Ministry of Secondary Education
Progressive Comprehensive High School
PCHS Mankon – Bamenda
Department of **Computer Studies**

Republic of Cameroon
Peace – Work – Fatherland
School Year: 2013/2014

TOPIC: BOOLEAN ANGEBRA AND LOGIC GATES

Class: Comp. Sc. A/L **By:** DZEUGANG PLACIDE

We are familiar with the arithmetic operators add (+), subtract (-), multiply (x) and divide (/) which operate n numbers. Apart from these arithmetic operators, we can define logical operators which operate on what we called **truth values**. A truth value can be **TRUE** or **FALSE** and the common operators are: **AND, OR, NOT**. These logical operators are also called Boolean operators, from the English mathematician **George Boole**, who developed the theory of mathematic logic in the 19th century. This chapter aims to present the concept of Boolean algebra and its applications on logic circuits

Learning objectives

After studying this lesson, student should be able to:

- Use AND, OR and NOT operators and apply them on lo.gic gates
- Understand the laws and the principles of Boolean algebra.
- Gibe the logic function of a logic circuit and vice-versa
- Understand and use the rules of Boolean algebra to simplify functions and to minimize logic circuits

Contents

I. LOGIC GATES

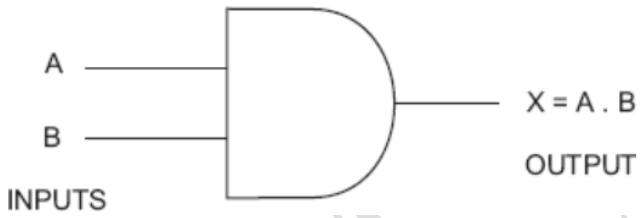
The term gate is used to describe the set of the basic electronic components, which when combined with each other are able to perform complex logical and arithmetic operations. As discussed earlier, everything in the digital world is based on the binary number system. Numerically, this involves only two symbols: 0 and 1. According to the need, we can use these symbols or we can equate them with others. Thus, when dealing with digital logic, we can specify that**: 0 = False = No. 1 = True = Yes**

I.1 Logic Operations and basic logic gates

There exist basically three logical operations: AND, OR and NOT. Each operator

I.1.1 AND Operation and AND Gate

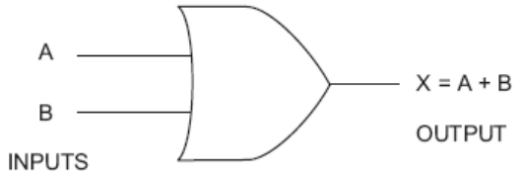
In the **AND operation**, a result of 1 occurs when all the input variables are 1. The **AND gate** is composed of two or more inputs and a single output, and performs logical multiplication. The multiplication sign stands for the AND operation, same for ordinary multiplication of 1s and 0s. The standard symbol for the AND gate and its truth are shown below. The expression **X = A.B** reads as *'X equals A AND B'*.



A	B	X=A.B
0	0	0
0	1	0
1	0	0
1	1	1

I.1.2 OR Operation and OR Gate

According to the **OR operation**, a result of 1 is obtained when any of the input variable is 1. In addition, the OR operation produces a result of 0 only when all the input variables are 0. The OR gate is composed of two or more inputs and a single output, and performs logical addition. The + sign stands for the OR operation, and not for ordinary addition.

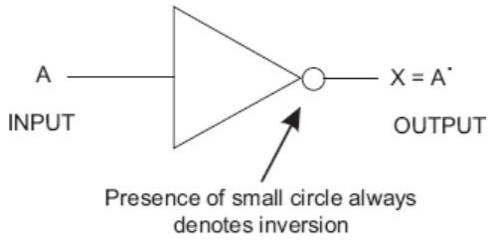


A	B	X=A+B
0	0	0
0	1	1
1	0	1
1	1	1

I.1.3 NOT Operation and NOT Gate

The NOT operation is unlike the OR and AND operations. This operation can be performed on a single-input variable. The NOT gate performs a basic logic function called inversion or complementation. The purpose of this gate is to change one logic level (HIGH / LOW) to the

opposite logic level. In terms of bits, it changes '1' to '0' and vice versa. The standard logic symbol for the NOT gate and the truth table illustrating the relationship between the variables and the logic gate operation are shown below respectively.



