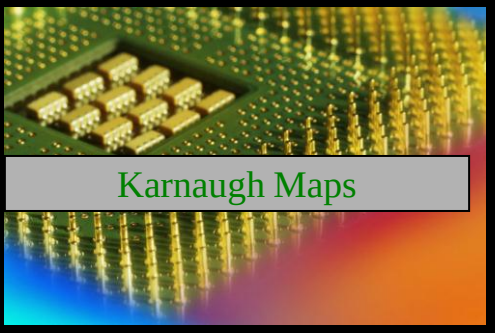
EEE251

Logic Design and Circuits

Assist.Prof. Yavuz BAYAM



# Karnaugh Maps

November 5, 2012

## Introduction

- Karnaugh Map (or K-map) minimization is a visual minimization technique
  - Is an application of adjacency
  - Procedure guarantees a minimal expression
  - Easy to use; fast
  - Problems include:
    - Applicable to limited number of variables (4 ~ 8)
    - Errors in translation from TT to K-map
    - Errors in reading final expression

## Karnaugh Map

| 1st Column | 2nd Column | |
|---|---|---|
| $m_0$ | $m_2$ | 1st Row |
| $m_1$ | $m_3$ | 2nd Row |

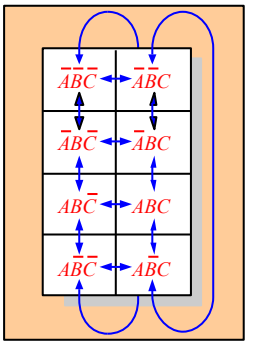
- Each cell contains one bit of output function
- Number of cells in table is $2^n$ (n=number of variable)
- In this case number of cell in the table for 2 variable $2^2=4$, for 3 variable $2^3=8$

## Karnaugh maps

The Karnaugh map (K-map) is a tool for simplifying combinational logic with 3 or 4 variables. For 3 variables, 8 cells are required ($2^3$).

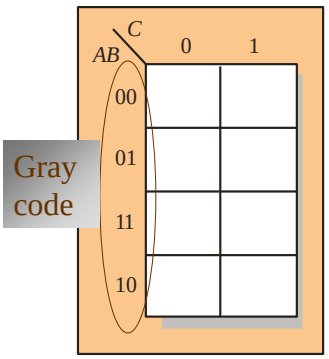
The map shown is for three variables labeled *A*, *B*, and *C*. Each cell represents one possible product term.

Each cell differs from an adjacent cell by only one variable.



## Karnaugh maps

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

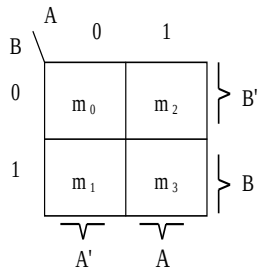



The numbers are entered in gray code, to force adjacent cells to be different by only one variable.

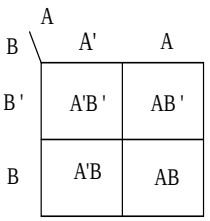
Ones are read as the true variable and zeros are read as the complemented variable.

## Karnaugh maps

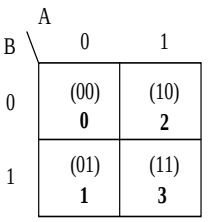
- 2 variable Karnaugh Maps have $2^2$=4 cells
  - Each cell contains one bit of output function




(a)

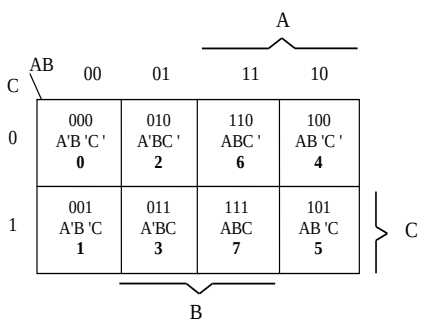


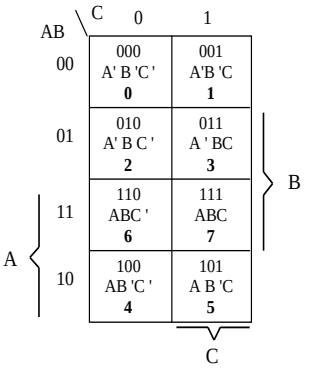

(b)




