Digital Logic Design

ENGG1015

1st Semester, 2011

Dr. Kenneth Wong

Department of Electrical and Electronic Engineering
Representing Logic Operations

• Each function can be represented equivalently in 3 ways:
  – Truth table
  – Boolean logic expression
  – Schematics
Determining output level from a diagram

\[
\begin{align*}
A &= 0 \\
B &= 1 \\
C &= 1 \\
D &= 1 \\
x &= 0
\end{align*}
\]
Implementing Circuits From Boolean Expressions

- When the operation of a circuit is defined by a Boolean expression, we can *draw a logic-circuit* diagram directly from that expression.
- Example: draw the circuit for  \( y = AC + \overline{BC} + \overline{ABC} \)
  - Done in two steps

![Logic circuit diagram](image)
Truth Table to Boolean Expression

- List all combinations that give 1 at output
- Sum up all terms
- Sum of products (SOP) – more later

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>x</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>0</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>

\[
\begin{align*}
\overline{A}\overline{B}C + \overline{A}B\overline{C} + ABC \\
x &= \ ?
\end{align*}
\]
Laws of Boolean Algebra

• Commutative Laws

\[ A + B = B + A \]

\[ A \cdot B = B \cdot A \]

• Associative Laws

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

\[ A \cdot (B \cdot C) = (A \cdot B) \cdot C \]
• Distributive Law

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

**Rules of Boolean Algebra**

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. \( \overline{A} = A \)
10. \( A + AB = A \)
11. \( A + \overline{A}B = A + B \)
12. \( (A + B)(A + C) = A + BC \)
• Rule 1

\[ X = A + 0 = A \]

\[ A = 1 \quad X = 1 \quad A = 0 \quad X = 0 \]

\[ A = 1 \quad X = 1 \quad A = 0 \quad X = 1 \]

• Rule 2

\[ X = A + 1 = 1 \]

\[ 0 \quad 0 \quad 0 \]
\[ 0 \quad 1 \quad 1 \]
\[ 1 \quad 0 \quad 1 \]
\[ 1 \quad 1 \quad 1 \]

OR Truth Table

• Rule 3

\[ X = A \cdot 0 = 0 \]

\[ A = 1 \quad X = 0 \quad A = 0 \quad X = 0 \]

\[ A = 0 \quad X = 0 \quad A = 1 \quad X = 1 \]

• Rule 4

\[ X = A \cdot 1 = A \]

\[ 0 \quad 0 \quad 0 \]
\[ 0 \quad 1 \quad 0 \]
\[ 1 \quad 0 \quad 0 \]
\[ 1 \quad 1 \quad 1 \]

AND Truth Table
• Rule 5

\[ X = A + A = A \]

\[ X = A + \overline{A} = 1 \]

• Rule 6

\[ X = A \cdot A = A \]

\[ X = A \cdot \overline{A} = 0 \]

• Rule 7

\[ X = A + A = A \]

\[ X = A + \overline{A} = 1 \]

• Rule 8
• **Rule 9**

• **Rule 10: A + AB = A**
  - Proof: $A + AB = A(1 + B)$ by distributive law
    
    $= A \cdot 1$ by rule 2
    
    $= A$ by rule 4
  - Or prove by truth table

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$AB$</th>
<th>$A + AB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

straight connection
• **Rule 11:** \[ A + \overline{AB} = A + B \]

• **This rule can only be proved by constructing the truth table.**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>\overline{AB}</th>
<th>A + \overline{AB}</th>
<th>A + B</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \text{equal} \]

• **Rule 12:** \[ (A + B)(A + C) = A + BC \]

• **This rule can only be proved by constructing the truth table.**

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

\[ \text{equal} \]
• **Examples:**
  - **Simplify** $y = ABD + ABD$
    
    $y = A\overline{B}(D + \overline{D})$
    
    $= AB \cdot 1$
    
    $= \overline{AB}$
    
    by distributive law
    
    by rule 6
    
    by rule 4

  - **Simplify** $z = (\overline{A} + B)(A + B)$

    $z = \overline{A}A + \overline{A}B + BA + BB$
    
    $= 0 + \overline{A}B + BA + B$
    
    $= AB + BA + B$
    
    $= B(A + A + 1)$
    
    $= B \cdot 1$
    
    $= B$
    
    by distributive law
    
    by rule 8 and rule 7
    
    by rule 1
    
    by distributive law
    
    by rule 6 and rule 2
    
    by rule 4

  - **Simplify** $x = A\overline{CD} + A\overline{BCD}$

    $x = CD(A + \overline{A}B)$
    
    $= CD(A + B)$
    
    $= A\overline{CD} + B\overline{CD}$
    
    by distributive law
    
    by rule 11
    
    by distributive law
Review Questions

• Simplify
  \[ y = \overline{AC} + AB \overline{C} \]
  \[ y = \overline{AC} \]

