Boolean Algebra and Basic Operations

Due to historical reasons, digital circuits are called switching circuits, digital circuit functions are called switching functions and the algebra is called switching algebra. The algebraic system known as Boolean algebra named after the mathematician George Boole. George Boole Invented multi-valued discrete algebra (1854) and E. V. Huntington developed its postulates and theorems (1904). Historically, the theory of switching networks (or systems) is credited to Claude Shannon, who applied mathematical logic to describe relay circuits (1938). Relays are controlled electromechanical switches and they have been replaced by electronic controlled switches called logic gates. A special case of Boolean Algebra known as **Switching Algebra** is a useful mathematical model for describing the combinational circuits. In this section we will briefly discus how the Boolean algebra is applied to the design of digital systems.

Examples of **Huntington's postulates** are given below:

Closure

If X and Y are in set (0, 1) then operations $X+Y$ and $X \cdot Y$ are also in set (0, 1)

Identity

$X+0=X$ $\qquad X \cdot 1=X$

Distributive

$$X \cdot (Y+Z)=(X \cdot Y)+(X \cdot Z)$$
$$X+(Y \cdot Z)=(X+Y) \cdot (X+Z)$$

Complement

$$X+\overline{X}=1$$
$$X \cdot \overline{X}=0$$

Note that for each property, one form is the dual of the other; (zeros to ones, ones to zeros, '.' operations to '+' operations, '+' operations to '.' operations).

From the above postulates the following theorems could be derived.

Associative

$$X+(Y+Z)=(X+Y)+Z$$
$$X \cdot (Y \cdot Z)=(X \cdot Y) \cdot Z$$

Dempotence

$$X \cdot X = X$$
$$X + X = X$$

Absorption

$$X + (X \cdot Y) = X$$
$$X \cdot (X + Y) = X$$

Simplification

$$X + (\overline{X} \cdot Y) = X + Y$$
$$X \cdot (\overline{X} + Y) = X \cdot Y$$

Consensus

$$X \cdot Y + \overline{X} \cdot Z + Y \cdot Z = X \cdot Y + \overline{X} \cdot Z$$
$$(X + Y) \cdot (\overline{X} + Z) \cdot (Y + Z) = (X + Y) \cdot (\overline{X} + Z)$$

Adjacency

$$X \cdot Y + X \cdot \overline{Y} = X$$
$$(X + Y) \cdot (X + \overline{Y}) = X$$

Demorgans

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

In general form

$$\overline{F(\cdot, +, X_1, \ldots X_n)} = G(+, \cdot, \overline{X_1}, \ldots \overline{X_n})$$

Very useful for complementing function expressions; for example

$$F = X + Y \cdot Z; \qquad \overline{F} = \overline{X + Y \cdot Z}$$
$$\overline{F} = \overline{X} \cdot \overline{Y \cdot Z} \qquad F = \overline{X} \cdot (\overline{Y} + \overline{Z})$$
$$\overline{F} = \overline{X} \cdot \overline{Y} + \overline{X} \cdot \overline{Z}$$

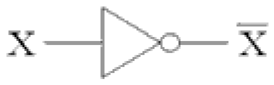
Switching Algebra Operations

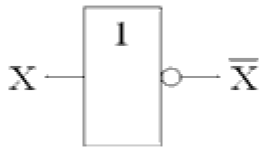
A set is a collection of objects (or elements) and for example a set Z {0, 1} means that Z is a set containing two elements distinguished by the symbols 0 and 1. There are three primary operations AND , OR and NOT.

NOT

It is anary complement or inversion operation. Usually shown as over bar ($\overline{X}$), other forms are X' and $\sim X$

X	$\overline{X}$
0	1
1	0

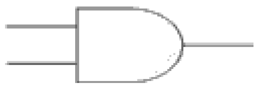


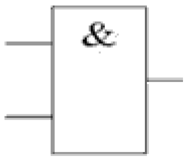


AND

Also known as the conjunction operation; output is true (1) only if all inputs are true. Algebraic operators are '.', '&', ' $\wedge$'

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

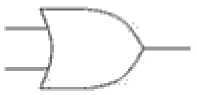


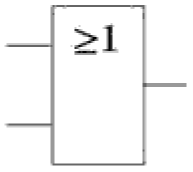


OR

Also known as the disjunction operation; output is true (1) if any input is true. Algebraic operators are '+', '|', '$\vee$'

X	Y	$X+Y$
0	0	0
0	1	1
1	0	1
1	1	1





AND and OR are called binary operations because they are defined on two operands X and Y. Not is called a unary operation because it is defined on a single operand X. All of these operations are closed. That means if one applies the operation to two elements in a set Z {0, 1}, the result will be always an element in the set B and not something else.

