Course Title:
Digital Logic Design

Shahriar Rahman Khan
Lecturer
Dept. of CSE, MIST

Course Code: CSE 103
Credit Hr: 3.00
Contact Hr: 3.00

Overview

- Map Method
  - Two and Three Variable Maps
  - Four Variable Map
  - Five Variable Map
- Product of Sums Simplification
- Don’t Care Conditions
- NAND Implementation
- NOR Implementation
- Other Two Level Implementations
- Determination of Prime Implicants
- Selection of Prime Implicants

Circuit Optimization

- Goal: To obtain the simplest implementation for a given function
- Optimization is a more formal approach to simplification that is performed using a specific procedure or algorithm
- Optimization requires a cost criterion to measure the simplicity of a circuit

Karnaugh Maps (K-map)

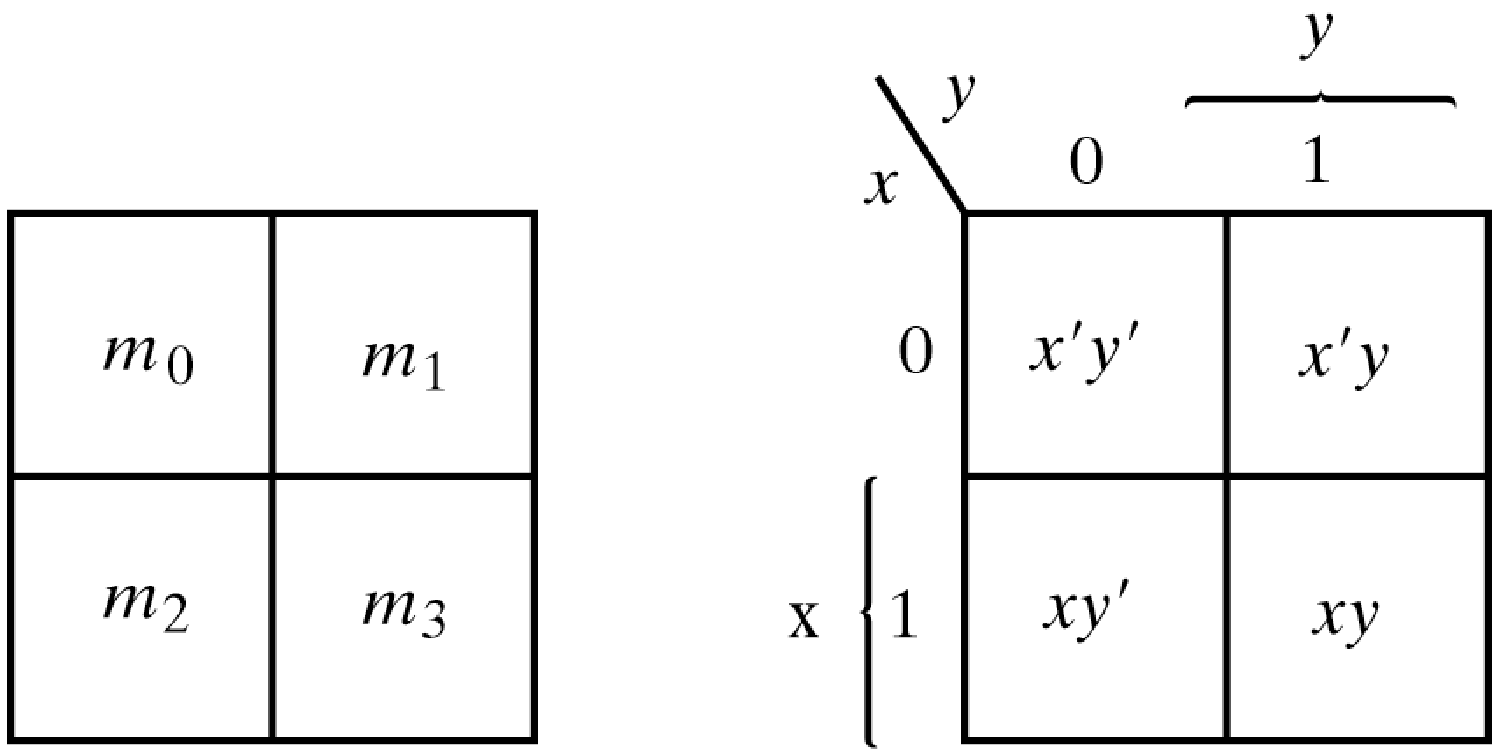
- **A K-map is a collection of squares**
  - Each square represents a minterm
  - The collection of squares is a graphical representation of a Boolean function
  - Adjacent squares differ in the value of one variable

- **The K-map can be viewed as**
  - A reorganized version of the truth table

Two Variable Maps

- A 2-variable Karnaugh Map:
  - Note that minterm m_0 and minterm m_1 are "adjacent" and differ in the value of the variable y
  - Similarly, minterm m_0 and minterm m_2 differ in the x variable.
  - Also, m_1 and m_3 differ in the x variable as well.
  - Finally, m_2 and m_3 differ in the value of the variable y

	y = 0	y = 1
x = 0	$m_0 = \overline{x}\,\overline{y}$	$m_1 = \overline{x}\,y$
x = 1	$m_2 = x\,\overline{y}$	$m_3 = x\,y$



(a) (b)

Fig. 3-1 Two-variable Map

K-Map Function Representation

- **Example: $F(x,y) = x$**

F = x	y = 0	y = 1
x = 0	0	0
x = 1	1	1

- **For function $F(x,y)$, the two adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$F(x, y) = x\,\overline{y} + x\,y = x$$

K-Map Function Representation

- **Example: G(x,y) = x + y**

G = x+y	y = 0	y = 1
x = 0	0	1
x = 1	1	1

- **For G(x,y), two pairs of adjacent cells containing 1's can be combined using the Minimization Theorem:**

$$G(x,y) = \left(x\,\overline{y} + x\,y\right) + \left(xy + \overline{x}\,y\right) = x + y$$

