CHAPTER – 3

BOOLEAN ALGEBRA AND LOGIC GATES

3.1 Basic Definition

Boolean algebra is used to analyze and simplify the digital circuits. Because it uses only the binary numbers i.e., 0 and 1, it is also called "binary algebra" or "logical algebra". The rules of Boolean algebra are different from those of the conventional algebra in the following manner:

i. Symbols used in Boolean algebra (usually letters) do not represent numerical values.

ii. Arithmetic operations like subtraction, division are not performed. Also, there are no fractions, negative numbers, square, square root, logarithms, imaginary number, etc.

iii. The third and most important point is that Boolean algebra allows only two possible values (0 or 1) for any variable.

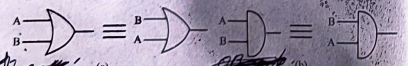
3.2 Basic Properties and Theorem of Boolean Algebra

The Boolean laws are:

1. **Commutative law:** This law allows change in the position of the input variables in OR and AND expression.

   i. $A + B = B + A$

   ii. $A.B = B.A$



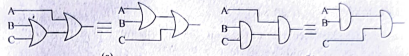
(a) (b)

Fig.: (a) Illustration of (i) (b) Illustration of (ii)

2. **Associative law:** The law states that it make no difference in what order the variables are grouped when ORing or ANDing more than two variables.

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

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



(a) (b)

Fig.: (a) Illustration of (i) (b) Illustration of (ii)

3. **Distributive law:** This law permits factoring or multiplying out an expression.

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

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

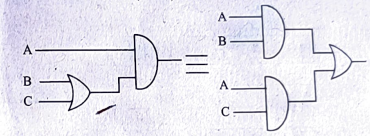


Fig.: Illustration of (i) and (ii)

4. **OR law:**

   i. $A + 0 = A$

   ii. $A + 1 = 1$

   iii. $A + A = A$

   iv. $A + \bar{A} = 1$

   For law (i), A can be 0 or 1

   Let $A = 0$

   $A + 0 = A$

   $0 + 0 = 0$

   $\therefore 0 = 0$

   Likewise, we can prove others.

5. **AND law:**

   i. $A.0 = 0$

   ii. $A.1 = A$

   iii. $A.A = A$

   iv. $A.\bar{A} = 0$

6. **Inversion law:**

   $A = \bar{\bar{A}}$

Theorem 1: **This theorem states that the complement of a product is equal to addition of the complements.**

It shows that NAND gate is equivalent to bubbled OR gate.

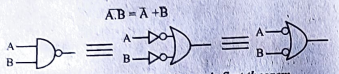


Fig.: Illustration of De-Morgan's first theorem.

Table: *Verification of De-Morgan's first theorem*

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

Theorem 2: **This theorem states that the complement of a sum is equal to the product of complements.**

It shows that NOR gate is equivalent to bubbled AND gate.

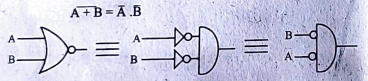


Fig.: Illustration of De-Morgan's second theorem.

Table: *Verification of De-Morgan's second theorem*

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

Duality Theorem

The duality theorem is one of those elegant theorems proved in advanced mathematics. We will state the theorem without proof. Here is what duality theorem says. Starting with a Boolean relation, we can derive another Boolean relation by:

i. changing each OR sign to an AND sign

ii. changing each AND sign to an OR sign

iii. complementing any 0 or 1 appearing in the expression.

For example, for the expression A+0 = A, the dual relation is A.1 = A.

The dual property is obtained by changing the OR sign to an AND sign, and by complementing the 0 to get a 1. The duality theorem is useful because it sometimes produce a new Boolean relation.

For example, A (B + C) = AB + AC

By changing each OR and AND operation, we get the dual relation

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

EXAMPLE:

1. Simplify using Boolean algebra:

i. $\bar{x}\,\bar{y}\,z + \bar{x}\,yz + x\bar{y}$

ii. $xy + \bar{x}\,z + yz$

iii. $\bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,y\bar{z} + xy\bar{z} + x\,\bar{y}\,\bar{z}$

⇒ **i.** $\bar{x}\,\bar{y}\,z + \bar{x}\,yz + x\bar{y}$

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

$= \bar{x}\,z + x\bar{y} \quad [\because y + \bar{y} = 1]$

ii. $xy + \bar{x}\,z + yz$

$= xy + \bar{x}\,z + yz\,(x + \bar{x})$

$= xy + \bar{x}\,z + xyz + \bar{x}\,yz$

$= xy(1 + z) + \bar{x}\,z\,(1 + y)$

$= xy + z\bar{x} \quad [\because 1 + z = 1 \text{ and } 1 + y = 1]$

iii. $\bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,yz + xy\bar{z} + x\,\bar{y}\,\bar{z}$

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

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

$= \bar{z}\,(\bar{x} + x)$

$= \bar{z}$

3.4 Logic Gates and Truth Table

Logic means certain declarative statement is true if certain conditions are fulfilled and false if does not meet that. The manner in which the digital circuit responds to an input is referred to as circuit's logic.

