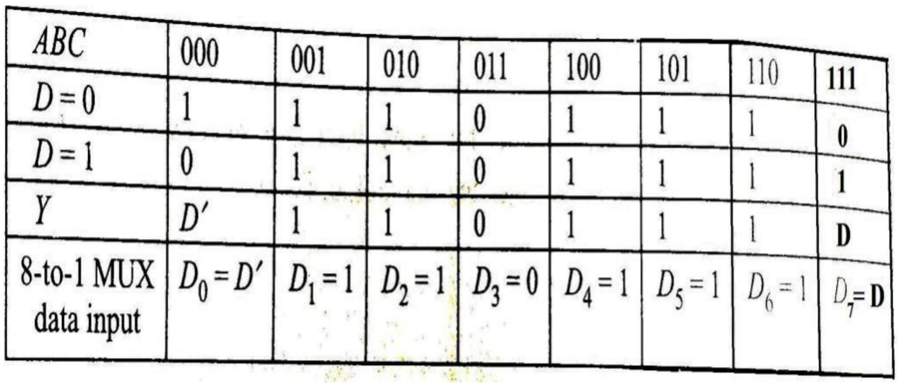
Pin Diagram of 74151:



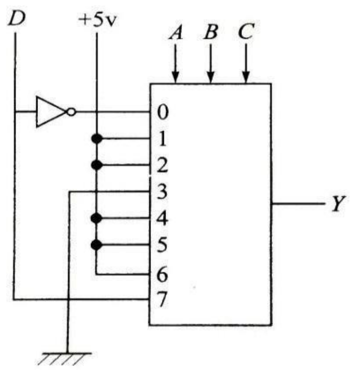
Universal Logic Circuits

- Multiplexer is sometimes called universal logic circuits because a 2^n-to-1 multiplexer can be used as a design solution for any n variable truth table.
- We can realize 16-to-1 multiplexer using 8-to-1 multiplexer.
- Let consider the truth table given below.
- A,B,C variables to be fed as select inputs, the fourth variable D then has to be present as data input. The method is shown in figure below by using entered variable map.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1
1	1	1	1	1



ABC	000	001	010	011	100	101	110	111
$D=0$	1	1	1	0	1	1	1	0
$D=1$	0	1	1	0	1	1	1	1
Y	D'	1	1	0	1	1	1	D
8-to-1 MUX data input	$D_0=D'$	$D_1=1$	$D_2=1$	$D_3=0$	$D_4=1$	$D_5=1$	$D_6=1$	$D_7=D$



- Write all the combinations of 3 select inputs in first row along different columns.

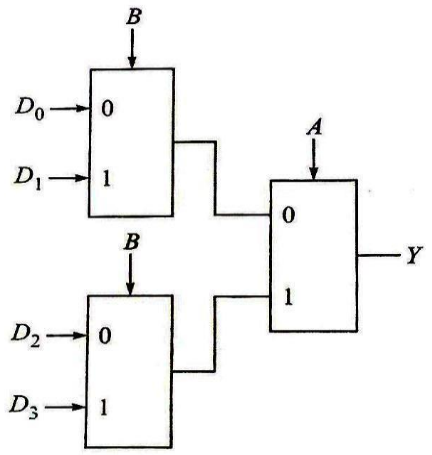
- Now corresponding to each value of 4th variable D, truth table output is written in 2nd and 3rd row. The 4th row writes Y as a function of D. In fifth row we assign data input values for 8-to-1 multiplexer simply copying Y values obtained in previous row. This is because for each select variable combination a multiplexer transfers a particular input to its output.
- In 8-to-1 multiplexer ABC=000 selects D_0, ABC=001 selects D_1 and so on.

Prob-4) Show how 4-to-1 multiplexer can be obtained using only 2-to-1 multiplexer.

Soln: Logic equation for 2-to-1 Multiplexer: $Y = \overline{A}\, D_0 + A D_1$

Logic equation for 4-to-1 Multiplexer: $Y = \overline{A}\,\overline{B}\, D_0 + \overline{A}\, B\, D_1 + A\,\overline{B}\, D_2 + A\, B\, D_3$

This can be rewritten as, $Y = \overline{A}\,(\overline{B}\,.\, D_0 + B.\, D_1) + A\,(\overline{B}.\, D_2 + B.\, D_3)$



Prob-5) Design a 32-to-1 multiplexer using two 16-to-1 multiplexer and one 2-to-1 multiplexer.

Soln: A 32-to-1 multiplexer requires $\log_2 32 = 5$ select lines say, ABCDE.

The lower 4 select lines BCDE chose 16-to-1 multiplexer outputs.

The 2-to-1 multiplexer chooses one of the output of two 16-to-1 multiplexers depending on what appears in the 5th select line, A.

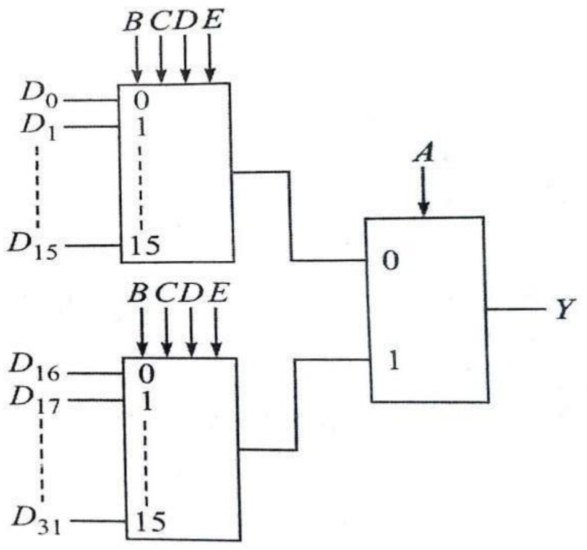
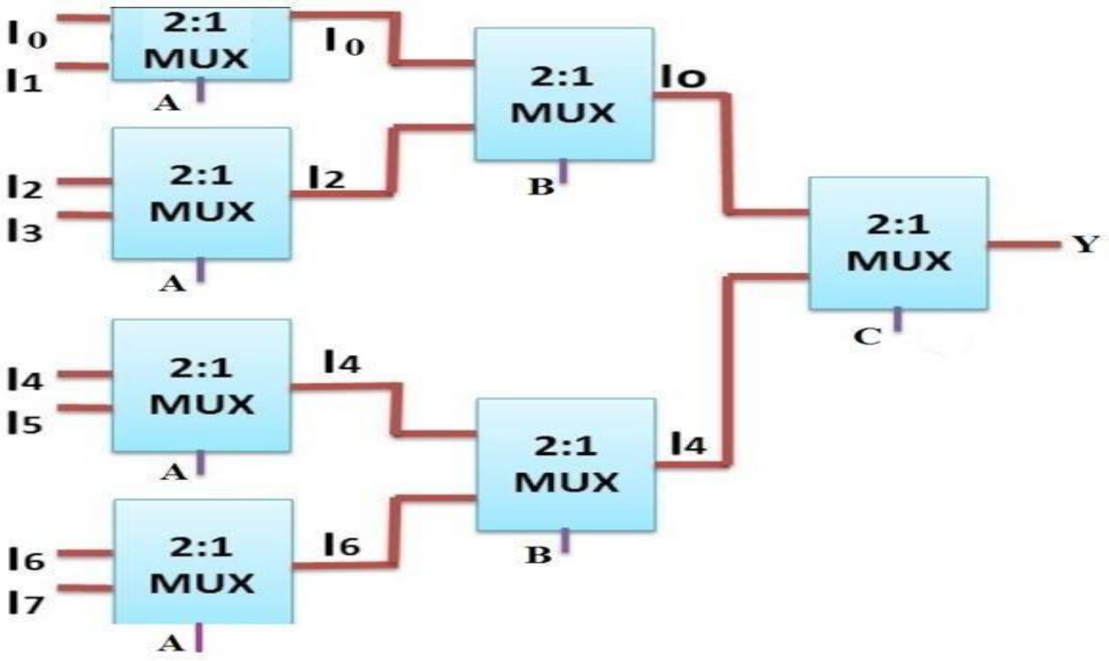


Fig: Realization of higher order multiplexers using lower orders

Prob-6) Design 8-to-1 multiplexer using 2-to-1 multiplexer.

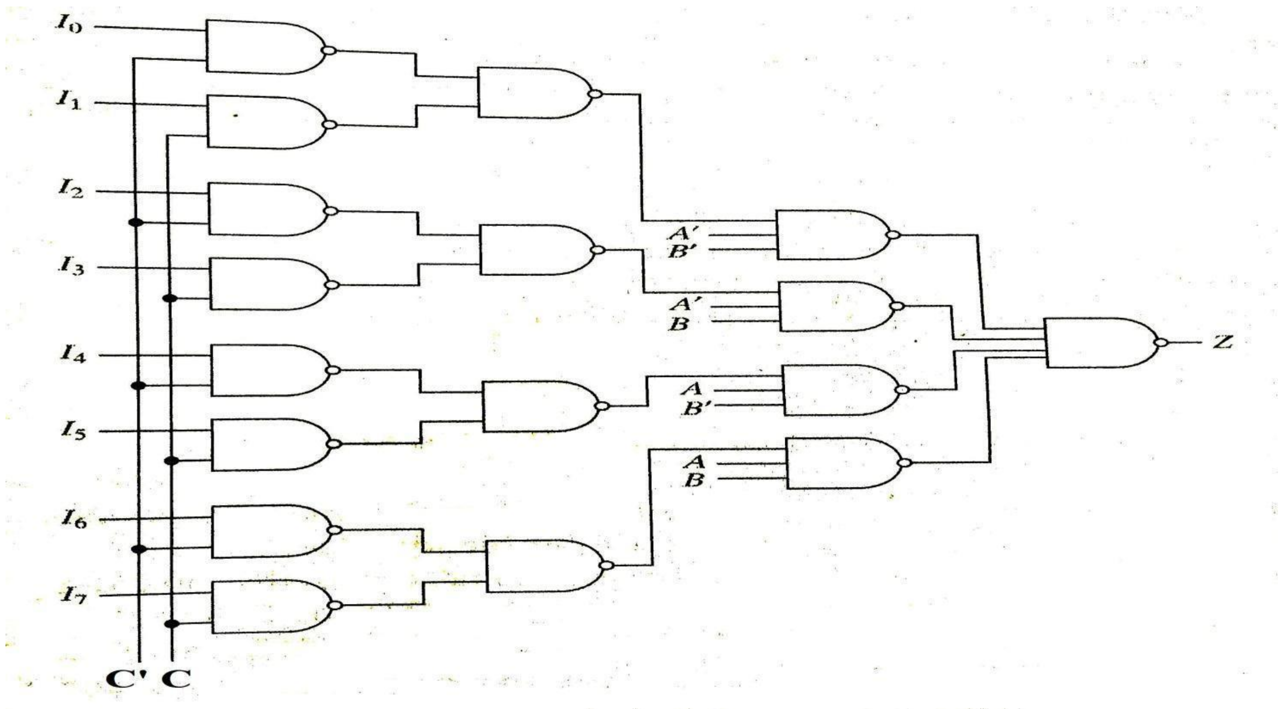


Prob-7) Design 8-to-1 multiplexer using NAND gate.

Soln:

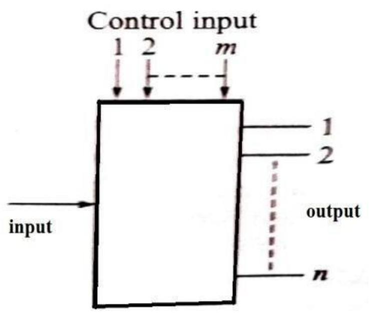
$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

$$Z = A'B'(C'I_0 + CI_1) + A'B(C'I_2 + CI_3) + AB'(C'I_4 + CI_5) + AB(C'I_6 + CI_7)$$



DEMULTIPLEXERS

- A de-multiplexer is a logic circuit with one input and many outputs by applying control signals, we can steer the input signal to one of the output lines.
- Block diagram of de-multiplexer is shown below.
- The circuit has 1 input signals , ‘m’ control signals or select signals, ‘n’ output signals, where $n<=2^m$



De-multiplexer Block diagram

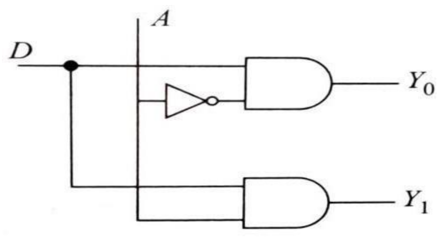
1- to-2 De-multiplexer

- 1-to-2 de-multiplexer has one data input ‘D’,1 control signal ‘A’ and 2 outputs Y_0 and Y_1

Logic circuit of 1-to-2 De-multiplexer

- When A = 0, upper AND gate is enabled and lower AND gate is disabled therefore the data bit D is transmitted only to the Y0 giving Y0 = D.

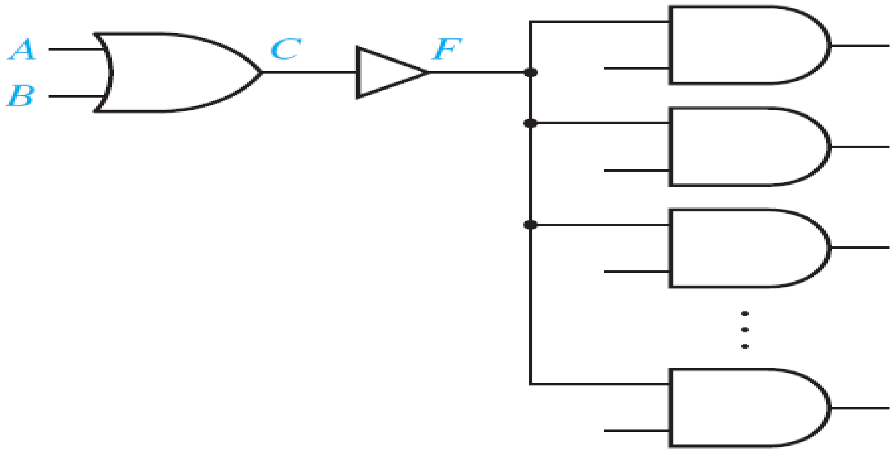
- Similarly When A = 1, upper AND gate is disabled and lower AND gate is enabled therefore the data bit D is transmitted only to the Y1 giving Y1 = D.



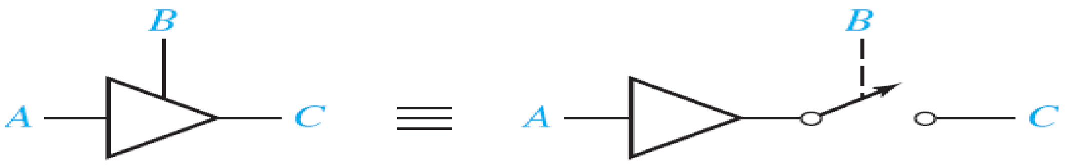
Three-State Buffers:

- ❖ A gate output can only be connected to a limited number of other device inputs without degrading the performance of a digital system.
- ❖ A simple buffer may be used to increase the driving capability of a gate output. Figure shows a buffer connected between a gate output and several gate inputs.

- ❖ Normally, a logic circuit will not operate correctly if the outputs of two or more gates or other logic



devices are directly connected to each other.
- ❖ For example, if one gate has a 0 output (a low voltage) and another has a 1 output (a high voltage), when the gate outputs are connected together the resulting output voltage may be some intermediate value that does not clearly represent either a 0 or a 1.
- ❖ Use of three-state logic permits the outputs of two or more gates or other logic devices to be connected together.
- ❖ Figure below shows a three-state buffer and its logical equivalent.

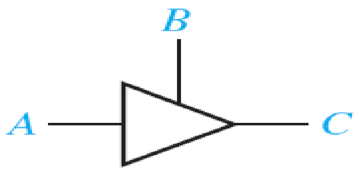


Three-State Buffer

- When the enable input *B* is 1, the output *C* = *A*; when *B* is 0, the output *C* acts like an open circuit, when *B* is 0, the output *C* is disconnected from the buffer output so that no current can flow. This is often referred to as a Hi-Z (high-impedance) state.
- Three-state buffers are also called tri-state buffers.
- There are four types three-state buffer.

Type 1

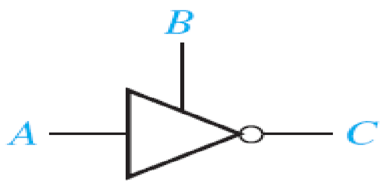
- In this type buffer input B is not inverted so buffer o/p is enabled when B=1, disabled when B =0.
- Buffer operates normally when *B* =1, and the buffer output is effectively an open circuit when *B* = 0.We use the symbol Z to represent this high-impedance state.



B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1

Type 2

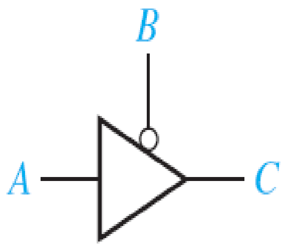
- In this type buffer input B is not inverted so buffer o/p is enabled when B=1, disabled when B =0.
- Buffer output is inverted so that $C = \overline{A}$ when buffer is enabled.



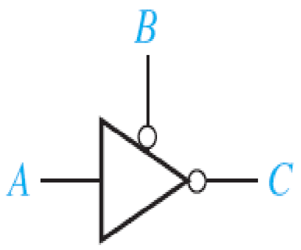
B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0

Type 3 and Type 4

- In this type buffer input B is inverted so buffer o/p is enabled when B=0, disabled when B =1.



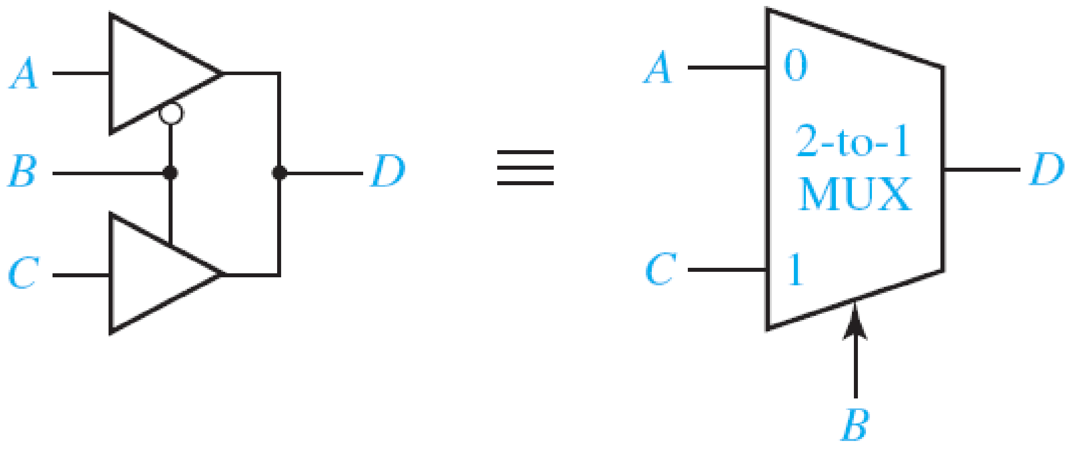
B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z



B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

Problem-8] implement 2:1 MUX using three-state buffers

Soln:



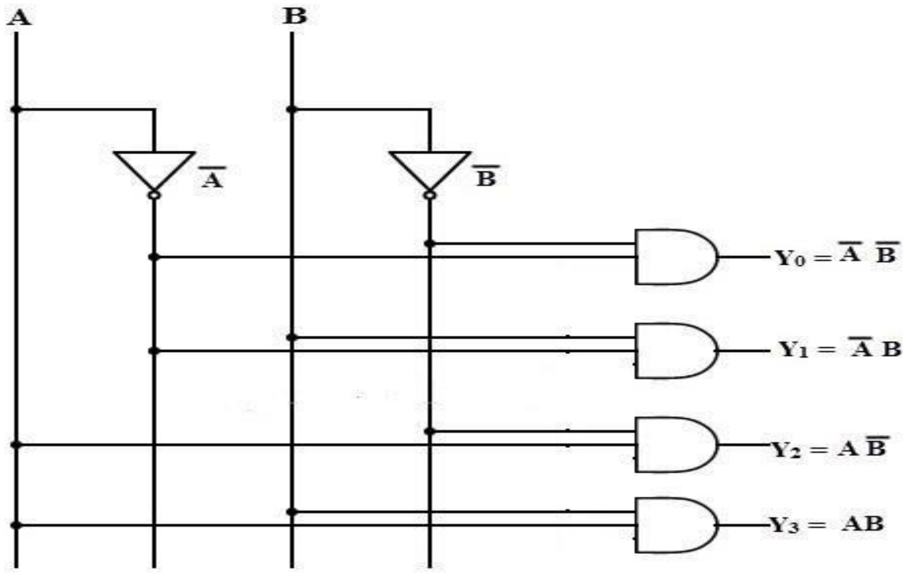
- ❖ In above figure the outputs of two three-state buffers are tied together. When *B* = 0, the top buffer is enabled, so that *D* = *A*.
- ❖ When *B* = 1, the lower buffer is enabled, so that *D* = *C*. Therefore $D = B'A + BC.$.
- ❖ This is logically equivalent to using a 2-to-1 multiplexer to select the *A* input when *B*= 0 and the *C* input when *B* = 1.

Decoders and Encoders:

Decoders

- A decoder is similar to a de-multiplexer with one exception there is no data input.
- The only inputs are the control bit.

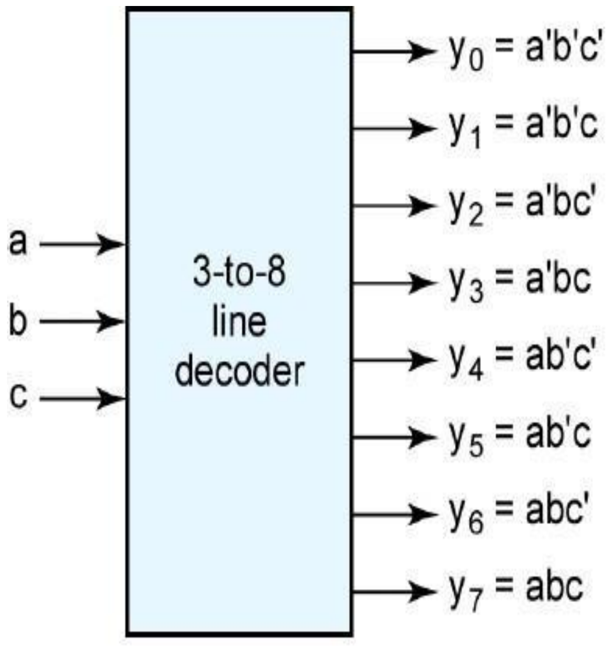
1:4 Decoder:



- This logic circuit is called as 1-of-4 decoder because only 1 of the 4 output line is high.
- For example If AB is 01, only the Y_1 AND gate has all inputs high as a result, only the Y_1 o/p goes high.
- If AB is 10, only the Y_2 AND gate has all inputs high as a result, only the Y_2 o/p goes high.
- If we check the other AB possibilities (00 to 11) we find that the subscript of the high o/p always equals the decimal equivalent of AB.
- For this reason, the circuit is sometimes called as a **binary to decimal decoder.**
- Because it has 2 input lines and 4output lines the circuit is also known as a **2 line to 4 line decoder.**

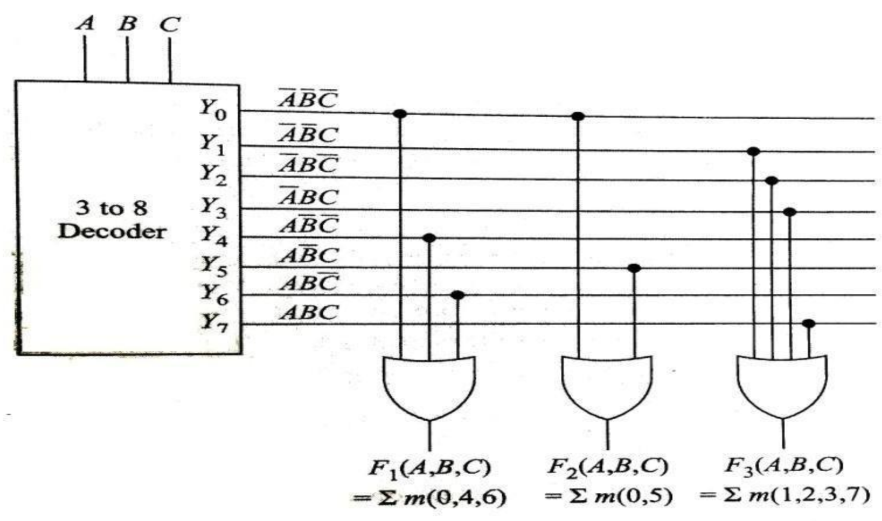
3:8 line decoder:

❖ Figure shows the diagram and truth table for a 3-to-8 line decoder.
❖ This decoder generates all of the minterms of the three input variables (a, b, c).
❖ Exactly one of the output lines will be 1 for each combination of the input values.

