Rajmani Kumar,
Lecturer, Dept. of BCA
S.U.College, Hilsa (Nalanda)
Patliputra University, Patna

BCA-2nd Year **Paper-III**

BOOLEAN ALGEBRA

Introduction:

George Boole, a nineteenth-century English Mathematician, developed a system of logical algebra by which reasoning can be expressed mathematically. In 1854, Boole published a classic book, "An Investigation of the Laws of thought" on which he founded the Mathematical theories of Logic and Probabilities.

Boole‟s system of logical algebra, now called Boolean algebra, was investigated as a tool for analyzing and designing relay switching circuits by Claude E. Shannon at the Massachusetts institute of Technology in 1938. Shannon, a research assistant in the Electrical Engineering Department, wrote a thesis entitled "A" symbolic Analysis of Relay and Switching Circuits. As a result of his work, Boolean algebra is now, used extensively in the analysis and design of logical circuits. Today Boolean algebra is the backbone of computer circuit analysis.

Two Valued Logical Symbol:

Aristotle made use of a two valued logical system in devising a method for getting to the truth, given a set of true assumptions. The symbols that are used to represent the two levels of a two valued logical system are 1 and 0. The symbol 1 may represent a closed switch, a true statement, an "on" lamp, a correct action, a high voltage, or many other things. The symbol "O" may represent on open switch, a false statement, an "off" lamp, an incorrect action, a low voltage, or many other things.

For the electronics circuits and signals a logic 1 will represent closed switch, a high voltage, or an "on" lamp, and a logic 0 will represent an open switch, low voltage, or an "off" lamp. These describe the only two states that exist in digital logic systems and will be used to represent the in and out conditions of logic gates.

BOOLEAN OPERATIONS AND EXPRESSIONS

Variable, complement, and literal are terms used in Boolean algebra. A variable is a symbol used to represent a logical quantity. Any single variable can have a 1 or a 0 value. The complement is the inverse of a variable and is indicated by a bar over variable (overbar). For example, the complement of the variable A is $\overline{A}$. If A = 1, then $\overline{A}$ = 0. If A = 0, then $\overline{A}$ = 1. The complement of the variable A is read as "not A" or "A bar." Sometimes a prime symbol rather than an overbar is used to denote the complement of a variable; for example, B' indicates the complement of B. A literal is a variable or the complement of a variable.

1.Boolean Addition

Boolean addition is equivalent to the OR operation. In Boolean algebra, a sum term is a sum of literals. In logic circuits, a sum term is produced by an OR operation with no AND operations involved. Some examples of sum terms are A + $\overline{B}$, A + B, A + B + $\overline{C}$, and A + B + C + $\overline{D}$.

A sum term is equal to 1 when one or more of the literals in the term are 1. A sum term is equal to 0 only if each of the literals is 0.

When the + (the logical addition) symbol is placed between two variables, say X and Y, since both X and Y can take only the role 0 and 1, we can define the + Symbol by listing, all possible combinations for X and Y and the resulting value of X + Y.

The possible input and out put combinations may arranged as follows:

0 + 0 = 0
0 + 1 =1
1 + 0 = 1
1 + 1 = 1

Example
Determine the values of A, B, C, and D that make the sum term A + $\overline{B}$ + C + $\overline{D}$ equal to 0.

2.Boolean Multiplication

Boolean multiplication is equivalent to the AND operation. In Boolean algebra, a product term is the product of literals. In logic circuits, a product term is produced by an AND operation with no OR operations involved. Some examples of product terms are AB, A$\overline{B}$, ABC, and AB$\overline{C}$D.

A product term is equal to 1 only if each of the literals in the term is 1. A product term is equal to 0 when one or more of the literals are 0.

We can define the "." (logical multiplication) symbol or AND operator by listing all possible combinations for (input) variables X and Y and the resulting (output) value of X. Y as,

0 .0 = 0
0 .1 = 0
1 .0 = 0
1 .1 = 1

Note :Three of the basic laws of Boolean algebra are the same as in ordinary algebra; the commutative law, the associative law and the distributive law.

Example
Determine the values of A, B, C, and D that make the product term $AB\overline{CD}$ equal to 1.

