

## Week No. 07: KARNAUGH MAP

### Karnaugh Map:

- A Karnaugh Map provides a systematic method for simplifying Boolean expressions and will produce the simplest SOP or POS expression which is known as the minimum expression.
- The purpose of a Karnaugh Map is to simplify a Boolean expression.
- A KM is similar to a truth table because it presents all of the possible values of input variables and the resulting output for each value.
- The KM is an array of cells in which each cell represents a binary value of the input variables.
- The cells are arranged in a way so that simplification of a given expression is simply grouping the cells.
- KM can be used for expressions with two, three, four, and five variables.
- The number of cells in a KM as well as the number of rows in a truth table is equal to the total number of possible input variable combinations.
- For three variables, the number of cells is  $2^3 = 8$  and for four variables, the number of cells is  $2^4 = 16$ .

### 3 – Variable Karnaugh Map:

- The 3 – variable KM is an array of eight cells. In this case, A, B, and C are used for the variables and also other letters could be used.

	C	0	1
AB	00	000	001
01	01	010	011
11	11	110	111
10	10	100	101

	C	0	1
AB	00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	11	$AB\bar{C}$	$ABC$
10	10	$A\bar{B}\bar{C}$	$A\bar{B}C$

3 – Variable Karnaugh Map

### 4 – Variable Karnaugh Map:

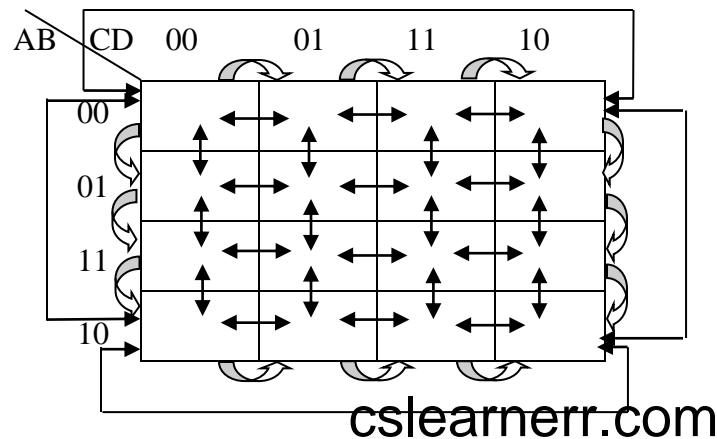
- The 4 – variable KM is an array of sixteen cells. In this case, variables are A, B, C, and D.

	CD	00	01	11	10
AB	00	0000	0001	0011	0010
01	01	0100	0101	0111	0110
11	11	1100	1101	1111	1110
10	10	1000	1001	1011	1010

	CD	00	01	11	10
AB	00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}\bar{B}CD$
01	01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BC\bar{D}$	$\bar{A}BCD$
11	11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABC\bar{D}$	$ABCD$
10	10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}C\bar{D}$	$A\bar{B}CD$

## Cell Adjacency:

- The cells in a KM are arranged so that there is only a single – variable change between adjacent cells.
- Adjacency is defined by a single – variable change.
- Physically, each cell is adjacent to the cells that are immediately next to it on any of its four sides.
- A cell is not adjacent to the cells that diagonally touch any of its corners.
- Also the cells in the top row are adjacent to the corresponding cells in the bottom row.
- And cells in the outer left column are adjacent to the corresponding cells in the outer right column, this is called “Wrap – around” adjacency.
- Cells that differ by only one variable are adjacent.



## Karnaugh Map SOP Minimizations:

- Karnaugh Map is used for simplifying Boolean expressions to their minimum form.
- A minimized SOP expression contains the fewest possible terms with the fewest possible variables per term.
- Generally, a minimum SOP expression can be implemented with fewer logic gates than a standard expression.

## Mapping a Standard SOP expression:

- For an SOP expression in standard form, a 1 is placed on the KM for each product term in the expression.
- Each 1 is placed in a cell corresponding to the value of a product term. For example, for the product term  $\bar{A}\bar{B}C$ , a 1 goes in the 101 cell on a 3 – variable map.
- When an SOP expression is completely mapped. There will be a number of 1’s on the KM equal to the number of product terms in the standard SOP expression.
- The cells that do not have 1 are the cells for which the expression is 0.
- **Two steps of mapping:**
  1. Determine the binary value of each product term in the standard SOP expression.
  2. Place a 1 on the KM in the cell having the same value as the product term.

**For example:**  $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$   
 000    001    110    100

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

**Mapping a Non standard SOP expression:**

- A Boolean expression must first be in KM.
- If expression is not in standard form, then it must be converted to a standard form.

**For example:**  $\bar{A} + A\bar{B} + ABC\bar{C}$   
 $\bar{A}(B + \bar{B}) + A\bar{B}(C + \bar{C}) + ABC\bar{C}$   
 $\bar{A}B + A\bar{B} + A\bar{B}C + A\bar{B}\bar{C} + ABC\bar{C}$   
 $\bar{A}B(C + \bar{C}) + A\bar{B}(C + \bar{C}) + A\bar{B}C + A\bar{B}\bar{C} + ABC\bar{C}$   
 $\bar{A}BC + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + ABC\bar{C}$   
 011 010 001 000 101 100 110

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

**Karnaugh Map Simplification of SOP expressions:**

- The process that results in an expression containing the fewest possible terms with the fewest possible variables is called minimization.
- After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the 1's and determining the minimum SOP expression from the map.

**Example 1:**

$\bar{A}\bar{B}C + \bar{A}BC + \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + A\bar{B}\bar{C}$   
 101 011 001 000 100

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

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

←  $\bar{A}C$  (circled 1s at 00,1 and 01,1)  
 ←  $\bar{B}$  (1s at 10,0 and 10,1)

**Example 2:**

$\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D}$   
 $\bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D$   
 1000 0000 0100 1100 0011 1011 0010 0110 1110 1010

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

←  $\bar{D}$  (1s at 00,0 and 10,0)  
 ←  $\bar{B}C$  (1s at 00,1 and 10,1)

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

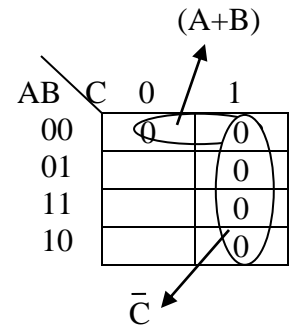
**Karnaugh Map POS Minimizations:**

- Place a 0 in the K – Map for each sum term in the expression, in the cell corresponding to the binary value of the sum term.
- Number of 0s in the KM is equal to number of sum terms in standard POS.

**Mapping and simplification of standard POS expressions:**

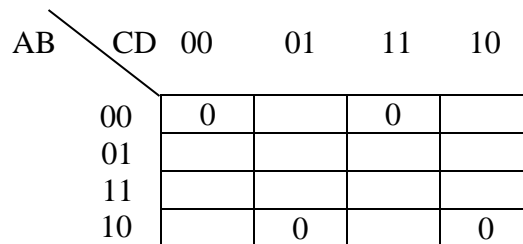
**Example 1:**  $(A+B+C) (A+B+\bar{C}) (A+\bar{B}+\bar{C}) (\bar{A}+B+\bar{C}) (\bar{A}+\bar{B}+\bar{C})$   
 $(0\ 0\ 0) (0\ 0\ 1) (0\ 1\ 1) (1\ 0\ 1) (1\ 1\ 1)$

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

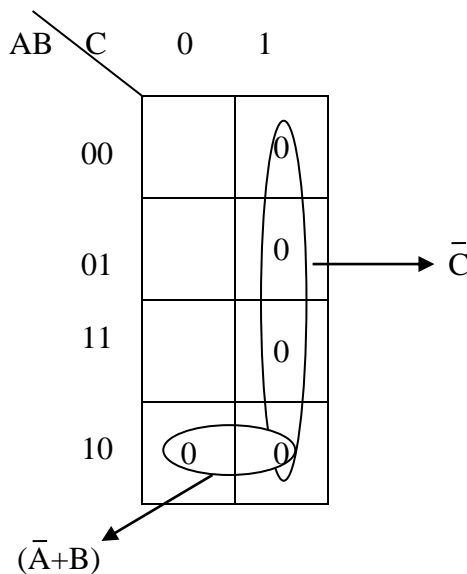


**Example 2:**  $(A+B+\bar{C}+\bar{D}) (A+B+C+D) (\bar{A}+B+\bar{C}+D) (\bar{A}+B+C+\bar{D})$   
 $(0\ 0\ 1\ 1) (0\ 0\ 0\ 0) (1\ 0\ 1\ 0) (1\ 0\ 0\ 1)$

So This POS expression is already simplified



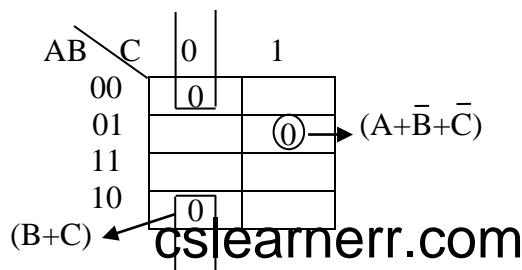
**Example 3:**  $(\bar{A}+B+C) (\bar{A}+B+\bar{C}) (A+B+\bar{C}) (\bar{A}+B+\bar{C}) (A+\bar{B}+\bar{C}) (\bar{A}+\bar{B}+\bar{C})$



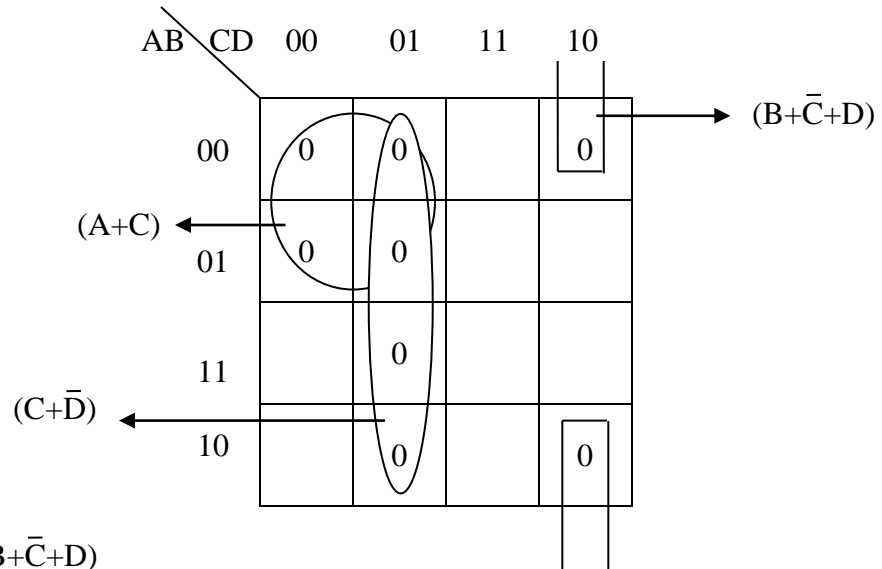
So  $(\bar{A}+B) (\bar{C})$

**Example 4:**  $(A+B+C) (\bar{A}+B+C) (A+\bar{B}+\bar{C})$

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

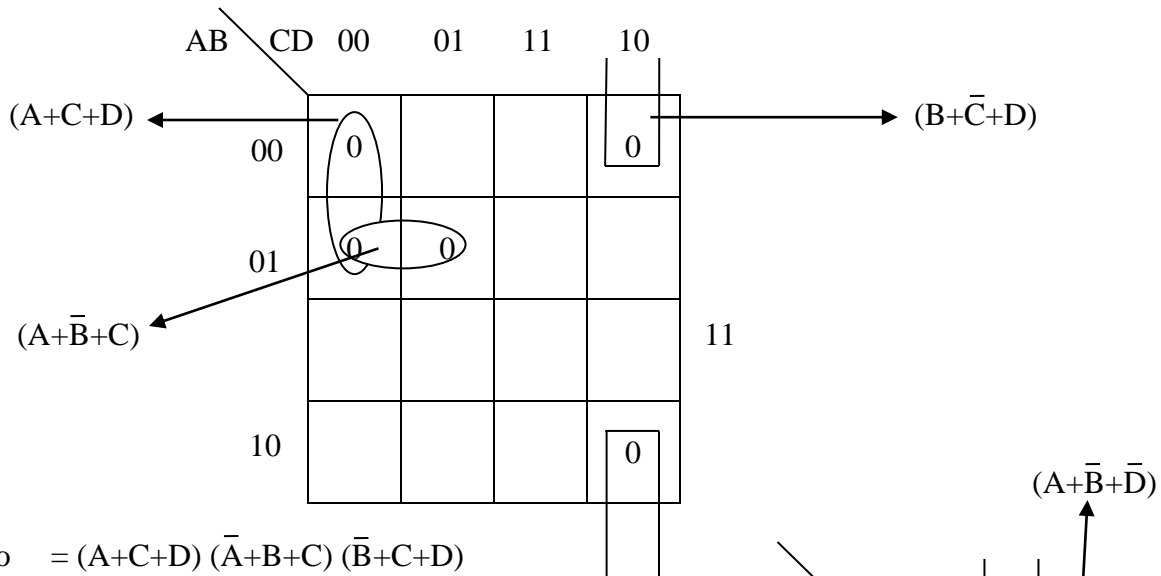


**Example 5:**  $(A+B+C+D) (A+B+C+\bar{D}) (A+\bar{B}+C+D) (A+\bar{B}+C+\bar{D}) (\bar{A}+\bar{B}+C+\bar{D}) (\bar{A}+B+C+\bar{D})$   
 $(A+B+\bar{C}+D) (\bar{A}+B+\bar{C}+D)$



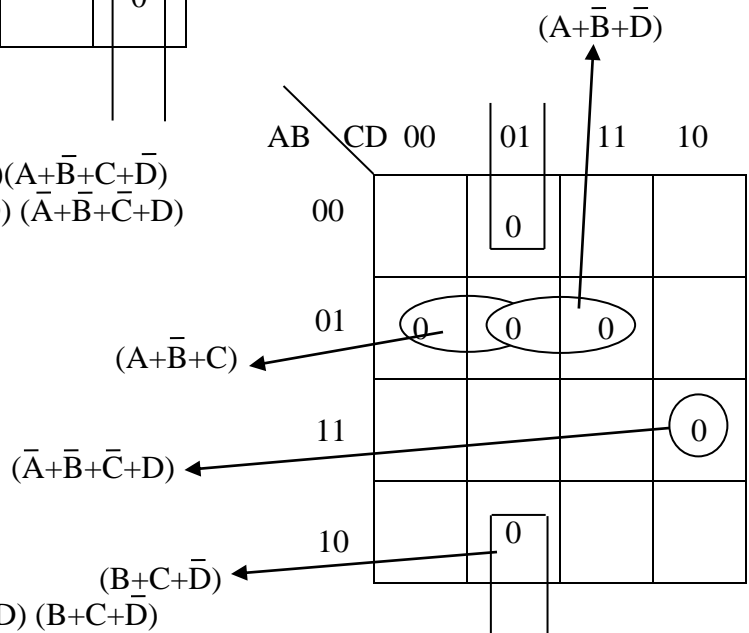
So  $= (A+C) (C+\bar{D}) (B+\bar{C}+D)$

**Example 6:**  $(A+B+C+D) (A+\bar{B}+C+D) (A+\bar{B}+C+\bar{D}) (A+B+\bar{C}+D) (\bar{A}+B+\bar{C}+D)$



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

**Example 7:**  $(A+B+C+\bar{D})(A+\bar{B}+C+D)(A+\bar{B}+C+\bar{D})$   
 $(A+\bar{B}+\bar{C}+\bar{D}) (\bar{A}+B+C+\bar{D}) (\bar{A}+\bar{B}+\bar{C}+D)$



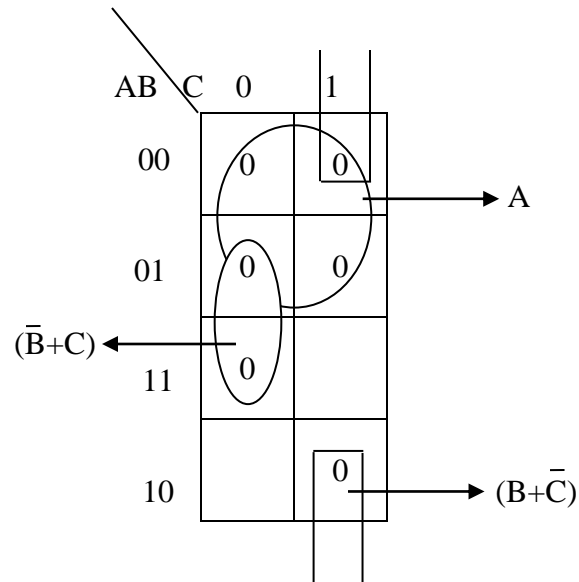
So  $= (A+\bar{B}+C) (A+\bar{B}+\bar{D}) (\bar{A}+\bar{B}+\bar{C}+D) (B+C+\bar{D})$

### Mapping and simplification of non-standard POS:

- A Boolean expression must be in a standard form before you are using a KM.
- If an expression is not in standard form then it must be converted to a standard form.

