UNIT 2
BOOLEAN ALGEBRA

Introduction

- Boolean algebra
  - Is the basic mathematics for logic design of digital systems
  - Differs from ordinary algebra in the values, operations, and laws
- History
  - George Boole developed Boolean algebra in 1847 and used it to solve problems in mathematical logic
  - British mathematician and philosopher
  - Claude Shannon first applied Boolean algebra to the design of switching circuits in 1939
    - American electrical engineer and mathematician
    - Master's thesis (21 years-old)
- In this unit, you will learn how to...
  - Use a truth table
  - Manipulate basic operations and apply laws of Boolean algebra
  - Relate Boolean expressions to basic logic gates

Boolean Algebra

Contents

- Introduction
- Basic operations
- Boolean expressions and truth tables
- Theorems and laws
  - Basic theorems
  - Commutative, associative, and distributive laws
  - Simplification theorems
  - Multiplying out and factoring
  - DeMorgan's laws

Reading

- Unit 2

Boolean Variables

- Switching devices we will use are generally two-state devices

- Represent an input or output by a Boolean variable
  - Can take on only two different values ⇒ Switching algebra
  - 1/0 for High/Low or True/False or Yes/No
    - Just symbols, NO numeric values
    - cf. binary arithmetic in Unit 1
### Basic Operations

**NOT, AND, OR**

Propositional logic

---

**Boolean Algebra**

Boolean algebra differs from ordinary algebra in values, operations, laws.

#### Operation -- Logic NOT

- **Complement = Inverse = Negate = NOT (\(^{'}\); \(\overline{\text{\(x\)}}\); \(\sim\); \(\neg\))
  - \(0' = 1, 1' = 0\)
  - **Symbol**
    - \(x'\)

- **Truth table**
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Truth table**
<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Basic switch**
  - \(A = 0 \Rightarrow \text{switch open}\)
  - \(A = 1 \Rightarrow \text{switch closed}\)

---

#### Operation -- Logic AND

- **AND (\(\cdot\); \(\land\))**
  - \(0 \cdot 0 = 0, 0 \cdot 1 = 0, 1 \cdot 0 = 0, 1 \cdot 1 = 1\)
  - **Symbol**
    - \(A \cdot B\)
  - **Truth table**
    | Inputs | Outputs |
    |--------|---------|
    | 0      | 0       |
    | 0      | 1       |
    | 1      | 0       |
    | 1      | 1       |

- **Switch (in series)**
  - \(T = 0 \Rightarrow 1 \rightarrow 2 \text{ open}\)
  - \(T = 1 \Rightarrow 1 \rightarrow 2 \text{ closed}\)

---

#### Operation -- Logic OR

- **OR (+; \(\lor\))**
  - \(0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 1\)
  - **Symbol**
    - \(A + B\)
  - **Truth table**
    | Inputs | Outputs |
    |--------|---------|
    | 0      | 0       |
    | 0      | 1       |
    | 1      | 0       |
    | 1      | 1       |

- **Switch (in parallel)**
  - \(T = 0 \Rightarrow 1 \rightarrow 2 \text{ open}\)
  - \(T = 1 \Rightarrow 1 \rightarrow 2 \text{ closed}\)

---

**Switch A closed and switch B closed**

**Switch A closed or switch B closed**
### Boolean Expressions vs. Logic Gates

- **A Boolean expression** is formed by basic operations on variables or constants, e.g., the simplest one: 0, 1, X, Y.

- **Realize a Boolean expression by a circuit of logic gates**
  - Perform operations in order: Parentheses → NOT → AND → OR
  - e.g., \( AB' + C \) → 1. \( B' \) → 2. \( AB' \) → 3. \( AB' + C \)

- e.g., \( [A(C + D)]' + BE \) → 1. \( C + D \) → 2. \( A(C + D) \) → 3. \( [A(C + D)]' \) → 4. \( BE \) → 5. \( [A(C + D)]' + BE \)

### Boolean Expressions vs. Truth Tables

- **A truth table** specifies the output values of a Boolean expression for all possible combinations of input values.

- **Check the equivalence between two expressions**
  - e.g., \( AB' + C = (A + C)(B' + C) \)

- **Example:** \( F = A' + B \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>B'</th>
<th>( AB' )</th>
<th>( AB' + C )</th>
<th>( A + C )</th>
<th>( B' + C )</th>
<th>( (A + C) )</th>
<th>( (B' + C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<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>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Expressions show the condition to make output == 1**

- **One function has different expressions**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>( A' )</th>
<th>( A' + 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>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Check the equivalence between two expressions**
  - e.g., \( AB' + C = (A + C)(B' + C) \)
Boolean algebra differs from ordinary algebra in values, operations, and laws.

**Basic Theorems (1/3)**
- **Operations with 0 and 1**
  - $X + 0 = X$
  - $X \cdot 1 = X$
  - $X + 1 = 1$
  - $X \cdot 0 = 0$
  - e.g., $(AB' + D)E + 1 = 1$

**Basic Theorems (2/3)**
- **Idempotent laws**
  - $X + X = X$
  - $X \cdot X = X$

**Basic Theorems (3/3)**
- **Involution law**
  - $(X')' = X$
- **Laws of complementarity**
  - $X + X' = 1$
  - $X \cdot X' = 0$
  - e.g., $(AB' + D)(AB' + D)' = 0$
Commutative/Associative Laws

- **Commutative laws for AND and OR**
  - $XY = YX$
  - $X + Y = Y + X$

- **Associative laws for AND and OR**
  - $(XY)Z = X(YZ) = XYZ$
  - $(X + Y) + Z = X + (Y + Z) = X + Y + Z$
  - e.g., 2-input gates $\Rightarrow$ multiple-input gates

**Distributive Laws (1/2)**

- **Ordinary distributive law**
  - $X(Y + Z) = XY + XZ$

- **Second distributive law (Important !)**
  - $X + YZ = (X + Y)(X + Z)$
  - Proof?

