Boolean Algebra

A. Chouraqui-Benchaib

Mathematics Department,
Tlemcen University.

Outline

This chapter is divided into two parts. We first give some definitions to introduce Boolean Algebra; we define logic gates which are electronic circuits that can be used to implement the logic expressions also known as Boolean expressions. There are three basic logic gates, namely the **OR** gate, the **AND** gate and the **NOT** gate, the **EXCLUSIVE-OR** gate and the **EXCLUSIVE-NOR** gate. In the second part we present postulates for the definition of Boolean algebra. Boolean algebra is mathematics of logic, developped in 1854 by George Boole to treat the logic functions, it is used for simplification of complex logic expressions. Other simple techniques of simplification are Karnaugh maps and the tabular method given by Quine-McCluskey.

Definition

The binary variables (boolean variable), as we know can have either of the two states, i.e. the logic $'0'$ state or the logic $'1'$ state. A variable in state 1 is referred to as active and the symbols 0 and 1 represent logical states but do not have a numerical value.

Example

A suitable K can only have two states: open $'0'$ or closed $'1'$.

Definition

A logical (binary or Boolean) function of binary variables is a function whose values can be either of the two states, $'0'$ or $'1'$.

Definition

A truth table of a logical function is a table who lists all possible combinations of input binary variables and the corresponding outputs of a logic system. Note that a truth table of logical function with n binary variables is a table with $n+1$ columns and 2^n lines.

Example

If the number of input binary variables is 2, then there are $2^2 = 4$ possible input combinations.

Example

Let f be a logical function with two binary variables a and b, defined by:

$$f(a,b) = \begin{cases} 1; \text{ if } a = b = 1 \\ 0 \text{ otherwise.} \end{cases}$$

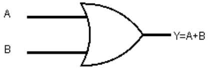
a	b	$f(a,b)$
0	0	0
0	1	0
1	0	0
1	1	1

We define a set of components, known as logic gates, each of one corresponds to a logical function. The three basic logic gates are the **OR** gate, the **AND** gate and the **NOT** gate.

OR Gate

An OR gate is a logic circuit with two ore more inputs and one output. The output of an OR gate is LOW (0) if and only if all of its inputs are LOW, for all other possible input combinations, the output is HIGH. Figure1 shows the circuit symbol and the truth table of a two-input OR gate $Y = A + B$.

OR Gate

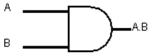


A	B	$A+B$
0	0	0
0	1	1
1	0	1
1	1	1

Figure: Two-input OR Gate

AND Gate

An AND gate is a logic circuit with two ore more inputs and one output. The output of an AND gate is HIGH (1) if and only if all of its inputs are HIGH, for all other possible input combinations, the output is LOW (0). Figure2 shows the circuit symbol and the truth table of a two-input AND gate $Y = A \cdot B$.



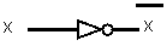
A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Figure: Two-input AND Gate

NOT Gate

An NOT gate is a logic circuit with one input and one output. The output is always the complement of the input, i.e a LOW input produces a HIGH output, and vice versa. Figure3 shows the circuit symbol and the truth table of a NOT gate ($\overline{X}$ is the complement of X).

NOT Gate



X	$\overline{X}$
0	1
1	0

Figure: NOT Gate

EXCLUSIVE-OR Gate

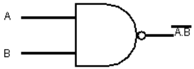
The EXCLUSIVE-OR gate, written as EX-OR gate, with two-input and one-output. Figure**??** shows the logic symbol and truth table of a two-input EX-OR gate. We see from the truth table, that the output of an EX-OR gate is a logic $'1'$ when the inputs are unlike and a logic $'0'$ when the inputs are like. The output of a two-input EX-OR gate is expressed by

$$Y = (A \oplus B) = \overline{A} \cdot B + A \cdot \overline{B}.$$

NAND Gate

NAND is an abbreviation of NOT AND, the truth table of a NAND gate is obtained from the truth table of an AND gate by complementing the output entries. Figure 16 shows the circuit symbol of a two-input NAND gate. The little invert bubble (small circle) on the right end of the symbol means to invert the output of AND.

NAND Gate



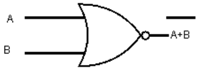
A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

Figure: NAND Gate

NOR Gate

NOR is an abbreviation of NOT OR, the truth table of a NOR gate is obtained from the truth table of an OR gate by complementing the output entries. Figure NOR shows the circuit symbol of a two-input NOR gate.

NOR Gate

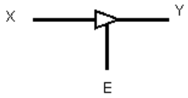


A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Figure: NOR Gate

Three-state buffer

A three-state buffer functions as a signal-controlled switch. An enable signal controls whether the input signal is sent to the output or isolated from the output, which remains in a high-impedance state. Figure19 shows circuit and the truth table of the three-state buffer; (Z: High impedance state)



E	X	Y
0	x	Z
1	0	0
1	1	1

Figure: Three-state buffer

Electric representation

The diagram is created by interaction using electronic components, in other words it is implemented practically with electrical components in a laboratory.