**Example:**  $(A+B)(A+\bar{B}+\bar{C})(\bar{A}+B+\bar{C})(\bar{B}+C)$

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



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



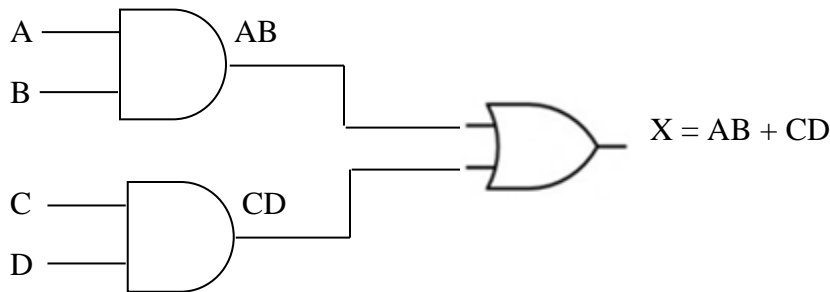
## Week No. 08: COMBINATIONAL LOGIC

- When two or more basic logic circuits works together or combine is called basic combinational logic circuits. The following are basic combinational logic circuits:
  - i. AND – OR logic
  - ii. AND – OR – Invert logic
  - iii. Exclusive – OR logic
  - iv. Exclusive – NOR logic

**Note:** X – OR & X- NOR gates are actually a form of AND – OR logic

### i. AND – OR logic:

- In general, an AND – OR circuit can have any number of AND gates, each with any number of inputs.
- AND – OR logic produces and SOP expression. For example, AND – OR logic circuit for the expression:  $X = AB + CD$  (SOP expression).



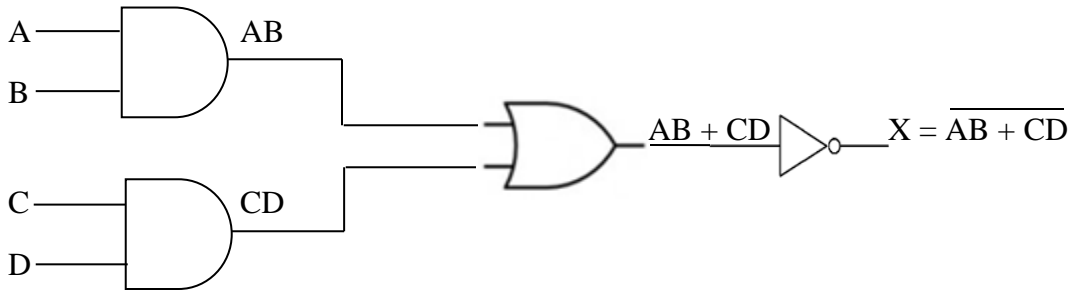
- Now Truth table for this expression:

A	B	C	D	AB	CD	X=AB+CD
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

- There are four variables are used as A, B, C, AND D, So  $N = 2^4 = 16$ .
- For a 4 – input AND – OR logic circuits, the output X is HIGH (1) if both A and B are HIGH (1) or both C and D are HIGH (1).

ii. **AND – OR – Invert logic:**

- When the output of an AND – OR circuit is complemented (inverted), it results in an AND – OR = Invert circuit.
- AND – OR logic directly implements SOP expression.
- POS expressions can be implemented with AND – OR – Invert logic. For example, AND – OR – Invert logic circuit for the expression:  $X = \overline{AB + CD} = (\overline{A + B})(\overline{C + D})$ .
- An AND – OR – Invert circuit produces a POS output.



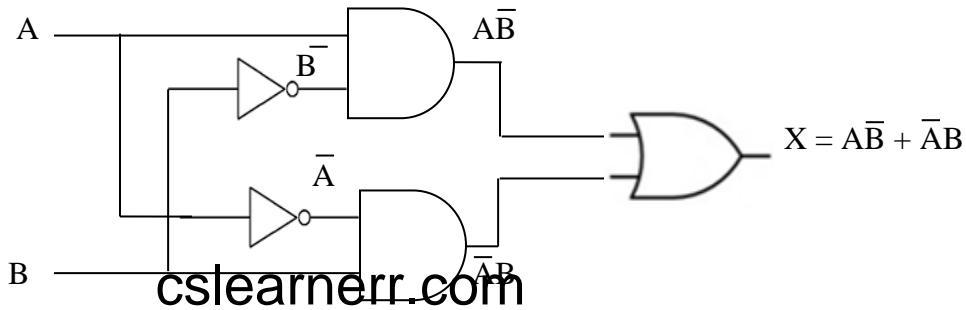
- For a 4 – input AND – OR – Invert logic circuit, the output X is LOW (0) if both input A and B are HIGH (1) or both C and D are HIGH (1).
- A Truth table can be developed from the AND – OR truth table, simply changing all 1's to 0's and all 0's to 1's in the output column.

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



iii. Exclusive – OR logic:

- The XOR gate is actually a combination of other gates.
- Because of its importance, this circuit is considered a type of logic gates with its own unique symbol, it is actually a combination of two AND gates, one OR gate and two inverters.



- Truth table for an Exclusive – OR logic for the above expression:

A	B	$\bar{A}B$	$A\bar{B}$	$X = \bar{A}B + A\bar{B}$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

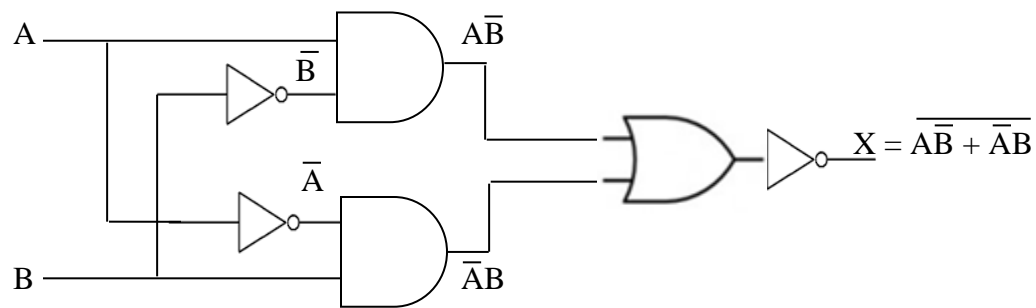
- Logic symbol for Exclusive – OR logic:



A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

iv. **Exclusive – NOR logic:**

- The complement of the XOR function is the Exclusive NOR.
- The exclusive – NOR can be implemented by simply inverting the output of an exclusive – OR. For example, Exclusive – NOR logic for the expression:  $X = \overline{A\overline{B}} + \overline{\overline{A}B}$



- Truth table for an Exclusive – NOR logic for the above expression:

A	B	$A\overline{B}$	$\overline{A}B$	$A\overline{B} + \overline{A}B$	$X = \overline{A\overline{B} + \overline{A}B}$
0	0	0	0	0	1
0	1	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	1

- Logic symbol for Exclusive – NOR logic:

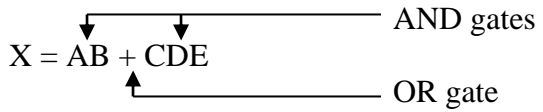


A	B	$A \oplus B$	$\overline{A \oplus B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

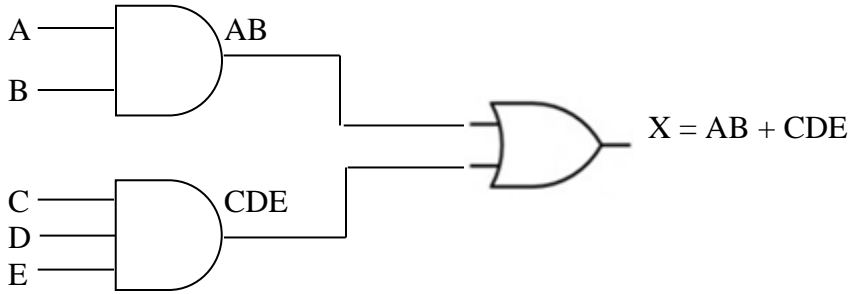
**Implementing Combinational Logic:**

- Now how we will implement a combinational logic circuit.

**From a Boolean Expression to a Logic Circuit:**

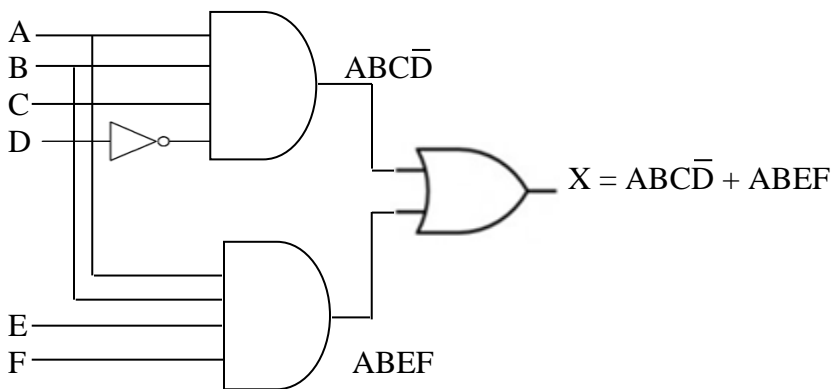
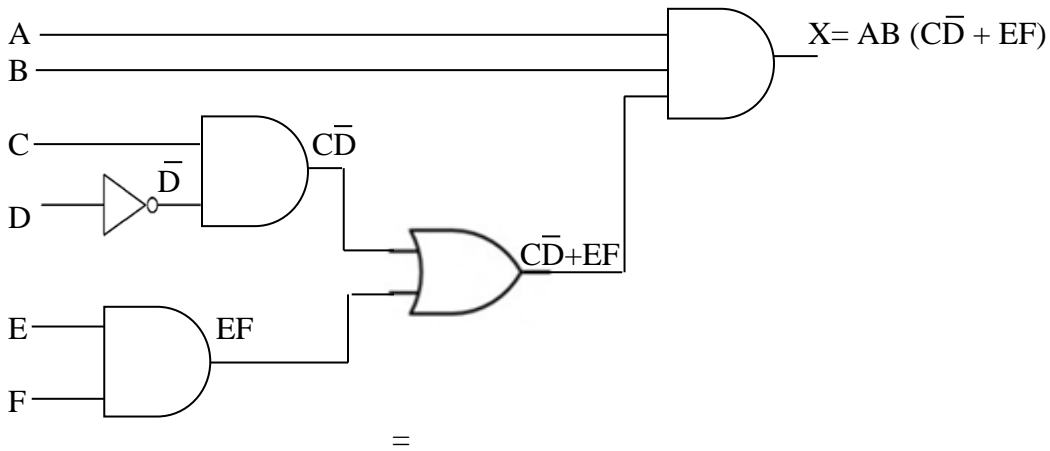
- Let's examine the Boolean expression:  $X = AB + CDE$   
 $X = AB + CDE$   


- Logic Circuit for the above expression:



- For every Boolean expression, there is a logic circuit and for every logic circuit, there is a Boolean expression.
- Now let's take another example, implement the following expression:

$$X = AB (C\bar{D} + EF) = ABC\bar{D} + ABEF$$

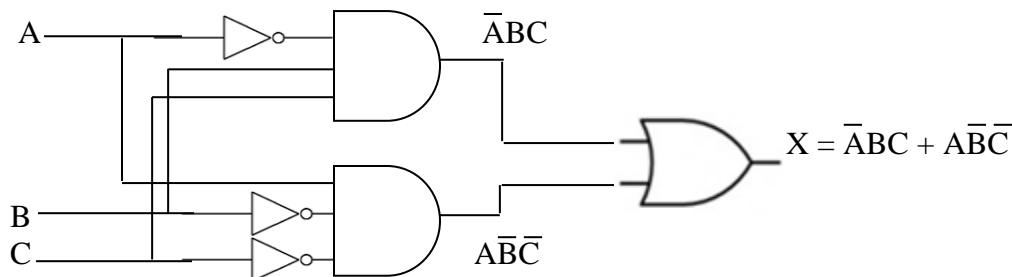


**From a Truth Table to a Logic Circuit:**

- The Boolean SOP expression obtained from the truth table by ORing the Product terms which X = 1.
- From the following truth table we obtained the SOP expression:  $X = \bar{A}BC + A\bar{B}\bar{C}$

A	B	C	X	Product term
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC$
1	0	0	1	$A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	0	

- Now Logic Circuit of the above expression:

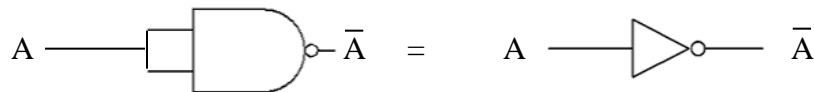


**The Universal Property of NAND and NOR gates:**

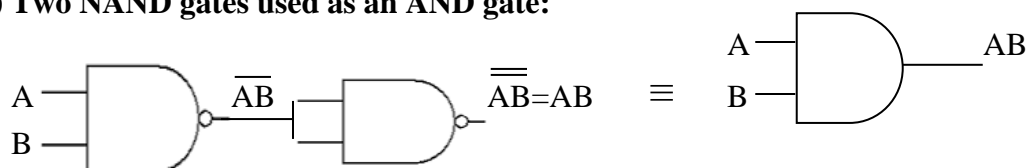
- **The NAND gate as a universal logic circuit:**

- The NAND gate is a universal gate because it can be used to produce the NOT, the AND, the OR and the NOR functions
- Combinations of NAND gates can be used to produce any logic function.
- The following are the universal properties of NAND gate:

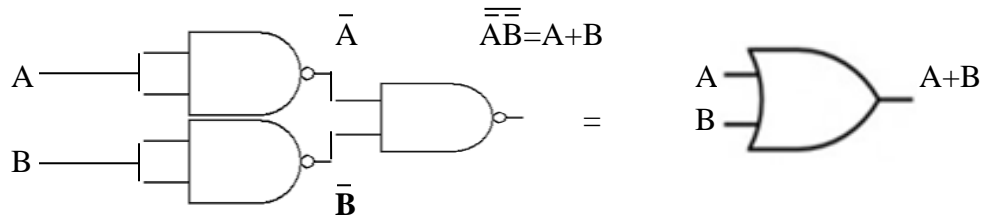
a) One NAND gate used as an inverter:



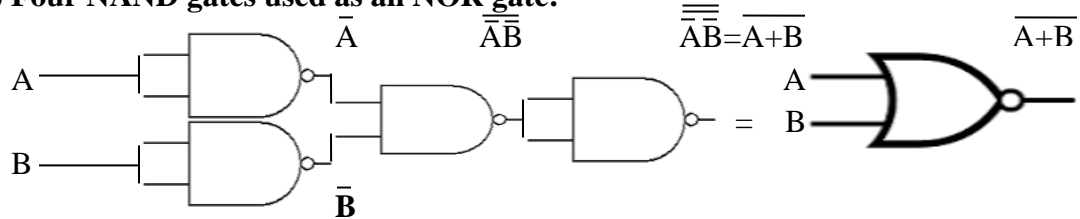
b) Two NAND gates used as an AND gate:



c) **Three NAND gates used as an OR gate:**



d) **Four NAND gates used as an NOR gate:**



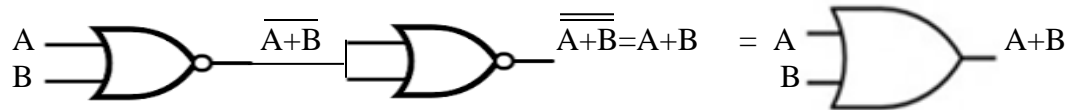
• **The NOR gate as a universal logic circuit:**

- Like the NAND gate, the NOR gate can be used to produce the NOT, AND, OR, and NAND functions.
- Combinations of NOR gates can be used to produce any logic function.
- The following are the universal properties of NOR gate:

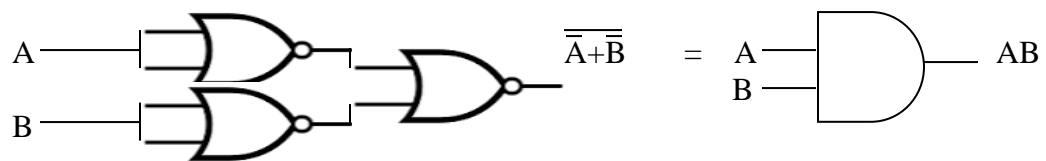
a) **One NOR gate used as an inverter:**



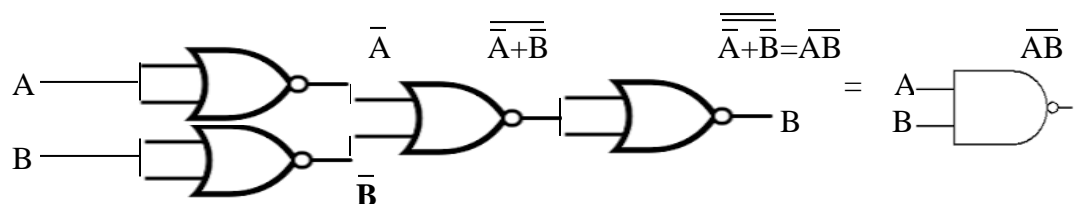
b) **Two NOR gate used as an OR gate:**



c) **Three NOR gate used as an AND gate:**



d) **Four NOR gate used as an NAND gate:**



## Combinational Logic using NAND and NOR gates:

- NAND Logic:** A NAND gate can function as either a NAND or a negative – OR by De Morgan Theorem. For example:

$$\overline{AB} = \overline{A + B} \quad \leftarrow \text{By De Morgan's Theorem}$$

