Why study Boolean Algebra?

It is highly desirable to find the simplest circuit implementation with the smallest number of gates or wires.

We can use Boolean minimization process to reduce a Boolean function (expression) to its simplest form: The result is an expression with the fewest literals and thus less wires in the final gate implementation.

Boolean Algebra (continued)

- George Boole (1815-1864), a mathematician introduced a systematic treatment of logic.

- He developed a consistent set of postulates that were sufficient to define a new type of algebra: Boolean Algebra (similar to Linear Algebra)

- Many of the rules are the same as the ones in Linear Algebra.
Laws of Boolean Algebra

• There are 6 fundamental laws, or axioms, used to formulate various algebraic structures:

1. **Closure**: Boolean algebra operates over a field of numbers, \( B = \{0,1\} \):
   
   For every \( x, y \) in \( B \):
   
   - \( x + y \) is in \( B \)
   - \( x \cdot y \) is in \( B \)

   \[ \begin{array}{c}
   (1,0) \rightarrow (1,0) \\
   (1,0) \rightarrow (1,0)
   \end{array} \]  

   » Similar to Linear Algebra

Laws of Boolean Algebra (continued)

2. **Commutative laws**: For every \( x, y \) in \( B \),
   
   - \( x + y = y + x \)
   - \( x \cdot y = y \cdot x \)

   » Similar to Linear Algebra

\[ \begin{array}{c}
\overset{x}{\rightarrow} F \equiv x + y \\
\overset{y}{\rightarrow} F \equiv x + y
\end{array} \]  

\[ \begin{array}{c}
\overset{x}{\rightarrow} F \equiv x \cdot y \\
\overset{y}{\rightarrow} F \equiv x \cdot y
\end{array} \]
Laws of Boolean Algebra (continued)

3. Associative laws: For every $x, y, z$ in $B$,
- $(x + y) + z = x + (y + z) = x + y + z$
- $(xy)z = x(yz) = xyz$

  » Similar to Linear Algebra

4. Distributive laws: For every $x, y, z$ in $B$,
- $x + (y.z) = (x + y)(x + z)$ [+ is distributive over .] 
  » NOT Similar to Linear Algebra
- $x.(y + z) = (x.y) + (x.z)$ [ . is distributive over + ]
  » Similar to Linear Algebra
 Laws of Boolean Algebra (continued)

5. Identity laws:
   - A set $B$ is said to have an identity element with respect to a binary operation $\{\cdot\}$ on $B$ if there exists an element designated by 1 in $B$ with the property: $1 \cdot x = x$
     
     Example: AND operation

   - A set $B$ is said to have an identity element with respect to a binary operation $\{+\}$ on $B$ if there exists an element designated by 0 in $B$ with the property: $0 + x = x$
     
     Example: OR operation

   » Similar to Linear Algebra

 Laws of Boolean Algebra (continued)

6. Complement

   For each $x$ in $B$, there exists an element $x'$ in $B$ (the complement of $x$) such that:
   - $x + x' = 1$
   - $x \cdot x' = 0$

   » Similar to Linear Algebra

   We can also use $\bar{x}$ to represent complement.
## Laws of Boolean Algebra (Summary)

<table>
<thead>
<tr>
<th><strong>Commutative</strong></th>
<th><strong>Identity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + y = y + x$</td>
<td>$x + 0 = x$</td>
</tr>
<tr>
<td>$xy = yx$</td>
<td>$x . 1 = x$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Associative</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x + y) + z = x + (y + z)$</td>
</tr>
<tr>
<td>$(xy)z = x(yz)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Distributive</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + (yz) = (x + y)(x + z)$</td>
</tr>
<tr>
<td>$x(y + z) = (xy) + (xz)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Identity</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + 0 = x$</td>
</tr>
<tr>
<td>$x . 1 = x$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Complement</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + \overline{x} = 1$</td>
</tr>
<tr>
<td>$x . \overline{x} = 0$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>OR with 1</strong></th>
<th><strong>AND with 0</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x + 1 = 1$</td>
<td>$x . 0 = 0$</td>
</tr>
</tbody>
</table>

## Other Theorems

- **Theorem 1(a):**
  
  $x + x = x$
  
  $x + x = x$
  
  $x + x = (x + x) . 1$
  
  $= (x + x)(x + x')$
  
  $= xx + xx'$
  
  $= x + 0$
  
  $= x$

- **Theorem 1(b):**
  
  $x . x = x$
  
  $x . x = x$
  
  $x . x = xx + 0$
  
  $= xx + xx'$
  
  $= x(x + x')$
  
  $= x . 1$
  
  $= x$
Other Theorems (continued)

- Theorem 2(a):
  \[ x + 1 = 1 \]
  \[ x + 1 = 1 \cdot (x + 1) \]
  \[ = (x + x')(x + 1) \]
  \[ = xx + x'.1 \]
  \[ = x + x' \]
  \[ = 1 \]

- Theorem 2(b):
  \[ x + xy = x \]
  \[ x + xy = x(1 + y) \]
  \[ = x(y + 1) \]
  \[ = x.1 \]
  \[ = x \]

Gate Equivalency and DeMorgan’s Law

\[ x' \cdot y' = (x + y)' \]

\[ x' + y' = (x \cdot y)' \]
Digital Logic Q’s & A’s

Q: Why is Gate Equivalency useful?

A: It allows us to build functions using only one gate type.

Q: Why are digital circuits constructed with NAND/NOR rather than with AND/OR?

A: NAND and NOR gates are smaller, faster, and easier to fabricate with electronic components. They are the basic gates used in all IC digital logic.

