Boolean Algebra and Logic Simplification

Any digital circuit no matter how complex can be described by Boolean Expressions. Boolean algebra is the mathematics of Digital Systems. Knowledge of Boolean algebra is indispensable to the study and analysis of logic gates. AND, OR, NOT, NAND and NOR gates perform simple Boolean operations and Boolean expressions represent the Boolean operations performed by the logic gates.

- **AND gate**  \( F = A \cdot B \)
- **OR gate**  \( F = A + B \)
- **NOT gate**  \( F = \overline{A} \)
- **NAND gate**  \( F = \overline{A \cdot B} \)
- **NOR gate**  \( F = \overline{A + B} \)

Boolean expressions which represent Boolean functions help in two ways. The function and operation of a Logic Circuit can be determined by Boolean expressions without implementing the Logic Circuit. Secondly, Logic circuits can be very large and complex. Such large circuits having many gates can be simplified and implemented using fewer gates. Determining a simpler Logic circuit having fewer gates which is identical to the original logic circuit in terms of the function it performs can be easily done by evaluating and simplifying Boolean expressions.

Boolean Algebra expressions are written in terms of variables and literals using laws, rules and theorems of Boolean Algebra. Simplification of Boolean expressions is also based on the Boolean laws, rules and theorems.

### Boolean Algebra Definitions

1. **Variable**
   
   A variable is a symbol usually an uppercase letter used to represent a logical quantity. A variable can have a 0 or 1 value.

2. **Complement**
   
   A complement is the inverse of a variable and is indicated by a bar over the variable. Complement of variable \( X \) is \( \overline{X} \). If \( X = 0 \) then \( \overline{X} = 1 \) and if \( X = 1 \) then \( \overline{X} = 0 \).

3. **Literal**
   
   A Literal is a variable or the complement of a variable.

### Boolean Addition

Boolean Addition operation is performed by an OR gate. In Boolean algebra the expression defining Boolean Addition is a sum term which is the sum of literals.

\[ A + B , \overline{A \cdot B}, \overline{A} + \overline{B} + C \]
A sum term is 1 when any one literal is a 1
A sum term is 0 when all literals are a 0.

**Boolean Multiplication**

Boolean Multiplication operation is performed by an AND gate. In Boolean algebra the expression defining Boolean Multiplication is a product term which is the product of literals.

\[ A.B, \ A\overline{B}, \ \overline{A}.B.C \]

A product term is 1 when all literal terms are a 1
A product term is 0 when any one literal is a 0.

**Laws of Boolean Algebra**

The basic laws of Boolean Algebra are the same as ordinary algebra and hold true for any number of variables.
1. Commutative Law for addition and multiplication
2. Associative Law for addition and multiplication
3. Distributive Law

1. **Commutative Law for Addition and Multiplication**
   - Commutative Law for Addition  \[ A + B = B + A \]
   - Commutative Law for Multiplication  \[ A.B = B.A \]

![Figure 8.1 Implementation of Commutative Laws](image)

In terms of implementation, the Boolean Addition and Multiplication of two or more literals is the same no matter how they are ordered at the input of an OR and AND Gates respectively. Commutative law for Addition and Multiplication holds true for \( n \) number of literals.

2. **Associative Law for Addition and Multiplication**
   - Associative Law for Addition  \[ A + (B + C) = (A + B) + C \]
   - Associative Law for Multiplication  \[ A.(B.C) = (A.B).C \]
In terms of implementation, the Associative ordering of literals for Boolean Addition and Multiplication is the same at the input of an OR and AND gates. Commutative law for Addition and Multiplication holds true for n number of literals. The addition of literals B and C followed by the addition of literal A with the result of B+C is the same as adding literals A and B followed by the addition of literal C.

The multiplication of literals B and C followed by the multiplication of the result of B.C with literal A is the same as multiplying literals A and B followed by the multiplication of literal C.