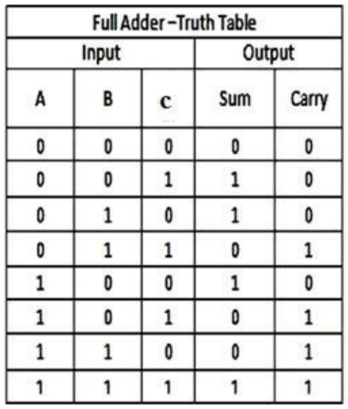


a	b	c	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Prob-9) Show how using a 3-to-8 decoder and multi input OR gates flowing Boolean function can be realized.
$F_1(A,B,C) = \sum m\ (0,4,6)$; $F_2(A,B,C) = \sum m\ (0,5)$;$F_3(A,B,C) = \sum m\ (1,2,3,7)$
Soln:

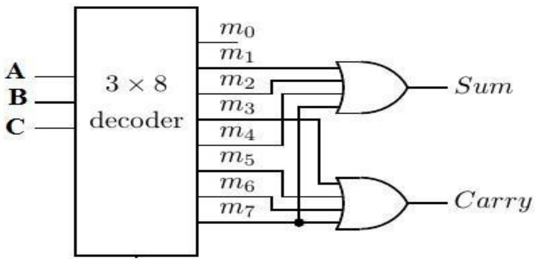

A B C
3 to 8 Decoder
Y_0 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
$\overline{A}\overline{B}\overline{C}$
$\overline{A}\overline{B}C$
$\overline{A}B\overline{C}$
$\overline{A}BC$
$A\overline{B}\overline{C}$
$A\overline{B}C$
$AB\overline{C}$
ABC
$F_1(A,B,C) = \Sigma m(0,4,6)$
$F_2(A,B,C) = \Sigma m(0,5)$
$F_3(A,B,C) = \Sigma m(1,2,3,7)$

Prob-10) Implement full adder output using a 3-to-8 decoder and multi input OR gates.
Soln:



Full Adder -Truth Table

Input			Output	
A	B	c	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

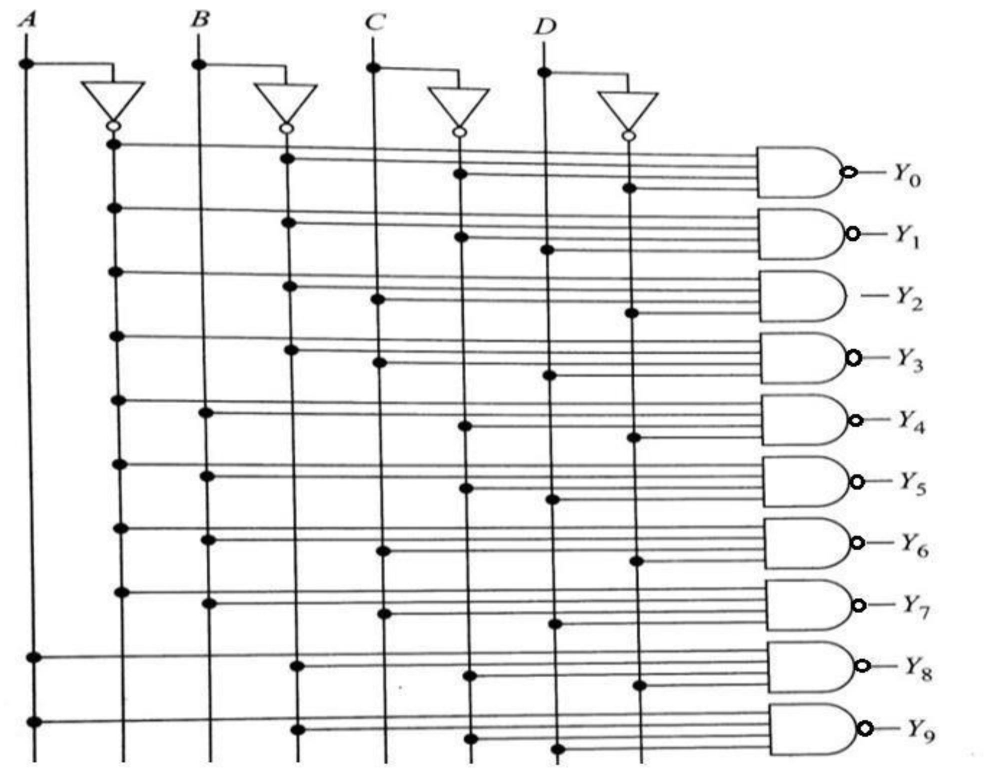


SUM = ∑m (1, 2, 4, 7)

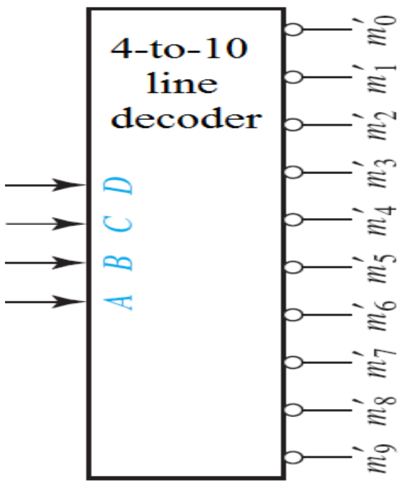
Carry =∑m (3, 5, 6, 7)

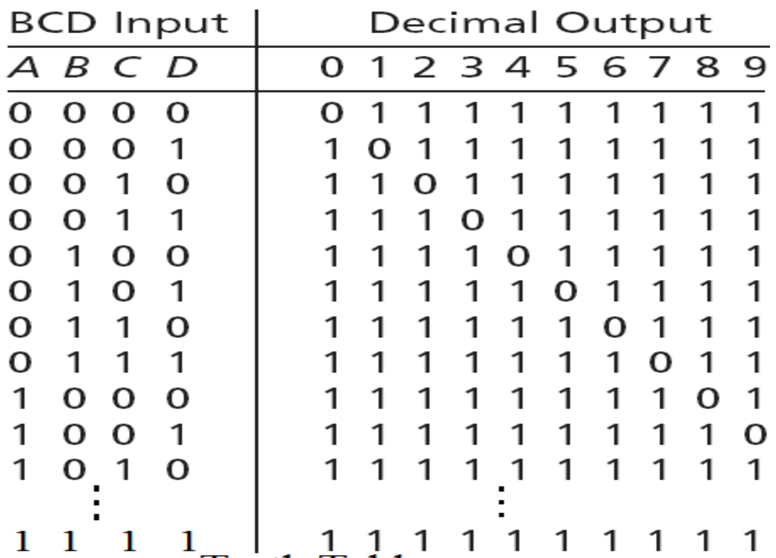
4- to-10 decoder:

- ❖ Figure illustrates a 4-to-10 decoder. This decoder has inverted outputs (indicated by the small circles).
- ❖ For each combination of the values of the inputs, exactly one of the output lines will be 0.



❖ For each combination of the values of the inputs, exactly one of the output lines will be 0.





BCD Input				Decimal Output									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
	⋮							⋮					
1	1	1	1	1	1	1	1	1	1	1	1	1	1

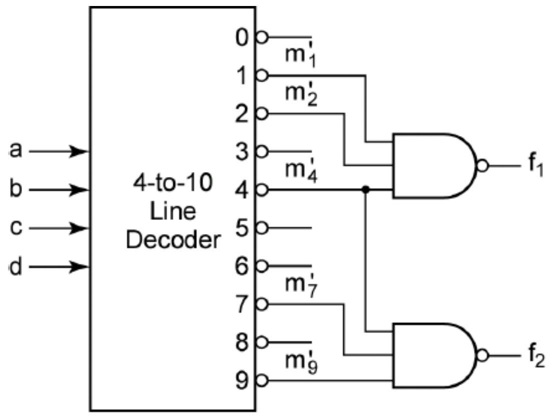
Truth Table

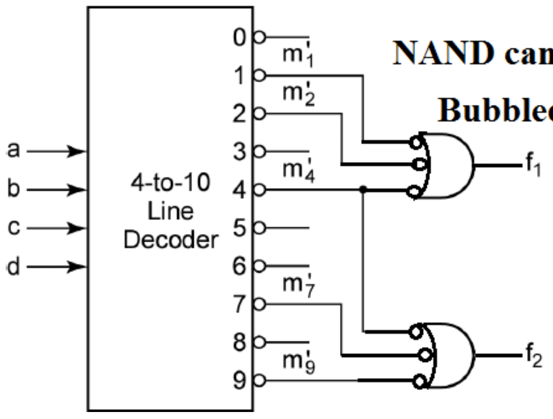
- ❖ If the decoder outputs are inverted, then NAND gates can be used to generate the functions, as illustrated in the following example.

Problem-11] Realize the following Boolean function using 4:10 line decoder

$$f_1(a,b,c,d) = m_1 + m_2 + m_4 \text{ and } f_2(a,b,c,d) = m_4 + m_7 + m_9$$

Soln:





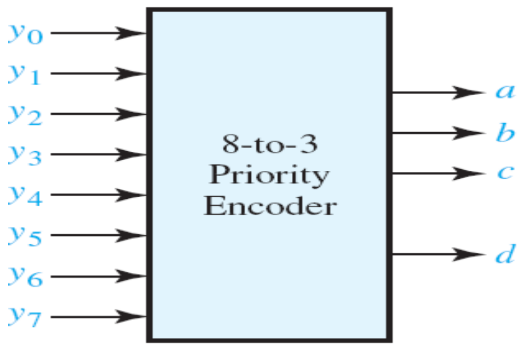
NAND can be replaced with Bubbled OR gate

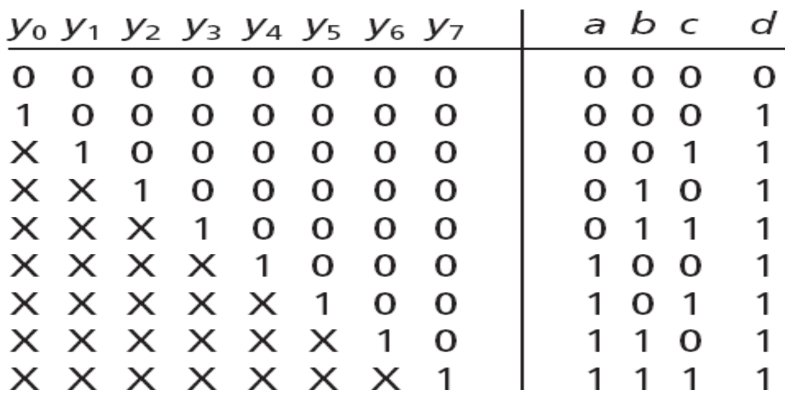
Encoder:

- ❖ Encoder converts the active input to coded binary output.
- ❖ It performs the inverse function of a decoder.

An 8-to-3 priority encoder:

- ❖ Figure shows an 8 to-3 priority encoder with inputs *Y*0 through *Y*7.
- ❖ If input *Yi* is 1 and the other inputs are 0, then the abc outputs represent a binary number equal to *i*.
- ❖ For example, if *Y*3 = 1, then *abc* = 011.
- ❖ If more than one input can be 1 at the same time, the output can be defined using a priority scheme.
- ❖ If more than one input is 1, the highest numbered input determines the output.
- ❖ For example, if inputs *y*1, *y*4, and *y*5 are 1, the output is *abc* = 101.
- ❖ The X's in the table are don't-cares.
- ❖ Output *d* is 1 if any input is 1, otherwise, *d* is 0. This signal is needed to distinguish the case of all 0 inputs from the case where only *y*0 is 1.

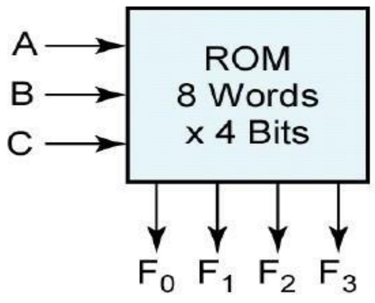




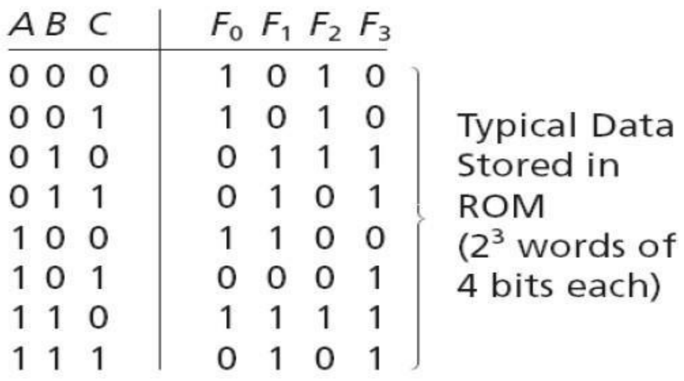
y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1

Read-Only Memories:

- A read-only memory (ROM) consists of an array of semiconductor devices that are interconnected to store an array of binary data.
- Once binary data is stored in the ROM, it can be read out whenever required.
- Figure (a) shows a ROM which has three input lines and four output lines. Figure (b) shows a typical



(a) Block diagram



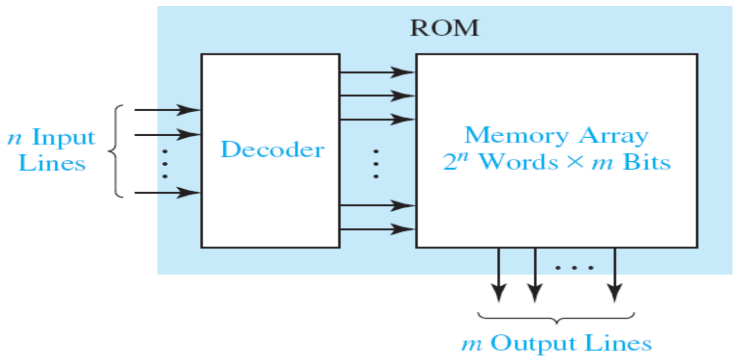
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Typical Data Stored in ROM (2^3 words of 4 bits each)

(b) Truth table for ROM

truth table which relates the ROM inputs and outputs.

- For each combination of input values on the three input lines, the corresponding pattern of 0's and 1's appears on the ROM output lines. For example, if the combination *ABC* = 010 is applied to the input lines, the pattern *F*0*F*1*F*2*F*3 = 0111 appears on the output lines.
- Each of the output patterns that is stored in the ROM is called a *word*. Because the ROM has three input lines, we have $2^3 = 8$ different combinations of input values.
- Each input combination serves as an *address* which can select one of the eight words stored in the memory.
- Bcoz there are four output lines, each word is four bits long, and the size of this ROM is 8 words X 4 bits.
- A ROM basically consists of a decoder and a memory array, as shown in below Figure.



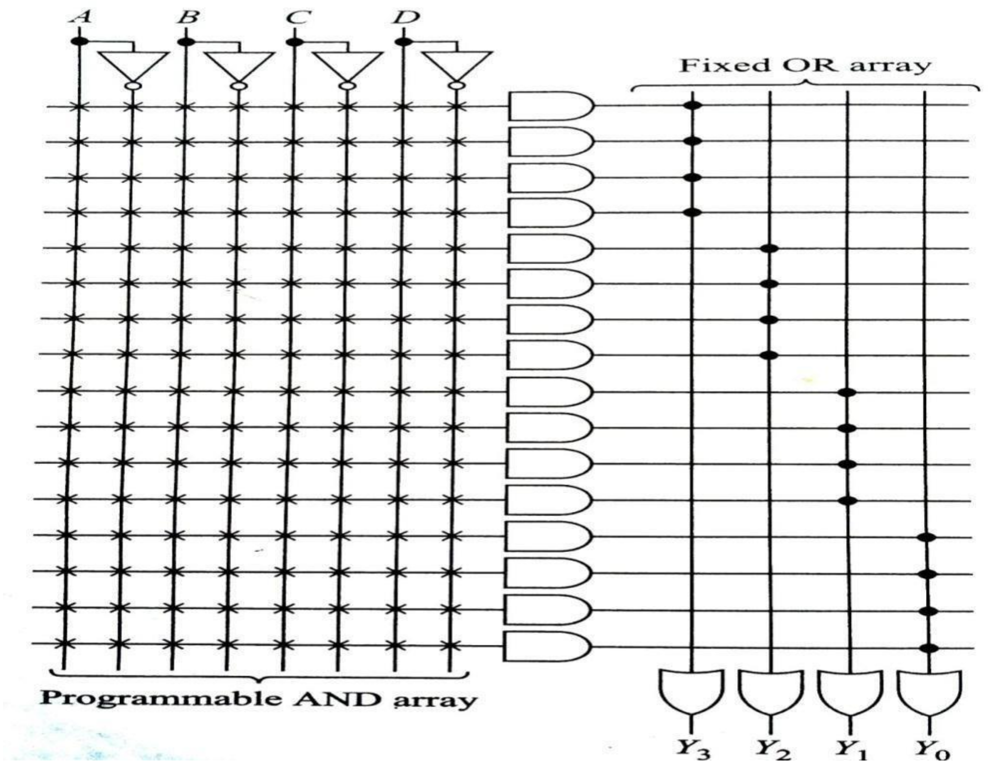
- When pattern of 0’s and 1’s is applied to the decoder inputs, exactly one of the 2^n decoder outputs is 1.
- This decoder output line selects one of the words in the memory array, and the bit pattern stored in this word is transferred to the memory output lines.

Programmable Logic Devices:

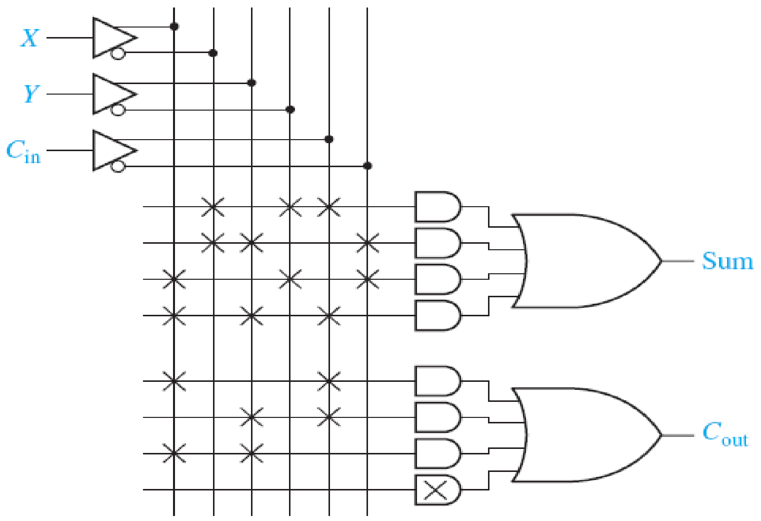
- A programmable logic device (or PLD) is a digital integrated circuit capable of being programmed to provide a variety of different logic functions.

Programmable Array Logic

- The PAL (programmable array logic) is a special case of the programmable logic array in which the AND array is programmable and the OR array is fixed.
- The basic structure of the PAL is the same as the PLA shown in Figure.
- Because only the AND array is programmable, the PAL is less expensive than the PLA, and the PAL is easier to program.
- For example below figure shows the Pal with 4 inputs and 4 outputs.
- The **X**’s on the input side are programmable links, while the solid black bullets (●) on the output side are fixed connection.
- A programmer can burn desired fundamental products, which are then ORed by the fixed output connections.



Problem-12] Implement the Full adder using PAL



Problem-13] Generate the following Boolean function using PAL

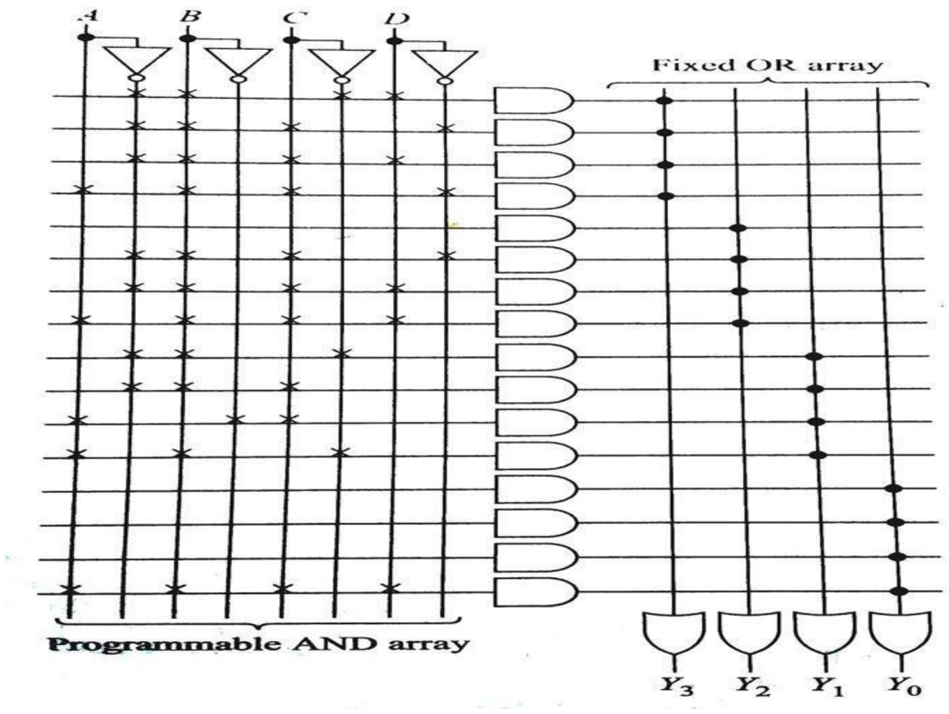
$$Y_3 = \overline{A}B\overline{C}D + \overline{A}BC\overline{D} + \overline{A}BCD + ABC\overline{D}$$

$$Y_2 = \overline{A}BC\overline{D} + \overline{A}BCD + ABCD$$

$$Y_1 = \overline{A}B\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

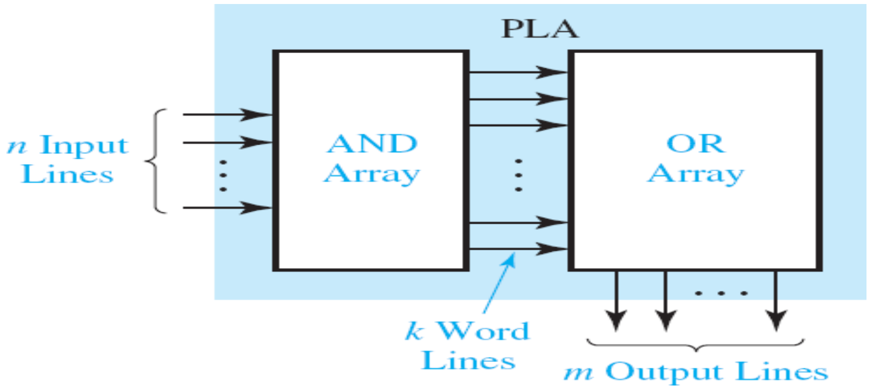
$$Y_0 = ABCD$$

Soln:

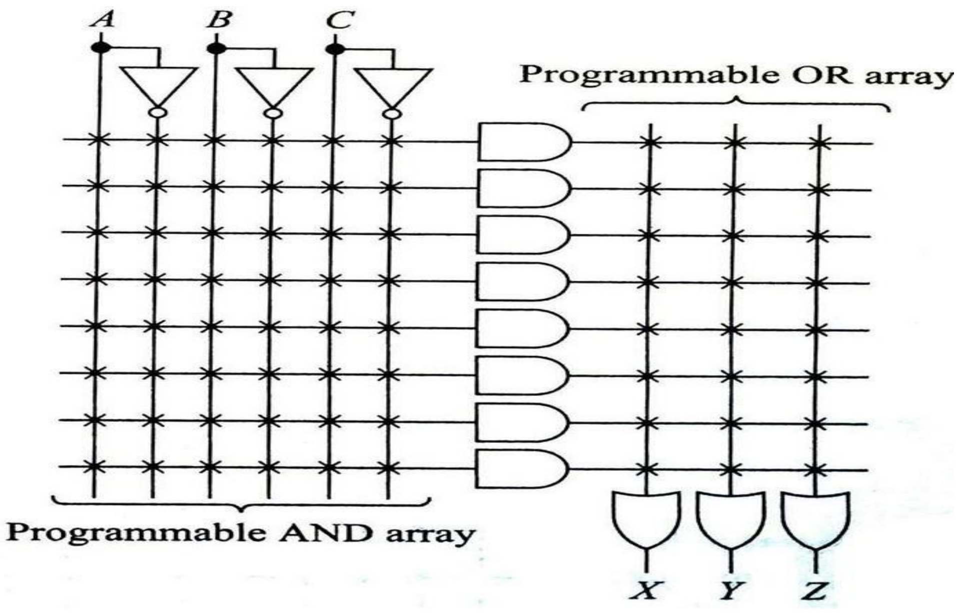


Programmable Logic Arrays

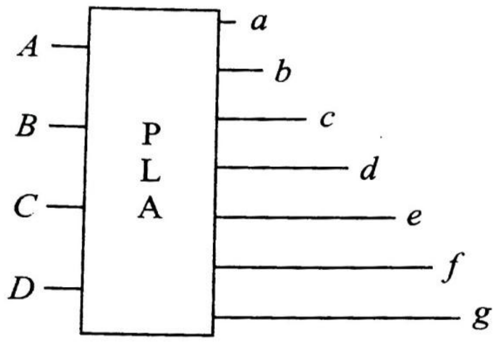
- ❖ A programmable logic array (PLA) performs the same basic function as a ROM.
- ❖ A PLA with *n* inputs and *m* outputs can realize *m* functions of *n* variables.
- ❖ The internal organization of the PLA is different from that of the ROM. The decoder is replaced with an AND array which realizes selected product terms of the input variables.
- ❖ The OR array ORs together the product terms needed to form the output functions, so a PLA implements a sum-of-products expression.
- ❖ A PLA has programmable AND array and programmable OR array.
- ❖ A PLA having 3 input variable (ABC) and 3 output variable (XYZ) is shown below.

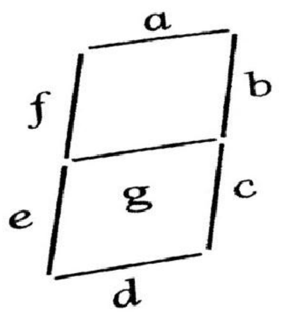


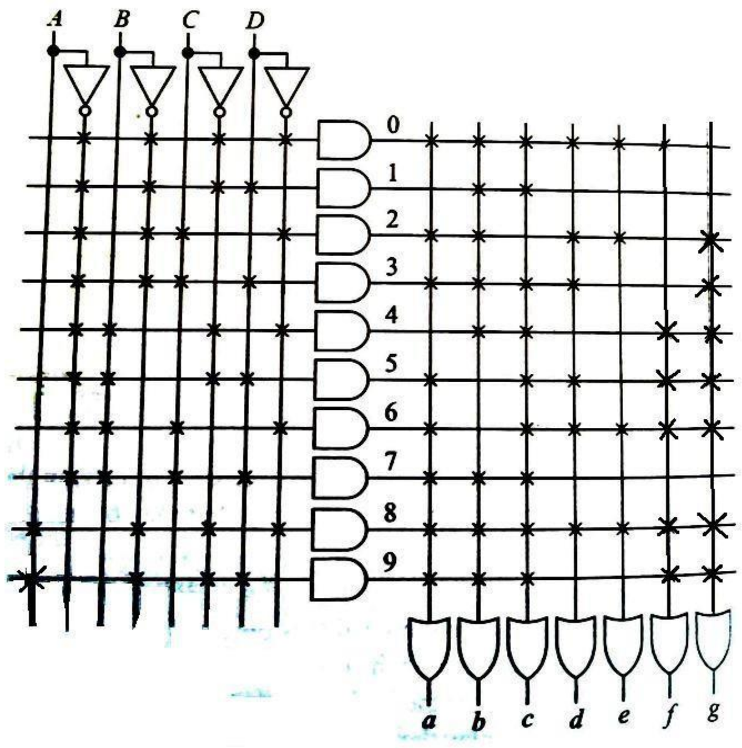
- ❖ Eight AND gates are required to decode the 8 possible input states. In this case there are three OR gates that can be used to generate logic function at the output.
- ❖ Note: there could be additional OR gates at the output if desired.



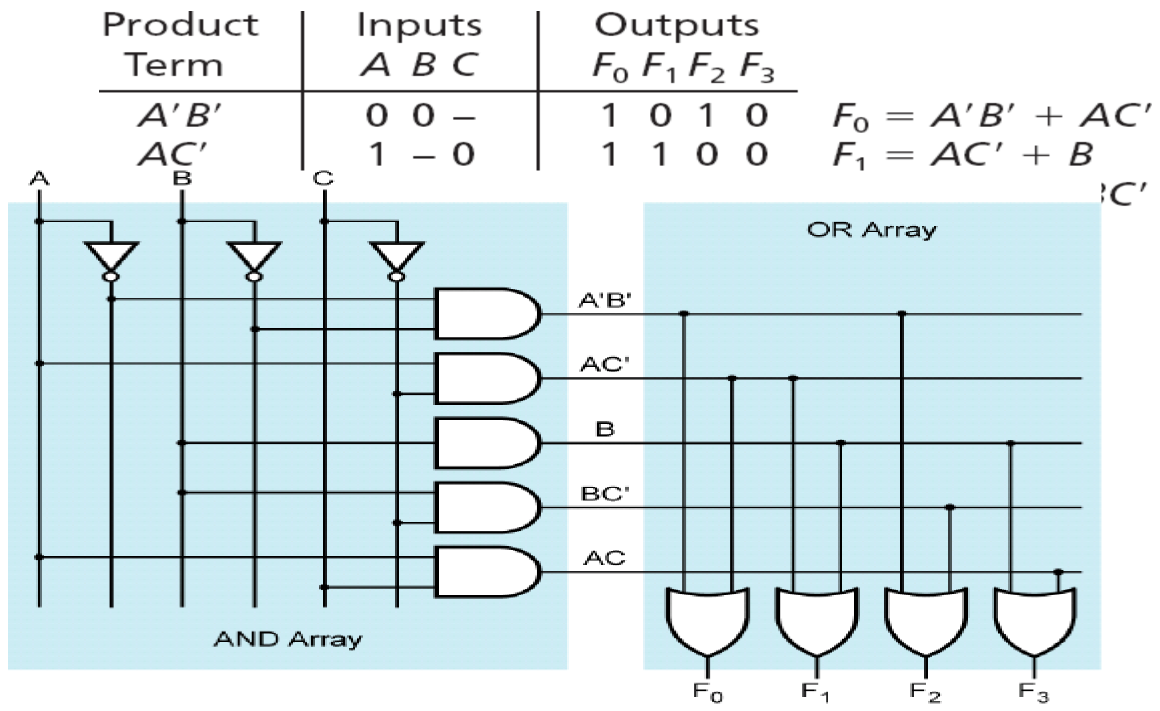
Prob-14) Design a PLA to recognize each of the 10 decimal digits represented in binary form and to drive a seven segment display?