(c)

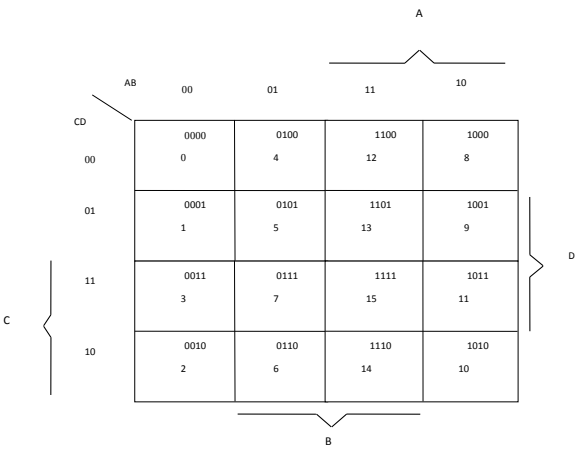
## Karnaugh maps




## Karnaugh maps

| A | B | C | D | Q |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | |
| 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 1 | |
| 1 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 1 | |



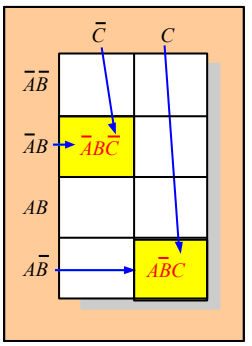
## Karnaugh maps

Alternatively, cells can be labeled with the variable letters. This makes it simple to read, but it takes more time preparing the map.

**Example** Read the terms for the yellow cells.

**Solution**

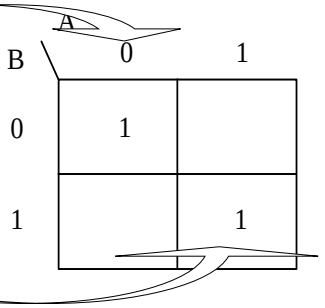
The cells are $\bar{A}B\bar{C}$ and $A\bar{B}C$.




## Formation of Karnaugh maps

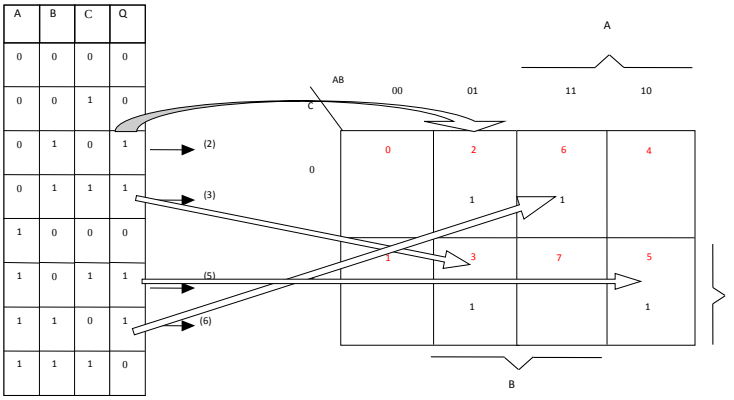
- Move output '1' from truth table into Karnaugh map
- Do not move zeros from truth table into Karnaugh map

| A | B | Q |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



## Formation of Karnaugh maps

| A | B | C | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |



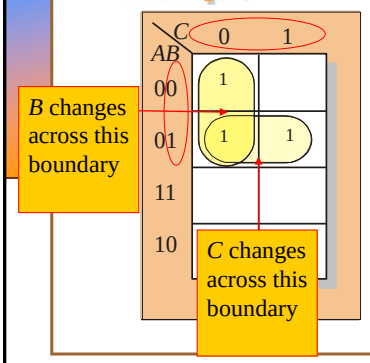

## Grouping

- Grouping occurs after converting truth table to Karnough Map
- If two cells have the same value and are next to each other, the terms are adjacent
- This adjacency is shown by enclosing them
- Procedures of grouping
  - **i-**Group size is a power of 2 and groups are rectangular ($2^0$=1, $2^1$=2, $2^2$=4, $2^3$=8, $2^4$=16,……).
  - **ii-** Different names are given to each groups
  - **iii-** Groups can have cells in common (overlap)
- This process helps to reduce the expression