Like standard algebra, switching algebra operators have a precedence of evaluation. The following rules are useful in this regard.

1. NOT operations have the highest precedence

2. AND operations are next

3. OR operations are lowest

4. Parentheses explicitly define the order of operator evaluation and it is a good practice to use parentheses especially for situations which can cases doubt.

Note that in Boolean algebra the operators AND and OR are not linear group operations; so one cannot solve equations by "adding to" of "multiplying" on both sides of the equal sign as is done with real, complex numbers in standard algebra.

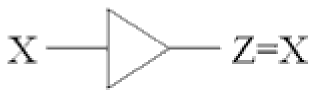
Additional Logic Operation

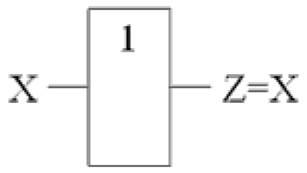
For two inputs, there are 16 ways we can assign output values. Besides AND and OR, there are five other operations which are useful.

BUFFER

The unary Buffer operation is useful in the real world

X	Z=X
0	0
1	1

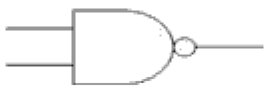


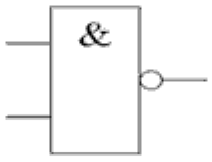


NAND

NAND (NOT - AND) is the complement of the AND operation

X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

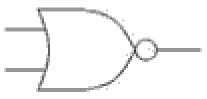


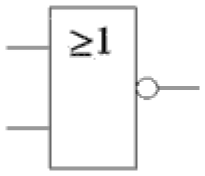


NOR

NOR (NOT - OR) is the complement of the OR operation

X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

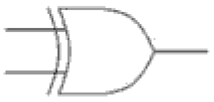


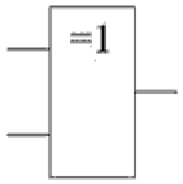


XOR

Exclusive OR is similar to the inclusive OR except output is 0 for 1. It is stated in other words as the output is 1 when modulo 2 input sum is equal to 1.

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

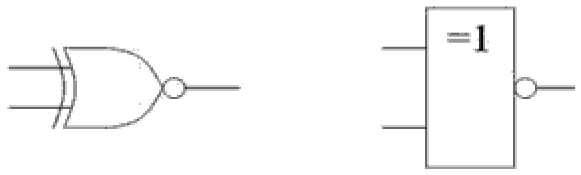




XNOR

Exclusive NOR is the complement of the XOR operation. Alternatively the output is 1 when modulo 2 input sum is not equal to 1.

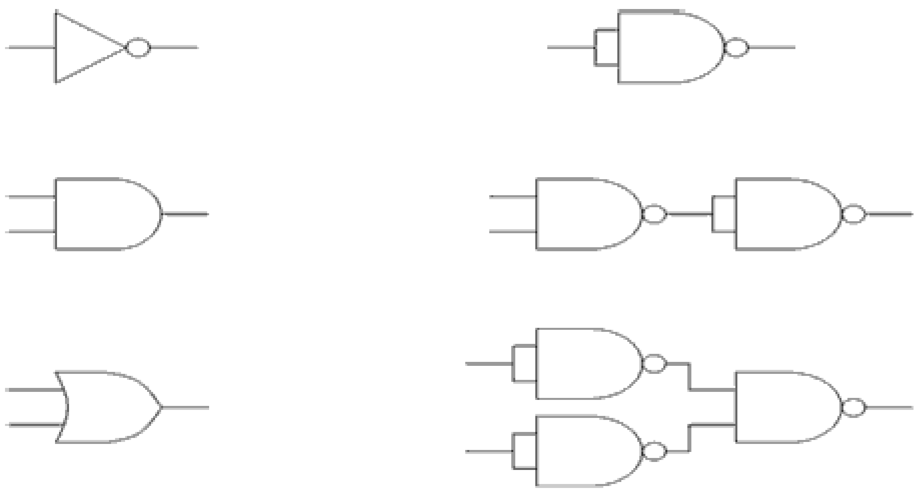
X	Y	$\overline{X \oplus Y}$
0	0	1
0	1	0
1	0	0
1	1	1



