MODULE – 1

ANALOG ELECTRONIC CIRCUITS

OPTOELECTRONIC DEVICES

Optoelectronic Devices is the field that deals with study of devices that emit, detect and control light in the wavelength spectrum ranging from ultraviolet to far infrared. They include electrical-to-optical (convert electrical energy into light energy) and optical-to-electrical (convert light energy into electrical energy) transducers. Optocouplers also come in this broad category.

PHOTODIODES:

Photodiode is a light detector semiconductor device that converts light energy into electric current or voltage which depends upon the mode of operation.

The upper cut-off wavelength of a photodiode is given by; $\lambda_c = \frac{1240}{E_g}$

where, λ_c is the cut-off wavelength in nm and E_g is the bandgap energy in eV.

A normal p-n junction diode allows a small amount of electric current, under reverse bias, due to minority charge carriers. To increase the electric current under reverse bias condition, we need to generate more minority carriers. The external reverse voltage applied to the p-n junction diode will supply energy to the minority carriers, but it will not increase the population of minority charge carriers.

A small number of minority carriers are generated due to external reverse bias voltage. The minority carriers generated at n-side or p-side will recombine in the same material, before they cross the junction. As a result, no electric current flows due to these charge carriers. For example, the minority carriers generated in the p-type material experience a repulsive force from the external voltage and try to move towards n-side. However, before crossing the junction, the free electrons recombine with the holes within the same material. As a result, no electric current flows.

To overcome this problem, we need to apply external energy directly to the depletion region to generate more charge carriers. A special type of diode called photodiode is designed to generate more number of charge carriers in depletion region. In photodiodes, we use light or photons as the external energy to generate charge carriers in depletion region.

Construction:

The typical construction of a photodiode is illustrated in the following Figure. This example uses a construction technique called *ion implantation* where the surface of a layer of N-type is bombarded with P- type silicon ions to produce a P-type layer of about 1 µm (micrometre) thick. During the formation of the diode, excess electrons move from N-type towards P-type and excess holes move from P-type towards

N-type; this process is called *diffusion*, resulting in the removal of free charge carriers close to the PN-junction, so creating a depletion layer as shown in the following Figure.

The (light facing) top of the diode is protected by a layer of Silicon Dioxide (SiO_2) in which there is a window for light to shine on the semiconductor. This window is coated with a thin anti-reflective layer of Silicon Nitride (SiN) to allow maximum absorption of light and an anode connection of aluminium (AI) is provided to the P-type layer. Beneath the N-type layer, there is a more heavily doped N+ layer to provide a low resistance connection to the cathode.

Working Principle:

When the conventional diode is reverse biased, the depletion region starts expanding and the current starts flowing due to minority charge carriers. With the increase of reverse voltage, the reverse current also starts increasing. The same condition can be obtained in photodiode without applying reverse voltage.

The following Figure shows photo diode bias symbol. The junction of Photodiode is illuminated by the light source; the photons strike the junction surface. The photons impart their energy in the form of light to the junction. Due to which electrons from valence band get the energy to jump into the conduction band. This leaves positively charged holes in the valence band, so producing 'electron-hole pairs' in the depletion layer. Some electron-hole pairs are also produced in P and N layers, but apart from those produced in the diffusion region N layers, most will be re-absorbed within the P and N materials as heat. The electrons in the depletion layer are then swept towards the positive potential on the cathode, and the holes swept towards the negative potential on the anode, so creating a photo current. In this way, the photodiode converts light energy into electrical energy.

V-I Characteristics of Photodiode:

The characteristics curve of the photodiode can be understood with the help of the following Figure. The characteristics are shown in the negative region because the photodiode can be operated in reverse biased mode only.

The reverse saturation current in the photodiode is denoted by I_0, It varies linearly with the intensity of photons striking the diode surface. The current under large reverse bias is the summation of reverse saturation current and short circuit current.

$$I = I_{sc} + I_0 \left(1 - e^{V/\Delta V t}\right)$$

Where I_{sc} is the short circuit current, V is positive for forward voltage and negative for reverse bias, V_t is volt equivalent for temperature, Δ is unity for germanium and, 2 for silicon.

Applications:

- Photodiodes are used in consumer electronics devices like smoke detectors, compact disc players, and televisions and remote controls in VCRs.
- In other consumer devices like clock radios, camera light meters, and street lights, photoconductors are more frequently used rather than photodiodes.
- Photodiodes are frequently used for exact measurement of the intensity of light in science and industry. Generally, they have an enhanced, more linear response than photoconductors.

LIGHT EMITTING DIODE (LED):

The LED is a PN-junction diode which emits light when an electric current passes through it in the forward direction. A P-N junction can convert absorbed light energy into a proportional electric current. The same process is reversed here (i.e. the P-N junction emits light when electrical energy is applied to it). This phenomenon is generally called *Electroluminescence*.

Electroluminescence is the properly of the material to convert electrical energy into light energy and later it radiates this light energy. Different sizes of light emitting diodes are available in market form $1mm^2$ to onward.

Construction:

The semiconductor material used in LED is *Galliurn Arsenide (GaAs), Gallium phosphide (GaP)* or *Gallium Arsenide Phosphide (GaAsP).* Any of the above-mentioned compounds can be used for the construction of LED, but the color of radiated light changes with the change in material (for example, GaP material gives green/red color with forward voltage of 2.2V).

The semiconductor layer of *P-type* is placed above *N-type* because the charge carrier recombination occurs in *P-type*. Besides, it is the surface of the device, and thus, the light emitted can be easily seen on the surface. If *P-type* is placed below the *N-type*, the emitted light cannot be seen. The following Figure shows cross sectional view of diffused LED.

The *P-type* layer is formed from diffusion of semiconductor material. On the other side, in *N-type* region, the epitaxial layer is grown on *N-type* substrate. The metal film is used on the *P-type* layer to provide anode connection to the diode. Similarly, Gold-film layer is coated on *N-type* to provide cathode connection. The Gold-film layer on N-type also provides reflection from the bottom surface of the diode. If any significant part of radiated light tends to hit bottom surface then that will be reflected from the bottom surface to the device top surface. This increases LED's efficiency.

Working Principle:

The charge carriers recombine in a forward-biased P-N junction as the electrons cross from the N-region and recombine with the holes existing in the P-region. Free electrons are in the conduction band of energy levels, while holes are in the valence energy band. Thus the energy level of the holes is less than the energy levels of the electrons. Some portion of the energy must be dissipated to recombine the electrons and the holes. This energy is emitted in the form of heat and light.

The working of the LED depends on the quantum theory. The quantum theory states that, when the energy of electrons decreases from the higher level to lower level, it emits energy in the form of photons. The energy of the photons is equal to the gap between the higher and lower level, as shown in the following Figure.

The LED is forward biased, which allows the current to flow in the forward direction. The flow of current is because of the movement of electrons in the opposite direction. The recombination shows that the electrons move from the conduction band to valence band and they emit electromagnetic energy in the form of photons. The energy of photons is equal to the gap between the valence and the conduction band. Color of light can be determined by the band gap of semiconductor material.

Applications:

- LEDs are used in remote control systems such TV or LCD remote.
- Used in traffic signals for controlling the traffic crowds in cites.
- Used in digital computers for displaying the computer data.
- Used in electronic calculators for showing the digital data.
- Used in digital watches and automotive heat lamps.

PHOTOCOUPLER:

Photocoupler or Optocoupler is a device that transfers electrical signals between two isolated circuits by using light.

Photocouplers are used in many functions: they can be used to link data across two circuits; they can be used within optical encoders, where the optocoupler provides a means of detecting visible edge transitions on an encoder wheel to detect position, etc., and they can be used in many other circuits where optical links and transitions are needed. As a result, optical couplers or photocouplers are found in many circuits.

Construction:

All optocouplers consist of two elements: a light source (a LED) and a photosensor (a photoresistor, photodiode, phototransistor, silicon-controlled rectifier (SCR), or triac); which are separated by a dielectric (non-conducting) barrier.

Working Principle:

When input current is applied to the LED, it switches ON and emits infrared light; the photosensor then detects this light and allows current to flow through the output side of the circuit; conversely, when the LED is off, no current will flow through the photosensor. By this method, the two flowing currents are electrically isolated. It consists of LED and photodiode; where the circuits are isolated electrically. In the following Figure, LED is forward biased, photodiode is reverse biased and output exists across R_2.

The Figure (a) describes the basic operation of an optocoupler. When current is not being applied via Pin 1, the LED is off, and the circuit connected to Pins 4 and 5 is experiencing no current flow. When power is applied to the input circuit, the LED switches on, the sensor detects the light, closes the switch and initiates current flow in the output circuit, as shown in the Figure (b).

Applications:

- Input and output switching in electronically noisy environments.
- Controlling transistors and triacs.
- Switch-mode power supplies.
- PC/ Modem communication.
- Signal isolation.
- Power control.

BJT BIASING

A *transistor* (*Bipolar Junction Transistor-BJT*) is a sandwich of one type of semiconductor (P-type or N-type) between two layers of other type. Transistors are of two types: *p-n-p transistor* and *n-p-n transistor*. There are three distinct regions (hence, terminals) in a transistor: *Emitter*, *Base*, and *Collector*.

For any circuit, four-terminals would be required: two-input-terminals and two-output-terminals. Hence, for a transistor, one of the three terminals will be common to both input and output in a circuit. Thus, there are three different modes of operation for a transistor: *Common-Base Connection (CB)*, *Common Emitter Connection (CE)*, and *Common Collector Connection (CC)*.

Transistor Biasing is the establishment of suitable dc-values such as I_C, V_{CE}, I_B, etc., by using a dc-source. When BJT is properly biased, amplification of signal takes place. There are mainly three types of biasing a transistor: *Base bias* or *Fixed bias*, *Collector-to-Base bias*, *Voltage-divider bias*.

BASE BIAS or FIXED BIAS:

Base biasing configuration is given in the following Figure. A base resistance R_B is used between V_{CC} and base to establish the base current I_B. Since V_{CC} and R_B are fixed quantities, I_B remains fixed.

Base bias circuit using a npn transistor

Base bias circuit using a pnp transistor

Applying Kirchoff's Voltage Law (KVL) to the base circuit;

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

Or, $I_B = (V_{CC} - V_{BE})/R_B$ ---------- *(1)*

V_{BE} is 0.7 V for Silicon and 0.3 V for Germanium transistor.

Applying the KVL to the collector circuit;

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

Or, $V_{CE} = V_{CC} - I_C R_C$ ---------- *(2)*

Note that, the voltage and current polarities are reversed in base bias circuits of npn- and pnp-transistors; but, the same KVL equations are applicable.

Example 1:

The base bias circuit is shown in Figure for the values indicated calculate I_B, I_C and V_{CE}.

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

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

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

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

Example 2:

Calculate the maximum and minimum levels of I_C and V_{CE} for the base bias circuit in Figure (a), when $h_{fe(\min)} = 50$ and $h_{fe(\max)} = 200$.

(a): Conditions for $h_{fe(\min)}$

(b): Conditions for $h_{fe(\max)}$

Given: $h_{fe(\min)} = 50$, $h_{fe(\max)} = 200$,

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

Case i:

$$h_{fe(\max)} = 200$$

$$I_C = h_{fe} I_B = 200 \times 36.8\,\mu\text{A} = 7.36\text{ mA}$$

$$V_{CE} = V_{CE} - I_C R_C = 18\text{ V} - (7.36\text{ mA} \times 2.2\text{ k}\Omega) = 1.8\text{ V}$$

Case ii:

$$h_{fe(\min)} = 50$$

$$I_C = h_{fe} I_B = 50 \times 36.8\ \mu\text{A} = 1.84\text{ mA}$$

$$V_{CE} = V_{CE} - I_C R_C = 18 - (1.84\text{ mA} \times 2.2\text{ k}\Omega) = 13.95\text{ V}$$

Two Q points in Figure (c) for a Transistor of a given type number always have a wide range of h_{fe} values (the h_{fe} spread). So $h_{fe(\max)}$ and $h_{fe(\min)}$ should always be used for practical circuit analysis. The base bias circuit is rarely employed because of the uncertainty of the Q point. More predictable results can be obtained with other types of bias circuit.

(c): The transistor h_{fe} value has a major effect on the Q point for a base bias circuit.

COLLECTOR-TO-BASE BIAS CIRCUIT:

The collector-to-base bias circuit shown in the following Figure, has the base resistor R_B connected between the transistor collector and base terminals. This circuit has significantly improved bias stability for h_{fe} changes compared to base bias.

Applying KVL to the outer loop;

$V_{CC} - (I_C + I_B) R_C - V_{CE} = 0$ Or, $V_{CE} = V_{CC} - (I_C + I_B) R_C$ ---------- *(3a)*

Applying KVL to the loop V_{CE}, I_BR_B, and V_{BE};

$V_{CE} - V_{BE} - I_BR_B = 0$ Or, $V_{CE} = V_{BE} + I_BR_B$ ---------- *(3b)*

Equating equations 3a and 3b;

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

Or, $(I_C + I_B) R_C + I_BR_B = V_{CC} - V_{BE}$ i.e., $I_B (R_C + R_B) + I_CR_C = V_{CC} - V_{BE}$

Substituting $I_C = \beta I_B$ in above equation, we get;

$I_B (R_C + R_B) + \beta I_BR_C = V_{CC} - V_{BE}$ Gives, $I_B = (V_{CC} - V_{BE})/(\beta+1)R_C + R_B$ ---------- *(4)*

Effect of I_C or β: Any changes in V_{CE} changes I_B, the change in I_B causes I_C to change. If I_C increases above the design level, there is an increased voltage drop across R_C; resulting in reduction of V_{CE}. The reduced V_{CE} causes I_B to be lower. If I_B decreases, I_C also decreases, as $I_C = \beta I_B$. Similarly, a reduction in I_C produces an increase in V_{CE}, which increase I_B, thus tending to increase I_C back to the original level. Thus an increase/ decrease in I_C produces a feedback effect that tends to return I_C toward its original level.

In the collector-to-base bias circuit, the feedback from the collector-to-base reduces the effect of β (due to transistor replacement). Thus collector-to-base bias has greater stability than base bias, for a given range of β values.

VOLTAGE DIVIDER (EMITTER CURRENT) BIAS CIRCUIT:

Voltage divider bias is the most stable of the three basic transistor biasing circuits. A voltage divider circuit is shown in the following Figure.

There is an emitter resistor R_E connected in series with Emitter terminal, so that the total dc load in series with the transistor is ($R_C + R_E$). Resistors R_1 and R_2 constitutes a voltage V_B.

Applying KVL to the loop V_{CC}, R_1, and R_2, we get;

$V_{CC} - I_1R_1 - I_2R_2 = 0$ Or, $I_1R_1 + I_2R_2 = V_{CC}$ ---------- *(5)*

We have; $I_1 = I_2 + I_B$

Voltage divider bias circuits are normally designed to have a voltage divider current I_2 very much greater than transistor base current I_B. i.e., $I_2 >> I_B$. Hence, $I_1 \approx I_2$ ---------- *(6)*

Using 6 in 5; $I_2R_1 + I_2R_2 = V_{CC}$ *i.e.,* $I_2 (R_1 + R_2) = V_{CC}$ Or, $\boldsymbol{I_2 = (V_{CC}) / (R_1 + R_2)}$

V_B is the voltage across R_2. i.e., $V_B = I_2R_2$ Or, $\boldsymbol{V_B = (V_{CC} * R_2) / (R_1 + R_2)}$

V_E is the voltage across R_E. i.e., $\boldsymbol{V_E = I_ER_E}$

Applying KVL to the base-emitter loop; $V_B - V_{BE} - V_E = 0$ i.e., $\mathbf{V_{BE} = V_B - V_E}$

Or, $V_E = V_B - V_{BE}$ i.e., $I_ER_E = V_B - V_{BE}$ Hence, $\boldsymbol{I_E = (V_B - V_{BE}) / R_E}$

Applying KVL to the collector-emitter loop; $V_{CC} - I_CR_C - V_{CE} - I_CR_E = 0$ $[I_E \approx I_C]$

i.e., $\boldsymbol{V_{CE} = V_{CC} - I_C (R_C + R_E)}$

MULTIVIBRATORS USING IC-555

A multivibrator (like an oscillator) is a circuit with regenerative feedback, which produces a pulsed output. There are three basic types of multivibrator circuits:

- *Astable* – has no stable states, but switches continuously between two states. This action produces a train of square wave pulses at a fixed frequency.
- *Monostable* – one of the states is stable, but the other state is unstable (transient).
- *Bistable* – the circuit is stable in either state.

Timer IC-555:

Timer IC-555 is the one of the most commonly used general-purpose linear integrated circuits.

Internal Schematic of Timer IC-555

The Timer IC 555 comprises two Op-Amp comparators, a flip-flop, a discharge transistor, a reset transistor, three identical resistors and an output stage. The resistors set the reference voltage levels at the non-inverting input of the lower comparator and inverting input of the upper comparator at $+V_{CC}/3$ and $2V_{CC}/3$, respectively. The output of two comparator feed SET and RESET inputs of the Flip-Flop. This decided the logic state of its output and subsequently the final output. The Flip-Flops complementary outputs feed the output stage and the base of the discharge transistor. Hence, when the output is HIGH, the discharge transistor is OFF and when the output is LOW, the discharge transistor is ON.

MONOSTABLE MULTIVIBRATOR:

Monostable multivibrator using 555 timer IC is as shown in the following Figure. This 555 timer is called monostable multivibrator because it has only one stable state. Resistor R_A and capacitor C are components connected externally to the IC-555. Threshold voltage (6) and discharge (7) pins are connected to each other.

- Initially the SR-FF is set ($Q = 1$); transistor T is driven into saturation, and the capacitor is by passed by the transistor. Therefore, the capacitor voltage $V_C = 0$, and also the output voltage $Vout = 0$ (as $Q = 1$).

- When the negative going trigger pulse (which should be more than *1/3* V_{CC}) is applied at trigger input of lower comparator, the comparator output goes high; and SR-FF is reset ($Q = 0$), forcing $\bar{Q}$ to go high and transistor *T* turns off.
- As *T* is off, capacitor starts charging through R_A. Now, the output will remain high (from t_1 to t_2, in waveform shown).
- At time t_2; the voltage across the capacitor V_{CC} becomes more than *2/3* V_{CC} and upper comparator output goes high. This will set the SR-FF ($Q = 1$).
- Since, SR-FF output Q = 1; transistor T is ON, and hence, capacitor discharges, and also output goes low. The output remains low till the next trigger pulse is applied.

From the waveform of monostable multivibrator, it is clear that, the ON time T_{ON} of the output voltage is same as charging time of the capacitor.

Therefore, $T_{ON} \rightarrow$ is the time taken by capacitor to charge from 0 to *2/3* V_{CC}.

The voltage across capacitor increases exponentially and is given by; $V_C = V_{CC}\,[1 - e^{-\frac{t}{RC}}]$

As capacitor charges through R_A; let us replace *R* by R_A: Hence, $V_C = V_{CC}\,[1 - e^{-\frac{t}{R_A C}}]$

At $t = t_2$ (T_{ON}), the capacitor voltage (V_C) reaches *2/3* V_{CC}:

Therefore, $V_C = \frac{2}{3}V_{CC} = V_{CC}\,[1 - e^{-\frac{T_{ON}}{R_A C}}]$ Or, $e^{-\frac{T_{ON}}{R_A C}} = 1 - 2/3 = 1/3$

Therefore, $\boldsymbol{T_{ON} = 1.1\, R_A C}$

Applications:

A monostable multivibrator can be used in many applications, few important applications are

1. Frequency divider
2. Missing pulse detector
3. Pulse width modulator
4. Pulse position modulator etc.

Example 1:

For a monostable muitivibrator time delay $T = 100$ ms and resistance $R = 105$ kΩ. Calculate the capacitor value.

Given: $T = 100$ ms, $R = 105$ kΩ

We knew that the pulse width W is given by

$$W = 1.1RC = T_{on}\,(\text{time delay})$$

$$C = \frac{T_{on}}{1.1R} = \frac{100\text{ ms}}{1.1\times 105\text{ k}}$$

$$\boxed{C = 0.865\,\mu\text{F}}$$

Example 2:

Design a monostable multivibrator circuit using 555 timer to produce an output pulse of 20sec width.

The output pulse [T_{on}] is given as

$$T_{on} = 20 \text{ sec}$$

$$\text{WKT } T_{on} = 1.1RC \text{ Assuming } C = 150\ \mu\text{F}$$

$$20 = 1.1\ R \times 150\ \mu\text{F}$$

$$R = \frac{20}{1.1\times150\times10^{-6}} = 121.21\ \text{k}\Omega$$

$$\boxed{R = 121.21\ \text{k}\Omega}$$

Example 3:

Find the resistive element value to generate T = 10 ms time delay, using 555 timer as a monostable multivibrator Assume C = 0.47 μF.

The on period of pulse is given by

$$T_{on} = 1.1\text{RC}$$

$$10 \times 10^{-3} = 1.1 \times \text{R} \times 0.47 \times 10^{-6}$$

$$R = \frac{10\times10^{-3}}{1.1\times0.47\times10^{-6}} = 19.34\ \text{k}\Omega$$

$$\boxed{R = 19.34\ \text{k}\Omega}$$ We can choose standard value as 18 kΩ.

Example 4:

Design a timer that can turn on a heater immediately after pressing the button, and it should hold the heater in on-state for 10 seconds.

The relay coil should be energized for 10 seconds to hold heater on so T_{on} is 10 seconds and choosing C = 47 μF.

$$\text{WKT } T_{on} = 1.1RC$$

$$10 = 1.1RC$$

$$\boxed{R = \frac{10}{1.1\times47\times10^{-6}} = 193.42\ \text{k}\Omega}$$

Monostable multivibrator used to turn–on relay

ASTABLE MULTIVIBRATOR:

An *astable multivibrator* does not have any stable state; it keeps changing its state from low to high and high to low. This multivibrator is also called *free running multivibrator* or *rectangular wave generator* circuit. *Astable multivibrator* does not require an external trigger pulse to change the state of the output. The circuit configuration of an astable rnultivibrator is as shown in the following Figure.

To understand the operation, let us divide the circuit operation into two time interval T_{ON} and T_{OFF}.

ON time operation:

- At $t = 0$, the voltage on the capacitor $V_{CC} = 0$, the same capacitor voltage is applied to both trigger point of lower comparator and threshold point of upper comparator. As capacitor voltage $V_C = 0$, which is less $1/3\ V_{CC}$, the output of lower comparator goes high ($\bar{Q} = 1$) and $Q = 0$. This

causes T to go off and capacitor starts charging through series resistors R_A and R_B. When capacitor voltages reaches $2/3\ V_{CC}$, on time is terminated.

OFF time operation:

- As soon as V_C exceeds $2/3\ V_{CC}$, the upper comparator output goes high and it will set the SR FF. i.e., $S = 1$ and $R = 0$ and $Q = 1$ and $\bar{Q} = 0$. This will turn on transistor T, and output at pin (3) goes low.
- Now, the capacitor discharges through R_B, and through transistor T. The discharge time (also called off time (T_D); and it depends on the values of R_B and C. When capacitor voltage is $V_C = 1/3\ V_{CC}$, lower comparator output goes high.
- This process of charging and discharging is continuous and hence circuit oscillates. The schematic diagram and waveforms are as shown in the Above Figure.

The output voltage waveform is the sum of charging and discharging periods (T_C, and T_D) of the capacitor. ∴ Period of one cycle $T = T_C + T_D$.

Frequency can be written as $f = \frac{1}{T} = \frac{1}{T_C + T_D}$

Voltage across charging capacitor is given by (initial voltage on capacitor is zero);

$$V_C = V_{CC}\,[1 - e^{-\frac{t}{RC}}]$$

If there is some initial voltage present, the voltage expression for capacitor changes, and is given by;

$$V_C = V_F + (V_i - V_F)e^{-\frac{t}{RC}}$$

Where V_F - final voltage capacitor can reach, and V_i - is Initial voltage on capacitor.

During charging time T_C, the initial voltage on the capacitor is $V_i = 1/3\ V_{CC}$, and the final voltage is $V_F = V_{CC}$. Also as charging takes place through both R_A and R_B, the above expression becomes;

$$\frac{2V_C}{3} = V_{CC} + (\frac{1}{3}V_{CC} - V_{CC})e^{-\frac{T_C}{(R_A+R_B)C}}$$

$$\frac{1}{3}V_{CC} = (\frac{2}{3}V_{CC})e^{-\frac{T_C}{(R_A+R_B)C}}$$

Therefore, $\boldsymbol{T_{ON} = T_C = 0.693\ (R_A + R_B)\ C}$

From discharging waveform; we can note that, the capacitor discharges from $2/3\ V_{CC}$ to $1/3\ V_{CC}$; and can be expressed as:

$$\frac{1}{3}V_{CC} = 0 + (\frac{2}{3}V_{CC} - 0)e^{-\frac{T_D}{R_B C}}$$

Or, $$\frac{1}{3}V_{CC} = (\frac{2}{3}V_{CC})e^{-\frac{T_D}{R_B C}}$$

Therefore, $\boldsymbol{T_{OFF} = T_D = 0.693\ R_B\ C}$

Hence, Total period, $T = T_{ON} + T_{OFF} = 0.693\,(R_A + 2R_B)\,C$

Frequency, $f = \frac{1}{T} = \frac{1.44}{(R_A+2R_B)C}$

Duty Cycle, %D = [(on time) / (total time)]*100

$$\%\,D = \frac{(R_A + R_B)}{(R_A + 2R_B)} * 100$$

Applications:

A nastable multivibrator can be used in many applications, few important applications are

1. Square-wave oscillator/ generator
2. Schmitt trigger using IC-555
3. Voltage controlled oscillator.

Example 1:

For an astable circuit $R_1 = 22\ k\Omega$, $R_2 = 30\ k\Omega$ and $C = 0.5\ \mu F$ Find on and off period of the output wave form shown in Figure 7-17.

FIGURE 7-17

The on period corresponds to charging period of a capacitor and charging takes place through resistors R_1 and R_2.

$$\therefore T_C = T_{on} = 0.693(R_1 + R_2)C$$

$$= 0.693(22+30)\times10^3\times0.5\times10^{-6}$$

$$\boxed{T_{on} = 18\ \text{msec}}$$

The off period corresponds to discharging time (T_D) of the capacitor and discharging taken place through R_2 only.

$$\therefore T_D = T_{off} = 0.693R_2C$$

$$= 0.693\times30\times10^3\times0.5\times10^{-6}$$

$$\boxed{T_{off} = 10.395\ \text{msec}}$$

Example 2:

A 555 timer is configured to operate in astable mode with $R_A = 5\ k\Omega$, $R_B = 5\ k\Omega$ and $C = 0.01\ \mu F$, Determine the frequency of the output and duty cycle.

Given: $R_A = 5\ k\Omega$, $R_B = 5\ k\Omega$, $C = 0.01\ \mu F$

Frequency of oscillation for astable multivibrator is given by

$$f = \frac{1}{T} = \frac{1.45}{(R_A + 2R_B)C} = \frac{1.45}{(5\,k + 2(5\,k))0.01\,\mu F}$$

$$= \frac{1.45}{5\times10^3 + 2(5\times10^3)\times0.01\times10^{-6}}$$

$$\boxed{f = 10.357\ kHz}$$

and Duty cycle $D = \dfrac{R_A + R_B}{R_A + 2R_B} = \dfrac{5\times10^3 + 5\times10^3}{5\times10^3 + 2(5\times10^3)}$

$$D = \frac{10\times10^3}{15\times10^3} = 66.66\%$$

$$\boxed{D = 66.66\%}$$

Example 3:

Design an astable multivibrator using 555 timer for a frequency of 2 kHz and a duty cycle of 75% Assume $C_1 = 0.1\ \mu F$.

Given: $f = 2$ KHz, duty cycle $= 75\%$, $C = 0.1\ \mu F$

The on period $T_{on} = 0.693\ (R_A + R_B)\ C$.

