# Digital Logic and Design

Karnaugh Map

# Recap: Boolean Algebra

- Simplification of Boolean expressions are done through Boolean algebra.
- The method wasn’t effective.
- Solving a problem through Boolean algebra was a bit tricky
- Need skill of applying rules and law.
- Doesn’t guarantee simplest form of expression

# Karnaugh Map

- K-map provides a systematic method
  - For simplifying Boolean expressions and minimizing expressions.
  - If properly used, will produce the simplest SOP and POS.
  - Similar to truth table, presents all possible values of input variables and resulting output for each value.

# Karnaugh Map

- Karnaugh map is an array of cells in which each cell represents a binary value of input variables.
- The cells are managed in a way so that simplification of a given expression is simply a matter of properly grouping the cells.
- Karnaugh maps can be used for expression with two, three, four and five variables, but we will discuss only 3-variable and 4-varable situations to illustrate the principles.

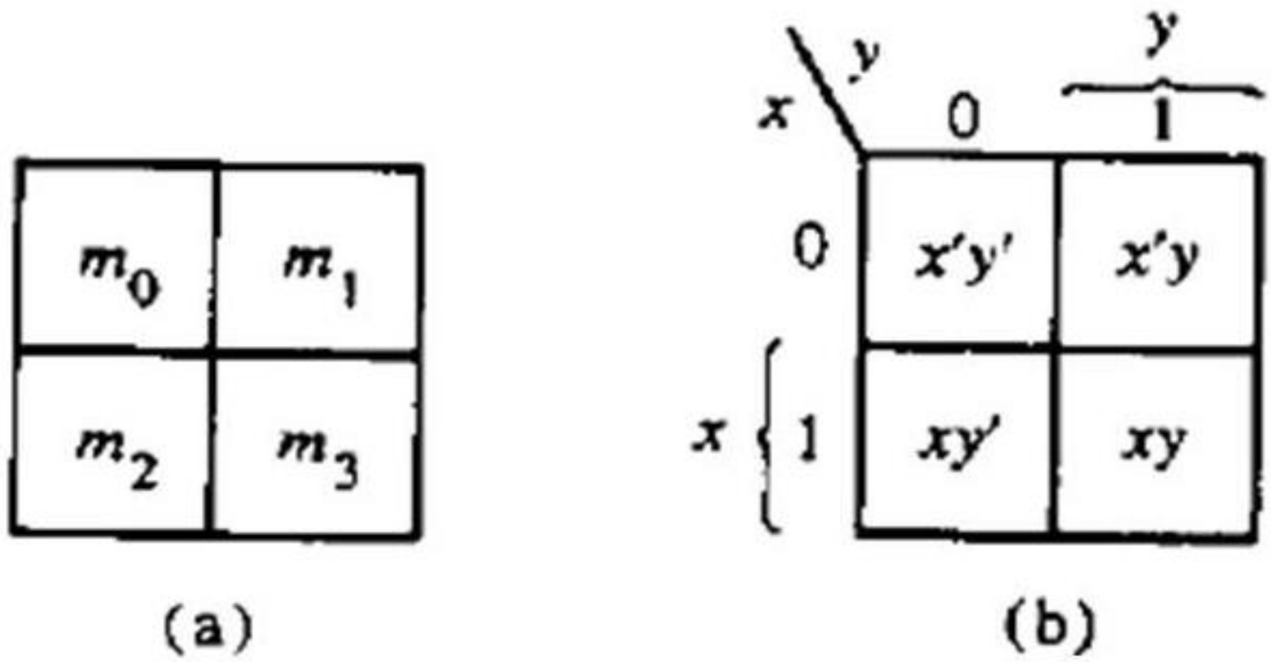
# Karnaugh Map

The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations as is the number of rows in a truth table.

- For two variables, the number of cells is $2^2 = 4$.
- For three variables, the number of cells is $2^3 = 8$.
- For four variables, the number of cells are $2^4 = 16$.

# 2-variable K-map

A two variable has four minterms, hence it has 4 squares one for each term, as shown below,



# 3-variable karnaugh map

- The 3-variable karnaugh map is an array of eight cells.
- In this case A, B and C are used for the variables although other letters could be used.
- Binary values of A is along the left side and the value of B and C is across the top.

# The 3-variable Karnaugh map

- The value of a given cell is the binary values of A at the left in the same row combined with the value of B and C at the top in the same column.
- For example: the cell in the upper left corner has a binary value of 000 and the cell in the lower right corner has a binary value of 110.

# 3-Variable K-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 |

# The 4-variable karnaugh map

- The 4-variable karnaugh map is an array of sixteen cells
- Binary values of A and B are along the left side and the values of C and D are across the top.
- The value of a given cell is the binary value of A and B at the left in the same row combined with the binary values of C and D at the top of the same column.

# 4-Variable K-map

| AB\CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

# Grouping & Adjacent Cells

- K-map is considered to be wrapped around
- All sides are adjacent to each other
- Adjacent cells which has 1’s in SOP can be grouped together in 2’s power.
  - 2 adjacent cells can be grouped (pair)
  - 4 adjacent cells can be grouped (Quads)
  - 8 adjacent cells can be grouped (octets)
- Groups can be row, column, square  or rectangular.
- Groups of diagonal cells are not allowed

# Mapping of Standard SOP expression

- Selecting n-variable K-map
- 1 marked in cell for each minterm

# Mapping of Standard SOP expression

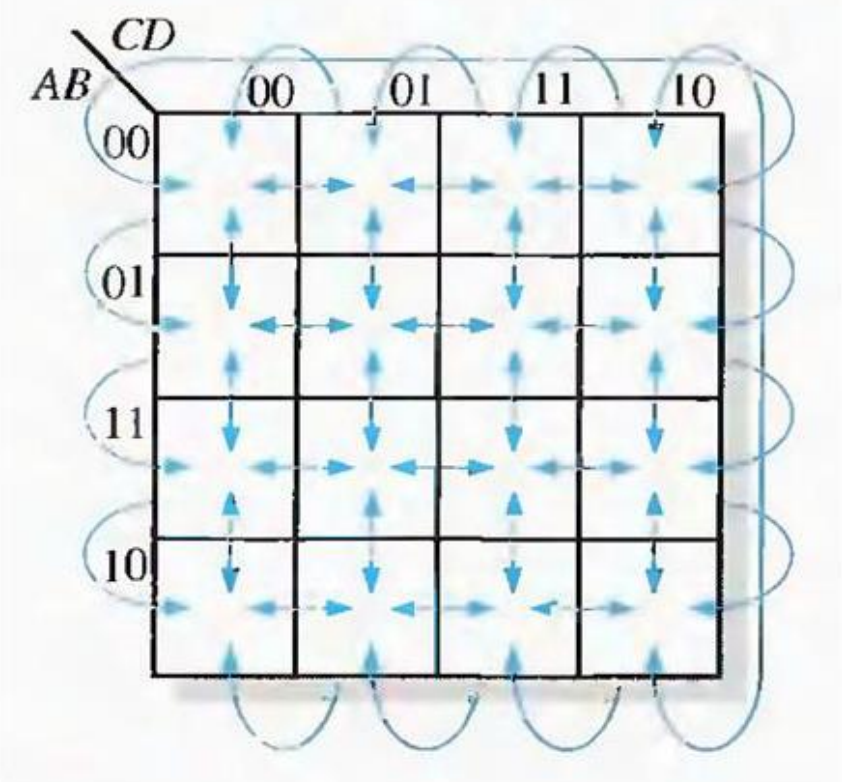
- SOP expression $AB\overline{C} + A\overline{B}\,\overline{C} + \overline{A}B\overline{C}$

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

# Cell Adjacency

- The cells in a karnaugh map are arranged so that there is only a single-variable change between adjacent cells.
- Adjacency is defined by a single variable change.
- Cells with values that differ by more than one variable are not adjacent.

# Cell Adjacency




Adjacent cells on a Karnaugh map are those that differ by only one variable. Arrows point between adjacent cells.

# Continued ( simplification through K maps)

- Karnaugh map is used for simplifying Boolean expressions to their minimum form.
- A minimized SOP expression contains the fewest possible terms with the fewest possible variables per term.
- Generally a minimum SOP expression can be implemented with fewer logic gates than a standard expression.

## Mapping a Standard SOP Expression

- For an SOP expression in standard form, a 1 is placed on the Karnaugh map for each product term in the expression.
- Each 1 is placed in a cell corresponding to the value of a product term. For example, for the product term AB′C, a 1 goes in the 101 cell on a 3-variable map.

# Mapping SOP expression

When an SOP expression is completely mapped, there will be a number of 1's on the Karnaugh map equal to the number of product terms in the standard SOP expression.

