

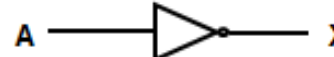
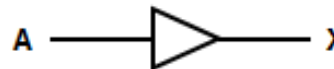






# K-Map

LECTURE 6  
CHAPTER#3

# Logic Gates

Name	Symbol	Function	Truth Table															
AND		$X = A \cdot B$ or $X = AB$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	0	1	0	0	1	1	1
A	B	X																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$X = A + B$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	1
A	B	X																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
I		$X = A$	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0									
A	X																	
0	1																	
1	0																	
Buffer		$X = A$	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	X	0	0	1	1									
A	X																	
0	0																	
1	1																	
NAND		$X = (AB)'$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	1	1	0	1	1	1	0
A	B	X																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$X = (A + B)'$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	1	0	1	0	1	0	0	1	1	0
A	B	X																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR Exclusive OR		$X = A \oplus B$ or $X = A'B + AB'$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	X	0	0	0	0	1	1	1	0	1	1	1	0
A	B	X																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR Exclusive NOR or Equivalence		$X = (A \oplus B)'$ or $X = A'B' + AB$	<table><tr><th>A</th><th>B</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	X	0	0	1	0	1	0	1	0	0	1	1	1
A	B	X																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

# Boolean Algebra

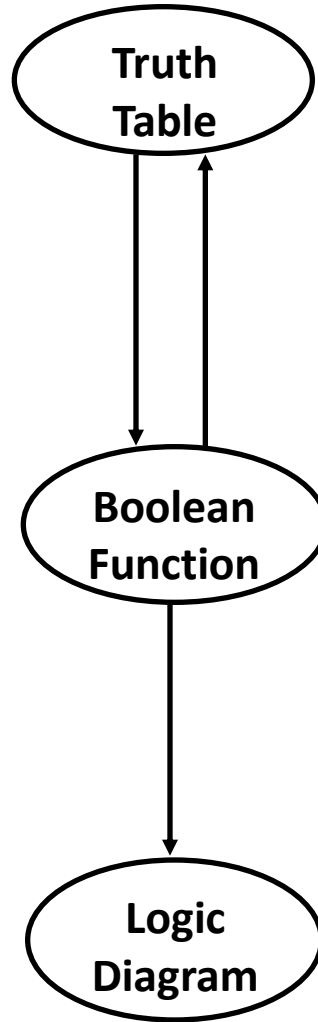
## Boolean Algebra

- \* Algebra with Binary(Boolean) Variable and Logic Operations
- \* Boolean Algebra is useful in Analysis and Synthesis of Digital Logic Circuits
  - Input and Output signals can be represented by Boolean Variables, and
  - Function of the Digital Logic Circuits can be represented by Logic Operations, i.e., **Boolean Function(s)**
  - From a Boolean function, a logic diagram can be constructed using AND, OR, and COMPLEMENT

## Truth Table

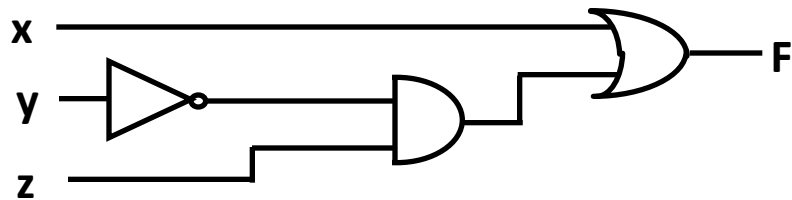
- \* The most elementary specification of the function of a Digital Logic Circuit is the Truth Table
  - Table that describes the Output Values for all the combinations of the Input Values, called *MINTERMS*
  - n input variables  $\rightarrow 2^n$  minterms

# Logic Circuit Design



x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

$$F = x + y'z$$



# Basic Identities Of Boolean Algebra

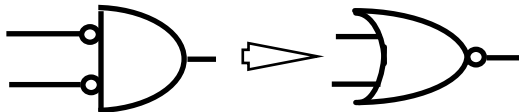
[1] $x + 0 = x$	[2] $x \cdot 0 = 0$
[3] $x + 1 = 1$	[4] $x \cdot 1 = x$
[5] $x + x = x$	[6] $x \cdot x = x$
[7] $x + x' = 1$	[8] $x \cdot x' = 0$
[9] $x + y = y + x$	[10] $xy = yx$
[11] $x + (y + z) = (x + y) + z$	[12] $x(yz) = (xy)z$
[13] $x(y + z) = xy + xz$	[14] $x + yz = (x + y)(x + z)$
[15] $(x + y)' = x'y'$	[16] $(xy)' = x' + y'$
[17] $(x')' = x$	

