Boolean Algebra and Logic Gates

So far we dealt with binary numbers that have two discrete values (0 or 1). Binary logic can also be defined with **Boolean Functions** 'set of **elements** (input variables) and ***operators*** (the rule that assigns to each pair of elements)'.

F = ABC

Boolean Algebra is the set of rules used to **simplify** the given logic expression without changing its functionality.

Why do use Boolean algebra?

Finding a simpler but equivalent representation of a circuit can reduce the overall cost of the design.

Laws of Boolean Algebra

There are only two binary operators commonly used in Boolean expression. These are the OR operator (**+**) and the AND operator (·). The rules for the two binary operators are shown in the truth tables below:

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

Commutative Law:

$$A.B = B.A$$
$$A + B = B + A$$

Associative Law:

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

Distributive Law:

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

Duality

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

Other important rules:

$$A + 0 = A \qquad A.0 = 0$$
$$A + 1 = 1 \qquad A.1 = A$$
$$A + A = A \qquad A.A = A$$
$$A + A' = 1 \qquad A.A' = 0$$

Ex) Simplify the Boolean Function F = AB+ BC + $\overline{B}$C

$$\begin{aligned} F &= AB + BC + \overline{B}C \\ &= AB + C(B + \overline{B}) \\ &= AB + C \end{aligned}$$

Ex) Simplify the Boolean Function F= A′C′ + ABC + AC'

Truth table: Table shows all the possible combinations of the **inputs** as well as the **output** based on the given function. In other words, it shows the input - output relationship.

For example, the distributive law $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ can be proved by forming a truth table of all possible values of x, y, and z. For each combination, we derive $x \cdot (y + z)$ and show that the value is the same as the value of $(x \cdot y) + (x \cdot z)$

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$y + z$	$x \cdot (y + z)$
0	0
1	0
1	0
1	0
0	0
1	1
1	1
1	1

$x \cdot y$	$x \cdot z$	$(x \cdot y) + (x \cdot z)$
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	1	1
1	0	1
1	1	1

Demorgan's theorems

Demorgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted.

- The first theorem says that when the OR of two variables is inverted, the result will be the AND of these inverted variables.

$$\overline{(x + y)} = \overline{x} \cdot \overline{y}$$

- The second Theorem (17) says that when the AND of two variables is inverted, this is the same as inverting each variable individually and then OR them.

$$\overline{(x \cdot y)} = \overline{x} + \overline{y}$$

Each of Demorgan's theorems can readily be proven by checking for all possible combinations of x and y in the truth table.

x	y	$x + y$	$\overline{(x + y)}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

$\overline{x}$	$\overline{y}$	$\overline{x}\,\overline{y}$
1	1	1
1	0	0
0	1	0
0	0	0

Ex) Simplify the following expression

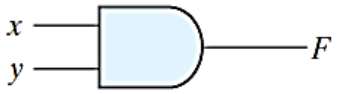
$$F = \overline{(A\overline{B} + C)} + \overline{B}\,\overline{C}$$

Logic Gates

Boolean functions are expressed in terms of AND, OR, and NOT operations. Therefore, it is Possible to implement all the Boolean functions with these basic electronic circuits. Logic gates are basically no more than a set of transistors, diodes and resistors combined together to perform the required function.

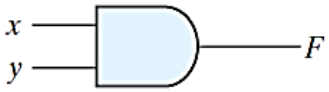
And Gate

This gate produces a **Logic 1** output only when all the inputs are Logic 1. When any of the inputs is LOW (Logic 0), the output is LOW.

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1

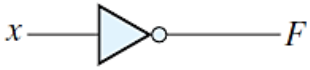
OR gate

This gate produces a High (Logic 1) on the output when any of the inputs is High. The output is Low only when all of the inputs are Low.

Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1

NOT (Inverter) Gate

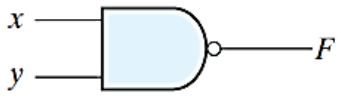
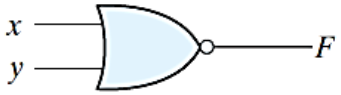
This gate produces an inversed value of the input. The output is high when the input is low and vice versa.

Name	Graphic symbol	Algebraic function	Truth table
Inverter		$F = x'$	x | F 0 | 1 1 | 0

Prepared by: Mohammed N. Y. Almaged

NAND and NOR gates

NAND and NOR gates are inverting of AND & OR gates. The standard symbols for each gate will have a bubble on the output of the gate AND & OR gates.

NAND		$F = (xy)'$	x	y	F
			0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$F = (x + y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
			1	1	0

Exclusive-OR (XOR)

The exclusive OR (XOR) is a logic gate that performs the following Boolean operation:

$$x \oplus y = xy' + x'y$$

The exclusive OR output is equal to one if x and y have different values, while it is equal to zero if they are the same.

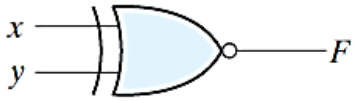
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x	y	F
			0	0	0
			0	1	1
			1	0	1
			1	1	0

Exclusive-NOR (XNOR)

The exclusive NOR (XNOR) is a logic gate that performs the following Boolean operation:

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

Which is the inversion of the X-OR

Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x	y	F
			0	0	1
			0	1	0
			1	0	0
			1	1	1