

Module 1

NETHRAVATHI H M

Assistant Professor

BGSIT,ACU

Optoelectronic Devices

- Study of devices that emit, detect and control light in the wavelength spectrum ranging from ultraviolet to far infrared
- Includes
 - Electrical-to-optical transducers
 - Optical-to-electrical transducers

Photodiode

- Light detector semiconductor device that converts light energy into electric current or voltage which depends upon the mode of operation
- Upper cut-off wavelength $\lambda = 1240/E_g$
- (E_g - bandgap energy)

p-n junction diode

- Under reverse bias, a small amount of electric current generated due to minority charge carriers
- Application of external reverse voltage to the p-n junction diode will not increase the population of minority charge carriers
- Reason: Minority carriers generated at n-side or p-side will recombine in the same material, before they cross the junction
- No electric current flows.

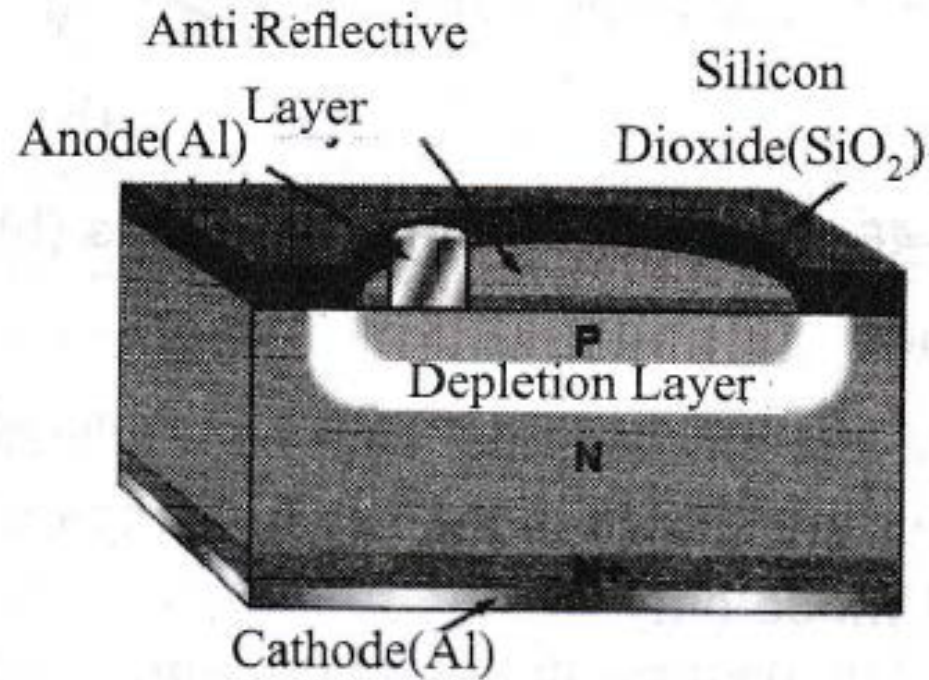
Photodiode

- Apply external energy directly to the depletion region to generate more charge carriers
- Photodiode is designed to generate more number of charge carriers in depletion region
- Light or photons as the external energy to generate charge carriers in depletion region

Photodiode : Construction

- *ion implantation* : surface of a layer of N-type is bombarded with P- type silicon ions to produce a P-type layer
- *Diffusion* : excess electrons move from N-type towards P-type and excess holes move from P-type towards N-type
- Results in the removal of free charge carriers close to the PN-junction, so creating a depletion layer

Photodiode : Construction



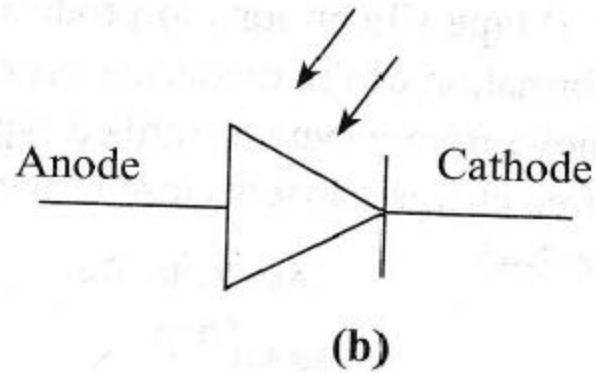
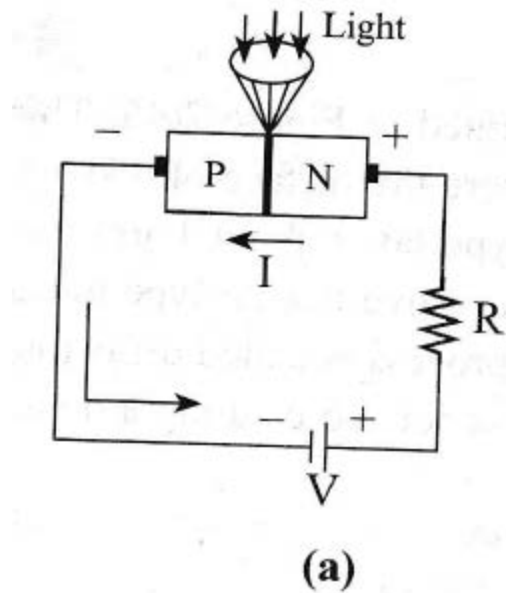
Photodiode : Construction

- Silicon Dioxide (SiO_2) in which there is a window for light to shine on the semiconductor.
- Silicon Nitride (SiN) to allow maximum absorption of light
- An anode connection of aluminium (Al) is provided to the P-type layer
- A more heavily doped N^+ layer to provide a low resistance connection to the cathode

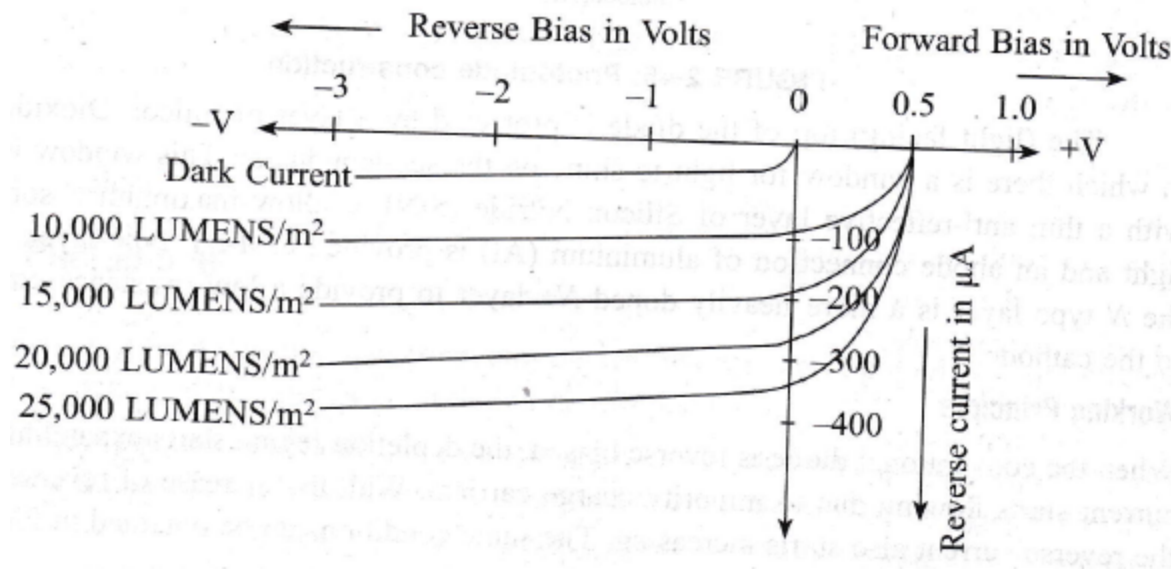
Photodiode : Working Principle

- 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.

Photodiode : Working Principle



Photodiode: V-I Characteristics



Reverse bias current is the summation of reverse saturation current and short circuit current.

Photodiode : Applications

- Consumer electronics devices like smoke detectors, compact disc players, and televisions and remote controls in VCRs.
- Photodiodes are frequently used for exact measurement of the intensity of light in science and industry.

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
- Electroluminescence is the property of the material to convert electrical energy into light energy

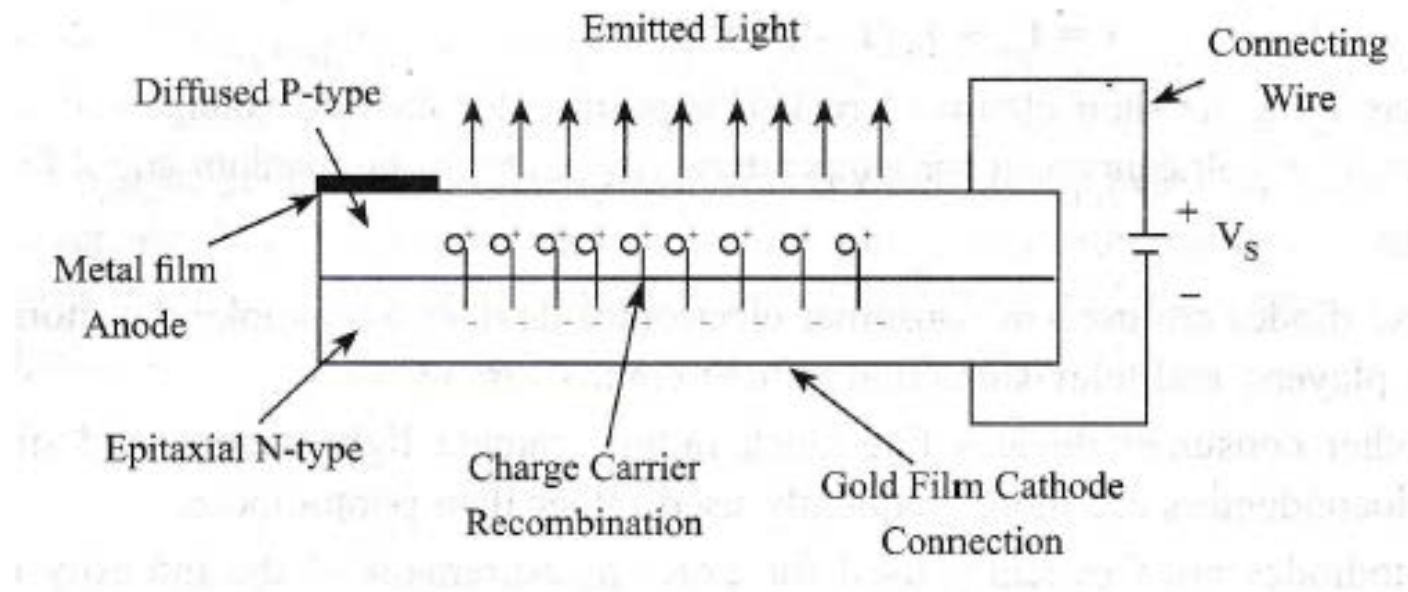
LED : Construction

- The semiconductor material used in LED is Gallium Arsenide (GaAs), Gallium phosphide (GaP) or Gallium Arsenide Phosphide (GaAsP).
- The semiconductor layer of P-type is placed above N-type because the charge carrier recombination occurs in P-type.
- If *P-type is placed below the N-type, the emitted light cannot be seen.*

LED : Construction

- The semiconductor material used in LED is Gallium Arsenide (GaAs), Gallium phosphide (GaP) or Gallium Arsenide Phosphide (GaAsP).
- The semiconductor layer of P-type is placed above N-type because the charge carrier recombination occurs in P-type.

LED : Construction



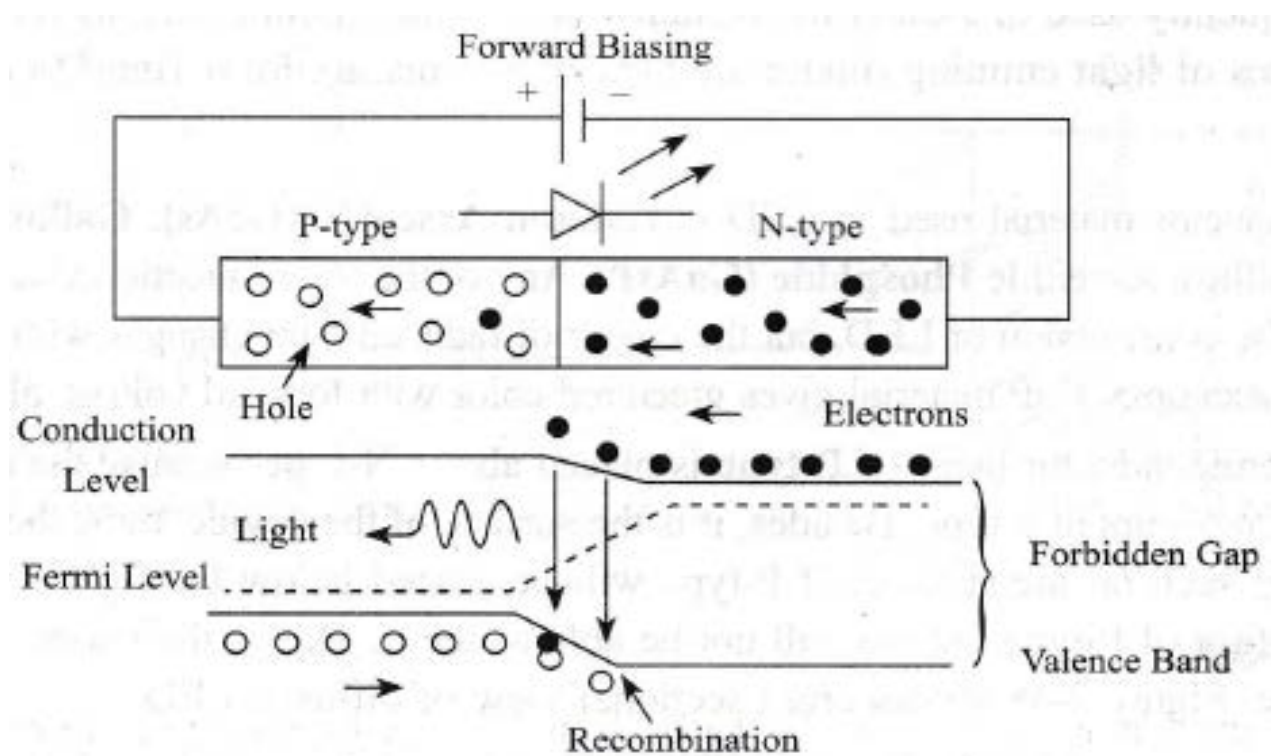
LED : Construction

- The P-type layer is formed from diffusion of semiconductor material.
- The metal film is used on the P-type layer to provide anode connection to the diode.
- 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.

LED : 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
- Holes are in the valence energy band.
- 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.

LED : Working Principle



LED : Working Principle

- Electron 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.

LED : Applications

- LEDs are used in remote control systems such TV or LCD remote.
- Used in traffic signals for controlling the traffic crowds in cities.
- 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.

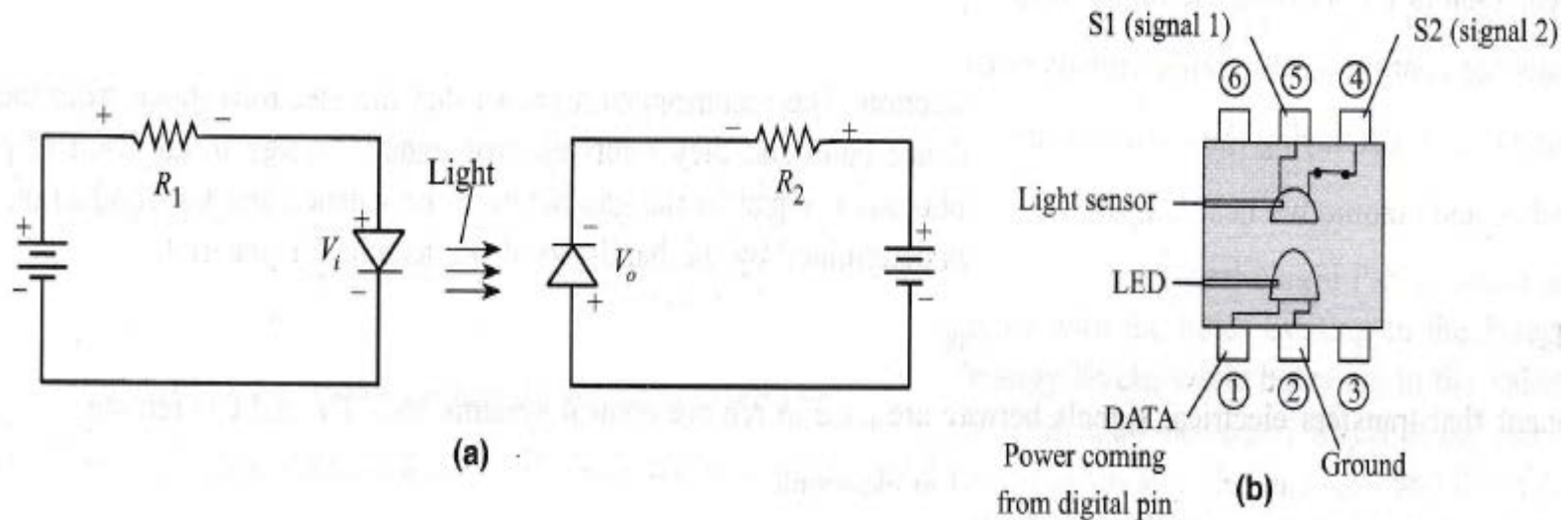
Photocoupler : Construction

- 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

Photocoupler : Working

- 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
- 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 R2.

Photocoupler : Working



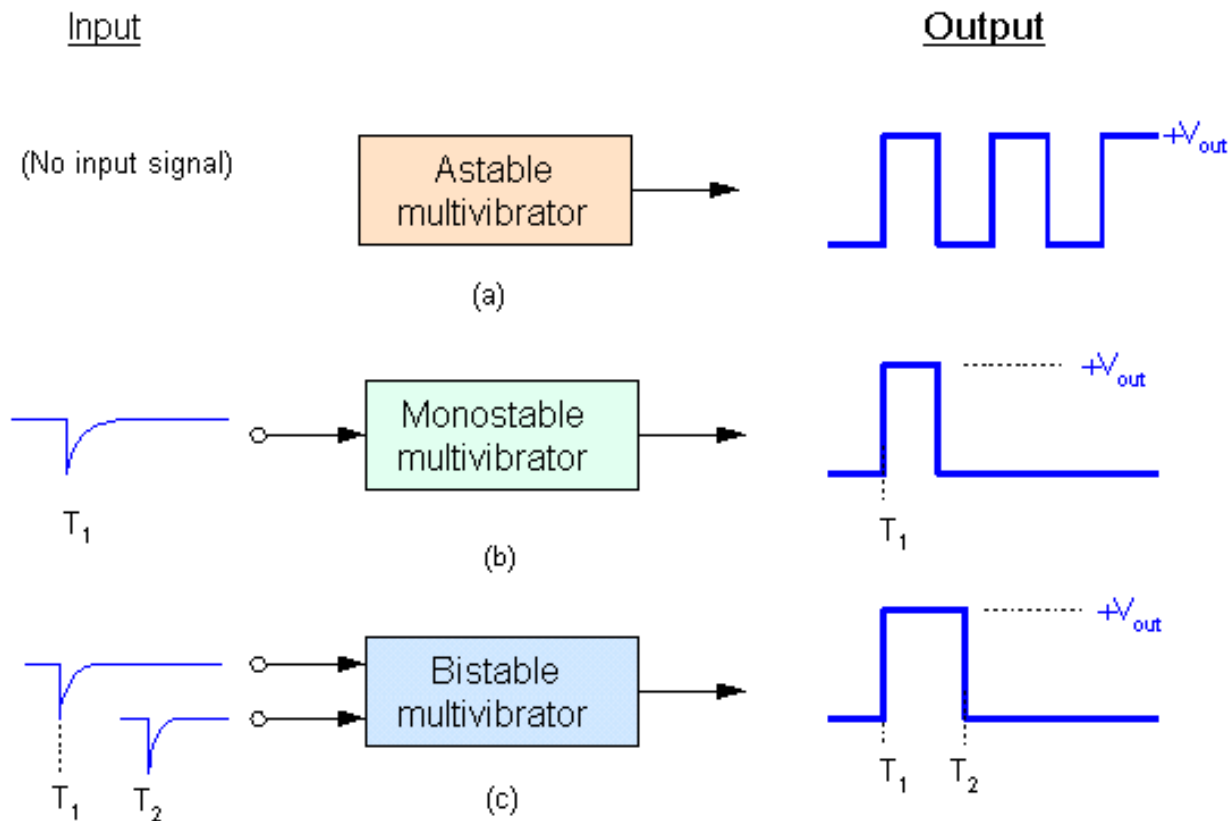
Photocoupler : Applications

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

MULTIVIBRATORS USING IC-555

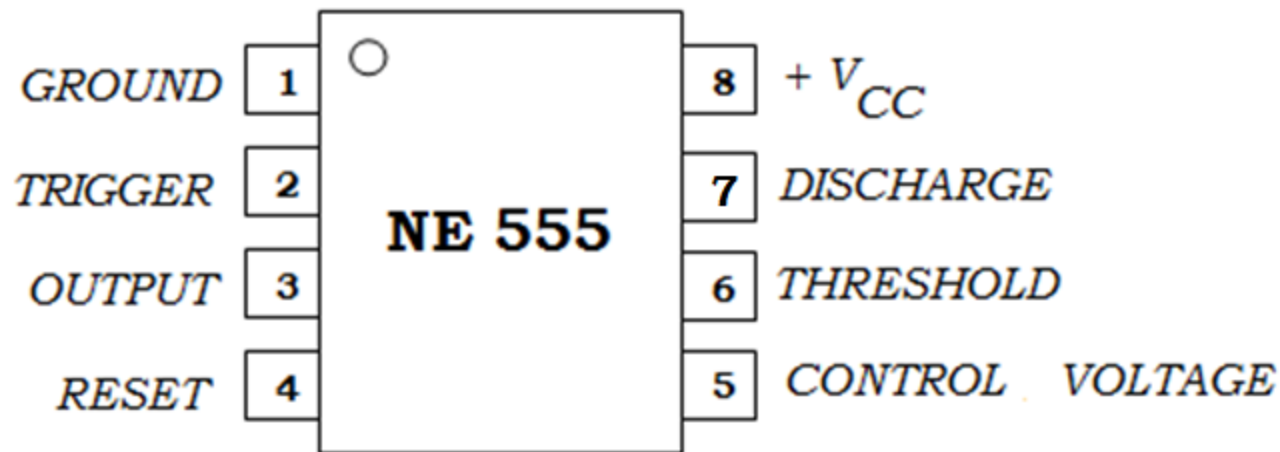
- A *multivibrator* circuit oscillates between a “HIGH” state and a “LOW” state producing a continuous output.
- It generates square, rectangular, pulse waveforms, also called nonlinear oscillators or function generators.
- There are basically three types of clock pulse generation circuits:
 - **Astable** – A *free-running multivibrator* that has **NO** stable states but switches continuously between two states this action produces a train of square/rectangular wave pulses at a fixed frequency.
 - **Monostable** – A *one-shot multivibrator* that has only **ONE** stable state and is triggered externally with it returning back to its first stable state.
 - **Bistable** – A *flip-flop* that has **TWO** stable states that produces a single pulse either positive or negative in

MULTIVIBRATORS USING IC-555

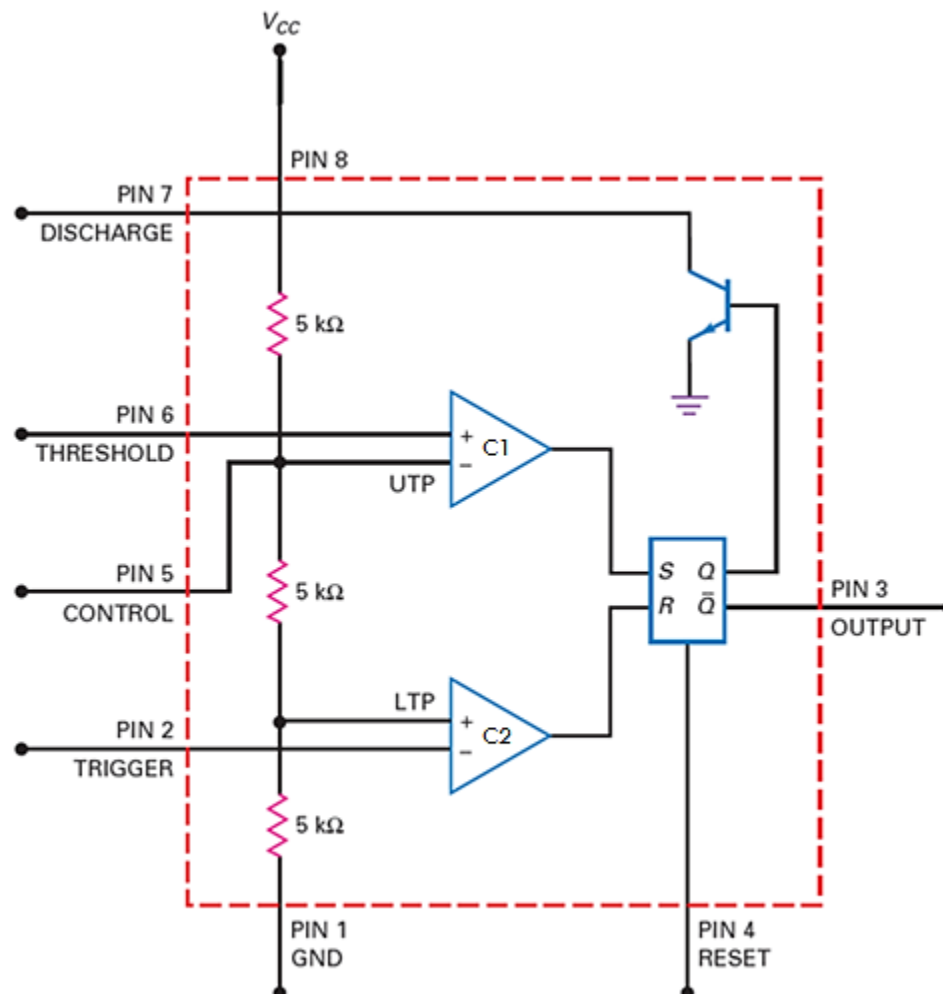


Integrated Circuit(IC) Multivibrators

- The NE555 (also LM555, CA555) is a widely used *IC timer*, a circuit that can run in either of two modes: **monostable** (one stable state) or **astable** (no stable states)



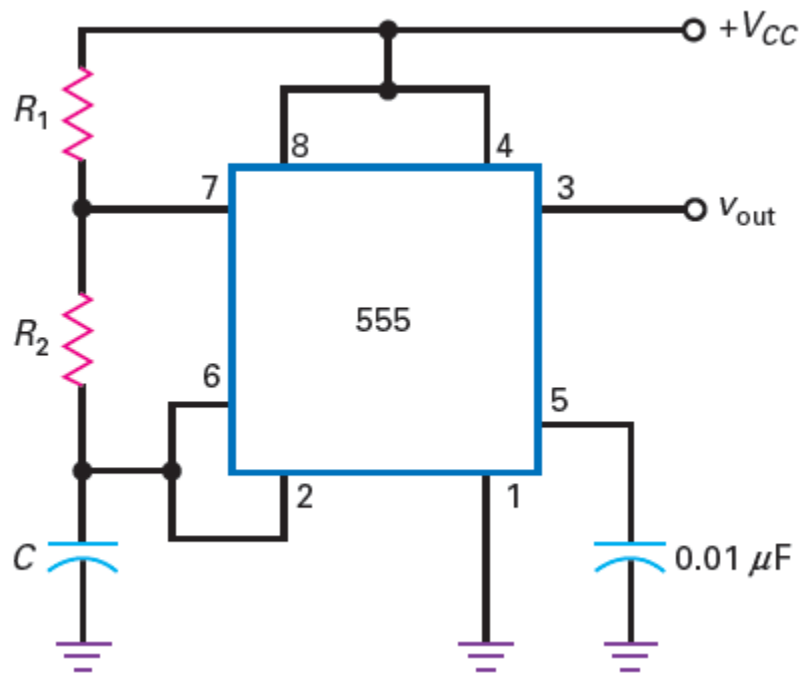
Integrated Circuit(IC) Multivibrators



Integrated Circuit(IC) Multivibrators

Astable Operation of the 555 Timer

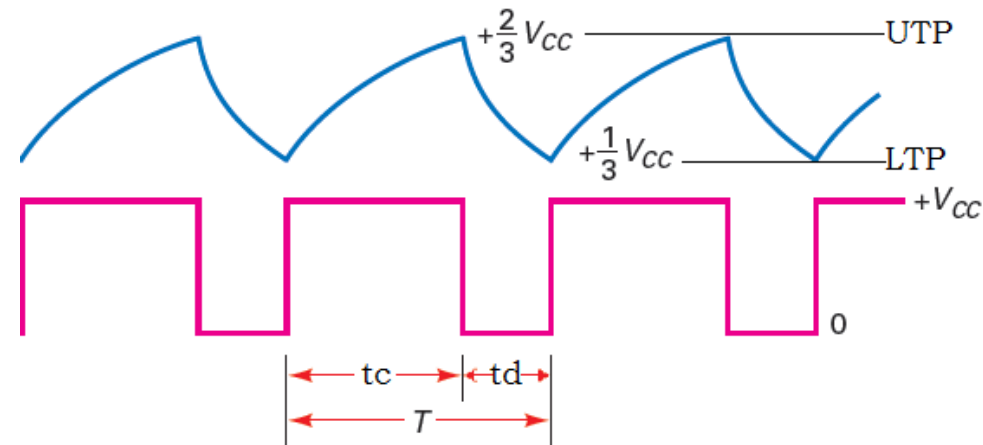
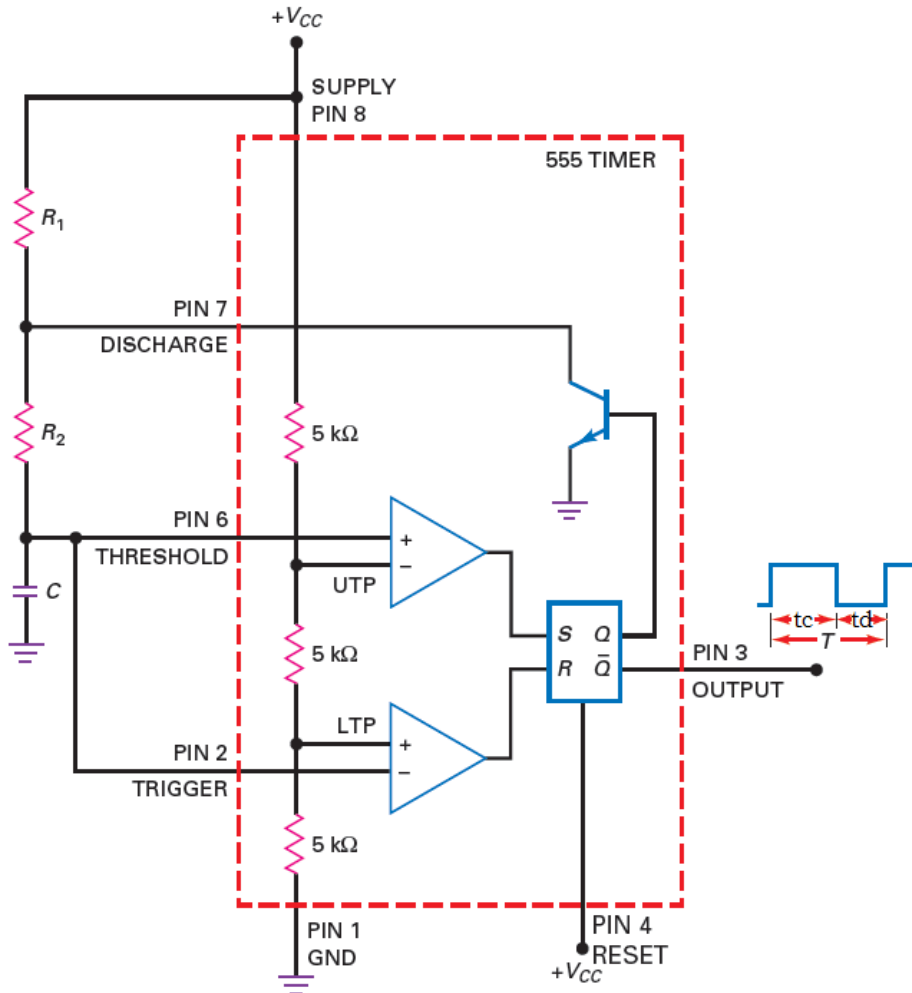
- Circuit Diagram



- Charge time (High Time)
 - $t_c = 0.693(R_1 + R_2)C$
- Discharge time (Low Time)
 - $t_d = 0.693R_2C$
- Total Time period T is
 - $T = t_c + t_d$
- The frequency is given by
 - $f_o = \frac{1}{T}$
- The duty cycle is
 - $\%D = \frac{t_c}{T}$

Integrated Circuit(IC) Multivibrators

Astable Operation of the 555 Timer



Capacitor and output waveforms

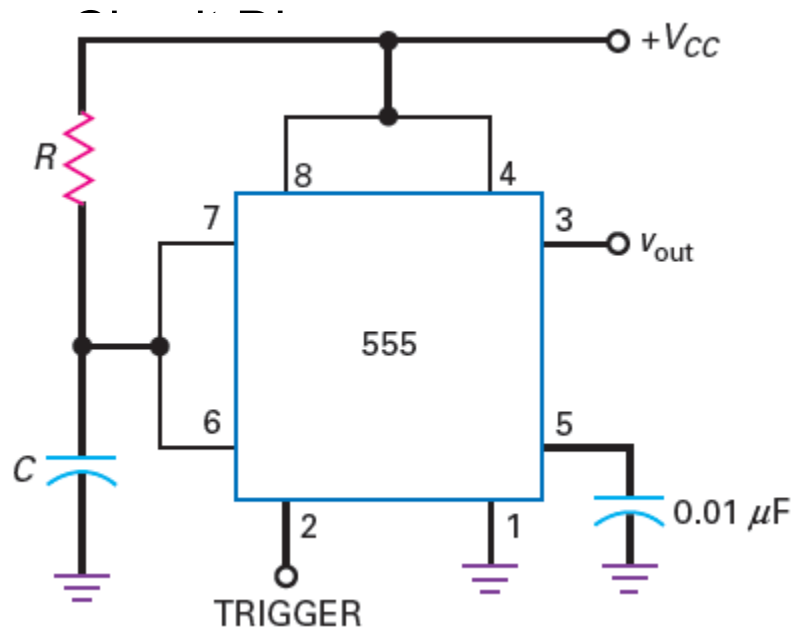
Integrated Circuit(IC) Multivibrators

Astable Operation of the 555 Timer

- When Q is low, the transistor is cut off and the capacitor is charging through R1 and R2 resistance. Because of this, the charging time constant is $(R1+R2)C$. As the capacitor charges, the threshold voltage (pin 6) increases. Eventually, the threshold voltage exceeds $\frac{2}{3}V_{CC}$. Then, the upper comparator sets the flip-flop.
- With Q high, the transistor saturates and grounds pin 7. The capacitor now discharges through R2. Therefore, the discharging time constant is $R2C$. When the capacitor voltage drops to slightly less than $\frac{1}{3}V_{CC}$, the lower comparator resets the flip-flop.
- The output is a rectangular wave that swings between 0 and V_{CC} . Since the charging time constant is longer than the discharging time constant, the output is nonsymmetrical. Depending on resistances R1 and R2, the duty cycle is between 50 and 100 percent.
- When R1 is much smaller than R2, the duty cycle approaches 50 percent. Conversely, when R1 is much greater than R2, the duty cycle approaches 100 percent.
- To make the duty cycle to become less than 50 percent. By placing a diode in parallel with R2 (anode connected to pin 7), the capacitor will effectively charge through R1 and the diode. The capacitor will discharge through R2.

Integrated Circuit(IC) Multivibrators

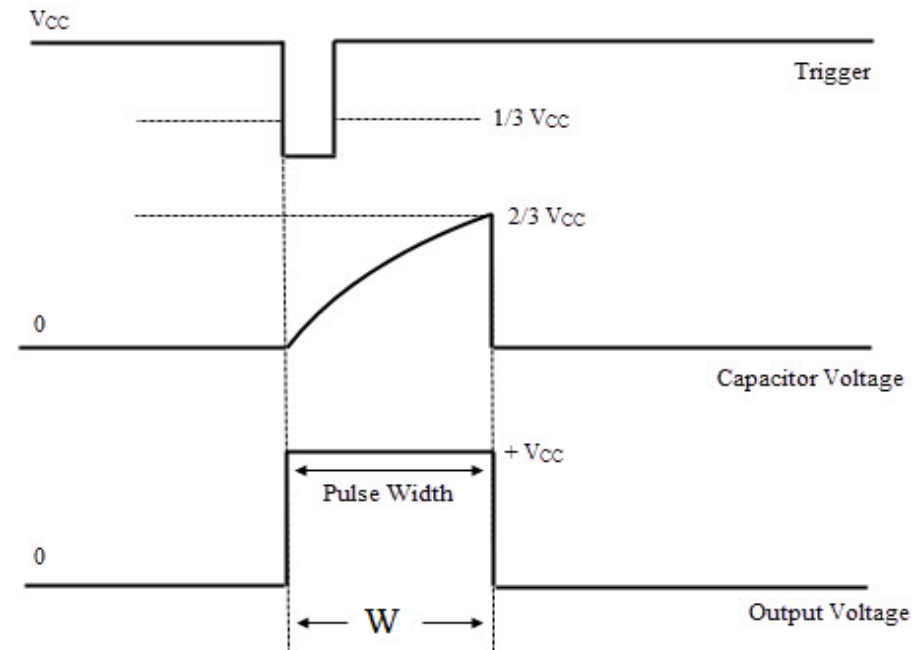
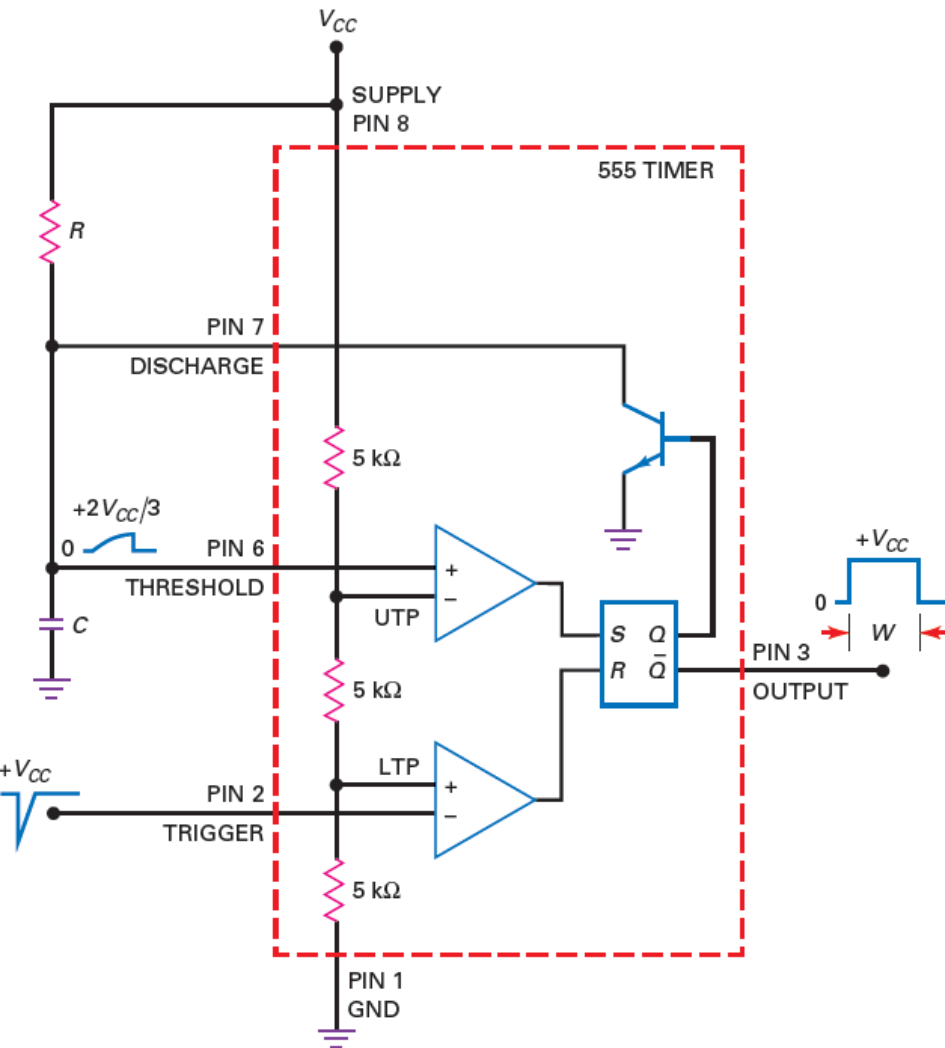
Monostable Operation of the 555 Timer



- Pulse Width is
 - $W=1.1RC$
- The circuit has an external resistor R and a capacitor C .
- The voltage across the capacitor is used for the threshold voltage to pin 6.
- When the trigger arrives at pin 2, the circuit produces a rectangular output pulse from pin 3.

Integrated Circuit(IC) Multivibrators

Monostable Operation of the 555 Timer

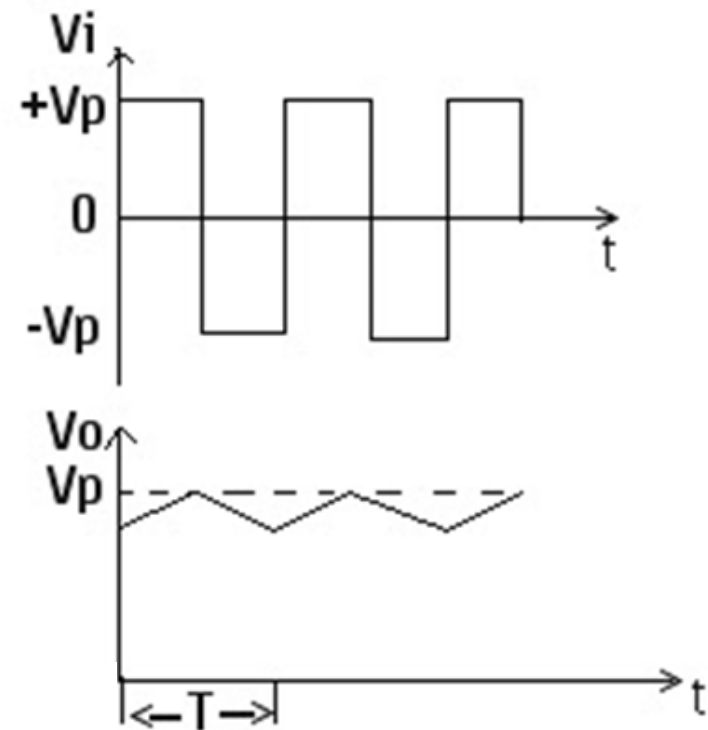
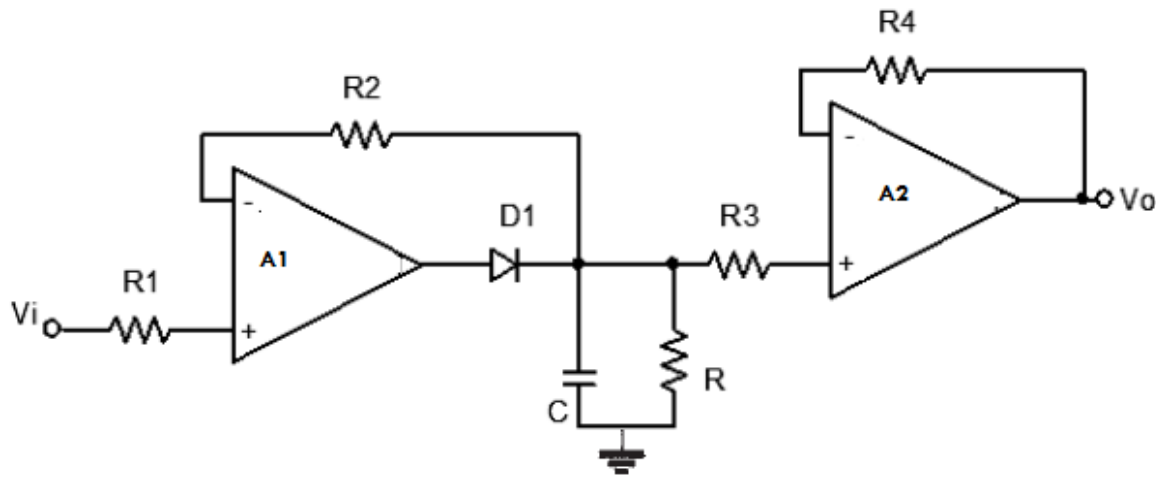


Integrated Circuit(IC) Multivibrators

Monostable Operation of the 555 Timer

- Initially, the Q output of the RS flip-flop is high. This turns ON the transistor and the capacitor discharges to ground through pin 7. The circuit will remain in this state until a trigger arrives at pin 2.
- When the trigger input falls to slightly less than $\frac{1}{3}V_{CC}$, the lower comparator resets the flip-flop. Since Q has changed to low, the transistor goes OFF, allowing the capacitor to charge. At this time, \bar{Q} has changed to high.
- The capacitor now charges exponentially through R as shown in waveform. When the capacitor voltage is slightly greater than $\frac{2}{3}V_{CC}$, the upper comparator sets the flip-flop. The high Q turns ON the transistor, which discharges the capacitor almost instantly. At the same instant, \bar{Q} returns to the low state and the output pulse ends.
- \bar{Q} remains low until another input trigger arrives.

Peak Detector Circuit



- $RC \geq 10 T$

Applications:

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

Peak Detector Circuit

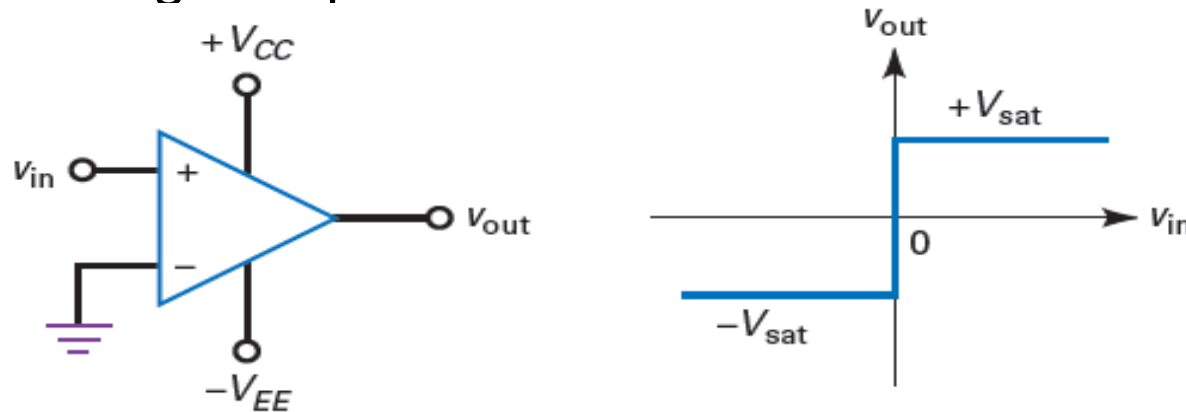
- During +ve half cycle when the input voltage is positive, the diode is conducting/ON and capacitor charges to the peak of the input voltage.
- Second, when the input voltage is negative during –ve half cycle, the diode is non-conducting/OFF and the capacitor discharges through the load resistor.
- As long as the discharging time constant is much greater than the period of the input signal (T), the output voltage will be approximately equal to the peak value of the input voltage.
- This can be achieved by making discharging time constant RC can be made much longer than the period of the input signal ($RC \geq 10 T$), will get almost perfect peak detection of low-level signals.
- If the peak-detected signal has to drive a small load, to avoid loading effects by connecting the voltage follower (op-amp buffer) isolates the small load resistor from the peak detector. This prevents the small load resistor from discharging the capacitor too quickly.

Comparator

- Comparator circuit compares a single voltage on one input of op-amp with a known voltage called reference voltage (Trip point or trigger point) on the other input and produces high or low output depending upon relative magnitude of two input.
 - Comparators with Zero Reference
 - Comparators with Nonzero References
 - Comparators with Hysteresis or Schmitt Trigger
 - Window Comparator

Comparators with Zero Reference

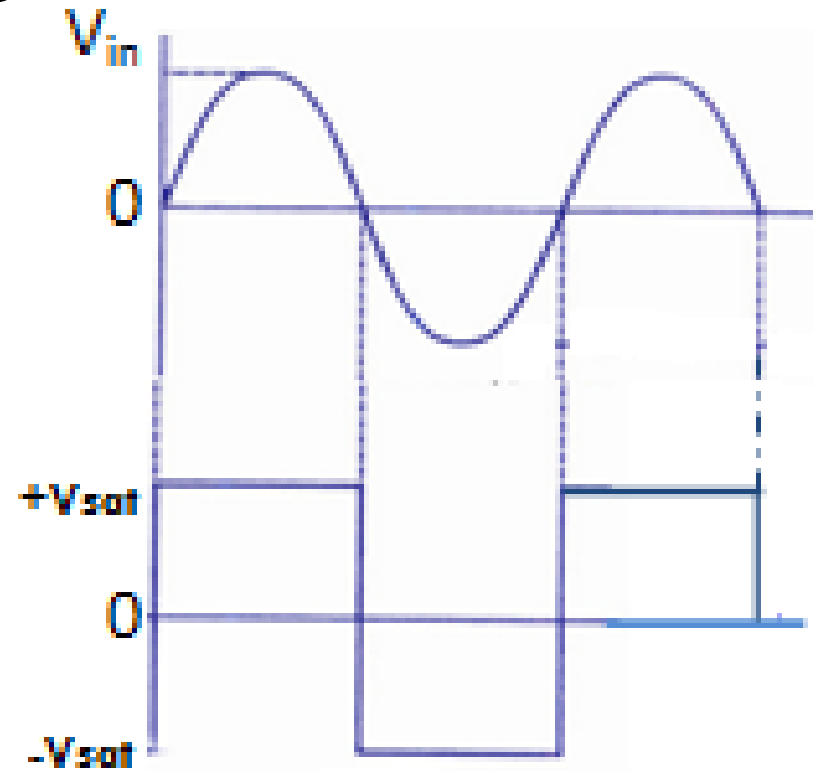
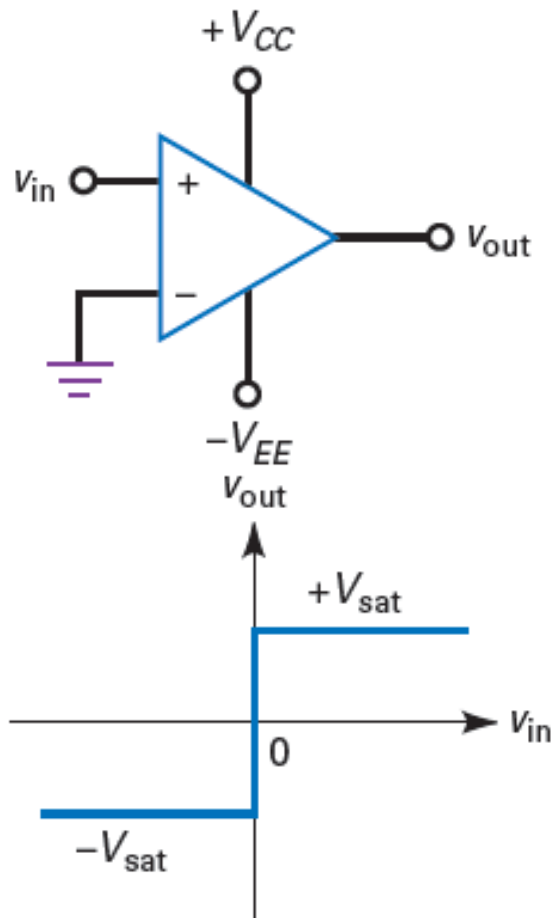
- Non-Inverting Comparator



- Because of the high open-loop voltage gain, a positive input voltage produces positive saturation, and a negative input voltage produces negative saturation.
- Above circuit is called a **zero-crossing detector** because the output voltage ideally switches from low to high or vice versa whenever the input voltage crosses zero (input compares with zero reference voltage).

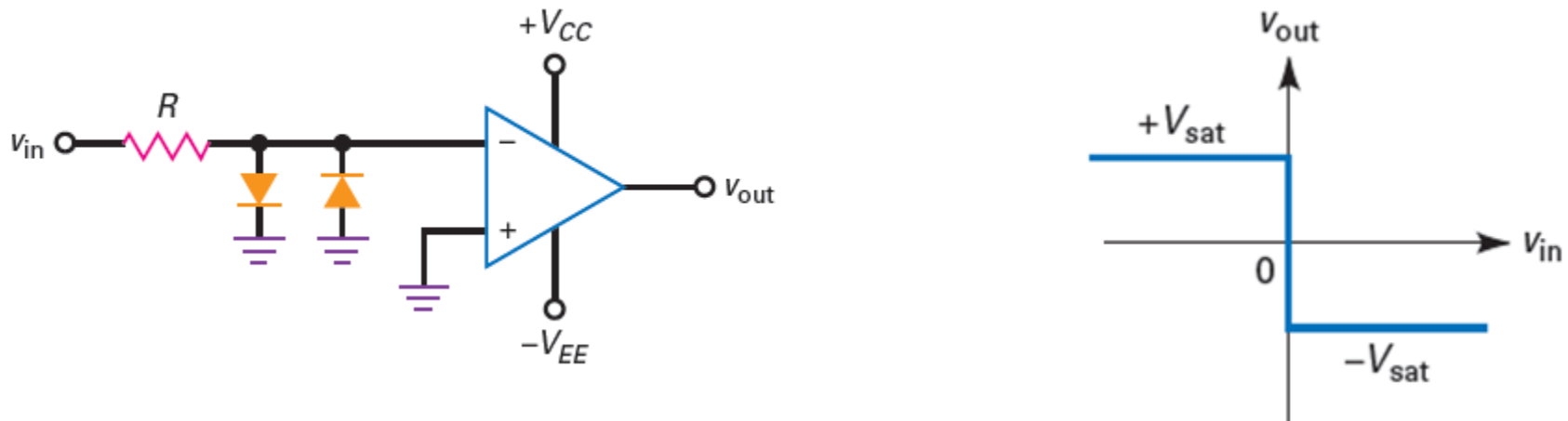
Comparators with Zero Reference

- Non-Inverting Comparator



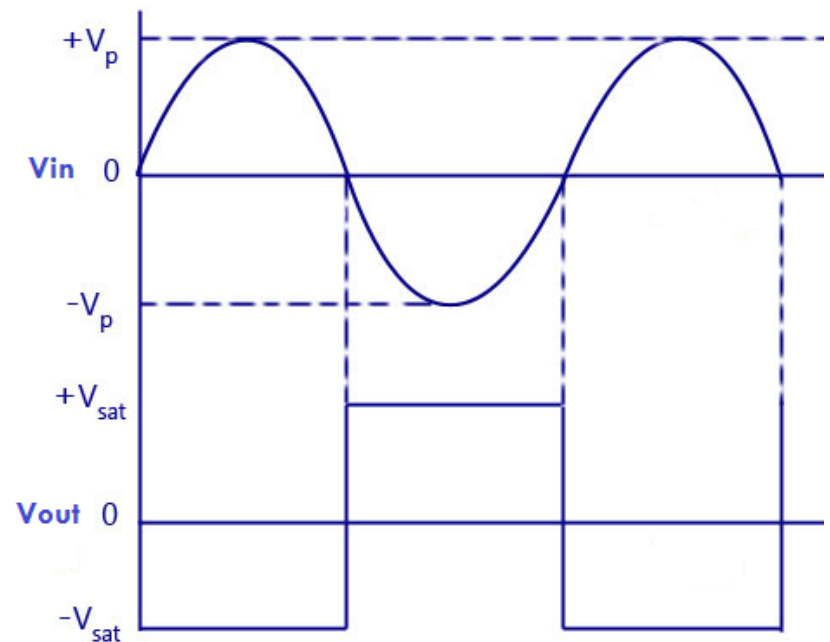
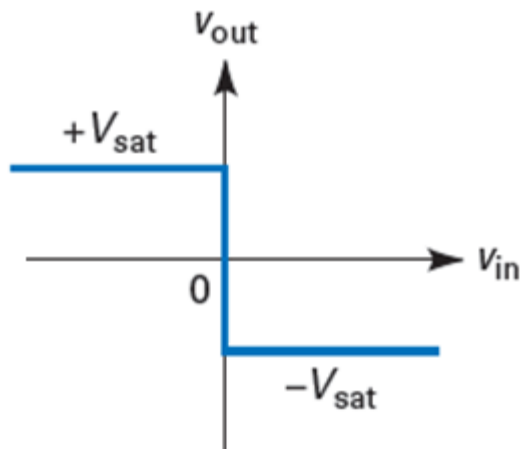
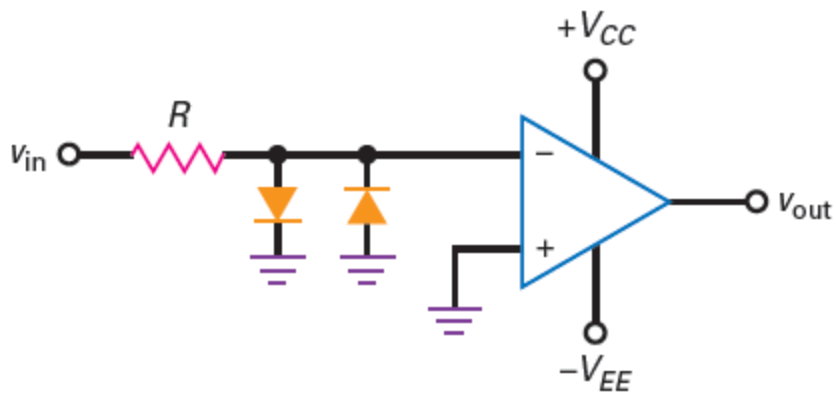
Comparator

Comparators with Zero Reference



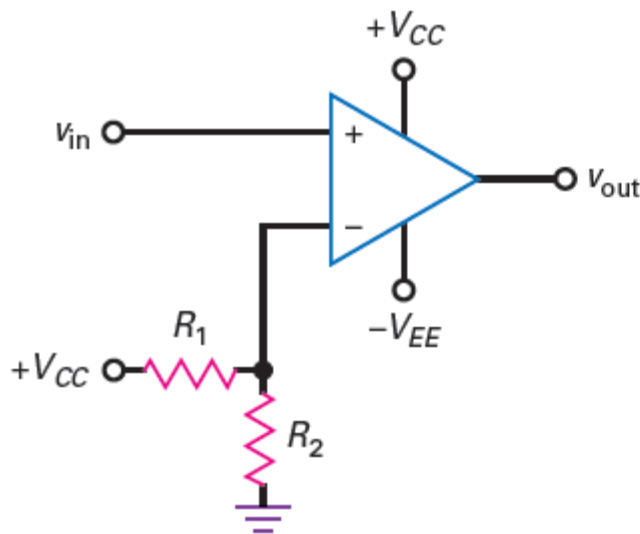
- The input signal drives the inverting input of the comparator. In this case, a positive input voltage produces a maximum negative saturation, as shown in above diagram. On the other hand, a negative input voltage produces a maximum positive saturation.

Comparators with Zero Reference

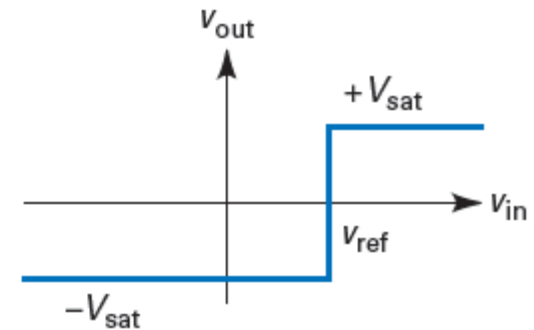


Comparators with Nonzero References

- Non-Inverting Comparator: Positive reference



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

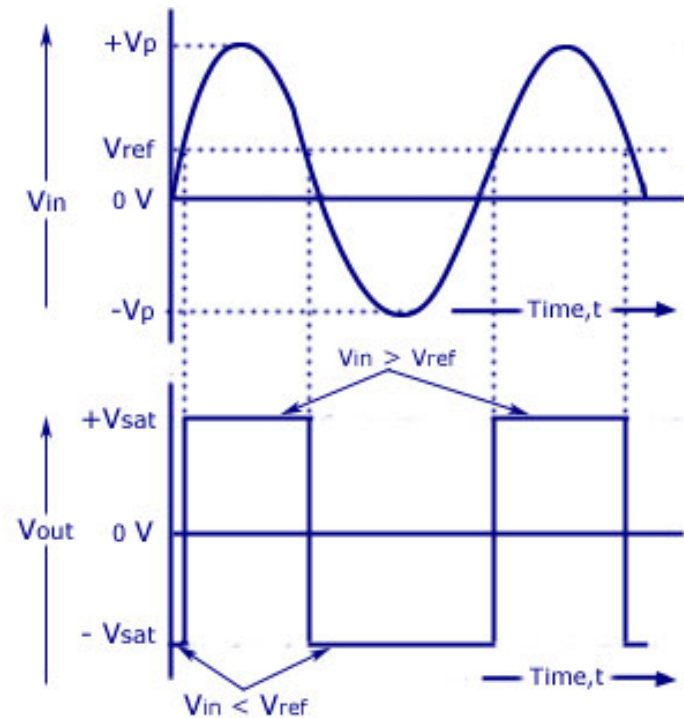
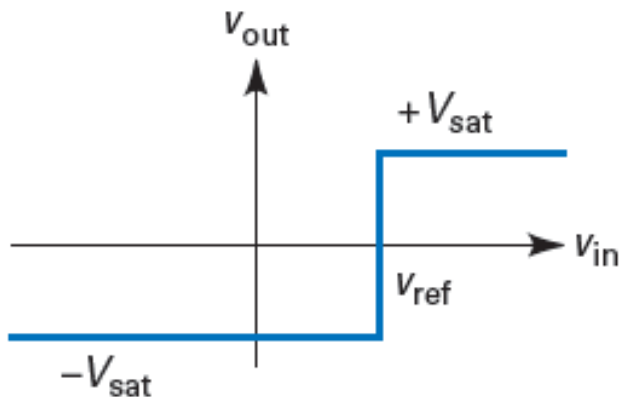


is positive and the output voltage is high ($+V_{sat}$). When V_{in} is less than V_{ref} , the differential input voltage is negative and the output voltage is low ($-V_{sat}$).

- $V_{in} > V_{ref}$ then $V_{out} = +V_{sat}$
- $V_{in} < V_{ref}$ then $V_{out} = -V_{sat}$

Comparators with Nonzero References

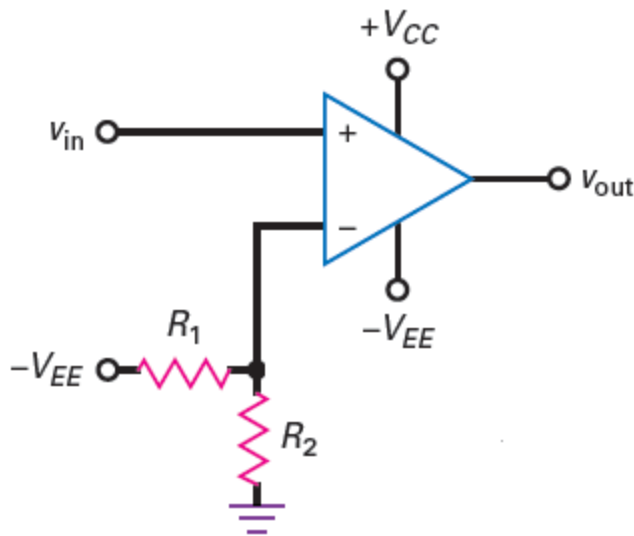
- Non-Inverting Comparator: Positive reference



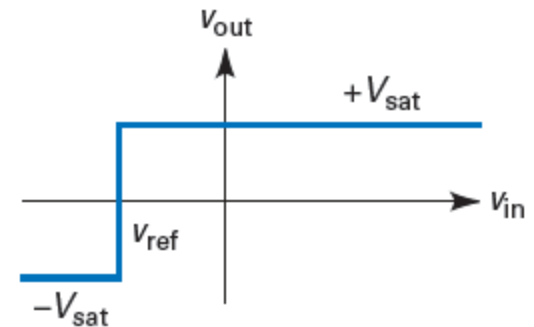
Input and Output Waveforms
For Positive V_{ref}

Comparators with Nonzero References

- Non-Inverting Comparator: Negative reference



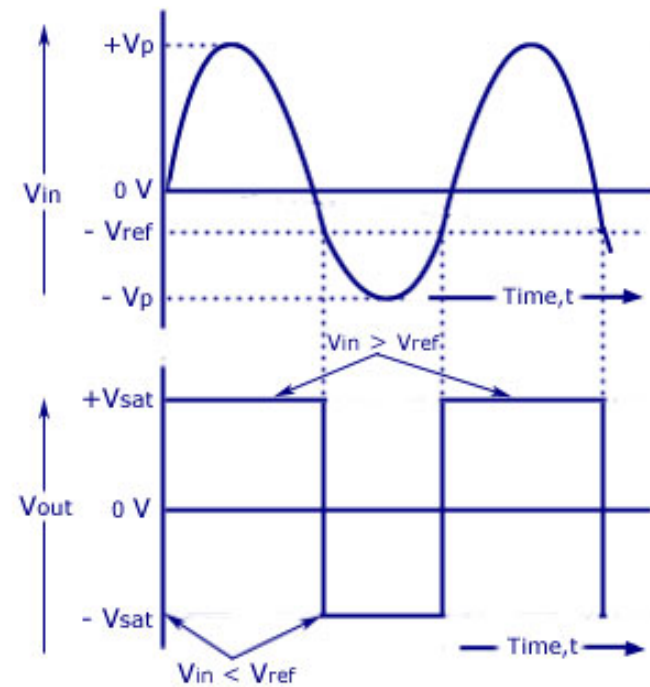
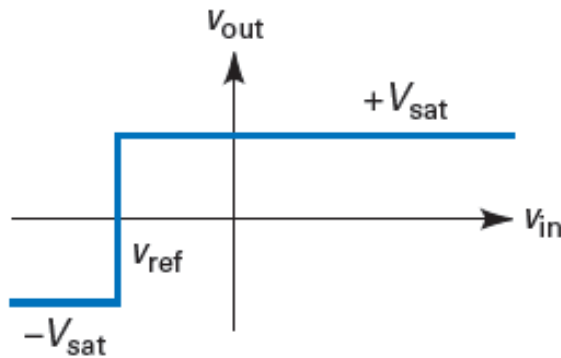
$$v_{ref} = \frac{R_2}{R_1 + R_2} -V_{EE}$$



- $V_{in} > V_{ref}$ then $V_{out} = +V_{sat}$
- $V_{in} < V_{ref}$ then $V_{out} = -V_{sat}$

Comparators with Nonzero References

- Non-Inverting Comparator: Negative reference

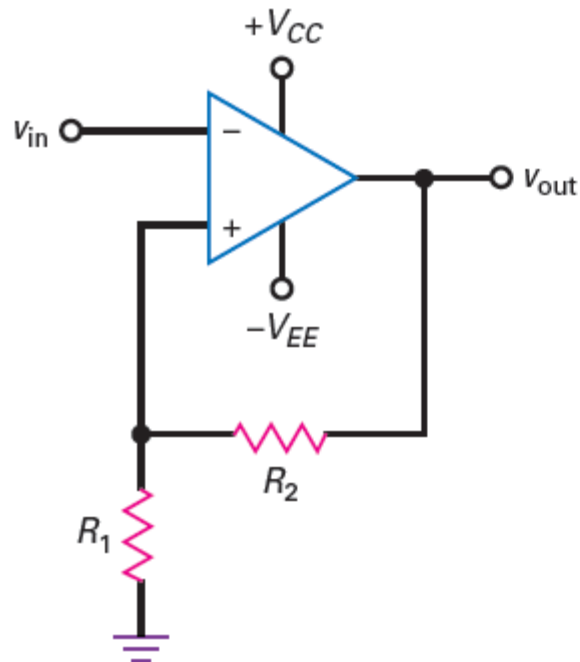


Input and Output Waveforms
For Negative V_{ref}

Comparators with Hysteresis /Schmitt Trigger

- **Inverting Schmitt trigger**
- When the comparator is positively saturated, a positive voltage is fed back to the noninverting input. This positive feedback voltage holds the output in the high state.
- Similarly, when the output voltage is negatively saturated, a negative voltage is fed back to the noninverting input, holding the output in the low state.

Schmitt Trigger

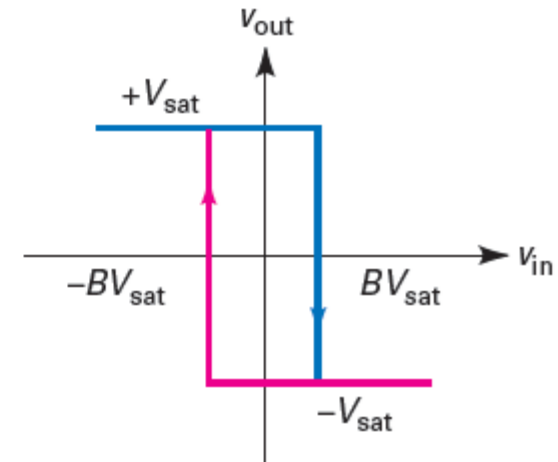


$$B = \frac{R_1}{R_1 + R_2}$$

$$\text{UTP} = BV_{\text{sat}}$$

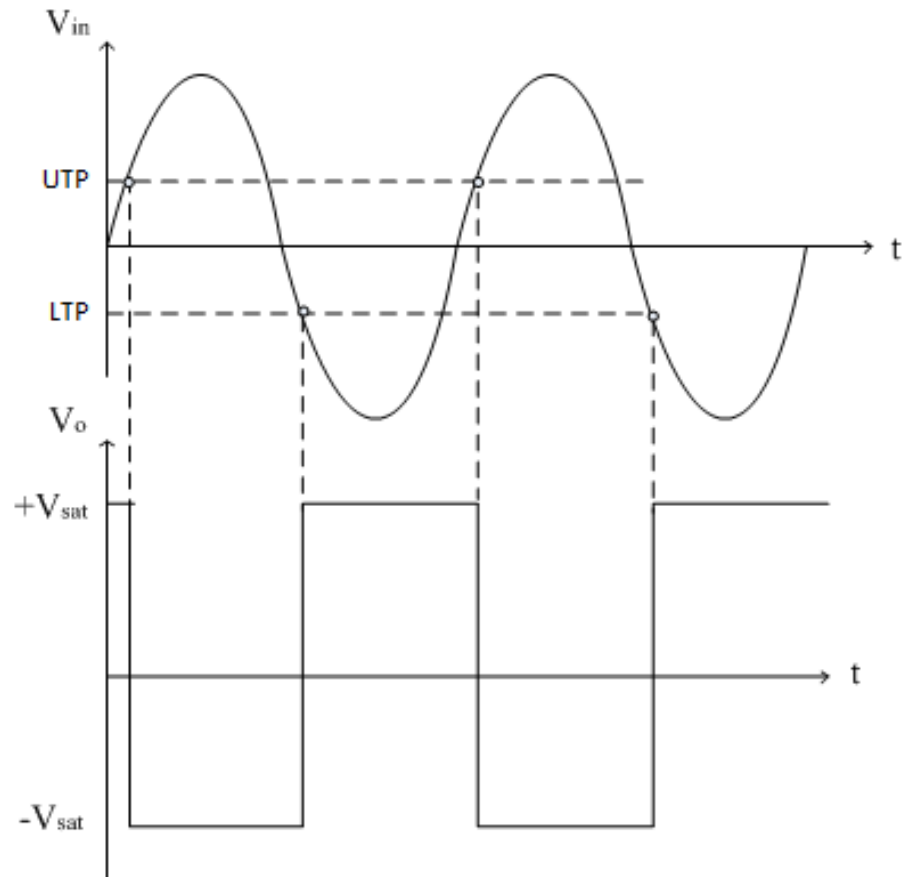
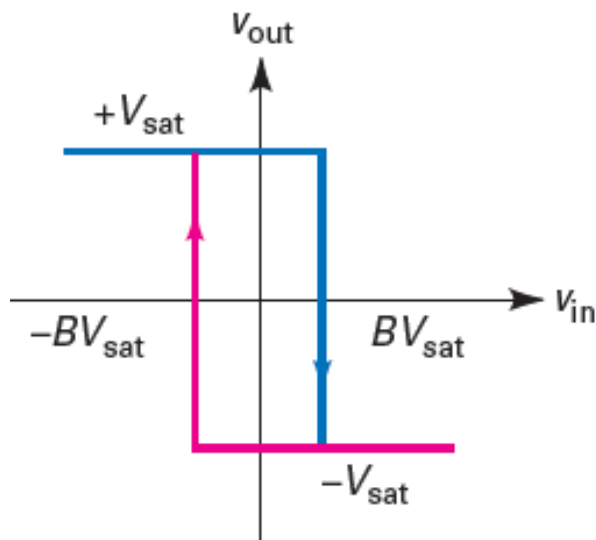
$$\text{LTP} = -BV_{\text{sat}}$$

$$V_H = 2BV_{\text{sat}}$$



Schmitt Trigger

- Inverting Schmitt trigger**



Schmitt Trigger

- **Inverting Schmitt trigger**
- The output voltage will remain in a given state until the input voltage exceeds the reference voltage for that state.
- For instance, if the output is positively saturated, the reference voltage is $+BV_{sat}$. The input voltage must be increased to slightly more than $+BV_{sat}$ to switch the output voltage from positive to negative, as shown in input/output response has hysteresis.
- Once the output is in the negative state, it will remain there indefinitely until the input voltage becomes more negative than $-BV_{sat}$. Then, the output switches from negative to positive shown in input/output response has hysteresis.

Schmitt Trigger

Applications of Schmitt Trigger:

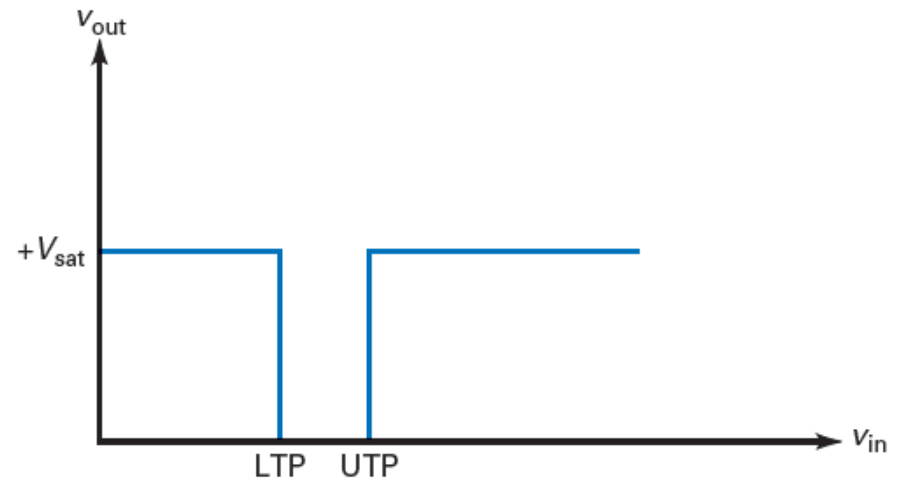
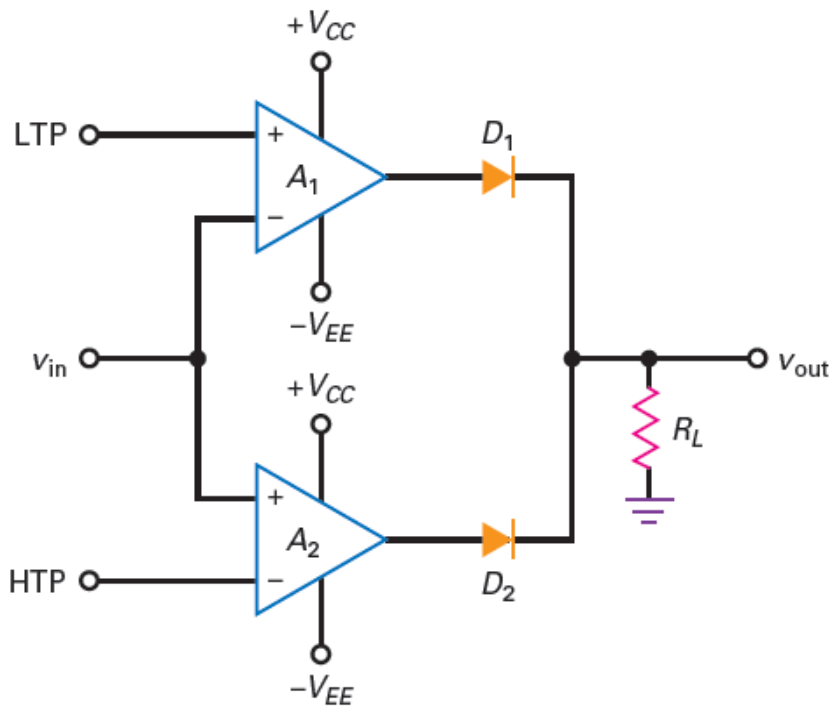
Schmitt trigger is used in many applications, where level needs to be sensed. Hysteresis is used to reduce the multiple transitions that can occur around.

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

Window Comparator

- An ordinary comparator indicates when the input voltage exceeds a certain limit or threshold.
- A **window comparator** (also called a *double-ended limit detector*) detects when the input voltage is between two limits called the *window*. To create a window comparator, will use two comparators with different thresholds.

Window Comparator



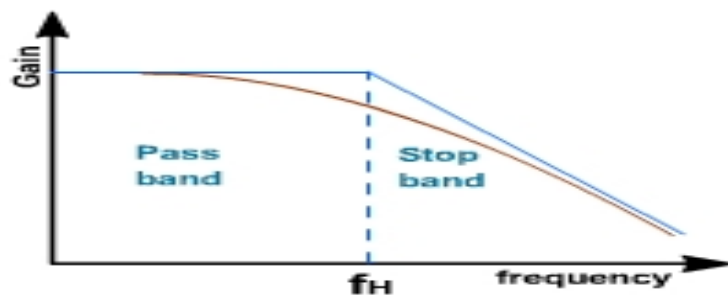
Window Comparator

- **Low Output between Limits**
- Circuit shows a window comparator that can produce a low output voltage when the input voltage is between a lower and an upper limit.
- When V_{in} is less than LTP or greater than UTP , the output is high. When V_{in} is between LTP and UTP , the output is low.
- Operation:
 - When $V_{in} < LTP$, comparator $A1$ has a positive output and $A2$ has a negative output. Diode $D1$ is on and $D2$ is off. Therefore, the output voltage is high.
 - Similarly, when $V_{in} > UTP$, comparator $A1$ has a negative output and $A2$ has a positive output. Diode $D1$ is off, $D2$ is on, and the output voltage is high.
 - When $LTP < V_{in} < UTP$, $A1$ has a negative output, $A2$ has a negative output, $D1$ is off, $D2$ is off, and the output voltage is low.

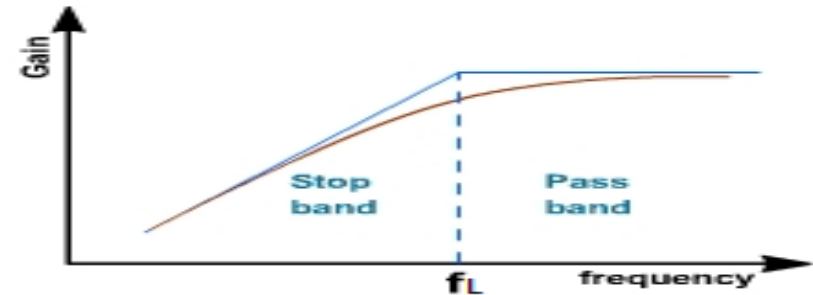
Active Filters

- An electric filter is often a frequency selective circuit that passes a specified band of frequency and blocks or attenuates signals of frequencies outside this band.
- Active filters employs transistor or op-amp in addition to resistor and capacitor.
- RC network are used for filter.
- The most commonly used filters are follows:
 - Low pass filters
 - High pass filter
 - Band pass filter
 - Band reject filter.
 - All pass filter
- Next slide shows the frequency response characteristics of the five types of filter. The ideal response is shown by dashed line. While the solid lines indicates the practical filter response.

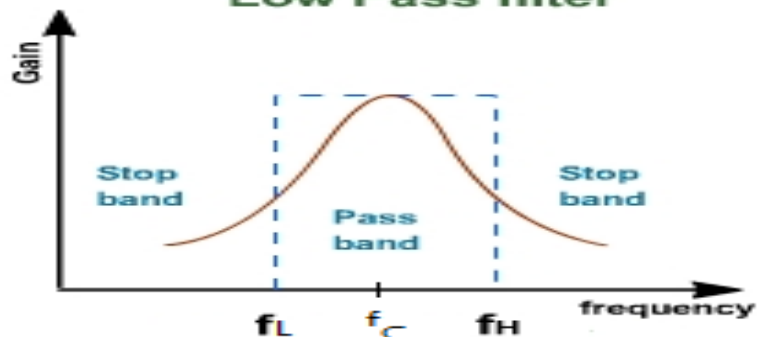
Active Filters



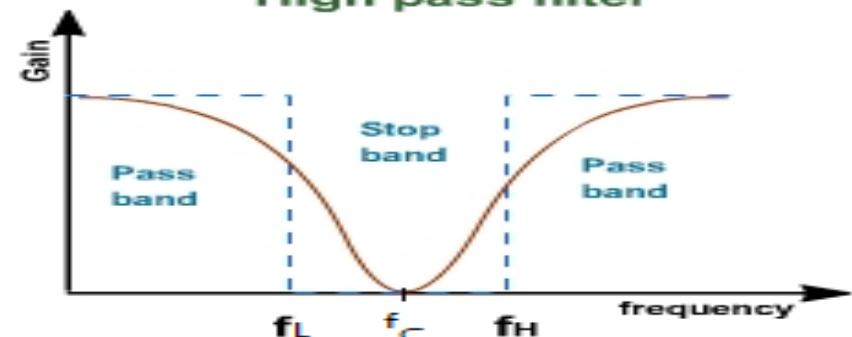
Low Pass filter



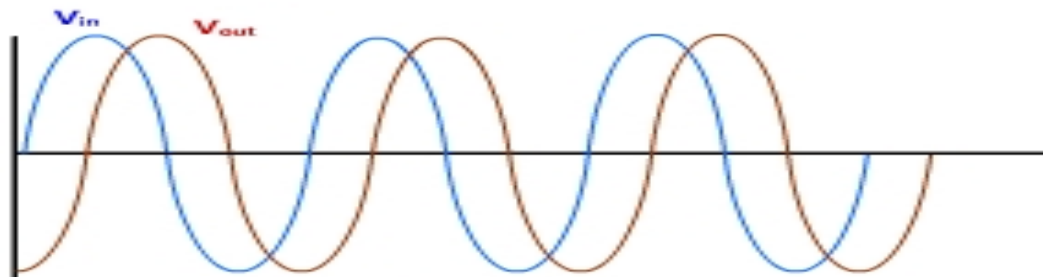
High pass filter



Band pass filter



Band reject filter



All pass filter

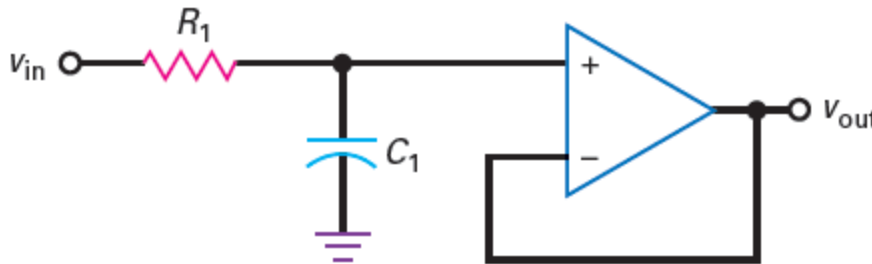
Phase shift between input and output of all pass filter

Active Filters

- A filter that provides a constant output from dc up to a cut-off frequency f_H and then passes no signal above that frequency is called an ideal low-pass filter.
- A filter that provides or passes signals above a cutoff frequency f_L is a high-pass filter, as shown in previous slide.
- When the filter circuit passes signals that are above one ideal cutoff frequency (f_L) and below a second cutoff frequency, (f_H) it is called a bandpass filter.
- Two types of filters
 - First Order Filter – One capacitor used
 - Second Order Filter – Two or more capacitor used

Active Filters- Low-pass filter

- **Non-Inverting unity gain**

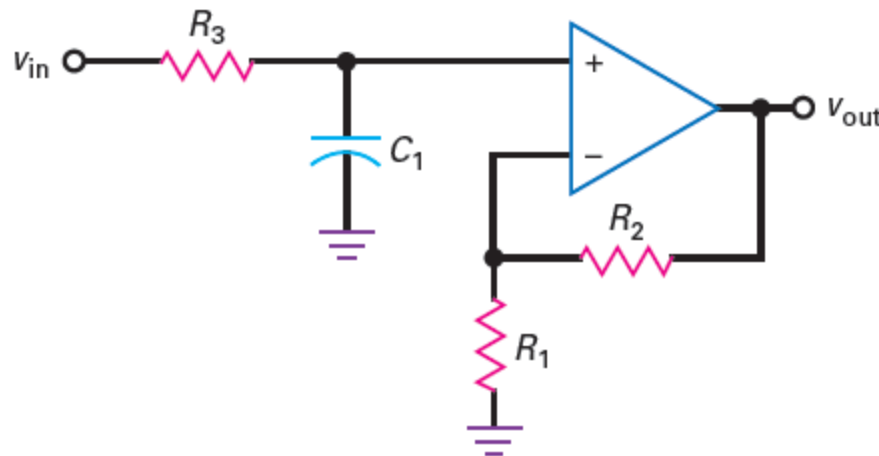


$$A_v = 1$$
$$f_c = \frac{1}{2\pi R_1 C_1}$$

- It is nothing more than an RC lag circuit and a voltage follower. The voltage gain is: $A_v = 1$.
- When the frequency increases above the cutoff frequency, the capacitive reactance decreases and reduces the noninverting input voltage.
- Since the R_1C_1 lag circuit is outside the feedback loop, the output voltage rolls off. As the frequency approaches infinity, the capacitor becomes a short and there is zero input voltage.

Active Filters- Low-pass filter

- **Non-Inverting with voltage gain**

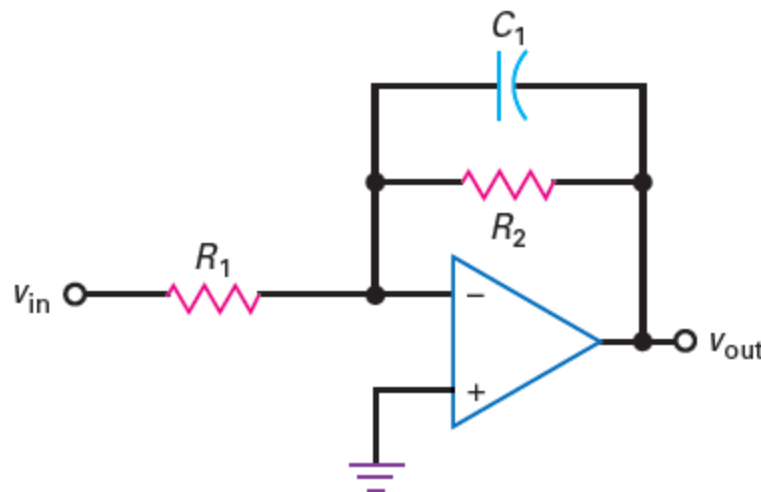


$$A_v = \frac{R_2}{R_1} + 1$$
$$f_c = \frac{1}{2\pi R_3 C_1}$$

- Although it has two additional resistors, it has the advantage of voltage gain.

Active Filters- Low-pass filter

- Inverting with voltage gain

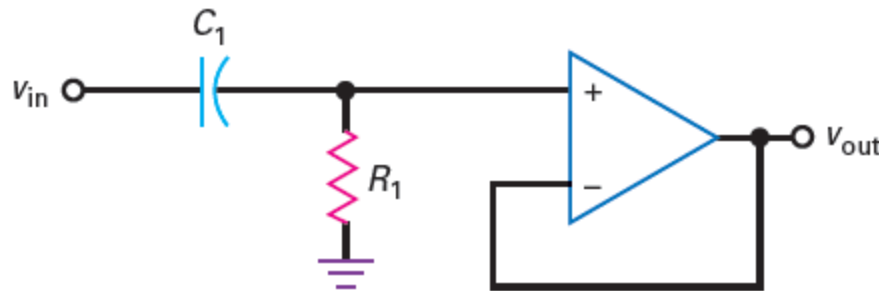


$$A_v = \frac{-R_2}{R_1}$$
$$f_c = \frac{1}{2\pi R_2 C_1}$$

- As the frequency increases, the capacitive reactance decreases and reduces the impedance of the feedback branch. This implies less voltage gain.
- As the frequency approaches infinity, the capacitor becomes a short and there is no voltage gain.

Active Filters- High-pass filter

- **Non inverting unity gain**

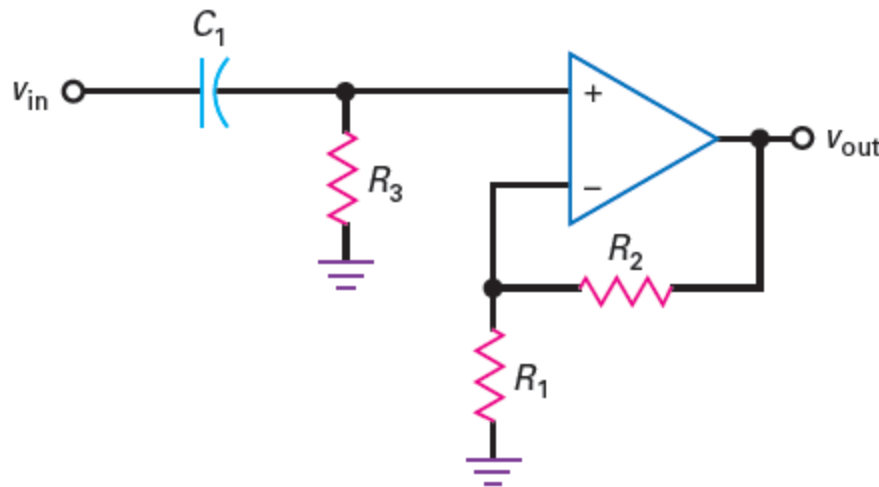


$$A_v = 1$$
$$f_c = \frac{1}{2\pi R_1 C_1}$$

- When the frequency decreases below the cutoff frequency, the capacitive reactance increases and reduces the noninverting input voltage.
- Since the $R_1 C_1$ circuit is outside the feedback loop, the output voltage rolls off. As the frequency approaches zero, the capacitor becomes an open and there is zero input voltage.

