REPRESENTATION OF LOGIC FUNCTIONS:

Boolean Variables & Truth Tables

Boolean algebra differs in a major way from ordinary algebra in that Boolean constants and variables are allowed to have only two possible values, 0 or 1. Boolean 0 and 1 do not represent actual numbers but instead represent the state of a voltage variable, or what is called its logic level.

Some common representation of 0 and 1 is shown in the following diagram.

In Boolean algebra, there are three basic logic operations: AND, OR, and NOT.

These logic gates are digital circuits constructed from diodes, transistors, and resistors connected in such a way that the circuit output is the result of a basic logic operation

(OR, AND, NOT) performed on the inputs.

Truth Table

A truth table is a means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs.

In the following two-input logic circuit, the table lists all possible combinations of logic levels present at inputs A and B along with the corresponding output level X.

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

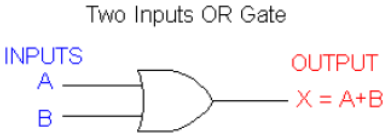
When either input A OR B is 1, the output X is 1. Therefore the "?" in the box is an OR gate.

OR Operation

The expression X = A + B reads as "X equals A OR B". The + sign stands for the OR operation, not for ordinary addition.

The OR operation produces a result of 1 when any of the input variable is 1.

The OR operation produces a result of 0 only when all the input variables are 0.

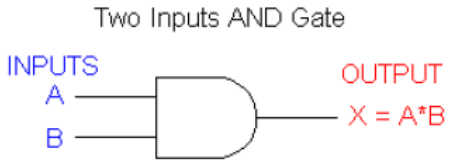


A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

AND Operation

The expression X = A * B reads as "X equals A AND B".

The multiplication sign stands for the AND operation, same for ordinary multiplication of 1s and 0s.The AND operation produces a result of 1 occurs only for the single case when all of the input variables are 1.The output is 0 for any case where one or more inputs are 0



A	B	X = A*B
0	0	0
0	1	0
1	0	0
1	1	1

NOT Operation

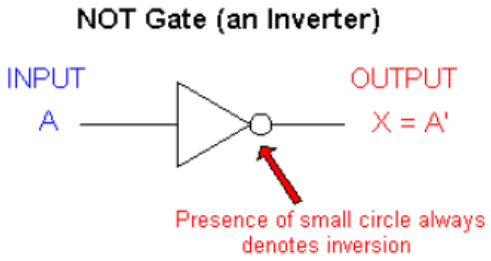
The NOT operation is unlike the OR and AND operations in that it can be performed on a single input variable. For example, if the variable A is subjected to the NOT operation, the result x can be expressed as x = A' where the prime (') represents the

NOT operation. This expression is read as:

x equals NOT A

x equals the inverse of A

x equals the complement of A



A	X = A'
0	1
1	0

Each of these is in common usage and all indicate that the logic value of x = A' is opposite to the logic value of A. The truth table of the NOT operation is as follows:

1'=0 because NOT 1 is 0

0' = 1 because NOT 0 is 1

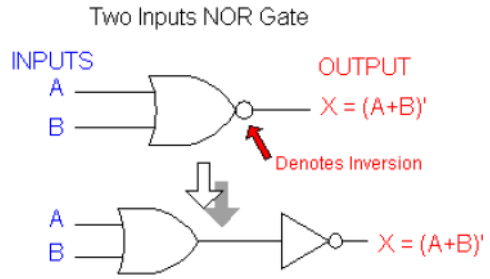
The NOT operation is also referred to as inversion or complementation, and these terms are used interchangeably.

NOR Operation

NOR and NAND gates are used extensively in digital circuitry. These gates combine the basic operations AND, OR and NOT, which make it relatively easy to describe then using Boolean algebra.

NOR gate symbol is the same as the OR gate symbol *except* that it has a small circle on the output. This small circle represents the inversion operation. Therefore the output expression of the two input NOR gate is:

X = (A + B)'

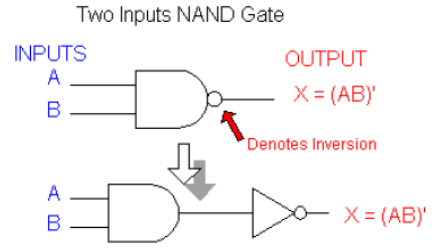


INPUTS		OR	NOR
A	B	X = A+B	X= (A+B)`
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NAND Operation

NAND gate symbol is the same as the AND gate symbol *except* that it has a small circle on the output. This small circle represents the inversion operation. Therefore the output expression of the two input NAND gate is:

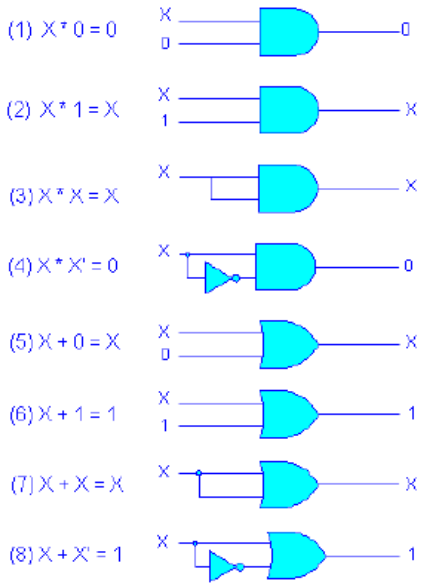
X = (AB)'



INPUTS		AND	NAND
A	B	X = AB	X= (AB)`
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Boolean Theorems

Investigating the various Boolean theorems (rules) can help us to simplify logic expressions and logic circuits.