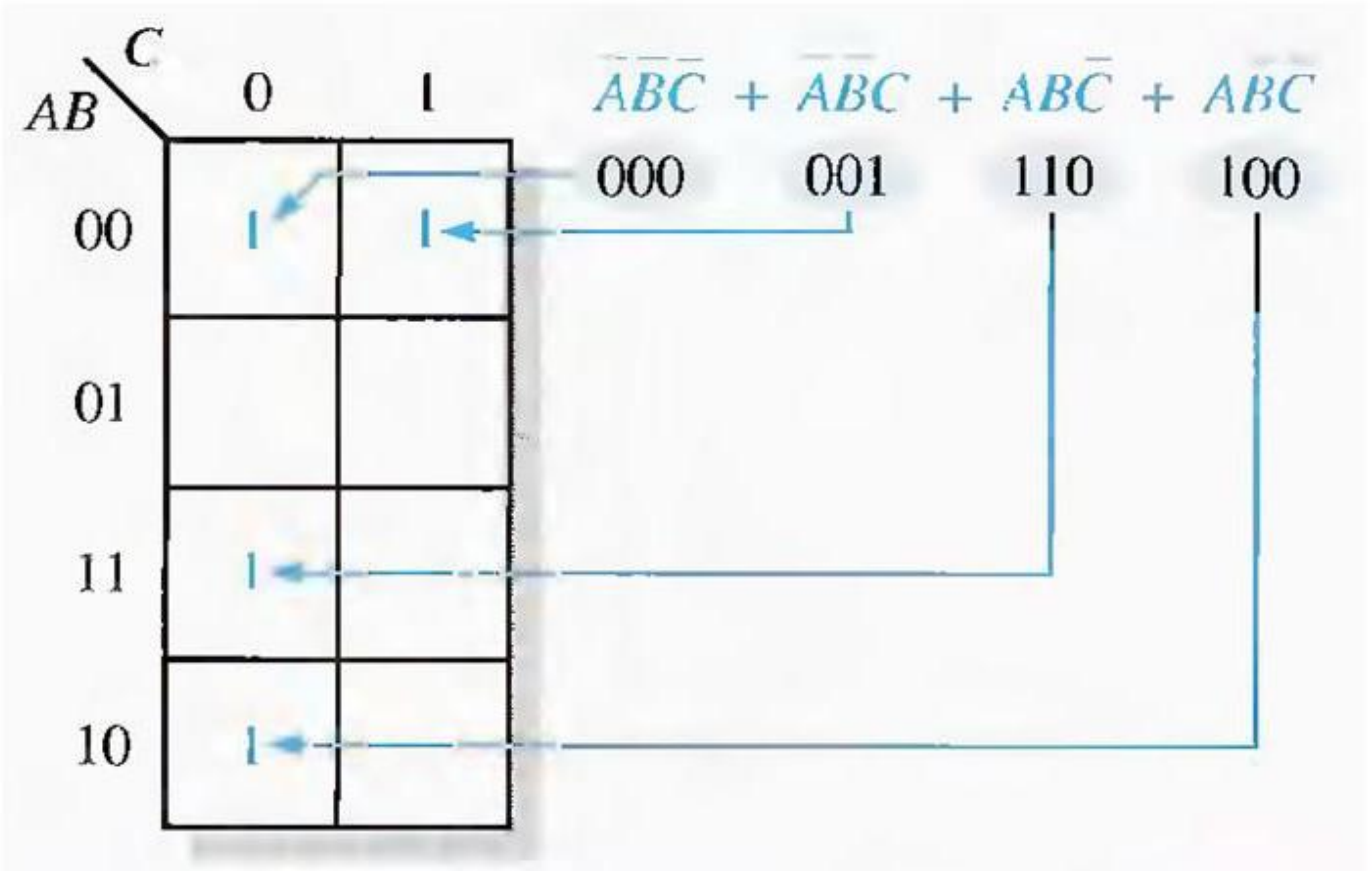
The cells that do not have a 1 are the cells for which the expression is O.

Usually, when working with SOP expressions, the O's are left off the map.

The following steps are used for mapping process.

- **Step 1-** Determine the binary value of each product term in the standard SOP expression. After some practice, you can usually do the evaluation of terms mentally.
- **Step 2.** As each product term is evaluated, place a 1 on the Karnaugh map in the cell having the same value as the product term.

# Mapping entries in K-map



# Class activity

Map the following standard SOP expression on a Karnaugh map:

$$A'B'C + A'BC' + ABC' + ABC$$

# Solution

**A′B′C + A′BC′ + ABC′ + ABC**

- Solution:

**A′B′C + A′BC′ + ABC′ + ABC**

0 0 1 + 0 1 0 + 1 1 0 + 1 1 1

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

- Map the standard SOP expression

**A′ BC + AB′ C + AB′ C′**

## Self Assessment

**Problem:**

- Map the standard SOP expression

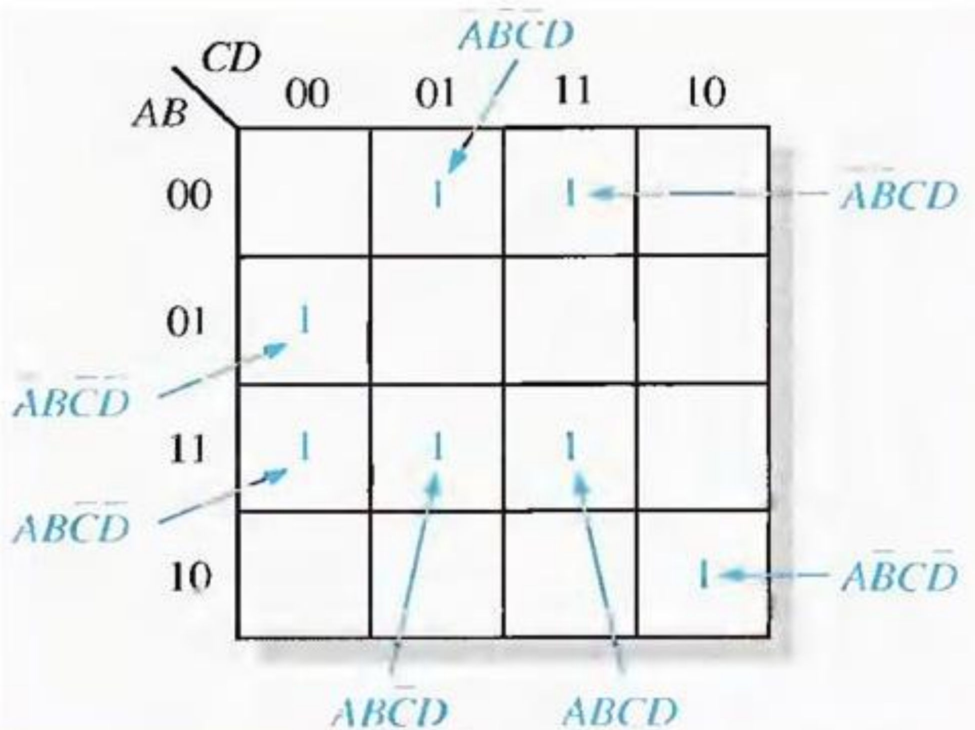
  **A′ BC + AB′ C + AB′ C′**

  on a Karnaugh map just in 3 minutes, hurry up.

Map the following standard SOP expression on a Karnaugh map:

- **A′ B′ CD + A′ BC′ D′ + ABC′ D + ABCD + ABC′ D′ + A′ B′ C′ D + AB′ CD′**

**Solution:**



## Self Assessment

**Problem:**

- Map the following standard SOP expression on a Karnaugh map:

$$A'BCD' + ABCD' + ABC'D' + ABCD$$

**Map the above expression on a K-Map in just 2 Minutes**, Hurry Up.

# Mapping a non standard SOP expression

A Boolean expression must first be in standard form before you use a Karnaugh map.

If an expression is not in standard form, then it must be converted to standard form.

It can also be done through numerical expansion.

Numerical expansion is probably the most efficient approach.

## Numerical Expansion of a Nonstandard Product Term:

- Recall that a nonstandard product term has one or more missing variables.
- Assume that one of the product term in a certain 3-variable SOP expression is AB′.
- First, write the binary value of the two variables and attach a 0 for the missing variable C′: 100.
- Next, write the binary value of the two variables and attach a 1 for the missing variable C: 101

# Mapping a non standard SOP expression

As another example, assume that one of the product terms in a 3-variable expression is B.

This term can be expanded numerically to standard form as follows.

Write the binary value of the variable; then attach all possible values for the missing variables A and C as follows:

| $B$ |
|---|
| 010 |
| 011 |
| 110 |
| 111 |

**Note:**

**The four resulting binary numbers are the values of the standard SOP terms are A′ BC′ , A′ BC, ABC′, and ABC.**

Map the following SOP expression on a Karnaugh map:

$$A' + AB' + ABC'$$

The first term is missing two variables, the second term is missing one variable, and the third term is standard.

Numerical expansion of the given expression can be done as follows

# Mapping non standard SOP into K-map

**Now map each of the resulting binary values by placing a 1 in the appropriate cell of the 3-variable Karnaugh map**

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

| $\overline{A}$ | $+\ A\overline{B}$ | $+\ AB\overline{C}$ |
|---|---|---|
| 000 | 100 | 110 |
| 001 | 101 | |
| 010 | | |
| 011 | | |

**Problem**

Map the SOP expression

**BC + A′ C′**

on a Karnaugh map.

Map the following SOP expression on a Karnaugh map:

$B'C' + AB' + ABC' + AB'CD' + A'B'C'D + AB'CD$

The SOP expression is obviously not in standard form because each product term does not have four variables.

The first and second terms are both missing two variables, the third term is missing one variable, and the rest of the terms are standard.

# Continued…

First expand the terms by including all combinations of the missing variables numerically as follows:

| $\overline{B}\,\overline{C}$ | $A\overline{B}$ | $+\ AB\overline{C}$ | $+\ A\overline{B}C\overline{D}$ | $+\ \overline{A}\,\overline{B}\,\overline{C}D$ | $+\ A\overline{B}CD$ |
|---|---|---|---|---|---|
| 0000 | 1000 | 1100 | 1010 | 0001 | 1011 |
| 0001 | 1001 | 1101 | | | |
| 1000 | 1010 | | | | |
| 1001 | 1011 | | | | |

1. **Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4- variable Karnaugh map.**
2. **Notice that some of the values in the expanded expression are redundant.**

# Continued…

After mapping the expression the resulting 4 veriable k map look like this,

**Related Problem:**

Map the expression

**A + C′ D + ACD′ + A′ BCD′**

on a Karnaugh map?

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | | | | |
| 11 | 1 | 1 | | |
| 10 | 1 | 1 | 1 | 1 |

# K-Map Simplification

After an SOP expression has been mapped, a minimum SOP expression is obtained by grouping the Is and determining the minimum SOP expression from the map.

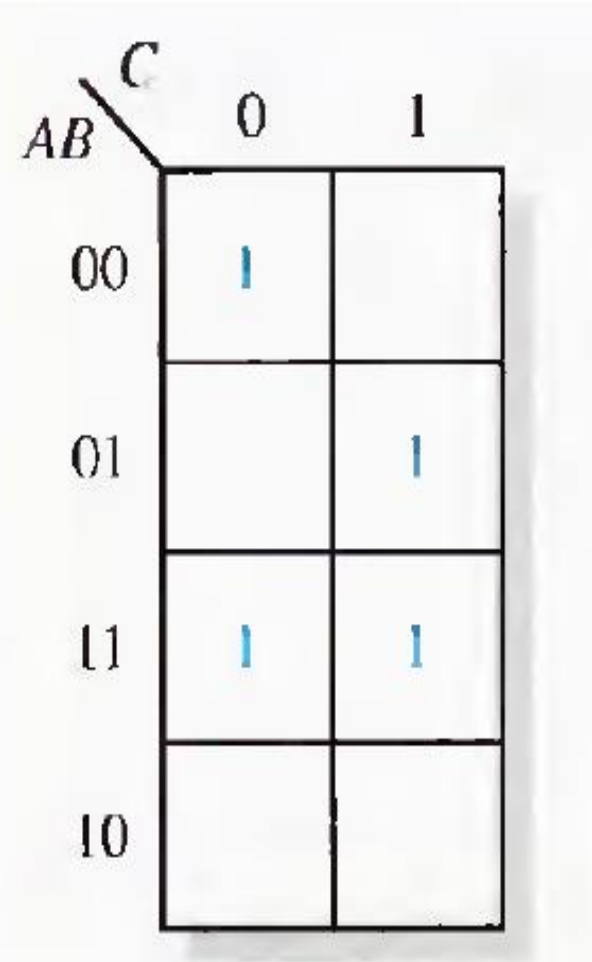
You can group Is on the Karnaugh map according to the following rules by enclosing those adjacent cells containing 1s.