Active Filters- High-pass filter

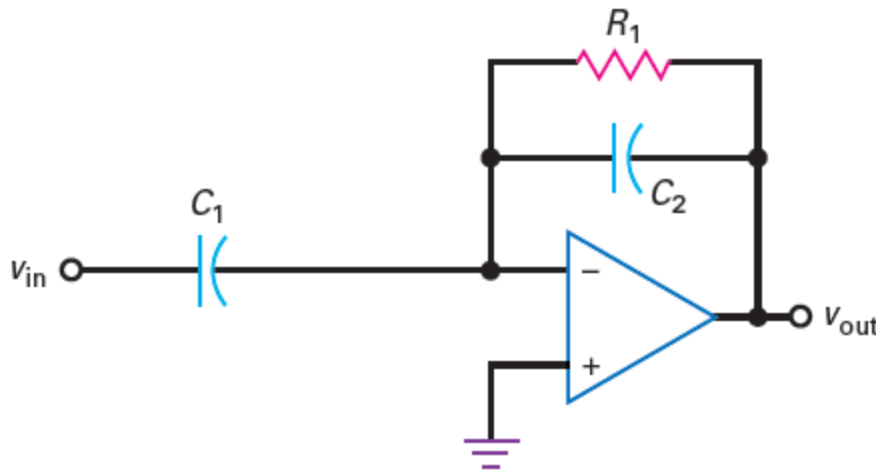
- Non-Inverting with voltage gain



$$A_v = \frac{R_2}{R_1} + 1$$
$$f_c = \frac{1}{2\pi R_3 C_1}$$

Active Filters- High-pass filter

- Inverting with voltage gain

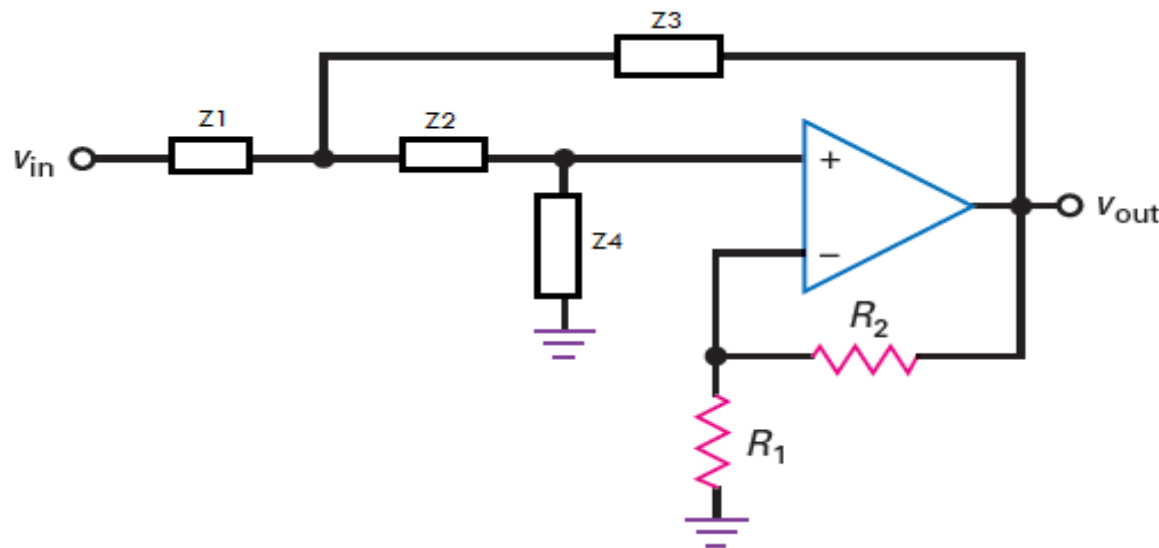


$$A_v = \frac{-C_1}{C_2}$$
$$f_c = \frac{1}{2\pi R_1 C_2}$$

Active Filters- Second Order Filter

Low Pass/High Pass Filter

- Generalized form of second order filter



$$A_v = \frac{R_2}{R_1} + 1$$

$$Q = \frac{1}{3 - A_v}$$

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

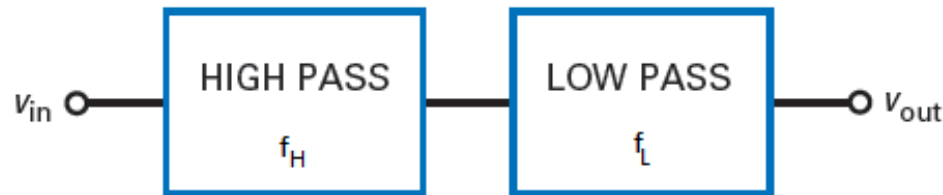
- If $Z1=Z2=R$ and $Z3=Z4=C$ get second order low pass filter
- If $Z1=Z2=C$ and $Z3=Z4=R$ get second order high pass filter

Active Filters- Band-pass Filter

- Two types of band pass filter
 - Wide band pass filter
 - Narrow band pass filter

Active Filters- Band-pass Filter

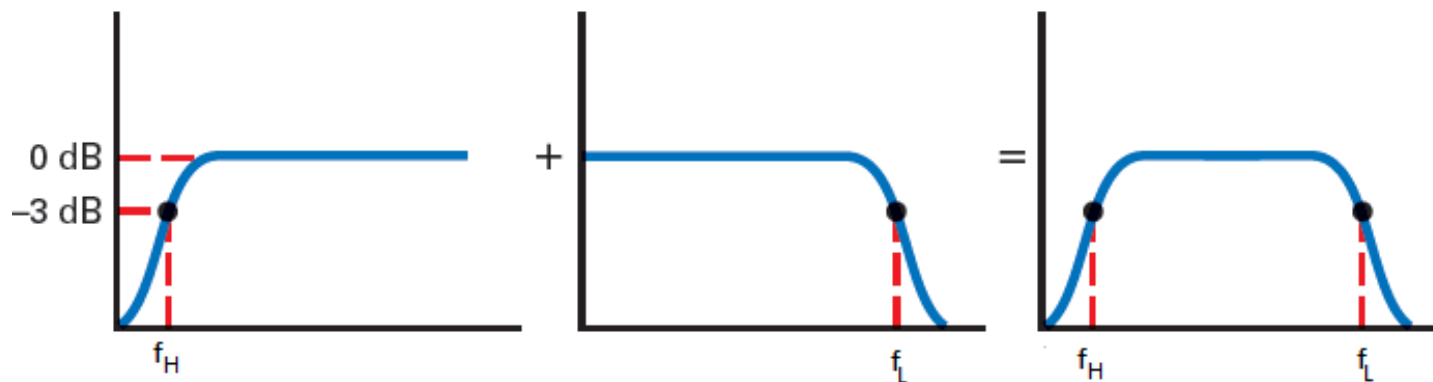
- **Wide Band Pass Filters**
 - Cascade of low-pass and high-pass filter



$$f_0 = \sqrt{f_1 f_2}$$

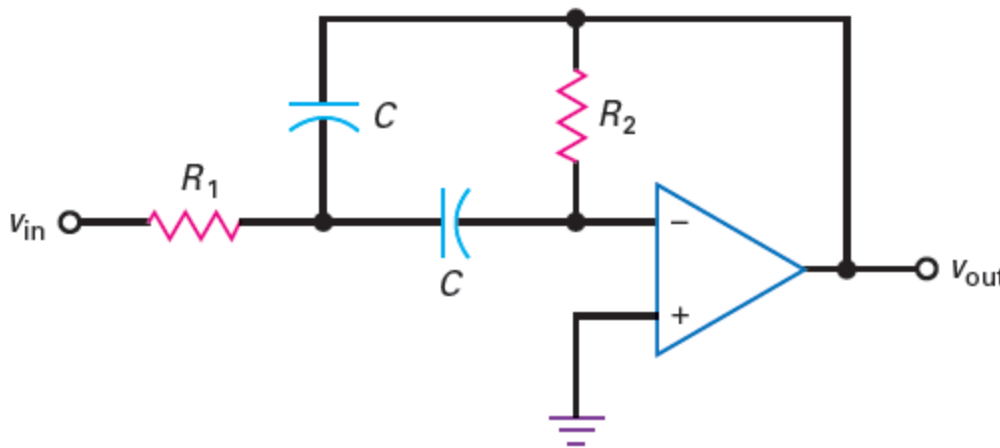
$$BW = f_2 - f_1$$

$$Q = \frac{f_0}{BW}$$



Active Filters- Band-pass Filter

- Narrow Band Pass Filters**



$$A_v = \frac{-R_2}{2R_1}$$

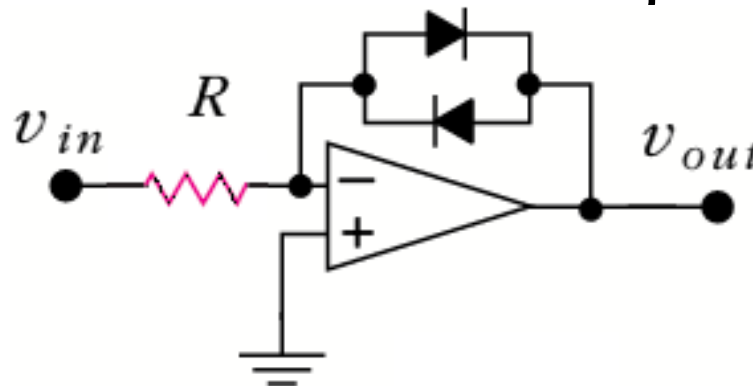
$$Q = 0.5 \sqrt{\frac{R_2}{R_1}}$$

$$f_0 = \frac{1}{2\pi C \sqrt{R_1 R_2}}$$

- In the circuit the input signal goes to the inverting input rather than the noninverting input. Also the circuit has two feedback paths, one through a capacitor and another through a resistor.

Non-Linear Amplifier

- In this amplifier the gain value is non-linear function of the amplitude of the input signal.
- The gain may be large for weak signal and very small for large signal this can be achieved using non-linear device such as PN junction diode as shown below.

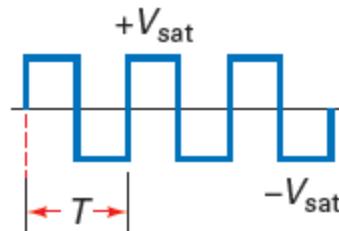
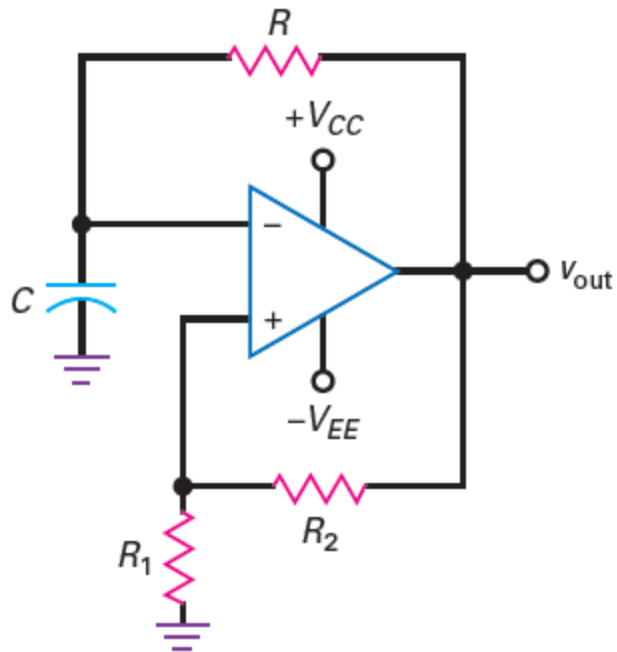


Non-Linear Amplifier

- **Working:**

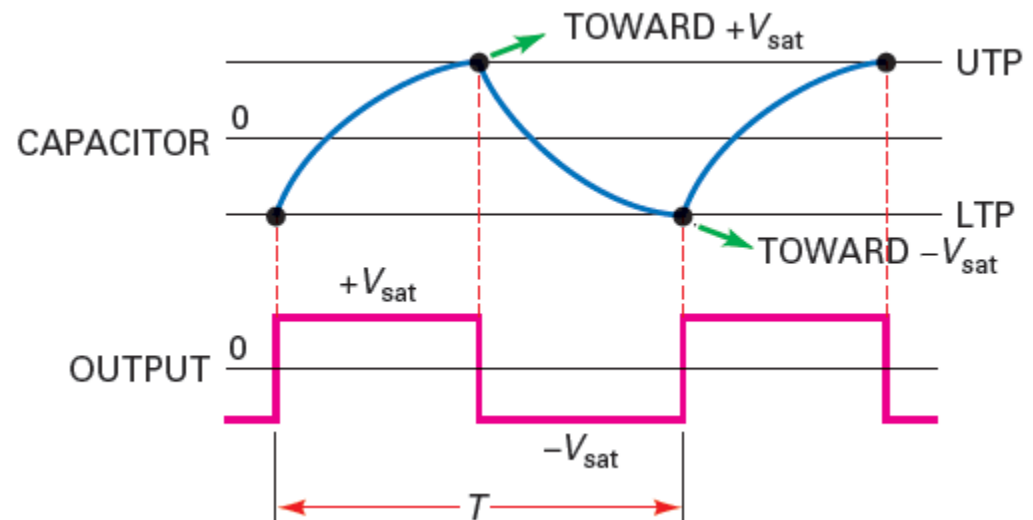
- For small value of input signal, diodes act as open circuit and the gain is high due to minimum feedback.
- When the amplitude of input signal is large, diodes offer very small resistance and thus gain is low.

Relaxation Oscillator



$$T = 2RC \ln \frac{1+B}{1-B}$$

$$B = \frac{R_1}{R_1 + R_2}$$

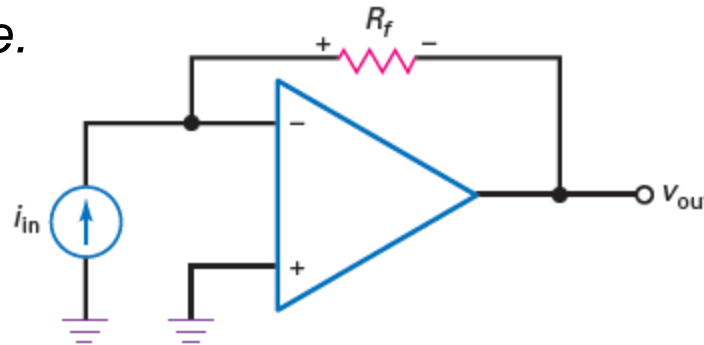


Relaxation Oscillator

- In circuit, there is no input signal.
- Nevertheless, the circuit produces a rectangular output signal. This output is a square wave that swings between $-V_{\text{sat}}$ and $+V_{\text{sat}}$. How is this possible?
- Assume that the output is in positive saturation. Because of feedback resistor R , the capacitor will charge exponentially toward $+V_{\text{sat}}$, as shown in waveform. But the capacitor voltage never reaches $+V_{\text{sat}}$ because the voltage crosses the UTP. When this happens, the output square wave switches to $-V_{\text{sat}}$.
- With the output now in negative saturation, the capacitor discharges, as shown in waveform. When the capacitor voltage crosses through zero, the capacitor starts charging negatively toward $-V_{\text{sat}}$. When the capacitor voltage crosses the LTP, the output square wave switches back to $+V_{\text{sat}}$. The cycle then repeats.

Current-To-Voltage Converter

- Also called transimpedance amplifier
- 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, $V_B = 0$.

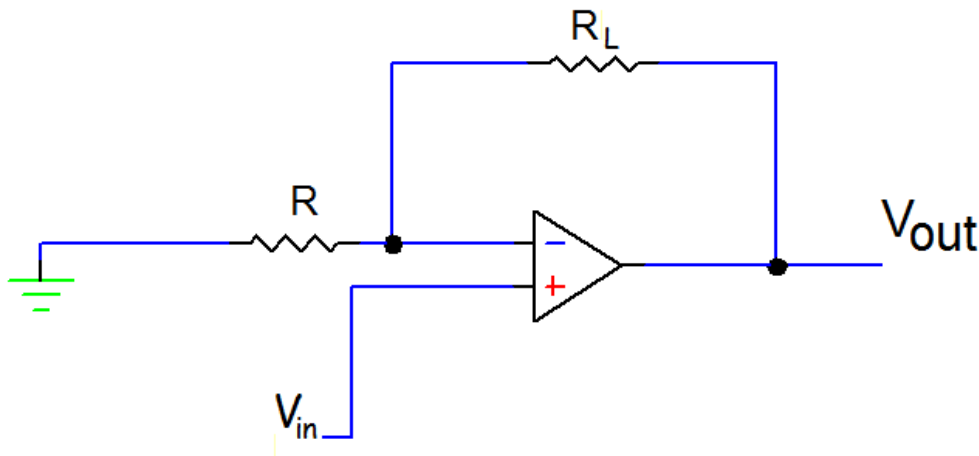
Therefore,

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

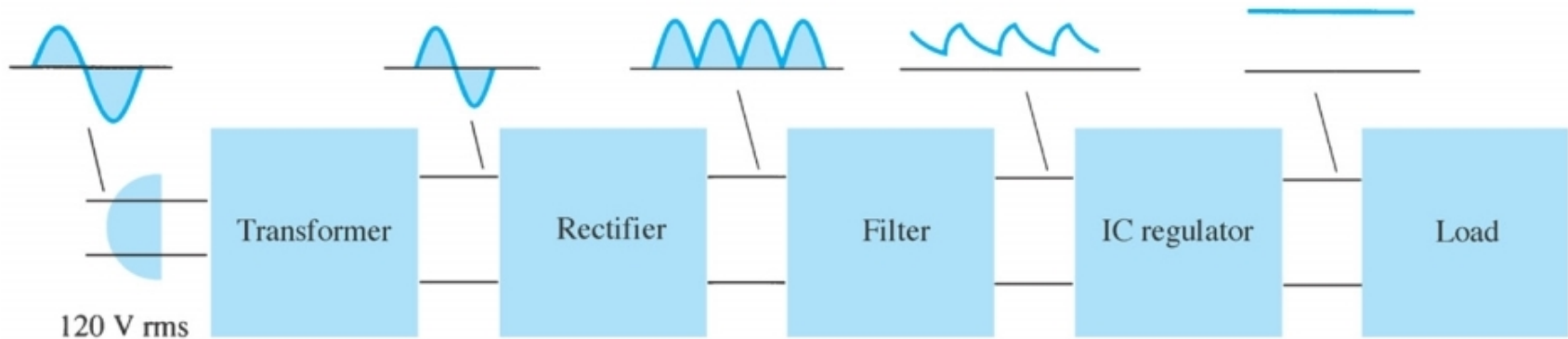
Thus, output is proportional to the input current I_s , and the circuit works as I to V converter.

Voltage-To-Current Converter

- Also called transconductance amplifier
- For a single input, the current in the load resistor is given by $I_L = I = V_{in}/R$. from this equation it is obvious that the output current I_L is independent of load resistance and is proportional to the input voltage. This is because of the virtual ground at the inverting input terminal of the op-amp.



VOLTAGE REGULATORS Introduction



- **Power supply:** a group of circuits that convert the **standard ac voltage** (120 V, 60 Hz) provided by the wall outlet to **constant dc voltage**
- **Transformer** : a device that step up or step down the **ac voltage** provided by the wall outlet to a desired amplitude through the *action of a magnetic field*

VOLTAGE REGULATORS Introduction

- **Rectifier**: a diode circuits that converts the **ac input voltage** to a **pulsating dc voltage**
- The pulsating dc voltage is **only suitable** to be used as a battery charger, but **not good enough** to be used as a dc power supply in a radio, stereo system, computer and so on.
- **Filter**: a circuit used to reduce the fluctuation in the rectified output voltage or ripple. This provides a **steadier** dc voltage.
- **Regulator**: a circuit used to produces a **constant** dc output voltage by reducing the ripple to negligible amount. One part of power supply.

VOLTAGE REGULATORS Introduction

- 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

A voltage regulator is used for two reasons:-

- To regulate or vary the output voltage of the circuit.
- To keep the output voltage constant at the desired value in spite of variations in the supply voltage or in the load current.

VOLTAGE REGULATORS:

Factors Affecting the Load Voltage:

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.

VOLTAGE REGULATORS: Performance Parameters of a Power Supply:

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.

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

VOLTAGE REGULATORS:

Performance Parameters of a Power Supply:

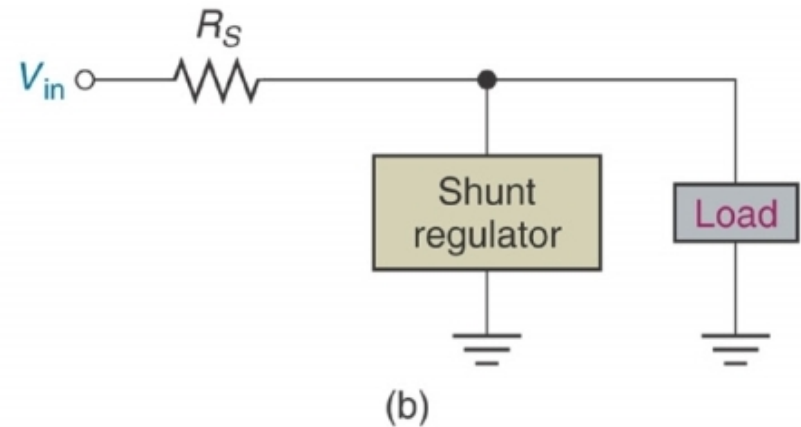
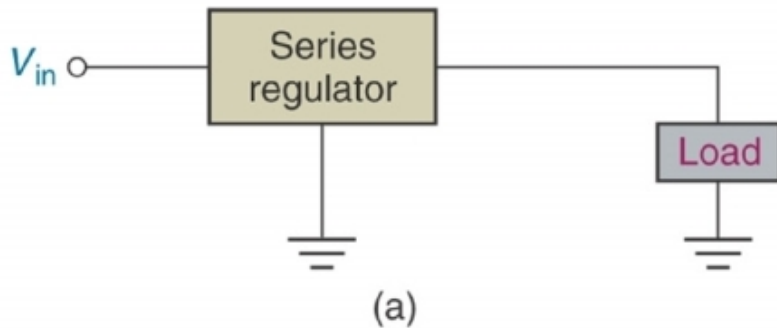
3. *Voltage Stability factor (S_V):* Voltage stability factor shows the dependency of output voltage on the input line voltage. *Voltage stability factor* is defined as the percentage change in the output voltage which occurs per volt change in input voltage, where load current and temperature are assumed to be constant. Smaller the value of this factor, better is the performance of power supply.
4. *Temperature Stability Factor (S_T):* 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_T must be as small as possible, and ideally 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.
$$\text{Ripple rejection factor} = \frac{V_{\text{RIPPLE (OUTPUT)}}}{V_{\text{RIPPLE (INPUT)}}$$

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

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

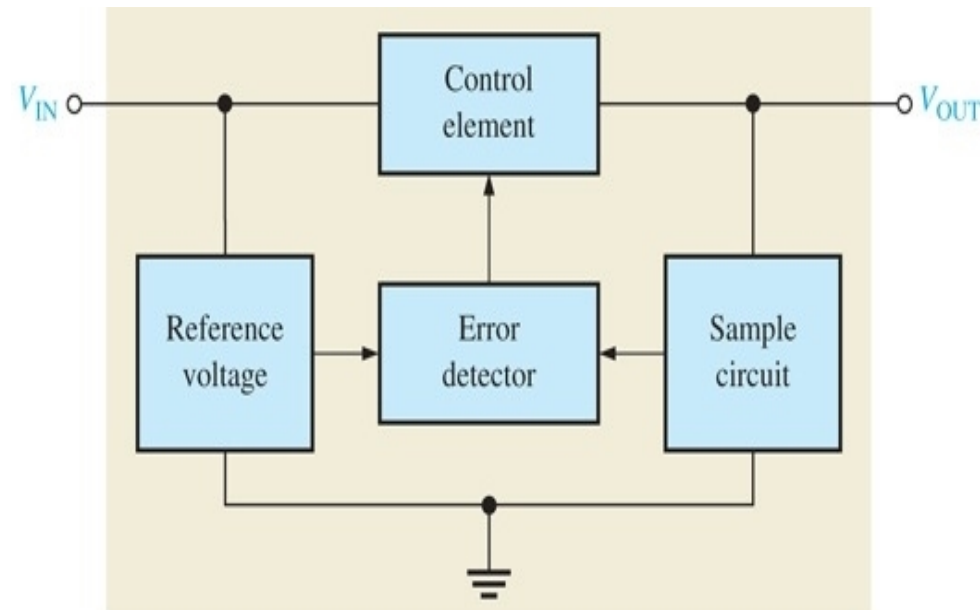
VOLTAGE REGULATORS: Types of Regulator

- The series regulator is connected in **series** with the load
- The shunt regulator is connected in **parallel** with the load.



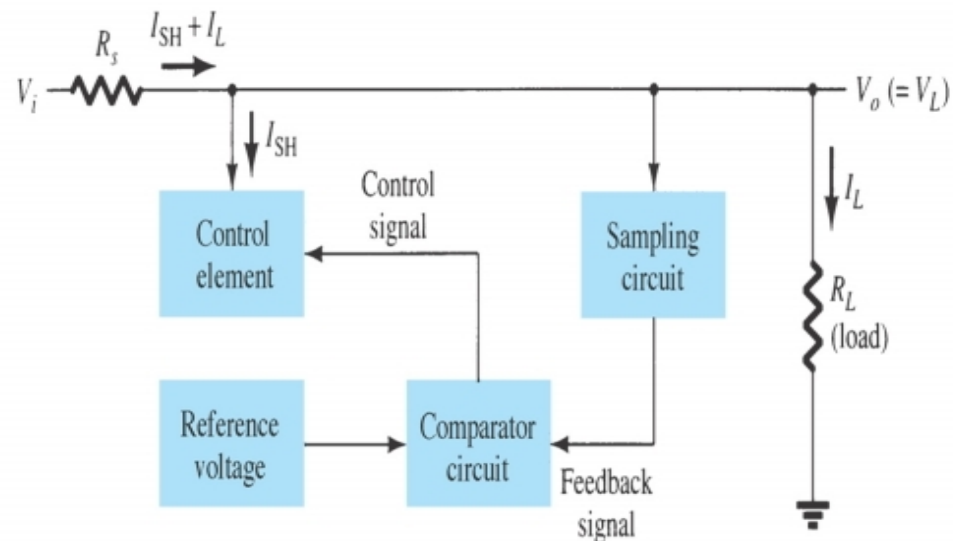
VOLTAGE REGULATORS: Series Regulator Circuit

- **Control element** in series with load between input and output.
- Output **sample circuit** senses a change in output voltage.
- **Error detector** compares sample voltage with reference voltage → causes control element to compensate in order to maintain a constant output voltage



VOLTAGE REGULATORS: Series Regulator Circuit

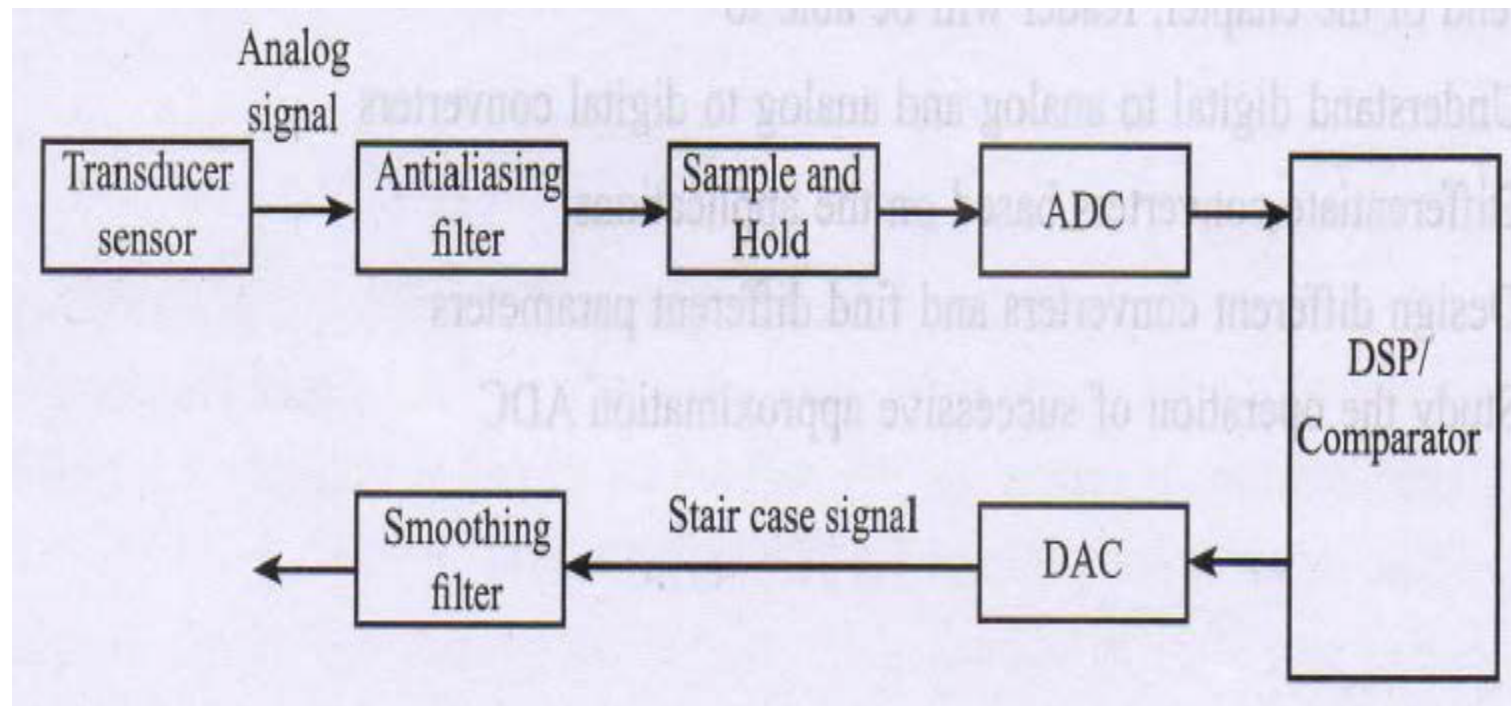
- The unregulated input voltage provides current to the load.
- Some of the current is pulled away by the **control element**.
- If the load voltage tries to change due to a change in the load resistance, the **sampling circuit** provides a feedback signal to a **comparator**.
- The resulting difference voltage then provides a control signal to vary the amount of the current shunted away from the load to maintain the regulated output voltage across the load.



D to A and A to D converter

- WHY A to D?
 - Real world signal appear in analog form
 - Difficult to process, transmit, store in physical form
 - Computer perform operation quickly , efficiently
 - Development in digital technology
- WHY D to A?
 - Computer need to communicate with physical processes , people

Typical A/D and D/A Converter



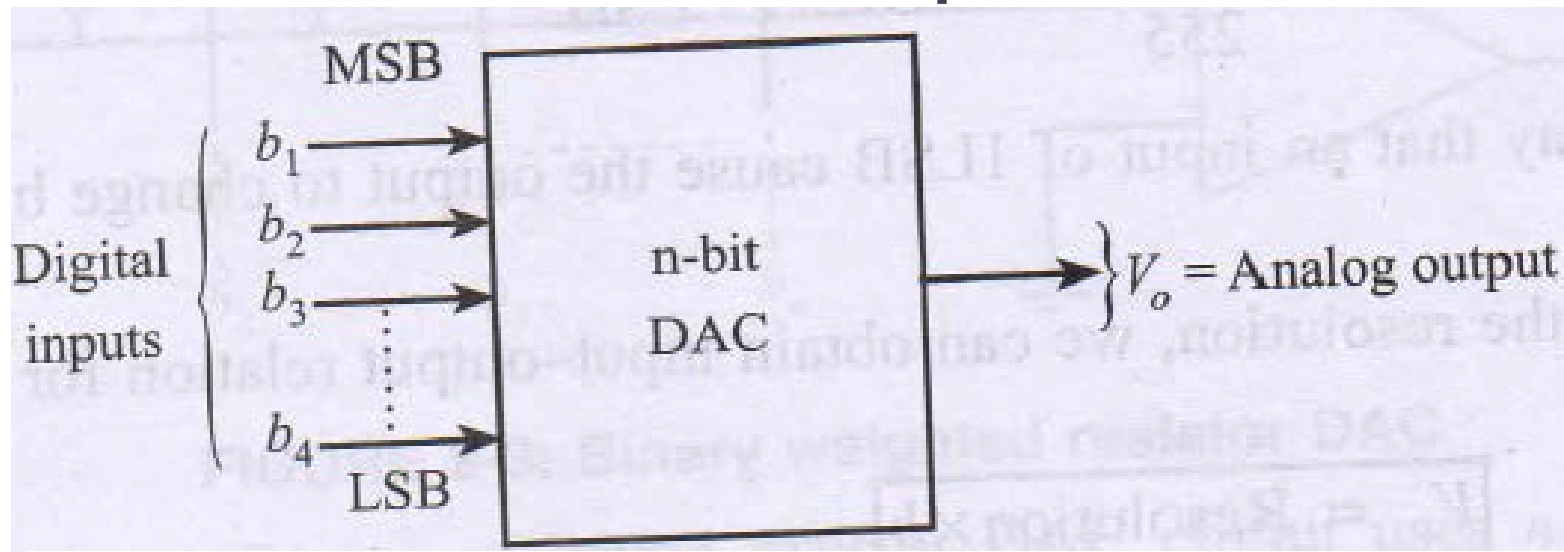
Typical A/D and D/A Converter

1. Obtain analog signal from sensor or transducer
2. Using anti aliasing filter restrict bandwidth of the signal
3. Sample the signal at a frequency rate more than twice maximum frequency of band limited frequency
4. Hold sampled signal using hold circuit during conversion

Typical A/D and D/A Converter

1. Feed the discrete signal to A to D Converter
2. A to D Converter converts the signal to digital signal
3. Digital signal is converted to analog signal using D to A Converter
4. Output of D to A Converter is stair case waveform
5. Stair case waveform is passed through smoothing filter to remove quantized noise

Basic DAC Techniques



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

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:

Ratio of change in output voltage resulting from a change of LSB at the digital input.

$$\text{Resolution} = V_{OFS} / 2^n - 1$$

$$V_o = \text{Resolution} \times b$$

2. Accuracy

Measure of how close the actual output voltage is to the theoretical output value

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

Performance Parameters of DAC

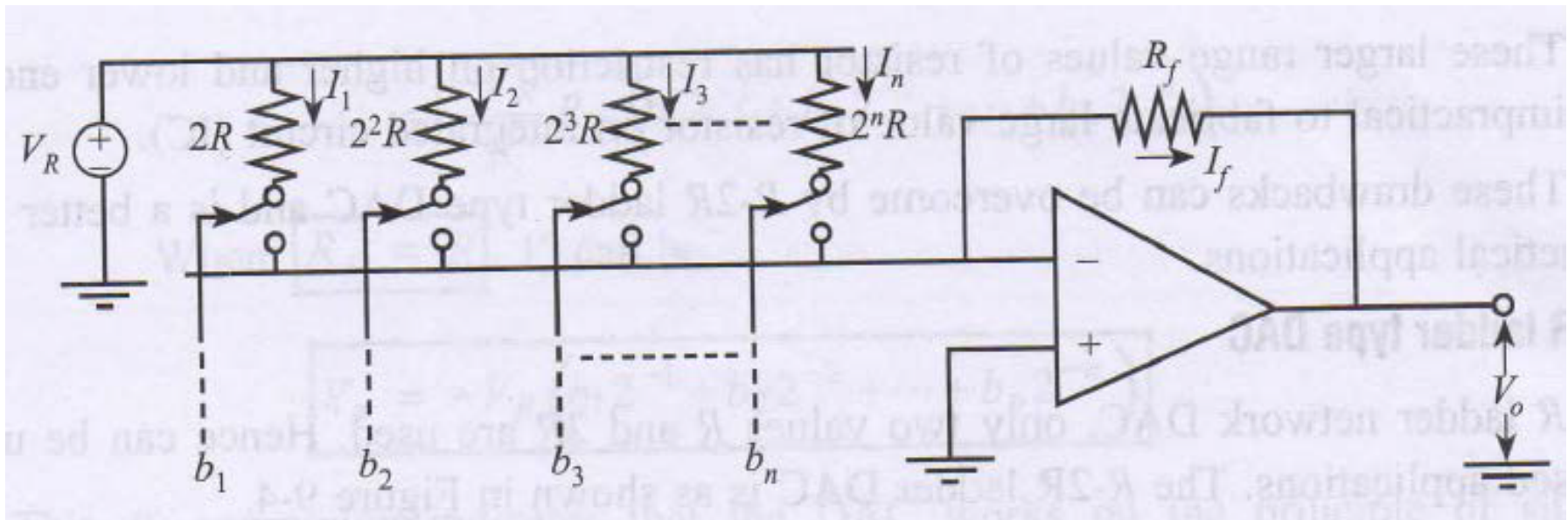
3. Setting Time:

Setting time is the time required for a DAC output to settle within $\pm 1/2$ LSB of final value for a given digital input.

3. 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



- When the switch is ON : $I = V_R/R$
- When the switch is OFF : $I = 0$.

Binary Weighted Resistor DAC

- total current through R_f

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

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

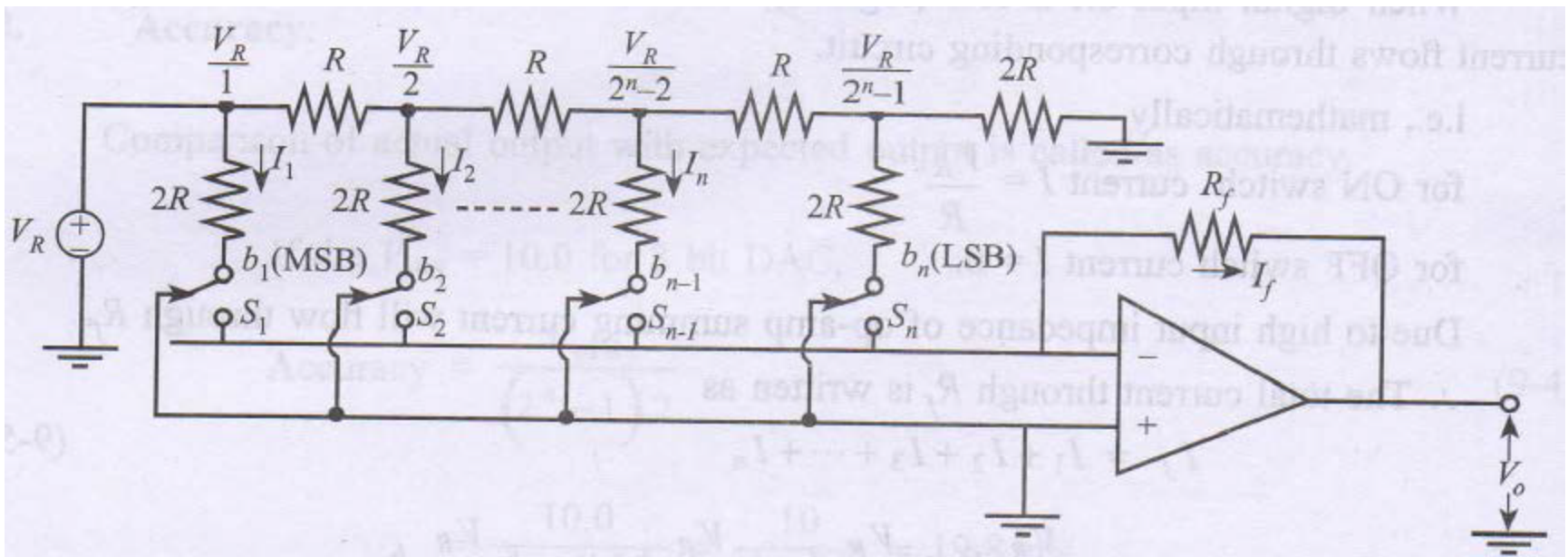
$$i.e., V_0 = \frac{-V_R}{R} R_f [b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \dots + b_n 2^{-n}]$$

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

Drawbacks of DAC

- Large range of resistor values are required, as resistance values increases like 2^1R , 2^2R , 2^3R , . . . , 2^nR .
- Practically it's difficult to fabricate large values of resistors on IC.

R-2R Ladder type DAC



- Due to virtual ground, both the positions of the switches are at ground potential, and currents through the resistances are constant.

R-2R Ladder type DAC

The current flowing through each of $2R$ resistances;

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

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

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

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

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

R-2R Ladder type DAC :

Advantages

- As it requires only two types of resistors, fabrication and accurate value of R -@ R can be designed.
- 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 $V_{oFS} = 15 \text{ V}$.

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

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

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

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

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

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 = $(10000000)_2$

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

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

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

$$D = \text{equivalent of } (10000000)_2 = 128$$

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

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,

$$\begin{aligned} V_o &= 10 \text{ V} \left(0 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8} + 1 \times \frac{1}{16} \right) \\ &= 1.875 \text{ volts.} \end{aligned}$$

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,

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

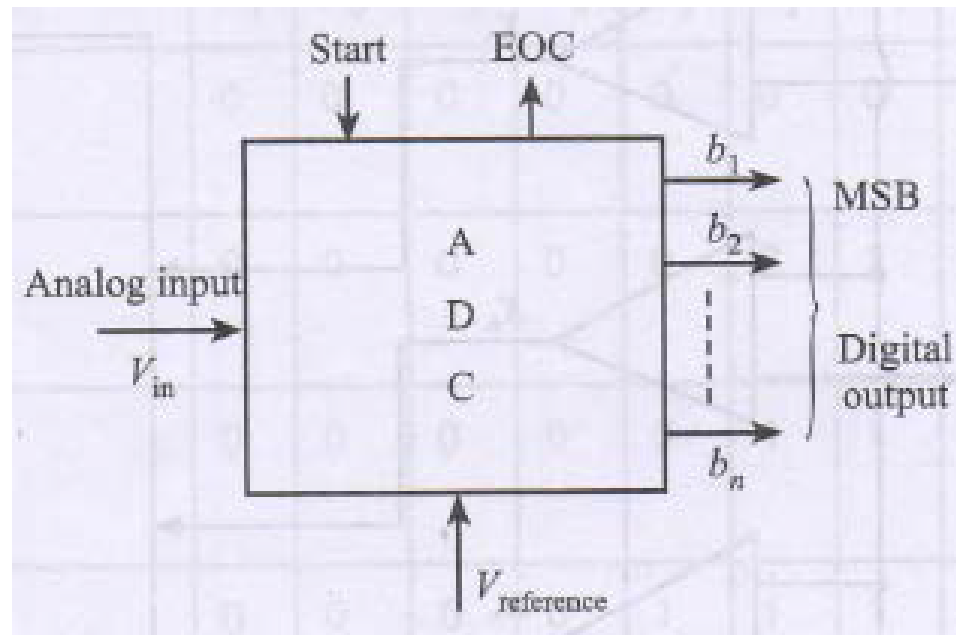
and

$$\begin{aligned} \text{MSB} &= \left(\frac{1}{2} \right) \text{Fullscale} \\ &= \frac{1}{2} \times 10 = 5 \text{ V} \end{aligned}$$

Full scale output = Full scale voltage – 1 LSB

$$= 10 - 625 \text{ mV} = 9.375 \text{ V}$$

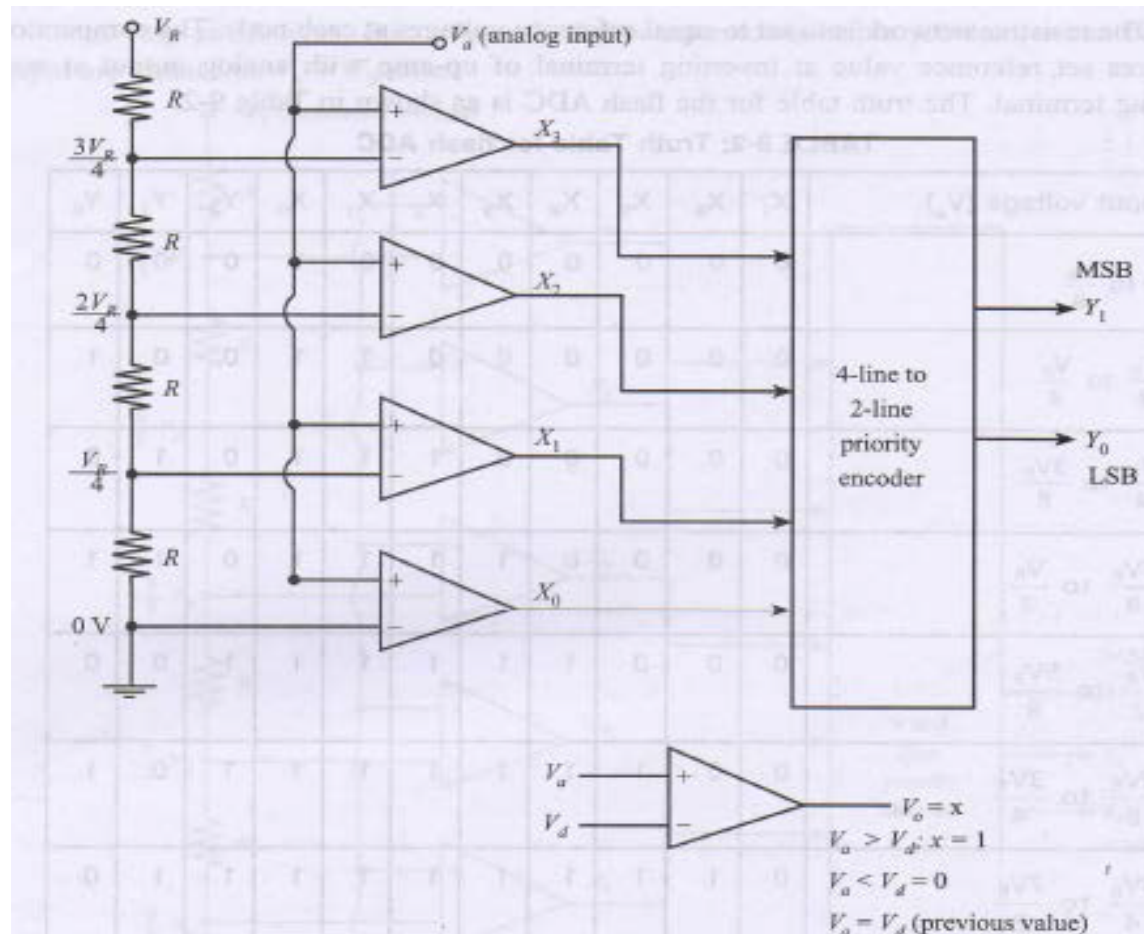
A-D Converters



- Types: Direct type ADCs and Integrated type ADCs

Flash (Comparator/ Parallel) type ADC:

- 2-bit Flash ADC

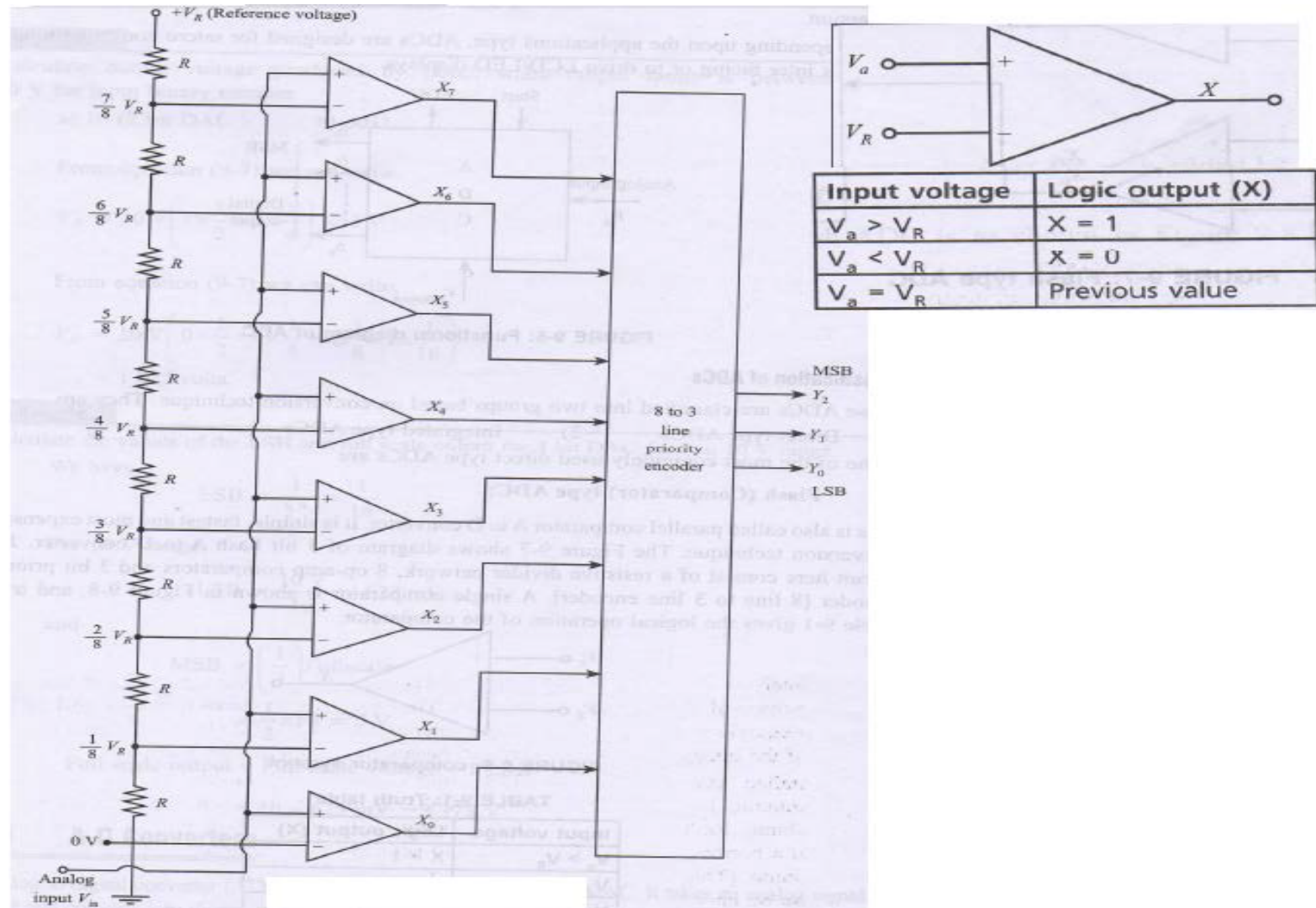


Flash (Comparator/ Parallel) type ADC:

- 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

3-bit Flash ADC



Flash (Comparator/ Parallel) type ADC:

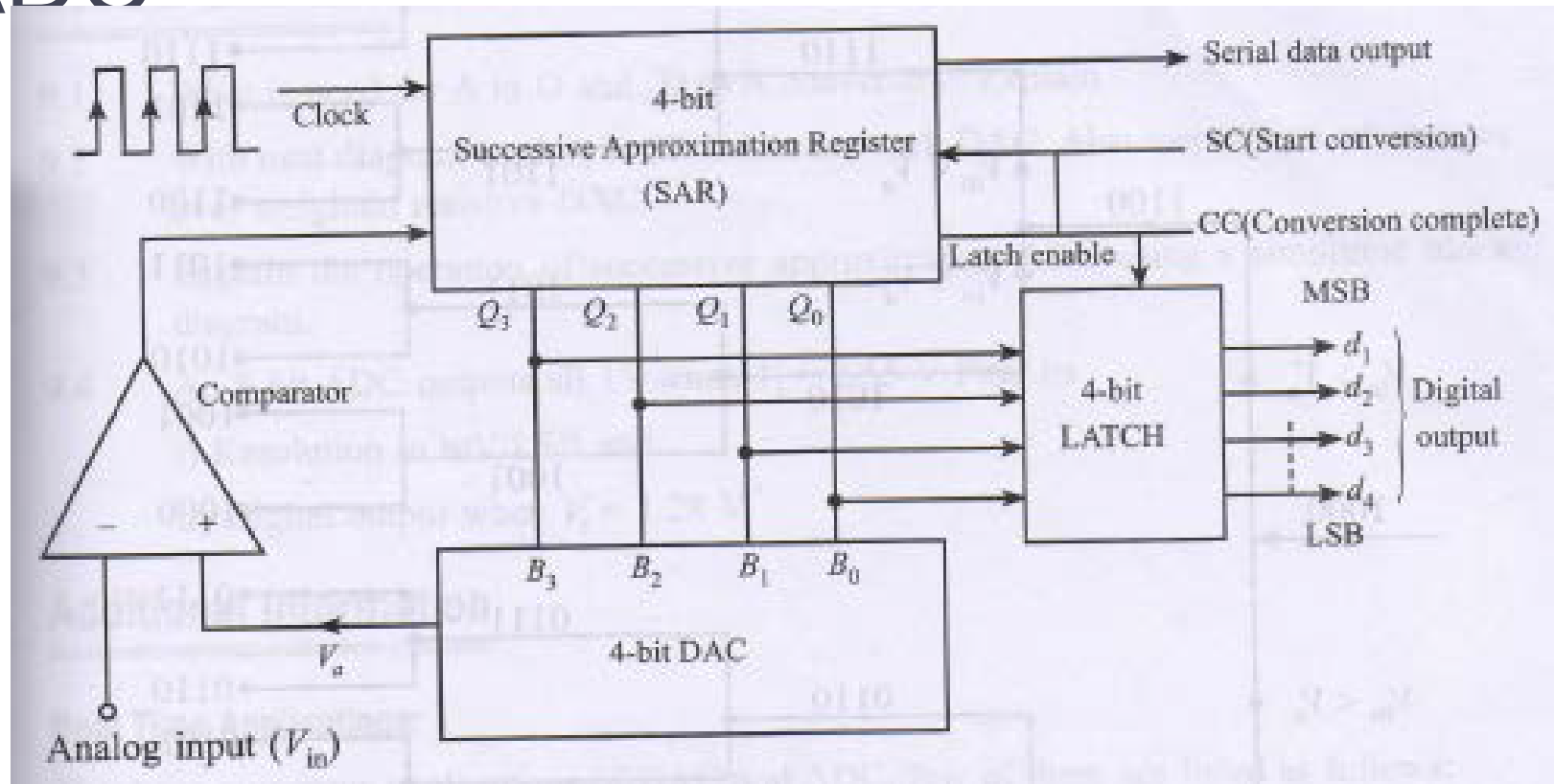
- ***Advantages:***

- High speed

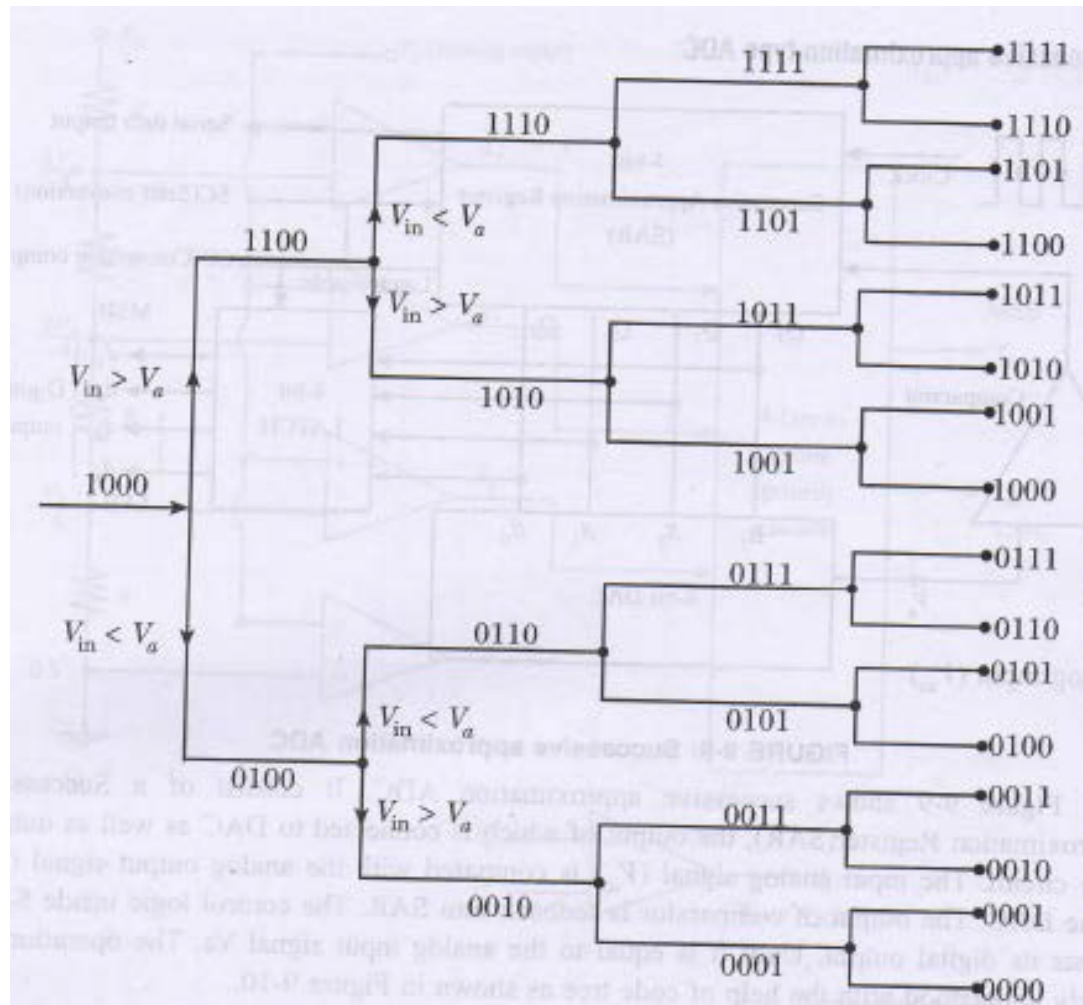
- ***Disadvantages:***

- Number of comparators required is almost double for each added bit
- Eg.: For 2-bit ADC; No. of Comparators = 4 (2²)
For 3-bit ADC; No. of Comparators = 8 (2³)

Successive Approximation type ADC



Successive Approximation type ADC



Successive Approximation type ADC

- The conversion time for *n-bit successive approximation ADC* is $(n + 2)$ clock periods.
- **Advantages:**
 - Considerably good speed
 - Good resolution.

Module-2

**The Basic Gates &
Combinational Logic Circuits**

11
0

Text Books Referred

- ◇ Donald P Leach, Albert Paul Malvino & Goutam Saha: Digital Principles and Applications, 7th Edition, Tata McGraw Hill, 2015

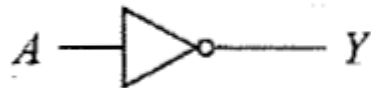
Basic Gates-1

- ❖ **A *logic gate* is a digital circuit with 1 or more input voltages but only 1 output voltage.**
- ❖ Logic gates are the fundamental building blocks of digital systems.
- ❖ By connecting the different gates in different ways, we can build circuits that perform arithmetic and other functions associated with the human brain.
- ❖ **Because the circuits simulate mental processes, gates are often called *logic circuits*. NOT, OR & AND gates are the basic types of gates.**
- ❖ **The inter-connection of gates to perform a variety of logical operations is called *logic design*.**
- ❖ The operation of a logic gate can be easily understood with the help of "truth table".
- ❖ **A *truth table* lists all possible combinations of inputs and the corresponding outputs.**

Basic Gates-2

NOT GATE (INVERTER)

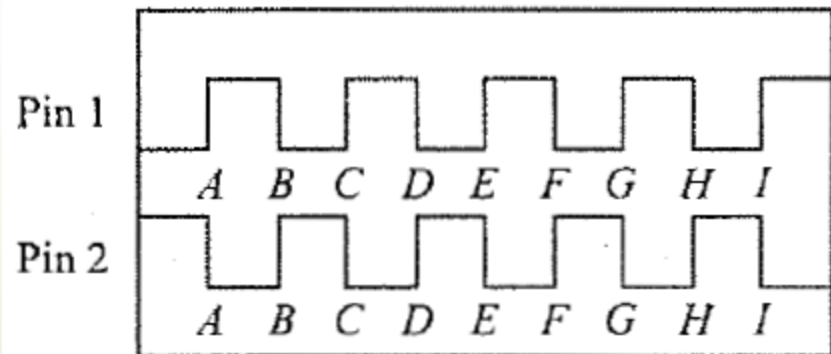
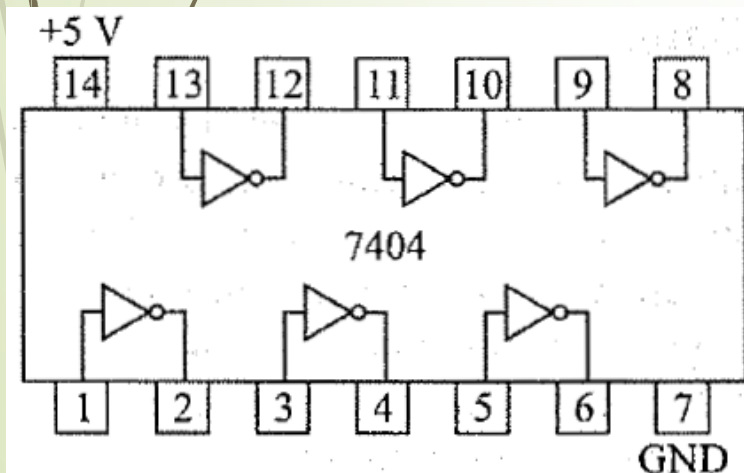
- It is a gate with only 1 input and a complemented output.



(a)

<i>A</i>	<i>Y</i>	<i>A</i>	<i>Y</i>
<i>L</i>	<i>H</i>	0	1
<i>H</i>	<i>L</i>	1	0

(b)

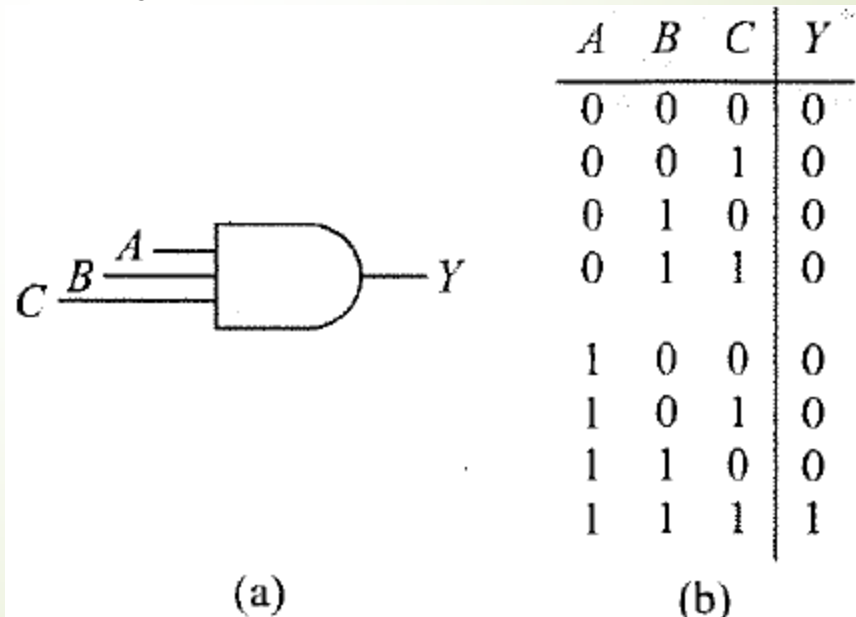
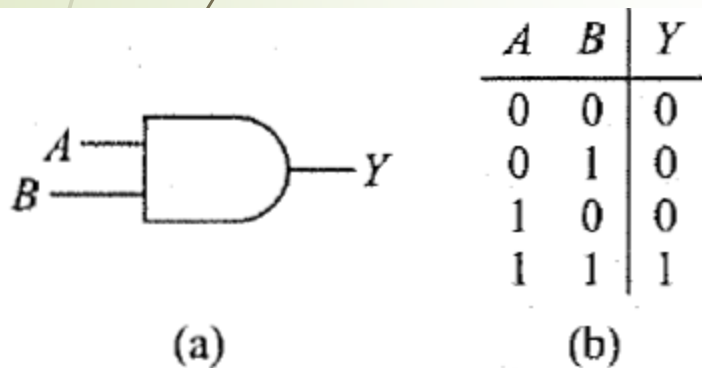


Basic Gates-3

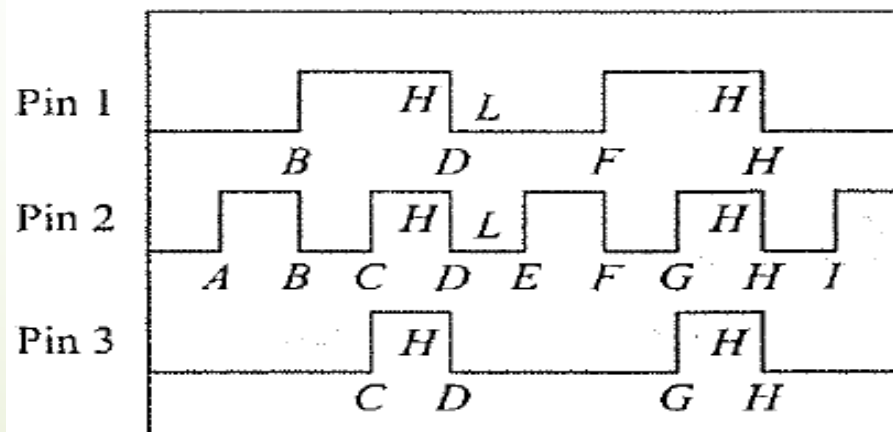
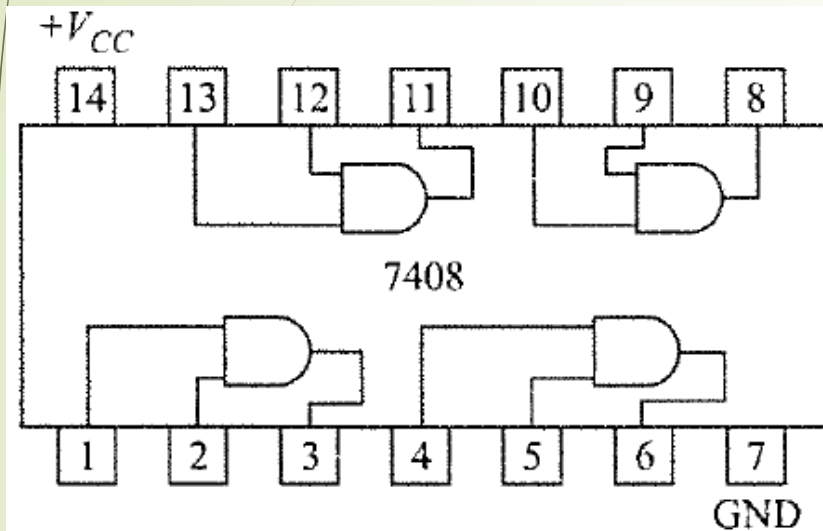
◇ AND GATE

◇ This is a gate with 2 or more inputs.

◇ The output is HIGH only when all inputs are HIGH.



Basic Gates-4

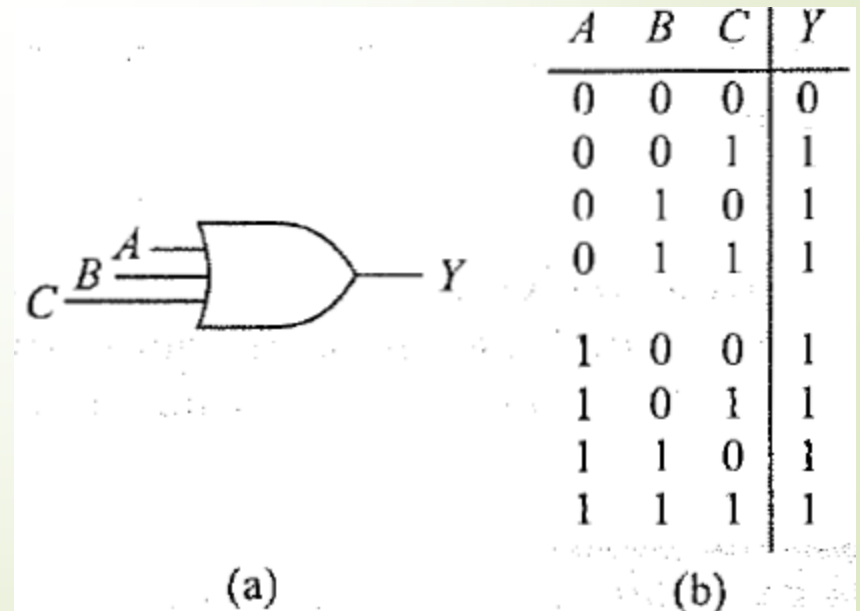
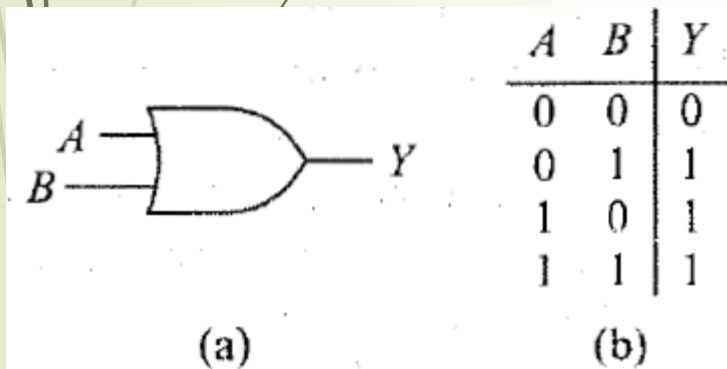


Timing diagram

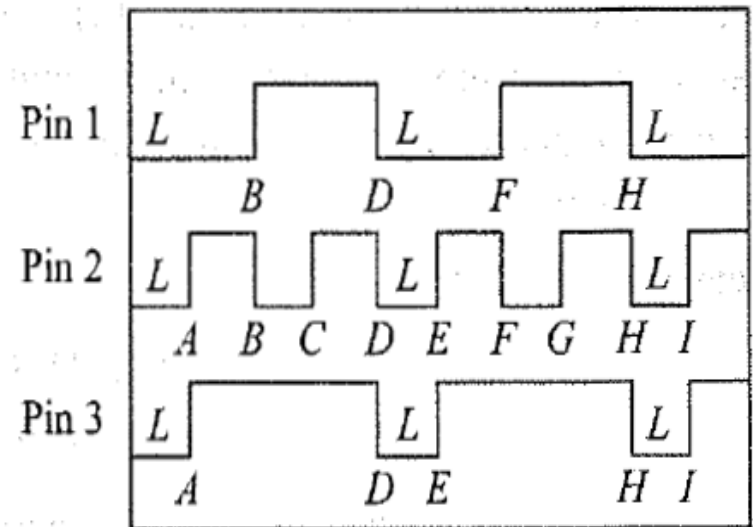
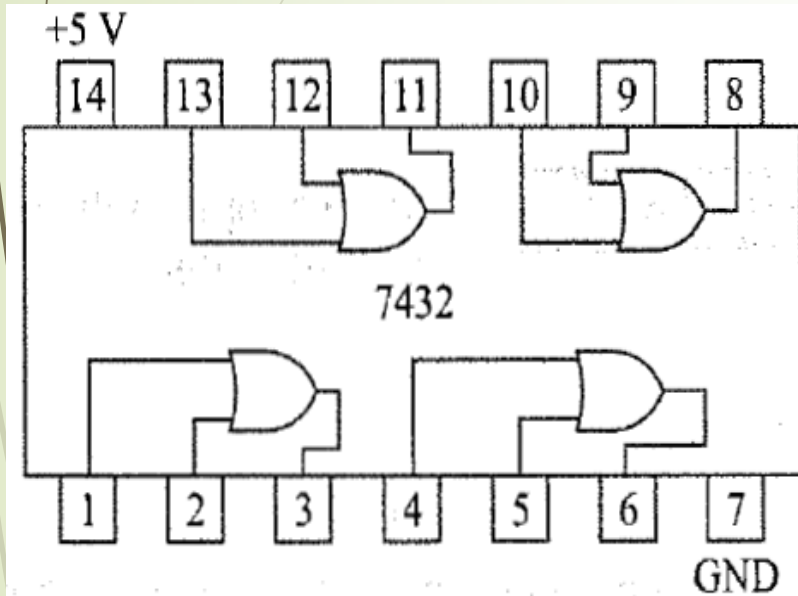
Basic Gates-5

◇ OR GATE

- ◇ This is a gate with 2 or more inputs.
- ◇ The output is HIGH when any input is HIGH.



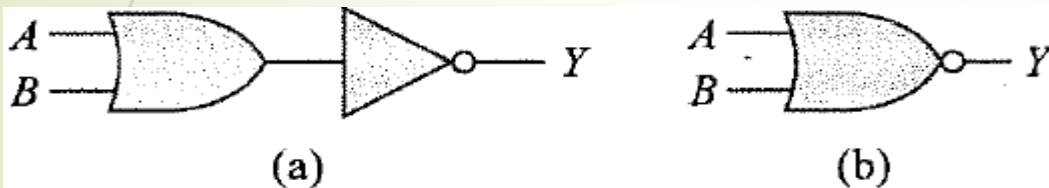
Basic Gates-6



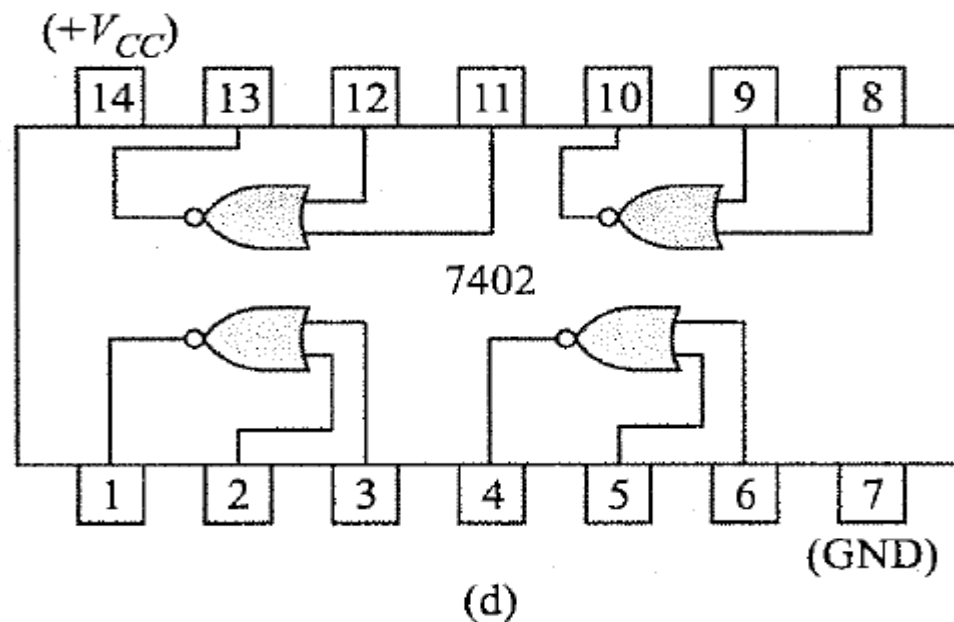
Basic Gates-9

◇ NOR GATE

◇ This represents an OR gate followed by an inverter.



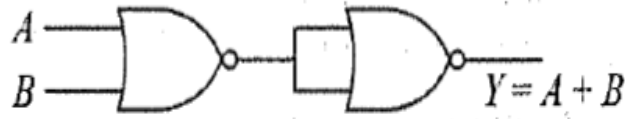
<i>A</i>	<i>B</i>	<i>Y</i>
0	0	1
0	1	0
1	0	0
1	1	0



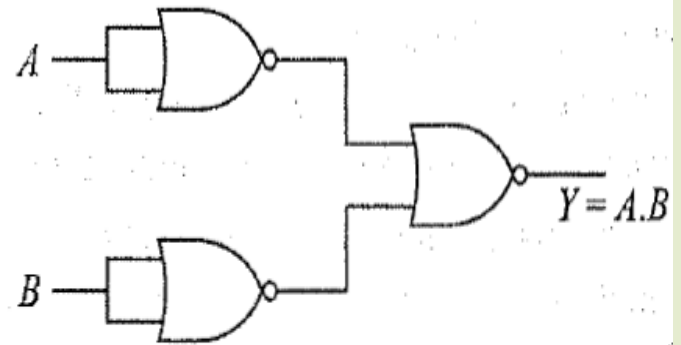
Basic Gates-10



(a)



(b)



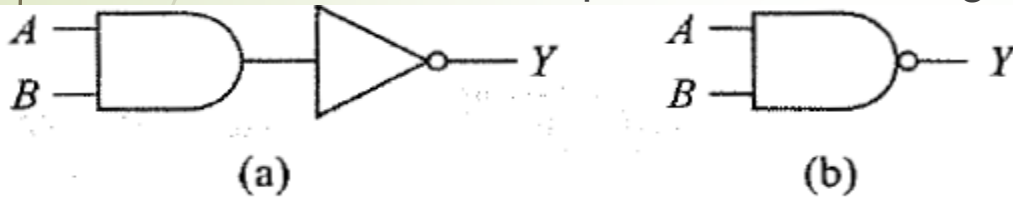
(c)

Universality of NOR gate (a) NOT from NOR, (b) OR from NOR,
(c) AND from NOR

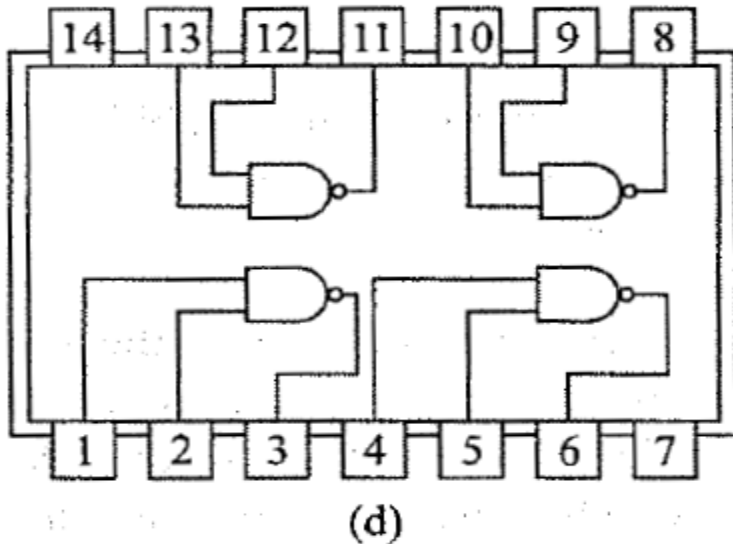
Basic Gates-11

◇ NAND GATE

◇ This represents an AND gate followed by an inverter.

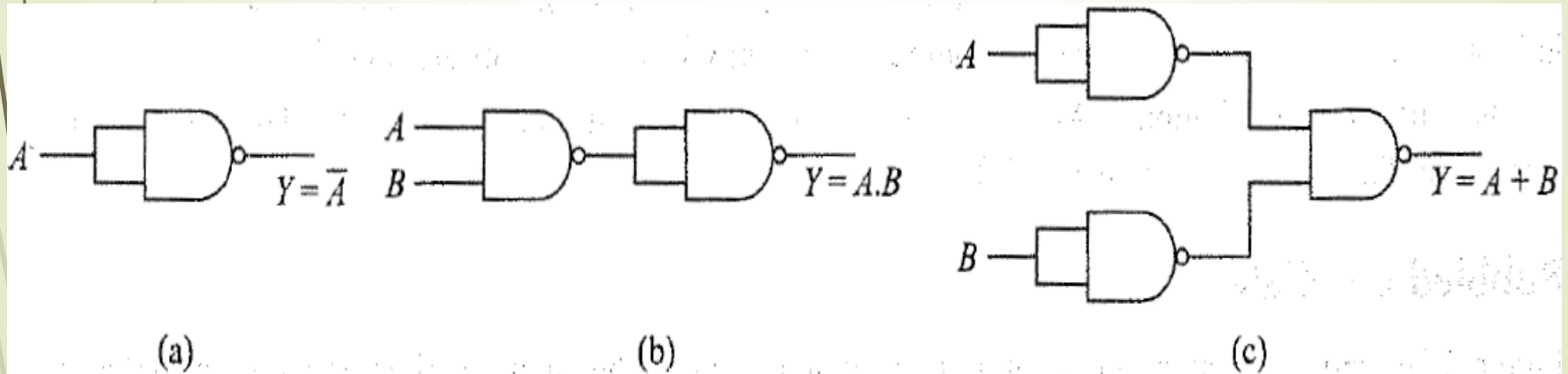


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



Basic Gates-12

◇ Universality of NAND Gate



Universality of NAND gate: (a) NOT from NAND, (b) AND from NAND, (c) OR from NAND

Positive and Negative Logic-1

- ❖ We use a binary 0 for low voltage and a binary 1 for high voltage. This is called *positive logic*.
- ❖ Another code known as *negative logic* where binary 0 stands for high voltage and binary 1 for low voltage.

Positive and Negative Logic-2

Positive and Negative Gates

- An OR gate in a positive logic system becomes an AND gate in a negative logic system.



<i>A</i>	<i>B</i>	<i>Y</i>
0	0	0
0	1	1
1	0	1
1	1	1

Positive logic system

<i>A</i>	<i>B</i>	<i>Y</i>
1	1	1
1	0	0
0	1	0
0	0	0

Negative logic system

Positive and Negative Logic-3

- ◇ In a similar way, we can show the truth table of other gates with positive or negative logic. By analysing the inputs and outputs in terms of 0s and 1s, you find these equivalences between the positive and negative logic:

Positive OR	↔ negative AND
Positive AND	↔ negative OR
Positive NOR	↔ negative NAND
Positive NAND	↔ negative NOR

Gate	Definition
Positive OR/negative AND	Output is high if any input is high.
Positive AND/negative OR	Output is high when all inputs are high.
Positive NOR/negative NAND	Output is low if any input is high.
Positive NAND/negative NOR	Output is low when all inputs are high.

Positive and Negative Logic-4

- ◆ **Assertion-level logic**
- ◆ **Logic circuits with bubbles on all pins with active-low signals and omit bubbles on all pins with active-high signals. This use of bubbles with active-low signals is called *assertion-level logic*.**
- ◆ If a low input signal turns on a chip, you show a bubble on that input. If a low output is a sign of chip action, you draw a bubble on that output. Once you get used to assertion-level logic, you may prefer drawing logic circuits this way.

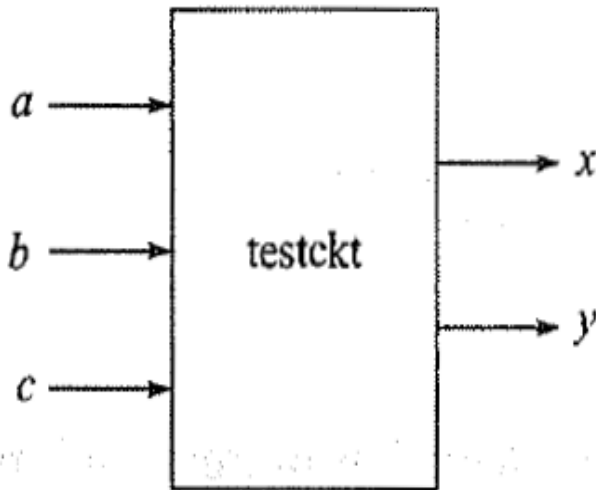
INTRODUCTION TO HDL-1

- ❖ HDL is a language that describes the hardware of digital systems in a textual form. It resembles a programming language, but is specifically oriented to describing hardware structures, dataflow and behaviors.
- ❖ The main difference with the traditional programming languages is HDL's representation of extensive parallel operations whereas traditional ones represents mostly serial operations. HDL can be used to represent logic diagrams, Boolean expressions, and other more complex digital circuits
- ❖ There are two standard HDL's that are supported by IEEE.
 - ❖ **VHDL** (*Very-High-Speed Integrated Circuits Hardware Description Language*) - Sometimes referred to as VHSIC HDL, this was developed from an initiative by US. Dept. of Defense.
 - ❖ **Verilog HDL** – developed by Cadence Data systems and later transferred to a consortium called *Open Verilog International* (OVI).

INTRODUCTION TO HDL-2

- ❖ Verilog: Verilog HDL has a syntax that describes precisely the legal constructs that can be used in the language.
- ❖ It uses about 100 keywords pre-defined, lowercase, identifiers that define the language constructs.
- ❖ Example of keywords: *module*, *endmodule*, *input*, *output* *wire*, *and*, *or*, *not* , etc.,
- ❖ Any text between two slashes (//) and the end of line is interpreted as a comment.
- ❖ Blank spaces are ignored and names are case sensitive.
- ❖ A *module* is the building block in Verilog. It is declared by the keyword *module* and is always terminated by the keyword *endmodule*. Each statement is terminated with a semicolon, but there is no semi-colon after *endmodule*.

INTRODUCTION TO HDL-3



```
module testckt(x,y,a,b,c); //module name with port list
```

```
input a,b,c; //defines input ports
```

```
output x,y; //defines output ports
```

```
//module body begins next describing logic relation
```

```
.
```

```
.
```

```
//module body ends
```

```
endmodule
```

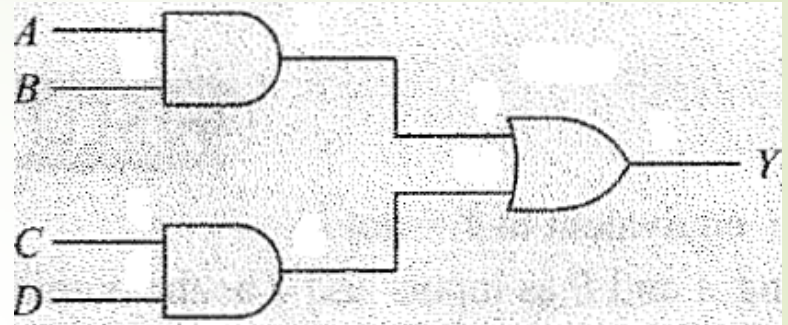
INTRODUCTION TO HDL-4

```
module or_gate (A, B, Y);  
input A, B;    // defines two input port  
output Y;      // defines one output port  
or g1(Y,A,B); /*Gate  declaration with predefined  
                keyword or representing logic          OR, g1 is optional  
user defined gate      indentifier*/  
endmodule
```

- ◇ The syntax for the basic gate
- ◇ E.g. For OR gate **or** (output, input 1, input 2, input 3, input 4)
- ◇ For NOT gate, **not** (output, input)

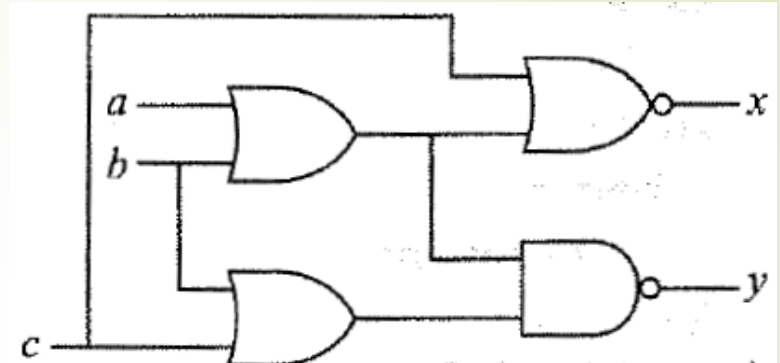
INTRODUCTION TO HDL-5

```
module eg1(A,B,C,D,Y);  
input A,B,C,D;  
output Y;  
wire and_op1, and_op2;  
and g1(and_op1,A,B); //g1 represents upper AND gate  
and g2(and_op2,C,D); //g2 represents lower AND gate  
or g3(Y,and_op1,and_op2); // g3 represents the OR gate  
endmodule
```



INTRODUCTION TO HDL-6

```
module eg2(a,b,c,x,y);  
  input a,b,c;  
  output x,y;  
  wire or_op1, or_op2;  
  or g1(or_op1,a,b);  
  or g2(or_op2,b,c);  
  nor g13(x,c,or_op1);  
  nand g4(y,or_op1,or_op2);  
endmodule
```



SUM-Of-PRODUCTS (SOP)

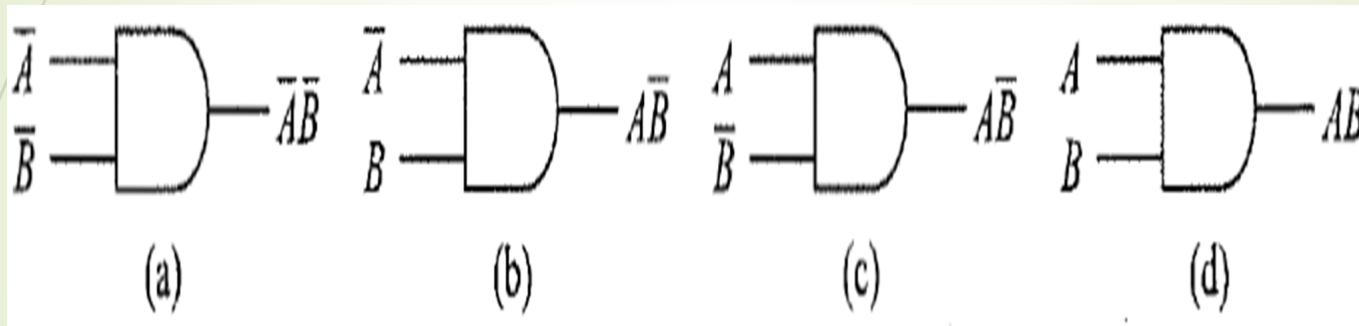
Method-1

- ❖ Possible ways to AND two or more input signals that are in complement and un-complement form.
- ❖ A SOP expression is two or more AND functions ORed together.

SUM-Of-PRODUCTS (SOP)

Method-2

◇ ANDing two variables and their complements



A	B	<i>Fundamental Product</i>
0	0	$\bar{A}\bar{B}$
0	1	$\bar{A}B$
1	0	$A\bar{B}$
1	1	AB

SUM-Of-PRODUCTS (SOP)

Method-3

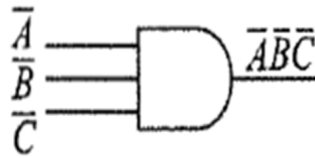
- The fundamental products are also called minterms.
- Products $\overline{A}\overline{B}$, $\overline{A}B$, $A\overline{B}$, AB are represented by m_0 , m_1 , m_2 and m_3 respectively. The suffix i of m_i comes from decimal equivalent of binary values that makes corresponding product term high.

SUM-Of-PRODUCTS (SOP)

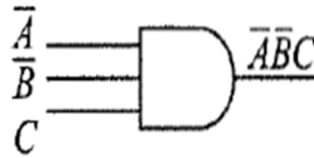
Method-4

◇ Example:

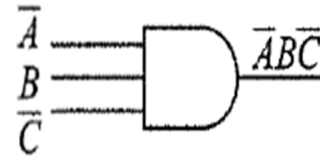
$$\overline{A}\overline{B}\overline{C}, \overline{A}\overline{B}C, \overline{A}B\overline{C}, \overline{A}BC, A\overline{B}\overline{C}, A\overline{B}C, AB\overline{C}, ABC$$



(a)



(b)



(c)

<i>A</i>	<i>B</i>	<i>C</i>	<i>Fundamental Products</i>
0	0	0	$\overline{A}\overline{B}\overline{C}$
0	0	1	$\overline{A}\overline{B}C$
0	1	0	$\overline{A}B\overline{C}$
0	1	1	$\overline{A}BC$
1	0	0	$A\overline{B}\overline{C}$
1	0	1	$A\overline{B}C$
1	1	0	$AB\overline{C}$
1	1	1	ABC

SUM-Of-PRODUCTS (SOP)

Method-5

- ◇ The above three variable minterms can alternatively be represented by m_0 , m_1 , m_2 , m_3 , m_4 , m_5 , m_6 , and m_7 respectively. Note that, for n variable problem there can be 2^n number of minterms.
- ◇ **The fundamental products by listing each one next to the input condition that results in a high output.**
- ◇ For instance, when $A = 1$, $B = 0$ and $C = 0$, the fundamental product results in an output of

$$Y = A\bar{B}\bar{C} = 1 \cdot \bar{0} \cdot \bar{0} = 1$$

Sum-of-Products Equation-1

- Sum-of-Products Equation
- The Sum-of-products solution, for given a truth table shown below.

<i>A</i>	<i>B</i>	<i>C</i>	<i>Y</i>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

- Write down the fundamental product for each output 1 in the truth table. For example, the first output 1 appears for an input of $A = 0$, $B = 1$, and $C = 1$. The corresponding fundamental product is $\bar{A}BC$.

Sum-of-Products Equation-2

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	$1 \rightarrow \bar{A}BC$
1	0	0	0
1	0	1	$1 \rightarrow A\bar{B}C$
1	1	0	$1 \rightarrow ABC$
1	1	1	$1 \rightarrow ABC$

- ◇ To get the sum-of-products equation, all you have to do is OR the fundamental products

$$Y = \bar{A}BC + A\bar{B}C + ABC + ABC$$

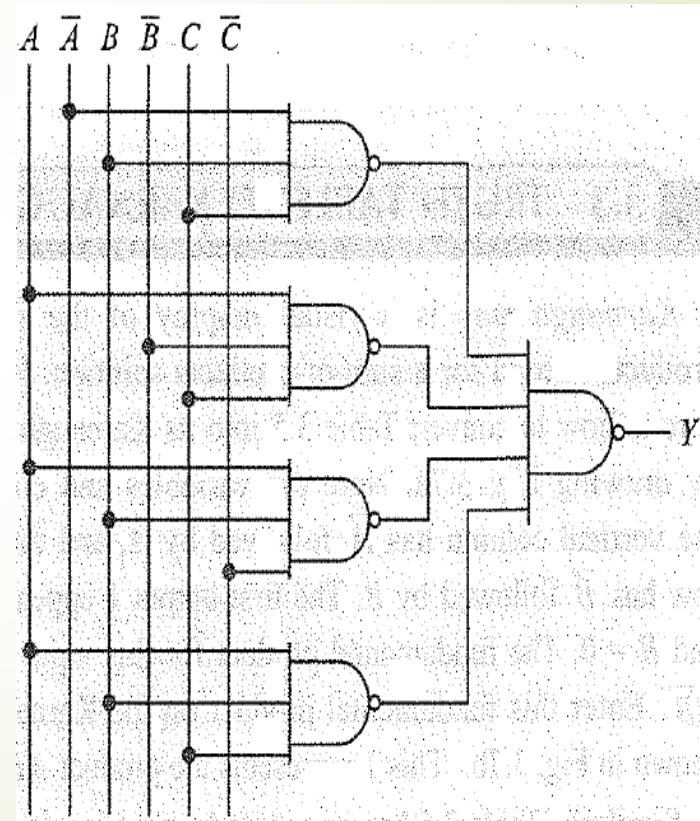
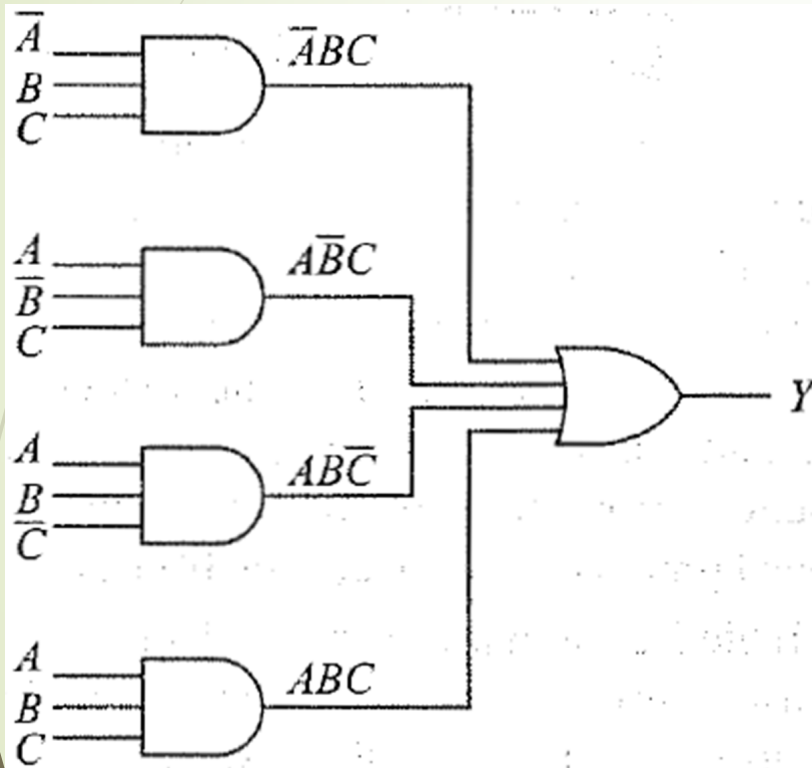
- ◇ Alternate representation

$$Y = F(A, B, C) = \sum m(3, 5, 6, 7)$$

Sum-of-Products Equation-3

- ◇ where ' Σ :' symbolizes summation or logical OR operation that is performed on corresponding minterm's and $Y = F(A, B, C)$ means Y is a function of three Boolean variables A, B and C. This kind of representation of a truth table is also known as canonical sum form.

Logic Circuit



Logic Circuit

Lab Experiment

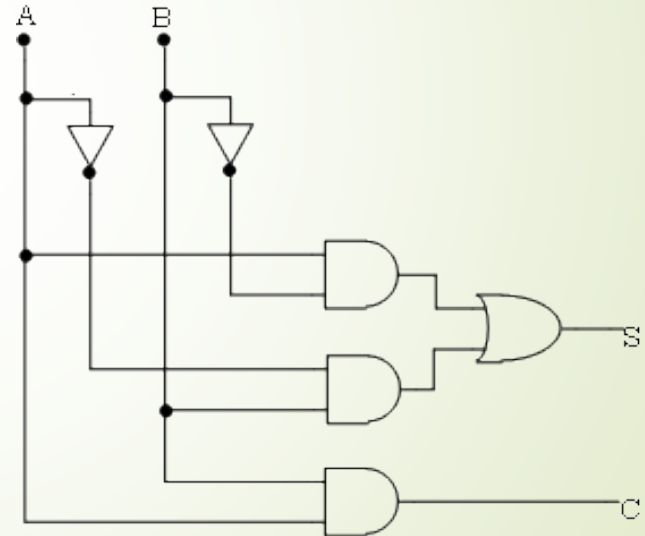
- Design and implement Half adder, Full Adder, Half Subtractor, Full Subtractor using basic gates.
- Half Adder:
- Truth Table for Half Adder

Input		Output	
A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic Expression

$$S = \bar{A}.B + A.\bar{B}$$

$$C = A.B$$



Logic Circuit

Laws and Rules of Boolean Algebra

Commutative Law	$A + B = B + A$	$A \cdot B = B \cdot A$
Associative Law	$A + (B + C) = (A + B) + C$	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Distributive Law	$A \cdot (B + C) = A \cdot C + A \cdot B$	$A + B \cdot C = (A + B) \cdot (A + C)$
Null Elements	$A + 1 = 1$	$A \cdot 0 = 0$
Identity	$A + 0 = A$	$A \cdot 1 = A$
Idempotence	$A + A = A$	$A \cdot A = A$
Complement	$A + \bar{A} = 1$	$A \cdot \bar{A} = 0$
Involution	$\bar{\bar{A}} = A$	
Absorption (Covering)	$A + A \cdot B = A$	$A \cdot (A + B) = A$
Simplification	$A + \bar{A} \cdot B = A + B$	$A \cdot (\bar{A} + B) = A \cdot B$
DeMorgan's Rule	$\overline{A + B} = \bar{A} \cdot \bar{B}$	$\overline{A \cdot B} = \bar{A} + \bar{B}$
Logic Adjacency (Combining)	$A \cdot B + A \cdot \bar{B} = A$	$(A + B) \cdot (A + \bar{B}) = A$
Consensus	$A \cdot B + B \cdot C + \bar{A} \cdot C = A \cdot B + \bar{A} \cdot C$	$(A + B) \cdot (B + C) \cdot (\bar{A} + C) = (A + B) \cdot (\bar{A} + C)$

TRUTH TABLE TO KARNAUGH MAP-1

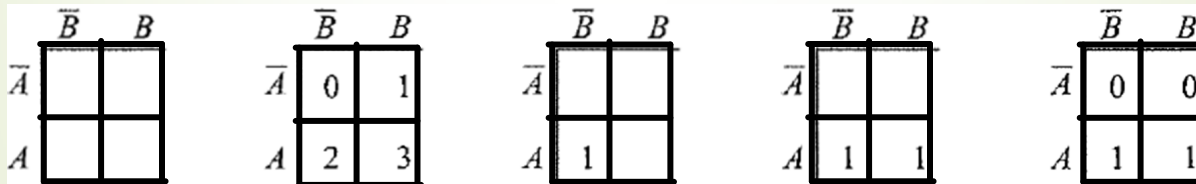
- ◇ A Karnaugh map (K-Map) is a visual display of the fundamental products needed for a sum-of-products solution.
- ◇ Also simplification of logical expression.

