

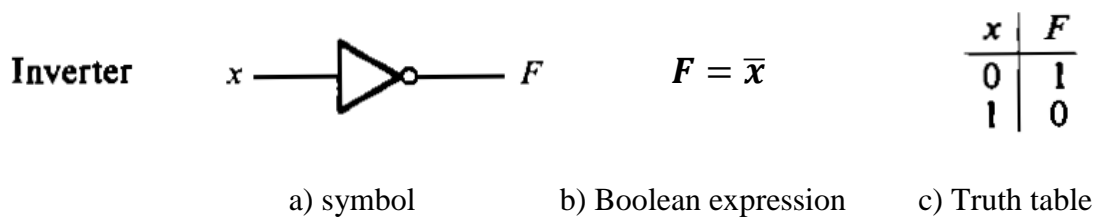
Logic Gates

Introduction:

The logic gate is the basic building block in digital systems. Logic gates operate with binary numbers. Gates are therefore referred to as binary logic gates. All voltages used with logic gates will be either HIGH or LOW. In this lecture, a HIGH voltage will mean a binary 1. A LOW voltage will mean a binary 0. Remember that logic gates are electronic circuits. These circuits will respond only to HIGH voltages (called 1s) or LOW (ground) voltages (called 0s). All digital systems are constructed by using only three basic logic gates. These basic gates are called the AND gate, the OR gate, and the NOT gate.

1- The NOT gate:

a NOT gate is also called an inverter. a NOT gate, or inverter, is an unusual gate. The NOT gate has only one input and one output. Many symbols can be used for NOT gate such as: $\bar{\quad}$, \prime . Fig(1) illustrates the logic symbol for the NOT gate, Boolean expression and the truth table. Boolean expression is a form of symbolic logic that shows how logic gates operate.



Fig(1). The NOT gate symbol, Boolean expression and truth table

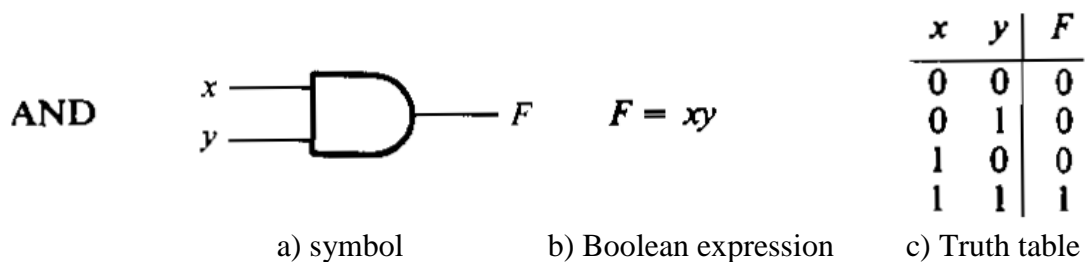
The input is always changed to its opposite. If the input is 0, the NOT gate will give its complement, or opposite, which is 1. If the input to the NOT gate is a 1, the circuit will complement it to give a 0. The double inverted x is equal to the original x .

The laws of Boolean algebra govern how NOT gates operate are:

$$\begin{array}{llll}
 & \bar{0} = 1 & \text{and} & \bar{1} = 0 \\
 \text{If} & x = 0 & \text{then} & \bar{x} = 1 \\
 \text{If} & x = 1 & \text{then} & \bar{x} = 0 \\
 & \overline{\bar{x}} = x & &
 \end{array}$$

2- The AND gate:

The AND gate is called the “all or nothing” gate. The standard logic symbol for the AND gate is drawn in Fig.(2.a). This symbol shows the inputs as x and y . The output is shown as F . This is the symbol for a 2-input AND gate. The Boolean expression of this AND gate is shown in Fig.(2.b) . The truth table for the 2-input AND gate is shown in Fig. (2.c). The inputs are shown as binary digits (bits). Note that only when both inputs x and y are 1 will the output be 1.



