ElectronicsTutorials

BOOLEAN ALGEBRA

For Students, Professionals and Beyond

eBook 20

ASPENCORE

WWW.ELECTRONICS-TUTORIALS.WS

TABLE OF
CONTENTS

Our Terms of Use

This **Basic Electronics Tutorials eBook** is focused on Boolean algebra and switching theory with the information presented within this ebook provided "as-is" for general information purposes only.

This free e-book is presented as general information and study reference guide for the education of its readers who wish to learn Electronics. While every effort and reasonable care has been taken with respect to the accuracy of the information given herein, the author makes no representations or warranties of any kind, expressed or implied, about the completeness, accuracy, omission of errors, reliability, or suitability with respect to the information or related graphics contained within this e-book for any purpose.

As such it is provided for personal use only and is not intended to address your particular problem or requirement. Any reliance you place on such information is therefore strictly at your own risk. We can not and do not offer any specific technical advice, troubleshooting assistance or solutions to your individual needs.

We hope you find this guide useful and enlightening. For more information about any of the topics covered herein please visit our online website at:

www.electronics-tutorials.ws

1. Introduction

In 1854, *George Boole* performed an investigation into the "laws of thought" which were based around a simplified version of the "group" or "set" theory. From his investigations *Boolean Algebra* was developed. **Boolean Algebra** deals mainly with the theory that logic and set operations are either "TRUE" or "FALSE" but cannot be both at the same time.

Thus, Boolean Algebra is a form of mathematics based on symbolic logic. It has its own set of rules or laws, which we can use to analyse digital circuits and logic gates in an attempt to reduce the number of logical devices required (and therefore cost).

Each variable (input or output) used in any Boolean Algebra expression can only have one of two distinct logic values, a "0" and a "1", commonly referred to as "truth values" (hence the term truth table). Then Boolean algebra is an algebraic system consisting of a set of variables (0 and 1) and implementation using operations that we call *Boolean functions*.

But a Boolean expression can have an infinite number of variables within it, all labelled individually to represent the individual inputs for that expression. For example, we could use variables A, B, C etc. to give us a logical expression of A+B = C. But again each variable can ONLY have a value of logic-0 or a logic-1.

Switching Theory allows us to understand the operation and relationship between Boolean Algebra and two-level logic functions with regards to digital logic gates whose operation can best be described with a Boolean expression.

> *Boolean Algebra is a form of logic algebra invented by George Boole*

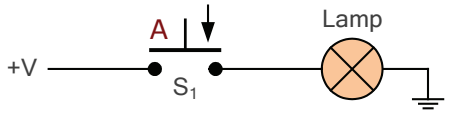
Switching theory can be used to further develop the theoretical knowledge and concepts of digital circuits when viewed as a complex network of switching elements that switch between zero and one to produce a defined output state or condition.

These logical states can be presented using electromechanical contacts in the form of switches or relays as a logic circuit element. The implementation of switching functions in digital logic circuits is nothing new, but it can give us a better understanding of how a single digital logic gate works.

2. The Switching Theory of a Single Switch

You may think that a switch is, well just a switch, that can be used to turn a light "ON" or "OFF". But a switch can also be a complex electromechanical device used as a two-value element performing logical operations. In which its two physical states, open and closed, correspond to the logic values of 0 and 1 respectively as shown in Figure 1.

Figure 1. A Single Switching Circuit



Here in this single switch example of Figure 1. If switch S_1 is not-pressed and therefore open, the lamp will be OFF. Likewise, if switch S_1 is pressed closing it, the lamp will be ON. Under normal steady state conditions, the switch is permanently open so the lamp is always OFF.

We can develop this switching theory idea further by saying that when the lamp is ON or illuminated, its switching algebra variable will be a logic-1, and when the lamp is OFF and not illuminated, its switching algebra variable will be a logic-0.

Thus we can use switching algebra to describe the operation of the circuit containing the switch in Figure 1. If we label the normally-open switch as a Boolean variable with the letter "A", when the switch is open, that is "A" is not-pressed, we can define the value of "A" as being "0". Again, when the switch is closed, that is "A" is pressed, we can define the value of "A" as being "1". This switching algebra assumption is true for ALL normally-open switch configurations.

