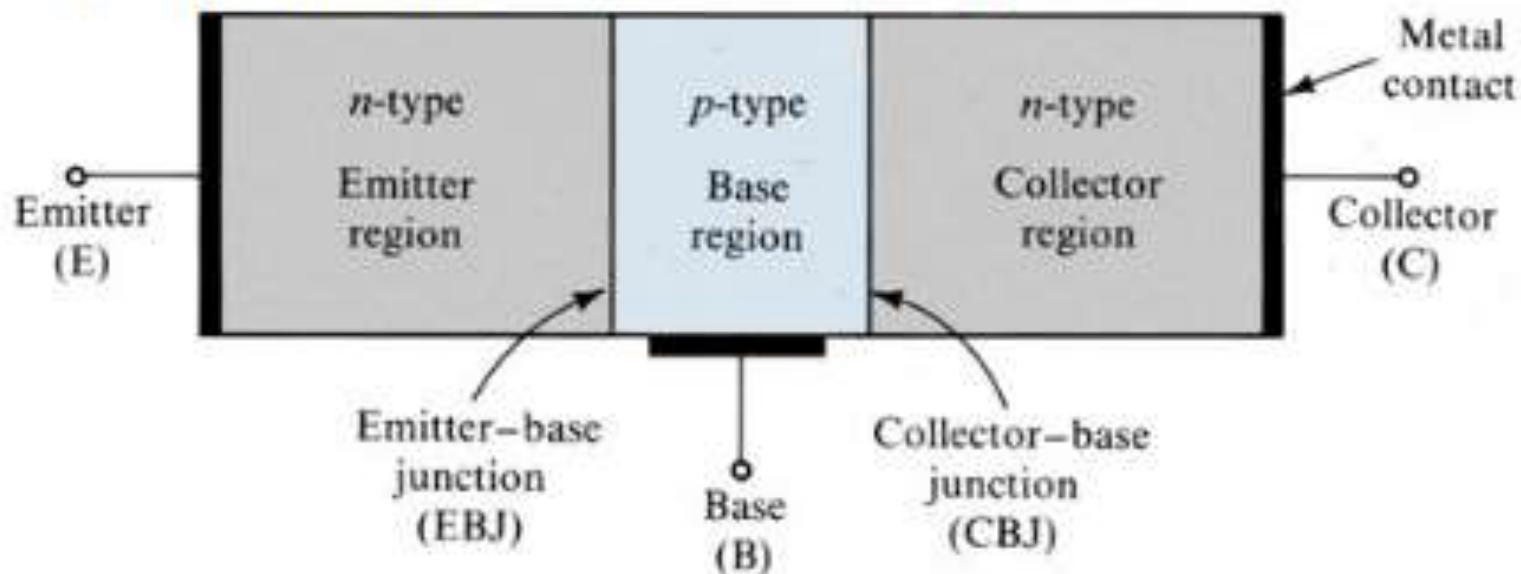


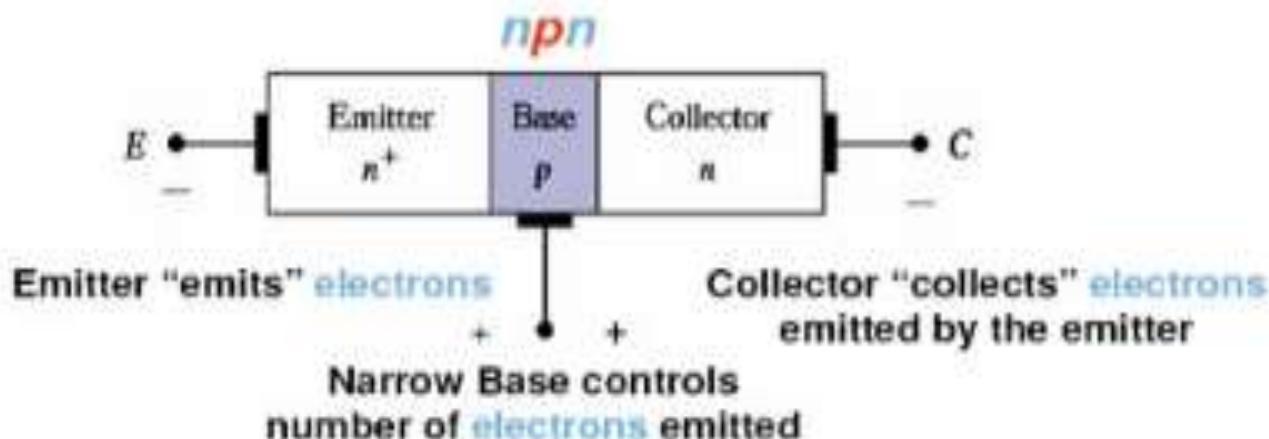
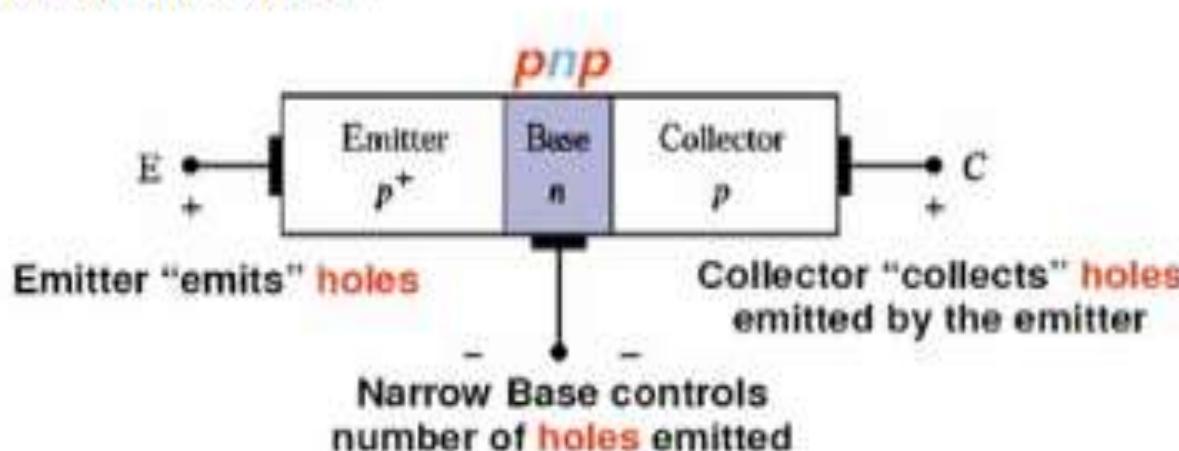
# BJT- Bipolar Junction Transistor

# The Bipolar Junction Transistor (BJT)

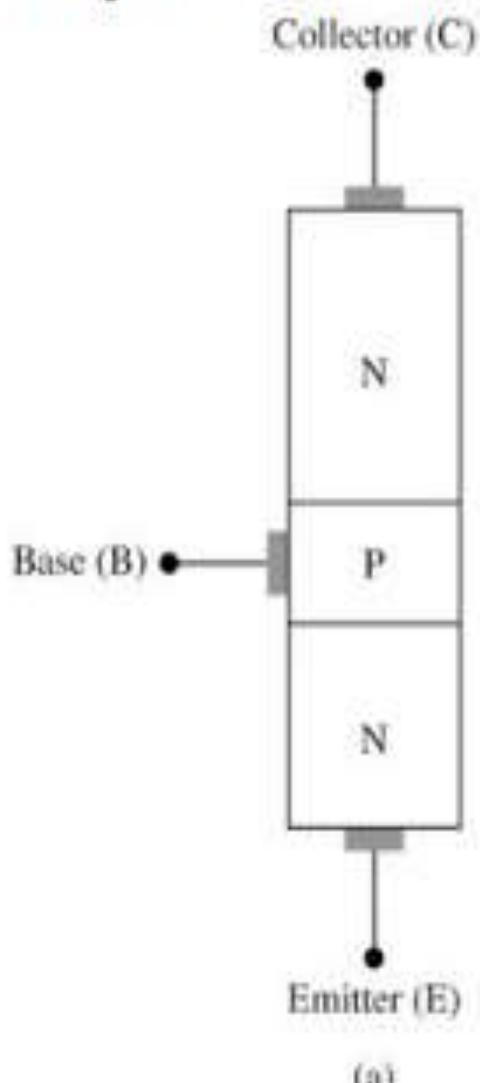


- **Bipolar** : both electrons and holes are involved in current flow.
- **Junction** : has two *p-n* junctions.
- **Transistor** : Transfer + Resistor.
- It can be either *n-p-n* type or *p-n-p* type.
- Has three regions with three terminals labeled as
  - i. Emitter (E)
  - ii. Base (B) and
  - iii. Collector (C)

- **Base** is made much narrow.
- **Emitter** is heavily doped ( $p^+$ ,  $n^+$ ).
- **Base** is lightly doped ( $p$ ,  $n$ ).
- **Collector** is lightly doped ( $p$ ,  $n$ ).

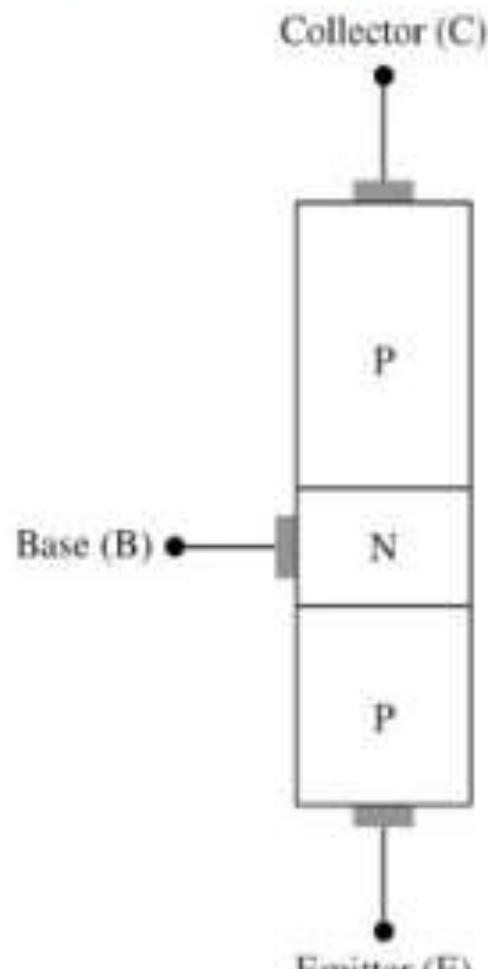


# Layout and Circuit Symbol: n-p-n Transistor

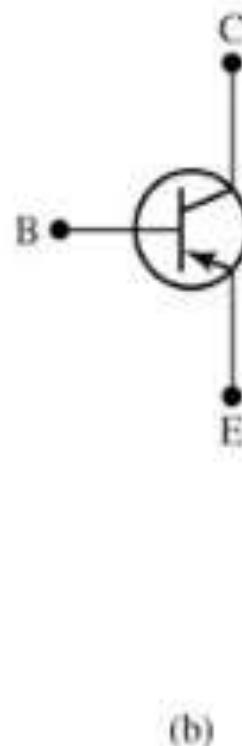


- The arrow indicates the direction of current flow.
- The current flows from collector to emitter in an n-p-n transistor.
- The arrow is drawn on the emitter.
- The arrow always points towards the n-type. So the emitter is n-type and the transistor is n-p-n type.

# Layout and Circuit Symbol: p-n-p Transistor



(a)



(b)

- The arrow indicates the direction of current flow.
- The current flows from emitter to collector in an p-n-p transistor.
- The arrow points towards the n-type.
- So the base is n-type and transistor is p-n-p type.

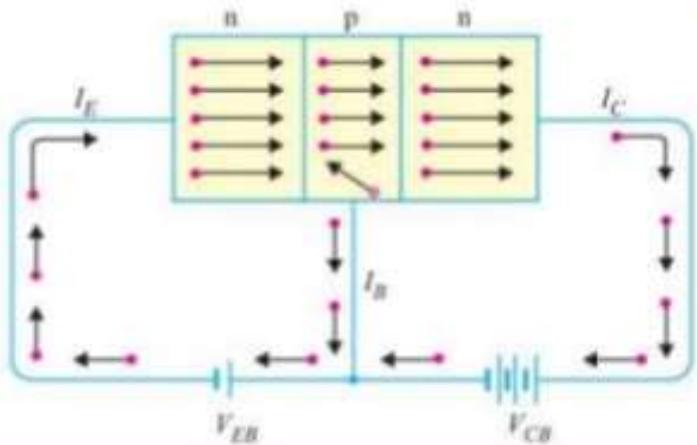
 Clip slide

## Some important factors to be remembered-

- The junction between emitter and base is called emitter-base junction(emitter diode) and junction between base and collector is called collector-base junction(collector diode).
- The emitter diode is always forward biased and collector diode is reverse biased.
- The resistance of emitter diode is very small(forward) and resistance of collector diode is high(reverse).

# Transistor Operation

## 1) Working of npn transistor:



✓ Forward bias is applied to emitter-base junction and reverse bias is applied to collector-base junction.

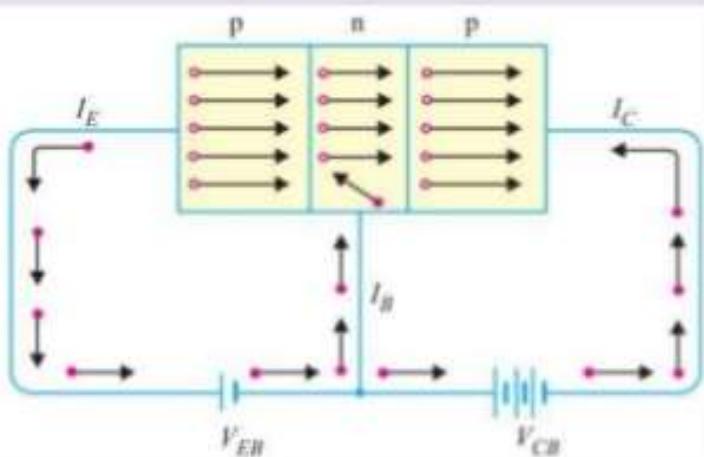
✓ The forward bias in the emitter-base junction causes electrons to move toward base. This constitute emitter current,  $I_E$

### 1) Working of npn transistor:

- ✓ As these electrons flow toward p-type base, they try to recombine with holes. As base is lightly doped only few electrons recombine with holes within the base.
- ✓ These recombined electrons constitute small base current.
- ✓ The remainder electrons cross base and constitute collector current.

$$I_E = I_B + I_C$$

## 2) Working of pnp transistor:



✓ Forward bias is applied to emitter-base junction and reverse bias is applied to collector-base junction.

✓ The forward bias in the emitter-base junction causes holes to move toward base. This constitute emitter current,  $I_E$

## Transistor Operation

### 2) Working of pnp transistor:

- ✓ As these holes flow toward n-type base, they try to recombine with electrons. As base is lightly doped only few holes recombine with electrons within the base.
- ✓ These recombined holes constitute small base current.
- ✓ The remainder holes cross base and constitute collector current.

## Transistor Operating Modes

- Active Mode
  - Base-Emitter junction is forward and Base-Collector junction is reverse biased.
- Saturation Mode
  - Base-Emitter junction is forward and Base-Collector junction is forward biased.
- Cut-off Mode
  - Both junctions are reverse biased.

# Transistor Configuration

The transistor has three terminals – emitter (E), base (B) and collector (C).

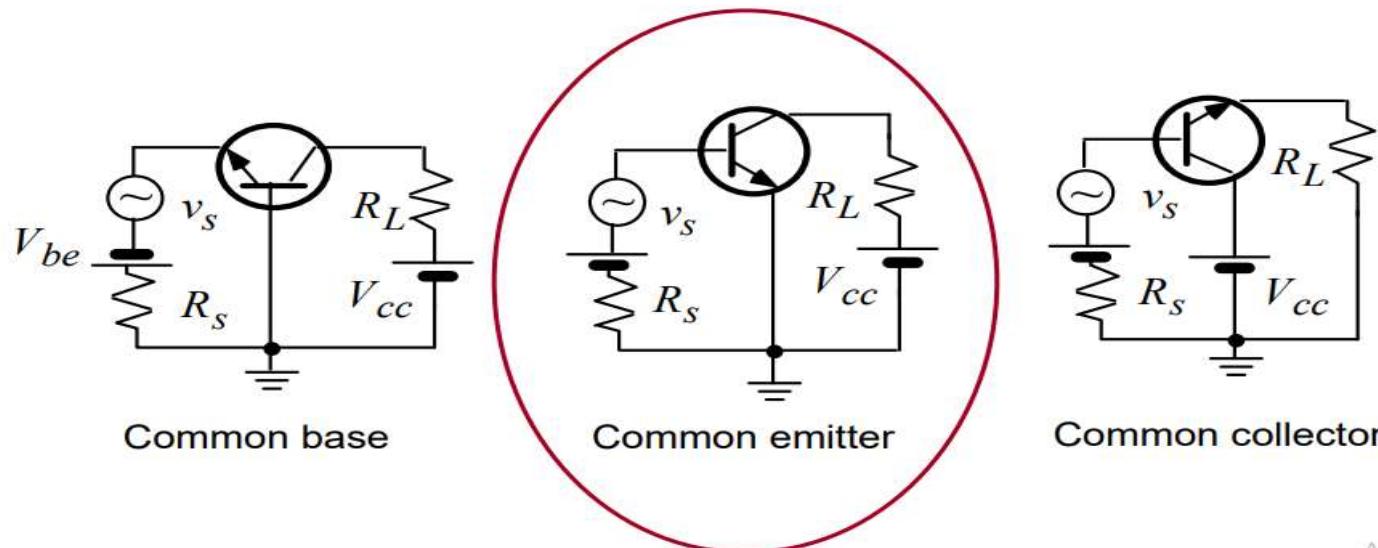
But in the circuit connections we need four terminals, two terminals for input and another two terminals for output. To overcome these problems we use one terminal as common for both input and output actions.

The behaviour of these three different configurations of transistors with respect to gain

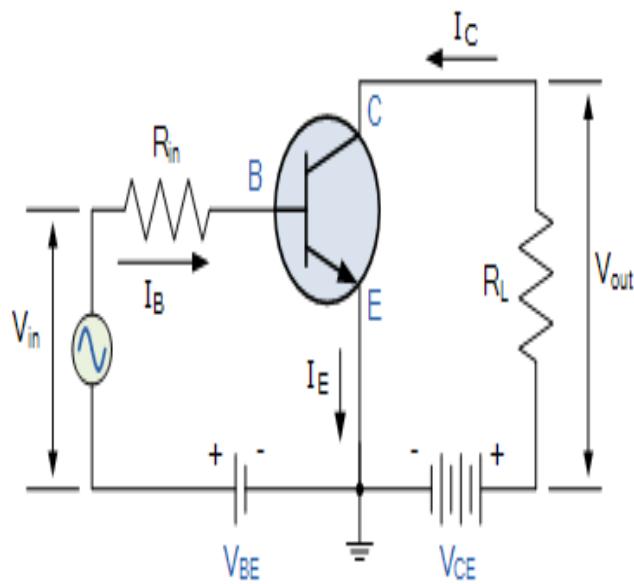
**Common Emitter (CE) Configuration:** current gain and voltage gain

**Common Collector (CC) Configuration:** current gain but no voltage gain

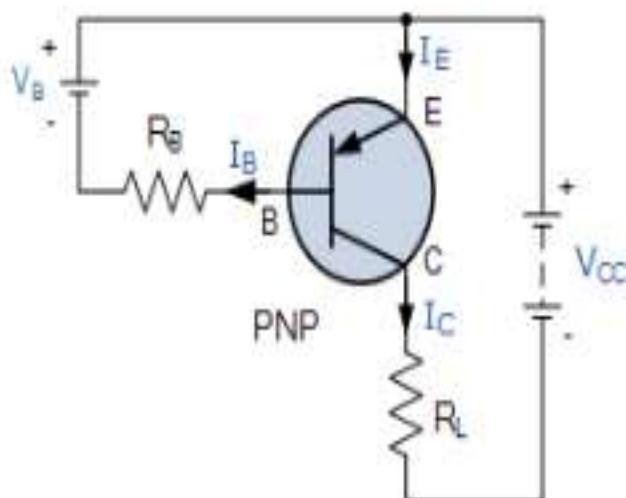
**Common Base (CB) Configuration:** no current gain but voltage gain



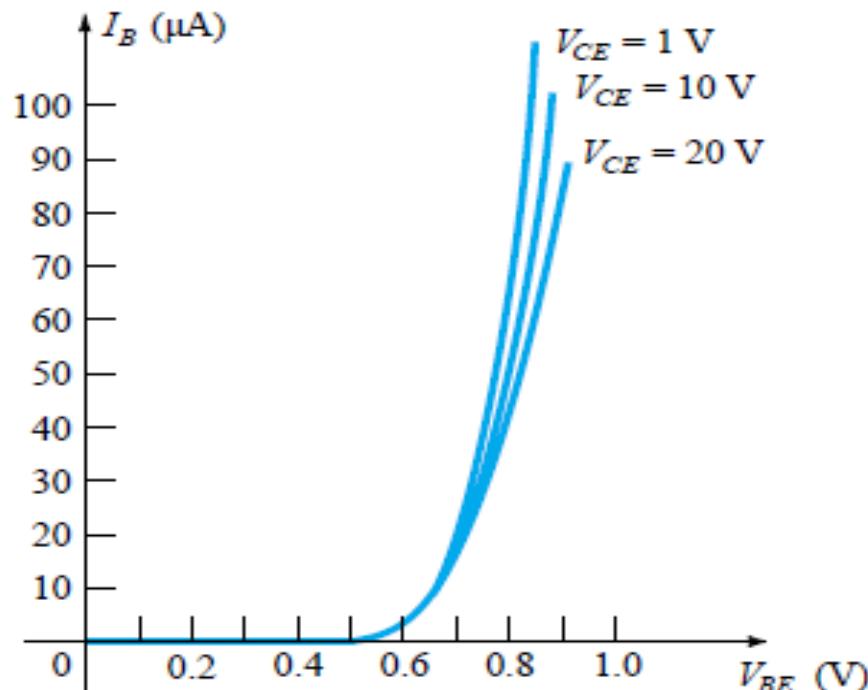
# Common Emitter for NPN and PNP Transistor



- Most common configuration of transistor is as shown
- *emitter* terminal is common to input and output circuits this is a **common-emitter** configuration
- we will look at the characteristics of the device in this configuration
- The current relations are still applicable, *i.e.*,
- $I_E = I_C + I_B$  and  $I_C = \alpha I_E$



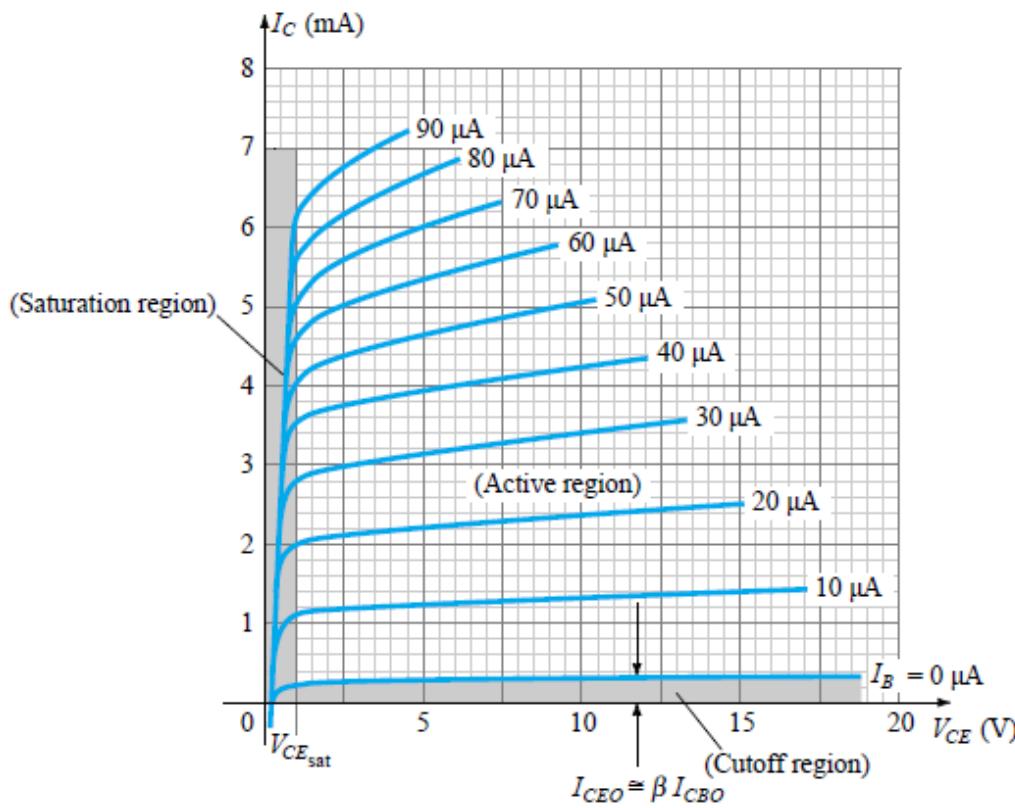
# Common Emitter Input Characteristics



- the input takes the form of a forward-biased *pn* junction
- the input characteristics are therefore similar to those of a semiconductor diode

An input current ( $I_B$ ) is a function of an input voltage ( $V_{BE}$ ) for various of output voltage ( $V_{CE}$ ).

# COMMON Emitter OUTPUT CHARACTERISTICS



## Output characteristics

- The magnitude of  $I_B$  is in  $\mu$ A and not as horizontal as  $I_E$  in common-base circuit.
- The output set relates an output current ( $I_C$ ) to an output voltage ( $V_{CE}$ ) for various of level of input current ( $I_B$ ).

- There are three portions as shown:

### Active region

- The active region, located at upper-right quadrant, has the greatest linearity.
- The curve for  $I_B$  are nearly straight and equally spaced.
- In active region, the *B-E junction* is forward-biased, whereas the *C-B junction* is reverse-biased.
- The active region can be employed for voltage, current or power amplification.

Clip slide

## Cutoff region

Clip sick

- The region below  $I_B = 0\mu A$  is defined as cutoff region.
- For linear amplification, cutoff region should be avoided.

## Saturation region:

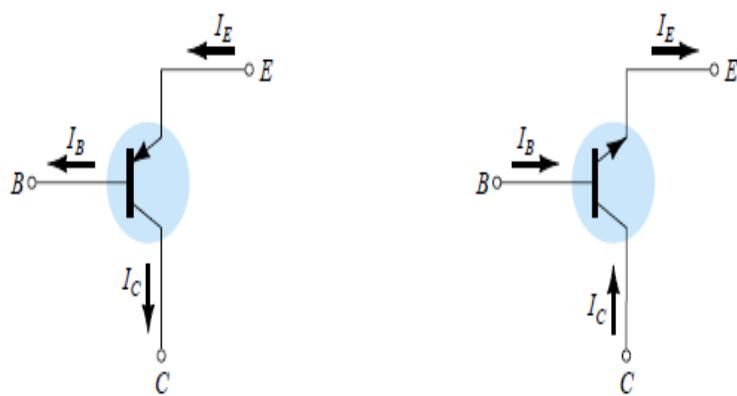
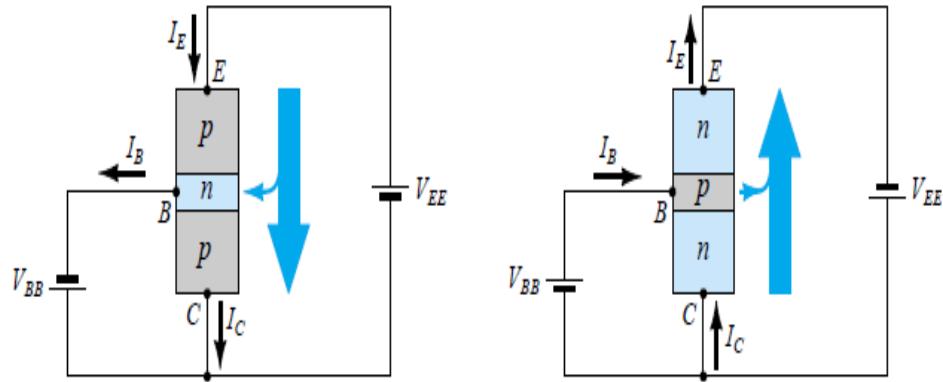
- The small portion near the ordinate, is the saturation region, which should be avoided for linear application.
- In the dc mode, the levels of  $I_C$  and  $I_B$  at the operation point are related by: Normally,  $\beta$  ranges from 50 to 400.

$$\beta_{dc} = I_C / I_B$$

For ac situations,  $\beta$  is defined as

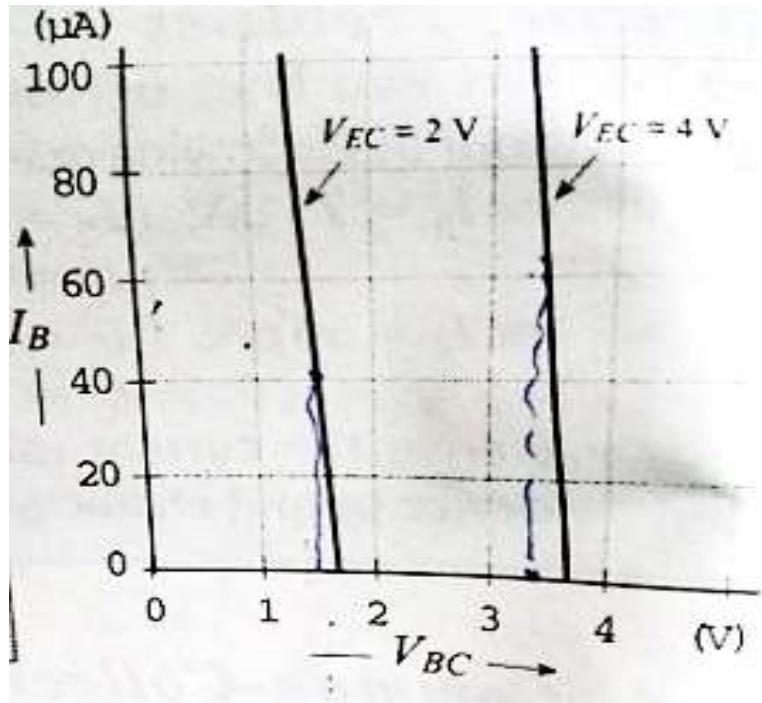
$$\beta_{ac} = \left. \frac{\Delta I_C}{\Delta I_B} \right|_{V_{CE}=\text{constant}}$$

# COMMON COLLECTOR FOR PNP AND NPN TRANSISTORS



- It is used primarily for impedance-matching purpose since it has a high input impedance and low output impedance.
- The load resistor can be connected from emitter to ground.
- The collector is tied to ground and the circuit resembles common-emitter circuit.
- The output set relates an output current ( $I_E$ ) to an output voltage ( $V_{CE}$ ) for various levels of input current ( $I_B$ ).

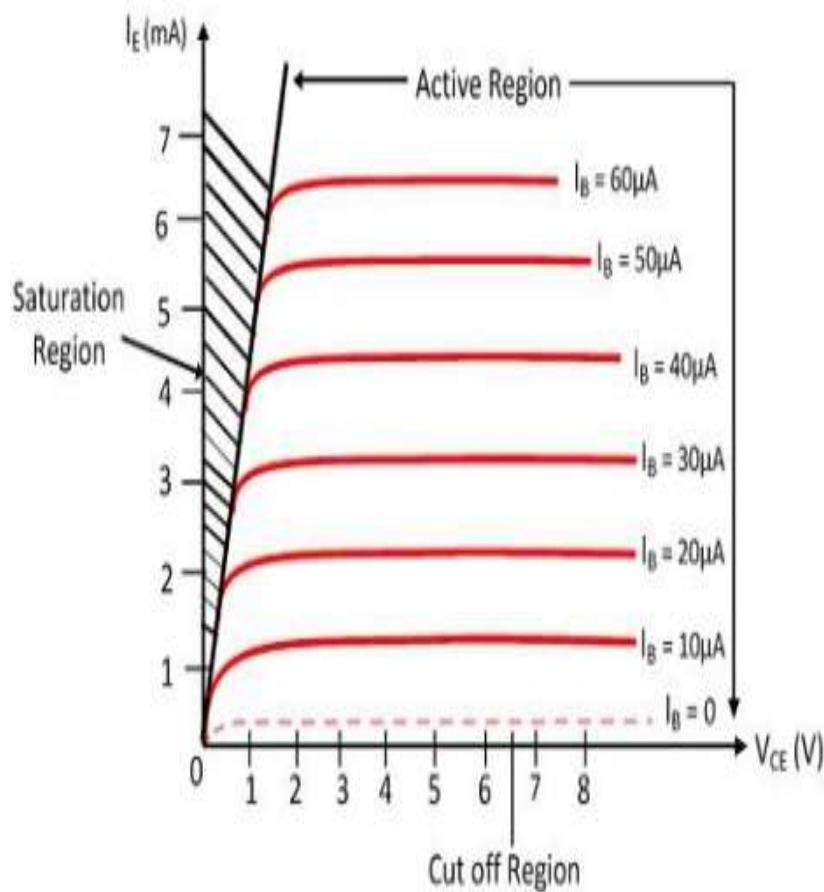
# COMMON COLLECTOR INPUT CHARACTERISTICS



- It is a curve which shows the relationship between the base current,  $I_B$  and the collector base voltage  $V_{CB}$  at constant  $V_{CE}$ . This method of determining the characteristic is as follows.
  - First, a suitable voltage is applied between the emitter and the collector.
  - Next the input voltage  $V_{CB}$  is increased in a number of steps and corresponding values of  $I_E$  are noted.
  - The base current is taken on the y-axis, and the input voltage is taken on the x-axis. Fig. shows the family of the input characteristic at different collector- emitter voltages.

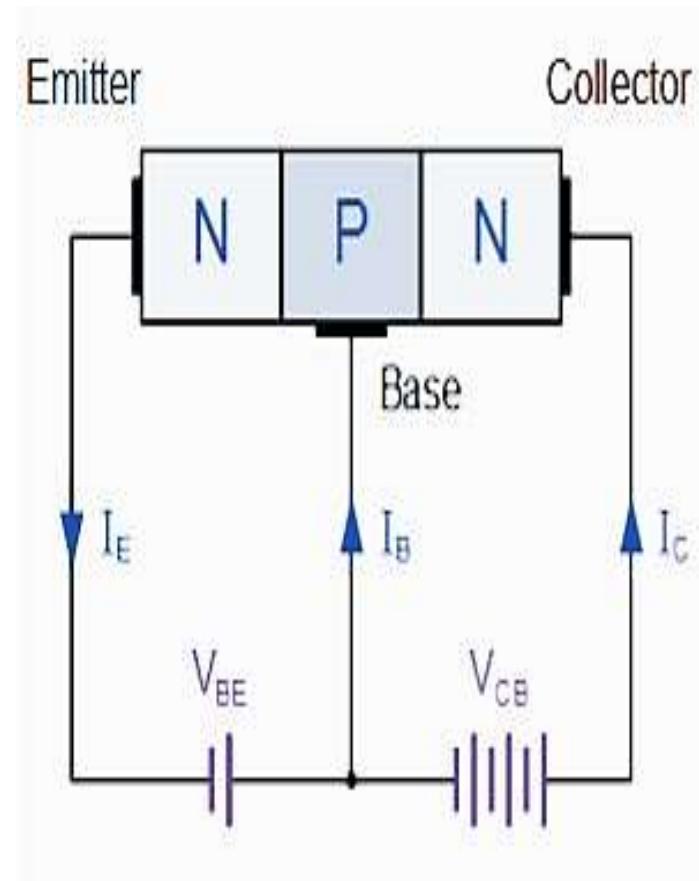
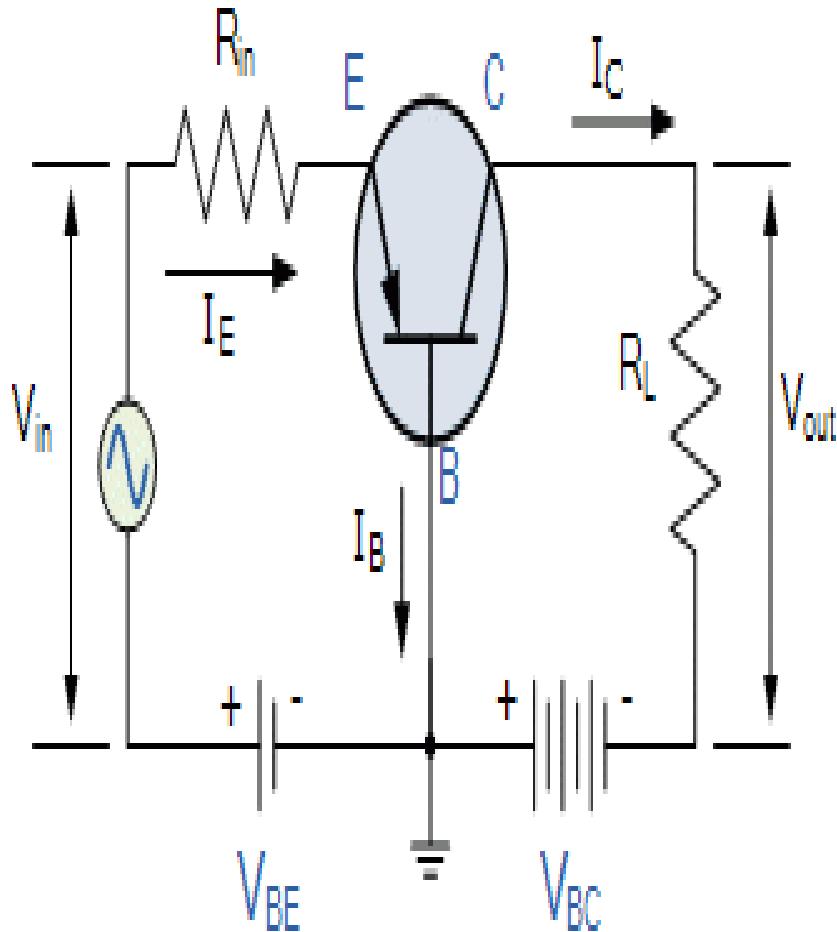
- The following points may be noted from the family of characteristic curves.
- Its characteristic is quite different from those of common base and common emitter circuits.
- When  $V_{CB}$  increases,  $I_B$  is decreased.

# COMMON COLLECTOR OUTPUT CHARACTERISTICS



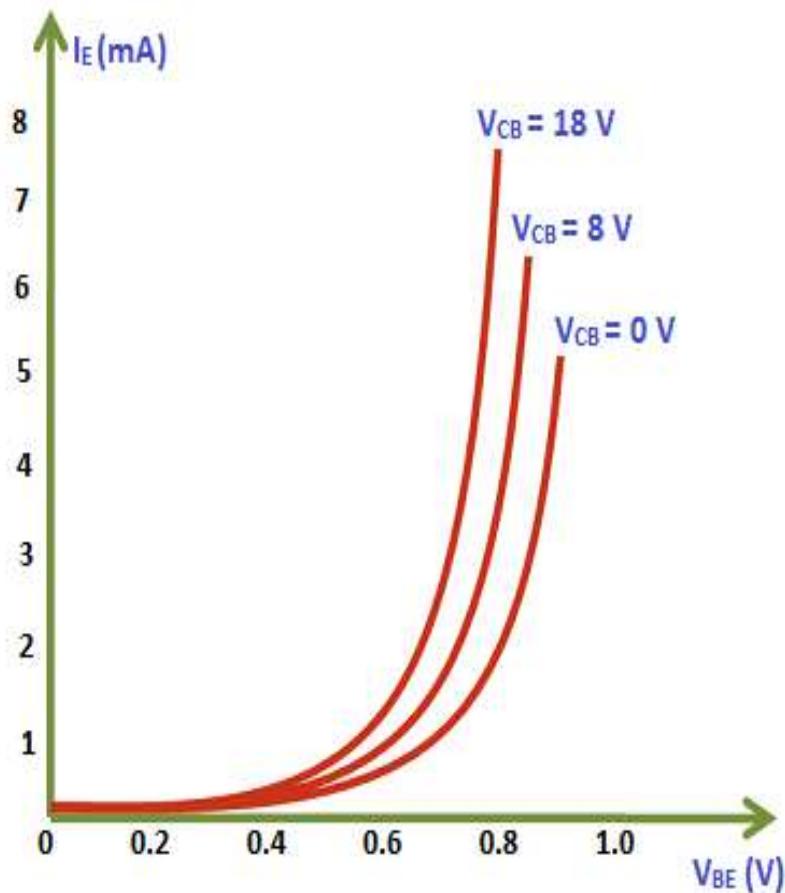
- This is almost the same as the output characteristics of common-emitter circuit, which are the relations between  $I_C$  and  $V_{CE}$  for various of level of input current  $I_B$ .
- Since that:  $I_E \cong I_C$ .

# COMMON BASE FOR PNP AND NPN TRANSISTORS



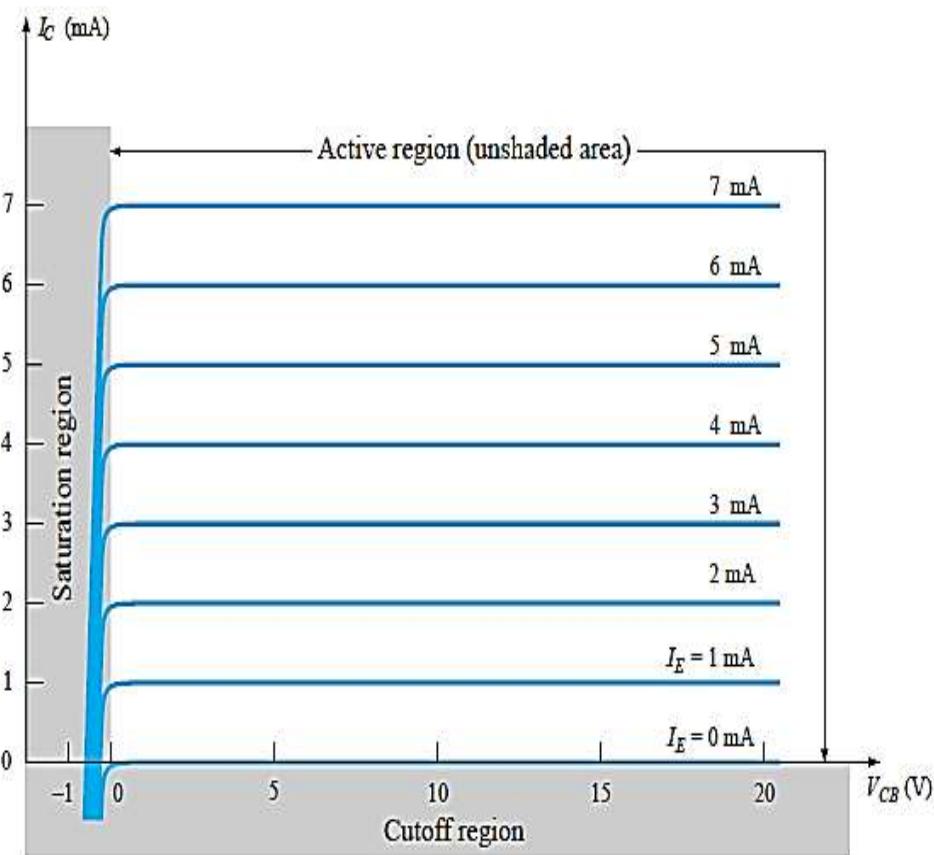
- The common-base configuration with *pnp* and *npn* transistors are shown in the figures in the previous slide..
- The term *common-base* is derived from the fact that the base is common to both the input and output sides of the configuration.
- The arrow in the symbol defines the direction of emitter current through the device.
- The applied biasing are such as to establish current in the direction indicated for each branch.
- That is, direction of  $I_E$  is the same as the polarity of  $V_{EE}$  and  $I_C$  to  $V_{CC}$ .
- Also, the equation  $I_E = I_C + I_B$  still holds.

# COMMON BASE INPUT CHARACTERISTICS



- The *driving point or input parameters* are shown in the figure.
- An input current ( $I_E$ ) is a function of an input voltage ( $V_{BE}$ ) for various of output voltage ( $V_{CB}$ ).
- This closely resembles the characteristics of a diode.

# COMMON BASE OUTPUT CHARACTERISTICS



- The output set relates an output current ( $I_C$ ) to an output voltage ( $V_{CB}$ ) for various levels of input current ( $I_E$ ).

There are three regions of interest:

## Active region

- In the active region, the *b-e junction* is forward-biased, whereas the *c-b junction* is reverse-biased.
- The active region is the region normally employed for linear amplifier. Also, in this region,

$$I_C \approx I_E$$

## Cutoff region

- The cutoff region is defined as that region where the collector current is 0 A.
- In the cutoff region, the *B-E* and *C-B* junctions of a transistor are both reverse-biased.

## Saturation region:

- It is defined as that region of the characteristics to the left of  $V_{CB} = 0$  V.
- In saturation region, the *B-E* and *C-B* junctions of a transistor are both forward biased.

**1. In a common base connection, current amplification factor is 0.9. If the emitter current is 1mA, determine the value of base current.**

**Solution :**

Here,  $\alpha = 0.9$ ,  $I_E = 1 \text{ mA}$

Now

$$\alpha = \frac{I_C}{I_E}$$

or

$$I_C = \alpha I_E = 0.9 \times 1 = 0.9 \text{ mA}$$

Also

$$I_E = I_B + I_C$$

$\therefore$  Base current,  $I_B = I_E - I_C = 1 - 0.9 = 0.1 \text{ mA}$

**2. In a common base connection,  $I_C = 0.95 \text{ mA}$  and  $I_B = 0.05 \text{ mA}$ . Find the value of  $\alpha$ .**

**Solution :**

We know  $I_E = I_B + I_C = 0.05 + 0.95 = 1 \text{ mA}$

$\therefore$  Current amplification factor,  $\alpha = \frac{I_C}{I_E} = \frac{0.95}{1} = 0.95$

**3. In a common base connection,  $I_E = 1\text{mA}$ ,  $I_C = 0.95\text{mA}$ . Calculate the value of  $I_B$ .**

**Solu**Using the relation,  $I_E = I_B + I_C$

$$1 = I_B + 0.95$$

$$I_B = 1 - 0.95 = 0.05 \text{ mA}$$

**4. In a common base connection, the emitter current is 1mA.**

**If the emitter circuit is open, the collector current is 50  $\mu\text{A}$ . Find the total collector current. Given that  $\alpha = 0.92$ .**

**Solution :**

Here,  $I_E = 1\text{mA}$ ,  $\alpha = 0.92$ ,  $I_{CBO} = 50\text{ }\mu\text{A}$

$$\begin{aligned}\therefore \text{Total collector current, } I_C &= \alpha I_E + I_{CBO} = 0.92 \times 1 + 50 \times 10^{-3} \\ &= 0.92 + 0.05 = 0.97 \text{ mA}\end{aligned}$$

**5. Calculate  $I_E$  in a transistor for which  $\beta = 50$  and  $I_B = 20 \mu\text{A}$ .**

**Solution :**

Here  $\beta = 50$ ,  $I_B = 20 \mu\text{A} = 0.02 \text{ mA}$

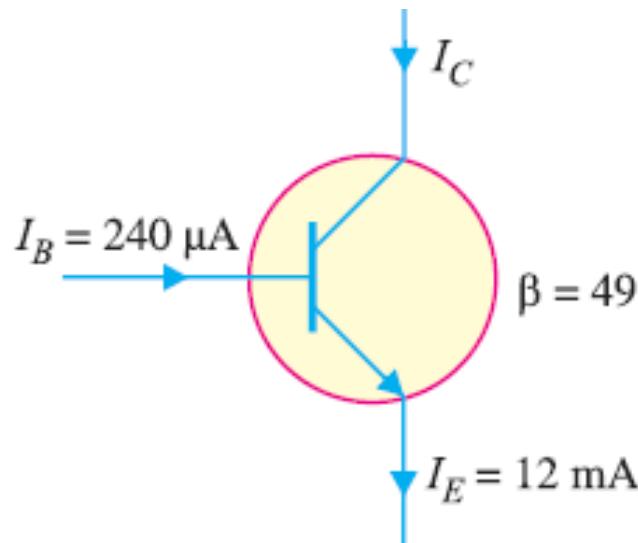
Now

$$\beta = \frac{I_C}{I_B}$$

$$I_C = \beta I_B = 50 \times 0.02 = 1 \text{ mA}$$

Using the relation,  $I_E = I_R + I_C = 0.02 + 1 = 1.02 \text{ mA}$

**6. Find the  $\alpha$  rating of the transistor shown in Fig. Hence determine the value of  $I_C$  using both  $\alpha$  and  $\beta$  rating of the transistor.**



$$\alpha = \frac{\beta}{1 + \beta} = \frac{49}{1 + 49} = 0.98$$

The value of  $I_C$  can be found by using either  $\alpha$  or  $\beta$  rating as under :

$$I_C = \alpha I_E = 0.98 (12 \text{ mA}) = 11.76 \text{ mA}$$

$$\text{Also } I_C = \beta I_B = 49 (240 \mu\text{A}) = 11.76 \text{ mA}$$

# COMPARISON OF CB,CE,CC

Characteristic	Common base (CB)	Common emitter,(CE)	Common collector,(CC)
Input Dynamic Resistance	Very Low(less than 100 ohm)	Low(less than 1K)	Very High(750K)
Output Dynamic Resistance	Very High	High	Low
Current Gain	Less than 1	High	Very High
Voltage gain	Greater than CC but less than CE	Highest	Lowest(less than 1)
Power gain	Medium	Highest	Medium
Leakage current	Very small	Very large	Very large
Relationship between I/p and o/p	In phase	Out of phase( $180^\circ$ )	In phase
Application	For High freq. applications	For Audio freq. Applications	For impedance Matching Applications

# RELATIONSHIP OF $\alpha, \beta, \gamma$

$$\alpha_{dc} = \frac{I_C}{I_E}$$

$$\gamma = \frac{I_E}{I_B}$$

$$\beta_{dc} = \frac{I_C}{I_B}$$

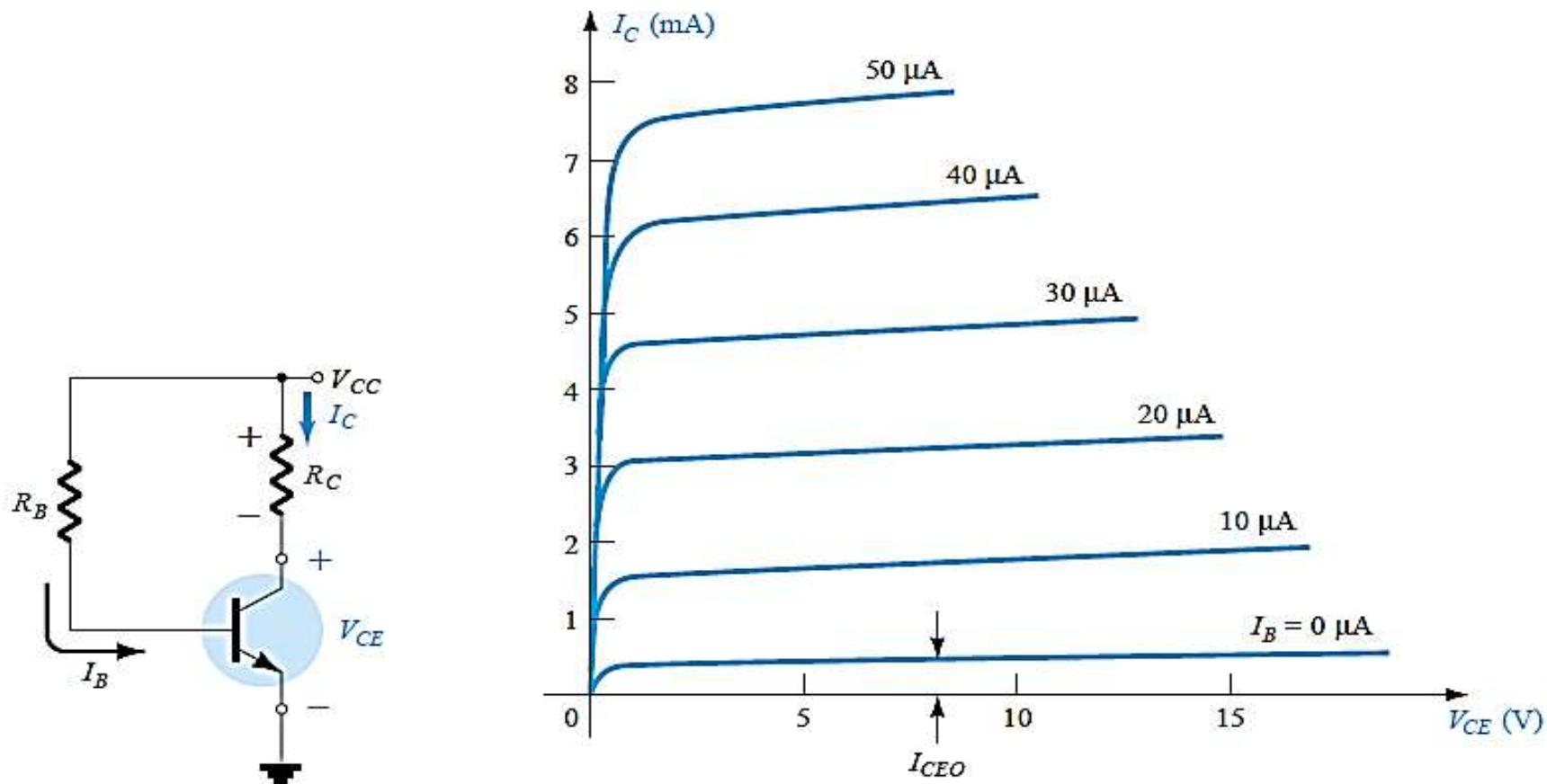
$$\gamma = \beta + 1$$

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

$$\beta = \frac{\alpha}{1 - \alpha}$$

# DC BIASING

Transistor Biasing is the process of setting a transistors DC operating voltage or current conditions to the correct level so that any AC input signal can be amplified correctly by the transistor



# DC BIASING

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

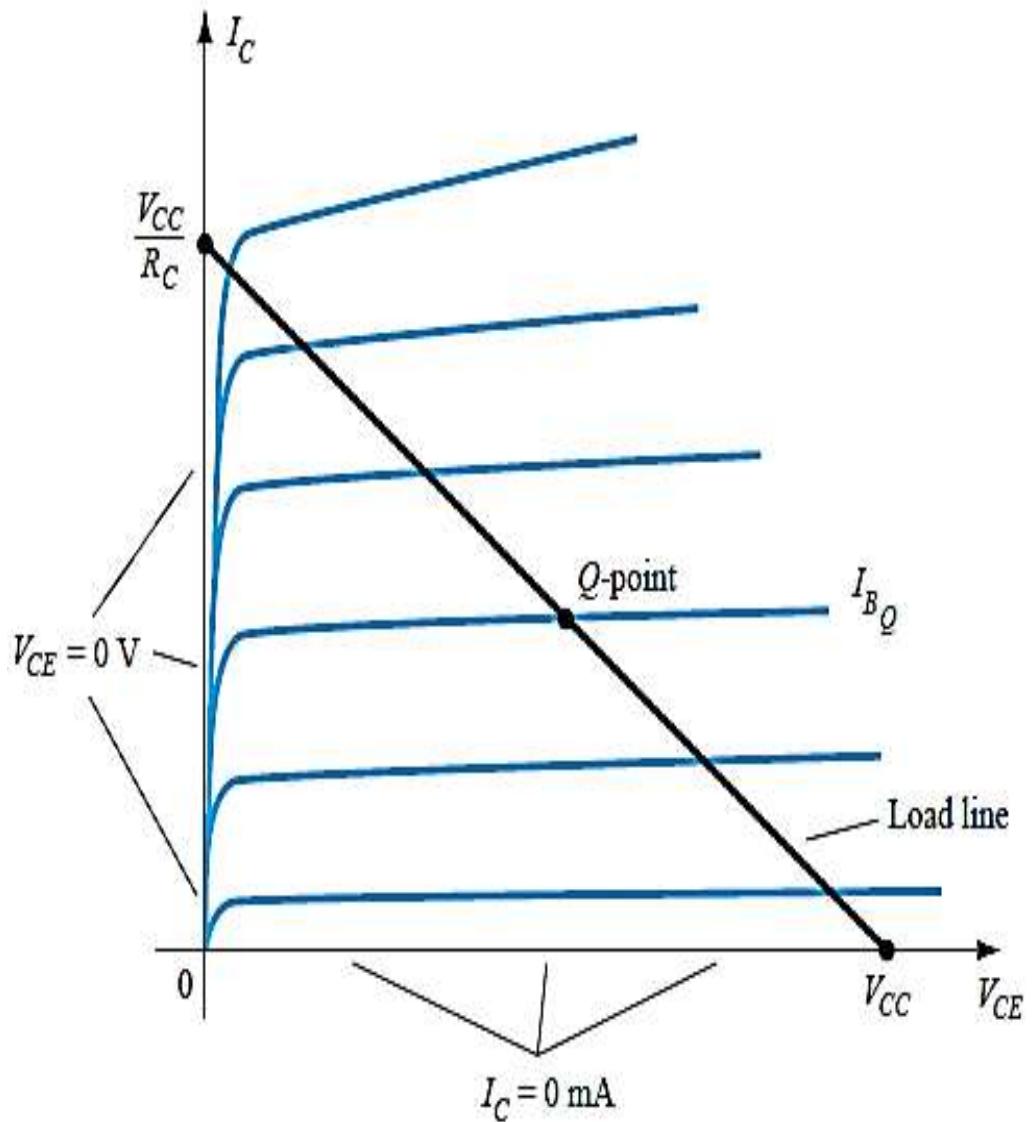
$$V_{CE} = V_{CC} - (0)R_C$$

$$V_{CE} = V_{CC} \Big|_{I_C=0 \text{ mA}}$$

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

$$I_C = \frac{V_{CC}}{R_C} \Big|_{V_{ce} = 0 \text{ V}}$$

# DC BIASING



- The correct operating point requires the selection of bias resistors and load resistors to provide the appropriate input current and collector voltage conditions.
- The correct biasing point for a bipolar transistor, either NPN or PNP, generally lies somewhere between the two extremes of operation with respect to it being either “fully-ON” or “fully-OFF” along its DC load line. This central operating point is called the “Quiescent Operating Point”, or **Q-point** for short.

Figure 4.12 Fixed-bias load line.

# Types of DC biasing

- Fixed **bias**.
- Collector-to-base **bias**.
- Fixed **bias** with emitter resistor.
- Voltage divider **bias** or potential divider.
- Emitter **bias**.

# Uses of BJT

The bipolar junction transistor (BJT) is used in logic circuits.

The BJT is used as an oscillator.

It is used as an amplifier.

It is used as a multivibrator.

For wave shaping, it is used in clipping circuits.

Used as a detector or demodulator.

It is also used as a modulator.

Used in timer and time delay circuits.

It is used in electronics switch.

It is used in switching circuits.

# **UNIT 1**

## **Introduction to Analog Electronics**

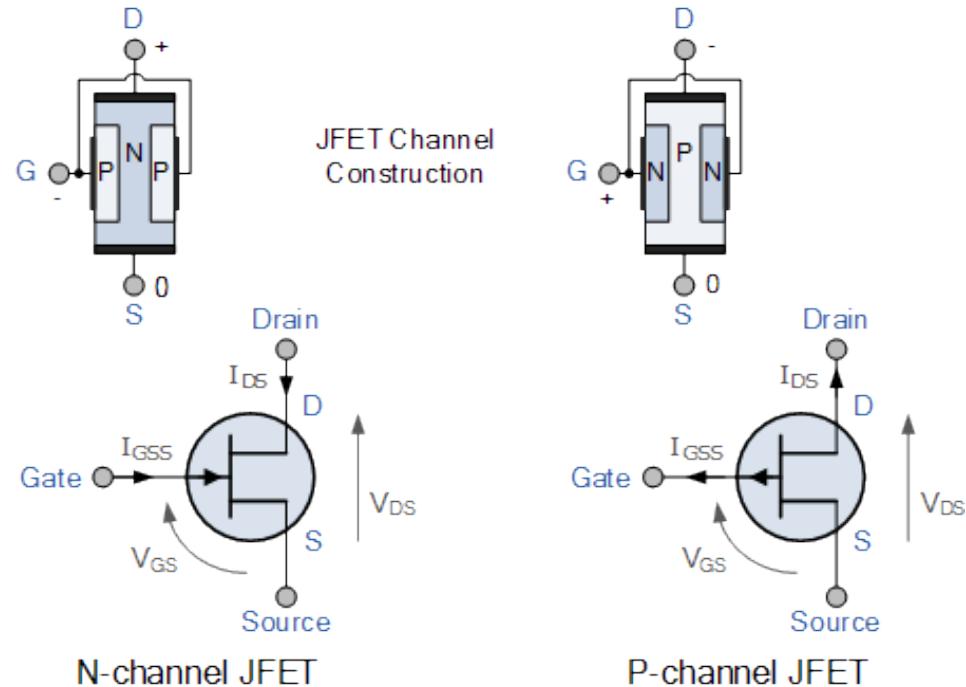
### **Session-2**

Characteristics of JFET (Common Source, Common Drain and Common Gate configurations) and uses

Differences between BJT and JFET

# JUNCTION FIELD EFFECT TRANSISTOR (JFET)

- There are two types of JFET's:  
n-channel and p-channel.
- The **n-channel** is widely used.
- Three terminals:
  - Drain (D) and Source (S) are connected to n-channel
  - Gate (G) is connected to the p-type material.
- Gate is always **reverse biased**
- Gate current,  $I_G=0$

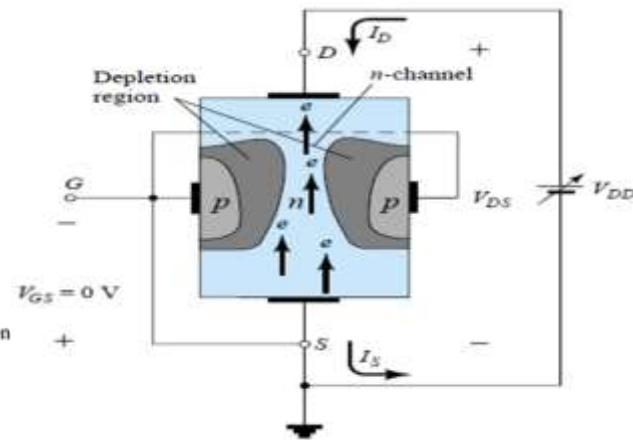
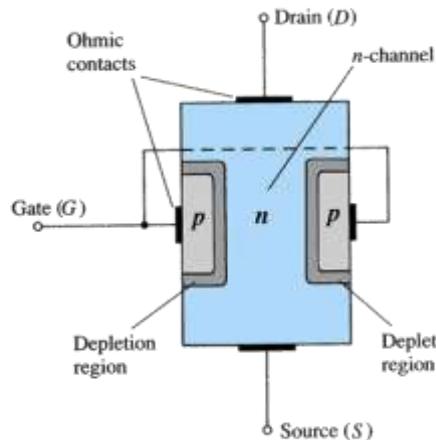


## CONSTRUCTION, TYPES & SYMBOLS

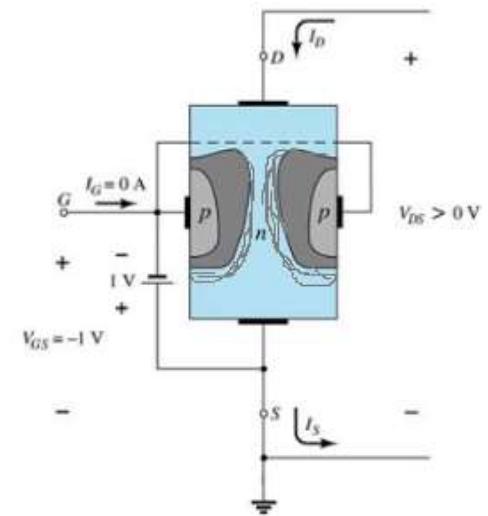
(Image source: [www.electronics-tutorials.ws](http://www.electronics-tutorials.ws))

# JFET (CS) WORKING

The depletion region becomes thicker with increased reverse bias



When  $V_{GS} = 0$



When  $V_{GS} < 0$

(Adapted from: Electronic devices and Circuit theory, Robert L Boylestad and Louis Nashelsky)

# Application

- The junction field effect transistor (JFET) is used as a constant current source.
- The JFET is used as a buffer amplifier.
- The JFET is used as an electronic switch.
- The JFET is used as a phase shift oscillator.
- The JFET is used as high impedance wide band amplifier.
- The JFET is used as a voltage variable resistor (VVR) or voltage development resistor (VDR).
- The JFET is used as a chopper

# JFET AMPLIFIER-CONFIGURATIONS

## 1. Common Source (CS) configuration (Good voltage amplifier)

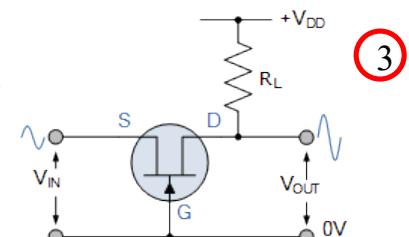
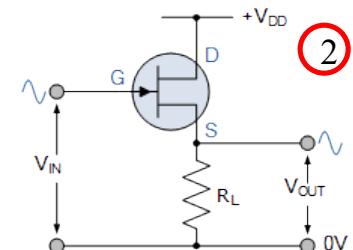
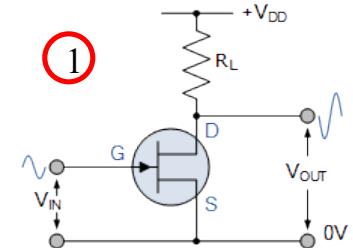
- Mostly used, Similar to CE transistor
- Generally used in audio frequency amplifiers and in high input impedance pre-amplifier stages.

## 2. Common Drain (CD) configuration (Good voltage buffer)

- Source follower
- High input impedance and a low output impedance
- Approx. unity voltage gain-used in buffer amplifiers.
- referred to as “Common Drain” because there is no signal available at the drain connection.

## 3. Common Gate (CG) configuration (Good current buffer)

- Has a low input impedance, but a high output impedance.
- Applied in high frequency circuits or in impedance matching circuits where a low input impedance needs to be matched to a high output impedance
- Microphone amplifiers

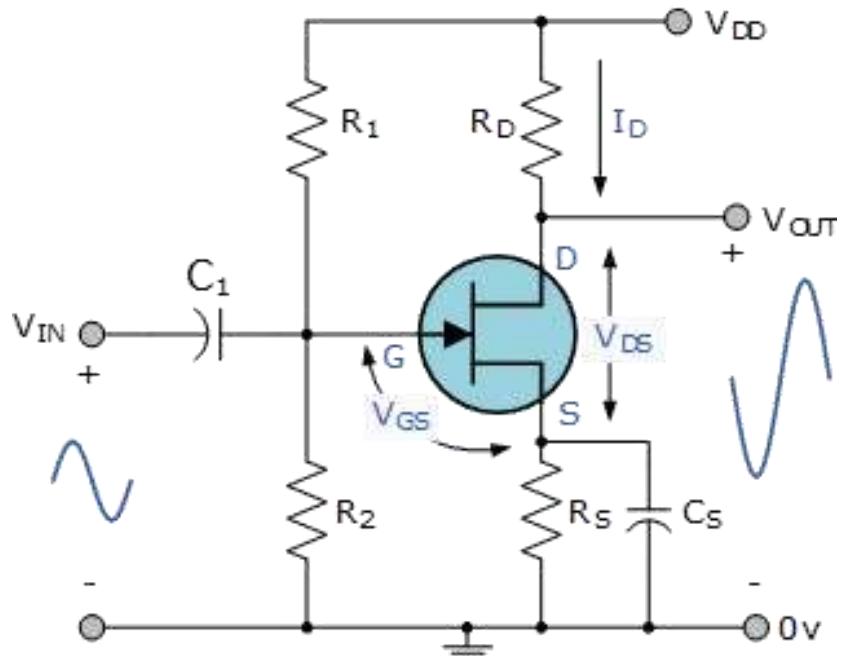


# COMMON SOURCE AMPLIFIER

- In CS configuration, the input is given to the gate and the output is taken from the drain.
- 180 degree phase shift between input and output.

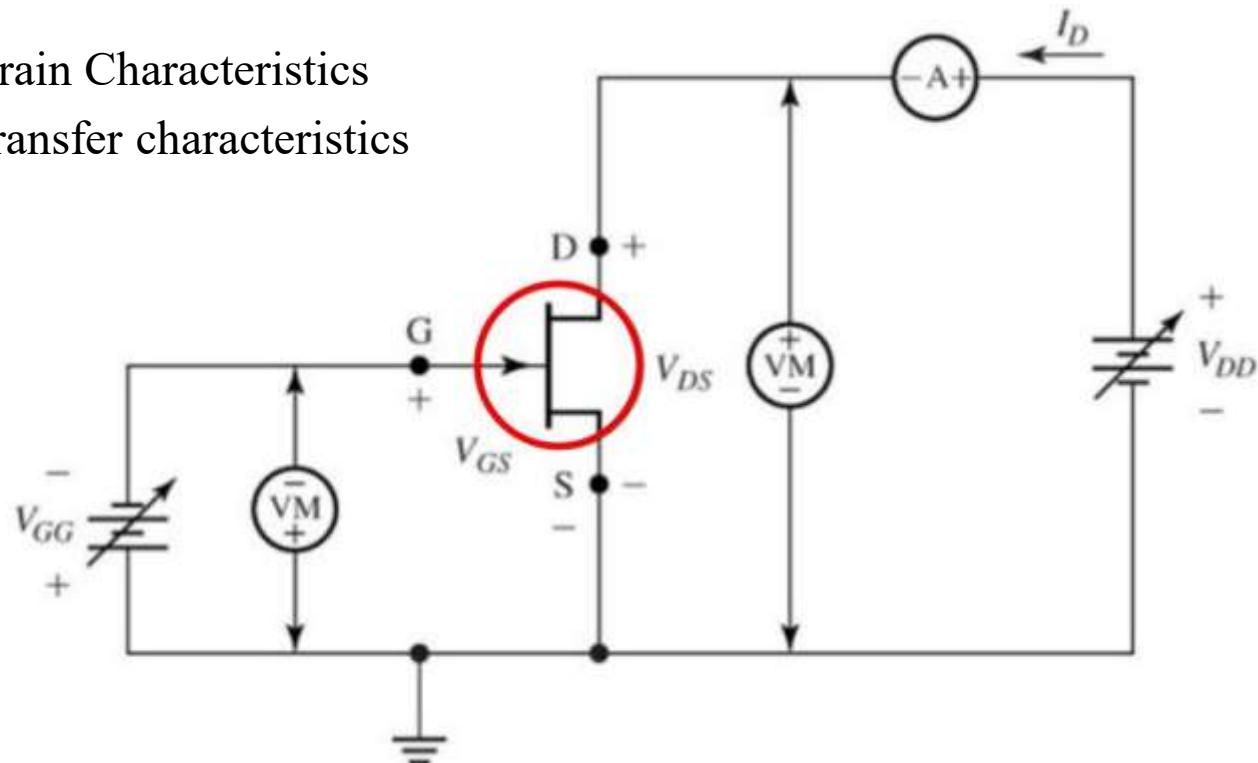
## Features

- Large voltage gain
- Good voltage amplifier
- Better transconductance amplifier
- High input resistance
- Medium / high output resistance



## CONNECTION DIAGRAM FOR DETERMINING THE CS AMPLIFIER CHARACTERISTICS

- Drain Characteristics
- Transfer characteristics



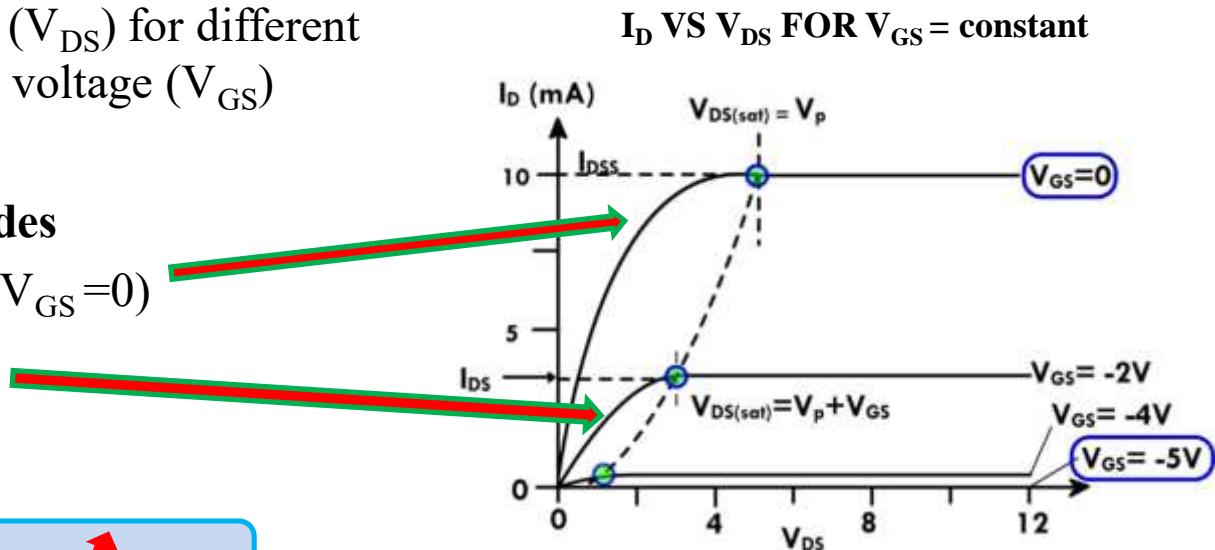
# CS AMPLIFIER CHARACTERISTICS

## Drain or V-I characteristics

- Output characteristics
- Relationship between Drain current ( $I_D$ ) and Drain to source voltage ( $V_{DS}$ ) for different values of gate to source voltage ( $V_{GS}$ )

## Operation analysis includes

- Without external bias: ( $V_{GS} = 0$ )
- With external bias.



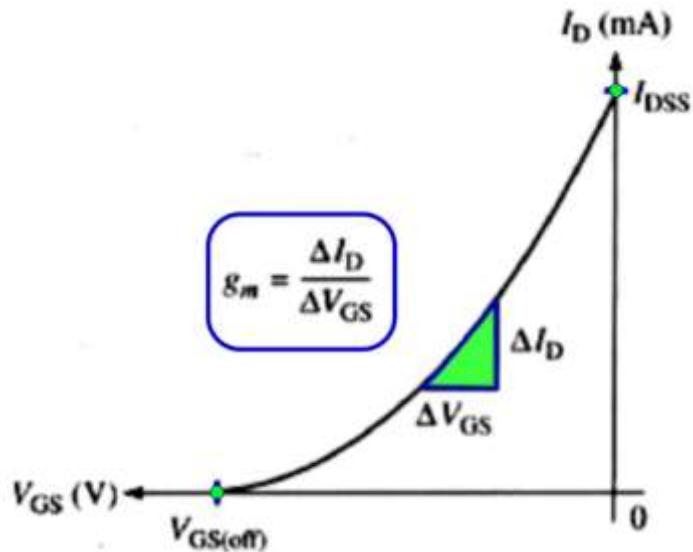
# CS AMPLIFIER CHARACTERISTICS

## The transfer characteristics

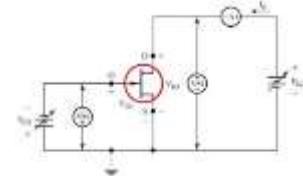
- The plot of the drain current Vs gate-source voltage for different values of drain to source voltage ( $V_{GS}$ )
- It is observed that the value of drain current varies **inversely** with respect to gate-source voltage when the drain-source voltage is constant.

$$I_D = I_{DSS} \left( 1 - \frac{V_{GS}}{V_P} \right)^2$$

The transfer characteristics is drawn between input voltage and output current.



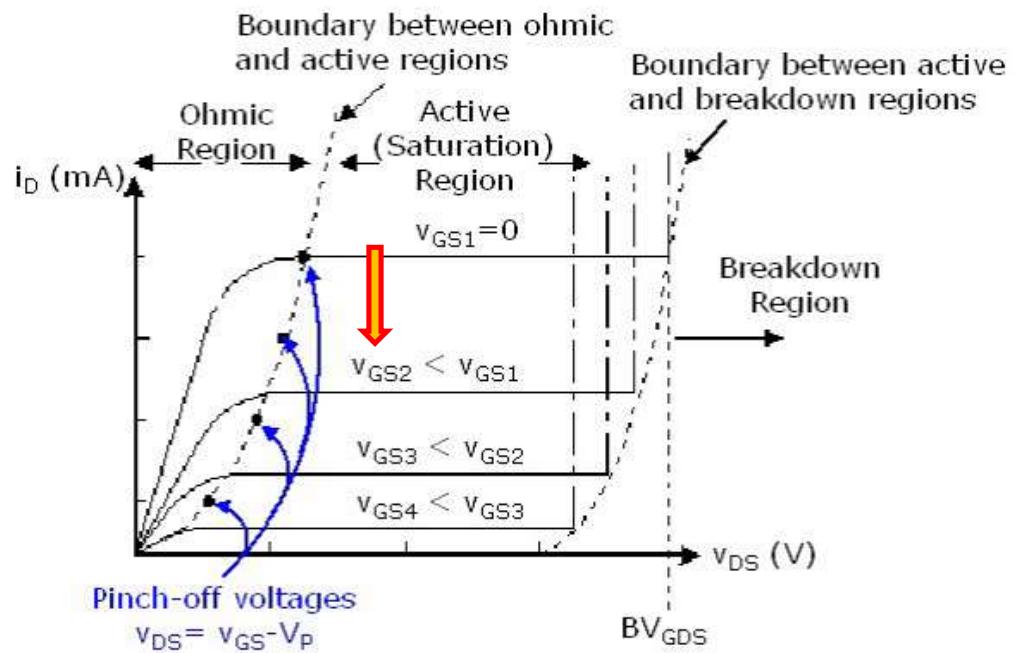
$I_D$  VS.  $V_{GS}$  FOR  $V_{DS}$  = constant



# CS AMPLIFIER CHARACTERISTICS

At the pinch-off point,

- $V_{GS}$  at pinch-off is denoted as  $V_p$
- any further increase in  $V_{DS}$  does not produce any increase in  $I_D$
- $I_D$  is at saturation or maximum. It is referred to as  $I_{DSS}$ .



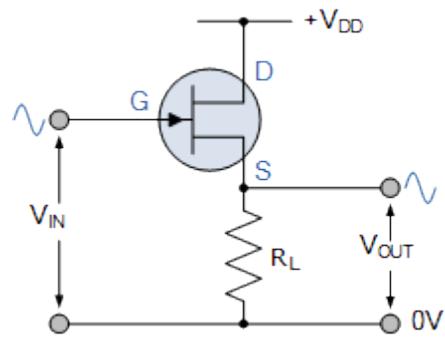
(Image source: [www.rezzonics.blogspot.com](http://www.rezzonics.blogspot.com))

# CS AMPLIFIER CHARACTERISTICS

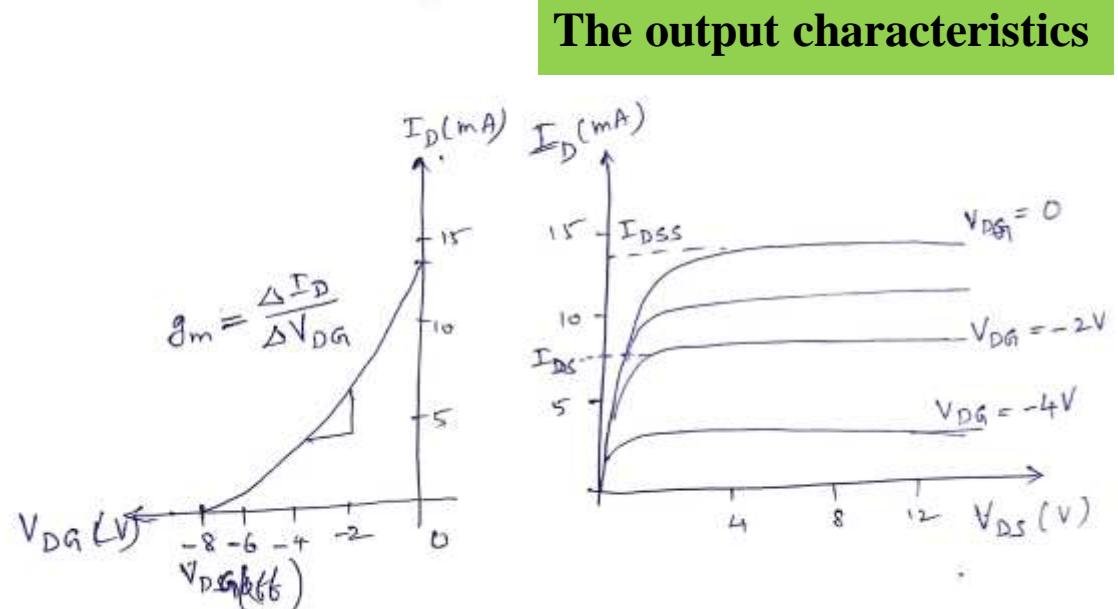
## Important terms in JFET drain characteristics

- **Knee Point:** Where the variation of drain current with drain-source voltage appears to be linear. But beyond this point, the linearity changes into a curve.
- **Channel Ohmic Region:** The region to the left of the knee point in the characteristics curve.
- **Pinch-off point:** The point in the curve beyond which the drain current will not increase further no matter how much we are increasing the drain to source voltage.
- **Pinch-off Voltage:** The voltage at the pinch-off point is termed as pinch-off voltage because at this voltage, the current is completely turned to be constant.
- **Drain-Source Saturation Current:** The drain to source saturation current is the current which becomes constant and enters into a saturation state.

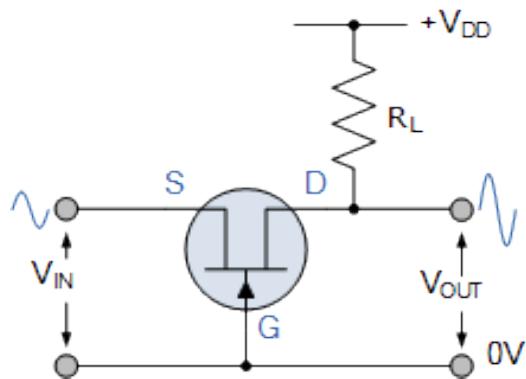
# COMMON DRAIN JFET AMPLIFIER (Source follower)



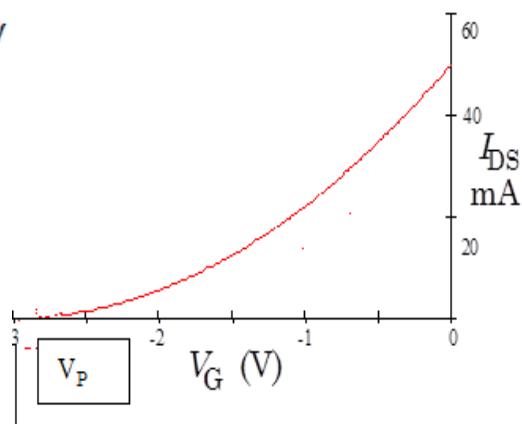
Connect voltmeters and ammeters to measure the necessary voltages and currents (as explained in CS amplifier connections)



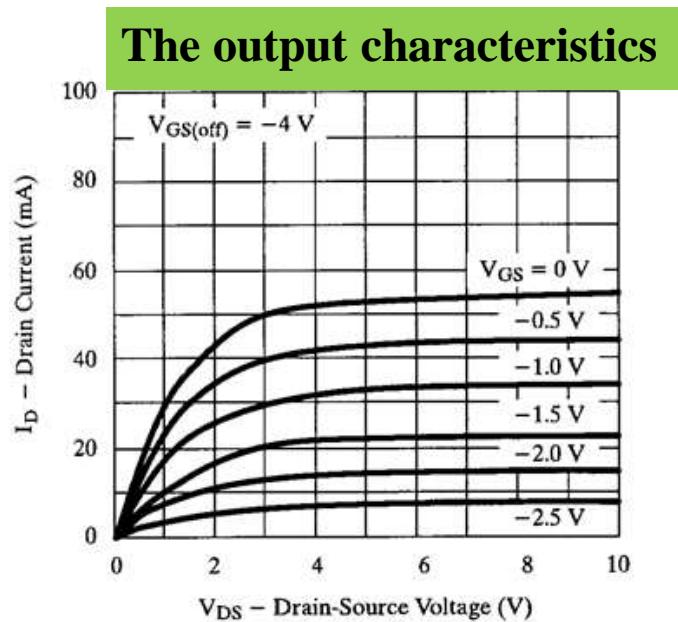
# COMMON GATE JFET AMPLIFIER-CHARACTERISTICS



Connect voltmeters and ammeters to measure the necessary voltages and currents (as explained in CS amplifier connections)



The transfer characteristics



## COMPARISON BETWEEN BJT AND JFET

BJT	JFET
Bipolar device (current condition, by both types of carriers, i.e. majority and minority-electrons and hole).	Unipolar device (current conduction is only due to one type of majority carrier either electron or hole).
The operation depends on the injection of minority carries across a forward biased junction.	The operation depends on the control of a junction depletion width under reverse bias.
Current controlled device. The base current controls the output current.	Voltage controlled device. The gate voltage controls output current.
High noise level. (current conduction through junctions)	Low noise level. ( current conduction is through n-channel or p-channel and no junction crossing)
Low input impedance (due to forward bias at input side).	High input impedance (due to reverse bias).
Gain is characterized by voltage gain.	Gain is characterised by transconductance.
Low thermal stability. (positive temperature coefficient at high current levels lead to thermal breakdown)	Better thermal stability.(NTC at high current levels prevent thermal breakdown)
Cheaper	Relatively costly

## **REVIEW QUESTIONS**

**Source follower is a circuit that provides\_\_\_\_\_**

**In CS amplifier, the drain-source current is largest when the gate-source voltage  $V_{GS}$  is\_\_\_\_\_**

**Which JFET amplifier configuration is used in microphone amplifiers?**

**JFET is a \_\_\_ controlled device**

# **UNIT 1**

## **Introduction to Analog Electronics**

**NEXT**

**Session-3**

Transistor Amplifier: CE amplifier

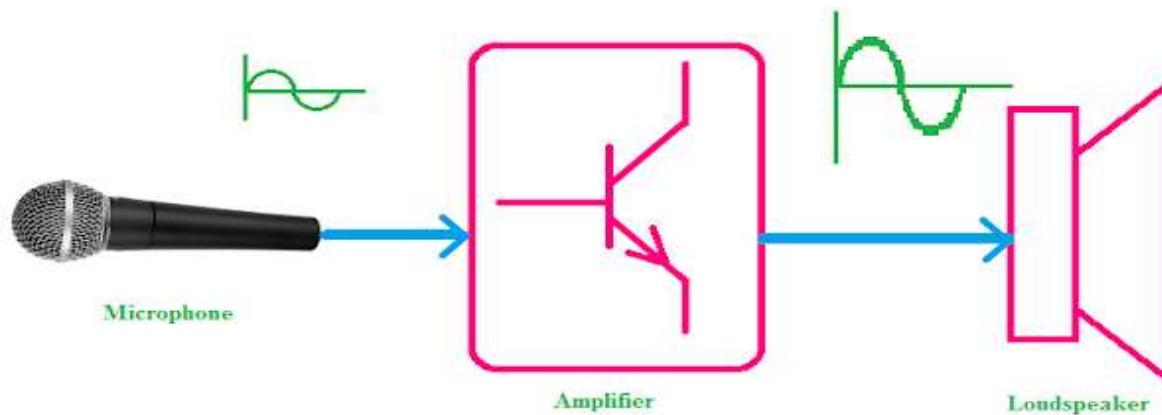
Transistor Amplifier: CC ,CB amplifier

UNIT 1  
Introduction to Analog Electronics  
Session-3

Transistor Amplifier: CE amplifier  
Transistor Amplifier: CB and CC amplifier

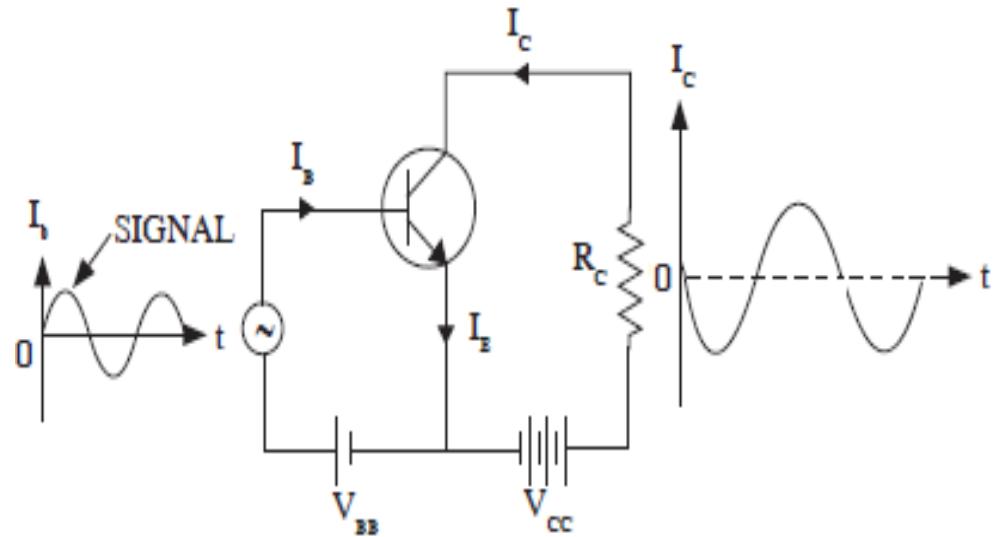
# Transistor Amplifier

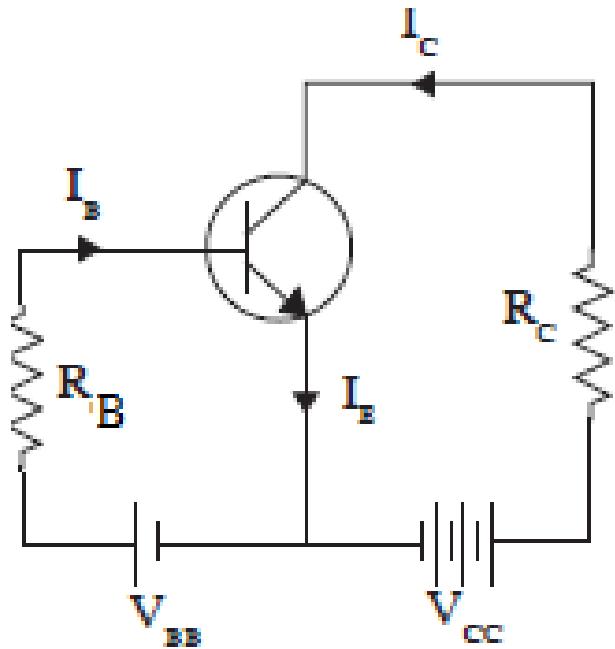
- The transistor raises the strength of a weak signal and hence acts **an amplifier**
- **The transistor works as an amplifier in all configuration (CE, CB and CC)**



# Transistor Amplifier: CE amplifier

- npn transistor in CE mode
- Input between Base and Emitter
- Output between Collector and Emitter
- Input current  $I_B$
- Output current  $I_C$
- Input junction (BE)-forward biased
- Output junction (CE)-Reverse biased
- The  $V_{BB}$  battery provides the forward bias
- The  $V_{CC}$  battery provides the reverse bias





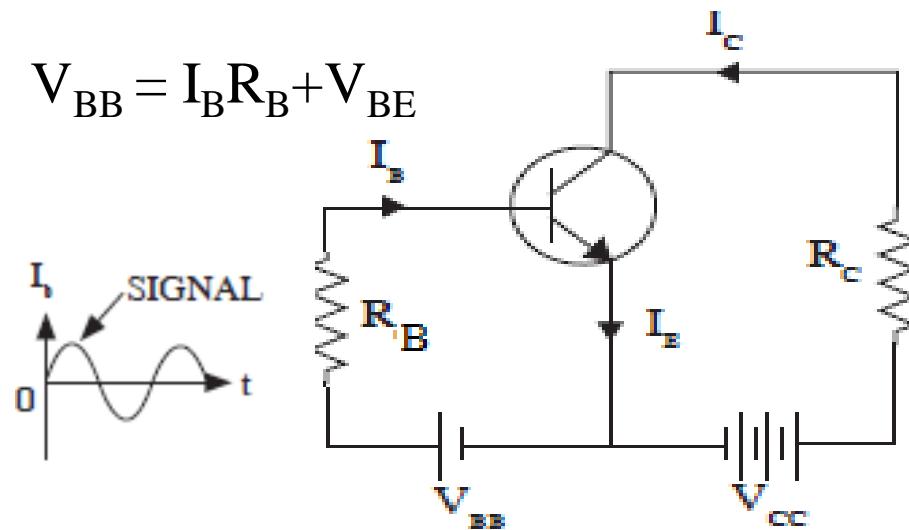
Write KVL in Input circuit

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

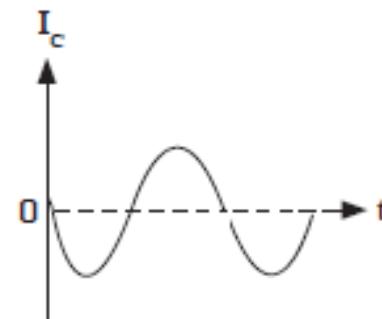
Write KVL in output circuit

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

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

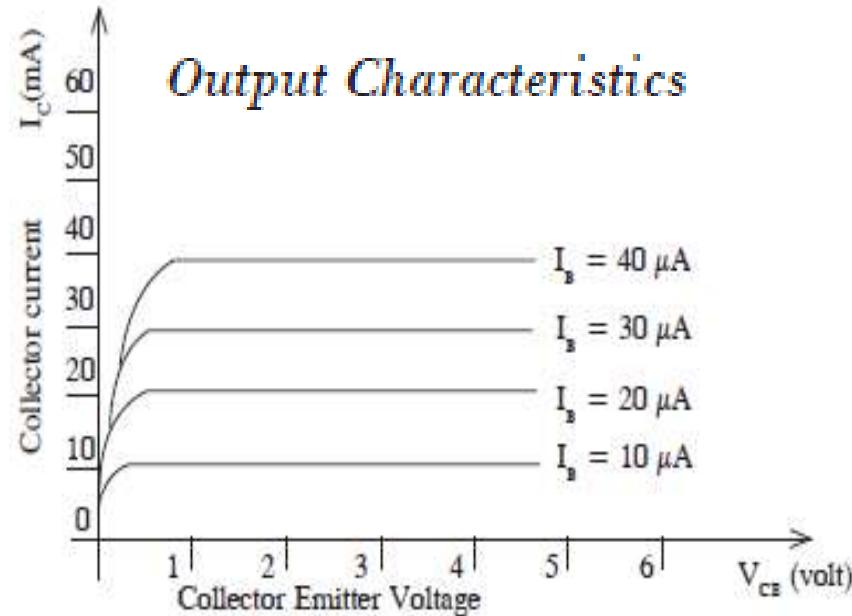


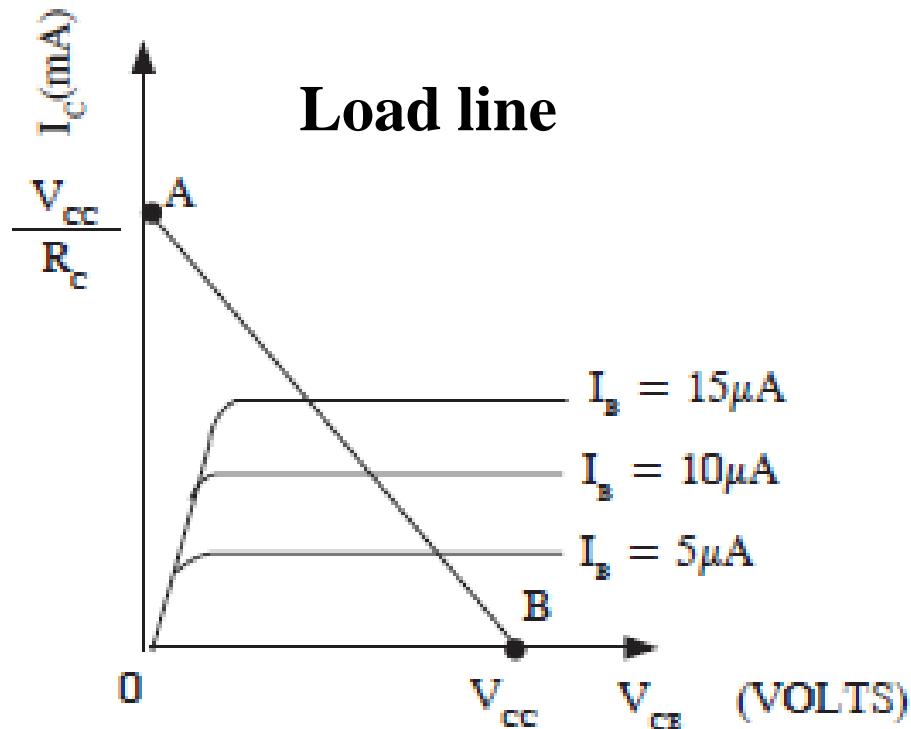
$$V_{CE} = V_{CC} - I_C R_C$$



Link between input and output current

- $I_B$  increases  $I_C$  increases
- $I_B$  decreases  $I_C$  decreases
- Variation in collector current gives variation in output voltage
- Transistor works as a amplifier in proper operating point





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

$$V_{CE} = 0$$

$$I_C(\text{sat}) = V_{CC}/R_C$$

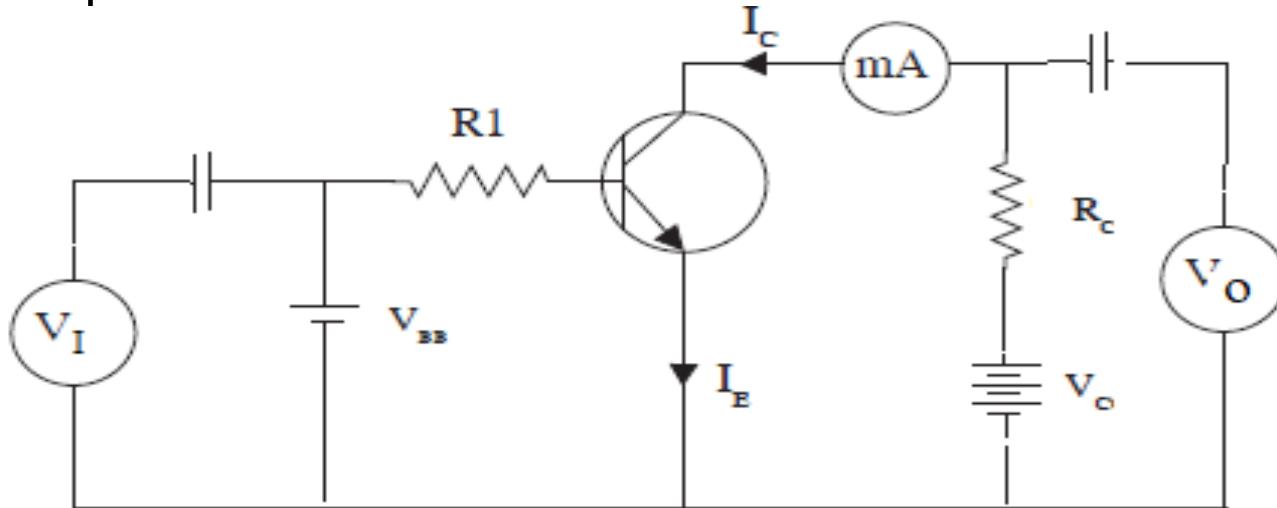
$$V_{CE} = V_{CC} - I_C R_C$$

$$I_C = 0$$

$$V_{CE \text{ Max}} = V_{CC} \text{ or } V_{CE \text{ (Cut)}} = V_{CC}$$

- Each point of load line is the operating point of a transistor

# Amplifier



- Capacitor connected to the input and output side remove **DC component**

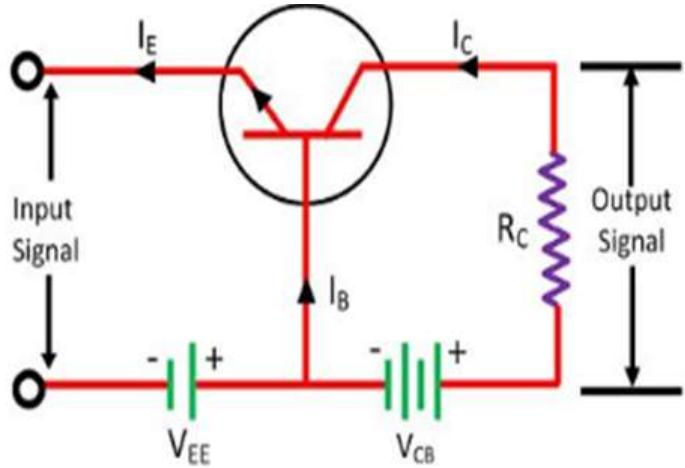
$$V_{CE} = V_{CC} - I_C R_C$$

- $I_C$  increases :  $V_{CE}$  voltage decreases
  - Voltage gain  $A_V = V_{CE}/V_I$

## Operation of Common Emitter Amplifier

- When a signal is applied across the emitter-base junction during the positive half cycle the forward bias across this junction increases.
- This increases the flow of electrons from the emitter to a collector through the base, thus increases the collector current.
- The increasing collector current induces more voltage drops across the collector load resistor  $R_C$ .
- The negative half cycle decreases the forward bias voltage across the emitter-base junction.
- The decreasing collector-base voltage reduces the collector current in the whole collector resistor  $R_C$ .
- Thus, the amplified load resistor appears across the collector resistor

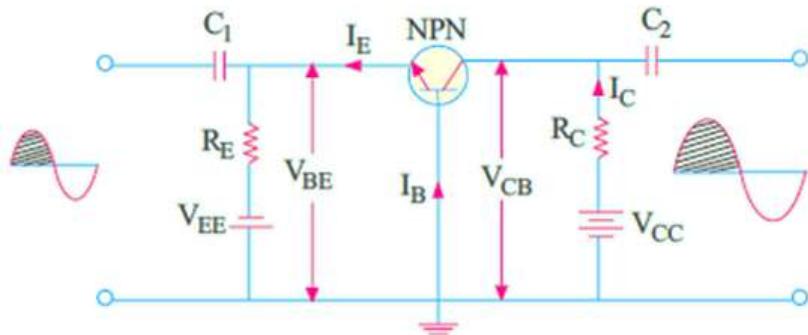
## Common Base Transistor as an Amplifier



- The emitter and base of the transistor are connected in forward biased and the collector base region is in reverse bias.
- The input signal or weak signal is applied across the emitter base and the output is obtained to the load resistor  $R_C$  which is connected in the collector circuit.
- The DC voltage  $V_{EE}$  is applied to the input circuit along with the input signal to achieve the amplification

- When a weak signal is applied to the input, a small change in signal voltage causes a change in emitter current (or we can say a change of 0.1V in signal voltage causes a change of 1mA in the emitter current) because the **input circuit has very low resistance**.
- This change is almost the same in collector current because of the transmitter action.
- In the collector circuit, a load resistor  **$R_C$  of high value is connected**.
- When collector current flows through such a high resistance, it produces **a large voltage drop across it**.
- Thus, a weak signal (0.1V) applied to the input circuit appears in the amplified form (10V) in the collector circuit.

## Common Base (CB) Amplifier

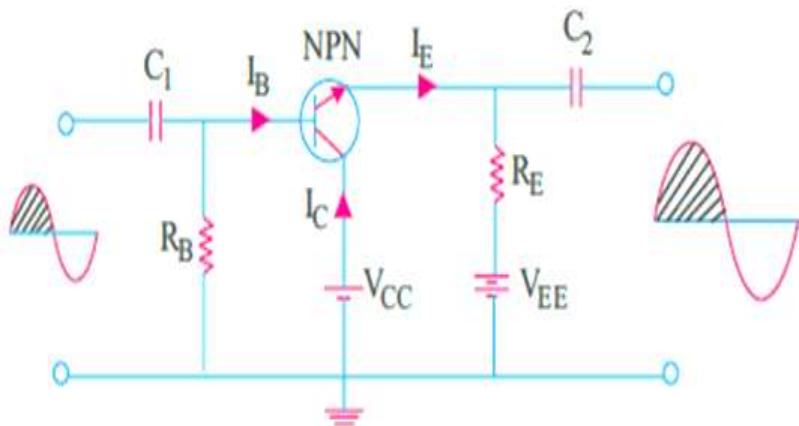


When positive half-cycle of the signal is applied, then

1. forward bias is **decreased** because  $V_{BE}$  is already negative with respect to the ground
2. consequently,  $I_B$  is **decreased**.
3.  $I_E$  and hence  $I_C$  are **decreased**
4. The drop  $I_C R_C$  is **decreased**.
5. Hence,  $V_{CB}$  is **increased**

- Since a **positive-going** input signal produces a **positive-going** output signal, there is no phase reversal
- Voltage amplification in this circuit is possible by reason of relative input and output circuitry rather than current gain ( $\alpha$ ) which is always less than unity.
- The input circuit has low resistance whereas output circuit has very large resistance.
- Although changes in input and output currents are the same, the ac drop across  $RL$  is very large.
- Hence, changes in  $V_{CB}$  (which is the output voltage) are much larger than changes in input ac signal. Hence, the voltage amplification.

## Common Collector (CC) Amplifier



1. forward bias is **increased** since  $VBE$  is positive w.r.t. collector i.e. ground,
2. base current is **increased**,
3. emitter current is **increased**,
4. drop across  $RE$  is **increased**,
5. hence, output voltage (*i.e.* drop across  $RE$ ) is **increased**.

- When a **positive-going** input signal results in a **positive going** output signal and, consequently, the input and output signals are in phase

# **POWER AMPLIFIERS**

- Power amplifiers amplify the power level of the signal.
- The power of the input signal is increased to a level high enough to drive loads of output devices like speakers, headphones, RF transmitters etc.

## **CLASSIFICATIONS**

### **Based on their frequencies.**

- Audio Power Amplifiers – Audio frequency range 20 Hz to 20 KHz. Also Known as Small Signal Power Amplifiers.
- Radio Power Amplifiers – Power level up to 3 Hz to 300 Hz. Also termed as Tuned Amplifiers and also known as Large Signal Power Amplifiers.

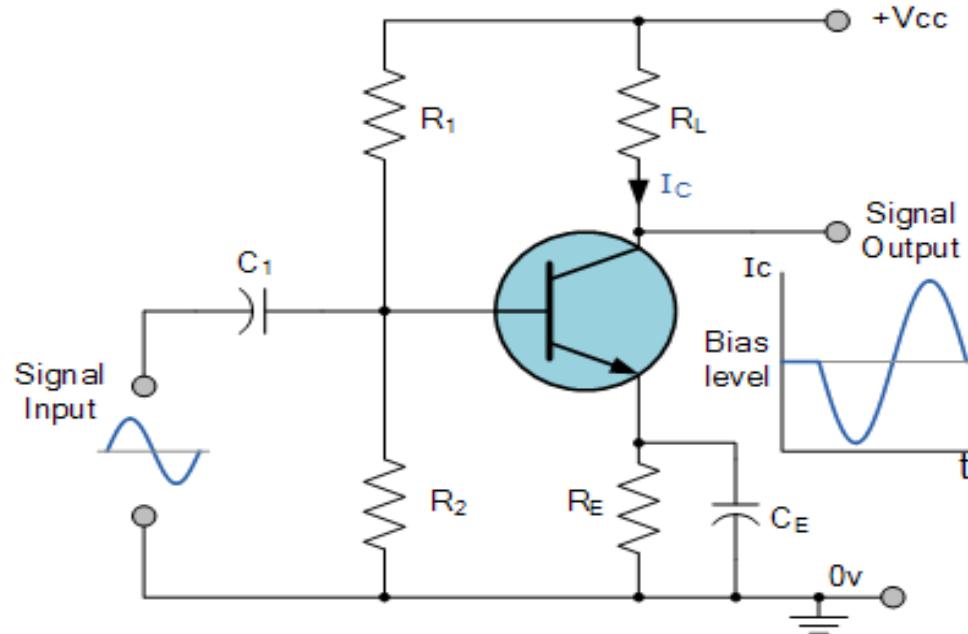
### **Mode of operation**

The portion of the input cycle during which collector current flows, the power amplifiers may be classified as follows

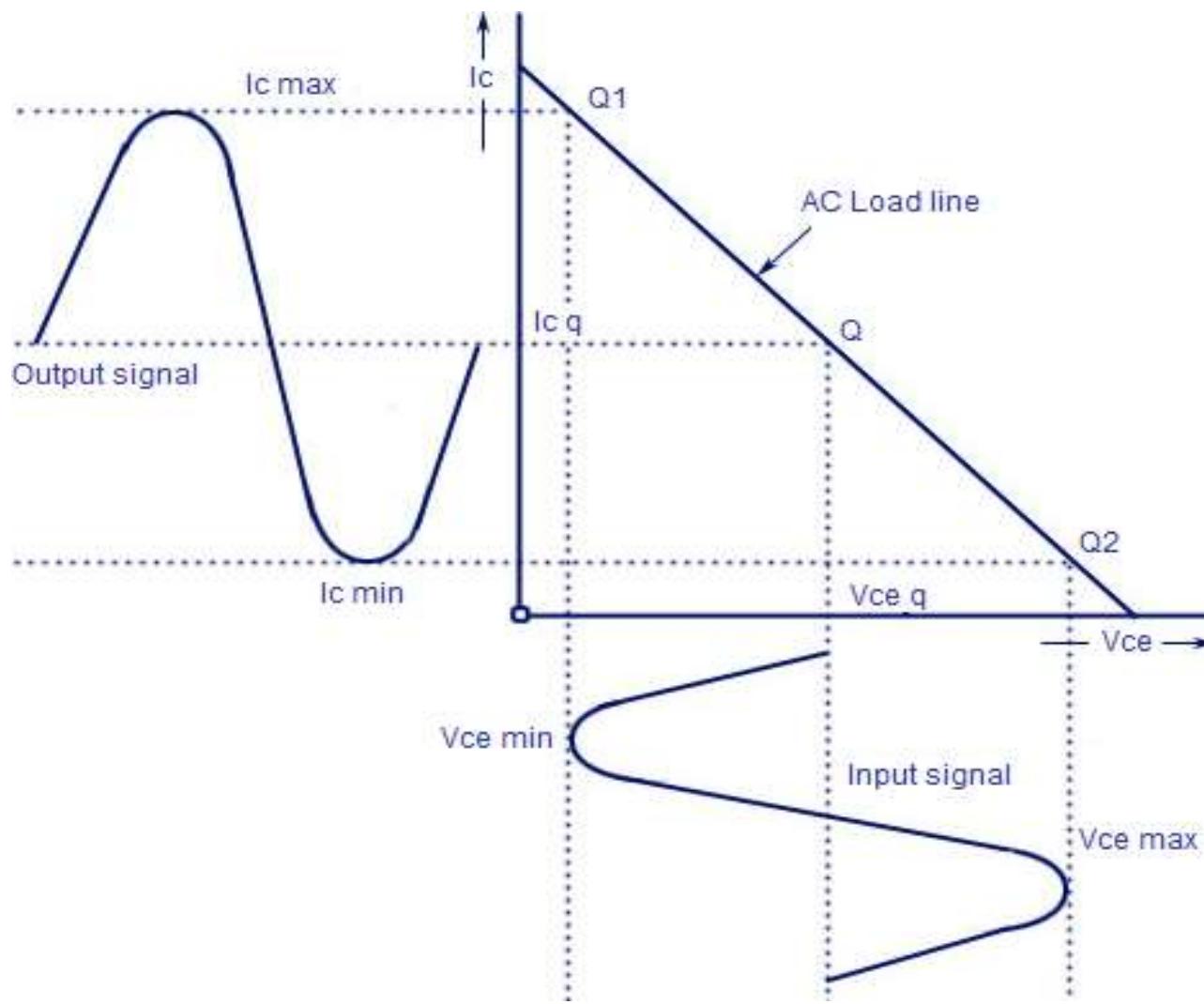
- **Class A Power amplifier** - When the collector current flows at all times during full cycle of the input signal.
- **Class B Power amplifier** - When the collector current flows at only during half cycle of the input signal.
- **Class C Power amplifier** - When the collector current flows for less than half cycle of the input signal.

# CLASS - A AMPLIFIER

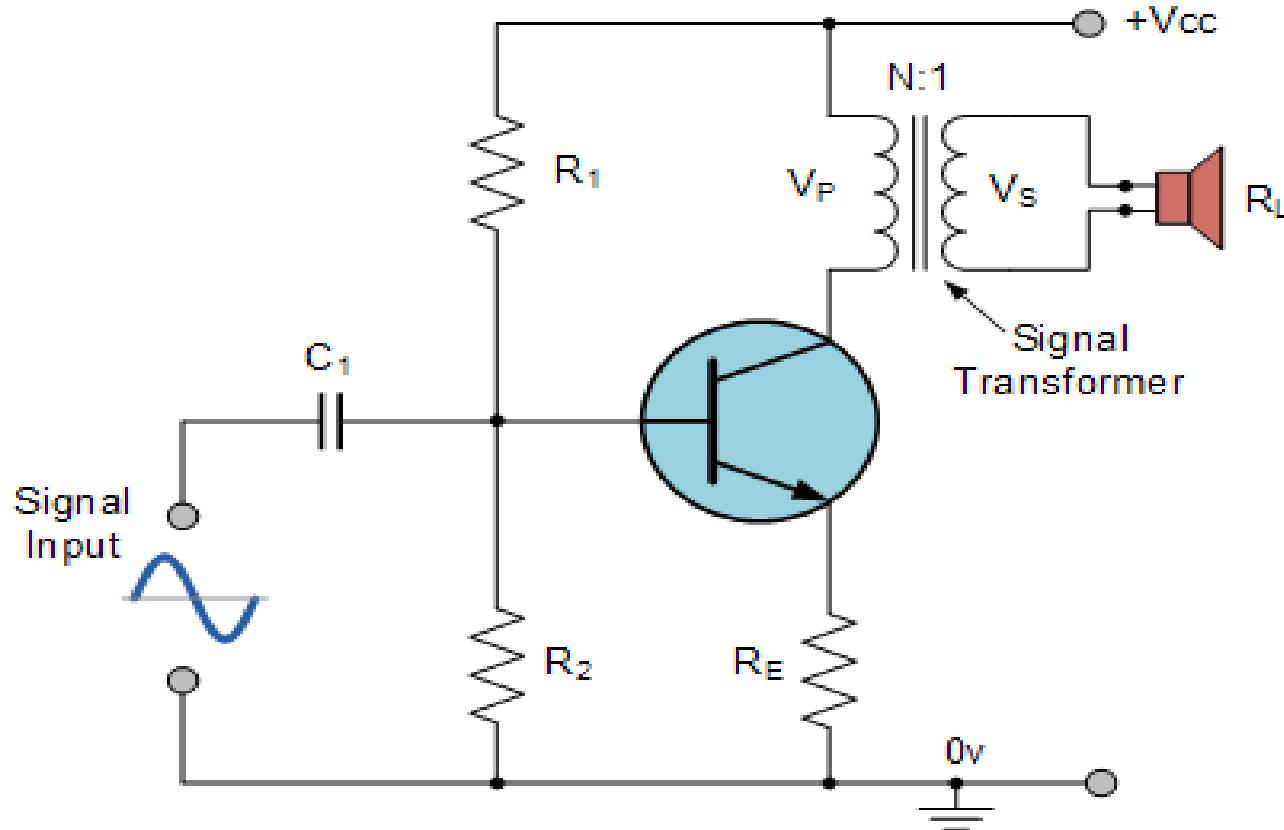
- Commonly used type of power amplifier configuration is the **Class A Amplifier**.
- It uses a single switching transistor in the standard common emitter circuit configuration.
- The transistor is always biased “ON” so that it conducts during one complete cycle of the input signal waveform producing minimum distortion and maximum amplitude of the output signal.



# CHARACTERISTIC CURVES- CLASS A POWER AMPLIFIER

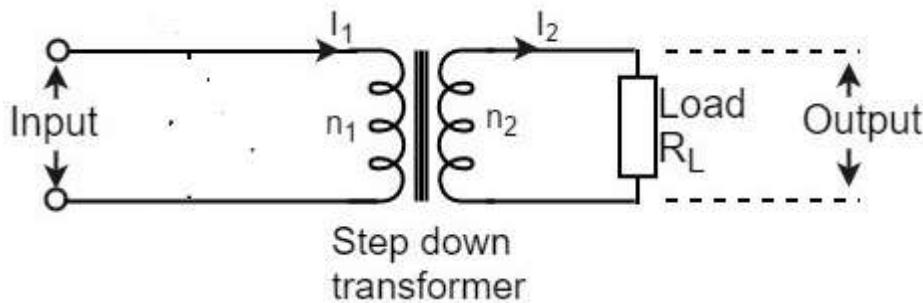


# TRANSFORMER COUPLED CLASS – A AMPLIFIER



TRANSFORMER COUPLED CLASS - A AMPLIFIER

- $R_1$  and  $R_2$  -potential divider. Resistor  $R_e$  provides stabilization,  $R_e$  to prevent A.C. voltage.
- The high impedance primary of the transformer is connected to the high impedance collector circuit. The low impedance secondary is connected to the load (generally loud speaker). Transformer used in the collector circuit is for impedance matching.
- $R_L$  is the load connected in the secondary of a transformer.
- Number of turns in the primary are  $n_1$  and the secondary are  $n_2$ .
- $V_1$  and  $V_2$  - primary and secondary voltages.  $I_1$  and  $I_2$  be the primary and secondary currents respectively.



- As the Collector current,  $I_C$  is reduced to below the quiescent Q-point set up by the base bias voltage, due to variations in the base current, the magnetic flux in the transformer core collapses causing an induced EMF in the transformer primary windings.
- This causes an instantaneous collector voltage to rise to a value of twice the supply voltage  $2V_{CC}$  giving a maximum collector current of twice  $I_C$  when the Collector voltage is at its minimum.
- The efficiency of this type of Class A amplifier configuration can be calculated as follow

$$\eta_{(\text{max})} = \frac{P_{\text{ac}}}{P_{\text{dc}}} \times 100\%$$

- R.M.S. Collector voltage is given as

$$V_{CE} = \frac{V_{C(\text{max})} - V_{C(\text{min})}}{2\sqrt{2}} = \frac{2V_{CC} - 0}{2\sqrt{2}}$$

- R.M.S. Collector current is given as

$$I_{CE} = \frac{I_{C(\text{max})} - I_{C(\text{min})}}{2\sqrt{2}} = \frac{2I_C - 0}{2\sqrt{2}}$$

- The r.m.s. Power delivered to the load ( $P_{ac}$ ) is therefore given as

$$P_{ac} = V_{CE} \times I_{CE} = \frac{2V_{CC}}{2\sqrt{2}} \times \frac{2I_C}{2\sqrt{2}} = \frac{2V_{CC} 2I_C}{8}$$

- The average power drawn from the supply ( $P_{dc}$ ) is given by

$$P_{dc} = V_{CC} \times I_C$$

- Efficiency of a Transformer-coupled Class A amplifier is given as

$$\eta_{(max)} = \frac{P_{ac}}{P_{dc}} = \frac{2V_{CC} 2I_C}{8V_{CC} I_C} \times 100\%$$

- An output transformer improves the efficiency of the amplifier by matching the impedance of the load with that of the amplifiers output impedance.
- By using an output or signal transformer with a suitable turns ratio, class-A amplifier efficiencies reaching 50%.

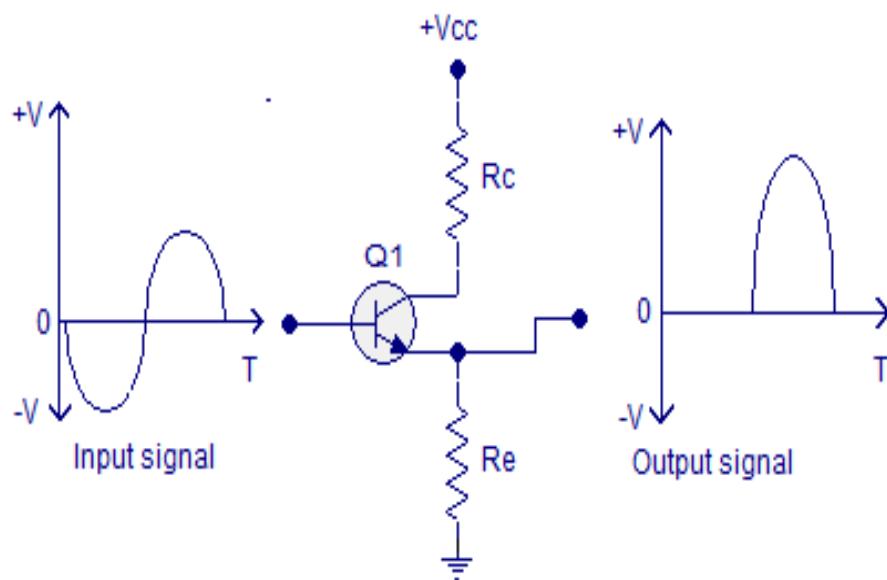
### **Advantages**

- Provides good DC isolation as there is no physical connection between amplifier output and load

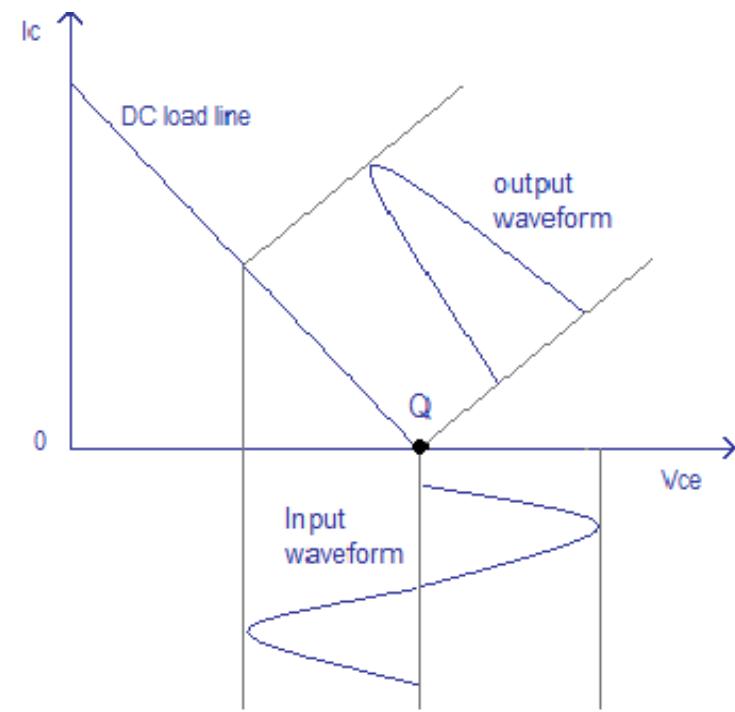
**Disadvantage** - Additional cost and size of the audio transformer required.

# CLASS B POWER AMPLIFIER

- It is a type of power amplifier where the active device (transistor) conducts only for one half cycle of the input signal. The conduction angle is  $180^\circ$  for a Class B amplifier.
- The active device is switched off for half the input cycle, the active device dissipates less power and hence the efficiency is improved.
- Theoretical maximum efficiency of Class B power amplifier is 78.5%.



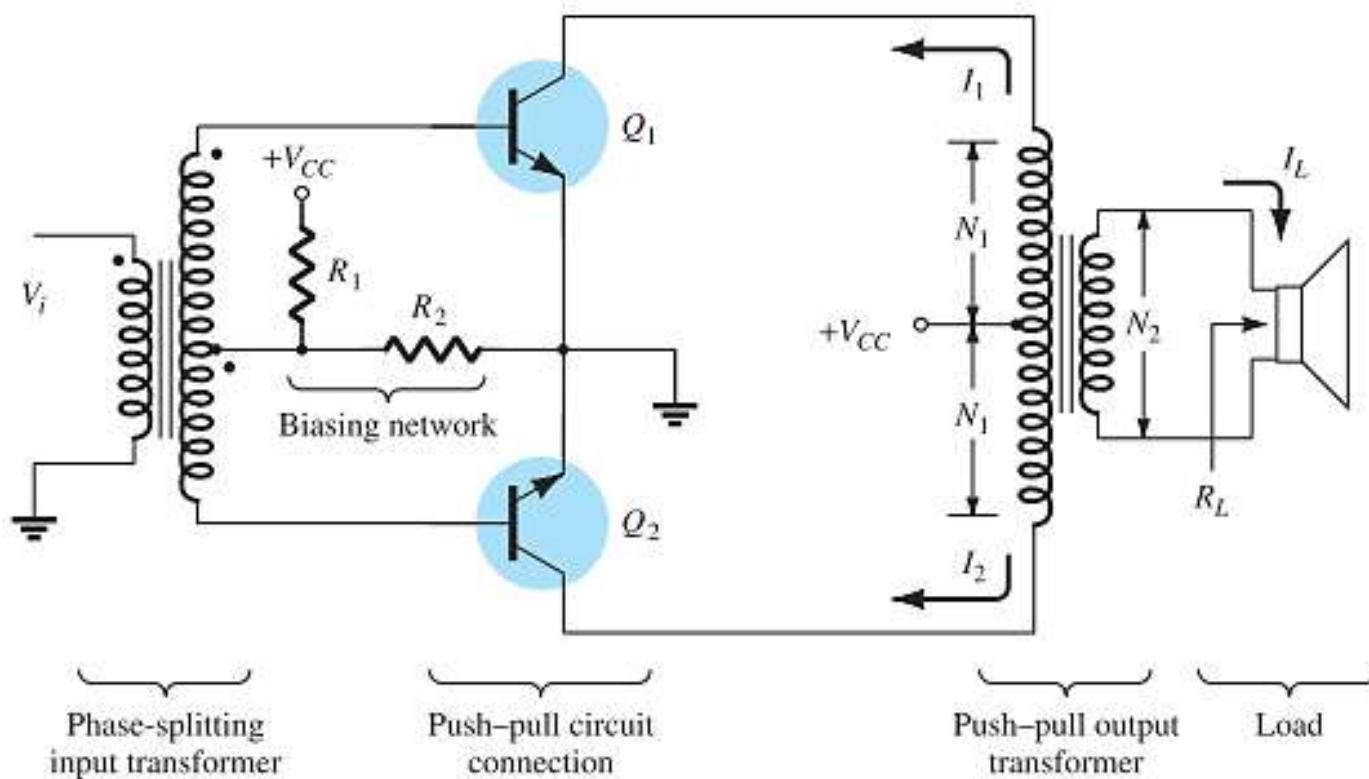
**Single ended Class B amplifier**



**Class B – Output characteristics**

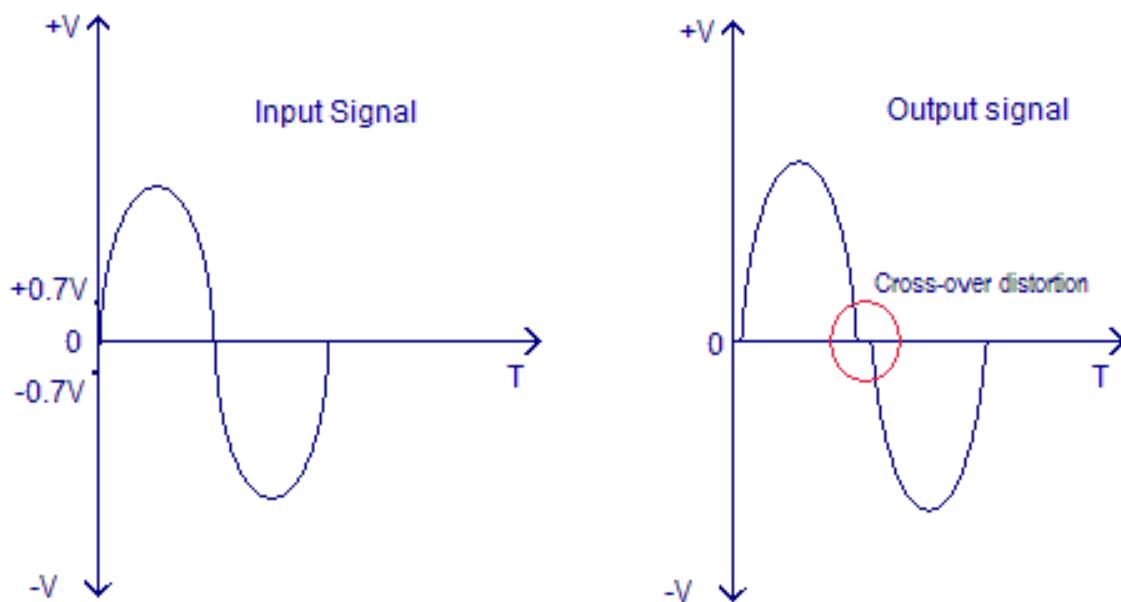
## CLASS B PUSH-PULL AMPLIFIER

- In practical Class B amplifier a pair of active devices (transistors) arranged in push-pull mode where one transistor conducts one half cycle and the other transistor conducts the other half cycle.
- The output from both transistors are then combined together to get a scaled replica of the input.



# CROSS OVER DISTORTION

- The regions of the input signal where the amplitude is less than 0.7V will be missing in the output signal and it is called cross over distortion.



## **ADVANTAGES OF CLASS B AMPLIFIER**

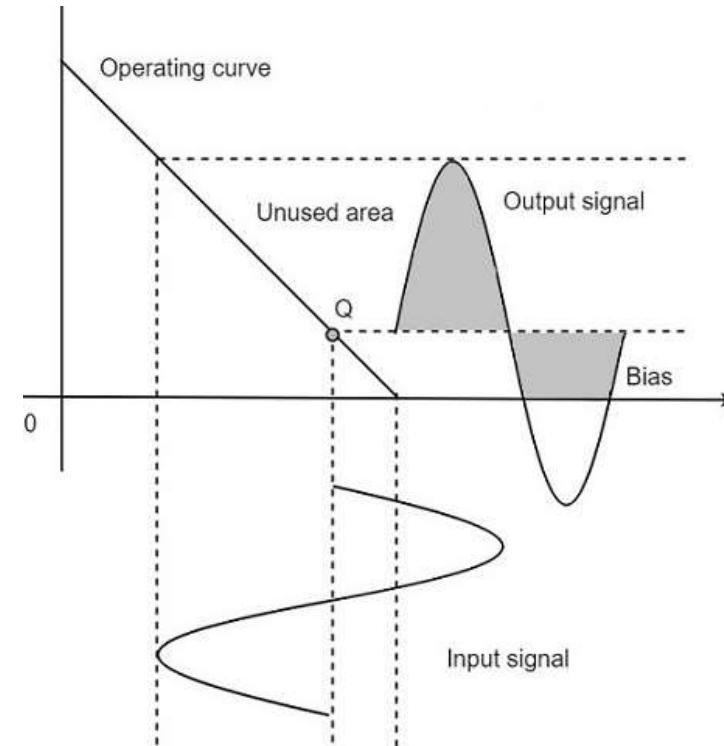
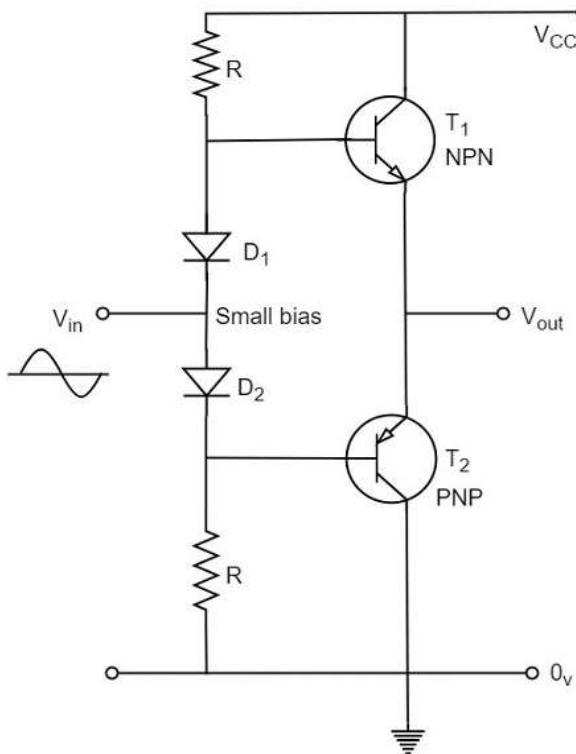
- High efficiency when compared to the Class A configurations.
- Push-pull mechanism avoids even harmonics.
- No DC components in the output (in ideal case).

## **DISADVANTAGES OF CLASS B AMPLIFIER.**

- The major disadvantage is the cross-over distortion. Coupling transformers increases the cost and size.
- It is difficult to find ideal transformers.
- Transformer coupling causes hum in the output and also affects the low frequency response.
- Transformer coupling is not practical in case of huge loads.

# CLASS AB POWER AMPLIFIER

- class AB is a combination of class A and class B type of amplifiers. As class A has the problem of low efficiency and class B has distortion problem, this class AB is emerged to eliminate these two problems, by utilizing the advantages of both the classes.
- The cross over distortion is the problem that occurs when both the transistors are OFF at the same instant, during the transition period. In order to eliminate this, the condition has to be chosen for more than one half cycle. Hence, the other transistor gets into conduction, before the operating transistor switches to cut off state.



## **Advantages of Class AB power amplifier.**

No cross over distortion.

No need for the bulky coupling transformers.

No hum in the output.

## **Disadvantages of Class AB power amplifier.**

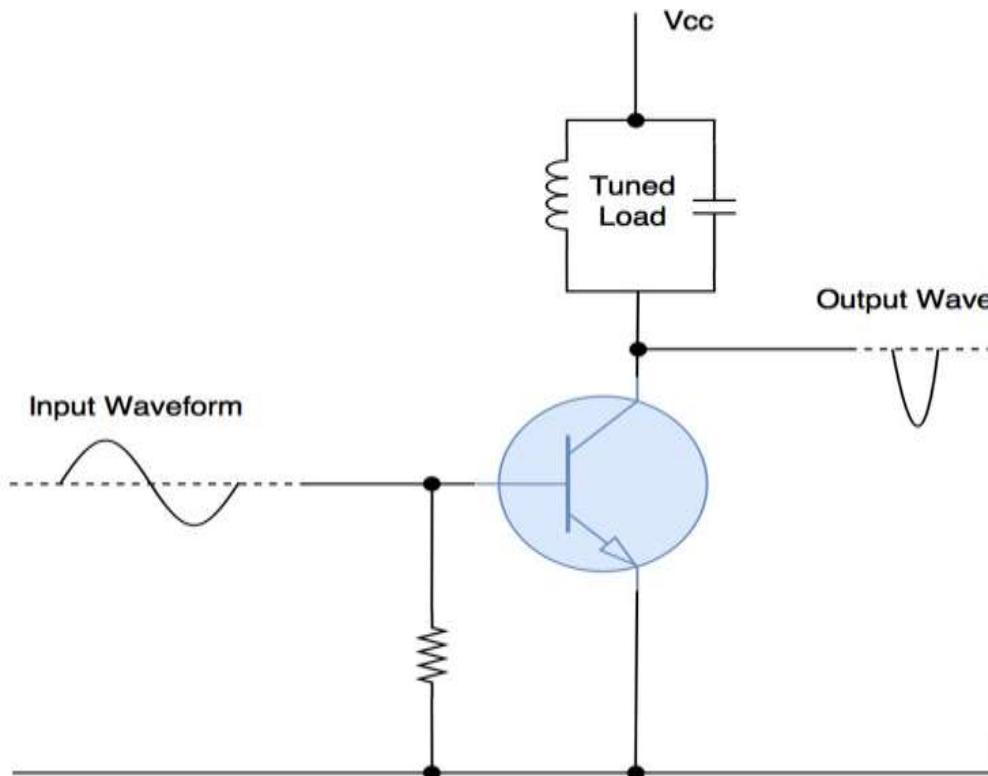
Efficiency is slightly less when compared to Class B configuration.

There will be some DC components in the output as the load is directly coupled.

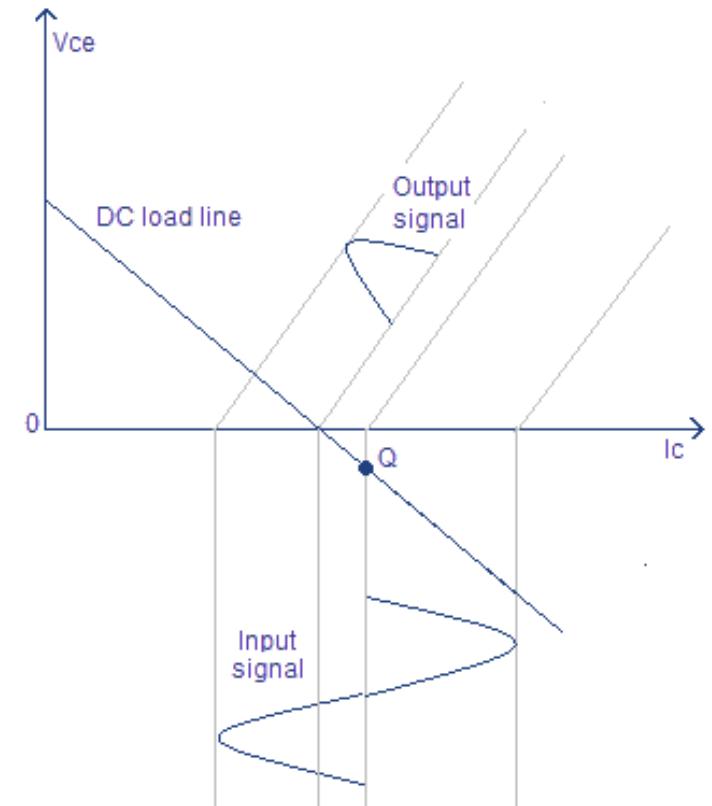
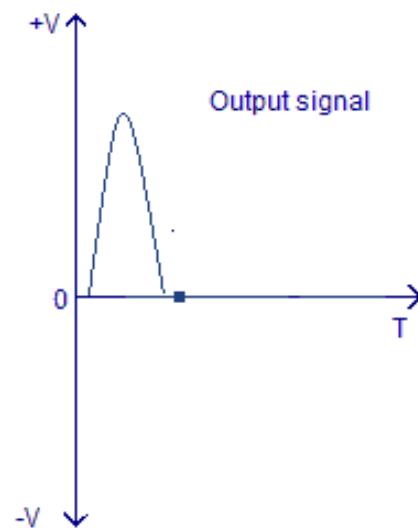
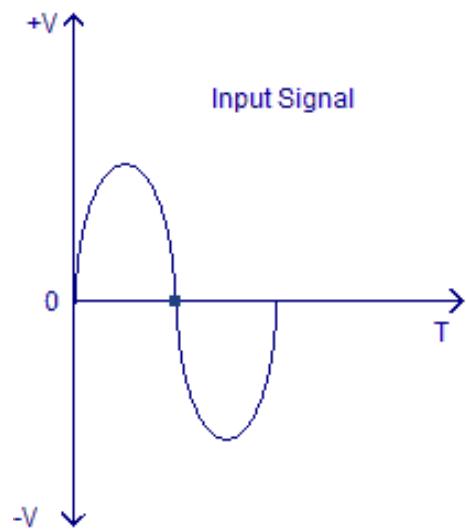
Capacitive coupling can eliminate DC components but it is not practical in case of heavy loads.

# CLASS C POWER AMPLIFIER

- Class C power amplifier is a type of amplifier where the active element (transistor) conduct for less than one half cycle of the input signal.
- Less than one half cycle means the conduction angle is less than  $180^\circ$  and its typical value is  $80^\circ$  to  $120^\circ$ . The reduced conduction angle improves the efficiency to a great extend but causes a lot of distortion.
- Theoretical maximum efficiency of a Class C amplifier is around 90%.



# CLASS C POWER AMPLIFIER



**Input and output waveforms - Class C power amplifier**

**Output Characteristics of Class C Power Amplifier**

# **CLASS C POWER AMPLIFIER.**

## **Advantages**

- High efficiency.
- Excellent in RF applications.
- Lowest physical size for a given power output.

## **Disadvantages**

- Lowest linearity.
- Not suitable in audio applications.
- Creates a lot of RF interference.
- It is difficult to obtain ideal inductors and coupling transformers.
- Reduced dynamic range.

## **Applications**

- RF oscillators.
- RF amplifier.
- FM transmitters.
- Booster amplifiers.
- High frequency repeaters.
- Tuned amplifiers etc.

# *Operational Amplifiers*

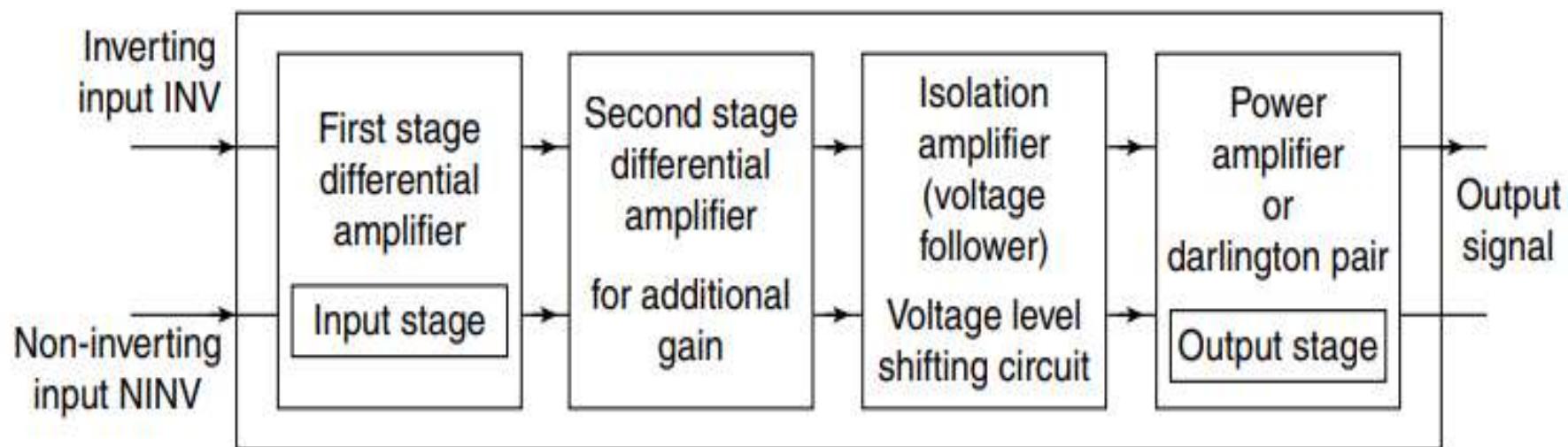
- Operational Amplifiers: Ideal v/s practical Op-amp
- Performance Parameters
- Op-Amp Applications
  - Peak detector circuit ,
  - Comparator,
  - Inverting and
  - Non-Inverting Amplifiers

- **Operational Amplifier, or op-amp,** is a very *high gain differential amplifier* with *high input impedance and low output impedance*.
- **Uses:**
  - provide voltage amplitude changes (amplitude and polarity),
  - oscillators,
  - filter circuits, and
  - many types of instrumentation circuits
- It contains a *number of differential amplifier stages to achieve a very high voltage gain*

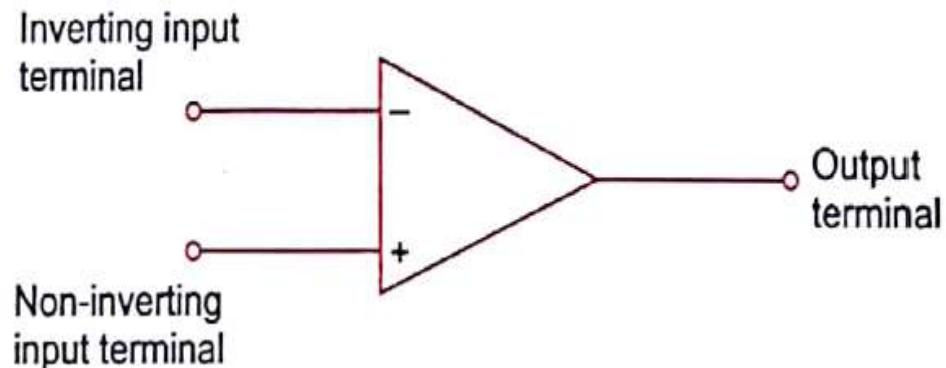
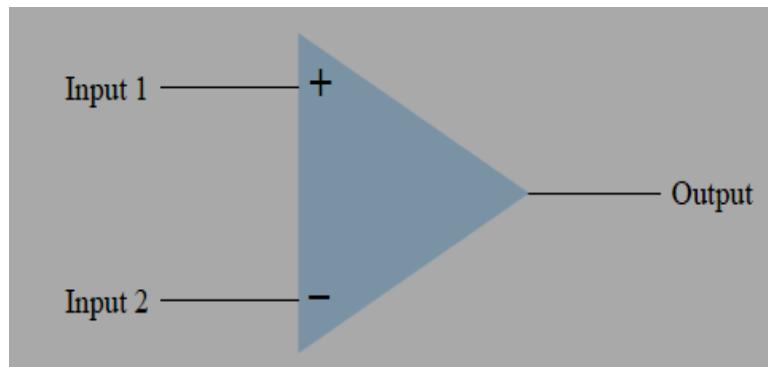
# Block diagram of an Operational Amplifier

Basic building blocks inside an op amp IC are as follows:

- First-stage differential amplifier.
- Second-stage differential amplifier.
- Voltage-level shifter.
- Power output stage.



# Circuit Symbol



An *Operational Amplifier* is basically a three-terminal device which consists of two high impedance inputs.