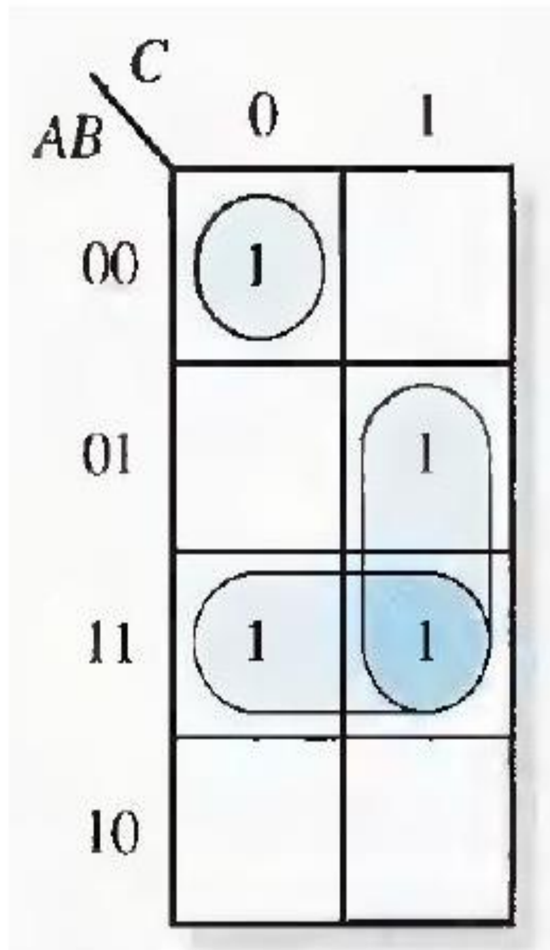
The goal is to maximize the size of the groups and to minimize the number of groups.

## Steps for Grouping

- A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells is the maximum group.
- Each cell in a group must be adjacent to one or more cells in that same group. but all cells in the group do not have to be adjacent to each other.
- Always include the largest possible number of 1's in a group in accordance with rule 1.
- Each 1 on the map must be included in at least one group. The Is already in a group can be included in another group as long as the overlapping groups include noncommon 1's.

