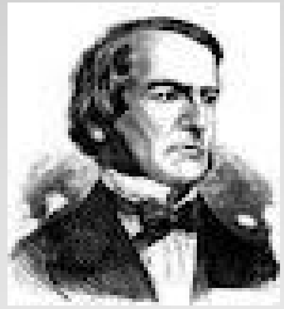
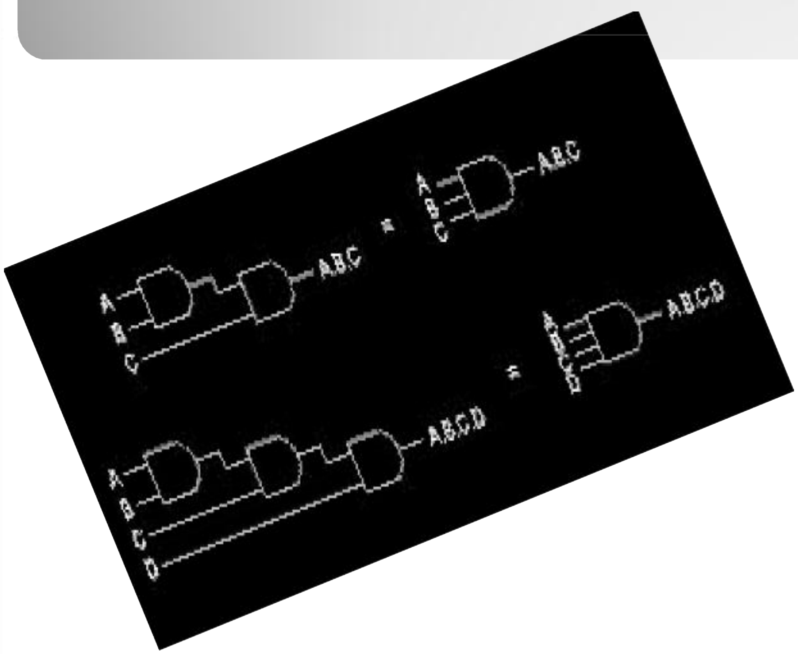


Boolean Algebra



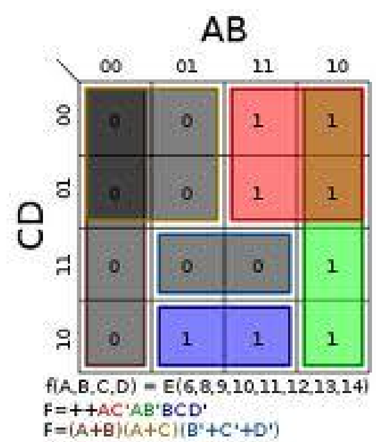


Table of Contents

Development of Boolean Algebra :

- ✓ **Boolean algebra**, as developed in 1854 by George Boole in his book An Investigation of the Laws of Thought, is a variant of ordinary elementary algebra differing in its values, operations, and laws.
- ✓ Boolean algebra is the algebra of truth values 0 and 1.
- ✓ As logic problems are binary decisions and Boolean algebra effectively deals with these binary values.
- ✓ So it is also called **'Switching Algebra'**.

Binary valued Quantities :

- ✓ The decision which results into either YES(TRUE) or NO(FALSE) is called a **Binary Decision**.
- ✓ The statements which can be determined to be True or False are called **logical statements** or **truth functions**.
- ✓ The result TRUE or FALSE are called **truth values**.
- ✓ Truth values are represented by logical constant TRUE and FALSE or 1 and 0.
- ✓ 1 means TRUE and 0 means FALSE.
- ✓ Variables which can store these truth values are called **logical variables** or **binary valued variables**.

- Logical Function or Compound Statement
- Logical Operators
- Evaluation of Boolean Expression Using Truth Table

Logical Function or Compound Statement :

- ✓ Logic statements or truth functions are combined with the help of Logical Operators like AND, OR and NOY to form a *Compound statement* or *Logical function, such as*

He prefers tea not coffee.

- ✓ The logical operators are also used to combine logical variables and logical variables and logical constants to form *logical expression*, such as x, y and z are logical variables

X NOT Y OR Z

Y AND X OR Z

Truth Table :

- ✓ Truth Table is a table which represents all the possible values of logical variables/ statements along with all the possible results of the given combinations of values.
- ✓ For example, we have two logical variable x and y which can have either TRUE or False values.
- ✓ Following is the all the possible combination of values these variable can have in tabular form :

X	Y	R
1	1	1
1	0	0
0	1	0
0	0	0

- ✓ 1 represents TRUE value and 0 represents FALSE value.
- ✓ If result of any logical statement or expression is always TRUE or 1, it is called *Tautology*.
- ✓ If the result is always FALSE or 0, it is called *Fallacy*.

BACK

Logical Operators :

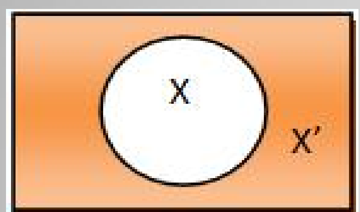
NOT Operator :

- ✓ Operates on a single variable.
- ✓ Singular or Unary operation.
- ✓ Operation performed by NOT is called complementation.
- ✓ '-' (bar) or " ' "symbols are used.
- ✓ X' means complement of X.

Rules : 0 = 1

1 = 0

In Venn diagram shaded area represents x'.



Truth Table :

X	Y
0	1
1	0

OR Operator :

✓Denotes operation called logical addition.

✓' + ' symbol is used.

✓X + Y can be read as X OR Y.

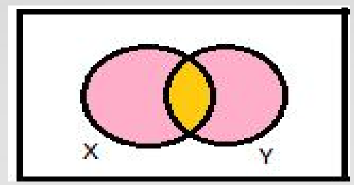
Rules : 0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 1

In Venn diagram shaded area represents X + Y



Truth Table :

X	Y	X + Y
0	0	0
0	1	1
1	0	1
1	1	1

AND Operator :

✓Denotes operation called logical multiplication.

✓' . ' symbol is used.

✓X . Y can be read as X AND Y.

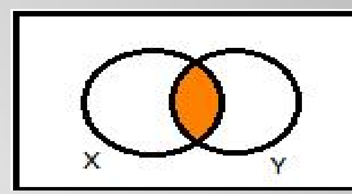
Rules : 0 . 0 = 0

0 . 1 = 0

1 . 0 = 0

1 . 1 = 1

In Venn diagram shaded area represents (X . Y).



Truth Table :

X	Y	X + Y
0	0	1
0	1	0
1	0	0
1	1	0

BACK

Evaluation of Boolean Expression Using Truth Table :

- ✓ Logical variables are combined by means of logical operators (AND, OR, NOT) to form *Boolean expression*.
- ✓ Consider the expression
- ✓ Possible combination of values may be arranged in ascending order as in following table.

X	Y	Z	$Y \cdot Z$	$\overline{YZ}$	$X + \overline{YZ}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	1

BACK

NOTE : While evaluating Boolean expression we always have to taken care of precedence. The order of evaluation of logical operator is firstly NOT then AND and then OR. If there are parenthesis, then the expression in parenthesis is evaluated first.

- ✓ In the computers, Boolean operations are performed by logic gates.
- ✓ A **Gate** is simply an electronic circuit which operates on one or more signals to produce an output signal.
- ✓ As gates can be evaluated with Boolean algebra they are often called *logic circuits*.
- ✓ Input and output signals are either low(denotes 0) voltage or high(denotes 1) voltage.
- ✓ There are three types of gate : **1. Inverter (NOT Gate)**
  2. OR Gate
  3. AND Gate

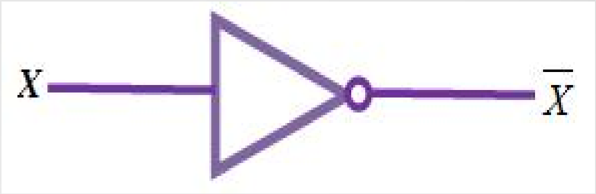
1. Inverter (NOT Gate) :

- ✓ An Inverter (NOT Gate) is a gate with only one input signal and one output signal.
- ✓ The output state is always the opposite of the input state.
- ✓ Output is sometimes called *complement* of the input.

Truth Table

X	$\overline{X}$
0	1
1	0

NOT gate symbol



2. OR Gate :

- ✓ The OR Gate has two or more input signals but only one output signal.
- ✓ If any of the input signal is 1, the output signal is 1.
- ✓ If all inputs are 0 then output is also 0.

Truth Table :

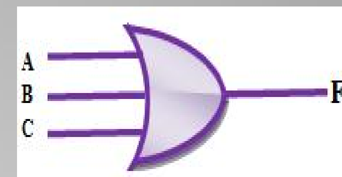
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

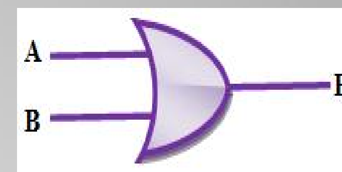
F = X . Y

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

F = X . Y . Z

Symbol :





3. AND Gate :

- ✓ The AND Gate can have two or more than two input and produce an output signal.
- ✓ If any of the input signal is 0, the output signal is 0.
- ✓ If all inputs are 0 then output is also 0.

Truth Table :

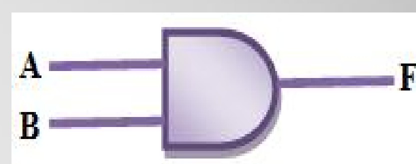
X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

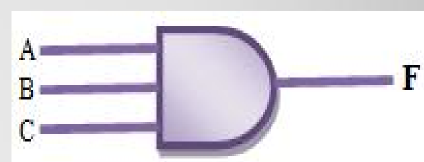
F = X . Y

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

F = X . Y . Z

Symbol :

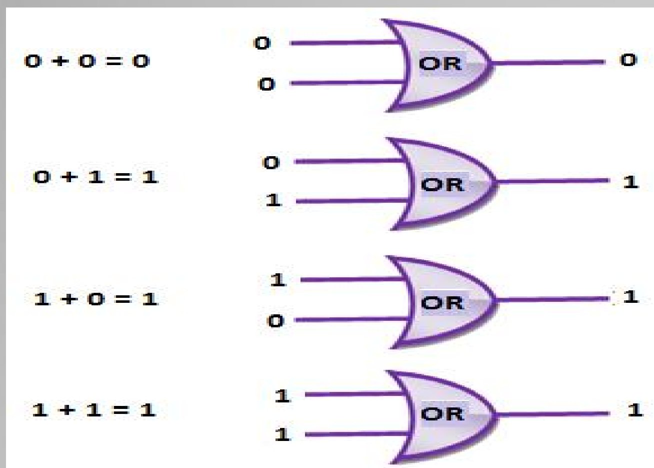




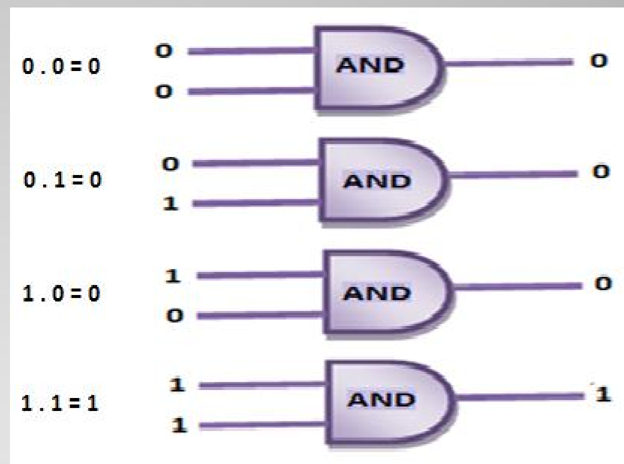
- ✓ Boolean Algebra consists of *fundamental laws* which are known as *Basic postulates of Boolean algebra.*
- ✓ These postulates state basic relation in Boolean algebra, that follow :