One of the inputs is called the **Inverting Input**, marked with a negative or “minus” sign, ( - ).

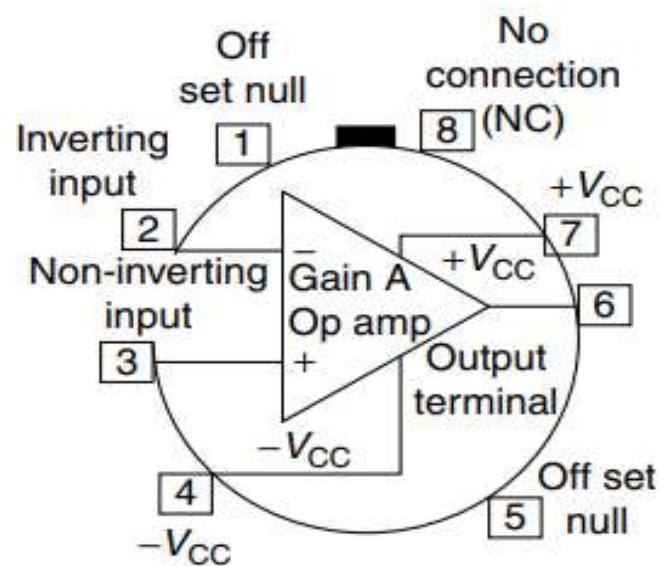
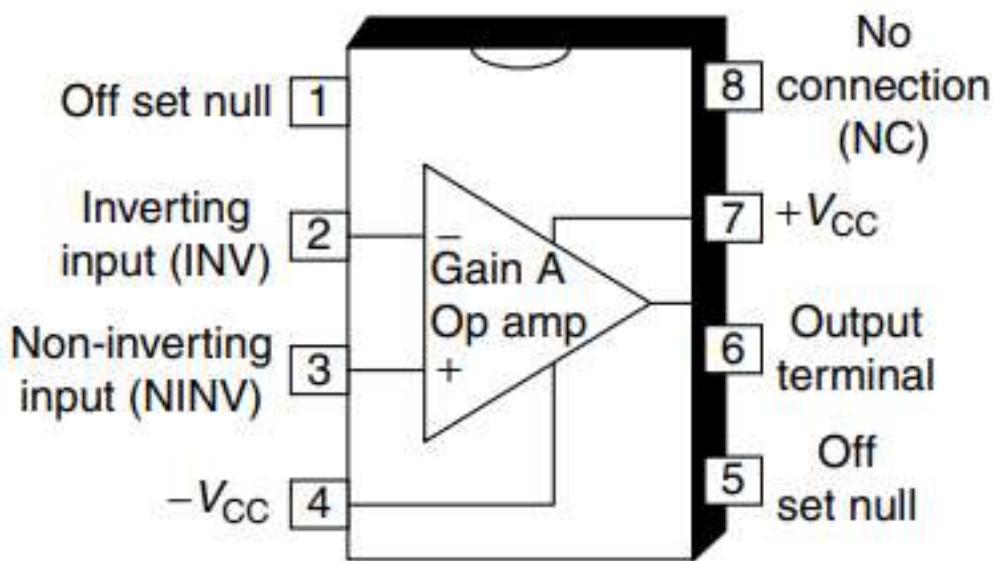
The other input is called the **Non-inverting Input**, marked with a positive or “plus” sign ( + ).

# PIN configuration of Operational Amplifier

## Packages

There are three popular packages available:

1. The metal can (TO) package
2. The dual-in-line package (DIP)
3. The flat package or flat pack



Pin-1: Offset null terminal.

Pin-2: Inverting (INV) input terminal. All input signals connected to this terminal on op amp will appear at the output terminal with  $180^\circ$  phase shift.

Pin-3: Non-inverting (N-INV) input terminal. All input signals applied to this terminal on op amp appears as it is in waveform (shape) at the output terminal.

Changes occur only in levels of amplitudes at the output voltage.

Pin-4: Supply voltage - VCC (negative supply voltage terminal).

Pin-5: Offset null terminal.

Pin-6: Output terminal; output signal.

1. Output signal will have  $180^\circ$  phase shift to input signals applied to inverting input terminal pin-2.

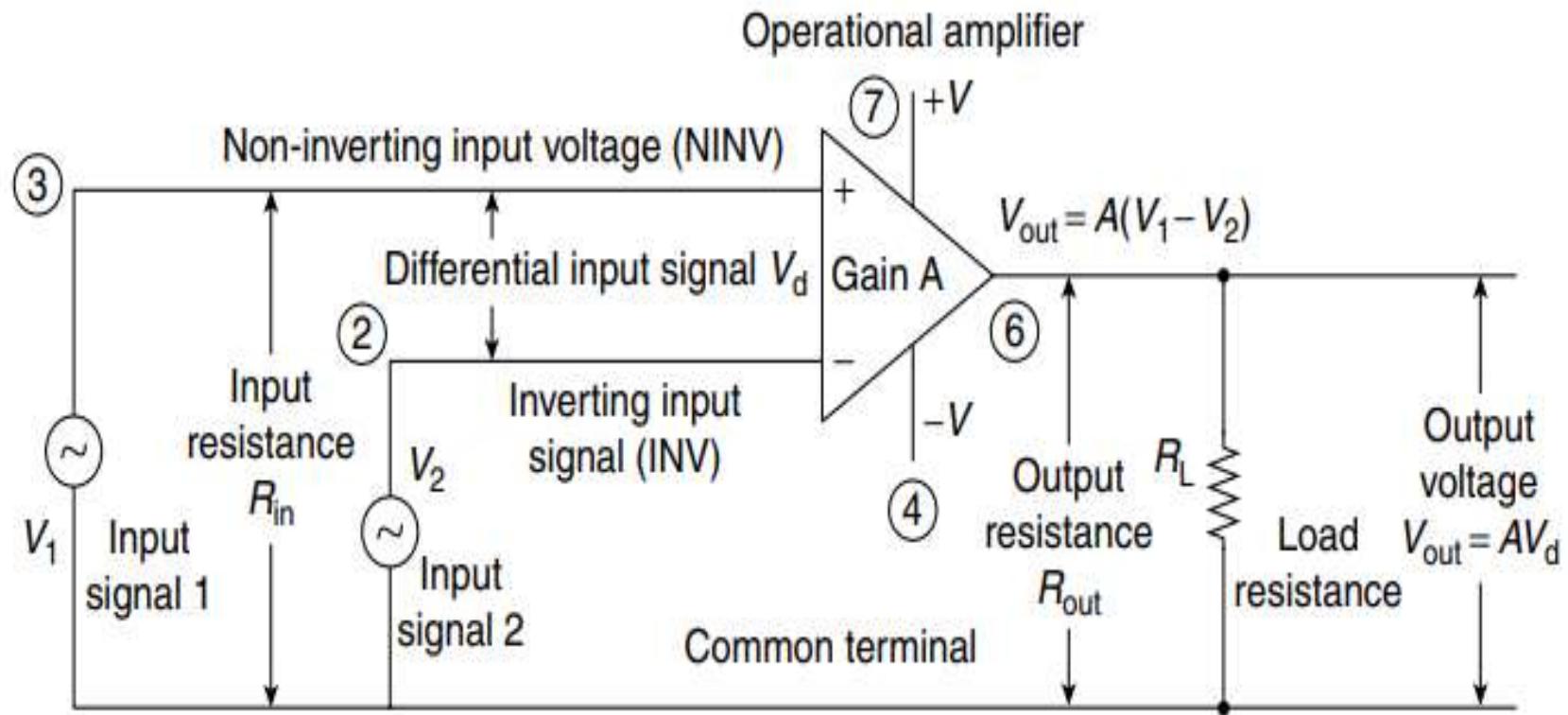
2. Output signal will be in-phase to input signals applied to non-inverting input terminal pin-3.

3. Output signal depends upon the voltage gain of operational amplifier.

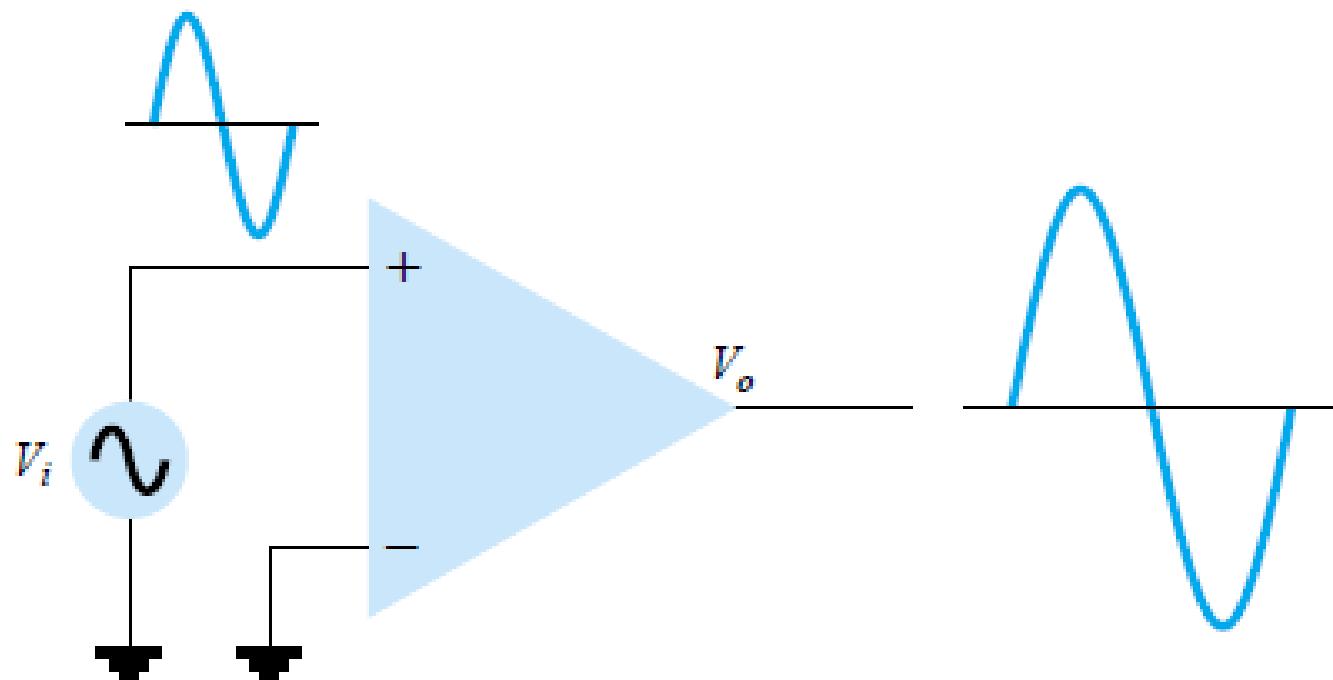
Pin-7: supply voltage +VCC terminal (positive supply voltage terminal).

Pin-8: no connection.

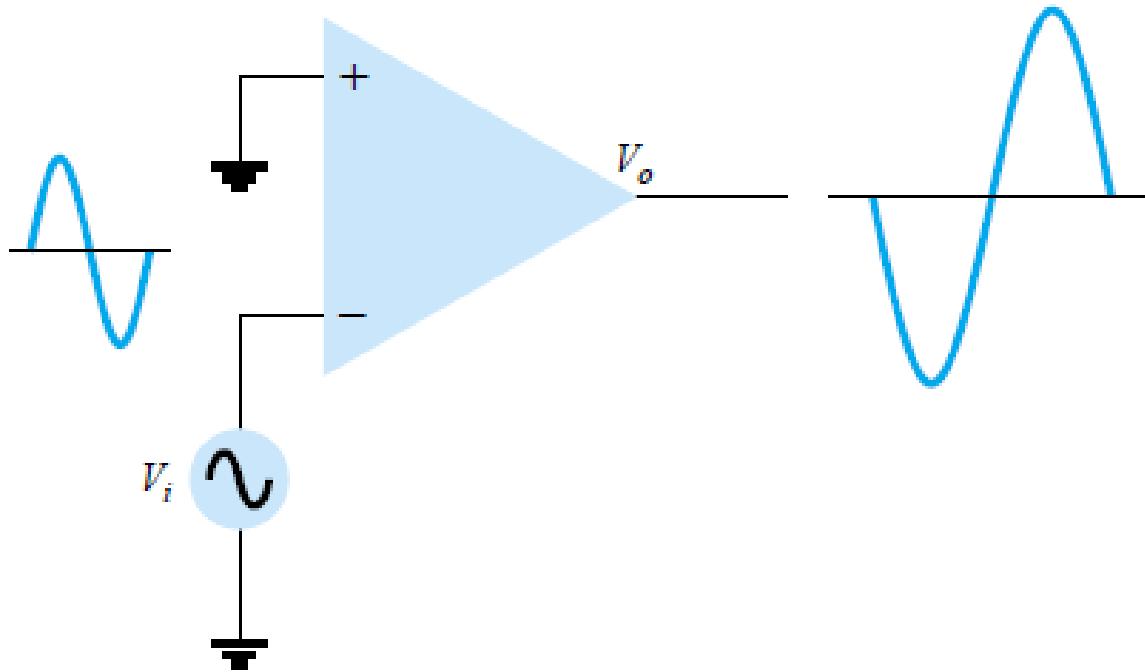
## SCHEMATIC DIAGRAM OF OPERATIONAL AMPLIFIER



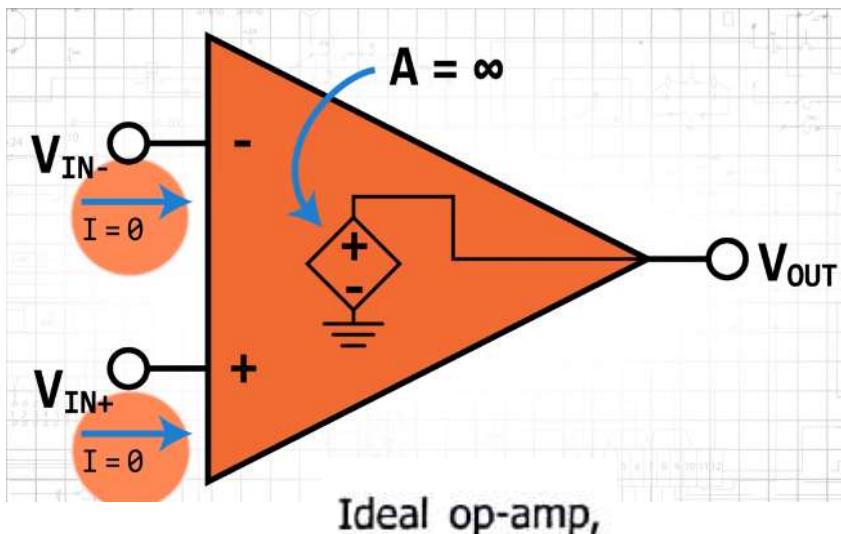
# Non-inverting Input



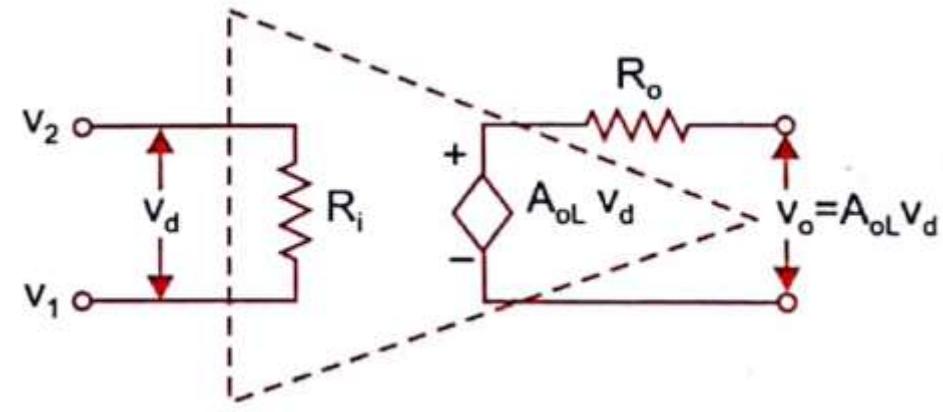
# Inverting Input



# THE IDEAL OPERATIONAL AMPLIFIER



Ideal op-amp,



Equivalent circuit of an op-amp

- The input resistance of an op amp must be very high whereas the output resistance should be quite low.
- An op amp should also have very high open loop gain.
- In Ideal Cases, the input resistance and open loop gain of an op amp should be infinity whereas the output resistance would be zero.

Characteristic	Value
Open Loop Gain (A)	$\infty$
Input Resistance (Impedance)	$\infty$
Output Resistance (Impedance)	0
Bandwidth of Operation	$\infty$
Offset Voltage	0

$$V_o = A(V_1 - V_2)$$

It can be observed that

- (i) An ideal op-amp allows zero current to enter into its input terminals, i.e.  $i_1 = i_2 = 0$ .

Due to infinite input impedance, any signal with source impedance can drive the op-amp without getting inflicted with any loading effect.

(ii) The gain of the ideal op-amp is infinite.

Hence, the voltage between the inverting and non-inverting terminals is essentially zero for a finite output voltage.

(iii) The output voltage  $V_o$  is independent of the output current drawn from the op-amp, since  $R_o = 0$ .

This means that the output can drive an infinite number of output devices of any impedance value.

# Op-Amp Parameters

- Common-mode Rejection (CMRR)
- Common-mode input voltage
- Input offset voltage
- Input bias current
- Input impedance
- Input offset current
- Output impedance
- Slew rate

# Op-Amp Parameters

- **Common-Mode Rejection Ratio (CMRR)**

- The ability of amplifier to reject the common-mode signals (unwanted signals) while amplifying the differential signal (desired signal)
- Ratio of open-loop gain,  $A_{ol}$  to common-mode gain,  $A_{cm}$
- The open-loop gain is a datasheet value

$$CMRR = \frac{A_{ol}}{A_{cm}}$$

$$CMRR = 20 \log \left( \frac{A_{ol}}{A_{cm}} \right)$$

- The higher the CMRR, the better, in which the open-loop gain is high and common-mode gain is low.
- CMRR is usually expressed in dB & decreases with frequency

# Op-Amp Parameters

- **Common-Mode Input Voltage**

- The range of input voltages which, when applied to both inputs, will not cause clipping or other output distortion.

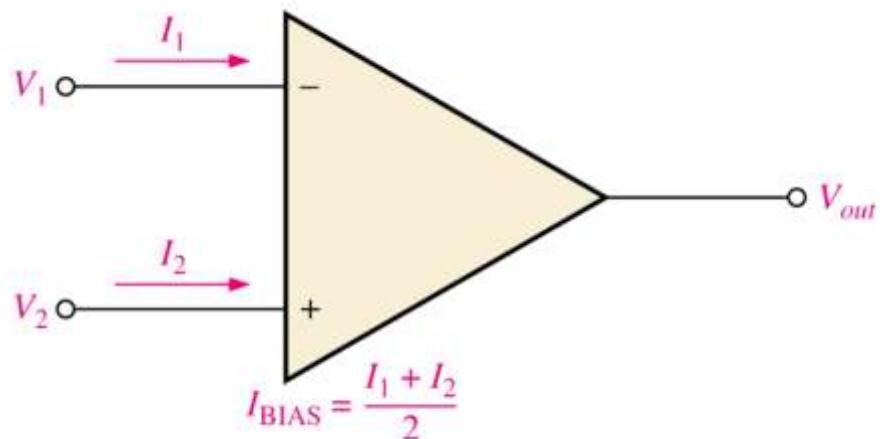
- **Input Offset Voltage**

- Ideally, output of an op-amp is 0 Volt if the input is 0 Volt.
- Realistically, a small dc voltage will appear at the output when no input voltage is applied.
- Thus, differential dc voltage is required between the inputs to force the output to zero volts.
- This is called the Input Offset Voltage,  $V_{os}$ . Range between 2 mV or less.

# Op-Amp Parameters

- **Input Bias Current**

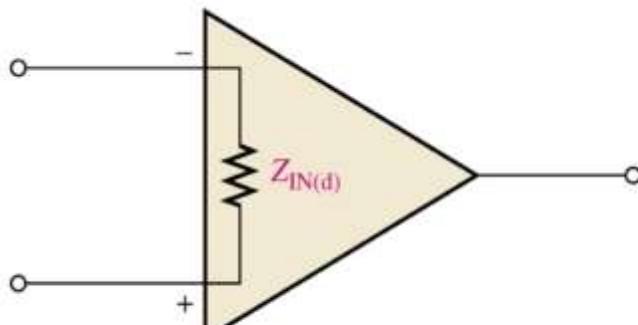
- Ideally should be zero
- The dc current required by the inputs of the amplifier to properly operate the first stage.
- Is the average of both input currents



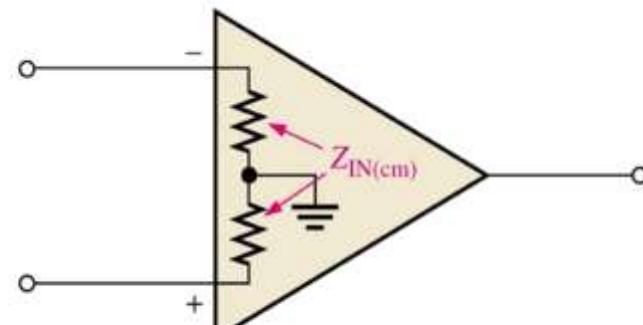
# Op-Amp Parameters

- **Input Impedance**

- Is the total resistance between the inverting and non-inverting inputs.
- Differential input impedance : total resistance between the inverting and non-inverting inputs
- Common-mode input impedance: total resistance between each input and ground



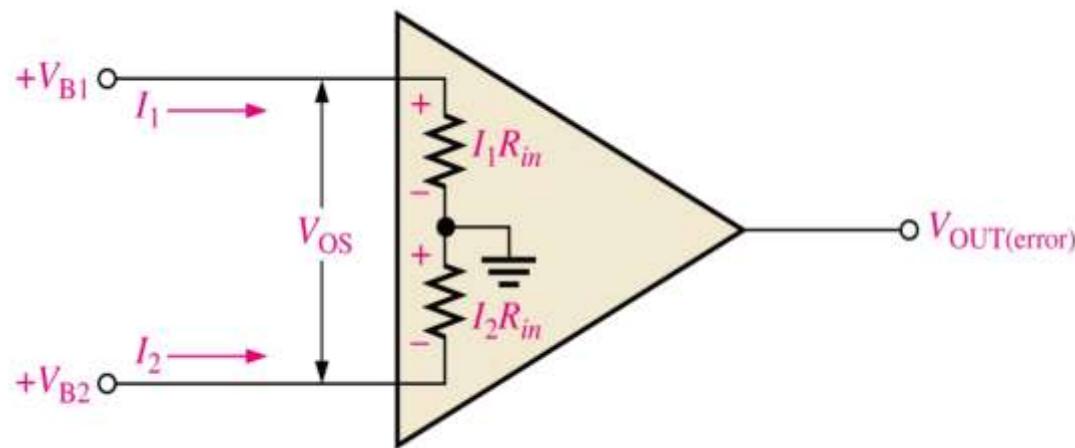
(a) Differential input impedance



(b) Common-mode input impedance

# Op-Amp Parameters

- **Input Offset Current**
  - Is the difference of input bias currents



Input offset current

$$I_{os} = |I_1 - I_2|$$

Thus, error

$$V_{out(error)} = A_v I_{os} R_{in}$$

Offset voltage

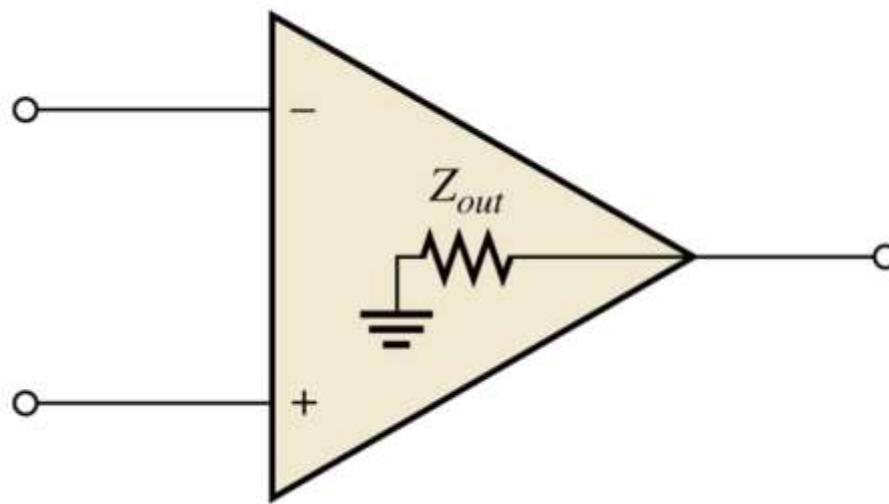
$$V_{os} = I_1 R_{in} - I_2 R_{in} = (I_1 - I_2) R_{in}$$

$$V_{os} = I_{os} R_{in}$$

# Op-Amp Parameters

- **Output Impedance**

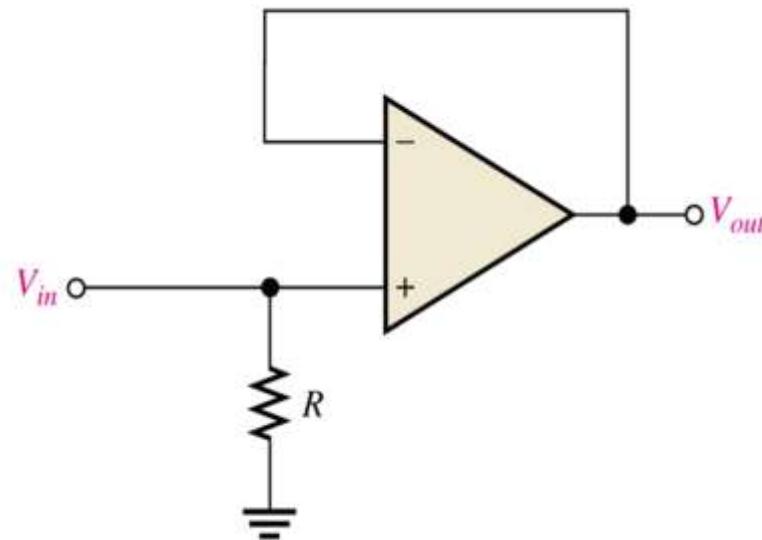
- Ideally should be zero
- Is the resistance viewed from the output terminal of the op-amp.  
In reality, it is non-zero.



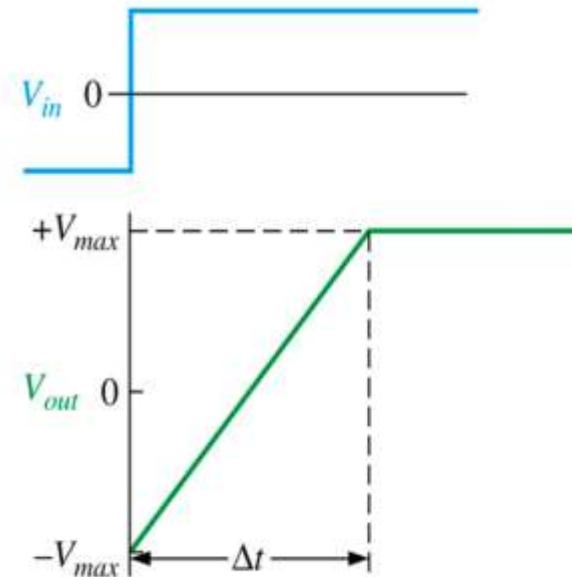
# Op-Amp Parameters

- **Slew Rate**

- Is the maximum rate of change of the output voltage in response to a step input voltage.



(a) Test circuit



(b) Step input voltage and the resulting output voltage

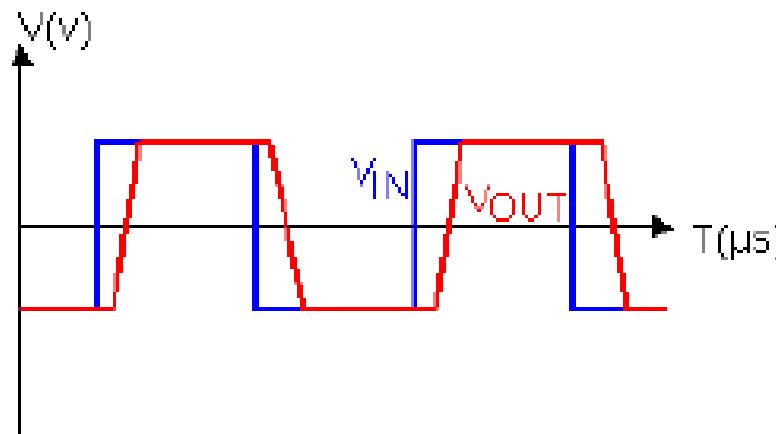
$$SlewRate = \frac{\Delta V_{out}}{\Delta t}$$

where  $\Delta V_{out} = +V_{max} - (-V_{max})$

# Op-Amp Parameters

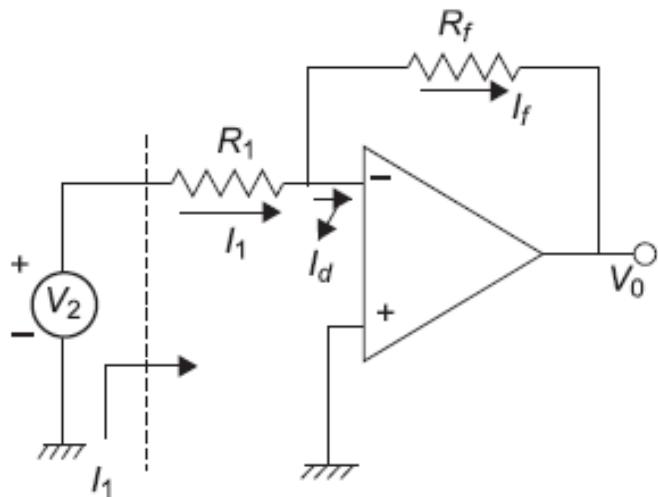
- Slew Rate

- It's a measure of how fast the output can “follow” the input signal.

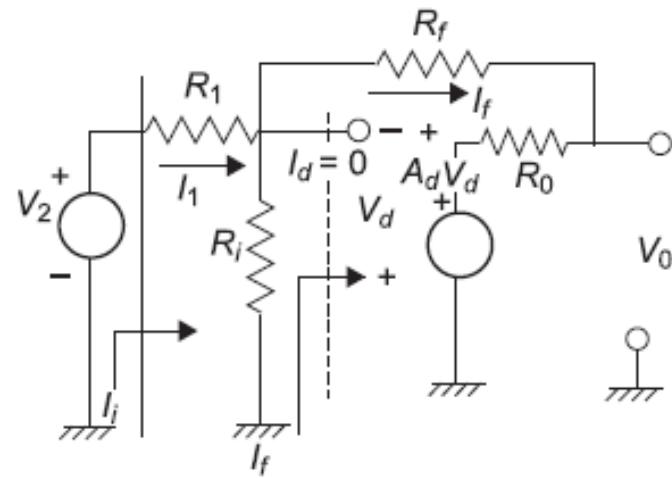


# Linear Applications of Operational Amplifier

## INVERTING AMPLIFIER



(a) Inverting Amplifier Circuit



(b) Equivalent Circuit

## Ideal Case

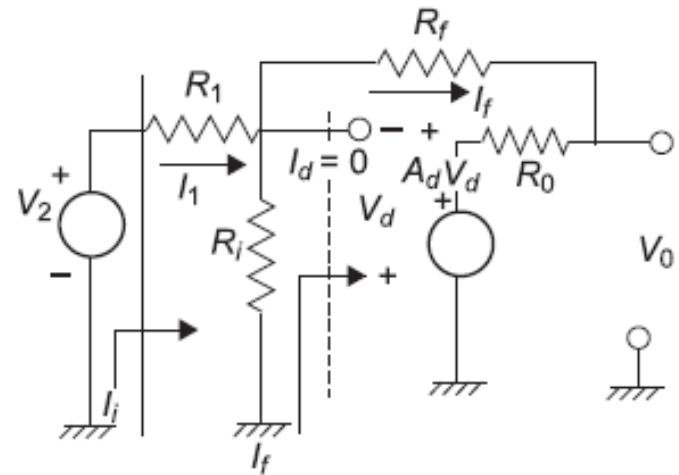
Since for ideal amp  $I_d = 0$

$$I_1 = I_f, \quad I_1 = \frac{V_2 + V_d}{R_1} = I_f = \frac{-V_d - V_o}{R_f}$$

Since  $V_d \rightarrow 0$

$$\frac{V_2}{R_1} = -\frac{V_o}{R_f}; \quad A_v = \frac{V_o}{V_2} = \frac{-R_f}{R_1}$$

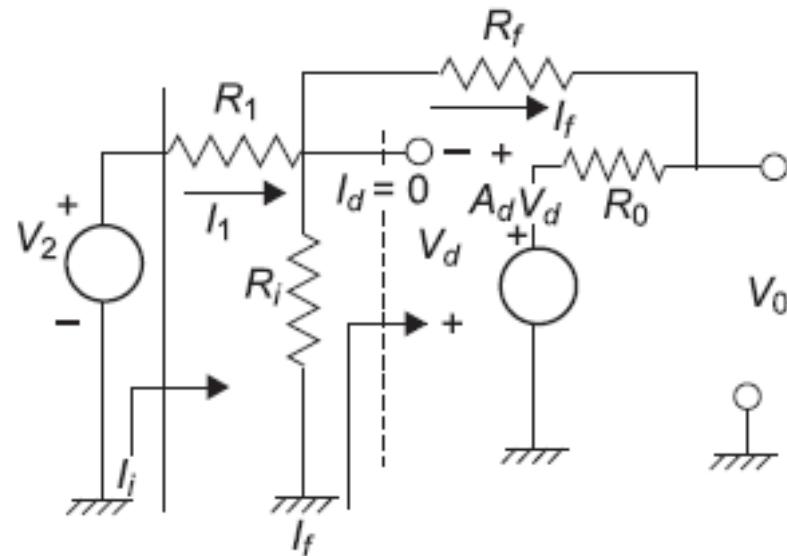
$$A_v = \frac{V_o}{V_2}; \quad \therefore \quad \frac{V_o}{V_2} = -\frac{R_f}{R_1} \quad \boxed{A_v = -\frac{R_f}{R_1}}$$



(b) Equivalent Circuit

## Input Impedance

$$r_i = \frac{V_2}{I_1} = \frac{I_1 R_1 - V_d}{I_1} \text{ since } V_d = 0 \quad r_i \approx R_i$$



(b) Equivalent Circuit

## Output Impedance

$$r_o = V_o / I_o$$

$$I_o = \frac{V_o - A_d V_d}{R_o} + \frac{V_o}{R_l + R_f}$$

since  $R_i \gg R_l$  (Fig. 1.19)

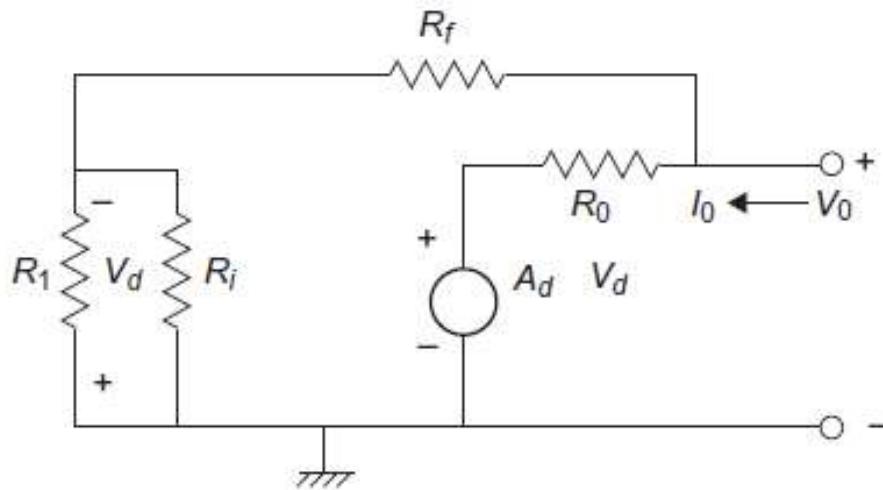


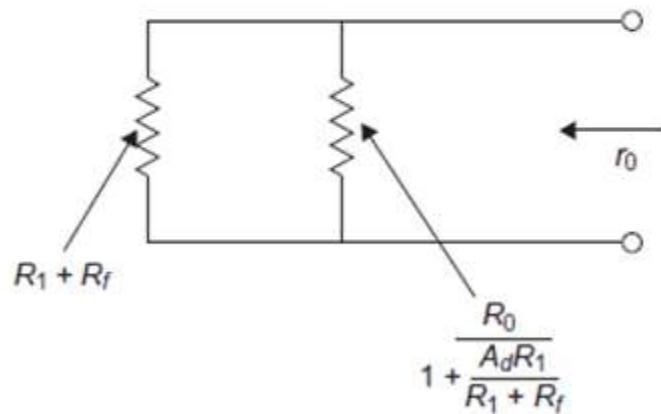
Fig. 1.19 Equivalent Circuit to Show Output Impedance

$$-V_d = \left( \frac{R_l}{R_l + R_f} \right) V_o$$

$$I_o = \left[ \frac{V_o + \left( \frac{A_d R_l}{R_l + R_f} \right) V_o}{R_o} + \frac{V_o}{R_l + R_f} \right]$$

$$\frac{I_o}{V_o} = \frac{1 + \frac{A_d R_l}{R_l + R_f}}{R_o} + \frac{1}{R_l + R_f}$$

$$r_o \approx \frac{R_o}{1 + \frac{A_d R_l}{R_l + R_f}} \text{ since } R_l + R_f \text{ is large (Fig. 1.20)}$$

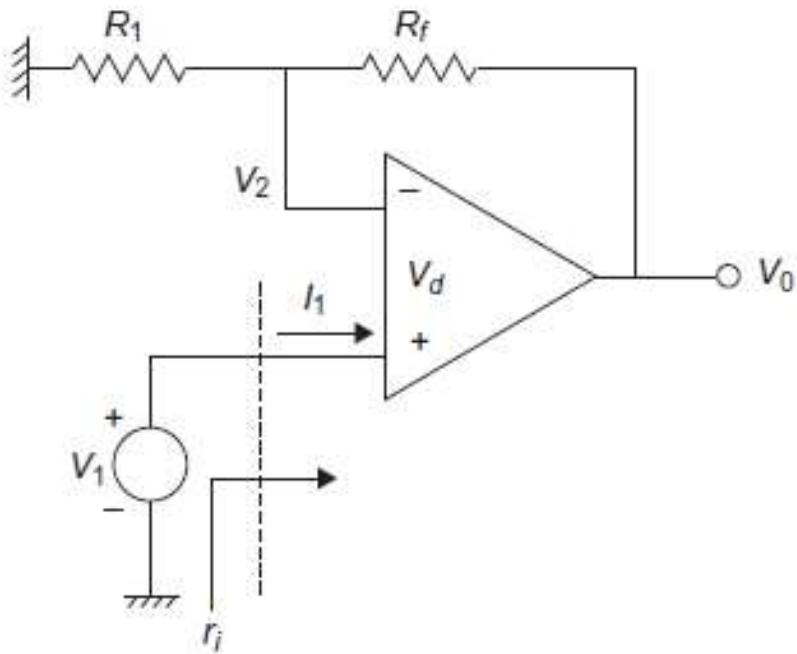


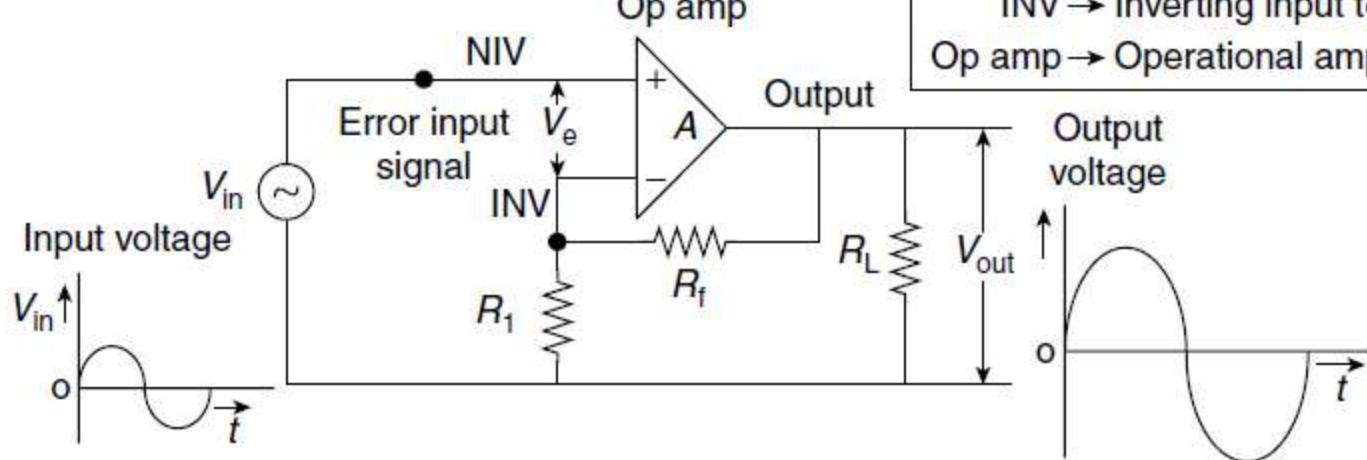
## Ideal Case: Non-inverting Amplifier

$$V_1 = V_2$$

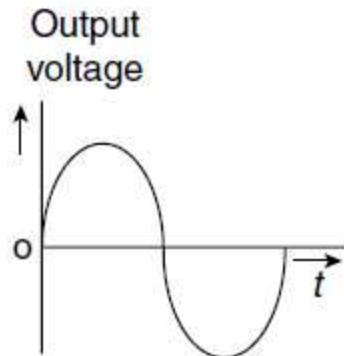
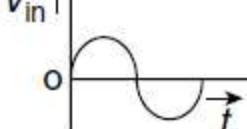
$$V_2 = \left( \frac{R_l}{R_l + R_f} \right) V_o$$

$$\frac{V_o}{V_1} = \frac{R_l + R_f}{R_l} = \left( 1 + \frac{R_f}{R_l} \right)$$

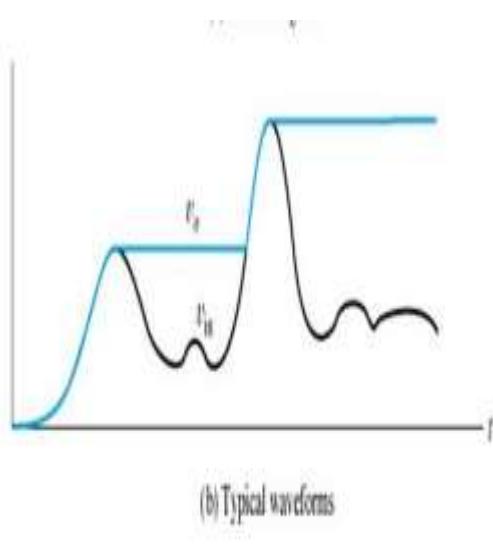
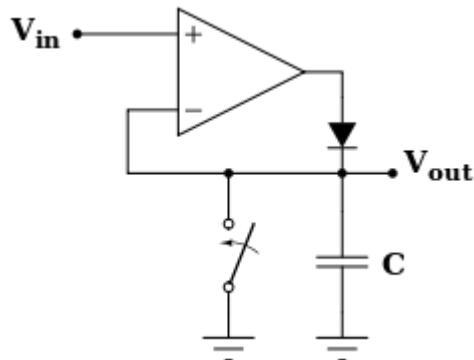




NIV → Non inverting input terminal  
 INV → Inverting input terminal  
 Op amp → Operational amplifier



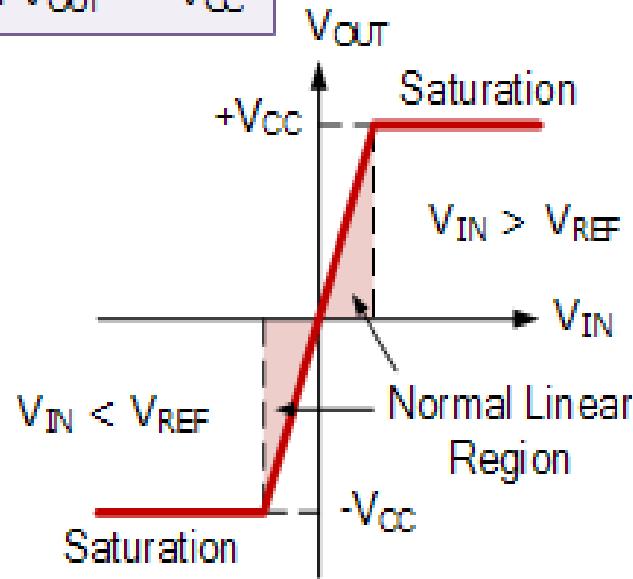
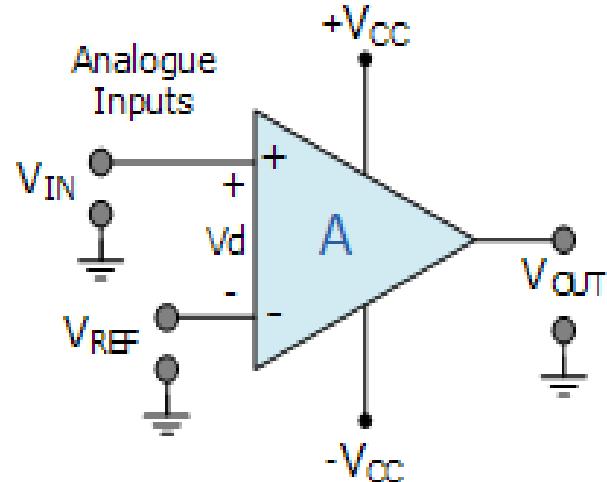
# Peak detector:



- The function of a peak detector is to compute the peak value of the input.
- The circuit follows the voltage peaks of a signal and stores the highest value on a capacitor. The highest peak value is stored until the capacitor is discharged.
- $V_{out} < V_{in}$ ; D ON and C charges to peak value of input, (i.e) voltage follower.
- $V_{out} > V_{in}$ ; D OFF and C holds the peak value of input.
- The circuit can be reset, capacitor voltage can be made zero by connecting low leakage MOSFET switch across the capacitor.
- Applications: Measurement and instrumentation, Amplitude modulation.

**Comparator:** A circuit which compares a signal voltage applied at one input of an op amp.

If  $V_{IN} > V_{REF}$  then  $V_{out} = +V_{cc}$   
If  $V_{IN} < V_{REF}$  then  $V_{out} = -V_{cc}$



# Example Problem

### Example No1

Find the closed loop gain of the given inverting amplifier circuit.

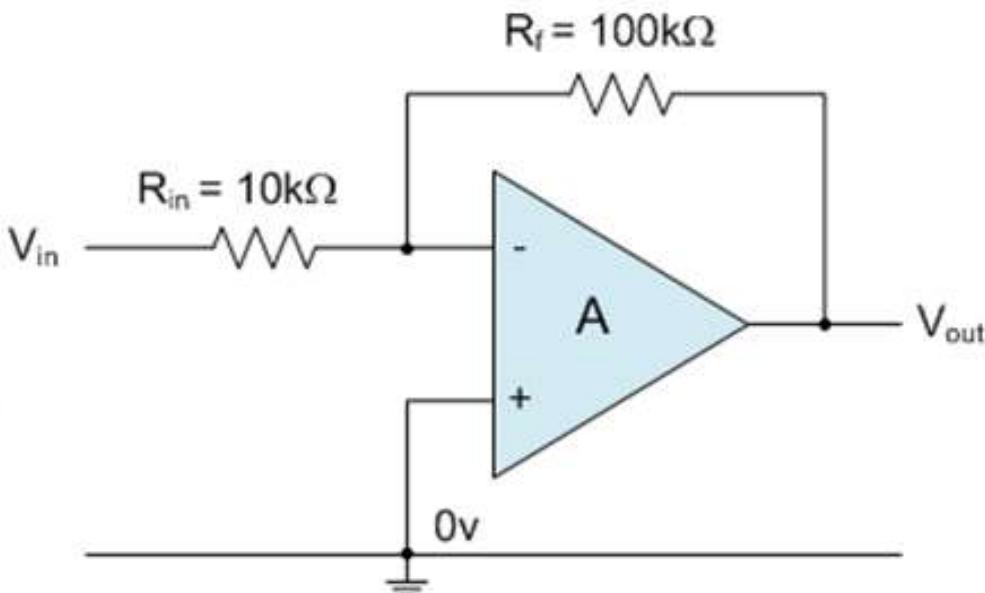
Using the previously found formula  
for the gain of the circuit

$$\text{Gain} = \frac{V_o}{V_{in}} = -\frac{R_f}{R_{in}}$$

$R_{in} = 10k\Omega$  and  $R_f = 100k\Omega$ .

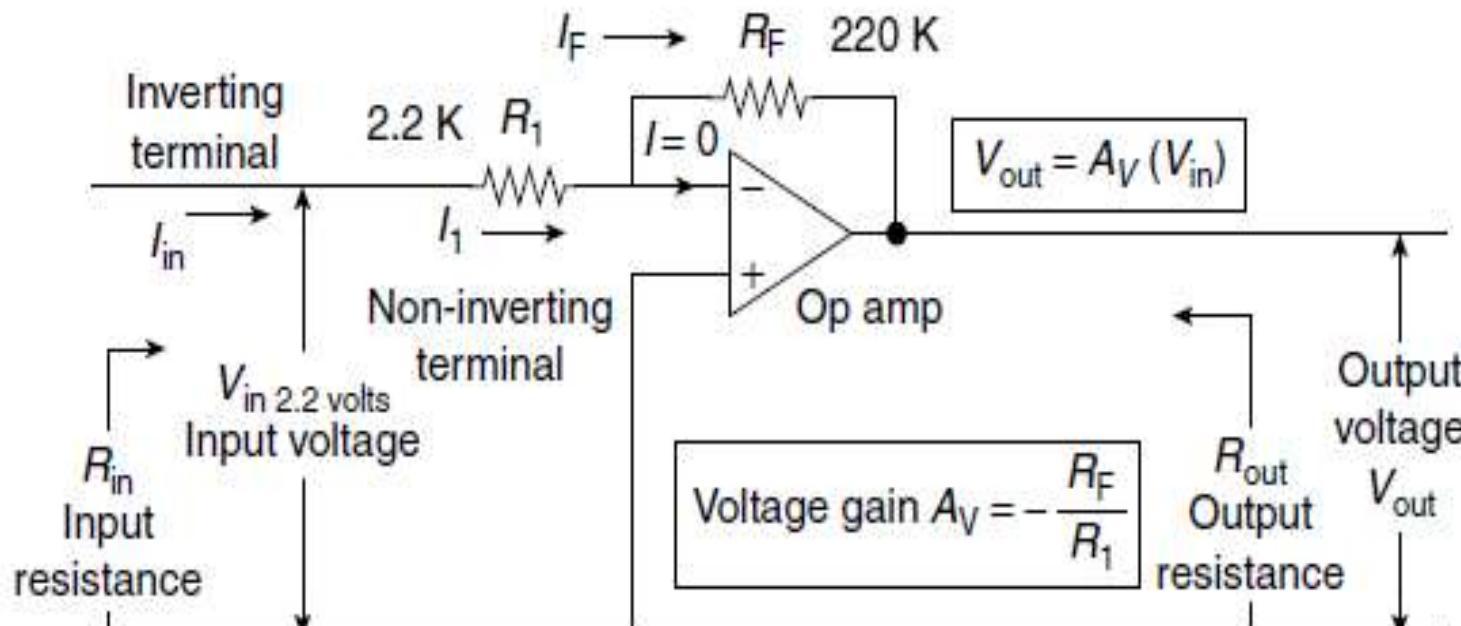
Gain =  $-R_f/R_{in} = 100k/10k = 10$ .

Therefore, the closed loop gain of the  
given inverting amplifier circuit is  
given **10 or 20dB**.



## Example

For the operational amplifier circuit shown in Fig. , calculate input current, output voltage, and voltage gain for the given data in the circuit.



Operational Amplifier with Negative Feedback

$$\text{Solution: Input current } I_1 = \frac{V_{\text{in}}}{R_1} = \frac{2.2 \text{ V}}{2.2 \text{ K}} = 1 \text{ mA}$$

Total current passing through resistor  $R_1$  will flow through feedback resistor  $R_F$ , because the current ( $I$ ) through input port of op amp is zero. Inverting input terminal is at virtual ground. Therefore, the current drawn by op amp  $I = 0 \text{ mA}$ . Hence,  $I_1 = I_F = 1 \text{ mA}$ .

Output voltage  $V_{\text{out}} = -I_F \times R_F = 1 \times 10^{-3} \times 220 \times 10^3 = 220 \text{ V}$

Output voltage  $V_{\text{out}} = -I_F \times R_F = -I_{\text{in}} \times R_F = -\frac{V_{\text{in}}}{R_{\text{in}}} \times R_F$

Therefore, voltage gain  $A_V = \frac{V_{\text{out}}}{V_{\text{in}}} = -\frac{R_F}{R_{\text{in}}} = \frac{220 \text{ K}}{2.2 \text{ K}} = 100$

## Example

Operational amplifier has differential gain of 80 dB and CMRR of 100 dB. If  $V_1 = 2 \mu\text{V}$  and  $V_2 = 1.6 \mu\text{V}$ , then calculate output voltages for differential and common-mode input signal operations.

*Solution:* Differential gain in dB =  $20 \log_{10} A_d = 80$  dB.

Therefore, anti log  $\frac{80}{20} = 10^4$ . Hence,  $A_d = 10^4$

CMRR in dB =  $20 \log_{10} \text{CMRR} = 100$  dB.

Therefore, antilog  $\frac{100}{20} = 10^5$ . Hence, CMRR =  $10^5$

Input voltage  $V_1 = 2 \mu\text{V}$  and input voltage  $V_2 = 1.6 \mu\text{V}$ .

Differential voltage  $V_d = (V_1 - V_2) = (2 - 1.6) \times 10^{-6} \text{ V} = 0.4 \times 10^{-6} \text{ V}$

Common-mode voltage  $V_{\text{Cm}} = \frac{(V_1 + V_2)}{2} = \frac{(2 + 1.6) \times 10^{-6}}{2} = 1.8 \times 10^{-6} \text{ V}$

Output voltage

$$V_{\text{out}} = A_d V_d \left[ 1 + \frac{1}{\text{CMRR}} \times \frac{V_{\text{Cm}}}{V_d} \right] = 10^4 \times 0.4 \times 10^{-6} \left[ 1 + \frac{1.8 \times 10^{-6}}{10^5 \times 0.4 \times 10^{-6}} \right] \approx 4 \text{ mV}$$

The performance of an operational amplifier depends upon the following important parameters.

## Example

---

Consider an operational amplifier with resistors  $R_1 = 1 \text{ k}\Omega$  and  $R_f = 99 \text{ k}\Omega$ . When the two input terminals are grounded so that voltage  $V_1 = 0$  and  $V_2 = 0$ . Calculate the output offset voltage  $V_{\text{out}}(0)$ , if there is an input offset voltage  $V_{\text{in}}(0) = 2 \text{ mV}$ .

*Solution:*

Voltage gain

$$A_V = \left[ 1 + \frac{R_f}{R_1} \right] = \left[ 1 + \frac{99 \text{ k}\Omega}{1 \text{ k}\Omega} \right] = 100$$

Output offset voltage

$$V_{\text{out}}(0) = A_V \times V_{\text{in}}(0) \text{ V} = 100 \times 2 \text{ mV} = 200 \text{ mV} = 0.2 \text{ V.}$$

## **Example**

---

1. Calculate the voltage gain of an inverting op amp with  $R_1 = 3.3 \text{ k}\Omega$  and resistor  $R_2 = 33 \text{ k}\Omega$ .
2. Calculate the output voltage when the input voltage = 0.5 V.
3. Calculate the total output offset voltage for input offset voltage = 0.05 V.

*Solution:*

1. Voltage gain of an inverting op amp  $A_V = \left( \frac{R_2}{R_1} \right) = \frac{33 \times 10^3}{3.3 \times 10^3} = 10$
2. Output voltage  $= V_{\text{out}} = A_V \times V_{\text{in}} = 10 \times 0.5 = 5 \text{ V}$ .
3. Output voltage due to offset voltage  $V_{\text{out}}(0) = A_V (V_{\text{in}} + V_{\text{in}}(0))$

$$V_{\text{out}}(0) = 10 (0.5 + 0.05) = 5.5 \text{ V.}$$

## Example

---

1. Calculate the voltage gain of non-inverting op amp if  $R_1 = 2.2 \text{ k}\Omega$  and resistor  $R_2 = 22 \text{ k}\Omega$
2. Calculate output voltage for input voltage of 0.2 V.
3. Calculate the total offset voltage due to input offset voltage of 0.02 V.

*Solution:*

1. Voltage gain of non-inverting op amp =  $A_V = \left(1 + \frac{R_2}{R_1}\right) = \left(1 + \frac{22 \times 10^3}{2.2 \times 10^3}\right) = 11$
2. Ideal output voltage with zero input offset voltage  $V_{\text{out}} = A_V \times V_{\text{in}} = 11 \times 0.2 = 2.2 \text{ V}$ .
3. Output voltage with finite input offset voltage  $V_{\text{out}}(0) = A_V (V_{\text{in}} + V_{\text{in}}(0))$

$$V_{\text{out}}(0) = A_V (V_{\text{in}} + V_{\text{in}}(0)) = 11 \times (0.2 + 0.02) = 11 \times 0.22 = 2.42 \text{ V.}$$

## Example :

An operational amplifier with a slew rate of  $1 \text{ V}/\mu\text{s}$  is designed to produce a voltage gain  $A_V = 5$ . If the maximum amplitude of input signal is 2 V, calculate the maximum frequency of operation of operational amplifier. Calculate  $V_{\text{out}} (\text{max})$  and  $f_{\text{max}}$  when  $V_{\text{in}} = 0.5 \text{ V}$ .

*Solution:* Input signal  $V_{\text{in}} = 2 \text{ V}$ , voltage gain  $A_V = 5$

Output voltage  $V_{\text{out}}$  has maximum amplitude  $V_{\text{out}} (\text{max}) = A_V \times V_{\text{in}} (\text{max})$ .

Therefore,  $V_{\text{out}} (\text{max}) = 5 \times 2 = 10 \text{ V}$

Slew Rate =  $1 \text{ V}/\mu\text{s}$ .

$$\text{Maximum frequency of operation } f_m = \frac{\text{SR}}{2\pi V_{\text{out}} (\text{max})} = \frac{1 \text{ V}}{10^{-6} \times 2\pi \times 10} = \frac{10^6}{20\pi} \cong 16 \text{ kHz.}$$

If the input signal is reduced to 0.5 V, then the corresponding output voltage

$$V_{\text{out}} (\text{max}) = A_V \times V_{\text{in}} (\text{max}) = 5 \times 0.5 = 2.5 \text{ V.}$$

$$\text{Maximum frequency of operation } f_m = \frac{\text{SR}}{2\pi V_{\text{out}} (\text{max})} = \frac{1 \text{ V}}{10^{-6} \times 2\pi \times 2.5} = \frac{10^6}{5\pi} \cong 63.7 \text{ kHz}$$

## Example

---

An operational amplifier with a slew rate  $SR = 0.5 \text{ V}/\mu\text{s}$  is used as an inverting amplifier to obtain a gain  $A_V$  of 100. The voltage gain versus frequency response is flat up to 10 kHz. Determine the following:

1. The maximum peak to peak input signal that can be applied without any distortion to the output.
2. The maximum frequency of the input signal to obtain a sine wave output of peak voltage of 2 V.

*Solution:*

1. Slew rate  $SR = 0.5 \text{ V}/\mu\text{s}$ .

Maximum frequency (bandwidth)  $f_{\max} = 10 \text{ kHz}$

Gain  $A_V = 100$

$$f_{\max} = \frac{\text{maximum slew rate without causing distortion in output signal}}{2\pi V_{\max}(\text{output})}$$

$$\text{Maximum output voltage } V_{\max}(\text{output}) = \frac{\text{slew rate (SR)}}{2 \times \pi \times f_{\max}}$$

$$V_{\max}(\text{output}) = \frac{0.5}{2 \times \pi \times 10 \times 10^3 \times 10^{-6}} = 7.96 \text{ V}$$

$$\text{Input voltage } V_{\text{in}} = \frac{V_{\max}(\text{output})}{A_V} = \frac{7.96}{100} = 79.6 \text{ mV}$$

2. Slew rate SR = 0.5 V/ $\mu$ s.

Peak (maximum) value of output voltage =  $V_{\max}$  (output) = 2 V.

Maximum frequency (Bandwidth) is calculated as in the following:

$$f_{\max} = \frac{\text{maximum slew rate without causing distortion in output signal}}{2\pi V_{\max}(\text{output})}$$

$$f_{\max} = \frac{0.5}{2\pi \times 2 \times 10^{-6}} = \frac{100 \times 10^3}{4\pi} = 7.96 \text{ kHz}$$

### **Example**

---

An op amp has a slew rate of 2 V/ $\mu$ s. What is the maximum frequency of an output sinusoid of peak value 5 V at which the distortion sets in due to the slew rate limitation?

*Solution:* Slew rate SR = 2 V/ $\mu$ s.

Peak (maximum) value of output voltage =  $V_{\max}$  (output) = 5 V.

Maximum frequency (bandwidth)  $f_{\max}$  is calculated as in the following:

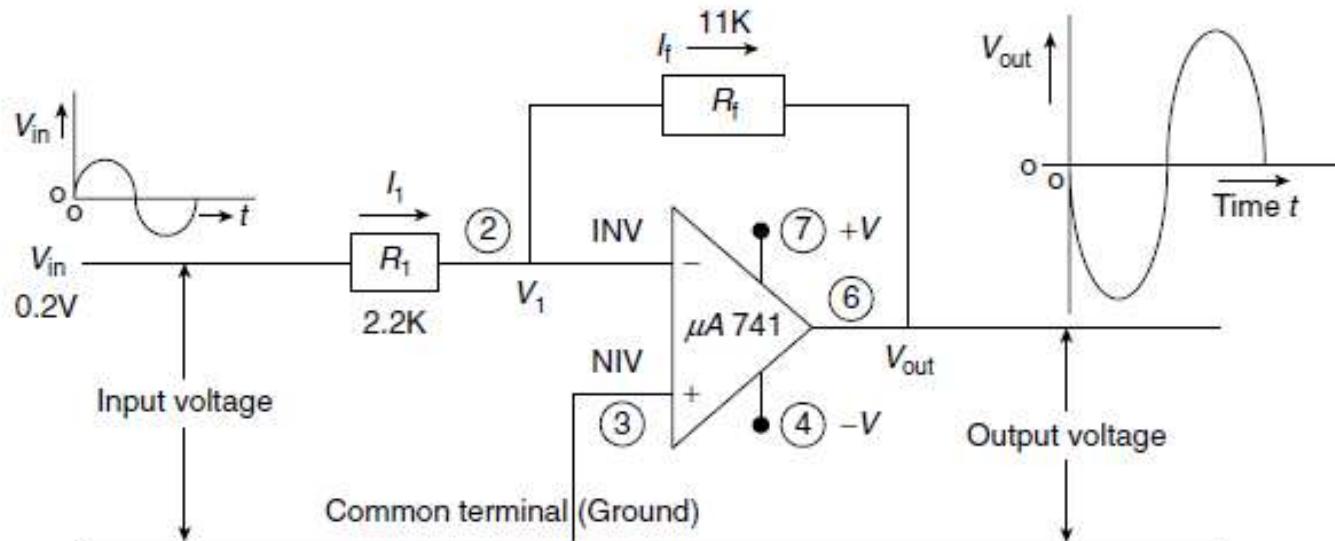
$$f_{\max} = \frac{\text{maximum slew rate without causing distortion in output signal}}{2\pi V_{\max}(\text{output})}$$

$$f_{\max} = \frac{2}{2\pi \times 5 \times 10^{-6}} = \frac{100 \times 10^3}{5\pi} = 6.365 \text{ kHz}$$

## Example

Calculate the following:

1. Voltage gain  $A_V$  of operational amplifier in the following circuit with input resistor  $R_1 = 2.2 \text{ k}\Omega$  and feedback  $R_f = 11 \text{ k}\Omega$ .
2. Output voltage when input signal is 0.2 V.



Sign Changer (Inverting Voltage Amplifier)

Solution:

$$1. \text{ Voltage gain } A_V = -\frac{R_f}{R_1} = \frac{11\text{k}\Omega}{2.2\text{k}\Omega} = 5$$

Negative sign indicates that the output voltage is inverted ( $180^\circ$  phase shift to input signal)

$$2. \text{ Output voltage } V_{out} = A_V \times V_{in} = 5 \times 0.2 \text{ V} = 1 \text{ V.}$$

## Example

An inverting amplifier circuit is shown Fig. External circuit components are  $R_1 = 2.5 \text{ k}\Omega$ , feedback resistor  $R_f = 25 \text{ k}\Omega$ , and input signal voltage  $V_1 = 0.5 \text{ V}$ . Calculate the magnitudes of the following:

1. Input current  $I_1$
2. Voltage gain  $A_V$
3. Output voltage  $V_{\text{out}}$
4. Load current  $I_L$  in the amplifier circuit, when the load resistance  $R_L = 10 \text{ k}\Omega$
5. Input resistance  $Z_{\text{in}}$  (INV) when  $Z_{\text{in}} (\text{INV}) = R_1$
6. Output resistance when  $Z_{\text{out}} (\text{INV}) = 100 \Omega$ .

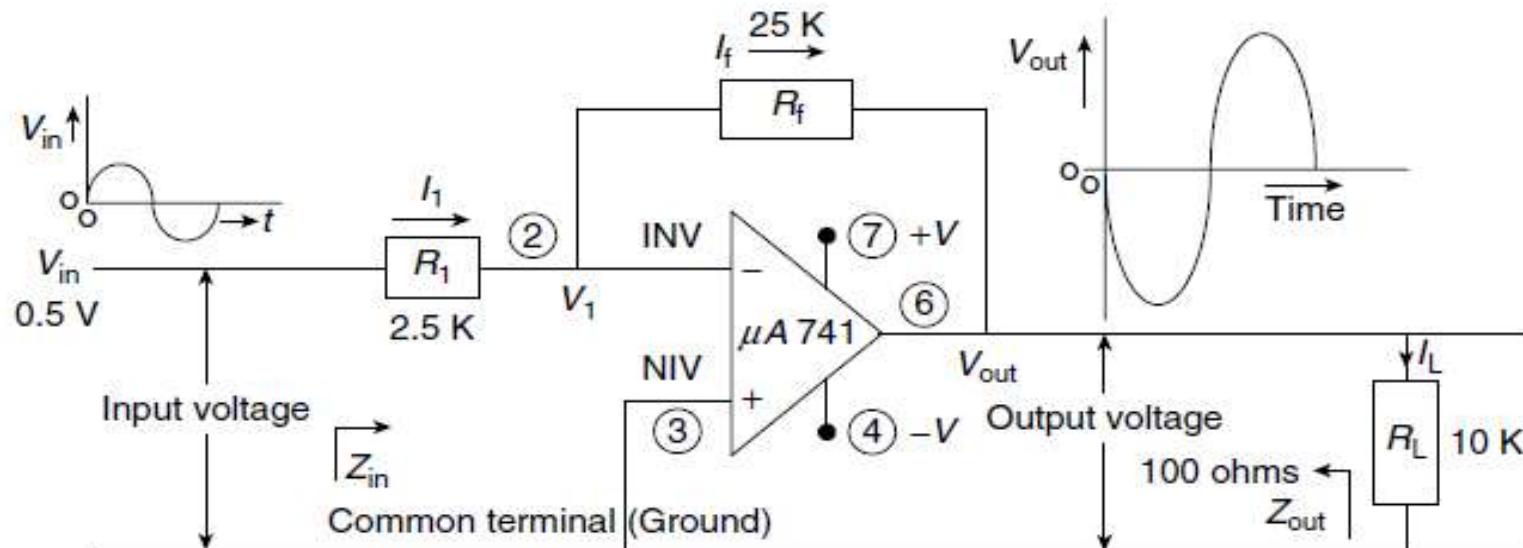


Fig. Sign Changer (Inverting Voltage Amplifier)

*Solution:*

1. Input current  $I_1 = \frac{V_1}{R_1} = \frac{0.5 \text{ V}}{2.5 \text{ k}\Omega} = 0.2 \text{ mA}$

2. Voltage gain  $A_V = -\frac{R_f}{R_1} = \frac{25 \text{ k}\Omega}{2.5 \text{ k}\Omega} = -10.$

3. Output voltage  $V_{\text{out}} = A_V \cdot V_1 = -10 \times 0.5 \text{ V} = 5 \text{ V}$

4. Load current  $I_L = \frac{V_{\text{out}}}{R_L} = \frac{5 \text{ V}}{10 \text{ k}\Omega} = 0.5 \text{ mA}$

5. Input resistance of inverting amplifier  $Z_{\text{in}} (\text{INV}) = R_1 = 2.5 \times 10^3 \Omega.$

6. Feedback factor  $\beta = \frac{R_f}{[R_1 + R_f]} = \frac{2.2 \times 10^3}{[2.2 + 25] \times 10^3} = 0.1$

7. Output resistance of inverting amplifier

$$Z_{\text{out}} (\text{INV}) = \frac{Z_{\text{out}}}{[1 + A_V \times \beta]} = \frac{100}{[1 + 10 \times 0.1]} = 50 \Omega$$

## Example

Component values of non-inverting amplifier circuit in Fig. are  $R_1 = 2.2 \text{ k}\Omega$ ,  $R_f = 22 \text{ k}\Omega$ , load resistance  $R_L = 2.5 \text{ k}\Omega$ , and input voltage  $V_{in} = 0.5 \text{ V}$ .

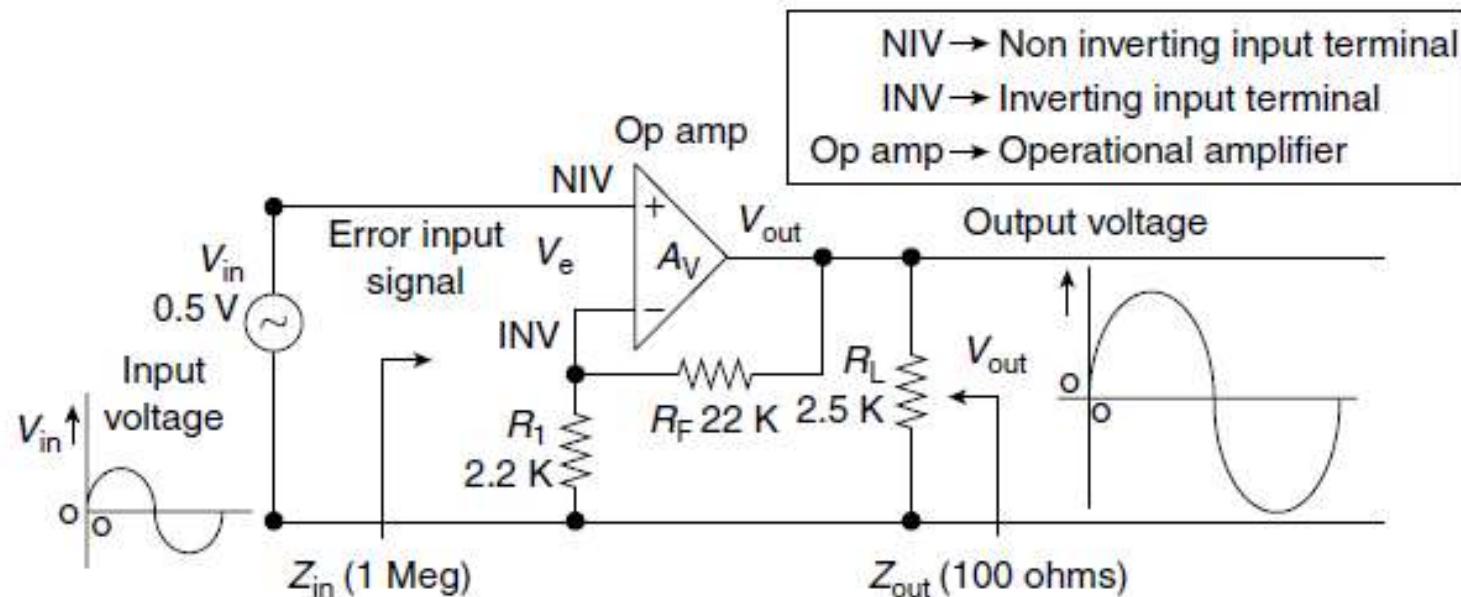


Fig. Non-inverting Voltage Amplifier Using Op Amp

Calculate (a) voltage gain (b) output voltage (c) load current  $I_L$  (d)  $\beta$  (e) input resistance  $Z_{in}$  (NIV) when the input resistance  $Z_{in}$  of normal operational amplifier is  $1 \text{ m}\Omega$ , and (f) output impedance  $Z_{out}$  (NIV) when the output resistance  $Z_{out}$  is  $100 \Omega$ .

*Solution:*

$$1. \text{ Voltage gain } A_V = \left[ 1 + \frac{R_f}{R_l} \right] = \left[ 1 + \frac{22 \times 10^3}{2.2 \times 10^3} \right] = 11$$

$$2. \text{ Output voltage } V_{\text{out}} = A_V \times V_{\text{in}} = 11 \times 0.5 \text{ V} = 5.5 \text{ V}$$

$$3. \text{ Load current } I_L = \frac{V_{\text{out}}}{R_L} = \frac{5.5 \text{ V}}{2.2 \times 10^3} = 2.2 \text{ mA}$$

$$4. \beta = \frac{R_l}{[R_l + R_f]} = \frac{2.2 \times 10^3}{[2.2 + 22] \times 10^3} \cong 0.1$$

$$5. Z_{\text{in}} (\text{NIV}) = Z_{\text{in}} \times [1 + A_V \times \beta] = 1 \times 10^6 [1 + 11 \times 0.1] = 2.1 \times 10^6 \Omega$$

$$6. Z_{\text{out}} (\text{NIV}) = \frac{Z_{\text{out}}}{[1 + A_V \times \beta]} = \frac{100}{[1 + 11 \times 0.1]} = \frac{10}{2.1} \cong 5 \Omega$$

## Example

Component values of non-inverting amplifier circuit in Fig. are  $R_1 = 2.0 \text{ k}\Omega$ ,  $R_f = 18 \text{ k}\Omega$ , load resistance  $R_L = 2.5 \text{ k}\Omega$ , and input voltage  $V_{in} = 0.5 \text{ V}$ .

Calculate (a) voltage gain, (b) output voltage, (c) load current  $I_L$ , (d)  $\beta$ , (e) input resistance  $Z_{in}$  (NIV) when the input resistance  $Z_{in}$  of normal Operational Amplifier is  $1\text{m}\Omega$ , and (f) output impedance  $Z_{out}$  (NIV) when the output resistance  $Z_{out}$  is  $100 \Omega$ .

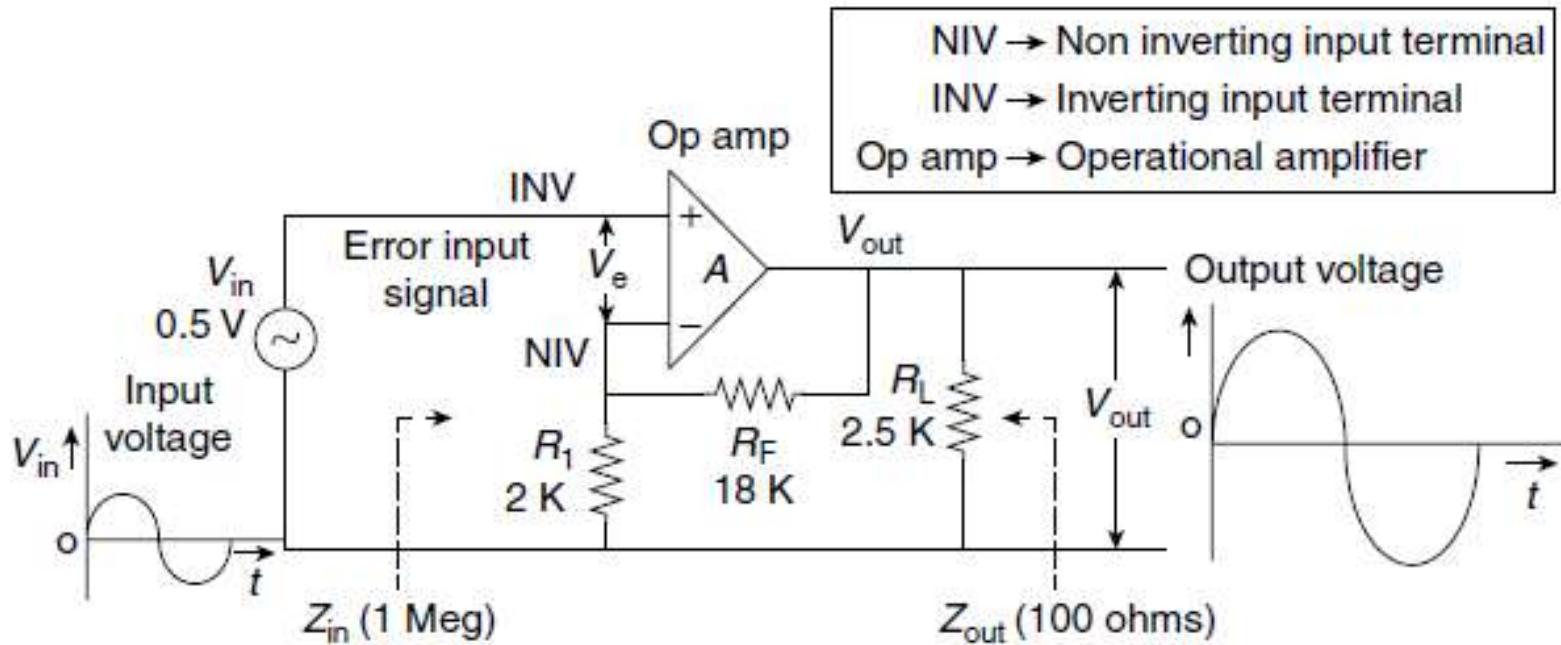


Fig. Non-inverting Voltage Amplifier Using Op Amp

*Solution:*

$$1. \text{ Voltage gain } A_V = \left[ 1 + \frac{R_f}{R_i} \right] = \left[ 1 + \frac{18 \times 10^3}{2.0 \times 10^3} \right] = 10$$

$$2. \text{ Output voltage } V_{\text{out}} = A_V \times V_{\text{in}} = 10 \times 0.5 \text{ V} = 5.0 \text{ V}$$

$$3. \text{ Load current } I_L = \frac{V_{\text{out}}}{R_L} = \frac{5.0 \text{ V}}{2.5 \times 10^3} = 2.0 \text{ mA}$$

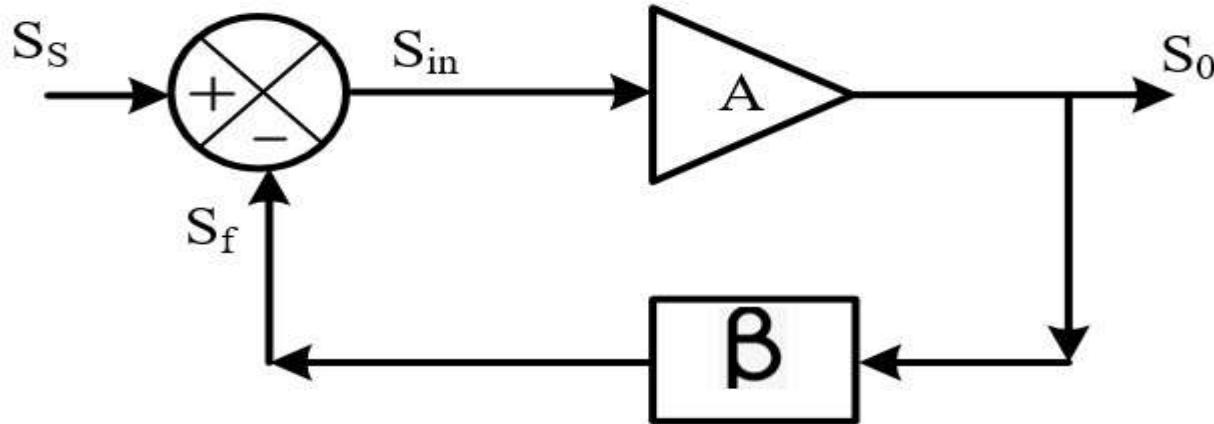
$$4. \beta = \frac{R_i}{[R_i + R_f]} = \frac{2.0 \times 10^3}{[2.0 + 18] \times 10^3} = 0.1$$

$$5. Z_{\text{in}} (\text{NIV}) = Z_{\text{in}} \times [1 + A_V \times \beta] = 1 \times 10^6 [1 + 10 \times 0.1] = 2 \times 10^6 \Omega$$

$$6. Z_{\text{out}} (\text{NIV}) = \frac{Z_{\text{out}}}{[1 + A_V \times \beta]} = \frac{100}{[1 + 10 \times 0.1]} = \frac{10}{2} \cong 5 \Omega$$

# FEED BACK SYSTEM

Basic model of feedback amplifier consist of

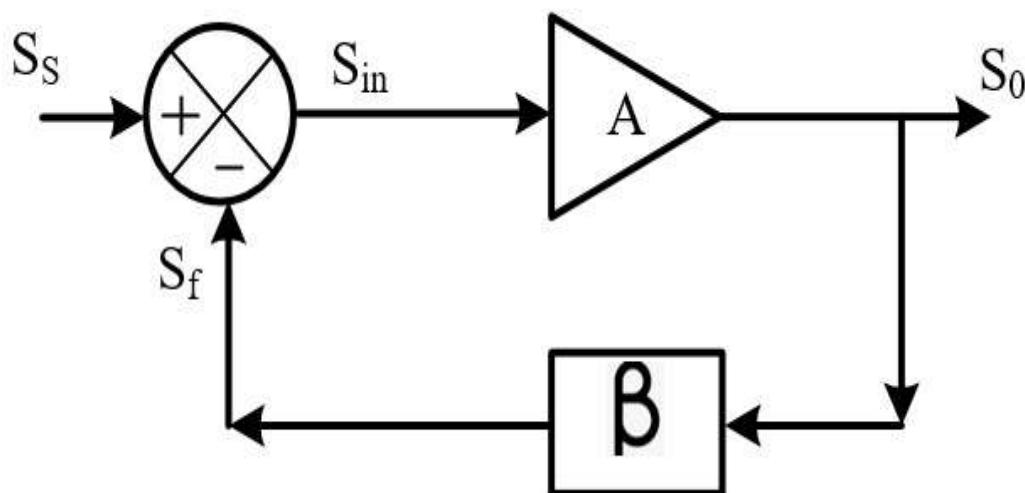


- **Forward amplifier:** The input signal to the forward amplifier is  $S_{in}$  and the output signal from the amplifier is  $S_o$ . The gain of the amplifier is  $A$ .
- **Feedback network:** The input signal to the feedback network is  $S_o$  and the output signal from the feedback network is the feedback signal  $S_f$ . The gain of the feedback network is  $\beta$ .
- **Signal sampler:** Signal sampler is taping the output signal for feedback.
- **Signal mixer:** Signal mixer is mixing the feedback signal  $S_f$  with the primary input signal  $S_s$  to generate the input signal to the amplifier  $S_{in}$ . The mixer is multiplying the input signals with +1 or -1 and then mix it.

Types of Feedback system

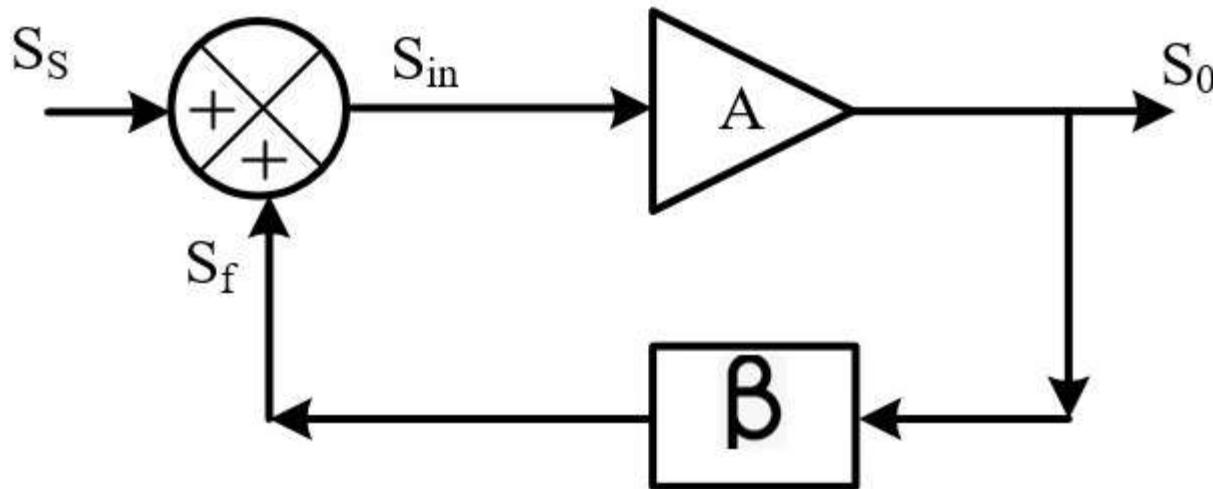
- 1) Negative feedback system
- 2) Positive feedback system

# Negative Feedback System



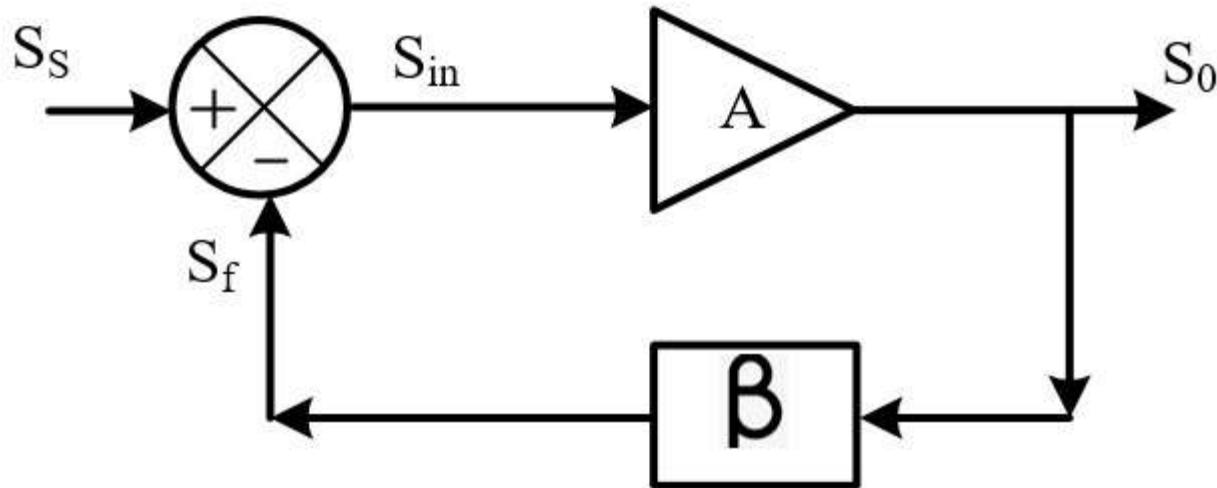
- For a change at a point in the system/ circuit if the created effect coming back to the point through the feedback path negate the original change then the feedback system is called negative feedback system.
- Assume  $S_{in}$  is increased. If  $A$  is positive  $S_o$  also increases. If  $\beta$  is positive  $S_f$  also increases. When the signal is going through the mixer, if it is negative the signal coming back to the input of the amplifier is in the opposite direction [negating the original change]. Hence it is called as the negative feedback system.
- Based on the sign of mixer,  $A$  and  $\beta$  we will get negative feedback.

# Positive Feedback System



- For a change at a point in the system/circuit if the created effect coming back to the point through the feedback path aggravate the original change then the feedback system is called as positive feedback system.
- Assume  $S_{in}$  is increased. If  $A$  is positive  $S_0$  also increases. If  $\beta$  is positive  $S_f$  also increases. When the signal is going through the mixer, if it is positive the signal coming back to the input of the amplifier is in the same direction or phase [aggravate the original change]. This is known as positive feedback system.
- Based on the sign of mixer,  $A$  and  $\beta$  we will get positive feedback system.

# Parameters of negative Feedback system



$$\text{Gain with feedback } A_f = \frac{S_o}{S_s}$$

$$S_o = AS_{in} = A(S_s - S_f) = A(S_s - \beta S_o) = AS_s - A\beta S_o$$

$$S_o(1 + A\beta) = AS_s$$

$$\frac{S_o}{S_s} = \frac{A}{1 + \beta A} = A_f$$

Loop gain =  $-A\beta$

Without feedback gain is  $A$

With feedback gain is reduced by  $(1 + \beta A)$

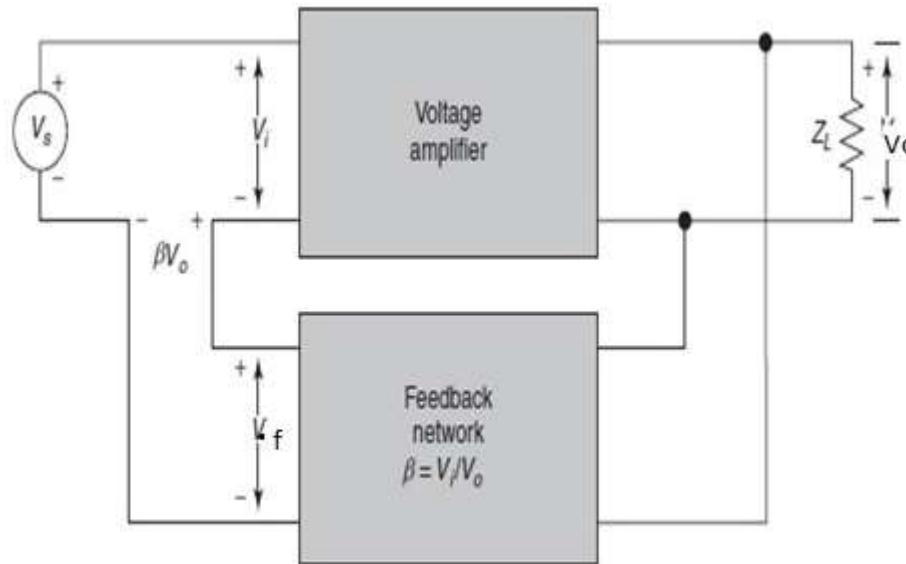
$(1 + \beta A)$  is known as the Desensitivity factor.

# Feedback Connection Types

- ❖ Feedback can be connected in four basic ways.
- ❖ Both voltage and current can be feed back to the input either in series or parallel.
  - 1) Voltage – series feedback.
  - 2) Voltage – shunt feedback.
  - 3) Current – series feedback.
  - 4) Current – shunt feedback.
  - Series feedback connections increase the input resistance.
  - Shunt feedback connections decrease the input resistance.
  - Voltage feedback decrease the output impedance.
  - Current feedback increase the output impedance.

# 1) Voltage-series feedback:

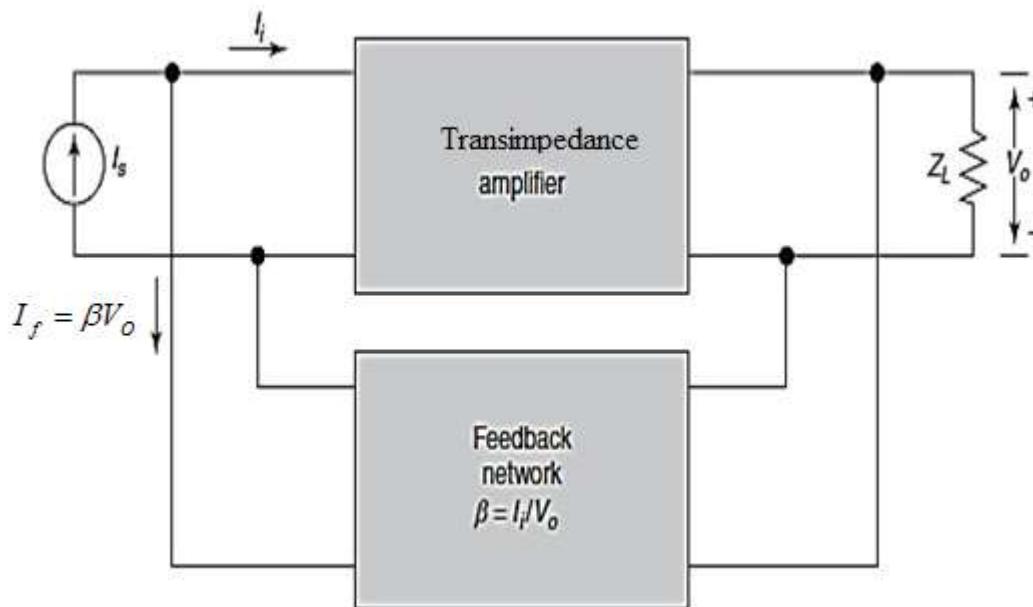
- The input signal is voltage and the output signal is voltage. Voltage is sampled at the output and *feedback in the form of voltage at the input*. Forward amplifier converts voltage to voltage. So the gain of the amplifier is voltage gain  $A_v$ . The feedback path converts voltage to voltage and its gain is  $\beta$ .
- The voltage gain with feedback is  $A_{vf} = \frac{A_v}{1 + \beta A_v}$
- The voltage gain with feedback reduces the gain  $A_v$  by the factor  $(1 + \beta A_v)$ . When  $\beta A_v \gg 1$ ,  $A_{vf} \approx \frac{1}{\beta}$
- The voltage gain is stabilized to  $A_{vf} \approx \frac{1}{\beta}$



Voltage amplifiers with voltage-series feedback

## 2) Voltage-Shunt Feedback

- The input signal is current and the output signal is voltage. Voltage is sampled at the output and feedback in the form of current at the input. Forward amplifier converts current to voltage. So the gain of the amplifier is trans impedance  $Z_m$ . The feedback path converts voltage to current and its gain is  $\beta$ .
- The trans impedance with feedback is  $Z_{mf} = \frac{Z_m}{1 + \beta Z_m}$
- The gain with feedback reduces the gain  $Z_m$  by the factor  $(1 + \beta Z_m)$ . When  $\beta Z_m \gg 1$ ,  $Z_{mf} \approx \frac{1}{\beta}$
- The trans impedance with feedback is stabilized to  $Z_{mf} \approx \frac{1}{\beta}$



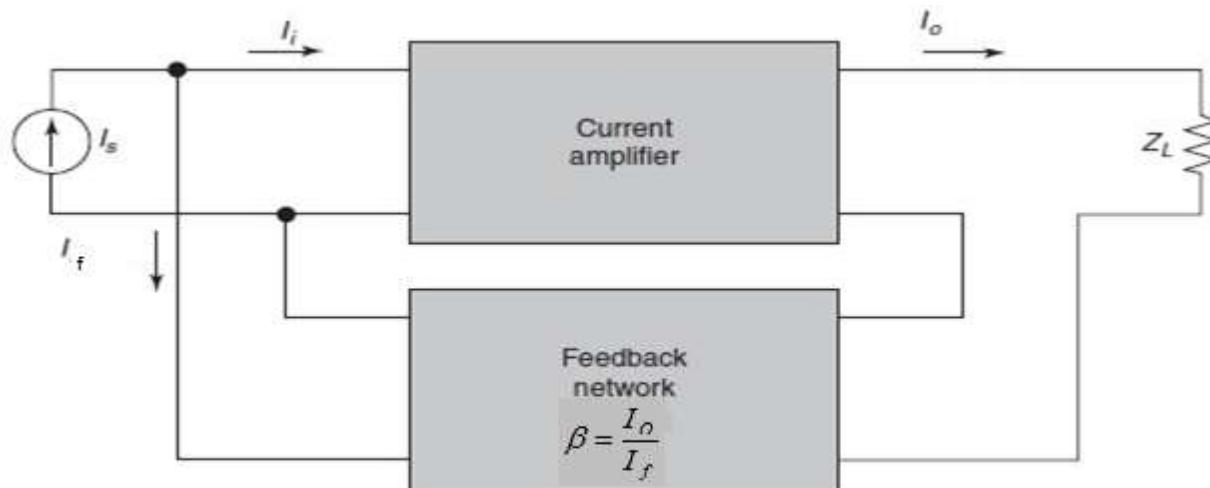
### 3) Current-shunt feedback

- The input signal is current and the output signal is current. Current is sampled at the output and *feedback in the form of current at the input. Forward amplifier converts current to current. So the gain of the amplifier is current gain  $A_i$ . The feedback path converts current to current and its gain is  $\beta$ .*
- The current gain with feedback
- The current gain with feedback reduces the gain  $A_v$  by the factor  $(1+\beta A_v)$ . When  $\beta \gg 1$ ,*
- The current gain with feedback is stabilized to*

$$A_{if} = \frac{A_i}{1 + \beta A_i}$$

$$\beta A_i \quad A_{if} \approx \frac{1}{\beta}$$

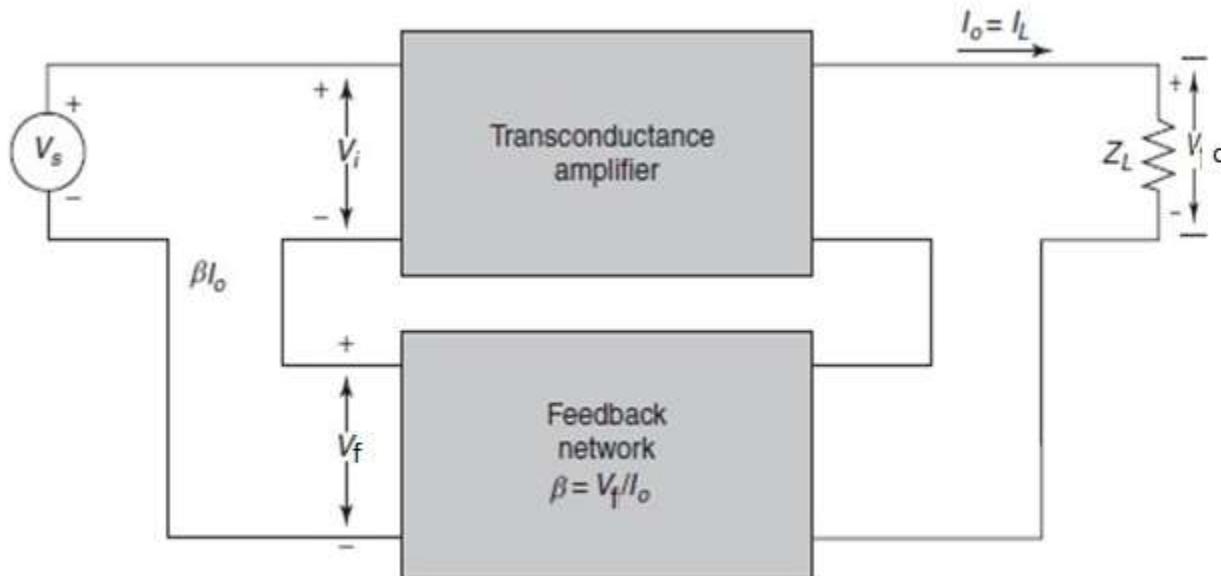
$$A_{if} \approx \frac{1}{\beta}$$



Current amplifiers with current-shunt feedback

## 4) Current-Series Feedback:

- The input signal is voltage and the output signal is current. Current is sampled at the output and *feedback in the form of voltage at the input*. *Forward amplifier converts voltage to current. So the gain of the amplifier is trans conductance. The feedback path converts current to voltage and its gain is  $\beta$ .*
- The trans conductance with feedback  $G_{mf} = \frac{G_m}{1 + \beta G_m}$*
- The gain with feedback reduces the gain by the factor  $(1 + \beta G_m)$ . When  $\beta G_m \gg 1$ ,  $G_{mf} \approx \frac{1}{\beta}$*
- The trans conductance with feedback is stabilized to  $G_{mf} \approx \frac{1}{R}$*



Transconductance amplifier with current-series feedback

# Summary

TABLE 1 Summary of Gain, Feedback, and Gain with Feedback

		Voltage-Series	Voltage-Shunt	Current-Series	Current-Shunt
Gain without feedback	$A$	$\frac{V_o}{V_i}$	$\frac{V_o}{I_i}$	$\frac{I_o}{V_i}$	$\frac{I_o}{I_i}$
Feedback	$\beta$	$\frac{V_f}{V_o}$	$\frac{I_f}{V_o}$	$\frac{V_f}{I_o}$	$\frac{I_f}{I_o}$
Gain with feedback	$A_f$	$\frac{V_o}{V_s}$	$\frac{V_o}{I_s}$	$\frac{I_o}{V_s}$	$\frac{I_o}{I_s}$

# Summary of effects of negative feedback

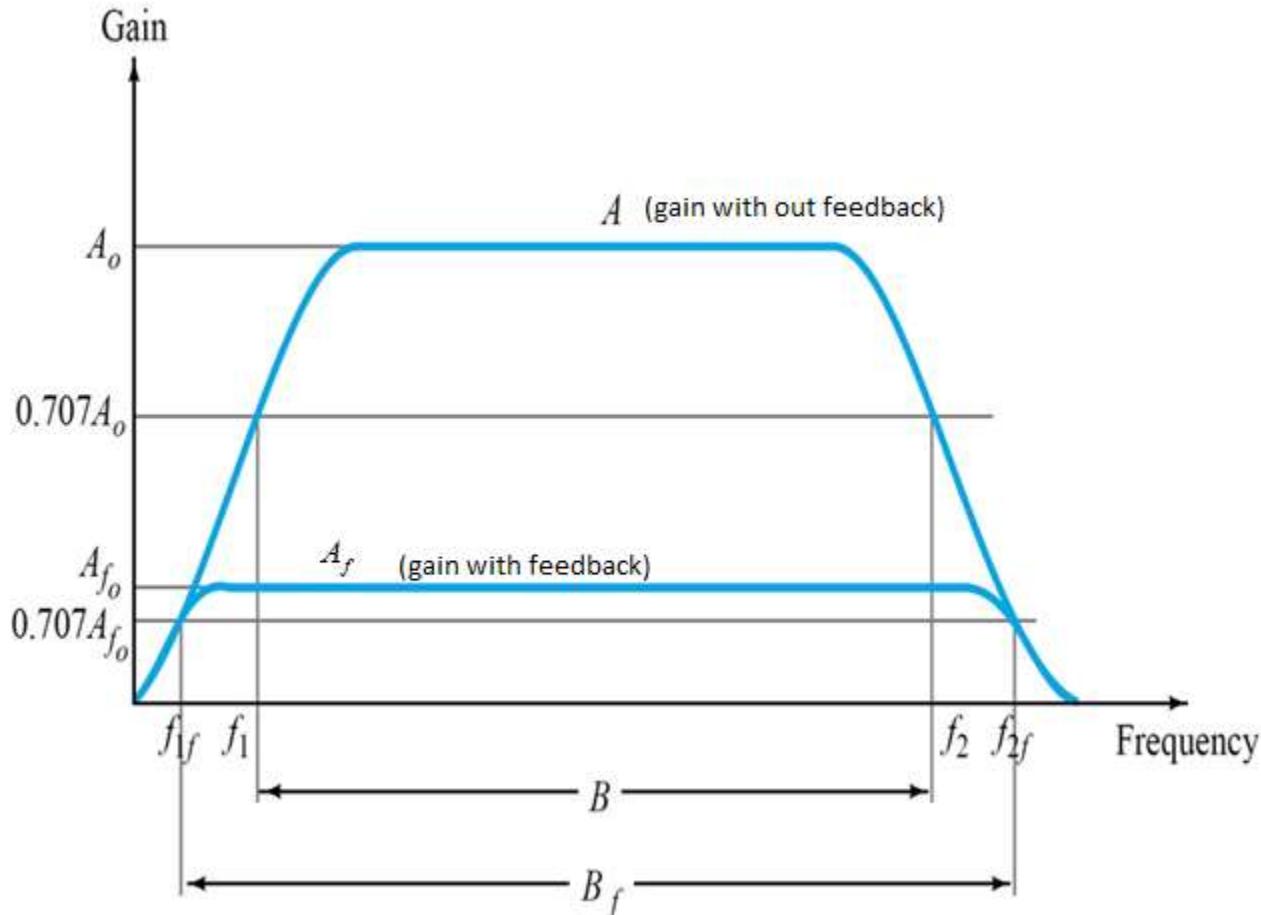
TABLE 1.2 Effect of Feedback Connection on Input and Output Impedance

Voltage-Series	Current-Series	Voltage-Shunt	Current-Shunt
$Z_{if} = Z_i(1 + \beta A)$ (increased)	$Z_i(1 + \beta A)$ (increased)	$\frac{Z_i}{1 + \beta A}$ (decreased)	$\frac{Z_i}{1 + \beta A}$ (decreased)
$Z_{of} = \frac{Z_o}{1 + \beta A}$ (decreased)	$Z_o(1 + \beta A)$ (increased)	$\frac{Z_o}{1 + \beta A}$ (decreased)	$Z_o(1 + \beta A)$ (increased)

# Effect of negative feedback on frequency response

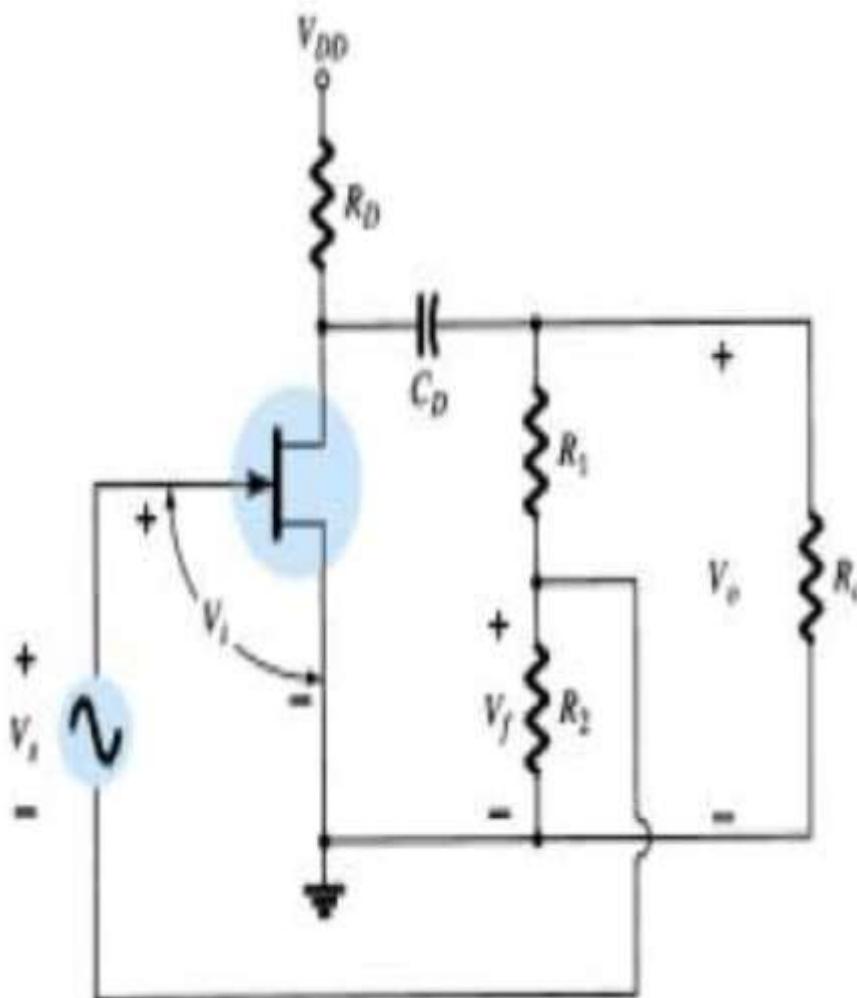
- The gain of the forward amplifier is A.
- The gain of the feedback amplifier is  $A_f = \frac{A}{1 + \beta A}$
- The reduction in gain of the amplifier is  $(1 + A\beta)$
- The bandwidth increases by the same factor  $(1 + A\beta)$
- Hence the gain bandwidth product remain the same.
- Unity gain frequency is remaining the same.

# Frequency response of negative feedback amplifier



# Practical Feedback Circuits (1/2)

## □ Voltage series feedback:



- Here,  $R_1$  and  $R_2$  resistors are used as a feedback network.
- A part of output signal is obtained from  $R_2$  to ground.
- $v_f$  is connected in series with the source signal  $v_s$ .
- Without feedback the amplifier gain is:  
$$A = v_o/v_i = -g_m R_L$$
- where  $R_L$  is the parallel combination of  $R_D$ ,  $R_o$  and  $(R_1 + R_2)$ .

# Practical Feedback Circuits (2/2)

- The feedback factor:

$$\beta = \frac{v_f}{v_o} = \frac{-R_2}{R_1 + R_2}$$

- We know the gain with negative feedback is:

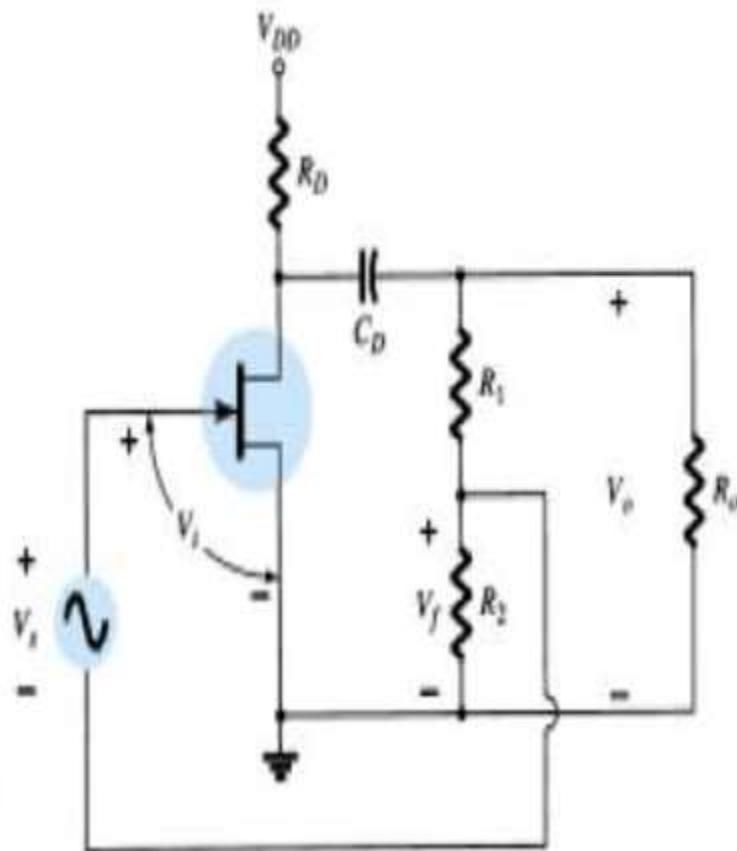
$$A_f = \frac{A}{1 + \beta A} = \frac{-g_m R_L}{1 + \frac{R_2}{R_1 + R_2} R_L g_m}$$

- If  $\beta A \gg 1$  we have:

$$A_f \cong \frac{1}{\beta} = -\frac{R_1 + R_2}{R_2}$$

# Example

Eg. 01) Calculate the gain without and with feedback for the FET amplifier circuit of figure bellow and the following values:  $R_1 = 80 \text{ k}\Omega$ ,  $R_2 = 20 \text{ k}\Omega$ ,  $R_o = 10 \text{ k}\Omega$ ,  $R_D = 10 \text{ k}\Omega$ , and  $g_m = 4000 \mu\text{S}$ ,



# Example

Sol<sup>n</sup>:

$$R_L \cong \frac{R_o R_D}{R_o + R_D} = \frac{10 \text{ k}\Omega (10 \text{ k}\Omega)}{10 \text{ k}\Omega + 10 \text{ k}\Omega} = 5 \text{ k}\Omega$$

Neglecting 100 kΩ resistance of  $R_1$  and  $R_2$  in series

$$A = -g_m R_L = -(4000 \times 10^{-6} \mu\text{S})(5 \text{ k}\Omega) = -20$$

The feedback factor is

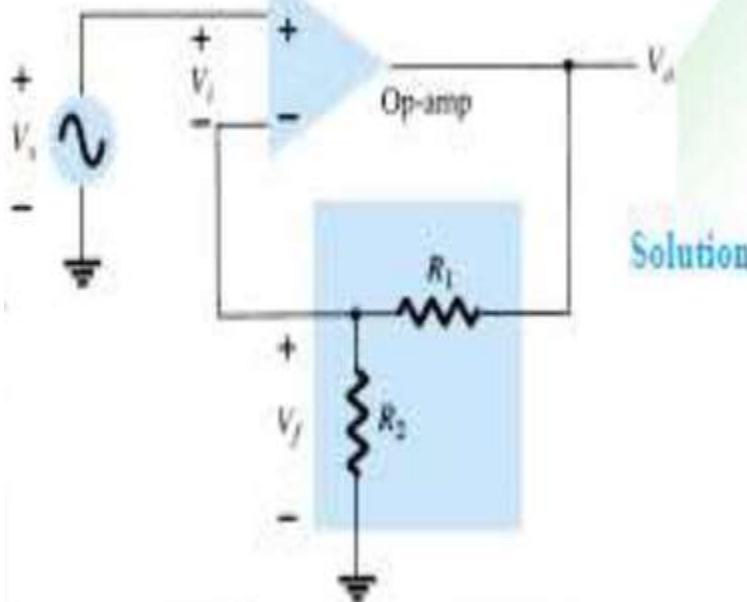
$$\beta = \frac{-R_2}{R_1 + R_2} = \frac{-20 \text{ k}\Omega}{80 \text{ k}\Omega + 20 \text{ k}\Omega} = -0.2$$

The gain with feedback is

$$A_f = \frac{A}{1 + \beta A} = \frac{-20}{1 + (-0.2)(-20)} = \frac{-20}{5} = -4$$

# Example

Eg. 02) Calculate the amplifier gain of the circuit of Fig. bellow for op-amp gain  $A=100,000$  and resistances  $R_1=1.8\text{ k}\Omega$  and  $R_2=200\Omega$



Solution

$$\beta = \frac{R_2}{R_1 + R_2} = \frac{200\Omega}{200\Omega + 1.8\text{ k}\Omega} = 0.1$$

$$A_f = \frac{A}{1 + \beta A} = \frac{100,000}{1 + (0.1)(100,000)}$$
$$= \frac{100,000}{10,001} = 9.999$$

Note that since  $\beta A \gg 1$ ,

$$A_f \cong \frac{1}{\beta} = \frac{1}{0.1} = 10$$

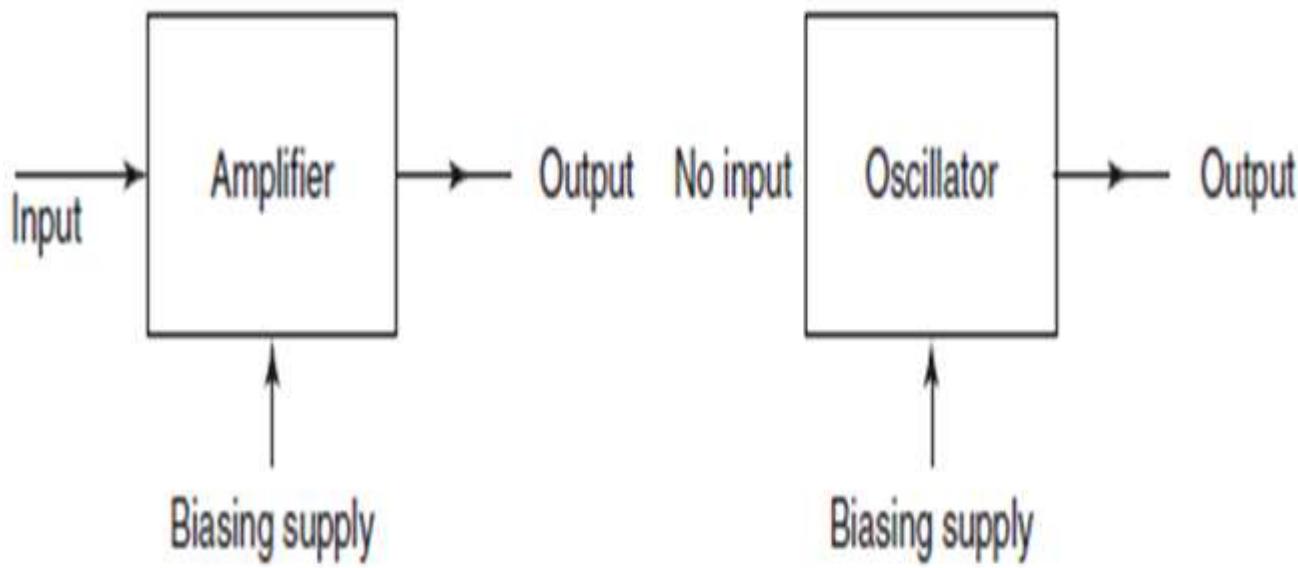
# Oscillator



# INTRODUCTION:

- ▶ An oscillator is an electronic circuit that gives an output without any input signal.
- ▶ It comprises active and passive circuit elements and sinusoidal produces repetitive waveforms at the output **without the application of a *direct external input signal to the circuit.***
- ▶ *It* converts the dc power from the source to ac power in the load. A rectifier circuit converts ac to dc power, but an oscillator converts dc noise signal/power to its ac equivalent.
- ▶ The general form of a harmonic oscillator is an electronic amplifier with the output attached to a narrow-band electronic filter and the output of the filter attached to the input of the amplifier.

# Difference between an amplifier and an oscillator:



Schematic block diagrams showing the difference between an amplifier and an oscillator

# CLASSIFICATIONS OF OSCILLATORS:

- ▶ Oscillators are classified based on the type of the output waveform.
- ▶ If the generated waveform is *sinusoidal* or close to *sinusoidal* (with a certain frequency) then the oscillator is said to be a *Sinusoidal Oscillator*.  
If the output waveform is *non-sinusoidal*, which refers to square/saw-tooth waveforms, the oscillator is said to be a *Relaxation Oscillator*.
- ▶ An oscillator has a positive feedback with the loop gain infinite.

# CONDITIONS FOR OSCILLATION: BARKHAUSEN CRITERIA

Earlier we established that the overall gain of a feedback amplifier is  $A_f = A/(1 + A\beta)$ , where  $A$  is the gain of the internal amplifier,  $\beta$  is the feedback ratio, and  $-A\beta$  is the loop gain. For positive feedback, the feedback gain is expressed as:

$$A_f = \frac{A}{1 - A\beta}$$

For  $A\beta = 1$ , Eq. yields  $A_f = \infty$ . The amplifier then produces an output voltage without any externally applied input voltage. Thus, the amplifier becomes an oscillator. When the signal equals  $V'_o \Rightarrow A\beta V_o = V_o$  or  $A\beta = 1$ , the output voltage regenerates itself and the amplifier oscillates. This condition is called the Barkhausen criterion. This condition means that  $|A\beta| = 1$  and the phase angle of  $A\beta$  is zero or an integral multiple of 360. The basic conditions for oscillation in a feedback amplifier are: (1) the feedback must be regenerative, (2) the loop-gain must be unity, and (3) the phase difference must be zero or an integral multiple of 360.

# CLASSIFICATIONS OF OSCILLATORS:

- The classification of various oscillators is shown in the table

Different types of oscillators and their frequency ranges

Type of Oscillator	Frequency Range Used
1. Audio-frequency oscillator	20 Hz – 20 kHz
2. Radio-frequency oscillator	20 kHz – 30 MHz
3. Very-high-frequency oscillator	30 MHz – 300 MHz
4. Ultra-high-frequency oscillator	300 MHz – 3 GHz
5. Microwave oscillator	3 GHz – 30 GHz
6. Millimeter wave oscillator	30 GHz – 300 GHz

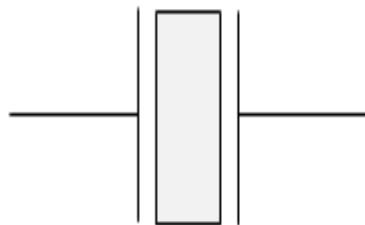
# CHARACTERISTICS OF OSCILLATORS:

- ▶ Oscillators are a common element of almost all electronic circuits. They are used in various applications, and their use makes it possible for circuits and subsystems to perform numerous useful functions.
- ▶ In oscillator circuits, oscillation usually builds up from zero when power is first applied under linear circuit operation.
- ▶ The oscillator's amplitude is kept from building up by limiting the amplifier saturation and various non-linear effects.
- ▶ Oscillator design and simulation is a complicated process. It is also extremely important and crucial to design a good and stable oscillator.
- ▶ Oscillators are commonly used in communication circuits. All the communication circuits for different modulation techniques—AM, FM, PM—the use of an oscillator is must.
- ▶ Oscillators are used as stable frequency sources in a variety of electronic applications. (Quartz watches)
- ▶ Oscillator circuits are used in computer peripherals, counters, timers, calculators, phase-locked loops, digital multi-metres, oscilloscopes, and numerous other applications.

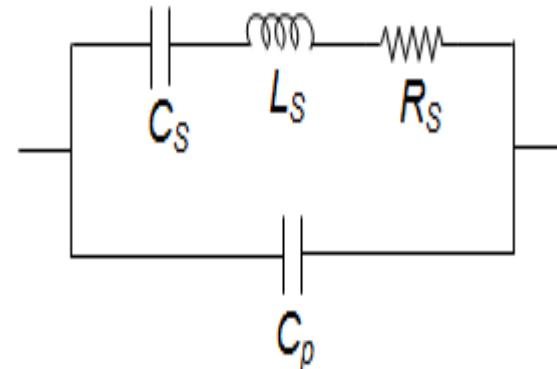
# CRYSTAL OSCILLATOR:

- ▶ Crystal oscillator is most commonly used oscillator with high-frequency stability. They are used for laboratory experiments, communication circuits and biomedical instruments. They are usually fixed frequency oscillators where stability and accuracy are the primary considerations.
- ▶ In order to design a stable and accurate LC oscillator for the upper HF and higher frequencies it is absolutely necessary to have a crystal control; hence, the reason for crystal oscillators.
- ▶ Crystal oscillators are oscillators where the primary frequency determining element is a quartz crystal. Because of the inherent characteristics of the quartz crystal the crystal oscillator may be held to extreme accuracy of frequency stability. Temperature compensation may be applied to crystal oscillators to improve thermal stability of the crystal oscillator.
- ▶ The crystal size and cut determine the values of  $L$ ,  $C$ ,  $R$  and  $C'$ . *The resistance  $R$  is the friction of the vibrating crystal, capacitance  $C$  is the compliance, and inductance  $L$  is the equivalent mass. The capacitance  $C'$  is the electrostatic capacitance between the mounted pair of electrodes with the crystal as the dielectric.*

## Equivalent Circuit Diagram of Piezoelectric Crystal :



(a)



(b)

Figure (a) Quartz Crystal (b) Equivalent Electric Circuit

Due to the presence of  $C_p$ , the crystal will resonate at two different frequencies viz.,

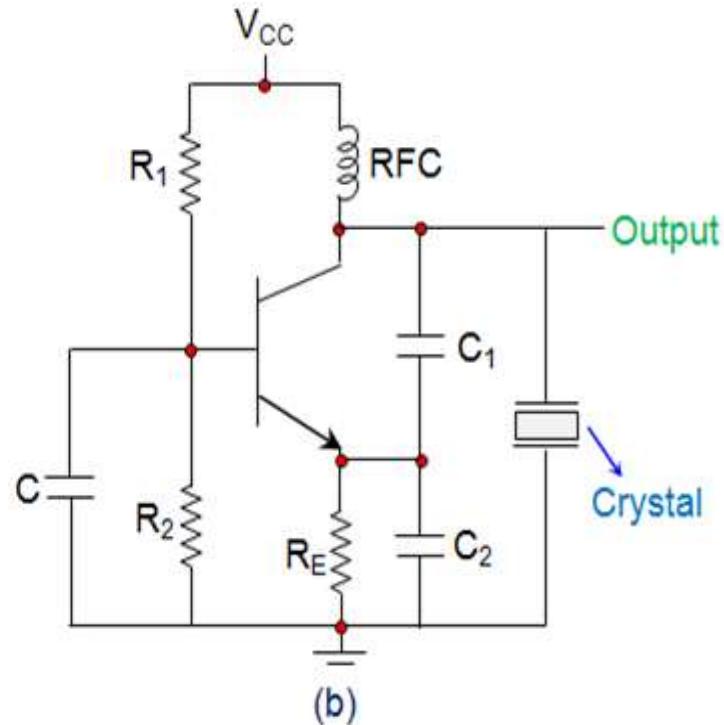
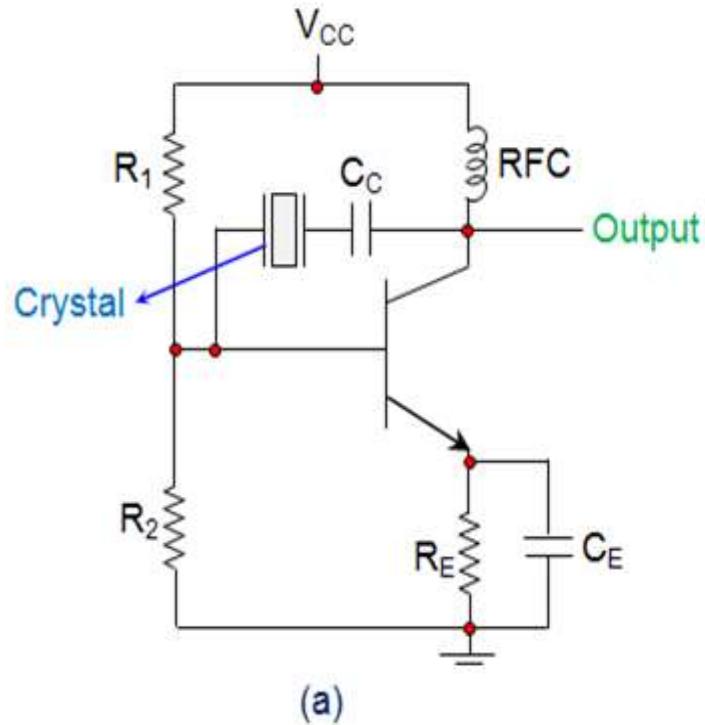
1. Series Resonant Frequency,  $f_s$  which occurs when the series capacitance  $C_S$  resonates with the series inductance  $L_S$ . At this stage, the crystal impedance will be the least and hence the amount of feedback will be the largest. Mathematical expression for the same is given as

$$f_s = \frac{1}{2\pi\sqrt{L_s C_s}}$$

2. Parallel Resonant frequency,  $f_p$  which is exhibited when the reactance of the  $L_S C_S$  leg equals the reactance of the parallel capacitor  $C_p$  i.e.  $L_S$  and  $C_S$  resonate with  $C_p$ . At this instant, the crystal impedance will be the highest and thus the feedback will be the least. Mathematically it can be given as

$$f_p = \frac{1}{2\pi\sqrt{L_S \frac{C_p C_S}{C_p + C_S}}}$$

# Circuit Diagram of crystal oscillator:



Crystal Oscillator Operating in (a) Series Resonance (b) Parallel Resonance

# Operation of Crystal Oscillator:

- ▶ In the circuits shown, the resistors  $R_1$  and  $R_2$  form the voltage divider network while the emitter resistor  $R_E$  stabilizes the circuit. Further,  $C_E$  acts as an AC bypass capacitor while the coupling capacitor  $C_C$  is used to block DC signal propagation between the collector and the base terminals.
- ▶ Next, the capacitors  $C_1$  and  $C_2$  form the capacitive voltage divider network. In addition, there is also a Radio Frequency Coil (RFC) in the circuits which offers dual advantage as it provides even the DC bias as well as frees the circuit-output from being affected by the AC signal .
- ▶ On supplying the power to the oscillator, the amplitude of the oscillations in the circuit increases until a point is reached wherein the nonlinearities in the amplifier reduce the loop gain to unity. Next, on reaching the steady-state, the crystal in the feedback loop highly influences the frequency of the operating circuit. Further, here, the frequency will self-adjust so as to facilitate the crystal to present a reactance to the circuit such that the Barkhausen phase requirement is fulfilled.
- ▶ The typical operating range of the crystal oscillators is from 40 KHz to 100 MHz

## Advantages of Crystal Oscillator:

- The crystal oscillator has very low frequency drift due to change in temperature and other parameters.
- The crystal oscillator Q is very high.
- It has Automatic amplitude control.
- It has very high frequency **stability**.
- The crystal oscillator is possible to obtain very high precise and **stable** frequency of oscillators.



## **Disadvantages of crystal oscillator:**

- Crystals of low fundamental frequencies are not easily available.
- These are suitable for high frequency application.

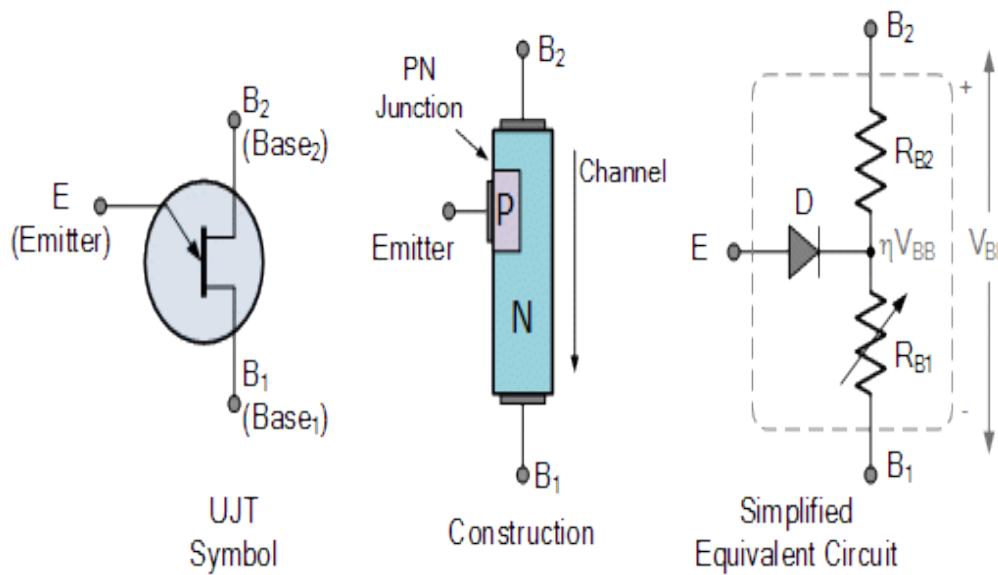
## **Applications of crystal oscillator:**

- The crystal oscillators are used in radio and TV transmitters.
- It is used as a crystal clock in microprocessors.
- It is used in the frequency synthesizers.
- It is used in special types of receivers.

# OVERVIEW OF UJT, RELAXATION OSCILLATOR

## OVERVIEW OF UJT:

- An UJT is formed from a P-type and the N-type material to create a single junction.
- These transistors can be utilized during the switching of the devices to ON/OFF. They don't suit for amplification techniques.

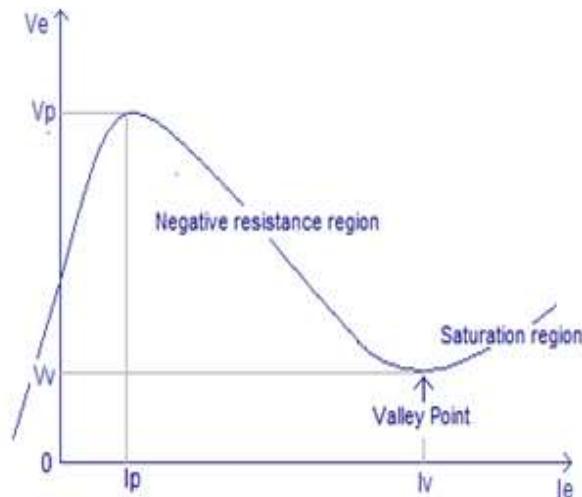
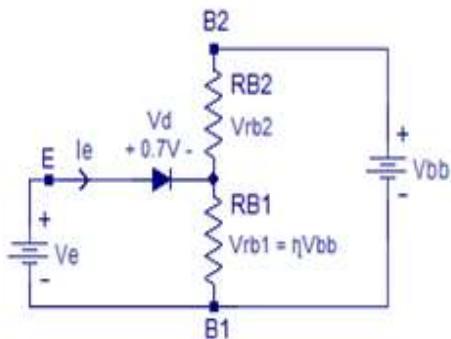


# UJT Characteristics:

The characteristics of the UJT are as follows:

- It requires very low amount of voltage to get triggered.
- It is capable of controlling the current pulse.
- It consists of the negative value of the resistance.
- The cost of this transistor is very low.

- As the current in the UJT tends to increase there can be evident drop in voltage value. Hence this transistor shows the negative characteristics of resistance. This paves the way to make the UJT to work as a relaxation oscillator.



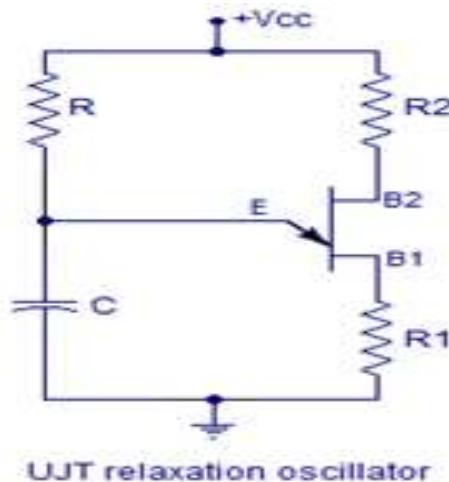
# UJT RELAXATION OSCILLATOR:

- UJT is a transistor with one junction.
- This possesses the resistance with negative characteristics.
- This makes the UJT to function as an oscillator.
- This is an oscillator with the basic resistor and capacitor.
- As it is good at switching, it takes minimum value of the nano seconds for switching the devices.

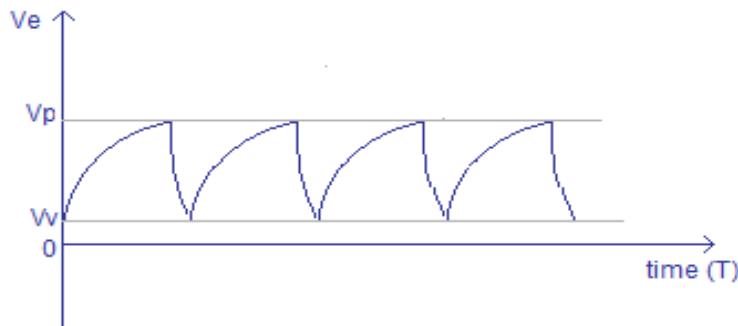
- The circuit diagram of a UJT relaxation oscillator is given below.
- R<sub>1</sub> and R<sub>2</sub> are current limiting resistors. Resistor R and capacitor C determines the frequency of the oscillator.
- The frequency of the UJT relaxation oscillator can be expressed by the equation,

$$F = 1/(RC \ln(1/(1-\eta)))$$

where  $\eta$  is the intrinsic standoff ratio and  $\ln$  stand for natural logarithm.



- When power supply is switched ON, the capacitor C starts charging through resistor R.
- The capacitor keeps on charging until the voltage across it becomes equal to  $0.7V + \eta V_{bb}$ .
- This voltage is the peak voltage point “ $V_p$ ” denoted in the characteristics curve. After this point the emitter to RB1 resistance drops drastically and the capacitors starts discharging through this path.
- When the capacitor is discharged to the valley point voltage “ $V_v$ ”, the emitter to RB1 resistance climbs again and the capacitor starts charging.

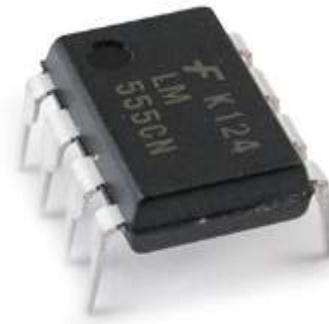


Wave form across the capacitor in a UJT relaxation oscillator

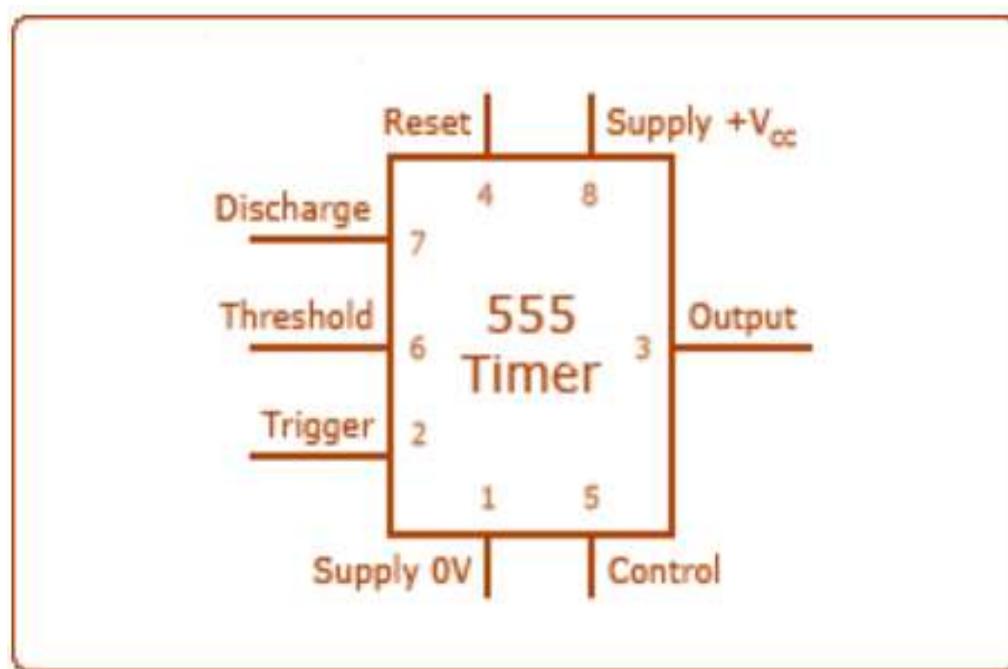
# **IC 555 TIMER**

# The 555 Timer

- The 555 Timer is one of the most popular and versatile integrated circuits ever produced.
- It is 50 years old and still being used.
- It is a combination of digital and analog circuits.
- It is known as the “**time machine**” as it performs a wide variety of timing tasks.
- The 555 timer IC is shown here.



## 555 Timer Pin Connection



# 555 Timer Pin Function

PIN 1: Ground – Connect this to ground. Remember to connect all grounds together. in a circuit

PIN 2: Trigger – A short low (less than 1/3 Vcc) pulse on the trigger starts By connecting this to ground we "turn on" the 555 timer.

PIN 3: Output – During a timing interval, the output stays at +VCC. Can source up to 200ma.

PIN 4: Reset – Forces pin 3 low if pulled to ground.

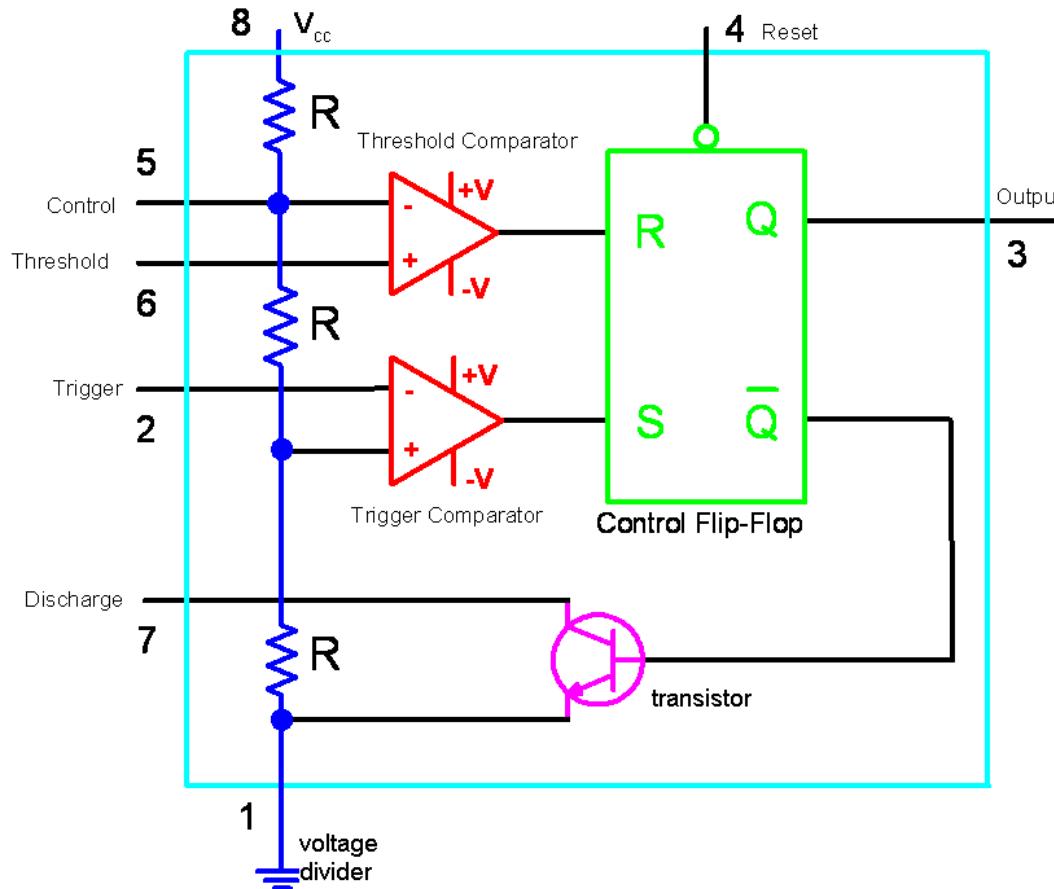
PIN 5: Control – Can be used to adjust threshold trigger voltage. Not used in our applications. Connect to ground with a .01uF cap to eliminate supply noise from Vcc.

PIN 6: Threshold – When threshold crosses above 2/3 Vcc timing interval ends.

PIN 7: Discharge – Connects to ground when output goes low. Controls timing.

PIN 8: Vcc – Power supply. Typical range 4.5v to 16v.

# Internal Circuit of 555 Timer



## Inside the 555 Timer

- The **voltage divider (blue)** has three equal 5K resistors. It divides the input voltage ( $V_{cc}$ ) into three equal parts.
- The **two comparators (red)** are op-amps that compare the voltages at their inputs and saturate depending upon which is greater.
  - The Threshold Comparator saturates when the voltage at the Threshold pin (pin 6) is greater than  $(2/3)V_{cc}$ .
  - The Trigger Comparator saturates when the voltage at the Trigger pin (pin 2) is less than  $(1/3)V_{cc}$

- The **flip-flop (green)** is a bi-stable device. It generates two values, a “high” value equal to Vcc and a “low” value equal to 0V.
  - When the Threshold comparator saturates, the flip flop is Reset (R) and it outputs a low signal at pin 3.
  - When the Trigger comparator saturates, the flip flop is Set (S) and it outputs a high signal at pin 3.
- The **transistor (purple)** is being used as a switch, it connects pin 7 (discharge) to ground when it is closed.
  - When Q is low, Qbar is high. This closes the transistor switch and attaches pin 7 to ground.
  - When Q is high, Qbar is low. This opens the switch and pin 7 is no longer grounded

# Applications for the 555 Timer

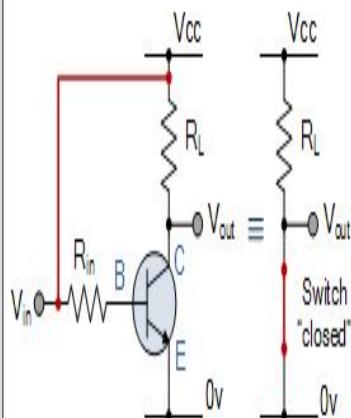
- The 555 is a **general purpose IC** that can be used for
  - Precision timing .
  - Pulse generators – Multivibrators.
  - Sequential timing.
  - Time delay generation.
  - Linear ramp generation.
  - Cascaded timers.
  - Frequency dividers.
  - Voltage-controlled oscillators.
  - LED flashers.

