

# Digital Design & Computer Organization [BCS302].

Prof. Yogesh. N <sup>①</sup>  
Assistant Professor  
Dept. of CSE  
ATMECE, Mysuru

## Module 1 :- Introduction to Digital Design.

### Binary Logic :-

- \* Binary logic deals with variables that take on two discrete values and with operations that assume logical meaning.
- \* The two values the variables take may be called by different names [eg:- true/false, yes/no, etc], but for our purpose it is convenient to think in terms of bits & assign the values 1 and 0.
- \* Binary logic is used to describe the manipulation and processing of binary information. It is particularly suited for the analysis & design of digital systems.
- \* The digital logic circuits that perform the binary arithmetic are circuits whose behavior is most conveniently expressed by means of binary variables and logical operations.

### Definition of Binary Logic :-

- \* Binary logic consists of binary variables and logical operations. The variables are designated by letters of the alphabet such as A, B, C, x, y, z, etc., with each variable having two and only two distinct possible values: 1 and 0.
- \* There are three basic logical operations
  - ① AND
  - ② OR
  - ③ NOT

### ① AND :-

The operation is represented by a dot or by the absence of an operator. Eg:-  $x \cdot y = z$  or  $xy = z$  is read "x AND y is equal to z".

The logical operation AND is interpreted to mean that  $z=1$  if and only if  $x=1$  and  $y=1$ ; otherwise  $z=0$ .

### ② OR :-

This operation is represented by a plus sign.

Eg:-  $x + y = z$  is read "x OR y is equal to z", meaning that  $z=1$  if  $x=1$  or if  $y=1$  or if both  $x=1$  and  $y=1$ .

If both  $x=0$  and  $y=0$ , then  $z=0$ .

### ③ NOT :-

This operation is represented by a prime (sometimes by bar).

Eg:-  $x' = z$  (or  $\bar{x} = z$ ) is read "x not is equal to z", meaning that  $z$  is what  $x$  is not. In other words, if  $x=1$ , then  $z=0$  but if  $x=0$ , then  $z=1$ .

\* For each combination of the values of  $x$  and  $y$ , there is a value of  $z$  specified by the definition of the logical operation. These definitions may be listed in a compact form using truth tables.

\* A truth table is a table of all possible combinations of the variables showing the relation between the values that the variables may take and the result of the operation.

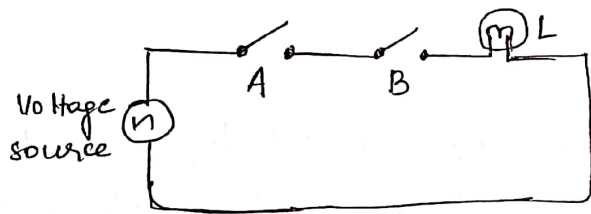
AND		
x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR		
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

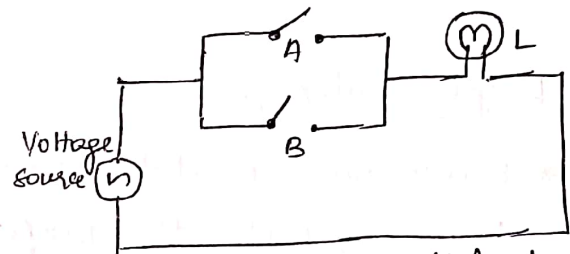
NOT	
x	$\bar{x}$
0	1
1	0

### Switching Circuits and Binary Signals :-

- \* The use of binary variables and the application of binary logic are demonstrated by the simple switching circuits of fig. below.



(a) Switches in series - logic AND



(b) Switches in parallel - logic OR

- \* Let the manual switches A and B represent two binary variables with values equal to 0 when the switch is open and 1 when the switch is closed.
- \* Let the lamp L represents a third binary variable equal to 1 when the light is ON and 0 when OFF.
- \* For the switches in series, the light turns ON if A and B are closed. Similarly for the switches in parallel, the light turns ON if A or B is closed.
- \* The two circuits can be expressed by means of binary logic with the AND and OR operations respectively.

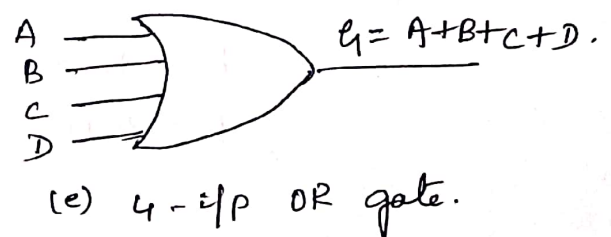
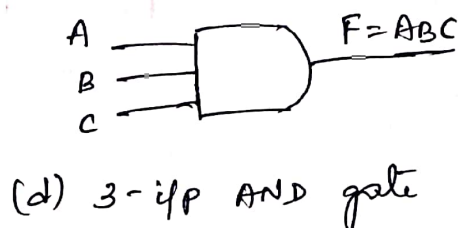
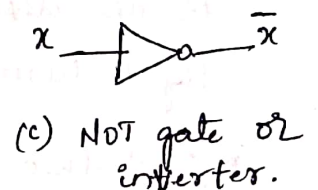
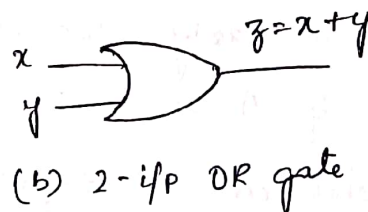
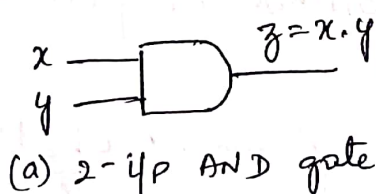
$$L = A \cdot B$$

$$L = A + B$$

- \* Electronic digital circuits are sometimes called switching circuits because they behave like a switch, with the active element such as a transistor either conducting (switch closed) or not conducting (switch open).
- \* Instead of changing the switch manually, an electronic switching circuit uses binary signals to control the conduction or non-conduction state of the active element.
- \* Electrical signals such as voltages or currents exist throughout a digital system in either one of two recognizable binary values i.e., logic 1 or logic 0.

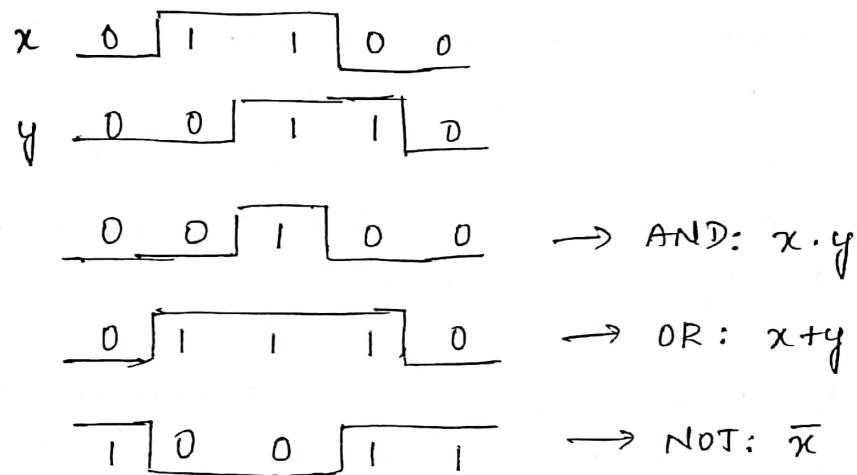
### Logic Gates :-

- \* Electronic digital circuits are also called logic circuits because, with the proper input, they establish logical manipulation paths.
- \* Any desired information for computing or control can be operated upon by passing binary signals through various combinations of logic circuits, each signal representing a variable and carrying one bit of information.
- \* Logic circuits that perform the logical operations of AND, OR and NOT are shown with their symbols in fig. below.





- \* These circuits, called gates, are blocks of hardware that produce a logic-1 or logic-0 output signal if input logic requirements are satisfied.
- \* The input signals  $x$  and  $y$  in the two-input gates of the figure of symbols shown above may exist in one of four possible states: 00, 10, 11 or 01.
- \* These input signals are shown in fig. below, together with the output signals for the AND, OR & NOT gates.



- \* AND and OR gates may have more than two inputs. An AND gate with three inputs and an OR gate with four inputs are shown in figure.
- \* The mathematical system of binary logic is better known as Boolean or switching algebra. This algebra is conveniently used to describe the operation of complex networks of digital circuits.
- \* Designers of digital systems use Boolean algebra to transform circuit diagrams to algebraic expressions & vice versa.

## Boolean Algebra:-

- \* Boolean algebra may be defined ~~as~~ with a set of elements, a set of operators, & a number of unproved axioms or postulates.
- \* The postulates of a mathematical system form the basic assumptions from which it is possible to deduce the rules, theorems and properties of the system.

