Fig. 3-1 Two-variable Map

(a) $m_0 \quad m_1$

$\begin{array}{cc}m_0 & m_1 \\ m_2 & m_3\end{array}$

(b) $\begin{array}{cc}y & y \\ x & 0 \\ & 1 \\ 0 & x'y' & x'y \\ x & 1 & xy' & xy\end{array}$

Fig. 3-2 Representation of Functions in the Map

(a) $\begin{array}{cc}y & y \\ x & 0 \\ & 1 \\ 0 & 1 & 1 & 1 \\ x & 1 & 1 & 1\end{array}$

(b) $\begin{array}{cc}y & y \\ x & 0 \\ & 1 \\ 0 & 1 & 1 & 1 \\ x & 1 & 1 & 1\end{array}$

Friday, February 17, 12
Fig. 3-3 Three-variable Map
Fig. 3-4 Map for Example 3-1: \( F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy' \)
Fig. 3-5 Map for Example 3-2; $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$
Fig. 3-6 Map for Example 3-3; $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$
Fig. 3-7 Map for Example 3-4: $A'C + A'B + AB'C + BC = C + A'B$
Example: \( f(w,x,y,z) = \sum (0,1,2,4,5,6,8,9,12,13,14) \)
Fig. 3-9 Map for Example 3-5: $F(w, x, y, z)$

$$= \Sigma (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$$
Fig. 3-10 Map for Example 3-6: $A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'.
PRIME IMPLICANTS

• In choosing adjacent squares in a map, we must ensure that
  • all the minterm of the function are covered when we combine the squares
  • the number of terms in the expression is minimized
  • there are no redundant terms (i.e. minterms covered by other terms)

• Prime implicant (PI): product term obtained by combining the maximum possible number of adjacent squares.

• If a minterm in a square is covered by only one PI then the PI is essential.

• To avoid redundant terms, do (1) essential prime implicants, (2) prime implicants, (3) other terms
Map in (b): do the 1’s in (a) first, then CD and AB’
Example: \( F(A,B,C,D,E) = A'B'E' + BD'E + ACE \)
**FIGURE 3.13**
Map for Example 3.7, \( F = A'B'E' + BD'E + ACE \)
Fig. 3-13  Map for Example 3-7; \( F = A'B'E' + BD'E + ACE \)
\[ F(A,B,C,D) = \sum (0,1,2,5,8,9,10) \]

- Example 3-8: Simplify to a minimal expression using the:
  - 1’s to produce a sum of products (AND-OR)
  - 0's to produce a complemented sum of products (AND-NOR)
  - 0’s to produce a product of sums (OR-AND)
  - I’ to produce a complemented product of sums (OR-NAND)
Fig. 3-14 Map for Example 3-8: \( F(A, B, C, D) = \Sigma (0, 1, 2, 5, 8, 9, 10) \)
\[ = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D) \]
Fig. 3-15 Gate Implementation of the Function of Example 3-8

(a) \( F = B' D' + B' C' + A' C' D \)

(b) \( F = (A' + B')(C' + D')(B' + D) \)
Table 3.2
Truth Table of Function F

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>F</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>1</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>0</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>0</td>
</tr>
</tbody>
</table>

Fig. 3-16 Map for the Function of Table 3-2
(a) $F = yz + w'x'$

Fig. 3-17 Example with don’t-care Conditions

(a) $F = yz + w'z$
Fig. 3-18  Logic Operations with NAND Gates
Fig. 3-19 Two Graphic Symbols for NAND Gate
Fig. 3-20 Three Ways to Implement $F = AB + CD$
Fig. 3-21 Solution to Example 3-10
(a) AND-OR gates

(a) NAND gates

Fig. 3-22 Implementing $F = A(CD + B) + BC$
Fig. 3-23 Implementing $F = (AB' + A'B)(C + D')$
Fig. 3-24 Logic Operations with NOR Gates
Fig. 3-25 Two Graphic Symbols for NOR Gate
Fig. 3-26 Implementing \( F = (A + B)(C + D)E \)
Fig. 3-27 Implementing $F = (AB' + A'B)(C + D')$ with NOR Gates
Fig. 3-28 Wired Logic

(a) Wired-AND in open-collector TTL NAND gates.
(AND-OR-INVERT)

(b) Wired-OR in ECL gates
(OR-AND-INVERT)
Fig. 3-29  AND-OR-INVERT Circuits; $F = (AB + CD + E)'$
Fig. 3-30 OR-AND-INVERT Circuits; $F = [(A + B)(C + D)E]'$
(a) Map simplification in sum of products.

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

\[
F = x' y' z' + x y z' \\
F' = x' y + x y' + z
\]

(b) \( F = (x' y + x y' + z)' \) 

(c) \( F = [(x + y + z) (x' + y' + z)]' \)

Fig. 3-31 Other Two-level Implementations
(a) With AND-OR-NOT gates

(b) With NAND gates

Fig. 3-32 Exclusive-OR Implementations
(a) Odd function
\[ F = A \oplus B \oplus C \]

(b) Even function
\[ F = (A \oplus B \oplus C)' \]

Fig. 3-33 Map for a Three-variable Exclusive-OR Function
Fig. 3-34 Logic Diagram of Odd and Even Functions
(a) Odd function
\[ F = A \oplus B \oplus C \oplus D \]

(b) Even function
\[ F = (A \oplus B \oplus C \oplus D)' \]

Fig. 3-35 Map for a Four-variable Exclusive-OR Function
Fig. 3-36  Logic Diagram of a Parity Generator and Checker

(a) 3-bit even parity generator

(a) 4-bit even parity checker
Fig. 3-37  Circuit to Demonstrate HDL
Fig. 3-38  Simulation Output of HDL Example 3-3