TRUTH TABLE TO KARNAUGH MAP-2

Two-Variable Maps

Example

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



Example: $Y = F(A, B) = \sum m(1, 2, 3)$

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

Truth Table.

Karnaugh map for Y . The map has values: top-left 0, top-right 1, bottom-left 1, bottom-right 1. A blue arrow points to the top-right cell (1).

TRUTH TABLE TO KARNAUGH MAP-3

Three-Variable Maps

Example: $Y = F(A, B, C) = \Sigma m(2, 6, 7)$

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

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

	\bar{C}	C
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	2	3
AB	6	7
$A\bar{B}$	4	5

	\bar{C}	C
$\bar{A}\bar{B}$		
$\bar{A}B$	1	
AB	1	1
$A\bar{B}$		

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

TRUTH TABLE TO KARNAUGH MAP-4

$\overline{A}\overline{B}$, $\overline{A}B$, AB , $A\overline{B}$?

	\overline{C}	C		\overline{C}	C
$\overline{A}\overline{B}$			$\overline{A}\overline{B}$	0	1
$\overline{A}B$			$\overline{A}B$	2	3
AB			AB	6	7
$A\overline{B}$			$A\overline{B}$	4	5
(a)			(b)		

	\overline{C}	C
$\overline{A}\overline{B}$	0	1
$\overline{A}B$	2	3
AB	6	7
$A\overline{B}$	4	5
(b)		

TRUTH TABLE TO KARNAUGH MAP-4

$\overline{A}\overline{B}$, $\overline{A}B$, AB , $A\overline{B}$?

	\overline{C}	C		\overline{C}	C
$\overline{A}\overline{B}$			$\overline{A}\overline{B}$	0	1
$\overline{A}B$			$\overline{A}B$	2	3
AB			AB	6	7
$A\overline{B}$			$A\overline{B}$	4	5

(a)

	\overline{C}	C
$\overline{A}\overline{B}$	0	1
$\overline{A}B$	2	3
AB	6	7
$A\overline{B}$	4	5

(b)

TRUTH TABLE TO KARNAUGH MAP-5

❖ Four-Variable Maps

❖ Example: $Y = F(A,B,C,D) = \sum m(1,6,7,14)$

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>Y</i>
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

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

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

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

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

TRUTH TABLE TO KARNAUGH MAP-6

◇ Five-Variable Maps

◇ Example:

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

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

TRUTH TABLE TO KARNAUGH MAP-6

Five-Variable Maps

Example:

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	3	7	5
$\bar{A}B$	9	11	15	13
AB	25	27	31	29
$A\bar{B}$	17	19	23	21

E

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	2	6	4
$\bar{A}B$	8	10	14	12
AB	24	26	30	28
$A\bar{B}$	16	18	22	20

\bar{E}

TRUTH TABLE TO KARNAUGH MAP-6

◇ Five-Variable Maps

◇ Example:

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

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

TRUTH TABLE TO KARNAUGH MAP-6

Five-Variable Maps

Example:

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

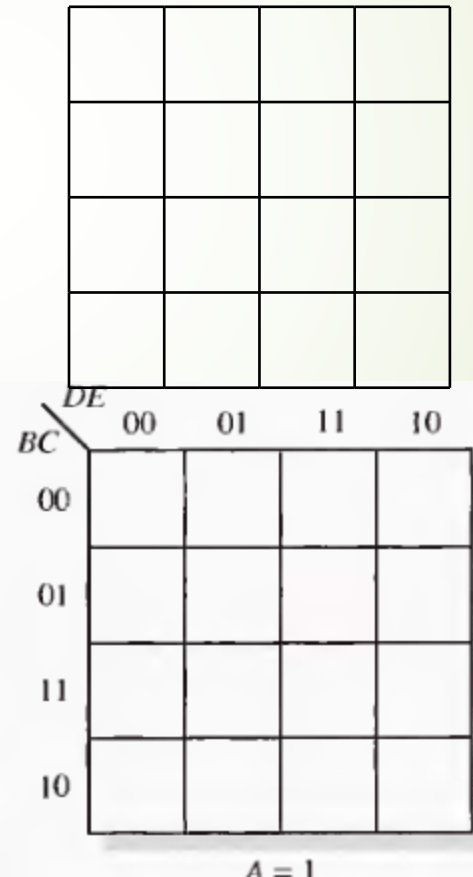
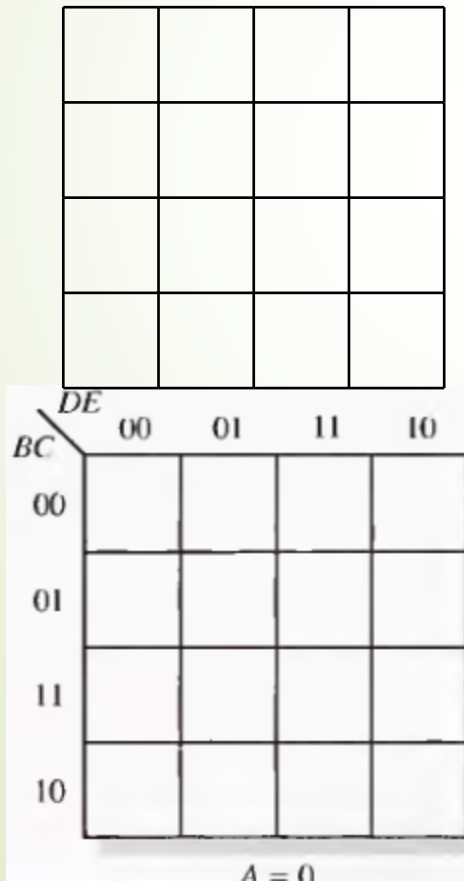
E

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

\bar{E}

TRUTH TABLE TO KARNAUGH MAP-7

◇ Five-Variable Maps



PAIRS, QUADS, AND OCTETS-1

❖ Pairs

- ❖ The map contains a pair of 1s that are horizontally or vertically adjacent.

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

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

❖ $Y = ABC$

PAIRS, QUADS, AND OCTETS-2

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

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

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

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

PAIRS, QUADS, AND OCTETS-3

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

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

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

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

PAIRS, QUADS, AND OCTETS-4

- ◇ The Octet
- ◇ The *octet* is a group of eight 1 s.
- ◇ Octet eliminates *three variables and their complements*.

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

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

KARNAUGH

SIMPLIFICATIONS-1

- ❖ The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together adjacent cells containing *ones*.
- ❖ After drawing a Karnaugh map,
 - ❖ Encircle the octets first,
 - ❖ The quads second, and
 - ❖ The pairs last.

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

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

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

KARNAUGH

SIMPLIFICATIONS-2

Groups may not include any cell containing a zero.

❖

A B	0	1
0	0	
1	1	

X

❖

A B	0	1
0	0	
1	1	1

✓

WRONG X

RIGHT ✓

❖ Groups may be horizontal or vertical, but not diagonal.

❖

A B	0	1
0	0	1
1	1	0

X

WRONG X

❖

A B	0	1
0	0	1
1	1	1

✓

RIGHT ✓

KARNAUGH

SIMPLIFICATIONS-3

- ❖ Groups must contain 1, 2, 4, 8, or in general 2^n cells.
- ❖ That is if $n = 1$, a group will contain two 1's since $2^1 = 2$.
- ❖ If $n = 2$, a group will contain four 1's since $2^2 = 4$.

A \ B	0	1
	0	1
0	1	1
1	0	0

Group of 2

RIGHT ✓

A \ B	0	1
	0	1
0	1	1
1	1	1

Group of 4

RIGHT ✓

A \ B \ C	00	01	11	10
	0	1	1	1
0	0	1	1	1
1	0	0	0	0

Group of 3

WRONG ✗

A \ B \ C	00	01	11	10
	0	1	1	1
0	1	1	1	1
1	0	0	0	1

Group of 5

WRONG ✗

KARNAUGH

SIMPLIFICATIONS-4

- Each group should be as large as possible.

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

RIGHT ✓

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

WRONG ✗

(Note that no Boolean laws broken, but not sufficiently minimal)

- Each cell containing a *one* must be in at least one group.

AB \ C	00	01	11	10
0	0	0	1	1
1	0	0	0	1

Group I

Group II

1 present in at least one group.

KARNAUGH

SIMPLIFICATIONS-5

- Groups may overlap.

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups overlapping. ✓

RIGHT ✓

AB \ C	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups not overlapping. ✗

WRONG ✗

- Groups may wrap (Rolling the Map) around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.

AB \ C	00	01	11	10
0	1	1	1	1
1	1		1	1

Top cell

Leftmost cell

Bottom cell

Rightmost cell

KARNAUGH SIMPLIFICATIONS-6

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

(a)

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

(b)

$$Y = B\overline{C}\overline{D} + BCD$$

$$Y = B\overline{D}$$

KARNAUGH SIMPLIFICATIONS-7

- ◇ Always overlap groups if possible
- ◇ Use the 1s more than once to get the largest groups you

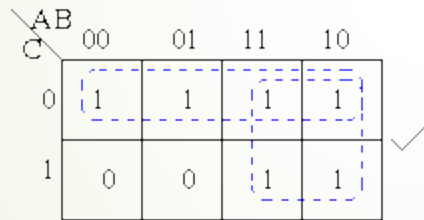
	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	0
$\overline{A}B$	0	1	0	0
AB	1	1	1	1
$A\overline{B}$	1	1	1	1

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

KARNAUGH SIMPLIFICATIONS-8

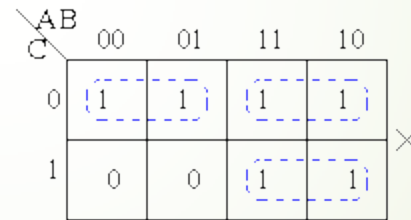
- ❖ There should be as few groups as possible, as long as this does not contradict any of the previous rules.

AB	00	01	11	10
C	0	1	1	1
1	0	0	1	1



RIGHT ✓

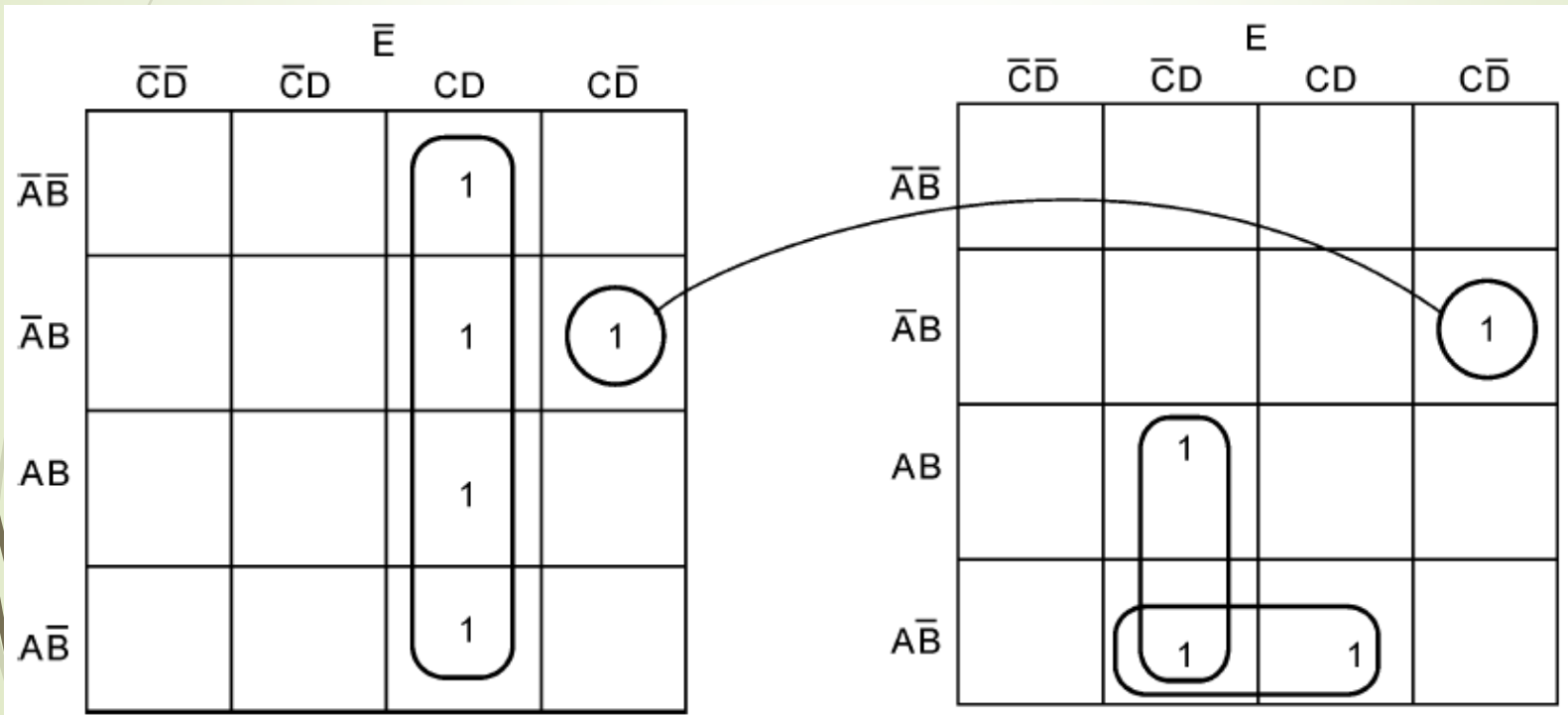
AB	00	01	11	10
C	0	1	1	1
1	0	0	1	1



WRONG ✗

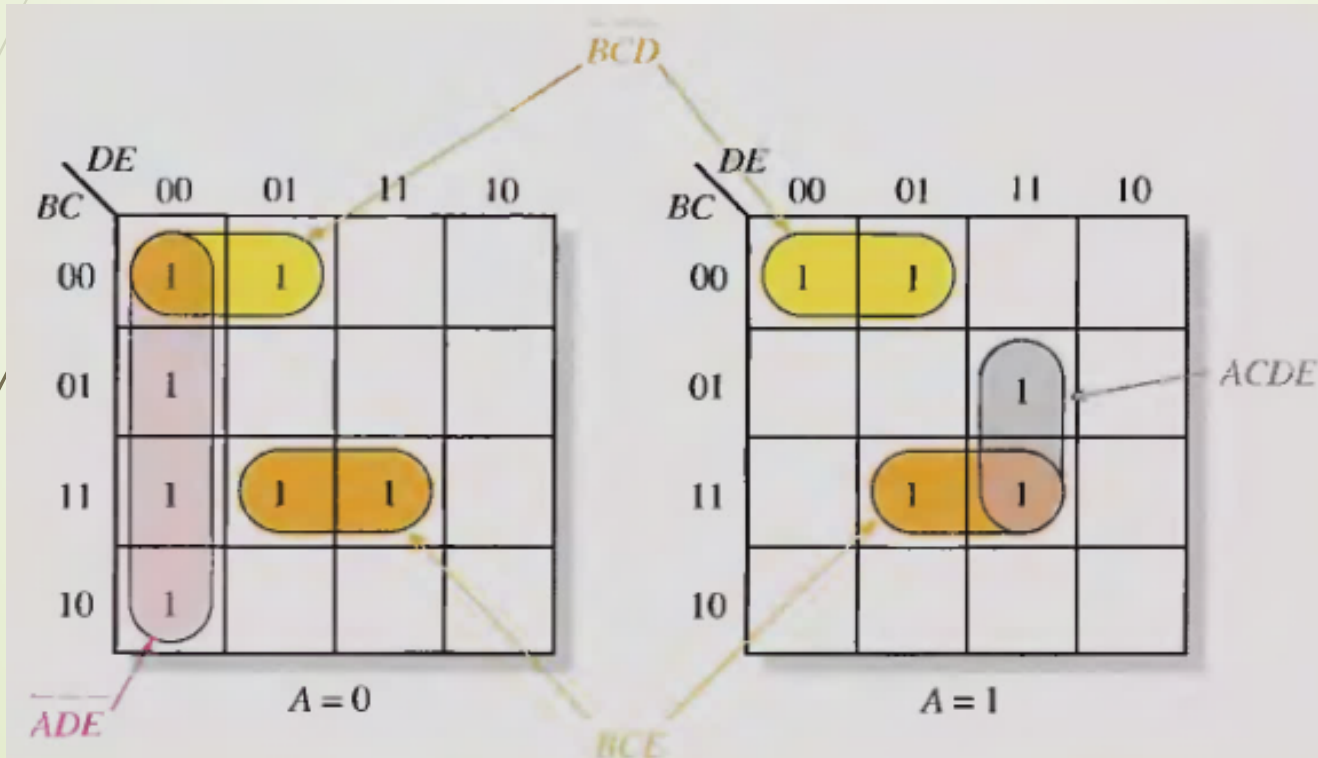
KARNAUGH SIMPLIFICATIONS-9

◇ Five Variable



$$Y = C.D.\bar{E} + \bar{A}.B.C.\bar{D} + A.\bar{C}.D.E + A.\bar{B}.D.E$$

KARNAUGH SIMPLIFICATIONS-10



$$X = \overline{A}\overline{D}\overline{E} + \overline{B}\overline{C}\overline{D} + BCE + ACDE$$

KARNAUGH SIMPLIFICATIONS-11

1. No zeros allowed (Only in SOP).
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. If possible roll and overlap to get the largest groups you can find.
9. Fewest number of groups possible.

K- map grouping Examples-1

Rolling and overlapping

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

$$Y = \bar{C} + BCD$$

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

$$Y = \bar{C} + B\bar{D}$$

K- map grouping Examples-2

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

$$Y = \bar{C} + \bar{A}C\bar{D} + A\bar{B}C\bar{D}$$

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

$$Y = \bar{C} + \bar{A}\bar{D} + A\bar{B}\bar{D}$$

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

$$Y = \bar{C} + \bar{A}\bar{D} + \bar{B}\bar{D}$$

Eliminating Redundant Groups-1

- ❖ A groups of 1s (or 0s for POS) whose all members are overlapped by other groups is called redundant group. We don't consider this group while writing the simplified equations from the K-map.

- In the above K-map the group which is represented by the oval is a redundant group and hence while writing the equations we ignore it or we don't make this kind of group and the K-map representation becomes as given next:
- The equation we get is

$$F = yz'w' + x'z'w \text{ (ignoring the redundant group)}$$
- If we consider this group then equation would be $F = yz'w' + x'z'w + x'yz'$
 And this is a not the simplified expression and hence **WRONG**.

zw \ xy	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	1	0	0	0
10	0	0	0	0

Eliminating Redundant Groups-1

- ❖ A groups of 1s (or 0s for POS) whose all members are overlapped by other groups is called redundant group. We don't consider this group while writing the simplified equations from the K-map.

- In the above K-map the group which is represented by the oval is a redundant group and hence while writing the equations we ignore it or we don't make this kind of group and the K-map representation becomes as given next:
- The equation we get is

$$F = yz'w' + x'z'w \text{ (ignoring the redundant group)}$$
- If we consider this group then equation would be $F = yz'w' + x'z'w + x'yz'$
 And this is a not the simplified expression and hence **WRONG**.

zw \ xy	00	01	11	10
00	0	1	0	0
01	1	1	0	0
11	1	0	0	0
10	0	0	0	0

Eliminating Redundant Groups-2

◇ A group whose 1s are already used by other groups.

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

Eliminating Redundant Groups-2

◇ A group whose 1s are already used by other groups.

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

Eliminating Redundant Groups-2

- ❖ A group whose 1s are already used by other groups.

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

- ❖ All the 1s of the quad are used by the pairs. Because of this, the quad is redundant and can be eliminated to get

Eliminating Redundant Groups-2

- ◇ A group whose 1s are already used by other groups.

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

summary of the Karnaugh-map method for simplifying Boolean equations

1. Enter a 1 on the Karnaugh map for each fundamental product that produces a 1 output in the truth table. Enter 0s elsewhere.
2. Encircle the octets, quads, and pairs. Remember to roll and overlap to get the largest groups possible.
3. If any isolated 1s remain, encircle each.
4. Eliminate any redundant group.
5. Write the Boolean equation by ORing the products corresponding to the encircled groups.

K- map grouping Examples

- ◇ For problem on K-Map refer text book and VTU question papers.

Entered Variable Map-1

- ❖ One of the input variable is placed inside Karnaugh map
- ❖ This reduces the Karnaugh map size by one degree.
- ❖ i.e. a three variable problem that requires $2^3 = 8$ locations in Karnaugh map will require $2^{(3-1)} = 4$ locations in entered variable map.
- ❖ This technique is particularly useful for mapping problems with more than four input variables.

Entered Variable Map-2

◇ Example

◇ Example: $Y = F(A, B, C) = \sum m(2,6,7)$

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

Truth Table

A	B	Y
0	0	0
0	0	
0	1	No Image
0	1	
1	0	0
1	0	
1	1	1
1	1	

EVM Table

	\overline{B}	B
\overline{A}	0	\overline{C}
A	0	1

K-MAP for
EVM Table

Entered Variable Map-3

- ▶ Let's choose C as *map entered variable* and see how output Y varies with C for different combinations of other two variables A and B .
- ▶ For $AB = 00$ we find $Y = 0$ and is not dependent on C .
- ▶ For $AB = 01$ we find Y is complement of C thus we can write $Y = \bar{C}$.
- ▶ Similarly, for $AB = 10$, $Y = 0$ and for $AB = 11$, $Y = 1$.

Entered Variable Map-4

- ◇ If choose A as map entered variable write the corresponding entered variable map.

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

\bar{B}	\bar{C}	C
0	0	0
1	1	A

- ◇ If choose B as map entered variable write the corresponding entered variable map.

A	C	Y
0	0	0
0	1	No Image
1	0	C
1	1	1

\bar{B}	B
0	\bar{C}
1	C

Simplification of Entered Variable Map-1

A	B	Y
0	0	0
0	0	
0	1	No Image
0	1	
1	0	0
1	0	
1	1	1
1	1	

This is similar to Karnaugh map

A	B	Y
0	0	0
0	1	No Image
1	0	0
1	1	1

$$Y = B\bar{C} + AB$$

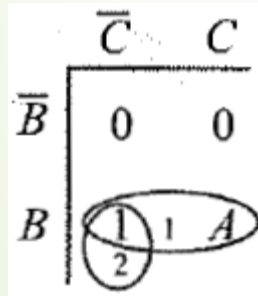
	\bar{B}	B
\bar{A}	0	\bar{C}
A	0	1

	\bar{B}	B
\bar{A}	0	\bar{C} 1
A	0	2 1

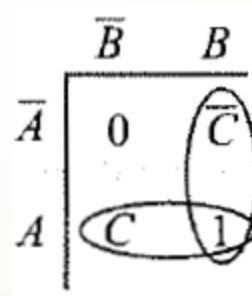
Note that \bar{C} is grouped with 1 to get a larger group as 1 can be written as $1 = 1 + \bar{C}$.

Simplification of Entered Variable Map-2

- ◇ The product term representing each group is obtained by including map entered variable in the group as an additional ANDed term.



$$Y = B\bar{C} + AB$$



$$Y = AC + B\bar{C}$$

Example-1

- What is the simplified Boolean equation for the following logic equation expressed by minterms?
 $Y = F(A, B, C, D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$

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

Example-1

- What is the simplified Boolean equation for the following logic equation expressed by minterms?
 $Y = F(A, B, C, D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$

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

Example-1

- What is the simplified Boolean equation for the following logic equation expressed by minterms?
 $Y = F(A, B, C, D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$

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

Example-1

- ❖ What is the simplified Boolean equation for the following logic equation expressed by minterms?
 $Y = F(A, B, C, D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$

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

Example-1

- What is the simplified Boolean equation for the following logic equation expressed by minterms?
 $Y = F(A, B, C, D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$

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

$$Y = AB + AC + AD + BCD$$

Example-2

- Simplify the following Boolean function in sum of products form (SOP) $F(x, y, z, w) = \sum m(0, 1, 2, 5, 8, 9, 10)$

zw \ xy	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

zw \ xy	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

zw \ xy	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

➤ $F = \bar{y}\bar{w} + \bar{y}\bar{z} + \bar{x}\bar{z}w$

DON'T-CARE CONDITIONS-1

- ❖ In 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*. Whenever you see an X in a truth table, you can let it equal either 0 or 1, whichever produces a simpler logic circuit.

DON'T-CARE CONDITIONS-2



True

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

ons



$$Y = F(A, B, C, D) = \sum m(9) + d(10, 11, 12, 13, 14, 15)$$

DON'T-CARE CONDITIONS-3

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

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

$$Y = AD$$

DON'T-CARE CONDITIONS-4

◆ Ideas about don't-care conditions:

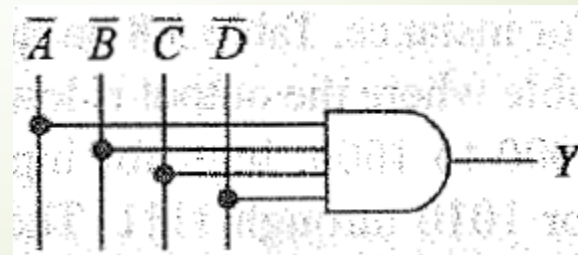
1. Given the truth table, draw a Karnaugh map with 0s, 1s, and don't-cares (X).
2. Encircle the actual 1s on the Karnaugh map in the largest groups you can find by treating the don't cares as 1s.
3. After the actual 1s have been included in groups, disregard the remaining don't cares by visualizing them as 0s.

DON'T-CARE CONDITIONS-5

What is the simplest logic circuit for the following truth

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

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

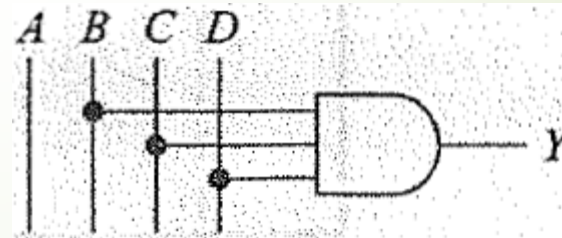


DON'T-CARE CONDITIONS-6

- ◇ Give the simplest logic circuit for following logic equation where d represents don't-care condition for following locations.

$$F(A, B, C, D) = \sum m(7) + d(10, 11, 12, 13, 14, 15)$$

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



Exam Questions

◇ VTU QP

cd	00	01	11	10
ab				
00	0 ₀	X ₁	0 ₃	0 ₂
01	X ₄	X ₅	1 ₇	1 ₆
11	0 ₁₂	1 ₁₃	0 ₁₅	0 ₁₄
10	0 ₈	1 ₉	X ₁₁	1 ₁₀

yz	00	01	11	10
wx				
00	0 ₀	1 ₁	X ₃	1 ₂
01	1 ₄	1 ₅	X ₇	1 ₆
11	0 ₁₂	X ₁₃	1 ₁₅	0 ₁₄
10	1 ₈	1 ₉	1 ₁₁	0 ₁₀

PRODUCT-Of-SUMS METHOD-1

- ◇ Given a truth table, you identify the fundamental sums needed for a logic design. Then by ANDing these sums, you get the product-of-sums equation corresponding to the truth table.
- ◇ But, in the sum-of-products method, the fundamental product produces an output 1 for the corresponding input condition. But with the product of- sums method, the fundamental sum produces an output 0 for the corresponding input condition.

PRODUCT-Of-SUMS METHOD-2

A	B	C	Y	Maxterm
0	0	0	$0 \rightarrow A + B + C$	M_0
0	0	1	1	M_1
0	1	0	1	M_2
0	1	1	$0 \rightarrow A + \bar{B} + \bar{C}$	M_3
1	0	0	1	M_4
1	0	1	1	M_5
1	1	0	$0 \rightarrow \bar{A} + \bar{B} + C$	M_6
1	1	1	1	M_7

$$Y = A + B + C = 0 + 0 + 0 = 0$$

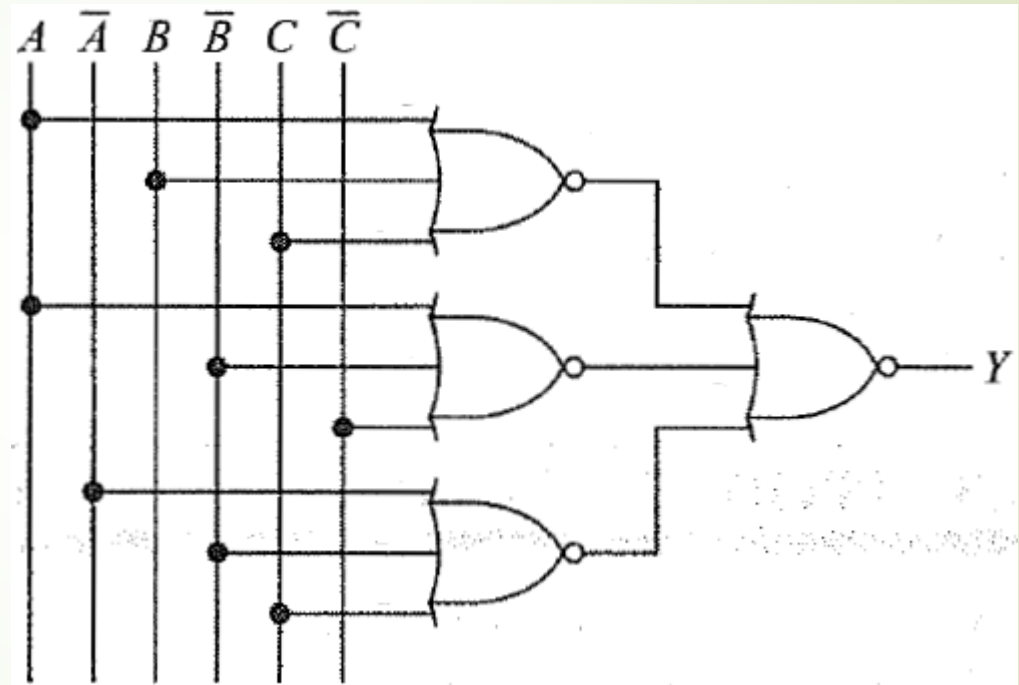
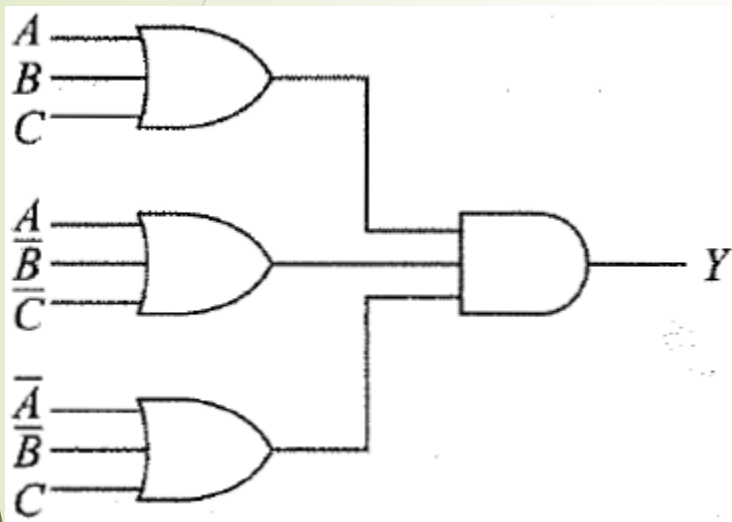
$$Y = A + \bar{B} + \bar{C} = 0 + \bar{1} + \bar{1} = 0 + 0 + 0 = 0$$

$$Y = \bar{A} + \bar{B} + C = \bar{1} + \bar{1} + 0 = 0 + 0 + 0 = 0$$

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

$$Y = F(A, B, C) = \prod M(0, 3, 6)$$

PRODUCT-Of-SUMS METHOD-3



PRODUCT-Of-SUMS METHOD-4

❖ Conversion between SOP and POS

❖ SOP and POS occupy complementary locations in a truth table.

- ❖ Identifying complementary locations,
- ❖ Changing minterm to maxterm or reverse, and
- ❖ Changing summation by product or reverse.

$$Y = F(A, B, C) = \Pi M(0, 3, 6) = \Sigma m(1, 2, 4, 5, 7)$$

$$Y = F(A, B, C) = \Sigma m(3, 5, 6, 7) = \Pi M(0, 1, 2, 4)$$

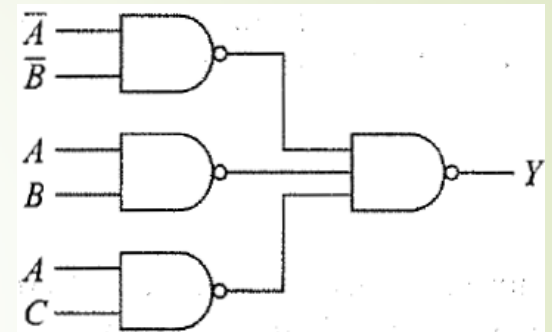
Conversion between SOP and POS

- Mapping between canonical forms
- Minterm to maxterm conversion
 - use maxterms whose indices do not appear in minterm expansion
 - e.g. $F(A,B,C) = \sum m(1,3,5,6,7) = \prod M(0,2,4)$
- Maxterm to minterm conversion
 - use minterms whose indices do not appear in maxterm expansion
 - e.g. $F(A,B,C) = \prod M(0,2,4) = \sum m(1,3,5,6,7)$
- Minterm expansion of F to minterm expansion of \bar{F}
 - use minterms whose indices do not appear
 - e.g., $F(A,B,C) = \sum m(1,3,5,6,7)$ $\bar{F}(A,B,C) = \sum m(0,2,4)$
- • Maxterm expansion of F to maxterm expansion of \bar{F}
 - use maxterms whose indices do not appear
 - e.g., $F(A,B,C) = \prod M(0,2,4)$ $\bar{F}(A,B,C) = \prod M(1,3,5,6,7)$

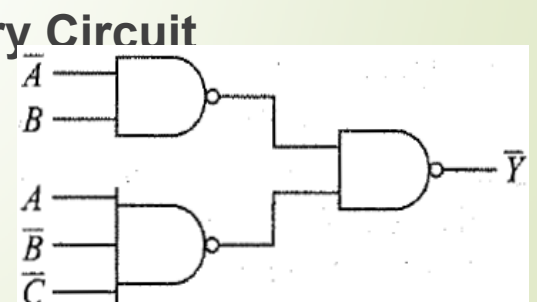
PRODUCT-OF-SUMS SIMPLIFICATION

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

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

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


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



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

PRODUCT-Of-SUMS SIMPLIFICATION-2

Karnaugh map b

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	0	0	0 ³
$\overline{A}B$	0 ₁	0	0 ₂	1
AB	1	1	1	1
$A\overline{B}$	1	1	1	1

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

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0 ²	0	1	0 ²
$\overline{A}B$	0	0	1	1
AB	x	x	x	1
$A\overline{B}$	2x	x	x	0 ₂

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

Example-1

◇ Give SOP form of $Y = F(A, B, C, D) = \prod M(0, 3, 4, 5, 6, 7, 11, 15)$

CD AB	00	01	11	10
00	0 ₀	1 ₁	0 ₃	1 ₂
01	0 ₄	0 ₅	0 ₇	0 ₆
11	1 ₁₂	1 ₁₃	0 ₁₅	1 ₁₄
10	1 ₈	1 ₉	0 ₁₁	1 ₁₀

Exam Questions

- Simplify the following using K-Map

$$Y = F(A, B, C) = \Pi M(0, 3, 6) = \Sigma m(1, 2, 4, 5, 7)$$

- By Grouping 1's

By Grouping 0's

A \ BC				
	00	01	11	10
0	0 ₀	1 ₁	0 ₃	1 ₂
1	1 ₄	1 ₅	1 ₇	0 ₆

A \ BC				
	00	01	11	10
0	0 ₀	1 ₁	0 ₃	1 ₂
1	1 ₄	1 ₅	1 ₇	0 ₆

- $Y = \bar{B}C + A\bar{B} + AC + AB\bar{C}$

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

Example

- ◇ Find the SOP for following expression using K-Map
- $$f(x_1, x_2, x_3, x_4, x_5) = \sum m(0, 1, 4, 8, 13, 15, 20, 21, 23, 26, 31) + D(5, 10, 24, 28)$$

$x_3 x_4$ \ $x_1 x_2$		00	01	11	10
		00	1	1	d
	01			d	1
	11				
	10	1		d	1

$x_5 = 0$

$x_3 x_4$ \ $x_1 x_2$		00	01	11	10
		00	1		
	01				
	11		1	1	1
	10	d	1		1

$x_5 = 1$

$$f = \bar{x}_1 \bar{x}_2 \bar{x}_4 + x_2 \bar{x}_3 \bar{x}_5 + \bar{x}_2 x_3 \bar{x}_4 + \bar{x}_1 x_2 x_3 x_5 + x_1 x_3 x_4 x_5$$

SIMPLIFICATION BY QUINE-McCLUSKY METHOD-1

❖ Tabular Method of Minimisation

- ❖ Karnaugh map method though very simple and intuitively appealing is somewhat subjective.
- ❖ It depends on the user's ability to identify patterns that gives largest size.
- ❖ Also the method becomes difficult to adapt for simplification of 5 or more variables.
- ❖ Quine-McClusky method involves preparation of two tables; one determines *prime implicants* and the other selects *essential prime implicants* to get minimal expression.

QUINE-McCLUSKY METHOD- 2

- ◇ Literal: Each appearance of a variable, either uncomplemented or complemented, is called a *literal*.
- ◇ Implicant: A product term that indicates the input valuation(s) for which a given function is equal to 1 is called an *implicant* of the function.
- ◇ **Prime Implicant**
- ◇ An implicant is called a *prime implicant* if it cannot be combined into another implicant that has fewer literals. Another way it is impossible to delete any literal in a prime implicant and still have a valid implicant.
- ◇ Prime implicants are expressions with least number of literals that represents all the terms given in a truth table. Prime implicants are examined to get essential prime implicants for a particular expression that avoids any type of duplication.

QUINE-McCLUSKY METHOD- 3

◇ Determination of Prime Implicants

◇ Example

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

QUINE-McCLUSKY METHOD-

4

- ❖ In Stage 1 of the process, to find out all the terms that gives output 1 from truth table and put them in different groups depending on how many 1s input variable combinations ($ABCD$) have.
- ❖ For example, first group has no 1 in input combination, second group has only one 1, third two 1s, fourth three 1s and fifth four 1s.

Stage 1	
$ABCD$	
0 0 0 0	(0)✓
0 0 0 1	(1)✓
0 0 1 0	(2)✓
0 0 1 1	(3)✓
1 0 1 0	(10)✓
1 1 0 0	(12)✓
1 0 1 1	(11)✓
1 1 0 1	(13)✓
1 1 1 0	(14)✓
1 1 1 1	(15)✓

QUINE-McCLUSKY METHOD- 5

- ❖ In Stage 2, we first try to combine first and second group of Stage 1, on a member to member basis.
- ❖ The rule is to see if only one binary digit is differing between two members and we mark that position by '-'. This means corresponding variable is not required to represent those members.
- ❖ Thus (0) of first group combines with (1) of second group to form (0,1) in Stage 2 and can be represented by

$A'B'C'$ (0 0 0 -).

- ❖ Proceed in the same manner to find rest of the combinations in successive groups of Stage 1 and table them as in figure.
- ❖ Note that, we need not look beyond successive groups to find such combinations as groups that are not adjacent, differ by more than one binary digit. Also note that each combination of Stage 2 can be represented by three literals.
- ❖ All the members of particular stage, which finds itself in at least one combination of next stage are tick (✓) marked. This is followed for Stage 1 terms as well as terms of other stages.

QUINE-McCLUSKY METHOD- 6

<u>Stage 1</u>		<u>Stage 2</u>	
<i>ABCD</i>		<i>ABCD</i>	
0 0 0 0	(0)✓	0 0 0 -	(0,1)✓
		0 0 - 0	(0,2)✓
<hr/>		<hr/>	
0 0 0 1	(1)✓		
0 0 1 0	(2)✓	0 0 - 1	(1,3)✓
		0 0 1 -	(2,3)✓
<hr/>		- 0 1 0	(2,10)✓
0 0 1 1	(3)✓		
1 0 1 0	(10)✓	- 0 1 1	(3,11)✓
1 1 0 0	(12)✓	1 0 1 -	(10,11)✓
<hr/>		1 - 1 0	(10,14)✓
1 0 1 1	(11)✓	1 1 0 -	(12,13)✓
1 1 0 1	(13)✓	1 1 - 0	(12,14)✓
1 1 1 0	(14)✓		
<hr/>		1 - 1 1	(11,15)✓
1 1 1 1	(15)✓	1 1 - 1	(13,15)✓
		1 1 1 -	(14,15)✓

QUINE-McCLUSKY METHOD- 7

- ❖ In Stage 3, we combine members of different groups of Stage 2 in a similar way. Now it will have two '-' elements in each combination. This means each combination requires two literals to represent it.
- ❖ For example (0,1,2,3) is represented by $A'B'$ (0 0 - -).
- ❖ There are three other groups in Stage 3; (2,10,3,11) represented by $B'C$, (10,14,11,15) by AC and (12,13,14,15) by AB .
- ❖ Note that, (0,2,1,3), (10,11,14,15) and (12,14,13,15) get represented by $A'B$, AC and AB respectively and do not give any new term.

QUINE-McCLUSKY METHOD- 8

<u>Stage 1</u>		<u>Stage 2</u>		<u>Stage 3</u>	
<i>ABCD</i>		<i>ABCD</i>		<i>ABCD</i>	
0 0 0 0	(0)✓	0 0 0 -	(0,1)✓	0 0 - -	(0,1,2,3)
		0 0 - 0	(0,2)✓	0 0 - -	(0,2,1,3)
0 0 0 1	(1)✓				
0 0 1 0	(2)✓	0 0 - 1	(1,3)✓	- 0 1 -	(2,10,3,11)
		0 0 1 -	(2,3)✓		
		- 0 1 0	(2,10)✓	1 - 1 -	(10,11,14,15)
0 0 1 1	(3)✓			1 - 1 -	(10,14,11,15)
1 0 1 0	(10)✓	- 0 1 1	(3,11)✓	1 1 - -	(12,13,14,15)
1 1 0 0	(12)✓	1 0 1 -	(10,11)✓	1 1 - -	(12,14,13,15)
		1 - 1 0	(10,14)✓		
		1 1 0 -	(12,13)✓		
1 0 1 1	(11)✓	1 1 - 0	(12,14)✓		
1 1 0 1	(13)✓				
1 1 1 0	(14)✓	1 - 1 1	(11,15)✓		
		1 1 - 1	(13,15)✓		
1 1 1 1	(15)✓	1 1 1 -	(14,15)✓		

QUINE-McCLUSKY METHOD- 9

- ◇ There is no Stage 4 for this problem as no two members of Stage 3 has only one digit changing among them. This completes the process of determination of prime implicants.
- ◇ The rule is all the terms that are not ticked at any stage is treated as prime implicants for that problem.

QUINE-McCLUSKY METHOD- 10

- ❖ Selection of Prime Implicants
- ❖ Once we are able to determine prime implicants that covers all the terms of a truth table we try to select essential prime implicants and remove redundancy or duplication among them.
- ❖ To find a minimum expression we construct a *prime implicant table* in which there is a row for each prime implicant and a column for each minterm that must be covered.
- ❖ Note- don't care values are not included.
- ❖ Then we place check marks to indicate the minterms covered by each prime implicant.

QUINE-McCLUSKY METHOD-

	0	1	2	3	10	11	12	13	14	15
$A'B'$ (0,1,2,3)	√	√	√	√						
$B'C$ (2,3,10,11)			√	√	√	√				
AC (10,11,14,15)					√	√			√	√
AB (12,13,14,15)							√	√	√	√

- ❖ Selection of essential prime implicants from this table is done in the following way.
- ❖ If column has a single √, then the implicant associated with the row is essential. It must appear in final expression.
- ❖ Find minimum set of rows that cover the remaining columns Or Find minimum number of prime implicants that covers all the minterms.
- ❖ E.g. Find $A'B'$ and AB cover terms that are not covered by others and they are essential prime implicants. $B'C$ and AC among themselves cover 10, 11 which are not covered by others.
- ❖ So, one of them has to be included in the list of essential prime implicants making it three.

$$Y = A'B' + B'C + AB \text{ or } Y = A'B' + AC + AB$$

QUINE-McCLUSKY METHOD-12

$$Y = F(A, B, C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C$$

◇ Get a minimized expression for

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

Stage 1	Stage 2
ABC	ABC
000 (0) ✓	00- (0, 1)
001 (1) ✓	0-1 (1, 3)
011 (3) ✓	-01 (1, 5)
101 (5) ✓	

	0	1	3	5
A'B'	✓	✓		
A'C		✓	✓	
B'C		✓		✓

All are essential
 $Y = A'B' + A'C + B'C$

Prime implicants only from stage 2.
 They are:
 00-(A'B'), 0-1 (A'C) and -01 (B'C)

QUINE-McCLUSKY METHOD- 13

◇ Get simplified expression of $Y = F(A, B, C, D, E)$
 $= \sum m(0, 1, 2, 3, 4, 5, 12, 13, 14, 26, 27, 28, 29, 30)$ using Quine-McClusky method.

Decimal	A	B	C	D	E
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
26	1	1	0	1	0
27	1	1	0	1	1
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	0

HAZARDS AND HAZARD COVERS-1

- ❖ Simplification techniques that give minimal expression for a logic equation which in turn requires minimum hardware for realization.
- ❖ Some practical problems, in certain cases we may prefer to include more terms than given by simplification techniques.
- ❖ So far we considered gates generating outputs instantaneously. But practical circuits always offer finite propagation delay though very small, in nanosecond order.
- ❖ This gives rise to several *hazards (Glitch)* and *hazard covers*

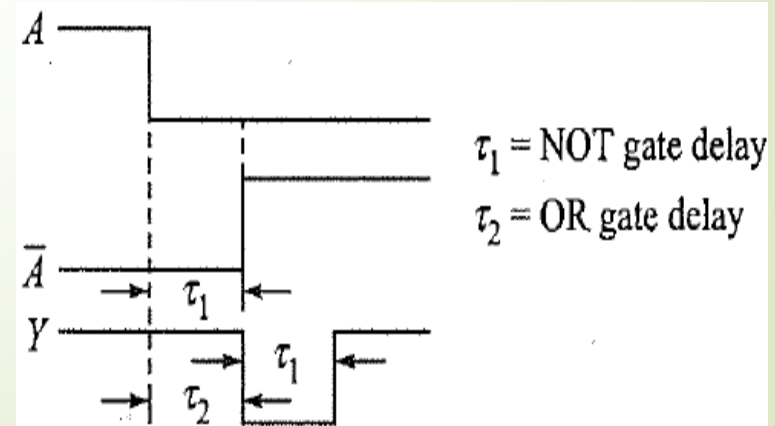
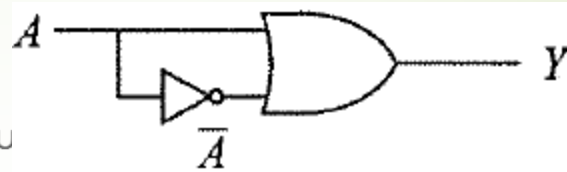
HAZARDS AND HAZARD COVERS-2

- ◇ Static-1 Hazard $1 \rightarrow 0$
- ◇ Static-0 Hazard $0 \rightarrow 1$
- ◇ Dynamic Hazard

HAZARDS AND HAZARD COVERS-3

Static-1 Hazard

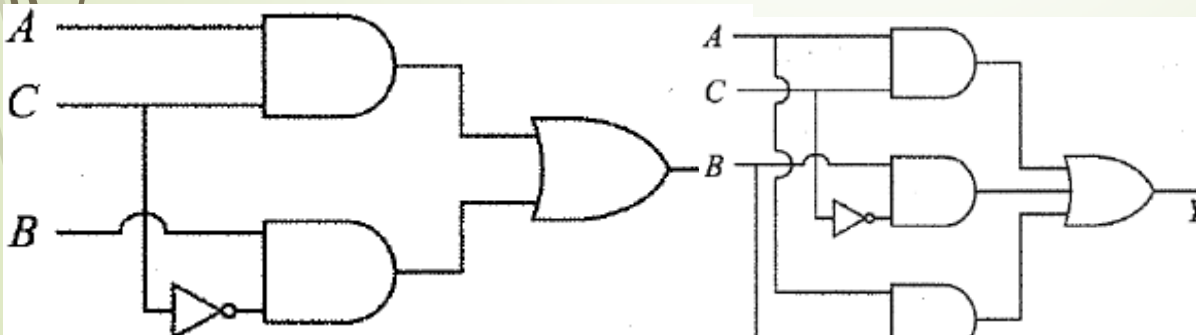
- For example $Y = A + \bar{A}$ ou is always logic 1.
- But the NOT gate output takes finite time to become 1 following $1 \rightarrow 0$ transition of A.
- Thus for the OR gate there are two zeros appearing at its input for that small duration, resulting a 0 at its output. The width of this zero is in nanosecond order and is called a glitch.
- For combinational circuits it may go unnoticed but in sequential circuit may cause major malfunctioning.



HAZARDS AND HAZARD COVERS-4

- For this circuit input $B = 1$ and $A = 1$ and then C makes transition $1 \rightarrow 0$. The output shows glitch.
- Another grouping of same K-Map, the additional term AB is term included.
- This circuit though require more hardware than minimal representation, is hazard free.
- The additional term AB ensures $Y = 1$ for $A = 1, B = 1$ through the third input of final OR gate and a $Y = B\bar{C} + AC + AB$ $1 \rightarrow 0$ transition at C does not affect output.

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

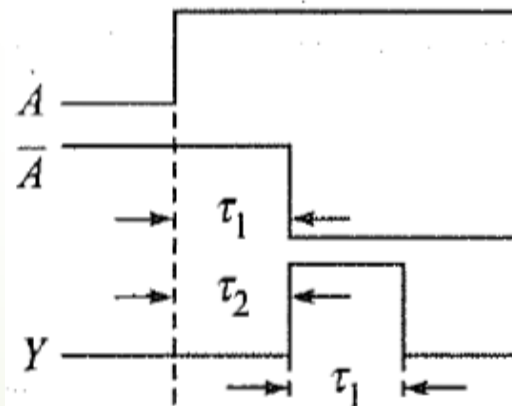
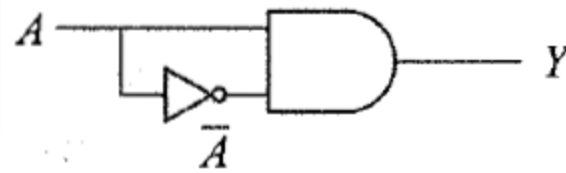


	\bar{C}	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	1	0
AB	1	1
$A\bar{B}$	0	1

$Y = B\bar{C} + AC + AB$

HAZARDS AND HAZARD COVERS-5

- **Static-0 Hazard**
- For example $Y = A.\bar{A}$ output is always logic 0.
- But the NOT gate output takes finite time to become 0 following a 0→1 transition of A.
- Thus for final AND gate there are two ones appearing at its input for a small duration resulting a 1 at its output. This $Y=1$ occurs for a very small duration.

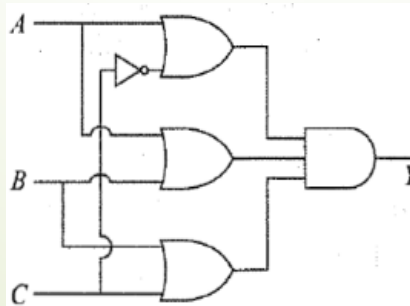
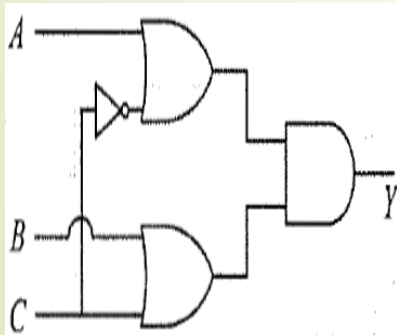


τ_1 = NOT gate delay

τ_2 = OR gate delay

HAZARDS AND HAZARD COVERS-6

- ❖ If $B = 0$, $A = 0$ and C makes a transition $0 \rightarrow 1$ there will be static-0 hazard occurring at output.
- ❖ To prevent this we add one additional group, i.e. one more sum term ($A + B$).
- ❖ The additional term ($A + B$) ensures $Y = 0$ for $A = 0$, $B = 0$ through the third input of final AND gate and a $0 \rightarrow 1$ transition at C does not affect output.



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

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

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

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

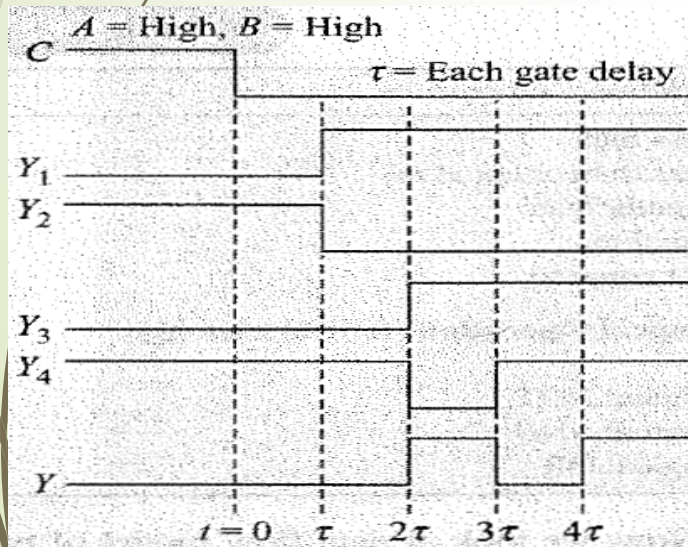
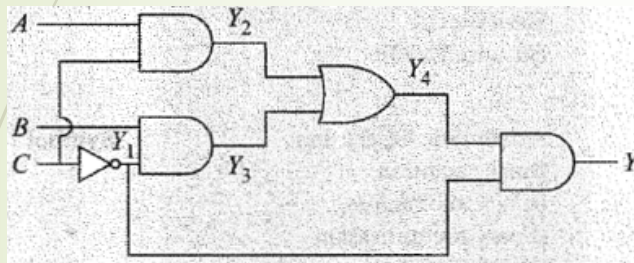
HAZARDS AND HAZARD COVERS-7

➤ Dynamic Hazard

- Dynamic hazard occurs when circuit output makes multiple transitions before it settles to a final value while the logic equation asks for only one transition.
- An output transition designed as $1 \rightarrow 0$ may give $1 \rightarrow 0 \rightarrow 1 \rightarrow 0$ when such hazard occurs and a $0 \rightarrow 1$ can behave like $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$.
- E.g. $Y = A + \bar{A}.A$ or $Y = (A + \bar{A}).A$
- These occur in multilevel circuits having implicit static-1 and/or static-0 hazards.

HAZARDS AND HAZARD COVERS-8

- Check if the circuit shown below exhibit dynamic hazard. Show how output varies with time if dynamic hazard occurs. Consider all the gates have equal propagation delay of τ nanosecond. Also mention how the hazard can be prevented.



- Form circuit equation
- Clearly for $A = 1, B = 1$ we get $f = (C + C') \cdot C$ which shows potential dynamic hazard with an implicit static-1 hazard. Figure 3.38b shows how
- a transition $1 \rightarrow 0$ at input C for $AB = 1$ causes dynamic hazard at the output.
- The hazard can be prevented by using an additional two input AND gate fed by input A and B and replacing two input OR gate by a three input OR gate. The additional (third) input of OR gate will be fed by output of the new AND gate.

HDL IMPLEMENTATION MODELS-1

❖ Dataflow Modeling

- ❖ Verilog provides a keyword **assign** and a set of operators to describe a circuit through its behavior or function.
- ❖ No need to define explicitly any gate structure using **and**, **or** etc. and it is not necessary to use intermediate variables through **wire** showing gate level interconnections.
- ❖ Verilog compiler handles this while compiling such a model.
- ❖ All **assign** statements are concurrent, i.e. order in which they appear do not matter and also continuous,

HDL IMPLEMENTATION

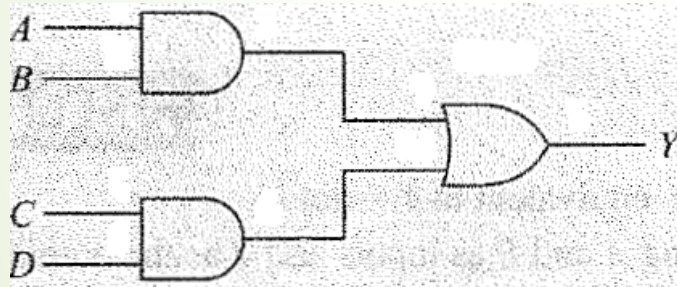
MODELS-2

◇ List of Verilog Operator

<i>Relational Operation</i>	<i>Symbol</i>	<i>Bit-wise Operation</i>	<i>Symbol</i>
Less than	<	Bit-wise NOT	~
Less than or equal to	<=	Bit-wise AND	&
Greater than	>	Bit-wise OR	
Equal to	==	Bit-wise Ex-OR	^
Not equal to	!=		
Logical Operation (for expressions)	Symbol	Arithmetic Operation	Symbol
Logical NOT	!	Binary addition	+
Logical AND	&&	Binary subtraction	-
Logical OR		Binary multiplication	*
		Binary division	/

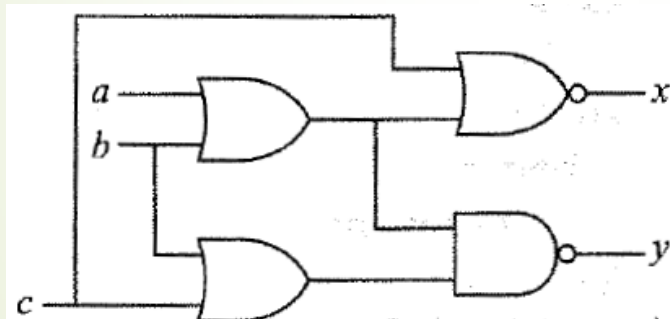
- ~ operator has higher precedence over & and |; while & and | are at same level.
- To avoid confusion and improve readability it is always advised to use parentheses (...) that has second highest precedence below bit select [...].

HDL IMPLEMENTATION MODELS-3



```
module ex1(A,B,C,D,Y);  
input A,B,C,D;  
output Y;  
assign Y=(A&B)|(C&D);  
endmodule
```

HDL IMPLEMENTATION MODELS-4



```
module ex2(a,b,c,x,y);  
input a,b,c;  
output x,y;  
assign x=~((a|b)|c);  
assign y=~((a|b)|(b|c));  
endmodule
```

HDL IMPLEMENTATION MODELS-5

❖ Behavioural Modeling

- ❖ In a behavioral model, statements are executed sequentially following algorithmic description.
- ❖ It always uses **always** keyword followed by a sensitivity list. The procedural statements following **always** is executed only if any variable within sensitivity list changes its value.
- ❖ Procedure assignment or output variables within **always** must be of register type, defined by **reg** which unlike **wire** is not continuously updated but only after a new value is assigned to it.

HDL IMPLEMENTATION MODELS-6

- ◇ Write verilog code for $Y = AB + CD$, i.e. $Y = 1$ if $AB = 11$ or if $CD = 11$, otherwise $Y = 0$ using if ... else.

```
module eg3(A,B,C,D,Y)
input A,B,C,D;
output Y;
reg Y;          /* Y is a output after procedural assignment within always block, hence defined as reg*/
always @ (A or B or C or D)    //A,B,C,D form sensitivity list
if((A==1) && (B==1))
    Y=1;
else if((C==1) && (D==1))
    Y=1;
else
    Y=0;
endmodule
```

Module-3

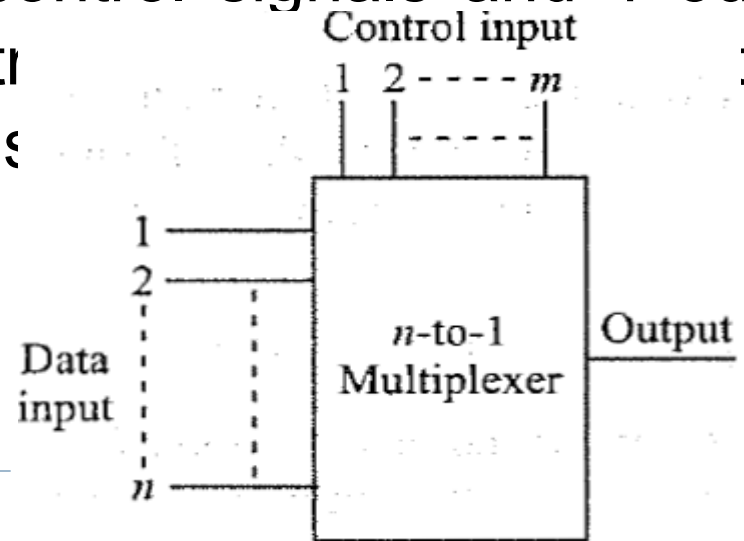
Data-Processing Circuits and Flip- Flops

Text Books Referred

- “ Donald P Leach, Albert Paul Malvino & Goutam Saha: Digital Principles and Applications, 7th Edition, Tata McGraw Hill, 2015

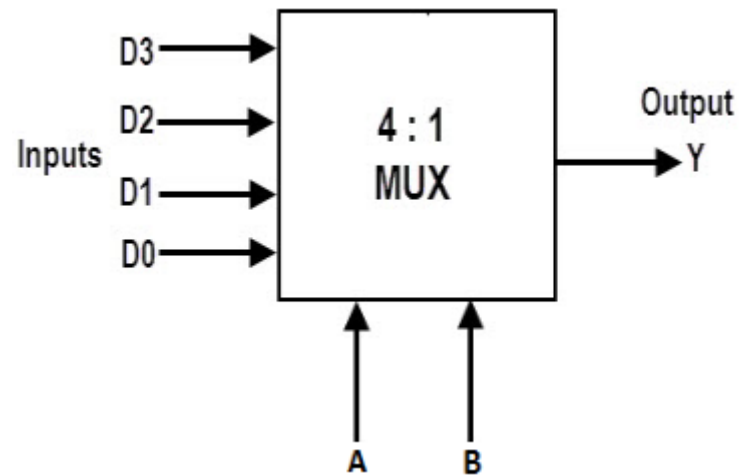
MULTIPLEXERS-1

- “ *Multiplex means many into one.*
- “ A multiplexer (also called data selector) is a circuit with many inputs but only one output. By applying control signals (Select Input), can steer any input to the output.
- “ Below shows the block diagram of MUX. The circuit has n input signals, m control signals and 1 output signal. Note that, m control signals can select the most 2^m input signals thus



MULTIPLEXERS-2

- “ The block diagram of a 4-to-1 multiplexer is shown below and its truth table.
- “ Depending on control inputs A , B one of the four inputs D_0 to D_3 is steered to output Y .



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

MULTIPLEXERS-3

- “ 4-to-1 MUX logic circuit
- “ Logic equation of this circuit is a SOP representation.

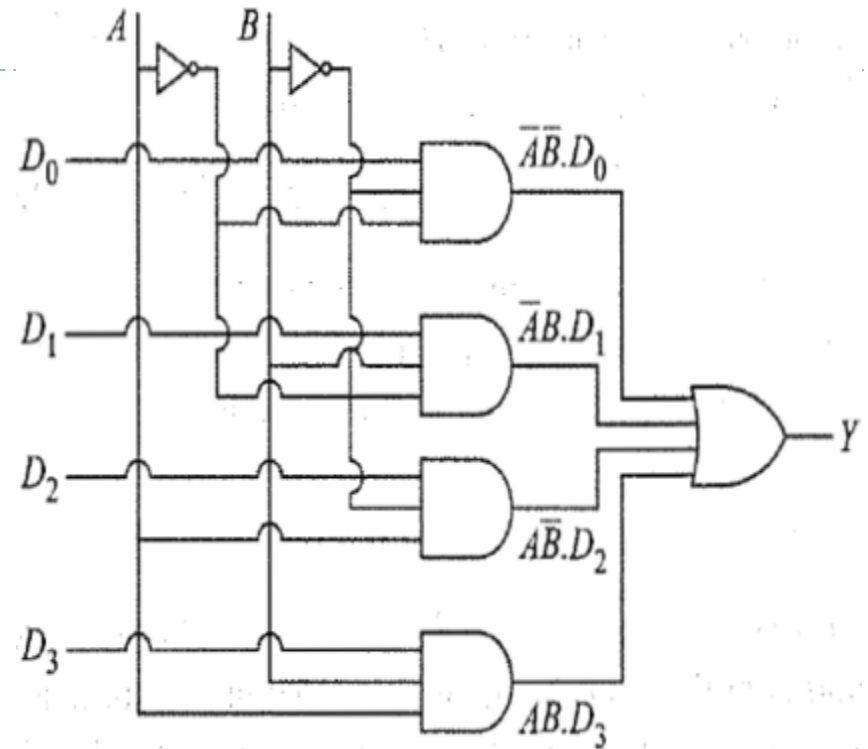
$$Y = A'B'.D_0 + A'B.D_1 + AB'.D_2 + AB.D_3$$

If $A = 0, B = 0,$

$$Y = 0'0'.D_0 + 0'.0.D_1 + 0.0'.D_2 + 0.0.D_3$$

$$Y = 1.1.D_0 + 1.0.D_1 + 0.1.D_2 + 0.0.D_3$$

$$Y = D_0$$



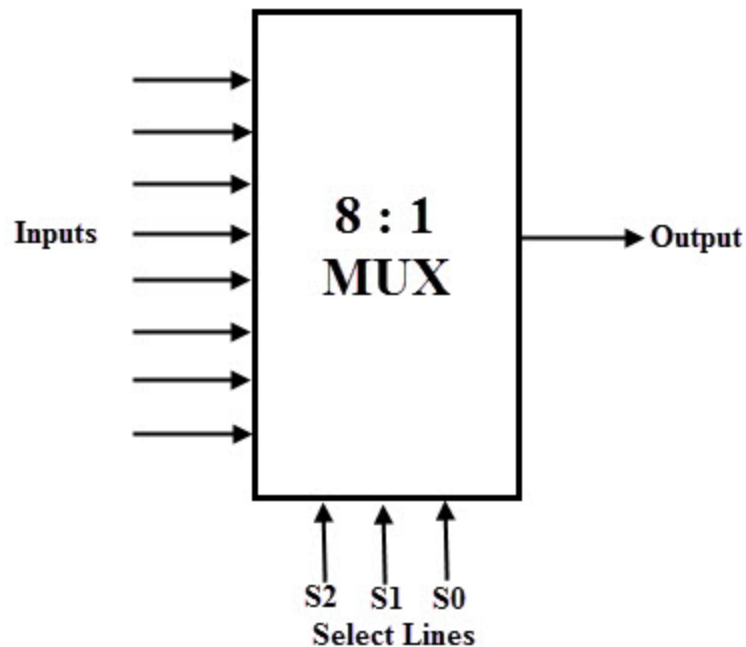
A	B	Y
0	0	D ₀
0	1	D ₁
1	0	D ₂
1	1	D ₃

MULTIPLEXERS-4

- “ In other words, for $AB = 00$, the first AND gate to which $D0$ is connected remains active and equal to $D0$ and all other AND gate are inactive with output held at logic 0.
- “ Thus, multiplexer output Y is same as $D0$. If $D0 = 0$, $Y = 0$ and if $D0 = 1$, $Y = 1$.
- “ Commercial multiplexers ICs come in integer power of 2, e.g. 2-to-1, 4-to-1, 8-to-1, 16-to-1 multiplexers.
- “ Write the 2 to 1 MUX block diagram, equation and its truth table.

MULTIPLEXERS-5

- “ Write the 8 to 1 MUX block diagram, equation and it truth table.

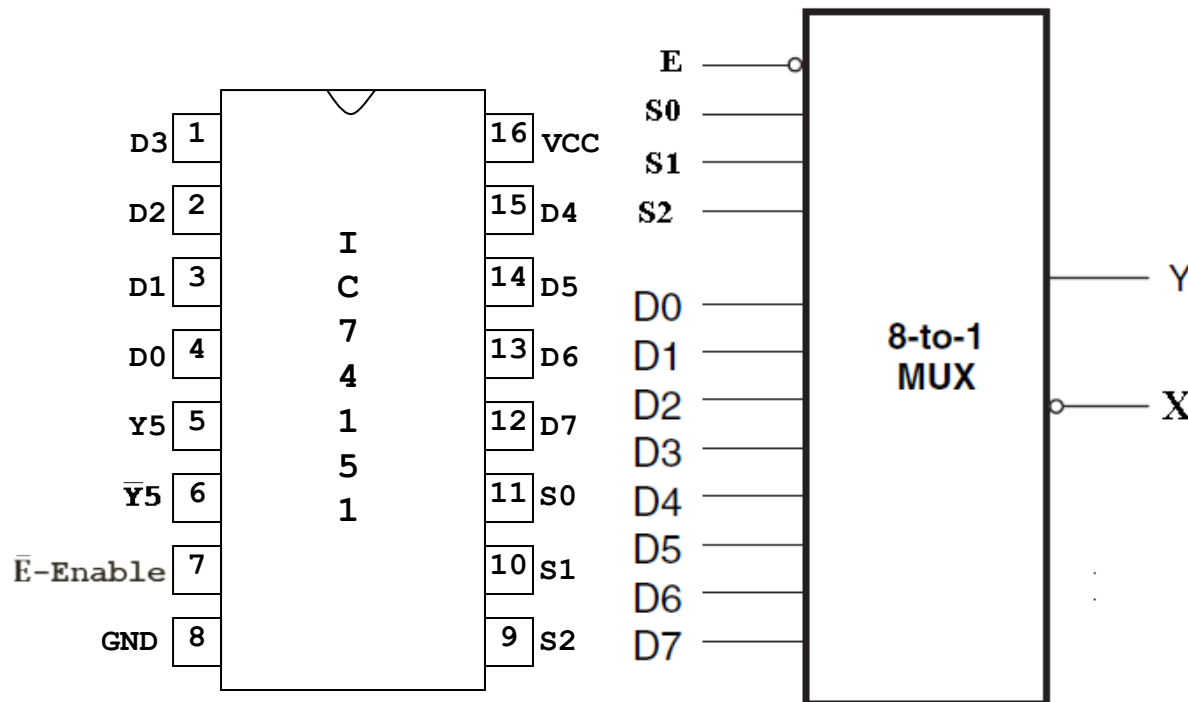


Select Data Inputs			Output
S ₂	S ₁	S ₀	Y
0	0	0	D ₀
0	0	1	D ₁
0	1	0	D ₂
0	1	1	D ₃
1	0	0	D ₄
1	0	1	D ₅
1	1	0	D ₆
1	1	1	D ₇

$$Y = A'B'C'.D_0 + A'B'C.D_1 + A'BC'.D_2 + A'BC.D_3 + AB'C'.D_4 + AB'C.D_5 + ABC'.D_6 + ABC.D_7$$

MULTIPLEXERS-6

“ IC 74151

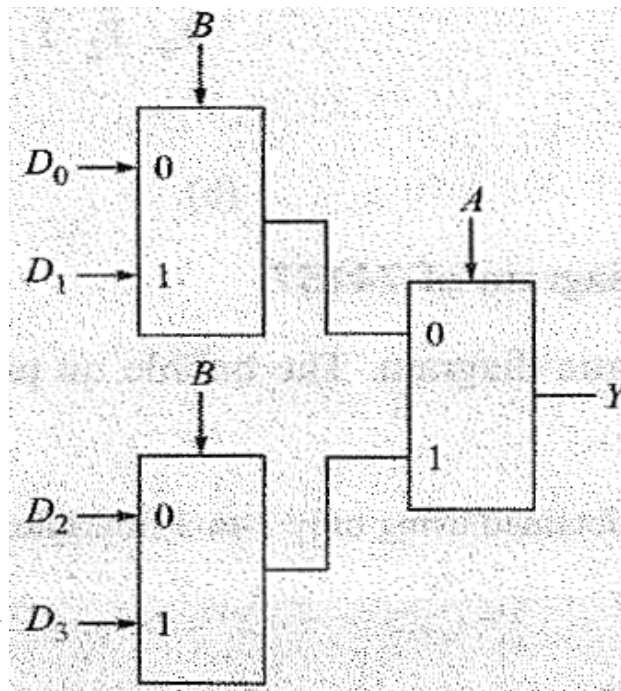


Inputs				Output	
Select			Enable E	Y X	
S2	S1	S0			
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

E : ENABLE input
 S0,S1,S2 : Select inputs
 D0-D7 : Data inputs
 Y,X : outputs

MULTIPLEXERS-7

- “ Show how 4-to-1 multiplexer can be obtained using only 2-to-1 multiplexer.
- “ Logic equation for 2-to-1 Multiplexer: $Y = A'D_0 + AD_1$
- “ Logic equation for 4-to-1 Multiplexer:
 $Y = A'B'D_0 + A'BD_1 + AB'D_2 + ABD_3$
- “ This can be rewritten as, $Y = A'(B'D_0 + BD_1) + A(B'D_2 + BD_3)$



MULTIPLEXERS-8

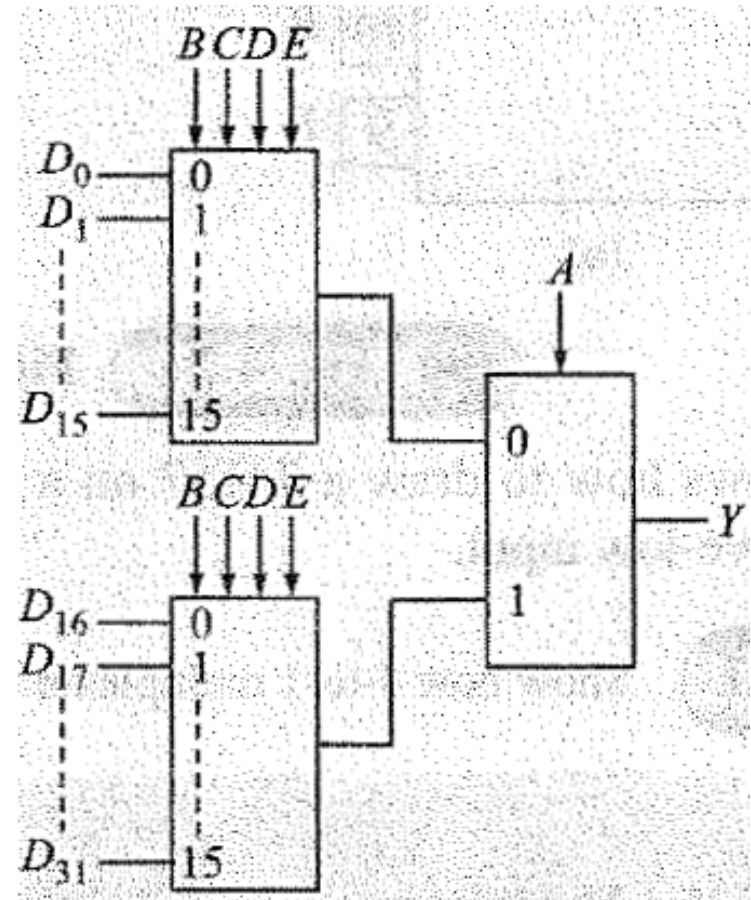
- “ Realize $Y = A'B + B'C' + ABC$ using an 8-to-1 multiplexer.
- “ First we express Y as a function of minterms of three variables. Thus
- “ $Y = A'B + B'C' + ABC$
- “ $Y = A'B(C' + C) + B'C'(A' + A) + ABC$ [As, $X + X' = 1$]
- “ $Y = A'B'C' + A'BC' + A'BC + AB'C' + ABC$
- “ Comparing this with equation of 8 to 1 multiplexer, we find by substituting $D0 = D2 = D3 = D4 = D1 = 1$ and $D1 = D5 = D6 = 0$.

MULTIPLEXERS-9

- “ Can it be realized above equation with a 4-to-1 multiplexer?
- “ The 4-to-1 multiplexer generates 4 minterms for different combinations of AB . We rewrite given logic equation in such a way that all these terms are present in the equation.
- “ $Y = A'B + B'C' + ABC$
- “ $Y = A'B + B'C'(A' + A) + ABC$ $[As, X + X' = 1]$
- “ $Y = A'B'.C' + A'B.1 + AB'.C' + AB.C$
- “ Compare above with equation of a 4-to-1 multiplexer. We see $D0 = C'$, $D1 = 1$, $D2 = C'$ and $D3 = C$ generate the given logic function.

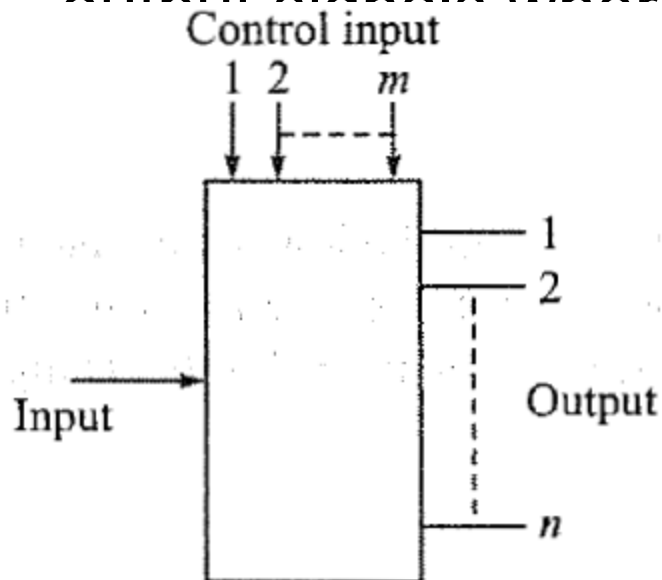
MULTIPLEXERS-10

- “ Design a 32-to-1 multiplexer using two 16-to-1 multiplexers and one 2-to-1 multiplexer.
- “ A 32-to-1 multiplexer requires $\log_2 32 = 5$ select lines say, $ABCDE$. The lower 4 select lines $BCDE$ choose 16-to-1 multiplexer outputs. The 2-to-1 multiplexer chooses one of the output of two 16-to-1 multiplexers depending on what appears in the 5th select line, A .



DEMULTIPLEXERS-1

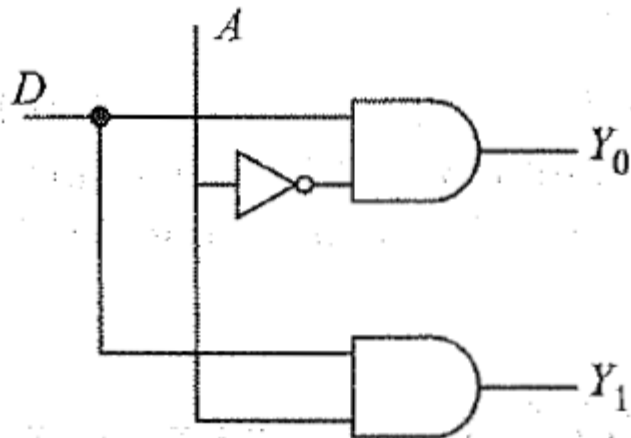
- “ Demultiplex means one into many.
- “ A *demultiplexer* is a logic circuit with one input and many outputs. By applying control signals, can steer the input signal to one of the output lines. The circuit has 1 input signal, m control or select signals and n output signals where $n \leq 2^m$.



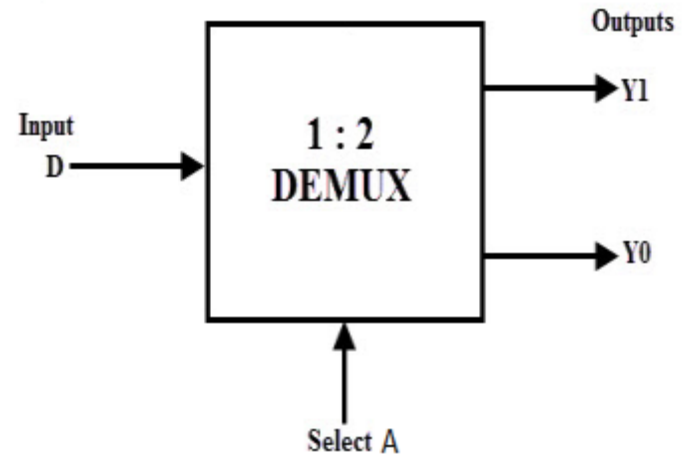
Demultiplexer block diagram

DEMULTIPLEXERS-2

- “ For 1 to 2 DMUX
- “ The logic equation
- “ $Y_0 = D A'$ $Y_1 = D A$

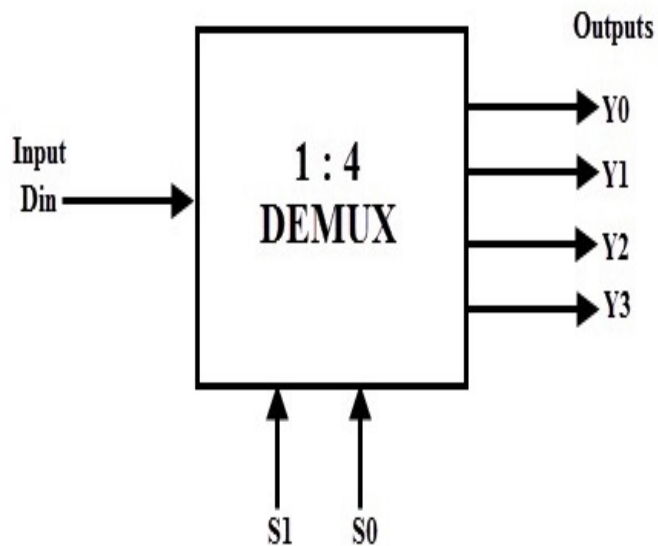


Logic circuit of 1-to-2 demultiplexer



DEMULTIPLEXERS-3

- Write block diagram, truth table and logic equation of 1 to 4 DMUX.

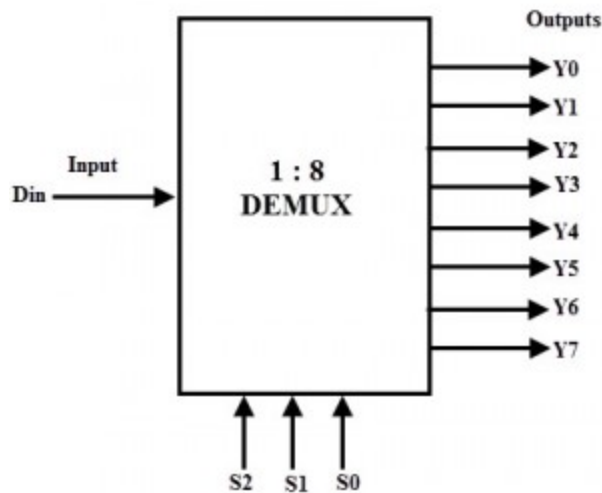


Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

- $Y_0 = \overline{S_1}\overline{S_0}D$
- $Y_1 = \overline{S_1}S_0D$
- $Y_2 = S_1\overline{S_0}D$
- $Y_3 = S_1S_0D$

DEMULTIPLEXERS-4

“ Write block diagram, truth table and log equation of 1 to 8 DMUX.



$$Y_0 = D \overline{S_2} \overline{S_1} \overline{S_0}$$

$$Y_1 = D \overline{S_2} \overline{S_1} S_0$$

$$Y_2 = D \overline{S_2} S_1 \overline{S_0}$$

$$Y_3 = D \overline{S_2} S_1 S_0$$

$$Y_4 = D S_2 \overline{S_1} \overline{S_0}$$

$$Y_5 = D S_2 \overline{S_1} S_0$$

$$Y_6 = D S_2 S_1 \overline{S_0}$$

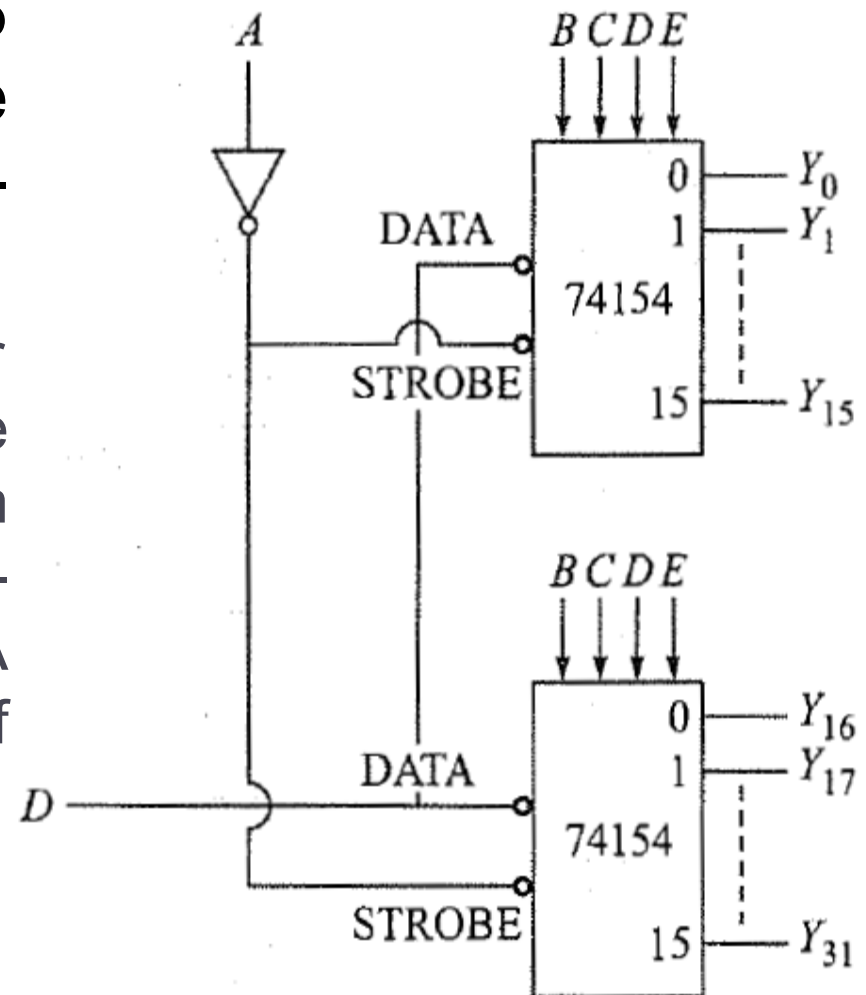
$$Y_7 = D S_2 S_1 S_0$$

Data Input	Select Inputs			Outputs							
D	S ₂	S ₁	S ₀	Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
D	0	0	0	0	0	0	0	0	0	0	D
D	0	0	1	0	0	0	0	0	0	D	0
D	0	1	0	0	0	0	0	0	D	0	0
D	0	1	1	0	0	0	0	D	0	0	0
D	1	0	0	0	0	0	D	0	0	0	0
D	1	0	1	0	0	D	0	0	0	0	0
D	1	1	0	0	D	0	0	0	0	0	0
D	1	1	1	D	0	0	0	0	0	0	0

DEMULTIPLEXERS-5

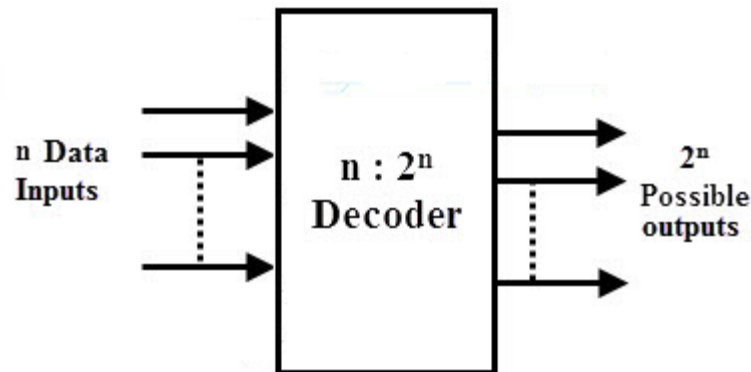
“ Show how two 1-to-16 demultiplexers can be connected to get a 1-to-32 demultiplexer.

“ A 1-to-32 demultiplexer has 5 select variable ABCDE. Four of them (BCDE) are fed to two 1-to-16 DMUX. Fifth A used to select the one of these DMUX



DECODER-1

- “ **A *decoder* is similar to a demultiplexer, with one exception-there is no data input.**
- “ *Also called binary-to-decimal decoder.*
- “ The name decoder means translating of coded information from one format into another.
- “ A binary decoder is a multi-input, multi-output combinational circuit that converts a binary code of n input lines into a one out of 2^n output code.
- “ Depending on the number of input lines, the inputs of a binary code can be 2-bit or 3-bit or 4-bit codes. Upon the availability of 2^n lines, it activates the one of its output by deactivating (making logic 0) all other input whenever it receives n inputs.

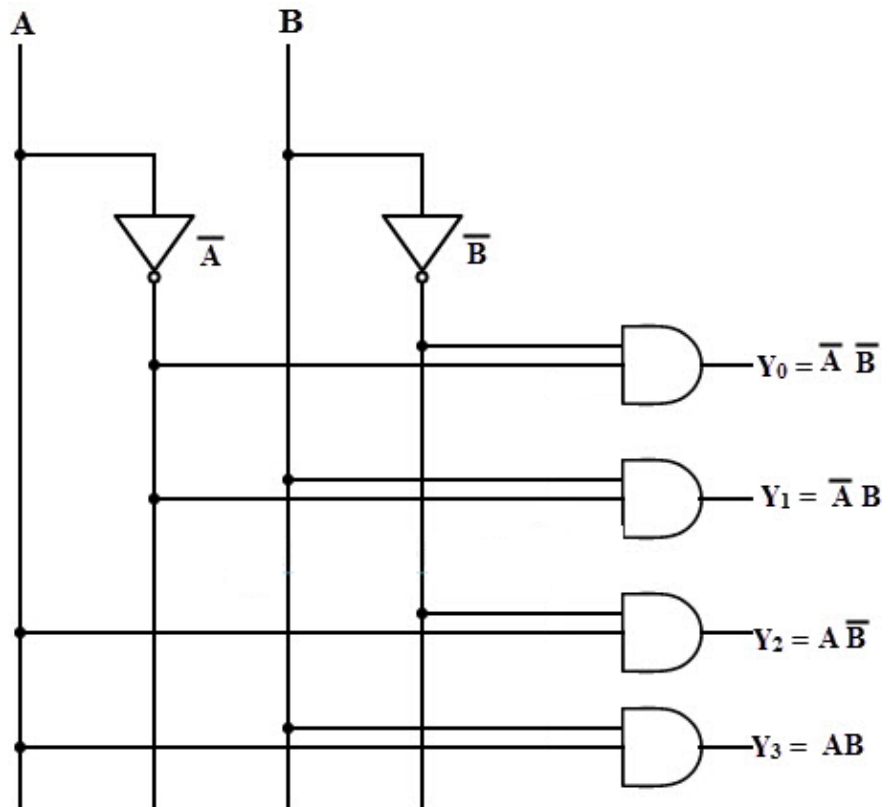
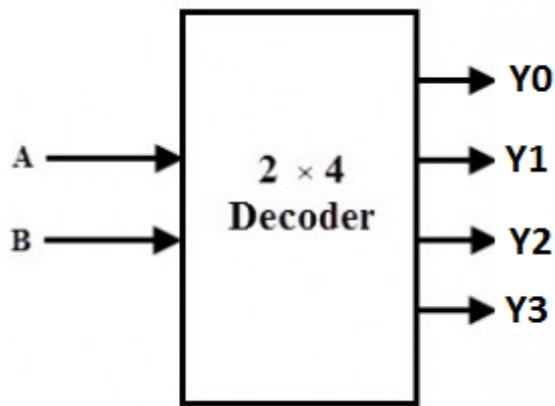


DECODER-2

- “ The most commonly used practical binary decoders are 2-to-4 decoder, 3-to-8 decoder and 4-to-16 line binary decoder.
- “ 2-to-4 decoder also called 1 of 4
- “ 3-to-8 decoder also called 1 of 8
- “ 4-to-16 line binary decoder also called 1 of 16

DECODER-3

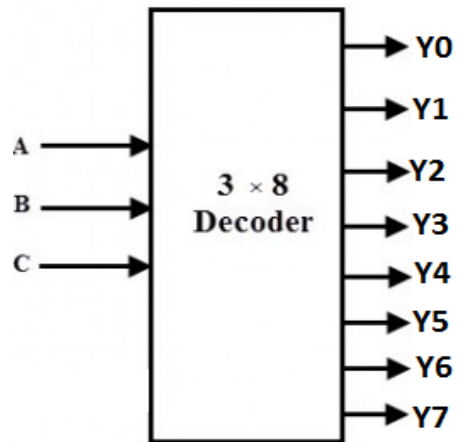
“ 2-to-4 Binary Decoder (1 of 4 Decoder)”



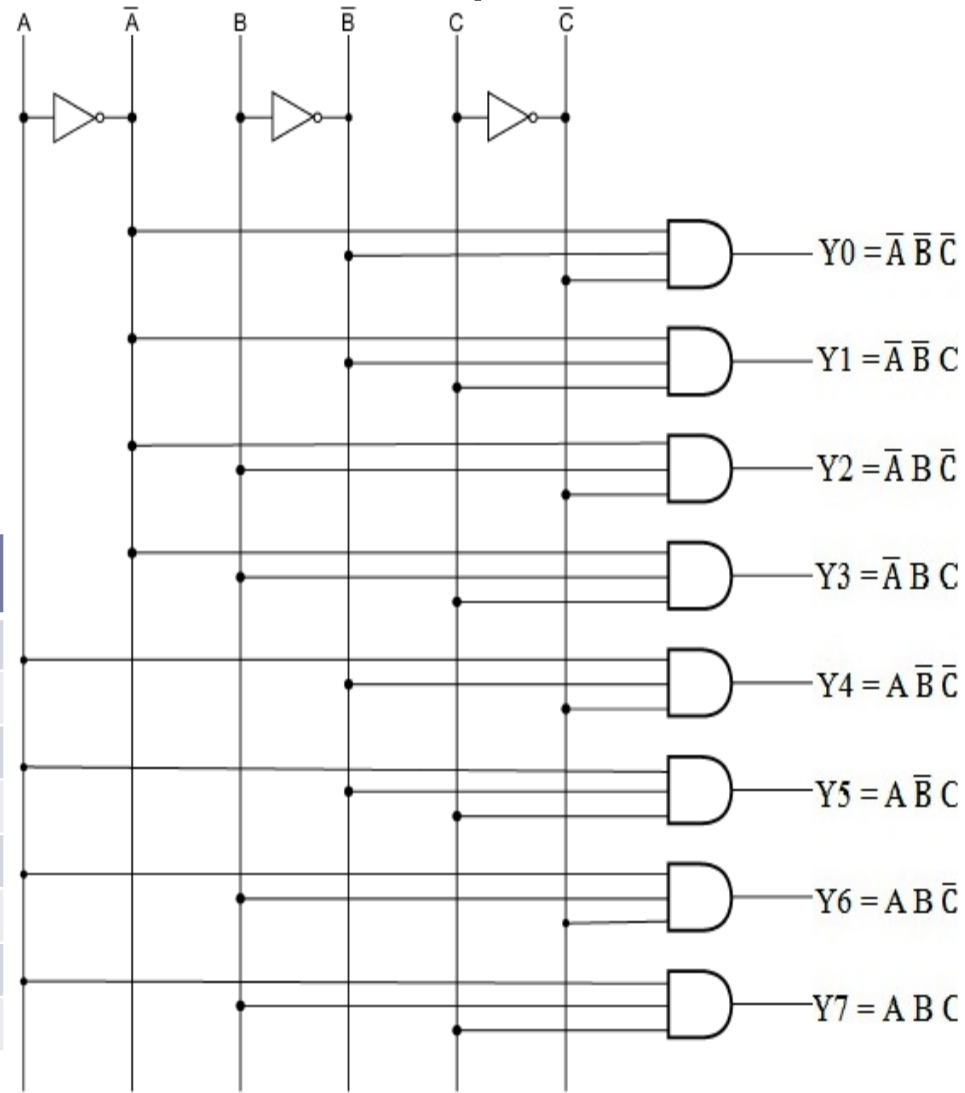
Input		Output			
A	B	Y0	Y1	Y2	Y3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

DECODER-4

“ 3-to-8 Binary Decoder (1 of 8 Decoder)



A	B	C	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
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



DECODER-5

“ Similarly for 4 to 16 or 1 of 16 Decoder

DECODER-6

“ Applications of Decoders

- “ Decoders are greatly used in applications where the particular output or group of outputs to be activated only on the occurrence of a specific combination of input levels.