LAWS AND RULES OF BOOLEAN ALGEBRA

■ Laws of Boolean Algebra

The basic laws of Boolean algebra-the commutative laws for addition and multiplication, the associative laws for addition and multiplication, and the distributive law-are the same as in ordinary algebra.

1.Commutative Laws

- The commutative law of addition for two variables is written as
  A+B = B+A

This law states that the order in which the variables are ORed makes no difference. Remember, in Boolean algebra as applied to logic circuits, addition and the OR operation are the same. Fig.(1) illustrates the commutative law as applied to the OR gate and shows that it doesn't matter to which input each variable is applied. (The symbol ≡ means "equivalent to.").

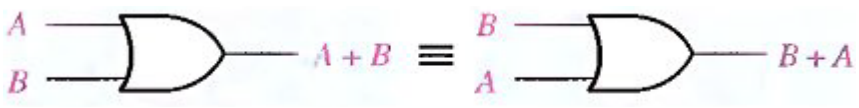


Fig.(1) Application of commutative law of addition.

- The commutative law of multiplication for two variables is

$$A.B = B.A$$

This law states that the order in which the variables are ANDed makes nodifference. Fig.(4-2), il1ustrates this law as applied to the AND gate.

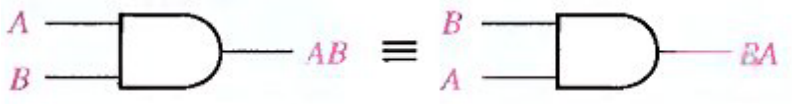


Fig.(2) Application of commutative law of multiplication.

The commutative law for addition and multiplication of two variables is written as,

$$A + B = B + A$$

And $$A \cdot B = B \cdot A$$

2. Associative Laws :

- The associative law of addition is written as follows for three variables:

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

This law states that when ORing more than two variables, the result is the same regardless of the grouping of the variables. Fig.(3), illustrates this law as applied to 2-input OR gates.

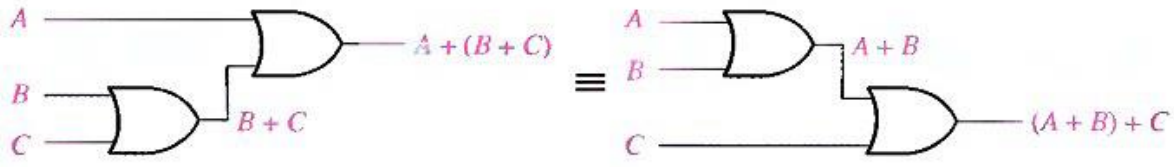


Fig.(3) Application of associative law of addition.

- The associative law of multiplication is written as follows for three variables:

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

This law states that it makes no difference in what order the variables are grouped when ANDing more than two variables. Fig.(4) illustrates this law as applied to 2-input AND gates.

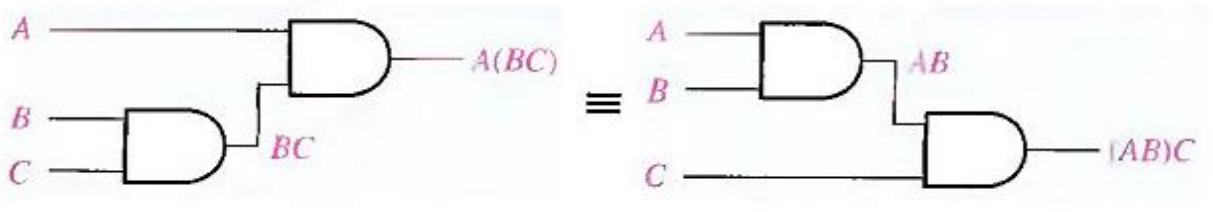


Fig.(4) Application of associative law of multiplication.

The associative law for addition and multiplication of three variables is written as,

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

And $$(A .B) . C = A. (B. C)$$

3.Distributive Law:

- The distributive law is written for three variables as follows:

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

This law states that ORing two or more variables and then ANDing the result with a single variable is equivalent to ANDing the single variable with each of the two or more variables and then ORing the products. The distributive law also expresses the process of factoring in which the common variable A is factored out of the product terms, for example,

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

Fig.(5) illustrates the distributive law in terms of gate implementation.