Duplicate xy

Three Variable Maps

- A three-variable K-map:

	yz=00	yz=01	yz=11	yz=10
x=0	m_0	m_1	m_3	m_2
x=1	m_4	m_5	m_7	m_6

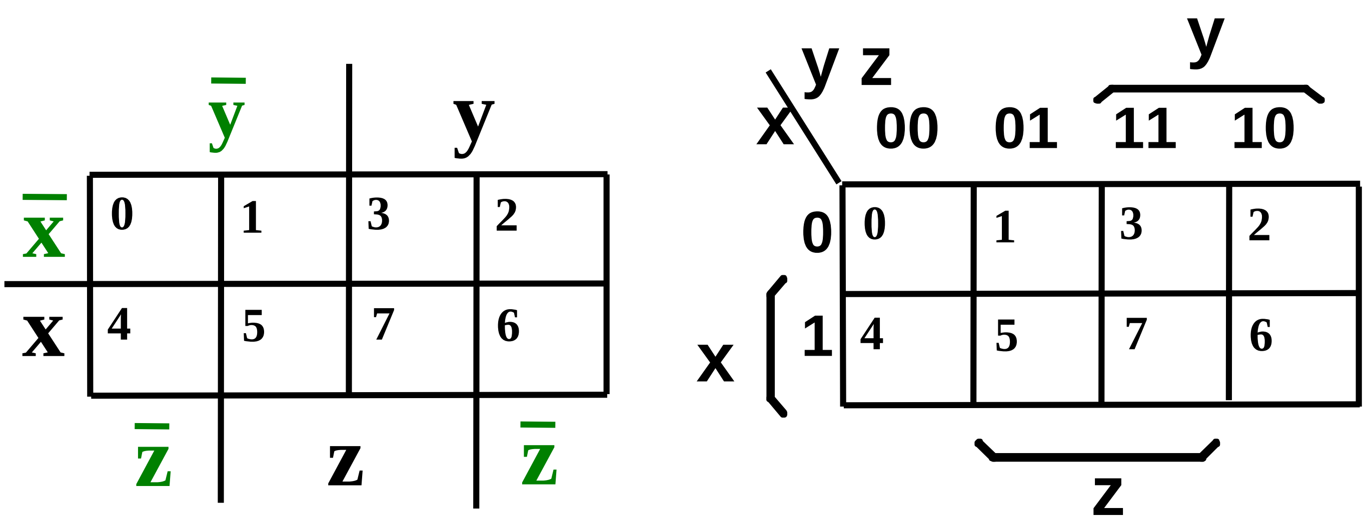
- Where each minterm corresponds to the product terms:

	yz=00	yz=01	yz=11	yz=10
x=0	$\overline{x}\,\overline{y}\,\overline{z}$	$\overline{x}\,\overline{y}\,z$	$\overline{x}\,y\,z$	$\overline{x}\,y\,\overline{z}$
x=1	$x\,\overline{y}\,\overline{z}$	$x\,\overline{y}\,z$	$x\,y\,z$	$x\,y\,\overline{z}$

- Note that if the binary value for an <u>index</u> differs in one bit position, the minterms are adjacent on the K-Map

Alternative Map Labeling

- **Map use largely involves:**
  - **Entering values into the map, and**
  - **Reading off product terms from the map.**
- **Alternate labelings are useful:**



Example

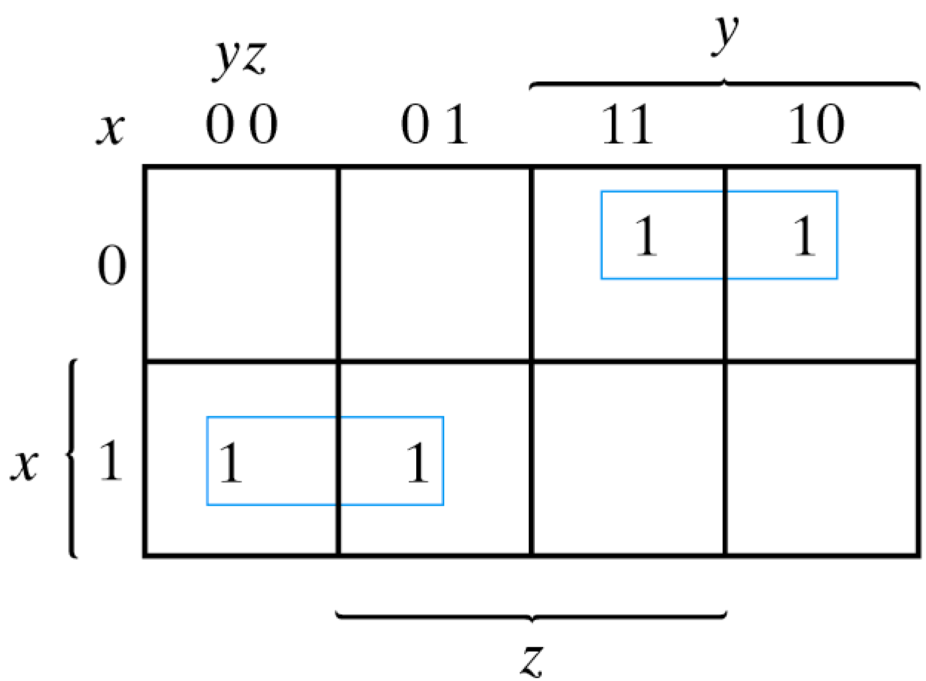


Fig. 3-4 Map for Example 3-1; $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