## 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**

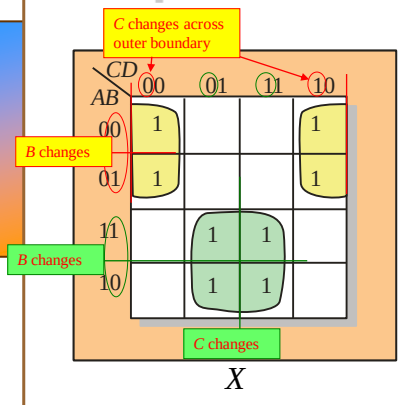
1. Group the 1's into two overlapping groups as indicated.
2. Read each group by eliminating any variable that changes across a boundary.
3. The vertical group is read $\bar{A}\bar{C}$.
4. The horizontal group is read $\bar{A}B$.

$$X = \bar{A}\bar{C} + \bar{A}B$$



## Karnaugh maps

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

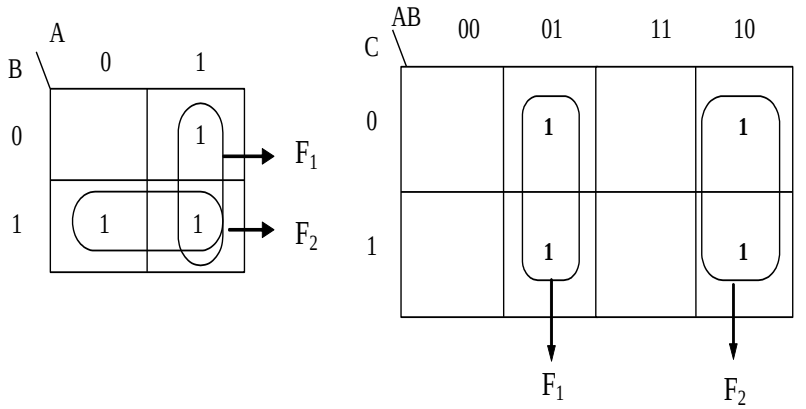



**Solution**

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 $\bar{A}\bar{D}$.
4. The lower (green) group is read as $AD$.