**UNIT-II**

**LOGIC FAMILIES**

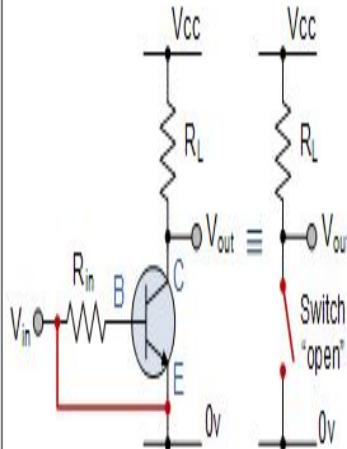
# Transistor as a switch

## Saturation Characteristics



- The input and Base are connected to  $V_{CC}$
- Base-Emitter voltage  $V_{BE} > 0.7\text{v}$
- Base-Emitter junction is forward biased
- Base-Collector junction is forward biased
- Transistor is "fully-ON" (saturation region)
- Max Collector current flows ( $I_C = V_{CC}/R_L$ )
- $V_{CE} = 0$  (ideal saturation)
- $V_{OUT} = V_{CE} = "0"$
- Transistor operates as a "closed switch"

## Cut-off Characteristics



- The input and Base are grounded (0v)
- Base-Emitter voltage  $V_{BE} < 0.7\text{v}$
- Base-Emitter junction is reverse biased
- Base-Collector junction is reverse biased
- Transistor is "fully-OFF" (Cut-off region)
- No Collector current flows ( $I_C = 0$ )
- $V_{OUT} = V_{CE} = V_{CC} = "1"$
- Transistor operates as an "open switch"

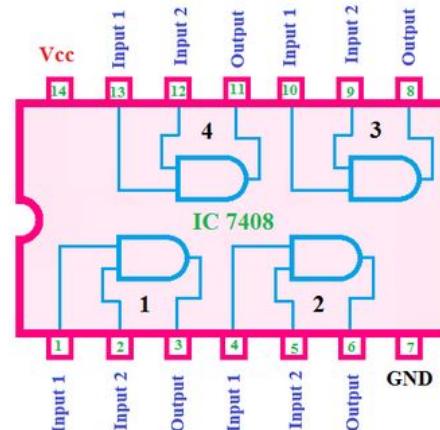
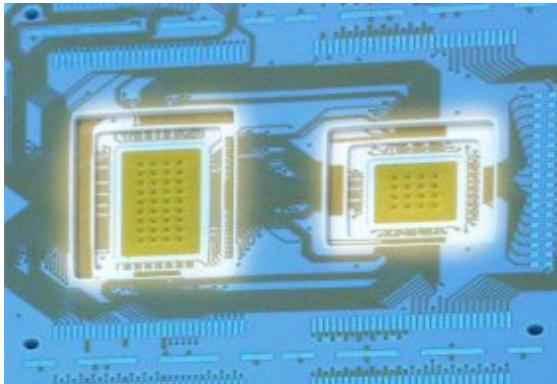
# Typical npn Transistor Parameters

Region	$V_{BE}$ (V)	$V_{CE}$ (V)	Current Relation
Cutoff	< 0.7	Open circuit	$I_B = I_C = 0$
Active	0.7-0.8	> 0.8	$I_C = h_{FE} I_B$
Saturation	0.8-0.9	0.2	$I_B \geq I_C / h_{FE}$

# Characteristics of digital Ics

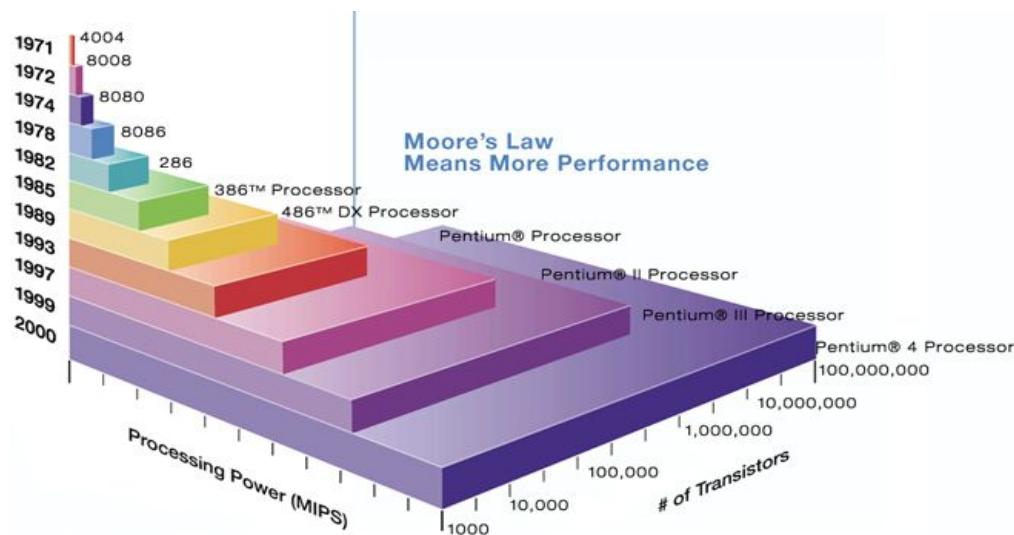
## Integration Levels

- Gate/transistor ratio is roughly 1/10
  - SSI(Small Scale Integration) < 12 gates/chip
  - MSI(Medium Scale Integration) < 100 gates/chip
  - LSI(Large Scale Integration) ...1K gates/chip
  - VLSI(Very Large Scale Integration )...10K gates/chip
  - ULSI(Ultra Large Scale Integration )...100K gates/chip
  - GSI(Giant Scale Integration ) ...1Meg gates/chip



# Moore's law

- A prediction made by Moore (a co-founder of Intel) in 1965:  
“a number of transistors to double every 2 years.”



# Positive logic

Positive logic: H is set to be binary 1  
L is set to be binary 0

Inputs		Output
x	y	z
L	L	H
L	H	H
H	L	H
H	H	L

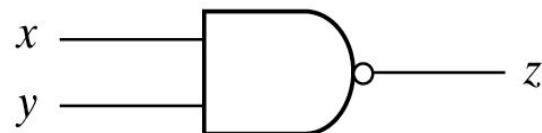


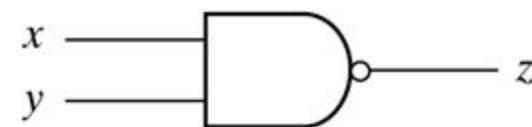
Fig. 10-1 Positive Logic NAND Gate

# Negative logic

Negative logic: L is set to be binary 1

H is set to be binary 0

INPUT		OUTPUT
X	Y	Z
H	H	L
H	L	L
L	H	L
L	L	H



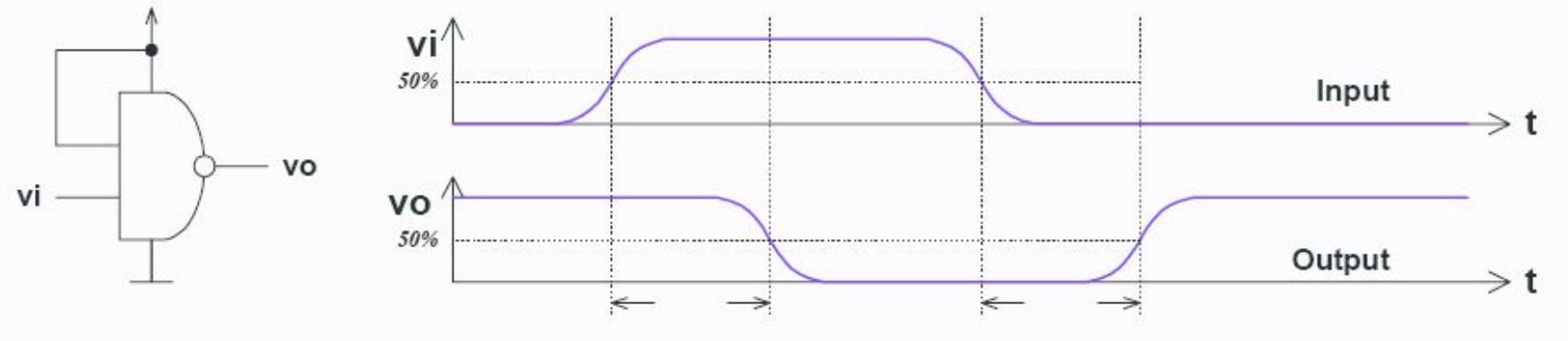
# Digital IC specifications (or) characteristics of digital logic families

- The most useful specifications or feature to be concerned of IC logic families:
  - Threshold voltage
  - Propagation delay
  - Power dissipation
  - Fan-in
  - Fan-out
  - Voltage and current requirement
  - Noise Margin
  - Operating temperature
  - Speed power product

# Digital IC Specifications

- **Threshold voltage:** The voltage at the input of a gate which causes a change in the state of the output from one logic level to the other.
- **Propagation delay:** Time interval b/w the application of an i/p pulse & occurrence of resulting o/p pulse.
  - The average *transition-delay time* for the signal to propagate from input to output when the binary signal changes in value.
  - A pulse through a gate takes a certain amount of time to propagate from input to output. This interval of time is known as the propagation delay of the gate.
  - Average transition delay time  $tpd = (t_{PLH} + t_{PHL}) / 2$

# Logic families: propagation delay



$T_{PD,HL}$  – input-to-output propagation delay from HI to LO output

$T_{PD,LH}$  – input-to-output propagation delay from LO to HI output

# Power dissipation

- Every logic gate draws some current from the supply for its operation
- The current drawn in HIGH state is different from that drawn in LOW state.
- **Power dissipation of a logic gate is the power required by the gate to operate with 50% duty cycle at specified frequency.**
- The power needed by the gate expressed in mW
- $P_d = V_{cc} * I_{cc(\text{avg})} * n$ . where  $V_{cc}$  is the gate supply voltage,  $I_{cc(\text{avg})}$  is the average current drawn from the supply by the entire IC,  $n$  is the number of gates in the IC.
- $I_{cc(\text{avg})} = (ICCH + ICCL)/2$
- ICCH-current drawn by the IC when all the gates in the IC are in HIGH state,  
ICCL-current drawn by the IC when all the gates in the IC are in LOW state,
- Total power consumed by an IC is equal to the product of the power dissipated by each gate and the no. of gates in that IC.

# Fan-in and Fan-out

- **Fan-in** :The fan-in of a logic gate is defined as the number of inputs that the gate is designed to handle.
- **Fan-out:** The number of standard loads can be connected to the output of the gate without degrading its normal operation. Sometimes the term *loading* is used
  - High state fan-out : the fan-out of the gate when its output is logic 1-driving gate sources current
  - Low state fan-out : the fan-out of the gate when its output is logic 0- driving gate sinks current
  - Actual fan out is smaller of these two numbers.

# Logic families: Voltage levels

$V_{OH}(\min)$  – The minimum voltage level at an output in the logical “1” state under defined load conditions

$V_{OL}(\max)$  – The maximum voltage level at an output in the logical “0” state under defined load conditions

$V_{IH}(\min)$  – The minimum voltage required at an input to be recognized as “1” logical state

$V_{IL}(\max)$  – The maximum voltage required at an input that still will be recognized as “0” logical state

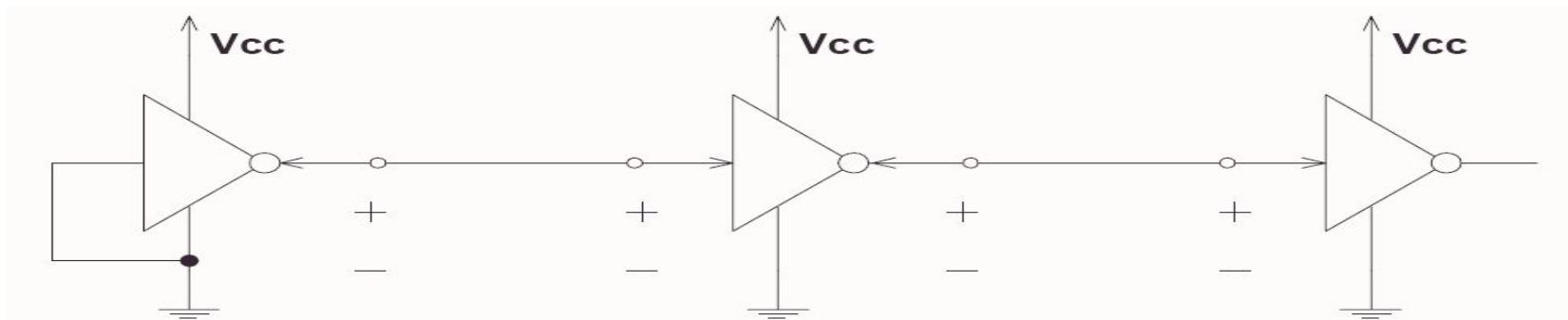
# Logic families: current requirements

$I_{OH}$  – Current flowing into an output in the logical “1” state under specified load conditions

$I_{OL}$  – Current flowing into an output in the logical “0” state under specified load conditions

$I_{IH}$  – Current flowing into an input when a specified HI level is applied to that input

$I_{IL}$  – Current flowing into an input when a specified LO level is applied to that input



# Noise margin

- The unwanted signals are referred to as *noise*
- Noise margin is the *maximum noise* added to an input signal of a digital circuit that does not cause an undesirable change in the circuit output .
- High state noise margin is the difference between the lowest possible high output and the minimum input voltage required for a HIGH.
- Low state noise margin is the difference between the largest possible low output and the maximum input voltage required for a LOW.

### **- operating temperature**

- The IC gates and other circuits are temperature sensitive.
- They are designed to operate satisfactorily over a specified range of temperature.
- Range for commercial application is 0° to 70° C, industrial 0° to 85° C and for military applications -55° to 125° C.

### **- Speed power product**

- Multiply the gate propagation delay by the gate power dissipation
- Smaller value is desirable. The smaller value gives better overall performance.
- It has the units of energy and expressed in picco-joules
- The IC has an average propagation delay of 10ns and an average power dissipation of 5mw, the speed power product is  
 $10 \text{ ns} * 5\text{mw} = 5 * 10^{-12} \text{ watts-seconds} = 5 \text{ picco-joules}$

# COMPARISON OF LOGIC FAMILIES

TYPICAL VALUES OF SPEED AND POWER FROM  
VARIOUS TTL LOGIC FAMILIES

lettering	numbering	prop . delay	— typical — pwer consp.	delay power	max FF clock
TTL	7400	10 ns	10 mW	100 pJ	35 MHz
L-TTL	74L00	33	1	33	3
H-TTL	74H00	6	22	132	50
F-TTL	74F00	3 ns	20 mW	60	100
S-TTL	74S00	3 ns	19 mW	57	125 MHz
LS-TTL	74LS00	9	2	18	45
AS-TTL	74AS00	1.7	8	14	200
ALS-TTL	74ALS00	4	1.2	5	70

**UNIT-II**

**LOGIC FAMILIES**

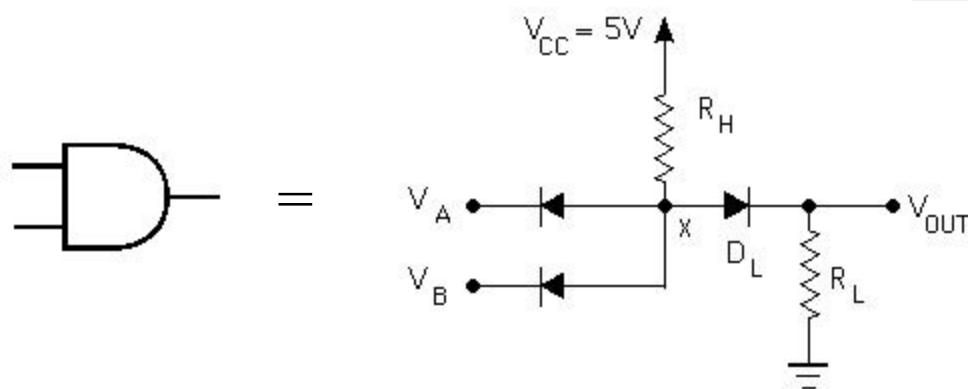
# IC digital logic families

- IC digital logic families
  - -- DL(Diode-logic)
    - RTL(Resistor-transistor logic)
    - DTL(Diode-transistor logic)
    - TTL(Transistor -transistor logic)
    - ECL(Emitter-coupled logic)
    - CMOS(Complementary Metal-oxide semiconductor)

# Diode Logic (DL)

- simplest; does not scale
- NOT not possible (need an active element)

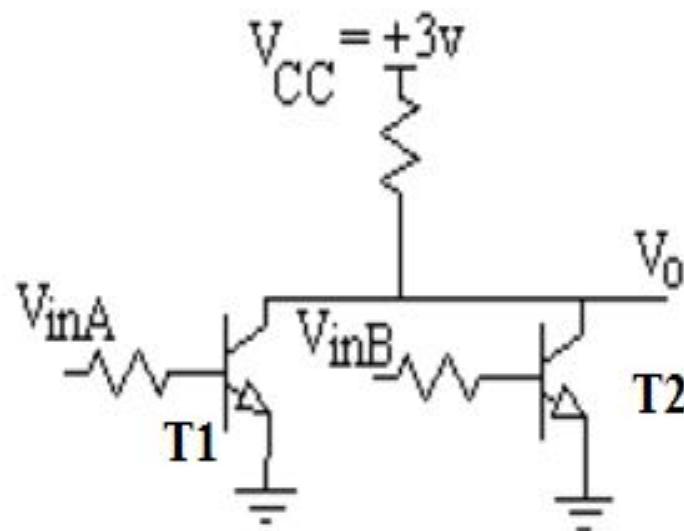
A	B	AND o/p
0	0	0
0	1	0
1	0	0
1	1	1



# **Resistor-Transistor Logic (RTL)**

- O/p voltage drops about 1v when the fan-out is 5
- Power dissipation is about 12mw.
- Propagation delay averages 25ns.

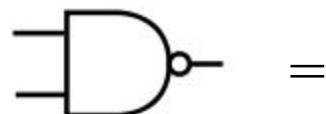
# RTL--NOR



$V_{inA}$	$V_{inB}$	T1	T2	Y
0	0	CUT-OFF	CUT-OFF	1
0	1	CUT-OFF	SAT	0
1	0	SAT	CUT-OFF	0
1	1	SAT	SAT	0

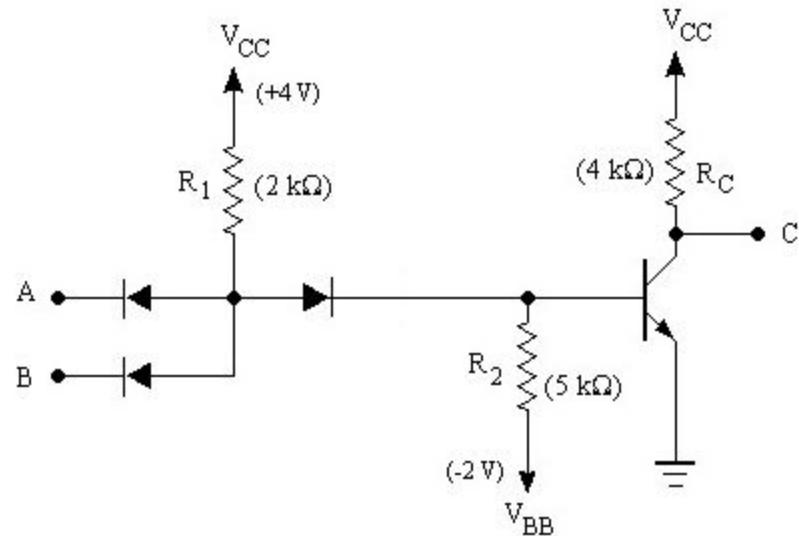
# Diode-Transistor Logic (DTL)

- essentially diode logic with transistor amplification
- reduced power consumption
- faster than RTL



=

A	B	NAND o/p
0	0	1
0	1	1
1	0	1
1	1	0



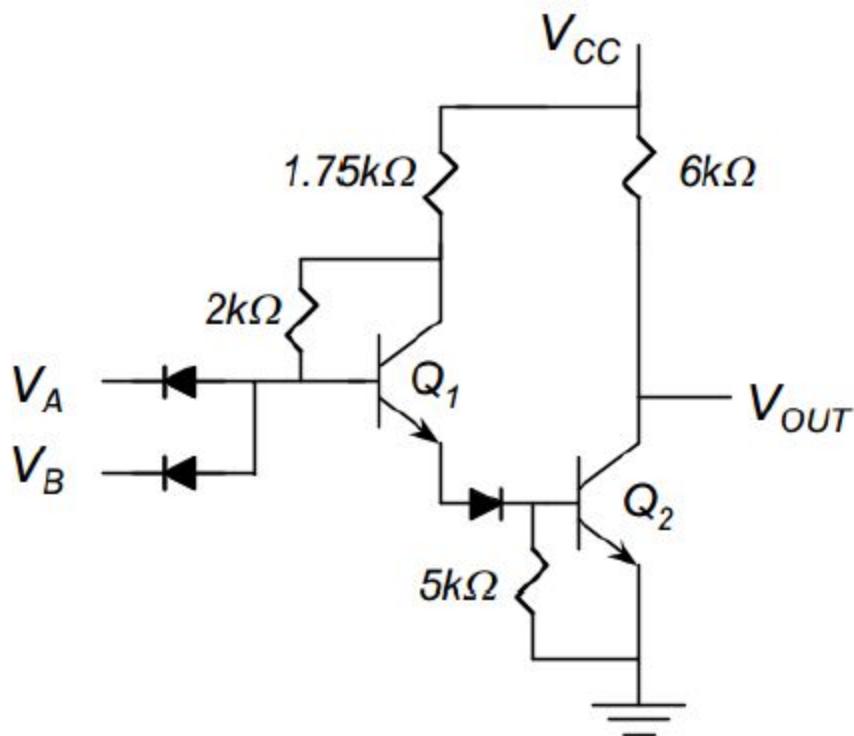
DL AND gate

inverter

# **DTL-NAND**

- Input Diodes Da,Db,Dc conduct through resistor R, if the corresponding input is in the LOW state, while the corresponding to HIGH state the is non conducting.
- If at least one of the input is LOW, the diode connected to their input conducts and the voltage at the output is one diode drop above the low level voltage at the input. this voltage keep the transistor in cut-off. The output of transistor is Vcc.
- If all the three i/p's are in High state ,the i/p diodes are cut-off and current flowing from Vcc through R should be sufficient to drive transistor in saturation. Therefore the o/p of transistor is Vce(sat).
- Voltage corresponds to logic 1 and logic 0 as Vcc and Vce(sat).
- Delays are associated with the turning on and off of the output transistor.
- While turning on, any capacitance shunting the output of the gate discharges rapidly through the low impedance of the o/p transistor in saturation
- At turn off, the shunt capacitor must charge through the pull-up resistor Rc in addition to the storage time delay. Turn off delay is larger than turn on delay by a factor of 2 or 3.
- Propagation delay of DTL gates are 30 to 80ns.

Modified DTL-this increases fan out capability



# TTL (Transistor-Transistor Logic)

# **Disadvantages of RTL & DTL**

## **RTL :**

1. Poor Noise Margin
2. Poor Fan out
3. Low Speed
4. High power dissipation

## **DTL :**

1. Slow Speed
2. Improved noise margin & Fan out

# **TTL (Transistor-Transistor Logic)**

- The original basic TTL gate was a slight improvement over the DTL gate.
- Introduced several TTL subfamilies or series
- First introduced by in 1964 (Texas Instruments)
- TTL has shaped digital technology in many ways
- Transistor alone performs the logical operation & Logic gate operations
- Transistors operated in saturated mode and cut off mode.
- Fastest of the saturated logic families
- Good speed
- Low manufacturing cost
- Wide range of circuits & applications
- Availability in SSI & MSI
- Relatively high power consumption

# TTL (Transistor-Transistor Logic)

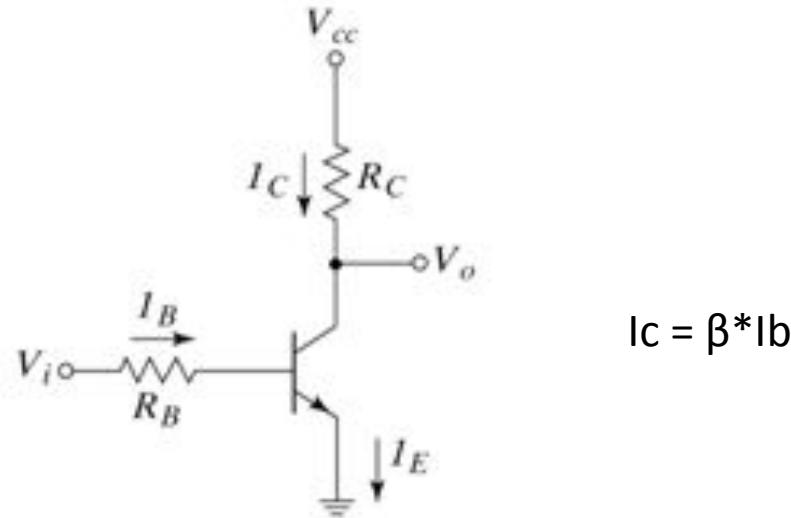
## TTL subfamilies

- Standard TTL
- High speed TTL
- Low power TTL
- Schottky TTL
- Low power Schottky TTL
- Fast TTL

## Three different types of output configurations

1. Open-collector output
2. Totem-pole output
3. Three-state (or tristate) output

# Transistor Operation



(a) Inverter circuit

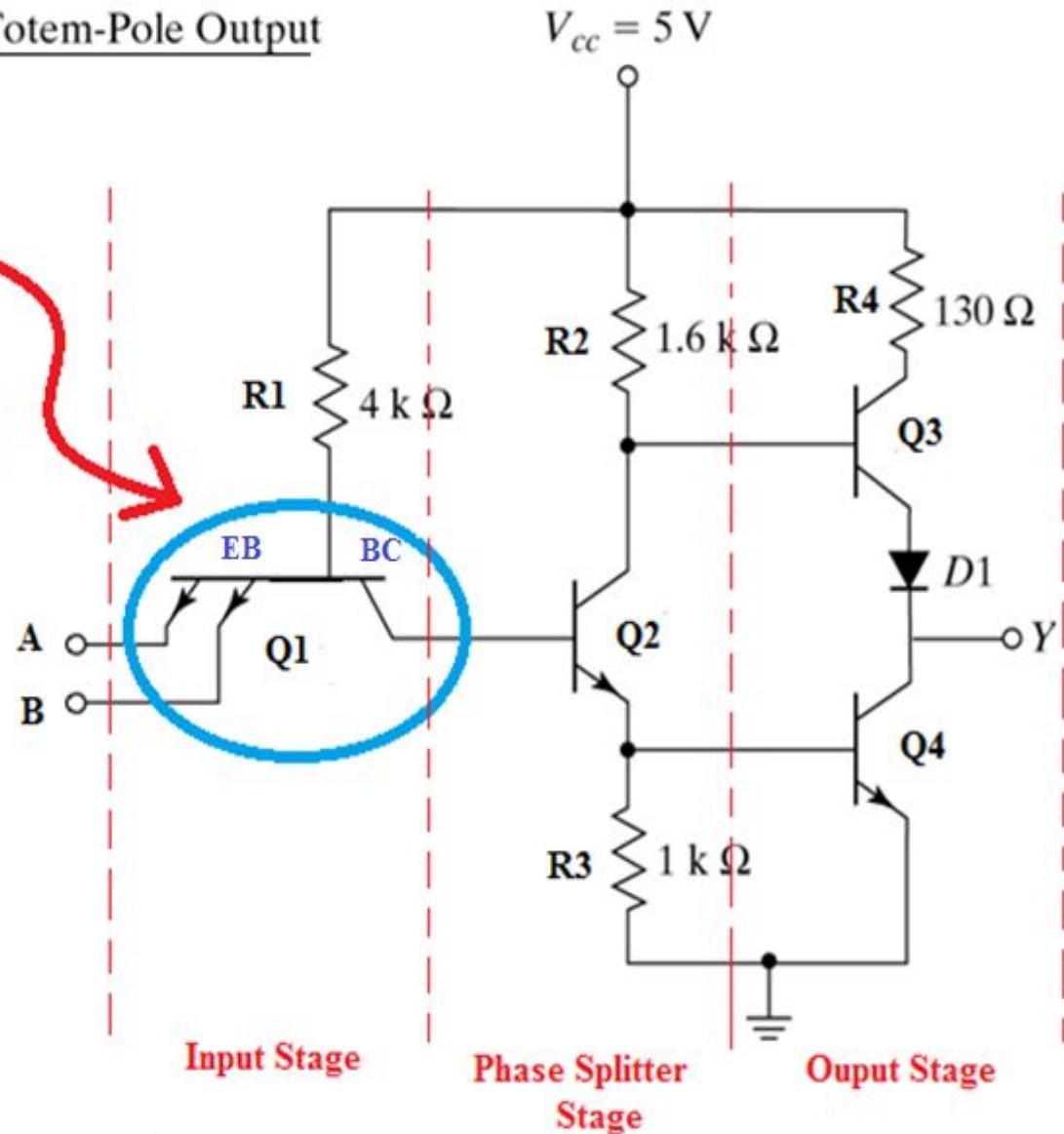
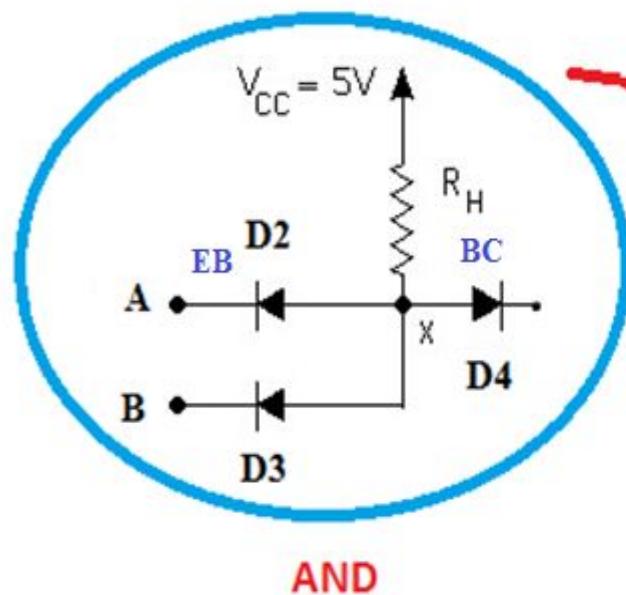
- ❖ When  $V_{in}$  equals 1 (+5V), the transistor is turned on (saturation) and  $V_{out}$  equals 0 (0V).
- ❖ When  $V_{in}$  equals 0 (0V), the transistor is turned off and  $V_{out}$  equals 1 (5V)

# **TTL(Transistor-Transistor Logic)**

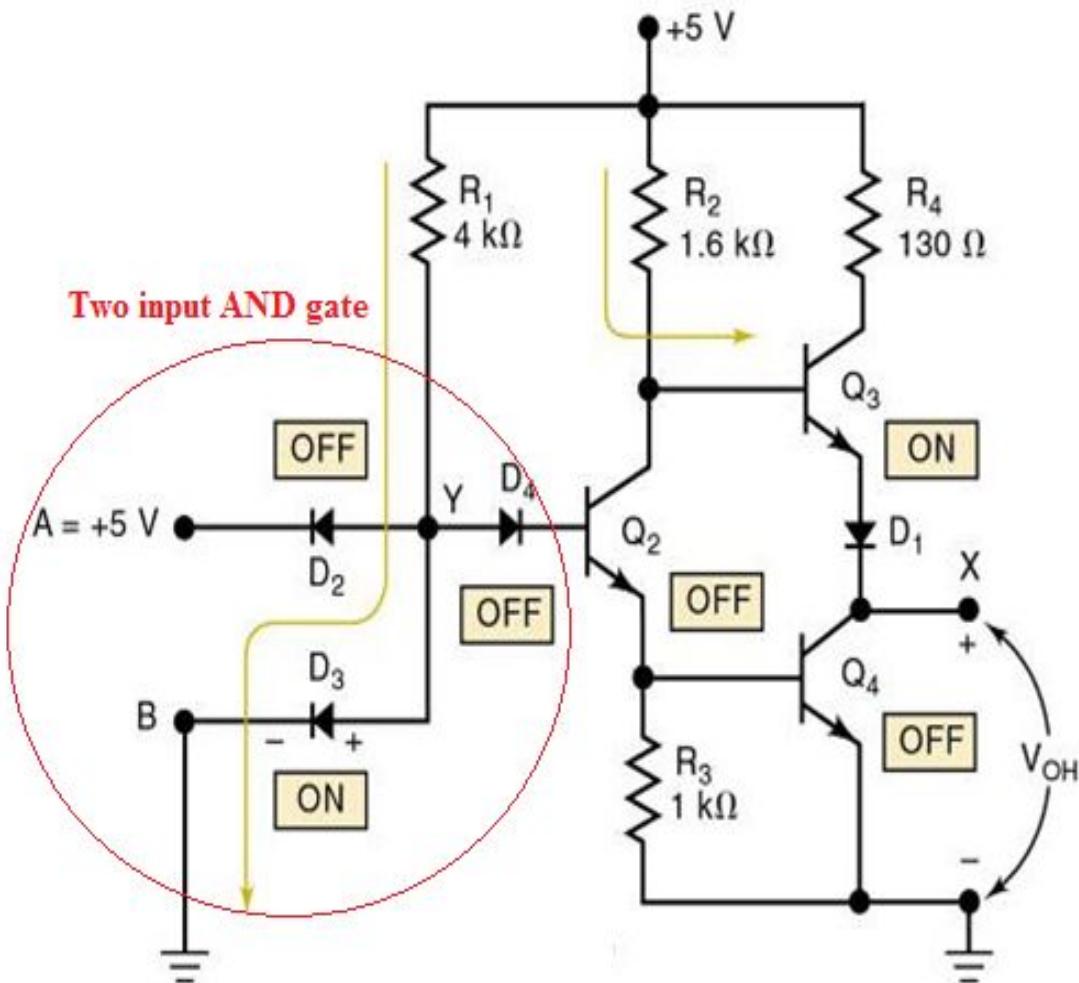
- TTL can overcome disadvantages of RTL and DTL.
- The basic gate was constructed with **different resistor value** to produce gate with **lower dissipation or higher speed**.
- The **propagation delay** of a saturated logic family depends largely on **storage time and RC time constants**.
- Reduce the storage time - decrease the propagation delay.
- **Reduce the resistor values in the circuit, decreases the time constant (RC) and decreases the propagation delay.** But power dissipation is high due to lower resistance draw more current from the supply.
- **Speed of the gate is inversely proportional to propagation delay.**
- In low power TTL the resistor values are higher than standard TTL gate to reduce power dissipation, but increase propagation delay.
- In high speed TTL resistor values are lowered to reduce the propagation delay but the power dissipation is increased.

# TTL Gate with Totem-Pole Output

TTL Gate with Totem-Pole Output



## TTL – NAND Gate



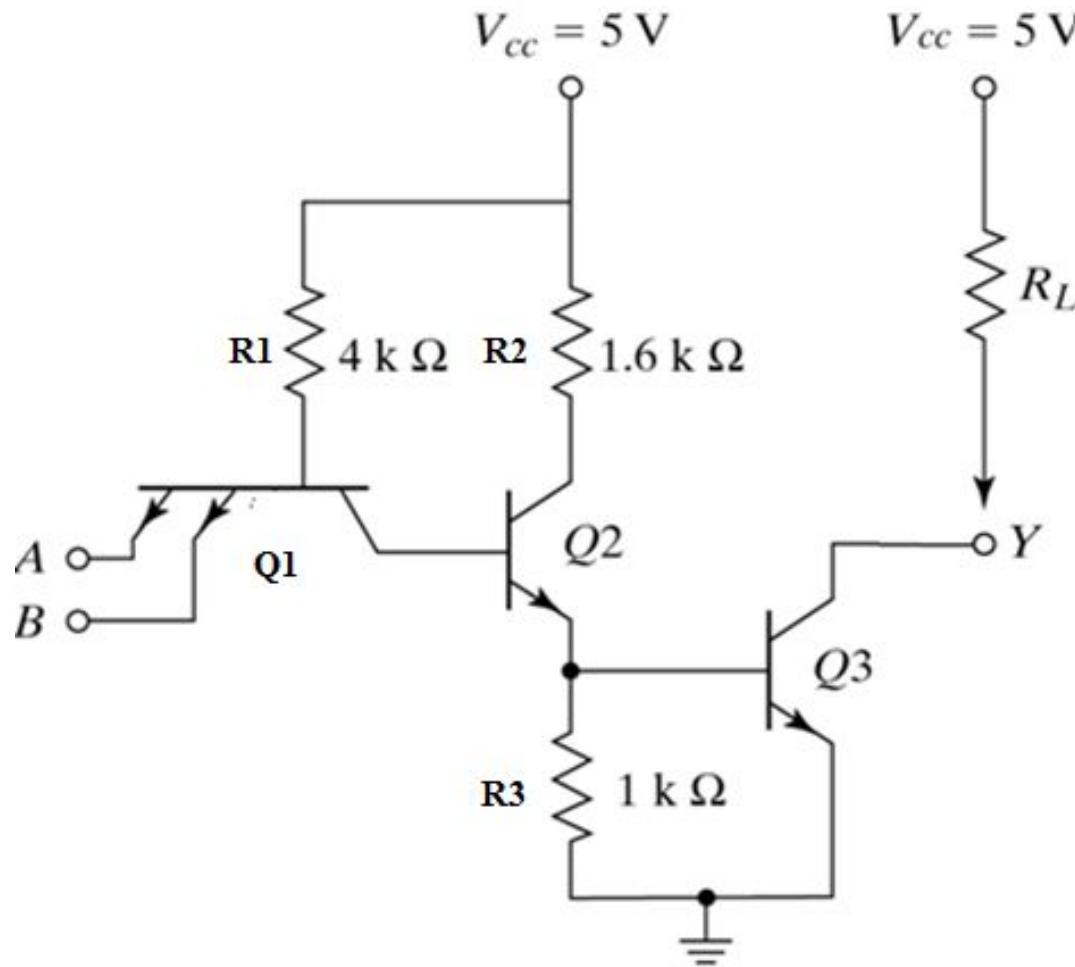
## TTL Gate with Totem-Pole Output

- O/p impedance of the gate normally resistive &capacitive loads.
- Capacitive load consist of capacitance of o/p transistor, capacitance of fan-out gates and any stray wiring capacitances.
- When o/p changes from low to high, the o/p transistor of the gate goes from saturation to cut-off and the total load capacitance ‘C’ charges from low to high with time constant  $RC$ .
- For open collector gate  $R_l=4\text{kohm}$ ,  $c=15\text{pf}$  , $tpd=35\text{ns}$ . With active pull-up is replaced by passive pull-up circuit  $tpd$  is reduced to  $10\text{ns}$ . This is called totem pole output.
- When o/p  $Y=\text{low}$ ,  $Q_2 \& Q_3$  are in saturation.  
 $V_c Q_2 = V_{be}(Q_3) + V_{ce}(Q_2) = 0.7 + 0.2 = 0.9\text{V}$ ,  $V_{ce}(Q_3) = 0.2\text{V}$
- $Q_4$  is cut-off bcos base of  $Q_4$  require  $2 * 0.6 = 1.2\text{V}$ .

# TTL Gate with Totem-Pole Output

- When o/p Y=high, one of the i/p drops to low ,Q2&Q3 are in cut-off. o/p remains low bcos the voltage across the load capacitance cannot change instantaneously. As soon as Q2 turns off Q4 conducts bcos base is connected to Vcc through 1.6Kohm.
- The current needed to charge the load ‘ C’ causes Q4 to saturate, o/p voltage rises with time constant RC.
- $R=130\text{ohm}+R_{\text{sat}}(\text{Q4})+R_d \sim=150\text{ohm}$  less compare to passive pull up resistor, so transition from low to high is faster.
- When C charges ,o/p voltage rises and current in Q4 decreases so Q4 comes to active region. Final o/p=5v-Vbedrop in Q4-D1 drop=3.6v Q3 goes cut-off very fast. But initial transition Both Q3 &Q4 are on ,peak ct' is drawn from the supply. This ct' spikes generates noise.
- Change of state is frequent current spikes increases and PD also increases.
- Wired connection is not allowed.

# Open-collector TTL – NAND Gate



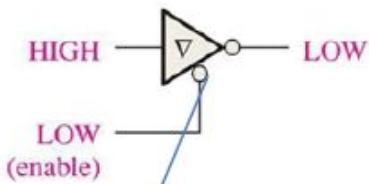
NAND Gate

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

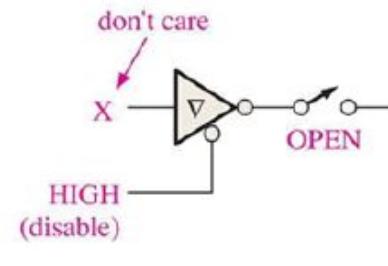
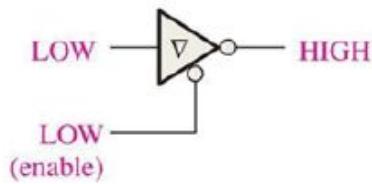
## Open-collector TTL – NAND Gate

- Multiple emitter of Q1 connected to i/ps. Compare to DTL ,i/p diodes--Q1, D1--B-C jn, D2—Q2.
- o/p is taken from open collector of Q3.
- Pull up resistor connected to Vcc, to pull up the o/p to high level when Q3 is off. Otherwise the o/p acts as an open circuit.
- 0.2v for low level,2.4 to 5v for high level.
- If any i/p is low , corresponding B-E jn of Q1 is F.B. the voltage at base of Q1 is  $0.9v(i/p\ 0.2+V_{be}\ drop\ 0.7)$ .
- In order for Q3 to start conducting the path from Q1-Q3 must overcome the potential  $1.8v$  (Diode drop in B-C jn drop+ two Vbe drop of Q2,Q3). But at Q1 is  $0.9v$  so Q3 is cut-off , o/p is high.
- If all i/ps are high Q2, Q3 conduct and saturate. B-E jns of Q1 are all R.B. when Q3 saturates o/p goes low to  $0.2v$ .
- Applications are driving a lamp or relay, performing wired logic and construction of common bus system

# The three states of a tristate circuit.



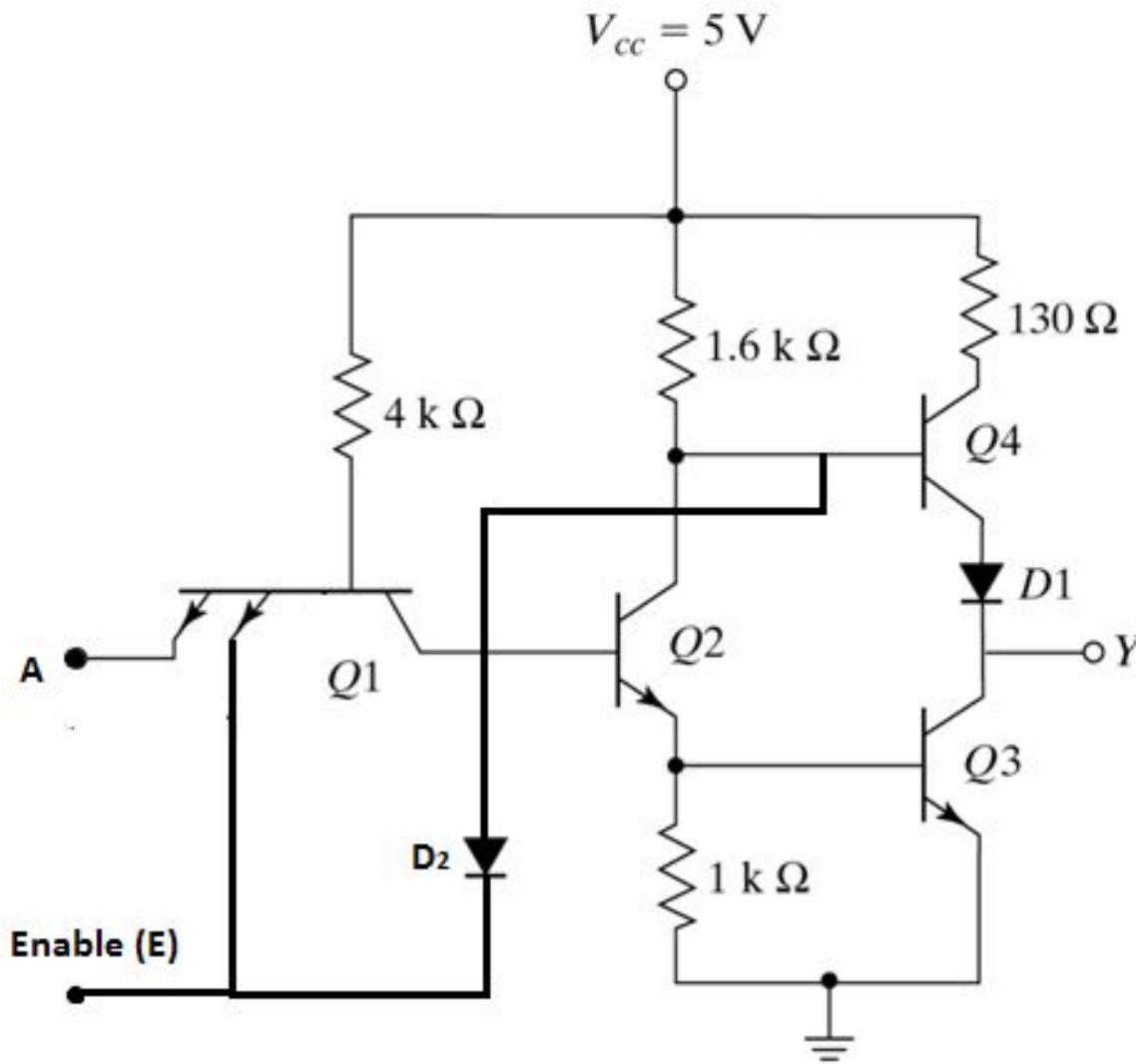
(a) Enabled for normal logic operation



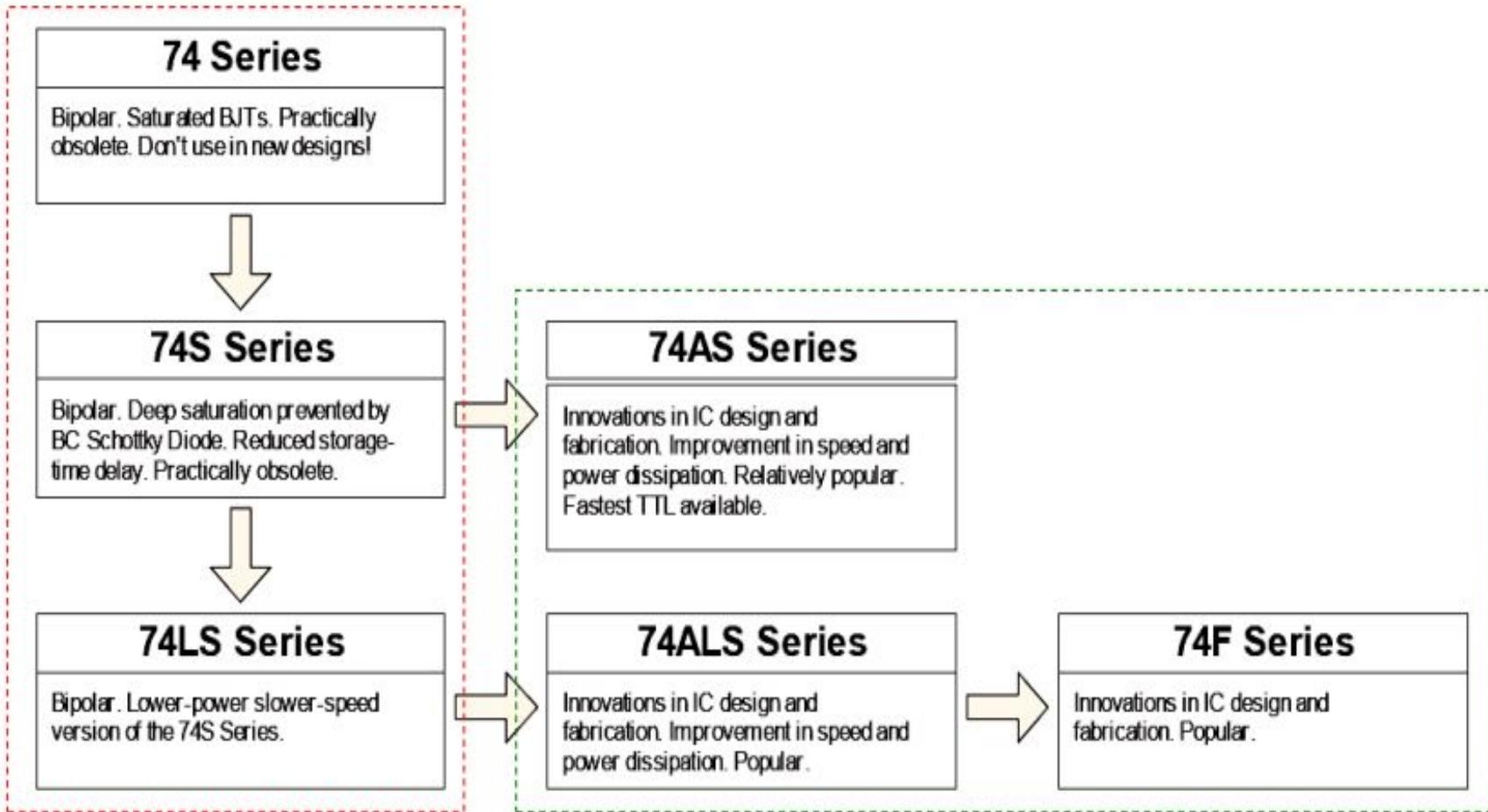
(b) High-Z state

Active low  
Active high also can be used

## Three-state TTL with Inverter operation



# TTL family Evolution



**Legacy: don't use  
in new designs**

**Widely used today**

**UNIT-II**

**LOGIC FAMILIES**

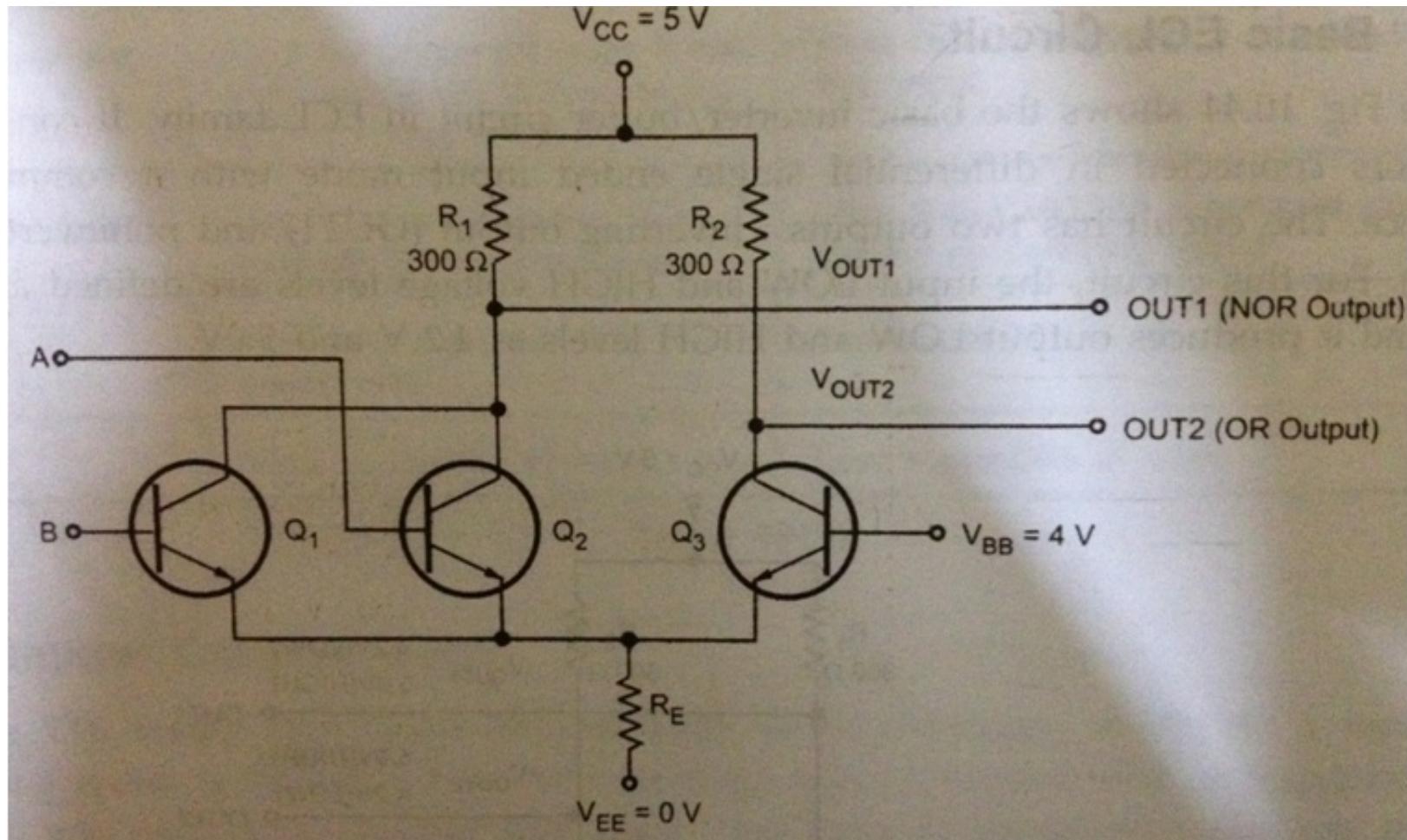
# Emitter-Coupled Logic (ECL)

- Non saturated digital logic family & Used in **high speed circuits**
- **Propagation rate as low as 1-2ns**
- **Noise immunity and power dissipation is high compare of all logic families.**
- Including
  - Differential input amplifier
  - Internal temperature and voltage compensated bias network
  - Emitter-follower outputs

## ECL NOR / OR

- The circuit has 2 outputs- inverting and non inverting output, input low is 3.6 V, input high is 4.4V.
- When  $A=B=0$ , so  $Q1, Q2$  are OFF, then  $Out1=High$  and  $Out2=Low$ .
- When  $A=B=1$ , so  $Q1, Q2$  are ON, then  $Out1= low$  and  $Out2=High$ .

# ECL (NOR/OR GATE)



# Operating Principles of I<sup>2</sup>L gates

- Integrated Injection Logic Operation is explained as follows. To understand the working of a typical I<sup>2</sup>L gate, consider the figure, which consists of five transistors, connected as shown.
- It can be seen that the base current  $I_{B1}$  of  $T_1$  is derived from  $+V_{CC}$  through the input terminal B. The base-bias current  $I_{B3}$  of transistor  $T_3$  and the collector current  $I_{C1}$  of  $T_1$  are obtained from the collector current of transistor  $T_2$  whose injector (emitter) also is connected to  $+V_{CC}$ .

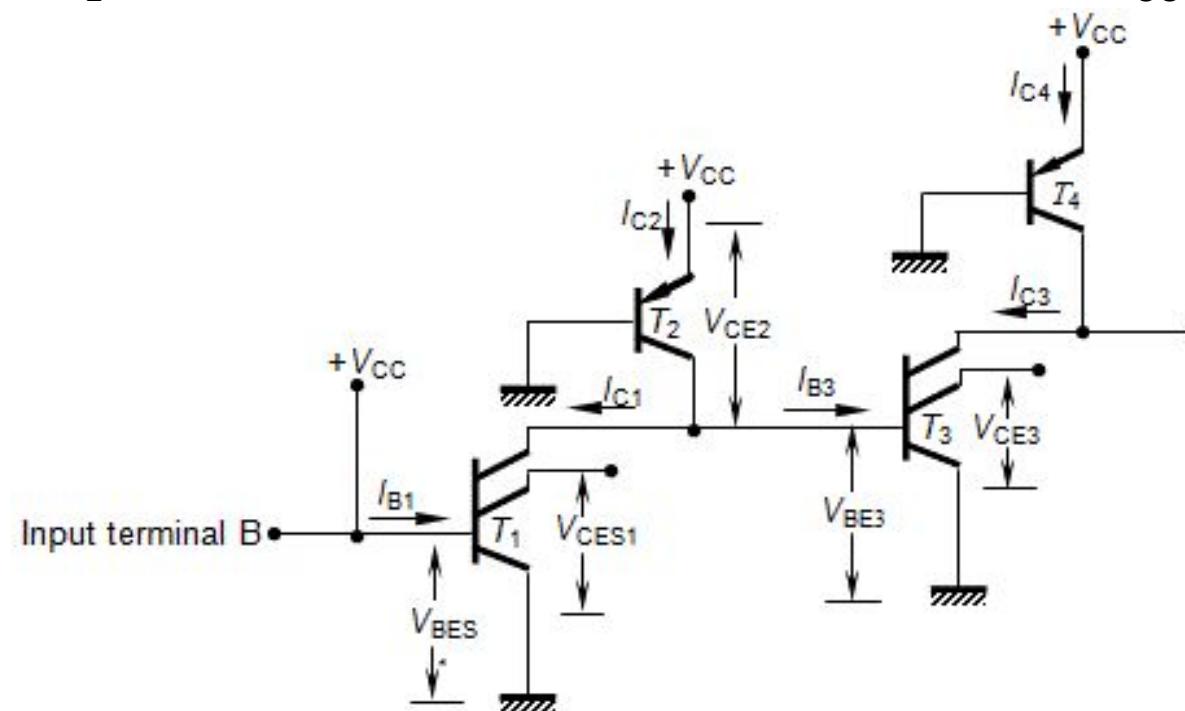


Fig. 3.46 Details of terminal voltages and currents in an I<sup>2</sup>L gate

- If  $I_{B1}$  is sufficient, the base-to-emitter voltage  $V_{BE1}$  of  $T_1$  will become the saturation base-emitter voltage  $V_{BES1}$  ( $= 0.8$  volt), and in this condition, its collector-to-emitter voltage  $V_{CE1}$  will become saturation base-emitter voltage  $V_{CES1}$  ( $= 0.2$  volt).
- So, we find that  $T_1$  is in the ON-state and its output is at logic-**0** level. Since  $V_{CES1} = 0.2$  volt, the collector current  $I_{C2}$  of  $T_2$  will flow through the collector of  $T_1$  as  $I_{C1}$ , and  $V_{BE3}$  of  $T_3 = 0.2$  V  $= V_{CES1}$ .
- This means that at this moment,  $T_3$  is OFF and its collector-emitter voltage  $V_{CE3} = 0.8$  volt. Now, let the input voltage  $V_i = 0$  V. This makes transistor  $T_1$  to be in OFF-state, and  $I_{C2}$  to flow through the base of  $T_2$ .
- Thus,  $V_{BE3} = V_{BES3} = 0.8$  V. This makes  $V_{CE1} = V_{BES3} = \text{logic 1}$ , and  $V_{CE3} = V_{CES3} = \text{logic 0}$ .
- Thus, the transistors perform inversion operation, with logic **0**  $= V_{CES} = 0.2$  V, and logic **1**  $= V_{BES} = 0.8$  V. The voltage swing of this gate, therefore, is

$$V_{\text{swing}} = V_{BES} - V_{CES} = 0.8 - 0.2 = 0.6 \text{ volt}$$

- It can be seen that the logic swing of I<sup>2</sup>L gates is very low and this is a major defect of these gates.
-

# Advantages of I<sup>2</sup>L Gates

- Since I<sup>2</sup>L gates are made up only of BJTs, they possess high speed of operation. Because only transistors are used for the construction, I<sup>2</sup>L gates have high packing density, and are hence suitable for construction of VLSI circuits.
- • Very low power-supply requirement (1 volt) .
- • Low power dissipation
- • Number of processing steps required is small; hence cost per gate is low.
- • Several functions are possible on the same chip.
- • Using bipolar technology, it is possible to combine I<sup>2</sup>L gates with other logic families.

# Disadvantages of I<sup>2</sup>L Gates

- Very low voltage swing ( $\approx 0.6$  V); the swing is between  $V_{BES}$  (0.8 V) and  $V_{CES}$  (0.2 V).
- . Lower packing density than NMOS.
- . Lower noise margin.
- . External resistance required for proper functioning.
- . I<sup>2</sup>L technology, at present, is dormant.
-

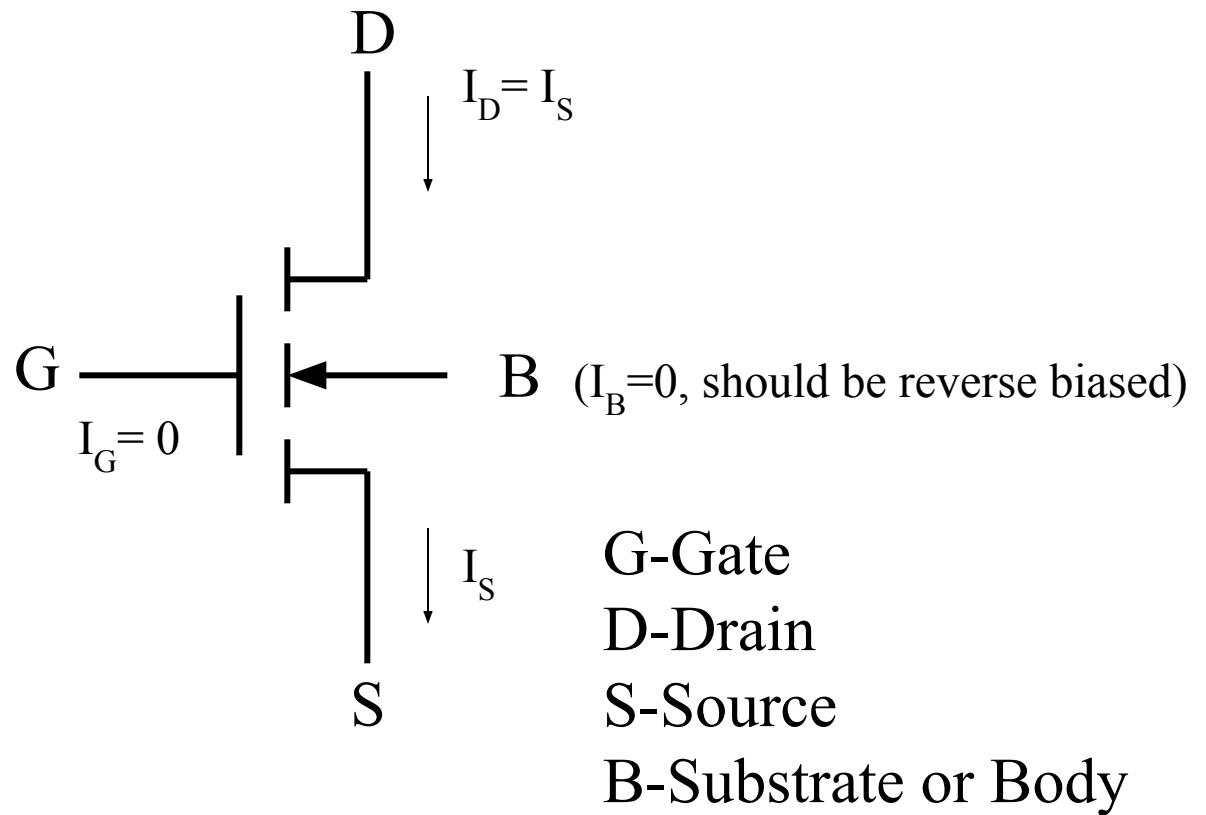
# Unit 2

S-6	SLO-1	Characteristics of MOSFET
	SLO-2	MOSFET Logic

# MOSFETs

- There are 2 types of MOSFET's (metal–oxide–semiconductor field-effect transistor)
- Depletion mode MOSFET (D-MOSFET)
  - Operates in Depletion mode the same way as a JFET when  $V_{GS} \leq 0$
  - Operates in Enhancement mode like E-MOSFET when  $V_{GS} > 0$
- Enhancement Mode MOSFET (E-MOSFET)
  - Operates only in Enhancement mode
  - $I_{DSS} = 0$  until  $V_{GS} > V_T$  (threshold voltage)

# Circuit Symbol MOSFET – N Channel



# MOSFET (Types)

- Four types:
  - n-channel enhancement mode
    - Most common since it is cheapest to manufacture
  - p-channel enhancement mode
  - n-channel depletion mode
  - p-channel depletion mode

# MOSFET characteristics

- Basically low voltage device. High voltage device are available up to 600V but with limited current. Can be paralleled quite easily for higher current capability.
- Internal (dynamic) resistance between drain and source during on state,  $R_{DS}(ON)$ , , limits the power handling capability of MOSFET. High losses especially for high voltage device due to  $R_{DS}(ON)$  .
- Dominant in high frequency application ( $>100\text{kHz}$ ). Biggest application is in switched-mode power supplies.

- The transistor consists of three regions, labeled the ``source'', the ``gate'' and the ``drain''.
- The area labeled as the gate region is actually a ``sandwich'' consisting of the underlying substrate material, which is a single crystal of semiconductor material (usually silicon); a thin insulating layer (usually silicon dioxide); and an upper metal layer.
- Electrical charge, or current, can flow from the source to the drain depending on the charge applied to the gate region.
- The semiconductor material in the source and drain region are ``doped'' with a different type of material than in the region under the gate, so an NPN or PNP type structure exists between the source and drain region of a MOSFET.

## 1. Cut-Off Region

Cut-off region is a region in which the MOSFET will be OFF as there will be no current flow through it. In this region, MOSFET behaves like an open switch and is thus used when they are required to function as electronic switches.

## 2. Ohmic or Linear Region

Ohmic or linear region is a region where the current  $I_{DS}$  increases with an increase in the value of  $V_{DS}$ . When MOSFETs are made to operate in this region, they can be used as amplifiers.

## 3. Saturation Region

In saturation region, the MOSFETs have their  $I_{DS}$  constant inspite of an increase in  $V_{DS}$  and occurs once  $V_{DS}$  exceeds the value of pinch-off voltage  $V_P$ . Under this condition, the device will act like a closed switch through which a saturated value of  $I_{DS}$  flows. As a result, this operating region is chosen whenever MOSFETs are required to perform switching operations.

## **Difference between enhancement and depletion type MOSFET**

Depletion mode MOSFET is normally turned on at zero gate voltage. Such devices are used as load resistors.

MOSFETs with enhancement modes are the common switching elements in most MOSs. These devices are deactivated at zero gate voltage and can be switched on by powering the gate.

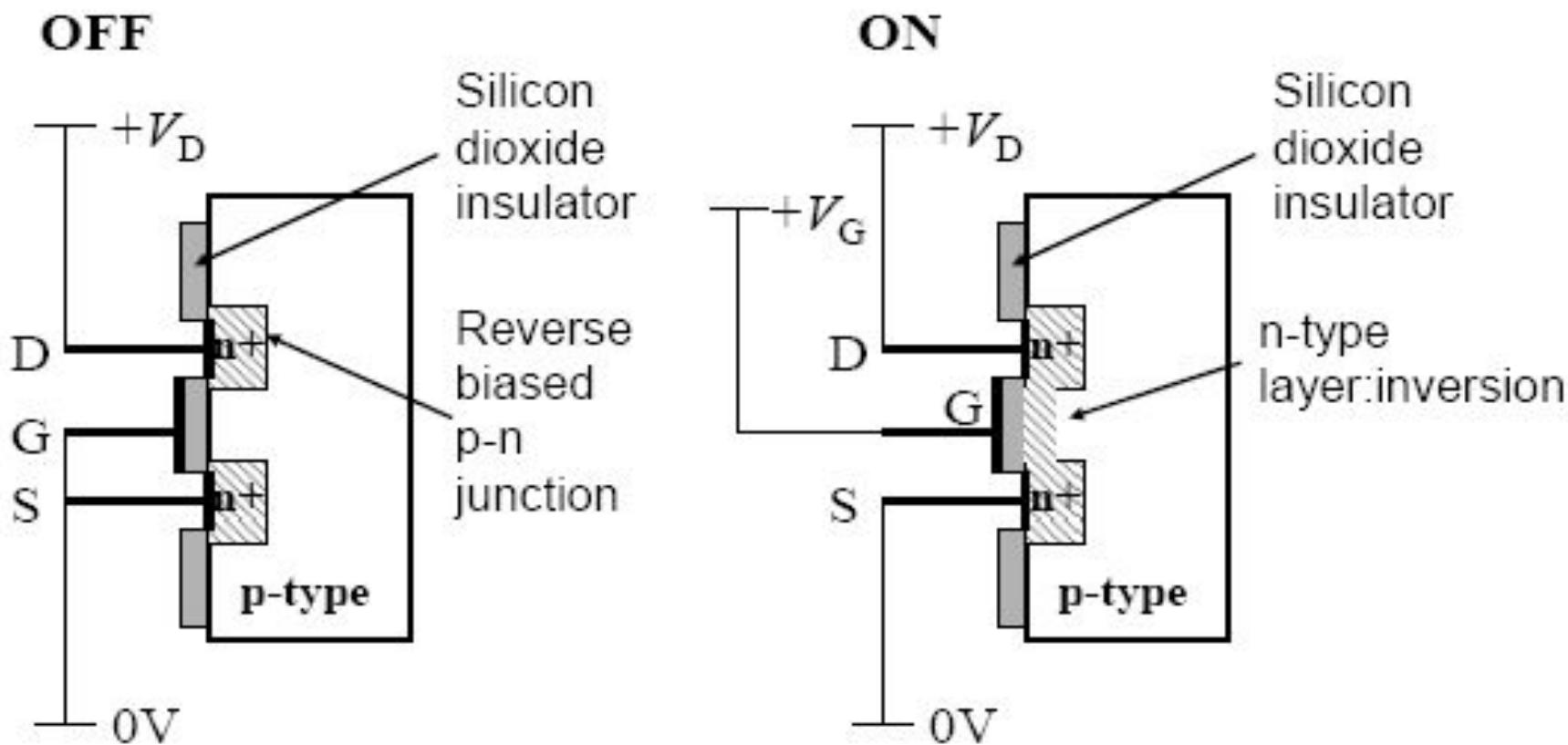
# Enhancement MOSFET

- MOSFETs with enhancement modes can be switched on by powering the gate either higher than the source voltage for NMOS or lower than the source voltage for the PMOS.
- In most circuits, this means that pulling a MOSFET gate voltage into the leakage boost mode becomes ON.
- For N-type discharging devices, the threshold voltage could be about -3 V, so it could be stopped by dragging the 3 V negative gate (leakage by comparison is more positive than the NMOS source).
- In PMOS, polarities are reversed.
- The mode can be determined by the voltage threshold sign (gate voltage versus source voltage at the point where only a layer inversion is formed in the channel):
  - For a N-type FET, modulation devices have positive and depleted thresholds – modulated devices have negative thresholds;
  - For a P-type FET, positive mode to improve negative mode, depletion.

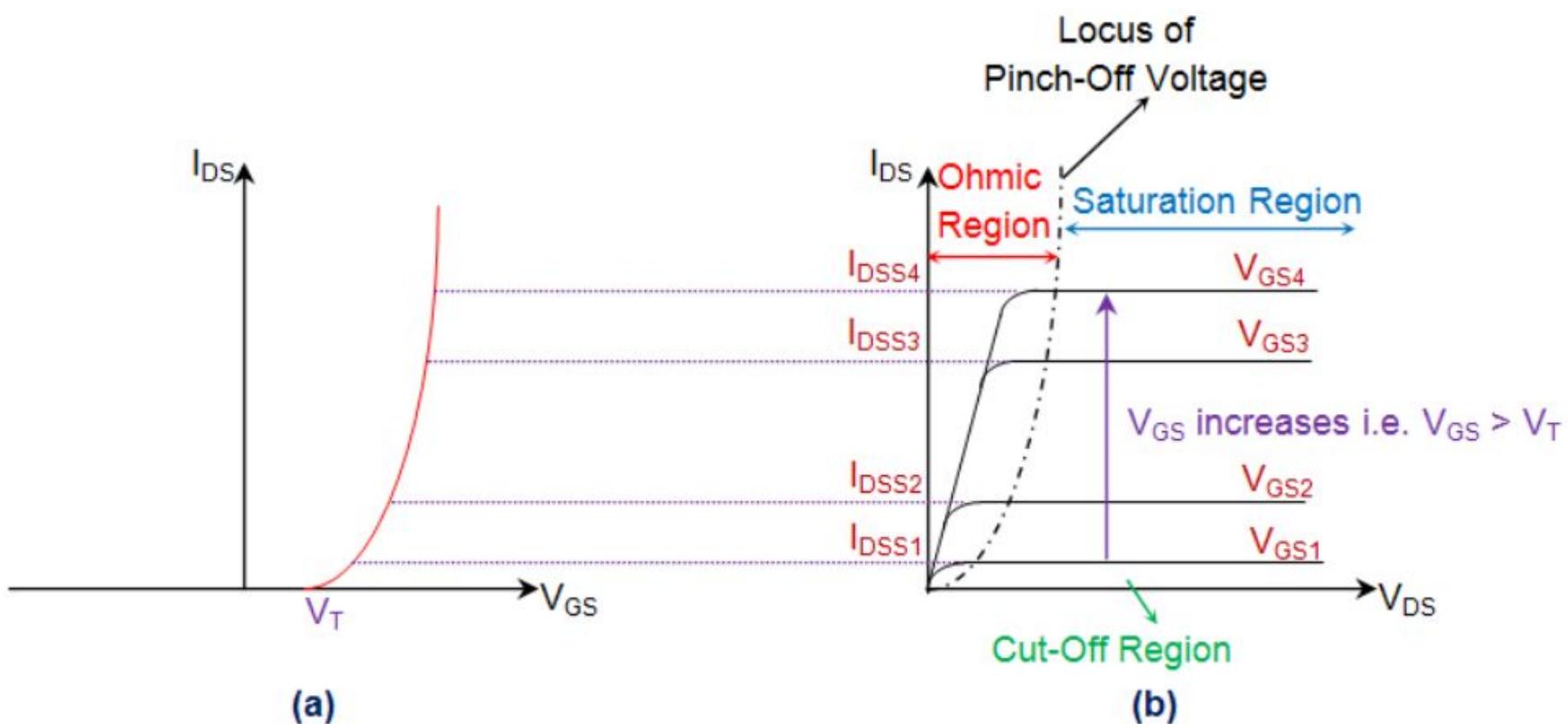
# Depletion type MOSFET

- Junction-effect junction transistors are the depletion mode because the gate junction would transmit the bias if the gate was taken more than a bit from the source to the drain voltage.
- Such devices are used in gallium-arsenide and germanium chips, where it is difficult to make an oxide isolator.
- Figure describes the construction of MOSFET type of exhaustion. Also note the MOSFET circuit type N exhaust channel symbol.
- Due to its construction, it offers very high entry strength (approximately 10<sup>10</sup> to 10<sup>15</sup>). Significant current flows for  $V_{DS}$  data at 0 volts  $V_{GS}$ .
- When the gate (ie, a capacitor plate) is made positive, the channel (i.e., the other capacitor plate) will have a positive charge induced therein.
- This will lead to the depletion of the major bearers (ie electrons) and therefore to the reduction in conductivity.

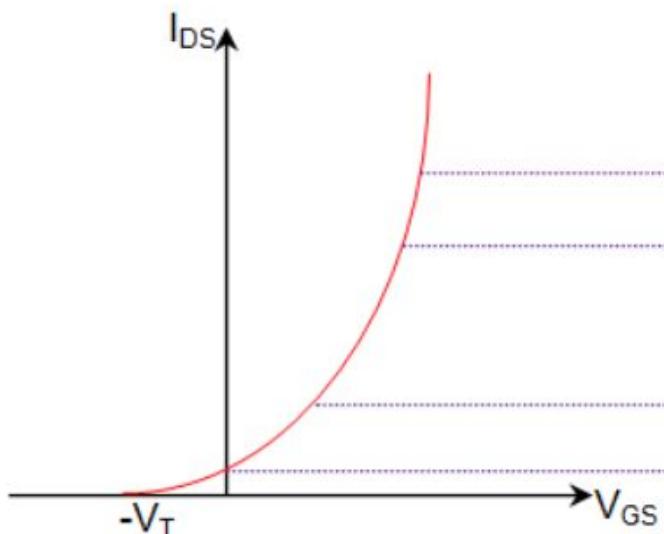
# n-Channel MOSFET



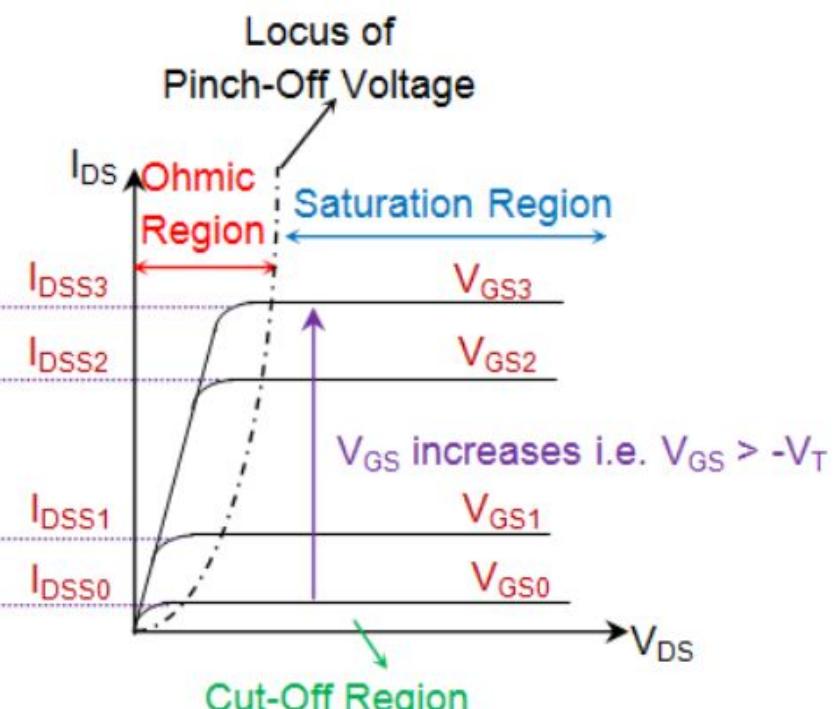
**n-Channel Enhancement type MOSFET (a) Transfer Characteristics (b) Output Characteristics**



**n-Channel Depletion type MOSFET (a) Transfer Characteristics (b) Output Characteristics**

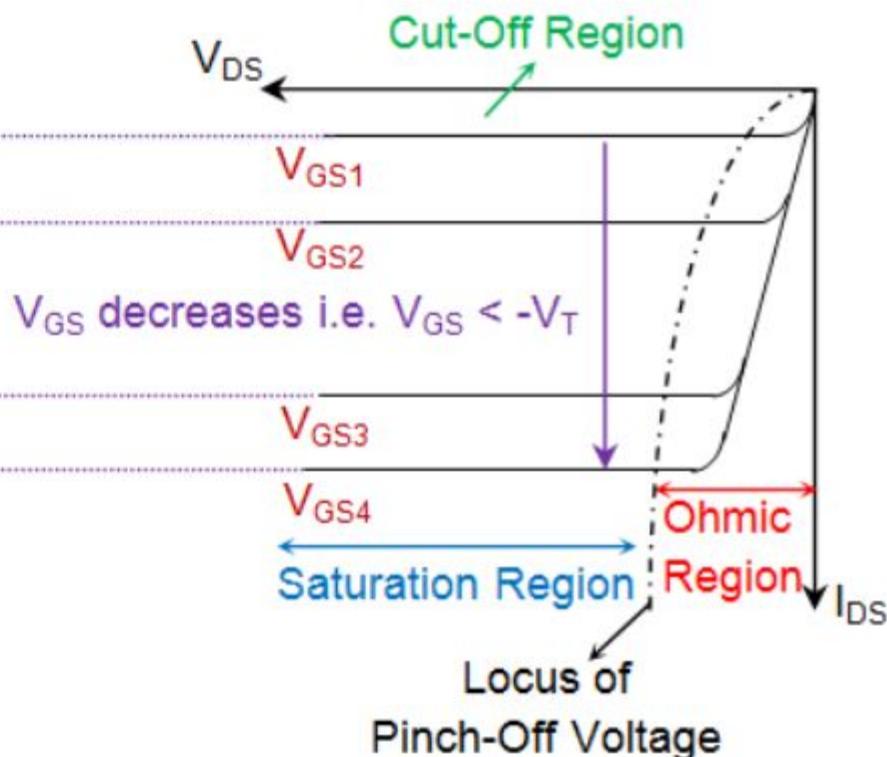
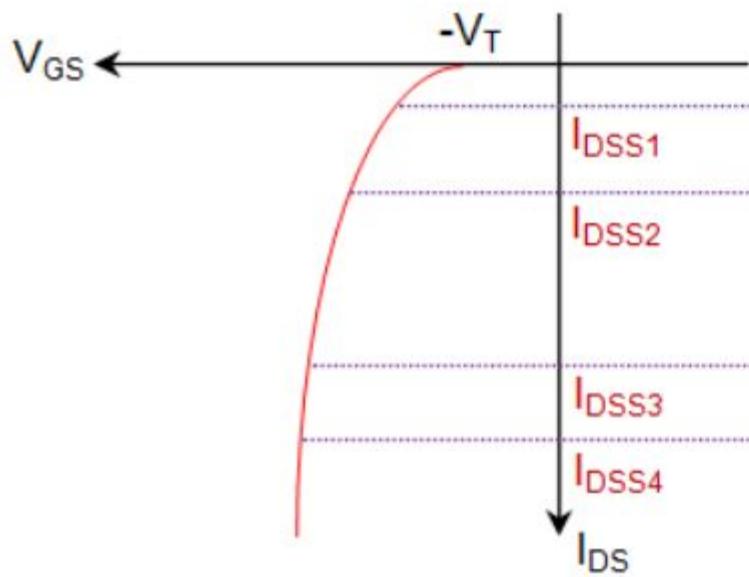


(a)



(b)

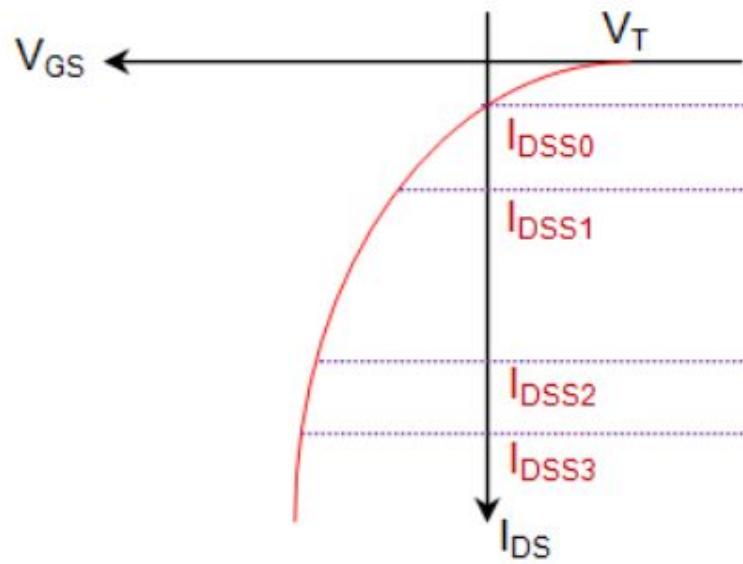
**p-Channel Enhancement type MOSFET (a) Transfer Characteristics (b) Output Characteristics**



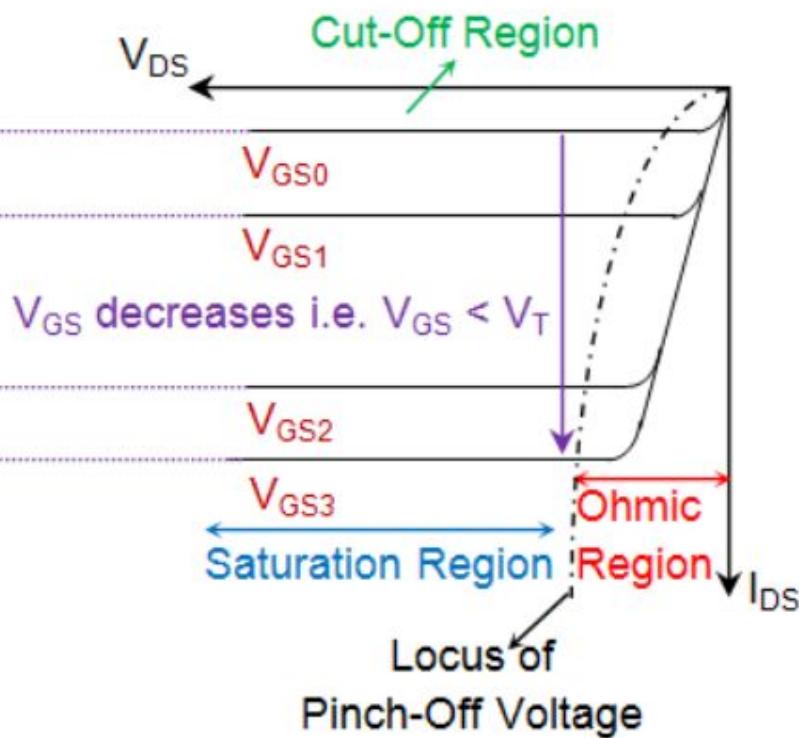
(a)

(b)

**p-Channel Depletion type MOSFET (a) Transfer Characteristics (b) Output Characteristics**



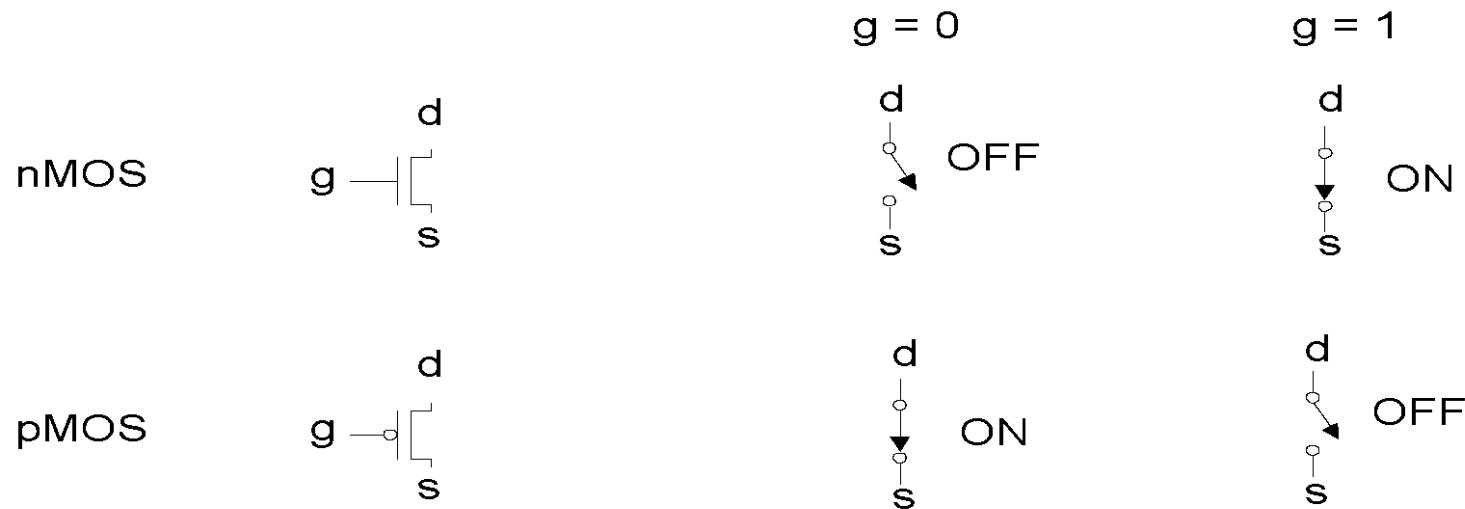
(a)



(b)

# Switch models of MOSFETs

$V_i$ ( $V_{gs}$ )	Logic Level	nMOS	pMOS
$V_{dd}$	1	ON	OFF
$V_{ss}$ (GND)	0	OFF	ON



# MOSFET - Uses

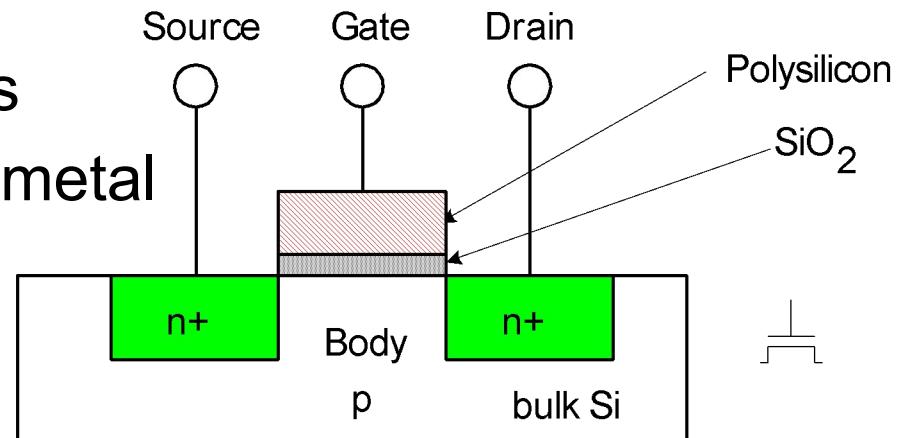
- **Most important device in digital design**
- **Very good as a switch**
- **Relatively few parasitics**
- **Rather low power consumption**
- **High integration density**
- **Simple manufacturing**
- **Economical for large complex circuits**

# Unit 2

S-7	SLO-1	PMOS,NMOS
	SLO-2	CMOS Logic

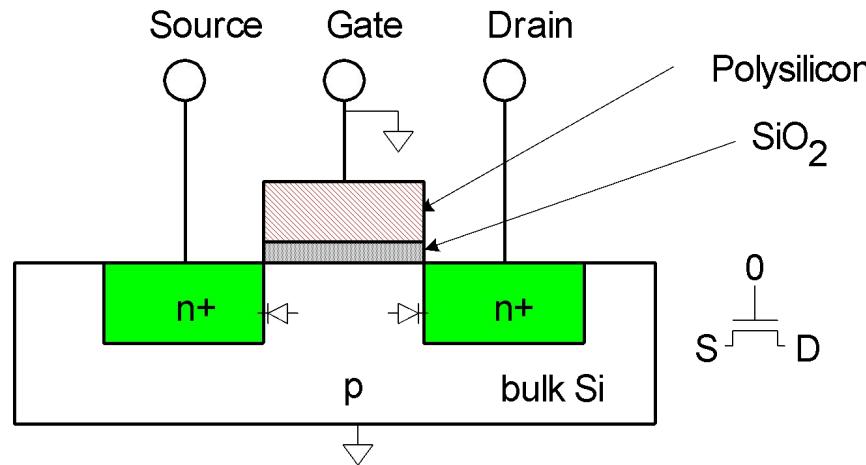
# nMOS Transistor

- Four terminals: gate, source, drain, body
- Gate – oxide – body stack looks like a capacitor
  - Gate and body are conductors
  - $\text{SiO}_2$  (oxide) is a very good insulator
  - Called metal – oxide – semiconductor (MOS) capacitor
  - Even though gate is no longer made of metal



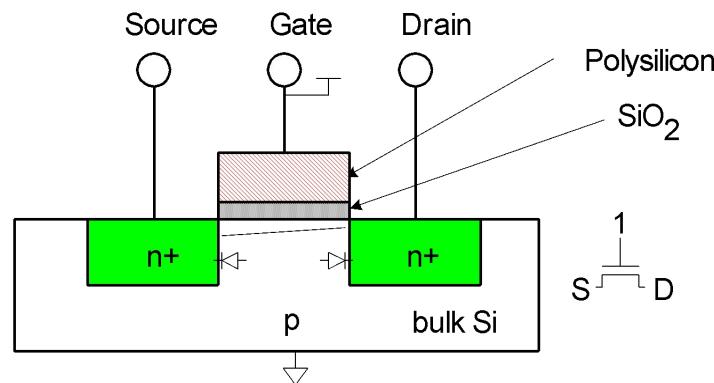
# nMOS Operation

- Body is usually tied to ground (0 V)
- When the gate is at a low voltage:
  - P-type body is at low voltage
  - Source-body and drain-body diodes are OFF
  - No current flows, transistor is OFF



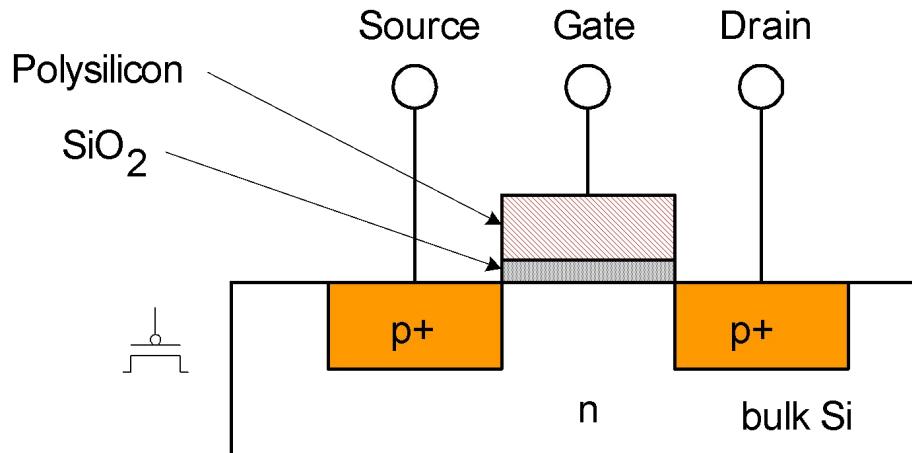
# nMOS Operation Cont.

- When the gate is at a high voltage:
  - Positive charge on gate of MOS capacitor
  - Negative charge attracted to body
  - Inverts a channel under gate to n-type
  - Now current can flow through n-type silicon from source through channel to drain, transistor is ON



# pMOS Transistor

- Similar, but doping and voltages reversed
  - Body tied to high voltage ( $V_{DD}$ )
  - Gate low: transistor ON
  - Gate high: transistor OFF
  - Bubble indicates inverted behavior

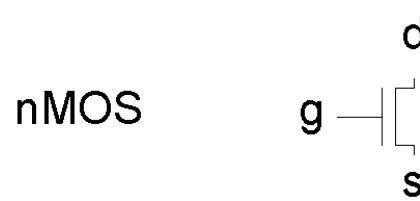


# Power Supply Voltage

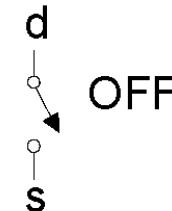
- GND = 0 V
- In 1980's,  $V_{DD} = 5V$
- $V_{DD}$  has decreased in modern processes
  - High  $V_{DD}$  would damage modern tiny transistors
  - Lower  $V_{DD}$  saves power
- $V_{DD} = 3.3, 2.5, 1.8, 1.5, 1.2, 1.0, \dots$

# Transistors as Switches

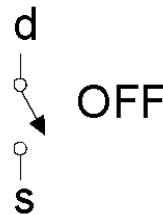
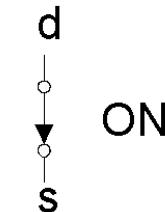
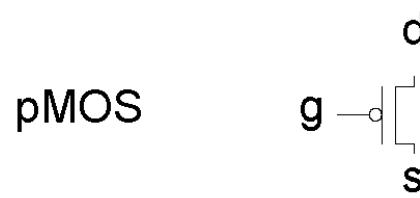
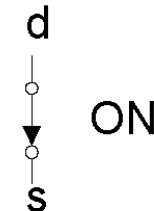
- We can view MOS transistors as electrically controlled switches
- Voltage at gate controls path from source to drain



$g = 0$

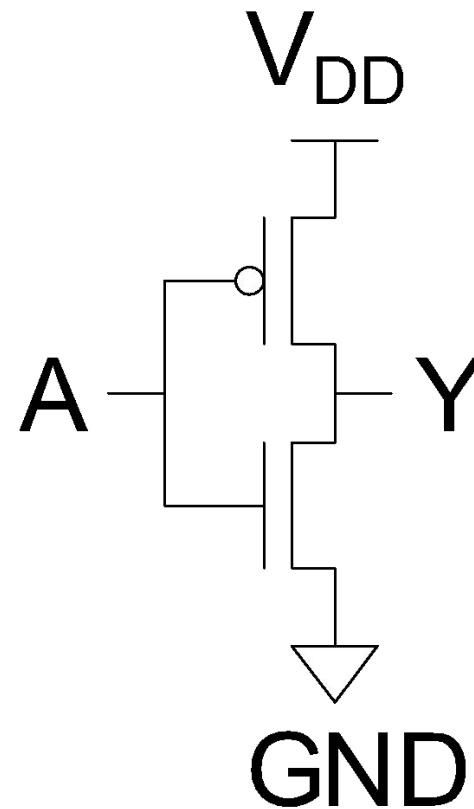
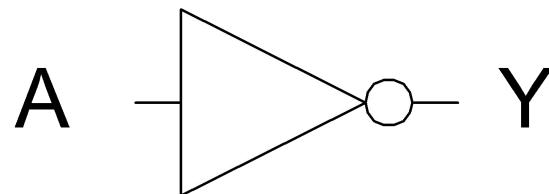


$g = 1$



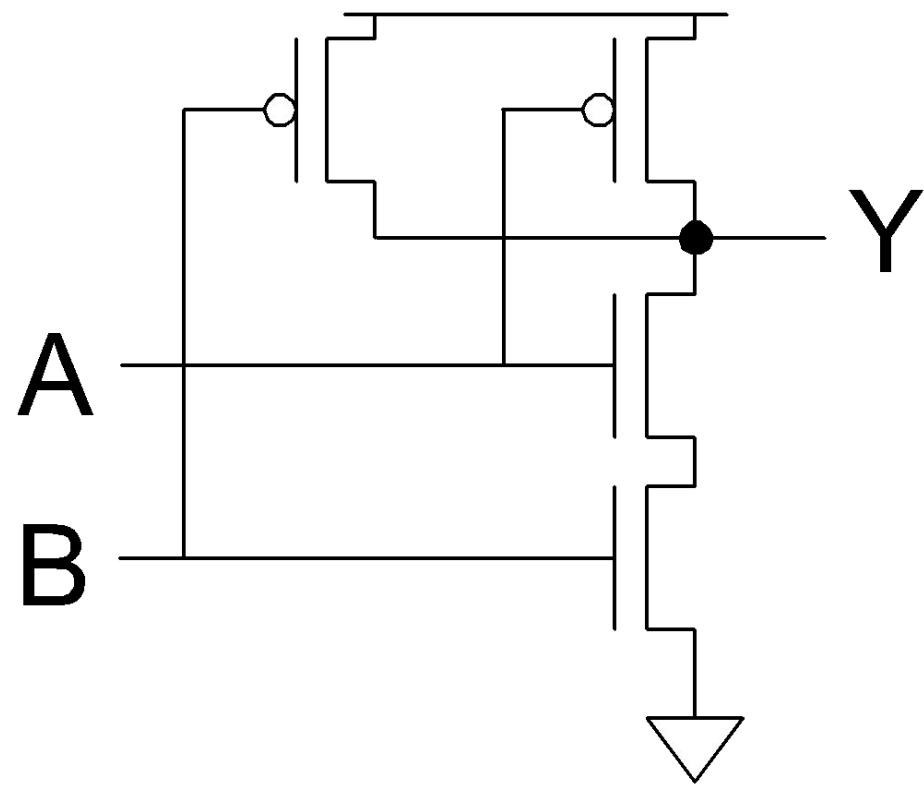
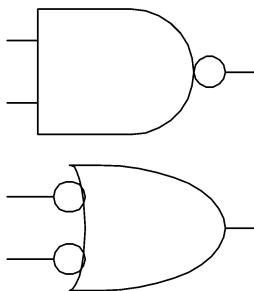
# CMOS Inverter

A	Y
0	1
1	0



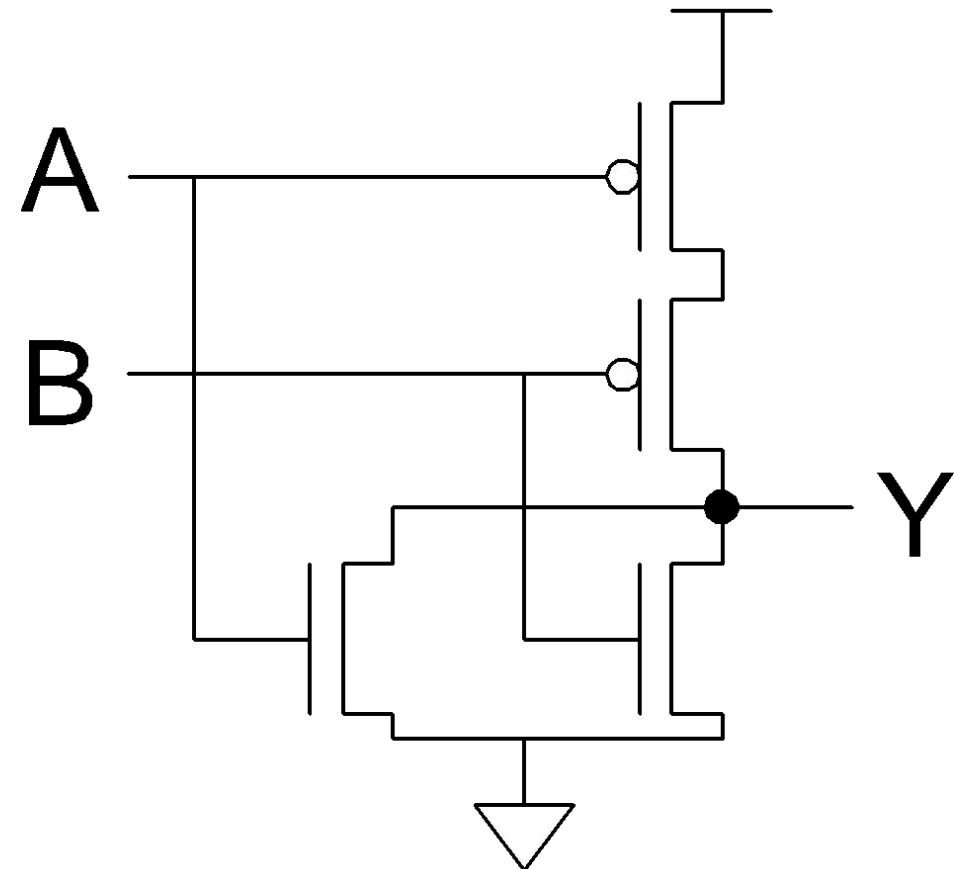
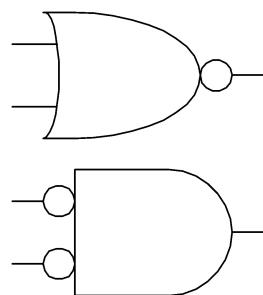
# CMOS NAND Gate

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



# CMOS NOR Gate

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

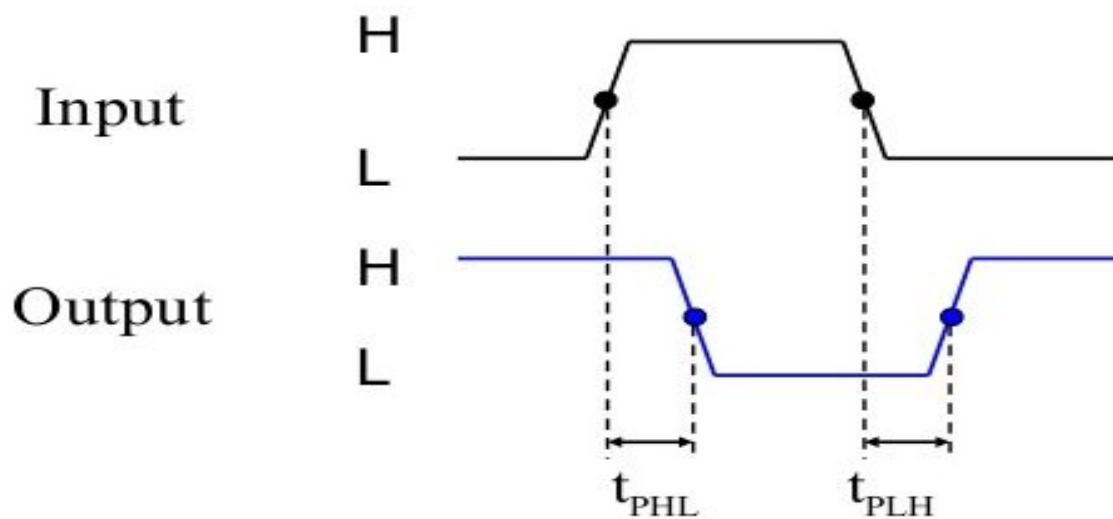
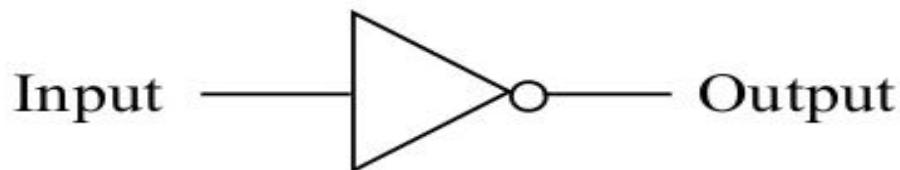


# **PROPAGATION DELAY, TRISTATE LOGIC & IT'S APPLICATIONS**

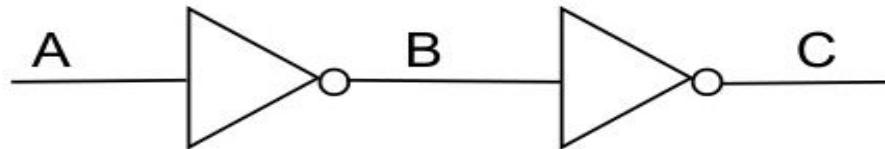
## **PROPAGATION DELAY**

- Every logic gate experiences some delay (though very small) in propagating signals forward. This delay is called Gate (Propagation) Delay( $t_{PD}$ ).
- Formally, it is the average transition time taken for the output signal of the gate to change in response to changes in the input signals.
- Three different propagation delay times associated with a logic gate:
  - $t_{PHL}$ : output changing from the High level to Low level
  - $t_{PLH}$ : output changing from the Low level to High level
  - $t_{PD} = (t_{PLH} + t_{PHL})/2$  (average propagation delay)

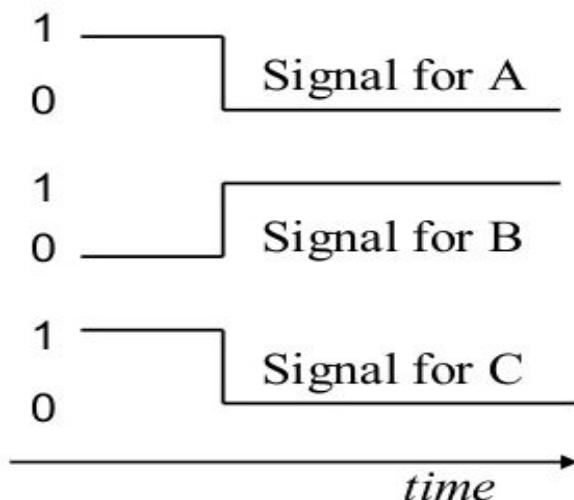
# PROPAGATION DELAY



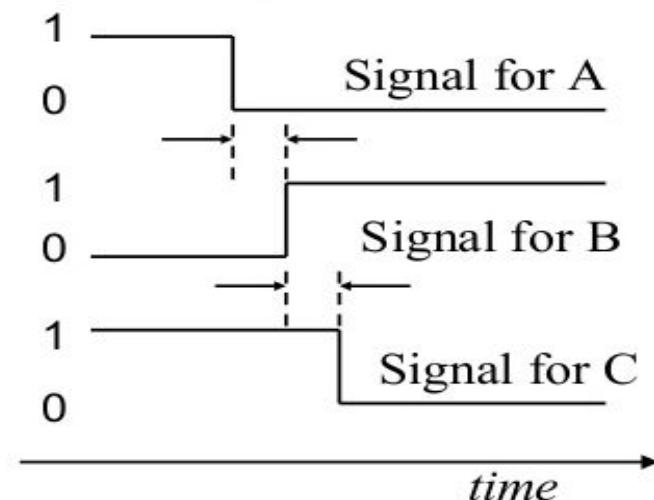
# PROPAGATION DELAY



- Ideally no delay:



- In reality, output signals normally lag behind input signals:



# **PROPAGATION DELAY**

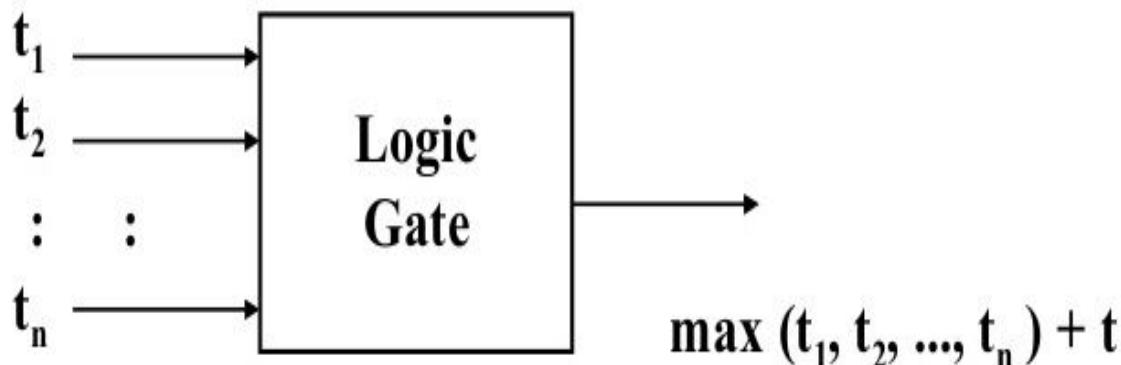
## **Calculation of Circuit Delays:**

- **Amount of propagation delay per gate depends on:**
  - (i) gate type (AND, OR, NOT, etc)
  - (ii) transistor technology used (TTL, ECL, CMOS etc),
  - (iii) miniaturization (SSI, MSI, LSI, VLSI)
- **To simplify matters, one can assume**
  - (i) an average delay time per gate, or
  - (ii) an average delay time per gate-type.
- **Propagation delay of logic circuit**
  - = longest time it takes for the input signal(s) to propagate to the output(s)
  - = earliest time for output signal(s) to stabilize, given that input signals are stable at time 0

# PROPAGATION DELAY

## Calculation of Circuit Delays:

- In general, given a logic gate with delay,  $t$



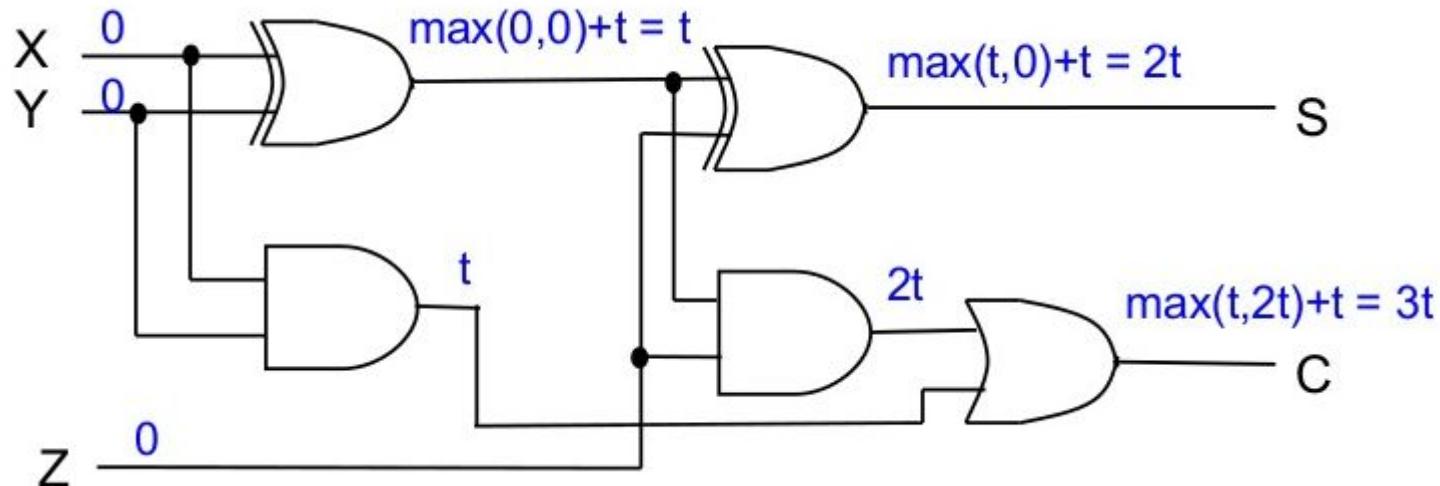
- If inputs are stable at times  $t_1, t_2, \dots, t_n$ , respectively; then the earliest time in which the output will be stable is:

$$\max(t_1, t_2, \dots, t_n) + t$$

# PROPAGATION DELAY

## Calculation of Circuit Delays:

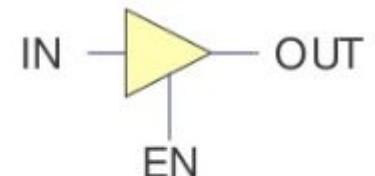
- As a simple example, consider the full adder circuit where all inputs are available at time 0. (Assume each gate has delay  $t$ .)



- where outputs S and C, experience delays of  $2t$  and  $3t$ , respectively.

# Tristate (Three-State) Logic

- Tristate output combines the advantages of the IN OUT totem-pole and open collector circuits
- Three output states are HIGH, LOW, and high EN impedance (Hi-Z)
- For the symbol and truth table, IN is the data input and enable input, and EN, the additional enable input for control
- For EN = 0, regardless of the value on IN (denoted by X), the output value is Hi-Z
- For EN = 1, the output value follows the input value
- Variations:
  - Data input, IN, can be inverted
  - EN is to make output Hi-Z or Control input, EN, can be inverted by addition of "bubbles" to signals



**Two Inputs:**  
**Input & Enable**

**Truth Table**

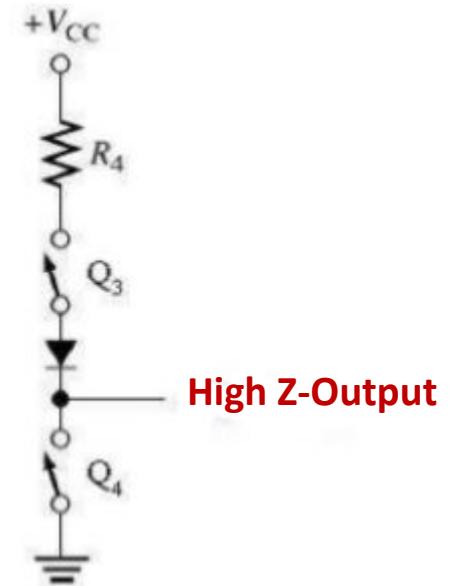
EN	IN	OUT
0	X	HI-Z
1	0	0
1	1	1

**EN is to make output Hi-Z or follow input**

# Tristate (Three-State) Logic

## Hi-Impedance Outputs

- Tristate gate utilize the high-speed operation of the totem-pole arrangement when input enabled
- Permit outputs to be connected together
- What is a Hi-Z value?
  - Both transistor are turned off in the totem- pole arrangement
  - This means that, looking back into the circuit, the output appears to be disconnected



An equivalent circuit (open circuit) for the tristate output in the high-Z state

# Tristate (Three-State) Logic

## Applications:

- Three-state buffers can be used to implement efficient multiplexers, especially those with large numbers of inputs.
- Three-state buffers are essential to the operation of a shared electronic bus.
- Three-state logic can reduce the number of wires needed to drive a set of LEDs (tri-state multiplexing or Charlieplexing).

# Unit - II

Hardware Description Language

# Hardware Description Language - Introduction

- 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 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.
- The most common use of a HDL is to provide an alternative to schematics.

# HDL – Introduction

- When a language is used for the above purpose (i.e. to provide an alternative to schematics), it is referred to as a *structural description* in which the language describes an interconnection of components.
- Such a structural description can be used as input to logic simulation just as a schematic is used.
- Models for each of the primitive components are required.
- If an HDL is used, then these models can also be written in the HDL providing a more uniform, portable representation for simulation input.

# HDL – Introduction

- HDL can be used to represent logic diagrams, Boolean expressions, and other more complex digital circuits.
- Thus, in top down design, a very high-level description of a entire system can be precisely specified using an HDL.
- This high-level description can then be refined and partitioned into lower-level descriptions as a part of the design process.

# HDL – Introduction

- As a documentation language, HDL is used to represent and document digital systems in a form that can be read by both humans and computers and is suitable as an exchange language between designers.
- The language content can be stored and retrieved easily and processed by computer software in an efficient manner.
- There are two applications of HDL processing: ***Simulation*** and ***Synthesis***

# Logic Simulation

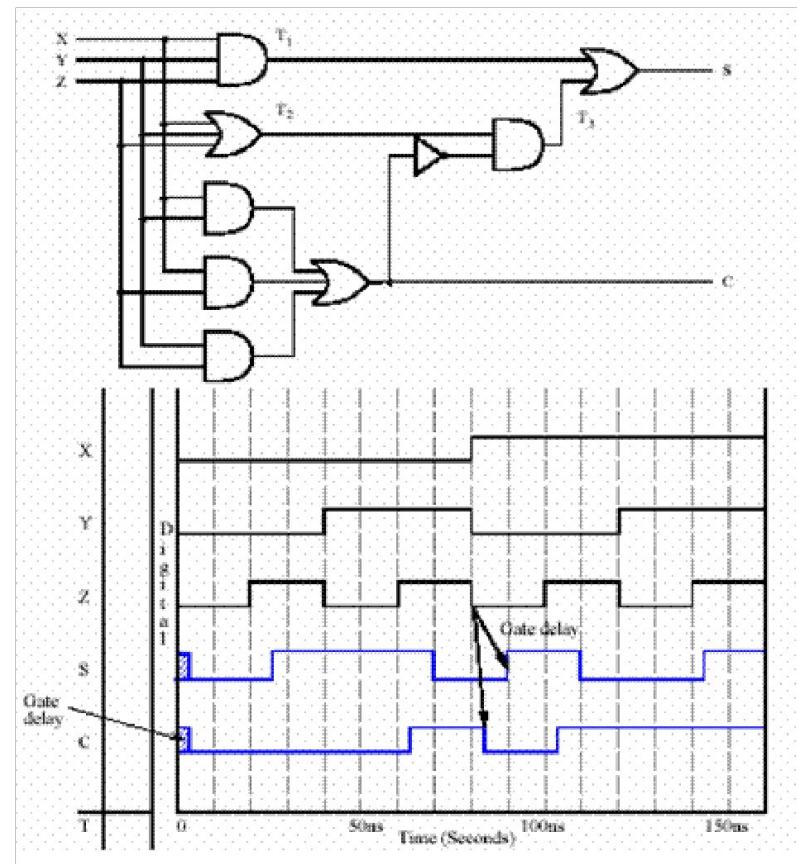
- A simulator interprets the HDL description and produces a readable output, such as a timing diagram, that predicts how the hardware will behave before its is actually fabricated.
- Simulation allows the detection of functional errors in a design without having to physically create the circuit.

# Logic Simulation

- The stimulus that tests the functionality of the design is called a test bench.
- To simulate a digital system
  - Design is first described in HDL
  - Verified by simulating the design and checking it with a test bench which is also written in HDL.

# Logic Simulation

- Logic simulation is a fast, accurate method of analyzing a circuit to see its waveforms



# Types of HDL

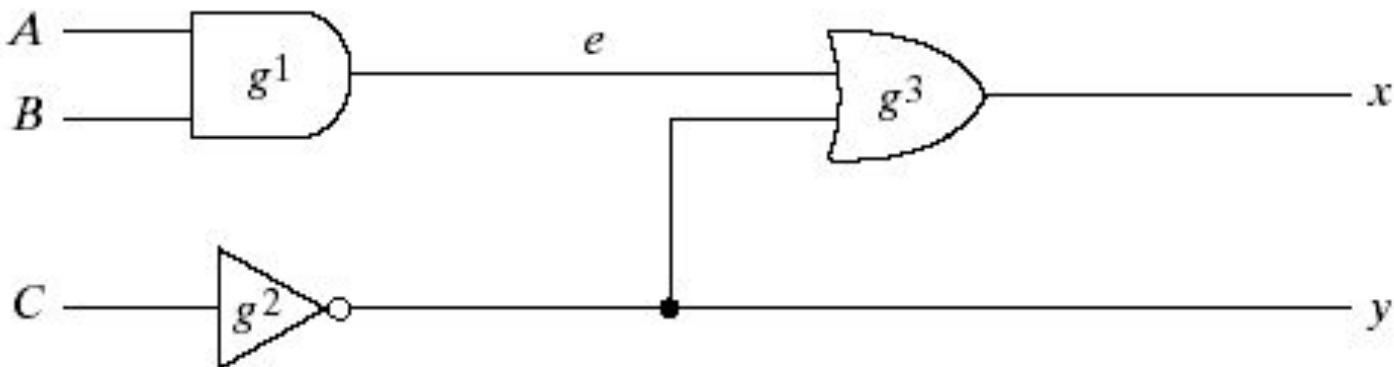
- There are two standard HDL's that are supported by IEEE.
  - **VHDL** (*Very-High-Speed Integrated Circuits Hardware Description Language*) - Sometimes referred to as VHSIC HDL, this was developed from an initiative by US. Dept. of Defense.
  - **Verilog HDL** – developed by Cadence Data systems and later transferred to a consortium called *Open Verilog International* (OVI).

# 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.

# Verilog Module

- 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*.



# Verilog Module for simple circuit

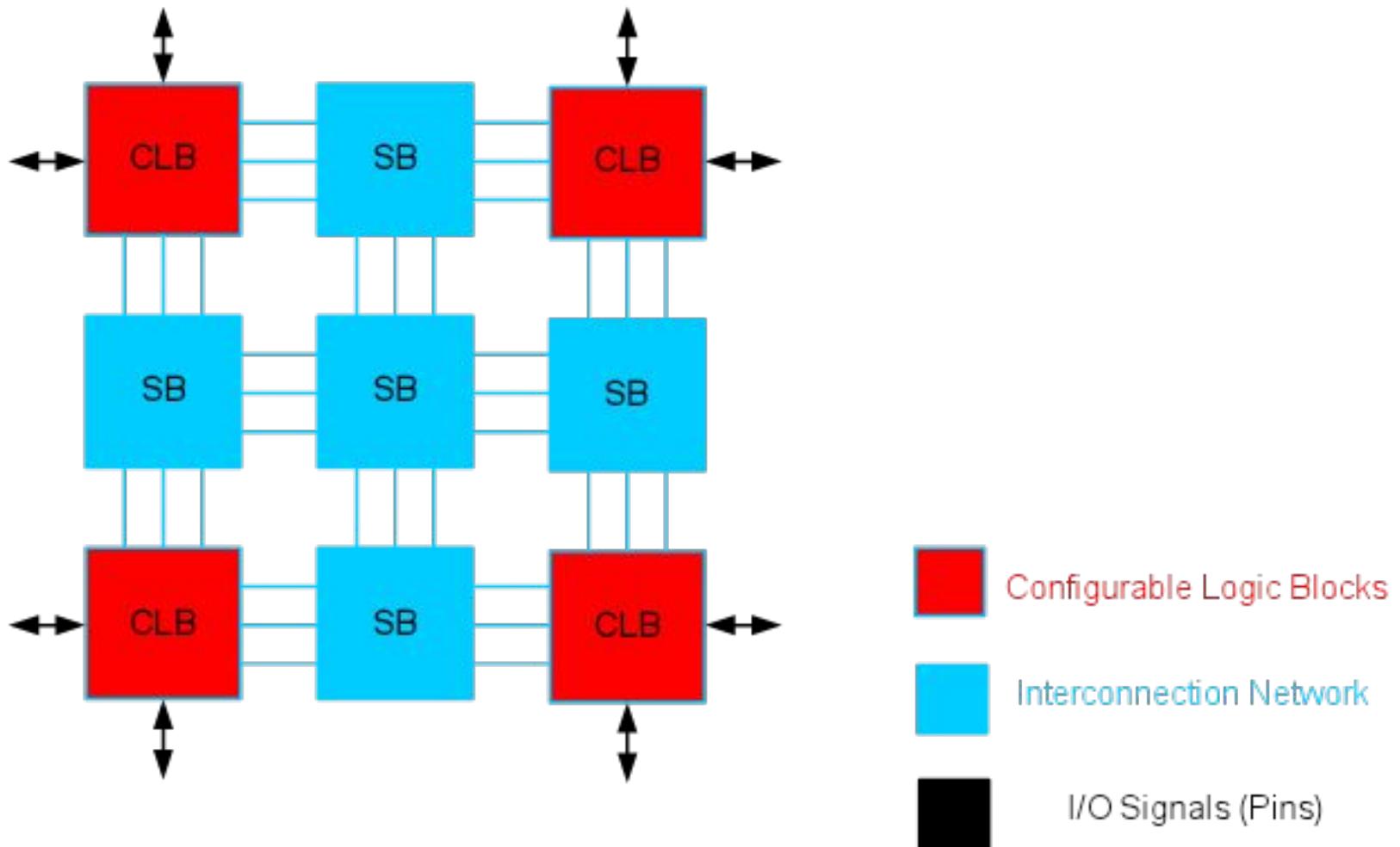
## HDL Example

```
module smpl_circuit(A,B,C,x,y);  
    input A,B,C;  
    output x,y;  
    wire e;  
    and g1(e,A,B);  
    not g2(y,C);  
    or g3(x,e,y);  
endmodule
```

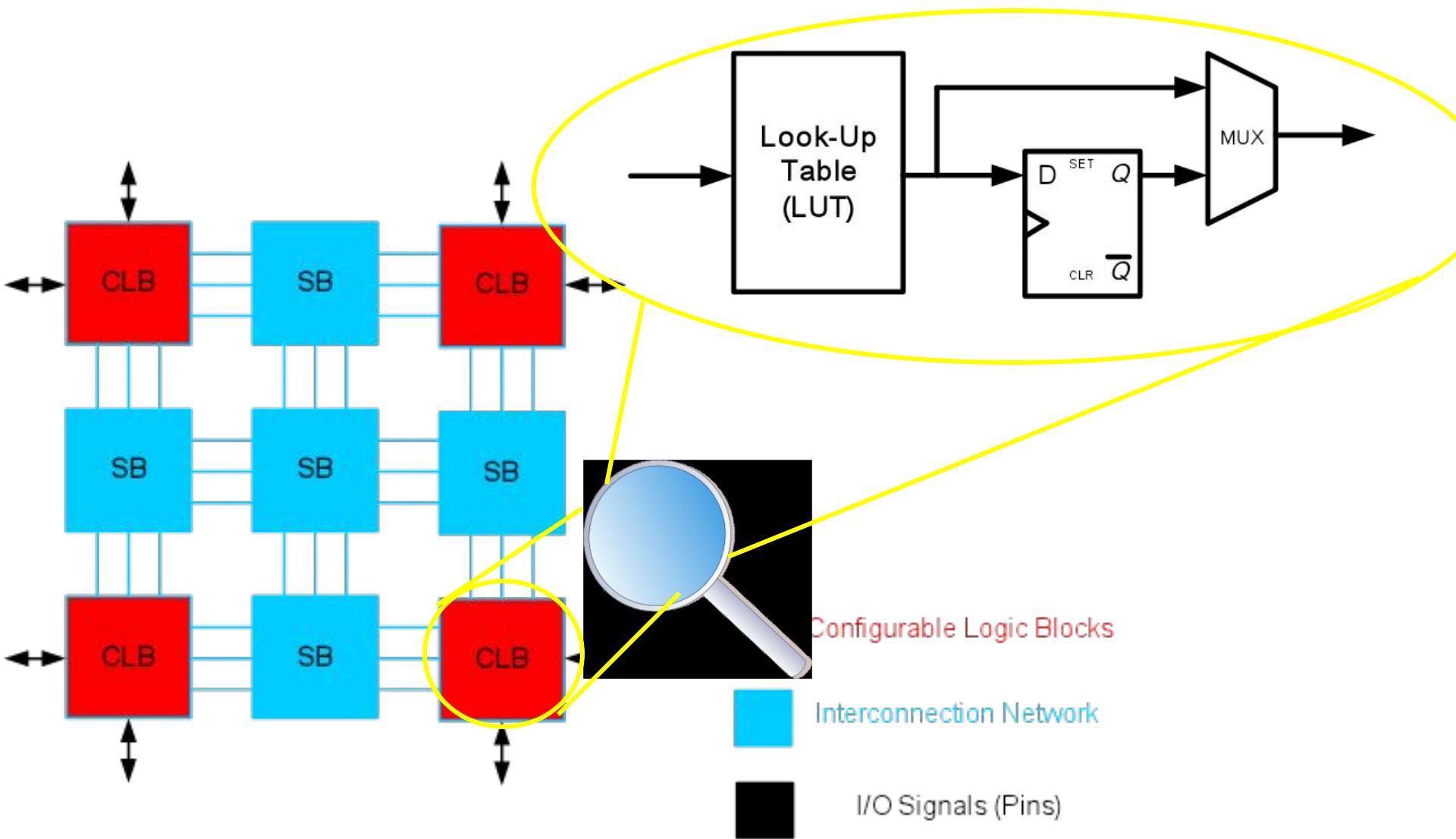
# FPGA

- A Field-Programmable Gate Array (FPGA) is an integrated circuit that can be configured by the user to emulate any digital circuit as long as there are enough resources.
- The FPGA configuration is generally specified using a hardware description language (HDL)
- An FPGA can be seen as an array of Configurable Logic Blocks (CLBs) connected through programmable interconnect (Switch Boxes).

# FPGA structure



# Simplified CLB Structure



# HDL user defined primitives

## Verilog HDL: User-Defined Primitives(UDP)

### User-Defined Primitives(UDP):

- One of the advanced concepts in verilog
- non synthesizable
- single output many input
- consume very less memory
- I/Os must be scalar (i.e. bit)
- 'z' value is not supported.

```
primitive myprimitive(c,a,b)
  input a,b;
  output c;
  table
    0 0:0
    0 1:0
    1 0:0
    1 1:0
  endtable
endprimitive
```

# HDL user defined primitives

*User Defined Primitives define new primitives.*

## Syntax:

```
primitive UDP_name (output, input, ...);  
    port_declaration  
    [ reg output; ]  
    [ initial output = initial_value; ]  
    table  
        truth_table  
    endtable  
endprimitive
```

# HDL user defined primitives

## UDP Rules :

- 1) UDPs can take only scalar input terminals (1 bit). Multiple input terminals are permitted.
- 2) UDPs can have only one scalar output terminal (1 bit). The output terminal must always appear first in the terminal list. Multiple output terminals are not allowed.
- 3) In the declarations section, the output terminal is declared with the keyword output. Since sequential UDPs store state, the output terminal must also be declared as a reg.
- 4) The inputs are declared with the keyword input.
- 5) The state in a sequential UDP can be initialized with an ‘initial’ statement. This statement is optional. A 1-bit value is assigned to the output, which is declared as reg.
- 6) The state table entries can contain values 0,1,or x . UDPs do not handle z values. Z values passed to a UDP are treated x values.
- 7) UDPs are defined at the same level as modules. UDPs can not be defined inside modules. They can be instantiated only inside modules. UDPs are instantiated exactly like gate primitives.
- 8) UDPs do not support inout ports.

# HDL user defined primitives

Combinational UDPs :

Eg :

```
primitive udp_and (out,a,b);
    output out;
    input a,b;
    table
        //a      b : out
        -----
        0      0 : 0;
        0      1 : 0;
        1      0 : 0;
        1      1 : 1;

    endtable
endprimitive
```

# HDL user defined primitives

## Sequential UDPs :

- >the output of a sequential UDP is always declared as a reg.
- >an initial statement can be used to initialize output of sequential UDPs.
- >the format of a state table using entry is slightly different

```
.....[ ] ::;
```

- >if a sequential UDP is sensitive to input levels, it is called a level sensitive sequential UDP.
- >if a sequential UDP is sensitive to edge transition on inputs, it is called an edge-sensitive sequential UDP.

# Sequential Level Sensitive UDP's example

Eg :

```
//Edge triggered T-FF
```

```
primitive T-FF (output reg q,input clk,clear);
```

```
//no initialization of q; T-FF will be initialized with clear signal.
```

```
table
```

	//clk	clear : q : q+ ;	
?	1	: ? : 0 ;	//asynchronous clear condition
?	(10)	: ? : - ;	//ignore -ve edge of clear
(10)	0	: 1 : 0 ;	//toggle FF at -ve edge of clk
(10)	0	: 0 : 1 ;	
(0?)	0	: ?: - ;	//ignore +ve edge of clock

```
endtable
```

```
endprimitive
```

# Stimulus to the design

- Checking the response of a design
- Verifying output – using listings and waveforms
- Signal Sampling
- Automating output verification
- Golden vectors and output timing
- Deadlock and deadlock recovery
- Predicting the output

# VERIFYING OUTPUT

- Generating stimulus is only half of the job. Actually, it is about only 30% of the job. You control the value and timing of the stimulus.
- Knowing the value and timing of the output given an input, or a sequence of inputs, is key to performing verification
- Verifying that the output is as expected
  - is more time consuming
  - is more error prone
- Specific points in time that are significant differ from design to design

# Unit 3

Combinational Logic Circuits

Quine-McCluskey minimization technique  
Introduction to Combinational Circuits

Multiplexer  
Demultiplexer

Decoder  
Encoder

Binary adder  
Binary adder as subtractor

Carry look ahead adder  
Decimal adder

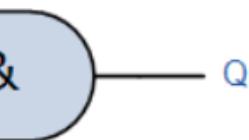
Magnitude Comparator  
Problem solving session

Read –only Memory  
Arithmetic Logic Unit

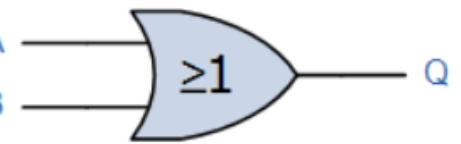
Programmable Logic Arrays  
HDL Gate and Data Flow modeling  
HDL Behavioral modeling

# Basic Logic Gates

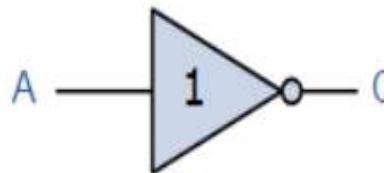
## The Logic AND Gate

Symbol	Truth Table		
	A	B	Q
 2-input AND Gate	0	0	0
	0	1	0
	1	0	0
	1	1	1
Boolean Expression Q = A.B	Read as A AND B gives Q		

## The Logic OR Gate

Symbol	Truth Table		
	A	B	Q
 2-input OR Gate	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression Q = A+B	Read as A OR B gives Q		

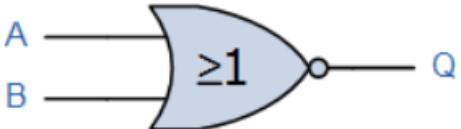
## The Logic NOT Gate

Symbol	Truth Table	
	A	Q
	0	1
	1	0
Boolean Expression $Q = \text{NOT } A$ or $\bar{A}$	Read as inversion of A gives Q	

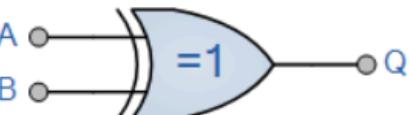
## The Logic NAND Gate

Symbol	Truth Table		
 2-input NAND Gate	A	B	Q
	0	0	1
	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = \overline{A \cdot B}$	Read as A AND B gives NOT-Q		

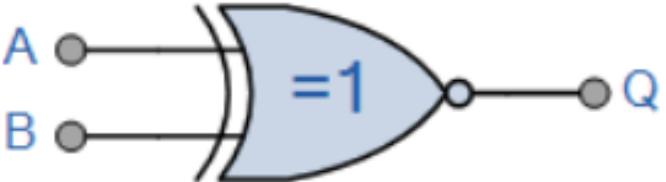
## The Logic NOR Gate

Symbol	Truth Table		
	A	B	Q
	0	0	1
2-input NOR Gate	0	1	0
	1	0	0
	1	1	0
Boolean Expression $Q = \overline{A+B}$	Read as A OR B gives NOT-Q		

## The Logic EXOR Gate

Symbol	Truth Table		
	B	A	Q
	0	0	0
2-input EXOR Gate	0	1	1
	1	0	1
	1	1	0
Boolean Expression $Q = A \oplus B$	Read as A OR B but not BOTH gives Q (odd)		

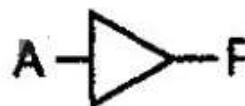
## The Logic EXNOR Gate

Symbol	Truth Table		
	B	A	Q
	0	0	1
	0	1	0
	1	0	0
	1	1	1
Boolean Expression $Q = \overline{A} \oplus B$	Read if A AND B the SAME gives Q (even)		

# Consolidation of Logic Gates

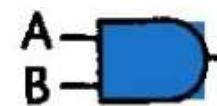
Inputs		Truth Table Outputs For Each Gate						
A	B	AND	NAND	OR	NOR	EX-OR	EX-NOR	
0	0	0	1	0	1	0	1	
0	1	0	1	1	0	1	0	
1	0	0	1	1	0	1	0	
1	1	1	0	1	0	0	1	

Logic Function	Boolean Notation
AND	$A \cdot B$
OR	$A + B$
NOT	$\bar{A}$
NAND	$\overline{A \cdot B}$
NOR	$\overline{A + B}$
EX-OR	$(A \cdot \bar{B}) + (\bar{A} \cdot B)$ or $A \oplus B$
EX-NOR	$(A \cdot B) + (\bar{A} \cdot \bar{B})$ or $\overline{A \oplus B}$



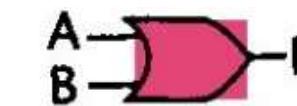
Buffer  
 $F = A$

A	F
0	0
1	1



AND  
 $F = AB$

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



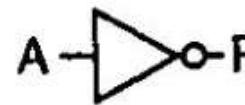
OR  
 $F = A+B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



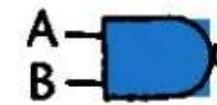
XOR  
 $F = A \oplus B$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



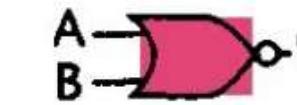
Inverter  
 $F = \bar{A}$

A	F
0	1
1	0



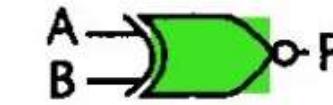
NAND  
 $F = \overline{AB}$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



NOR  
 $F = \overline{A+B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



XNOR  
 $F = \overline{A \oplus B}$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

# Boolean Rules and Laws

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

# **Sum of Product (SOP) and Product of Sum (POS)**

# Minterm and Maxterm

A **minterm** is defined as the product term of n variables, in which each of the n variables will appear once either in its complemented or un-complemented form.

A **maxterm** is defined as the sum term of n variables, in which each of the n variables will appear once either in its complemented or un-complemented form.

Variables			Min terms	Max terms
A	B	C	$m_i$	$M_i$
0	0	0	$A' B' C' = m_0$	$A + B + C = M_0$
0	0	1	$A' B' C = m_1$	$A + B + C' = M_1$
0	1	0	$A' B C' = m_2$	$A + B' + C = M_2$
0	1	1	$A' B C = m_3$	$A + B' + C' = M_3$
1	0	0	$A B' C' = m_4$	$A' + B + C = M_4$
1	0	1	$A B' C = m_5$	$A' + B + C' = M_5$
1	1	0	$A B C' = m_6$	$A' + B' + C = M_6$
1	1	1	$A B C = m_7$	$A' + B' + C' = M_7$

# Sum of Product (SOP) Form

The **Sum of Product form** is a form of expression in Boolean algebra in which different product terms of inputs are being summed together.

Example:

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

$$F = \sum(m_1, m_2, m_3, m_5)$$

$$F = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C$$

Canonical or  
Standard SOP form

# Product of Sum (POS) Form

The **Product of Sum form** is a form in which products of different sum terms of inputs are taken.

Example:

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

$$F = \prod (M_0, M_4, M_6, M_7)$$

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

Canonical or  
Standard POS form

# Conversion from Minimal SOP to Canonical SOP Form

Example:

$$F = \bar{A}B + \bar{B}C$$

The term  $\bar{A}B$  is missing input C. So we will multiply  $\bar{A}B$  with  $(C + \bar{C})$  because  $(C + \bar{C} = 1)$ .  
The term  $\bar{B}C$  is missing input A, so it will be multiplied with  $(A + \bar{A})$

Missing input  
variables need to  
be multiplied

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

$$F = \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$$

# Conversion from Minimal POS to Canonical POS Form

Example:

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

( $\bar{A} + \bar{B}$ ) term is missing C input so we will add ( $C\bar{C}$ ) with it.  
( $B + C$ ) term is missing A input so we will add ( $A\bar{A}$ ) with it.

Missing input  
variables need to  
be added

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

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

# Conversion of SOP to POS

To convert the SOP form into POS form, first we should change the  $\Sigma$  to  $\Pi$  and then write the numeric indexes of missing variables of the given Boolean function.

**Example:**

$$F = \sum_{A, B, C} (0, 2, 3, 5, 7) = A' B' C' + A B' C' + A B' C + ABC' + ABC$$

Note: writing the missing indexes of the terms i.e., 1 - 001, 4 - 100 and 6 - 110.

$$001 = (A + B + C'), \quad 100 = (A' + B + C), \quad 110 = (A' + B' + C)$$

Hence,

$$F = \prod_{A, B, C} (1, 4, 6) = (A + B + C') * (A' + B + C) * (A' + B' + C)$$

# Conversion of POS to SOP

To convert the POS form into SOP form, first we should change the  $\Pi$  to  $\Sigma$  and then write the numeric indexes of missing variables of the given Boolean function.

**Example:**

$$F = \Pi_{A, B, C} (2, 3, 5) = (A + B' + C) * (A + B' + C') * (A' + B + C')$$

Note: writing the missing indexes of the terms, 0 - 000, 1 - 001, 4 - 100, 6 - 110, and 7 - 111.

000 =  $A' * B' * C'$ , 001 =  $A' * B' * C$ , 100 =  $A * B' * C'$ , 110 =  $A * B * C'$ , 111 =  $A * B * C$

Hence,

$$\begin{aligned} F = \Sigma_{A, B, C} (0, 1, 4, 6, 7) = & (A' * B' * C') + (A' * B' * C) + (A * B' * C') + \\ & (A * B * C') + (A * B * C) \end{aligned}$$

# Karnaugh Map (K-Map)

# K-map

- K-Map is used to minimize the number of logic gates by minimizing the logical expression.
- The minimization will reduce cost, complexity and power consumption.
- An n-variable K-map has  $2^n$  cells with each cell corresponding to an n-variable truth table value.
  - If 2 variable – 4 cells
  - If 3 variable – 8 cells
  - If 4 variable – 16 cells
- K-map cells are arranged such that adjacent cells correspond to truth rows that differ in only one bit position.

## 2 variables K-Map

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

0	1
2	3

## 3 variables K-Map

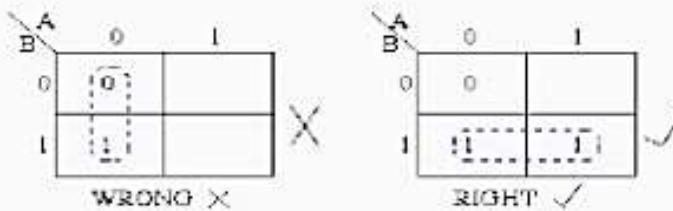
		00	01	11	10
		$y'z'$	$y'z$	$yz$	$yz'$
		$m_0$ $x'y'z'$ 0	$m_1$ $x'y'z$ 1	$m_3$ $x'yz$ 3	$m_2$ $x'yz'$ 2
0	$x'$	$m_0$ $x'y'z'$ 0	$m_1$ $x'y'z$ 1	$m_3$ $x'yz$ 3	$m_2$ $x'yz'$ 2
	$x$	$m_4$ $xy'z'$ 4	$m_5$ $xy'z$ 5	$m_7$ $xyz$ 7	$m_6$ $xyz'$ 6

# 4 variables K-Map

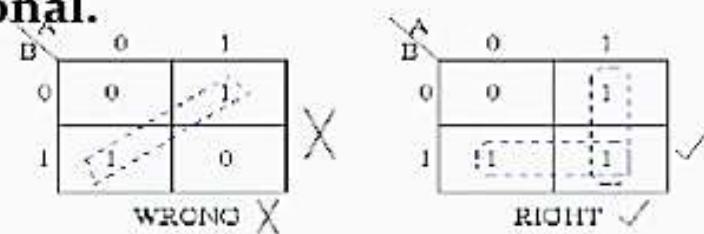
		0 0	0 1	1 1	1 0	
		$\bar{C}\bar{D}$	$\bar{C}D$	$C\bar{D}$	$CD$	
00		$\bar{A}\bar{B}$	0	1	3	2
01		$\bar{A}B$	4	5	7	6
11		$A\bar{B}$	12	13	15	14
10		$AB$	8	9	11	10

# K-map Rules

1. Groups may not include any cell containing a zero.



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



# K-map Rules

3. Groups must contain 1, 2, 4, 8, or in general  $2^n$  cells.

A B	0	1
0	(1)	(1)
1	0	0

Group of 2  
RIGHT ✓

A B C	00	01	11	10
0	0	(1)	(1)	(1)
1	0	0	0	0

Group of 3  
WRONG ✗

A B	0	1
0	(1)	(1)
1	(1)	(1)

Group of 4  
RIGHT ✓

A B C	00	01	11	10
0	(1)	(1)	(1)	(1)
1	0	0	0	1

Group of 5  
WRONG ✗

# K-map Rules

4. Each group should be as large as possible.

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

RIGHT ✓

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

WRONG ✗

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

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

◻ ◻

6. Groups may overlap.

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

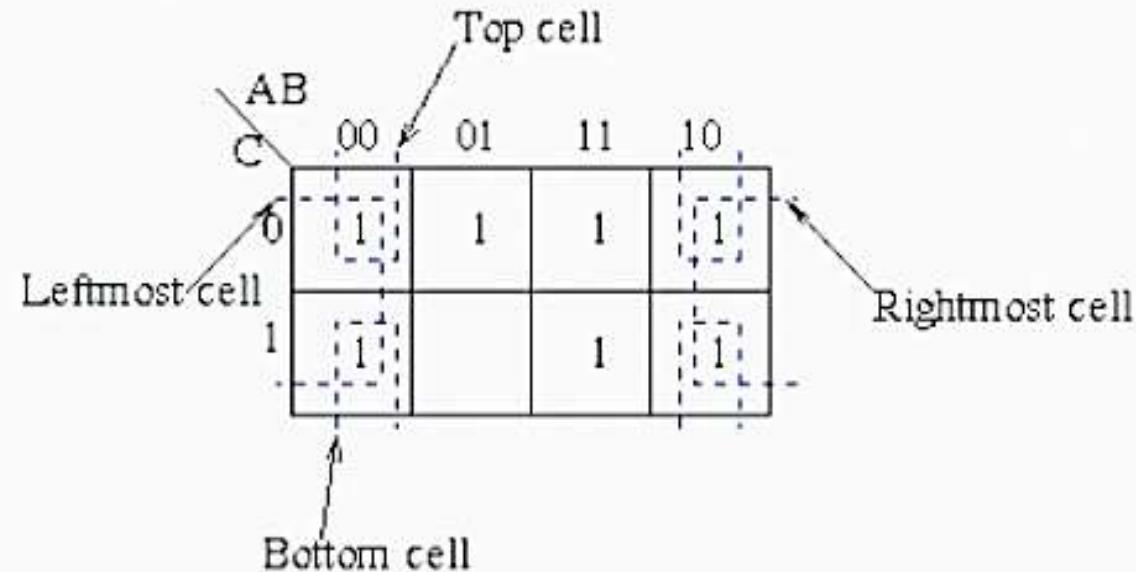
Groups overlapping.