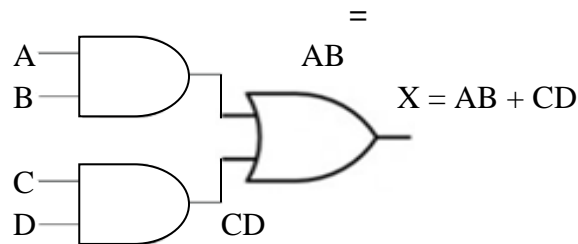
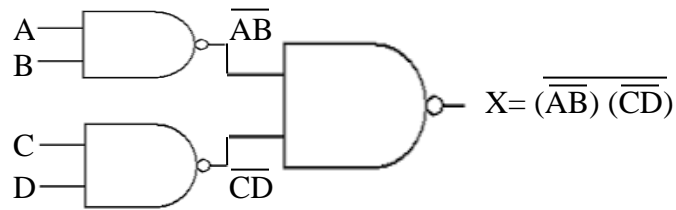
$\uparrow$                        $\uparrow$   
 Negative – OR gate  
 NAND gate

Now expression:  $X = \overline{\overline{AB}} (\overline{CD}) = \overline{(\overline{A + B}) (\overline{C + D})}$

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

$$= \overline{\overline{AB} + \overline{CD}}$$

$$= AB + CD$$



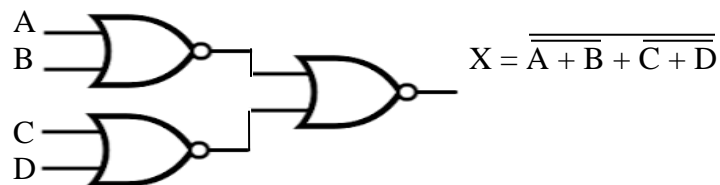
- NOR Logic:** A NOR gate can function as either a NOR or a negative – AND by De Morgan's Theorem. For example:

$$\overline{A + B} = \overline{AB} \quad \leftarrow \text{By De Morgan's Theorem}$$

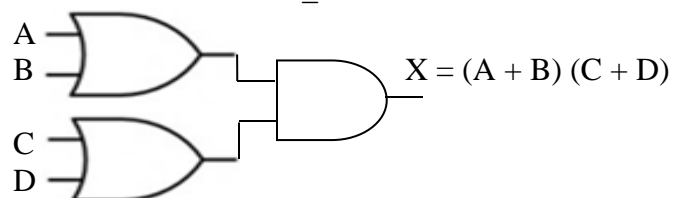
$\uparrow$                        $\uparrow$   
 Negative - AND  
 NOR gate

Now NOR logic circuit for the following expression:

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



cslearnerr.com



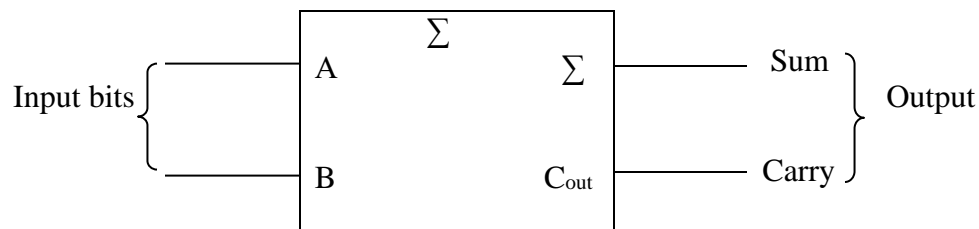
## Week No. 09: BASIC ADDERS

- Adders are important in computers and also in other types of digital systems in which numerical data are processed. There are two types of adders:

1. Half Adder
2. Full Adder

### 1. Half Adder:

- A half adder adds two bits and produces a sum and an output carry.
- The operations are performed by a logic circuit called a half adder.
- The half adder accepts two binary digits on its inputs and produces two binary digits on its output sum bit and a carry bit.
- Logic symbol for a half adder:

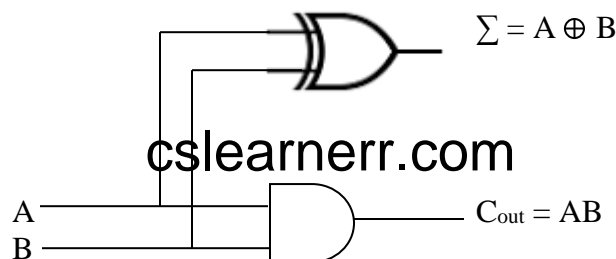


- Truth table for half adder:

$\Sigma$  = Sum  
 $C_{out}$  = Output Carry  
 A and B = Input Variables

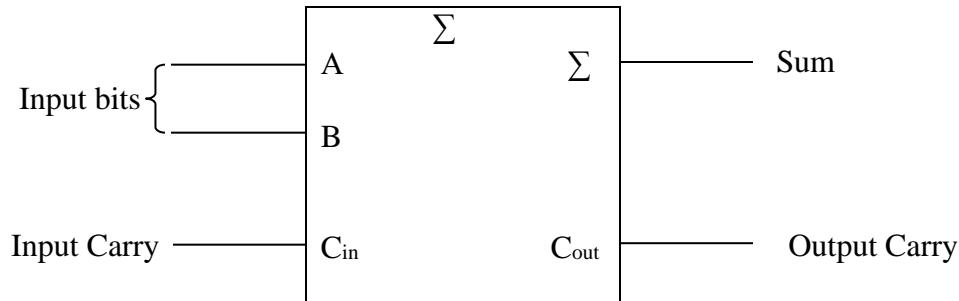
A	B	$C_{out}$	$\Sigma$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- The output carry ( $C_{out}$ ) is a 1 only when both A and B are 1's. Therefore  $C_{out}$  can be expressed as the AND of the input variables:  $C_{out} = AB$
- The sum output ( $\Sigma$ ) is a 1 only if input variables A and B are not equal. Therefore the sum can be expressed as the Exclusive – OR of the input variables:  $\Sigma = A \oplus B$
- Half Adder logic diagram:



## 2. Full Adder:

- The second category of adder is the full adder.
- A full adder has an input carry while the half adder does not.
- The full adder accepts two input bits and an input carry and generates a sum output and an output carry.
- The basic difference between a full adder and a half adder is that the full adder accepts an input carry.
- A logic symbol for a full adder:

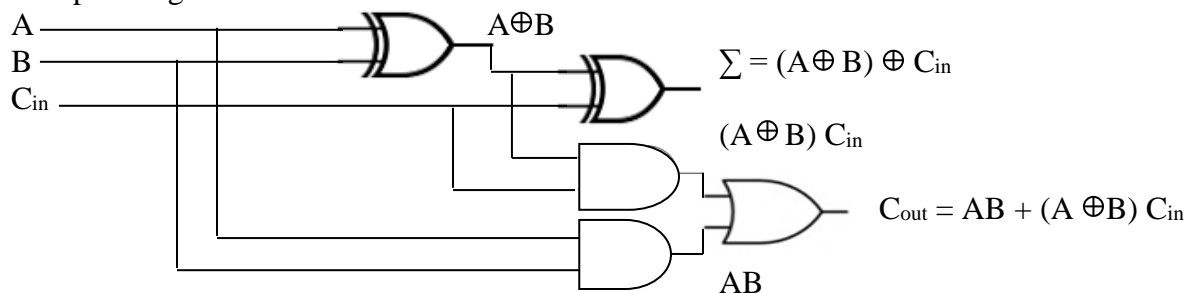


- Truth table for a full adder:

$C_{in}$  = Input Carry  
 $C_{out}$  = Output Carry  
 $\Sigma$  = Sum  
 A and B = Input variables

A	B	$C_{in}$	$C_{out}$	$\Sigma$
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

- For the input carry ( $C_{in}$ ) to be added to the input bits, it must be exclusive – OR<sub>ed</sub> with  $A \oplus B$ , yielding the equation for the sum output of the full – adder:  $\Sigma = (A \oplus B) \oplus C_{in}$
- This means that to implement the full adder function, two 2 – inputs exclusive – OR gates can be used.
- The first must generate the term  $A \oplus B$ , and second has as its inputs the output of the first XOR gate and the input carry.
- Complete logic circuit for a full adder:

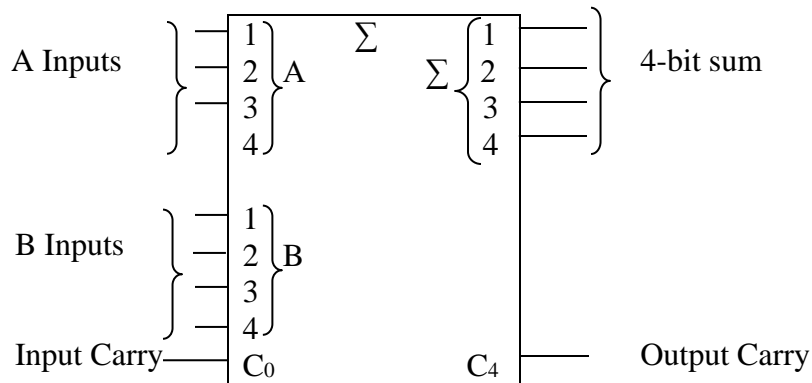


- The output carry is a 1 when both inputs to the first XOR gate are 1's or when both inputs to the second XOR gate are 1's.
- The output carry of the full adder is therefore produced by input A AND<sub>ed</sub> with input B and  $A \oplus B$  AND<sub>ed</sub> with  $C_{in}$  and these two terms are OR<sub>ed</sub>:  $C_{out} = AB + (A \oplus B) C_{in}$

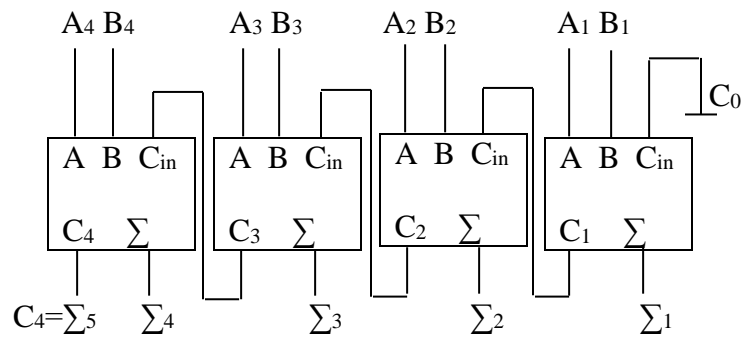




## 4 – Bit Parallel Adders:



(a) Logic Symbol



(b) Block Diagram

- Truth table for a 4 – bit Parallel Adders:

$C_{n-1}$	$A_n$	$B_n$	$\Sigma_n$	$C_n$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

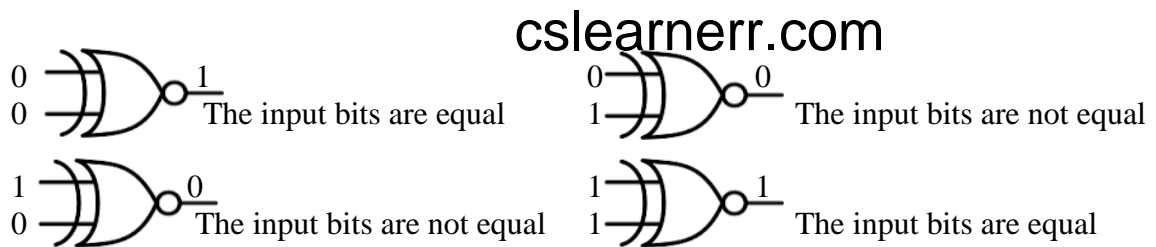
- The subscript n represents the adder bits and can be 1, 2, 3, or 4 adders.
- $C_{n-1}$  is the carry from previous adder
- Carries  $C_1$ ,  $C_2$ , and  $C_3$  are generated internally.
- $C_0$  is an external carry input.
- $C_4$  is the output carry.

## Comparators:

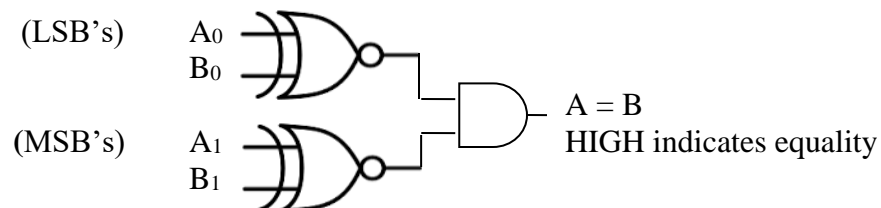
- The basic function of a comparator is to compare the magnitudes of two binary quantities to determine the relationship of those quantities.
- In its simplest form, a comparator circuit determines whether two numbers are equal.

### Equality:

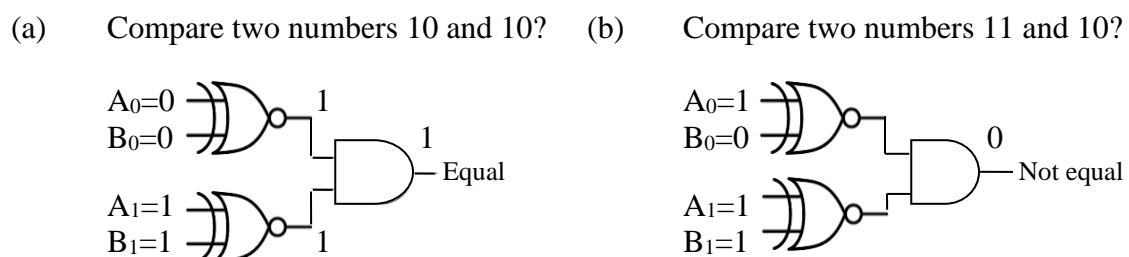
- The exclusive – NOR gate can be used as a basic comparator because its output is a 0 if the two input bits are not equal and a 1 if the input bits are equal. The following figure shows the exclusive – NOR gate as a 2 – bit comparator.



- In order to compare binary numbers containing two bits each, an additional exclusive – NOR gate is necessary.
- The two LSB's of the two numbers are compared by gate  $G_1$  and the two MSB's are compared by gate  $G_2$ .
- If the two numbers are equal, their corresponding bits are the same, and the output of each exclusive – NOR gate is a 1.
- If the corresponding sets of bits are not equal, a 0 occurs on that exclusive – NOR gate output.
- **Logic diagram for equality comparison of two 2 – bit numbers:**

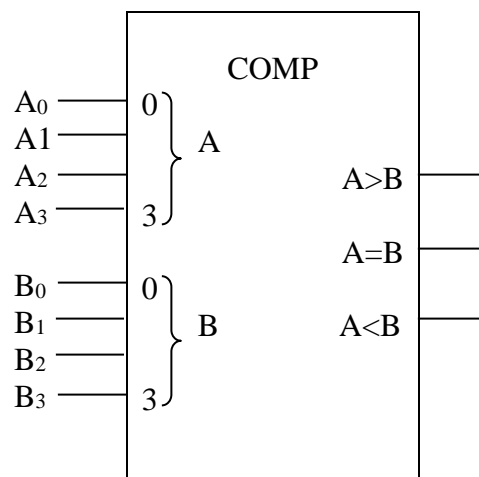


- In order to produce a single output indication an equality or inequality of two numbers, an AND gate can be combined with exclusive – NOR gates.
- Thus the output of the AND gate indicates equality (1) or inequality (0) of the two numbers. For example:



## Inequality:

- In addition to the equality output, many comparators provide an additional output that indicates which of the two binary numbers being compared is the larger.
- There is an output that indicates when number A is greater than B ( $A > B$ ) and an output that indicates when number A is less than number B ( $A < B$ ).
- To determine an inequality of binary numbers A and B, you first examine the highest order bit in each number. The following conditions are possible:
  1. If  $A_3 = 1$  and  $B_3 = 0$ , Number A is greater than B.
  2. If  $A_3 = 0$  and  $B_3 = 1$ , Number A is less than B.
  3. If  $A_3 = B_3 = 1$ , Then you must examine the next lower bit position for an in equality.
- These three operations are valid for each bit position in the numbers.
- **Logic symbol for a 4 – bit Comparator with inequality indication:**



- The general procedure used in a comparator is to check for an inequality in a bit position, starting with highest order bits (MSB's).
- When such an inequality is found, the relationship of the two numbers is established and any other inequalities in lower order bit positions must be ignored because it is possible for an opposite indication to occur, the highest order indication must take precedence.