I. If X not equal to 0 then X equal to 1; and If X not equal to 1 then X equal to 0

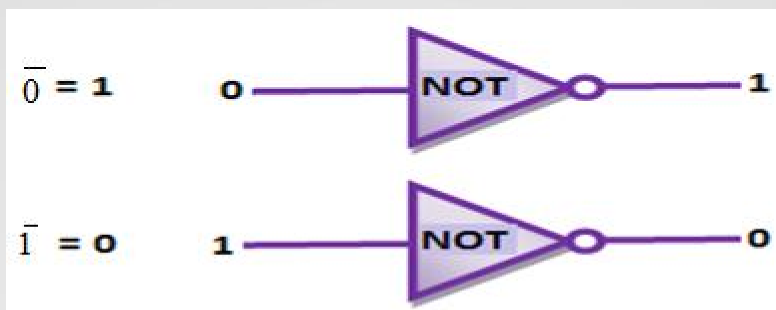
II. OR Relations (Logical Addition)



III. AND Relations (Logical Multiplication)



IV. Complement Rules



- ✓ The *principle of duality* states that starting with a Boolean relation, another Boolean relation can be derived by :

  1 Changing each OR sign(+) to an AND sign(.).

  2. Changing each AND sign(.) to an OR sign(+).

  3. Replacing each 0 by 1 and each 1 by 0.
- ✓ The derived relation using duality principle is called *dual of original expression*.
- ✓ For example *postulate II* states

  (a) 0 + 0 = 0 (b) 0 + 1 = 1 (c) 1 + 0 = 1 (d) 1 + 1 = 1
- ✓ Now according to *principle of duality* we changed '+' to '.' and 0 to 1.
- ✓ These become,

  (i) 1 . 1 = 1 (ii) 1 . 0 = 0 (iii) 0 . 1 = 0 (iv) 0 . 0 = 0

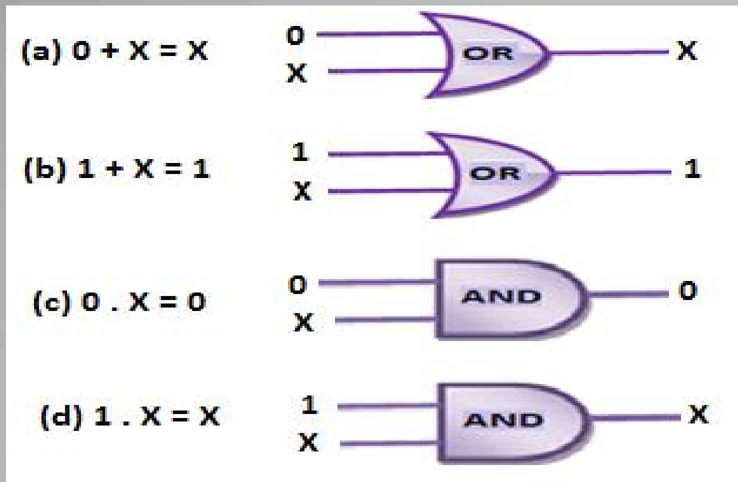
  which is same as *postulate III.*
- ✓ So i, ii, iii, iv are duals of a, b, c, d.

Basic Theorems of Boolean Algebra :

- ✓ Basic postulates of Boolean algebra are used to define basic theorems of *Boolean algebra*.
- ✓ Provide all the tools necessary for manipulating Boolean expression.

1. Properties of 0 and 1
2. Indempotence Law
3. Involution
4. Complementarity Law
5. Commutative Law
6. Associative Law
7. Distributive law
8. Absorption Law
9. Some Other Rules of Boolean Algebra

1. Properties of 0 and 1



Truth Table

0	X	R
0	0	0
0	1	1

1	X	R
1	0	1
1	1	1

0	X	R
0	0	0
0	1	0

1	X	R
1	0	0
1	1	1

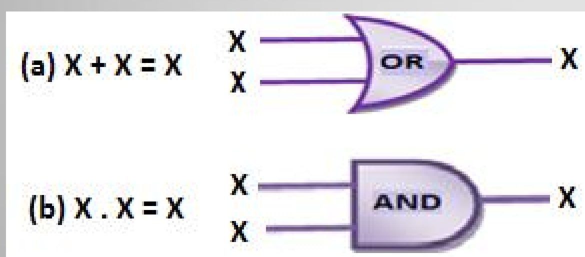
✓From Truth Table w can see that all four properties are proved.

✓In (b) both the values (0 and 1) ORed with 1 produce the output as 1. So *1 + X = 1* is a ***tautology.***

✓In (c) both the values (0 and 1) ANDed with 0 produce the output as 0. So *0 . X=0* is a ***fallacy.***

✓Here we can observe that propertied *b* and *c* are duals of each other and properties *a* and *d* are duals of each other.

2. Indempotence Law



Truth Table

X	X	R
0	0	0
1	1	1

0 + 0 = 0

1 + 1 = 1

X	X	R
0	0	0
1	1	1

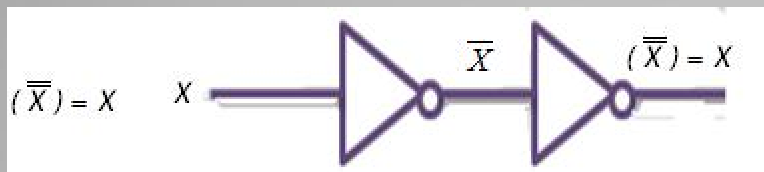
0 . 0 = 0

1 . 1 = 1

✓ Form Truth Table it is proved that *X + X = X* and *X . X = X* , as it holds true for both values of X.

Here, It is proved that *(a)* and *(b)* are duals of each other.

3. Involution

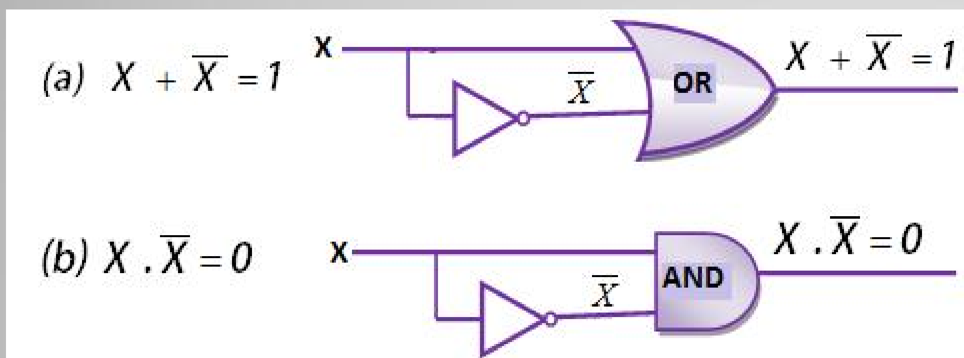


Truth Table

X	$\overline{X}$	$\overline{\overline{X}}$
0	1	0
1	0	1

- ✓ By Truth Table it is proved that $(\overline{\overline{X}}) = X$
- ✓ This low is also called *double-inversion rule.*

4. Complementarity Law



Truth Table

X	$\overline{X}$	$X + \overline{X}$
0	1	0
1	0	1

0 + 1 = 1
1 + 0 = 1

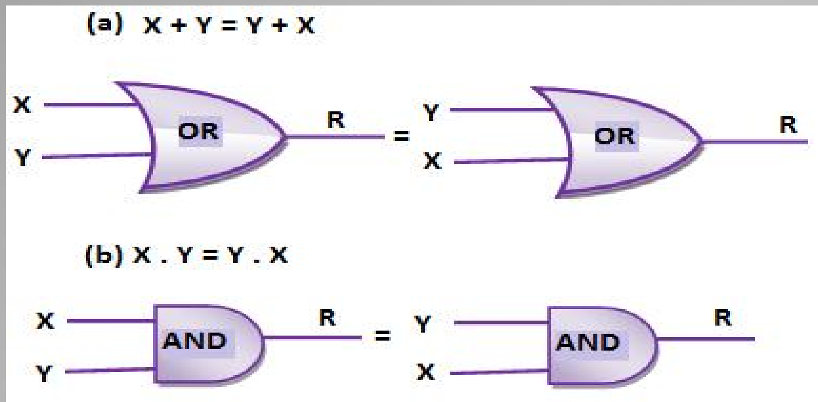
X	$\overline{X}$	$X . \overline{X}$
0	1	0
1	0	0

0 . 1 = 0
1 . 0 = 0

- ✓ $X + \overline{X} = 1$, as it holds true for both possible values of x. Thus proved. It is ***tautology***.
- ✓ $X . \overline{X} = 0$, as it holds true for both possible values of x. Thus proved.
- ✓Observe here $X . \overline{X} = 0$ is dual of $X + \overline{X} = 1$.
- ✓Changing (+) to (.) and 1 to 0, we get $X . \overline{X} = 0$. It is a ***fallacy.***

BACK

5. Commutative Law



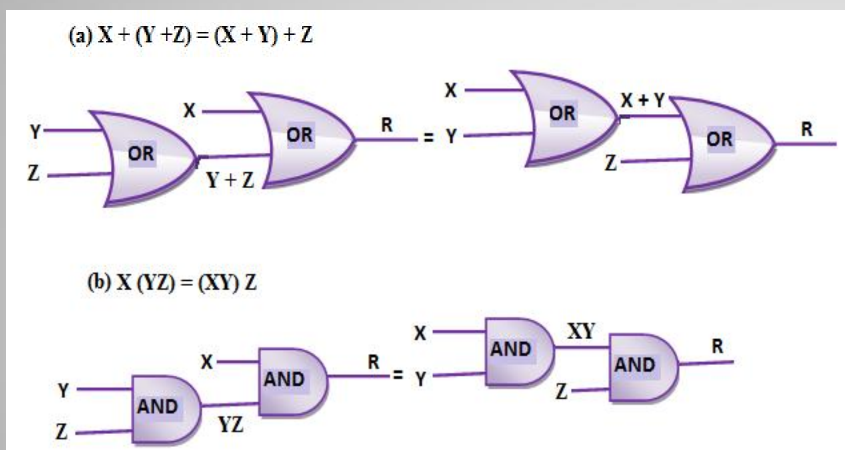
Truth Table

X	Y	X + Y	Y + X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

X	Y	X.Y	Y.X
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

✓ Form Truth Table it is proved that ,
X + Y = Y + X and
X . Y = Y . X

6 Associative Law



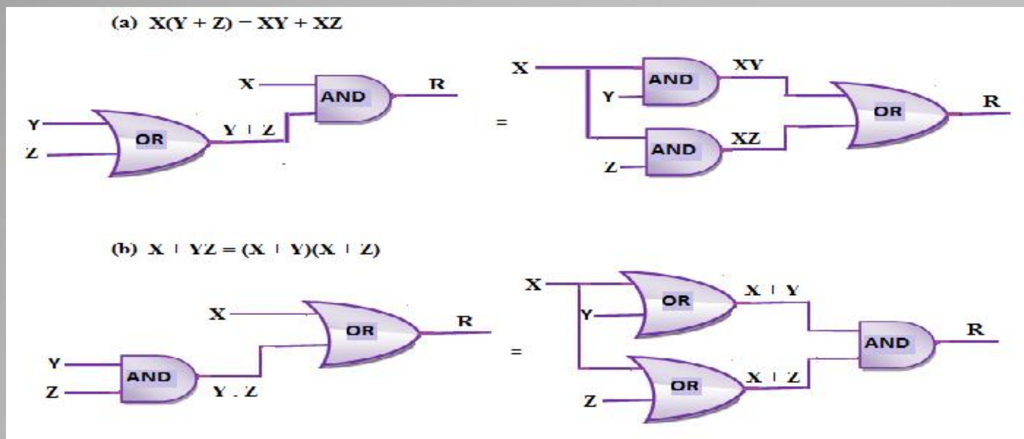
Truth Table

X	Y	Z	Y+Z	X+Y	X+(Y+Z)	(X+Y)+Z
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

✓From Truth Table it is proved that X + (Y+Z) = (X+Y) + Z.
✓Since rule (b) is dual of rule (a), so it is also proved.

