Lecture 7

◆ Logistics
  - HW2 due now
  - Lab2 going on
  - 10/10 This Friday lecture: Prof. Paul Beame

◆ Last lecture
  - Logic simplification
    📝 Boolean cubes
    📝 Karnaugh maps

◆ Today
  - Continuing on K-maps (lots of examples)
  - Don't cares

The “WHY” slide

◆ Karnaugh map (K-map)
  - A visualization tool for logic simplification. It is a flatten version of Boolean cube that allows you to solve even difficult and complex Boolean expression into the minimized form. A convenient tool for you to know for up to 6 input variable expressions (e.g. $X = f(A, B, C, D, E, F)$)

◆ Don't cares
  - Sometimes the logic output doesn't matter. When we don't care if the output is 0 or 1, rather than assigning random outputs, it is best to denote it as “Don't care.” If you learn how to use the “don't care's”, you will be able to build even more efficient circuits than without them.
K-map minimization example: 3 variables

Find the least number of subcubes, each as large as possible, that cover the ON-set. Make sure subcubes contain 1, 2, 4, or 8 items (remember the Boolean cube).

\[ \text{Cout} = AB + BCin + ACin \]

A | B | Cin | Cout
---|---|---|---
0 | 0 | 0 | 0
0 | 0 | 1 | 0
0 | 1 | 0 | 0
0 | 1 | 1 | 1
1 | 0 | 0 | 0
1 | 0 | 1 | 1
1 | 1 | 0 | 1
1 | 1 | 1 | 1

One more example: 3 variables

Find the least number of subcubes, each as large as possible, that cover the ON-set. Make sure subcubes contain 1, 2, 4, or 8 items (remember the Boolean cube).

\[ \text{Cout} = A + BCin \]

A | B | Cin | Cout
---|---|---|---
0 | 0 | 0 | 0
0 | 0 | 1 | 0
0 | 1 | 0 | 0
0 | 1 | 1 | 1
1 | 0 | 0 | 1
1 | 0 | 1 | 1
1 | 1 | 0 | 1
1 | 1 | 1 | 1
K-map minimization example: minterms

\[ F(A, B, C) = \Sigma m(0, 4, 5, 7) = B'C' + AC \]

K-map minimization example: complement

\[ F(A, B, C) = \Sigma m(0, 4, 5, 7) = B'C' + AC \]

\[ F'(A, B, C) = \Sigma m(1, 2, 3, 6) = A'C + BC' \]
K-map minimization example: 4 variables

- Minimize $\text{F}(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$
- Answer: $\text{F} = C + A'B'D + B'D'$

Find the least number of subcubes, each as large as possible, that cover the ON-set

K-map minimization examples: on whiteboard

$\text{F}(A,B,C) = \Sigma m(0,3,6,7)$
$\text{F}(A,B,C) =$
$\text{F}'(A,B,C) =$

$\text{F}(A,B,C,D) = \Sigma m(0,3,7,8,11,15)$
$\text{F}(A,B,C,D) =$
$\text{F}'(A,B,C,D) =$
How about Karnaugh Maps, 6 dimensions

K-maps become 3D for 5 & 6 variables

\[
\begin{array}{c|ccc}
& CD & EF & \hline \\
AB = 00 & 00 & 01 & 11 & 10 & 00 & 01 & 11 & 10 \\
00 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
01 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
11 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
& CD & EF & \hline \\
AB = 01 & 00 & 01 & 11 & 10 & 00 & 01 & 11 & 10 \\
00 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
01 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
11 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|ccc}
& CD & EF & \hline \\
AB = 11 & 00 & 01 & 11 & 10 & 00 & 01 & 11 & 10 \\
00 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
01 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
11 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\text{OUTPUT} = A'BC'D'F' + CF + BC'D'E
\]

Incompletely specified functions: Don’t cares

- Functions of n inputs have \(2^n\) possible configurations
  - Some combinations may be unused
  - Call unused combinations “don’t cares”
  - Exploit don’t cares during logic minimization
  - Don’t care ≠ no output

- Example: A BCD increment-by-1
  - Function F computes the next number in a BCD sequence
    - If the input is 0010\(_2\), the output is 0011\(_2\)
    - BCD encodes decimal digits 0–9 as 0000\(_2\)–1001\(_2\)
      - Don’t care about binary numbers 1010\(_2\)–1111\(_2\)
### Truth table for a BCD increment-by-1

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>OUTPUTS</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td></td>
<td></td>
</tr>
<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>1</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>0</td>
<td>0</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>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>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>(X)</td>
<td>(X)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>(X)</td>
<td>(X)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>(X)</td>
<td>(X)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>(X)</td>
<td>(X)</td>
</tr>
</tbody>
</table>

- Don’t care set for W: \(X\) \(X\) \(X\) 

---

### Notation

- **Don't cares in canonical forms**
  - Three distinct logical sets: \{on\}, \{off\}, \{don't care\}

- **Canonical representations of a BCD increment-by-1**
  - Minterm expansion
    - \(W = m_7 + m_8 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15}\)
    - \(= \Sigma m(7,8) + d(10,11,12,13,14,15)\)
  - Maxterm expansion
    - \(W = M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 \cdot M_6 \cdot M_9 \cdot D_{10} \cdot D_{11} \cdot D_{12} \cdot D_{13} \cdot D_{14} \cdot D_{15}\)
    - \(= \Pi M(0,1,2,3,4,5,6,9) \cdot D(10,11,12,13,14,15)\)

- **In K-maps, can treat ‘don't cares’ as 0s or 1s**
  - Depending on which is more advantageous
Example: with don’t cares

\[ F(A,B,C,D) = \Sigma m(1,3,5,7,9) + d(6,12,13) \]

- \( F = A'D + B'C'D \) \textit{without using don't cares}
- \( F = A'D + C'D \) \textit{using don't cares}

Assign \( X = 1 \) ⇒ allows a 2-cube rather than a 1-cube