

Boolean Algebra Part-2



Boolean Algebra Part-2

Content:

1. Axioms Of B.A.
2. Some Useful Theorems
3. Unique Features
4. Boolean Sub algebra
5. Basic Operations
6. Logic Gates
7. SOP and POS

A non-empty set B with two binary operations $+$ and $.$, a unary operation $'$, and two distinct elements 0 and 1 is called a Boolean algebra denoted by $(B, +, ., ', 0, 1)$ if and only if the following properties are satisfied.

Axioms of Boolean algebra

If $a, b, c \in B$, then

1. Commutative laws:

- (a) $a + b = b + a$
- (b) $a.b = b.a$

2. Distributive laws:

- (a) $a + b = b + a$
- (b) $a.(b + c) = (a.b) + (a.c)$

3. identity laws:

- (a) $a + a' = 1$
- (b) $a.a' = 0$

4. complement laws:



Gradeup UGC NET **Super Subscription**
Access to all Structured Courses & Test Series



(a) $a + a' = 1$

(b) $a.a' = 0$

Basic theorems

Let $a, b, c \in B$, then

1. idempotent laws:

(a) $a + a = a$

(b) $a.a = a$

2. Boundedness laws

(a) $a + 1 = 1$

(b) $a.0 = 0$

3. Absorption laws:

(a) $a + (a.b) = a$

(b) $a.(a + b) = a$

4. Associative laws:

(a) $(a + b) + c = a + (b + c)$

(b) $(a.b).c = a.(b.c)$

5. Uniqueness of complement:

$a + x = 1$ and $ax = 0$, then $x = a'$

6. Involution law:

a. $a' = a$

b. $0' = 1$



c. $1' = 0$

7. De-Morgan's laws:

(a) $(a + b)' = a' \cdot b'$

(b) $(a \cdot b) = a' + b'$

Note: where there is no danger of confusion, the symbol can be deleted just as in writing algebraic products i.e., $a \cdot b$ as ab .

Algebraic proofs of some laws are given and the rest are left as exercises for the readers.

Each of the laws can also be proved using a truth table.

Theorem : $a + a = a$

Theorem: $a \cdot a = a$

Theorem: $a + 1 = 1$

Theorem : $a + (a \cdot b) = a$

Theorem: for each $a \in B$, a' is unique.

Theorem: The element 0 and 1 are unique

Theorem: (i) $(a+b)' = a'b'$ (ii) $(a \cdot b)' = a' + b'$

Unique Feature:

The striking unique features Boolean algebra , which distinguishes it from the other branches a algebra are:

(a) Boolean algebra has only a finite set of elements , namely 1 and 0 , which are usually called true and false , where as ordinary algebra deals with a set of an infinite number of elements. Thus all the variables and constant are allowed to have only two possible value 0 and 1. Hence it provides the operations and rules for working with the set $\{0,1\}$;

(b) Boolean algebra does not have operations equivalent to subtraction and division ;

(c) In ordinary algebra , there is no equivalent of the unary operation of bar or/over a variable known as complementing .



Boolean Sub algebra:

A non-empty subset A of a Boolean algebra $(B, +, \cdot, 0, 1)$ is called a Boolean sub algebra if A is a Boolean algebra under the same set binary and unary operations i.e. $a, b \in A \rightarrow a + b, a \cdot b, a' \in A$.

For example, consider a set $D_{30} = \{1, 2, 3, 5, 6, 10, 15, 30\}$, the divisor of 30 and let $+, \cdot$ And be defined on D_{30} as follows

$$a + b = \text{lcm}(a, b)$$

$$a \cdot b = \text{gcd}(a, b) \text{ and } a^1 = 30/a$$

then D_{30} is a boolean algebra and $A_1 = \{1, 5, 6, 30\}$ and $A_3 = \{1, 2, 3, 6\}$ are Boolean sub algebra of D_{30} since A_1, A_2 and A_3 are closed under $+, \cdot$. But $A_4 = \{1, 2, 3, 6\}$ is not a Boolean sub algebra because $6 = 30/6 = 5$ does not belong to A_4 . To show that A is a Boolean sub algebra it is not necessary to verify the closurenens with respect to all three operations. It is enough to show that A is closed under the set of operations $\{+, '\}$ or $\{\cdot, '\}$.

Basic operations

The basic operations used in Boolean algebra are logical addition, logical multiplication and complementation. These basic operations are called logical operations. Boolean 1 and 0 do not represent actual numbers. In the digital logic field several terms are used continuously with 0 and 1.

Logic 0	False	Off	Low	Open	No
Logic 1	True	On	High	Closed	Yes

Logical addition(OR operation)

Each variable in Boolean algebra has either two values: true or false (1 or 0), the operation can be defined as follows

$$0 + 0 = 0$$



$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

Where logical addition is denoted by + or by OR . if we write $C = A + B$, one can determine C by listing all possible combinations for A and B in a tabular form which is known as truth table of logical addition.

