3.3.2 Boolean Algebra

We have met gate logic and combination of gates. Another way of representing gate logic is through Boolean algebra, a way of algebraically representing logic gates. You should have already covered the symbols, below is a quick reminder:

Bitwise Operator	NOT($\overline{A}$)	AND(.)	OR(+)	XOR($\oplus$)	NAND($\overline{A.B}$)	NOR($\overline{A+B}$)
Description	invert input	where exactly two 1s	where one or more 1s	where exactly one 1	where less than two 1s	where exactly two 0s

Boolean Operations and Expressions

"Variable", "Complement", and "Literal" are terms used in Boolean Algebra.

- A **variable** is a symbol used to represent a logical quantity. Any single variable can have a **"1"** or a **"0"** value
- The **complement** is the inverse of a variable and is indicated by a bar over the variable. **The complement of a variable is not considered as a different variable.**
- Every occurrence of a variable or its complement is called a **Literal**. It could be its true form or its complement, both of them are called Literals
- A **SUM TERM** is the SUM of literals (A+B+C+D)
  - A sum term is equal to **1** if one or all of its inputs are **1**. And is equal to **0** only if all of its inputs are zero.
- A **PRODUCT TERM** is the PRODUCT of literals (A.B.C.D)
  - A product term is equal to **1** if all of its inputs are 1. And is equal to **0** if any one (or all) of its inputs are zero.

3.3.2 Boolean Algebra

Describing Logic Circuits Algebraically:

- Any logic circuit, no matter how complex, may be completely described using the Boolean operations previously defined. Because the **OR** gate, **AND** gate, and **NOT** gate are the basic building blocks of digital circuits.

- Boolean algebra provides a concise way to express the operation of a logic circuit formed by a combination of logic gates so that the output can be determined for various combinations of input values.

- To derive the Boolean expression for a given logic circuit, begin at the left most inputs and work towards the final output, writing the expression for each gate.

Laws and Rules of Boolean Algebra:

- **Equivalent and Complement of Boolean Expressions**

  - Two given Boolean expressions are said to be equivalent if one of them equals “1” only when the other also equals “1” and same case with “0”

  - They are said to be **complement** of each other if one expression equals “1” only when the other equals “0” and vice versa.

- **Postulates of Boolean Algebra**
  The following are the important postulates of Boolean algebra:

1.	$1.1 = 1$	&	$0+0 = 0$
2.	$1.0 = 0.1 = 0$	&	$0+1 = 1+0 = 1$
3.	$0.0 = 0$	&	$1+1 = 1$
4.	$\overline{1} = 0$	&	$\overline{0} = 1$

3.3.2 Boolean Algebra

Theorems of Boolean Algebra

- Boolean theorems can be useful in simplifying a logic expression. That is, in reducing the number of terms in the expression.

- It is useful in the sense that the number of gates are reduced which in turn also reduces heat dissipation from the circuit (saves energy)

- When this is done, the reduced expression will produce a circuit that is less complex than the one which the original expression would have produced.

- **Commutative Laws**
  - For Addition:
    - **X + Y = Y + X**
  - For Multiplication:
    - **X.Y = Y.X**

- **Associative Laws**
  - For Addition:
    - **X+(Y+Z) = Y+(Z+X) = Z+(X+Y)**
  - For Multiplication
    - **X.(Y.Z) = Y.(Z.X) = Z.(X.Y)**

- **Distributive Laws**
  - **X.(Y+Z) = X.Y + X.Z**
  - **(X.Y) + (X.Z) = X(Y+Z)**

Operations with '0' and '1'

- **RULE 1:**
  - **0+X = X**
- **RULE 2:**
  - **1+x = 1**
- **RULE 3:**
  - **0.X = 0**
- **RULE 4:**
  - **1.X = X**

3.3.2 Boolean Algebra

Idempotent or Identity Laws:

- **RULE 5:**
  - $X.X.X.X \ldots\ldots\ldots\ldots X = X$
- **RULE 7:**
  - $X+X+X+X+\ldots\ldots\ldots\ldots+X = X$

Complementation Law:

- **RULE 6:**
  - $X.\overline{X} = 0$
- **RULE 8:**
  - $X+\overline{X} = 1$

Involution Law:

- **RULE 9:**
  - $\overline{\overline{X}} = X$

Absorption Law or Redundancy Law:

- **RULE 10:**
  - $X+X.Y = X$
- **RULE 11:**
  - $X.\overline{Y} + Y = X+Y$
- **RULE 12:**
  - $(X+Y).(X+Z) = X+Z.Y$

RULE 10

$A + AB = A$

L.H.S = $A + AB$

$= A(1+B)$

$= A.1$ {Rule 2}

$= A$ {Rule 4}

= R.H.S

RULE 11

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

$A + \overline{A}B = (A + AB) + \overline{A}B$ {Rule 10}

$= (AA + AB) + \overline{A}B$ {Rule 7}

$= AA + AB + A\overline{A} + \overline{A}B$ {Rule 8}

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

$= 1.(A+B)$ {Rule 6}

$= A+B$ {Rule 4}

RULE 12

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

$(A + B)(A + C) = AA + AC + BA + BC$

$= A + AC + BA + BC$ {Rule 7}

$= A(1+C) + BA + BC$

$= A.1 + BA + BC$ {Rule 2}

$= A + BA + BC$ {Rule 4}

$= A(1 + B) + BC$

$= A.1 + BC$ {Rule 2}

$= A + BC$ {Rule 4}

3.3.2 Boolean Algebra

DeMorgan's Theorem

- DeMorgan's theorems provide mathematical verification of the equivalency of the NAND and negative-OR gates & the equivalency of the NOR and negative-AND gates.

  - **Theorem 1:**
    - The complement of a product of variables is equal to the sum of the complements of the variables
  - **Theorem 2:**
    - The complement of a sum of variables is equal to the product of the complements of the variables.

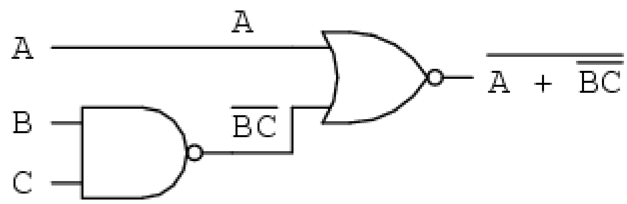
$$(a)\ [\overline{X_1 + X_2 + X_3 + \ldots + X_n}] = \overline{X_1}.\overline{X_2}.\overline{X_3}.\ \ldots\ .\overline{X_n}$$

$$(b)\ [\overline{X_1.X_2.X_3.\ \ldots\ .X_n}] = [\overline{X_1} + \overline{X_2} + \overline{X_3} + \ldots + \overline{X_n}]$$

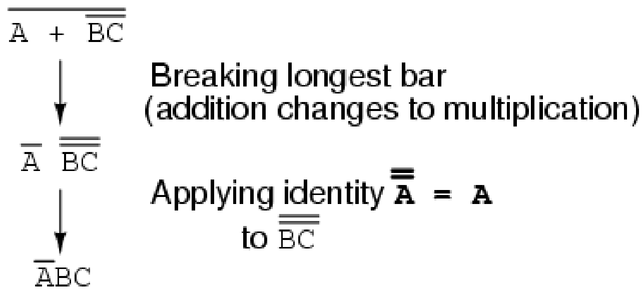
DeMorgan's Theorems

3.3.2 Boolean Algebra

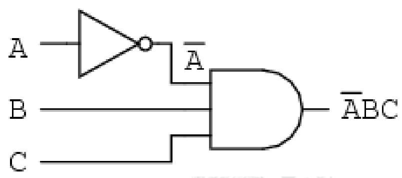
When multiple “layers” of bars exist in an expression, you may only break one bar at a time, it is generally easier to begin simplification by breaking the longest (uppermost) bar first. To illustrate, let’s take the expression (A + (BC))



Following the advice of breaking the longest (uppermost) bar first, we begin by breaking the bar covering the entire expression as a first step:



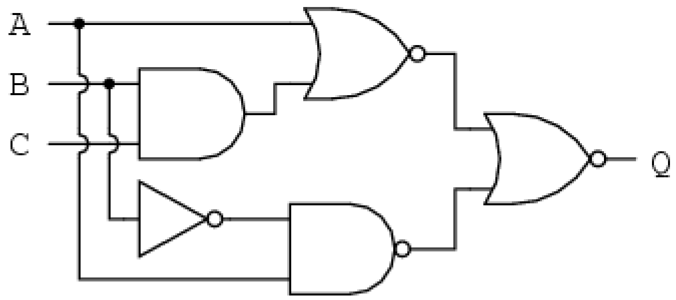
As a result, the original circuit is reduced to a three-input AND gate with the A input inverted:



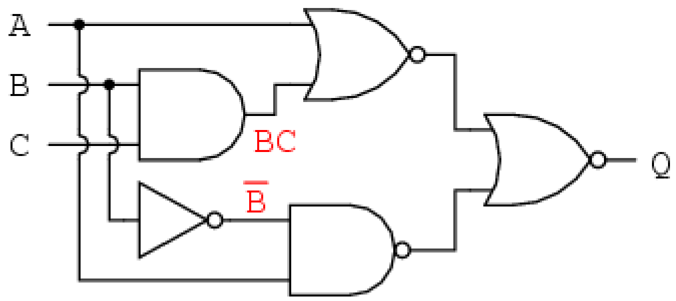
YOU SHOULD NEVER BREAK MORE THAN ONE BAR IN A SINGLE STEP.

3.3.2 Boolean Algebra

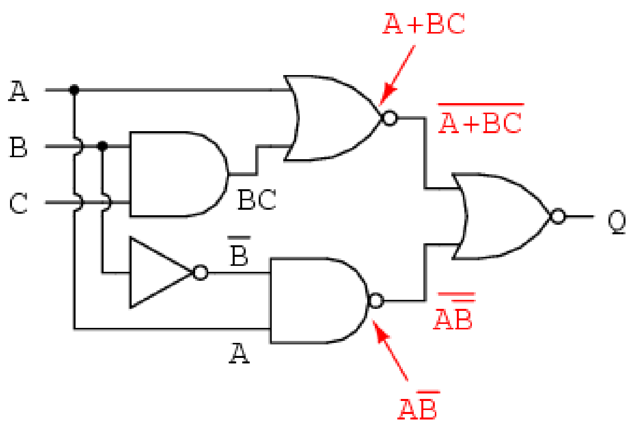
Let us apply DeMorgan's theorems to the simplification of a gate circuit:



As always, our first step in simplifying this circuit must be to generate an equivalent Boolean expression. We can do this by placing a sub-expression label at the output of each gate, as the inputs become known. Here is the first step in this process:

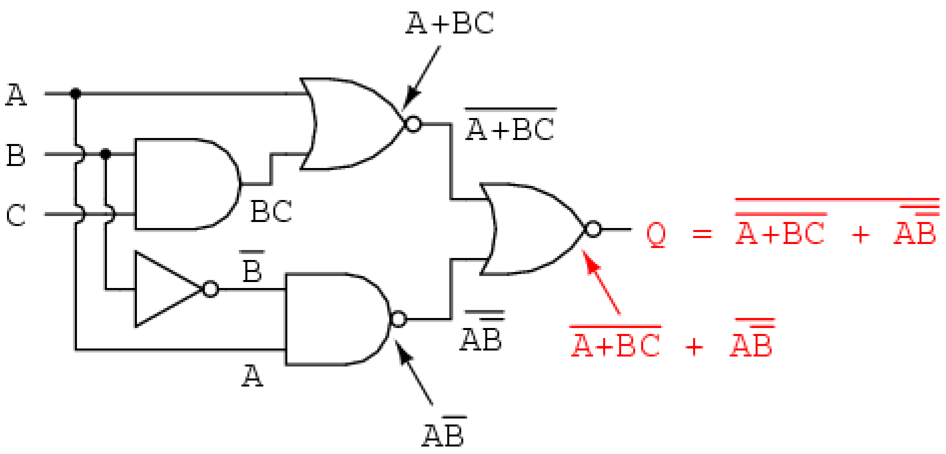


Next we can label the outputs of the first NOR gate and the NAND gate. When dealing with inverted-output gates, you may find it easier to write an expression for the gate's output **without** the final inversion, with an arrow pointing to just before the inversion bubble. Then, at the wire leading out of the gate (after the bubble), write the full complemented expression. This helps ensure you don't forget a complementing bar in the sub expression.