“ Binary to Decimal Decoder

- “ Decoders are used to get the decimal digit corresponding to a specific input combination. A BCD number needs 4 binary digits to represent the 0 to 9 decimal digits, thus it consists of 4 input lines. It consists of 10 output lines corresponding to 0 to 9 decimal digits. (1 of 10 line decoder)

“ Address Decoders

- “ Amongst its many uses, a decoder is widely used to decode the particular memory location in the computer memory system. Decoders accept the address code generated by the CPU which is a combination of address bits for a specific location in the memory. In a memory system, there are several memory ICs are combined and each one has their unique address to distinguish from other memory locations. In such cases a decoder built in the memory ICs circuitry, is used to select a memory IC in response to a range of addresses by decoding the most significant bits of the systems address, thereby a particular memory location or IC is selected.

“ Instruction Decoder

- “ Another application of the decoder can be found in the control unit of the central processing unit. This decoder is used to decode the program instructions in order to activate the specific control lines such that different operations in the ALU of the CPU are carried out.

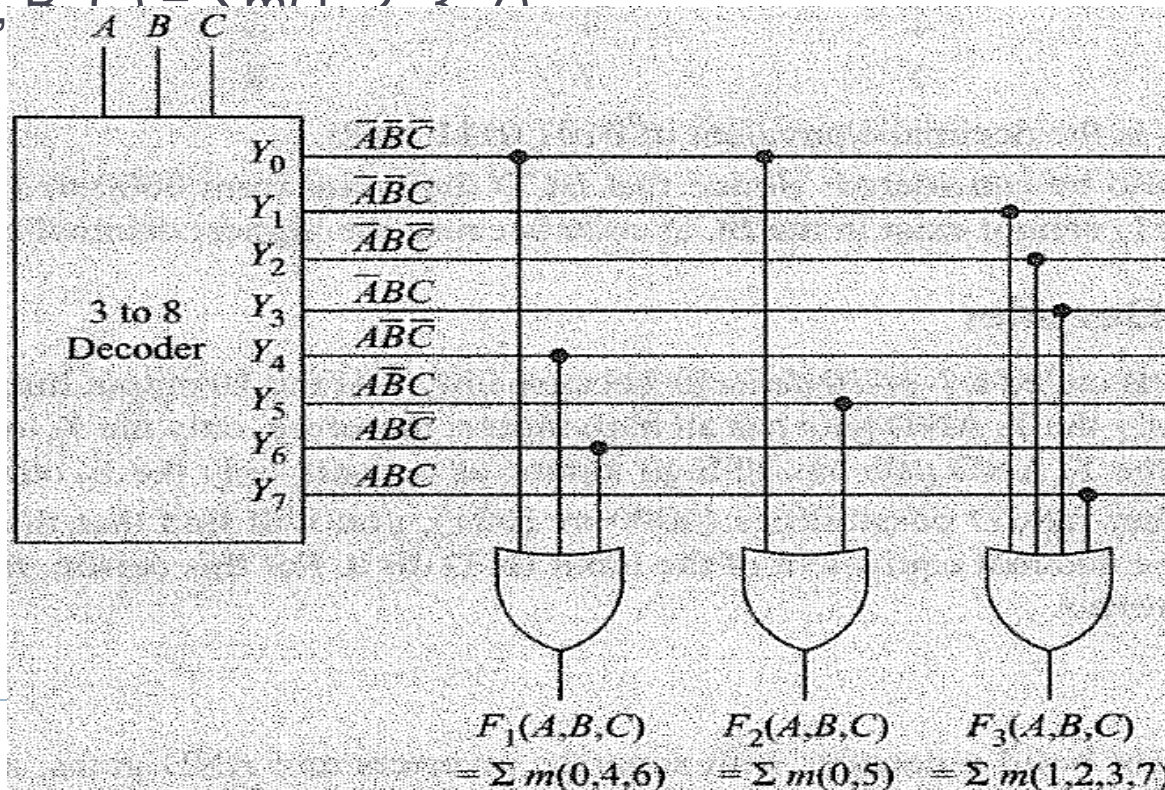
DECODER-7

“ Show how using a 3-to-8 decoder and multi-input OR gates following Boolean expressions can be realized simultaneously.

$$F_1(A, B, C) = \sum m(0, 4, 6);$$

$$F_2(A, B, C) = \sum m(0, 5);$$

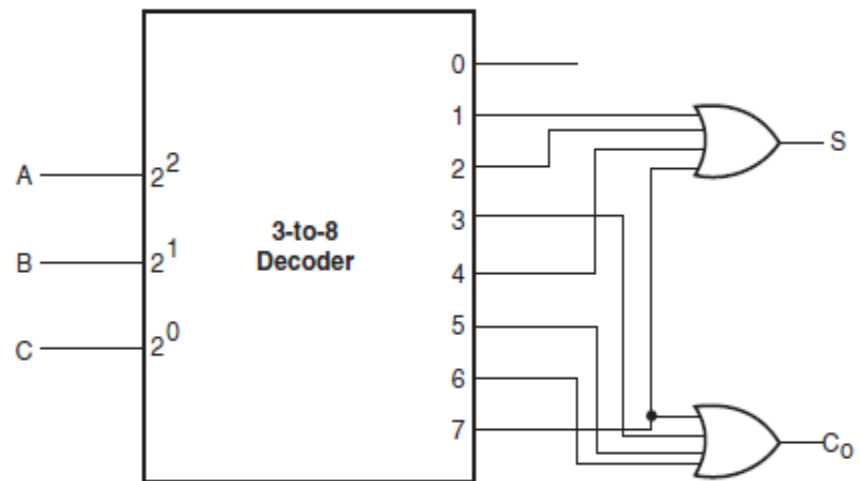
$$F_3(A, B, C) = \sum m(1, 2, 3, 7)$$



DECODER-8

- “ Implement a full adder circuit using a 3-to-8 line decoder.
- “ Sum output $S = \sum m(1\ 2\ 4\ 7)$
- “ Carry output $C_o = \sum m(3\ 5\ 6\ 7)$

<i>A</i>	<i>B</i>	<i>C</i>	<i>S</i>	<i>C_o</i>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



DECODER-9

“ BCD-TO-DECIMAL DECODERS (IC 7445)

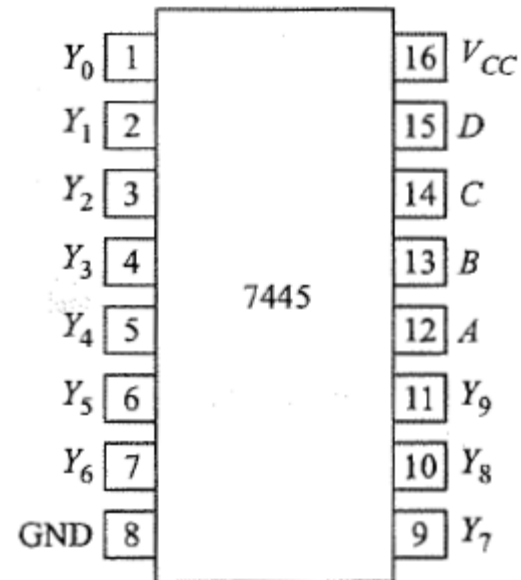
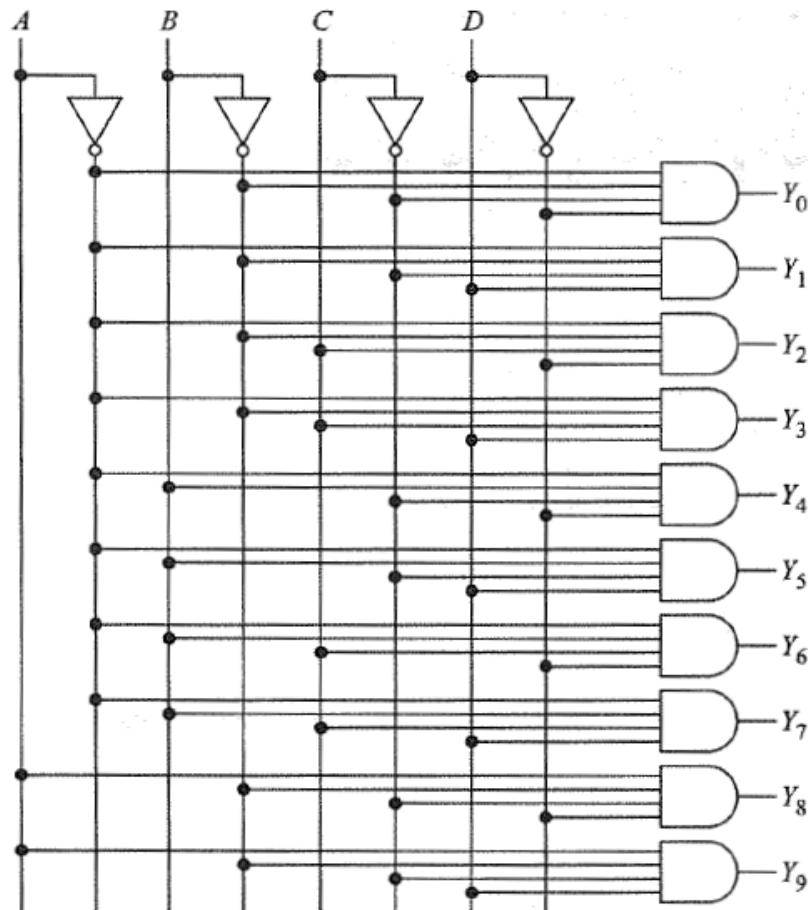


Fig. 4.18

1-of-10 decoder

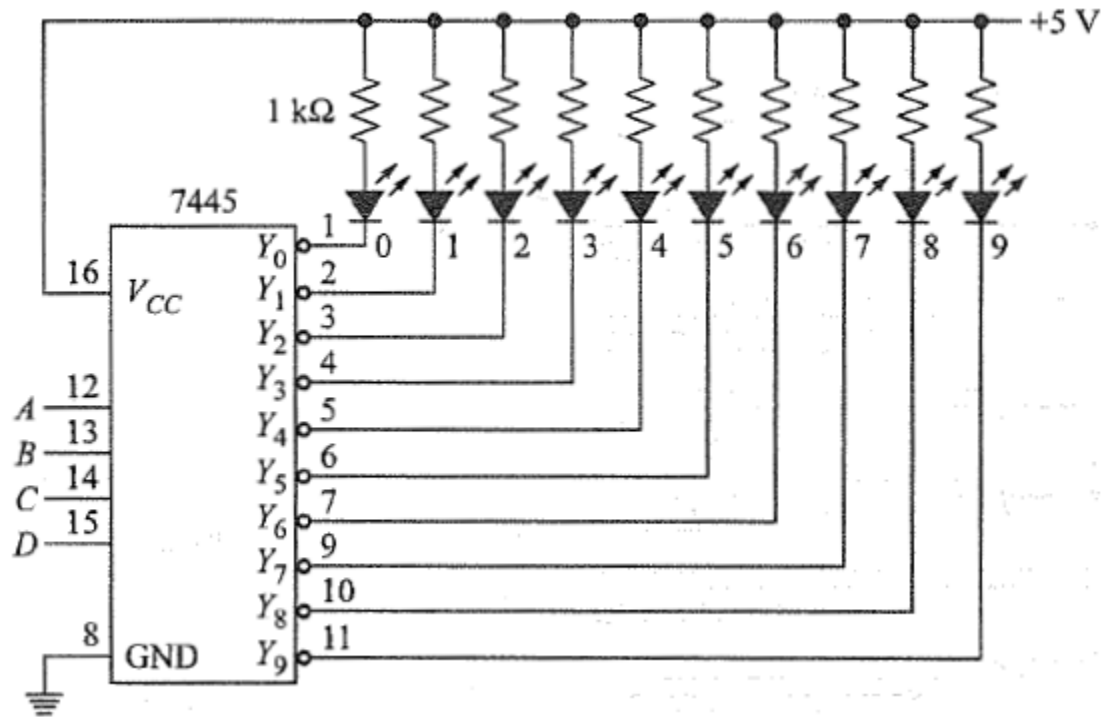
DECODER-10

“ Truth Table of 1 of 10 decoder

<i>Inputs</i>					<i>Outputs</i>									
<i>No.</i>	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>Y₀</i>	<i>Y₁</i>	<i>Y₂</i>	<i>Y₃</i>	<i>Y₄</i>	<i>Y₅</i>	<i>Y₆</i>	<i>Y₇</i>	<i>Y₈</i>	<i>Y₉</i>
0	L	L	L	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	H	H	H	H	H	H	H	L
	H	L	H	L	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	H	H	H	H	H	H	H	H	H	H

DECODER-11

“ Circuit Connection

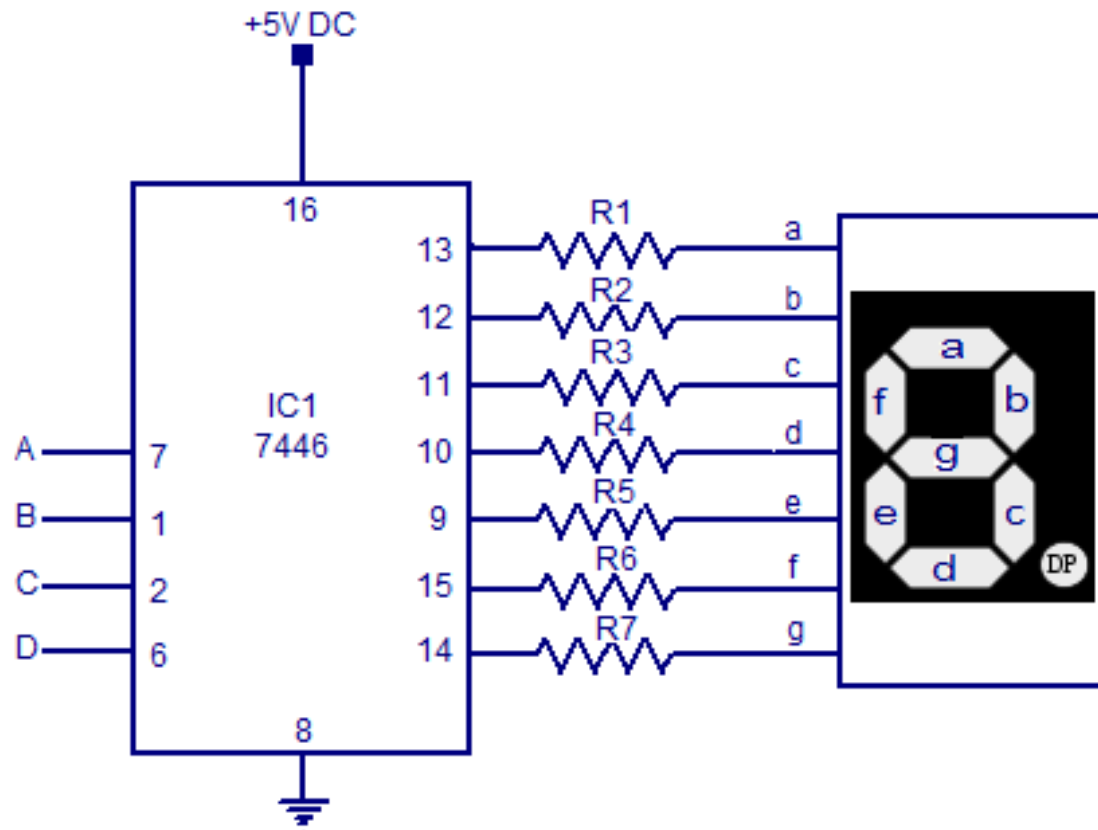


“ Seven Segment decoder



DECODER-13

“ 7446 decoder-driver

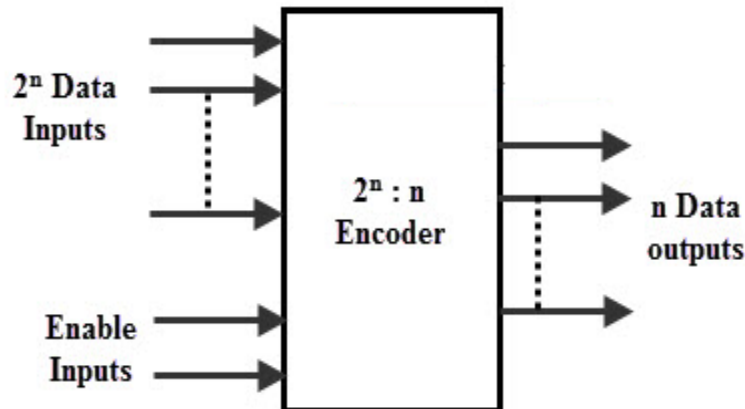


ENCODERS-1

- “ An encoder converts an active input signal into a coded output signal.
- “ An encoder is a device which converts familiar numbers or characters or symbols into a coded format. It accepts the alphabetic characters and decimal numbers as inputs and produces the outputs as a coded representation of the inputs.
- “ It is a combinational circuit that performs the opposite function of a decoder.
- “ These are mainly used to reduce the number of bits needed to represent given information.

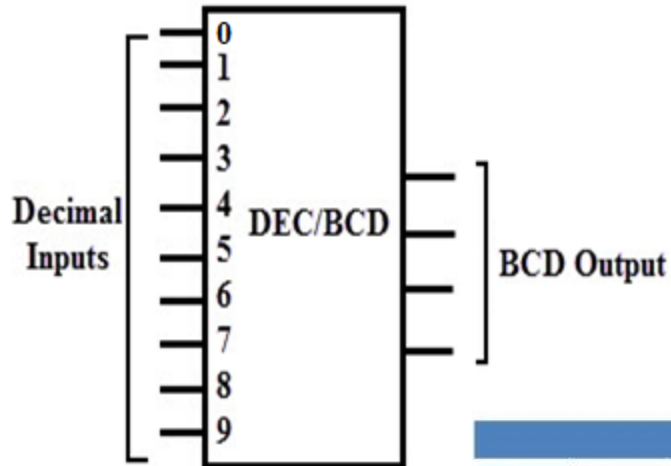
ENCODERS-2

- “ Depending on the number of input lines, digital or binary encoders produce the output codes in the form of 2 or 3 or 4 bit codes.
- “ **An *encoder* is a multiplexer without its single output line.**
- “ It is a combinational logic function that has 2^n (or fewer) input lines and n output lines, which correspond to n selection lines in a multiplexer.
- “ The n output lines generate the binary code for the possible 2^n input lines.



ENCODERS-3

“ Decimal-to-BCD Encoder



Input										Output			
D ₉	D ₈	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

ENCODERS-4

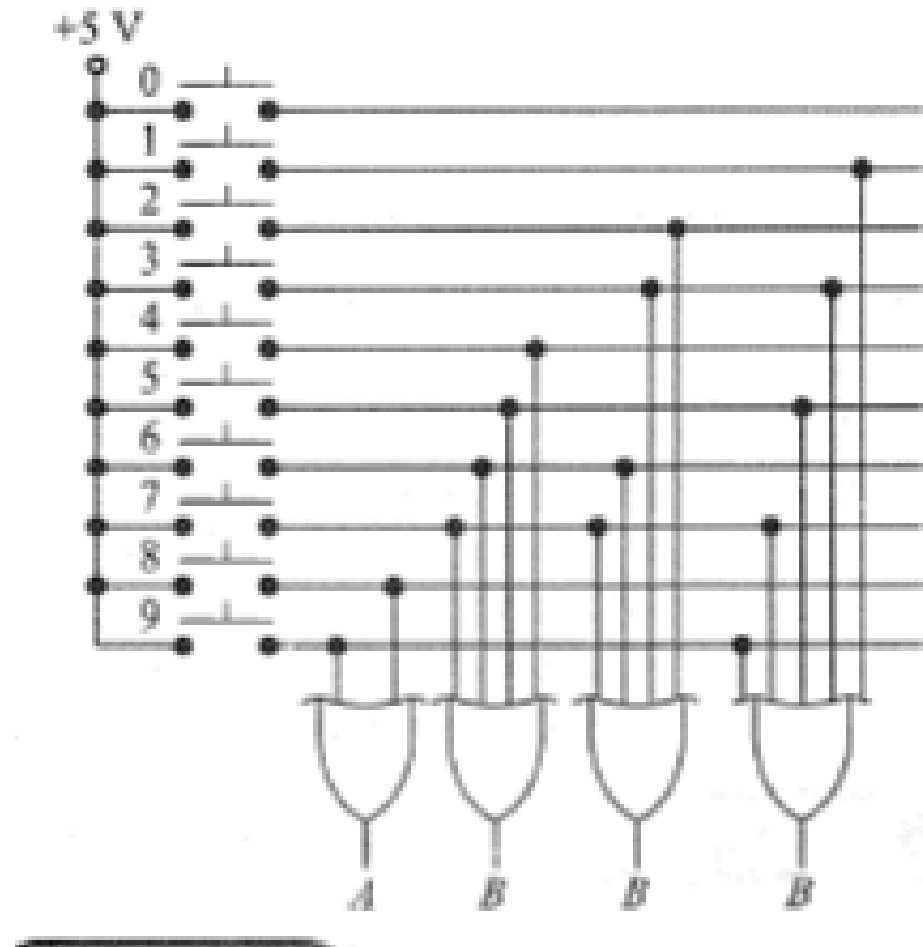
“ Decimal-to-BCD Encoder

“ $A = D_8 + D_9$

“ $B = D_4 + D_5 + D_6 + D_7$

“ $C = D_2 + D_3 + D_6 + D_7$

“ $D = D_1 + D_3 + D_5 + D_7 + D_9$



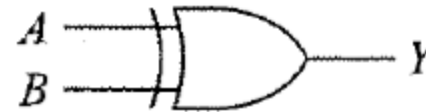
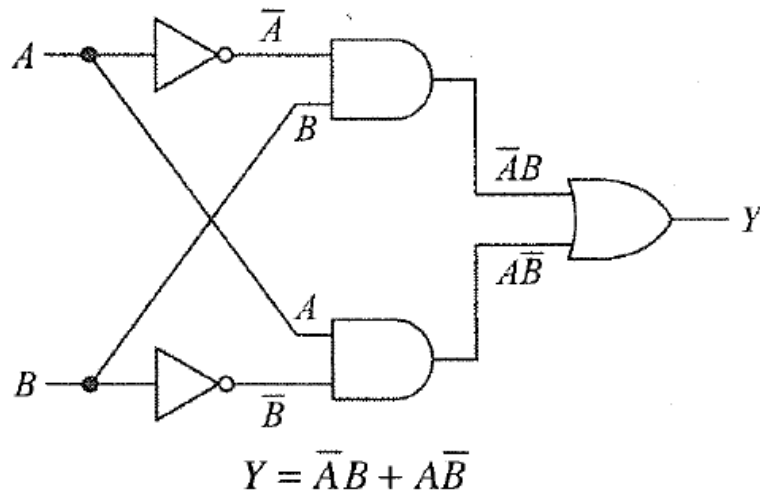
ENCODERS-5

“ Priority Encoder

- “ A *priority encoder* is a practical form of an encoder. The encoders available in IC form are all priority encoders.
- “ In this type of encoder, a priority is assigned to each input so that, when more than one output is simultaneously active, the input with the highest priority is encoded.
- “ Let us assume that the octal-to-binary encoder has an input priority for higher-order digits.
- “ Let us also assume that input lines D2, D4 and D7 are all simultaneously in logic ‘1’ state. In that case,
▶ only D7 will be encoded and the output will be 111.

EXCLUSIVE-OR GATES-1

“ The *exclusive-OR* gate has a high output only when an odd number of inputs is high.

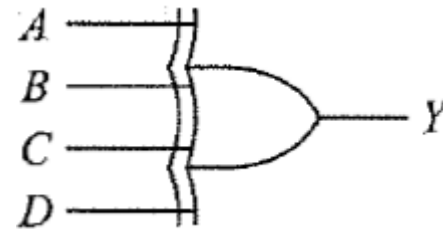
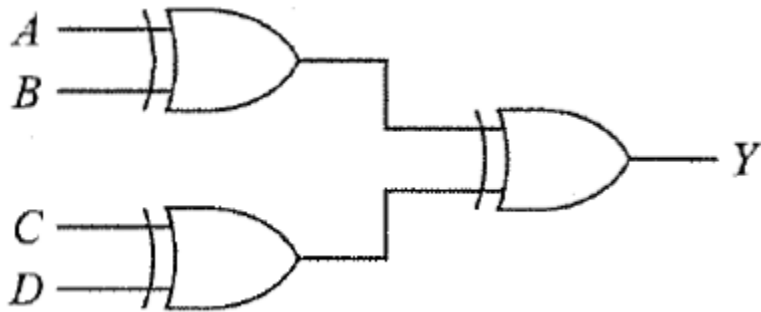


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

“ OR

“ $Y = A \oplus B$

EXCLUSIVE-OR GATES-2



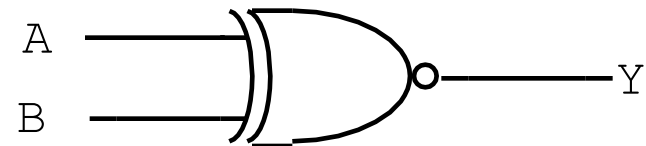
Comment	A	B	C	D	Y
Even	0	0	0	0	0
Odd	0	0	0	1	1
Odd	0	0	1	0	1
Even	0	0	1	1	0
Odd	0	1	0	0	1
Even	0	1	0	1	0
Even	0	1	1	0	0
Odd	0	1	1	1	1
Odd	1	0	0	0	1
Even	1	0	0	1	0
Even	1	0	1	0	0
Odd	1	0	1	1	1
Even	1	1	0	0	0
Odd	1	1	0	1	1
Odd	1	1	1	0	1
Even	1	1	1	1	0

EXCLUSIVE-OR GATES-2

- EX-NOR Gate
- Truth Table

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

- $Y = A \odot B$
- OR
- $Y = AB + \overline{A}\overline{B}$ or $Y = \overline{A \oplus B}$



EXCLUSIVE-OR GATES-2

- **Example**
- $Y = A \oplus B \oplus C$
- $Y = (A \oplus B) \oplus C$
- $Y = (A\bar{B} + \bar{A}B) \oplus C$
- $Y = (A\bar{B} + \bar{A}B)\bar{C} + \overline{(A\bar{B} + \bar{A}B)}C$
- $Y = (A\bar{B} + \bar{A}B)\bar{C} + (AB + \bar{A}\bar{B})C$
- We can write
- $Y = (A \oplus B)\bar{C} + \overline{(A \oplus B)}C$

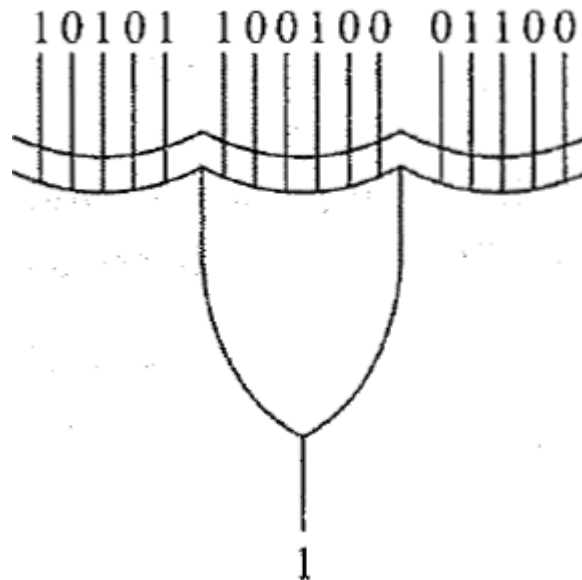
EXCLUSIVE-OR GATES-3

“ **PARITY GENERATORS AND CHECKERS**

- “ *Even parity* means an n-bit input has an even number of 1s. For instance, 110011 has even parity because it contains four 1s.
- “ *Odd parity* means an n-bit input has an odd number of 1s. For example, 110001 has odd parity because it contains three 1s.

EXCLUSIVE-OR GATES-4

“ Parity Checker



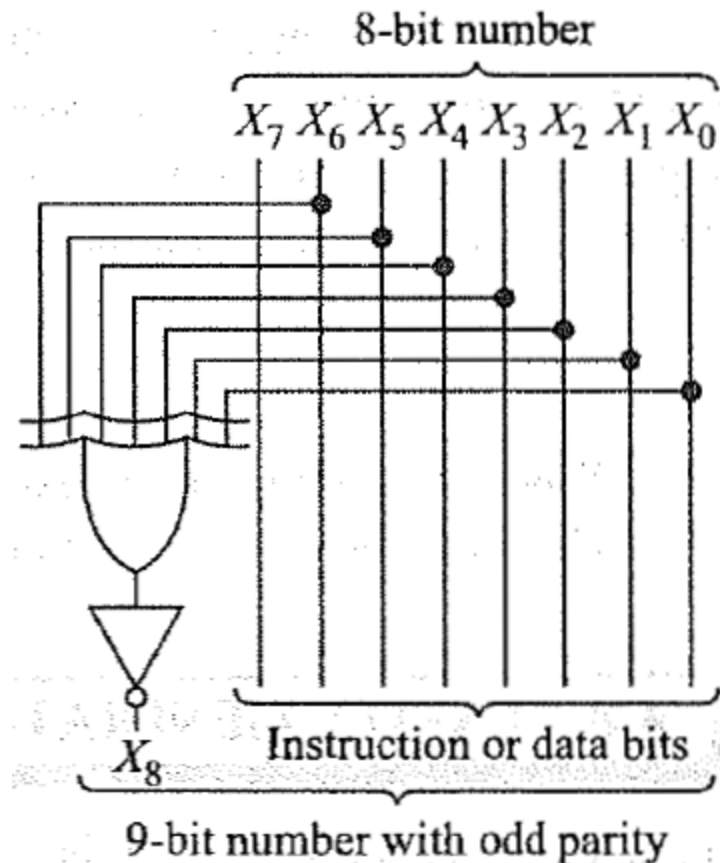
EXCLUSIVE-OR GATES-5

“ Parity Generation

- “ In a computer, a binary number may represent an instruction that tells the computer to add, subtract, and so on; or the binary number may represent data to be processed like a number, letter, etc. In either case, you sometimes will see an extra bit added to the original binary number to produce a new binary number with even or odd parity.

EXCLUSIVE-OR GATES-6

- “ 9th -bit Shows Odd or Even parity
- “ If $X_8=0$ Indicate odd parity
- “ $X_8=1$ even parity



EXCLUSIVE-OR GATES-7

“ Application

- “ Practical application of parity generation and checking
- “ Because of transients, noise, and other disturbances, 1-bit errors sometimes occur when binary data is transmitted over telephone lines or other communication paths.
- “ One way to check for errors is to use an odd-parity generator at the transmitting end and an odd-parity checker at the receiving end. If no 1-bit errors occur in transmission, the received data will have odd parity.
- “ But if one of the transmitted bits is changed by noise or any other disturbance, the received data will have even parity.

EXCLUSIVE-OR GATES-8

“ Lab Experiment

- “ Design and verify the Truth Table of 3-bit Parity Generator and 4-bit Parity Checker using basic Logic Gates with an even parity bit.

“ Parity Generator

Input			Output			
A	B	C	A	B	C	P _G
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	1	0	0	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	1	1	1	1

A \ BC	00	01	11	10
0	0	①	0	①
1	①	0	①	0

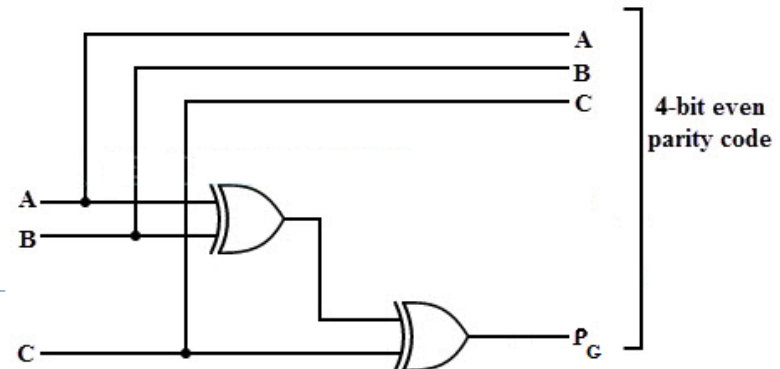
A = A

B = B

C = C

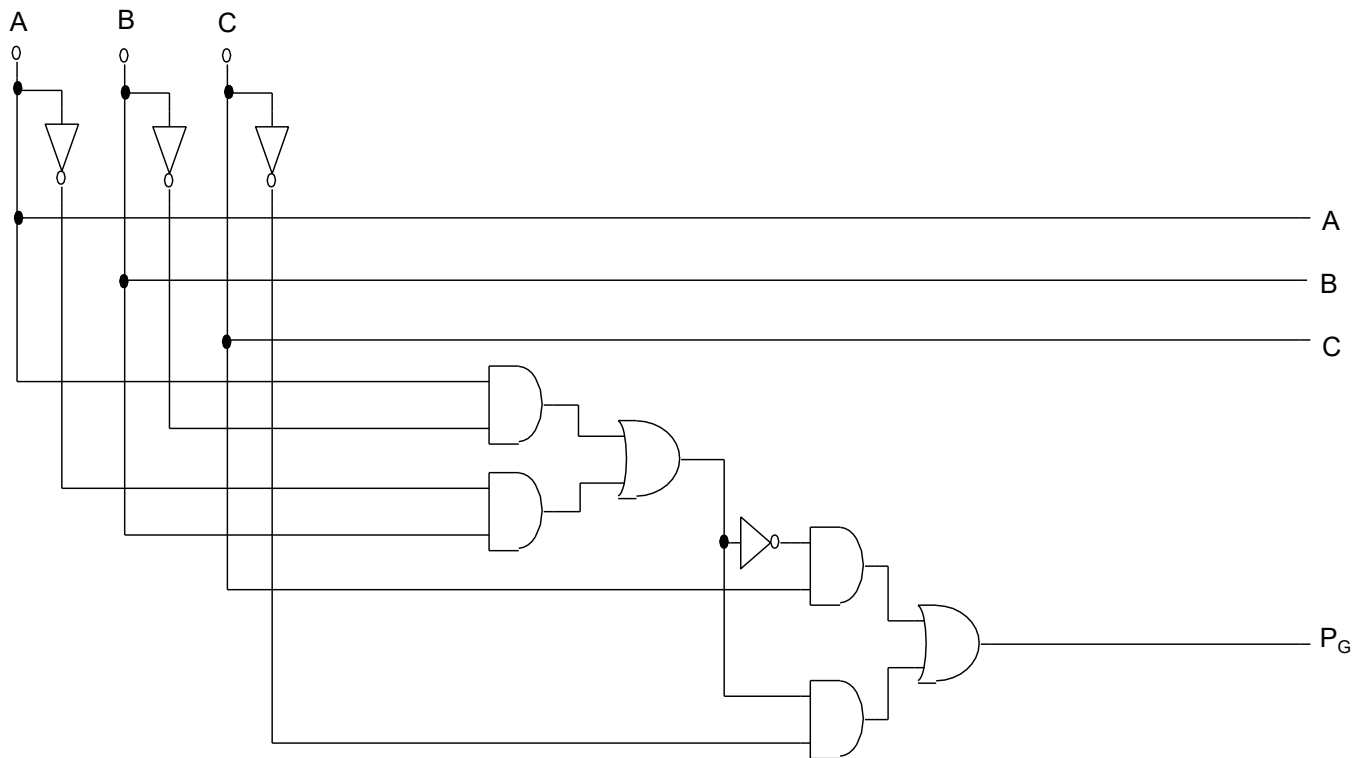
$$P_G = A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$$

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



EXCLUSIVE-OR GATES-8

“ Implementation using Basic Gates



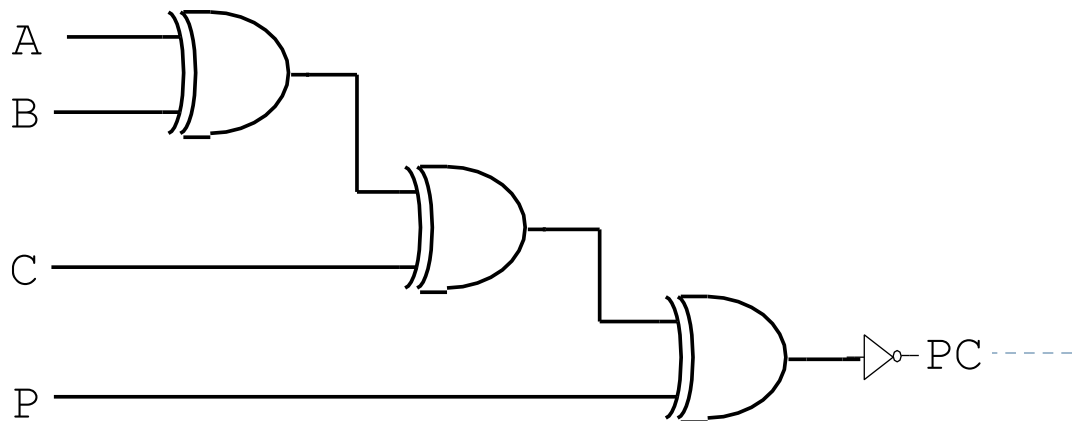
EXCLUSIVE-OR GATES-9

“ Parity Checker

A	B	C	P	P _C
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

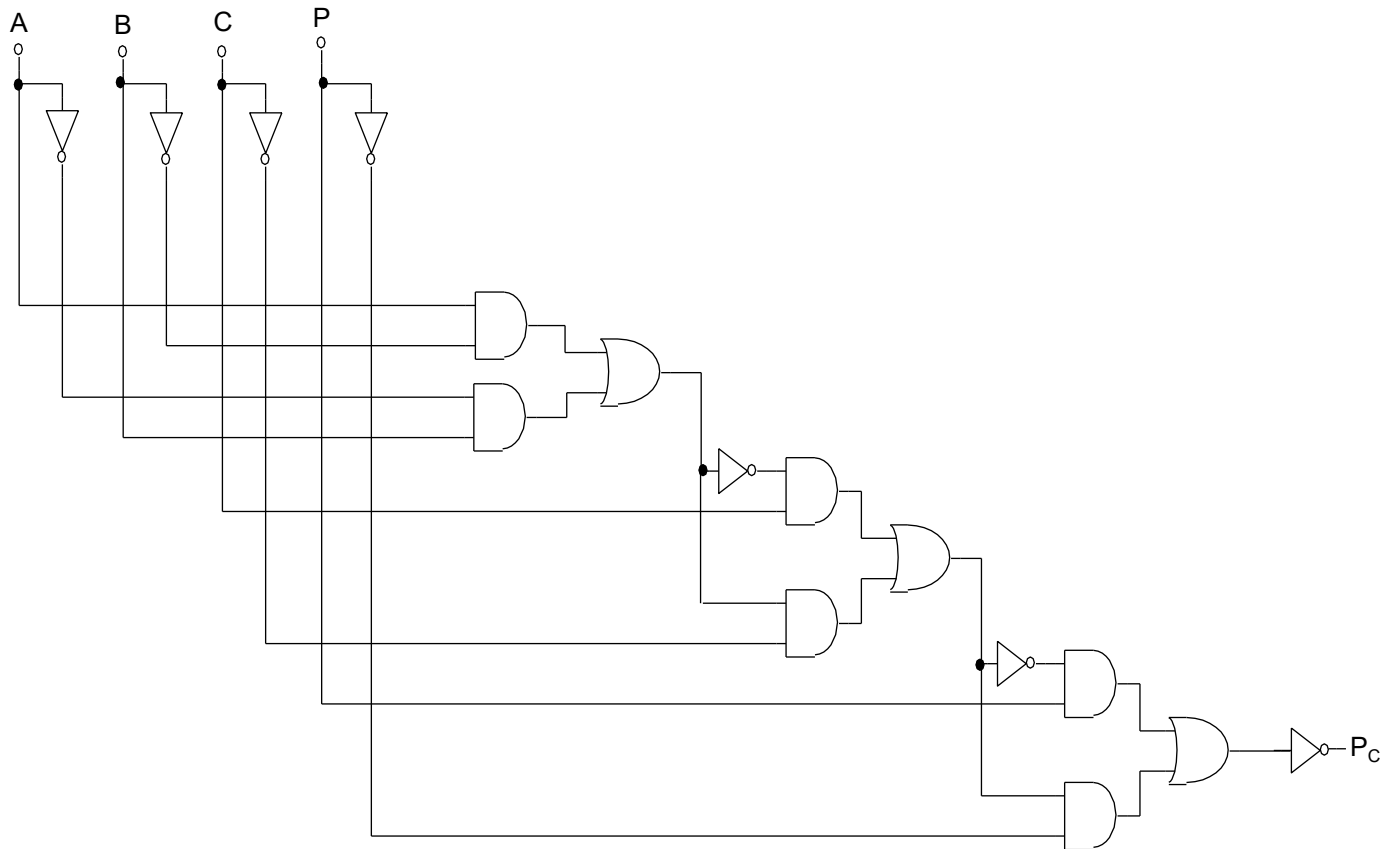
AB \ CP				
	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

$$P_C = \overline{[(A \oplus B) \oplus C] \oplus P}$$



EXCLUSIVE-OR GATES-9

“ Implementation using Basic Gates

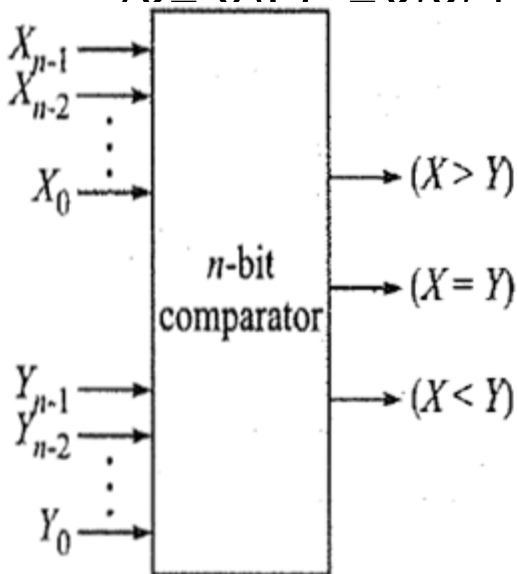


MAGNITUDE COMPARATOR-1

- “ Magnitude comparator compares magnitude two *n-bit* binary numbers.
- “ *X* and *Y* two 1-bit Input, then three outputs $X = Y$, $X > Y$ and $X < Y$.
- “ The logic equations for the outputs can be written as follows, where *G*, *L*, *E* stand for greater than, less than and equal to respectively.

$$(X > Y): G = XY'$$

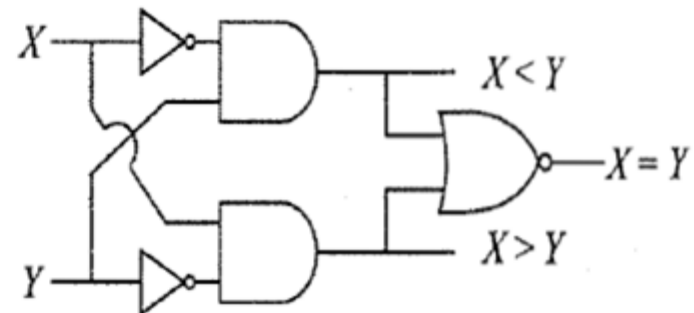
$$(X < Y): L = X'Y$$



(a)

Input		Output		
<i>X</i>	<i>Y</i>	$X > Y$	$X = Y$	$X < Y$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

(b)



(c)

(a) Block diagram of Magnitude comparator, (b) Truth table, (c) Circuit for 1-bit comparator

MAGNITUDE COMPARATOR-2

“ Design of a 2-bit comparator

- “ 4-variable (X: $X1, X0$ and Y: $Y1, Y0$) truth table and get logic equations through any simplification technique like k-Map.
- “ But this procedure will become very complex when the designing a comparator for 3-bit numbers or more.
- “ So from 1 -bit comparator we can write expression for greater than, less than and equal terms.
- “ bit-wise greater than terms (G): $G1 = X1 Y1'$, $G0 = X0 Y0'$
- “ bit-wise less than term (L): $L1 = X1' Y1$, $L0 = X0' Y0$
- “ bit-wise equality term (E): $E1 = (G1 + L1)'$, $E0 = (G0 + L0)'$

MAGNITUDE COMPARATOR-3

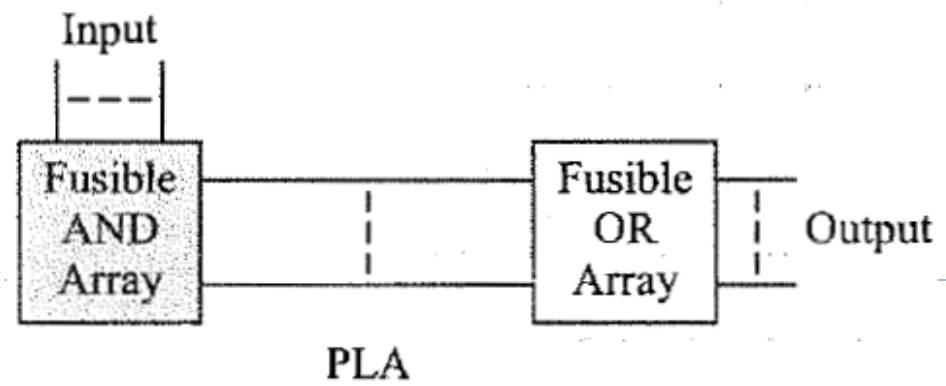
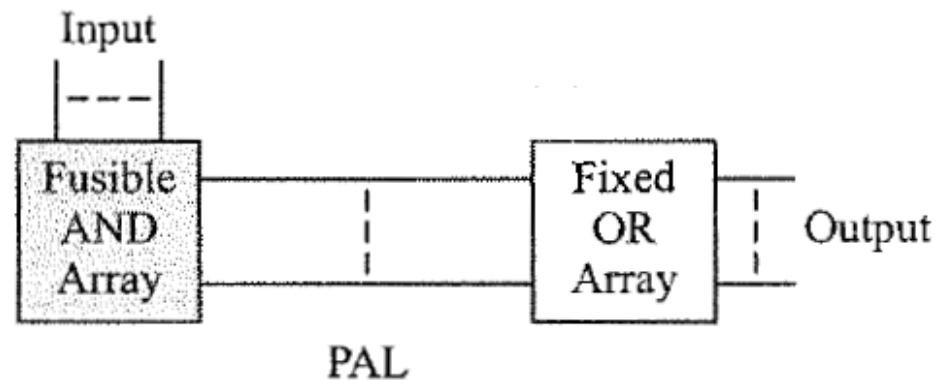
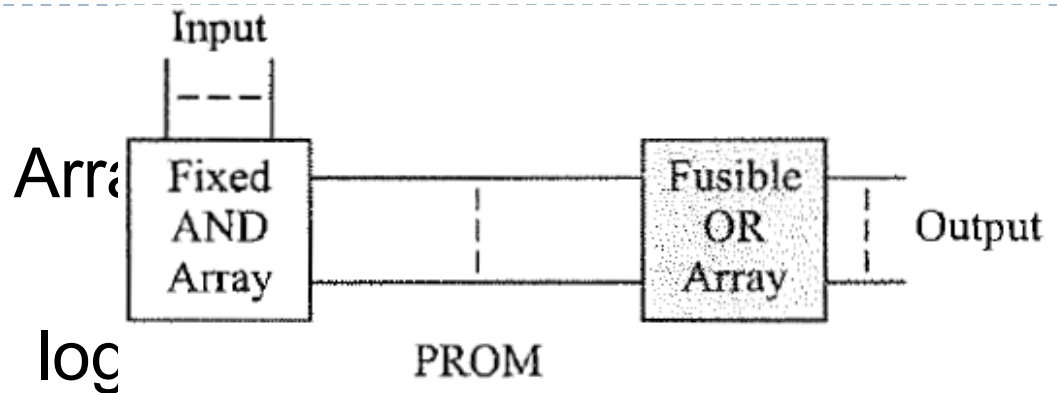
- “ bit-wise greater than terms (G) : $G1 = X1 Y1'$, $G0 = X0Y0'$
- “ bit-wise less than term (L) : $L1 = X1'Y1$, $L0 = X0'Y0$
- “ bit-wise equality term (E): $E1 = (G1 + L1)'$ $E0 = (G0 + L0)'$
- “ If $X = Y$ when both the bits are equal then we can write:
 $(X = Y) = E1.E0$
- “ $X > Y$ if MSB of $X1$ is higher ($G1 = 1$) than that of $Y1$. If MSB is equal, given by $E1 = 1$, then LSB of X and Y is checked and if found higher ($G0 = 1$) the condition $X > Y$ is fulfilled so we can write:
 $(X > Y) = G1 + E1.G0$
- “ Similarly for $X < Y$:
 $(X < Y) = L1 + E1.L0$

MAGNITUDE COMPARATOR-4

- “ X: $X_{n-1} X_{n-2} \dots X_0$ and Y: $Y_{n-1} Y_{n-2} \dots Y_0$
- “ We can write,
- “ $(X = Y) = E_{n-1}.E_{n-2} \dots E_0$
- “ $(X > Y) = G_{n-1} + E_{n-1}.G_{n-2} + \dots + E_{n-1}.E_{n-2} \dots E_1.G_0$
- “ $(X < Y) = L_{n-1} + E_{n-1}.L_{n-2} + \dots + E_{n-1}.E_{n-2} \dots E_1.L_0$
- “ where E_i , G_i and L_i represent for i^{th} bit $X_i = Y_i$, $X_i > Y_i$ and $X_i < Y_i$ terms respectively.

Programmable Logic Devices (PLDs)

- “ PROMs
- “ Programmable Logic (PALs)
- “ Programmable arrays (PLAs)

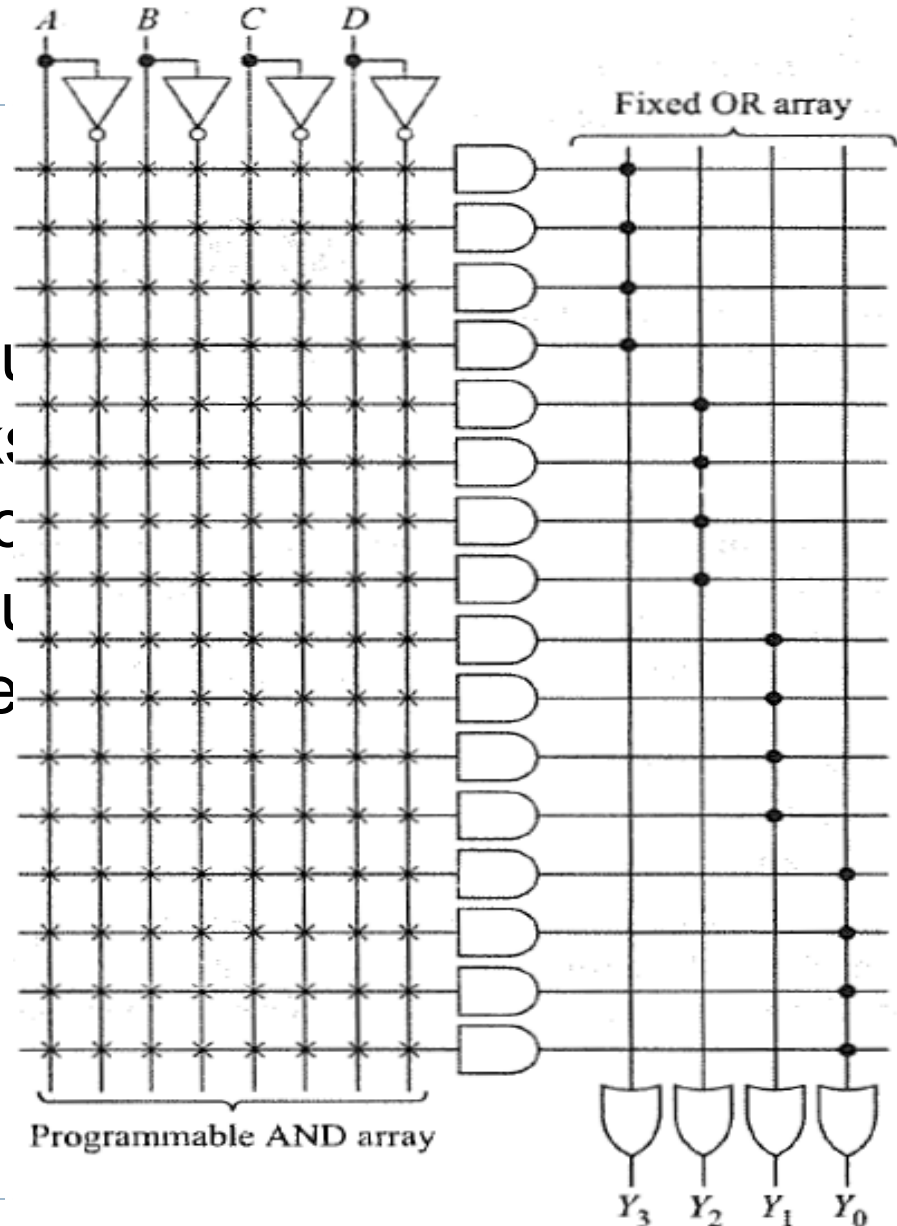


Programmable Array Logic (PAL)-1

- “ *Programmable array logic (PAL)* is a programmable array of logic gates on a single chip. PALs are another design solution, similar to a sum-of-products solution, product-of-sums solution, and multiplexer logic.
- “ A PAL has a programmable AND array and a fixed OR array.

Programmable Array Logic (PAL)-2

- “ PAL with 4 inputs and outputs
- “ In the x's on the input side are fusible links while the solid black bullets on the output side are fixed connections.



Structure of PAL

Programmable Array Logic (PAL)-3

- “ Example of how to program a PAL.

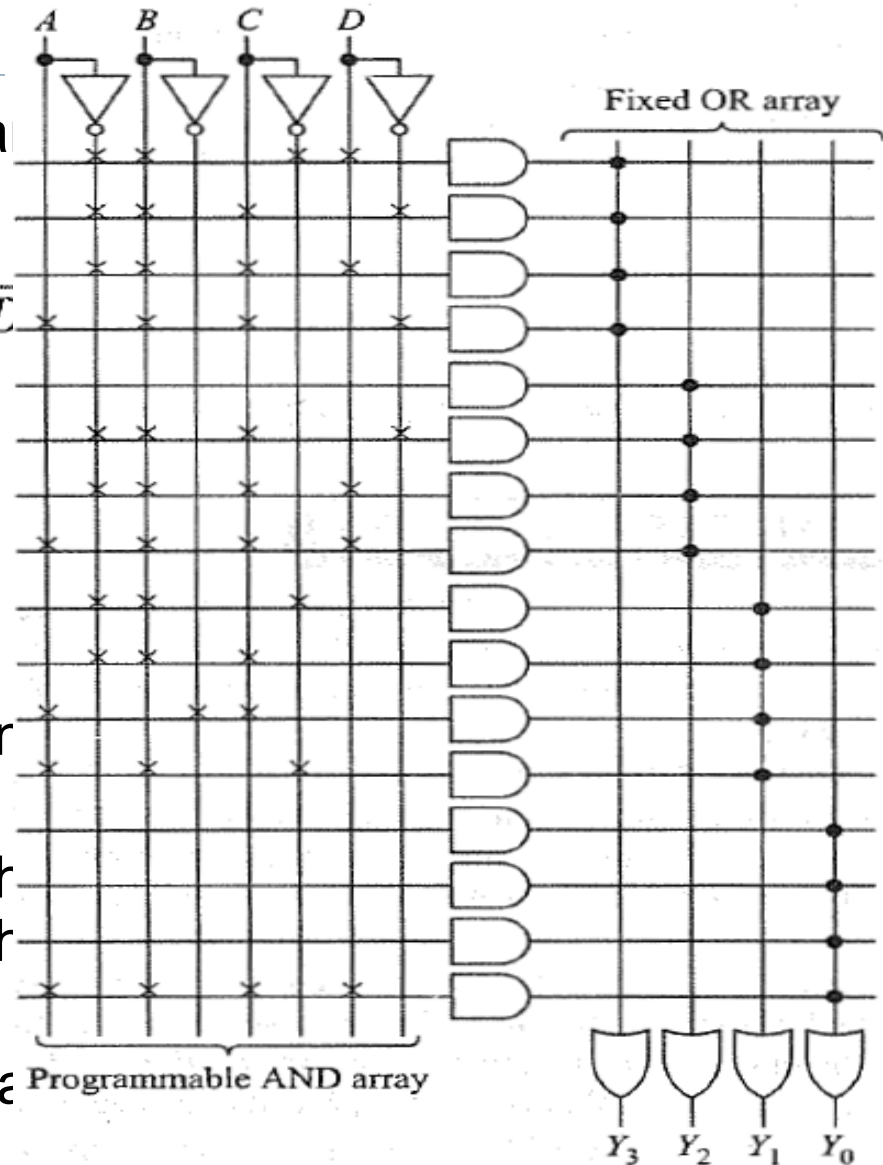
$$Y_3 = \bar{A}\bar{B}\bar{C}D + \bar{A}BC\bar{D} + \bar{A}BCD + ABC\bar{D}$$

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

$$Y_1 = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$$

$$Y_0 = ABCD$$

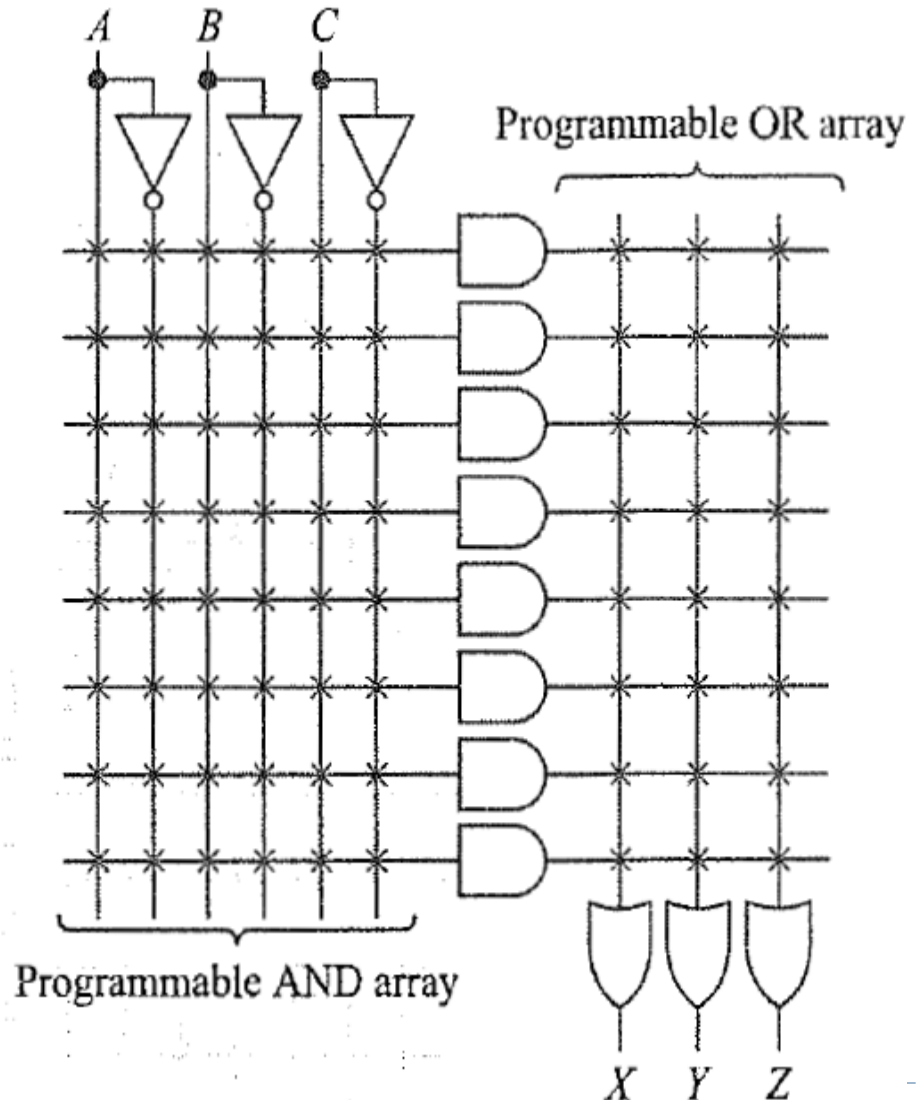
- “ The first desired product $A'BC'D$ On the top input line in figure shown
- “ Remove the first x, the fourth x, the fifth x, and the eighth x.
- “ Then the top AND gate has an output of $A'BC'D$



Example of programming a PAL

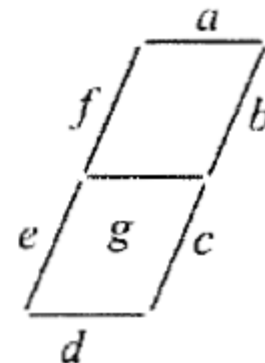
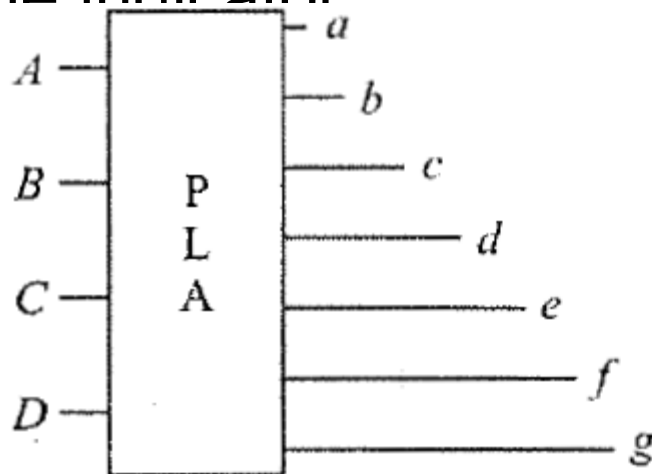
Programmable Logic Arrays (PLAs)-1

- “ Both its AND-gate array and its OR-gate array are fusible-linked and programmable.
- “ A PLA having 3 input variables (ABC) and output variables (XYZ).
- “ There could be additional OR gates to the output if desired.



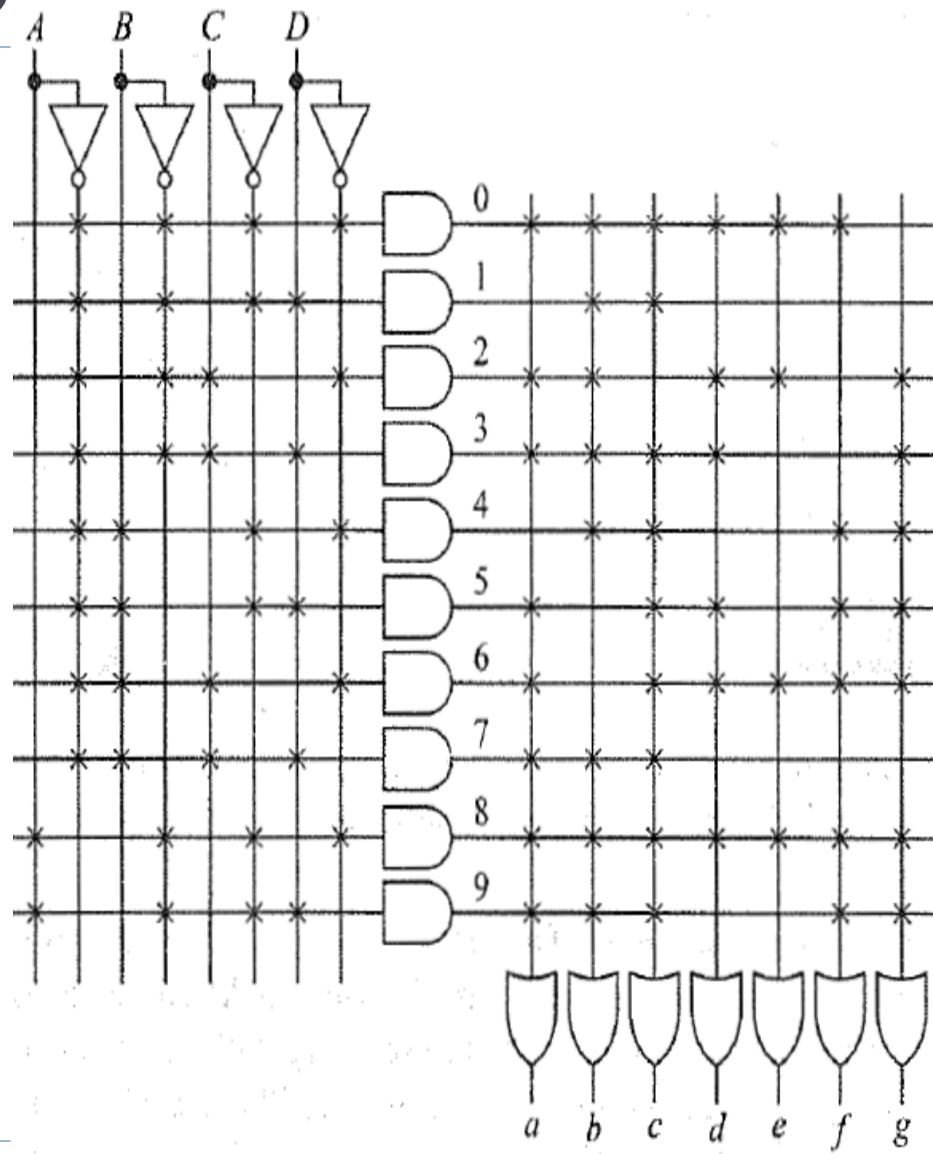
Programmable Logic Arrays (PLAs)-2

- “ An example, use a PLA to recognize each of the 10 decimal digits represented in binary form and to correctly drive a 7-segment display.
- “ Four bits ($ABCD$) are required to represent the 10 decimal numbers. There must be 7 outputs ($abcdefg$), 1 output to drive each of the 7 segments of the indicator



Programmable Logic Arrays (PLA) -3

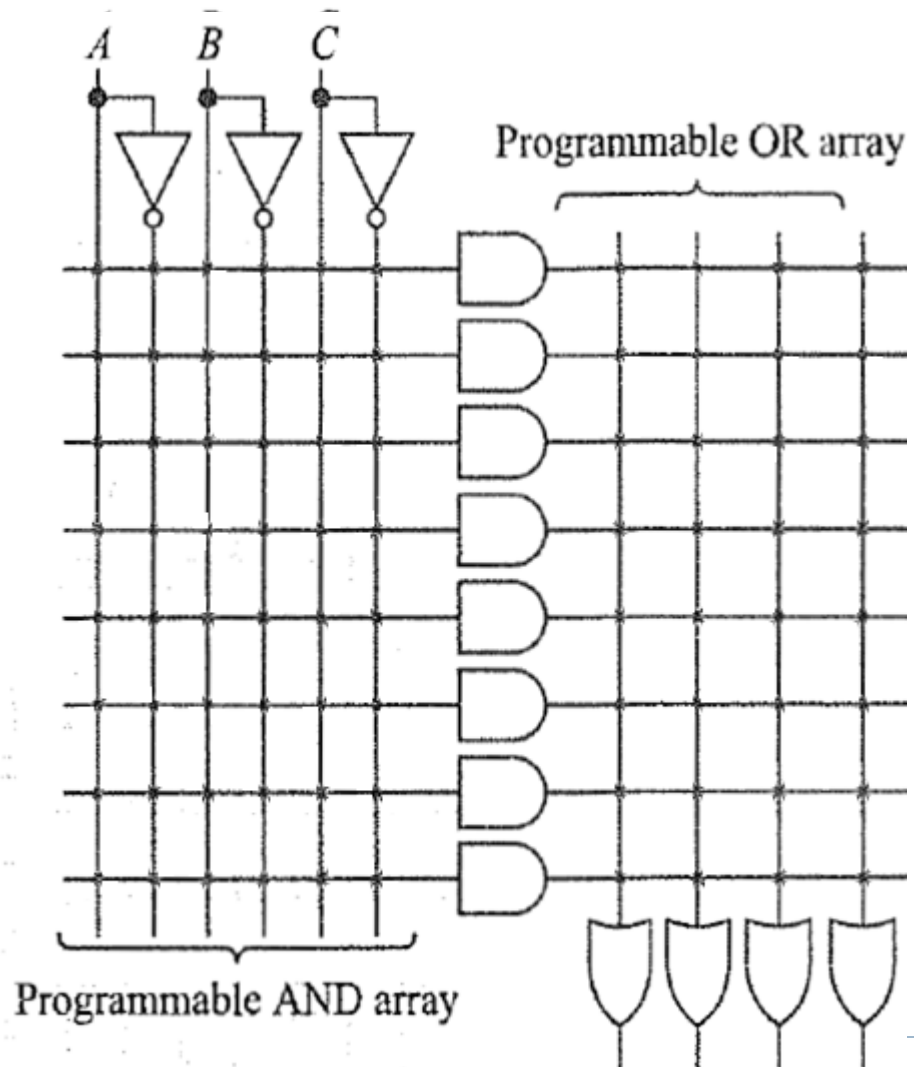
Segments Inputs							7 Segment Display Output
a	b	c	d	e	f	g	
1	1	1	1	1	1	0	0
0	1	1	0	0	0	0	1
1	1	0	1	1	0	1	2
1	1	1	1	0	0	1	3
0	1	1	0	0	1	1	4
1	0	1	1	0	1	1	5
1	0	1	1	1	1	1	6
1	1	1	0	0	0	0	7
1	1	1	1	1	1	1	8
1	1	1	1	0	0	1	9



7-segment decoder using PLA

Programmable Logic Arrays (PLAs)-4

- “ Implement the following Boolean function using suitable PLA $f_1 = \sum(0,1,4,6)$, $f_2 = \sum(2,3,4,6,7)$, $f_3 = \sum(0,1,2,6)$, $f_4 = \sum(2,3,5,6,7)$
- “ Draw a PLA circuit simultaneously realize the Boolean function $Y_3 = A'BC'$; $Y_2 = AC$; $Y_1 = AB'C + A'BC + ABC'$; $Y_0 = A'BC' + A'BC + A'B'C' + ABC$.
- “ Draw a PLA circuit simultaneously realize the Boolean function $X = A'B'C + AB'C' + B'C$; $Y = A'B'C + AB'C'$; $Z = B'C$.
- “ Design 7-segment decoder using PLA.



HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS-1

- “ The data flow model provides a different use of keyword **assign** in the form of
assign $X = S ? A : B;$
- “ If, $S = 1$, $X = A$ and if $S = 0$, $X = B$.

HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS-2

Realize a 2 to 1 multiplexer using assing statement

```
module  
mux2to1(A,D0,D1,Y);  
input A,D0,D1;  
output Y;  
assign Y=(~A&D0) | (A&D1);  
endmodule
```

```
module  
mux2to1(A,D0,D1,Y);  
input A,D0,D1;  
output Y;  
assign Y= A? D1:D0;  
/*Conditional  
assignment*/  
endmodule
```

HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS-3

2 to 1 MUX using the behavioural model using if ... else statement and using case statement.

```
module
mux2to1(A,D0,D1,Y);
input A,D0,D1;
output Y;
reg Y;
always@ (A or D0 or
D1)
if (A==1) Y=D1;
else Y=D0;
endmodule
```

```
module
mux2to1(A,D0,D1,Y);
input A,D0,D1;
output Y;
reg Y;
always@ (A or D0 or D1)
case (A)
    0 : Y=D0;
    1 : Y=D1;
endcase
endmodule
```

HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS-4

- “ Design a 4 to 1 multiplexer, using conditional ‘assign’ and ‘case’ statements.
- “ We can use nested condition for assign statement.
- “ If $A=1$, condition ($B ? D3:D2$) is evaluated.
- “ Then if $B = 1$, $Y=D3$.
- “ Similarly, the other combinations of A and B are evaluated and Y is assigned a value from $D2$ to $D1$.
- “ For case statement concatenate A and B by using operator $\{...\}$ and generated four possible combinations, For a particular value of AB .

HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS-5

Using assign

```
module mux4to1  
(A,B,DO,DI,D2,D3,Y);  
input A,B,DO,DI,D2,D3  
output Y;  
assign Y =  
A?(B?D3:D2):(B?D1:D0);  
endmodule
```

Using case

```
module mux4to1(A,B,DO,DI,D2,D3,Y);  
input A,B,DO,DI,D2,D3;  
output Y;  
reg Y;  
always @(A or B or DO or DI or D2 or  
D3)  
case ({A,B})  
0: Y=D0;  
1: Y=D1;  
2: Y=D2;  
3: Y=D3;  
endcase  
endmodule
```

HDL IMPLEMENTATION OF DATA PROCESSING CIRCUITS 6

- “ A verilog HDL code for a digital circuit is given as follows. Can you describe the function it performs?

```
module unknow (A,B,C,Y);  
input [3:0] A,B;  
input [2:0] C;  
output [2:0] Y;  
reg [2:0] Y;  
always @ (A or B or C)  
if (A<B) Y=3'b001;  
else if (A>B) Y=3'b010;  
else Y=C;  
endmodule
```

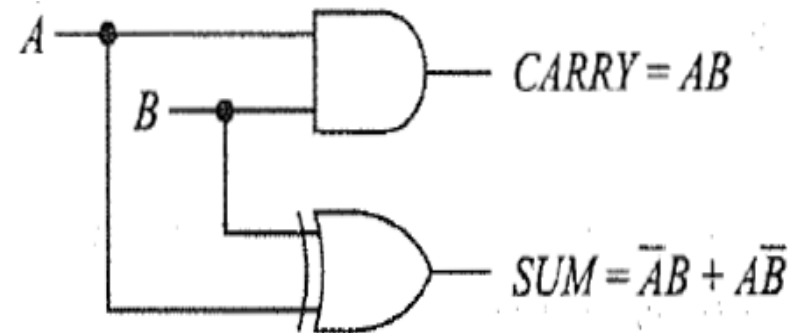
- “ HDL compares two 4-bit numbers A and B and generates a 3 bit output Y.

Arithmetic Building Blocks-1

“ Half-Adder

- “ When we add two binary numbers, we start with the least significant column. This means that we have to add two bits with the possibility of a carry. The circuit used for this is called a *half-adder*.

<i>A</i>	<i>B</i>	<i>CARRY</i>	<i>SUM</i>
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

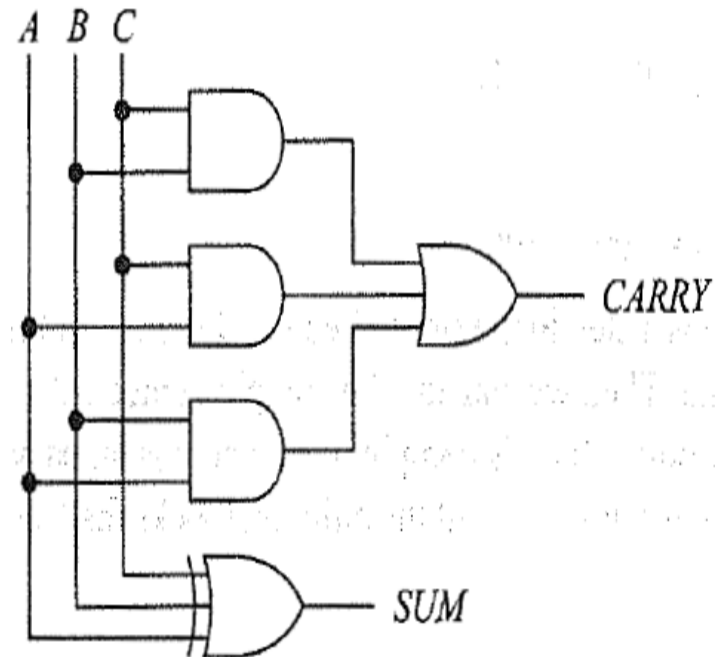


Arithmetic Building Blocks-2

“ Full-Adder

“ A full-adder, a logic circuit that can add 3 bits at

A	B	C	CARRY	SUM
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



C \ A B	00	01	11	10
0	0	0	1	0
1	0	1	0	1

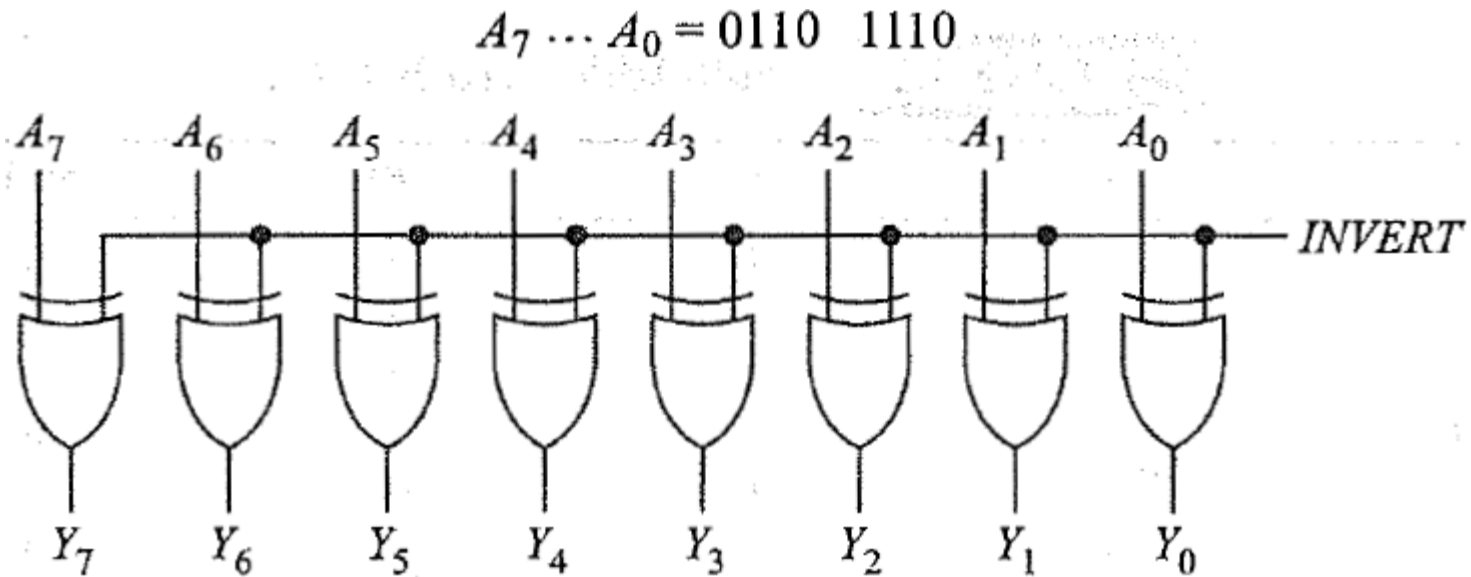
$$\text{Carry} = AB + BC + AC$$

C \ A B	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$\text{SUM} = A \oplus B \oplus C$$

Arithmetic Building Blocks-3

“ Controlled Inverter



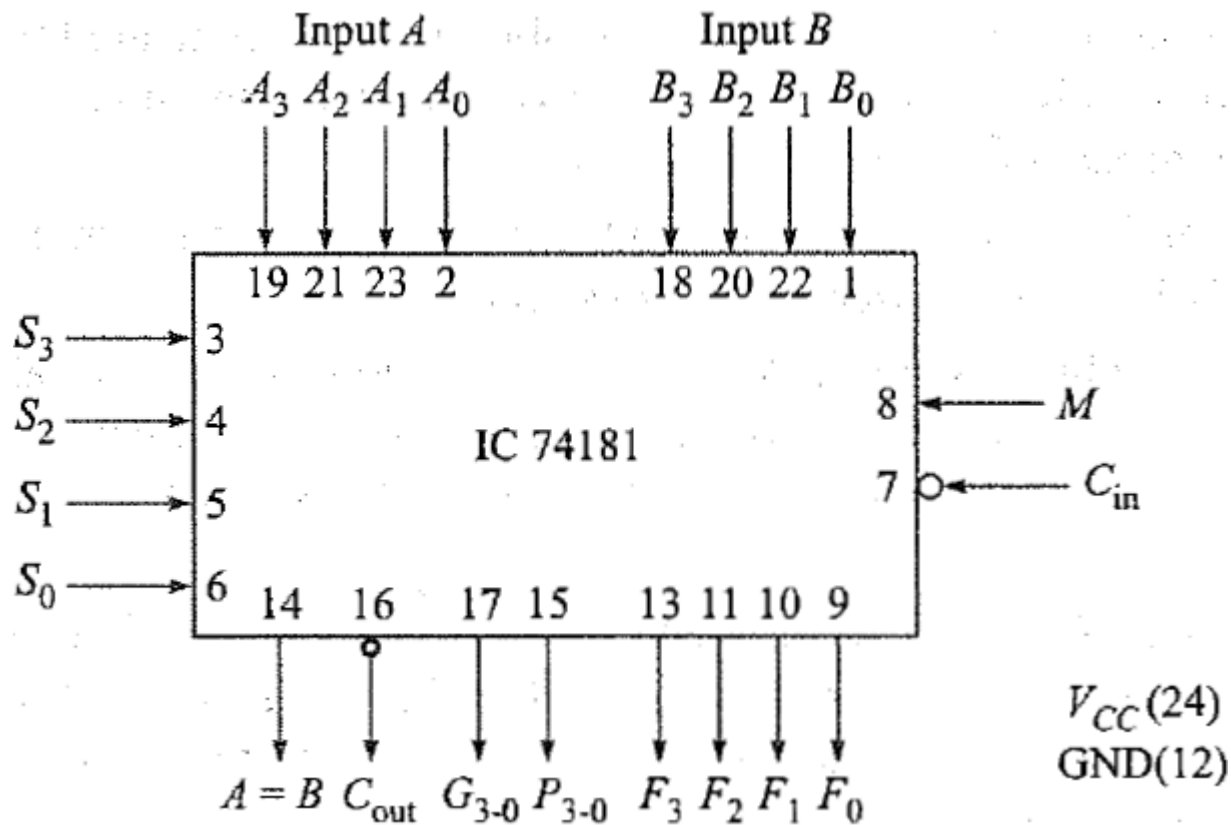
Arithmetic Building Blocks-4

- “ A Low INVERT produces
 - “ $Y_1 \cdots Y_0 = 0110\ 1110$
- “ But a High INVERT results in
 - “ $Y_1 \cdots Y_0 = 1001\ 0001$
- “ During a subtraction, we first need to take the 2's complement of the subtrahend. Then we can add the complemented subtrahend to obtain the answer.
- “ With a controlled inverter, we can produce the 1's complement. There is an easy way to get the 2 's complement.

ARITHMETIC LOGIC UNIT-1

- “ ALU is an integral part of central processing unit or CPU of a computer. It comes in various forms with wide range of functionality.
- “ Like normal addition, subtraction, increment, decrement operations.
- “ As logic unit it performs usual AND, OR, NOT, EX-OR and many other complex logic functions.
- “ It also comes with PRESET and CLEAR options, invoking which all the function outputs are made 1 and 0 respectively.
- “ Normally, a mode selector input (M) decides whether ALU performs a logic operation or an arithmetic operation.

ARITHMETIC LOGIC UNIT-2



ARITHMETIC LOGIC UNIT-3

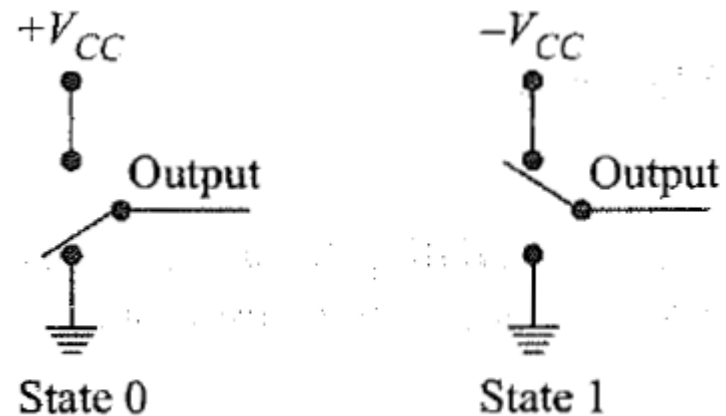
$S_3 S_2 S_1 S_0$	$M = 1$ (Logic Function)	$M = 0$ (Arithmetic Function) $C_{in} = 1$ (For $C_{in} = 0$, add 1 to F)
0 0 0 0	$F = A'$	$F = A$
0 0 0 1	$F = (A + B)'$	$F = A + B$
0 0 1 0	$F = A'B$	$F = A + B'$
0 0 1 1	$F = 0$	$F = \text{minus } 1$
0 1 0 0	$F = (AB)'$	$F = A \text{ plus } (AB)'$
0 1 0 1	$F = B'$	$F = (A + B) \text{ plus } (AB)'$
0 1 1 0	$F = A \oplus B$	$F = A \text{ minus } B \text{ minus } 1$
0 1 1 1	$F = AB'$	$F = AB' \text{ minus } 1$
1 0 0 0	$F = A' + B$	$F = A \text{ plus } (AB)$
1 0 0 1	$F = (A \oplus B)'$	$F = A \text{ plus } B$
1 0 1 0	$F = B$	$F = (A + B') \text{ plus } (AB)$
1 0 1 1	$F = AB$	$F = AB \text{ minus } 1$
1 1 0 0	$F = 1$	$F = A \text{ plus } A$
1 1 0 1	$F = A + B'$	$F = (A + B) \text{ plus } A$
1 1 1 0	$F = A + B$	$F = (A + B') \text{ plus } A$
1 1 1 1	$F = A$	$F = A \text{ minus } 1$

Introduction to Flip Flop

- “ The outputs of the digital circuits are dependent entirely on their inputs.
- “ However, there are requirements for a digital device or circuit whose output will remain unchanged, once set, even if there is a change in input level(s). Such a device could be used to store a binary number called Flip-Flop.
- “ Flip-flops are used in the construction of registers and counters, and in numerous other applications.
- “ In a sequential logic circuit flip-flops serve as key memory elements.
- “ Analysis of such circuits are done through truth tables or characteristic equations of flip-flops. The analysis result is normally presented through state table or state transition diagram and also through timing diagram.

RS FLIP-FLOPS-1

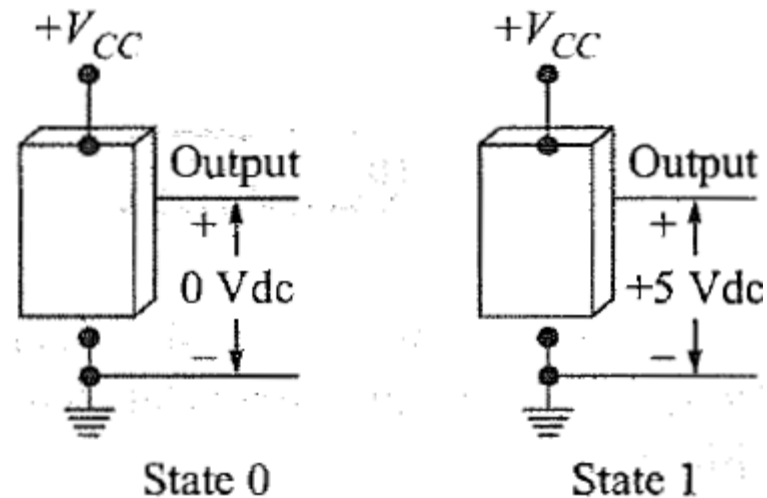
- “ Any device or circuit that has two stable states is said to be *bistable*.
- “ For instance, a toggle switch has two stable states.
- “ It is either up or down, depending on the position of the switch.
- “ The switch is also said to have *memory* since it will remain as set until someone changes its position.



(a) Toggle switch

RS FLIP-FLOPS-2

- “ A *flip-flop* is a bistable electronic circuit that has two stable states—that is, its output is either 0 or +5V.
- “ The flip-flop also has memory since its output will remain as set until something is done to change it.
- “ In fact, any bistable device can be used to store one binary digit (bit).
- “ For instance, when the flip-flop has its output set at 0V dc, it storing a logic 0 and when its output is set at + 5 V dc, as storing a logic 1.
- “ The flip-flop is often called a *latch*, since it will hold, or latch, in either stable state.

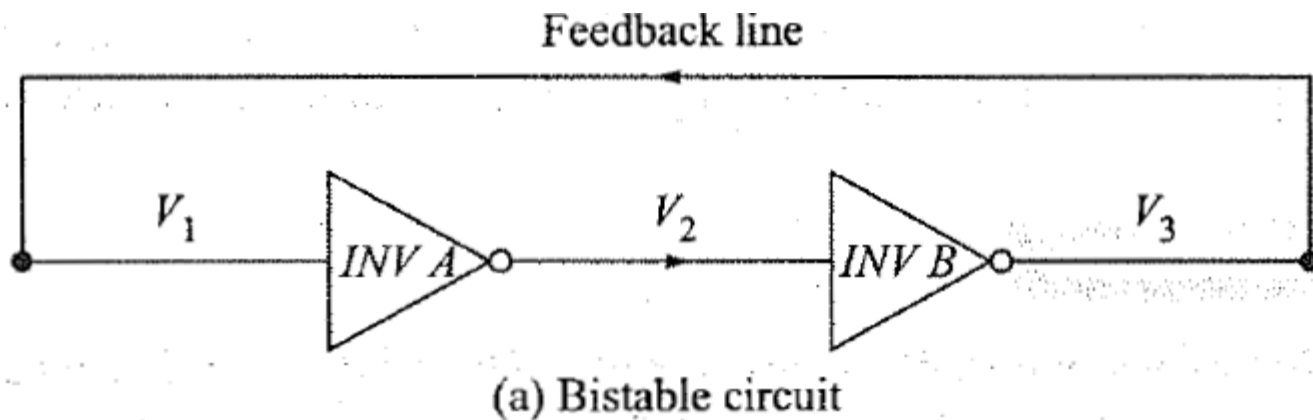


(b) Flip-flop

RS FLIP-FLOPS-3

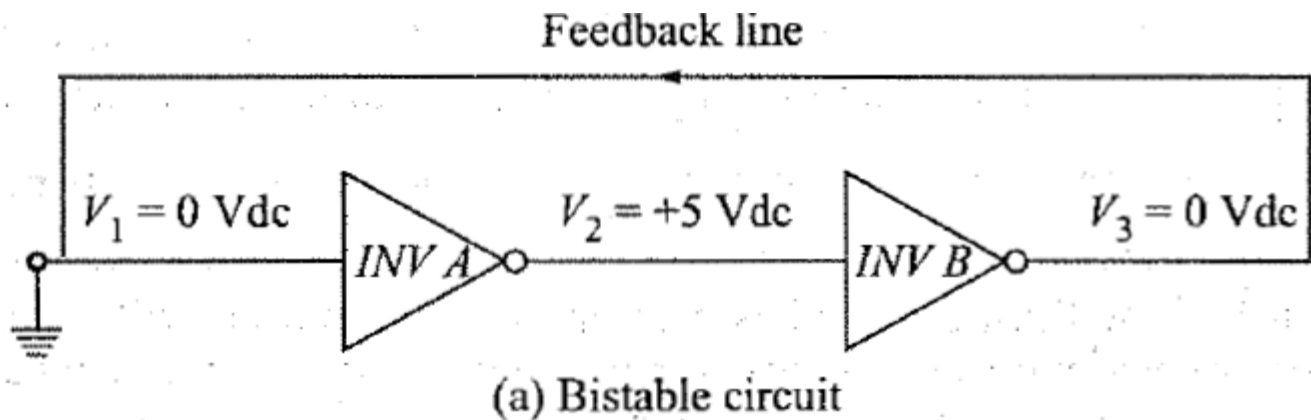
“ Basic Idea

- “ One of the easiest ways to construct a flip-flop is to connect two inverters in series as shown in Figure. The line connecting the output of inverter *B* (*INV B*) back to the input of inverter *A* (*INV A*) is referred to as the *feedback line*.



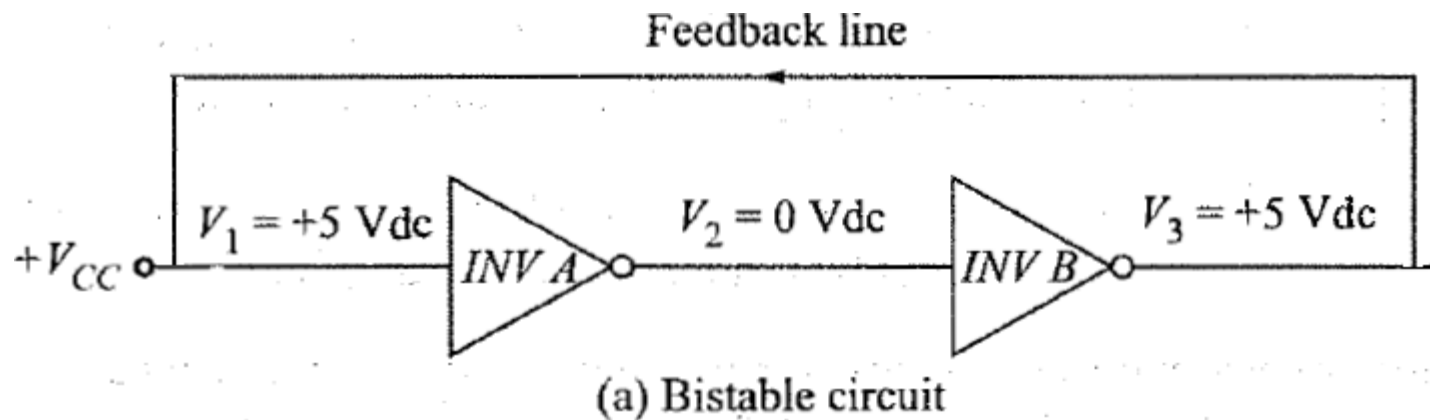
RS FLIP-FLOPS-4

- If V_1 is 0 Vdc, V_3 will also be 0 Vdc as seen in Figure.
- The feedback line can again be used to hold V_1 at 0V dc since V_3 is also at 0V dc. This is then the second stable state $V_3 = 0$ V dc.