The off period $T_{off} = 0.693 R_B\ C$

Total period $T = T_{on} + T_{off} = 0.693(R_A + R_B)\ C$

$\therefore$ Duty cycle D is given by

$$D = \frac{T_{on}}{T_{on} + T_{off}} = \frac{T_{on}}{T} = \frac{\cancel{0.693}\,(R_A + R_B)\cancel{C}}{\cancel{0.693}\,(R_A + 2R_B)\cancel{C}}$$

$$D = \frac{R_A + R_B}{R_A + 2R_B} = \frac{3}{4} = 0.75$$

$\therefore\ 3\ (R_A + 2\ R_B) = 4\ (R_A + R_B)$

$3R_A + 6R_B = 4\ R_A + 4R_A + 4R_B$

$2R_B = R_A$ Or, $R_B = \frac{1}{2}R_A$

Substituting for R_B in expression for T

$$\therefore T = 0.693(R_A + R_B)C$$

$$T = 0.693\left(R_A + \frac{1}{2}R_A\right)\times0.1\times10^{-6}$$

$$T = 0.693\left(\frac{3}{2}R_A\right)\times0.1\times10^{-6}$$

$$\boxed{f = \frac{1}{T}}$$

$$1\times10^3 = f = \frac{1}{T} = \frac{1}{0.693(1.5R_A)\times0.1\times10^{-6}}$$

$$R_A = \frac{1}{0.693(1.5)0.1\times10^{-6}\times10^3}$$

$$\boxed{R_A = 9.6\,k\Omega}$$

$$R_B = \frac{1}{2}R_A$$

$$= \frac{1}{2}\times9.6\times10^3$$

$$\boxed{R_B = 4.8\,k\Omega}$$

OPERATIONAL AMPLIFIER (OP-AMP) APPLIATION CIRCUITS

An **Op-Amp** is a direct-coupled high gain, high bandwidth differential amplifier with very high value of input impedance and very low value of output impedance.

Basic Differential Amplifier & Circuit Representation of an OP-Amp

The ideal Op-Amp model was derived to simplify circuit calculations. The ideal Op-Amp model makes three assumptions:

1. Input resistance (impedance), $R_i = \infty$
2. Output resistance (impedance), $R_o = 0$
3. Open-loop (differential voltage) gain, $A_d = \infty$

Based on these three assumptions,

other assumptions can be derived:

1. Since $R_i = \infty$, $I_i = I_{ni} = 0$
2. Since $R_o = 0$, $V_o = A_d * V_d$
3. Zero DC input and output offset voltages
4. Bandwidth and slew rate are also infinite, as no freq
5. Drift is also zero, as there is no changes in performance over time, temperature, power supply variations, and so on
6. Since output voltage depends only on differential input voltage, it rejects any voltage common to both inputs. Hence, common mode gain = 0

Ideal Op-Amp	Practical Op-Amp
Internal Impedance is infinite	Input Impedance range 100KΩ to 1000MΩ
Output Impendence is zero	Output impedance range from 10Ω to 100Ω
Open loop differential voltage gain is infinite	Open loop gain is in the range of 10,000 to 100,000
Bandwidth is infinite	Bandwidth is limited
DC input and output offset voltage is zero	Finite DC input and output offset voltage
Input differential voltage is zero	Finite differential voltage is finite

Open-loop gain is the differential voltage gain in the absence of any positive or negative feedback. Practical Op-Amps have –

1. Input impedance can vary from hundred of kilo-ohms (for some low-grade Op-Amps) to tera-ohms (for high grade Op-Amps).
2. Output impedance may be in the range of 10 to 100 Ω
3. Open-loop gain in the range of 10,000 to 1, 00,000
4. Bandwidth is limited and is specified by gain-bandwidth product
5. There may be some finite DC output (referred to as output offset voltage), even when both the inputs are grounded.

PEAK DETECTORS:

Peak detector detects holds the most positive value attained by the input signal. The following Figure shows peak detector circuit.

During positive half cycle of the input, D-conducts and capacitor charges to peak (highest) value of the input. Capacitor retains its charged value unless and until it discharges with a help of switch.

The op-amp is connected as a voltage follower and its output voltage will be equal the drop across capacitor which is positive peak value of the applied voltage and will remain that for long periods until next more higher peak occurs at the input. For negative cycle of input, the diode is reverse biased and capacitor retains its value.

Modified Peak Detector:

More sophisticated peak detector that buffers the signal source from the capacitor is shown in the following Figure.

As Op-Amp (A_1) is connected as voltage follower, the circuit presents very high impedance to the signal source. Op-Amp (A_2) acts as a buffer between the capacitor and the load. Output (V_0) at any given time is equal to the voltage on the capacitor which is nothing but, the peak value of the input occurred up to that time.

Whenever the input signal has higher peak than the present one, the capacitor charges up to new high input level. Whenever input level gets dropped, then capacitor retains the peak value of input, as

diode D_1 gets reverse biased and diode D_2 prevents amplifier A_1 output from going into negative saturation.

To hold the negative peak of the input signal, reverse the diode connections in the above Figure.

Applications:

- Used for AM in communication
- Used in test and measurement instrumentation applications.

SCHMITT TRIGGER (REGENERATIVE) COMPARATOR:

A Schmitt trigger is a fast-operating voltage level detector.

Inverting Schmitt Trigger:

The input voltage V_{in} is applied to the inverting input terminal and the feedback voltage goes to the non-inverting terminal. This means, the circuit uses positive voltage feedback (i.e., feedback voltage aids the input voltage).

If the input voltage at the inverting terminal is slightly positive than feedback voltage at the non-inverting terminal, the output voltage will be negative (negative saturation, $-V_{sat}$); and if the input voltage more negative than the reference feedback voltage, the output will be positive (positive saturation, $+V_{sat}$).

Hence, the voltage at the output switches from $+V_{sat}$ to $-V_{sat}$ or vice-versa; are called *Upper Trigger Point (UTP)* and *Lower Trigger Point (LTP)*. The difference between two trigger points is called *Hysteresis*.

The upper and lower trigger points can be written as;

$$UTP = \frac{R_2}{(R_1+R_2)} e\, V_{sat} \qquad LTP = \frac{R_2}{(R_1+R_2)} (-V_{sat})$$

$$V_{hys} = UTP - LTP = \frac{R_2}{(R_1+R_2)} e\, V_{sat} - \frac{R_2}{(R_1+R_2)} e\,(-V_{sat}) = 2\left(\frac{R2}{R_1+R_2}\right) V_{sat} = 2\beta V_{sat}$$

$$\beta = \frac{R2}{R_1+R_2}$$

Non-Inverting Schmitt Trigger:

The input voltage V_{in} is applied to the non-inverting input terminal and the feedback voltage also goes to the non-inverting terminal. The inverting terminal is grounded..

Initially, assume that the output is in the negative saturation ($-V_{sat}$). Then the feedback voltage is also negative. This feedback voltage will hold the output in negative saturation, until the input voltage becomes positive enough to make voltage positive.

Let V_A is the voltage at point A. Hence, $V_A = IR_2$.

Since no current passes through the Op-Amp, entire current flows through R_2.

$$\text{Therefore, } I = \frac{V_0}{R_1} = \frac{+V_{sat}}{R_1}$$

When V_{in} becomes positive and its magnitude becomes greater than $(R_2/R_1)V_{sat}$, then the output switches to $+V_{sat}$. Therefore, the UTP at which the output switches to $+V_{sat}$ is given by;

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

Similarly, when V_{in} becomes negative and its magnitude becomes greater than $(R_2/R_1)V_{sat}$, then the output switches to $-V_{sat}$. Therefore, the LTP at which the output switches to $-V_{sat}$ is given by;

$$UTP = -\frac{R_2 V_{sat}}{R_1}$$

$$V_{hys} = UTP - LTP = 2\left(\frac{R2}{R_1}\right)V_{sat} = 2\beta V_{sat}$$

$$\beta = \frac{R2}{R_1}$$

Applications of Schmitt Trigger:

Schmitt trigger is used in many applications, where level needs to be sensed. Hysteresis is used to reduce the multiple transitions that can occur around.

- Digital to analog conversion
- Level detection
- Line reception.

Example 2:

Design a Schmitt trigger whose threshold voltages are ± 5 V. Draw its wave forms.

Choosing op-amp with $V_{sat} = \pm 13.5$ V with supply ± 15 V.

$$V_{UTP} = +5\text{ V}$$

$$\text{Now } V_{UTP} = \frac{R_2}{R_1+R_2}.V_{sat} \quad \text{i.e., } 5 = \frac{R_2}{R_1+R_2}\times 13.5$$

$$\therefore R_1+R_2 = 2.7\,R_2 \quad \text{i.e. } R_1 = 1.7\,R_2$$

$$\text{Choose } R_2 = 10\text{ k}\Omega \quad \therefore R_1 = 17\text{k}\Omega\ (\text{Use } 18\text{ k}\Omega)$$

The designed circuit with waveform are shown below.

Example 3:

For the circuit shown, R_2 = 120 Ω and R_1 = 51 kΩ Determine the threshold voltages, if power supply applied to the op-amps are +15 V and −15 V.

Given

$$V_{sat}^{+} = +V_{CC} = +15\text{ V}.$$

$$V_{sat}^{-} = -V_{EE} = -15\text{ V}.$$

$$R_1 = 51\text{ k}\Omega,\ R_2 = 120\ \Omega$$

WKT

$$V_{UTP} = \frac{V_{sat}^{+}\,R_2}{R_1+R_2} = \frac{15\times 120}{51\times 10^3+120} = 35.2\text{ mV}.$$

$$V_{LTP} = \frac{V_{sat}^{-}\,R_2}{R_1+R_2} = \frac{-15\times 120}{51\times 10^3+120} = -35.2\text{ mV}$$

Example 4:

For the circuit shown in Figure below find the value of R_1 and R_2 if supply voltages are +12 and – 12 V. Assume hysteresis with = 6 V.

Given: $V_{sat}^{+} = 12\text{ V}$; $V_{sat}^{-} = -12\text{ V}$; and $V_H = 6\text{ V}$

Hysteresis is given by

$$V_H = \frac{R_2 V_{sat}^{+}}{R_1 + R_2} - \frac{R_2 V_{sat}^{-}}{R_1 + R_2} = \frac{R_2}{R_1 + R_2}\left[V_{sat}^{+} - V_{sat}^{-}\right]$$

$$\therefore \frac{R_2}{R_1 + R_2} = \frac{V_H}{\left[V_{sat}^{+} - V_{sat}^{-}\right]}$$

$$\therefore \frac{R_2}{R_1 + R_2} = \frac{6}{12 - [-12]} \qquad \therefore \frac{R_2}{R_1 + R_2} = 0.25$$

$$\therefore R_2 = 0.25\,R_1 + 0.25\,R_2 \qquad \therefore 0.75\,R_2 = 0.25\,R_1$$

$$\frac{R_2}{R_1} = \frac{0.25}{0.75} \qquad \text{Assuming } \boxed{\begin{aligned} R_2 &= 10\text{ k}\Omega \\ R_1 &= 30\text{ k}\Omega \end{aligned}}$$

Example 5:

Design Schmitt trigger circuit with UTP = 4 and LTP = 2 V [both positive] Assume V_{sat} = 14 V.

Given: $V_{sat} = 14\text{ V}$, UTP = 4 and LTP = 2 V.

WKT

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

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

$$V_H = V_{UTP} - V_{LTP} = 2.V_{sat}.\frac{R_2}{R_1 + R_2}$$

$$= 4 - 2 = 2\times 14\times\frac{R_2}{R_1 + R_2}$$

$$2 = 28\frac{R_2}{R_1+R_2} \qquad 28R_2 = 2R_1+2R_2 \qquad 26R_2 = 2R_1 \qquad \text{i.e., } \boxed{R_1 = 13R_2}$$

Let $\boxed{R_2 = 10\,\text{k}\Omega}$ $\qquad R_1 = 13\times 10\,\text{k} = 130\,\text{k}\Omega$

Example 6:

Design schmitt trigger circuit with UTP = – 2 and LTP = – 4 V. V_{sat} = 14 V.

We have, $\text{UTP} - \text{LTP} = \dfrac{2R_2}{R_1+R_2}V_{sat}$

$$-2+4 = 2 = \frac{2R_2V_{sat}}{R_1+R_2} = \frac{2R_2}{R_1+R_2}\times 14$$

$$2 = \frac{28R_2}{R_1+R_2} \Rightarrow R_1 = 13R_2$$

Let $R_2 = 10\,\text{k}$ $\qquad \therefore R_1 = 130\,\text{k}\Omega\,(120\,\text{k std})$

Also WKT

$$\text{UTP} + \text{LTP} = \frac{2R_1.V_{Ref}}{R_1+R_2}$$

$$V_{Ref} = \frac{(R_1+R_2)(\text{UTP}+\text{LTP})}{2R_1} \qquad V_{Ref} = 3.27\text{ V.}$$

Example 7:

For the schmitt trigger R_1 = 3 kΩ and R_2 = 1 kΩ calculate the V_{UTP}, V_{LTP} and V_H. Assume saturation voltages as ± 12 V.

Given: $V_{sat}^{+} = 12\text{ V}$; $V_{sat}^{-} = -12\text{ V}$; $R_1 = 2\,\text{k}\Omega$; $R_2 = 3\,\text{k}\Omega$

WKT

$$V_{UTP} = \frac{(V_{sat}^{+})R_1}{R_2} = \frac{(12)\times 1\,\text{k}}{3\,\text{k}} = 4\text{ V}$$

$$V_{LTP} = \frac{-(V_{sat}^{-})R_1}{R_2} = \frac{-(12)\times 1\,\text{k}}{3\,\text{k}} = -4\text{ V}$$

The hysteresis width is given by

$$V_H = \frac{R_1}{R_2}\left[V_{sat}^{+} - V_{sat}^{-}\right] = \frac{1\,\text{k}}{3\,\text{k}}\left[12-(-12)\right] = \frac{24}{3} = 8\text{ V.}$$

ACTIVE FILTERS:

Filter is a frequency selective circuit commonly used in signal processing that passes signal of specified range of frequencies and blocks the signals of frequencies outside the band. Active filters are attractive due to their –

- Flexibility in gain control
- Small component size
- No loading Problem
- Pass band gain
- Use of the inductors can be avoided

Filters are useful in many areas of applications, such as Communication and Signal Processing. They are found in electronic systems like Radio, Television, Telephones, Radars, satellites, and Biomedical instruments.

Broader Classification of Filters

Passive filters work for high frequencies; but at audio frequencies, the inductors become problematic, as they are large, heavy, high power dissipation, and expensive.

Active filters use Op-Amp as the active element, resistors and capacitors as passive elements.

SNo.	Passive Filters	Active Filters
1	Filters with only components like resistors, capacitors, and inductors are known as passive filters.	Filters with components such as Op-Amps, transistors, and other active elements are known as active filters.
2	Passive filters do not require an external power source for operation; incapable of providing power gain.	Active filters require an external power supply for operation; capable of providing power gain.
3	Better stability and can withstand large currents.	Oscillations and noise will be generated due to feedback loops.
4	A passive filter has no frequency	Due to active elements, active filters have

	limitations.	frequency limitations.
5	Passive filters circuits are bulky/ heavy due to the presence of inductors; they consume more power and operate with limited speed.	Active filters circuits are more compact, less heavy; and operate with high speed.
6	Difficult to fabricate in IC form and usually designed using discrete components.	Can be fabricated in IC form and usually designed using discrete components.

Active filters offer the following advantages over Passive filters:

- Gain and frequency adjustment flexibility
- No loading problem & No insertion loss
- Size and weight
- Cost.

Most commonly used active filters are –

- Low-pass filter, High-pass filter, Band-pass filter, Band-stop filter (Band-reject filter), and All-pass filter.

Frequency Response:

Design:

An active filter generally uses Op-Amp. Op-Amp has very high input impedance and low output impedance. The gain is determined by the resistive network in the feedback loop.

First Oder Active Low-Pass Filter (LPF):

The first-order low-pass butter worth filter consists of a single RC filter stage, providing a low frequency path to the non-inverting input of an Op-Amp. The circuit diagram and the frequency response of the circuit is given below:

- From the graph; the gain (A_F) is almost constant for the frequency range: $0 < f < f_H$.
- At cut-off frequency, $f = f_H$, the gain is $0.707A_F$.
- After cut-off frequency f_H, the gain decreases at the rate of 20 dB/decade.
- The cut-off frequency is given by: $f_H = \frac{1}{2\pi RC}$
- Pass band gain is given by: $A_F = 1 + \frac{R_f}{R_1}$

Some applications of low-pass filters are –

- Low-pass filters are used in Audio amplifiers
- LPFs are used in equalizers or speakers to reduce the high frequency noise.

Example 1:

Design a first order low pass filter with cut-off frequency of 2.2 kHz and with pass band gain of 2.

Given: cut-off frequency $f_H = 2.2$ KHz.

Let us choose $C = 0.01$ μF.

WKT $$f_H = \frac{1}{2\pi RC}$$

$$\Rightarrow \quad R = \frac{1}{2\pi f_H C} = \frac{1}{2\pi \times 2.2\times10^3 \times 0.01\times10^{-6}}$$

$$R = 7.233\,k\Omega$$

For frequency scaling, use the potentiometer of value calculated below.

W.K.T

$$A_F = 1 + \frac{R_f}{R_1} = 2$$

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

Therefore $R_f = R_1 = 10$ kΩ(Take).

The low pass filter circuit with designed values is as follows:

Example 2:

Determine the value of the resistance required for the low pass filter with cut-off frequency 30 k rad/sec and capacitor value = 0.001 μF.

Given $\omega_H = 30$k rad/sec and $C = 0.001$ μF.

WKT $\omega_H = 2\pi f_H$

and $$f_H = \frac{1}{2\pi RC}$$

$$\therefore \quad \omega_H = \cancel{2\pi} \times \frac{1}{\cancel{2\pi} R \times 0.001\times10^{-6}} = 30\times10^3$$

$$R = \frac{1}{30\times10^3 \times 0.001\times10^{-6}}$$

$$= \frac{1}{30\times\cancel{10^3}\times\cancel{10^{-3}}\times10^{-6}} = 0.33\times10^{+5}$$

$$\boxed{R = 33\ \text{k}\Omega}$$

Second Order Low-Pass Filter:

First order filter can be converted to second order filter by adding an extra RC-network, as shown in the following Figure. The frequency response of second order low-pass filter is same as the first order low-pass filter except that the gain at the stop band rolls off at the rate of 40 dB/decade.

- After cut-off frequency f_H, the gain decreases at the rate of 40 dB/decade.
- The cut-off frequency is given by: $f_H = \frac{1}{2\pi\sqrt{R_2R_3C_2C_3}}$.
  - If $R_2 = R_3 = R$ & $C_2 = C_3 = C$; then $f_H = \frac{1}{2\pi RC}$
- Pass band gain is given by: $A_F = 1 + \frac{R_f}{R_1}$

Example 1:

Determine the values of Resistance required for second order low pass Butter worth filter having cut-off frequency as 15 k rad/sec with capacitor value as 0.01 μF.

Given: $\omega_H = 15$ k rad/sec, $C = 0.01$ F.

From equation (7.38)

$$f_H = \frac{1}{2\pi\sqrt{R_2R_3C_2C_3}}$$

But $R_2 = R_3 = R$ and $C_2 = C_3 = C$ and C as 0.01 μF

$$f_H = \frac{1}{2\pi\sqrt{(R)^2\times(0.01\times10^{-6})^2}} = \frac{1}{2\pi\times R\times0.01\times10^{-6}} \quad (*)$$

and $\omega_H = 2\pi f_H = 15$ k rad/sec $= 2\pi \times f_H = 15 \times 10^3$ $\quad f_H = \frac{15\times10^3}{2\pi} = 2.3870$ kHz

on substituting f_H in equation(*), it becomes

$$2.3870\times10^3 = \frac{1}{2\pi\times R\times0.01\times10^{-6}} \qquad R = 6.67\ \text{k}\Omega \qquad R_2 = R_3 = 6.67\ \text{k}\Omega$$

Example 2:

Design a second order low pass filter with cut-off frequency of 10 kHz and unity gain at low frequency. Also calculate the voltage transfer function magnitude at 15 kHz for the filter.

Given: Cut-off frequency $f_H = \frac{1}{2\pi RC}$

$$\therefore \qquad RC = \frac{1}{2\pi f_H} = \frac{1}{2\pi\times10\times10^3} = 15.92\times10^{-6}$$

Let $R = 100$ kΩ, then $\quad C = \frac{15.92\times10^{-6}}{100\times10^3} = 0.159$ pF

Therefore $C_2 = 0.159$ pF and $C_3 = 0.159$ pF

The voltage transfer function is given by

$$\frac{V_o(s)}{V_{in}(s)} = \frac{A_F}{\sqrt{1+\left(\frac{f}{f_H}\right)^4}} = \frac{1}{\sqrt{1+\left(\frac{15\times\cancel{10^3}}{10\times\cancel{10^3}}\right)}} = 0.406$$

Example 3:

Design a second order low pass butter worth filter for the cut-off frequency $f_H = 200$ Hz and draw the circuit diagram.

Given : $f_H = 200$ Hz.

Choose $R_2 = R_3 = R$ and $C_2 = C_3 = C = 0.1\ \mu F$

Now, let us find value of R

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

$$200 = \frac{1}{2\pi \times R \times 0.1 \times 10^{-6}}$$

$$\boxed{R = 7.96\ k\Omega.}$$

For second order filter

$$R_f = 0.586R,$$

i.e $$A_F = 1 + \frac{R_f}{R_1} = 1.586.$$

choosing $R_1 = 10\ k\Omega$

$$1 + \frac{R_f}{R_1} = 1.586$$

$$1 + \frac{R_f}{10\ k\Omega} = 1.586$$

$$\frac{R_f}{10\ k\Omega} = 0.586$$

$$R_f = 5.86\ k\Omega$$

For adjustment use 10 kΩ pot.

High-Pass Butter Worth Filter:

High-pass filters passes higher frequency signals, attenuating all signals below cut-off frequency, f_L.

First Order High-Pass Butter Worth Filter:

The filter circuit consists of a passive filter followed by a non-inverting amplifier.

- At low frequency: $f < f_L$, $\frac{V_0}{V_{in}} < A_F$. A_F increases at the rate of 20 dB/decade till $f = f_L$.
- At cut-off frequency, $f = f_L$, the gain is $0.707A_F$.
- At very high frequency $f > f_L$, $\frac{V_0}{V_{in}} = A_F$ is constant.
- The cut-off frequency is given by: $f_L = \frac{1}{2\pi RC}$
- Pass band gain is given by: $A_F = 1 + \frac{R_f}{R_1}$

Example 1:

Design a first order high pass filter with a cut-off frequency of 10 kHz with pass band gain of 2.

Given: f_L = 101 kHz (given)

Choose $C < 1\ \mu F$.

Let $C = 0.01\ \mu F$

Calculate $R = \dfrac{1}{2\pi fc}$

$$R = \frac{1}{2\pi \times 10 \times 10^3 \times 0.01 \times 10^{-6}}$$

$$\boxed{R = 1.5913\ k\Omega}$$

Step 4: Gain $A_F = 2 = 1 + \dfrac{R_f}{R_1}$

$$\therefore \quad \frac{R_f}{R_1} = 1 \Rightarrow R_f = R_1$$

Choosing R_1 = 10 kΩ(say).

Second Order High-Pass Filter:

A first order high-pass filter can be converted into a second order high-pass filter by using an extra RC-network in the input side. The frequency response of second order high-pass filter is same as the first order high-pass filter except that the gain at the stop band rolls off at the rate of 40 dB/decade.

- The cut-off frequency is given by: $f_L = \frac{1}{2\pi\sqrt{R_2R_3C_2C_3}}$.
  - If $R_2 = R_3 = R$ & $C_2 = C_3 = C$; then $f_L = \frac{1}{2\pi RC}$
- Pass band gain is given by: $A_F = 1 + \frac{R_f}{R_1}$

Example 1:

For the circuit shown in Figure find (a) lower cut-off frequency (b) pass band gain

Given: $R_2 = R_3 = 16$ kΩ and $C_2 = C_3 = 0.01$ μF

(a) The lower cut-off frequency for second order high pass filter is given by

$$f_L = \frac{1}{2\pi\sqrt{R_2R_3C_2C_3}} = \frac{1}{2\pi\sqrt{\left(16\times10^3\right)^2\left(0.01\times10^{-6}\right)^2}} = 1\,\text{kHz}$$

$$\boxed{f_L = 1\,\text{kHz}}$$

b) The pass band gain is

$$A_F = 1 + \frac{R_f}{R_1} = 1 + \frac{31.6\times10^3}{54\times10^3} = 1.586$$

$$\boxed{A_F = 1.586}$$

Active Band-Pass Filter:

Active band-pass filters provide an effective means of making a filter to pass only a given band of frequencies. An active band-pass filter can be constructed by cascading a single low-pass filter with a single high-pass filter, as shown below:

The cut-off frequency of the low-pass filter is higher than the cut-off frequency of the high-pass filter; and the difference between these frequencies at the –3dB point will give the '*bandwidth*' of the band-pass filter. A band-pass filter can be characterized by Quality factor (Q). The relation between Q, 3dB bandwidth, and the centre frequency, f_C, is given by; $Q = \frac{f_C}{BW} = \frac{f_C}{f_H - f_L}$

Wide Band-Pass Filter:

A low *Q*-filter will have a wide-pass-band; i.e., with $Q < 10$. It has wide flat response over the range of frequencies and bandwidth is large.

Single stage first order Bandpass filter:

Narrow Band-Pass Filter:

A high *Q*-filter will have a narrow-pass-band; i.e., with $Q > 10$. It has a sharp bell type response, with high gain and high selectivity.

- $f_C = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$.
- $Q_{BP} = \frac{f_C}{BW_{3dB}} = \frac{1}{2}\sqrt{\frac{R_1}{R_2}}$

- Maximum Gain, $A_v = -\frac{R_2}{2R_1} = -2Q^2$

Applications of Band-Pass Filters (BPF):

- BPFs are also used in optics like Lasers, LIDARS, etc.
- BPFs are extensively used in wireless transmitters and receivers.
- BPFs are used in electronic devices like Sonar, Seismology; and medical applications like ECG, and electrocardiograms.
- BPFs are extensively used for Audio signal processing, where a particular range of frequencies of sound is required while removing the rest.

Band-Stop Filter (Band-Reject Filter):

Band-Stop Filer (BSF) is another type of frequency selective circuit that functions in exactly opposite to the band-pass filter. BSF passes all frequencies with the exception of those within a specified stop band, which are attenuated. If this stop band is very narrow and highly attenuated over a few hertz, then the band-stop filter is referred as a *notch filter*.

Wide Band-Reject Filter:

The frequency response curve and the circuit diagram of a wide band-reject filter is given below.

Example 2:

Design a basic wide-band, RC band stop filter with a lower cut-off frequency of 200Hz and a higher cut-off frequency of 800Hz. Find the geometric center frequency, –3dB bandwidth and Q of the circuit.

Given: $f_L = 200$ Hz, $f_H = 800$ Hz

We have

$$f = \frac{1}{2\pi RC} Hz$$

The upper and lower cut-off frequency points for a band stop filter can be found using the same formula as that for both the low and high pass filters.

Assuming a capacitor, C value for both filter sections of 0.1 uF, the values of the two frequency determining resistors, R_L and R_H are calculated as follows.

High Pass Filter Section

$$f_L = \frac{1}{2\pi R_L C} = 200\,\text{Hz} \quad \text{and} \quad C = 0.1\,\mu\text{F}$$

$$\therefore R_L = \frac{1}{2\pi \times 200 \times 0.1 \times 10^{-6}} = 7958\,\Omega \text{ or } 8\,\text{k}\Omega$$

Low Pass Filter Section

$$f_H = \frac{1}{2\pi R_H C} = 800\,\text{Hz} \quad \text{and} \quad C = 0.1\,\mu\text{F}$$

$$\therefore R_H = \frac{1}{2\pi \times 800 \times 0.1 \times 10^{-6}} = 1990\,\Omega \text{ or } 2\,\text{k}\Omega$$

From this, we can calculate the geometric center frequency, f_C as:

$$f_C = \sqrt{f_L \times f_H} = \sqrt{200 \times 800} = 400 \text{ Hz}$$

$$\text{BW} = f_H - f_L = 800 - 200 = 600 \text{ Hz}$$

$$Q = \frac{f_C}{\text{BW}} = \frac{400}{600} = 0.67 \text{ or } -3.5 \text{ dB}$$

Narrow Band-Stop Filter (Notch Filter):

Notch filters are highly selective, high-*Q* form of band-stop filter, which can be used to reject a single or very small band of frequencies. The most common notch filter design is the twin-T notch filter network (shown below).

$$R_1 = R_2 = 2(R_3)$$

$$C_2 = C_3 = (0.5)C_1$$

$$\therefore f_N = f_{\text{notch}} = \frac{1}{4\pi R_3 C_3}$$

A twin-T network offers very high reactance at the resonance frequency and very low reactance at frequency off-resonance. In the circuit diagram, very low frequency signals find their way to the output via low-pass filter (formed by $R_1 - R_2 - C_3$); and very high frequency signals find their way to the output via high-pass filter (formed by $C_1 - C_2 - R_3$). Hence, in an intermediate band of frequencies, both filters pass signals to the output; due to cancellation of +ve phase shift of high-pass filter with the –ve phase shift of the low-pass filter.