## Basic Theorems & Properties of Boolean Algebra:-

### a) Duality :-

- \* The Huntington postulates have been listed in pairs and designated by part (a) and part (b).
- \* One part may be obtained from the other if the binary operators and the identity elements are interchanged. This important property of Boolean algebra is called the duality principle.
- \* It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators & identity elements are interchanged.
- \* Consider the following Boolean laws

$$A + 0 = A \text{ and } A \cdot 1 = A$$

The first of these laws implies Boolean addition or logical OR operation, & the second implies Boolean multiplication or logical AND operation.

These laws can be derived directly from each other on the basis of the principle of identity. The rules are

- (i) Replace all (+) symbols by (.) symbols
- (ii) Replace all (.) symbols by (+) symbols
- (iii) Replace every 0 by 1, and every 1 by 0.

\* consider the Boolean law  $A + 0 = A$  ( $A$  is a variable)

Following the rule stated earlier, we get the following relationship

$A \cdot 1 = A$  which is the dual of  $A + 0 = A$ .

( $+$  is replaced by  $\cdot$  and  $0$  is replaced by  $1$ ).

Eg:-  $A + A = A$  in relation to logical OR operation

Its dual is  $A \cdot A = A$  in relation to logical AND operation.

|||<sup>14</sup> we have

$$A + \bar{A} = 1 \quad \text{and} \quad A \cdot \bar{A} = 0$$

$$A + 1 = 1 \quad \text{and} \quad A \cdot 0 = 0.$$

Postulate 2

(a)  $A + 0 = A$

(b)  $A \cdot 1 = A$

Postulate 5

(a)  $A + \bar{A} = 1$

(b)  $A \cdot \bar{A} = 0$

Theorem 1

(a)  $A + A = A$

(b)  $A \cdot A = A$

Theorem 2

(a)  $A + 1 = 1$

(b)  $A \cdot 0 = 0$

Theorem 3

(a)  $\overline{(\bar{A})} = A$

[Involution]

Postulate 3

(a)  $A + B = B + A$

(b)  $AB = BA$

[commutative]

Theorem 4

(a)  $A + (B + C) = (A + B) + C$

(b) ~~A~~  $A(BC) = (AB)C$

[Associative]

Postulate 4

(a)  $A(B + C) = AB + AC$

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

[Distributive]

Theorem 5

(a)  $\overline{(A + B)} = \bar{A} \cdot \bar{B}$

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

[DeMorgan]

Theorem 6

(a)  $A + AB = A$

(b)  $A(A + B) = A$

[Absorption]

Table 1 :- Postulates & Theorems of Boolean Algebra.

### Basic Theorems :-

- \* Table 1 lists 6 theorems of Boolean algebra and 4 of 15 postulates.
- \* The theorems and postulates listed are the most basic relationships in Boolean algebra.
- \* The theorems, like the postulates are listed in pairs; each relation is the dual of the one paired with it.
- \* The postulates are basic axioms of the algebraic structure & need no proof. The theorems must be proven from the postulates.

### Proof :-

#### Theorem 1

(a)  $A + A = A$

$$A + A = (A + A) \cdot 1 \rightarrow \text{by postulate 2(b)}$$

$$= (A + A)(A + \bar{A}) \quad 5(a)$$

$$= A + A\bar{A} \quad 4(b)$$

$$= A + 0 \quad 5(b)$$

$$\boxed{A + A = A} \quad 2(a)$$

$$AA + A\bar{A} + AA + A\bar{A}$$

$$A + A\bar{A} + A + A\bar{A}$$

$$A + A\bar{A}$$

(b)  $A \cdot A = A$

$$A \cdot A = AA + 0 \rightarrow \text{by postulate 2(a)}$$

$$= AA + A\bar{A} \quad 5(b)$$

$$= A(A + \bar{A}) \quad 4(a)$$

$$= A \cdot 1 \quad 5(a)$$

$$\boxed{A \cdot A = A} \quad 2(b)$$

Note :- Theorem 1(b) is the dual of theorem 1(a). Any dual theorem can be similarly derived from the proof of its corresponding pair.



Theorem 2 :-

(a)  $A + 1 = 1$

$$A + 1 = 1 \cdot (A + 1) \rightarrow \text{by postulate 2(b)}$$

$$= (A + \bar{A})(A + 1) \quad 5(a)$$

$$= A + \bar{A} \cdot 1 \quad 4(b)$$

$$= A + \bar{A} \quad 2(b)$$

$$5(a)$$

$$\boxed{A + 1 = 1}$$

(b)  $A \cdot 0 = 0$  by duality.

Theorem 3 :-

$$(\bar{\bar{A}}) = A.$$

From postulate 5, we have  $A + \bar{A} = 1$  and  $A \cdot \bar{A} = 0$ , which defines the complement of  $A$ . The complement of  $\bar{A}$  is  $A$  and it is also  $(\bar{\bar{A}})$ . Therefore, since the complement is unique, we have that  $(\bar{\bar{A}}) = A$ .

Theorem 6

(a)  $A + AB = A.$

$$A + AB = A \cdot 1 + AB \quad \text{--- -- } 2(b)$$

$$= A(1 + B) \quad \text{--- -- } 4(a)$$

$$= A(B + 1) \quad \text{--- -- } 3(a)$$

$$= A \cdot 1 \quad \text{--- -- } 2(a)$$

$$\boxed{A + AB = A}$$

$$\text{--- -- } 2(b)$$

(b)  $A(A + B) = A$  by duality.

## Theorem 5 :- DeMorgan's theorem

### ① DeMorgan's first theorem :-

This theorem states that the complement of a sum is equal to the product of the complements.

$$\overline{A+B} = \bar{A} \cdot \bar{B}$$

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

proved.

### ② DeMorgan's second theorem :-

This theorem states that the complement of a product is equal to the sum of the complements.

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

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

proved.

## Theorem 4 :-

$$(a) A + (B + C) = (A + B) + C$$

$$\begin{aligned}
 \text{LHS} &= A + (B + C) \cdot 1 \\
 &= A + (B + C) \cdot (C + \bar{C}) \\
 &= A + BC + CC + B\bar{C} + C\bar{C} \\
 &= A + B(C + \bar{C}) + C + 0 \\
 &= A + B(1) + C \\
 &= A + (B + C) \quad \checkmark
 \end{aligned}$$

$$\begin{aligned}
 \text{RHS} &= (A + B) + C \\
 &= (A + B) + C \cdot (C + \bar{C}) \\
 &= (A + B) + CC + C\bar{C} \\
 &= (A + B) + C + 0 \\
 &= A + (B + C)
 \end{aligned}$$

## Operator Precedence :-

\* The operator precedence for evaluating Boolean expressions

is (1) Parentheses

(2) NOT

(3) AND

(4) OR.

\* The expression inside the parenthesis must be evaluated before all other operations. The next operation that holds precedence is the complement, then follows the AND, & finally the OR.

\* Eg:- consider the expression  $(\overline{A+B}) = \overline{A} \cdot \overline{B}$

① Expression inside the parentheses is evaluated first and the result the complemented. — left side.

② Complement of A and B are both evaluated first and the result is then ANDed. — right side.

Eg:- Using basic Boolean theorem prove;

$$\textcircled{1} (x+y)(x+z) = x + yz$$

$$\text{sol}^n:- (x+y)(x+z)$$

$$\rightarrow x \cdot x + x \cdot z + y \cdot x + y \cdot z$$

$$= x + xz + yx + yz$$

$$\because x \cdot x = x$$

$$= x(1+z) + yx + yz$$

$$= x + yx + yz$$

$$\because (1+z) = 1$$

$$= x(1+y) + yz$$

$$= \underline{\underline{x + yz}}$$

$$\because (1+y) = 1$$

$$\textcircled{2} \quad xy + xz + y\bar{z} = xz + y\bar{z}$$

sol:-  $xy + xz + y\bar{z}$

$$= xy(z + \bar{z}) + xz(y + \bar{y}) + y\bar{z}(x + \bar{x})$$

$$= \underline{xyz} + \underline{x\bar{y}z} + \underline{xy\bar{z}} + x\bar{y}z + \underline{x\bar{y}\bar{z}} + \bar{x}y\bar{z}$$

$$= xyz + x\bar{y}\bar{z} + x\bar{y}z + \bar{x}y\bar{z}$$

$$= xyz + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}y\bar{z}$$

$$= xz(y + \bar{y}) + y\bar{z}(x + \bar{x})$$

$$= \underline{\underline{xz + y\bar{z}}}$$

$$\therefore x + \bar{x} = 1$$

$$y + \bar{y} = 1$$

$$z + \bar{z} = 1$$

$$\therefore xyz + x\bar{y}z = x\bar{y}z$$

$$x\bar{y}\bar{z} + x\bar{y}z = x\bar{y}z$$

→ Rearranging

$$\therefore y + \bar{y} = 1, \quad x + \bar{x} = 1$$



## Boolean Functions :-

- \* A binary variable can take the value of 0 or 1.
- \* A Boolean function is an expression formed with binary variables, the two binary operators OR and AND, the unary operator NOT, parentheses and Equal sign.
- \* For a given value of the variables, the function can be either 0 or 1. Eg:- the Boolean function:

$$F_1 = xyz'$$

The function  $F_1$  is equal to 1 if  $x=1$  and  $y=1$  and  $z'=1$ . Otherwise  $F_1=0$ . The above is an example of a Boolean function represented as an algebraic expression.