Input		Output
A	B	$C = A + B$
0	0	0
1	0	1
0	1	1
1	1	1

The symbol + does not have the normal meaning, but it is a logical addition and is referred to as OR operation. The equation $A + B = C$ can be read as A OR B equals C. this concept can be extended to any number of variables. To avoid ambiguity, a number of other symbols have been recommended as replacement for + sign.

Logical multiplication (AND operation)

The logical multiplication can be defined as follows

$$0.0 = 0$$

$$0.1 = 0$$

$$1.0 = 0$$

$$1.1 = 1$$



where logical multiplication is denoted by, or by AND. If we write $C = AB$, we can determine C by listing all possible combination of A and B. The truth table for logical multiplication of two variables.

Input		Output
A	B	$C = A \cdot B$
0	0	0
1	0	0
0	1	0
1	1	1

It is apparent from the table that the AND operation is exactly the same as ordinary multiplication. The equation $A \cdot B = C$ can be read as A AND B equals C.

Complementation

In Boolean algebra, an operation called complementation is used for which symbol is (bar over)' or ' and this can be defined as $0' = 1$ and $1' = 0$.

Here A' means the complement of A and read as NOT A. It is opposite to the logic value of $A=1$.

Input	Output
A	$B = A'$
0	1
1	0

Duality



Given a Boolean expression, the dual is formed by replacing AND with OR, OR with AND, 0 with 1, and 1 with 0. Variables and complements are left unchanged. This rule for forming the dual can be summarized as follows:

$$[f(x_1, x_2, \dots, x_n, 0, 1)]^D = f(x_1, x_2, \dots, x_n, 1, 0, \dots)$$

This important property of changing operations and Identity elements in Boolean algebra is known as duality principle. This principle ensures that if a theorem is proved based on the axioms of the Boolean algebra, then dual theorem obtained by interchanging + with and 0 with 1 automatically holds. Axioms and basic theorems are divided into two parts (a) and (b), one part may be obtained by other by using duality principle. For example. If $f = (a \cdot 1) \cdot (0 \cdot a')$ then $f^D = (a + 0) + (1 + a')$ which is obtained interchanging + and \cdot , and interchanging 0 and 1.

De- Morgan's Theorem

The complement of any Boolean expression can be found by successively applying the theorem referred to as De Morgan's theorem. The two theorems are

$$(x + y)' = x' \cdot y'$$

$$(xy)' = x' + y'$$

x	y	x'	y'	x + y	(x + y)'	x' y'	xy	(xy)'	x' + y'
0	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	0	0	1	0	0

These theorems can be extended to more than one variable. For instance, consider $(x + y + z)'$



$$(x + y + z)' = (A + z)'$$

$$= A' z'$$

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

$$= X' y' z'$$

Similarly, it can be proved $(xyz)' = x' + y' z'$

Proceeding similarly, we can extend the results to any number of variables.

The theorems are equally valid where x and y are expressions that contain more than one variable. **For example,**

$$(AB' + C)' = (AB')' C'$$

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

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

$$= A' C' + BC'$$

We can combine the two forms of De Morgan's theorem into a single rule which will enable us to complement an expression. To obtain the complement of a Boolean expression replace

- a) 0 with 1
- b) 1 with 0
- c) + with *
- d) . with +
- e) Each variable with its complement.

Which can be summarized as

$$[f(X_1, X_2, \dots, X_N, 0, 1, +)]' = f(X_1', X_2', \dots, X_N', 1, 0, \dots, +).$$

Logical Gate

A logic gate is an electronic circuit that operates on one or more input signals to produce an output signal. Gates are digital circuits and often called logic circuits. Gates are represented by symbols, the lines entering the symbol from left are inputs, and the line on the right is the output. Placing a small circle on an input or output complements the signal on that line.