✓

# K-map Rules

7. Groups may wrap 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.



# K-map Rules

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	0	(1)	1	
		1	0	0	(1)	1	
		1	0	0	(1)	1	

RIGHT ✓

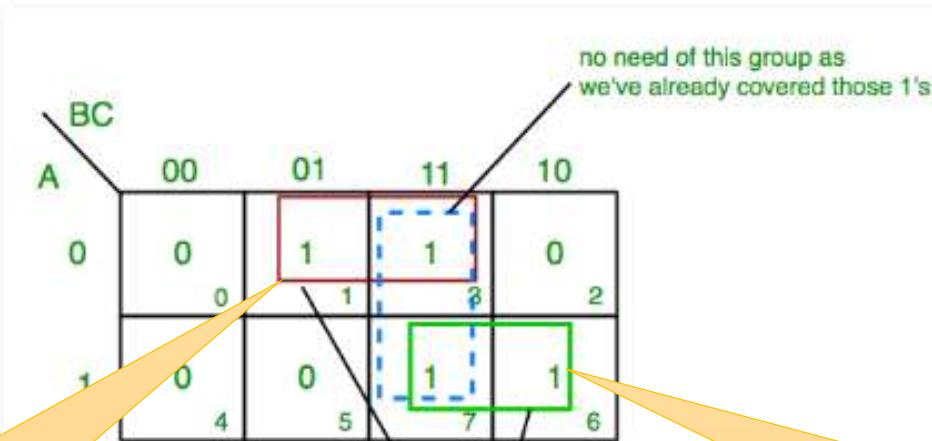
		AB	00	01	11	10	
		C	0	(1)	1	1	1
		0	0	0	(1)	1	
		1	0	0	(1)	1	
		1	0	0	(1)	1	

WRONG ×

# Example 1

For the given minterms, find the reduced logical expression using K-Map

$$Z = \sum_{A,B,C}(1,3,6,7)$$



From **red** group we  
get product term—  
 $A'C$

Groups of two elements  
in one group

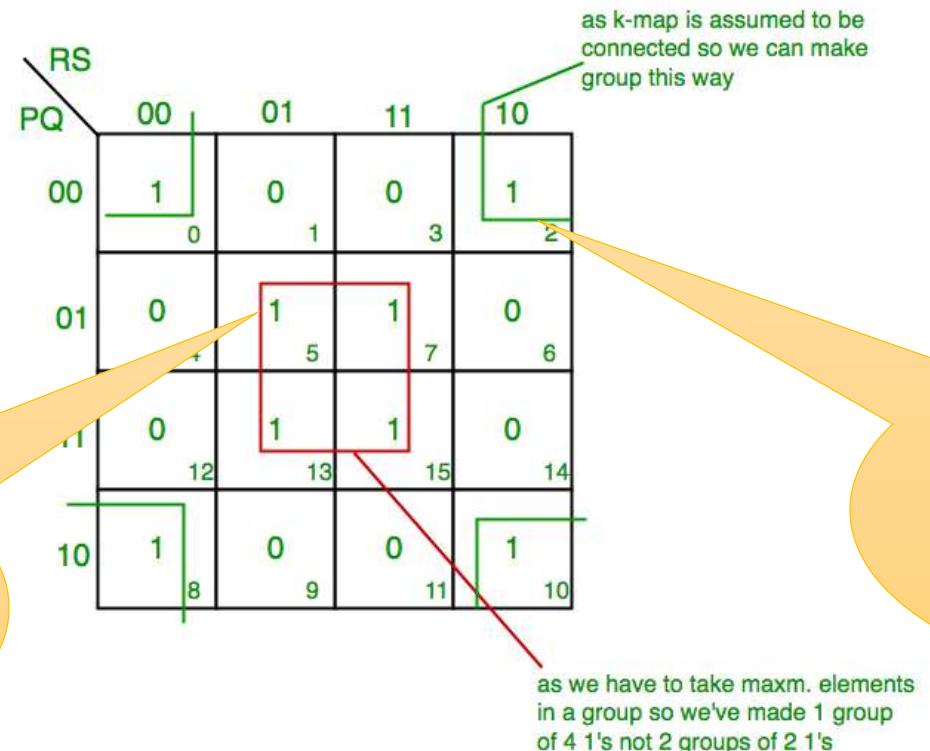
From **green** group we  
get product term—  
 $AB$

$$Z = A'C + AB$$

# Example 2

For the given minterms, find the reduced logical expression using K-Map

$$F(P,Q,R,S) = \sum(0,2,5,7,8,10,13,15)$$



$$F = QS + Q'S'$$

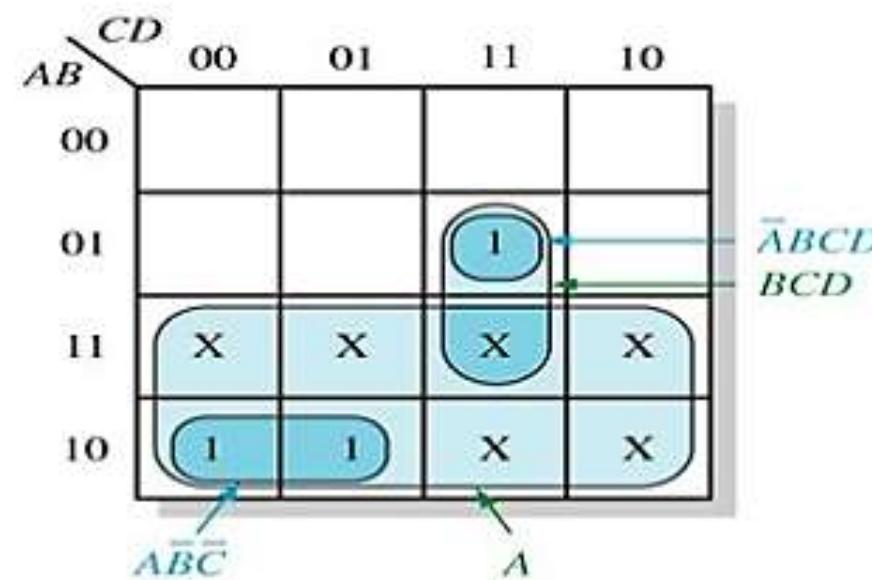
# K-map with Don't cares

- A “don’t care” condition is a combination of inputs for which the output may either ‘1’ or ‘0’.
- Hence “don’t care” conditions may either be included or excluded while grouping is made.
- It is usually denoted as ‘X’ in the K-Map table.

# Example 3

For the given minterms, find the reduced logical expression using K-Map

$$Z = \sum_{A,B,C}(7, 8, 9) + \sum_d(10, 11, 12, 13, 14, 15)$$



$$\begin{aligned} Z &= AB'C' + A'BCD \quad (\text{without don't cares}) \\ \checkmark \quad Z &= A + BCD \quad (\text{with don't cares}) \end{aligned}$$

# Quine-McCluskey Method (Tabulation Method)

- A systematic simplification procedure to reduce a minterm expansion to a minimum sum of products.
- Use  $XY + XY' = X$  to eliminate as many as literals as possible.
  - The resulting terms = prime implicants.
- Use a prime implicant chart to select a minimum set of prime implicants.

# Determination of Prime Implicants

✓ Eliminate literals

Two terms can be combined if they differ in exactly one variable.

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

$$\begin{array}{r} \underline{1 \ 0 \ 1 \ 0} \\ X \quad Y \end{array} + \begin{array}{r} \underline{1 \ 0 \ 1 \ 1} \\ X \quad Y' \end{array} = \begin{array}{r} \underline{1 \ 0 \ 1} \\ X \end{array}$$

$A'BC'D + A'BCD'$  (won't combine)

$0 \ 1 \ 0 \ 1 + 0 \ 1 \ 1 \ 0$  (check # of 1's)

We need to compare and combine whenever possible.

# Sorting to Reduce Comparisons

✓ Sort into groups according to the number

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

- No need for comparisons
  - (1) Terms in nonadjacent group
  - (2) Terms in the same group

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

# Comparison of adjacent groups

Use  $X + X = X$  repeatedly between adjacent groups

Those combined are checked off.

Combine terms that have the same dashes and differ one in the number of 1's. (for column II and column III)

Determination of Prime Implicants

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

# Prime Implicants

- The terms that have not been checked off are called prime implicants.

$$\begin{aligned} f &= 0\cdot 01 + 01\cdot 1 + 011\cdot - + -00\cdot \\ &\quad + -0\cdot 0 + --10 \\ &= \underline{a'c'd} + a'bd + \underline{a'bc} + b'c' + \\ &\quad \underline{b'd'} + cd' \end{aligned}$$

- Each term has a minimum number of literals, but minimum SOP for  $f$ :

$$\begin{aligned} f &= a'bd + b'c' + cd' \\ (a'bd, cd') &\Rightarrow a'bc \\ (a'bd, b'c') &\Rightarrow a'c'd \\ (b'c', cd') &\Rightarrow b'd' \end{aligned}$$

# Definition of Implicant

- Definition
  - Given a function of F of n variables, a product term P is an implicant of F iff for every combination of values of the n variables for which P = 1, F is also equal to 1.
    - Every minterm of F is an implicant of F.
    - Any term formed by combining two or more minterms is an implicant.
    - If F is written in SOP form, every product term is an implicant.
- Example:  $f(a,b,c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac$ 
  - If  $a'b'c' = 1$ , then  $F = 1$ , if  $ac = 1$ , then  $F = 1$ .  $a'b'c'$  and  $ac$  are implicants.
  - If  $bc = 1$ , (but  $a = 0$ ),  $F = 0$ , so  $bc$  is not an implicant of F.

# Definition of Prime Implicant

- Definition
  - A prime implicant of a function F is a product term implicant which is no longer an implicant if any literal is deleted from it.
  - Example:  $f(a,b,c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac$ 
    - Implicant  $a'b'c'$  is not a prime implicant. Why? If  $a'$  is deleted,  $b'c'$  is still an implicant of F.
    - $b'c'$  and  $ac$  are prime implicants.
  - Each prime implicant of a function has a minimum number of literals that no more literals can be eliminated from it or by combining it with other terms.

# Quine McClusky Procedure

- QM procedure:
  - Find all product term implicants of a function
  - Combine non-prime implicants.
    - Remaining terms are prime implicants.
  - A minimum SOP expression consists of a sum of some (not necessarily all) of the prime implicants of that function.
    - We need to select a minimum set of prime implicants.
  - If an SOP expression contains a term which is not a prime implicant, the SOP cannot be minimum

# Prime Implicant Chart

- Chart layout

- Top row lists minterms of the function
- All prime implicants are listed on the left side.
- Place x into the chart according to the minterms that form the corresponding prime implicant.

- Essential prime implicant

- If a minterm is covered only by one prime implicant, that prime implicant is called essential prime implicant. (9 & 14).
  - Essential prime implicant must be included in the minimum sum of the function.

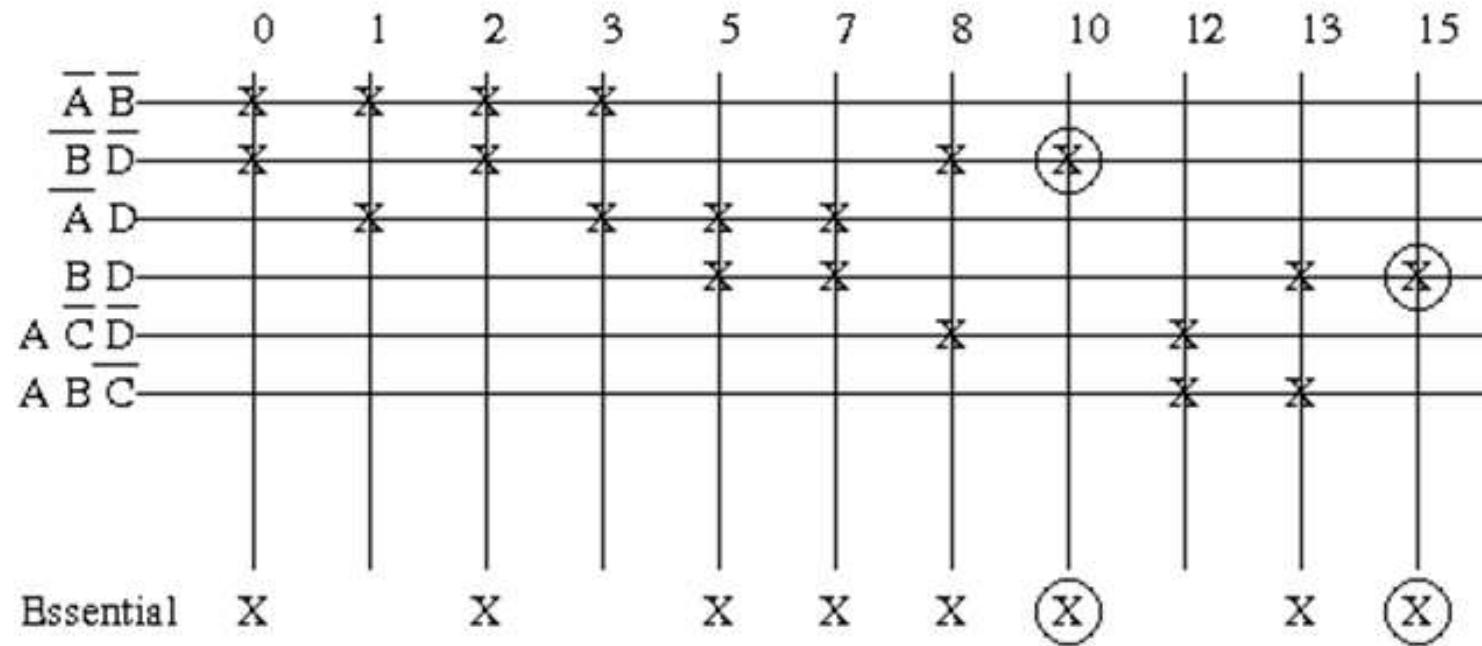
		0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	$b'c'$		X	X				X	(X)		
(0, 2, 8, 10)	$b'd'$	X		X			X		X	X	
(2, 6, 10, 14)	$cd'$			X	X				X	(X)	
(1, 5)	$a'c'd$		X	X							
(5, 7)	$a'bd$				X	X					
(6, 7)	$a'bc$						X	X			

First List				
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
8	1	0	0	0
3	0	0	1	1
5	0	1	0	1
10	1	0	1	0
12	1	1	0	0
7	0	1	1	1
13	1	1	1	1
15	1	1	1	1

Second List				
	A	B	C	D
0,1	0	0	0	—
0,2	0	0	—	0
0,8	—	0	0	0
1,3	0	0	—	1
1,5	0	—	0	1
2,3	0	0	1	—
2,10	—	0	1	0
8,10	1	0	—	0
8,12	1	—	0	0
3,7	0	—	1	1
5,7	0	1	—	1
5,13	—	1	0	1
12,13	1	1	0	—
7,15	—	1	1	1
13,15	1	1	—	1

Third List				
	A	B	C	D
0,1,2,3	0	0	—	—
0,2,1,3	0	0	—	—
0,2,8,10	—	0	—	0
0,8,2,10	—	0	—	0
1,3,5,7	0	—	—	1
1,5,3,7	0	—	—	1
5,7,13,15	—	1	—	1
5,13,7,15	—	1	—	1

The prime implicants are:  $\bar{A}\bar{B} + \bar{B}\bar{D} + \bar{A}D + BD + A\bar{C}\bar{D} + AB\bar{C}$



Thus, one minimal solution is:  $Z = \overline{B}\overline{D} + BD + \overline{A}\overline{B} + A\overline{C}\overline{D}$

$$F(A,B,C,D) = \Sigma m(2,6,8,9,10,1,14,15)$$

Group Name	Min terms	W	X	Y	Z	Group Name	Min terms	W	X	Y	Z
GA1	2	0	0	1	0	GB1	2,6	0	-	1	0
	8	1	0	0	0		2,10	-	0	1	0
GA2	6	0	1	1	0		8,9	1	0	0	-
	9	1	0	0	1		8,10	1	0	-	0
	10	1	0	1	0	GB2	6,14	-	1	1	0
GA3	11	1	0	1	1		9,11	1	0	-	1
	14	1	1	1	0		10,11	1	0	1	-
GA4	15	1	1	1	1		10,14	1	-	1	0
						GB3	11,15	1	-	1	1
							14,15	1	1	1	-

Group Name	Min terms	W	X	Y	Z
GB1	2,6,10,14	-	-	1	0
	2,10,6,14	-	-	1	0
	8,9,10,11	1	0	-	-
	8,10,9,11	1	0	-	-
GB2	10,11,14,15	1	-	1	-
	10,14,11,15	1	-	1	-

Group Name	Min terms	W	X	Y	Z
GC1	2,6,10,14	-	-	1	0
	8,9,10,11	1	0	-	-
GC2	10,11,14,15	1	-	1	-

	2	6	8	9	10	11	14	15
Min terms / Prime Implicants								
$YZ'$	1	1			1		1	
$WX'$			1	1	1	1		
$WY$					1	1	1	1

$$f(W, X, Y, Z) = YZ' + WX' + WY.$$

$$F(A,B,C,D) = \sum m(0,1,3,7,8,9,11,15)$$

Group	Minterms	Variables A B C D	Remark
0	0	0000	✓
1	1	0001	✓
	8	1000	✓
2	3	0011	✓
	9	1001	✓
3	7	0111	✓
	11	1011	✓
4	15	1111	✓

Group	Minterms	Variables A B C D	Remark
0	0,1	000-	✓
	0,8	-000	✓
1	1,3	00-1	✓
	1,9	-001	✓
	8,9	100-	✓
2	3,7	0-11	✓
	3,11	-011	✓
	9,11	10-1	✓
3	7,15	-111	✓
	11,15	1-11	✓

Group	Minterms	Variables ABCD	Remark
0	(0,1),(8,9)	-00-	✓
	(0,8),(1,9)	-00-	✓
1	(1,3),(9,11)	-0-1	✓
	(1,9),(3,11)	-0-1	✓
2	(3,7),(11,15)	--11	✓
	(3,11),(7,15)	--11	✓

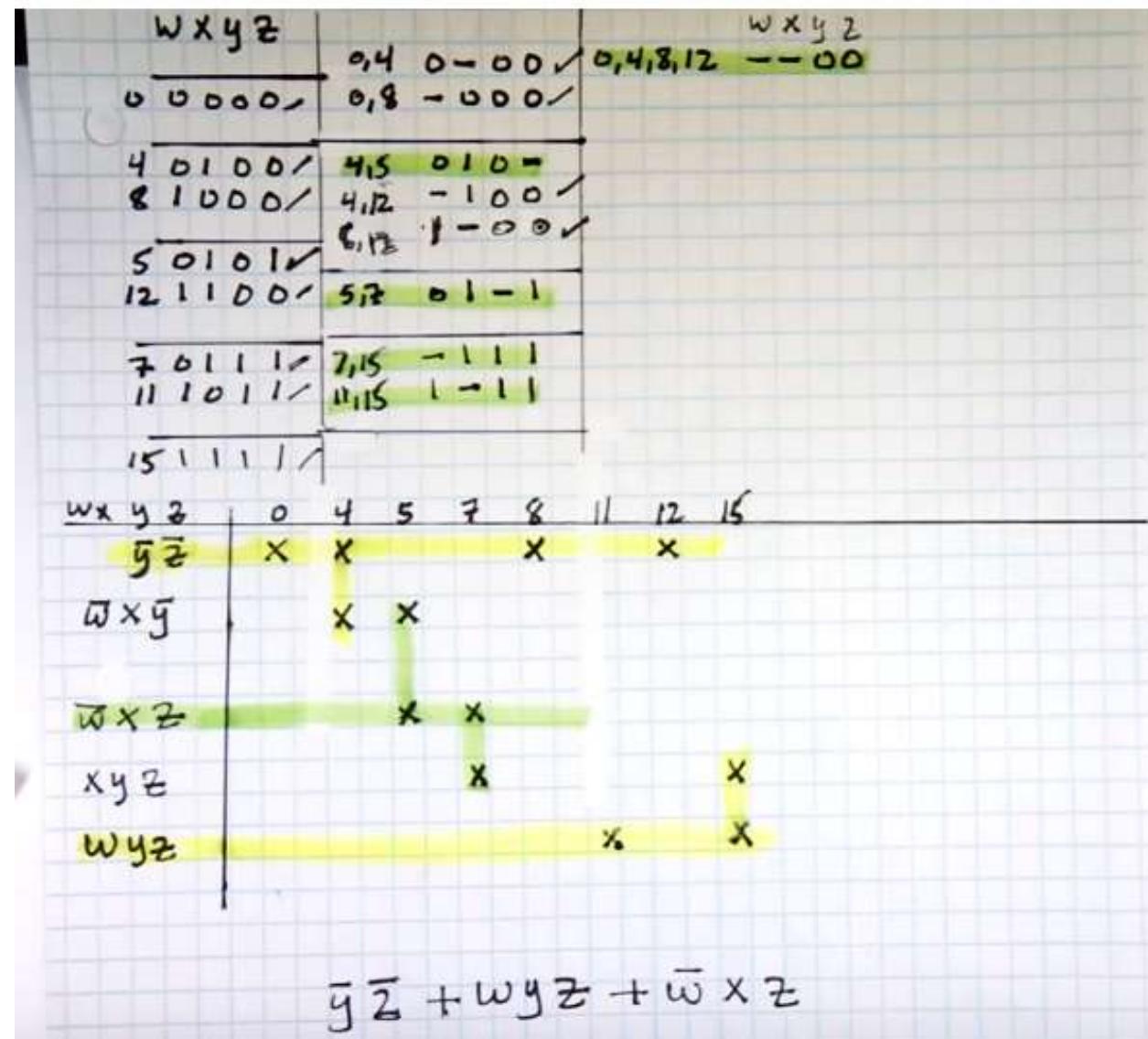
$(B'C' + CD)$ .

<http://myclassbook.wordpress.com>

PI terms	Group of Minterms	Minterms						
		0	1	3	7	8	9	11
$B'C'$	0,1,8,9	(X)	X		(X)	X		
$B'D$	1,3,9,11		X	X		X	X	
$CD$	3,7,11,15		X	(X)		X	(X)	

<http://myclassbook.wordpress.com>

$$F(w,x,y,z) = \sum m(0,4,5,7,8,11,12,15)$$



$$\bullet f(a,b,c,d) = \sum m(0,1,2,3,9,10) + \sum d(4,5,6)$$

$$f(a,b,c,d) = \sum m(0,1,2,3,9,10) + \sum d(4,5,6)$$

a b c d	a b c d	a b c d	a b c d	
0 0 0 0	0 0 0 0	0 0 0 0	0 1 0 0	0, 1, 2, 3
0 0 0 1	0 0 0 1	0 0 0 1	0 1 0 1	0, 1, 4, 5
0 0 1 0	0 0 1 0	0 0 1 0	0 1 0 0	0, 2, 1, 3
0 0 1 1	0 1 0 0	0 0 1 1	0 1 0 1	0, 2, 4, 6
0 1 0 0	0 0 1 1	0 - 0 1	0 - 0 1	0, 4, 1, 5
0 1 0 1	0 1 0 1	0 - 0 1	0 - 0 1	0, 4, 2, 6
0 1 1 0	0 1 1 0	0 0 1 -	0 - 0 1	0, 4, 2, 6
1 0 0 1	1 0 0 1	0 - 1 0	0 - 1 0	0, 6
1 1 0 0	1 0 1 0	0 1 0 -	0 1 0 -	0, 5
		0 1 0 0	0 1 0 0	0, 6
		- 1 0 0	- 1 0 0	0, 10

PI	Minterms	0	1	2	3	9	10
$\bar{b}\bar{c}d$ ✓	1, 9		✗			✗	✗
$b\bar{c}d$ ✓	4, 10						✗
$\bar{a}\bar{b}$ ✓	0, 1, 2, 3	✗	✗	✗	✗	✗	
$\bar{a}\bar{c}$	0, 1, 4, 5	✗	✗				
$\bar{a}\bar{d}$	0, 2, 4, 6	✗		✗			

$$f(a, b, c, d) = \bar{b}\bar{c}d + b\bar{c}d + \bar{a}\bar{b}$$

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

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

<u>1 000 1 ✓</u>	<u>(1,3) 00 - 1 ✓</u>	<u>(1,3,9,11) - 0 - 1</u>
<u>2 001 0 ✓</u>	<u>(1,9) - 0 0 1</u>	<u>(2,3,10,11) - 0 1 -</u>
<u>3 001 1 ✓</u>	<u>(2,3) 00 1 - ✓</u>	<u>(3,7,11,15) - - 1 1</u>
<u>9 1 00 1 ✓</u>	<u>(2,10) - 0 1 0</u>	<u>(9,11,13,15) 1 - - 1</u>
<u>10 1 0 1 0 ✓</u>	<u>(3,7) 0 - 1 1</u>	
<u>7 0 1 1 1 ✓</u>	<u>(3,11) - 0 1 1</u>	
<u>11 1 0 1 1 ✓</u>	<u>(9,11) 1 0 - 1 ✓</u>	
<u>13 1 1 0 1 ✓</u>	<u>(9,13) 1 - 0 1</u>	
<u>15 1 1 1 1 ✓</u>	<u>(10,11) 1 0 1 - ✓</u>	
	<u>(7,15) - 1 1 1</u>	
	<u>(11,15) 1 - 1 1</u>	
	<u>(13,15) 1 1 - 1</u>	

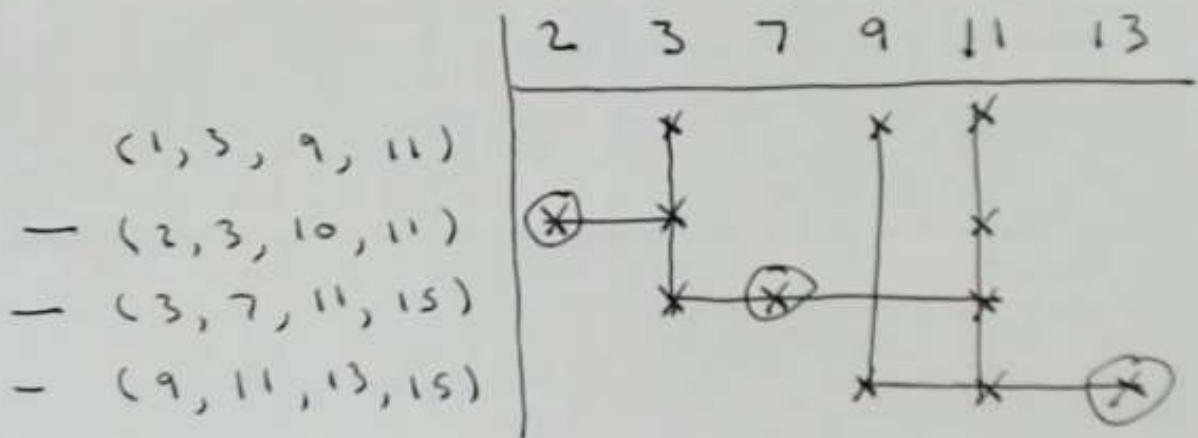
A B C D

$$(1, 3, 9, 11) \quad -0-1$$

$$(2, 3, 10, 11) \quad -01-$$

$$\underline{(3, 2, 11, 15)} \quad - - 11$$

$$(9, 11, 13, 15) \quad 1 - - 1$$

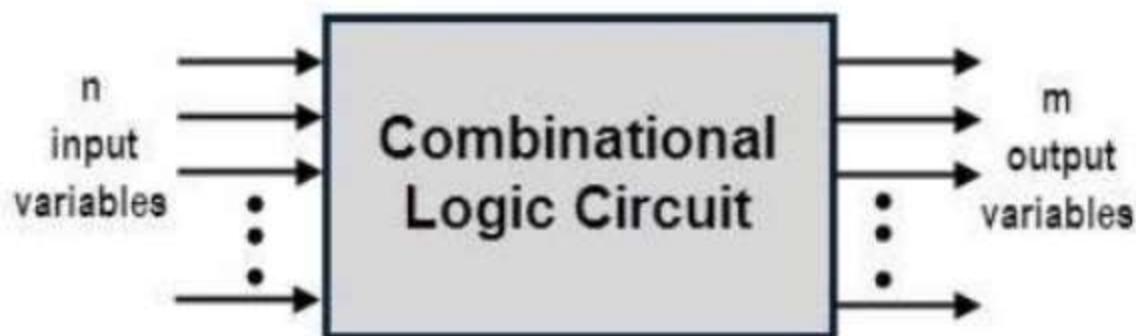


$$F(A, B, C, D) : \bar{B}C + CD + AD$$

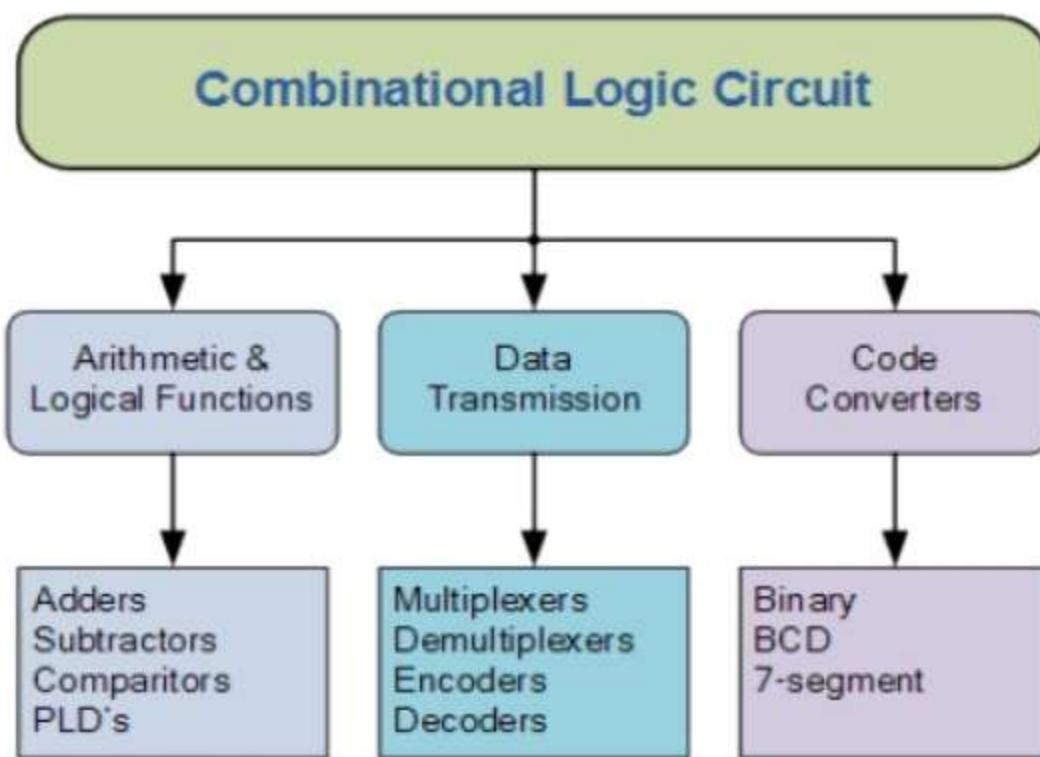
# Combinational Circuits

- A combinational logic circuit is one in which the present state of the combination of the logic inputs decides the output .
- The term combination logic means combining of two or more logic gates to form a required function where the output at a given time depends only on the input.

- The required output data is obtained from this process by transforming the binary information given at the input.



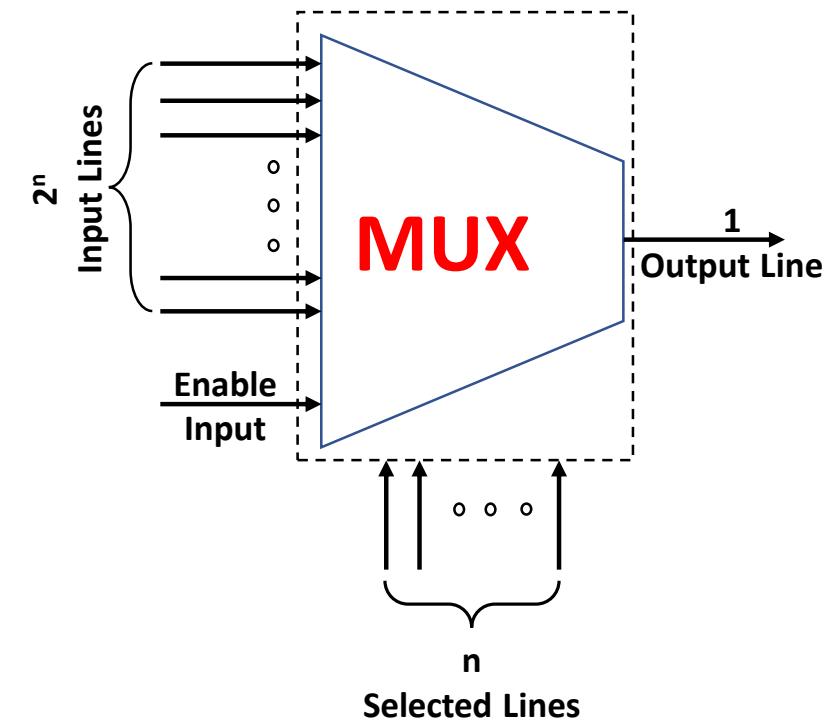
# Classification of Combinational Logic



# MULTIPLEXER

# Multiplexer:

- It is a combinational logic circuit.
- It is a device that selects between several analog or digital input signals and forwards it to a single output line.
- Input –  $2^n$
- Output – 1
- Multiplexer –  $2^n : 1$
- The selection of particular input line is controlled by a set of selection lines.
- It is also called Data Selector.
- There is a Enable input, which is required to connect two or three multiplexer in parallel.



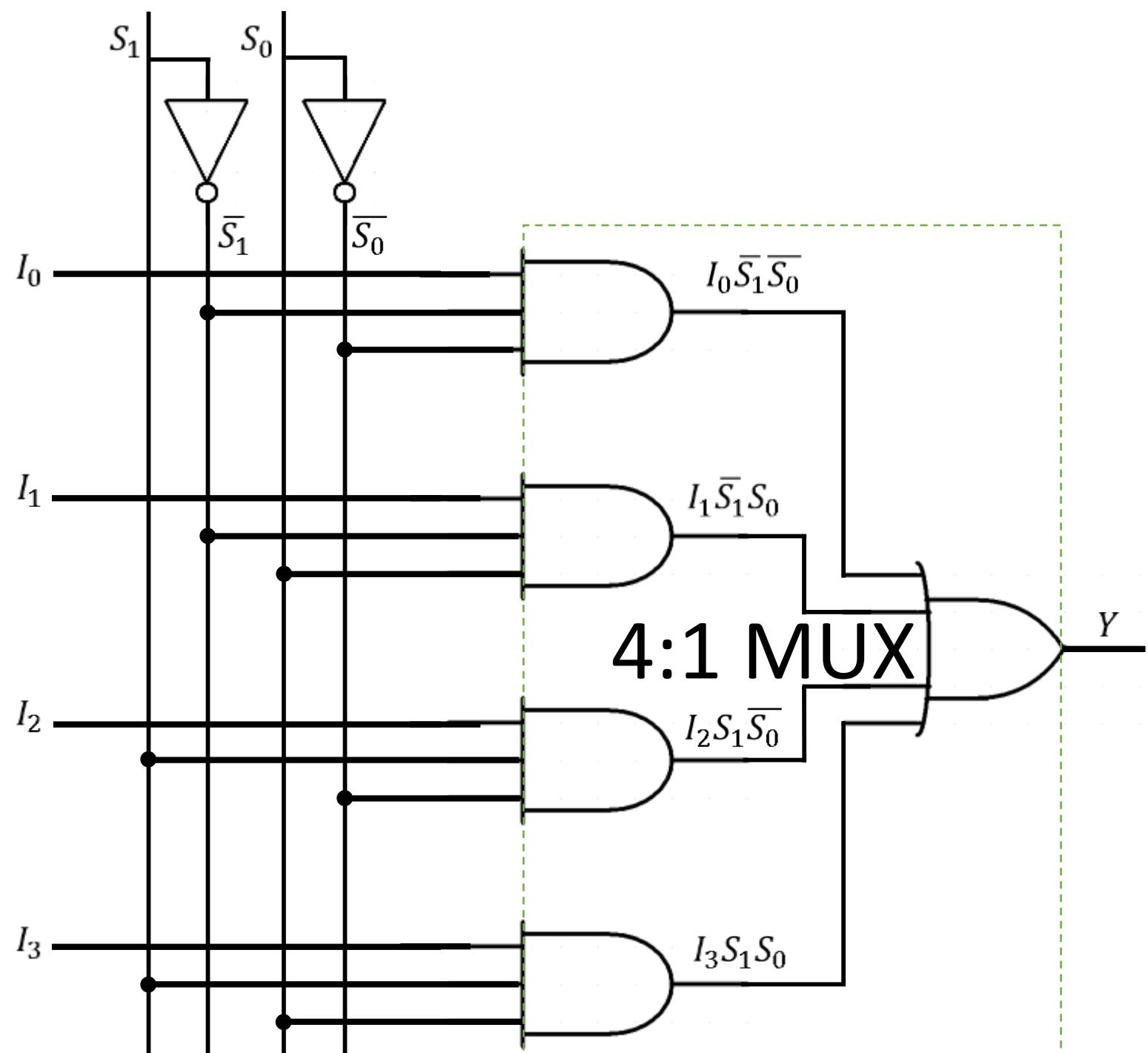
# Design of 4:1 Mux:

$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

- Input: 4 ( $I_0, I_1, I_2$  &  $I_3$ )
- Output: 1 ( $Y$ )
- Selection Line: 2 ( $S_0$  &  $S_1$ )
- Output is,

$$Y = I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0$$

$$Y = I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0$$

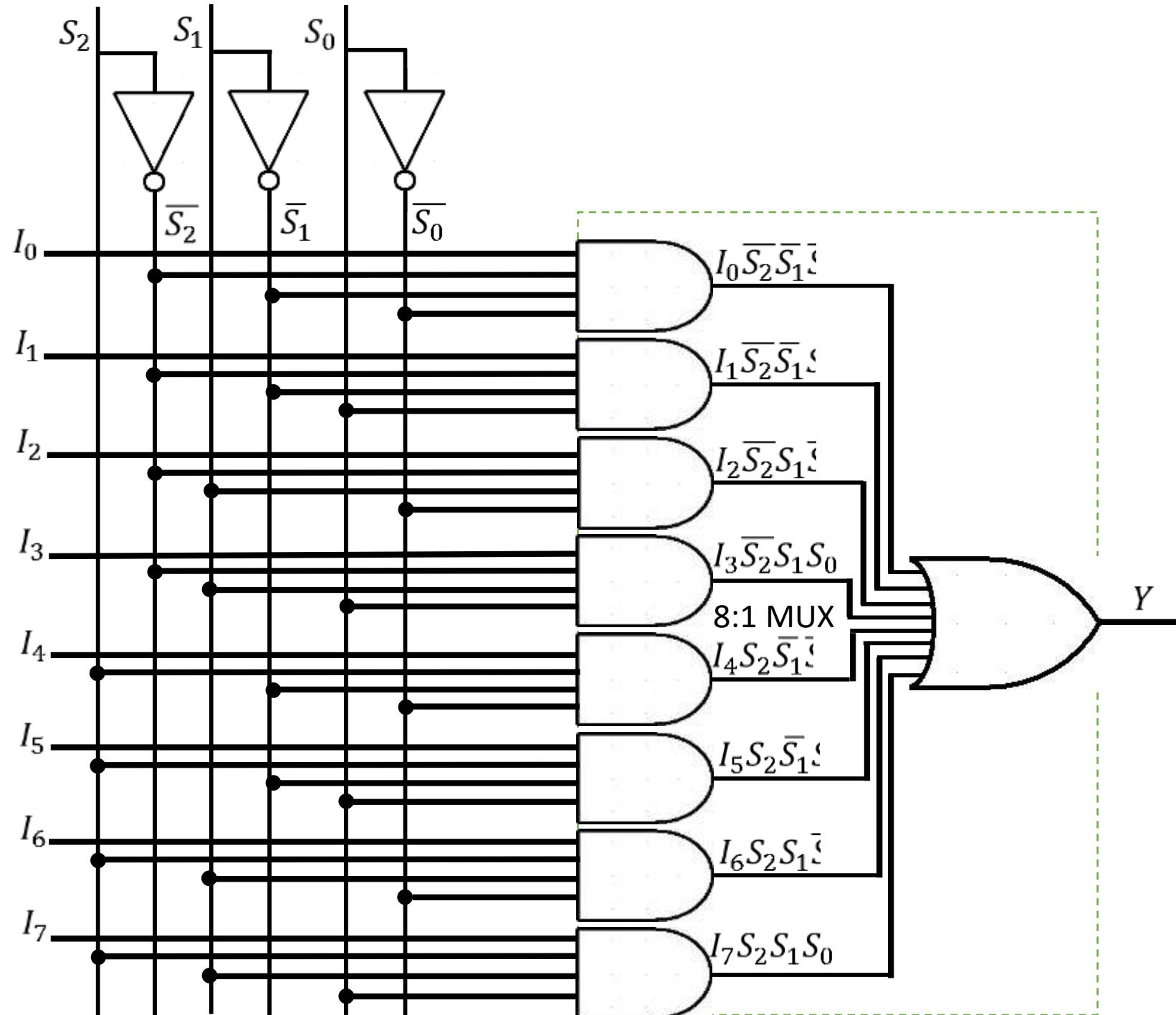


## Design of 8:1 Mux:

$S_2$	$S_1$	$S_0$	$Y$
0	0	0	$I_0$
0	0	1	$I_1$
0	1	0	$I_2$
0	1	1	$I_3$
1	0	0	$I_4$
1	0	1	$I_5$
1	1	0	$I_6$
1	1	1	$I_7$

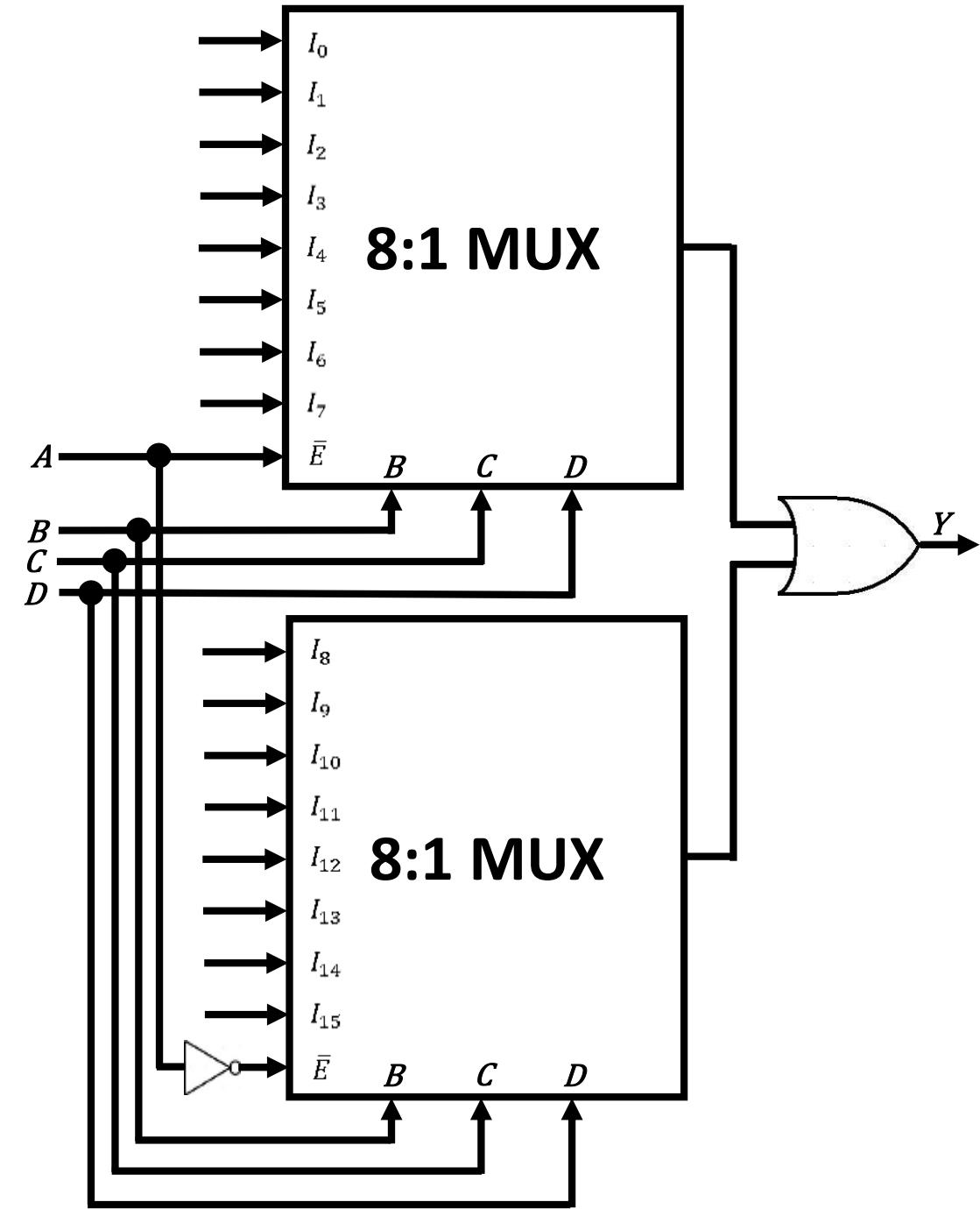
- Input: 8 ( $I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7$  &  $I_8$ )
- Output: 1 ( $Y$ )
- Selection Line: 3 ( $S_0, S_1$  &  $S_2$ )
- Output is,

$$Y = I_0 \bar{S}_2 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_2 \bar{S}_1 S_0 + I_2 \bar{S}_2 S_1 \bar{S}_0 + I_3 S_2 \bar{S}_1 \bar{S}_0 + I_4 S_2 \bar{S}_1 S_0 + I_5 S_2 S_1 \bar{S}_0 + I_6 S_2 S_1 S_0 + I_7 S_2 \bar{S}_1 S_0$$



# Design of 16:1 MUX using 8:1 MUX

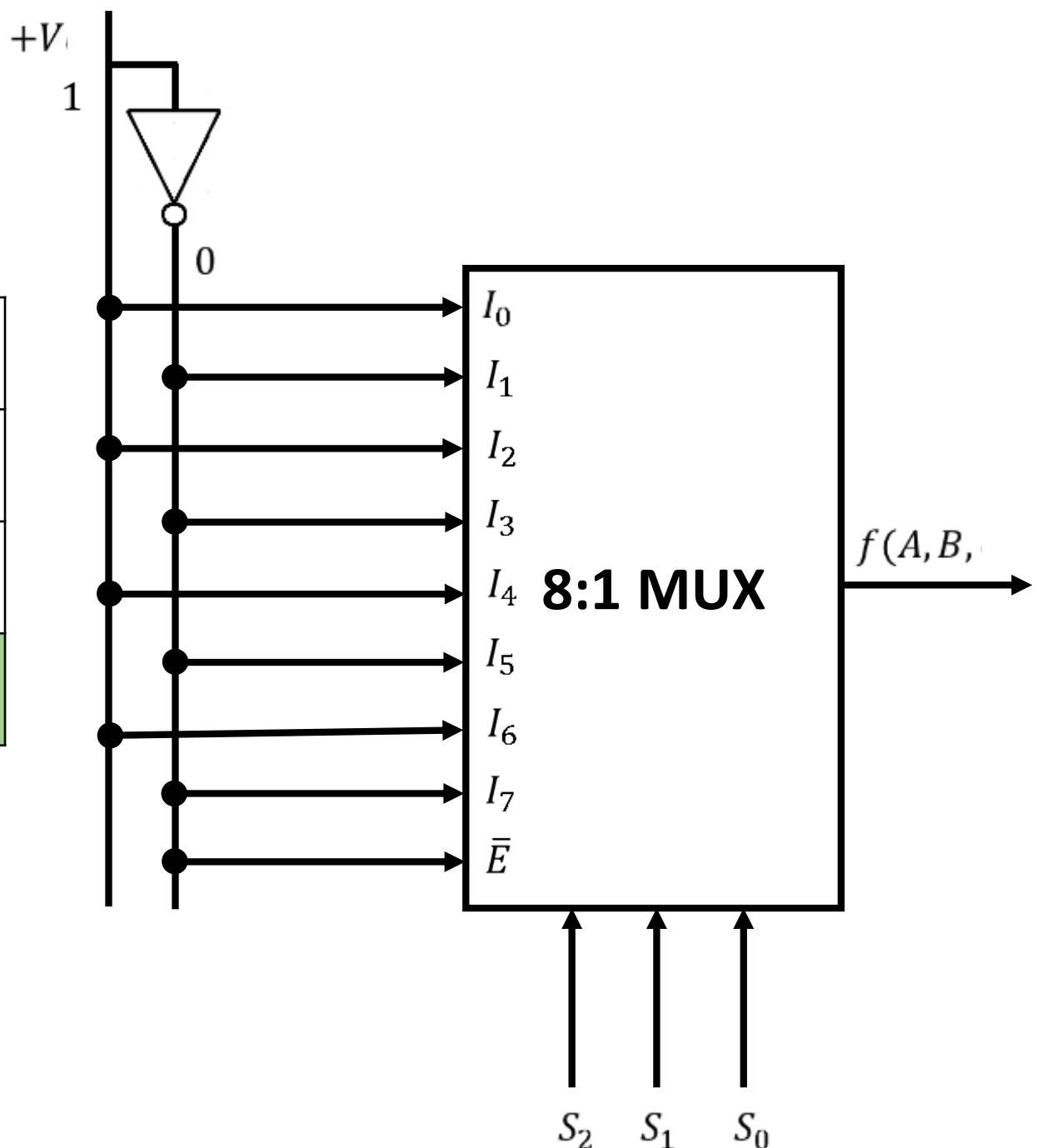
$S_3$	$S_2$	$S_1$	$S_0$	$Y$
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	



Implement the following function using 8:1 MUX

$$f(A, B, C, D) = \sum_m(0, 2, 4, 6, 8, 10, 12, 14)$$

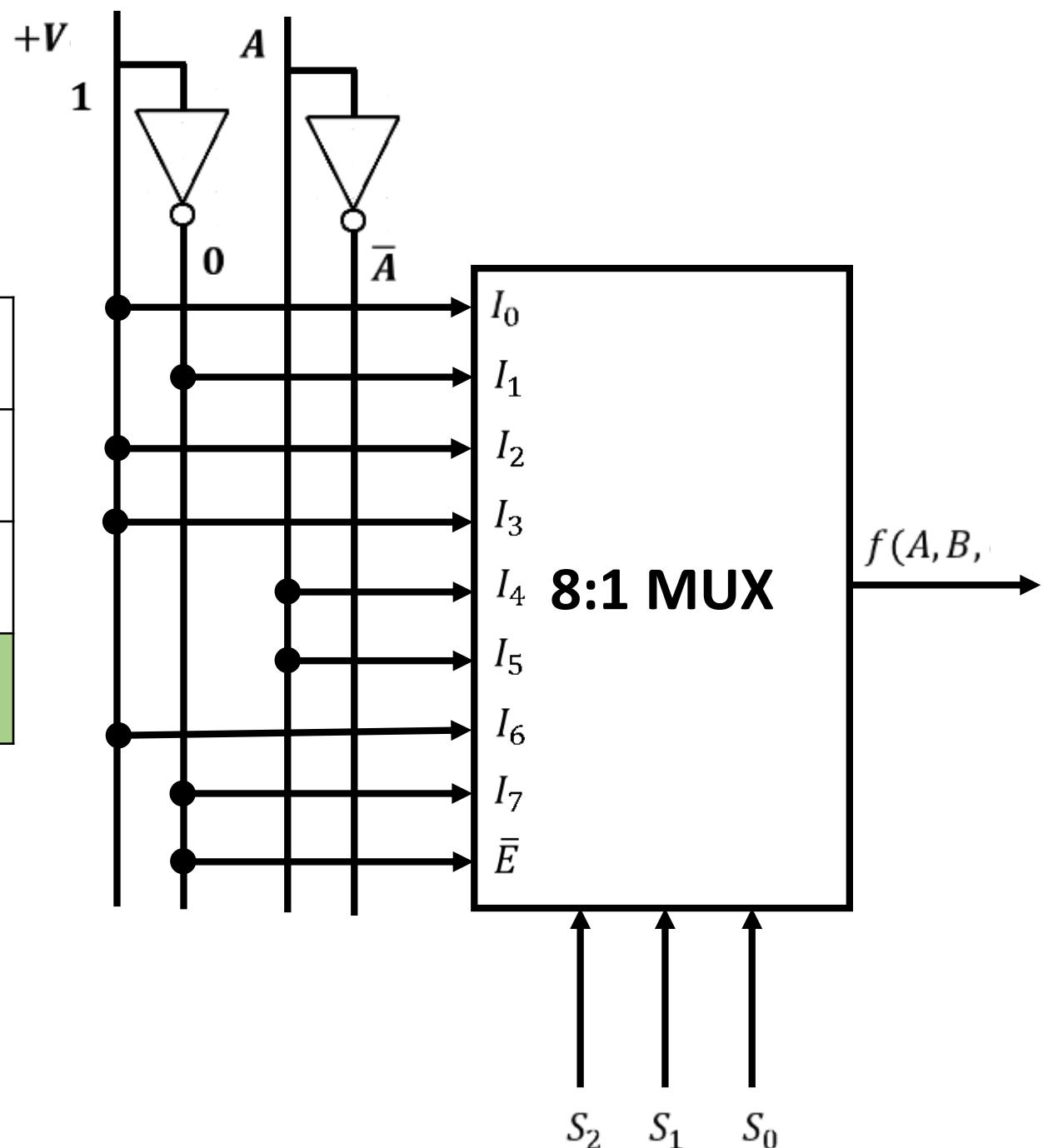
0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
1	0	1	0	1	0	1	0	



Implement the following function using 8:1 MUX

$$f(A, B, C, D) = \sum_m(0, 2, 6, 10, 11, 12, 13) + d(3, 8, 14)$$

0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
1	0	1	1	A	A	1	0	



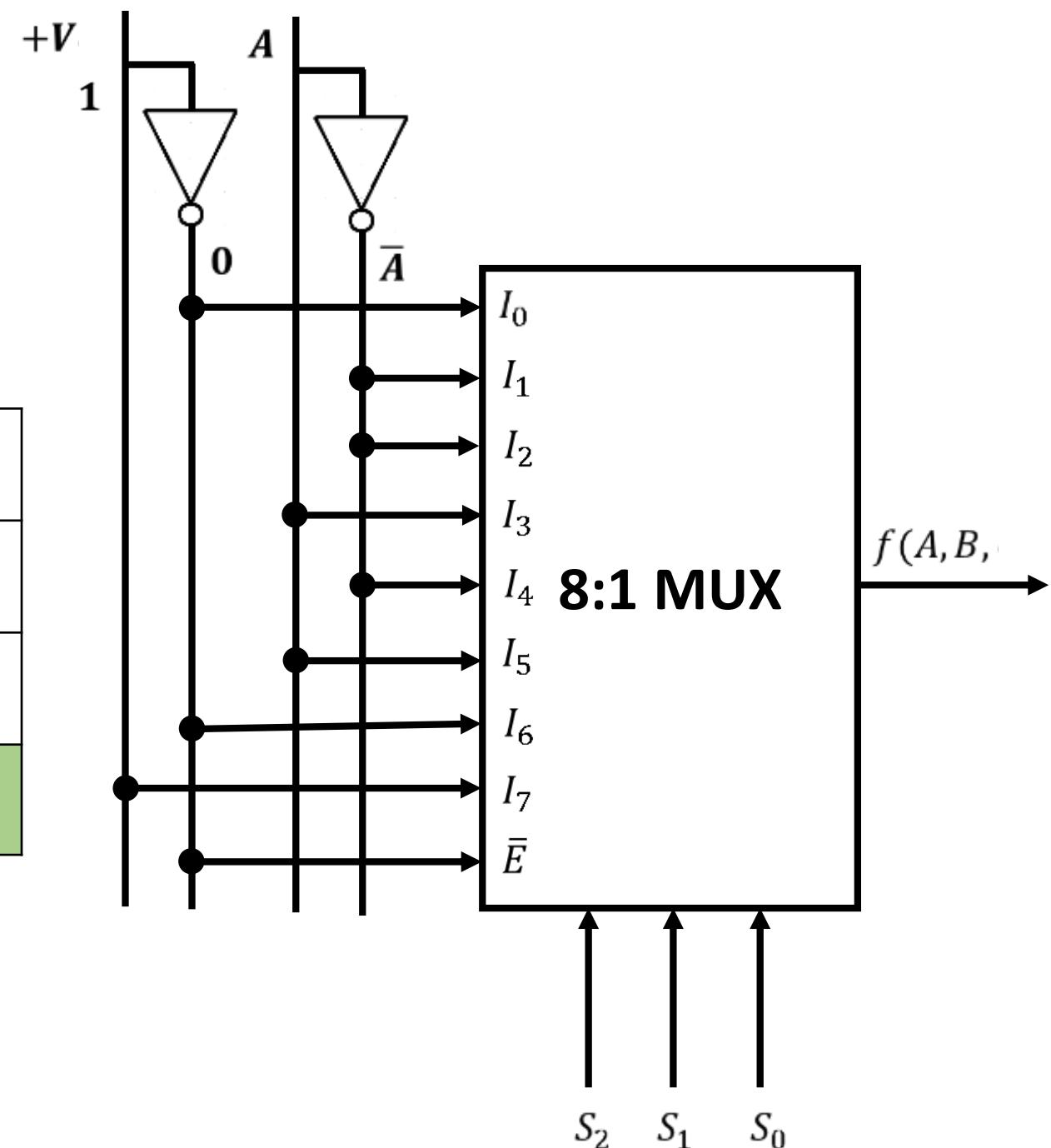
Implement the following function using 8:1 MUX

$$f(A, B, C, D) = \prod_M(0, 3, 5, 6, 8, 9, 10, 12, 14)$$

PoS function should be converted into SoP function,

$$f(A, B, C, D) = \sum_m(1, 2, 4, 7, 11, 13, 15)$$

	0	1	2	3	4	5	6	7
	8	9	10	11	12	13	14	15
	0			A		A	0	1

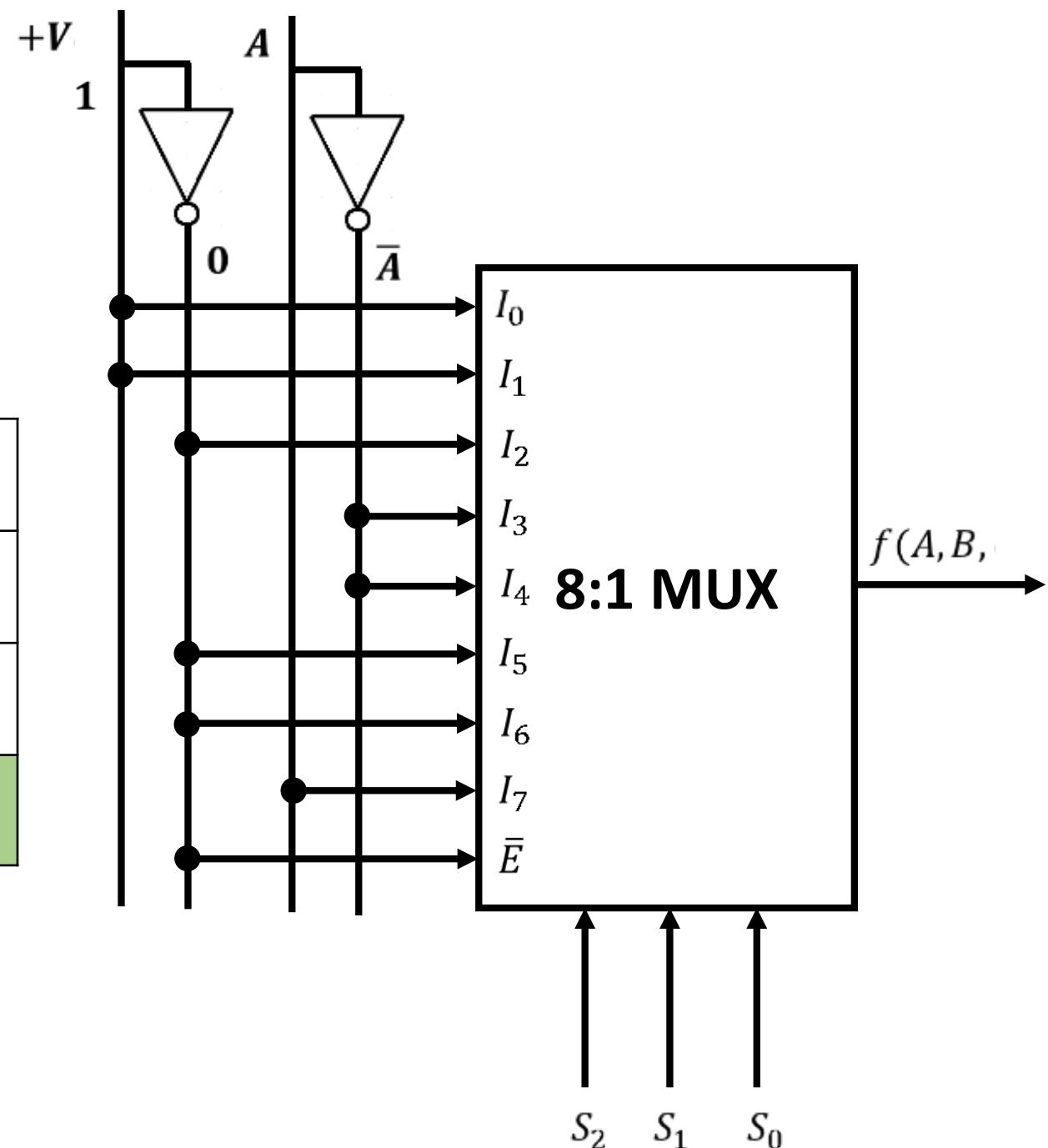


Implement the following function using 8:1 MUX  
and 4:1 MUX

$$f(A, B, C, D) = \sum_m(0, 1, 3, 4, 8, 9, 15)$$

## Using 8:1 Mux

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
1	1	0			0	0	A

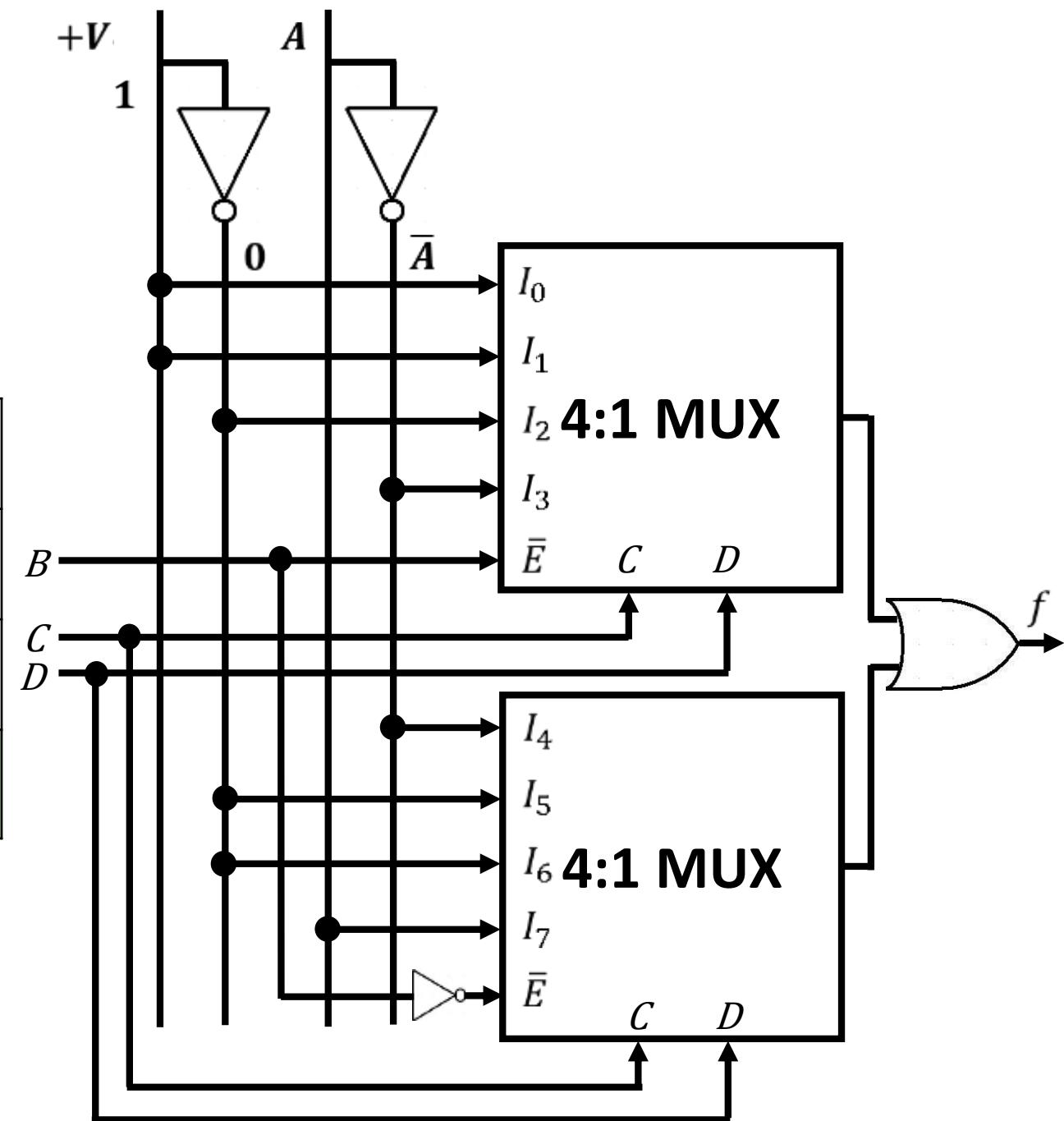


Implement the following function using 8:1 MUX  
and 4:1 MUX

$$f(A, B, C, D) = \sum_m(0, 1, 3, 4, 8, 9, 15)$$

## Using 4:1 Mux

0	1	2	3	4	5	6	7	
8	9	10	11	12	13	14	15	
1	1	0			0	0	A	



# Application of Multiplexer

- It is used as a data selector to select one out of many data inputs.
- It is used to implement combinational logic circuit.
- It is used in time multiplexing systems.
- It is used in ADC and DAC converter.
- It is used in data acquisition systems.

**THANK YOU...**

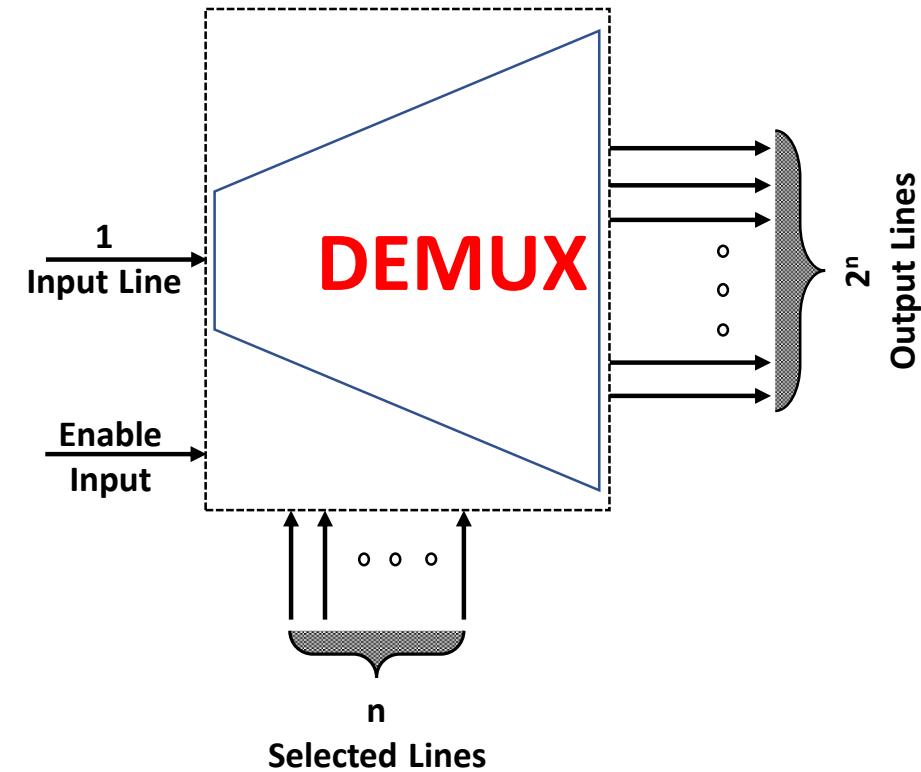
**ANY QUERIES???**

# DEMULTIPLEXER

PREPARED BY  
Mr.P.KANAKARAJ  
AP/EEE  
SRMIST-KTR

# Demultiplexer:

- It is a combinational logic circuit.
- It is a circuit that receives information on a single line and transmits this information on one of  $2^n$  possible output lines.
- The selection of specific output line is controlled by the values of  $n$  selection lines.
- Input – 1
- Output –  $2^n$
- Selection lines –  $n$
- De-Multiplexer –  $1 : 2^n$
- There is a Enable input, which is required to connect two or three multiplexer in parallel.



# Design of 1:4 DEMUX:

## Truth Table

Selection Lines		Outputs			
$S_1$	$S_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
0	0	$I$	0	0	0
0	1	0	$I$	0	0
1	0	0	0	$I$	0
1	1	0	0	0	$I$

- Input: 1 ( $I$ )
- Output: 4 ( $Y_0, Y_1, Y_2$  &  $Y_3$ )
- Selection Line: 2 ( $S_0$  &  $S_1$ )
- Outputs are,

$$Y_0 = I\bar{S}_1\bar{S}_0 = I$$

$$Y_1 = I\bar{S}_1S_0 = I$$

$$Y_2 = IS_1\bar{S}_0 = I$$

$$Y_3 = IS_1S_0 = I$$

# Logic Diagram of 1:4 DEMUX:

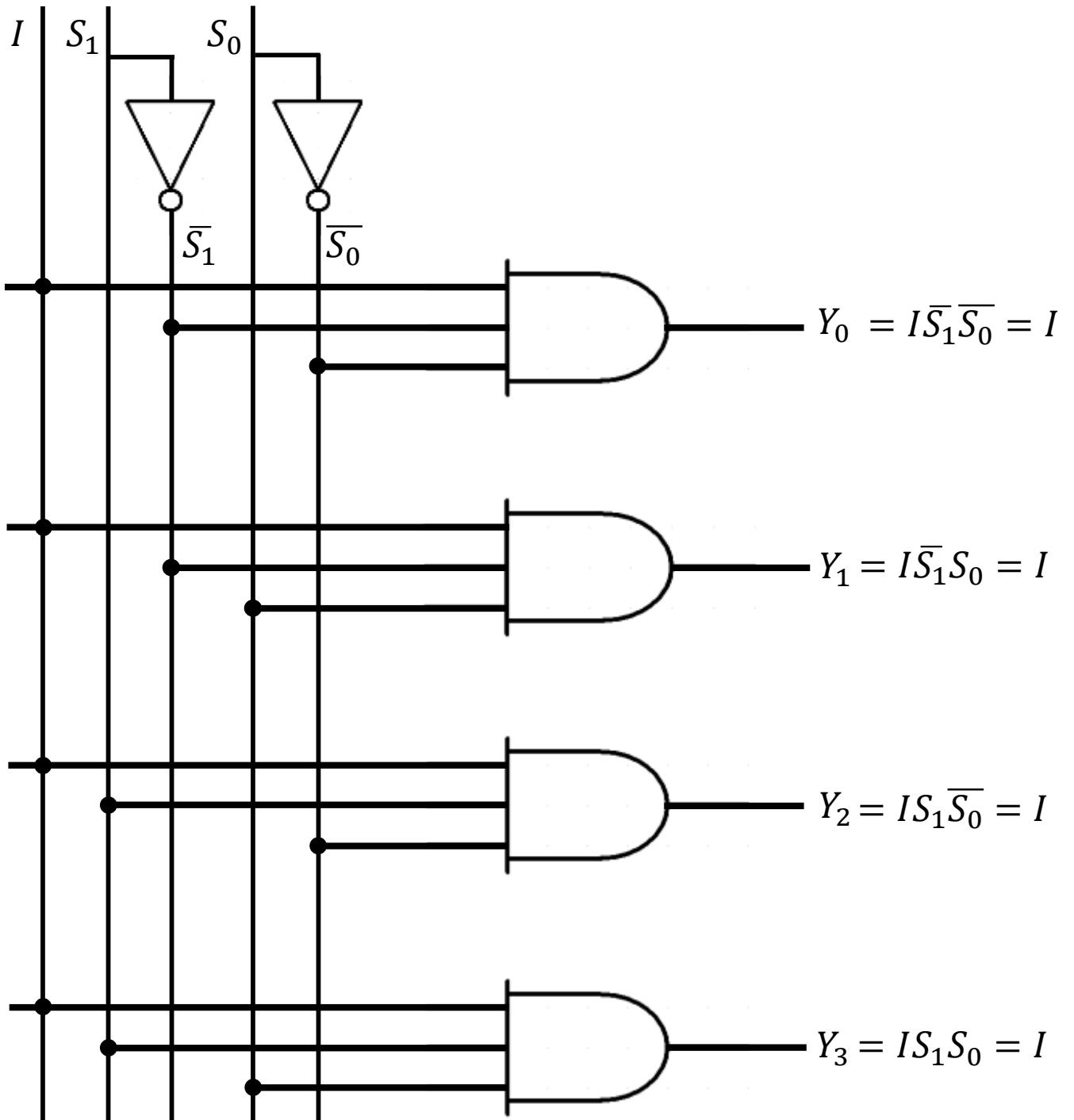
Outputs are,

$$Y_0 = I\bar{S}_1\bar{S}_0 = I$$

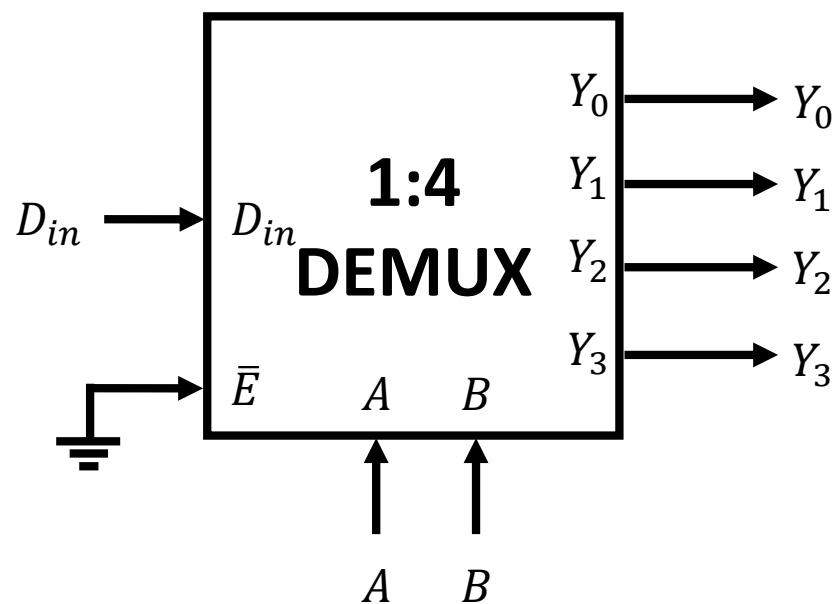
$$Y_1 = I\bar{S}_1S_0 = I$$

$$Y_2 = IS_1\bar{S}_0 = I$$

$$Y_3 = IS_1S_0 = I$$



# 1:4 DEMUX



# Design of 1:8 DEMUX

Truth Table

Selection Lines			Outputs							
$S_0$	$S_1$	$S_2$	$Y_0$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$
0	0	0	$I$	0	0	0	0	0	0	0
0	0	1	0	$I$	0	0	0	0	0	0
0	1	0	0	0	$I$	0	0	0	0	0
0	1	1	0	0	0	$I$	0	0	0	0
1	0	0	0	0	0	0	$I$	0	0	0
1	0	1	0	0	0	0	0	$I$	0	0
1	1	0	0	0	0	0	0	0	$I$	0
1	1	1	0	0	0	0	0	0	0	$I$

- Input: 1 ( $I$ )
- Output: 8 ( $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$ )
- Selection Line: 3 ( $S_0, S_1 \& S_2$ )
- Outputs are,

$$Y_0 = I\bar{S}_2\bar{S}_1\bar{S}_0$$

$$Y_4 = IS_2\bar{S}_1\bar{S}_0$$

$$Y_1 = I\bar{S}_2\bar{S}_1S_0$$

$$Y_5 = IS_2\bar{S}_1S_0$$

$$Y_2 = I\bar{S}_2S_1\bar{S}_0$$

$$Y_6 = IS_2S_1\bar{S}_0$$

$$Y_3 = I\bar{S}_2S_1S_0$$

$$Y_7 = IS_2S_1S_0$$

# Logic Diagram of 1:8 DEMUX:

Outputs are,

$$Y_0 = I\bar{S}_2\bar{S}_1\bar{S}_0$$

$$Y_4 = I\bar{S}_2\bar{S}_1\bar{S}_0$$

$$Y_1 = I\bar{S}_2\bar{S}_1S_0$$

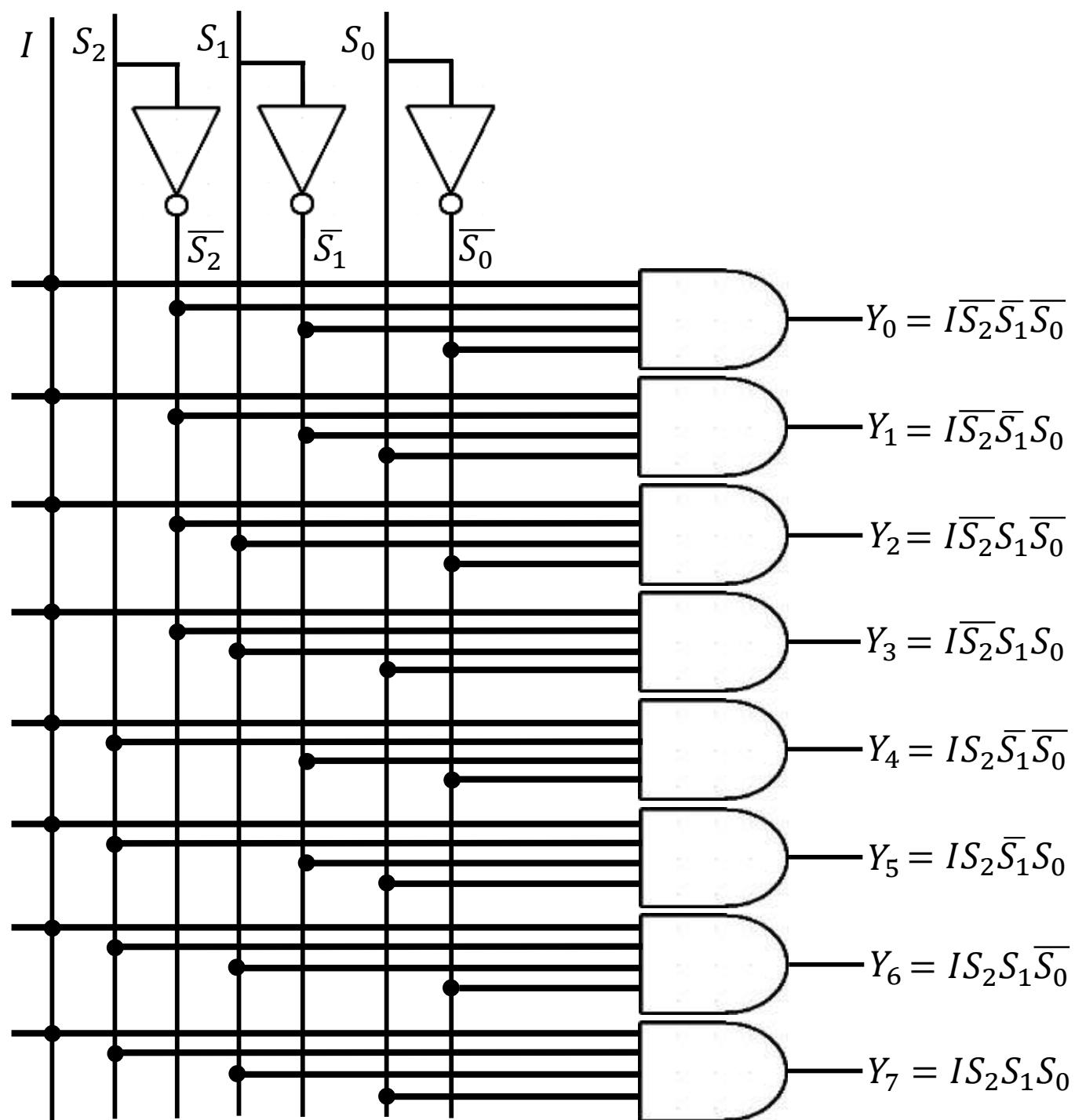
$$Y_5 = I\bar{S}_2\bar{S}_1S_0$$

$$Y_2 = I\bar{S}_2S_1\bar{S}_0$$

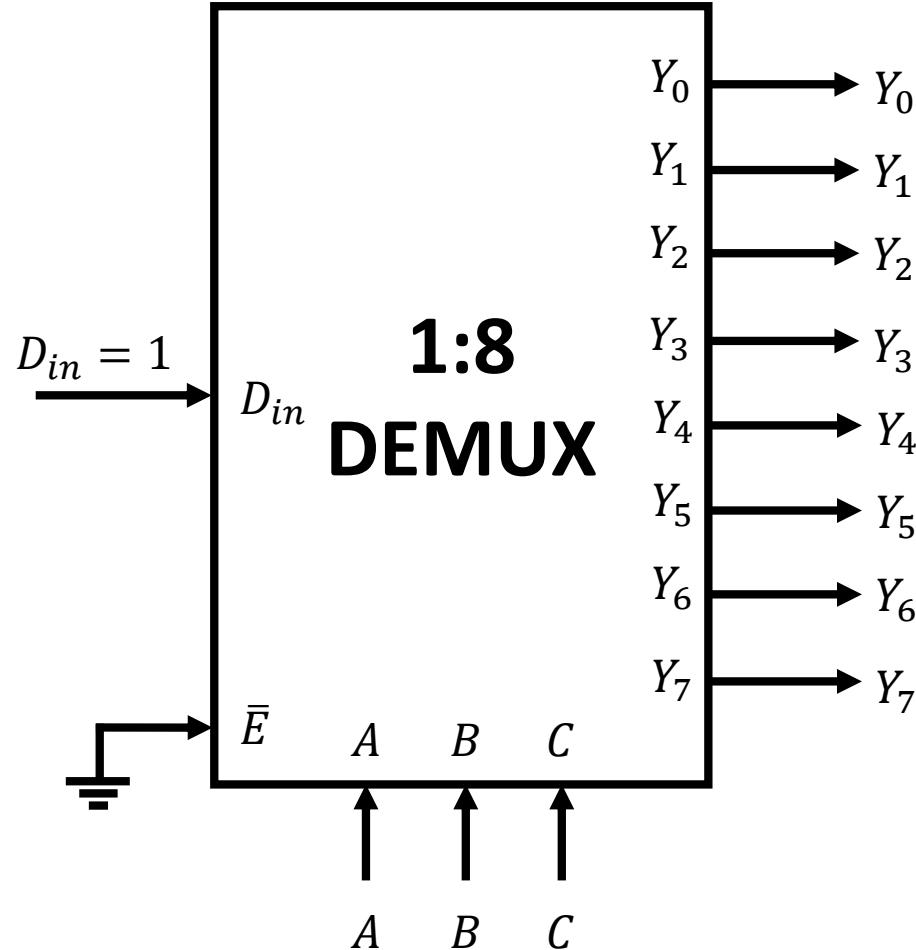
$$Y_6 = I\bar{S}_2S_1\bar{S}_0$$

$$Y_3 = I\bar{S}_2S_1S_0$$

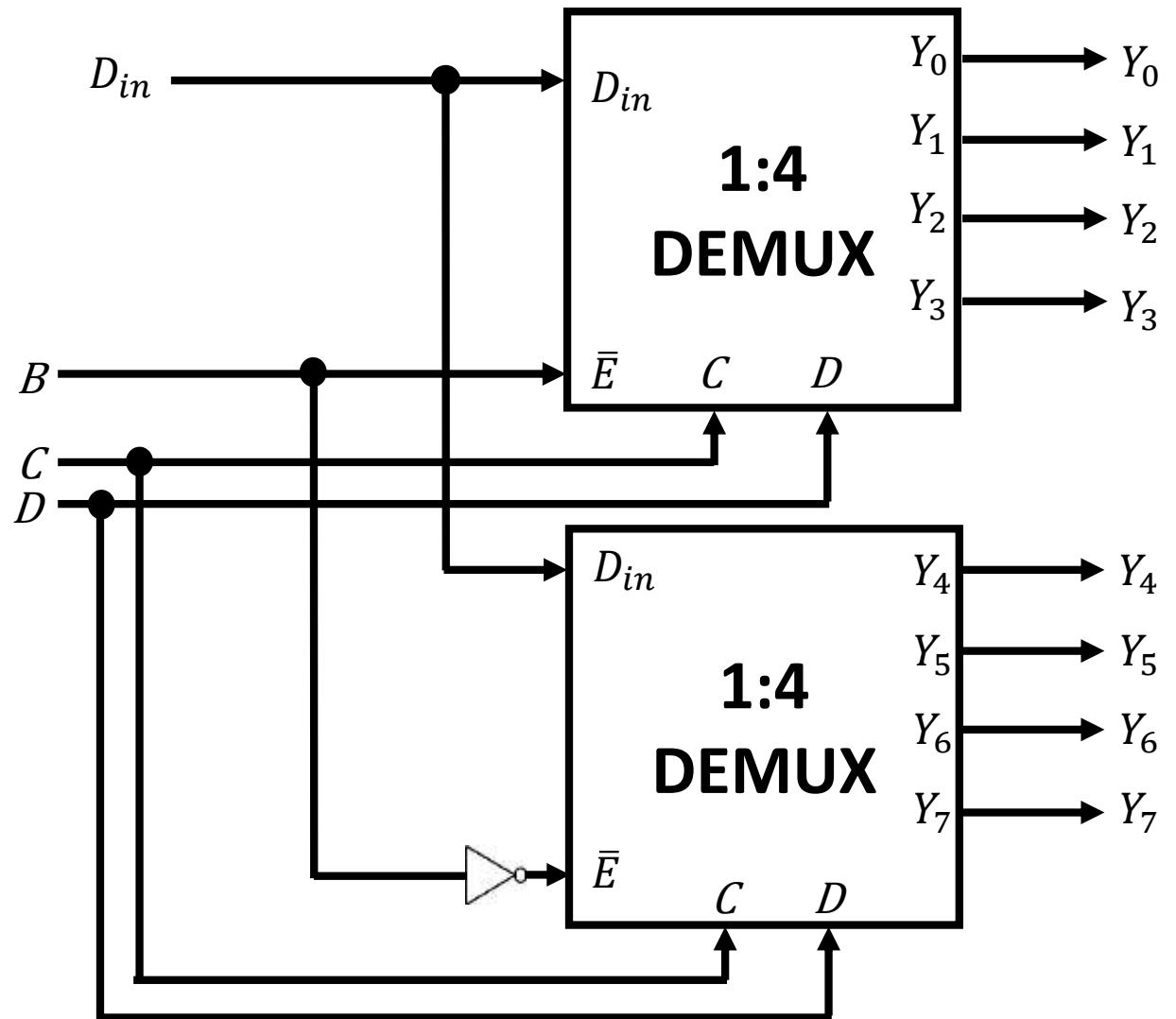
$$Y_7 = I\bar{S}_2S_1S_0$$



# 1:8 DEMUX



# Design of 1:8 DEMUX Using 1:4 DEMUX



# Implementation of Full Adder using Demultiplexer

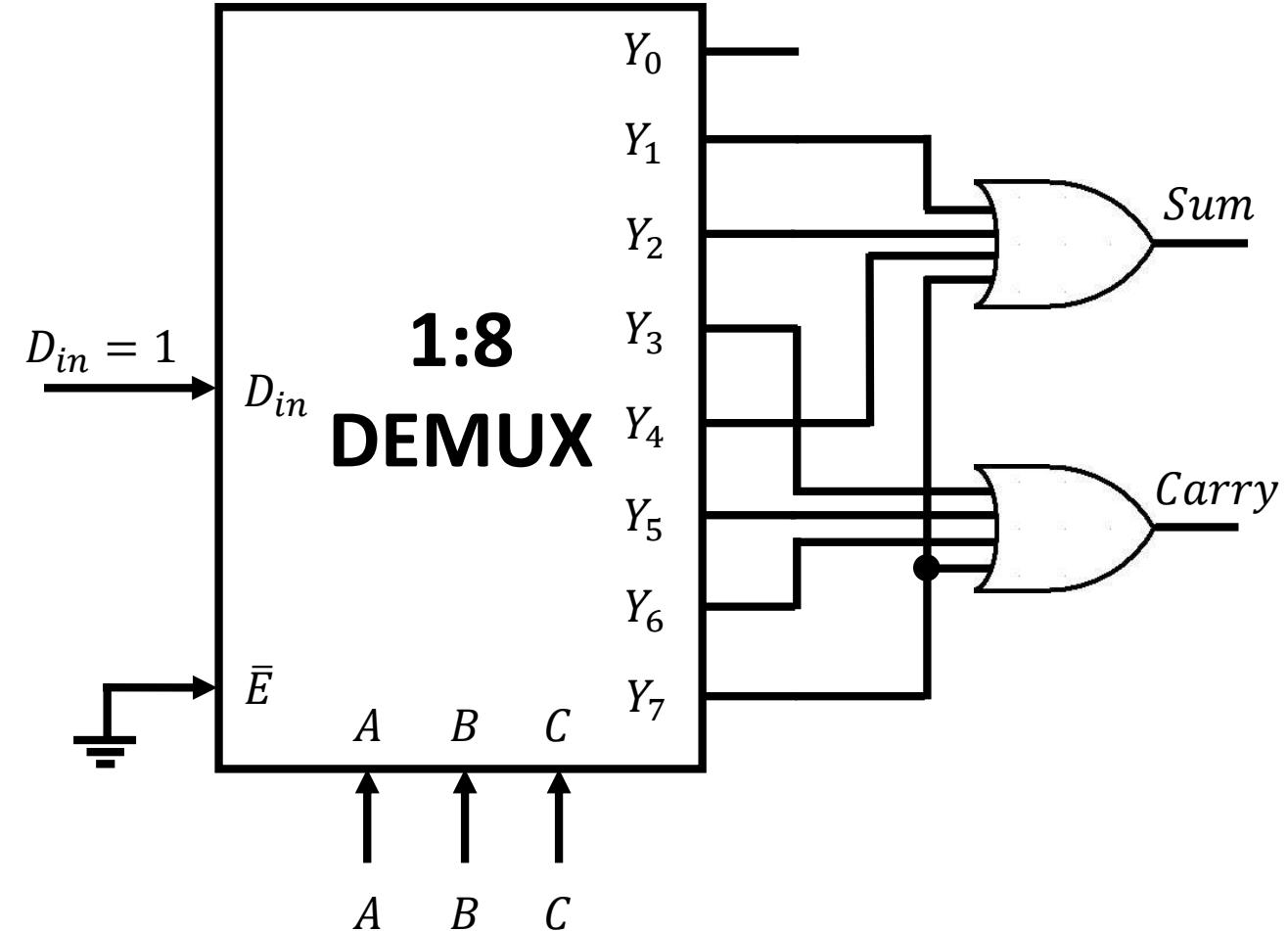
Truth Table

Decimal	Inputs			Outputs	
	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Output Expressions are,

$$\text{Sum} = \sum_m(1,2,4,7)$$

$$\text{Carry} = \sum_m(3,5,6,7)$$



# Implementation of Full Subtractor using Demultiplexer

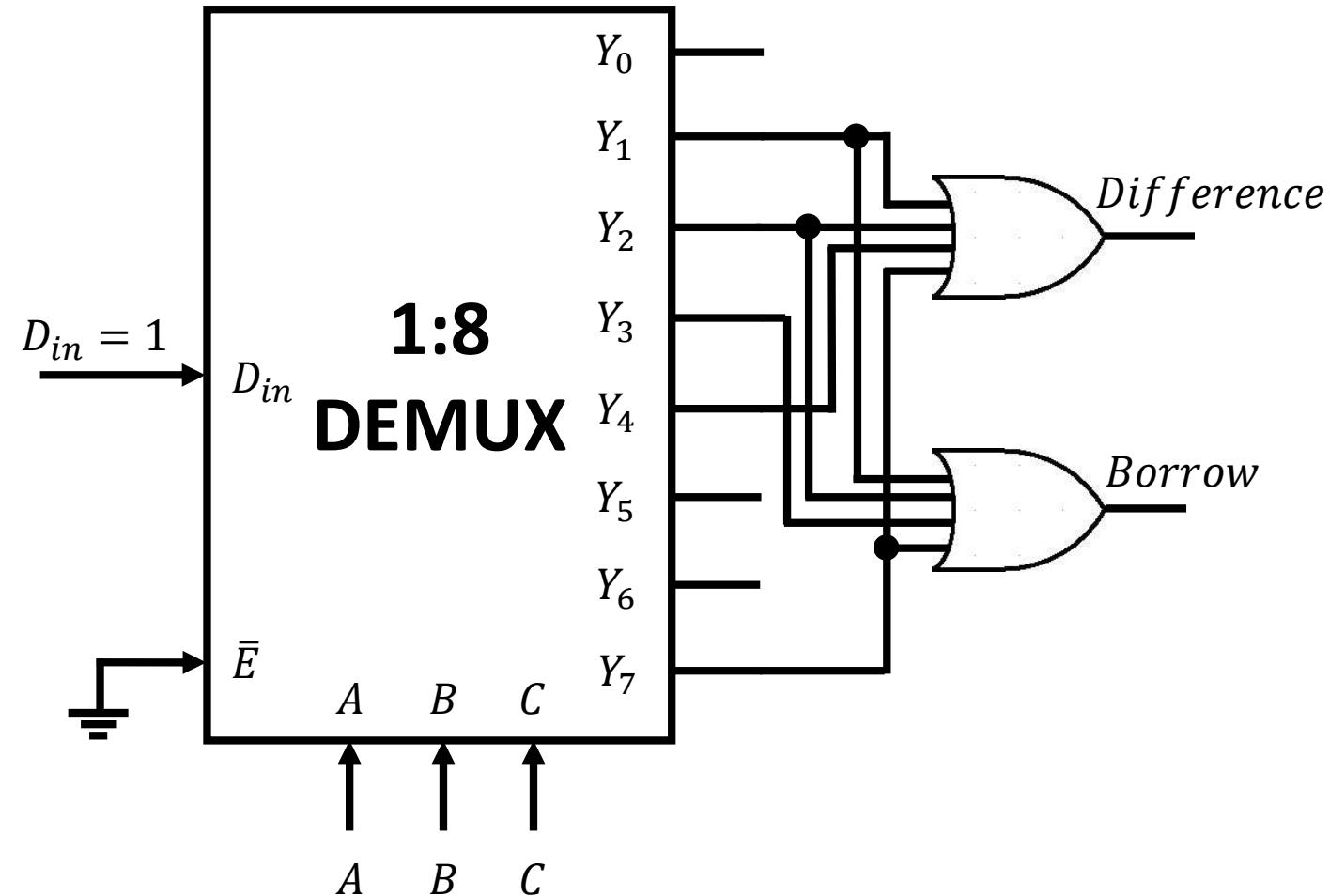
Truth Table

Decimal	Inputs			Outputs	
	A	B	C	Difference	Borrow
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

Output Expressions are,

$$\text{Difference} = \sum_m(1,2,4,7)$$

$$\text{Borrow} = \sum_m(1,2,3,7)$$



# Applications of Demultiplexer:

- Clock demultiplexer
- Security monitoring system
- Synchronous data transmission system

**THANK YOU...**

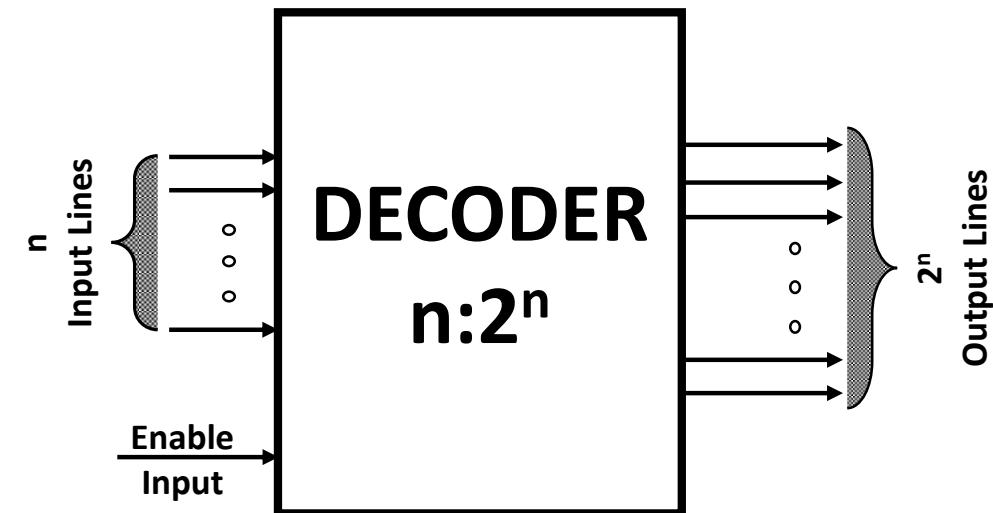
**ANY QUERIES???**

# DECODER

PREPARED BY  
Mr.P.KANAKARAJ  
AP/EEE  
SRMIST-KTR

# Decoder:

- It is a combinational logic circuit.
- It has an  $n$ -bit binary input code and a one activated output out of  $2^n$  output code is called binary decoder.
- A binary decoder is used when it is necessary to activate exactly one of  $2^n$  outputs based on an  $n$ -bit input value.
- It is similar to demultiplexer, with only one exception that it has no data input.
- Input –  $n$
- Output –  $2^n$
- Decoder –  $n : 2^n$
- There is a Enable input, which is required to connect two or three decoder in parallel.



# Decoder Vs Demultiplexer

- **Decoder**

- Decoder is a many inputs to many outputs device.
- There are no selection lines.

- **Demultiplexer**

- Demultiplexer is a one input to many outputs device.
- The selection of specific output line is controlled by the value of selection lines.

# Design of 2:4 DECODER

## Truth Table

Input			Output			
$E$	$S_1$	$S_0$	$Y_0$	$Y_1$	$Y_2$	$Y_3$
1	$X$	$X$	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

- Input: 1 ( $I$ )
- Output: 4 ( $Y_0, Y_1, Y_2$  &  $Y_3$ )
- Selection Line: 2 ( $S_0$  &  $S_1$ )
- Outputs are,

$$Y_0 = \bar{E} \bar{S}_1 \bar{S}_0$$

$$Y_1 = \bar{E} \bar{S}_1 S_0$$

$$Y_2 = \bar{E} S_1 \bar{S}_0$$

$$Y_3 = \bar{E} S_1 S_0$$

# Logic Diagram of 2:4 Decoder

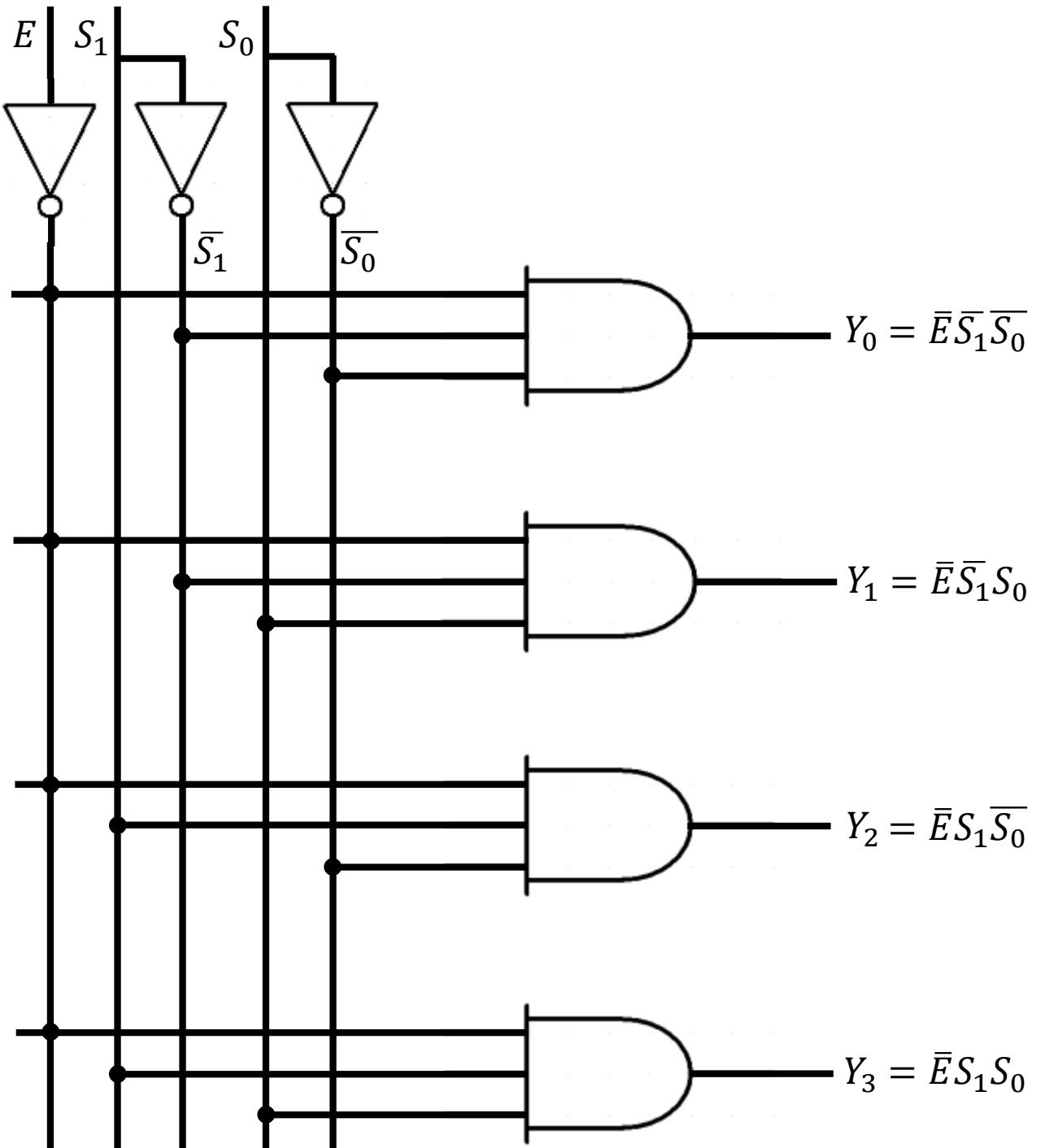
Outputs are,

$$Y_0 = \bar{E} \bar{S}_1 \bar{S}_0$$

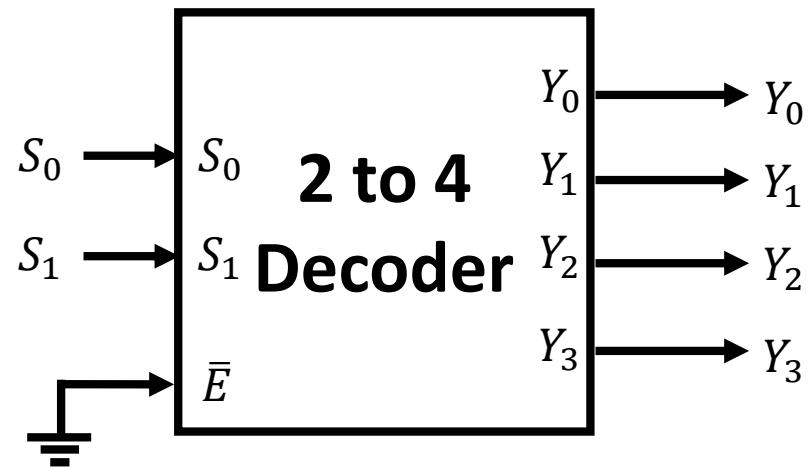
$$Y_1 = \bar{E} \bar{S}_1 S_0$$

$$Y_2 = \bar{E} S_1 \bar{S}_0$$

$$Y_3 = \bar{E} S_1 S_0$$



# 2 to 4 Line Decoder



# Three to Eight line Decoder

Truth Table

Input			Output							
$S_0$	$S_1$	$S_2$	$Y_0$	$Y_1$	$Y_2$	$Y_3$	$Y_4$	$Y_5$	$Y_6$	$Y_7$
0	0	0	I	0	0	0	0	0	0	0
0	0	1	0	I	0	0	0	0	0	0
0	1	0	0	0	I	0	0	0	0	0
0	1	1	0	0	0	I	0	0	0	0
1	0	0	0	0	0	0	I	0	0	0
1	0	1	0	0	0	0	0	I	0	0
1	1	0	0	0	0	0	0	0	I	0
1	1	1	0	0	0	0	0	0	0	I

- Input: 3 ( $S_0, S_1, S_2$ )
- Output: 8 ( $Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$ )
- Outputs are,

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

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

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

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

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

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

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

$$Y_7 = S_2 S_1 S_0$$

# Logic Diagram of 3 to 8 line Decoder

Outputs are,

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

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

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

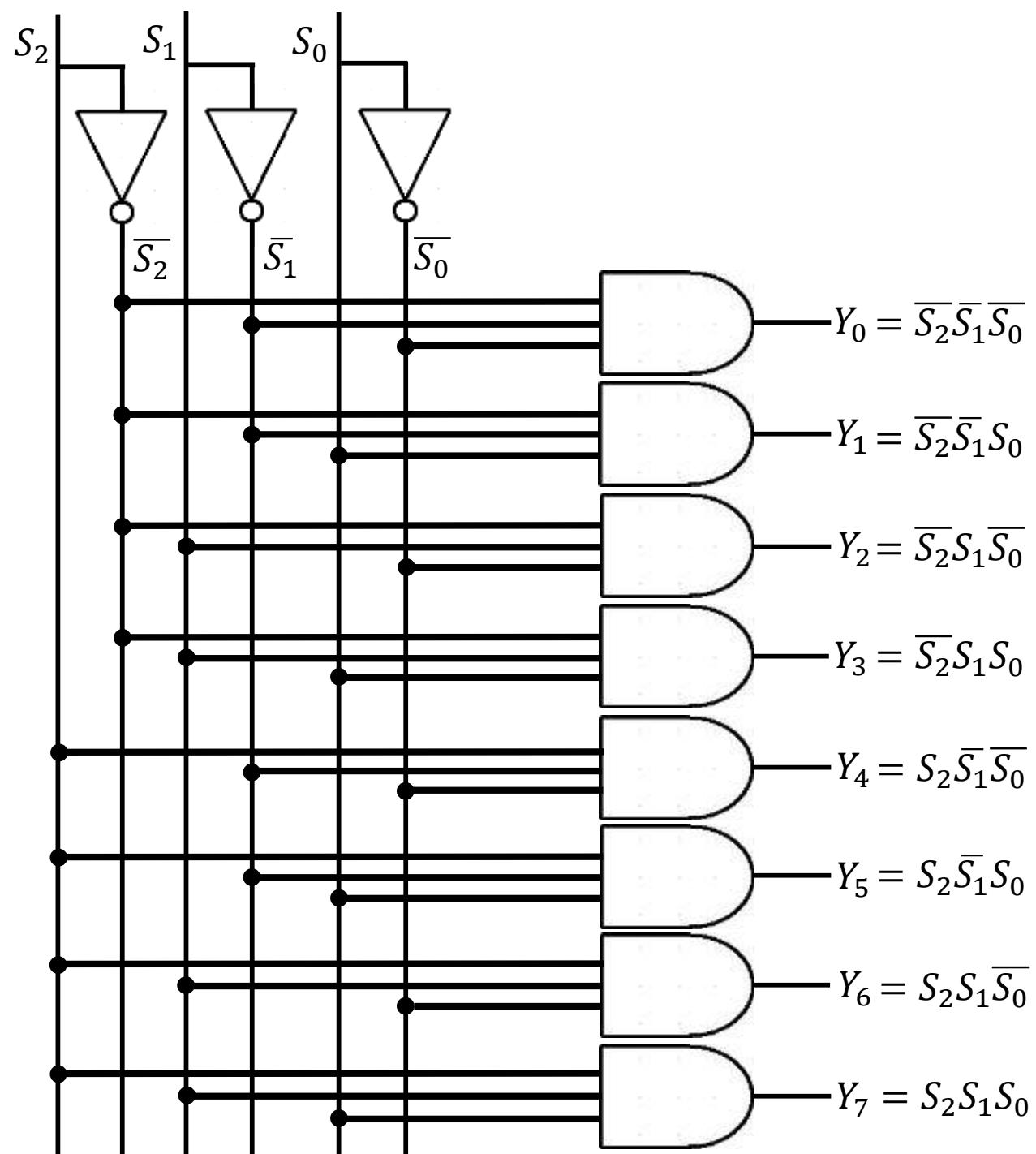
$$Y_5 = S_2 \overline{S_1} S_0$$

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

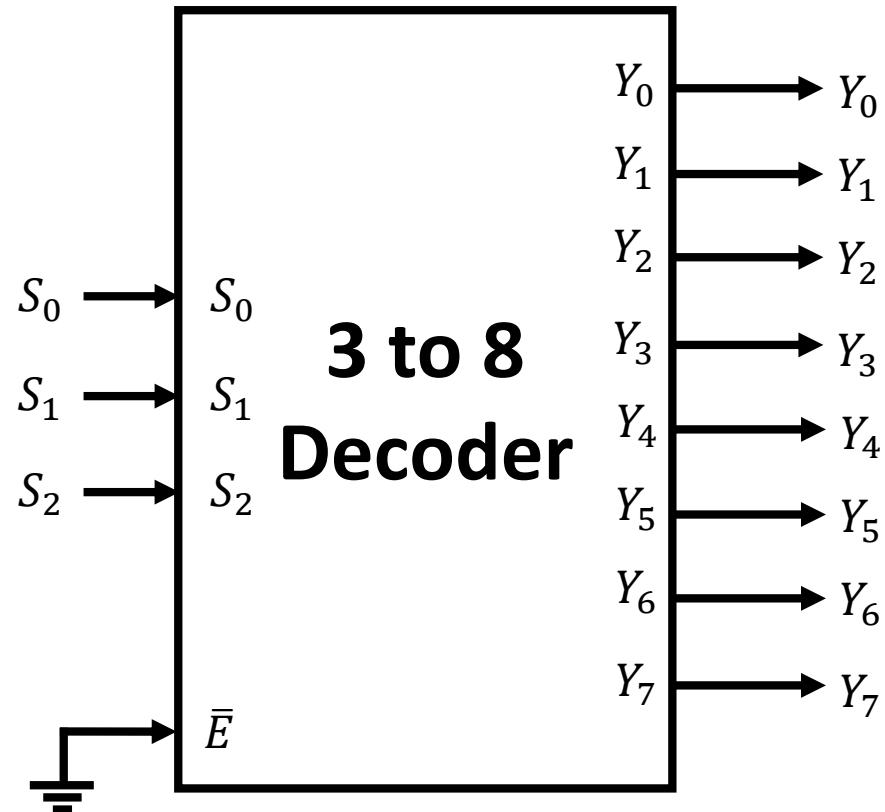
$$Y_6 = S_2 S_1 \overline{S_0}$$

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

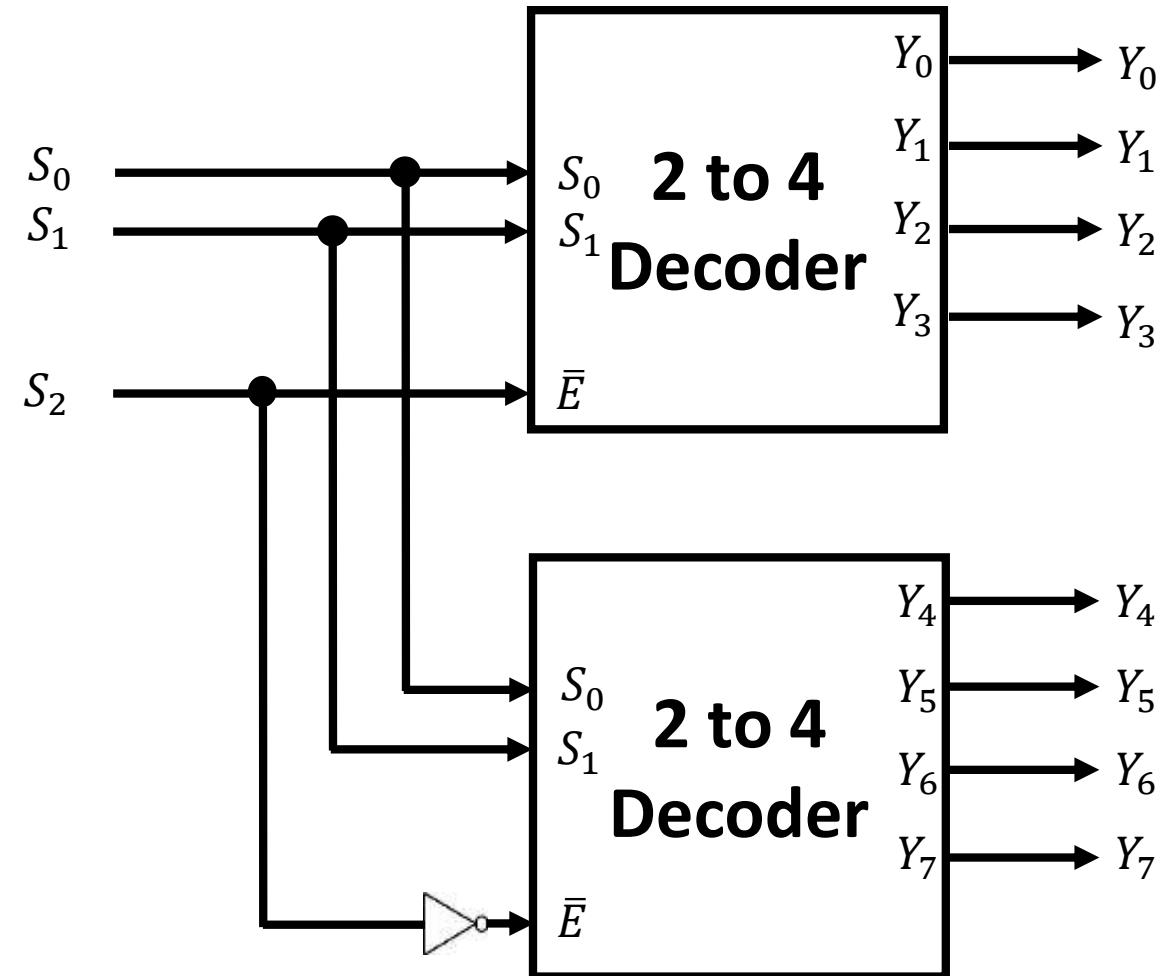
$$Y_7 = S_2 S_1 S_0$$



# 3 to 8 Line Decoder



# Design of 3 to 8 line Decoder Using 2 to 4 line Decoder



# Implementation of Full Adder using Decoder

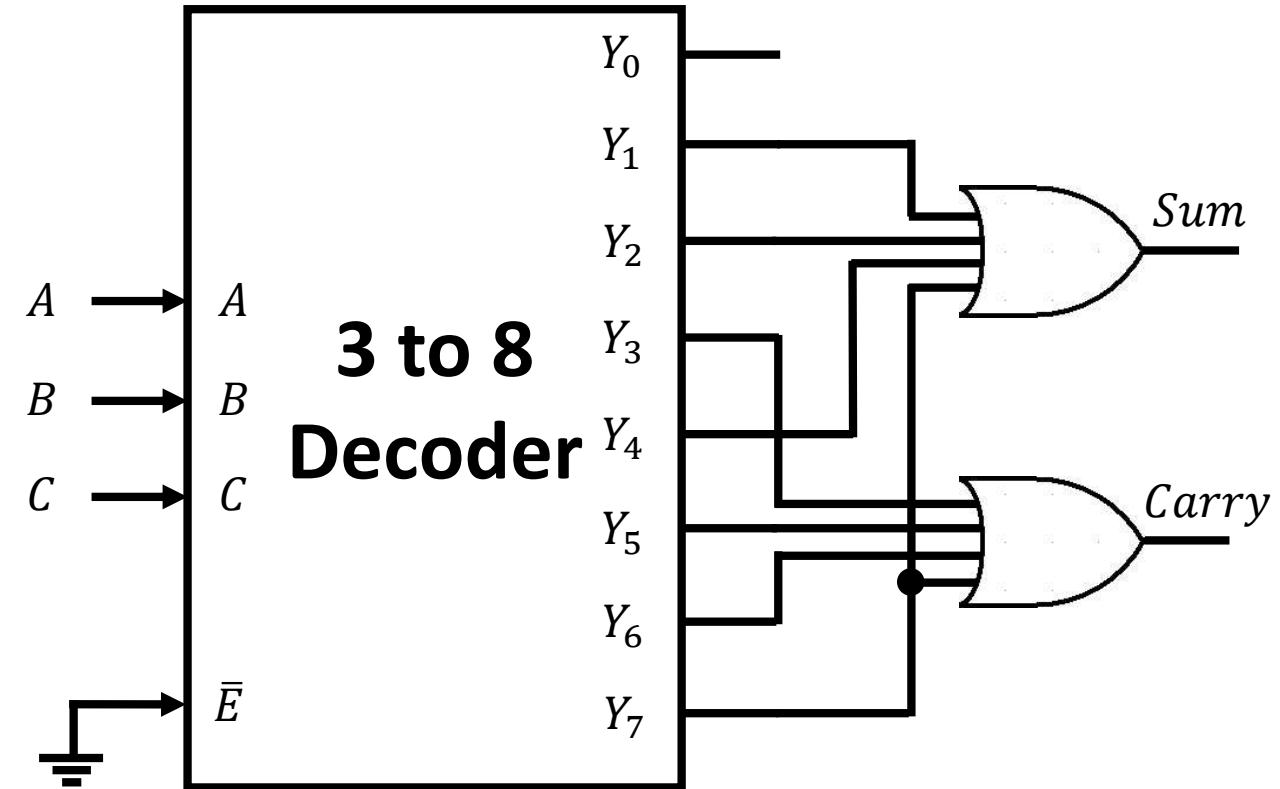
Truth Table

Decimal	Inputs			Outputs	
	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Output Expressions are,

$$\text{Sum} = \sum_m(1,2,4,7)$$

$$\text{Carry} = \sum_m(3,5,6,7)$$



# Implementation of Full Subtractor using Demultiplexer

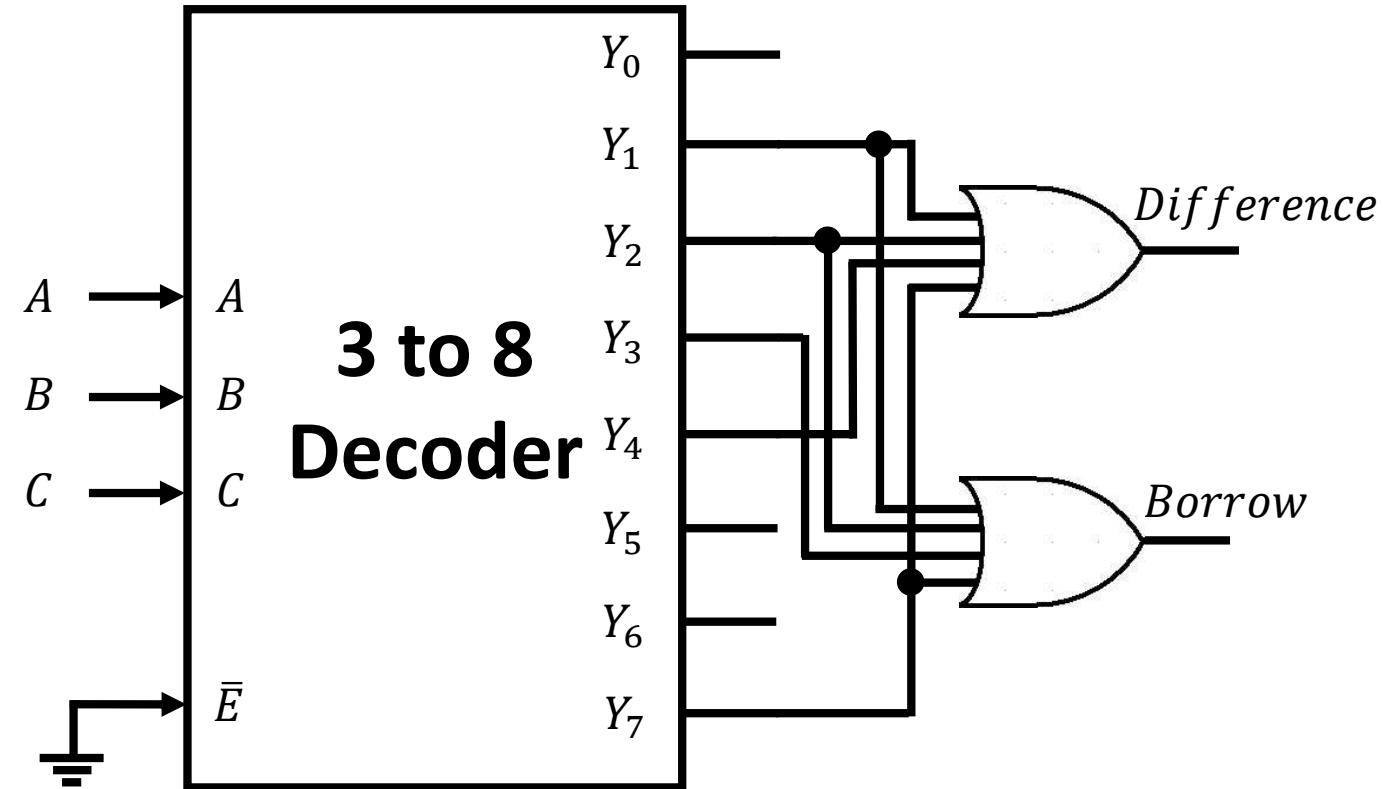
Truth Table

Decimal	Inputs			Outputs	
	A	B	C	Difference	Borrow
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

Output Expressions are,

$$\text{Difference} = \sum_m(1,2,4,7)$$

$$\text{Borrow} = \sum_m(1,2,3,7)$$



# Implementation of given Boolean function using Decoder

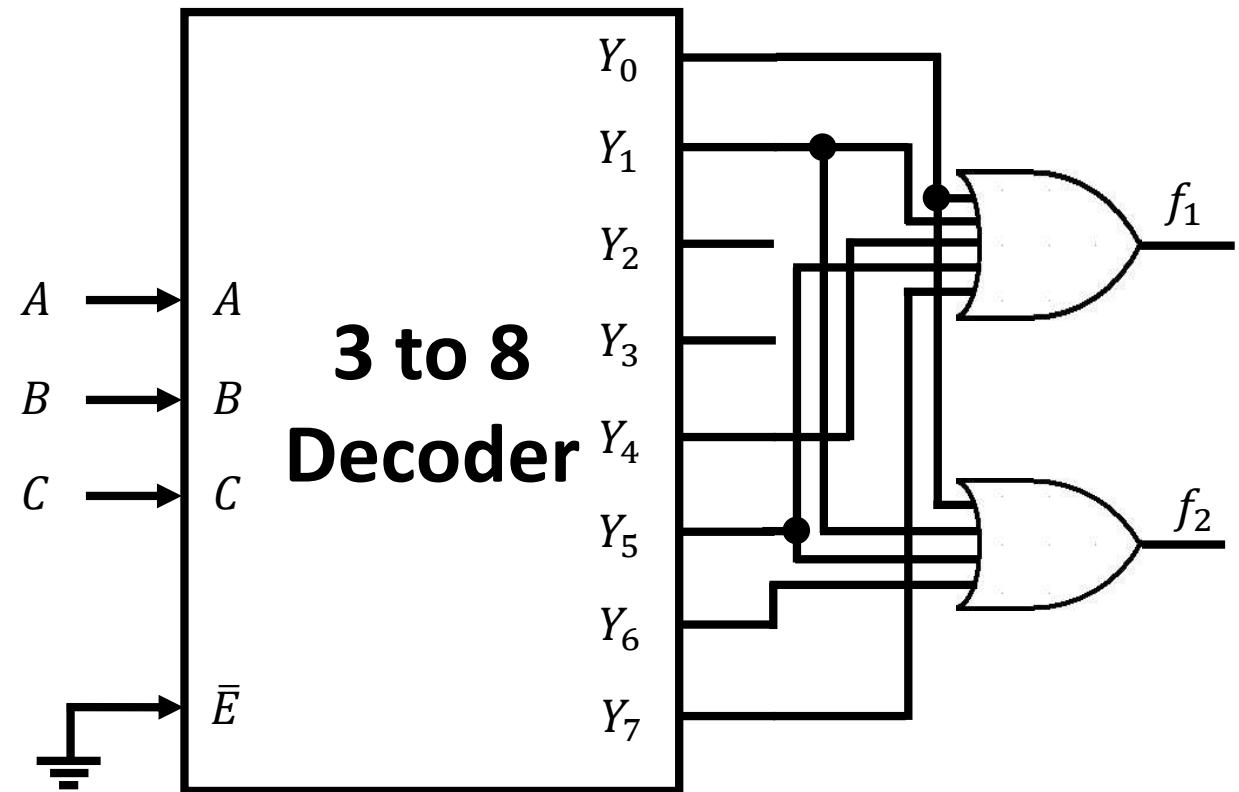
$$f_1 = \sum_m(0,1,4,5,7)$$

$$f_2 = \sum_m(0,1,5,6)$$

Given Expressions are,

$$f_1 = \sum_m(0,1,4,5,7)$$

$$f_2 = \sum_m(0,1,5,6)$$



# Implementation of given Boolean function using Decoder

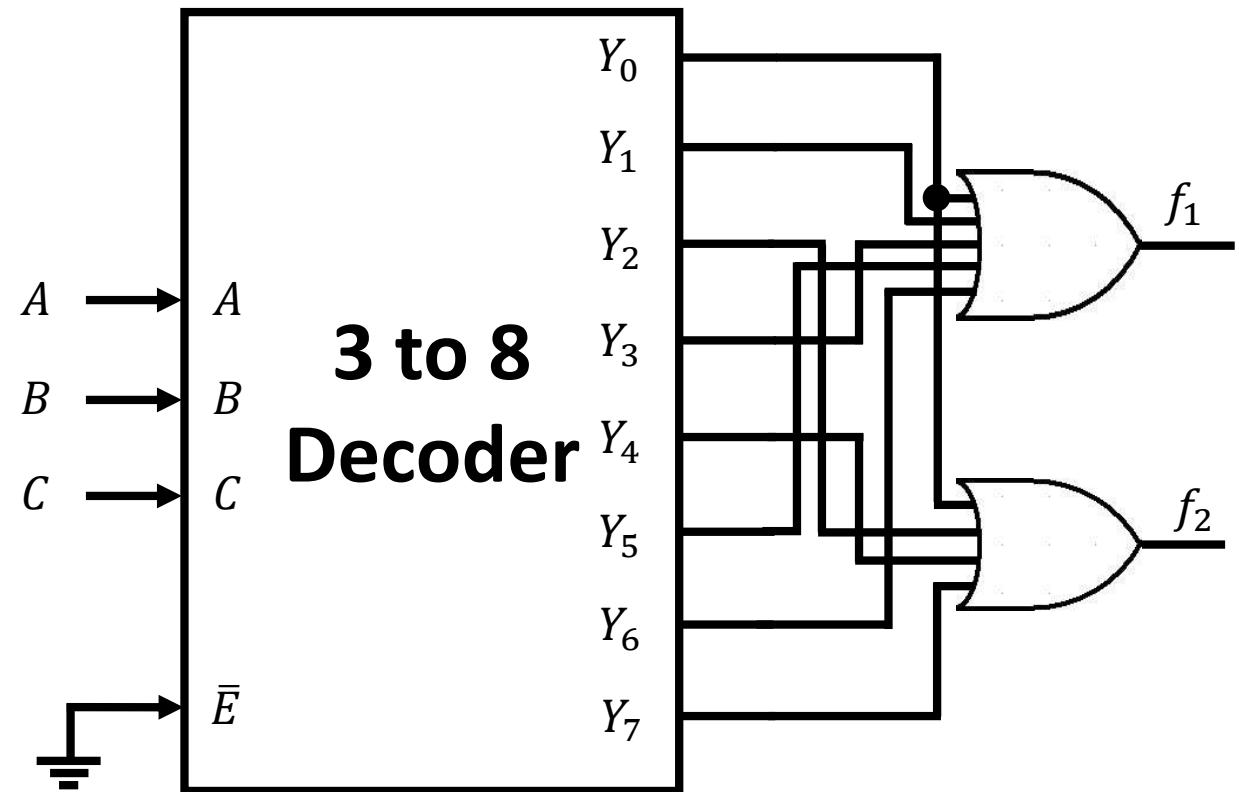
$$f_1 = \sum_m(0,1,3,5,6)$$

$$f_2 = \sum_m(0,2,4,7)$$

Given Expressions are,

$$f_1 = \sum_m(0,1,3,5,6)$$

$$f_2 = \sum_m(0,2,4,7)$$



# Applications of Decoder

- It can be used to implement combinational circuit
- It can be used to convert BCD into 7-segment code
- It is used in memories to select particular register

**THANK YOU...**

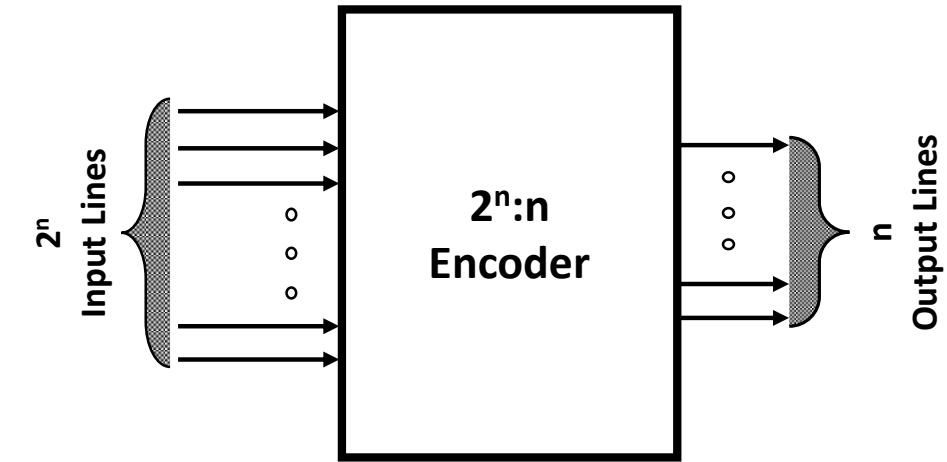
**ANY QUERIES???**

# **ENCODER**

**Prepared By**  
**Mr.P.KANAKARAJ**  
**AP/EEE**  
**SRMIST-KTR**

# Encoder:

- It is a combinational logic circuit.
- An encoder has  $2^n$  input lines and  $n$  output lines.
- The  $n$  output lines generate the binary code for the possible  $2^n$  input lines.
- Input –  $2^n$
- Output –  $n$
- Encoder –  $2^n : n$



# Four to Two Line Binary Encoder

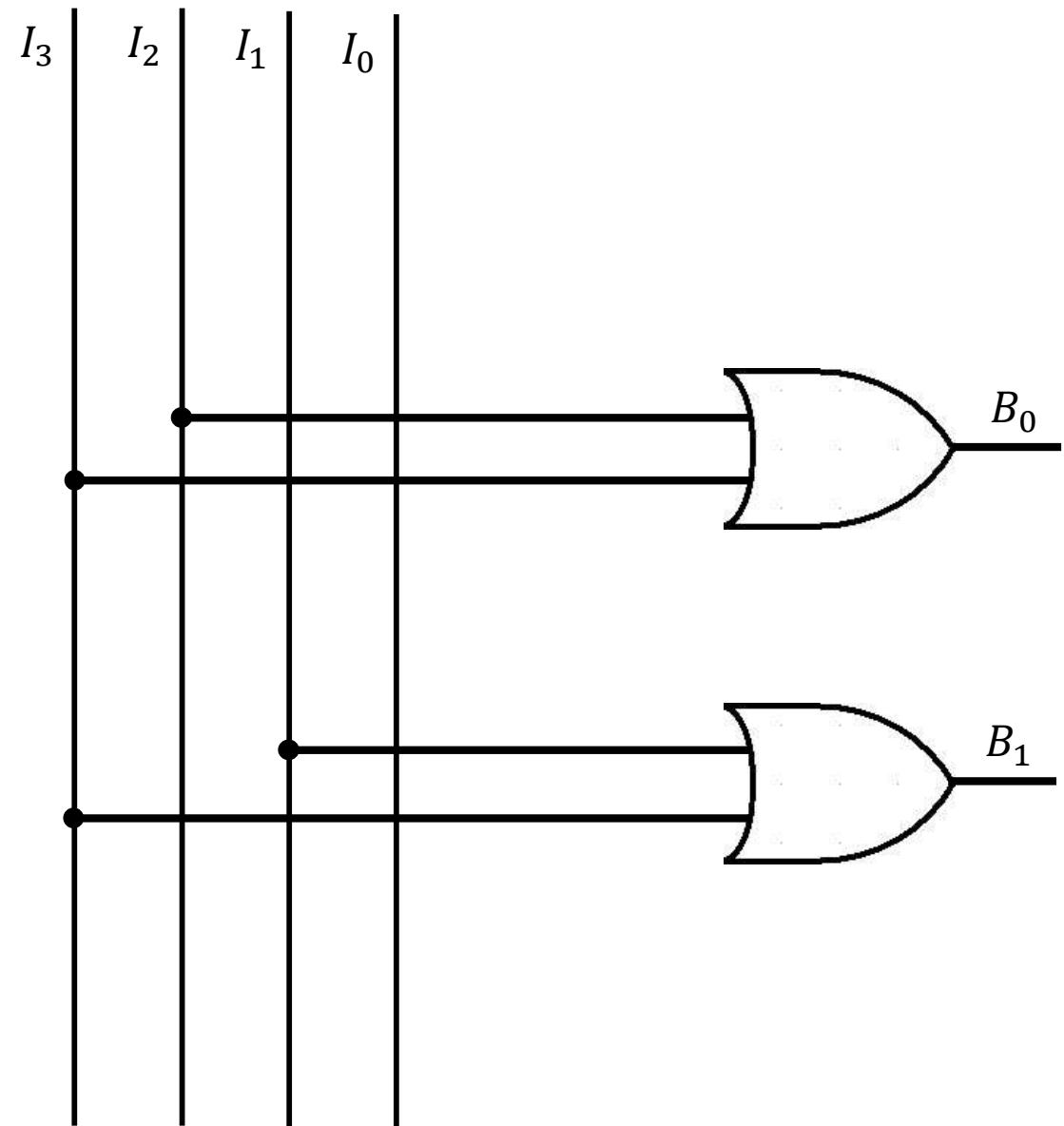
Truth Table

Decimal Value	Input				Output	
	$I_3$	$I_2$	$I_1$	$I_0$	$B_1$	$B_0$
0	0	0	0	1	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	0
3	1	0	0	0	1	1

- Input – 4
- Output – 2
- Outputs are,

$$B_0 = I_2 + I_3$$

$$B_1 = I_1 + I_3$$

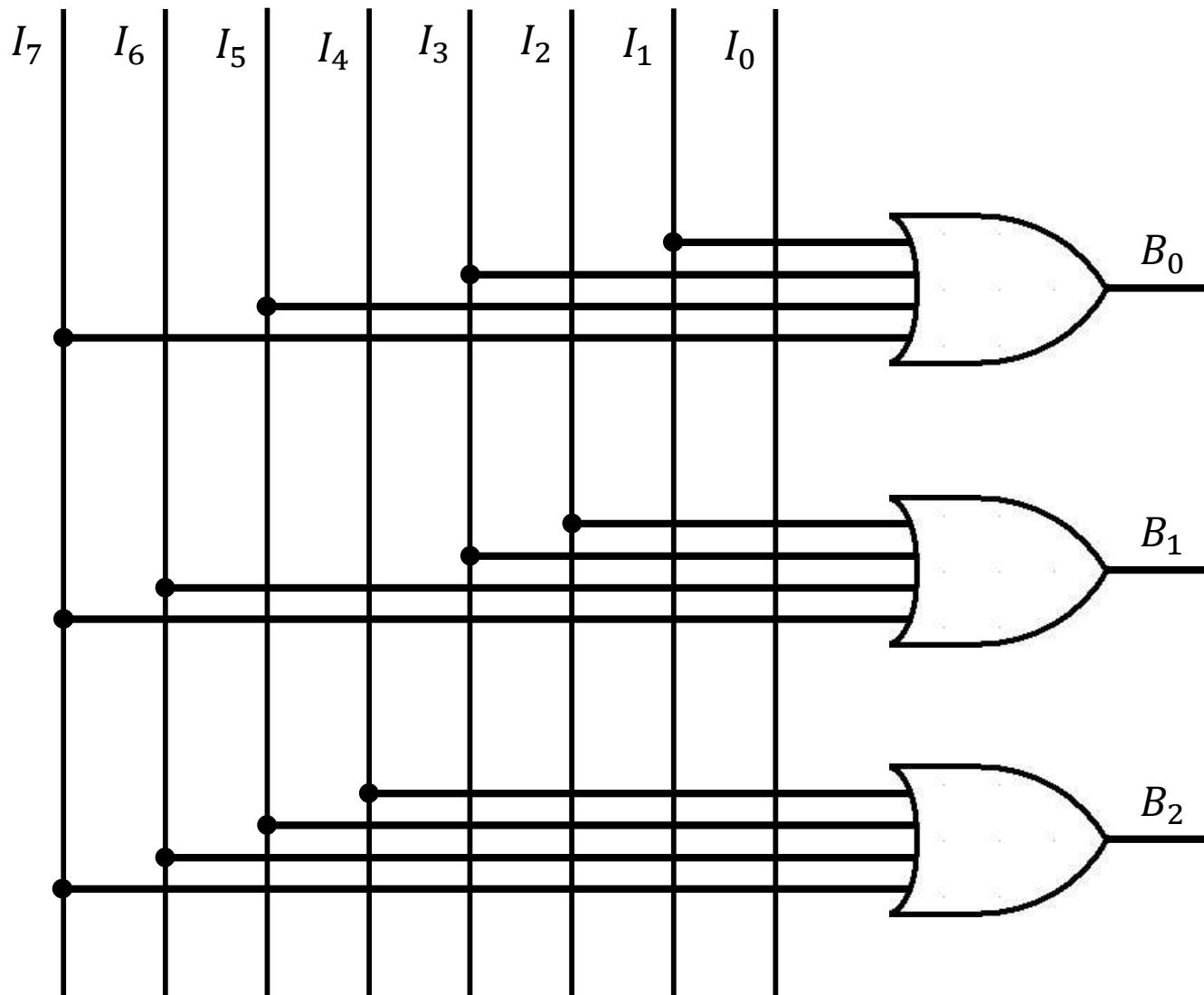


# Eight to Three Line Binary Encoder (Octal to Binary Encoder)

## Truth Table

Decimal Value	Input									Output		
	$I_7$	$I_6$	$I_5$	$I_4$	$I_3$	$I_2$	$I_1$	$I_0$	$B_2$	$B_1$	$B_0$	
0	0	0	0	0	0	0	0	1	0	0	0	
1	0	0	0	0	0	0	1	0	0	0	1	
2	0	0	0	0	0	1	0	0	0	1	0	
3	0	0	0	0	1	0	0	0	0	1	1	
4	0	0	0	1	0	0	0	0	1	0	0	
5	0	0	1	0	0	0	0	0	1	0	1	
6	0	1	0	0	0	0	0	0	1	1	0	
7	1	0	0	0	0	0	0	0	1	1	1	

- Input – 8  $B_0 = I_1 + I_3 + I_5 + I_7$
  - Output – 3  $B_1 = I_2 + I_3 + I_6 + I_7$
  - Outputs are,  $B_2 = I_4 + I_5 + I_6 + I_7$



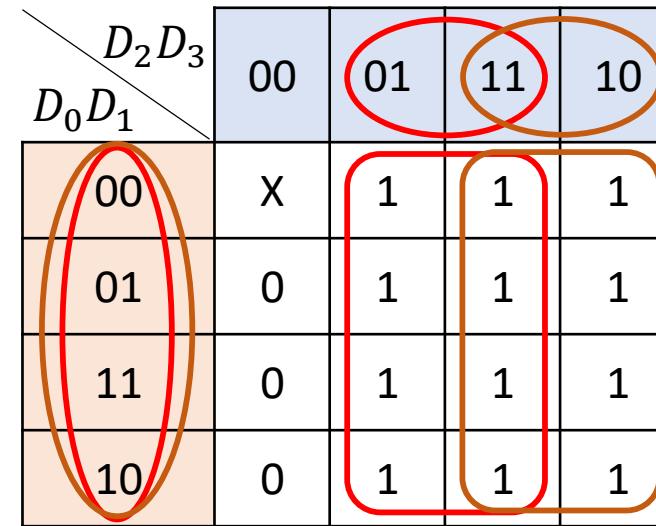
# Priority Encoder:

- It is an encoder circuit that includes the priority function.
- In that priority encoder, if two or more inputs are equal to 1 at the same time, the input having the highest priority will take precedence.

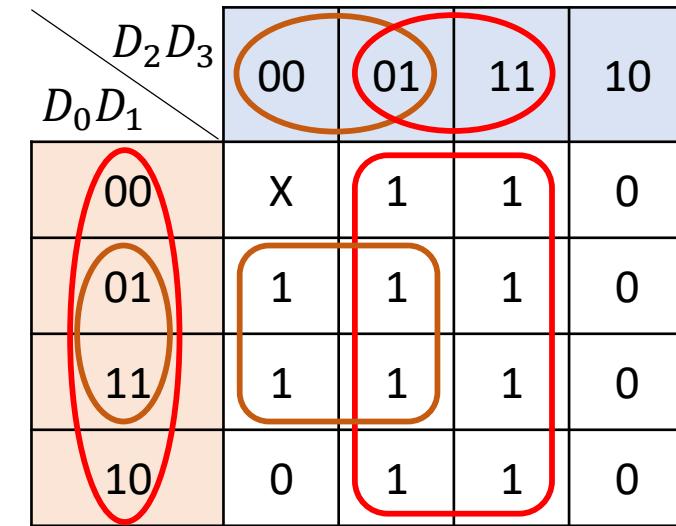
Truth Table

Input				Output		
$D_0$	$D_1$	$D_2$	$D_3$	$B_1$	$B_0$	$V$
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

$B_1$ : K-Map Simplification



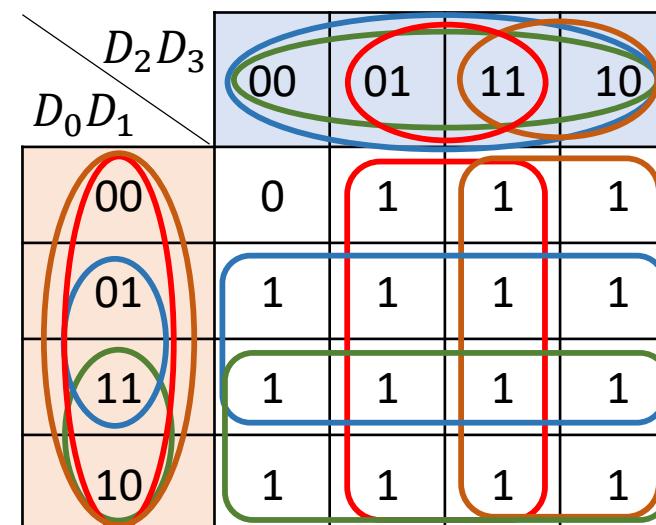
$B_0$ :



$$B_1 = D_2 + D_3$$

$V:$

$$B_0 = D_3 + D_1 \overline{D_2}$$



$$V = D_0 + D_1 + D_2 + D_3$$

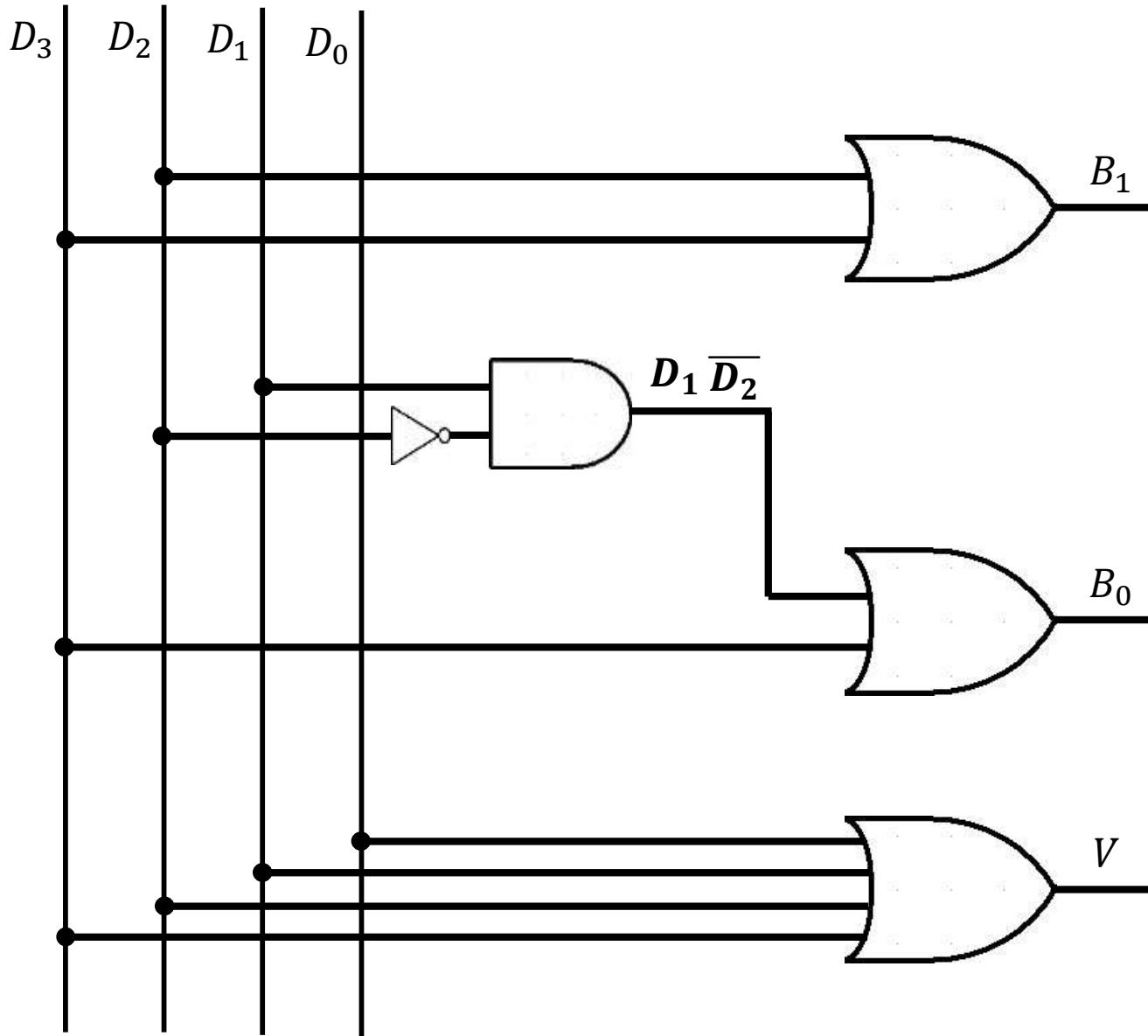
# Logic Diagram for Priority Encoder

- Expressions are,

$$B_1 = D_2 + D_3$$

$$B_0 = D_3 + D_1 \overline{D}_2$$

$$V = D_0 + D_1 + D_2 + D_3$$



**THANK YOU...**

**ANY QUERIES???**

Unit - 3

# Combinational Logic Circuits

# Contents

- ❖ Binary adder
- ❖ Binary adder as subtractor
- ❖ Carry look ahead adder
- ❖ Decimal adder
- ❖ Magnitude Comparator

# Binary adder



**Half Adder:** is a combinational circuit that performs the addition of two bits, this circuit needs two binary inputs and two binary outputs.