[15] and [16] : De Morgan's Theorem

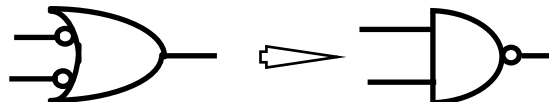
## Usefulness of this Table

- Simplification of the Boolean function
- Applications of De Morgans Theorem

$$x'y' = (x + y)'$$



$$x' + y' = (xy)'$$



# Equivalent Circuits

Many different logic diagrams are possible for a given Function

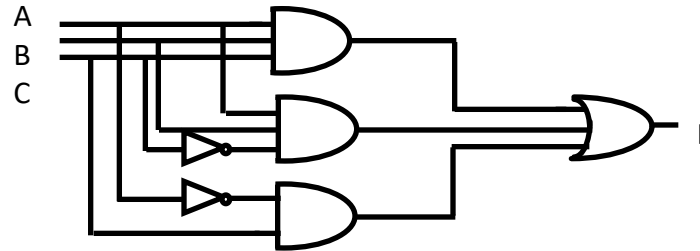
$$F = ABC + ABC' + A'C \quad \dots\dots\dots (1)$$

$$= AB(C + C') + A'C \quad [13] \dots\dots(2)$$

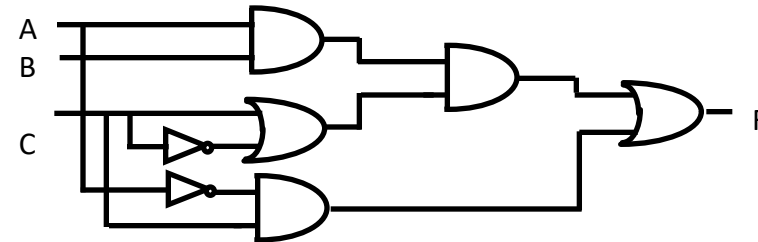
$$= AB \cdot 1 + A'C \quad [7]$$

$$= AB + A'C \quad [4] \dots\dots(3)$$

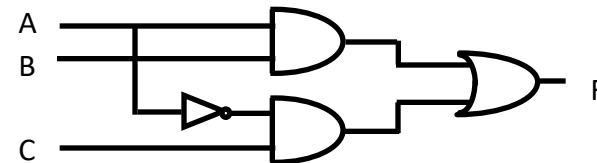
(1)



(2)



(3)



# Complement Of Functions

- ❖ A Boolean function of a digital logic circuit is represented by only using logical variables and AND, OR, and Invert operators.
- ❖ Complement of a Boolean function
- ❖ Replace all the variables and subexpressions in the parentheses appearing in the function expression with their respective complements

$$A,B,...,Z,a,b,...,z \Rightarrow A',B',...,Z',a',b',...,z'$$

- ❖ Replace all the operators with their respective complementary operators

$$\text{AND} \Rightarrow \text{OR}$$

$$\text{OR} \Rightarrow \text{AND}$$

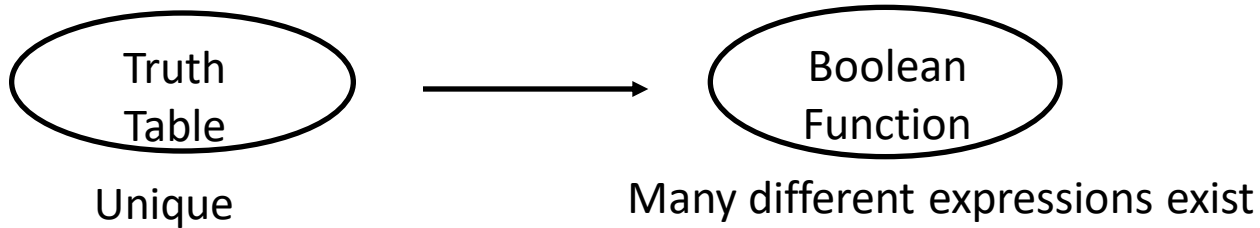
- ❖  $F=AB+C'D'+B'D \xrightarrow{\hspace{1cm}} F'=(A'+B')(C+D)(B+D')$

- ❖ Basically, extensive applications of the De Morgan's theorem

$$(x_1 + x_2 + \dots + x_n)' \Rightarrow x_1' x_2' \dots x_n'$$

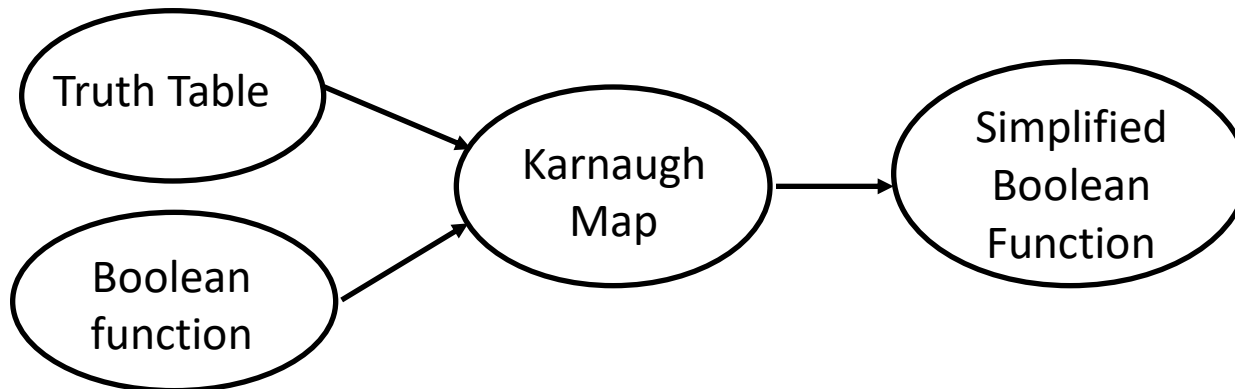
$$(x_1 x_2 \dots x_n)' \Rightarrow x_1' + x_2' + \dots + x_n'$$

# Map Simplification



Simplification from Boolean function

- ❖ Finding an equivalent expression that is least expensive to implement
- ❖ For a simple function, it is possible to obtain simple expression for low cost implementation
- ❖ But, with complex functions, it is a very difficult task
- ❖ **Karnaugh Map (K-map)** is a simple procedure for simplifying Boolean expressions.