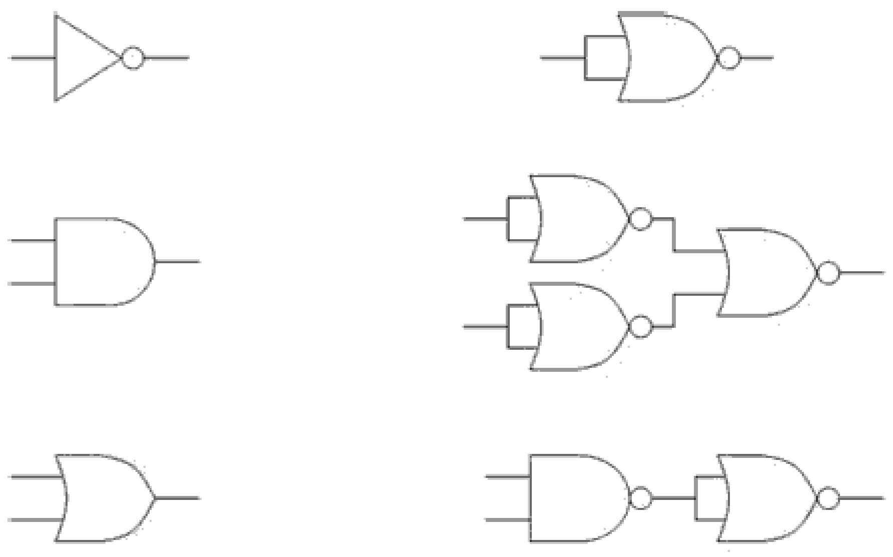
Minimal Logic Operator Sets

AND , OR, NOT are all that's needed to express any combinational logic function as switching algebra expression. However two other minimal logic operator sets are also possible with NAND gates or NOR gates. The following is a demonstration of how just NANDs or NORs can do AND , OR, NOT operations.

NAND as a Minimal Set



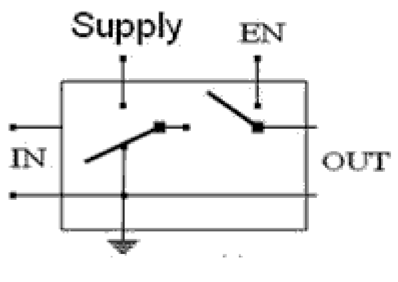
NOR as a Minimal Set



Three State Outputs

Standard logic gate outputs only have two states; high and low. Outputs are effectively either connected to +V or ground, that means there is always a low impedance path to the supply rails. In certain applications require a logic output that we can "turn off" or disable. It means that the output is disconnected (high impedance state). This is the three-state output and can be implemented by a stand-alone unit (a buffer) or part of another function output. This circuit is so-called tri-state because it has three output states: high (1), low (0), and high impedance (Z).

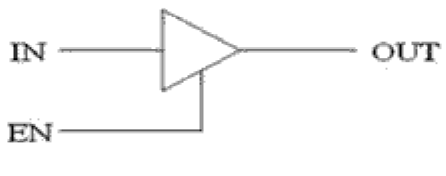
In the logic circuit of Fig. 15(a), there is an additional switch to a digital buffer, which is called as enabled input denoted by *E* . When *E* is low, the output is disconnected from the input circuit. When *E* is high, the switch is connected and the circuit behaves like a digital buffer. All these states are listed in Truth Table 15(b). Figure 8.14(c) depicts the symbol of a Tri-state Buffer.



(a)

IN	EN	OUT
X	0	HI-Z
0	1	0
1	1	1

(b)



(c)

Fig. 15: (a) Switch Configuration, (b) Truth Table, and (c) Symbol of a Tri-state Buffer