RS FLIP-FLOPS-3

- If V_1 is +5 Vdc, V_3 will also be +5 Vdc as seen in Figure.
- The feedback line can again be used to hold V_1 at + 5 V dc since V_3 is also at + 5 V dc. This is then the second stable state- $V_3 = + 5$ V dc.



RS FLIP-FLOPS-4

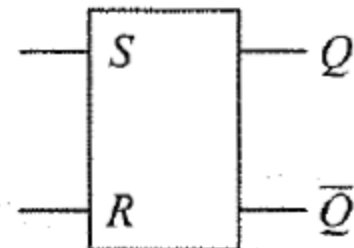
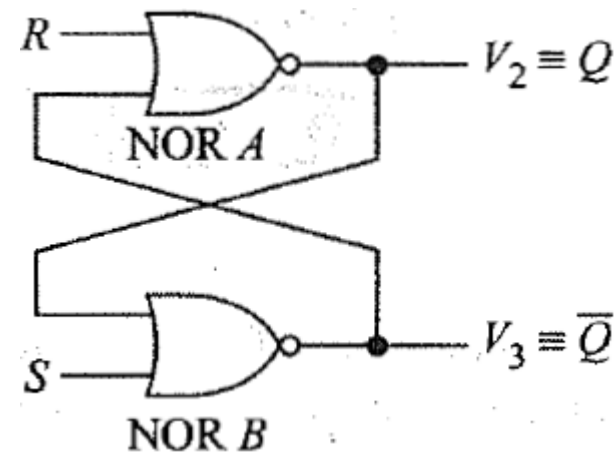
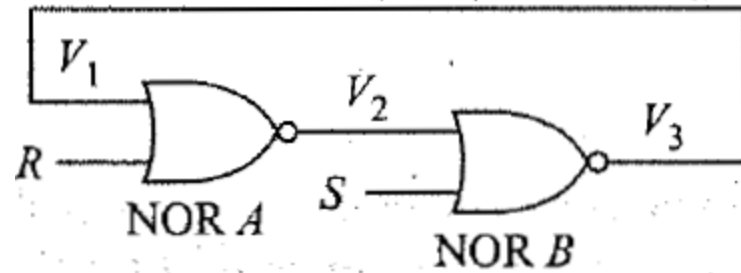
“ **NOR-Gate latch**

- “ The basic flip-flop can be improved by replacing the inverters with either NAND or NOR gates.
- “ The additional inputs on these gates provide a convenient means for application of input signals to switch the flip-flop from one stable state to the other.

RS FLIP-FLOPS-5

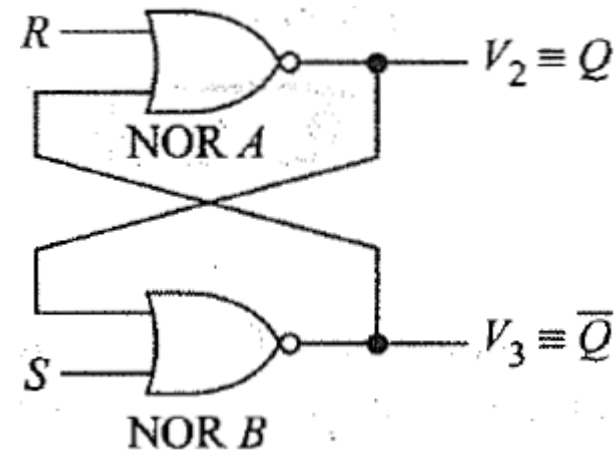
- “ Two inputs labelled R and S .
- “ Two outputs, defined in more general terms as Q and Q' .

R	S	Q	Action
0	0	Last state	No change
0	1	1	SET
1	0	0	RESET
1	1	?	Forbidden



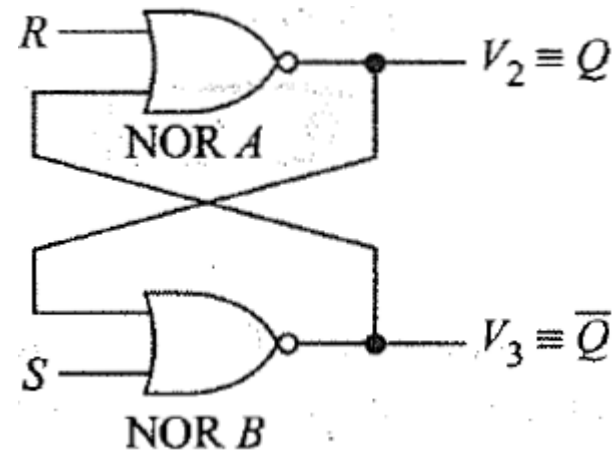
RS FLIP-FLOPS-6

- “ To aid in understanding the operation of this circuit, recall that an $H = 1$ at any input of a NOR gate forces its output to an $L = 0$.
- “ The first input condition in the truth table is $R = 0$ and $S = 0$. Since a 0 at the input of a NOR gate has no effect on its output, the flip-flop simply remains in its present state; that is, Q remains unchanged.



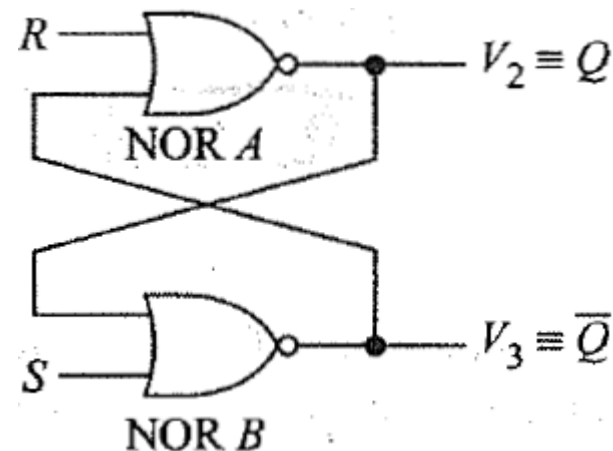
RS FLIP-FLOPS-7

- “ The second input condition $R = 0$ and $S = 1$ forces the output of NOR gate B low.
- “ Both inputs to NOR gate A are now low, and the NOR-gate output must be high.
- “ Thus a 1 at the S input is said to *SET* the flip-flop, and it switches to the stable state where $Q = 1$.



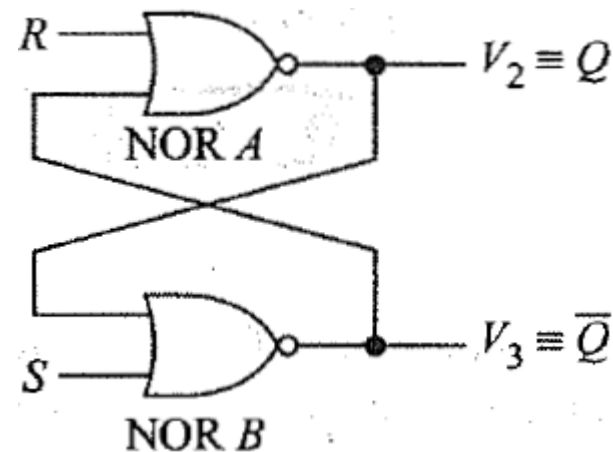
RS FLIP-FLOPS-8

- “ The third input condition is $R = 1$ and $S = 0$. This condition forces the output of NOR gate A low, and since both inputs to NOR gate B are now low, the output must be high.
- “ Thus a 1 at the R input is said to RESET the flip-flop and it switches to the stable state where $Q = 0$ (or $Q' = 1$).



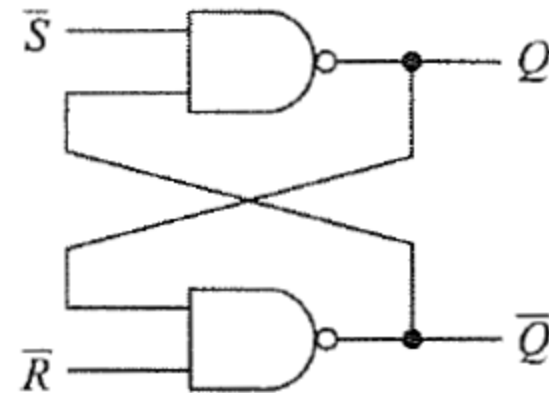
RS FLIP-FLOPS-9

- “ The last input condition in the table, $R = 1$ and $S = 1$, is forbidden, as it forces the outputs of both NOR gates to the low state.
- “ In other words, both $Q = 0$ and $Q' = 0$ at the same time.
- “ This violates the basic definition of a flip-flop that requires Q to be the complement of Q' , and so it is generally agreed never to impose this input condition.

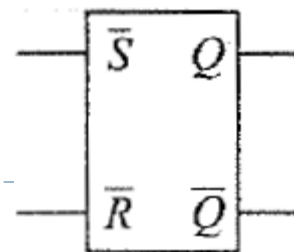


RS FLIP-FLOPS-10

- ▶ **NAND-Gate latch**
- ▶ Also called as $\bar{R}\bar{S}$ flip-flop.
- ▶ A low on any input to a NAND gate will force its output high.
- ▶ Thus a low on the S input will set the latch ($Q = 1$ and $Q' = 0$).
- ▶ A low on the R input will reset it ($Q = 0$).
- ▶ If both R and S are high, the flip-flop will remain in its previous state.
- ▶ R and S low simultaneously is forbidden since this forces both Q and Q' high.

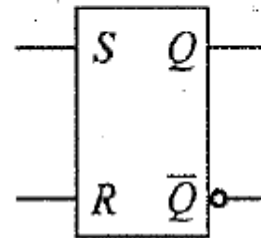
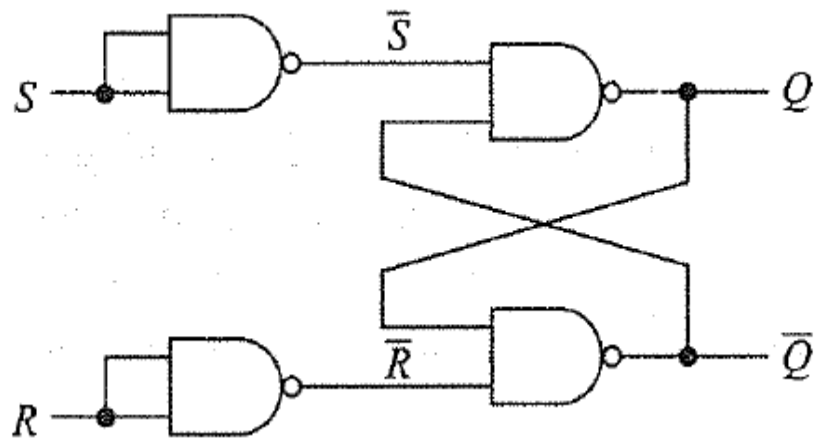


\bar{R}	\bar{S}	Q
1	1	Last state
1	0	1
0	1	0
0	0	? (Forbidden)



RS FLIP-FLOPS-11

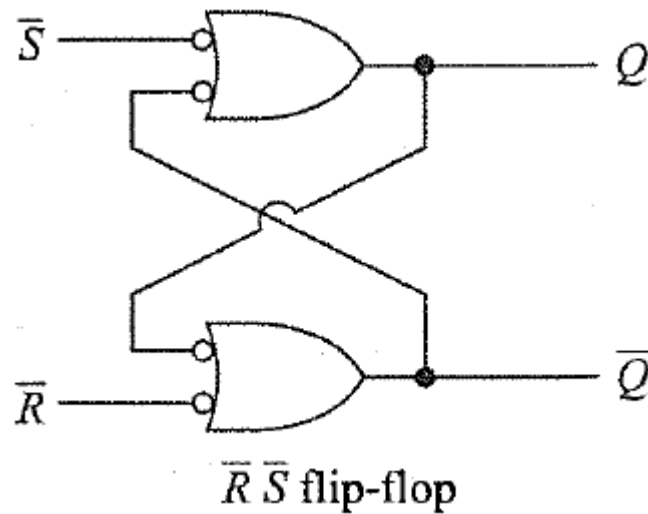
- Show how to convert the $\overline{R}\overline{S}$ flip-flop into an RS flip-flop.
- By placing an inverter at each input $\overline{R}\overline{S}$ flip-flop, the 2 inputs are now R and S , and the resulting circuit behaves exactly as the RS flip-flop.



R	S	Q
0	0	Last state
0	1	1
1	0	0
1	1	? (Forbidden)

RS FLIP-FLOPS-12

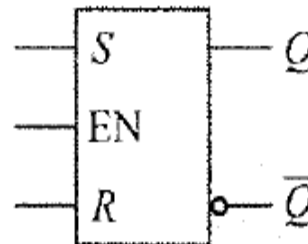
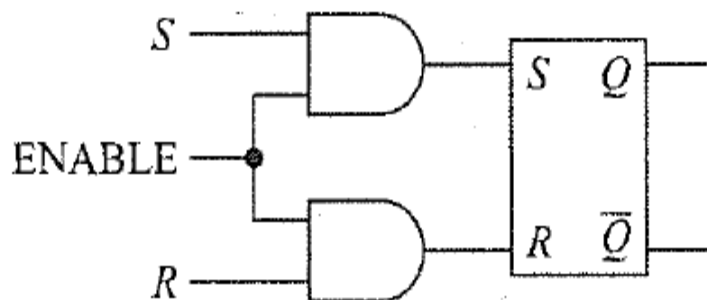
- Bubbled OR-gate equivalent to $\overline{R}\overline{S}$ flip-flop



GATED FLIP-FLOPS-1

“ Clocked RS Flip-Flops

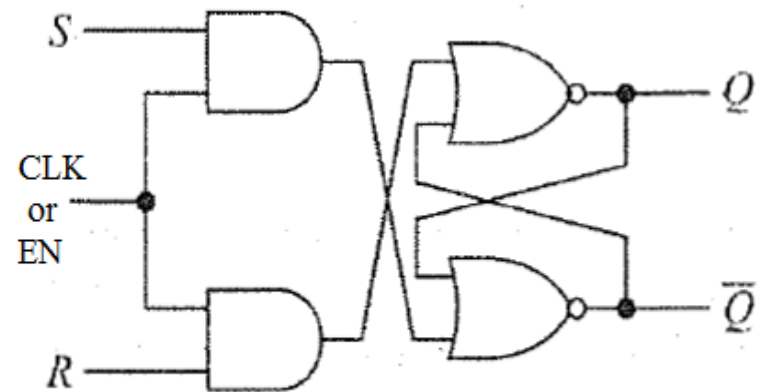
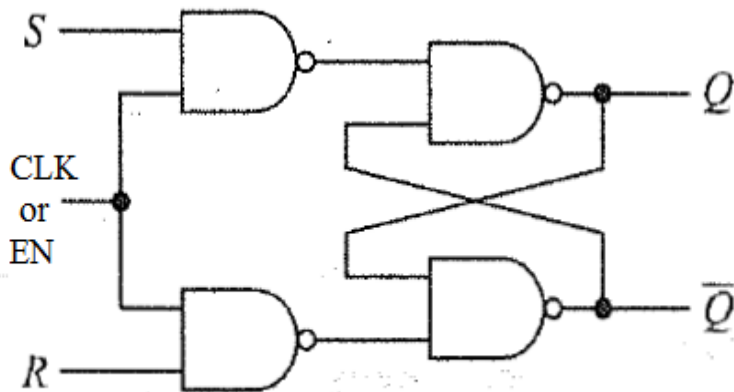
- “ The addition of two AND gates at the R and S inputs as shown in figure will result in a flip-flop that can be enabled or disabled.
- “ When the ENABLE input is low, the AND gate outputs must both be low and changes in neither R or S will have any effect on the flip-flop output Q . The latch is said to be *disabled*.
- “ *Strobe* or *clock* the flip-flop in order to store information (set it or reset it) at any time, and then hold the stored information for any desired period of time. This flip-flop is called a *gated* or *clocked RS flip-flop*.
- “ The time before the ENABLE goes low Q_n and the time just after ENABLE goes low Q_{n+1} .



EN	S	R	Q_{n+1}
1	0	0	Q_n (no change)
1	0	1	0
1	1	0	1
1	1	1	? (Illegal)
0	X	X	Q_n (no change)

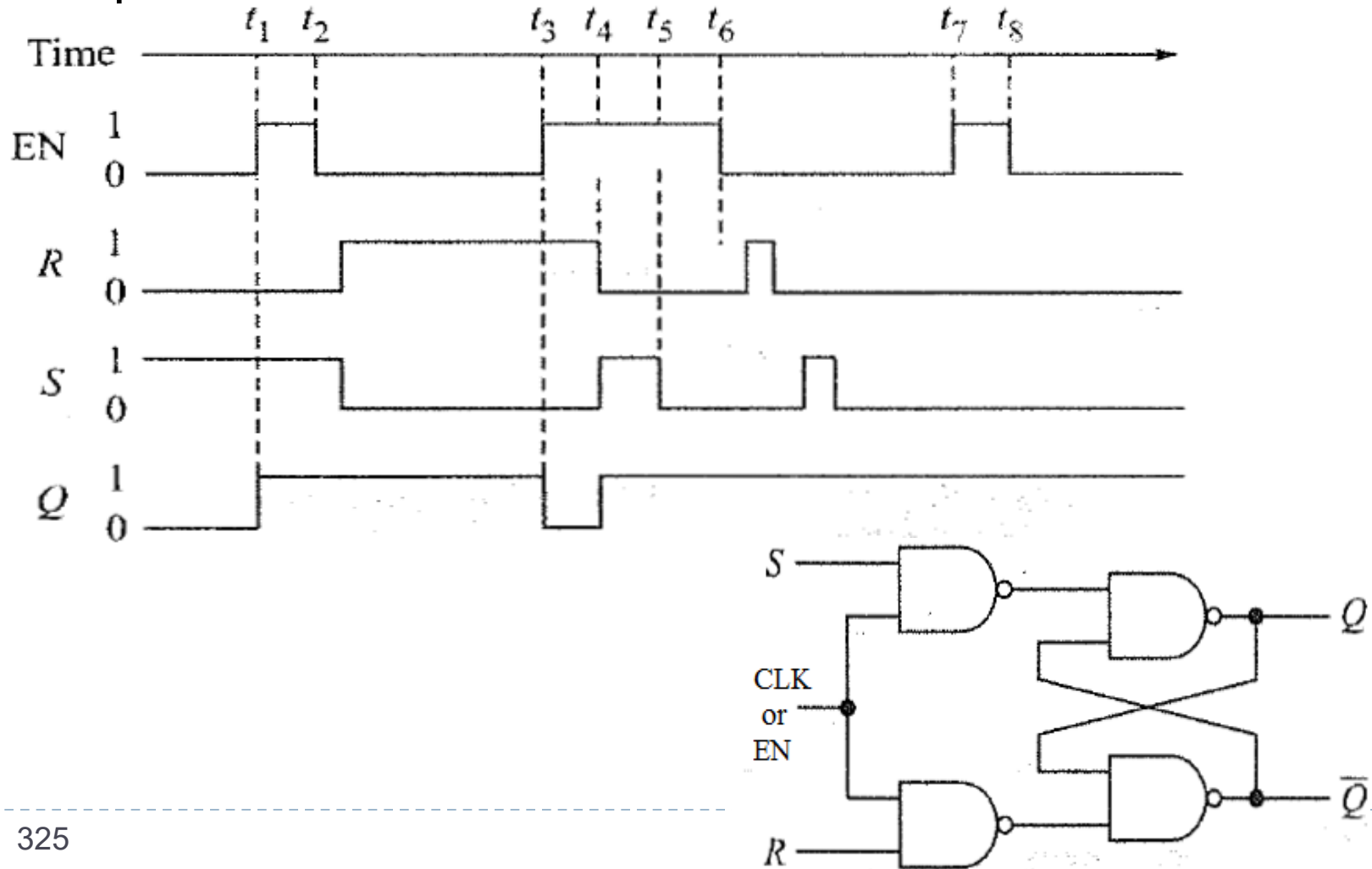
GATED FLIP-FLOPS-2

“ Two different realizations for a clocked *RS* flip-flop



GATED FLIP-FLOPS-3

“ Below figure shows the input waveforms R , S , and EN applied to a clocked RS flip-flop. Explain the output waveform Q .



GATED FLIP-FLOPS-4

“ Clocked D Flip-Flops

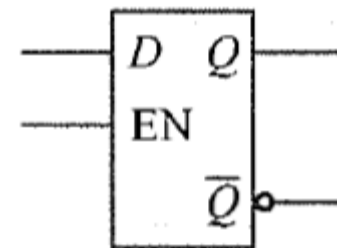
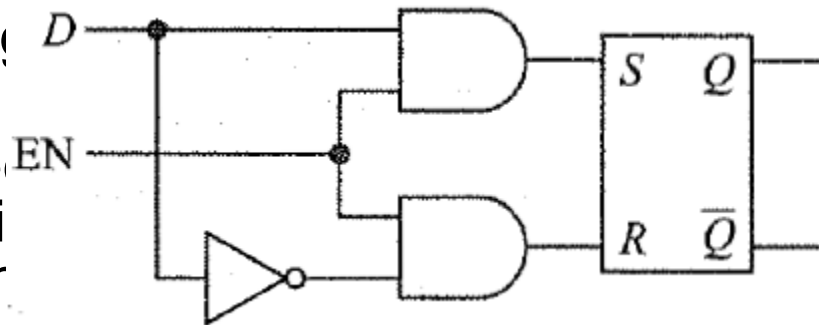
“ A D (Data) flip-flop having one input.

“ This flip-flop is disabled when EN is low, but it is *transparent* when EN is high.

“ D flip-flop in which Q can follow the value of D while EN is high.

“ In other words, if the data bit changes while EN is high, the last value of D before EN returns low is the value of D that is stored.

“ This kind of D flip-flop is often called a D latch.

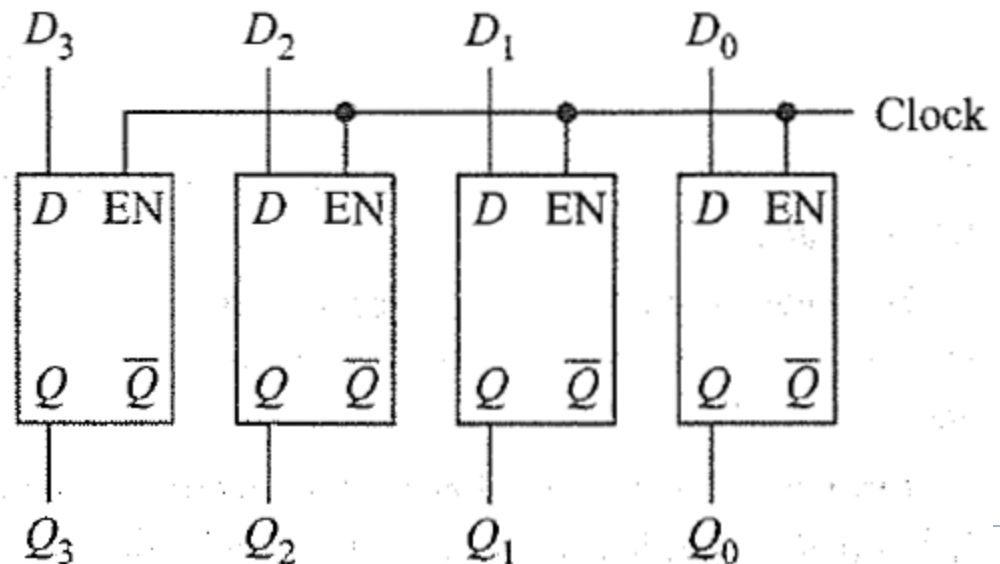


Logic symbol

EN	D	Q_{n+1}
0	X	Q_n (last state)
1	0	0
1	1	1

GATED FLIP-FLOPS-5

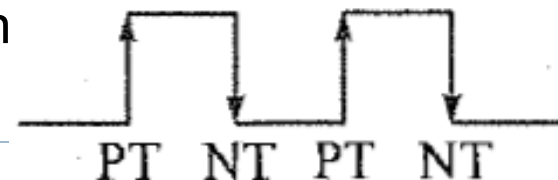
- “ **The idea of data storage in D Flip-Flop**
- “ Four *D* latches are driven by the same clock signal.
- “ When the clock goes high, input data is loaded into the flip-flops and appears at the output.
- “ Then when the clock goes low, the output retains the data.



EDGE-TRIGGERED RS FLIP-FLOPS -

1

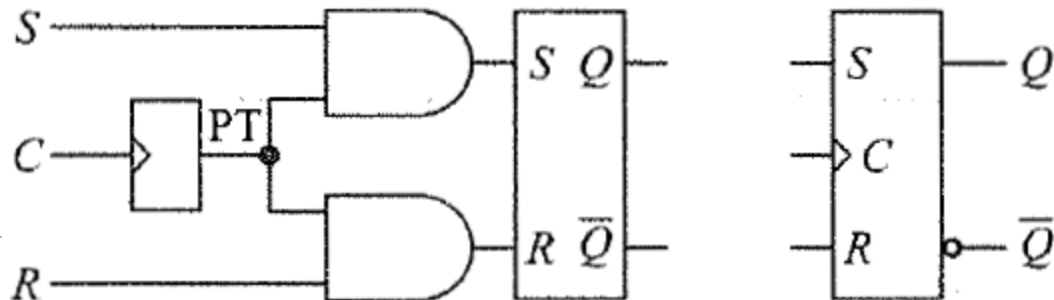
- “ If any of these flip-flops are used in a synchronous system, care must be taken to ensure that all flip-flop inputs change state in synchronism with the clock.
- “ One way of resolving the problem for gated flip-flops is to allow changes in R , S , and D input levels only when EN is low or require fixed levels at R , S , and D any time EN is high.
- “ Nearly all of the circuits in a digital system (computer) change states in *Synchronism* with the system clock.
- “ A change of state will either occur as the clock transitions from low to high or as it transitions from high to low.
- “ The low-to-high transition is frequently called the *positive transition (PT)*, as shown in Figure. The PT is given emphasis by drawing a small arrow on the *rising edge* of the clock waveform.
- “ A circuit that changes state at this time is said to be *positive-edge-triggered*. The high-to-low transition *transition (NT)*, as shown in Figure.



EDGE-TRIGGERED RS FLIP-FLOPS - 2

“ Positive-Edge-Triggered *RS* Flip-flops

- “ The small triangle inside the symbol (dynamic input indicator) indicates that Q can change state only with PTs of the clock (C).
- “ Each PT of the clock produces a very narrow PT that is applied to the AND gates.
- “ The AND gates are active only while the PT is high (perhaps 25 ns), and thus Q can change state only during this short time period. In this manner Q changes state in synchronism with the PTs of the clock.

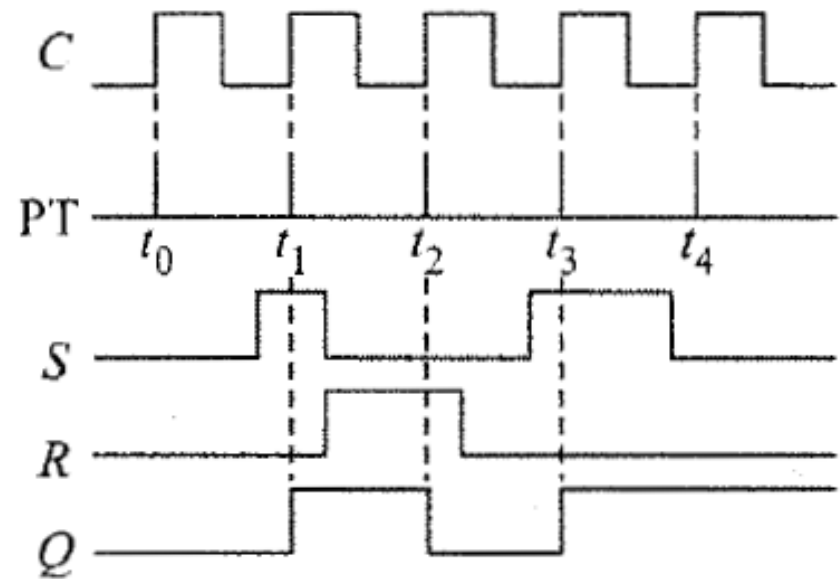


EDGE-TRIGGERED RS FLIP-FLOPS - 3

“ Truth Table and Timing Diagram

C	S	R	Q_{n+1}	Action
↑	0	0	Q_n	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	?	Illegal

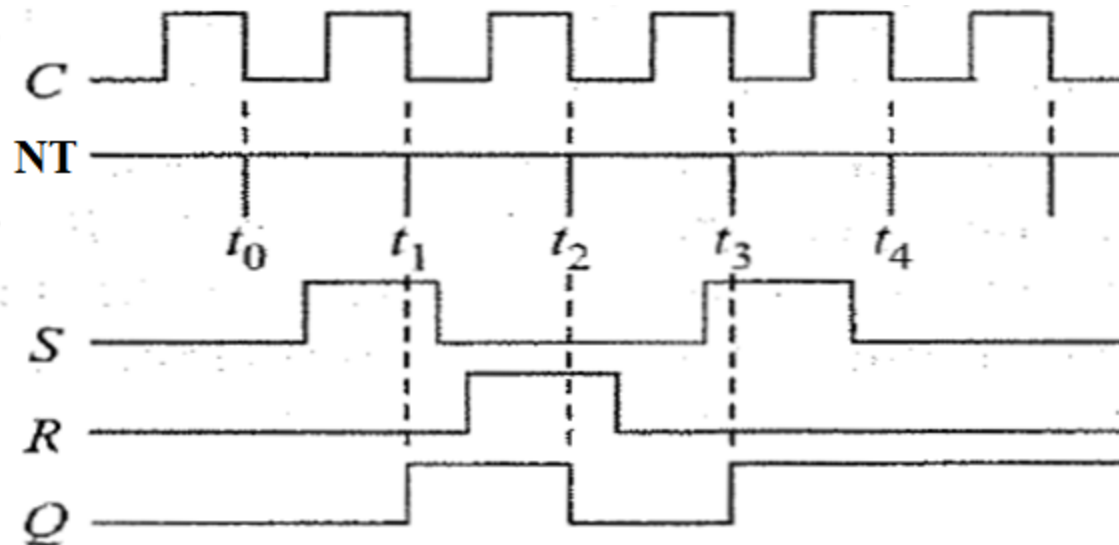
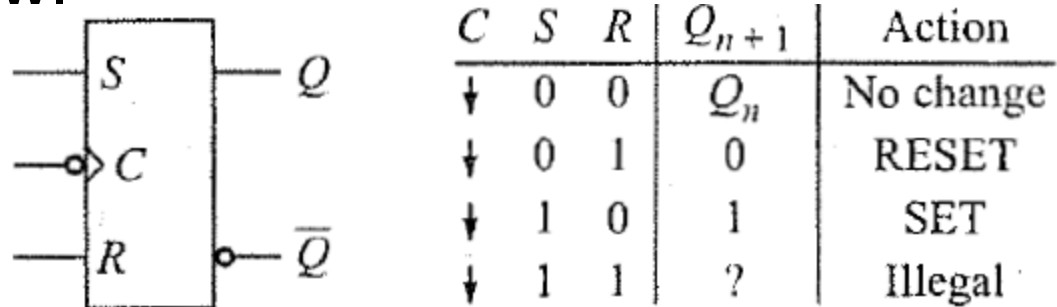
(c) Truth table



(d) Positive-edge-triggered RS flip-flop

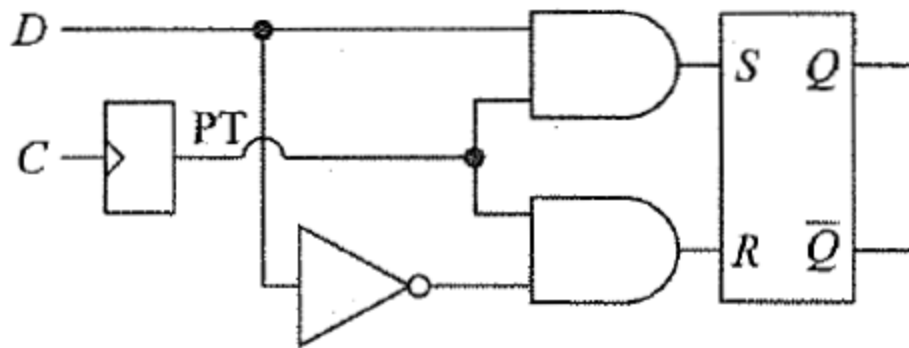
EDGE-TRIGGERED RS FLIP-FLOPS -4

- “ **Negative-Edge-Triggered RS Flip-Flops**
- “ The small bubble on the clock input (C) means active-low.



EDGE-TRIGGERED D FLIP-FLOPS -1

- “ When the clock is low, D is a don't care and Q is latched in its last state.
- “ On the leading edge of the clock (PT), designated by the up arrow, the data bit is loaded into the flip-flop and Q takes on the value of D .



(a) Circuit diagram

C	D	Q_{n+1}
0	X	Q_n (last state)
↑	0	0
↑	1	1

(c) Truth table

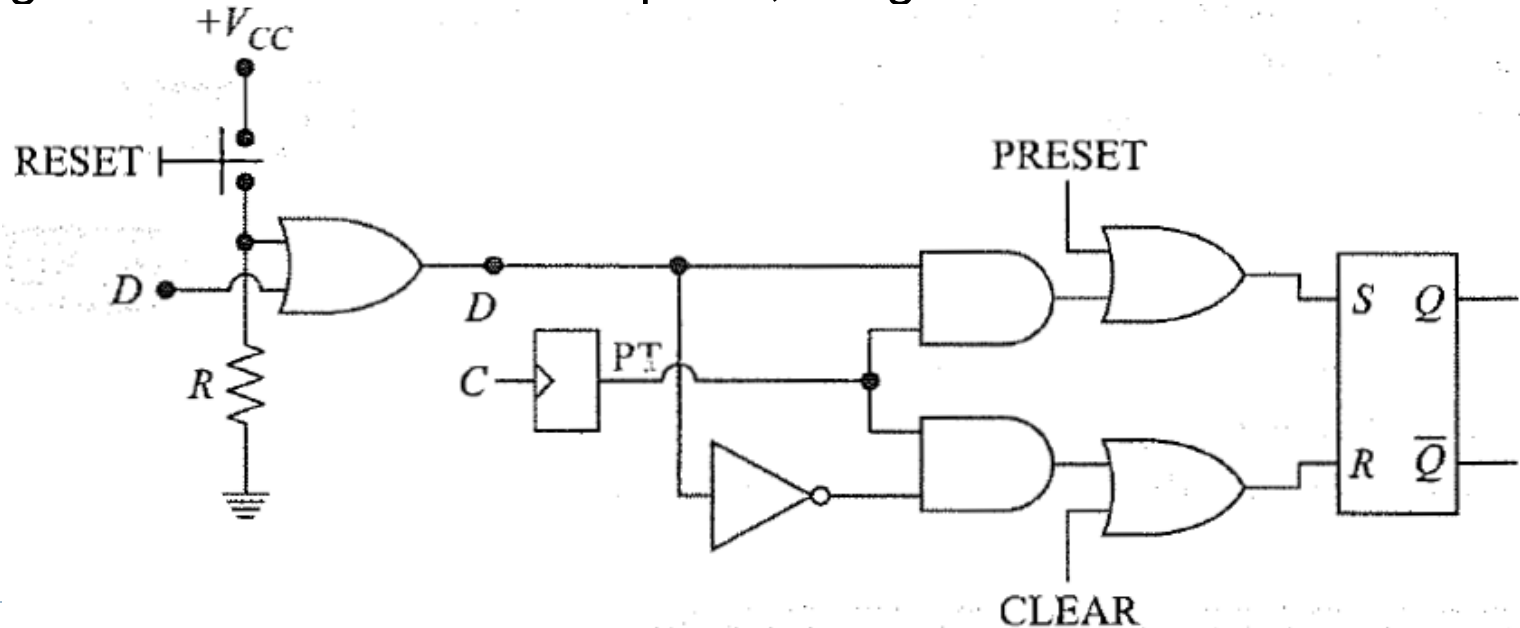
Positive-edge-triggered D flip-flop

EDGE-TRIGGERED D FLIP-FLOPS -2

- “ To get some computers started, an operator has to push a RESET button. This sends a CLEAR or PRESET signal to all flip-flops. Also, it's necessary in some digital systems to preset (synonymous with set) certain flip-flops.

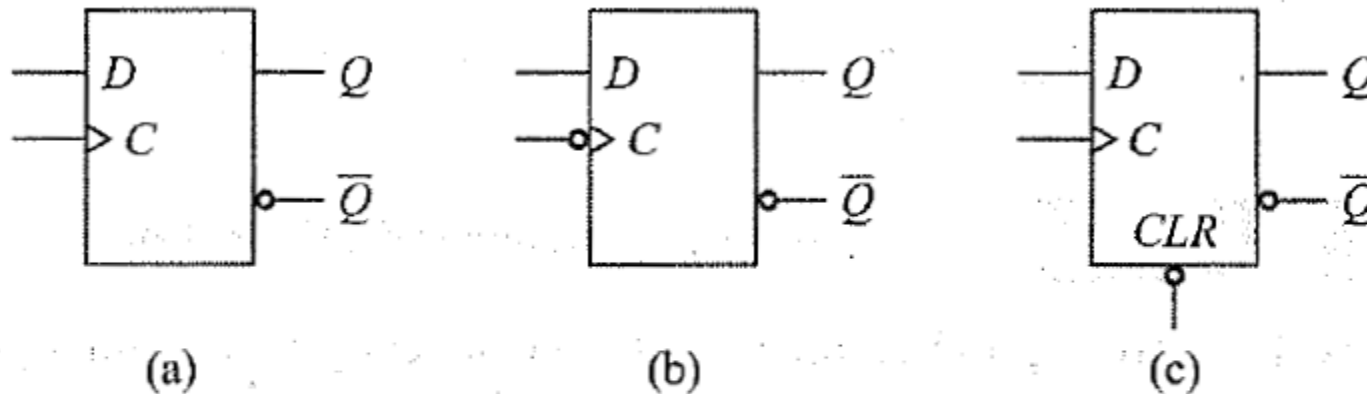
EDGE-TRIGGERED D FLIP-FLOPS -3

- “ Depressing the RESET button will set Q to 1 with the first PT of the clock. Q will remain high as long as the button is held closed. The first PT of the clock after releasing the button will set Q according to the D input.
- “ Furthermore, the OR gates allow us to slip in a high PRESET or a high CLEAR when desired.
- “ A high PRESET forces Q to equal 1; a high CLEAR resets Q to 0.



PRESET and CLEAR functions

EDGE-TRIGGERED D FLIP-FLOPS -4



► D flip-flop symbols: (a) Positive-edge-triggered, (b) Negative-edge-triggered, (c) Positive-edge-triggered with active low clear

EDGE-TRIGGERED JK FLIP-FLOPS - 1

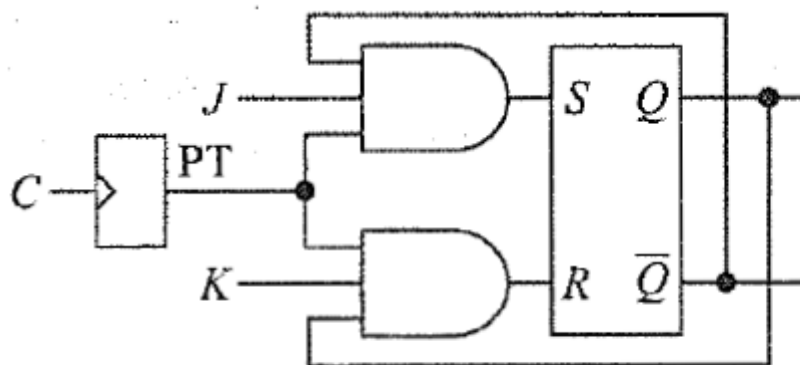
- J-Jack K- Kilby (Jack Kilby inventor of Integrated Circuits in 1958 and the Hand-held Calculator in 1966)
- Setting $R = S = 1$ with an edge-triggered RS flip-flop forces both Q and \bar{Q} to the same logic level. This is an *illegal* condition, and it is not possible to predict the final state of Q .
- The JK flip-flop accounts for this illegal input, and is therefore a more versatile circuit.
- Flip-flops can be used to build counters. Counters can be used to count the number of PTs or NTs of a clock.

EDGE-TRIGGERED JK FLIP-FLOPS -

2

▶ Positive-Edge-Triggered J/K Flip-Flops

- ▶ The basic circuit is identical to the previous positive-edge-triggered RS flip-flop, with two important additions:
 - ▶ The Q output is connected back to the input of the lower AND gate.
 - ▶ The \bar{Q} output is connected back to the input of the upper AND gate.
- ▶ This cross-coupling from outputs to inputs changes the RS flip-flop into a JK flip-flop. The previous S input is now labelled J , and the previous R input is labelled K .



C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	\bar{Q}_n (toggle)	Toggle

(a) One way to implement a JK flip-flop

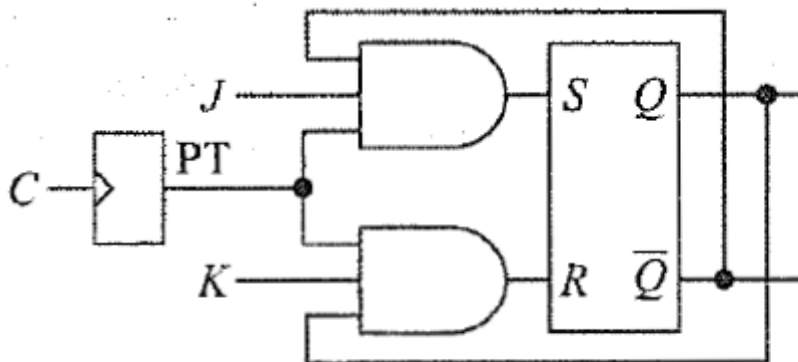
(b) Truth table

A positive-edge-triggered JK flip-flop

EDGE-TRIGGERED JK FLIP-FLOPS - 3

“ How it works:

- “ When J and K are both low, both AND gates are disabled. Therefore, clock pulses have no effect.
- “ This first possibility is the initial entry in the truth table. As shown, when J and K are both 0s, Q retains its last value.



(a) One way to implement a JK flip-flop

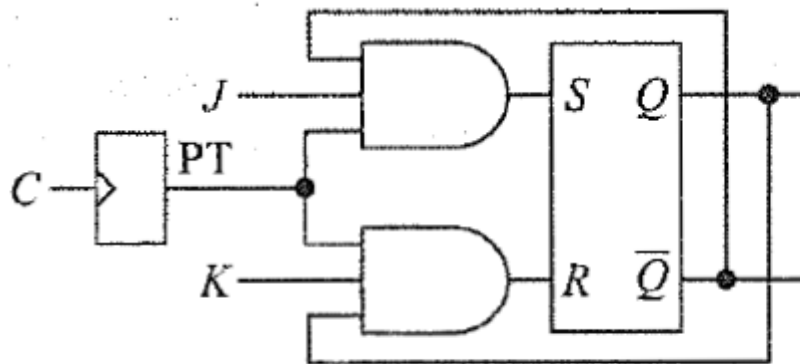
C	J	K	Q_{n+1}	Action
\uparrow	0	0	Q_n (last state)	No change
\uparrow	0	1	0	RESET
\uparrow	1	0	1	SET
\uparrow	1	1	\bar{Q}_n (toggle)	Toggle

(b) Truth table

A positive-edge-triggered JK flip-flop

EDGE-TRIGGERED JK FLIP-FLOPS - 4

- “ When J is low and K is high, the upper gate is disabled, so there's no way to set the flip-flop. The only possibility is reset.
- “ When Q is high, the lower gate passes a RESET pulse as soon as the next positive clock edge arrives. This forces Q to become low (the second entry in the truth table). Therefore, $J = 0$ and $K = 1$ means that the next PT of the clock resets the flip-



C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	\overline{Q}_n (toggle)	Toggle

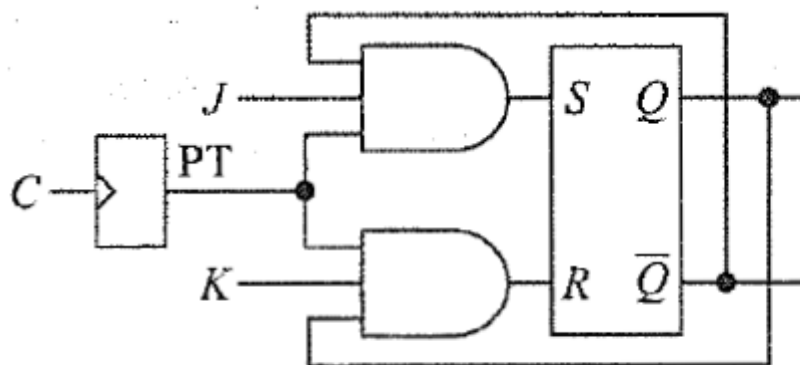
(a) One way to implement a JK flip-flop

(b) Truth table

A positive-edge-triggered JK flip-flop

EDGE-TRIGGERED JK FLIP-FLOPS - 5

- When J is high and K is low, the lower gate is disabled, so it's impossible to reset the flip-flop.
- When Q is low, \bar{Q} is high; therefore, the upper gate passes a SET pulse on the next positive clock edge. This drives Q into the high state (the third entry in the truth table).
- As, $J = 1$ and $K = 0$ means that the next PT of the clock sets the flip-flop (unless Q is already high).



C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	\bar{Q}_n (toggle)	Toggle

(a) One way to implement a JK flip-flop

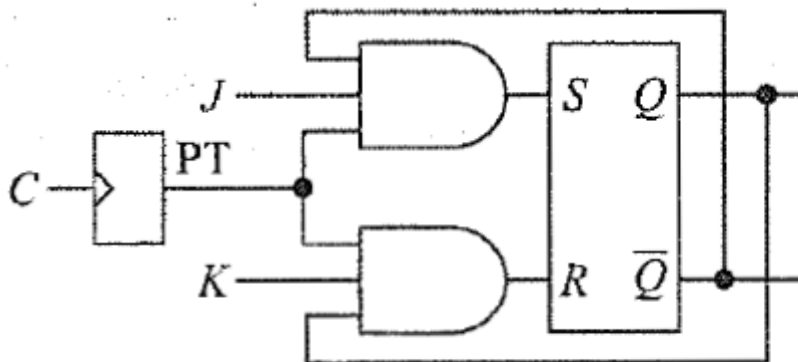
(b) Truth table

A positive-edge-triggered JK flip-flop

EDGE-TRIGGERED JK FLIP-FLOPS -

6

- “ When J and K are both high, it's possible to set or reset the flip-flop.
- “ If Q is high, the lower gate passes a RESET pulse on the next PT. On the other hand, when Q is low, the upper gate passes a SET pulse on the next PT.
- “ Either way, Q changes to the complement of the last state (see the truth table).
- “ Therefore, $J = 1$ and $K = 1$ mean the flip-flop will *toggle* (switch to the opposite state) on the next positive clock edge



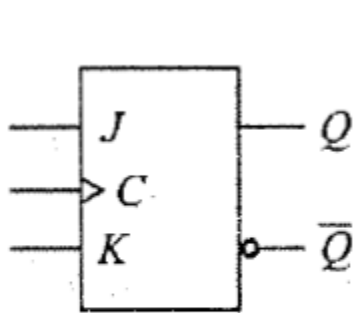
C	J	K	Q_{n+1}	Action
↑	0	0	Q_n (last state)	No change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	\bar{Q}_n (toggle)	Toggle

(a) One way to implement a JK flip-flop

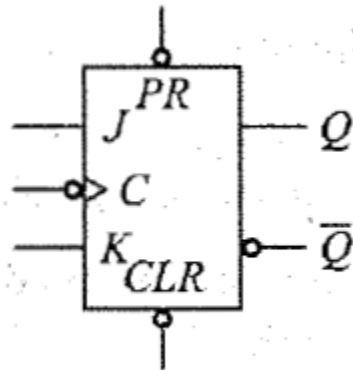
(b) Truth table

A positive-edge-triggered JK flip-flop

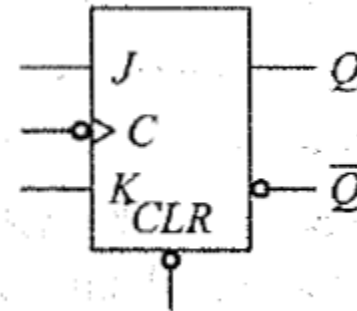
EDGE-TRIGGERED JK FLIP-FLOPS - 7



(a) Basic symbol



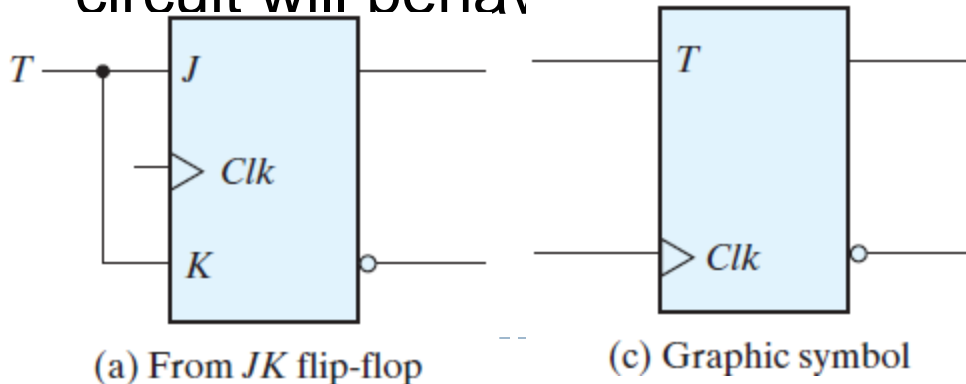
(b) 74LS76A



(c) 74LS73A

EDGE-TRIGGERED JK FLIP-FLOPS -8

- “ Toggle flip-flop, popularly known as T flip-flop has following input-output relation.
- “ When input $T = 0$, the output Q does not change its state. For $T = 1$, the output Q toggles its value.
- “ For JK flip-flop input $J = K = 0$, the output $Q_{n+1} = Q_n$ i.e. output does not change its state.
- “ For $J = K = 1$, the output $Q_{n+1} = Q_n'$ i.e. output toggles.
- “ Thus, if we tie J and K inputs of JK flip-flop together and make a common input $T = J = K$, the resulting circuit will behave as T flip-flop



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

Module-4

Flip-Flops, Registers and Counters

Text Books Referred

- “ Donald P Leach, Albert Paul Malvino & Goutam Saha: Digital Principles and Applications, 7th Edition, Tata McGraw Hill, 2015
- “ M. Morris Mano, Michael D. Ciletti : Digital Design With an Introduction to the Verilog HDL, 5th Edition, Pearson Education, Inc

FLIP-FLOP TIMING-1

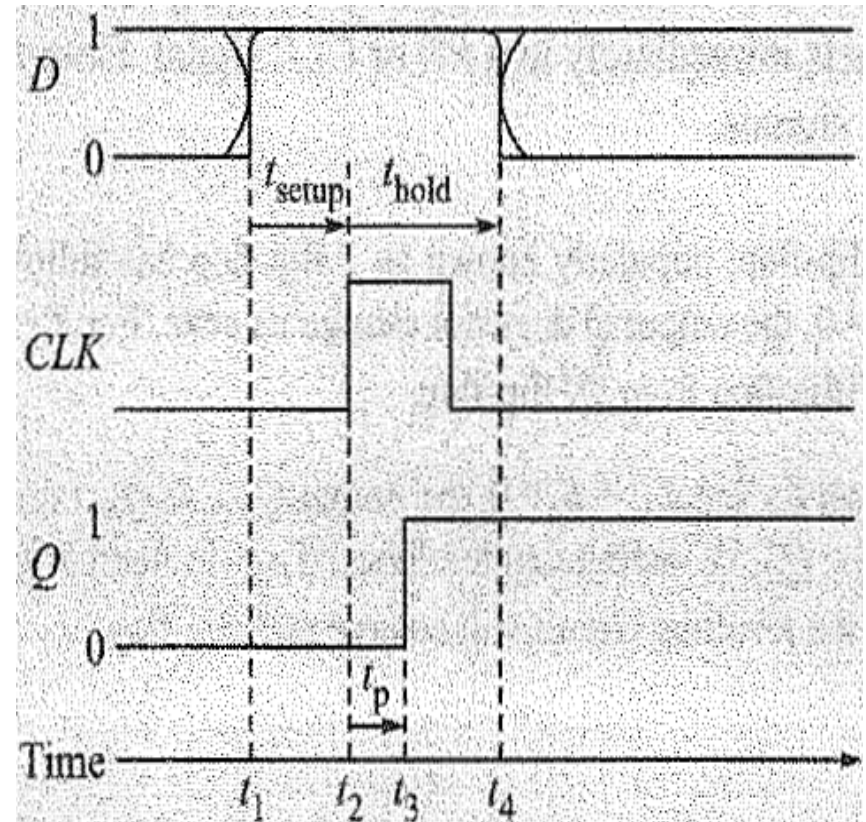
- “ Diodes and transistors cannot switch states immediately.
- “ It always takes a small amount of time to turn a diode on or off. A transistor takes time for to switch from saturation to cut-off, and vice versa.
- “ For bipolar diodes and transistors, the switching time is in the nanosecond region.
- “ Switching time is the main cause of **propagation delay (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 instance, if an edge-triggered D flip-flop lists $t_p = 10$ ns, it takes about 10 ns for Q to change states after D has been sampled by the clock edge.
- “ This propagation delay time has been used to construct the "pulse-forming circuit" used with edge-triggered flip-flops.
- “ When flip-flops are used to construct counters, the propagation delay is often small enough to be ignored.

FLIP-FLOP TIMING-2

- “ The D input makes it necessary for data bit 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.
- “ E.g. a D flip-flop has a setup time of 15 ns, the data bit to be stored must be at the D input at least 15 ns before the clock edge arrives; otherwise, the not guarantee correct sampling and storing.
- “ Data bit D has to be held long enough for the internal transistors to switch states. Only after the transition is assured can allow data bit D to change.
- “ **Hold time** (t_{hold}) is the minimum amount of time that data bit D must be present after the PT of the clock.
- “ E.g., if $t_{\text{setup}} = 15$ ns and $t_{\text{hold}} = 5$ ns, the data bit has to be at the D input at least 15 ns before the clock edge arrives and held at least 5 ns after the clock PT.

FLIP-FLOP TIMING-3

- “ Typical waveforms for setting a 1 in a positive-edge-triggered flip-flop.
- “ Prior to t_1 , the data can be a 1 or a 0, or can be changing.
- “ From time t_1 to t_2 , the data line D must be held steady. This is the setup time t_{setup} .
- “ Data is shifted into the flip-flop at time t_2 but does not appear at Q until time t_3 .
- “ The time from t_2 to t_3 is the propagation time t_p .
- “ In order to guarantee proper operation, the data line must be held steady from time t_2 until t_4 ; this is the hold time t_{hold} .
- “ After t_4 , D is free to change state.



RACE AROUND CONDITION OF JK FLIP FLOP -1

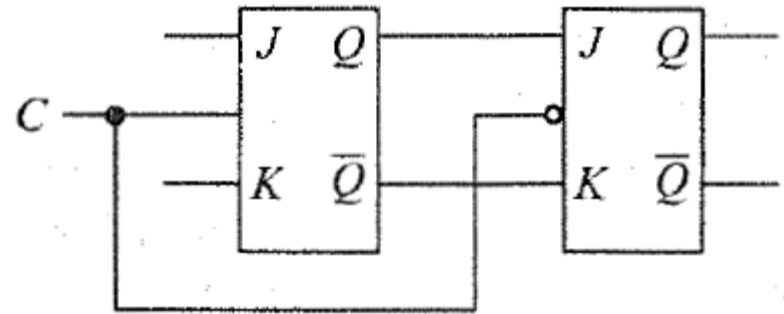
- “ In JK flip flop as long as clock is high for the input conditions $J \& K$ equals to the output changes or complements its output from $1 \rightarrow 0$ and $0 \rightarrow 1$.
- “ This is called toggling output or uncontrolled changing or racing condition.
- “ Consider above J&K circuit diagram as long as clock is high and $J \& K = 11$ then two upper and lower AND gates are only triggered by the complementary outputs Q and $Q(\text{bar})$.
- “ I.e. in any condition according to the propagation delay one gate will be enabled and another gate is disabled.
- “ If upper gate is disabled then it sets the output and in the next lower gate will be enabled which resets the flip flop output.

RACE AROUND CONDITION OF JK FLIP FLOP-2

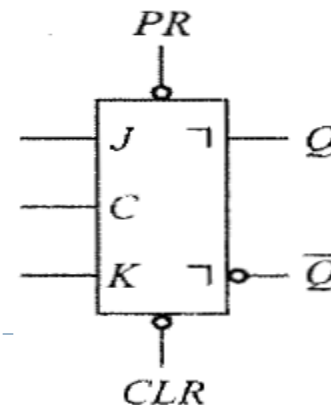
- “ **Steps to avoid racing condition in JK Flip flop:**
 - “ If the Clock On or High time is less than the propagation delay of the flip flop then racing can be avoided. This is done by using edge triggering rather than level triggering.
 - “ If the flip flop is made to toggle over one clock period then racing can be avoided. This introduced the concept of Master Slave JK flip flop.

JK MASTER-SLAVE FLIP-FLOPS-1

- “ To begin with, the master is positive-level-triggered and the slave is negative-level-triggered.
- “ Therefore, the master responds to its J and K inputs before the slave.
- “ If $J = 1$ and $K = 0$, the master sets on the positive clock 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.

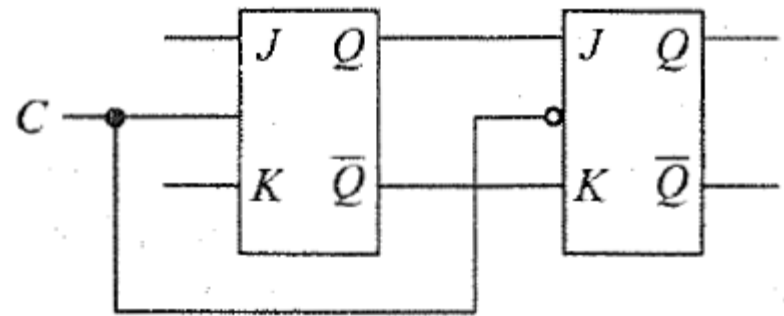


C	J	K	Q_{n+1}	Action
\neg	L	L	Q_n	No change
\neg	L	H	L	RESET
\neg	H	L	H	SET
\neg	H	H	\bar{Q}_n	Toggle

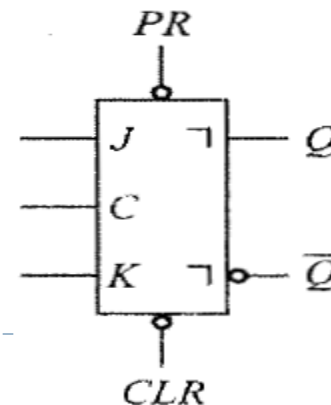


JK MASTER-SLAVE FLIP-FLOPS-2

- If $J = 0$ and $K = 1$, 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.

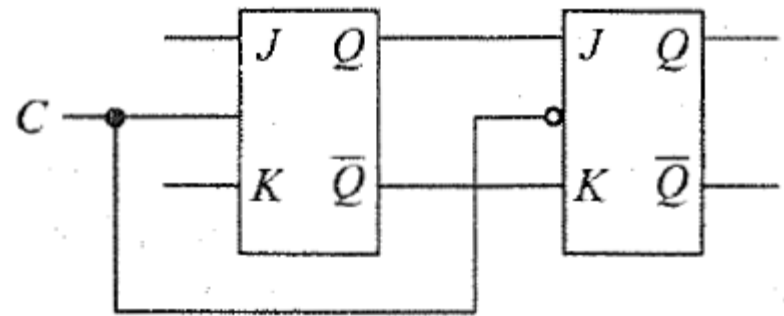


C	J	K	Q_{n+1}	Action
\neg	L	L	Q_n	No change
\neg	L	H	L	RESET
\neg	H	L	H	SET
\neg	H	H	\bar{Q}_n	Toggle

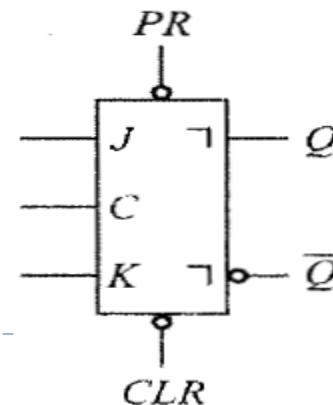


JK MASTER-SLAVE FLIP-FLOPS-3

- “ If the master's J and K inputs are both high, it toggles on the PT of the clock and the slave then toggles on the clock NT.
- “ Regardless of what the master does, therefore, the slave copies it: if the master sets, the slave sets; if the master resets, the slave resets.
- “ If $J = K = 0$, the flip-flop is disabled and Q remains unchanged.

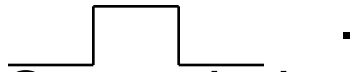



C	J	K	Q_{n+1}	Action
\neg	L	L	Q_n	No change
\neg	L	H	L	RESET
\neg	H	L	H	SET
\neg	H	H	\bar{Q}_n	Toggle



JK MASTER-SLAVE FLIP-FLOPS-4

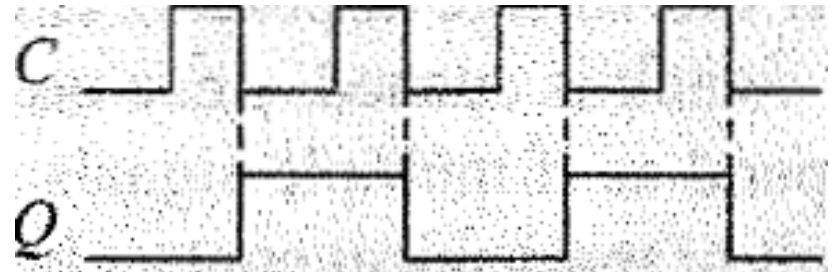
- “ The truth table reveals this action by means of the pulse symbol



- “ Second, the symbol  appearing next to the Q and the Q' outputs is the IEEE designation for a *postponed output*.
- “ In this case, it means Q does not change state until the clock makes an NT.
- “ In other words, the contents of the master are shifting into the slave on the clock NT, and at this time Q changes state.
- “ To summarize: 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 (Q changes state) when the clock goes low.
- “ This flip-flop might be referred to as *pulse-triggered*, to distinguish it from the edge-triggered flip-flops.

JK MASTER-SLAVE FLIP-FLOPS-5

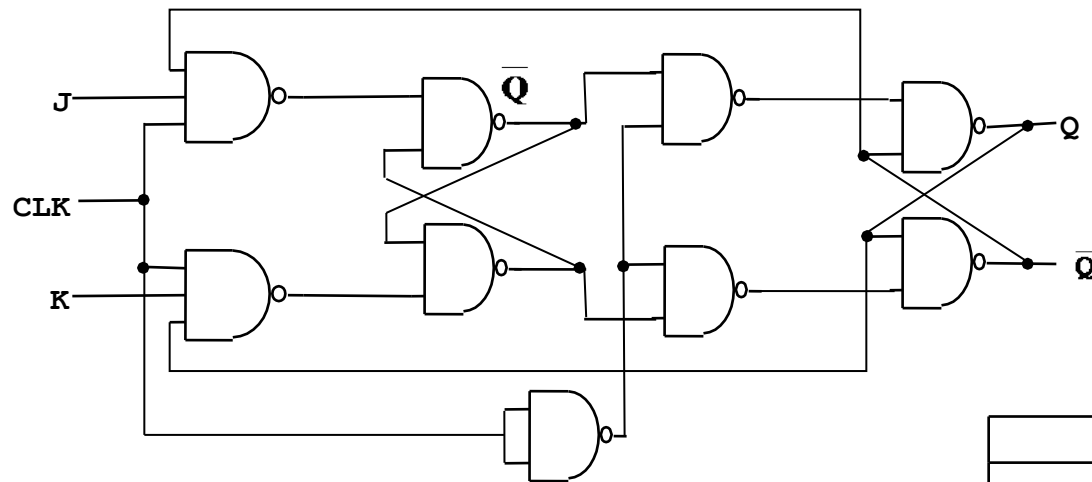
- “ The *JK* master-slave flip-flop *J* and *K* inputs tied to + *Vcc* and a series of pulses (actually a square wave) are applied to its *C* input.
- “ Since $J=K=1$, the flip-flop simply toggles each time the clock goes low.
- “ The waveform at *Q* has a period twice that of the *C* waveform.
- “ In other words, the frequency of *Q* is only one-half that of *C*.
- “ This circuit acts as a frequency divider - the output frequency is equal to the input frequency divided by 2.



JK MASTER-SLAVE FLIP-FLOPS-5

“ Lab Experiment:

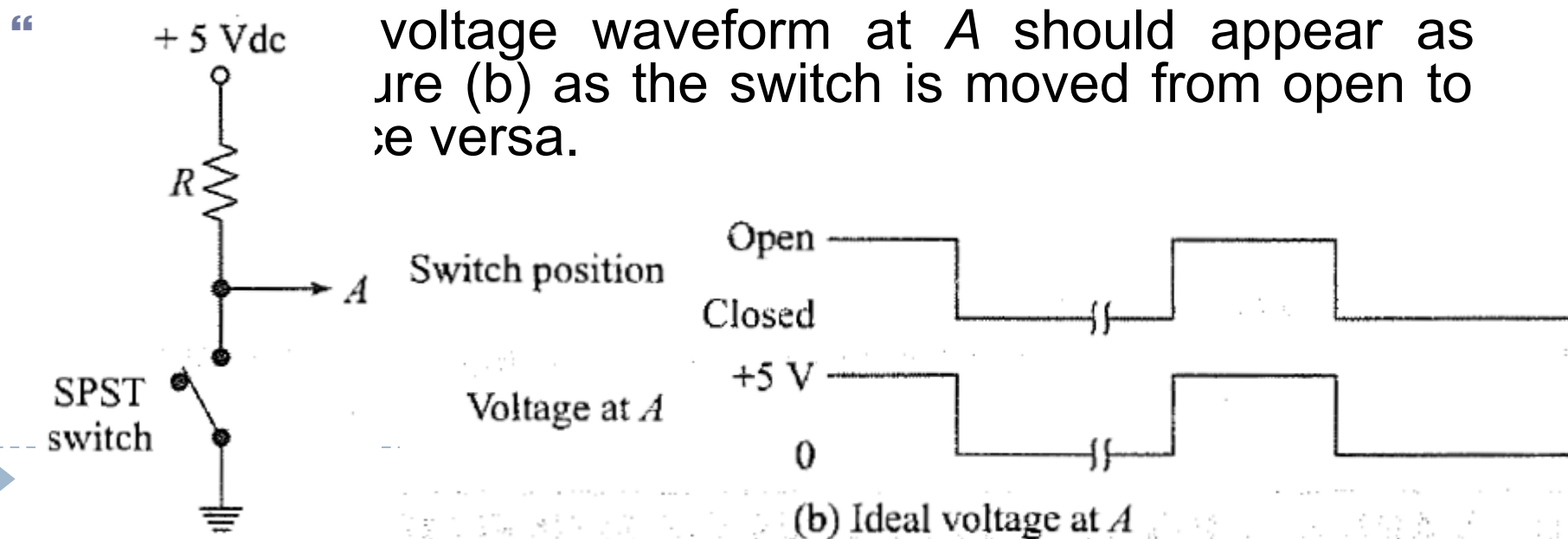
- “ Realize a J-K Master-Slave Flip-Flop using NAND gates and verify its truth table.



Input		Output
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	No Image

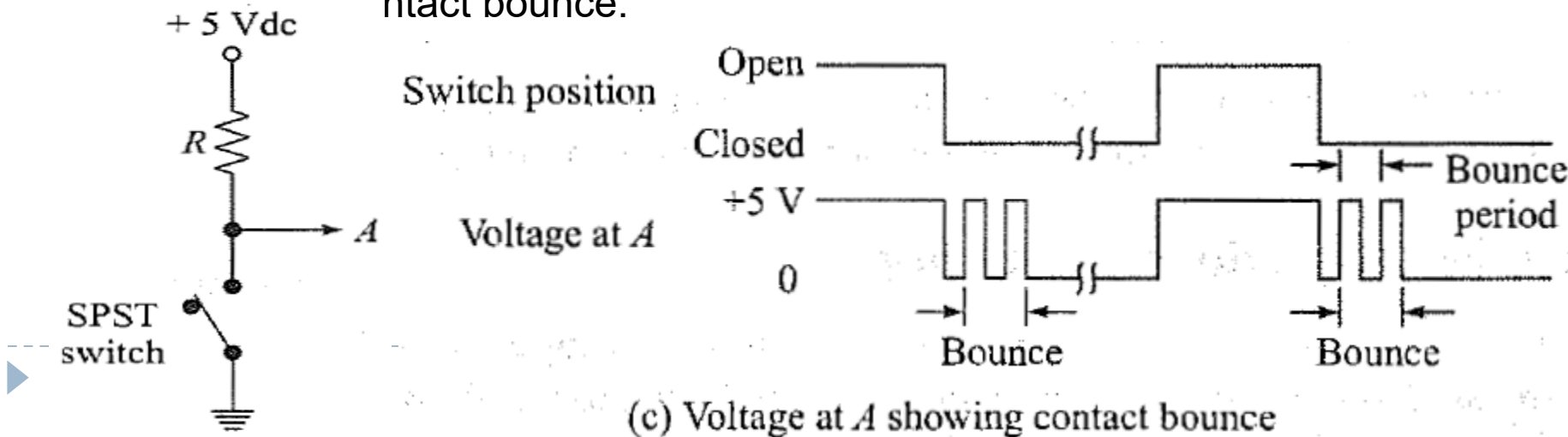
SWITCH CONTACT BOUNCE CIRCUITS-1

- “ Keyboard mechanical switch used to generate the logic signal.
- “ The single-pole single-throw (SPST) switch shown in figure.
- “ When the switch is open, the voltage at point A is $+5\text{ V}$ dc; when the switch is closed, the voltage at point A is 0 V dc.



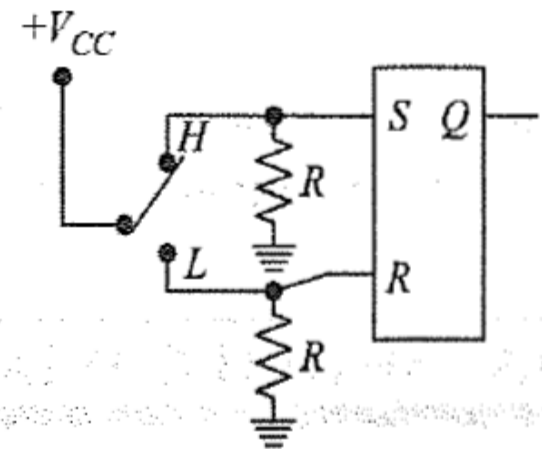
SWITCH CONTACT BOUNCE CIRCUITS-2

- “ But actually the waveform at point *A* will appear more or less as shown in figure (c), as the result of a phenomenon known as *contact bounce*.
- “ Any mechanical switching device consists of a moving contact arm restrained by some sort of a spring system. As a result, when the arm is moved from one stable position to the other, the arm bounces, much as a hard ball bounces when dropped on a hard surface. The number of bounces that occur and the period of the bounce differ for each switching device.
- “ Notice carefully that in this particular instance, even though actual physical contact bounce occurs each time the switch is opened or closed, contact bounce appears in the voltage level at point *A* only when the switch is closed.
- “ When the switch is closed, the circuit will respond as if multiple signals were applied, rather than the single-switch closure intended-the undesired result of mechanical contact bounce.

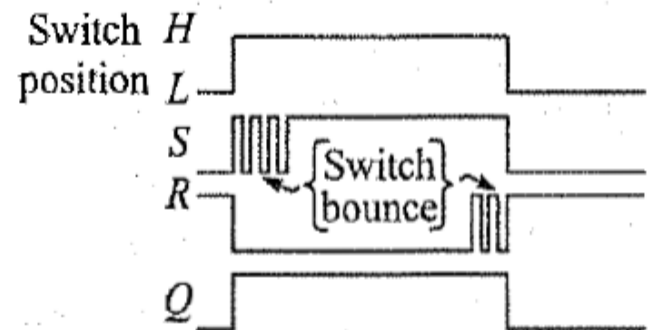


SWITCH CONTACT BOUNCE CIRCUITS-3

- “ A Simple RS latch Debounce Circuit to eliminate the contact bounce problem.
- “ When the switch is moved to position H , $R = 0$ and $S = 1$. Bouncing occurs at the S input due to the 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, losing contact, the input signals are $R = S = 0$, therefore the flip-flop remains set ($Q = 1$). When the switch regains contact, $R = 0$ and $S = 1$; this causes an attempt to again set the flip-flop. But since the flip-flop is already set, no changes occur at Q .
- “ The result is that the flip-flop responds to the first, and only to the first, high level at its S input, resulting in a "clean" low-to-high signal at its output (Q).



(a) Switch contact bounce eliminator



(b) Switch bounce

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-1

“ 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.

R	S	Q_n	Action
0	0	Last state	No change
0	1	1	SET
1	0	0	RESET
1	1	?	Forbidden

D	Q_{n+1}
X	Q_n (last state)
0	0
1	1

J	K	Q_{n+1}
0	0	Q_n (last state)
0	1	0
1	0	1
1	1	\bar{Q}_n (toggle)

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

SR	Q_n	00	01	11	10
0	0	0	0	×	1
1	1	1	0	×	1

(a) $Q_{n+1} = S + \bar{R} Q_n$

D	Q_n	0	1
0	0	0	1
1	1	0	1

(b) $Q_{n+1} = D$

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

(c) $Q_{n+1} = J \bar{Q}_n + \bar{K} Q_n$

T	Q_n	0	1
0	0	0	1
1	1	1	0

(d) $Q_{n+1} = T \bar{Q}_n + \bar{T} Q_n$

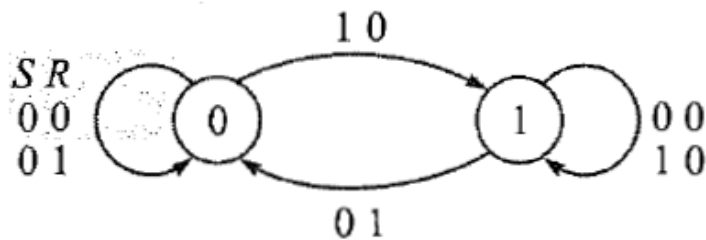
Characteristic equations of (a) SR flip-flop, (b) D flip-flop, (c) JK flip-flop, (d) T flip-flop

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-2

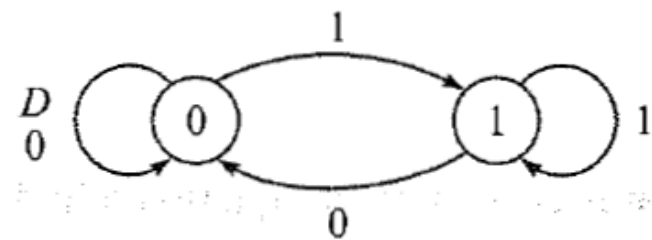
- “ **Flip-Flops as (FSM) Finite State Machine :**
- “ In a sequential logic circuit the value of all the memory elements at a given time define the *state* of that circuit at that time.
- “ Finite State Machine (FSM) concept offers a better alternative to truth table in understanding progress of sequential logic with time. For a complex circuit a truth table is difficult to read as its size becomes too large.
- “ In FSM, functional behavior of the circuit is explained using finite number of states.

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-3

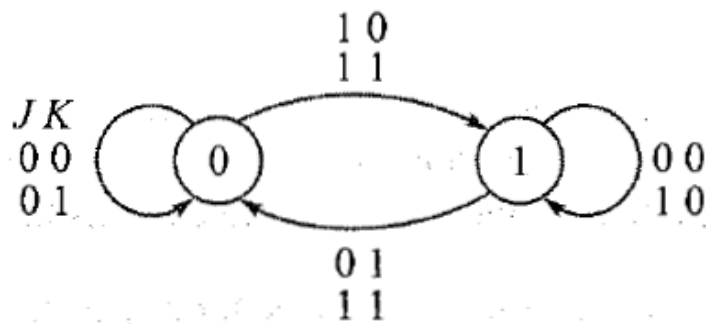
“ All the flip-flops are represented as finite state machine through their state transition diagrams.



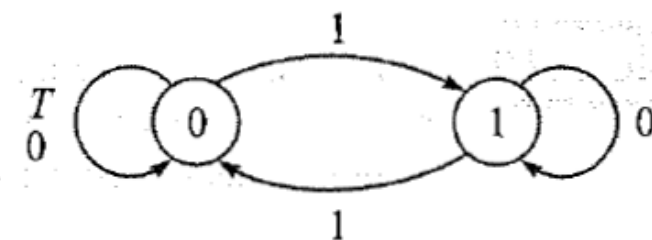
(a) *SR* flip-flop



(b) *D* flip-flop



(c) *JK* flip-flop

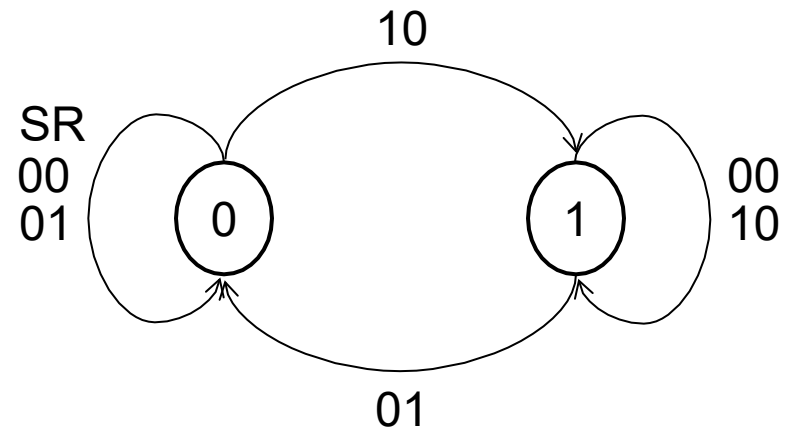
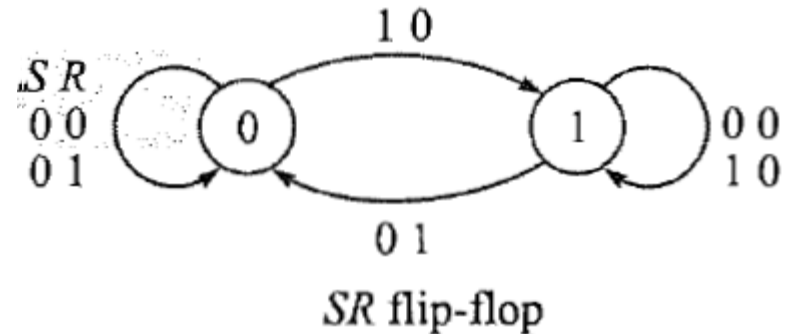


(d) *T* flip-flop

State transition diagram of (a) *SR* flip-flop, (b) *D* flip-flop, (c) *JK* flip-flop, (d) *T* flip-flop

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-4

- “ SR flip-flop is developed from its truth table or characteristic equation. Each flip-flop can be at either of 0 or 1 state defined by its stored value at any given time. Application of input may change the stored value, i.e. state of the flip-flop. This is shown by directional arrow and the corresponding input is written alongside.
- “ If SR flip-flop stores 0, then for $SR = 00$ or 01 the stored value does not change.
- “ For $SR = 10$, flip-flop output changes to 1.
- “ Note that, $SR = 11$ is not allowed in SR flip-flop.
- “ When SR flip-flop stores 1, application of $SR = 00$ or 10 does not change its value and only when $SR = 01$, output



VARIOUS REPRESENTATIONS OF FLIP-FLOPS-5

“ Flip-Flop Excitation Table:

- “ Excitation table of a flip-flop is looking at its truth table in a reverse way.
- “ 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 and directly from its state transition

$Q_n \rightarrow Q_{n+1}$		S	R	J	K	D	T
0	0	0	×	0	×	0	0
0	1	1	0	1	×	1	1
1	0	0	1	×	1	0	1
1	1	×	0	×	0	1	0

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-6

- “ Excitation table for SR Flip-Flop
- “ One can see if present state is 0 application of $SR = 0x$ does not alter its value where 'x' denotes don't care condition in R input.
- “ 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 state 1 is maintained if $SR = 0$, i.e. $SR = 00$ or $SR = 01$.

$Q_n \rightarrow Q_{n+1}$		S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

VARIOUS REPRESENTATIONS OF FLIP-FLOPS-7

- “ Example:
- “ A fictitious flip-flop with two inputs A and B functions like this. For $AB = 00$ and 11 the output becomes 0 and 1 respectively. For $AB = 01$, flip-flop retains previous output while output complements for $AB = 10$. Draw the truth table, excitation table and state transition diagrams of this flip-flop.

A	B	Q_{n+1}
0	0	0
0	1	Q_n
1	0	$\overline{Q_n}$
1	1	1

Truth table

$Q_n \rightarrow Q_{n+1}$	A	B
$0 \rightarrow 0$	0	\times
$0 \rightarrow 1$	1	\times
$1 \rightarrow 0$	\times	0
$1 \rightarrow 1$	\times	1

Excitation table

HDL Implementation of FLIP-FLOP-1

- “ Use behavioural model is preferred for sequential circuit and **always** keyword is used in all these circuits.
- “ For any flip-flop if $EN = 1$, output changes according to equation and if $EN = 0$, output does not change, i.e. remains latched to previous value.

HDL Implementation of FLIP-FLOP-2

D Flip Flop

```
module (D,EN, Q);  
Input D,EN;  
Output Q;  
always @ ( EN or D)  
if (EN)  
Q=D;  
//from characteristic  
equation  
endmodule
```

SR 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) ;  
//from characteristic  
equation  
endmodule
```

HDL Implementation of FLIP-FLOP-3

- “ A clocked flip-flop.
- “ E.g. A *D* flip-flop with positive edge trigger, negative edge trigger and positive edge trigger with reset (CLR).
- “ Here, the CLR input is active low, i.e. it clears the output ($Q = 0$) when CLR is 0.
- “ Use keywords **posedge** and **negedge** for this.
- “ With keyword **always** it ensures execution of always block once every clock cycle at corresponding edge.
- “ The **always** sensitivity list (after@) contains any number of edge statements including clock and asynchronous inputs.
- “ The **always** block puts all asynchronous conditions in the beginning through **else** or **else if** and the *last else* statement responds to clock transition.

HDL Implementation of FLIP-FLOP-4

D flip-flop with
Positive edge trigger

```
module  
DFFpos(D,C,Q);  
input D,C;  
//C is clock  
output Q;  
reg Q;  
always@ (posedge C)  
Q=D;  
endmodule
```

D flip-flop with
Negative edge trigger

```
module  
DFFneg(D,C,Q);  
input D,C;  
//C is clock  
output Q;  
reg Q;  
always @ (negedge C)  
Q=D;  
endmodule
```

D flip-flop with positive
edge trigger with reset
(CLR).