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

**Truth table**

The simplified Boolean function from the truth table:

$$\begin{cases} S = \bar{X}Y + X\bar{Y} \\ C = XY \end{cases} \quad 1 \} \text{ (Using sum of product form)}$$

Where **S** is the sum and **C** is the carry.

$$\begin{cases} S = X \oplus Y \\ C = XY \end{cases} \quad 2 \} \text{ (Using XOR and AND Gates)}$$

# Half adder

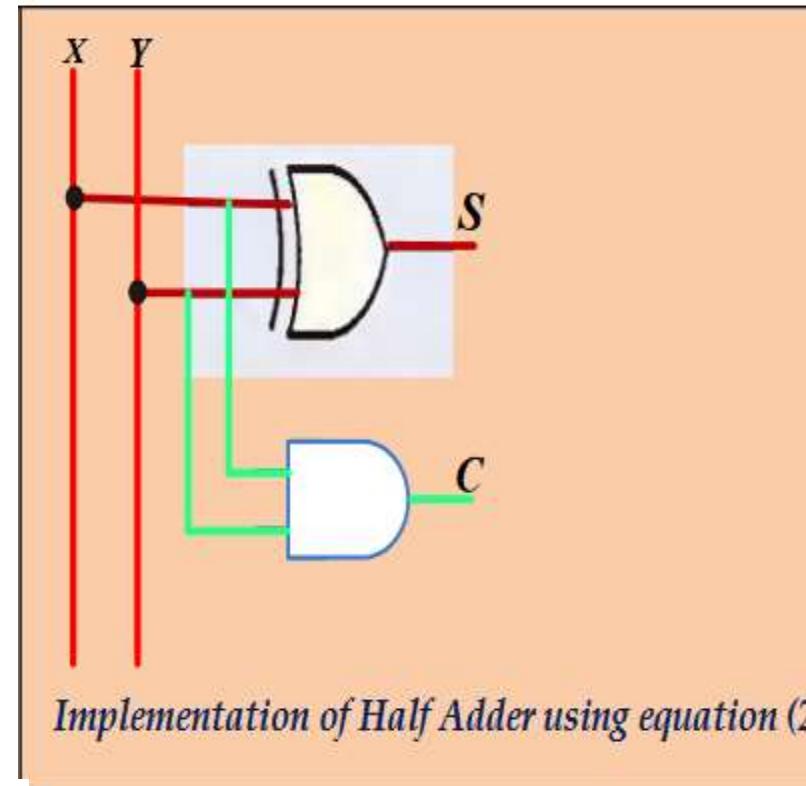
K-map for half adder

	Y 0	Y 1
X 0	0	1
X 1	2	3

K-map for Carry

	Y 0	Y 1
X 0	0	1
X 1	1	2

K-map for Sum



$$\left. \begin{array}{l} S = X \oplus Y \\ C = XY \end{array} \right\} \text{(Using XOR and AND Gates)}$$

# Binary Adder

- Full Adder

- Adds 1-bit plus 1-bit plus 1-bit
- Produces Sum and Carry



$x$	$y$	$z$	$C$	$S$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

			$y$	
	0	1	0	
$x$	1	0	1	0
			$z$	

$$\begin{array}{r}
 x \\
 + y \\
 + z \\
 \hline
 C \quad S
 \end{array}$$

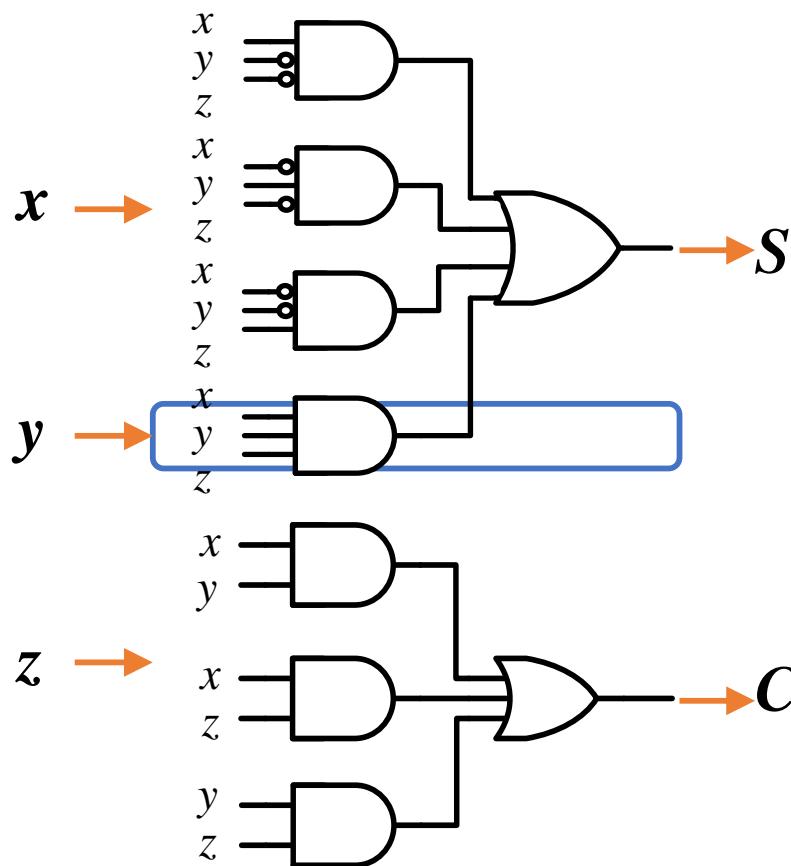
$$S = xy'z' + x'y'z' + x'y'z + xyz = x \oplus y \oplus z$$

		$y$		
	0	1	0	
$x$	0	1	1	1
		$z$		

$$C = xy + xz + yz$$

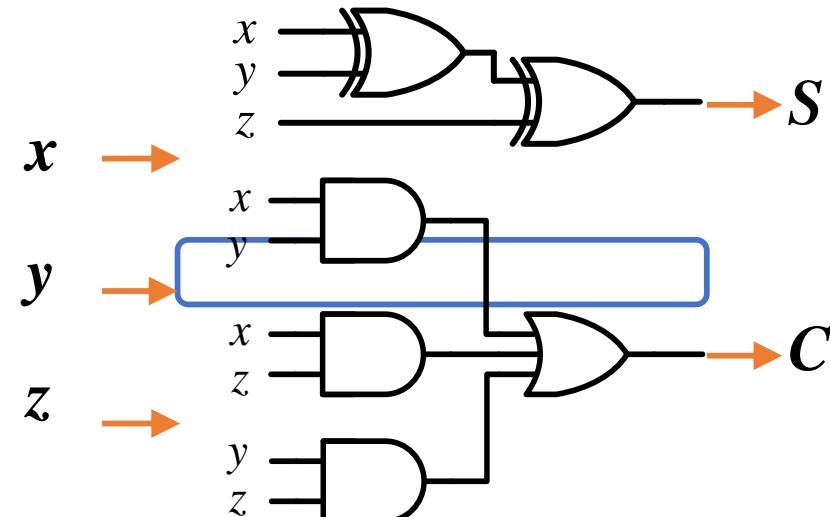
# Binary Adder

- Full Adder



$$S = xy'z' + x'yz' + x'y'z + xyz = x \oplus y \oplus z$$

$$C = xy + xz + yz$$



# Binary Adder

Full adder design from two half adder

***The sum:***

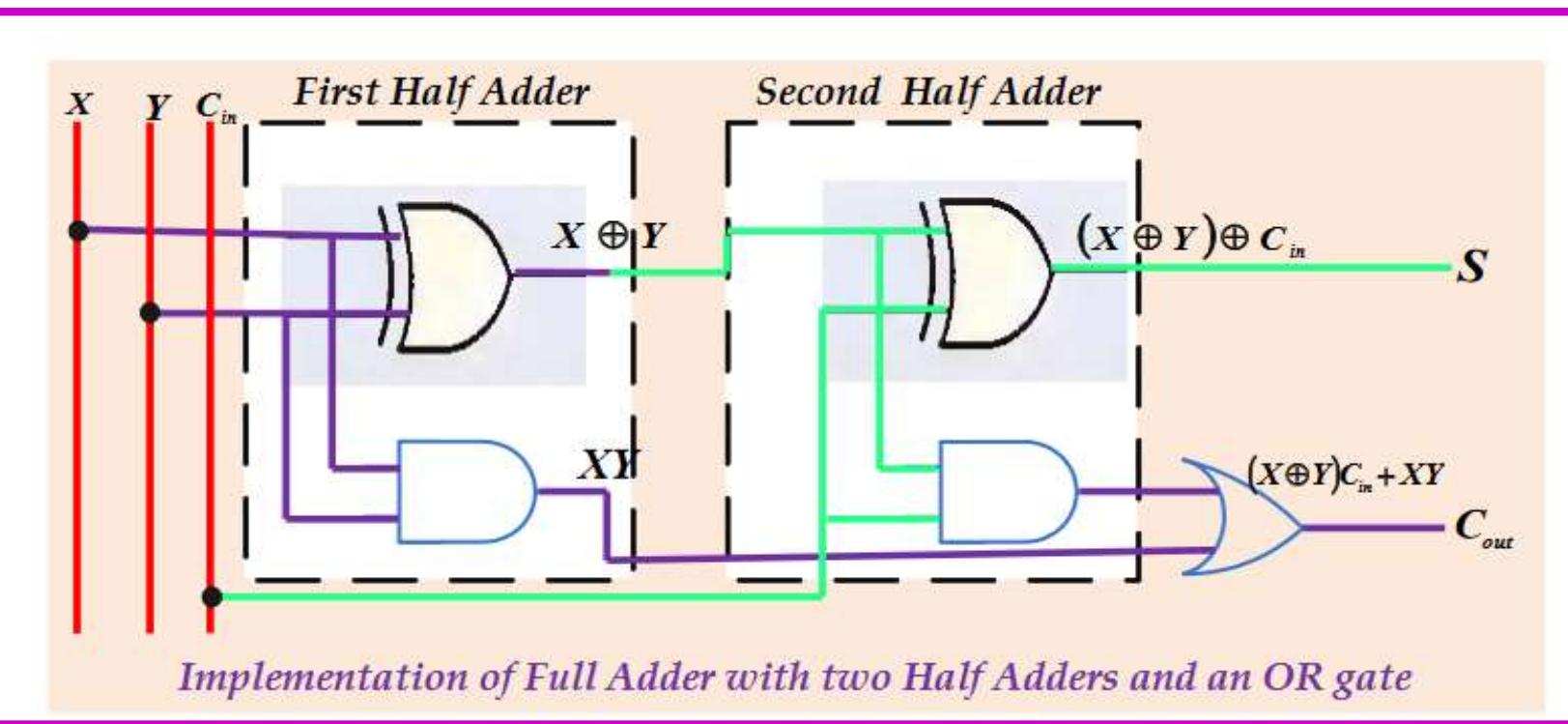
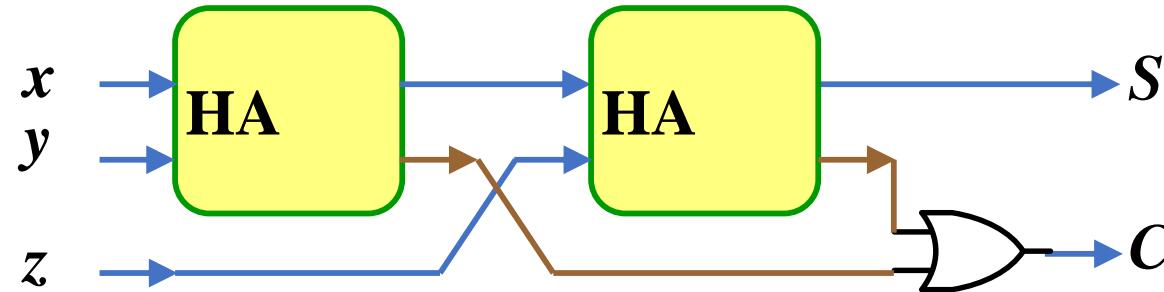
$$\begin{aligned} S &= \bar{X}\bar{Y}C_{in} + \bar{X}Y\bar{C}_{in} + X\bar{Y}\bar{C}_{in} + XYC_{in} \\ &= \bar{C}_{in}(XY + X\bar{Y}) + C_{in}(\bar{X}\bar{Y} + X\bar{Y}) \\ &= \bar{C}_{in}(XY + X\bar{Y}) + C_{in}(\overline{\bar{X}Y + X\bar{Y}}) \\ S &= C_{in} \oplus (X \oplus Y) \end{aligned}$$

***The carry output:***

$$\begin{aligned} C_{out} &= \bar{X}Y\bar{C}_{in} + X\bar{Y}C_{in} + XY\bar{C}_{in} + X\bar{Y}\bar{C}_{in} \\ &= C_{in}(\bar{X}Y + X\bar{Y}) + XY(C_{in} + \bar{C}_{in}) \\ C_{out} &= C_{in} \cdot (X \oplus Y) + XY \end{aligned}$$

# Binary Adder

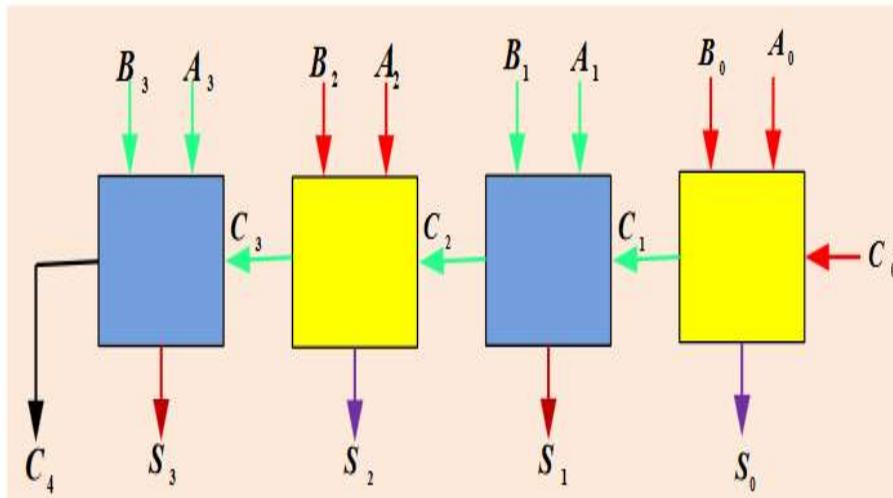
- Full Adder



# Binary Adder

## Binary Adder (Asynchronous Ripple-Carry Adder)

- A binary adder is a digital circuit that produces the *arithmetic sum of two binary numbers*.
- A binary adder can be constructed with *full adders connected in cascade* with the output carry from each full adder connected to the input carry of the next full adder in the chain.
- The *four-bit adder* is a typical example of a *standard component*. It can be used in many applications involving arithmetic operations.



$$\begin{array}{r} c_3 \ c_2 \ c_1 \\ + x_3 \ x_2 \ x_1 \ x_0 \\ + y_3 \ y_2 \ y_1 \ y_0 \\ \hline cy \ s_3 \ s_2 \ s_1 \ s_0 \end{array}$$

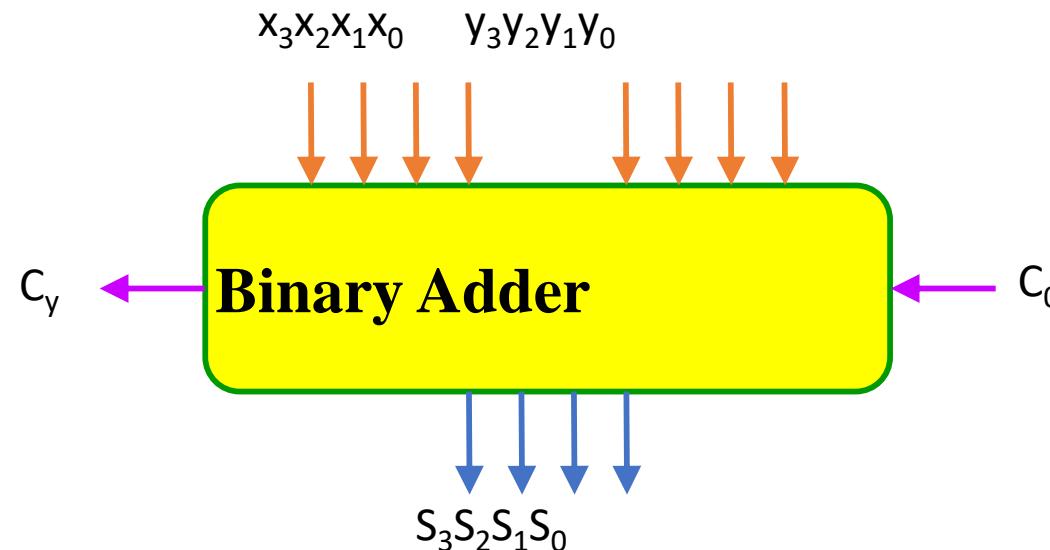
Example:

$$A + B \quad (A = 1011) \text{ and } (B = 0011)$$

Subscript $i$	3	2	1	0	
Input Carry	0	1	1	0	$C_i$
A	1	0	1	1	$A_i$
+	+ sign				
B	0	0	1	1	$B_i$
Sum	1	1	1	0	$S_i$
Output Carry	0	0	1	1	$C_{i+1}$

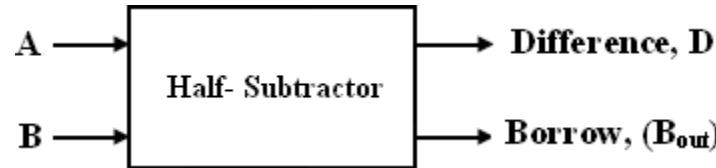
# Binary Adder

- The input carry to the adder is  $C_0$  and it ripples through the full adders to the output carry  $C_4$ .
- $n$ -bit binary adder requires  $n$  full adders.



# Binary Subtractor

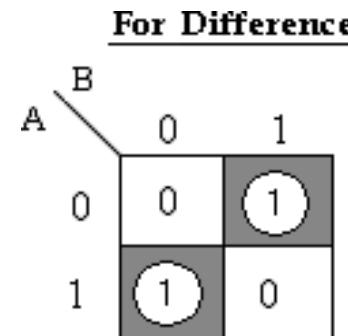
## Block schematic of half-subtractor



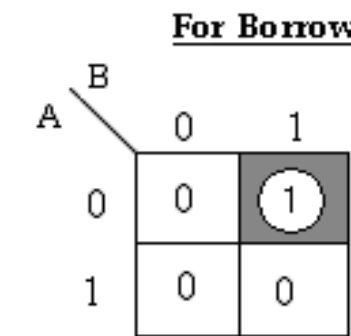
Logical expression

## Truth table

Input		Output	
A	B	Difference (D)	Borrow (Bout)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

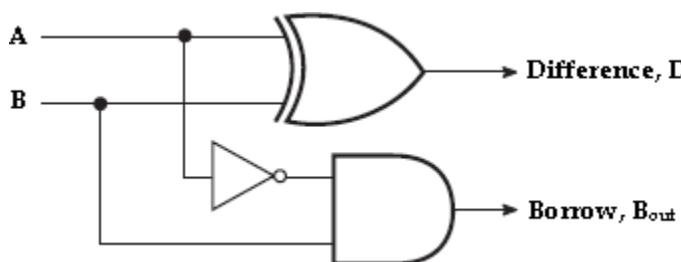


$$\text{Difference} = AB' + A'B$$
$$= A \oplus B$$



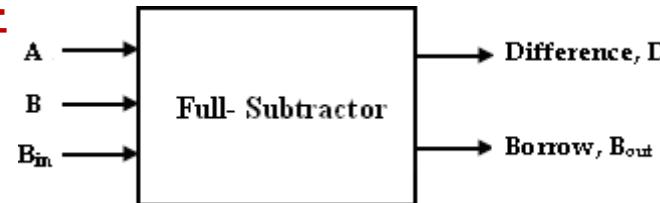
$$\text{Borrow} = \overline{A} \cdot B$$

## Logical design



# Binary Subtractor

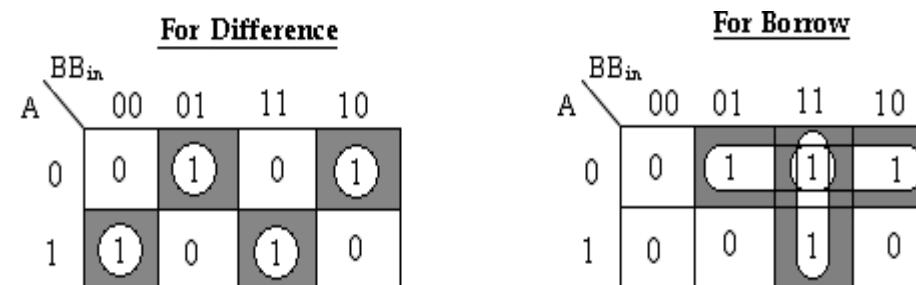
## Block schematic of full-subtractor



## Truth table

Inputs			Outputs	
A	B	B <sub>in</sub>	Difference(D)	Borrow(B <sub>out</sub> )
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

## Logical expression

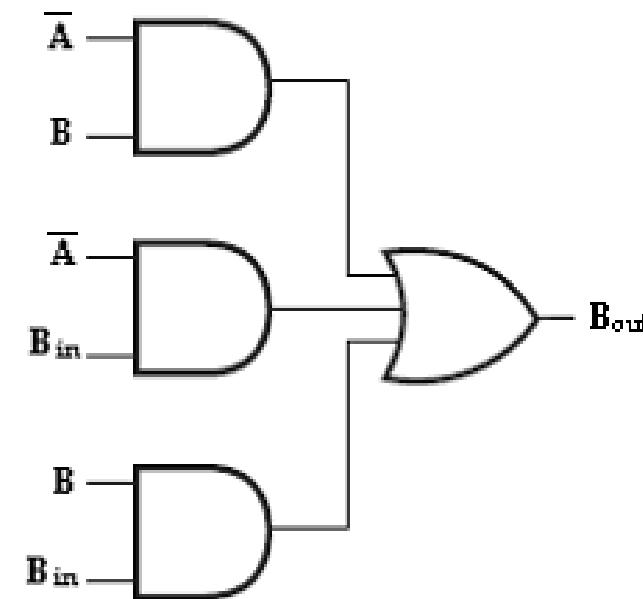
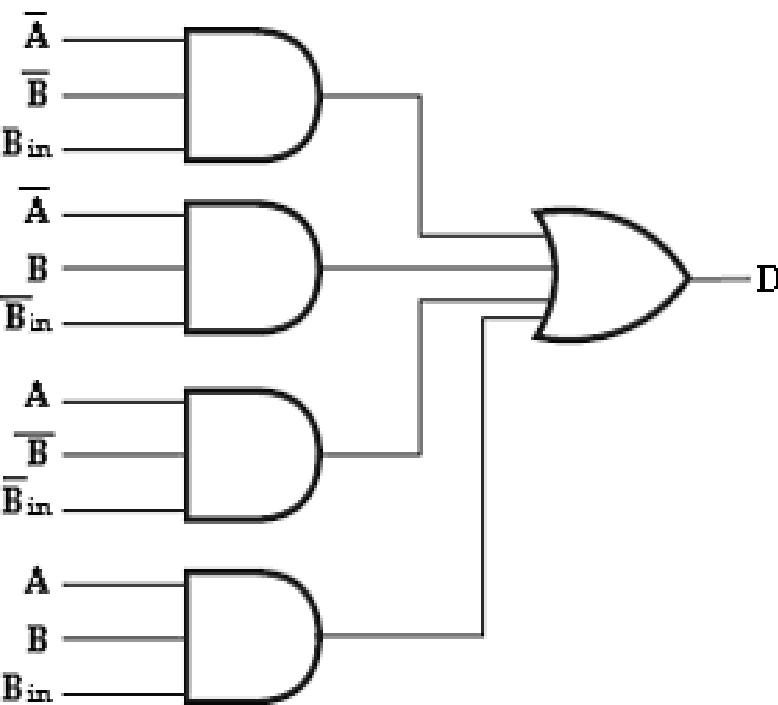


$$\text{Difference, } D = A'B'B_{\text{in}} + A'BB'_{\text{in}} + AB'B'_{\text{in}} + ABB_{\text{in}}$$

$$\text{Borrow, } B_{\text{out}} = A'B + A'B_{\text{in}} + BB_{\text{in}}$$

# Binary Subtractor

## Logical design



# Binary Subtractor

## ❖ 1's Complement of a Binary Number

There is a simple algorithm to convert a binary number into 1's complement. To get 1's complement of a binary number, simply invert the given number.

## ❖ 2's Complement of a Binary Number

There is a simple algorithm to convert a binary number into 2's complement. To get 2's complement of a binary number, simply invert the given number and add 1 to the least significant bit (LSB) of given result.

### Example

Binary representation of 3 is: 0 0 1 1

1's Complement of 3 is: 1 1 0 0

2's Complement of 3 is: (1's Complement + 1) i.e.

1 1 0 0 (1's Compliment)

+ 1

1 1 0 1 (2's Complement i.e. -3)

Now  $2 + (-3) = 0 0 1 0$  (2 in binary)

+ 1 1 0 1 (-3)

1 1 1 1 (-1)

Now, to check whether it is -1 or not simply. takes 2's Complement of -1 and kept -ve sign as it is.

-1 = 1 1 1 1

2's Complement = -(0 0 0 0)

+ 1

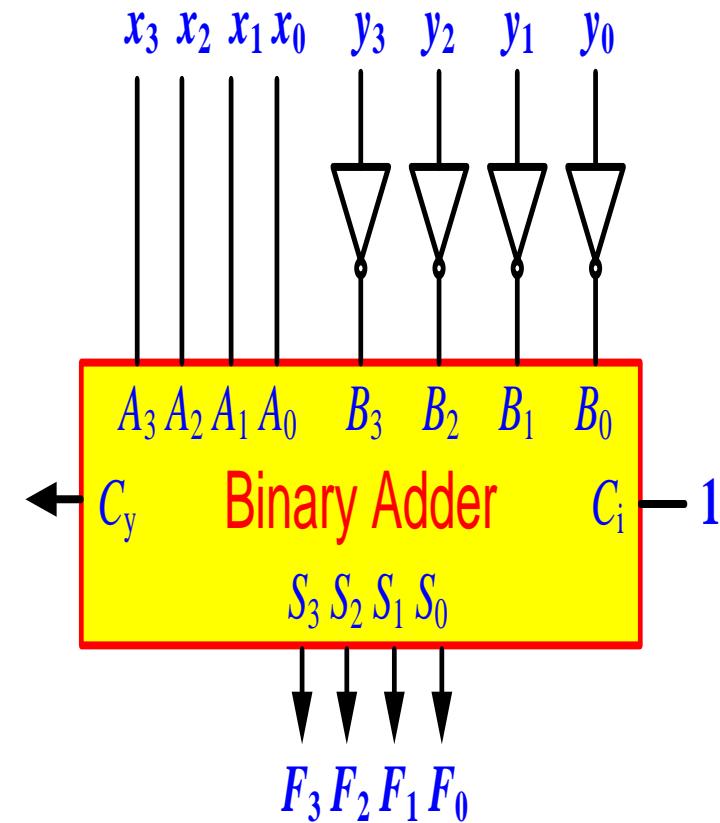
-(0 0 0 1) i.e. -1

# Binary Subtractor

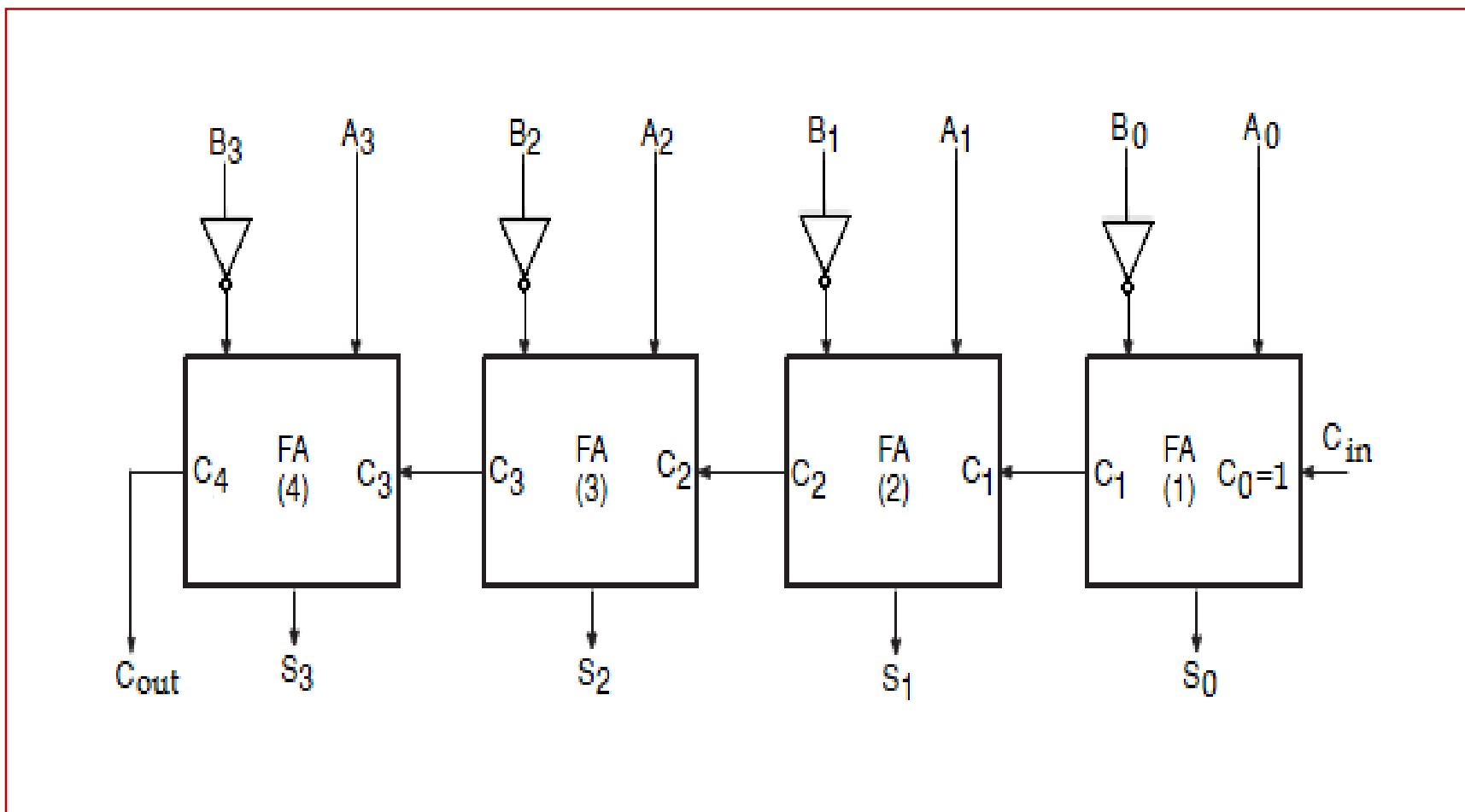
- Use 2's complement with binary adder
  - $x - y = x + (-y) = x + y' + 1$

## Binary Subtractor

- To perform the subtraction  $-B$ , we can use the **2's complements**, so the subtraction can be converted to addition.
- **2's complement** can be obtained by taking the **1's complement** and adding **1** to the **LSD** bit.
  - 1) **1's complement** can be implemented with invertors.
  - 2) **1** can be added to the sum through the input carry.
- The circuit for subtracting  $A - B$  consists of an adder with inverters placed between each data input  $B$  and the corresponding input of the full adder. The input carry  $C_0$  must be equal to **1**.



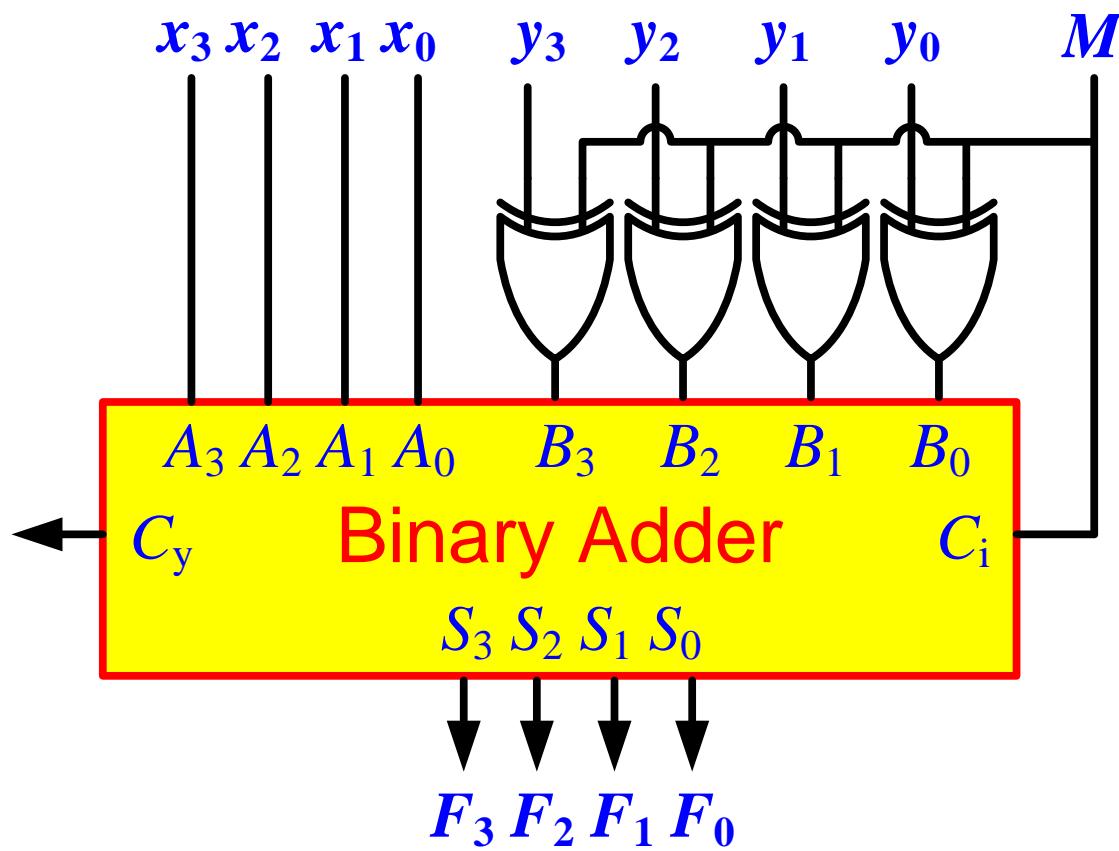
# Binary Subtractor



# Binary Adder/Subtractor

- $M$ : Control Signal (Mode)

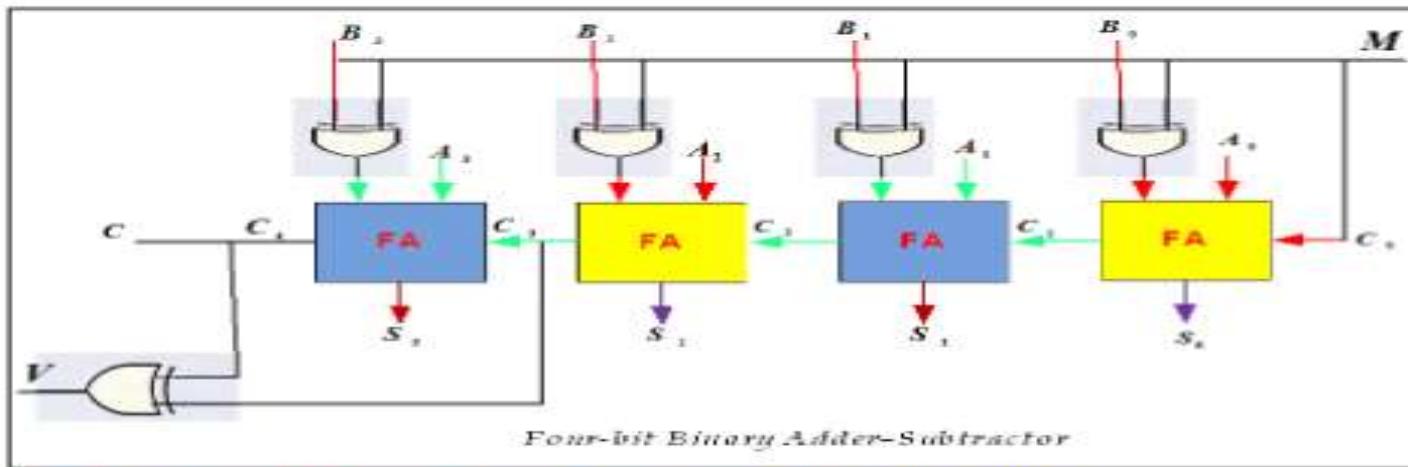
- $M=0 \rightarrow F = x + y$
- $M=1 \rightarrow F = x - y$



# Binary Subtractor

## Binary Adder–Subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder by including an *exclusive-OR* gate with each full-adder.



The mode input  $M$  controls the operation as the following:

- ( $M = 0 \rightarrow$  adder).
- ( $M = 1 \rightarrow$  subtractor).
- Each *XOR* gate receives  $M$  signal and  $B$ .
  - When  $M = 0$  then  $B \oplus 0 = B$  and the carry = 0, then the circuit performs the operation  $A + B$ .
  - When  $M = 1$  then  $B \oplus 1 = \bar{B}$  and the carry = 1, then the circuit performs the operation  $A - B$ .
- The *exclusive-OR* with output  $V$  is for detecting an overflow.

# Carry look ahead Adder

- The **carry propagation time** is an important attribute of the adder because it limits the speed with which two numbers are added.
- To reduce the carry propagation delay time:
  - 1) Employ faster gates with reduced delays.
  - 2) Employ the principle of **Carry Lookahead Logic**.

**Proof:** (using carry lookahead logic)

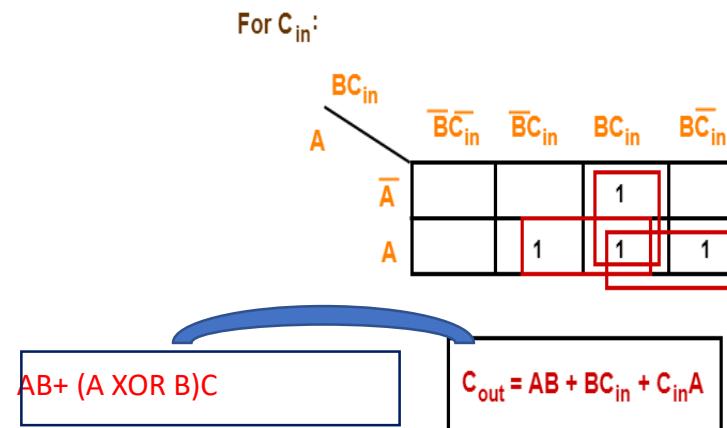
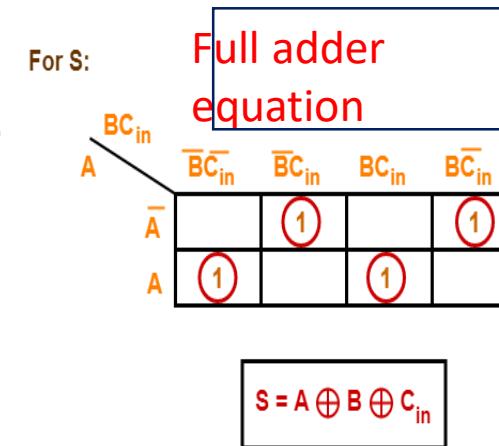
$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

The output sum and carry are:

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$



# Carry look ahead Adder

- ✓  $G_i$ -called a **carry generate**, and it produces a carry of 1 when both  $A_i$  and  $B_i$  are 1.
- ✓  $P_i$ -called a **carry propagate**, it determines whether a carry into stage  $i$  will propagate into stage  $i + 1$ .
- ✓ The **Boolean function** for the carry outputs of each stage and substitute the value of each  $C_i$  from the previous equations:

$$\left\{ \begin{array}{l} C_0 = \text{input carry} \\ C_1 = G_0 + P_0 C_0 \\ C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) \\ \quad = G_1 + P_1 G_0 + P_1 P_0 C_0 \\ C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_0) \\ \quad = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \end{array} \right\}$$

$P_i = A_i \oplus B_i$   
 $G_i = A_i B_i$   
The output sum and carry are:  
 $S_i = P_i \oplus C_i$   
 $C_{i+1} = G_i + P_i C_i$

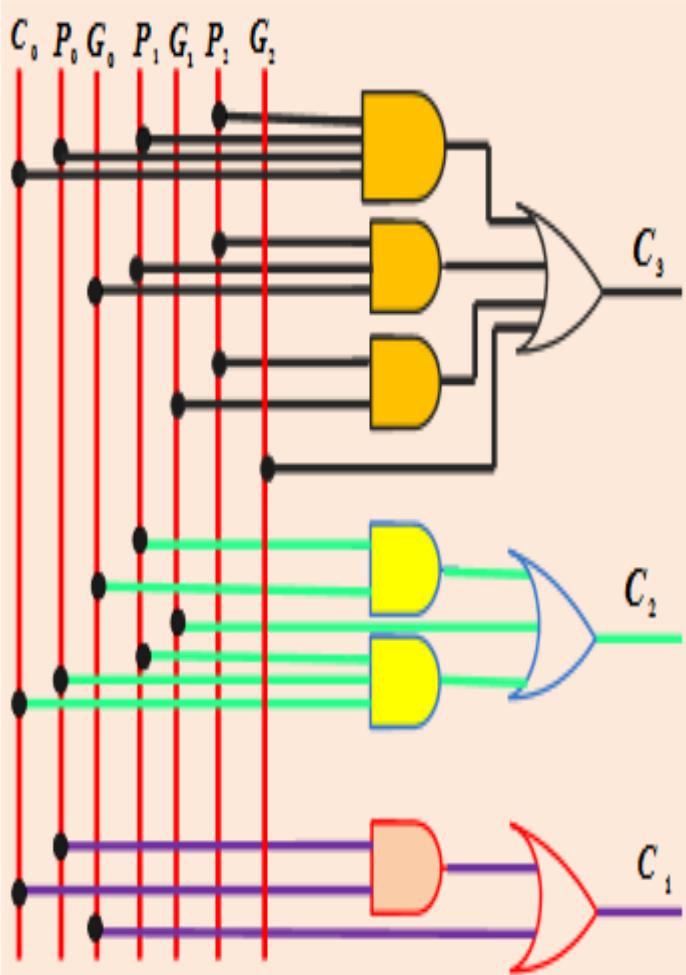


- The three Boolean functions  $C_1$ ,  $C_2$  and  $C_3$  are implemented in the **carry lookahead generator**.

*The two level-circuit for the output carry  $C_4$  is not shown, it can be easily derived by the equation.*

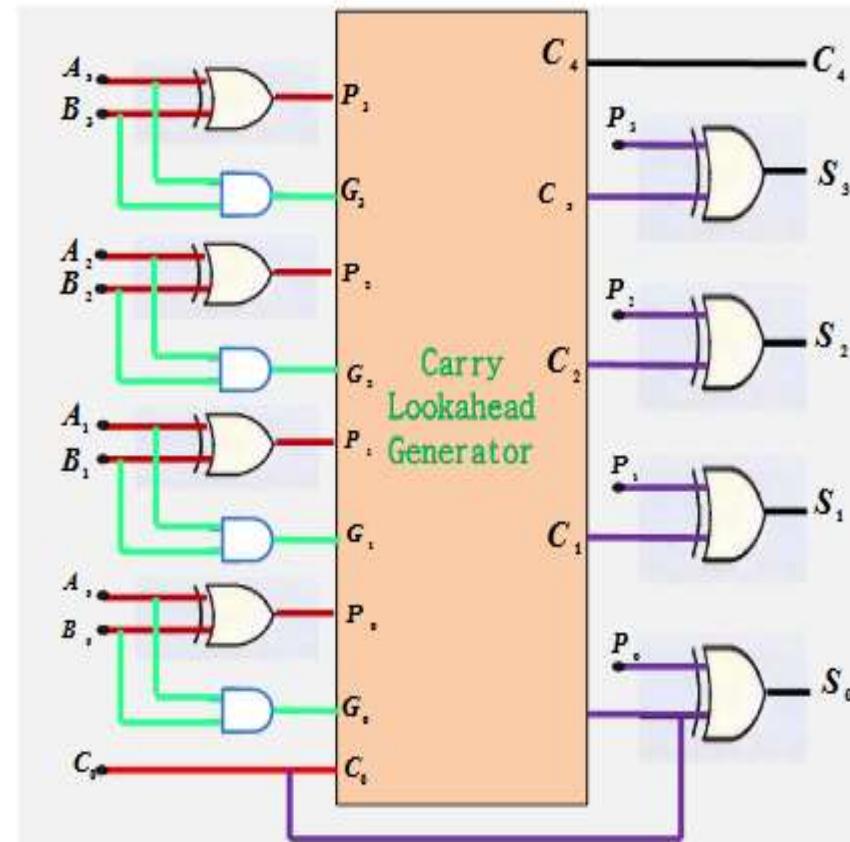
- $C_3$  does not have to wait for  $C_2$  and  $C_1$  to propagate, in fact  $C_3$  is propagated at the same time as  $C_1$  and  $C_2$ .

# Carry look ahead Adder



Logic Diagram for Carry Lookahead Generator

- The construction of a *four-bit adder with a carry lookahead scheme* is the following:



four-bit adder with a carry lookahead scheme

# Decimal Adder

A decimal adder requires a minimum of 9 inputs and 5 outputs

- 1 digit requires 4-bit
- Input: 2 digits + 1-bit carry
- Output: 1 digit + 1-bit carry

BCD adder

- Perform the addition of two decimal digits in BCD, together with an input carry from a previous stage.
- The output sum cannot be greater than 19 (9+9+1)

# BCD Adder

- 4-bits plus 4-bits
- Operands and Result: 0 to 9

$$\begin{array}{r}
 + x_3 \ x_2 \ x_1 \ x_0 \\
 + y_3 \ y_2 \ y_1 \ y_0 \\
 \hline
 \end{array}$$

$$Cy \ S_3 \ S_2 \ S_1 \ S_0$$

$X + Y$	$x_3 \ x_2 \ x_1 \ x_0$	$y_3 \ y_2 \ y_1 \ y_0$	$Sum$	$Cy$	$S_3 \ S_2 \ S_1 \ S_0$
$0 + 0$	0 0 0 0	0 0 0 0	= 0	0	0 0 0 0
$0 + 1$	0 0 0 0	0 0 0 1	= 1	0	0 0 0 1
$0 + 2$	0 0 0 0	0 0 1 0	= 2	0	0 0 1 0
$0 + 9$	0 0 0 0	1 0 0 1	= 9	0	1 0 0 1
$1 + 0$	0 0 0 1	0 0 0 0	= 1	0	0 0 0 1
$1 + 1$	0 0 0 1	0 0 0 1	= 2	0	0 0 1 0
$1 + 8$	0 0 0 1	1 0 0 0	= 9	0	1 0 0 1
$1 + 9$	0 0 0 1	1 0 0 1	= A	0	1 0 1 0
$2 + 0$	0 0 1 0	0 0 0 0	= 2	0	0 0 1 0
$9 + 9$	1 0 0 1	1 0 0 1	= 12	1	0 0 1 0

Invalid Code

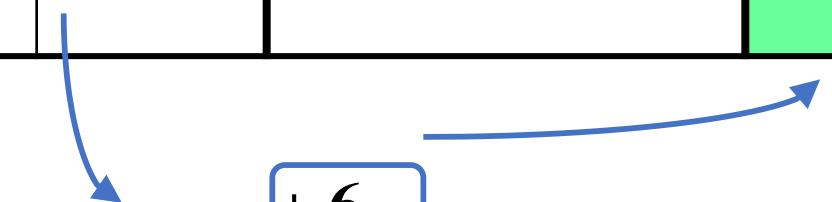
Wrong BCD Value

0001 1000

# BCD Adder

$X + Y$	$x_3 x_2 x_1 x_0$	$y_3 y_2 y_1 y_0$	Sum	$Cy$	$S_3 S_2 S_1 S_0$	Required BCD Output	Value
$9 + 0$	1 0 0 1	0 0 0 0	= 9	0	1 0 0 1	0 0 0 0 1 0 0 1	= 9
$9 + 1$	1 0 0 1	0 0 0 1	= 10	0	1 0 1 0	0 0 0 1 0 0 0 0	= 16
$9 + 2$	1 0 0 1	0 0 1 0	= 11	0	1 0 1 1	0 0 0 1 0 0 0 1	= 17
$9 + 3$	1 0 0 1	0 0 1 1	= 12	0	1 1 0 0	0 0 0 1 0 0 1 0	= 18
$9 + 4$	1 0 0 1	0 1 0 0	= 13	0	1 1 0 1	0 0 0 1 0 0 1 1	= 19
$9 + 5$	1 0 0 1	0 1 0 1	= 14	0	1 1 1 0	0 0 0 1 0 1 0 0	= 20
$9 + 6$	1 0 0 1	0 1 1 0	= 15	0	1 1 1 1	0 0 0 1 0 1 0 1	= 21
$9 + 7$	1 0 0 1	0 1 1 1	= 16	1	0 0 0 0	0 0 0 1 0 1 1 0	= 22
$9 + 8$	1 0 0 1	1 0 0 0	= 17	1	0 0 0 1	0 0 0 1 0 1 1 1	= 23
$9 + 9$	1 0 0 1	1 0 0 1	= 18	1	0 0 1 0	0 0 0 1 1 0 0 0	= 24

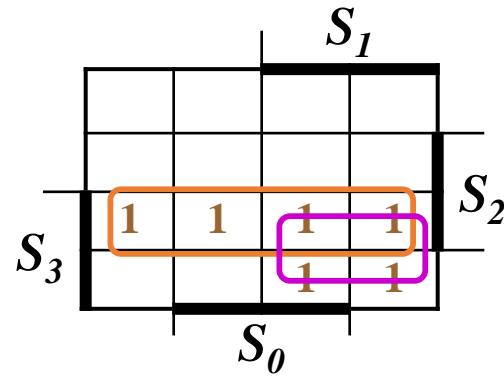
+ 6



# BCD Adder

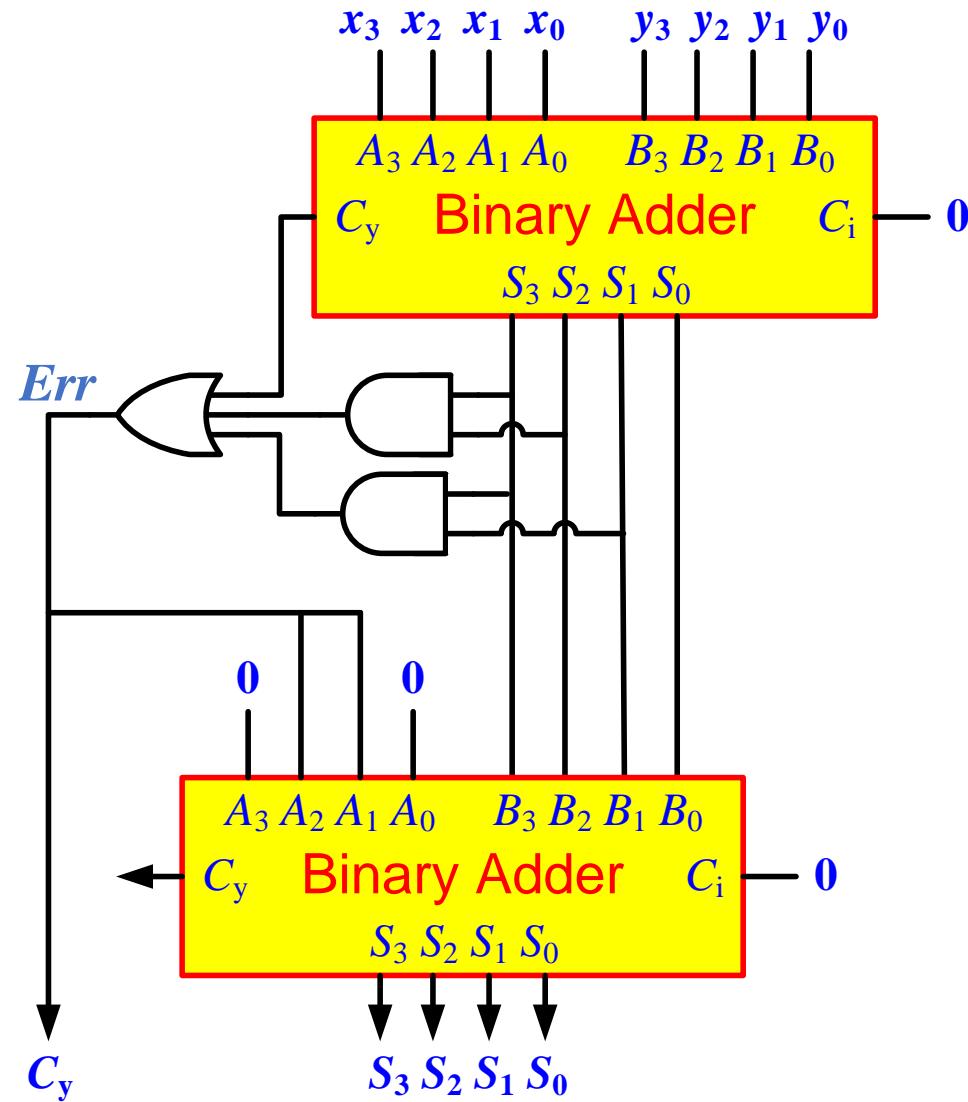
- Correct Binary Adder's Output (+6)
  - If the result is between 'A' and 'F'
  - If  $C_y = 1$

$S_3\ S_2\ S_1\ S_0$	$yI$
0 0 0 0	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



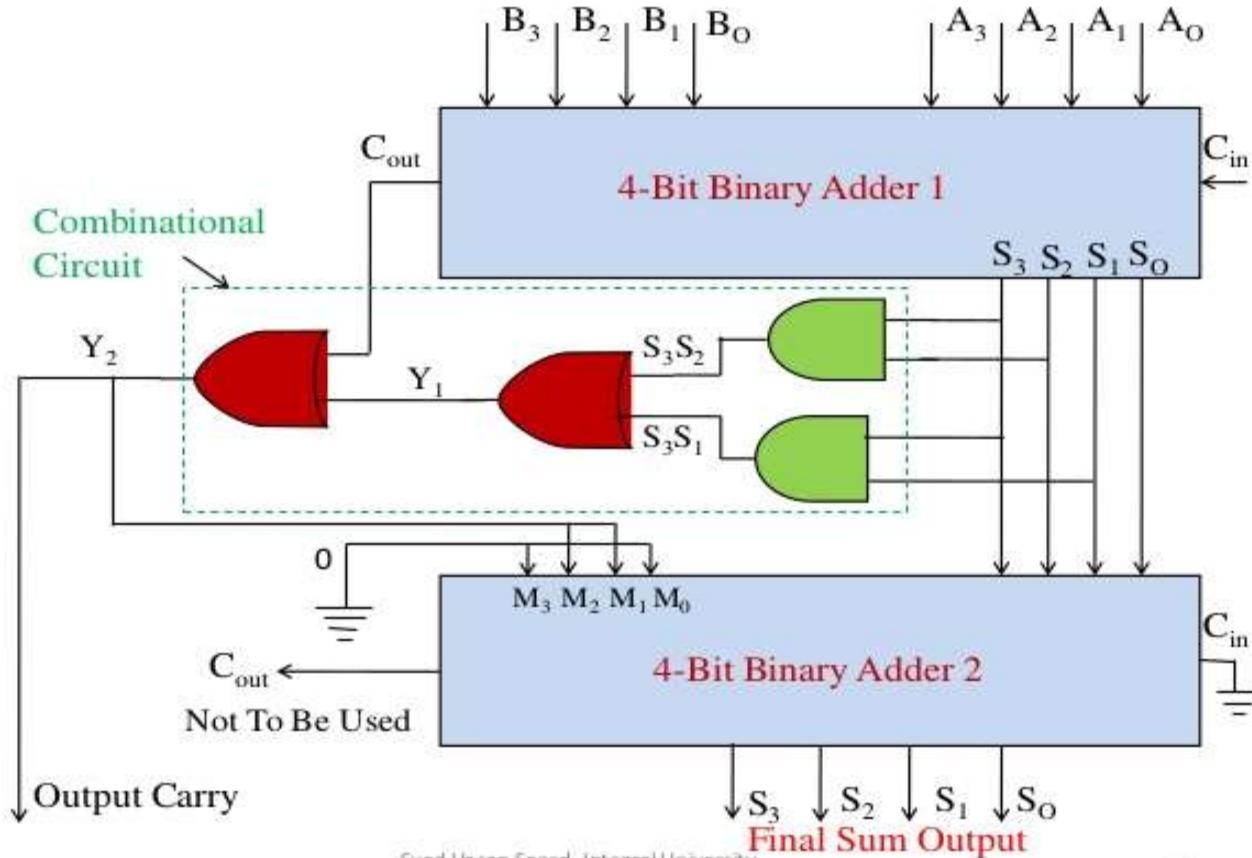
$$Err = S_3S_2 + S_3S_1$$

# BCD Adder



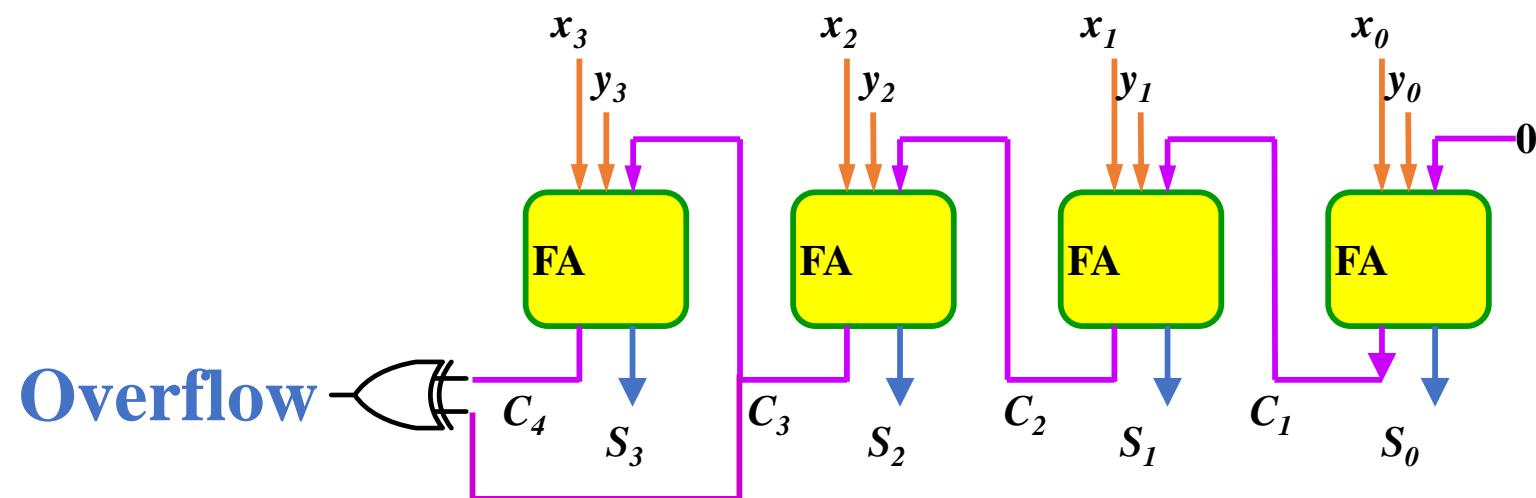
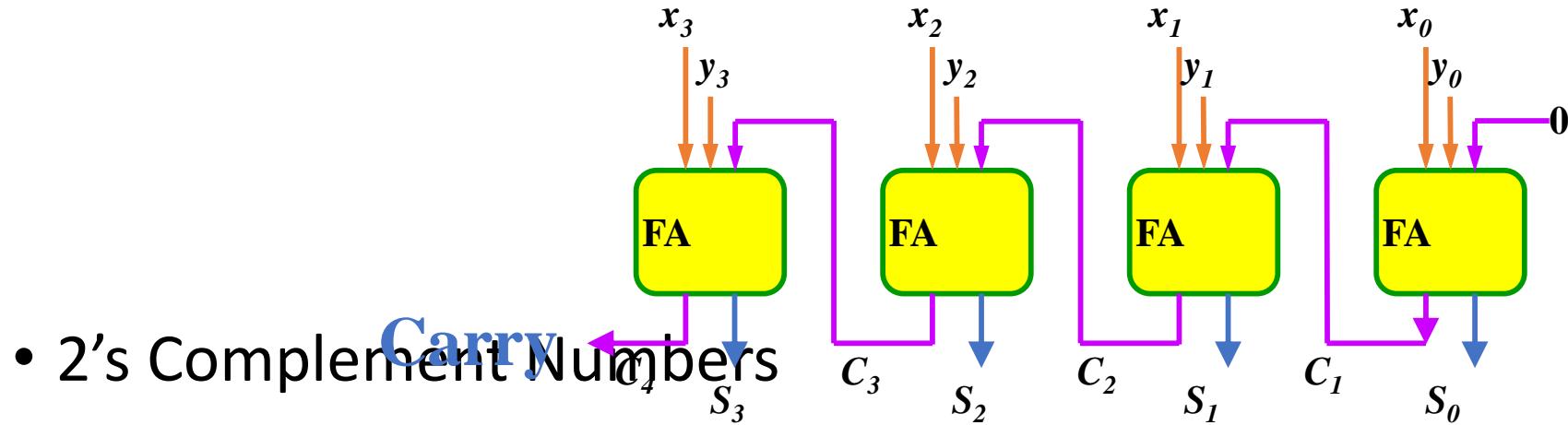
# BCD Adder

Fig. 1: BLOCK DIAGRAM OF BCD ADDER



# Overflow

- Unsigned Binary Numbers



# Magnitude Comparator

## 1- Bit magnitude comparator

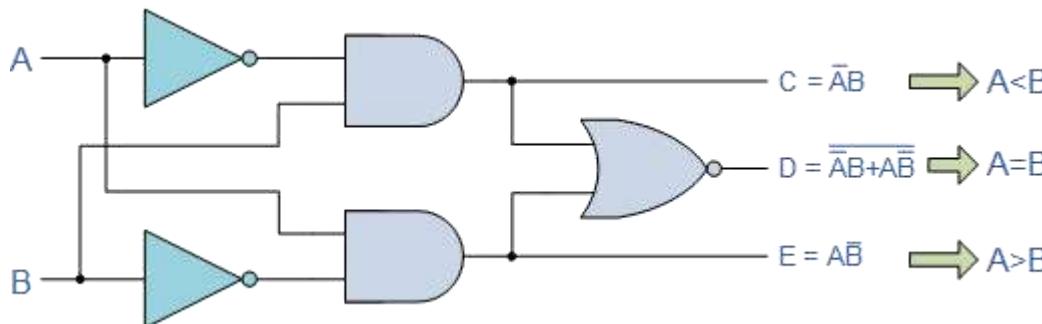
- ❖ A comparator used to compare two bits is called a single bit comparator.
- ❖ It consists of two inputs each for two single bit numbers and three outputs to generate less than, equal to and greater than between two binary numbers.



Truth table

A	B	A < B	A = B	A > B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Logic design



Equation from truth table

$$A < B = A' B$$

$$A > B = A B'$$

$$A = B = A' B' + A B$$

# Magnitude Comparator

## 2- Bit magnitude comparator

- ❖ A comparator used to compare two binary numbers each of two bits is called a 2-bit Magnitude comparator. It consists of four inputs and three outputs to generate less than, equal to and greater than between two binary numbers.

Truth table

INPUT				OUTPUT		
A1	A0	B1	B0	A<B	A=B	A>B
0	0	0	0	0	1	0
0	0	0	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

## K map for 2- Bit magnitude comparator

		A>B			
		00	01	11	10
A1A0	00	0	0	0	0
	01	1	0	0	0
11	1	1	0	1	
	00	1	1	0	0

		A = B			
		00	01	11	10
A1A0	00	1	0	0	0
	01	0	1	0	0
11	0	0	1	0	
	00	0	0	0	1

		A < B			
		00	01	11	10
A1A0	00	0	1	1	1
	01	0	0	1	1
11	0	0	0	0	
	00	0	0	1	0

## Design equations

$$A>B: A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$$

$$A=B: A_1'A_0'B_1'B_0' + A_1'A_0B_1'B_0 + A_1A_0B_1B_0 + A_1A_0'B_1B_0'$$

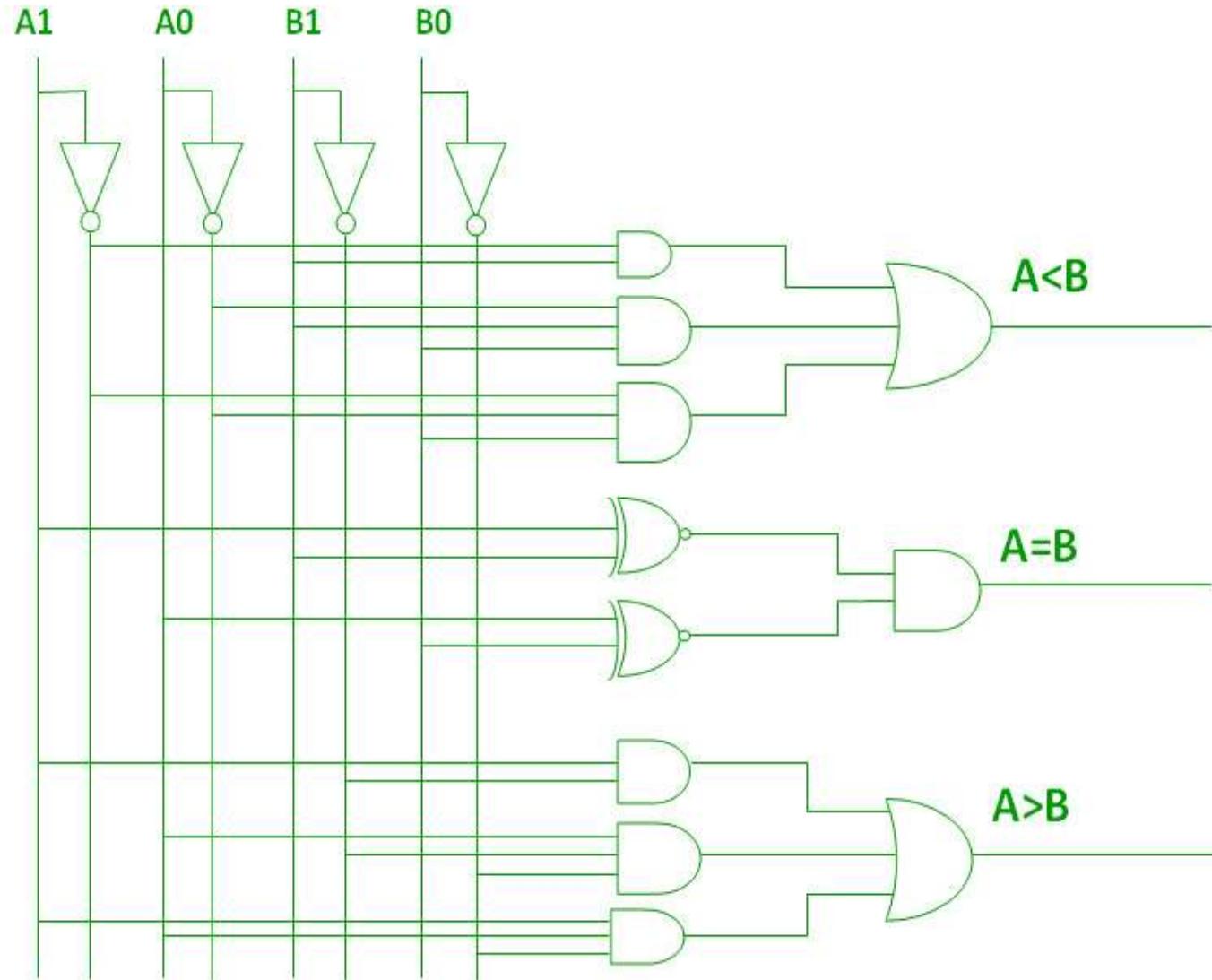
$$: A_1'B_1' (A_0'B_0' + A_0B_0) + A_1B_1 (A_0B_0 + A_0'B_0')$$

$$: (A_0B_0 + A_0'B_0') (A_1B_1 + A_1'B_1')$$

$$: (A_0 \text{ Ex-Nor } B_0) (A_1 \text{ Ex-Nor } B_1)$$

$$A<B: A_1'B_1 + A_0'B_1B_0 + A_1'A_0'B_0$$

## Logic design of 2 –bit magnitude comparator



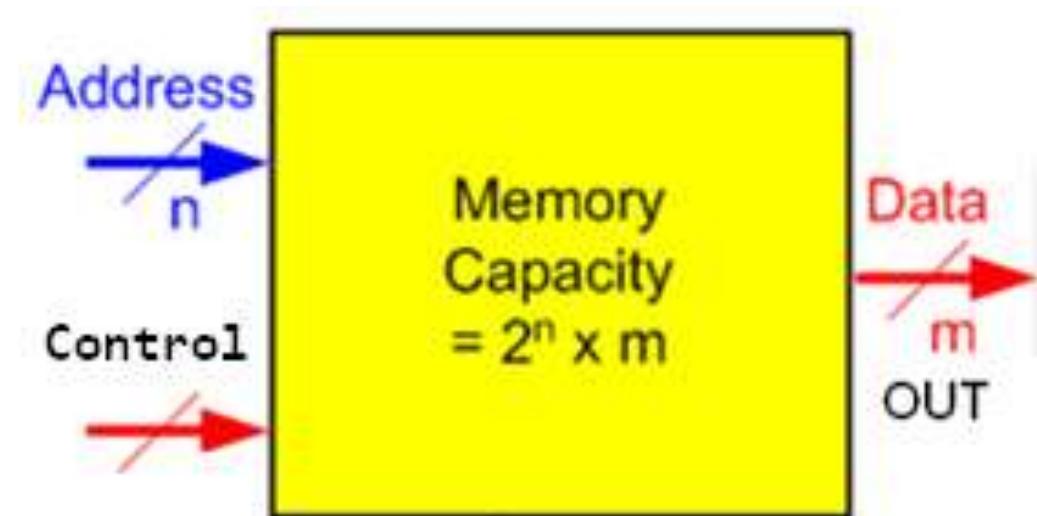
# **ANALOG AND DIGITAL ELECTRONICS**

**UNIT 3 – COMBINATIONAL LOGIC CIRCUITS**

**READ ONLY MEMORY**

**-S.Vaishali**

- Data stored in a ROM is non-volatile, i.e. this data is permanently stored until erased or changed through re-programming (if applicable)
- The ROM has n input lines for the address and m output data lines
- Total memory capacity of a ROM is  $2^n \times m$  bits
- ROMs do not have input lines as a write operation does not exist in them
- Programmable ROMs receive data to be programmed on the output lines
- Generally, system-level programs that need to be accessed frequently and at power up access are stored in the computer's ROM, e.g. the BIOS firmware

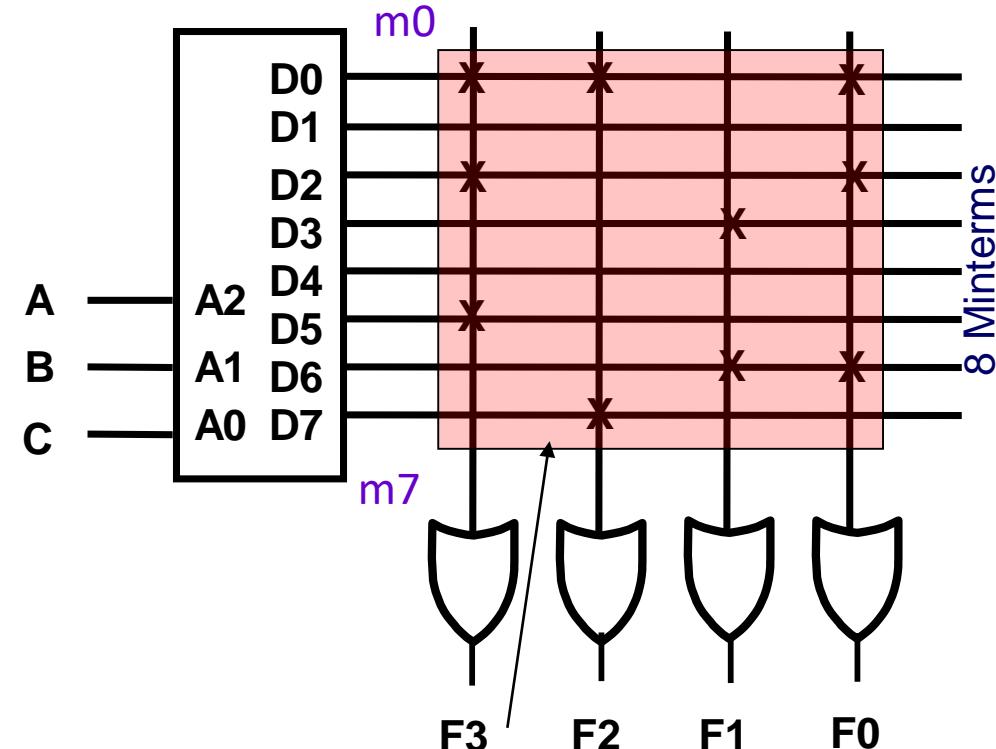


# Read Only Memory (ROM)

## Programmable sum of (fixed) minterms

- Example: 8 X 4 PROM ( $n = 3$  input lines,  $m = 4$  output lines)
- The fixed "AND" array is a "decoder" with 3 inputs and 8 outputs implementing minterms
- The programmable "OR" array uses a single line to represent all inputs to an OR gate. An "X" in the array corresponds to attaching the minterm to the OR
- Read Example: For input  $(A_2, A_1, A_0) = 010$ , output is  $(F_3, F_2, F_1, F_0) = 1001$ .
- What are functions  $F_3, F_2, F_1$  and  $F_0$  in terms of  $(A_2, A_1, A_0)$ ?

8 X 3-input fixed ANDs give all 8 minterms



$2^n \times m$  Programmable Connections

- Read Only Memories (ROM) have:
  - $n$  input (address) lines  $\rightarrow 2^n$  locations  $\rightarrow 2^n$  decoded minterms
  - $m$  output lines (word width)
- Fixed array of  $2^n$  AND gates implementing all the  $N$ -literal minterms.
- Programmable OR Array with  $m$  outputs lines to form up to  $m$  expressions, each being a sum of selected minterm.
- The program for a PROM is simply the multiple-output truth table to be implemented
  - If a 1 entry, a connection is made to the corresponding minterm for the corresponding output
  - If a 0, no connection is made
- Can be viewed as a *memory* with the inputs as *addresses* of *data* (output values), hence ROM or PROM names!  
Device on previous slide is an  $8 \times 4$  memory (8 locations, each 4 bits)
- Truth table is a listing of the memory contents

### Advantages:

Can implement any function (all the minterms are available)

Program is derived directly from the truth table  
(uses the canonical form)

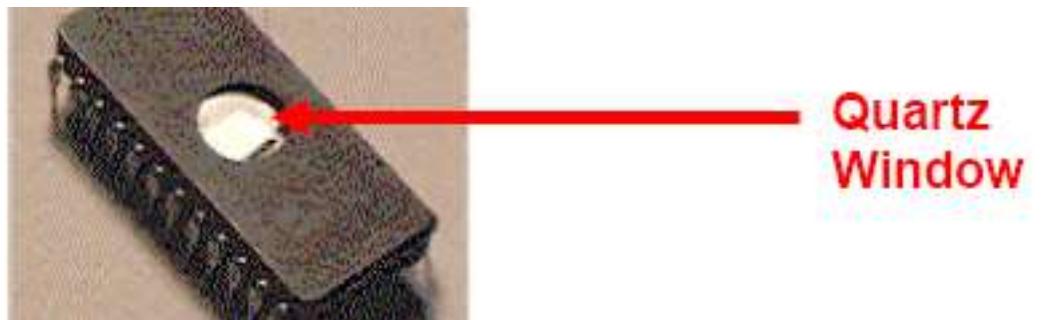
### Disadvantages:

Becomes complex for a large number of inputs n  
(# of ANDs =  $2^n$ , each n-input wide)

Does not support multi-level circuits (no outputs brought back as inputs)

- Simply ROM: Programmed only once and by the manufacturer (in factory), based on the client's truth table
- PROM: A ROM programmable only once by the user (in the field). The user blows fuses to remove unwanted connections. This process is irreversible and hence device is programmed only once
- EPROM: Erasable, Programmable ROMs. Can have their data erased using Ultraviolet light and reprogrammed. The user can then reprogram the ROM many times using special programmers Off-situ.

Off-situ: Remove from computer to erase/program



- EEPROMs: Electrically Erasable Programmable ROMs. Have memory cells that can be erased and reprogrammed by exposure to electrical signals.  
Erasure/Programming is now much easier and in-situ. The processor can now “write” into the EEPROM.
- Flash memory devices:
  - Memory cells are erased in blocks not one-by-one as in EEPROMs → Shorter life but faster operation

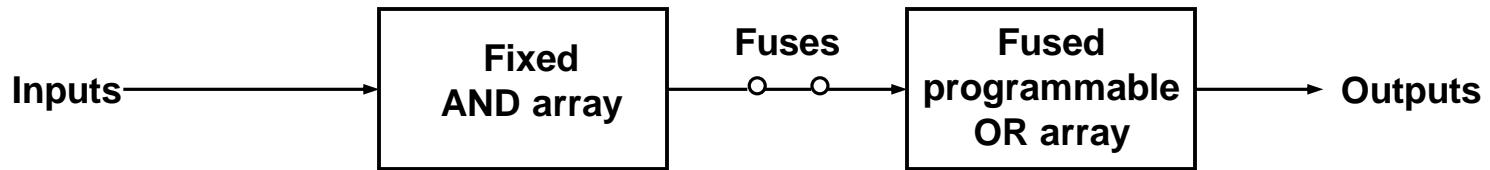
## **UNIT – IV**

**Programmable Logic devices,  
Memory and Logic Families**

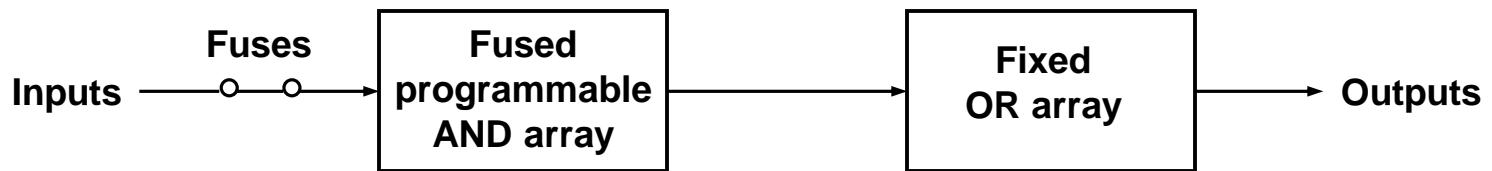
# Programmable Logic Devices (PLD)

- Programmable Logic Devices (PLDs) are IC chips with internal logic gates connected by electronic fuses.
- These fuses can be ‘blown’ (by programming) to obtain different circuit configurations.
- Semi-customized chips that give high packing density at reasonable cost.
- Three classes of PLDs are :
  - ❖ Programmable Logic Array (PLA)
  - ❖ Programmable Read Only Memory (PROM)
  - ❖ Programmable Array Logic (PAL)

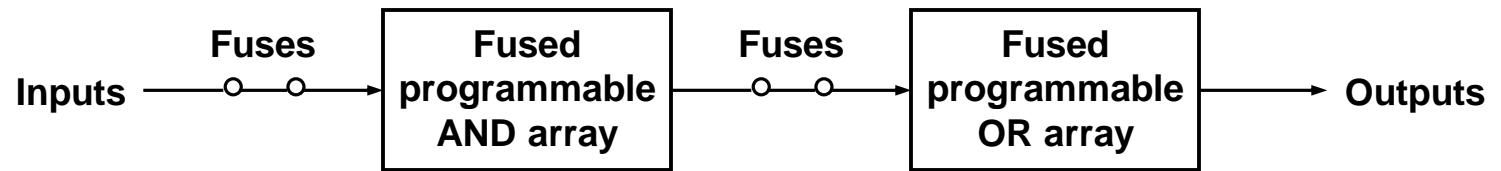
# Programmable Logic Devices



**Programmable Read Only Memory (PROM)**



**Programmable Array Logic (PAL)**



**Programmable Logic Array (PLA)**

# Programmable Logic Array (PLA)

- Combination of a programmable AND array followed by a programmable OR array.
- Example: Design a PLA to realise the following three logic functions and show the internal connections.

$$f_1(A,B,C,D,E) = A'.B'.D' + B'.C.D' + A'.B.C.D.E'$$

$$f_2(A,B,C,D,E) = A'.B.E + B'.C.D'.E$$

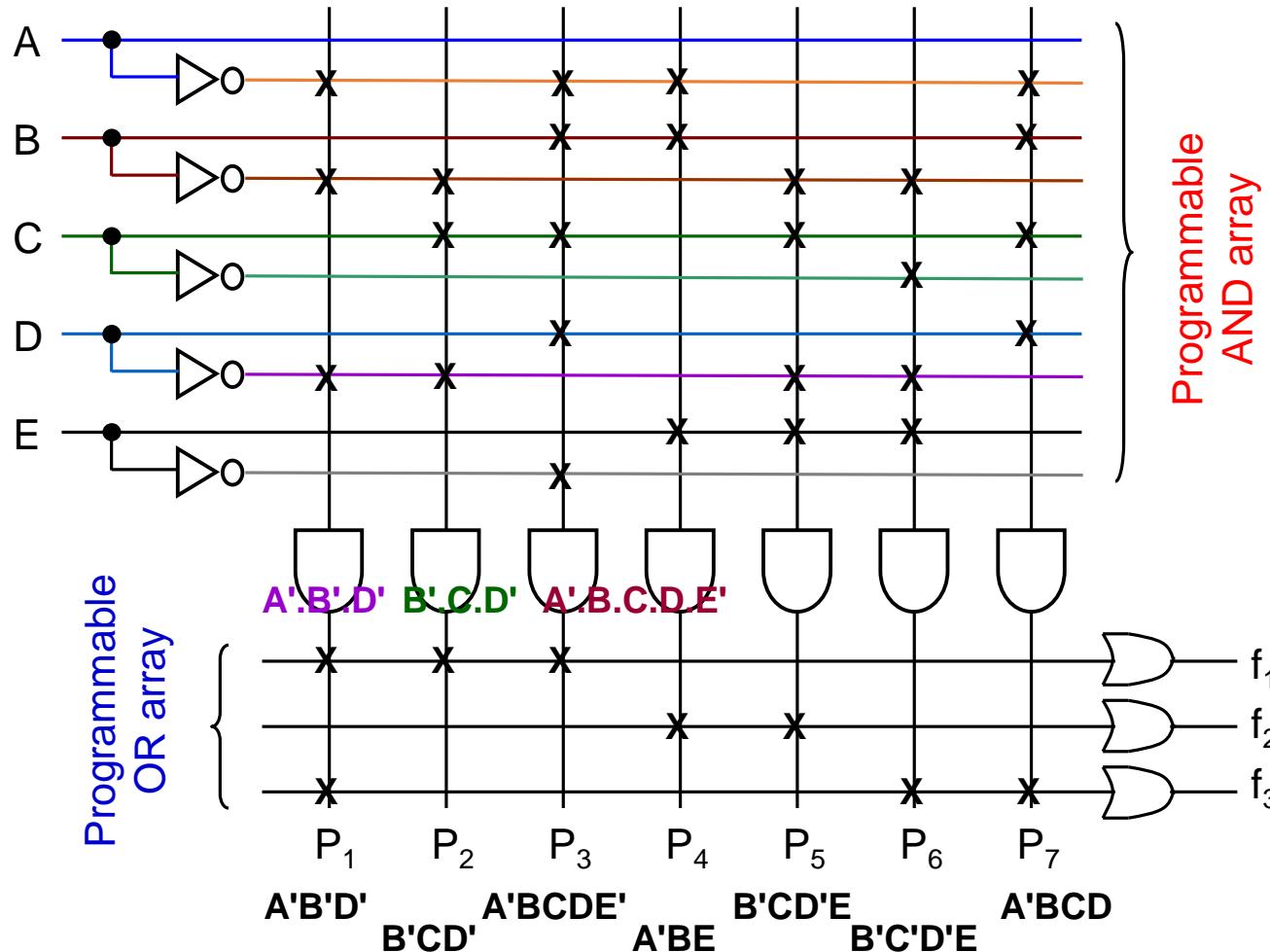
$$f_3(A,B,C,D,E) = A'.B'.D' + B'.C'.D'.E + A'.B.C.D$$

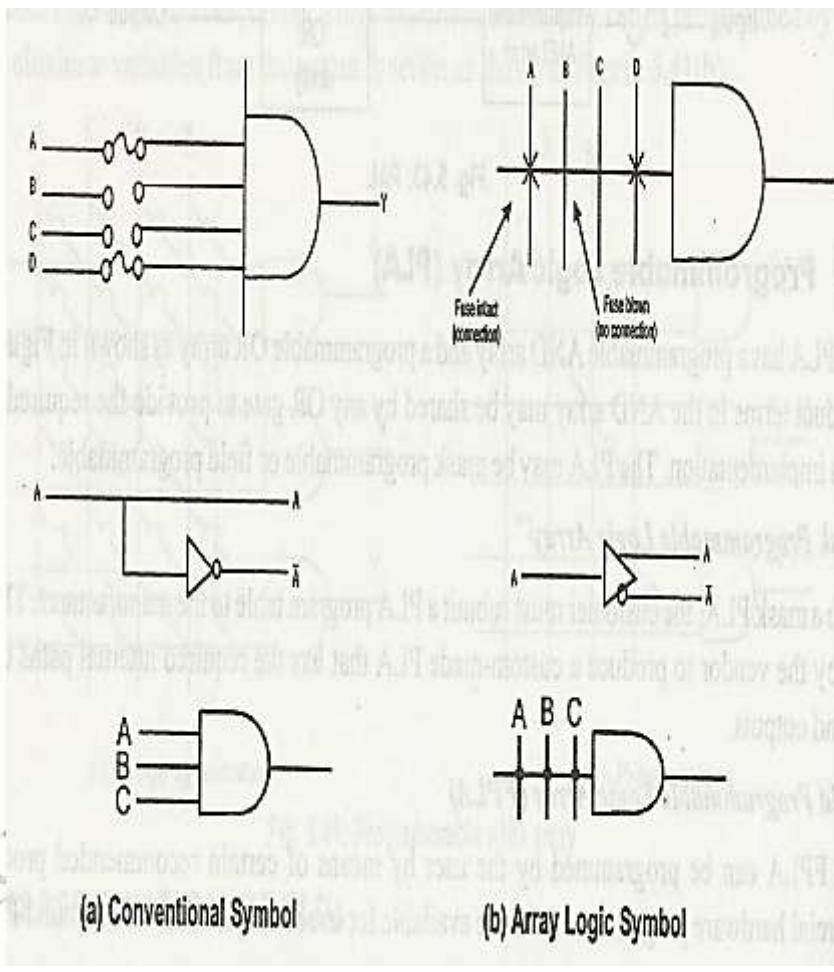
# Realising Logic Functions with PLAs

$$f_1(A,B,C,D,E) = A'.B'.D' + B'.C.D' + A'.B.C.D.E'$$

$$f_2(A,B,C,D,E) = A'.B.E + B'.C.D'.E$$

$$f_3(A,B,C,D,E) = A'.B'.D' + B'.C'.D'.E + A'.B.C.D$$





Implement the following functions using PLA.

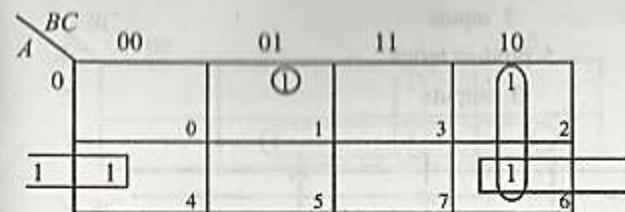
$$F_1 = \sum m(1, 2, 4, 6); F_2 = \sum m(0, 1, 6, 7)$$

$$F_3 = \sum m(2, 6)$$

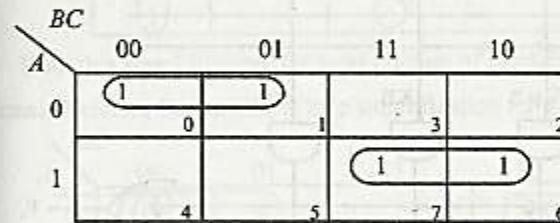
Solution: Step 1: Truth Table for given functions:

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

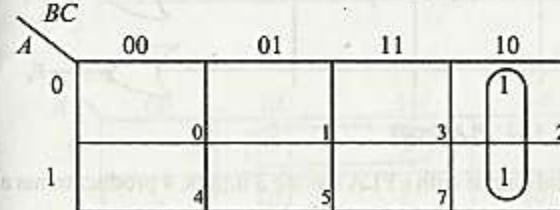
Step 2: K map Simplification:



$$F_1 = \overline{ABC} + A\overline{C} + \overline{B}\overline{C}$$



$$F_2 = \overline{A}\overline{B} + AB$$

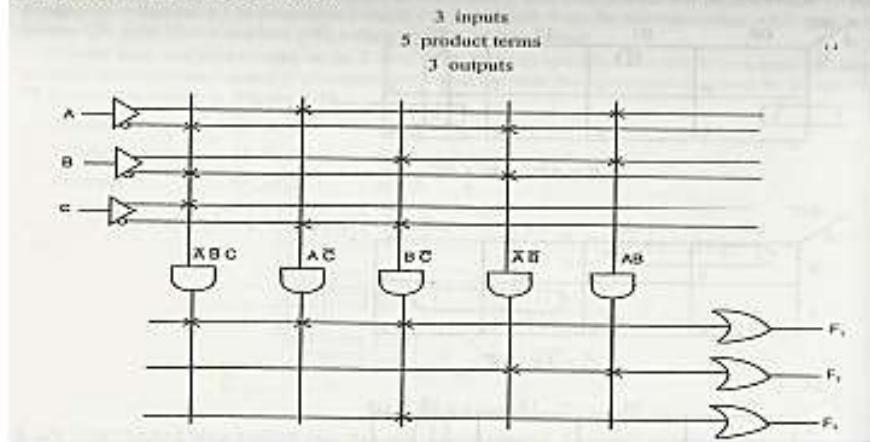


$$F_3 = B\overline{C}$$

Step 3: PLA Program Table:

Product Term	Inputs			Outputs		
	A	B	C	$F_1$	$F_2$	$F_3$
$\bar{A}\bar{B}C$	1	0	0	1	-	-
$A\bar{C}$	2	1	-	0	1	-
$B\bar{C}$	3	-	1	0	1	1
$\bar{A}B$	4	0	0	-	-	1
$AB$	5	1	1	-	-	T/C

Step 4: Draw the PLA circuit with



Implement the combinational circuit with PLA having 3 inputs, 4 product terms and 2 output for the functions:

$$F_1 = \sum (3, 5, 6, 7), F_2 = \sum (0, 2, 4, 7)$$

Solution :

Step 1: Truth table for Boolean Functions:

A	B	C	$F_1$	$F_2$
0	0	0	0	1
0	0	1	0	0
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

Step 2: K map Simplification:

		BC	00	01	11	10	
		A	0			1	
		1			1		1

$$F_1 = AC + AB + BC$$

		BC	00	01	11	10	
		A	0	1			
		1	1			1	

$$\bar{F}_1 = \bar{B}\bar{C} + \bar{A}\bar{C} + ABC$$

With this simplification, the total number of product terms is 6. But we require only 4 product terms. Therefore find out the K map simplification for  $\bar{F}_1$  and  $\bar{F}_2$ .

		BC	00	01	11	10	
		A	0	(0)	0		0
		1	0				

$$\bar{F}_1 = \bar{B}\bar{C} + \bar{A}\bar{C} + \bar{A}\bar{B}$$

		BC	00	01	11	10	
		A	0		(0)	0	
		1		0			0

$$\bar{F}_2 = \bar{A}C + \bar{B}C + ABC\bar{C}$$

Now select the functions  $F_1$  and  $\bar{F}_2$ , since the common product terms are  $(\bar{B}\bar{C}, \bar{A}\bar{C}, \bar{A}\bar{B}, ABC)$

Step 3: PLA Program Table:

Product Term	Inputs			Outputs	
	A	B	C	$F_1$	$F_2$
$\overline{BC}$	1	-	0	0	1
$\overline{AC}$	2	0	-	0	1
$\overline{AB}$	3	0	0	-	1
$ABC$	4	1	1	1	-
				$C$	$T$
					$T/C$

$$F_1 = (\overline{B}\overline{C} + \overline{A}\overline{C} + \overline{A}\overline{B}) ; F_2 = \overline{B}\overline{C} + \overline{A}\overline{C} + ABC$$

Step 4: Draw the PLA circuit with

3 inputs, 4 product terms and 2 outputs.

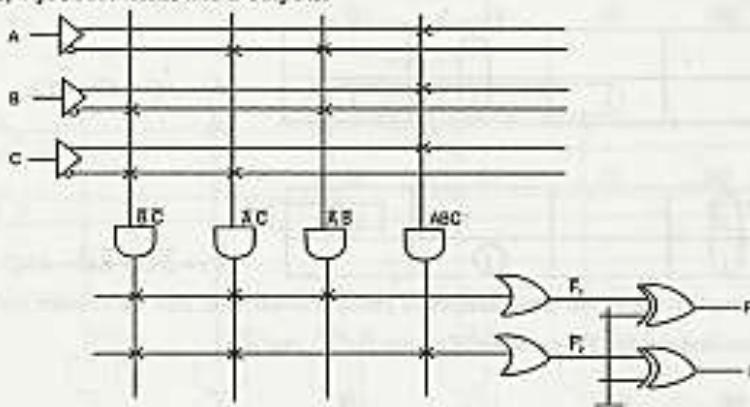


Fig. 5.51: PLA circuit

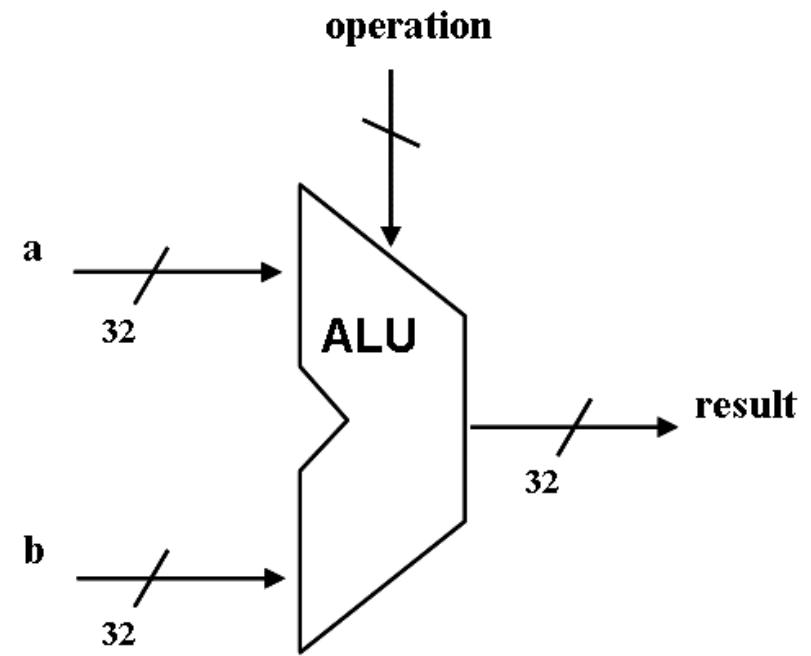
It should be noted that  $\overline{F}_1$  really occurs at one of the outputs of the OR-array. By programming the corresponding EX-OR gate fuse,  $\overline{F}_1 - F_1$  appears at the output of the PLA.

# **ANALOG AND DIGITAL ELECTRONICS**

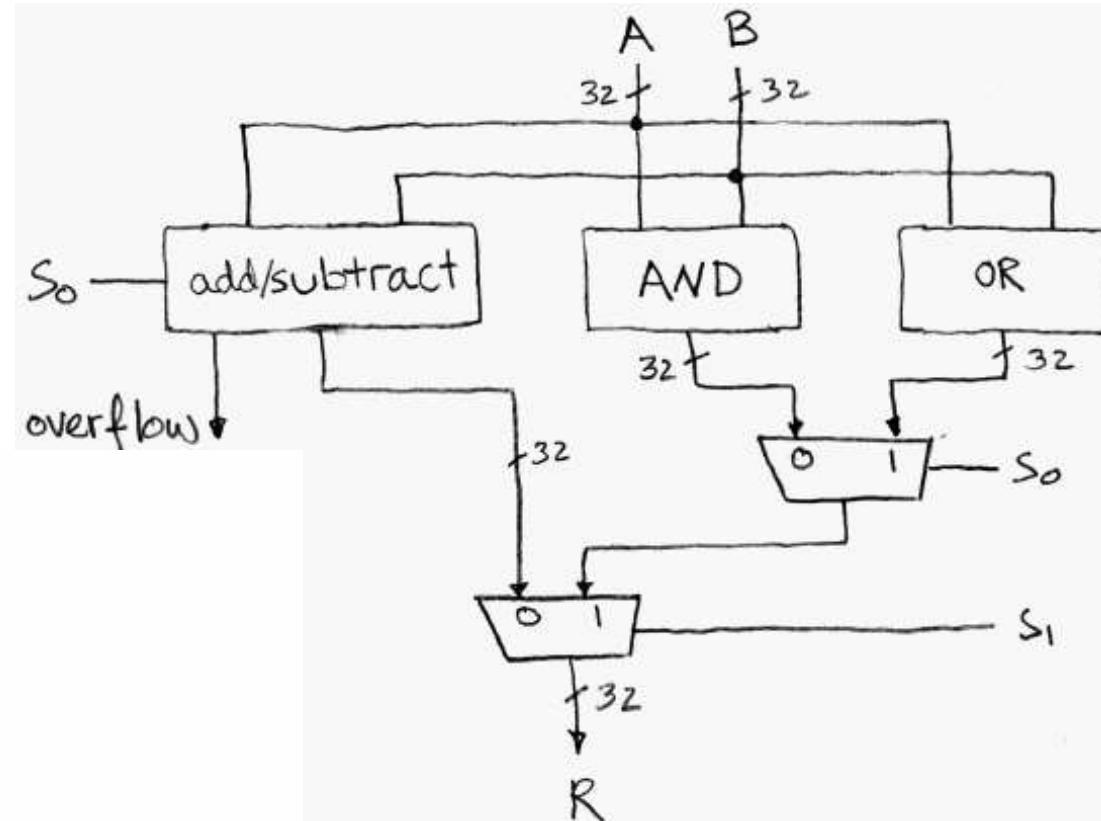
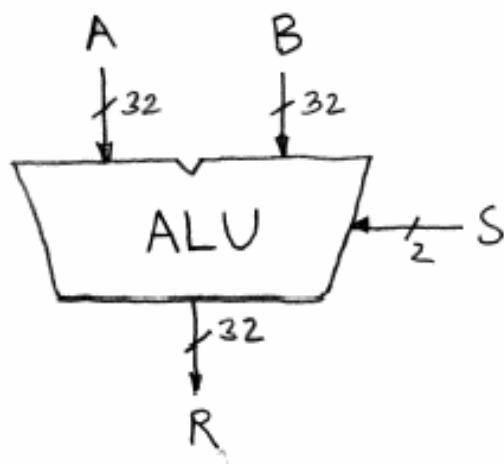
**UNIT 3 – COMBINATIONAL LOGIC CIRCUITS  
ARITHMETIC LOGIC UNIT**

**-S.Vaishali**

- An **arithmetic logic unit (ALU)**
  - Performs arithmetic and logic operations
  - A fundamental building block of the Central Processing Unit (CPU) of a computer
  - Even the simplest microprocessors contain one for purposes such as maintaining timers
  - A combinational logic circuit



# Simple ALU

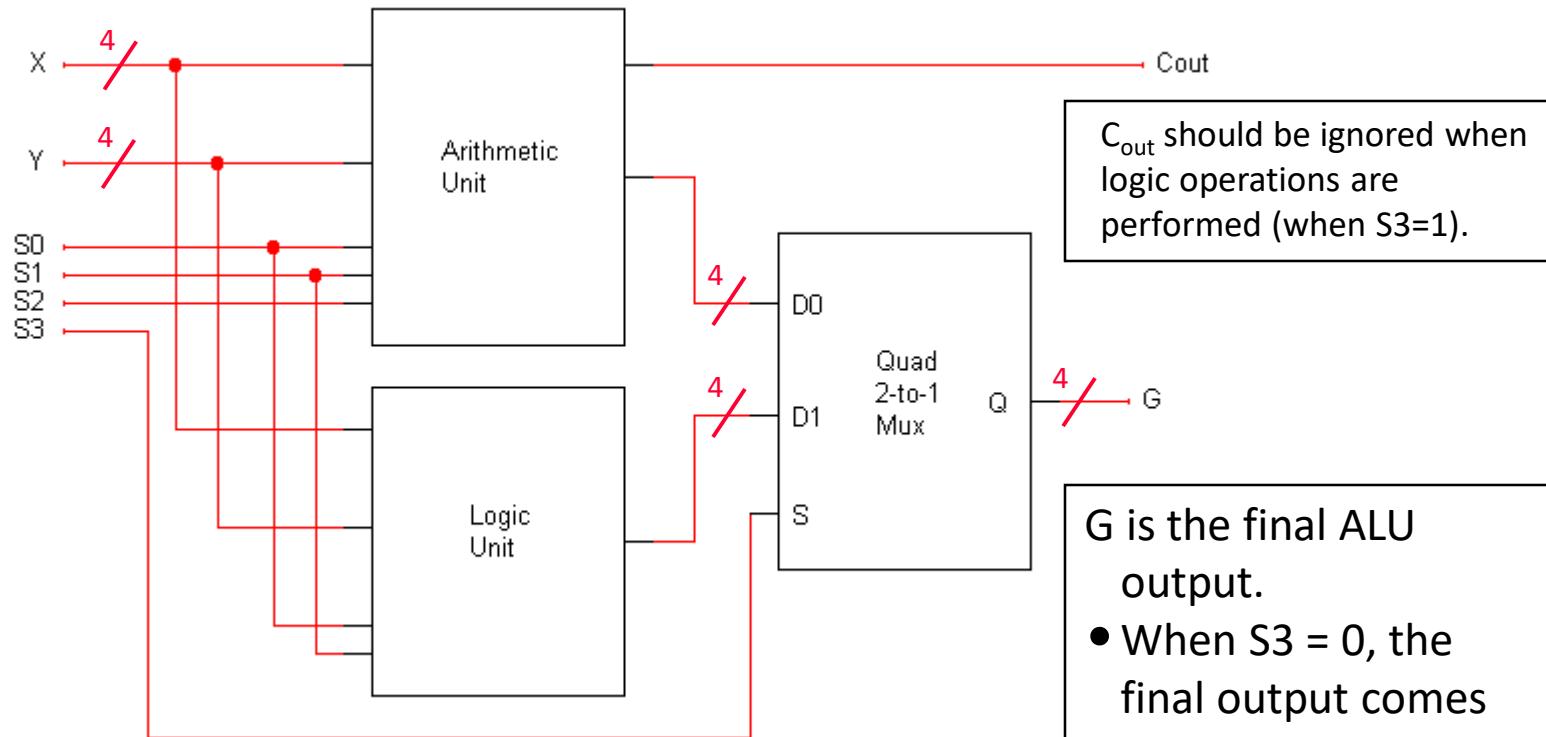


when  $S=00$ ,  $R=A+B$   
when  $S=01$ ,  $R=A-B$   
when  $S=10$ ,  $R=A \text{ AND } B$   
when  $S=11$ ,  $R=A \text{ OR } B$

The S input is controlled by the processor based on the op code

# A complete ALU circuit

The / and 4 on a line indicate that it's actually *four* lines.



The arithmetic and logic units share the select inputs S1 and S0, but only the arithmetic unit uses S2.

- When S3 = 0, the final output comes from the arithmetic unit.
- When S3 = 1, the output comes from the logic unit.

Unit - 3

Hardware Description Language

# Hardware Description Language - Introduction

- 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 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.
- The most common use of a HDL is to provide an alternative to schematics.

# HDL – Introduction

- When a language is used for the above purpose (i.e. to provide an alternative to schematics), it is referred to as a *structural description* in which the language describes an interconnection of components.
- Such a structural description can be used as input to logic simulation just as a schematic is used.
- Models for each of the primitive components are required.
- If an HDL is used, then these models can also be written in the HDL providing a more uniform, portable representation for simulation input.

# HDL – Introduction

- HDL can be used to represent logic diagrams, Boolean expressions, and other more complex digital circuits.
- Thus, in top down design, a very high-level description of a entire system can be precisely specified using an HDL.
- This high-level description can then be refined and partitioned into lower-level descriptions as a part of the design process.

# HDL – Introduction

- As a documentation language, HDL is used to represent and document digital systems in a form that can be read by both humans and computers and is suitable as an exchange language between designers.
- The language content can be stored and retrieved easily and processed by computer software in an efficient manner.
- There are two applications of HDL processing: ***Simulation*** and ***Synthesis***

# Logic Simulation

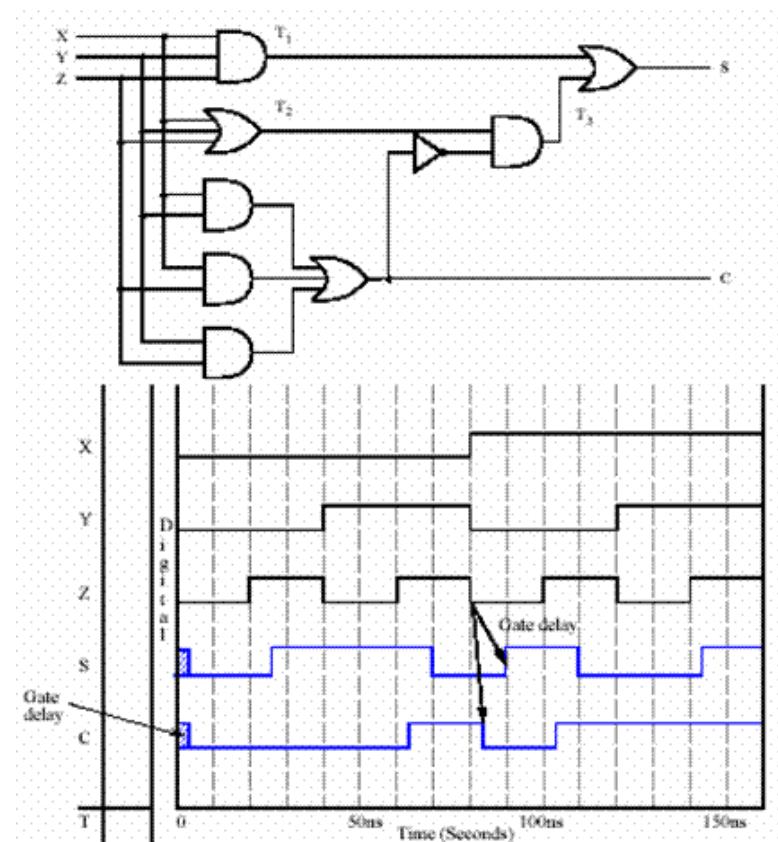
- A simulator interprets the HDL description and produces a readable output, such as a timing diagram, that predicts how the hardware will behave before it is actually fabricated.
- Simulation allows the detection of functional errors in a design without having to physically create the circuit.

# Logic Simulation

- The stimulus that tests the functionality of the design is called a test bench.
- To simulate a digital system
  - Design is first described in HDL
  - Verified by simulating the design and checking it with a test bench which is also written in HDL.

# Logic Simulation

- Logic simulation is a fast, accurate method of analyzing a circuit to see its waveforms



# Types of HDL

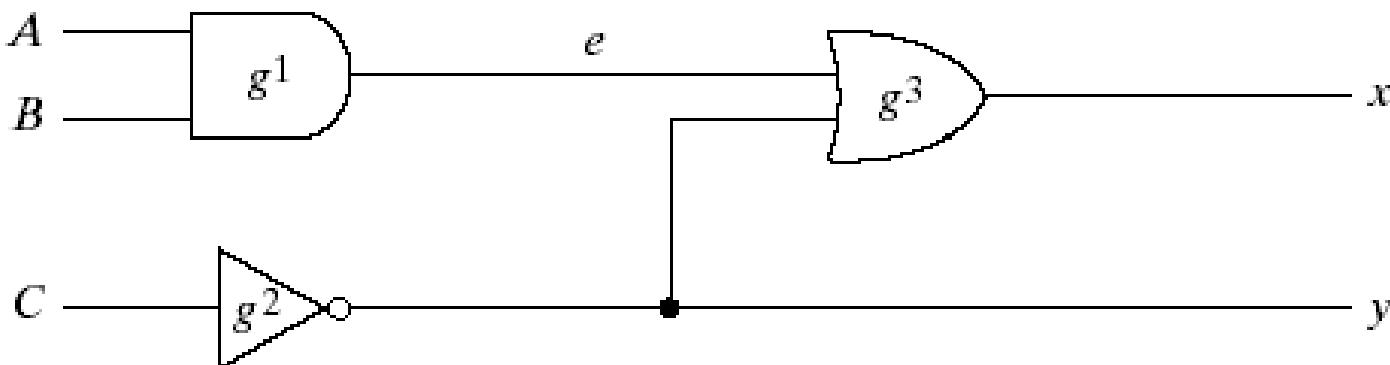
- There are two standard HDL's that are supported by IEEE.
  - **VHDL** (*Very-High-Speed Integrated Circuits Hardware Description Language*) - Sometimes referred to as VHSIC HDL, this was developed from an initiative by US. Dept. of Defense.
  - **Verilog HDL** – developed by Cadence Data systems and later transferred to a consortium called *Open Verilog International* (OVI).

# 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.

# Verilog Module

- 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*.



# Verilog Module for simple circuit

## HDL Example

```
module smpl_circuit (A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and g1 (e,A,B);
    not g2 (y,C);
    or g3 (x,e,y);
endmodule
```

# Verilog – Gate Delays

- Sometimes it is necessary to specify the amount of delay from the input to the output of gates.
- In Verilog, the delay is specified in terms of time units and the symbol #.
- The association of a time unit with physical time is made using ***timescale*** compiler directive.
- Compiler directive starts with the “backquote (`)” symbol.  
`***timescale*** 1ns/100ps
  - The first number specifies the *unit of measurement* for time delays.
  - The second number specifies the *precision* for which the delays are rounded off, in this case to 0.1ns.

# Verilog Module with Gate Delays

```
//Description of circuit with delay
module circuit_with_delay (A,B,C,x,y);
    input A,B,C;
    output x,y;
    wire e;
    and #(30) g1(e,A,B);
    or # (20) g3(x,e,y);
    not #(10) g2(y,C);
endmodule
```

# Test Bench

- In order to simulate a circuit with HDL, it is necessary to apply inputs to the circuit for the simulator to generate an output response.
- An HDL description that provides the stimulus to a design is called a **test bench**.
- The ***initial*** statement specifies inputs between the keyword ***begin*** and ***end***.
- Initially ABC=000 (A,B and C are each set to 1'b0 (one binary digit with a value 0)).
- **\$finish** is a *system task*.

# Test Bench for Simple Circuit

```
module circuit_with_delay
(A,B,C,x,y);
input A,B,C;
output x,y;
wire e;
and #(30) g1(e,A,B);
or #(20) g3(x,e,y);
not #(10) g2(y,C);
endmodule
```

```
//Stimulus for simple circuit
module stimcrct;
reg A,B,C;
wire x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
begin
A = 1'b0; B = 1'b0; C = 1'b0;
#100
A = 1'b1; B = 1'b1; C = 1'b1;
#100 $finish;
end
endmodule
```

# Verilog Module for Boolean Expressions

## Bitwise operators

- Bitwise NOT : ~
- Bitwise AND: &
- Bitwise OR: |
- Bitwise XOR: ^
- Bitwise XNOR:  $\sim\wedge$  or  $\wedge\sim$

# Verilog Module for Boolean Expressions

## Boolean Expressions:

- These are specified in Verilog HDL with a continuous assignment statement consisting of the keyword *assign* followed by a Boolean Expression.
- The earlier circuit can be specified using the statement:

**assign** x = (A&B) | ~C)

E.g.  $x = A + BC + B'D$

$y = B'C + BC'D'$

# Verilog Module for Boolean Expressions

```
//Circuit specified with Boolean equations
module circuit_bln (x,y,A,B,C,D);
    input A,B,C,D;
    output x,y;
    assign x = A | (B & C) | (~B & C);
    assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

# User Defined Primitives

## User Defined Primitives (UDP):

- The logic gates used in HDL descriptions with keywords *and*, *or*, etc., are defined by the system and are referred to as *system primitives*.
- The user can create additional primitives by defining them in tabular form.
- These type of circuits are referred to as *user-defined primitives*.

# User Defined Primitives

UDP features ....

- UDP's do not use the keyword module. Instead they are declared with the keyword ***primitive***.
- There can be **only one output** and it must be listed first in the port list and declared with an ***output*** keyword.
- There can be any number of inputs. The order in which they are listed in the ***input*** declaration must conform to the order in which they are given values in the table that follows.
- The truth table is enclosed within the keywords ***table*** and ***endtable***.
- The values of the inputs are listed with a colon (:). The output is always the last entry in a row followed by a semicolon (;).
- It ends with the keyword ***endprimitive***.

# Verilog Module for User Defined Primitives

```
//User defined primitive(UDP)
primitive crctp (x,A,B,C);
    output x;
    input A,B,C;
//Truth table for x(A,B,C) = Minterms (0,2,4,6,7)
    table
//      A   B   C   :   x   (Note that this is only a comment)
        0   0   0   :   1;
        0   0   1   :   0;
        0   1   0   :   1;
        0   1   1   :   0;
        1   0   0   :   1;
        1   0   1   :   0;
        1   1   0   :   1;
        1   1   1   :   1;
    endtable
endprimitive
// Instantiate primitive
module declare_crctp;
    reg      x,y,z;
    wire    w;
    crctp (w,x,y,z);
endmodule
```

# Types of Modeling

- A module can be described in any one (or a combination) of the following modeling techniques.
  - **Gate-level modeling** using instantiation of primitive gates and user defined modules.
    - This describes the circuit by specifying the gates and how they are connected with each other.
  - **Dataflow modeling** using continuous assignment statements with the keyword **assign**.
    - This is mostly used for describing combinational circuits.
  - **Behavioral modeling** using procedural assignment statements with keyword **always**.
    - This is used to describe digital systems at a higher level of abstraction.

# Gate-Level Modeling

- Here a circuit is specified by its **logic gates** and their **interconnections**.
- It provides a textual description of a schematic diagram.
- Verilog recognizes 12 basic gates as predefined primitives.
  - 4 primitive gates of 3-state type.
  - Other 8 are: **and, nand, or, nor, xor, xnor, not, buf**
- When the gates are simulated, the system assigns a four-valued logic set to each gate – *0, 1, unknown (x) and high impedance (z)*

# Gate-level modeling

- When a primitive gate is incorporated into a module, we say it is *instantiated* in the module.
- In general, component instantiations are statements that reference lower-level components in the design, essentially creating unique copies (or *instances*) of those components in the higher-level module.
- Thus, a module that uses a gate in its description is said to *instantiate* the gate.

# Gate-level Modeling

- Modeling with vector data (multiple bit widths):
  - A vector is specified within square brackets and two numbers separated with a colon.

e.g. **output** [0 : 3] D; - This declares an output vector D with 4 bits, 0 through 3.

**wire** [7 : 0] SUM; – This declares a wire vector SUM with 8 bits numbered 7 through 0.

The first number listed is the most significant bit of the vector.

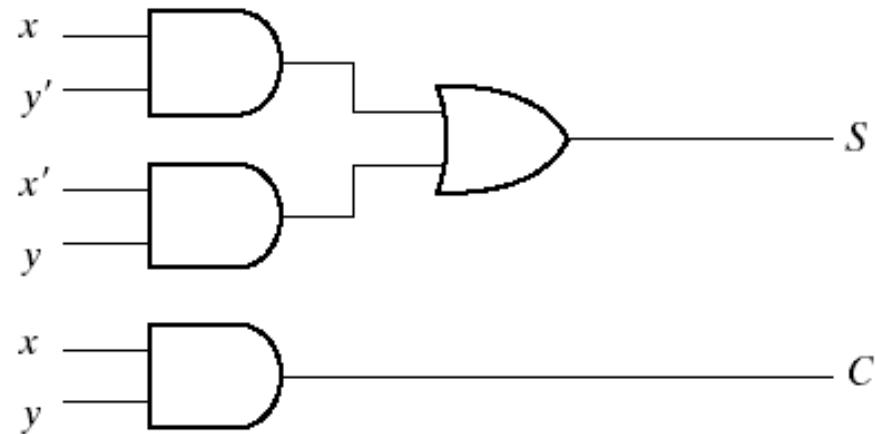
# Gate-level Modeling

- Two or more modules can be combined to build a hierarchical description of a design.
- There are two basic types of design methodologies.
  - **Top down:** In top-down design, the top level block is defined and then sub-blocks necessary to build the top level block are identified.
  - **Bottom up:** Here the building blocks are first identified and then combine to build the top level block.
- In a top-down design, a 4-bit binary adder is defined as top-level block with 4 full adder blocks. Then we describe two half-adders that are required to create the full adder.
- In a bottom-up design, the half-adder is defined, then the full adder is constructed and the 4-bit adder is built from the full adders.

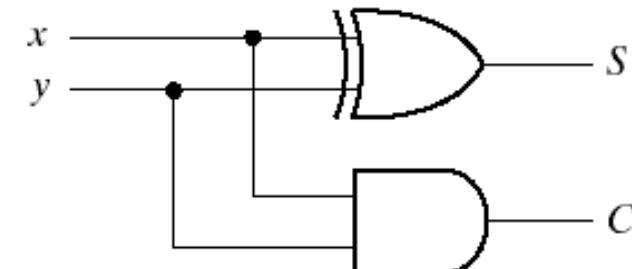
# Gate-level Modeling

- A bottom-up hierarchical description of a 4-bit adder is described in Verilog as
  - Half adder: defined by instantiating primitive gates.
  - Then define the full adder by instantiating two half-adders.
  - Finally the third module describes 4-bit adder by instantiating 4 full adders.
- **Note:** In Verilog, one module definition cannot be placed within another module description.

# 4-bit Half Adder



$$(a) S = xy' + x'y \\ C = xy$$



$$(b) S = x \oplus y \\ C = xy$$

# 4-bit Full Adder

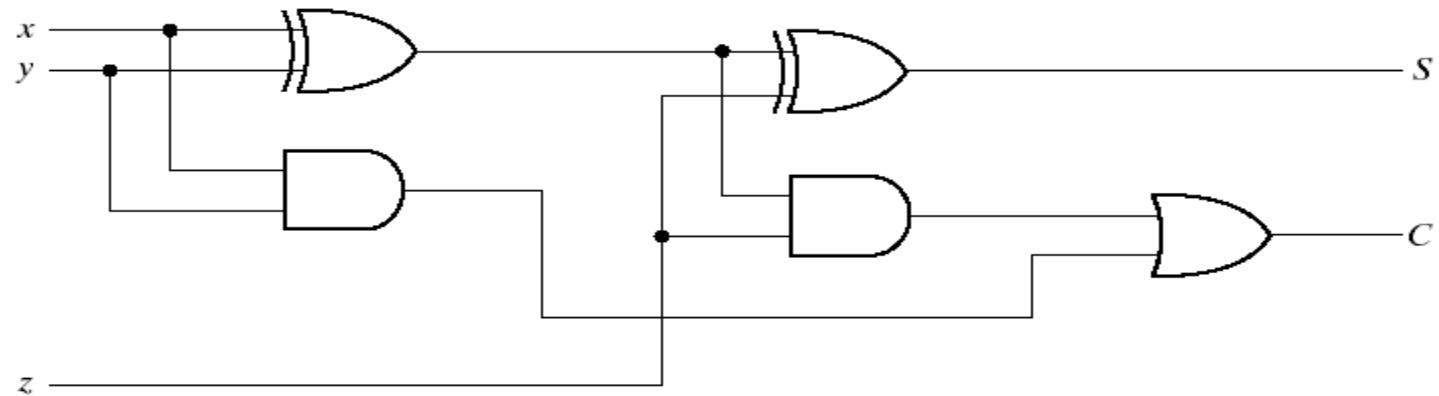


Fig. 4-8 Implementation of Full Adder with Two Half Adders and an OR Gate

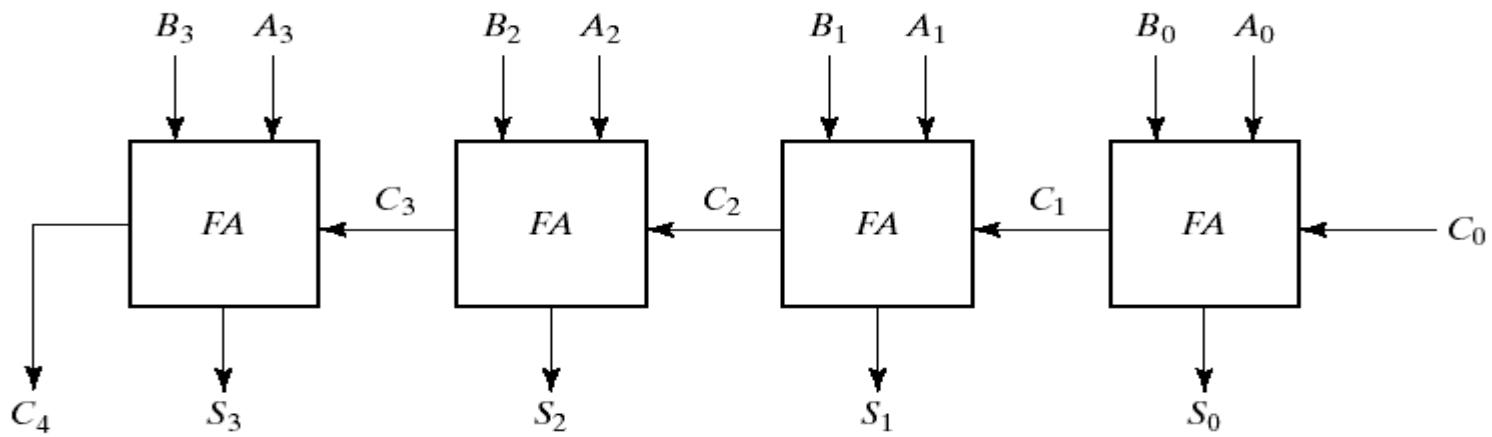


Fig. 4-9 4-Bit Adder

# 4-bit Full Adder

```
//Gate-level hierarchical description of 4-bit adder
module halfadder (S,C,x,y);
    input x,y;
    output S,C;
    //Instantiate primitive gates
    xor (S,x,y);
    and (C,x,y);
endmodule

module fulladder (S,C,x,y,z);
    input x,y,z;
    output S,C;
    wire S1,D1,D2; //Outputs of first XOR and two AND
gates
    //Instantiate the half adders
    halfadder HA1(S1,D1,x,y), HA2(S,D2,S1,z);
    or g1(C,D2,D1);
endmodule
```

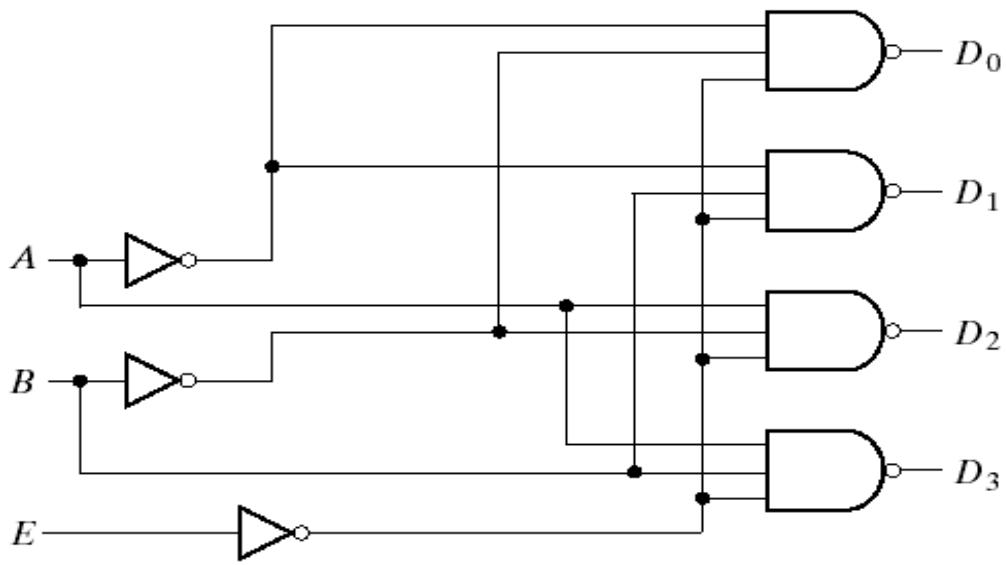
# 4-bit Full Adder

```
module _4bit_adder (S,C4,A,B,C0);
    input [3:0] A,B;
    input C0;
    output [3:0] S;
    output C4;
    wire C1,C2,C3; //Intermediate carries

    //Instantiate the full adder
    fulladder FA0 (S[0],C1,A[0],B[0],C0),
                FA1 (S[1],C2,A[1],B[1],C1),
                FA2 (S[2],C3,A[2],B[2],C2),
                FA3 (S[3],C4,A[3],B[3],C3);

endmodule
```

# 2 to 4 Decoder



(a) Logic diagram

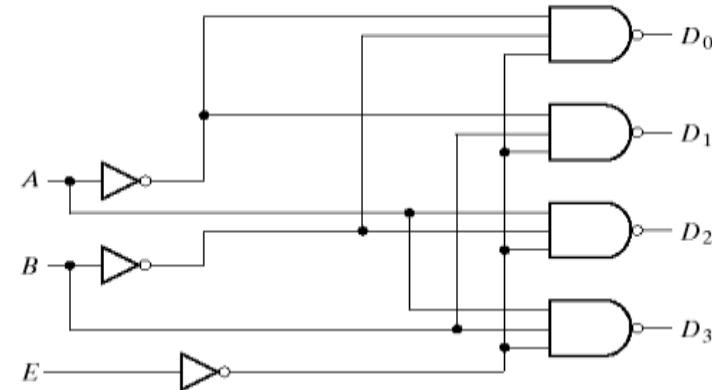
E	A	B	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

Fig. 4-19 2-to-4-Line Decoder with Enable Input

## 2 to 4 Decoder

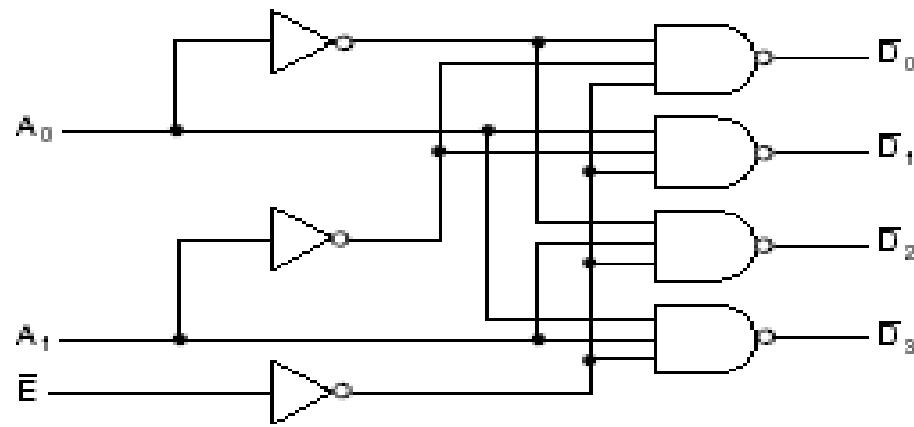
```
//Gate-level description of a 2-to-4-line decoder
module decoder_gl (A,B,E,D);
    input A,B,E;
    output [0:3] D;
    wire Anot,Bnot,Enot;
    not
        n1 (Anot,A),
        n2 (Bnot,B),
        n3 (Enot,E);
    nand
        n4 (D[0],Anot,Bnot,Enot),
        n5 (D[1],Anot,B,Enot),
        n6 (D[2],A,Bnot,Enot),
        n7 (D[3],A,B,Enot);
endmodule
```



(a) Logic diagram

Fig. 4-19 2-to-4-Line Decoder with

# 2-to-4 Line Decoder



(a) Logic diagram

E	A <sub>1</sub>	A <sub>0</sub>	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

(b) Truth table

$$D_0 = \overline{E} \overline{A}_1 A_0$$

$$D_1 = \overline{E} \overline{A}_1 A_0$$

$$D_2 = \overline{E} A_1 \overline{A}_0$$

$$D_3 = E A_1 A_0$$

(c) Logic Equations

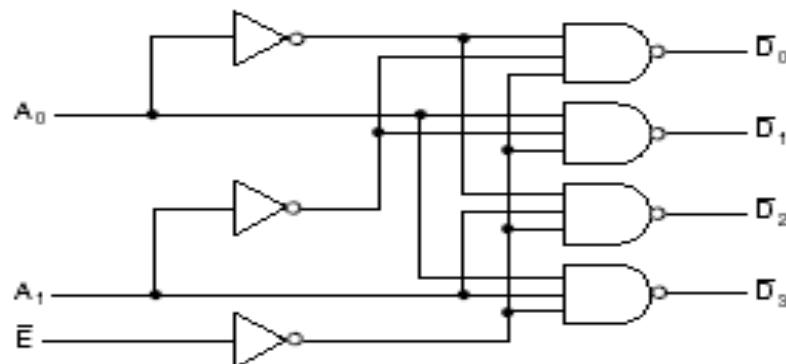
Fig.3-14 A 2-to-4-Line Decoder

## 2-to-4 Line Decoder

```
//2 to 4 line decoder
module decoder_2_to_4_st_v(E_n, A0, A1, D0_n, D1_n,
D2_n, D3_n);

input E_n, A0, A1;
output D0_n, D1_n, D2_n, D3_n;
wire A0_n, A1_n, E;
not g0(A0_n, A0), g1(A1_n, A1), g2(E, E_n);
nand g3(D0_n, A0_n, A1_n, E), g4(D1_n, A0, A1_n, E),
g5(D2_n, A0_n, A1, E), g6(D3_n, A0, A1, E);

endmodule
```



(a) Logic diagram

E	A <sub>1</sub>	A <sub>0</sub>	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

(b) Truth table

$$\begin{aligned}D_0 &= \overline{E} \overline{A}_1 \overline{A}_0 \\D_1 &= \overline{E} \overline{A}_1 A_0 \\D_2 &= \overline{E} A_1 \overline{A}_0 \\D_3 &= E A_1 A_0\end{aligned}$$

(c) Logic Equations

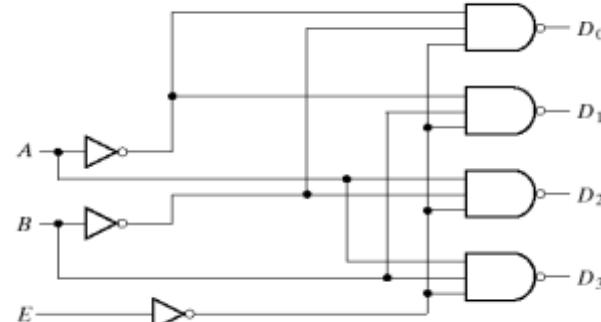
# Dataflow Modeling

- Dataflow modeling uses a number of operators that act on operands to produce desired results.
- Verilog HDL provides about 30 operator types.
- Dataflow modeling uses continuous assignments and the keyword **assign**.
- A continuous assignment is a statement that assigns a value to a net.
- The value assigned to the net is specified by an expression that uses operands and operators.

# Dataflow Modeling - 2-to-4-line decoder

```
//Dataflow description of a 2-to-4-line decoder
module decoder_df (A,B,E,D);

    input A,B,E;
    output [0:3] D;
    assign D[0] = ~ (~A & ~B & ~E) ,
              D[1] = ~ (~A & B & ~E) ,
              D[2] = ~ (A & ~B & ~E) ,
              D[3] = ~ (A & B & ~E) ;
endmodule
```



A 2-to-1 line multiplexer with data inputs A and B, select input S, and output Y is described with the continuous assignment

```
assign Y = (A & S) | (B & ~S)
```

# Dataflow Modeling - 4-bit adder and 4-bit comparator

```
//Dataflow description of 4-bit adder
module binary_adder (A,B,Cin,SUM,Cout);
    input [3:0] A,B;
    input Cin;
    output [3:0] SUM;
    output Cout;
    assign {Cout,SUM} = A + B + Cin;
endmodule
```

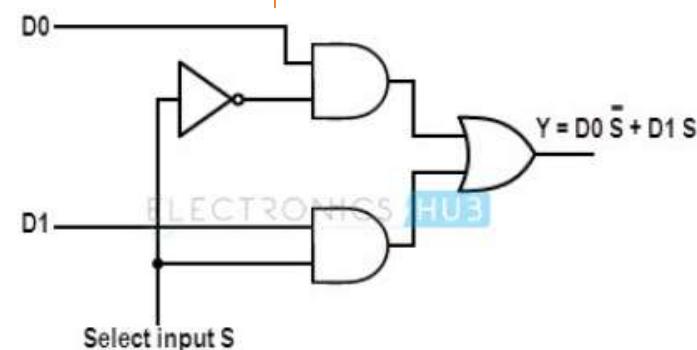
```
//Dataflow description of a 4-bit comparator.
module magcomp (A,B,ALTB,AGTB,AEQB);
    input [3:0] A,B;
    output ALTB,AGTB,AEQB;
    assign ALTB = (A < B),
            AGTB = (A > B),
            AEQB = (A == B);
endmodule
```

# Dataflow Modeling - 2-to-1-line mux

- Dataflow Modeling provides the means of describing combinational circuits by their function rather than by their gate structure.
- **Conditional operator (?:)**  
condition ? true-expression : false-expression;
- A 2-to-1 line multiplexer  
**assign** OUT = select ? A : B;

```
//Dataflow description of 2-to-1-line mux

module mux2x1_df (A,B,select,OUT);
    input A,B,select;
    output OUT;
    assign OUT = select ? A : B;
endmodule
```



# HDL Behavioral Modeling

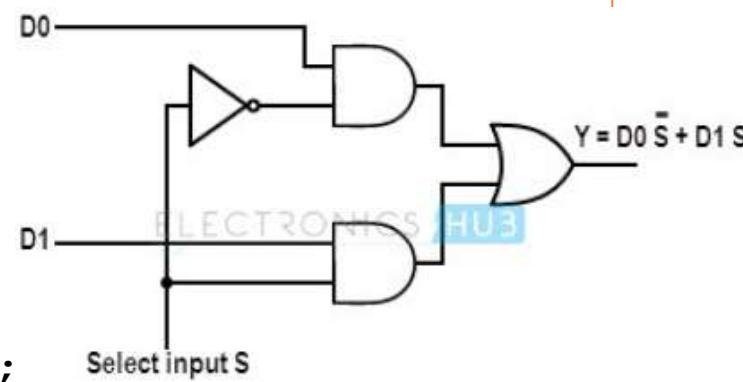
# Behavioral Modeling

- Behavioral modeling represents digital circuits at a functional and algorithmic level.
- It is used mostly to describe sequential circuits, but can also be used to describe combinational circuits.
- Behavioral descriptions use the keyword **always** followed by a list of procedural assignment statements.
- The target output of procedural assignment statements must be of the **reg** data type.
- A **reg** data type retains its value until a new value is assigned.

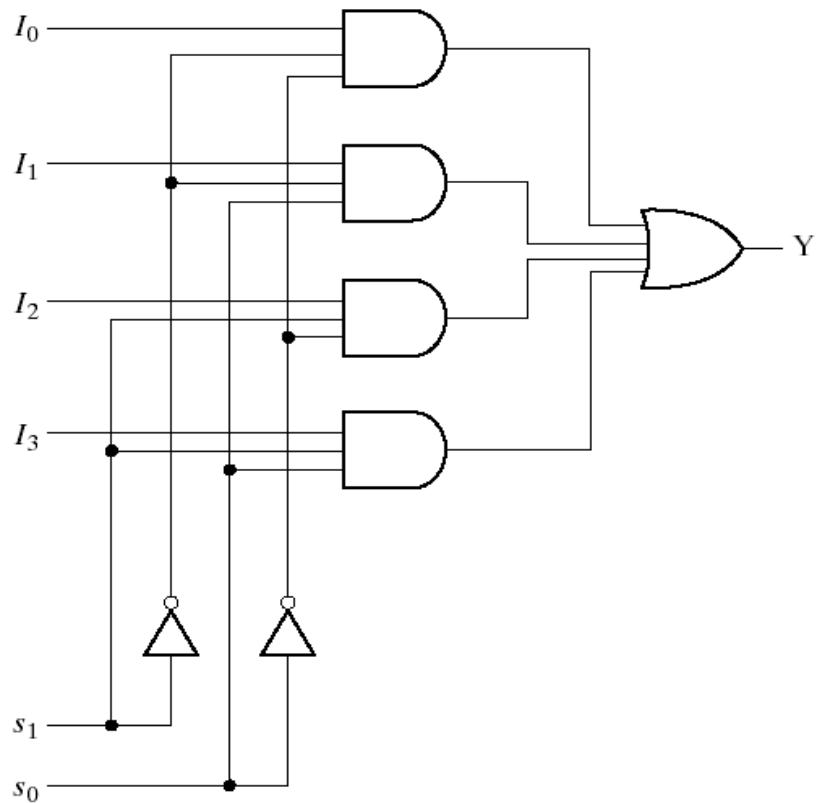
# Behavioral Modeling 2-to-1-line mux

- The procedural assignment statements inside the **always** block are executed every time there is a change in any of the variable listed after the @ symbol. (Note that there is no ";" at the end of **always** statement)

```
//Behavioral description of 2-to-1-line multiplexer
module mux2x1_bh(A,B,select,OUT);
    input A,B,select;
    output OUT;
    reg OUT;
    always @(select or A or B)
        if (select == 1) OUT = A;
        else OUT = B;
endmodule
```



# Behavioral Modeling - 4-to-1 line multiplexer



(a) Logic diagram

$s_1$	$s_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

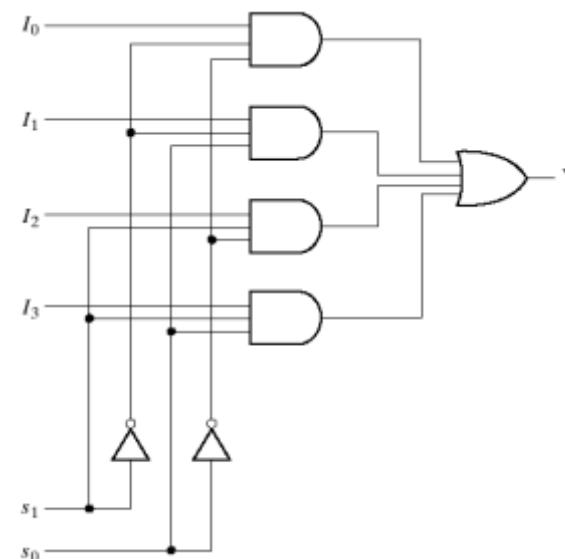
(b) Function table

4-to-1 line  
multiplexer

# Behavioral Modeling 4-to-1 line multiplexer

```
//Behavioral description of 4-to-1 line mux
```

```
module mux4x1_bh (i0,i1,i2,i3,select,y);  
    input i0,i1,i2,i3;  
    input [1:0] select;  
    output y;  
    reg y;  
    always @ (i0 or i1 or i2 or i3 or select)  
        case (select)  
            2'b00: y = i0;  
            2'b01: y = i1;  
            2'b10: y = i2;  
            2'b11: y = i3;  
        endcase  
endmodule
```



# Behavioral Modeling 4-to-1 line multiplexer

- In 4-to-1 line multiplexer, the select input is defined as a 2-bit vector and output y is declared as a reg data.
- The always block has a sequential block enclosed between the keywords **case** and **endcase**.
- The block is executed whenever any of the inputs listed after the @ symbol changes in value.

# Descriptions of Circuits

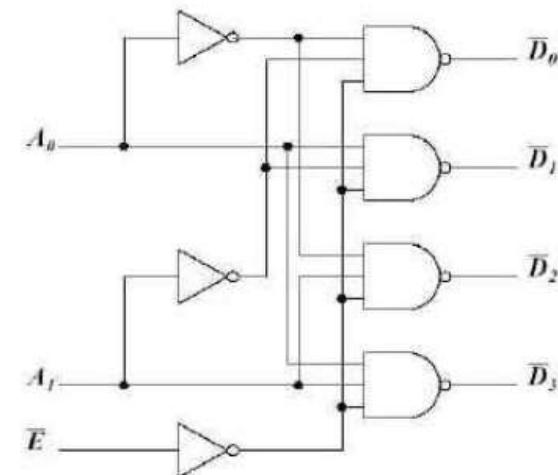
- ***Structural Description*** – This is directly equivalent to the schematic of a circuit and is specifically oriented to describing hardware structures using the components of a circuit.
- ***Dataflow Description*** – This describes a circuit in terms of function rather than structure and is made up of concurrent assignment statements or their equivalent. Concurrent assignments statements are executed concurrently, i.e. in parallel whenever one of the values on the right hand side of the statement changes.

# Descriptions of Circuits

- **Hierarchical Description** – Descriptions that represent circuits using hierarchy have multiple entities, one for each element of the Hierarchy.
- **Behavioral Description** – This refers to a description of a circuit at a level higher than the logic level. This type of description is also referred to as the *register transfers level*.

# 2-to-4 Line Decoder – Data flow description