Example

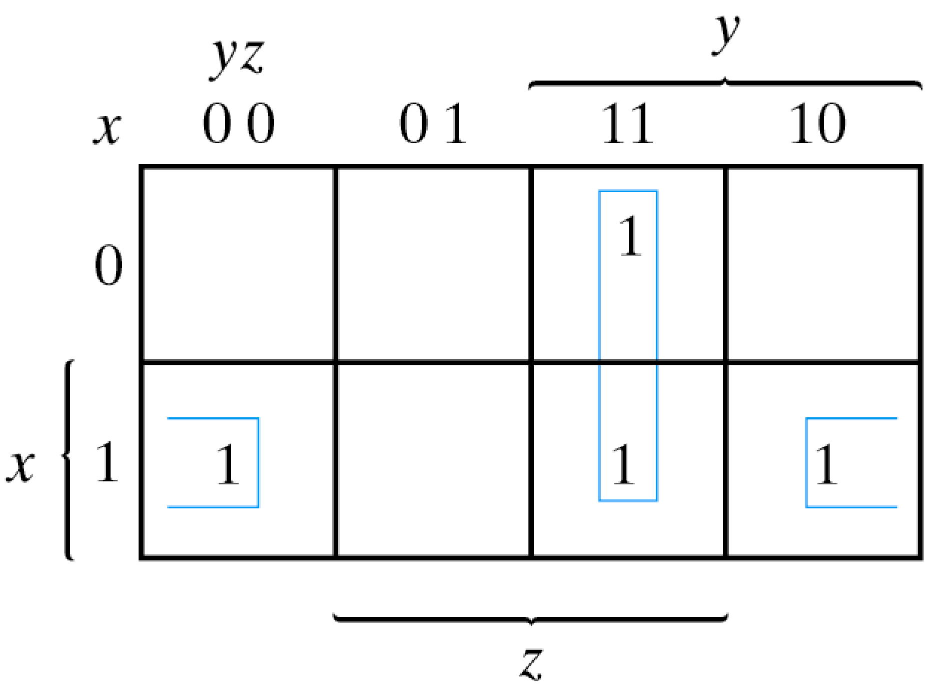


Fig. 3-5 Map for Example 3-2; $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

Example

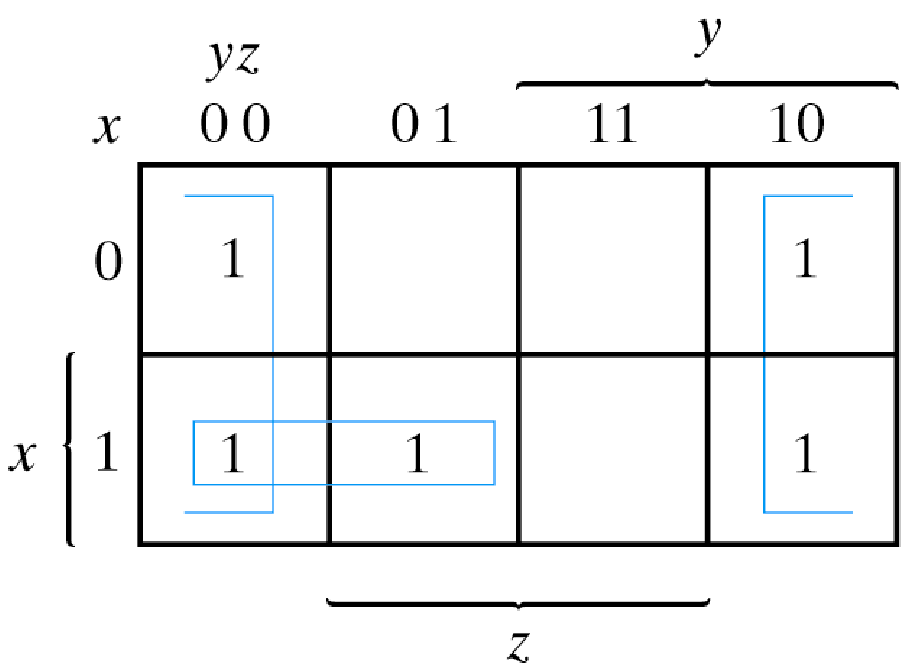


Fig. 3-6 Map for Example 3-3; $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

Example

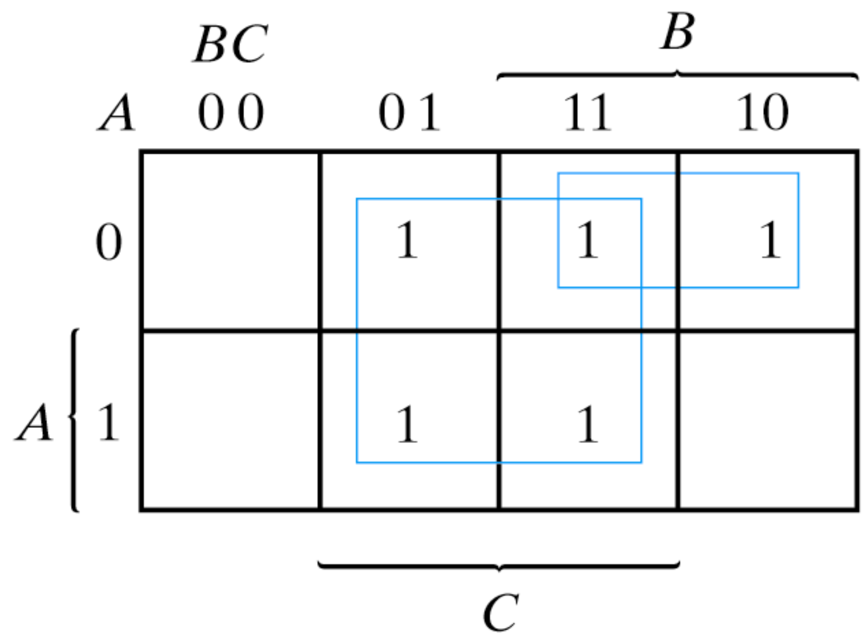


Fig. 3-7 Map for Example 3-4; $A'C + A'B + AB'C + BC = C + A'B$

Example Functions

- By convention, we represent the minterms of F by a "1" in the map and leave the minterms of $\overline{F}$ blank or fill the blank cells with 0.
- Example:

$$F(x, y, z) = \Sigma_m(2,3,4,5)$$

	z' y'	z y'	z y	z' y
x'	^{0}	^{1}	^{3}1	^{2}1
x	^{4}1	^{5}1	^{7}	^{6}

- Example:

$$G(a, b, c) = \Sigma_m(3,4,6,7)$$

	z' y'	z y'	z y	z' y
x'	^{0}	^{1}	^{3}1	^{2}
x	^{4}1	^{5}	^{7}1	^{6}1

- <u>Learn</u> the locations of the 8 indices based on the variable order shown (x, most significant and z, least significant) on the map boundaries

Combining Squares

- By combining squares, we reduce number of literals in a product term, reducing the literal cost, thereby reducing the other two cost criteria
- On a 3-variable K-Map:
  - One square represents a minterm with three variables
  - Two adjacent squares represent a product term with two variables
  - Four "adjacent" terms represent a product term with one variable
  - Eight "adjacent" terms is the function of all ones (no variables) = 1.

