DeMorgan’s Law and Bubble Pushing

\[ \overline{A + B} = \overline{A} \cdot \overline{B} \quad \overline{A \cdot B} = \overline{A} + \overline{B} \]

Bubble Pushing Example

Representing Combinational Logic

- Truth Table
- Boolean Formula

Circuit

For combinational logic, these three:
- are equivalently
- straight-forward to
- have no
2-Level Logic

- Represent ______ logic function(s)
  - Utilizing just two types of gates

  (assuming we get NOT for free)
- Two forms
  - Sum of products
  - Product of sums

- Relationship with truth table
  - Generate a gate level implementation of any set of logic functions
  - Allows for simple reduction/minimization

Example

- Show the sum of products for the following truth table.
  - Strategy: _________ all the products where the output is ________

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

- z =

- Is this optimal?

Minimization by Hand

- Sum of Products:                                                Truth Table:
  \[ z = (\overline{A} \cdot \overline{B} \cdot C) + (A \cdot \overline{B} \cdot \overline{C}) + (A \cdot B \cdot C) + (A \cdot B \cdot C) \]

- Okay to duplicate terms while minimizing

Reduction/Minimization

- Reduction is important to reduce the size of the circuit that performs the function. This, in turn, reduces the cost of, and delay through, the circuit.

- What?
  - Less power consumption
  - Less heat
  - Less space
  - Less time to propagate a signal through the circuit
  - Less points of possible failure

- It makes good engineering and economic sense!
Karnaugh Maps (k-Maps)

- A graphical (pictorial) method used to minimize Boolean expressions.
- Don’t require the use of Boolean algebra theorems and equation manipulations.
- A special version of a truth table.
- Works with two to four input variables (gets more and more difficult with more variables)
- Groupings must be __________________
- Final result is in _____________________ form

Karnaugh Maps (k-Maps) Example #1

- Lets create a k-map table
  - Borders represent all possible conditions
    - NOT in counting order
    - Be consistent
  - -What are the values for the map?
    - The values of ___
  - To reduce, circle our powers of 2!

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

- Result:

K-Maps Example #2

Suppose we already have this k-Map. Minimize the function.

<table>
<thead>
<tr>
<th></th>
<th>CD</th>
<th>CD</th>
<th>CD</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{AB}$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\overline{AB}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$AB$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\overline{AB}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Every “1” must be ______________ by at least one term
- Larger blocks in k-Map produce smaller product terms

Truth Table and Logical Circuit Example

- How does a truth table and subsequent sum of products equation create a logic circuit?
  - From the earlier example:
    \[ z = \overline{A} \cdot C + \overline{A} \cdot \overline{B} + \overline{A} \\
  \]
  - Lets build the logical circuit:
    - Which gates do we need?
    - How many inputs do we have?
    - How do we connect the circuit?
**Example Circuit**

\[ z = \overline{A} \cdot \overline{C} + A \cdot \overline{B} + A \cdot C \]

**Don’t Cares**

- Sometimes don’t care about the output.
- Each X can be either a 0 or 1 (helps with minimization)
- But in actual circuit, each X will have some specific value

<table>
<thead>
<tr>
<th>( \overline{C} \overline{D} )</th>
<th>( \overline{C} \overline{D} )</th>
<th>CD</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>( AB )</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \overline{A} \overline{B} )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( AB )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( \overline{A} \overline{B} )</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**General Skills**

- Make sure you can populate a K-Map from a truth table
- Make sure you can populate a truth table from a K-Map
- Given a circuit, know how to construct a truth table
- Given a truth table, know how to produce a sum-of-products, and how to draw a circuit
- Be able to understand minimization and use it
- Know DeMorgan’s Law and other Boolean laws