# K-Map

- ❖ Simplifying Boolean Expressions using the laws, rules and theorems do not guarantee the simplest form of expression.
- ❖ The Karnaugh Map provides a systematic method for simplifying Boolean expressions.
- ❖ A Karnaugh Map is organized in the form of an array.
- ❖ Adjacent cells of the array can be grouped together to result in simplification of a given expression.
- ❖ Karnaugh Maps can be used to simplify expressions of 2, 3, 4 and 5 variables.

# K-Map

- ❖ A 3-variable K-Map has an array of 8 cells.
- ❖ The 8 cells can be arranged in 2 columns and 4 rows representing the column form of the Karnaugh Map.
- ❖ Alternately, the 8 cells can be organized in 2 rows and 4 columns representing the row form of the Karnaugh map.
- ❖ Any of the two forms of the Karnaugh Map can be used to simplify Boolean expressions.
- ❖ The simplified expressions using either of the two K-maps are identical.

# K-Map

- ❖ Considering first the column based 3-variable Karnaugh map.
- ❖ The binary values 00, 01, 11 and 10 in the left most column of the K-map represent the binary values of variables A and B.
- ❖ The binary values 0 and 1 in the top row of the K-map represent the binary values of variable C.
- ❖ The 3-variable K-Map based on the row representation is considered next.
- ❖ The binary values 0 and 1 in the left most column of the K-map represent the binary values of variable A. <sup>11</sup>

# K-Map

- ❖ The binary values 00, 01, 11 and 10 in the top row of the K-map represent the binary values of variables B and C
- ❖ The numbers in the cells represent the Minterms or Maxterms of an expression that is to be represented using the K-map.
- ❖ The cell marked 0 for example, represents the minterm 0 or the maxterm 0 having binary value of variables A, B and C equal to 000.
- ❖ Similarly cell marked 5 represents the minterm 5 or the maxterm 5 having binary values of variables A, B and C equal to 101.

# K-Map

## The 3-variable Karnaugh Map

AB\C	0	1
00	0	1
01	2	3
11	6	7
10	4	5

A\BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Figure 10.1 Column and Row based 3-variable Karnaugh Maps

# K-Map

- ❖ A 4-variable K-Map has an array of 16 cells
- ❖ The numbers in the cells represent the Minterms and Maxterms of an expression that is to be represented using the K-map.
- ❖ The 4-variable K-Map has a square format with four rows and four columns of cells.
- ❖ The binary values 00, 01, 11 and 10 in the left most column of the K-map represent the binary values of variables A and B.
- ❖ The binary values 00, 01, 11 and 10 in the top row of the K-map represents the binary values of variables C and D

# K-Map

- ❖ The 16 cells marked with numbers 0 to 15 represent the cells 0 to 15 corresponding to the minterms 0 to 15 or the maxterms 0 to 15 in a 4 variable Boolean expression.
- ❖ The cell marked 6 for example, represents the minterm 6 or the maxterm 6 having binary value of variables A, B, C and D equal to 0110.
- ❖ Similarly cell marked 13 represents the minterm 13 or the maxterm 13 having binary values of variables A, B, C and D equal to 1101.

# K-Map

		$B$	
		0	1
$A$	0	0	1
	1	2	3

(a) Two-variable map

		$B$			
		$BC$	00	01	11
$A$	0	0	1	3	2
	1	4	5	7	6

(b) Three-variable map

		$C$			
		$CD$	00	01	11
$A$	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10

(c) Four-variable map

		$y$			
		$yz$	00	01	11
$x$	0	$m_0$ $x'y'z'$	$m_1$ $x'y'z$	$m_3$ $x'yz$	$m_2$ $x'yz'$
	1	$m_4$ $xy'z'$	$m_5$ $xy'z$	$m_7$ $xyz$	$m_6$ $xyz'$
		$z$			



# K-Map

- ❖ Each combination of truth table is called minterm
- ❖ The truth table can be represented as:

$$F(X,Y,Z)=\sum (1,4,5,6,7)$$

- ❖ Minterms that produce 1 are written in decimal while others that produce 0 are not
- ❖ The map is a diagram made up of squares with each square representing one min term

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- ❖ The squares corresponding to minterms that produce 1 for the function are marked by 1 and others are marked by 0 or empty

# K-Map

- ❖ The squares containing 1's are combined in groups of adjacent squares
- ❖ These groups must contain number of 1's as integral power of 2 e.g. 2,4,8....
- ❖ Group of combined adjacent squares may share one or more squares with one or more groups
- ❖ Each group of squares represents an algebraic term, and the OR of those terms gives simplified algebraic expression for the function

# K-Map

- ❖ Karnaugh Map Array is considered to be wrapped around where all sides are adjacent to each other.
- ❖ Groups of 2, 4, 8, 16, 32 etc. adjacent cells are formed. Adjacent cells can be
  - ❖ row wise
  - ❖ column wise
  - ❖ four corner cells
  - ❖ row-column groups of 4, 8, 16, 32 etc
- ❖ Groups are formed on the basis of 1s (Minterms) or 0s (maxterms).