Example: Combining Squares

- Example: Let $F = \Sigma m(2,3,6,7)$

			y	y
	0	1	3 **1**	2 **1**
x	4	5	7 **1**	6 **1**
		z	z	

- Applying the Minimization Theorem three times:

$$
\begin{aligned}
F(x,y,z) &= \bar{x}\,y\,z + x\,y\,z + \bar{x}\,y\,\bar{z} + x\,y\,\bar{z} \\
&= yz + y\bar{z} \\
&= y
\end{aligned}
$$

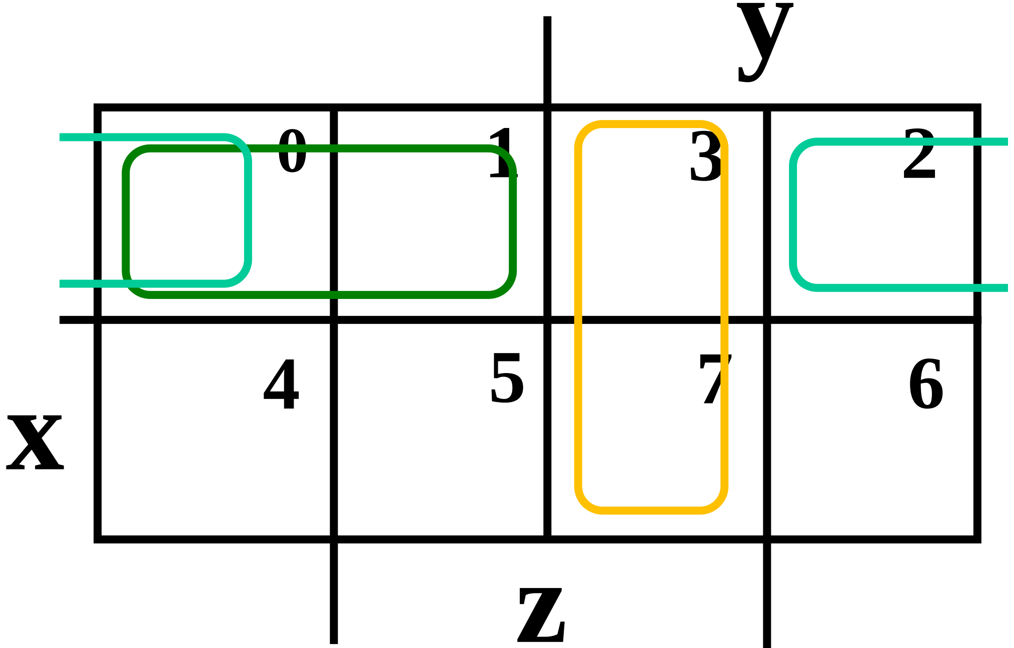
- Thus the four terms that form a 2 × 2 square correspond to the term "y".

Three-Variable Maps

- Reduced literal product terms for SOP standard forms correspond to <u>rectangles</u> on K-maps containing cell counts that are powers of 2.
- Rectangles of 2 cells represent 2 adjacent minterms; of 4 cells represent 4 minterms that form a "pairwise adjacent" ring.
- Rectangles can contain non-adjacent cells as illustrated by the "pairwise adjacent" ring above.

Three-Variable Maps

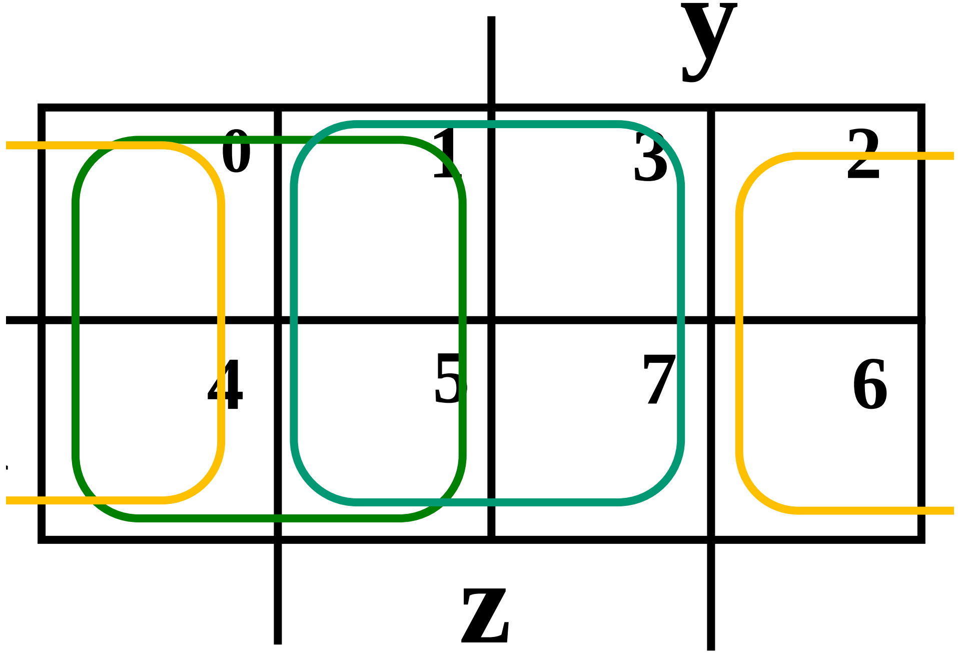
- **Example Shapes of 2-cell Rectangles:**