BACK

7. Distributive law

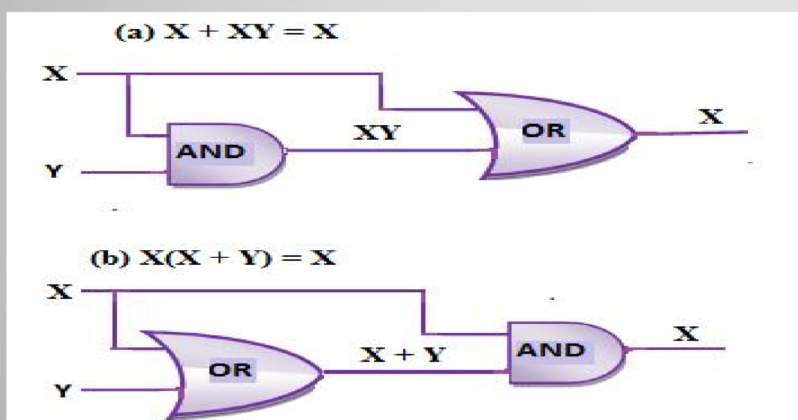


Truth Table

X	Y	Z	Y+Z	XY	XZ	X(Y+Z)	XY+XZ
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

- ✓ By Truth Table it is proved that *X(Y + Z) = XY + XZ*.
- ✓ Since rule *(b)* is dual of rule *(a)*, hence it is also proved.

8. Absorption Law



Truth Table

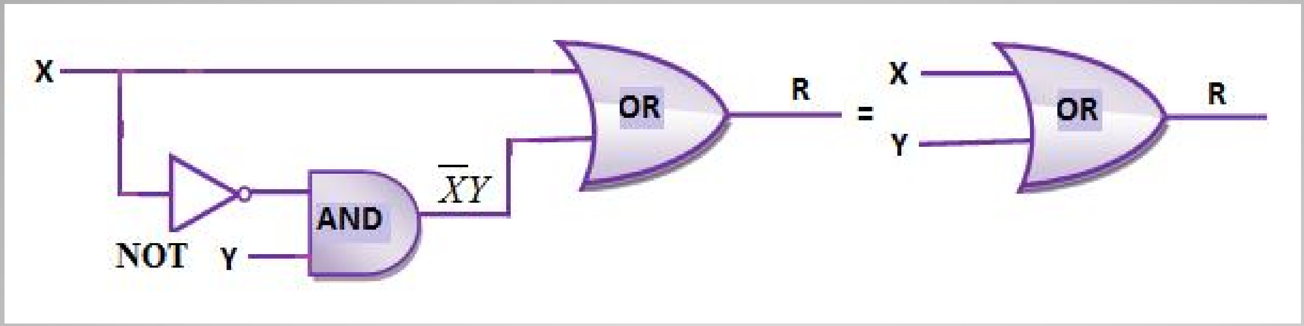
X	Y	XY	X+XY
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

- ✓ By Truth Table it is proved that *X* + *XY* = *X*.
- ✓ Since rule *(b)* is dual of rule *(a)*, hence it is also proved.

Some Other Rules of Boolean Algebra

There are some more rules of Boolean algebra which are given below :

$$X + \overline{X}Y = X + Y$$



Proof . L.H.S. = $X + \overline{X}\,Y$

$= X \,.\, 1 + \overline{X}\,Y$ (X . 1 = X property of 0 and 1)

$= X(1 + Y) + \overline{X}\;Y$ (1 + Y = 1 property of 0 and 1)

$= X + XY + \overline{X}\;Y$

$= X + Y\;(X + \overline{X})$

$= X + Y \,.\, 1$ (X + $\overline{X}$ = 1 complimentarily low)

$= X + Y$ (Y . 1 = Y property of 0 and 1)

= R.H.S. Hence proved.

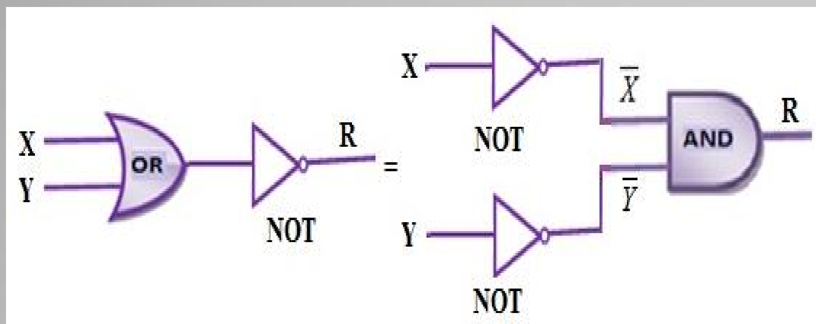
BACK

A mathematician named Demorgan developed a pair of important rules which is the most powerful identities used in Boolean algebra.

<u>Demorgan's First Theorem</u>

✓It states that $\overline{X+Y} = \overline{X}\,\overline{Y}$

<u>Truth Table</u>



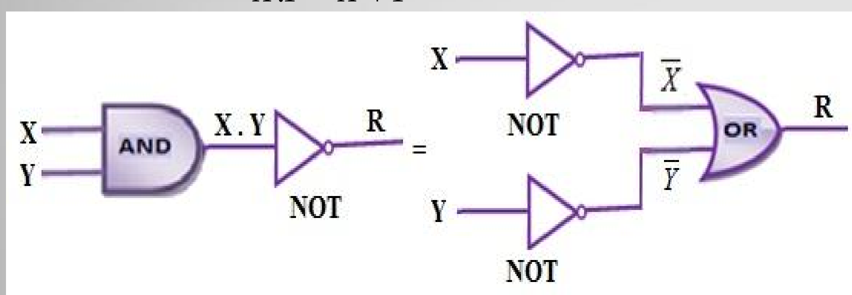
X	Y	$\overline{X}$	$\overline{Y}$	X + Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

✓From Truth Table it is proved that $\overline{X+Y} = \overline{X}\,\overline{Y}$

<u>Demorgan's Second Theorem</u>

✓It states that $\overline{X.Y} = \overline{X} + \overline{Y}$

<u>Truth Table</u>



X	Y	$\overline{X}$	$\overline{Y}$	X.Y	$\overline{X}\,\overline{Y}$	$\overline{X}+\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

✓From Truth Table it is proved that $\overline{X.Y} = \overline{X} + \overline{Y}$

Boolean expression which consist of a single variable or its complement such as, *X* or *Y* or are known as *literals*.

- **Minterms**
- **Maxterms**
- **Canonical Expression**
  - Sum-of-Products(S-O-P)
  - Product-of-Sum form
  - Sum term v/s Maxterm and Product term v/s Minterm

$\overline{Z}$

Minterms :

Minterm is a product of all the literals within the logic system.

Steps:

1. Convert the given expression in sum of products form.
2. If any variable is missing, multiply that term with (missing term + missing term) factor.
3. Expand the expression.
4. Remove all duplicate terms and we will have minterm form of an expression.

Example **:** Convert $X + Y$ to minterms.

$X + Y = X.1 + Y.1$

$= X.(Y + \overline{Y}) + Y.(X + \overline{X})$

$(X + \overline{X} = 1$ complementarity law)

$= XY + X\overline{Y} + XY + \overline{X}Y$

$= XY + XY + X\overline{Y} + \overline{X}Y$

$= XY + X\overline{Y} + \overline{X}Y$

$(X . X = X$ Idempotent law)

Shorthand minterm notation

To form this notation, following steps are to be followed :

Let's take example to find minterm designation of $X\overline{Y}\overline{Z}$

1. Copy Original form = $X\overline{Y}\overline{Z}$
2. Substitute 1's for non barred and 0's for barred letters.

   Binary equivalent = 100

   Decimal equivalent of 100 = $1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$

   $= 4 + 0 + 0 = 4$
3. Express as decimal subscript of $m = m4$

   Thus $X\overline{Y}\overline{Z} = m_4$

BACK

Maxterms :

Maxterm is a product of all the literals within the logic system.

Steps:

1. Convert the given expression in products of sum form.
2. If any variable is missing, sum that term with (missing term + missing term) factor.
3. Expand the expression.
4. Remove all duplicate terms and we will have maxterm form of an expression.

Example **:** Convert $X \cdot Y$ to maxterms.

$$X \cdot Y = (X+0) \cdot (Y+0)$$

$$= (X+(Y \cdot \overline{Y})) \cdot (Y+(X \cdot \overline{X}))$$ $(X \cdot \overline{X} = 0$ complementarity law$)$

$$= (X+Y) \cdot (X+\overline{Y}) \cdot (Y+X) \cdot (Y+\overline{X})$$

$$= (X+Y) \cdot (X+\overline{Y}) \cdot (Y+\overline{X})$$ $(X + X = X$ Idempotent law$)$

Minterms & Maxterms for 2 variables

x	y	Index	Minterm	Maxterm
0	0	0	$m_0 = x\ y$	$M_0 = x + y$
0	1	1	$m_1 = x\ y$	$M_1 = x + y$
1	0	2	$m_2 = x\ y$	$M_2 = x + y$
1	1	3	$m_3 = x\ y$	$M_3 = x + y$

- ✓ There are 2^n minterms and maxterms for n variables.
- ✓ The minterm m_i should evaluate to 1 for each combination of x and y.
- ✓ The maxterm is the complement of the minterm.

BACK

Canonical Expression

Boolean Expression composed entirely either of Minterms or maxterms is referred to as *canonical Expression.*

There are two types of canonical Expression as following :

(i)Sum-of-Products (S-O-P) form (ii) Product-of-sums (P-O-S) form

Sum-of-Products (S-O-P) :

- ✓ When a boolean expression is represented purely as sum of minterms, it is said to be in *Canonical Sum-of-Products Form.*
- ✓ This form of expression is also referred to as *Mnterm canonical form of boolean expression.*
- ✓ A logical expression is derived from two sets of known values:
  1. various possible input values
  2. the desired output values for each of the input combinations.

The truth table method for arriving at the desired expresion is as follows:

1. For a given expression, prepare a truth table for all possible combinations of inputs.
2. Add a new column for miterms and list the minterms for all the combinations.
3. Add all the minterms for which there is output as 1. This gives you the desired canonical *S-O-P* expression.