# K-Map

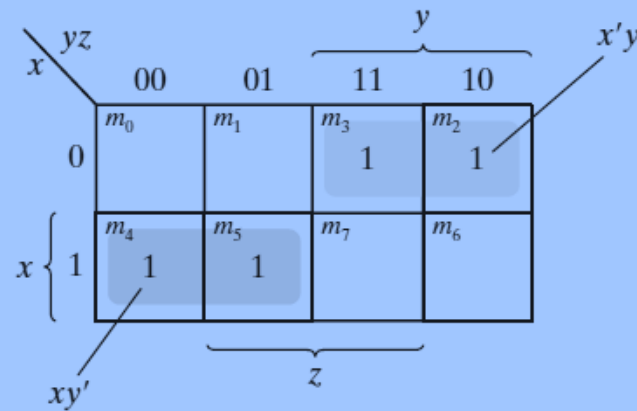
- ❖ A group is selected to have maximum number of cells of Minterms or Maxterms, keeping in view that the size of the group should be a power of 2.
- ❖ The idea is to form minimal number of largest groups that uniquely cover all the cells, thereby ensuring that all minterms or maxterms are included.

# K-Map

Simplify the Boolean function

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

First, a 1 is marked in each minterm square that represents the function. This is shown in Fig. 3.4, in which the squares for minterms 010, 011, 100, and 101 are marked with 1's. The next step is to find possible adjacent squares. These are indicated in the map by two shaded rectangles, each enclosing two 1's. The upper right rectangle represents the area enclosed by  $x'y$ . This area is determined by observing that the two-square area is in row 0, corresponding to  $x'$ , and the last two columns, corresponding to  $y$ . Similarly, the lower left rectangle represents the product term  $xy'$ . (The second row represents  $x$  and the two left columns represent  $y'$ .) The sum of four minterms can be replaced by a sum of



**FIGURE 3.4**

Map for Example 3.1,  $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

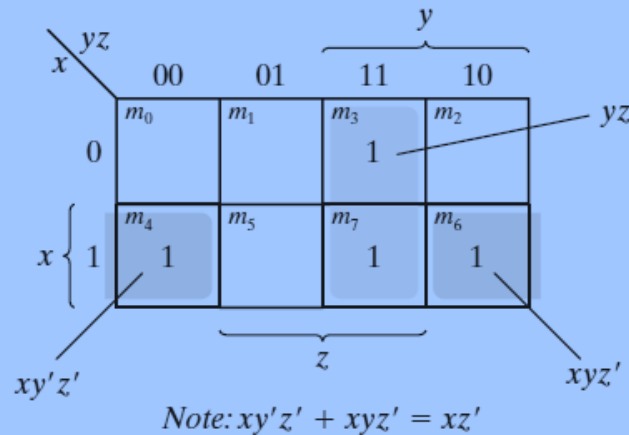
# K-Map

Simplify the Boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

The map for this function is shown in Fig. 3.5. There are four squares marked with 1's, one for each minterm of the function. Two adjacent squares are combined in the third column to give a two-literal term  $yz$ . The remaining two squares with 1's are also adjacent by the new definition. These two squares, when combined, give the two-literal term  $xz'$ . The simplified function then becomes

$$F = yz + xz'$$



**FIGURE 3.5**

Map for Example 3.2,  $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

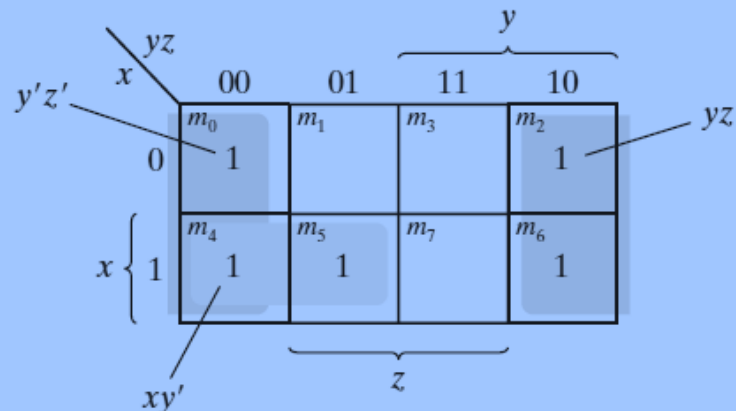
# K-Map

Simplify the Boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

The map for  $F$  is shown in Fig. 3.6. First, we combine the four adjacent squares in the first and last columns to give the single literal term  $z'$ . The remaining single square, representing minterm 5, is combined with an adjacent square that has already been used once. This is not only permissible, but rather desirable, because the two adjacent squares give the two-literal term  $xy'$  and the single square represents the three-literal minterm  $xy'z$ . The simplified function is

$$F = z' + xy'$$



Note:  $y'z' + yz' = z'$

**FIGURE 3.6**

Map for Example 3.3,  $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

# K-Map

## Example 1 & 2

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

A\BC	00	01	11	10
0	0	1	1	1
1	1	0	0	0

Figure 10.10 Simplification of SOP expression using a 3-variable K-Map



# K-Map

## Example 3 & 4

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

A\BC	00	01	11	10
0	0	0	1	1
1	1	1	1	0

Figure 10.11 Simplification of SOP expression using a 3-variable K-Map

# K-Map

One square represents one minterm, giving a term with four literals.