Problem-15] Implement the following table using PLA



Product Term	Inputs $A\ B\ C$	Outputs $F_0\ F_1\ F_2\ F_3$
$A'B'$	0 0 –	1 0 1 0
AC'	1 – 0	1 1 0 0

$F_0 = A'B' + AC'$

$F_1 = AC' + B$

Assignments:

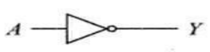
1. Implement the following Boolean function using PLA [$F_1(A,B,C,D) = \sum m\ (0,1,4,6)$; $F_2(A,B,C,D) = \sum m\ (2,3,4,6,7)$; $F_3(A,B,C,D) = \sum m\ (0,1,2,6)$; $F_4(A,B,C,D) = \sum m\ (2,3,5,6,7)$
2. Design a seven segment decoder using PLA
3. Draw the PLA circuit and realize the following Boolean function:
   X = A`B`C + AB`C` + B`C, Y = A`B`C + AB`C`, Z = B`C
4. Implement the following Boolean function using a 8:1 Multiplexer
   $F(A,B,C,D) = \sum m\ (0,1,5,6,8,10,12,15)$
5. Design a 16-to-1 multiplexer using two 8-to-1 multiplexer and one 2-to-1 multiplexer.
6. What is multiplexer? Explain with neat diagram 2:1 MUX
7. What is decoder? Explain 3:8 decoder with truth table?
8. What is encoder? Explain 8:3 encoder?
9. Explain with neat diagram PAL and PLA? Mention Two difference between them?
10. Explain with neat diagram ROM?
11. What is three state buffer ? Explain different types three state buffer
12. Realize the following function using 3: 8 decoder?
    $F_1(A,B,C) = \sum m\ (1,2,3,4)$; $F_2(A,B,C) = \sum m\ (3,5,7)$

Overview of Basic Gates

- A logic gate is an electronic circuit/device which makes the logical decisions.
- Logic gates have one or more than one inputs and only one output.

The Inverter (NOT Gate):-

- The NOT gate performs the logical function called inversion or complementation.
- NOT gate is also called inverter. It has one input and one output.
- When a HIGH level is applied to an inverter, a LOW level appears on its output and vice versa.
- Inverter symbol, truth table and pinout diagram of a 7404and timing diagram is given below.



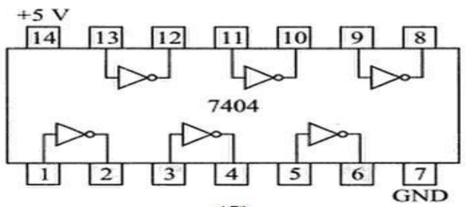
(a)

A	Y
L	H
H	L

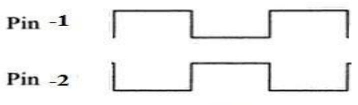
OR

A	Y
0	1
1	0

(b)



(C)



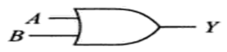
(D)

- **In equation form Y= not A i.e. Y=A` . so that, if A=0, Y=0`=1 and if A=1, Y=1`=0.**

Note: **Timing diagram** shows the input and output waveforms of a logic circuits.
Truth tableshows all of the input-output possibilities of a logic circuits.

OR GATE:

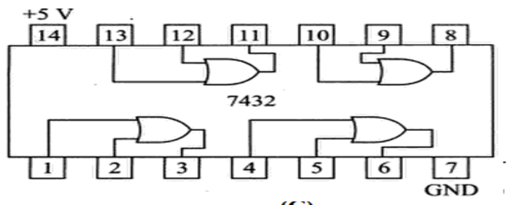
- The OR gate performs logical addition
- An OR gate has two or more input signals but only one output signal.
- OR gate symbol, truth table and pinout diagram of a 7432 and timing diagram is given below.
- The output of a 2-input OR gate is high if either or both inputs are high.



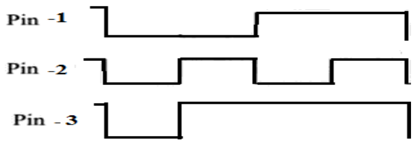
(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)



(C)



(D)

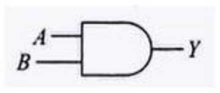
- In Boolean equation form, **Y=A OR B,** i.e **Y=A+B**
- So that Y=0+0=0, Y=0+1=1, Y=1+0=1 and Y=1+1=1.
- The '+' sign here represents logical OR operation and not addition operation of basic arithmetic.

AND GATE

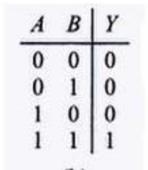
- The AND gate performs logical multiplication
- The AND gate has a high output only when all inputs are high.
- AND gate symbol, truth table and pinout diagram of a 7408 and timing diagram is given below.
- In Boolean equation form **Y=A AND B; i.e. Y=A.B or Y= AB**
- If **Y=0.0=0, Y=0.1=0, Y=1.0=0, Y=1.1=1**
- The ' . ' sign here represents logical AND operation and not multiplication operation of basic arithmetic

.

1) **What is the Boolean equation for the logic circuit given below?**

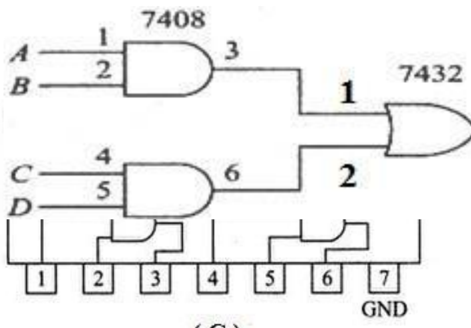


(a)

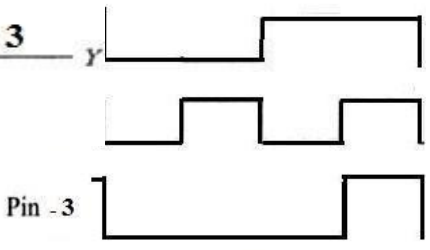


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(b)



(C)

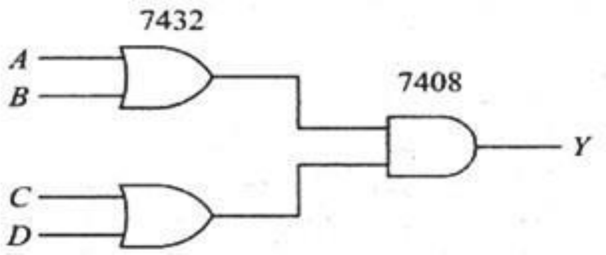


(D)

Solution:

- ✓ The outputs is Y=AB+CD. This form is referred as **Sum-of-products equation.**
- ✓ **<u>AND-OR networks always produce sum-of-products equation.</u>**

2) **Write the Boolean equation for the given network**



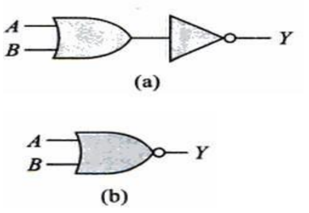
Solution:

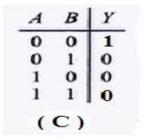
- ✓ The output is Y=(A+B)(C+D). This equation is called product of sums.
- ✓ **<u>OR-AND networks always produce product-of-sums equations</u>.**

Universal logic gates:

<u>NOR:</u>

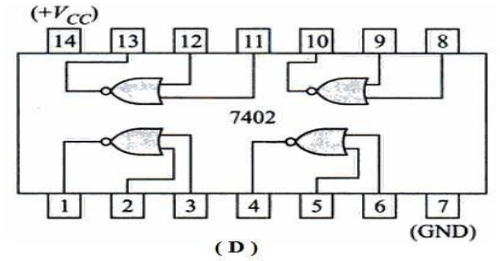
1. NOR gate is a cascade of OR gate and NOT gate.
2. It has two or more inputs and only one output.
3. The output of NOR gate is HIGH when any all its inputs are LOW
4. NOR symbol, truth table and pinout diagram of a 7402 and timing diagram is given below



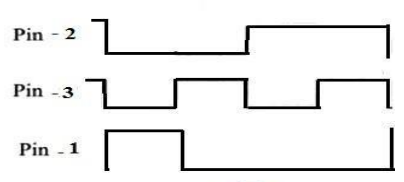


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(C)



(D)

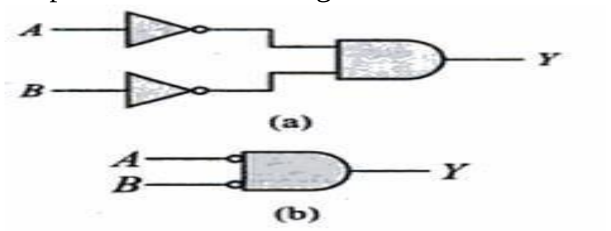


(E)

- In equation form **Y = A + B**

Bubbled AND gate

- Given any logic circuits with NOR gate we can replace it by bubbled AND gate
- An inverter is connected at the input side of the AND gate.



De Morgan's first theorem

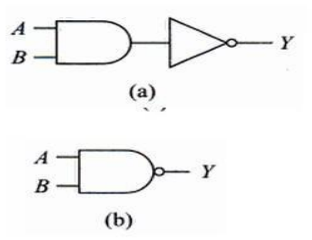
- The Boolean equation for NOR gate is Y = $\overline{A + B}$
- The Boolean equation for Bubbled AND gate is Y = $\overline{A}\,.\overline{B}$
- The outputs are equal for the same inputs, we can write

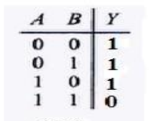
$$\overline{\mathbf{A + B}} = \overline{\mathbf{A}}\,.\,\overline{\mathbf{B}}$$

- This identity known as De Morgan's first theorem
- The complement of a sum equals the product of the complements.

<u>NAND Gate</u>

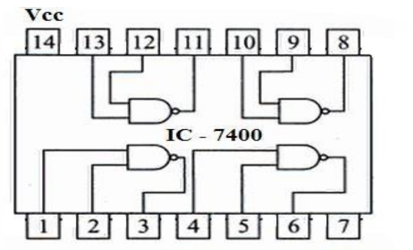
- NAND gate is a cascade of AND gate and NOT gate.
- It has two or more inputs and only one output.
- The output of NAND gate is HIGH if either or both inputs are LOW
- NAND symbol, truth <u>table</u> and pinout diagram of a 7400 and timing diagram is given below
- In equation form **Y = AB**



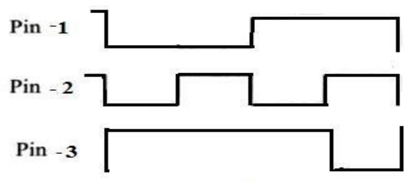


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(C)



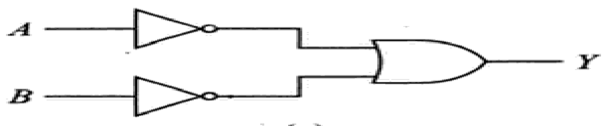
(D)



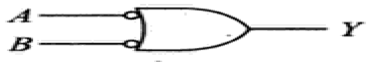
(E)

Bubbled OR gate

- Given any logic circuits with NAND gate we can replace it by bubbled OR gate
- An inverter is connected at the input side of the OR gate.



(a)



(b)

De Morgan's second theorem

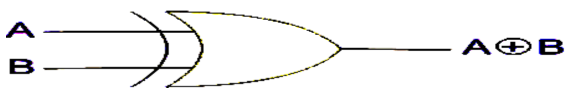
- The Boolean equation for NAND gate is Y = $\overline{AB}$
- The Boolean equation for Bubbled OR gate is Y = $\overline{A} + \overline{B}$
- The outputs are equal for the same inputs, we can write

$$\overline{AB} = \overline{A} + \overline{B}$$

- This identity known as De Morgan's second theorem
- The complement of a product equals the sum of the complements

XOR Gate

- An Exclusive-OR (XOR) gate is gate with two or three or more inputs and one output.
- The output of a two-input XOR is HIGH if either input A or input B is HIGH exclusively, and LOW when both are 1 or 0 simultaneously.
- If A and B are two inputs, then output Y can be represented mathematically as Y = A $\oplus$ B, Here $\oplus$ denotes the XOR operation.



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

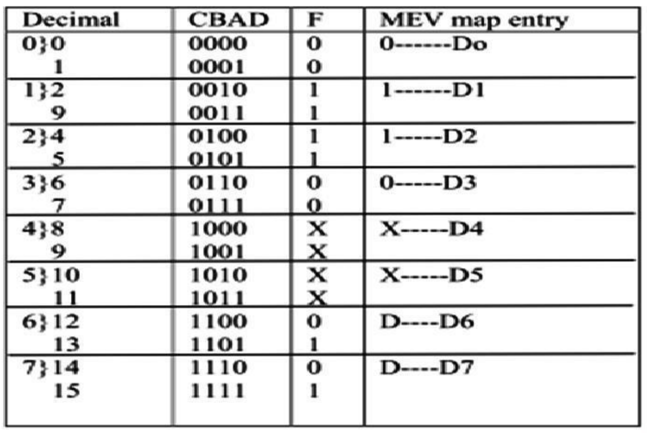
Experiment:5

8:1 MUX

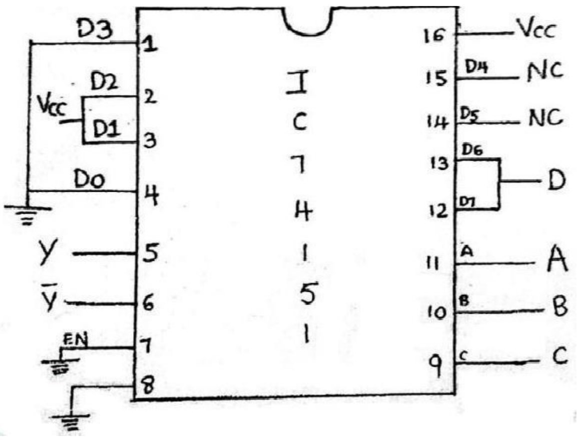
Given a 4-variable logic expression, simplify it using appropriate technique and realize the simplified logic expression using 8:1 multiplexer IC. And implement the same in HDL.

Given: Simplify the function using MEV technique:- F(A,B,C,D)=∑m(2,3,4,5,13,15)+dc(8,9,10,11)

Truth table and Circuit Diagram:



Decimal	CBAD	F	MEV map entry
0}0	0000	0	0------Do
1	0001	0	
1}2	0010	1	1------D1
9	0011	1	
2}4	0100	1	1-----D2
5	0101	1	
3}6	0110	0	0-----D3
7	0111	0	
4}8	1000	X	X-----D4
9	1001	X	
5}10	1010	X	X-----D5
11	1011	X	
6}12	1100	0	D----D6
13	1101	1	
7}14	1110	0	D----D7
15	1111	1	



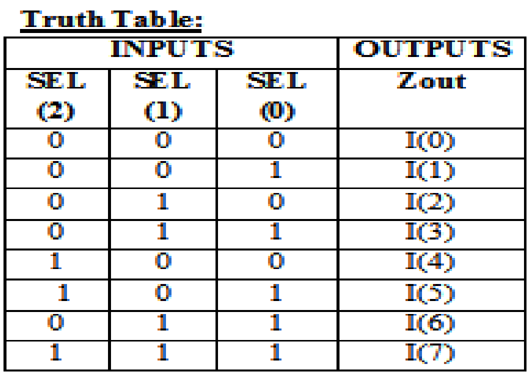
Theory:
Multiplexer has many inputs but only one output, by applying control signal we can steer any input to the output.

Procedure:

1) Verify all components and patch chords whether they are in good condition ornot.
2) Make connection as shown in the circuitdiagram.
3) Give supply to the trainerkit.
4) Provide input data to circuit viaswitches.
5) Verify truth table sequence and observeoutputs.

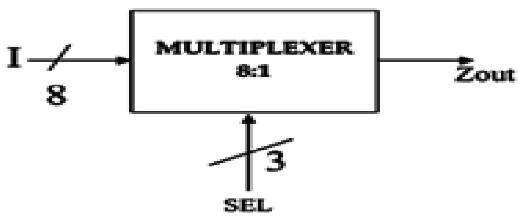
Result: Verified truth table sequence and observed the outputs.

Simulation of 8:1 MUX using VHDL:



Truth Table:

INPUTS			OUTPUTS
SEL (2)	SEL (1)	SEL (0)	Zout
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)
0	1	1	I(6)
1	1	1	I(7)



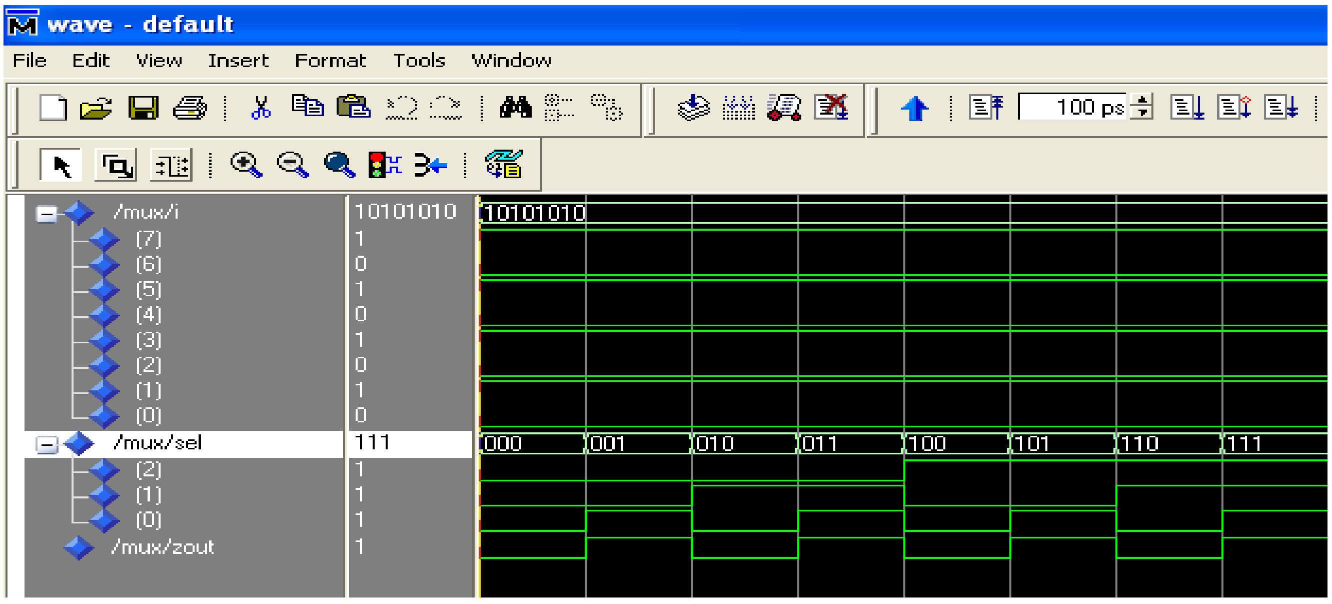
Digital multiplexer

Define variables as:- I-> in-> select bus-> 07-00, sel->in->select bus->02-00, zout->out

VHDL code for 8 to 1 mux (Behavioral modeling):

```
 library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 entity mux1 is
    Port ( I : in std_logic_vector(7 downto
 0); sel : in std_logic_vector(2 downto 0);
 zout : out
 std_logic); end
 mux1;
 architecture behavioral of
 mux1 is begin
 zout<=I(0) when sel="000"else
I(1) when sel="001"else
I(2) when sel="010" else
I(3) when sel="011" else
I(4) when sel="100" else
I(5) when sel="101" else
I(6) when sel="110" else
I(7);
 end behavioral;
```

Waveform:



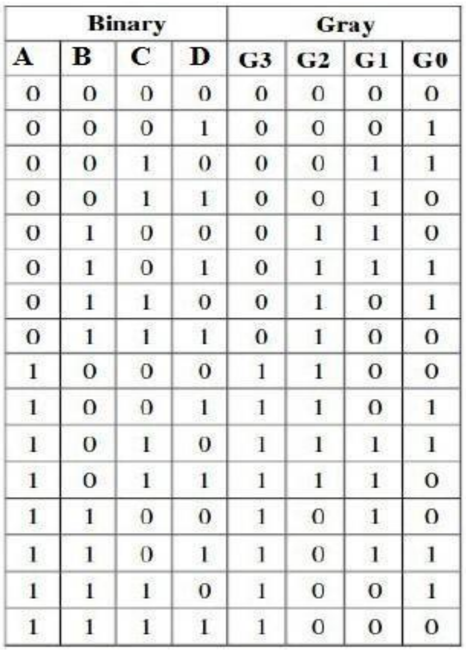
Experiment: 6-

Code Converter

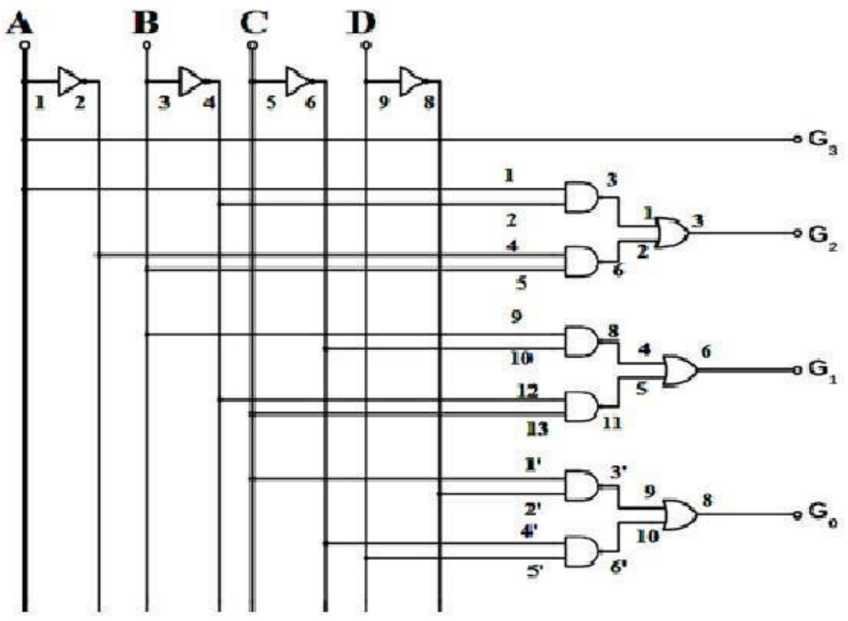
Design and implement code converter I)Binary to Gray (II) Gray to Binary Code using basic gates.

Components Required :IC 74LS04, 74LS08, 74LS32, 74LS86, Patch chords, and Trainer kit.

Circuit diagram and truth table:



Binary				Gray			
A	B	C	D	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0



Theory:
Binary Codes:

A symbolic representation of data/ information is called code. The base the binary number is 2. Hence,it has two independent symbols. The symbols used are 0 and 1. A binary number consists of sequence of bits, each of which is either a 0 or1.

[i] Binary toGray

Follow the below steps to convert Binary to Gray Code

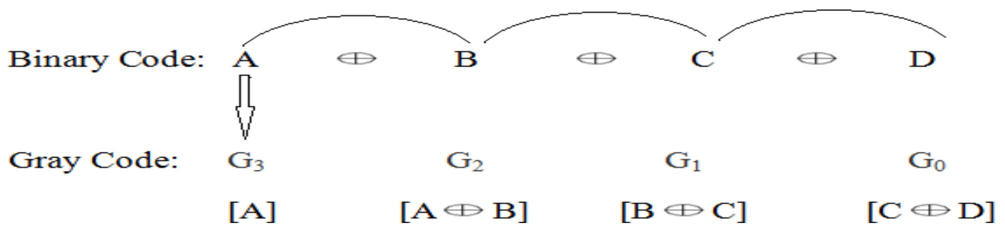
1. Most significant bit (A) is same as the most significant bit in Gray Code G_3 i.e. A =G_3
2. To find next bit perform Ex-OR between the Current binary bit and previousbit.
3. Look the below Image for Binary to Gray codeConversion
4. **By solvingthe truth table using K- map we get the Boolean expression for Binary to Gray**

$G_3 = A$

$G_2 = A \oplus B = \bar{A}B + A\bar{B}$

$G_1 = B \oplus C = \bar{B}C + B\bar{C}$

$G_0 = C \oplus D = \bar{C}D + C\bar{D}$


Binary Code: A ⊕ B ⊕ C ⊕ D
Gray Code: G_3 G_2 G_1 G_0
[A] [A ⊕ B] [B ⊕ C] [C ⊕ D]

Gray Codes:

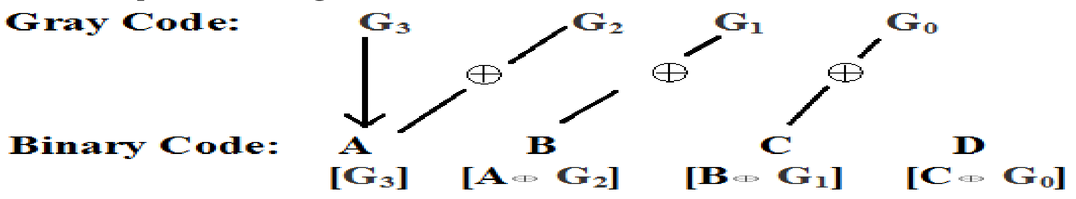
It is a cyclic code. In gray code the bit patterns for two consecutive numbers will differ in one bit position. It is not a suitable for arithmetic operations.

[ii] GraytoBinary

Follow the below steps to convert Gray Code to Binary

1. Most significant bit (G_3) is same as the most significant bit in Binary Code (G_3 =A)
2. The next number can be obtained by taking Ex-OR operation between the previous binary bit, and

   the current gray code bit and write down thevalue.
3. Repeat the Above Step forremaining.


Gray Code: G_3 G_2 G_1 G_0
Binary Code: A B C D
[G_3] [A ⊕ G_2] [B ⊕ G_1] [C ⊕ G_0]

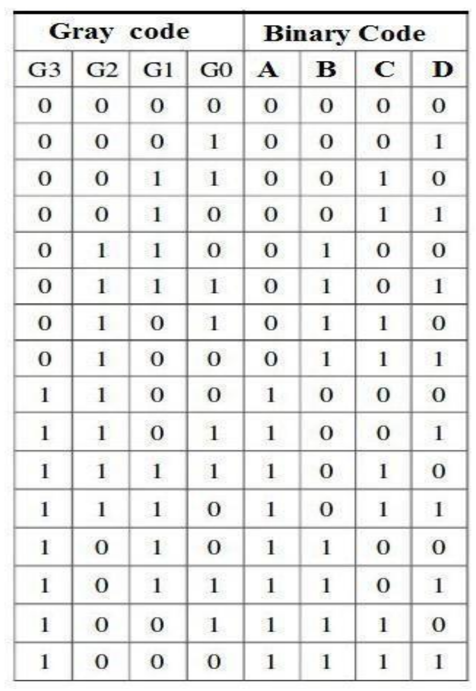
By solving the truth table using K- map we get the Boolean expression for Gray to Binary

A= G_3

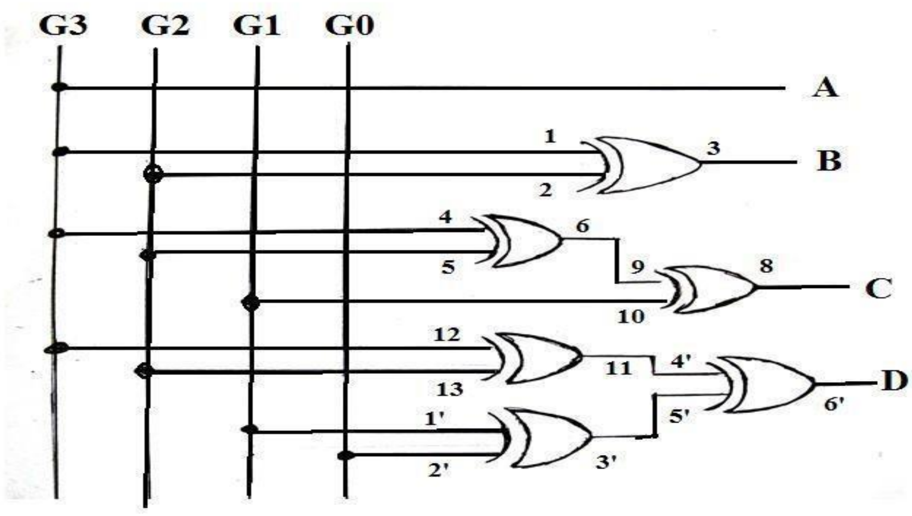
B=$G_3 \oplus G_2$

C=$G_3 \oplus G_2 \oplus G_1$

D=$G_3 \oplus G_2 \oplus G_1 \oplus G_0$



Gray code				Binary Code			
G3	G2	G1	G0	A	B	C	D
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1



PROCEDURE:

- Check all the components for theirworking.
- Insert the appropriate IC into the trainerkit.
- Make connections as shown in the circuitdiagram.
- Verify the Truth Table and observe theoutputs

RESULTS:

Binary to gray code conversion and vice versa is realized using basic gates.

