Chapter 4 (continued ….)
Combinational Logic Network design

K-MAP: Summary

- **Row and columns are labeled so that adjacent Rows** (columns) labels differ only by one variable
- **1-cells represent minterms** while **0-cell are maxterms**

- **Minimization of SOP (sum of product terms)**
  - Make groups of 1-cells
  - Size of a group is a power of 2 (2, 4, 8 ....)
  - When possible include each 1-cell in largest possible group
  - A group of $2^k$ 1-cell allows the elimination of k variables from the product terms
  - A variable that changes over the labels associated with a group is dropped
  - In the logic of a group only variables that remains constant over the group are kept (1 corresponds to the non completed form of the variable and à corresponds to the complemented form)

- **Don’t care in K-Map** (d-cells)
  - They correspond to input combinations for which the output is not defined (invalid)
  - A don’t care d is considered as 1 only if it helps create a larger group of 1-cell
K-MAP: Minimizing Sums of Products

- **Minimal Sum of a Logic function** one variable
  A sum of product expression of F such that
  - No sum S’ has fewer product terms
  - If a sum S’ has the same number of terms, then S’ can’t have fewer variables

- **Prime implicant**
  - An implicant F is a prime implicant if it is not contained in a larger block of minterms (1-cells)
    
    Note: Larger implicant contain fewer terms
  - Prime implicant theorem: A minimum sum of a function F is a sum of prime implicant

- **Distinguished 1-cell:**
  A minterm that is covered by only one prime implicant

- **Essential prime implicant**
  A prime implicant that covers one or more distinguished cells

- **Minimal cover of a logic function**
  - Select all essential prime implicants (they cover some or all 1-cells of the function)
  - Select a set of non essential prime implicants to cover remaining 1-cells that are not covered by the essential prime implicants
Combinational Logic Network design

K-MAP: Minimizing Sums of Products

**Example**

F = \( \Sigma_{W,X,Y,Z} \) (0,1,2,3,4,5,7,14,15)

<table>
<thead>
<tr>
<th>WX</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>YZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>*</td>
<td>*</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Prime implicants:
- W'X'
- W'Y'
- W'Z
- XYZ
- WXY

Essential prime implicants:
- W'X'
- W'Y'
- WXY

Note:
Implicants W'Y'Z', W'Y'Z are not prime
- are contained in larger block W'Y'

* : Distinguished cells
- W'XY'Z', W'X'YZ', WXYZ'
K-MAP: Minimizing Sums of Products

Example: minimum coverage of F

\[ F = \sum_{W,X,Y,Z} (0,1,2,3,4,5,7,14,15) \]

<table>
<thead>
<tr>
<th>WX</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>YZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Only one minterm is not covered by essential prime implicants: \( W'XYZ \)

To cover the remaining 1-cell, choose the largest containing block (prime implicant): \( W'Z \)

\[ F = W'Y' + W'X' + WXY + W'Z \]
Timing diagram - Hazards

- **Timing Diagram**
  Plot a logic expression as a function time (Discrete function ….)

- **Circuit Delay**
  - Time it takes a gate or circuit to react to input and produce the corresponding output
  - Circuit delay is non zero

\[ \Delta : \text{gate delay} \]
Timing diagram - Hazards

- Circuit delay

Δ: gate delay
Combinational Logic Network design

Timing diagram - Hazards

- **Transient behavior of a circuit:**
  Intermediate non desired output may be produced before the circuit reaches a steady state
  - Glitch
  - Short pulse in the output
  - Hazards

- **Types of hazards**
  - Static hazard
    - Static-1 hazard
    - Static-0 hazard
  - Dynamic hazard
Combinational Logic Network design

Timing diagram - Hazards

**Static hazard**

- **Static-1 hazard**
  We expect the output to stay at 1, but output dips temporary to 0 and goes back to 1

  ![Static-1 hazard diagram](image)

- **Static-0 hazard**
  We expect the output to stay at 0, but output goes temporary to 1 and back to 0

  ![Static-0 hazard diagram](image)
Combinational Logic Network design

Timing diagram - Hazards

Static hazard

Example: Given $F = AB' + BC$

\[\begin{align*}
A: & 1 \\
B: & 1 \rightarrow 0 \\
C: & 1
\end{align*}\]

Initial values: $ABC = 111, \quad F = 1$

Input change: $B \ 1 \rightarrow 0, \quad F = ?$

• Steady state analysis (Zero circuit delay): When $B \ 1 \rightarrow 0, \quad AB' = 1, BC = 0$ and $F = 1$

• Transient behavior (Non zero circuit delay):
  • Two different paths arrive at the OR gate
  • Unequal propagation delays associated with the paths
  • One path has $B$ (non complemented) and the other has $B'$ (complemented form)
  Output of the top AND gate takes longer than the output of the bottom AND gate
Example: Given $F = AB' + BC$

Initial values: $ABC = 111$, $F = 1$

Input change: $B \ 1 \rightarrow 0$, $F =$?

0 Glitch
Timing diagram - Hazards

Static hazard

Example: Given \( F = AB' + BC \)

- This transaction is hazardous, no prime implicant covers both input \( ABC=111 \) and \( ABC = 101 \).
- Solution: add the consensus term to cover the above input.
- (See example on page 247 text, figure 4-48)
Combinational Logic Network design

Timing diagram - Hazards

Static hazard: Another example

\[ F = XY'Z' + W'Z + WY \]

To get rid of hazards, add extra product terms

Possible hazards
Combinational Logic Network design

Timing diagram - Hazards

Static hazard: Another example

To get rid of hazards, add extra product terms
Combinational Logic Network design

Timing diagram - Hazards

Dynamic hazard

Output changes 3 or more times, instead of changing once:

Case 1

Transient behavior

\[
\begin{align*}
1 & \rightarrow 0 \rightarrow 1 \rightarrow 0 \\
0 & \rightarrow 1 \rightarrow 0 \rightarrow 1
\end{align*}
\]

Expected output

\[
\begin{align*}
1 & \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow 0 \\
0 & \rightarrow 1 \rightarrow 0
\end{align*}
\]

Case 2

Transient behavior

\[
\begin{align*}
1 & \rightarrow 0 \rightarrow 1 \\
0 & \rightarrow 1 \rightarrow 0 \rightarrow 1
\end{align*}
\]

Expected output

\[
\begin{align*}
1 & \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow 0 \\
0 & \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow 1
\end{align*}
\]
Combinational Logic Network design

Timing diagram - Hazards

Dynamic hazard

Example: Given \( F = A(A' + B') + ABC \)

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
W & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
X & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
Y & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
Z & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[ \Delta < \Delta' \]

\( \Delta \)

\( \Delta' \)

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
F & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\Delta < \Delta'
\]

- Multiple paths from the changing input to the changing output

\( WXYZ = 0001 \quad ----> \quad WXYZ = 0101 \)
The END