Table 1. Truth Table of Figure 1.

Switch (A)	Lamp (L)
0	0
1	1
L = A	

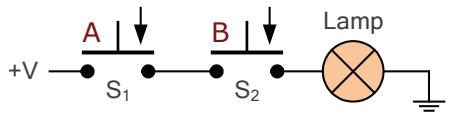
Thus, when the switch is pressed (activated) the lamp, (L) is ON, so A = 1 and L = 1. When the switch is not-pressed (unactivated) the lamp (L) is OFF, so A = 0 and L = 0.

Therefore, we can correctly say that the output L = A for the switching theory of the lamp as shown in the truth table for Figure 1.

3. The Switching Theory of Series Switches

We have seen in Figure 1. that the lamp circuit above can be controlled using a single switch, S_1 and when S_1 is closed (pressed) the lamp is "ON" representing a logic-1 condition. But what if we added a second switch in series with S_1, how would that affect the switching function of the circuit and the illumination of the lamp.

Figure 2. Two Switches in Series



The switching circuit of Figure 2. consists of two switches connected in series to a lamp across a voltage source, +V. To distinguish the operation of each individual switch, we shall label the first switch, S_1 with the letter "A", and label second switch, S_2 with the letter "B". Thus switches A and B represent two different Boolean variables.

When either switch is open, that is not-pressed, we can define the value of A as being at logic-0 and B as also being at logic-0. Likewise, when either switch is closed or pressed, we can define the value of A as being logic-1 or B as being at logic-1. The same as before.

As there are two switches, A and B, we can see that there are four possible combinations of the Boolean variables A and B to illuminate the lamp. For example, A is open and B is closed, or A is closed and B is open, or both A and B are open or closed at the same time. Then we can define these operations in the following truth table of Table 2.

Table 2. Switching Truth Table of Figure 2.

Switch (A)	Switch (B)	Lamp (L)
0	0	0
0	1	0
1	0	0
1	1	**1**
L = A AND B		

Table 2. shows that the lamp will only be "ON" and illuminated when BOTH switch, A AND switch, B are pressed and closed as pressing only one switch on its own will not illuminate the lamp.

This proves that when two Boolean variables A and B are connected in series, the only condition that will illuminate the lamp is when both switches are closed giving the Boolean expression of: L = A AND B.

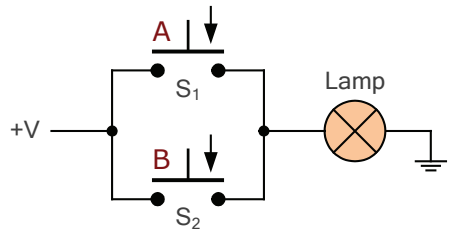
In Boolean Algebra terms, this expression is that of the **AND** function which is denoted by a single dot or full stop symbol, (**.**) between the variables giving us the Boolean expression of: L = A.B.

Thus when switches are connected together in series their operation and switching theory is the same as for the digital logic "**AND**" gate because if both inputs are logic-1, then the output is at logic-1, otherwise the output is at logic-0.

4. The Switching Theory of Parallel Switches

If we now connect the two switches, S_1 and S_2 in parallel as shown, how would this arrangement affect the switching function of the circuit and the illumination of the lamp.

Figure 3. Two Switches in Parallel



The switching circuit now consists of the two switches in parallel with the voltage source and the lamp.

As before, with two switches, A and B, there are four possible combinations of the Boolean variables required to illuminate the lamp. That is A is open and B is closed, or A is closed and B is open, both A and B are open, or both closed at the same time. As shown in the following truth table of Table 3.

Table 3. Switching Truth Table of Figure 3.

Switch (A)	Switch (B)	Lamp (L)
0	0	0
0	1	**1**
1	0	**1**
1	1	**1**
L = A OR B		

Table 3. shows that the lamp will only be "ON" and illuminated when EITHER switch, A OR switch, B are pressed and closed.