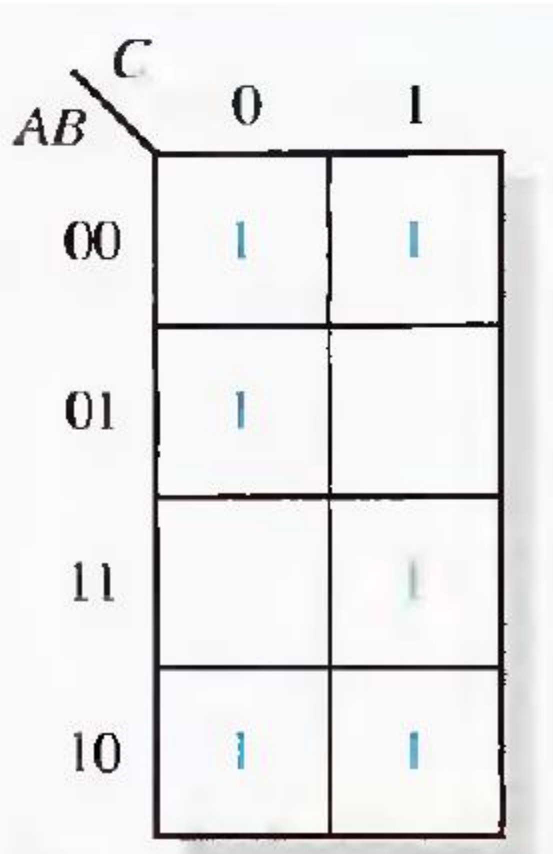
# For Example

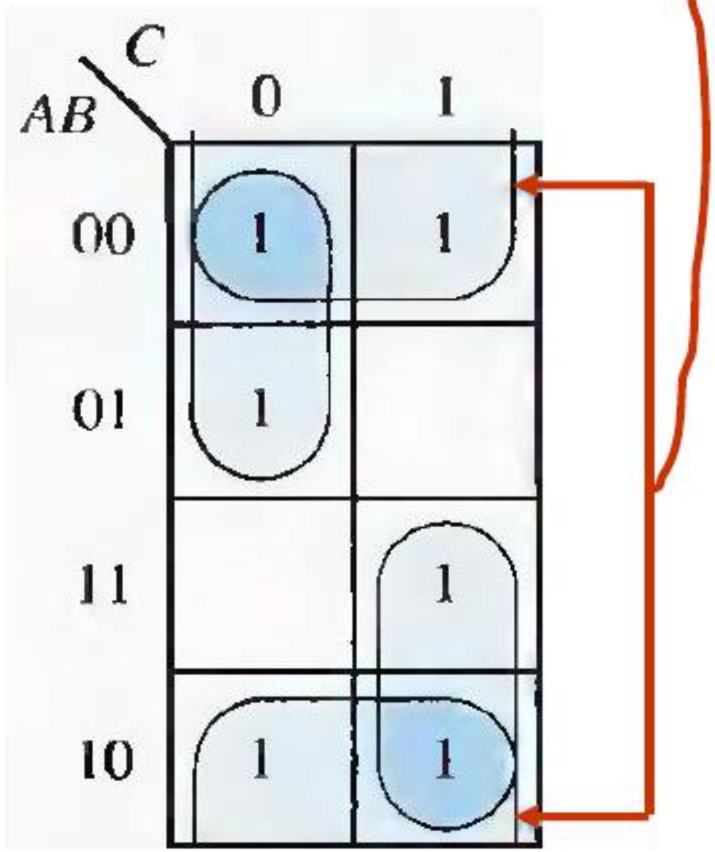


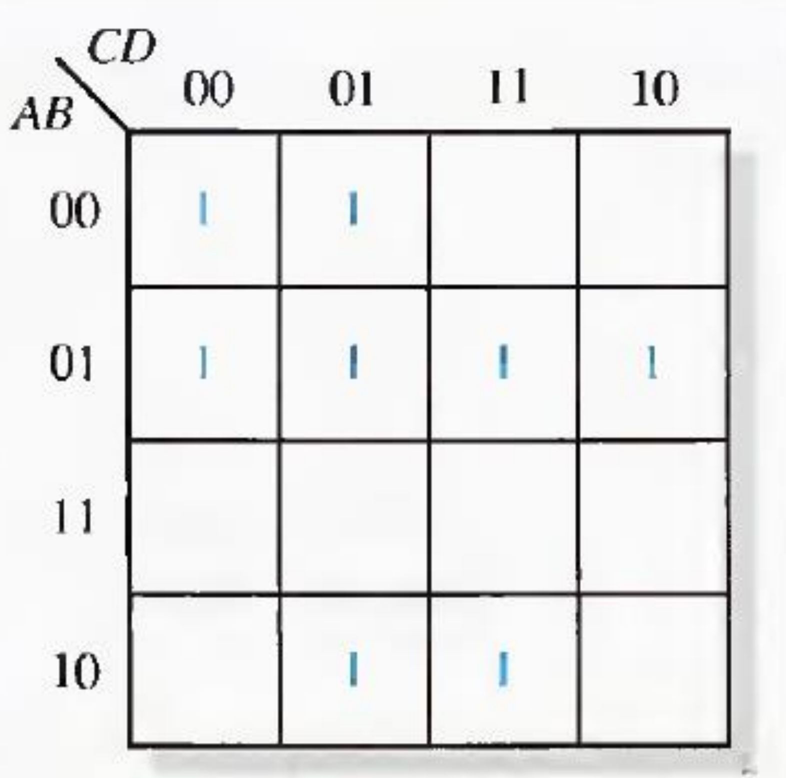
# Example

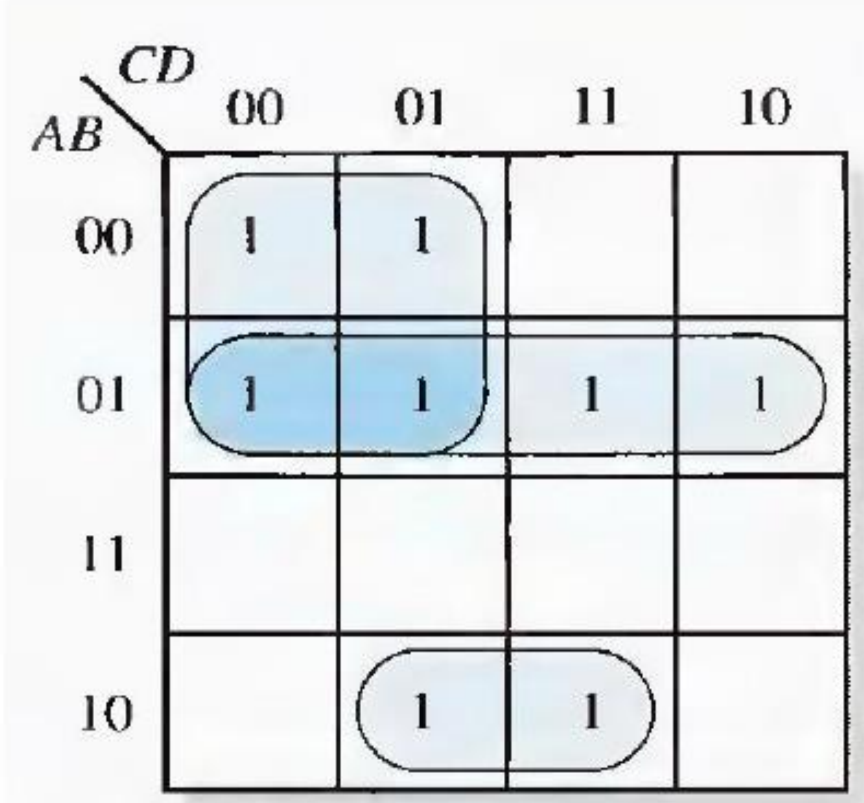
**Wrap around adjacency**



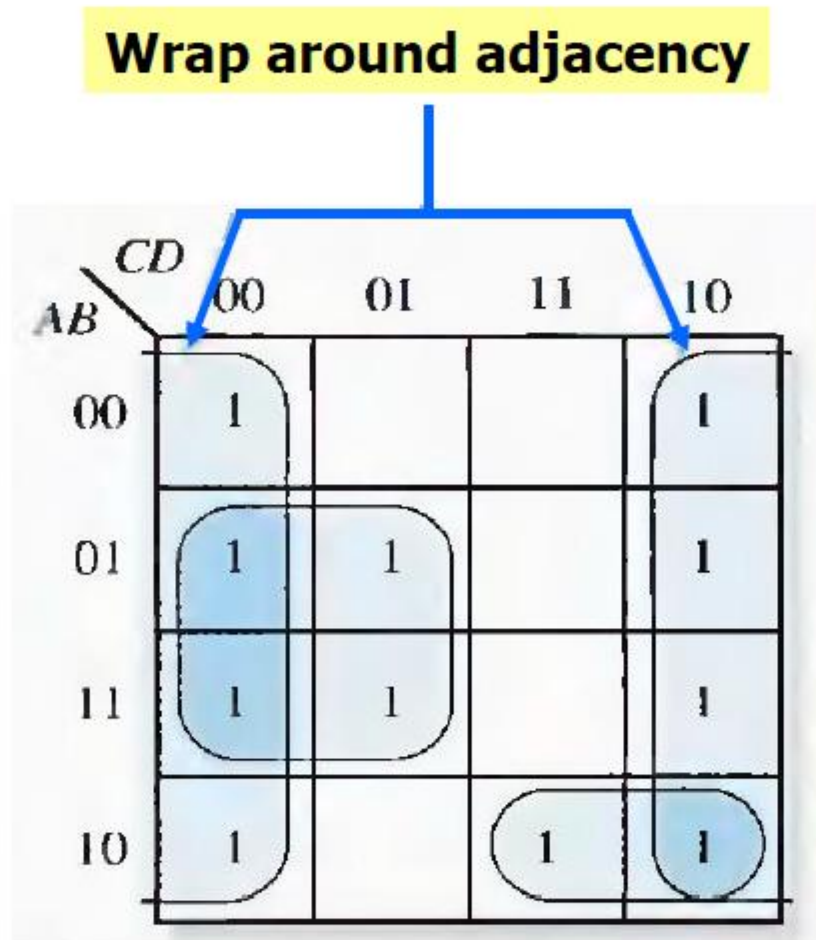
# K-Maps


CD
AB
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1


CD
AB
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1

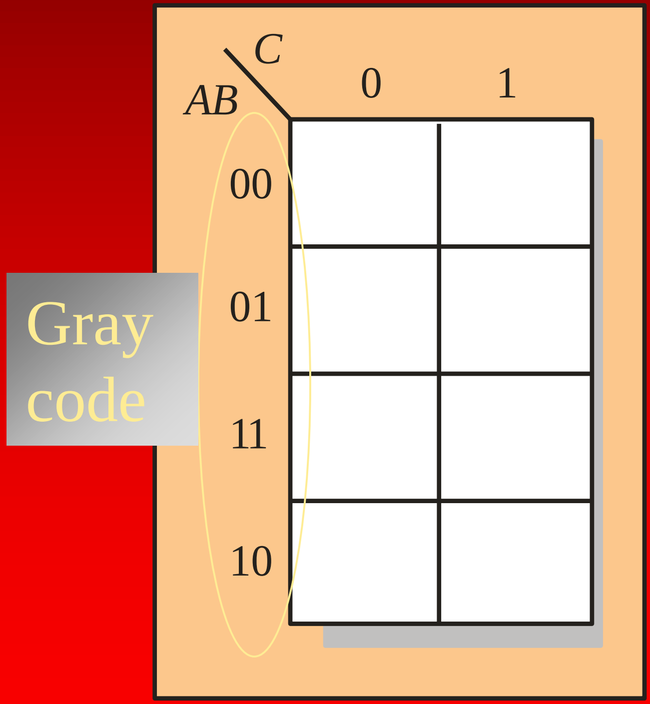
# K-Maps

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | | | 1 |
| 01 | 1 | 1 | | 1 |
| 11 | 1 | 1 | | 1 |
| 10 | 1 | | 1 | 1 |



## Karnaugh maps

Cells are usually labeled using 0's and 1's to represent the variable and its complement.




The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

Ones are read as the true variable and zeros are read as the complemented variable.

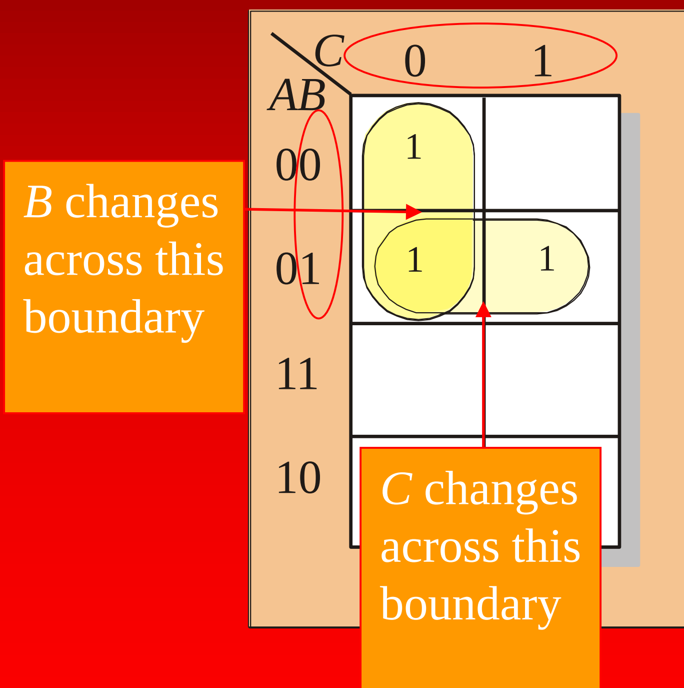
## Determining the Minimum SOP Expression from the Map.

- When all the 1's representing the standard product terms in an expression are properly mapped and grouped, the process of determining the resulting minimum SOP expression begins.
- Rules
  - Group the cells that have 1's. Each group of cells containing I s creates one product term composed of all variables that occur in either form within the group.
  - Variables that occur both uncomplemented and complemented within the group are eliminated. These are called contradictory variables.

## Karnaugh maps

K-maps can simplify combinational logic by grouping cells and eliminating variables that change.

**Example** Group the 1's on the map and read the minimum logic.



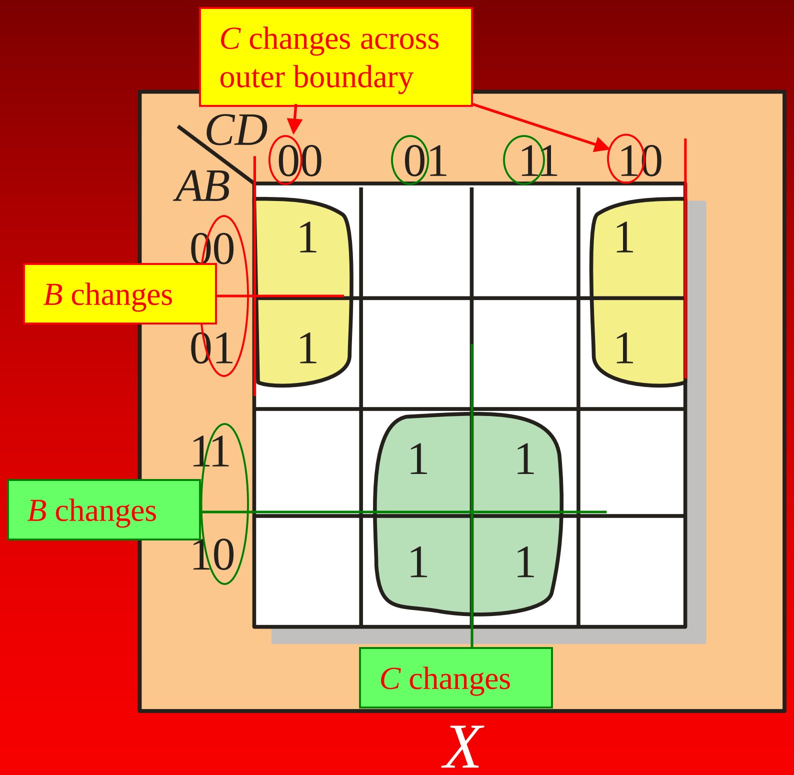

**Solution**

1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\overline{A}\overline{C}$.
4. The horizontal group is read $\overline{A}B$.

$$X = \overline{A}\overline{C} + AB$$

# Karnaugh maps

**Example** Group the 1's on the map and read the minimum logic.




**Solution**

1. Group the 1's into two separate groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The upper (yellow) group is read as $\overline{A}\overline{D}$.
4. The lower (green) group is read as $AD$.