Algebric representation

For the algebric representation, we use the logical operations such that $+, \cdot, \oplus$.

Example

f is a logical function with algebric representation.

$$f(a,b) = \underbrace{a \cdot b}_{\text{logical term}} + \overline{a} \cdot b$$

Arithmetic representation

The arithmetic representation means representation by the truth table.

Example

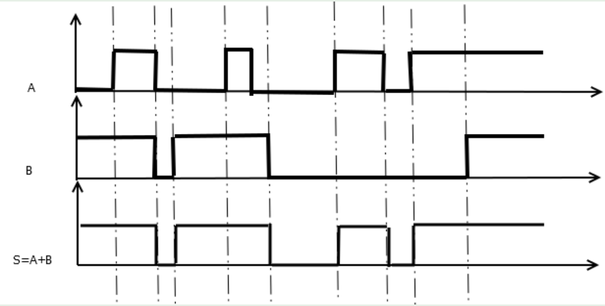
The arithmetic representation of $f(A, B) = A \cdot B + A + C \cdot D$ is given by the following truth table

Time representation or Timing diagram-Chronogram

The timing diagram is a graphical representation that illustrates the timing relationships between various signals or events in a system; often employed in electronics, digital design, and other fields to study and understand system behaviour across time.

Example

Complete the timing diagram of the logical function $S = A + B$.



Graphic representation-diagram

We can define a logical function by representing it using logic gates, then read it from left to right. We call this representation Diagram.

Example

The following diagram represents the logical function

$$f(a,b) = a \cdot b + \overline{a} \cdot b.$$

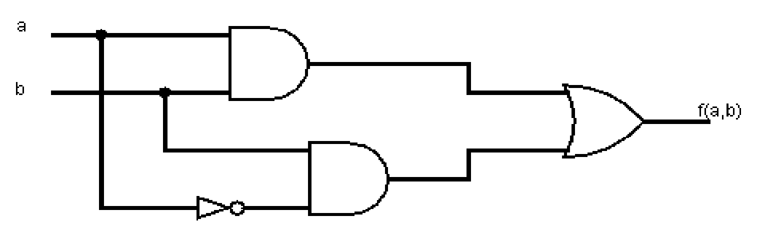


Figure: Diagram of f

Boolean Algebra

Boolean algebra, is simpler than ordinary algebra, it is also composed of a set of symbols and a set of rules to manipulate these symbols.

Definition

Let B be a set of logical variables supplied with two binary operations **AND** noted by "$\cdot$ ", **OR** noted by "+ " and **NOT** noted by "$^-$ ", $(B, \cdot, +, ^-)$ is a Boolean algebra if and only if the following postulates (axioms) are verified.

Axioms

Definition

(A) Commutativity: $\forall\ a, b \in B : a + b = b + a$ and $a \cdot b = b \cdot a$.

(B) associativity:
$\forall\ a, b, c \in B : (a+b)+c = a+(b+c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

(C) distrubitivity: $\forall\ a, b, c \in B : a \cdot (b + c) = a \cdot b + a \cdot c$ and $a + b \cdot c = (a + b) \cdot (a + c)$.

(D) $\forall a \in B : a + 0 = a$ and $a \cdot 1 = a$.

(E) For all element a in B, there exist a unique complement element noted by $\overline{a}$ such that $a + \overline{a} = 1$ and $a \cdot \overline{a} = 0$.

Example

1. $(\mathcal{P}(E), \cap, \cup, \mathcal{C})$ is a Boolean algebra.
2. $(P, \wedge, \vee, ^{-})$ is a Boolean algebra; where P is a set of propositions.

The dual of a Boolean expression is obtained by replacing all "$\cdot$" operations by "$+$" operations, all "$+$" operations by "$\cdot$" operations, all 0s by 1s and all 1s by 0s and living all literals unchanged.

Example

The coresponding dual of the Boolean expression $(a+b)\cdot(\overline{a}+\overline{b})$ is $(a \cdot b)+(\overline{a}\cdot\overline{b})$

Duals of Boolean expressions are mainly of interest in the study of Boolean postulates and theorems.

Idempotent

We apply theorems of Boolean algebra to simplify Boolean expressions and transform them into a more useful and meaningful expressions. We note that if a given expression is valid, its dual will also be valid.