```
//2-to-4 Line Decoder: Dataflow
module dec_2_to_4_df(E_n,A0,A1,D0_n,D1_n,D2_n,D3_n);
    input E_n, A0, A1;
    output D0_n,D1_n,D2_n,D3_n;
    assign D0_n=~ (~E_n&~A1&~A0);
    assign D1_n=~ (~E_n&~A1& A0);
    assign D2_n=~ (~E_n& A1&~A0);
    assign D3_n=~ (~E_n& A1& A0);
endmodule
```



# 4-to-1 Multiplexer

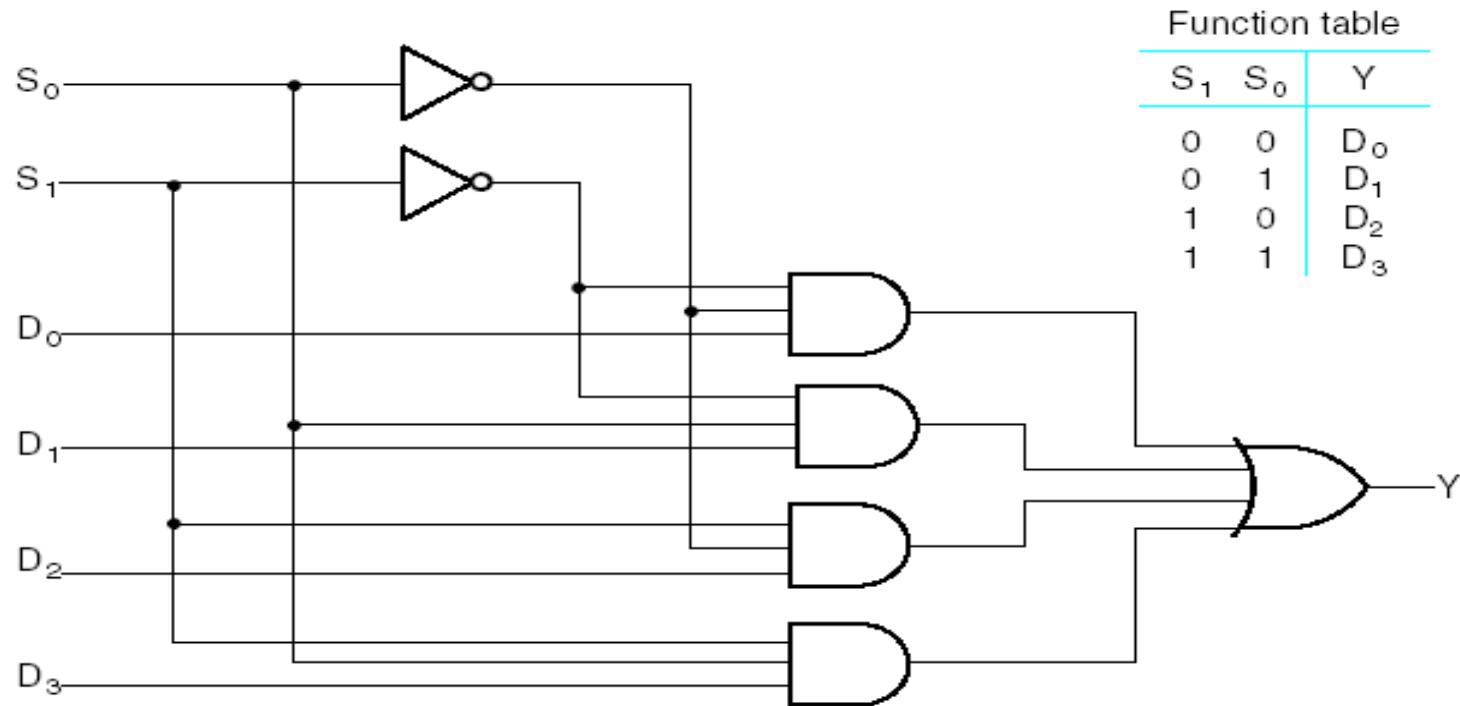


Fig. 3-19 4-to-1-Line Multiplexer

# 4-to-1 Multiplexer

```
//4-to-1 Mux: Structural Verilog
module mux_4_to_1_st_v(S,D,Y);
    input [1:0]S;
    input [3:0]D;
    output Y;
    wire [1:0]not_s;
    wire [0:3]N;
    not g0(not_s[0],S[0]),g1(not_s[1],S[1]);
    and g2(N[0],not_s[0],not_s[1],D[0]),
        g3(N[1],S[0],not_s[1],D[0]),
        g4(N[2],not_s[0],S[1],D[0]),
        g5(N[3],S[0],S[1],D[0]);
    or g5(Y,N[0],N[1],N[2],N[3]);
endmodule
```

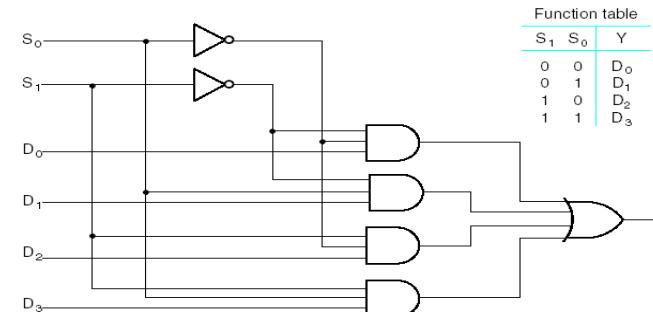
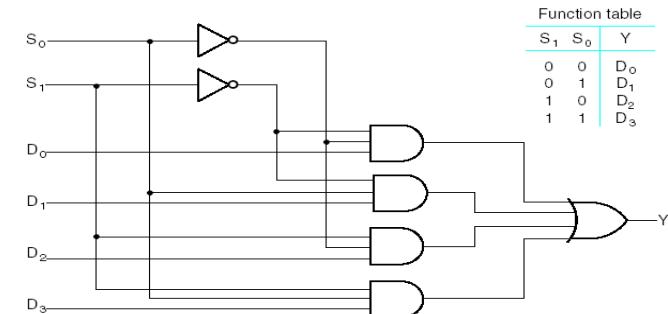


Fig. 3-19 4-to-1-Line Multiplexer

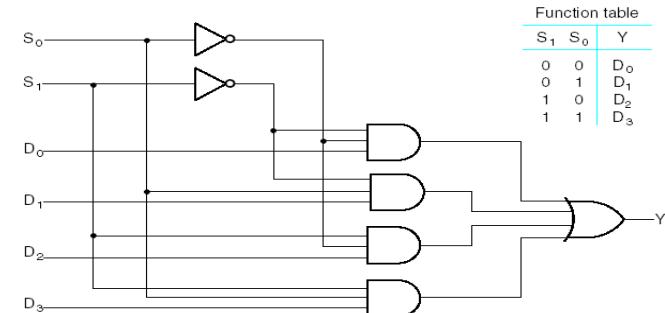
# 4-to-1 Multiplexer – Data Flow

```
//4-to-1 Mux: Dataflow description
module mux_4_to_1(S, D, Y);
    input [1:0]S;
    input [3:0]D;
    output Y;
    assign Y = (~S[1]&~S[0]&D[0]) | (~S[1]&S[0]&D[1])
              | (S[1]&~S[0]&D[2]) | (S[1]&S[0]&D[3]);
endmodule
```



```
//4-to-1 Mux: Conditional Dataflow description
module mux_4_to_1(S, D, Y);
    input [1:0]S;
    input [3:0]D;
    output Y;
    assign Y = (S==2'b00)?D[0] : (S==2'b01)?D[1] :
               (S==2'b10)?D[2] : (S==2'b11)?D[3]:1'bx;;
endmodule
```

# 4-to-1 Multiplexer



Function table		
$S_1$	$S_0$	$Y$
0	0	$D_0$
0	1	$D_1$
1	0	$D_2$
1	1	$D_3$

Fig. 3-19 4-to-1-Line Multiplexer

```
//4-to-1 Mux: Dataflow Verilog Description
module mux_4_to_1(S,D,Y);
    input [1:0]S;
    input [3:0]D;
    output Y;
    assign Y=S[1]? (S[0]?D[3]:D[2]):(S[0]?D[1]:D[0]);
endmodule
```

# Adder

□ TABLE 3-7  
Truth Table of Half Adder

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

Table 3-7 Truth Table of Half Adder

□ TABLE 3-8  
Truth Table of Full Adder

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

Table 3-8 Truth Table of Full Adder

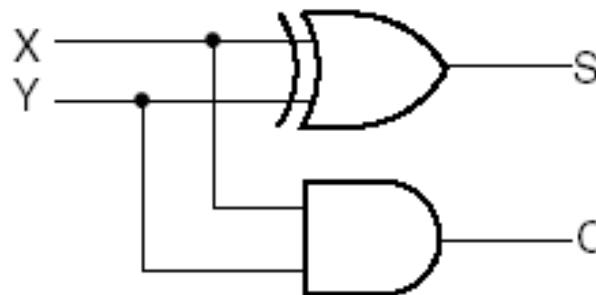
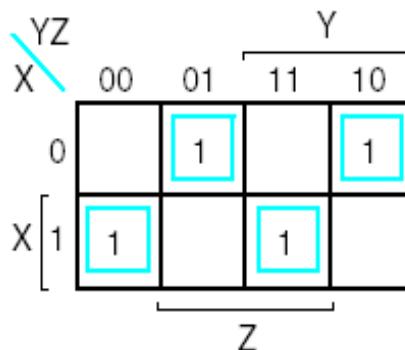
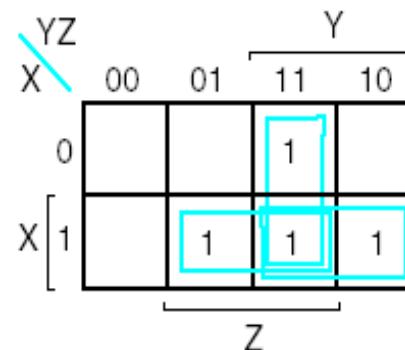


Fig. 3-25 Logic Diagram of Half Adder



$$\begin{aligned} S &= \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ \\ &= X \oplus Y \oplus Z \end{aligned}$$



$$\begin{aligned} C &= XY + XZ + YZ \\ &= XY + Z(\bar{X}Y + \bar{X}Y) \\ &= XY + Z(X \oplus Y) \end{aligned}$$

Fig. 3-26 Maps for Full Adder

# 4-bit Adder

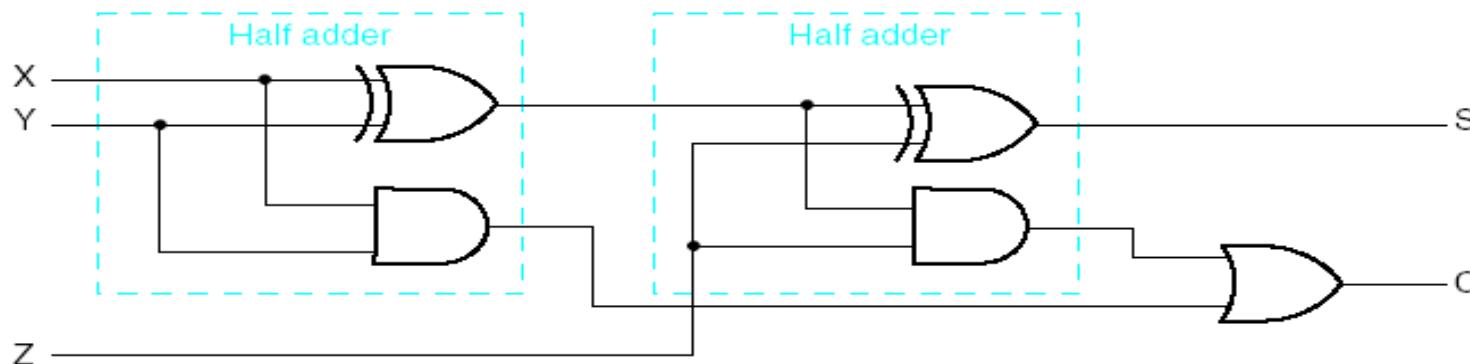


Fig. 3-27 Logic Diagram of Full Adder

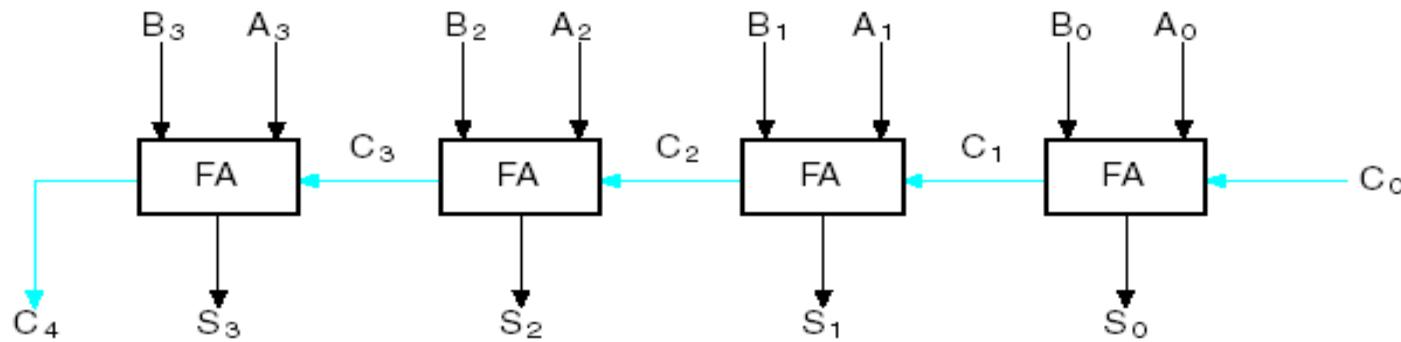


Fig. 3-28 4-Bit Ripple Carry Adder

## 4-bit-Adder

```
// 4-bit Adder: Hierarchical Dataflow/Structural
// (See Figures 3-27 and 3-28 for logic diagrams)

module half_adder_v(x, y, s, c);
    input x, y;
    output s, c;

    assign s = x ^ y;
    assign c = x & y;

endmodule

module full_adder_v(x, y, z, s, c);
    input x, y, z;
    output s, c;

    wire hs, hc, tc;

    half_adder_v    HA1(x, y, hs, hc),
                    HA2(hs, z, s, tc);
    assign c = tc | hc;

endmodule

module adder_4_v(B, A, C0, S, C4);
    input[3:0] B, A;
    input C0;
    output[3:0] S;
    output C4;

    wire[3:1] C;

    full_adder_v Bit0(B[0], A[0], C0, S[0], C[1]),
                  Bit1(B[1], A[1], C[1], S[1], C[2]),
                  Bit2(B[2], A[2], C[2], S[2], C[3]),
                  Bit3(B[3], A[3], C[3], S[3], C4);

