… and now for the Final Topic:

**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, 1 + 0 = 1, 0 + 1 = 1, 0 + 0 = 0

The **Boolean product**, denoted by \(\cdot\) 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 \leq i \leq n\}, to B is called a **Boolean function of degree n**.

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

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 \(-z\).
Since xy and \(-z\) are Boolean expressions, so is xy + \(-z\).

... 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:

<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: \( F(x, y) = (-x) \cdot y \)

---

**Another Example:** Possible solution I:

\[
F(x, y, z) = -(xz + y)
\]

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

Possible solution II:

\[
F(x, y, z) = (-(xz))(-y)
\]

---

**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 \( y_i = -x_i \).

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

Consider \( F(x, y, z) \) again: \( F(x, y, z) = 1 \) if and only if:

\[
\begin{align*}
x & = y = z = 0 \\
x & = y = 0, z = 1 \\
x & = 1, y = z = 0
\end{align*}
\]

Therefore,

\[
F(x, y, z) = (-x)(-y)(-z) + (-x)(-y)z + x(-y)(-z)
\]

---

**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.

---

**Definition:** The complement of the Boolean function \( F \) is the function \( \neg F \), where \( \neg F(b_1, b_2, \ldots, b_n) = -(F(b_1, b_2, \ldots, b_n)) \).

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

\[
\begin{align*}
(F + G)(b_1, b_2, \ldots, b_n) &= F(b_1, b_2, \ldots, b_n) + G(b_1, b_2, \ldots, b_n) \\
(FG)(b_1, b_2, \ldots, b_n) &= F(b_1, b_2, \ldots, b_n) \cdot G(b_1, b_2, \ldots, b_n)
\end{align*}
\]
**Question:** How many different Boolean functions of degree 1 are there?

**Solution:** There are four of them, \( F_1, F_2, F_3, \) and \( F_4: \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>( F_1 )</th>
<th>( F_2 )</th>
<th>( F_3 )</th>
<th>( F_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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

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

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

**Identities**

There are useful identities of Boolean expressions that can help us to transform an expression \( A \) into an equivalent expression \( B \), e.g.:

<table>
<thead>
<tr>
<th>Identity Name</th>
<th>AND Form</th>
<th>OR Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity Law</td>
<td>( x = x )</td>
<td>( 0 \times x = x )</td>
</tr>
<tr>
<td>Null or Dominance Law</td>
<td>( 0 \times 0 = 0 )</td>
<td>( x = 1 )</td>
</tr>
<tr>
<td>Idempotent Law</td>
<td>( x \times x = x )</td>
<td>( x = x )</td>
</tr>
<tr>
<td>Inverse Law</td>
<td>( \overline{\overline{x}} = x )</td>
<td>( x \times 1 = x )</td>
</tr>
<tr>
<td>Commutative Law</td>
<td>( x = y )</td>
<td>( x + y = y + x )</td>
</tr>
<tr>
<td>Associative Law</td>
<td>( (x \times y) \times z = x \times (y \times z) )</td>
<td>( (x \times (y + z)) \times (x \times y) \times (x \times y) \times (x \times z) \times (y \times z) )</td>
</tr>
<tr>
<td>Distributive Law</td>
<td>( x + (y \times z) = (x + y) \times (x + z) )</td>
<td>( x \times (y + z) = (x \times y) \times (x \times z) )</td>
</tr>
<tr>
<td>Absorption Law</td>
<td>( x \times y = x )</td>
<td>( x + 0 = x )</td>
</tr>
<tr>
<td>DeMorgan's Law</td>
<td>( \overline{x + y} = \overline{x} \times \overline{y} )</td>
<td>( \overline{x \times y} = \overline{x} + \overline{y} )</td>
</tr>
</tbody>
</table>

**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.
Logic Gates

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

- **Inverter**
  \[
  x \rightarrow \overline{x}
  \]

- **Or gate**
  \[
  x \lor y \rightarrow x + y
  \]

- **And gate**
  \[
  x \land y \rightarrow xy
  \]

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