- **Read off the product terms for the rectangles shown**

Three-Variable Maps

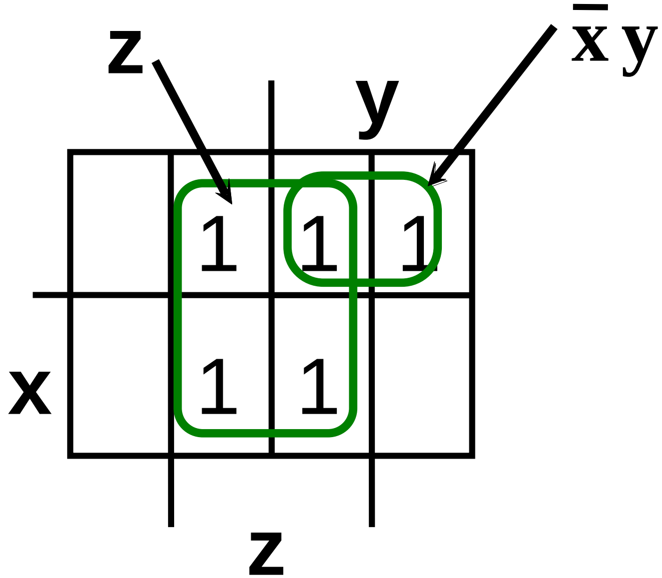
- Example Shapes of 4-cell Rectangles:



- Read off the product terms for the rectangles shown

Three Variable Maps

- K-Maps can be used to simplify Boolean functions by systematic methods. Terms are selected to cover the "1s"in the map.
- Example: Simplify $F(x, y, z) = \Sigma_m(1,2,3,5,7)$



$$F(x, y, z) = \quad z + \bar{x}\,y$$

Three-Variable Map Simplification

- **Use a K-map to find an optimum SOP equation for** $F(X,Y,Z) = \Sigma m(0,1,2,4,6,7)$

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

$wx \backslash yz$	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

(b)

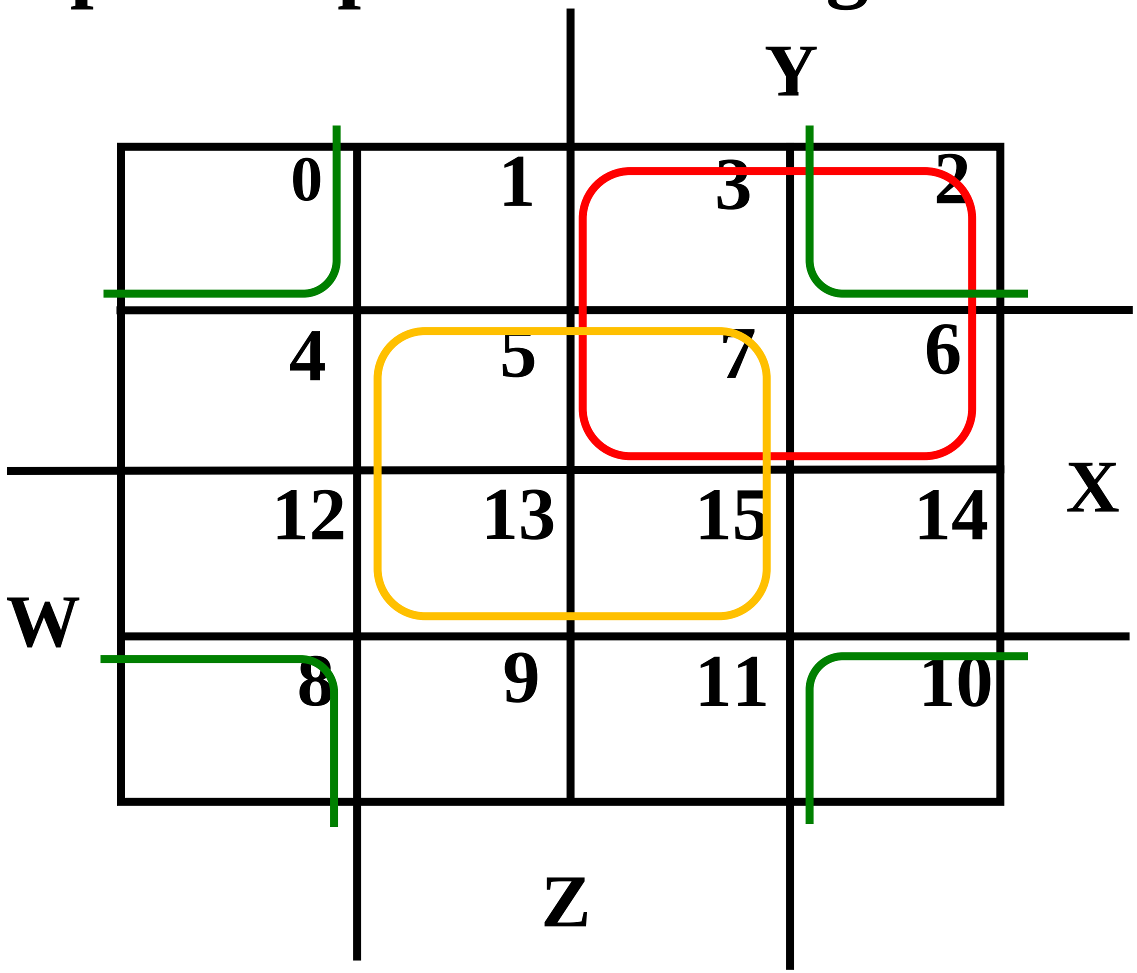
Fig. 3-8 Four-variable Map

Four Variable Terms

- **Four variable maps can have rectangles corresponding to:**
  - **A single 1 = 4 variables, (i.e. Minterm)**
  - **Two 1s = 3 variables,**
  - **Four 1s = 2 variables**
  - **Eight 1s = 1 variable,**
  - **Sixteen 1s = zero variables (i.e. Constant "1")**