3. Distributive Law
   • Distributive Law \[ A(B + C) = A.B + A.C \]

Distributive law holds true for any number of literals. Adding literals B and C followed by multiplying the result with literal A is the same as multiplying literal A with literal B and adding the result to the product of literals A and C.
Rules of Boolean Algebra

Rules of Boolean Algebra can be proved by replacing the literals with Boolean values 0 and 1.

1. \( A + 0 = A \)
2. \( A + 1 = 1 \)
3. \( A.0 = 0 \)
4. \( A.1 = A \)
5. \( A + A = A \)
6. \( A + \bar{A} = 1 \)
7. \( A.A = A \)
8. \( A.\bar{A} = 0 \)
9. \( \bar{A} = A \)
10. \( A + A.B = A \)
    \[ = A.(1 + B) \] where \((1+B)\) according to Rule 2 is equal to 1
    \[ = A \]
11. \( A + \bar{A}.B = A + B \)
    \[ = A(B+1) + \bar{A}.B \] according to Rule 2 \((B+1) = 1\)
    \[ = AB + A + \bar{A}.B \]
    \[ = B(A+\bar{A}) + A \] according to Rule 6 \(A + \bar{A} = 1\)
    \[ = B + A \]
12. \( (A+B).(A+C) = A+B.C \)
    \[ = AA+AC+AB+BC \] applying the Distributive Law
    \[ = A(1+C+B) + BC \] according to Rule 2 \((1+B+C) = 1\)
    \[ = A+BC \]

Demorgan’s Theorems

Demorgan’s First Theorem states: The complement of a product of variables is equal to the sum of the complements of the variables.

\[ \overline{AB} = \overline{A} + \overline{B} \]

Demorgan’s Second Theorem states: The complement of sum of variables is equal to the product of the complements of the variables.

\[ \overline{A + B} = \overline{A} \overline{B} \]

Demorgan’s two theorems prove the equivalency of the NAND and negative-OR gates and the NOR and negative-AND gates respectively. Figure 8.4
Demorgan’s Theorems can be applied to expressions having any number of variables
- $X.Y.Z = X + \overline{Y} + \overline{Z}$
- $X + Y + Z = \overline{X}.\overline{Y}.\overline{Z}$

Demorgan’s Theorem can be applied to a combination of other variables
- $(A + B.C).(A.C + B) = (A + B.C) + (A.C + B)$
- $= \overline{A}.(B.C) + (A.C).B$
- $= \overline{A}.(\overline{B} + \overline{C}) + (A + \overline{C}).\overline{B}$
- $= \overline{A}.\overline{B} + \overline{A}.C + \overline{A}.B + \overline{B}.C$
- $= \overline{A}.B + \overline{A}.C + B.C$

Boolean Analysis of Logic Circuits
Boolean algebra provides a concise way to represent the operation of a logic circuit. The complete function of the logic circuit can be determined by evaluating the Boolean expression using different input combinations.
The expression \((AB + \overline{C})D\) can be derived from the circuit by starting from the left hand, input side of the Logic Circuit. The AND gate provides the output \(AB\). The OR gate adds the product term \(AB\) and the complement \(C\) to result in \((AB + \overline{C})\) term. The AND gate on the right hand side of the circuit performs a multiplication operation between the term \((AB + \overline{C})\) and the literal \(D\) resulting in \((AB + \overline{C})D\).

There are four variables, therefore the function table or truth table for the logic circuit has 16 possible input combinations. The expression can be evaluated by trying out the 16 combinations. Alternately, the input combinations \(A, B, C\) and \(D\) that set the output of the expression \((AB + \overline{C})D\) to 1 can be easily determined.

