Gate-Level Minimization

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Gate-Level Minimization – The Map Method

- Truth table is unique
- Many different algebraic expression
- Boolean expressions may be simplified by algebraic means
- But, awkward due to the lack of specific rules
- Karnaugh Map or K-map method
  - Pictorial form of truth table
  - A simple and straightforward procedure
Why Need to be Simple?

- Produces a circuit diagram with a minimum number of gates and the minimum number of inputs to the gate
- Simplest expression is not unique

Two-Variable Map

Fig. 3-1 Two-variable Map

$$m_1 + m_2 + m_3 = x'y' + xy' + xy = x + y$$
Three-Variable Map

Not in a binary sequence, but in a sequence similar to Gray code

\[ m_5 + m_7 = xy'z' + xyz = xz(y + y') = xz \]
\[ m_0 + m_2 = x'y'z' + x'yz' = x'z' + y = x'z' \]

**Examples**

Ex 3-1) Simplify the Boolean function, \( F(x, y, z) = \Sigma(2, 3, 4, 5) \)

\[ F = x'y + xy' \]

Ex 3-4) Simplify the Boolean Function, \( F(x, y, z) = \Sigma(0, 2, 4, 5, 6) \)

\[ F = z' + xy' \]
Example

Ex 3–4) Given Boolean function, \( F = A'C + A'B + AB'C + BC \)

a) express it in sum of minterms
\[
F(x, y, z) = \Sigma(1, 2, 3, 5, 7)
\]

b) find the minimal sum of products
\[
F = C + A'B
\]

![Map for Example 3-4](image)

Fig. 3-7 Map for Example 3-4; \( A'C + A'B + AB'C + BC = C + A'B \)

Four-Variable Map

Ex 3–5) Simplify the Boolean function,
\[
F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)
\]

\[
F = y' + w'z' + xz'
\]

![Four-variable Map](image)

Fig. 3-8 Four-variable Map

![Map for Example 3-5](image)

Fig. 3-9 Map for Example 3-5; \( F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz' \)
Examples

1. Simplify the Boolean function
   \[ F(x, y, z) = \Sigma(3, 4, 6, 7) \]
   \[ yz + xz' \]

2. Simplify the Boolean function
   \[ F(x, y, z) = \Sigma(0, 2, 4, 5, 6) \]
   \[ z' + xy' \]

Prime Implicants

A **Prime Implicant** is a product term obtained by combining the maximum possible number of adjacent squares in the map.

\[ F(A,B,C,D) = \Sigma(0,2,3,5,7,8,9,10,11,13,15) \]

\[ F = BD + B'D' + CD + AD \]
\[ = BD + B'D' + CD + AB' \]
\[ = BD + B'D' + B'C + AD \]
\[ = BD + B'D' + B'C + AB' \]

![Prime Implicants Diagram](image-url)
Ex 3–7) Simplify the Boolean function,
\[ F(A,B,C,D,E) = \Sigma(0,2,4,6,9,13,21,23,25,29,31) \]
\[ F = A'B'E' + BD'E + ACE \]
Examples

1. Simplify the following Boolean functions by first finding the essential prime implicants:

   \[ F(A,B,C,D) = \Sigma(0,2,3,5,7,8,10,11,14,15) \]

   i) find the essential prime implicants \( CD + B'D' \)
   
   ii) find the non essential prime implicants \( AC + A'BD \)
   
   iii) simplify function \( F = CD + B'D' + AC + A'BD \)

2. Simplify the following Boolean functions, using five-variable maps:

   \[ F(A,B,C,D,E) = \Sigma(0,1,4,5,16,17,21,25,29) \]

   Ans) \( A'B'D' + B'C'D' + AD'E \)

Product of Sums Simplification

Ex 3–8) Simplify the Boolean function,

\[ F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10) \]

   a) sum of products
   
   \[ F = B'D' + B'C' + A'C'D \]
   
   b) product of sum
   
   \[ F' = AB + CD + BD' \]

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

Fig. 3-14 Map for Example 3-8: \( F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10) \)

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

(b) \[ F = (A' + B')(C' + D')(B' + D) \]