3.3.2 Boolean Algebra

Finally, we write an expression (or pair of expressions) for the last NOR gate:



Now we reduce this expression using identities, properties, rules and theorems (DeMorgan's) of Boolean algebra:

$$\overline{\overline{A + BC} + \overline{A\overline{B}}}$$

Breaking longest bar

$$(\overline{\overline{A + BC}})\ (\overline{\overline{A\overline{B}}})$$

Applying identity $\overline{\overline{A}} = A$ wherever double bars of equal length are found

$$(A + BC)(A\overline{B})$$

Distributive property

$$AA\overline{B} + BCA\overline{B}$$

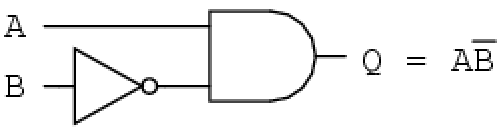
Applying identity $AA = A$ to left term; applying identity $A\overline{A} = 0$ to B and $\overline{B}$ in right term

$$A\overline{B} + 0$$

Applying identity $A + 0 = A$

$$A\overline{B}$$

Hence, the equivalent gate circuit for this much-simplified expression is as follows:



3.3.2 Boolean Algebra

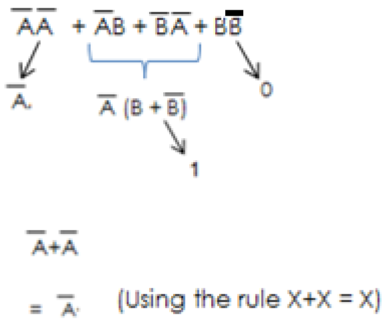
Here are some examples of Boolean algebra simplifications. Generally, there are several ways to reach the result.

- Simplify: $C + \overline{BC}$:

Expression	Rule(s) Used
$C + \overline{BC}$	Original Expression
$C + (\overline{B} + \overline{C})$	DeMorgan's Law.
$(C + \overline{C}) + \overline{B}$	Commutative, Associative Laws.
$1 + \overline{B}$	Complement Law.
1	Identity Law.

- Simplify: $\overline{AB}(\overline{A} + B)(\overline{B} + B)$:

Expression	Rule(s) Used
$\overline{AB}(\overline{A} + B)(\overline{B} + B)$	Original Expression
$\overline{AB}(\overline{A} + B)$	Complement law, Identity law.
$(\overline{A} + \overline{B})(\overline{A} + B)$	DeMorgan's Law



- Simplify: $(A + C)(AD + A\overline{D}) + AC + C$:

Expression	Rule(s) Used
$(A + C)(AD + A\overline{D}) + AC + C$	Original Expression
$(A + C)A(D + \overline{D}) + AC + C$	Distributive.
$(A + C)A + AC + C$	Complement, Identity.
$A((A + C) + C) + C$	Commutative, Distributive.
$A(A + C) + C$	Associative, Idempotent.
$AA + AC + C$	Distributive.
$A + (A + 1)C$	Idempotent, Identity, Distributive.
$A + C$	Identity, twice.

3.3.2 Boolean Algebra

- Simplify: $\overline{A}(A + B) + (B + AA)(A + \overline{B})$:

Expression	Rule(s) Used
$\overline{A}(A + B) + (B + AA)(A + \overline{B})$	Original Expression
$\overline{A}A + \overline{A}B + (B + A)A + (B + A)\overline{B}$	Idempotent (AA to A), then Distributive, used twice.
$\overline{A}B + (B + A)A + (B + A)\overline{B}$	Complement, then Identity. (Strictly speaking, we also used the Commutative Law for each of these applications.)
$\overline{A}B + BA + AA + B\overline{B} + A\overline{B}$	Distributive, two places.
$\overline{A}B + BA + A + A\overline{B}$	Idempotent (for the A's), then Complement and Identity to remove $B\overline{B}$.
$\overline{A}B + AB + AT + A\overline{B}$	Commutative, Identity; setting up for the next step.
$\overline{A}B + A(B + T + \overline{B})$	Distributive.
$\overline{A}B + A$	Identity, twice (depending how you count it).
$A + \overline{A}B$	Commutative.
$(A + \overline{A})(A + B)$	**Rule 11**
$A + B$	Complement, Identity.

(T = 1)