**UNIT 4**

**MINTERM AND MAXTERM EXPANSIONS**

Iris Hui-Ru Jiang  
Spring 2010

---

**Objectives**

- Design a combinational logic circuit starting with a word description of the desired circuit behavior
- Steps:
  1. Translate the word description into a switching function
     - Boolean expression or truth table
  2. Simplify the function
  3. Realize it using available logic gates
- e.g., Mary watches TV if it is Monday night and she has finished her homework
  - ⇒ Translate...
  - Mary watches TV \((F)\) if
  - it is Monday night \((A)\)
  - and
  - she has finished her homework \((B)\)
  - ⇒ \(F = A \land B\)

---

**Example: Designing an Alarm**

- The alarm will ring iff the alarm switch is turned on and the door is not closed, or it is after 6 P.M. and the window is not closed
- ⇒ Translate...
- The alarm will ring \((Z)\) iff
  - the alarm switch is on \((A)\) and the door is not closed \((B')\)
  - or
  - it is after 6 P.M. \((C)\) and the window is not closed \((D')\)
- ⇒ \(Z = A \land B' + C \land D'\)
Logic Design using a Truth Table

A truth table helps translation

Semantics

Design a detector that outputs 1 when input is greater than 2

Inputs \((A, B, C)\) represent a binary number \(N\);
if \(N = (A, B, C) \geq 3 = 111\) binary, \(f = 1\); otherwise \(f = 0\)

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
<th>(f)</th>
<th>(f')</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ f = A'B'C' + ABC' + AB'C + ABC \] (SOP)
\[ f = A'B' + AC \] (Simplified using Boolean Algebra, Units 2&3)

Threshold Detector (1/2)

Counting 1’s, we have SOP

What if counting 0’s?

Show the condition to make output ==1

\[ f = A'C' + AB'C' + ABC' + ABC \]
\[ f = A'B'C' + A'B'C + ABC' \] (POS)

Taking the complement by DeMorgan's law,
\[ f' = [(A'B'C')' + (A'B'C')']' \]
\[ = ((A'B'C) + (A'B'C))' \]
\[ = (A + B + C)(A + B' + C) \] (POS)

Summary: Logic Design using a Truth Table

1. Make a truth table according to the word description
2. Generate a Boolean expression
   - SOP: check 1’s
   - POS: check 0’s
     - First of all, have \(f'\) in SOP, then derive \(f\) in POS
3. Simplify the Boolean expression

Words
Translation
IC
Minterm & Maxterm Expansions

### Minterm Expansion

- A minterm expansion or a standard sum of products is a function written as a sum of minterms. 
- Implication: counting 1's
- Example:
  - \( f = A'B'C' + A'B'C + AB'C' + ABC \)
  - \( = \sum m(3, 4, 5, 6, 7) \)

<table>
<thead>
<tr>
<th>Row No.</th>
<th>ABC</th>
<th>Minterms ( m_i )</th>
<th>Maxterms ( M_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>( m_0 = A'B'C' )</td>
<td>( M_0 = A + B + C )</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>( m_1 = A'B'C )</td>
<td>( M_1 = A + B' + C )</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>( m_2 = A'B C' )</td>
<td>( M_2 = A + B + C' )</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>( m_3 = A'B C )</td>
<td>( M_3 = A + B + C )</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>( m_4 = AB'C' )</td>
<td>( M_4 = A' + B' + C )</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>( m_5 = AB'C )</td>
<td>( M_5 = A' + B + C )</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>( m_6 = AB C' )</td>
<td>( M_6 = A' + B + C )</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>( m_7 = AB C )</td>
<td>( M_7 = A' + B' + C )</td>
</tr>
</tbody>
</table>

### Maxterm Expansion

- A maxterm expansion or a standard product of sums is a function written as a product of maxterms. 
- Implication: counting 0's
- Example:
  - \( f' = A'B'C' + A'B'C + A'BC' \)
  - \( = \prod M(0, 1, 2) \)