This therefore shows that when the two switches A and B are connected in parallel, the will lamp illuminate when any one of the switches, or both are closed. This gives the Boolean expression of: L = A OR B.

In Boolean Algebra terms, this expression is that of the OR function which is denoted by an addition or plus sign, (+) between the variables giving us the expression of: L = A+B.

Thus when switches are connected together in parallel their switching theory is the same as for the digital logic "OR" gate because if both inputs are logic-0, then the output is 0, otherwise the output is at logic-1.

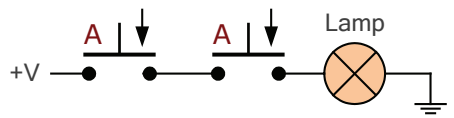
5. The Idempotent Law of Switches

Thus far we have seen how to connect two switches together either in series or parallel to illuminate a lamp. But what if the two switches representing a Boolean AND function or the Boolean OR function (operations of multiplication and sum) are of the same single Boolean variable, A.

In Boolean Algebra there are various laws and theorems which can be used to define the mathematics of various logic circuits. One such theorem in which combinations of a single variable are connected with itself is known by the name of the *idempotent law*.

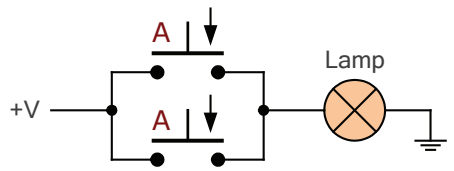
Idempotent Laws used in switching theory states that AND-ing or OR-ing a variable with itself will produce the original variable. For example, variable "A" *AND'ed* with "A" can be reduced to a single element, A. Likewise the Boolean variable "A" *OR'ed* with itself will also result in a single element, A. Thus allowing us to simplify our switching circuits.

Figure 4. Idempotent Law of the AND Function



A . A = A

Figure 5. Idempotent Law of the OR Function



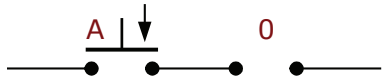
A + A = A

6. The Annulment Law of Switches

Boolean algebra expressions can also consist of operations using 0 and 1 in the form of postulates. While not Boolean laws in their own right, they can still be used in the simplification of Boolean Expressions.

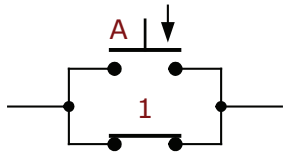
The **Annulment Law** states that anything *AND'ed* with a 0 will always equal 0 as shown in Figure 6. Also, anything *OR'ed* with a 1 will always be equal to a 1 as shown in Figure 7.

Figure 6. Annulment Law of the AND Function



A . 0 = 0

Figure 7. Annulment Law of the OR Function

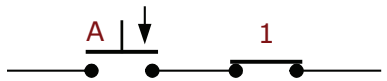


A + 1 = 1

7. The Identity Law of Switches

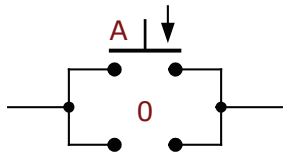
The **Identity Law** states that anything *AND'ed* with a 1 will always equal itself as shown in Figure 8. While anything *OR'ed* with a 0 will always be equal to itself as shown in Figure 9.

Figure 8. Identity Law of the AND Function



A . 1 = A

Figure 9. Identity Law of the OR Function



A + 0 = A

As well as identity **AND** and **OR** operations consisting of a single Boolean variable with the property of an element **0** or a **1**, these two binary postulates (postulates are elements taken as facts, so no proof is required) can also be used together and simplified as shown.

AND Function Identity Elements

0.0 = 0 Thus a **0** *AND'ed* with itself is always equal to **0**

1.1 = 1 Thus a **1** *AND'ed* with itself is always equal to **1**

1.0 = 0 Thus a **1** *AND'ed* with an element **0** is equal to **0**

OR Function Identity Elements

0+0 = 0 Thus a **0** *OR'ed* with itself is always equal to **0**

1+1 = 1 Thus a **1** *OR'ed* with itself is always equal to **1**

1+0 = 1 Thus a **1** *OR'ed* with an element **0** is equal to **1**

Note that a logic element "0" can be viewed as an *additive identity*. While a logic element "1" can be viewed as a *multiplicative identity*.