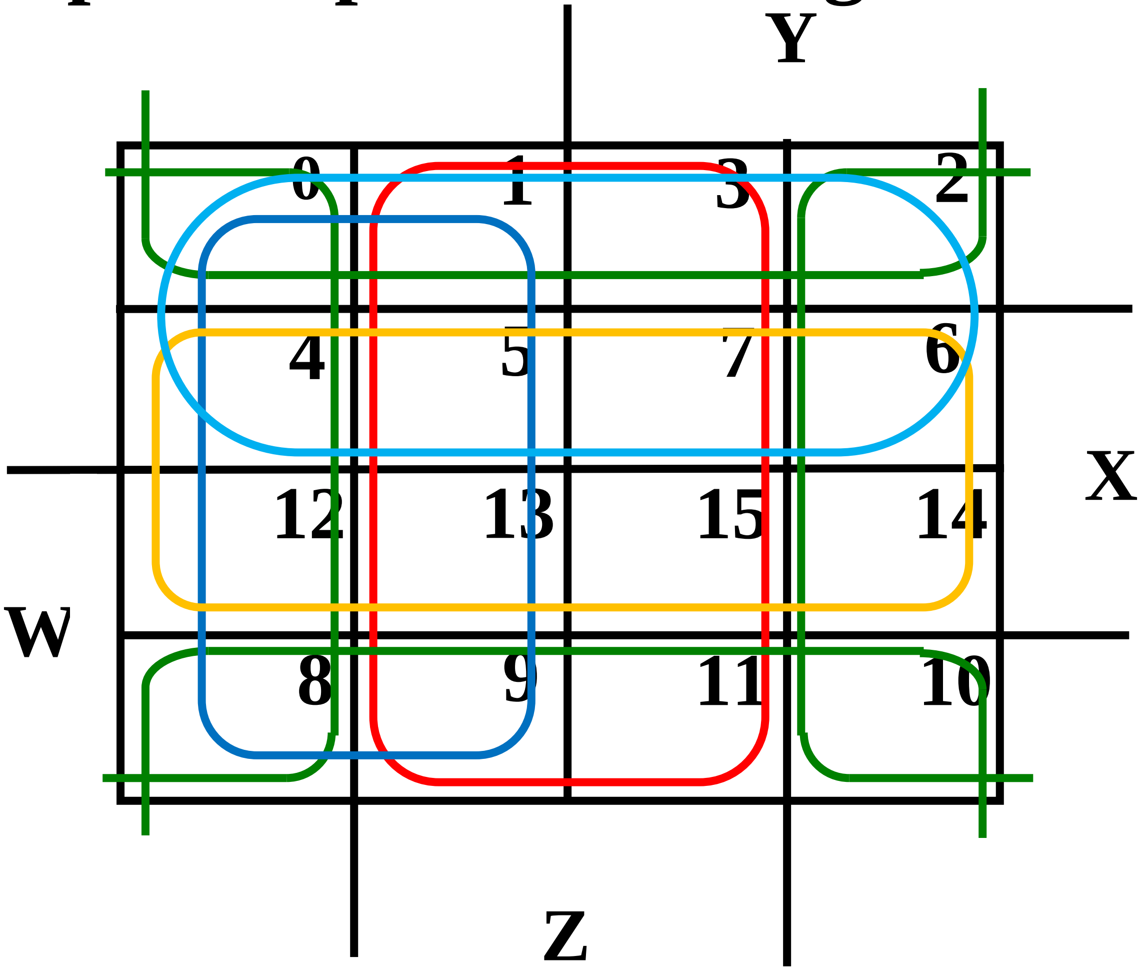
Four-Variable Maps

- **Example Shapes of Rectangles:**



Four-Variable Maps

- **Example Shapes of Rectangles:**



Four-Variable Map Simplification

- $F(W, X, Y, Z) = \Sigma_m(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