Discover comprehensive notes for BSCS, BSIT and BSAI courses conveniently gathered on our user-friendly website, [www.cslearnerr.com](http://www.cslearnerr.com)

## Week No. 10: DECODERS

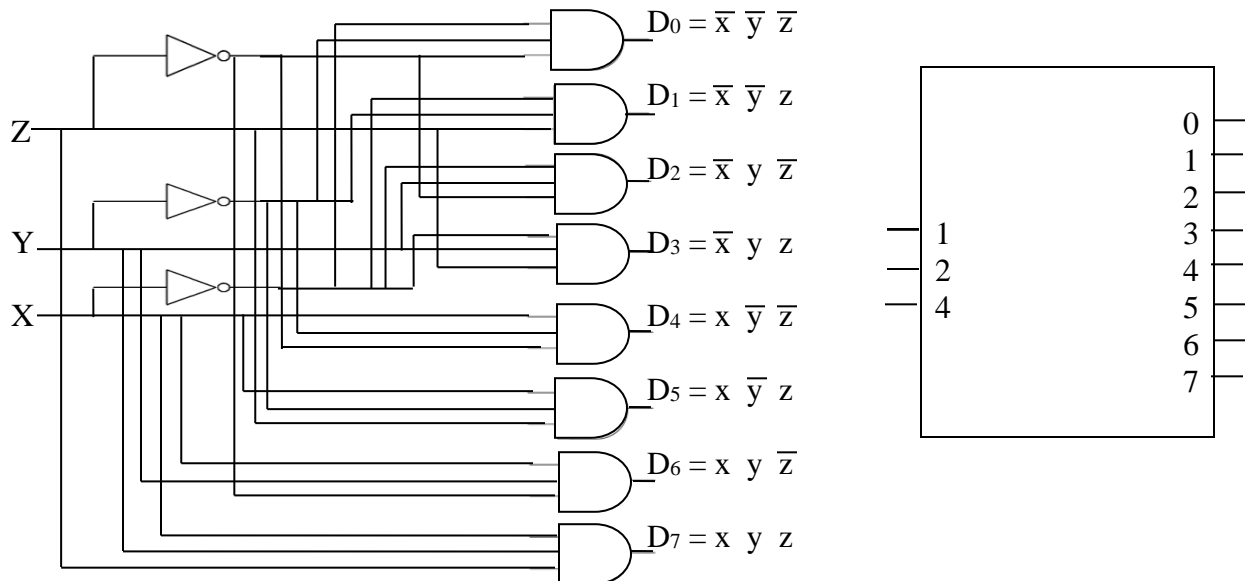
### Decoders:

- A decoder is a combinational circuit that converts binary information from n inputs lines to a maximum of  $2^n$  unique output lines. OR a decoder is a digital circuit that detects the presence of a specified combination of bits (code) on its inputs and indicates the presence of that code by a specified output level.
- The decoders are also called n (input) – to – m (maximum= $2^n$ ) line decoders where  $m < 2^n$ .
- An AND gate can be use as the basic decoding element because it produces a HIGH output only when all of its inputs are HIGH.
- A NAND also can be used as the basic decoding element with an active – LOW output.

### 3 – bit Decoder:

- In order to decode all possible combinations of 3 bits, eight decoding gates are required ( $2^3 = 8$ ). This type of decoder is commonly called a 3 – line – to – 8 – line decoder because there are 3 inputs and eight outputs.

- **Logic symbol & diagram for 3 – line – to – 8 – line Decoder:**

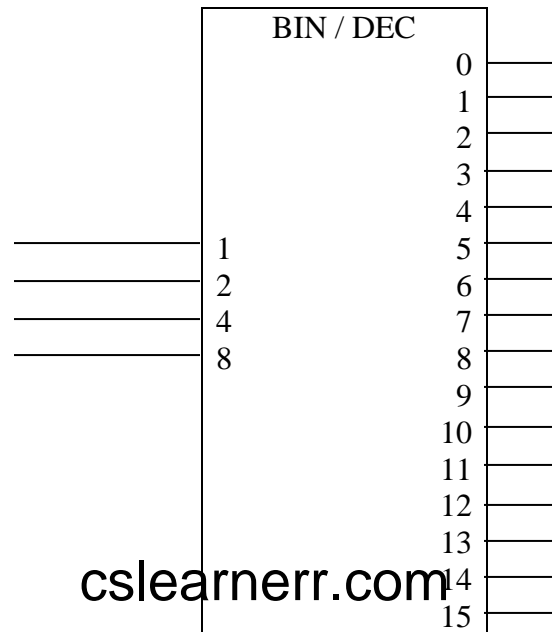


- **Truth table of 3 – line – to – 8 – line Decoder (Using AND gate with an active – HIGH output):**

Decoding Function	x	y	z	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
$D_0 = \bar{x} \bar{y} \bar{z}$	0	0	0	1	0	0	0	0	0	0	0
$D_1 = \bar{x} \bar{y} z$	0	0	1	0	1	0	0	0	0	0	0
$D_2 = \bar{x} y \bar{z}$	0	1	0	0	0	1	0	0	0	0	0
$D_3 = \bar{x} y z$	0	1	1	0	0	0	1	0	0	0	0
$D_4 = x \bar{y} \bar{z}$	1	0	0	0	0	0	0	1	0	0	0
$D_5 = x \bar{y} z$	1	0	1	0	0	0	0	0	1	0	0
$D_6 = x y \bar{z}$	1	1	0	0	0	0	0	0	0	1	0
$D_7 = x y z$	1	1	1	0	0	0	0	0	0	0	1

### 4 – bit Decoder:

- In order to decode all possible combinations of four bits, sixteen decoding gates are required ( $2^4 = 16$ ). This type decoder is commonly called either a 4 – lines – to – 16 – lines decoder or 1 – of – 16 decoder because for any given code on the inputs, one of the sixteen outputs is activated.
- **Logic symbol for a 4 – lines – to – 16 – lines Decoder:**



- If an active – LOW output is required for decoded number, the entire decoder can be implemented with NAND gates and Inverters.
- **Truth table for a 4 – lines – to – 16 – lines Decoder with active – LOW outputs:**
- In order to decode each of the sixteen binary codes, sixteen NAND gates are required (as AND gates can be used to produce active – HIGH outputs).

w	x	y	z	Decoding Function	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	D <sub>8</sub>	D <sub>9</sub>	D <sub>10</sub>	D <sub>11</sub>	D <sub>12</sub>	D <sub>13</sub>	D <sub>14</sub>	D <sub>15</sub>
0	0	0	0	$\bar{w} \bar{x} \bar{y} \bar{z} = D_0$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	$\bar{w} \bar{x} \bar{y} z = D_1$	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	$\bar{w} \bar{x} y \bar{z} = D_2$	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	$\bar{w} \bar{x} y z = D_3$	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	0	$\bar{w} x \bar{y} \bar{z} = D_4$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	$\bar{w} x \bar{y} z = D_5$	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	$\bar{w} x y \bar{z} = D_6$	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	1	1	1	$\bar{w} x y z = D_7$	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
1	0	0	0	$w \bar{x} \bar{y} \bar{z} = D_8$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	0	0	1	$w \bar{x} \bar{y} z = D_9$	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
1	0	1	0	$w \bar{x} y \bar{z} = D_{10}$	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
1	0	1	1	$w \bar{x} y z = D_{11}$	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
1	1	0	0	$w x \bar{y} \bar{z} = D_{12}$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
1	1	0	1	$w x \bar{y} z = D_{13}$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	1	1	0	$w x y \bar{z} = D_{14}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	$w x y z = D_{15}$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

### BCD – to – Decimal Decoder:

- The BCD – to – Decimal decoder converts each BCD code into one of ten possible decimal digit indications.
- It is referred as a 4 – line – to – 10 – lines decoder or 1 – of – 10 decoders.
- **Truth table for BCD – to – Decimal (4 – line – to – 10 line) Decoder with active – LOW outputs:**
- In order to decode each of the ten binary codes, ten NAND gates are required.

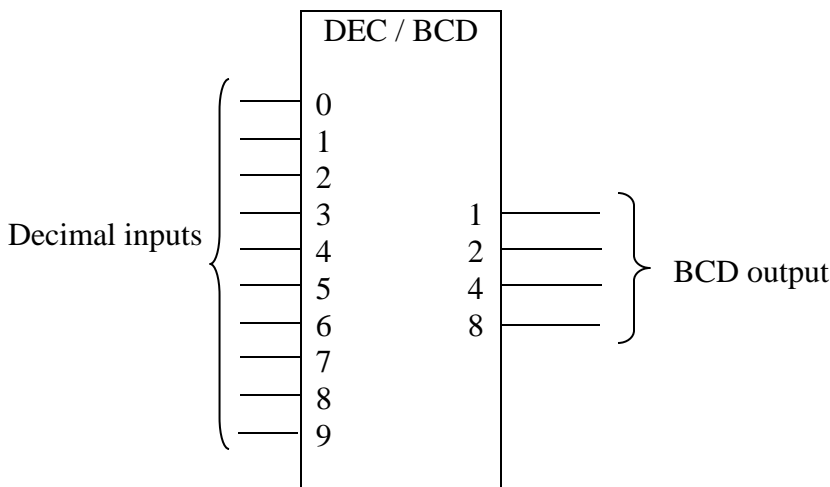
Decimal Digit	w	x	y	z	Decoding Function	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	D <sub>8</sub>	D <sub>9</sub>
0	0	0	0	0	$\bar{w} \bar{x} \bar{y} \bar{z} = D_0$	0	1	1	1	1	1	1	1	1	1
1	0	0	0	1	$\bar{w} \bar{x} \bar{y} z = D_1$	1	0	1	1	1	1	1	1	1	1
2	0	0	1	0	$\bar{w} \bar{x} y \bar{z} = D_2$	1	1	0	1	1	1	1	1	1	1
3	0	0	1	1	$\bar{w} \bar{x} y z = D_3$	1	1	1	0	1	1	1	1	1	1
4	0	1	0	0	$\bar{w} x \bar{y} \bar{z} = D_4$	1	1	1	1	0	1	1	1	1	1
5	0	1	0	1	$\bar{w} x \bar{y} z = D_5$	1	1	1	1	1	0	1	1	1	1
6	0	1	1	0	$\bar{w} x y \bar{z} = D_6$	1	1	1	1	1	1	0	1	1	1
7	0	1	1	1	$\bar{w} x y z = D_7$	1	1	1	1	1	1	1	0	1	1
8	1	0	0	0	$w \bar{x} \bar{y} \bar{z} = D_8$	1	1	1	1	1	1	1	1	0	1
9	1	0	0	1	$w \bar{x} \bar{y} z = D_9$	1	1	1	1	1	1	1	1	1	0

### Encoders:

- An encoder is a digital function that produces a reverse operation from that of a decoder. An encoder has  $2^n$  (or less) input lines and n output lines.
- The output lines generate the binary code for  $2^n$  input variables.
- OR gate can be used as the basic encoding element because it produces a HIGH output only when one of its inputs is HIGH.

### The Decimal – to – BCD Encoder:

- This type of encoder has ten inputs – one for each decimal digit and four outputs corresponding to the BCD code.
- This is a basic 10 – line – to – 4 line encoder.
- **Logic symbol for a Decimal – to – BCD encoder:**

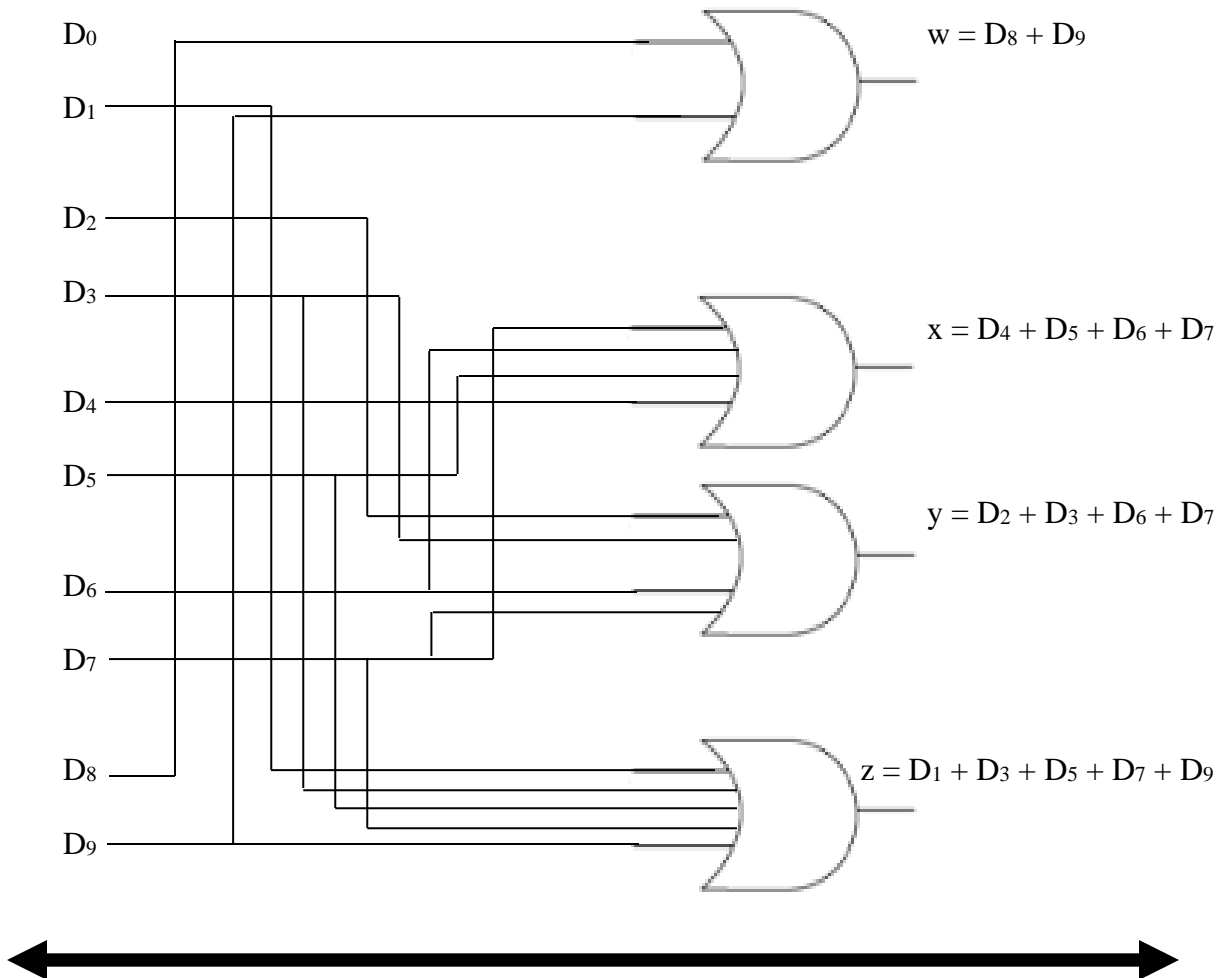


• Truth table of Decimal – to – BCD Encoder:

Decimal Digit	D <sub>0</sub>	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	D <sub>8</sub>	D <sub>9</sub>	w	x	y	z
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	1	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	1	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	1	0	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	0	0	0	0	1	0	1
6	0	0	0	0	0	0	1	0	0	0	0	1	1	0
7	0	0	0	0	0	0	0	1	0	0	0	1	1	1
8	0	0	0	0	0	0	0	0	1	0	1	0	0	0
9	0	0	0	0	0	0	0	0	0	1	1	0	0	1

- $w = D_8 + D_9$
- $x = D_4 + D_5 + D_6 + D_7$
- $y = D_2 + D_3 + D_6 + D_7$
- $z = D_1 + D_3 + D_5 + D_7 + D_9$

• Basic logic diagram of a Decimal – to – BCD Encoder:





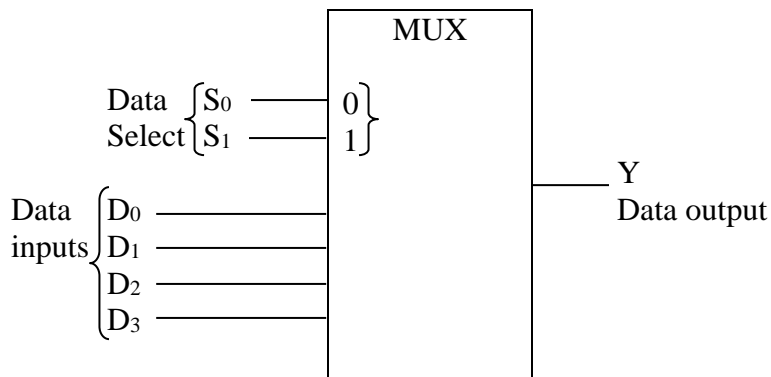
## Week No. 11: MULTIPLEXERS

### Multiplexers:

- A multiplexer (MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.
- The basic multiplexer has several data input lines and a single output line.
- Multiplexer also has data – select inputs, which permit digital data on any one of the inputs to be switched to the output line.
- Multiplexers are also known as data selectors.
- Normally, there are  $2^n$  input lines and 'n' selection lines whose bit combinations determine which input is selected.

### **4 – to – 1 line Multiplexer:**

- A logic symbol for a 4 – input multiplexer (MUX):



- In multiplexer, data goes from several lines to one line.
- **Note** that there are two data – select lines because with two select bits, any inputs one of the four data – input lines can be selected.

### **Data selection for a 4 – to – 1 line Multiplexer:**

- The data output is equal to  $D_0$  only if  $S_1 = 0$  and  $S_0 = 0$ ,  $Y = D_0 \bar{S}_1 \bar{S}_0$ .
- The data output is equal to  $D_1$  only if  $S_1 = 0$  and  $S_0 = 1$ ,  $Y = D_1 \bar{S}_1 S_0$ .
- The data output is equal to  $D_2$  only if  $S_1 = 1$  and  $S_0 = 0$ ,  $Y = D_2 S_1 \bar{S}_0$ .
- The data output is equal to  $D_3$  only if  $S_1 = 1$  and  $S_0 = 1$ ,  $Y = D_3 S_1 S_0$ .