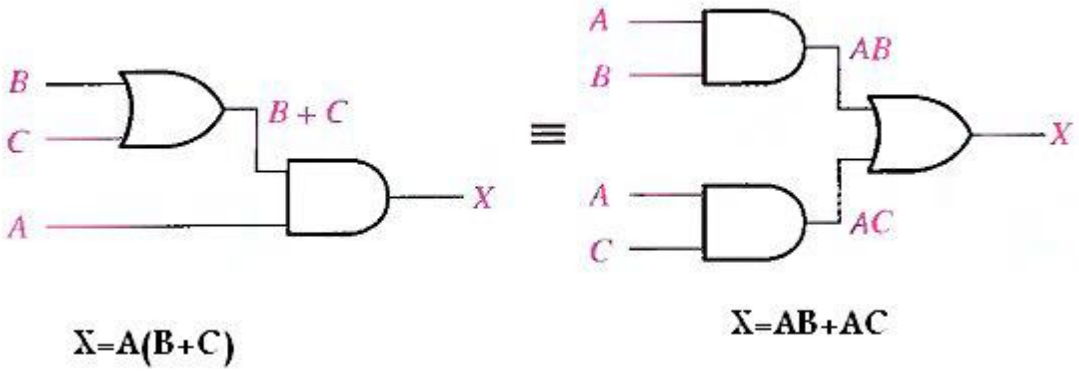


Fig.(5) Application of distributive law.

The distributive law for three variables involves, both addition and multiplication and is written as,

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

Note that while either '+' and „.“ s can be used freely. The two cannot be mixed without ambiguity in the absence of further rules.

For example does A . B + C means (A . B) + C or A . (B+ C)

These two form different values for A = O, B = 1 and C = 1,

because we have (A . B) + C = (0.1) + 1 = 1

and A . (B + C) = 0 . (1 + 1) = 0

which are different. The rule which is used is that „.“ is always performed before '+'. Thus X . Y + Z is (X.Y) + Z.

Rules of Boolean Algebra

Table(1) lists 12 basic rules that are useful in manipulating and simplifying oolean expressions. Rules 1 through 9 will be viewed in terms of their application to logic gates. Rules 10 through 12 will be derived in terms of the simpler rules and the laws previously discussed.

Table (1) Basic rules of Boolean algebra.

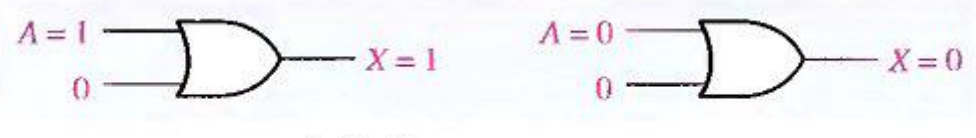
1. $A + 0 = A$	**7.** $A \cdot A = A$
2. $A + 1 = 1$	**8.** $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	**9.** $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	**10.** $A + AB = A$
5. $A + A = A$	**11.** $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	**12.** $(A + B)(A + C) = A + BC$

A, B, or *C* can represent a single variable or a combination of variables.

Rule (1).

A + 0 = A

A variable ORed with 0 is always equal to the variable. If the input variable A is 1, the output variable X is 1, which is equal to A. If A is 0, the output is 0, which is also equal to A. This rule is illustrated in Fig.(6), where the lower input is fixed at 0.



A+0=A

Fig.(6)

Rule 2.

A + 1 = 1

A variable ORed with 1 is always equal to 1. A 1 on an input to an OR gate produces a 1 on the output, regardless of the value of the variable on the other input. This rule is illustrated in Fig.(7), where the lower input is fixed at 1.

X=A+1=1

Fig.(7)

Rule 3.

A . 0 = 0

A variable ANDed with 0 is always equal to 0. Any time one input to an AND gate is 0, the output is 0, regardless of the value of the variable on the other input. This rule is illustrated in Fig.(8), where the lower input is fixed at 0.

X= A.0 = 0

Fig.(8)

Rule 4.

A . 1 = A

A variable ANDed with 1 is always equal to the variable. If A is 0 the output of the AND gate is 0. If A is 1, the output of the AND gate is 1 because both inputs are now 1s. This rule is shown in Fig.(9), where the lower input is fixed at 1.