• Simplify
  \[ y = \overline{ABCD} + \overline{ABCD} \]
  \[ y = \overline{ABD} \]

• Simplify
  \[ y = \overline{AD} + ABD \]
  \[ y = \overline{AD} + BD \]
DeMorgan’s Theorems

• **Theorem 1**
  \[ x + y = \overline{x \cdot y} \]

  **Remember:**
  “Break the bar, change the operator”

• **Theorem 2**
  \[ x \cdot y = \overline{x + y} \]

  – DeMorgan's theorem is very useful in digital circuit design
  – It allows **ANDs to be exchanged with ORs by using invertors**
  – DeMorgan's Theorem can be extended to any number of variables. E.g, for three variables \( x, y \) and \( z \)
    \[ x + y + z = \overline{x \cdot y \cdot z} \]
    \[ x \cdot y \cdot z = \overline{x + y + z} \]
Quick Quiz (1)

- **Simplify** \((\overline{A\overline{B}} + C)\)
  - A
  - B
  - C
  - D
  - AC + BC
  - \(\overline{A\overline{C}} + B\overline{C}\)
  - \(\overline{A C} + B\overline{C}\)
  - \(\overline{A C} + B C\)

- **Simplify** \((\overline{A} + C) \cdot (B + \overline{D})\)
  - A
  - B
  - C
  - D
  - \(\overline{A\overline{C}} + B\overline{D}\)
  - \(\overline{A\overline{C}} + B\overline{D}\)
  - \(\overline{A\overline{C}} + B\overline{D}\)
  - AC + B\overline{D}
  - \(\overline{A C} + B\overline{D}\)
Quick Quiz (2)

- Simplify \( AB \cdot CD \cdot EF \)
  - \( AB + CD + EF \) [Red]
  - \( AB + CD + EF \) [Blue]
  - \( \overline{AB} + \overline{CD} + \overline{EF} \) [Black]
  - \( ABCDEF \) [Circle]

- Determine the output expression for the below circuit and simplify it using DeMorgan’s Theorem
  - \( ABC \) [Red]
  - \( \overline{ABC} \) [Blue]
  - \( A + B + \overline{C} \) [Black]
  - \( \overline{A + \overline{B} + C} \) [Red]
  - \( \overline{A + B + C} \) [Circle]
Examples (Summary):

\[ a) \quad (A\overline{B} + C) = A\overline{B} \cdot \overline{C} = (\overline{A} + \overline{B}) \cdot \overline{C} = \overline{A}\overline{C} + B\overline{C} \]

\[ b) \quad (\overline{A} + C) \cdot (B + \overline{D}) = (\overline{A} + C) + (B + \overline{D}) = \overline{A} \cdot \overline{C} + B \cdot \overline{D} = A\overline{C} + B\overline{D} \]

\[ c) \quad A + \overline{B} \cdot C = \overline{A} \cdot (\overline{B} \cdot C) = \overline{A} \cdot (\overline{B} + \overline{C}) = \overline{A}(B + \overline{C}) \]

\[ d) \quad (A + BC) \cdot (D + EF) = (A + BC) + (D + EF) \]

\[ = (\overline{A} \cdot \overline{BC}) + (\overline{D} \cdot \overline{EF}) \]

\[ = \overline{A} \cdot (\overline{B} + \overline{C}) + \overline{D} \cdot (\overline{E} + \overline{F}) \]

\[ = \overline{A} \overline{B} + \overline{A} \overline{C} + \overline{D} \overline{E} + \overline{D} \overline{F} \]

\[ e) \quad \overline{A} \overline{B} \cdot \overline{C} \cdot \overline{EF} = \overline{A} \overline{B} + \overline{C} + \overline{EF} = AB + CD + EF \]

\[ f) \quad \text{Determine the output expression for the below circuit and simplify it using DeMorgan’s Theorem} \]

\[ z = A \cdot \overline{B} \cdot \overline{C} = \overline{A} + \overline{B} + \overline{C} = \overline{A} + B + C \]
Review Questions

- Using DeMorgan’s Theorems to convert the expressions to one that has only single-variable inversions.

\[
\begin{align*}
y &= \overline{RST} + Q \\
z &= (\overline{A + B}) \cdot \overline{C} \\
y &= A + \overline{B} + \overline{CD}
\end{align*}
\]

\[
\begin{align*}
y &= (\overline{R} + S + \overline{T})Q \\
z &= \overline{A}B + C \\
y &= \overline{A}B(C + \overline{D})
\end{align*}
\]
Implications of DeMorgan’s Theorems

\[ x + y = \overline{x \cdot y} \]

- i.e., The AND gate with INVERTERs on each of its inputs is equivalent to a NOR gate

\[ x \cdot y = \overline{x + y} \]

- i.e., The OR gate with INVERTERs on each of its inputs is equivalent to a NAND gate
Universality of NAND gates

• Any expression can be implemented using combinations of OR gates, AND gates and INVERTERs
• However, it is also possible to implement any logic expression using only NAND gates and no other type of gate
• This is because NAND gates, in proper combination, can perform Boolean operations OR, AND, and INVERTER
• Example: Suppose we want to implement \( x = AB + CD \).