A	$X = \bar{A}$
0	1
1	0

x = A′ or **$\bar{A}$** where the prime (′) represents the NOT operation. This expression is read as:x equals NOT A. x equals the inverse of A. x equals the complement of A.

II. BOOLEAN ALGEBRA

Boolean algebra is a mathematical system, developed by the English mathematician, George Boole, which is used for the formulation of the logical statements with symbols so that the problems can be solved in a definite manner of ordinary algebra. In short, Boolean algebra is the mathematics of digital systems.

Since Boolean algebra deals with the binary number system, the variables used in the Boolean equations have only two possible values (0 or 1). Thus, for performing the logical algebraic operations, that is, 'addition' and 'multiplication', Boolean algebra follows certain rules.

II.1 Rules of Boolean algebra

These rules are shown in the table below

	Addition Rules (a)	**Multiplication Rules (b)**
1	0 + 0 = 0	0 · 0 = 0
2	0 + 1 = 1	0 · 1 = 0
3	1 + 0 = 1	1 · 0 = 0
4	1 + 1 = 1	1 · 1 = 1
5	A + 0 = A	A · 0 = 0
6	A + 1 = 1	A · 1 = A
7	A + A = A	A · A = A
8	A + $\bar{A}$ = 1	A · $\bar{A}$ = 0
9	A + AB = A	
10	A + $\bar{A}$ B = A + B	
11	(A+B)(A+C) = A + BC	
12	AC+BC = A(B+C)	

These rules can be checked by the use of truth table. Some of these rules can be derived from simpler identities derived in this package.

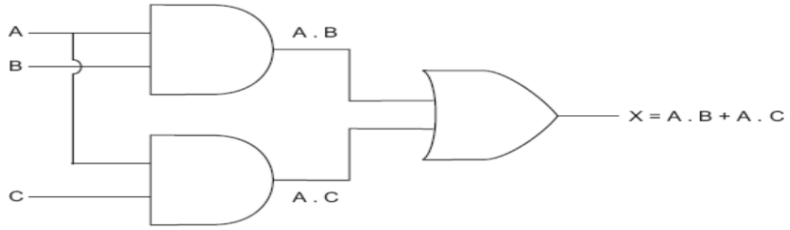
II.2 Laws of Boolean algebra

The following are different laws of Boolean algebra:

Law	Addition	Multiplication
Commutative	A + B = B + A	AB = BA
Associative	A + (B + C) = (A + B) + C	A (BC) = (AB) C
Distributive	A (B + C) = AB + AC A + (BC) = (A + B) (A + C)	
De Morgan's Law	$\overline{A+B} = \bar{A}.\bar{B}$ $\overline{AB} = \bar{A}+\bar{B}$	

III. LOGIC CIRCUITS BOOLEAN FUNCTIONS

A **Logic circuit** to a number of gates joined together to produce a specific output from given inputs. For example, let's consider the following circuit: It contains three inputs A, B and C



A **Boolean function** is an expression formed with binary variables and logical operators (OR, AND, NOT and equal sign). In essence, a truth table is a list, which defines a Boolean function. For example, X = A · B + A · C. Let us consider the truth table of the given value as shown

Inputs					Output
A	**B**	**C**	**A.B**	**A.C**	**X = A·B + A·C**
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	**1**
1	1	0	1	0	**1**
1	1	1	1	1	**1**

According to the truth table above we realize that $X = A\bar{B}C + AB\bar{C} + ABC$. This is called the **disjunctive normal form** of the function f; the combinations formed by considering the rows with an output value of 1 are joined by the disjunctive connective, OR.