8. The Complement Law of Switches

Complements (or negation, or inversion) of a Boolean variable also exists to produce the opposite of its value. The complement of a variable is represented by the **NOT** operation which is commonly denoted with a bar (**–**) symbol above it.

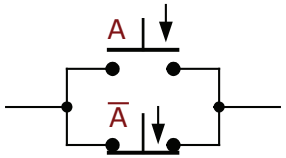
So for example, the complement of the Boolean variable **A** would be $\overline{A}$. Likewise, the complement of **B** would therefore be $\overline{B}$, and so on. So, if the variable **A** has a value of **0**, then the complement of **A** would be **1**, and the complement of **1** would be **0**. Note that $\overline{A}$ and **A'** can both be used interchangeably to represent the complement of a variable.

The **Complement Law** states that any variable *AND'ed* with its complement will always be equal to **0** as shown in Figure 10. If a variable is *OR'ed* with its complement, it will always be equal to **1** as shown in Figure 11.

Figure 10. Complement Law of the AND Function

$A.\overline{A} = 0$

Figure 11. Complement Law of the OR Function



$A+\overline{A} = 1$

The Boolean Law of complementation, or inversion, $A.\overline{A} = 0$ and $A+\overline{A} = 1$ is an important law to both understand an use in the simplification of Boolean expressions.

9. The Double Complement Law

As well as producing one complement of a Boolean variable, it is also possible during the simplification of long Boolean expressions to end up with a double complement, or double negation (or more) situation.

The **Double Complement Law** states that if you complement or invert a variable two times, you will end up with the variables original Boolean value. That is the complement of the complemented of a variable is always equal to the variable, as two negatives make a positive since it is analogous to multiplying by **-1** in ordinary or real-number algebra.

Thus if we take the variable **A** and complement (or invert) it once, we get **not-A** presented as $\overline{A}$. If we then take $\overline{A}$ and complement it again, we get **NOT**(**not-A**) presented as $\overline{\overline{A}}$ which represents a double complement, which is the original variable as shown.

If **A = 0**, then $\overline{A} = 1$ therefore $\overline{\overline{A}} = 0 = A$. Likewise, if **A = 1**, then $\overline{A} = 0$ therefore $\overline{\overline{A}} = 1 = A$

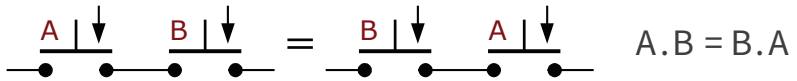
That is: $\overline{\overline{A}}$ is always equal to **A**, or could also be: $A = \overline{\overline{A}}$. Note that the complement of any given Boolean variable can be taken repeatedly, but will always result in the original value every even number of times it is complemented (inverted).

10. The Commutative Law

The **Commutative Law** states that the order of application of two separate variables is not important as it will not affect the result of an AND or OR operation.

Since A and B in series is identical in all respects as B and A in series. Thus, the order in which two (or more) variables are *AND'ed* together makes no difference as their Boolean multiplication is commutative as shown in Figure 11.

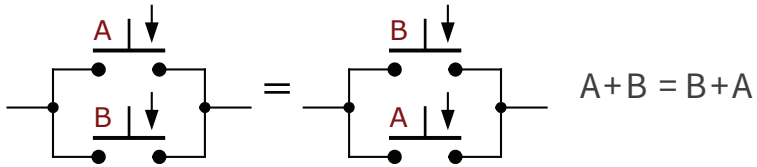
Figure 11. Commutative Law of the AND Function



$$A.B = B.A$$

Likewise, if A and B are in parallel, it is the same as B and A are in parallel as the order in which two (or more) variables are *OR'ed* makes no difference since their Boolean addition is commutative as shown in Figure 12.