  -- If we directly implement the expression, we need 2 AND gates and 1 OR gate \( \Rightarrow \) we need two ICs

  -- If we transform the circuit into one having only NAND gates, we need only one IC

![IC in Dual-in-line package (DIP)](image)

![Pin diagrams for the ICs containing NAND, AND, and OR gates](image)
Universality of NOR gate

- In a similar manner, it can be shown that NOR gates can be arranged to implement any of the Boolean operations.
Alternate Logic-Gate Representations

- Apart from standard logic gate symbols we have seen so far, there are alternate symbols for the commonly used gates.
- The alternate symbols are obtained by performing the following two steps:
  1. Invert each input and output of the standard symbol. This is done by adding bubbles (small circles) on input and output lines that do not have bubbles and by removing bubbles that are already there.
  2. Change the operation symbol from AND to OR, or from OR to AND. (In the special case of the INVERTER, the operation symbol is not changed.)
Several points to note

• The equivalences can be extended to gates with any number of inputs.
• None of the standard symbols have bubbles on their inputs, and all the alternate symbols do.
• The standard and alternate symbols for each gate represent the same physical circuit; there is no difference in the circuits represented by the two symbols.
• NAND and NOR gates are inverting gates, and so both the standard and the alternate symbols for each will have a bubble on either the input or the output, AND and OR gates are non-inverting gates, and so the alternate symbols for each will have bubbles on both inputs and output.
Why we need alternate representation?

• Proper use of the alternate gate symbols in the circuit diagram can make the circuit operation much more easy to understand.

• E.g., the two circuits on the right are equivalent.

• However, circuit (a) does not facilitate an understanding of how the circuit function.

• Circuit (b) is easy to understand: Z will go HIGH whenever either A=B=1 or C=D=1 (or both).

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</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>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</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>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</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>0</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>
</tr>
</tbody>
</table>
Canonical Form

• Boolean expression can be expressed in many different ways
  
  \[(A + D)(B + C) \equiv AB + AC + BD + CD\]

• Two standard ways:
  – Sum of Product
  – Product of Sum

• Canonical SOP
• Canonical POS
Canonical SOP

• Boolean expression expressed as a sum of product of basic inputs
  – Basic input may optionally negated

\[ \overline{ABC} + \overline{BD} + AD \]

– NOT canonical: \[ A + B(C + D) \]
– No parenthesis

• Most natural for human
Canonical POS

- Boolean expression expressed as a product of sum of basic inputs
  - Basic input may optionally negated

\[
(A + \overline{B} + C)(B + \overline{D})(A + D)
\]

- NOT canonical:
  - Many parenthesis \( A + B(C + D) \)

- Not too natural for human, but equally good for computers.
**Example:** This example illustrate the complete procedure for designing a logic circuit. Suppose the logic circuit having 3 inputs, A, B, C will have its output HIGH only when a majority of the inputs are HIGH.

**Step 1** Set up the truth table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>x</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>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- $\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ → $\overline{ABC}$
- $ABC + \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ → $\overline{ABC}$
- $ABC + \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ → $ABC$

**Step 2** Write the AND term for each case where the output is a 1.

**Step 3** Write the SOP form the output

$$x = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

**Step 4** Simplify the output expression

$$x = \overline{A}BC + ABC + A\overline{B}C + ABC + ABC$$

$$= BC(\overline{A} + A) + AC(\overline{B} + B) + AB(C + C)$$

$$= BC + AC + AB$$

**Step 5** Implement the circuit
Example: Conversion through the opposite direction:

**Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>y</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>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Schematics**

**Boolean Expression**

\[
\overline{ABC} \leftrightarrow \\
\begin{align*}
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
\end{align*} \\
\rightarrow \overline{ABC} \\
\rightarrow \overline{ABC} \\
\rightarrow ABC
\]

**Booleans**

**Step 1** Start from the circuit

**Step 2** Obtain Boolean expression

from the circuit (in SOP form)

\[
y = AC + B\overline{C} + \overline{A}BC
\]

**Step 3** Write the truth table
Forms of Boolean Expressions

• There are two general forms of logic expression: SOP, POS

• Sum-of-products form (SOP)
  – first the product (AND) terms are formed then these are summed (OR)
  – eg: ABC + DEF + GHI

• Product-of-sum form (POS)
  – first the sum (OR) terms are formed then the products are taken (AND)
  – eg: (A+B+C)(D+E+F)(G+H+I)