MODULE 4 INTRODUCTION OF VHDL

The acronym VHDL stands for VHSIC-HDL (Very High Speed Integrated Circuit-Hardware Description Language). *VHDL* is a hardware description language that is used to describe the behavior and structure of digital systems. *VHDL* is a general-purpose hardware description language which can be used to describe and simulate the operation of a wide variety of digital systems, ranging in complexity from a fewgates to an interconnection of many complex integrated circuits.

VHDL was originally developed to allow a uniform method for specifying digital systems. The VHDL language became an IEEE standard in 1987, and it is widely used in industry. IEEE published a revised VHDL standard in 1993.

VHDL can describe a digital system at several different levels—behavioral, data flow, and structural. For example,

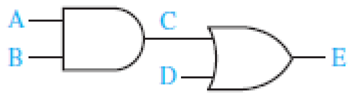
1. A binary adder could be described at the *behavioral* level in terms of its function of adding twobinary numbers, without giving any implementation details.
2. The same adder could be described at the *data flow* level by giving the logic equations for theadder.
3. Finally, the adder could be described at the *structural* level by specifying the interconnections ofthe gates which make up the adder.

VHDL DESCRIPTION OF COMBINATIONAL CIRCUITS:

In VHDL, a signal assignment statement has the form:

```
signal_name <= expression [after delay];
```

The expression is evaluated when the statement is executed, and the signal on the left side is scheduled tochange after delay. The square brackets indicate that after delay is optional. If after delay is omitted, then the signal is scheduled to be updated after a *delta delay*, Δ (infinitesimal delay). A VHDL *signal* is usedto describe a signal in a physical system. The VHDL language also includes *variables* similar to variablesin programming languages. In general, VHDL is *not case sensitive*, that is, capital and lower case letters are treated the same by the compiler and the simulator. Signal names and other VHDL identifiers may contain letters, numbers, and the underscore character (_). An identifier must start with a letter, and it cannot end with an underscore. Thus, C123 and ab_23 are legal identifiers, but 1ABC and ABC_ are not. Every VHDL statement must beterminated with a semicolon. Spaces, tabs, and carriage returns are treated in the same way. This means that a VHDL statement can be continued over several lines, or several statements can be placed on one line. In a line of VHDL code, anything following a double dash (--) is treated as a comment. Words such as *and*, *or*, and *after* are reserved words (or keywords) which have a



special meaning to the VHDL compiler.

The gate circuit of the following Figure has five signals: A, B, C, D, and E. The symbol " <= " is the *signal assignment operator* which indicates that the value computed on the right-hand side is assigned to the signal on the left side.

<u>Dataflow Description:</u> The two assignment statements (given below) give a dataflow description of the above circuit, where it is assumed that each gate has a 5-ns propagation delay. When these statements are simulated, the first statement will be evaluated any time A or B changes, and the second statement will be evaluated any time C or D changes. Suppose that initially A = 1, and B = C = D = E = 0; and if B changes to 1 at time 0, C will change to 1 at time = 5 ns. Then, E will change to 1 at time = 10 ns.

*C <= A **and** B **after** 5 ns;*

*E <= C **or** D **after** 5 ns;*

VHDL signal assignment statements (as given above) are *concurrent statements*. The VHDL simulator monitors the right side of each concurrent statement, and any time a signal changes, the expression on the right side is immediately re-evaluated. The new value is assigned to the signal on the left side after an appropriate delay. This is exactly the way the hardware works. Any time a gate input changes, the gate output is recomputed by the hardware, and the output changes after the gate delay. Unlike a sequential program, the order of the above concurrent statements is unimportant.

<u>Behavioral Description:</u> A behavioral description of the above circuit shown is

*E <= D **or** (A **and** B);*

Parentheses are used to specify the order of operator execution.

<u>Structural Description:</u> The above circuit shown can also be described using structural VHDL code. To do so requires that a two-input AND-gate component and a two-input OR-gate component be declared and defined.

Components may be declared and defined either in a library or within the architecture part of the VHDL code. Instantiation statements are used to specify how components are connected. Each copy of a component requires a separate instantiation statement to specify how it is connected to other components

and to the port inputs and outputs. An instantiation statement is a concurrent statement that executes

anytime one of the input signals in its port map changes. The circuit shown is described by instantiating the AND gate and the OR gate as follows:

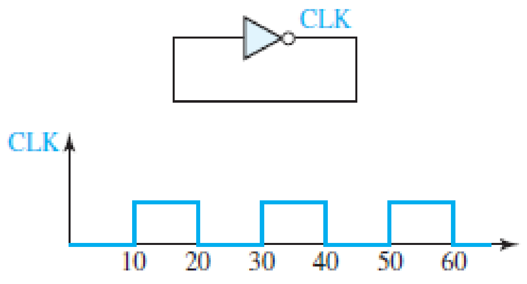
Gate1: AND2 ***port map*** *(A, B, D); Gate2:*

OR2 ***port map*** *(C, D, E);*

The port map for Gate1 connects A and B to the AND-gate inputs, and it connects D to the AND-gate output. Since an *instantiation statement* is concurrent, whenever A or B changes, these changes go to the Gate1 inputs, and then the component computes a new value of D. Similarly, the second statement passes changes in C or D to the Gate 2 inputs, and then the component computes a new value of E. This is exactly how the real hardware works. (The order in which the instantiation statements appear is irrelevant).

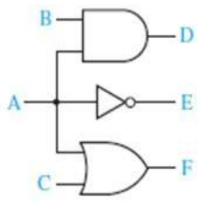
Instantiating a component is different than calling a function in a computer program. A function returns a new value whenever it is called, but an instantiated component computes a new output value whenever its input changes.

The following Figure shows an inverter with the output connected back to the input. If the output is „0‟, then this „0‟ feeds back to the input and the inverter output changes to „1‟ after the inverter delay, assumed to be 10 ns. Then, the „1‟ feeds back to the input, and the output changes to „0‟ after the inverterdelay. The signal CLK will continue to oscillate between „0‟ and „1‟, as shown in the waveform. The corresponding concurrent VHDL statement will produce the same result. If CLK is initialized to „0‟, the statement executes and CLK changes to „1‟ after 10 ns. Because CLK has changed, the statement executes again, and CLK will change back to „0‟ after another 10 ns. This process will continue indefinetly.



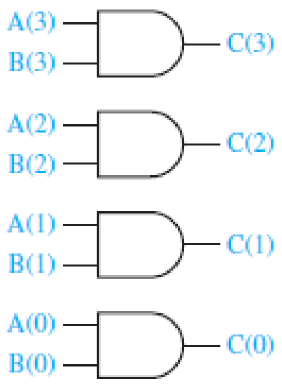
```
CLK <= not CLK after 10 ns;
```

The following Figure shows three gates that have the signal A as a common input and the corresponding VHDL code. The three concurrent statements execute simultaneously whenever A changes, just as the three gates start processing the signal change at the same time. However, if the gates have different delays, the gate outputs can change at different times. If the gates have delays of 2 ns, 1 ns, and 3 ns, respectively, and A changes at time 5 ns, then the gate outputs D, E, and F can change at times 7 ns, 6 ns, and 8 ns, respectively. However, if no delays were specified, then D, E, and F would all be updated at time 5 + Δ.



```
-- when A changes, these concurrent
-- statements all execute at the same time
D <= A and B after 2 ns;
E <= not A after 1 ns;
F <= A or C after 3 ns;
```

In these examples, every signal is of type bit, which means it can have a value of „0‟ or „1‟. (Bit values in VHDL are enclosed in single quotes to distinguish them from integer values). In digital design, we often need to perform the same operation on a group of signals. A one-dimensional array of bit signals is referred to as a bit-vector. If a 4-bit vector named B has an index range 0 through 3, then the four elements of the bit-vector are designated B(0), B(1), B(2), and B(3). The statement B <= “0110”, assigns „0‟ to B(0), „1‟ to B(1), „1‟ to B(2), and „0‟ to B(3). The following Figure shows an array of four AND gates. The inputs are represented by bit-vectors A and B, and the outputs by bit-vector C. Although we can write four VHDL statements to represent the four gates, it is much more efficient to write a single VHDL statement that performs the and operation on the bit-vectors A and B. When applied to bit-vectors, the and operator performs the and operation on corresponding pairs of elements.



```
-- the hard way
C(3) <= A(3) and B(3);
C(2) <= A(2) and B(2);
C(1) <= A(1) and B(1);
C(0) <= A(0) and B(0);
```

```
-- the easy way
C <= A and B;
```

Inertial delay model: Signal assignment statements containing “after delay” create what is called an inertial delay model. Consider a device with an inertial delay of D time units. If an input change to the device will cause its output to change, then the output changes D time units later. However, this is not what happens if the device receives two input changes within a period of D time units and both input changes should cause the output to change. In this case the device output does not change in response to either input change.

Example: consider the signal assignment *C <= A **and** B **after** 10*

Assume A and B are initially 1, and A changes to 0 at 15 ns, to 1 at 30 ns, and to 0 at 35 ns. Then C changes to 1 at 10 ns and to 0 at 25 ns, but C does not change in response to the A changes at 30 ns and 35 ns; because these two changes occurred less than 10 ns apart.

A device with an inertial delay of D time units filters out output changes that would occur in less than or equal to D time units.

Ideal (Transport) delay: VHDL can also model devices with an ideal (transport) delay. Output changes caused by input changes to a device exhibiting an ideal (transport) delay of D time units are delayed by D time units, and the output changes occur even if they occur within D time units. The VHDL signal assignment statement that models ideal (transport) delay is

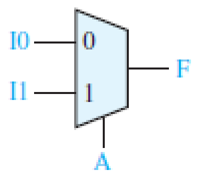
signal_name <= transport expression after delay

Example: consider the signal assignment *C <= **transport** A **and** B **after** 10 ns;*

Assume A and B are initially 1 and A changes to 0 at 15 ns, to 1 at 30 ns, and to 0 at 35 ns. Then C changes to 1 at 10 ns, to 0 at 25 ns, to 1 at 40 ns, and to 0 at 45 ns. Note that the last two changes are separated by just 5 ns.

VHDL MODELS FOR MULTIPLEXERS:

The following Figure shows a 2-to-1 multiplexer (MUX) with two data inputs and one control input.



The MUX output is $F = A'I_0 + AI_1$. The corresponding VHDL statement is

F <= (**not** A **and** I0) **or** (A **and** I1);

Alternatively, we can represent the MUX by a conditional signal assignment statement,

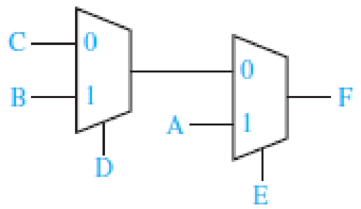
F <= I0 **when** A = „0" **else** I1;

The general form of a conditional signal assignment statement is

```
signal_name <= expression1 when condition1
               else expression2 when condition2
               [else expressionN];
```

The following Figure shows how two cascaded MUXes can be represented by a conditional signal assignment statement. The output MUX selects A when E = „1‟; or else it selects the output of the first MUX, which is B when D = „1‟, or else it is C.

The following Figure shows a 4-to-1 MUX with four data inputs and two control inputs, A and B. The



```
F <= A when E = '1'
     else B when D = '1'
     else C;
```

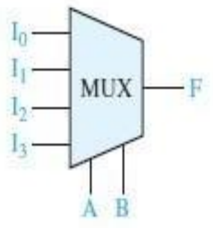
control inputs select which one of the data inputs is transmitted to the output. The logic equation for the 4-to-1 MUX is $F = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$.

One way to model the MUX is with the VHDL statement

*F <= (**not** A **and not** B **and** I0) **or** (**not** A **and** B **and** I1) **or***

*(A **and not** B **and** I2) **or** (A **and** B **and** I3);*

Another way to model the 4-to-1 MUX is to use a conditional assignment statement (given in Figure below):



```
sel <= A&B;
-- selected signal assignment statement
with sel select
  F <= I0 when "00",
       I1 when "01",
       I2 when "10",
       I3 when "11";
```

```
F <= I0 when A = '0' and B = '0'
  else I1 when A = '0' and B = '1'
  else I2 when A = '1' and B = '0'
  else I3;
```

The expression A&B means A concatenated with B, that is, the two bits A and B are merged together to form a 2-bit vector. This bit vector is tested, and the appropriate MUX input is selected. For example, if A = „1‟ and B = „0‟, A&B = "10" and I2 is selected.

Instead of concatenating A and B, we could use a more complex condition also (as given in

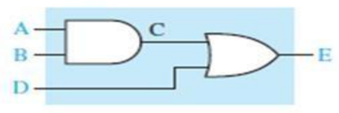
```
with expression_s select
   signal_s <= expression1 [after delay-time] when choice1,
               expression2 [after delay-time] when choice2,
               . . .
               [expression_n [after delay-time] when others];
```

aboveFigure).

First, expression_s is evaluated. If it equals choice1, signal_s is set equal to expression1; if it equals choice2, signal_s is set equal to expression2; etc. If all possible choices for the value of expression_s are given, the last line should be omitted; otherwise, the last line is required. When it is present, if expression_s is not equal to any of the enumerated choices, signal_s is set equal to expression_n. The signal_s is updated after the specified delay-time, or after if the "after delay-time" is omitted.

VHDL MODULES:

To write a complete VHDL module, we must declare all of the input and output signals using an entity declaration, and then specify the internal operation of the module using an architecture declaration. As an example, consider the following Figure.



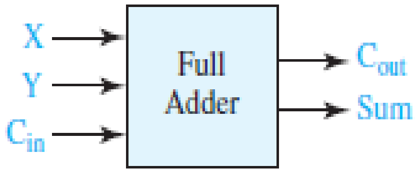
```
entity two_gates is
    port (A,B,D: in bit; E: out bit);
end two_gates;
architecture gates of two_gates is
    signal C: bit;
begin
    C <= A and B; -- concurrent
    E <= C or D; -- statements
end gates;
```

When we describe a system in VHDL, we must specify an entity and architecture at the top level. The entity declaration gives the name "two_gates" to the module. The port declaration specifies the inputs and outputs to the module. A, B, and D are input signals of type bit, and E is an output signal of type bit. The architecture is named "gates". The signal C is declared within the architecture because it is an internal signal. The two concurrent statements that describe the gates are placed between the keywords begin and end.

Example: *To write the entity and architecture for a full adder module.*

The entity specifies the inputs and outputs of the adder module, as shown in the following Figure. The port declaration specifies that X, Y and Cin are input signals of type bit, and that Cout and Sum are output signals of type bit.

The operation of the full adder is specified by an architecture declaration:



```
entity FullAdder is
    port (X,Y,Cin: in bit;          -- Inputs
            Cout, Sum: out bit);  -- Outputs
end FullAdder;
```

VHDL MODULES

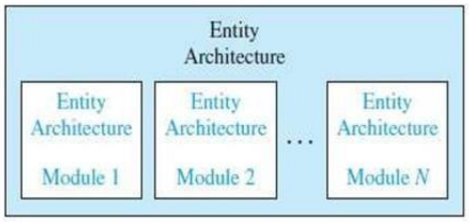
To write a complete VHDL module, we must declare all of the input and output signals using an **entity** declaration, and then specify the internal operation of the module using an **architecture** declaration.
The two concurrent statements that describe the gates are placed between the keywords **begin**
and **end**.

When we describe a system in VHDL, we must specify an entity and an architecture at the top level, and also specify an entity and architecture for each of the component modules that are part of the system. Each entity declaration includes a list of interface signals that can be used to connect to other
modules or to the outside world. We will use entity declarations of the form:

entity entity-name **is** [**port**(interface-signal-
declaration);]**end** [**entity**] [entity-name];

The items enclosed in square brackets are optional. The interface-signal-declarationnormally has
the following form:
entity entity-name **is** [**port**(interface-signal-
declaration);]**end** [**entity**] [entity-name];

FIGURE 10-9
VHDL Program
Structure



The items enclosed in square brackets are optional. The interface-signal-declarationnormally has
the following form:

The curly brackets indicate zero or more repetitions of the enclosed clause. Input signals areof mode **in**, output signals are of mode **out**, and bi-directional signals are of mode **inout.**

Associated with each entity is one or more architecture declarations of the form

architecture architecture-name **of** entity-name **is**
[declarations] **begin**
architecture body
end [**architecture**] [architecture-name];

In the declarations section, we can declare signals and components that are used within the architecture. The architecture body contains statements that describe the operation of the module.

```
architecture Equations of FullAdder is
begin           -- concurrent assignment statements
    Sum <= X xor Y xor Cin after 10 ns;
    Cout <= (X and Y) or (X and Cin) or (Y and Cin) after 10 ns;
end Equations;
```

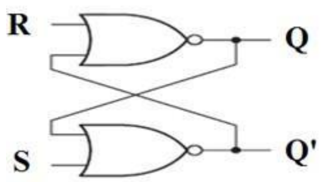
In this example, the architecture name (Equations) is arbitrary, but the entity name (FullAdder) must match the name used in the associated entity declaration.The VHDL assignment statements for Sum and Cout represent the logic equations for the full adder. Several other architectural descriptions such as a truth table or an interconnection of gates could have been used instead. In the Cout equation, parentheses are required around (X and Y) because VHDL does not specify an order of precedence for the logic operators

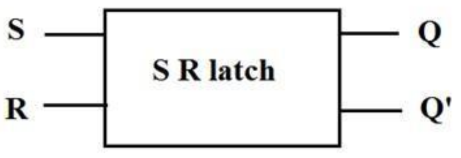
LATCHES AND FLIP FLOPS

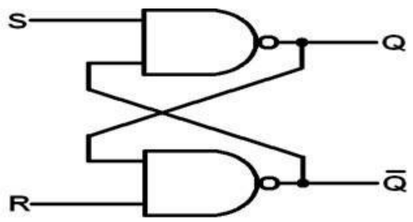
Sequential switching circuits have the property that the output depends not only on the present input but also on the past sequence of inputs. In effect, these circuits must be able to "remember" something about the past history of the inputs in order to produce the present output. Latches and flip-flops are commonly used memory devices in sequential circuits. Basically, latches and flip-flops are memory devices which can assume one of two stable output states and which have one or more inputs that can cause the output state to change.

Set Reset Latch

- Basic storage element is called as latch; it stores a one bit data i.e. either 0 or 1.
- Latches are called as un-clocked flip flops.
- SR latch can be constructed using either two NOR gates or two NAND gates.
- Reset means Q =0, Set means Q=1.

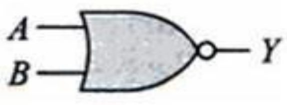




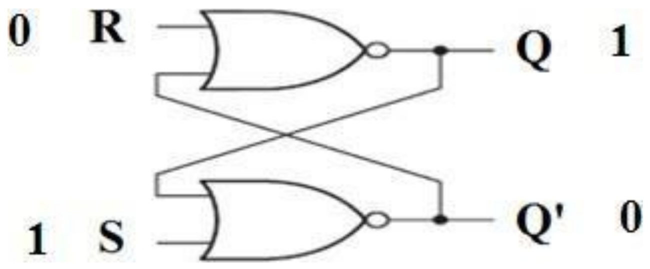


- Let consider S-R latch using NOR gates:

✓ Case(i) When S=0, R=1

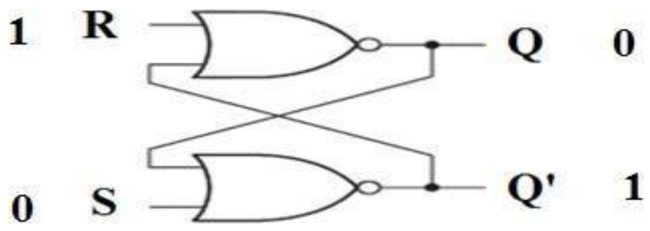


A	*B*	*Y*
0	0	1
0	1	0
1	0	0
1	1	0

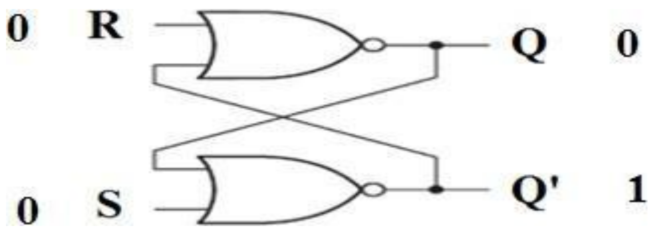


- For nor gate if one of the input is one it produces 0 output.
- i.e. Q is 0 and this leads Qbar is 1

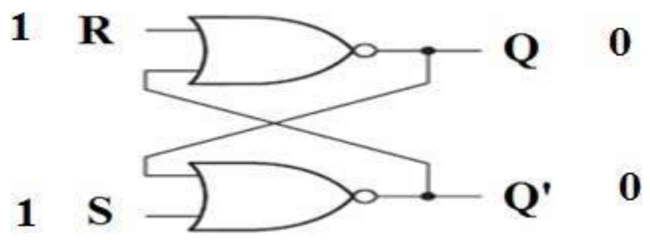
✓ Case(ii) When S=0, R=0



✓ Case(iii) When S=1, R=0



✓ Case (iv) When S=1, R=1, the latch will not operate properly.



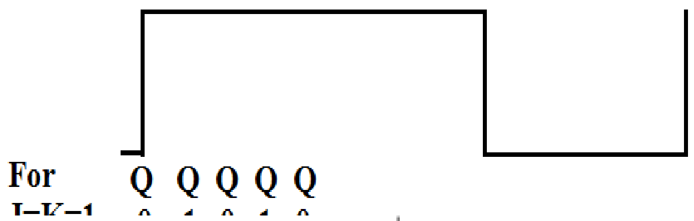
Application of SR latch (Switch debounce):

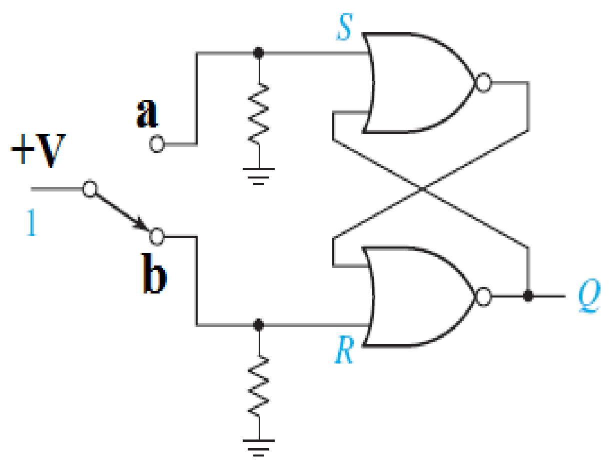
- When mechanical switch is opened or closed they will vibrate several times before settling down to final position.
- This produces a noisy transition and this interferes with the proper operation of logic circuit.
- The pull down resistor are connected to switch contacts **"a"** and **"b"** to ensure that when the switch is between **a** and **b** the input to the SR latch will be al always logic 0 and output will not change.

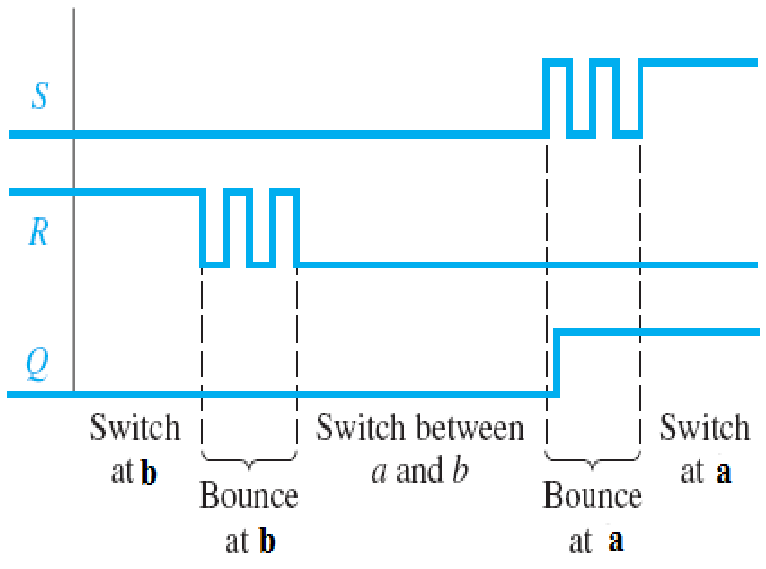
Note: When the contact is move between one point to another point (i.e. when the point between a&b) high impedance state, the pull down resistor pull down the high impedance to 0V.

Note: Race around condition

- For JK flip flop if J=K=1 and clock is too long then the state of flip flop is keep on toggle which leads to difficulty in determining the output of flip flop. This problem is called as race around condition.

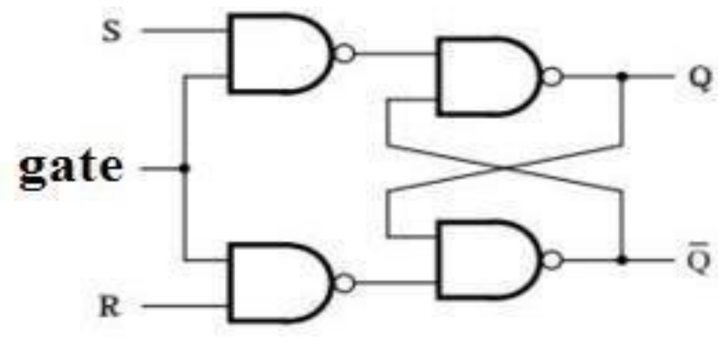






Gated Latches

- **Gated lat**ches have an additional input called gate or enable input.
- When the gate input is inactive the state of the latch is cannot change.
- When gate input is active the state of latch is controlled by other inputs.

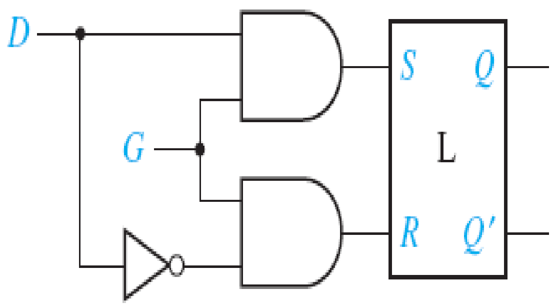


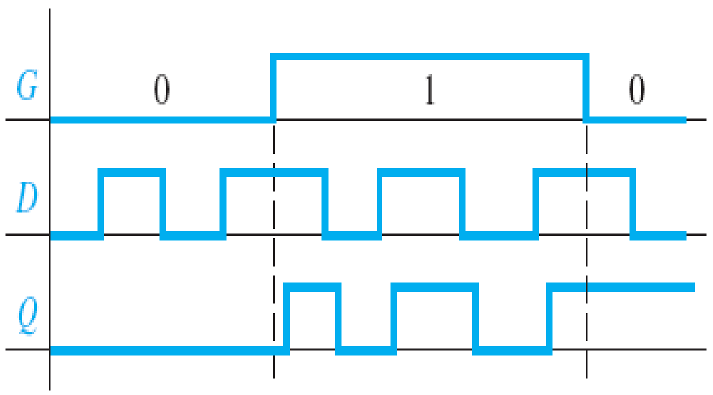
Gated SR latch with NAND gates.