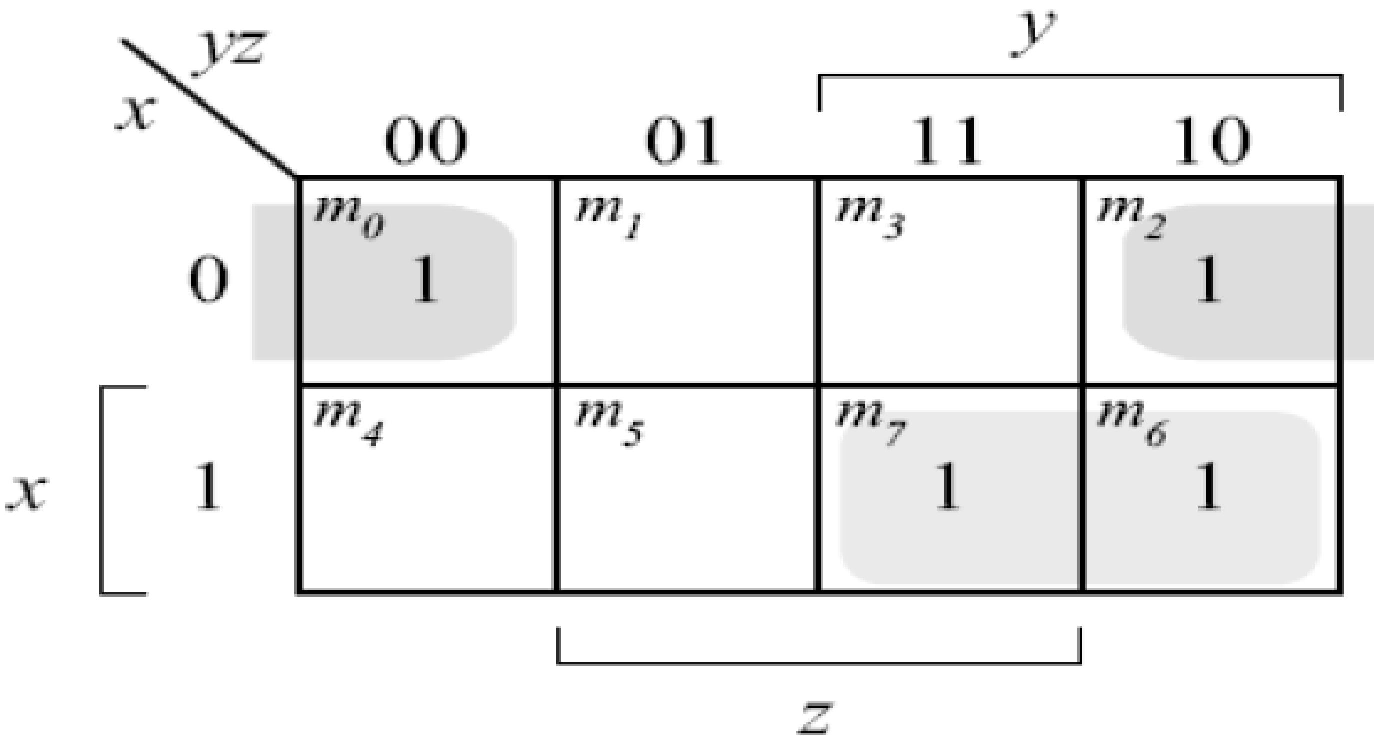
$$X = \overline{A}\overline{D} + AD$$

# Related Example

- Minimize the following SOP expression,
- $F(x, y, z) = \Sigma(0,2,6,7)$
- $F(x, y, z) = \Sigma(0,2,3,4,6)$
- $F(x, y, z) = \Sigma(0,1,2,3,7)$
- $F(x, y, z) = \Sigma(3,5,6,7)$
- $F(x, y, z) = \Sigma(0,1,5,7)$
- $F(x, y, z) = \Sigma(0,1,6,7)$
- $F(x, y, z) = \Sigma(1,2,3,6,7)$

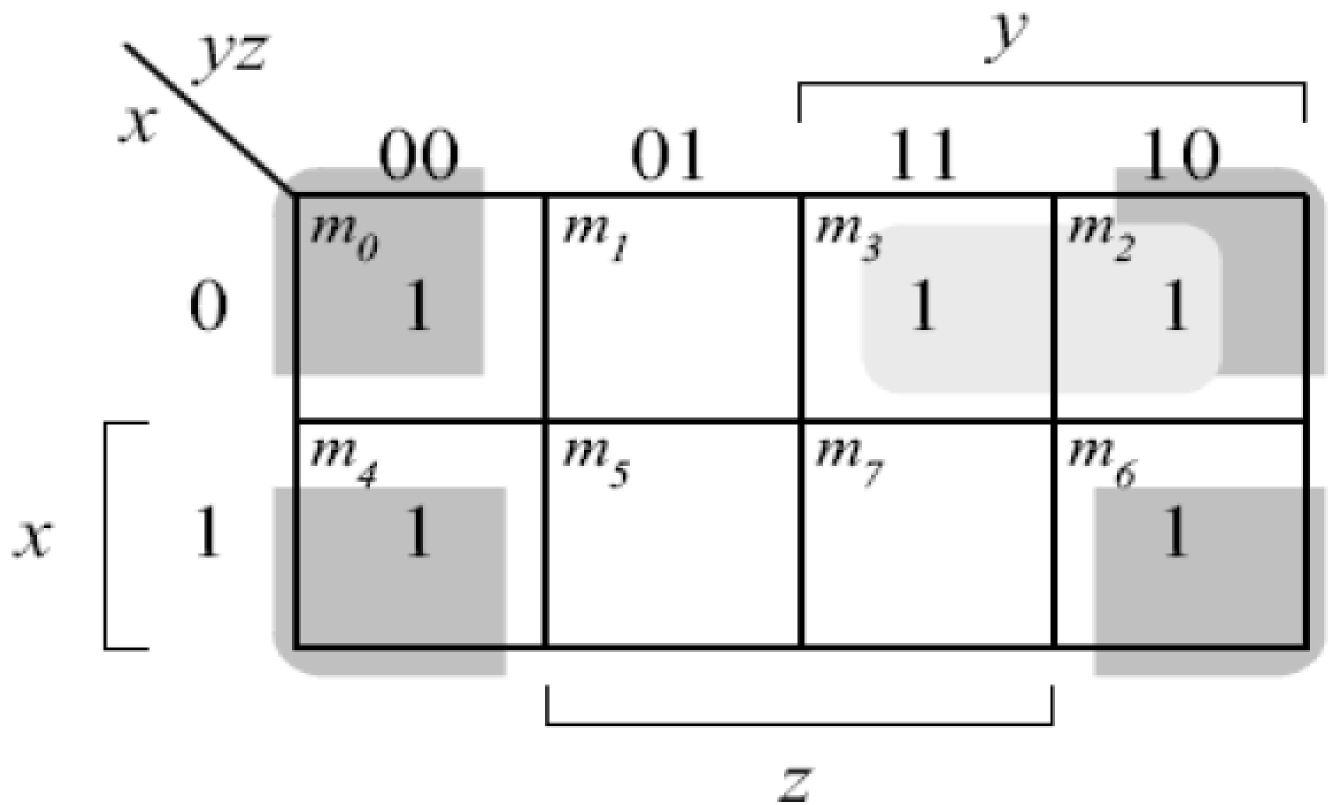
# SOLUTION(1-7)