• It is possible to convert between these two forms using Boolean algebra
Simplifying Logic Circuits

- Once the expression for a logic circuit is obtained, we may try to simplify it, so that the implementation requires fewer gates
- Example: below two circuits are the same, but the second one is much more simpler

![Logic Circuit Diagram](image)

- Two methods for simplifying
  - Algebraic method (use Boolean algebra theorems)
  - Karnaugh mapping method (next lesson)
Minimization by Boolean Algebra

• Make use of relationships and theorems of Boolean algebra to simplify the expressions
  – this method relies on your algebraic skill

• Mainly consists of two steps:
  – The original expression is put into SOP form by repeated application of DeMorgan’s theorems and multiplication of terms
  – Once the original expression is in SOP form, the product terms are checked for common factors, and factoring is performed wherever possible

• Example: Simplify \[ z = ABC + A\overline{B} \cdot (\overline{A}\overline{C}) \]

\[
\begin{align*}
z &= ABC + A\overline{B} \cdot (\overline{A} + \overline{C}) \\
&= ABC + A\overline{B} \cdot (A + C) \quad \text{[by DeMorgan thm]} \\
&= ABC + A\overline{B}A + A\overline{B}C \quad \text{[cancel double inversions]} \\
&= ABC + A\overline{B} + A\overline{B}C \quad \text{[multiply out]} \\
&= ABC + A\overline{B} + A\overline{B}C \quad \text{[A \cdot A = A]} \\
&= AC(B + \overline{B}) + A\overline{B} \\
&= AC + A\overline{B} \quad \text{[B + \overline{B} = 1]} \\
&= A(C + \overline{B})
\end{align*}
\]
Example: Simplify \( z = A\overline{B}\overline{C} + A\overline{B}C + ABC \)

- The expression is already in SOP form

\[
z = A\overline{B}(\overline{C} + C) + ABC \\
= A\overline{B}(1) + ABC \\
= A\overline{B} + ABC \\
= A(\overline{B} + BC) \\
= A(\overline{B} + C) \quad \text{[by rule 11 in previous chapter]}
\]

Example: Simplify \( x = (\overline{A} + B)(A + B + D)\overline{D} \)

\[
x = \overline{A}A\overline{D} + \overline{A}B\overline{D} + \overline{A}D\overline{D} + B\overline{A}\overline{D} + B\overline{B}\overline{D} + B\overline{D}\overline{D} \quad \text{[multiply out]}
\]

\[
= \overline{A}B\overline{D} + B\overline{A}\overline{D} + B\overline{D} \quad \text{[\(\overline{A}A = 0\) and \(D\overline{D} = 0\)]}
\]

\[
= B\overline{D}(\overline{A} + A + 1) \quad \text{[factoring]}
\]

\[
= B\overline{D} \quad \text{[\(\overline{A} + A = 1\) and \(1+1=1\)]}
\]
• **Example: Simplify** $z = \overline{AC(\overline{ABD})} + \overline{ABC\overline{D}} + A\overline{B}C$
  
  – First expand it into SOP form
  
  $$z = \overline{AC(A + \overline{B} + \overline{D})} + \overline{ABC\overline{D}} + A\overline{B}C$$  
  \[\text{[DeMorgan thm]}\]
  
  $$= \overline{ACA} + \overline{ACB} + \overline{ACD} + \overline{ABC\overline{D}} + A\overline{B}C$$  
  \[\text{[multiply out]}\]
  
  $$= \overline{ACB} + \overline{ACD} + \overline{ABC\overline{D}} + A\overline{B}C$$  
  \[\overline{AA} = 0\]
  
  – Then look for the largest common factor between any two or more product terms: first and last terms have $\overline{BC}$, while the second and third terms share $\overline{AD}$
  
  – Grouping the terms gives
  
  $$z = \overline{BC}(\overline{A} + A) + \overline{AD}(C + B\overline{C})$$
  
  $$= \overline{BC} + \overline{AD}(C + B)$$  
  \[\text{[by } \overline{A} + A = 1, \ C + B\overline{C} = C + B\text{]}\]

• **We might think that the above expression is the simplest since it cannot be simplified further**

• **However, in fact, the simplest form of this equation is** $z = \overline{ABD} + \overline{BC}$

• **It turns out that we missed an operation earlier that could have led to the simpler form**

• **Question: How could we have known that we missed a step??**

• **Ans: There is no way we can know. This illustrate the frustration often encountered in Boolean simplification**
In conclusion…

• All logic functions can be represented as (1) truth table (2) schematics (3) Boolean expressions, interchangeably

• Laws of Boolean algebra helps to simplify the Boolean expression

• DeMorgan’s theorems

• NAND/NOR gates are universal gates

• Non-standard representation is equivalent to DeMorgan’s theorems

• Canonical SOP/POS