**Distributive Laws (2/2)**

- **Prove a Boolean theorem/law by:**
  1. Truth table
  2. Basic theorems

**Simplification Theorems**

- **Useful simplification theorems**
  - $XY + XY' = X$
  - $X + XY = X$
  - $(X + Y) Y = XY$
  - $(X + Y')(X + Y) = X$
  - $X(X+Y) = X$
  - $XY' + Y = Y + X$
  - $X + XY = X(1+Y) = X(1+1) = X$
  - $X(X+Y) = XX + XY = X + XY = X$
  - $(Y+X)(Y+Y') = (Y+X) \cdot 1 = Y + X$

- **Proof:**
  - Use switches

- **Duality**

- **If switch $Y$ open $\Rightarrow$ switch $Y'$ closed**
**Simplification Examples**

1. \( A(A' + B) = AB \)

2. \( Z = [A + B'C + D + EF][A + B'C + (D + EF)'] \)
   \[ = [A + B'C + D + EF][A + B'C + (D + EF)'] \]
   \[ = [X + Y][X + Y'] \]
   \[ = X = A + B'C \]

3. \( Z = (AB + C)(B'D + C'E') + (AB + C)' \)
   \[ = (AB + C)(B'D + C'E') + (AB + C)' \]
   \[ = XY + X'Y = (X+Y)Y + X' \]
   \[ = Y + X' = B'D + C'E' + (AB + C)' \)

**Multiplying Out**

- **Use the ordinary distributive law**
  \( X(Y + Z) = XY + XZ \)
  to multiply out an expression to obtain a sum-of-products form

- **Example:**
  \( (A + B)(C + D + E) \)
  \[ = (A + B)(C + D + E) \]
  \[ = (A + B)(C + D + E) \]
  \[ = A + BC + BCD + BCE \]
  \[ = A + BC + BCD + BCE \]

- **Factoring**
  - **Use the second distributive law**
    \( X + YZ = (X + Y)(X + Z) \)
    to factor an expression to obtain a product-of-sums form

- **Example:**
  \( (A + B)(C + D + E) \)
  \[ = (A + B)(C + D + E) \]
  \[ = (A + B)(C + D + E) \]
  \[ = A + BC + BCD + BCE \]
  \[ = A + BC + BCD + BCE \]
  \[ = A + BC + BCD + BCE \]

**Boolean Algebra**

- **SOP:** Sum of products of only single variables
- **POS:** Products of sums of only single variables
Example: Factoring

1. Factor $A + B'CD$
   
   $A + B'CD = (A + B')(A + CD) = (A + B')(A + C)(A + D)$

2. Factor $AB'C'D$
   
   $AB'C'D = (AB' + C'D)(AB' + D) = (A + C')(B' + C')(A + D)(B' + D)$

3. DIY: Factor $C'D + C'E' + G'H$

   
   Iteratively apply 2nd distributed law

SOP vs. Logic Gates

- Realize SOPs by two-level circuits (AND-OR)
  - $AB' + C'D + E + AC'E'$
  - $A + B' + C + D'E$

- Realize POSs by two-level circuits (OR-AND)
  - $(A + B')(C + D + E)(A + C' + E')$
  - $(X + Y + X_n) = X' + Y'$

DeMorgan's Laws

- Complement a Boolean expression by DeMorgan's laws
  - $(X + Y)' = X'Y'$
  - $(XY)' = X' + Y'$

- Generalize to $n$ variables:
  - $(X_1X_2...X_n)' = X_1'X_2'...X_n'$
  - $(X_1X_2...X_n)' = X_1' + X_2' + ... + X_n'$

- One-step rule:
  - $[(x_1x_2...x_n, 0, 1, +, \cdot)]' = f(x_1', x_2', ..., x_n', 1, 0, +)$
  - $x \leftrightarrow x'$; $+ \leftrightarrow \cdot$; $0 \leftrightarrow 1$
Example: Complementing $(AB' + C)D' + E$

1. Iteratively apply DeMorgan's laws:
   
   \[
   ((AB' + C)D' + E)' = ([AB' + C]' + D')'E' = ([AB' + C]' + D')E' = (A' + B)(C') + D'E' \]

2. Or, use one-step rule:

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

NOT is applied only to single variables

Duality

- Dual:
  \[ ([x_1, x_2, ..., x_n, 0, 1, +, *])′ = f(x_1, x_2, ..., x_n, 1, 0, +) \]

- Cf. DeMorgan's laws:
  \[ ([x_1, x_2, ..., x_n, 0, 1, +, *])' = f(x_1', x_2', ..., x_n', 1, 0, +) \]

- Find the dual of an expression:
  - Complement the entire expression
  - Complement each individual variable

- e.g., $(XYZ...)' = X + Y + Z + ...$
- e.g., $(AB' + C)' = ?$
- $(AB' + C)' = (A' + B)C' \Rightarrow (AB' + C)' = (A + B')C$

Application: $F = G \iff F' = G'$

Homework for Unit 2

- Problems
  - 2.13
  - 2.14
  - 2.16
  - 2.30
- Homework #1 covers Units 1--3

Comprehensive Example: 2.2(a)

Simplify $F$: 

- $A \cdot B \cdot C \cdot D \oplus E \oplus F$