We can then prove that the expressions $A\bar{B}C + AB\bar{C} + ABC$ and $AB + AC$ are equivalent

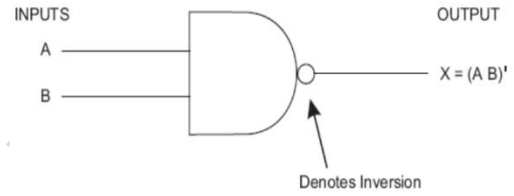
$$X = A\bar{B}C + AB\bar{C} + ABC$$
$$X = A(\bar{B}C + B\bar{C} + BC)$$
$$X = A[\bar{B}C + B(\bar{C} + C)]$$
$$X = A(\bar{B}C + B)$$
$$X = A(B + C)$$
$$X = AB + AB$$

IV. COMBINATION OF LOGIC GATES

Using combinations of logic gates, complex operations can be performed. Although there is no limit to the number of gates that can be arrayed together in a single device, nevertheless, in practice, there is a limit to the number of gates that can be packed into a given physical space. Some basic combination gates are: **NAND** gate, **NOR** gate, Exclusive-OR (**XOR**) and Exclusive-NOR (**XNOR**) gate

IV.1. NAND Gate

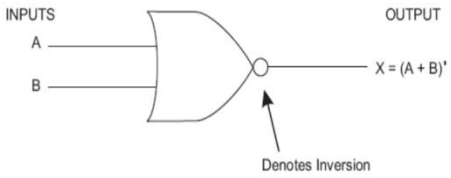
The NAND, which is composed of two or more inputs and a single output, is a very popular logic element because it may be used as a universal function. That is, it may be employed to construct an inverter, an AND gate, an OR gate or any combination of these functions. The term 'NAND' is formed by the combination of NOT-AND and implies an AND function with an inverted output. The standard symbol and the truth table for the NAND gate are shown below. The logical operation of the NAND gate is such that the output is LOW (0) only when all the inputs are HIGH (1)



A	B	$X = A \uparrow B$ (NAND)
0	0	1
0	1	1
1	0	1
1	1	0

IV. 2. NOR Gate

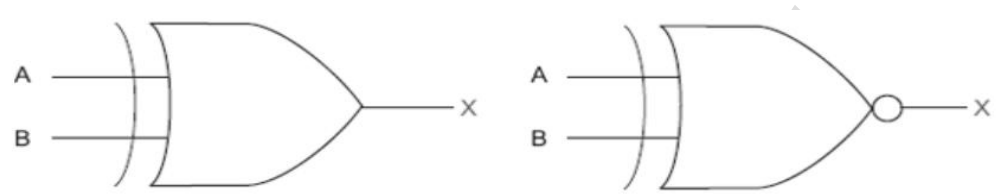
The NOR gate, which is composed of two or more inputs and a single output, also has a universal property. The term 'NOR' is formed by the combination of NOT-OR and implies an OR function with an inverted output. The standard symbol and the truth table for the NOR gate are shown below. The logical operation of the NOR gate is such that the output is HIGH (1) only when all the inputs are LOW



A	B	$X = A \downarrow B$ (NOR)
0	0	1
0	1	0
1	0	0
1	1	0

IV.3. Exclusive-OR (XOR) and Exclusive-NOR (XNOR) Gate

These gates are usually formed from the combination of the other logic gates already discussed. However, because of their functional importance, these gates are treated as basic gates with their own unique symbols. The truth tables and the standard symbols for the XOR and XNOR gates, are listed below respectively. The exclusive-OR is an 'inequality' function and the output is HIGH (1), when the inputs are not equal to each other. Conversely, the exclusive-NOR is an 'equality' function and the output is HIGH (1) when the inputs are equal to each other.



A	B	$X = A \oplus B$ (XOR)	$X = A \odot B$ (XNOR)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The exclusive-OR gate and exclusive-NOR gate are denoted by the $\oplus$ and $\odot$, respectively. In addition, these gates perform the following Boolean functions:

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

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

IV.4. Minimisation with NAND or NOR gates

When designing combinatorial circuits, efficiency is sought by minimising the number of gates in a circuit. Many computer circuits make use NAND gate or NOR gate which are used to replace NOT AND and NOT OR respectively, thereby reducing the number of gates. It has been proved that all the basic logical operators can be represented using only NAND operator or NOR operator. They are then call **universal gates**