Logic gates are the basic building blocks of any digital system. It is an electronic circuit having one or more than one inputs and only one output. The relationship between the input and the output is based on certain logic. Based on logic, the gates are named as NOT gate, AND gate, OR gate, NAND gate, NOR gate, etc.

In the decimal system, which is used in our day-to-day life, the arithmetic operations such as addition, subtraction, multiplication, division, square, square root, modulus, etc. are used to solve equations. Similarly, to solve or simplify the logical expressions, used in digital circuits, we need to use logical operators. The three main logic operators may be listed as:

i. AND operator (A.B → logical multiplication)

ii. OR operator (A+B → logical addition)

iii. NOT operator ($\overline{A}$ → logical inversion)

The operation of a logic gate can be best understood with the help of a table called "truth table". The truth table consists of all the possible combinations of the inputs and the corresponding state of output of a logic gate. The relationship between the inputs and the outputs of a gate can be expressed mathematically by means of Boolean expression.

Logic gates are classified into three categories namely, the basic gates, the universal gates, and the special purpose gates.

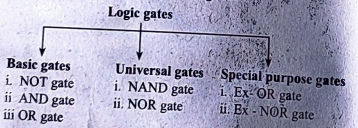


Fig.: Classification of gates.

The Basic Gates (NOT, OR, AND)

i. NOT Gate or Inverter

The NOT gate is a logic gate having one input (A) and one output (Y). It is also known as inverter because its output is the inverted version of its input.

Fig.: Symbol of NOT gate.

ii. AND Gate

AND gate performs the logical multiplication on its input. The output is high (Y = 1) if and only if all the inputs to the gate are high (1). The output is low (0), if at least one of the inputs is low (0). AND gate can have two or more inputs and only one output.

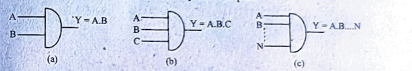


Fig.: (a) Two input AND gate (b) Three input AND gate. (c) N input AND gate.

iii. OR Gate

OR gate perform the logical addition on its inputs. The output will be high (1) if anyone input is high. The output will be low (0) if and only if all inputs are simultaneously low (0).

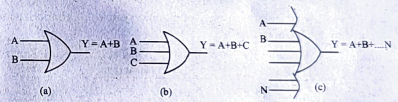


Fig.: (a) Two input OR gate (b) Three input OR gate. (c) N input OR gate.

The Universal Logic Gates (NOR, NAND)

i. **NAND Gate:** NAND gate is the combination of an AND gate and a NOT gate. A NAND gate is equivalent to an AND gate followed by an inverter. The output is high (1) if one or more inputs are low (0).

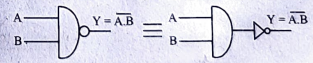


Fig.: Two input NAND gate

ii. **NOR Gate:** NOR gate is the combination of an OR gate and a NOT gate. It is opposite of OR gate. NOR gate is equivalent to an OR gate followed by an inverter. The output is low (0) if one or more inputs are high (1).

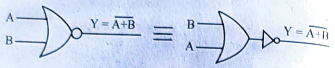


Fig.: Two input NOR gate

The Special Purpose Logic Gates (X-OR, X-NOR)

i. **Exclusive OR gate (X-OR):** The output is high if inputs are different and low if both inputs are same.

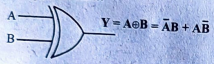


Fig.: Two input X-OR gate

ii. **Exclusive NOR gate (X-NOR):** The output is high if both inputs are same and low if inputs are different.

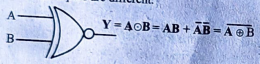
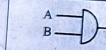
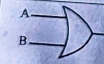
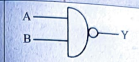
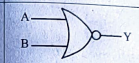


Fig.: Two input X-NOR gate

The following table illustrates the graphic symbol, algebraic function, and truth table of each gate.

Name	Graphic symbol	Algebraic function	Truth table		
AND		Y = A.B	A	B	Y
			0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		Y = A + B	A	B	Y
			0	0	0
			0	1	1
			1	0	1
			1	1	1
NOT		Y = $\bar{A}$	A		Y
			0		1
			1		0

Name	Graphic symbol	Algebraic function	Truth table		
NAND		Y = $\overline{A.B}$	A	B	Y
			0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		Y = $\overline{A+B}$	A	B	Y
			0	0	1
			0	1	0
			1	0	0
			1	1	0
X – OR		Y = $\bar{A}B + A\bar{B}$ = A⊕B	A	B	Y
			0	0	0
			0	1	1
			1	0	1
			1	1	0
X – NOR		Y = $AB + \bar{A}\bar{B}$ = $\overline{A \oplus B}$ = A⊙B	A	B	Y
			0	0	1
			0	1	0
			1	0	0
			1	1	1

3.5 Universality of NAND and NOR Gates

NAND gates and NOR gates are called universal gates because any type of gates or logic functions can be implemented by these gates. They are small in size and easy for fabrication.