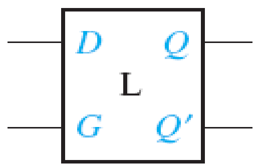
Gated D Latch

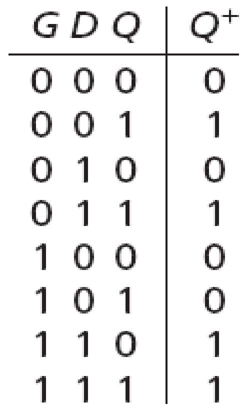
- A gated D latch has two inputs—a data input (*D*) and a gate input (*G*).
- The D latch can be constructed from an S-R latch and two AND gates and an inverter.
- Q of gated D latch remains unchanged when G is inactive.
- Q is equal to D after some delay when G is active

- **Symbol and Truth Table for Gated Latch**

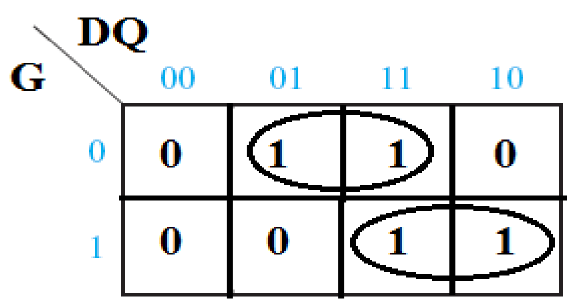








G	D	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



G \ DQ	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$$Q^+ = G'Q + GD$$

Symbol and Truth Table for Gated Latch

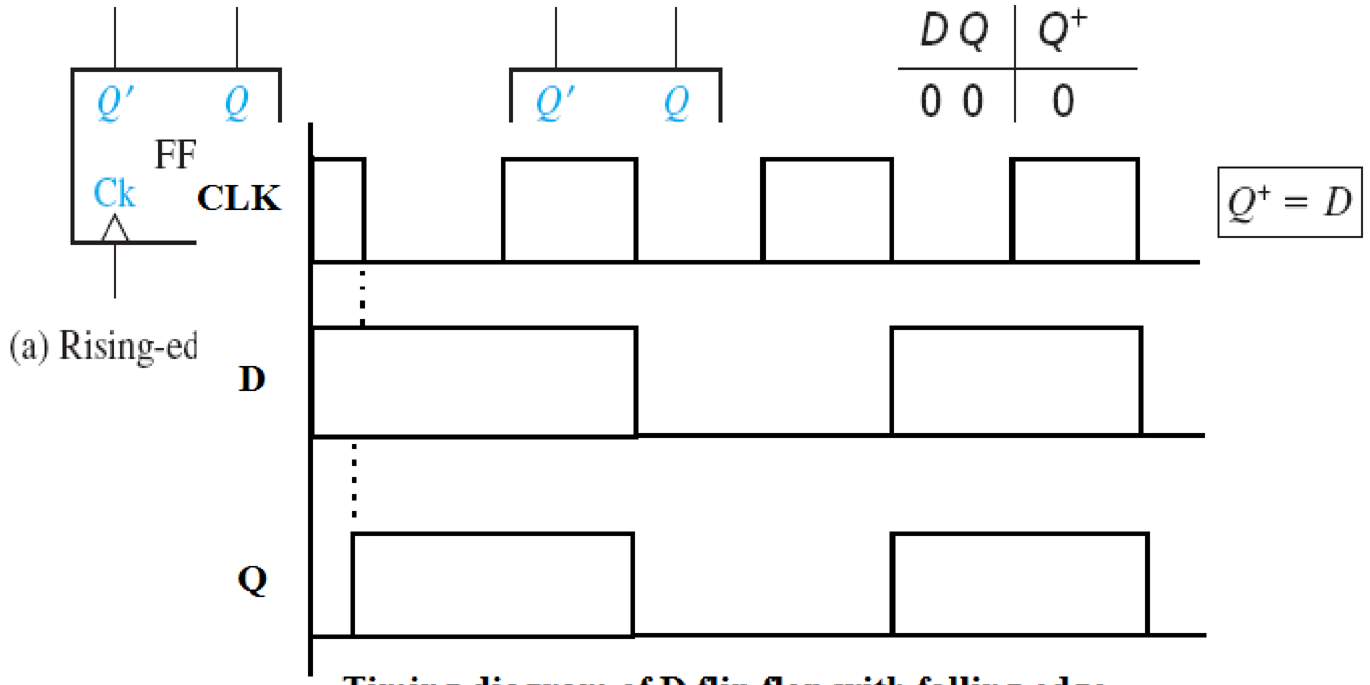
Edge-Triggered D Flip-Flop

- A D flip-flop has two inputs, *D* (data) and Ck (clock).
- The small arrowhead on the flip-flop symbol identifies the clock input.
- If the output of the flip flop is changed in response to a 0 to 1 transition on the clock input, we say that the flip-flop is triggered on the *rising edge* (or positive edge) of the clock.
- No bubble on the clock input indicates a *rising-edge trigger*.
- If the output of the flip flop is changed in response to a 1 to 0 transition on the clock input, we say that the flip flop is triggered on the *falling edge* (or negative edge) of the clock.
- An inversion bubble on the clock input indicates a *falling-edge trigger* .

Flip-Flops:

- A flip-flop stores a one bit of data (i.e. binary digit either 0 or 1).
- If the o/p of flip flop is 0, it stores 0, and if the o/p of flip flop is 1, it stores 1.
- Flip-flops are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

Flip Flop timings:



Timing diagram of D flip flop with falling edge

Set up time: the minimum amount of time required for data inputs to be present before the clock trigger arrives.

Hold time: the minimum amount of time that data must be present after the clock trigger arrives.

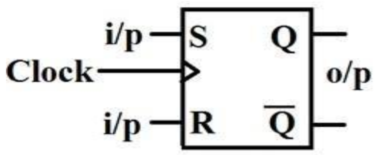
Propagation Delay: the amount of time it takes for the output to change state after an input arrives

Various representations of flip flops:

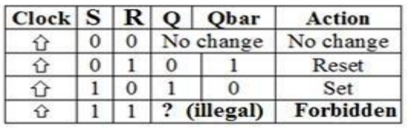
- There are various ways to represent a flip flop.
- The **characteristic equation** is a logic expression describing a flip flop
- The next output Qn+1 is expressed as a function of present output Qn and input of flip flop.
- K-map is used to get the optimized expression.
- For all the flip flops the logic equation are written using SOP form by grouping 1's.
- **Finite state machine** offers better alternative to truth table understanding the logic.
- All the flip flops are represented as FSM through their state transition diagrams.

SR(set-reset) flip-flop:

- In SR flip flop, input S=1 sets the Q o/p to 1 and R =1 resets the Q o/p to 0.
- For SR flip flop, input S=R=1 not allowed we can have don't care "X" in a corresponding location in K-map.
- The difference between SR flip flop and SR latch is clock input.

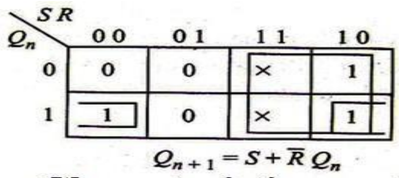


Block Diagram

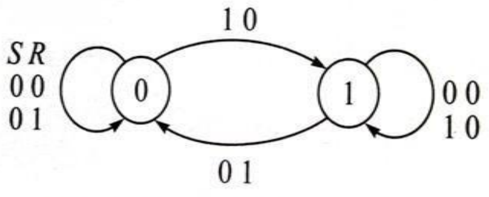


Clock	S	R	Q	Qbar	Action
⇧	0	0	No change	No change	
⇧	0	1	0	1	Reset
⇧	1	0	1	0	Set
⇧	1	1	? (illegal)		Forbidden

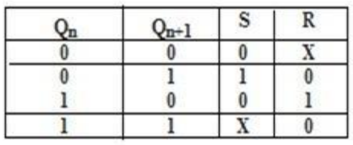
Truth table



Characteristic equation



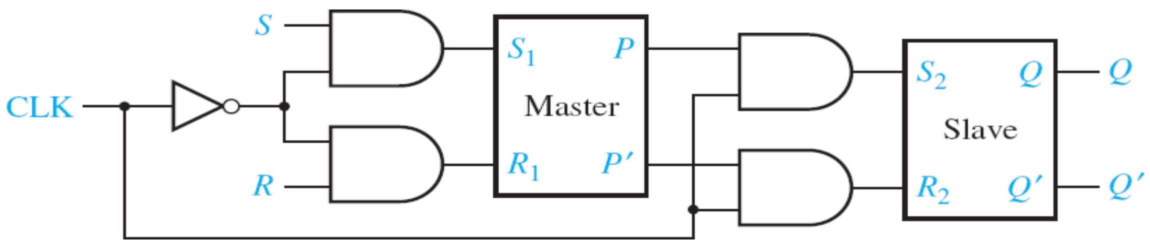
State diagram or FSM

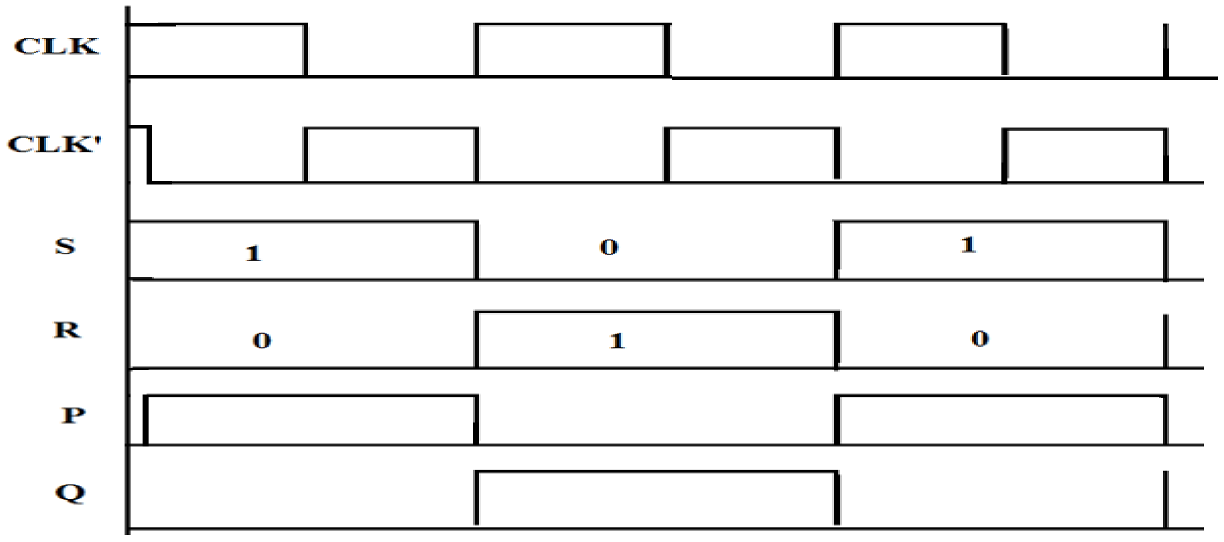


Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table

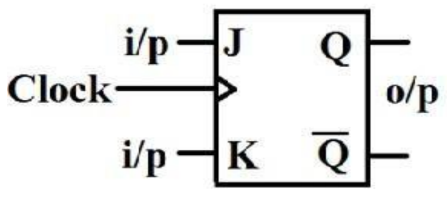
Mater slave S R flip flop Implementation with two latches



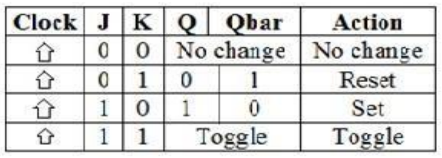


<u>JK (jack kilby) flip flop:</u>

- In S R flip flop when S=R=1 the output is forbidden, to overcome this disadvantage J K flip flop is designed.
- The J-K flip-flop is an extended version of the S-R flip-flop.
- The J-K flip-flop has three inputs—*J*, *K*, and the clock (CK).
- If *J* =1 and *K* = 0, the flip-flop output is set i.e. *Q* =1 after the active clock edge and if *J* =0and *K* =1, the flip-flop output is reset i.e. *Q* =0 after the active edge.
- When *J* =*K* =1, the active edge will cause *Q* to change from 0 to 1, or from 1 to 0.

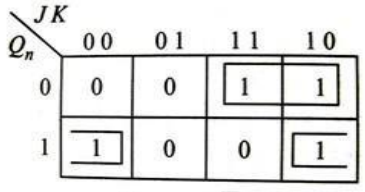


Block Diagram



Clock	J	K	Q	Qbar	Action
⇧	0	0	No change		No change
⇧	0	1	0	1	Reset
⇧	1	0	1	0	Set
⇧	1	1	Toggle		Toggle

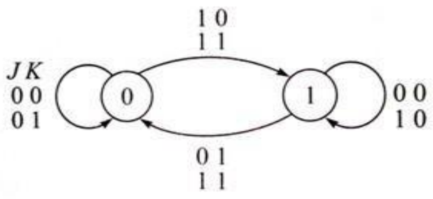
Truth table



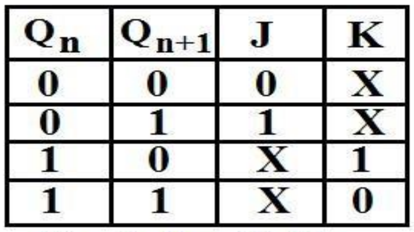
Q_n \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

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

Characteristic equation

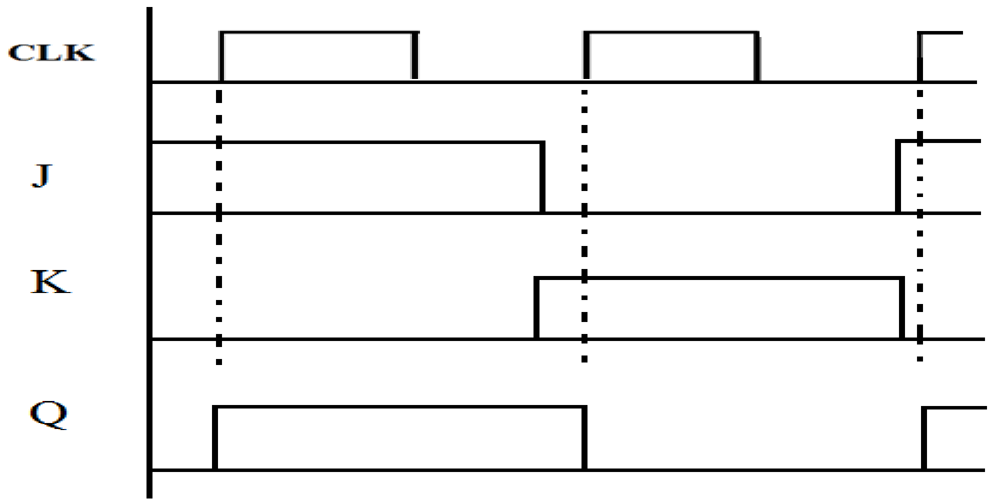


State diagram or FSM

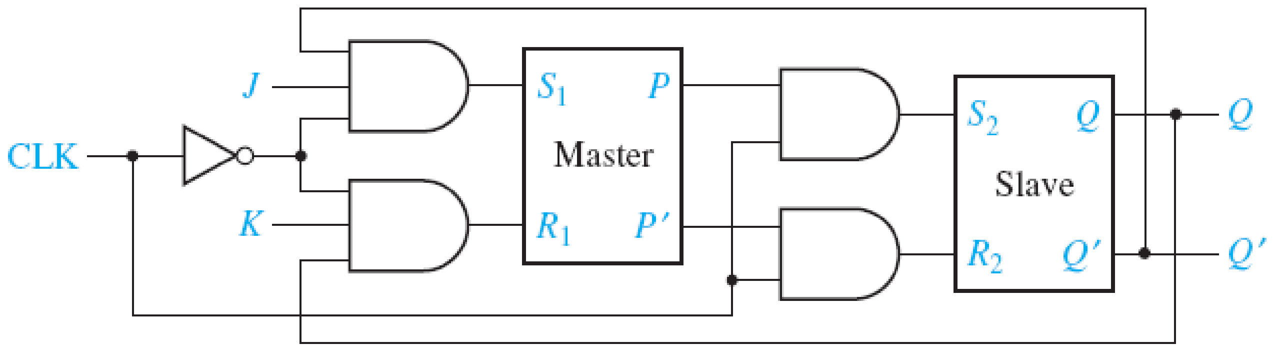


Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table

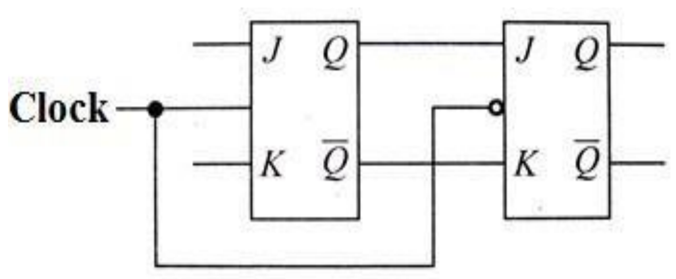


- One way to realize the J-K flip-flop is with two S-R latches connected in a master-slave arrangement, as shown in Figure.



JK - Master Slave Flip Flop:

- A JK master flip flop is positive edge triggered, whereas slave is negative edge triggered. Therefore master first responds to J and K inputs and then slave.



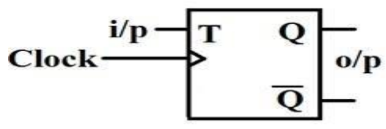
Clock	J	K	Q	Qbar	Action
⇧	0	0	No change		No change
⇧	0	1	0	1	Reset
⇧	1	0	1	0	Set
⇧	1	1	Toggle		Toggle

- If J=1 and K=0, master sets on arrival of positive clock edge. High output Q of the master drives the J input of the slave on negative edge of the clock.
- If J=0 and K=1, master resets on positive clock. High output $\overline{Q}$ of the master drives the K input of the slave. Negative edge of the clock forces the slave to reset.
- If the master's J and K inputs are high, it changes the state or toggles on the positive clock and the slave toggles on the negative clock.
- If J= K= 0, the flip flop is disabled and Q remains unchanged. The slave does exactly what the master does.

D(Delayed flip flop):

T(toggle) flip flop:

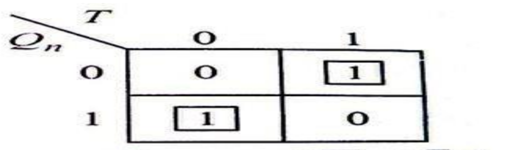
- T flip flop has T and clock input.
- When T=1, flip flop changes the state after the arrival of clock. When T=0, no change of state.



Block Diagram

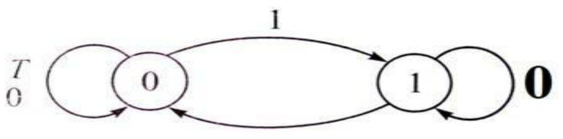
Clock	T	Q
⇧	0	1
⇧	1	0

Truth table



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

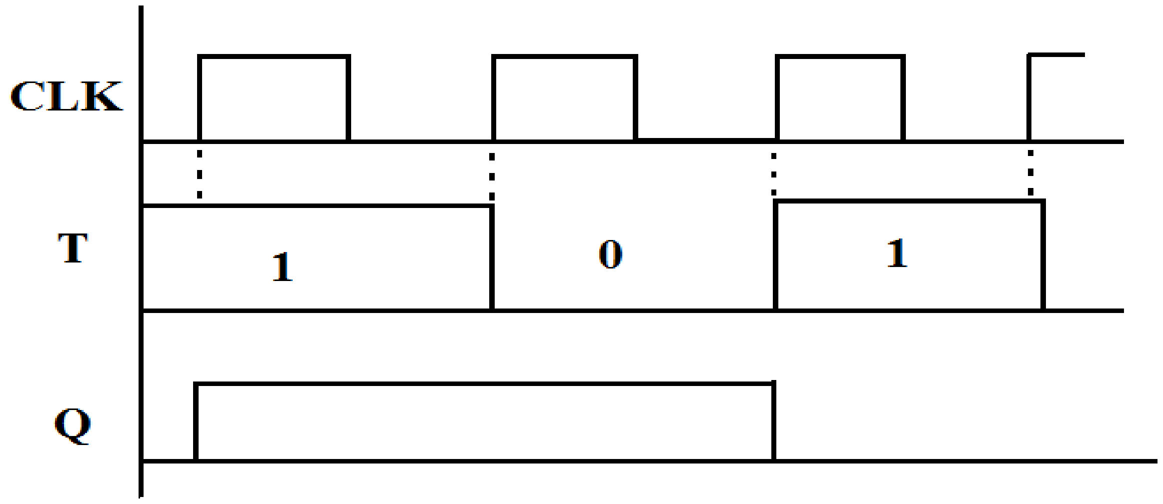
Characteristic equation



State diagram or FSM

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

Excitation Table


CLK
T
1
0
1
Q

Experiment: 7

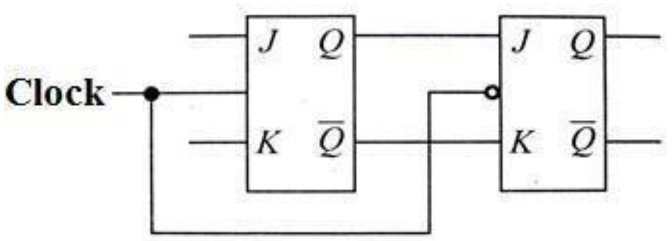
J-K Master/Slave FF using NAND gates

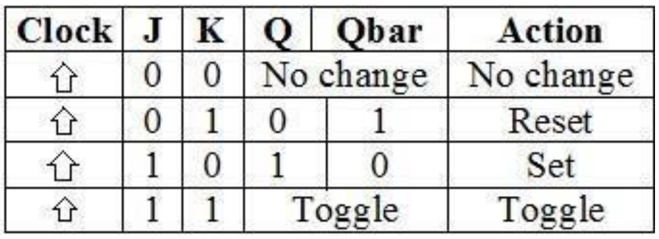
Realize a J-K Master / Slave Flip-Flop using NAND gates and verify its truth table. Andimplement the same in HDL.

<u>Components used:</u>IC 74LS00, IC 74LS10, Power chords, Patch chords, Trainer kit.

<u>Theory:</u>

Master Slave Flip Flop:

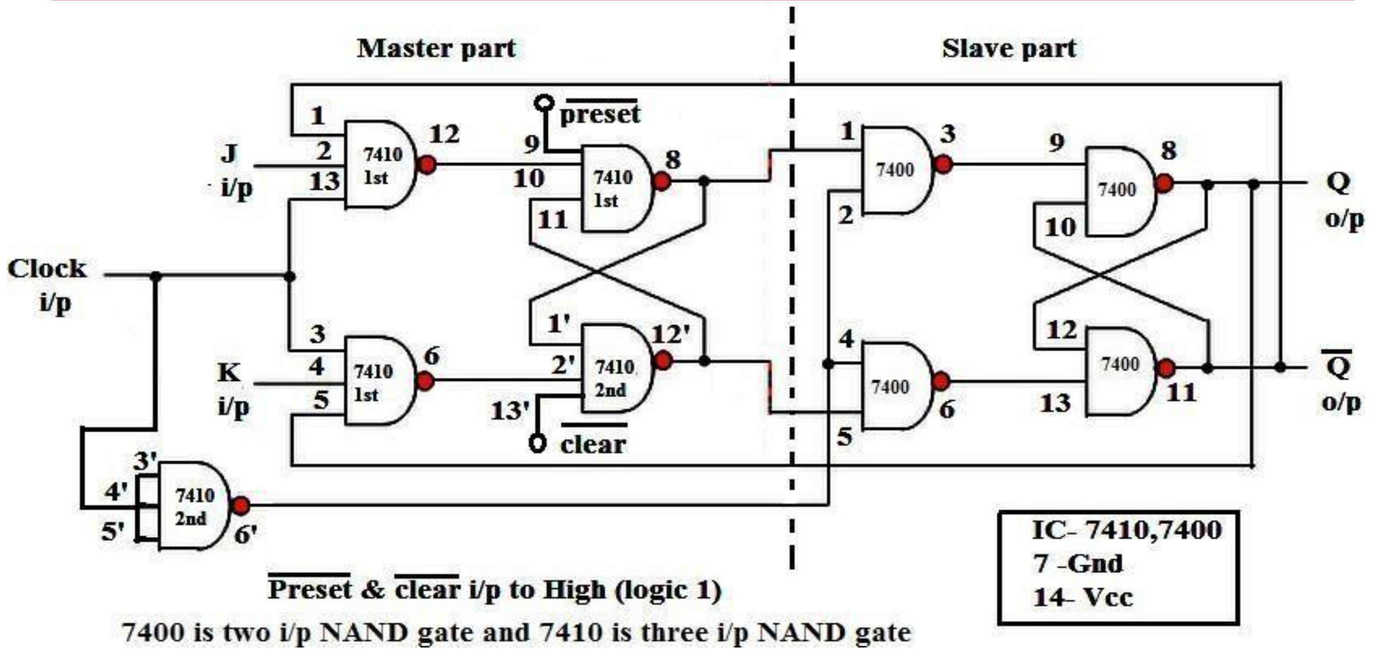




Clock	J	K	Q	Qbar	Action
⇧	0	0	No change		No change
⇧	0	1	0	1	Reset
⇧	1	0	1	0	Set
⇧	1	1	Toggle		Toggle

- A JK master flip flop is positive edge triggered, whereas slave is negative edge triggered.Therefore master first responds to J and K inputs and thenslave.
- If J=1 and K=0, master sets on arrival of positive clock edge. High output Q of the masterdrives the J input of the slave on negative edge of theclock.
- If J=0 and K=1, master resets on positive clock. High output Q of the master drives the Kinput of the slave. Negative edge of the clock forces the slave toreset.
- If the master's J and K inputs are high, it changes the state or toggles on the positive clockand the slave toggles on the negativeclock.
- If J= K= 0, the flip flop is disabled and Q remains unchanged. The slave does exactly whatthe master does.

<u>Circuit Diagram of Master slave JK Flip –Flop using NAND gate:</u>



Procedure:

- Verify all components and patch chords whether they are in good condition ornot
- Make the connection as shown in the circuitdiagram
- Give supply to the trainerkit
- Provide input data to thecircuit
- Verify the truth table sequence and observe theoutputs.

RESULTS:
Verified truth table sequence and observed the outputs.

