Chapter #2: Two-Level Combinational Logic

Section 2.3, 2.4 -- Switches and Tools
Gate Logic: Two-Level Simplification

**Key Tool: The Uniting Theorem**  \[ A (B' + B) = A \]

\[ F = A \overline{B}' + AB = A (B' + B) = A \]

- B's values change within the on-set rows
- \( B \) is eliminated, \( A \) remains
- A's values don't change within the on-set rows

\[ G = A' \overline{B}' + A \overline{B}' = (A' + A) \overline{B}' = \overline{B}' \]

- B's values stay the same within the on-set rows
- \( A \) is eliminated, \( B \) remains
- A's values change within the on-set rows

**Essence of Simplification:**
find two element subsets of the ON-set where only one variable changes its value. This single varying variable can be eliminated!
Gate Logic: Two-Level Simplification

Karnaugh Map Method

K-map is an alternative method of representing the truth table that helps visualize adjacencies in up to 6 dimensions.

Beyond that, computer-based methods are needed.

Numbering Scheme: 00, 01, 11, 10
Gray Code — only a single bit changes from code word to next code word.
Gate Logic: Two-Level Simplification

Karnaugh Map Method

Adjacencies in the K-Map

Wrap from first to last column

Top row to bottom row
**Gate Logic: Two-Level Simplification**

*K-Map Method Examples -- Minimum sum of products*

- **F = A**
  - A asserted, unchanged
  - B varies
  - \( F = A \)

- **G = B'**
  - B complemented, unchanged
  - A varies
  - \( G = B' \)

\[\begin{align*}
\text{Cout} &= \text{Bcin} + AB + ACin \\
\text{F}(A,B,C) &= A
\end{align*}\]
Gate Logic: Two-Level Simplification

More K-Map Method Examples, 3 Variables

F(A,B,C) = \Sigma m(0,4,5,7)

F' simply replace 1's with 0's and vice versa

F'(A,B,C) = \Sigma m(1,2,3,6)
Gate Logic: Two-Level Simplification

**K-map Method Examples: 4 variables**

\[
F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)
\]

\[
F = C + A'BD + B'D'
\]
Gate Logic: Two-Level Simplification

K-map Method: Circling Zeros -- Minimum Product of Sums

Replace $F$ by $ar{F}$, 0’s become 1’s and vice versa

$$F = (B + C + D) \ (A + C + D) \ (B + C + D)$$

$$\bar{F} = B \ C \ D + A \ C \ D + B \ C \ D$$

$$\bar{F} = (B + C + D) \ (A + C + D) \ (B + C + D)$$
Gate Logic: Two-Level Simplification

K-map Example: Don't Cares

Don't Cares can be treated as 1's or 0's if it is advantageous to do so.

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

F = A'D + B'CD w/o don't cares

F = A'D + C'D w/ don't cares
Gate Logic: Two Level Simplification

Definition of Terms

- **implicant**: single element of the ON-set or any group of elements that can be combined together in a K-map

- **prime implicant**: implicant that cannot be combined with another implicant to eliminate a term

- **essential prime implicant**: if an element of the ON-set is covered by a single prime implicant, it is an essential prime

Objective:

- grow implicants into prime implicants
- cover the ON-set with as few prime implicants as possible
- essential primes participate in ALL possible covers
Gate Logic: Two Level Simplication

Examples to Illustrate Terms

6 Prime Implicants:

- A' B' D
- B C'
- A C
- A' C' D
- A B
- B' C D

Essential minimum cover = B C' + A C + A' B' D

5 Prime Implicants:

- B D
- A B C'
- A C D
- A' B C
- A' C' D

Essential implicants form minimum cover

Minimum cover = ABC' + ACD + A'BC + A’C’D
Gate Logic: Two Level Simplification

More Examples

Prime Implicants:
- B D
- C D
- A C
- B' C

Essential primes form the minimum cover

Minimum cover = B D + AC + B'C
Gate Logic: Two-Level Simplification

Algorithm: Minimum Sum of Products Expression from a K-Map

Step 1: Choose an element of ON-set not already covered by an implicant

Step 2: Find "maximal" groupings of 1's and X's adjacent to that element. Remember to consider top/bottom row, left/right column, and corner adjacencies. This forms prime implicants (always a power of 2 number of elements).