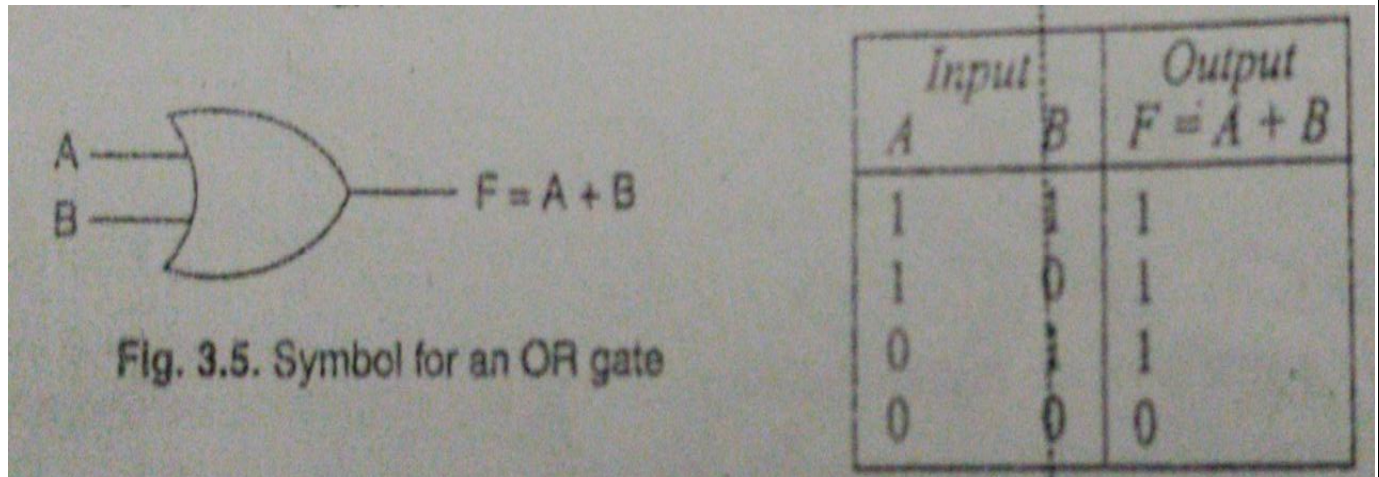
OR gate



Gradeup UGC NET **Super Subscription**
Access to all Structured Courses & Test Series

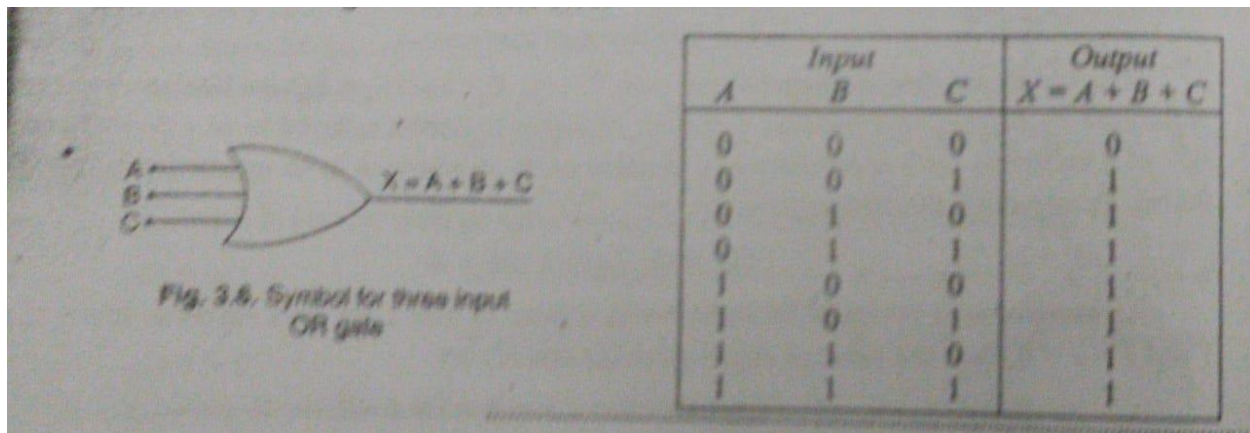


The OR gate is an electronic device which receives two or more inputs and produces an output equal to the OR sum of the inputs. It is the symbol for two input OR gate and its truth table. An input signal applied to gate has only two stable states either 1 (high) or 0 (low) and output is 1 (high) if at least one of the inputs is high. The output is 0 if all inputs are 0.



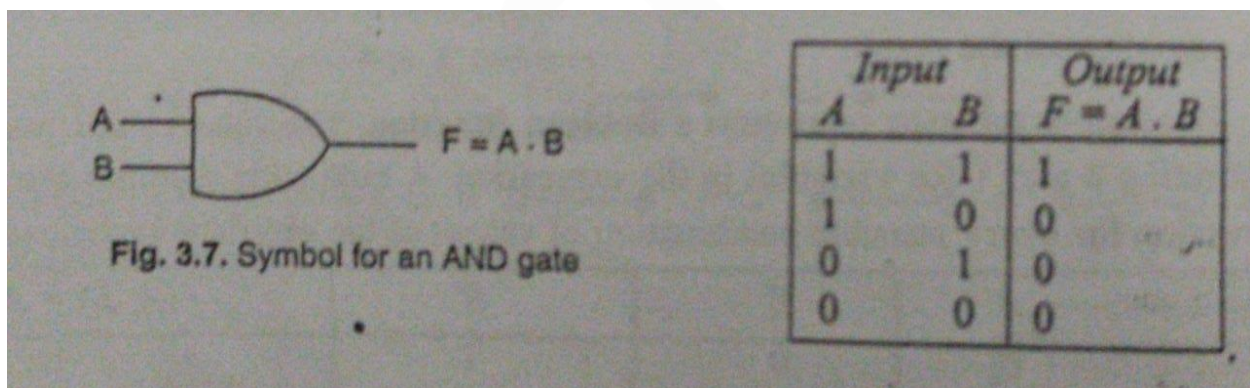
The same idea can be extended to more than two inputs. A three input OR gate and its truth table.