Truth Table for Minterms (2- input) :

X	Y	Z	Minterms
0	0	1	$\overline{X}\,\overline{Y}$
0	1	0	$\overline{X}Y$
1	0	1	$X\overline{Y}$
1	1	1	XY

- ✓ Adding all the minterms for which output is 1,we get

$$\overline{X}\,\overline{Y} + X\overline{Y} + XZ = Z$$

- ✓ The relationship between inputs and outputs is to be as follows:
  a. When X = 0 and Y = 0 then Z = 1
  b. When X = 0 and Y = 1 then Z = 0
  c. When X = 1 and Y = 0 then Z = 1
  d. When X = 1 and Y = 1 then Z = 1

Continued

BACK

Sum-of-Products (S-O-P) Continued :

Truth Table for Minterms (3- input) :

X	Y	X	F	Minterms
0	0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$
0	0	1	1	$\overline{X}\,\overline{Y}Z$
0	1	0	1	$\overline{X}Y\overline{Z}$
0	1	1	0	$\overline{X}YZ$
1	0	0	1	$X\overline{Y}\,\overline{Z}$
1	0	1	0	$X\overline{Y}Z$
1	1	0	0	$XY\overline{Z}$
1	1	1	0	XYZ

- ✓ A boolean function F defined on three input variables X, Y and Z is 1 if and only if number of 1 inputs is odd otherwise output is 0.
- ✓ All the possible combinations when one of input is odd are X = 1, Y = 0, Z = 0
  X = 0, Y = 1, Z = 0
  X = 0, Y = 0, Z = 1

Adding all the minterms for which output is 1, we get

$$\overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} = F$$

Let's take another example with use of **Algebraic Method**.

EX. Convert $\overline{(\overline{X}Y) + (\overline{X}\,\overline{Z})}$ *into canonical sum of products form.*

Rule 1. Simplify the given expression using appropriate theorem/rules.

$$\overline{(\overline{X}Y) + (\overline{X}\,\overline{Z})} = (X + \overline{Y})(X + Z)$$

$$= X + \overline{Y}Z$$

Rule 2. Wherever a literal is missing, multiply that term with (missing variable + $\overline{\text{missing variable}}$)

$$= X + \overline{Y}Z$$

$$= X(Y + \overline{Y})(Z + \overline{Z}) + (X + \overline{X})\overline{Y}Z$$

$$= (XY + X\overline{Y})(Z + \overline{Z}) + X\overline{Y}Z + \overline{X}\,\overline{Y}Z$$

$$= Z(XY + X\overline{Y}) + \overline{Z}(XY + X\overline{Y}) + X\overline{Y}Z + \overline{X}\,\overline{Y}Z$$

$$= XYZ + X\overline{Y}Z + XY\overline{Z} + X\overline{Y}\,\overline{Z} + X\overline{Y}Z + \overline{X}\,\overline{Y}Z$$

Rule 3. Remove duplicate terms

$$XYZ + X\overline{Y}Z + XY\overline{Z} + X\overline{Y}\,\overline{Z} + \overline{X}\,\overline{Y}Z$$

This is the desired Canonical Sum-of-Products form.

Continued

BACK

Sum-of-Products (S-O-P) Continued :

Converting Shorthand Notation to Minterms

Rule 1. Find binary equivalent of decimal subscript such as, for m6 subscript is 6, binary equivalent of 6 is 110.

Rule 2. Foe every 1's write the variable as it is and for 0's write variable's complemented form *i.e.* , for 110 it is $XY\overline{Z}$, $XY\overline{Z}$ is the required minterm for m_6 .

Example :

Convert the three input fuction F=∑(0, 1, 2, 5) into its canonical Sum-of-Products form.

Solution : If three inputs we take as X, Y and Z then

$$F = m_0 + m_1 + m_2 + m_5$$

$$m_0 = 000 \Rightarrow \overline{X}\,\overline{Y}\,\overline{Z}$$

$$m_1 = 001 \Rightarrow \overline{X}\,\overline{Y}\,Z$$

$$m_2 = 010 \Rightarrow \overline{X}Y\overline{Z}$$

$$m_5 = 101 \Rightarrow X\overline{Y}Z$$

Canonical S-O-P form of the expression is

$$\overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}Z$$

BACK

Derivation of Boolean Expression Continued :

Product-of-sums (P-O-S) :

When a Boolean expression is represented purely as product of Maxterms, it is said to be in *Canonical Product-of-Sums Form.*

This form of expression is also referred to as *Maxterm canonical form of Boolean expression.*

(a) Truth Table Method

The truth table method for arriving at the desired expression is as follows :

1.Prepare a table of inputs and outputs

2. Add one additional column of sum terms.

3.For each row of the table, a sum term is formed by adding all the variables in complemented or uncomplemented form *i.e.*, if input value for a given variable is 1, variable is complemented and if 0, not complemented, such as for X=0, Y=1, Z=1, sum term will be $X + \overline{Y} + \overline{Z}$

4. The desired expression is the product of the sums form the rows in which the output is 0.

X	Y	X	F	Maxterms
0	0	0	1	$X + Y + Z$
0	0	1	0	$X + Y + \overline{Z}$
0	1	0	1	$X + \overline{Y} + Z$
0	1	1	0	$X + \overline{Y} + \overline{Z}$
1	0	0	1	$\overline{X} + Y + Z$
1	0	1	0	$\overline{X} + Y + \overline{Z}$
1	1	0	1	$\overline{X} + \overline{Y} + Z$
1	1	1	1	$\overline{X} + \overline{Y} + \overline{Z}$

By multiplying all the maxterms for which output is 0, we get $(X + Y + \overline{Z})(X + \overline{Y} + \overline{Z})(\overline{X} + Y + \overline{Z})$

Continued

BACK

Product-of-sums (P-O-S) Continued:

(a) Algebraic Method

Let's understand this method with the help of following example.

Example : *Express* $\overline{X}Y+Y(\overline{Z}(\overline{Z}+Y))$ *into canonical product-of-sums form.*

Rule 1. Simplify the given expression using appropriate theorems/rules :

$\overline{X}Y+Y(\overline{Z}(\overline{Z}+Y))$ $=\overline{X}Y+Y(\overline{Z}\,\overline{Z}+Y\overline{Z})$ (using the distributive law *i.e.*, X(Y+Z) = XY + XZ)

$=\overline{X}Y+Y(\overline{Z}+Y\overline{Z})$ $(\overline{Z}.\overline{Z}=\overline{Z}\,as\,X.X=X)$

$=\overline{X}Y+Y.\overline{Z}(1+Y)$

$=\overline{X}Y+Y\overline{Z}.1$ $(1+Y=1)$

$=\overline{X}Y+Y\overline{Z}$

Rule 2. To convert into product of sums form, apply the Boolean algebra rule which states that

$X + YZ = (X+Y)(X+Z)$

Now applying this rule we get,

$\overline{X}Y+Y\overline{Z}=(\overline{X}Y+Y)(\overline{X}Y+\overline{Z})$

$=(Y+\overline{X}Z)(\overline{Z}+\overline{X}Y)$

$=(Y+\overline{X})(Y+Y)(\overline{Z}+\overline{X})(\overline{Z}+Y)$

$=(\overline{X}+Y)(Y)(\overline{X}+\overline{Z})(Y+\overline{Z})$ $(\because X+Y=Y+X)$

Rule 3. After converting into product of sum terms, in a sum term for a missing variable add $(\because Y+Y=Y)$

(missing variable . missing variable) such as, if variable Y is missing add $Y\overline{Y}$.

i.e., $(\overline{X}+Y)(Y)(\overline{X}+\overline{Z})(Y+\overline{Z})$

Terms: **1** **2** **3** **4** $=(\overline{X}+Y+Z\overline{Z})(X\overline{X}+Y+Z\overline{Z})(\overline{X}+Y\overline{Y}+\overline{Z})(X\overline{X}+Y+\overline{Z})$

Continued

BACK

Product-of-sums (P-O-S) Continued:

Rule 4. Keep on simplifying the expression (using the rule, *X+YZ= (X+Y)(X+Z)*) until you get product of sum terms which are Maxterms.

$$\Rightarrow(\overline{X}+Y+Z\overline{Z})(X\overline{X}+Y+Z\overline{Z})(\overline{X}+Y\overline{Y}+\overline{Z})(X\overline{X}+Y+\overline{Z})$$

$$=(\overline{X}+Y+Z)(\overline{X}+Y+\overline{Z})(X\overline{X}+Y+Z)(X\overline{X}+Y+Z)(\overline{X}+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(X+Y+\overline{Z})(\overline{X}+Y+\overline{Z})$$

$$=(\overline{X}+Y+Z)(\overline{X}+Y+\overline{Z})(X+Y+Z)(\overline{X}+Y+Z)(X+Y+\overline{Z})(\overline{X}+Y+\overline{Z})(\overline{X}+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})(X+Y+\overline{Z})(\overline{X}+Y+\overline{Z})$$

Rule 5. Removing all the duplicate terms we get

$$(\overline{X}+Y+Z)(\overline{X}+Y+\overline{Z})(X+Y+Z)(X+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})$$

This is the desired canonical product of sums form of expression.

Shorthand Maxterm Notation

Shorthand notation for the above given canonical product of sums expression is

$$F = \prod (0, 1, 4, 5, 7)$$

This specifies thet output F is product of 0th, 1st, 4th,5th and 7th Maxterms

i.e., $F = M_0 \,.\, M_1 \,.\, M_4 \,.\, M_5 \,.\, M_7$

Here M_0 means Maxterm for Binary equivalent of 0 i.e., 0 0 0

$\Rightarrow$ X= 0, Y = 0, Z = 0

And Maxterm will be (X + Y + Z) (Complement the variable if input is 1 otherwise not)

Similarly, M1 means 0 0 1 $\Rightarrow$ X + Y + Z

As $F = M_0 \,.\, M_1 \,.\, M_4 \,.\, M_5 \,.\, M_7$

\and $M_0 = 0\,0\,0$ $X + Y + Z$ $M_4 = 1\,0\,0$ $\overline{X} + Y + Z$ $M_7 = 1\,1\,1$ $\overline{X} + \overline{Y} + \overline{Z}$

$M_1 = 0\,0\,1$ $X + Y + \overline{Z}$ $M_5 = 1\,0\,1$ $\overline{X} + Y + \overline{Z}$

$\Rightarrow$ $F= (\overline{X}+Y+Z)(\overline{X}+Y+\overline{Z})(X+Y+Z)(X+Y+\overline{Z})(\overline{X}+\overline{Y}+\overline{Z})$

BACK

Derivation of Boolean Expression Continued :

Sum term v/s Maxerm :

Sum term	Maxterm
Sum term means sum of the variables. It does not necessarily mean that all the variables must be included.	Maxterm means sum term having all the variables.
For example, for a 3 variables F(X, Y, Z), Functions $X + Y, X + Z, \overline{Y} + Z$ *etc*. are sum terms.	Whereas $X + Y + Z + \overline{X} + Y + \overline{Z} + \overline{X} + \overline{Y} + Z$ *etc*. are Maxterms.

Product term v/s Minterm :

Product term	Minterm
Product term means product of the variables, not necessarily all the variables.	Minterm means product of all the variables.
For a 3 variable *(a, b, c)* function $ab, \overline{b}\,\overline{c}, \overline{b}c, \overline{a}c$ *etc* are product terms.	Whereas $ab\overline{c}, abc, \overline{a}\,\overline{b}c$ *etc*. ate Maxterms.

