Logic Gates and Boolean Algebra

• Logic Gates
  – Inverter, OR, AND, Buffer, NOR, NAND, XOR, XNOR

• Boolean Theorem
  – Commutative, Associative, Distributive Laws
  – Basic Rules

• DeMorgan’s Theorem

• Universal Gates
  – NAND and NOR

• Canonical/Standard Forms of Logic
  – Sum of Product (SOP)
  – Product of Sum (POS)
  – Minterm and Maxterm
Boolean Theorem

• Commutative Law
  – In terms of the result, the order in which variables are ORed or ANDed makes no difference.

\[ A + B = B + A \]

\[ AB = BA \]
Boolean Theorem

- **Associative Law**
  - When ORing or ANDing more than two variables, the result is the same regardless of the grouping of the variables.

\[ A + (B + C) = (A + B) + C \]

\[ A(BC) = (AB)C \]
Boolean Theorem

• Distributive Law
  – A common variable can be factored from an expression just as in ordinary algebra.

\[ AB + AC = A(B + C) \]
# Boolean Theorem

## Basic Rules

1. \( A + 0 = A \)
2. \( A + 1 = 1 \)
3. \( A \cdot 0 = 0 \)
4. \( A \cdot 1 = A \)
5. \( A + A = A \)
6. \( A + \overline{A} = 1 \)
7. \( A \cdot A = A \)
8. \( A \cdot \overline{A} = 0 \)
9. \( \overline{\overline{A}} = A \)
10. \( A + AB = A \)
11. \( A + \overline{AB} = A + B \)
12. \( (A + B)(A + C) = A + BC \)
Boolean Simplification - Example

- Using boolean theorem, Simplify the expression:

\[ AB + A(B + C) + B(B + C) \]

Apply distributive law,

\[ AB + AB + AC + BB + BC \]

Apply rule 7 (BB = B), and rule 5 (AB + AB = AB)

\[ AB + AC + B + BC \]

Apply rule 10 (B + BC = B)

\[ AB + AC + B \]
Boolean Simplification - Example

\[ AB + AC + B \]

Apply rule 10 (\( AB + B = B \))

\[ B + AC \]

At this point, the expression is simplified as much as possible

Original expression is \( AB + A(B + C) + B(B + C) \)

Which is logically equal to \( B + AC \)

what is the advantage of Boolean simplification?
Boolean Simplification - Example

Original expression is \( AB + A(B + C) + B(B + C) \)

Which is logically equal to \( B + AC \)

Faster
Compact design
Lower cost
Boolean Simplification - Example

• Applying boolean theorem for logic simplification depends on a thorough knowledge of boolean algebra, with some ingenuity and cleverness

• Please look at Floyd’s book examples 4-10, 4-11, and 4-12, as well as some exercises in the book to gain experience
DeMorgan’s Theorem

• The complement of a product of variables is equal to the sum of the complemented variables

Theorem 1

\[ \overline{A \cdot B} = \overline{A} + \overline{B} \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>\overline{A \cdot B}</th>
<th>\overline{A} + \overline{B}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
DeMorgan’s Theorem

Theorem 2

\[ \overline{A + B} = \overline{A} \cdot \overline{B} \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( \overline{A + B} )</th>
<th>( \overline{A} \cdot \overline{B} )</th>
</tr>
</thead>
<tbody>
<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>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>0</td>
</tr>
</tbody>
</table>
Universal Gates

• NAND and NOR gates are known as Universal gates because all logic gates can be represented by NAND and NOR
• NAND and NOR are the cheapest and smallest to manufacture in Integrated Circuits compared to AND and OR
• Therefore NAND and NOR are always used in practical circuit design
NAND Universal Gates

• How to represent inverter using NAND gates?

NAND gate truth table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z = \overline{X \cdot Y}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Short the inputs together
NAND Universal Gates

• How to represent OR gate using NAND gates?

Logic expression for OR gate:

\[ X + Y \equiv \overline{X} \cdot \overline{Y} \]

Using DeMorgan’s Theorem,
NAND Universal Gates

• How to represent AND gate using NAND gates?

Logic expression for AND gate:

\[ X \cdot Y \equiv \overline{X \cdot Y} \]
NOR Universal Gates

\[ X \rightarrow Z \]

\[ X \quad Y \rightarrow Z \]

\[ X \quad Y \rightarrow Z \]

\[ X \quad Y \rightarrow Z \]