AND gate

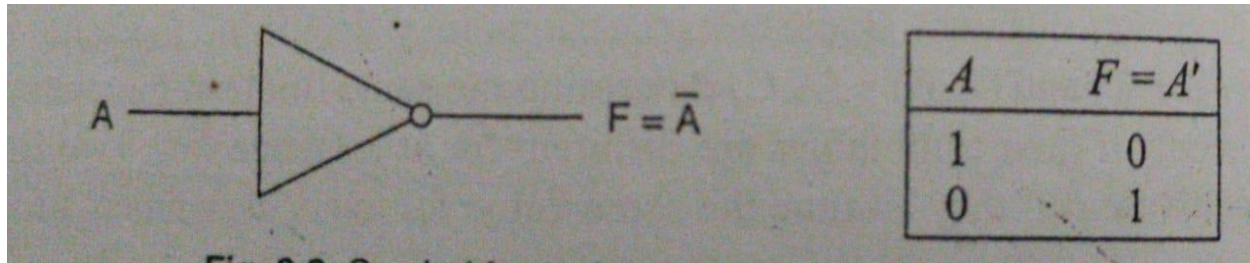
AND gate is an electronic device that accepts the values of two or more Boolean variables as input and produces their Boolean sum as output. Same operation holds good for an AND gate with more than the two inputs.



NOT gate

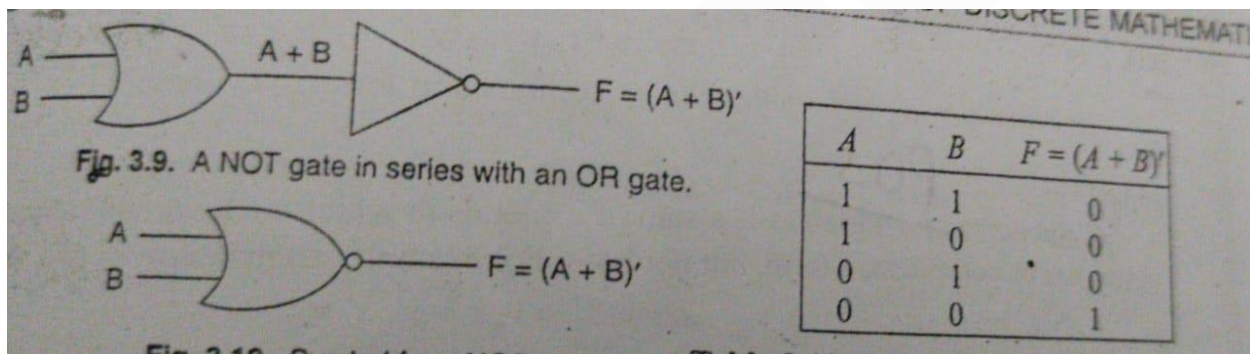
A NOT gate has only one input and one output. It is also called an inverter. Its output is the complement of the input.





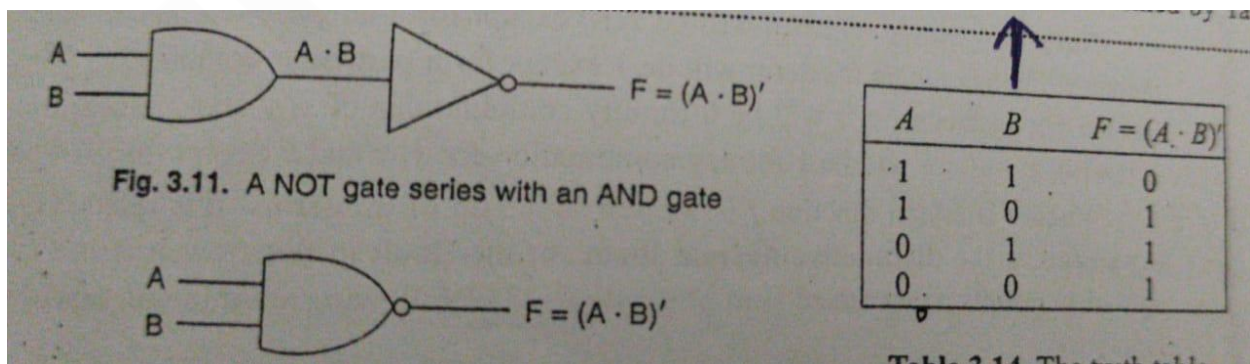
NOR gate

This gate is logically equivalent to a NOT gate with an OR gate. The NOR function is the complement of OR function.