Multivariable Theorems

The theorems presented below involve more than one variable:

(9) x + y = y + x *(commutative law)*

(10) x * y = y * x *(commutative law)*

(11) x+ (y+z) = (x+y) +z = x+y+z *(associative law)*

(12) x (yz) = (xy) z = xyz *(associative law)*

(13a) x (y+z) = xy + xz

(13b) (w+x)(y+z) = wy + xy + wz + xz

(14) x + xy = x [proof see below]

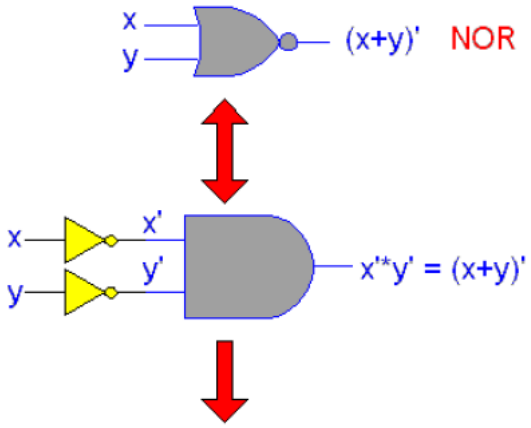
(15) x + x'y = x + y

DeMorgan's Theorem

DeMorgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted. The two theorems are:

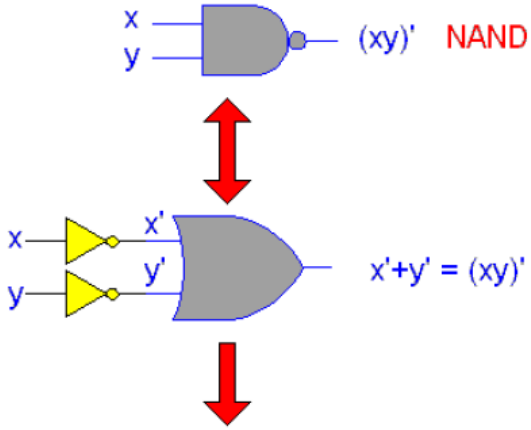
(16) (x+y)' = x' * y'

Theorem (16) says that when the OR sum of two variables is inverted, this is the same as inverting each variable individually and then ANDing these inverted variables.



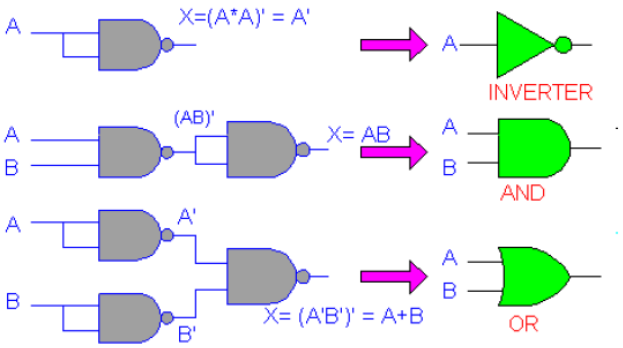
(17) (x*y)' = x' + y'

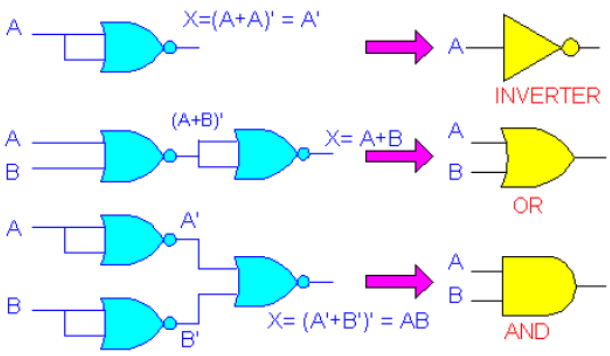
Theorem (17) says that when the AND product of two variables is inverted, this is the same as inverting each variable individually and then ORing them.



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 INVERT.





Alternate Logic Gate Representations

The left side of the illustration shows the standard symbol for each logic gate, and the right side shows the alternate symbol. The alternate symbol for each gate is obtained from the standard symbol by doing the following:

1. Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles, and by removing bubbles that are already there.

2. Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed.)

Boolean Function Truth Table

Boolean function can be represented by truth table as well. If the function has n variables, its truth table will have 2n rows

e.g. f = x • y + x • z'

f has 3 variables so 23 combinations

f is 1 when the expression is evaluated to 1 otherwise it is 0.

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