- \* A Boolean function may also be represented in a truth table. To represent a function in a truth table, we need a list of the  $2^n$  combinations of 1's and 0's of the  $n$  binary variables, & a column showing the combinations for which the function is equal to 1 or 0.
- \* As shown in table below, there are 8 possible distinct combinations for assigning bits to three variables.

$x$	$y$	$z$	$F_1 = xyz'$	$F_2 = x+y'z$	$F_3 = x'y'z + x'yz + xy'$	$F_4$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

$$F_4 = xy' + x'z.$$

\* The table shows that the function  $F_1$  is equal to 1 only when  $x=1$ ,  $y=1$  and  $z=0$ . It is equal to 0 otherwise.

$F_2=1$  if  $x=1$  or if  $y=0$ , while  $z=1$ .

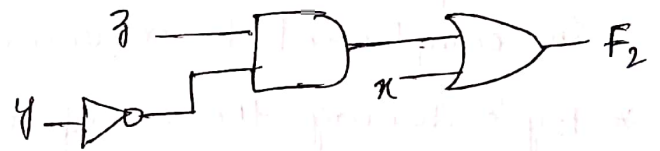
$F_3$  and  $F_4$  has four 1's & four 0's.

\* A Boolean function may be transformed from an algebraic expression into a logic diagram composed of AND, OR and NOT gates.

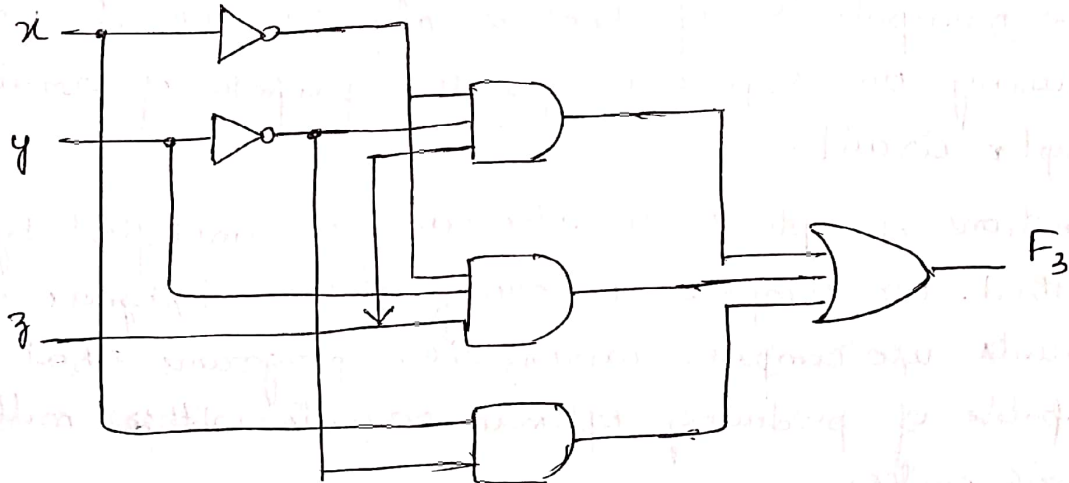
\* The implementation of the four functions introduced in the discussion is shown in fig. below.



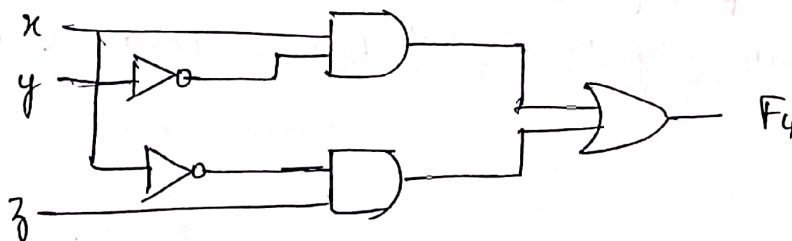
$$(a) F_1 = xyz'$$



$$(b) F_2 = x + y'z$$



$$(c) F_3 = x'y'z + x'yz + xy'$$



$$(d) F_4 = xy' + x'z$$

- \* From the diagram, it is obvious that the implementation of  $F_4$  requires fewer gates and fewer inputs than  $F_3$ .
- \* To find simpler circuits, one must know how to manipulate Boolean functions to obtain equal and simpler expressions.

### Algebraic Manipulation :-

- \* When a Boolean Expression is implemented with logic gates, Each term requires a gate and Each variable within the term designates an input to the gate.
- \* We define a literal to be a single variable within a term, in complemented or uncomplemented form.
- \* By reducing the no of terms, no of literals or both in a Boolean expression, it is often possible to obtain a simpler circuit.
- \* The manipulation of Boolean algebra consists mostly of reducing an expression for the purpose of obtaining a simpler circuit.
- \* Functions of upto 5 variables can be simplified by the map method. For complete Boolean functions, designers of digital circuits use computer minimization programs that are capable of producing optimal circuits with millions of logic gates.
- \* The function of fig (a) [previous topic] has 3 terms and 8 literals. Similarly in fig (d) has 2 terms & 4 literals.

### Complement of a function :-

- \* The complement of a function  $F$  is  $F'$  and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of  $F$ .
- \* The complement of a function may be derived algebraically through DeMorgan's theorems, listed in table shown earlier for two variables.
- \* DeMorgan's theorems can be extended to 3 or more variables. The three-variable form of the first DeMorgan's theorem is derived as follows, from postulates & theorems

$$\begin{aligned}
 (A+B+C)' &= (A+x)' && \text{Let } B+C = x \\
 &= A'x' && \text{by theorem 5(a) (DeMorgan)} \\
 &= A'(B+C)' && \text{substitute } B+C = x \\
 &= A'(B'C') && \text{by theorem 5(a) (DeMorgan)} \\
 &= A'B'C' && \text{by theorem 4(b) (associative)}
 \end{aligned}$$

- \* DeMorgan's theorems for any number of variables resemble the two-variable case in form & can be derived by successive substitutions similar to the method used in the preceding derivation. These theorems can be generalized as follows:

$$\begin{aligned}
 (A+B+C+D+\dots+F)' &= A'B'C'D'\dots F' \\
 (ABCD\dots F)' &= A'+B'+C'+D'+\dots+F'
 \end{aligned}$$

- \* The generalized form of DeMorgan's theorems states that the complement of a function is obtained by interchanging AND and OR operators & complementing each literal.



### Example

- ① Find the complement of the functions  $F_1 = x'y'z' + x'y'z$ .  
and  $F_2 = x(y'z' + yz)$ .

Applying DeMorgan's theorems,

$$\begin{aligned} F_1' &= (x'y'z' + x'y'z)' \\ &= (x'y'z')' \cdot (x'y'z)' \\ &= (x+y'+z)(x+y+z') \end{aligned}$$

$$\begin{aligned} F_2' &= [x(y'z' + yz)]' \\ &= x' + (y'z' + yz)' \\ &= x' + (y'z')' \cdot (yz)' \\ &= x' + (y+z)(y'+z') \\ &= x' + yz' + y'z \end{aligned}$$

- ② Find the complement of the functions by taking their duals & complementing each literal.

a)  $F_1 = x'y'z' + x'y'z$ .

Dual of  $F_1$  is  $(x'+y+z')(x'+y'+z)$ .

complement each literal:  $(x+y+z)(x+y+z') = F_1'$

b)  $F_2 = x(y'z' + yz)$

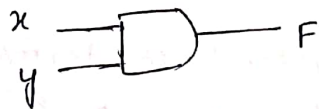
Dual of  $F_2 \rightarrow x'' + (y'+z')(y+z)$

complement  $\rightarrow x' + (y+z)(y'+z') = F_2'$ .