$F(x, y, z) = \Sigma(0,2,6,7)$ ➔ 1



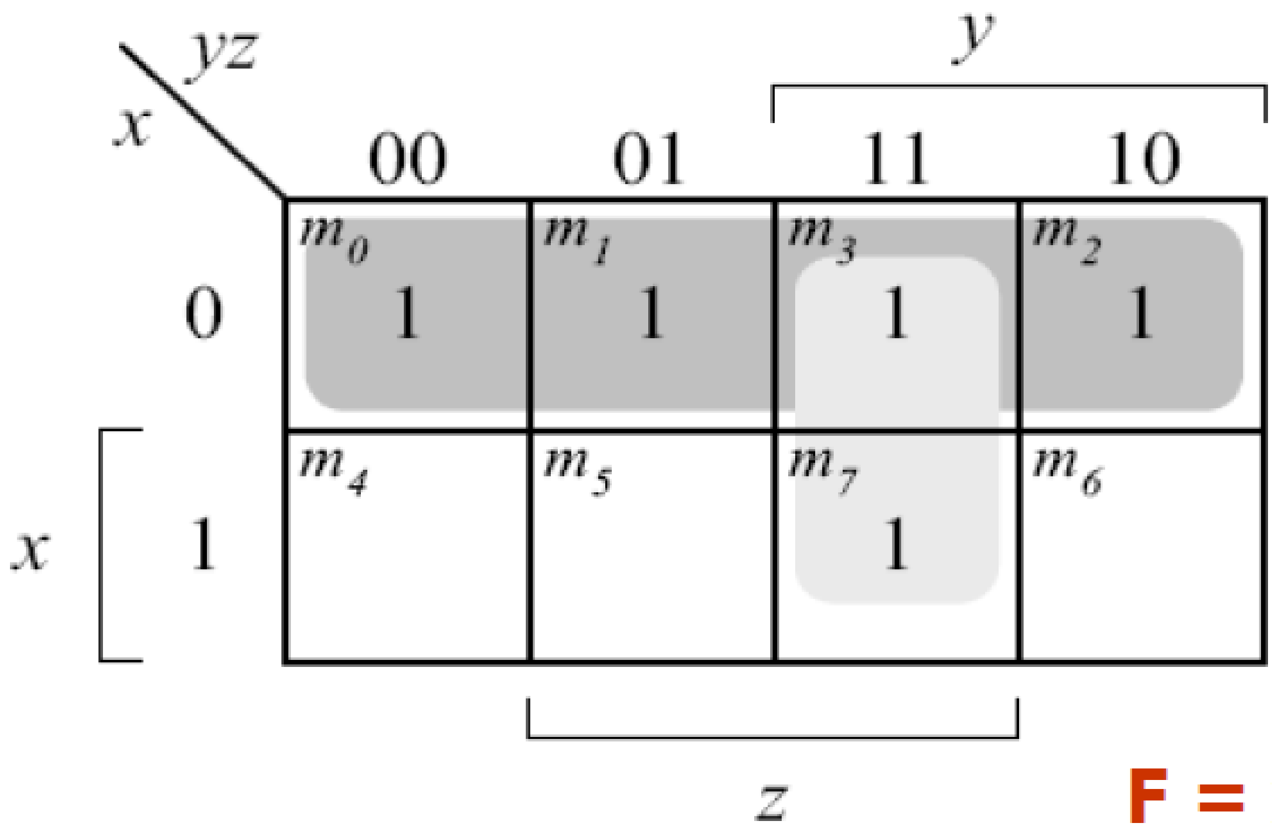

**$F = x\,y + x'\,z'$**

$F(x, y, z) = \Sigma(0,2,3,4,6)$ → 2



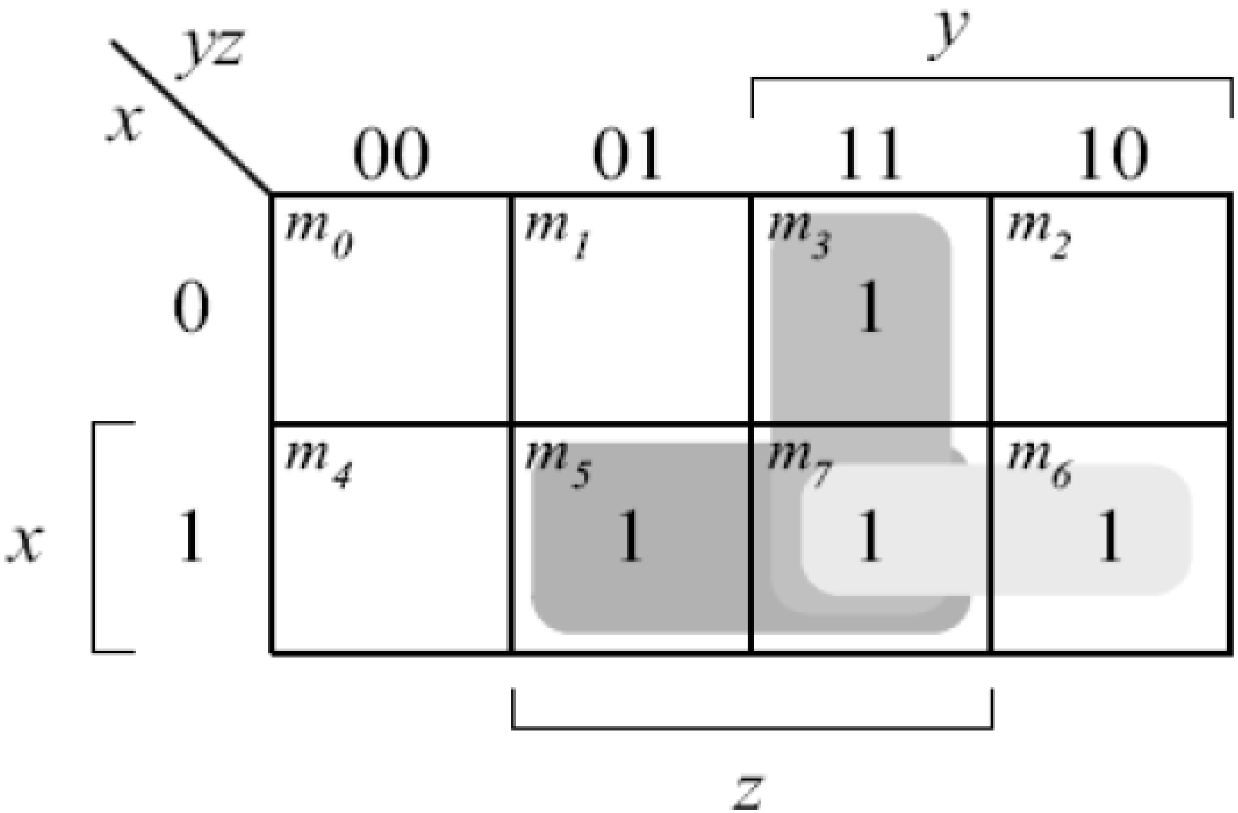

**$F = z' + x'y$**

$F(x, y, z) = \Sigma(0,1,2,3,7)$ → 3



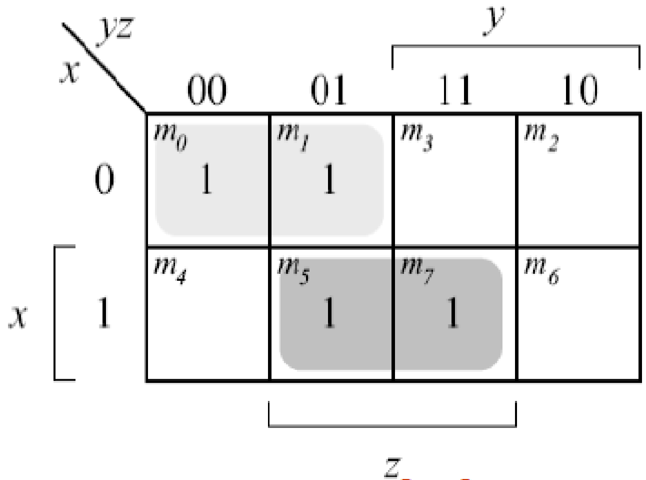

**F = x' + y z**

$F(x, y, z) = \Sigma(3,5,6,7)$ ➔ 4

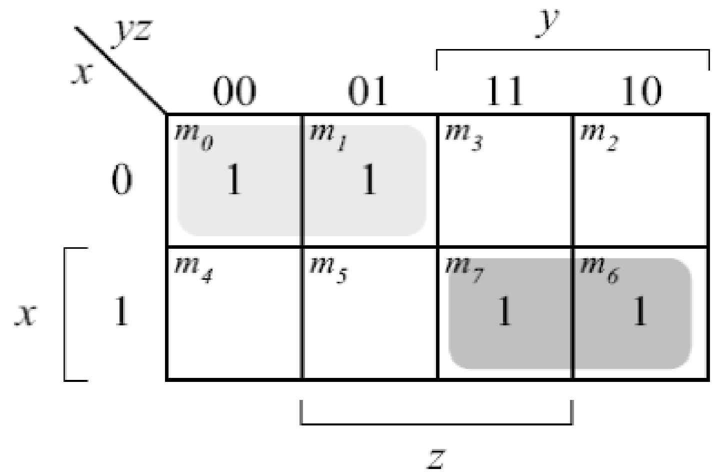



**$F = xy + xz + yz$**

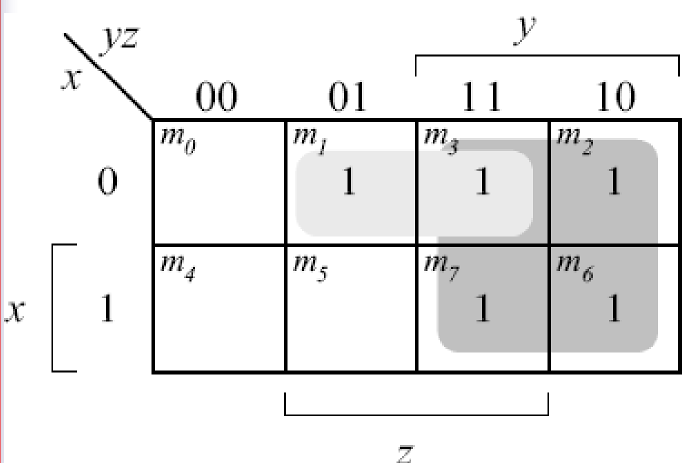
# Solution of 5, 6, 7




**5. F = x\` y' + x z**




**6. F = x\` y' + x y**

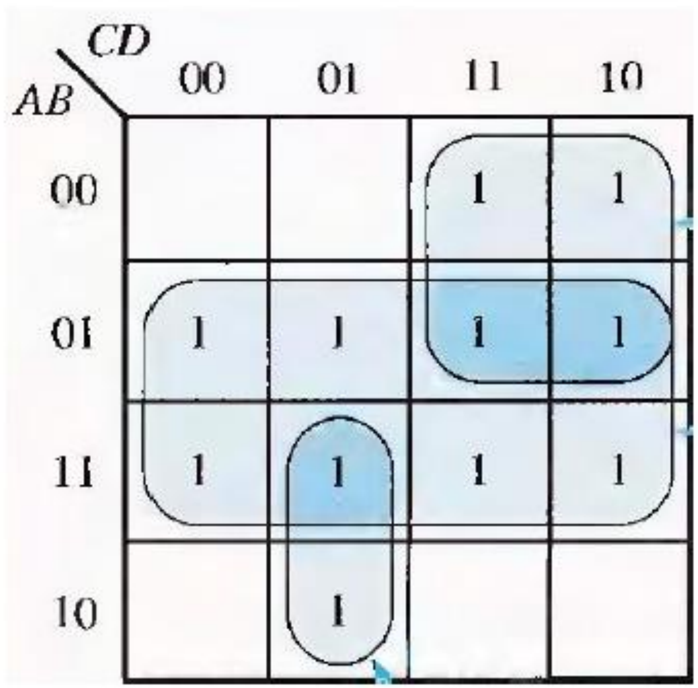



**7. F = y + x\` z**

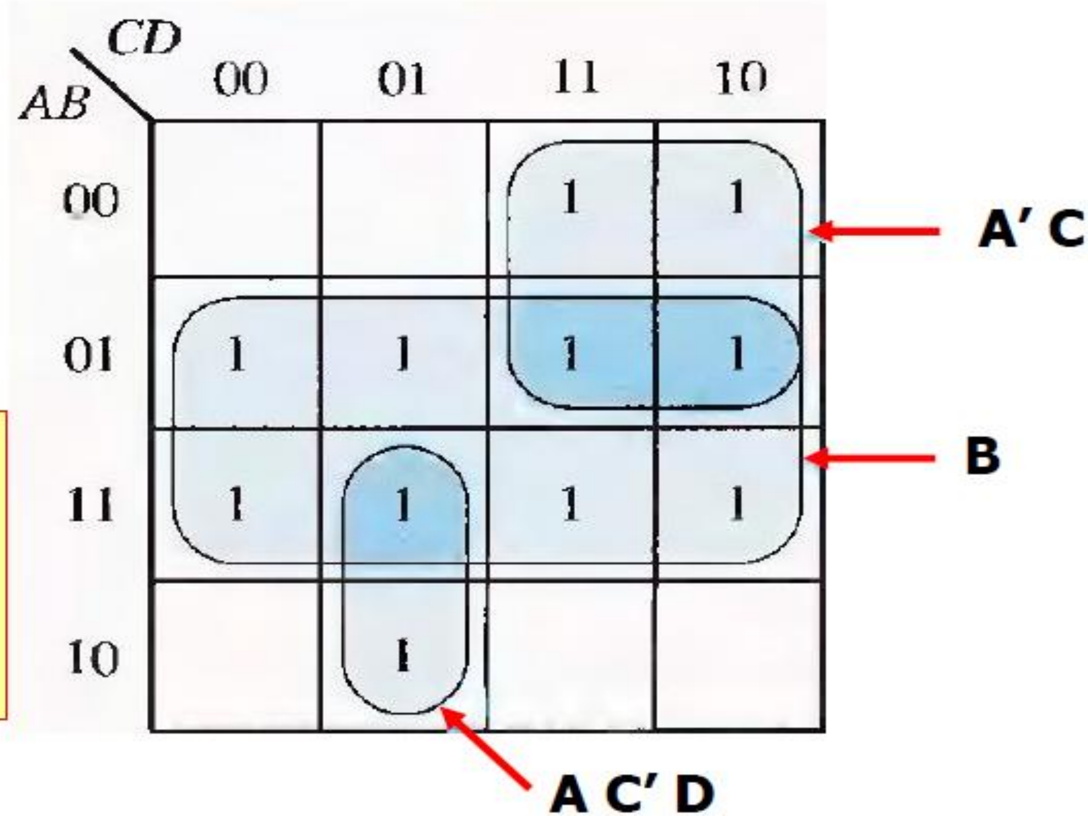
# K-Maps

## Example

- Determine the product terms for the Karnaugh map given and write the resulting minimum SOP expression?



# Solution




The resulting minimum SOP expression is,

**B + A′ C + AC′ D**

the sum of these product terms:

# Self Assessment

**Problem:** For the Karnaugh map on the previous slide, add a 1 in the lower right cell (1010) and determine the resulting SOP expression.