Theorem

(Idempotent or Identity Laws)

$$\forall a \in B;\ a + a = a \text{ and } a \cdot a = a.$$

Theorem2

Theorem

$$\overline{0} = 1 \textit{ and } \overline{1} = 0.$$

Theorem3

Theorem

(Operations with '0' and '1')

$$\forall a \in B; \ a \cdot 0 = 0 \ and \ a + 1 = 1.$$

Involution law

Theorem

$$\forall a \in B; \ \overline{\overline{a}} = a.$$

Absoption Law1

Theorem

$$\forall a, b \in B;\ a + a \cdot b = a \text{ and } a \cdot (a + b) = a$$

Absorption Law2

Theorem

(Absorption Law2)

$$\forall a, b \in B;\ a + \overline{a} \cdot b = a + b \text{ and } a \cdot (\overline{a} + b) = a \cdot b.$$

Consensus Theorem

Theorem

$$\forall a, b, c \in B;\ \ a \cdot b + b \cdot c + \overline{a} \cdot c = a \cdot b + \overline{a} \cdot c,$$
$$(a + b) \cdot (b + c) \cdot (\overline{a} + c) = (a + b) \cdot (\overline{a} + c)$$

Theorem 8

Theorem

$$
\begin{aligned}
\forall a, b, c \in B;\ (a+c)\cdot(\overline{a}+b) &= a\cdot b+\overline{a}\cdot c,\\
a\cdot c+\overline{a}\cdot b &= (a+b)\cdot(\overline{a}+c).
\end{aligned}
$$

DeMorgan's Theorem

Theorem

$$\forall a, b \in B;\ \overline{a+b} = \overline{a} \cdot \overline{b} \text{ and } \overline{a \cdot b} = \overline{a} + \overline{b}.$$

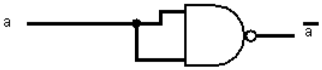
The above theorems and postulats are used to simplify boolean expressions, we call this method the algebric simplification and the theorem of involution law is the basis of finding the equivalent product-of-sums expression for a given sum-of-products expression, and vice versa.

Example

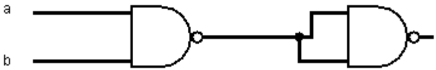
Apply boolean laws and theorems to modify a two-input OR gate into two-input NAND gates only.

The two-input NAND gate (NOR resp.) is a complete gate because we can use it to implement AND, OR, NOT, NAND, NOR and NOT with many inputs.

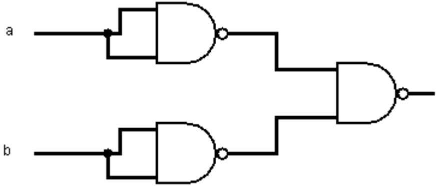
$\overline{a} = \overline{a \cdot a}$ Idempotent law.



$$a \cdot b = \overline{\overline{a \cdot b}},$$

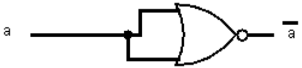


$$a + b = \overline{\overline{a + b}} = \overline{\overline{a} \cdot \overline{b}},$$

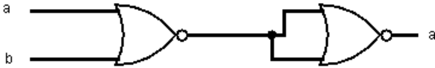


Similary,

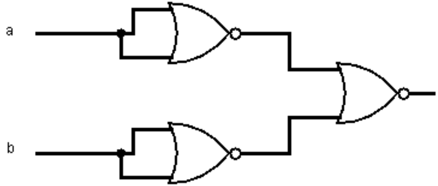
$a = \overline{a + a}$ Idemptent law,



$$a + b = \overline{\overline{a + b}},$$



$$a \cdot b = \overline{\overline{a \cdot b}} = \overline{\overline{a} + \overline{b}},$$



Example1

Example

Show that.

1. $(a \cdot b) \oplus (a + b) = a \oplus b.$
2. $a \cdot b + \overline{c} + c \cdot (\overline{a} + \overline{b}) = 1.$

Example2

Example

Simplify the following expression.

$$S = \overline{a} \cdot b \cdot c + a \cdot \overline{b} \cdot \overline{c} + \overline{a} \cdot \overline{b} \cdot \overline{c} + a \cdot \overline{b} \cdot c + a \cdot b \cdot c.$$

The primary objective of simplification is to obtain an expression that has the minimum number of logical terms.
There are other methods of simplification than the application of laws and theorems of Boolean algebra such as the Karnaugh map method and the Quine-Mc Cluskey tabular method.
We first describe sum-of products and product-of sums Boolean expressions, to will be able to minimize expressions in the same or the other form

SOP

A sum-of-products expression contains the sum of different terms (product of all logical variables). It can be obtained directly from the truth table by considering those input combinations that produce $'1'$ at the output.
Different terms are given by the product of the inputs, where $'0'$ means complemented variable and $'1'$ means uncomplemented variable, and the sum of all such terms gives the expression.

Example of SOP

Example

We give a boolean expression of a truth table.....

minterms

Definition

We mean by minterm the term m_i, where i represents the decimal equivalent of the logical values product of the inputs from the truth table, m_i is the product of logical variables complemented or not. With n logical variables, we construct 2^n minterms.