## Digital Logic Gates :-

- \* Since Boolean functions are expressed in terms of AND, OR and NOT operations, it is easier to implement a Boolean function with these types of gates.
- \* Factors to be weighed when considering the construction of other types of logic gates are
  - ① the feasibility and economy of producing the gate with physical components
  - ② the possibility of extending the gate to more than two inputs
  - ③ the basic properties of the binary operator such as commutativity and associativity
  - ④ the ability of the gate to implement Boolean functions alone or in conjunction with other gates.
- \* The eight logical functions are used as standard gates in digital design.
  - complement  $\rightarrow$  NOT
  - transfer  $\rightarrow$  Buffer
  - logical AND  $\rightarrow$  AND
  - logical OR  $\rightarrow$  OR
  - logical NAND  $\rightarrow$  NAND
  - logical NOR  $\rightarrow$  NOR
  - logical Ex-OR  $\rightarrow$  EX-OR.
  - Equivalence  $\rightarrow$  EX-NOR.
- \* The gates except for the inverter, & buffer, can be extended to have more than two inputs.

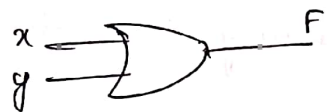
AND



$$F = xy$$

x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

OR



$$F = x + y$$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

NOT  
(Inverter)



$$F = \bar{x}$$

x	F
0	1
1	0

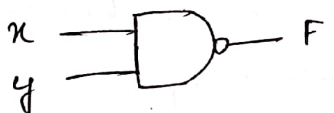
Buffer



$$F = x$$

x	F
0	0
1	1

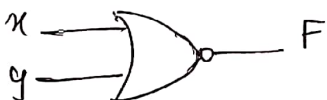
NAND



$$F = \overline{xy}$$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR



$$F = \overline{(x + y)}$$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

X-OR

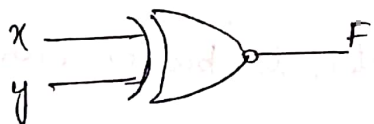


$$F = x\bar{y} + \bar{x}y$$

$$= x \oplus y$$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

X-NOR

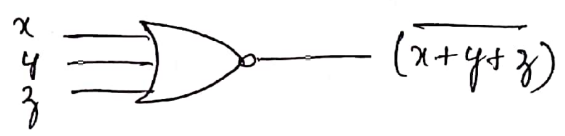


$$F = xy + \bar{x}\bar{y}$$

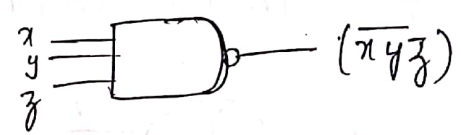
$$= x \odot y$$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

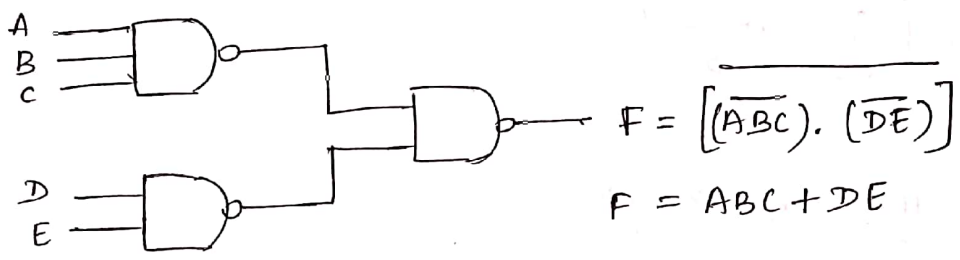
\* The graphic symbols for the 3-input gates are shown in fig. below.



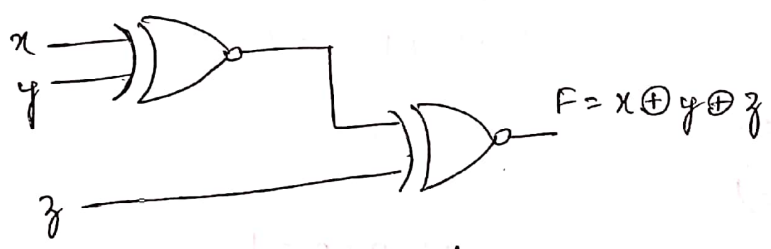
(a) 3-i/p NOR gate



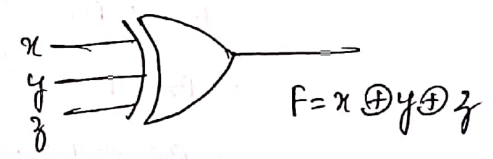
(b) 3-i/p NAND gate



(c) Cascaded NAND gate.



(a) Using 2-i/p gates.



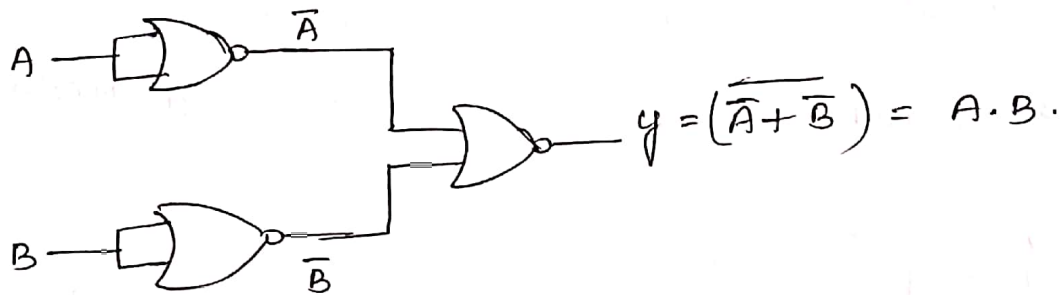
(b) A 3-i/p gate

x	y	z	F
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



Example :- Construct AND gate using NOR gate

Sol<sup>n</sup> :-  $(\overline{A \cdot B}) = (\overline{A + B})$  using De Morgan's theorem



Simplify the expressions :-

$$\begin{aligned}
 \textcircled{1} \quad Y &= ABC + \bar{A}BC + A\bar{B}C \\
 &= BC(A + \bar{A}) + A\bar{B}C \\
 &= BC \cdot 1 + A\bar{B}C \\
 &= BC(1 + A) + A\bar{B}C \\
 &= BC + ABC + A\bar{B}C \\
 &= BC + AC(B + \bar{B}) \\
 &= BC + AC \cdot 1 \\
 &= C(B + A) \\
 \underline{Y} &= \underline{C(A + B)}
 \end{aligned}$$

$$\therefore A + \bar{A} = 1$$

$$\therefore 1 + A = 1$$

$$\therefore B + \bar{B} = 1$$

$$\therefore B + A = A + B$$

$$\begin{aligned}
 \textcircled{2} \quad Y &= (A + B)(\bar{A} + \bar{B}) \\
 &= (A + B)(\bar{A} + \bar{B}) \\
 &= A\bar{A} + A\bar{B} + \bar{A}\bar{B} + B\bar{B} \\
 &= A\bar{B} + \bar{A}\bar{B} + B \\
 &= \bar{B}(\bar{A} + A) + B \\
 &= \bar{B} \cdot 1 + B \\
 &= \bar{B} + B \\
 &= B
 \end{aligned}$$

$$\therefore A\bar{A} = 0 \quad \& \quad B\bar{B} = 0$$

$$\therefore A + \bar{A} = 1$$

$$\therefore B + \bar{B} = 1 \quad (B(B + \bar{B}) = B \cdot 1 = B)$$

③  $Y = \overline{AB + \bar{A}\bar{B}}$  using DeMorgan's theorem.

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

$$= (\overline{AB}) \cdot (\overline{\bar{A}\bar{B}}) \quad \text{DeMorgan's 1st theorem}$$

$$= (\bar{A} + \bar{B})(A + B) \quad \leftarrow \text{2nd theorem}$$

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

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

$$\therefore \bar{A}A = 0 \text{ \& } \bar{B}B = 0$$

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

④ Simplify  $ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$  & realize the Expression using basic gates.