## Example 1

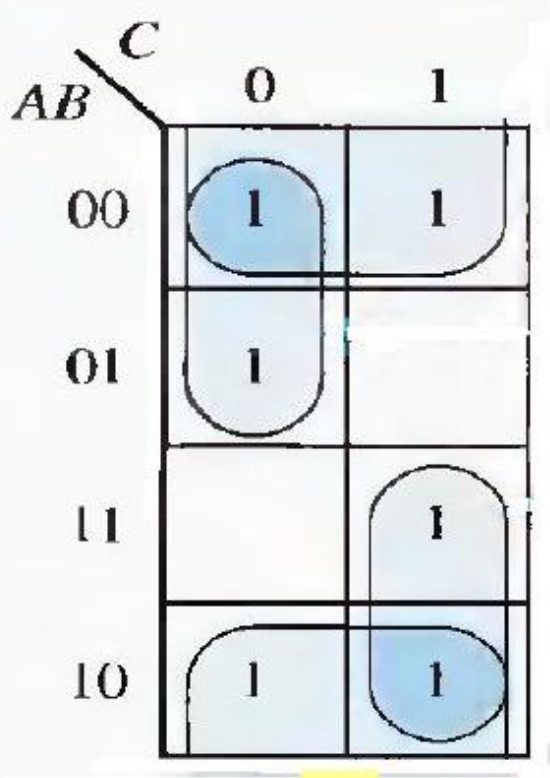
| AB \ C | 0 | 1 |
|---|---|---|
| 00 | 1 | |
| 01 | | 1 |
| 11 | 1 | 1 |
| 10 | | |

**a**

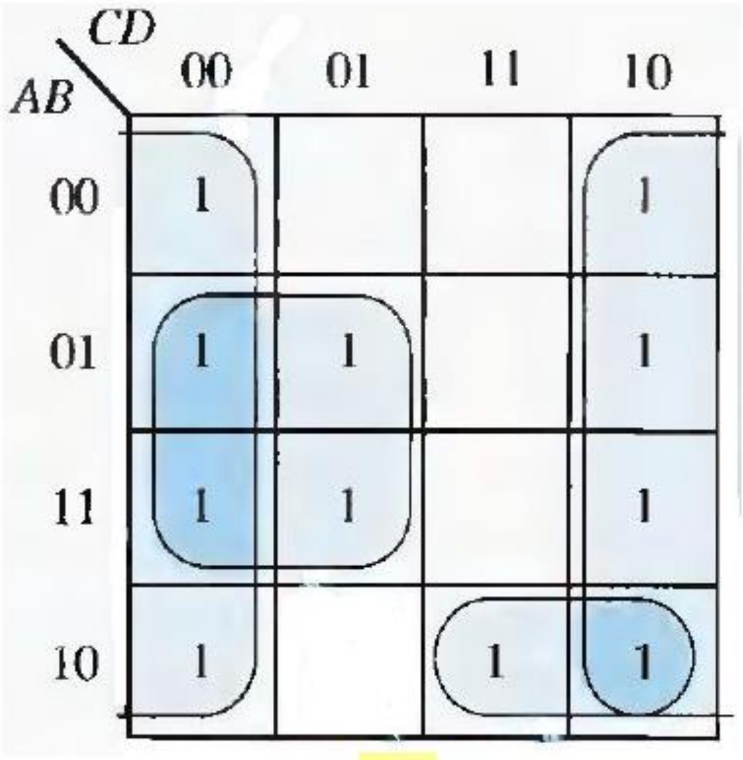
| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 1 | | |
| 01 | 1 | 1 | 1 | 1 |
| 11 | | | | |
| 10 | | 1 | 1 | |

**b**

# Example 2


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

c


CD
AB
00
01
11
10
00
01
11
10
1
1
1
1
1
1
1
1
1
1
1

d

## Solution

The resultant SOP expression against a, b, c and d are,

a. $AB+BC+A'B'C'$

b. $A'B + A'C' + AB'D$

c. $B' + A'C' + AC$

d. $D' + AB'C + BC'$

# Class Assignment

Using k-map simplify the following expression,

$F(A,B,C,D) = \Sigma(0,2,3,4,6,8,10,11,12,14)$

**Just do it in 3min. Only.**

**The resulting minimum SOP expression is**

**$D' + B'C$**

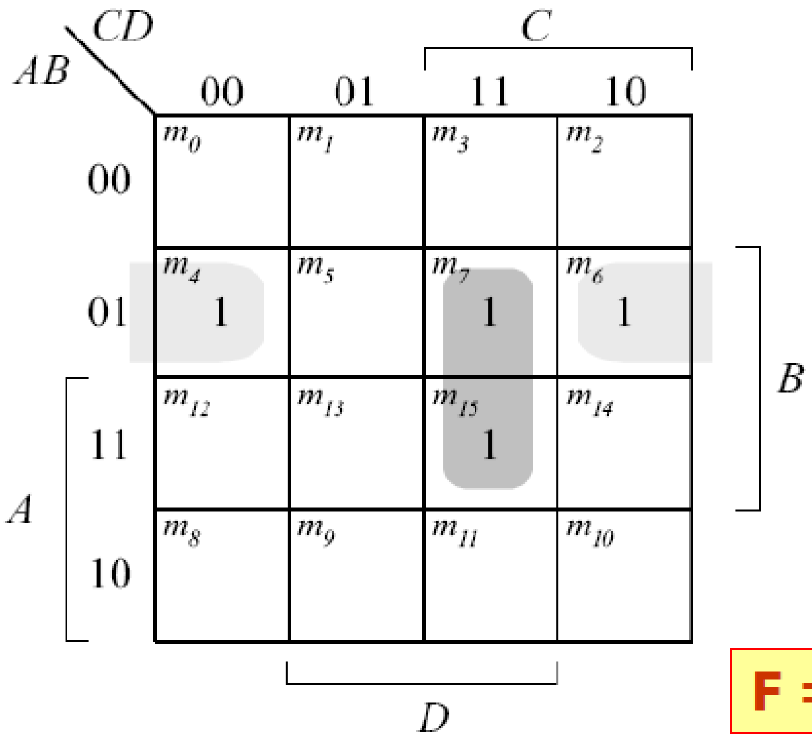
# Examples

Find the minimized SOP expression against the following,

1. $F(A,B,C,D) = \Sigma(4,6,7,15)$
2. $F(A,B,C,D) = \Sigma(3,7,11,13,14,15)$
3. $F(A,B,C,D) = \Sigma(0,1,5,8,9)$
4. $F(A,B,C,D) = \Sigma(1,4,5,6,12,14,15)$

## Solution

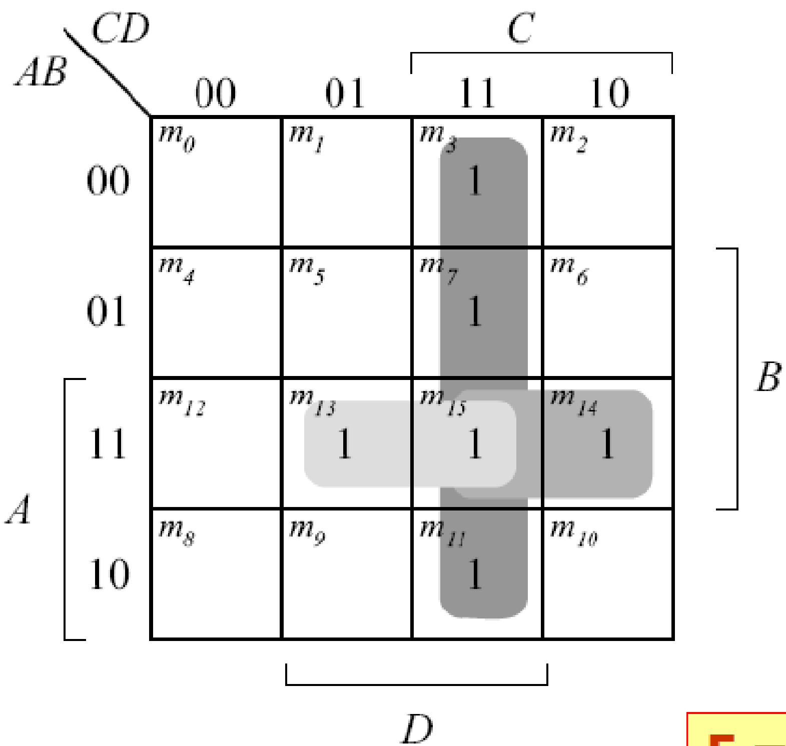
**1. F (A,B,C,D) = Σ(4,6,7,15)**




**F = BCD + A' BD'**

# Solution

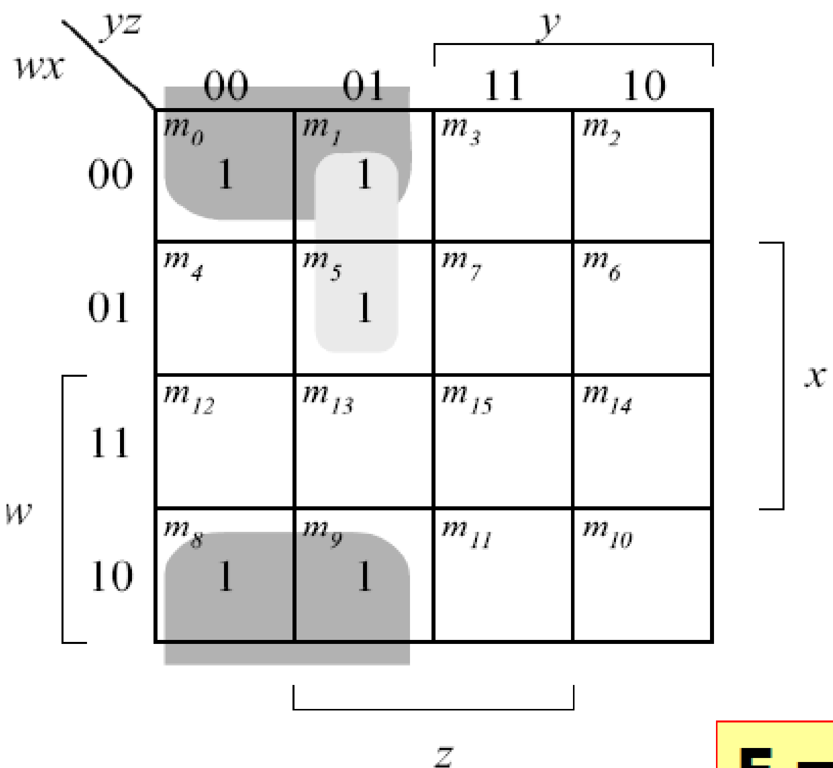
2. F (A,B,C,D) = $\Sigma(3,7,11,13,14,15)$