Two adjacent squares represent a term with three literals.

Four adjacent squares represent a term with two literals.

Eight adjacent squares represent a term with one literal.

Sixteen adjacent squares produce a function that is always equal to 1.

No other combination of squares can simplify the function. The next two examples show the procedure used to simplify four-variable Boolean functions.

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

(a)

		$y$			
		$yz$		11	10
$w$	$wx$	00	01		
	00	$m_0$ $w'x'y'z'$	$m_1$ $w'x'y'z$	$m_3$ $w'x'yz$	$m_2$ $w'x'yz'$
	01	$m_4$ $w'xy'z'$	$m_5$ $w'xy'z$	$m_7$ $w'xyz$	$m_6$ $w'xyz'$
	11	$m_{12}$ $wxy'z'$	$m_{13}$ $wxy'z$	$m_{15}$ $wxyz$	$m_{14}$ $wxyz'$
	10	$m_8$ $wx'y'z'$	$m_9$ $wx'y'z$	$m_{11}$ $wx'yz$	$m_{10}$ $wx'yz'$
		$z$			

(b)

**FIGURE 3.8**

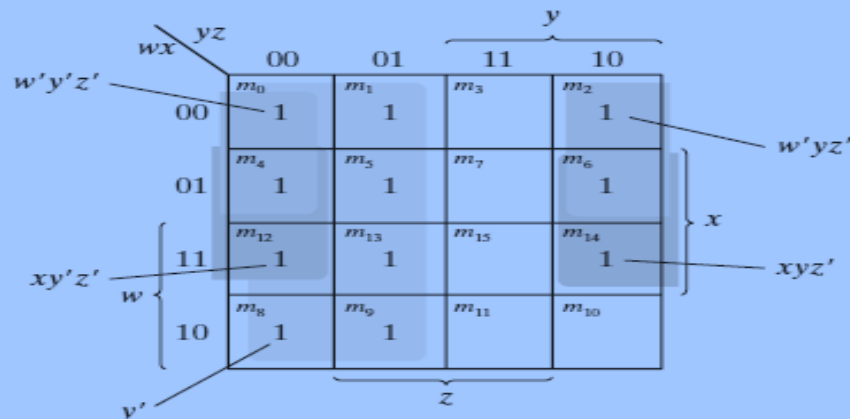
Four-variable map

# K-Map

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

Since the function has four variables, a four-variable map must be used. The minterms listed in the sum are marked by 1's in the map of Fig. 3.9. Eight adjacent squares marked with 1's can be combined to form the one literal term  $y'$ . The remaining three 1's on the right cannot be combined to give a simplified term; they must be combined as two or four adjacent squares. The larger the number of squares combined, the smaller is the number of literals in the term. In this example, the top two 1's on the right are combined with the top two 1's on the left to give the term  $w'z'$ . Note that it is permissible to use the same square more than once. We are now left with a square marked by 1 in the third row and fourth column (square 1110). Instead of taking this square alone (which will give a term with four literals), we combine it with squares already used to form an area of four adjacent squares. These squares make up the two middle rows and the two end columns, giving the term  $xz'$ . The simplified function is

$$F = y' + w'z' + xz'$$



$$\begin{aligned} \text{Note: } w'y'z' + w'yz' &= w'z' \\ xy'z' + xyz' &= xz' \end{aligned}$$

**FIGURE 3.9**

Map for Example 3.5,  $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

# K-Map

## Example 5

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

Figure 10.12 Simplification of SOP expression using a 4-variable K-Map

# K-Map

## Example 6

An SOP expression having 8 minterms is mapped to a 4-variable based K-map. One group of two cells and two groups of four cells are formed.

- The first group of 1s comprising of cells 8 and 12 forms the product term  $A.\overline{C}.\overline{D}$
- The second group of 1s comprising of cells 3, 7, 11 and 15 forms the product term  $C.D$
- The third group of 1s comprising of cells 6, 7, 14 and 15 forms the product term  $B.C$

The eight term SOP expression has simplified to a 3 term expression  $A.\overline{C}.\overline{D} + C.D + B.C$

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

Figure 10.13 Simplification of SOP expression using a 4-variable K-Map

# K-Map

## Example 7

An SOP expression having 9 minterms is mapped to a 4-variable based K-map. Two group of two cells and two groups of four cells are formed.