endmodule
```

## 4-bit Adder

```
// 4-bit adder : dataflow description
module adder_4bit (A,B,C0,S,C4);
    input [3:0] A,B;
    input C0;
    output [3:0] S;
    output C4;
    assign {C4,S} = A + B + C0;
endmodule
```

B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

UNIT 4- Session-1

Introduction to Sequential circuits, Latch and Flip-Flops

# Sequential Circuits

Every digital system is likely to have combinational circuits, most systems encountered in practice also include **storage elements**, which require that the system be described in term of **sequential logic**.

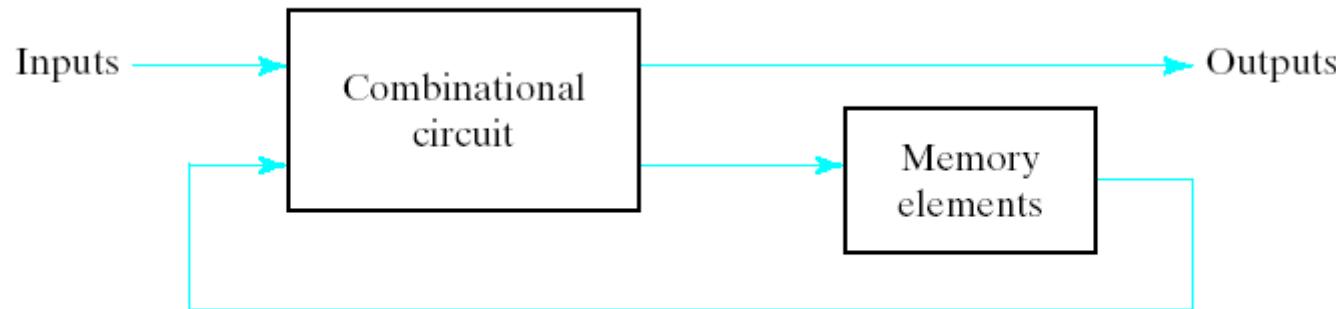


Fig. 5-1 Block Diagram of Sequential Circuit

# Synchronous Clocked Sequential Circuit

A sequential circuit may use many **flip-flops** to store as many bits as necessary. The outputs can come either from the combinational circuit or from the flip-flops or both.

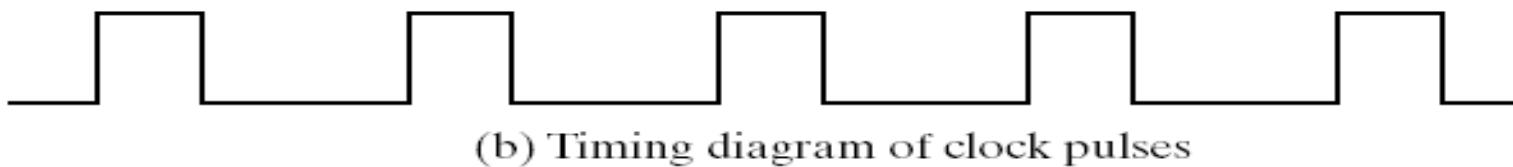
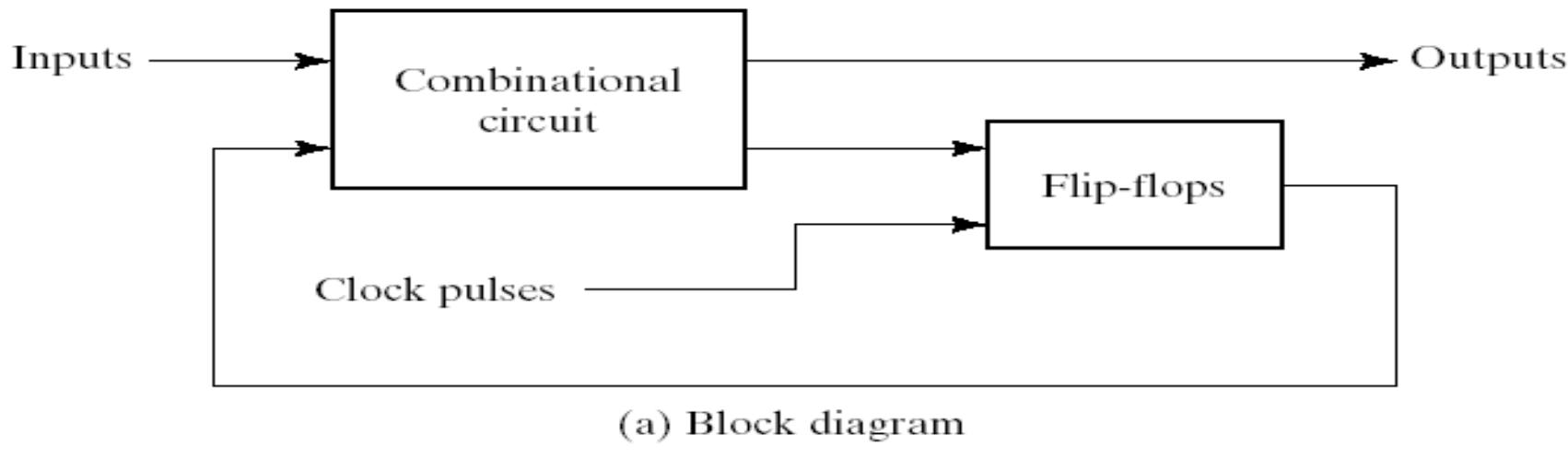
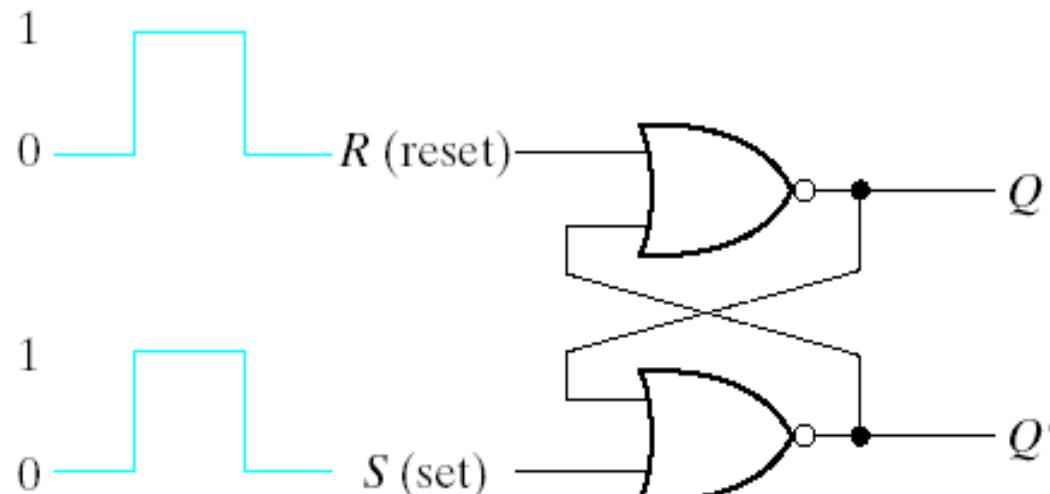


Fig. 5-2 Synchronous Clocked Sequential Circuit

# Latches --SR Latch

The **SR latch** is a circuit with two cross-coupled **NOR gates** or two cross-coupled **NAND gates**. It has two inputs labeled S for set and R for reset.



(a) Logic diagram

S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

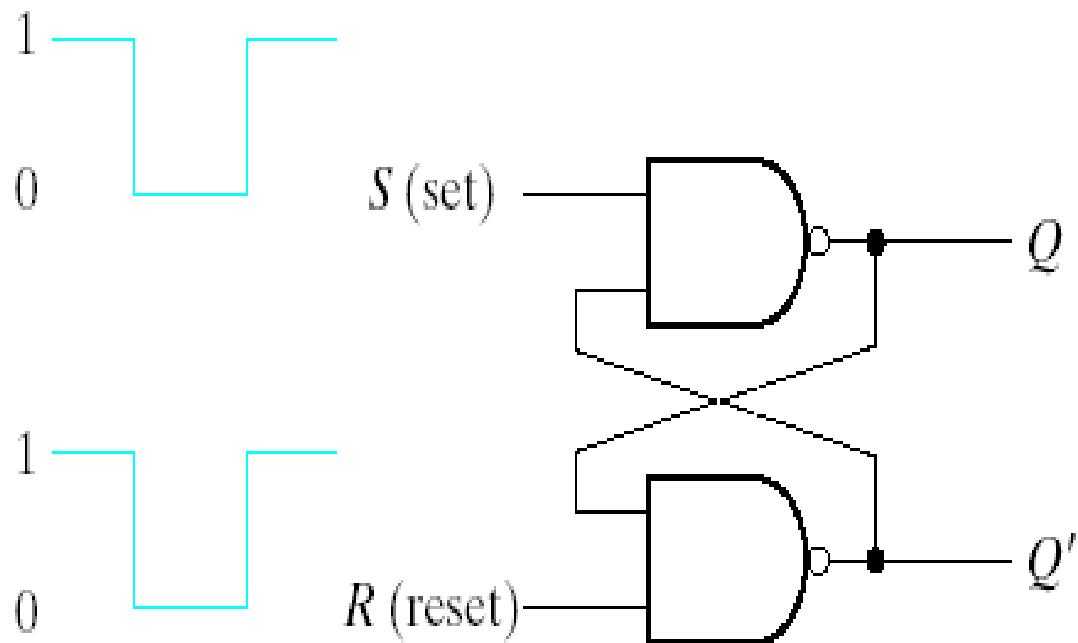
(after S = 1, R = 0)

(after S = 0, R = 1)

(b) Function table

Fig. 5-3 SR Latch with NOR Gates

# SR Latch with NAND Gates



(a) Logic diagram

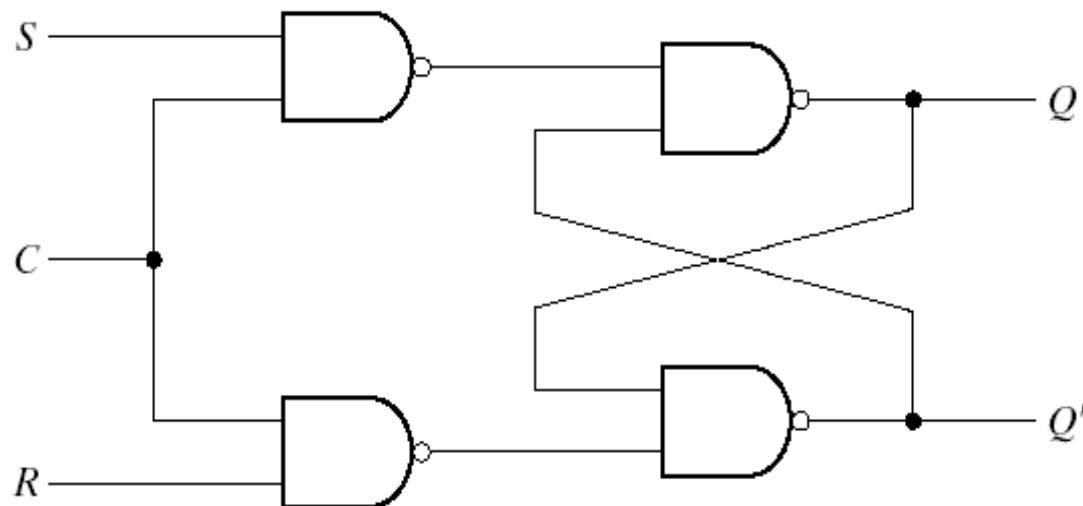
S	R	Q	$Q'$	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$ )
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$ )
0	0	1	1	

(b) Function table

Fig. 5-4 SR Latch with NAND Gates

# SR Latch with Control Input

The operation of the basic SR latch can be modified by providing an additional control input that determines when the state of the latch can be changed. In Fig. 5-5, it consists of the **basic SR latch** and **two additional NAND gates**.



(a) Logic diagram

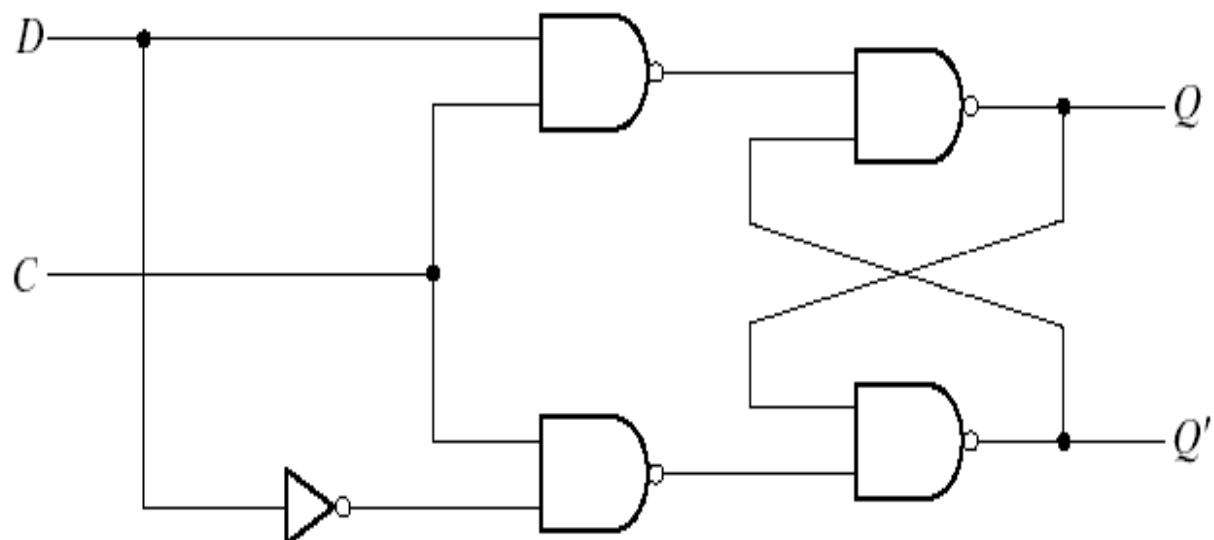
C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$ ; Reset state
1	1	0	$Q = 1$ ; set state
1	1	1	Indeterminate

(b) Function table

Fig. 5-5 SR Latch with Control Input

# D Latch

One way to eliminate the undesirable condition of the indeterminate state in SR latch is to ensure that **inputs S and R are never equal to 1 at the same time** in Fig 5-5. This is done in the **D latch**.



(a) Logic diagram

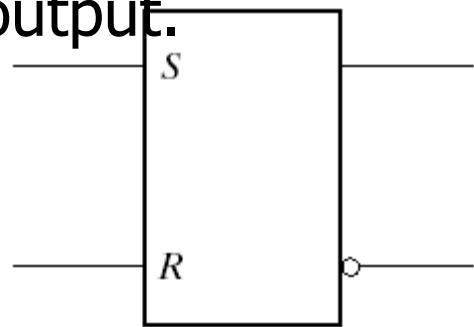
$C$	$D$	Next state of $Q$
0	X	No change
1	0	$Q = 0$ ; Reset state
1	1	$Q = 1$ ; Set state

(b) Function table

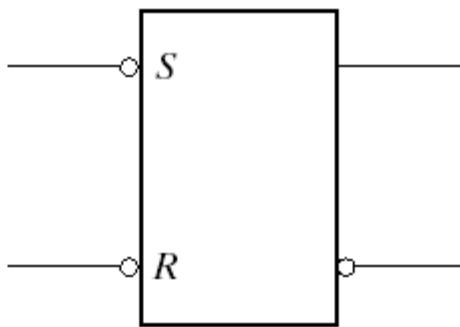
Fig. 5-6 D Latch

# Graphic Symbols for latches

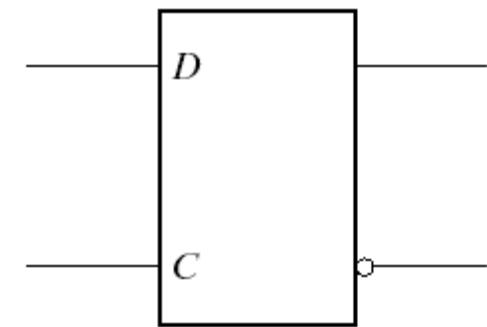
A latch is designated by a rectangular block with inputs on the left and outputs on the right. One output designates the normal output, and the other designates the complement output.



SR



$\bar{S}\bar{R}$



D

Fig. 5-7 Graphic Symbols for Latches

# Flip-Flops

The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a **trigger** and the transition it cause is said to trigger the flip-flop. The D latch with pulses in its control input is essentially a flip-flop that is triggered every time the pulse goes to the **logic 1 level**. As long as the pulse input remains in the level, any changes in the data input will change the output and the state of the latch.

# Clock Response in Latch

In Fig (a) a positive level response in the control input allows changes, in the output when the D input changes while the clock pulse stays at logic 1.



(a) Response to positive level

# Clock Response in Flip-Flop



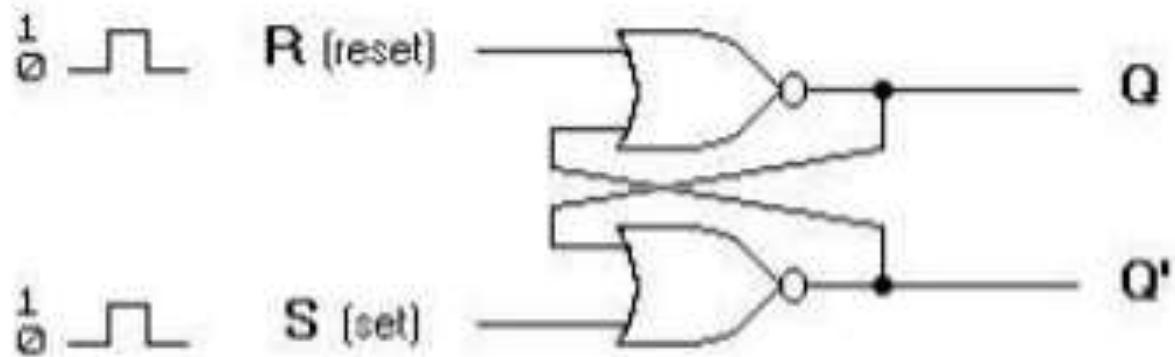
(b) Positive-edge response



(c) Negative-edge response

# S-R Flip Flop

- The SET-RESET flip flop is designed with the help of two NOR gates and also two NAND gates. These flip flops are also called S-R Latch.
- S-R Flip Flop using NOR Gate The design of such a flip flop includes two inputs, called the SET [S] and RESET [R].
- There are also two outputs, Q and Q'.



(a) Logic diagram

$S$	$R$	$Q$	$Q'$
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

(after  $S=1, R=0$ )

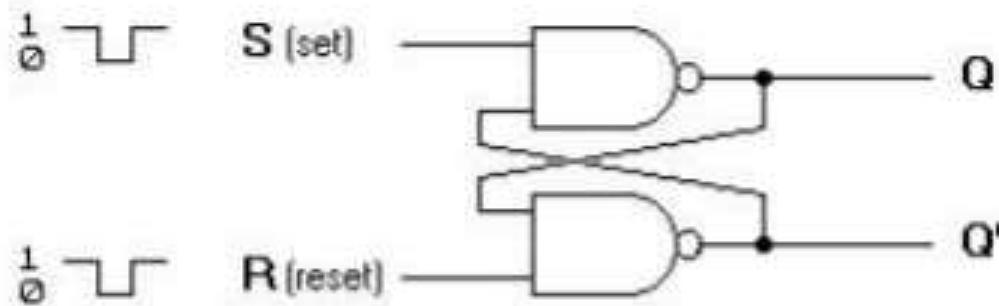
(after  $S=0, R=1$ )

(b) Truth table

Basic flip-flop circuit with NOR gates

- From the diagram it is evident that the flip flop has mainly four states. They are  $S=1$ ,  $R=0-Q=1$ ,  $Q'=0$  This state is also called the SET state.
- $S=0$ ,  $R=1-Q=0$ ,  $Q'=1$  This state is known as the RESET state. In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the value of S.
- $S=0$ ,  $R=0-Q \& Q' =$  Remember If both the values of S and R are switched to 0, then the circuit remembers the value of S and R in their previous state.
- $S=1$ ,  $R=1-Q=0$ ,  $Q'=0$  [Invalid] This is an invalid state because the values of both Q and Q' are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

# S-R Flip Flop using NAND Gate



(a) Logic diagram

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after S=1, R=0)
0	1	1	0	
1	1	1	0	(after S=0, R=1)
0	0	1	1	

(b) Truth table

Basic flip-flop circuit with NAND gates

- Like the NOR Gate S-R flip flop, this one also has four states. They are  $S=1$ ,  $R=0—Q=0$ ,  $Q'=1$  This state is also called the SET state.
- $S=0$ ,  $R=1—Q=1$ ,  $Q'=0$  This state is known as the RESET state. In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the compliment value of S.
- $S=0$ ,  $R=0—Q=1$ , &  $Q' =1$  [Invalid] If both the values of S and R are switched to 0 it is an invalid state because the values of both Q and  $Q'$  are 1. They are supposed to be compliments of each other. Normally, this state must be avoided.
- $S=1$ ,  $R=1—Q & Q'=$  Remember If both the values of S and R are switched to 1, then the circuit remembers the value of S and R in their previous state.

B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

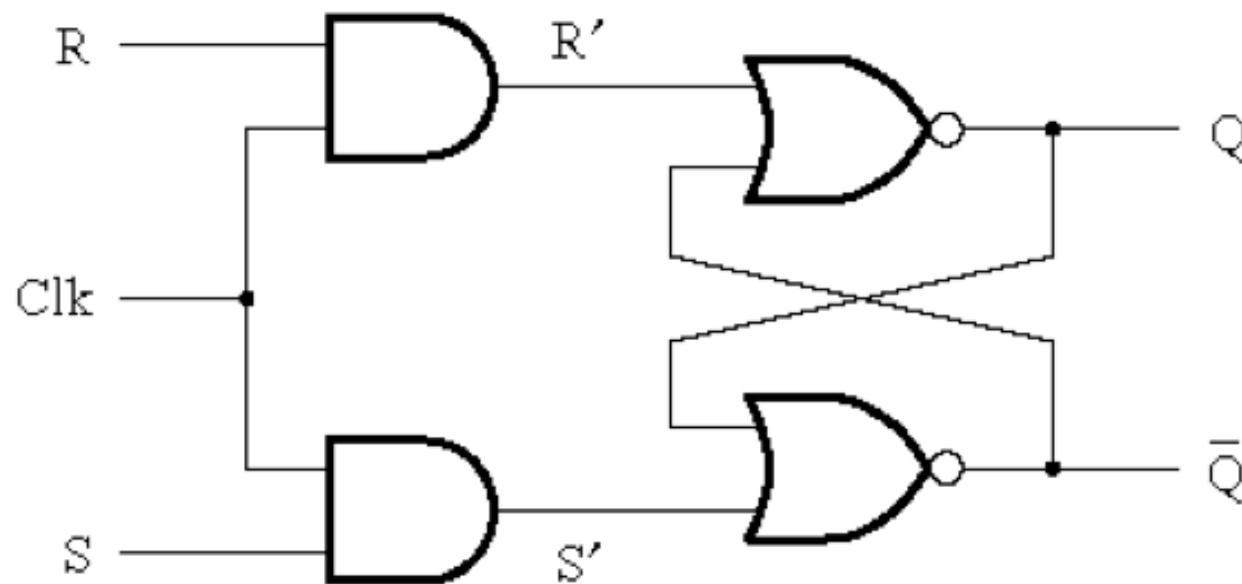
UNIT 4

Session-2 ( Gated SR Flip Flops, Edge Triggered RS Flip-flop)  
Session-3(Edge Triggered D Flip-flops, Edge Triggered T Flip-flops)

# **Gated SR Latch**

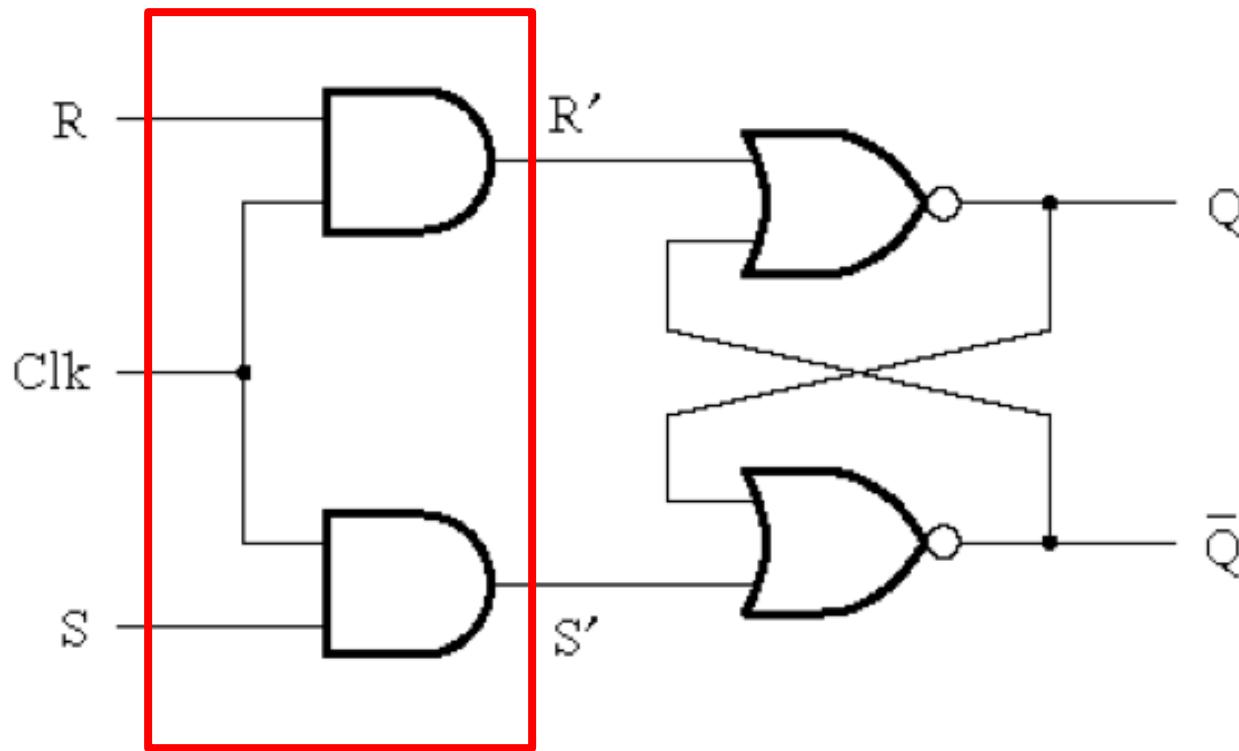
- The basic latch changes its state when the input signals change
- It is hard to control when these input signals will change and thus it is hard to know when the latch may change its state.
- We want to have something like an Enable input
- In this case it is called the “Clock” input because it is desirable for the state changes to be synchronized

# Circuit Diagram for the Gated SR Flip-Flop



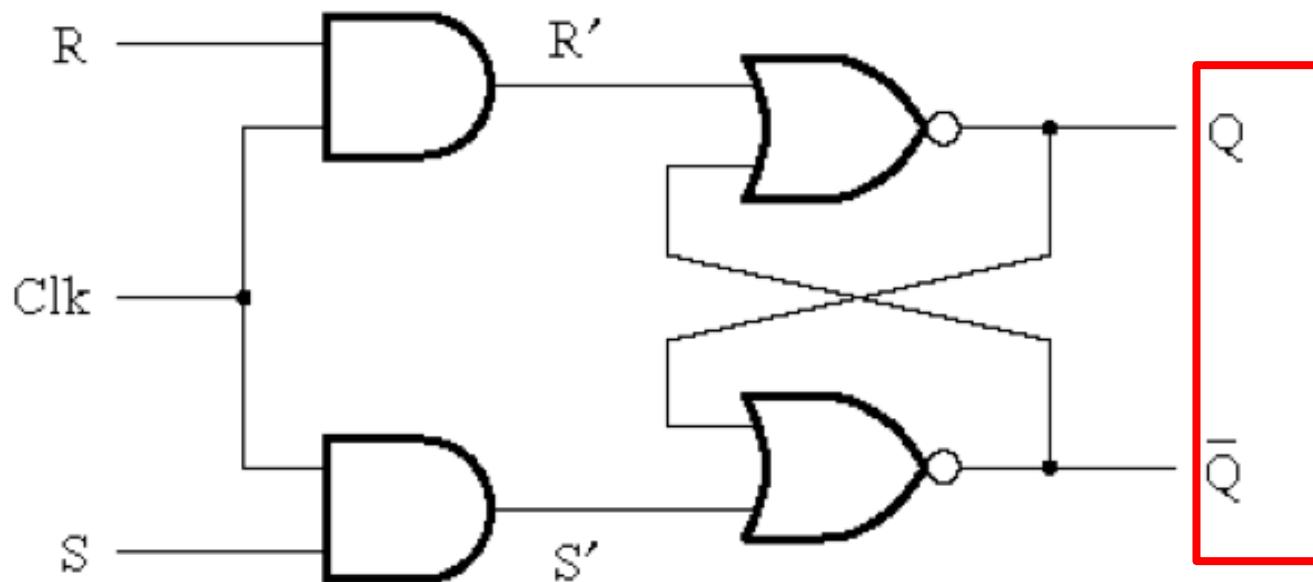
[ Figure 5.5a from the textbook ]

# Circuit Diagram for the Gated SR Flip-Flop



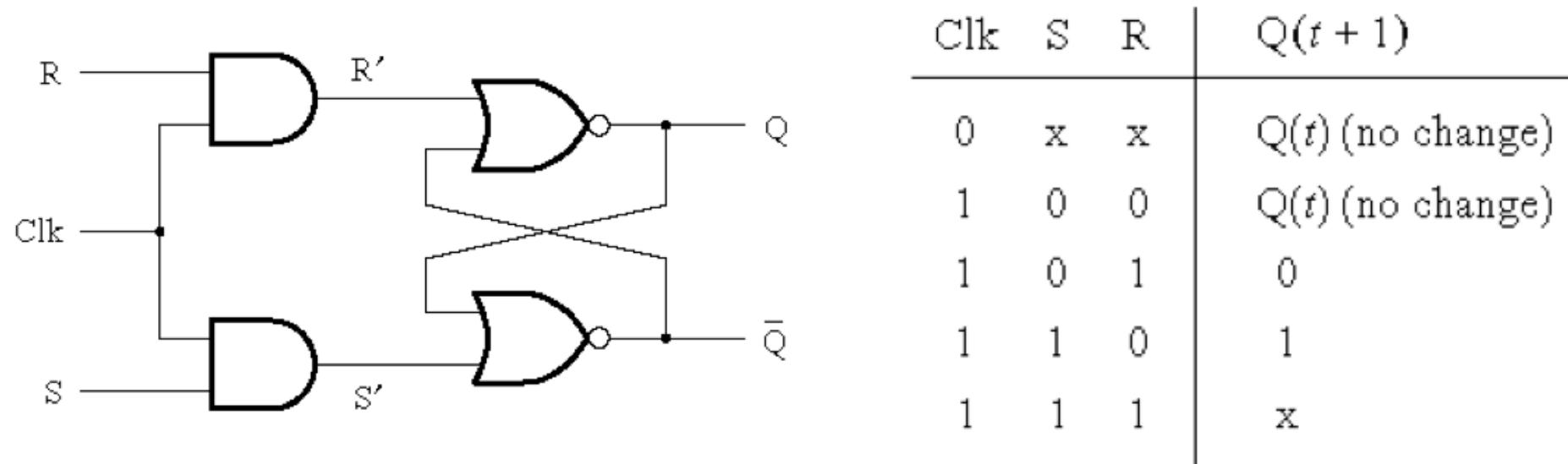
This is the “gate”  
of the gated latch

# Circuit Diagram for the Gated SR Flip-Flop



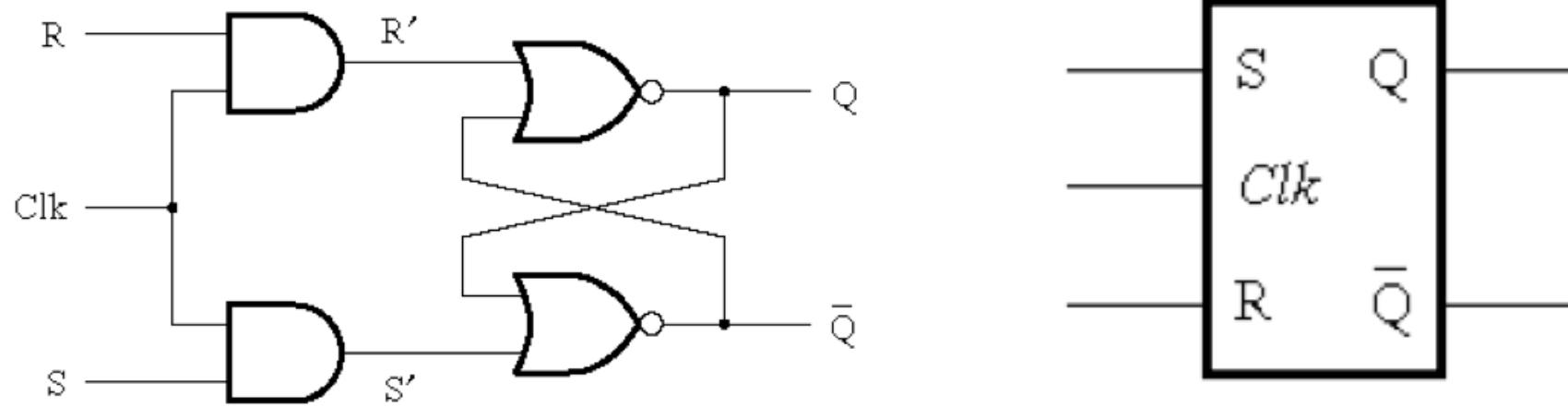
Notice that these  
are  
complements of  
each other

# Circuit Diagram and Characteristic Table for the Gated SR Flip-Flop



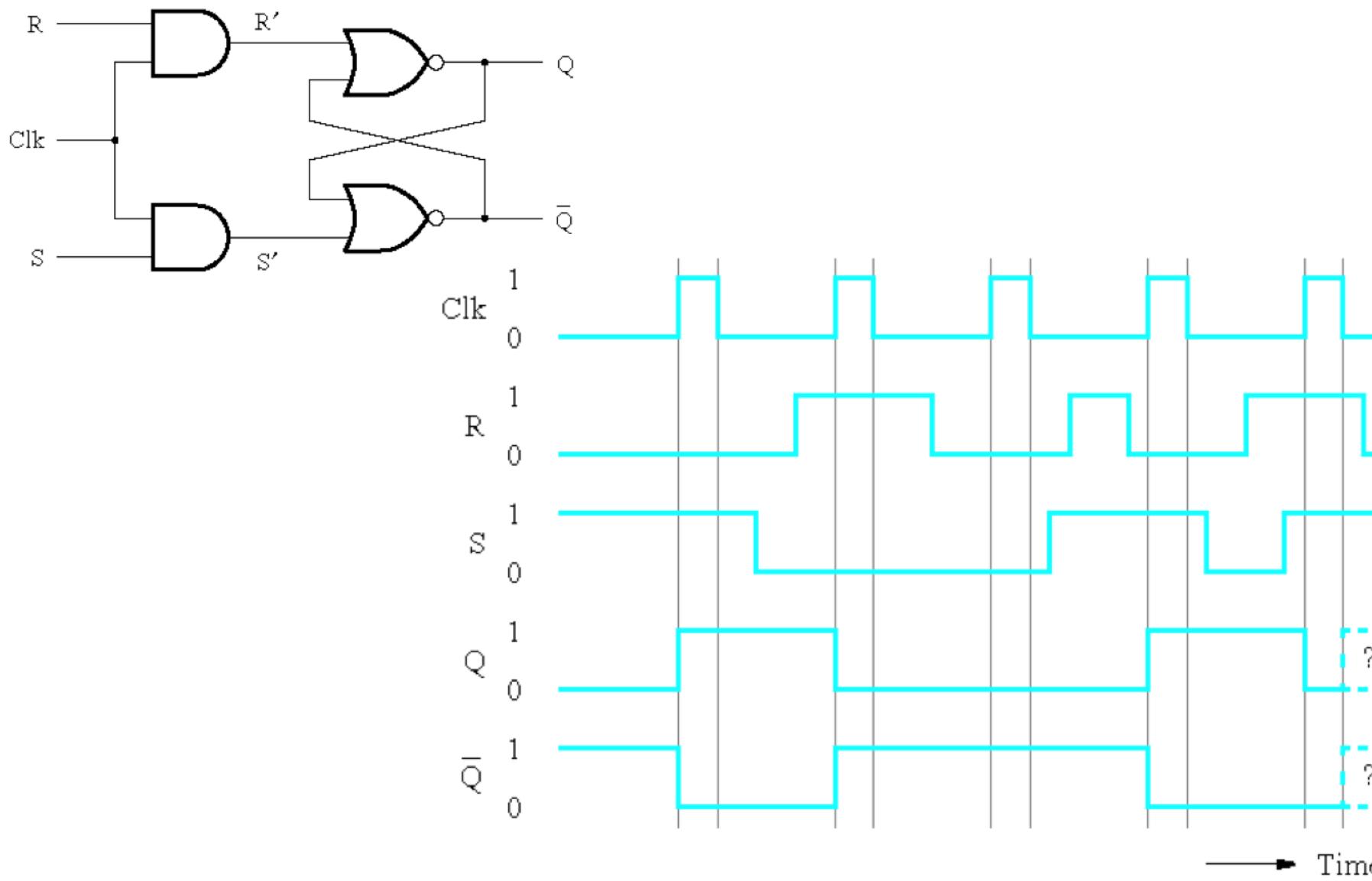
[ Figure 5.5a-b from the textbook ]

# Circuit Diagram and Graphical Symbol for the Gated SR Flip-Flop



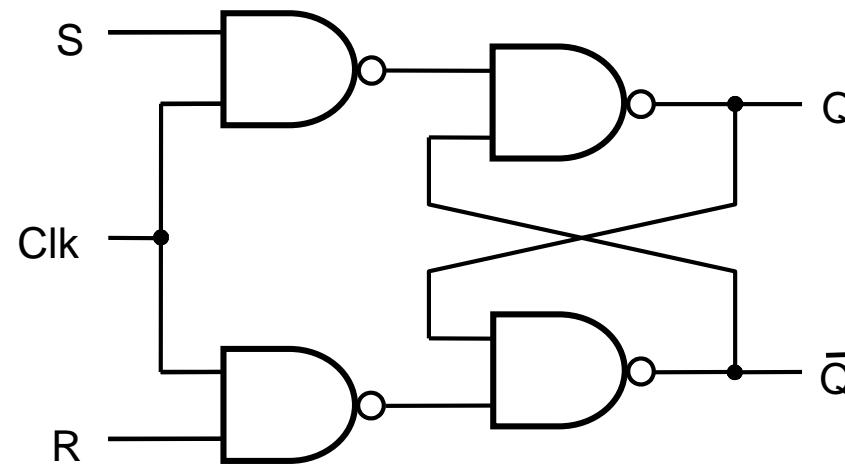
[ Figure 5.5a,c from the textbook ]

# Timing Diagram for the Gated SR Flip-Flop



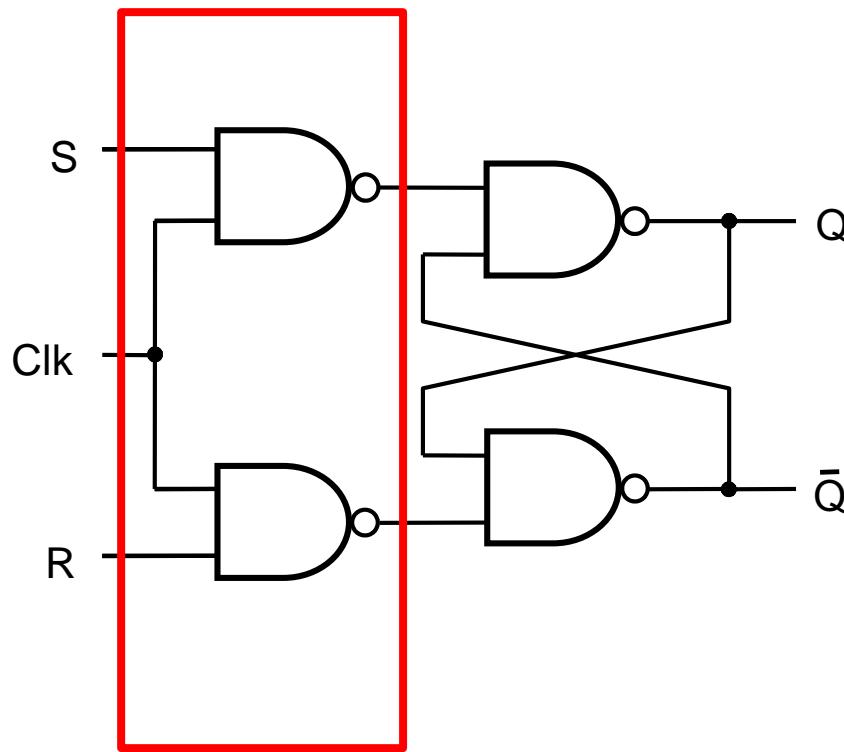
[ Figure 5.5c from the textbook ]

# Gated SR Flip-Flop with NAND gates



[ Figure 5.6 from the textbook ]

# Gated SR Flip-Flop with NAND gates

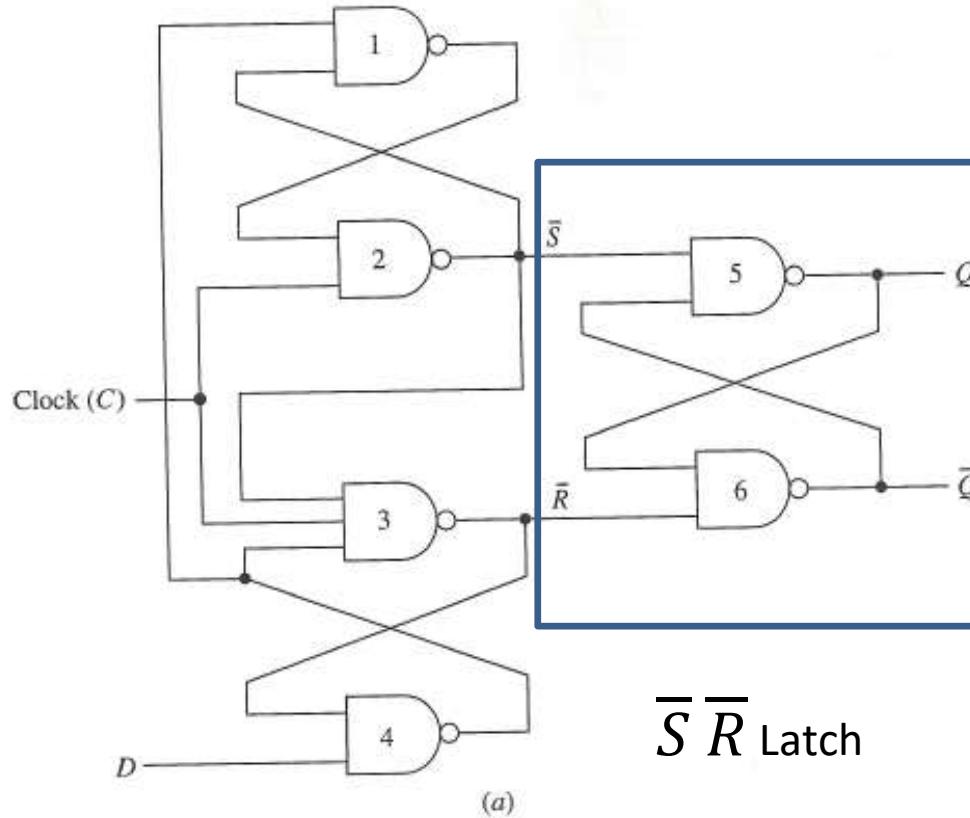


In this case the “gate” is  
constructed using  
NAND gates! Not AND  
gates.

# Edge-Triggered Flip-Flops

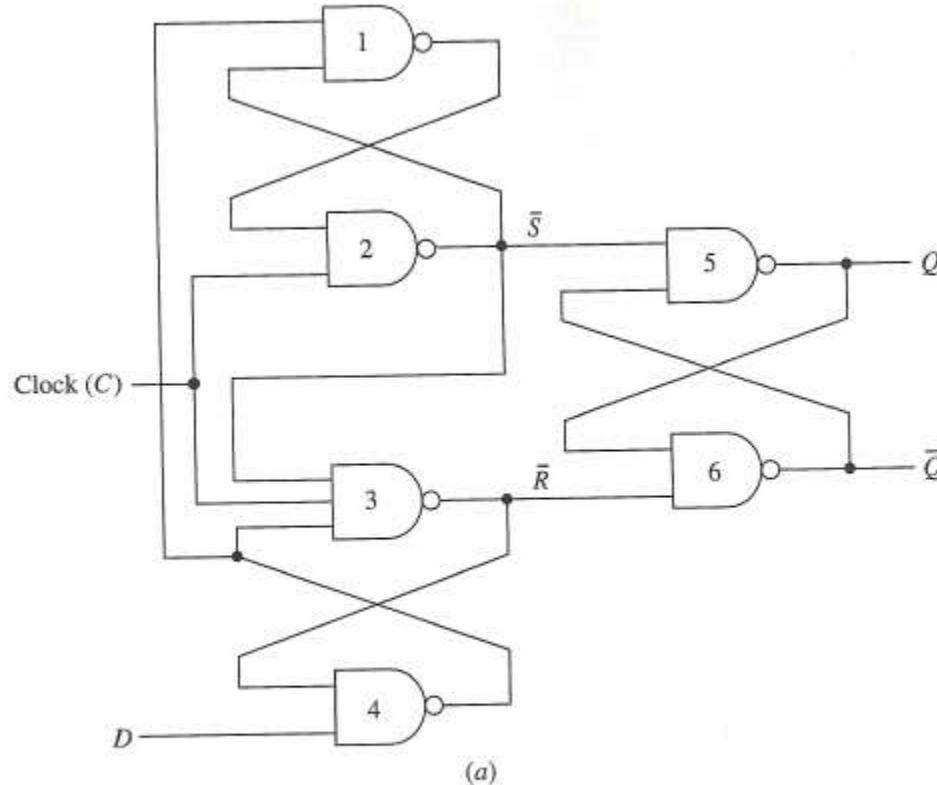
- In basic master-slave flip-flops, master is enabled during the entire period the control input is 1.
  - This can result in 0's and 1's catching.
  - To avoid this, signals on information lines are restricted from changing during the time the master is enabled.
  - Also a delay in the output since master's state is established during the positive edge and transferred to the slave on the negative edge of clock.
- Edge-triggered flip-flops use just one of the edges of the clock signal.
  - This is referred to as the triggering edge.
- Response to triggering edge at the output of the flip-flop is almost immediate (depends only on propagation delay times).
- Once triggering occurs, flip-flop is unresponsive to information input changes until the next triggering edge.

# Edge-Triggered SR Flip-Flops



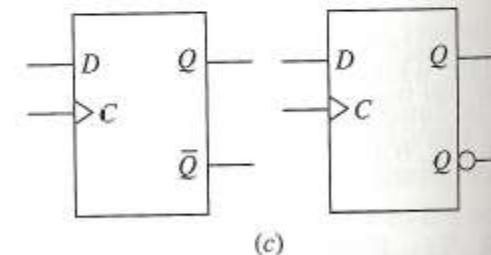
1.  $C = 0$ . Regardless of input at D, outputs of gates 2,3 are 1. So  $\bar{S} = \bar{R} = 1$ . State of latch is held.
2. Assume  $D = 0$ : Output of gate 4 is 1, output of gate 1 is 0. When C goes to 1: all inputs to gate 3 are 1, output changes to 0. Output of gate 2 remains at 1 since output of gate 1 is 0. So  $\bar{S} = 1, \bar{R} = 0$ . Output of gate 3 (0) is fed to input of gate 4. Output of gate 4, gate 1 not affected by changes to D.
3. Assume  $C = 0, D = 1$ . Outputs of gates 2,3, are 1. Output of gate 4 is 0, output of gate 1 is 1. When C goes to 1: output of gate 2 is 0, output of gate 3 remains at 1. So  $\bar{S} = 0, \bar{R} = 1$ . Output from gate 2 is input to gates 1, 3 so their outputs remain at 1. Changes in D have no affect on state of flip-flop while C = 1.

# Edge-Triggered SR Flip-Flops



Inputs		Outputs	
$D$	$C$	$Q^+$	$\bar{Q}^+$
0	↑	0	1
1	↑	1	0
X	0	$Q$	$\bar{Q}$
X	1	$Q$	$\bar{Q}$

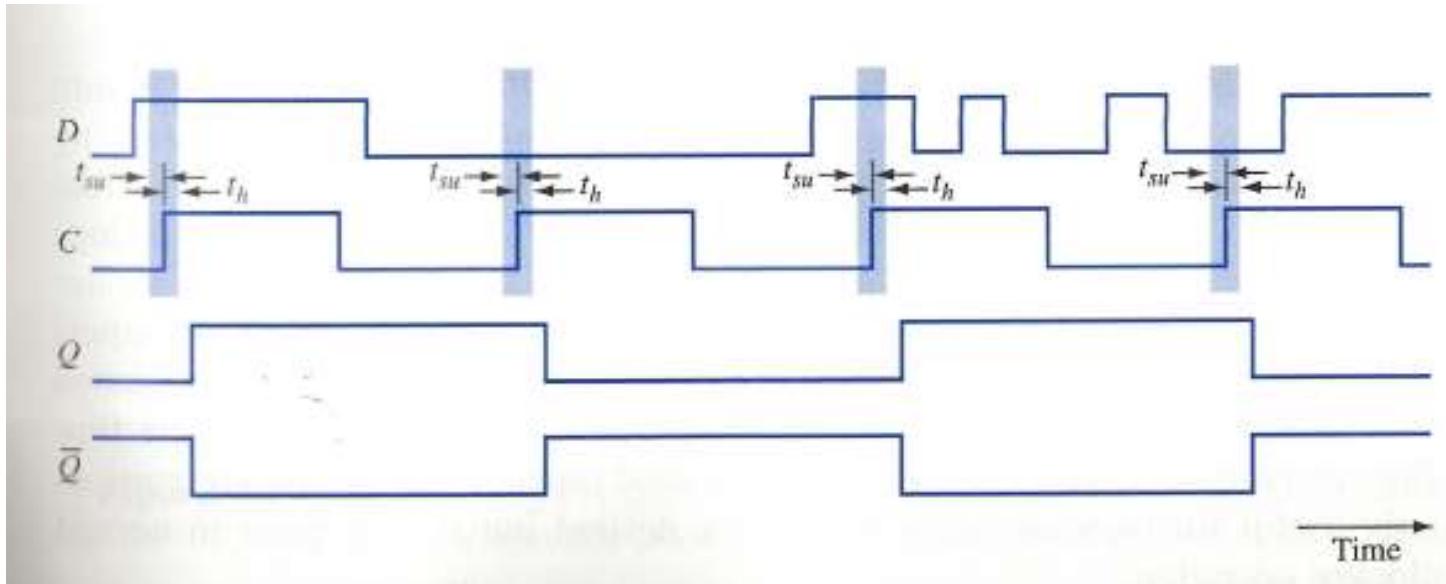
(b)



(c)

**Figure 6.18** Positive-edge-triggered  $D$  flip-flop. (a) Logic diagram. (b) Function table where  $Q^+$  denotes the output  $Q$  in response to the inputs. (c) Two logic symbols.

# Timing Diagram



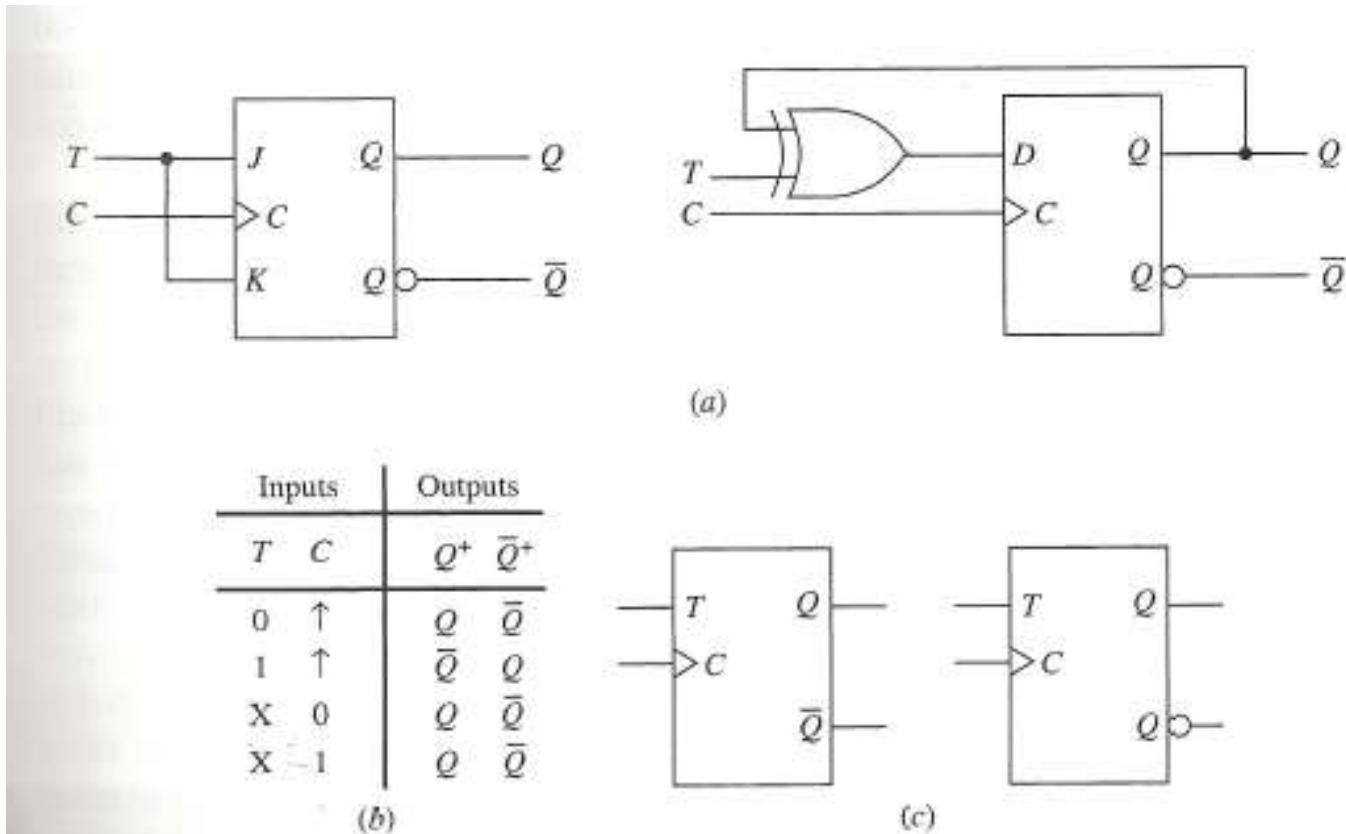
**Figure 6.19** Timing diagram for a positive-edge-triggered  $D$  flip-flop.

During setup and hold times  $t_{su}, t_h$  with respect to the triggering edge of the clock, D input must not change.

# Negative-Edge Triggered D Flip-Flop

- A falling edge (high to low transition) of control signal is used to sample the D input line.
- Simply place inverter at the control input of the flip-flop.

# Positive-Edge Triggered T-Flip-Flop



**Figure 6.23** Positive-edge-triggered  $T$  flip-flop. (a) Logic diagrams. (b) Function table where  $Q^+$  denotes the output  $Q$  in response to the inputs. (c) Two logic symbols.

# Characteristic Equations

- Next state table: Shows the value of the next state of the flip-flop for each combination of values to the present state of the flip-flops and their information lines.
- The algebraic description of the next-state table of a flip-flop is called the characteristic equation of the flip-flop.
- Obtained by constructing the K-map for  $Q^+$  in terms of the present state and information input variables.

# Next State Tables

**Table 6.1** Simplified flip-flop function tables.  $Q$  denotes the current state and  $Q^+$  denotes the resulting state as a consequence of the information inputs and the control signal.  
 (a) SR flip-flop. (b) D flip-flop.  
 (c) JK flip-flop. (d) T flip-flop.

S	R	$Q^+$	D	$Q^+$
0	0	$Q$	0	0
0	1	0	1	1
1	0	1		
1	1	—		

(a)

(b)

J	K	$Q^+$	T	$Q^+$
0	0	$Q$	0	$Q$
0	1	0	1	$\bar{Q}$
1	0	1		
1	1	$\bar{Q}$		

(c)

(d)

**Table 6.2** Flip-flop next-state tables.  $Q$  denotes the current state and  $Q^+$  denotes the resulting state as a consequence of the information inputs and the control signal. (a) SR flip-flop. (b) D flip-flop. (c) JK flip-flop. (d) T flip-flop.

S	R	$Q$	$Q^+$	D	$Q$	$Q^+$
0	0	0	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	1
0	1	1	0	1	1	1
1	0	0	1			
1	0	1	1			
1	1	0	—	} Inputs not allowed		
1	1	1	—	} Inputs not allowed		

(a)

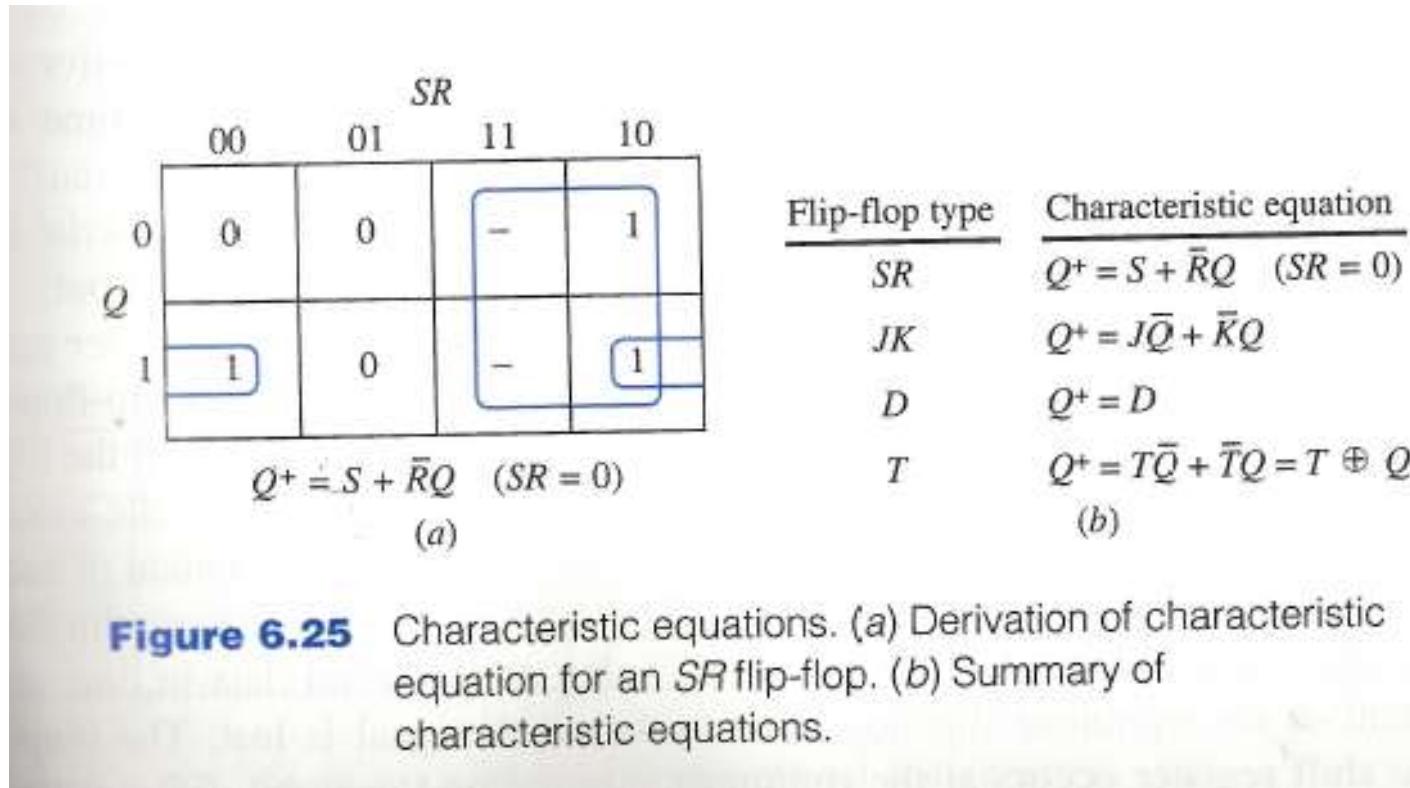
(b)

J	K	$Q$	$Q^+$	T	$Q$	$Q^+$
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	1			
1	0	1	1			
1	1	0	1			
1	1	1	0			

(c)

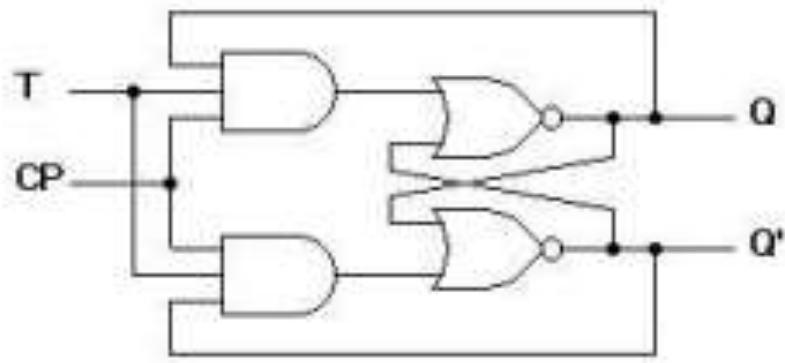
(d)

# Characteristic Equations

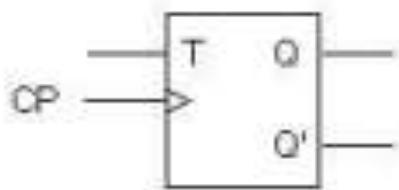


# T Flip Flop

- This is a much simpler version of the J-K flip flop. Both the J and K inputs are connected together and thus are also called a single input J-K flip flop.
- When clock pulse is given to the flip flop, the output begins to toggle. Here also the restriction on the pulse width can be eliminated with a masterslave or edge-triggered construction.



(a) Logic diagram



(b) Graphical symbol

Q	T	$Q[t+1]$
0	0	0
0	1	1
1	0	1
1	1	0

(c) Transition table

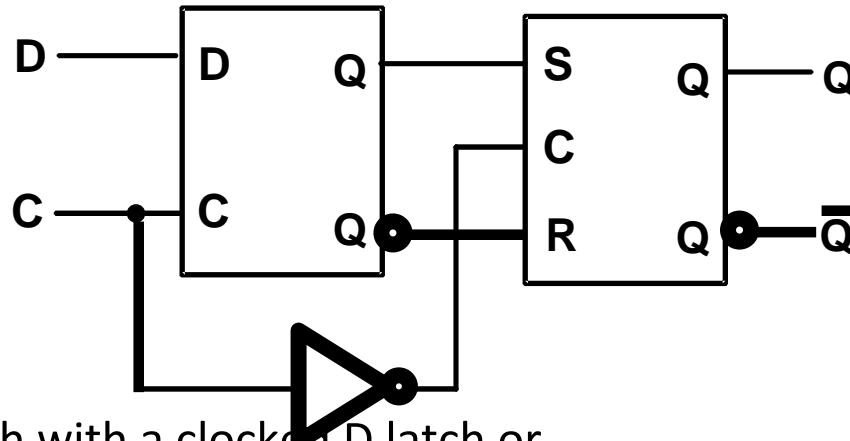
Clocked T flip-flop

# Flip-Flop Solution

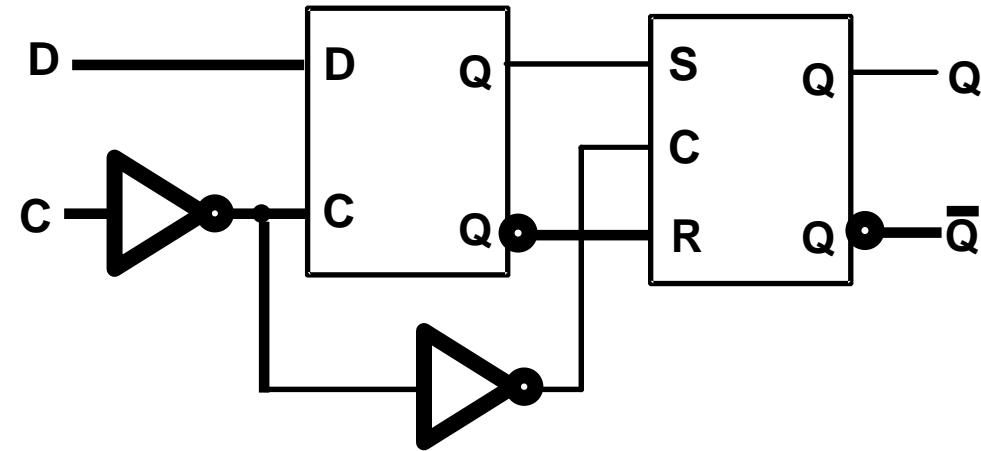
- Use edge-triggering instead of master-slave
- An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a transition of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A master-slave D flip-flop which also exhibits edge-triggered behavior can be used.

# Edge-Triggered D Flip-Flop

- The edge-triggered D flip-flop is the same as the master-slave D flip-flop
- It can be formed by:
  - Replacing the first clocked S-R latch with a clocked D latch or
  - Adding a D input and inverter to a master-slave S-R flip-flop
- The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs
- The change of the D flip-flop output is associated with the negative edge at the end of the pulse
- It is called a *negative-edge triggered* flip-flop

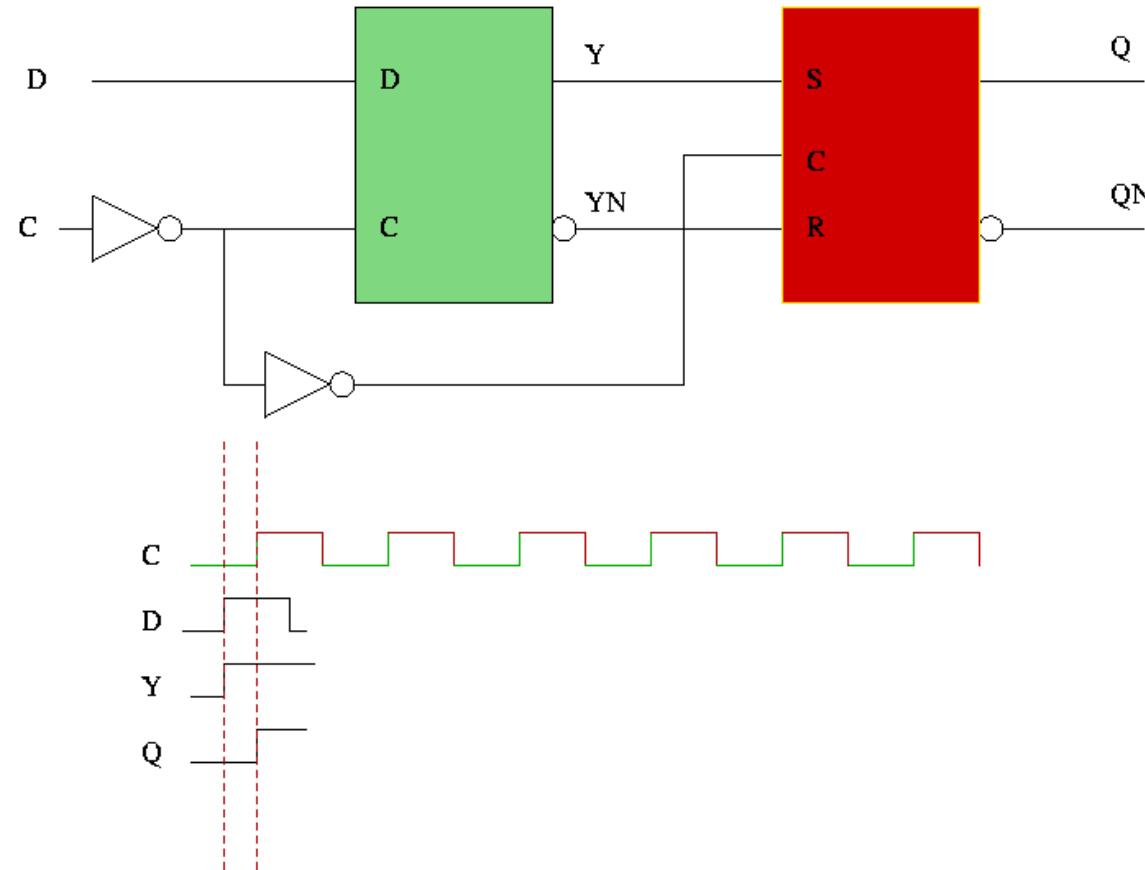


# Positive-Edge Triggered D Flip-Flop

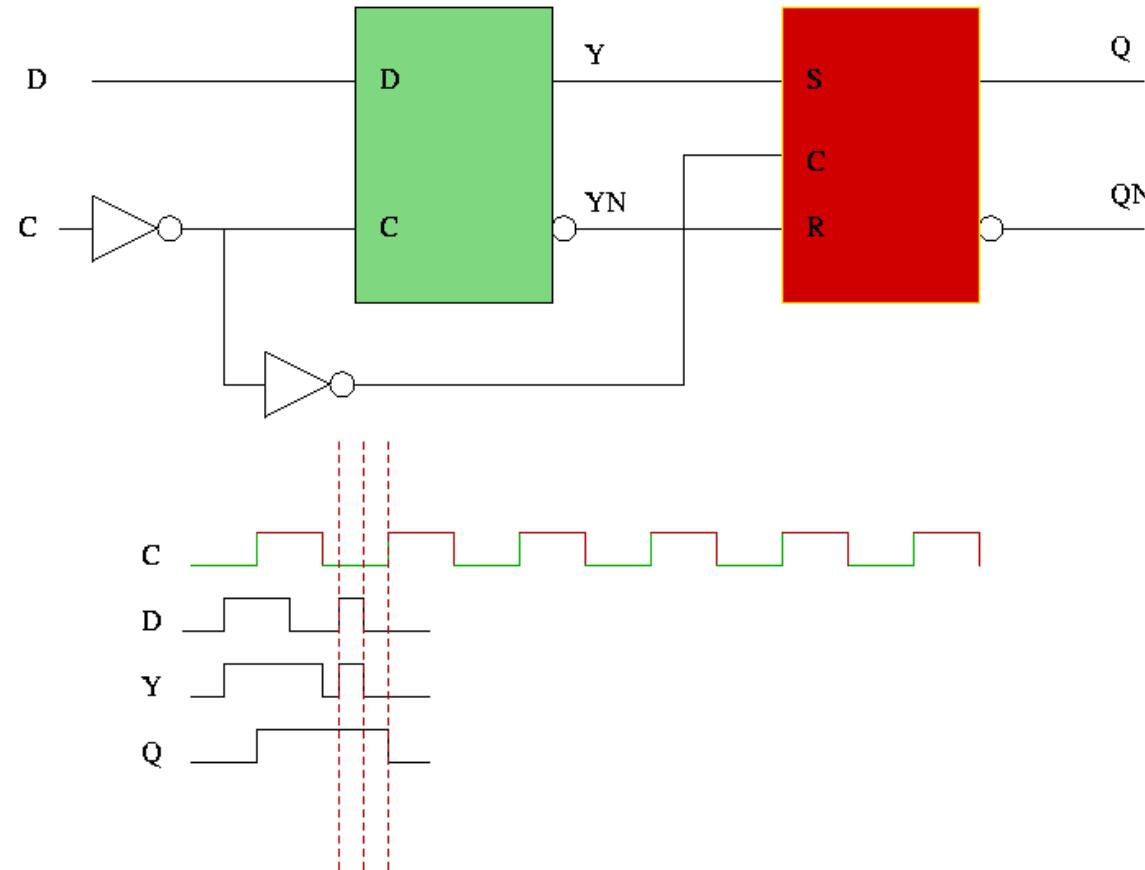


- Formed by adding inverter to clock input
- Q changes to the value on D applied at the positive clock edge within timing constraints to be specified
- Our choice as the standard flip-flop for most sequential circuits

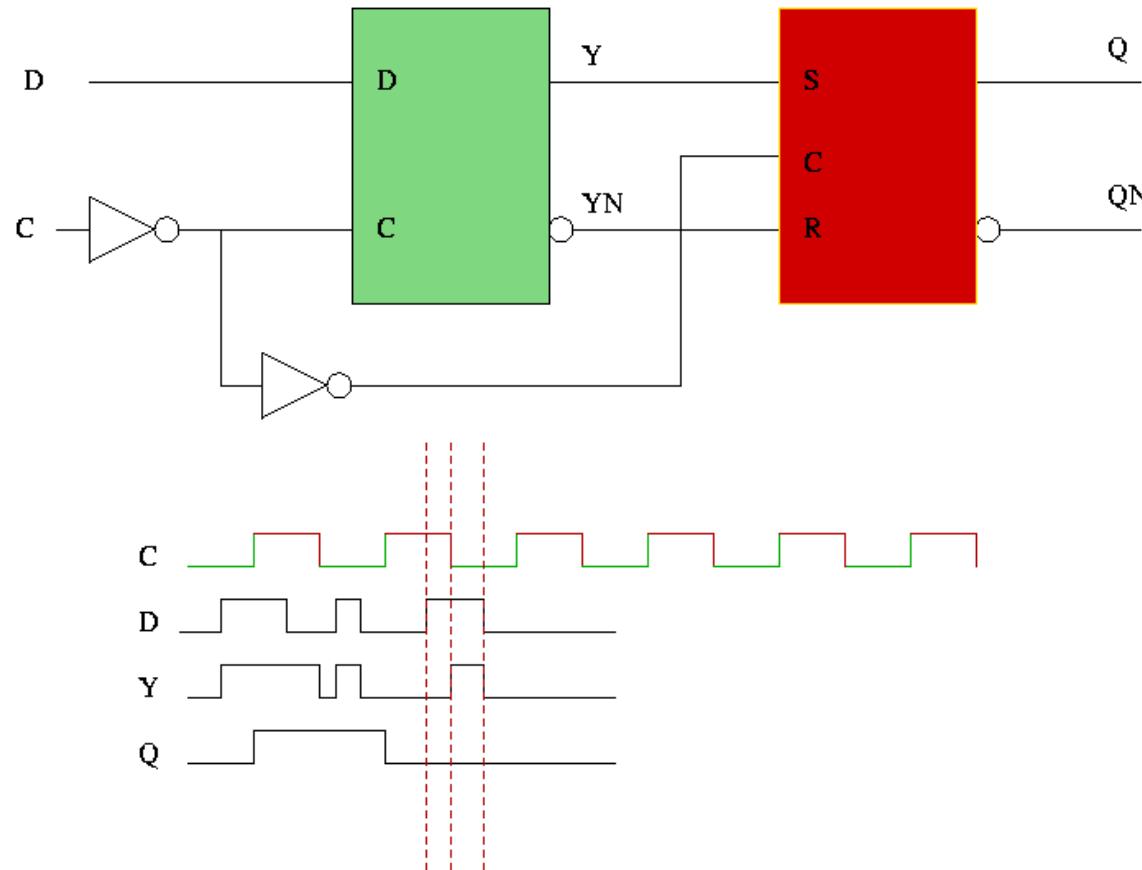
# Edge-Triggered FF Operation



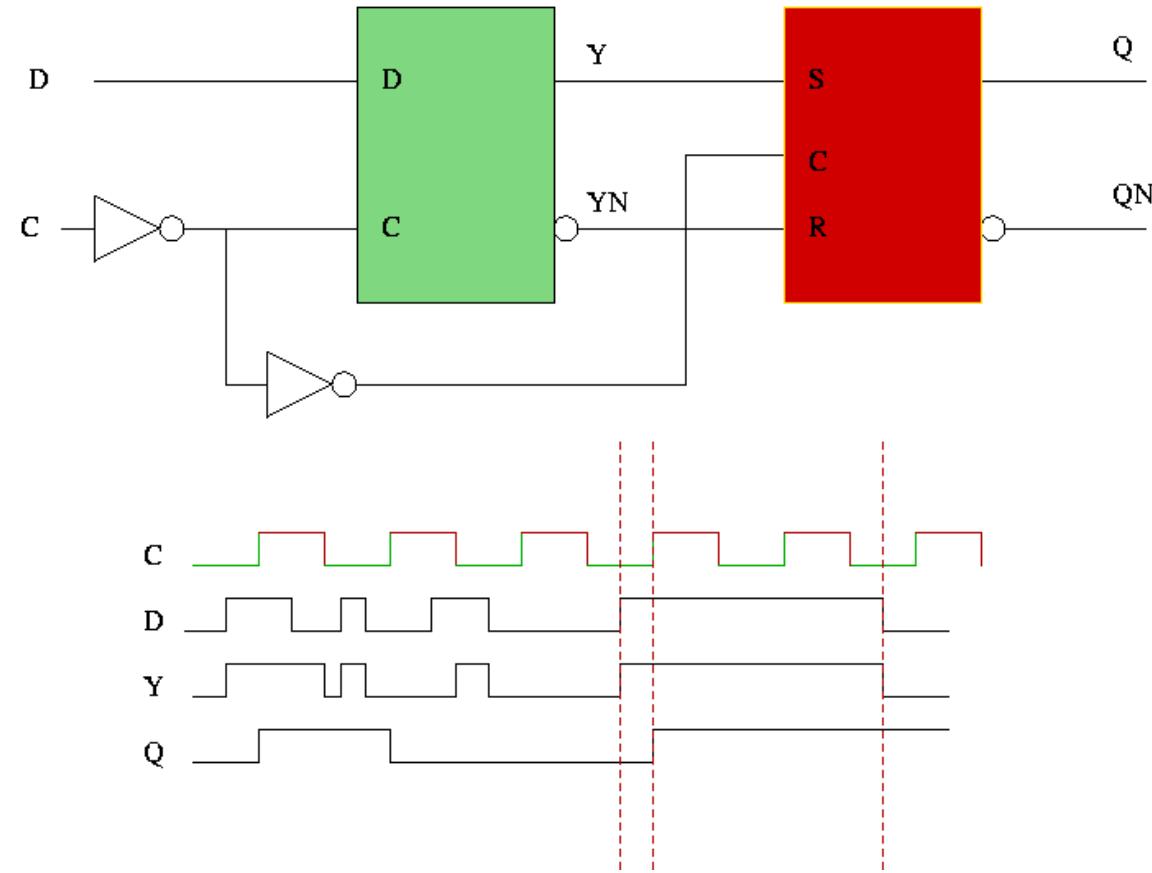
# Edge-Triggered FF Operation



# Edge-Triggered FF Operation



# Edge-Triggered FF Operation



**B.Tech – CSE – Sem. 3**

**18CSS201J – ANALOG AND DIGITAL ELECTRONICS**

**(Regulations 2018)**

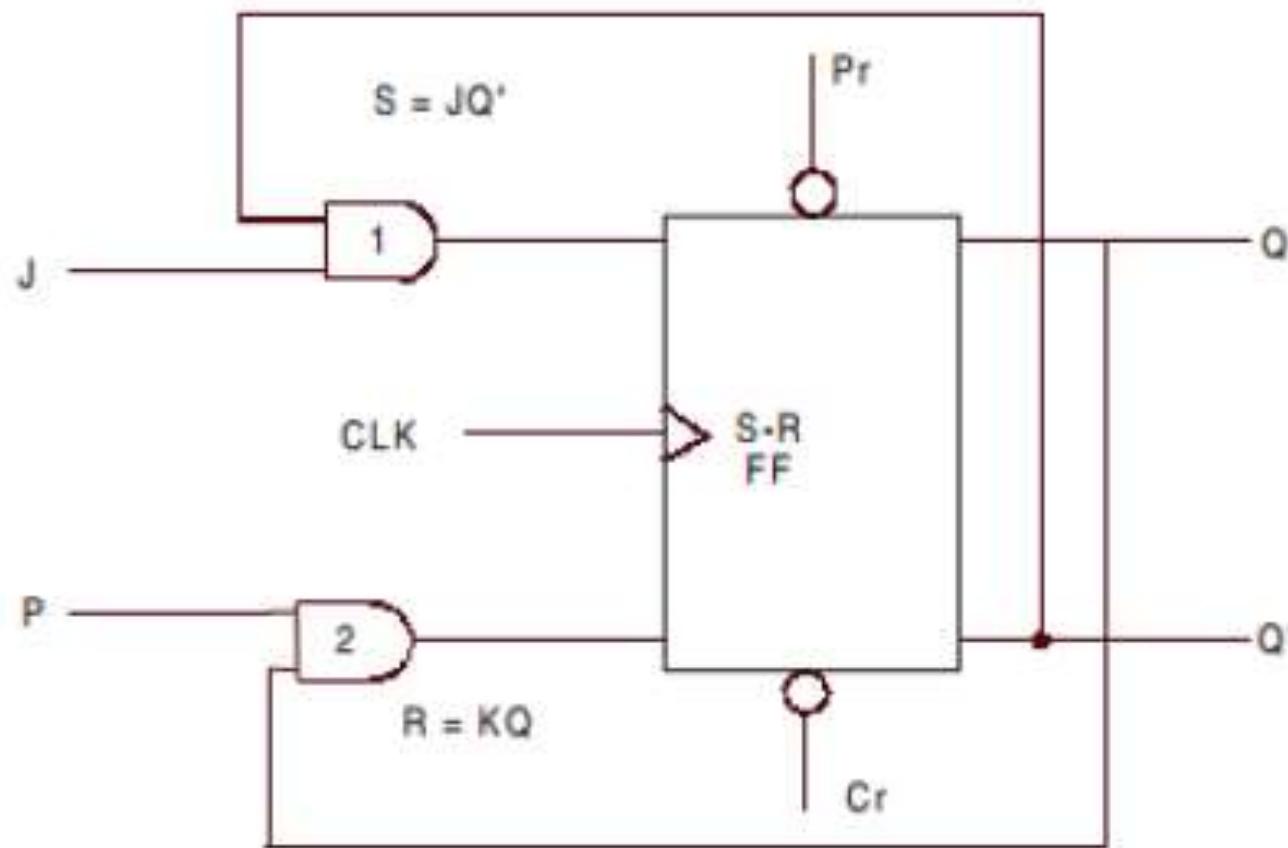
**UNIT 4**

**Session-6(Edge Triggered J-K Flip-flop, Edge Triggered Master  
Slave Flip-flop)**

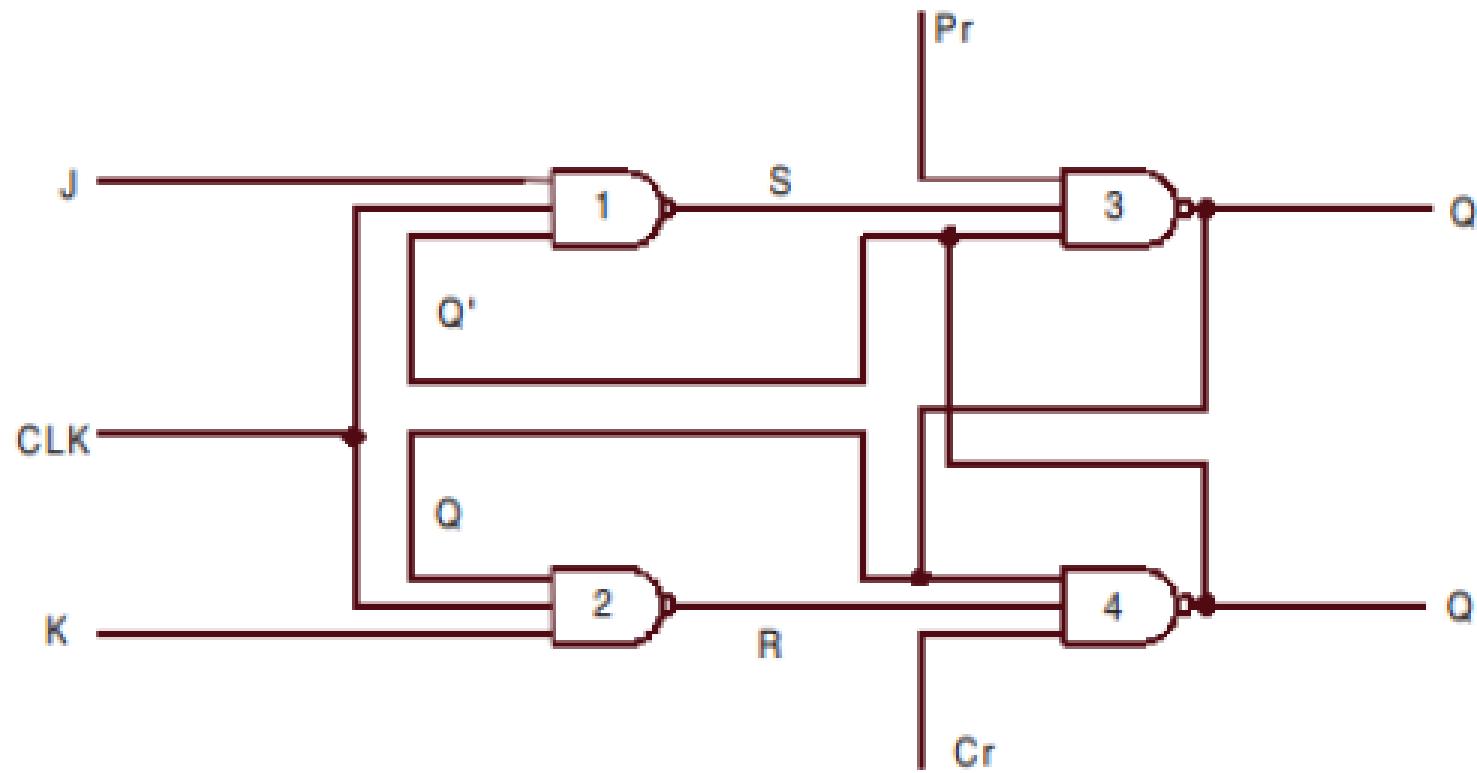
# J-K FLIP-FLOP

- A J-K flip-flop has very similar characteristics to an S-R flip-flop. The only difference is that the undefined condition for an S-R flip-flop, *i.e.*,  $S_n = R_n = 1$  condition, is also included in this case. Inputs J and K behave
- like inputs S and R to set and reset the flip-flop respectively. When  $J = K = 1$ , the flip-flop is said to be in a
- ***toggle state***, which means the output switches to its complementary state every time a clock passes.
- The data inputs are J and K, which are ANDed with  $Q'$  and Q respectively to obtain the inputs for S and R

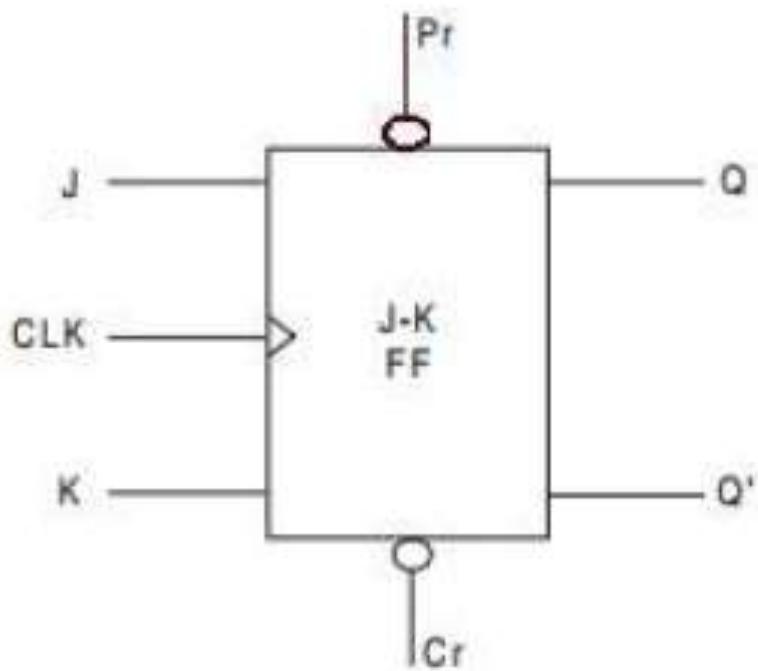
# An S-R flip-flop converted into a J-K flip-flop



# A J-K flip-flop using NAND gates



# Logic symbol of a J-K flip-flop



# The TRUTH table for JK flip-flop

<i>Data inputs</i>		<i>Outputs</i>		<i>Inputs to S-R FF</i>		<i>Output</i>
$J_n$	$K_n$	$Q_n$	$Q'_n$	$S_n$	$R_n$	$Q_{n+1}$
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	0	0	1	0
1	0	0	1	1	0	1
1	0	1	0	0	0	1
1	1	0	1	1	0	1
1	1	1	0	0	1	0

Inputs		Output
$J_n$	$K_n$	$Q_{n+1}$
0	0	$Q_n$
0	1	0
1	0	1
1	1	$Q'_n$

**Case 1.** When the clock is applied and  $J = 0$ , whatever the value of  $Q'n$  (0 or 1), the output of NAND gate 1 is 1.

Similarly, when  $K = 0$ , whatever the value of  $Q_n$  (0 or 1), the output of gate 2 is also 1. Therefore, when  $J = 0$  and  $K = 0$ , the inputs to the basic flip-flop are  $S = 1$  and  $R = 1$ . This condition forces the flip-flop to remain in the same state.

**Case 2.** When the clock is applied and  $J = 0$  and  $K = 1$  & the previous state of the flip-flop is reset (i.e.,  $Q_n = 0$  and  $Q'n = 1$ ), then  $S = 1$  and  $R = 1$ . Since  $S = 1$  and  $R = 1$ , the basic flip-flop does not alter the state and remains in the reset state. But if the flip-flop is in set condition (i.e.,  $Q_n = 1$  &  $Q'n = 0$ ), then  $S = 1$  and  $R = 0$ . Since  $S = 1$  and  $R = 0$ , the basic flip-flop changes its state and resets.

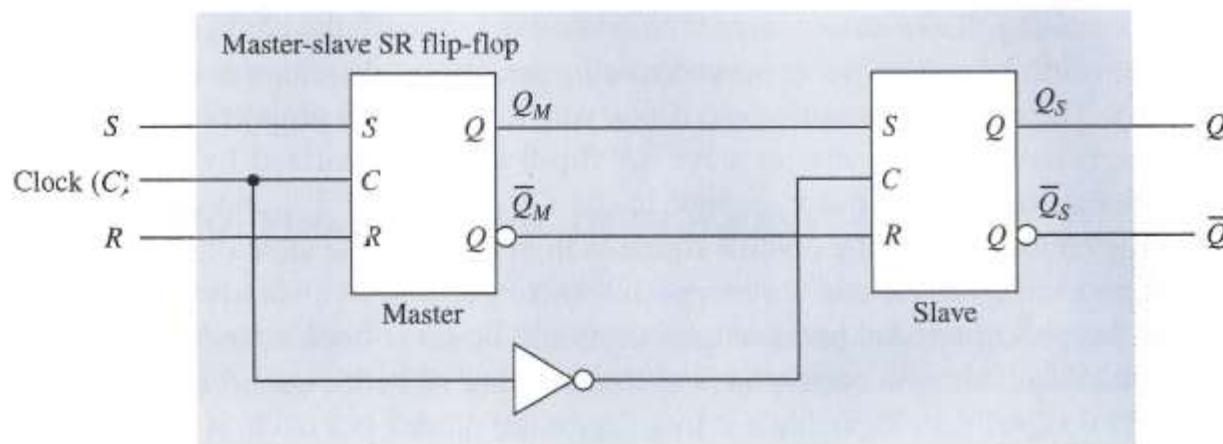
- **Case 3.** When the clock is applied and  $J = 1$  and  $K = 0$  and the previous state of the flip-flop is reset (*i.e.*,  $Q_n = 0$  and  $Q'n = 1$ ), then  $S = 0$  and  $R = 1$ . Since  $S = 0$  and  $R = 1$ , the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (*i.e.*,  $Q_n = 1$  and  $Q'n = 0$ ), then  $S = 1$  and  $R = 1$ . Since  $S = 1$  and  $R = 1$ , the basic flip-flop does not alter its state and remains in the set state.
- **Case 4.** When the clock is applied and  $J = 1$  and  $K = 1$  and the previous state of the flip-flop is reset (*i.e.*,  $Q_n = 0$  and  $Q'n = 1$ ), then  $S = 0$  and  $R = 1$ . Since  $S = 0$  and  $R = 1$ , the basic flip-flop changes its state and goes to the set state. But if the flip-flop is already in set condition (*i.e.*,  $Q_n = 1$  and  $Q'n = 0$ ), then  $S = 1$  and  $R = 0$ . Since  $S = 1$  and  $R = 0$ , the basic flip-flop changes its state and goes to the reset state. So we find that for  $J = 1$  and  $K = 1$ , the flip-flop toggles its state from *set* to *reset* and vice versa. Toggle means to switch to the opposite state.

# Master-Slave Flip-Flops (Pulse Triggered Flip-Flops)

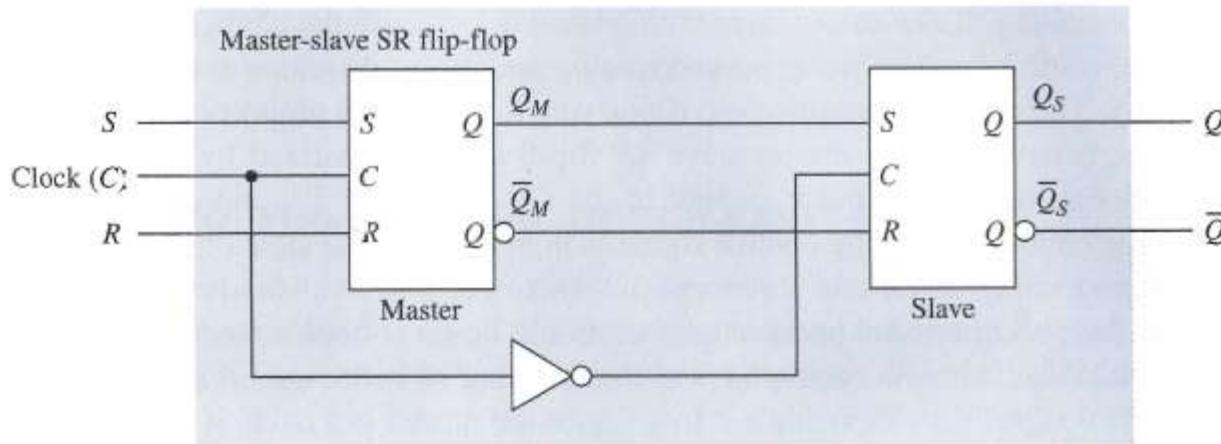
- Aside from latches, two categories of flip-flops.
  - Master-slave flip-flops (pulse-triggered flip-flops)
  - Edge-triggered flip-flops
- Latches have immediate output response (known as transparency)
- May be undesirable:
  - May be necessary to sense the current state of a flip-flop while allowing new state information to be entered.

# Master-Slave SR Flip-Flop

- Two sections, each capable of storing a binary symbol.
- First section is referred to as the master and the second section as the slave.
- Information is entered into the master on one edge or level of a control signal and is transferred to the slave on the next edge or level of the control signal.
- Each section is a latch.

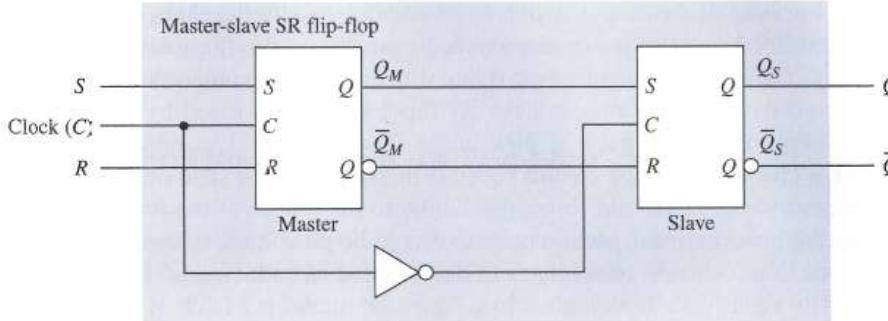


# Master-Slave SR Flip-Flop

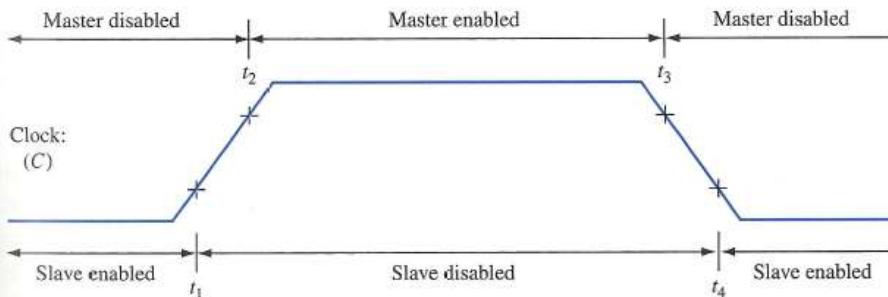


- C = 0:
  - Master is disabled. Any changes to S,R ignored.
  - Slave is enabled. Is in the same state as the master.
- C = 1:
  - Slave is disabled (retains state of master)
  - Master is enabled, responds to inputs. Changes in state of master are not reflected in disabled slave.
- C = 0:
  - Master is disabled.
  - Slave is enabled and takes on new state of the master.
- Important: For short periods during rising and falling edges, both master and slave are disabled.

# Master-Slave SR Flip-Flop



(a)



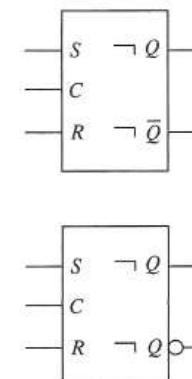
Slave only takes on state of the master at  $t_4$ .

Postponed output indicator: output change postponed until end of pulse

Pulse symbol indicates master enabled when  $C = 1$  and state of master transferred to slave at the end of the pulse period.

Inputs			Outputs	
S	R	C	$Q^+$	$\bar{Q}^+$
0		0	Q	$\bar{Q}$
1		0	0	1
1	0	0	1	0
1	1	0	Undefined	Undefined
X	X	0	Q	$\bar{Q}$

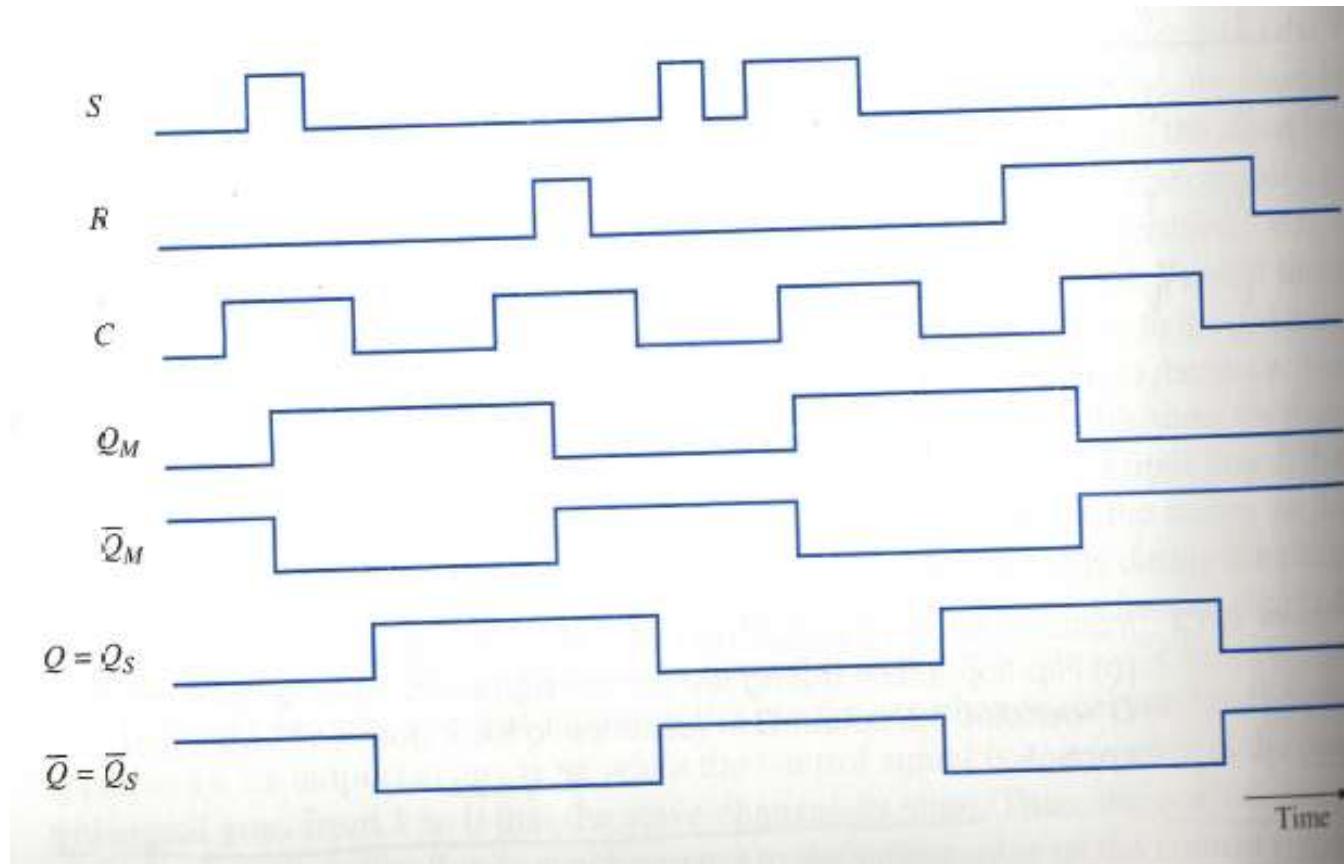
(c)



(d)

If  $S, R = 1$  when control signal goes from high to low we are in an unpredictable state. Can cause metastable state.

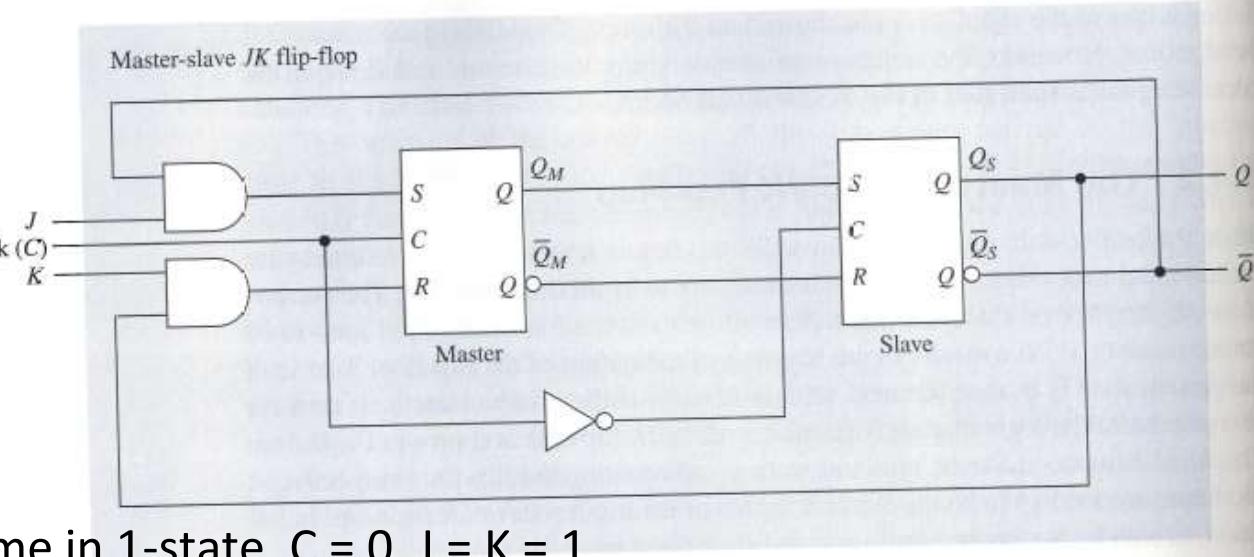
# Timing Diagram for Master-Slave SR flip-flop



# Master-Slave JK Flip-Flop

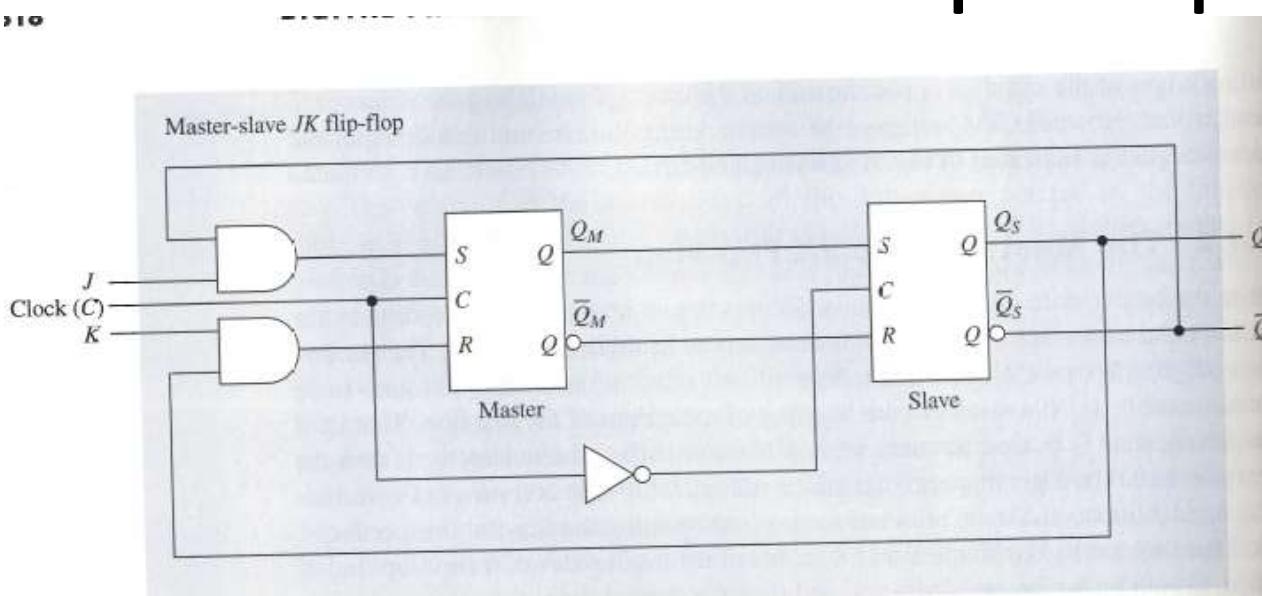
- The output state of a master-slave SR flip-flop is undefined upon returning the control input to 0 when  $S = R = 1$ .
  - Necessary to avoid this condition.
- Master-slave JK flip-flop allows its two information input lines to be simultaneously 1.
  - Results in toggling the output of the flip flop.

# Master-Slave JK Flip-Flop



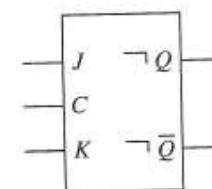
- Assume in 1-state, C = 0, J = K = 1.
  - Due to feedback, the output of the J-gate is 0, output of K-gate is 1.
  - If clock is changed to C = 1 then master is reset.
- Assume in 0-state, C = 0, J = K = 1.
  - Due to feedback, the output of the J-gate is 1, output of K-gate is 0.
  - If clock is changed to C = 1 then master is set.
- 1 on J input line, 0 on K input line sets the flip-flop.
  - If in 1-state, unchanged b/c S,R set to 0.
  - If in 0-state, S set to 1, R set to 0.
- 0 on J input, 1 on K input line resets the flip-flop. Why?

# Master-Slave JK Flip-Flop

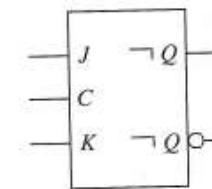


Inputs			Outputs	
<i>J</i>	<i>K</i>	<i>C</i>	$Q^+$	$\bar{Q}^+$
0	0	High	$Q$	$\bar{Q}$
0	1	High	0	1
1	0	High	1	0
1	1	High	$\bar{Q}$	$Q$
X	X	0	$Q$	$\bar{Q}$

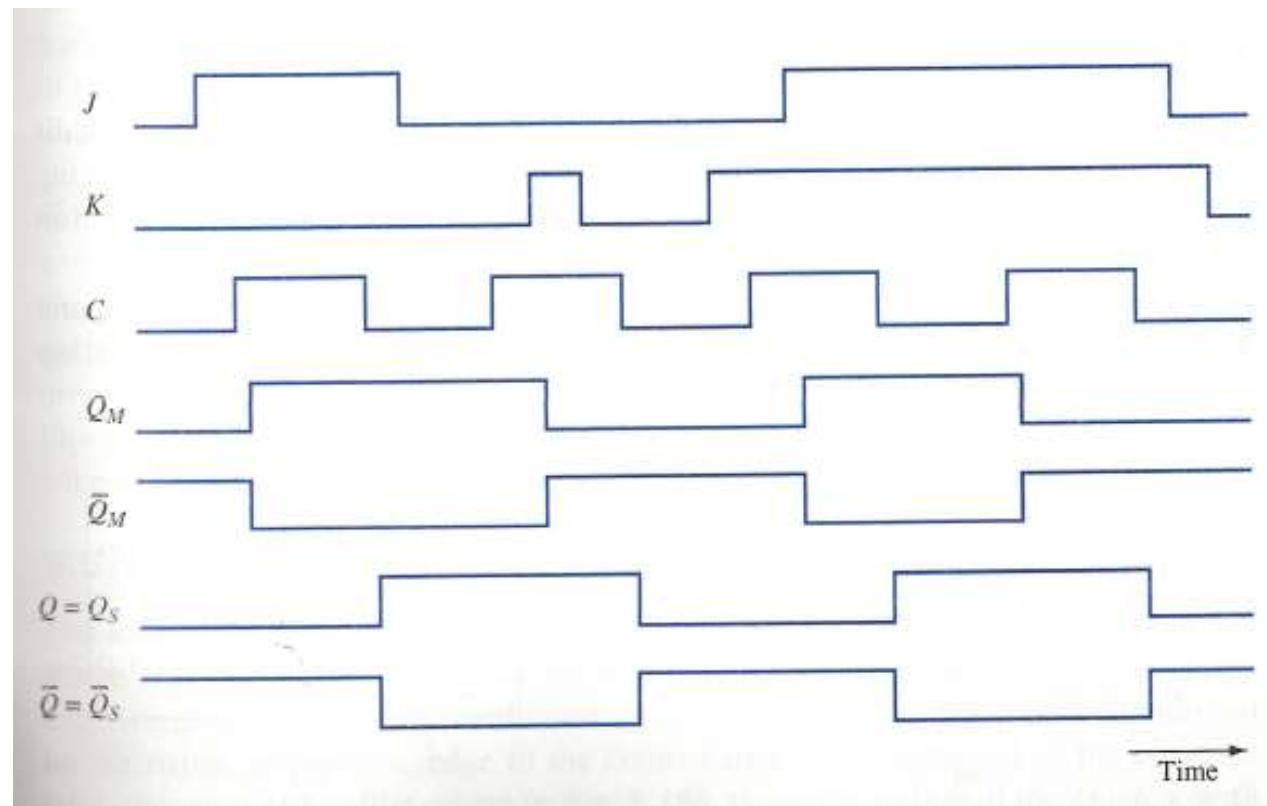
(b)



(c)



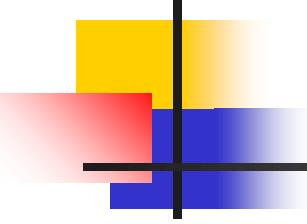
# Timing Diagram for Master-Slave JK Flip-Flop



B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

UNIT 4

Session-7( Analysis of synchronous sequential circuits: State Equation,  
State Table, State Diagram)



## Analysis of Clocked Sequential Circuits

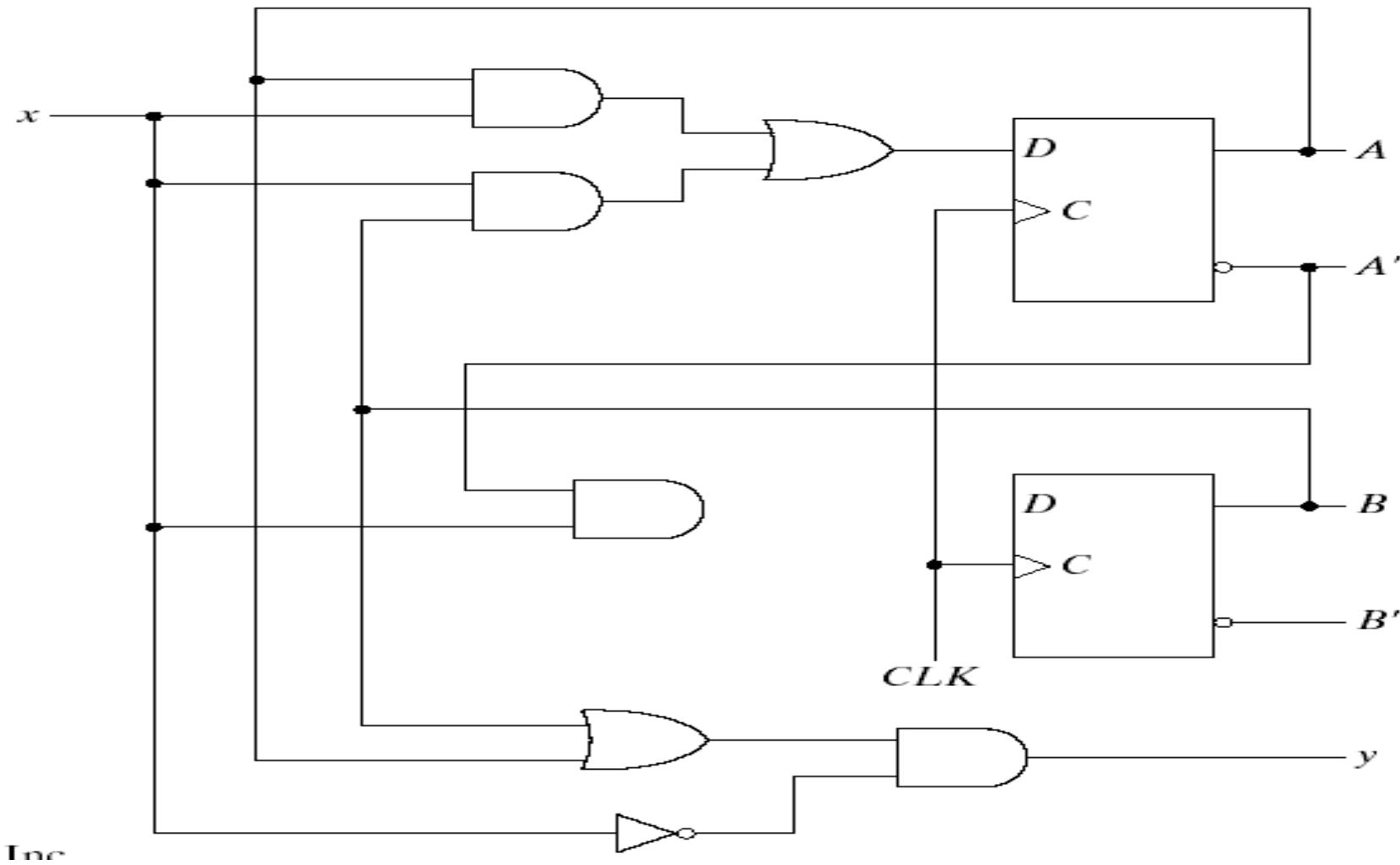
---

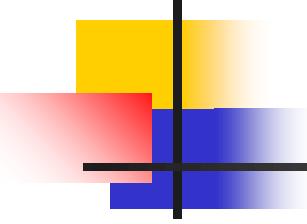
The analysis of a sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states. It is also possible to write Boolean expressions that describe the behavior of the sequential circuit. These expressions must include the necessary time sequence, either directly or indirectly.

# State Equations

The behavior of a clocked sequential circuit can be described algebraically by means of state equations. A state equation specifies the next state as a function of the present state and inputs. Consider the sequential circuit. It consists of two D flip-flops A and B, an input x and an output y.

## Fig.5-15 Example of Sequential Circuit





# State Equation

---

$$A(t+1) = A(t) \cdot x(t) + B(t) \cdot \bar{x}(t)$$

$$B(t+1) = A'(t) \cdot x(t)$$

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation with  $(t+1)$  denotes the next state of the flip-flop one clock edge later. The right side of the equation is Boolean expression that specifies the present state and input conditions that make the next state equal to 1.

$$Y(t) = (A(t) + B(t)) \cdot x(t)'$$

# State Table

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a **state table** (sometimes called **transition table**).

**Table 5-2**  
*State Table for the Circuit of Fig. 5-15*

Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

**Table 5-3**  
*Second Form of the State Table*

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

# State Diagram

The information available in a state table can be represented graphically in the form of a **state diagram**. In this type of diagram, a state is represented by a circle, and the transitions between states are indicated by directed lines connecting the circles.

1/0 : means input =1  
output=0

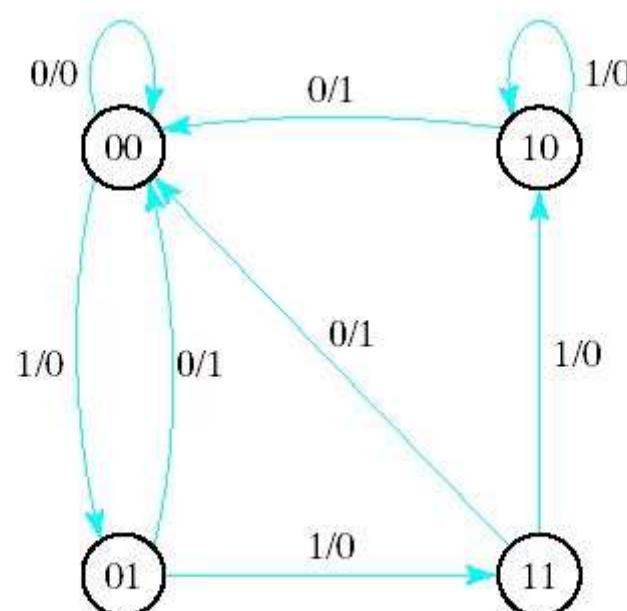
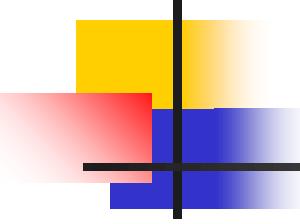


Fig. 5-16 State Diagram of the Circuit of Fig. 5-15



## Flip-Flop Input Equations

---

The part of the combinational circuit that generates external outputs is described algebraically by a set of Boolean functions called **output equations**. The part of the circuit that generates the inputs to flip-flops is described algebraically by a set of Boolean functions called **flip-flop input equations**. The sequential circuit consists of two D flip-flops A and B, an input x, and an output y. The logic diagram of the circuit can be expressed algebraically with **two flip-flop input equations** and **an output equation**:

$$D_A = Ax + Bx$$

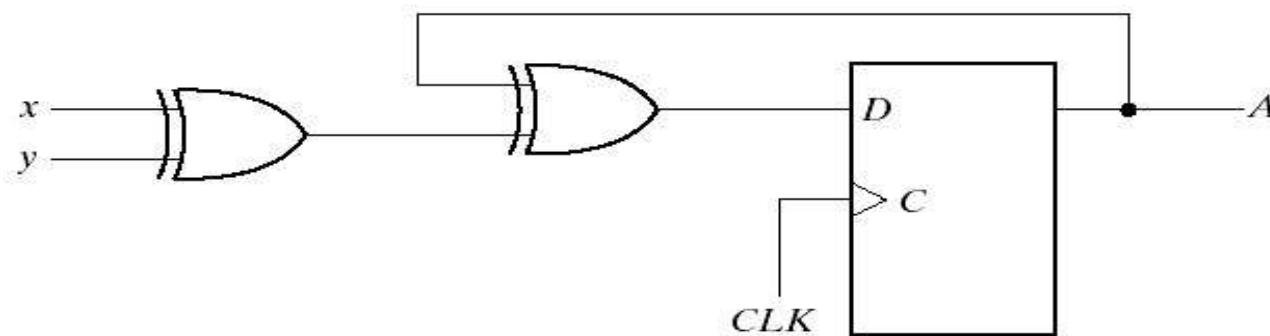
$$D_B = A'x$$

$$y = (A + B)x'$$

# Analysis with D Flip-Flop

The circuit we want to analyze is described by the input equation  $D_A = A \oplus x \oplus y$

The  $D_A$  symbol implies a D flip-flop with output A. The x and y variables are the inputs to the circuit. No output equations are given, so the output is implied to come from the output of the flip-flop.



(a) Circuit diagram

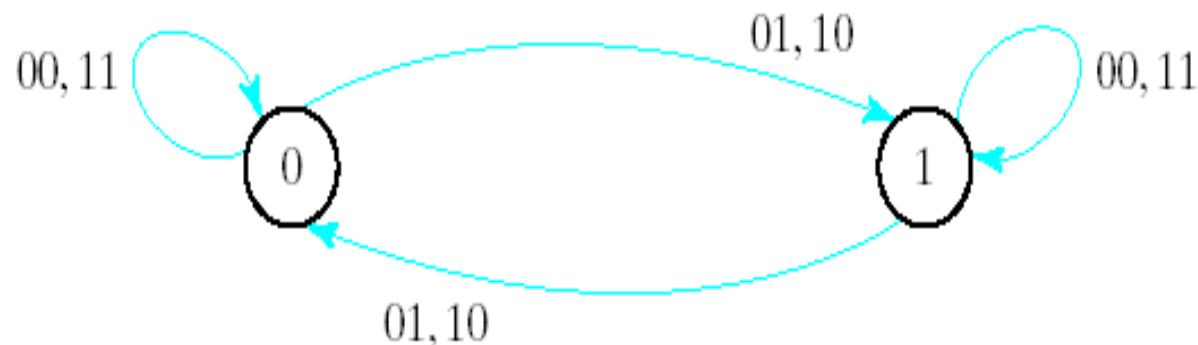
# Analysis with D Flip-Flop

The binary numbers under A<sub>xy</sub> are listed from 000 through 111 as shown in Fig. 5-17(b). The next state values are obtained from the state equation  $A(t+1) = A \oplus x \oplus y$

The state diagram consists of two circles-one for each state as shown in Fig. 5-17(c)

Present state	Inputs		Next state
<i>A</i>	<i>x</i>	<i>y</i>	<i>A</i>
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table



(c) State diagram

# Analysis with JK Flip-Flops

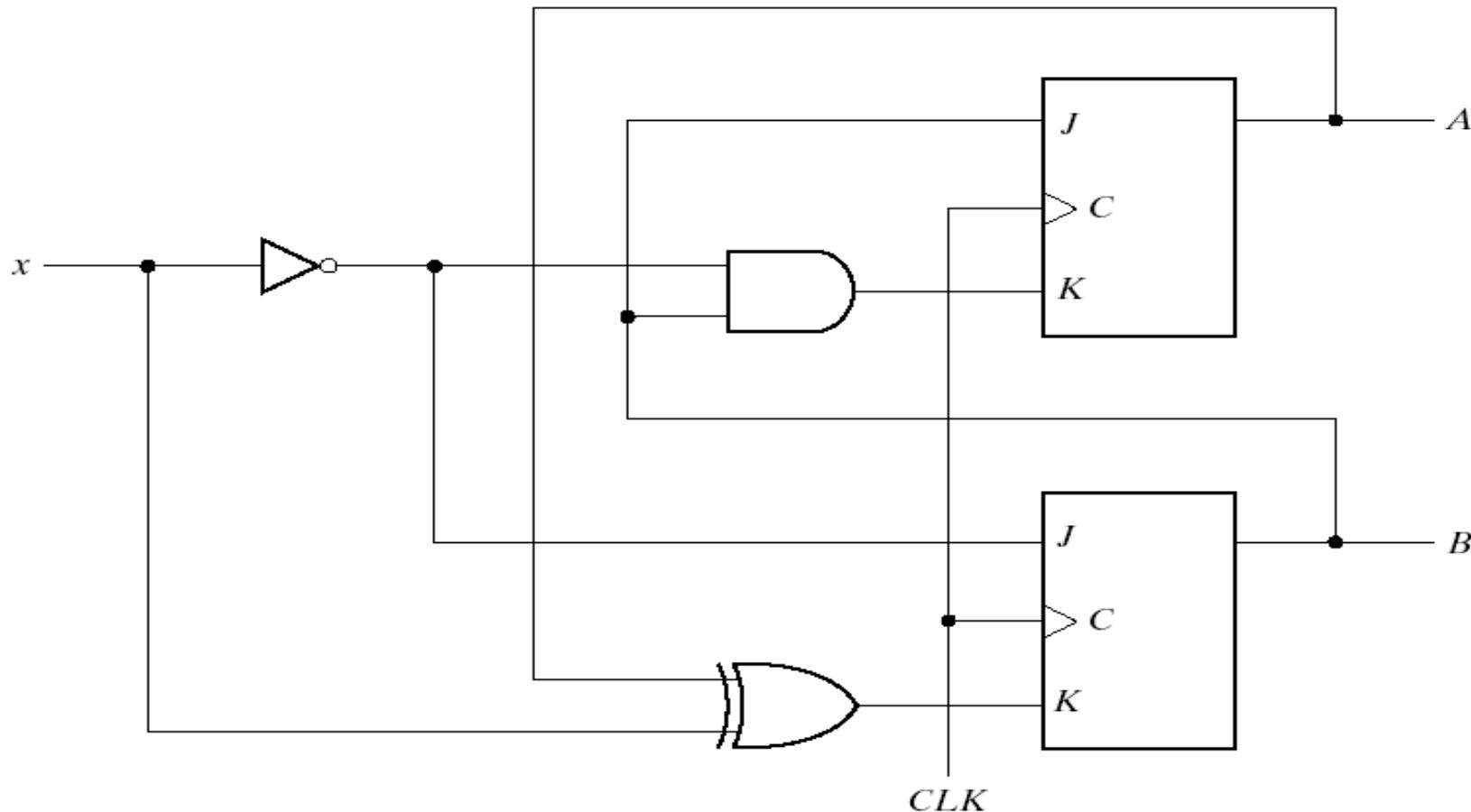


Fig. 5-18 Sequential Circuit with  $JK$  Flip-Flop

# Analysis with JK Flip-Flop

The circuit can be specified by the flip-flop input equations

$$J_A = B$$

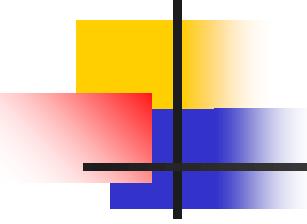
$$K_A = Bx'$$

$$J_B = x'$$

$$K_B = A'x + Ax' = A \oplus x$$

**Table 5-4**  
*State Table for Sequential Circuit with JK Flip-Flops*

Present State		Input x	Next State A    B		Flip-Flop Inputs J <sub>A</sub> K <sub>A</sub> J <sub>B</sub> K <sub>B</sub>			
A	B		A	B	J <sub>A</sub>	K <sub>A</sub>	J <sub>B</sub>	K <sub>B</sub>
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0



## Analysis with JK Flip-Flops

$$\begin{aligned}A(t + 1) &= JA' + K'A \\B(t + 1) &= JB' + K'B\end{aligned}$$

Substituting the values of JA and KA from the input equations, we obtain the state equation for A:

$$A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$$

The state equation provides the bit values for the column under next state of A in the state table. Similarly, the state equation for flip-flop B can be derived from the characteristic equation by substituting the values of JB and KB:

$$B(t + 1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

# Analysis with JK Flip-Flops

The state diagram of the sequential circuit is shown in Fig. 5-19.

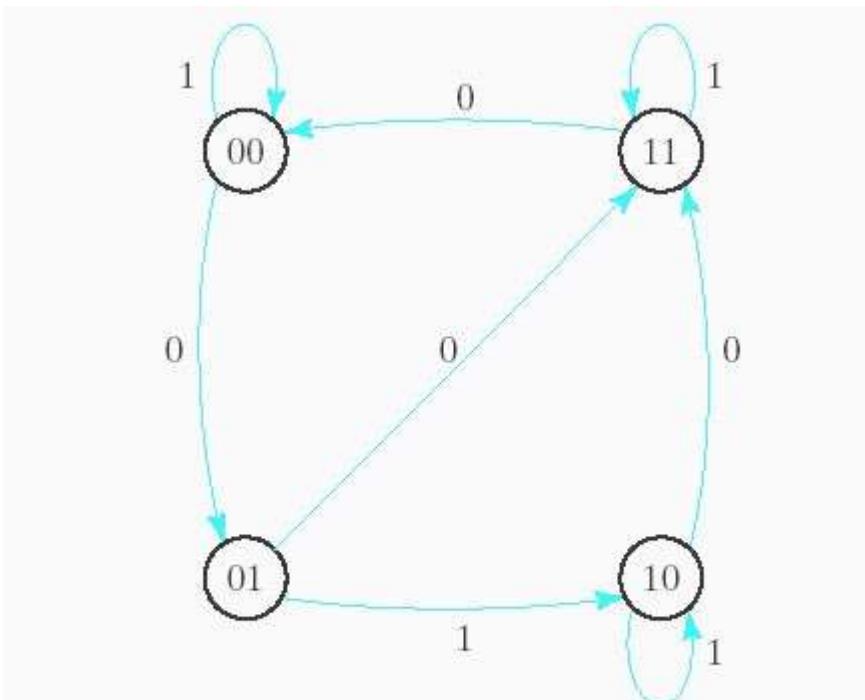
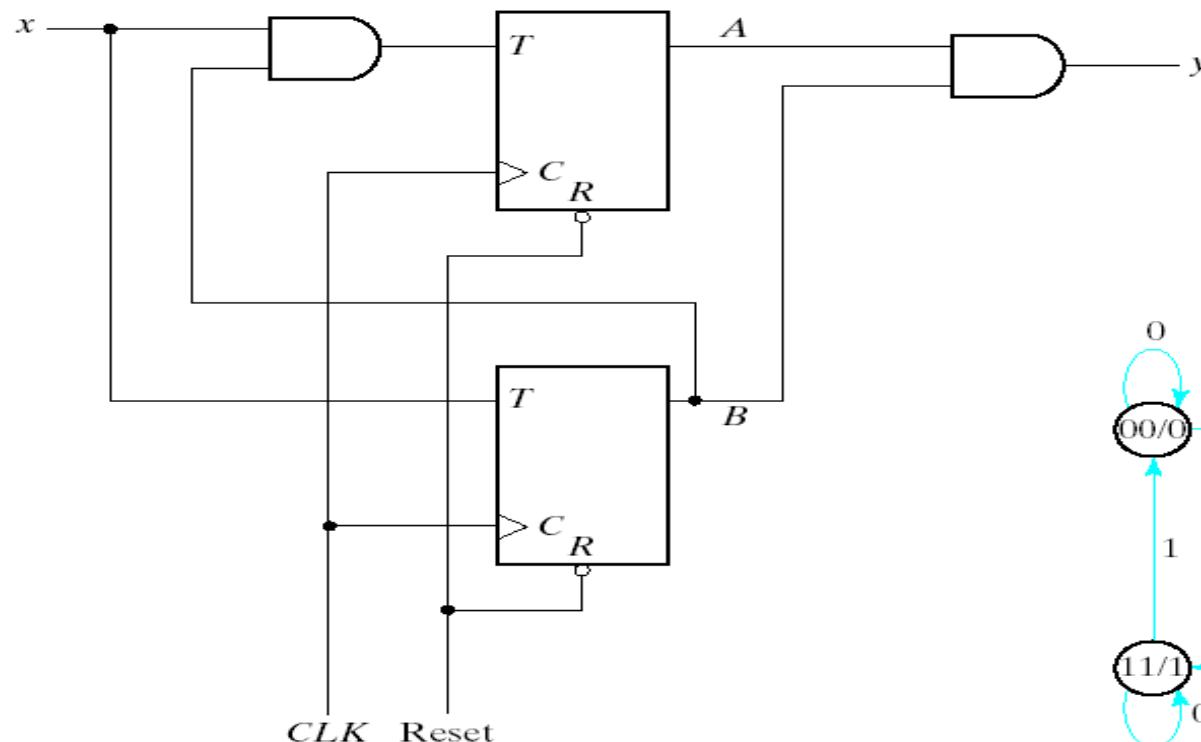


Fig. 5-19 State Diagram of the Circuit of Fig. 5-18

# Analysis With T Flip-Flops

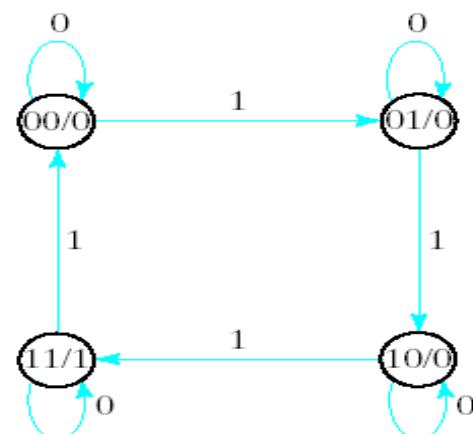
Characteristic equation

$$Q(t + 1) = T \oplus Q = T'Q + TQ'$$



(a) Circuit diagram

00/0 : means  
state is 00  
output is 0



(b) State diagram

# Analysis With T Flip-Flops

Consider the sequential circuit shown in Fig. 5-20. It has two flip-flops A and B, one input x, and one output y. It can be described algebraically by two input equations and an output equation:

$$T_A = Bx$$

$$T_B = x$$

$$y = AB$$

$$\begin{aligned}A(t+1) &= (Bx)'A + (Bx)A' \\&= AB' + Ax' + A'Bx\end{aligned}$$

$$B(t+1) = x \oplus B$$

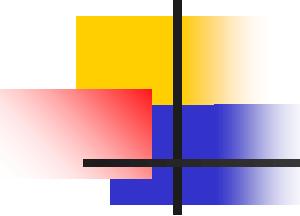
**Use present state as inputs**

Present State		Input x	Next State		Output y
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

UNIT 4

Session-8 ( Synthesis of Sequential Circuits, Problem Session)



## Synthesis Sequential Circuits: Design Procedure

---

The procedure for designing synchronous sequential circuits can be summarized by a list of recommended steps.

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary-coded state table.
5. Choose the type of flip-flops to be used.
6. Derive the simplified flip-flop input equations and output equations.
7. Draw the logic diagram.

## Example D Flip-Flop Design Procedure

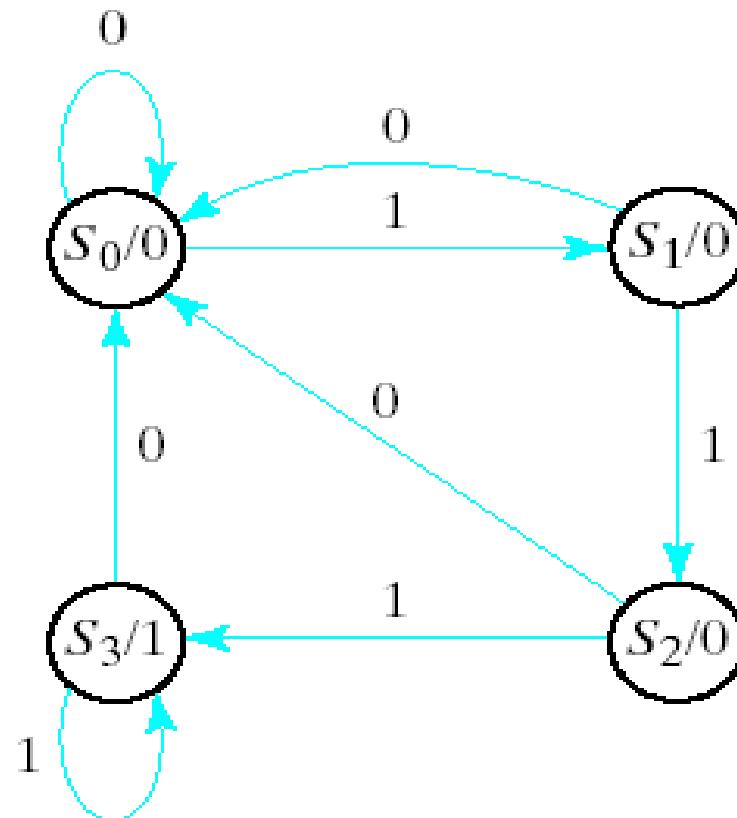


Fig. 5-24 State Diagram for Sequence Detector

# Synthesis Using D Flip-Flops

$$A(t + 1) = D_A(A, B, x) = \Sigma (3, 5, 7)$$

$$B(t + 1) = D_B(A, B, x) = \Sigma (1, 5, 7)$$

$$y(A, B, x) = \Sigma (6, 7)$$

**Table 5-11**  
*State Table for Sequence Detector*

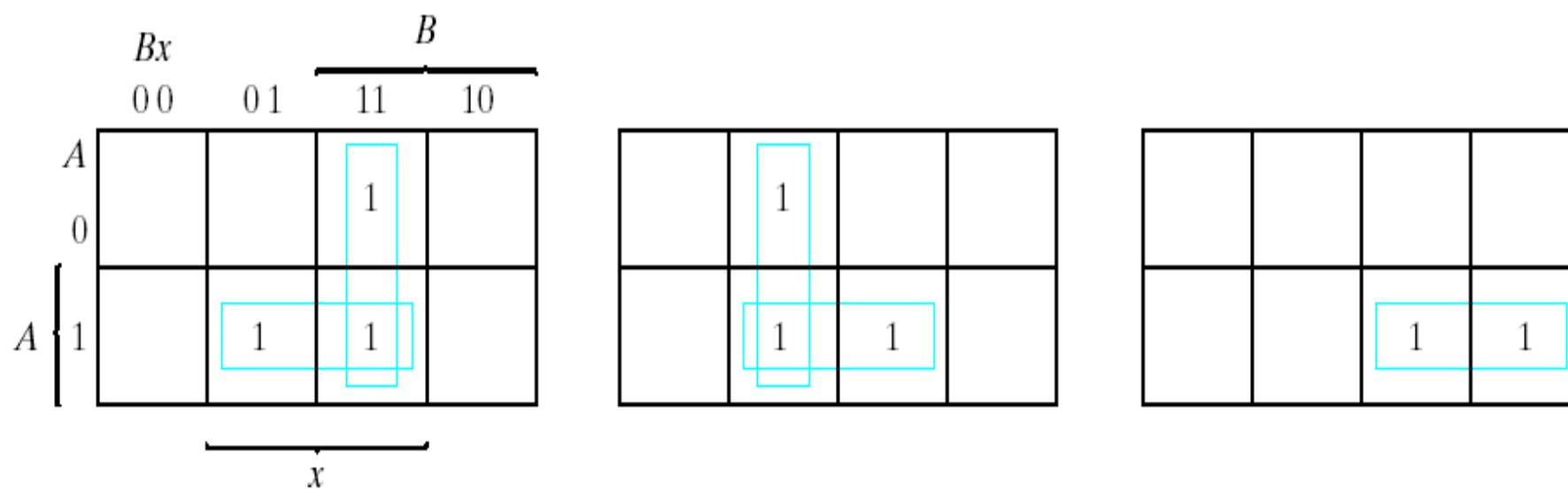
Present State		Input	Next State		Output
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

# Synthesis Using D Flip-Flops

$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$



$$D_A = Ax + Bx$$

$$D_B = Ax + B'x$$

$$y = AB$$

Fig. 5-25 Maps for Sequence Detector

# Synthesis Using D Flip-Flops

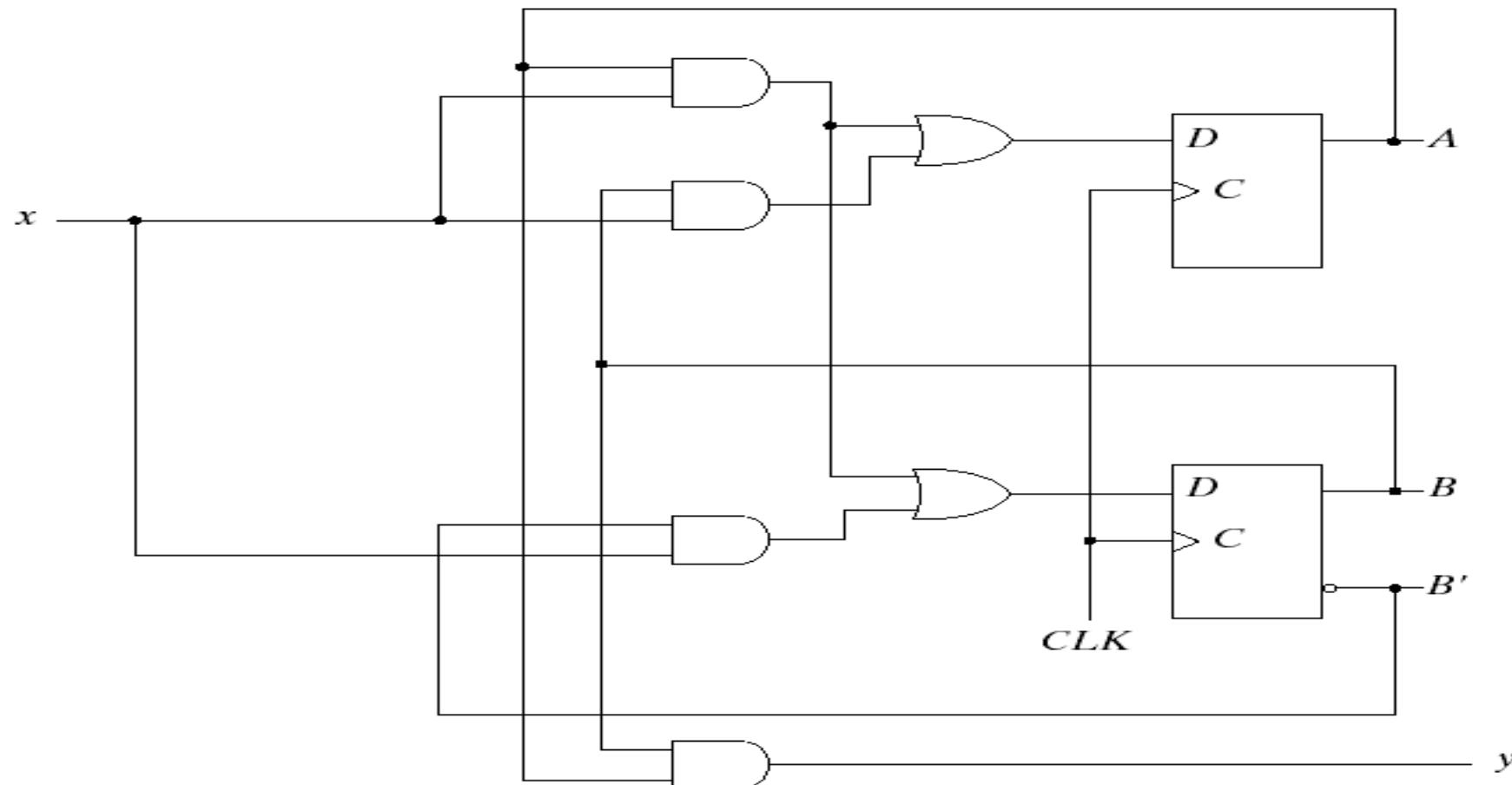


Fig. 5-26 Logic Diagram of Sequence Detector

# Synthesis Using JK Flip-Flops

Different from Table 5-11 !!

**Table 5-12**  
*Flip-Flop Excitation Tables*

$Q(t)$	$Q(t + 1)$	$J$	$K$
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) JK

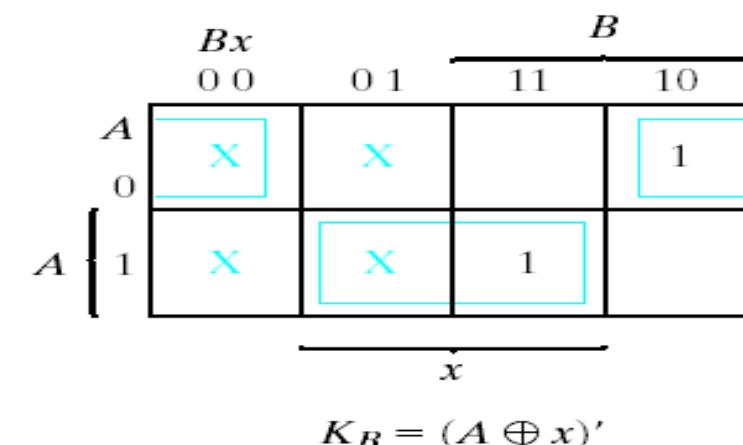
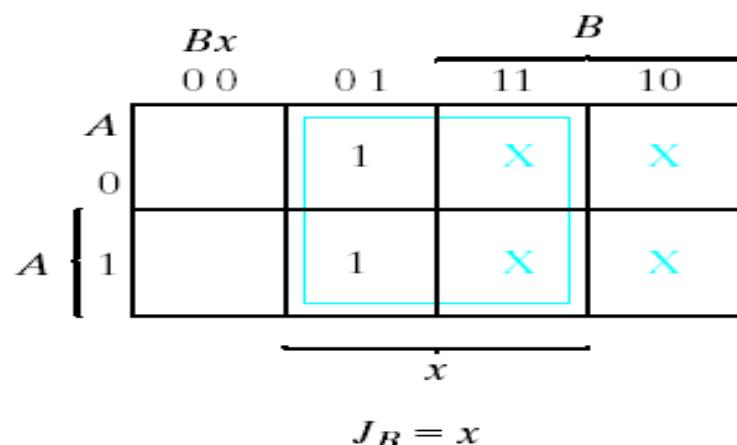
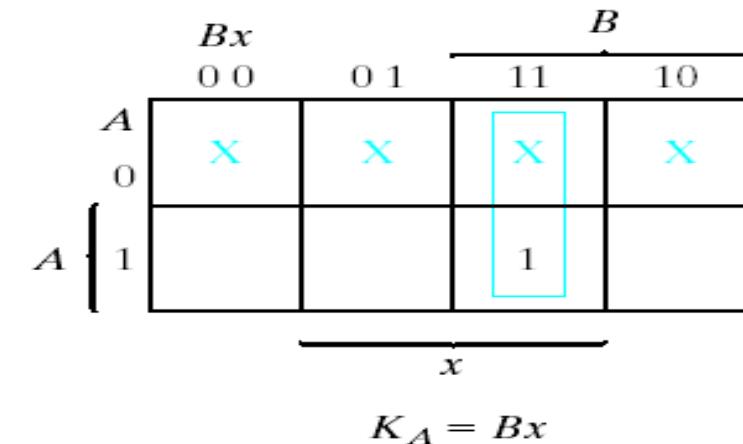
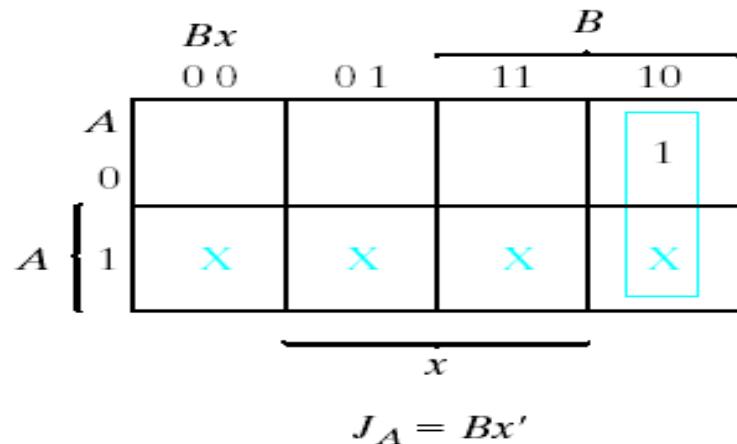
Ref. Table 5-1

**Table 5-13**  
*State Table and JK Flip-Flop Inputs*



Present State		Input	Next State		Flip-Flop Inputs			
$A$	$B$	$x$	$A$	$B$	$J_A$	$K_A$	$J_B$	$K_B$
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

# Synthesis Using JK Flip-Flops



# Synthesis Using JK Flip-Flops

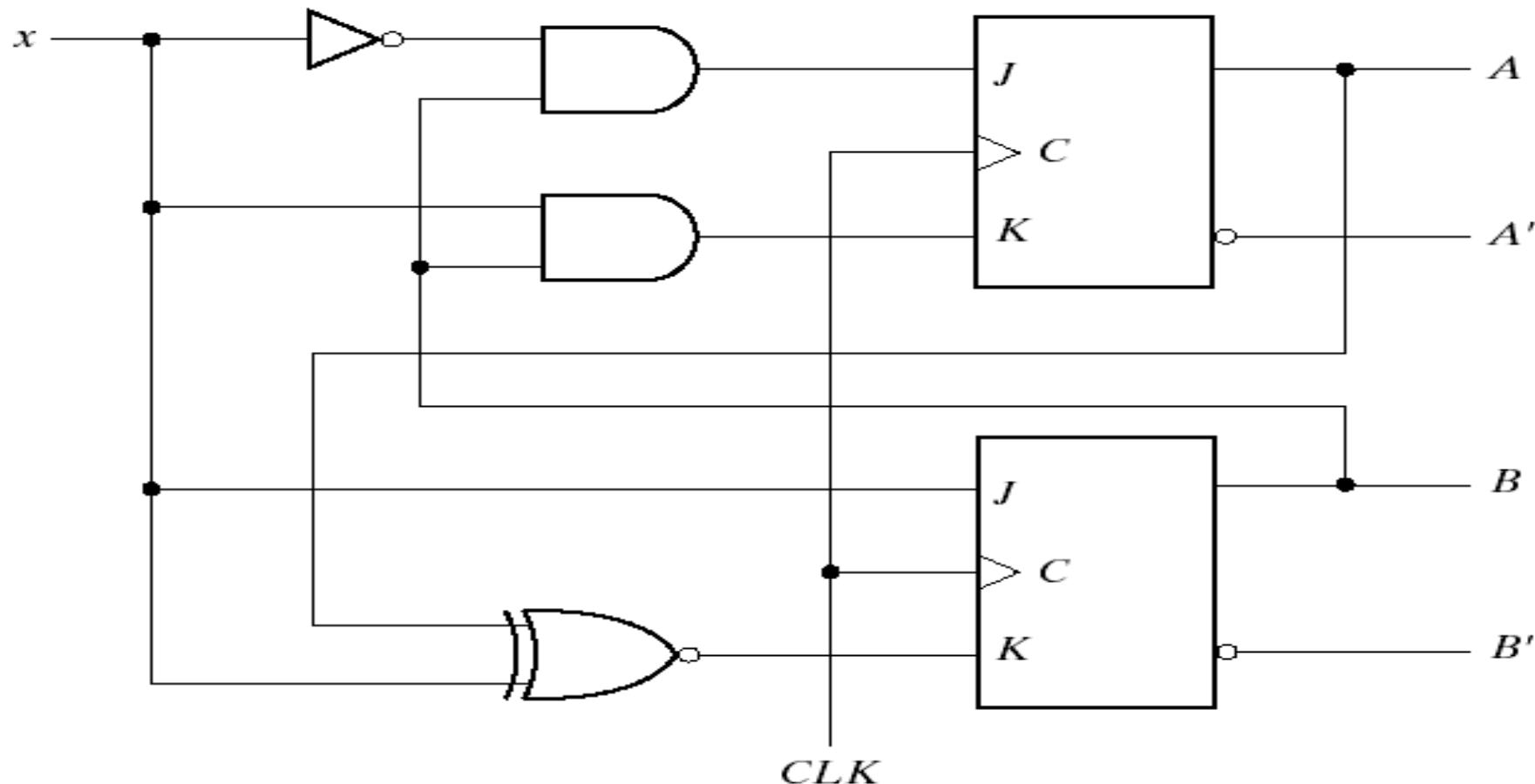


Fig. 5-28 Logic Diagram for Sequential Circuit with JK Flip-Flops

# Synthesis Using T Flip-Flops

The synthesis using **T flip-flops** will be demonstrated by designing a binary counter. An n-bit binary counter consists of n flip-flops that can count in binary from 0 to  $2^n-1$ . The state diagram of a 3-bit counter is shown in Fig. 5-29.

$Q(t)$	$Q(t + 1)$	$T$
0	0	0
0	1	1
1	0	1
1	1	0

Ref. Table 5-1 (b)T

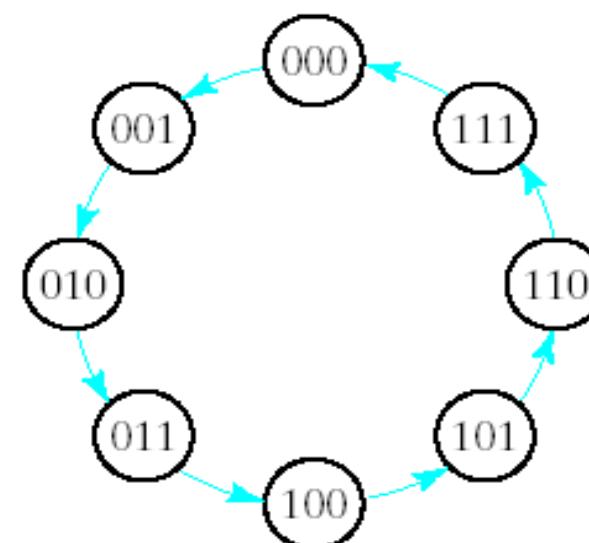


Fig. 5-29 State Diagram of 3-Bit Binary Counter

# Synthesis Using T Flip-Flops

**Table 5-14**  
*State Table for 3-Bit Counter*

Present State			Next State			Flip-Flop Inputs		
$A_2$	$A_1$	$A_0$	$A_2$	$A$	$A_0$	$T_{A2}$	$T_{A1}$	$T_{A0}$
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

# Synthesis Using T Flip-Flops

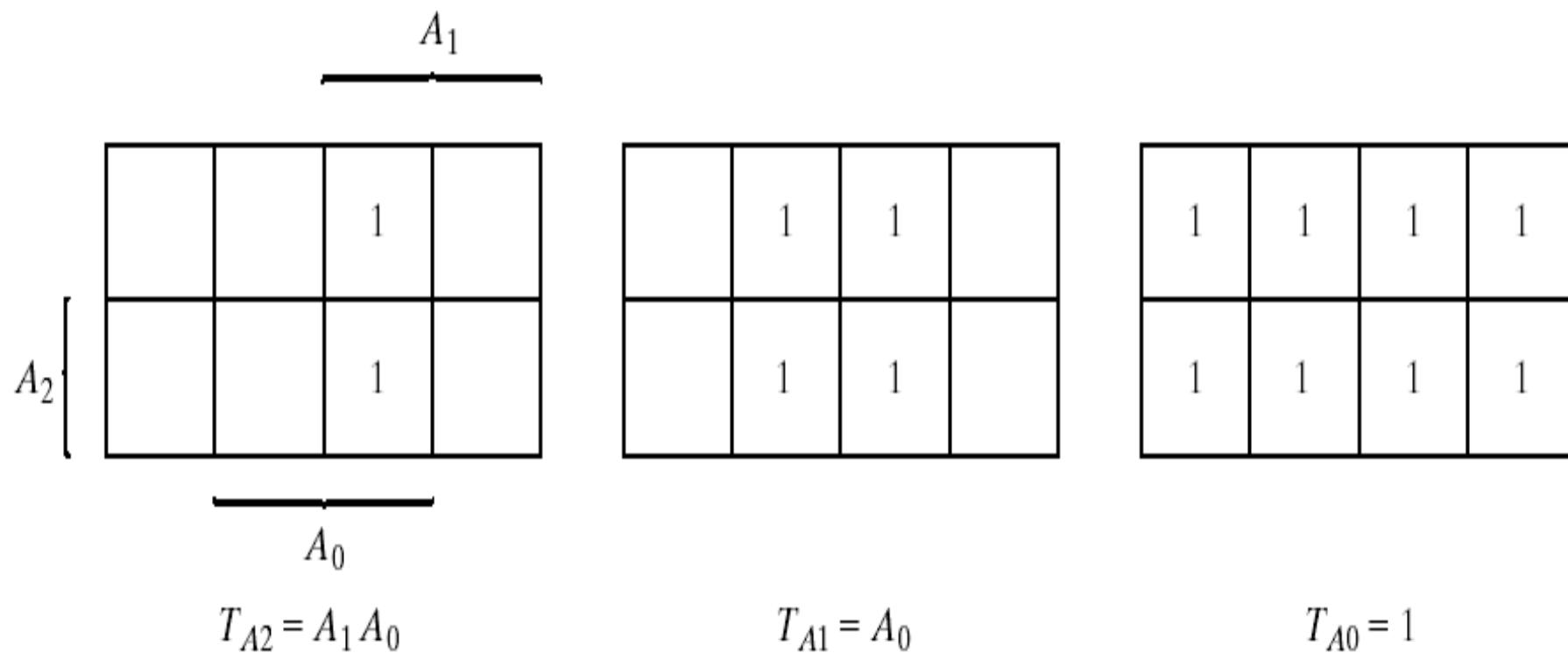


Fig. 5-30 Maps for 3-Bit Binary Counter

# Synthesis Using T Flip-Flops

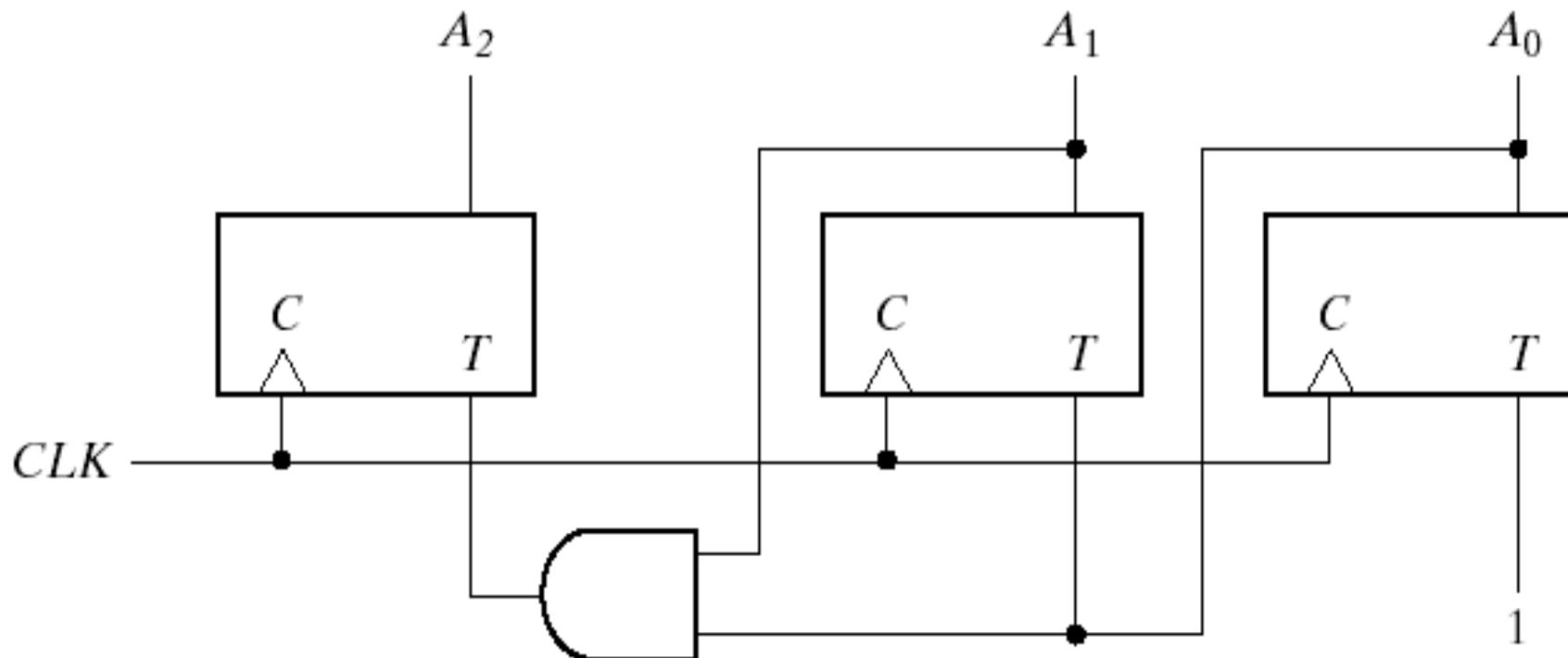


Fig. 5-31 Logic Diagram of 3-Bit Binary Counter

# State Reduction and Assignment

## State Reduction

- Reducing the number of states in a state table, while keeping the external input-output requirements unchanged
- Example:
  - Total 7 states
  - A sequence as follows

state	a	a	b	c	d	e	f	f	g	f	g	a
input	0	1	0	1	0	1	1	1	0	1	0	0
output	0	0	0	0	0	1	1	0	1	0	0	0

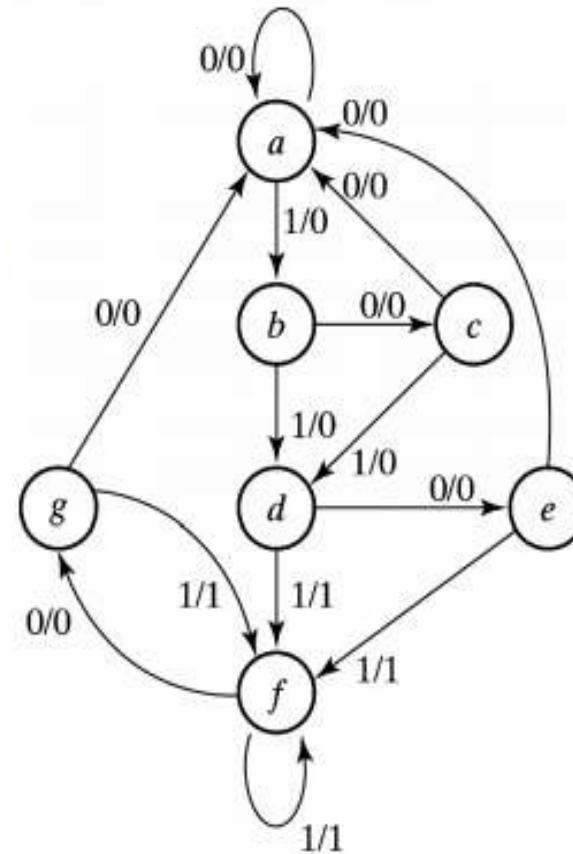


Fig. 5-22 State Diagram

# State Reduction and Assignment (Contd.)

## State Reduction Rules

- Two states are said to be equivalent if, for every possible inputs, they give exactly the same output and have equivalent next state

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

delete state g and replaced with state e

# State Reduction and Assignment (Contd.)

## Further State Reduction

- After the first reduction, we can see that state d and state f will have the same output and next state for both  $x=0$  and  $x=1$ 
  - Further reduce one state

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

delete state f and replaced with state d

# State Reduction and Assignment (Contd.)

## Reduced State Diagram

- After reduction, the circuit has only 5 states with same input/output requirements
- Original output sequence:

state	a	a	b	c	d	e	f	f	g	f	g	a
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

- Reduced output sequence:

state	a	a	b	c	d	e	d	d	e	d	e	a
input	0	1	0	1	0	1	1	0	1	0	0	
output	0	0	0	0	0	1	1	0	1	0	0	

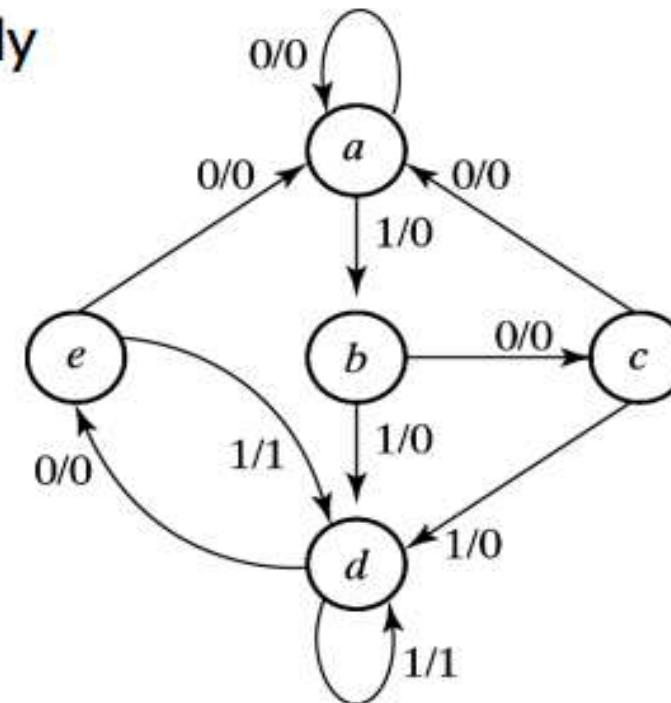


Fig. 5-23 Reduced State Diagram

# State Reduction and Assignment

## (Contd.)

### State Assignment

State	Binary	Gray Code	One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

# State Reduction and Assignment (Contd.)

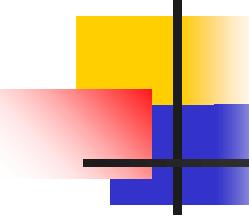
## Reduced State Table: Binary State Assignment

State	Next State		Output	
	x=0	x=1	x=0	x=1
001	000	001	0	0
010	010	011	0	0
011	000	011	0	0
100	100	011	0	1
101	000	011	0	1

# State Reduction and Assignment (Contd.)

## Reduced State Table: Binary State Assignment

State	Next State		Output	
	x=0	x=1	x=0	x=1
001	000	001	0	0
010	010	011	0	0
011	000	011	0	0
100	100	011	0	1
101	000	011	0	1



# Design Procedure

---

- Develop State Diagram From Specs
- Reduce States
- Assign Binary values to States
- Write Binary-coded State Table
- Choose Flip-Flops
- Derive Input and Output Equations
- Draw the Logic Diagram

# Develop State Diagram: Sequence Detector

- Detect 3 or more 1s in sequence (a Moore Model)

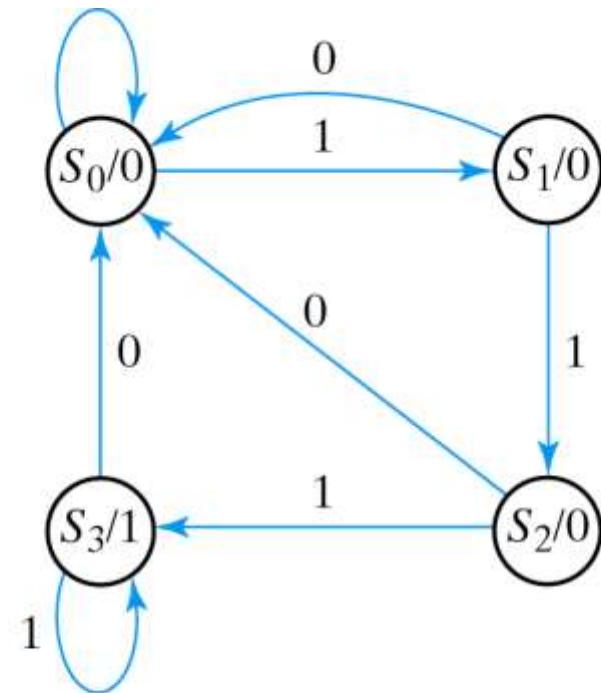


Fig. 5-24 State Diagram for Sequence Detector

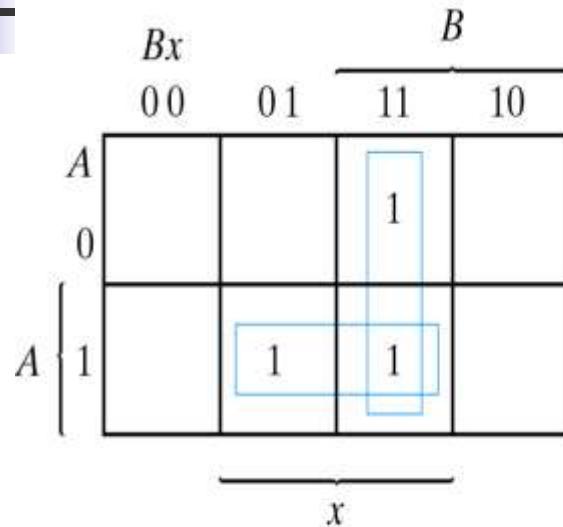
# D Flip-Flop Input Equations

State		Input x	Next State		Output y
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

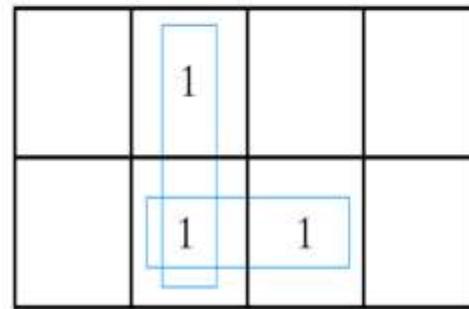
Input equations come directly from the next state in D Flip-Flop design

$$\begin{aligned}A(t+1) &= D_A(A, B, x) = \Sigma(3, 5, 7) \\B(t+1) &= D_B(A, B, x) = \Sigma(1, 5, 7) \\y(A, B, x) &= \Sigma(6, 7)\end{aligned}$$

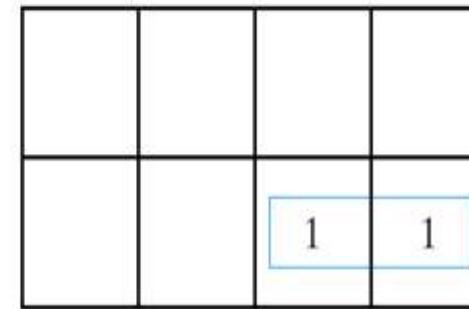
# Simplified Boolean Equations



$$D_A = Ax + Bx$$



$$D_B = Ax + B'x$$



$$y = AB$$

Fig. 5-25 Maps for Sequence Detector

# Sequence Detector: D Flip-Flops

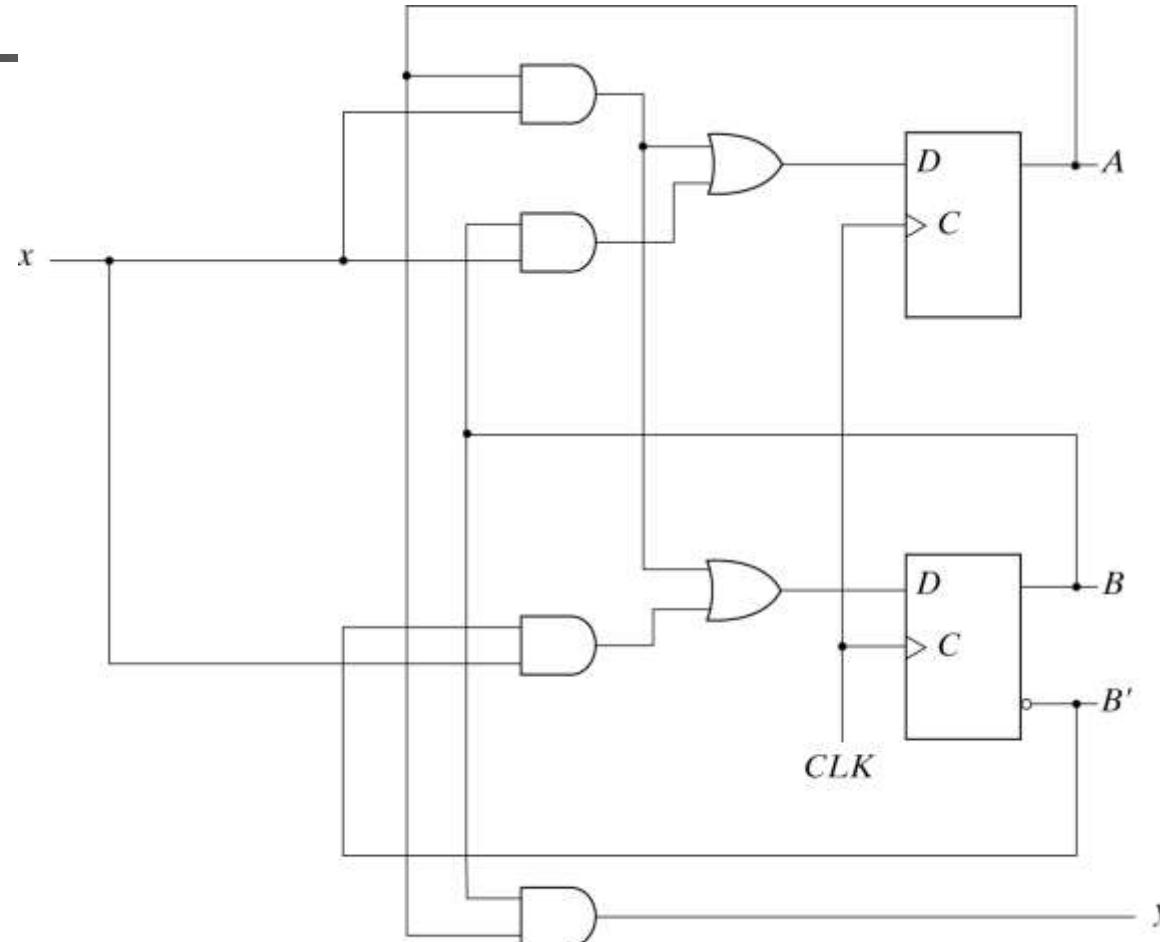


Fig. 5-26 Logic Diagram of Sequence Detector

# Using JK or T Flip-Flops

## 1. Develop Excitation Table Using Excitation Tables

JK Flip-Flop				T Flip-Flop		
Q(t)	Q(t+1)	J	K	Q(t)	Q(t+1)	T
0	0	0	X	0	0	0
0	1	1	X	0	1	1
1	0	X	1	1	0	1
1	1	X	0	1	1	0

# State Table: JK Flip-Flop Inputs

Present State		Input x	Next State		Flip-Flop Inputs			
A	B		A	B	$J_A$	$K_A$	$J_B$	$K_B$
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0	1
1	0	1	1	0	1	0	1	0
1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

# Maps for J and K Input Equations

		Bx		B	
		00	01	11	10
A					
A	0				1
A	1	X	X	X	X

$$J_A = Bx'$$

		Bx		B	
		00	01	11	10
A					
A	0	X	X	X	X
A	1			1	

$$K_A = Bx$$

		Bx		B	
		00	01	11	10
A					
A	0		1	X	X
A	1		1	X	X

$$J_B = x$$

		Bx		B	
		00	01	11	10
A					
A	0	X	X		1
A	1	X	X	1	

$$K_B = (A \oplus x)'$$

Fig. 5-27 Maps for J and K Input Equations

# JK Flip-Flop Sequence Detector

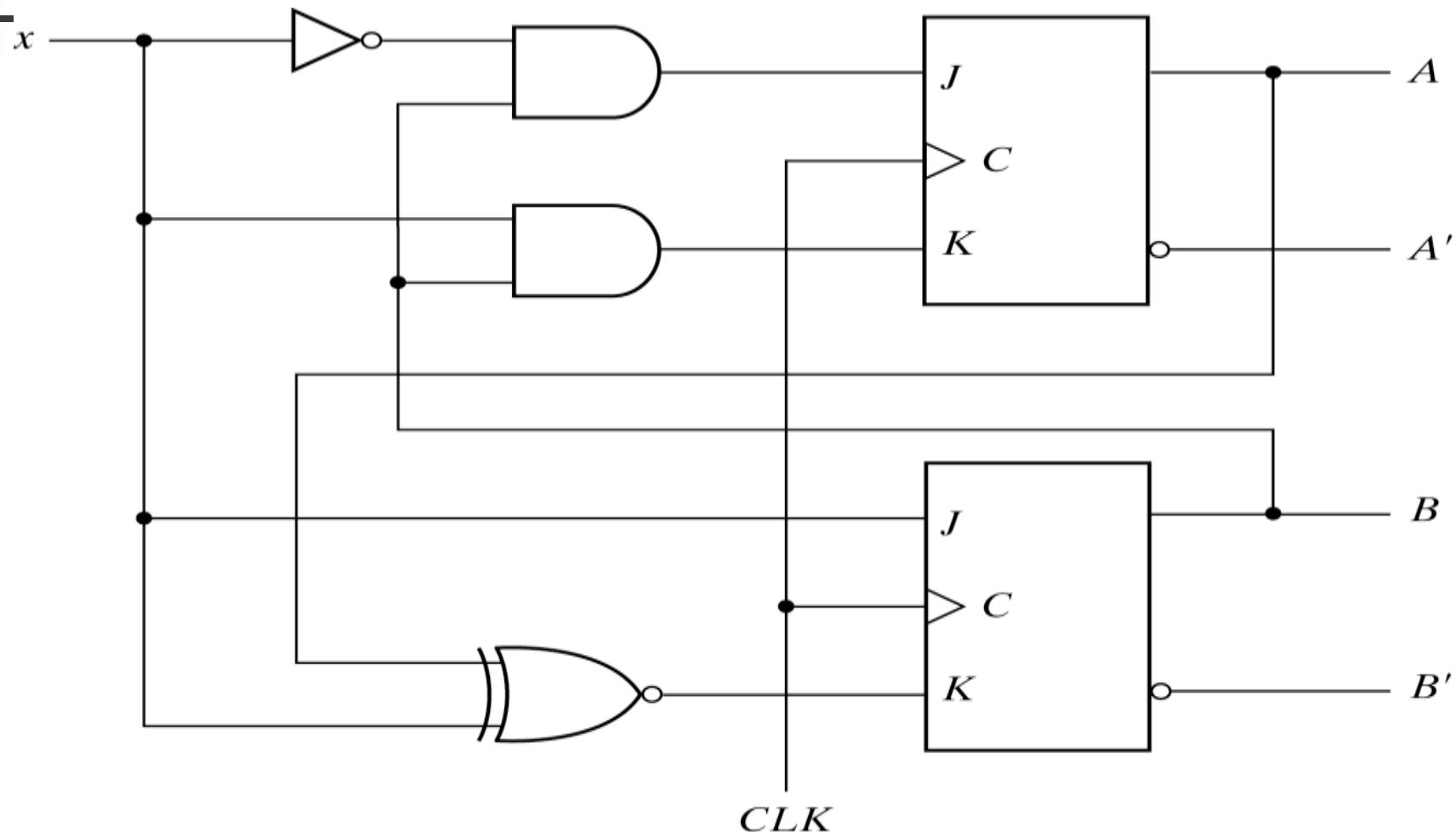


Fig. 5-28 Logic Diagram for Sequential Circuit with *JK* Flip-Flops

B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

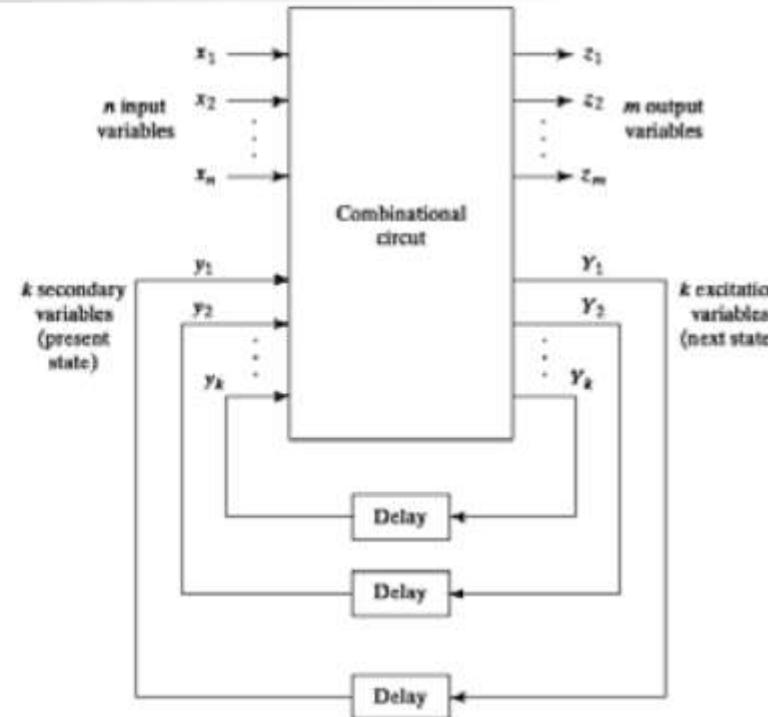
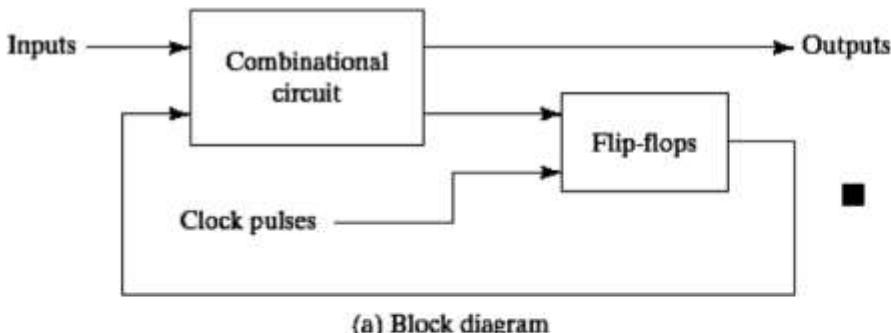
UNIT 4

Session-11( Asynchronous Sequential Circuits, Transition Table)

Session-12( State Table, Flow Table)

# Synchronous vs. Asynchronous

- Asynchronous sequential circuits
  - Internal states can change at **any instant** of time when there is a change in the input variables
  - **No clock** signal is required
  - Have better performance but hard to design due to timing problems



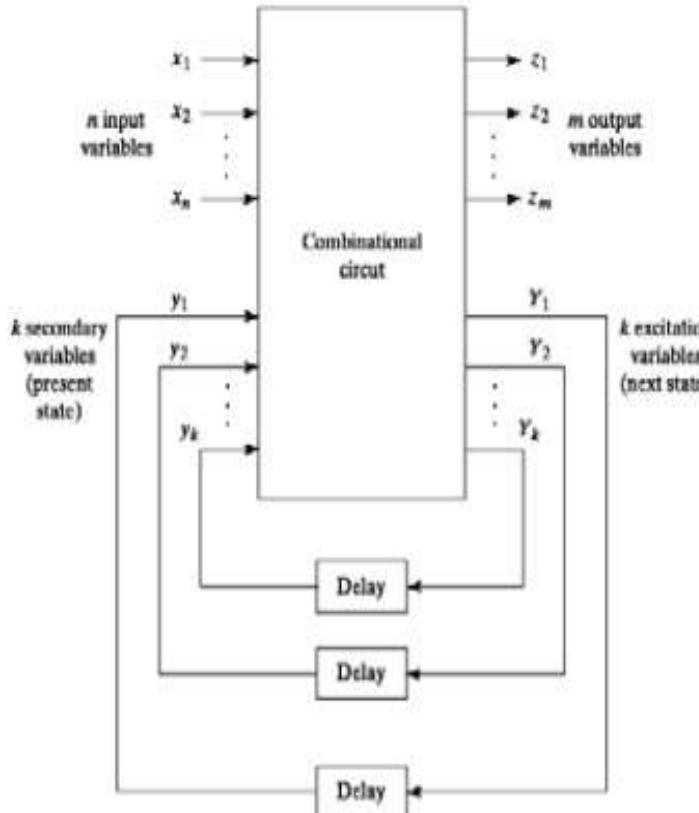
- Synchronous sequential circuits
  - Synchronized by a **periodic** train of clock pulses
  - Much easier to design (preferred design style)

# Why Asynchronous Circuits ?

- Used when speed of operation is important
  - Response quickly without waiting for a clock pulse
- Used in small independent systems
  - Only a few components are required
- Used when the input signals may change independently of internal clock
  - Asynchronous in nature
- Used in the communication between two units that have their own independent clocks
  - Must be done in an asynchronous fashion

# Definitions of Asyn. Circuits

- Inputs / Outputs
- Delay elements:
  - Only a short term memory
  - May not really exist due to original gate delay
- Secondary variable:
  - Current state (small  $y$ )
- Excitation variable:
  - Next state (big  $Y$ )
  - Have some delay in response to input changes



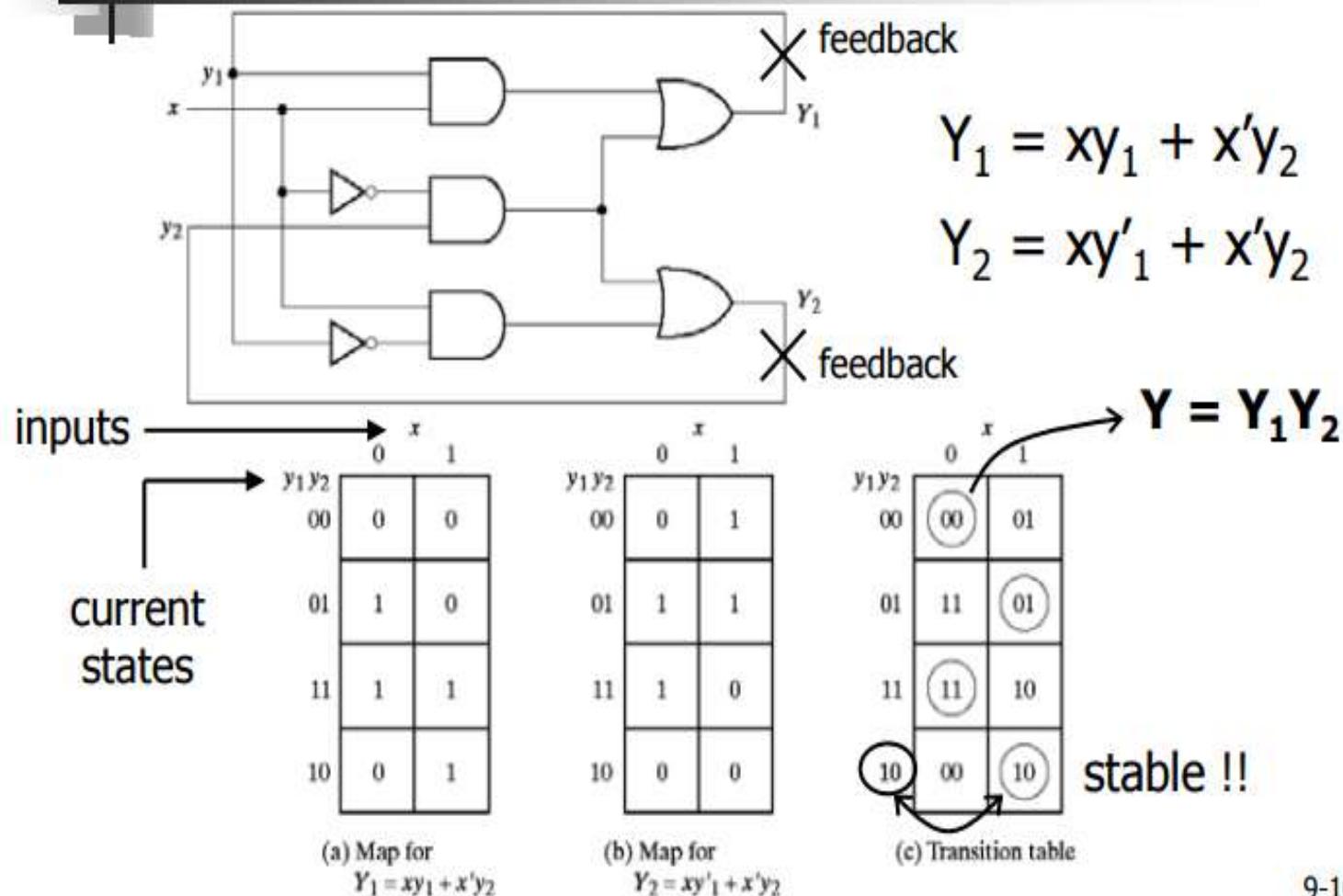
# Operational Mode

- Steady-state condition:
  - Current states and next states are the same
  - Difference between  $Y$  and  $y$  will cause a transition
- Fundamental mode:
  - No simultaneous changes of two or more variables
  - The time between two input changes must be longer than the time it takes the circuit to a stable state
  - The input signals change one at a time and only when the circuit is in a stable condition

# Transition Table

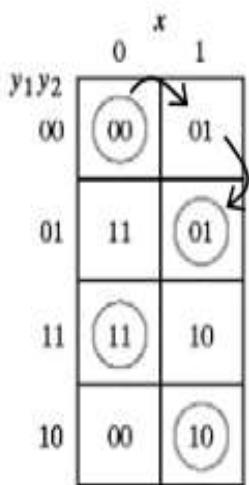
- Transition table is useful to analyze an asynchronous circuit from the circuit diagram
- Procedure to obtain transition table:
  1. Determine all feedback loops in the circuits
  2. Mark the input ( $y_i$ ) and output ( $Y_i$ ) of each feedback loop
  3. Derive the Boolean functions of all  $Y$ 's
  4. Plot each  $Y$  function in a map and combine all maps into one table
  5. Circle those values of  $Y$  in each square that are equal to the value of  $y$  in the same row

# An Example of Transition Table



# State Table

- When input  $x$  changes from 0 to 1 while  $y=00$ :
  - $Y$  changes to 01 → unstable
  - $y$  becomes 01 after a short delay → stable at the second row
  - The next state is  $Y=01$
- Each row must have ***at least one*** stable state
- Analyze each state in this way can obtain its state table



(c) Transition table

$y_1y_2$	$x$	Present State		Next State		$y_1y_2x$ : total state
				X=0	X=1	
00	0	00	01	0	0	4 stable
01	1	11	01	0	1	total states: 000,011,
11	0	11	10	1	1	110,101
10	1	00	10	0	0	

# Flow Table

- Similar to a transition table except the states are represented by **letter symbols**
- Can also include the output values
- Suitable to obtain the logic diagram from it
- Primitive flow table:

**only one** stable state in each row  
(ex: 9-4(a))

Equivalent to 9-3(c) if  
 $a=00$ ,  $b=01$ ,  $c=11$ ,  $d=10$

		$x$	
		0	1
$a$	0	$a$	$b$
	1	$c$	$b$
	0	$c$	$d$
	1	$a$	$d$

(a) Four states with one input

		$x_1 x_2$			
		00	01	11	10
$a$	00	$a$ , 0	$a$ , 0	$a$ , 0	$b$ , 0
	10	$a$ , 0	$a$ , 0	$b$ , 1	$b$ , 0

(b) Two states with two inputs and one output

# Flow Table to Circuits

- Procedure to obtain circuits from flow table:
  - Assign to each state a distinct binary value  
(convert to a transition table)
  - Obtain circuits from the map
- Two difficulties:
  - The binary state assignment (to avoid race)
  - The output assigned to the unstable states

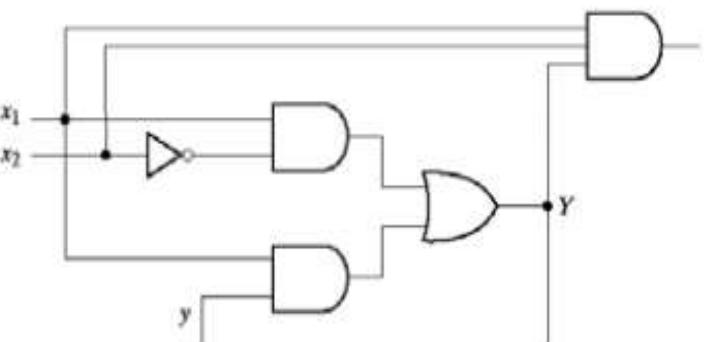
Ex: from the flow table 9-4(b)

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

(a) Transition table  
 $Y = x_1x_2' + x_1y$

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

(b) Map for output  
 $z = x_1x_2y$



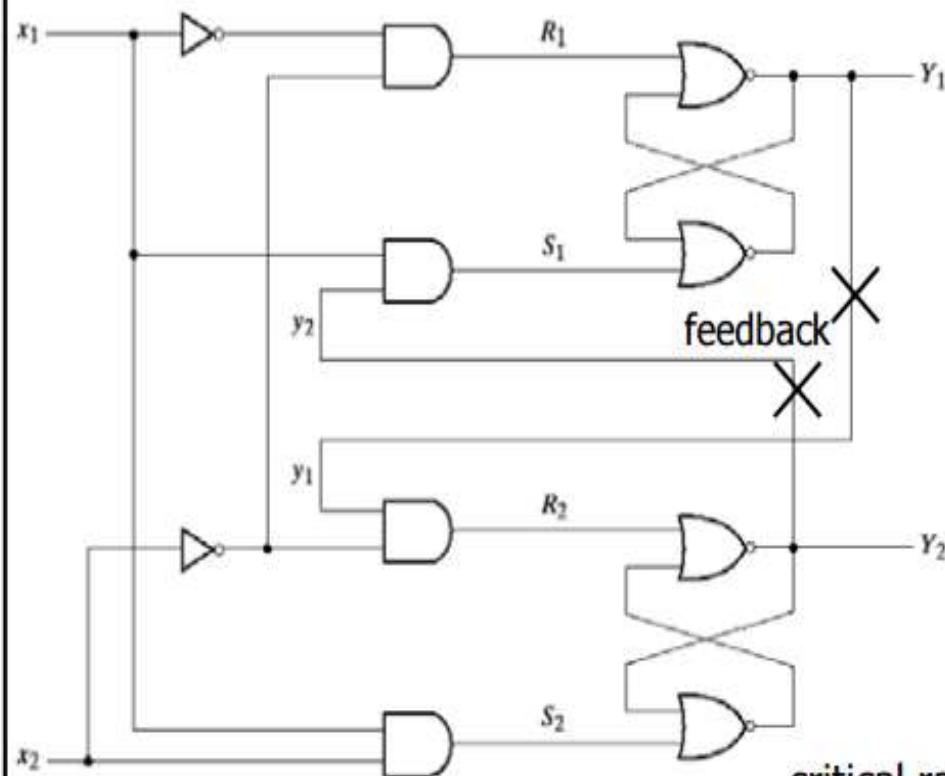
(c) Logic diagram

# Analysis Procedure

- Procedure to analyze an asynchronous sequential circuits with SR latches:
  1. Label each latch output with  $Y_i$  and its external feedback path (if any) with  $y_i$
  2. Derive the Boolean functions for each  $S_i$  and  $R_i$
  3. Check whether  $SR=0$  (NOR latch) or  $S'R'=0$  (NAND latch) is satisfied
  4. Evaluate  $Y=S+R'y$  (NOR latch) or  $Y=S'+Ry$  (NAND latch)
  5. Construct the transition table for  $Y=Y_1Y_2\dots Y_k$
  6. Circle all stable states where  $Y=y$

# Analysis Example

$$S_1 = x_1 y_2 \quad R_1 = x'_1 x'_2 \Rightarrow S_1 R_1 = x_1 y_2 x'_1 x'_2 = 0 \text{ (OK)}$$
$$S_2 = x_1 x_2 \quad R_2 = x'_2 y_1 \Rightarrow S_2 R_2 = x_1 x_2 x'_2 y_1 = 0 \text{ (OK)}$$



$$Y_1 = S_1 + R'_1 y_1 \\ = x_1 y_2 + (x_1 + x_2) y_1 \\ = x_1 y_2 + x_1 y_1 + x_2 y_1$$
$$Y_2 = S_2 + R'_2 y_2 \\ = x_1 x_2 + (x_2 + y'_1) y_2 \\ = x_1 x_2 + x_2 y_2 + y'_1 y_2$$

		$x_1 x_2$			
		00	01	11	10
$y_1 y_2$	00	(00)	(00)	01	(00)
	01	(01)	(01)	11	11
11	00	00	(11)	(11)	10
	10	00	(10)	11	(10)

critical race !!

# **Definitions:**

## **State Table:**

The relationship that exists among the inputs, outputs, present **states** and next **states** can be specified by either the **state table** or the **state diagram**

## **Transition Table:**

Transition table is table of states and transition, useful to analyze an asynchronous circuit from the circuit diagram

## **Flow Table:**

In a flow table the states are named by letter symbols

B.Tech – CSE – Sem. 3  
18CSS201J – ANALOG AND DIGITAL  
ELECTRONICS  
(Regulations 2018)

UNIT 4

Session- melborP dna sisylanA stiucriC latineuqeS suonorhcnyS )13  
(noisseS gnivloS

# Asynchronous Sequential Circuits

Asynchronous sequential circuits basics

- ✓ No clock signal is required
- ✓ Internal states can change at any instant of time when there is a change in the input variables
- ✓ Have better performance but hard to design due to timing problems

Why Asynchronous Circuits?

- ✓ Accelerate the speed **of the machine** (no need to wait for the **next clock pulse**).
- ✓ Simplify the circuit **in the small independent gates**.
- ✓ Necessary when having **multi circuits** each having its own clock.

Analysis Procedure

- ✓ The analysis consists of obtaining a table or a diagram that describes the sequence of internal states and outputs as a function of changes in the input variables.

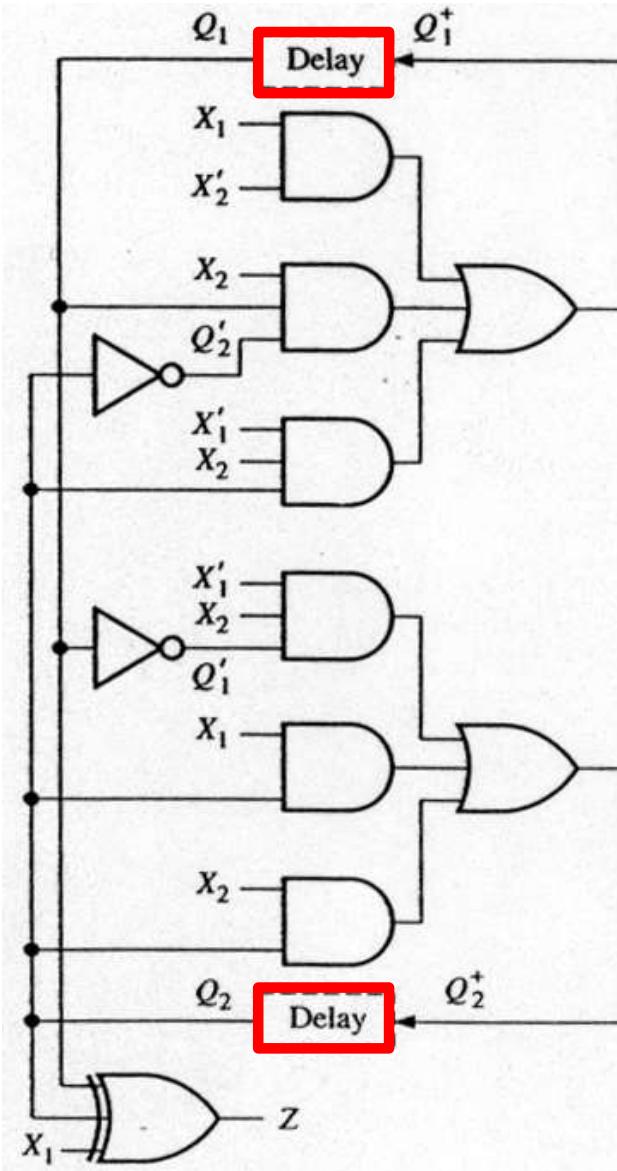
# Example Circuit

- ✓ Construction of Asynchronous Circuits:
  - using only gates
  - with feedback paths
- ✓ Analysis:
  - Lump all of the delay associated with each feedback path into a “delay” box
  - Associate a state variable with each delay output
  - Construct the flow table
- ✓ Network equations

$$Q_1^+ = X_1 X_2' + X_1' X_2 Q_2 + X_2 Q_1 Q_2'$$

$$Q_2^+ = X_1' X_2 Q_1' + X_1 Q_2 + X_2 Q_2$$

$$Z = X_1 \oplus Q_1 \oplus Q_2$$



# Example Circuit: Output Table

- ✓ 1. Starting in total state  $x_1x_2q_1q_2=0000$
  - ✓ 2. Input changes to 01
    - Internal state changes to 01 and then to 11.
  - ✓ 3. Input changes to 11.
    - Go to unstable total state 1111 and then to 1101.
  - ✓ 4. Input changes to 10.
    - Go to unstable total state 1001 and then to 1011.
  - ✓ The output sequence:  
 $0(0)(1)0(1)0(0)1$ 
    - Condensed to the form  
 $0(1)0(1)01$ .
    - Two transient 1 outputs can be eliminated by proper design.

$X_1 X_2$	00	01	11	10
$Q_1 Q_2$	00	01	00	10
	00	11	01	11
	00	11	01	11
	00	10	10	10

$X_1 X_2$	00	01	11	10
$Q_1 Q_2$	0	0	1	1
	1	1	0	0
	0	0	1	1
	1	1	0	0

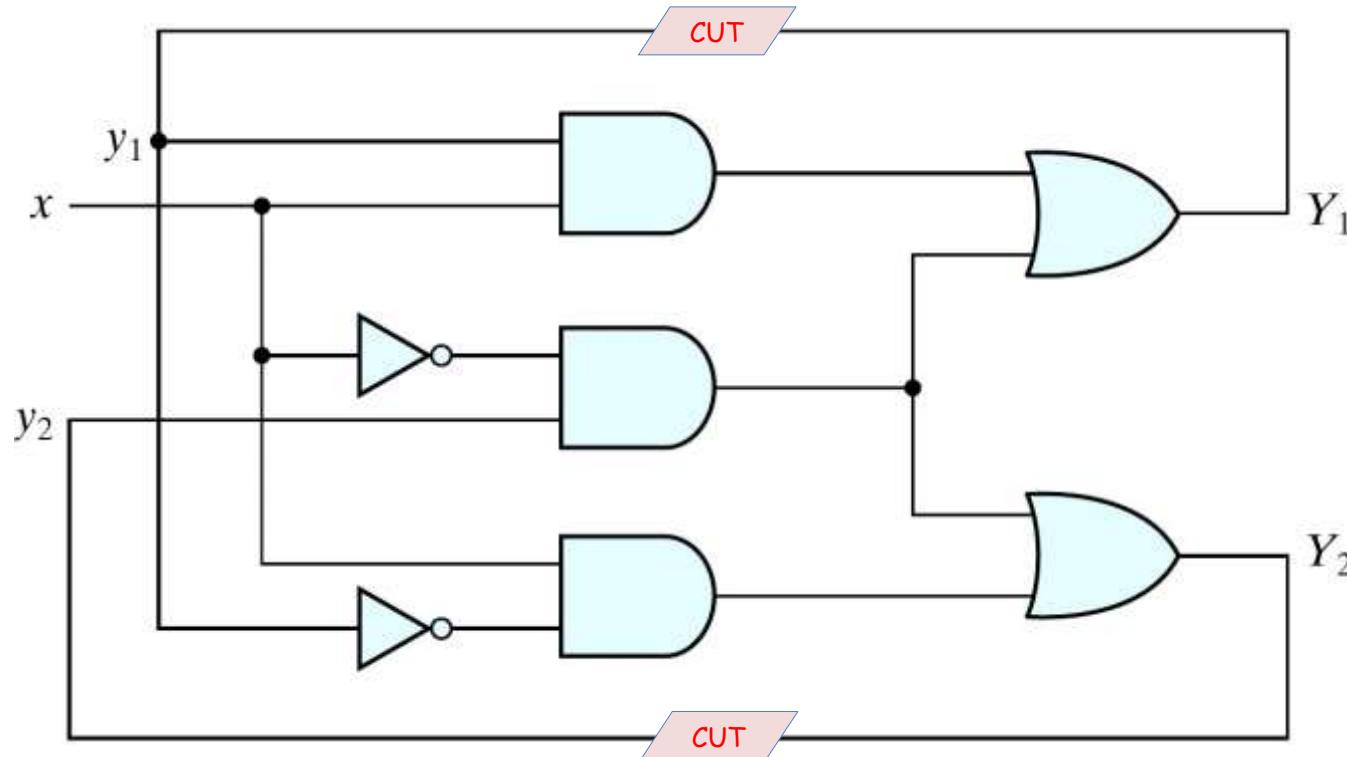
# Transition Table

- ✓ Transition table is useful to analyze an asynchronous circuit from the circuit diagram. Procedure to obtain transition table:
  1. Determine all feedback loops in the circuits
  2. Mark the input ( $y_i$ ) and output ( $Y_i$ ) of each feedback loop
  3. Derive the Boolean functions of all  $Y$ 's
  4. Plot each  $Y$  function in a map and combine all maps into one table (flow table)
  5. Circle those values of  $Y$  in each square that are equal to the value of  $y$  in the same row

# Asynchronous Sequential Circuit

✓ The excitation variables:  $Y_1$  and  $Y_2$

- $Y_1 = \underline{x}\bar{y}_1 + xy_2$
- $Y_2 = xy_1 + \bar{x}y_2$



# Transition Table

- ✓ Combine the internal state with input variables
  - Stable total states:

		$x$
		0      1
$y_1 y_2$	00	0      0
	01	1      0
	11	1      1
	10	0      1

(a) Map for  
 $Y_1 = xy_1 + x'y_2$

		$x$
		0      1
$y_1 y_2$	00	0      1
	01	1      1
	11	1      0
	10	0      0

(b) Map for  
 $Y_2 = xy'_1 + x'y_2$

		$x$
		0      1
$y_1 y_2$	00	00      01
	01	11      01
	11	11      10
	10	00      10

(c) Transition table

# Transition Table

- ✓ In an asynchronous sequential circuit, the internal state can change immediately after a change in the input.
- ✓ It is sometimes convenient to combine the internal state with input value together and call it the **Total State of the circuit.**  
(Total state = Internal state + Inputs)
- ✓ In the example , the circuit has
  - 4 **stable total states:** ( $y_1y_2x=000, 011, 110, \text{ and } 101$ )
  - 4 **unstable total states:** ( $y_1y_2x=001, 010, 111, \text{ and } 100$ )

$y_1y_2$	$x$	
	0	1
00	00	01
01	11	01
11	11	10
10	00	10

# Transition Table

- ✓ If  $y=00$  and  $x=0 \rightarrow Y=00$  (Stable state)
- ✓ If  $x$  changes from 0 to 1 while  $y=00$ , the circuit changes  $Y$  to 01 which is temporary unstable condition ( $Y \neq y$ )
- ✓ As soon as the signal propagates to make  $Y=01$ , the feedback path causes a change in  $y$  to 01.  
(transition from the first row to the second row)
- ✓ If the input alternates between 0 and 1, the circuit will repeat the sequence of states

		$x$	
		0	1
$y_1 y_2$	00	00	01
	01	11	01
11	11	11	10
	10	00	10



# Flow Table

- ✓ A flow table is similar to a transition table except that the internal state are symbolized with letters rather than binary numbers.
- ✓ It also includes the output values of the circuit for each stable state.

	$x$	0	1
$y$	a	(a)	b
	b	c	(b)
	c	(c)	d
	d	a	(d)

(a) Four states with one input

	$x_1x_2$	00	01	11	10
$a$	(a), 0	(a), 0	(a), 0	b , 0	
	a , 0	a , 0	(b), 1	(b), 0	

(b) Two states with two inputs and one output

# Flow Table

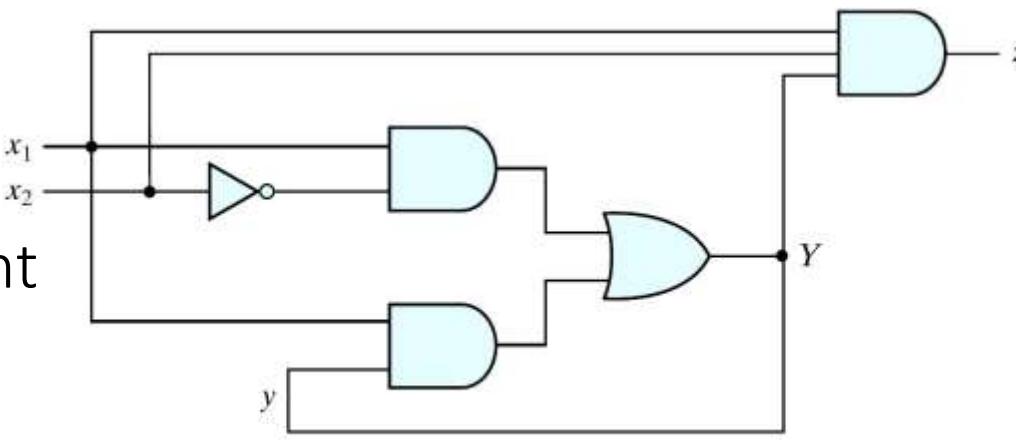
- ✓ In order to obtain the circuit described by a flow table, it is necessary to convert the flow table into a transition table from which we can derive the logic diagram.

$y$	$x_1x_2$	00	01	11	10
0	0	0	0	1	
1	0	0	1	1	

(a) Transition table  
 $Y = x_1x'_2 + x_1y$

$y$	$x_1x_2$	00	01	11	10
0	0	0	0	0	
1	0	0	1	0	

(b) Map for output  
 $z = x_1x_2 y$



(c) Logic diagram

# Race condition

- ✓ Two or more binary state variables will change value when one input variable changes.
- ✓ Cannot predict state sequence if unequal delay is encountered.
- ✓ Non-critical race: The final stable state does not depend on the change order of state variables
- ✓ Critical race: The change order of state variables will result in different stable states.  
Must be avoided !!

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

(a) Possible transitions:

$00 \rightarrow 11$   
 $00 \rightarrow 01 \rightarrow 11$   
 $00 \rightarrow 10 \rightarrow 11$

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

(b) Possible transitions:

$00 \rightarrow 11 \rightarrow 01$   
 $00 \rightarrow 01$   
 $00 \rightarrow 10 \rightarrow 11 \rightarrow 01$

		0	1
		00	11
00	00	00	11
01		<td>01</td>	01
11		<td>11</td>	11
10		<td>10</td>	10

(a) Possible transitions:

$00 \rightarrow 11$   
 $00 \rightarrow 01$   
 $00 \rightarrow 10$

		0	1
		00	11
00	00	00	11
01		<td>11</td>	11
11		<td>11</td>	11
10		<td>10</td>	10

(b) Possible transitions:

$00 \rightarrow 11$   
 $00 \rightarrow 01 \rightarrow 11$   
 $00 \rightarrow 10$

# Race Solution

- ✓ It can be solved by making a proper binary assignment to the state variables.
- ✓ The state variables must be assigned binary numbers in such a way that only one state variable can change at any one time when a state transition occurs in the flow table.

$y_1y_2$	$x$	0	1
00	00	01	
01		11	
11		10	
10		10	

(a) State transition:  
 $00 \rightarrow 01 \rightarrow 11 \rightarrow 10$

$y_1y_2$	$x$	0	1
00	00	01	
01		11	
11		11	
10		10	

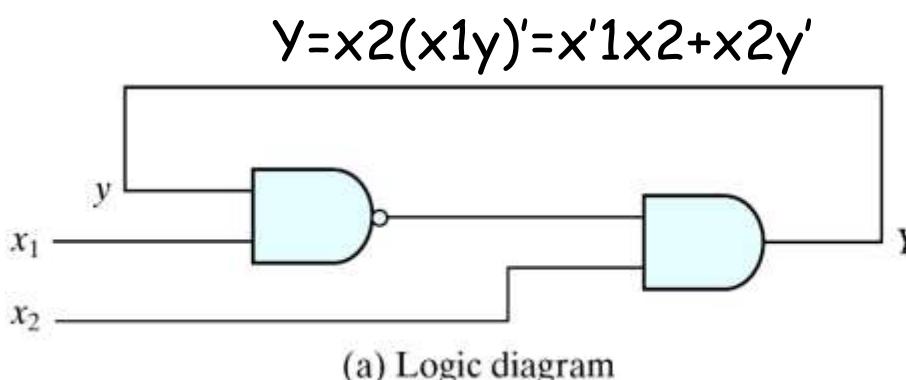
(b) State transition:  
 $00 \rightarrow 01 \rightarrow 11 \rightarrow 10$

$y_1y_2$	$x$	0	1
00	00	01	
01		11	
11		10	
10		01	

(c) Unstable  
 $\rightarrow 01 \rightarrow 11 \rightarrow 10 \rightarrow \dots$

# Stability Check

- ✓ Asynchronous sequential circuits may oscillate between unstable states due to the feedback
  - Must check for stability to ensure proper operations
- ✓ Can be easily checked from the transition table
  - Any column has no stable states  $\rightarrow$  unstable  
Ex: when  $x_1x_2=11$  in (b), Y and y are never the same



	$x_1x_2$	00	01	11	10
0	0	1	1	0	
1	0	1	0	0	

(b) Transition table

# SHIFT REGISTERS

S-1

SLO-1

Registers and Types of Registers- Serial In - Serial Out, Serial In -  
Parallel out

SLO-2

Parallel In - Serial Out, Parallel In - Parallel Out

# Registers

- A collection of flip-flops taken as an entity.
- Function: Hold information within a digital system so that it is available to the logic elements during the computing process.
- Each combination of stored information is known as the state or content of the register.
- Shift register: Registers that are capable of moving information upon the occurrence of a clock-signal.