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

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

$$= BC \cdot 1 + A\bar{B}C + AB\bar{C}$$

$$\therefore A + \bar{A} = 1$$

$$= BC(1 + A) + A\bar{B}C + AB\bar{C}$$

$$\therefore 1 + A = 1$$

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

$$= BC + AC(B + \bar{B}) + AB\bar{C}$$

$$= BC + AC \cdot 1 + AB\bar{C}$$

$$\therefore B + \bar{B} = 1$$

$$= BC + AC(1 + B) + AB\bar{C}$$

$$\therefore 1 + B = 1$$

$$= BC + AC + ACB + AB\bar{C}$$

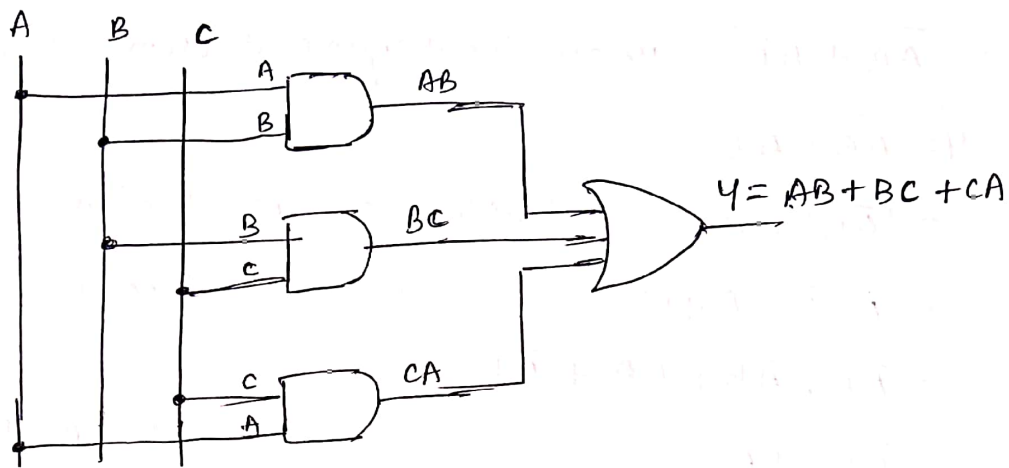
$$= BC + AC + AB(C + \bar{C})$$

$$= BC + AC + AB \cdot 1$$

$$\therefore C + \bar{C} = 1$$

$$= BC + AC + AB$$

$$Y = \underline{\underline{AB + BC + CA}}$$



④ Find the complement of the boolean expression  $AB + AC$  using De Morgan's theorems & realize the same using basic gates only.

$$Y = AB + AC$$

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

$$= (\overline{AB}) \cdot (\overline{AC})$$

De Morgan's 1<sup>st</sup> theorem

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

← 2<sup>nd</sup> theorem.

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

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

$$\because \overline{A}\overline{A} = \overline{A}$$

$$= \overline{A}(1 + \overline{C}) + \overline{B}\overline{A} + \overline{B}\overline{C}$$

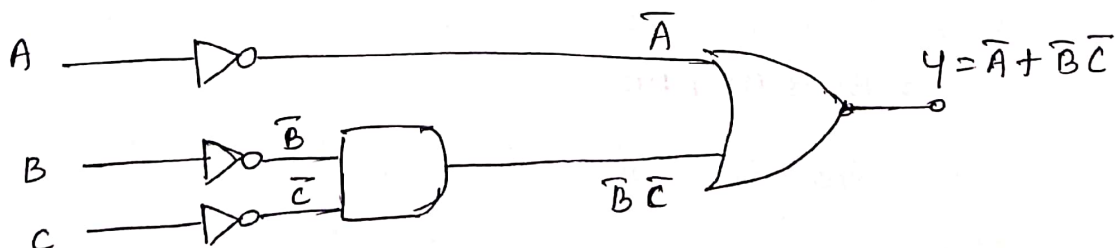
$$= \overline{A} \cdot 1 + \overline{A}\overline{B} + \overline{B}\overline{C}$$

$$\because 1 + \overline{C} = 1$$

$$= \overline{A}(1 + \overline{B}) + \overline{B}\overline{C}$$

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

$$\because 1 + \overline{B} = 1$$



## Gate-Level Minimization :-

- \* Gate level minimization is the design task of finding an optimal gate level implementation of the Boolean functions describing a digital circuit.
- \* This task is well understood, but is difficult to execute by manual methods when the logic has more than a few inputs.
- \* Fortunately, computer based logic synthesis tools can minimize a large set of Boolean equations efficiently & quickly.
- \* Nevertheless, it is important that a designer understand the underlying mathematical description and solution of the problem.

## The Map method :-

- \* The complexity of the digital logic gates that implement a Boolean function is directly related to the complexity of an algebraic expression from which the function is implemented.
- \* Although the truth table representation of a function is unique, when it is expressed algebraically it can appear in many different but equivalent forms.
- \* Boolean expressions may be simplified by algebraic means & this procedure of minimization is awkward because it lacks specific rules to predict each succeeding step in the manipulative process.
- \* The map method presented here provides a simplest, straight forward procedure for minimizing Boolean functions. This method may be regarded as a pictorial form of a truth table. It is also known as Karnaugh map or K-map.

- \* K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized.
- \* The map presents a visual diagram of all possible ways a function may be expressed in standard form.
- \* The simplified expressions produced by the map are always in one of the two standard forms
  - ① Sum of products (SOP)
  - ② Product of sums (POS).

### Two Variable K-map :-

- \* The two-variable map is shown in fig(a). below.

$m_0$	$m_1$
$m_2$	$m_3$

Fig (a)

		$y$	
		0	1
$x$	0	$m_0$ $x'y'$	$m_1$ $x'y$
	1	$m_2$ $xy'$	$m_3$ $xy$

Fig (b)

Fig. Two-variable K-map.

- \* There are 4 minterms for two variables; hence the map consists of 4 squares, one for each minterm.
- \* The map is redrawn in fig. (b) to show the relationship between the squares & the two variables  $x$  &  $y$ .
- \* The 0 and 1 marked in each row and column designate the values of variables.



### Three Variable K-map :-

\* A three-variable K-map is shown in fig below.

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$

Fig (a)

		$yz$			
		00	01	11	10
$x$	0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
$x$	1	$xy'z'$	$xy'z$	$xyz$	$xyz'$

Fig (b).

Fig:- Three-variable K-map.

- \* There are 8 minterms for three binary variables, therefore the map consists of 8 squares.
- \* Note that the minterms are arranged, not in a binary sequence but in a sequence similar to the Gray code. The characteristic of this sequence is that only one bit changes in value from one adjacent column to the next.
- \* The map drawn in Fig(b) is marked with numbers in each row and each column to show the relationship between the squares and the three variables.
- \* To understand the usefulness of the map in simplifying Boolean functions we must recognize the basic property possessed by adjacent squares: Any two adjacent squares in the map differ by only one variable, which is primed in one square & unprimed in the other.

① Simplify the Boolean function

$$F(x, y, z) = \sum(2, 3, 4, 5)$$

		y			
		00	01	11	10
x	0	m <sub>0</sub>	m <sub>1</sub>	m <sub>3</sub> 1	m <sub>2</sub> 1
	1	m <sub>4</sub> 1	m <sub>5</sub> 1	m <sub>7</sub>	m <sub>6</sub>

$$\begin{aligned}
 F(x, y, z) &= \bar{x}yz + \bar{x}y\bar{z} + x\bar{y}\bar{z} + x\bar{y}z \\
 &= \bar{x}y(z + \bar{z}) + x\bar{y}(\bar{z} + z) \\
 &= \bar{x}y + x\bar{y}
 \end{aligned}$$

② Simplify the Boolean function

$$F(x, y, z) = \sum(3, 4, 6, 7)$$

		y			
		00	01	11	10
x	0	m <sub>0</sub>	m <sub>1</sub>	m <sub>3</sub> 1	m <sub>2</sub>
	1	m <sub>4</sub> 1	m <sub>5</sub>	m <sub>7</sub> 1	m <sub>6</sub> 1

$$\begin{aligned}
 F &= \bar{x}yz + x\bar{y}\bar{z} + xyz + xy\bar{z} \\
 &= yz(\bar{x} + x) + x\bar{z}(\bar{y} + y) \\
 &= yz + x\bar{z}
 \end{aligned}$$

## Pairs, Quads and Octets :-

\* By grouping together adjacent 1's in the k-map, it is possible to eliminate some of the variables & obtain directly the simplified forms of Boolean expressions.

### ① 2-variable K-map

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

A variable can be eliminated -

$$m_0 = 0, m_1 = 1, m_2 = 0, m_3 = 1.$$

$$\begin{aligned} \therefore f(A, B) &= \sum(1, 3) \\ &= \bar{A}B + AB \\ &= B(\bar{A} + A) \\ &= \underline{\underline{B}}. \end{aligned}$$

### ② 3-variable K-map

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

$\rightarrow$  c can be eliminated

$$\therefore f = \bar{A}B + A\bar{B}$$

$\rightarrow$  c can be eliminated.