- The first group of 1s comprising of corner cells 0, 2, 8 and 10 forms the product term  $\overline{B}.\overline{D}$
- The second group of 1s comprising of cells 2, 3, 10 and 11 forms the product term  $\overline{B}.C$
- The third group of 1s comprising of cells 13 and 15 forms the product term  $A.B.D$
- The fourth group of 1s comprising of cells 2 and 6 forms the product term  $\overline{A}.C.\overline{D}$

The nine term SOP expression has simplified to a 4 term SOP expression  $\overline{B}.\overline{D} + \overline{B}.C + A.B.D + \overline{A}.C.\overline{D}$

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

Figure 10.14 Simplification of SOP expression using a 4-variable K-Map

# POS/SOP

Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

The 1's marked in the map of Fig. 3.12 represent all the minterms of the function. The squares marked with 0's represent the minterms not included in  $F$  and therefore denote the complement of  $F$ . Combining the squares with 1's gives the simplified function in sum-of-products form:

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

If the squares marked with 0's are combined, as shown in the diagram, we obtain the simplified complemented function:

$$F' = AB + CD + BD'$$

Applying DeMorgan's theorem (by taking the dual and complementing each literal as described in Section 2.4), we obtain the simplified function in product-of-sums form:

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

The gate-level implementation of the simplified expressions obtained in Example 3.7 is shown in Fig. 3.13. The sum-of-products expression is implemented in (a) with a group of AND gates, one for each AND term. The outputs of the AND gates are connected to the inputs of a single OR gate. The same function is implemented in (b) in its product-of-sums

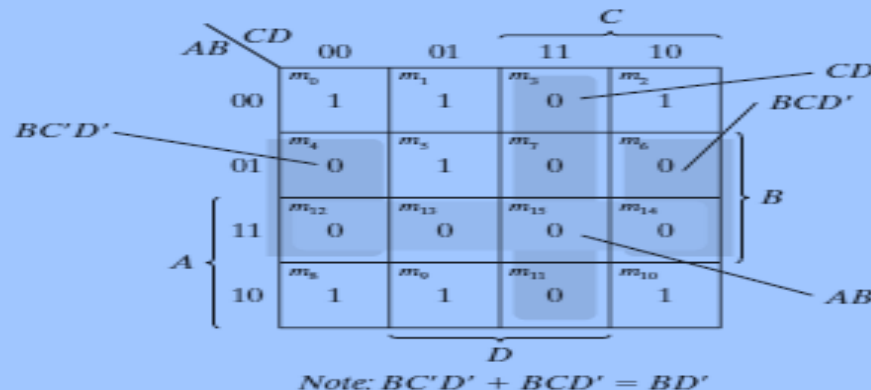
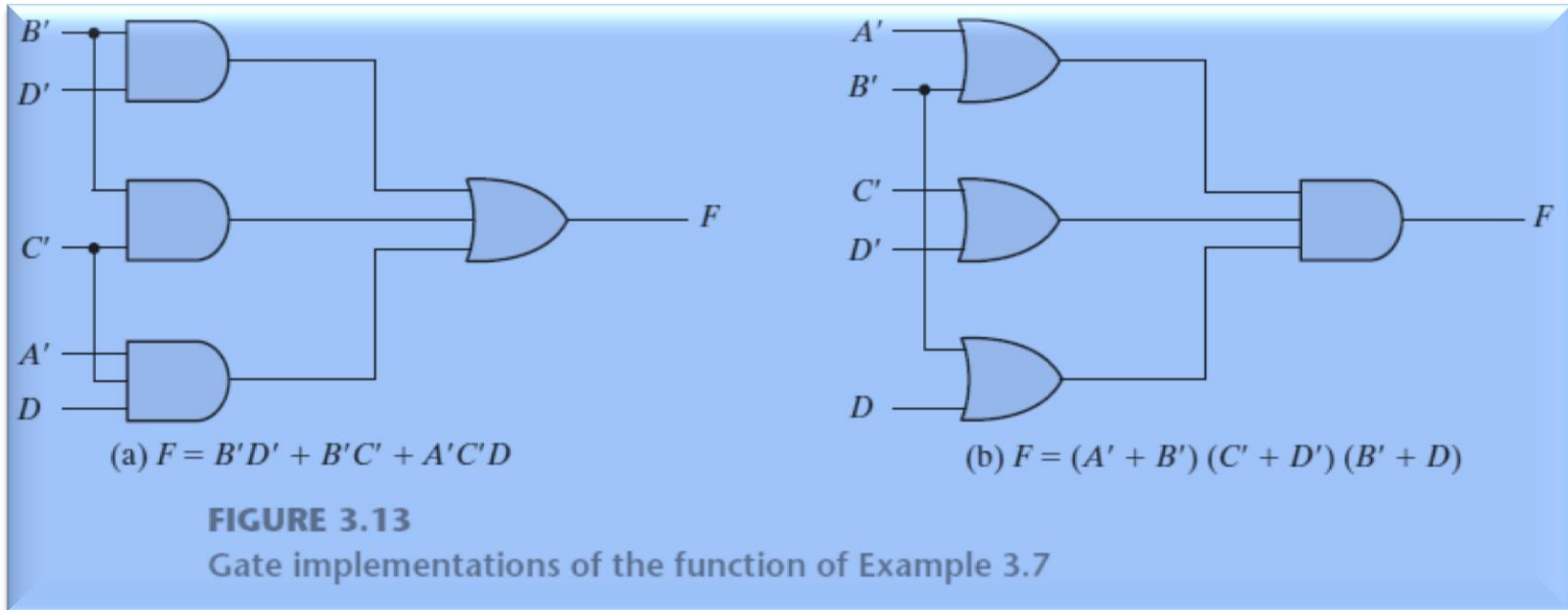


