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 will use a minus sign). It is defined by
\(-0 = 1\) and \(-1 = 0\).

The Boolean sum, denoted by + or by OR, has the following values:
\(1 + 1 = 1,\quad 1 + 0 = 1,\quad 0 + 1 = 1,\quad 0 + 0 = 0\)

The Boolean product, denoted by \(\cdot\) or by AND, has the following values:
\(1 \cdot 1 = 1,\quad 1 \cdot 0 = 0,\quad 0 \cdot 1 = 0,\quad 0 \cdot 0 = 0\)

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 \(\{(x_1, x_2, \ldots, x_n) \mid x_i \in B, 1 \leq i \leq n\}\), 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:
• \(0, 1, x_1, y_1, \ldots, x_n\) 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

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

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y)</th>
<th>(F(x, y))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Possible solution: \(F(x, y) = (-x) \cdot y\)
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 \( x_1, x_2, \ldots, x_n \) is a Boolean product \( y_1y_2\ldots y_n \) where \( y_i = x_i \) or \( y_i = \neg x_i \).

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

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

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

**Solution:** There are 16 of them, \( F_1, F_2, \ldots, F_{16} \):

| \( x \) | \( y \) | \( F_1 \) | \( F_2 \) | \( F_3 \) | \( F_4 \) | \( F_5 \) | \( F_6 \) | \( F_7 \) | \( F_8 \) | \( F_9 \) | \( F_{10} \) | \( F_{11} \) | \( F_{12} \) | \( F_{13} \) | \( F_{14} \) | \( F_{15} \) |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 1     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 1     | 0     | 0     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |
| 1     | 1     | 0     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |

**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).
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.

Examples:
The dual of $x(y + z)$ is $x + yz$.
The dual of $-x1 + (-y + z)$ is $(-x + 0)((-y)z)$.
The dual is essentially the complement, but with any variable $x$ replaced by $-x$. (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 \( \lor \) and \( \land \), elements 0 and 1, and a unary operation \( - \) such that the following properties hold for all \( x, y, \) and \( z \) in \( B \):

1. \( x \lor 0 = x \) and \( x \land 1 = x \) (identity laws)
2. \( x \lor (-x) = 1 \) and \( x \land (-x) = 0 \) (domination laws)
3. \((x \lor y) \lor z = x \lor (y \lor z)\) and \((x \land y) \land z = x \land (y \land z)\) (associative laws)
4. \( x \lor (y \land z) = (x \lor y) \land (x \lor z)\) and \( x \land (y \lor z) = (x \land y) \lor (x \land z)\) (distributive laws)
5. \(\ x \lor y = y \lor x\ \) and \( x \land y = y \land x \) (commutative laws)

Examples of Boolean Algebras are:

1. The algebra of all subsets of a set \( U \), with \( + = \cup \), \( \cdot = \cap \), \(- = \text{complement}, \) \( 0 = \emptyset \), \( 1 = U \).
2. The algebra of propositions with symbols \( p_1, p_2, \ldots, p_n \), with \( + = \lor \), \( \cdot = \land \), \( - = \neg \), \( 0 = F \), \( 1 = T \).
3. If \( B_1, \ldots, B_n \) are Boolean Algebras, so is \( B_1 \times \ldots \times B_n \), 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.

- **Inverter:** \( x \rightarrow -x \)
- **OR gate:** \( x \lor y \)
- **AND gate:** \( x \land y \)

Example: How can we build a circuit that computes the function \( xy + (-x)y \)?

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 \( xy + (-x)(-y) \) will do the job.

Multi switch light circuit

This is because if both \( x \) and \( y \) are "on" (1) or "off" (0) \( xy + (-x)(-y) \) will be 1, and otherwise will be 0.

We can generalize this method. For three switches the Boolean expression \( xyz + x(-y)(-z) + (-x)y(-z) + (-x)(-y)z \) will work.

Can you draw circuits implementing these expressions? (see pp. 625, 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 get 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 \( (x+y)(-xy) \), or \( (x+y)(-x + -y) \).

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
\[
xy c_0 + x(-y)(-c_0) + (-x)y(-c_0) + (-x)(-y)c_0
\]
would work, with carry bit
\[
xy c_0 + xy(-c_0) + x(-y)c_0 + (-x)y c_0
\]
See p. 827 to check your implementation.

Minimizing Circuits

A Boolean function can be implemented by many different Boolean expressions.

Disjunctive normal form, the sum-of-products expansion we got from the table of values of the expression, is often not the most efficient.

For example, the Boolean expression
\[
x_1(-x_2)x_3 + x_1x_2x_3 + (-x_1)x_3
\]
\[
= x_1((-x_2)+x_2)x_3 + (-x_1)x_3
\]
\[
= x_1x_3 + (-x_1)x_3 = (x_1 + (-x_1))x_3 = x_3
\]
This last expression is a lot easier to compute. No gates required. Much simpler circuit.

Karnaugh Maps and the Quine-McCluskey Method are used for simplifying Boolean expressions.

See section 12.4.

We’ll do some examples on the board.