Gandhinagar Institute Of Technology

Subject – Digital Electronics (2140910)

Branch – Electrical

Topic – K - Map

Name	Enrollment No.
Abhishek Chokshi	140120109005
Himal Desai	140120109008
Harsh Dedakia	140120109012

Guided By – Prof. Gunjan Sir

The Karnaugh Map

Karnaugh Maps

- Karnaugh maps (K-maps) are *graphical* representations of Boolean functions.
- One map cell corresponds to a row in the truth table.
- Also, one map cell corresponds to a minterm or a maxterm in the Boolean expression
- Multiple-cell areas of the map correspond to standard terms.
- A K-map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the minimum expression.

What is K-Map

- It's similar to truth table; instead of being organized (i/p and o/p) into columns and rows, the K-map is an array of cells in which each cell represents a binary value of the input variables.
- The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells.
- K-maps can be used for expressions with 2, 3, 4, and 5 variables.

Two-Variable Map

$x_1 \backslash x_2$	0	1
0	0 m_0	1 m_1
1	2 m_2	3 m_3

OR

$x_2 \backslash x_1$	0	1
0	0 m_0	2 m_2
1	1 m_1	3 m_3

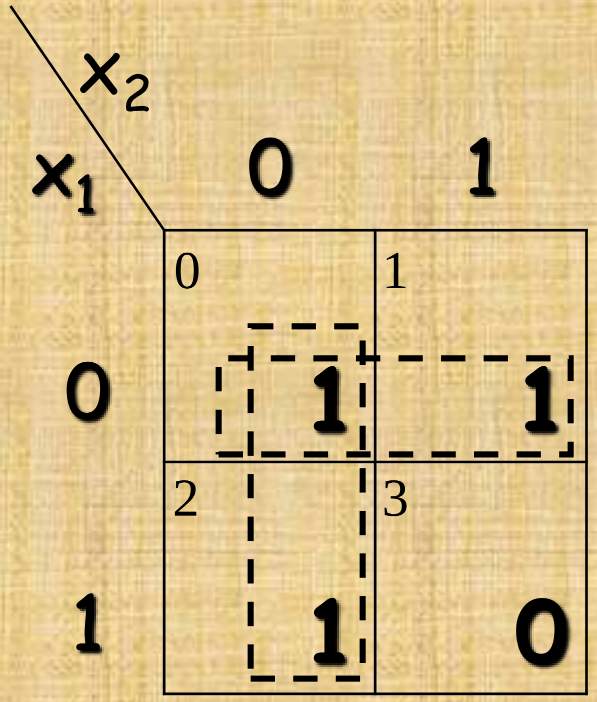
- ordering of variables is IMPORTANT for $f(x_1,x_2)$, x_1 is the row, x_2 is the column.
- Cell 0 represents $x_1'x_2'$; Cell 1 represents $x_1'x_2$; etc. If a minterm is present in the function, then a 1 is placed in the corresponding cell.

Two-Variable Map

- Any two adjacent cells in the map differ by ONLY one variable, which appears complemented in one cell and uncomplemented in the other.
- Example:
  m_0 ($=x_1'x_2'$) is adjacent to m_1 ($=x_1'x_2$) and m_2 ($=x_1x_2'$) but NOT m_3 ($=x_1x_2$)

2-Variable Map -- Example

- $f(x_1,x_2) = x_1'x_2' + x_1'x_2 + x_1x_2'$
  $= m_0 + m_1 + m_2$
  $= x_1' + x_2'$
- 1s placed in K-map for specified minterms m_0, m_1, m_2
- Grouping of 1s allows simplification
- What (simpler) function is represented by each dashed rectangle?
  - $x_1' = m_0 + m_1$
  - $x_2' = m_0 + m_2$
- Here m_0 covered twice



The 3 Variable K-Map

- There are 8 cells as shown:

AB \ C	0	1
00	$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$
01	$\bar{A}B\bar{C}$	$\bar{A}BC$
11	$AB\bar{C}$	ABC
10	$A\bar{B}\bar{C}$	$A\bar{B}C$