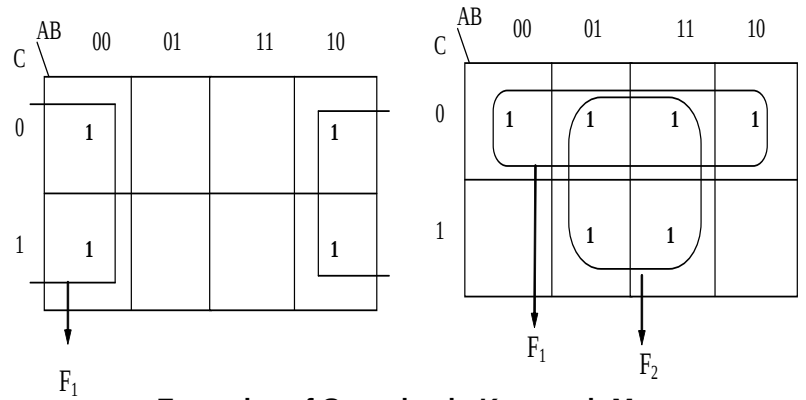
$$X = \bar{A}\bar{D} + AD$$

## Grouping



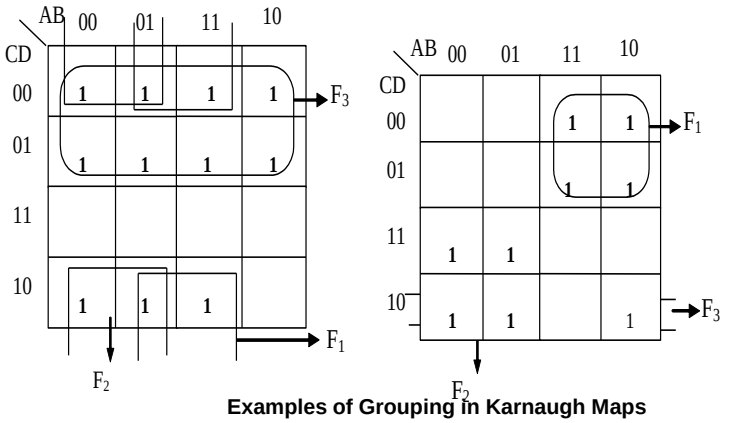

**Examples of Grouping in Karnaugh Maps**

## Grouping




**Examples of Grouping in Karnaugh Maps**

## Grouping




**Examples of Grouping in Karnaugh Maps**

## Grouping

**iv-** Adjacency also applies to the edges of the map. The left and right are adjacent as are the top and bottom

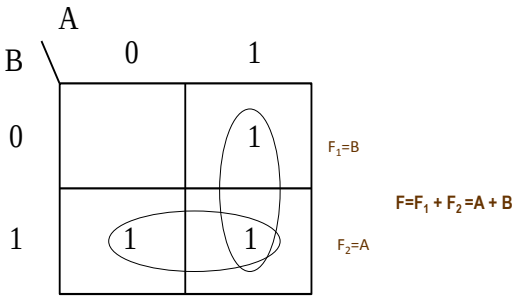
**v-** If there are four ones in 2 variable or 8 ones in 3 vaiable result of the function is 1

**vi-** Within group, note when variable values change as you go cell to cell. This determines how the term expression is formed according to the following table

| Variable changes | Exclude |
|---|---|
| Variable constant 0 | Inc. comp |
| Variable constant 1 | Inc. true |