Example

(a) $\bar{A} = \bar{A} + \bar{A} = \overline{A\bar{A}} = A \uparrow A$ (b) $\bar{A} = \overline{A + A} = A \downarrow A$

c) $A + B = \overline{\overline{A + B}} = \overline{\bar{A}\bar{B}} + \bar{A} \uparrow \bar{B} + (A \uparrow A) \uparrow (B \uparrow B)$

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

EXERCISES

Exercise 1 MCQ

1. Select the Boolean expression that is not equivalent to $x \cdot x + x \cdot x$

(a) $x \cdot (x + x)$ *(b)* $(x + x) \cdot x$ *(c)* $\bar{x}$ *(d)* x

2. Select the expression which is equivalent to $x \cdot y + x \cdot y \cdot z$

(a) $x \cdot y$ *(b)* $x \cdot z$ *(c)* $y \cdot z$ *(d)* $x \cdot y \cdot z$

3. Select the expression which is equivalent to $(x + y) \cdot (x + y)$

(a) y *(b)* $\bar{y}$ *(c)* x *(d)* $\bar{x}$

4. Select the expression that is not equivalent to $x \cdot (x+ y) + y$

(a) $x \cdot x+ y \cdot (1 + x)$ *(b)* $0 + x \cdot y + y$ *(c)* $x \cdot y$ *(d)* y

Exercise 2 :

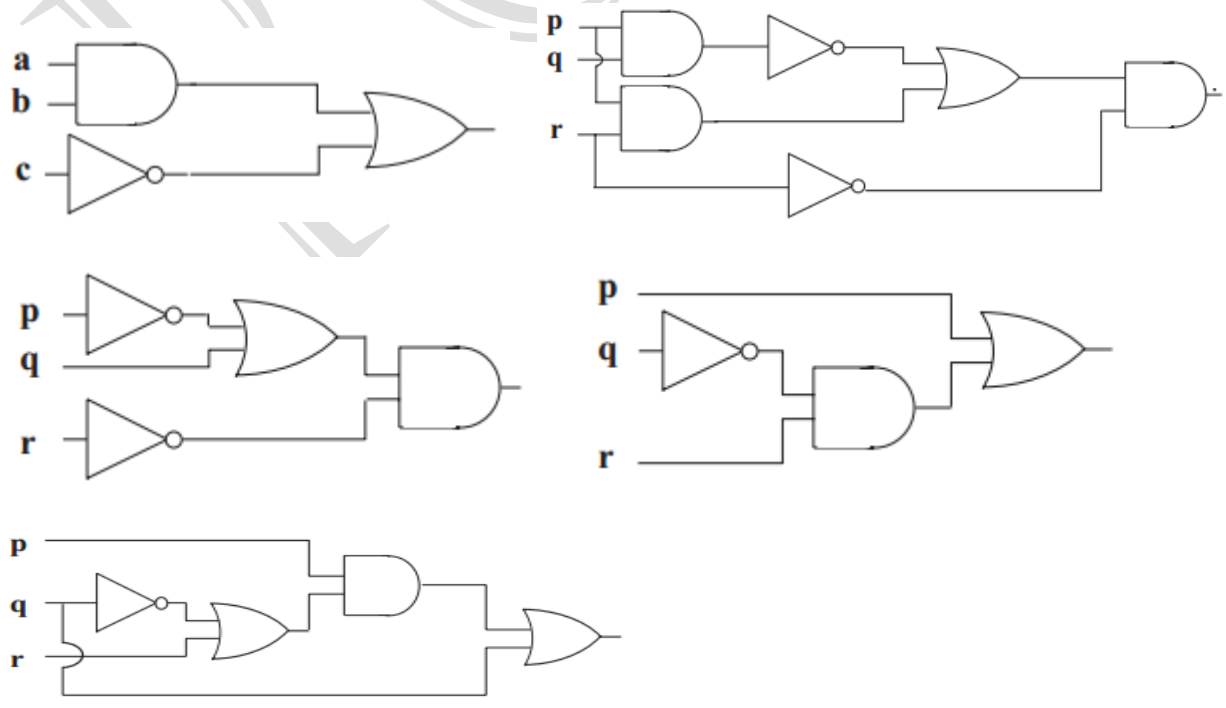
Use logic gates to represent these expressions and draw up the corresponding truth tables.