Canonical S-O-P or P-O-S expression v/s Simple S-O-P or P-O-S expression :

Canonical S-O-P or P-O-S expression	Simple S-O-P or P-O-S expression
Must have all the Maxterms or Minterms respectively	Can just have product terms or sum terms.

BACK

A minimized Boolean expression means less number of gates which means simplified circuitary. This section deals with two methods of simplification of Boolean expressions.

1. Algebraic Method

This method makes use of Boolean postulates, rules and theorems to simplify the expressions.

Example 1 : Simplify $A\overline{B}C\overline{D} + A\overline{B}CD + ABC\overline{D} + ABCD$.

Solution :

$$A\overline{B}C\overline{D} + A\overline{B}CD + ABC\overline{D} + ABCD$$

$$= A\overline{B}C(\overline{D} + D) + ABC(\overline{D} + D) = A\overline{B}C.1 + ABC.1 \qquad (\overline{D} + D = 1)$$

$$= AC(\overline{B} + B) = AC.1 = AC \qquad (\overline{B} + B = 1)$$

Example 2: *Reduce* $\overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XY\overline{Z}$.

Solution : $\overline{X}\,\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XY\overline{Z} = \overline{X}(\overline{Y}\,\overline{Z} + Y\overline{Z}) + X(\overline{Y}\,\overline{Z} + Y\overline{Z})$

$$= \overline{X}(\overline{Z}(\overline{Y} + Y) + X(\overline{Z}(\overline{Y} + Y))$$

$$= \overline{X}(\overline{Z}.1) + X(\overline{Z}.1) \qquad (\overline{Y} + Y = 1)$$

$$= \overline{X}\,\overline{Z} + X\overline{Z}$$

$$= \overline{Z}(\overline{X} + X)$$

$$= \overline{Z}.1 \qquad (\overline{X} + X = 1)$$

$$= \overline{Z}$$

Continued

BACK

Example 3: Minimise $AB+\overline{AC}+A\overline{B}C(AB+C)$

Solution : $AB+\overline{AC}+A\overline{B}C(AB+C)=AB+\overline{AC}+A\overline{B}CAB+A\overline{B}CC$

$= AB + \overline{AC} + AAB\overline{B}C + A\overline{B}CC$

$= AB + \overline{AC} + 0 + A\overline{B}CC$ *(putting $B\overline{B}=0$)*

$= AB + \overline{AC} + A\overline{B}.C$ *(putting C.C =C)*

$= AB + \overline{A} + \overline{C} + A\overline{B}C$ *(putting $\overline{AC}=\overline{A}+\overline{C}$ Demorgan's 2nd theorem)*

$= \overline{A} + AB + \overline{C} + A\overline{B}C$ *(rearranging the terms)*

$= \overline{A} + B + \overline{C} + A\overline{B}C$ *(putting $\overline{A}+AB = A+B$ because $X+\overline{X}Y = X+Y$)*

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

(putting $B+\overline{B}AC = B+AC$ because $X+\overline{X}Y = X+Y$)

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

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

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

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

$= 1 + B + \overline{C}$ *(as 1 + X = 1 i.e., anything added to 1 results in 1)*

$= 1$

BACK

2. Simplification Using Karnaugh Maps

- ✓ *Karnaugh map or K-map* is a graphical display of the fundamental products in a truth table.
- ✓ Karnaugh map is nothing but a rectangle made up of certain number of squares, each square representing a *Maxterm* or *Minterm*.
- ✓ These maps are sometimes also called *Veitch diagrams*.

(a) Sum-of-Products Reduction using Karnaugh Map

- ✓ In *S-O-P* reduction each square of K-map represents a minterm of the given function.
- ✓ For a function of n variables, there would be a map $\mathbf{2^n}$ squares, each representing a minterm.
- ✓ Given a K-map, for *S-O-P* reduction the map is filled in by placing 1s in squares whose minterms lead to a 1 output.

    See 2, 3, 4 variable K-maps for S-O-P reduction in next slide.

- ✓ Note thar in every square a subscripted number is written which denotes that this square corresponds to that number's minterm.
- ✓ For example, in 3 variable map $\overline{X}Y\overline{Z}$ box has been given number 2 which means this square corresponds to m_2.Similarly, box number 7 means it corresponds to m_7 and so on.
- ✓ Note the numbering scheme here, it is 0, 1, 3, 2 then 4, 5, 7, 6 and so on.
- ✓ Squares are marked using this scheme while making a K-map.

Continued

BACK

$X \backslash Y$	[0]$\overline{Y}$	[1]Y
[0]$\overline{X}$	$\overline{X}\,\overline{Y}$ 0	$\overline{X}Y$ 1
[1]X	$X\overline{Y}$ 2	XY 3

(*a*)`

$X \backslash Y$	[0]$\overline{Y}$	[1]Y
[0]$\overline{X}$	0	1
[1]X	2	3

(*b*)

2 – variable K-map representation minterms.

$X \backslash YZ$	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[0]$\overline{X}$	$\overline{X}\,\overline{Y}\,\overline{Z}$ 0	$\overline{X}\,\overline{Y}Z$ 1	$\overline{X}YZ$ 3	$\overline{X}Y\overline{Z}$ 2
[1]X	$X\overline{Y}\,\overline{Z}$ 4	$X\overline{Y}Z$ 5	XYZ 7	$XY\overline{Z}$ 6

(*a*)`

$X \backslash YZ$	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[0]$\overline{X}$	0	1	3	2
[1]X	4	5	7	6

(*b*)

3 – variable K-map representation minterms.

Continued

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[11] YZ	[10] $Y\overline{Z}$
[00] $\overline{W}\,\overline{X}$	$\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ 0	$\overline{W}\,\overline{X}\,\overline{Y}Z$ 1	$\overline{W}\,\overline{X}YZ$ 3	$\overline{W}\,\overline{X}Y\overline{Z}$ 2
[01] $\overline{W}X$	$\overline{W}X\overline{Y}\,\overline{Z}$ 4	$\overline{W}X\overline{Y}Z$ 5	$\overline{W}XYZ$ 7	$\overline{W}XY\overline{Z}$ 6
[11] WX	$WX\overline{Y}\,\overline{Z}$ 12	$WX\overline{Y}Z$ 13	$WXYZ$ 15	$WXY\overline{Z}$ 14
[10] $W\overline{X}$	$W\overline{X}\,\overline{Y}\,\overline{Z}$ 8	$W\overline{X}\,\overline{Y}Z$ 9	$W\overline{X}YZ$ 11	$W\overline{X}Y\overline{Z}$ 10

(*a*)`

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[11] YZ	[10] $Y\overline{Z}$
[00] $\overline{W}\,\overline{X}$	0	1	3	2
[01] $\overline{W}X$	4	5	7	6
[11] WX	12	13	15	14
[10] $W\overline{X}$	8	9	11	10

(*b*)

4 – variable K-map representation minterms.

- ✓ See the binary numbers at the top of K-map which do not follow binary progression, instead they differ by only one place when moving from lest to right : 00, 01, 11, 10.
- ✓ It is done so that only one variable changes from complemented to uncomplemented form or vice versa. See $\overline{A}\,\overline{B}, \overline{A}B, AB, A\overline{B}$.
- ✓ This binary code 00, 01, 11, 10 is called ***Gray Code***.
- ✓ *Gray Code* is the binary code in which each successive number differs only in one place.

Continued

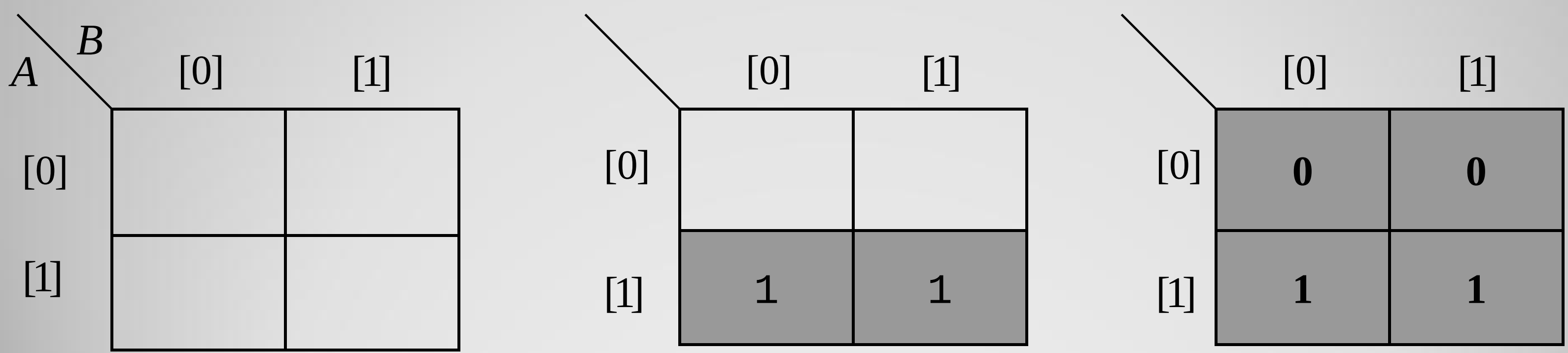
BACK

How to map in K-map ?

Let’s understand this by taking 2-variable map as following :

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

- ✓ Canonical *S-O-P* expression for this table is $F = AB + AB$ or $F = \sum(2,3)$.
- ✓ To map this function first draw an empty 2-variable K-map as shown in Fig. (a).
- ✓ Now for minterm m2 and m3 the output is 1 in truth table so mark 1 in th squares numbered as 2 and 3.
- ✓ Now K-map will look like Fig. (b).
- ✓ After entering 1’s for all 1 outputs, enter 0’s in the rest of the squares.
- ✓ Now K-map will look like Fig. (b).
- ✓ Same method is used for mapping 3-variable and 4-variable maps.



(*a*) (*b*) (*c*)

Continued

BACK

How to reduce Boolean expression in S-O-P form using K-map ?

- ✓ First mark pairs, quads and octets.
- ✓ To reduce an expression, adjacent 1's are encircled.
- ✓ If 2 adjacent 1's are encircled, it makes a pair; if 4 adjacent 1's are encircled, it makes a quad; if 2 adjacent 8's are encircled, it makes a octet.
- ✓ While encircling group of 1's, firstly search for octet and mark them, then fir quads and lastly go for pairs because a bigger group removes more variables and making the resultant expression simpler.

Reduction of pair :

- ✓ In the following K-map, after mapping a given function *F(W, X, Y, Z)* two pairs have been marked.
- ✓ Pair-1 is m_0 + m_4 and Pair-2 is m_4 + m_{15}.
- ✓ Pair-1 is vertical pair, in which *X* is changing its state from $\overline{X}$ to *X* as m_0 is $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ and m_4 is $\overline{W}X\overline{Y}\,\overline{Z}$ so the *X* can be removed.

WX \ YZ	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[10]$Y\overline{Z}$	[11]YZ
[00]$\overline{W}\,\overline{X}$	1 (0)	0 (1)	0 (3)	0 (2)
[01]$\overline{W}X$	1 (4)	0 (5)	0 (7)	0 (6)
[11]WX	0 (12)	0 (13)	1 (15)	1 (14)
[10]$W\overline{X}$	0 (8)	0 (9)	0 (11)	0 (10)

Pairs in a given K-map.