Realization of Various Logic Gates by NAND

i. **NOT gate using NAND**

Y = $\overline{A.B}$

= $\overline{A.A}$ [∵ A = B]

= $\bar{A}$

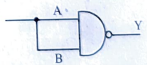


Fig.: NOT gate using NAND

ii. AND using NAND

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

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

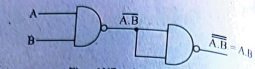


Fig.: AND using NAND

iii. OR using NAND

$Y = A + B$

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

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

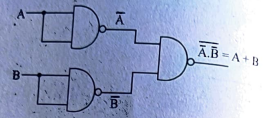


Fig.: OR using NAND

iv. NOR using NAND

$Y = \overline{A + B}$

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

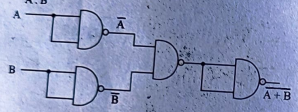


Fig.: NOR using NAND

v. X-OR using NAND

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

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

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

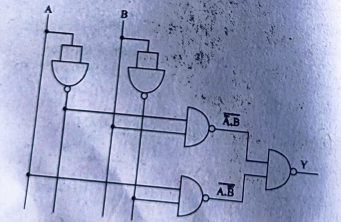


Fig.: X-OR using NAND

vi. X-NOR using NAND

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

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

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

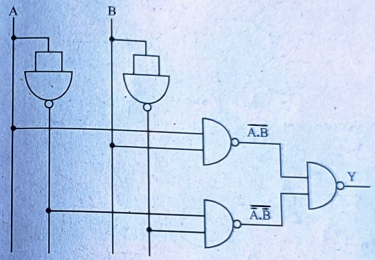


Fig.: X-NOR using NAND

Realization of Various Logic Gates by NOR

i. NOT gate using NOR

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

$= \overline{A + A}$ [$\because A = B$]

$= \overline{A}$

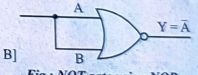


Fig.: NOT gate using NOR

ii. OR using NOR

$Y = A + B$

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

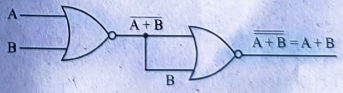


Fig.: OR using NOR

iii. **AND using NOR**

$Y = AB$

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

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

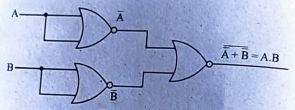


Fig.: AND using NOR

iv. **NAND using NOR**

$Y = \overline{A.B}$

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

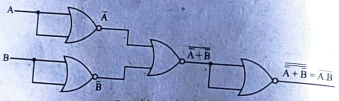


Fig.: NAND using NOR

v. **X-OR using NOR**

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

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

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

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

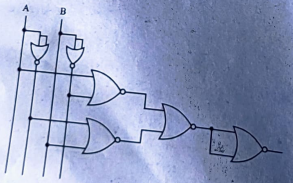


Fig.: X-OR using NOR

vi. **X-NOR using NOR**

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

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

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

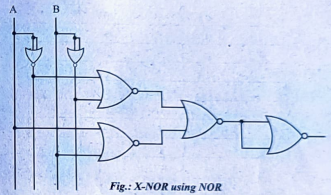


Fig.: X-NOR using NOR

3.6 AND-OR-INVERT Gates

AND-OR-INVERT (AOI) logic gates are two level compound (or complex) logic function constructed from the combination of one or more AND gates followed by a NOR gate. For a 4 input AND-OR- INVERT logic circuit, the output is low if both inputs A and B are high or both inputs C and D are high.

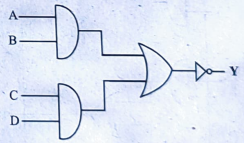


Fig.: AND-OR-INVERT gates

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

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

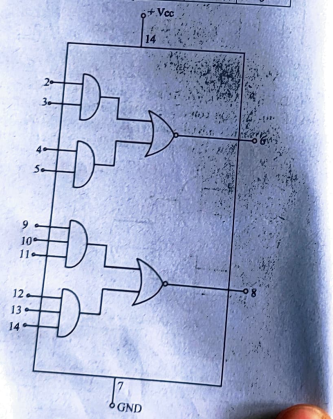


Fig.: AND-OR-INVERT IC

3.7 Positive and Negative Logic

The binary signals at the inputs or outputs of any gate may be one of two values, except during transitions. One signal value represents logic 1, and the other represents logic 0.