Example 3 var. k-map

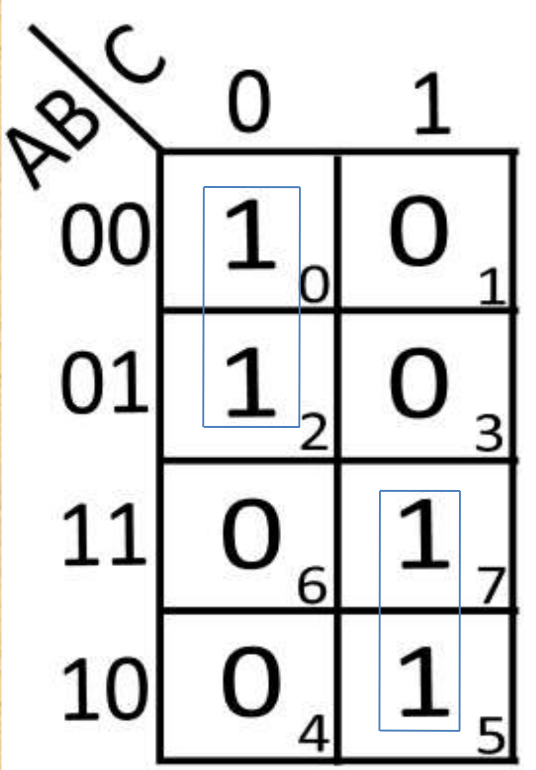
- Minimize the following equation using k-map
  $y=\bar{A}\bar{B}\bar{C}+\bar{A}B\bar{C}+A\bar{B}C+ABC$

$\bar{A}\bar{B}\bar{C} = 000 = 0 \qquad \bar{A}B\bar{C} = 010 = 2$

$A\bar{B}C = 101 = 5 \qquad ABC = 111 = 7$

Using this fill the k-map

- Grouping – here 2 groups of 2 1's
Is possible



- For upper group A and C are constants and B is varying. Neglect B.A and C both are 0. Hence output of this group is $\bar{A}\bar{C}$

- For upper group A and C are constants and B is varying. Neglect B.A and C both are 0. Hence output of this group is AC

- Thus output Y is given by ,

$$Y = AC + \bar{A}\bar{C}$$
$$= A \odot B$$

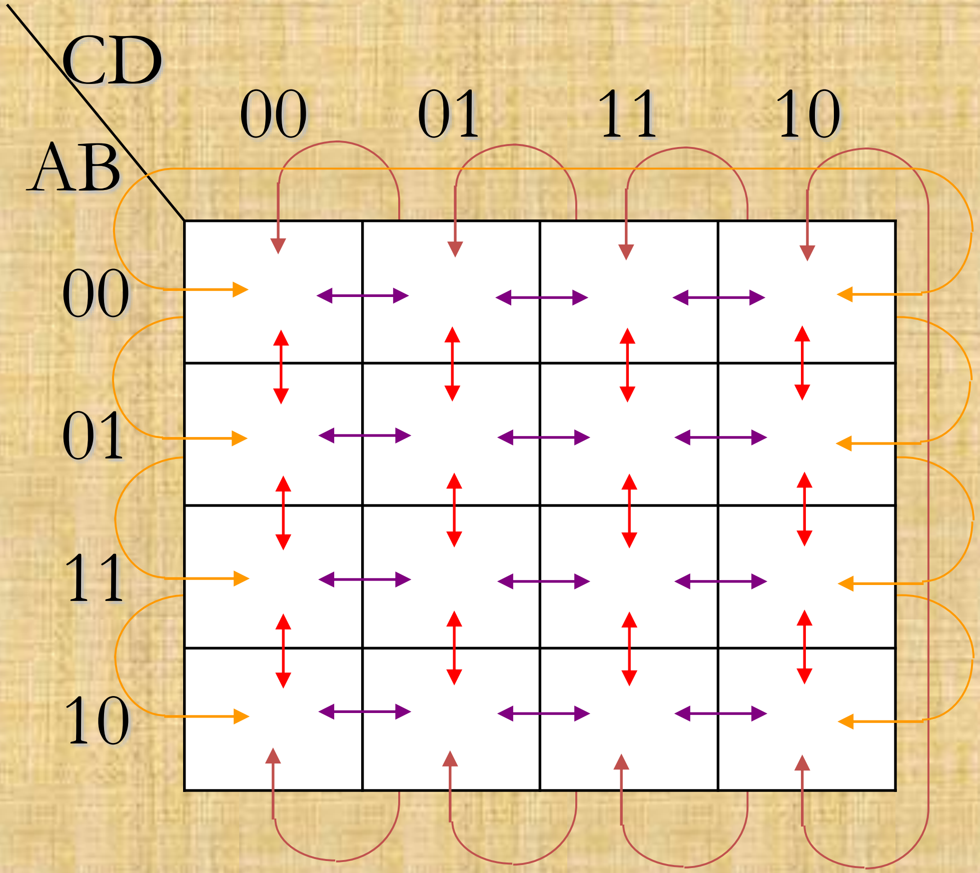
AB \ C	0	1
00	1_0	0_1
01	1_2	0_3
11	0_6	1_7
10	0_4	1_5