```
module  
DFFpos_clr(D,C,CLR,Q);  
input D,C,CLR;  
//C is clock  
output Q;  
reg Q;  
always@ (posedge C or  
negedge CLR)  
if (~CLR) Q=1'b0;  
//Q stores 1 binary bit 0  
else Q=D;  
endmodule
```

Introduction

- “ A register is simply a group of flip-flops that can be used to store a binary number. There must be one flip-flop for each bit in the binary number.
- “ A register is a digital circuit with two basic functions: data storage and data movement.
 - “ E.g. a register used to store an 8-bit binary number must have eight flip-flops.
- “ Naturally the flip-flops must be connected such that the binary number can be entered (shifted) into the register and possibly shifted out.
 - “ A group of flip-flops connected to provide either or both of these functions is called a **shift register**.
- “ Applications:
 - “ A data register is often used to momentarily store binary information e.g. RAM.

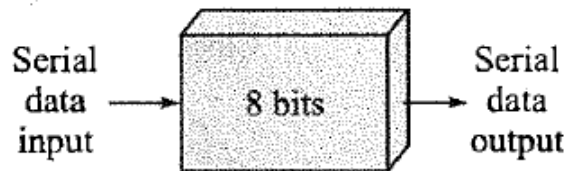
▶ 371 A register used in a microprocessor chip. E.g. Processor Register

“ Shift Register is a sequence generator and sequence detector and

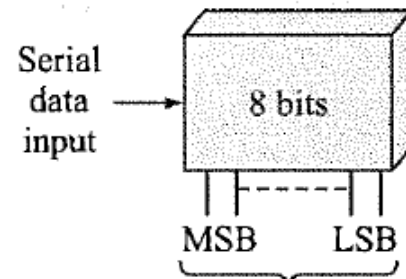
TYPES OF REGISTERS-1

- “ The bits in a binary number can be moved from one place to another in either of two ways.
- “ The first method involves shifting the data 1 bit at a time in a serial fashion, beginning with either the most significant bit (MSB) or the least significant bit (LSB). This technique is referred to as *serial shifting*.
- “ The second method involves shifting all the data bits simultaneously and is referred to as *parallel shifting*.
- “ There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register.
- “ This leads to the construction of four basic register types:
 - “ 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) 54/74198, 8 bits

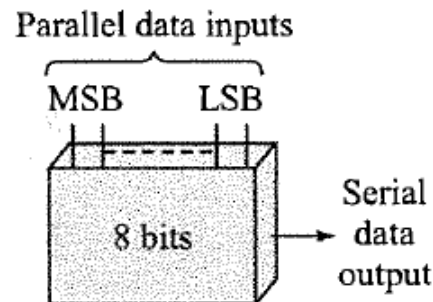
TYPES OF REGISTERS-2



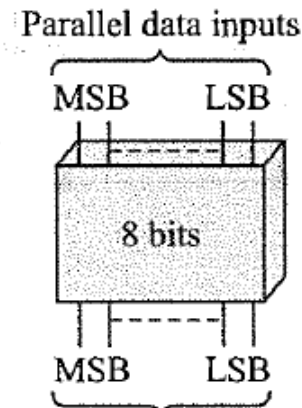
(a) Serial in-serial out



Parallel data outputs
(b) Serial in-parallel out



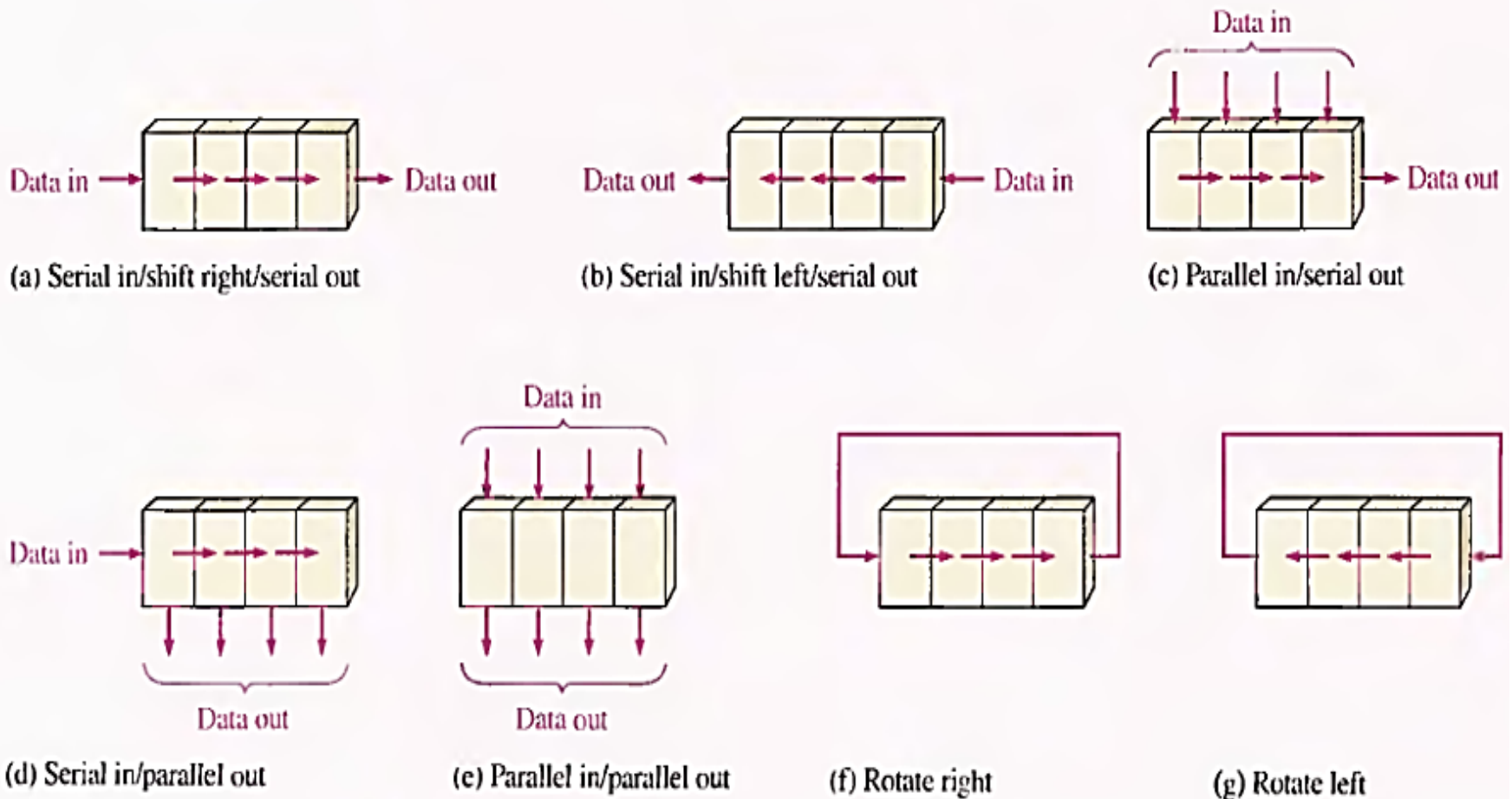
(c) Parallel in-serial out



(d) Serial in-parallel out

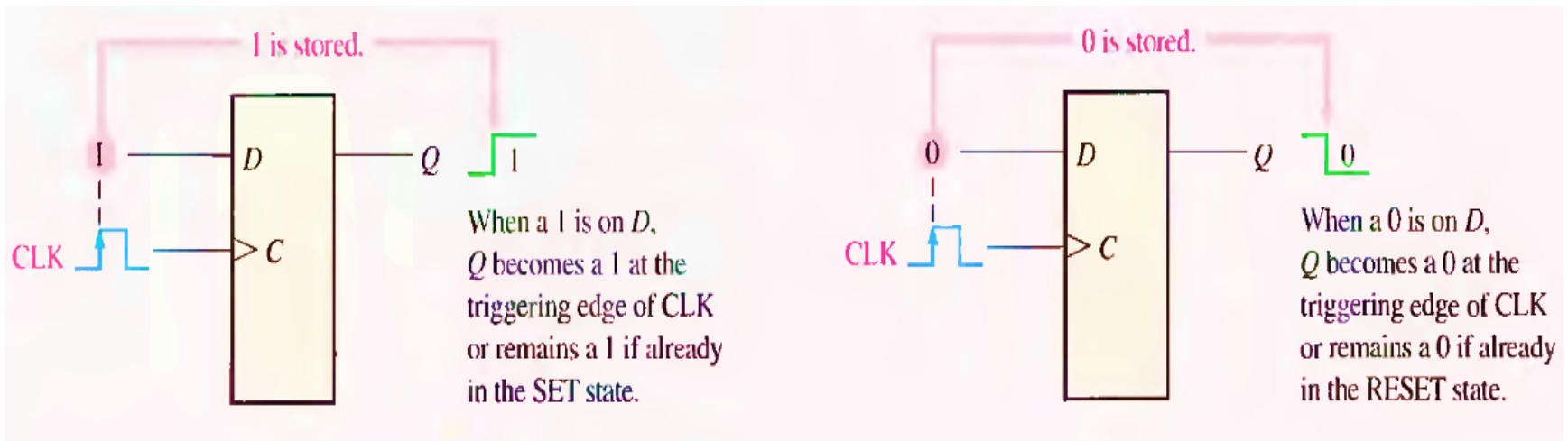
Shift register types

TYPES OF REGISTERS-3



Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

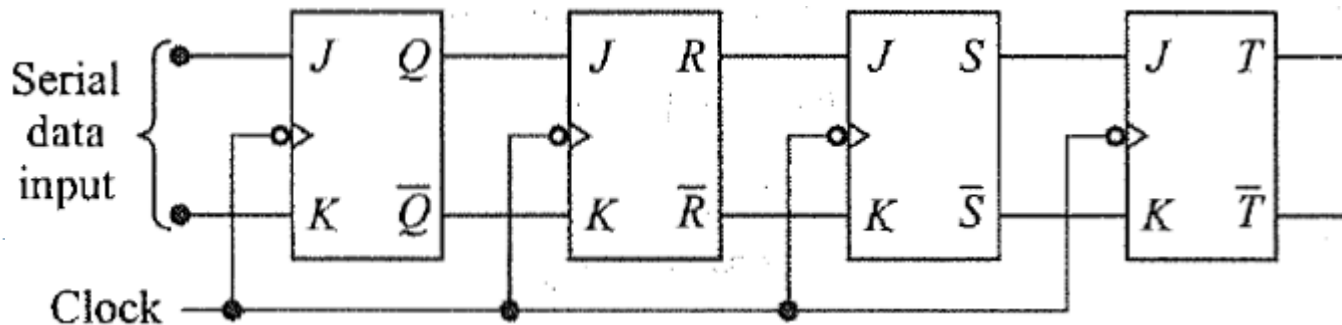
TYPES OF REGISTERS-4



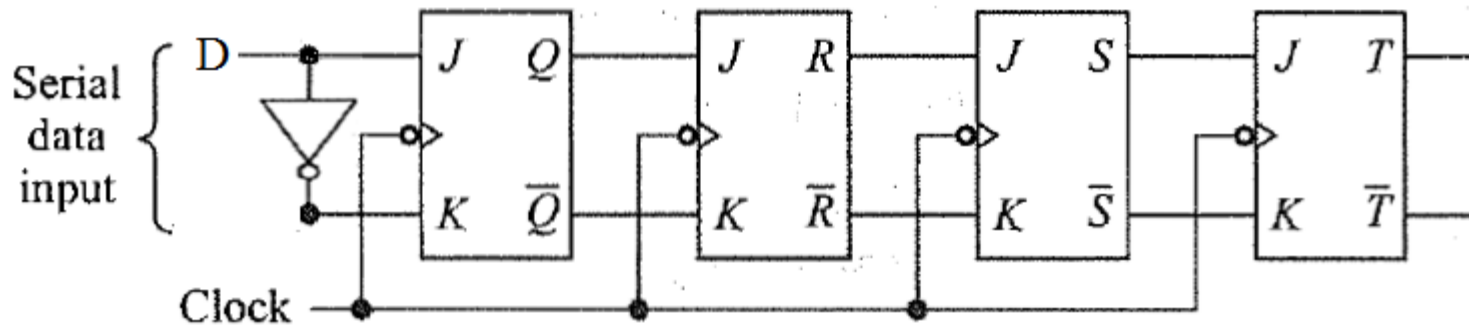
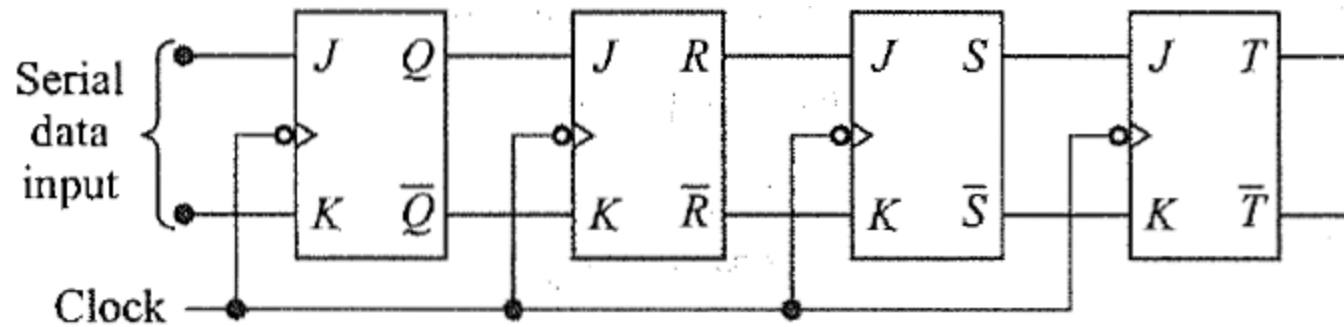
The flip-flop as a storage element.

SERIAL IN-SERIAL OUT-1

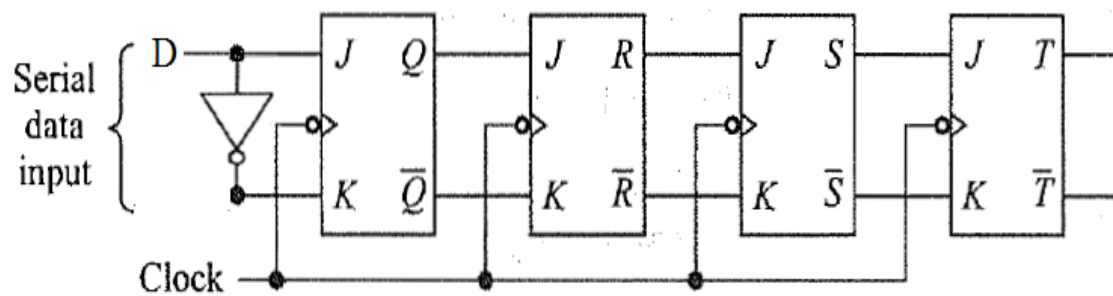
- “ The flip-flops used to construct registers are usually edge-triggered JK , SR or D types.
- “ D flip-flops connected as shown in figure forming 4-bit shift register. A common clock provides trigger at its negative edge to all the flip-flops.
- “ As output of one D flip-flop is connected to input 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 $Q \rightarrow R$, $R \rightarrow S$, $S \rightarrow T$ and *serial data input* is transferred to Q .



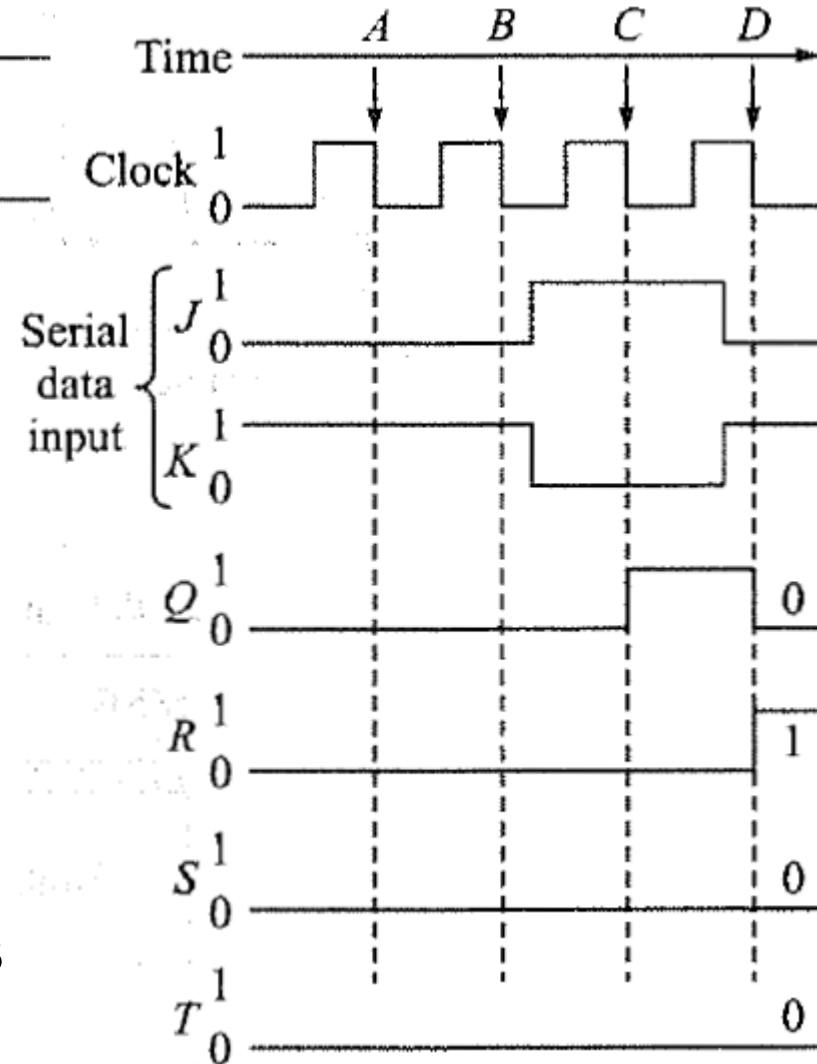
SERIAL IN-SERIAL OUT-1



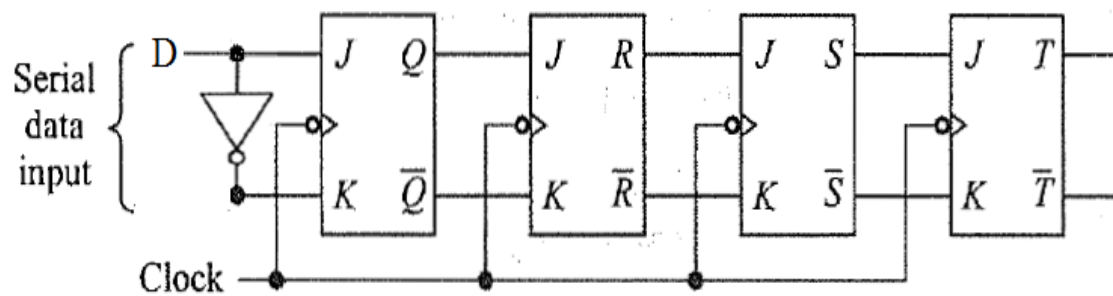
SERIAL IN-SERIAL OUT-2



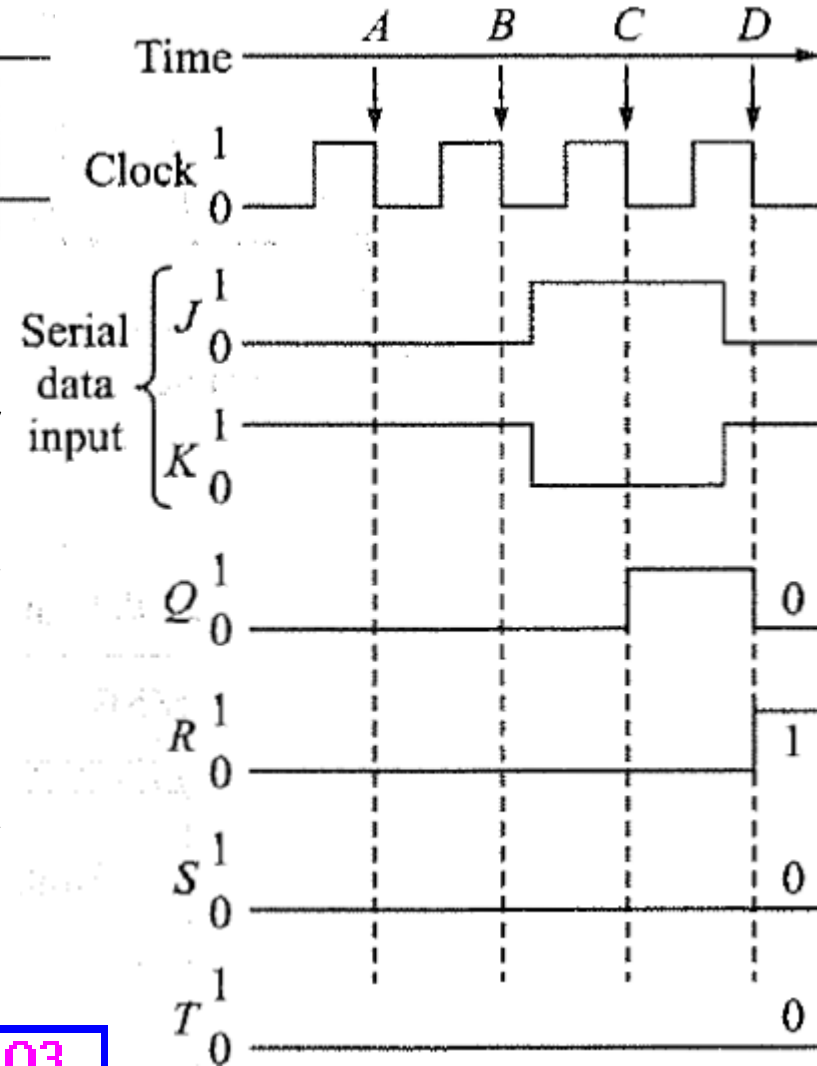
- “ At clock edge A,
- “ $D = 0 \rightarrow Q$, $Q \rightarrow R$, $R \rightarrow S$, $S \rightarrow T$
- “ When clock triggers, these inputs get transferred to corresponding flip-flop outputs simultaneously so that $QRST = 0000$.
- “ Thus at clock trigger, values at $DQRS$ is transferred to $QRST$.



SERIAL IN-SERIAL OUT-3



- “ **At clock edge B**, serial data in $D=0$, i.e. $DQRS = 0000$. So after NT at B, $QRST = 0000$.
- “ Serial data becomes 1 in next clock cycle.
- “ **At clock edge C**, $DQRS = 1000$ and after NT $QRST = 1000$.
- “ Serial data goes to 0 in next clock cycle
- “ **At clock edge D**, $DQRS = 0100$ and after NT $QRST = 0100$.

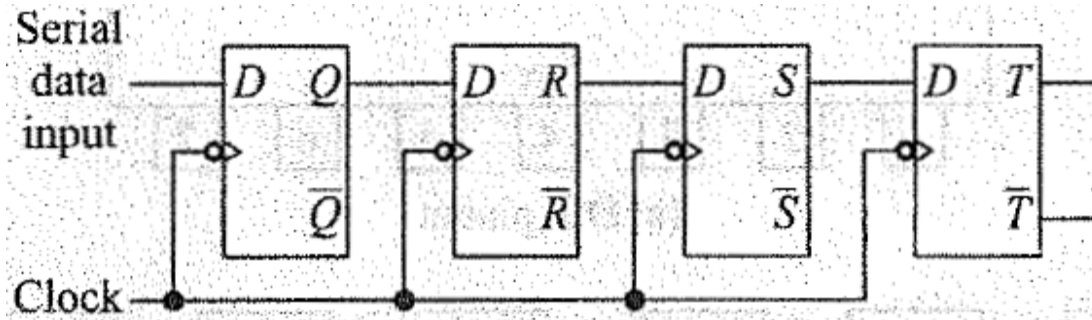


CLEAR
1001

Q0	Q1	Q2	Q3
0	0	0	0

SERIAL IN-SERIAL OUT-4

- “ Example: Show how a number 0100 is entered serially in a 4-bit shift register using D flip-flop. Also write state table.

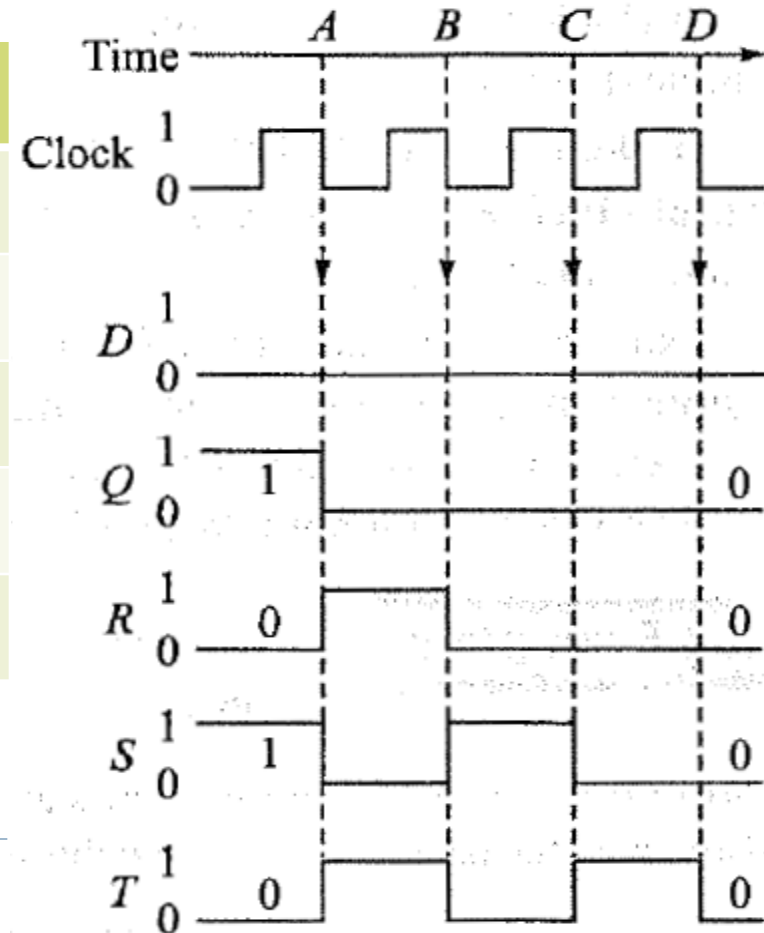


Clock	Serial input	Q	R	S	T
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4		0	0	1	0

SERIAL IN-SERIAL OUT-5

- “ Example: Suppose that it has the 4-bit number $QRST = 1010$ stored in it so draw the waveform

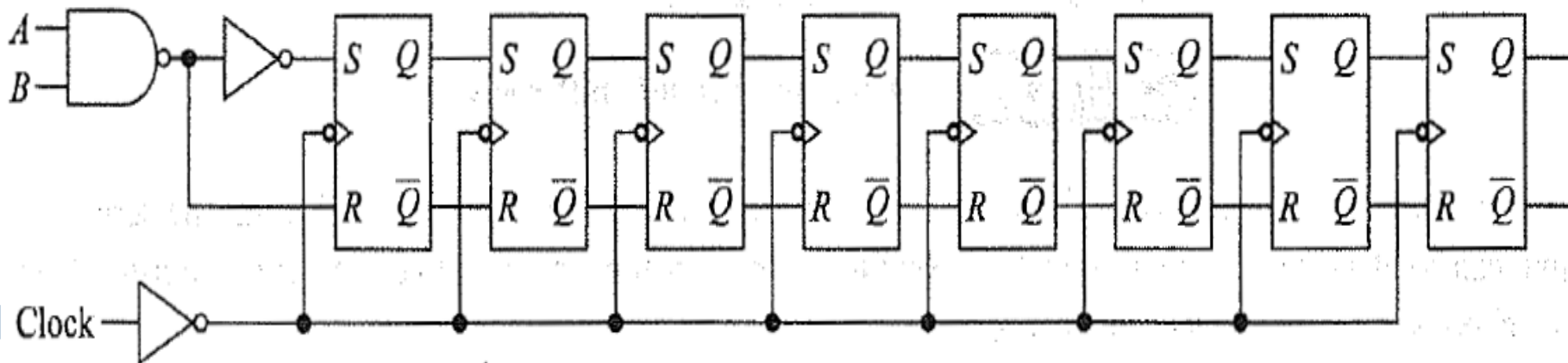
Clk	Serial In	Q	R	S	T
0	0	1	0	1	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4		0	0	0	0



SERIAL IN-SERIAL OUT-6

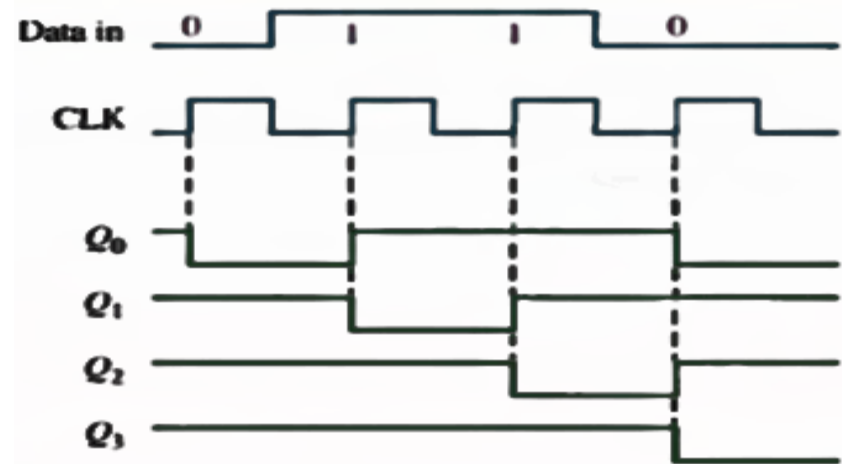
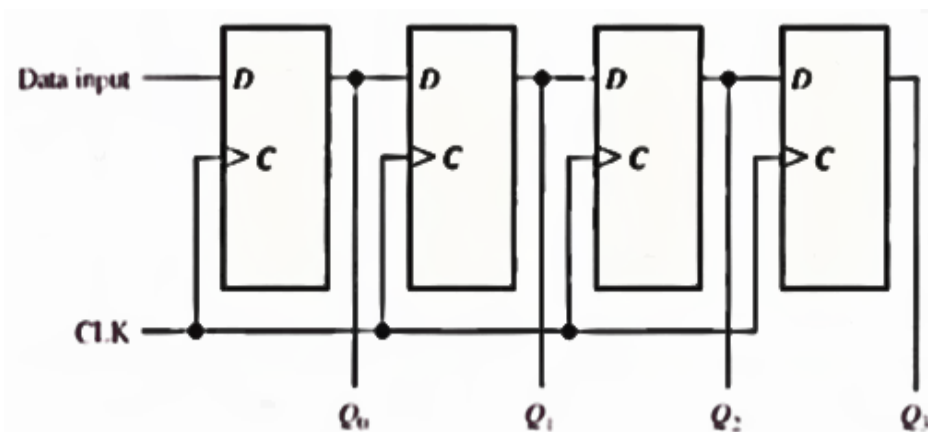
“ 74LS91 8-bit shift register

- “ The data input is applied at either *A* (pin 10) or *B* (pin 12). Notice that a data level at *A* (or *B*) is complemented by the NAND gate and then applied to the *R* input of the first flip-flop.
- “ The same data level is complemented by the NAND gate and then complemented again by the inverter before it appears at the *S* input. So, a 1 at input *A* will set the first flip-flop (in other words, this 1 is shifted into the first flip-flop) on a positive clock transition.
- “ The NAND gate with inputs *A* and *B* simply provides a gating function for the input data stream if desired.
- “ If gating is not desired, simply connect pins 10 and 12 together and apply the input data stream to this connection.



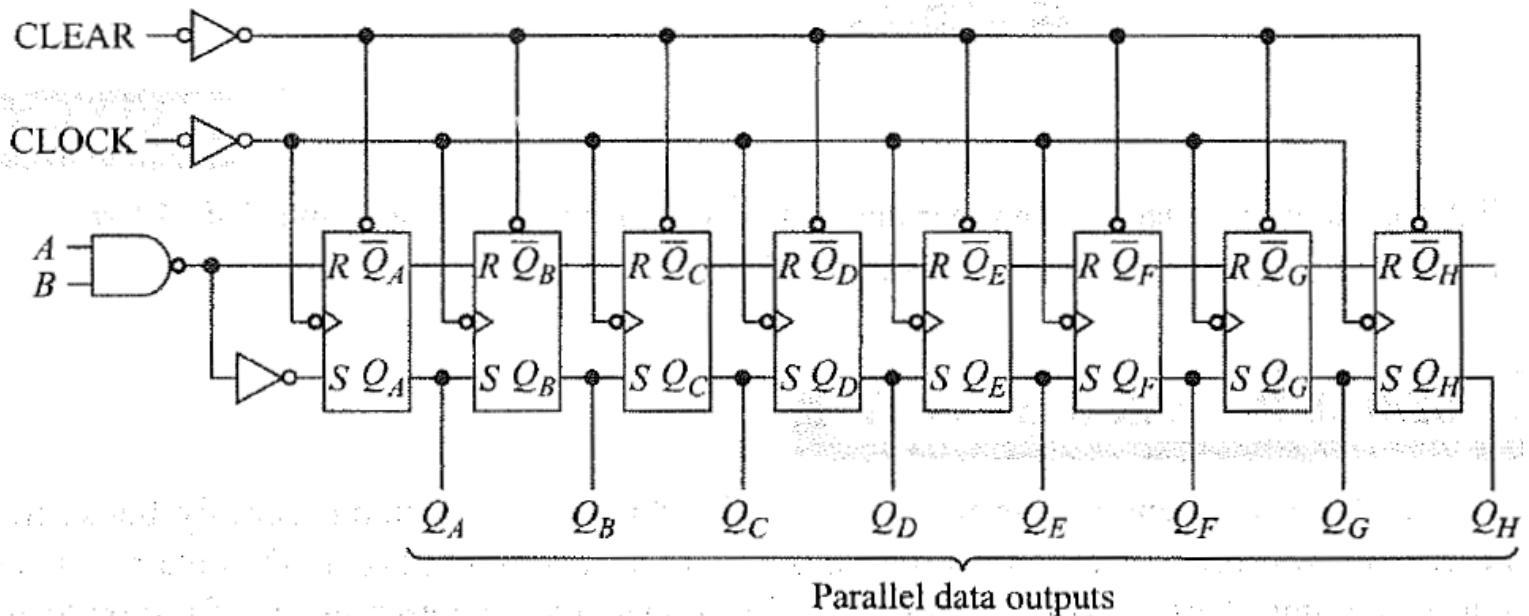
SERIAL IN-PARALLEL OUT-1

- “ Data is shifted in serially, but shifted out in parallel.
- “ In order to shift the data out in parallel, it is simply necessary to have all the data bits available as outputs at the same time.
- “ E.g. 4-bit shift register



SERIAL IN-PARALLEL OUT-1

“ E.g. 8-bit shift register



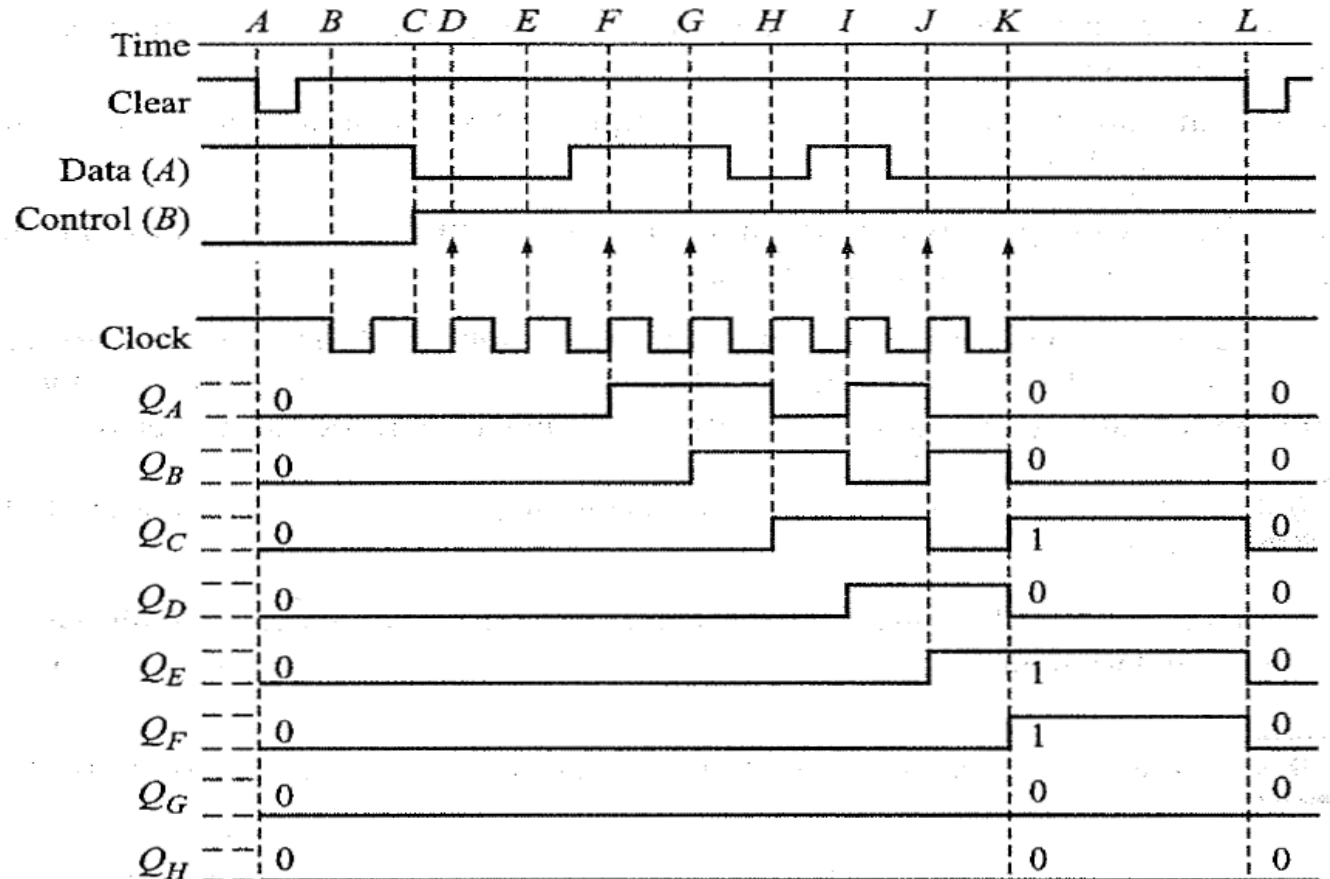
(b) Logic diagram

SERIAL IN-PARALLEL OUT-2

- “ How long will it take to shift an 8-bit number into a 8-bit shift register if the clock is set at 10 MHz?
- “ A minimum of eight clock periods will be required since the data is entered serially. One clock period is 100 ns, so it will require 800 ns minimum.

SERIAL IN-PARALLEL OUT-3

- “ The waveforms shown below, show the typical response of a 54/74164. The serial data is input at *A* (pin 1), while a gating control signal is applied at *B* (pin 2). The first clear pulse occurs at time *A* and simply resets all flip-flops to 0.

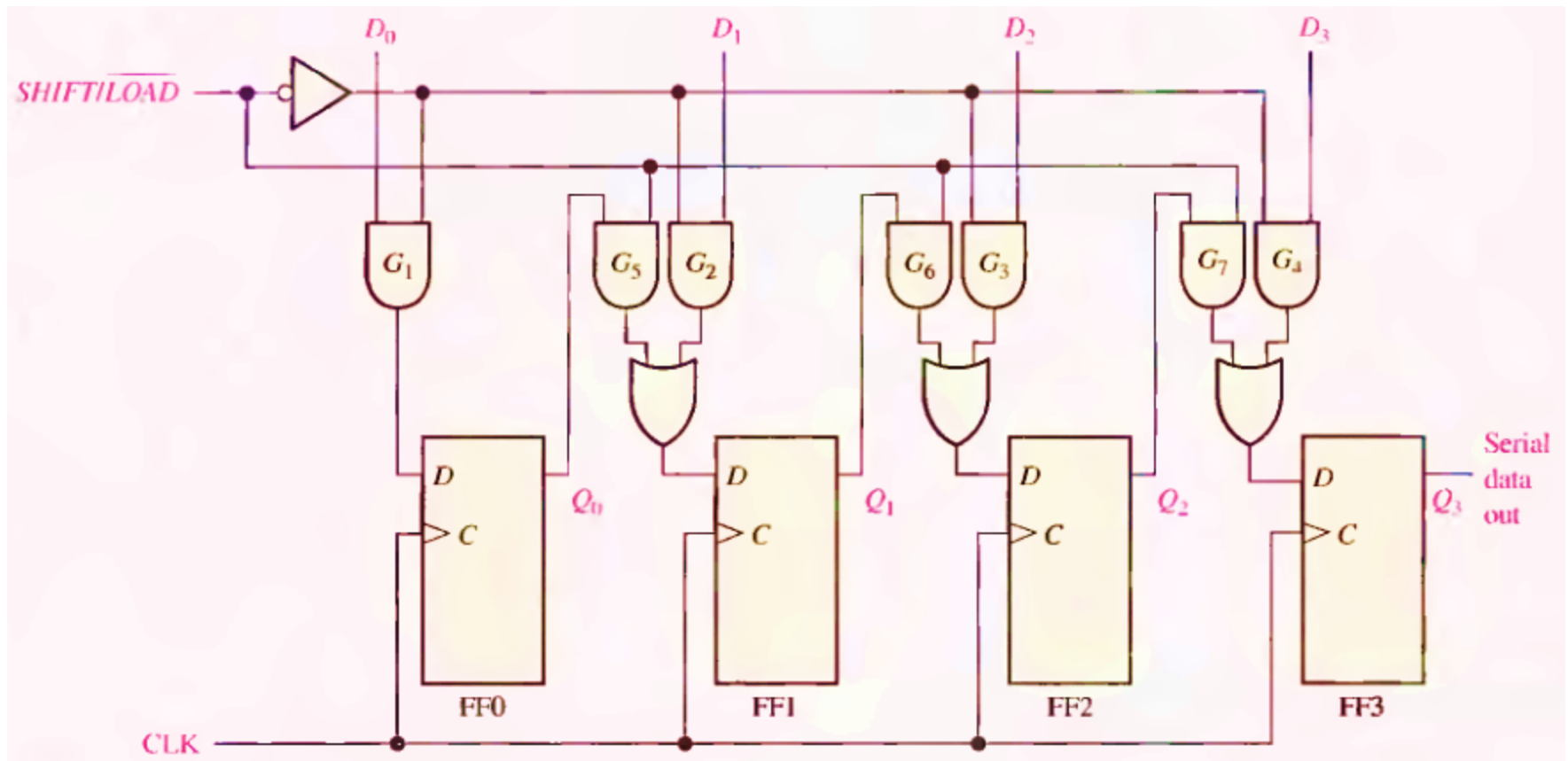


PARALLEL IN-SERIAL OUT-1

- “ For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs.
- “ Next slide illustrates a 4-bit parallel in/serial out shift register and a typical logic symbol.
- “ Notice that there are four data-input lines, D_0 , D_1 , D_2 and D_3 and a SHIFT/LOAD input, which allows four bits of data to load in parallel into the register.
- “ When SHIFT/LOAD is LOW, gates G1 through G4 are enabled, allowing each data bit to be applied to the D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with $D = 1$ will set and those with $D = 0$ will reset. thereby storing all four bits simultaneously.
- “ When SHIFT/LOAD is HIGH, gates G1 through G4 are disabled and gates G5 through G 7 are enabled, allowing the data bits to shift right from one stage to the next.
- “ The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

PARALLEL IN-SERIAL OUT-2

“ 4-bit parallel in/serial out shift register.



PARALLEL IN-SERIAL OUT-3

“ The 54/74166 is an 8-bit shift register.

Inputs						Internal Levels		Outputs
Clear	Shift/load	Clock inhibit	Clock	Serial	Parallel $A \dots H$	Q_A and Q_B		Q_H
L	X	X	X	X	X	L	L	L
H	X	L	L	X	X	Q_{AO}	Q_{BO}	Q_{HO}
H	L	L	\uparrow	X	$a \dots h$	a	b	h
H	H	L	\uparrow	H	X	H	Q_{An}	Q_{Gn}
H	H	L	\uparrow	L	X	L	Q_{An}	Q_{Gn}
H	X	H	\uparrow	X	X	Q_{AO}	Q_{BO}	Q_{HO}

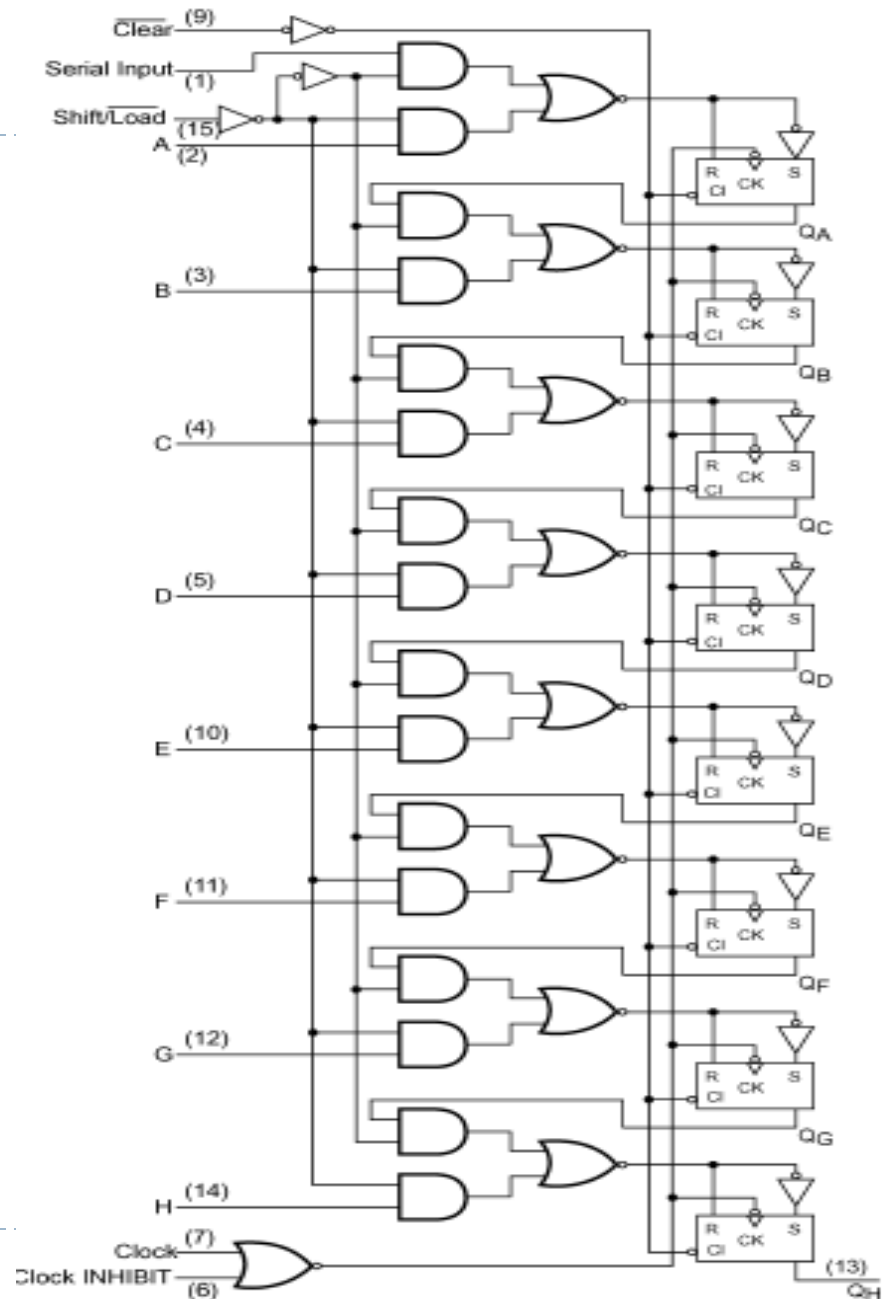
X = Irrelevant, H = High level, L = Low level

\uparrow = Positive transition

$a \dots h$ = Steady state input level at $A \dots H$ respectively

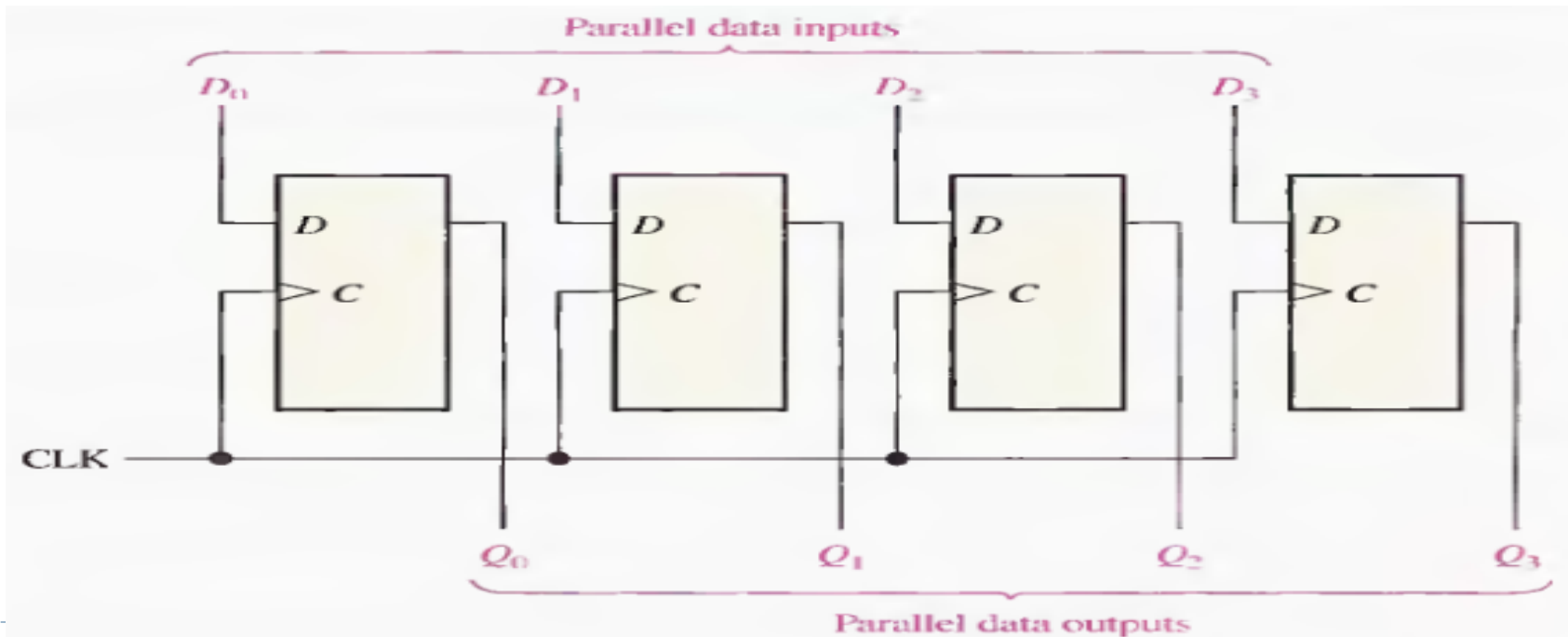
Q_{AO}, Q_{BO} = Level at $Q_A, Q_B \dots$ before steady state

Q_{An}, Q_{Gn} = Level of Q_A or Q_B before most recent transition () \uparrow



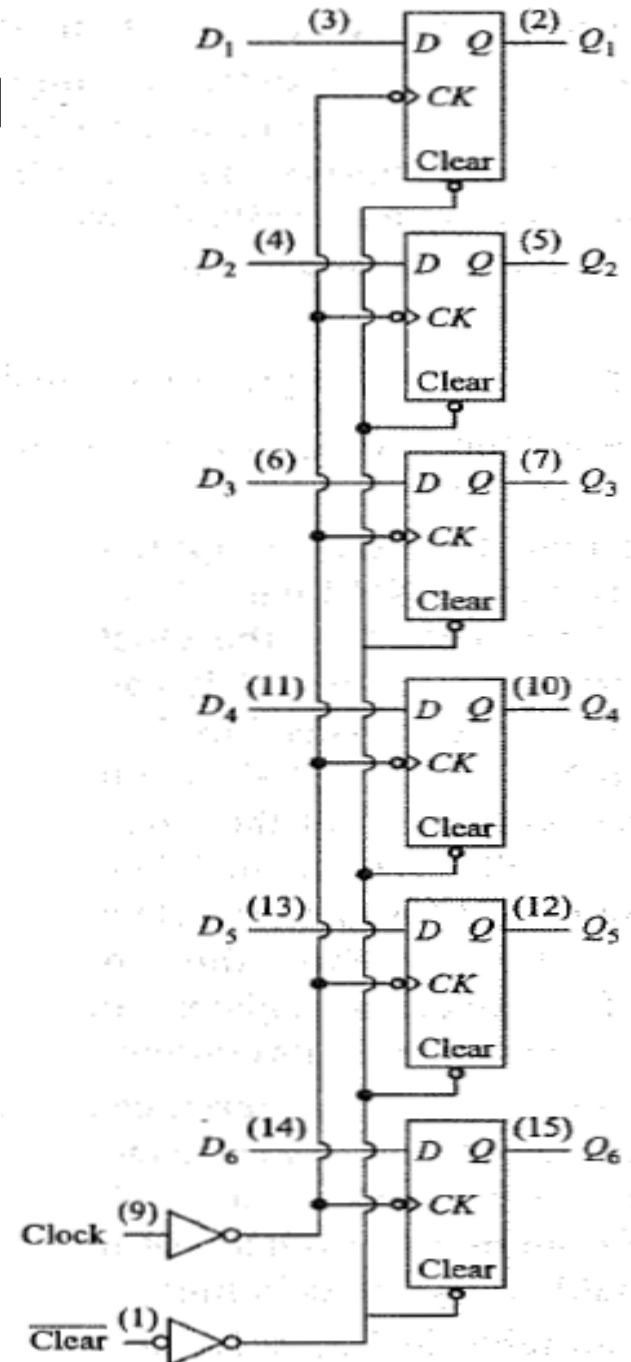
PARALLEL IN- PARALLEL OUT-1

- “ The parallel in/parallel out register employs both methods. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs.



PARALLEL IN- PARALLEL O

- “ The 74174 is an example of a 6-bit parallel in-parallel out register.

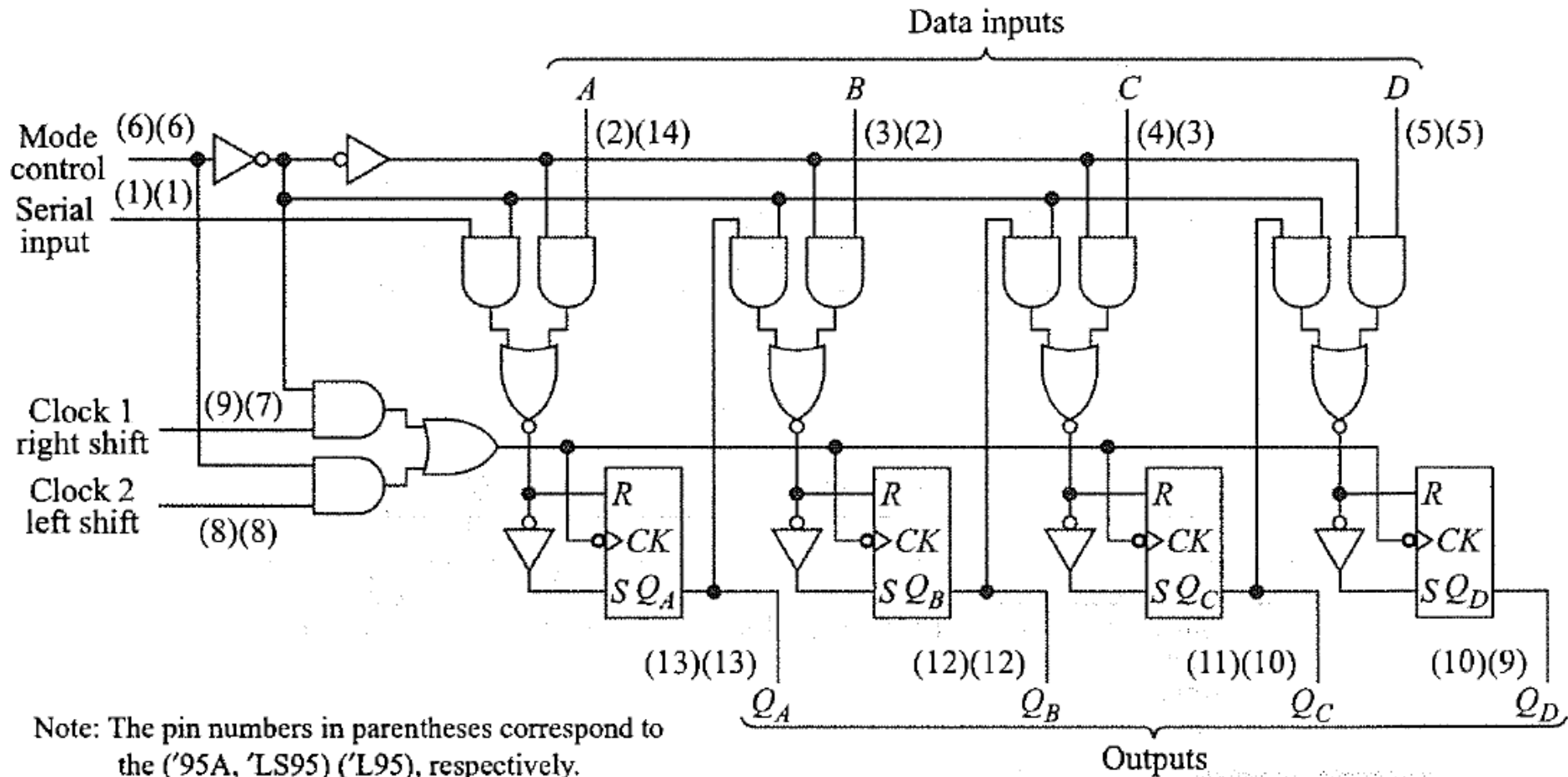


PARALLEL IN- PARALLEL OUT-3

- “ The 74LS174 data sheet gives a setup time of 20 ns and a hold time of 5 ns. What is the minimum required width of the data input levels ($D1 \dots D6$) for the 74LS174?

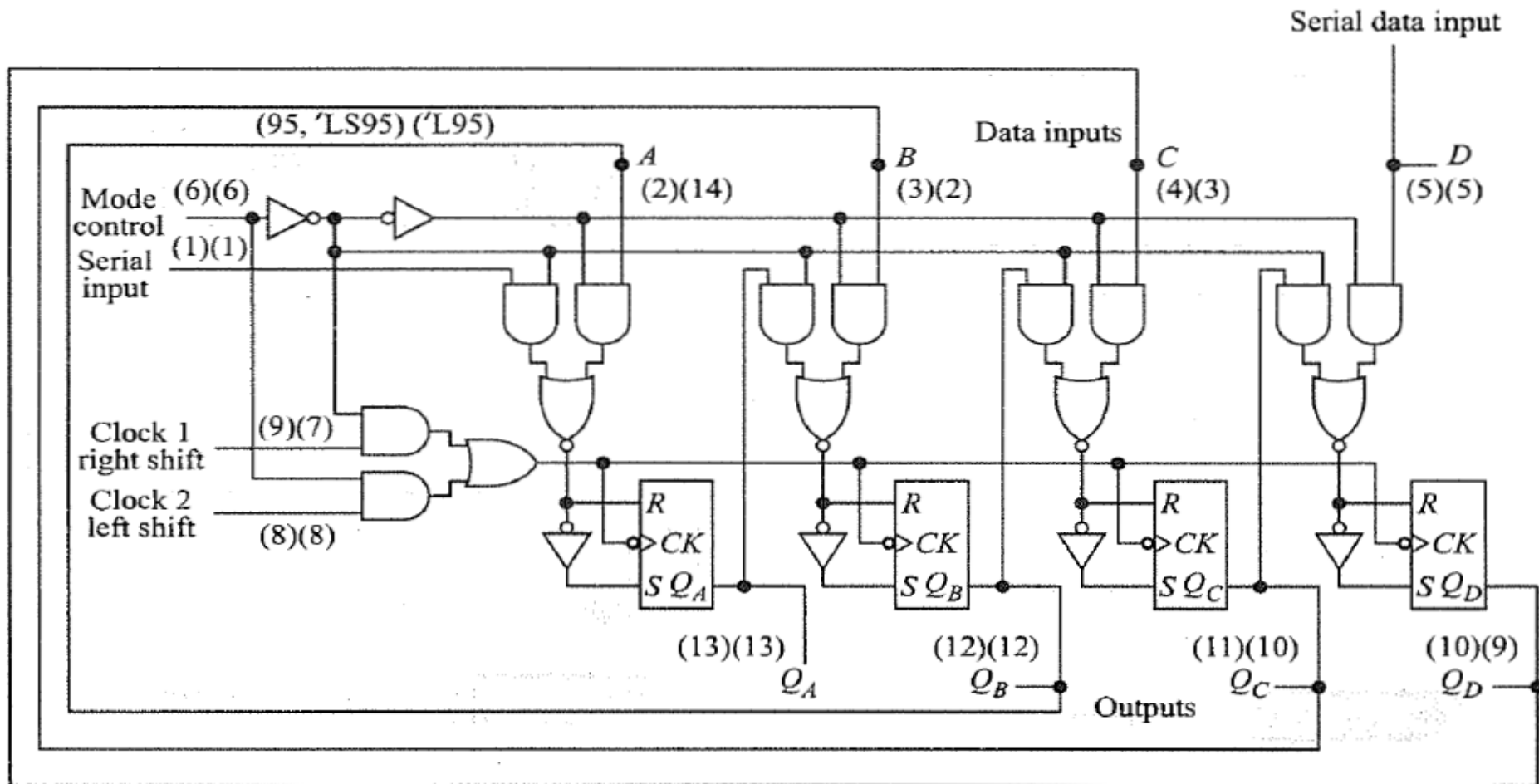
PARALLEL IN- PARALLEL OUT-4

- “ The 5417495A describes it as a 4-bit parallel-access shift register. It also has serial data input and can be used to shift data to the right (from QA toward QB) and in the opposite direction to the left.



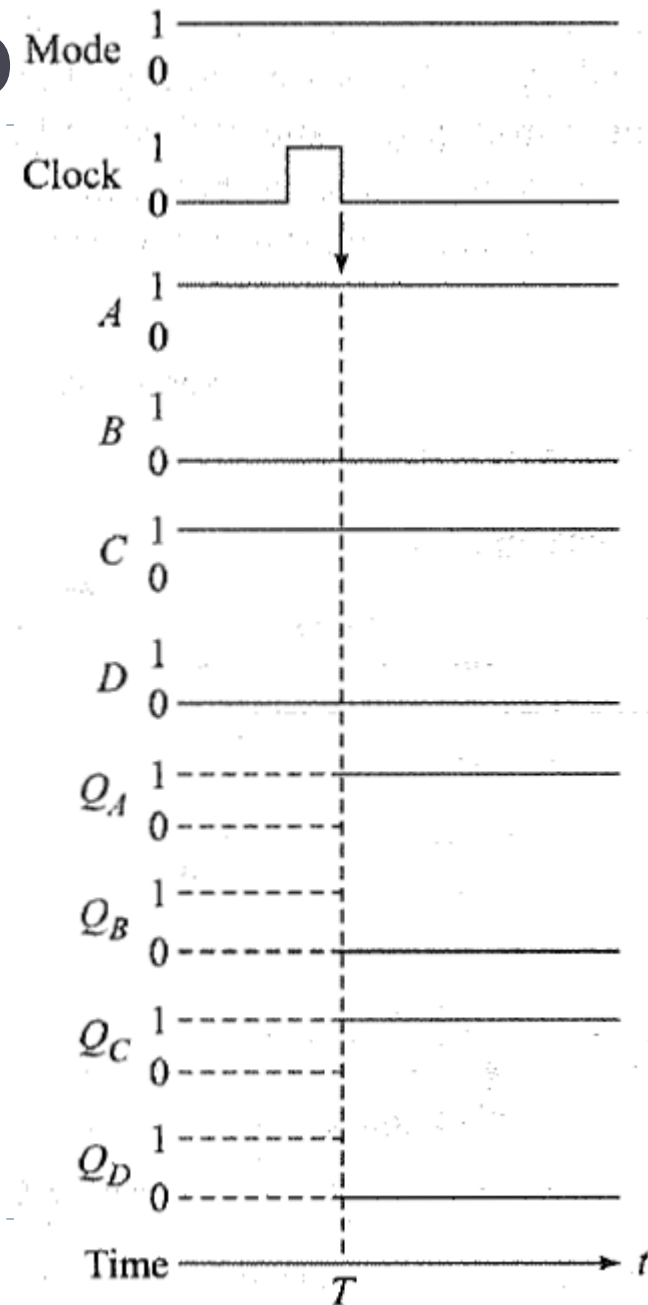
PARALLEL IN- PARALLEL OUT-4

- “ The 5417495A describes it as a 4-bit parallel-access shift register. It also has serial data input and can be used to shift data to the right (from QA toward QB) and in the opposite direction to the left.



PARALLEL IN- PARALLEL O

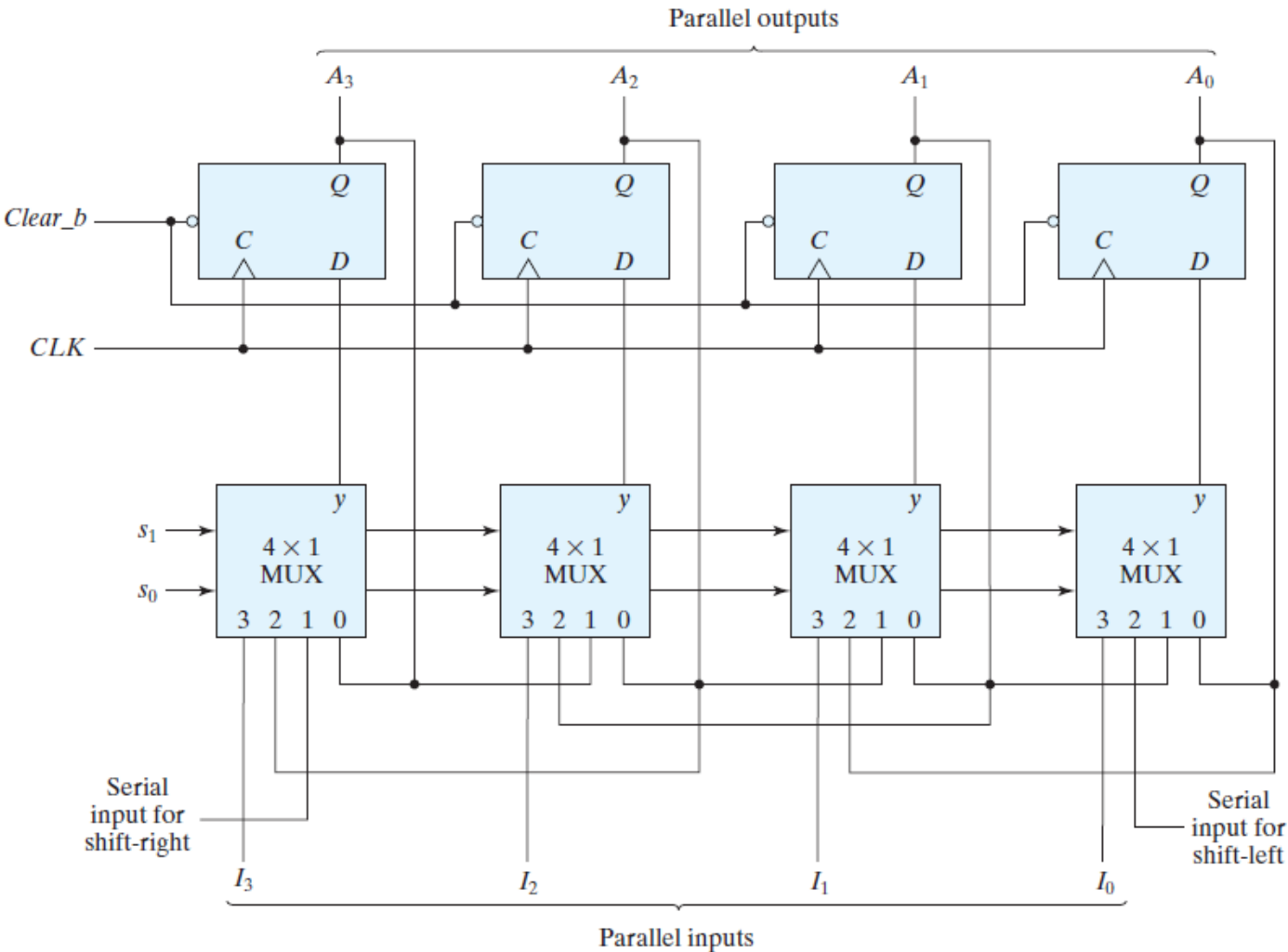
- “ Draw the waveforms you would expect if the 4-bit binary number 1010 were shifted into a 5417495A in parallel.
- “ The mode control line must be high, The data. input line must be stable for more than 10 ns prior to the clock NTs (setup time for the data sheet information). A single clock NT will enter the data. If the clock is stopped after the transition time T , the levels on the input data lines may be changed, However, if the clock is not stopped, the input data line levels must be maintained.



UNIVERSAL SHIFT REGISTER-1

- “ Basic types of shift register ,the following operations are possible-serial in-serial out, serial in-parallel out, parallel in-serial out, and parallel in-parallel out. Serial in or serial out again can be made possible by shifting data in any of the two directions, left shift ($QA \leftarrow QB \leftarrow QC \leftarrow QD \leftarrow \text{Data in}$) and right shift ($\text{Data in} \rightarrow QA \rightarrow QB \rightarrow QC \rightarrow QD$). A universal shift register can perform all the four operations and is also bidirectional in nature.

UNIVERSAL SHIFT REGISTER-2



Mode Control		Register Operation
s_1	s_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

APPLICATIONS Of SHIFT REGISTERS-1

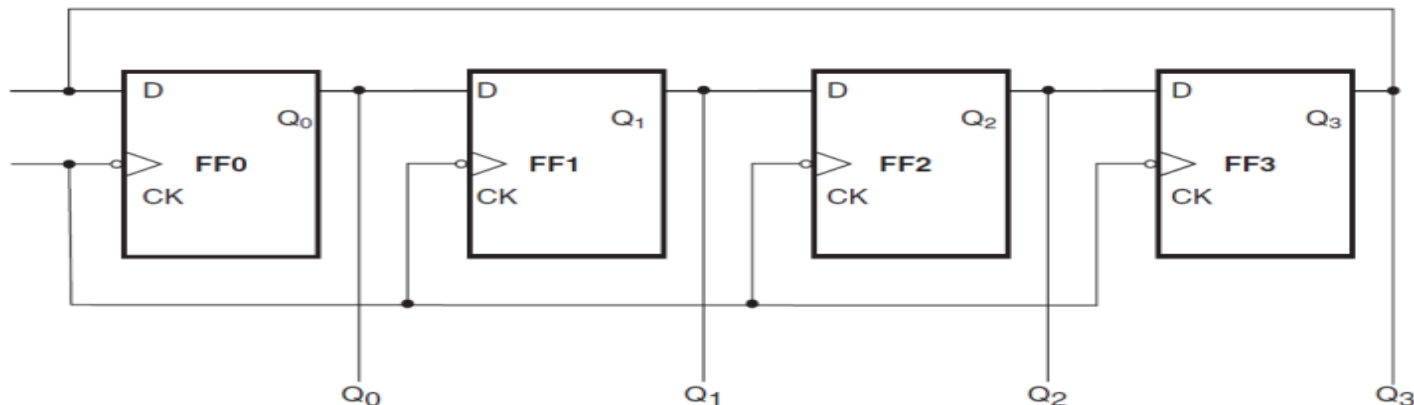
- “ Shift register can be used to count number of pulses entering into a system as ring counter or switched-tail counter. As ring counter it can generate various control signals in a sequential manner.
- “ Shift register can also generate a prescribed sequence repetitively or detect a particular sequence from data input. It can also help in reduction of hardware by converting parallel data feed to serial one.
- “ Serial adder is one such application discussed in this section.

APPLICATIONS Of SHIFT REGISTERS-2

“ Ring Counter

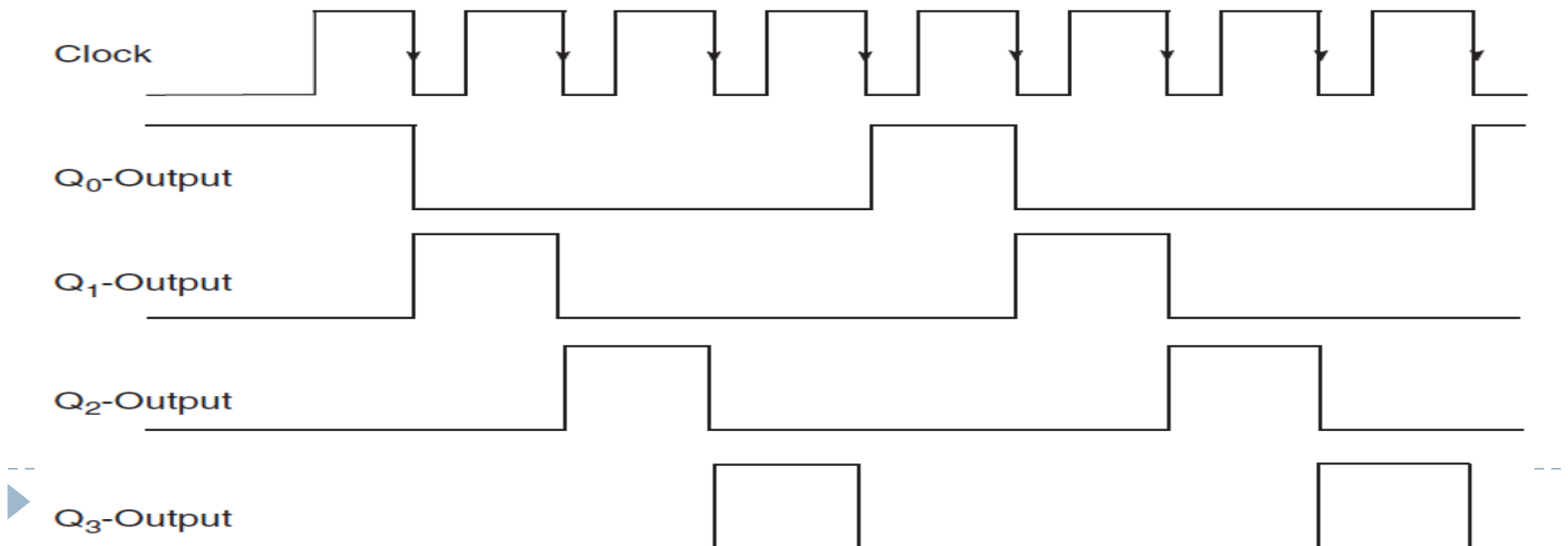
- “ A *ring counter* is obtained from a shift register by directly feeding back the true output of the output flip-flop to the data input terminal of the input flip-flop.
- “ If D flip-flops are being used to construct the shift register, the ring counter, also called a circulating register, can be constructed by feeding back the Q output of the output flip-flop back to the D input of the input flip-flop.

Clock Pulse	Q0	Q1	Q2	Q3
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0
6	0	1	0	0
7	0	0	1	0



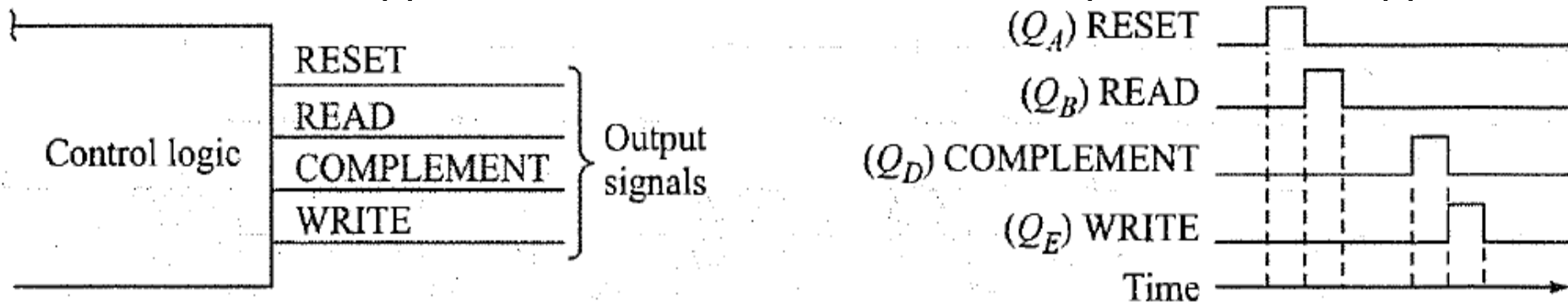
APPLICATIONS OF SHIFT REGISTERS-3

- “ Flip-flop FF0 is initially set to the logic ‘1’ state and all other flip-flops are reset to the logic ‘0’ state.
- “ The counter output is therefore 1000. With the first clock pulse, this ‘1’ gets shifted to the second flip-flop output and the counter output becomes 0100.
- “ Similarly, with the second and third clock pulses, the counter output will become 0010 and 0001. With the fourth clock pulse, the counter output will again become 1000. The count cycle repeats in the subsequent clock pulses.



APPLICATIONS Of SHIFT REGISTERS-4

- “ Waveforms of this type are frequently used in the control section of a digital system. They are ideal for controlling events that must occur in a strict time sequence—that is, event *A*, then event *B*, then *C*, and so on.
- “ For instance, the logic diagram in figure shows how to generate RESET, READ, COMPLEMENT, and WRITE (a fictitious set of control signals) as a set of control pulses that occur one after the other sequentially.
- “ There is, however, a problem with such ring counters. In order to produce the waveforms shown in figure, the counter should have one, and only one, 1 in it. The chances of this occurring naturally when power is first applied are very remote indeed. If the flip-flops should all happen to be in the reset state when power is first applied

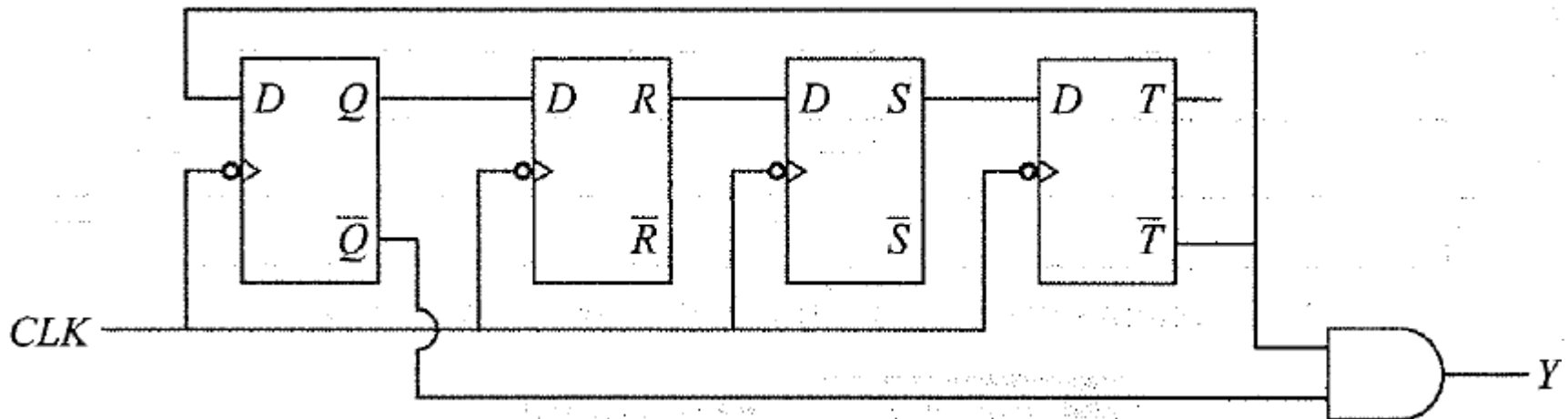


APPLICATIONS Of SHIFT REGISTERS-5

“ Switched-Tail Counter or Johnson Counter

- “ In Ring counter what happens if non-inverting output of the first flip-flop is fed back to first flip flop of the shift register. If we instead feed inverting output back (or switch the tail) as shown in figure for a 4-bit shift register we get *switched tail counter*, also known as *twisted tail counter* or *Johnson counter*.

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



APPLICATIONS Of SHIFT REGISTERS-6

- “ From state table similar. Assume all the flip-flops are cleared in the beginning. Then all the flip-flop inputs have 0 except the first one, *serial data* in which is complement of the last flip-flop, i.e. 1.
- “ When clock trigger occurs flip-flop stores $QRST$ as 1 000. This makes 1100 at the input of $QRST$ when the next clock trigger comes and that gets transferred to output at NT .
- “ Note that output $Y = Q'T'$ and state of the circuit repeats every eighth clock cycle. Thus this 4-bit shift register circuit can count 8 clock pulses or called modulo-8 counter.
- “ Following above logic and preparing state table for any N -bit shift register we see switched-tail configuration can count up to $2N$ number of clock pulse and gives modulo- $2N$ counter. The output Y , derived similarly by AND operation of first and last flip-flop inverting outputs gives a logic high at every $2N$ -th clock cycle.
- “ This two-input AND gate which decodes states repeating in the memory units to generate output that signals counting of a given number of clock pulses is called decoding gate.

APPLICATIONS Of SHIFT REGISTERS-7

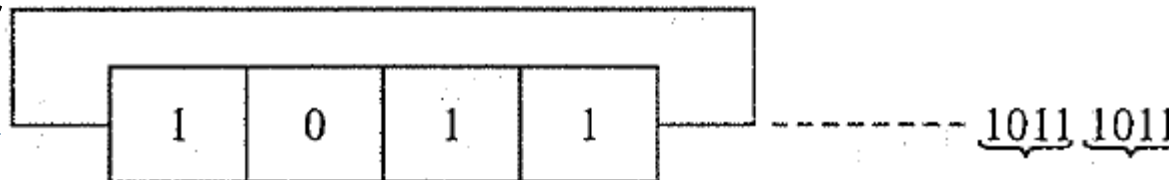
“ **Sequence Generator and Sequence Detector**

- “ Sequence generator is useful in generating a sequence pattern repetitively.
- “ It may be the synchronizing hit 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.

APPLICATIONS Of SHIFT REGISTERS-8

“ Sequence Generator

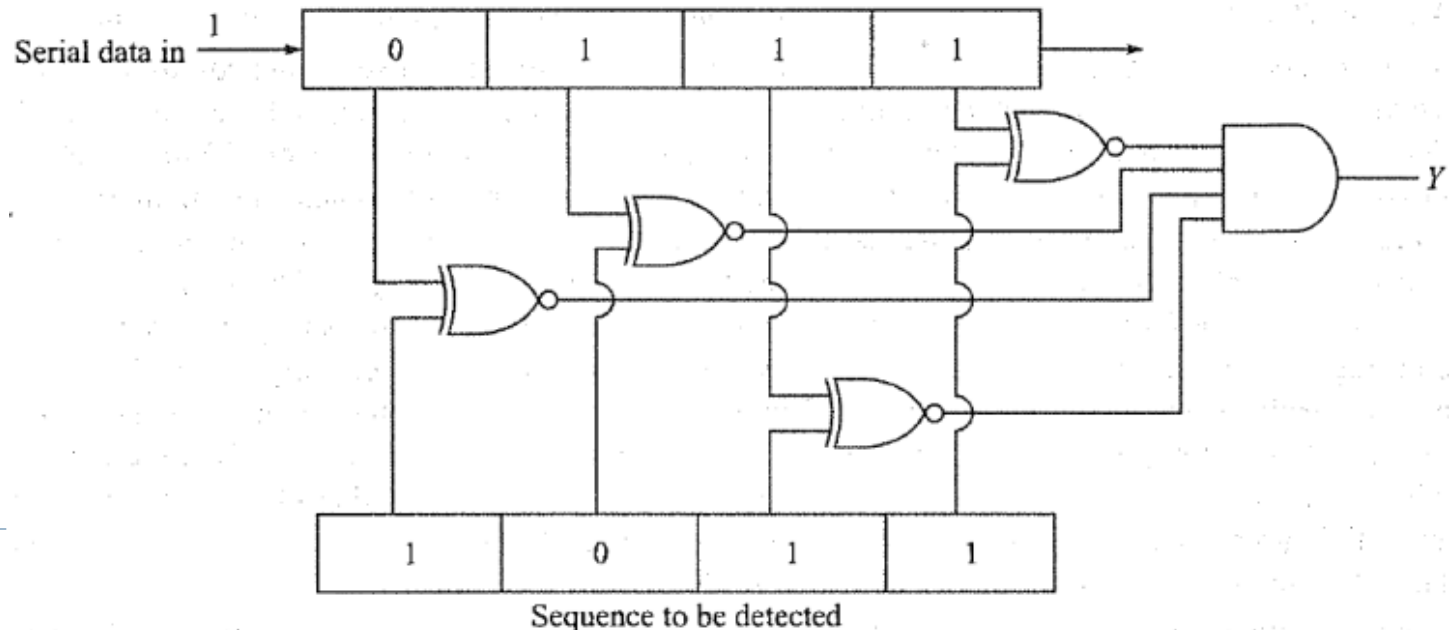
- “ Figure gives the basic block diagram of a sequence generator where shift register is presented as pipe full of data and each flip-flop represents one compartment of it.
- “ The leftmost flip-flop is connected to serial data in and rightmost provides serial data out. The clock is implied and data transfer takes place only when a clock trigger arrives.
- “ Note that the shift register is connected like a ring counter and with triggering of clock the binary word stored in the clock comes out sequentially from serial out but does not get lost as it is fed back as serial in to fill the register all over again. Sequence generated for binary word 1011 is shown in the figure and for any n -bit long sequence to be generated for this configuration we need to store the sequence in an n -bit shift register



APPLICATIONS Of SHIFT REGISTERS-9

“ Sequence detector

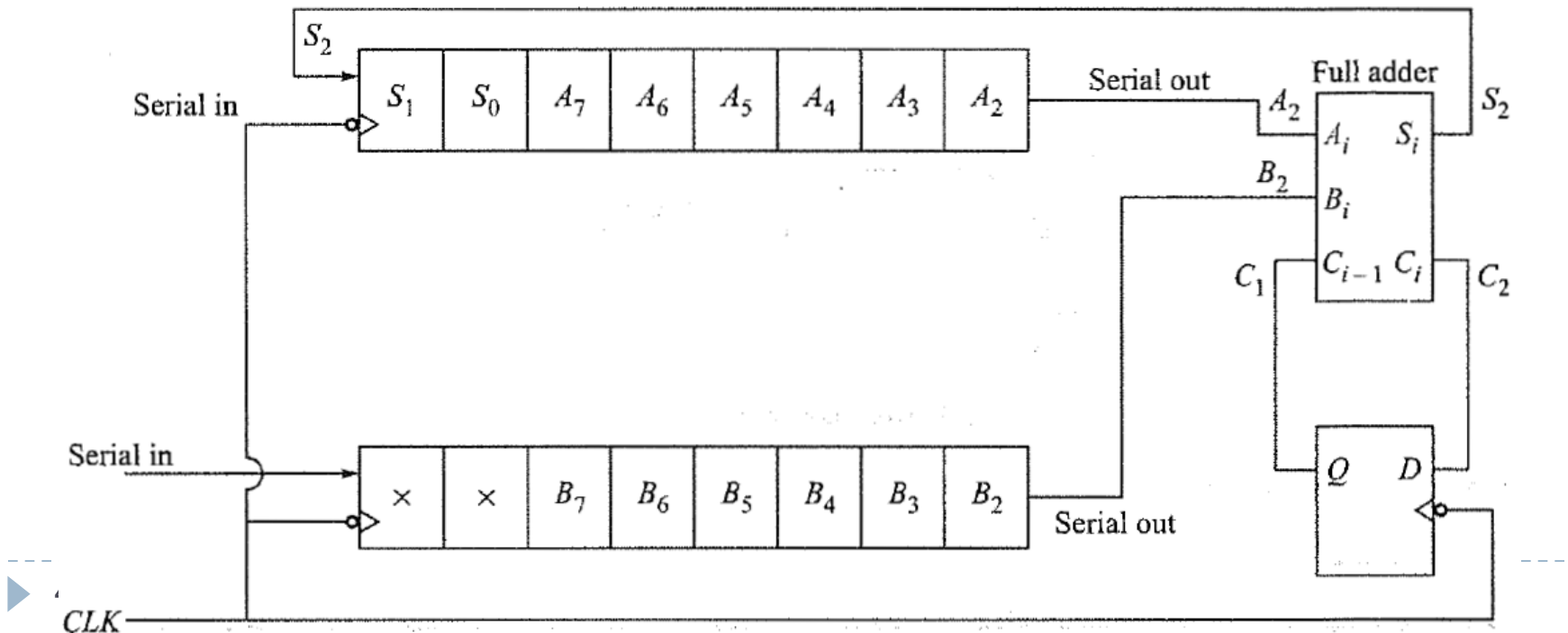
- “ The circuit that can detect a 4-bit binary sequence is shown in figure. It has one register to store the binary word we want to detect from the data stream. Input data stream enters a shift register as serial data in and leaves as serial out.
- “ At every clocking instant, bit-wise comparisons of these two registers are done through Ex-NOR gate as shown in the figure. Two input Ex-NOR gives logic high when both inputs are low or both of them are high, i.e. when both are equal. The final output is taken from a four input AND gate, which becomes 1 only when all its inputs are 1, i.e. all



APPLICATIONS Of SHIFT REGISTERS-10

“ Serial Adder

- “ For 8 bit full adder (FA) circuit need 8 FA units. There the addition is done in parallel. Using shift register we can convert this parallel addition to serial one and reduce number of FA units to only one. The benefit of this technique is more pronounced if the hardware unit that's needed to be used in parallel is very costly.



APPLICATIONS Of SHIFT REGISTERS-11

- “ The LSBs of two numbers (A_0 and B_0) appearing at serial out of respective registers are added by FA during 1st clock cycle and generate sum (S_0) and carry (C_0). S_0 is available at serial data input of register A and C_0 at input of D flip-flop.
- “ At NT of clock shift registers shift its content to right by one unit. S_0 becomes MSB of A and C_0 appears at D flip-flop output. Therefore in the second clock cycle FA is fed by second bit (A_1 and B_1) of two numbers and previous carry (C_0).
- “ In second clock cycle, S_1 and C_1 are generated and made available at serial data in of A register and input of D flip-flop respectively. At NT of clock S_1 becomes MSB of A and S_0 occupies next position. A_2 and B_2 now appear at FA data input and carry input is C_1 .
- “ In 3rd clock cycle, S_2 and C_2 are generated and they get transferred similarly to register and flip-flop. This process goes on and is stopped by inhibiting the clock after 8 clock cycles. At that time shift register A stores the sum bits, S_7 in leftmost (MSB) position and S_0 in rightmost (LSB) position. The final carry is available at D flip-flop output.
- “ The limitation of this scheme is that the final addition result is delayed by eight clock cycles. In parallel adder the result is obtained almost instantaneously, after nanosecond order propagation delay of combinatorial circuit. However, using a high frequency clock the delay factor can be reduced considerably.

REGISTER IMPLEMENTATION IN HDL-1

The PIPO, When Clear is activated (active LOW) all 6 outputs of Q are reset.

```
module regpipo
(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;
//Q stores 6 binary 0
else Q=D;
endmodule
```

Shift right register , where T is the final output and Q, R, S are internal outputs.

```
module SRreg (D, Clock,
T);
input Clock, D;
output T;
reg T;
reg Q,R,S;
always @ (negedge Clock)
begin
    Q<=D;
    R<=Q;
    S<=R;
    T<=S;
end
endmodule
```

SIPO

