Boolean Algebra

Boolean algebra provides the operations and the rules for working with the set **{0, 1}.**

These are the rules that underlie **electronic circuits**, and the methods we will discuss are fundamental to **VLSI design**.

We are going to focus on three operations:

- Boolean complementation,
- Boolean sum, and
- Boolean product

Boolean Operations

The **complement** is denoted by a bar (on the slides, we may use a minus sign). It is defined by

$\overline{0} = 1$ and $\overline{1} = 0$ (or $-0 = 1$ and $-1 = 0$)

The **Boolean sum**, denoted by + or by OR, has the following values:

$1 + 1 = 1$, $1 + 0 = 1$, $0 + 1 = 1$, $0 + 0 = 0$

The **Boolean product**, denoted by · or by AND, has the following values:

$1 \cdot 1 = 1$, $1 \cdot 0 = 0$, $0 \cdot 1 = 0$, $0 \cdot 0 = 0$

Boolean Functions and Expressions

Definition: Let B = {0, 1}. The variable x is called a **Boolean variable** if it assumes values only from B.

A function from B^n, the set $\{(x_1, x_2, \ldots, x_n) \mid x_i \in B, 1 \le i \le n\}$, to B is called a **Boolean function of degree n**.

Boolean functions can be represented using expressions made up from the variables and Boolean operations.

Boolean Functions and Expressions

The **Boolean expressions** in the variables $x_1, x_2, \ldots, x_n$ are defined recursively as follows:

- $0, 1, x_1, x_2, \ldots, x_n$ are Boolean expressions.
- If E_1 and E_2 are Boolean expressions, then $(-E_1)$, (E_1E_2), and $(E_1 + E_2)$ are Boolean expressions.

Each Boolean expression represents a Boolean function. The values of this function are obtained by substituting 0 and 1 for the variables in the expression.

Boolean Functions and Expressions

For example, we can create Boolean expression in the variables x, y, and z using the "building blocks" 0, 1, x, y, and z, and the construction rules:

Since x and y are Boolean expressions, so is xy.

Since z is a Boolean expression, so is $(\overline{z})$.

Since xy and $(\overline{z})$ are expressions, so is $xy + (\overline{z})$.

… and so on…

Boolean Functions and Expressions

Example: Give a Boolean expression for the Boolean function F(x, y) as defined by the following table:

x	y	F(x, y)
0	0	0
0	1	1
1	0	0
1	1	0

Possible solution: $F(x, y) = (\overline{x}) \cdot y$

Boolean Functions and Expressions

There is a simple method for deriving a Boolean expression for a function that is defined by a table. This method is based on **minterms**.

Definition: A **literal** is a Boolean variable or its complement. A **minterm** of the Boolean variables x_1, x_2, …, x_n is a Boolean product $y_1y_2\dots y_n$, where $y_i = x_i$ or $y_i = \overline{x}_i$.

Hence, a minterm is a product of n literals, with one literal for each variable.

Boolean Functions and Expressions

Definition: The Boolean functions F and G of n variables are **equal** if and only if $F(b_1, b_2, \dots, b_n) = G(b_1, b_2, \dots, b_n)$ whenever $b_1, b_2, \dots, b_n$ belong to B.

Two different Boolean expressions that represent the same function are called **equivalent**.

For example, the Boolean expressions xy, xy + 0, and $xy\cdot 1$ are equivalent.

Boolean Functions and Expressions

The **complement** of the Boolean function F is the function $\overline{F}$, where $\overline{F}(b_1, b_2, \dots, b_n) = \overline{F(b_1, b_2, \dots, b_n)}$.

Let F and G be Boolean functions of degree n. The **Boolean sum F+G** and **Boolean product FG** are then defined by

$(F+G)(b_1, b_2,\dots, b_n) = F(b_1, b_2, \dots, b_n) + G(b_1, b_2,\dots, b_n)$

$(FG)(b_1, b_2, \dots, b_n) = F(b_1, b_2, \dots, b_n)\, G(b_1, b_2, \dots, b_n)$

Boolean Functions and Expressions

Question: How many different Boolean functions of degree 2 are there?

Solution: There are 16 of them, $F_1, F_2, \dots, F_{16}$:

x	y	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Functions and Expressions

Question: How many different Boolean functions of degree n are there?

Solution:

There are 2^n different n-tuples of 0s and 1s.

A Boolean function is an assignment of 0 or 1 to each of these 2^n different n-tuples.

Therefore, there are 2^{2^n} different Boolean functions.

Boolean Identities

There are useful identities of Boolean expressions that can help us to transform an expression A into an equivalent expression B (see Table 5 on page 815 [6th edition: page 753] in the textbook).

$\bar{\bar{x}}$ = --x = x, law of double complement
x+x = x, idempotent laws
x·x = x
x+0 = x, identity laws
x·1 = x
x+1 = 1, domination laws
x·0 = 0
x+y = y+x, commutative laws
x·y = y·x

x+(y+z) = (x+y)+z, associative laws
x·(y·z) = (x·y)·z
x+yz = (x+y)(x+z), distributive laws
x·(y+z) = (x·y)+(x·z)
$\overline{(xy)} = \bar{x} + \bar{y}$, De Morgan's laws
$\overline{(x+y)} = (\bar{x})(\bar{y})$
x+xy = x, Absorption laws
x(x+y) = x
$x+\bar{x} = 1$, unit property
$x\bar{x} = 0$, zero property

Duality

We can derive additional identities with the help of the **dual** of a Boolean expression.