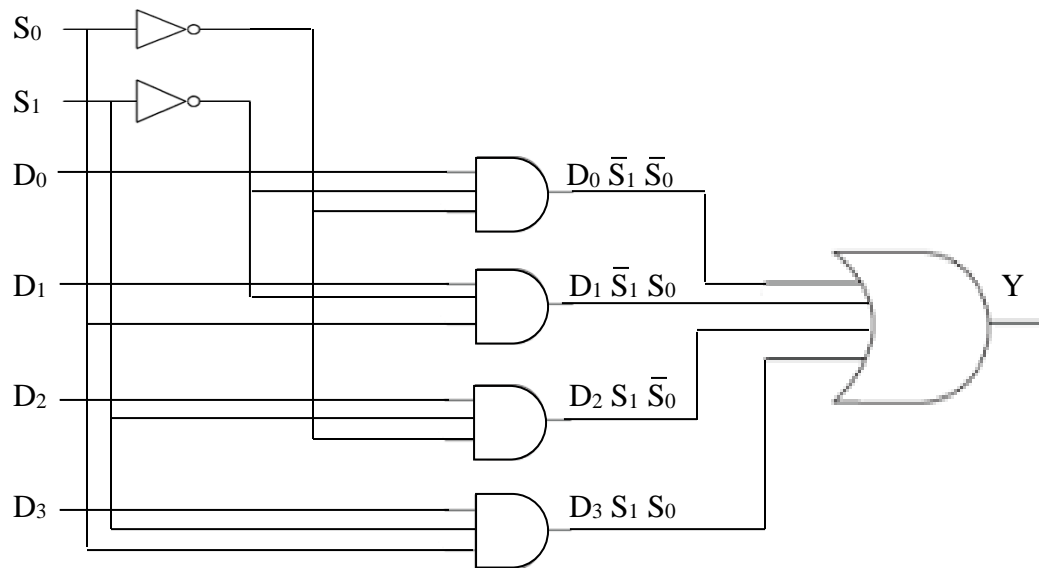
Data select inputs		Input switched
$S_1$	$S_0$	
0	0	$D_0$
0	1	$D_1$
1	0	$D_2$
1	1	$D_3$

- When these terms are OR<sub>ed</sub>, the total expression for the data output is:

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0$$

- The implementation of this equation requires four 3 – input AND gates, a 4 – input OR gate, and two inverters to generate the complements of  $S_1$  and  $S_0$ .
- Because data can be selected from any one of the input lines, the following circuit is also referred to as a data selector.

- **Logic diagram for a 4 – input multiplexer:**



$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0$$

**For example:**  $D_0 = 1, D_1 = 1, D_2 = 0, D_3 = 1$

- i.  $S_1 = 0$  and  $S_0 = 0$

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0 = 1.0.\bar{0} + 1.0.\bar{0} + 0.0.0 + 1.0.0$$

$$= 1.1.1 + 1.1.0 + 0.0.1 + 1.0.0 = 1 + 0 + 0 + 0 = 1$$

- ii.  $S_1 = 1$  and  $S_0 = 1$

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0 = 1.1.\bar{1} + 1.1.\bar{1} + 0.1.1 + 1.1.1$$

$$= 1.0.0 + 1.0.1 + 0.1.0 + 1.1.1 = 0 + 0 + 0 + 1 = 1$$

- iii.  $S_1 = 1$  and  $S_0 = 0$

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0 = 1.1.\bar{0} + 1.1.\bar{0} + 0.1.0 + 1.1.0$$

$$= 1.0.1 + 1.0.0 + 0.1.1 + 1.1.0 = 0 + 0 + 0 + 0 = 0$$

- iv.  $S_1 = 0$  and  $S_0 = 1$  [cslearnerr.com](http://cslearnerr.com)

$$Y = D_0 \bar{S}_1 \bar{S}_0 + D_1 \bar{S}_1 S_0 + D_2 S_1 \bar{S}_0 + D_3 S_1 S_0 = 1.0.\bar{1} + 1.0.\bar{1} + 0.0.1 + 1.0.1$$

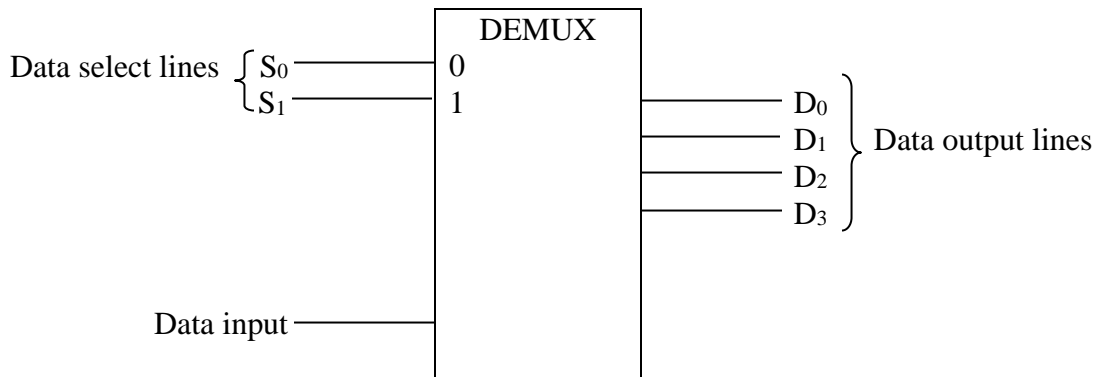
$$= 1.1.0 + 1.1.1 + 0.0.0 + 1.0.1 = 0 + 1 + 0 + 0 = 1$$

**Demultiplexer:**

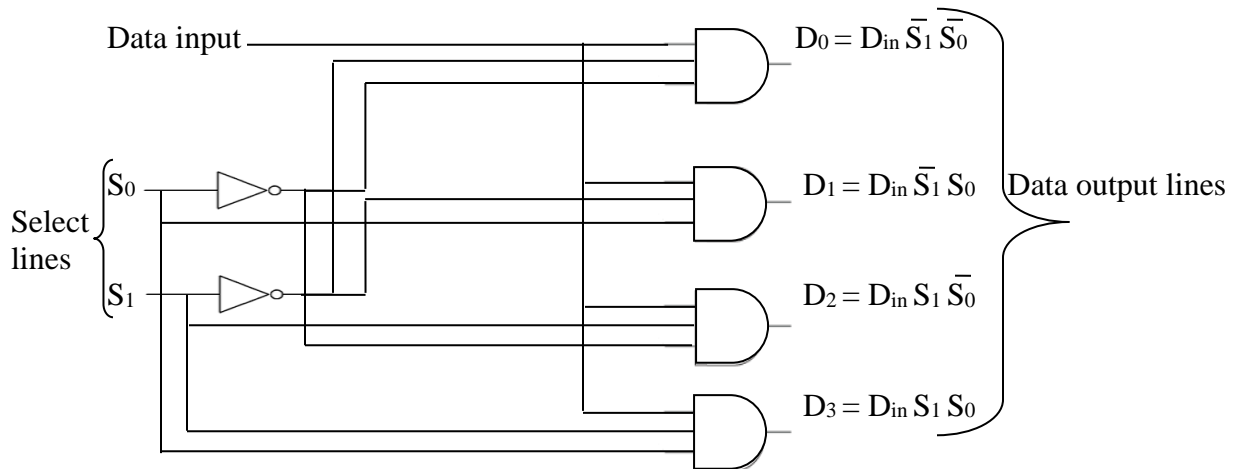
- A Demultiplexer (DEMUX) basically reverses the multiplexing function.
- DEMUX takes digital information from one line and distributes it to a given number of output lines.
- Demultiplexer is also known as a data distributor.
- Decoder can also be used as Demultiplexer.
- In a Demultiplexer, data goes from one line to several lines.

**1 – to – 4 line Demultiplexer:**

- The following figure shows a 1 – to – 4 line Demultiplexer (DEMUX) circuit.
- The data – input line goes to all of the AND gates.
- The two data – select lines enable only one gate at a time, and the data appearing on the data – input line will pass through the selected gate to the associated data – output line.
- **Logic symbol & logic diagram for 1 – to – 4 line Demultiplexer:**



**Logic symbol**



**Logic diagram**

**For example:**  $D_{in} = 10101$

- $S_1 = 0$  &  $S_0 = 0$  Then  $D_0 = D_{in} \bar{S}_1 \bar{S}_0 = 10101$ ,
- $S_1 = 0$  &  $S_0 = 1$  Then  $D_1 = D_{in} \bar{S}_1 S_0 = 10101$
- $S_1 = 1$  &  $S_0 = 0$  Then  $D_2 = D_{in} S_1 \bar{S}_0 = 10101$ ,
- $S_1 = 1$  &  $S_0 = 1$  Then  $D_3 = D_{in} S_1 S_0 = 10101$

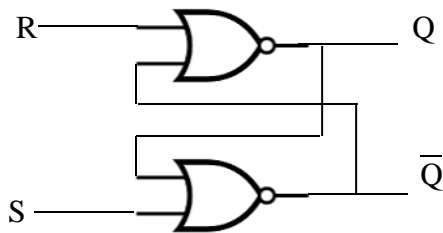


**Latches:**

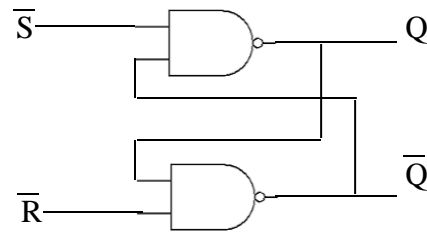
- The latch is a type of temporary storage device that has two stable states (bistable) and is normally placed in a category separate from that of flip – flops.
- Latches are similar to flip – flops because they are bistable devices that can reside in either of two states using a feedback arrangement in which the outputs are connected back to the opposite inputs.
- The main difference between latches and flip – flops is in the method used for changing their state.

**The S – R (SET – RESET) Latch:**

- A latch is a type of bistable logic device or multivibrator.
- An active – HIGH input S – R latch is formed with two cross – coupled NOR gates as shown in the following figure (a).
- An active – LOW input  $\bar{S}$  –  $\bar{R}$  latch is formed with two cross – coupled NAND gates as shown in the following figure (b).
- Note that the output of each gate is connected to an input of the opposite gate. This produces the regenerative feedback that is characteristic of all latches and flip – flops.



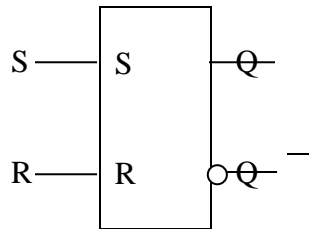
(a) Active – HIGH input S – R latch



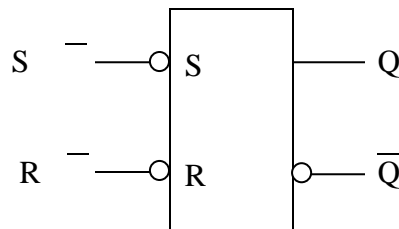
(b) Active – LOW input  $\bar{S}$  –  $\bar{R}$  latch

- A latch can reside in either of its two states, SET or RESET.
- SET means that the Q output is HIGH.
- RESET means that the Q output is LOW.

**Logic Symbols:**



(a) Active – HIGH input S – R latch



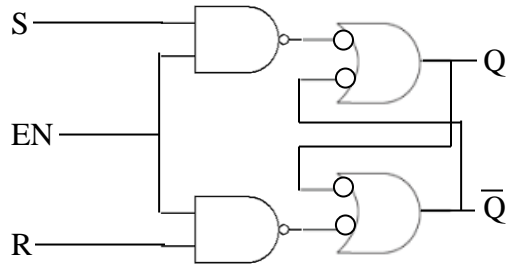
(b) Active – LOW  $\bar{S}$  –  $\bar{R}$  latch

**Truth table for an active – LOW input S – R latch:**

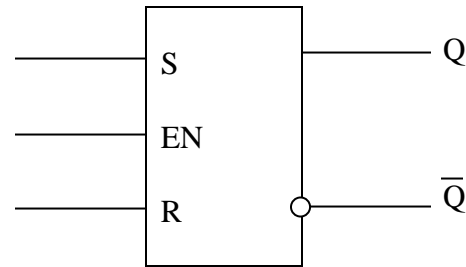
Inputs		Outputs		Comments
S	R	Q	$\bar{Q}$	
1	1	NC	NC	No change. Latch remains in present state
0	1	1	0	Latch SET
1	0	0	1	Latch RESET
0	0	1	1	Invalid Condition

### The Gated S – R Latch:

- A gated latch requires an enable input, EN (G is also used to designate an enable input).
- The logic diagram and logic symbol for a gated S – R latch are shown in the following figure.
- The S and R inputs control the state to which the latch will go when a HIGH level is applied to the EN input.
- The latch will not change until EN is HIGH, but as long as it remains HIGH, the output is controlled by the state of the S and R inputs.
- In this circuit, the invalid state occurs when both S and R are simultaneously HIGH and EN is also HIGH.



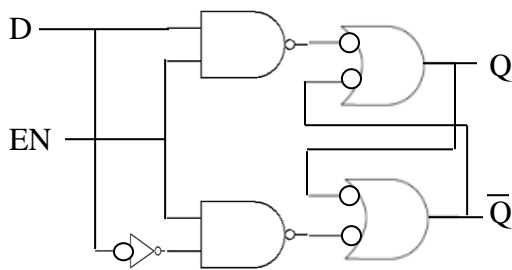
(a) Logic Diagram



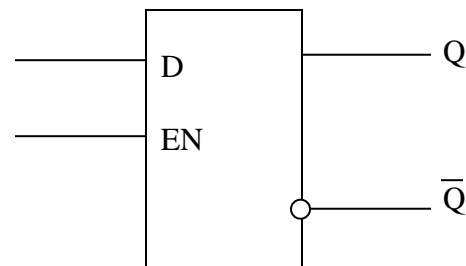
(b) Logic Symbol

### The Gated D Latch:

- Another type of gated latch is called the D latch.
- It differs from the S –R latch because it has only one input in addition to EN. This input is called the D (data) input. The following figure contains a logic diagram and logic symbol of a D latch.
- When the D input is HIGH and EN input is HIGH, the latch will SET.
- When the D input is LOW and EN is HIGH, the latch will RESET.
- Stated another way, the output Q follows the input D when EN is HIGH.



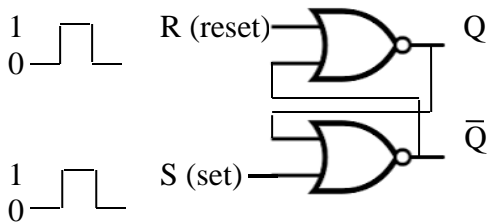
(a) Logic Diagram



(b) Logic Symbol

### Edge – Triggered Flip – Flops:

- Flip – Flops are synchronous bistable devices, also known as bistable multivibrators.
- In this, the term synchronous means that output changes state only at a specified point on the triggering input called the clock (CLK), which is designated as a control input, C, that is, changes in the output occur in synchronization with the clock.
- A Flip – Flop circuit can be constructed from two NAND gates or two NOR gates.

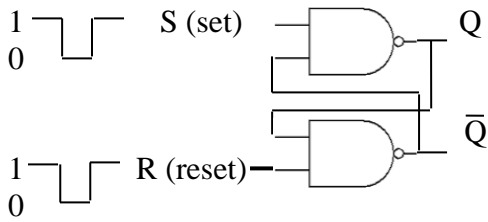


(a) Logic Diagram

S	R	Q	$\bar{Q}$
1	0	1	0
0	0	1	0 (after S=1, R=0)
0	1	0	1
0	0	0	1 (after S=0, R=1)
1	1	0	0

(b) Truth Table

- Basic Flip – Flop circuit with NOR gates



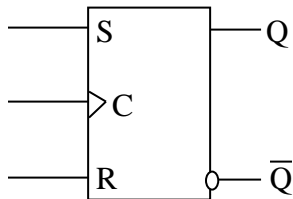
(c) Logic Diagram

S	R	Q	$\bar{Q}$
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

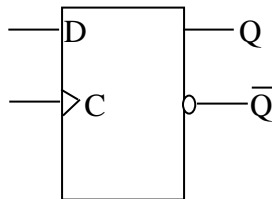
(d) Truth Table

- Basic Flip – Flop circuit with NAND gates

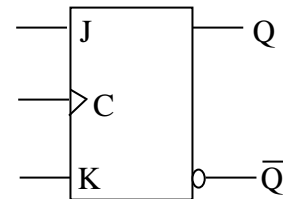
- An edge – triggered flip – flop changes state either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse and is sensitive to its inputs only at the transition of the clock.
- **Three types of edge – triggered flip – flops are:**
  1. S – R flip – flop (S – R is the basis for the D and J – K flip – flops)
  2. D flip – flop
  3. J – K flip – flop
- **Logic symbols for all of these flip – flops:**



(a) S – R Flip – Flop



(b) D Flip – Flop



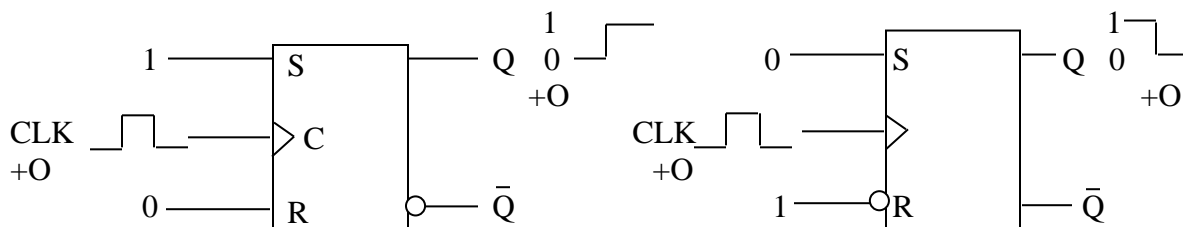
(c) J – K Flip – Flop

cslearner.com

- **Note** that each type can either positive edge – triggered (No bubble at C input) or negative edge – triggered (bubble at C input). The key to identifying an edge – triggered flip – flop by its logic symbol is the small triangle inside the clock at clock (C) input. This triangle is called the dynamic input indicator.
- The dynamic input indicator  $\Delta$  means the flip – flop changes state only the edge of a clock pulse.

