Lecture 6

◆ Logistics
  ■ HW2 out, due 4/16 Wednesday
  ■ Lab 2 ongoing

◆ Last lecture
  ■ Canonical forms
  ■ NAND and NOR

◆ Today’s lecture
  ■ One more pushing bubble example
  ■ Logic simplification
    ☑ Boolean cubes
    ☑ Karnaugh maps

Example of bubble pushing: NAND/NAND

\[ F = A'B'C + A'B'C + AB'C + ABC + A'B + ABC \]

\[ F = (A + B + C)(A + B + C)(A' + B + C) \]
Example of bubble pushing: NOR/NOR

\[ F = A'B'C + A'BC + AB'C + ABC + A'B + ABC \]

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

Goal: Minimize two-level logic expression

- Algebraic simplification
  - not an systematic procedure
  - hard to know when we reached the minimum
- Computer-aided design tools
  - require very long computation times (NP hard)
  - heuristic methods employed – "educated guesses"
- Visualization methods are useful
  - our brain is good at figuring things out over computers
  - many real-world problems are solvable by hand
Key tool: The Uniting Theorem

- The uniting theorem \( A(B' + B) = A \)
- The approach:
  - Find some variables don’t change (the A’s above) and others do (the B’s above)
  - Eliminate the changing variables (the B’s)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- A has the same value in both “on-set” rows \( \Rightarrow \) keep A
- B has a different value in the two rows \( \Rightarrow \) eliminate B

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

Boolean cubes

- Visualization tool for the uniting theorem
- n input variables = n-dimensional “cube”
Mapping truth tables onto Boolean cubes

- **ON set = solid nodes**
- **OFF set = empty nodes**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Look for on-set adjacent to each other

Sub-cube (a line) comprises two nodes

A varies within the sub-cube; B does not

This sub-cube represents B'

Example using Boolean cube

- **Binary full-adder carry-out logic**

  - On-set is covered by the OR of three 2-D subcubes

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ (A'+A)BCin \]

\[ A(B+B')Cin \]

\[ AB(Cin'+Cin) \]

\[ Cout = BCin + AB + ACin \]
M-dimensional cubes in n-dimensional space

- In a 3-cube (three variables):
  - A 0-cube (a single node) yields a term in 3 literals
  - A 1-cube (a line of two nodes) yields a term in 2 literals
  - A 2-cube (a plane of four nodes) yields a term in 1 literal
  - A 3-cube (a cube of eight nodes) yields a constant term "1"

F(A,B,C) = \sum m(4,5,6,7)
On-set forms a square (a 2-D cube)
A is asserted (true) and unchanging
B and C vary
This sub-cube represents the literal A

Karnaugh maps (K-map)

- Flat representation of Boolean cubes
  - Easy to use for 2- 4 dimensions
  - Hard for 4 - 6 dimensions
  - Virtually impossible for 6+ dimensions
  - Use CAD tools
- Help visualize adjacencies
  - On-set elements that have one variable changing are adjacent

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
2, 3, and 4 dimensional K-maps

- Uses Gray code: Only one bit changes between cells
  - Example: 00 → 01 → 11 → 10

Adjacencies

- Wrap-around at edges
  - First column to last column
  - Top row to bottom row
K-map minimization example: 2 variables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

F = B'

K-map minimization example: 3 variables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<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>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Cout = AB + BCin + ACin
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 $F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$
  - Find the least number of subcubes, each as large as possible, that cover the ON-set

Answer: $F = C + A'B'D + B'D'$
K-map minimization examples: do it yourself

\[ F(A,B,C) = \Sigma m(0,3,6,7) \]
\[ F'(A,B,C) = \]

\[ F(A,B,C,D) = \Sigma m(0,3,7,8,11,15) \]
\[ F'(A,B,C,D) = \]