### verification

$$\begin{aligned} f &= \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{B}C \\ &= \bar{A}B(C + \bar{C}) + A\bar{B}(\bar{C} + C) \\ &= \bar{A}B + A\bar{B} \end{aligned}$$

$$f = \underline{\underline{A \oplus B}}$$

Ex2:-

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

$$\begin{aligned} f &= \bar{B}\bar{C} + A\bar{B} \\ &= \bar{B}(A + \bar{C}) \end{aligned}$$

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

$$f = C.$$

$A\overline{A}$  &  $\overline{B}B$  will be eliminated

Verification:-

$$\begin{aligned}
 f &= \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}C + ABC \\
 &= \overline{B}C(\overline{A}+A) + BC(\overline{A}+A) \\
 &= \overline{B}C + BC \\
 &= C(\overline{B}+B) \\
 f &= \underline{\underline{C}}
 \end{aligned}$$

Ex:-

1	1	1	1
0	0	0	0

1	0	0	1
1	0	0	1

### ③ 4-variable K-map

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

$$f = BD.$$

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

$$f = \overline{B}\overline{D}.$$

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

$$f = \underline{\underline{D}}$$

Note:- Pair eliminates One variable  
 Quad eliminates Two variables  
 Octet eliminates Three variables

## NAND and NOR implementation :-

- \* Digital circuits are more frequently constructed with NAND or NOR gates than with AND and OR gates.
- \* NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.
- \* Because of the prominence of NAND and NOR gates in the design of digital circuits, rules and procedures have been developed for the conversion from Boolean functions given in terms of AND, OR & NOT into equivalent NAND or NOR logic diagrams.

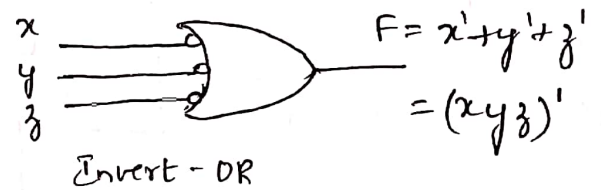
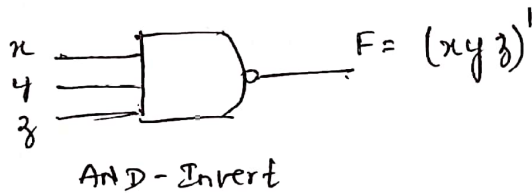


Fig. Two graphic symbols for NAND gate.

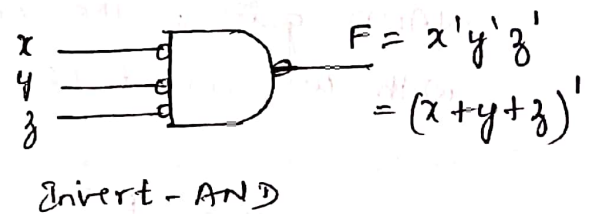
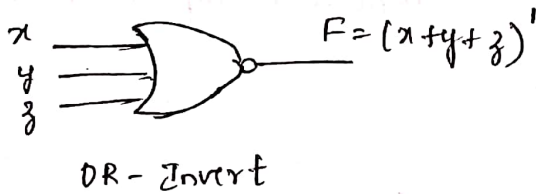


Fig. Two graphic symbols for NOR gate.

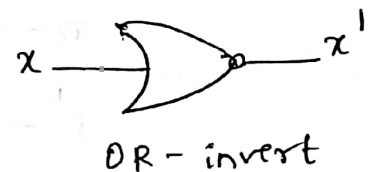
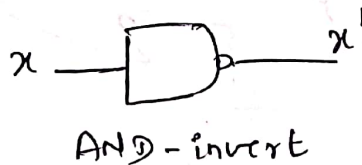
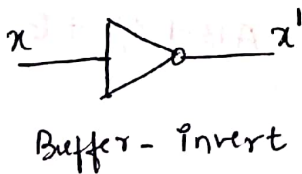


Fig. Three graphic symbols for inverter

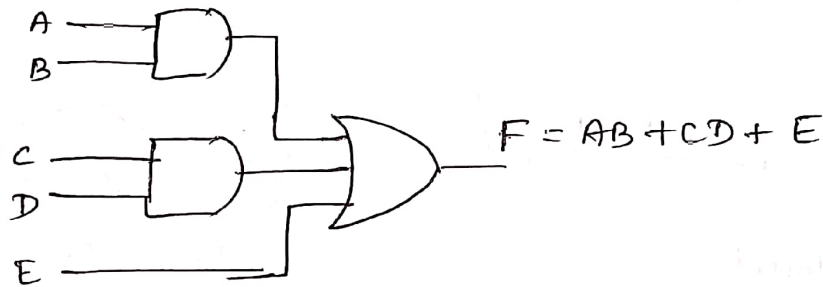


## NAND Implementation :-

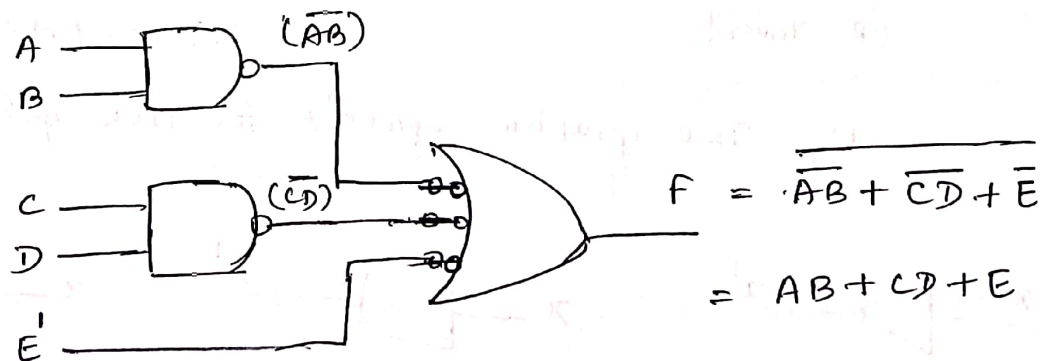
- \* The implementation of a Boolean function with NAND gates requires that the function be simplified in the sum of products form.
- \* Consider the following example to see the relationship between a sum of products expression & its equivalent NAND implementation.

$$F = AB + CD + E$$

The function is implemented in following fig. in sum of products form with AND and OR gates.



In the following fig, the AND gates are replaced by NAND gates & the OR gate is replaced by a NAND gate with an invert OR symbol.



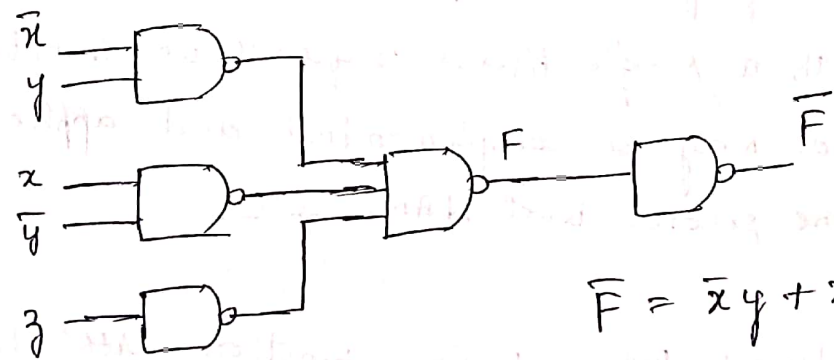
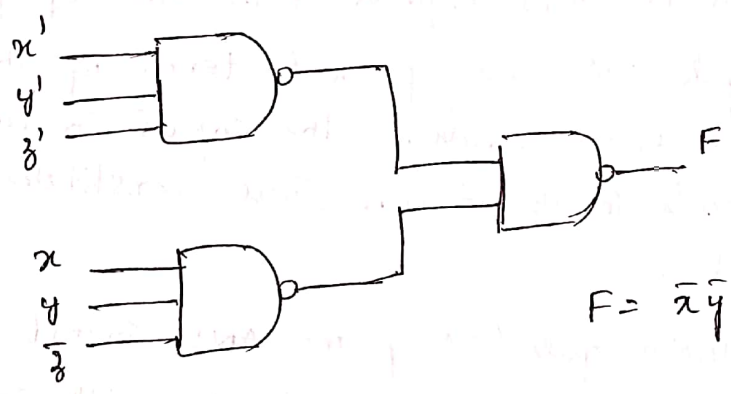
x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	$yz$
$\bar{x}$	1	0	0	0
x	0	0	0	1

$$F = \bar{x}\bar{y}\bar{z} + xy\bar{z}$$

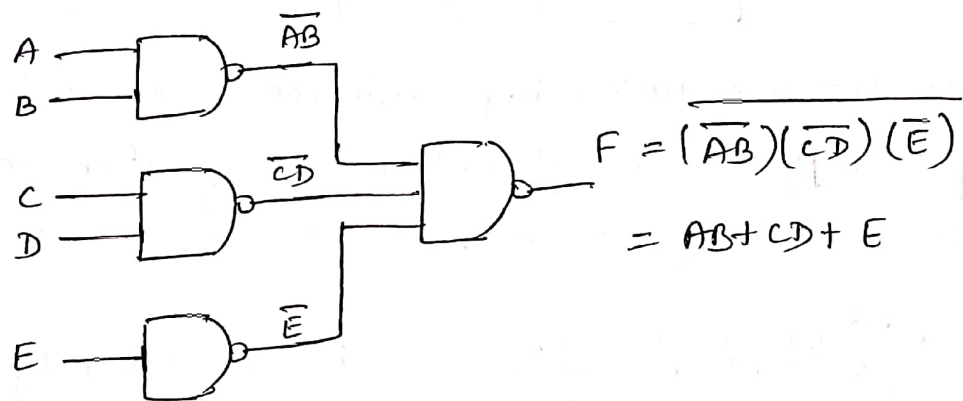
The two level NAND implementation is shown in fig. below, by simplify the complement of the function in sum of products. This is done by combining the 0's in the map:

x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	$y\bar{z}$	$yz$
$\bar{x}$	1	0	0	0
x	0	0	0	1

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



\* In the following fig, the output NAND gate is redrawn with the conventional symbol.