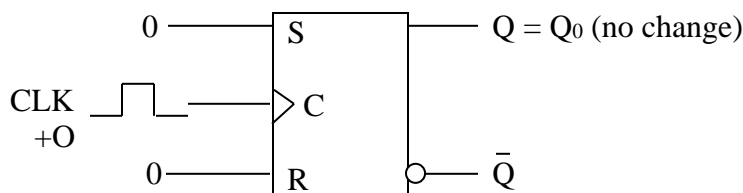
### 1. The Edge – Triggered S – R Flip – Flop:

- An S – R flip – flop should never have both S and R inputs HIGH at the same time when clocked.
- The S and R inputs of the S – R flip – flop are called synchronous inputs because data on these inputs are transferred to the flip – flops output only on the triggering edge of the clock pulse.
- When S is HIGH and R is LOW, the Q output goes HIGH on the triggering edge of the clock pulse, and the flip – flop is SET.
- When S is LOW and R is HIGH, the Q output goes LOW on the triggering edge of the clock pulse, and the flip – flop is RESET.
- When both S and R are LOW, the output does not change from its prior state.
- An invalid condition exists if S and R are both HIGH when an active clock edge occurs.
- This basic operation of a positive edge – triggered flip – flop is shown in the following figure and the Truth Table of flip – flop.
- Remember, the flip – flop cannot change state except on the triggering edge of a clock pulse. The S and R inputs can be changed at any time when the clock input is LOW or HIGH (except for a very short interval around the triggering transition of the clock) without affecting the output.



(a) S=1, R=0 flip – flop SETs on positive clock edge. (If already SET, it remains SET)

(b) S=0, R=1 flip – flop RESETs on positive clock edge. (If already RESET, it remains RESET)



(b) S=0, R=0 flip – flop does not change. (If SET, it remains SET, if RESET, it remains RESET)

- **Truth Table for a positive edge – triggered S – R flip – flops:**

S	R	CLK	Q	$\bar{Q}$	Comments
0	0	X	$Q_0$	$\bar{Q}_0$	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	?	?	Invalid

↑ = Clock transition LOW or HIGH

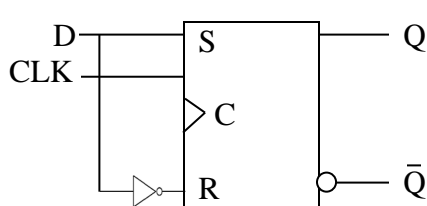
X = Irrelevant (“don’t care”)

$Q_0$  = Output level prior to clock transition

- The operation and truth table for a negative edge – triggered S – R flip – flop are the same as those for a positive edge – triggered device except that the falling edge of the clock pulse is triggering edge.

## 2. The Edge – Triggered D Flip – Flop:

- The D flip – flop is useful when a single data bit (1 or 0) is to be stored.
- The addition of an inverter to an S – R flip – flop creates a basic D flip – flop as in the following figure, where a positive edge – triggered type.
- Note that the flip – flop in the following figure has only one input, the D input in addition to the clock.
- If there is a HIGH on D input when a clock pulse is applied, the flip – flop will set, and the HIGH on the D input is stored by the flip – flop on the positive – going edge of the clock pulse.
- If there is a LOW on the D input when the clock pulse is applied, the flip – flop will reset, and the LOW on the D input is stored by the flip – flop on the leading edge of the clock pulse.
- In the SET state the flip – flop is storing a 1, and in the RESET state it is storing a 0.
- The logical operation of the positive edge – triggered D flip – flop is summarized in the following table.
- The operation of a negative edge – triggered device is the same except that triggering occurs on the falling edge of the clock pulse.
- Remember, Q follows D at the active or triggering clock edge.



D	CLK	Q	$\bar{Q}$	Comments
1	↑	1	0	SET (stores a 1)
0	↑	0	1	RESET (stores a 0)

↑ = Clock transition, LOW or HIGH

- A positive edge – triggered D flip – flop formed with an S – R flip – flop and an Inverter.

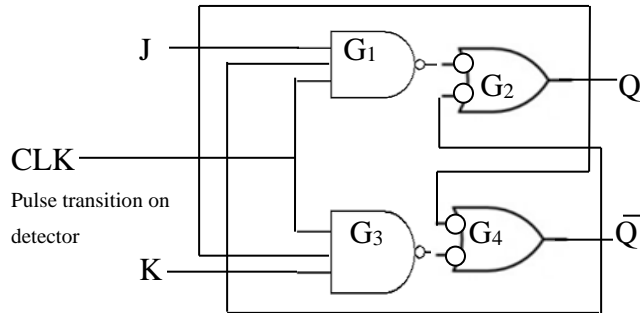
## 3. The Edge – Triggered J – K Flip – Flop:

- The J – K flip – flop is versatile and is a widely used type of flip – flop.
- The functioning of the J – K flip – flop is identical to that of the S – R flip – flop in the SET, RESET, and no – change condition of operation. The difference is that the J – K flip – flop has no invalid state as does the S – R flip – flop.
- The following figure shows the basic internal logic for a positive edge – triggered flip – flop. It differs from S – R edge – triggered J – K flip – flop in



that the Q output is connected back to the input of gate G<sub>2</sub> and the Q output is connected back to the input of gate G<sub>1</sub>.

- The two control inputs are labeled J and K in honor of Jack Kilby, who invented the integrated circuit.
- A J – K flip – flop can also be of the negative edged - triggered type, in which case the clock input is inverted.



J	K	CLK	Q	Q̄	Comments
0	0	↑	Q <sub>0</sub>	Q̄ <sub>0</sub>	No change
0	1	↑	0	1	RESET
1	0	↑	1	0	SET
1	1	↑	Q̄ <sub>0</sub>	Q <sub>0</sub>	Toggle

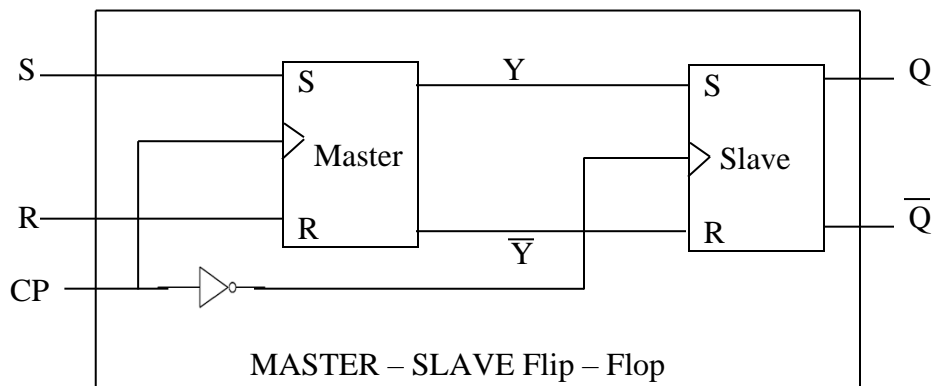
↑ = Clock transition, LOW or HIGH  
Q<sub>0</sub> = Output prior to clock transition

- Logic diagram for a positive edge – triggered J – K flip – flop.

cslearnerr.com

### Master – Slave Flip – Flop:

- A Master – Slave flip – flop is constructed from two separate flip – flops. One circuit serves as a master and the other as a slave, and the overall circuit is referred to as a master – slave flip – flop.
- The logic diagram of an R S master – slave flip – flop is shown in the following figure. It consists of a master flip – flop, a slave flip – flop, and an inverter.
- When clock pulse CP is 0, the output of the inverter is 1. Since the clock input of the slave is 1, the flip – flop is enabled and output Q is equal to Y, while Q̄ is equal to Ȳ.
- The master flip – flop is disabled because CP = 0.
- When the pulse becomes 1, the information then at the external R and S inputs is transmitted the master flip – flop. The slave flip – flop, however, is isolated as long as the pulse is at its 1 level, because the output of the inverter is 0. When the pulse returns to 0, the master flip – flop is isolated, which prevents the external inputs from affecting it. The slave flip – flop then goes to the same state as the master flip – flop.



- Logic diagram of Master – Slave flip – flop.



## Week No. 13: COUNTERS

### Counter:

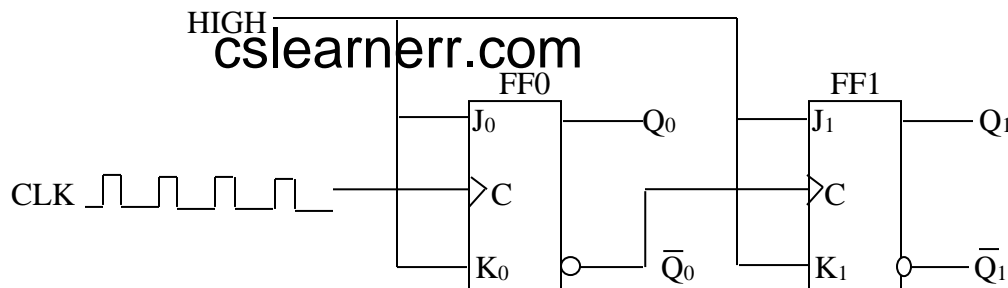
- Flip – Flops can be connected together to perform counting operations, such a group of flip – flops is a counter.
- The number of flip – flops used and the way in which they are connected determine the number of states (called the modulus) and also the specific sequence of states that the counter goes through during each complete cycle.
- Counters are classified into two broad categories according to the way they are clocked: Asynchronous and Synchronous. Within each of these two categories, counters are classified primarily by the type of sequence, the number of states, or the number of flip – flops in the counter.

### Asynchronous Counter:

- The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, don't occur at the same time.
- An asynchronous counter is one in which the flip – flops (FF) within the counter don't change states at exactly the same time because they don't have a common clock pulse.
- In asynchronous counters, the first flip – flop is clocked by the external clock pulse and then each successive flip – flop is clocked by the output of the preceding flip – flop.
- The clock input of an asynchronous counter is always connected only to the LSB flip – flop.
- Asynchronous counters are also known as ripple counters.

#### **2 – Bit Asynchronous Binary Counter:**

- Figure 8-1 shows a 2-bit counter connected for asynchronous operation.
- Notice that the clock (CLK) is applied to the clock input (C) of only the first flip – flop, FF0, which is always the least significant bit (LSB). The second flip – flop, FF1, is triggered by the  $\bar{Q}_0$  output of FF0. FF0 changes state at the positive – going edge of each clock pulse, but FF1 changes only when triggered by a positive - going transition of the  $\bar{Q}_0$  output of FF0. Because of the inherent propagation delay time through a flip – flop, a transition of the input clock pulse (CLK) and a transition of the  $Q_0$  output of FF0 can never occur at exactly the same time. Therefore, the two flip – flops are never simultaneously triggered, so the counter operation is asynchronous.
- In digital logic,  $Q_0$  is always the LSB unless otherwise specified.



- Figure 8-1: A 2-bit asynchronous binary counter.

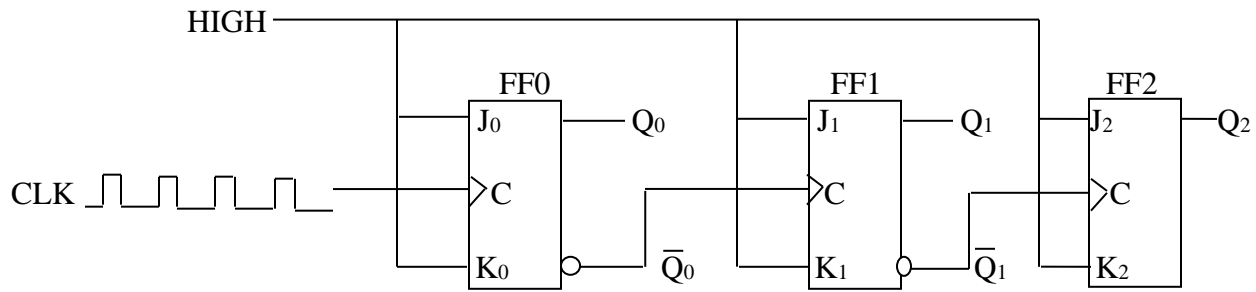
- **State Table for Figure 8-1:**

<b>Clock Pulse</b>	<b>Q<sub>1</sub></b>	<b>Q<sub>0</sub></b>
Initially	0	0
1	0	1
2	1	0
3	1	1
4 (recycles)	0	0

- **Binary state sequence for the counter in Figure 8-1.**

### 3 – Bit Asynchronous Binary Counter:

- The state sequence for a 3-bit binary counter is listed in Table 8-2, and a 3-bit asynchronous binary counter is shown in Figure 8-3. The basic operation is the same as that of the 2-bit counter except that the 3-bit counter has eight states, due to its three flip – flops.



- **Figure 8-3: A 3-bit asynchronous binary counter**

- **State Table 8-2:**

<b>Clock Pulse</b>	<b>Q<sub>2</sub></b>	<b>Q<sub>1</sub></b>	<b>Q<sub>0</sub></b>
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

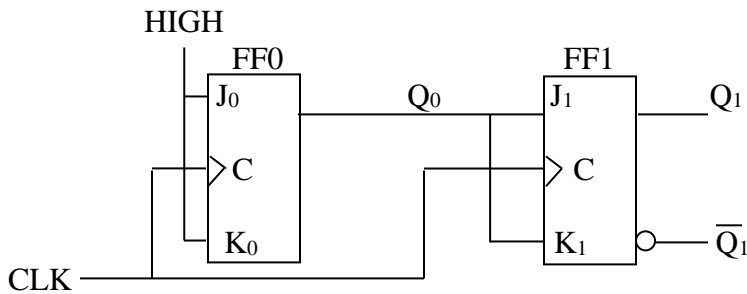
- **State sequence for a 3-bit binary counter**

### Synchronous Counter:

- The term synchronous refers to events that have a fixed time relationship with each other.
- A synchronous counter is one in which all the flip – flops in the counter are clocked at the same time by a common clock pulse.
- In synchronous counters, the clock input is connected to all of the flip – flops so that they are clocked (means the clock input goes to each flip – flop) simultaneously.

## 2 – Bit Synchronous Binary Counter:

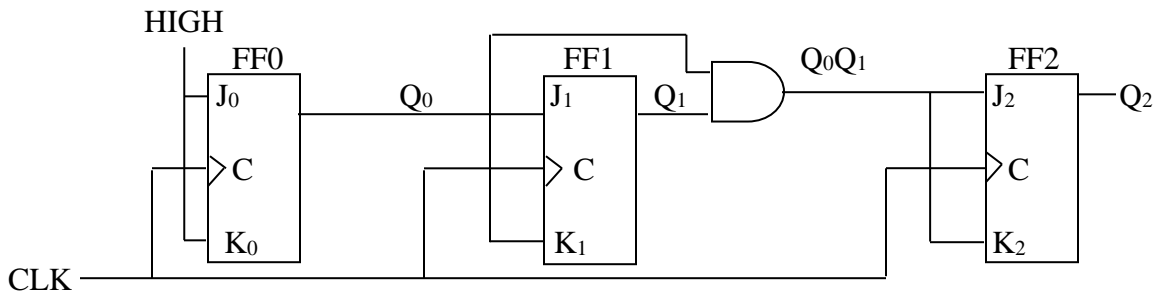
- Figure 8-11 shows a 2-bit synchronous binary counter. Notice that an arrangement different from that for the asynchronous counter must be used for the  $J_1$  and  $K_1$  inputs of FF1.



- Figure 8-11: A 2-bit synchronous binary counter

## 3 – Bit Synchronous Binary Counter:

- A 3-bit synchronous binary counter is shown in Figure 8-14. We can understand this counter operation by examining its sequence of states as shown in Table 8 – 3.