Four-Variable Map Simplification

- $F(W, X, Y, Z) = \Sigma_m(3,4,5,7,9,13,14,15)$

Example

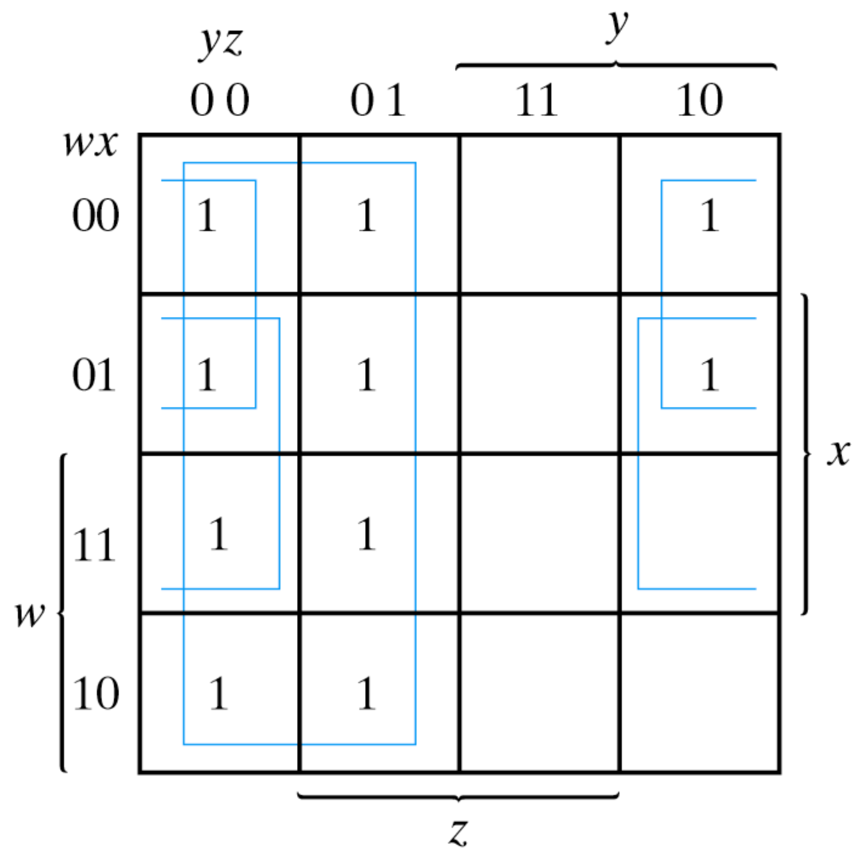


Fig. 3-9 Map for Example 3-5; $F(w, x, y, z)$

$= \Sigma\,(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

Example

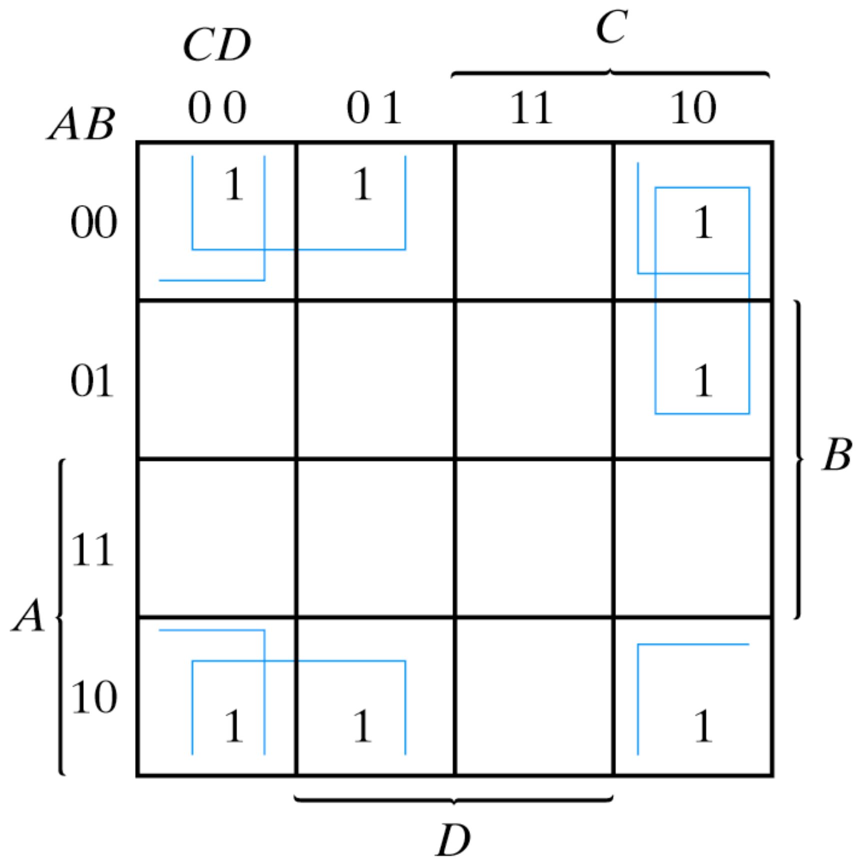
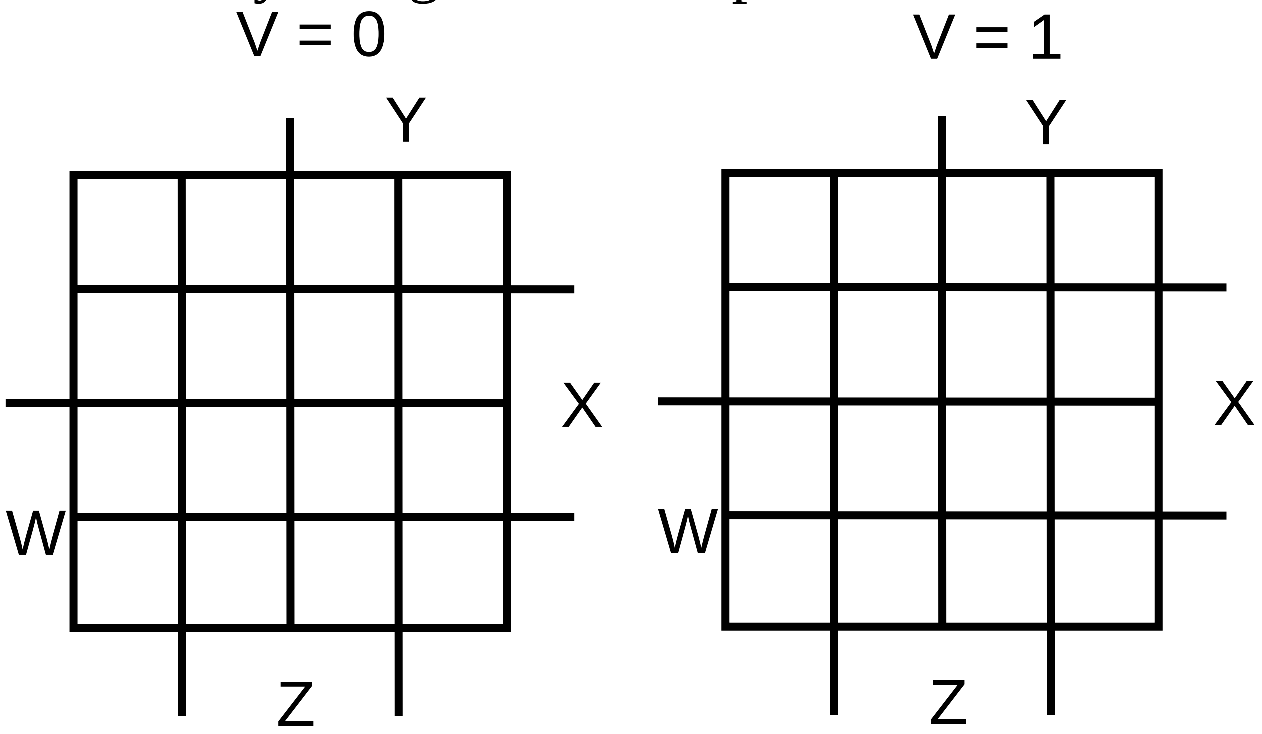


Fig.3-10 Map for Example 3-6; $A'B'C + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

Five Variable or More K-Maps

- For five variable problems, we use *two adjacent K-maps*. It becomes harder to visualize adjacent minterms for selecting PIs. You can extend the problem to six variables by using four K-Maps.



Five Variable or More K-Maps

AB \ CDE	000	001	011	010	110	111	101	100
00								
01								
11								
10								

5-variable Karnaugh map (Gray code)