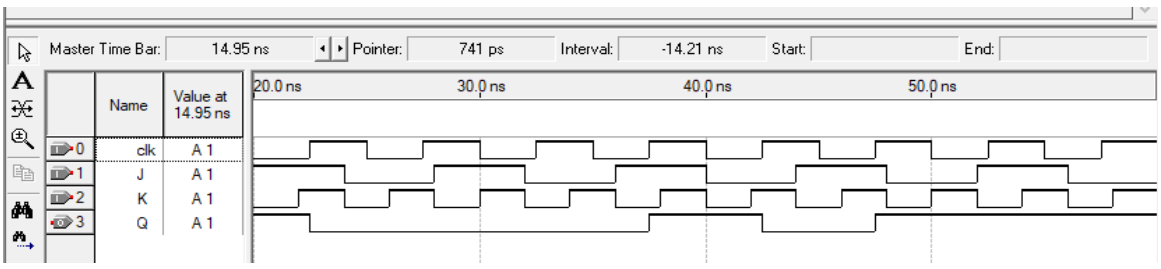
SIMULATION :

VHDL code for JK master slave Flipflop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entityjkflip is
    Port ( J, K, clk : in std_logic;
                   Q : buffer std_logic);
endjkflip;

architecture Behavioral of jkflipis
begin
 process(clk)
begin
    ifrising_edge(clk) then
     Q<= ((J and (not Q)) or ((not K) and Q));
  end if;
end process;
end Behavioral;
```

Waveform


Master Time Bar: 14.95 ns
Pointer: 741 ps
Interval: -14.21 ns
Start:
End:
Name
Value at 14.95 ns
20.0 ns
30.0 ns
40.0 ns
50.0 ns
0 clk A 1
1 J A 1
2 K A 1
3 Q A 1

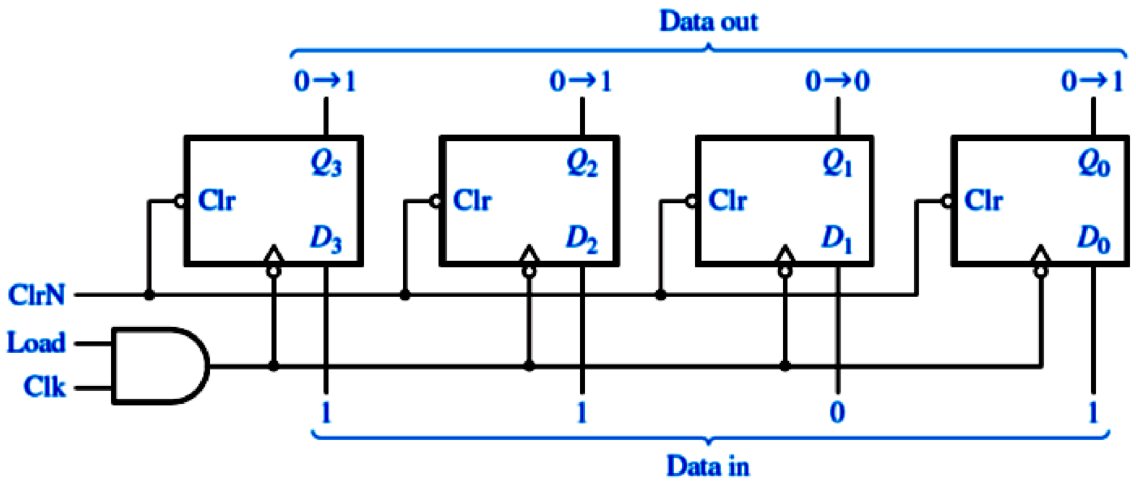
Module -5 Registers Counters, and Sequential Circuits

- ❖ A register consists of a group of flip-flops with a common clock input.
- ❖ Registers are commonly used to store and shift binary data.

Registers and Register Transfers

Registers:

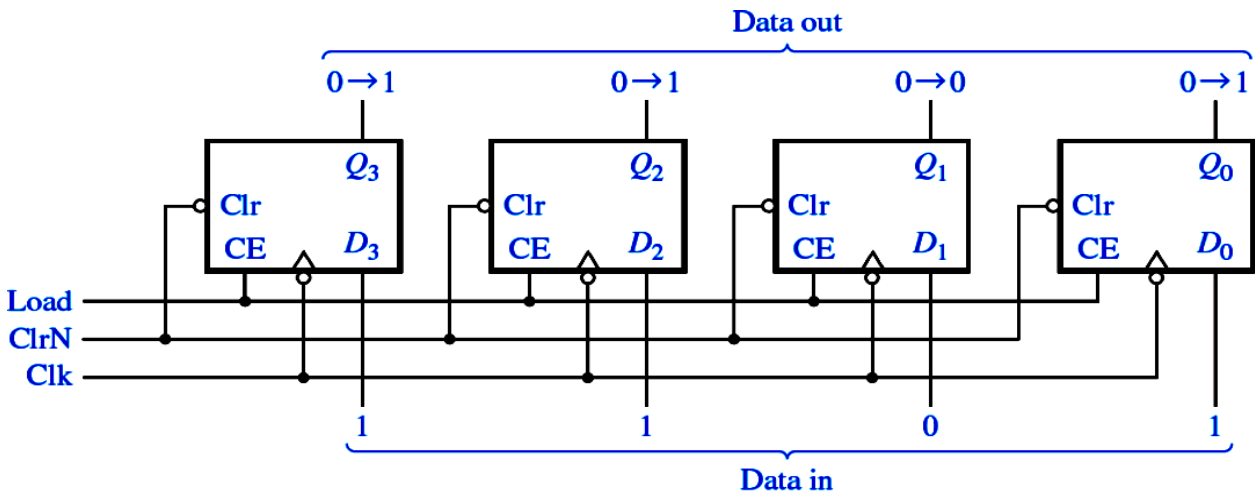
- Several D flip-flops may be grouped together with a common clock to form a register [Figure-a].
- Because each flip-flop can store one bit of information, this register can store four bits of information.
- This register has a load signal that is **ANDed** with the **clock**.
- When Load=0, the register is not clocked, and it holds its present value.
- When Load=1, the clock signal (Clk) is transmitted to the flip-flop clock inputs and the data can be loaded into the flip-flops on the falling edge of the clock.
- For example, if the *Q* outputs are 0000 (*Q*3=*Q*2=*Q*1=*Q*0=0) and the data inputs are 1101 (*D*3=1, *D*2=1, *D*1=0 and *D*0=1), after the falling edge *Q* will change from 0000 to 1101 as indicated.



(a) using gated clock

4-Bit D Flip-Flop Registers with Data, Load, Clear, and Clock Inputs

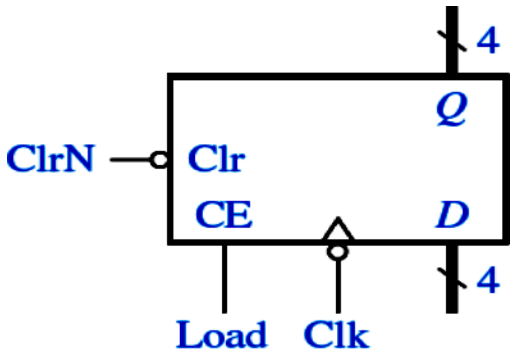
- If flip-flops have clock, the register can be designed as shown in Figure-(b).



(b) With clock enable

- The load signal is connected to all four CE inputs. When Load =0, the clock is disabled and the registe

- holds its data.
- When Load is 1, the clock is enabled, and data can be loaded into the flip-flops on the falling edge of the clock.
- Figure (c) shows a symbol for the 4-bit register using bus notation for the *D* inputs and *Q* outputs.



(c) Symbol

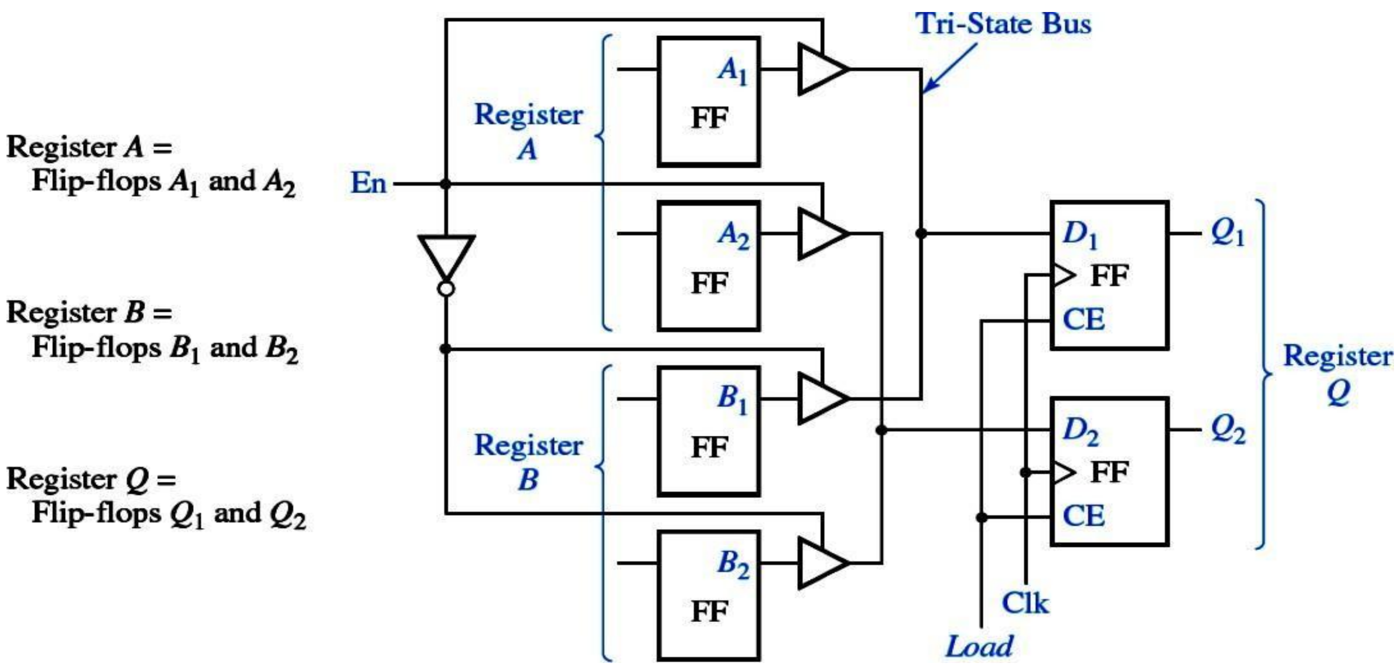
Register Transfers:

- Figure below shows how data can be transferred from the output of one of two registers into a third register using tri-state buffers.
- If En =1 and Load =1, the output of register *A* is enabled onto the tri-state bus and the data in register *A* will be stored in *Q* after the rising edge of the clock.
- If En =0 and Load =1, the output of register *B* will be enabled onto the tri-state bus and stored in *Q* after the rising edge of the clock.

Data Transfer between Registers

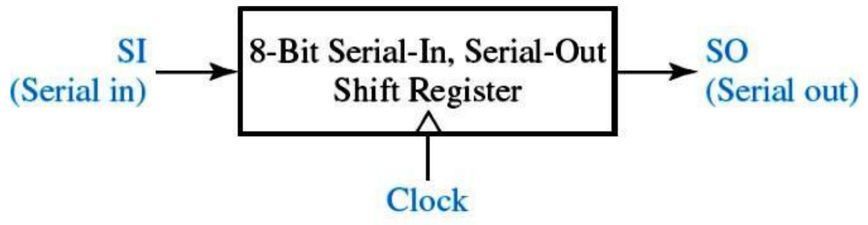
<u>Shift Registers</u>

- A shift register is a register in which binary data can be stored, and this data can be shifted to the left or right when a shift signal is applied.
- Shifting the 1 bit data at a time in a serial fashion, beginning with either the most significant bit (MSB) or the least significant bit (LSB) is referred to as **serial shifting.**
- Shifting all the data bits simultaneously and is referred to as **parallel shifting.**
- TYPES OF REGISTERS: Serial in-Serial out, Serial in-Parallel out, Parallel in-Serial out and Parallel
- in-Parallel out.


Tri-State Bus
Register A = Flip-flops A1 and A2
Register B = Flip-flops B1 and B2
Register Q = Flip-flops Q1 and Q2
En
Register A
Register B
A1
A2
B1
B2
FF
D1
D2
CE
Q1
Q2
Register Q
Clk
Load

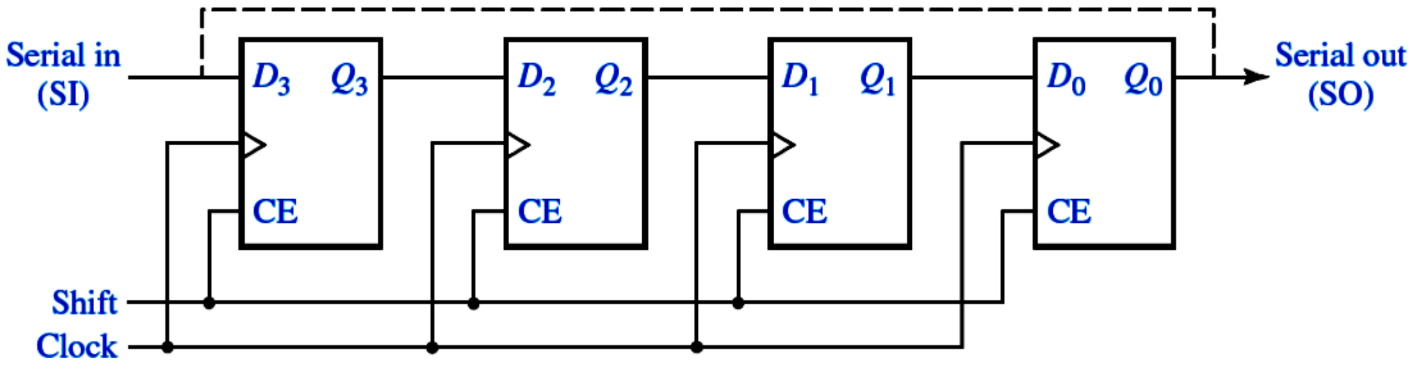
SERIAL IN-SERIAL OUT

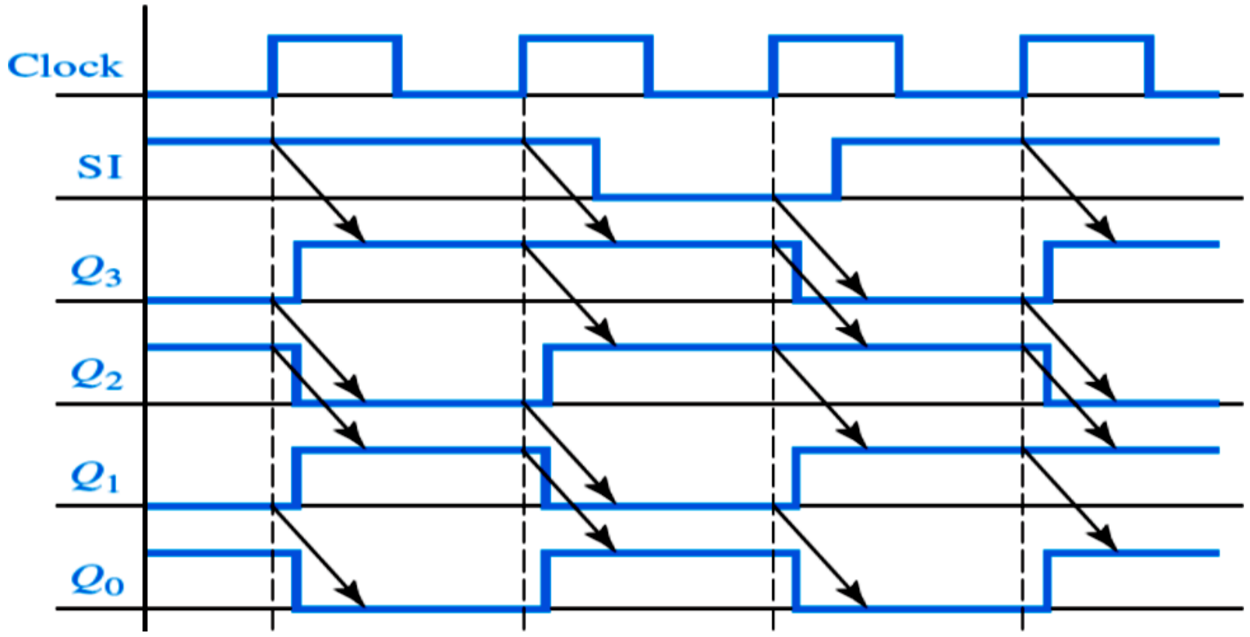
- ❖ Figure (a) illustrates a 4-bit right-shift register with serial input and output constructed from D flip-flops.
- ❖ When Shift =1, the clock is enabled and shifting occurs on the rising clock edge.
- ❖ When Shift = 0, no shifting occurs and the data in the register is unchanged.
- ❖ The serial input (SI) is loaded into the first flip-flop (*Q*3) by the rising edge of the clock. At the same time, the output of the first flip-flop is loaded into the second flip-flop, the output of the second flip-flop is loaded into the third flip-flop, and the output of the third flip-flop is loaded into the last flip-flop.
- ❖ Figure (b) illustrates the timing when the shift register initially contains 0101 and the serial input sequence is 1, 1, 0, 1.The sequence of shift register states is 0101, 1010, 1101, 0110, 1011.
- ❖



(a) Block diagram

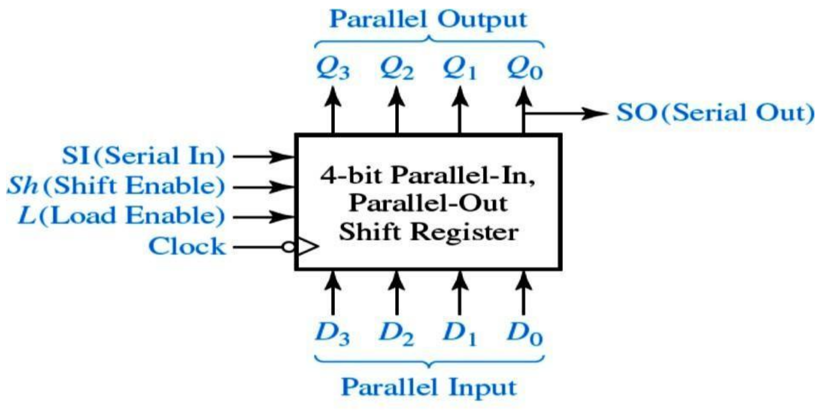
❖ *Serial in* means that data is shifted into the first flip-flop one bit at a time, *Serial out* means that data can only be read out of the last flip flop.





PARALLEL IN-PARALLEL OUT

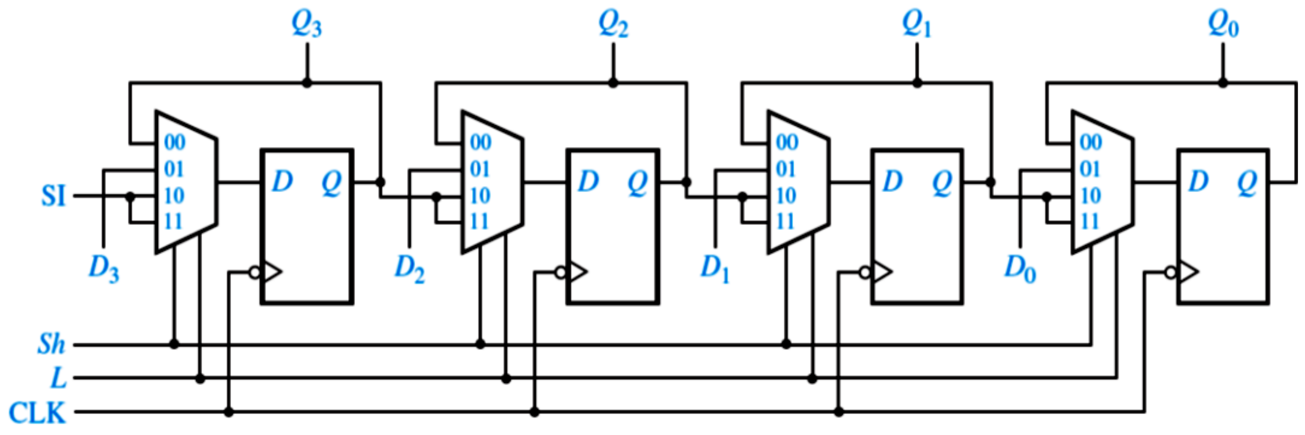
- Figure shows a 4-bit parallel-in, parallel-out shift register.
- Parallel in implies that all four bits can be loaded at the same time, and parallel-out implies that all bits can be read out at the same time.
- The shift register has two control inputs, shift enable (*Sh*) and load enable (*L*).
- If *Sh* =1 (and *L* =1 or *L* = 0), clocking the register causes the serial input (SI) to be shifted into the first flip-flop, while the data in flip-flops *Q*3,*Q*2, and *Q*1 are shifted right.
- If *Sh* =0 and *L* =1, clocking the shift register will cause the four data inputs (*D*3, *D*2, *D*1, *D*0) to be loaded in parallel into the flip-flops.
- If *Sh* = *L*= 0, clocking the register causes no change of state.
- Table below summarizes the operation of this shift register. All state changes occur immediately following the falling edge of the clock.



(a) Block diagram

Inputs		Next State				
Sh (Shift)	*L* (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No change
0	1	D_3	D_2	D_1	D_0	Load
1	X	SI	Q_3	Q_2	Q_1	Right shift

❖ The shift register can be implemented using MUXes and D flip-flops, as shown in below Figure.



(b) Implementation using flip-flops and MUXes

❖ For the first flip-flop, when *Sh* = *L* = 0, the flip-flop *Q*3 output is selected by the MUX and no state change occurs.

❖ When *Sh* =0 and *L* =1, the data input *D*3 is selected and loaded into the flip-flop.

$$Q_3^+ = Sh' \cdot L' \cdot Q_3 + Sh' \cdot L \cdot D_3 + Sh \cdot \mathrm{SI}$$
$$Q_2^+ = Sh' \cdot L' \cdot Q_2 + Sh' \cdot L \cdot D_2 + Sh \cdot Q_3$$
$$Q_1^+ = Sh' \cdot L' \cdot Q_1 + Sh' \cdot L \cdot D_1 + Sh \cdot Q_2$$
$$Q_0^+ = Sh' \cdot L' \cdot Q_0 + Sh' \cdot L \cdot D_0 + Sh \cdot Q_1$$

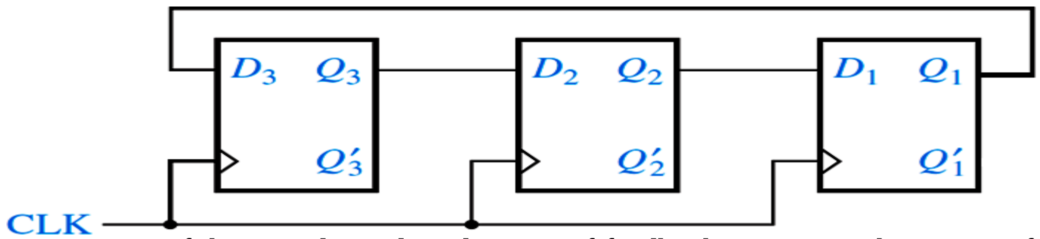
❖ When *Sh* =1 and *L* = 0 or 1, SI is selected and loaded into the flip-flop.

The second MUX selects *Q*2, *D*2, or *Q*3, etc

APPLICATIONS OF SHIFT REGISTERS

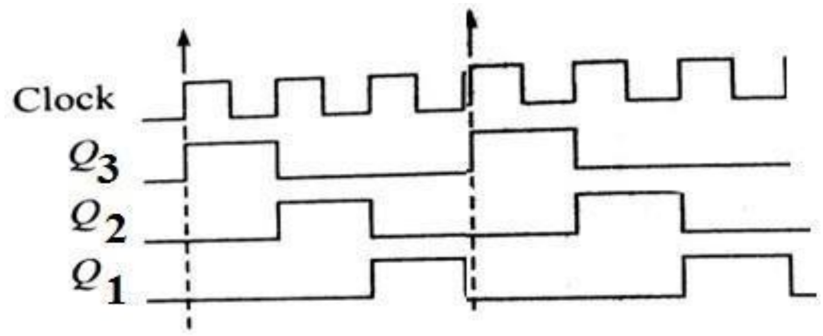
Ring Counter, Switched Tail Counter or *Johnson counter* etc.

Ring Counter



One of the most logical application of feedback is connect the output of the last flip flop **Q_1** back to the

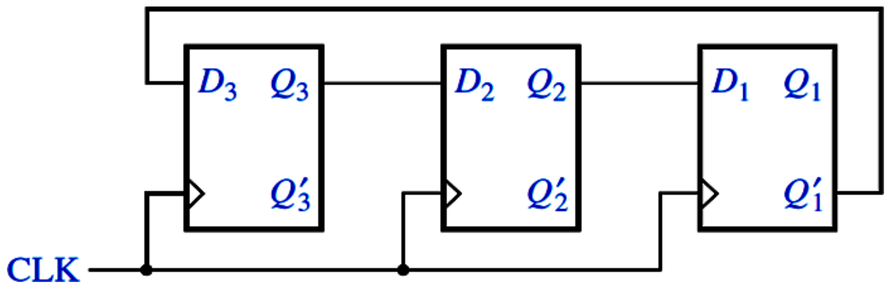
- Data input of the first flip flop **D3**.
- Ring counter is a basic shift register with direct feedback such that the contents of the register simply circulate around the register when the clock is running.



- Figure above shows the Ring Counter.
- Suppose all flip-flops are reset and the clock is allowed to run , then nothing happens because every time the clock goes high, the zero in each flip-flop will be shifted into the next flip-flop, while the zero in the last flip-flop ‘Q1’ will travel around the feedback loop and shift into the first flip-flop D3.
- Let consider, Q_3 is high and all other flip-flops are low. Apply the clock pulse. We can observe the following waveform.
- From the above waveform, we can observe ‘1’ is circulated around the register in a clockwise dirction, moving ahead one flip-flop with each clock . Henece it is reffered as a circulating register or a Ring Counter.

<u>Johnson counters:</u>

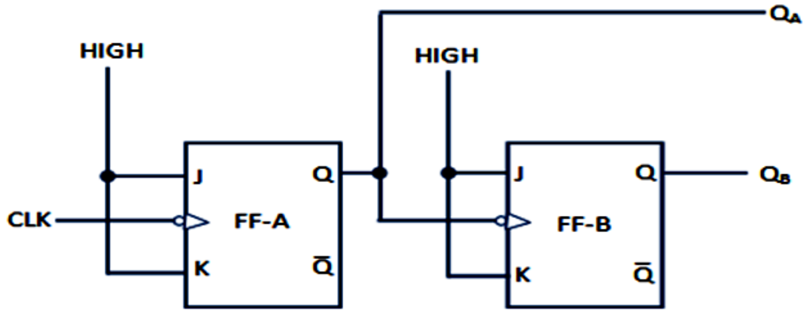
- Figure below shows a 3-bit shift register with the $Q1'$ output from the last flip flop fed back into the D input of the first flip-flop.
- A shift register with inverted feedback is often called a *Johnson counter*.



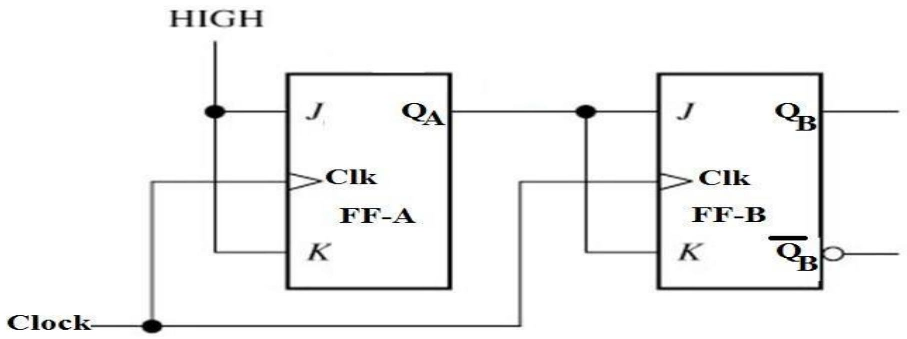
COUNTERS:

- A counter is a register capable of counting the number of clock pulses arriving at its clock input.
- In up counter on arriving of each clock pulse, the counter is incremented by one. In case of down counter it is decremented by one.
- The counter can be used as an instrument for measuring time and therefore period or frequency.
- There are basically two different types of counters—synchronous and asynchronous.

Asynchronous Counter: in this type of counter flip-flops are connected such a way that output of first flip flop drives the clock input of the next flip flop. All the flip flops are not clocked simultaneously. Design involves simple logic circuits. It is slower. Only fixed sequence can be designed.



Synchronous Counter: in this type of counter flip-flops there is no connection between output of first flip flop and the clock input of the next flip flop. All the flip flops are clocked simultaneously.. Design involves complex logic circuits. It is faster. Any required sequence can be designed.

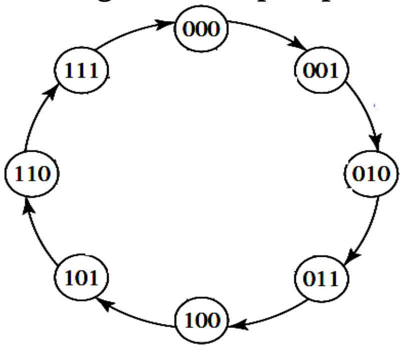


- ❖ All of the counter progress one count at a time and they all have a modulus given by 2^n, where n indicates the number of flip-flops. such counters are said to have a **"natural count" of 2^n.**
- ❖ **For ex:** A mod-2 counter consists of a single flip-flop, mod-4 counter requires two flip-flops, mod-8 counter requires three flip-flops, mod-16 counter requires four flip-flops and so on.
- ❖ Thus we can construct counters that have a natural count of 2, 4, 8, 16, 32, and so on by using the proper number of flip-flops.