- Figure 8-14: A 3-bit synchronous binary counter

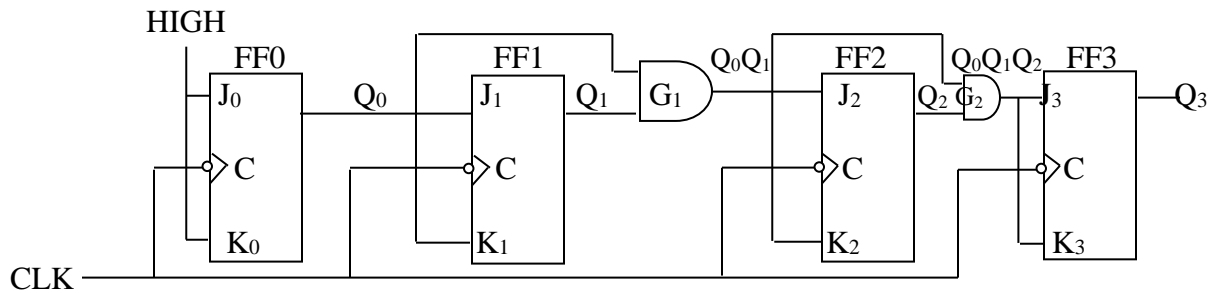
- State Table 8-3:

Clock Pulse	$Q_2$	$Q_1$	$Q_0$
Initially	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8 (recycles)	0	0	0

- State sequence for a 3 – bit binary counter

#### 4 – Bit Synchronous Binary Counter:

- Figure 8 – 16 shows a 4 – bit synchronous binary counter. This particular counter is implemented with negative edge – triggered flip – flops.



- Figure 8-16: A 4-bit synchronous binary counter

Discover comprehensive notes for BSCS,  
BSIT and BSAI courses conveniently  
gathered on our user-friendly website,  
[www.cslearnerr.com](http://www.cslearnerr.com)

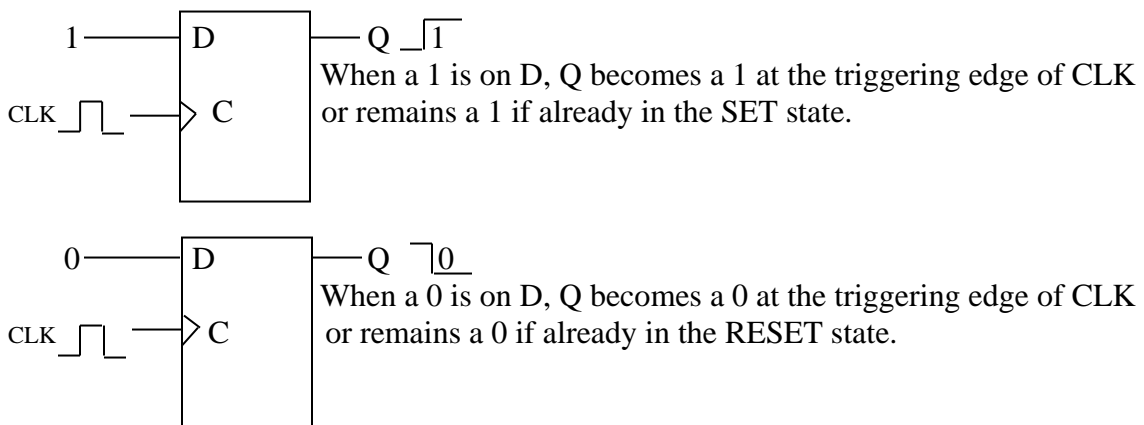
## Week No. 14: BASIC SHIFT REGISTER FUNCTIONS

### Register:

- A register is a digital circuit with two basic functions: Data Storage and Data Movement.
- A register can consist of one or more Flip – Flops used to store and shift data.

### Shift Registers:

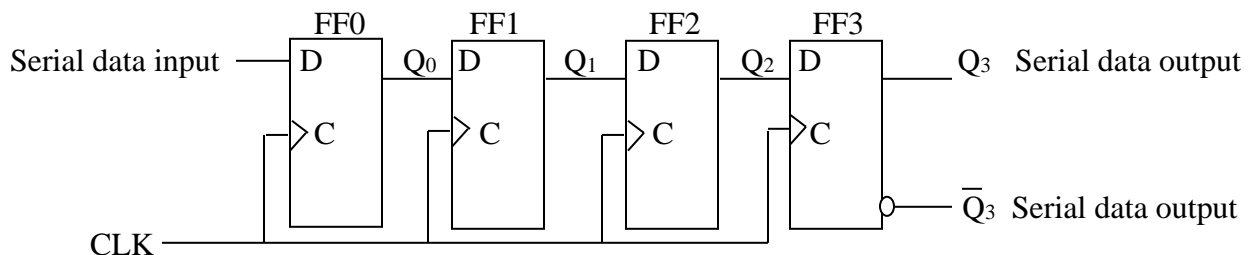
- A register capable of shifting its binary information either to the right or to the left, is called a shift register.
- The logical configuration of a shift register consists of a chain of flip – flops connected, with the output of one flip – flop connected to the input of the next flip – flop.
- All flip – flops receive a common clock pulse which causes the shift from one stage to the next.
- The following figure shows the concept of storing a 1 or 0 in a D flip – flop. A 1 is applied to the data input and a clock pulse is applied that stores the 1 by setting the flip – flop. When the 1 on the input is removed, the flip – flop remains in the SET state, thereby storing the 1. A similar procedure applies to the storage of a 0 by resetting the flip – flop.



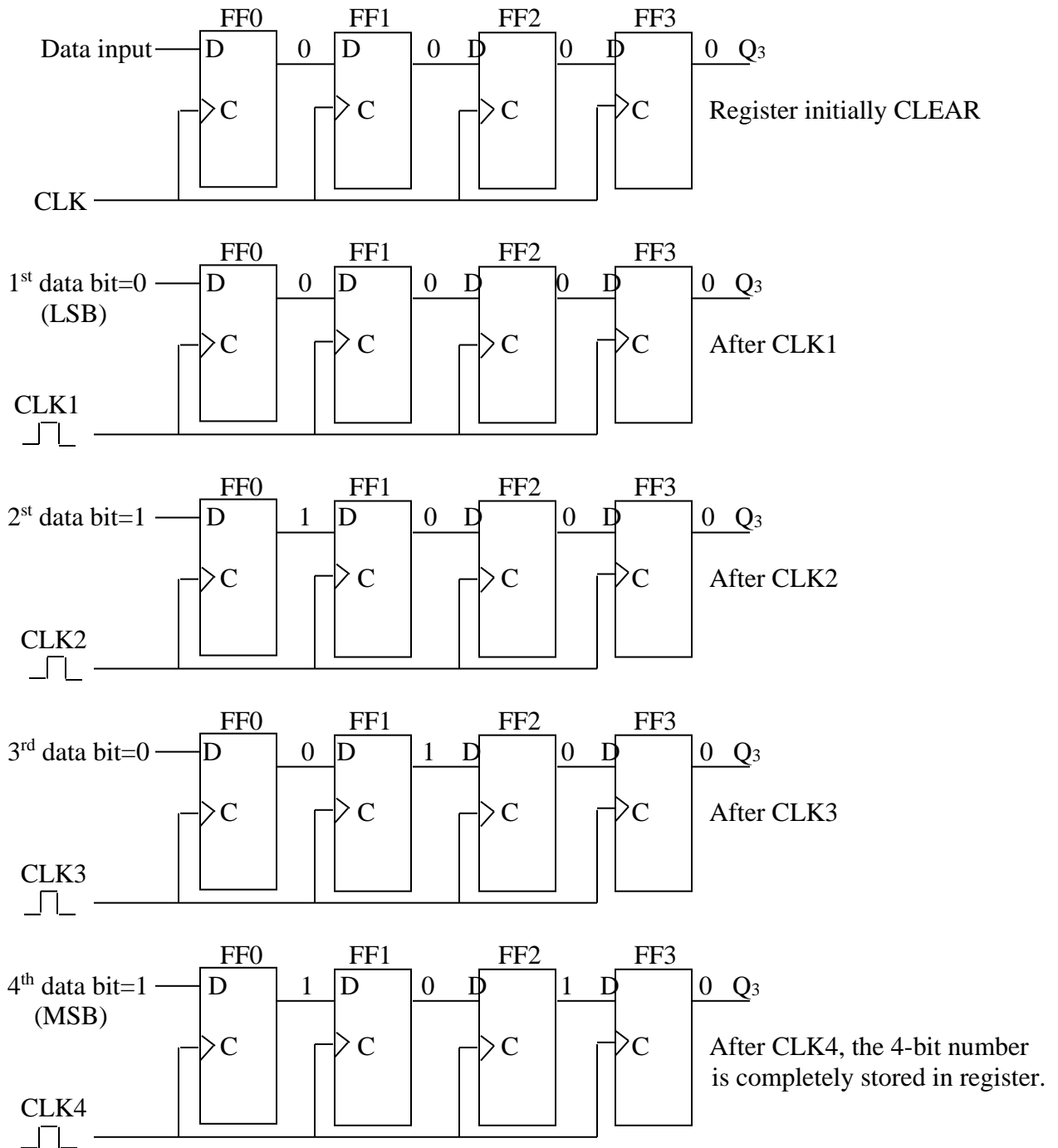
- The storage capacity of a register is the total number of bits (1s & 0s) of digital data.
- Each stage (flip – flop) in a shift register represents one bit of storage capacity; therefore, the number of stages in a register determines its capacity.
- The shift capability of a register permits the movement of data from stage to stage within application of clock pulses.

### Serial In/Serial Out Shift Registers:

- The serial in/serial out shift register accepts data serially – that is, one bit at a time on a single line means one bit at a time is transferred.
- It produces the stored information on its output also in serial form.
- Let's first look at the serial entry of data into a typical shift register.
- The following figure shows a 4 – bit device implemented with D flip – flops with four stages, this register can store up to four bits of data.

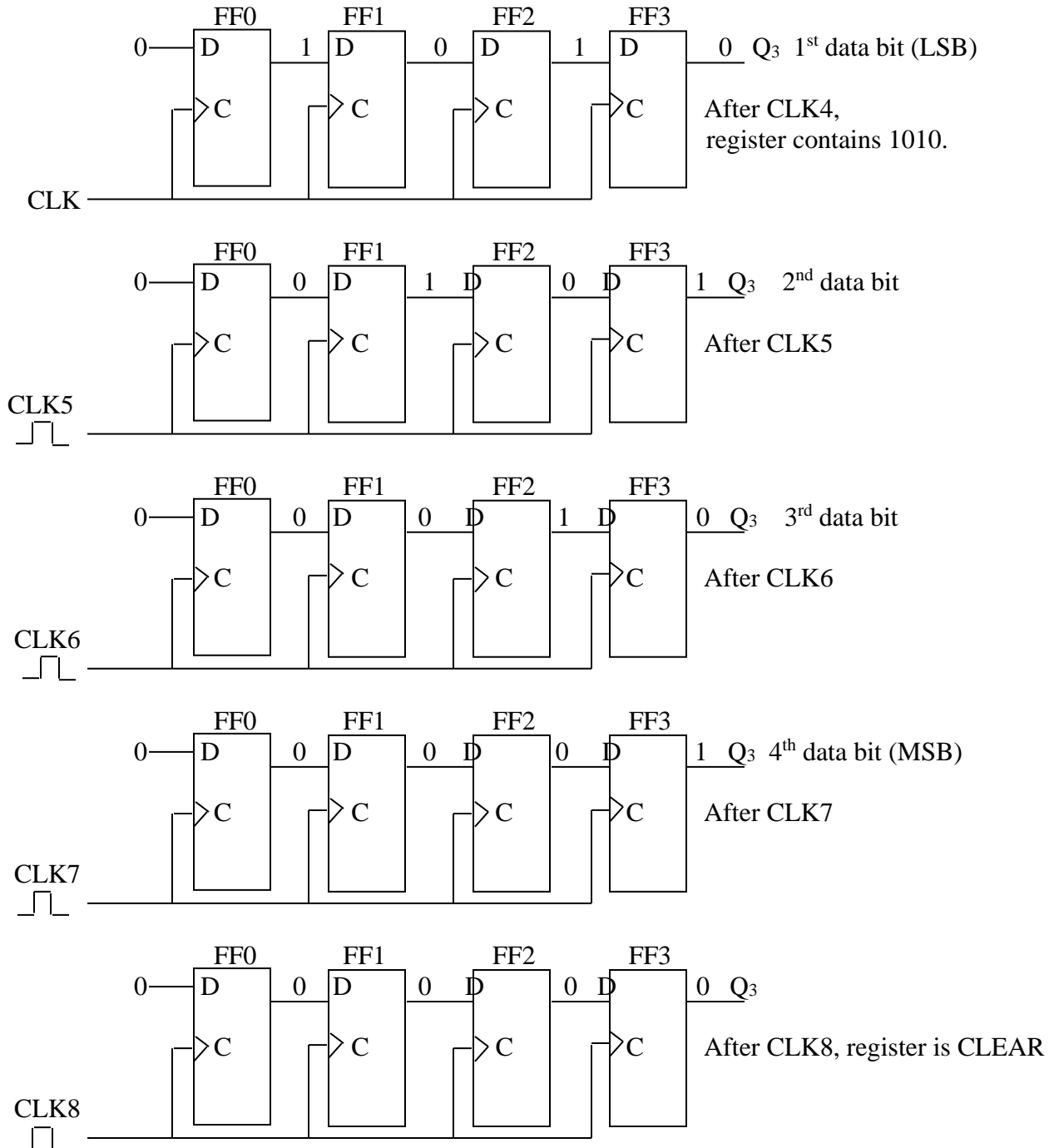


- The following figure illustrates entry of the four bits 1010 into the register, beginning with the least significant bit. The register is initially clear.
- The 0 is put onto the data input line, making  $D = 0$  for FF0. When the first clock pulse is applied, FF0 is reset, thus storing the 0.
- Next the second bit, which is a 1, is applied to the data input, making  $D = 1$  for FF0 and  $D = 0$  for FF1 because the D input of FF1 is connected to the Q0 output. When the second clock pulse occurs, the 1 on the data input is shifted into FF0, causing FF0 to set and the 0 that was in FF0 is shifted into FF1.



- The third bit, a 0 is now put onto the data input line, and a clock pulse is applied. The 0 is entered into FF0, the 1 stored in FF0 is shifted into FF1, and the 0 stored in FF1 is shifted into FF2.

- The last bit, a 1 is now applied to the data input, and a clock pulse is applied. This time the 1 is entered into FF0, the 0 stored in FF0 is shifted into FF1, the 0 stored in FF1 is shifted into FF2, and the 0 stored in FF2 is shifted into FF3. This completes the serial entry of the four bits into the shift register.
- If you want to get the data out of the register, the bits must be shifted out serially and taken off Q3 output as showing in the following figure:
- Four bits (1010) being serially shifted out of the register and replaced by all zeros.



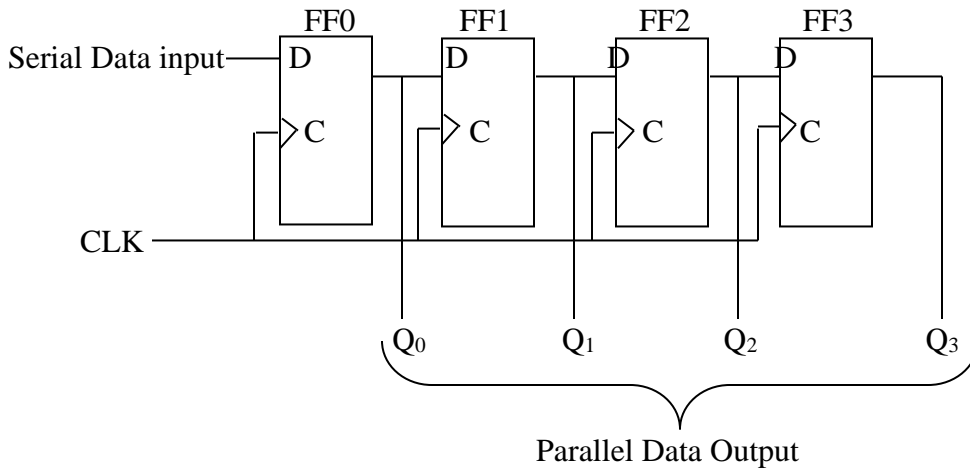
- After CLK4 in the data entry operation, the LSB, 0, appears on Q3 output. When clock pulse CLK5 is applied, the second bit appears on the Q3 output. Clock pulse CLK6 shifts the third bit to the output, and CLK7 shifts the fourth bit to the output. While the original four bits are being shifted out, more bits can be shifted in. All zeros are shown being shifted in.