<table>
<thead>
<tr>
<th>Row No.</th>
<th>ABC</th>
<th>Minterms ( m_i )</th>
<th>Maxterms ( M_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>( m_0 = A'B'C' )</td>
<td>( M_0 = A + B + C )</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>( m_1 = A'B'C )</td>
<td>( M_1 = A + B' + C )</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>( m_2 = A'B C' )</td>
<td>( M_2 = A + B + C' )</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>( m_3 = A'B C )</td>
<td>( M_3 = A + B + C )</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>( m_4 = AB'C' )</td>
<td>( M_4 = A' + B' + C )</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>( m_5 = AB'C )</td>
<td>( M_5 = A' + B + C )</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>( m_6 = AB C' )</td>
<td>( M_6 = A' + B + C )</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>( m_7 = AB C )</td>
<td>( M_7 = A' + B' + C )</td>
</tr>
</tbody>
</table>

Each minterm corresponds to one input condition.
Complement using Minterms/Maxterms

- \((m_i)^* = M_i\)
- **Complement of f:**
  1. Count 0’s in \(f\) (find \(f’\) directly)
     \[ f' = m_0 + m_1 + m_2 = \Sigma m(0, 1, 2) \]
     \[ f' = M_3M_4M_5M_7 = \Pi M(3, 4, 5, 6, 7) \]
  2. Count 1’s in \(f\) (find \(f\) and then complement it)
     \[ f' = (f')' = (m_3 + m_4 + m_5 + m_6 + m_7)' \]
     \[ = M_0' + M_1' + M_2' \]
     \[ = m_0 + m_1 + m_2 = \Sigma m(0, 1, 2) \]

Example: Minterm/Maxterm Expansion

- e.g., \(f(a, b, c, d) = a'(b' + d) + acd'\)
  \[ f(a, b, c, d) = a'(b' + d) + acd' \]
  \[ = a'b' + a'd + acd' \]
  \[ = a'b'(c + c')(d + d') + a'd(b + b')(c + c') + acd'(b + b') \]
  \[ = a'b'cd + a'b'c'd + a'b'cd' + a'b'cd + ab'cd + abcd' + ab'cd' \]
  \[ = \Sigma m(0, 1, 2, 3, 5, 7, 10, 14) \text{ (Minterm Expansion)} \]
  \[ = \Pi M(4, 6, 8, 9, 11, 12, 13, 15) \text{ (Maxterm Expansion)} \]

Summary (1/2)

- Convert a Boolean expression to a minterm/maxterm expansion
  - Use truth table
    - Sometimes there are too many terms
  - Use Boolean algebra
    - SOP → multiply out and use \((X + X') = 1\) → minterm expansion
    - POS → factor and use \(XX' = 0\) → maxterm expansion

Summary (2/2)

- Convert between a minterm and a maxterm expansion
  - If \(f = \Sigma m_i\), then \(f = \Pi M_j\), where each \(m_i\) is not in \(f\)

<table>
<thead>
<tr>
<th>DESIRED FORM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Minterm</strong> Expansion of (f)</td>
</tr>
<tr>
<td><strong>Minterm</strong> Expansion of (f)</td>
</tr>
<tr>
<td><strong>Maxterm</strong> Expansion of (f)</td>
</tr>
</tbody>
</table>

- There is a 1-to-1 mapping between a truth table and the minterm/maxterm expansion
AND of Two Minterm Expansions

- e.g., Given \( f_1 = \sum m(0, 2, 3, 5, 9, 11) \), \( f_2 = \sum m(0, 3, 9, 11, 13, 14) \), find \( f_1 f_2 =? \)
  - AND: take the numbers that appear in both expansions:
    \( f_1 f_2 = \sum m(0, 3, 9, 11) \)
- Q: What if AND for two maxterm expansions?

General Truth Table

- Q: Given three Boolean variables \( A, B, C \), how many different Boolean functions can you produce?
- A:
  - Each \( a_i \) can be assigned with either 0 or 1
  - \( \Rightarrow 2^8 = 256 \)

Incompletely Specified Functions (1/2)

- A large digital system is usually divided into subcircuits
  - Assume \( N_1 \) never generates \( ABC = 001/110 \) for any \( w, x, y, z \)
  - \( F \): Incompletely specified function
  - \( A'B'C, ABC' \): don't care terms
  - "don't care" (DC) terms can be assigned with either "0" or "1"
Incompletely Specified Functions (2/2)