- ❖ For example, a counter having a modulus of 3, or 5 constructed from a larger modulus counter by skipping states. Such counters are said to have a **modified count.**
- ❖ Example, a mod-7 counter requires three flip-flops, since 8 is the lowest natural count greater than thedesired modified count of 7

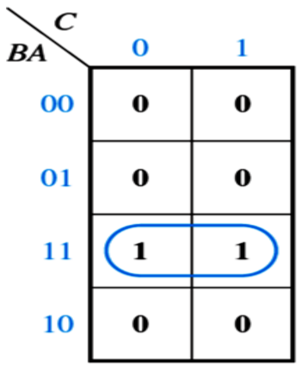
Design of Binary Counters

- The counters discussed in this chapter are all synchronous counters. This means the operation of the flip-flops is synchronized by a common clock pulse

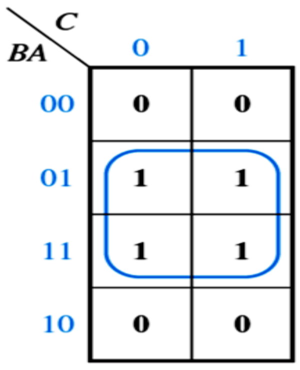
(i) Design Mod-8 binary counter using three T flip-flops to count clock pulses.



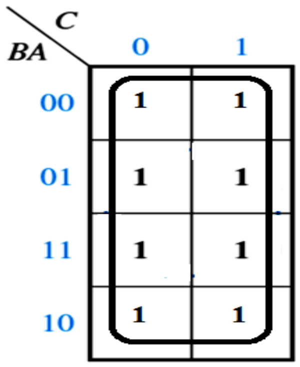
Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1



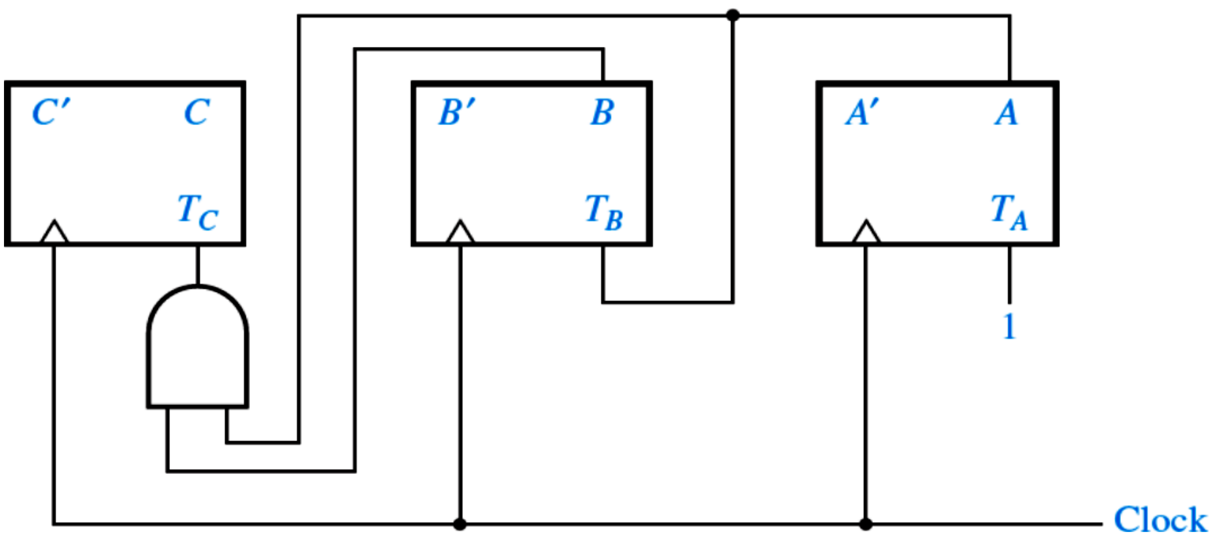
$T_C = BA$



$T_B = A$

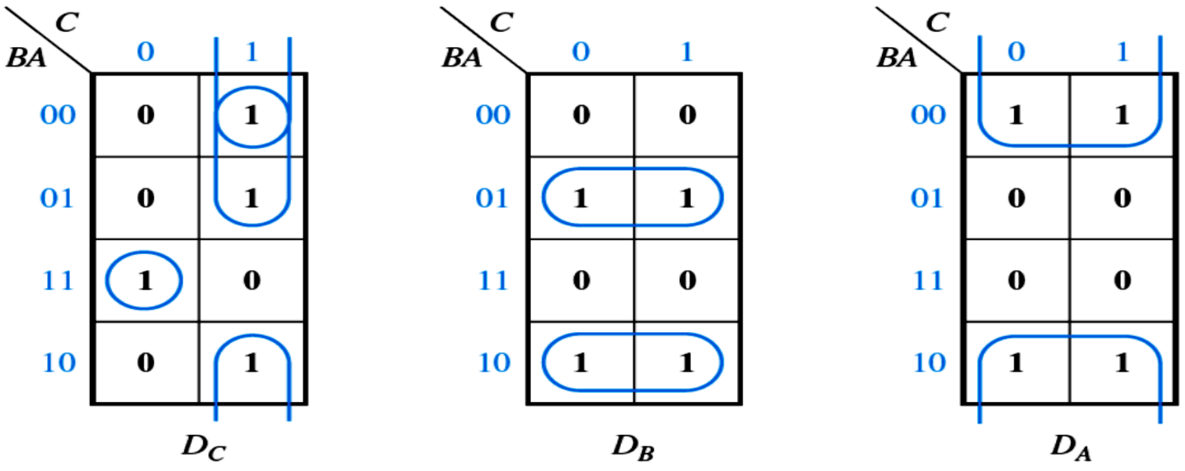


$T_A = 1$

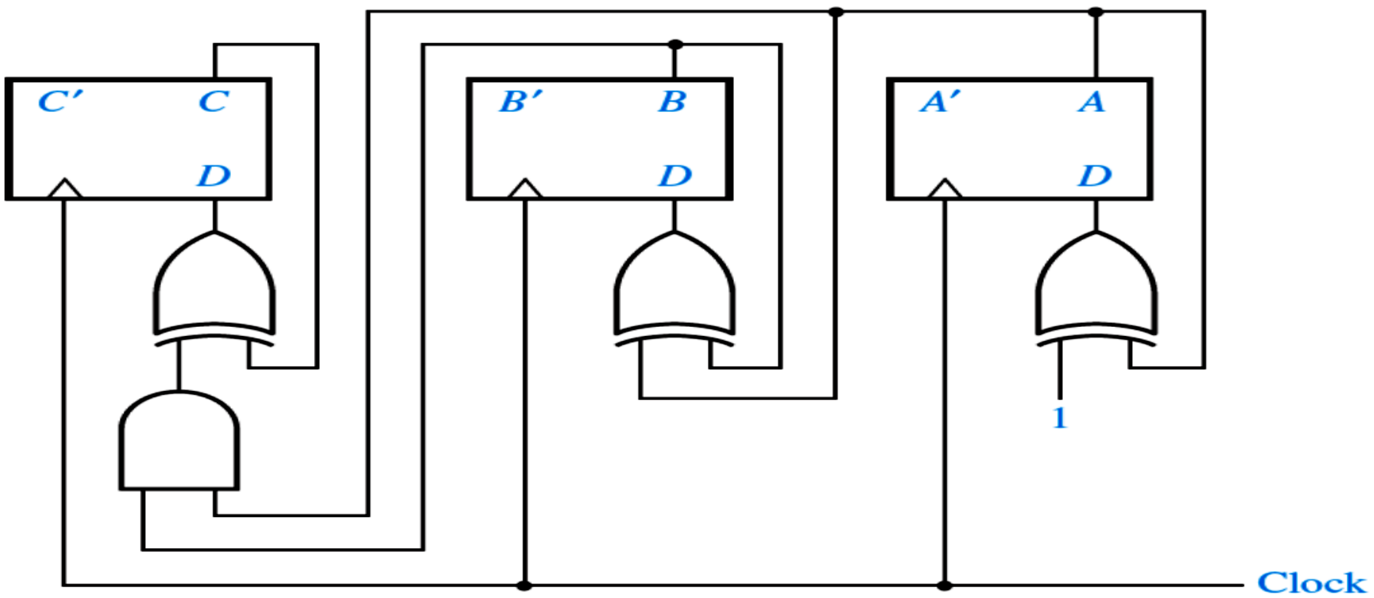


(ii) Design Mod-8 binary counter using three D flip-flops to count clock pulses.

Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	D_C	D_B	D_A
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

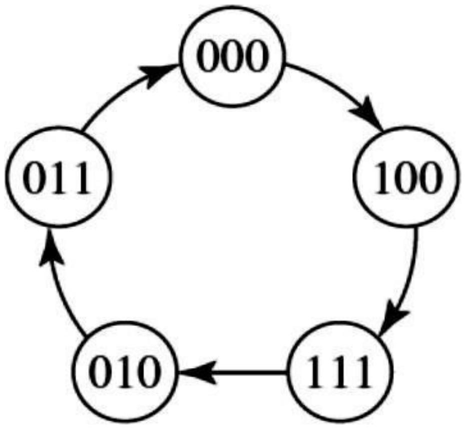


$$D_A = A'$$
$$D_B = BA' + B'A = B \oplus A$$
$$D_C = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$



Counters for Other Sequences Using T Flip Flop

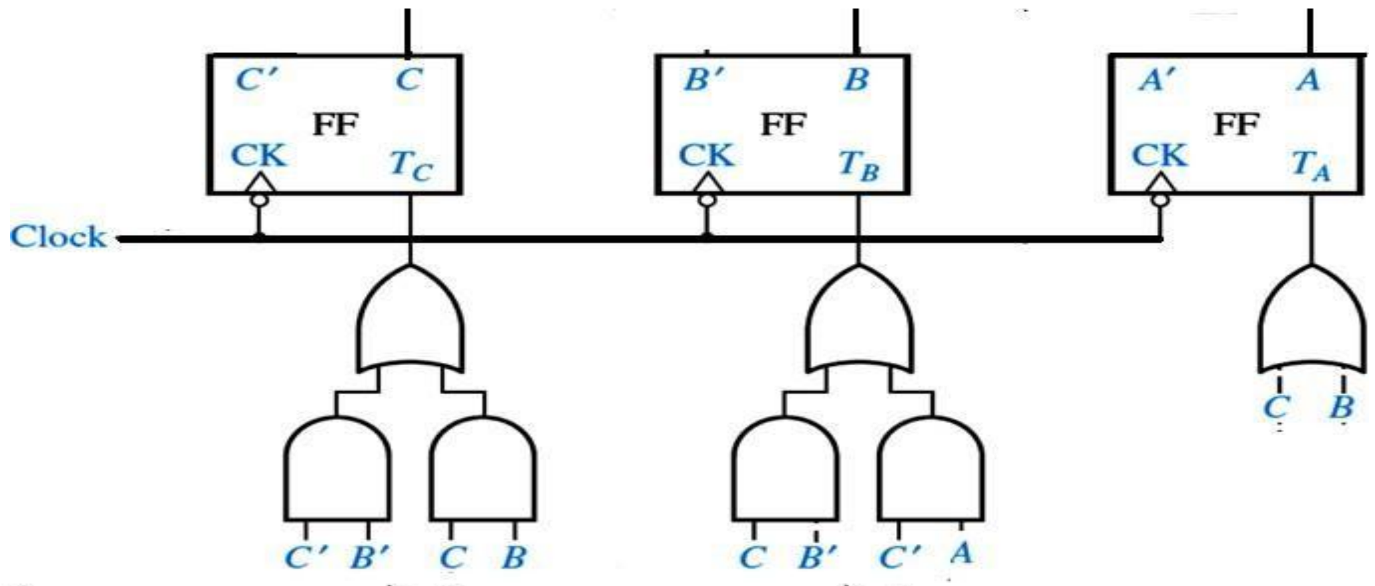
- ❖ In some applications, the sequence of states of a counter is not in straight binary order.
- ❖ Figure below shows the state graph for such a counter. The arrows indicate the state sequence.

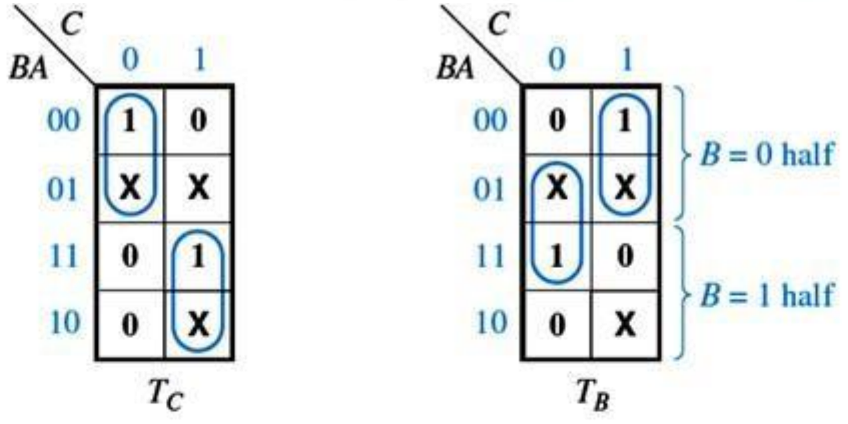


C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0	1	0	0
0	0	1	X	X	X	X	X	X
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	1	1	1
1	0	1	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X
1	1	1	0	1	0	0	0	1

EXCITATION TABLE FOR T-FLIP FLOP

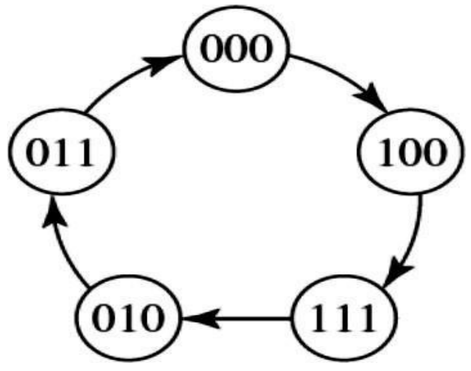
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0





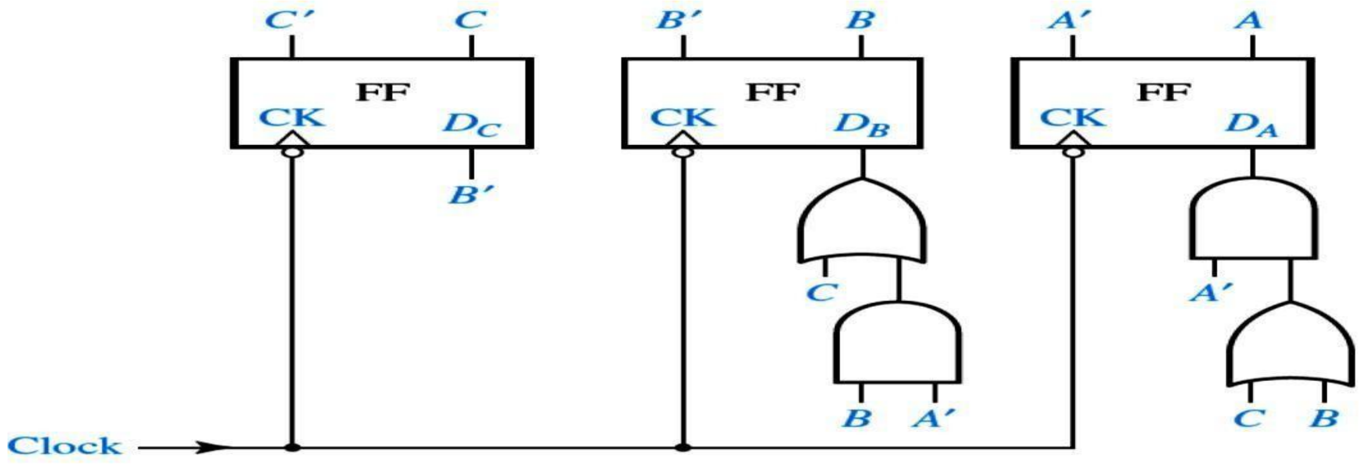
$T_C = C'B' + CB$ $T_B = C'A + CB'$ $T_A = C + B$

Counter Design Using D Flip-Flops

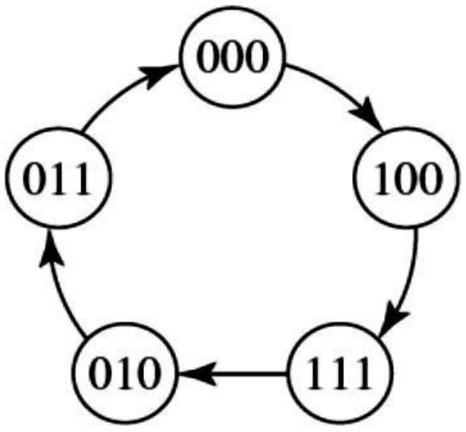


Note: draw the table for next ,present state and flip flop input using excitation table of D flip flop

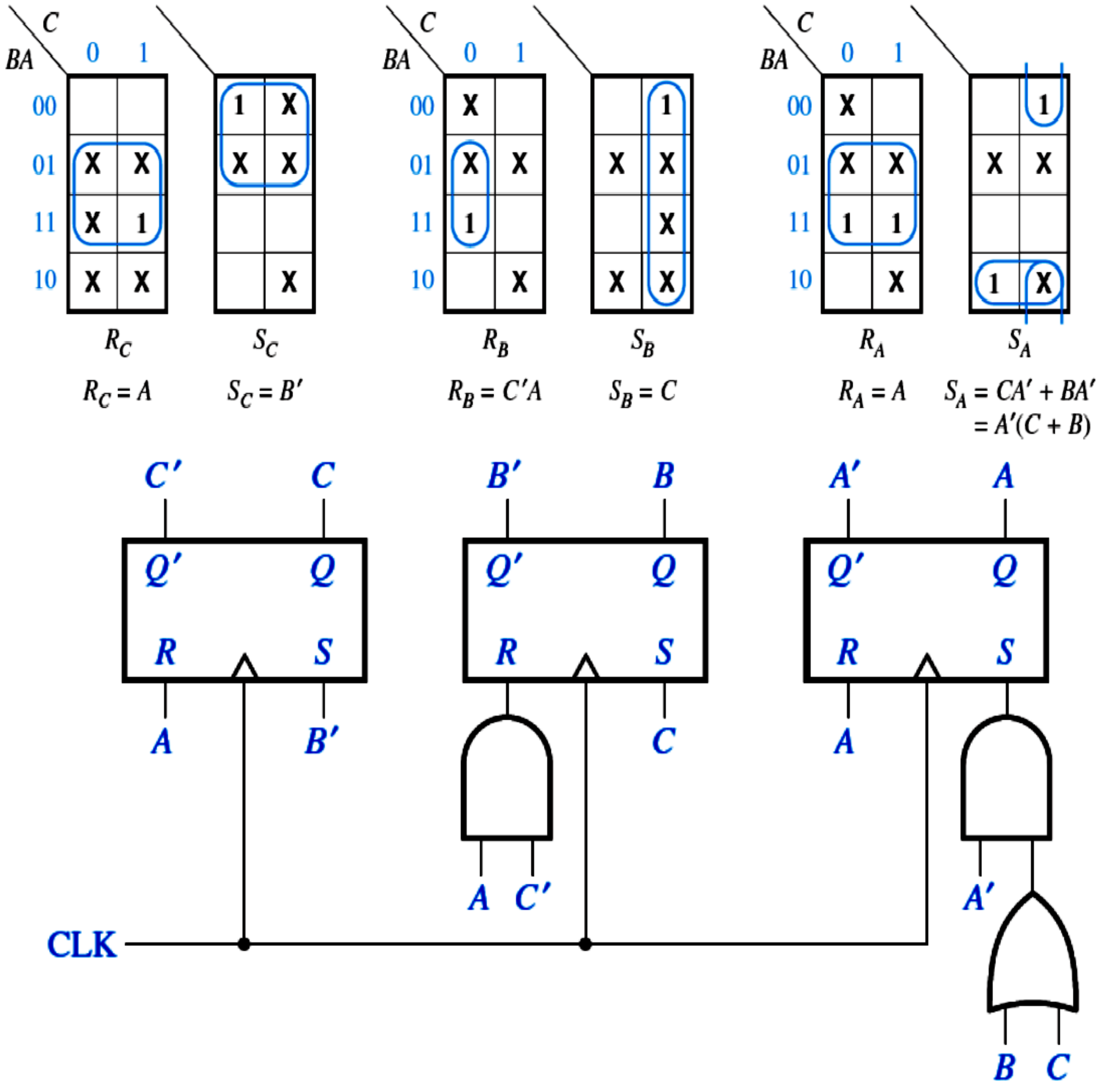
$$D_C = B'$$
$$D_B = C + BA'$$
$$D_A = CA' + BA' = A'(C + B)$$



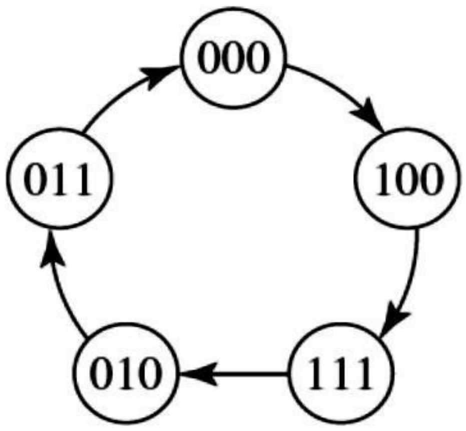
Counter Design Using S-R Flip-Flops



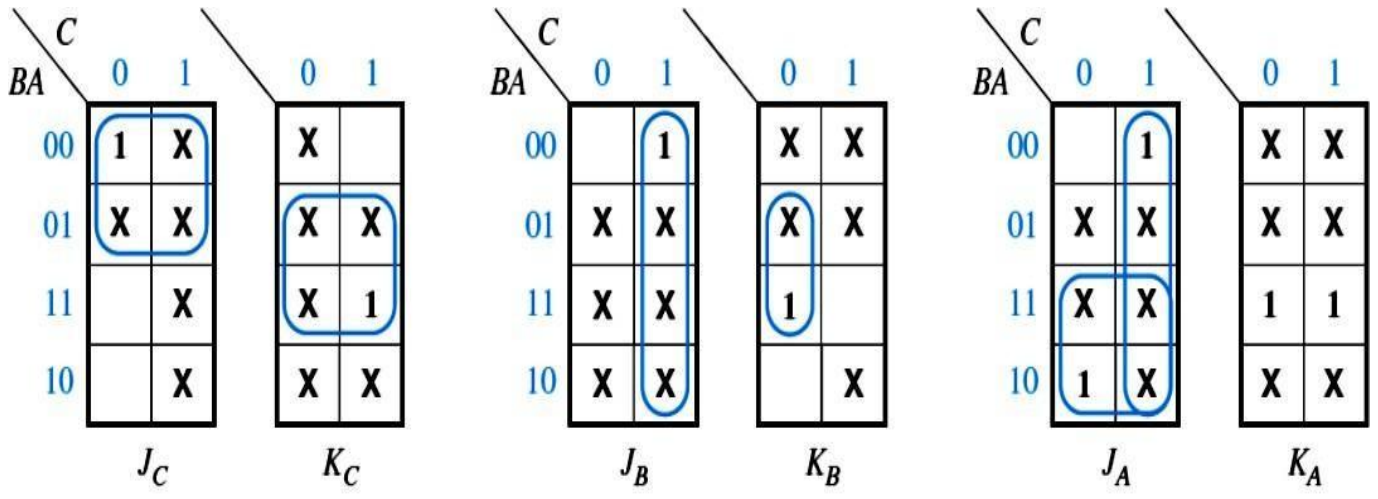
C	B	A	C^+	B^+	A^+	S_C	R_C	S_B	R_B	S_A	R_A
0	0	0	1	0	0	1	0	0	X	0	X
0	0	1	X	X	X	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	0
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1


C
BA
0
1
00
01
11
10
R_C
S_C
R_B
S_B
R_A
S_A
$R_C = A$
$S_C = B'$
$R_B = C'A$
$S_B = C$
$R_A = A$
$S_A = CA' + BA' = A'(C + B)$
C'
C
B'
B
A'
A
Q'
Q
R
S
A
B'
A C'
C
A
A'
B C
CLK

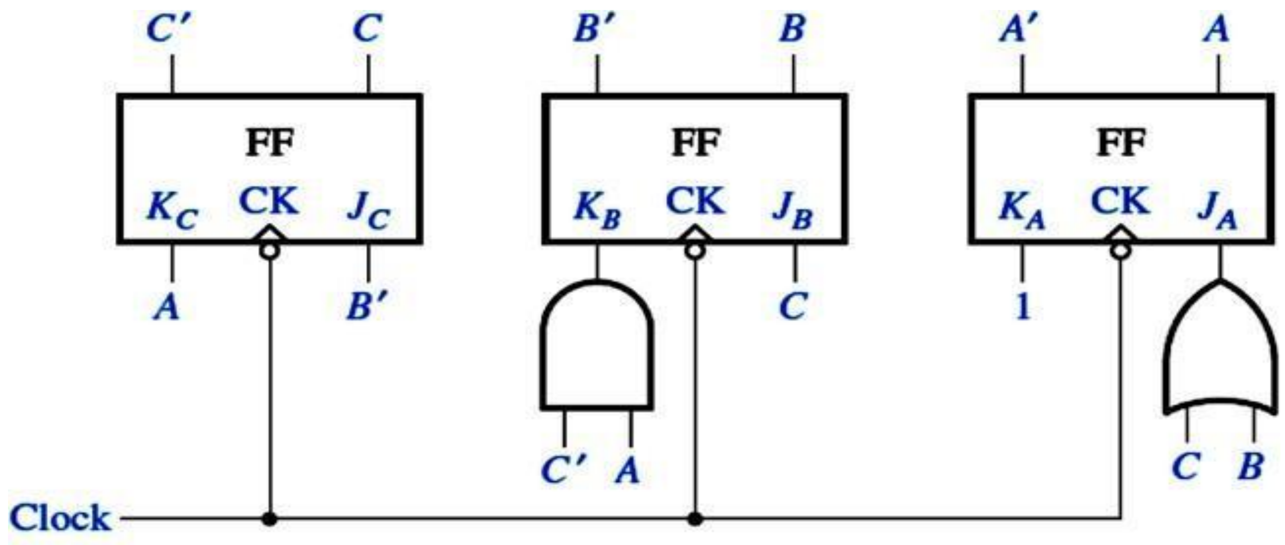
Counter Design Using J-K Flip-Flops



C	B	A	C^+	B^+	A^+	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	1	0	0	1	X	0	X	0	X
0	0	1	X	X	X	X	X	X	X	X	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	0	0	X	X	1	X	1
1	0	0	1	1	1	X	0	1	X	1	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1



$J_C = B'$ $K_C = A$ $J_B = C$ $K_B = C'A$ $J_A = C + B$ $K_A = 1$



Logic circuit (omitting the feedback lines)

Experiment: 8

SYNCHRONOUS UP COUNTER USING JK Flip-Flop

Design and implement a mod-n (n<8) synchronous up counter using J-K Flip-Flop ICs and demonstrate its working.

Components Required :IC 74LS76, IC 74LS08, Patch chords, power chords, and Trainer kit.

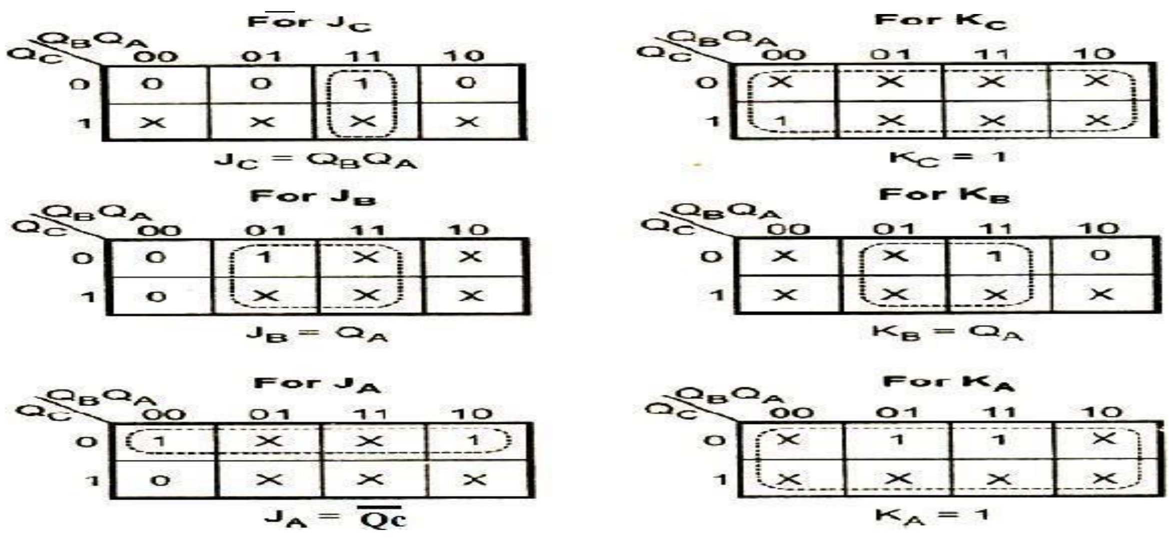
Execution Table of JK :

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Transition Table for Mod-5 Counter:

Present state			Next state			Flip Flop Inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

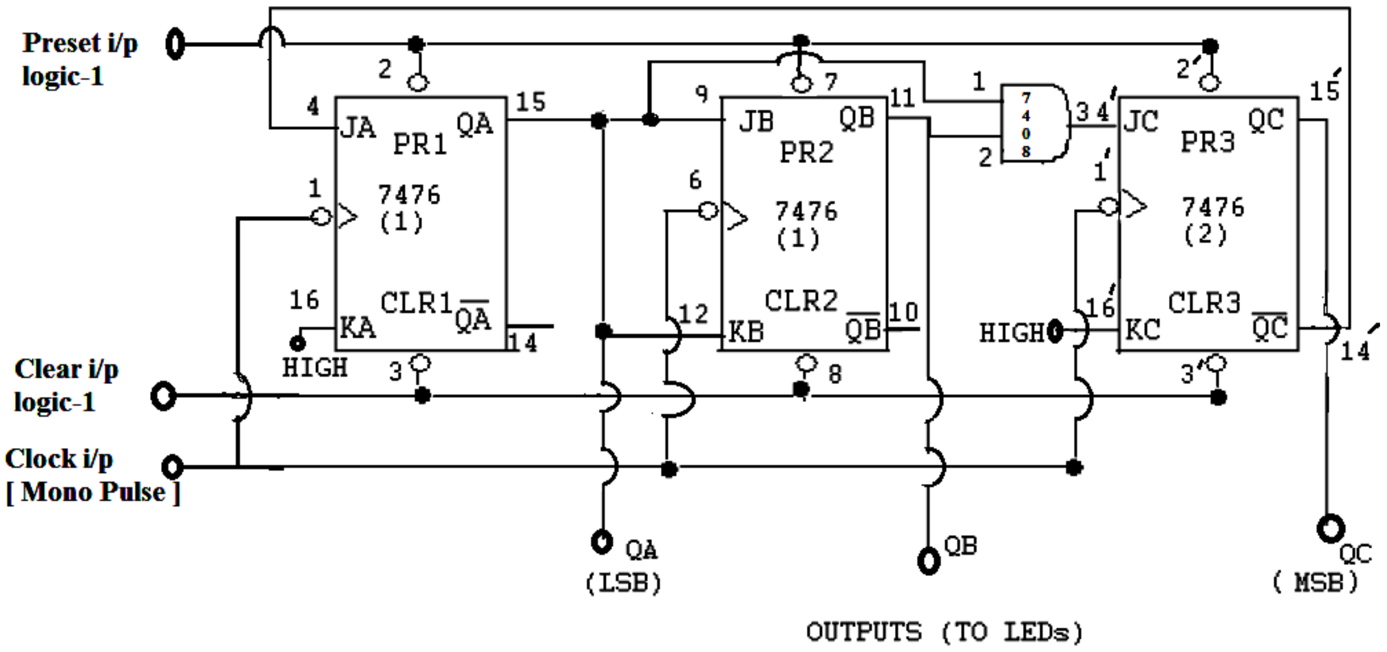
K-Maps:



Now we know the value of**J_a=** Q_c circuit diagram for Mod-5 Counter.**, J_b= Q_a, J_c=Q_aQ_b,K_a=1, K_b=Q_a, K_c=1.**

Based on these values design the

Circuit Diagram:



Theory:

- A **timer** is a specialized type of clock for measuring timeintervals.
- A **counter** is a device capable of counting number of clock pulses arriving at its clockinput.
- There are two types of counters – Synchronous Counter & AsynchronousCounter.
- In Asynchronous Counter the flip flops are connected in a such a way that o/p of first flip flop drives the clock input for the next flip flop.(All the flip flops are not clockedsimultaneously).
- In Synchronous Counter there is a no connection between o/p of first flip flop and the clock input of the next flip flop.(All the flip flops are clockedsimultaneously).
- A **flip-flop** or **latch** is a circuit that has two stable states and can be used to store stateinformation.
- A flip-flop stores a single *bit* (binary digit) of data; one of its two states represents a "one" and the other represents a "zero".

Procedure:

1) Verify all components and patch chords whether they are in good condition ornot.
2) Make connection as shown in the circuitdiagram.
3) Give supply to the trainerkit.
4) Provide input data to circuit viaswitches.
5) Verify truth table sequence and observeoutputs.

Result: Verified truth table sequence and observed the outputs.

DECADE COUNTER

Experiment: 9

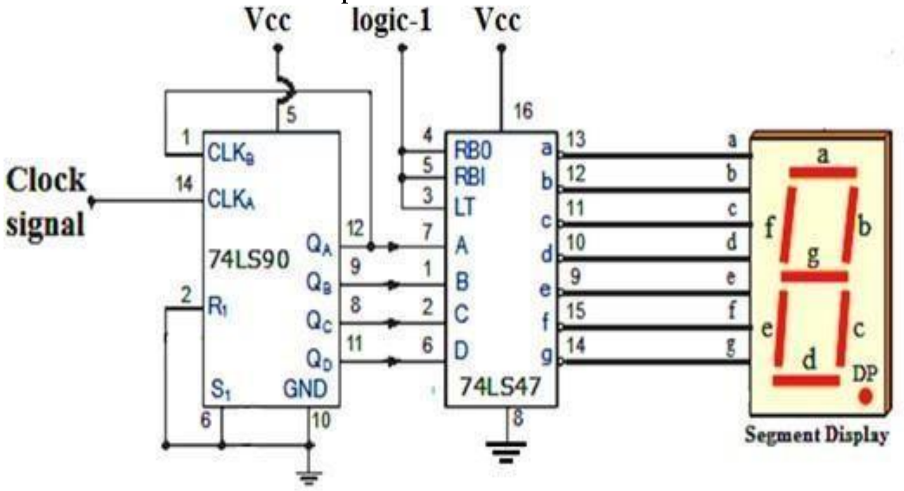
Design and implement an asynchronous counter using decade counter IC to count up from 0 to n (n<=9) and demonstrate on 7-segment display (using IC-7447).

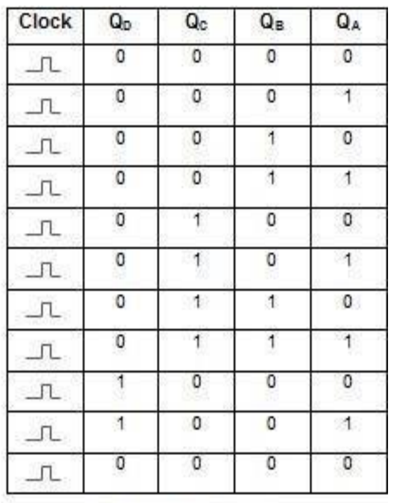
Components Required: IC 74LS90, IC-7447,Patch chords, and Trainer kit.

Circuit Diagram:

Theory:

- ❖ Asynchronous counter is a counter in which the clock signal is connected to the clock input of only first stage flip flop. The clock input of the second stage flip flop is triggered by the output of the first stage flip flop and so on. This introduces an inherent propagation delay time through a flip flop.
- ❖ A decade counter [7490] is one that counts in decimal digits, rather than binary. It counts from 0 to 9 and then resets tozero.
- ❖ To make 7490 to work in normal mode the pin numbers 2, 3, 6, and 7 should hold at Low state. QA, QB, QC, QD are 4 output pins which gives the binary value of the decimal count.
- ❖ Pin 14 is Clock input.





Clock	Q_D	Q_C	Q_B	Q_A
⎍	0	0	0	0
⎍	0	0	0	1
⎍	0	0	1	0
⎍	0	0	1	1
⎍	0	1	0	0
⎍	0	1	0	1
⎍	0	1	1	0
⎍	0	1	1	1
⎍	1	0	0	0
⎍	1	0	0	1
⎍	0	0	0	0

Truth table

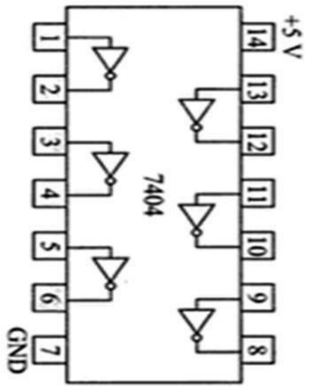
- ❖ Pin 2 and 3: Set inputs. They held to Low to activate 7490 IC as decade counter. At any instant of time, if they provide High signal then the output will hold at Low state until Pin 2 and 3 brought to Low voltage.
- ❖ Pin 6 and 7: Clear inputs. At any instant of time, if they provide High signal then the output will hold at High state until Pin 6 and 7 brought to Low voltage.
- ❖ 7447 IC decoder used to drive seven segment indicators. It converts the BCD input to the required output. It has four inputs and sevenoutputs.

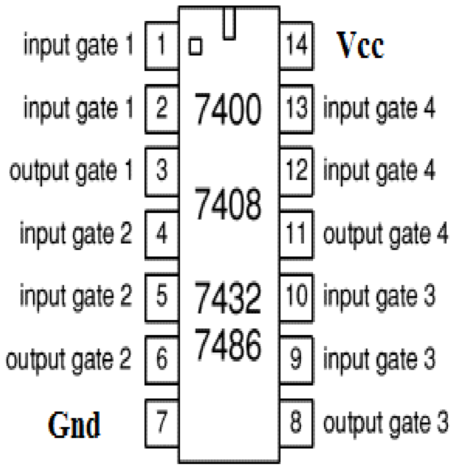
Procedure:

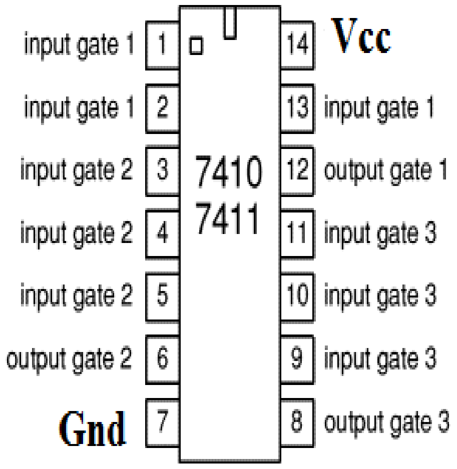
1) Verify all components and patch chords whether they are in good condition ornot.
2) Make connection as shown in the circuitdiagram.
3) Give supply to the trainerkit.
4) Provide input data to circuit viaswitches.
5) Verify truth table sequence and observeoutputs.

Result: Verified truth table sequence and observed the outputs.

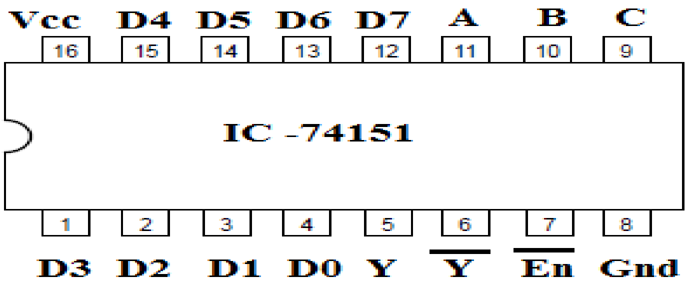
Pin Diagram of 7404, 7408, 7432, 7400, 7486, 7410, 7411:



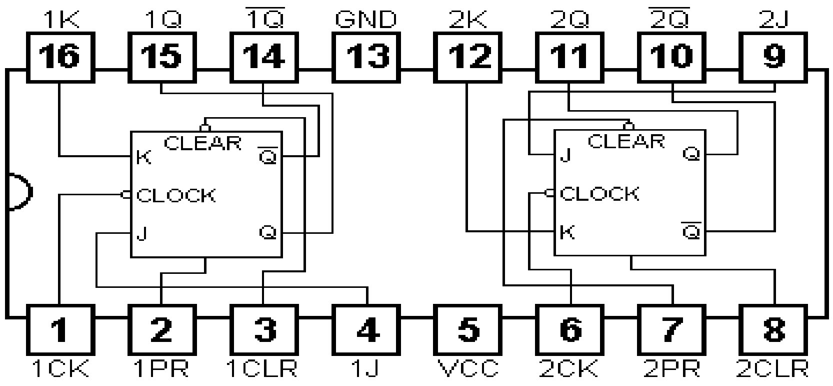




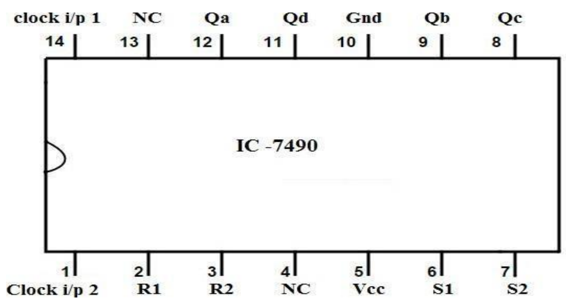
Pin Diagram of 74151:

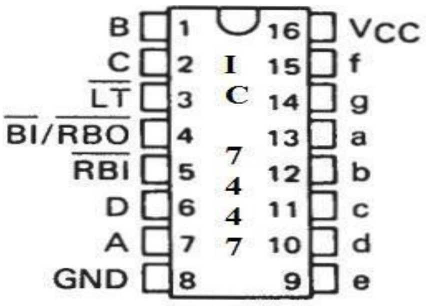


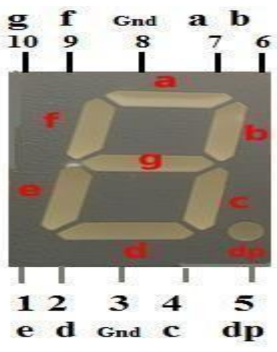
Pin diagram of 7476



Pin diagram of 7490, 7447 and LT-542





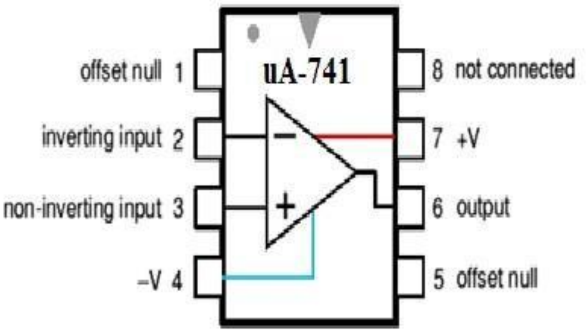


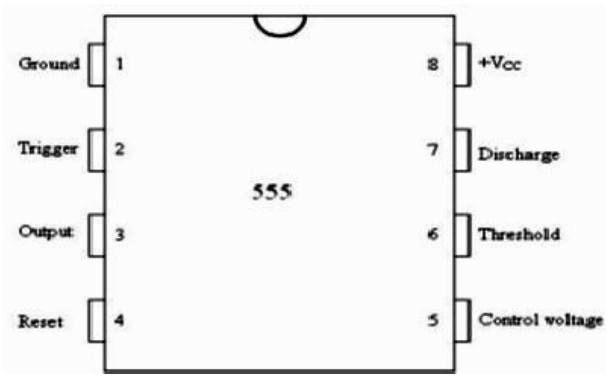
Pin diagram of µA-741 and 555-Timer:

Viva questions:

1. **What is feedback? its Type?**

Soln: Feedback: is a process where some portion of the output signal of system is fedback to the input. This is used to control the dynamic behavior of the system.In**positive feedback** (regenerative), a portion of the output signal is added to the input signal to produce a desired output. In **negative feedback** (degenerative), a portion of the output signal is subtracted from the input signal to produce a desired output.





2. **Explain about Barkhausencriteria?**

i. The first condition is that the magnitude of the loop gain (Aβ) =1 A = Amplifier gain and β = Feedbackgain.

ii. The second condition is that the phase shift around the loop must be 360° or0°.

3. What is **Operational Amplifiers (OAs)**?

Operational Amplifiers (OAs) are highly stable, high gain dc difference amplifiers. Since there is no capacitive coupling between their various amplifying stages, they can handle signals from zero frequency (dc signals) up to a few hundred kHz.

4. **What is anoscillator?**

It is a regenerative network generating an o/p without any i/p signal.

5. **What is condition required foroscillation?**

The amplifier should produce 180° phase shift to the i/p signal.The feedback network should produce an additional phase shift of 180°.So the total phase shift should be 360° and gain with feedback should be greater than **1(One)** to get sustained oscillations.

6. **Is an oscillator positive or negative feedbackamplifier?**

It is a positive feedback amplifier.

7. **Is an oscillator isconverter?**

Yes,it is a DC to AC converter.

8. **Application of anoscillator?**

Micro-computers,television,oscilloscope and clocks in digital systems.

9. **How does an oscillator operates without an input signal?**

Due to random movement of electrons inside any electronic device such as resistors, a noise voltage is generated which is a random in nature is fed back to the amplifier input.The amplifier amplifies the noise and oscillates at the tunedfrequency