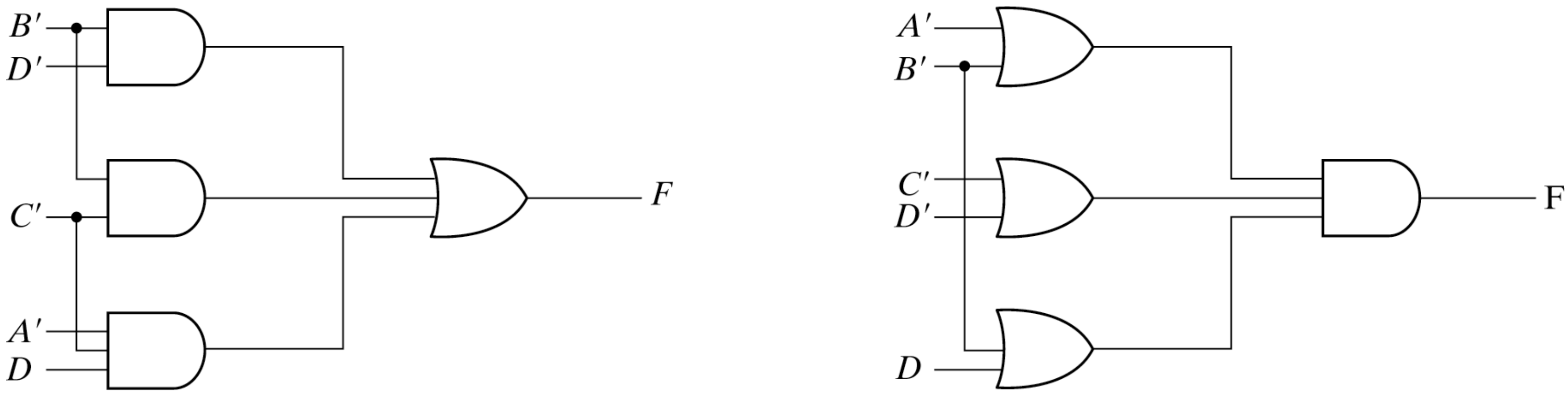
POS Simplification using K-Map

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

Fig. 3-14 Map for Example 3-8; $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

$= B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$

POS Simplification using K-Map



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

(b) $F = (A' + B')(C' + D')(B' + D)$

Fig. 3-15 Gate Implementation of the Function of Example 3-8

Don't Cares in K-Maps

- Up to this point we have considered logic reduction problems where the input conditions were completely specified. That is, a 3-variable truth table or Karnaugh map had 2n = 23 or 8-entries, a full table or map. It is not always necessary to fill in the complete truth table for some real-world problems. We may have a choice to not fill in the complete table.
- Sometimes a function table or map contains entries for which it is known:
  - the input values for the minterm will never occur, or
  - The output value for the minterm is not used
- In these cases, the output value need not be defined
- Instead, the output value is defined as a “don't care”
- By placing “don't cares” (an “x” entry) in the function table or map, the cost of the logic circuit may be lowered.

Don't Cares in K-Maps

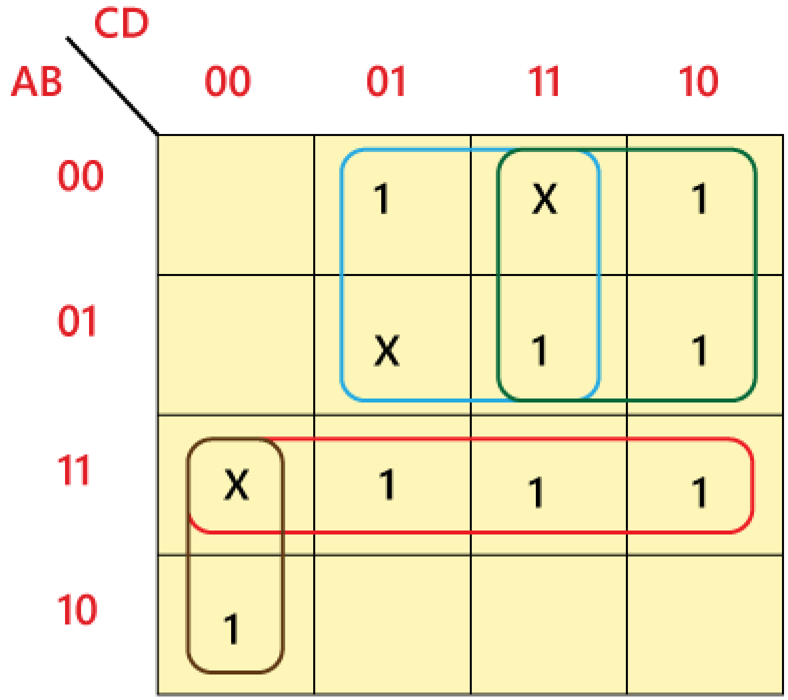
- The "Don't care" condition says that we can use the blank cells of a K-map to make a group of the variables. To make a group of cells, we can use the "don't care" cells as either 0 or 1, and if required, we can also ignore that cell. We mainly use the "don't care" cell to make a large group of cells.

- The cross(x) symbol is used to represent the "don't care" cell in K-map. This cross symbol represents an invalid combination. The "don't care" in excess-3 code are 0000, 0001, 0010, 1101, 1110, and 1111 because they are invalid combinations. Apart from this, the 4-bit BCD to Excess-3 code, the "don't care" are 1010, 1011, 1100, 1101, 1110, and 1111.

Don't Cares in K-Maps

Minimize the following function in SOP minimal form using K-Maps:
F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(3, 5, 12)

The SOP K-map for the given expression is:



Therefore,
f = AC'D' + A'D + A'C + AB

Don't Cares in K-Maps

Example 2: Minimize F(A,B,C,D) = m(0,1,2,3,4,5) + d(10,11,12,13,14,15) in POS minimal form

The POS form of the given function is:
F(A,B,C,D) = M(6,7,8,9) + d(10,11,12,13,14,15)

The POS K-map for the given expression is:

AB \ CD	00	01	11	10
00				
01			0	0
11	X	X	X	X
10	0	0	X	X

So, the minimized POS form of the function is:
F = A'(B' + C')

Significance of Don't Cares in K-Maps

- **Simplification:**

These conditions denote the set of inputs that never occurs for given digital circuits. Therefore, to simplify the Boolean output expressions, the 'don't care' are used.

- **Reduced Power Consumption:**

The switching of the state is reduced when we group the terms long with "don't care". This reduces the required memory space resulting in lower power consumption.

- **Lesser number of gates:**

For reducing the number of gates that are used to implement the given expression, simplification places an important role. So, the 'don't care' makes the logic design more economical.

Product of Sums of Don't Care Example

- Find the optimum POS solution:

$$F(A,B,C,D) = \Sigma_m(3,9,11,12,13,14,15) + \Sigma d\,(1,4,6)$$

Hint: Use $\overline{F}$ and complement it to get the result.


GOOD NEWS!
THE CLASS IS OVER…
THANK YOU!
AFSALCMK