Example 1:

Design a two op-amp narrow-band, RC notch filter with a center notch frequency, f_N of 1 kHz and a –3dB bandwidth of 100 Hz. Use 0.1 μF capacitors in your design and calculate the expected notch depth in decibels.

Given: f_N = 1000 Hz, BW = 100 Hz and C = 0.1 uF.

1. Calculate value of R for the given capacitance of 0.1 uF

$$R = \frac{1}{4\pi f_N C} = \frac{1}{4\pi \times 1000 \times 0.1 \times 10^{-6}}$$

$$\therefore \quad R = 795\ \Omega \text{ or } 800\ \Omega$$

2. Calculate value of Q

$$Q = \frac{f_N}{BW} = \frac{1000}{100} = 10$$

3. Calculate value of feedback fraction k

$$K = 1 - \frac{1}{4Q} = 1 - \frac{1}{4 \times 10} = 0.975$$

4. Calculate the values of resistors R_3 and R_4

$$K = 0.975 = \frac{R_4}{R_3 + R_4}$$

Assume R_4 = 10 kΩ, then R_3 equals:

$R_3 = R_4 - 975\ R_4 = 10000 - 0.975 \times 10000,$

$$\therefore \quad R_3 = 250\ \Omega$$

5. Calculate expected notch depth in decibels, (dB)

$$\frac{1}{Q} = \frac{1}{10} = 0.1$$

$$f_{N(\text{d B})} = 20 \log (0.1) = -20 \text{ dB}$$

NON LINEAR AMPLIFIER:

Non linearity is the behavior of a circuit, particularly an amplifier, in which the output signal strength do not vary in direct proportion to the input signal strength. A non-linear amplifier in as circuit which gives non linear relationship between its input and output signals.

The Non linear amplification can be achieved in a simple way by just connecting a non-linear device such as PN-junction diode in the feedback path. In the circuit shown is the following Figure, large change in input voltage causes small change in the output voltage. This circuit is a log amplifier, hence the output voltage is logarithm of the input voltage.

The above Figure shows a non-linear amplifier, where diode '*D*' is used in negative feedback path. By virtual ground concept; as node *B* is grounded, node *A* will be virtually grounded. Therefore, $V_A = 0$.

We have; $I = \frac{V_{in} - V_A}{R} = \frac{V_{in}}{R}$ since $V_A = 0$.

Let I_f be the current through the diode. The voltage across diode is $V_A - V_0$. Since, $V_A = 0$, the voltage across diode is $-V_0$.

Diode equation: $$-V_0 = \eta V_T \ln\left[\frac{I_f}{I_r}\right]$$

Where, V_T – Voltage equivalent of Temperature

I_f – Diode forward current

I_r – Diode reverse saturation current.

Since, current through the Op-Amp is negligible; $I = I_f$

Therefore, $I_f = I = \frac{V_{in}}{R}$

Gives, $V_0 = \eta V_T \ln\left[\frac{V_{in}}{I_r R}\right] = \eta V_T \ln\left[\frac{V_{in}}{V_{ref}}\right]$ $I_r R = V_{ref}$ is a constant.

The above equation shows that, the output voltage is a logarithmic function of input voltage.

Applications: Non-linear Amplifiers are used in AC bridge balance detectors.

RELAXATION OSCILLATOR:

Relaxation oscillator is a non-linear electronic oscillator circuit that generates a continuous non-sinusoidal output signal in the form of rectangular wave, triangular wave or a saw-tooth wave. The time period of non-sinusoidal output depends on the charging time of the capacitor connected in the oscillator circuit.

The relaxation oscillator basically contains a feedback loop that has a switching device in the form of transistor, relays, operational amplifiers, comparators, or a tunnel diode that charges a capacitor through a resistance till it reaches a threshold level then discharges it again. The following Figure shows the basic circuit of an Op-Amp based relaxation oscillator.

Assume that, the output is initially in positive saturation. As a result, voltage at non-inverting input of Op-Amp is $+V_{SAT} * R_1 / (R_1 + R_2)$. This force the output to stay in positive saturation as the capacitor C is initially in fully discharged state. Capacitor C starts charging towards $+V_{SAT}$ through R. The moment the capacitor voltage exceeds the voltage appearing at the non-inverting input, the output switches to $-V_{SAT}$.

Now, the voltage appearing at the non-inverting input changes to $-V_{SAT} * R_1 / (R_1 + R_2)$. The capacitor starts discharging and after reaching zero, it begins to discharge towards $-V_{SAT}$. Again, as soon as it becomes more negative than the voltage appearing at the non-inverting input of the Op-Amp, the output switches back to $+V_{SAT}$.

The expression for the time period of the output rectangular waveform is given by; $T = 2\,RC \ln\left(\frac{1+\beta}{1-\beta}\right)$

In the above equation; the natural logarithm is used, which is logarithm to base *e*. By varying the value of resistor R, the time period of the output waveform can be varied.

β is the feedback fraction/ factor and is given by $\beta = R1/(R1+R2)$

When the output voltage V_0 is at at +VSAT the feedback voltage is known as upper threshold voltage V_{UTP} and is given by $+V_{SAT} \times R1/(R1+R2)$

When the output voltage V_0 is at at –VSAT the feedback voltage is known as lower threshold voltage V_{LTP} and is given by $-V_{SAT} \times R1/(R1+R2)$

VOLTAGE TO CURRENT (V TO I) CONVERTER:

In many applications, we have to convert a voltage to a proportionate current. These voltage-to-current converter circuits can be of *two types*:

a) **Voltage to Current Converter with Floating Load:** The circuit is shown in the following Figure, where R_L is the floating load.

Since, voltage at node A is V_i; $V_i = I_L R_1$ Or, $I_L = \frac{V_i}{R_1}$

i.e., input voltage V_i is converted into and output current.

b) **Voltage to Current Converter with Grounded Load:** The circuit is shown in the following Figure.

$$I_1 = \frac{V_{in} - V_2}{R} \qquad I_2 = \frac{V_0 - V_2}{R}$$

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

Therefore, $I_L = \frac{V_{in} - V_2}{R} + \frac{V_0 - V_2}{R} = \frac{V_{in} + V_0 - 2V_2}{R}$

For a non-inverting amplifier, we know that; $A_v = 1 + \frac{R}{R} = 2 \quad So, V_0 = 2V_2$

Therefore, $I_L = \frac{V_{in} + 2V_2 - 2V_2}{R} \quad Or, I_L = \frac{V_{in}}{R}$

Thus, the current I_L is proportionate to voltage.

CURRENT TO VOLTAGE (C TO V) CONVERTER:

Consider the simple Op-Amp circuit to convert *I* to *V*, as shown in the following Figure.

Since, current through the Op-Amp is negligible; $I_S = I_f$

$$I_s = I_f = \frac{V_B - V_0}{R_f}$$

By virtual ground concept; as node *A* is grounded, node *B* will be virtually grounded. Therefore, $V_B = 0$. Therefore,

$$I_s = \frac{-V_0}{R_f} \quad Or, \quad V_0 = I_S R$$

Thus, output is proportional to the input current I_S, and the circuit works as *I* to *V* converter.

VOLTAGE REGULATORS

All electronic systems that we use daily, requires a stable power supply voltage source; and voltage regulators accomplish that. *Voltage regulator* is a circuit that keeps the output voltage constant under all operating conditions. *Voltage regulation* is the process of keeping a voltage steady under conditions of changing applied voltage, changing load and temperature.

There are *two types* of voltage regulators: *shunt* and *series*.

Need for Regulators:

In ordinary power supplies, the voltage regulation is very poor. The DC output voltage changes appreciably with change in load current. The output voltage also changes due to fluctuations in the input AC supply. This is due the following reasons:

1. In practice, there are considerable fluctuations in line voltage caused by external factors. This changes the DC output voltage. Most of the electronic circuits will refuse to work satisfactorily on such output voltage fluctuations. Hence, regulated power supply is the solution.
2. The internal resistance of ordinary power supply is relatively large (> 30 Ω). Therefore, output voltage is affected by the amount of load current drawn from the supply. These variations in DC voltage may cause erratic operation of circuits. Without stable potentials, circuit performance degrades and if the variations are large enough, the components may get destroyed. In order to avoid this, regulated power supply is used.

Input to the voltage regulator is unregulated pulsating DC obtained from filter rectifier. Its output is constant DC voltage which is almost ripple free. The following Figure shows block diagram of regulated power supply.

The *transformer* provides voltage transformation and electrical isolation between the input power supply (AC mains) and the DC output. The *rectifier circuit* changes the AC voltage appearing across the transformer secondary to DC (unidirectional output). The rectifier circuit always has some AC content known as ripple. The *filter circuit* smoothens the ripple of the rectifier circuit. The *regulator* is a type of feedback circuit that ensures that the output DC voltage does not change from its nominal value due to change in line voltage or load current.

Factors Affecting the Load Voltage:

The variables affecting the load voltage in a power supply are given below:

1. ***Load current (I_L):*** Ideally the output voltage should remain constant in-spite of changes in the load current, but practically the power supply without regulator, the load voltage decreases as load current, I_L, increases. For practical power supply regulator, the load voltage must be constant through load to full load condition.
2. ***Line voltage:*** The input to the rectifier is AC (230 V) is the line voltage. This input decides the output voltage level. If input changes, output also changes. So this affects the performance of power supply. So ideally voltage must remain constant irrespective of any changes in the line voltage.
3. ***Temperature:*** In the power supply, the rectifier unit is used which uses PN-junction diode. As the diode characteristics are temperature dependent, the overall performance of the power supply is temperature dependent.

Performance Parameters of a Power Supply:

The power supply is judged by some parameters, called as *performance parameters*. These performance parameters are explained below:

1. ***Line Regulation:*** If the input to the rectifier unit i.e. 230 V changes, the output DC of rectifier will also change and since the output of rectifier is applied to the regulator, the output of regulator will also vary. Thus the source causes the change in output. This is as *source regulation* or *line regulation*. It is defined as the change in regulated DC output for a given change in input (line) voltage. Ideally the source regulation should be zero and practically it should be as low as possible.
2. ***Load Regulation:*** *Load regulation* is defined as the change in the regulated output voltage when load current is changed from zero (no load) to maximum value (full load). The load regulation ideally should be zero, but practically it should be as small as possible. The following Figure shows the load regulation characteristics. Percentage load regulation = $\left[\frac{V_{NL}-V_{FL}}{V_{FL}}\right] * 100$

3. ***Voltage Stability factor (S_V):*** *Voltage stability factor* shows the dependency of output voltage on the input line voltage. *Voltage stability factor* is defined as the percentage change in the output voltage which occurs per volt change in input voltage, where load current and temperature are assumed to be constant. Smaller the value of this factor, better is the performance of power supply.
4. ***Temperature Stability Factor (S_r):*** As in the chain of power supply we are using semiconductor devices (diodes in rectifier block) the output voltage is temperature dependent. Thus the *temperature stability* of the power supply will be determined by temperature coefficients of various temperature sensitive semiconductor devices. So, it is better to choose the low temperature coefficient devices to keep output voltage constant and independent of temperature. S_r must be as small as possible, and ideally it should it should be zero for a power supply.
5. ***Ripple Rejection Factor (RR):*** The output of rectifier and filter consists of ripples. *Ripple rejection* is defined as a factor which shows how effectively the regulator rejects the ripples and attenuates it from input to output. As ripples in the output are small compared to input, the *RR* is very small and in dB, it is in negative value. Ripple rejection factor = $\frac{V_{RIPPLE}\ (OUTPUT)}{V_{RIPPLE}\ (INPUT)}$

   When expressed in decibels, ripple rejection equals $20\log\left[\frac{V_{RIPPLE}\ (OUTPUT)}{V_{RIPPLE}\ (INPUT)}\right] dB$

   Also, $V_{RIPPLE}(output) = \frac{V_{RIPPLE}\ (INPUT)}{1+Loop\ Gain}$

Example: *Two power supplies A and B are available in the market. Power supply A has no-load and full-load voltages of 40 V and 30 V respectively; whereas these values are 30 V and 28 V for power supply B. Which one do you think is better power supply?*

<u>Supply A:</u> V_{NL} = 40 V, V_{FL} = 30 V

% Voltage regulation = $[(V_{NL} - V_{FL})/V_{FL}]*100$ = [(40 – 30)/30]*100 = 33 %.

<u>Supply B:</u> V_{NL} = 30 V, V_{FL} = 28 V

% Voltage regulation = $[(V_{NL} - V_{FL})/V_{FL}]*100$ = [(30 – 28)/28*]*100 = 7 %.

<u>Conclusion:</u> The power supply which has lower voltage regulation is better. Hence, power supply *B* is between than power supply *A*.

Example: *A regulated power supply operates from 220 $\pm$ 20 VAC. It produces a no-load regulated output voltage of 24 $\pm$ 0.5 VDC. Also, the regulated output voltage falls from 24 VDC to 23.8 VDC as the load changes from no-load to full-load condition for the nominal value of input voltage. Determine (a) line regulation and (b) load regulation.*

Line regulation = (24.5 – 23.5)/24 = 1/24 = 0.0417 = 4.17%

Load regulation = (24 – 23.8)/23.8 = 0.2/23.8 = 0.0084 = 0.84%.

Example: *A regulated power supply provides a ripple rejection of –80dB. If the ripple voltage in the unregulated input were 2V, determine the output ripple.*

Ripple rejection in dB is given by; $20 \log \left[\frac{V_{RIPPLE}\ (OUTPUT)}{V_{RIPPLE}\ (INPUT)}\right] dB = -80dB$

Or $\log \left[\frac{V_{RIPPLE}\ (OUTPUT)}{V_{RIPPLE}\ (INPUT)}\right] = -4$

Or $\left[\frac{V_{RIPPLE}\ (OUTPUT)}{V_{RIPPLE}\ (INPUT)}\right] = 10^{-4}$

Therefore, output ripple = $2 * 10^{-4}$ V = 0.2mV

Example: *The following Figure shows load voltage versus load current characteristics of a regulated power supply. Determine the output impedance of the power supply.*

Output impedance is given by ratio of change in the output voltage for known change in the load current. From the given characteristic curve, output impedance = (24 – 23.5)/((10 – 0) = 0.5/10 = 0.05Ω = 50mΩ.

Three-Terminal Regulators:

Three-terminal regulators require no external components. These are available in fixed output voltage (positive and negative) as well as adjustable output voltage (positive and negative) types with current rating 100mA, 500mA, 1.5A and 3.0A.

- LM/MC 78XX-series and LM 140XX/340XX-series are the popular three-terminal positive output voltage regulators.
- LM/MC 79XX-series and LM 120XX/320XX-series are the popular three-terminal negative output voltage regulators.
- LM 117/217/317 is common adjustable positive output voltage regulators.
- LM 137/237/337 is common adjustable negative output voltage regulators.
    - ✓ A two-digit number in place of "XX" indicates the regulated output voltage.

The minimum unregulated input to regulated output differential voltage required for the regulator to produce the intended regulated output voltage is known as *dropout voltage*. For example, consider a 5V regulator with a 2V dropout voltage; for this to give a regulated output, the input voltage must be least

equal to the output voltage (5V) plus the dropout voltage (2V), which is 7V; any input below 7V will result into unregulated voltage output.

The following Figures show the basic application circuits using LM/MC 78XX-series and Lm/MC 79XX-series three-terminal regulators.

Basic Application Circuits using Three-Terminal Regulators

Example: *Refer three-terminal regulator circuit of following Figure. Determine (a) load current; (b) current through LM 7812; (c) current through external transistor; (d) power dissipated in LM 7812. Take* V_{BE} *(*Q_1*) = 0.7V.*

(a) Load current = 12/5 = 2.4A

(b) Current through regulator = 0.7/1 = 0.7A

(c) Current through external transistor = 2.4 – 0.7 = 1.7A

(d) Voltage appearing at regulator input = 15 – 0.7 = 14.3V

(e) Power dissipated in the regulator = (14.3 – 12) * 0.7 = 1.61W.

Adjustable Voltage Regulator:

An adjustable voltage regulator is a kind of regulator, whose regulated output voltage can be varied over a range. There are positive adjustable voltage regulators and negative adjustable regulators in practice.

LM317 is a classic example of positive adjustable voltage regulator, whose output voltage can be varied over a range of 1.2 V to 57 V. LM337 is an example of negative adjustable voltage regulator. LM337 is actually a compliment of LM317 which are similar in operation and design with the only difference being polarity of regulated output voltage.

Connection of LM317 Adjustable Voltage Regulator

The resistors R_1 and R_2 determine the output voltage V_{out}. The resistor R_2 can be adjusted to get the output voltage in the range of 1.21 V to 57 V. The output voltage is given by;

$$V_{out} = V_R (1 + R_2/R_1) + I_{ADJ}R_2$$

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.

BASIC DAC TECHNIQUES:

The DAC converts digital or binary data into its equivalent analog value. The symbolic representation of an n-bit DAC is given below:

n-bit DAC

The DAC output can either be a voltage or current signal. For a voltage output DAC, the conversion characteristic can be expressed by;

$$V_0 = kV_{FS}(b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \ \dots\ b_n 2^{-n})$$

Where, V_0 – Output voltage

V_{FS} – Full scale output voltage

k – Scaling factor (usually 1)

$b_1 \dots, b_n$ – n-bit binary fractional word with decimal point located at the left

b_1 – MSB with a weight = $V_{FS}/2$

b_n – LSB with a weight = $V_{FS}/2^n$

Performance Parameters of DAC:

1. **Resolution:** *Resolution* is the number of various analog output values that is provided by a DAC.

   For n-bit DAC; *Resolution* = 2^n

   Resolution can also be defined as the ratio of change in output voltage resulting from a change of LSB at the digital input. For n-bit DAC; *Resolution* = $V_{OFS}/2^n - 1$

Where, V_{OFS} – Full scale output voltage.

If we know the resolution, we can obtain input-output relation for DAC:

$$V_0 = Resolution\ x\ b$$

Where, b – Decimal values of digital input.

For example, in a 4-bit system, using ladder, the LSB has a weight of $\frac{1}{16}$. This means that, the smallest increment in the output voltage is $\frac{1}{16}$ of the input voltage. If we assume that, this 4-bit system has input voltage levels of +16 V; (since has a weight of $\frac{1}{16}$) a change in LSB results in a change of 1 V in the output. Thus, the output voltage changes in steps of 1 V.

Hence, this converter can be used to represent analog voltages from 0 to +15 V in 1-V increments. But, this converter cannot be used to resolve voltages into increments smaller than 1V. If we desire to produce +4.2 V, using this converter, the actual output voltage would be +4.0 V. This converter is not capable of distinguishing voltages finer than 1 V, which is the resolution of the converter.

If we want to represent voltages to a finer resolution, we would have to use a converter with more input bits. For example, the LSB of a 10-bit converter has a weight of 1/1024. If this converter has a +10 V full-scale output, the resolution is approximately, +10 * $\frac{1}{1024}$ = 10 mV.

Problem: *What is the resolution of a 9-bit D/A converter which uses a ladder network? What is the resolution expressed as a percentage? If the full-scale output voltage of this converter is +5 V, what is resolution in volts?*

Solution: The LSB in a 9-bit D/A converter has a weight of $\frac{1}{512}$. Thus, this converter has a resolution of 1 part in 512.

The resolution expressed in percentage is $\frac{1}{512}$ * 100 percent = 0.2 %.

The voltage resolution is obtained by multiplying the weight of the LSB by the full-scale output voltage. Thus, the resolution in volts is: $\frac{1}{512}$ * 5 = 10 mV.

Problem: *How many bits are required at the input of a converter, if it is necessary to resolve voltage to 5 mV and the ladder has +10 V full-scale?*

Solution: The LSB of an 11-bit D/A converter has a resolution of $\frac{1}{2048}$. This would provide a resolution at the output of $\frac{1}{2048}$ * 10 = +5 mV.

2. **Accuracy:** The *accuracy of the D/A converter* is primarily a function of the accuracy of the precision resistors used in the ladder and the precision of the reference voltage supply used. *Accuracy* is a measure of how close the actual output voltage is to the theoretical output value.

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

Problem: *Calculate the accuracy of a 8-bit DAC, if the full scale output voltage is 10.0 V?*

Solution: $Accuracy = \frac{V_{oFS}}{(2^n-1)2} = \frac{10.0}{(2^8-1)2} = 19.8\, mV.$

3. **Setting Time:** *Setting time* is the time required for a DAC output to settle within $\pm \frac{1}{2}$ *LSB* of final value for a given digital input.

4. **Stability:** The performance of a DAC is not *stable* due to the parameters such as temperature, power supply variations, and ageing.

Binary Weighted Resistor DAC:

The following Figure shows binary weighted resistor DAC circuit using *n*-electronic switches to control the binary inputs $b_1, b_2, \ldots, b_n$.

When the switch is ON; $I = \frac{V_R}{R}$ When the switch is OFF; I = 0.

Due to very high input impedance of Op-Amp, the toatl current *I* will flow through R_f. The total current trough R_f is; $I = I_1 + I_2 + I_3 + \ldots + I_n$

$$= \frac{V_R}{2^1R}b_1 + \frac{V_R}{2^2R}b_2 + \frac{V_R}{2^3R}b_3 + \ldots + \frac{V_R}{2^nR}b_n$$

$$= \frac{V_R}{R}[b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \ldots + b_n 2^{-n}]$$

The output voltage is; $V_0 = -I_f R_f$

$$i.e., V_0 = \frac{-V_R}{R} R_f [b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \ldots + b_n 2^{-n}]$$

If $R_f = R$; Or, $\boldsymbol{V_0 = -V_R[b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \ldots + b_n 2^{-n}]}$

Drawbacks: (1) Large range of resistor values are required, as resistance values increases like $2^1R, 2^2R, 2^3R, \ldots, 2^nR$.

(2) Practically it's difficult to fabricate large values of resistors on IC.

R-2R Ladder type DAC:

In R-2R ladder, only two values, R and 2R are used. The circuit diagram is given below:

Each binary bit connects switch either to ground (non-inverting input) or to the inverting terminal of Op-Amp. Due to virtual ground, both the positions of the switches are at ground potential, and currents through the resistances are constant.

The current flowing through each of 2R resistances;

$$I_1 = \frac{V_R}{2R} \qquad I_2 = \frac{V_R/2}{2R} = \frac{V_R}{4R} \qquad I_3 = \frac{V_R/4}{2R} = \frac{V_R}{8R} \qquad I_n = \frac{V_R/(2^n-1)}{2R}$$

$$\text{But, } V_0 = -I_f R_f = -R_f(I_1 + I_2 + \ldots + I_N)$$

$$i.e., V_0 = -R_f\left[\frac{V_R}{2R}b_1 + \frac{V_R}{4R}b_2 + \ldots + \frac{V_R}{2^n R}b_n\right]$$

$$\text{Or, } V_0 = -\frac{V_R}{R}R_f[b_1 2^{-1} + b_2 2^{-2} + \ldots + b_n 2^{-n}]$$

$$\text{If } R_f = R; \qquad \boldsymbol{V_0 = -V_R[b_1 2^{-1} + b_2 2^{-2} + \ldots + b_n 2^{-n}]}$$

Advantages: (1) As it requires only two types of resistors, fabrication and accurate value of R-@R can be designed.

(2) Node voltage remains constant, and hence, slow down effect can be avoided.

Example 1:

The digital input for a 4 bit DAC is D = 0111. Calculate its output voltage take V_{oFS} = 15 V.

$$\text{Resolution} = \frac{V_{oFS}}{2^n - 1} = \frac{15}{2^4 - 1} = 1\text{ V/LSB}$$

$$\therefore V_o = \text{Resolution} \times D$$

$$D = \text{Decimal values } (0111) = 7$$

$$V_o = \frac{1\text{ V}}{\text{LSB}} \times 7 = 7\text{ V}$$

$$\therefore \boxed{V_o = 7\text{ V}}$$

Example 2:

A 8 bit DAC having resolution of 22mv/LSB. Calculate V_{oFS} and output if the input is $(10000000)_2$.

Given: resolution = 22 mV, Input = $(1000000)_2$

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

$$22mv = \frac{V_{oFS}}{2^8 - 1}$$

$$V_{oFS} = 5.6 \text{ V}$$

D = equivalent of $(10000000)_2 = 128$

$V_o = 22 \times 10^{-3} \times 128 = 2.8$ V.

Example 3:

Calculate output voltage produced by DAC, when output range is between 0 and 10 V for input binary number.

a) 10 (2 bit DAC) b) 0011

a) From equation (9-7) we can write,

$$V_o = 10\text{ V}\left(1\times\frac{1}{2}+0\times\frac{1}{4}\right) = 5\text{ V}$$

b) From equation (9-7) we can write,

$$V_o = 10\text{ V}\left(0\times\frac{1}{2}+0\times\frac{1}{4}+1\times\frac{1}{8}+1\times\frac{1}{16}\right)$$

$$= 1.875 \text{ volts.}$$

Example 4:

Calculate the values of the LSB and full scale output for 4 bit DAC for 0 to 10 V range.

We have,

$$\text{LSB} = \frac{1}{2^4} = \frac{1}{16}$$

For 10 V range,

$$\text{LSB} = \frac{10\text{ V}}{16} = 625\text{ mV}$$

and

$$\text{MSB} = \left(\frac{1}{2}\right)\text{Fullscale}$$

$$= \frac{1}{2}\times 10 = 5\text{ V}$$

Full scale output = Full scale voltage – 1 LSB

= 10 – 625 mV = 9.375 V

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:

1. 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 (V_{in}) is compared with the analog output signal (V_a) 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 (V_a) proportional to 1000. This analog voltage is compared with input analog signal (V_{in}).

If $V_{in} > V_a$, the comparator output will be high and SAR keeps Q3 high. On the other hand, if $V_{in} < V_a$, then the comparator output becomes low and SAR resets Q3 to low. If $V_{in} > V_a$, SAR follows the upward path in code tree and if $V_{in} < V_a$, 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.

MODULE – 2

THE COMBINATIONAL LOGIC CIRCUITS

THE BASIC GATES

PREREQUISITES:

Electronic circuits and systems can be divided into two broad categories – *analog* and *digital*. Analog circuits are designed for use with small signals and are used in a linear fashion. Digital circuits are generally used with large signals and are considered nonlinear. Any quantity that changes with time can be represented as an analog signal or it can be treated as digital signal.

Digital electronics involves circuits that have exactly two possible states. A system having only two states is said to be *binary*. The binary number system is widely used in digital electronics.

Hexa-Decimal	Decimal	Binary		Hexa-Decimal	Decimal	Binary
0	0	0 0 0 0		8	8	1 0 0 0
1	1	0 0 0 1		9	9	1 0 0 1
2	2	0 0 1 0		A	10	1 0 1 0
3	3	0 0 1 1		B	11	1 0 1 1
4	4	0 1 0 0		C	12	1 1 0 0
5	5	0 1 0 1		D	13	1 1 0 1
6	6	0 1 1 0		E	14	1 1 1 0
7	7	0 1 1 1		F	15	1 1 1 1

The operation of electronic circuits can be described in terms of its voltage levels – *high* (H) level and *low* (L) level. This could be related to the binary number system by assigning L = 0 = F (false) and H = 1 = T (true).

Symmetrical Signal & Asymmetrical Signal

The frequency is defined as, f = 1 / T where, T is the period of the signal.