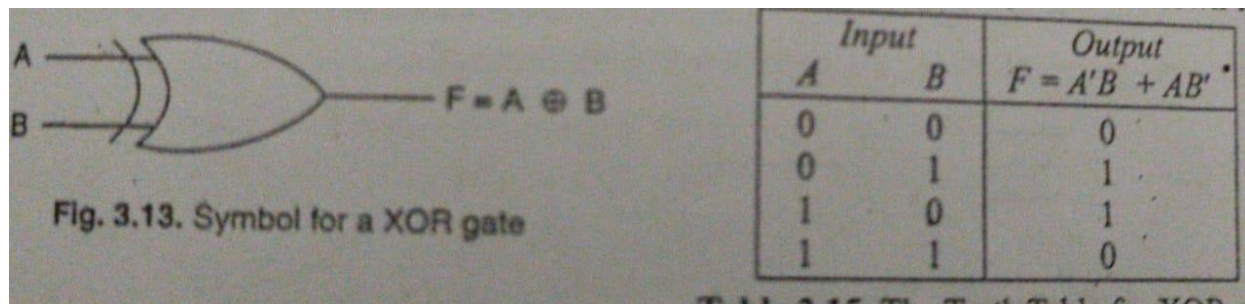
NAND gate

This gate is logically equivalent to a NOT gate in series with an AND gate.



Exclusive OR gate

The full name of OR gate is the inclusive OR gate. The exclusive OR gate is different. It gives an output of 1 when either but not both inputs are 1. In Boolean algebra, the sign stands for XOR operation.



Exclusive – NOR(XNOR) gate

XNOR gate is equivalent to an Exclusive OR gate followed by an inverter. The output is 1 only when both inputs are either 0 or 1.

Input		Output
A	B	$F = AB + A'B'$
0	0	1
0	1	0
1	0	0
1	1	1

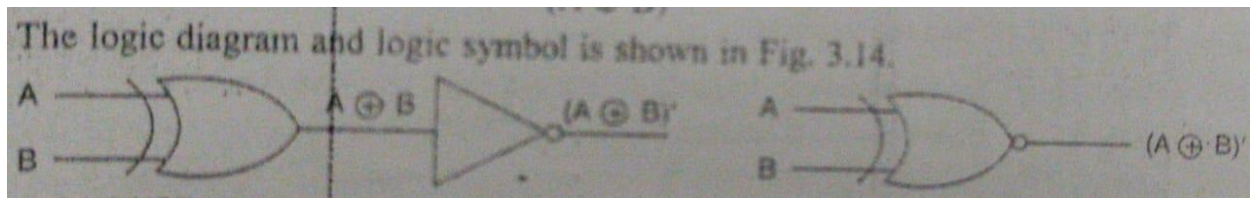
XNOR operation can be written as follows:

$$F = AB + A'B'$$

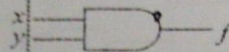

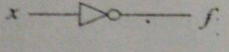
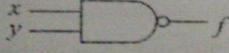



$$= A \text{ XNOR } B$$



$$= (A \oplus B)'$$



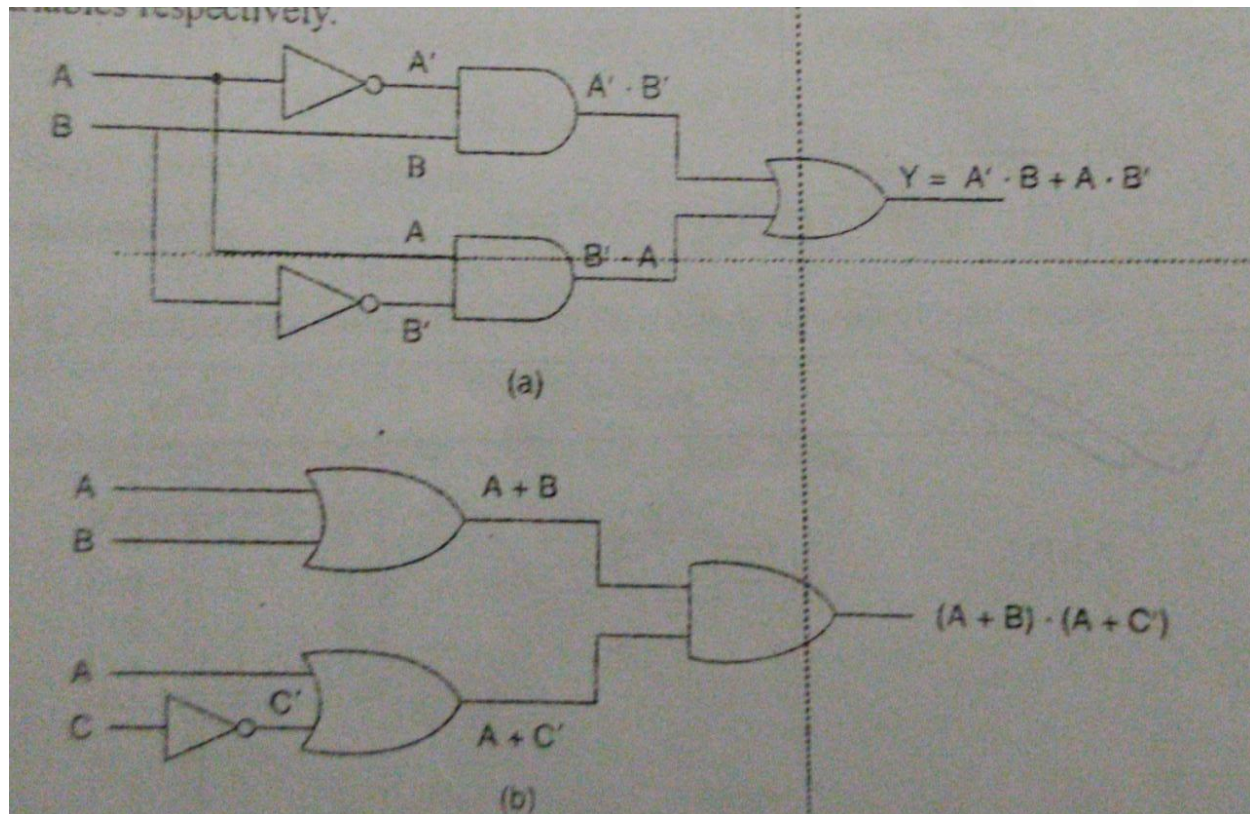
Logic Gates With truth Table :