FIGURE 3.12

Map for Example 3.7,  $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10) = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$

# POS/SOP



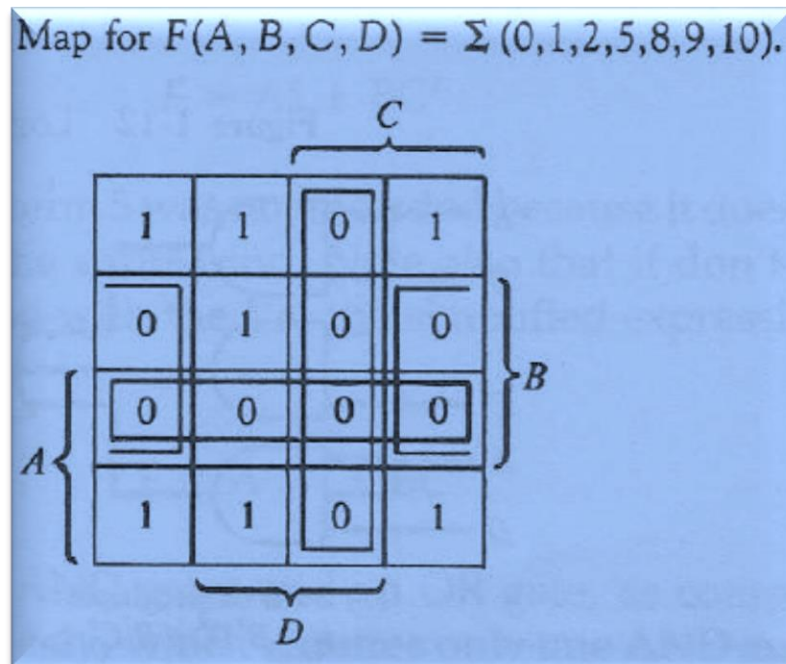


# Product Of Sum (POS)/SOP

❖ Suppose we have

$$F(A, B, C, D) = \sum (0, 1, 2, 5, 8, 9, 10)$$

❖ We will simplify it by SOP/POS



# Product Of Sum (POS)/SOP

- ❖ By combining 1's we get SOP

$$F = B'D' + B'C' + A'C'D$$

- ❖ If the squares marked with 0's are combined we obtain simplified complemented function

$$F' = AB + CD + BD'$$

- ❖ Taking the complement of  $F'$  we obtain simplified function in POS form

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

## K-MAPS Don't-Care Conditions

In some logic circuits, the output responses for some input conditions are don't care whether they are 1 or 0.

In K-maps, don't-care conditions are represented by  $\times$  in the corresponding cells. Don't-care conditions are useful in minimizing the logic functions using K-map.

- Can be considered either 1 or 0
- Thus increases the chances of merging cells into the larger cells
- > Reduce the number of variables in the product terms

The 1's and 0's in the map represent the minterms that make the function equal to 1 or 0. There are occasions when it does not matter if the function produces 0 or 1 for a given minterm. Since the function may be either 0 or 1, we say that we don't care what the function output is to be for this minterm. Minterms that may produce either 0 or 1 for the function are said to be don't-care conditions and are marked with an  $\times$  in the map. These don't-care conditions can be used to provide further simplification of the algebraic expression.

# Don't care condition

Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

which has the don't-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

		y			
		00	01	11	10
wx	yz	m <sub>0</sub>	m <sub>1</sub>	m <sub>3</sub>	m <sub>2</sub>
	00	X	1	1	X
w'x'	01	m <sub>4</sub>	m <sub>5</sub>	m <sub>7</sub>	m <sub>6</sub>
	01	0	X	1	0
w	11	m <sub>12</sub>	m <sub>13</sub>	m <sub>15</sub>	m <sub>14</sub>
	11	0	0	1	0
w	10	m <sub>8</sub>	m <sub>9</sub>	m <sub>11</sub>	m <sub>10</sub>
	10	0	0	1	0

Diagram (a) shows a 4x4 Karnaugh map for the function  $F(w, x, y, z)$ . The map is labeled with variables  $w, x, y, z$  and their complements. The function is defined by the minterms  $\Sigma(1, 3, 7, 11, 15)$  and the don't-care conditions  $\Sigma(0, 2, 5)$ . The map shows the following values:

- Row 0 (w'x'): m<sub>0</sub> = X, m<sub>1</sub> = 1, m<sub>3</sub> = 1, m<sub>2</sub> = X
- Row 1 (w'x'): m<sub>4</sub> = 0, m<sub>5</sub> = X, m<sub>7</sub> = 1, m<sub>6</sub> = 0
- Row 2 (w): m<sub>12</sub> = 0, m<sub>13</sub> = 0, m<sub>15</sub> = 1, m<sub>14</sub> = 0
- Row 3 (w): m<sub>8</sub> = 0, m<sub>9</sub> = 0, m<sub>11</sub> = 1, m<sub>10</sub> = 0

The map is grouped into four 2x2 blocks, each representing a prime implicant:  $yz$  (top-left),  $w'x'$  (top-right),  $w'z$  (bottom-left), and  $wz$  (bottom-right). The function is simplified as  $F = yz + w'x'$ .