Pair Reduction Rule :

- ✓ Remove the variable which changes its state from complemented to uncomplemented or vice versa.
- ✓ Pair removes one variable only.
- ✓ Reduced expression for Pair-1 is $\overline{W}\,\overline{Y}\,\overline{Z}$ as $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$ (m_0) changes to $\overline{W}X\overline{Y}\,\overline{Z}$ (m_4).
- ✓ Prove algebraically as follows :

$$\begin{aligned}\text{Pair-1} = m_0 + m_4 &= \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{W}X\overline{Y}\,\overline{Z}\\ &= \overline{W}\,\overline{Y}\,\overline{Z}(\overline{X} + X)\\ &= \overline{W}\,\overline{Y}\,\overline{Z}.1 \qquad (\overline{X} + X = 1)\\ &= \overline{W}\,\overline{Y}\,\overline{Z}\end{aligned}$$

- ✓ Similarly, reduced expression for Pair-2(m14 + m15) will be WXY as WXYZ(m14) changes to (m15).
- ✓ Z will be removed as it is changing its state from $\overline{Z}$ to Z.

Continued

BACK

Reduction of quad :

- ✓ Suppose we are given with the K-map shown here in which two quads have been marked.
- ✓ Quad-1 is m0 + m4 + m12 + m8 and Quad-2 is *m7* + *m6* + *m15* + *m14* .In quad-1, two variables change their states *i.e.*, W and X are changing their states, so these two variables will be removed.

Quad Reduction Rule :

- ✓ Remove the variable which changes their states.
- ✓ Removes two variable.
- ✓ Reduced expression for Quad-1 is as *W* and *X* are removed.
- ✓ In Quad-2, horizontally moving, variable Z is removed as (*m7*) changes to (*m6*) and vertically moving, variable *W* is removed as (*m7*) changes to *WXYZ*. *So*, *r*educed expression for quad-2 is *XY*.

Reduction of an octet :

✓Suppose we have K-map with an octet marked as shown here in which moving horizontally two variables Y and Z are removed and moving vertically one variable X is removed.

✓So reduced expression for the octet is W only.

Octet reduction rule :

✓Remove the three variables which change their states.

✓An octet removes 3-variables.

WX \ YZ	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[00]$\overline{W}\,\overline{X}$	1 $_{0}$	0 $_{1}$	0 $_{3}$	0 $_{2}$
[01]$\overline{W}X$	1 $_{4}$	0 $_{5}$	1 $_{7}$	1 $_{6}$
[11]WX	1 $_{12}$	0 $_{13}$	1 $_{15}$	1 $_{14}$
[10]$W\overline{X}$	1 $_{8}$	0 $_{9}$	0 $_{11}$	0 $_{10}$

Quads in a given K-map.

WX \ YZ	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[00]$\overline{W}\,\overline{X}$	0 $_{0}$	0 $_{1}$	0 $_{3}$	0 $_{2}$
[01]$\overline{W}X$	0 $_{4}$	0 $_{5}$	0 $_{7}$	0 $_{6}$
[11]WX	1 $_{12}$	1 $_{13}$	1 $_{15}$	1 $_{14}$
[10]$W\overline{X}$	1 $_{8}$	1 $_{9}$	1 $_{11}$	1 $_{10}$

Octets in a given K-map.

Continued

Map Rolling :

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[10] $Y\overline{Z}$	[11] YZ
[00] $\overline{W}\,\overline{X}$	0	1	1 3	2
[01] $\overline{W}X$	1 4	5	7	1 6
[11] WX	12	13	15	14
[10] $W\overline{X}$	8	9	1 11	10

(a)` **Pairs**

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[10] $Y\overline{Z}$	[11] YZ
[00] $\overline{W}\,\overline{X}$	0	1 1	1 3	2
[01] $\overline{W}X$	1 4	5	7	1 6
[11] WX	1 12	13	15	1 14
[10] $W\overline{X}$	8	1 9	1 11	10

(b) **Quads**

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[10] $Y\overline{Z}$	[11] YZ
[00] $\overline{W}\,\overline{X}$	1 0	1	1 3	1 2
[01] $\overline{W}X$	4	5	7	6
[11] WX	12	13	15	14
[10] $W\overline{X}$	1 8	9	1 11	1 10

(c) **quad**

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[11] YZ	[10] $Y\overline{Z}$
[00] $\overline{W}\,\overline{X}$	1 0	1 1	1 3	1 2
[01] $\overline{W}X$	4	5	7	6
[11] WX	12	13	15	14
[10] $W\overline{X}$	1 8	1 9	1 11	1 10

(d) **octet**

Continued

BACK

Map Rolling Continued :

- ✓ Map Rolling means roll the map i.e., consider the map as if its left edges are touching the right edges and top edges are touching bottom edges.
- ✓ This is the special property of K-map that its opposite edges squares and corner squares are considered continuous.
- ✓ As in opposite edges squares and in corner squares only one variable changes its state from complemented to uncomplemented state or vice versa.
- ✓ So, while marking the pairs, quards and octets, map must be rolled.

Overlapping Groups :

- ✓ Overlapping means same 1 can be encircled more than once.
- ✓ Suppose we have a K-map as shown here :
- ✓ Observe that 1 for m_7 has been encircled twice.
- ✓ Once for Pair-1 ($m_5 + m_7$) and again for Quad ($m_7 + m_6 + m_{15} + m_{14}$).
- ✓ Also 1 for m14 has been encircled twice. For the Quad and for pair-2($m_{14} + m_{10}$).
- ✓ Overlapping always leads to simpler expressions.
- ✓ Here, reduced expression for Pair –1 is $\overline{W}XZ$
  reduced expression for Quad is XY
  reduced expression for Pair –2 is $WY\overline{Z}$
- ✓ Thus final reduced expression for this map is $\overline{W}XY + XY + WY\overline{Z}$
- ✓ Thus reduced expression for entire K-map is sum of all reduced expressions in the very K-map.

WX \ YZ	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[00]$\overline{W}\,\overline{X}$	0	1	3	2
[01]$\overline{W}X$	4	1 5	1 7	1 6
[11]WX	12	13	1 15	1 14
[10]$W\overline{X}$	8	9	11	1 10

Overlapping Groups.

Continued

BACK

Reduntant Group :

$WX \backslash YZ$	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[00]$\overline{W}\,\overline{X}$	0	1	3	2
[01]$\overline{W}X$	1 1	1 5	7	6
[11]WX	12	1 13	1 15	14
[10]$W\overline{X}$	8	9	11	10

(a) K-map with redundant group.

$WX \backslash YZ$	[00]$\overline{Y}\,\overline{Z}$	[01]$\overline{Y}Z$	[11]YZ	[10]$Y\overline{Z}$
[00]$\overline{W}\,\overline{X}$	0	1	3	2
[01]$\overline{W}X$	1 4	1 5	7	6
[11]WX	12	1 13	1 15	14
[10]$W\overline{X}$	8	9	11	10

(a) K-map without redundant group.

- ✓**Reduntant Group** is a group whose all 1's are overlapped by other groups.
- ✓Removel of reduntant group leads to much simpler expression.
- ✓Fig. *(a)* has a reduntant group. There are three pairs : Pair-1 (m_4+m_5), Pair-2 (m_5+m_{13}), Pair-3 ($m_{13}+m_{15}$).
- ✓But Pair-2 is a reduntant group as its all 1's are marked by other groups.
- ✓With this reduntant group, the reduced expression will be $\overline{W}X\overline{Y} + X\overline{Y}Z + WXZ$.
- ✓For simpler expression, Reduntant Group must be removed.
- ✓After removing the reduntant group, we get the K-map shown in Fig. *(b)*.
- ✓The reduced expression, for K-map in Fig. *(b)*, will be $\overline{W}X\overline{Y} + WXZ$, which is much simpler expression.

Continued

BACK

Summary of all the rules for S-O-P reduction using K-map :

1. Prepare the truth table for given function.
2. Draw an empty K-map for the given function.
3. Map the given function by entering 1's for the output as 1 in the corresponding squares.
4. Enter 0's in all left out empty squares.
5. Encircle adjacent 1's in form of octets, quads and pairs. Do not forget to roll the map and overlap.
6. Remove reduntant group if any.
7. Write the reduced expression for all the group and OR(+) them.

Continued

BACK

Example: Reduce $F(w, x, y, z) = \sum (0, 2, 7, 8, 10, 15)$ using Karnaugh map.

Solution : $F(w, x, y, z) = \sum (0, 2, 7, 8, 10, 15)$

$= m_0 + m_2 + m_7 + m_8 + m_{10} + m_{15}$

Truth Table for the given function is as follows :

W	X	Y	Z	F
0	0	0	0	1
0	0	0	1	
0	0	1	0	1
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	1
1	0	0	0	1
1	0	0	1	
1	0	1	0	1
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	1

$m_0 = 0000 = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$

$m_2 = 0010 = \overline{W}\,\overline{X}YZ$̄

$m_7 = 0111 = \overline{W}XYZ$

$m_8 = 1000 = W\overline{X}\,\overline{Y}\,\overline{Z}$

$m_{10} = 1010 = W\overline{X}Y\overline{Z}$

$m_{15} = 1111 = WXYZ$

Mapping the given function in a K-map, we get

WX \ YZ	[00] $\overline{Y}\,\overline{Z}$	[01] $\overline{Y}Z$	[11] YZ	[10] $Y\overline{Z}$
[00] $\overline{W}\,\overline{X}$	1 0	0 1	0 3	1 2
[01] $\overline{W}X$	0 4	0 5	1 7	0 6
[11] WX	0 12	0 13	1 15	0 14
[10] $W\overline{X}$	1 8	0 9	0 11	1 10

✓In above K-map two groups have been marked, one Pair and one Quad.

✓Pair is $m_7 + m_{15}$ and Quad is $m_0 + m_2 + m_8 + m_{20}$

✓Reduced expression for pair $(m_7 + m_{15})$ is XYZ as W is removed.

✓Reduced expression for quad $(m_0 + m_2 + m_8 + m_{10})$ is $\overline{X}\,\overline{Z}$ as for horizontal corners Y is removed and for vertical corners W is removed.

✓The final reduced expression is $XYZ + \overline{X}\,\overline{Z}$.

BACK

(b) Products-of-Sum Reduction using Karnaugh Map

- ✓ In *P-O-S* reduction each square of K-map represents a Maxterm of the given function.
- ✓ Karnaugh map is just the same as that of used in *S-O-P* reduction.
- ✓ For a function of n variables, there would be a map $\mathbf{2^n}$ squares, each representing a Maxterm.
- ✓ For P-O-S reduction map is filled by placing 0's in squares whose Maxterm lead to output 0.
- ✓ The numbers in the squares represents Maxterm subscripts.
- ✓ Box with number 1 represents M1, number 6 box represents M6, and so on.
- ✓ The box numbering scheme is the same as S-O-P i.e., 0, 1, 3, 2; 4, 5, 7, 6; 12, 13, 15, 14; 8, 9, 11, 10.
- ✓ One more similarity in S-O-P K-map and P-O-S K-map is that they are binary progression in Gray code only.
- ✓ One major difference is that in P-O-S K-map, complemented letters represent 1's and uncomplemented letters represent 0's whereas it is just the opposite in s-O-P k-map.
- ✓ Following are 2, 3, 4 variable K-map for P-O-S reduction :

$X \backslash Y$	[0]Y	[1]$\overline{Y}$
[0]X	0	1
[1]$\overline{X}$	2	3

(*a*)`

$X \backslash Y$	[0]$\overline{Y}$	[1]Y
[0]X	$(X+Y)$ 0	$(X+\overline{Y})$ 1
[1]$\overline{X}$	$(\overline{X}+Y)$ 2	$(\overline{X}+\overline{Y})$ 3

(*b*)

2 – variable K-map representation minterms.