Duty Cycle is a convenient measure of how symmetrical or how unsymmetrical a waveform is.

$$Duty\ Cycle = \frac{T_{on}}{T_{on}+T_{off}} \qquad Duty\ Cycle, H = \frac{T_{on}}{T_{on}+T_{off}} \qquad Duty\ Cycle, L = \frac{T_{off}}{T_{on}+T_{off}}$$

REVIEW OF LOGIC GATES:

Circuit Symbol	Truth Table			VHDL	IC Details
	A	B	X		
NOT Gate: $X = \bar{A}$	0	-	1	X = ~A X <= **not** A;	
	1	-	0		
OR Gate: $X = A + B$	0	0	0	X = A | B X <= A **or** B;	
	0	1	1		
	1	0	1		
	1	1	1		
AND Gate: $X = A.B$	0	0	0	X = A & B X <= A **and** B;	
	0	1	0		
	1	0	0		
	1	1	1		
NOR Gate: $X = \bar{A}.\bar{B}$	0	0	1	X = ~(A | B) X <= A **nor** B;	
	0	1	0		
	1	0	0		
	1	1	0		
NAND Gate: $X = \bar{A} + \bar{B}$	0	0	1	X = ~(A & B) X <= A **nand** B;	
	0	1	1		
	1	0	1		
	1	1	0		
XOR Gate: $X = A \oplus B$ $= \bar{A}B + A\bar{B}$	0	0	0	X = A ^ B X <= A **xor** B;	
	0	1	1		
	1	0	1		
	1	1	0		
XNOR Gate: $X = A \odot B$ $= \bar{A}\bar{B} + AB$	0	0	1	X = ~(A ^ B) X <= A **xnor** B;	
	0	1	0		
	1	0	0		
	1	1	1		

Universality of NOR Gate:

Universality of NAND Gate:

Bubbled AND Gate:

Bubbled AND gate and NOR gate are equivalent

De Morgan's First Theorem:

The complement of a sum equals the product of the complements. $\overline{A+B} = \bar{A}.\bar{B}$

Proof:

A	B	A+B	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

NOR Gate Bubbled AND Gate

Bubbled OR Gate:

Bubbled OR gate and NAND gate are equivalent

De Morgan's Second Theorem:

The complement of a sum equals the product of the complements. $\overline{AB} = \bar{A} + \bar{B}$

Proof:

A	B	AB	$\overline{AB}$	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

NAND Gate Bubbled OR Gate

Duality Theorem: *Starting with a Boolean relation, you can derive another Boolean relation by –*

1. *Changing each OR sign to an AND sign*
2. *Changing each AND sign to an OR sign*
3. *Complementing any 0 or 1appearing in the expression.*

Example: 1. We say that, A+0 = A; the dual is, A.1 = A

2. Consider, A(B+C) = AB + AC

By changing the OR and AND operation, we get the dual relation:

A + BC = (A+B)(A+C)

Laws of Boolean Algebra:

- The following laws are of immense use in the simplification of Boolean expressions.

- Note that, if A is a variable, then either A = 0 or A = 1. Also, when A = 0, A ≠ 1; and when A = 1, A ≠ 0.

De Morgan's First Theorem:-

The complement of sum is equal to the product of the complements.

(A + B)' = A' . B' i.e., a bubbled AND gate & a NOR gate are equivalent.

De Morgan's Second Theorem:-

The complement of a product is equal to the sum of the compliments.

(A . B)' = A' + B' i.e., a bubbled OR gate & a NAND gate are equivalent.

1) *Commutative Law:-*

   A + B = B + A and A . B = B . A

2) *Associative Law:-*

   A + (B + C) = (A + B) + C and A . (BC) = (AB) . C

3) *Distributive Law:-*

   A(B + C) = AB + AC

4) *In relation to OR operation, the following laws hold good:-*

   A + 0 = A

   A + A = A

   A + 1 = 1 and

   A + A' = 1

5) *In relation to AND operation, the following laws hold good:-*

   A . 1 = A

   A . A = A

   A . 0 = 0

   A . A' = 0

   A'' = A

6) *Some more useful Boolean relations:-*

   A + AB = A

   A + A'B = A + B

   A (A + B) = A

   A (A' + B) = AB

   A + (B . C) = (A + B) (A + C)

Simplification of Boolean Expressions:-

- The following **hints** are found to be of use, in reducing complex Boolean expressions –

1. If there are parentheses present in the given expression, they are removed first; since, multiplication should precede addition.

   E.g.:- AB + C (A + B) = AB + AC + BC

2. If there are several identical terms, all except one can be removed.

   E.g.:- A + B + C + A . 1 = A + B + C + A = A + B + C

3. If a variable repeats in a term, only one variable may be retained.

   E.g.:- A . A = A

   B .B . C = BC

4. If in any term, both a variable & its complement are present, that term may be removed; since, AA' = 0.

   E.g.:- XX'Y = 0 . Y = 0

5. Identify pairs of terms which contains same variables. If in a pair, a variable is absent in one term, it can be removed.

   E.g.:- ABCD + ABC = ABC (D + 1)

   = ABC . 1 since, 1 + D =1

   = ABC

6. If, in a pair of terms, several variables are common, and another variable is present in one term & its complement is present in another term, this variable & its complement can be removed.

   E.g.:- ABC + A'BC = BC (A' + A)

   = BC . 1 since, A' + A = 1

   = BC

KARNAUGH MAPS

MINIMUM FORMS OF SWITCHING FUNCTIONS:

When a function is realized using AND and OR gates, *the cost of realizing the function is directly related to the number of gates and gate inputs used*. The Karnaugh map techniques developed, lead directly to *minimum cost* two-level circuits composed of AND and OR gates. An expression consisting of a *sum-of-product* terms corresponds directly to a two-level circuit composed of a group of AND gates feeding a single OR gate (see the following Figure). Similarly, a *product-of-sums* expression corresponds to a two-level circuit composed of OR gates feeding a single AND gate.

Therefore, to find minimum cost two-level AND-OR gate circuits, we must find minimum expressions in sum-of-products or product-of-sums form.

A *minimum sum-of-products* expression for a function is defined as a sum of product terms which

a) has a minimum number of terms and
b) of all those expressions which have the same minimum number of terms, has a minimum number of literals.

The minimum sum of products corresponds directly to a minimum two-level gate circuit which has

a) a minimum number of gates and
b) a minimum number of gate inputs.

Unlike the minterm expansion for a function, the minimum sum of products is not necessarily unique; that is, a given function may have two different minimum sum-of-products forms, each with the same number of terms and the same number of literals.

Given a minterm expansion, the minimum sum-of products form can often be obtained by the following procedure:

1. Combine terms by using $XY' + XY = X(Y' + Y) = X$. Do this repeatedly to eliminate as many literals as possible. A given term may be used more than once because X+X=X.
2. Eliminate redundant terms by using the theorems of Boolean Algebra.

Example

Find a minimum sum-of-products expression for

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

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

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

None of the terms in the above expression can be eliminated by consensus. However, combining terms in a different way leads directly to a minimum sum of products:

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

$$= a'b' + bc' + ac$$

A *minimum product-of-sums* expression for a function is defined as a product of sum terms which

a) has a minimum number of factors, and
b) of all those expressions which have the same number of factors, has a minimum number of literals.

Unlike the maxterm expansion, the minimum product-of-sums form of a function is not necessarily unique. Given a maxterm expansion, the minimum product of sums can often be obtained by a procedure similar to that used in the minimum sum-of-products case, except that the theorem $(X + Y')(X + Y) = X$ is used to combine terms.

Example

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

$$= (A + B' + D') \quad (A + B' + C') \quad (B' + C' + D) \quad (B + C' + D)$$

$$= (A + B' + D') \quad \underbrace{(A + B' + C')} \quad (C' + D)$$

eliminate by consensus

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

A	B	C	Y – Fundamental Product	Min-term
0	0	0	0	m0
0	0	1	0	m1
0	1	0	0	m2
0	1	1	1 – $\bar{A}BC$	m3
1	0	0	0	m4
1	0	1	1 – $A\bar{B}C$	m5
1	1	0	1 – $AB\bar{C}$	m6
1	1	1	1 – ABC	m7

$Y_{SOP} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

$= \sum m(1, 2, 4, 7)$.

$$Y_{SOP} = AB + BC + AC$$

SOP Circuit Diagram:

No. of Gates = 4

No. of Gate Inputs = 9

A	B	C	Y – Fundamental Sum	Max-term
0	0	0	0 – A + B + C	M0
0	0	1	0 – A + B + $\bar{C}$	M1
0	1	0	0 – A + $\bar{B}$ + C	M2
0	1	1	1	M3
1	0	0	0 – $\bar{A}$ + B + C	M4
1	0	1	1	M5
1	1	0	1	M6
1	1	1	1	M7

$Y_{POS} = (A + B + C)\,(A + B + \bar{C})\,(A + \bar{B} + C)\,(\bar{A} + B + C)$

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

$$Y_{POS} = (A + B)(B + C)(A + C)$$

POS Circuit Diagram:

No. of Gates = 4

No. of Gate Inputs = 9

Example: Adders & Subtractors

Adder *circuit is a combinational digital circuit that is used for adding two numbers. A typical adder circuit produces a sum bit (denoted by S) and a carry bit (denoted by C) as the output. Adder circuits are of two types: Half adder ad Full adder.*

Subtractor *is the one which used to subtract two binary numbers (digits) and provides Difference and Borrow as an output.*

Half Adder & Half Subtractor:

Schematic

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth table

Realization

$$Sum, S = \bar{A}B + A\bar{B} \qquad Carry, C = AB$$

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$Difference, D = \bar{A}B + A\bar{B} \qquad Borrow, B_o = \bar{A}B$$

Full Adder & Full Subtractor:

A	*B*	*Ci*	*S*	*Co*		*A*	*B*	*Ci*	*D*	*Bo*
0	*0*	*0*	*0*	*0*		*0*	*0*	*0*	*0*	*0*
0	*0*	*1*	*1*	*0*		*0*	*0*	*1*	*1*	*1*
0	*1*	*0*	*1*	*0*		*0*	*1*	*0*	*1*	*1*
0	*1*	*1*	*0*	*1*		*0*	*1*	*1*	*0*	*1*
1	*0*	*0*	*1*	*0*		*1*	*0*	*0*	*1*	*0*

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

Sum, ***S*** = ∑***m (1, 2, 4, 7)*** = ∏***M(0, 3 , 5, 6)***.

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

$$Or, S = \bar{C}(\bar{A}B + A\overline{\bar{B}}) + C(\bar{A}\bar{B} + AB)$$

$$Or, S = \bar{C}(A \oplus B) + C(A \odot B)$$

$$Or, S = \bar{C}(A \oplus B) + C(\overline{\overline{A \oplus B}})$$

$$Therefore, \boldsymbol{S = A \oplus B \oplus C}$$

Difference, ***D*** = ∑***m (1, 2, 4, 7)*** = ∏***M(0, 3, 5, 6)***.

$$Diff., D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Or, D = \bar{C}(\bar{A}B + A\overline{\bar{B}}) + C(\bar{A}\bar{B} + AB)$$

$$Or, D = \bar{C}(A \oplus B) + C(A \odot B)$$

$$Or, D = \bar{C}(A \oplus B) + C(\overline{\overline{A \oplus B}})$$

$$Therefore, \boldsymbol{D = A \oplus B \oplus C}$$

Carry Out, ***Co*** = ∑***m (3, 5, 6, 7)*** = ∏***M(0, 1, 2, 4)***.

$$Carry\ Out, Co = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$Or, Co = \bar{C}(AB) + C(\bar{A}B + A\bar{B} + AB)$$

$$Therefore, \boldsymbol{Co = AB + BC + AC}$$

Borrow, ***Bo*** = ∑***m (1, 2, 3, 7)*** = ∏***M(0, 4, 5, 6)***.

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

$$Or, Bo = \bar{C}(\bar{A}B) + C(\bar{A}\bar{B} + \bar{A}B + AB)$$

$$Therefore, \boldsymbol{Bo = \bar{A}B + BC + \bar{A}C}$$

TWO AND THREE VARIABLE KARNAUGH MAPS:

Just like a truth table, the Karnaugh map of a function specifies the value of the function for every combination of values of the independent variables. The following Figure shows the truth table for a function F and the corresponding Karnaugh map:

The following Figure shows a three-variable truth table and the corresponding Karnaugh map:

Example: *Write the Karnaugh Map for – (a) f = ∑m (1, 3, 5)* (b) $f(a,b,c) = abc' + b'c + a'$ *(c) F = ∑m (0, 1, 2, 5, 6, 7)*

Solution: (a) f = ∑m (1, 3, 5) (b) $f(a,b,c) = abc' + b'c + a'$ *(c) F = ∑m (0, 1, 2, 5, 6, 7)*

Example: *Find two different minimum sum-of-products expressions for the function G = ∑m (0, 2, 3, 4, 5, 7).*

Solution: Given, G = ∑m (0, 2, 3, 4, 5, 7);

FOUR-VARIABLE KARNAUGH MAPS:

The following Figure shows the location of minterms on a four-variable map & plot of four-variable expression $f(a,b,c,d) = acd + a'b + d'$ on a Karnaugh map:

f	$\bar{a}\bar{b}$	$\bar{a}b$	ab	$a\bar{b}$
$\bar{c}\bar{d}$	0	4	12	8
$\bar{c}$d	1	5	13	9
cd	3	7	15	11
$c\bar{d}$	2	6	14	10

f	00	01	11	10
00				
01				
11				
10				

f (a, b, c, d) =

Example: *Write the Karnaugh map for* (a) $f = x'z' + wxy + x'y$

(b) $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C} + AB\bar{C}$

Solution:

f	$\bar{w}\bar{x}$	$\bar{w}x$	wx	$w\bar{x}$
$\bar{y}\bar{z}$				
$\bar{y}z$				
yz				
$y\bar{z}$				

Y	0	1
00		
01		
11		
10		

f (w, x, y, z) =

Y (A, B, C) =

Example: *Write the Karnaugh map for (a) f1 = = ∑m (3, 4, 5, 6, 7, 9, 12, 13); (b) f2 = = ∑m (2, 3, 5, 7, 8, 10, 11, 15).*

Solution: (a) Given, f1 = = ∑m (3, 4, 5, 6, 7, 9, 12, 13) & f2 = = ∑m (2, 3, 5, 7, 8, 10, 11, 15);

cd \ ab	00	01	11	10
00		1	1	
01		1	1	1
11	1	1		
10		1		

cd \ ab	00	01	11	10
00				1
01		1		
11	1	1	1	1
10	1			1

f_1 = _____ f_2 = _____

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 Karnaugh Map with an Example.

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

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.)

PAIRS, QUADS, AND OCTETS:

Pairs: The following K-map contains a pair of 1s that are horizontally adjacent. Two adjacent 1s, such as these are called a *pair*. *A pair eliminates one variable and its complement*.

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

The sum-of-product equation is:

$Y = ABCD + ABCD' = ABC(D + D') = ABC$

Quad: A *quad* is a group of four 1s that are horizontally or vertically adjacent. *A quad eliminates two variables and their complements*.

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

The sum-of-product equation is:

$Y = ABC' + ABC = AB(C + C') = AB$

The Octet: The *octet* is a group of eight 1s, as shown in the following Fig. *An octet eliminates three variables and their complements*.

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

The sum-of-product equation is:

$Y = AB + AB' = A(B + B') = A$

KARNAUGH SIMPLIFICATIONS:

A pair eliminates one variable and its complement. A quad eliminates two variables and their complements. An octet eliminates three variables and their complements. Because of this, after drawing the K-map, first encircle the octets, then the quads, and finally the pairs, to get highest simplification.

Example: *Using K-map, simplify; Y = ∑m (1, 2, 3, 6, 8, 9, 10, 12, 13, 14).*

Solution:

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

$$Y = AC' + CD' + A'B'D$$

Overlapping Groups: Always overlap groups.

Y1	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	0	1	0	0
CD	1	1	1	1
$C\bar{D}$	1	1	1	1

Y1 =

Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	0	1	0	0
CD	1	1	1	1
$C\bar{D}$	1	1	1	1

Y2 =

Rolling the Map:

Y3	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	0	0	1
CD	1	0	0	1
$C\bar{D}$	0	0	0	0

Y3 =

Y4	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	0	0	0	0
$\bar{C}D$	1	0	0	1
CD	1	0	0	1
$C\bar{D}$	0	0	0	0

Y4 =

Rolling and Overlapping:

Y1	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	0
$\bar{C}D$	1	1	0	1
CD	1	1	0	1
$C\bar{D}$	1	1	0	0

Y1 =

Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	0
$\bar{C}D$	1	1	0	1
CD	1	1	0	1
$C\bar{D}$	1	1	0	0

Y2 =

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

Y3 =

Y4	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	1
$\bar{C}D$	1	1	0	1
CD	1	1	0	0
$C\bar{D}$	1	1	0	1

Y4 =

Y5	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1	0	1
$\bar{C}D$	1	1	0	1
CD	1	1	0	0
$C\bar{D}$	1	1	0	1

Y5 =

Eliminating Redundant Groups: After encircling groups, eliminate any *redundant groups*. This is a group whose 1s are already used by other groups.

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

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

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

Homework:

1] Determine the minimum sum-of-products for –

a) *f1 (a, b, c) = ∑(1, 3, 4, 5, 6, 7)*
b) *f2 (a, b, c) = Π (2, 4, 7)*
c) *f3 (a, b, c, d) =* $b'c'd' + bcd + acd' + a'b'c + a'bc'd$

2] Determine the minimum product-of-sums for –

a) *f1 (a, b, c) = ∑ (0, 1, 2,3, 4, 6, 7)*
b) *f2 (a, b, c) = Π (1, 4, 5)*
c) *f3 (a, b, c, d) =* $b'c'd' + bcd + acd' + a'b'c + a'bc'd$

3] Solve for the simplified Boolean expression using K-Map:

a) $f1\,(a,b,c,d) = \bar{a}\bar{c}d + \bar{a}cd + \bar{b}\bar{c}\bar{d} + a\bar{b}c + \bar{a}\bar{b}c\bar{d}$
b) $f2\,(a,b,c,d) = (a + b + \bar{d})(\bar{a} + b + \bar{d})(a + \bar{b} + \bar{c} + d)(\bar{a} + \bar{b} + \bar{c} + \bar{d})(\bar{a} + \bar{b} + \bar{c} + d)$

4] Find the minimum sum-of-products for –

(a) f1 (a, b, c) = m0 + m2 + m5 + m6
(b) f2 (d, e, f) = ∑m (0, 1, 2, 4)
(c) f3 (r, s, t) = $rt' + r's' + r's$)
(d) f4 (x, y, z) = M0 . M5

5] Design a 3-input, 1-output, minimal two-level gate combinational circuit; which has an output equal to 1 when majority of its inputs are at logic 1, and has output 0 when majority of inputs are at logic 0.

6] Design a minimal sum and minimal product combinational gate circuit to generate the odd parity bit for an 8421 BCD code.

QUINE McCLUSKEY (QM) METHOD

The Karnaugh map method is an effective way to simplify switching functions which have a small number of variables. When the number of variables is large or if several functions must be simplified, the use of a digital computer is desirable.

The *Quine-McCluskey* method provides a systematic simplification procedure which can be readily programmed for a digital computer. The Quine-McCluskey method reduces the minterm expansion (standard sum-of-products form) of a function to obtain a minimum sum of products.

DETERMINATION OF PRIME IMPLICANTS:

- ✓ In order to apply the Quine-McCluskey method to determine a minimum sum-of-products expression 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 exactly one variable. The examples given below show both the binary notation and its algebraic equivalent.

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

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

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

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

- 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 those in 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 to eliminate 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 terms in groups 1 and 2. Comparing term 1 with all terms in group 2, we find that it combines with 5 and 9 but not with 6 or 10. Similarly, term 2 combines only with 6 and 10, and term 8 only with 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	Column 2	Column 3
abcd	abcd	abcd

- Note that the terms in Column 2 have been divided into groups. In order to combine two terms, the terms 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 corresponding places and which differ by exactly one in the number of 1's.
- Terms in the first group in Column 2 need only be compared with terms in the second group which have dashes in the same places. Term 000– (0, 1) combines only with term 100– (8, 9) to yield –00– ($b'c'$).
- The resulting term is listed in Column 3 along with the designation 0, 1, 8, 9 to indicate that it was formed by combining minterms 0, 1, 8, and 9.
- Term (0, 2) combines only with (8, 10), and term (0, 8) combines with both (1, 9) and (2, 10).
- 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).
- Note that there are three pairs of duplicate terms in Column 3. These duplicate terms were formed in each case by combining the same set of four minterms in a different order.

- ✓ After deleting the duplicate terms, we compare terms from the two groups in Column 3. Because no further combination is possible, the process terminates.

- ✓ 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;

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

Definition:

- ✓ Given a function F of n variables, a product term P is an ***implicant*** of F iff for every combination of 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 if any literal is deleted from it.

Consider an Example:

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

- o In the above function, the implicant $a'b'c'$ is not a prime implicant because a can be eliminated, and the resulting term $b'c'$ is still an implicant of F. The implicants $b'c'$ and ac 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 the remaining 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 not been 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) $(a'bc)$										

- ✓ In the first row, *X*'s are placed in columns 0, 1, 8, and 9, because prime implicant $b'c'$ was formed from the sum of minterms 0, 1, 8, and 9. Similarly, the all other *X*'s are placed.
- ✓ If a minterm is covered by only one prime implicant, then that prime implicant is called an *essential prime implicant* and must be included in the minimum sum of products. Essential prime implicants are 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.
- ✓ Each time a prime implicant is selected for inclusion in the minimum sum, the corresponding row should be crossed out. After doing this, the columns which correspond to all minterms covered by that prime implicant should also be crossed out.
- ✓ 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	

Column 1 abc	Column 2 abc

The following Table shows the resulting prime implicants chart:

-	0	1	2	5	6	7

Therefore, ***F =***

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

Example: *Solve, using Quine Mc-Cluskey method & K-Map method: F = ∑m (0, 1, 2, 8, 10, 11, 14, 15).*

Solution:

Quine Mc-Clusky method:

A	B	C	D	F
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	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Stage 1 ABCD	Stage 2 ABCD	Stage 3 ABCD

-	0	1	2	8	10	11	14	15
(0, 1) $(A'B'C')$								
(0, 2, 8, 10) $(B'D')$								
(10, 11, 14, 15) (AC)								

Therefore, $\boldsymbol{F} = AC + B'D' + A'B'C'$

K-Map Method:

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

$\boldsymbol{F} = AC + B'D' + A'B'C'$

Homework: *Using Quine-McClusky method, simplify;*

a) $Y = \sum m\ (0, 1, 2, 3, 10, 11, 12, 13, 14, 15)$

b) $Y = \sum m\ (2, 6, 7)$.

PATRICK'S METHOD:

- ✓ Petrick's method is a technique for determining all minimum sum-of-products solutions from a prime implicant chart. The example discussed above has two minimum solutions. As the number of variables increases, the number of prime implicants and the complexity of the prime implicant chart may increase significantly. In such cases, a large amount of trial and error may be required to find the minimum solution(s).
- ✓ Petrick's method is a more systematic way of finding all minimum solutions from a prime implicant chart than the method used previously. Before applying Petrick's method, all essential prime implicants and the minterms they cover should be removed from the chart.

Consider the following Table:

- ✓ First, we will label the rows of the table P1, P2, P3, etc. We will form a logic function, P, which is true when all of the minterms in the chart have been covered. Let P1 be a logic variable which is true when the prime implicant in row P1 is included in the solution, P2 be a logic variable which is true when the prime implicant in row P2 is included in the solution, etc.
- ✓ Since, column 0 has X's in rows P1 and P2, we must choose row P1 or P2 in order to cover minterm 0. Therefore, the expression (P1+P2) must be true.
- ✓ In order to cover minterm 1, we must choose row P1 or P3; therefore, (P2+P3) must be true. In order to cover minterm 2, (P2+P4) must be true.
- ✓ Similarly, in order to cover minterms 5, 6, and 7, the expressions (P3+P5), (P4+P6) and (P5+P6) must be true.
- ✓ Since we must cover all of the minterms, the following function must be true:

- ✓ The next step is to reduce P to a minimum sum-of-products. This is easy because there are no complements. First, we multiply out, using $(X+Y)(X+Z) = X + YZ$ and the ordinary distributive law:

- ✓ Next we use $X + XY = X$ to eliminate redundant terms from P, which gives;

- ✓ Because P must be true (P = 1) in order to cover all of the minterms, we can translate the equation back into words as follows. In order to cover all of the minterms, we must choose rows P1 and P4 and P5, or rows P1 and P2 and P5 and P6, or . . . or rows P2 and P3 and P6.
- ✓ Although there are five possible solutions, only two of these have the minimum number of rows. Thus, the two solutions with the minimum number of prime implicants are obtained by choosing rows P1, P4, and P5 or rows P2, P3, and P6.
- ✓ Thus; $F = a'b' + bc' + ac$ *o*r F = a'c' + b'c + ab are two minimum solutions.

SIMPLIFICATION OF INCOMPLETELY SPECIFIED FUNCTIONS:

In some digital systems, certain input conditions never occur during normal operation; therefore, the corresponding output never appears. Since the output never appears, it is indicated by an *X* in the truth table. The *X* is called a *don't-care condition*.

Remember these points about don't-care conditions:

1. Given the truth table, draw the K-map and transfer 0s, 1s, and don't-care terms.
2. Encircle the actual 1s on the K-map in the largest groups you can find treating don't cares as 1s.
3. After the actual1s have been included in the groups, disregard the remaining don't cares by visualizing them as 0s.

Example: Consider the following truth table with don't care conditions for all the inputs from 1010 to 1111.

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

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

Y =

Problem: *What is the simplest logic circuit for –*

a) *Y1 = F (A, B, C, D) = ∑m (0) + ∑d (8, 9, 10, 11, 14, 15)*
b) *Y2 = F (A, B, C, D) = ∑m (0) + ∑d (12, 13, 14, 15)*
c) *Y3 = F (A, B, C, D) = ∑m (7) + ∑d (10, 11, 12, 13, 14, 15).*

Solution:

(a)

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

Therefore, Y1 = $B'C'D'$

(b)

Y2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}D$				
CD				
$C\bar{D}$				

Therefore, Y2 = $A'B'C'D'$

(c)

Y3	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$				
$\bar{C}D$				
CD				
$C\bar{D}$				

Therefore, Y3 = BCD

Don't-Care Conditions in Quine McCluskey Method:

- In the process of finding the prime implicants, we will treat the don't-care terms as if they were required minterms. In this way, they can be combined with other minterms to eliminate as many literals as possible. When forming the prime implicant chart, the don't-cares are not listed at the top.
- This way, when the prime implicant chart is solved, all of the required minterms will be covered by one of the selected prime implicants. However, the don't-care terms are not included in the final solution unless they have been used in the process of forming one of the selected prime implicants.

Homework: *Using Quine-McCluskey method (same questions can be asked to solve by using Patrick's method also), simplify;*

a) *f (a, b, c, d) = ∑m (3, 4, 5, 7, 10, 12, 14, 15) + ∑d (2)*
b) *f (a, b, c, d) = ∑m (1, 5, 7, 9, 11, 12, 14, 15)*
c) *f (a, b, c, d) = ∑m (0, 1,0 3, 5, 6, 7, 8, 10, 14, 15)*
d) *f (a, b, c, d) = ∑m (1, 3, 4, 5, 6, 7, 10, 12, 13) + ∑d (2, 9, 5)*
e) *f(a, b, c, d) = ∑m (9, 12, 13, 15) + ∑d (1, 4, 5, 7, 81 11, 14).*

Example: *Solve using QM method: F (A, B, C, D) = ∑m (2, 3, 7, 9, 11, 13) + ∑d (1, 10, 15).*

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

A	B	C	D	F
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	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Column 1 ABCD	Column 2 ABCD	Column 3 ABCD

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

-	2	3	7	9	11	13

Therefore, F =

SIMPLIFICATION USING MAP-ENTERED VARIABLES:

Although the Quine-McCluskey method can be used with functions with a fairly large number of variables, it is not very efficient for functions that have many variables and relatively few terms. Some of these functions can be simplified by using a modification of the Karnaugh map method. By using map-entered variables, Karnaugh map techniques can be extended to simplify functions with more than four or five variables. The following Figure shows a four-variable map with two additional variables entered in the squares in the map.