\* The rule for obtaining the NAND logic diagram from a Boolean function is as follows:

- ① Simplify the function and express it in sum of products
- ② Draw a NAND gate for each product term of the function that has at least two literals. The inputs in each NAND gate are the literals in the term. This constitutes a group of first level gates.
- ③ Draw a single NAND gate (using the AND-invert or invert-OR graphic symbol) in the second level, with inputs coming from outputs of first level gates.
- ④ A term with a single literal requires an inverter in the first level or may be complemented and applied as an input to the second-level NAND gate.

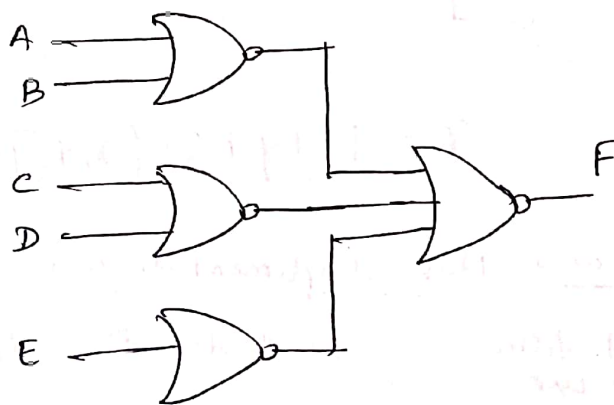
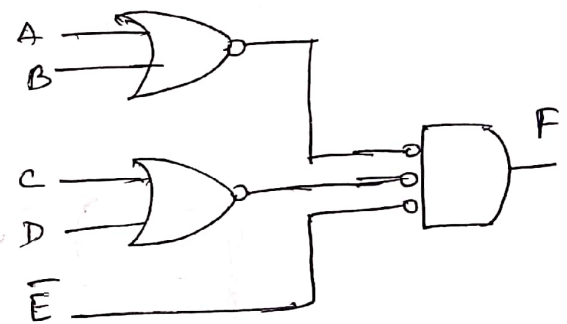
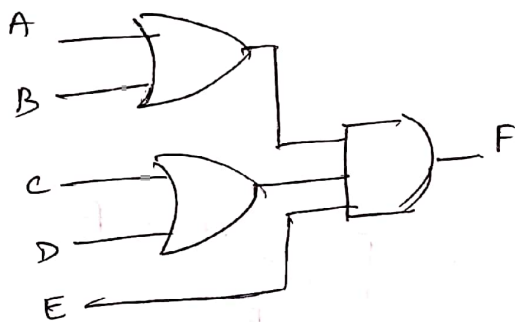
Example:- Implement the following function with NAND gates

$$F(x, y, z) = \Sigma(0, 6)$$

## NOR Implementation :-

- \* The NOR function is the dual of the NAND function. For this reason, all procedures & the rules for NOR logic are the dual of the corresponding procedures and rules developed for NAND logic.
- \* The implementation of a Boolean function with NOR gates requires that the function be simplified in products of sums form.
- \* Consider the following product of sums expression

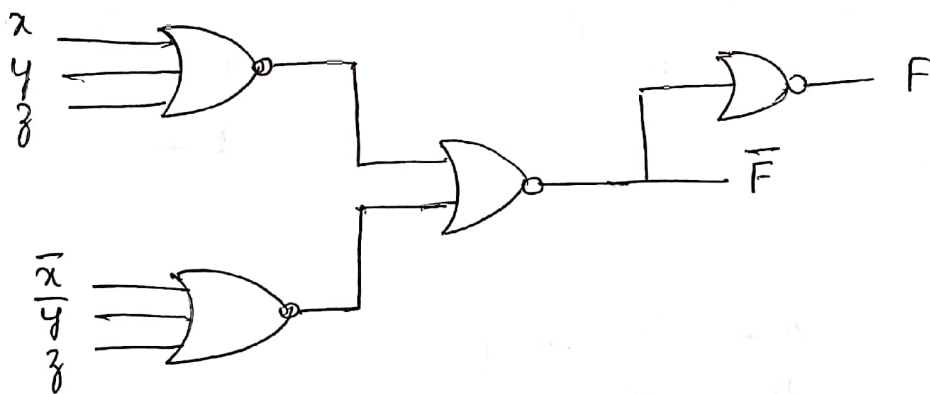
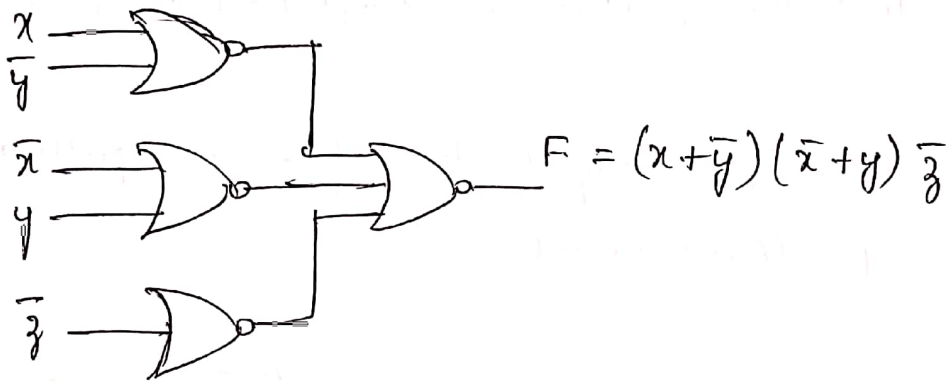
$$F = (A+B)(C+D)E$$



Example :- Implement the function with NOR gates.

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

This is the complement of the function in sum of products. Complement  $F'$  to obtain the simplified function in product of sums as required for NOR implemented.



$$\bar{F} = (x + y + z)(\bar{x} + \bar{y} + \bar{z})$$

Rules for NAND and NOR Implementation :-

Case	Function to simplify	Std. form to use	How to derive	Implement with	No. of levels to F
(a)	F	SOP <sub>x</sub>	Combine 1's in map	NAND	2
(b)	$\bar{F}$	SOP <sub>x</sub>	Combine 0's in map	NAND	3
(c)	F	POS <sub>x</sub>	Complement $F'$ in (b)	NOR	2
(d)	$\bar{F}$	POS <sub>x</sub>	Complement F in (a)	NOR	3



# Four-Variable K-map :-

$m_0$	$m_1$	$m_3$	$m_2$
$m_4$	$m_5$	$m_7$	$m_6$
$m_{12}$	$m_{13}$	$m_{15}$	$m_{14}$
$m_8$	$m_9$	$m_{11}$	$m_{10}$

$wx \backslash yz$	00	01	11	10
00	$\bar{w}\bar{x}\bar{y}\bar{z}$	$\bar{w}\bar{x}\bar{y}z$	$\bar{w}\bar{x}yz$	$\bar{w}\bar{x}y\bar{z}$
01	$\bar{w}x\bar{y}\bar{z}$	$\bar{w}x\bar{y}z$	$\bar{w}xyz$	$\bar{w}xy\bar{z}$
11	$wx\bar{y}\bar{z}$	$wx\bar{y}z$	$wxyz$	$wxy\bar{z}$
10	$w\bar{x}\bar{y}\bar{z}$	$w\bar{x}\bar{y}z$	$w\bar{x}yz$	$w\bar{x}y\bar{z}$

① Simplify the Boolean function

$$F(w, x, y, z) = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

$wx \backslash yz$	$\bar{y}\bar{z}$	$\bar{y}z$	$yz$	$y\bar{z}$
$\bar{w}\bar{x}$	1	1		1
$\bar{w}x$	1	1		1
$wx$	1	1		1
$w\bar{x}$	1	1		

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

In the map,

Quads - 2

Pairs - 1

Singles - 0

② Simplify the Boolean function.