```
module SR2(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
```

REGISTER IMPLEMENTATION IN HDL-2

- “ Assignment operator `<=` within **always** block which unlike `=` operator executes all associated statements concurrently.
- “ Assignment operator must start with **begin**

REGISTER IMPLEMENTATION IN HDL-3

- “ Write Verilog code for switched tail counter

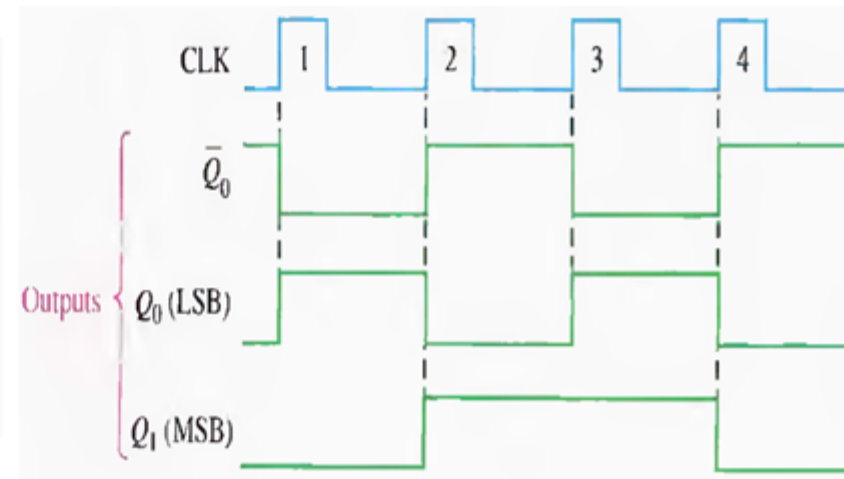
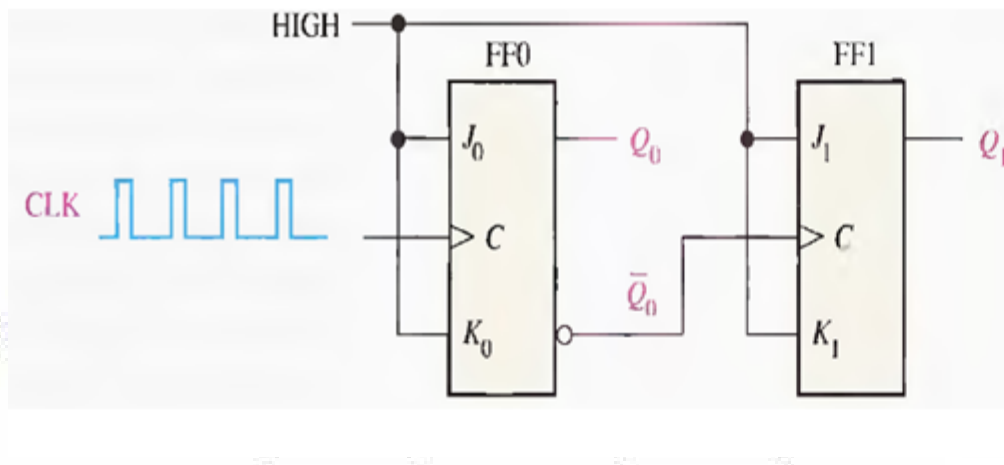
```
module STC(Clock,Clear,Y);    //Switched Tail Counter
input Clock, Clear;
output Y;
reg Q,R,S,T;                //internal outputs of flip-flops
assign Y= (~Q)&(~T);
always@ (negedge Clock)
begin
  if (~Clear) Q=6'b0;        //Q stores 6
  else
    begin
      Q <= ~T;                //Tail is switched and connected to input
      R <= Q;
      S <= R;
      T <= S;
    end
  end
endmodule
```

COUNTER INTRODUCTION

- “ A counter driven by a clock can be used to count the number of clock cycles.
- “ Since the clock pulses occur at known intervals, the counter can be used as an instrument for measuring time and therefore period or frequency.
- “ There are basically two different types of counters- synchronous and asynchronous.
- “ Serial, or asynchronous counter is defined as each flip-flop is triggered by the previous flip-flop, and thus the counter has a cumulative settling time.
- “ An increase in speed of operation can be achieved by use of a parallel or synchronous counter. Here, every flip-flop is triggered by the clock (in synchronism), and thus settling time is simply equal to the delay time of a single flip-flop.
- “ Serial and parallel counters are used in combination to compromise between speed of operation and hardware count.

ASYNCHRONOUS COUNTERS-1

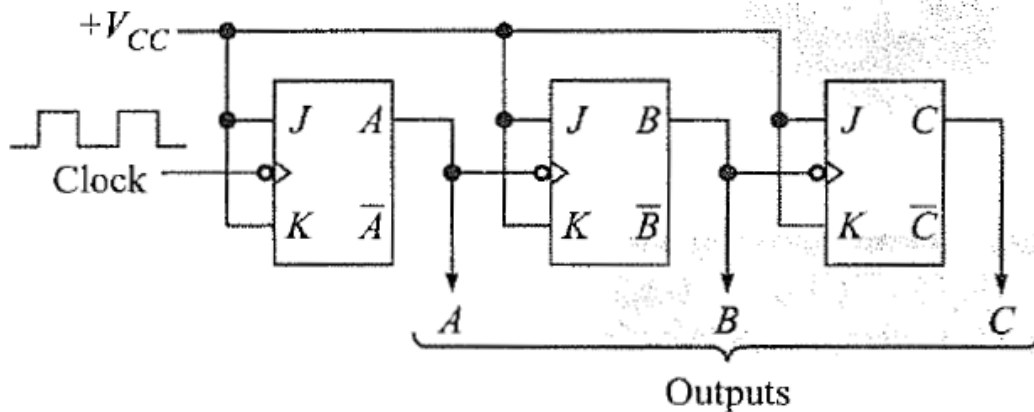
- The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time.
- An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.
- **A 2-Bit Asynchronous Binary Counter**
- Figure shows a 2-bit counter connected for asynchronous operation. Notice that the clock (CLK) is applied to the clock input (C) of only the first flip-flop, FF0, which is always the least significant bit (LSB). The second flip-flop, FF1, is triggered by the \bar{Q}_0 output of FF0. FF0 changes state at the positive-going edge of each clock pulse, but FF1 changes only when triggered by a negative-going transition of the Q_0 output of FF0.



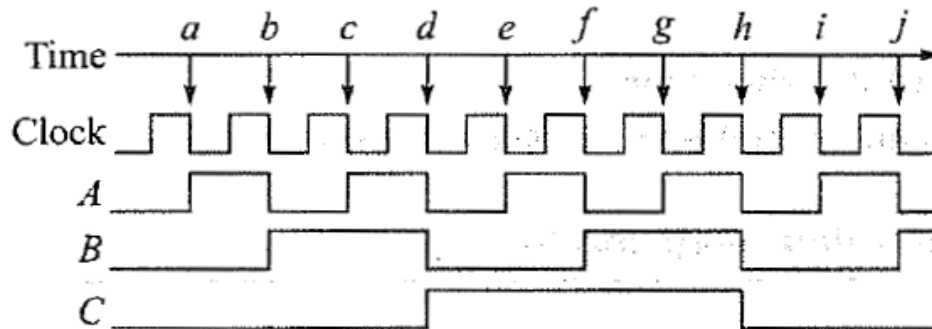
ASYNCHRONOUS COUNTERS-2

- “ **Ripple Counters** (Up Counter)
- “ Figure shows (Next slide) three negative edge- triggered, *JK* flip-flops connected in cascade.
- “ The system clock, a square wave, drives flip-flop *A*. The output of *A* drives *B*, and the output of *B* drives flip-flop *C*. All the *J* and *K* inputs are tied to $+V_{CC}$. This means that each flip-flop will change state (toggle) with a negative transition at its clock input.
- “ When the output of a flip-flop is used as the clock input for the next flip-flop, we call the counter a *ripple counter*, or *asynchronous counter*.
- “ The *A* flip-flop must change state before it can trigger the *B* flip-flop, and the *B* flip-flop has to change state before it can trigger the *C* flip-flop.
- “ If each flip-flop in this three-flip-flop counter has a propagation delay time of 10 ns, the overall propagation delay time for the counter is 30 ns.

ASYNCHRONOUS COUNTERS-3



(a) Three-bit binary ripple counter



(b) Waveforms

Negative clock transitions	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

(c) Truth table

ASYNCHRONOUS COUNTERS-4

- “ The waveform at the output of flip-flop *A* is one-half the clock frequency.
- “ The waveform at the output of flip-flop *B* is one-half the frequency of *A* and one-fourth the clock frequency.
- “ The frequency of the waveform at *C* is one-half that at *B*, but it is only one-eighth the clock frequency.
- “ What is the clock frequency, if the period of the waveform at *C* is 24 μs ?
- “ Since there are eight clock cycles in one cycle of *C*, the period of the clock must be $24/8 = 3 \mu\text{s}$. The clock frequency must then be $1/(3 \times 10^{-6}) = 333 \text{ kHz}$.

ASYNCHRONOUS COUNTERS-5

- “ A binary ripple counter counts in a straight binary sequence, a counter having n flip-flops will have 2^n output conditions.
- “ For instance, the three-flip-flop counter just discussed has $2^3 = 8$ output conditions (000 through 111). Five flip-flops would have $2^5 = 32$ output conditions (00000 through 11111), and so on.
- “ The largest binary number that can be represented by n cascaded flip-flops has a decimal equivalent of 2^{n-1} .
- “ For example, the three-flip-flop counter reaches a maximum decimal number of 2^{3-1} .
- “ The maximum decimal number for five flip-flops is $2^{5-1} = 31$, while six flip-flops have a maximum count of 63.
- “ A three-flip-flop counter is often referred to as a modulus-8 (or mod-8) counter since it has eight states. Similarly, a four-flip-flop counter is a mod-16 counter, and a six-flip-flop counter is a mod-64 counter.
- “ The *modulus* of a counter is the total number of states through which the counter can progress.

ASYNCHRONOUS COUNTERS-6

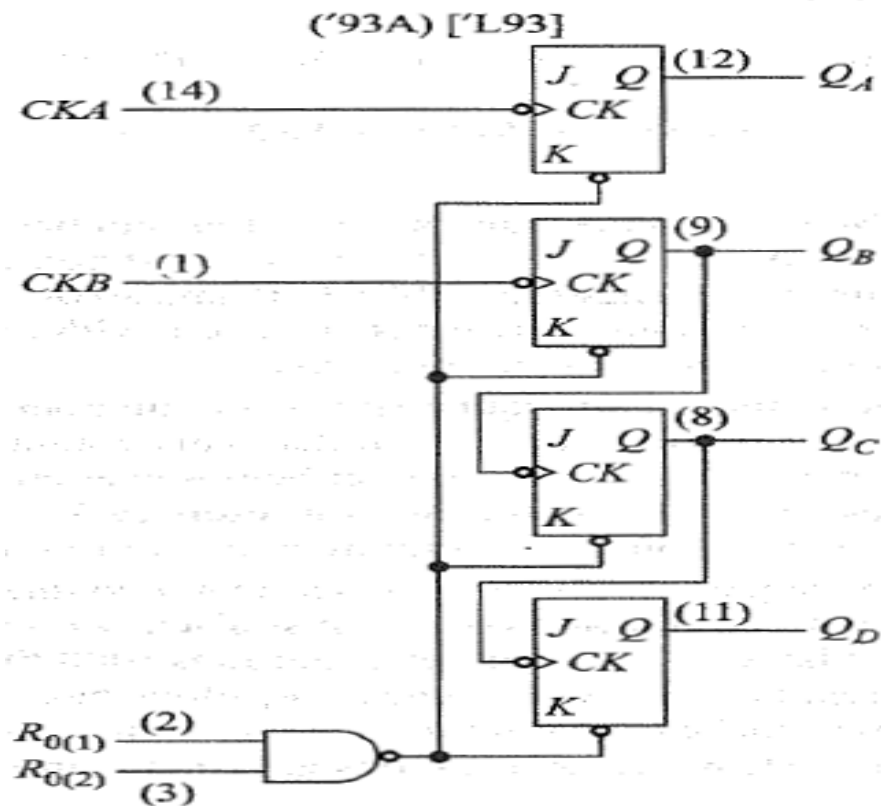
- “ How many flip-flops are required to construct a mod-128 counter? A mod-32? What is the largest decimal number that can be stored in a mod-64 counter?
- “ A mod-128 counter must have seven flip-flops, since $2^7 = 128$.
- “ Five flip-flops are needed to construct a mod-32 counter.
- “ The largest decimal number that can be stored in a six-flop flip counter (mod-64) is $111111 = 63$.
- “ Note carefully the difference between the modulus (total number of states) and the maximum decimal number.

ASYNCHRONOUS COUNTERS-8

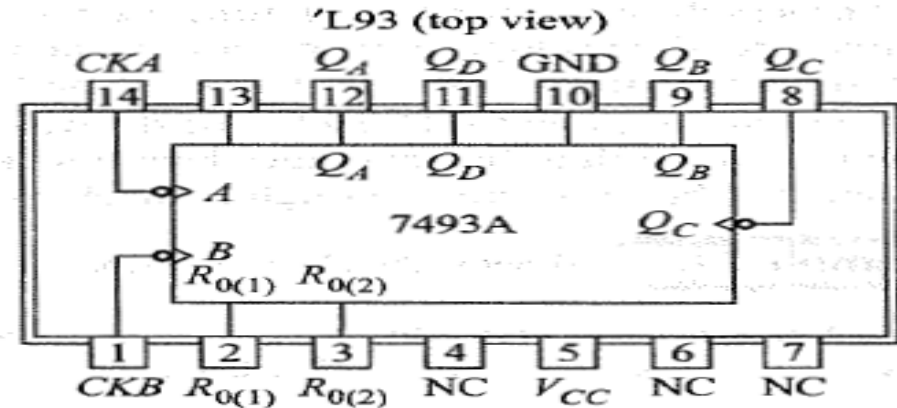
“ The 54/7493A

- “ A 4-bit binary counter that can be used in either a mod-8 or a mod-16 configuration. If the clock is applied at input *CKB*, the outputs will appear at *QB*, *QC*, and *QD*, and this is a mod-8 binary ripple counter.
- “ If the clock is applied at input *CKA* and flip-flop *QA* is connected to input *CKB*, have a mod-16, 4-bit binary ripple counter. The outputs are *QA*, *QB*, *QC*, and *QD*.

ASYNCHRONOUS COUNTERS-9



(a) Logic diagram



Positive logic: see function tables
NC – No internal connection

(b) DIP pinout

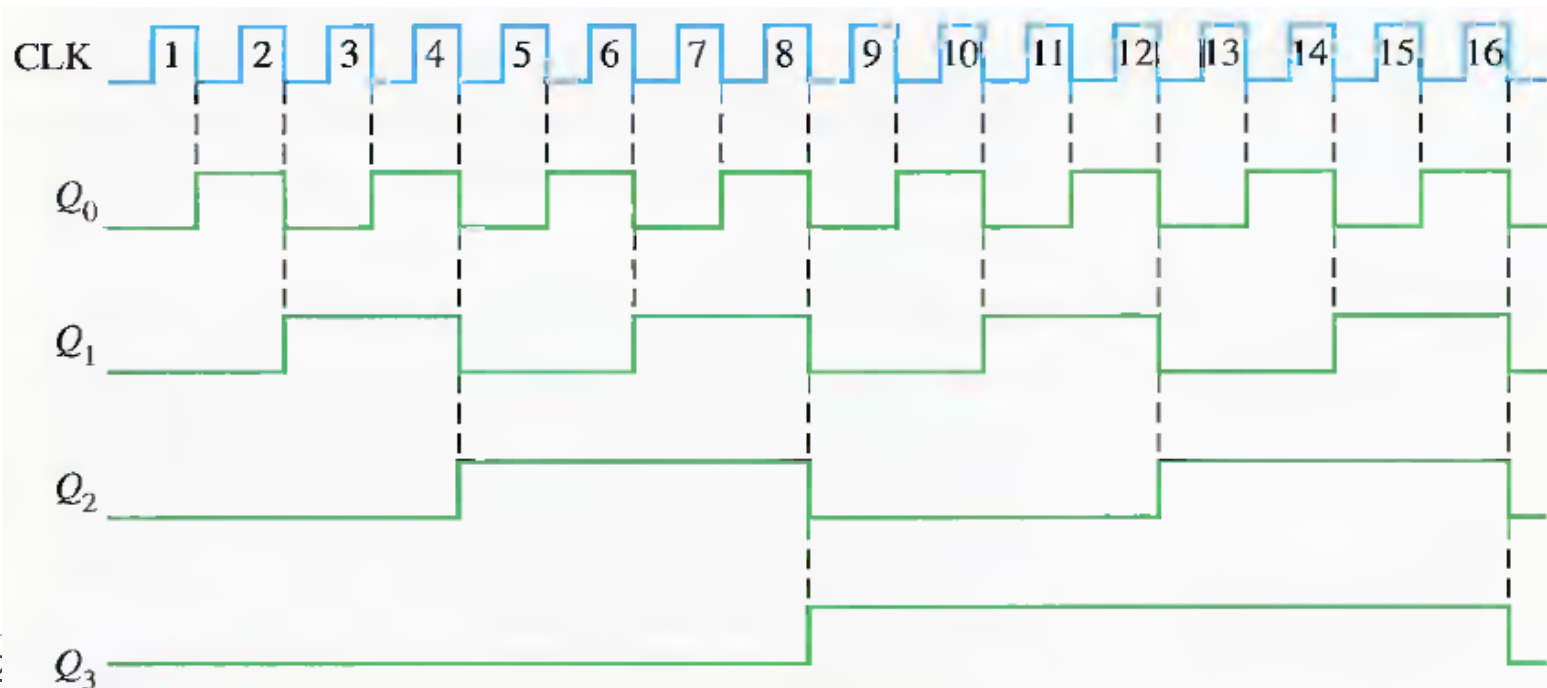
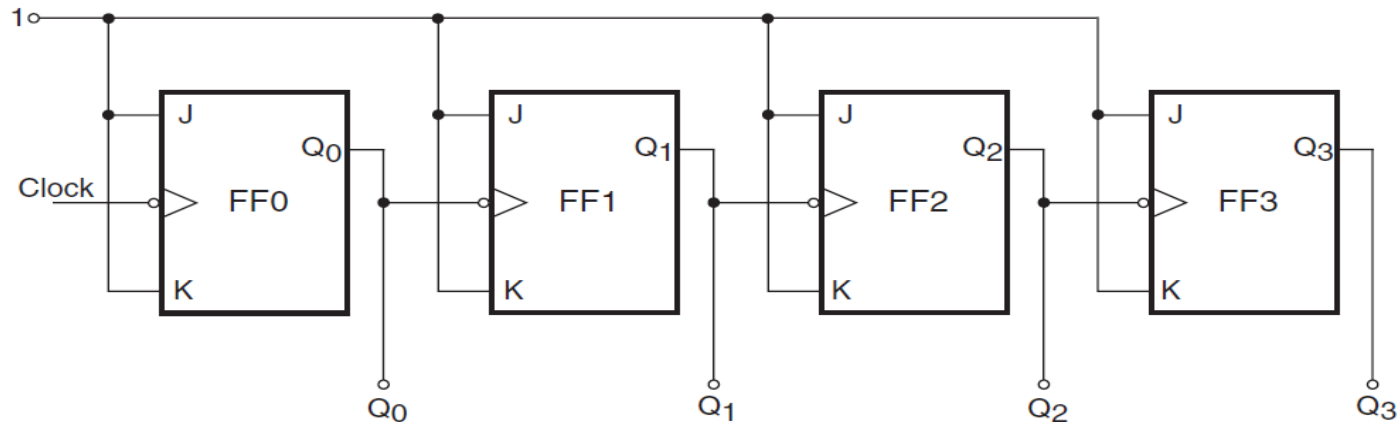
'L93A, 'L93, 'LS93 Count sequence

Count	Output			
	Q _D	Q _C	Q _B	Q _A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

(c) Truth table

ASYNCHRONOUS COUNTERS-8

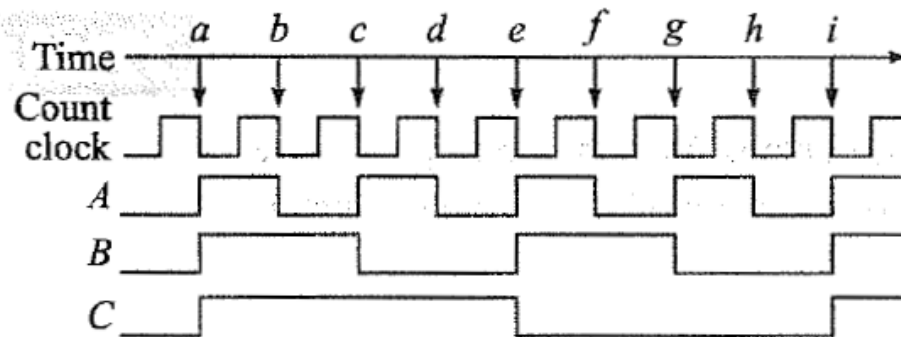
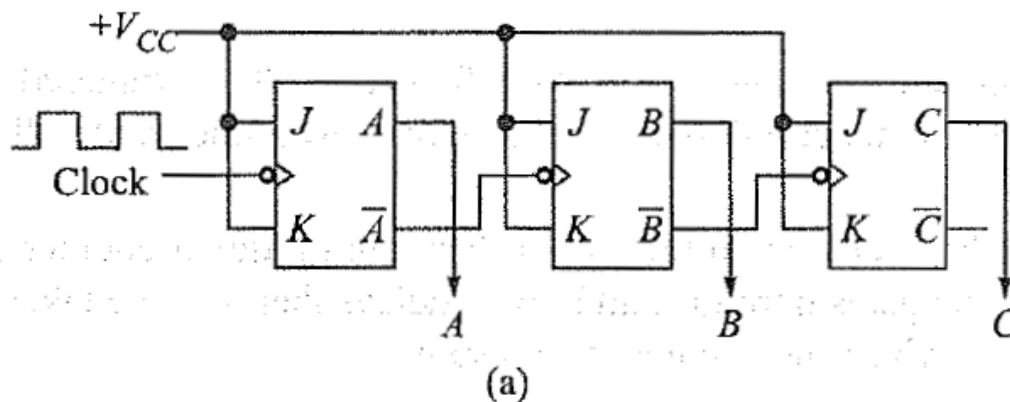
“ Example



ASYNCHRONOUS COUNTERS-9

“ Ripple Counters (Down Counter)

- “ In 3-bit ripple up counter the system clock is still used at the clock input to flip-flop A, but the complement of A, A' , is used to drive flip-flop B, likewise; B' is used to drive flip-flop C.

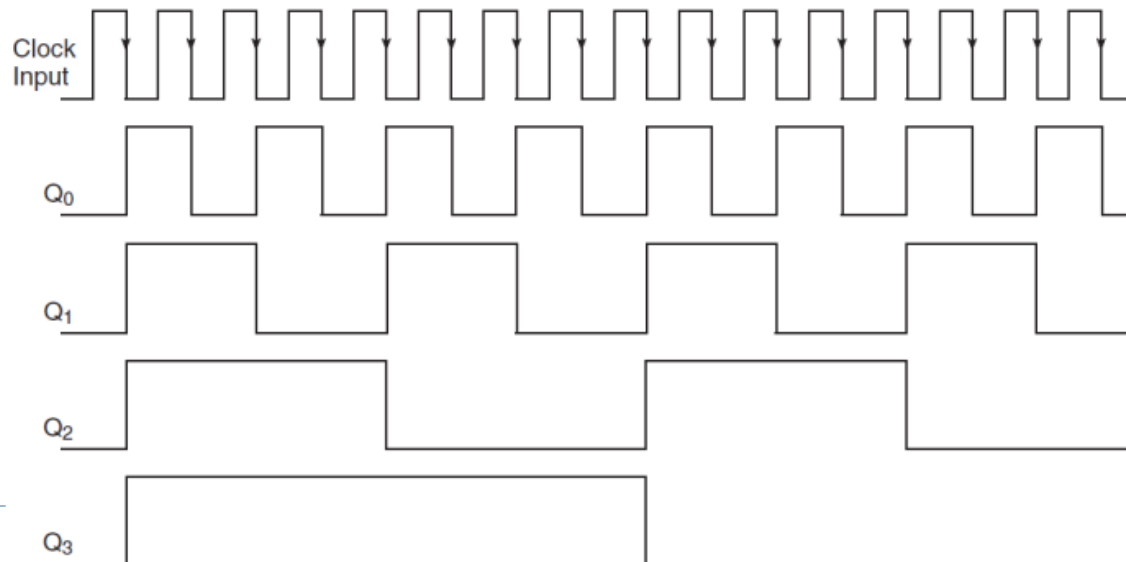
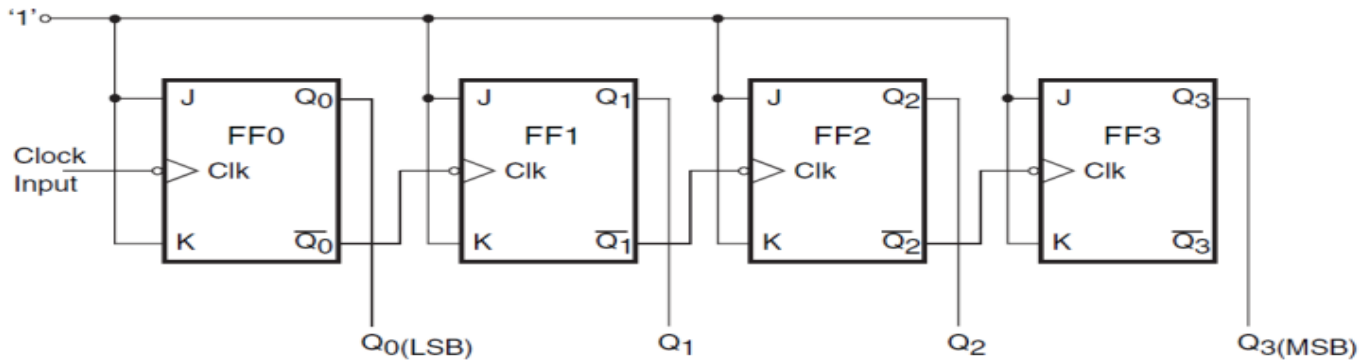


Count	C	B	A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0
7	1	1	1

(c)

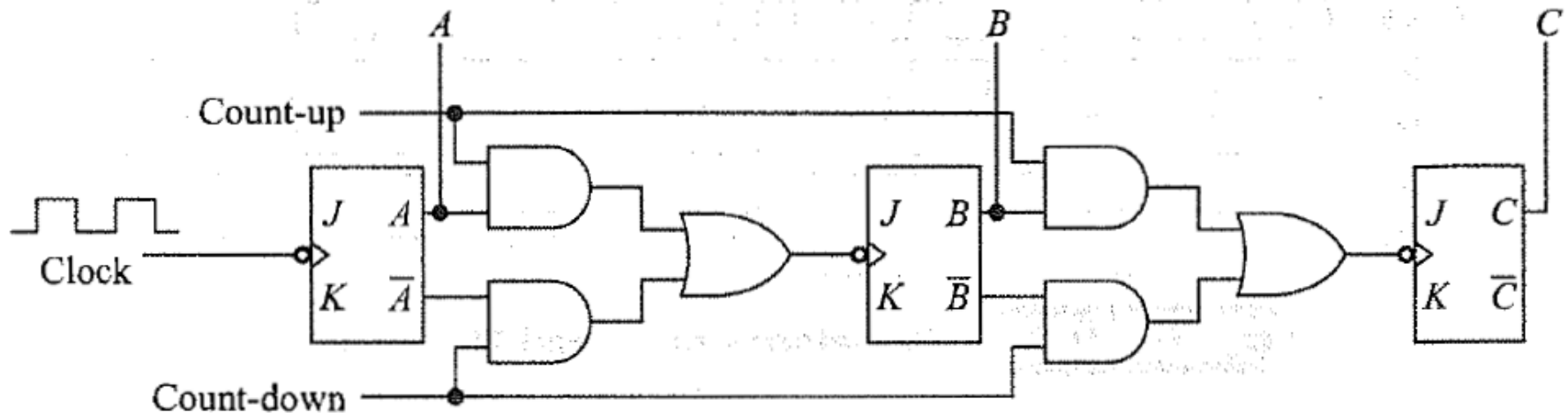
ASYNCHRONOUS COUNTERS-10

“ Ripple Counters (4-bit Down Counter)



ASYNCHRONOUS COUNTERS-11

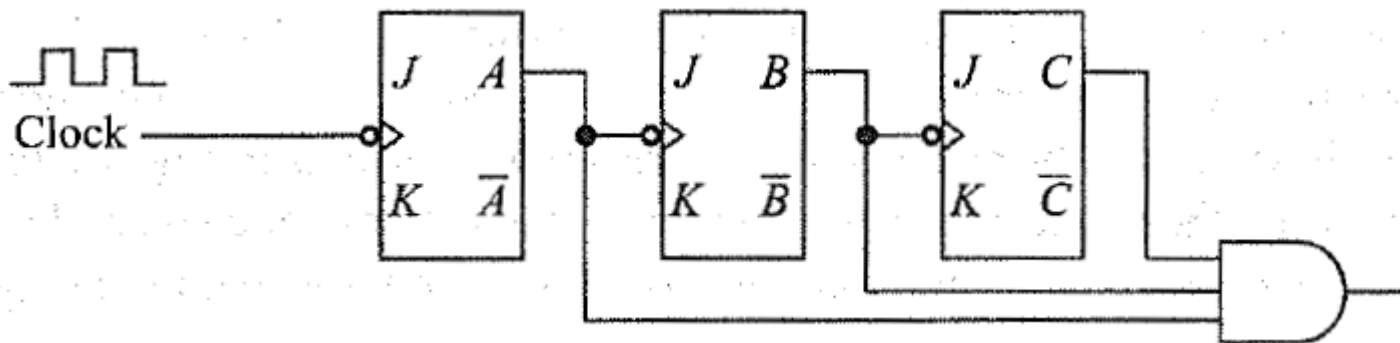
- “ **Asynchronous up-down counter**
- “ Count-up=1 works as Up counter
- “ Count-down=1 works as down counter
- “ Both should not set to 1



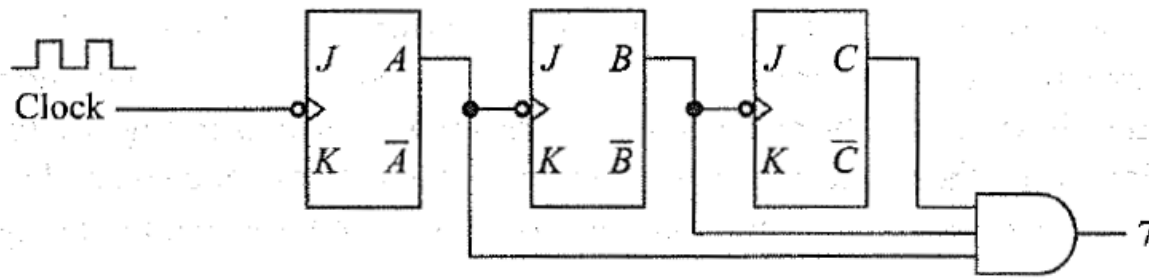
Note: The J and K inputs are all tied to $+V_{CC}$.
The counter outputs are A , B , and C .

DECODING GATES-1

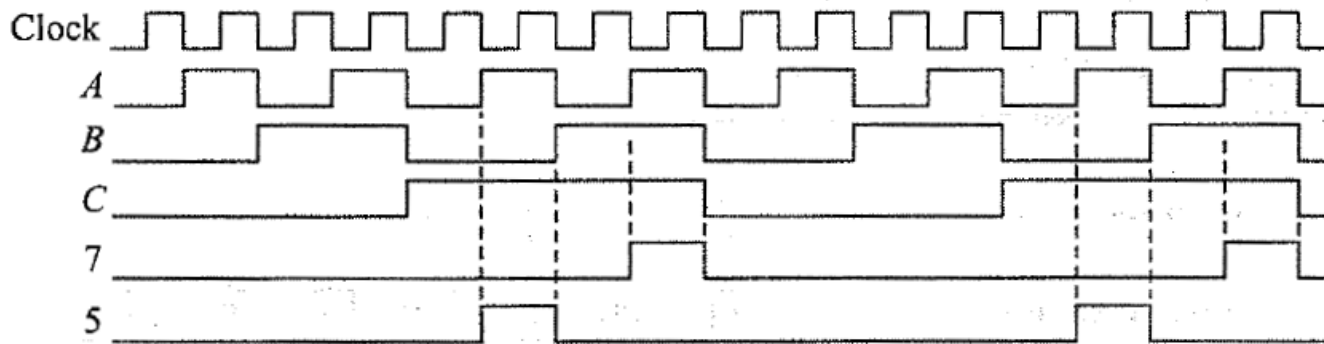
- “ **A decoding gate** can be connected to the outputs of a counter in such a way that the output of the gate will be high (or low) only when the counter contents are equal to a given state.
- “ For instance, the decoding gate connected to the 3-bit ripple counter in below figure will decode state 7 ($CBA = 111$). Thus the gate output will be high only when $A = 1$, $B = 1$, and $C = 1$ and the waveform appearing at the output of the gate is labelled 7. The Boolean expression for this gate can be written $7 = CBA$.



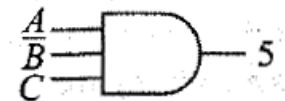
DECODING GATES-2



(a) Decoding gate for state 7



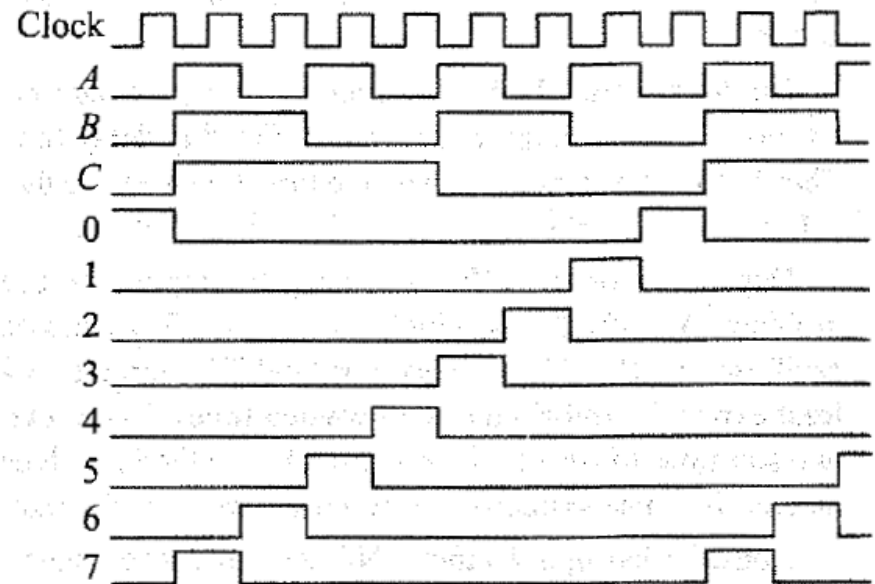
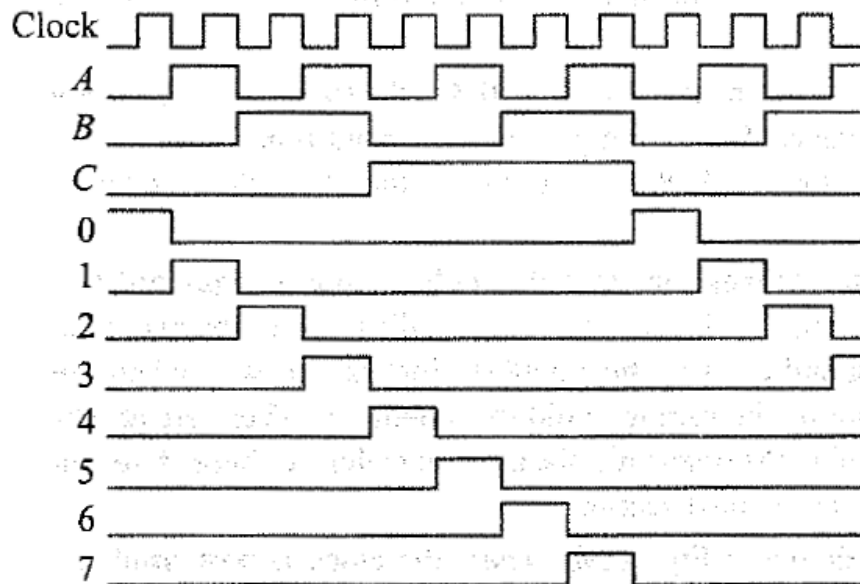
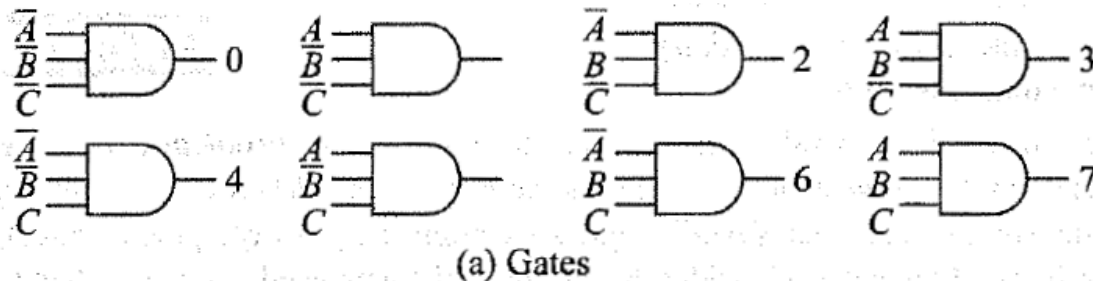
(b) Waveforms



(c) Gate to decode state 5

DECODING GATES-3

“ Decoding gates for a 3-bit binary ripple counter



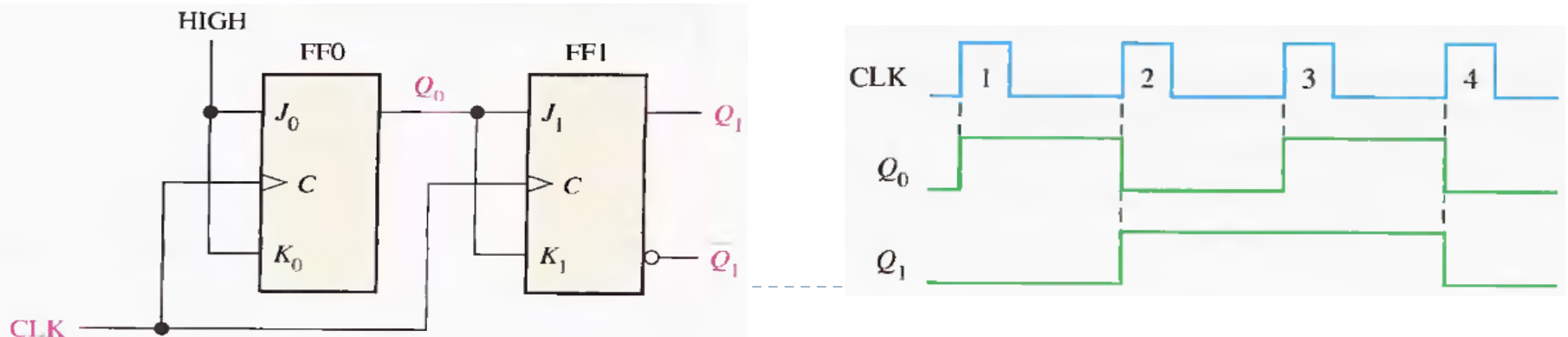
SYNCHRONOUS COUNTERS-1

- “ The ripple counter is the simplest to build, but there is a limit to its highest operating frequency. As each flip-flop has a delay time.
- “ In a ripple counter these delay times are additive, and the total "settling" time for the counter is approximately the delay time times the total number of flip-flops.
- “ There is the possibility of glitches occurring at the output of decoding gates used with a ripple counter.
- “ To overcome by the use of a synchronous parallel counter.
- “ The main difference here is that every flip-flop is triggered in synchronism with the clock.
- “ The term synchronous refers to events that have a fixed time relationship with each other.
- “ A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse.

SYNCHRONOUS COUNTERS-2

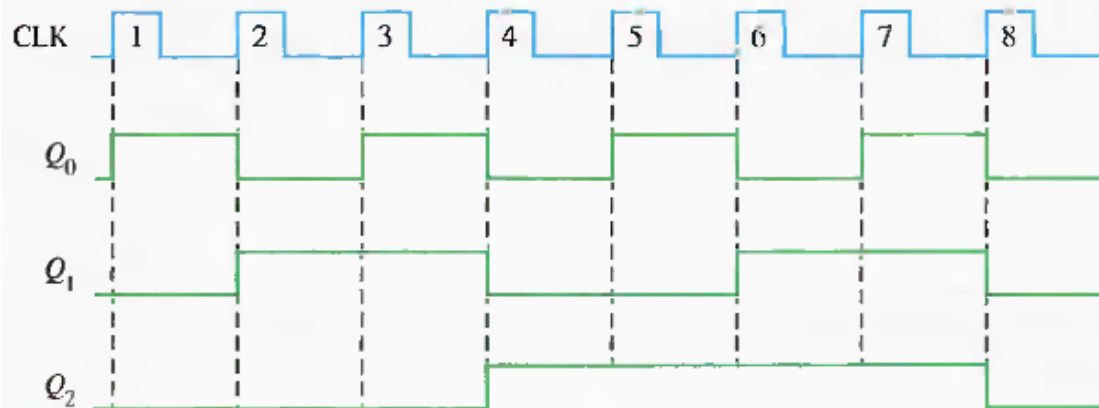
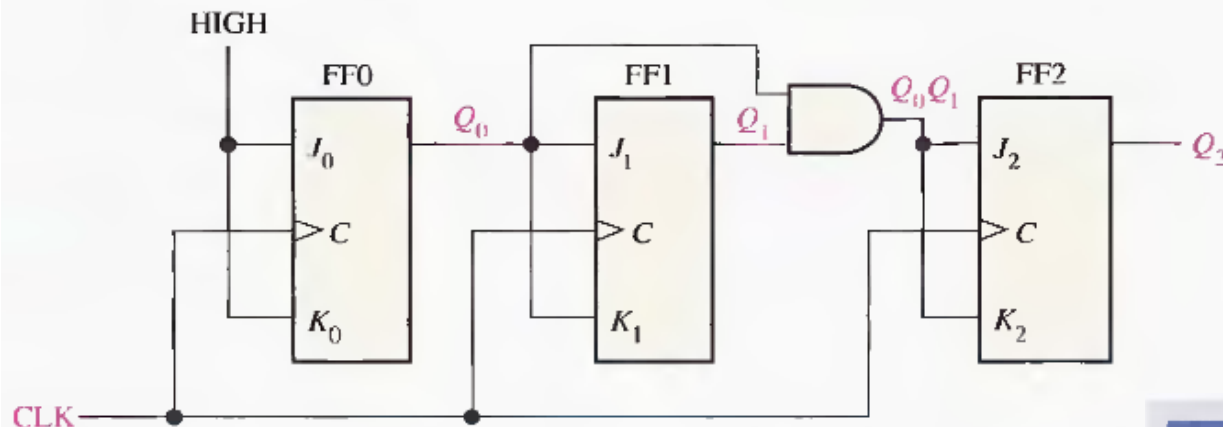
“ A 2-Bit Synchronous Binary Counter

- “ Notice that an arrangement different from that for the asynchronous counter must be used for the J and K inputs of FF1 in order to achieve a binary sequence.
- “ The operation of this synchronous counter is as follows: First, assume that the counter is initially in the binary 0 state that is both flip-flops are RESET. When the positive edge of the first clock pulse is applied, FF0 will toggle and Q_0 will therefore go HIGH. What happens to FF1 at the positive-going edge of CLK1? To find out, let's look at the input conditions of FF1. Inputs J_1 and K_1 are both LOW because Q_0 , to which they are connected, has not yet gone HIGH. Remember, there is a propagation delay from the triggering edge of the clock pulse until the Q output actually makes a transition.



SYNCHRONOUS COUNTERS-3

“ A 3-Bit Synchronous Binary Counter



CLOCK PULSE	Q_2	Q_1	Q_0
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

SYNCHRONOUS COUNTERS-4

- “ First, let's look at Q0. Notice that Q0 changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state.
 - “ Next, let's see how FF2 is made to change at the proper times according to the binary sequence. Notice that both times Q2 changes state it is preceded by the unique condition in which both Q0 and Q1 are HIGH.
 - “ This condition is detected by the AND gate and applied to the J2 and K2 inputs of FF2. Whenever both Q0 and Q1 are HIGH, the output of the AND gate makes the J2 and K2 inputs of FF2 HIGH, and FF2 toggles on the following clock pulse.
 - “ At all other times, the J2 and K2 inputs of FF2 are held LOW by the AND gate output, and FF2 does not change state.
-

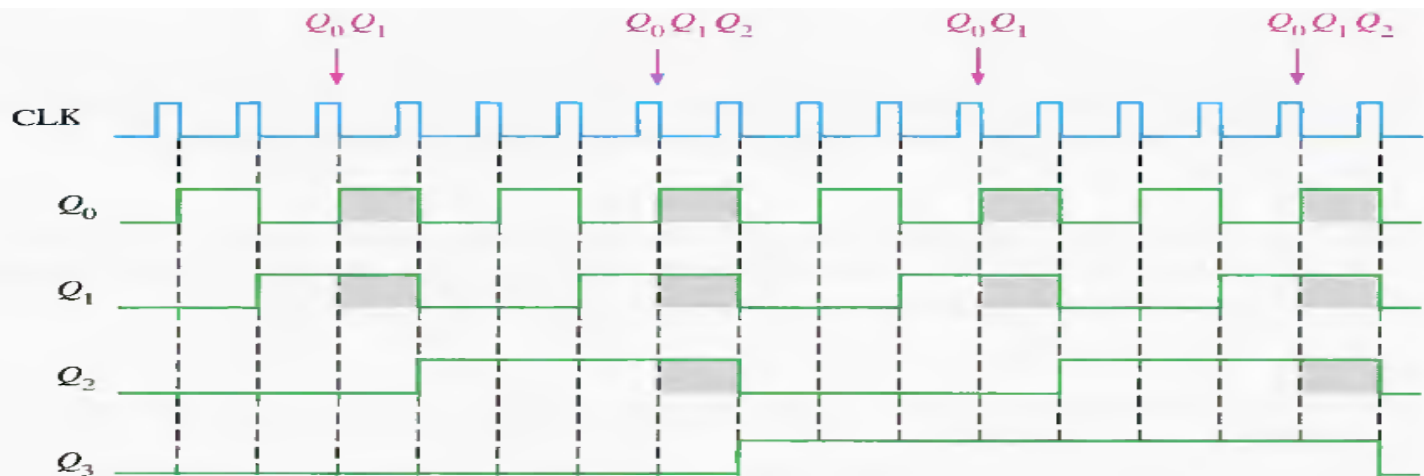
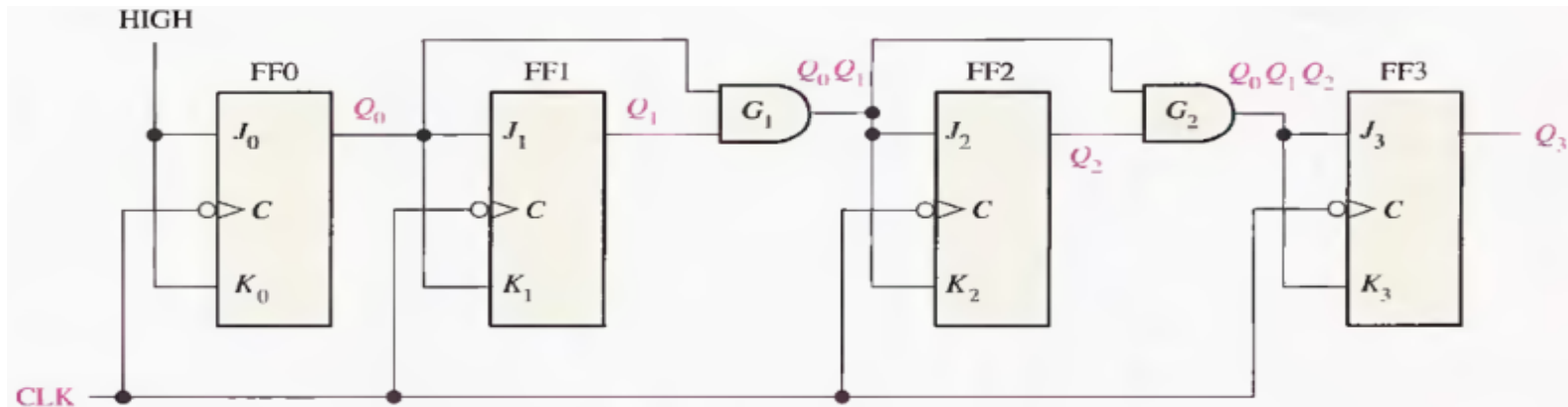
SYNCHRONOUS COUNTERS-4

- “ Each flip-flop should have its J and K inputs connected such that they are HIGH only when the outputs of all lower order flip-flops are in the HIGH state.

SYNCHRONOUS COUNTERS-5

“ A 4-Bit Synchronous Binary Counter

- “ Notice that both of these transitions occur following the times that Q_0 , Q_1 and Q_2 are all HIGH. This condition is decoded by AND gate G_2 so that when a clock pulse occurs $FF3$ will change state.



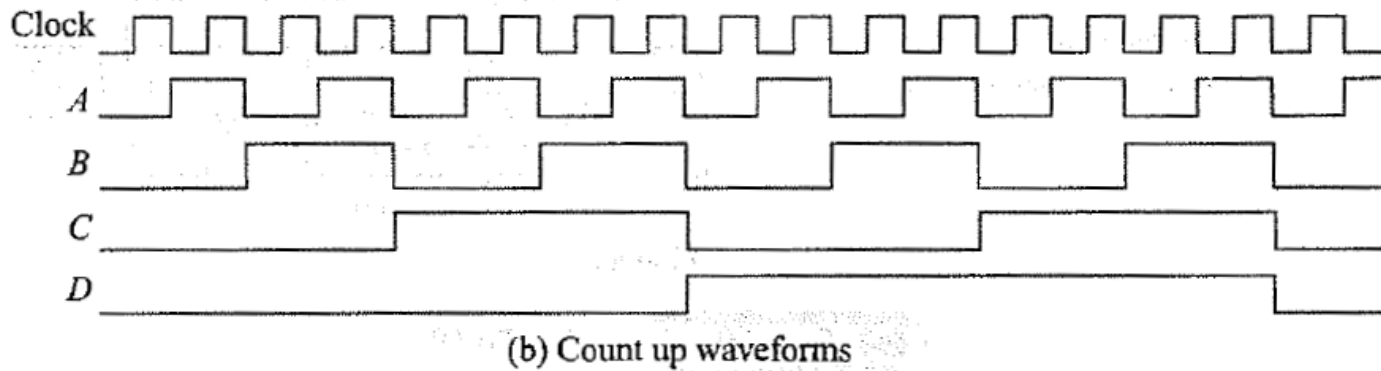
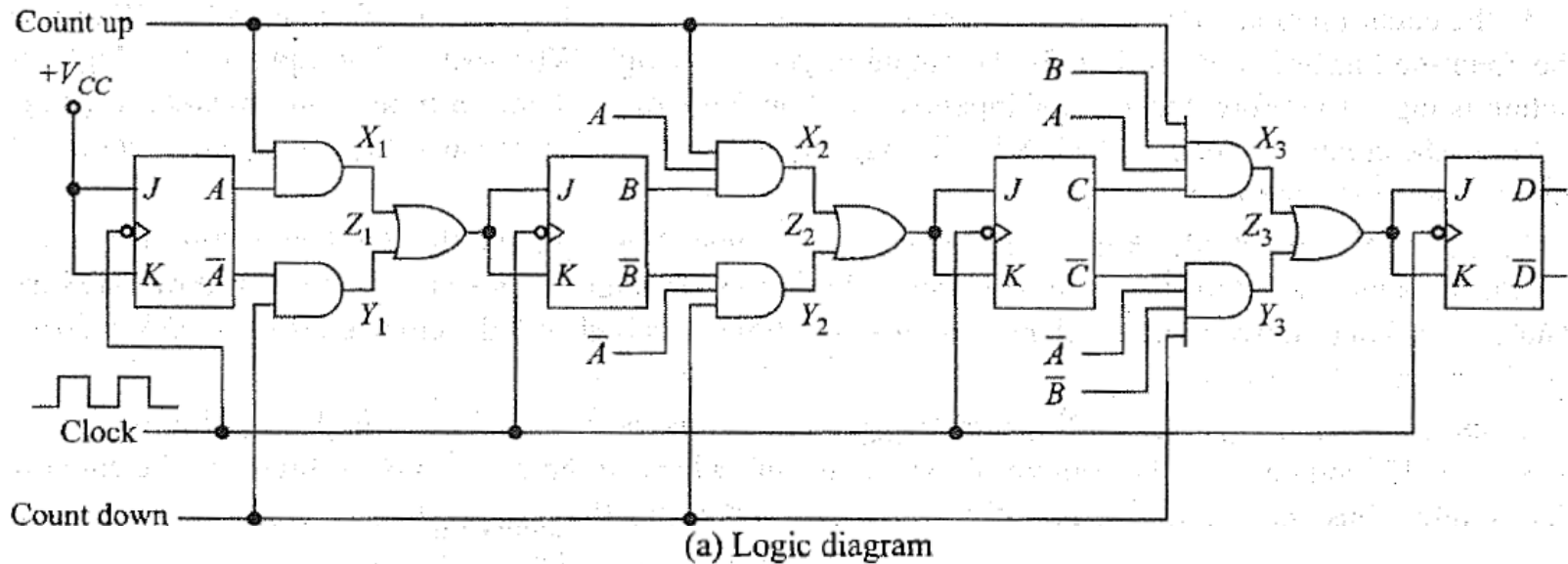
SYNCHRONOUS COUNTERS-6

“ UP/DOWN SYNCHRONOUS COUNTERS

- “ Below shows Synchronous 4-bit up-down counter.
- “ To operate in the count-up mode, the logic High is applied at the count-up input, while the count-down input is held low.
- “ To operate in the count-down mode, the logic Low is applied at the count-down input while holding the count-up input low.

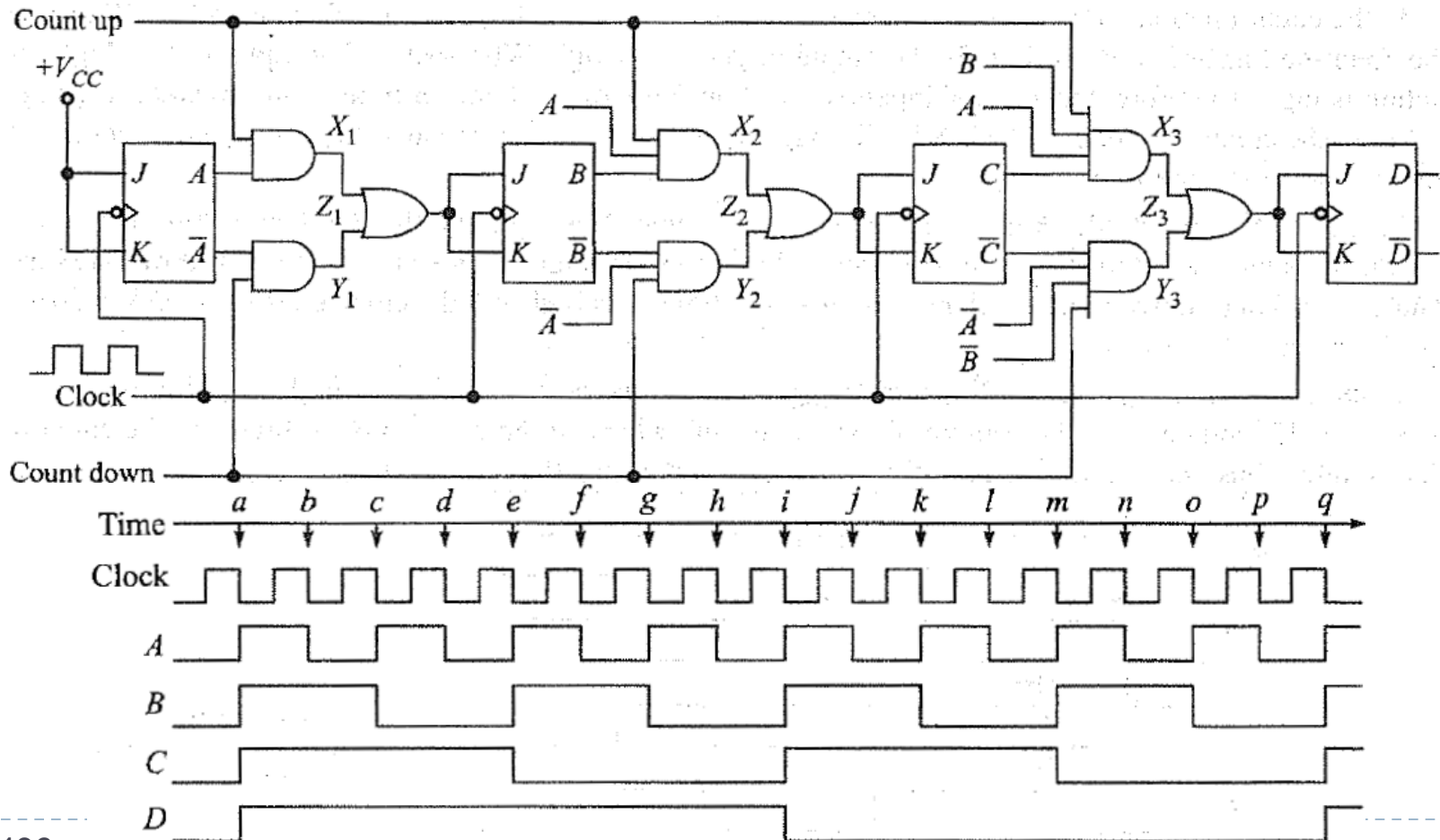
SYNCHRONOUS COUNTERS-7

“ UP/DOWN SYNCHRONOUS COUNTERS



SYNCHRONOUS COUNTERS-8

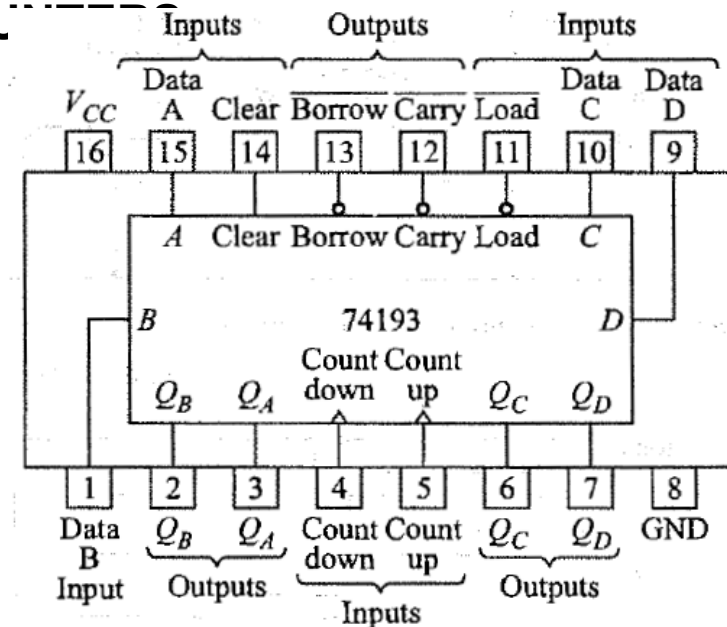
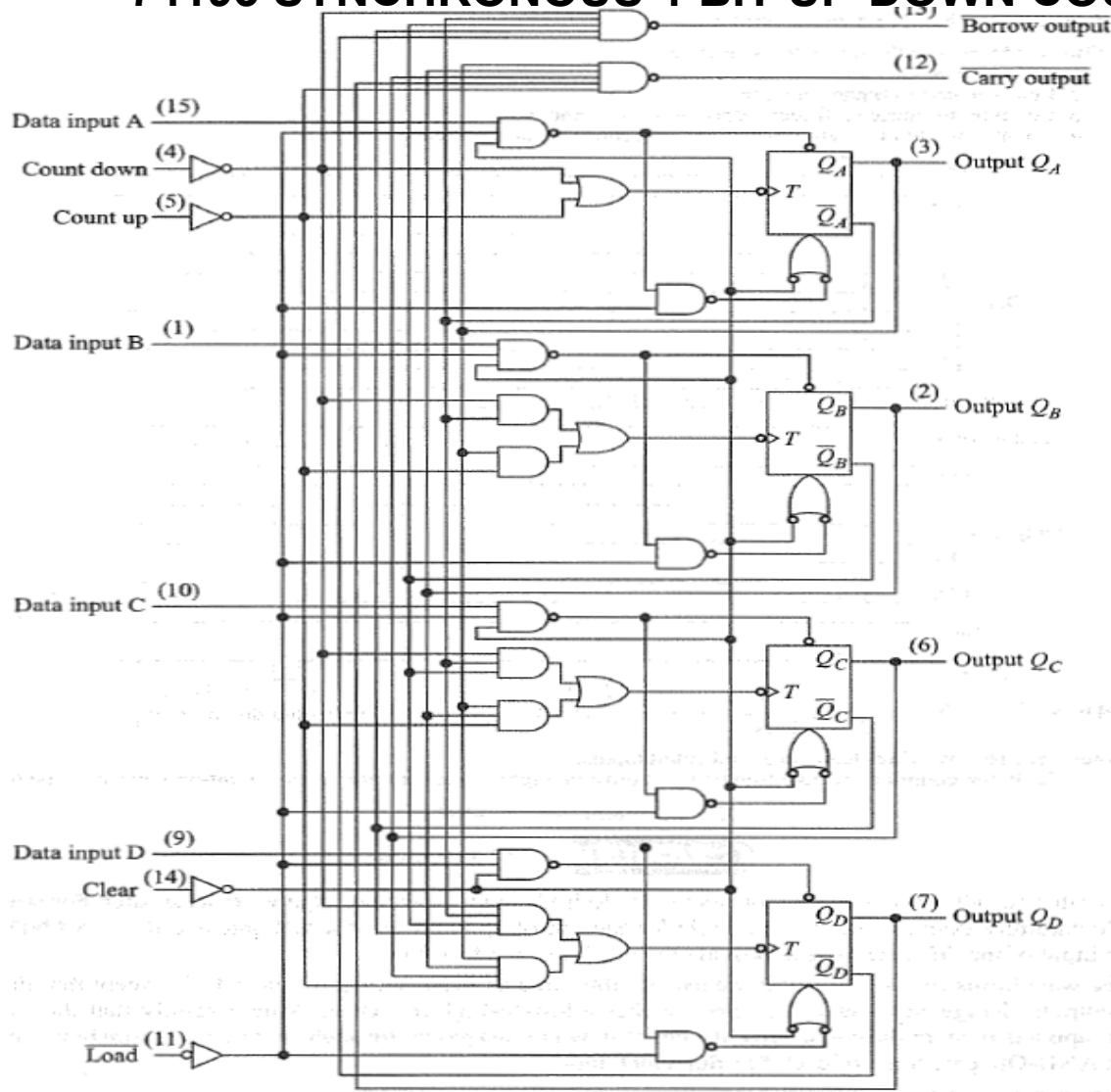
“ UP/DOWN SYNCHRONOUS COUNTERS



(c) Count down waveforms

SYNCHRONOUS COUNTERS-8

“ 74193 SYNCHRONOUS 4-BIT UP-DOWN COUNTER”



“ Counter that can also be cleared and preset to any desired count-attributes that we discuss later. Now, should carefully examine the steering logic for each flip-flop and study the OR gate and the two AND gates at the input of the OR gate used to provide the clock to each flip-flop.

“ Parallel up-down counter

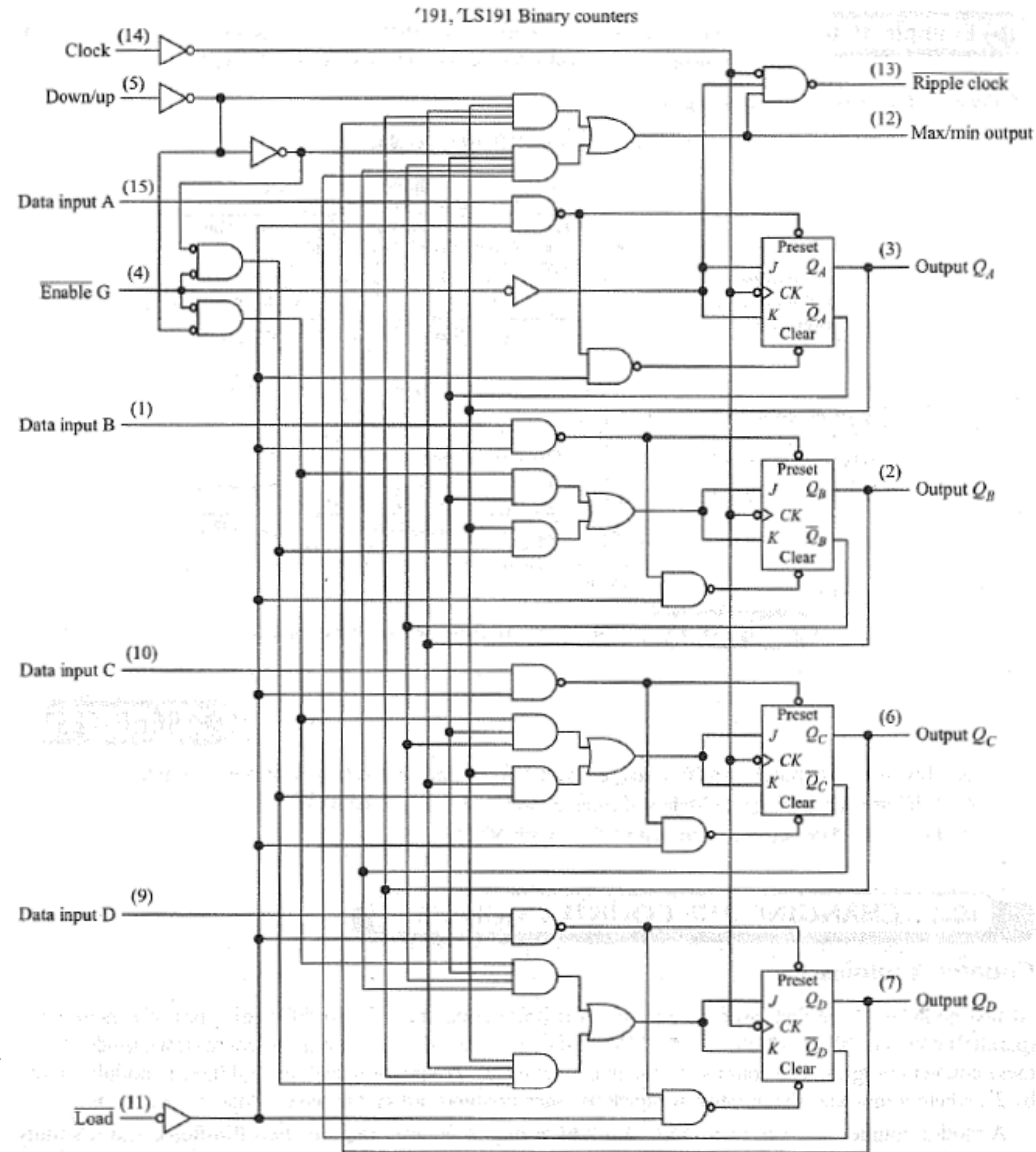


SYNCHRONOUS COUNTERS-9

“ 54/74191 synchronous up-down counter

“ A careful examination of the AND-OR-gate logic used to precondition the J and K inputs to each flip-flop will reveal that this counter uses look-ahead logic.

“ Additional logic allows one to clear or preset this counter to any desired count.



CHANGING THE COUNTER MODULUS-1

“ Counter Modulus

- “ A modulus given by 2^n , where n indicates the number of flip-flops. Such counters are said to have a "natural count" of 2^n .
- “ A mod-2 counter consists of a single flip-flop; a mod-4 counter requires two flip-flops, and it counts through four discrete states. Three flip-flops form a mod-8 counter, while four flip-flops form a mod-16 counter.
- “ Thus we can construct counters that have a natural count of 2, 4, 8, 16, 32, and so on by using the proper number of flip-flops.
- “ It is often desirable to construct counters having a modulus other than 2, 4, 8, and so on. For example, a counter having a modulus of 3, or 5, would be useful. A small modulus counter can always be constructed from a larger modulus counter by skipping states. Such counters are said to have a *modified count*.
- “ It is first necessary to determine the number of flip-flops required. The correct number of flip-flops is determined choosing the lowest natural count that is greater than the desired modified count.
- “ For example, a mod-7 counter requires three flip-flops, since 8 is the lowest natural count greater than the desired modified count of 7.
 - “ MOD number = 2^N
 - “ where N is the number of flip-flops connected

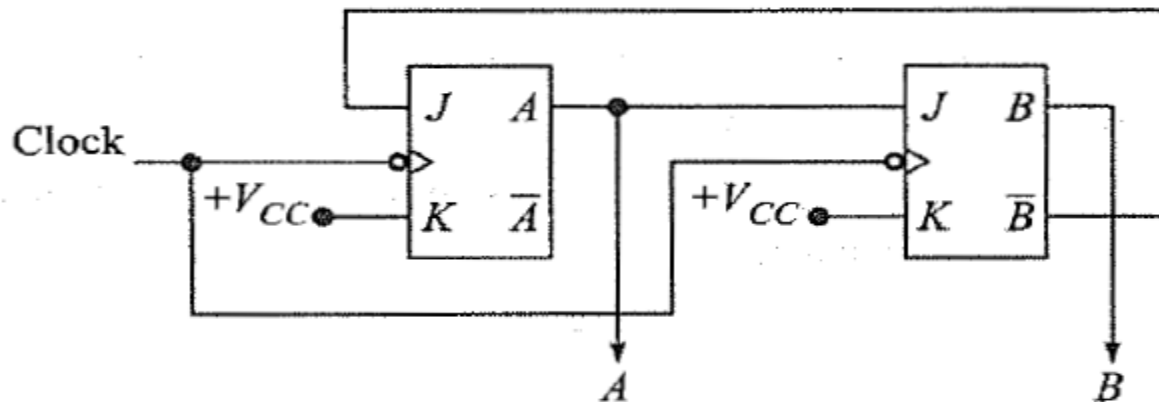
CHANGING THE COUNTER MODULUS-2

“ Question

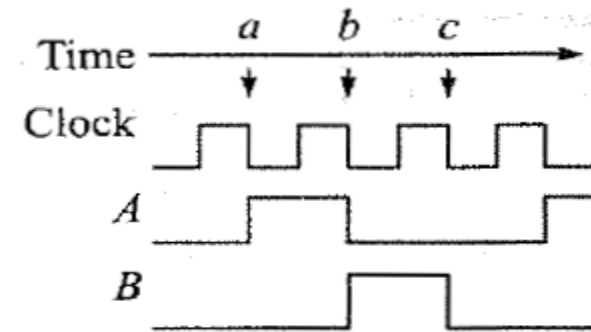
- “ Indicate how many flip-flops are required to construct each of the following counters: (a) mod-3, (b) mod-6, and (c) mod-9.
- “ The lowest natural count greater than 3 is 4. Two flip-flops provide a natural count of 4. Therefore, it requires at least two flip-flops to construct a mod-3 counter.
- “ Construction of a mod-6 counter requires at least three flip-flops, since 8 is the lowest natural count greater than 6.
- “ A mod-9 counter requires at least four flip-flops, since 16 is the lowest natural count greater than 9.

CHANGING THE COUNTER MODULUS-3

“ A Mod-3 Counter



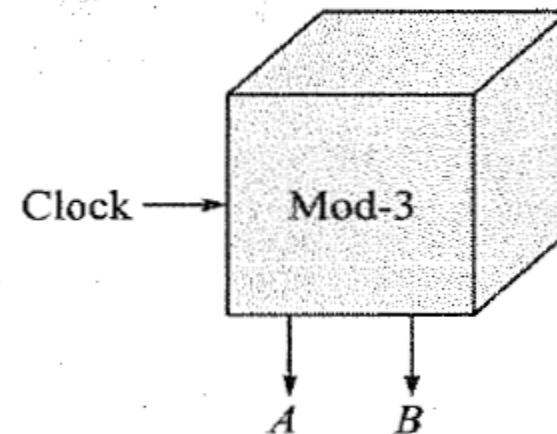
(a) Logic diagram



(b) Waveforms

B	A	Count
0	0	0
0	1	1
1	0	2
<hr/>		
0	0	0

(c) Truth table



(d) Logic block

CHANGING THE COUNTER MODULUS-4

“ **A Mod-3 Counter**

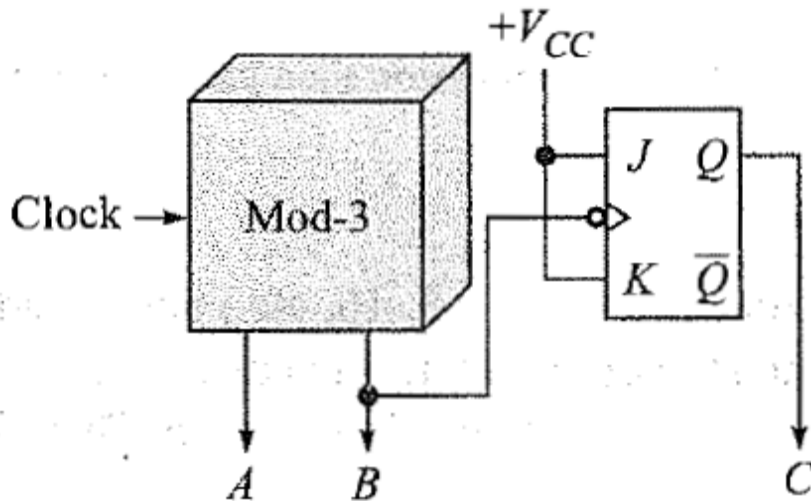
- “ Above circuit considered as a divide-by-3 block, since the output waveform at B (or at A) has a period equal to three times that of the clock-in other words, this counter
- “ divides the clock frequency by 3.

CHANGING THE COUNTER MODULUS-5

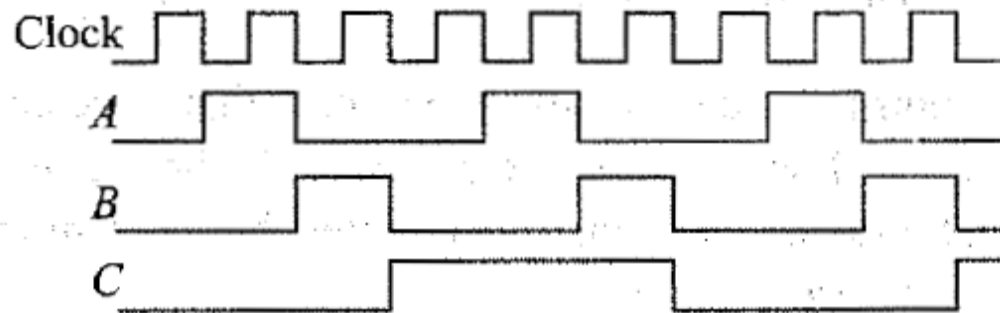
- “ If we consider a basic flip-flop to be a mod-2 counter, we see that a mod-4 counter (two flip-flops in series) is simply two mod-2 counters in series.
- “ Similarly, a mod-8 counter is simply a $2 \times 2 \times 2$ connection, and so on.
- “ Thus a great number of higher-modulus counters can be formed by using the product of any number of lower modulus counters.

CHANGING THE COUNTER MODULUS-6

- “ **A Mod-6 Counter**
- “ Mod-6 counter is a ($3 \times 2 = 6$).
- “ The output of the single flip-flop is labeled C. Notice that it is a symmetrical waveform, and it also has a frequency of one-sixth that of the input clock.
- “ Also, this can no longer be considered a synchronous counter since flip flop C is triggered by flip-flop B; that is, the flip-flops do not all change status in synchronism with the clock.



(a) 3×2 Mod-6 counter



(b) Waveforms

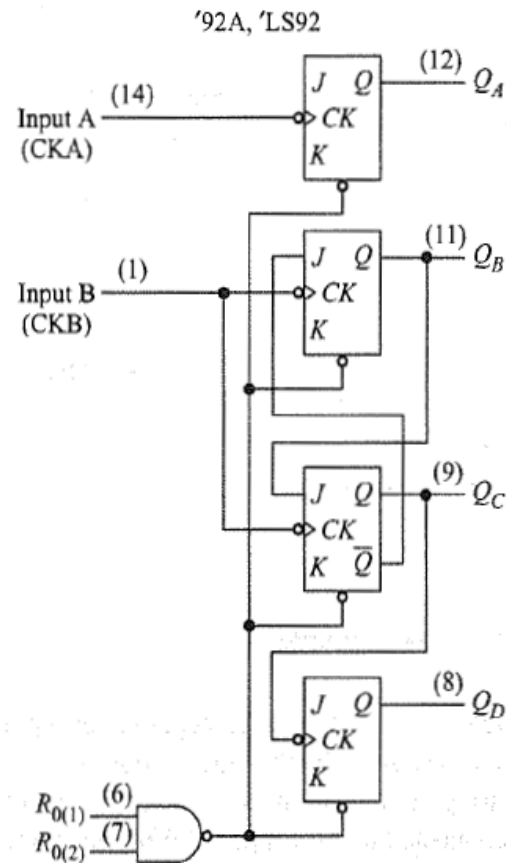
CHANGING THE COUNTER MODULUS-6

“ The 54/7492A

“ The 5417492A ('92A) is a divide-by-12. A careful examination of the logic diagram will reveal that flip-flops Q_B , Q_C , and Q_D are exactly the same as the 3 x 2 counter.

“ Thus if the clock is applied to input B of the '92A and the outputs are taken at Q_B , Q_C , and Q_D , this is a mod-6 counter.

“ On the other hand, if the clock is applied at input A and Q_A is connected to input B , we have a 2 x 3 x 2 mod-12 counter.

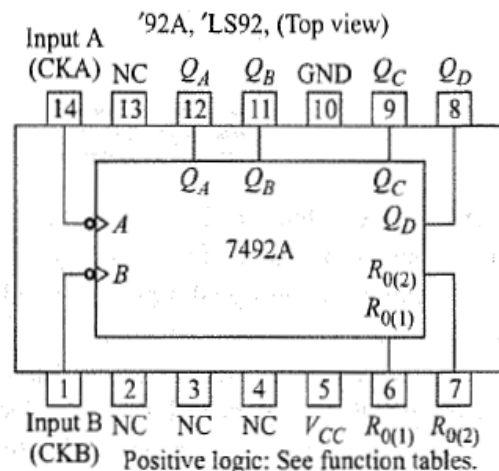


(a) Logic

'92A, 'LS92 Count sequence
(See Note C)

Count	Output			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	H	L	L	L
7	H	L	L	H
8	H	L	H	L
9	H	L	H	H
10	H	H	L	L
11	H	H	L	H

(b) Truth table



Note: Output Q_A connected to input B

(c) Pinout

MODULE-5

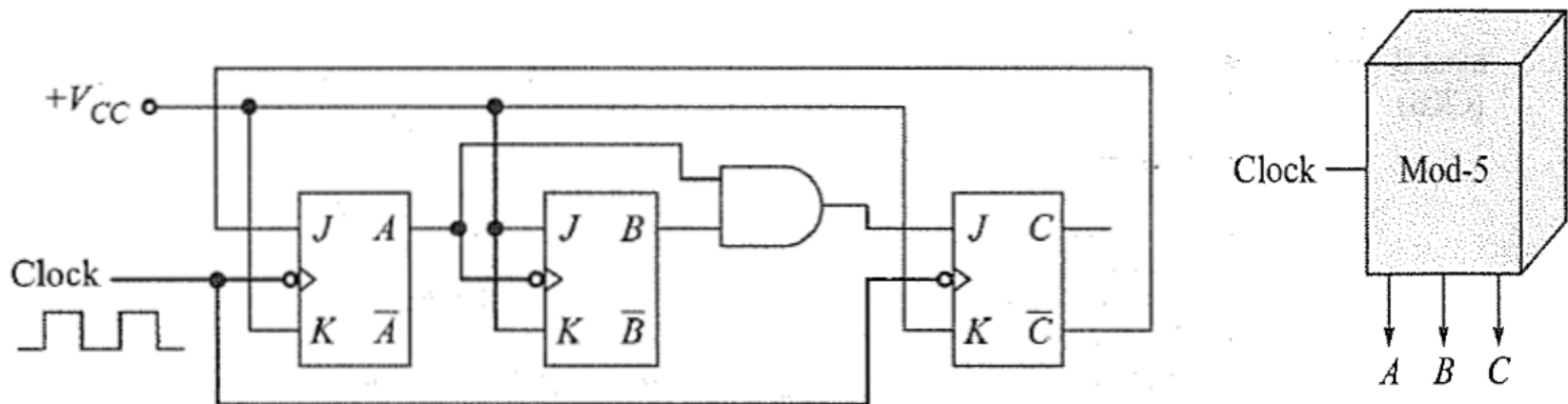
Counters
&
DAC and ADC

Text Books Referred

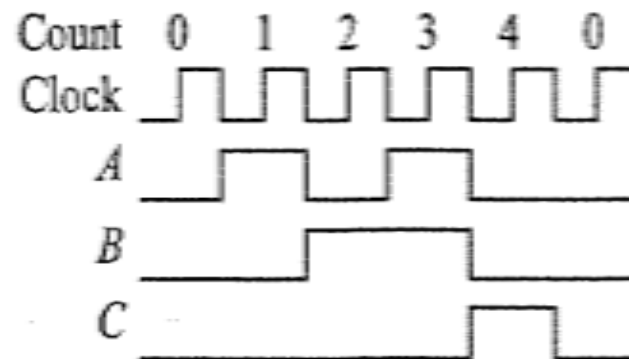
- “ Donald P Leach, Albert Paul Malvino & Goutam Saha: Digital Principles and Applications, 7th Edition, Tata McGraw Hill, 2015
- “ Thomas L. Floyd: Digital Fundamentals, 9th Edition., Pearson International Edition.

DECADE COUNTERS-1

“ A Mod-5 Counter



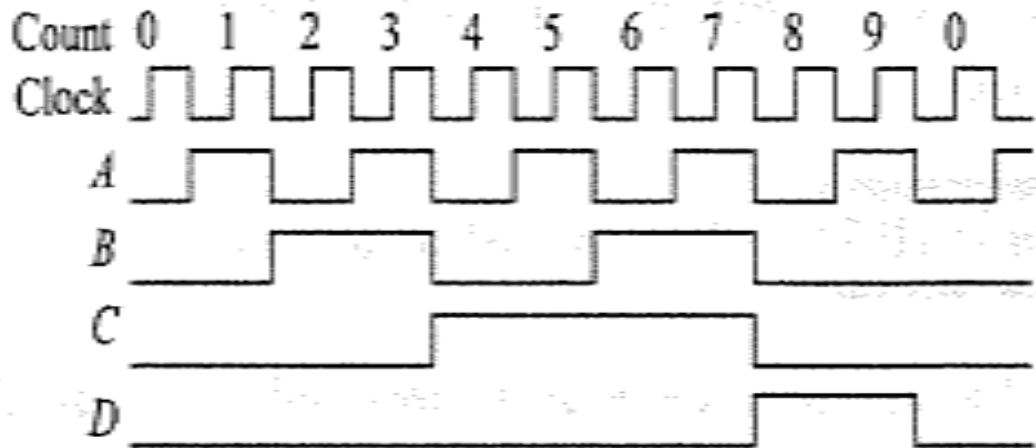
C	B	A	Count
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
0	0	0	0



DECADE COUNTERS-2

- “ **A Mod-5 Counter**
- “ A modulo – 5 counter, the counter should reset when it reaches state 101.

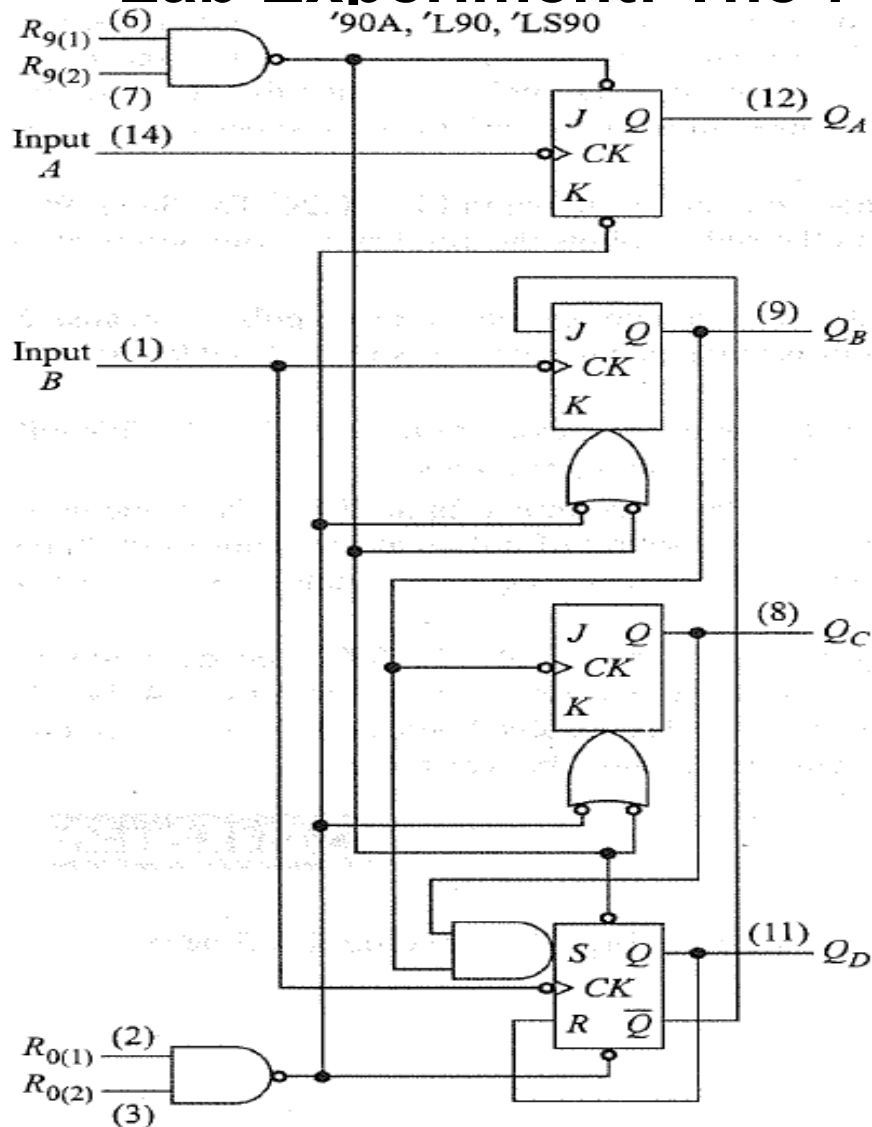
“ A Mod-10 Counter



D	C	B	A	Count
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
0	0	0	0	0

DECADE COUNTERS-4

Lab Experiment: The 7490



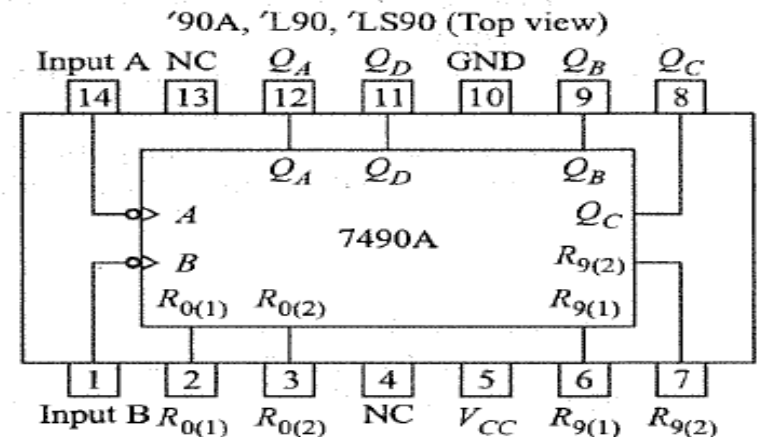
'90A, 'L90, 'LS90
BCD count sequence
(See note A)

Count	Output			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

'90A, 'L90, 'LS90
Bi-quinary (5-2)
(See note B)

Count	Output			
	Q_A	Q_D	Q_C	Q_B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

(b) Truth table



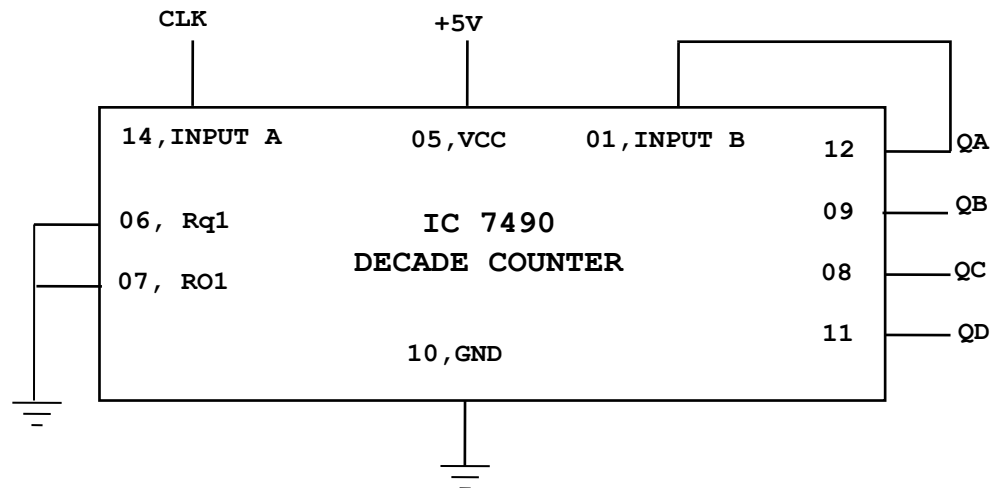
Positive logic: See function tables

Note: A output Q_A connected to input B
B output Q_D connected to input A

DECADE COUNTERS-5

- “ Lab Experiment: The 7490
- “ MOD-10 Counter

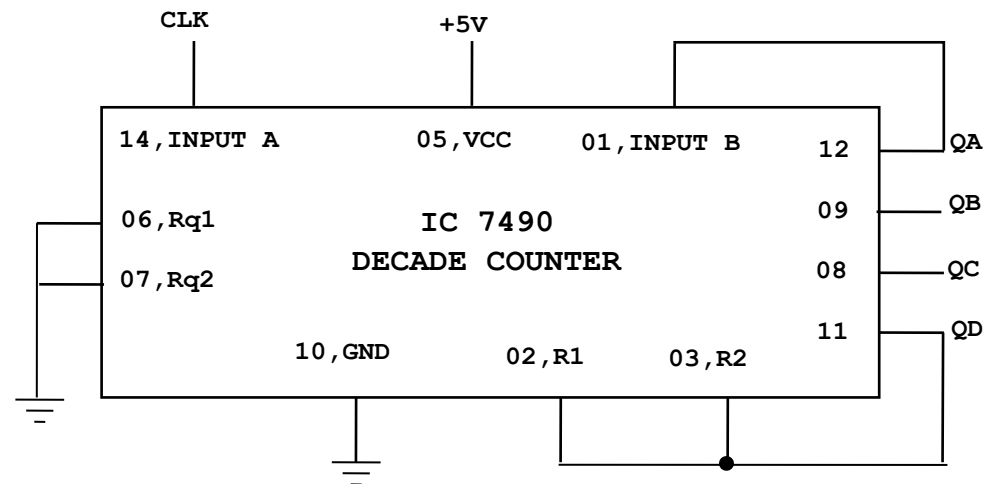
Clock	Output			
	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1



DECADE COUNTERS-6

- “ Lab Experiment: The 7490
- “ MOD-8 Counter

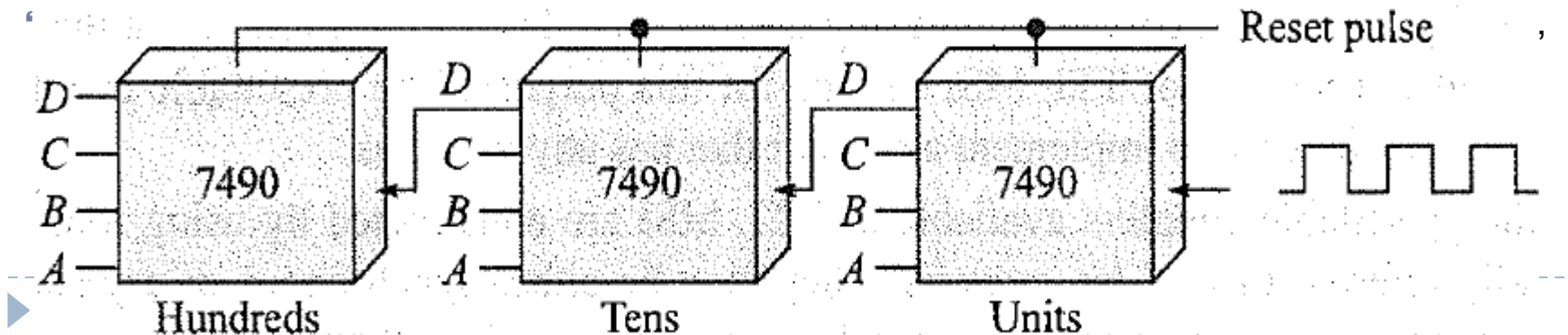
Clock	Output			
	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1



DECADE COUNTERS-7

“ Cascaded 7490's can count to 999

- “ An interesting application using three decade counters is shown in figure below.
- “ The three '90A counters are connected in series such that the first one (on the right) counts the number of input pulses at its clock input. We call it a *units counter*.
- “ The middle '90A will advance one count each time the units counter counts 10 input pulses, because *D* from the units counter will have a single negative transition as that counter progresses from count 9 to 0. This middle block is then called *tens counter*.
- “ The left '90A will advance one count each time the tens counter progresses from count 9 to 0. This will occur once for every 100 input pulses. Thus this block is called the *hundreds counter*.
- “ Now the operation should be clear. This logic circuit is capable of counting input pulses from one up to 999.
- “ The procedure is to reset all the '90As and then count the number of pulses at the input to the units counter.



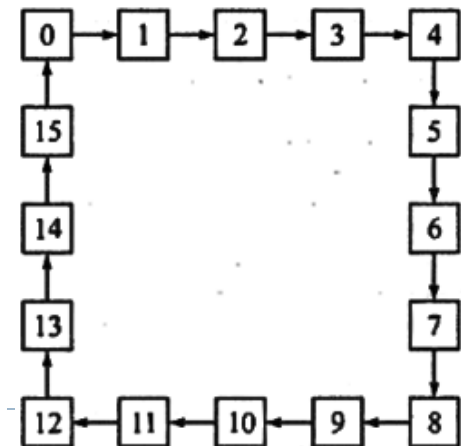
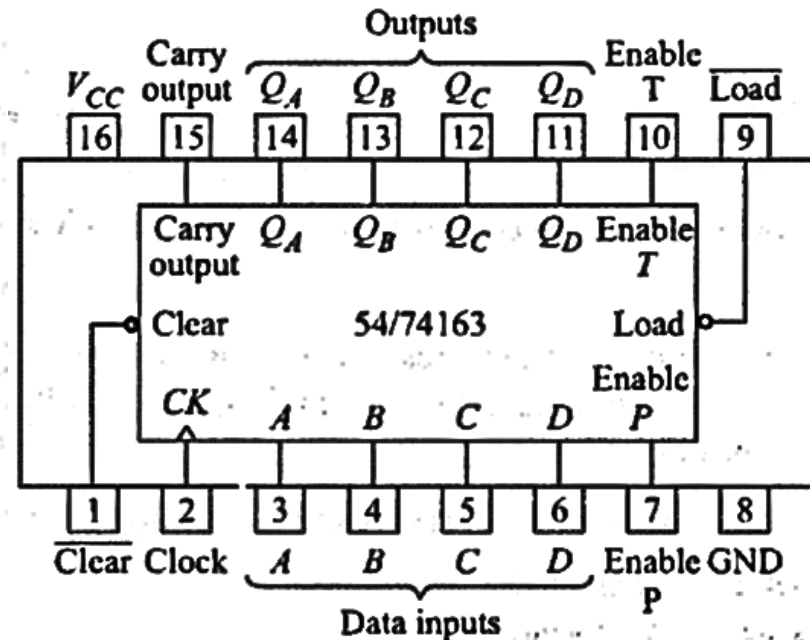
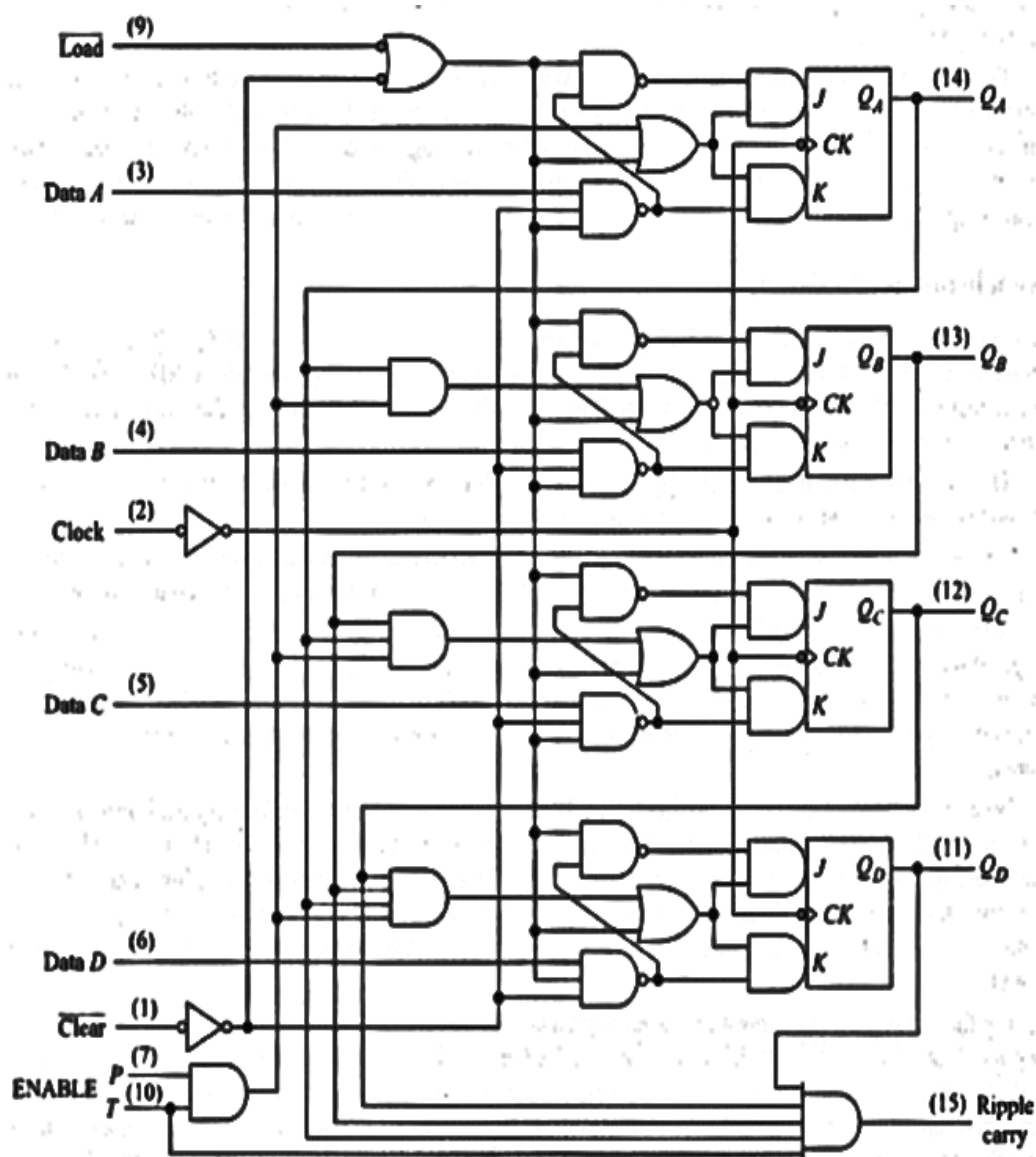
PRESETTABLE COUNTERS-1

- “ As many counters ICs are of type synchronous or asynchronous, but have little control over the internal logic used to implement each counter.
- “ So efforts are concentrated on inputs, outputs, and control signals for implementing desired counters.

PRESETTABLE COUNTERS-2

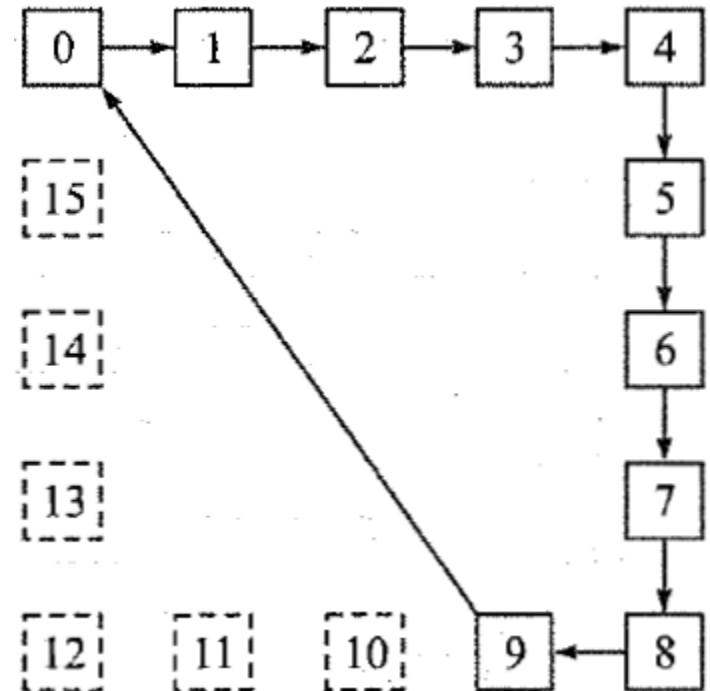
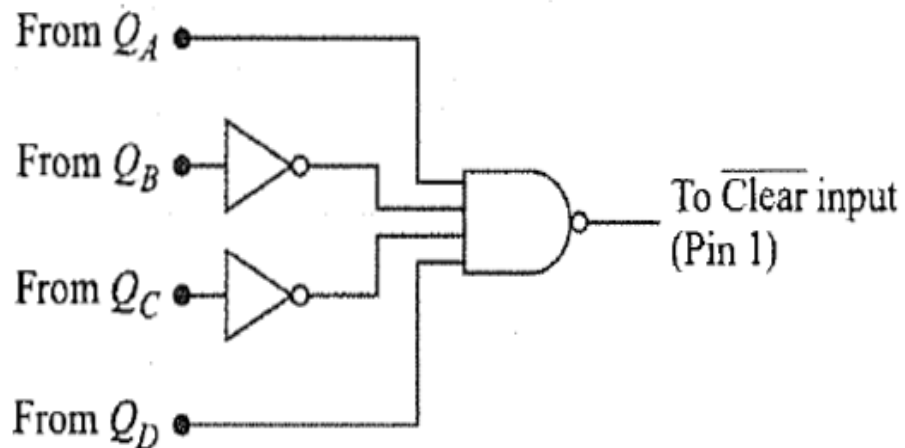
- “ **Synchronous Up Counters**
- “ **A 54/74163 synchronous 4-bit counter.**
- “ The four flip-flop outputs are QA , QB , Qc , and QD , while the CARRY output on pin 15 can be used to enable successive counter stages (e.g. in a units, tens, hundreds application).
- “ The two ENABLE inputs (P on pin 7 and T on pin 10) are used to control the counter.
- “ A low level on the CLEAR input will reset all flip-flop outputs low at the very next clock transition.
- “ When a low level is applied to the LOAD input, the counter is disabled.
- “ The count length can be very easily modified by making use of the synchronous CLEAR input. It is a simple matter to use a NAND gate to decode the maximum count desired, and use the output of this NAND gate to clear the counter synchronously to count 0000.

PRESETTABLE COUNTERS-3



PRESETTABLE COUNTERS-4

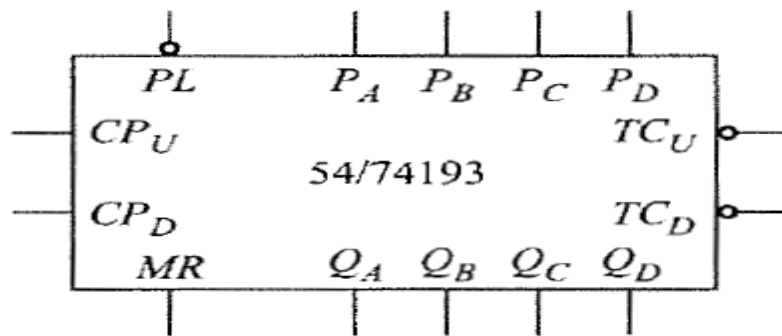
“ Mod-10 Counter



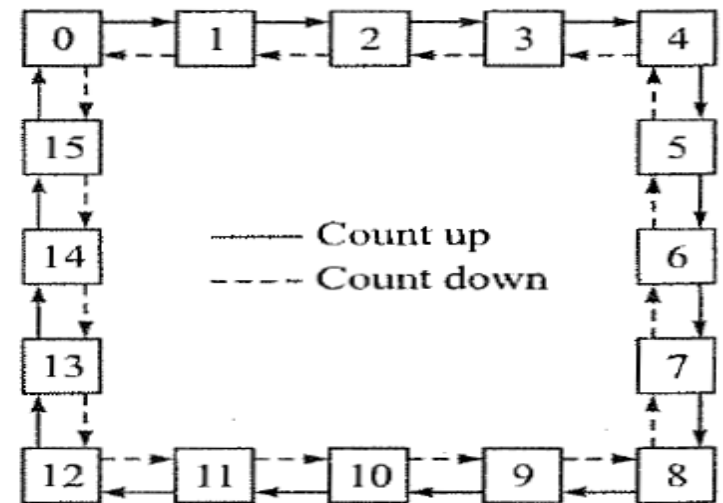
PRESETTABLE COUNTERS-5

“ Synchronous Up-Down Counters

- “ 54/74193 is a 4-bit synchronous up-down binary counter
- “ Pin PL (active low) is a control input for loading data into pins PA, PB, PC, and PD. *PL* is low, the data present at these four inputs is shifted into the counter
- “ Pin MR is the master reset, a high level on MR will reset all flip-flops.
- “ Outputs *TCU* and *TCD* are to be used to drive the following units, such as in a cascade arrangement.
- “ Placing the clock on *CP_U* will cause the counter to count up, and placing the clock on *CP_D* will cause the counter to count down.



(a)

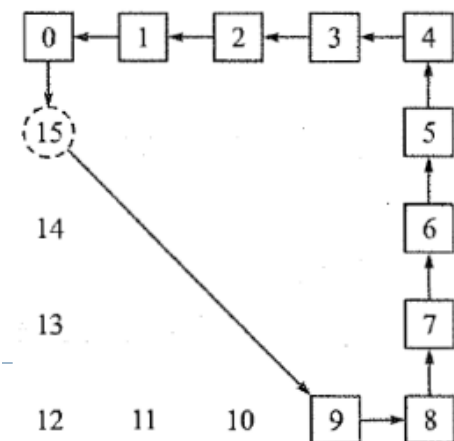
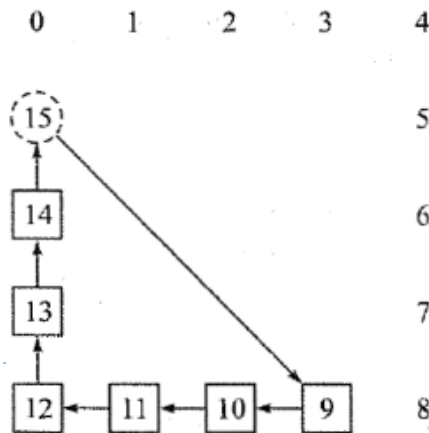


(b)



PRESETTABLE COUNTERS-6

- “ Here is another technique for modifying the count. Simply use a NAND gate to detect any of the stable states, say, state 15 (1111), and use this gate output to take *PL* low.
- “ The only time *PL* will be low is when *QD*, *QC*, *QB* and *QA* are all high, or state 15(1111). At this time, the counter will be preset to the data *PD,PC,PB,PA*.
- “ For example, suppose that *PDPCPBPA* = 1001 (the number 9). When the clock is applied, the counter will progress naturally to count 15(1111). At this time, *PL* will go low and the number 9 (1001) will be shifted into the counter.
- “ The counter will then progress through states 9, 10, 11, 12, 13, and 14, and at count 15 it will again be preset to 9. (Similarly Down Counter can designed)



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-1

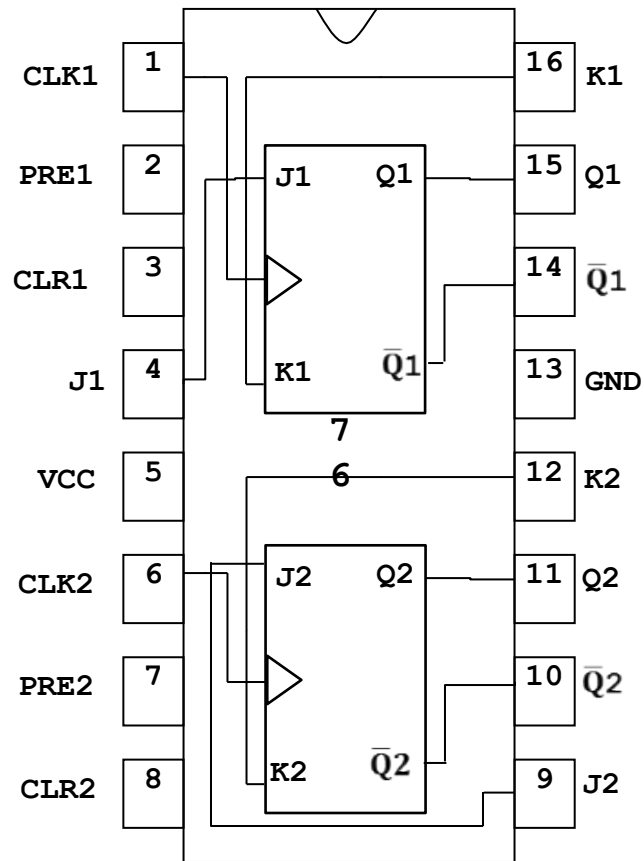
“ DESIGN OF SYNCHRONOUS COUNTERS

- “ At any given time the memory is in a state called the present state and will advance to a next state on a clock pulse a determined by conditions on the excitation lines.
- “ Steps used in the design of the counter follows. In general, these steps can be applied to any sequential circuit:
 1. Specify the counter sequence and draw a state diagram.
 2. Derive a next-state table from the state diagram.
 3. Develop a State (synthesis) Table or transition table showing the flip-flop inputs required for each transition. The transition table is always the same for a given type of flip-flop (Here will take JK flip-flop).
 4. Transfer the J and K states from the transition table to Karnaugh maps. There is a Karnaugh map for each input of each flip-flop.
 5. Group the Karnaugh map cells to generate and derive the logic expression for each flip-flop input.
 6. Implement the expressions with combinational logic and combine with the flip- flops to create the counter.

► 402 We follow the step with help of example as **Mod-6 (Lab Experiment)**

COUNTER DESIGN AS A SYNTHESIS PROBLEM-1

“ IC 7476



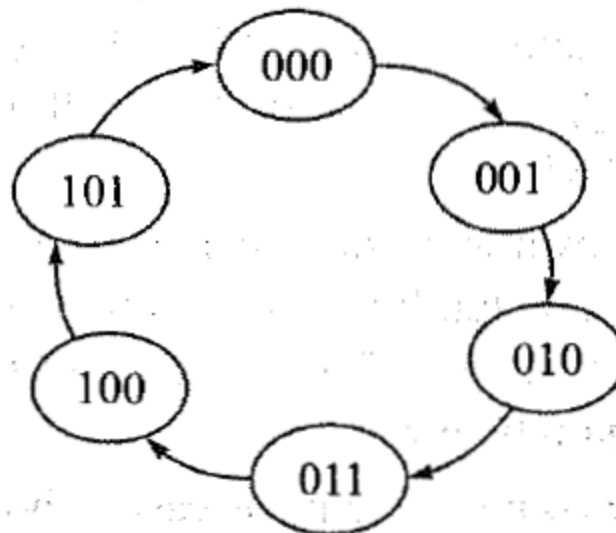
COUNTER DESIGN AS A SYNTHESIS

PROBLEM-2

“ Step 1: State Diagram

“ A state diagram shows the progression of states through which the counter advances when it is clocked.

“ Mod-6



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-3

“ Step 2: Next-State Table

- “ Derive a next- state table, which lists each state of the counter (present state) along with the corresponding next state.
- “ The next state is the state that the counter goes to from its present state upon application of a clock pulse. The next-state table is derived from the state diagram.

Present State			Next State		
C_n	B_n	A_n	C_{n+1}	B_{n+1}	A_{n+1}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0

COUNTER DESIGN AS A SYNTHESIS

PROBLEM-4

“ Step 3: State (synthesis) Table

- “ State table written from Excitation table of the flip-flop.
- “ Excitation table gives inputs need to be present when clock triggers a certain $Q_n \rightarrow Q_{n+1}$ transition of the flip-flop.
- “ State table shows the present state, next state and input state of the each flip-flop.
- “ Combine the Step 2 and 3

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

Present State			Next State			Inputs					
C_n	B_n	A_n	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1

COUNTER DESIGN AS A SYNTHESIS

PROBLEM-5

“ Step 4&5: Karnaugh Maps

- “ Karnaugh maps can be used to determine the logic required for the J and K inputs of each flip-flop in the counter. There is a Karnaugh map for the J input and a Karnaugh map for the K input of each flip-flop. In this design procedure, each cell in a Karnaugh map represents one of the present states in the count

		B_n			
		A_n 00	01	11	10
C_n	0	0	0	1	0
	1	X	X	X	X

$$J_C = A_n B_n$$

		B_n			
		A_n 00	01	11	10
C_n	0	X	X	X	X
	1	0	0	1	0

$$K_C = A_n B_n$$

		B_n			
		A_n 00	01	11	10
C_n	0	0	1	X	X
	1	0	1	X	X

$$J_B = A_n$$

		B_n			
		A_n 00	01	11	10
C_n	0	X	X	1	0
	1	X	X	1	0

$$K_B = A_n$$

		B_n			
		A_n 00	01	11	10
C_n	0	1	X	X	1
	1	1	X	X	X

$$J_A = 1$$

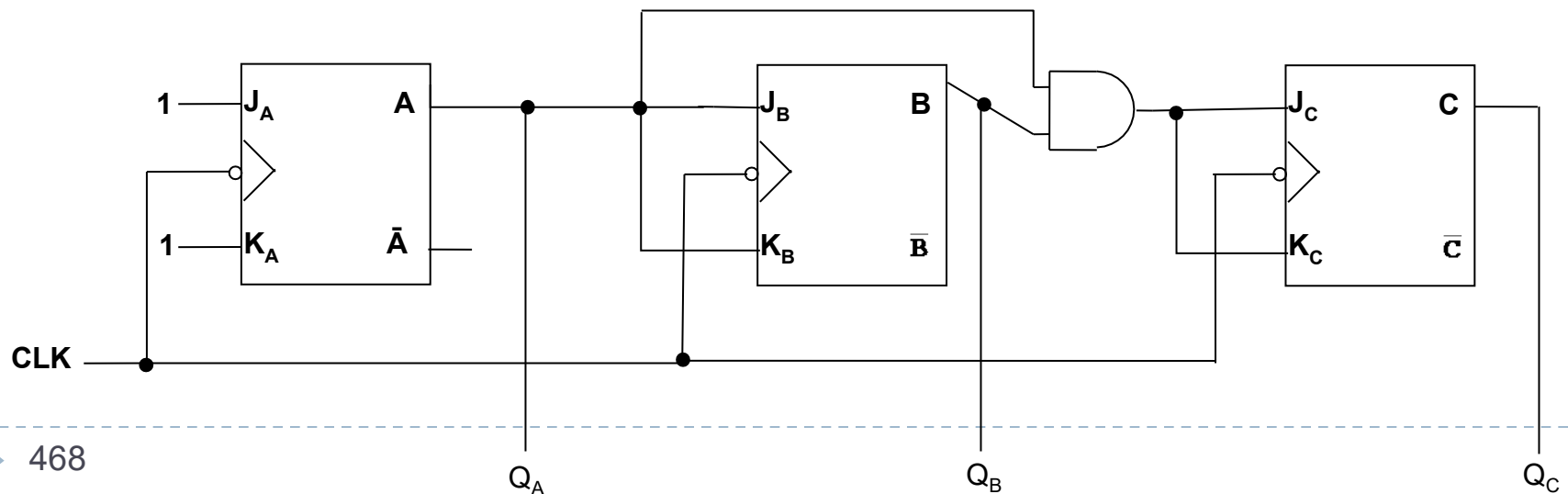
		B_n			
		A_n 00	01	11	10
C_n	0	X	1	1	X
	1	X	1	1	X

$$K_A = 1$$

COUNTER DESIGN AS A SYNTHESIS

PROBLEM-6

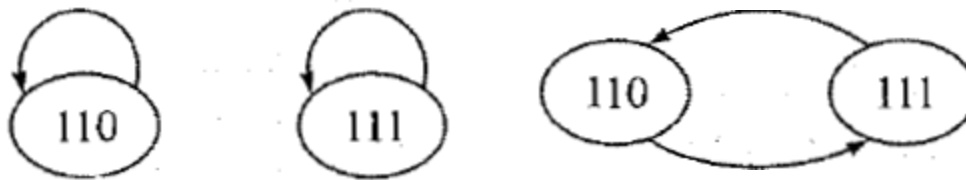
- “ **Step 6: Counter Implementation (Circuit Diagram)**
- “ From the Karnaugh maps obtained the expressions for the J and K inputs of each flip-flop.
- “ The final step is to implement the combinational logic from the expressions for the J and K inputs and connect the flip-flops to form the complete Mod-6 counter.



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-7

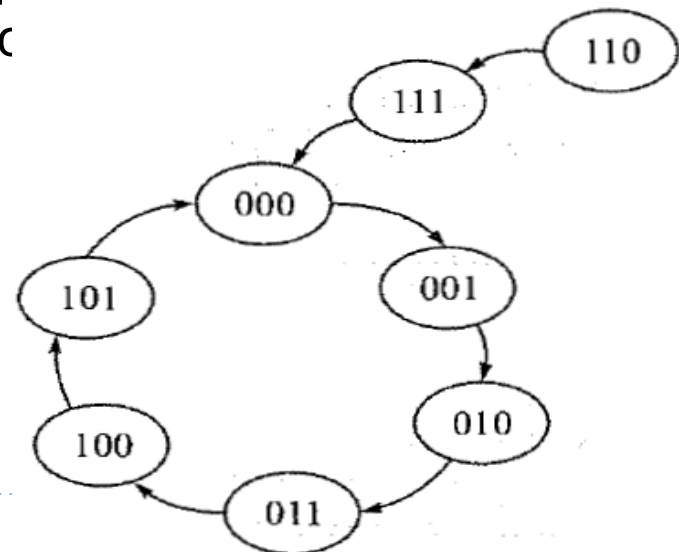
- “ An ***irregular counter*** is the one which does not follow any regular binary sequence but has N number of distinct states and thus qualifies as a modulo-N counter.
- “ What happens if the circuit for any reason goes to one of the unused state? Does it come back to any of the valid counting state or in the worst case gets locked as shown in below Fig.?



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-7

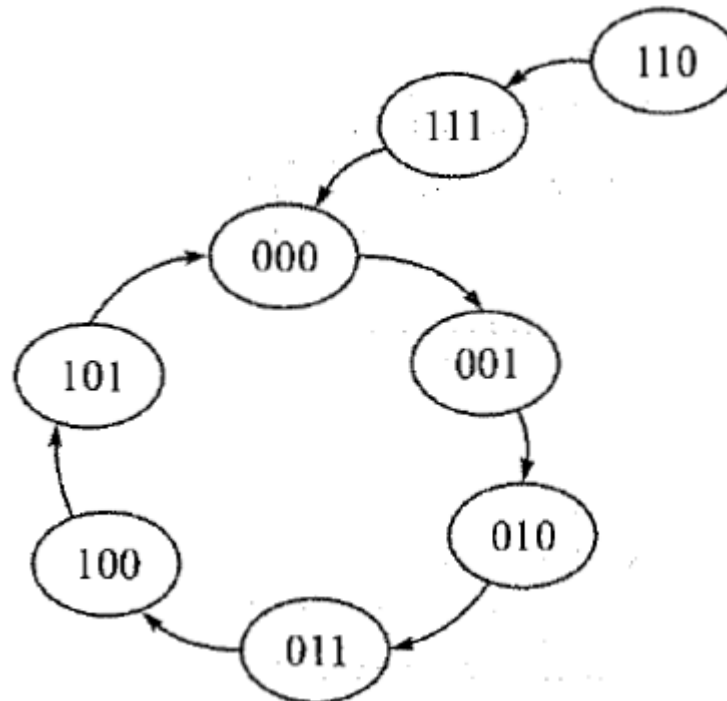
- “ For Mod-6, initializing the designed circuit with 110 or 111 unused state we find that they get back to counting sequence as shown in below Fig.
- “ **Lock-out condition** : Lock-out of a counter occurs when the counter remains locked into unused states and does not function properly.
- “ However, a designer may not leave unused states to chances and want them to follow certain course if the circuit accidentally enters into one of them .
- “ This can be achieved by Self-correcting as the circuit comes out on its own from an invalid state to a valid c



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-7

- “ Design a *self-correcting* modulo-6 counter, in which all the unused state leads to state $CBA = 000$.
- “ **Step-1**



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-8

“ Step-2 & 3

Present State			Next State			Inputs					
C_n	B_n	A_n	C_{n+1}	B_{n+1}	A_{n+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	0	X	1	0	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

COUNTER DESIGN AS A SYNTHESIS

PROBLEM-9

“ Step-4 & 5

		$B_n A_n$			
		00	01	11	10
C_n	0	0	0	1	0
	1	×	×	×	×

$$J_C = B_n A_n$$

		$B_n A_n$			
		00	01	11	10
C_n	0	×	×	×	×
	1	0	1	1	1

$$K_C = A_n + B_n$$

		$B_n A_n$			
		00	01	11	10
C_n	0	0	1	×	×
	1	0	0	×	×

$$J_B = \bar{C}_n A_n$$

		$B_n A_n$			
		00	01	11	10
C_n	0	×	×	1	0
	1	×	×	1	1

$$K_B = A_n + C_n$$

		$B_n A_n$			
		00	01	11	10
C_n	0	1	×	×	1
	1	1	×	×	0

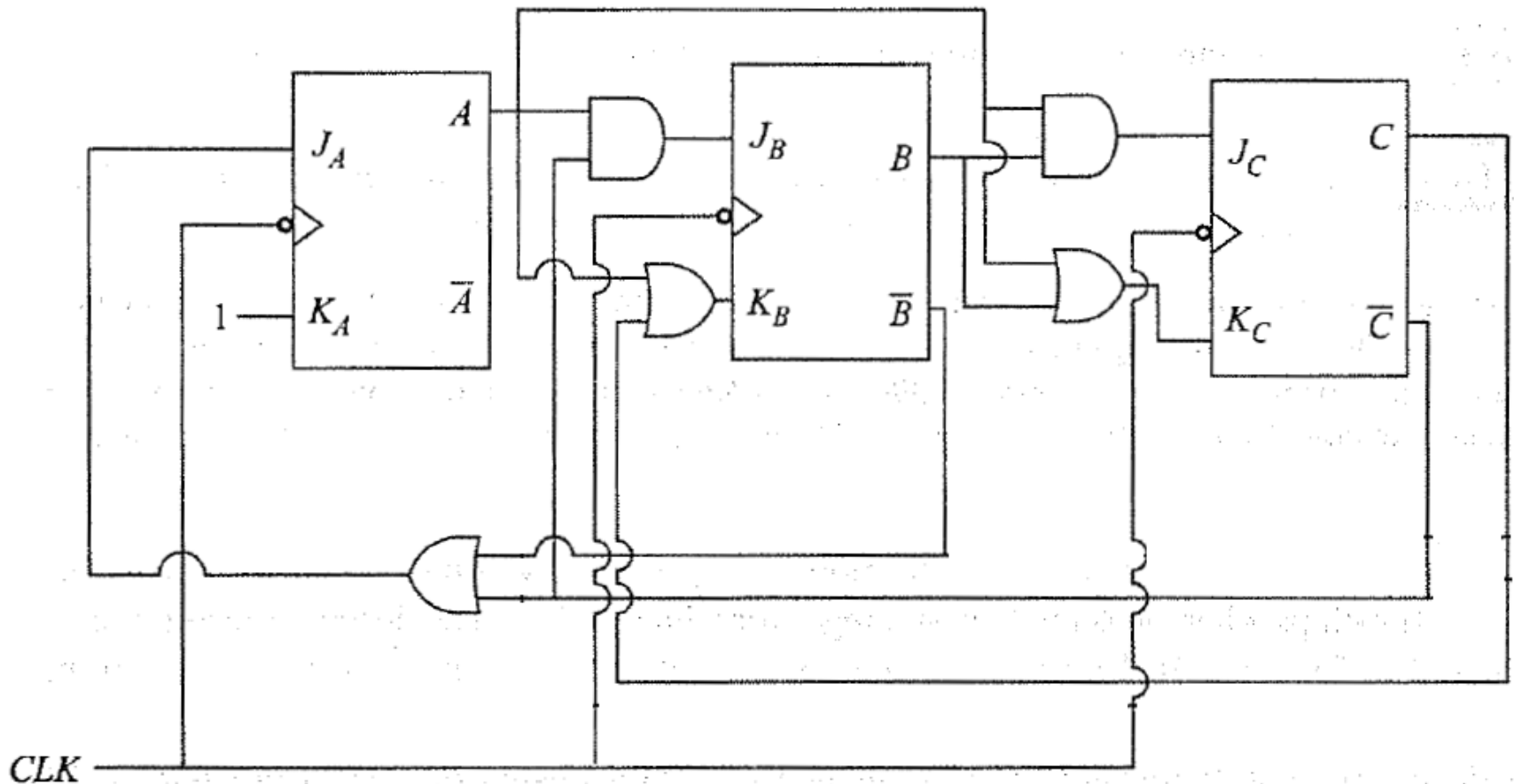
$$J_A = \bar{C}_n + \bar{B}_n$$

		$B_n A_n$			
		00	01	11	10
C_n	0	×	1	1	×
	1	×	1	1	×

$$K_A = 1$$

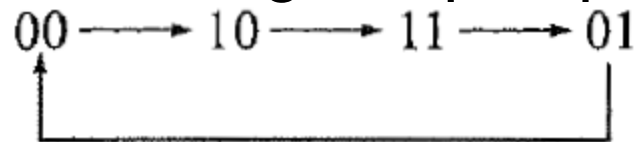
COUNTER DESIGN AS A SYNTHESIS PROBLEM-10

“ Step-6

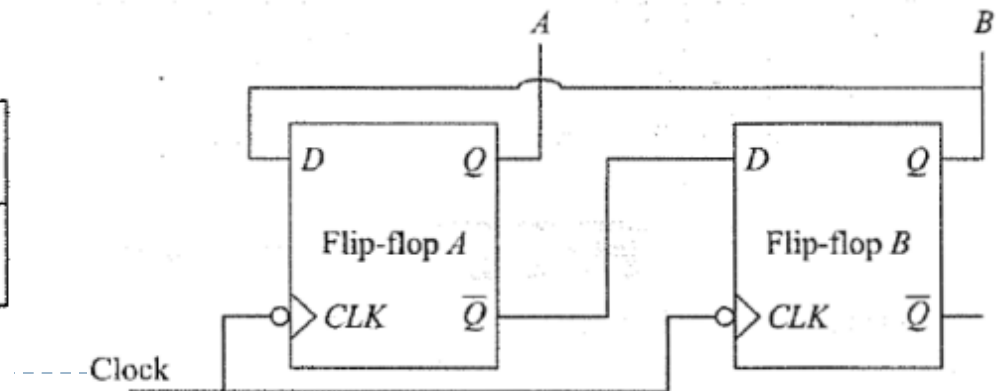
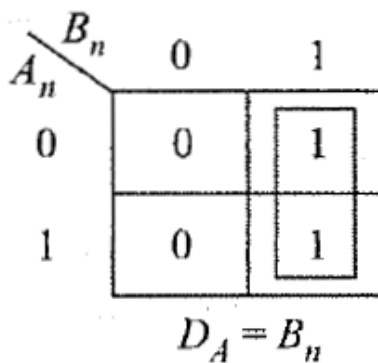
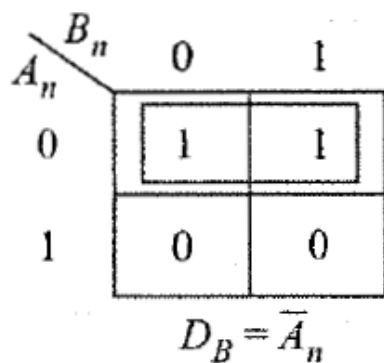


COUNTER DESIGN AS A SYNTHESIS PROBLEM-11

- “ Design a modulo-4 irregular counter with following counting sequence using D flip-flop.



B_n	A_n	B_{n+1}	A_{n+1}	D_B	D_A
0	0	1	0	1	0
0	1	0	0	0	0
1	0	1	1	1	1
1	1	0	1	0	1



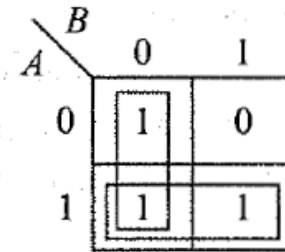
COUNTER DESIGN AS A SYNTHESIS

PROBLEM-12

- “ Show how a modulo-4 counter designed with two flip-flops can generate a repetitive sequence of binary word '1101' with minimum number of memory elements?
- “ The corresponding output is $1 \rightarrow 1 \rightarrow 0 \rightarrow 1$. As shown in Figure the sequence '1101' will be generated repetitively by Y Karnaugh Map representation of Y and will get $Y =$

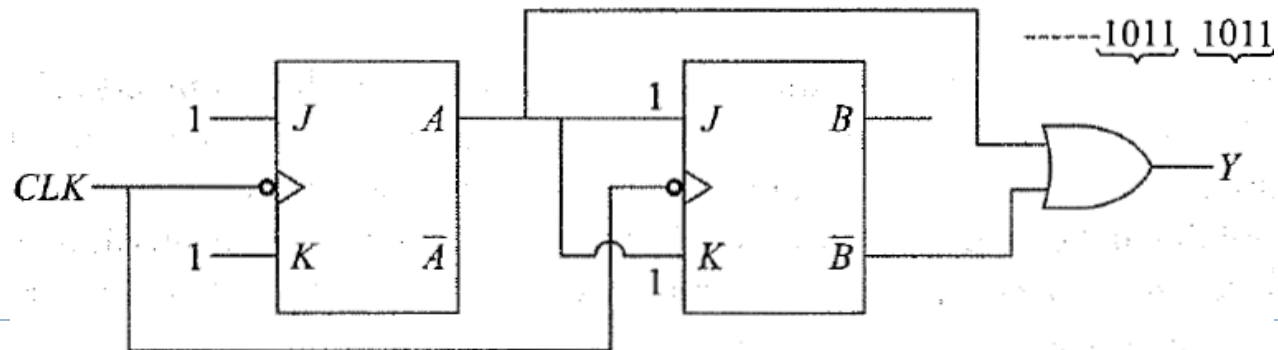
A	B	Y
0	0	1
0	1	1
1	0	0
1	1	1

(a)



$$Y = A + \bar{B}$$

(b)

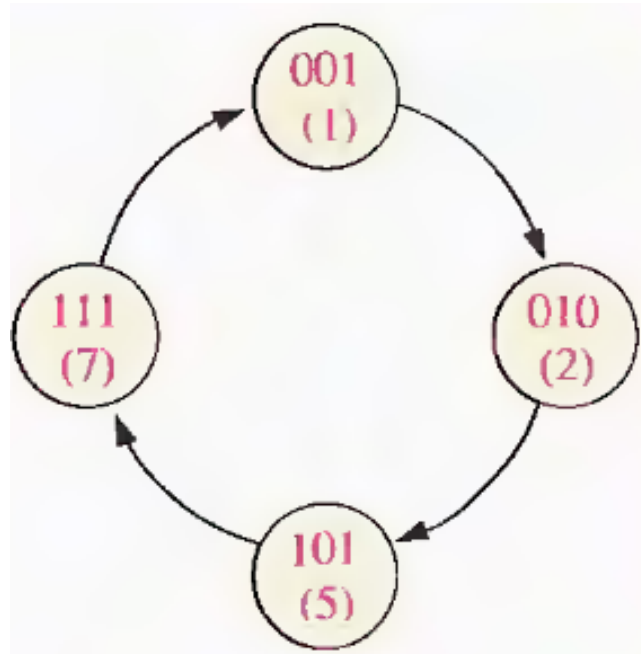


(c)

COUNTER DESIGN AS A SYNTHESIS

PROBLEM-13

- “ Design a counter with the irregular binary count sequence shown in the state diagram below. Use J-K flip-flops.



COUNTER DESIGN AS A SYNTHESIS

PROBLEM-14

“ State Table

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

Present State			Next State			Inputs					
Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	1	1	X	X	1	1	X
1	0	1	1	1	1	X	0	1	X	X	0
1	1	1	0	0	1	X	1	X	1	X	0

COUNTER DESIGN AS A SYNTHESIS PROBLEM-15

“ K-Map Simplification

Q_2Q_1		Q_0	
		0	1
00		X	0
01		1	X
11		X	X
10		X	X

$$J_2 = Q_1$$

Q_2Q_1		Q_0	
		0	1
00		X	1
01		X	X
11		X	X
10		X	1

$$J_1 = 1$$

Q_2Q_1		Q_0	
		0	1
00		X	X
01		1	X
11		X	X
10		X	X

$$J_0 = 1$$

Q_2Q_1		Q_0	
		0	1
00		X	X
01		X	X
11		X	1
10		X	0

$$K_2 = Q_1$$

Q_2Q_1		Q_0	
		0	1
00		X	X
01		1	X
11		X	1
10		X	X

$$K_1 = 1$$

Q_2Q_1		Q_0	
		0	1
00		X	1
01		X	X
11		X	0
10		X	0

$$K_0 = \bar{Q}_2$$

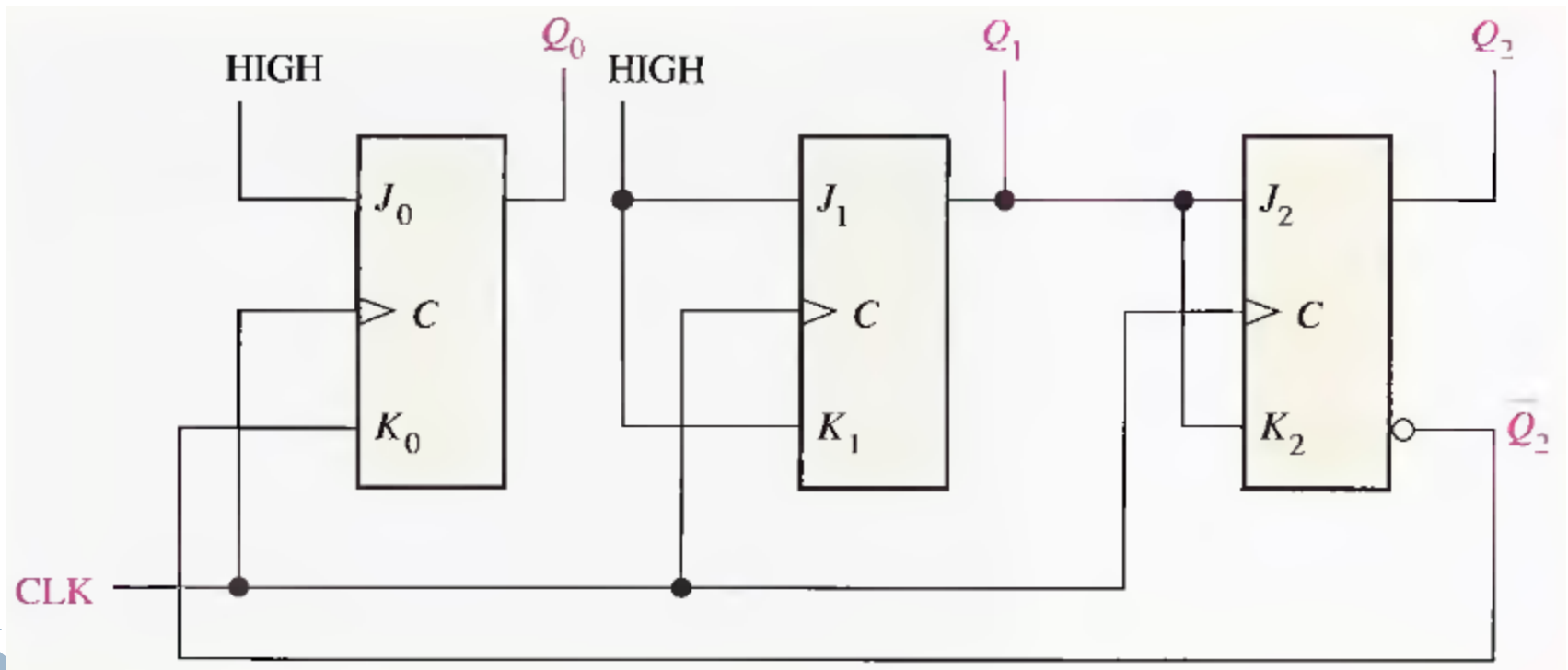
COUNTER DESIGN AS A SYNTHESIS PROBLEM-16

“ Circuit Diagram

$$J_0 = 1, K_0 = \overline{Q_2}$$

$$J_1 = K_1 = 1$$

$$J_2 = K_2 = Q_1$$



Difference between Asynchronous and Synchronous Counter

Asynchronous Counter

- 1) Clock input is applied to LSB flip-flop. The output of first flip-flop is connected as clock to next FF.
- 2) All Flip-Flops are toggle flip-flop.
- 3) Speed depends on no. of flip-flop used for n bit .
- 4) No extra Logic Gates are required.
- 5) Cost is less.

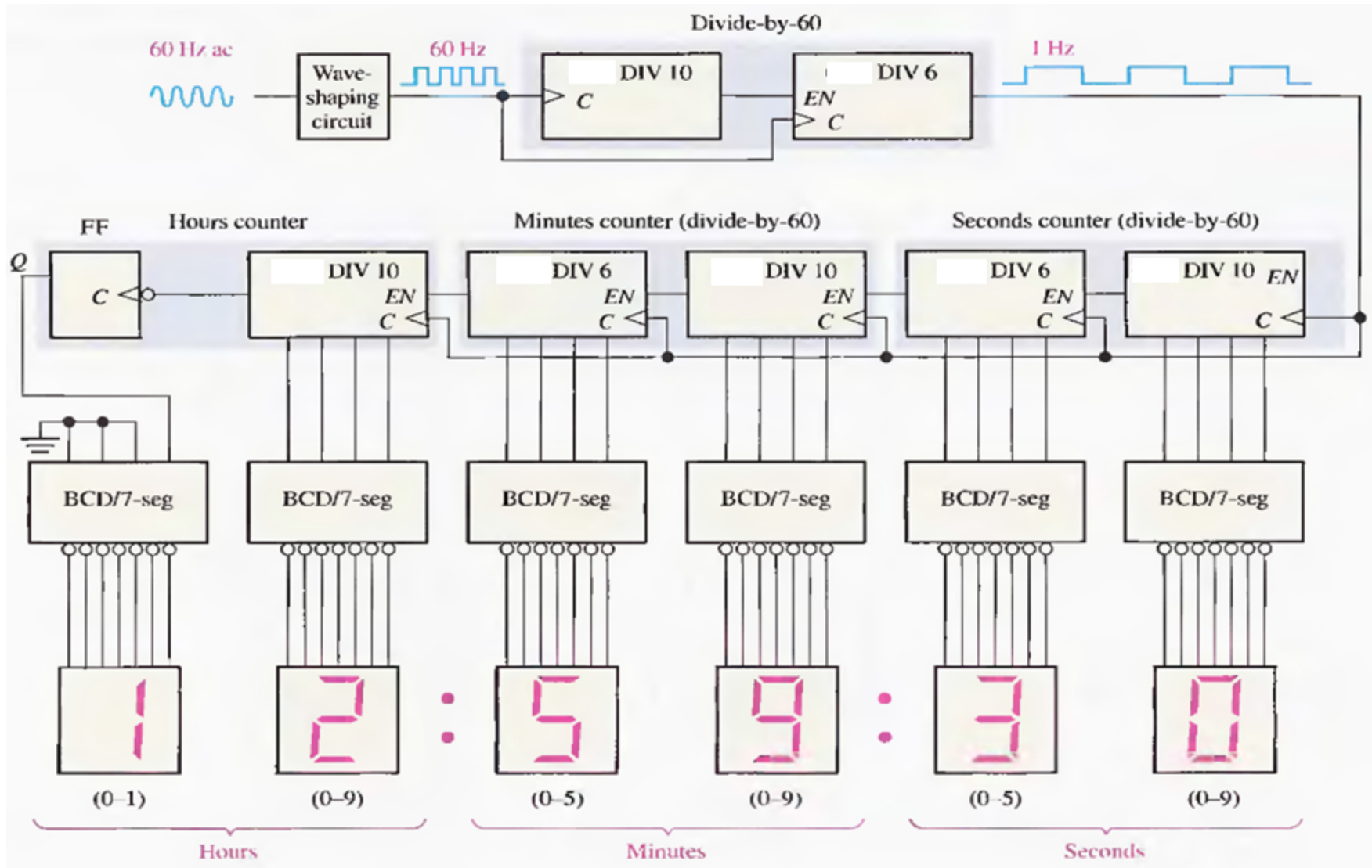
Synchronous Counter

- 1) Clock input is common to all flip-flop.
- 2) Any flip-flop can be used.
- 3) Speed is independent of no. of flip-flop used.
- 4) Logic Gates are required based on design.
- 5) Cost is more.

A DIGITAL CLOCK-1

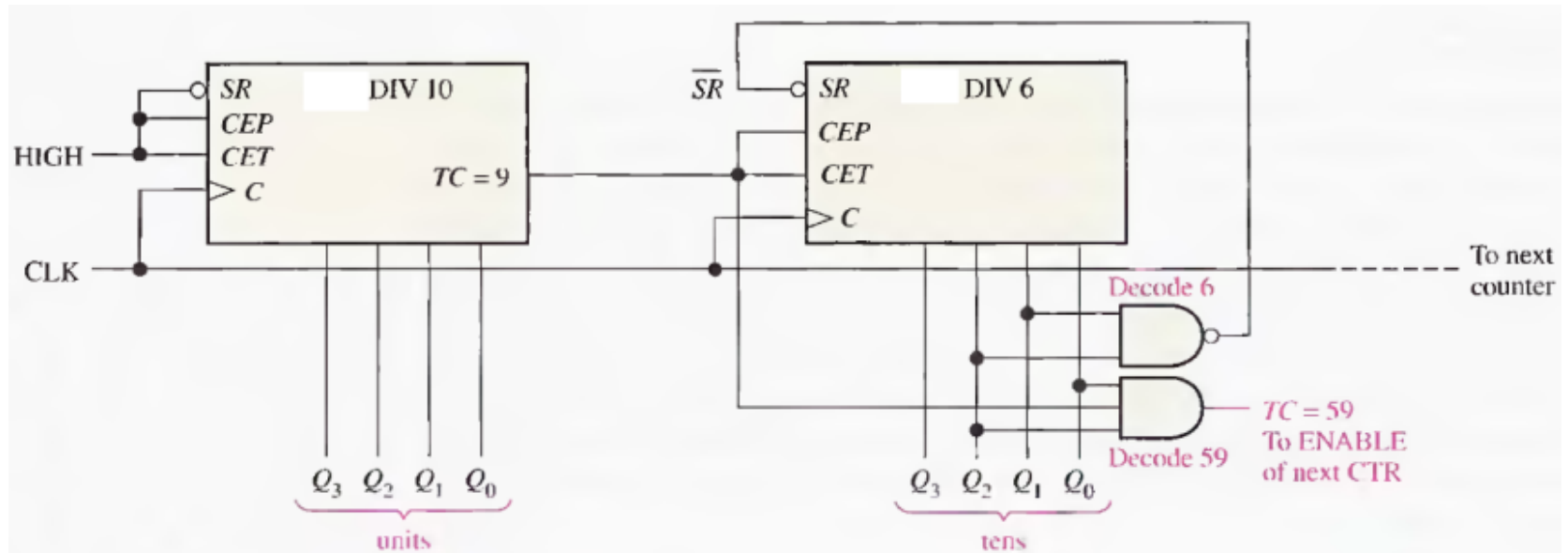
- “ Logic diagram of a digital clock that displays seconds, minutes, and hours is shown in next slide.
- “ First, a 60 Hz sinusoidal ac voltage is converted to a 60 Hz pulse waveform and divided down to a 1 Hz pulse waveform by a divide-by-60 counter formed by a divide-by-10 counter followed by a divide-by-6 counter.
- “ Both the seconds and minutes counts are also produced by divide-by-60 counters the details of which are shown in next slide. These counters count from 0 to 59 and then recycle to 0; synchronous decade counters are used in this particular implementation.
- “ Notice that the divide-by-6 portion is formed with a decade counter with a truncated sequence achieved by using the decoder count 6 to asynchronously clear the counter. The terminal count, 59, is also decoded to enable the next counter in the chain.
- “ The hours counter is implemented with a decade counter and a flip-flop as shown in next slide. Consider that initially both the decade counter and the flip-flop are RESET, and the decode-12 gate and decode-9 gate outputs are HIGH. The decade counter advances through all of its states from zero to nine, and on the clock pulse that recycles it from nine back to zero, the flip-flop goes to the SET state ($J = 1$, $K = 0$). This illuminates a 1 on the tens-of-hours display. The total count is now ten (the decade counter is in the zero state and the flip-flop is SET).

A DIGITAL CLOCK-2



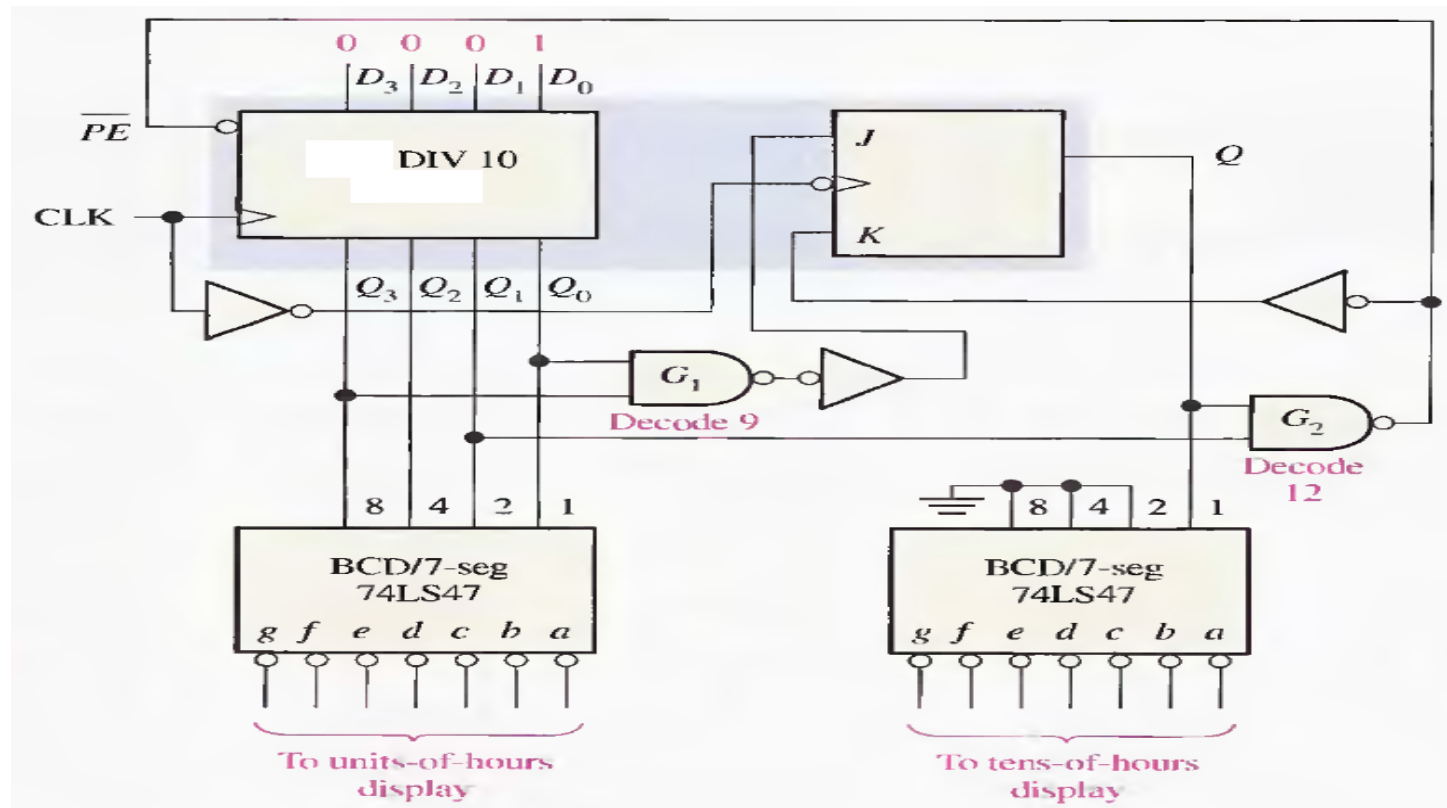
A DIGITAL CLOCK-3

- “ Logic diagram of typical divide-by-60 counter using synchronous decade counters. Note that the outputs are in binary order (the right-most bit is the LSB).



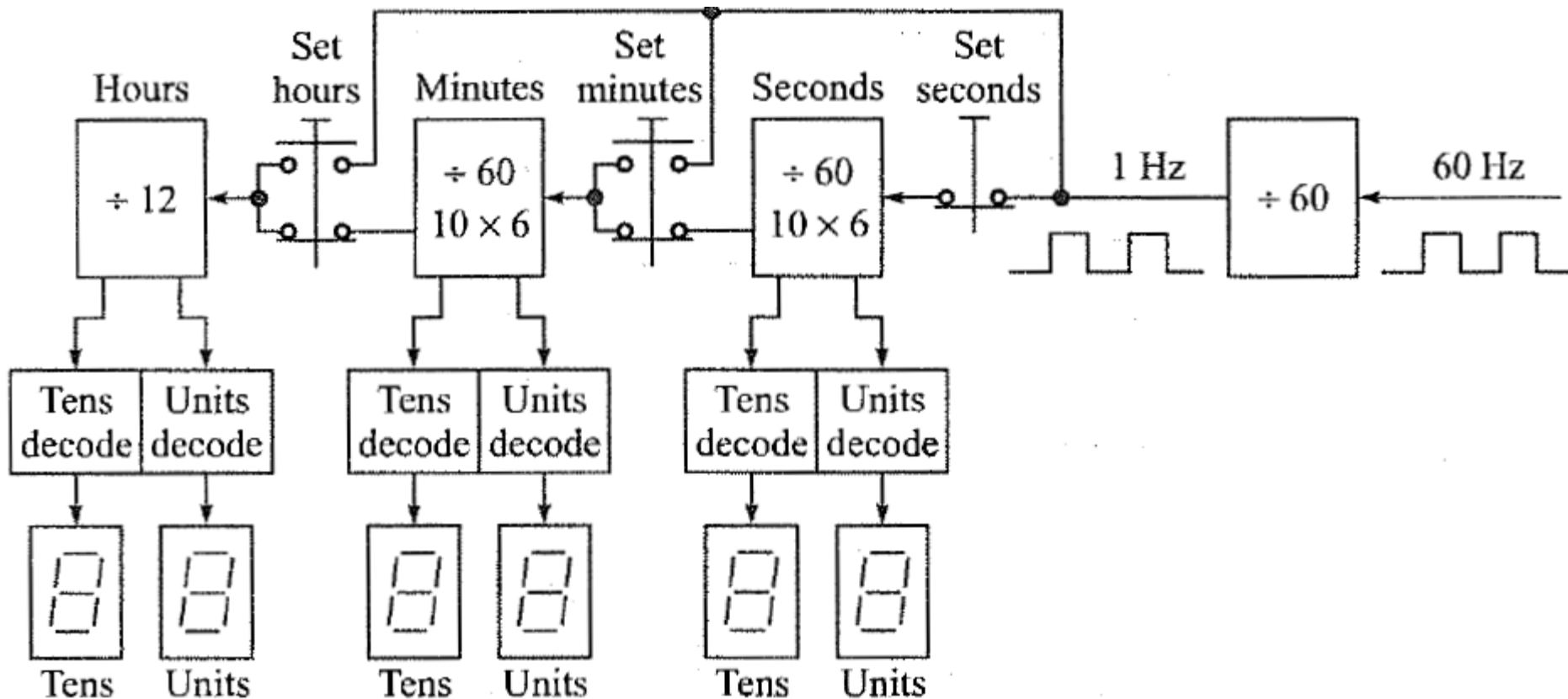
A DIGITAL CLOCK-4

- “ Logic diagram for hours counter and decoders. Note that on the counter inputs and outputs, the right-most bit is the LSB.



A DIGITAL CLOCK-5

“ Setting the clock can be quite easily accomplished by means of the SET push-buttons shown in diagram.



COUNTER DESIGN USING HDL-1

- “ Counter design in HDL is straight forward if one uses arithmetic operator $+$ and $-$ that corresponds to binary addition and subtraction respectively.

COUNTER DESIGN USING HDL-2

- Mod-8 Counter