CD \ AB	00	01	11	10
00	1			
01	X	E	X	F
11	1	E	1	1
10	1			X

G

CD \ AB	00	01	11	10
00	1			
01	X		X	
11	1		1	1
10	1			X

$E = F = 0$
$MS_0 = A'B' + ACD$

CD \ AB	00	01	11	10
00	X			
01	X	1	X	
11	X	1	X	X
10	X			X

$E = 1, F = 0$
$MS_1 = A'D$

CD \ AB	00	01	11	10
00	X			
01	X		X	1
11	X		X	X
10	X			X

$E = 0, F = 1$
$MS_2 = AD$

When *E* appears in a square, this means that if *E = 1*, the corresponding minterm is present in the function G, and if *E = 0*, the minterm is absent. Thus, the map represents the six-variable function;

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

Example: *Simplify Y (A, B, C) = ∑m (2, 6, 7) by using entered variable map method by taking –*

a) "C" as map entered variable

b) "A as map entered variables.

Solution: Let Y = ∑m (2, 6, 7)

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

A	B	Y_C
0	0	
0	1	
1	0	
1	1	

B	C	Y_A
0	0	
0	1	
1	0	
1	1	

(a)	A'	A
B'	0	0
B	C'	1

$Y_C = BC' + AB$

(b)	B'	B
C'	0	1
C	0	A

$Y_A = AB + BC'$

(c)	A'	A
B'	0	C
B	C'	1

$Y = AB + BC'$

Simplification is similar to K-map method. In Fig (a), C' is grouped with 1 to get a larger group as 1 can be written ac 1 = 1 + C'. Similarly, A is grouped with 1 in Fig (b).

Now, the product term representing each group is obtained by including map entered variable (MEV) in the group as an additional ANDed term.

Hence, for Fig (a): $Y = B\bar{C} + AB$. *For Fig (b):* $Y = B\bar{C} + AB$.

Consider the EBM shown in Fig (c). This has only two product terms; and doesn't need a separate coverage for 1. This is because, one can write 1 = C + C', and C is included in one group and C' is included in other group.

Example: *Simplify Y (A, B, C) = ∑m (1, 2, 3, 4, 8, 9, 10, 13, 14) by using entered variable map method by taking –*

a) "D" as map entered variable

b) "C and D" as map entered variables.

Solution:

A	B	C	D	Y	
0	0	0	0	0	D
0	0	0	1	1	C+D
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	1	$\overline{D}$
0	1	0	1	0	$\overline{C+D}$
0	1	1	0	0	0
0	1	1	1	0	
1	0	0	0	1	1
1	0	0	1	1	$\overline{C\cdot D}$
1	0	1	0	1	$\overline{D}$
1	0	1	1	0	
1	1	0	0	0	D
1	1	0	1	1	C⊕D
1	1	1	0	1	$\overline{D}$
1	1	1	1	0	

A	B	C	Y
0	0	0	D
0	0	1	1
0	1	0	$\overline{D}$
0	1	1	0
1	0	0	1
1	0	1	$\overline{D}$
1	1	0	D
1	1	1	$\overline{D}$

A \ BC	00	01	11	10
0	D	1	0	$\overline{D}$
1	1	$\overline{D}$	$\overline{D}$	D

Compressed one time

A	B	Y
0	0	C+D
0	1	$\overline{C+D}$
1	0	$\overline{C\cdot D}$
1	1	C⊕D

A \ B	0	1
0	C+D	$\overline{C+D}$
1	$\overline{C\cdot D}$	C⊕D

Compressed two times

Example: *Solve by using (a) K-Map method & (b) MEV method taking "D" as map entered variable:* $F(A,B,C,D) = A'B'C + A'BC + A'BC'D + ABCD + (AB'C)$*; where* $AB'C$ *is a don't-care term.*

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

i.e., $F = A'B'C(D+D') + A'BC(D+D') + A'BC'D + ABCD + [AB'C(D+D')]$

or $F = A'B'C'D' + A'B'C'D + A'BCD' + A'BCD + ABCD + (AB'CD' + AB'CD)$

i.e., F = ∑m (0, 1, 6, 7, 15) + ∑d (10, 11).

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

Y(A, B, C, D) =

MEV method taking "D" as map entered variable:

A	B	C	D	Y	Y_D
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	0
0	0	1	1	0	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	0
1	0	0	1	0	
1	0	1	0	X	X
1	0	1	1	X	
1	1	0	0	0	0
1	1	0	1	0	
1	1	1	0	0	D
1	1	1	1	1	

Y (A, B, C) =

Exercise:

a) *Design (a) Binary-to-Gray Code Converter, and (b) Gray-to-Binary Code Converter*

b) *A switching circuit has two control inputs (C1 and C2), two data inputs (X1 and X2), and one output (Z). The circuit performs one of the logic operations AND, OR, EQU (equivalence), or XOR (exclusive OR) on the two data inputs. The function performed depends on the control inputs:*

C_1	C_2	Function Performed by Circuit
0	0	OR
0	1	XOR
1	0	AND
1	1	EQU

(i) *Derive a truth table for Z*

(ii) *Use a Karnaugh Map to find minimum AN-OR Gate Circuit to realize Z.*

c) *A logic circuit realizing the function f has four inputs a, b, c, d. The three inputs a, b, and c are the binary representation of the digits 0 through 7 with a being the most significant bit. The input*

d is an odd-parity bit; that is, the value of d is such that a, b, c, and d always contains an odd number of 1's. (For example, the digit 1 is represented by abc = 001 and d = 0, and the digit 3 is represented by abcd = 0111.) The function f has value 1 if the input digit is a prime number. (A number is prime if it is divisible only by itself and 1; 1 is considered to be prime, and 0 is not.)

a. *Draw a Karnaugh map for f*
b. *Find all prime implicants of f*
c. *Find all minimum sum of products for f*
d. *Find all prime implicants of f'*
e. *Find all minimum product of sums for f.*

Try these: *Design a minimal sum combinational circuit to –*

a) *Find the 9s complement of BCD numbers*
b) *Convert BCD to Excess-3*
c) *Multiply two 2-bit numbers*
d) *Output a 1 when an illegal BCD code occurs*
e) *Output the 2s complement of a 4-bit binary number.*

MODULE – 3

COMBINATIONAL LOGIC CIRCUITS

COMBINATIONAL CIRCUIT DESIGN & SIMULATION USING GATES

REVIEW OF COMBINATIONAL CIRCUIT DESIGN:

Steps involved in the design of a combinational switching circuit:

1. 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 corresponding output values are don't-cares.
2. 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.
3. 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.
4. 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* $f(a,b,c,d) = \Sigma m(0,3,4,5,8,9,10,14,15)$ *using three input NOR gates.*

Solution:

f	$\bar{a}\bar{b}$	$\bar{a}b$	ab	$a\bar{b}$
$\bar{c}\bar{d}$	1	1	0	1
$\bar{c}$d	0	1	0	1
cd	1	0	1	0
$c\bar{d}$	0	0	1	1

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, two four-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., $f' = abc' + a'c(b + d') + b'd(a'c' + ac)$

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

Example: *Realize the following functions using only two-input NAND gates and inverters.*

$f1 = \Sigma m(0,2,3,4,5)$ $\quad$ $f2 = \Sigma m(0,2,3,4,7)$ $\quad$ $f3 = \Sigma m(1,2,6,7)$

Solution:

$f_1 = \Sigma\, m(0, 2, 3, 4, 5)$

$f_1 = b'c' + ab' + a'b$

$f_2 = \Sigma\, m(0, 2, 3, 4, 7)$

$f_2 = b'c' + bc + a'b$

$f_3 = \Sigma\, m(1, 2, 6, 7)$

$f_3 = a'b'c + ab + bc'$

Each function requires a three-input OR gate; so we will factor to reduce the number of gate inputs:

$$f_1 = b'(\underline{a + c'}) + \underline{a'b}$$
$$f_2 = b(a' + c) + b'c' \qquad \text{or} \qquad f_2 = (b' + c)(b + c') + \underline{a'b}$$
$$f_3 = a'b'c + b(\underline{a + c'})$$

The second expression for f2 has a term common to f1, so we will choose the second expression. Also, we can eliminate the remaining three-input gate from f3 by noting that; $a'b'c = a'(b'c) = a'(b + c')'$.

The following Figures show the resulting circuits:

GATE DELAYS AND TIMING DIAGRAMS:

When the input to a logic gate is changed, the output will not change instantaneously. The gates take a finite time to react to a change in input, so that the change in the gate output is delayed with respect to the input change. The following Figure shows possible input and output waveforms for an inverter:

If the change in output is delayed by time, ε, with respect to the input, we say that, the gate has a propagation delay of ε. In practice, the propagation delay for a 0 to 1 output change may be different than the delay for a 1 to 0 change. In many cases these delays can be neglected. However, in the analysis of some types of sequential circuits, even short delays may be important.

The following Figure shows the timing diagram for a circuit with two gates:

Assume that, each gate has a propagation delay of 20 ns (nanoseconds). This timing diagram indicates what happens when gate inputs B and C are held at constant values 1 and 0, respectively, and input A is changed to 1 at *t = 40 ns* and then changed back to 0 at *t = 100 ns*. 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.

- If, 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***.
- Similarly, if the output may momentarily go to 1 when it should remain a 0, we say that the circuit has a ***static 0-hazard***.
- If, when the 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 following Figure shows possible outputs from a circuit with hazards:

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

Note that hazards are properties of the circuit and are independent of the delays existing in the circuit.

The following Figure illustrates a circuit with a static 1-hazard.

(a) Circuit with a static 1-hazard

(b) Timing chart

If $A = C = 1$, then $F = B + B' = 1$, so the F output should remain a constant 1 when B changes from 1 to 0. However, as shown in Figure (b), if each gate has a propagation delay of 10 ns, E will go to 0 before D goes to 1, resulting in a momentary 0 (a glitch caused by the 1-hazard) appearing at the output F. Note that right after B changes to 0, both the inverter input (B) and output (B') are 0 until the propagation delay has elapsed. During this period, both terms in the equation for F are 0, so F momentarily goes to 0.

Detection of Static 1 Hazard: Hazards can be detected using a Karnaugh map (see above Figure). As seen on the map, no loop covers both minterms ABC and ABC'. So if A = C = 1 and B changes, both terms can momentarily go to 0, resulting in a glitch in F.

We can detect hazards in a two-level AND-OR circuit, using the following procedure:

1. Write down the sum-of-products expression for the circuit.
2. Plot each term on the map and loop it.
3. If any two adjacent 1's are not covered by the same loop, a 1-hazard exists for the transition between the two 1's. For an n-variable map, this transition occurs when one variable changes and the other $n - 1$ variables are held constant.

To Eliminate Static 1 Hazard: If we add a loop to the map of above Figure and, then, add the corresponding gate to the circuit (as shown in the following Figure), this eliminates the hazard. The term AC remains 1 while B is changing, so no glitch can appear in the output. Note that F is no longer a minimum sum of products.

Detection of Static 0 Hazard: The following Figure shows a circuit with several 0-hazards. The product-of-sums representation for the circuit output is $F = (A + C)(A' + D')(B' + C' + D)$

The Karnaugh map for this function (see following Figure) shows four pairs of adjacent *0*'s that are not covered by a common loop as indicated by the arrows. Each of these pairs corresponds to a 0-hazard. For example, when A = 0, B = 1, D = 0, and C changes from 0 to 1, a spike may appear at the Z output for some combination of gate delays. The timing diagram of (shown below) illustrates this assuming gate delays of 3 ns for each inverter, and of 5 ns for each AND gate and each OR gate.

(a) Circuit with a static 0-hazard

(b) Karnaugh map for circuit of (a)

(c) Timing diagram illustrating 0-hazard of (a)

To Eliminate Static 0 Hazard: We can eliminate the 0-hazards by looping additional prime implicants that cover the adjacent 0's that are not already covered by a common loop. This requires three additional loops as shown in shown in the following Figure. The resulting circuit requires seven gates in addition to the inverters, as given by below expression.

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

Hazards in circuits with more than two levels can be determined by deriving either a SOP or POS expression for the circuit that represents a two-level circuit containing the same hazards as the original circuit. The SOP or POS expression is derived in the normal manner except that the complementation laws are not used, i.e., $xx' = 0$ and $x + x' = 1$ are not used. Consequently, the resulting SOP (POS) expression may contain products (sums) of the form $xx'\alpha$ $(x + x' + \beta)$. (α is a product of literals or it may be null; β is a sum of literals or it may be empty).

Dynamic Hazard: A dynamic hazard exists if there is a term of the form $xx'\alpha$ and two conditions are satisfied: (1) There are adjacent input combinations on the Karnaugh map differing in the value of x, with $\alpha = 1$ and with opposite function values, and (2) for these input combinations the change in x propagates over at least three paths through the circuit.

Consider the following expression and its Karnaugh map;

$$f = a'c'd + bc'd + bcd' + acd'$$

$$\begin{aligned} f &= (c' + ad' + bd')(c + a'd + bd) \\ &= cc' + acd' + bcd' + a'c'd + aa'dd' + a'bdd' + bc'd + abdd' + bdd' \\ &= cc' + acd' + bcd' + a'c'd + aa'dd' + bc'd + bdd' \end{aligned}$$

The circuit does not contain any static 1-hazards because each pair of adjacent 1's are covered by one of the product terms. Potentially, the terms cc' and bdd' may cause either static 0- or dynamic hazards or both; the first for c changing and the second for d changing.

To design a circuit which is free of static and dynamic hazards, the following procedure may be used:

1. Find a sum-of-products expression (F^t) for the output in which every pair of adjacent 1's is covered by a 1-term. (The sum of all prime implicants will always satisfy this condition.) A two-level AND-OR circuit based on this F^t will be free of 1-, 0-, and dynamic hazards.
2. If a different form of the circuit is desired, manipulate F^t to the desired form by simple factoring, DeMorgan's laws, etc. Treat each x_i and x_i' as independent variables to prevent introduction of hazards.

SIMULATION AND TESTING OF LOGIC CIRCUITS:

An important part of the logic design process is verifying that the final design is correct and debugging the design if necessary. Logic circuits may be tested either by actually building them or by simulating them on a computer. Simulation is generally easier, faster, and more economical. As logic circuits become more and more complex, it is very important to simulate a design before actually building it. This is particularly true when the design is built in integrated circuit form, because fabricating an integrated circuit may take a long time and correcting errors may be very expensive.

Simulation is done for following reasons: (1) Verification that the design is logically correct, (2) Verification that the timing of the logic signals is correct, and (3) Simulation of faulty components in the circuit as an aid to finding tests for the circuit.

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. The outputs of the gates which changed in the previous step are fed into the next level of gate inputs. If the input to any gate has changed, then the output of that gate is calculated.
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.

Four-Valued Logic Simulator: 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***. At other times we may have no logic signal at an input, as in the case of an open circuit when an input is not connected to any output. We use the logic value ***Z*** to represent an ***open circuit***, or ***high impedance*** (hi-Z) connection.

The following Figure shows a typical simulation screen on a personal computer.

(a) Simulation screen showing switches (b) Simulation screen with missing gate input

The following Table shows AND and OR functions for four-valued logic simulation. These functions are defined in a manner similar to the way real gates work.

•	0	1	X	Z
0	0	0	0	0
1	0	1	X	X
X	0	X	X	X
Z	0	X	X	X

+	0	1	X	Z
0	0	1	X	X
1	1	1	1	1
X	X	1	X	X
Z	X	1	X	X

For an AND gate,

- If one of the inputs is *0*, the output is always *0* regardless of the other input
- If one input is *1* and the other input is *X* (we do not know what the other input is), then the output is *X* (we do not know what the output is)
- If one input is *1* and the other input is *Z* (it has no logic signal), then the output is *X* (we do not know what the hardware will do).

For an OR gate,

- If one of the inputs is *1*, the output is **1** regardless of the other input
- If one input is 0 and the other input is *X* or *Z*, the output is unknown.

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

   If the circuit is built in lab, other possible causes include

4. Defective gates
5. Defective connecting wires.

Example: The function $F = AB(C'D + CD') + (A'B'(C + D)$ is realized by the circuit shown below:

When a student builds the circuit in a lab, he finds that when A = B = C = D = 1, the output F has the wrong value, and that the gate outputs are as shown in above Figure. The reason for the incorrect value of F can be determined as follows:

1. The output of gate 7 (F) is wrong, but this wrong output is consistent with the inputs to gate 7, that is, 1 + 0 = 1.Therefore, one of the inputs to gate 7 must be wrong.
2. In order for gate 7 to have the correct output (F = 0), both inputs must be 0. Therefore, the output of gate 5 is wrong. However, the output of gate 5 is consistent with its inputs because 1.1.1 = 1. Therefore, one of the inputs to gate 5 must be wrong.
3. Either the output of gate 3 is wrong, or the A or B input to gate 5 is wrong. Because $C'D + CD' = 0$, the output of gate 3 is wrong.
4. The output of gate 3 is not consistent with the outputs of gates 1 and 2 because $0 + 0 \neq 1$. Therefore, either one of the inputs to gate 3 is connected wrong, or gate 3 is defective, or one of the input connections to gate 3 is defective.

Problem: *Complete the timing diagram for the given circuit. Assume that both gates have a propagation delay of 5 ns.*

Solution:

Problem: *Draw the timing diagram for V and Z for the circuit. Assume that the AND gate has a delay of 10 ns and the OR gate has a delay of 5 ns.*

Solution:

Problem: *Complete the timing diagram for the given circuit. Assume that both gates have a propagation delay of 5 ns.*

Solution:

Problem: *Consider the 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 and OR Gates. Identify two hazards in each circuit.*

(b) *Find an AND-OR circuit for F which has no hazard.*

(c) *Find an OR-AND circuit for F which has no hazard.*

Solution: Given, F (A, B, C, D) = ∑m (0, 4, 5, 10, 11, 13, 14, 15)

(a) K-Map1:

F1	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$	1	1		
$\bar{C}D$		1	1	
CD			1	1
$C\bar{D}$			1	1

F1 =

Hazards: adjacent 1's are not covered by the same loop

1] A'BC'

2] ABD

K-Map2:

F2	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}\bar{D}$			0	0
$\bar{C}D$	0			0
CD	0	0		
$C\bar{D}$	0	0		

F2 =

Hazards: adjacent 0's are not covered by the same loop

1] (A + B + D')

2] (A' + B + C')

(b) & (c) AND-OR & OR-AND circuit for F which has no hazard.

K-Map (AND-OR):

K-Map (OR-AND):

F1 =

Hazard free AND-OR Network:

F2 =

Hazard Free OR-AND Network:

Problem: *For the following circuit:*

(a) *Assume that the inverters have a delay of 1 ns and the other gates have a delay of 2 ns. Initially, A = 0 and B = C = D = 1, and C changes to 0 at time = 2 ns. Draw a timing diagram and identify the transient that occurs.*

(b) *Modify the circuit to eliminate the hazard.*

Solution: (a) Timing Diagram:

(b) Expression for $G\ (A, B, C, D) = BC + A'C'D$

K-Map (AND-OR):

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

G =

Hazard free Circuit Diagram:

Problem: *For the following circuit:*

(a) *Assume the inverters have a delay of 1 ns and the other gates have a delay of 2 ns. Initially A = B = 0 and C = D = 1; C changes to 0 at time 2 ns. Draw a timing diagram showing the glitch corresponding to the hazard.*

(b) *Modify the circuit so that it is hazard free. (Leave the circuit as a two-level, OR-AND circuit.).*

Solution: (a) Timing Diagram:

(b) Expression for $H\ (A, B, C, D) = (A' + C)(C + D')(B + C')$

K-Map (OR-AND):

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

H =

Hazard free Circuit Diagram:

Homework:

1] F (A, B, C, D) = ∑ m (0, 2, 3, 5, 6, 7, 8, 9, 13, 15).

(a) Find three different minimum AND-OR circuits that implement F. Identify two hazards in each circuit. Then find an AND-OR circuit for F that has no hazards.

(b) There are two minimum OR-AND circuits for F; each has one hazard. Identify the hazard in each circuit, and then find an OR-AND circuit for F that has no hazards.

2] Consider the following logic function: F(A, B, C, D) = ∑ m(0, 2, 5, 6, 7, 8, 9, 12, 13, 15).

(a) Find two different minimum AND-OR circuits which implement F. Identify two hazards in each circuit. Then find an AND-OR circuit for F that has no hazards.

(b) The minimum OR-AND circuit for F has one hazard. Identify it, and then find an OR-AND circuit for F that has no hazards.

MULTIPLEXERS, DECODERS, AND PROGRAMMABLE LOGIC DEVICES

MULTIPLEXERS:

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 following Figure shows a 2-to-1 multiplexer.

When the control input A is 0, the switch is in the upper position and the MUX output is Z = I_0; when A is 1, the switch is in the lower position and the MUX output is Z = I_1. In other words, a MUX acts like a switch that selects one of the data inputs (I_0 or I_1) and transmits it to the output. The logic equation for the 2-to-1 MUX is therefore: $Z = A' I_0 + AI_1$

The following Figure shows diagrams for a 4-to-1 multiplexer, 8-to-1 multiplexer, and 2^n-to-1 multiplexer.

The 4-to-1 MUX acts like a four-position switch that transmits one of the four inputs to the output. Two control inputs (A and B) are needed to select one of the four inputs. If the control inputs are AB = 00, the output is I_0; similarly, the control inputs 01, 10, and 11 give outputs of I_1, I_2, and I_3, respectively. The 4-to-1 multiplexer is described by the equation: $Z = A' B' I_0 + A' B I_1 + AB' I_2 + AB I_3$

Similarly, the 8-to-1 MUX selects one of eight data inputs using three control inputs.

It is described by the equation: $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$. Multiplexers can also have an additional input called an *enable* input.

If the OR gate in the above Figure is replaced by a NOR gate, then the 8-to-1 MUX inverts the selected input. To distinguish between these two types of multiplexers, we will say that the multiplexers without the inversion have *active high* outputs, and the multiplexers with the inversion have *active low* outputs.

In general, a multiplexer with *n* control inputs can be used to select any one of 2^n data inputs. The general equation for the output of a MUX with *n* control inputs and 2^n data inputs is:

$$Z = \sum_{k=0}^{2^n-1} m_k I_k$$

Where m_k is a minterm of the *n* control variables and I_k is the corresponding data input.

Multiplexers are frequently used in digital system design to select the data which is to be processed or stored.

The following Figure shows how a quadruple 2-to-1 MUX is used to select one of two 4-bit data words. If the control A = 0, the values of x0, x1, x2,and x3 will appear at the z0, z1, z2, and z3 outputs; if A = 1, the values of y0, y1, y2, and y3 will appear at the outputs.

Multiplexer Logic: A digital design usually begins with a truth table. The problem is to come up with a logic circuit that has the same truth table. We have two standard methods for implementing a truth table – the SOP and the POS solution. The third method is the *multiplexer solution*.

Problem: *Implement Y (A, B, C, D) = ∑m (0, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 15) using 16-to-1 multiplexer (IC 74150) & 8-to-1 multiplexer.*

Solution: Notice that, output is an active low signal in IC 74150.

A	B	C	D	Y	8-to-1 MUX Data Inputs
0	0	0	0	1	$\overline{D}$
0	0	0	1	0	
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	0	0
0	1	1	1	0	
1	0	0	0	1	1
1	0	0	1	1	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	0	D
1	1	1	1	1	

We follow a procedure that is similar to the one that we adopted in Entered Variable Map method to implement Y using 8-to-1 MUX.

A	B	C	8-to-1 MUX Data Inputs
0	0	0	$\overline{D}$
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	D

Problem: *Design a 32-to-1 multiplexer using two 16-to-1 multiplexer and one 2-to-1 multiplexer.*

Solution: The circuit diagram is shown in the following Fig. A 32-to-1 multiplexer required 5 ($\log_2 32$) select lines (say, ABCDE). The lower four 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 the 5th select line (A).

Problem: *Realize* $Y = \bar{A}B + B\bar{C} + ABC$ *using an 8-to-1 multiplexer. Also, realize the same with a 4-to-1 multiplexer.*

Solution: Given,

$$Y = \bar{A}B + \bar{B}\bar{C} + ABC$$

$$Y = \bar{A}B(\bar{C} + C) + \bar{B}\bar{C}(\bar{A} + A) + ABC$$

$$Y = \bar{A}B\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Y = \sum m\,(0, 2, 3, 4, 7).$$

Hence, to generate the given logic function, using *8-to-1 multiplexer*, we find $D_0 = D_2 = D_3 = D_4 = D_7 = 1$ and $D_1 = D_5 = D_6 = 0$.

$$Y = \bar{A}B\bar{C} + \bar{A}BC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + ABC$$

$$Y = \bar{A}\bar{B}(\bar{C}) + \bar{A}B(\bar{C}) + \bar{A}B(C) + A\bar{B}(\bar{C}) + AB(C)$$

Hence, for a *4-to-1 multiplexer*,

we find $D_0 = C'$, $D_1 = 1$, $D_2 = C'$, and $D_3 = C$ generates the given function.

Alternate Method:

A	B	C	8-to-1 MUX Data Inputs	4-to-1 MUX Data Inputs
0	0	0	$1 = D_0$	$\bar{C} = D_0$
0	0	1	$0 = D_1$	
0	1	0	$1 = D_2$	$1 = D_1$
0	1	1	$1 = D_3$	
1	0	0	$1 = D_4$	$\bar{C} = D_2$
1	0	1	$0 = D_5$	
1	1	0	$0 = D_6$	$C = D_3$
1	1	1	$1 = D_7$	

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. The following Figure shows a buffer connected between a gate output and several gate inputs. Because no bubble is present at the buffer output, this is a non-inverting buffer, and the logic values of the buffer input and output are the same, that is, $F = C$.

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. Use of three-state logic permits the outputs of two or more gates or other logic devices to be connected together. The following Figure shows a three-state buffer and its logical equivalent.

When the enable input B is 1, the output C equals A; when B is 0, the output C acts like an open circuit. In other words, when B is 0, the output C is effectively disconnected from the buffer output so that no current can flow. This is often referred to as a Hi-Z (high-impedance) state of the output because the circuit offers a very high resistance or impedance to the flow of current. Three-state buffers are also called *tri-state buffers*.

The following Figure shows the truth tables for four types of three-state buffers.

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

(a)

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

(b)

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

(c)

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

(d)

In Figures (a) and (b), the enable input B is not inverted, so the buffer output is enabled when B = 1 and disabled when B = 0. That is, the 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.

In Figure (b), the buffer output is inverted so that *C = A'* when the buffer is enabled.

The buffers in Figures (c) and (d) operate the same as in (a) and (b) except that the enable input is inverted, so the buffer is enabled when B = 0.

In the following 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.

When we connect two three-state buffer outputs together, as shown in the following Figure, if one of the buffers is disabled (output = Z), the combined output F is the same as the other buffer output. If both buffers are disabled, the output is Z. If both buffers are enabled, a conflict can occur. If A = 0 and C = 1, we do not know what the hardware will do, so the F output is unknown (X). If one of the buffer inputs is unknown, the F output will also be unknown. The table in the following Figure summarizes the operation of the circuit. S1 and S2 represent the outputs the two buffers would have if they were not connected together. When a bus is driven by three-state buffers, we call it a three-state bus. The signals on this bus can have values of 0, 1, Z, and perhaps X.

	S2			
S1	X	0	1	Z
X	X	X	X	X
0	X	0	X	0
1	X	X	1	1
Z	X	0	1	Z

A multiplexer may be used to select one of several sources to drive a device input. For example, if an adder input must come from four different sources; a 4-to-1 MUX may be used to select one of the four sources. An alternative is to set up a three-state bus, using three-state buffers to select one of the sources (see the following Figure). In this circuit, each buffer symbol actually represents four three-state buffers that have a common enable signal.

Integrated circuits are often designed using bi-directional pins for input and output. Bi-directional means that the same pin can be used as an input pin and as an output pin, but not both at the same time. To accomplish this, the circuit output is connected to the pin through a three-state buffer, as shown in the following Figure. When the buffer is enabled, the pin is driven with the output signal. When the buffer is disabled, an external source can drive the input pin.

DECODERS AND ENCODERS:

The ***decoder*** is another commonly used type of integrated circuit. The following 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. Exactly one of the output lines will be 1 for each combination of the values of the input variables.

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

The following 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. When a binary-coded-decimal digit is used as an input to this decoder, one of the output lines will go low to indicate which of the 10 decimal digits is present.

(a) Logic diagram

(b) Block diagram

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

(c) Truth Table

In general, an n-to-2^n line decoder generates all 2^n minterms (or maxterms) of the n input variables. The outputs are defined by the equations:

$$y_i = m_i, \qquad i = 0 \text{ to } 2^n - 1 \text{ (non-inverted outputs)}$$

or

$$y_i = m_i' = M_i, \qquad i = 0 \text{ to } 2^n - 1 \text{ (inverted outputs)}$$

where m_i is a minterm of the n input variables and Mi is a maxterm.