Digital IC’s – Transistor Level

\[
z = x \cdot y
\]

\[
x \text{ or } y: \text{‘low’} \\
\text{transistor 1 or 2 is OFF} \\
\text{transistor 3 or 4 is ON} \\
\Rightarrow z = \text{‘high’}
\]

\[
x \text{ and } y: \text{‘high’} \\
\text{transistor 1 and 2 are ON} \\
\text{transistor 3 and 4 are OFF} \\
\Rightarrow z = \text{‘low’}
\]
Digital IC’s (continued)

\[ z = a + b \]

\[ z = (x+y) \cdot z \]

Implementation of Boolean Functions

Example 1: \( F_1 = x + y' \cdot z \)
Implementation of Boolean Functions

Example 2: \[ F_1 = x'y'z + x'yz + xy' \]

Try another implementation using a simplified \( F_2 \):

\[ F_2 = x'y'z + x'yz + xy' \]

\[ = x'z(y'+y) + xy' \]

\[ = x'z(1) + xy' \]

\[ = x'z + xy' \]

What are the advantages of this implementation?

This implementation has fewer gates and fewer inputs to the gates (or wires) than the previous one.
Simplifying Boolean Functions

- Simplify the following Boolean function to a minimum number of terms: \( F_3 = xy + x'z + yz \)

\[
F_3 = xy + x'z + yz \\
= xy + x'z + yz(x + x') \\
= xy + x'z + xyz + x'y'z \\
= xy(1 + z) + x'z(1 + y) \\
= xy + x'z
\]

More on complements (DeMorgan)

- Find the complement of: \( F = (AB'+C)D+E \)

\[
F' = [(AB'+C)D+E]' \\
= [(AB'+C)' + D]E' \\
= [(A'+B'C') + D]E' \\
= (A'+B'C' + D)E' \\
= (A'+B)C'E' + DE'
\]

- Show that the complement of \( x(x + y) = x' \)

\[
[x(x + y)]' = x' + (x + y)' \\
= x' + x'y' \\
= x'(1 + y') \\
= x'(1) = x'
\]
Implementation of Boolean Functions

- Draw the logic diagram for the following function: $F = (a.b) + (b.c)$

\[ F = \overline{a\cdot\overline{b}} + \overline{b\cdot\overline{c}} \]

Implementation of Boolean Functions

- Using ONLY NAND gates, draw a schematic for the following function: $F = (a.b) + (b.c)$

\[
(F')' = [(a.b) + (b.c)]'
= [(a\cdot\overline{b})' \cdot (b\cdot\overline{c})']
\]

\[ F = \overline{a\cdot\overline{b}} + \overline{b\cdot\overline{c}} \]
**Implementation of Boolean Functions**

- Using only OR and NOT gates, draw a schematic for the following function: \( F = xy + x'y' + y'z \)

\[ (F')' = ((xy + x'y' + y'z)'')' \]
\[ = [(xy)'(x'y')'(y'z)']' \]
\[ = [(x'+y')(x+y)(y+z)]' \]
\[ = (x'+y')'(x+y)'(y+z)' \]

![Schematic diagram](image)

**Minterms and Maxterms**

- MINTERMS AND MAXTERMS:

  \( n \) binary variables can be combined to form \( 2^n \) terms (AND terms), called minterms (SOP).

  In a similar fashion, \( n \) binary variables can be combined to form \( 2^n \) terms (OR terms), called maxterms (POS).

  * Note that each maxterm is the complement of its corresponding minterm and vice versa.
Minterms and Maxterms (continued)

Table 2-3: Minterms and Maxterms for Three Binary Variables

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>minterms</th>
<th>Maxterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$x'y'z'$</td>
<td>$m_0$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$x'y'z$</td>
<td>$m_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$x'yz'$</td>
<td>$m_2$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$x'yz$</td>
<td>$m_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$xy'z'$</td>
<td>$m_4$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$xyz'$</td>
<td>$m_5$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$xyz$</td>
<td>$m_6$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$xyz$</td>
<td>$m_7$</td>
</tr>
</tbody>
</table>

Σminterms and Πmaxterms

- Given the truth table, express $F_1$ in sum of minterms

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$F_1$</th>
<th>$F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$F_1(x, y, z) = \Sigma(1,4,5,6,7) = m_1 + m_4 + m_5 + m_6 + m_7$

$= (x'y'z) + (xy'z') + (xy'z) + (xyz') + (xyz)$

- Find $F_2$
\textbf{Σminterms and Πmaxterms}

- Repeat for product of maxterms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>$F_1$</th>
<th>$F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>[0]</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>[0]</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$F_1(x, y, z) = \prod(0,2,3) = M_0 \cdot M_2 \cdot M_3$

$= (x + y + z)(x + y' + z)(x + y' + z')$

\textbf{Σminterms and Πmaxterms}

Express the Boolean function $F = x + y' z$ in a sum of minterms, and then in a product of Maxterms.

$x = x(y + y') = xy + xy'$

$xy = xy(z + z') = xyz + xyz'$

$xy' = xy'(z + z') = xy' z + xy' z'$

$y' z = y' z(x + x') = xy' z + x' y' z$

Adding all terms and excluding recurring terms:

$F(x, y, z) = x' y' z + xy' z' + xy' z + xyz' + xyz$ \hspace{1cm} \text{(SOP)}$

$F(x, y, z) = m_1 + m_4 + m_5 + m_6 + m_7 = \Sigma(1,4,5,6,7)$

Product of maxterms (POS)? $\prod(0,2,3) = M_0 \cdot M_2 \cdot M_3$
XOR Logic gate

3-input exclusive-OR (XOR) logic gate:

\[ F = X \oplus Y \oplus Z \]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( Z )</th>
<th>( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>