Figure 12. Commutative Law of the OR Function



$$A+B = B+A$$

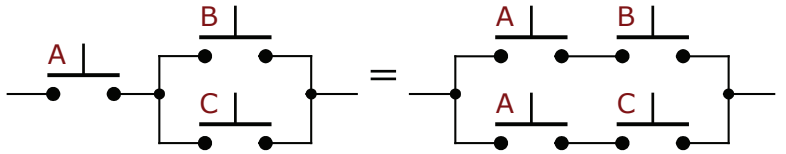
11. The Distributive Law

While some of the previous laws of Boolean algebra may have seemed rather self-evident, or obvious. The distributive laws of AND and OR are not quite so clear.

The **Distributive Law** permits the multiplying or factoring out of a Boolean expression in much the same way as for ordinary real-number algebra. For example, in standard arithmetic the expression 3(2 + 4) is equal to saying (3x2)+(3x4) as the final result of 18 is the same for both equations since the multiplication operator distributes over the addition operator.

Similarly, in Boolean Algebra, the AND (logical multiplication) operator will distribute over an OR (logical addition) operator. So, for example: A.(B+C) is the same as saying A.B + A.C as shown in Figure 13.

Figure 13. Distributive Law of the AND Operator

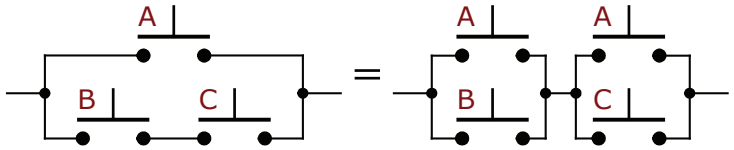


$$A(B+C) = AB+AC$$

Since both switching networks produce the same result, they are therefore electrically equivalent. Thus we have distributed the AND operator with the two OR operators and then *OR-ing* at the end.

Generally, the multiplication AND operator would take priority mathematically over the addition OR operator. However, in Boolean Algebra the OR operator can still distribute evenly over an AND operator. For example: A+(B.C) would become (A+B).(A+C) as shown in Figure 14.

Figure 14. Distributive Law of the OR Operator



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

Here in Figure 14. we have distributed the OR operator with the AND operator for B and C and then *AND-ing* at the end producing two identical switching networks.

Then factorisation of any Boolean expression can be achieved by the application of the distribution law allowing an expression to be expanded out by multiplying one term by the other term.

12. The Absorptive Law

The **Absorptive Law** does not have an exact equivalent in conventional real-number algebra but enables the reduction of a complicated Boolean expression to a much shorter or simpler one by absorbing one or more like terms. In other words, the absorption law allows for the manipulation and simplification of Boolean expressions by absorbing some terms into other terms, thereby making them easier to understand and analyse.

While the law of absorption (sometimes known as the *Redundancy Law*) is the result of the application of several previous laws, nevertheless it can still be used to good effect.

For example assume the following Boolean expressions of: A+(A.B) and A(A+B).

$$A + (A.B) = A.1 + A.B = A(1+B) = A.1 = A$$

$$A(A + B) = A.A + A.B = A + A.B = A(1 + B) = A.1 = A$$

Other laws of absorption in Boolean algebra include:

$$A(\overline{A} + B) = A.\overline{A} + A.B = 0 + A.B = A.B$$

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

$$A + (\overline{A}.B) = A + A.B + \overline{A}.B = A + B(A + \overline{A}) = A + B.1 = A + B$$

Then we can see that in all cases, the output is a simplification of the initial expression reducing them to their simplest form. The advantage of the absorption law is that, it helps identify and remove redundant terms. So, if a smaller Boolean term appears within a much larger expression, then the larger term effectively becomes redundant as shown.

$$A + A.\overline{B} + A.\overline{B}.\overline{C} + A.\overline{B}.C + A.B.\overline{C} = A$$

Here, the letter "A" is repeated in each term reducing down to just A.

In all cases, if necessary the law of absorption can be proven by means of truth tables.

13. The Associative Law

The **Associative Law** is another laws which governs the manipulation and simplification of Boolean expressions. It allows the removal of brackets from an expression and the regrouping of the variables without changing the meaning of the expression.