Example

For two inputs a, b, we have four minterms $m_0 = \overline{a} \cdot \overline{b}, m_1 = \overline{a} \cdot b, m_2 = a \cdot \overline{b}$ and $m_3 = a \cdot b$ which correspond, respecively to $00, 01, 10$ and 11.
We write the above boolean function with minterms...

POS

A product-of-sums expression contains the product of different terms (sum of all logical variables). It can be obtained directly from the truth table by considering those input combinations that produce $'0'$ at the output.
Different terms are given by the sum of the inputs, where $'0'$ means uncomplemented variable and $'1'$ means complemented variable, the product of such terms gives the expression.
An OR gate produces a logic $'0'$ only when all its inputs are in the logic $'0'$ state.

Example

See your document....

Maxterm

A product-of-sums expression is also known us a maxterm expression.

Definition

We mean by maxterm the term M_i, where i represents the decimal equivalent of the logical values sum of the inputs from the truth table, M_i is the sum of logical variables complemented or not. With n logical variables, we construct 2^n maxterms.

Example

For two inputs a, b, we have four maxterms $M_0 = a + b, m_1 = a + \overline{b}, m_2 = \overline{a} + b$ and $m_3 = \overline{a} + \overline{b}$ which correspond, respecively to $00, 01, 10$ and 11.

a	b	minterms	maxterms	Binary form	Decimal form
0	0	$m_0 = \overline{a} \cdot \overline{b}$	$M_0 = a + b$	00	0
0	1	$m_1 = \overline{a} \cdot b$	$M_1 = a + \overline{b}$	01	1
1	0	$m_2 = a \cdot \overline{b}$	$M_2 = \overline{a} + b$	10	2
1	1	$m_3 = a \cdot b$	$M_3 = \overline{a} + \overline{b}$	11	3

$\overline{a} + b = 0 \iff a = 1 \text{and} b = 0$ then the binary form of this term is 10.

Example

Example

We consider the logical function f defined by

$$f(a,b,c) = a + \overline{b} \cdot c.$$

Let us give the SOP form.

$$\begin{aligned}
f(a,b,c) &= a + \overline{b} \cdot c \\
&= a \cdot 1 \cdot 1 + 1 \cdot \overline{b} \cdot c \\
&= a \cdot (b + \overline{b}) \cdot (c + \overline{c}) + (a + \overline{a}) \cdot \overline{b} \cdot c \\
&= a \cdot b \cdot c + a \cdot b \cdot \overline{c} + \underbrace{a \cdot \overline{b} \cdot c} + a \cdot \overline{b} \cdot \overline{c} + \underbrace{a \cdot \overline{b} \cdot c} + a \cdot \overline{b} \cdot \overline{c} + \\
&+ \overline{a} \cdot \overline{b} \cdot c \\
&= \underbrace{a \cdot b \cdot c}_{111} + \underbrace{a \cdot b \cdot \overline{c}}_{110} + \underbrace{a \cdot \overline{b} \cdot c}_{101} + \underbrace{a \cdot \overline{b} \cdot \overline{c}}_{100} + \underbrace{\overline{a} \cdot \overline{b} \cdot c}_{001}
\end{aligned}$$

Example

Let us give the POS form.

$$
\begin{aligned}
f(a,b,c) &= a + \overline{b} \cdot c \\
&= (a + \overline{b}) \cdot (a + c) \\
&= (a + \overline{b} + 0) \cdot (a + 0 + c) \\
&= (a + \overline{b} + c \cdot \overline{c}) \cdot (a + b \cdot \overline{b} + c) \\
&= (a + \overline{b} + c) \cdot (a + \overline{b} + \overline{c}) \cdot (a + b + c) \cdot (a + \overline{b} + c) \\
&= (a + \overline{b} + c) \cdot (a + \overline{b} + \overline{c}) \cdot (a + b + c) \\
&= M_0 \cdot M_2 \cdot M_3. \\
&= \prod(0,2,3).
\end{aligned}
$$

We will note the complement of $f(a,b,c)$ by $f'(a,b,c)$, then from SOP form

$$\begin{aligned} f'(a,b,c) &= (\overline{a}+\overline{b}+\overline{c})\cdot(\overline{a}+\overline{b}+c)\cdot(\overline{a}+b+\overline{c})\cdot(\overline{a}+b+c)\cdot(a+b \\ &= \prod(1,4,5,6,7), \end{aligned}$$

and from the POS form, we obtain

$$\begin{aligned} f'(a,b,c) &= \overline{a}\cdot b\cdot\overline{c}+\overline{a}\cdot b\cdot c+\overline{a}\cdot\overline{b}\cdot\overline{c} \\ &= \sum(0,2,3). \end{aligned}$$