Systems I: Computer Organization and Architecture

Lecture 4 - Karnaugh Maps

Mapping Boolean Functions

- Boolean expressions can be fairly complex.
  - This leads to overly-complex digital circuits.
  - This necessitates simplification of Boolean expressions.
- These expressions can become too complex for simplification by Boolean algebra.
- The use of maps such as Karnaugh maps makes it much easier to simplify such expressions.
Two-variable maps

Each square represents a minterm.

Representing 2-Variable Functions

\[ F = xy \]

\[ F = x + y = x'y + xy' + xy \]
Three-variable maps

• With only two dimensions on the page, we need to graph more than one dimension together whenever we go beyond two-variable maps.
• We arrange the minterms in an order that resembles Gray codes, where only one bit varies between adjacent squares.
• This allows us to recognize simpler terms quickly.

3-Variable Karnaugh Maps

<table>
<thead>
<tr>
<th></th>
<th>m₀</th>
<th>m₁</th>
<th>m₃</th>
<th>m₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>m₄</td>
<td>m₅</td>
<td>m₇</td>
<td>m₆</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccc}
00 & 01 & 11 & 10 \\
\hline
x'y'z' & x'y'z & x'y z & x'y z' \\
xy'z' & xy'z & xyz & xyz' \\
\end{array}
\]
Simplifying 3-Variable Expressions

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

Simplifying 3-Variable Expressions (continued)

\[ F = x'y'z + xy'z' + xyz + xyz' \]
\[ = xz' + yz \]
Simplifying 3-Variable Expressions (continued)

F = A’C + A’B + AB’C + BC
= C + A’B

F = Σ (0, 2, 4, 5, 6)
= z’ + xy’
Four-Variable Maps

Simplifying 4-variable Maps

\[ F = \sum(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz' \]
F = Σ(1, 3, 4, 5, 9, 11, 12, 13, 14, 15) = wx + xy' + x'z

F = A'B'C' + B'CD + A'BCD + AB'C' = B'D + B'C' + A'CD
Don’t Care Conditions

- Until now, all the spaces on a Karnaugh map indicates where the function has a value of “1” or “0” (assumed by the square being left empty).
- Sometimes we don’t care what value the square holds – it is irrelevant.
- We can’t leave it blank (assumed to be 0), 0 or 1, so we mark it with an “X” to indicate that we *don’t care* about its value.

Don’t-Care Conditions – An Example

\[ F = \Sigma(1,3,7,11,15) \]
\[ d = \Sigma(0, 2, 5) \]
\[ F = w'z + yz = z (w' + y) \]
Don’t-Care Conditions – An Example

\[ F = \Sigma(1,2,3,5,7) \]
\[ d = \Sigma(10,11,12, 13 14, 15) \]

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

5-Variable Maps

<table>
<thead>
<tr>
<th></th>
<th>CDE</th>
<th>AB</th>
<th>000</th>
<th>001</th>
<th>011</th>
<th>010</th>
<th>110</th>
<th>111</th>
<th>101</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>00</td>
<td>01</td>
<td>01</td>
<td>01</td>
<td>11</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>8</td>
<td>9</td>
<td>11</td>
<td>10</td>
<td>14</td>
<td>15</td>
<td>13</td>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>24</td>
<td>25</td>
<td>27</td>
<td>26</td>
<td>30</td>
<td>31</td>
<td>29</td>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>16</td>
<td>17</td>
<td>19</td>
<td>18</td>
<td>22</td>
<td>23</td>
<td>21</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Product of Maxterms

- Normally, we express Boolean function as a sum of minterms, e.g.,
  - $xy + x'z$
  - $A + B'C$
- Each of the $2^n$ functions of n binary variables can be rewritten as a product of maxterms, e.g.,
  $$xy + x'z = (x' + y)(x + z)(y + z)$$
Using Karnaugh Maps to Find Product of Maxterms

\[ F' = x'y' + x'z' \]
\[ F = (x' + y)(x + z) \]

Using Karnaugh Maps and Don’t-Care Conditions to Find Product of Maxterms

\[ F' = z' + w'y' \]
\[ F = z(w' + y) \]