**F =CD + ABD + ABC**

# Solution

3. F (A,B,C,D) = Σ(0,1,5,8,9)

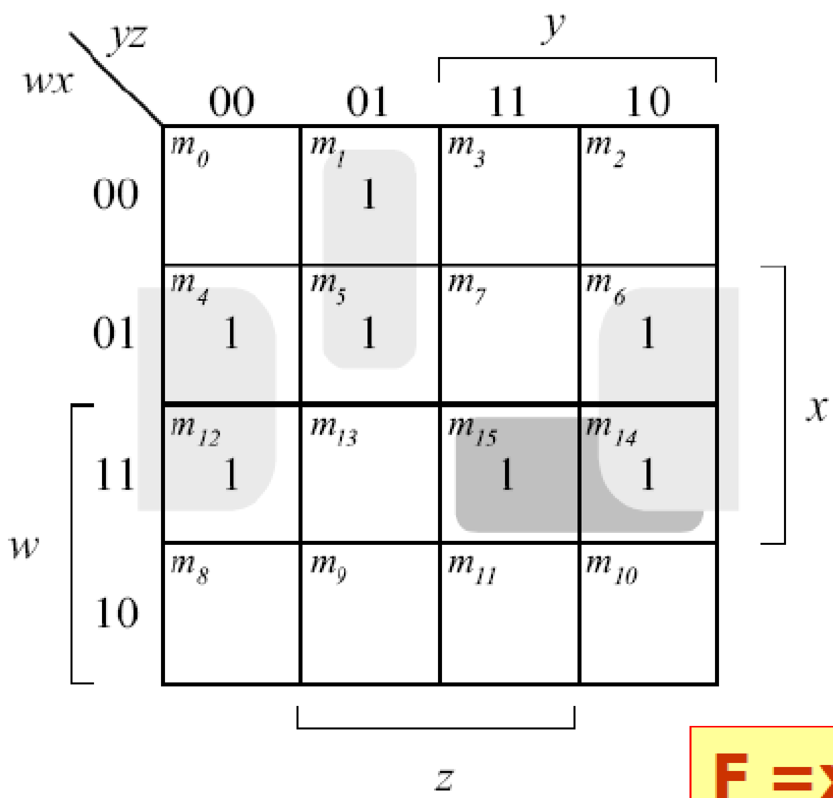



**F = x'y' + w'y'z**

# Solutions

## Solution

4. F (A,B,C,D) = Σ(1,4,5,6,12,14,15)




**F =xz' + w'y'z+ wxy**

# Don’t care conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code.

- There are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111.
- Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output.
- The "don't care" terms can be used to advantage on the Karnaugh map.

## Cont…

For each "don't care" term, an X is placed in the cell.

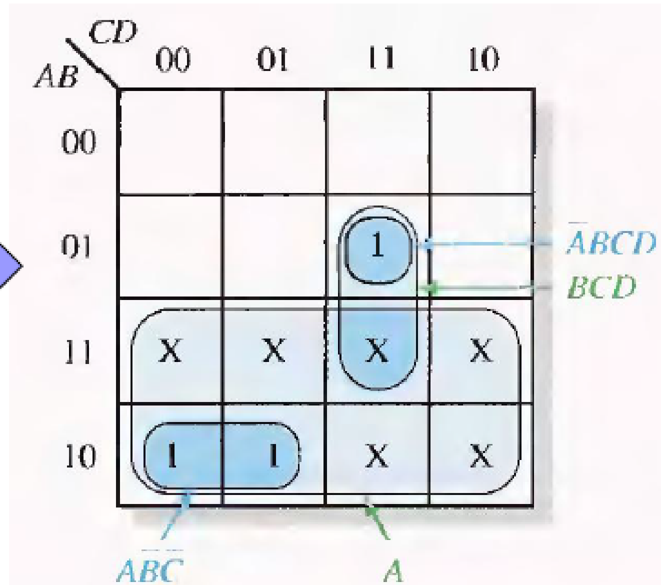
- When grouping the 1's, the X's can be treated as I s to make a larger grouping or as O's if they cannot be used to advantage.
- The larger a group, the simpler the resulting term will be.

| Inputs A B C D | Output Y |
|---|---|
| 0 0 0 0 | 0 |
| 0 0 0 1 | 0 |
| 0 0 1 0 | 0 |
| 0 0 1 1 | 0 |
| 0 1 0 0 | 0 |
| 0 1 0 1 | 0 |
| 0 1 1 0 | 0 |
| 0 1 1 1 | 1 |
| 1 0 0 0 | 1 |
| 1 0 0 1 | 1 |
| 1 0 1 0 | X |
| 1 0 1 1 | X |
| 1 1 0 0 | X |
| 1 1 0 1 | X |
| 1 1 1 0 | X |
| 1 1 1 1 | X |

# DON'T care conditions

The truth table in Figure below describes a logic function that has a I output only when the BCD code for 7,8, or 9 is present on the inputs. if the "don't cares" are used as Is, the resulting expression for the function is A + BCD.

| Inputs ABCD | Output Y |
|---|---|
| 0 0 0 0 | 0 |
| 0 0 0 1 | 0 |
| 0 0 1 0 | 0 |
| 0 0 1 1 | 0 |
| 0 1 0 0 | 0 |
| 0 1 0 1 | 0 |
| 0 1 1 0 | 0 |
| 0 1 1 1 | 1 |
| 1 0 0 0 | 1 |
| 1 0 0 1 | 1 |
| 1 0 1 0 | X |
| 1 0 1 1 | X |
| 1 1 0 0 | X |
| 1 1 0 1 | X |
| 1 1 1 0 | X |
| 1 1 1 1 | X |




Without "don't cares"
Y = AB'C' + A'BCD
With ""don't cares"
Y = A + BCD

# Related Problem

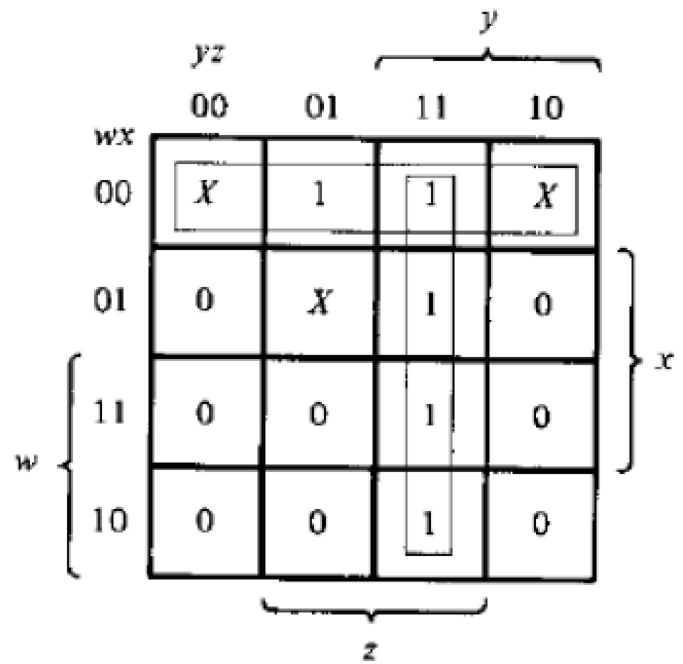
Simplify the Boolean functions

F(w,x,y,z)=Σ(1.3.7.11.15)

Don't care conditions are,

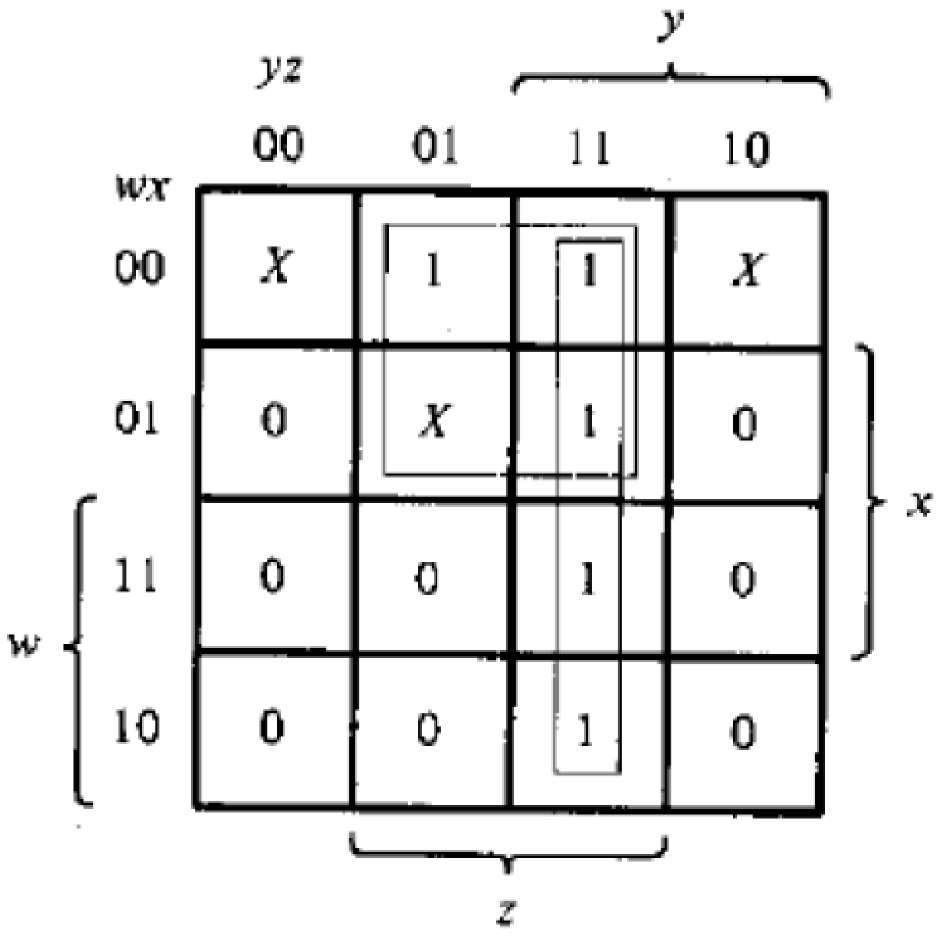
d(w,x,y,z)= Σ(0.2.5)

**Solution**




**F=yz+w'x'**

# Example




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

END OF LECTURE