Fig(2) The AND gate symbol , Boolean expression and truth table

The Boolean expression reads x AND y equals the output F . The formal laws for the AND function are:

$$x \cdot 1 = x$$

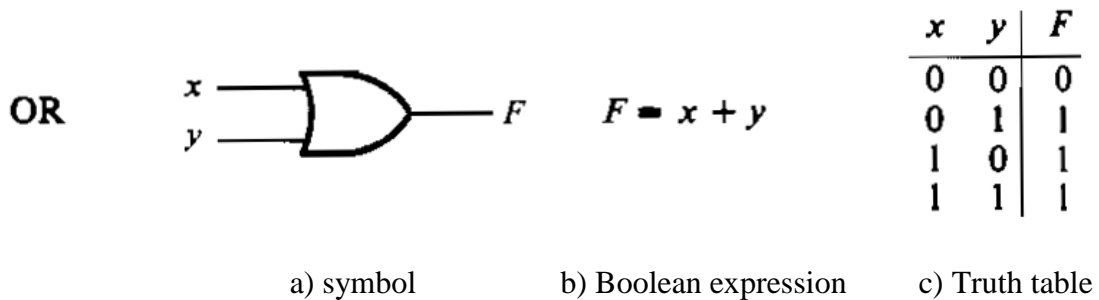
$$x \cdot 0 = 0$$

$$x \cdot x = x$$

$$x \cdot \bar{x} = 0$$

3- The OR gate:

The OR gate is called the “any or all” gate. The standard logic symbol for an OR gate is drawn in Fig (3). The OR gate has two inputs labeled x and y. The output is labeled F. The shorthand Boolean expression for this OR function is given as $x + y = F$. Note that the plus (+) symbol means OR in Boolean algebra. The expression ($x + y = F$) is read as x OR y equals output F. You will note that the plus sign does not mean to add as it does in regular algebra.



Fig(3) The OR gate symbol , Boolean expression and truth table

The formal laws for the OR function are:

$$x + 1 = 1$$

$$x + 0 = x$$

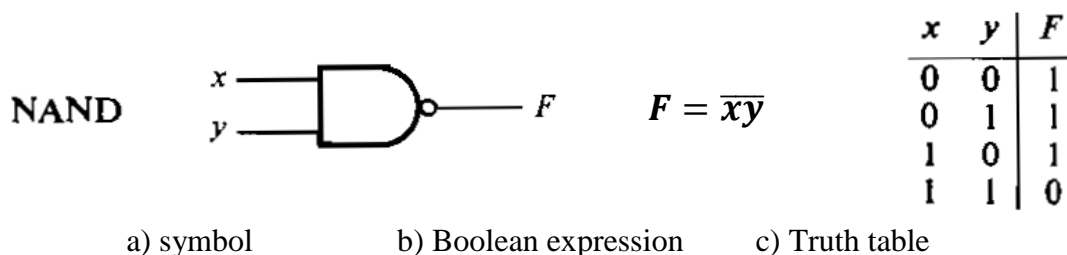
$$x + x = x$$

$$x + \bar{x} = 1$$

4-The NAND gate :

This is implemented from the AND gate with NOT gate , so that it is the complement of the AND gate.

The NAND gate symbol, Boolean expression and truth table are shown in fig(4).




Fig(4) The NAND gate symbol , Boolean expression and truth table

5- The NOR gate :

This is implemented from the OR gate with NOT gate , so that it is the complement of the OR gate.

The NOR gate symbol, Boolean expression and truth table are shown in fig(5).

NOR



$F = \overline{x + y}$

<i>x</i>	<i>y</i>	<i>F</i>
0	0	1
0	1	0
1	0	0
1	1	0


a) symbol b) Boolean expression c) Truth table

Fig(5) The NOR gate symbol , Boolean expression and truth table

6- The Exclusive OR (XOR) gate :

This is implemented from the (OR, NOT, AND) gates , as you can see it's Boolean expression. The XOR gate symbol, Boolean expression and truth table are shown in fig(6).