There are two versions of the Boolean associative law: one for logical OR operations and one for logical AND operations. Then we can associate terms by using the AND or OR function.

For example, If we have a two term expression A.B, and we want to add a third term to produce A.B.C. Since A and the B are already grouped or associated, all we have to do is add C to the result.

12.1 Associative Law for the Logical AND (.) Function

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

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

Note that the associate law of multiplication is, by default associative, which means that the product of three or more terms *AND'ed* together is the same in whatever order they are grouped together.

12.2 Associative Law for the Logical OR (+) Function

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

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

For the OR associate law of addition, the order does not matter providing that the resulting circuit only contains the same type of OR gates. Indicating that the OR gate inputs can be interchanged and if required, the OR function can be extended to include three or more input variables.

14. De Morgan's Theorem

De Morgan's Theorem is a fundamental law in Boolean algebra, which provides a relationship between the complement of logical operations allowing us to simplify an expression.

De Morgan's theorem uses two sets of rules or laws to solve various Boolean algebra expressions. One for the complement (negation) of the logical **OR** operation changing *OR's* to *AND's*, and one for the complement of the logical **AND** operation *AND's* to *OR's*.

We have seen previously that Boolean algebra uses a set of laws and rules to define the operation of a digital logic circuit with 0's and 1's being used to create truth tables and mathematical expressions to define the digital operation of a logic **AND**, **OR** and **NOT** (or inversion) operations.

13.1 De Morgan's First Law of OR to AND

The first law states that the complement of the logical **OR** of two variables is equal to the logical **AND** of their complements. That is two separate terms *NOR'ed* together is the same as the two terms inverted (Complement) and *AND'ed*. For example: $\overline{A+B} = \overline{A}.\overline{B}$ as shown in Table 4.

Table 4. Proving De Morgan's First Law

Inputs		Truth table Output For Each Term				
B	A	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	0	1	0
1	0	1	0	1	0	0
1	1	1	0	0	0	0

In other words, the negation of the **OR** of two variables is equivalent to the **AND** of their individual negations.

13.2 De Morgan's Second Law of AND to OR

The second law states that the complement of the logical **AND** of two variables is equal to the logical **OR** of their complements. That is, two separate terms *NAND'ed* together is the same as the two terms inverted (Complement) and *OR'ed.* For example: $\overline{A.B} = \overline{A}+\overline{B}$, and we can prove this operation in Table 5.

Table 5. Proving De Morgan's Second Law

Inputs		Truth table Output For Each Term				
B	A	A.B	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	0	1	1	1	1
0	1	0	1	0	1	1
1	0	0	1	1	0	1
1	1	1	0	0	0	0

That is, the negation of the **AND** of two variables is equivalent to the **OR** of their individual negations.

Although we have used DeMorgan's theorems with only two input variables A and B, they are equally valid for use with three, four or more input variable expressions, for example:

For a 3-variable input

$$\overline{A.B.C} = \overline{A}+\overline{B}+\overline{C} \quad \text{and also} \quad \overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$$

For a 4-variable input

$$\overline{A.B.C.D} = \overline{A}+\overline{B}+\overline{C}+\overline{D} \quad \text{and also} \quad \overline{A+B+C+D} = \overline{A}.\overline{B}.\overline{C}.\overline{D}$$

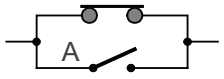
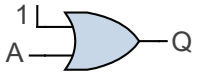
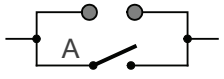
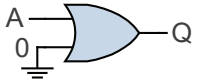
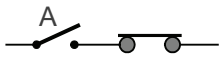
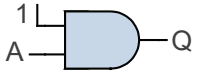
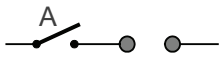
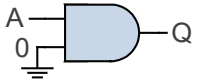
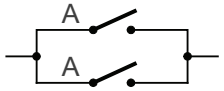
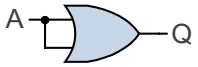
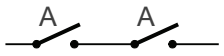
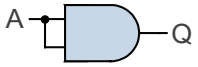
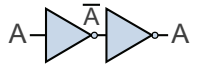
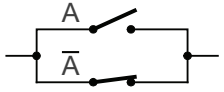
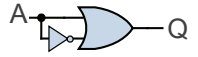
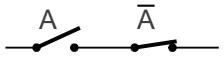
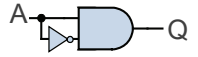
and so on.