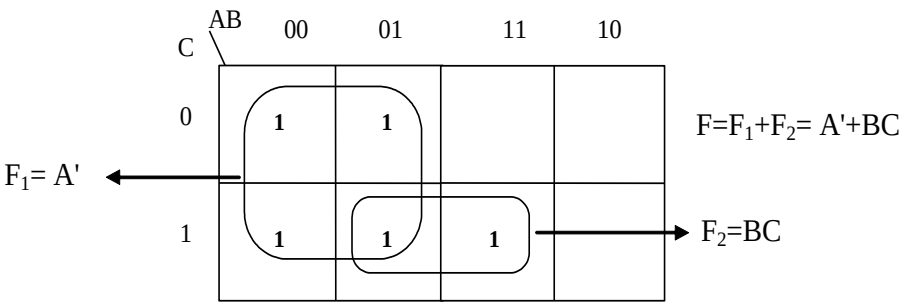
## Grouping

**Example :** Group the Karnough Map below

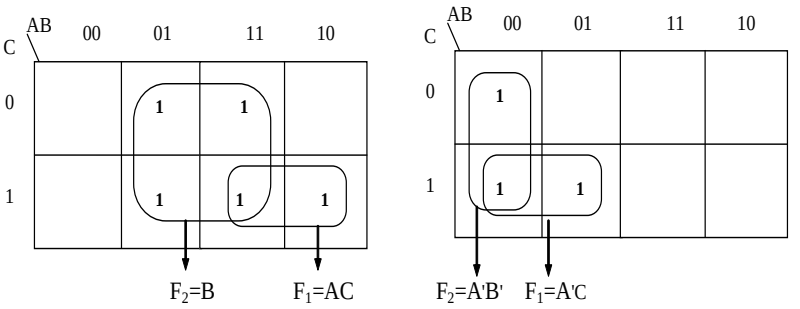



## Grouping

**Example:** Group the Karnaugh Map Below and write equations

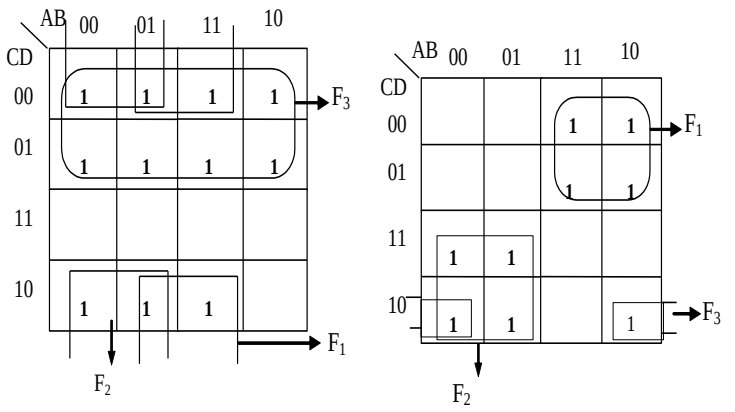


## Grouping

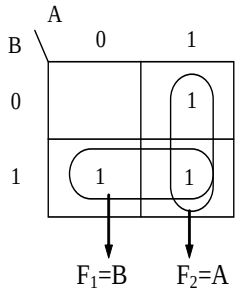


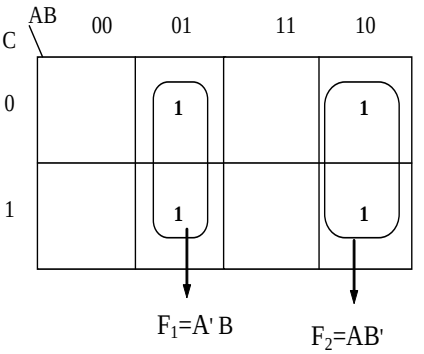

## Grouping

**Example:** Group the Karnough Maps below

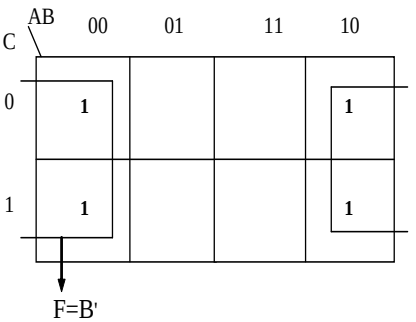


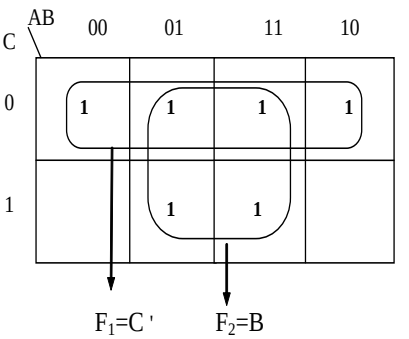
## Grouping



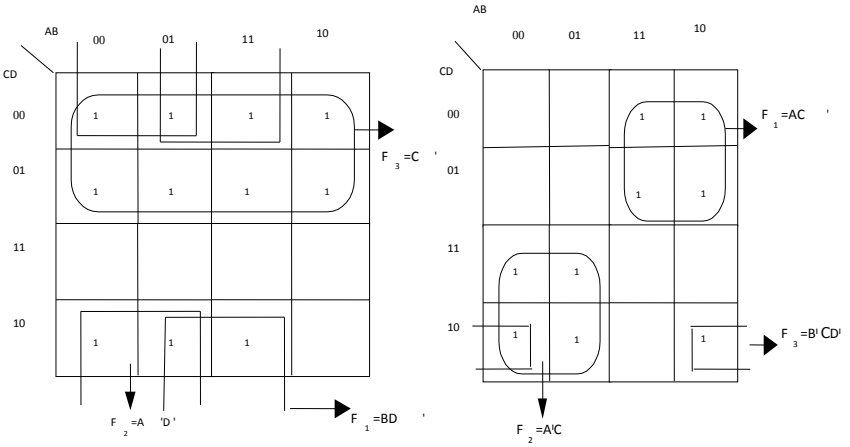




## Grouping



## Grouping

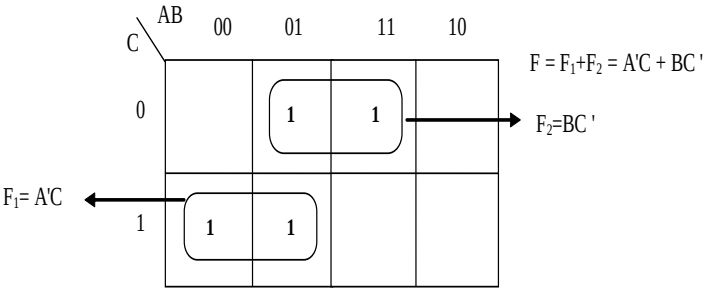