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

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

Octets - 0

Quads - 2

pairs - 1

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

③ Simplify the Boolean function

$$F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

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

Quads - 4

Octets - 0

pairs - 0

$$F = CD + \bar{A}\bar{B} + BD + \bar{B}\bar{D}$$

- ④ Find the simplified Boolean expression for the 4-variable K-map shown in fig. below.

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

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

- ⑤ Obtain the simplified form of the Boolean expressions for the truth table given below, using K-map.

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

Sol<sup>n</sup>: From truth table.

Minterms are.

$m_0, m_1, m_2, m_3, m_7, m_8, m_9, m_{10}, m_{11}, m_{12},$  and  $m_{13}$ .

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

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

### K-map for Boolean expression in POS form:-

Eg:-  $F(A, B, C) = \sum m(1, 3, 4, 6)$

F is in terms of minterms. Minterm values are 1 in  $m_1, m_3, m_4$  &  $m_6$ . Hence in K-map, 1's are entered in cells 1, 3, 4 & 6. In all other cells viz cells 0, 2, 5 & 7, 0's are entered.

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

$$\overline{F}(A, B, C) = \overline{A}\overline{C} + AC$$

In order to obtain the Boolean expression for  $F(A, B, C)$  in POS form, DeMorgan's theorem is applied.

$$F(A, B, C) = \overline{\overline{F}} = \overline{\overline{A}\overline{C} + AC}$$

$$= (\overline{\overline{A}\overline{C}}) + (\overline{AC})$$

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

Note 1:- If  $f(A, B, C) = \sum m(1, 3, 4, 6)$

(i)  $\overline{f}(A, B, C) = \prod M(1, 3, 4, 6)$

(ii)  $f(A, B, C) = \prod M(0, 2, 5, 7)$

① Consider the 4-variable Boolean Expression

$$F(A, B, C, D) = \prod M(3, 7, 10, 11, 15)$$

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

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

In POS form,  $\bar{F}$

$$\bar{F} = \bar{f} = \overline{CD + A\bar{B}C}$$

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

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

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

② Obtain using K-map, the minimised Boolean Expression in (i) SOP form (ii) POS form for the truth table shown.

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

Sol: minterms with value 0.

$m_0, m_2, m_3, m_4, m_8, m_9, m_{10}, m_{14}$ .

minterms with value 1.

$m_1, m_5, m_6, m_7, m_{11}, m_{12}, m_{13} \text{ \& } m_{15}$ .

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



(i) In SOP form.

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

$$F = \bar{A}\bar{C}D + \bar{A}BC + AB\bar{C} + ACD$$

(ii) In POS form.

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

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

$$F = \bar{\bar{F}} = \overline{\bar{A}\bar{C}\bar{D} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + AC\bar{D}}$$

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

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

⑤ Simplify the expression  $\overline{AB + \overline{AB} + \overline{B}}$  using DeMorgan's theorems.

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

$$= (\overline{AB}) (\overline{AB}) (\overline{B})$$

DeMorgan's 1<sup>st</sup> theorem.

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

————— 2<sup>nd</sup> theorem &  $BB = B$

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

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

$$= 0 + 0$$

$$\therefore \overline{A}A = 0 \quad B\overline{B} = 0$$

$$Y = 0$$

⑥ Simplify  $AB + \overline{A}C + \overline{B}C$

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

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

$$= AB + C(\overline{AB})$$

$\therefore$  DeMorgan's 2<sup>nd</sup> theorem.

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

$$= AB \cdot 1 + \overline{A}B\overline{C}$$

$$= AB(1+C) + \overline{A}B\overline{C}$$

$$\therefore 1+C = 1$$

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

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

$$Y = AB + C$$

$$\therefore AB + \overline{A}B = 1$$

⑦  $Y = (\overline{A+B})(\overline{A+C})(\overline{B+C})$

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

$$= \overline{A} \cdot \overline{B} [\overline{A}\overline{B} + \overline{A}\overline{C} + \overline{B}\overline{C} + \overline{C}\overline{C}]$$

$$= \overline{A}\overline{B} + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + 0$$

$$\therefore \overline{A}\overline{A} = \overline{A} \quad \& \quad \overline{C}\overline{C} = 0$$

$$= \overline{A}\overline{B} [1 + \overline{C}] + \overline{A}\overline{B}\overline{C}$$

$$= \overline{A}\overline{B} \cdot 1 + \overline{A}\overline{B}\overline{C}$$

$$\therefore 1 + \overline{C} = 1$$

$$= \overline{A}\overline{B} (1 + \overline{C})$$

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

$$\therefore 1 + \overline{C} = 1$$



⑧ Realize the expressions using basic gates after the simplification.

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

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

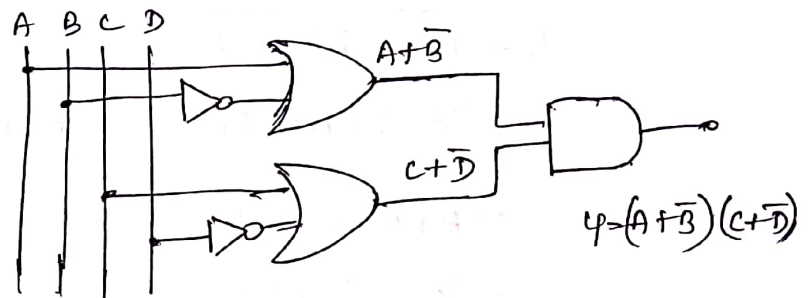
$$= (\overline{A}B)(\overline{C}D)$$

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

$$Y = (A+B)(C+D) \quad \checkmark$$

$$Y = AC + AD + BC + BD$$

↓  
more no of logic gates required.

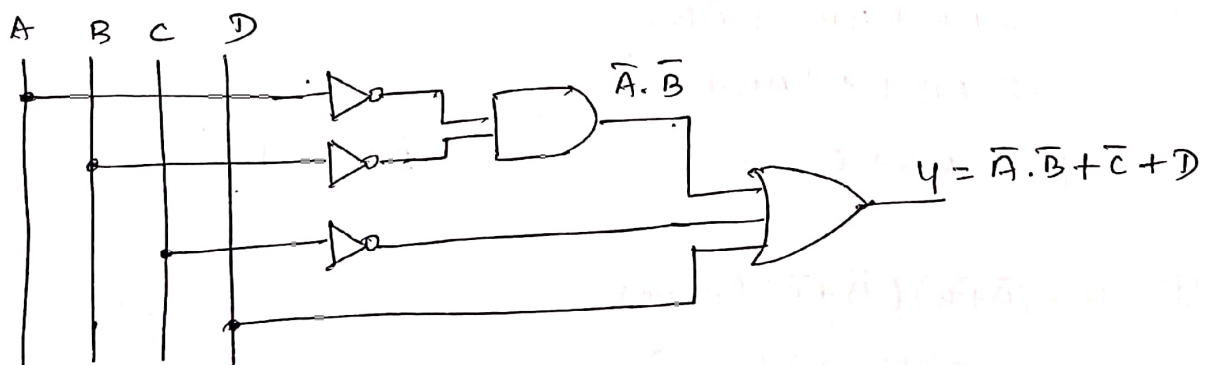


(ii)  $\overline{(A+B)C\overline{D}}$

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

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

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



## Prime Implicants:-

\* In choosing adjacent squares in a map, we must ensure that

- ① all the minterms of the function are covered when we combine the squares.
- ② the number of terms in the expression is minimized.
- ③ there are no redundant terms (i.e., minterms already covered by other terms).

\* Sometimes there may be 2 or more expressions that satisfy the simplification criteria. The procedure for combining squares in the map may be made more systematic if we understand the meaning of two special types of terms.

\* A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map.

\* If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be essential.

\* The prime implicants of a function can be obtained from the map by combining all possible maximum numbers of squares.

- ① a single 1 on a map represents a prime implicant if it is not adjacent to any other 1's.
- ② Two adjacent 1's form a prime implicant, provided that they are not within a group of 4 adjacent squares.
- ③ Four adjacent 1's form a prime implicant if they are not within a group of 8 adjacent squares & so on.

Ex-1-  $F(A, B, C, D) = \sum (0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$

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

Implicants  $\rightarrow$  All groupings of 1's are implicants.

$CD, A\bar{B}, AD, \bar{B}C, \bar{B}\bar{D}, ABD.$

Prime implicants  $\rightarrow$