The 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

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}\bar{B}CD$	$\bar{A}\bar{B}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\bar{C}\bar{D}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Cell Adjacency



- Solve the given k-map

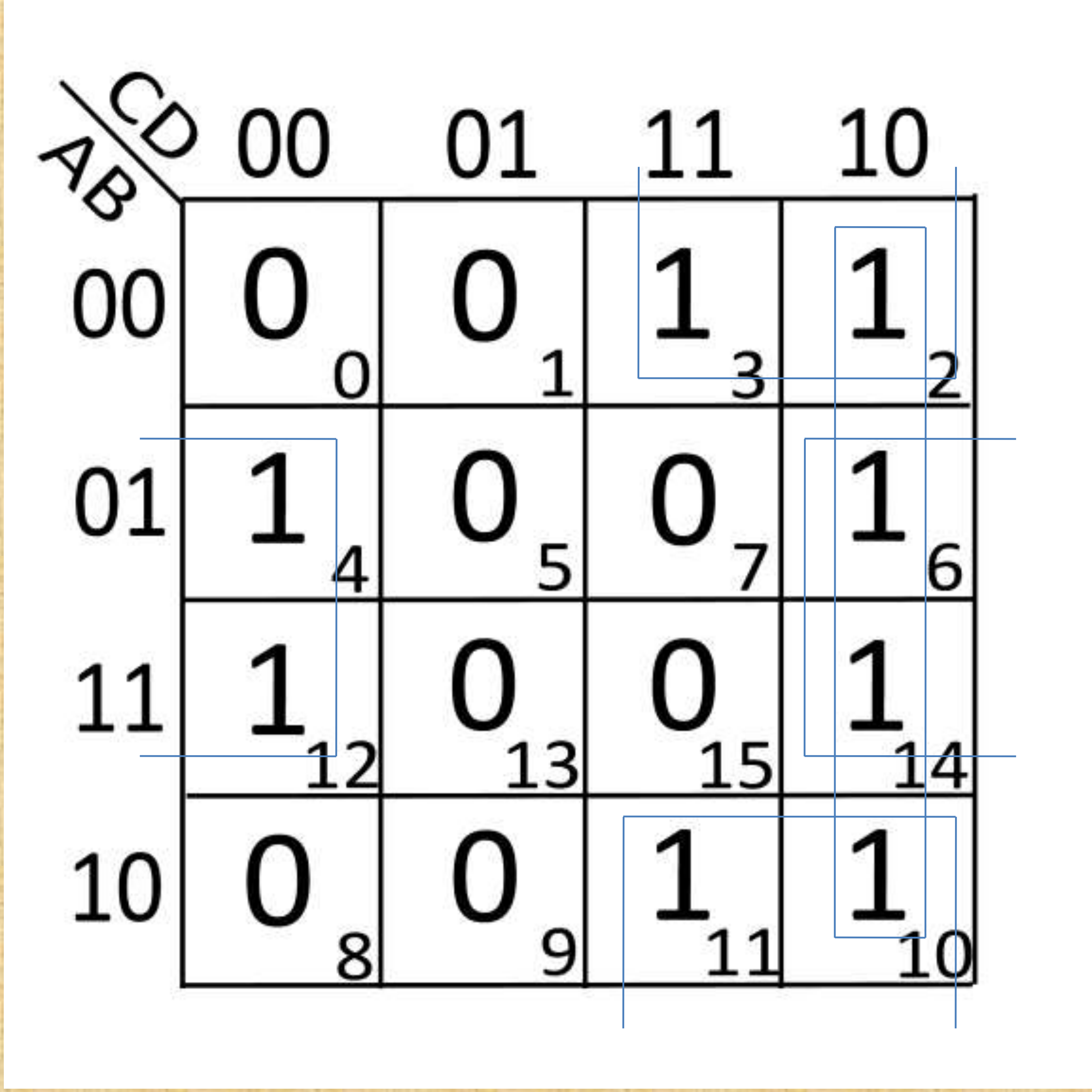
- Step I -grouping

- Step II -output of each group

- Step III -final output

➢ Here answer is ,

$$Y = C\overline{D} + \overline{B}C + B\overline{D}$$