Exclusive-OR (XOR)



$F = x \oplus y$
 $= x\bar{y} + \bar{x}y$

<i>x</i>	<i>y</i>	<i>F</i>
0	0	0
0	1	1
1	0	1
1	1	0


a) symbol b) Boolean expression c) Truth table

Fig(6) The XOR gate symbol , Boolean expression and truth table

7- The Exclusive NOR (XNOR) gate :

This is the complement of the XOR gate . The XNOR gate symbol, Boolean expression and truth table are shown in fig(7).

Exclusive-NOR or equivalence



$F = x \odot y$
 $= xy + \bar{x}.\bar{y}$

<i>x</i>	<i>y</i>	<i>F</i>
0	0	1
0	1	0
1	0	0
1	1	1

a) symbol b) Boolean expression c) Truth table

Fig(7). The XNOR gate symbol , Boolean expression and truth table

The summary of all logic gates is shown in fig(8)




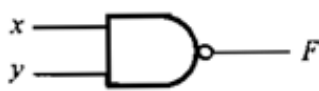
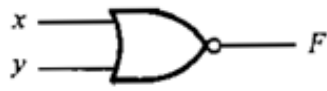

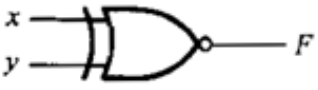
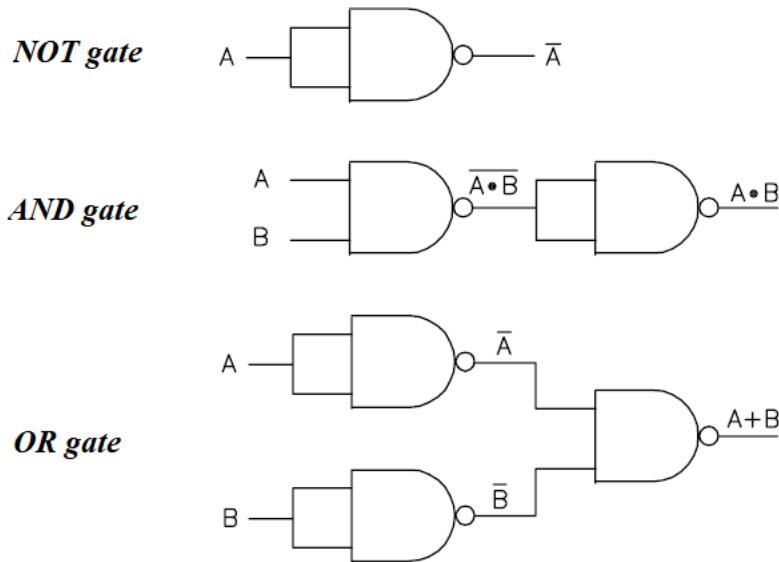
Name	Graphic symbol	Algebraic function	Truth table															
AND		$F = xy$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	y	F	0	0	0	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$F = x + y$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	1
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		$F = \bar{x}$	<table border="1"> <thead> <tr> <th>x</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	F	0	1	1	0									
x	F																	
0	1																	
1	0																	
NAND		$F = \overline{xy}$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	F	0	0	1	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$F = \overline{x + y}$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	0
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
Exclusive-OR (XOR)		$F = x \oplus y$ $= x\bar{y} + \bar{x}y$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	x	y	F	0	0	0	0	1	1	1	0	1	1	1	0
x	y	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR		$F = x \odot y$ $= xy + \bar{x}.\bar{y}$	<table border="1"> <thead> <tr> <th>x</th> <th>y</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	x	y	F	0	0	1	0	1	0	1	0	0	1	1	1
x	y	F																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Fig (8) . The logic gates summary

Universality of NAND & NOR Gates

It is possible to implement any logic expression using only NAND gates and no other type of gate. This is because NAND gates, in the proper combination, can be used to perform each of the Boolean operations OR, AND, and NOT.



In a similar manner, it can be shown that NOR gates can be arranged to implement any of the Boolean operations.