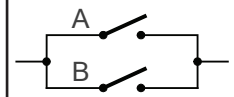
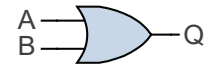
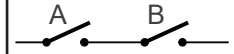
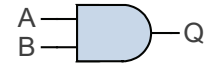
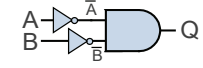
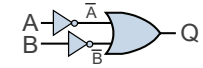
The De Morgan's two theorems allows us to simplify complex Boolean expressions, rewriting them in alternative forms, or convert between different representations.

15. Boolean Algebra Laws

The previous laws, rules and theorems for Boolean Algebra are summarised in Table 5.

Table 5. Switching Equivalents of Boolean Expressions

Boolean Expression	Description	Equivalent Switching Circuit	Equivalent Logic gate	Boolean Algebra Law or Rule
A+1 = 1	A in parallel with closed = CLOSED			Annulment
A+0 = A	A in parallel with open = A			Identity
A.1 = A	A in series with closed = A			Identity
A.0 = 0	A in series with open = OPEN			Annulment
A+A = A	A in parallel with A = A			Idempotent
A.A = A	A in series with A = A			Idempotent
NOT(not-A) = A	NOT NOT A = A (double negative)			Double Negation
$A+\overline{A} = 1$	A in parallel with NOT-A = CLOSED			Complement
$A.\overline{A} = 0$	A in series with NOT-A = OPEN			Complement

A+B = B+A	A in parallel with B = B in parallel with A			Commutative
A.B = B.A	A in series with B = B in series with A			Commutative
$\overline{A+B} = \overline{A}.\overline{B}$	invert and replace OR with AND			de Morgan's Theorem
$\overline{A.B} = \overline{A}+\overline{B}$	invert and replace AND with OR			de Morgan's Theorem

The basic Boolean algebra laws which relate the *Commutative Law* to a change in position for addition and multiplication. The *Associative Law* for the removal of brackets for addition and multiplication. The *Distributive Law* allowing the factoring of an expression, are all the same as in ordinary algebra.

Each of the Boolean Laws outlined in Table 5 are presented with just one single or two input variables, but the number of variables defined by a single law is not limited to just two, since there can be an infinite number of variables as inputs too any expression.

When Boolean expressions are implemented using logic gates, each term requires a gate, and each variable within that term represents an input to that gate. For 2-input logic gates, the basic logical operations where A and B are logic (or Boolean) input binary variables, and whose values can only be either logic-0 or logic-1. Which produces four possible input combinations of: 00, 01, 10, and 11 as shown in table 6.

Table 6. Truth table For Each Logical Expressions

B	A	AND	NAND	OR	NOR	NOT	Ex-OR	Ex-NOR
0	0	0	1	0	1	1	0	1
0	1	0	1	1	0	0	1	0
1	0	0	1	1	0	1	1	0
1	1	1	0	1	0	0	0	1

16. Boolean Algebra Functions

Using the information we have learnt in this *Boolean Algebra eBook*, simple 2-input **AND**, **OR** and **NOT** Gates can be represented by 16 possible functions as shown in Table 7.

Table 7. Boolean Algebra Functions table

Function	Description	Expression
1.	NULL	0
2.	IDENTITY	1
3.	Input A	A
4.	Input B	B
5.	not-A	$\overline{A}$
6.	not-B	$\overline{B}$
7.	A AND B (AND Gate)	A.B
8.	A AND not-B	A.$\overline{B}$
9.	not-A AND B	$\overline{A}$.B
10.	NOT AND (NAND Gate)	$\overline{A.B}$
11.	A OR B (OR Gate)	A+B
12.	A OR not-B	A+$\overline{B}$
13.	not-A OR B	$\overline{A}$+B
14.	NOT OR (NOR Gate)	$\overline{A+B}$
15.	Exclusive-OR	A.$\overline{B}$+$\overline{A}$.B
16	Exclusive-NOR	A.B+$\overline{A}$.$\overline{B}$

17. Boolean Algebra Examples

Using the previous rules and theorems, we will practice here with a few worked examples as a guide.