Name	Symbol	Truth Table	Boolean Formula															
AND		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	f	0	0	0	0	1	0	1	0	0	1	1	1	$f = xy$
x	y	f																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	f	0	0	0	0	1	1	1	0	1	1	1	1	$f = x + y$
x	y	f																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT (inverter)		<table><tr><th>x</th><th>f</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	x	f	0	1	1	0	$f = x'$									
x	f																	
0	1																	
1	0																	
NAND		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	f	0	0	1	0	1	1	1	0	1	1	1	0	$f = (xy)'$
x	y	f																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	f	0	0	1	0	1	0	1	0	0	1	1	0	$f = (x + y)'$
x	y	f																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR (Exclusive-OR)		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	x	y	f	0	0	0	0	1	1	1	0	1	1	1	0	$f = x \oplus y$ $= x'y + xy'$
x	y	f																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR (Exclusive-NOR)		<table><tr><th>x</th><th>y</th><th>f</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	x	y	f	0	0	1	0	1	0	1	0	0	1	1	1	$f = (x \oplus y)'$ $= xy + x'y'$
x	y	f																
0	0	1																
0	1	0																
1	0	0																
1	1	1																



Interconnecting gates

The gates can be interconnected to form gating or log networks. The Boolean expression corresponding to given network, can be derived by systematically progressing from input to output on the gates. When gating network is formed, some gates may share inputs. One method is to indicate the inputs separately for each gate, The other method is to use branching that indicate all the gates that use a given input.



Sum of products and product of sums form

A Boolean expression E is said to be in a sum of products form if E is a sum of two or more products of variables (complemented or uncomplemented), none of which is included in another.

Some examples of this form are

1. $Xz' + x'yz' + xy'z$

2. $abc + ac + a'bc'$



3. $xy + x'yz' + y'z' + w$

Note: In a sum of product expression, one complementation sign can not cover more than one variable in a term (eg, we can not have $(xyz)'$ or $(ab)'c$)

A Boolean expression E is said to be in a product of sums if it consists of several sum terms logically multiplied. The variables may or may not be complemented. Here are some product of sum expression.

1. $(x + y)(x + y)$

2. $(x + y' + z)(x + z)$

3. $(a + b')(c' + d)f$

Normal form

Minterm

A minterm of variables is A product of a literals in which each variable appears exactly once in either true or complimented form, but not both For example, the list of all of the minterms of the two variables x and y are

$xy, x'y, xy', x'y'$

$xyz, xyz', xy'z, x'yz, xy'z', x'y'z, x'yz', x'y'z'$

in a similar way, n variables can be combined to form 2^n minterms.

Maxterm

A maxterm of n variable is a sum of n literals in which each variable appears exactly once in either true or complicated form, but not both. For example, all maxterm of two variables x and y are

$x + y, x' + y, x + y', x' + y'$

And all max terms of three variables x, y and z are

$x' + y' + z', x' + y' + z, x' + y + z', x + y' + z', x + y' + z, x + y + z', x' + y + z, x + y + z$

In a similar manner, n variables forming maxterms, with each variable being complemented or uncomplementented, provide 2^n possible combination.

Note: Any single minterm will be zero except for a single combination of the values of the literals while any single maxterm will be 1 except for a particular combination of the values of the literals. For instances, $xy'z$ will be 0 for any combinations of x, y and z except $x = 1, y$



= 0 and $z = 1$. Also, $x + y' + z$ will be 1 for any combination for x , y and z except for $x = 0$, $y = 1$ and $z = 0$.