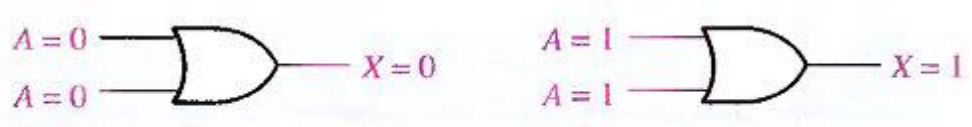
X = A . 1 = A

Fig.(9)

Rule 5.

A + A = A

A variable ORed with itself is always equal to the variable. If A is 0, then 0 + 0 = 0; and if A is 1, then 1 + 1 = 1. This is shown in Fig.(10), where both inputs are the same variable.



$X = A + A = A$

Fig.(10)

Rule 6.

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

A variable ORed with its complement is always equal to 1. If A is 0, then $0 + \overline{0} = 0 + 1 = 1$.

If A is l, then 1 + 1 = 1+ 0 = 1. See Fig.(11), where one input is the complement of the other.

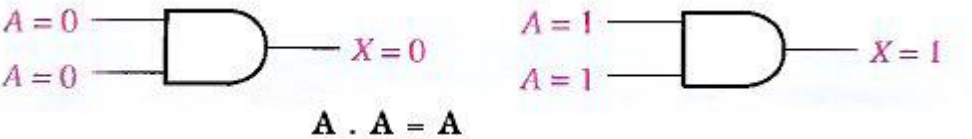
$X = A + \overline{A} = 1$

Fig.(11)

Rule 7.

$$A \cdot A = A$$

A variable ANDed with itself is always equal to the variable. If A = 0, then 0.0 = 0; and if A = 1. then 1.1 = 1. Fig.(12) illustrates this rule.



$A \cdot A = A$

Fig.(12)

Rule 8.

$$A \cdot \overline{A} = 0$$

A variable ANDed with its complement is always equal to 0. Either A or $\overline{A}$ will always be 0: and when a 0 is applied to the input of an AND gate. The output will be 0 also. Fig.(13) illustrates this rule.

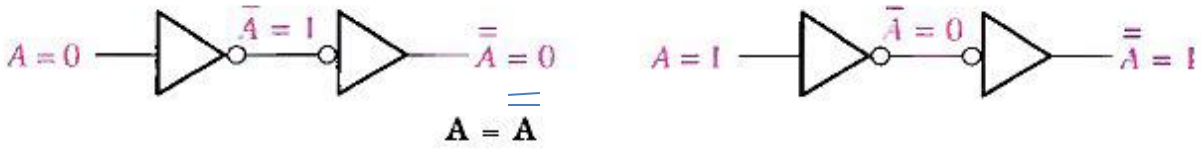
$A \cdot \overline{A} = 0$

Fig.(13)

Rule 9.

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

The double complement of a variable is always equal to the variable. If you start with the variable A and complement (invert) it once, you get $\overline{A}$. If you then take $\overline{A}$ and complement (invert) it, you get A, which is the original variable. This rule is shown in Fig.(14) using inverters.



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

Fig.(14)

Rule 10.

$$A + AB = A$$

This rule can be proved by applying the distributive law, rule 2, and rule 4 as follows:

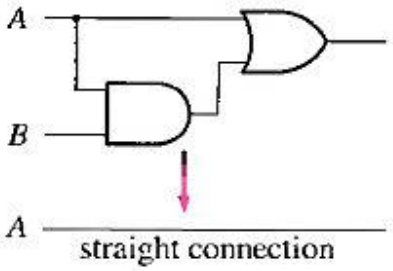
A + AB = A(1 + B)	Factoring (distributive law)
= A . l	Rule 2: (1 + B) = 1
= A	Rule 4: A . 1 = A

The proof is shown in Table (2), which shows the truth table and the resulting logic circuit simplification.

Table (2)

A	B	AB	A + AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

equal



Rule 11.