Continued

BACK

X \ YZ	$[00] Y+Z$	$[01] Y+\overline{Z}$	$[11] \overline{Y}+\overline{Z}$	$[10] \overline{Y}+Z$
$[0] X$	0	1	3	2
$[1] \overline{X}$	4	5	7	6

(c)`

X \ YZ	$[00] Y+Z$	$[01] Y+\overline{Z}$	$[11] \overline{Y}+\overline{Z}$	$[10] \overline{Y}+Z$
$[0] X$	$X+Y+Z$ 0	$X+Y+\overline{Z}$ 1	$X+\overline{Y}+\overline{Z}$ 3	$X+\overline{Y}+Z$ 2
$[1] \overline{X}$	$\overline{X}+Y+Z$ 4	$\overline{X}+Y+\overline{Z}$ 5	$\overline{X}+\overline{Y}+\overline{Z}$ 7	$\overline{X}+\overline{Y}+Z$ 6

(d)

3 – variable K-map representation minterms.

WX \ YZ	[00]	[01]	[11]	[10]
[00]	0	1	3	2
[01]	4	5	7	6
[11]	12	13	15	14
[10]	8	9	11	10

(e)`

WX \ YZ	$[00] Y+Z$	$[01] Y+\overline{Z}$	$[11] \overline{Y}+\overline{Z}$	$[10] \overline{Y}+Z$
$[00] Y+Z$	$W+X+Y+Z$ 0	$W+X+Y+\overline{Z}$ 1	$W+X+\overline{Y}+\overline{Z}$ 3	$W+X+\overline{Y}+Z$ 2
$[01] Y+\overline{Z}$	$W+\overline{X}+Y+Z$ 4	$W+\overline{X}+Y+\overline{Z}$ 5	$W+\overline{X}+\overline{Y}+\overline{Z}$ 7	$W+\overline{X}+\overline{Y}+Z$ 6
$[11] \overline{Y}+\overline{Z}$	$\overline{W}+\overline{X}+Y+Z$ 12	$\overline{W}+\overline{X}+Y+\overline{Z}$ 13	$\overline{W}+\overline{X}+\overline{Y}+\overline{Z}$ 15	$\overline{W}+\overline{X}+\overline{Y}+Z$ 14
$[10] \overline{Y}+Z$	$\overline{W}+X+Y+Z$ 8	$\overline{W}+X+Y+\underline{Z}$ 9	$\overline{W}+X+\overline{Y}+\overline{Z}$ 11	$\overline{W}+X+\overline{Y}+Z$ 10

(f)

4 – variable K-map representation minterms

Continued

BACK

Summary of all the rules for P-O-S reduction using K-map :

1. Prepare the truth table for given function.
2. Draw an empty K-map for the given function.
3. Map the given function by entering 0's for the output as 0 in the corresponding squares.
4. Enter 1's in all left out empty squares.
5. Encircle adjacent 0's in form of octets, quads and pairs. Do not forget to roll the map and overlap.
6. Remove reduntant group if any.
7. Write the reduced expression for all the group and AND (.) them.

Continued

BACK

Minimization of Boolean Expression Continued :

Example : Reduce the following Karnaugh map in Product-of-sums form :

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

Solution :

First of all, we'll have to encircle all possible groups of adjacent 0's.Encircling we get the following K-map.

A \ BC	[00] $B+C$	[01] $B+\overline{C}$	[11] $\overline{B}+\overline{C}$	[10] $\overline{B}+C$
[0] A	0 (0)	0 (1)	0 (3)	1 (2)
[1] $\overline{A}$	0 (4)	1 (5)	1 (7)	1 (6)

✓There are 3 pairs which are :

✓ Pair-1 = $M_0 \,.\, M_1$;

✓ Pair-2 = $M_0 \,.\, M_4$;

✓ Pair-3 = $M_1 \,.\, M_3$.

✓There is one reduntant group which is Pair-1.

✓By removing this Pair-1, we have only two groups now.

✓Reduced *P-O-S* expression for Pair-2 is $(B + C)$, as while moving across Pair-2, A changes its state from A to $\overline{A}$, thus A is removed.

✓Reduced *P-O-S* expression for Pair-3 is $(A+\overline{C})$, as while moving across Pair-3 B changes to $\overline{B}$, hence eliminated.

✓Final *P-O-S* expression will be $(B+C).(A+\overline{C})$

BACK

More About Logic Gates:

- ✓ There are some more logic gates which are NOR, NAND, XOR, XNOR gates.
- ✓ These gates are derived from three basic gates *i.e.*, AND, OR and NOT and are widely used in industry.
- ❑ NOR Gate
- ❑ NAND Gate
- ❑ XOR Gate
- ❑ XNOR Gate
- ❑ NAND to NAND and NOR to NOR design

1. NOR Gate :

- ✓ Has two or more input signal but only one output signal.
- ✓ Is inverted OR gate.
- ✓ If all the inputs are 0, then output is 1.

2-input NOR gate

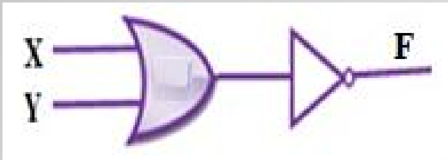
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

3-input NOR gate

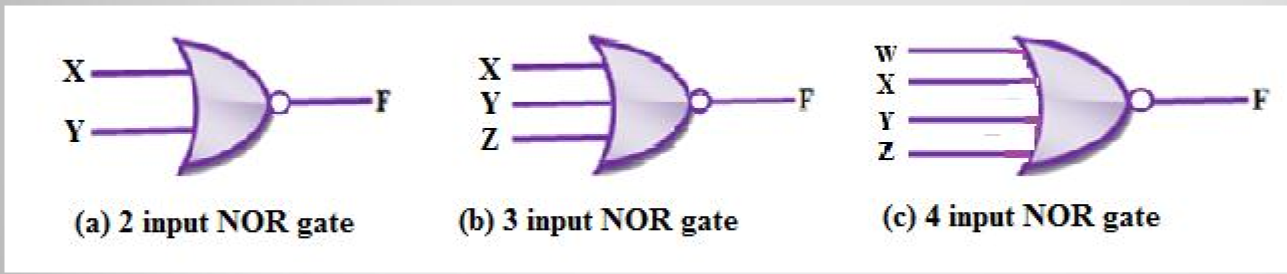
X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$$F = \overline{X + Y + Z}$$

The Logical meaning of NOR gate can be shown as follows : $F = \overline{X + Y}$



The symbol of 2, 3, 4 input NOR gates are given below :



BACK

2. NAND Gate :

- ✓ Has two or more input signal but only one output signal.
- ✓ Is inverted AND gate.
- ✓ If all the inputs are 1, then output is 0.

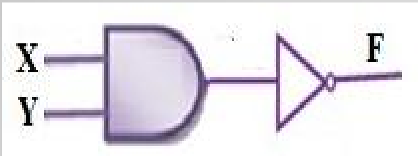
2-input NAND gate

X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

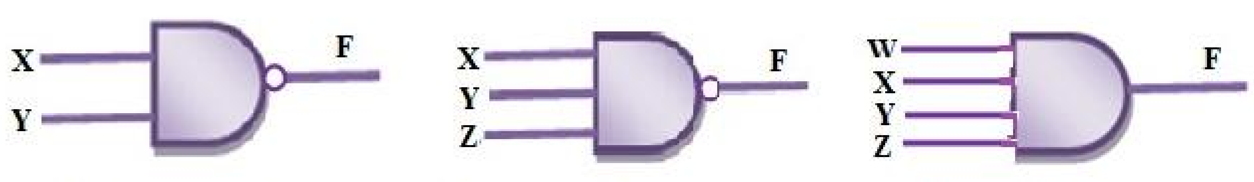
3-input NAND gate

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

The Logical meaning of NAND gate can be shown as follows :



The symbol of 2, 3, 4 input NAND gates are given below :



(a) 2-input NAND gate (b) 3-input NAND gate (c) 4-input NAND gate

BACK

3. XOR Gate :

- ✓ Has two or more input signal but only one output signal.
- ✓ Produces output 1 for only those input combination that have odd number 1's.
- ✓ $\oplus$ sign stands for XOR operation.
- ✓ Thus A XOR B can be written as $A \oplus B$
- ✓ XOR addition can be summarized as follows :

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0,$$

2-input NOR gate

No. of 1's even/odd	X	Y	F
Even	0	0	0
Odd	0	1	1
Odd	1	0	1
Even	1	1	0

3-input NOR gate

No of 1's	X	Y	Z	F
Even	0	0	0	0
Odd	0	0	1	1
Odd	0	1	0	1
Even	0	1	1	0
Odd	1	0	0	1
Even	1	0	1	0
Even	1	1	0	0
Odd	1	1	1	1

The symbol of 2, 3, 4 input XOR gates are given below :

(a) 2-input XOR gate (b) 3-input XOR gate (c) 4-input XOR gate

4. XNOR Gate :

- ✓ Logically equivalent to an inverted XOR. i.e., XOR followed by a NOT gate(inventor).
- ✓ Produces 1 output when the input combination has even number of 1's.
- ✓ The bubble(small circle), on the outputs of NAND, NOR, XNOR gates represents complementation.

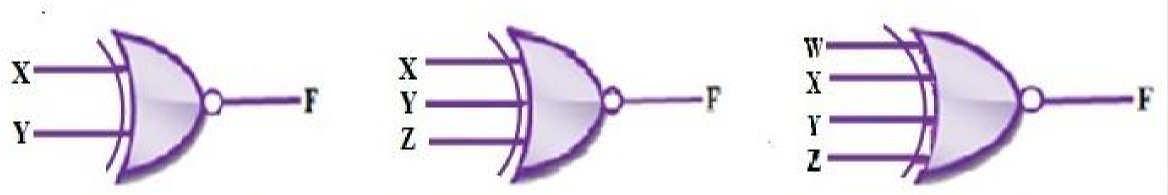
2-input NOR gate

No. of 1's even/odd	X	Y	F
Even	0	0	1
Odd	0	1	0
Odd	1	0	0
Even	1	1	1

3-input NOR gate

No of 1's	X	Y	Z	F
Even	0	0	0	1
Odd	0	0	1	0
Odd	0	1	0	0
Even	0	1	1	1
Odd	1	0	0	0
Even	1	0	1	1
Even	1	1	0	1
Odd	1	1	1	0

The symbol of 2, 3, 4 input XNOR gates are given below :