Impact of don’t care terms on Boolean simplification:
- Try exhaustive combinations of DCs to find the best (may be stupid but works for now)
  1. Assign “0” to both “X”:
     \[ F = A'B'C' + A'BC + ABC = A'B'C' + BC \]
  2. Assign “1” to 1st “X”, and “0” to 2nd “X”:
     \[ F = A'B'C' + A'B'C + ABC + ABC = A'B' + BC \]
  3. Assign “1” to both “X”:
     \[ F = A'B'C' + A'B'C + ABC + ABC = A'B' + BC + AB \]

2. is the simplest solution

Notation:
- \( F = \sum m(0, 3, 7) + \sum d(1, 6) \)
- \( F = \Pi M(2, 4, 5) \cdot \Pi D(1, 6) \)

Truth Table Construction

Minterm & maxterm expansions

Error Detector for 6-3-1-1 Codes (1/2)

Design an error detector for 6-3-1-1 codes:
- The output \( F = 1 \) iff inputs \((A, B, C, D)\) represent an invalid code combination

Minterm & maxterm expansions

Error Detector for 6-3-1-1 Codes (2/2)

Minterm & maxterm expansions
Example: Truth Table Construction

1. Construct the truth table

2. Simplify the function

3. Realize it

Minterm & maxterm expansions

Addition table:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>C_in</th>
<th>C_out</th>
<th>S</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>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</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>
<td>0</td>
</tr>
</tbody>
</table>

C = XY
S = X'Y + XY' = X ⊕ Y

Minterm & maxterm expansions

1-Bit Half Adder (HA)

1. Construct the truth table

2. Simplify the function

3. Realize it

Minterm & maxterm expansions

1-Bit Full Adder (FA)

1. Construct the truth table

2. Simplify the function

3. Realize it

Minterm & maxterm expansions
### Designing a 4-Bit Parallel Adder (1/3)

- **Definition:**
  \[ A = (A_3, A_2, A_1, A_0), B = (B_3, B_2, B_1, B_0) \]

- **Carry out**:
  \[ C_4 \Leftarrow S_3, S_2, S_1, S_0 \Leftarrow C_0 \text{ Carry in} \]

- **Example:**
  \[
  \begin{align*}
  1 & 0 & 1 & 1 & 0 \quad \text{Carries} \\
  1 & 0 & 1 & 1 & \\
  1 & 0 & 1 & 1 & \quad C_4 (=0)
  \end{align*}
  \]

- **How?**
  - Use a single truth table?
  - Too big!

### Designing a 4-Bit Parallel Adder (2/3)

- **Decompose the 4 bit adder into four modules**
  - Each module adds two bits and a carry ⇒ use full adder

- **Extend to negative numbers**
  - Consider one's complement (What if two's complement?)
    - 1. Add just as if all numbers were positive
    - 2. End-around carry: Add the carry out back to the rightmost bit

- **How to detect overflow?**
  - Check sign!
  - \((+) + (+) \Rightarrow (-)\) or \((-) + (-) \Rightarrow (+)\)

### Designing a 4-Bit Parallel Adder (3/3)

- **Overflow detection?**
  \[
  \begin{array}{c|cccc|c}
  A & B & S & A_3 & B_3 & \text{Overflow} \\
  + & + & - & 0 & 0 & 1 & 1 \\
  + & - & + & 1 & 1 & 0 & 1 \\
  \end{array}
  \]

  \[ V = A_3B_3'S_3 + A_2B_2S_2' \]

### Designing a Binary Subtractor (1/2)

- **Consider**
  \[ A - B = A + (-B) = A + B^* = A + B_2 + 1 \]

- **Convert**
  - \(B\) to 2's complement: inverse and then add 1
  - Discard the carry from the sign bit

### Designing a Binary Subtractor (2/2)

- **Overflow**
  \[
  C_{4} = 1
  \]

- **Minterm & maxterm expansions**
  \[
  \begin{align*}
  \text{Add}(A, B) & < 2^n-1 \\
  \text{Add}(A, -B) & > A \\
  \text{Add}(-A, B) & > A \\
  \text{Add}(-A, -B) & < 2^n-1
  \end{align*}
  \]
Or design a full subtracter

\[ X - Y = D \text{ (difference), } B: \text{ borrow} \]

DIY!

Homework for Unit 4

- Problems
  - 4.18
  - 4.22
  - 4.25
- Homework #2 covers Units 4–5