1. $x(\bar{y}+x)$ 2. $a+(\bar{b}c)$ 3. $b(a+(b+c))$

4. $\overline{AB + C}$ 5. $AB + \bar{C}$ 6. $\bar{C}(AB + \overline{AB})$

Exercise 3:

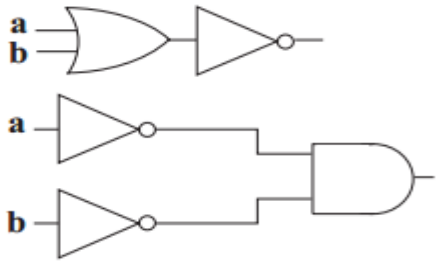
Write down the Boolean expression for each of the circuits below.



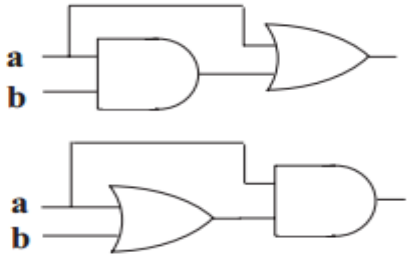
Exercise 4:

Show if these combinatorial circuits are equivalent by working out the Boolean expression and the truth table for each circuit.

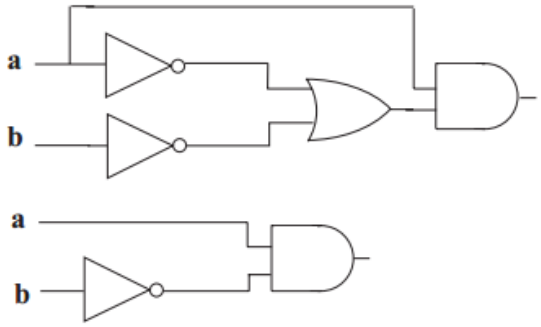
1.



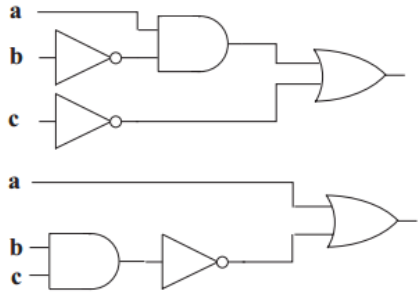
2.



3.



4.



Exercise 5

Draw the truth table for the Boolean function defined as $f(a,b,c) = a(\bar{b} + c)$

Exercise 6

For the given truth table, form a Boolean function

A	***B***	***C***	***f(A,B,C)***
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Exercise 7:

A burglar alarm for a house is controlled by a switch. When the switch is on, the alarm sounds if either the front or back doors or both doors are opened. The alarm will not work if the switch is off. Design a circuit of logic gates for the alarm and draw up the corresponding truth table.

Exercise 8

Find the disjunctive normal form of the Boolean function for these truth tables:

1.

a	b	f (a, b)
0	0	1
0	1	0
1	0	1
1	1	0

2.

a	b	f (a, b)
0	0	1
0	1	1
1	0	0
1	1	1

3.

a	b	c	f (a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x	y	z	f (x, y, z)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Exercise 9

Design circuits for each of the following using only NAND gates and then NOR gates.

1) AB 2) A$\bar{B}$ 3)$\bar{A}\bar{B} + \bar{B}$ 4) $\bar{A}$+B

Exercise 10

Simplify the following expressions and check your answer by drawing up truth tables.

(a) $abc + \bar{a}bc$
(b) $a + \bar{a}bc + \bar{a}b\bar{c}$
(c) $pq + (\bar{p} + \bar{q})(r + s)$

Exercise 11

(a) Establish a truth table for the Boolean function $f(a, b, c) = (\bar{a} + b)(\bar{c} + b)$
(b) Design a circuit using as few AND, OR and NOT gates as possible to model the function in (a).