(a) 2-input XNOR gate (b) 3-input XNOR gate (c) 4-input XNOR gate

Continued

BACK

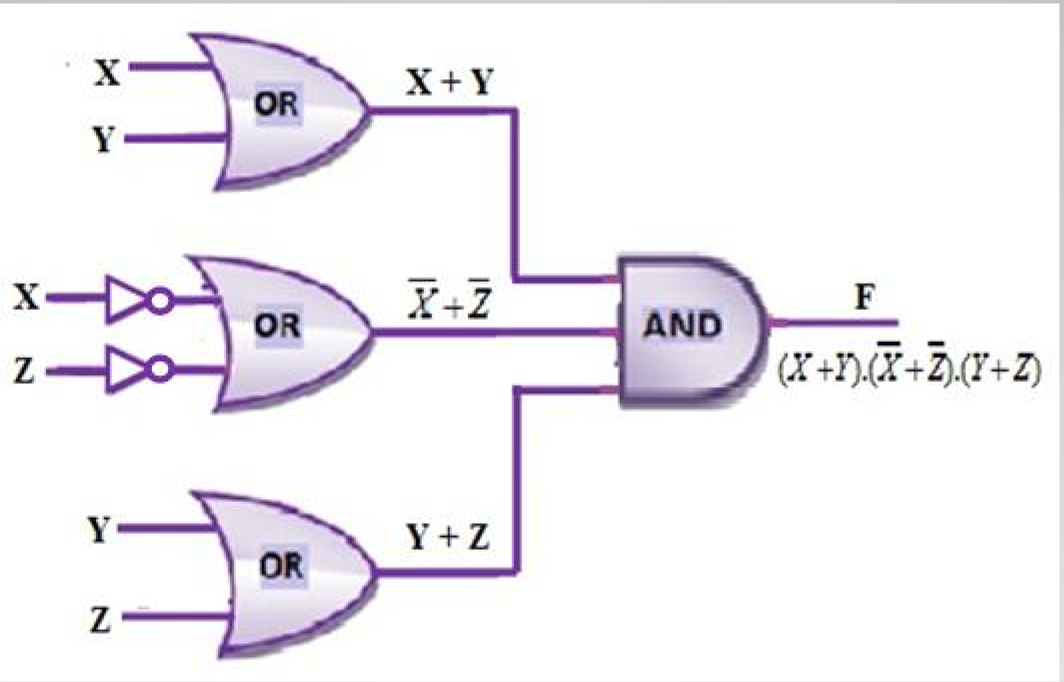
Example : *Draw the diagram of digital circuit for the function :*

$$F(X,Y,Z) = (X+Y).(\overline{X}+\overline{Z}).(Y+Z).$$

Solution : Above expression can also be written as :

F(X, Y, Z) = (X OR Y) AND ((NOT X) OR (NOT Z) AND (Y OR Z)

Thus circuit diagram will be



BACK

NAND to NAND and NOR to NOR design

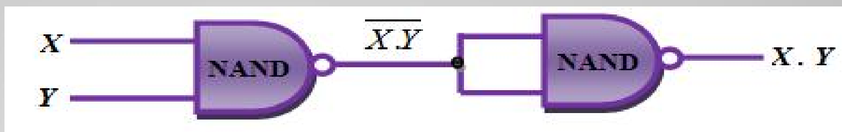
- ✓ NAND and NOR gates are more popular for design circuits as these are less expensive and easier to design.
- ✓ Other switching functions (AND , OR) can easily be implemented using NAND/NOR gates.
- ✓ Also referred to as *Universal Gates.*

NAND-to-NAND logic

AND and OR operation from NAND gates are shown below :

AND *operation*

AND operation using NAND is



AND operation using NAND-NAND logic

X .Y = (X NAND Y) NAND (X NAND Y)

Proof. X NAND Y $= \overline{X.Y}$

$= \overline{X} + \overline{Y}$ (*DeMorgan's Second Theorem*)

(X NAND Y) NAND (X NAND Y)

$= (\overline{X}+\overline{Y}) NAND (\overline{X}+\overline{Y})$

$= \overline{(\overline{X}+\overline{Y}).(\overline{X}+\overline{Y})}$

$= \overline{(\overline{X}+\overline{Y})} + \overline{(\overline{X}+\overline{Y})}$ (*DeMorgan's Second Theorem*)

$= \overline{\overline{X}}.\overline{\overline{Y}} + \overline{\overline{X}}.\overline{\overline{Y}}$ (*DeMorgan's Second Theorem*)

$= X.Y + X.Y$ $\overline{\overline{X}} = X$

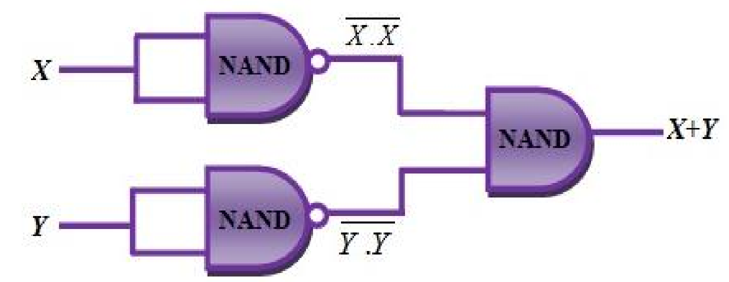
$= XY$ $(X + X = X)$

Continued

BACK

OR operation

OR operation using NAND is



OR operation NAND to NAND logic

X +Y = (X NAND X) NAND (Y NAND Y)

Proof. X NAND Y $= \overline{X\,.X}$

$= \overline{X} + \overline{X}$ (*DeMorgan's Second Theorem*)

$= \overline{X}$ $(X + X = X)$

Similarly, Y NAND Y $= \overline{Y}$

Therefore, (X NAND X) NAND (Y NAND Y)

$= \overline{X}\ NAND\ \overline{Y}$

$= \overline{\overline{X}.\overline{Y}}$

$= \overline{\overline{X}} + \overline{\overline{Y}}$ (*DeMorgan's Second Theorem*)

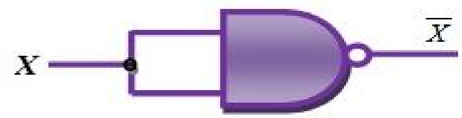
$= X + Y$ $(\ \overline{\overline{X}} = X\ ,\ \overline{\overline{Y}} = Y\ )$

Continued

BACK

NOT operation

NOT operation using NAND is



NOT operation using NAND gate

NOT X = X NAND X

Proof. X NAND X $= \overline{X.X} = \overline{X}$ $(\because X.X = X)$

NAND-to-NAND logic is best suited for Boolean expression in *Sum-of-Products* form.

Design rule for NAND-TO-NAND logic Network (only for 2 level circuit)

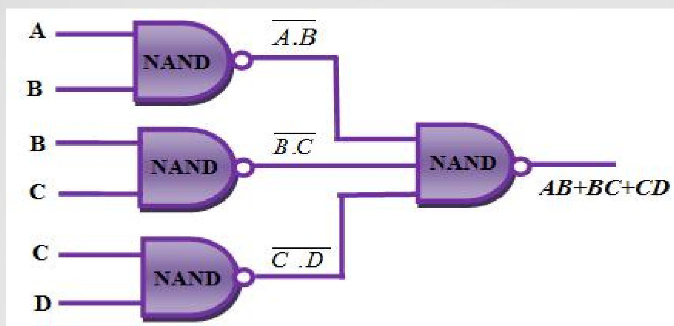
1. Derive simplified sum-of-products expression.
2. Draw a circuit diagram using AND, OR gates.
3. Replace all basic gates (AND, OR, NOT) with NAND gates.

Example : Draw the diagram of digital circuit for

F(a, b, c) = AB + BC + CD using NAND-to-NAND logic.

Solution: F(a, b, c) = AB + BC + CD

= (A NAND B) NAND (B NAND C) NAND (C NAND D)



Continued

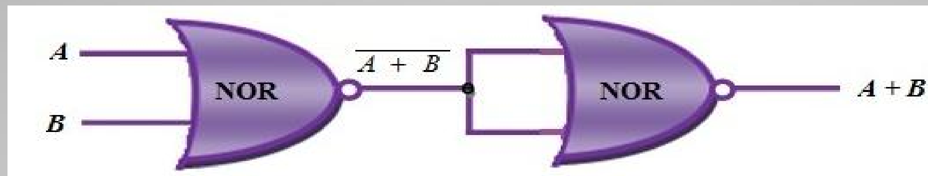
BACK

NOR-to-NOR logic

AND and OR operation can be implemented in NOR-to-NOR form as shown bellow :

OR operation

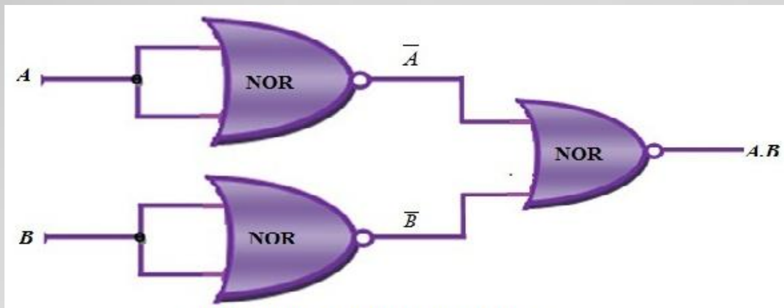
A + B = (A NOR B) NOR (A NOR B)



OR operation using NOR to NOR logic.

AND operation

A . B = (A NOR A) NOR (B NOR B)

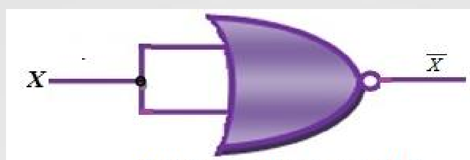


AND operation using NOR to NOR logic.

NOR-to-NOR logic Continued :

NOT operation

NOT X = X NOR X



NOR operation using NOR.

NOR-to-NOR logic is best suited for Boolean expression in *Product-of-Sums form.*

Continued

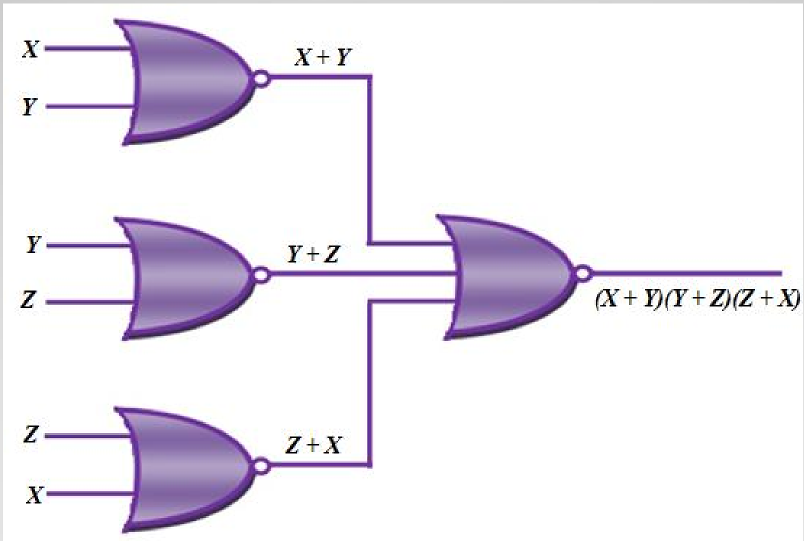
BACK

Design rule for NOR-to-NOR logic Network (only for 2 level circuit)

1. Derive simplified Product-of-Sums form of the expression.
2. Draw a circuit diagram using OR, AND gates.
3. Finally substitute NOR gates for OR,NOT and AND gates.

Example : Represent (X + Y) (Y + Z) (Z + X) in NOR-to-NOR form.

Solution: (X + Y) (Y + Z) (Z + X) = (X NOR Y) NOR (Y NOR Z) NOR (Z NOR X)



BACK