## Simplification of Boolean Equations by Karnaugh Maps

- Prepare a Karnaugh map according to number of inputs and put ‘1’ in the cell where there is a minterm in equation
- For example; for combination of A'BC put ‘1’ into 011 cell, and for AB'C' put ‘1’ into 100 cell
- Group after completion of moving process
- Write equations for groups and express them in one equation as minterms

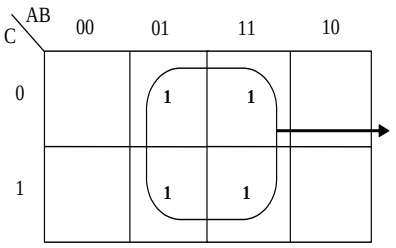
## Simplification of Boolean Equations by Karnaugh Maps

**Example:** Simplify F=A'BC+A'B'C+ABC'+A'BC'




## Simplification of Boolean Equations by Karnaugh Maps

- **Example:** Simplify F=ABC'+A'BC+A'BC'+ABC via Karnaugh map and check the result

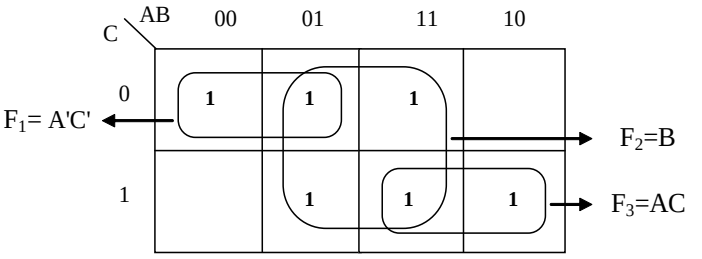



Proof:

F=ABC'+A'BC+A'BC'+ABC
=ABC'+A'BC'+AB(C+C')
1
=A'B(C+C')+AB=A'B+AB=B(A+A')
1 1
=B

## Simplification of Boolean Equations by Karnaugh Maps

**Example:** Simplify F=A'B'C'+A'BC'+ABC'+A'BC+ABC+AB'C via Karnaugh map

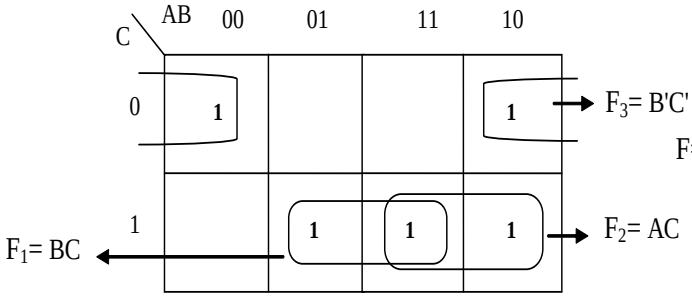



$F=F_1+F_2+F_3$

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

## Simplification of Boolean Equations by Karnaugh Maps

**Example:** Simplify F = A'B'C'+AB'C'+A'BC+AB'C+ABC via Karnaugh map

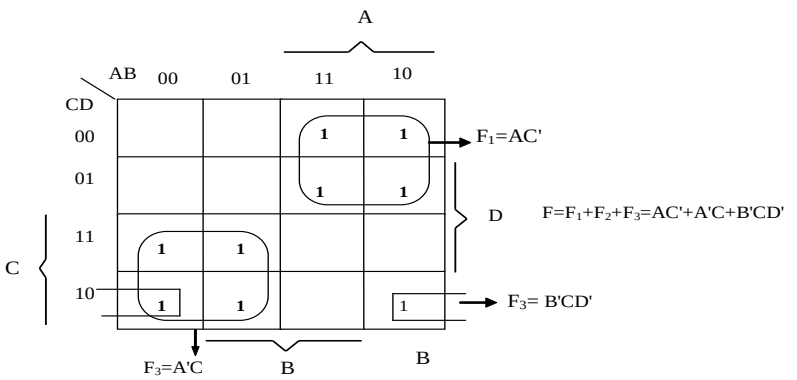



$F= BC+AC+B'C'$

## Simplification of Boolean Equations by Karnaugh Maps

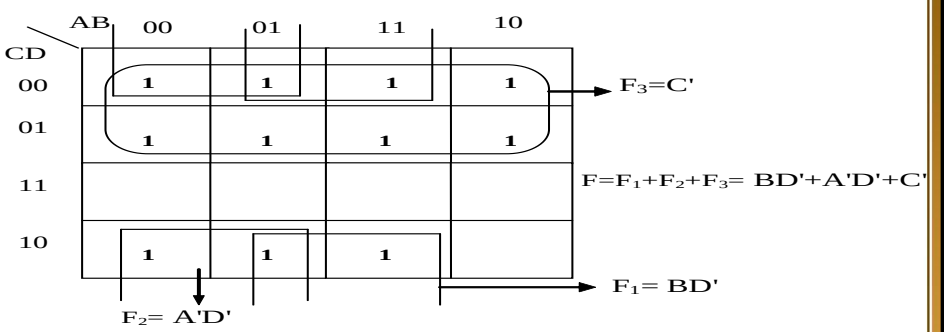
**Example:** Simplify Boolean equation below via Karnaugh map

F=ABC'D'+AB'C'D'+ABC'D+AB'C'D+A'B'CD+A'BCD+A'B'CD'+A'BCD'+AB'CD'



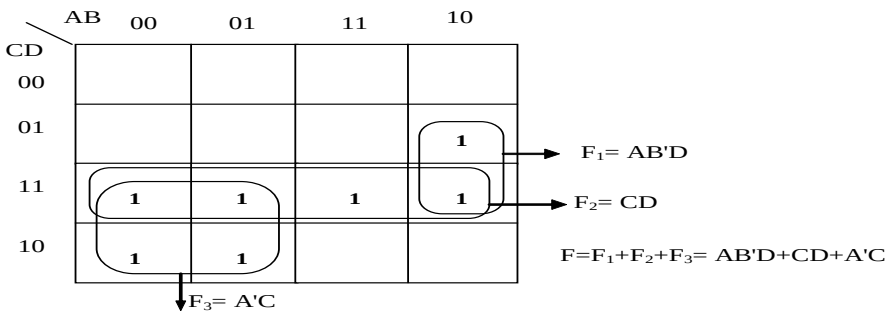

## Simplification of Boolean Equations by Karnaugh Maps

**Example:** Simplify minterm expression F=∑(0,1,2,4,5,6,8,9,12,13,14) via Karnaugh map



## Simplification of Boolean Equations by Karnaugh Maps

**Example:** Simplify F = A'B'CD+ABCD+A'CD'+A'CD+AB'D via Karnaugh Maps



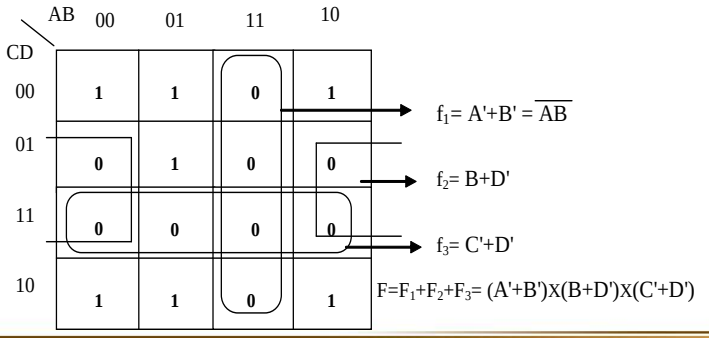

## POS Minimization

- For POS ’0’s are carried to the karnaugh map.
- 
- If the equation is in SOP form, first 1s are placed in the karnaugh map then rest will be filled with ‘0’s.
- 
- 0s in cells are grouped and POS form is written.