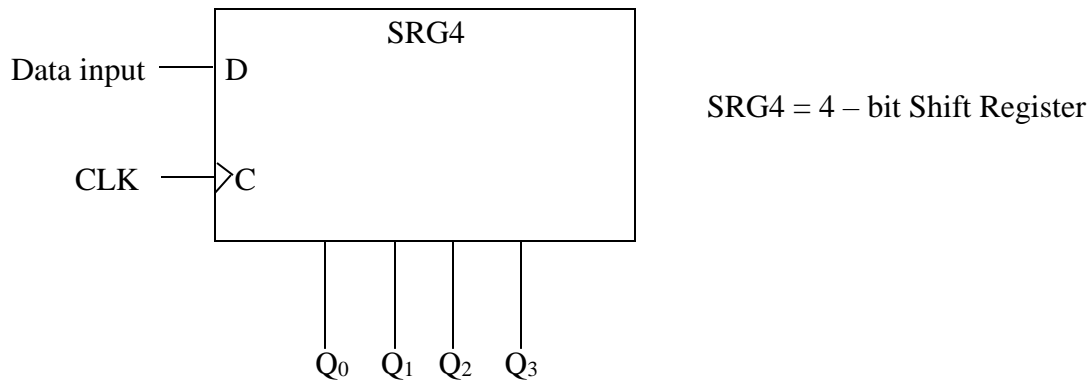
**Serial In/Parallel Out Shift Registers:**

- Data bits are entered serially (Least significant bit first) into a serial in/parallel out shift register.
- In the parallel output register, the output of each stage is available.
- Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously, rather than on a bit – by – bit basis as with the serial output.
- The following figure shows a 4 – bit serial in/parallel out shift register and its logic block symbol.

cslearnerr.com



**(a) 4 – Bit Serial In/Parallel Out Shift Register.**



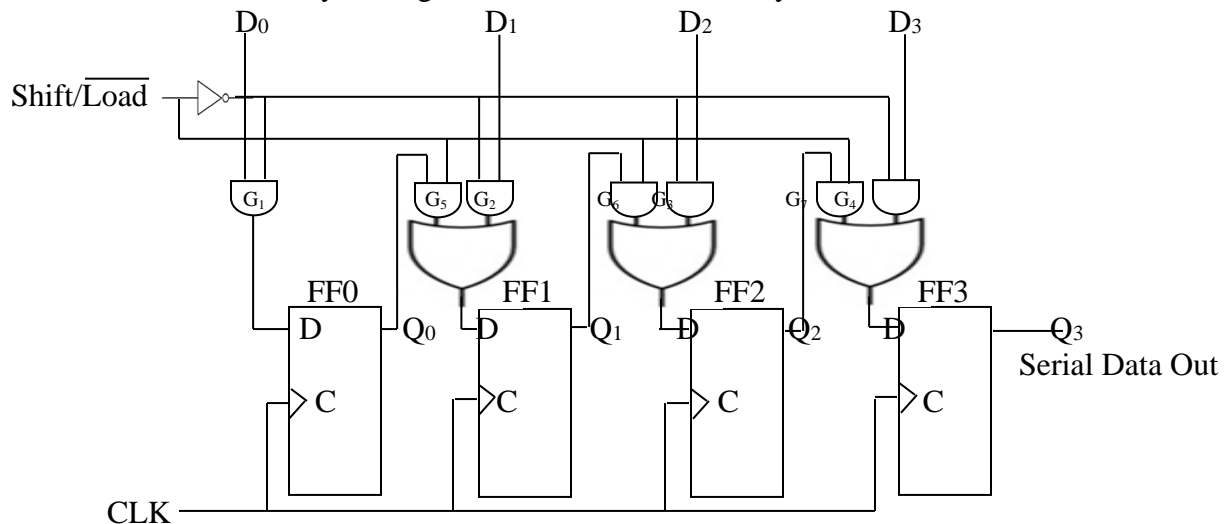
**(b) 4 – Bit Serial In/Parallel Out Shift Register Logic Block Symbol.**



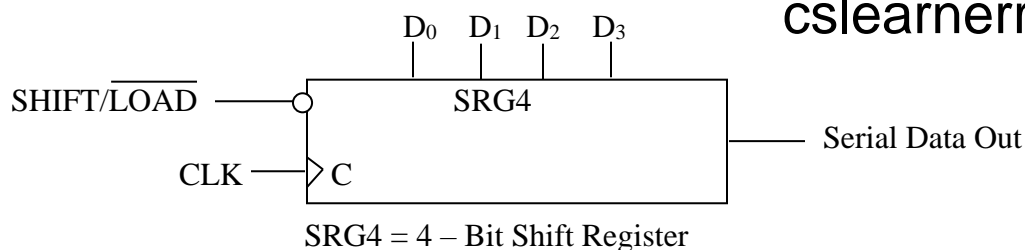
## Week No. 15: PARALLEL IN/SERIAL OUT SHIFT REGISTERS

### Parallel In/Serial Out Shift Registers

- For register with parallel data inputs, the bits are entered simultaneously into their respective states on parallel lines rather than on a bit – by – bit basis on one line as with serial data inputs. Once the data are completely stored in the register then that stored data appears on its output in serial form.
- For parallel data, multiple bits are transferred at one time.
- The following figure shows a 4 – bit parallel in/serial shift register and a typical logic symbol.
- There are four data – input lines D0, D1, D2, and D3, and a  $\overline{\text{SHIFT/LOAD}}$  input, which allows four bits of data to load in parallel into the register.
- When  $\overline{\text{SHIFT/LOAD}}$  is LOW, gates G1 through G4 are enabled, allowing each data bit to be applied to the D input of its respective flip – flop.
- When a clock pulse is applied, the flip – flops with D = 1 will set and those with D = 0 will reset, thereby storing all four bits simultaneously.



(a) Logic diagram

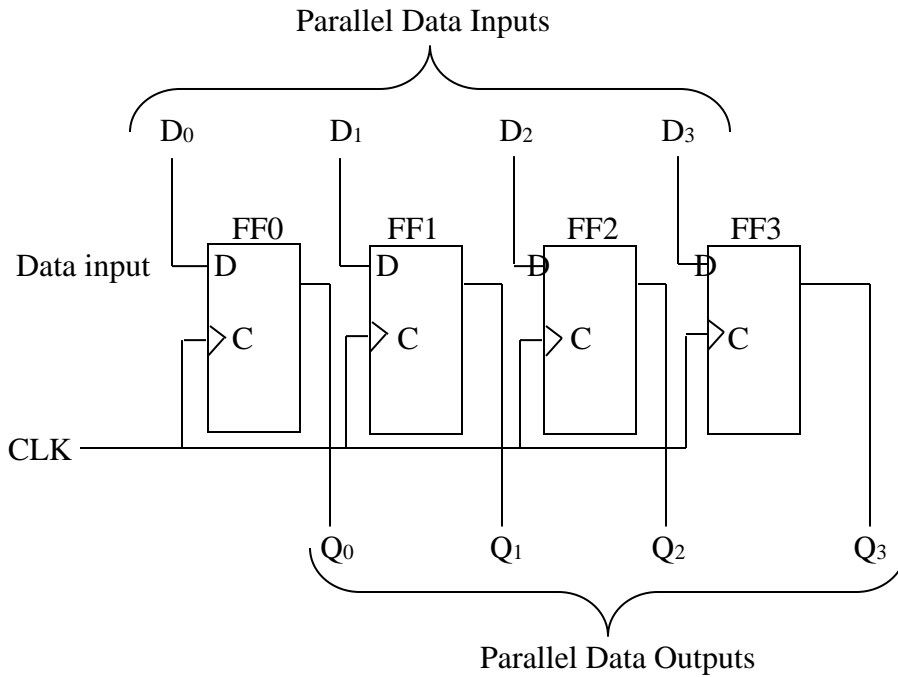


(b) Logic Symbol

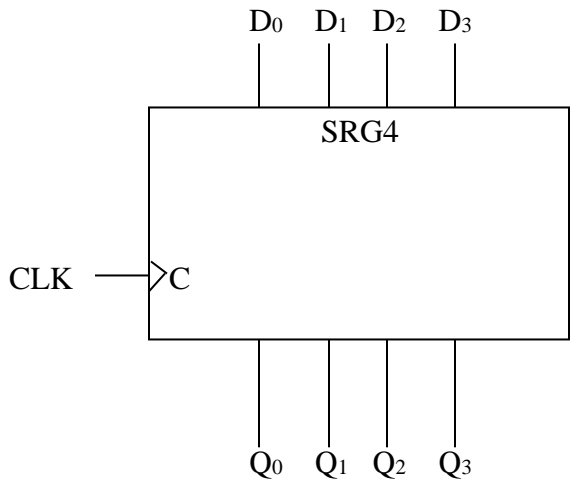
- **$\overline{\text{SHIFT/LOAD}}$ :** When  $\overline{\text{SHIFT/LOAD}}$  is HIGH, the data are shifted right one bit per clock pulse. When  $\overline{\text{SHIFT/LOAD}}$  is LOW, the data on the parallel inputs are loaded into the register.
- When  $\overline{\text{SHIFT/LOAD}}$  is HIGH, gates G1 through G4 are disabled and gates G5 through G7 are enabled, allowing the data bits to shift right from one stage to the next.
- The OR gates allow either the normal shifting operation or the parallel data entry operation, depending on which AND gates are enabled by the level on the  $\overline{\text{SHIFT/LOAD}}$  input.
- Note that FF0 has a single AND to disable the parallel input, D0. It does not require an AND/OR arrangement because there is no serial data in.

**Parallel In/Parallel Out Shift Registers:**

- For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines. Once the data are stored, each bit appears on its respective output lines and all bits are available simultaneously.
- The following figure shows a parallel In/Parallel Out Register:



**(a) Logic diagram**



**(b) Logic Symbol**



## Week No. 16: MICROPROCESSOR AND MEMORY/STORAGE DEVICES

### Memory:

- Memory is the portion of a computer or other system that stores binary data. In a computer, memory is accessed millions of times per second, so the requirement for speed and accuracy is paramount. Very fast semiconductor memory is available today in modules with over 4 GB of capacity.
- **The two major categories of semiconductor memories are the RAM and the ROM:**
  - **RAM** (random-access memory) is a type of memory in which all addresses are accessible in an equal amount of time and can be selected in any order for a read or write operation. All RAMs have both read and write capability. Because RAMs lose stored data when the power is turned off, they are volatile memories.
  - **ROM** (read-only memory) is a type of memory in which data are stored permanently or semi permanently. Data can be read from a ROM, but there is no write operation as in the RAM. The ROM, like the RAM, is a random-access memory but the term RAM traditionally means a random-access read/write memory. Because ROMs retain stored data even if power is turned off, they are nonvolatile memories.

### The Random – Access Memory (RAM):

- A RAM is a read/write memory in which data can be written into or read from any selected address in any sequence.
- When a data unit is written into a given address in the RAM, the data unit previously stored at that address is replaced by the new data unit.
- When a data unit is read from a given address in the RAM, the data unit remains stored and is not erased by the read operation. This nondestructive read operation can be viewed as copying the content of an address while leaving the content intact.
- A RAM is typically used for short-term data storage because it cannot retain stored data when power is turned off.
- **The two major categories of RAM are the static RAM (SRAM) and the dynamic RAM (DRAM):**
  - **SRAMs** generally use latches as storage elements and can therefore store data indefinitely as long as DC power is applied.
  - **DRAMs** use capacitors as storage elements and cannot retain data very long without the capacitors being recharged by a process called refreshing.
  - Both SRAMs and DRAMs will lose stored data when DC power is removed and, therefore, are classified as volatile memories.
  - Data can be read much faster from SRAMs than from DRAMs. However, DRAMs can store much more data than SRAMs for a given physical size and cost because the DRAM cell is much simpler, and more cells can be crammed into a given chip area than in the SRAM.
  - The basic types of SRAM are the asynchronous SRAM and the synchronous SRAM with a burst feature.
  - The basic types of DRAM are the Fast Page Mode DRAM (FPM DRAM), the Extended Data Out DRAM (EDO DRAM), the Burst EDO DRAM (BEDO DRAM), and the synchronous DRAM (SDRAM).

- **Cache Memory:** One of the major applications of SRAMs is in cache memories in computers. Cache memory is a relatively small, high-speed memory that stores the most recently used instructions or data from the larger but slower main memory. Cache memory can also use dynamic RAM (DRAM). Typically, SRAM is several times faster than DRAM. Overall, a cache memory gets stored information to the microprocessor much faster than if only high-capacity DRAM is used. Cache memory is basically a cost-effective method of improving system performance without having to resort to the expense of making all of the memory faster. The concept of cache memory is based on the idea that computer programs tend to get instructions or data from one area of main memory before moving to another area.

### **The Read-Only Memory (ROM):**

- A ROM contains permanently or semi permanently stored data, which can be read from the memory but either cannot be changed at all or cannot be changed without specialized equipment.
- A ROM stores data that are used repeatedly in system applications, such as tables, conversions, or programmed instructions for system initialization and operation.
- ROMs retain stored data when the power is off and are therefore nonvolatile memories.
- **The major categories of semiconductor ROMs:** [cslearnerr.com](http://cslearnerr.com)
  - The Mask ROM is the type in which the data are permanently stored in the memory during the manufacturing process.
  - The PROM, or programmable ROM, is the type in which the data are electrically stored by the user with the aid of specialized equipment. Both the mask ROM and the PROM can be of either MOS or bipolar technology.
  - The EPROM, or erasable PROM, is strictly a MOS device.
  - The UV EPROM is electrically programmable by the user but the stored data must be erased by exposure to ultraviolet light over a period of several minutes.
  - The electrically erasable PROM (EEPROM or E<sup>2</sup>PROM) can be erased in a few milliseconds.

### **The Flash Memory:**

- The ideal memory has high storage capacity, non-volatility, in-system read and write capability, comparatively fast operation, and cost effectiveness. The traditional memory technologies such as ROM, PROM, EPROM, EEPROM, SRAM, and DRAM individually exhibit one or more of these characteristics. Flash memory has all of the desired characteristics.
- Flash memories are high-density read/write memories (high density translates into large bit storage capacity) that are non-volatile, which means that data can be stored indefinitely without power. They are sometimes used in place of floppy or small-capacity hard disk drives in portable computers.

## Magnetic and Optical Storage:

- These storage media are important, particularly in computer applications, where they are used for mass nonvolatile storage of data and programs.
- **Magnetic Storage:**
  - **Magnetic Hard Disks:** Computers use hard disks as the internal mass storage media.
  - Hard disks are rigid “platters” made of aluminum alloy or a mixture of glass and ceramic covered with a magnetic coating.
  - Hard disk drives mainly come in two diameter sizes, 5.25 inch and 3.5 inch although 2.5 inch and 1.75 inch are also available.
  - A hard disk drive is hermetically sealed to keep the disks dust-free.
  - **Floppy Disks:** the floppy disk is an older technology and derives its name because it is made of a flexible polyester material with a magnetic coating on both sides.
  - The early floppy disks were 5.25 inches in diameter and were packaged in a semi flexible jacket.
  - Floppy disks or diskettes are 3.5 inches in diameter and are encased in a rigid plastic jacket.
- **Optical Storage:**
  - **CD – ROM:** The basic Compact Disk-Read-Only Memory is a 120 mm diameter disk with a sandwich of three coatings: a polycarbonate plastic on the bottom, a thin aluminum sheet for reflectivity, and a top coating of lacquer for protection.
  - The CD-ROM disk is formatted in a single spiral track with sequential 2 kb sectors and has a capacity of 680 MB. Data are prerecorded at the factory in the form of minute indentations called pits and the flat area surrounding the pits called lands. The pits are stamped into the plastic layer and cannot be erased.
  - **WORM:** Write Once/Read Many (WORM) is a type of optical storage that can be written onto one time after which the data cannot be erased but can be read many times.
  - To write data, a low-power laser is used to burn microscopic pits on the disk surface.
  - 1s and 0s are represented by the burned and non-burned areas.
  - **CD-R:** This is essentially a type of WORM. The difference is that the CD-Recordable allows multiple write sessions to different areas of the disk.
  - The CD-R disk has a spiral track like the CD-ROM, but instead of mechanically pressing indentations on the disk to represent data, the CD-R uses a laser to burn microscopic spots into an organic dye surface.
  - **CD-RW:** The CD-Rewritable disk can be used to read and write data. Instead of the dye based recording layer in the CD-R, the CD-RW commonly uses a crystalline compound with a special property.
  - **DVD-ROM:** Originally DVD was an abbreviation for Digital Video Disk but eventually came to represent Digital Versatile Disk, Like the CD-ROM, DVD-ROM data are process stored on the disk. However, the pit size is smaller than for the CD-ROM, allowing more data to be stored on a track.
  - The major difference between CD-ROM and DVD-ROM are that the CD is single-sided, while the DVD has data on both sides.

## Introduction to Microprocessor and Microcomputer:

### **Microprocessor:**

- The microprocessor is a digital integrated circuit that can be programmed with a series of instructions to perform various operations on data.
- A microprocessor is the CPU of a computer. It can do arithmetic and logic operations, move data from one place to another, and make decisions based on certain instructions.
- A microprocessor consists of several units, each designed for a specific job. The specific units, their design and organization, are called the architecture.
- The architecture determines the instruction set and the process for executing those instructions.
- Four basic units that are common to all microprocessors are the arithmetic logic unit (ALU), the instructions decoder, the register array, and the control unit.

### **Microcomputer:**

- A typical microcomputer system consists of a microprocessor plus memory, and I/O interface.
- The various components that form the system are linked through buses that transfer instructions, data, addresses, and control information among the IC components.
- The following Figure shows the block diagram of a microcomputer system:
- Typically, the microcomputer has a single microprocessor. If many processors are included, then we have a multiprocessor system – which is a valid possibility.
- A number of RAM and ROM chips are combined to form a given memory size.
- The interface units communicate with various external devices through the I/O bus.
- At any given time, the microprocessor selects one of the units through its address bus.
- Data are transferred to and from the selected unit and the microprocessor via the data bus.
- Control information is usually transferred through individual lines, each specifying a particular control function.