From the expression, the output is a 1 if both variable \(D = 1\) and term \((AB + \overline{C}) = 1\). The term \((AB + \overline{C}) = 1\) only if \(AB=1\) or \(C=0\). Thus expression \((AB + \overline{C})D = 1\) if \(D=1\) AND \((C=0\ OR\ AB=1)\)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.1 Function table for expression \((AB + \overline{C})D\)

In the function table the input conditions for variables \(A, B, C\) and \(D\) that satisfy the condition \(D=1\) AND \(C=0\) are 0001, 0101, 1001. The condition \(D=1\) AND \(AB=1\) are satisfied by input combination 1111. The condition \(D=1\) AND \((C=0\ OR\ AB=1)\) is satisfied by the input combination 1101.
Simplification using Boolean Algebra

Many times a Boolean expression has to be simplified using laws, rules and theorems of Boolean Algebra. The simplified expression results in fewer variables and a simpler circuit. Consider the Boolean expression \( AB + A(B+C) + B(B+C) \) and the Logic Circuit represented by the expression. Figure 8.6. The simplification of the expression results in an expression \( B + AC \) represented by a simpler circuit having fewer gates. Figure 8.7

\[
\begin{align*}
&= AB + A(B+C) + B(B+C) \\
&= AB + AB + AC + BB + BC & \text{using Distributive Law} \\
&= AB + AC + B + BC & BB = B \text{ using rule 7} \\
&= AB + AC + B & (B+BC) = B \text{ using rule 10} \\
&= B + AC & (B+AB) = B \text{ using rule 10}
\end{align*}
\]

Figure 8.6  Logic Circuit represented by Boolean expression \( AB + A(B+C) + B(B+C) \)

Figure 8.7  Simplified Logic Circuit represented by Boolean expression \( B+AC \)

Standard Form of Boolean Expressions

All Boolean expressions can be converted into and represented in one of the two standard forms

- Sum-of-Products form
- Product-of-Sums form
1. Sum of Product form
   When two or more product terms are summed by Boolean addition, the result is a
   Sum-of-Product or SOP expression.
   - $AB + ABC$
   - $ABC + CDE + BCD$
   - $\overline{A}B + \overline{AB}C + AC$

   The Domain of an SOP expression is the set of variables contained in the
   expression, both complemented and un-complemented. A SOP expression can have a
   single variable term such as $A$. A SOP expression can not have a term of more than one
   variable having an over bar extending over the entire term, such as $\overline{AB} + C$.

2. Product of Sums form
   When two or more sum terms are multiplied by Boolean multiplication, the result
   is a Product-of-Sum or POS expression.
   - $(\overline{A} + B)(A + B + C)$
   - $(\overline{A} + B + C)(C + D + E)(B + C + D)$
   - $(A + B)(A + B + C)(\overline{A} + C)$

   The Domain of a POS expression is the set of variables contained in the
   expression, both complemented and un-complemented. A POS expression can have a
   single variable term such as $A$. A POS expression can not have a term of more than one
   variable having an over bar extending over the entire term such as $(\overline{A} + B)(A + B + C)$.

Implementation of an SOP and POS expression
   A SOP expression can be implemented by an AND-OR combination of gates. The
   product terms are implemented by an AND gate and the SOP expression is implemented
   by OR gate connected to the outputs of the AND gates. Figure 8.8

![Figure 8.8 SOP Implementation of Boolean expression B+AC+AD](image-url)
A POS expression can be implemented by an OR-AND combination of gates. The sum terms are implemented by OR gates and the POS expression is implemented by AND gate connected to the outputs of the OR gates.

![Diagram of POS implementation](image)

Figure 8.9  POS Implementation of Boolean expression \((A+B)(B+C+D)(A+C)\)

Conversion of a general expression to SOP form

Any logical expression can be converted into SOP form by applying techniques of Boolean Algebra

- \(AB + B(CD + EF) = AB + BCD + BEF\)
- \((A + B)(B + C + D) = AB + AC + AD + B + BC + BD = AC + AD + B\)
- \((A + B) + C = (A + B)\overline{C} = (A + B)\overline{C} = AC + BC\)