Example: *Realize the following functions using a 4-to-10 decoder.*

f1 (a, b, c, d) = m1 + m2+ m4 and f2 (a, b, c, d) = m4 + m7 +m9

Solution: An n-input decoder generates all of the minterms of n variables. Hence, n-variable functions can be realized by ORing together selected minterm outputs from a decoder.

Rewriting given f1 and f2; we have: f1 = (m1' m2'm4')' and f2 = (m4' m7'm9')'

Now, f1 and f2 can be generated using NAND gates, as shown in the following Figure.

Problem: *Show how using a 3-to8 decoder and multi-input OR gates following Boolean expression can be realized simultaneously.*

$F_1 (A, B, C) = \sum m (0, 4, 6)$ $F_2 (A, B, C) = \sum m (0, 5)$ $F_3 (A, B, C) = \sum m (1, 2, 3, 7)$.

Solution: Since, at the decoder output, we get all the min-terms, we use them as shown in the following Fig, to get the required Boolean expression:

Seven-Segment Decoders: The following Fig shows a *seven-segment indicator*, i.e. seven LEDs labeled *a* through *g* (actually, eight LEDs labeled through *a* through *h*). By forward biasing the LEDs, we can display the digits 0 through 9. For example, to display the digit 0, we need to light-up the segments *a*, *b*, *c*, *d*, *e*, and *f*. Similarly, to light-up the digit 5, we need segments *a*, *c*, *d*, *f*, and *g*.

Seven-Segment Indicator: Common Anode Type & Common Cathode Type

Seven-segment indicators may be *common-anode* type; where all anodes are connected together (as shown above) or *common-cathode* type; where all cathodes are connected together (as shown above).

The 7446 & The 7448: A *seven-segment decoder-driver* is an IC decoder that can be used to drive a seven-segment indicator. There are two types of decoder-drivers, corresponding to common-anode (IC 7446) and common cathode (IC 7448) indicators. Each decoder driver has 4 input pins (the BCD input) and 7 output pins (*a* through *h* segments), as shown in the following Fig.

7446 Decoder-driver & 7448 Decoder-driver

The logic circuits inside 7446 / 7448 convert the BCD input to the required output. For Example, if the BCD input is 0111, the internal logic of the 7446 / 7448 will force segments *a*, *b*, and *c* to conduct. As a result, digit 7 will appear on the seven-segment indicator.

An ***encoder*** (*converts an active input signal to a coded output signal*) performs the inverse function of a decoder. The following Figure shows a *8-to-3 priority encoder* with inputs y0 through y7. 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 y3 = 1, then abc = 011.

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

If more than one input is 1 at the same time, the output can be defined using a priority scheme. The truth table in the above Figure uses the following scheme: If more than one input is 1, the highest numbered input determines the output. For example, if inputs y1, y4, and y5 are 1, the output is abc = 101.The X's in the table are don't-cares; for example, if y5 is 1, we do not care what inputs y0 through y4 are. 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 y0 is 1.

PROGRAMMABLE LOGIC DEVICES (PLDs):

A *programmable logic device* (or *PLD*) is a general name for a digital integrated circuit capable of being programmed to provide a variety of different logic functions. *PLD*s are electronic components, used to build reconfigurable digital circuits. Programmable Read Only Memory (PROM), Programmable Array Logic (PAL), and Programmable Logic Array (PLA) are included in the general classification.

General Classification of PLDs

Programmable Logic Arrays (PLA):

A *PLA* with *n* inputs and *m* outputs (see the following Figure) can realize *m* functions of *n* variables. In PLA, the product terms of the input variables is realized by an AND array; and the OR array ORs together the product terms needed to form the output functions. Hence, a PLA implements a sum-of-products expression.

Example: *Realize the following functions using PLA:*

$F0 = \sum m\ (0, 1, 4, 6) = A'B' + AC'$ $\qquad$ $F1 = \sum m\ (2, 3, 4, 6, 7) = B + AC'$

$F2 = \sum m\ (0, 1, 2, 6) = A'B' + BC'$ $\qquad$ $F3 = \sum m\ (2, 3, 5, 6, 7) = AC + B$

Solution: The following Figure shows a PLA which realizes the said functions.

Product terms are formed in the AND array by connecting switching elements at appropriate points in the array. For example, to form A'B', switching elements are used to connect the first word line with the A' and B' lines.

Switching elements are connected in the OR array to select the product terms needed for the output functions. For example, because F0 = A'B' + AC', switching elements are used to connect the A'B' and AC' lines to the F0 line.

The connections in the AND and OR arrays of this PLA make it equivalent to the AND-OR array shown in the following Figure.

The contents of a PLA can be specified by a PLA table. The following Table specifies the PLA shown in the above Figure. The input side of the table specifies the product terms. The symbols 0, l, and – indicate whether a variable is complemented, not complemented, or not present in the corresponding product term. The output side of the table specifies which product terms appear in each output function. A 1 or 0 indicates whether a given product term is present or not present in the corresponding output function. Thus, the first row of Table indicates that the term A'B' is present in output functions F0 and F2, and the second row indicates that AC' is present in F0 and F1.

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

Example: *Realize the following functions using PLA:*

$$f1 = a'bd + abd + ab'c' + b'c$$

$$f2 = c + a'bd$$

$$f3 = bc + ab'c' \; abd$$

Solution: Based on the given expressions, we can construct a PLA table (see Figure (a)), with one row for each distinct product term. Figure (b) shows the corresponding PLA structure, which has four inputs, six product terms, and three outputs. A dot at the intersection of a word line and an input or output line indicates the presence of a switching element in the array.

a b c d	f_1 f_2 f_3
0 1 – 1	1 1 0
1 1 – 1	1 0 1
1 0 0 –	1 0 1
– 0 1 –	1 0 0
– – 1 –	0 1 0
– 1 1 –	0 0 1

(a) PLA table

(b) PLA structure

NOTE: Mask-programmable and field-programmable PLAs are available. The mask-programmable type is programmed at the time of manufacture. The field-programmable logic array (FPLA) has programmable interconnection points that use electronic charges to store a pattern in the AND and OR arrays.

Problem: *Design a PLA to recognize each of the 10 decimal digits represented in binary form and to correctly drive a 7-segment display.*

Solution: The PLA must have 4 inputs, as shown in the following Fig. Four bits are required to represent the 10 decimal numbers. There must be 7 outputs (abcdefg).

The circuit given in the following Figure shows the links after programming. The input AND-gate array is programmed such that, each AND gate decodes one of the decimal numbers. Then, links are removed from the output OR-gate array, such that proper segments of the indicator are illuminated. For example, when ABCD = LHLH, segments *afgcd* are illuminated to display the decimal number 5.

Programmable Array Logic (PAL):

A *PAL* is a special case of the programmable logic array (PLA) in which the AND array is programmable and the OR array is fixed. The following Figure represents a segment of an un-programmed PAL.

Consider the PAL segment of the following Figure (a), used to realize the function $I_1 I'_2 + I'_1 I_2$. The X's in the following Figure (b) indicate that I_1 *and* I'_2 lines are connected to the first AND gate, and the I'_1 *and* I_2 lines are connected to the other gate.

(a) Unprogrammed (b) Programmed

Example: *Implement Full Adder using PAL.*

Solution: The logic equations for the full adder are:

$$Sum = X'Y'Cin + X'YC'in + XY'C'in + XYCin$$

$$Cout = XY + YCin + XCin$$

The following Figure shows PAL where each OR gate is driven by four AND gates. The X's on the diagram show the connections that are programmed into the PAL to implement the full adder equations.

Problem: *Generate the following Boolean function using PAL.*

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

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

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

$$Y_0 = ABCD$$

Solution: Start with first equation. The first desired product is A'BC'D, which is marked as shown in the following Figure. The fixed OR connections on the output side imply that the first OR gate produces an output of first equation.

Homework:

1] Realize a full adder using a 3-to-8 line decoder and

(a) two OR gates, (b) two NOR gates.

2] Derive the logic equations for a 4-to-2 priority encoder.

3] Implement the following equations using PLA –

$$X = AB'D + AC' + BC + C'D'$$

$$Y = A'C' + AC + C'D'$$

$$Z = CD + A'C' + AB'D$$

4] Implement a full subtracter using a PAL.

5] Use a 4-to-1 multiplexer and a minimum number of external gates to realize the function

F(w, x, y, z) = ∑m (3, 4, 5, 7, 10, 14) + ∑d (1, 6, 15).

Try these –

1] Show how

(a) two 2-to-1 multiplexers (with no added gates) could be connected toform a 3-to-1 MUX. Input selection should be as follows:

If AB = 00, select I_0

If AB = 01, select I_1

If AB = 1– (B is a don't-care), select I_2

(b) two 4-to-1 and one 2-to-1 multiplexers could be connected to form an 8-to-1 MUX with three control inputs.

(c) four 2-to-1 and one 4-to-1 multiplexers could be connected to form an 8-to-1 MUX with three control inputs.

(d) to implement a 32-to-1 multiplexer using two 16-to-1 multiplexers and a 2-to-1 multiplexer.

2] Implement the function $R = ab'h' + bch' + eg'h + fgh$ *using –*

(a) 2-to-1 MUXes

(b) only tri-state buffers.

3] Implement a full adder

(a) using two 8-to-1 MUXes. Connect X, Y, and Cin to the control inputs of the MUXes and connect 1 or 0 to each data input.

(b) using two 4-to-1 MUXes and one inverter. Connect X and Y to the control inputs of the MUXes, and connect 1's, 0's, Cin, or Cin to each data input.

Flip-Flop Timing:

Diodes and transistors cannot switch states immediately. It always takes a small amount of time to turn a diode on or off. It takes time for a transistor to switch from saturation to cutoff and vice versa.

Switching time is the main cause of propagation delay, designated t_p. This represents the amount of time it takes for the output of a gate or flip-flop to change states after the input changes.

For example if the data sheet of an edge triggered D flip-flop lists $t_p = 10ns$, it takes about 10ns for Q to change state after D has been sampled by the clock edge.

The propagation delay time is used to construct the pulse forming circuit. When flip-flops are used to construct counters, the propagation delay is small & ignored.

Stray capacitance at the D input makes it necessary for data D to be at the input before the clock edge arrives. The setup time t_{setup} is the minimum amount of time that the data bit must be present before the clock edge hits. For example, if a D flip-flop has a setup of time of 15ns, the data bit to be stored must be at the D input at least 15ns before the clock edge arrives.

Available At: VTU HUB (Android App)

Scanned by CamScanner

Hold time thold is the minimum amount of time that data bit D must be present after the PT of the clock. For example if tsetup = 15ns and thold = 5ns the data bit has to be at the D input at least 15ns before the clock edge arrives and held at least 5ns after the clock PT.

JK Master - Slave Flip-flops.

Master Slave Flip-flop

C	J	K	Q_{n+1}	Action
↑	L	L	Q_n	No Change
↑	L	H	L	RESET
↑	H	L	H	SET
↑	H	H	$\bar{Q}_n$	Toggle.

Truth table

Symbol.

Master is positive edge triggered and the slave is negative edge triggered. Therefore the master responds to its J & K inputs before the slave.

1) If J=1 and K=D the master sets on the positive clock

Available At: VTU HUB (Android App)
Scanned by CamScanner

transition. The high Q output of the master drives the J input of the slave, so on the negative clock transition the slave sets, copying the action of the master.

2) If J=0 & K=1 the master resets on the PT of the clock. The high $\bar{Q}$ output of the master goes to the K input of the slave. Therefore, the NT of the clock forces the slave to reset. Again, the slave has copied the master.

3) If the master's J & K inputs are both high, it toggles on the PT of the clock and the slave then toggles on the clock NT. Hence slave copies the master.

4) If J=K=0 the flip-flop is disabled and Q remains unchanged

The master is set according to J and K while the clock is high, the contents of the master are then shifted into the slave when the clock goes low. This flip-flop can be ~~referred~~ referred as pulse triggered flip-flop.

Race around condition can be avoided by using a Master-Slave JK flip-flop.

Switch Contact Bounce Circuit:

In some occasion digital system uses mechanical contacts for the purpose of conveying an electrical signal. Example: switches used on the keyboard of a computer system.

In each case, the intent is to apply a high logic level (+5Vdc) or a low level (0Vdc). The single pole single throw switch is one such example (fig A)

Available At: VTU HUB (Android App)

Scanned by CamScanner

fig A

fig. (B) Ideal voltage at A

fig (C) Voltage at A showing contact bounce

When the switch is open, the voltage at point A is +5 Vdc, when switch is closed the voltage at point A is 0Vdc. Ideally the voltage waveform at A should appear as shown in fig (B) as the switch is moved from open to close or vice versa. But actually, the waveform at point A will appear more or less as shown in fig (C) as result of phenomenon known as ~~contact~~ contact bounce. Any mechanical switching device consists of a moving contact arm restrained by some sort of spring system. As a result, when the arm is moved from one stable position to the other, the arm bounces like hard ball bounce when dropped on the hard surface. The number of bounces that occur & the period of bounce differ from each switching device.

Note: Even though actual physical bounce occurs each time the switch is opened or closed, contact bounce appears in the voltage level at A only when the switch is closed.

A simple RS Latch Debounce Circuit:

The RS latch in below fig will remove any contact bounce due to switch. The output Q is used to generate

Available At: VTU HUB (Android App)
Scanned by CamScanner

the desired switch signal.

Switch contact bounce eliminator

When the switch is moved to position H, R = 0 and S = 1 bouncing occurs at S input due to switch. The flip-flop sees this as a series of high and low inputs, settling with a high level. The flip-flop will immediately be set with Q = 1 at the first high level on S. When the switch bounces the input signals are R = S = 0 therefore the flip-flop remains set. When the switch regains contact, R = 0 and S = 1, this set the flip-flop.

When the switch is moved to position L, S = 0 and R = 1, Bouncing occurs at the R input due to the switch. Again the flip-flop 'sees' this as a series of high & low inputs. It simply responds to the first high level and ignores all following transition. The result is a clean high to low signal at the flip-flop output.

Various Representations Of Flip-flops:

There are various ways a flip-flop can be represented each one suitable for certain application.

Available At: VTU HUB (Android App)
Scanned by CamScanner

Characteristic Equations of Flip-flops

The characteristic equations of flip-flops are useful in analyzing circuits made of them. Here next output Q_{n+1} is expressed as a function of present output Q_n and input to flip-flops.
K map can be used to get the optimized expression and truth table of each flip-flop is mapped into it.

analyzing circuits

RS flip-flop

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

	$\bar{S}\bar{R}$ 00	$\bar{S}R$ 01	SR 11	$S\bar{R}$ 10
$\bar{Q}_n$ 0	0	0	X	1
Q_n 1	1	0	X	1

$$Q_{n+1} = S + \bar{R}Q_n$$

D flip-flop

	$\bar{D}$ 0	D 1
$\bar{Q}_n$ 0	0	1
Q_n 1	0	1

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

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

JK flip-flop

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

	$\bar{J}\bar{K}$ 00	$\bar{J}K$ 01	JK 11	$J\bar{K}$ 10
$\bar{Q}_n$ 0	0	0	1	1
Q_n 1	1	0	0	1

$$Q_{n+1} = Q_n\bar{K} + \bar{Q}_nJ$$

T flip-flop

	$\bar{T}$ 0	T 1
$\bar{Q}_n$ 0	0	1
Q 1	1	0

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

$$T = T\bar{Q}_n + \bar{T}Q_n$$
$$= T \oplus Q_n$$

Available At: VTU HUB (Android App)

Scanned by CamScanner

Summary :

SR Flip-flop $Q_{n+1} = S + \bar{R}Q_n$

JK Flip-flop $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

D Flip-flop $Q_{n+1} = D$

T Flip-flop $Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n = T \oplus Q_n$

Flip-flop as Finite State Machine :

progress of sequential logic with time

In a sequential logic circuit the value of all the memory elements at a given time define the state of that circuit at that time. Finite State Machine (FSM) concept offers a better alternative to truth table in understanding progress of sequential logic with time

SR flip-flop

D flip-flop

JK flip-flop

T flip-flop.

Let us see how state transition diagram for SR flip-flop is developed from truth table. [illegible] state of flip-flop can be 0 or 1. If current state is 0 and SR = 00, flip-flop will remain same state. If current state is 0 and SR = 01, next state will be 0. If current state

Available At: VTU HUB (Android App)

Scanned by CamScanner

If current state is 0 and SR = 10 next state will be 1.
If current state is 1 and SR = 00 next state will be 1.
If current state is 1 and SR = 10 next state will be 1
If current state is 1 and SR = 01 next state will be 0.

Flip-flop Excitation Table.

truth table analysis prob.

Here flip-flop input is presented as a dependent function of transition $Q_n \rightarrow Q_{n+1}$ and comes later in the table. This is derived from flip-flop truth table or characteristic equation but more directly from its state transition diagram.

Q_n	Q_{n+1}	S	R	J	K	D	T
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	0	1
1	1	X	0	X	0	1	0

From SR flip-flop state transition diagram, if present state is 0 application of SR = 0X does not alter its value. State 0 to 1 transition occurs when SR = 10 is present at the input side while state 1 to 0 transition occurs if SR = 01. Present is maintained if SR = X0.

HDL implementation of Flip-flop

Behavioral model is preferred for sequential circuit and 'always' keyword is used in all those circuits.

Available At: VTU HUB (Android App)
Scanned by CamScanner

Enabled D Flip-flop.

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

```
module ENDFF(D, EN, Q);
input D, EN;
output Q;
reg Q;
always @ (EN or D)
if (EN) Q=D;
endmodule
```

Enabled RS flip-flop:

```
module SRLatch (S, R, En, Q)
input S, R, En;
Output Q;
reg Q;
always@(En or S or R)
    if (En) Q= S| (~R & Q);
endmodule
```

$Q = S +$

We use keyword posedge and negedge for positive edge and negative edge respectively.

positive edge triggered D

```
module DFFpos (D, C, Q)
input D, C;
output Q;
reg Q;
always @ (posedge C)
Q=D;
endmodule
```

negative edge triggered D

```
module DFFneg (D, C, Q)
input D, C;
output Q;
reg Q;
always @ (negedge C)
Q=D;
endmodule.
```

Available At: VTU HUB (Android App)

Scanned by CamScanner

D flip-flop with positive edge triggered and clear input:

```
module DFFpos_clr (D, C, CLR, Q);
input D, C, CLR;
output Q;
reg Q;
always @ (posedge or negedge CLR)
    if (~CLR) Q = 1'b0; // Q stores 0
    else Q = D;
endmodule
```

positive edge triggered SR flip flop

```
module SRFFpos (S, R, C, Q)
input S, R, C;
output Q;
reg Q;
always @ ( posedge C)
    Q = S | (~R & Q)
endmodule
```

Available At: VTU HUB (Android App)

Scanned by CamScanner

REGISTERS

A register is a very important digital building block.
A data register is used to store (momentarily) binary information appearing at the output of an encoding matrix

Registers are groups of flip-flops where each flip-flop is capable of storing one bit of information. An n bit register is a group of n flip-flops. The basic function of a register is to hold information in a digital system and make it available to the logic elements for the computing process. Two basic functions of register is data storage & data movement.
Register is capable of storing finite number 0-1 combination in finite number of flip-flops.

Types of Registers:

A register used to store an 8 bit binary number, then it has 8 flip-flops.

A shift register is a register that allows each of the flip flop to pass the stored information to its adjacent neighbour.

The bits in a binary number can be moved from one place to another in either of 2 ways.

Shifting the data 1 bit at a time in serial fashion i.e beginning with either the MSB or LSB is called serial shifting.

Available At: VTU HUB (Android App)
Scanned by CamScanner

Shifting of all the data bits simultaneously is called parallel shifting.

There are 2 ways to shift data into register and 2 ways to shift data out of the register. So there are 4 basic types of register (shift registers).

1) Serial in serial out : 54/74LS91 (8 bit)

2) Serial in parallel out : 54/74164 (8bits)

3) Parallel in serial out : 54/74165, 8 bits

4) Parallel in Parallel out : 54/74198, 8 bits

Available At: VTU HUB (Android App)

Scanned by CamScanner

Serial IN Serial Out:

Usually edge triggered JK, SR or D type flip-flops are used to construct registers.

In Serial in Serial out registers data is serially entered or exited from a shift register.

Consider 4 D flip-flop connected as shown below to form a 4 bit shift register:

A common clock provides trigger at its negative edge to all the flip-flops. As output of one D flip-flop connected to input of the next at every clock trigger data stored in one flip-flop is transferred to the next. For this circuit transfer takes place like this $Q \to R$, $R \to S$, $S \to T$ and serial data input is transferred to Q.

Assume all the flip-flop are initially cleared. Serial data to be inputed is 0100.

At clock edge A, flip-flop Q has input 0 from serial data in D flip-flop R has input 0 from output of Q, flip-flop S has input 0 from the output of R and flip-flop T has input 0 from outputs when clock triggers, these inputs get transferred to corresponding flip-flop outputs simultaneously so that QRST = 0000. Thus at clock trigger values at DQRS is transferred

Available At: VTU HUB (Android App)

Scanned by CamScanner

to QRST

At clock edge B : serial data in = 0, i.e DQRS = 0000, So after NT at B QRST = 0000

At clock edge C : serial data in = 1, i.e DQRS = 1000 and after NT QRST = 1000.

At clock edge D : serial data in = 0, i.e DQRS = 0100 and after NT QRST = 0100

Clock	Serial Input	Q	R	S	T
0	0	0	0	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	0	1	0	0	0
4	0	0	1	0	0

Suppose that register has the 4 bit number QRST = 1010 stored in it.

Before Time A : The register stores the number QRST = 1010 and T = 0 (appear at output)

At time A : The entire number is shifted one flip-flop to the right. The register holds the bits QRST = 0101 and T = 0

At time B : The register holds QRST = 0010 and T = 0 (appears at output)

At time C : The register holds QRST = 0001 and T = 1 comes out

At time D : The register holds QRST = 0000; MSB is shifted out the right end last.

We have caused the number stored in regr to appear at T

Available At: VTU HUB (Android App)

Scanned by CamScanner

1 bit at a time beginning with LSB, So stored data is shifted serially.

Clock	Q	R	S	T	Serial out
0	1	0	1	0	
1	0	1	0	1	0
2	0	0	1	0	1
3	0	0	0	1	0
4	0	0	0	0	1

A shift register can be implemented using JK or SR flip-flops also. [Q is connected J next flip-flop & $\bar{Q}$ is connected to K of next FF]

74LS91 is 8 bit shift register using 8 RS flip-flop as shown below:

Each flip-flop is negative-edge-triggered, clock is passed through an inverter, data will be shifted on positive edges. The inverter connected b/w R & S on the 1st flip-flop means that circuit functions as a D type flip-flop. Data passed at (A or B) get complemented at NAND gate & NOT gate [A=1 => S=1 or A=0 => S=0]

Available At: VTU HUB (Android App)
Scanned by CamScanner

When Data input is 1 then S=1 and R=0, flip-flop is set when clock goes high

When data input is 0 then S=0 and R=1, flip-flop is reset when clock goes high.

Serial In - Parallel Out:

Data is shifted in serially and shifted out parallelly. To shift the data out in parallel, all the data bits should be available at outputs at the same time; we can achive this by connecting output of each flip to the output pin.

The 54/74164 is an 8 bit serial input - parallel output shift register. It is constructed by using RS flip-flops having clock negative edge triggered.

74LS91 pin diagram

54/74164 pin diagram.

In 54/74164 output of each flip-flop is available simultaneously and each flipflop has an asynchronous clear input. Low to

Available At: VTU HUB (Android App)
Scanned by CamScanner

To clear will reset every flip-flop. When clear is high all flip-flop work normally

Shifting data into register in a serial fashion is exactly the same as previous 74LS91

R S	Q_n
0 0	Q_n
0 1	1
1 0	0

A B	NAND
0 0	1
0 1	1
1 0	1
1 1	0

A 0 1 } Q=D=A
B 1 0

A 1 0 } Q=1=A
B 1 +

Data are shifted into the register on PTs. (becoz of NOT gate)

B is held High: NAND gate is enabled and the serial input data passes through the NAND gate inverted. The input data is shifted into the register.

B is held Low: The NAND gate is forced high, the input data stream is inhibited.

When clear is low all flip-flop value is reset to 0.

Data input = 00101100

* Clock Begins at time B, no effect on Flip-flop data becoz control line is low
* First data bit '0' is shifted into register at time D.
* Next 7 data bits are shifted in order at time E, F, G, H, I, J, and K respectively.
* After K clock remains high 8 bit number 00101100 is available on output lines

Available At: VTU HUB (Android App)
Scanned by CamScanner

Parallel In Serial Out :

Parallel entry of data into the register an serial removal of data from the register.

54/74166 is commercially available PISO TTL device. This is an 8 bit shift register capable of either serial/parallel data entry and serial data output. It is controlled using eight negative edge triggered RS flip-flop.

Available At: VTU HUB (Android App)

Scanned by CamScanner

Available At: VTU HUB (Android App)

Scanned by CamScanner

In RS flip-flop R input is inverted and given to S so this becomes a D-type flip-flop.

X = 1

if $\bar{X} = 0$, then S = 1 and R = 0

1 will be clocked (inputted) to flip-flop

If X = 1, the 1 is clocked into the flip-flop. NOR gate differs to take data from a different source.

i.e. if X_1 is grounded X_2 is shifted in or if X_2 is grounded X_1 is shifted in.

R	S	AND	NOR	RS
0	0	0	1	Q_n
0	1	0	0	1
1	0	0	0	0
1	1	1	0	F

A control input is introduced by adding 2 AND gates and 2 NOT gates as shown in above fig.

When Control is high: Data bit at X_1 will be shifted into flip-flop at next clock transition

When Control is low: Data bit at X_2 will be shifted into flip-flop at next clock transition.

They are connected to allow 2 different operations

1) parallel entry of data 2) serially shifting of data through Q_n to Q_n

Available At: VTU HUB (Android App)

Scanned by CamScanner

Shift / Load is Low: A single clock transition loads 8 bits of data (ABCD EFGH) into the register in parallel (i.e lower AND gates are enabled)

Shift / Load is High: Clock transition will shift data through the register serially, with entering data applied at the SERIAL OUT.

When clear is low all flip-flops in register value is cleared to 0.

Clock is applied through a NOR gate (2 input) when clock inhibit is low, the clock signal pause through NOR gate inverted, hence data will shift into register on the PTs (flip-flops are NTs but NOT of NT is PT)

When clock inhibit is high NOR gate output is low, clock is prevented from reaching flip-flop, the register hold its contents.

C	C1	
0	0	1
1	0	0
0	1	0
1	1	0

Parallel In Parallel Out:

74174 is an example of a parallel in - parallel out register. In 74174 6 negative edge triggered D flip-flops are used. Because of NOT gate present at clock data is shifted in PTs. The 6 data bits D_1 through D_6 are all shifted into the register in parallel. The stored data is immediately available in parallel at the outputs Q_1 through Q_6. They are simply used to store data hence called data register / data latch. It is not

Available At: VTU HUB (Android App)

Scanned by CamScanner

Here it is not possible to shift stored data either to the right or left. Asynchronous clear low will clear all flip flops to low.

54/74198 is an 8 bit TTL having both parallel input and parallel output capacity. It uses positive edge triggered flip-flop. It can also be used to shift data through the register in either direction.

54/74195 is a 4 bit parallel access shift register. It also has serial data input and can be used to shift data to the right and to the left.

Available At: VTU HUB (Android App)

Scanned by CamScanner

Note: The pin numbers in parentheses correspond to the ('95A, 'LS95) ('L95), respectively.

(b) Logic diagram

[fig 1]

When the mode control line is held high, the AND gate on the right input to each NOR gate is enabled while the left AND is disabled. The data at inputs A, B, C, D will then be loaded into the -ve transition of the clock this is parallel data input.

[fig 2]

Shift-left.

When the mode control line is low, the AND gate on the right input to each NOR gate is disabled while the left AND gate is enabled. The data input to flip-flop Q_A is now at serial input; the data input to Q_B is Q_A and so down the line. On each clock NT a data bit is entered serially into the register at the 1st flip-flop Q_A and each stored data bit is shifted 1 flip-flop to the right. This is serial input of data and right shift operation [Fig 1]

In order to left shift input data must be connected as shown fig 2

Available At: VTU HUB (Android App)

Scanned by CamScanner

Universal Shift Register: A universal shift register can perform all the 4 operations and is also bidirection in nature. (4 operation are SISO, SIPO, PISO, PIPO)

4-bit universal shift register.