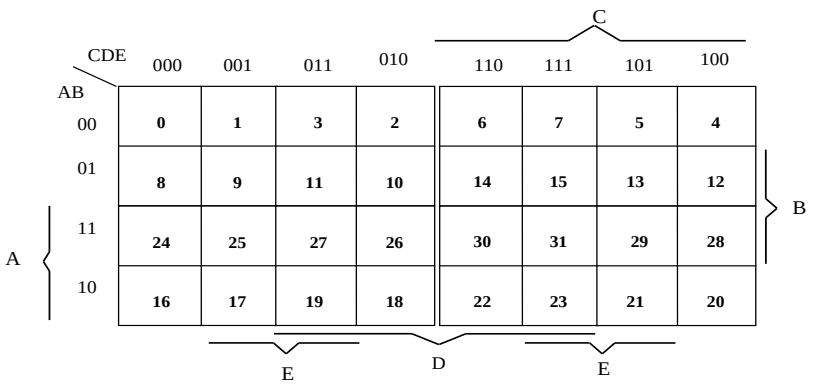
## POS Minimization

**Example:** Minimize SOP function by karnaugh map using 0s F(A,B,C,D) = (0, 2, 4, 5, 6, 8, 10).

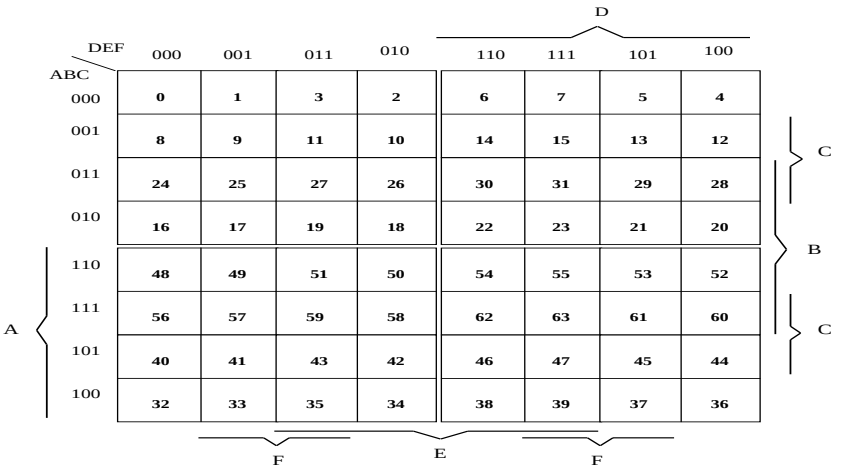
- put '1's in Karnaugh map.
- Fill rest of the cells with '0' and group.
- Write maxterms (POS)
- For POS if constant variable is '1' it's complement, if it is '0' then itself is written.




## Karnaugh Maps with 5 Variable



## Karnaugh Maps with 6 Variable




## Don't Care

- 1s and 0s in Karnaugh maps are meaningful for a logic function
- However, there might be situations that input value may noy not have certain value.
- For example in a decimal system expressed with 4 bit, numbers after 9 would never exist.
- In this case those combination values are ignored.
- They are called **'don't care situations'** and helpful for simplification.

## Don’t Care

- It’s not possible to put 1 or 0 in don’t care cells
- So those cells are filled with Xs or ds to differentiate.
- When groupin they can be assumed either 1 or zero.
- Decision is made based on the usefullness.

**Example:** simplify the function F = (1,3,7,11,15) with don’t cares d = (0,2,5).