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

This rule can be proved as follows:

A + $\overline{A}$B = (A + AB) + $\overline{A}$B	Rule 10: A = A + AB
= (AA + AB) + $\overline{A}$B	Rule 7: A = AA
=AA +AB +A$\overline{A}$ +$\overline{A}$B	Rule 8: adding AA = 0
= (A + $\overline{A}$)(A + B)	Factoring

= 1. (A + $\bar{B}$) Rule 6: A + $\bar{A}$ = 1

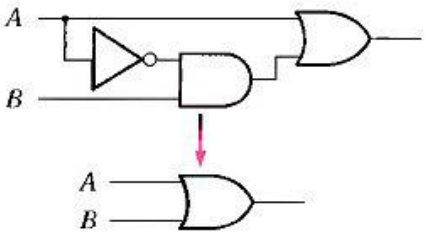
=A + B Rule 4: drop the 1

The proof is shown in Table (3), which shows the truth table and the resulting logic circuit simplification.

Table (3)

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

equal



Rule 12.

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

This rule can be proved as follows:

(A + B)(A + C) = AA + AC + AB + BC Distributive law

= A + AC + AB + BC Rule 7: AA = A

= A(1 + C) + AB + BC Rule 2: 1 + C = 1

= A. 1 + AB + BC Factoring (distributive law)

= A(1 + B) + BC Rule 2: 1 + B = 1

= A. 1 + BC Rule 4: A . 1 = A

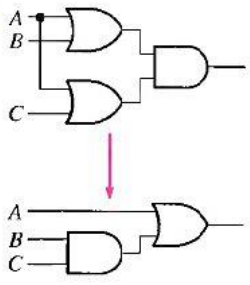
= A + BC

The proof is shown in Table (4), which shows the truth table and the resulting logic circuit simplification.

Table (4)

A	B	C	A + B	A + C	(A + B)(A + C)	BC	A + BC
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

equal



DEMORGAN'S THEOREMS

DeMorgan, a mathematician who knew Boole, proposed two theorems that are an important part of Boolean algebra. In practical terms. DeMorgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates and the equivalency of the NOR and negative-AND gates,

One of DeMorgan's theorems is stated as follows:

The complement of a product of variables is equal to the sum of the complements of the variables**,**

Stated another way,

The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables**.**

The formula for expressing this theorem for two variables is

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

DeMorgan's second theorem is stated as follows:

The complement of a sum of variables is equal to the product of the complements of the variables.

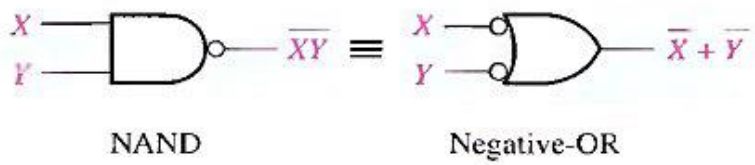
Stated another way,

The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables**,**

The formula for expressing this theorem for two variables is

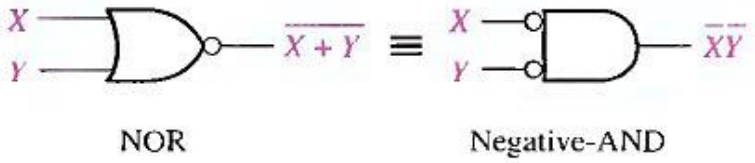
$$\overline{X + Y} = \overline{X}\,Y$$

Fig.(15) shows the gate equivalencies and truth tables for the twoequations above.



NAND Negative-OR

Inputs		Output	
X	Y	$\overline{XY}$	$\overline{X} + \overline{Y}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0



NOR Negative-AND

Inputs		Output	
X	Y	$\overline{X + Y}$	$\overline{XY}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Fig.(15) Gate equivalencies and the corresponding truth tables that illustrate DeMorgan's theorems.

As stated, DeMorgan's theorems also apply to expressions in which there are more than two variables. The following examples illustrate the application of DeMorgan's theorems to 3-variable and 4-variable expressions.

Example

Apply DeMorgan's theorems to the expressions $XY\overline{Z}$ and $X + \overline{Y + z}$.

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

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

Example

Apply DeMorgan's theorems to the expressions $WX\overline{YZ}$ and $W + \overline{X + y + z}$.

$WXYZ = \overline{W} + \overline{X} + \overline{y} + \overline{Z}$

$\overline{W + X + y + Z} = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$

Example

The Boolean expression for an exclusive-OR gate is $AB + \overline{A}\overline{B}$. With this as a starting point, use DeMorgan's theorems and any other rules or laws that are applicable to develop an expression for the exclusive-NOR gate.