For a positive logic system, the most positive voltage level represents logic 1 state or high level (H) and lowest voltage level represents logic 0 state or low level (L).

For a negative logic system, the most positive voltage level represents logic 0 state and the lowest voltage level represents logic 1 state.

Example: If the voltage levels are –1 volt and –10 volt, then in a positive logic system, –1 volt represents logic 1 and –10 volt represents logic 0, and in negative logic system, –1 volt represents logic 0 and –10 volt represents logic 1.

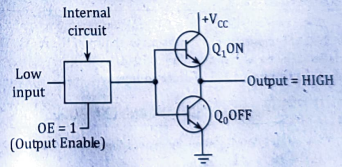
3.8 Tristate Logic

The three states in tristate logic are:

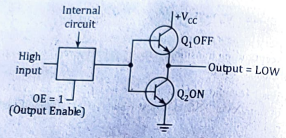
i. **High**

ii. **Low**

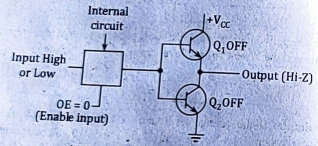
iii. **High impedance (Hi–Z) state**



(a) State 1: High



(b) State 2: Low



(c) State 3: High impedance state

An additional input called output enable (OE) is introduced. Q_1 and Q_2 are pull up and pull down transistors. The output stage block diagram of a tri-state inverter is shown in figure.

When OE = 1, the circuit operates as a normal inverter. If the input is 0 then, the output will be 1 (HIGH). The up transistor Q_1 will be ON. Similarly, when OE = 1 and input HIGH (1), the output will be LOW because the pull down transitor Q_2 will be ON.

High Impedance State

If the Output Enable (OE) = 0 (Zero), then irrespective of the status of input, both the transistors will remain off as shown in figure (c). The state of operation is called as high impedance (Hi-Z) state. In this state, the output terminal is open circuit i.e. oot connected anywhere.

SOLUTION TO IMPORTANT AND EXAM QUESTIONS

1. Prove the following Boolean expression: AB + AB'C + A'BC = AB + AC + BC.

Solution

L.H.S. = AB + AB'C + A'BC

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

= A(B + C) + A'BC [$\because$ A+A'B = A+B]

= AB + AC + A'BC

= AB + (A + A'B)C

= AB + (A + B)C

= AB + AC + BC

= R.H.S.

2. Prove the following Boolean expression:

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

Solution

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

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

= $A(B + C) + \bar{A}BC$ $\left[A + \bar{A}B = A + B\right]$

= $AB + AC + \bar{A}BC$

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

= $AB + C(A + B)$

= $AB + AC + BC$

= RHS.

3. Construct F = AB + CD using universal gates. [Spring 2018]

Solution:

F = AB + CD

Now, implementing using NAND universal gate,

F = AB + CD

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

$= \overline{(\overline{AB})\,.\,(\overline{CD})}$

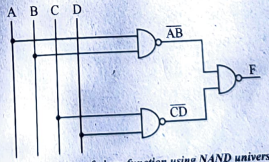


Fig.: Implementation of given function using NAND universal gate

Again, implementing using NOR universal gate,

$F = AB + \overline{C}D$

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

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

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

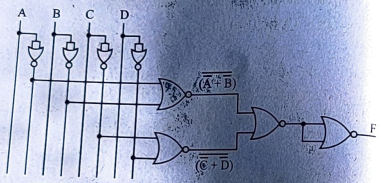


Fig.: Implementation of given function using NOR universal gates

4. Reduce the given expression in minimum number of literals using Boolean algebra and derive the truth table and implement in NAND logic.

A + B[AC + B {AC + CB + C')D}] [Spring 2013]

Solution:

$A + B[AC + B\{AC + CB + C')D\}]$

$= A + B[AC + B\{AC + BD + C'D\}]$

$= A + B[AC + ABC + BBD + BC'D]$

$= A + ABC + ABBC + BBBD + BBC'D$

$= A + ABC + ABC + BD + BC'D$ [$\because A.A = A$]

$= A + ABC + BD + BC'D$

$= A + ABC + BD\,(1 + C')$ [$\because 1 + A = 1$]

$= A\,(1 + BC) + BD$

$= A + BD$

Truth table

A	B	C	D	F = A + BD
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Now, implementation using NAND gates only,

$F = A + BD$

$= \{A + BD\}''$

$= \{A' . (BD)'\}'$

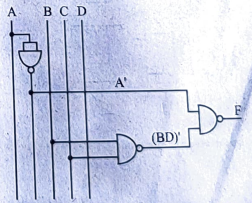


Fig.: Implementation of given function using NAND gates only