Optimizations and Tradeoffs

Combinational Logic Optimization
Optimization & Tradeoffs

- Up to this point, we haven’t really considered how to optimize our designs.

- Optimization is the process of transforming our design to improve criteria of interest without negatively effecting other criteria.

- Common Optimization Criteria:
  - Performance
  - Size
  - Power Consumption

- Tradeoffs are transformations that improve one or more criteria of interest at the expense of another
  - i.e. lower power circuits must generally operate at lower frequencies.
Two-level Optimization

- A boolean function can be implemented in only two levels using SOP or POS forms.
  - Results in minimal delays

- **Two-level logic size optimization** is a process that minimizes the number of transistors in a two-level logic circuit
  - This is the same as minimizing the numbers of literals and terms in a POS or Sop equation.
  - Two methods to accomplish this:
    1. Boolean Algebra Manipulation of the SOP
    2. Karnaugh Maps
Karnaugh Maps (K-maps)

- A visual method of algebraically minimizing SOP or POS functions.

Basic Approach:
- Plot the truth table outputs in a grid against the inputs
- $2^N$ cells in the grid, one for each possible input combination
  - The value of only one literal may change across rows or columns
  - This is a gray code
- Circle 1’s in groups of $2^N$
  - Allows the use of the formula $X + X' = 1$ to simplify and eliminate variables.
Two-variable K-maps

Example:
- \( Z = A'B + AB \)

1. Construct the grid.
   - \( N = 2 \rightarrow 2 \times 2 \) grid
   - Each axis is for one literal
     - A or B
   - Populate each axis with changing bit values
     - Only one bit can change at a time.
Two-variable K-maps

- Example:
  - \( Z = A'B + AB \)

2. Fill it in with values from the truth table.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>m0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>m1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>m2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>m3</td>
</tr>
</tbody>
</table>
Two-variable K-maps

- Example:
  - $Z = A'B + AB$

3. Circle 1’s in groups of $2^N$
   - N is a positive integer

4. Apply $X + X' = 1$ to the circled 1’s.
   - Note that two 1’s cover $A$ and $A'$, so we can discard it.

\[
\begin{array}{c|c|c}
\text{B} & 0 & 1 \\
\hline
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

$Z = B$
Three variable K-map

Now we will add a second literal on the upper axis.

Example: Optimize the given logic expression:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Q</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>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Three variable K-map

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

\[ Q = A'B + BC + AC \]
K-map Terminology

Before formalizing the process for optimally using K-maps to simplify output expressions, we need to define a few terms:

- **Implicant** – Product Term that covers more than one minterm
  - The previous example has 4 minterms
    - A'BC', AB'C, ABC', ABC
  - The K-map shows 3 implicants
    - A'B, BC and AC

- **Prime Implicant** – An implicant that is not completely covered by other implicants.
  - All 3 implicants above are prime implicants.

- **Distinguished 1-cell** – An input combination that is only covered by one prime implicant.
  - A'BC' and AB'C are distinguished 1 cells

- **Essential Prime Implicant** – A prime implicant that covers one or more distinguished 1-cells.
  - A'B and AC both contain distinguished 1-cells.
Optimal K-map Process

1. Identify and select all essential prime implicants.

2. Select the minimal number of remaining prime implicants to complete the cover of minterms.
Three variable K-map Revisited

\[
\begin{array}{ccc|c}
A & B & C & Q \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

\[
Q = A'B + BC + AC
\]

Applying these rules ...

\[
Q = A'B + AC
\]
SOP vs. POS

So far we have focused on the SOP form, but all of these procedures apply to the POS form as well.

**SOP - Minterms**
- Implicant
- Prime Implicant
- Essential Prime Implicant

**POS - Maxterms**
- Implicate
- Prime Implicate
- Essential Prime Implicate
POS Three variable K-map

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

Maxterms: $A+B+C$, $A+B+C'$, $A'+B+C$, $A'+B'+C$

Implicates: $B+C$, $A'+C$, $A+B$

Prime Implicates: $B+C$, $A'+C$, $A+B$

Essential Prime Implicates: $A+B$, $A'+C$

$Q = (A+B) \cdot (A'+C)$
Four Variable K-maps

- Now we will add a second literal on the side axis.

- Example: Optimize the given logic expression:
  - $F(A,B,C,D) = \text{sum}(m0, m4, m5, m7, m8, m10, m14, m15)$
Four Variable K-maps

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

Prime Implicants: \( A'C'D, B'C'D, BCD, ACD' \) (row)
\( A'BC', A'BD, ABC, AB'D' \) (col)

No Essential Prime Implicants!
Start with one prime and completely cover …
Four Variable K-maps

\[ F(A,B,C,D) = A'C'D' + A'BD + ABC + AB'D' \]

\[ F(A,B,C,D) = B'C'D' + A'BC' + BCD + ACD' \]

Both of these are equally optimal solutions!
Another 4-Variable Example

Determine the following:

- Prime Implicants:

- Essential Primes:

- Optimal Equation(s):
Don’t Cares

- Occasionally, there are certain input combinations where you don’t care what the output values is (0 or 1)
- This can be used to your advantage by choosing the most useful value!
Don’t Care Example

\[ F(W,X,Y,Z) = \sum(m_3, m_4, m_9, m_{15}) + d(m_0, m_1, m_5, m_{10}, m_{12}, m_{14}) \]

Note that if don’t cares are ignored, all 4 minterms are essential primes. How can this be improved by using the don’t cares?

\[ F(W,X,Y,Z) = W'X'Z + X'Y'Z + WXY + W'Y' \]
Automated Optimization

- K-maps have limited application because it can only handle a limited number of inputs
  - Anything larger than 5 inputs is cumbersome.
- Computer programs are generally used for larger numbers of inputs. We will consider two methods:
  - Quine-McClusky
    - A tabular method that is very adaptable to computer programming.
    - Achieves an optimal solution, like K-maps.
    - Computational time and memory resources can limit its usage.
  - Heuristic Method
    - Doesn’t create an optimal solution, but close.
    - Used when minimization problems are too large for other computational methods
Multi-level Optimization

- **Performance vs. Size**
  - Two level optimization leads to a design that is optimized in terms of delay, but that is not the only type of optimization possible.
  - By increasing the depth of gates, we create larger delays, but we may reduce the number of transistors (size).

- **Example:**

  - \( F_1 = ab + acd + ace \)
    - 22 transistors
    - 2 gate delays
    - (a)

  - \( F_2 = a(b+c(d+e)) \)
    - 16 transistors
    - 4 gate-delays
    - (b)