The dual of a Boolean expression is obtained by interchanging Boolean sums and Boolean products and interchanging 0s and 1s.

Duality

Examples:

The dual of x(y + z) is x + yz.
The dual of $\bar{x}\cdot 1 + (\bar{y} + z)$ is $(\bar{x} + 0)((\bar{y})z)$.
The dual is essentially the complement, but with any variable x replaced by $\bar{x}$. (exercise 29, p. 881)

The **dual of a Boolean function F** represented by a Boolean expression is the function represented by the dual of this expression.

This dual function, denoted by F^d, **does not depend** on the particular Boolean expression used to represent F. (exercise 30, page 881 [6th ed. p.756])

Duality

Therefore, an identity between functions represented by Boolean expressions **remains valid** when the duals of both sides of the identity are taken.

We can use this fact, called the **duality principle**, to derive new identities.

For example, consider the absorption law x(x + y) = x.

By taking the duals of both sides of this identity, we obtain the equation x + xy = x, which is also an identity (and also called an absorption law).

Definition of a Boolean Algebra

All the properties of Boolean functions and expressions that we have discovered also apply to **other mathematical structures** such as propositions and sets and the operations defined on them.

If we can show that a particular structure is a Boolean algebra, then we know that all results established about Boolean algebras apply to this structure.

For this purpose, we need an **abstract definition** of a Boolean algebra.

Definition of a Boolean Algebra

Definition: A Boolean algebra is a set B with two binary operations $\vee$ and $\wedge$, elements 0 and 1, and a unary operation – such that the following properties hold for all x, y, and z in B:

$x \vee 0 = x$ and $x \wedge 1 = x$ (identity laws)

$x \vee \bar{x} = 1$ and $x \wedge \bar{x} = 0$ (domination laws)

$(x \vee y) \vee z = x \vee (y \vee z)$ and
$(x \wedge y) \wedge z = x \wedge (y \wedge z)$ and (associative laws)

$x \vee y = y \vee x$ and $x \wedge y = y \wedge x$ (commutative laws)

$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ and
$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ (distributive laws)

Boolean Algebras

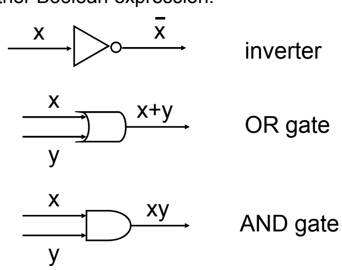
Examples of Boolean Algebras are:

1. The algebra of all subsets of a set U, with $+ = \cup$, $\cdot = \cap$, - = complement, $0 = \varnothing$, 1 = U.
2. The algebra of propositions with symbols $p_1, p_2, \ldots, p_n$, with $+ = \vee$, $\cdot = \wedge$, $- = \neg$, 0 = F, 1 = T.
3. If $B_1, \ldots, B_n$ are Boolean Algebras, so is $B_1 \times .. \times B_n$, with operations defined coordinate-wise.

Logic Gates

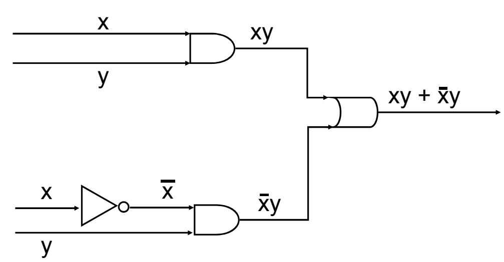
Electronic circuits consist of so-called gates. There are three basic types of gates.

In each case the input is a Boolean expression and the output is another Boolean expression.



Logic Gates

Example: How can we build a circuit that computes the function $xy + \bar{x}y$?



Multi switch light circuit

Suppose we want a circuit for a light controlled by two switches, where changing the state of either switch changes the state of the light (on or off).

If we let x and y be the states of the switches (0 or 1) then the boolean expression $xy + \bar{x}\,\bar{y}$ (- = complement) will do the job.

Multi switch light circuit

This is because if both x and y are “on” (1) or “off” (0) $xy + \bar{x}\,\bar{y}$) will be 1, and otherwise will be 0.

We can generalize this method. For three switches the Boolean expression $xyz + x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z$ will work.

Can you draw circuits implementing these expressions? (see pp. 825, 826 [6th ed. pp. 763, 764])

Adding binary integers

If we add two one bit integers x and y we get a sum for that bit position plus a carry bit.

If we don't consider a carry bit from a lower bit addition we get what's called a half adder.

If we do consider an input carry bit we have a full adder. (see p. 827, 6th ed.765)

Half Adder

Given input bits x and y, the result bit will be x+y unless both x and y are 1, in which case the result is 0.

This means that we can express the result bit as $(x+y)\overline{(xy)}$, or $(x+y)(\bar{x} + \bar{y})$.

The carry bit will be xy (we carry if both x and y are 1)

Full Adder

If we add a carry bit c_0 from the previous order bit sum our result for this bit would be 1 if one or three of c_0, x, y are 1, and 0 otherwise.

This means $xyc_0+x\bar{y}\overline{c_0}+\bar{x}y\overline{c_0}+\bar{x}\bar{y}c_0$ would work, with carry bit $xyc_0+xy\overline{c_0}+x\bar{y}c_0+\bar{x}yc_0$

See p. 827 to check your implementation.