Fig. 3-15 Gate Implementation of the Function of Example 3-8
Product of Sums Simplification

\[ F(x, y, z) = \Sigma(1, 3, 4, 6) = \Pi(0, 2, 5, 7) \]
\[ F = x'z + xz' \]
\[ F' = xz + x'z' \]
\[ F = (x'+z')(x + z) \]

**Examples**

Simplify the following Boolean functions in product of sums:

1. \( F(w, x, y, z) = \Sigma(0, 2, 5, 6, 7, 8, 10) \)
   \[ \text{Ans) } (w'+x')(x+z')(x'+y+z) \]

2. \( F(A, B, C, D) = \Pi(1,3,5,7,13,15) \)
   \[ \text{Ans} ) (B'+D')(A+D') \]
Ex 3–9) Simplify the Boolean function, \( F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15) \)
Don’t-care conditions, \( d(w, x, y, z) = \Sigma(0, 2, 5) \)

\[
F(w, x, y, z) = yz + w'x' = \Sigma(0, 1, 2, 3, 7, 11, 15) \\
F(w, x, y, z) = yz + w'z = \Sigma(1, 3, 5, 7, 11, 15)
\]

NAND and NOR Implementation

- Digital circuits are frequently constructed with NAND or NOR gates rather than AND and OR gates
- NAND and NOR gates are easier to fabricate with electronic components
- Basic gates used in all IC digital logic families
NAND Circuits

- NAND Circuit

NAND gate is a universal gate

Ex 3-10) Implement the following Boolean function with NAND gates:

\[ F(x, y, z) = \Sigma(1, 2, 3, 4, 5, 7) = xy' + x'y + z \]

Two-Level Implementation

\[ F = ((AB)'(CD)')' = AB + CD \]

Ex 3-10) Implement the following Boolean function with NAND gates:

\[ F(x, y, z) = \Sigma(1, 2, 3, 4, 5, 7) = xy' + x'y + z \]
Multilevel NAND Circuits

- Convert all AND to NAND with NAND–inverter
- Convert all OR to NAND with inverter–NAND
- Check all the inverter in the diagram. For every inverter that is not compensated by another circle along the same line, insert an inverter (one-input NAND gate) or complement the input literal

Example—Multilevel NAND Circuits

Fig. 3-22 Implementing $F = A(CD + B) + BC'$
Example—Multilevel NAND Circuits

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

(a) AND-OR gates

(b) NAND gates

NOR Implementation

(a) OR–invert

(b) Invert–AND

Fig. 3-25 Two Graphic Symbols for NOR Gate
NOR Operation is the Dual of the NAND

- OR gates to NOR gates with NOR–invert
- AND gates to NOR gates with invert–NOR
- Any Inverter that is not compensated by another inverter along the same line needs an inverter or the complementation of the input literal

Example

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

Fig. 3-27 Implementing \( F = (AB' + A'B)(C + D') \) with NOR Gates
Exclusive-OR Function

\[ \text{XOR} : x \oplus y = xy' + x'y \]
\[ \text{XNOR} : (x \oplus y)' = xy + x'y' \]

\[ x \oplus 0 = x \]
\[ x \oplus 1 = x' \]
\[ x \oplus x = 0 \]
\[ x \oplus x' = 1 \]
\[ x \oplus y' = x' \oplus y = (x \oplus y)' \]

Fig. 3-32 Exclusive-OR Implementations

Parity Generation and Checking

Table 3-4: Even-Parity-Generator Truth Table

<table>
<thead>
<tr>
<th>Three-Bit Message</th>
<th>Parity Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) 3-bit even parity generator

Table 3-5: Even-Parity-Checker Truth Table

<table>
<thead>
<tr>
<th>Four Bits Received</th>
<th>Parity Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( y )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(a) 4-bit even parity checker

Fig. 3-36 Logic Diagram of a Parity Generator and Checker
HDL (Hardware Description Language)

// HDL Example 3-1
// Description of the simple circuit of Fig. 3-37
module smpl_circuit(A, B, C, x, y);
    input A, B, C;
    output x, y;
    wire e;
    and g1(e, A, B);
    not g2(y, C);
    or g3(x, e, y);
endmodule