Chapter 4

Simplification of Boolean Expressions

Factors to be considered in evaluating the merit of a network include:
- Cost
- Reliability
- Propagation delay: the times it takes the network to respond to changes at its inputs

Cost measure

- The cost of a network can be measured in many different ways.
- In this course, unless otherwise specified, the cost of a network is defined to be the total number of gates plus the total number of gate inputs.
- Availability of inputs in complemented and uncomplemented form is assumed.

The cost of a Boolean expression

- is defined to be the cost of the corresponding network.

Relations among Boolean expressions

- Boolean expression $f_1$ implies Boolean expression $f_2$ if any assignment of values to the variables involved makes $f_1 = 1$, it also makes $f_2 = 1$. For example, $f_1 = x'z + y'z$ implies $f_2 = x'y + y'z$.
- A term $t_1$ subsumes term $t_2$ if any literal that occurs in $t_2$ also occurs in $t_1$. For example, $wx'yz$ subsumes $x'z$, and $x + y$ subsumes $x$.

Implicants

- A product term is said to be an implicant of a Boolean function if it implies the function.
- For example, if a function is expressed in sum of products, then any product term therein is an implicant. If $f(x, y) = x'y + xy'$ then $x'y$ is an implicant. So is $xy'$.
Prime implicants

• An implicant of a function is said to be a prime implicant if the implicant does not subsume any other implicant with fewer literals of that function.
• For example, consider \( f(x, y, z) = x'y + z \). Both \( x'y \) and \( xz \) are implicants of \( f \). But while \( x'y \) is a prime implicant of \( f \), \( xz \) is not because it subsumes \( z \).

Irredundant disjunctive normal formula

An irredundant disjunctive normal formula (IDNF) is a Boolean expression in sum-of-product form such that (1) every product term involved is a prime implicant, and (2) no product term may be eliminated without changing the definition of that function. For example, \( x'y + z \) is an IDNF but not \( x'y + z + x'z \) or \( x'y \).

The minimization problem

• The minimization problem to be discussed in the following is to find, for a given Boolean expression, an equivalent one that has the minimum cost, and that satisfies any other constraints imposed.

Simplification methods

• A graphic method that can handle Boolean expressions up to 6 variables - Karnaugh maps
• A tabular method that has no limit on the number of variable and can be implemented on a computer - Quine-McCluskey method

Graphic method

• It is a simplification method that makes use of the following relations:
  \[ x + x' = 1 \]
  \[ y \cdot 1 = 1 \cdot y = y \]
• It facilitates recognition of applicability of these relations by describing a Boolean function in a graphic form (Karnaugh map).

Basic idea

• Two product terms of a Boolean function can be combined and simplified if they have a distance of 1.
• The distance between two product terms is defined as the number of literals that occur differently (i.e., one is complemented while the other is not) in these terms.
Karnaugh maps

- The Karnaugh map of a function consists of a number of cells (squares) that is equal to number of its minterms.
- Each cell is associated with a minterm in such a way that, if two cells are immediately adjacent to each other, their corresponding minterms have a distance of 1.

Karnaugh maps (continued)

- The entry to a cell is equal to the value of the associated minterm.
- Possibilities of simplification is signified by the presence of 1’s occupying adjacent cells.
- A group of $2^n$ cells can be combined to form a simpler term.

Karnaugh maps and implicants

A Karnaugh map of a Boolean function allows us to visualize its implicants, prime implicants, and different ways to describe it as a irredundant disjunctive normal formula.

A different view

Conceptually, it may be useful to think of a Karnaugh map as a different form of the truth table. Each row of the truth table is embedded in a cell in the map in such a way that the possibility of simplification becomes obvious.

Karnaugh map of function of one variable.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$f(0)$</td>
</tr>
<tr>
<td>1</td>
<td>$f(1)$</td>
</tr>
</tbody>
</table>

(a)

Map of a function of 2 variables

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$f(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$f(0,0)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$f(0,1)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$f(1,0)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$f(1,1)$</td>
</tr>
</tbody>
</table>

(a)

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$f(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$f(0,0)$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$f(0,1)$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$f(1,0)$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$f(1,1)$</td>
</tr>
</tbody>
</table>

(b)
A variant of Karnaugh map

- There are many different ways to construct Karnaugh maps.
- The ones shown next make the simplification process less error prone.
Typical subcubes for elimination of two variables

Typical subcubes for elimination of three variables

Typical subcubes describing sum terms

Prime implicants on a map

An algorithm for finding all prime implicants

The map for \( f(x,y,z) = \Sigma m(0,1,5,7) \)
The map for \( f(w,x,y,z) = \Sigma m(1,2,3,5,6,7,8,13) \)

A quick way to construct the map of \( f = xy' + wxz + wx'yz \)

An example

An example
Map for the functions $f(w, x, y, z) = \sum m(1, 3, 4, 5, 6, 7, 11, 14, 15)$

Incompletely specified Boolean function

$$f(w, x, y, z) = \sum m(0, 3, 7, 8, 12) + dc(5, 10, 13, 14).$$

Five-variable Karnaugh maps.

(a) Reflective structure.

(b) Layer structure.

Typical subcubes on a five-variable map.

Figure 4.26

Figure 4.27
Maps for $f(x, y, z, t, v) = \Sigma m(0, 1, 2, 3, 6, 7, 11, 15, 16, 17, 19, 23, 27, 31)$

(a) Subcubes for the minimal sum. (b) Subcubes for the minimal product.

Six-variable Karnaugh maps. (a) Reflective structure. (b) Layer structure.

Typical subcubes on a six-variable map

General form of a combinational network

Minimization through sharing in a multiple output network

Minimization through sharing: optimization of individual output does not necessarily lead to overall optimization
Map compression of a three-variable function

Example of a variable-entered map.

Example of a variable-entered map with infrequently appearing variables.

Variable-entered maps grouping techniques

Minimization through a map with single-variable entries

Optimal groupings on a variable-entered map
Minimization through a map with single-variable entries

Obtaining a minimal sum for the incompletely specified Boolean function $f(w,x,y,z) = \Sigma m(3,5,6,7,8,9,10) + \Sigma d(4,11,12,14,15)$ using a variable-entered map. (a) Truth table. (b) Variable-entered map. (c) Step 1 map and subcubes. (d) Step 2 map and subcubes.

Figure 4.42

Obtaining a minimal sum for the incompletely specified Boolean function $f(w,x,y,z) = \Sigma m(0,4,5,6,13,14,15) + \Sigma d(2,7,8,9)$ using a variable-entered map. (a) Truth table. (b) Step 1 map and subcubes. (c) Step 2 map and subcubes.

Figure 4.43

Maps having entries involving more than one variable.
(a) Variable-entered map. (b) Grouping the $y$ literal. (c) Grouping the $z$ literal. (d) Grouping the not completely covered 1-cell.

Figure 4.45
Obtaining a minimal sum from a variable-entered map having several single-literal map entries. (a) Variable-entered map. (b) Optimal collection of subcubes.

Figure 4.46

Maps having sum terms as entries. (a) Variable-entered map. (b) Grouping the y literal. (c) Grouping the z literal. (d) Grouping the not completely covered 1-cell.

Figure 4.47

Maps having product terms as entries. (a) Variable-entered map. (b) Grouping the yz term. (c) Grouping the y literal. (d) Grouping the not completely covered 1-cell.

Figure 4.48

Maps having product and sum terms as entries.
(a) Variable-entered map. (b) Grouping the yz term. (c) Grouping the not completely covered 1-cell.

Figure 4.49