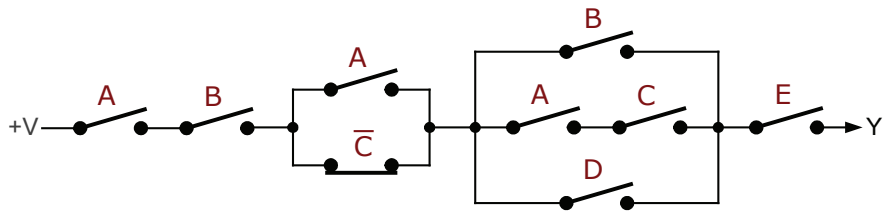
Example 17.1 Simplify the following Boolean expression: (A+B)(A+C)

(A+B).(A+C)	
(A.A) + A.C + A.B + B.C	Distributive law
(A.A) = A	Idempotent Law
(A + A.C) + A.B + B.C	Reduction
(A + A.C) = A	Absorption Law
(A + A.B) + B.C	Distributive Law
(A + A.B) = A	Absorption Law
A + B.C	

Thus the Boolean expression of (A + B)(A + C) can be reduced to just A + B.C using the various Boolean Algebra laws.

Example 17.2 Find the Boolean expression representing the switching circuit of Figure 15. Show the resulting switching circuit and its digital logic gate equivalent.

Figure 15. Complex Switching Circuit

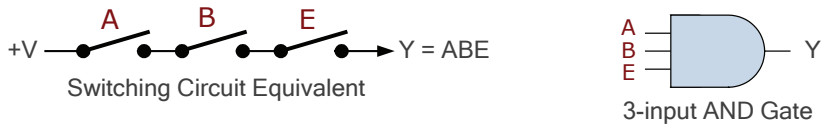


The initial Boolean expression is determined as being: Y = A.B(A+$\overline{C}$)(B+AC+D)E

$A.B(A+\overline{C})(B+AC+D)E$	
$A.B(B+A.C+D)E.A+A.B(B+A.C+D)E.\overline{C}$	Distribution
$A.A = A$	Idempotent Law
$A.B.E(B+A.C+D)+A.B.E(B+A.C+D)\overline{C}$	
$A + A.B = A$	Absorption Law
$A.B.E(B+A.C+D)$	
$A.B.E.B+A.B.E.A.C+A.B.E.D$	Distribution
$A.A = A$	Idempotent Law
$A.B.E+A.B.E.A.C+A.B.E.D$	
$A.A = A$	Idempotent Law
$A.B.E+A.B.E.C+A.B.E.D$	
$A + A.B = A$	Absorption Law
$A.B.E+A.B.E.D$	
$A + A.B = A$	Absorption Law
$Y = A.B.E$	

Thus, the switching circuit of Figure 15. can be reduced to **A.B.E** using the various laws and is given along with its logic gate equivalent of a 3-input AND gate in Figure 16.

Figure 16. Reduced Switching Circuit



End of this Boolean Algebra and Switching Theory eBook

Last revision: April 2023

With the completion of this Boolean Algebra and Switching Theory eBook you should have gained a good and basic understanding and knowledge of the various rules and laws of Boolean to help both reduce and simplify a complex Boolean expression. The information provided here should give you a firm foundation for continuing your study of electronics and electrical engineering as well as the study of digital logic switching circuits.

For more information about any of the topics covered here please visit our website at:

www.electronics-tutorials.ws

Main Headquarters
245 Main Street
Cambridge, MA 02142
www.aspencore.com

Central Europe/EMEA
Frankfurter Strasse 211
63263 Neu-Isenburg, Germany
info-europe@aspencore.com