  - Unidirectional
  - Bidirectional
- Two basic ways in which information can be entered/outputted
  - Parallel: All 0/1 symbols handled simultaneously. Require as many lines as symbols being transferred.
  - Serial: Involves the symbol-by-symbol availability of information in a time sequence.
- Four possible ways registers can transfer information:
  - Serial-in/serial-out
  - Serial-in/parallel-out
  - Parallel-in/parallel-out
  - Parallel-in/serial-out

# Serial in Serial out (SISO) Shift Register - serial right shift register

Reference:<https://www.electrical4u.com/serial-in-serial-out-siso-shift-register/>

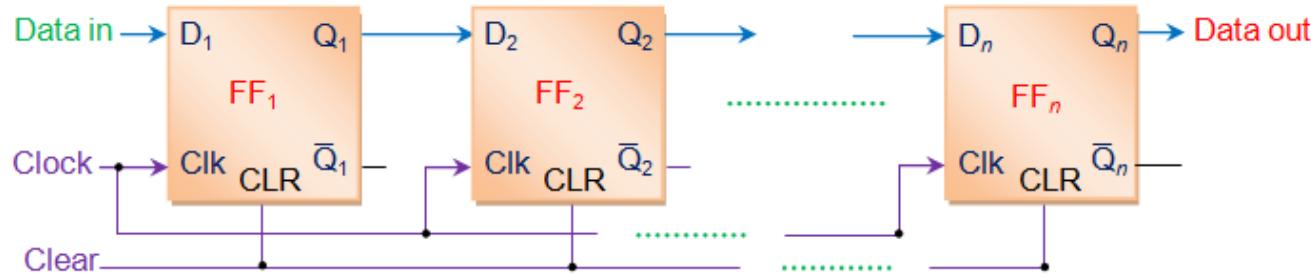


Figure 1 *n*-bit Right-Shift Serial-in Serial-Out Shift Register

- **Serial In Serial Out (SISO) shift registers** are a kind of shift registers where both data loading as well as data retrieval to/from the [shift register](#) occurs in serial-mode. Figure 1 shows a  $n$ -bit synchronous **SISO shift register** sensitive to positive edge of the clock pulse.
- The data word which is to be stored is fed bit-by-bit at the input of the first [flip-flop](#).
- It is seen that the inputs of all other flip-flops (except the first flip-flop  $FF_1$ ) are driven by the outputs of the preceding ones say for example, the input of  $FF_2$  is driven by the output of  $FF_1$ . At last the data stored within the register is obtained at the output pin of the  $n^{\text{th}}$  flip-flop in serial-fashion.
- Initially all the flip-flops in the register are cleared by applying high on their clear pins. Next the input data word is fed serially to  $FF_1$ .

# SISO Continued...

- This causes the bit appearing at the  $D_1$  pin ( $B_1$ ) to be stored into  $FF_1$  as soon as the first leading edge of the clock appears. Further at the second clock tick,  $B_1$  gets stored into  $FF_2$  while a new bit enters into  $FF_1$  ( $B_2$ ).
- This kind of shift in data bits continues for every rising edge of the clock pulse. This indicates that for every single clock pulse the data within the register moves towards right by a single bit. Thus the design shown in Figure 1 is regarded as a right-shift **SISO shift register**.
- Example** let the 4 bit binary number 1101 be entered into the register, beginning with the right most bit. (Note: Here  $QA$ ,  $QB$ ,  $QC$  and  $QD$  is  $Q1, Q2, Q3, Q4$ )  
**Reference:** Digital Circuits Design – S Salivahanan S Arivazhagan

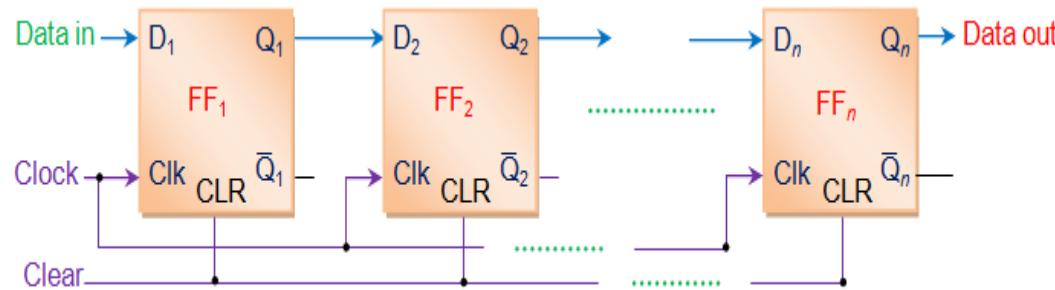
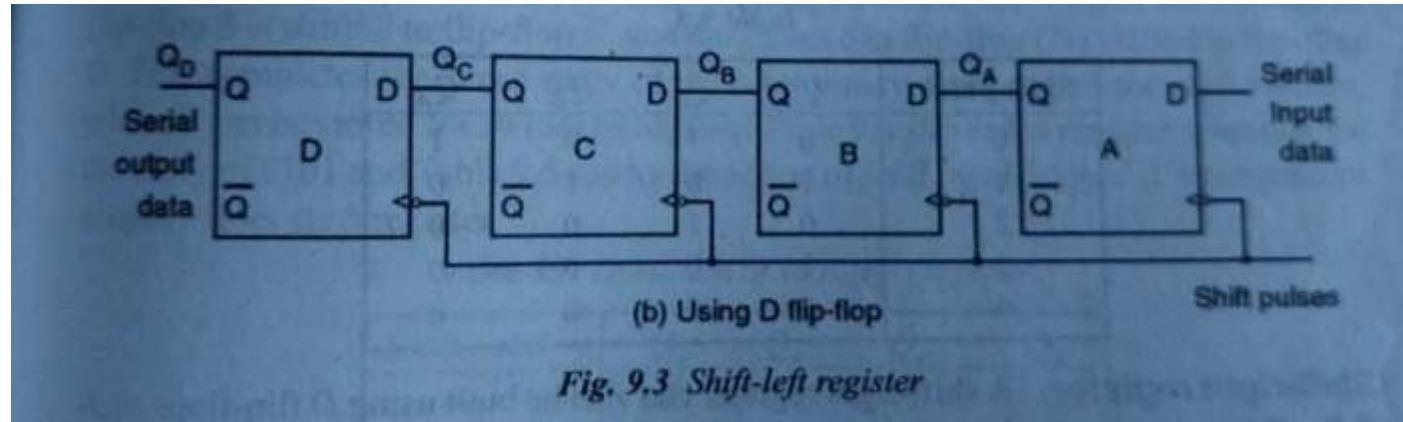


Figure 1 *n*-bit Right-Shift Serial-in Serial-Out Shift Register

Shift pulse	$Q_4$	$Q_3$	$Q_2$	$Q_1$
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1

## Serial left shift register

Reference: Digital Circuits Design – S Salivahanan S Arivazhagan

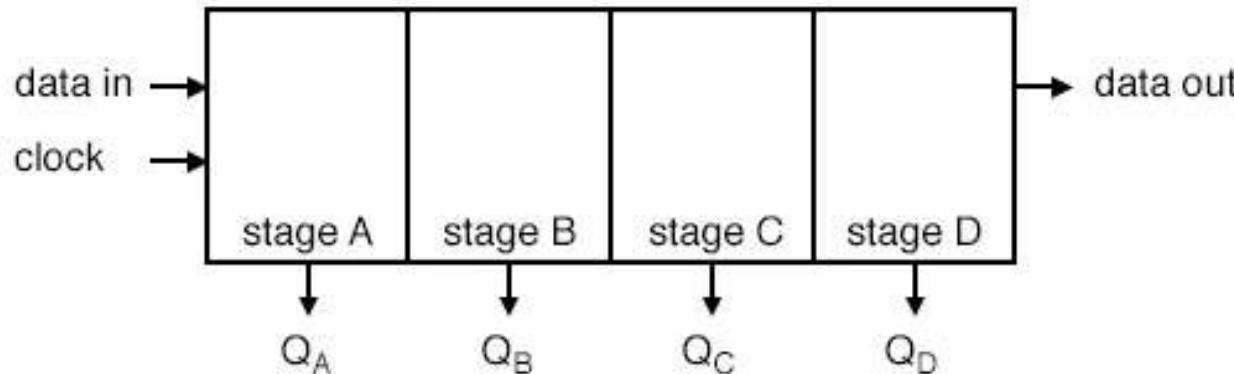


**Table 9.1** Operation of shift-left register

Shift pulse	$Q_D$	$Q_C$	$Q_B$	$Q_A$
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1

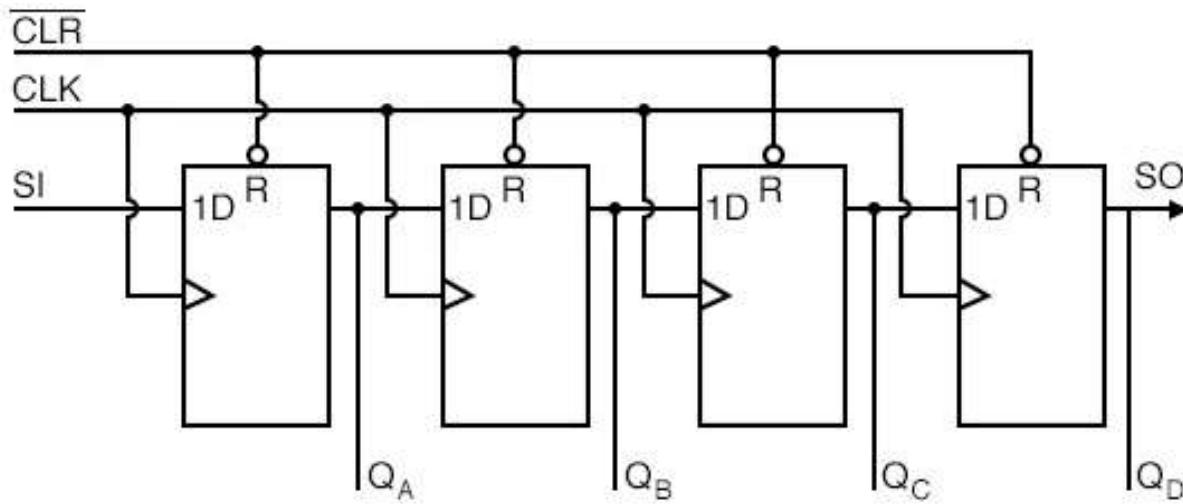
# SIFO

Reference:<https://www.allaboutcircuits.com/textbook/digital/chpt-12/serial-in-parallel-out-shift-register/>



Serial-in, parallel-out shift register with 4-stages

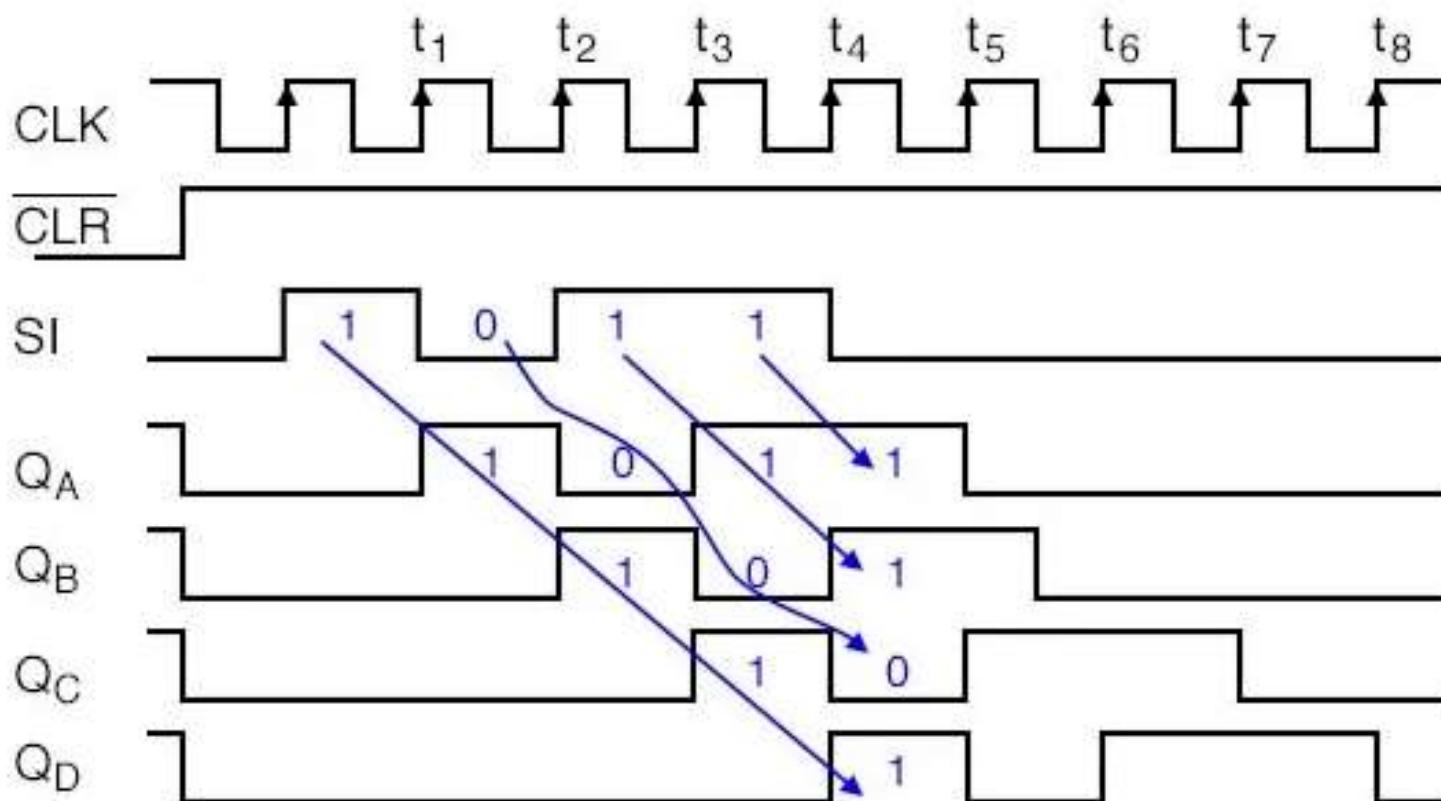
- If four data bits are shifted in by four clock pulses via a single wire at data-in, below, the data becomes available simultaneously on the four Outputs  $Q_A$  to  $Q_D$  after the fourth clock pulse.
- The practical application of the serial-in, parallel-out shift register is to convert data from serial format on a single wire to parallel format on multiple wires.



Serial-in/ Parallel-out shift register details

- The above details of the serial-in, parallel-out shift register are fairly simple. It looks like a serial-in, serial-out shift register with taps added to each stage output.
- Serial data shifts in at **SI** (Serial Input). After a number of clocks equal to the number of stages, the first data bit in appears at **SO** ( $Q_D$ ) in the above figure.
- In general, there is no **SO** pin. The last stage ( $Q_D$  above) serves as **SO** and is cascaded to the next package if it exists.

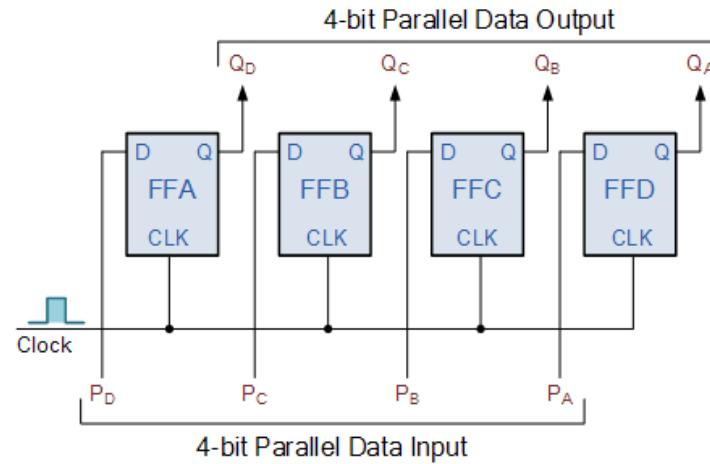
# SIPO Waveforms



Serial-in/ parallel-out shift register waveforms

# PIPO(Parallel in Parallel Out)

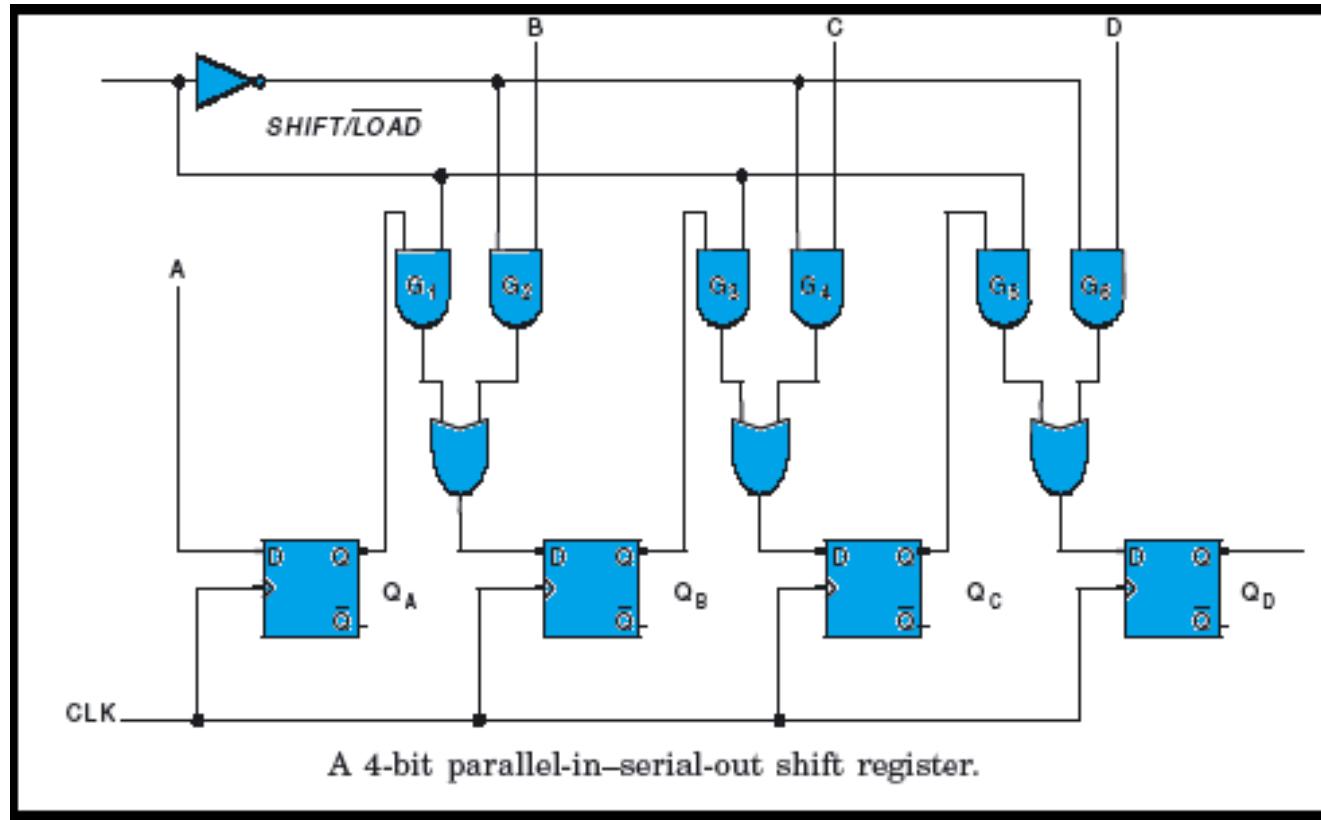
Reference: [https://www.electronics-tutorials.ws/sequential/seq\\_5.html](https://www.electronics-tutorials.ws/sequential/seq_5.html)



- This type of shift register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above.
- The data is presented in a parallel format to the parallel input pins P<sub>A</sub> to P<sub>D</sub> and then transferred together directly to their respective output pins Q<sub>A</sub> to Q<sub>D</sub> by the same clock pulse
- Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown.

# Parallel In - Serial Out Shift Registers

- The circuit uses D flip-flops and NAND gates for entering data (ie writing) to the register.
- D0, D1, D2 and D3 are the parallel inputs, where D0 is the most significant bit and D3 is the least significant bit.
- To **write** data in, the **mode control line is taken to LOW** and the data is clocked in.
- The data can be **shifted** when the **mode control line is HIGH** as SHIFT is active high.



	$Q_0$	$Q_1$	$Q_2$	$Q_3$	
Clear	0	0	0	0	
Write	1	0	0	1	
Shift	1	0	0	1	
	1	1	0	0	1
	1	1	1	0	01
	1	1	1	1	001
	1	1	1	1	1001

S-2

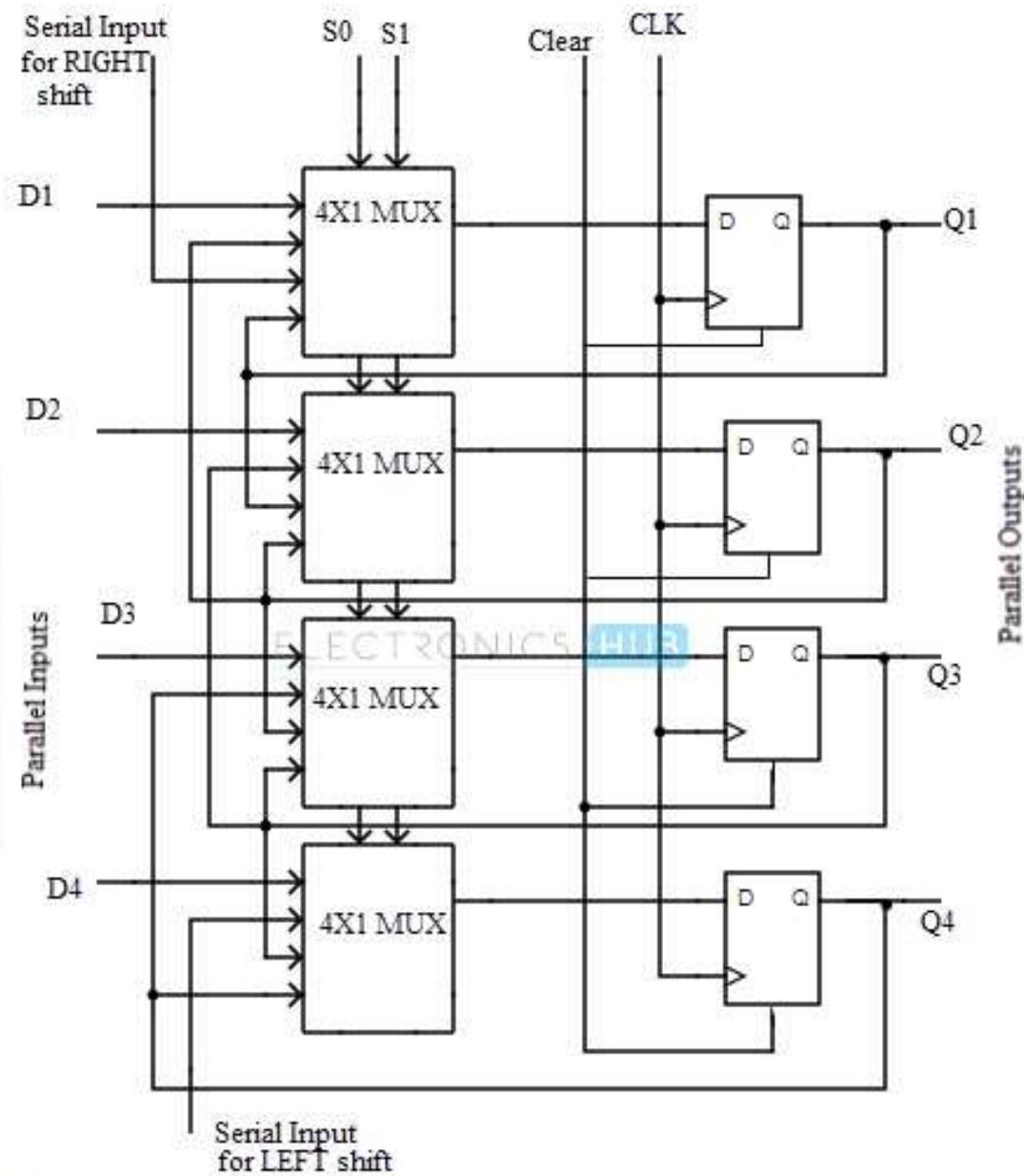
SLO-1      Universal Shift Register

SLO-2      Applications of Shift Registers

# Universal Shift Register

- The universal shift register can be defined as “The register which can be used to shift the data in both the directions like left, right and can load parallel data as well”.
- This register can perform three types of operations, stated below.
  - Parallel loading
  - Shifting left
  - Shifting right.

S0	S1	Operating Mode
0	0	Parallel Loading
1	0	Shift-Right
0	1	Shift-Left
1	1	Locked



# Application of Shift Register

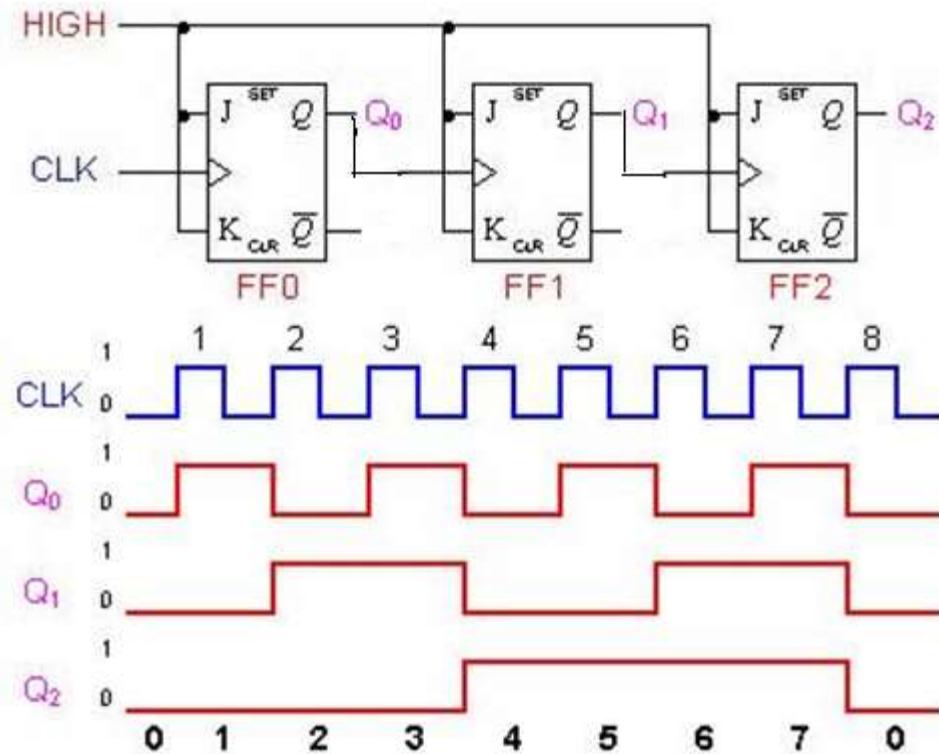
- Delay line
- Serial to parallel converter
- Parallel to serial converter
- Shift register counter
- Sequence generator

S-3

SLO-1  
SLO-2

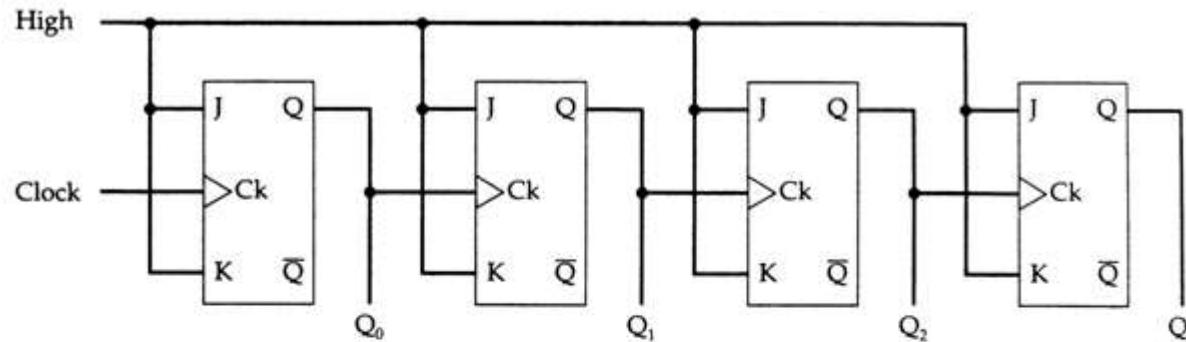
Synchronous Counters  
Asynchronous Counters

# 3-bit Asynchronous ripple up counter

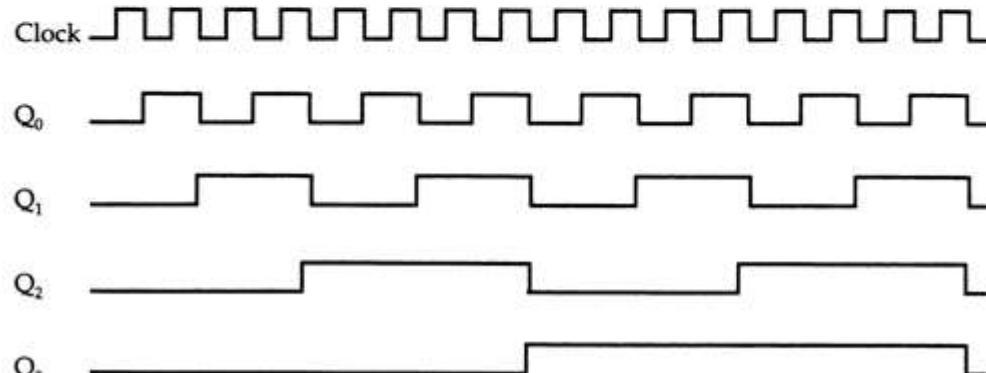


Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>0</sub>
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

# 4-bit Asynchronous ripple up counter



(a) Asynchronous counter



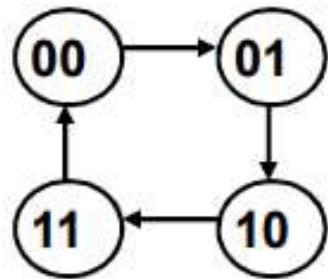
(b) Timing Diagram

Q3	Q2	Q1	Q0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

# Synchronous counters

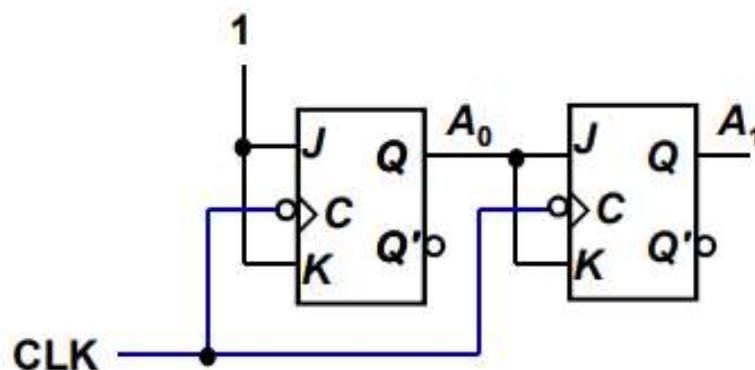
- Synchronous (parallel) counters: The flip-flops are clocked at the same time by a common clock pulse.
- Synchronous counters have been designed using the sequential logic design process.

Example: Design a 2-bit synchronous binary counter using T flip-flops



Present state		Next state		Flip-flop inputs	
$A_1$	$A_0$	$A_1^+$	$A_0^+$	$TA_1$	$TA_0$
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$$TA_1 = A_0$$
$$TA_0 = 1$$



## STEP 1 – Excitation Table

**Example: Design a 4-bit synchronous binary UP counter using JK flip-flops**

Present state				Next state				Flip Flop inputs							
QD	QC	QB	QA	QD	QC	QB	QA	JQD	KQD	JQC	KQC	JQB	KQB	JQA	KQA
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	1	1	1	1	X	0	X	0	X	0	1	X
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

## STEP 2 – K map

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	X	X	X	X

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	0	0	0	0

$$J_d = Q_a.Q_b.Q_c$$

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	0	0	1	0
01	X	X	X	X
11	X	X	X	X
10	0	0	1	0

$$J_c = Q_a.Q_b$$

$$K_d = Q_a.Q_b.Q_c$$

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	X	X	X	X
01	0	0	1	0
11	0	0	1	0
10	X	X	X	X

$$K_c = Q_a.Q_b$$

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	0	1	X	X
01	0	1	X	X
11	0	1	X	X
10	0	1	X	X

$J_b = Q_a$

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	X	X	1	0
01	X	X	1	0
11	X	X	1	0
10	X	X	1	0

$K_b = Q_a$

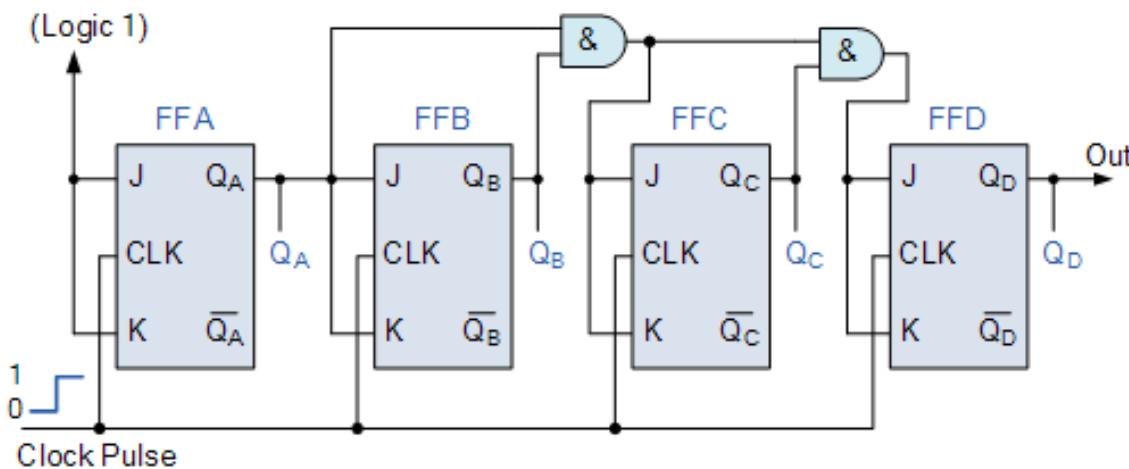
$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	1	1	1	1

$J_a = 1$

$Q_b, Q_a$ $Q_d, Q_c$	00	01	11	10
00	X	X	X	X
01	1	1	1	1
11	1	1	1	1
10	X	X	X	X

$K_a = 1$

### STEP 3 – Implementation



$$J_d = Q_a \cdot Q_b \cdot Q_c$$

$$K_d = Q_a \cdot Q_b \cdot Q_c$$

$$J_c = Q_a \cdot Q_b$$

$$K_c = Q_a \cdot Q_b$$

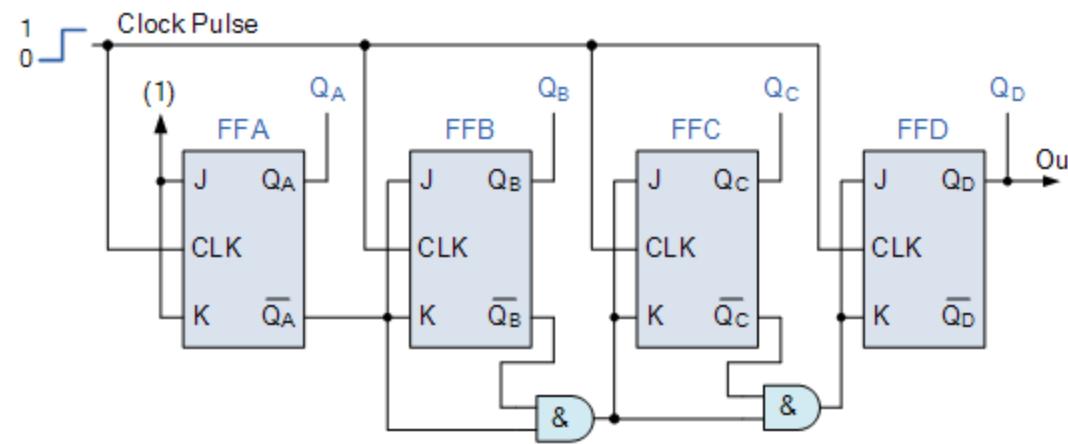
$$J_b = Q_a$$

$$K_b = Q_a$$

$$J_a = 1$$

$$K_a = 1$$

# 4-bit synchronous binary Down counter using JK flip-flops



# Up/Down Synchronous Counters

- Up/down synchronous counter: a bidirectional counter that is capable of counting either up or down.
- An input (control) line Up/Down (or simply Up) specifies the direction of counting.
- Up/Down = 1 Count upward Up/Down = 0 then Count downward

## Eg: 3-bit Up/Down Synchronous Counters

Clock pulse	Up	$Q_2$	$Q_1$	$Q_0$	Down
0		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	

$$TQ_0 = 1$$

$$TQ_1 = (Q_0 \cdot Up) + (Q_0' \cdot Up')$$

$$TQ_2 = (Q_0 \cdot Q_1 \cdot Up) + (Q_0' \cdot Q_1' \cdot Up')$$

Up counter

$$TQ_0 = 1$$

$$TQ_1 = Q_0$$

$$TQ_2 = Q_0 \cdot Q_1$$

Down counter

$$TQ_0 = 1$$

$$TQ_1 = Q_0'$$

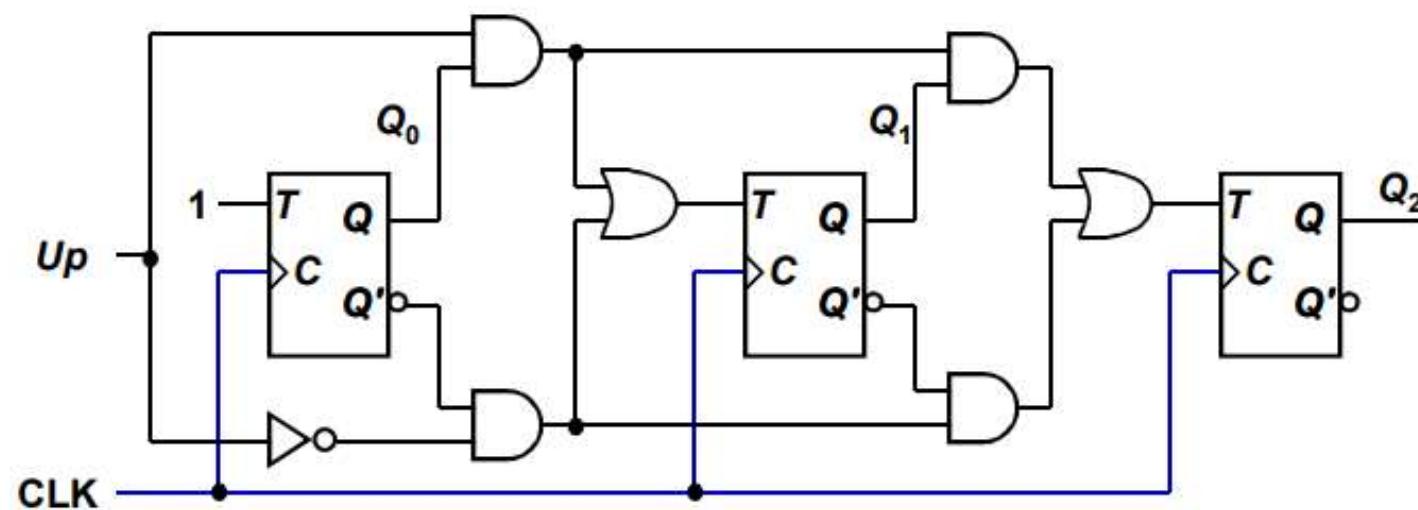
$$TQ_2 = Q_0' \cdot Q_1'$$

## Eg: 3-bit Up/Down Synchronous Counters

$$TQ_0 = '1$$

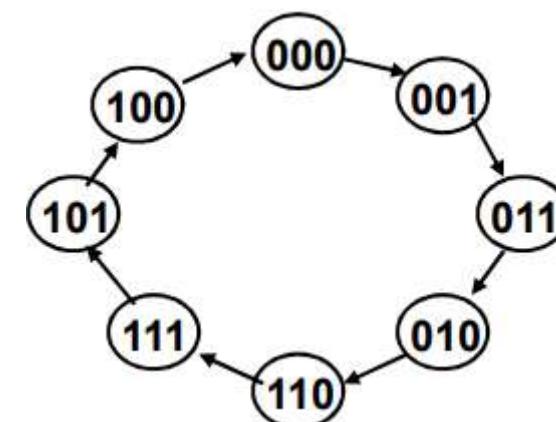
$$TQ_1 = (Q_0 \cdot Up) + (Q_0' \cdot Up')$$

$$TQ_2 = (Q_0 \cdot Q_1 \cdot Up) + (Q_0' \cdot Q_1' \cdot Up')$$



## Eg: 3 bit Gray code counter using JK flipflop

Present state			Next state			Flip-flop inputs					
$Q_2$	$Q_1$	$Q_0$	$Q_2^+$	$Q_1^+$	$Q_0^+$	$JQ_2$	$KQ_2$	$JQ_1$	$KQ_1$	$JQ_0$	$KQ_0$
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	1	0	X	1	X	X	0
0	1	0	1	1	0	1	X	X	0	0	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	0	0	X	1	0	X	0	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	1	0	1	X	0	X	1	X	0



Eg: 3 bit Gray code counter using JK flipflop

■ 3-bit Gray code counter: flip-flop inputs.

		$Q_1 Q_0$	00	01	11	10
		0	0			1
$Q_2$	1	X	X	X	X	

$JQ_2 = Q_1 \cdot Q_0'$

		$Q_1 Q_0$	00	01	11	10
		0	0	1	X	X
$Q_2$	1			X	X	

$JQ_1 = Q_2' \cdot Q_0$

		$Q_1 Q_0$	00	01	11	10
		0	1	X	X	
$Q_2$	1		X	X	1	

$JQ_0 = Q_2 \cdot Q_1 + Q_2' \cdot Q_1'$   
 $= (Q_2 \oplus Q_1)'$

		$Q_1 Q_0$	00	01	11	10
		0	X	X	X	X
$Q_2$	1	1				

$KQ_2 = Q_1' \cdot Q_0'$

		$Q_1 Q_0$	00	01	11	10
		0	X	X		
$Q_2$	1	X	X	1		

$KQ_1 = Q_2 \cdot Q_0$

		$Q_1 Q_0$	00	01	11	10
		0	X		1	X
$Q_2$	1	X	1		X	

$KQ_0 = Q_2 \cdot Q_1' + Q_2' \cdot Q_1$   
 $= Q_2 \oplus Q_1$

Eg: 3 bit Gray code counter using JK flipflop

3-bit Gray code counter: logic diagram.

$$JQ_2 = Q_1 \cdot Q_0'$$

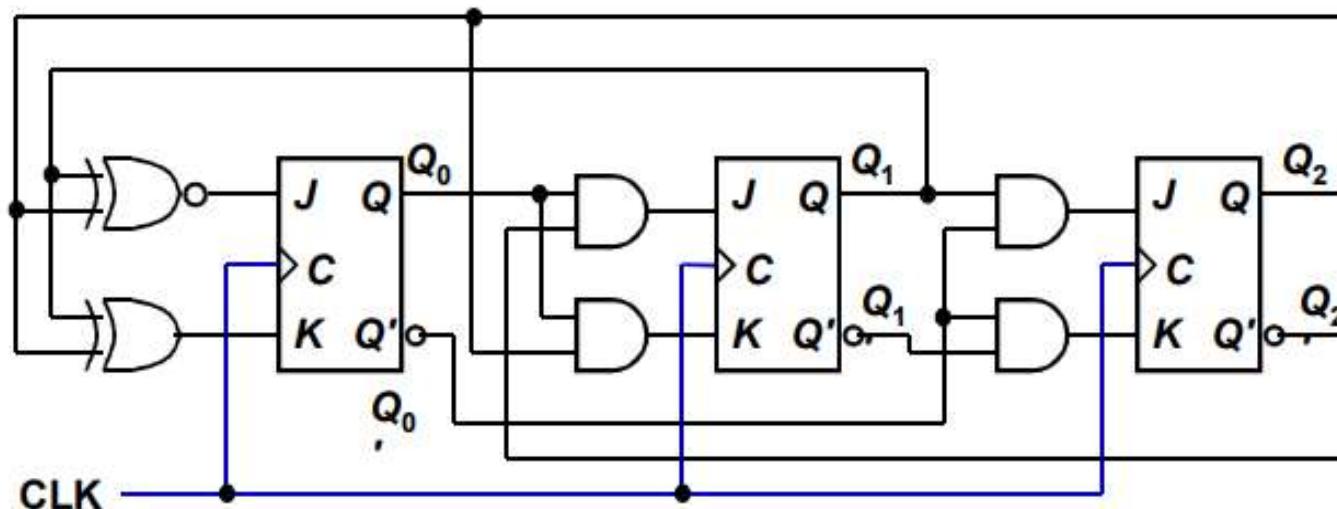
$$KQ_2 = Q_1' \cdot Q_0'$$

$$JQ_1 = Q_2' \cdot Q_0$$

$$KQ_1 = Q_2 \cdot Q_0$$

$$JQ_0 = (Q_2 \oplus Q_1)'$$

$$KQ_0 = Q_2 \oplus Q_1$$



S-6

SLO-1

Changing the Counter Modulus

SLO-2

Decade Counters

# Counter

- Counters are sequential logic devices that are activated or triggered by an external timing pulse or clock signal.
- A counter can be constructed to operate as a synchronous circuit or as an asynchronous circuit.
- With synchronous counters, all the data bits change synchronously with the application of a clock signal.
- Whereas an asynchronous counter circuit is independent of the input clock so the data bits change state at different times one after the other.
- The number of states or counting sequences through which a particular counter advances before returning once again back to its original first state is called the **modulus (MOD)**.
- **In other words, the modulus (or just modulo) is the number of states the counter counts and is the dividing number of the counter.**

•

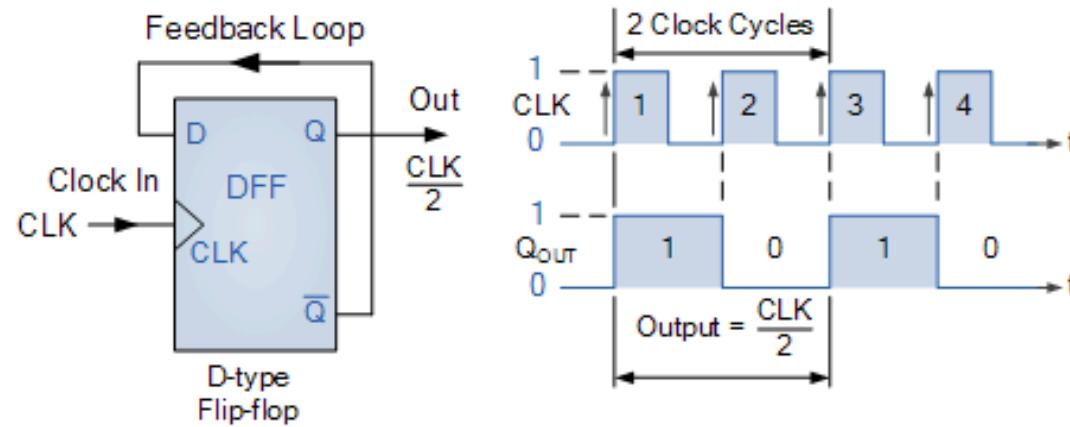
# Synchronous Mod Counter

# Modulus Counter

- **Modulus Counters**, or simply *MOD counters*, are defined based on the number of states that the counter will sequence through before returning back to its original value.
- For example, a 2-bit counter that counts from  $00_2$  to  $11_2$  in binary, that is 0 to 3 in decimal, has a modulus value of 4 ( $00 \rightarrow 1 \rightarrow 10 \rightarrow 11$ , and return back to 00) so would therefore be called a modulo-4, or mod-4, counter.
- Counters can be designed to count to any number of  $2^n$  states in their sequence by cascading together multiple counting stages to produce a single modulus or MOD-N counter.
- Therefore, a “Mod-N” counter will require “N” number of flip-flops connected together to count a single data bit while providing  $2^n$  different output states, (n is the number of bits).

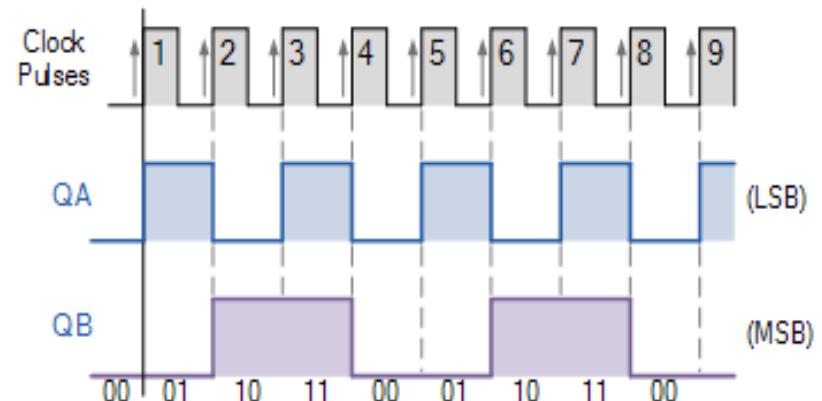
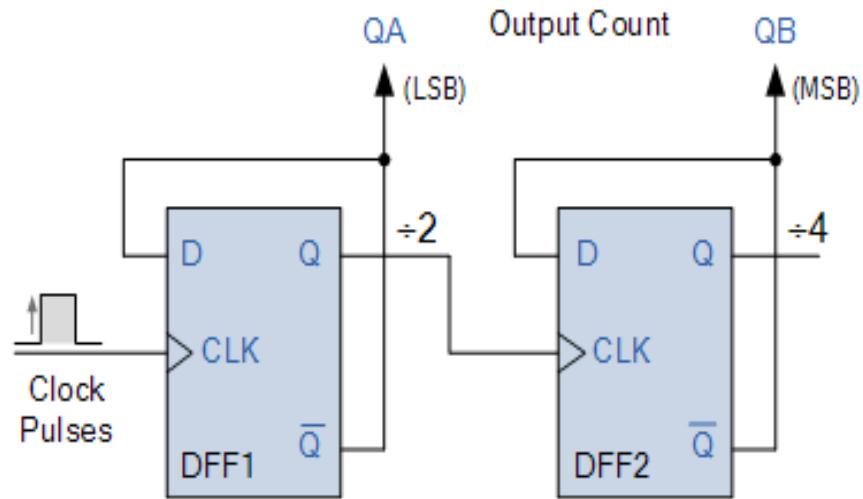
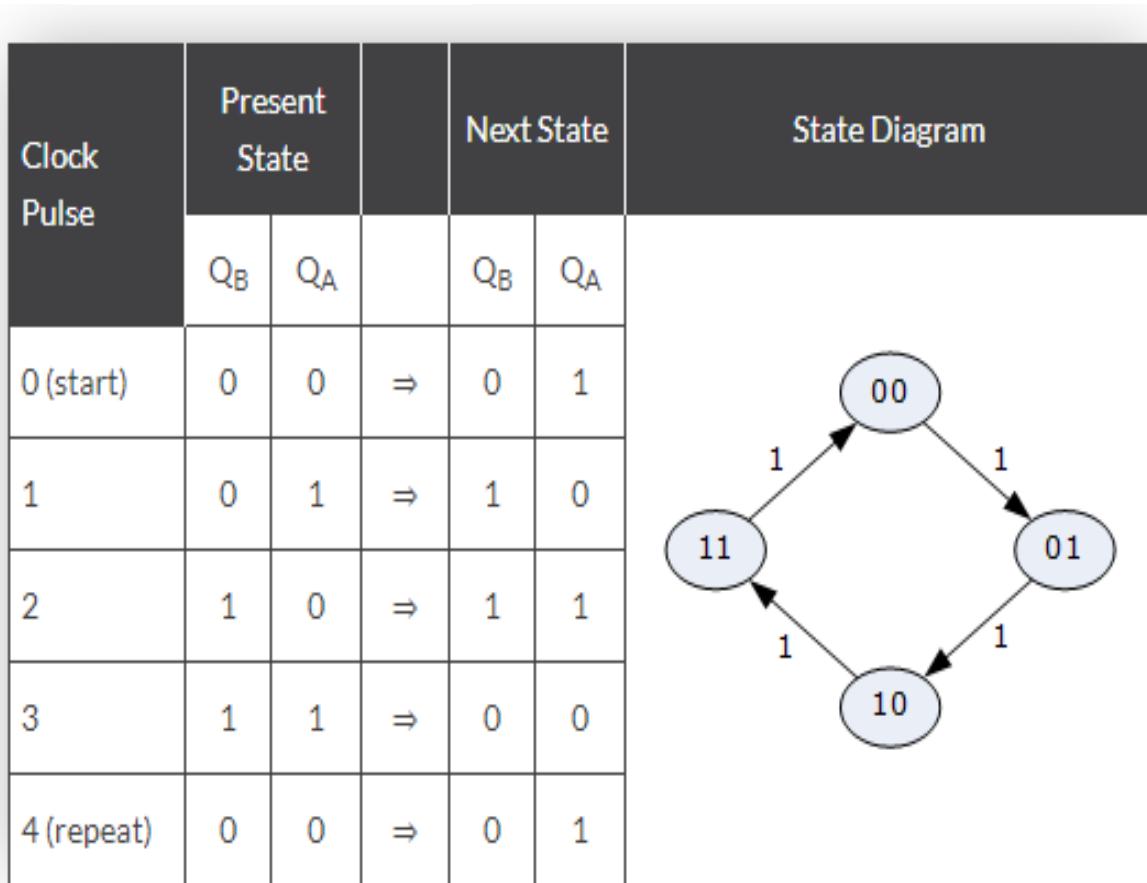
# Mod 2 Counter

- The edge-triggered D-type flip-flop is a useful and versatile building block to construct a MOD counter or any other type of sequential logic circuit.
- By connecting the Q output back to the “D” input as shown, and creating a feedback loop, we can convert it into a binary divide-by-two counter using the clock input only as the Q output signal is always the inverse of the Q output signal.

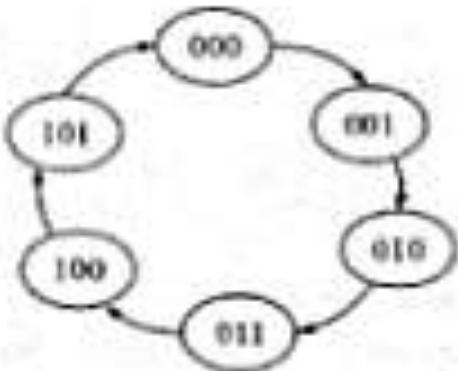


# Mod 4 Counter

Outputs		Input
$Q_n$	$Q_{n+1}$	D
0	0	0
0	1	1
1	0	0
1	1	1



# Mod 6 Counter



Q Output		Inputs	
Present State	Next State	$J_n$	$K_n$
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

$C_n$  \  $B_n A_n$

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

$J_C = B_n A_n$

$C_n$  \  $B_n A_n$

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

$K_C = A_n$

$C_n$  \  $B_n A_n$

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

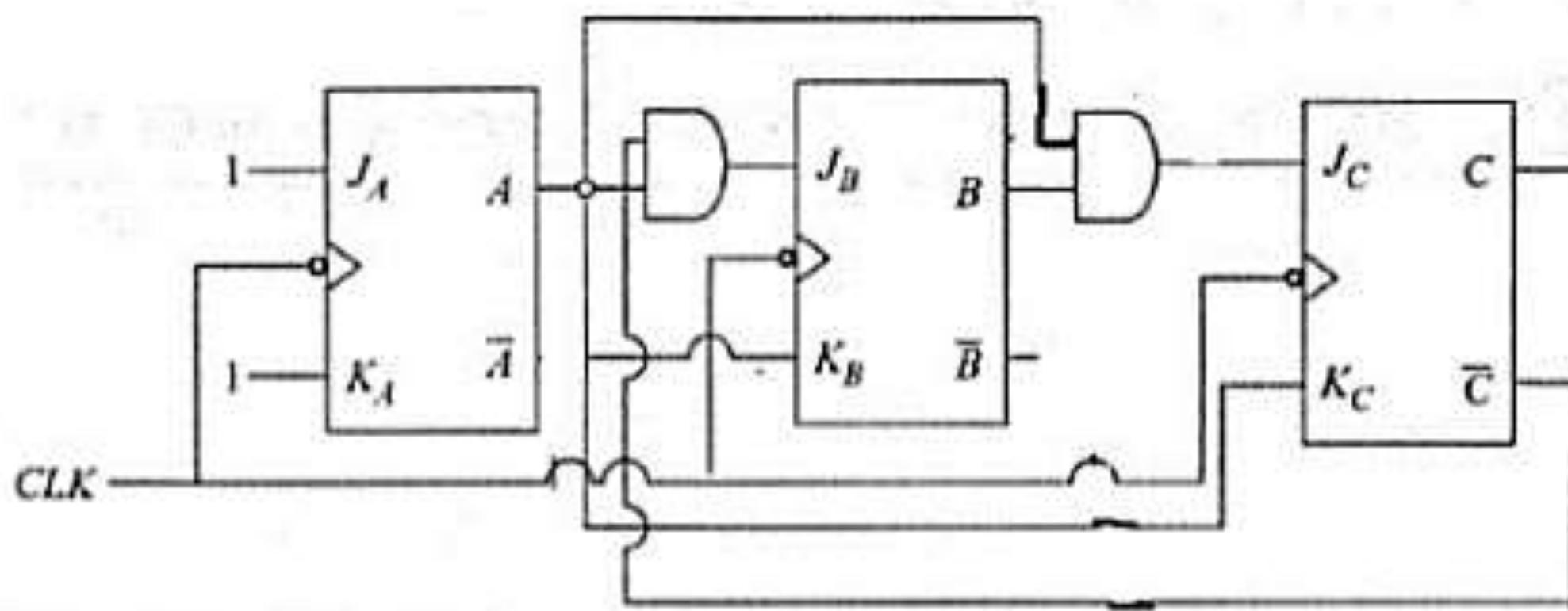
$J_B = \bar{C}_n A_n$

$K_B = A_n$

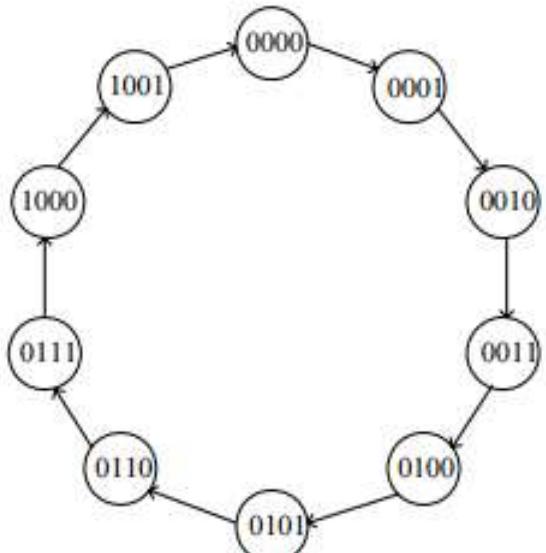
$C_n$	$B_n$	$A_n$	$C_{n+1}$	$B_{n+1}$	$A_{n+1}$	$J_C$	$K_C$	$J_B$	$K_B$	$J_A$	$K_A$
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	0	0	0	x	1	0	x	x	1
1	1	0	x	x	x	x	x	x	x	x	x
1	1	1	x	x	x	x	x	x	x	x	x

$J_A=1$  and  $K_A=1$

# Mod 6 Counter



# Decade Counter (Mod 10 Counter)



Q Output		Inputs	
Present State	Next State	$J_n$	$K_n$
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Present State				Next State				Output							
$Q_3$	$Q_2$	$Q_1$	$Q_0$	$Q_3$	$Q_2$	$Q_1$	$Q_0$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$	$J_0$	$K_0$
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1

# Decade Counter (Mod 10 Counter)

$Q_3 Q_2$	00	01	11	10	
$Q_1 Q_0$	00	1	X	X	1
00	1	X	X	1	
01	X	1	1	X	
11	X	X	X	X	
10	1	X	X	X	

$$J_0 = 1$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00	1	1	X
00	X	1	1	X
01	X	1	1	X
11	X	X	X	X
10	X	1	X	X

$$K_0 = 1$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00		1	
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10			X	X

$$J_2 = Q_1 Q_0$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00		X	X
00	X	X	1	
01			X	X
11	X	X	X	X
10			X	X

$$K_2 = Q_1 Q_0$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00	1	X	X
00	1	X	X	X
01			X	X
11	X	X	X	X
10			X	X

$$J_1 = \overline{Q}_3 Q_0$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00	X	1	
00	X	1	1	
01	X	X	1	
11	X	X	X	X
10	X	X	X	X

$$K_1 = \overline{Q}_3 Q_0$$

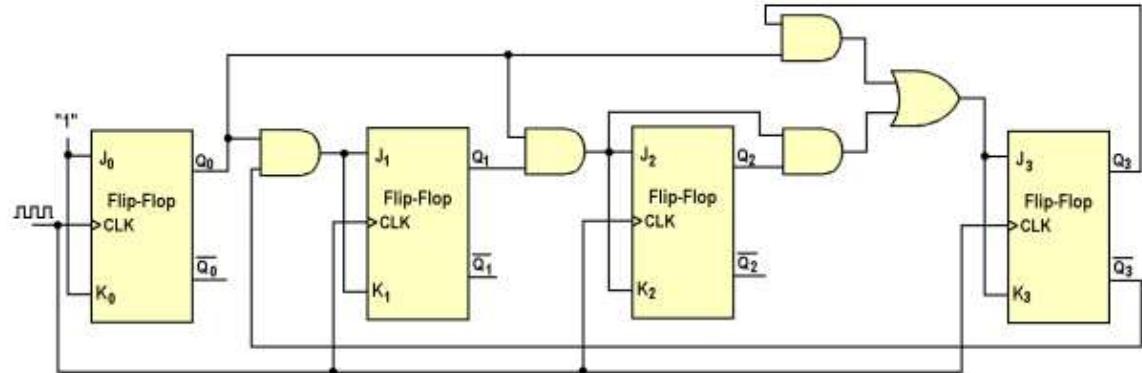
$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00			
00			1	
01			X	X
11	X	X	X	X
10	X	X	X	X

$$J_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

$Q_3 Q_2$	00	01	11	10
$Q_1 Q_0$	00		X	X
00	X	X	1	
01	X	X	X	X
11	X	X	X	X
10	1	X	X	X

$$K_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

# Decade Counter (Mod 10 Counter)



$$J_0 = 1$$

$$J_2 = Q_1 Q_0$$

$$J_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

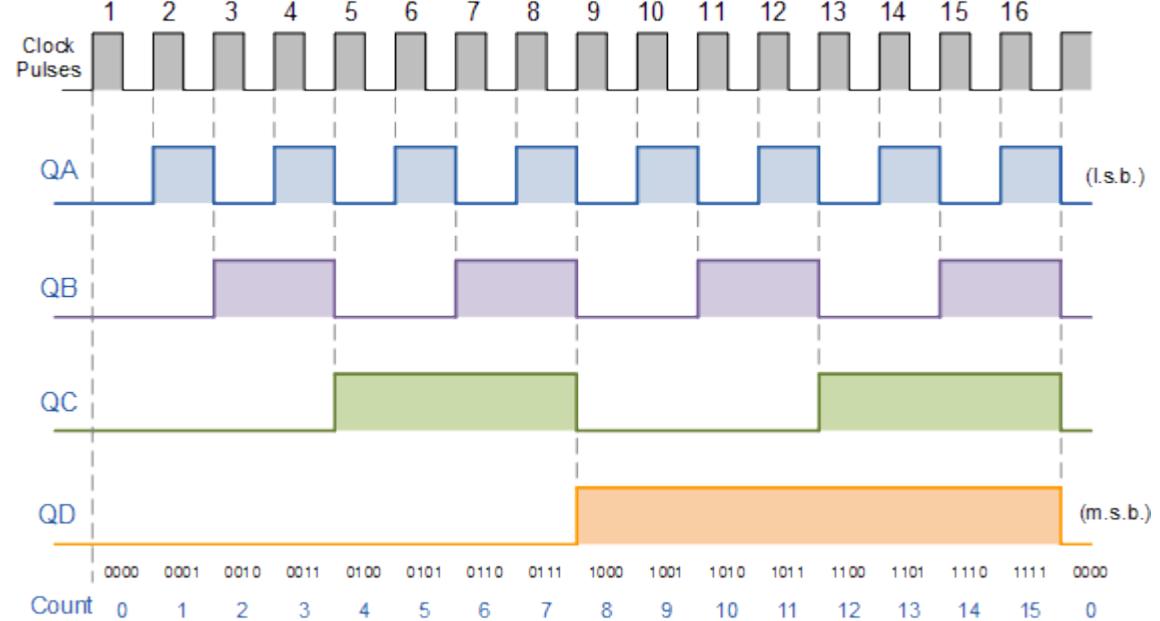
$$J_1 = \overline{Q}_3 Q_0$$

$$K_0 = 1$$

$$K_2 = Q_1 Q_0$$

$$K_3 = Q_3 Q_0 + Q_2 Q_1 Q_0$$

$$K_1 = \overline{Q}_3 Q_0$$

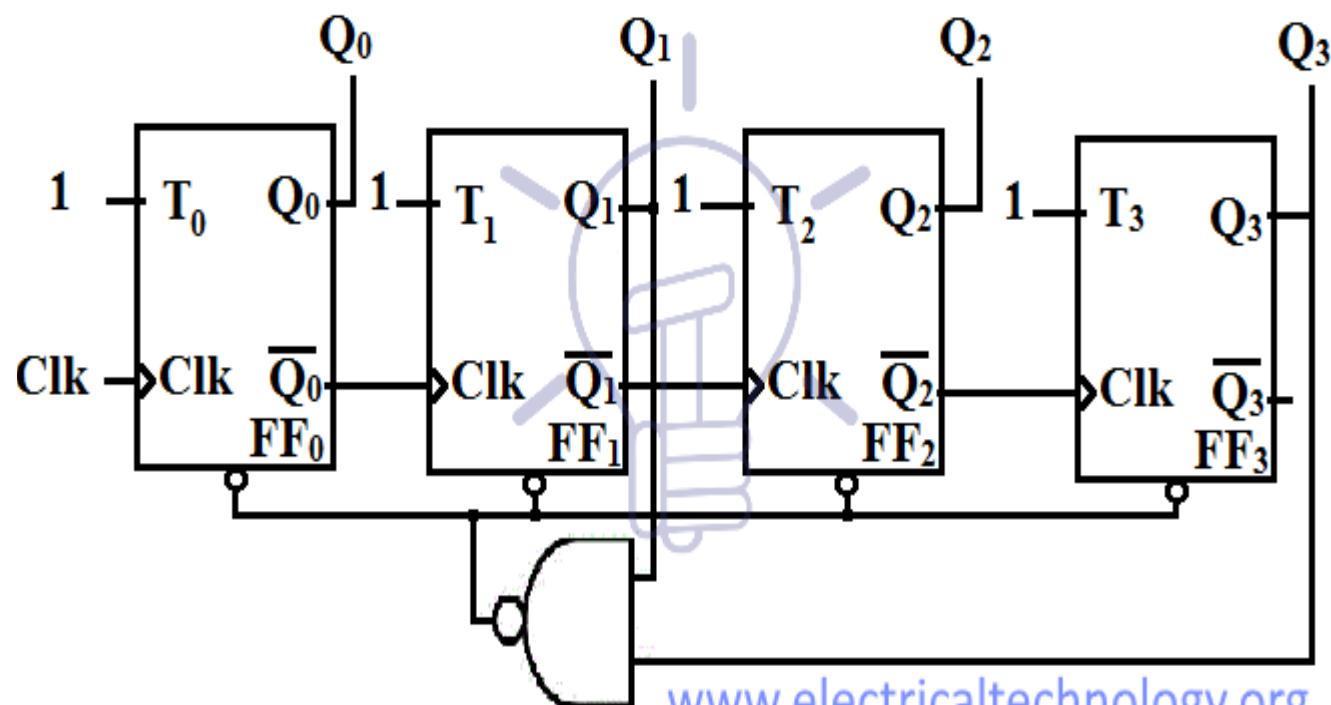


# Asynchronous Mod Counter

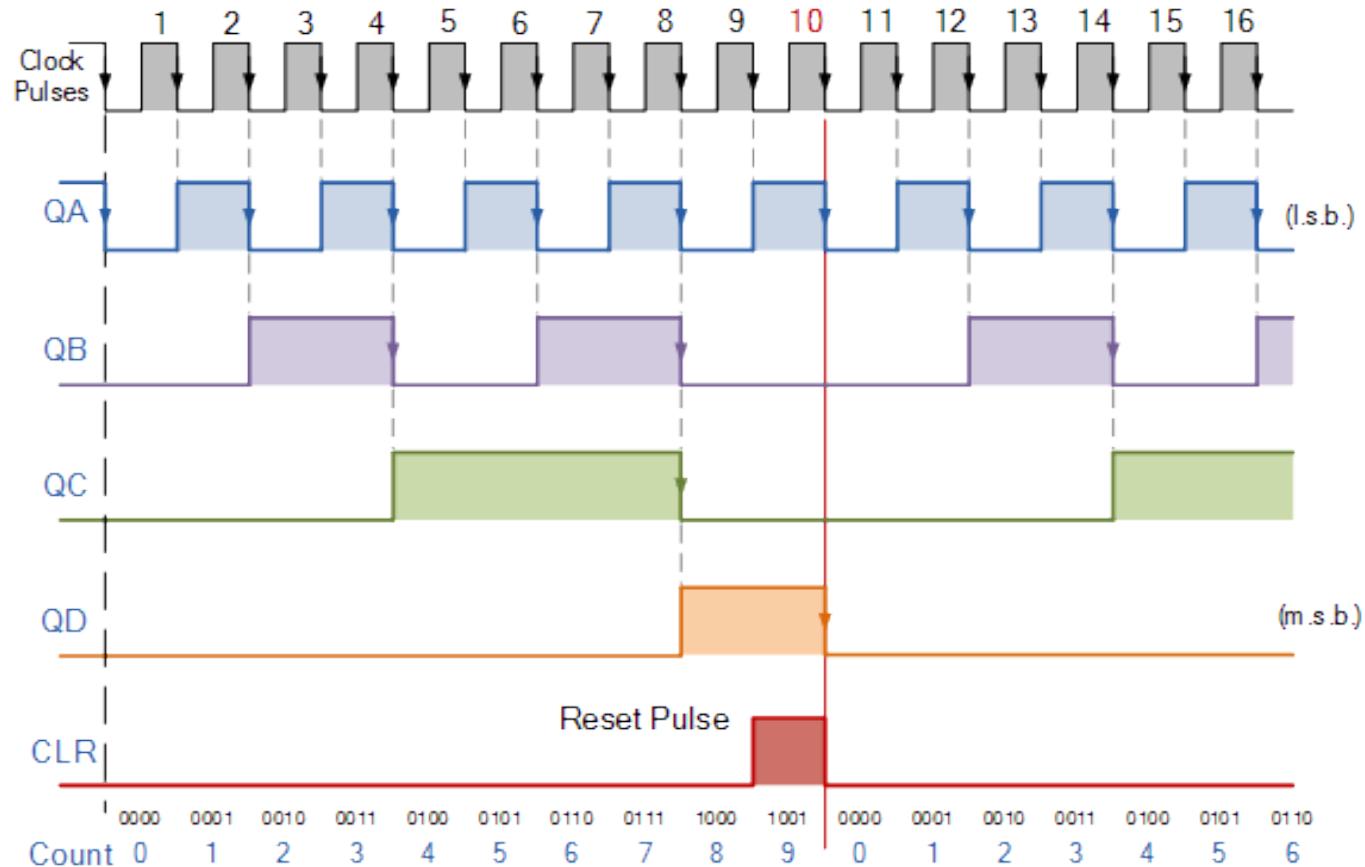
# Asynchronous Decade Counter

State	$Q_3$	$Q_2$	$Q_1$	$Q_0$
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
10 (reset)	1	0	1	0

*Reset to 0*

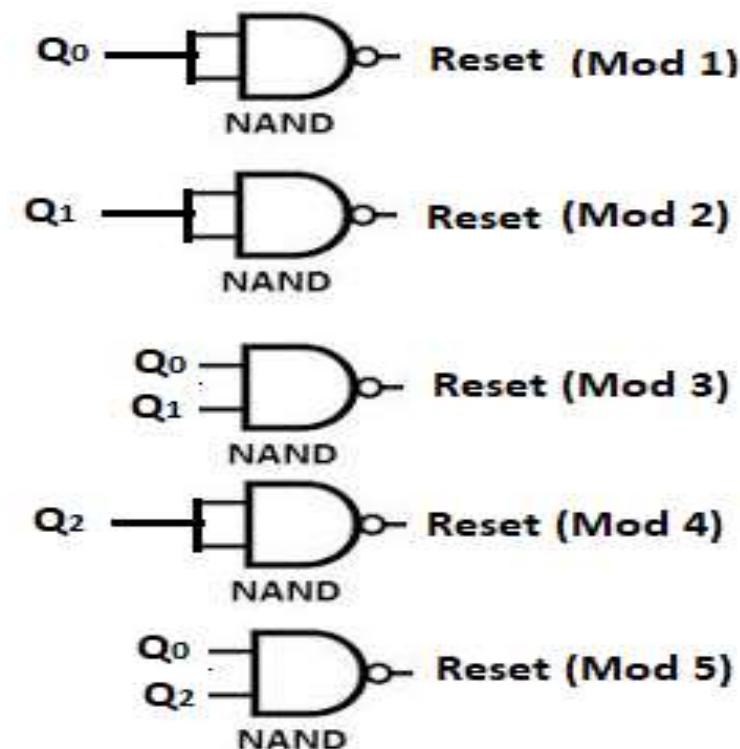


## Decade Counter Timing Diagram



# Asynchronous Modulus Counter

Mod	$Q_3$	$Q_2$	$Q_1$	$Q_0$
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
10	1	0	1	0



Similarly other Mod counters can be obtained.

**S-7**

**SLO-1**

**Presettable counters**

**SLO-2**

**Counter Design as a Synthesis problem**

**S-7**

**SLO-1**

**Presettable counters**

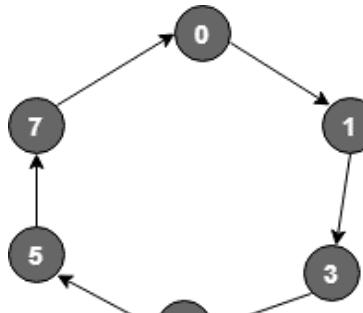
**SLO-2**

**Counter Design as a Synthesis problem**

# Counter design for given sequence

- **Problem** – Design synchronous counter for sequence:  $0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 0$ , using T flip-flop.
- **Understanding the problem-**

PRESENT STATE	NEXT STATE
0	1
1	3
3	4
4	5
5	7
7	0



- Count 7 is represented by 111, therefore 3 flip flops are required for the design

- State transition table and inputs to the flip flop for given sequence:

PRESENT STATE			NEXT STATE			INPUT TABLE OF FLIP-FLOPS		
$Q_3$	$Q_2$	$Q_1$	$Q_3(t+1)$	$Q_2(t+1)$	$Q_1(t+1)$	$T_3$	$T_2$	$T_1$
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	1	0	1	0
1	1	1	0	0	0	1	1	1

- Write the input table of all T flip-flops by using the excitation table of T flip-flop.
- Excitation table of T- flip flop

$Q_T$	$Q_{T+1}$	T
0	0	0
0	1	1
1	0	1
1	1	0

- Find value of T<sub>3</sub>, T<sub>2</sub>, T<sub>1</sub> in terms of Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub> using K-Map (Karnaugh Map):

$T_3$ $Q_3 \setminus Q_2 Q_1$	00	01	11	10
0	0	0	1	X
1	0	0	1	X

$$T_3 = Q_2$$

$T_2$ $Q_3 \setminus Q_2 Q_1$	00	01	11	10
0	0	1	1	X
1	0	1	1	X

$$T_2 = Q_1$$

$T_1$ $Q_3 \setminus Q_2 Q_1$	00	01	11	10
0	1	1	0	X
1	1	1	0	X

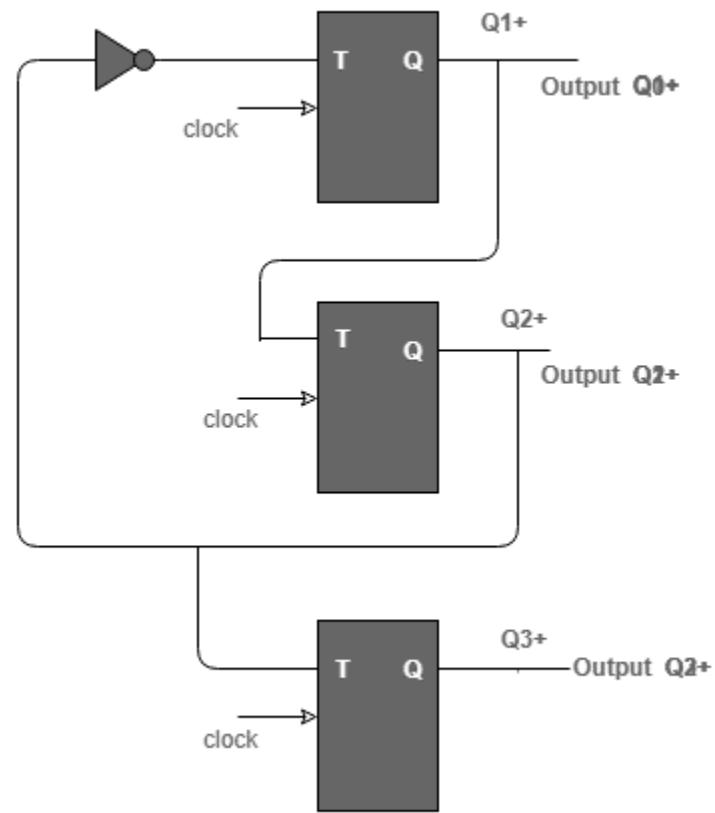
$$T_1 = Q_2'$$

Implementation of the design

$$T_1 = Q_2'$$

$$T_2 = Q_1 + Q_2$$

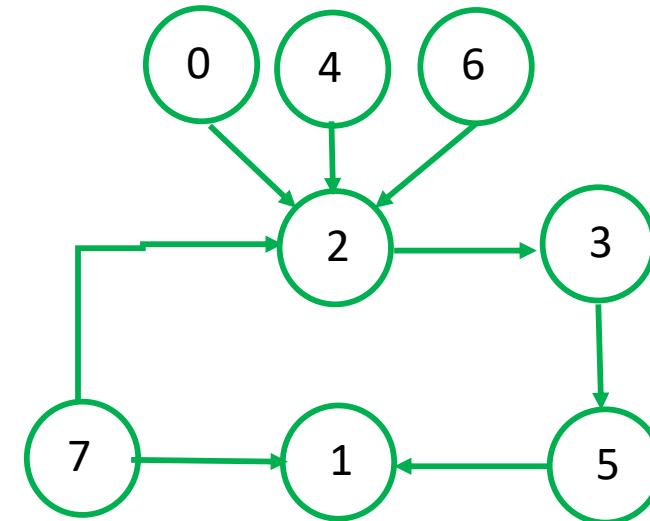
$$T_3 = Q_2$$



# Counter design for given sequence considering unused states

- **Problem** – Design a mod -5 counter for sequence:  $2 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 2$ , using D flip-flop. The counter must be self starting with count states 0,4,6 leading directly to 2.
- **Understanding the problem-**

PRESENT STATE	NEXT STATE
2	3
3	5
5	1
1	7
7	2
0	2
4	2
6	2



- Mod 5 counter requires 5 unique states, therefore 3 flip flops are required for the design

- State transition table and inputs to the flip flop for given sequence:

PRESENT STATE			NEXT STATE			INPUT TABLE OF FLIP-FLOPS		
$Q_3$	$Q_2$	$Q_1$	$Q_3(t+1)$	$Q_2(t+1)$	$Q_1(t+1)$	$D_3$	$D_2$	$D_1$
0	0	0	0	1	0	0	1	0
0	0	1	1	1	1	1	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0	1
1	1	0	0	1	0	0	1	0
1	1	1	0	1	0	0	1	0

- Write the input table of all T flip-flops by using the excitation table of D flip-flop.
- Excitation table of D- flip flop

$Q_T$	$Q(T+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

- Find value of D3, D2, D1 in terms of Q3, Q2, Q1 using K-Map (Karnaugh Map):

D3 Q3\Q2Q1	00	01	11	10
0	0	1	1	0
1	0	0	0	0

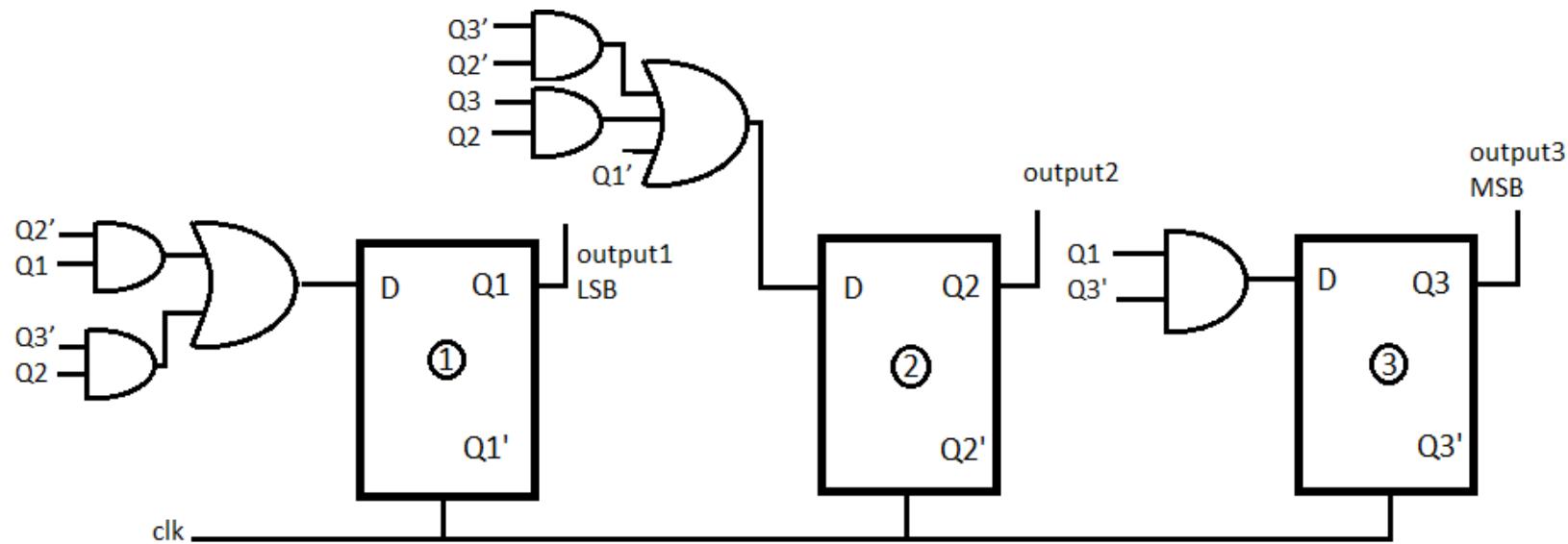
$$\begin{aligned}D3 &= Q3'Q1 \\D2 &= Q1' + Q3'Q2' + Q3Q2 \\D1 &= Q2'Q1 + Q3'Q2\end{aligned}$$

D2 Q3\Q2Q1	00	01	11	10
0	1	1	0	1
1	1	0	1	1

D1 Q3\Q2Q1	00	01	11	10
0	0	1	1	1
1	0	1	0	0

- Implementation of the design

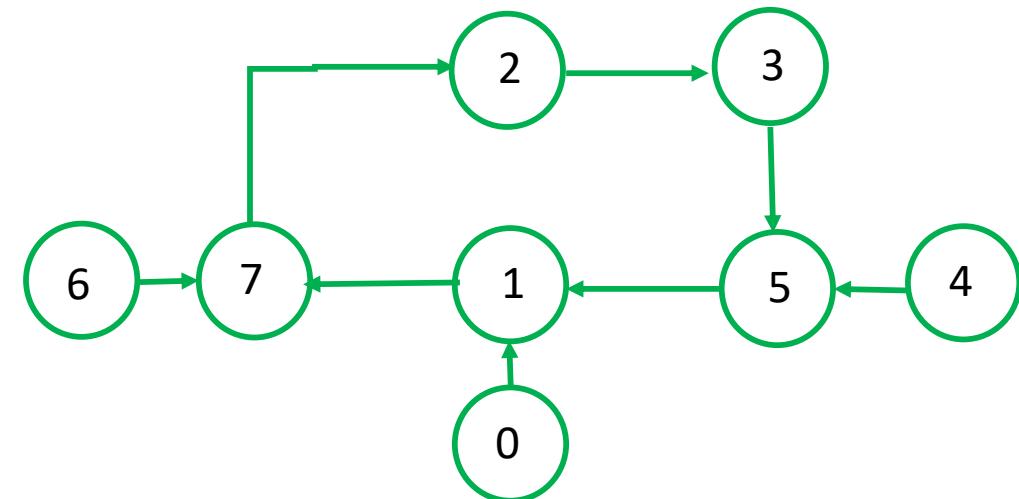
- $D_3 = Q_3'Q_1$
- $D_2 = Q_1' + Q_3'Q_2' + Q_3Q_2$
- $D_1 = Q_2'Q_1 + Q_3'Q_2$



# Counter design for given sequence considering unused states

- **Problem** – Design a mod -5 counter for sequence:  $2 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 2$ , using D flip-flop. The counter must be self starting with count states 0 to 1, 4 to 5, and 6 to 7.
- **Understanding the problem-**

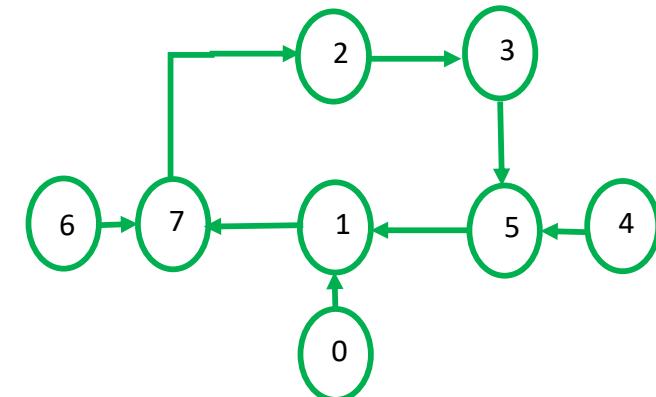
PRESENT STATE	NEXT STATE
2	3
3	5
5	1
1	7
7	2
0	1
4	5
6	7



- Mod 5 counter requires 5 unique states, therefore 3 flip flops are required for the design

- State transition table and inputs to the flip flop for given sequence:

PRESENT STATE			NEXT STATE			INPUT TABLE OF FLIP-FLOPS		
$Q_3$	$Q_2$	$Q_1$	$Q_3(t+1)$	$Q_2(t+1)$	$Q_1(t+1)$	$D_3$	$D_2$	$D_1$
0	0	0	0	0	1	0	0	1
0	0	1	1	1	1	1	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	0	1	0	0	1	0	0	1
1	1	0	1	1	1	1	1	1
1	1	1	0	1	0	0	1	0



- Write the input table of all T flip-flops by using the excitation table of D flip-flop.
- Excitation table of D- flip flop

$Q_T$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

- Find value of D<sub>3</sub>, D<sub>2</sub>, D<sub>1</sub> in terms of Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub> using K-Map (Karnaugh Map):

D <sub>3</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$$D_3 = Q_3'Q_1 + Q_3Q_1'$$

$$D_2 = Q_2Q_1' + Q_3'Q_2'Q_1 + Q_3Q_2$$

$$D_1 = Q_1' + Q_3' + Q_2'$$

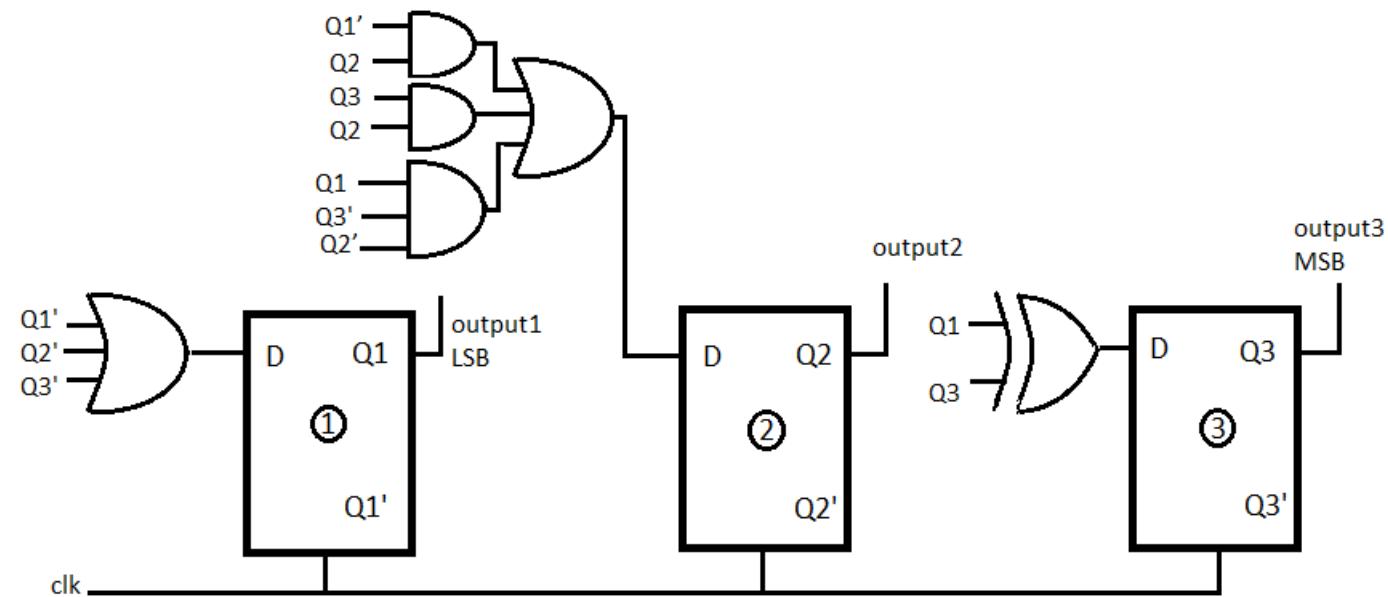
D <sub>2</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	0	1	0	1
1	0	0	1	1

D <sub>1</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	1	1	1	1
1	1	1	0	1

INPUT TABLE OF FLIP-FLOPS		
D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>
0	0	1
1	1	1
0	1	1
1	0	1
1	0	1
0	0	1
1	1	1
0	1	0

- Implementation of the design

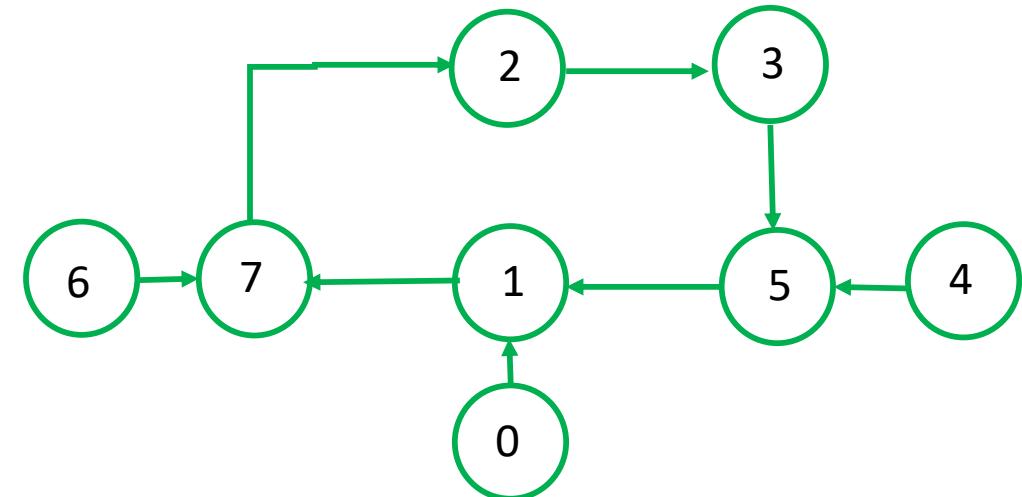
- $D_3 = Q_3'Q_1 + Q_3Q_1'$
- $D_2 = Q_2Q_1' + Q_3'Q_2'Q_1 + Q_3Q_2$
- $D_1 = Q_1' + Q_3' + Q_2'$



# Counter design for given sequence considering unused states

- **Problem** – Design a mod -5 counter for sequence:  $2 \rightarrow 3 \rightarrow 5 \rightarrow 1 \rightarrow 7 \rightarrow 2$ , using JK flip-flop. The counter must be self starting with count states 0 to 1, 4 to 5, and 6 to 7 so that the number gates for JK inputs are minimized.
- **Understanding the problem-**

PRESENT STATE	NEXT STATE
2	3
3	5
5	1
1	7
7	2
0	1
4	5
6	7



- Mod 5 counter requires 5 unique states, therefore 3 flip flops are required for the design

- State transition table and inputs to the flip flop for given sequence:

PRESENT STATE			NEXT STATE			INPUT TABLE OF FLIP-FLOPS					
$Q_3$	$Q_2$	$Q_1$	$Q_3(t+1)$	$Q_2(t+1)$	$Q_1(t+1)$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	1	1	1	X	1	X	X	0
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	1	1	X	X	1	X	0
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	0	0	1	X	1	0	X	X	0
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	0	X	1	X	0	X	1

- Write the input table of all T flip-flops by using the excitation table of D flip-flop.
- Excitation table of D- flip flop

$Q(t)$	$Q(t+1)$	$J$	$K$
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

- Find value of D<sub>3</sub>, D<sub>2</sub>, D<sub>1</sub> in terms of Q<sub>3</sub>, Q<sub>2</sub>, Q<sub>1</sub> using K-Map (Karnaugh Map):

J <sub>3</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	0	1	1	0
1	X	X	X	X

$$\begin{aligned} J_3 &= Q_1 \\ J_2 &= Q_3'Q_1 \\ J_1 &= 1 \end{aligned}$$

J <sub>2</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub> 0	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$$\begin{aligned} K_3 &= Q_1 \\ K_2 &= Q_3'Q_1 \\ K_1 &= Q_3Q_2 \end{aligned}$$

J <sub>1</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	1	X	X	1
1	1	X	X	1

INPUT TABLE OF FLIP-FLOPS					
J <sub>3</sub>	K <sub>3</sub>	J <sub>2</sub>	K <sub>2</sub>	J <sub>1</sub>	K <sub>1</sub>
0	X	0	X	1	X
1	X	1	X	X	0
0	X	X	0	1	X
1	X	X	1	X	0
X	0	0	X	1	X
X	1	0	X	X	0
X	0	X	0	1	X
X	1	X	0	X	1

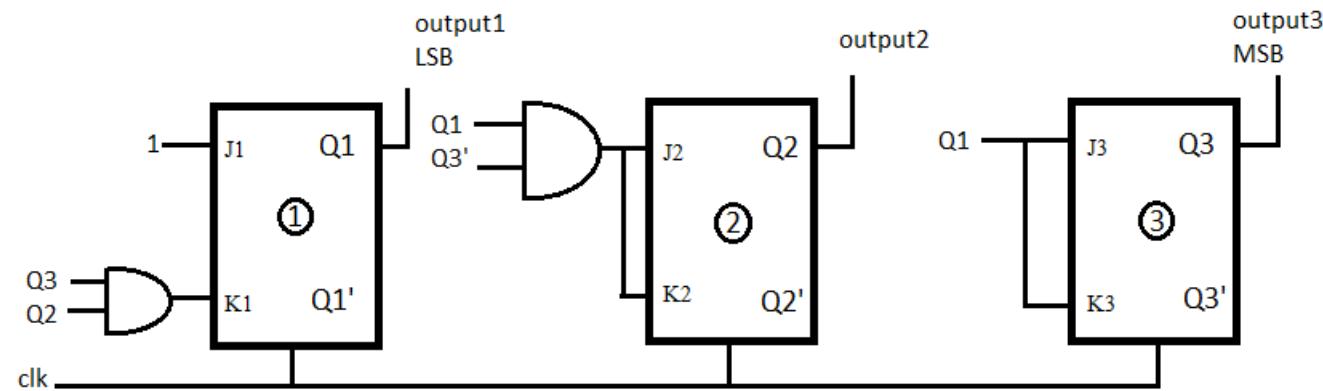
K <sub>3</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	X	X	X	X
1	0	1	1	0

K <sub>2</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	X	X	1	0
1	X	X	0	0

K <sub>1</sub> Q <sub>3</sub> \Q <sub>2</sub> Q <sub>1</sub>	00	01	11	10
0	X	0	0	X
1	X	0	1	X

- Implementation of the design

- $J_3 = Q_1$        $K_3 = Q_1$
- $J_2 = Q_3'Q_1$        $K_2 = Q_3'Q_1$
- $J_1 = 1$        $K_1 = Q_3Q_2$



# Presetable counters

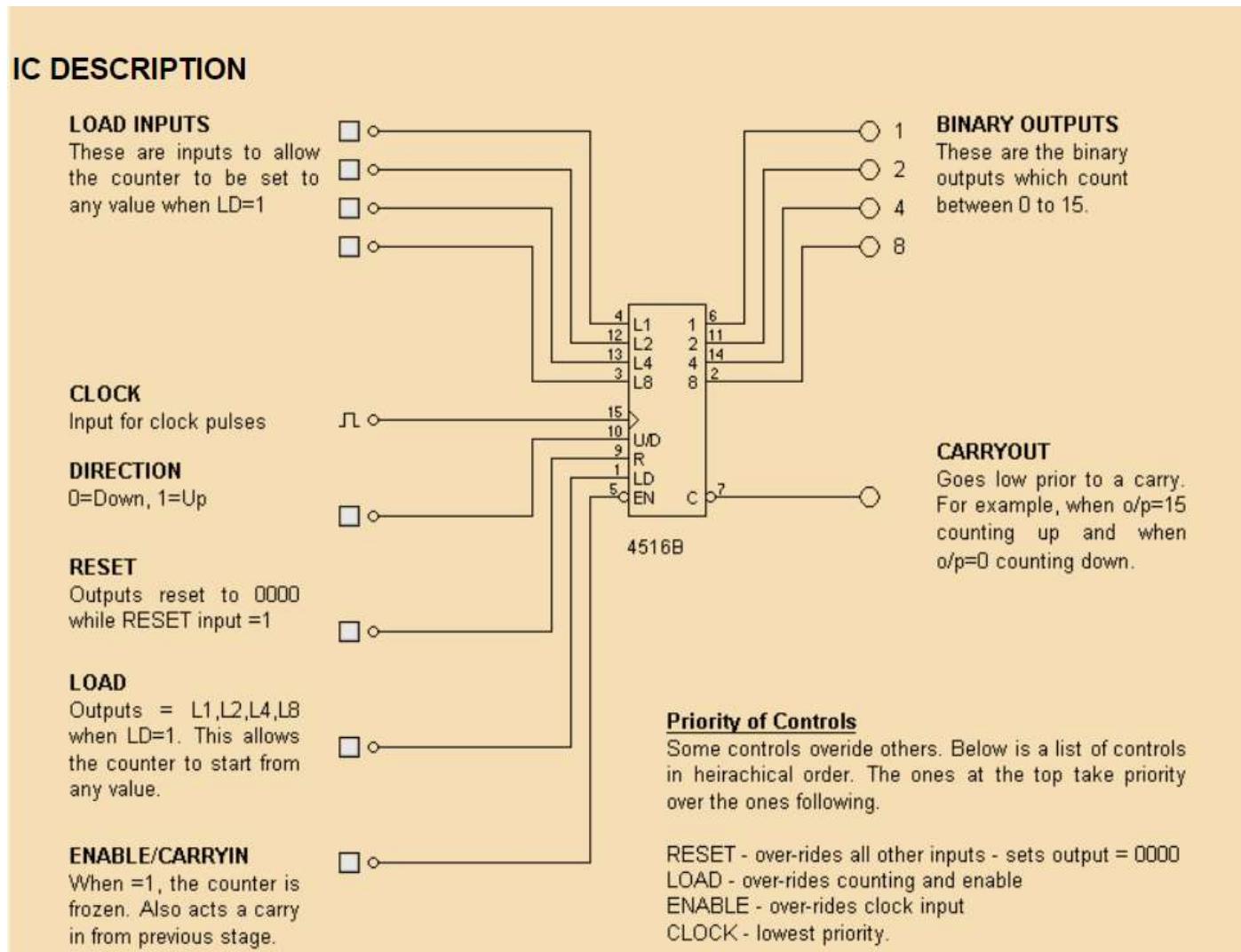
- USING THE 4516, 4 BIT BINARY COUNTER

- **Presetable:**

- This means when the LD input is high, then whatever binary value is present on LOAD INPUTS, will be immediately copied to the outputs and stay that way until LD goes low. This enables the counter to begin from any value.

- **Eg:** Count from 6 to 15.

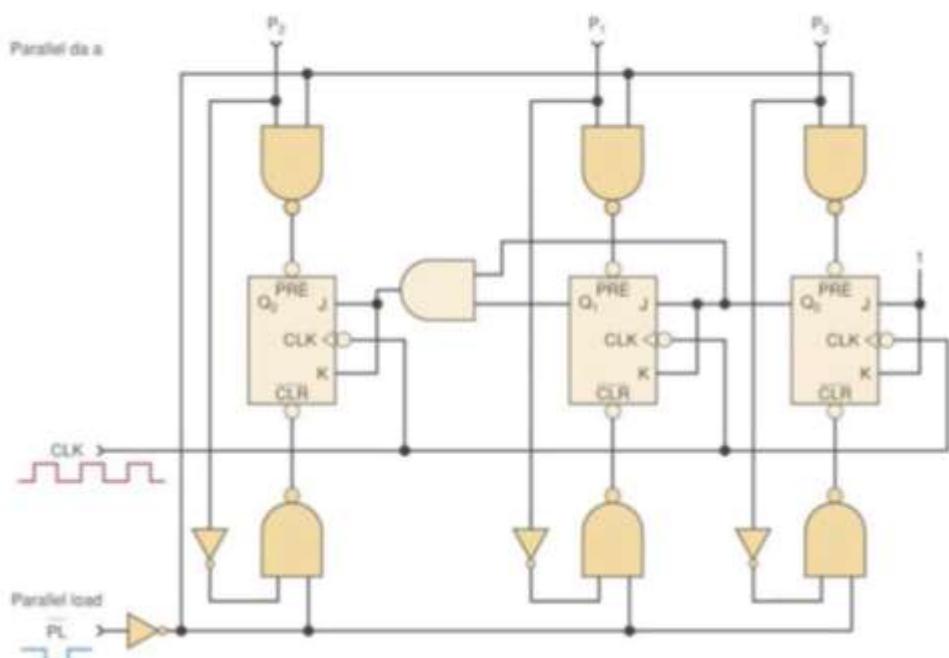
- **Note:** This only works if the RESET input is low.



# 7-8 Presettable Counters

- Many synchronous counters that are available as ICs are designed to be **presettable**.
- Presettable means that the counters can be preset to any desired starting count.
- The presetting operation is also referred to as **parallel loading** the counter.
- to perform asynchronous presetting. The counter is loaded with any desired count at any time by doing the following:
  1. Apply the desired count to the parallel data inputs,  $P_2$ ,  $P_1$ , and  $P_0$ .
  2. Apply a LOW pulse to the PARALLEL LOAD input,  $PL$ .

## 7-8 Presettable Counters



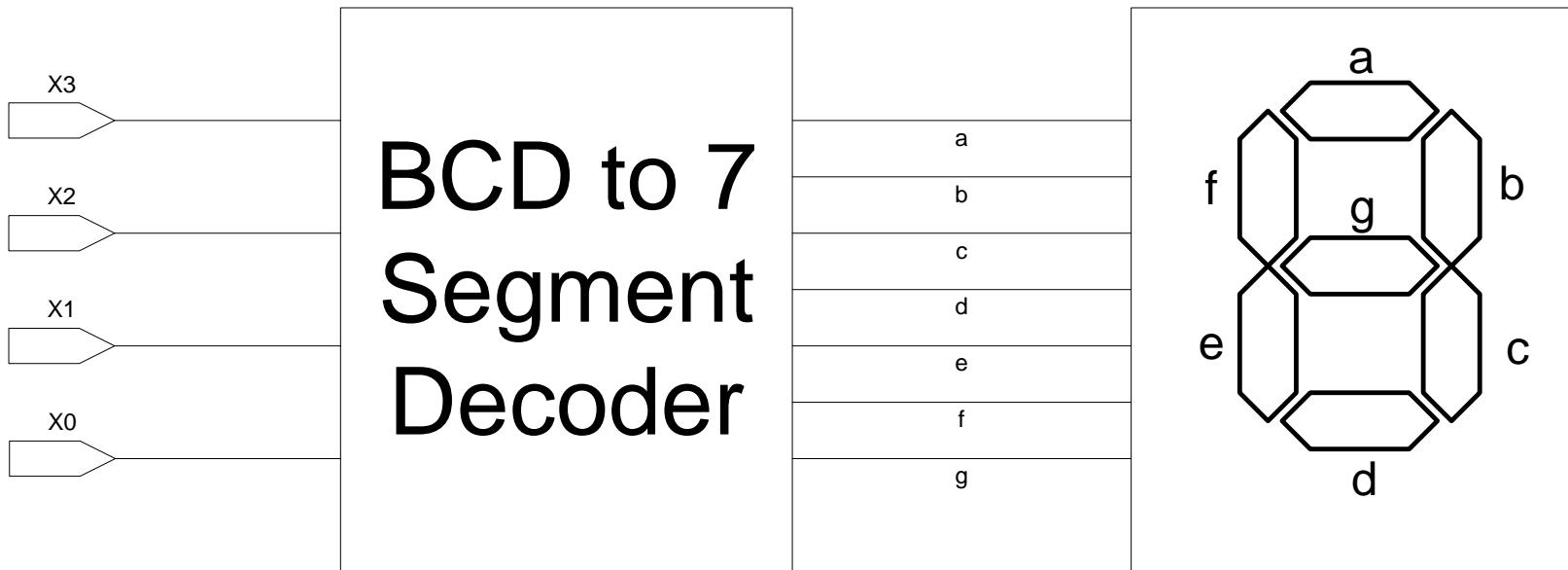
S-8

SLO-1      Seven segment Display and A Digital Clock.  
SLO-2      Problem solving session

# Seven segment Display

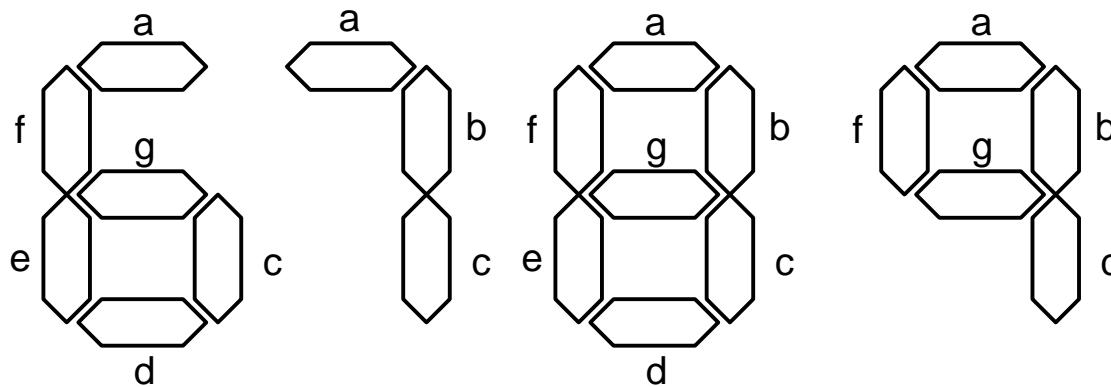
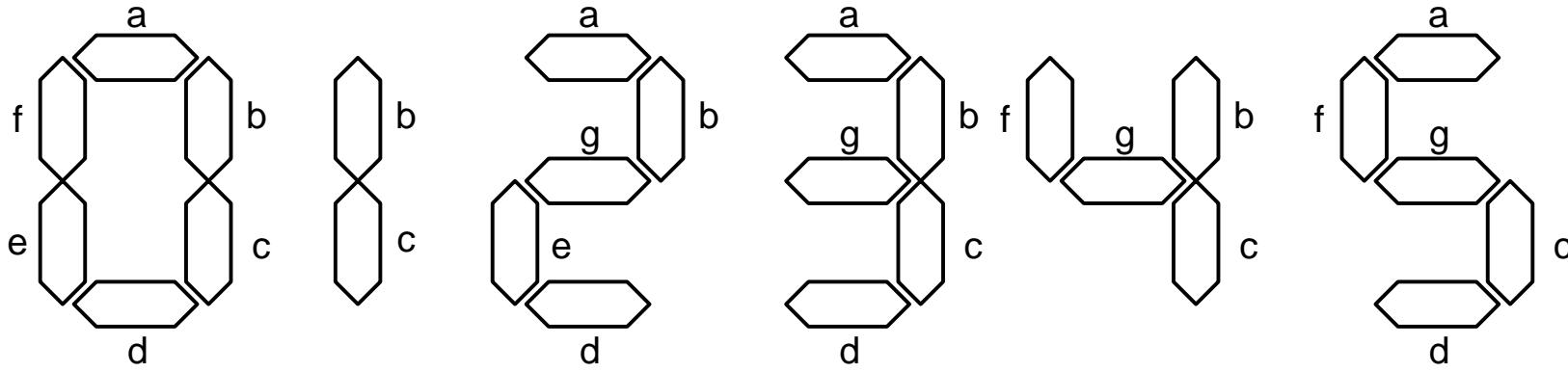
- A seven-segment display (SSD), or seven-segment indicator, is a form of electronic display device for displaying decimal numerals.
- The Seven segment display consists of seven LEDs arranged in a rectangular fashion.
- Each of the seven LEDs is called a segment.
- Each segment is labelled from “a” to “g”
- Display the any one of the decimal digits (0 to 9)
- BCD to seven segment decoder accepts a decimal digit in BCD and generate the corresponding

# Design Requirements



Design the logic circuitry that will drive a seven segment LED display and will be able to represent numbers from 0 to 9

# Possible numbers and their representation on 7 segment display



# Truth Table

Dec	X3	X2	X1	X0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1
10	1	0	1	0	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x

$$a = \sum_m (0,2,3,5,6,7,8,9) + \sum_d (10,11,12,13,14,15)$$

$$b = \sum_m (0,1,2,3,4,7,8,9) + \sum_d (10,11,12,13,14,15)$$

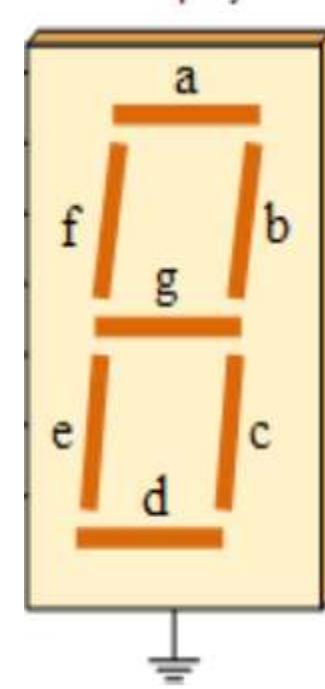
$$c = \sum_m (0,1,3,4,5,6,7,8,9) + \sum_d (10,11,12,13,14,15)$$

$$d = \sum_m (0,2,3,5,6,8,9) + \sum_d (10,11,12,13,14,15)$$

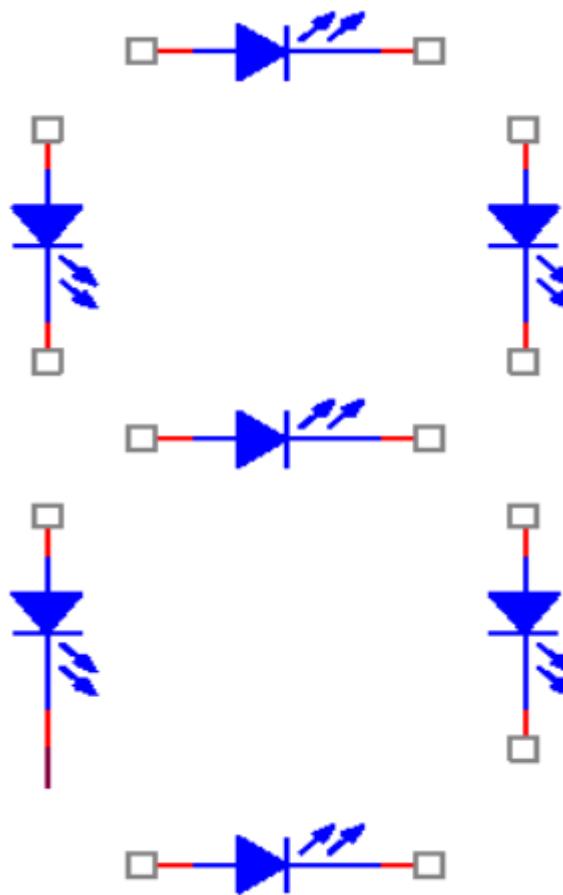
$$e = \sum_m (0,2,6,8) + \sum_d (10,11,12,13,14,15)$$

$$f = \sum_m (0,4,5,6,8,9) + \sum_d (10,11,12,13,14,15)$$

$$g = \sum_m (2,3,4,5,6,8,9) + \sum_d (10,11,12,13,14,15)$$



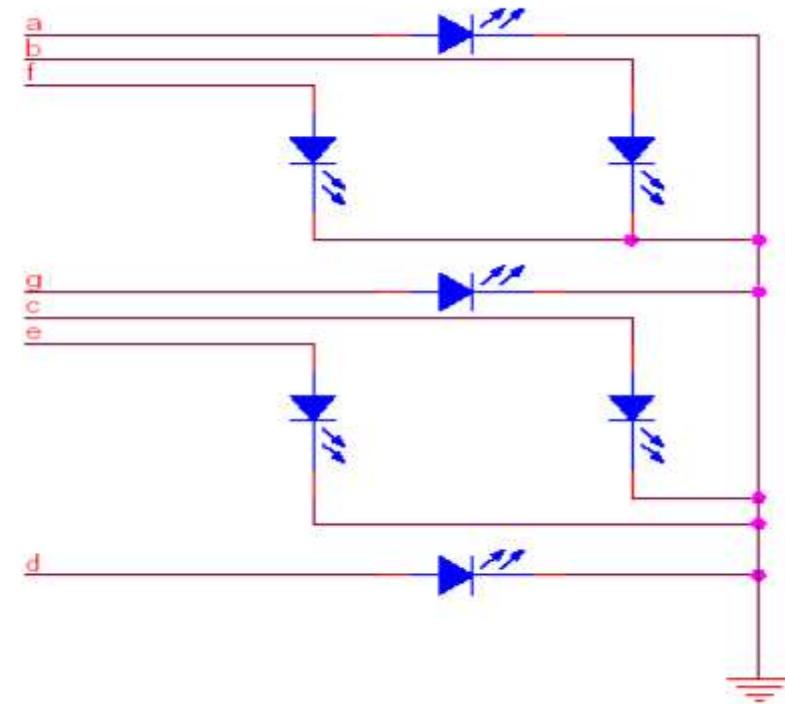
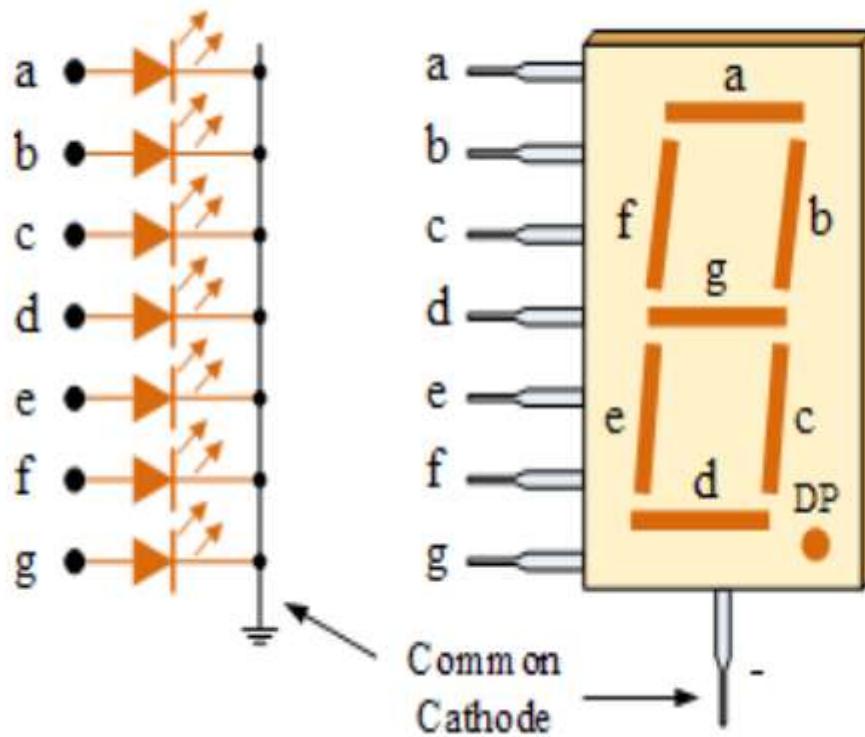
# Arrangement of LED inSSD



# Types of SSD

- The SSD common pin is generally used to identify which type of 7-segment display it is. As each LED has two connecting pins, one called the “Anode” and the other called the “Cathode”, there are therefore two types of LED 7-segment display called: Common Cathode (CC) and Common Anode(CA).
- The difference between the two displays, as their name suggests, is that the common cathode has all the cathodes of the 7-segments connected directly together and the common anode has all the anodes of the 7-segments connected together and is illuminated as follows.

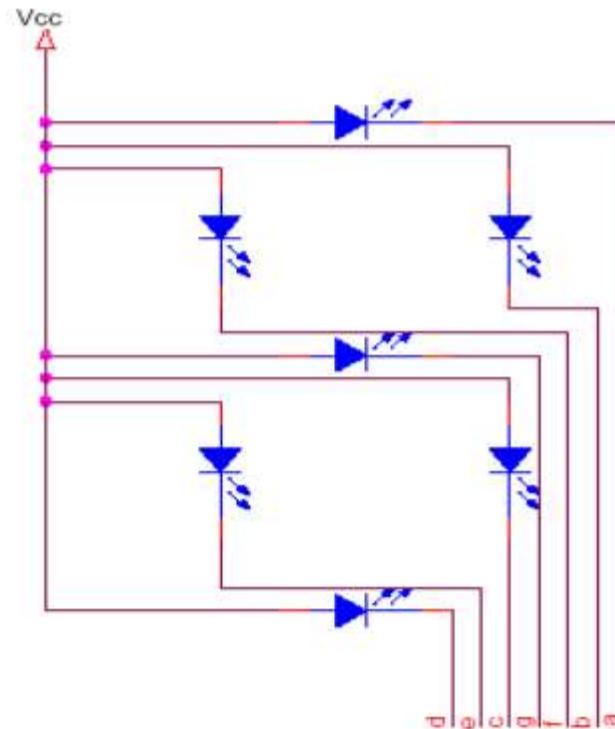
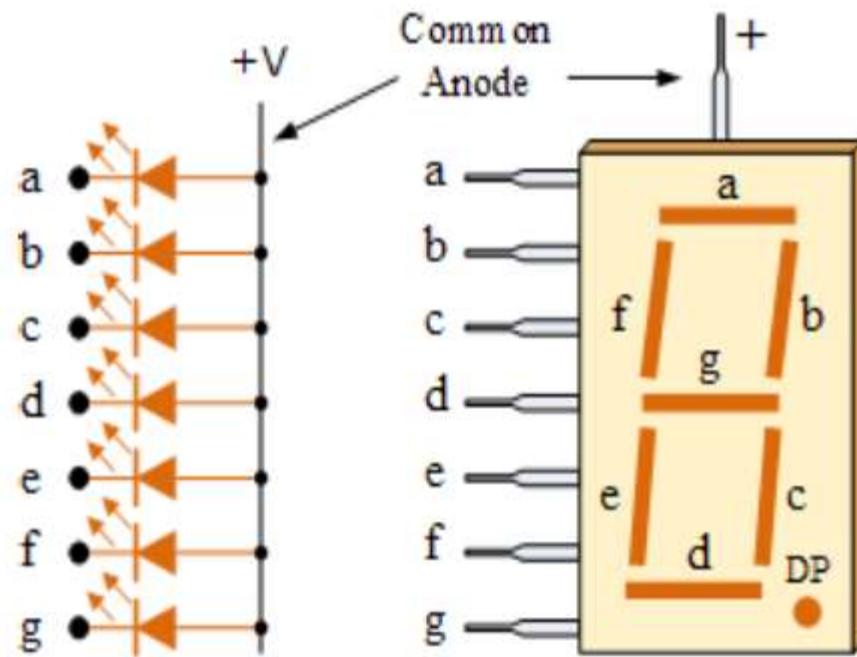
# Common cathode SSD



# Common Cathode (CC)

- In the common cathode display, all the cathode connections of the LED's are joined together to logic “0” or ground.
- The individual segments are illuminated by application of a “HIGH”, logic “1” signal to the individual Anode terminals.

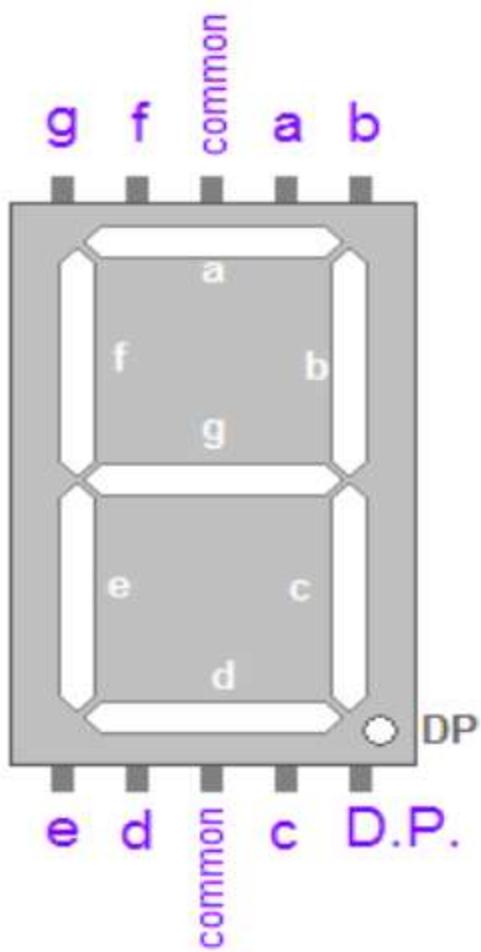
# Common Anode SSD



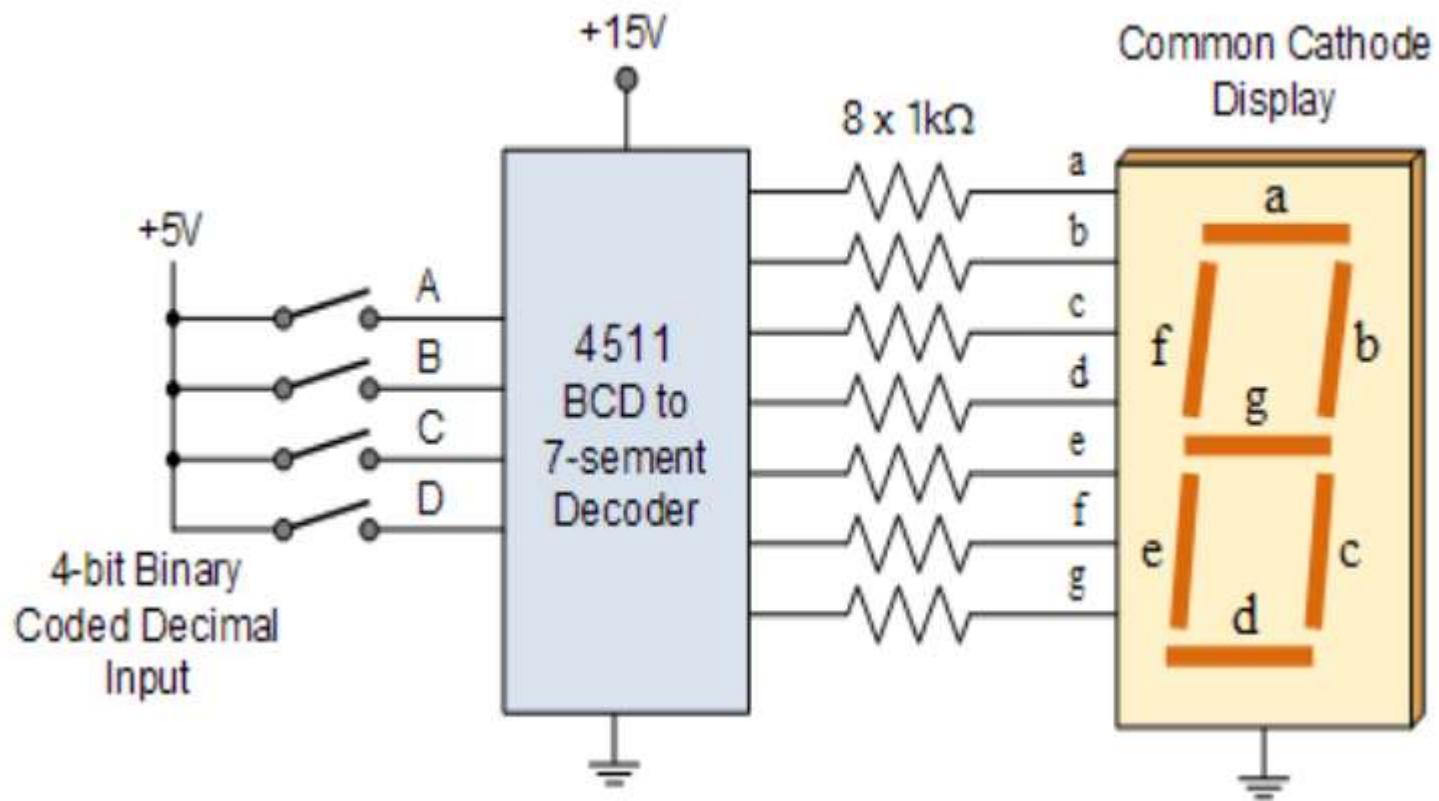
# Common Anode (CA)

- In the common anode display, all the anode connections of the LED's are joined together to logic "1".
- The individual segments are illuminated by connecting the individual Cathode terminals to a "LOW", logic "0" signal.

# Pin Configuration of SSD



# Driving 7 segment display using IC 4511



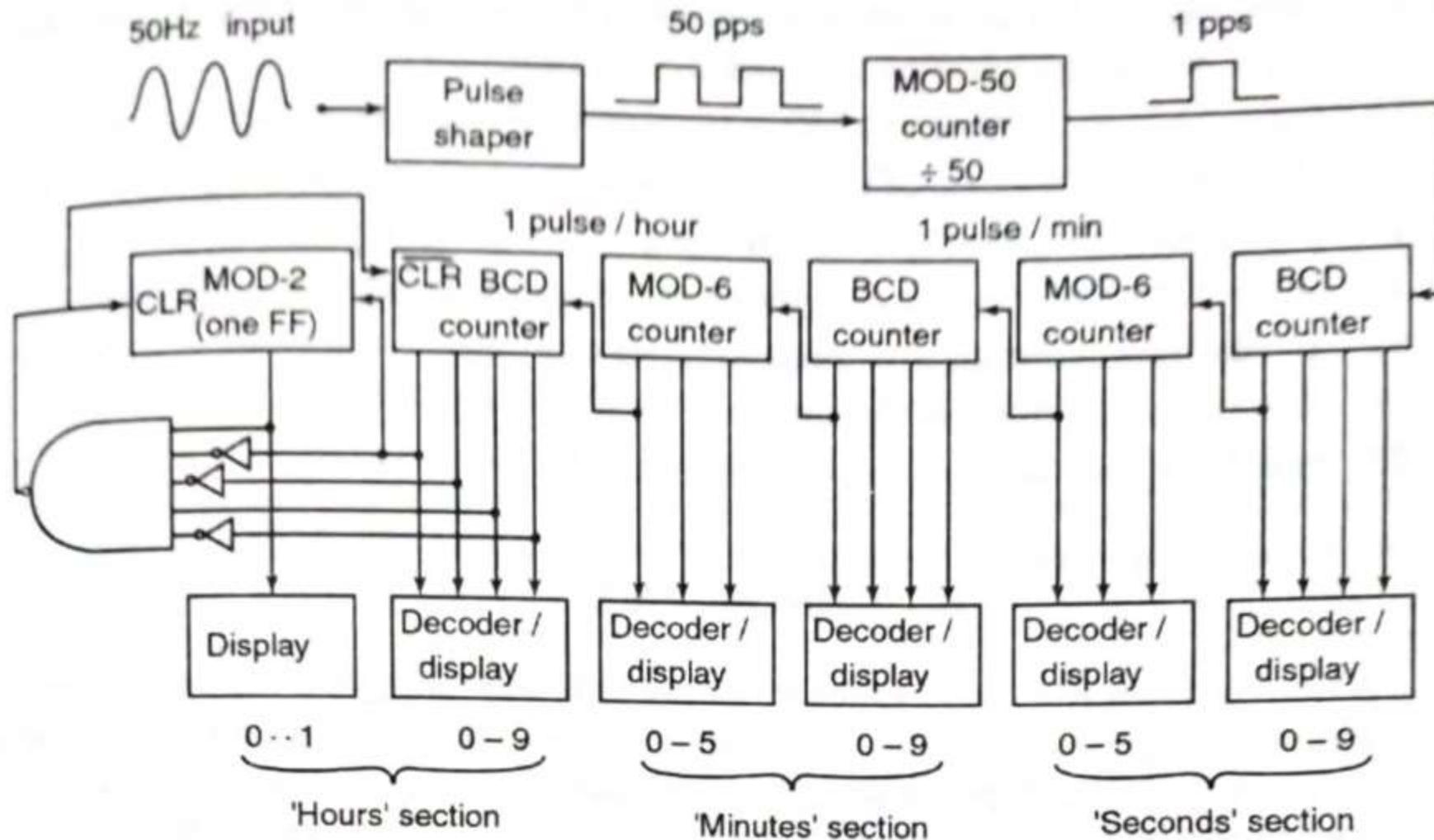
- 7-segment Displays are usually driven by a special type of integrated circuit (IC) like CMOS 4511. This 7-segment display driver is able to illuminate both common anode or common cathode displays.
- This BCD-to-7 segment driver takes a four-bit BCD input and has outputs that will pass current through the segments to display the decimal digit.

# Applications

- As Seven segment display provides a convenient way to display decimal numbers in digital form, it is helpful in general purpose devices and reduces the complexity of data delivered.
- It is used in many devices of our day to day life, like in traffic signals, watches, cd/dvd players etc.



# Digital Clock

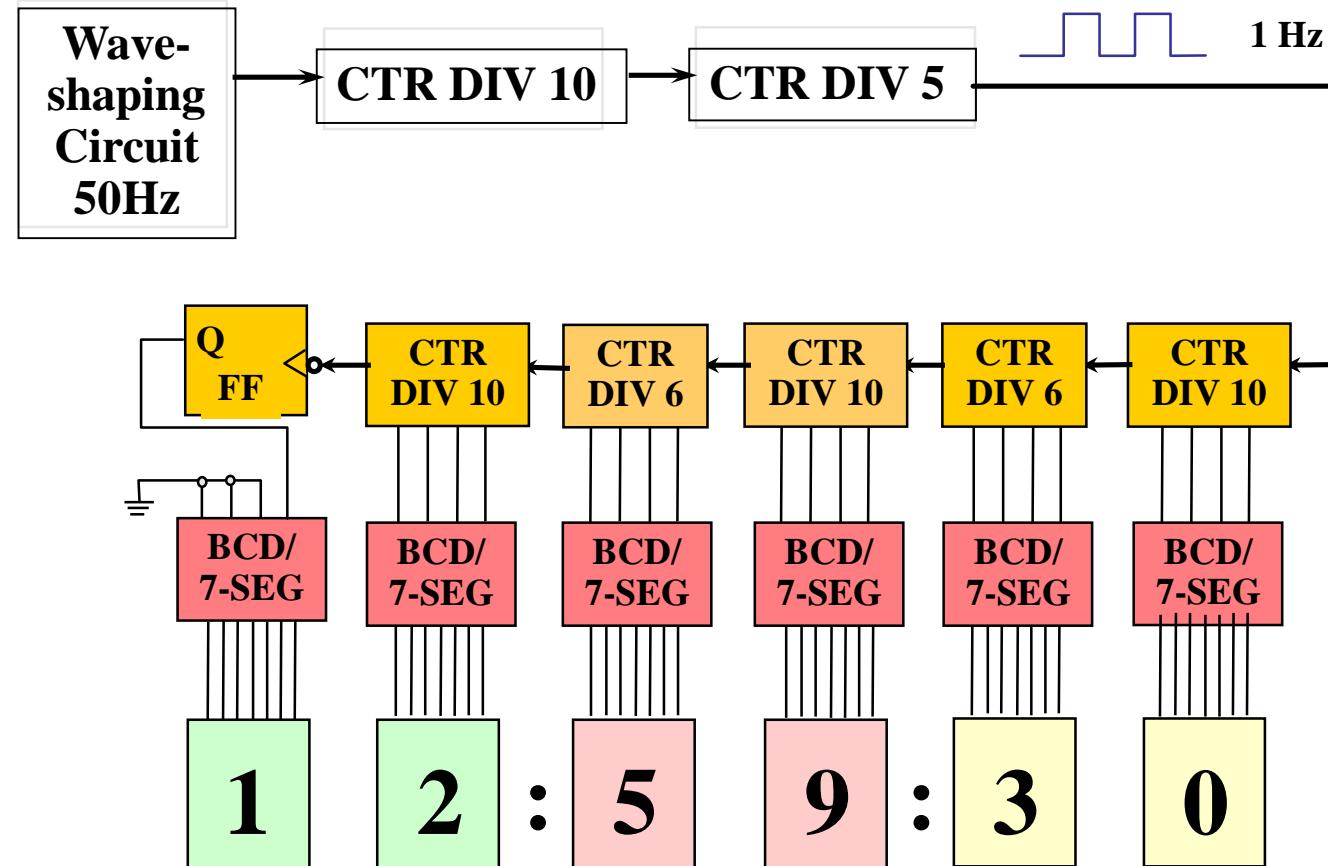


- Digital clock displays the time of day in hours, minutes and seconds.
- DC power – Frequency obtained from quartz –crystal oscillator.
- AC Power – Use power line frequency
- 50 Hz change to 1 Hz use by pulse shaper circuit and frequency divider.

# Circuit Operation

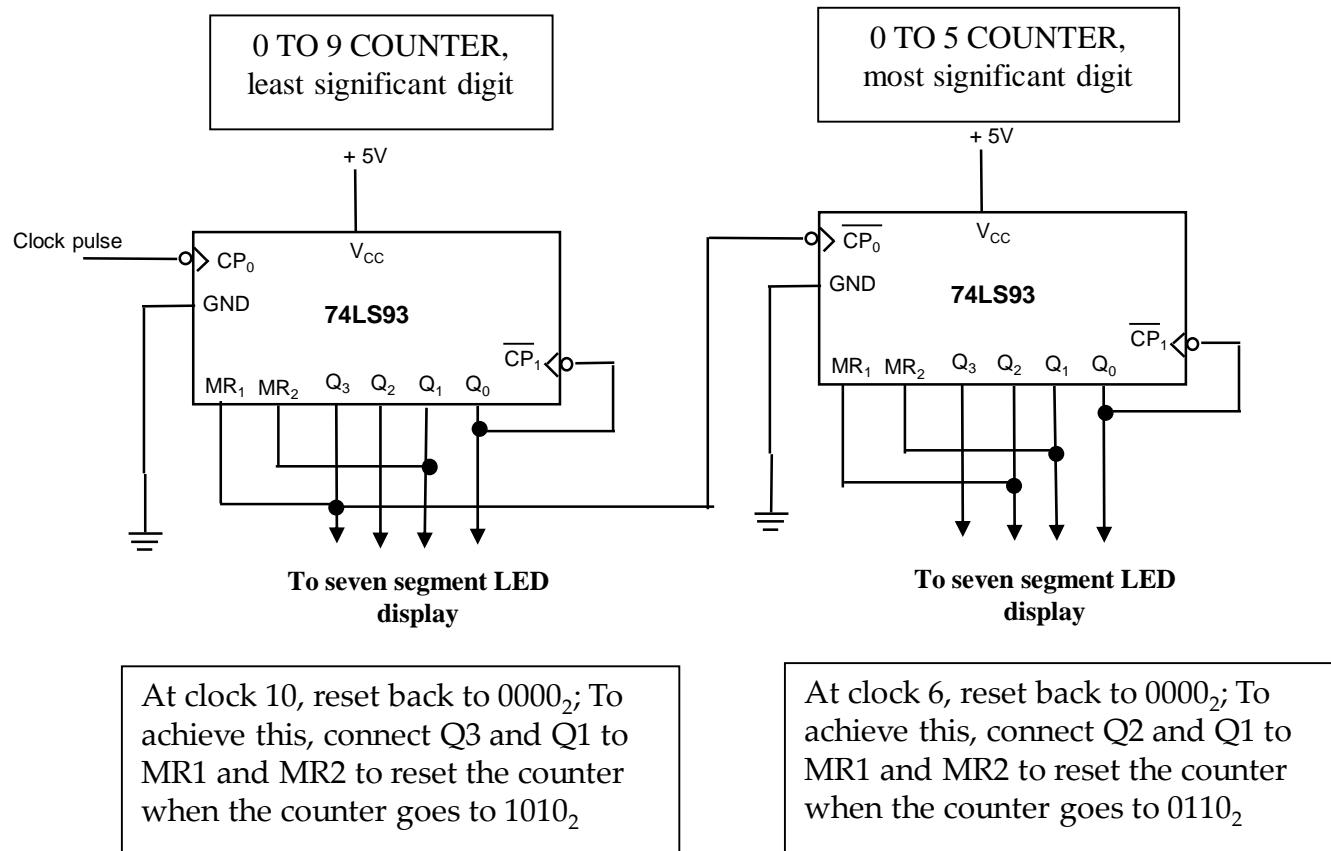
Section	Digit Level	Counter	Display
Seconds	First	BCD	0 – 9
	Second	MOD 6	0-5
Minutes	First	BCD	0 – 9
	Second	MOD 6	0-5
Hours (Reset zero when reach count 12 by using NAND gate)	First	BCD	0 – 9
	Second	MOD 2	0-1

# 12-hour Digital Clock



## Counter Applications – 60 seconds counter

We can design a 60-second counter (00-59) by cascading two 74LS93 counter ICs as shown:



# **CONVERSIONS DIGITAL to ANALOG SIGNALS D/A**

# Introduction

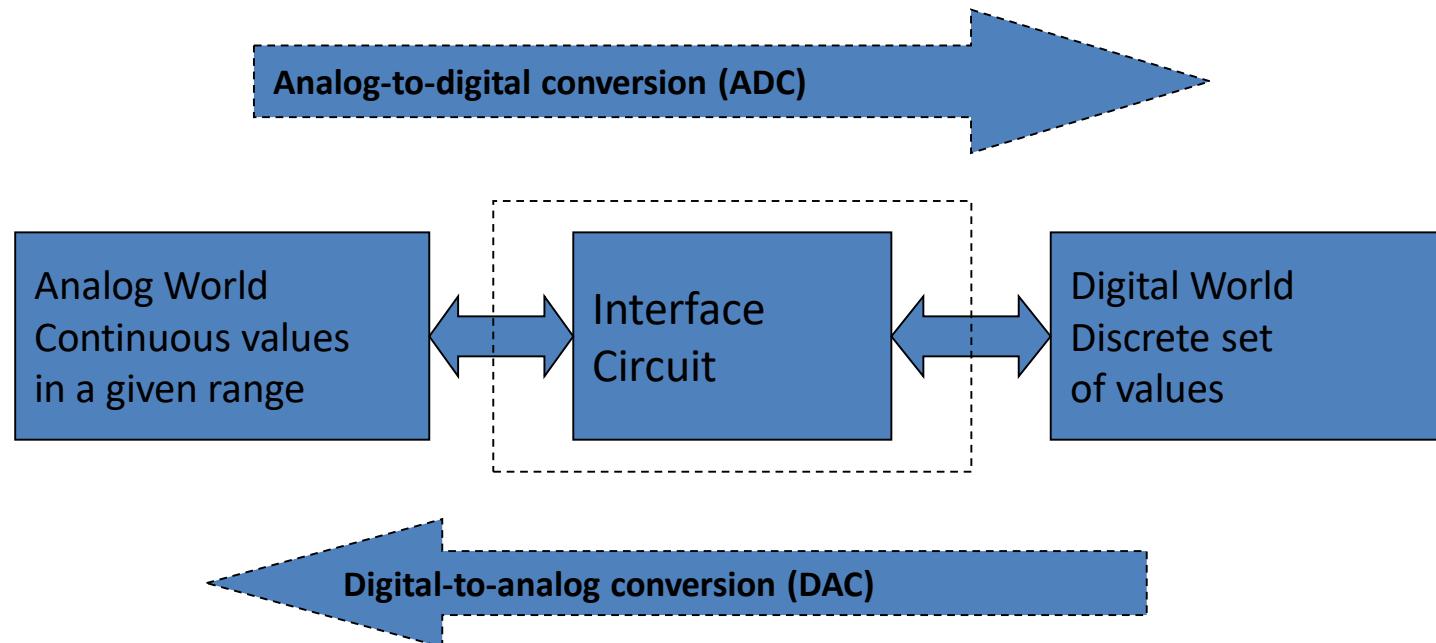
- To interface the digital and analog world, we need converters that can convert analog to digital (ADC) and vice-versa (DAC).
- Analog Signals:
  - Physical quantities are “**continuous**” infinite number of quantities over a given range of values.  
Examples: temperature, pressure, time, velocity, acceleration, fluid flow, etc
- Digital Signals:
  - Digital signals are those with **discrete values** and are represented by binary codes.  
Examples: digital audio, computer system signals, electronic music systems.

# Interfacing

- Interfacing is a process of making 2 or more devices or systems operationally compatible with each other so that they function together as required

... Interfacing Analog and Digital Worlds ...

converting digital codes to analog quantities and vice versa

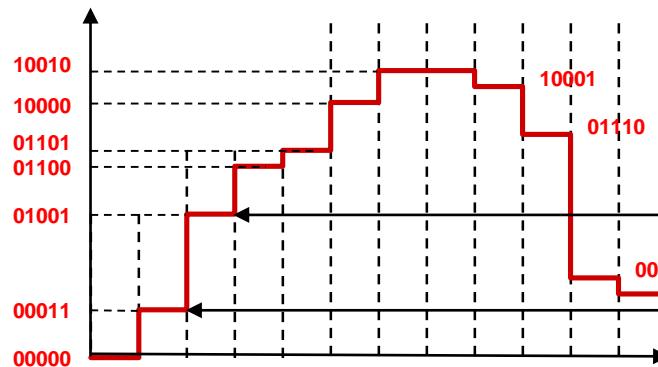


# Real World Interfacing

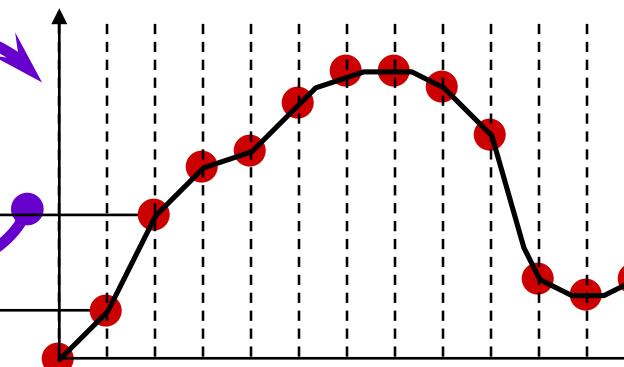
Examples:

- Electronic thermostat,
- CD player,
- Digital Audio Tape (DAT) Player/Recorder

Digital reproduction of  
the analog curve



Sample discrete points  
in an analog curve

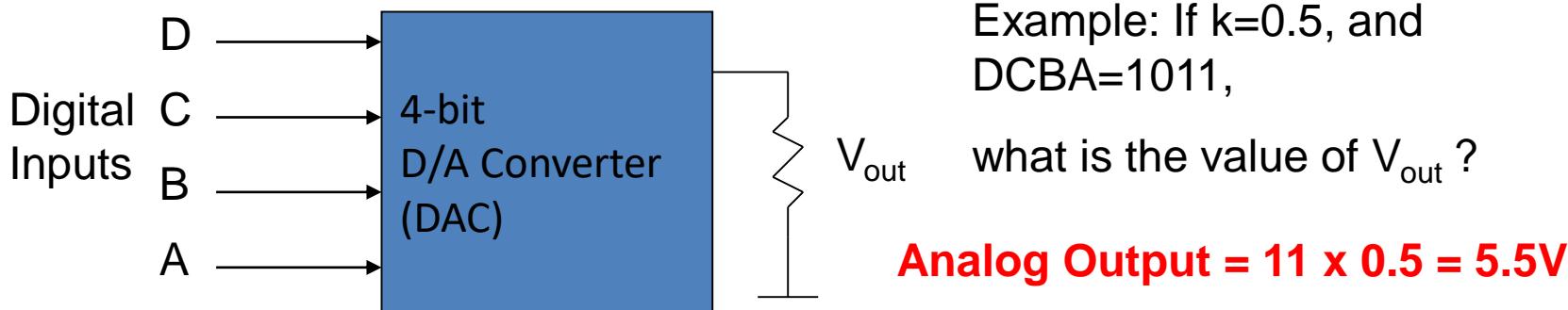


# Types of DACs

- Many types of DACs available.
- Usually switches, resistors, and op-amps used to implement conversion
- Two Types:
  - Binary Weighted Resistor
  - R-2R Ladder

# Digital-to-Analog (D/A) Conversion

- D/A conversion is the process of taking a value represented in digital code and converting it to a voltage or current that is proportional to the digital value   **Analog Output,  $V_{out}$  = digital input  $\times k$**
- For each digital input code, the output is a unique analog value



# Resolution & Full Scale Voltage

- **Resolution** or **step size** of a DAC is the smallest change that can occur in the analog output as a result of a step change in the digital input (**LSB weightage**)

Analog output = digital input x k,

k = analog output/digital input, (resolution or step size)

- **Full scale output** is the DAC output when digital input is at **maximum** value,

E.g. a 4-bit DAC FS output =  $1111_2 k = 15k = 15 \times \text{step size}$

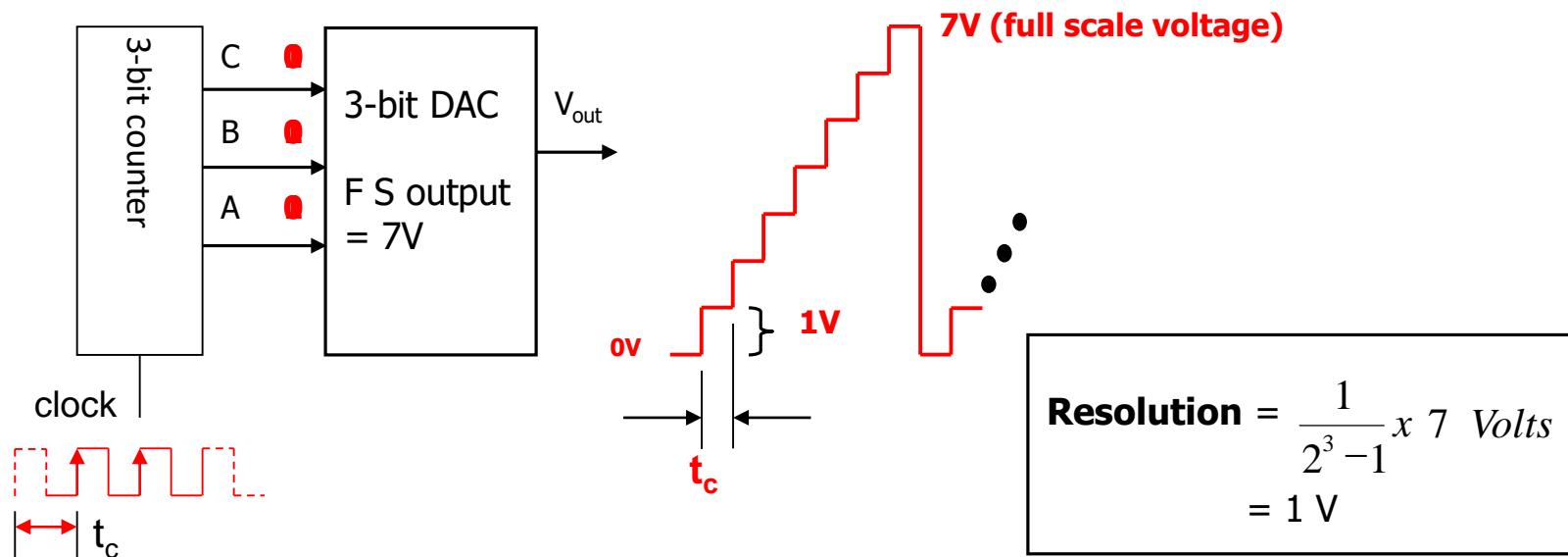
For n-bit DAC, full scale output =  $(2^n - 1)k$

- Example: If step size, k of a 10-bit DAC is 0.05V, the full scale output voltage is \_\_\_\_\_

$$1023 \times 0.05 = 51.15V$$

# Full Scale Voltage

- Full-scale voltage is the output voltage of DAC when maximum value of the digital inputs are applied.
- Refer to figure below, the full scale voltage (when input is 111) is 7V. Step size or resolution is 1 V



Another example: What is the step size for a 4-bit DAC operating with a full scale voltage of 8V?  $8 \div 15 = 0.53 \text{ V}$

# Percentage Resolution

- Although resolution can be expressed as the amount of output voltage per step, it is useful to express it as a percentage of the full scale output

$$\% \text{ resolution} = (\text{step size}) / (\text{full scale value}) \times 100\%$$

$$= k / (2^n - 1)k \times 100\%$$

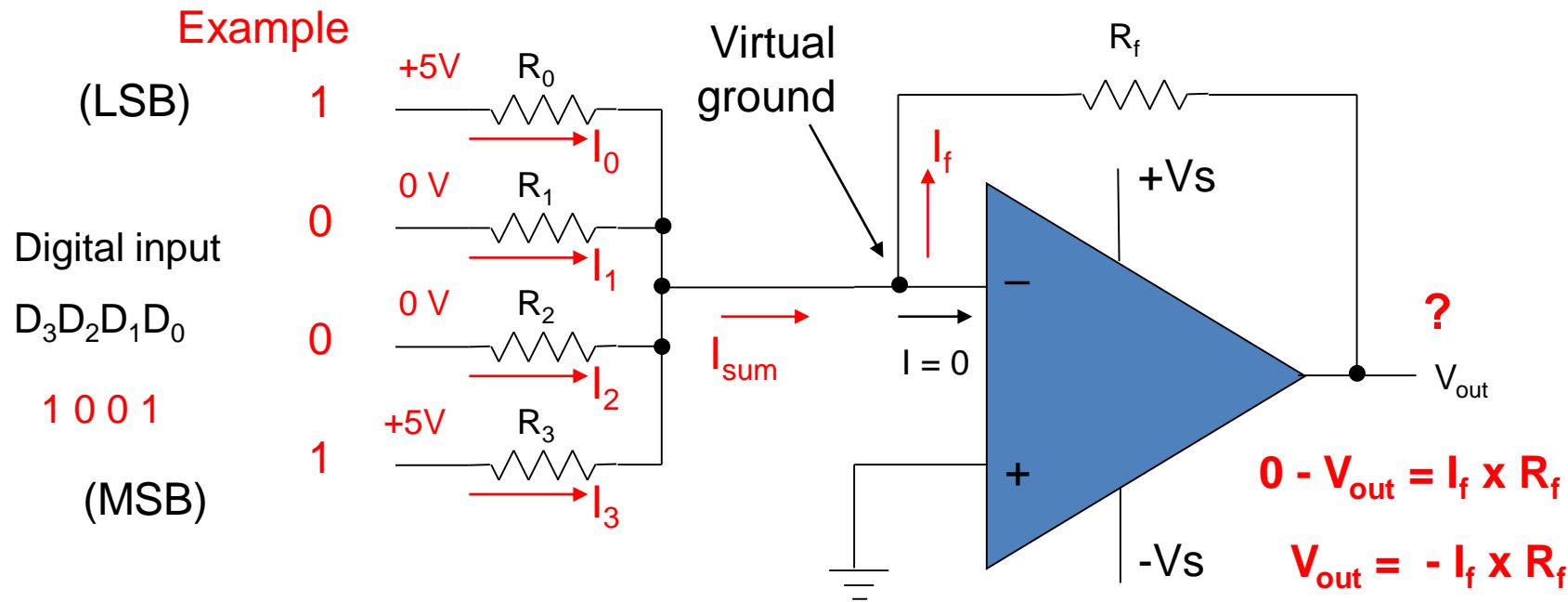
$$= \frac{1}{2^n - 1} \times 100\%$$

E.g. % Resolution of a 10-bit DAC is  $(1/1023) \times 100\%$

$$= 0.098\%$$

- Will % resolution change if the full scale voltage is changed?  
**NO, it only depends on the DAC number of bits**
-

# 4-bit DAC with Binary Weighted Inputs



1. For each input data bit,

- $V_n = V_{HI}$  (usually +5V) when digital data input  $D_n = 1$
- $V_n = 0 \text{ V}$  for  $D_n = 0$ , where  $n = 0$  to 3

2. That is,  $V_n = V_{HI}$  (5V) or 0V , when  $D_n = 1$  or 0 respectively

$$I_f = I_{sum} = I_3 + I_2 + I_1 + I_0 \quad \text{where } I_3 = V_3/R_3, I_2 = V_2/R_2, I_1 = V_1/R_1, I_0 = V_0/R_0$$

## 4-bit DAC with Binary Weighted Inputs

- In DAC, for every input bit, the voltage is assumed as 0V (if bit is logic 0) and +5V (if bit is logic 1).
  - Example: if bit  $D_0 = 0$ , then the voltage  $V_0 = 0V$
  - if bit  $D_0 = 1$ , then the voltage  $V_0 = +5V$
  - Thus voltage at input is either 0V or +5V.
- The input resistors  $R_0$  to  $R_3$  contribute the binary weightages. The current or voltage contribution from the LSB branch,  $I_0$  is the least and MSB branch,  $I_3$  is the most. To achieve this,

$$R_0 = 2R_1, R_1 = 2R_2, R_2 = 2R_3$$

**( $R_{LSB}$  has largest value while  $R_{MSB}$  has least value)**

## 4-bit DAC with Binary Weighted Inputs

- The operational amplifier is the summing amplifier and produces an output that is the weighted sum of the input voltages.
- $I_{\text{sum}}$  is the summation of all the currents in the resistors flowing through the feedback resistor  $R_f$ .

$$\begin{aligned} V_{\text{out}} &= - I_{\text{sum}} R_f = - (I_3 + I_2 + I_1 + I_0) R_f \\ &= - R_f \left( \frac{V_3}{R_3} + \frac{V_2}{R_2} + \frac{V_1}{R_1} + \frac{V_0}{R_0} \right) \end{aligned}$$

# 4-bit DAC with Binary Weighted Inputs

- $V_{out} = - I_{sum} R_f = - (I_3 + I_2 + I_1 + I_0) R_f$

$$V_{out} = - R_f \left( \frac{V_3}{R_3} + \frac{V_2}{R_2} + \frac{V_1}{R_1} + \frac{V_0}{R_0} \right)$$

where  $V_3, V_2, V_1, V_0$  is logic 0 (0V) or logic 1 (5V) for TTL

- For binary weighted currents to flow into  $R_0, R_1, R_2, R_3$ ,

$$R_0 = \text{LSB resistor}, R_1 = \frac{R_0}{2}, R_2 = \frac{R_0}{4}, R_3 = \frac{R_0}{8}$$

- Therefore,  $V_{out} = R_f \left( \frac{8V_3}{R_0} + \frac{4V_2}{R_0} + \frac{2V_1}{R_0} + \frac{1V_0}{R_0} \right)$

## 4-bit DAC with Binary Weighted Inputs

- $$V_{out} = -R_f \left( \frac{8V_3}{R_0} + \frac{4V_2}{R_0} + \frac{2V_1}{R_0} + \frac{V_0}{R_0} \right)$$
$$= -\frac{R_f}{R_0} (8V_3 + 4V_2 + 2V_1 + V_0)$$

where  $V_n$  is 0 or 5 volts

- Step-size is found by setting data input to  $0001_2$

$$V_{step\text{-size}} = \frac{5R_f}{R_{LSB}}$$

- Percentage resolution =  $\frac{1}{(2^n - 1)} \times 100\%$

# Full Scale Voltage

- Method 1:  $V_{out} = - R_f \left( \frac{V_3}{R_3} + \frac{V_2}{R_2} + \frac{V_1}{R_1} + \frac{V_0}{R_0} \right)$

Where  $V_3 = V_2 = V_1 = V_0 = 5V$

- Method 2:

For full scale output set input  $D_3D_2D_1D_0 = 1111_2 (15_{10})$

$$V_{FS} = -\frac{5R_f}{R_0} (1111_2) = -15 \frac{5R_f}{R_0}$$

= step size x maximum digital input value

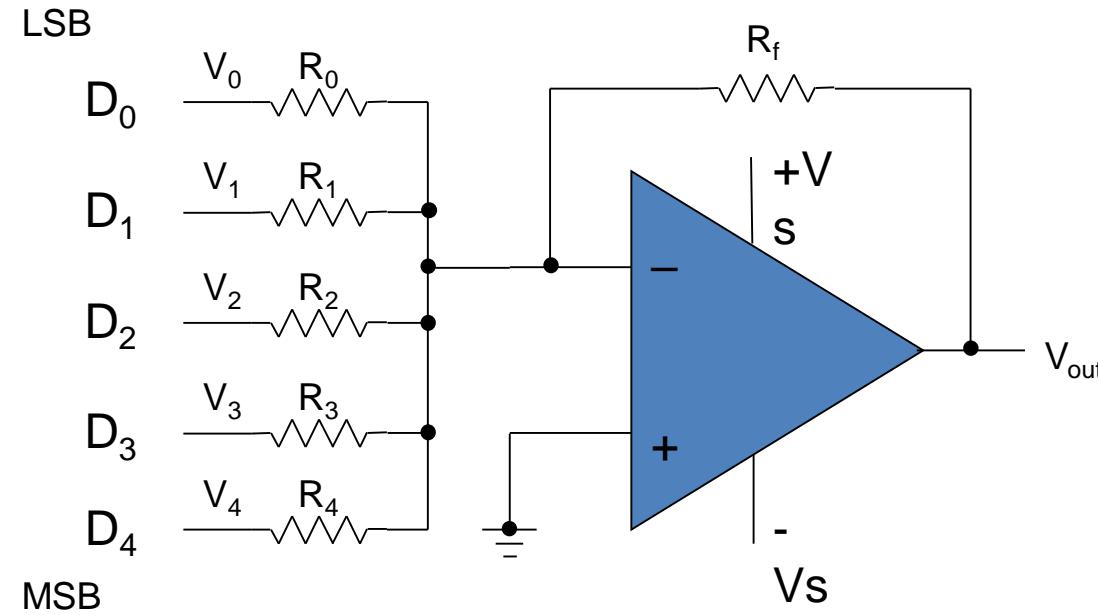
= step size x  $(2^n - 1)$  where n = no of bits

## Example 1

- A 5-bit binary weighted DAC is given in the next slide. Assuming the input voltage at each bit is either 0V or +5V for logic 0 and 1 respectively, calculate
  - (1) Step size or resolution
  - (2) Full scale output voltage,  $V_{out}$
  - (3)  $V_{out}$  for binary code of 11010
  - (4) Binary input for  $V_{out} = -0.75V$

Given that  $R_f = 10 \text{ K}\Omega$ ,  $R_4$  (MSB) =  $25 \text{ K}\Omega$

# Example 1



**Remember:**

**$R_0$  is resistor for LSB and has the largest value**

# Example 1

## 1. Step size

$$\text{Step size} = -\frac{5R_f}{R_0}$$

$$R_3 = 50\text{k}\Omega, R_2 = 100\text{k}\Omega, R_1 = 200\text{k}\Omega$$

$R_0 = 400 \text{ k}\Omega$  ( $R_0$  is LSB resistor)

$$\text{step size} = -5 R_f / R_0 = -5 \times 10 / 400 \text{ V} = -0.125\text{V}$$

## 2. Full Scale Output voltage

Data = 11111 for  $V_{\text{out(FS)}}$

$$V_{\text{out(FS)}} = (2^5 - 1) \times \text{step size} = -31 \times 0.125\text{V} = -3.875\text{V}$$

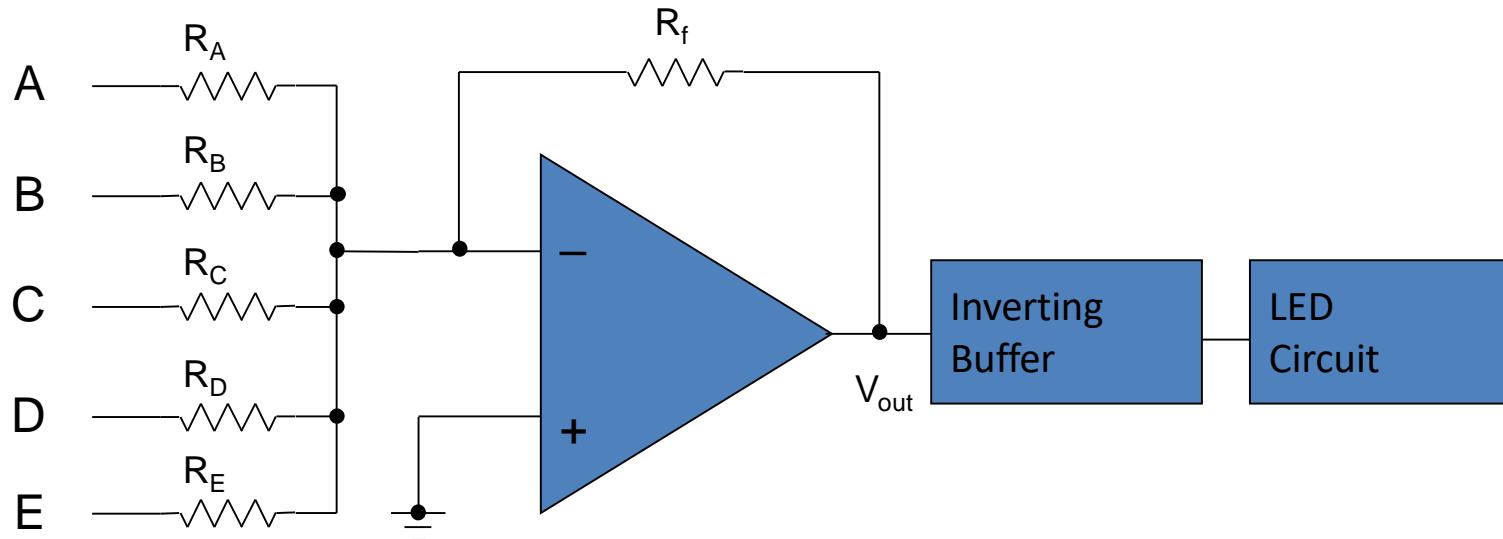
## 3. For data input = 11010,

$$V_{\text{out}} = 11010_2 \times \text{step size} = -26 \times 0.125\text{V} = -3.25\text{V}$$

## 4. Binary input = $V_{\text{out}}/\text{step-size} = (-0.75)/(-0.125) = 6_{10} = 00110_2$

## Example 2

The analog output of a 5-bit DAC as shown below is connected to a Light Emitting Diode (LED) circuit through an inverting unity gain buffer, so that its voltage is used to light up the LED.



## Example 2 (con't)

- a) Given that  $R_A$  is the LSB resistor (least weighted resistor) of the DAC having a value of  $800\text{ k}\Omega$ , write down the resistance values of all the DAC input resistors  $R_B$ ,  $R_C$ ,  $R_D$ , and  $R_E$ .
- b) Given that the magnitude of the full scale output of the DAC  $V_{out}$  is  $-3.875\text{V}$ , calculate the value of the feedback resistor  $R_f$ . Assume that a logic “1” at the inputs of the DAC is  $+5\text{V}$  and a logic “0” is  $0\text{V}$ .
- c) If the LED requires at least  $V_{out} = -1.5\text{V}$  to turn ON and that it can withstand a forward voltage of  $-15\text{V}$ , determine the range of input binary codes to the DAC that are able to turn ON the LED.

## Example 2 Solution

- a) The LSB resistor,  $R_A$  has the highest value (smallest input current contribution to summing amplifier).

$$R_B = R_A/2 = 800\text{k}/2 = 400 \text{ k}\Omega$$

$$R_C = R_B/2 \text{ or } R_A/4 = 200 \text{ k}\Omega$$

$$R_D = R_C/2 \text{ or } R_A/8 = 100 \text{ k}\Omega$$

$$R_E = R_D/2 \text{ or } R_A/16 = 50 \text{ k}\Omega$$

## Example 2 Solution

b) Full scale voltage = - 3.875V

$$\text{Step size} = -3.875/(2^5-1) = -0.125\text{V}$$

$$\text{Step size} = -5R_f/R_A = -0.125$$

$$\text{Therefore } R_f = (0.125 \times 800)/5 \text{ k}\Omega = \mathbf{20\text{k}\Omega}$$

c) For  $V_{out} = -1.5\text{V}$ , binary input =  $-1.5/\text{step-size}$

$$= 1.5/0.125 = 12 \text{ (or } 01100_2)$$

-15V exceeds the full scale output of -3.875V

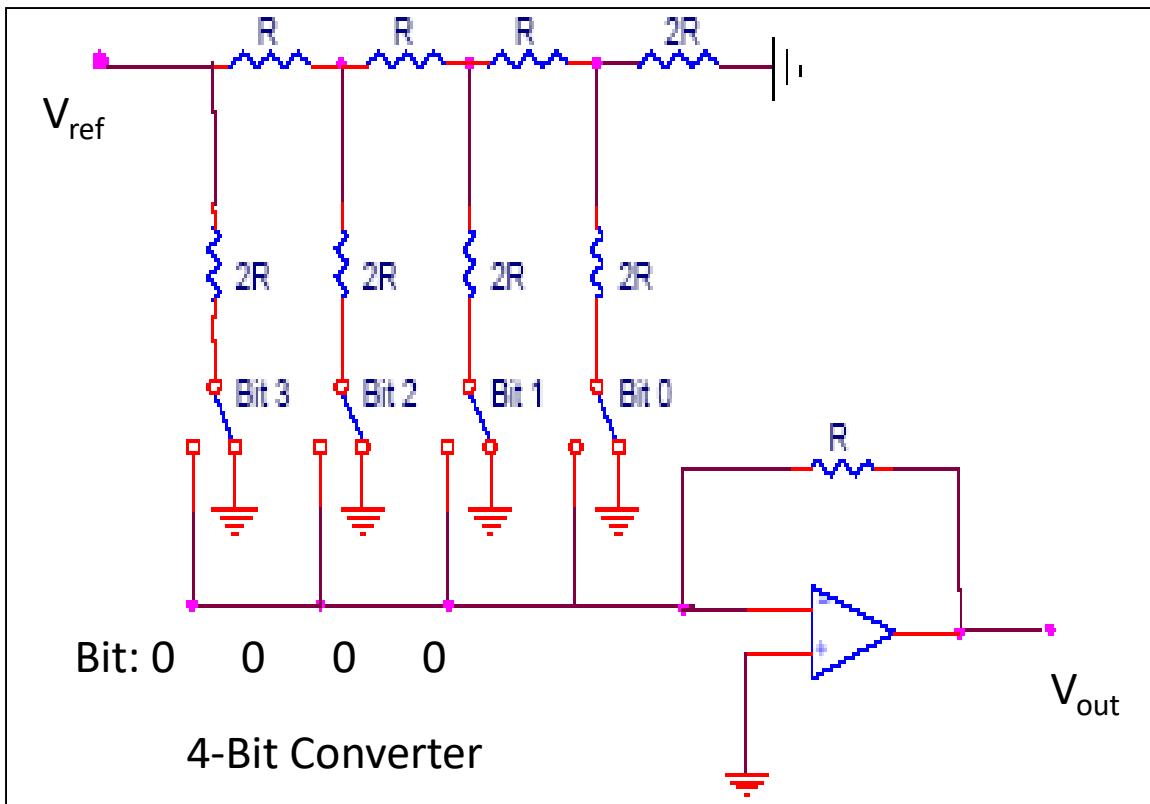
Input codes to turn LED on are from 12 to 31

or  $01100_2$  to  $11111_2$

# Binary Weighted Resistor

- Advantages
  - Simple Construction/Analysis
  - Fast Conversion
- Disadvantages
  - Requires large range of resistors (2000:1 for 12-bit DAC) with necessary high precision for low resistors
  - Requires low switch resistances in transistors
  - Can be expensive. Therefore, usually limited to 8-bit resolution.

# R-2R Ladder

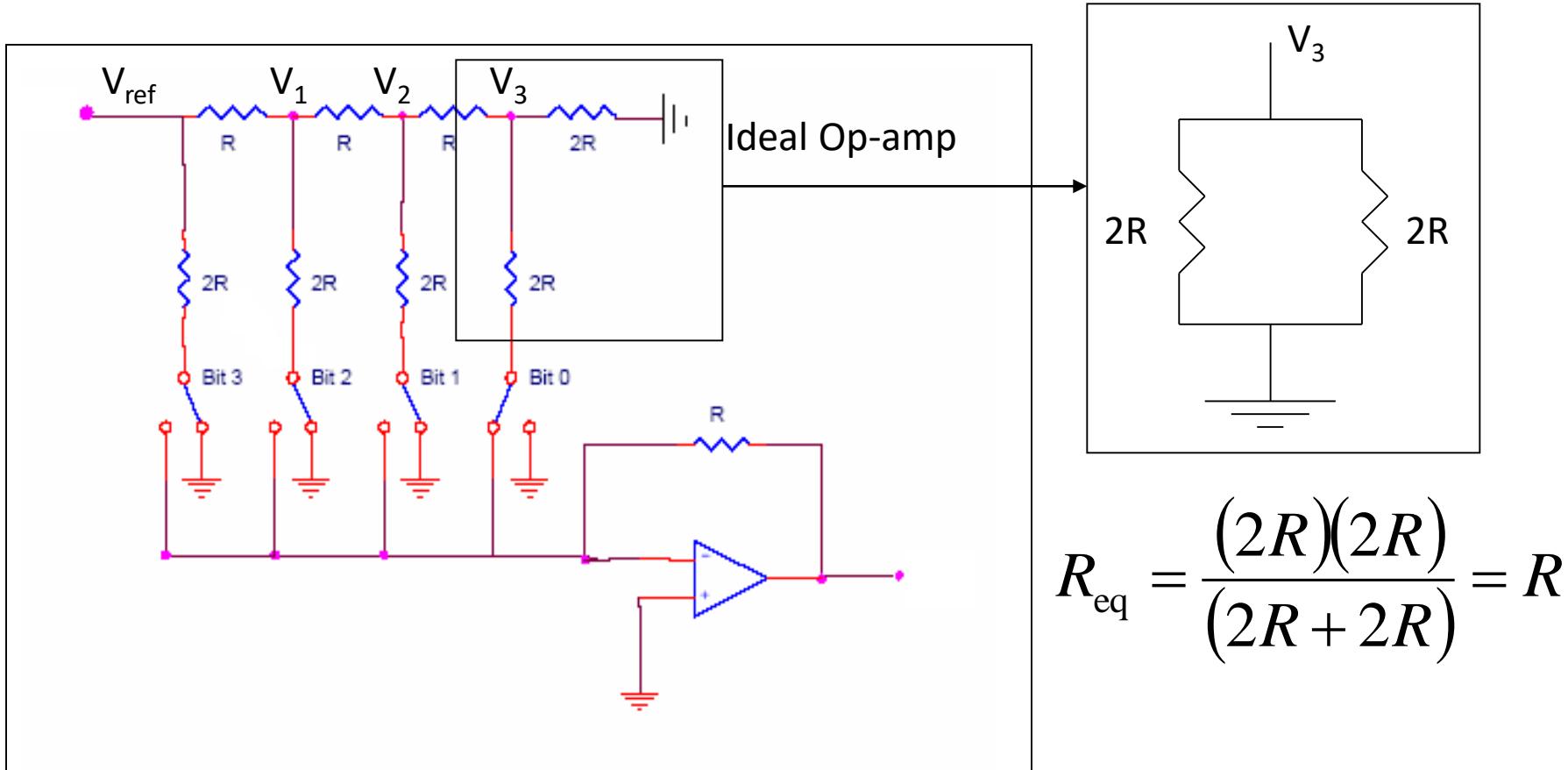


Each bit corresponds to a switch:

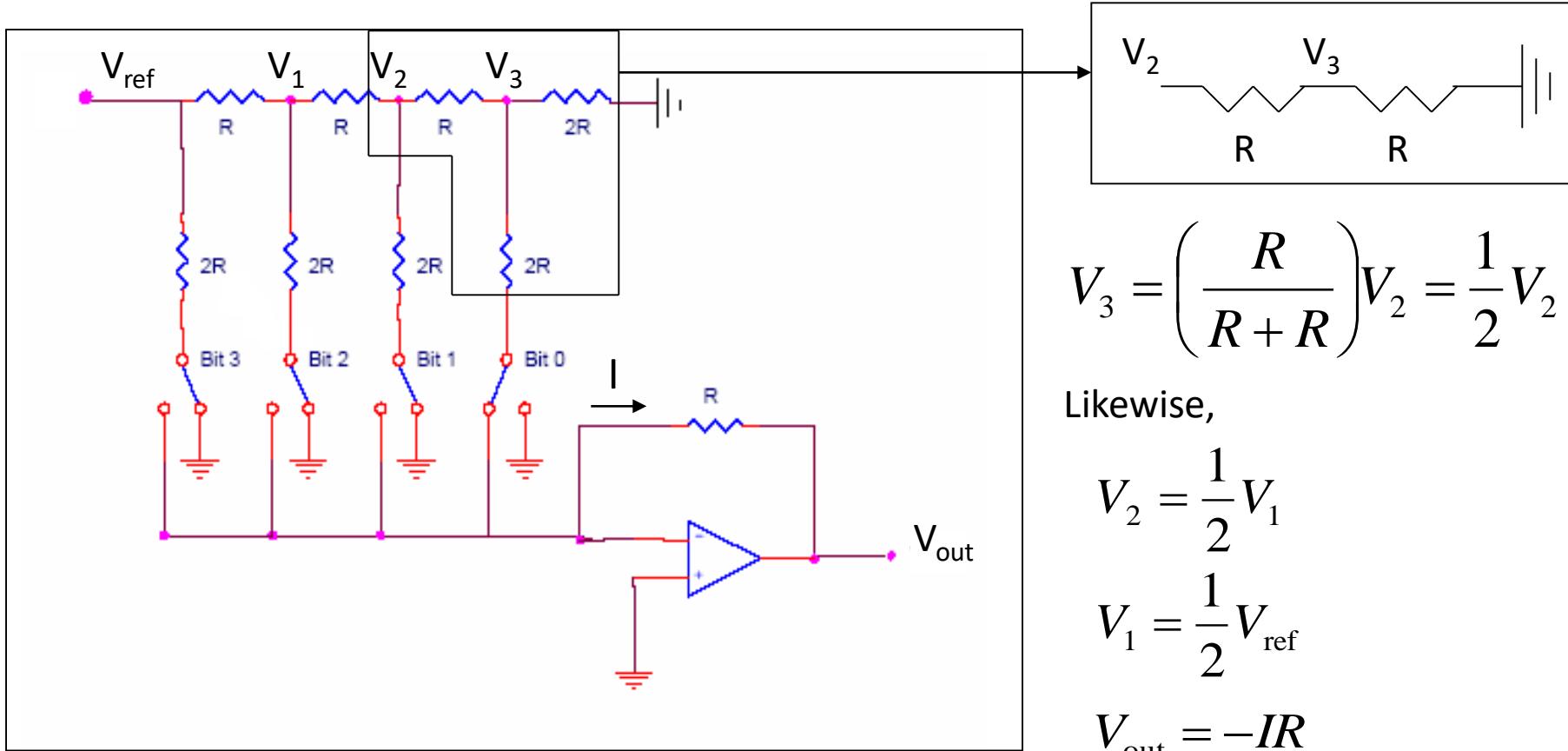
If the bit is high, the corresponding switch is connected to the inverting input of the op-amp.

If the bit is low, the corresponding switch is connected to ground.

# R-2R Ladder

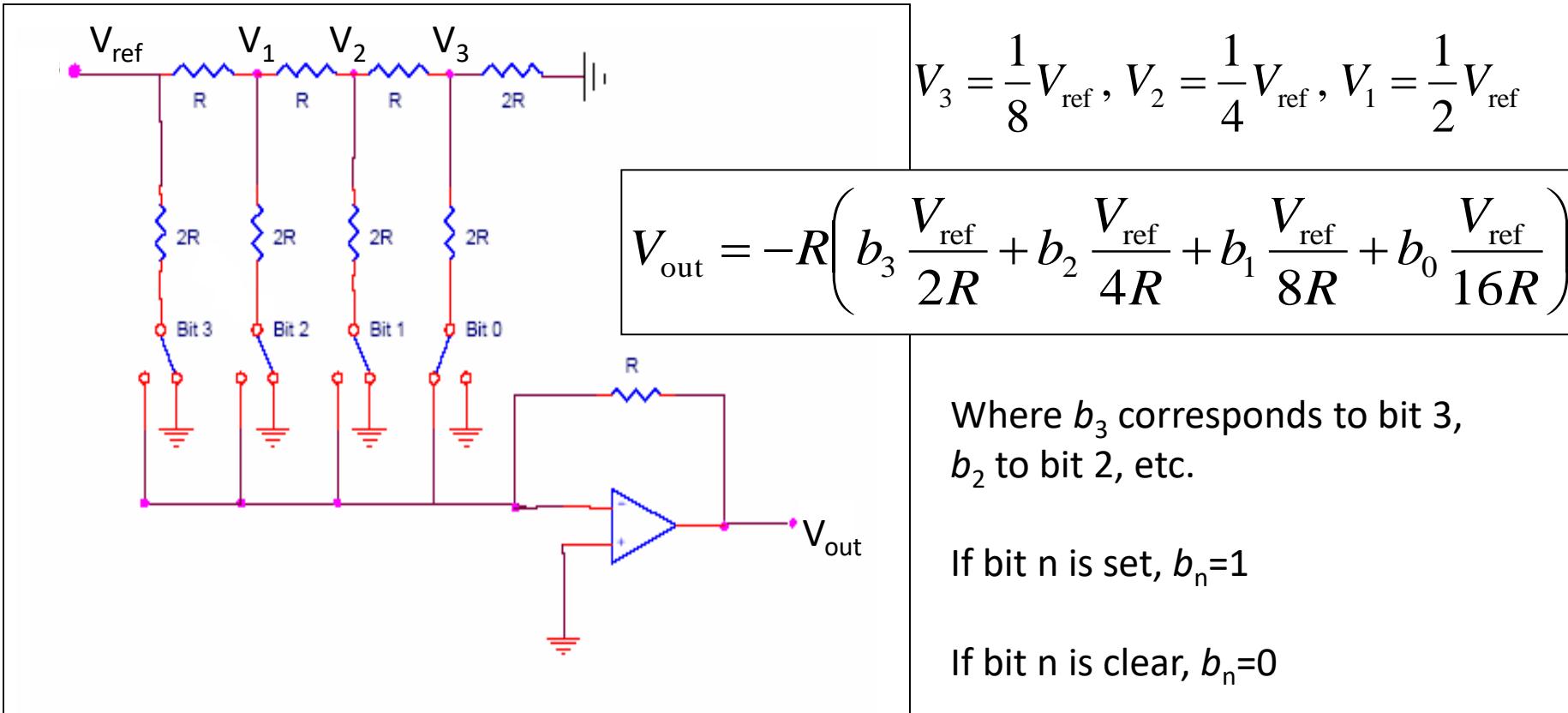


# R-2R Ladder



# R-2R Ladder

Results:



# R-2R Ladder

For a 4-Bit R-2R Ladder

$$V_{\text{out}} = -V_{\text{ref}} \left( b_3 \frac{1}{2} + b_2 \frac{1}{4} + b_1 \frac{1}{8} + b_0 \frac{1}{16} \right)$$

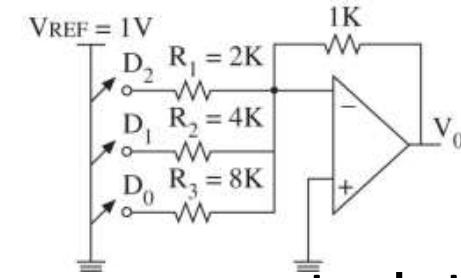
For general n-Bit R-2R Ladder or Binary Weighted Register DAC

$$V_{\text{out}} = -V_{\text{ref}} \sum_{i=1}^n b_{n-i} \frac{1}{2^i}$$

# Problem

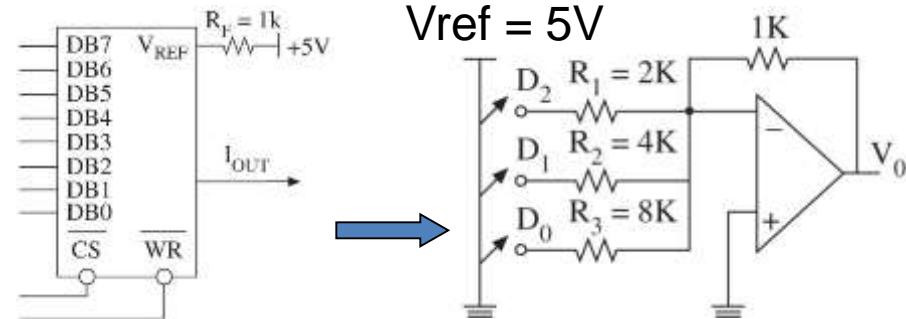
- If the reference voltage is 1 V, and if all switches are connected, the output current can be calculated as follows:

- Output voltage  $I_T = I_1 + I_2 + I_3 = \frac{V_{REF}}{R_1} + \frac{V_{REF}}{R_2} + \frac{V_{REF}}{R_3} = \frac{V_{REF}}{1k} \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{8} \right) = 0.875 \text{ mA}$



$$V_o = -R_f I_T = -(1k) \times (0.875 \text{ mA}) = -0.875 \text{ V} = \left| \frac{7}{8} \text{ V} \right|$$

# Problem



- What will be the analog equivalent of 1001 0001?

$$I_O = \frac{V_{REF}}{R_{REF}} \left( \frac{A_1}{2} + \frac{A_2}{4} + \frac{A_3}{8} + \dots + \frac{A_n}{2^n} \right)$$

$$I_O = \frac{V_{REF}}{R_{REF}} \left( \frac{DB7}{2} + \frac{DB6}{4} + \frac{DB5}{8} + \dots + \frac{DB0}{2^8} \right)$$

$$I_O = 5 \text{ mA} \left( \frac{1}{2} + \frac{0}{4} + \frac{0}{8} + \frac{1}{16} + \frac{0}{32} + \frac{0}{64} + \frac{0}{128} + \frac{1}{256} \right) = 2.832 \text{ mA}$$

$$V_O = -R_f I_T = -(1k) \times (2.832 \text{ mA}) = -2.832 \text{ V}$$

# R-2R Ladder

- Advantages
  - Only two resistor values ( $R$  and  $2R$ )
  - Does not require high precision resistors
- Disadvantage
  - Lower conversion speed than binary weighted DAC

# **ANALOG TO DIGITAL CONVERSION (ADC)**

# Analog-to-Digital Conversion

## Terminology

*analog*: continuously valued signal, such as temperature or speed, with infinite possible values in between

*digital*: discretely valued signal, such as integers, encoded in binary

analog-to-digital converter: ADC, A/D, A2D;  
converts an analog signal to a digital signal

# Analog Signals

Analog signals – directly measurable quantities  
in terms of some other quantity

## Examples:

- Thermometer – mercury height rises as temperature rises
- Car Speedometer – Needle moves farther right as you accelerate

# Digital Signals

Digital Signals – have only two states. For digital computers, we refer to binary states, 0 and 1.  
“1” can be on, “0” can be off.

## Examples:

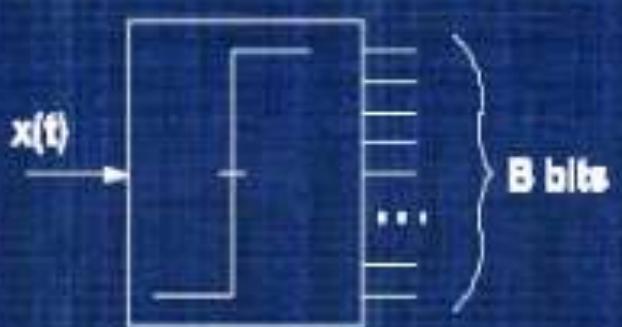
- Light switch can be either on or off
- Door to a room is either open or closed

## ADC Basic Principle:

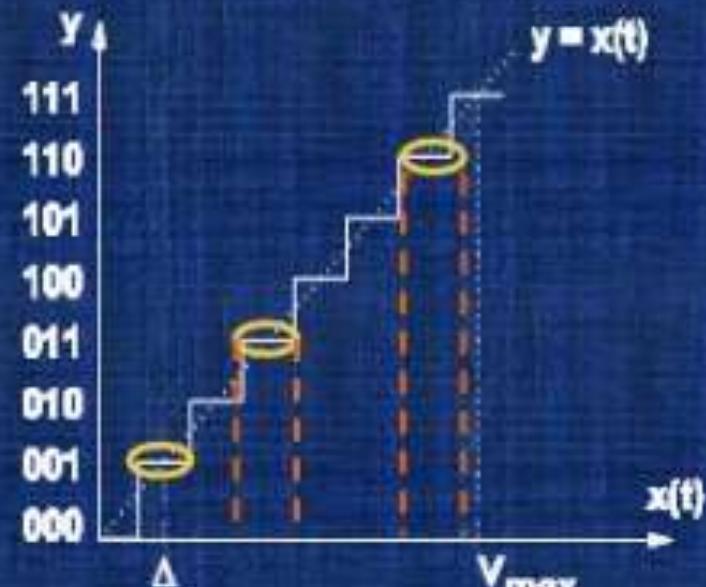
- The basic principle of operation is to use the comparator principle to determine whether or not to turn on a particular bit of the binary number output.
- It is typical for an ADC to use a digital-to-analog converter (DAC) to determine one of the inputs to the comparator.

# Quantization

- Quantization is the process of converting the sampled continuous-valued signals into discrete-valued data



(a)



(b)

# **Quantizing**

**The number of possible states that the converter can output is:**

$$N=2^n$$

where n is the number of bits in the AD converter

Example: For a 3 bit A/D converter,  $N=2^3=8$ .

**Analog quantization size:**

$$Q=(V_{\text{max}} - V_{\text{min}})/N = (10V - 0V)/8 = 1.25V$$

# Analog → Digital Conversion

## 2-Step Process:

- **Quantizing** - breaking down analog value is a set of finite states
- **Encoding** - assigning a digital word or number to each state and matching it to the input signal

# Step 1: Quantizing

Example:

You have 0-10V signals.  
Separate them into a set  
of discrete states with  
**1.25V increments.** (How  
did we get 1.25V?)

(Discussed in previous slide)

Output States	Discrete Voltage Ranges (V)
0	0.00-1.25
1	1.25-2.50
2	2.50-3.75
3	3.75-5.00
4	5.00-6.25
5	6.25-7.50
6	7.50-8.75
7	8.75-10.0

## Step 2. Encoding

- Here we assign the digital value (binary number) to each state for the computer to read.

Output States	Output Binary Equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

# Sampling

- It is a process of taking a sufficient number of discrete values at point on a waveform that will define the shape of waveform.
- The more samples you take, the more accurately you will define the waveform.
- It converts analog signal into series of impulses, each representing amplitude of the signal at given point.....

# Sampling

---

Collect sufficient data for correctly representing a continuous-time signal



(a)



(b)



(c)

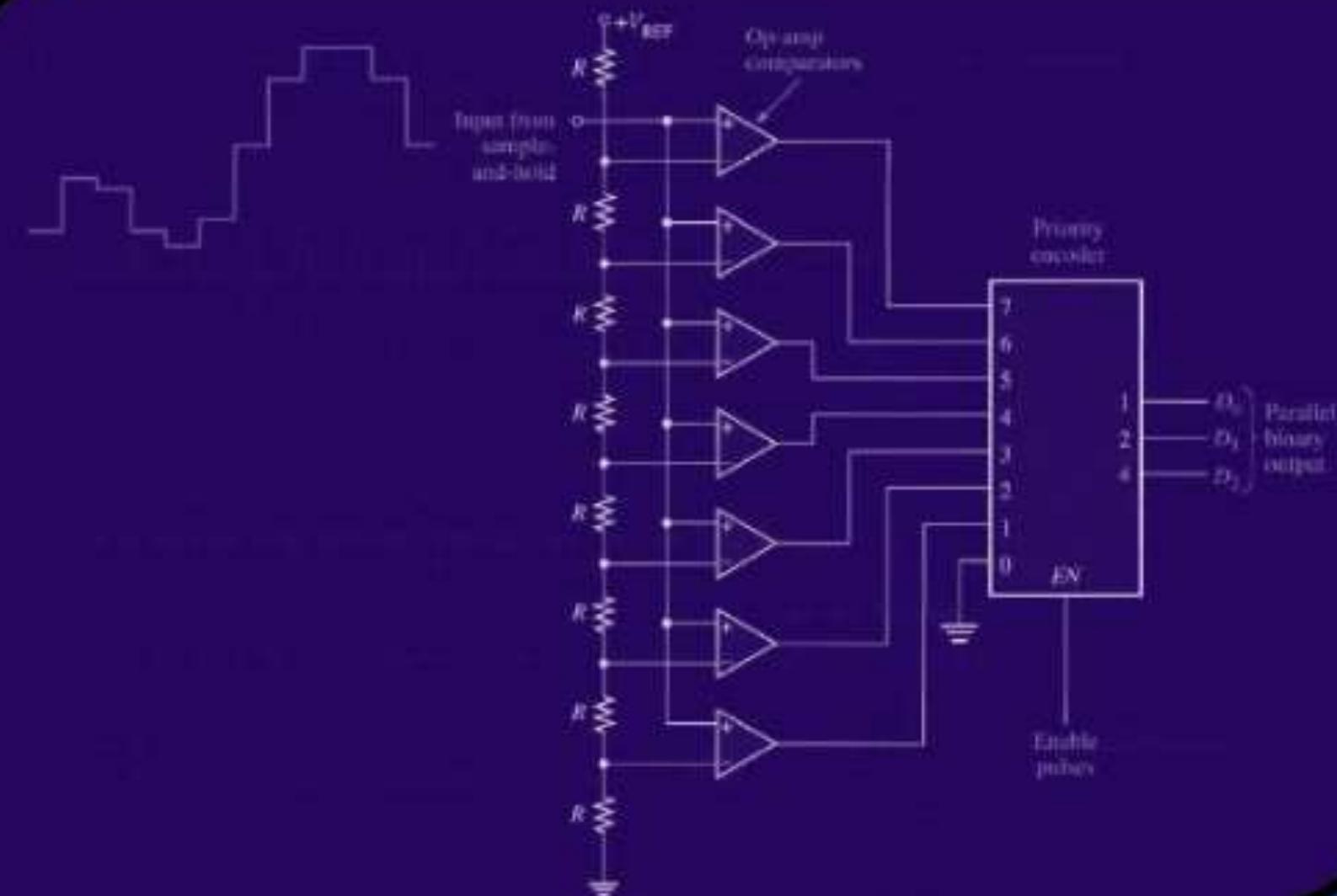
## 3 Basic Types

- Flash ADC
- Digital-Ramp/Dual slope/Counter slope ADC
- Successive Approximation ADC

## 1-> Flash ADC

- Consists of a series of comparators, each one comparing the input signal to a unique reference voltage.
- The comparator outputs connect to the inputs of a priority encoder circuit, which produces a binary output

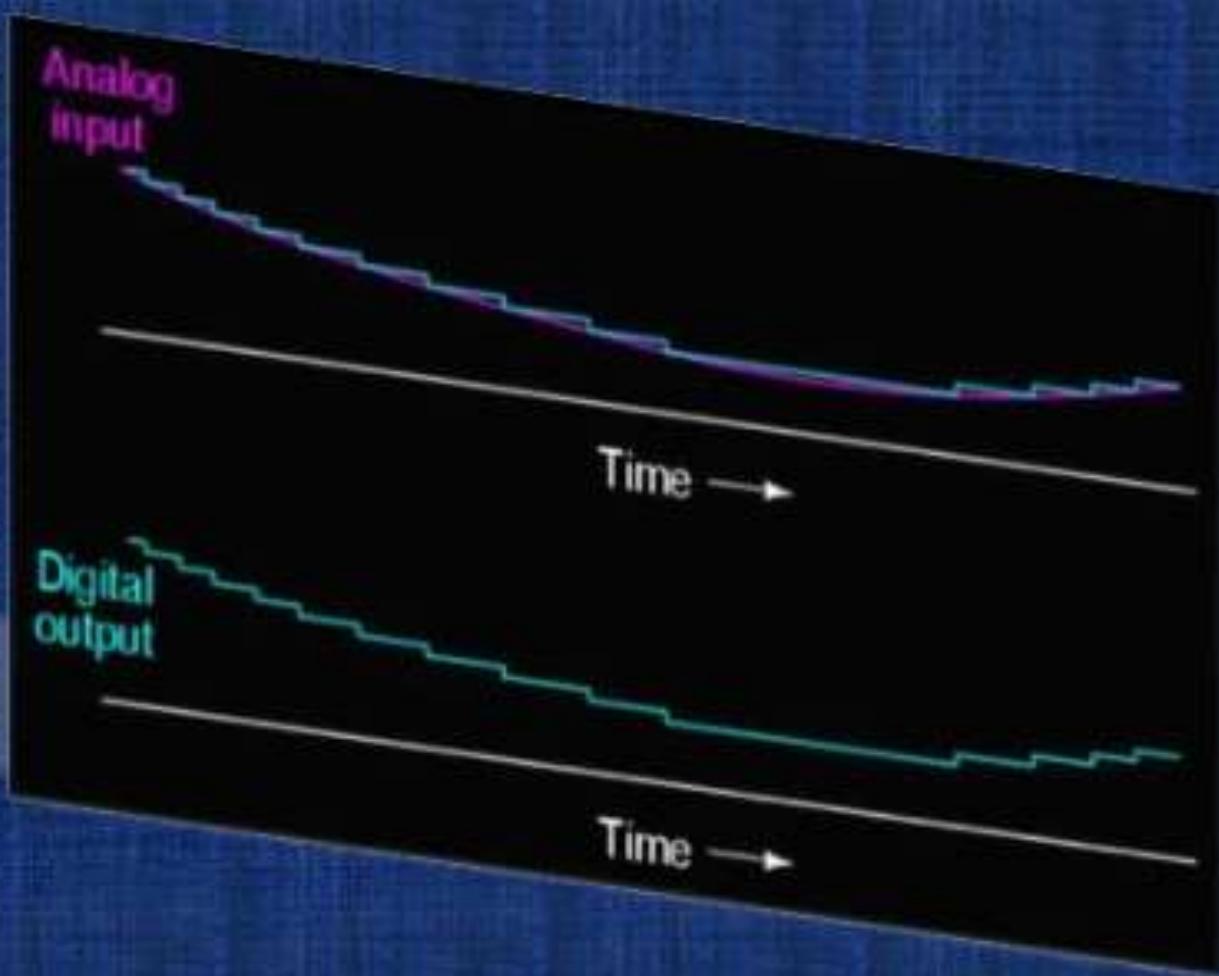
# 3 bit Flash ADC Circuit



# How Flash Works

- As the analog input voltage exceeds the reference voltage at each comparator, the comparator outputs will sequentially saturate to a high state.
- The priority encoder generates a binary number based on the highest-order active input, ignoring all other active inputs.

# ADC Output



# Flash

## *Advantages*

- Simplest in terms of operational theory
- Most efficient in terms of speed, very fast
  - limited only in terms of comparator and gate propagation delays

## *Disadvantages*

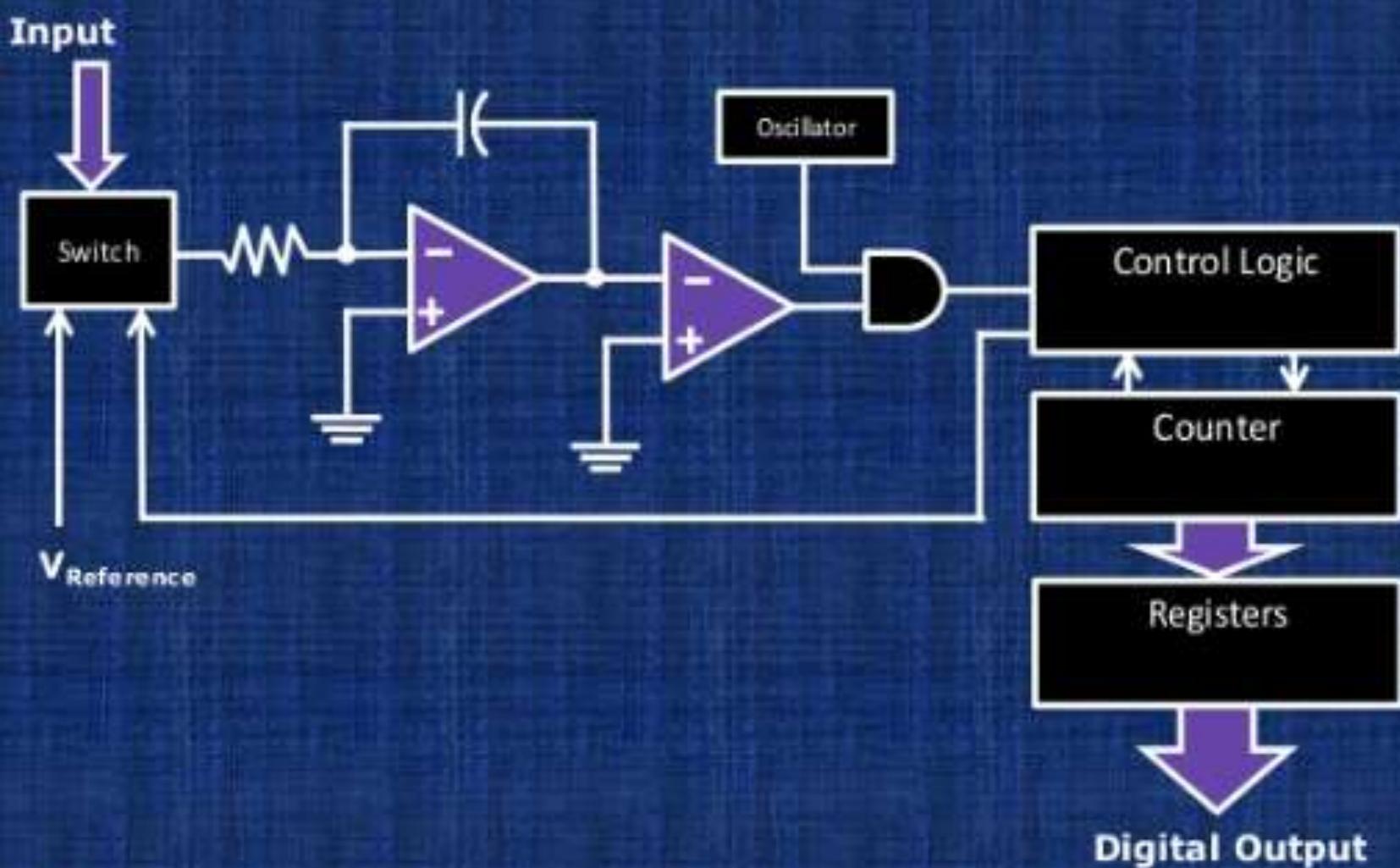
- Lower resolution
- Expensive
- For each additional output bit, the number of comparators is doubled
  - i.e. for 8 bits, 256 comparators needed

2->

## Dual Slope ADC

- Also known as Counter-Ramp or Digital Ramp ADC
- A dual slope ADC is commonly used in measurement instruments (such as DVM's).

# Dual Slope ADC circuit

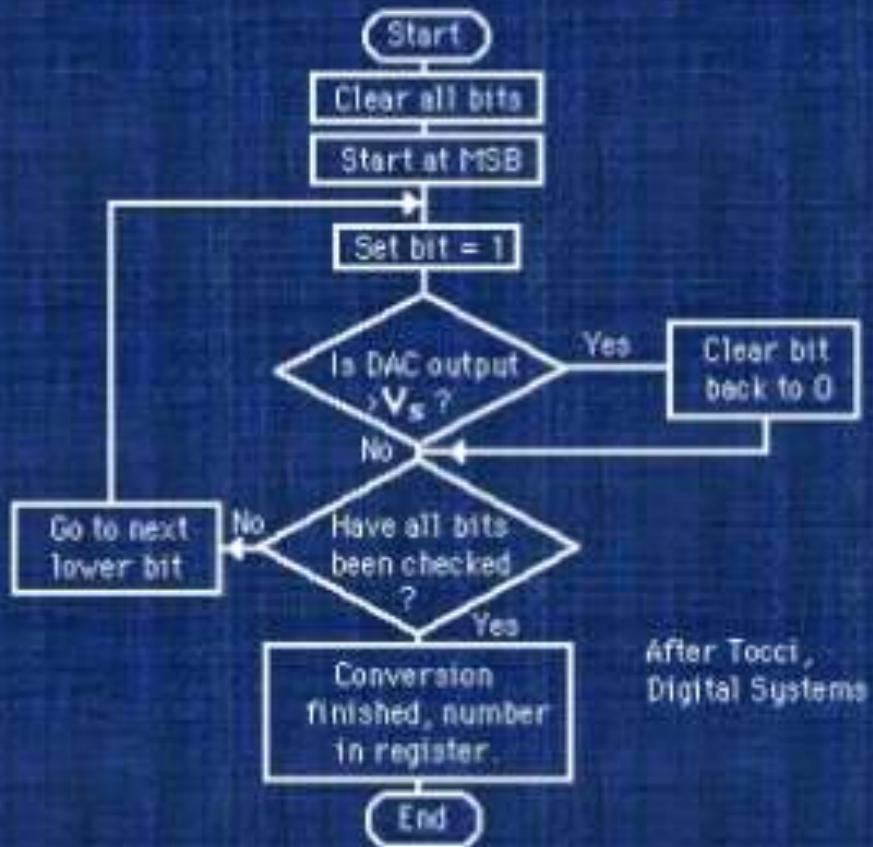


## Dual Slope Function

- The Dual Slope ADC functions in this manner:
  - When an analog value is applied the capacitor begins to charge in a linear manner and the oscillator passes to the counter.
  - The counter continues to count until it reaches a predetermined value. Once this value is reached the count stops and the counter is reset. The control logic switches the input to the first comparator to a reference voltage, providing a discharge path for the capacitor.
  - As the capacitor discharges the counter counts.
  - When the capacitor voltage reaches the reference voltage the count stops and the value is stored in the register.

# Successive approximation ADC

- Much faster than the digital ramp ADC because it uses digital logic to converge on the value closest to the input voltage.
- A comparator and a DAC are used in the process.

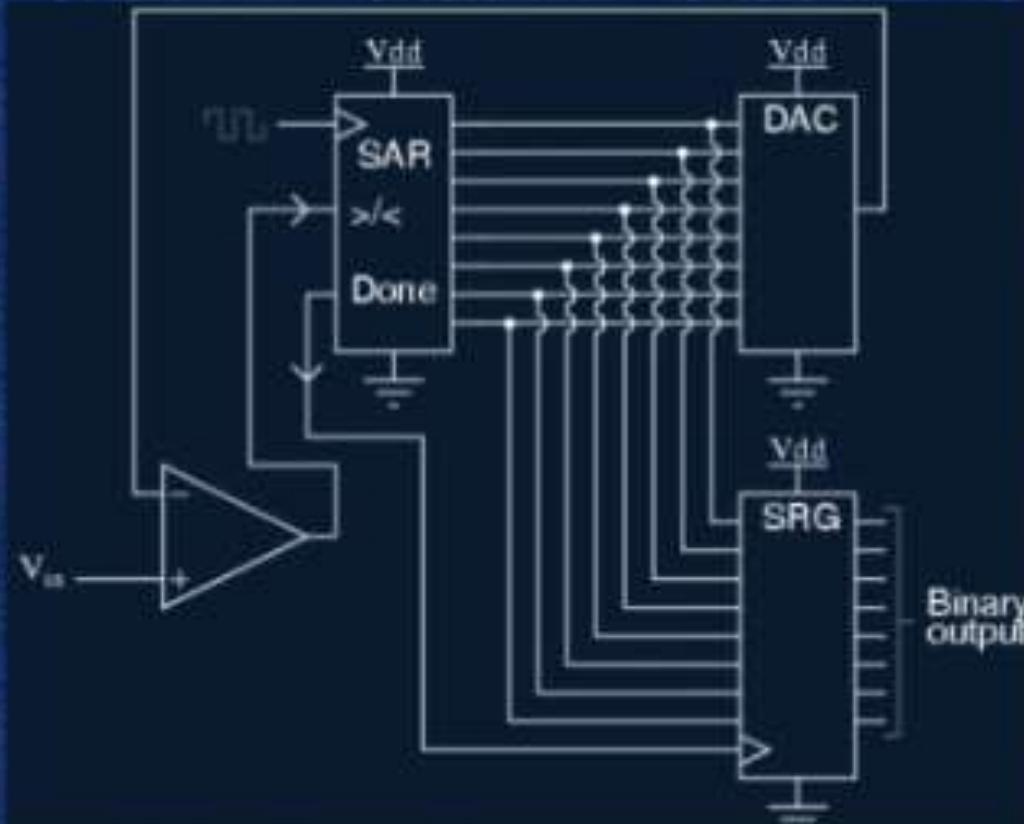


After Tocci,  
Digital Systems

## Successive Approximation ADC

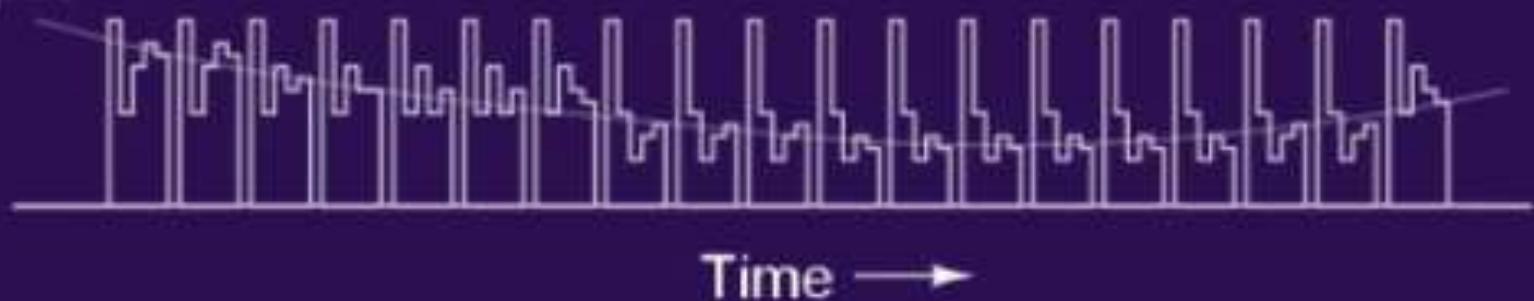
- A Successive Approximation Register (SAR) is added to the circuit
- Instead of counting up in binary sequence, this register counts by trying all values of bits starting with the MSB and finishing at the LSB.
- The register monitors the comparators output to see if the binary count is greater or less than the analog signal input and adjusts the bits accordingly

# Successive Approximation ADC Circuit



# Output

Analog  
input



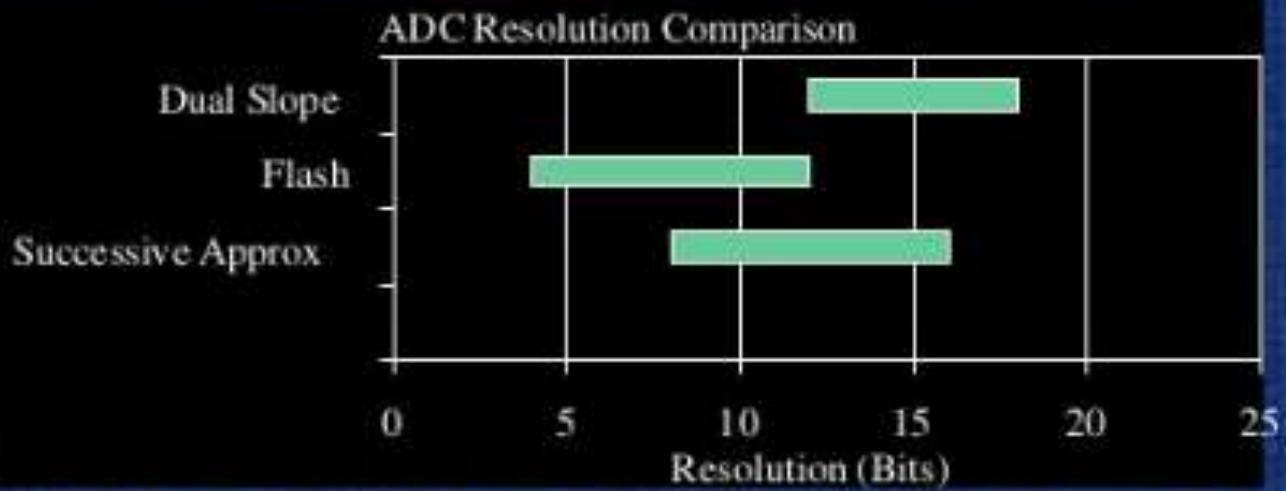
Time →

Digital  
output



Time →

# ADC Types Comparison



Type	Speed (relative)	Cost (relative)
Dual Slope	Slow	Med
Flash	Very Fast	High
Successive Approx	Medium – Fast	Low

# Examples of A/D Applications

- **Microphones** – take your voice varying pressure waves in the air and convert them into varying electrical signals
- **Strain Gages** - determines the amount of strain (change in dimensions) when a stress is applied
- **Thermocouple** – temperature measuring device converts thermal energy to electric energy
- **Voltmeters**
- **Digital Multimeters**