74194 is a 4 bit universal shift register. A, B, C, D are four parallel inputs and Q_A, Q_B, Q_C, Q_D are corresponding parallel output. There are 2 seperate inputs for serial data for left and right shift. There are 2 mode control inputs which select the mode of operation for the universal shift register according to Table.

S_1	S_2		Next State $Q_{A,n+1}$	$Q_{B,n+1}$	$Q_{C,n+1}$	$Q_{D,n+1}$
0	0	Hold	$Q_{A,n}$	$Q_{B,n}$	$Q_{C,n}$	$Q_{D,n}$
0	1	Shift right	D_{in}	$Q_{A,n}$	$Q_{B,n}$	$Q_{C,n}$
1	0	Shift left	$Q_{B,n}$	$Q_{C,n}$	$Q_{D,n}$	D_{in}
1	1	Load	A	B	C	D

The input Clear is active low and resets all the flip-flops asynchronously. Clock is positive edge triggered because of inversion.

Available At: VTU HUB (Android App)
Scanned by CamScanner

Application Of Shift Registers:

Shift register can be used to count number of pulses entering into a system as ring counter / switch tail counter

Shift register can generate a prescribed sequence repetitively or detect a particular sequence from data input.

It helps in reduction of hardware by converting parallel data feed to serial one. [eg: serial adder].

Ring Counter:

A ring counter is basically a circulating shift register in which the output of the most significant stage is fed back to the input of least significant stage.

The following is a 4 bit-ring counter constructed from D flip-flops. The output of each stage is shifted into the next stage on positive edge of a clock. If the CLEAR signal is high, all the flip-flop except the first one are reset to 0.

Advantage of ring counter over a binary counter is that it is self-decoding. i.e no extra decoding circuit is need to determine what state the counter is in.

Since the counter (4 bit) has 4 distinct states, the

Available At: VTU HUB (Android App)

Scanned by CamScanner

can be considered as a mod 4 counter. Only 4 of the max^m 16 states are used, making ring counters very inefficient:

54/74164 can be used as ring counter. If we connect the output of the last flip-flop Q_H back to the input of the 1st flip-flop, it will be converted into ring counter as show below fig.

(a) 54/74164 8-bit shift register with feedback line from Q_H to A-B

(b) Waveforms when register has a single one, and seven zeros

Suppose Q_A is high and all other flip-flops are low and then allow the clock to run. On first clock PT, the 1 in A will shift to B and A will be reset since the 0 in H will shift into A. All other flip-flops will still contain 0s. On 2nd clock PT, 1 in B will shift from B to C while B resets. Thus single 1 will shift down the register travelling from one flip-flop to the next flip-flop each time the clock goes high.

Consider the register as a tube full of ping-pong balls

Available At: VTU HUB (Android App)

Scanned by CamScanner

seven white ones and one black one. The ping-pong balls simply circulate around the register in a clockwise direction, moving ahead one flip-flop with each clock P.T. This is referred as circulating register / a ring counter.

Clock Pulse	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

Truth table for 4 bit ring counter.

repeats.

Switched tail Counter or Johnson Counter:

It is the variation of standard ring counters, with the inverted output of the last stage fed back to the input of the first stage

(a)

Clock	Serial in = T	Q	R	S	T	$Y = Q'T'$
0	1	0	0	0	0	1
1	1	1	0	0	0	0
2	1	1	1	0	0	0
3	1	1	1	1	0	0
4	0	1	1	1	1	0
5	0	0	1	1	1	0
6	0	0	0	1	1	0
7	0	0	0	0	1	0
8	1	0	0	0	0	1
9	1	1	0	0	0	0 repeats

(b)

Fig. 9.22 (a) 4-bit switched tail counter, (b) Its state table

They are also known as twisted ring counters. An n stage Johnson counter yields a count sequence of length 2n so

Available At: VTU HUB (Android App)

Scanned by CamScanner

It may be considered to be a mod-2n counter. The circuit shows a 4 bit Johnson counter. The state sequence for the counter is given in the state table

Maximum available states are not fully utilized (8 out of 16)

Both ring & Johnson counter must be initialized forcefully to a valid state.

We can use decoding gates to build

Sequence Generator and Sequence Detector

Sequence generator is useful in generating a sequence pattern repetitively. It may be the synchronizing bit pattern sent by a digital data transmitter or it may be a control word directing repetitive control task.

Sequence detector checks binary data stream and generates a signal when a particular sequence is detected.

The basic block diagram of a sequence generator is given below.

Here shift register is presented as pipe full of data and each flip-flop represents one compartment of it.

The leftmost flip-flop is connected to serial data in and

Available At: VTU HUB (Android App)

Scanned by CamScanner

rightmost provides serial data out. The clock is implied and data transfer takes only when a clock trigger arrives. The shift register is connected like a ring counter and with clock trigger binary word stored in the register comes out serially and does not get lost as it is fed back as serial in to fill the register all over again. Above sequence generator generates binary word 1011.

If we want to generate n bit long sequence, we need to store the sequence in an n bit shift register.

The block diagram of 4bit sequence detector is shown below:

Sequence to be detected

Eg: 0111 should be matched with 1011. The first bits are unmatched and corresponding Ex-NOR o/p is 1 & also final o/p Y=1

Available At: VTU HUB (Android App)

Scanned by CamScanner

One register is used to store the binary word we want to detect from the data stream. Input data stream enters one more register as serial data in and leaves as serial out. At every clocking instant, bitwise comparisions of these 2 register are done through Ex-NOR gate. When both the inputs are low & or high output of ExNOR will be high. i.e when both are equal. Final output is taken from 4 input AND gate which becomes 1 when all inputs are 1. i.e all bits are matched. If we want to change the binary word to be detected we can load that in bottom register. Hence it is called programmable sequence detector. For fixed sequence detector, we can reduce h/w cost removing bottom register and directly connecting to +Vcc & GND depending 1 or 0 bit.

Serial Adder:-

Available At: VTU HUB (Android App)
Scanned by CamScanner

Note:

Half Adder:

When we add binary number, we start with LSB column. Fig shows how to build a half adder.

OUTPUT of XOR gate is SUM and AND gate is CARRY.

Full Adder:

Can add 3 bits at a time

Fig shows the circuit of Full adder.

CARRY = AB + BC + AC , SUM = $A \oplus B \oplus C$

General representation:

$C_i = A_i B_i + (A_i + B_i) C_{i-1}$, $S_i = A_i \oplus B_i \oplus C_{i-1}$

Two 8-bit numbers, to be added ($A_7 A_6 A_5 \cdots A_0$ and $B_7 B_6 B_5 \cdots B_0$) are loaded in 2 8bit shift register A and B. LSB of each number appears in the rightmost position in 2 registers.

Serial data out of A and B are fed to data inputs of full adder. The carry in is fed from its own carry output delayed by one clock period by a D flip-flop, which is initially cleared. Both registers and D flip-flop are triggered by same clock. The sum (S) output of FA is fed to serial data in of Shift-Register A.

Serial addition process:

At 1st clock cycle, the LSBs of 2 numbers (A_0 & B_0) appear at serial of respective register and added by FA and generate sum S_0 and C_0. S_0 is available at serial input of register A. and C_0 at input of D flip-flop

At NT of clock register shift its content to right by one unit. S_0 becomes MSB of A and C_0 appears at output of D flip-flop.

Available At: VTU HUB (Android App)

Scanned by CamScanner

In Second clock cycle FA is fed by 2nd bits (A_1 & B_1) of 2 numbers and previous carry C_0, S_1 and C_1 are generated and made available at serial data in of A register & input of D flip flop respectively. At NT of clock S_1 becomes MSB of A and S_0 occupies next position. At 3rd clock cycle A_2 & B_2 appear at data input of FA & C_1 at carry input and S_2 & C_2 are generated. This process goes on and is stopped by inhibiting the clock after 8 clock cycles. Now shift register A store sum S_7 is MSB and leftmost position and S_0 in rightmost position and Final carry at D flip-flop output.

Limitation: Final addition result is delayed by 8 clock cycles.

Register Implementation in HDL:

```
module parallelinout (D, Clock, Clear, Q);
input Clock, Clear;
input [5:0] D;
output [5:0] Q;
reg [5:0] Q;
always @ (negedge Clock or negedge Clear)
  if (~Clear)
      Q = 6'b0;
else
      Q = D;
endmodule

module shiftrightreg (D, Clock, T);
input Clock D;
output T;
reg T;
reg Q, R, S; // since are used inside always block
always @ (negedge Clock)
```

Available At: VTU HUB (Android App)

Scanned by CamScanner

```
begin
    Q <= D;
    R <= Q;
    S <= R;
    T <= S;
end
endmodule

module serialinparalleloutl (D, Clock, Q);
input Clock, D;
output [3:0]Q;
reg [3:0]Q;
always @ (negedge clock)
begin
    Q[0] <= D;
    Q[1] <= Q[0];
    Q[2] <= Q[1];
    Q[3] <= Q[2];
end
endmodule

module switchtail count (Clock, reset, cnt);
input reset, Clock;
output [3:0] cnt;
reg [3:0] cnt = 4'b0;
always @ (posedge clock)
begin
    if(reset) cnt <= 4'b0;
    else
        begin cnt[3] <= ~cnt[0];
              cnt[2] <= cnt[3];
              cnt[1] <= cnt[2];
              cnt[0] <= cnt[1];
        end
end
endmodule
```

Available At: VTU HUB (Android App)

Scanned by CamScanner

Design an 8 bit sequence generator that generates the sequence 11000100 repetitively using shift register. We load 8 bit ringcounter as shown below fig with the given sequence and at the output the sequence will be generated repetitively.

Available At: VTU HUB (Android App)

Scanned by CamScanner

MODULE – 5

REGISTERS AND COUNTERS

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

Counters are another simple type of sequential circuits. A counter is usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received.

REGISTERS AND REGISTER TRANSFERS:

Several D flip-flops may be grouped together with a common clock to form a register (SEE THE FOLLOWING Figure). Since 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.

(a) Using gated clock

When Load = 0, the register is not clocked, and it holds its present value. Load = 1, the clock signal (Clk) is transmitted to the flip-flop clock inputs and the data applied to the D inputs will be loaded into the flip-flops on the falling edge of the clock.

For example, if the Q outputs are 0000 (Q3 = Q2 = Q1 = Q0 = 0) and the data inputs are 1101 (D3 = 1, D2 = 1, D1 = 0 and D0 = 1), after the falling edge of clock, Q will change from 0000 to 1101 as indicated in the above Figure (The notation 0 → 1 at the flip-flop outputs indicates a change from 0 to 1).

The flip-flops in the register have asynchronous clear inputs that are connected to a common clear signal, *ClrN*. The bubble at the clear inputs indicates that a logic 0 is required to clear the flip-flops. *ClrN* is normally 1, and if it is changed momentarily to 0, the Q outputs of all four flip-flops will become 0.

Gating the clock with another signal can cause timing problems. If flip-flops with clock enable are available, the register can be designed as shown in the following Figure (b).

(b) With clock enable

(c) Symbol

The load signal is connected to all four CE inputs. When Load = 0, the clock is disabled and the register holds its data. When Load = 1, the clock is enabled, and the data applied to the D inputs will be loaded into the flip-flops, following 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. A group of wires that perform a common function is often referred to as a *bus*. A heavy line is used to represent a bus, and a slash with a number beside it indicates the number of bits in the bus.

Transferring data between registers is a common operation in digital systems. The following Figure 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.

The following Figure (a) shows an integrated circuit register that contains eight D flip-flops with tri-state buffers at the flip-flop outputs. These buffers are enabled when En = 0. A symbol for this 8-bit register is shown in Figure (b).

The following Figure (c) shows how data can be transferred from one of four 8-bit registers into one of two other registers. Registers A, B, C, and D are of the type shown in above Figure.

The outputs from these registers are all connected in parallel to a common tri-state bus. Registers G and H are 8-bit D type registers (PIPO). The flip-flop inputs of registers G and H are also connected to the bus. When *EnA = 0*, the tri-state outputs of register A are enabled onto the bus. If *LdG = 1*, these signals on the bus are loaded into register G after the rising clock edge (or into register H if *LdH = 1*). Similarly, the data in register B, C, or D is transferred to G (or H) when *EnB*, *EnC*, or *EnD* is 0, respectively and *LdG = 1* (or *LdH = 1*). If *LdG = LdH = 1*, both G and H will be loaded from the bus. The four enable signals may be generated by a decoder. The operation can be summarized as follows:

If *EF = 00*, A is stored in G (or H) — If *EF = 01*, B is stored in G (or H).

If *EF = 10*, C is stored in G (or H) — If *EF = 11*, D is stored in G (or H).

Note that 8 bits of data are transferred in parallel from register A, B, C, or D to register G or H. As an alternative to using a bus with tri-state logic, eight 4-to-1 multiplexers could be used, but this would lead to a more complex circuit.

Parallel Adder with Accumulator:

In computer circuits, it is frequently desirable to store one number in a register of flip-flops (called an accumulator) and add a second number to it, leaving the result stored in the accumulator.

One way to build a parallel adder with an accumulator is to add a register to the adder as shown in the following Figure.

Suppose that the number $X = x_n \ldots x_2x_1$ is stored in the accumulator. Then, the number $Y = y_n \ldots y_2y_1$ is applied to the full adder inputs, and after the carry has propagated through the adders, the sum of X and Y appears at the adder outputs. An add signal (Ad) is used to load the adder outputs into the accumulator flip-flops on the rising clock edge. If $s_i = 1$, the next state of flip-flop x_i will be 1. If $s_i = 0$, the next state of flip-flop x_i will be 0.Thus, $x_i^+ = s_i$, and if $Ad = 1$, the number X in the accumulator is replaced with the sum of X and Y, following the rising edge of the clock.

Observe that the adder with accumulator is an iterative structure that consists of a number of identical cells. Each cell contains a full adder and an associated accumulator flip-flop. Cell i, which has inputs c_i and y_i and outputs $c_i = 1$ and x_i, is referred to as a typical cell.

Before addition can take place, the accumulator must be loaded with X. This can be accomplished in several ways. The easiest way is to first clear the accumulator using the asynchronous clear inputs on the flip-flops, and then put the X data on the Y inputs to the adder and add to the accumulator in the normal way. Alternatively, we could add multiplexers at the accumulator inputs so that we could select either the Y input data or the adder output to load into the accumulator. This would eliminate the extra step of clearing the accumulator but would add to the hardware complexity.

The following Figure shows a typical cell of the adder where the accumulator flip-flop can either be loaded directly from y_i or from the sum output (s_i).When $Ld = 1$ the multiplexer selects y_i, and y_i is loaded into the accumulator flip-flop (x_i) on the rising clock edge. When $Ad = 1$ and $Ld = 0$, the adder output (s_i) is loaded into x_i. The Ad and Ld signals are ORed together to enable the clock when either addition or loading occurs. When Ad = Ld = 0, the clock is disabled and the accumulator outputs do not change.

SHIFT REGISTERS:

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. Bits shifted out one end of the register may be lost, or if the shift register is of cyclic type, bits shifted out one end are shifted back in the other end.

There are two ways to shift data into a register – serial or parallel; and there are two ways to shift the data out of the register – serial or parallel. This leads to the construction of four basic types of registers, as shown in the following Figure. All of these configurations are commercially available as TTL MSI/LSI circuits. Examples: Serial in – serial out (SISO): 54/74LS91, 8-bits

Serial in – parallel out (SIPO): 54/74164, 8-bits

Parallel in – serial out (PISO): 54/74165, 8-bits

Parallel in – parallel out (PIPO): 54/74194, 4-bits & 54/74168, 8-bits.

Types of Shift Registers

The following 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 (Q3) 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. Because of the propagation delay of the flip-flops, the output value loaded into each flip-flop is the value before the rising clock edge.

(a) Flip-flop connections

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.

(b) Timing diagram

If we connect the serial output to the serial input, as shown by the dashed line, the resulting cyclic shift register performs an end-around shift. If the initial contents of the register is 0111, after one clock cycle the contents is 1011.After a second pulse, the state is 1101, then 1110, and the fourth pulse returns the register to the initial 0111 state.

Shift registers with 4, 8, or more flip-flops are available in integrated circuit form. The following Figure illustrates an 8-bit serial-in, serial-out shift register. Serial in means that data is shifted into the first flip-flop one bit at a time, and the flip-flops cannot be loaded in parallel. Serial out means that data can only be read out of the last flip-flop and the outputs from the other flip-flops are not connected to terminals of the integrated circuit.

8-Bit Serial-in, Serial-out Shift Register

(a) Block diagram

(b) Logic diagram

The inputs to the first flip-flop are $S = SI$ and $R = SI'$. Thus, if $SI = 1$, a 1 is shifted into the register when it is clocked, and if $SI = 0$, a 0 is shifted in. The following Figure shows a typical timing diagram.

Typical Timing Diagram for Shift Register

The following Figure (a) 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 Q3,Q2, and Q1 are shifted right. If $Sh = 0$ and $L = 1$, clocking the shift register will

cause the four data inputs (D3, D2, D1, D0) to be loaded in parallel into the flip-flops. If *Sh* = *L* = *0*, clocking the register causes no change of state.

(a) Block diagram

(b) Implementation using flip-flops and MUXes

The following Table summarizes the operation of this shift register. All state changes occur immediately following the falling edge of the clock.

Shift Register Operation

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 the above Figure (b). For the first flip-flop, when *Sh* = *L* = *0*, the flip-flop Q3 output is selected by the MUX, so $Q3^+$ = Q3 and no state change occurs. When *Sh* = *0* and *L* = *1*, the data input D3 is selected and loaded into the flip-flop. When *Sh* = *1* and *L* = *0* or *1*, *SI* is selected and loaded into the flip-flop. The second MUX selects Q2, D2, or Q3, etc. The next-state equations for the flip-flops are

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

A typical application of this register is the conversion of parallel data to serial data. The output from the last flip-flop (Q0) serves as a serial output as well as one of the parallel outputs. The following Figure shows a typical timing diagram.

The first clock pulse loads data into the shift register in parallel. During the next four clock pulses, this data is available at the serial output. Assuming that the register is initially clear (Q3Q2Q1Q0 = 0000), that the serial input is SI = 0 throughout, and that the data inputs D3D2D1D0 are 1011 during the load time (t_0), the resulting waveforms are as shown. Shifting occurs at the end of t_1, t_2, and t_3, and the serial output can be read during these clock times. During t_4, $Sh = L = 0$, so no state change occurs.

The following Figure (a) 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) Flip-flop connections

(b) State graph

If the initial state of the register is 000, the initial value of D3 is 1, so after the first clock pulse, the register state is 100. Successive states are shown on the state graph of Figure (b). When the register is in state 001, D3 is 0, and the next register state is 000. Then, successive clock pulses take the register around the loop again. Note that states 010 and 101 are not in the main loop. If the register is in state 010, then a shift pulse takes it to 101 and vice versa; therefore, we have a secondary loop on the state graph.

A circuit that goes through a fixed sequence of states is called a *counter*. A shift register with inverted feedback (*Q1'* connected to *D3*, as shown in above Figure) is often called a *Johnson counter* or *Twisted ring counter*. A shift register with non-inverted feedback (if *Q1* connected to *D3*, in above Figure) is often called a *Ring counter*.

DESIGN OF BINARY COUNTERS:

A *counter* is a sequential circuit that goes through a prescribed sequence of states up on the application of input pulse. Counters are in two categories –

- *Ripple (Asynchronous) Counter* – consists of a series connection of complementing flip-flops, with the output of each flip-flop connected to the clock-pulse input of the next higher-order flip-flop. The flip-flop holding the LSB receives the clock-pulse.
- *Synchronous Counter* – the input pulses/ clock-pulses are applied to all clock-pulse inputs of all the flip-flops simultaneously.

Asynchronous Counters:

Negative Clock	C	B	A	State or Count
-	0	0	0	0
a	0	0	1	1
b	0	1	0	2
c	0	1	1	3
d	1	0	0	4
e	1	0	1	5
f	1	1	0	6
g	1	1	1	7
h	0	0	0	0 repeats

3-Bit Binary Ripple Counter, Waveforms & Truth Table

The above Figure shows three negative-edge-triggered, JK flip-flops connected in cascade to form a 3-bit ripple counter. The system clock (a square wave), drives flip-flop A. The output of flip-flop A drives B,

and the output of B drives flip-flop C. All the J and K inputs are tied to $+V_{CC}$. Hence, flip-flops will toggle with a negative transition at its clock input.

Assume that, the flip-flops are all initially reset to 0 outputs. For every clock NT, flip-flop A will change state. Notice that, the waveform at the output of flip-flop A is one-half the clock frequency.

Since, A acts as clock for B, each time the waveform at A goes low, flip-flop B will toggle. Notice that, the waveform at the output of flip-flop B is one-half the frequency of A and one-fourth the clock frequency.

Since, B acts as clock for C, each time the waveform at B goes low, flip-flop C will toggle. The waveform at the output of flip-flop C is one-half the frequency of B and one-eighth the clock frequency.

Problem: *What is the clock frequency of a 3-bit ripple counter, if the period of the MSB waveform is 24 µs?*

Solution: Since there are eight clock cycles in one cycle of MSB, the period of the clock must be 24/8 = 3 µs. The clock frequency must be $1/(3*10^{-6})$ = 333 KHz.

NOTE:

1. A binary ripple counter in straight binary sequence will be as shown in the above table. A ripple counter having *n* flip-flops will have 2^n output conditions. For example, the three-flip-flop counter has 2^3 = 8 output conditions (000 to 111).

NOTE:

No. of Flip-Flops	*1*	*2*	*3*	*4*	*5*	*n*
No. of States	*2*	*4*	*8*	*16*	*32*	2^n

2. A three-flip-flop counter is often referred to as modulus-8 (or Mod-8) counter, since it has eight states. The *modulus* of a counter is the total number of states through which the counter can progress.

Problem: *How many flip-flops are required to construct a mod-128 counter? A mod-32 counter? What is the largest decimal number that can be stored in a mod-64 counter?*

Solution: A mod-128 counter must have seven flip-flops, since 2^7 = 128. Five flip-flops are needed to construct a mod-32 counter. The largest decimal number that can be stored in a mod-64 (six flip-flops) counter is 111111 = 63.

Synchronous Counters:

Synchronous means the operation of the flip-flops is synchronized by a common clock pulse so that when several flip-flops must change state, the state changes occur simultaneously.

Synchronous Binary Counters Using T Flip-Flops: Consider the following Figure, a binary counter using three T flip-flops to count clock pulses.

All the flip-flops change state a short time following the rising edge of the input pulse. The state of the counter is determined by the states of the individual flip-flops; for example, if flip-flop C is in state 0, B in state 1, and A in state 1, the state of the counter is 011.

Initially, assume that all flip-flops are set to the 0 state. When a clock pulse is received, the counter will change to state 001; when a second pulse is received, the state will change to 010, etc. The sequence of flip-flop states is CBA = 000, 001, 010, 011, 100, 101, 110, 111, 000, . . . Note that, when the counter reaches state 111, the next pulse resets it to the 000 state, and then the sequence repeats.

First, design the counter by inspection of the counting sequence; then, use a systematic procedure which can be generalized to other types of counters. The problem is to determine the flip-flop inputs—TC,TB, and TA. From the preceding counting sequence, observe that A changes state every time a clock pulse is received. Because A changes state on every rising clock edge, TA must equal 1. Next, observe that B changes state only if A = 1.Therefore, A is connected to TB as shown, so that if A = 1, B will change state when a rising clock edge occurs. Similarly, C changes state when a rising clock edge occurs only if B and A are both 1. Therefore, an AND gate is connected to TC so that C will change state if B = 1 and A = 1 when a rising clock edge occurs.

Now, verify that the circuit of above Figure counts properly by tracing signals through the circuit. Initially, CBA = 000, so only TA is 1 and the state will change to 001 when the first active clock edge arrives. Then, TB = TA = 1, and the state will change to 010 when the second active clock arrives. This process continues until finally when state 111 is reached, TC = TB = TA = 1, and all flip-flops return to the 0 state.

A state table (the following Table) shows the present state of flip-flops C, B, and A (before a clock pulse is received) and the corresponding next state (after the clock pulse is received). For example, if the flip-flops are in state CBA = 011 and a clock pulse is received, the next state will be $C^+ = B^+ = A^+ = 100$. Although the clock is not explicit in the table, it is understood to be the input that causes the counter to go

to the next state in sequence. A third column in the table is used to derive the inputs for TC,TB, and TA. Whenever the entries in the A and A^+ columns differ, flip-flop A must change state and TA must be 1. Similarly, if B and B^+ differ, B must change state so TB must be 1. For example, if CBA = 011, $C^+B^+A^+$ = 100, all three flip-flops must change state, so TCTBTA = 111.

State Table for Binary Counter

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	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table

State Diagram

$Q \rightarrow Q_{n+1}$		T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

TC, TB, and TA are now derived from the table as functions of C, B, and A. By inspection, TA = 1. The following Figure shows the Karnaugh maps for TC and TB, from which; TC = BA and TB = A. These equations yield the same circuit derived previously.

Karnaugh Maps for Binary Counter

Synchronous Binary Counters Using D Flip-Flops: Next, redesign the binary counter to use D flip-flops instead of T flip flops. The easiest way to do this is to convert each D flip-flop to a T flip-flop by adding an XOR (exclusive-OR) gate, as shown in the following Figure (The rightmost XOR gate can be replaced with an inverter because A ⊕ 1 = A).

We can also derive the D flip-flop inputs for the binary counter starting with its state table (given below).

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	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic Table

State Diagram

$Q \rightarrow Q_{n+1}$		D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation Table

For a D flip-flop, $Q^+ = D$. By inspection of the state table, the maps for DA, DB and DC are plotted as follows and D input equations are derived. This give the same logic circuit as was obtained by inspection.

Karnaugh Maps for D Flip-Flops

$$D_A = A^+ = A'$$
$$D_B = B^+ = BA' + B'A = B \oplus A$$
$$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$

Binary Up-Down Counter: The state graph and table for an up-down counter are shown in the following Figure.

State Graph and Table for Up-Down Counter

	$C^+B^+A^+$	
CBA	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

When U = 1, the counter counts up in the sequence 000, 001, 010, 011, 100, 101, 110, 111, 000 . . . When D = 1, the counter counts down in the sequence 000, 111, 110, 101, 100, 011, 010, 001, 000 . . .When U = D = 0, the counter state does not change, and U = D = 1 is not allowed.

By inspection of the table in above Figure, we can verify that these are the correct equations for a down counter. For every row of the table, $A^+ = A'$, so A changes state every clock cycle. For those rows where A = 0, $B^+ = B'$. For those rows where B = 0 and A = 0, $C^+ = C$.

The up-down counter can be implemented using D flip-flops and gates, as shown in the following Figure. The corresponding logic equations are also given in the following Figure.

$$D_A = A^+ = A \oplus (U + D)$$
$$D_B = B^+ = B \oplus (UA + DA')$$
$$D_C = C^+ = C \oplus (UBA + DB'A')$$

When U = 1 and D = 0, these equations reduce to equations for a binary up counter.

$$D_A = A^+ = A'$$
$$D_B = B^+ = BA' + B'A = B \oplus A$$
$$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$

When U = 0 and D = 1, these equations reduce to –

$$D_A = A^+ = A \oplus 1 = A' \qquad (A \text{ changes state every clock cycle})$$
$$D_B = B^+ = B \oplus A' \qquad (B \text{ changes state when } A = 0)$$
$$D_C = C^+ = C \oplus B'A' \qquad (C \text{ changes state when } B = A = 0)$$

Loadable Counter: The counter shown in the following Figure (a) has two control signals *Ld* (load) and *Ct* (count).

(a)

ClrN	*Ld*	*Ct*	C^+	B^+	A^+	
0	X	X	0	0	0	
1	1	X	D_C	D_B	D_A	(load)
1	0	0	C	B	A	(no change)
1	0	1	Present state + 1			

(b)

When *Ld = 1* binary data is loaded into the counter on the rising clock edge, and when *Ct = 1*, the counter is incremented on the rising clock edge. When *Ld = Ct = 0*, the counter holds its present state. When *Ld = Ct = 1*, load overrides count, and data is loaded into the counter. The counter also has an asynchronous

clear input that clears the counter when *ClrN* is 0. Figure (b) summarizes the counter operation. All state changes occur on the rising edge of the clock (except for the asynchronous clear).