10. What is anoscilloscope?

An oscilloscope, previously called an oscillograph, and informally known as a scope, CRO (for cathode ray oscilloscope), or DSO (for the more modern digital storage oscilloscope), is a type of electronic test instrument that allows observation of constantly varying signal voltages.

10. What is the type of feedback used in an op-amp Schmitttrigger?

The type of feedback used in an op-amp Schmitt trigger is positivefeedback.

11. What is a Schmitttrigger?

Schmitt trigger comparator. It converts sinusoidal input intosquare wave output.

12. What is meant by hysteresis?

The voltage difference between the upper and lower trigger point is referred to as hysteresis.

13. A**multivibrator**is an electronic ckt used to implement a variety of simple two state system such as timer, flip flops and oscillators. Multivibrator is a form of oscillator, which has a non-sinusoidal output. The output waveform isrectangular.

14. There are3-Types:

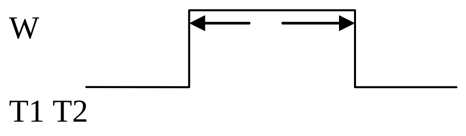
Astablein which ckt is not stable in either state,keeps on switching between 2 states by itself(low and high for a rectangular output) and it does not need any external triggering. It is also called as free running multivibrator. It is used in security alarms, pulse and tone generation.

Mono stablemultivibrator: in which one of the state is stable, but the other state is unstable. This is also called as one-shot multivibrator. Application includes missing pulse detection, frequency divider, capacitance measurement, pulse width modulation (PWM).

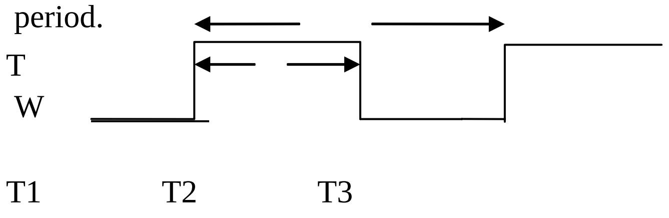
Bistable: in which the ckt is stable in either state. It can be flipped from one state to the other state by an external trigger pulse. This ckt is also called s flip-flop. It can be used to store one bit Information.

15. Frequency: is the number of cycles per unittime.

T= is the duration of one cycle in a repeating event. BW= width of the useful frequency range. PW: is defined as the time between T2 andT1.



Duty cycle: is the ratio of pulse width and period.



16. What is pinch offvoltage?

i. Drain source voltage above which the drain current become constant is known as pinch off voltage.The point N is called as pinch off point. Above the pinch off voltage the channel width becomes narrow and drain current remainsconstant.

17. What isMOSFET?

ii. The MOSFET is an abbreviation for Metal Oxide Semiconductor Field Effect Transistor. It isathree terminal semiconductor device similar to FET with gate insulated from thechannel.

PART –B

1. **Classify basic gates and universal gates**
OR, AND and NOT are basic gates NAND and NOR gates are universalgates

2. **Why NAND and NOR gates are called as universalgates**
We can realize all basic gates using NAND and NOR gates.

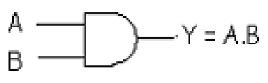
3. **What is truthtable?**
A table which lists the value of the dependent variable for each set of values of the independent variables is known as a truth table.

4. Describe Logic Design.
Logic Design: Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND and NOT are basic gates. NAND, NOR are known as universal gates. Basic gates form these gates.

- **AND GATE:** The AND gate performs a logical multiplication commonly known as AND function. The output is high when both the inputs are high. The output is low level when any one of the inputs is low.
- **OR GATE:** The OR gate performs a logical addition commonly known as OR function. The output is high when any one of the inputs is high. The output is low level when both the inputs are low.
- **NOT GATE:** The NOT gate is called an inverter. The output is high when the input is low. The output is low when the input is high.
- **NAND GATE:** The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the input is low .The output is low level when both inputs are high.
- **NOR GATE:** The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when one or both inputs are high.
- **X-OR GATE:** The output is high when any one of the inputs is high. The output is low when both the inputs are low and both the inputs are high.

AND GATE:

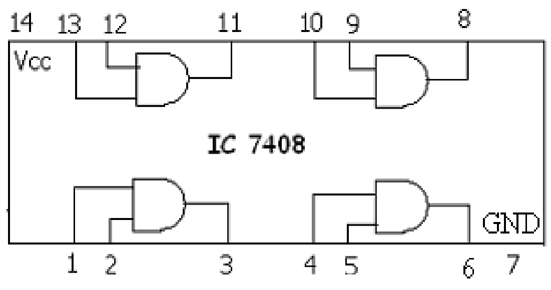
SYMBOL



TRUTHTABLE

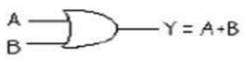
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

IC7408



OR GATE:

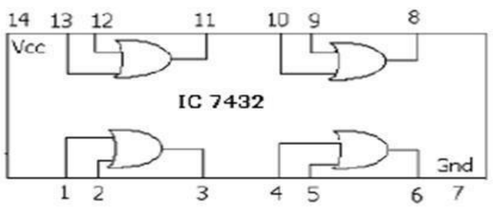
SYMBOL



TRUTH TABLE

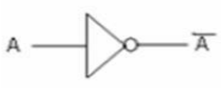
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

IC 7432



NOT GATE:

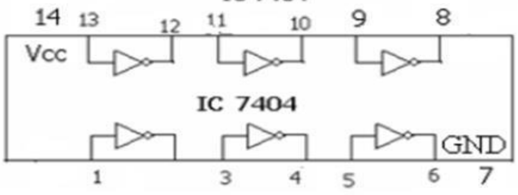
SYMBOL



TRUTH TABLE

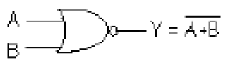
I/P (A)	O/P $(\overline{A})$
0	1
1	0

IC 7404



OR GATE:

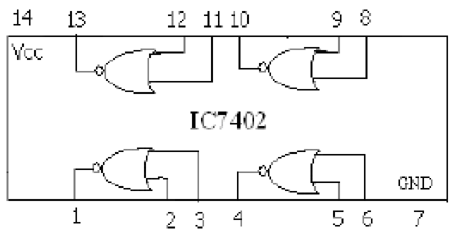
SYMBOL



TRUTH TABLE

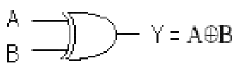
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

IC7402



X-OR GATE:

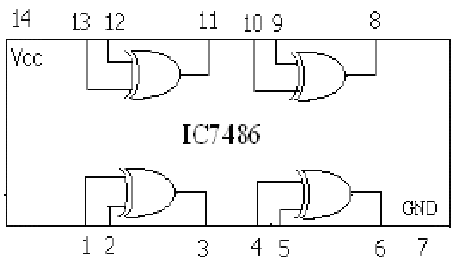
SYMBOL



TRUTH TABLE

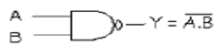
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

IC 7486



2-INPUT NAND GATE:

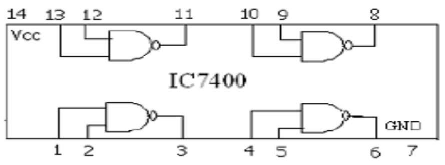
SYMBOL



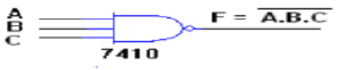
TRUTHTABLE

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

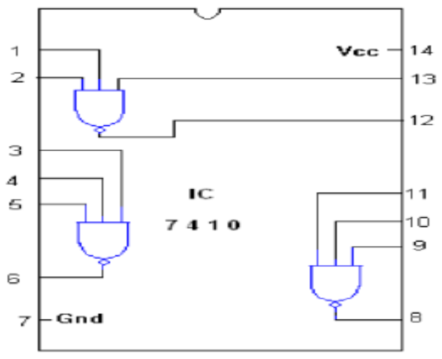
IC7408



3-INPUT NAND GATE:
SYMBOL:



PIN DIAGRAM:



TRUTH TABLE

A	B	C	$\overline{A.B.C}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

4. **Definemultiplexer?**
It is a circuit has many inputs but only one output. Out of many inputs only one is selected at a time based on the selection of control signals

5. **What isde-multiplexer?**
Which is opposite of MUX. It has fewer inputs and many outputs. Deistributes one input into many outputs.

5.What do you mean bycomparator?
A comparator is a special combinational circuit designed primarily to compare the relative magnitude of two binary numbers.

4. **Define Flipflop.**
The basic unit for storage is flip flop. A flip-flop maintains its output state either at 1 or 0 until directed by an input signal to change its state.

5. **What are the different types offlip-flop?**
There are various types of flip flops. Some of them are mentioned below theyare, RS flip-flop,SRflip-flop,D flip-flop (delayed), JK flip-flop (Jack kilby),
T flip-flop(toggle)

6. **What is the operation of Tflip-flop?**
T flip-flop is also known as Toggle flip-flop therefore for every clock pulse output changes..
• When T=0 there is no change in theoutput.
• When T=1 the output switch to the complement state (i.e. the outputtoggles).

7. **What is a master-slaveflip-flop?**
A master-slave flip-flop consists of two flip-flops where one circuit serves as a master and the other as a slave in series. Here clock input of the slave section in complement of the clock input of the master section.

8. **What is the difference between SR and JKFlip-flop?**
In SR-FF both S=R=1 condition is not allowed, where as in JK-FF both J=K=1 condition is allowed.

9. Explain different types oftriggering.

(i) Leveltriggering
(ii) Positive edge or leading edgetriggering
(iii) Negative edge or training edgetrigger

10. Give the comparison between combinational circuits and sequentialcircuits.

Sequential circuits	**Combinational circuits**
It should have memory	No memory element required
It should have at least one feedback path	No, feedback path required
It shoes the property of sequence	No sequence
e.g. flip-flops, counters shift registers etc.	E.g. Adders, subtractor code converters, parity multiplexers/de-mux etc.

11. What do you mean by presentstate?
The information stored in the memory elements at any given time defines the present state of the sequential circuit.

12. What do you mean by nextstate?
The present state and the external inputs determine the outputs and the next state of the sequential circuit.

13. What is acounter?
Counters are nothing but array of T-flip-flop and counts incoming clock pulses or sequences

14. Types of counters

i) Synchronous counters: in which all the flip-flops are simultaneouslyclocked.
ii) Asynchronous Counters: in which clock is applied to one flip-flop (LSB FF) and output of that is used as clock to the next flip-flop. They are also called ripple counters. They are slow but simple circuiting.

15. Which counter isfast?
Synchronous counters are faster than asynchronous counter (Bur they need complex circuits)

16. Application of counters?

i)Eventcounting
i) Frequencydivider
ii) Time periodmeasurement

16. What is the main difference between Ring counter and JohnsonCounter?
Ring counter is a basic register with direct feedback such that the contents of the registers simply circulate around the register when the clock is running.
Johnson counter is a basic register with inverse feedback is called shift counter or a Johnson counter or a twisted tail ring counters.

17. Defineregisters.
A register is a group of flip-flops flip-flop can store one bit information. So an n-bitregister has a group of n flip-flops and is capable of storing any binary information/number containing n-bits.

18. Define shiftregisters.
The binary information in a register can be moved from stage to stage within the register or into or out of the register upon application of clock pulses. This type of bit movement or shifting is essential for certain arithmetic and logic operations used in microprocessors. This gives rise to group of registers called shift registers.

19. How shift registers areconstructed?
Shift register are nothing but array of D-flip-flops and are used to store the data temporarily and to manipulate data.

20. What are the different types of shifttype? There are five types. Theyare,

1. Serial in Serial out ShiftRegister
2. Serial in Parallel out ShiftRegister
3. Parallel in Serial out ShiftRegister
4. Parallel in Parallel out ShiftRegister