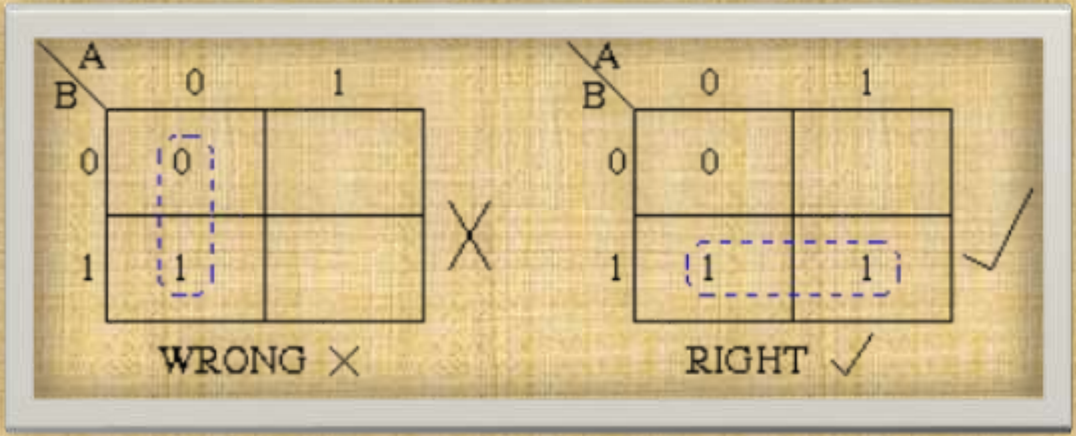
K-Map SOP Minimization

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

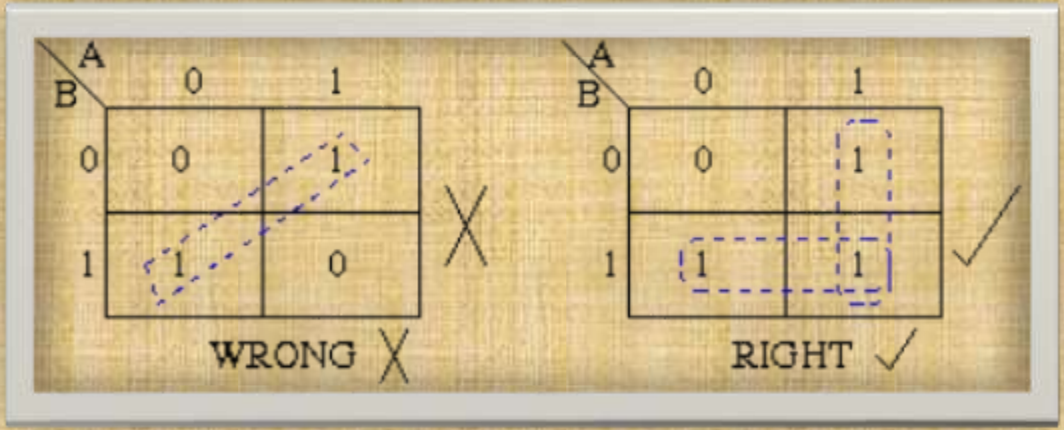
Grouping

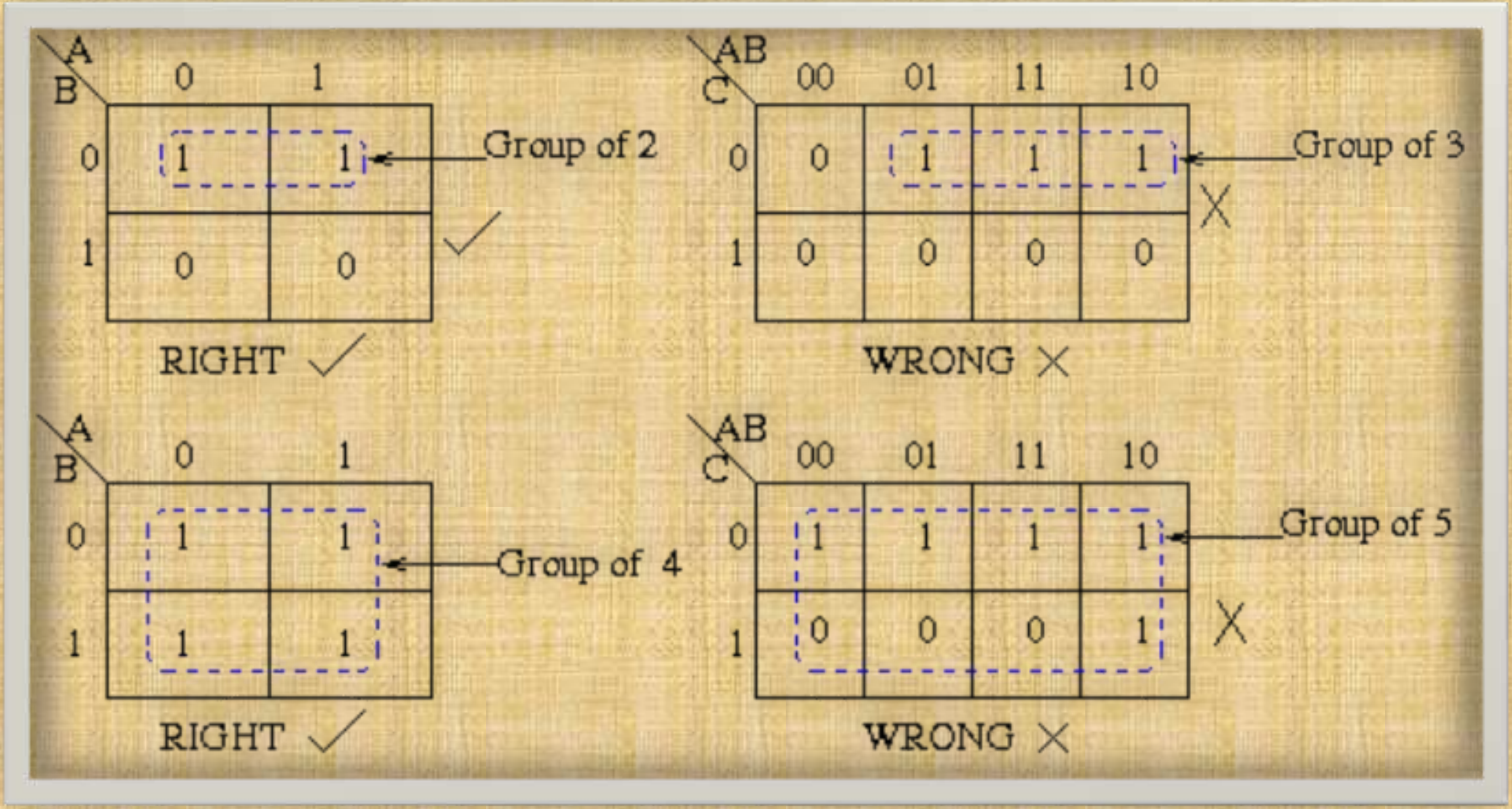
Rules of grouping -

1's & 0's can not be grouped



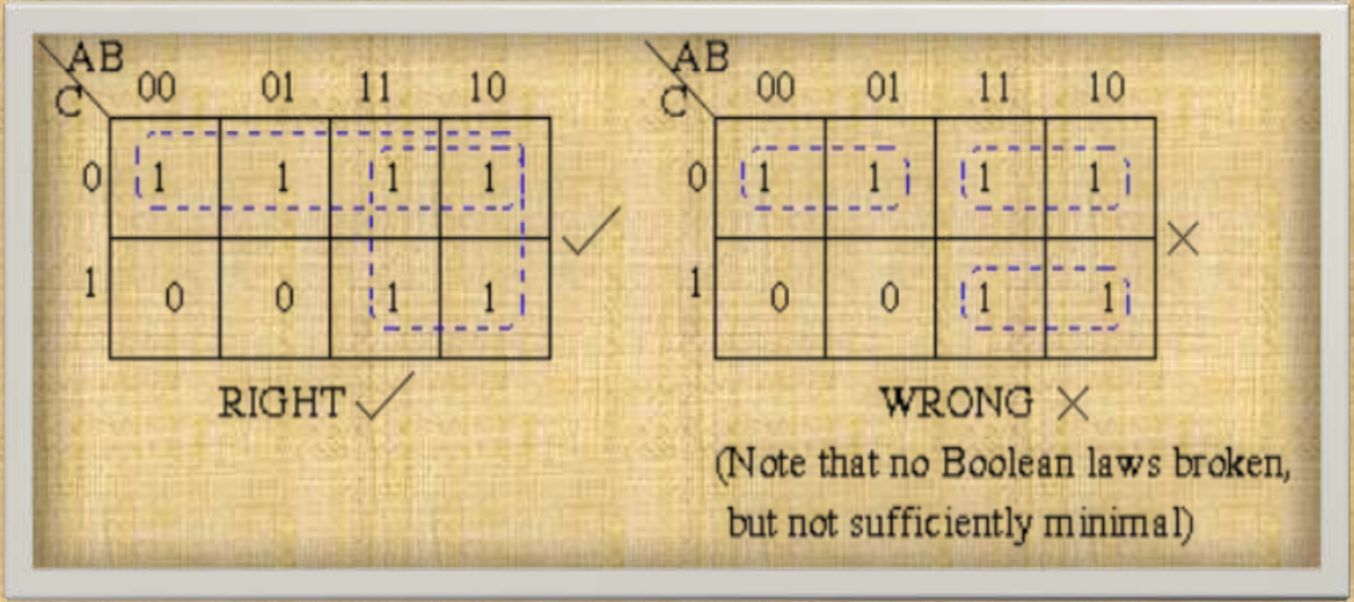
diagonal 1's can not be grouped



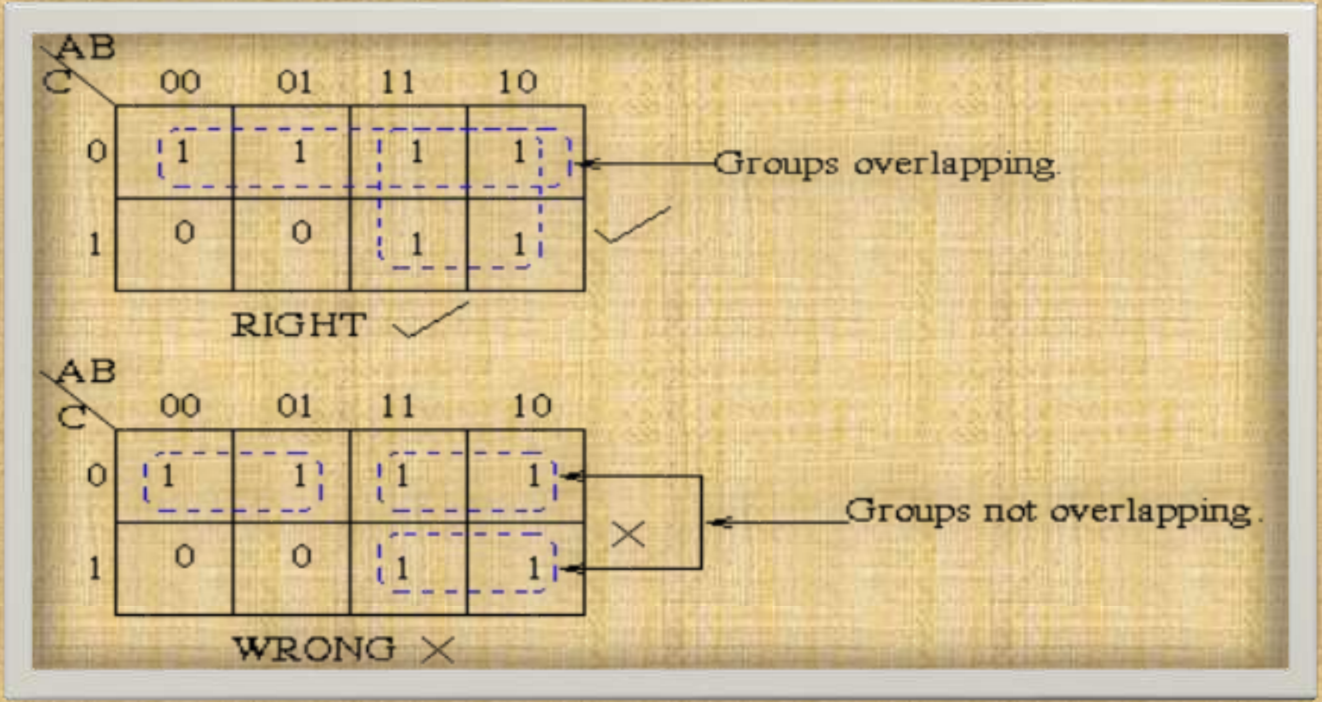


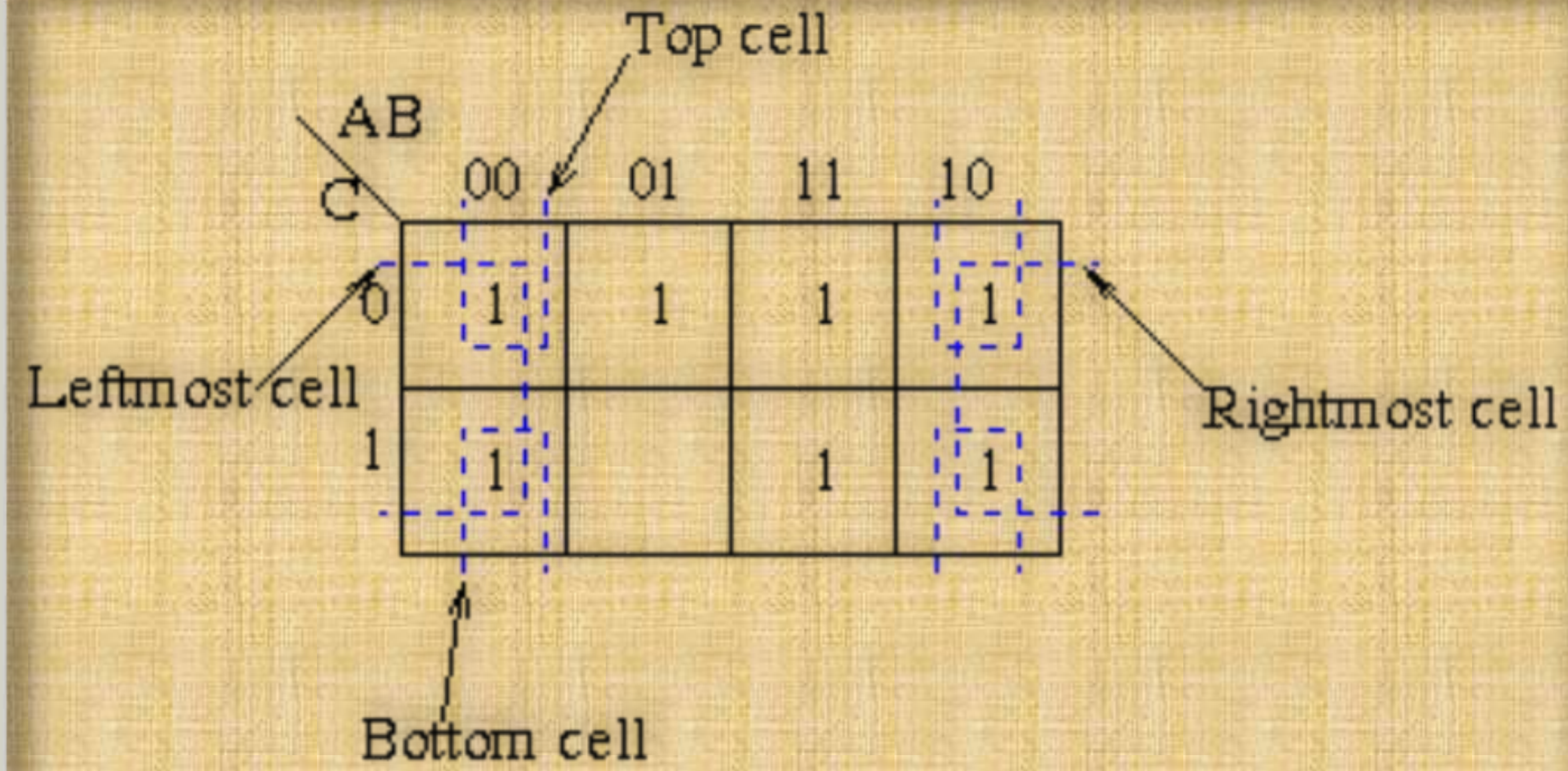
Elements in a group should be 2^n

Minimum Groups should be formed



For above rule group Overlapping is applicable




Top cell
AB
C
00
01
11
10
0
1
1
1
1
1
1
1
1
Leftmost cell
Rightmost cell
Bottom cell

Mapping a Standard SOP Expression

- For an SOP expression in standard form:
  - A 1 is placed on the K-map for each product term in the expression.
  - