Introduction to Boolean Algebra and Logic Circuits

Course No: E01-002

Credit: 1 PDH

Jeffrey Cwalinski, P.E.

Continuing Education and Development, Inc.
22 Stonewall Court
Woodcliff Lake, NJ 07677

P: (877) 322-5800
info@cedengineering.com

Introduction to Boolean Algebra and Logic Circuits

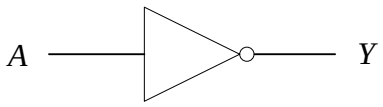
I. Boolean Variables

Boolean variables are associated with the Binary Number system and are useful in the development of equations to determine an outcome based on the occurrence of events. Boolean variables take on one of two (2) values: True or False. The True value is also called "1" or On or High; and the False value is also called "0" or Off or Low. These values can be associated with states of electrical parameters such as voltage and light. For example, when using Transistor Transistor Logic (TTL) circuits the True value generally corresponds to a voltage level of 2 volts to the supply voltage, Vcc, which is approximately 5 volts; and False generally corresponds to 0 volts to 0.8 volts. The transmission of Boolean values using fiber optic components can define a True as a condition where a laser diode, light emitting diode, or other light source is illuminating and a False as a condition where the light source is non-illuminating. Boolean values are also associated with a physical input such as a position of a switch or set of relay contacts. A True can be associated with a closed switch or closed set of relay contacts, while a False can be associated with an open switch or open set of relay contacts. Boolean values can also be associated with a physical output such as a relay coil, where a True can be associated with an energized relay coil and a False can be associated with a de-energized relay coil.

II. Boolean Operators

Boolean equations use Boolean operators, also called gates. The basic Boolean operators are NOT, AND, and OR. Diagrams representing each operator, equation for the operation, and input/output relationships are shown below.

A. NOT Operator



NOT

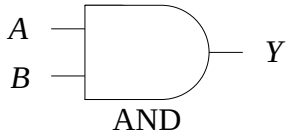
$Y = \overline{A}$

A	Y
1	0
0	1

The table above is called a Truth Table. It defines the relationship between the inputs, which are usually represented by a letter at the beginning of the alphabet, and the output, which is usually represented by

the letter *Y*. The output for the NOT operator is the negated value, or the complement, of the input. It is said the variable *Y* is equal to not *A*. In the equation the line over the top of the variable on the right side of the equal sign indicates the complement. On a diagram it is the circle at the end which signifies the complement. Alone the circle may not be visible in a diagram and could be easily overlooked. To avoid this it is placed at the end of the triangular portion, which traditionally represents a buffer. As shown above the circle may be placed at the output of the operator to represent the complement of the output. It may also be placed at the input of the operator to represent the complement of the input variable.

B. AND Operator



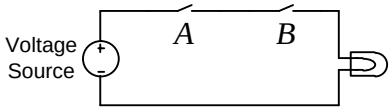
$$Y = A \bullet B$$

The dot between the *A* and the *B* is usually omitted and the equation is rewritten as:

$$Y = AB$$

A	*B*	*Y*
0	0	0
0	1	0
1	0	0
1	1	1

The output is a "1" when *A* and *B* are "1". Otherwise, the output is a "0". It is said the *Y* equals *A* and *B*. The electrical circuit below illustrates the concept.

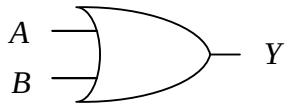


The light is illuminated (Boolean value of "1") when switch *A* is closed (Boolean value of "1") and when switch *B* is closed (Boolean value of "1"). When any or both switches are opened (Boolean value of "0") the light is extinguished (Boolean value of "0").

The AND operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change; that is the output will be "1" when all inputs are "1". It will be "0" otherwise.

The AND function is similar to algebraic multiplication; that is "0" multiplied by any quantity yields "0", and "1" multiplied by itself yields "1".

C. OR Operator

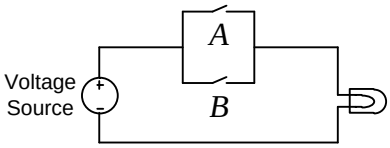


OR

$$Y = A + B$$

A	*B*	*Y*
0	0	0
0	1	1
1	0	1
1	1	1

The output is a "1" when *A* or *B* or both are "1". Otherwise, the output is a "0". It is said the *Y* equals *A* or *B*. The electrical circuit below illustrates the concept.



The light is illuminated (Boolean value of "1") when switch *A* is closed (Boolean value of "1") or when switch *B* is closed (Boolean value of "1"). When both switches are opened (Boolean value of "0") the light is extinguished (Boolean value of "0").

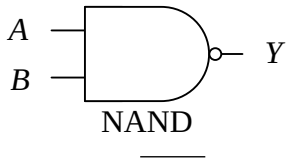
The OR operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change; that is the output will be "1" when one or more inputs are "1". It will be "0" otherwise.

The OR function is similar to algebraic addition; that is "0" added to itself yields "0", "0" added to "1" yields "1", the only exception is "1" added to itself which, in algebra, yields "2", but in Boolean algebra 1 + 1 yields "1".

From the three (3) basic Boolean operators described above, other operators that are commonly used are derived. They are the NAND

operator, the NOR operator, Exclusive OR (XOR) operator, and the NOT Exclusive OR (NXOR) operator.

D. NAND Operator

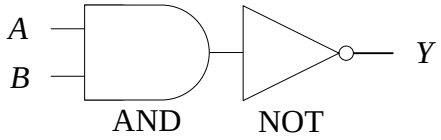


$$Y = \overline{A \bullet B}$$

The dot between the *A* and *B* is usually omitted and the equation is written as $Y = \overline{AB}$.

A	*B*	*Y*
0	0	1
0	1	1
1	0	1
1	1	0

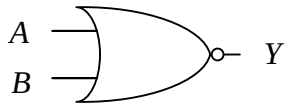
The NAND operator is constructed of the AND operator and the NOT operator, as shown below.



The output is a "0" when *A* and *B* are "1". Otherwise, the output is a "1". This operator is the AND operation followed by the NOT operation. It is said the *Y* equals not *A* and *B*. The circle at the output performs the complement. Note that it is incorrect to say *Y* equals not *A* and not *B*; this is a different Boolean equation, the variables are being complemented prior to the AND operation.

The NAND operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change; that is the output will be "1" when one or more inputs are equal to "0". It will be "0" otherwise.

E. NOR Operator

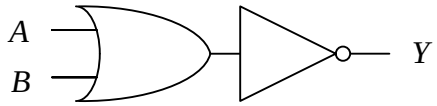


NOR

$$Y = \overline{A + B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

The NOR operator is constructed of the OR operator and the NOT operator, as shown below.

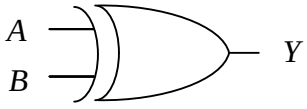


OR NOT

The output is a "0" when *A* or *B* or both are "1". Otherwise, the output is a "1". This operator is the OR operation followed by the NOT operation. It is said the *Y* equals not *A* or *B*. The circle at the output performs the complement. Note that it is incorrect to say *Y* equals not *A* or not *B*; this is a different Boolean equation, the variables are being complemented prior to the OR operation.

The NOR operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change; that is the output will be "1" when all inputs are "0". It will be "0" otherwise.

F. XOR Operator

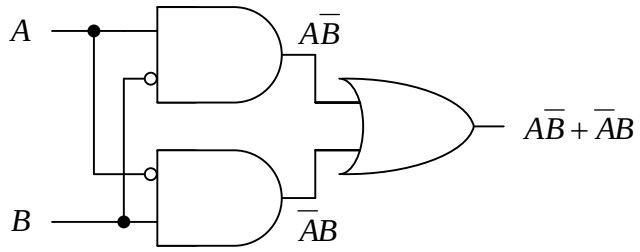


XOR

$$Y = A \oplus B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

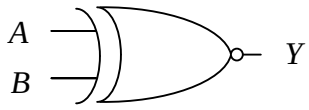
Using basic Boolean operators the logic for the XOR operator is drawn below.



The output is a "1" when *A* and *B* are of different values. The output is "0" when *A* and *B* are of the same value. It is said the *Y* equals *A* exclusive or'd with *B*.

The XOR operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change, that is the output will be "1" when one and only one input is of a different value. It will be "0" otherwise. Or, put in another way, if there are N inputs, then the output will be "1" if N-1 inputs are of the same value.

G. NXOR Operator



NXOR

$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

The output is a "1" when *A* and *B* are of the same value. The output is "0" when *A* and *B* are of different values. It is said the *Y* equals *A* exclusive nor'd with *B*. Note that it is incorrect to say *Y* equals not *A* exclusive

nor'd with not *B*; this is a different Boolean equation, the variables are being complemented prior to the NXOR operation.

The NXOR operator above is a two (2) input operator. It can be expanded to any number of inputs. The function will not change; that is the output will be "0" when one and only one input is of a different value. It will be "1" otherwise. Or, put in another way, if there are N inputs, then the output will be "0" if N-1 inputs are of the same value.

III. Analysis of Boolean Equations:

A. Format

Boolean equations have the same format as ordinary algebraic equations; an unknown equals a combination of variables. Some examples are:

1. $f(A,B,C,D) = Y = ABC + ABD + \overline{A}B\overline{C} + CD$, Equation 1
2. $f(A,B,C) = Y = A(B + C)$, Equation 2
3. $f(A,B,C,D) = Y = \overline{(AB + D)(AC + B) + ((B + D)A\overline{C})}$, Equation 3

B. Order of Operations

The evaluation of a Boolean equation occurs left to right: negation of single terms occurs first, followed by the AND operation, then followed by the OR operation. This sequence is repeated until the equation is evaluated to be a "0" or a "1". The order of operations may be changed by the use of parentheses, in which case the operation within the parentheses is to be performed first. This is similar to ordinary algebra. Equations 1, 2, and 3 above will be evaluated using the following values, *A*=1, *B*=1, *C*=0, and *D*=1.

For Equation 1 the *ABC* term evaluates to $1 \bullet 1 \bullet 0 = 0$, the *ABD* term evaluates to $1 \bullet 1 \bullet 1 = 1$, the $\overline{A}B\overline{C}$ term evaluates to $\overline{1} \bullet 1 \bullet \overline{0} = 0 \bullet 1 \bullet 1 = 0$, (note the complement occurs before the AND operation), the *CD* term evaluates to $0 \bullet 1 = 0$. Substituting in for ABC, ABD, $\overline{A}B\overline{C}$, and CD yields 0 + 1 + 0 + 0, respectively, which results in a value of "1".

For Equation 2 the expression inside the parentheses is evaluated first then an AND operation is performed with the *A* term. The $(B + C)$ evaluates to (1 + 0) = 1. Substituting in for *A* and $(B + C)$ yields $1 \bullet 1$, respectively, which results in a value of "1".

For Equation 3 the expressions inside the parentheses are evaluated first. The $(AB + D)$ evaluates to $1 \bullet 1 + 1 = 1$, the $(AC + B)$ evaluates to

$1 \bullet 0 + 1 = 1$. For the $((B + D)A\overline{C})$ term the inner most parentheses are evaluated first, $(B + D)$ evaluates to (1 + 1) = 1. The $((B + D)A\overline{C})$ term can now be evaluated, $1 \bullet 1 \bullet \overline{0} = 1 \bullet 1 \bullet 1 = 1$. The complement of *C* is performed prior to the AND operation. The results are placed in the equation, evaluated, then the complement of the single term is performed, $\overline{1+1+1} = \overline{1} = 0$.

C. Basic Theorems

There are eight (8) single variable identities that can be used to simplify Boolean equations:

1. $A + 0 = A$, a variable OR'd with "0" equals itself
2. $A + 1 = 1$, a variable OR'd with "1" equals "1"
3. $A + A = A$, a variable OR'd with itself equals itself
4. $A + \overline{A} = 1$, a variable OR'd with its complement equals "1"
5. $A \bullet 1 = A$, a variable AND'd with "1" equals itself
6. $A \bullet 0 = 0$, a variable AND'd with "0" equals "0"
7. $A \bullet A = A$, a variable AND'd with itself equals itself
8. $A \bullet \overline{A} = 0$, a variable AND'd with its complement equals "0"

The commutative, associative, and distributive properties apply to Boolean algebra:

1. $A \bullet B = B \bullet A$, and $A + B = B + A$, communicative property
2. $A \bullet (B \bullet C) = (A \bullet B) \bullet C = A \bullet B \bullet C$, and
   $A + (B + C) = (A + B) + C = A + B + C$, associative property
3. $A \bullet (B + C) = (A \bullet B) + (A \bullet C)$, and
   $A + (B \bullet C) = (A + B) \bullet (A + C)$, distributive property

D. DeMorgan's Theorem

DeMorgan's Theorem is given by:

1. $\overline{A \bullet B \bullet C \bullet \cdots} = \overline{A} + \overline{B} + \overline{C} + \cdots$ and
2. $\overline{A + B + C + \cdots} = \overline{A} \bullet \overline{B} \bullet \overline{C} \cdots$.

It can be stated as, for Case 1, the complement of an AND of variables is equal to the OR of the complements of the individual variables and, for Case 2, the complement of the OR of variables is equal to the AND of the complements of the individual variables.

Using Equation 3 in Section III as an example:

$$f(A,B,C,D) = Y = \overline{(AB + D)(AC + B) + ((B + D)\overline{AC})}.$$

Start by expanding the $(AB + D)(AC + B)$ term using algebraic techniques, and the Basic Theorems described above, which results in:

$$ABC + AB + ACD + BD.$$

Expand the $((B + D)\overline{AC})$ term, which results in: $AB\overline{C} + A\overline{C}D$.

The equation is now: $Y = \overline{ABC + AB + ACD + BD + AB\overline{C} + A\overline{C}D}$.

Applying DeMorgan's Theorem results in:

$$Y = \overline{ABC} \bullet \overline{AB} \bullet \overline{ACD} \bullet \overline{BD} \bullet \overline{AB\overline{C}} \bullet \overline{A\overline{C}D}.$$

DeMorgan's Theorem can be applied again to the individual terms. For example, $\overline{ABC} = \overline{A} + \overline{B} + \overline{C}$.

IV. Analysis of Logic Circuits

Logic circuits consists of multiple gates, which may have multiple inputs, connected together to perform a useful function. They are used in the design of control systems and other digital systems.

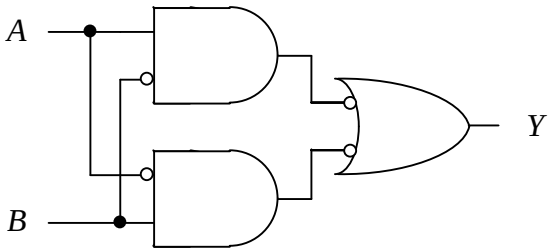
Examples would be the Exclusive OR and Exclusive NOR in Sections II.F and II.G.

The technique for the solution of logic circuits involves starting at the input of the logic circuit and determining an output equation for each gate. Then use that output equation for the input of the next gate and determine a subsequent output equation. Repeat this process until the final output equation is determined.

An example is shown below.

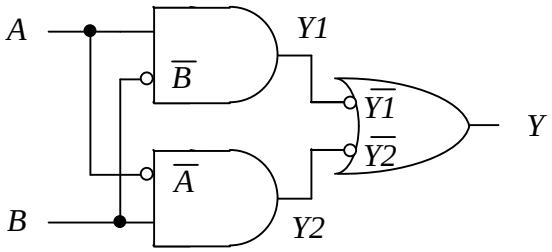
The equation for the output of logic gates below is:

a. $Y = AB$
b. $Y = \overline{A\bar{B}} + \overline{\bar{A}B}$
c. $Y = \bar{A}B$
d. $Y = A\bar{B} + \bar{A}B$



Answer: The correct answer is b

In solving logic gates it is helpful to define outputs at each gate, then solve for an output, use that output as the input for the subsequent gate, solve for the output of that subsequent gate, and repeat this process until the final equation is solved. A revised diagram with outputs at each gate, *Y1* and *Y2*, is shown below.



It should be noted that the circle indicates that the input is complemented and the complemented input is shown inside the gate for convenience. First generate the equation for *Y*1, $Y1 = A\bar{B}$. Second, generate the equation for *Y2*, $Y2 = \bar{A}B$. Third, generate the equation for *Y*, $Y = \overline{Y1} + \overline{Y2}$. Finally, substitute the equations for *Y*1 and *Y*2, $Y = \overline{A\bar{B}} + \overline{\bar{A}B}$.