(a)  $F = yz + w'x'$

		y			
		00	01	11	10
wx	yz	m <sub>0</sub>	m <sub>1</sub>	m <sub>3</sub>	m <sub>2</sub>
	00	X	1	1	X
w'z	01	m <sub>4</sub>	m <sub>5</sub>	m <sub>7</sub>	m <sub>6</sub>
	01	0	X	1	0
w	11	m <sub>12</sub>	m <sub>13</sub>	m <sub>15</sub>	m <sub>14</sub>
	11	0	0	1	0
w	10	m <sub>8</sub>	m <sub>9</sub>	m <sub>11</sub>	m <sub>10</sub>
	10	0	0	1	0

Diagram (b) shows a 4x4 Karnaugh map for the function  $F(w, x, y, z)$ . The map is labeled with variables  $w, x, y, z$  and their complements. The function is defined by the minterms  $\Sigma(1, 3, 7, 11, 15)$  and the don't-care conditions  $\Sigma(0, 2, 5)$ . The map shows the following values:

- Row 0 (w'x'): m<sub>0</sub> = X, m<sub>1</sub> = 1, m<sub>3</sub> = 1, m<sub>2</sub> = X
- Row 1 (w'x'): m<sub>4</sub> = 0, m<sub>5</sub> = X, m<sub>7</sub> = 1, m<sub>6</sub> = 0
- Row 2 (w): m<sub>12</sub> = 0, m<sub>13</sub> = 0, m<sub>15</sub> = 1, m<sub>14</sub> = 0
- Row 3 (w): m<sub>8</sub> = 0, m<sub>9</sub> = 0, m<sub>11</sub> = 1, m<sub>10</sub> = 0

The map is grouped into four 2x2 blocks, each representing a prime implicant:  $yz$  (top-left),  $w'z$  (top-right),  $w'x'$  (bottom-left), and  $wz$  (bottom-right). The function is simplified as  $F = yz + w'z$ .

(b)  $F = yz + w'z$

# Don't care condition

- ❖ When choosing adjacent squares for the function in map, the don't care may be assumed either 0 or 1
- ❖ In addition don't care need not to be include if it is not taking part in simplification

	$B$			
	0	1	0	1
$A$	0	1	0	1
	$C$			

$$F(A, B, C) = \sum (0, 2, 6)$$

$$d(A, B, C) = \sum (1, 3, 5)$$

$$F = A'C' + BC'$$

$$F = A' + BC'$$

# Mapping a Standard POS Expression

- POS expressions can be easily simplified by use of the K-Map in a manner similar to the method adopted for simplifying SOP expressions.
- After the POS expression is mapped on the K-map, groups of 0s are marked instead of 1s based on the rules for forming groups used for simplifying SOP.

# Mapping a Standard POS Expression

## Example 1 & 2

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

A\BC	00	01	11	10
0	0	1	1	1
1	1	0	0	0

Figure 11.2 Simplification of POS expression using a 3-variable K-Map

# Mapping a Standard POS Expression

A POS expression having 3 Maxterms is mapped to a 3-variable column based K-map. A single group of two cells and a group of one cell are formed.

- The first group of 0s comprising of cells 0 and 4 forms the sum term  $(B + C)$
- The second group comprising of cell 3 forms the sum term  $(A + \bar{B} + \bar{C})$

The three term POS expression simplifies to a 2 term POS expression  $(B + C).(A + \bar{B} + \bar{C})$ .

A POS expression having 4 Maxterms is mapped to a 3-variable column based K-map. Two groups of 2 cells each and a third group of single cell are formed.

- The single cell group comprising of cell 0 forms the sum term  $(A + B + C)$
- The second group of 0s comprising of cells 5 and 7 forms the sum term  $(\bar{A} + \bar{C})$
- The third group of 0s comprising of cells 6 and 7 forms the sum term  $(\bar{A} + \bar{B})$

The four term POS expression simplifies to a 3 term POS expression  $(A + B + C).(\bar{A} + \bar{C}).(\bar{A} + \bar{B})$ .



# Mapping a Standard POS Expression

## Example 3 & 4

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

A\BC	00	01	11	10
0	0	0	1	1
1	1	1	1	0

Figure 11.3 Simplification of POS expression using a 3-variable K-Map

A POS expression having 3 Maxterms is mapped to a 3-variable column based K-map. Two groups of two cells are formed.

- The first group of 0s comprising of cells 0 and 1 forms the sum term  $(A + B)$
- The second group of 0s comprising of cells 0 and 4 forms the sum term  $(B + C)$

The three term POS expression simplifies to a 2 terms POS expression  $(A + B).(B + C)$

A POS expression having 3 Maxterms is mapped to a 3-variable column based K-map. One group of 2 cells and another group of single cell are formed.

- The first group of 0s comprising of cell 0 and 1 forms the sum term  $(A + B)$
- The second group comprising of cell 6 forms the sum term  $(\bar{A} + \bar{B} + C)$

The three term POS expression simplifies to a 2 term POS expression  $(A + B).(\bar{A} + \bar{B} + C)$