Repeat Steps 1 and 2 to find all prime implicants

Step 3: Revisit the 1's elements in the K-map. If covered by single prime implicant, it is essential, and participates in final cover. The 1's it covers do not need to be revisited

Step 4: If there remain 1's not covered by essential prime implicants, then select the smallest number of prime implicants that cover the remaining 1's
Gate Logic: Two Level Simplification

Example: \( f(A,B,C,D) = \sum m(4,5,6,8,9,10,13) + \sum d(0,7,15) \)

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Initial K-map

© R.H. Katz  Transparency No. 4-14
**Gate Logic: Two Level Simplification**

**Example:** \( f(A, B, C, D) = \overline{m}(4, 5, 6, 8, 9, 10, 13) + d(0, 7, 15) \)

- **Initial K-map**
- **Primes around**  
  - \( A' B C' D' --m4 \)

\[ A' B, A' C'D \]**
Gate Logic: Two Level Simplification

Example: \( f(A,B,C,D) = \cdot m(4,5,6,8,9,10,13) + d(0,7,15) \)

Initial K-map

Primes around \( A' B C' D' \)

Primes around \( A B C' D' \) -- \( m5 \)

BD
Gate Logic: Two-Level Simplification

Example Continued

m6 -- no new implicant can be added because the set of implicants already contains A’B.
Gate Logic: Two-Level Simplification

Example Continued

Primes around
A B C' D

Primes around
A B' C' D' -- m8

m9 and m10 -- add no new prime implicants

AB'C', AB'D', B'C'D'
Gate Logic: Two-Level Simplification

Example Continued

Primes around A B C’ D

Primes around A B’ C’ D’

Essential Primes with Min Cover

A’B, AB’D’ cover m6 and m10 exclusively

Cover remaining 1’s not already covered by the essential primes. -- AC’D

F = A’B + AB’D’ + AC’D -- final minimized function
Gate Logic: Two-Level Simplification

Design Example: Two Bit Comparator

Block Diagram and Truth Table

A 4-Variable K-map for each of the 3 output functions
Gate Logic: Two-Level Simplification

Design Example: Two Bit Comparator

F1 =

F2 =

F3 =
Gate Logic: Two-Level Simplification

Design Example: Two Bit Adder

Block Diagram and Truth Table

A 4-variable K-map for each of the 3 output functions
Gate Logic: Two-Level Simplification

Design Example (Continued)

X =   

Z =   

Y =   

K-map for X

K-map for Y

K-map for Z
### Design Example: BCD Increment By 1

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<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>
<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>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</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>0</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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</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>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</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>
<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>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### Gate Logic: Two-Level Simplification

#### Design Example: BCD Increment By 1

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td><strong>B</strong></td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td><strong>C</strong></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>D</strong></td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

\[ W = \] 
\[ X = \] 
\[ Y = \] 
\[ Z = \]
Gate Logic: Two-Level Simplification

5-Variable K-maps

\[ f(A,B,C,D,E) = \Sigma m(2,5,7,8,10,13,15,17,19,21,23,24,29,31) \]
Gate Logic: Two-Level Simplification

5-Variable K-maps

\[ f(A, B, C, D, E) = \Sigma m(2, 5, 7, 8, 10, 13, 15, 17, 19, 21, 23, 24, 29, 31) \]

\[ = C \overline{E} + A \overline{B} E + B \overline{C} \overline{D} E' + \overline{A} \overline{C} D E' \]
Gate Logic: CAD Tools for Simplification

Quine-McCluskey Method

Karnaugh map method breaks down beyond six variables

Quine-McCluskey method finds the minimized representation of any Boolean expression by systematically finding all prime implicants and then extracting a minimum set of primes covering the on-set
Gate Logic: CAD Tools for Simplification

ESPRESSO Method

Problem with Quine-McCluskey: the number of prime implicants grows rapidly with the number of inputs

- upper bound: $3^n/n$, where $n$ is the number of inputs

- finding a minimum cover is NP-complete, i.e., a computational expensive process not likely to yield to any efficient algorithm

Espresso: trades solution speed for minimality of answer

- don't generate all prime implicants (Quine-McCluskey Stage 1)

- judiciously select a subset of primes that still covers the ON-set

- operates in a fashion not unlike a human finding primes in a K-map
Two-Level Logic: Summary

*Primitive logic building blocks*
  - INVERTER, AND, OR, NAND, NOR, XOR, XNOR

*Canonical Forms*
  - Sum of Products, Products of Sums
  - Incompletely specified functions/don't cares

*Logic Minimization*
  - Goal: two-level logic realizations with fewest gates and fewest number of gate inputs
    - Obtained via Laws and Theorems of Boolean Algebra
    - or Boolean Cubes and the Uniting Theorem
    - or K-map Methods up to 6 variables
    - or Quine-McCluskey Algorithm
    - or Espresso CAD Tool
Homework Assignment

HW#4  Chapter 2: Sections 2.3 and 2.4