When a Boolean function is written as a sum of minterms, it is referred to as a minterm expansion or the disjunctive normal form of the Boolean function. It is also called canonical sum of products or standard sum of products. The following are minterm expansions of Boolean functions

1. $f(x, y) = xy + x'y$

2. $f(x, y, z) = x'y'z + x'yz + xy'z' + xyz$

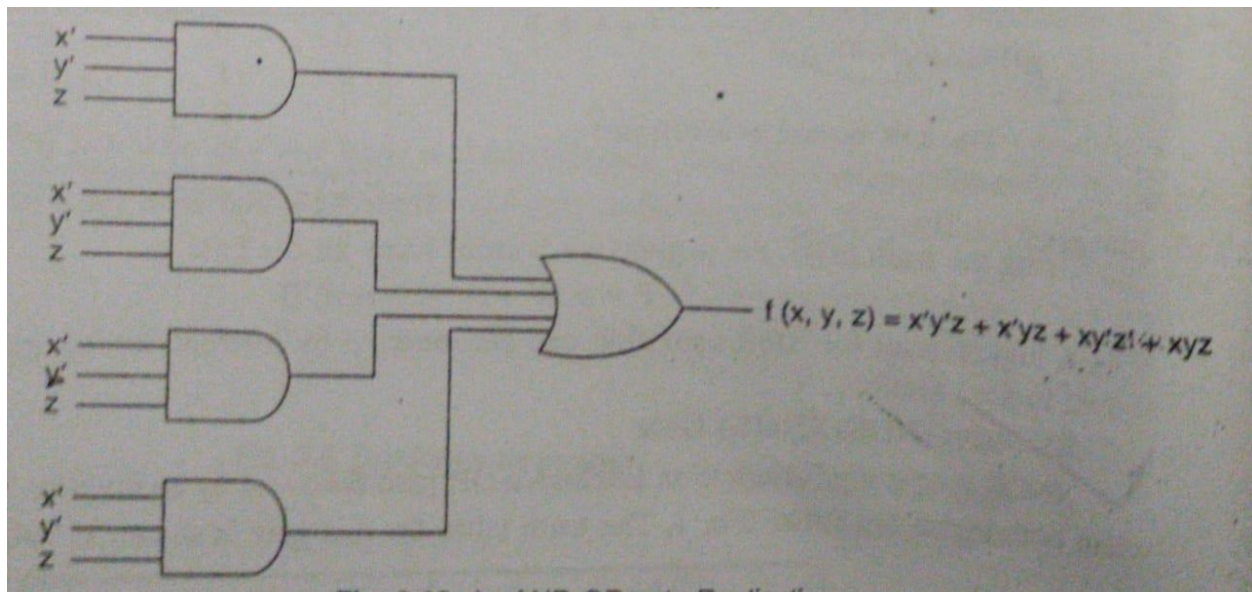
The canonical sum of products form can be implemented simply by an interconnection of gates. Note that each minterm is realised using single AND gate. The outputs of all the minterms are connected to the inputs of a single OR gate whose output is the desired function. Therefore, if there are n_1 minterms in the canonical sum of products, there will be $n_1 + 1$ gates used in its realisation.

When a Boolean function f is written as a product of maxterm, it is referred to as a maxterm expansion or the conjunctive normal form of the Boolean function. It is also called canonical product of sums and standard product of sums. The following are maxterm expansions of Boolean functions

1. $f(x, y) = (x + y)(x' + y')$

2. $f(x, y, z) = (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z)$





Actually either canonical form may use many more gates than is necessary. In later section we shall discuss a procedure for minimizing the number of gates required.

Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form.

A Boolean function when expressed as a sum of all 2^n minterms of n variables is called the complete minterm expansion or the complete disjunctive normal form. Similarly, a Boolean function when expressed as a product of all 2^n maxterms of n variables is called the complete maxterm expansion or complete conjunctive normal form.

Boolean function when expressed as a complete minterm expansion or complete maxterm expansion form is called complete canonical form

The complete minterm expansion in 'two variables x and y is
 $xy + xy' + x'y + x'y'$
 Which can be proved identically equal 1 as shown below

$$xy + x'y + xy' + x'y' = xy + xy' + x'y + x'y'$$

$$= x(y + y') + x'(y + y')$$

$$\begin{aligned} &= x.1 + x'.1 \\ &= x + x' \\ &= 1 \end{aligned}$$



In general the complete minterm expansion form in n variable is identically 1.

Expansion of a Boolean function as a canonical form

A Boolean function may be expressed to minterm or maxterm expansion either using a truth table or algebraically.

Form a given truth table

If a truth table to describe the behaviour of logic network is given, the corresponding logic expression in the minterm expansions and maxterm expansions can be obtained from the truth table by the method described below.

(a) Inspect the column corresponding to dependent variable from the top row a 4 pick the row with 1 entry.

(b) A term in the Boolean expression corresponding to this row is obtained ANDing all the independent variables in the truth table.