The following Figure shows how the loadable counter can be implemented using flip-flops, MUXes, and gates.

When *Ld = 1*, each MUX selects a *Di* input, and because the output of each AND gate is 0, the output of each XOR gate is *Di*, which gets stored in a flip-flop. When *Ld = 0* and *Ct = 1*, each MUX selects one of the flip-flop outputs (C, B, or A). The circuit then becomes equivalent to a binary counter, and the counter is incremented on the rising clock edge. The next-state equations for the counter of above Figure are –

$$A^+ = D_A = (Ld' \cdot A + Ld \cdot D_{Ain}) \oplus Ld' \cdot Ct$$
$$B^+ = D_B = (Ld' \cdot B + Ld \cdot D_{Bin}) \oplus Ld' \cdot Ct \cdot A$$
$$C^+ = D_C = (Ld' \cdot C + Ld \cdot D_{Cin}) \oplus Ld' \cdot Ct \cdot B \cdot A$$

When *Ld = 0* and *Ct = 1*, these equations reduce to $A^+ = A'$, $B^+ = B \oplus A$, and $C^+ = C \oplus BA$, which are the equations previously derived for a 3-bit counter.

COUNTER FOR OTHER SEQUENCES:

In some applications, the sequence of states of a counter is not in straight binary order. The following Figure shows the state graph for such a counter. The arrows indicate the state sequence.

If this counter is started in state 000, the first clock pulse will take it to state 100, the next pulse to 111, etc. The clock pulse is implicitly understood to be the input to the circuit and not shown on the graph. The corresponding state table for the counter is given the following Table. Note that the next state is unspecified for the present states 001, 101, and 110.

State Graph for Counter

State Table

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

We will design the counter specified by the above Table using T flip-flops. We could derive TC, TB, and TA directly from this table, as in the preceding example. However, it is often more convenient to plot next-state maps showing C^+, B^+, and A^+ as functions of C, B, and A, and then derive TC, TB, and TA from these maps.

Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	T_C	T_B	T_A
0	0	0	1	0	0			
0	0	1	-	-	-			
0	1	0	0	1	1			
0	1	1	0	0	0			
1	0	0	1	1	1			
1	0	1	-	-	-			
1	1	0	-	-	-			
1	1	1	0	1	0			

From the first row of the table, the CBA = 000; and hence, the C^+, B^+, and A^+ columns are filled in with 1, 0, and 0, respectively. From the second row, the CBA = 001; all three columns are filled in with don't-cares. From the third row, the CBA = 010; and the C^+, B^+, and A^+ columns are filled with 0, 1, and 1, respectively. The next-state columns can be quickly completed by continuing in this manner.

Next, we will derive the maps for the T inputs from the next-state maps.

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic Table

State Diagram

$Q \rightarrow Q_{n+1}$		T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table

Present State			Next State			Flip-Flop Inputs		
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
0	1	0	0	1	1	0	0	1
0	1	1	0	0	0	0	1	1
1	0	0	1	1	1	0	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	1	0	1

Now, draw the K-maps for TC, TB, and TA separately, and derive the expressions.

$T_C = C'B' + CB$ $\quad$ $T_B = C'A + CB'$ $\quad$ $T_A = C + B$

Derivation of T inputs

Finally, draw the counter circuit based on the expressions.

NOTE: In the above problem, for the design, initially, when the power is switched on, if circuit enters to an invalid state (state 1 or state 5 or state 6), then *lock-in condition* results. The designed counter will not give proper result. For this reason, all of the don't-care states in a counter should be checked to make sure that they eventually lead into the main counting sequence unless a power-up reset is provided.

The solution is *self correcting counter*.

Counter Design Using D Flip-Flops: For a D flip-flop, $Q^+ = D$, so the D input map is identical with the next-state map.

Present State			Next State			Flip-Flop Inputs		
C	B	A	C^+	B^+	A^+	D_C	D_B	D_A
0	0	0	1	0	0	1	0	0
0	0	1	-	-	-	x	x	x
0	1	0	0	1	1	0	1	1
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1
1	0	1	-	-	-	x	x	x
1	1	0	-	-	-	x	x	x
1	1	1	0	1	0	0	1	0

Draw the K-map; and from the map, we get the following expressions:

$$D_C = C^+ = B' \qquad D_B = B^+ = C + BA'$$
$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Corresponding counter circuit is –

COUNTER DESIGN USING S-R AND J-K FLIP-FLOPS:

The procedures used to design a counter with S-R OR flip-flops are similar to the procedures discussed. However, instead of deriving an input equation for each D or T flip-flop, the S and R input equations must be derived.

The following Table describes the behavior of S-R flip-flops:

S	R	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	– } inputs not allowed
1	1	1	– } inputs not allowed

Characteristic Table

State Diagram

Q	Q^+	S	R
0	0	0	0
		0	1
0	1	1	0
1	0	0	1
1	1	0	0
		1	0

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation Table

Next, we will redesign the counter for following Figure using S-R flip-flops.

State Graph for Counter

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
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
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	0	1	X	0	0	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs.

$R_C = A$ $\quad S_C = B'$ $\quad R_B = C'A$ $\quad S_B = C$ $\quad R_A = A$ $\quad S_A = CA' + BA' = A'(C + B)$

S-R flip-flop equations

Logic circuit

The procedure used to design a counter with J-K flip-flops is very similar to that used for S-R flip-flops. The J-K flip-flop is similar to the S-R flip-flop except that J and K can be 1 simultaneously, in which case the flip-flop changes state.

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Characteristic Table

State Diagram

Q	Q^+	J	K
0	0	0	0
		0	1
0	1	1	0
		1	1
1	0	0	1
		1	1
1	1	0	0
		1	0

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation Table

Now, we will redesign the counter for following Figure using JK flip-flops.

State Graph for Counter

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
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
1	1	0	–	–	–	X	X	X	X	X	X
1	1	1	0	1	0	X	1	X	0	X	1

Draw the K-map; and from the map, we get the expressions for flip-flop inputs; and finally draw the counter circuit using J-K flip-flops.

J-K flip-flop input equations

Logic circuit (omitting the feedback lines)

Homework:

1] Construct a 4-bit Johnson counter using

(a) D flip-flops *(b) J-K flip-flops.*

Clock	Serial in = $\bar{T}$	Q	R	S	T	$Y = \bar{Q}\bar{T}$
0	1	0	0	0	0	1
1	1	1	0	0	0	0
2	1	1	1	0	0	0
3	1	1	1	1	0	0
4	0	1	1	1	1	0
5	0	0	1	1	1	0
6	0	0	0	1	1	0
7	0	0	0	0	1	0
8	1	0	0	0	0	1
9	1	1	0	0	0	0 repeats

2] Design a 3-bit counter which counts in the sequence:

001, 011, 010, 110, 111, 101, 100, (repeat) 001, . . .

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

(c) Use S-R flip-flops *(d) Use J-K flip-flops.*

3] Design a 3-bit counter which counts in the sequence:

001, 100, 101, 111, 110, 010, 011, 001, . . .

(a) Use D flip-flops *(b) Use J-K flip-flops*

(c) Use T flip-flops *(d) Use S-R flip-flops*

(e) What will happen if the counter of (a) is started in state 000?

A SEQUENTIAL PARITY CHECKER:

When binary data is transmitted or stored, an extra bit (called a *parity bit*) is frequently added for purposes of error detection. For example, if data is being transmitted in groups of 7 bits, an eighth bit can be added to each group of 7 bits to make the total number of 1's in each block of 8 bits an odd number. When the total number of 1 bits in the block (including the parity bit) is odd, we say that the *parity is odd*. Alternately, the parity bit could be chosen such that the total number of 1's in the block is even, in which case we would have *even parity*. Some examples of 8-bit words with odd parity are –

If any single bit in the 8-bit word is changed from 0 to 1 or from 1 to 0, the parity is no longer odd. Thus, if any single bit error occurs in transmission of a word with odd parity, the presence of this error can be detected because the number of 1 bits in the word has been changed from odd to even.

As a simple example of a sequential circuit which has one input in addition to the clock, we will design a parity checker for serial data. (Serial implies that the data enters the circuit sequentially, one bit at a time.) This circuit has the form shown in the following Figure.

When a sequence of 0's and 1's is applied to the X input, the output of the circuit should be Z = 1, if the total number of 1 inputs received is odd; that is, the output should be 1 if the input parity is odd. Thus, if data which originally had odd parity is transmitted to the circuit, a final output of Z = 0 indicates that an error in transmission has occurred.

The value of X is read at the time of the active clock edge. The X input must be synchronized with the clock so that it assumes its next value before the next active clock edge. The clock input is necessary in order to distinguish consecutive 0's or consecutive 1's on the X input.

Block Diagram for Parity Checker

State Graph for Parity Checker

The sequential circuit must "*remember*" whether the total number of 1 inputs received is even or odd; therefore, only two states are required. We will designate these states as S0 and S1, corresponding respectively to an even number of 1's received and an odd number of 1's received. We will start the circuit in state S0 because initially zero 1's have been received, and zero is an even number.

As indicated in state graph (above Figure), if the circuit is in state S0 (even number of 1's received) and X = 0 is received, the circuit must stay in S0 because the number of 1's received is still even. However, if X = 1 is received, the circuit goes to state S1 because the number of 1's received is then odd.

Similarly, if the circuit is in state S1 (odd number of 1's received) a 0 input causes no state change, but a 1 causes a change to S0 because the number of 1's received is then even.

The output Z should be 1 whenever the circuit is in state S1 (odd number of 1's received). The output is listed below the state on the state graph.

The following Table (a) gives the same information as the state graph in tabular form. The table shows that if the present state is S0, the output is Z = 0, and if the input is X = 1, the next state will be S1.

(a)

Present State	Next State $X = 0$	Next State $X = 1$	Present Output
S_0	S_0	S_1	0
S_1	S_1	S_0	1

(b)

Q	Q^+ $X = 0$	Q^+ $X = 1$	T $X = 0$	T $X = 1$	Z
0	0	1	0	1	0
1	1	0	0	1	1

State Table for Parity Checker

Since only two states are required, a single flip-flop (Q) will suffice. We will let Q = 0 correspond to state S0 and Q = 1 corresponds to S1. We can then set up a table which shows the next state of flip-flop Q as a function of the present state and X. If we use a T flip-flop, T must be 1 whenever Q and Q^+ differ. From the above Table (b), the T input must be 1 whenever X = 1. The following Figure shows the resulting circuit.

The following Figure also shows the output waveform for the circuit. When X = 1, the flip-flop changes state after the falling edge of the clock. Note that the final value of Z is 0 because an even number of 1's was received. If the number of 1's received had been odd, the final value of Z would be l. In this case, it would be necessary to reset the flip-flop to the proper initial state (Q = 0) before checking the parity of another input sequence.

Parity Checker

Waveforms for Parity Checker

STATE TABLES AND GRAPHS:

The *state tables* and *graphs* provides a more systematic approach which is useful for the analysis of larger circuits and which leads to a general synthesis procedure for sequential circuits. The *state table* specifies the next state and output of a sequential circuit in terms of its present state and input.

The following method can be used to construct the state table:

1. Determine the flip-flop input equations and the output equations from the circuit.
2. Derive the next-state equation for each flip-flop from its input equations, using one of the following relations:

D flip-flop	$Q^+ = D$
D-CE flip-flop	$Q^+ = D \cdot CE + Q \cdot CE'$
T flip-flop	$Q^+ = T \oplus Q$
S-R flip-flop	$Q^+ = S + R'Q$
J-K flip-flop	$Q^+ = JQ' + K'Q$

3. Plot a next-state map for each flip-flop.
4. Combine these maps to form the state table. Such a state table, which gives the next state of the flip-flops as a function of their present state and the circuit inputs, is frequently referred to as a *transition table*.

Example: Derive the state table for the circuit of the following Figure (Moore Model):

1. The flip-flop input equations and output equation are

$$D_A = X \oplus B' \qquad D_B = X + A \qquad Z = A \oplus B$$

2. The next-state equations for the flip-flops are

$$A^+ = X \oplus B' \qquad B^+ = X + A$$

3. The corresponding maps are

(a)

AB	A^+B^+ X = 0	X = 1	Z
00	10	01	0
01	00	11	1
11	01	11	0
10	11	01	1

(b)

Present State	Next State X = 0	X = 1	Present Output (Z)
S_0	S_3	S_1	0
S_1	S_0	S_2	1
S_2	S_1	S_2	0
S_3	S_2	S_1	1

State Tables

4. Combining these maps yields the transition table (Table (a)), which gives the next state of both flip-flops (A^+B^+) as a function of the present state and input. The output function Z is then added to the table. In this example, the output depends only on the present state of the flip-flops and not on the input, so only a single output column is required.

Using the above Table (a), we can construct the timing chart for some given input sequence and specified initial state.

Timing Chart

Initially AB = 00 and X = 0, so Z = 0 and A^+B^+ = 10. This means that after the rising clock edge, the flip-flop state will be AB = 10. Then, with AB = 10, the output is Z = 1. The next input is X = 1, so A^+B^+ = 01 and the state will change after the next rising clock edge. Continuing in this manner, we can complete the timing chart.

If we are not interested in the individual flip-flop states, we can replace each combination of flip-flop states with a single symbol which represents the state of the circuit. Replacing 00 with S0, 01 with S1, 11 with S2, and 10 with S3 in Table (a) yields Table (b). The Z column is labeled Present Output because it is the output associated with the Present State.

(a)

AB	A^+B^+ X = 0	A^+B^+ X = 1	Z
00	10	01	0
01	00	11	1
11	01	11	0
10	11	01	1

(b)

Present State	Next State X = 0	Next State X = 1	Present Output (Z)
S_0	S_3	S_1	0
S_1	S_0	S_2	1
S_2	S_1	S_2	0
S_3	S_2	S_1	1

Moore State Graph

The state graph of (given in above Figure) represents Table (b). Each node of the graph represents a state of the circuit, and the corresponding output is placed in the circle below the state symbol. The arc joining two nodes is labeled with the value of X which will cause a state change between these nodes. Thus, if the circuit is in state S0 and X = 1, a clock edge will cause a transition to state S1.

Example: Derive the state table for the circuit of the following Figure (Mealy Model):

The next-state and output equations are –

$$A^+ = J_A A' + K'_A A = XBA' + X'A$$
$$B^+ = J_B B' + K'_B B = XB' + (AX)'B = XB' + X'B + A'B$$
$$Z = X'A'B + XB' + XA$$

The next-state and output maps (following Figure) combine to form the transition table (shown in Table (a)). Given values for A, B, and X, the current value of the output is determined from the Z column of this table, and the states of the flip-flops after the active clock edge are determined from the A^+B^+ columns.

We can construct the timing chart of (given in the following Figure) using Table (a) (given below).

(a)

AB	A^+B^+ X = 0	1	Z X = 0	1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)

Present State	Next State X = 0	1	Present Output X = 0	1
S_0	S_0	S_1	0	1
S_1	S_1	S_2	1	0
S_2	S_2	S_0	0	1
S_3	S_3	S_1	0	1

Initially with A = B - 0 and X = 1, the table shows that Z = 1 and A^+B^+ = 01. Therefore, after the falling clock edge, the state of flip-flop B will change to 1, as indicated in above Figure. Now, from the 01 row of the table, if X is still 1, the output will be 0 until the input is changed to X = 0. Then, the output is Z = 1, and the next falling clock edge produces no state change. Finish stepping through the state table in this manner and verify that A, B, and Z are as given in the above Figure.

If we let AB = 00 correspond to circuit state S0, 01 to S1, 11 to S2, and 10 to S3, we can construct the state table as given in Table (b) and the state graph of the following Figure.

Mealy State Tables

(a)

AB	A^+B^+ X = 0	A^+B^+ 1	Z X = 0	Z 1
00	00	01	0	1
01	01	11	1	0
11	11	00	0	1
10	10	01	0	1

(b)

Present State	Next State X = 0	Next State 1	Present Output X = 0	Present Output 1
S_0	S_0	S_1	0	1
S_1	S_1	S_2	1	0
S_2	S_2	S_0	0	1
S_3	S_3	S_1	0	1

Mealy State Graph

In Table (b), the Present Output column gives the output associated with the present state and present input. Thus, if the present state is S0 and the input changes from 0 to 1, the output will immediately change from 0 to 1. However, the state will not change to the next state (S1) until after the clock pulse. For the above Figure (Mealy State Graph), the labels on the arrows between states are of the form X/Z, where the symbol before the slash is the input and the symbol after the slash is the corresponding output.

Thus, in state S0 an input of 0 gives an output of 0, and an input of 1 gives an output of 1. For any given input sequence, we can easily trace out the state and output sequences on the state graph. For the input sequence X = 10101, verify that the corresponding output sequence is 11001. This agrees with timing chart; if the false outputs are ignored. Note that the false outputs do not show on the state graph because the inputs are read at the active clock edge, and no provision is made for extra input changes between active edges.

Operation of Serial Adder:

Consider a serial adder [following Figure (a)] that adds two n-bit binary numbers X = $x_{n-1} \ldots x_1x_0$ and Y = $y_{n-1} \ldots y_1y_0$.

(a) With *D* flip-flop

x_i	y_i	c_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

(b) Truth table

Serial Adder

Timing Diagram for Serial Adder

The binary numbers are fed in serially, one pair of bits at a time, and the sum is read out serially, one bit at a time. First, x_0 and y_0 are fed in; a sum digit s_0 is generated, and the carry c_1 is stored. At the next clock time, x_1 and y_1 are fed in and added to c_1 to give the next sum digit s_1 and the new carry c^2, which is

stored. This process continues until all bits have been added. A full adder is used to add the x_i, y_i, and c_i bits to form c_{i+1} and s_i. A D flip-flop is used to store the carry (c_{i+1}) on the rising edge of the clock. The x_i and y_i inputs must be synchronized with the clock.

The above Figure also shows a timing diagram for the serial adder. In this example, we add 10011 + 00110 to give a sum of 11001 and a final carry of 0. Initially the carry flip-flop must be cleared so that c_0 = 0.We start by adding the least-significant (rightmost) bits in each word. Adding 1 + 0 + 0 gives s_0 = 1 and c_1 = 0, which is stored in the flip-flop at the rising clock edge. Since y_1 is 1, adding 1 + 1 + 0 gives s_1 = 0 and c_2 = 1, which is stored in the flip-flop on the rising clock edge. This process continues until the addition is completed. Reading the sum output just before the rising edge of the clock gives the correct result.

Using the truth table for the full adder (Table given with above Figure), we can construct a state graph (following Figure) for the serial adder.

State Graph for Serial Adder

Present State	Next State X_1X_2 = 00	01	10	11	Present Output (Z_1Z_2) X_1X_2 = 00	01	10	11
S_0	S_3	S_2	S_1	S_0	00	10	11	01
S_1	S_0	S_1	S_2	S_3	10	10	11	11
S_2	S_3	S_0	S_1	S_1	00	10	11	01
S_3	S_2	S_2	S_1	S_0	00	00	01	01

A State Table with Multiple Inputs and Outputs

State Graph

The serial adder is a Mealy machine with inputs x_i and y_i and output s_i. The two states represent a carry (c_i) of 0 and 1, respectively. From the table, c_i is the present state of the sequential circuit, and c_{i+1} is the

next state. If we start in S_0 (no carry), and x_i y_i = 11, the output is s_i = 0 and the next state is S_1.This is indicated by the arrow going from state S_0 to S_1.

The above Table is the state table for a Mealy sequential circuit with two inputs and two outputs. The above Figure also shows the corresponding state graph. The notation 00, 01/00 on the arc from S_3 to S_2 means if $X_1 = X_2 = 0$ or $X_1 = 0$ and $X_2 = 1$, then $Z_1 = 0$ and $Z_2 = 0$.

Construction and Interpretation of Timing Charts:

Several important points concerning the construction and interpretation of timing charts are summarized as follows:

1. When constructing timing charts, note that a state change can only occur after the rising (or falling) edge of the clock, depending on the type of flip-flop used.
2. The input will normally be stable immediately before and after the active clock edge.
3. For a Moore circuit, the output can change only when the state changes, but for a Mealy circuit, the output can change when the input changes as well as when the state changes. A false output may occur between the time the state changes and the time the input is changed to its new value. (In other words, if the state has changed to its next value, but the old input is still present, the output may be temporarily incorrect).
4. False outputs are difficult to determine from the state graph, so use either signal tracing through the circuit or use the state table when constructing timing charts for Mealy circuits.
5. When using a Mealy state table for constructing timing charts, the procedure is as follows:
    a) For the first input, read the present output and plot it.
    b) Read the next state and plot it (following the active edge of the clock pulse).
    c) Go to the row in the table which corresponds to the next state and read the output under the old input column and plot it. (This may be a false output).
    d) Change to the next input and repeat steps (a), (b), and (c). (Note: If you are just trying to read the correct output sequence from the table, step (c) is naturally omitted).
6. For Mealy circuits, the best time to read the output is just before the active edge of the clock, because the output should always be correct at that time.

The example in the following Figure shows a state graph, a state table, a circuit that implements the table, and a timing chart.

When the state is S_0 and the input is X = 0, the output from the state graph, state table, circuit, and timing chart is Z = 1 (labeled A on the figure). Note that this output occurs before the rising edge of the clock. In a Mealy circuit, the output is a function of the present state and input; therefore, the output should be read just before the clock edge that causes the state to change.

Q	X = 0	1	0	1
0	S_0	S_1	S_1	1	0
1	S_1	S_0	S_1	0	1

As you continue to study this example, each time the input X changes, trace the changes on the state graph, the state table, the circuit, and the timing chart. Because the input X was 0 before the first rising edge of the clock, the state changes to S_1 after the first rising edge of the clock. Because of the state change, the output also changes (B on the timing chart), but because the input has not yet changed to its new value, the output value may not be correct. We refer to this as a *false output* or *glitch*. If the input changes several times before it assumes its correct value, the output may also change several times (C). The input must assume its correct value before the rising edge of the clock, and the output should be read at this time (D). After the rising clock edge, the state stays the same and the output stays the same for this particular example. In general, the state may change after a rising edge of the clock, and the state change may result in an output change. Again, the output value may be wrong because the input still has the old value (E).When the input is changed to its new value, the output changes to its new value (F), and this value should be read before the next rising clock edge. If we look at the input and output just before each rising edge of the clock, we find the following sequences: X = 0 1 0

Z =1 1 0

You should be able to verify the sequence for Z using the state graph, using the state table, and using the circuit diagram.

Programmed Exercise

1(a) In this exercise you will analyze the following sequential circuit using a state table and a timing chart.
Derive the next-state and output equations.

$A^+ =$ _____

$B^+ =$ _____

$Z =$ _____

Answer $Z = XA + X'B,\ B^+ = (A' \oplus X)B' + XB = A'B'X' + AB'X + XB$
$A^+ = B(A + X)$

1(b) Plot these equations on maps and complete the transition table.

AB	A^+B^+ X = 0	A^+B^+ X = 1	Z 0	Z 1
00				
01				
11				
10				

Answer to 1(b)

	AB	A^+B^+ $X=0$	A^+B^+ 1	Z 0	Z 1
S_0	00	01	00	0	0
S_1	01	00	11	1	0
S_2	11	10	11	1	1
S_3	10	00	01	0	1

1(c) Convert your transition table to a state table using the given state numbering.

	Next State $X=0$	Next State 1	Output 0	Output 1
S_0				
S_1				
S_2				
S_3				

Answer to 1(c)

	$X=0$	1	0	1
S_0	S_1	S_0	0	0
S_1	S_0	S_2	1	0
S_2	S_3	S_2	1	1
S_3	S_0	S_1	0	1

1(d) Complete the corresponding state graph.

Answer to 1(d)

1(e) Using this graph, determine the state sequence and output sequence if the initial state is S_0 and the input sequence is $X = 0, 1, 0, 1$.

(1) The initial output with $X = 0$ in state S_0 is $Z =$ _____ and the next state is _____ .

(2) The output in this state when the next input ($X = 1$) is applied is $Z =$ _____ and the next state is_____ .

(3) When the third input ($X = 0$) is applied, the output is $Z =$ _____ and the next state is _____ .

(4) When the last input is applied, $Z =$ _____ and the final state is _____.

In summary, the state sequence is S_0,_____, _____, _____, _____ . The output sequence is $Z =$ _____ .

Answer to 1(e) S_0, S_1, S_2, S_3, S_1 $\quad Z = 0011$

1(f) This sequence for Z is the correct output sequence. Next, we will determine the timing chart including any false outputs for Z. Assuming that X changes midway between falling and rising clock edges, draw the waveform for X ($X = 0, 1, 0, 1$).

Answer

AB	A^+B^+ $X = 0$	$X = 1$
00	01	00
01	00	11
11	10	11
10	00	01

1(g) Referring to the transition table, sketch the waveforms for A and B assuming that initially $A = B = 0$. The state sequence is

$AB = 00$, _____, _____, _____, _____ .

Answer (Note that A and B change immediately after the falling clock edge.)

AB	Z $X = 0$	Z 1
00	0	0
01	1	0
11	1	1
10	0	1

Check your state sequence against the answer to 13.1(e), noting that $S_0 = 00$, $S_1 = 01$, $S_2 = 11$ and $S_3 = 10$.

1(h) Using the output table, sketch the waveform for Z. At time t_1, $X = A = B = 0$, so $Z =$ _____. At time t_2, $X =$ _____ and $AB =$ _____, so $Z =$ _____. At time t_3, $X =$ _____ and $AB =$ _____, so $Z =$ _____. Complete the waveform for Z, showing the output at t_4, t_5, etc.

Answer (Note that Z can change immediately following the change in X or immediately following the falling clock edge.)

1(i) (1) Because this is a Mealy circuit, the correct times to read the output are during intervals t_1, _____, _____, and _____ .

(2) The correct output sequence is therefore $Z =$ _____ .

(3) False outputs may occur during intervals _____, _____, and _____ .

(4) In two of these intervals, false outputs actually occur. These intervals are _____ and _____.

Answer

(1) t_1, t_3, t_5, and t_7

(2) Check your Z sequence against the answer to 13.1(e).

(3) t_2, t_4, and t_6.

(4) t_2 and t_6 (output during t_4 is not false because it is the same as t_5).

1(j) Finally, we will verify part of the timing chart by signal tracing on the original circuit (see 1(a)).

(1) Initially, $A = B = 0$ and $X = 0$, so $D_A =$ _____, $J_B =$ _____, $K_B =$ _____, and $Z =$ _____ .

(2) After the clock pulse $A =$ _____, $B =$ _____, and $Z =$ _____ .

(3) After X is changed to 1, $D_A =$ _____, $J_B =$ _____, $K_B =$ _____, and $Z =$ _____ .

(4) After the clock pulse, $A =$ _____, $B =$ _____, and $Z =$ _____ .

Check your answers against the timing chart. Answer to (1) corresponds to t_1, (2) to t_2, (3) to t_3, and (4) to t_4.

Try these –

1] Construct a state graph for the shift register shown. (X is the input, and Z is the output). Is this a Mealy or Moore machine?

2] Below is a state transition table with the outputs missing. The output should be Z = X'B' + XB.

ABC	$A^+B^+C^+$ X = 0	X = 1
000	011	010
001	000	100
010	100	100
011	010	000
100	100	001

(a) Is this a Mealy machine or Moore machine?

(b) Fill in the outputs on the state transition table.

(c) Give the state graph.

(d) For an input sequence of X = 10101, give a timing diagram for the clock, X, A, B, C, and Z. State changes occur on the rising clock edge. What is the correct output sequence for Z? Change X between rising and falling clock edges so that we can see false outputs, and indicate any false outputs on the diagram.

3] (a) Construct a state table and graph for the circuit shown.

(b) Construct a timing chart for the circuit for an input sequence X = 10111. (Assume that initially Q_1 = Q_2 = 0 and that X changes midway between the rising and falling clock edges.)

(c) List the output values produced by the input sequence.

4] Consider the circuit shown.

(a) Construct a state table and graph for the following circuit. Is the circuit a Mealy or Moore circuit? Does the circuit have any unused states? Assume 00 is the initial state.

(b) Draw a timing diagram for the input sequence X = 01100.

(c) What is the output sequence for the input sequence?

****