```
module UC(Clock, Reset,Q);  
input Clock, Reset;  
output [2:0] Q;  
//modulo 8 requires 3 flip-flop  
reg [2 : 0]Q;  
always@ (negedge Clock or negedge Reset)  
if (~Reset) Q=3'b0;  
else Q= Q+1;  
endmodule
```

PROBLEM SOLVING WITH MULTIPLE METHODS-1

- “ Design a self correcting modulo-3 down counter.
- “ We need 2 flip-flop, say B and A for this purpose which has 4 states. Let the down counter count like $10 \rightarrow 01 \rightarrow 00 \rightarrow 10$ and undesired state 11 corrects itself to 10. The excitation table shown below is used for the design purpose.
- “ State table for the self correcting modulo 3 down counter and required inputs using
 - “ Design with SR flip-flops
 - “ Design with *JK* flip-flops
 - “ Design with *D* flip-flops
 - “ Design with *T* flip-flops

PROBLEM SOLVING WITH MULTIPLE METHODS-2

Present State $B_n A_n$	Next State $B_{n+1} A_{n+1}$	$S_B R_B$	$S_A R_A$	D_B	D_A	$J_B K_B$	$J_A K_A$	T_B	T_A
0 0	1 0	1 0	0 X	1	0	1 X	0 X	1	0
0 1	0 0	0 X	0 1	0	0	0 X	X 1	0	1
1 0	0 1	0 1	1 0	0	1	X 1	1 X	1	1
1 1	1 0	X 0	0 1	1	0	X 0	X 1	0	1

$A_n \backslash B_n$	0	1
0	1	0
1	0	X

$$S_B = A'_n B'_n$$

A_n	0	1
B_n	0	X
1	1	0

$$R_B = A'_n B_n$$

$A_n \backslash B_n$	0	1
0	0	0
1	1	0

$$S_A = A'_n B_n$$

A_n	0	1
B_n	0	1
0	X	1
1	0	1

$$R_A = A_n$$

A_n	0	1
B_n 0	1	0
1	0	1

$$D_B = A'_n B'_n + A_n B_n$$

A_n	0	1
B_n 0	0	0
1	1	0

$$D_A = A'_n B_n$$

		A_n	
		0	1
B_n	0	1	0
	1	X	X

$$J_B = A'_n$$

A_n	0	1
B_n	0	1
0	X	X
1	1	0

$$K_B = A'_n$$

A_n	0	1
B_n	0	1
0	0	X
1	1	X

$$J_A = B_n$$

$A_n \backslash B_n$	0	1
0	X	1
1	X	1

$$K_A = 1$$

A_n	0	1
B_n	0	0
	1	0

$$T_B = A'_n$$

$A_n \backslash B_n$	0	1
0	X	1
1	1	1

$$T_A = A_n + B_n$$

VARIABLE, RESISTOR NETWORKS-1

- “ Digital signal into an equivalent analog signal is to change the n digital voltage levels into one equivalent analog voltage.
- “ **Binary Equivalent Weight**
- “ To change the possible digital signals into equivalent analog voltages.
- “ E.g. For the 3-bit binary signal, the smallest number represented is 000, let make this equal to 0V. The largest number is 111: let make this equal to +7 V.
- “ So between 000 and 111 there are seven discrete levels to be defined. Therefore, it will be convenient to divide the analog signal into seven levels.

2^2	2^1	2^0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

VARIABLE, RESISTOR NETWORKS-2

- “ Smallest incremental change in the digital signal is represented by the least-significant bit (LSB), 2^0 . So to have this bit cause a change in the analog output that is equal to one seventh of the full-scale analog output voltage.
- “ The resistive divider will then be designed such that a 1 in the 2^0 position will cause $+ 7 \times 1/7 = + 1 \text{ V}$ at the output.
- “ For $2^1 = +2\text{V}$
- “ and $2^2 = +4\text{V}$ so continued for 3-bit binary number.
- “ The binary equivalent weight assigned to the LSB is $(2^n - 1)$, where n is the number of bits. The remaining weights are found by multiplying by 2, 4, 8, and so

VARIABLE, RESISTOR NETWORKS-3

Bit	Weight
2^0	1/7
2^1	2/7
2^2	4/7
Sum	7/7

Digital input			Analog output
0	0	0	+0 V
0	0	1	+1 V
0	1	0	+2 V
0	1	1	+3 V
1	0	0	+4 V
1	0	1	+5 V
1	1	0	+6 V
1	1	1	+7 V

VARIABLE, RESISTOR NETWORKS-4

- “ Find the binary equivalent weight of each bit in a 4-bit system.

Solution The LSB has a weight of $1/(2^4 - 1) = 1/(16 - 1) = \frac{1}{15}$, or 1 part in 15. The second LSB has a weight of $2 \times \frac{1}{15} = \frac{2}{15}$. The third LSB has a weight of $4 \times \frac{1}{15} = \frac{4}{15}$, and the MSB has a weight of $8 \times \frac{1}{15} = \frac{8}{15}$. As a check, the sum of the weights must equal 1. Thus $\frac{1}{15} + \frac{2}{15} + \frac{4}{15} + \frac{8}{15} = \frac{15}{15} = 1$.

Bit	Weight
2^0	$1/15$
2^1	$2/15$
2^2	$4/15$
2^3	$8/15$
Sum	$15/15$

VARIABLE, RESISTOR NETWORKS-5

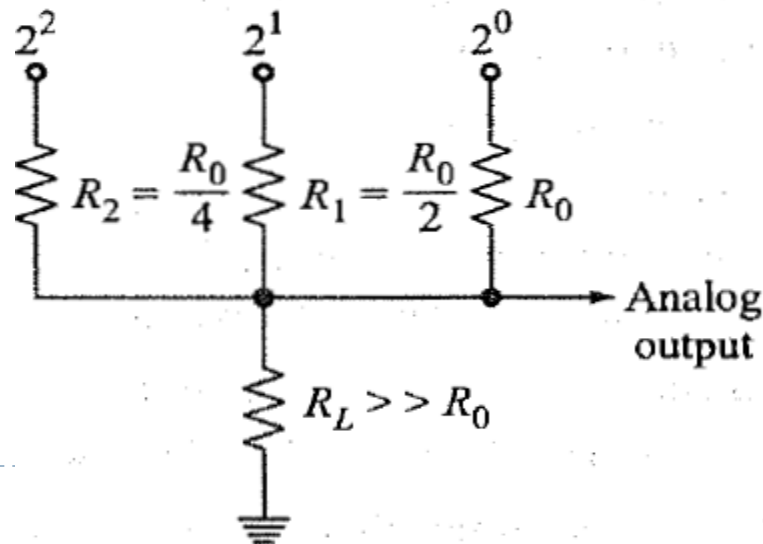
“ Resistive Divider

“ The other desired voltage levels are shown in Figure; they, too, are additive combinations of voltages.

Digital input			Analog output
0	0	0	+0 V
0	0	1	+1 V
0	1	0	+2 V
0	1	1	+3 V
1	0	0	+4 V
1	0	1	+5 V
1	1	0	+6 V
1	1	1	+7 V

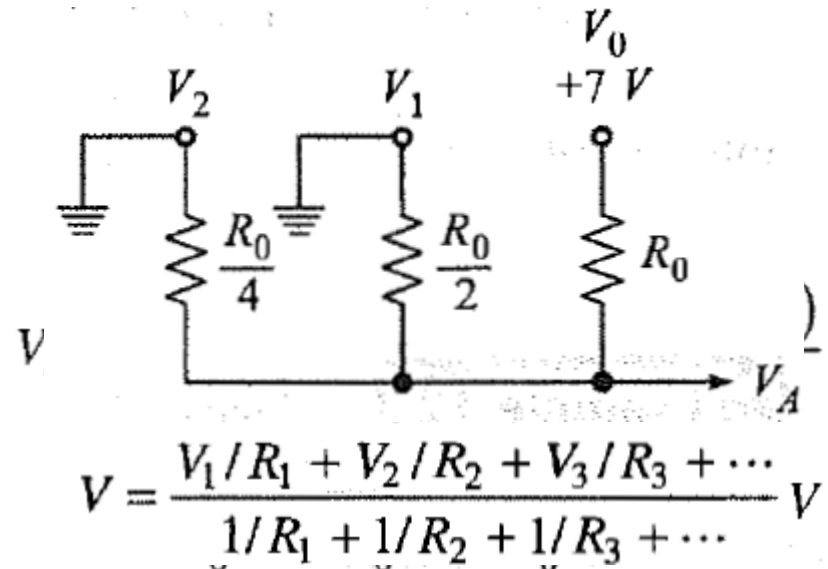
VARIABLE, RESISTOR NETWORKS-6

- “ Thus the resistive divider must do two things in order to change the digital input into an equivalent analog output voltage:
 - “ The 2^0 bit must be changed to + 1 V, and 2^1 bit must be changed to +2 V, and 2^2 bit must be changed to +4 V.
 - “ These three voltages representing the digital bits must be summed together to form the analog output voltage.



VARIABLE, RESISTOR NETWORKS-7

- “ Assume that the digital input signal 001 is applied to this network.
- “ Recalling that 0 = 0 V and 1 = + 7 V, so the equivalent circuit shown in Figure.
- “ Resistance R_L is considered large and is neglected.
- “ The analog output voltage V_A can be most easily found by use of Millman's theorem,
- “ Which states that the voltage appearing at any node in a resistive network is equal to the summation of the currents entering the node (found by assuming that the node voltage is zero) divided by the summation of the conductances connected to the node.
- “ So for 001 V_A is equal to



VARIABLE, RESISTOR NETWORKS-8

- “ A resistive divider can be built to change a digital voltage into an equivalent analog voltage. The following criteria can be applied to this divider:
- “ There must be one input resistor for each digital bit.
- “ Beginning with the LSB, each following resistor value is one-half the size of the previous resistor.
- “ The full-scale output voltage is equal to the positive voltage of the digital input signal. (The divider would work equally well with input voltages of 0 and $-V$.)
- “ The LSB has a weight of $1/(2^n - 1)$, where n is the number of input bits.
- “ The change in output voltage due to a change in the LSB is equal to $V/(2^n - 1)$, where V is the digital input voltage level.
- “ The output voltage V_A can be found for any digital input signal by using the following:

$$V_A = \frac{V_0 2^0 + V_1 2^1 + V_2 2^2 + V_3 2^3 + \dots + V_{n-1} 2^{n-1}}{2^n - 1}$$

- “ where $V_0, V_1, V_2, V_3, \dots, V_{n-1}$ are the digital input voltage levels (0 or V) and n is the number of input bits.

VARIABLE, RESISTOR NETWORKS-9

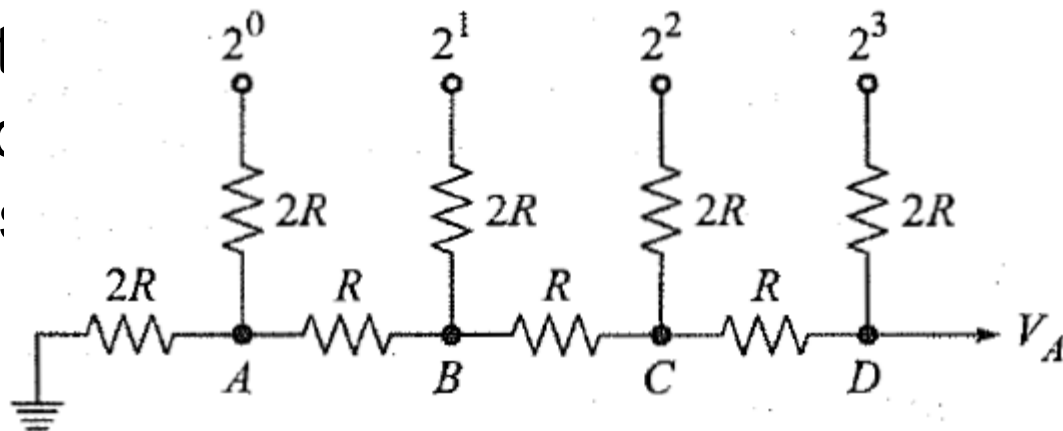
- “ This resistive divider has two serious drawbacks.
- “ The first is the fact that each resistor in the network has a different value. Since these dividers are usually constructed by using precision resistors, the added expense becomes unattractive.
- “ The resistor used for the MSB is required to handle a much greater current than that used for the LSB resistor. For example, in a 10-bit system, the current through the MSB resistor is approximately 500 times as large as the current through the LSB resistor.
- “ For these reasons, a second type of resistive network, called a *ladder*, has been developed.

VARIABLE, RESISTOR NETWORKS-10

“ BINARY LADDERS

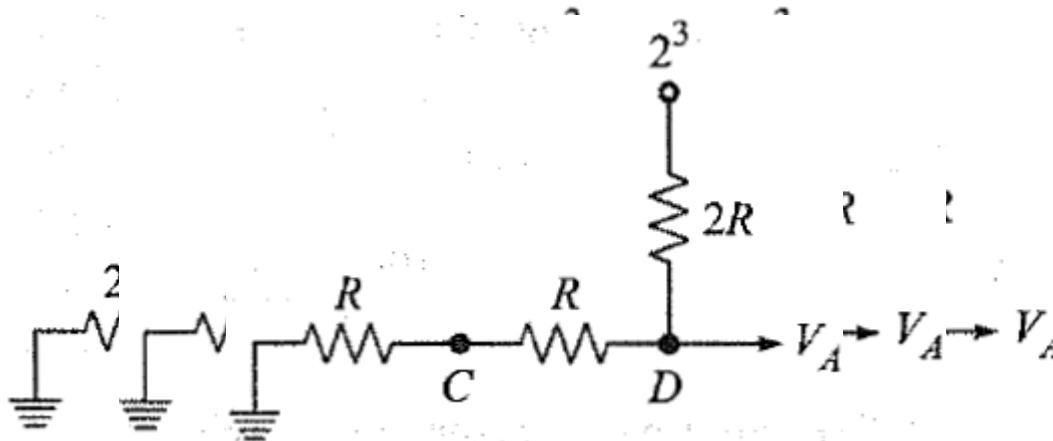
- “ The *binary ladder* is a resistive network whose output voltage is a properly weighted sum of the digital inputs.
- “ Such a ladder, designed for 4 bits, is shown in Figure.
- “ It is constructed of resistors that have only two values.
- “ The left

ated in a
adder (the



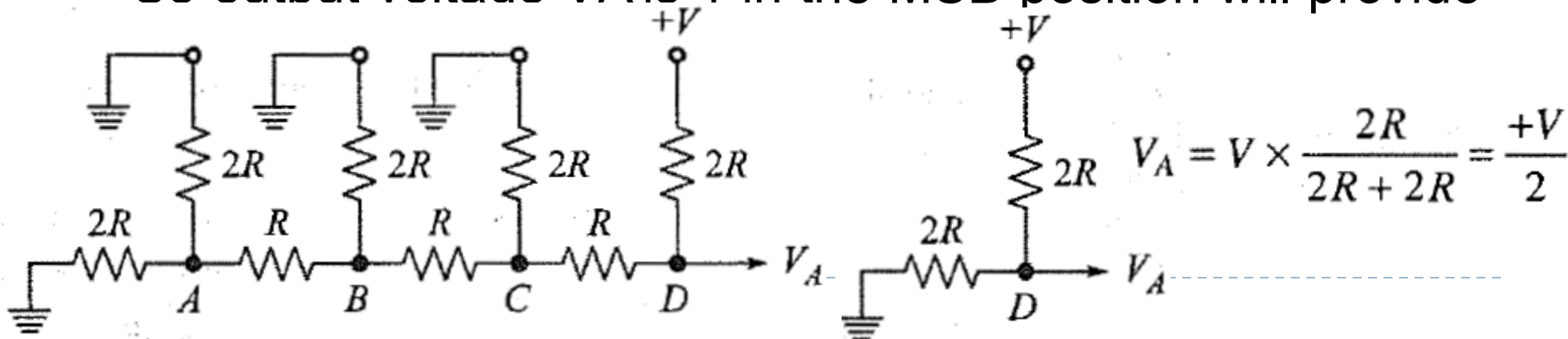
VARIABLE, RESISTOR NETWORKS-11

- “ Assuming that all the digital inputs are at ground.
- “ Beginning at node A , the total resistance looking into the terminating resistor is $2R$.
- “ The total resistance looking out toward the 2^0 input is also $2R$. These two resistors can be combined to form an equivalent resistor of value R as shown in Figure below.



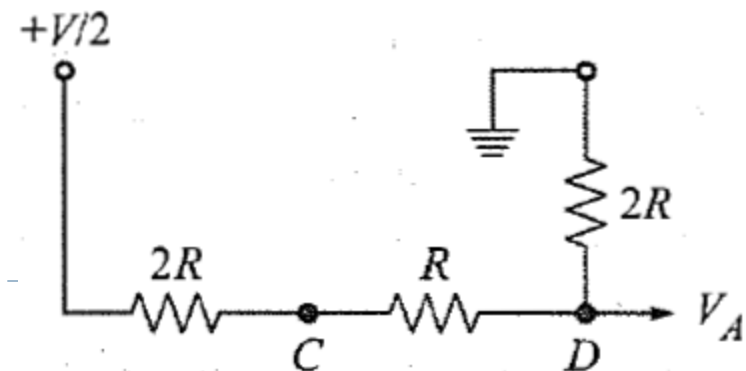
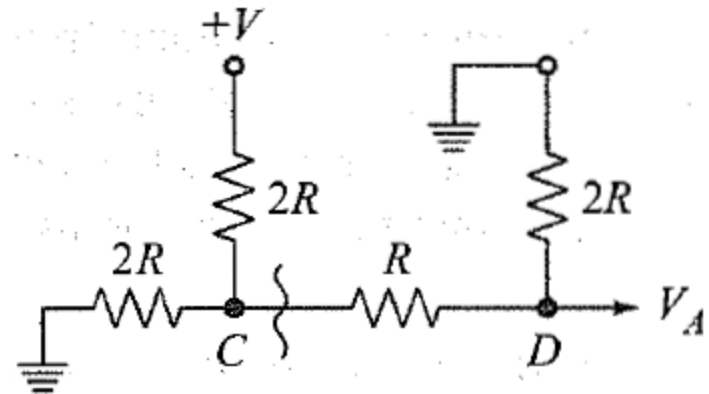
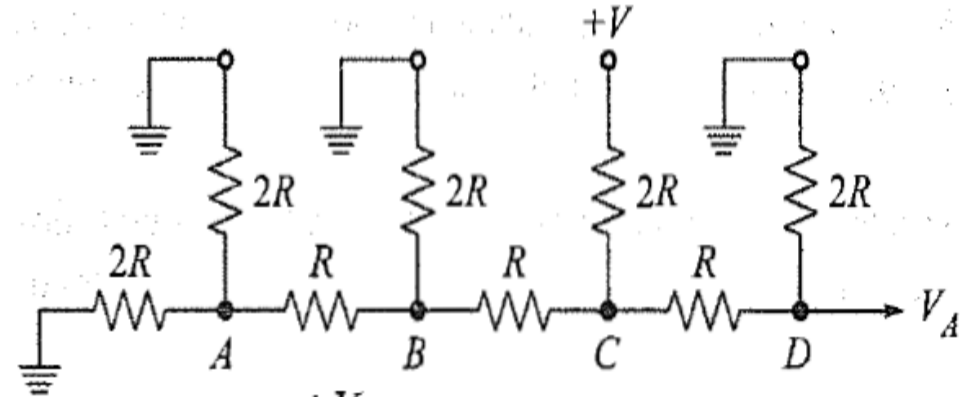
VARIABLE, RESISTOR NETWORKS-12

- “ We can conclude that the total resistance looking from any node back toward the terminating resistor or out toward the digital input is $2R$.
- “ Notice that this is true regardless of whether the digital inputs are at ground or $+V$.
- “ Assume that the digital input signal is 1000. With this input signal, the binary ladder can be drawn as shown in below.
- “ So output voltage V_A is 1 in the MSB position will provide



VARIABLE, RESISTOR NETWORKS-13

- “ Binary ladder with a digital input of 0100
- “ Partially reduced equivalent circuit
- “ Final equivalent circuit using Thevenin's theorem.
- “ Output voltage is second MSB provides an output voltage of $+V/4$.



$$503 V_A = \frac{+V}{2} \times \frac{2R}{R + R + 2R} = \frac{+V}{4}$$

VARIABLE, RESISTOR NETWORKS-14

- “ This process can be continued, and it can be shown that the third MSB provides an output voltage of $+V/8$, the fourth MSB provides an output voltage of $+V/16$, and so on.
- “ The output voltages for the binary ladder are summarized in figure; notice that each digital input is transformed into a properly weighted

Bit position	Binary weight	Output voltage
MSB	$1/2$	$V/2$
2d MSB	$1/4$	$V/4$
3d MSB	$1/8$	$V/8$
4th MSB	$1/16$	$V/16$
5th MSB	$1/32$	$V/32$
6th MSB	$1/64$	$V/64$
7th MSB	$1/128$	$V/128$
.	.	.
.	.	.
.	.	.
Nth MSB	$1/2^N$	$V/2^N$

VARIABLE, RESISTOR NETWORKS-15

“ In equation form, the output voltage is given by

“ $V_A = \frac{V}{2} + \frac{V}{4} + \frac{V}{8} + \frac{V}{16} + \dots + \frac{V}{2^n}$ number $V_A = V \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots + \frac{1}{2^n} \right)$

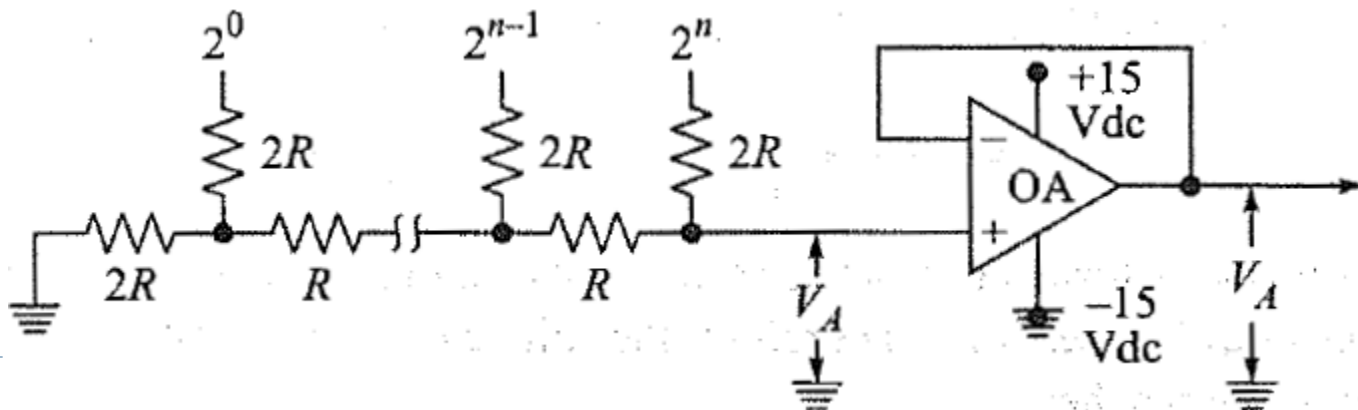
“ This equation can be simplified somewhat by factoring and collecting terms. The output voltage can then be given in the form

$$V_A = \frac{V_0 2^0 + V_1 2^1 + V_2 2^2 + V_3 2^3 + \dots + V_{n-1} 2^{n-1}}{2^n}$$

“ where $V_0, V_1, V_2, \dots, V_{n-1}$ are the digital input voltage levels. Equation can be used to find the output voltage from the ladder for any digital input signal.

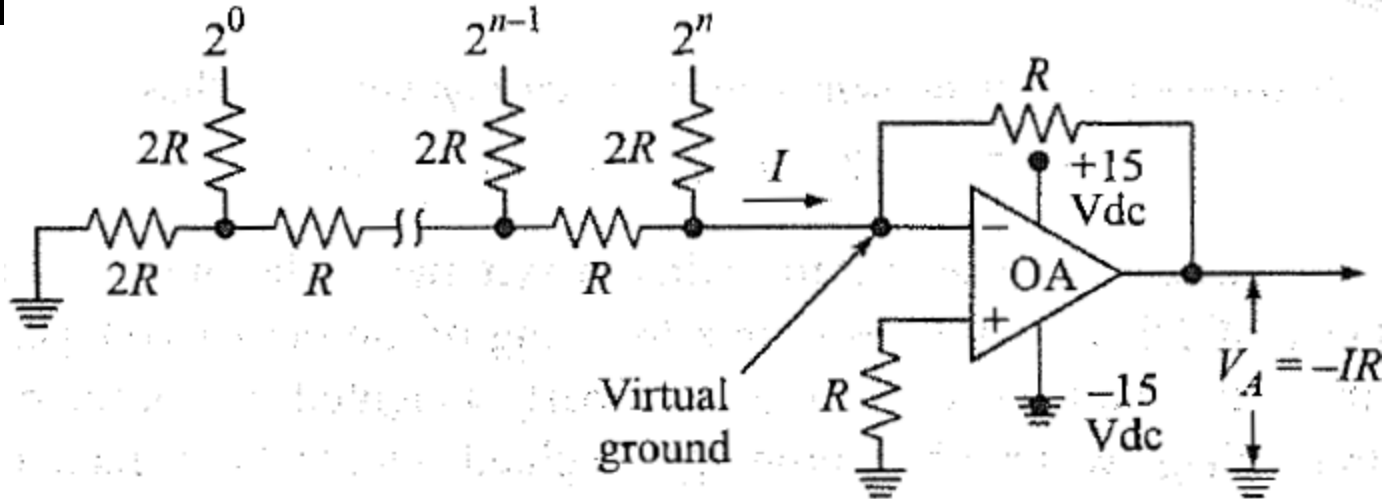
VARIABLE, RESISTOR NETWORKS-16

- “ The operational amplifier (OA) shown in Figure is connected as a unity-gain noninverting amplifier.
- “ It has a very high input impedance, and the output voltage is equal to the input voltage. It is thus a good buffer amplifier for connection to the output of a resistive ladder. It will not load down the ladder and thus will not disturb the ladder output voltage V_A ; V_A will then appear at the output of the OA.



VARIABLE, RESISTOR NETWORKS-17

- “ Connecting an OA with a feedback resistor R as shown in Figure results in an amplifier that acts as an inverting current-to-voltage amplifier.
- “ That is, the output voltage V_A is equal to the negative of the input current I multiplied by R . The input impedance to this amplifier is essentially $0\ \Omega$ thus, when it is connected to an R - $2R$ ladder, the connecting point is virtually at ground potential.



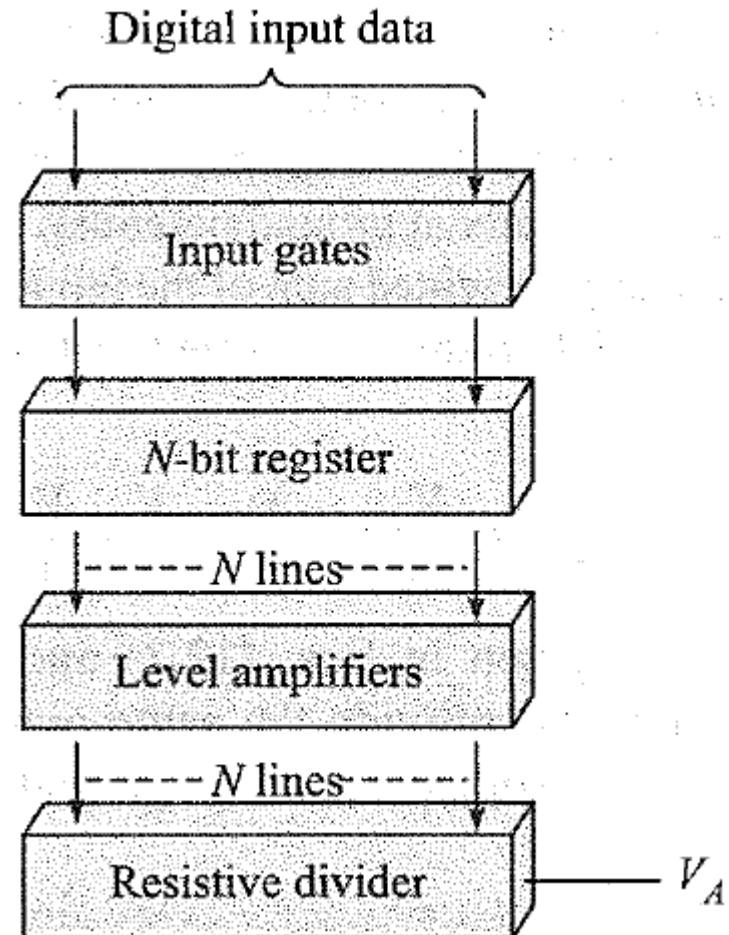
$$V_A = (-R) \left(\frac{V}{2R} + \frac{V}{4R} + \dots \right) = -\frac{V}{2} - \frac{V}{4} - \dots$$

D/A CONVERTERS-1

- “ Either the resistive divider or the ladder can be used as the basis for a digital-to-analog (D/A) converter.
- “ It is in the resistive network that the actual translation from a digital signal to an analog voltage takes place. There is, however, the need for additional circuitry to complete the design of the *D/A* converter.

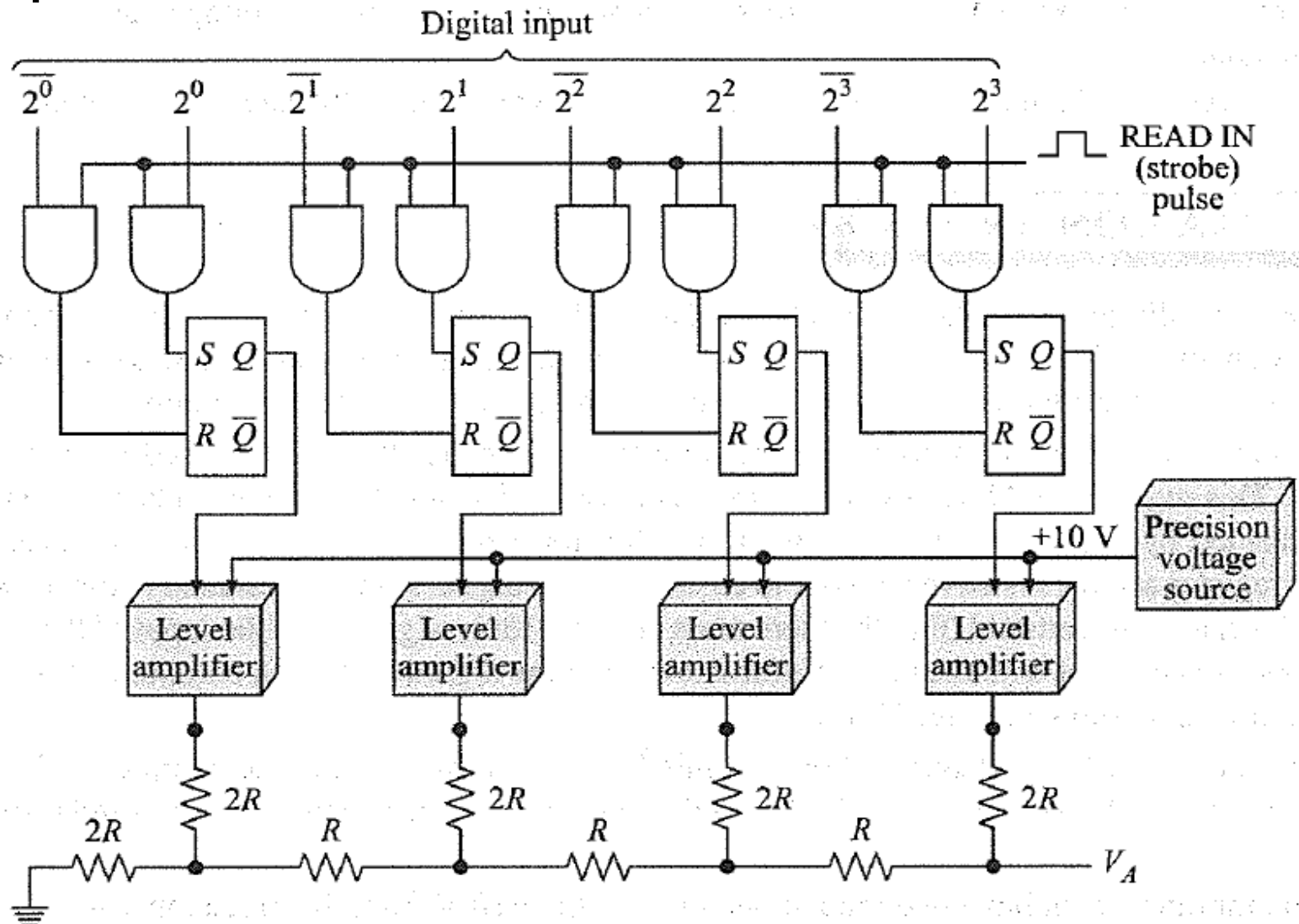
D/A CONVERTERS-2

- “ An integral part of the *D/A* converter are
- “ A register that used to store the digital information. This register could be any one of the many types (Using RS flip-flop or other type of flip-flop).
- “ The level amplifiers between the register and the resistive network to ensure that the digital signals presented to the network are all of the same level and are constant.
- “ Finally, the gating on the input of the register such that the flip-flops can be set with the proper information from the digital system.
- “ A complete *D/A* converter in block-diagram form is shown in Figure.



D/A CONVERTERS-3

“ Complete schematic for a 4-bit *D/A* converter



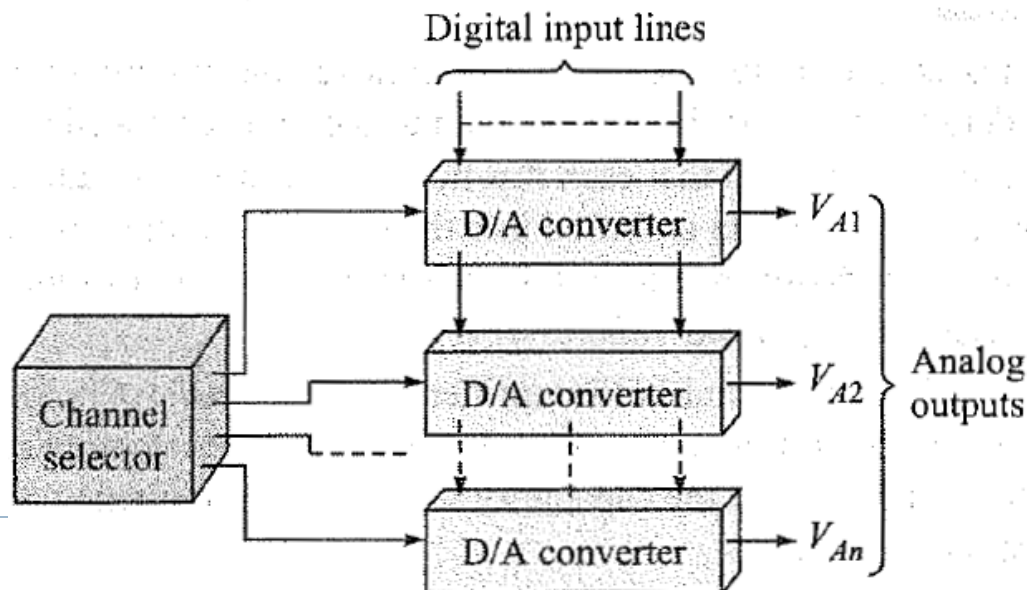
D/A CONVERTERS-4

- “ The level amplifiers each have two inputs: one input is the + 10 V from the precision voltage source, and the other is from a flip-flop.
- “ The amplifiers work as when the input from a flip-flop is high, the output of the amplifier is at +10V. When the input from the flip-flop is low, the output is 0V.
- “ The flip-flop on the right represents the MSB, and the flip-flop on the left represents the LSB.
- “ Each flip-flop is a simple *RS* latch and requires a positive level at the *R* or *S* input to reset or set it. With this particular gating scheme, the flip flops need not be reset (or set) each time new information is entered.
- “ When the READ IN line goes high, only one of the two gate outputs connected to each flip-flop is high, and the flip-flop is set or reset accordingly.
- “ Thus data are entered into the register each time the READ IN (strobe) pulse occurs.

D/A CONVERTERS-5

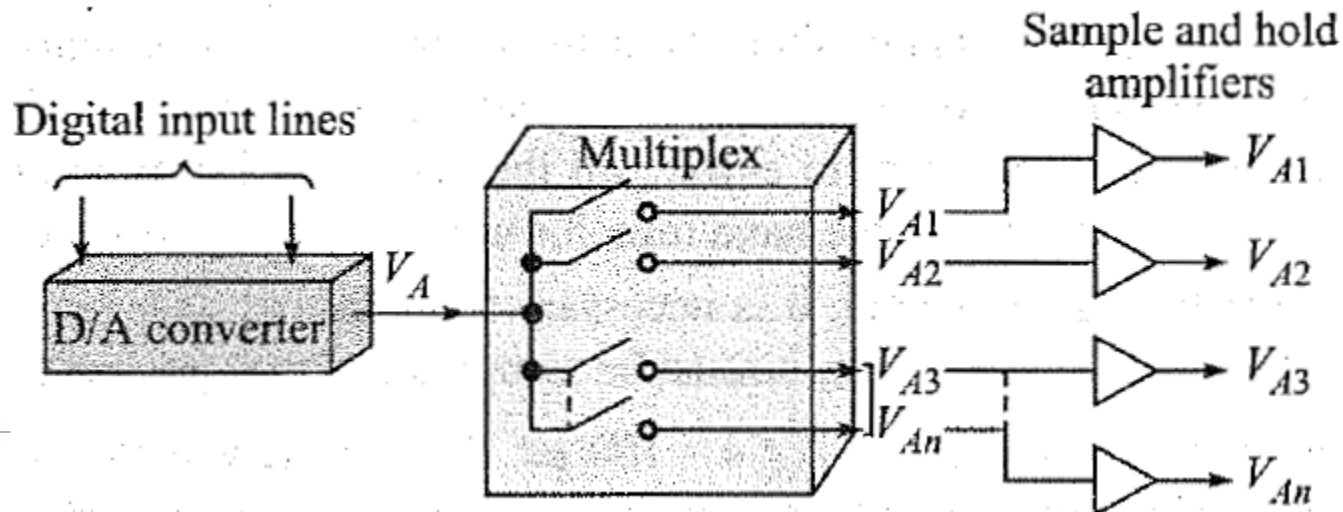
“ Multiple Signals

- “ Quite often it is necessary to decode more than one signal-for example, the X and Y coordinates for a plotting board.
- “ In this event, there are two ways in which to decode the signals.
- “ The first method is simply to use one D/A converter for each signal.
- “ The diagram shown in diagram, has the advantage that each signal to be decoded is held in its register and the analog output voltage is then held fixed.
- “ The digital input lines are connected in parallel to each converter.
- “ The proper converter is then selected for decoding by the select lines.



D/A CONVERTERS-6

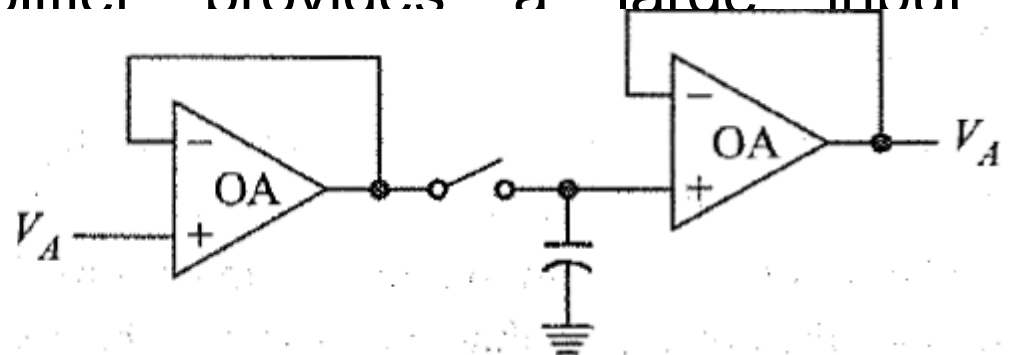
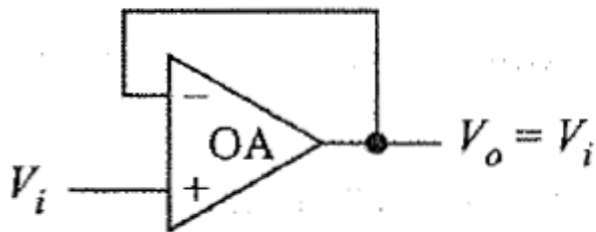
- “ The second method involves the use of only one *D/A* converter and switching its output. This is called *multiplexing*, and shown in figure.
- “ The disadvantage here is that the analog output signal must be held between sampling periods, and the outputs must therefore be equipped with sample-and-hold amplifiers.



D/A CONVERTERS-7

“ Sample and Hold Circuit

- “ An OA connected as in figure is a unity-gain noninverting voltage amplifier-that is, $V_o = V_i$. Two such OAs are used with a capacitor in figure to form a sample-and-hold amplifier.
- “ When the switch is closed, the capacitor charges to the *D/A* converter output voltage.
- “ When the switch is opened, the capacitor holds the voltage level until the next sampling time.
- “ The operational amplifier provides a large input



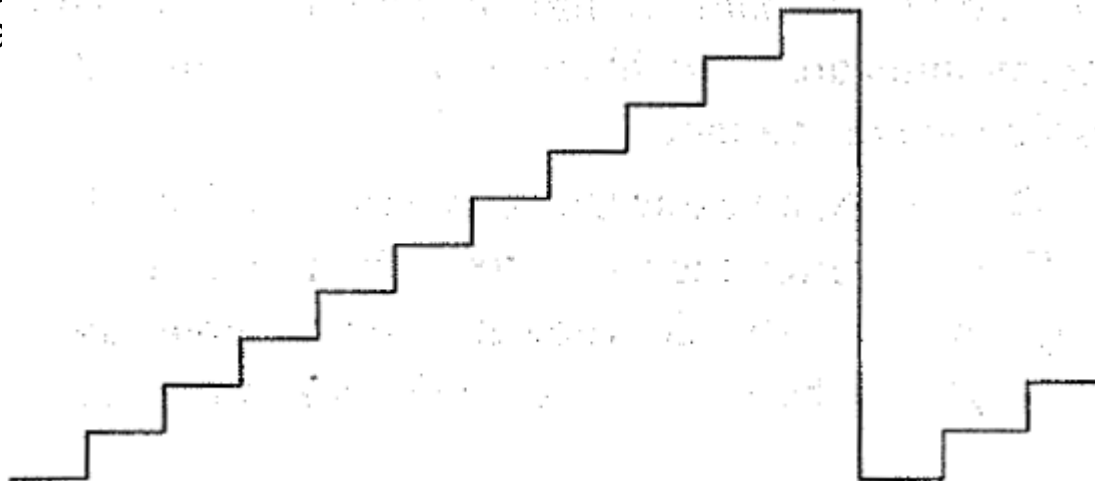
D/A CONVERTERS-8

- “ To reduce the sampling rate to the minimum necessary to extract all the necessary information from the signal.
- “ The solution to this problem involves more than we have time for here, but the results are easy enough to apply.
- “ If the signal is sinusoidal, it is necessary to sample at only *twice* the signal frequency. For instance if the signal is a 5kHz sine wave, it must be sampled at a rate greater than or equal to 10kHz.

D/A CONVERTERS-9

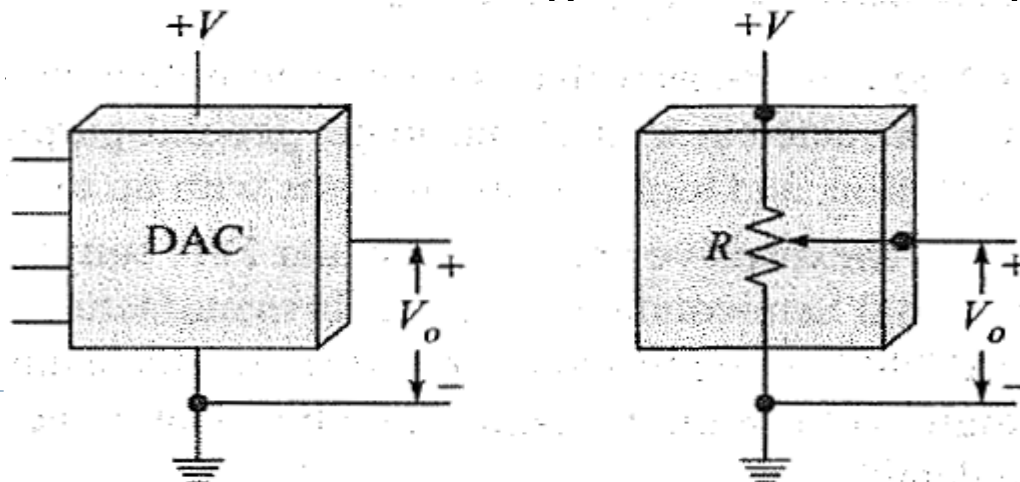
“ D/A Converter Testing

- “ Steady-state accuracy test and the Monotonicity test.
- “ The steady-state accuracy test involves setting a known digital number in the input register, measuring the analog output with an accurate meter, and comparing with the theoretical value.
- “ Monotonicity means checking that the output voltage increases regularly as the input digital signal increases. This can be accomplished by using a counter as the digital input signal and observing the analog output on an oscilloscope. For proper monotonicity, the output waveform should be a perfect staircase waveform, as shown in figure.
- “ The steps on the staircase waveform must be equally spaced and of the exact same amplitude.
- “ Missing steps, steps of different amplitude, or steps in a downward fashion indicate ma



D/A CONVERTERS-10

- “ A *D/A* converter can be regarded as a logic block having numerous digital inputs and a single analog output as shown in figure.
- “ It is interesting to compare this logic block with the potentiometer shown in Figure.
- “ The analog output voltage of the *D/A* converter is controlled by the digital input signals while the analog output voltage of the potentiometer is controlled by mechanical rotation of the potentiometer shaft.
- “ Considered in this fashion, it is easy to see how a *D/A* converter could be used to generate a voltage waveform (sawtooth, controlled

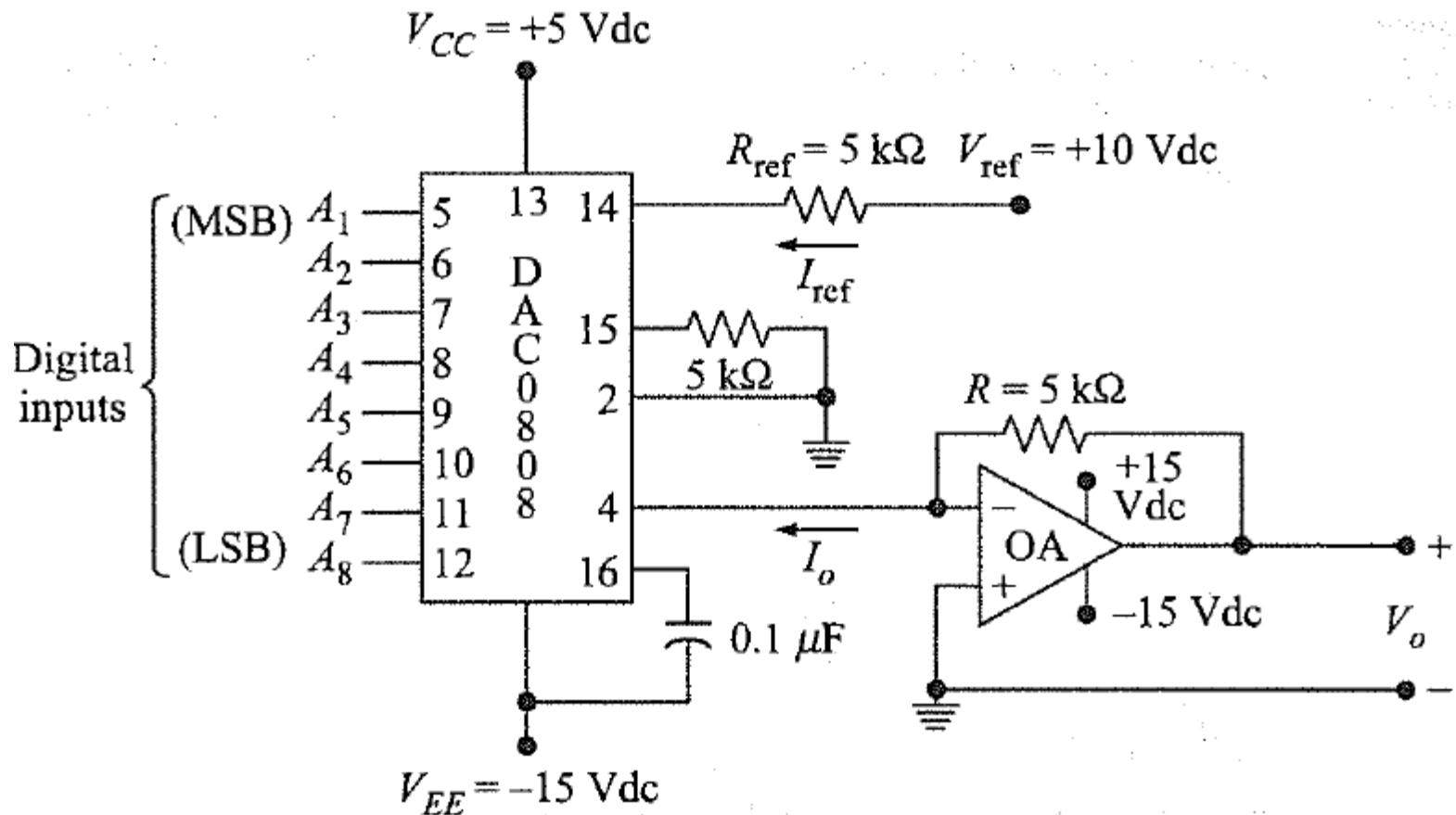


D/A CONVERTERS-11

“ Available D/A Converters

- “ Very popular *D/A* converter is the DAC0808, an 8-bit *D/A* converter.
- “ In Figure a DAC0808 is connected to provide a full-scale output voltage of $V_O = +10\text{ V dc}$ when all 8 digital inputs are 1s (high). If the 8 digital inputs are all 0s (low), the output voltage will be $V_O = 0\text{ Vdc}$.
- “ First of all, two dc power-supply voltages are required for the DAC0808: $V_{CC} = +5\text{ V dc}$ and $V_{EE} = -15\text{ Vdc}$.
- “ The $0.1\text{-}\mu\text{F}$ capacitor is to prevent unwanted circuit oscillations, and to isolate any variations in V_{EE} .
- “ Pin2 is ground (GND), and pin 15 is also referenced to ground through a resistor.

D/A CONVERTERS-12



D/A CONVERTERS-13

- “ The output of the D/A converter on pin 4 has a very limited voltage range (+0.5 to -0.6 V). Rather, it is designed to provide an output current I_o .
- “ The minimum current (all digital inputs low) is 0.0 mA, and the maximum current (all digital inputs high), is I_{ref} . This reference current is established with the resistor at pin 14 and the reference voltage as:

$$I_{ref} = V_{ref}/R_{ref}$$

- “ The D/ A converter output current I_o is given as

$$I_o = I_{ref} \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \dots + \frac{A8}{256} \right)$$

- “ where $A1, A2, A3, \dots, A8$ are the digital input levels (1 or 0).

D/A CONVERTERS-14

- “ The OA is connected as a current-to-voltage converter, and the output voltage is given as:

$$V_o = I_o \times R$$

- “ If we set $V_o = V_{\text{ref}} / R_{\text{ref}} \times \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \dots + \frac{A8}{256} \right) \times R$ equal to R_{ref} , then

$$V_o = V_{\text{ref}} \left(\frac{A1}{2} + \frac{A2}{4} + \frac{A3}{8} + \dots + \frac{A8}{256} \right)$$

D/A CONVERTERS-15

- “ Suppose all digital inputs are 0s (all low). Then

$$\begin{aligned} V_o &= V_{\text{ref}} \times \left(\frac{0}{2} + \frac{0}{4} + \frac{0}{8} + \dots + \frac{0}{256} \right) \\ &= V_{\text{ref}} \times 0 = 0.0 \text{ Vdc} \end{aligned}$$

- “ Now all digital inputs are 1s (all high). Then

$$\begin{aligned} V_o &= V_{\text{ref}} \times \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{256} \right) \\ &= (V_{\text{ref}}) \times \left(\frac{255}{256} \right) = 0.996 \times V_{\text{ref}} \end{aligned}$$

- “ Vref is +10 Vdc, the output voltage is seen to have a range between 0.0 and +9.96 Vdc. It doesn't quite reach +10 Vdc,