(c) Repeat steps (a) and (b) till all the entries in the dependent variable column is exhausted. Obtain a Boolean expression by ORing the terms corresponding to 1 entries of dependent variable.

The method is applied for obtaining minterm expansion of a Boolean function. The method for obtaining the maxterm expansion of a Boolean 1 similar, instead of picking the rows of 1 entry in dependent variable, the rows of 0 entry are to be picked up. The individual term is formed by taking the sum of independent variables. The expression is obtained by ANDing the individual terms. The following example clarifies this procedure.

Example: Consider the Following truth table and find the corresponding Min-term and Max-term:



Input			Output
X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Table 2.18

Solution. Corresponding to entry 1 of the output column in the first, second, third, fifth and sixth row the minterms are $X'Y'Z'$, $X'Y'Z$, $X'YZ'$, $XY'Z'$, $XY'Z$. Hence the minterm expansion of the Boolean function given in truth table is

$$F = X'Y'Z' + X'Y'Z + X'YZ' + XY'Z' + XY'Z$$

Corresponding to 0 entries of the output column in the fourth, seventh and eighth row the maxterms are $X + Y' + Z'$, $X' + Y' + Z$ and $X' + Y' + Z'$. Hence the maxterm expansion of the Boolean function given in truth table is

$$F = (X + Y' + Z')(X' + Y' + Z)(X' + Y' + Z')$$

Algebraic method

To obtain the minterm expression by algebraic method, first write the expression as a sum of products and then introduce the missing variable in each term by applying the theorem $a + a = 1$. For finding maxterm expansion, factor the function to obtain a product of sums,



introduce the missing variable in each sum term by using $aa' = 0$ and then factor again to obtain the maxterms.

Example . Express the Boolean function $f(x, y, z) = x + y'z$ in a sum of minterms.

Solution. The function has three variables x, y and z .

The first term x is missing two variables; therefore applying $a + a = 1$

$$x = x(y + y') = xy + xy'$$

This is still missing the variables z :

$$\begin{aligned} X &= xy(z + z') + xy'(z + z') \\ &= xyz + xyz' + xy'z + xy'z' \end{aligned}$$

The second term $y'z$ is missing one variable x :

$$y'z = y'z(x + x') = xy'z + x'y'z$$

$$\text{Hence } f(x, y, z) = x + y'z$$

$$= xyz + xyz' + xy'z + xy'z' \text{ using (1) and (2)}$$

$$= xyz + xyz' + xy'z + xy'z' + x'y'z \text{ applying the theorem}$$

$$a + a = a \text{ on } xy'z$$

Example . Express the Boolean function $f(a, b, c) = ab + a'c$ as a product of maxterms.

Solution. First convert the function into factor to obtain the product of sum forms by using distributive law $x + yz = (x + y)(x + z)$

$$f(a, b, c) = ab + a'c = (ab + a')(ab + c)$$

$$= (a + a)(b + a)(a + c)(b + c)$$

$$= (b + a')(a + c)(b + c)(a + a' = 1)$$

Since each sum term is missing one variable, applying $aa' = 0$ we get

$$a' + b = a' + b + cc' = (a' + b + c)(a' + b + c')$$

$$a + c = a + c + bb' = (a + b + c)(a + b' + c)$$

$$b + c = b + c + aa' = (a + b + c)(a' + b + c)$$

Now removing those terms that appear more than once $(a, a = a)$,



$$f(a, b, c) = (a + b + c) (a + b' + c) (a' + b + c) (a' + b + c')$$

which is a product of maxterms.

Conversion of Disjunctive Normal Form (DNF) of a Boolean function to its Conjunctive Normal Form (CNF) and vice- versa.

The following steps can be followed to convert a Boolean function given in disjunctive normal form to conjunctive normal form.

Conversion of DNF to CNF

1. Denote the given function which is in DNF by f .
2. Find f' which is the sum of those terms of the complete DNF in the variables of f , that are not present in the given function.
3. Take complement f'' of f' .
4. Simplify $R.H.S$ of f' by using De Morgan's laws. This will transform the $R.H.S$ in-to its DNF .
5. Since $f' = f$, the $R.H.S$ is the required DNF of the given function in CNF .

Example. Convert the Boolean function $f(x, y) = x.y' + x'y + x'y'$ to its conjunctive normal form.

Solution. Here $f(x, y) = x.y' + x'.y + x'.y'$

The complete DNF in two variables $x.y = x.y + x.y' + x'.y'$

Hence $f'(x, y) = x.y$

So, $f''(x, y) = (x.y)' = x' + y'$

Which is the required CNF of given function.





Gradeup UGC NET Super Superscription

Features:

1. 7+ Structured Courses for UGC NET Exam
2. 200+ Mock Tests for UGC NET & MHSET Exams
3. Separate Batches in Hindi & English
4. Mock Tests are available in Hindi & English
5. Available on Mobile & Desktop

Gradeup Super Subscription, Enroll Now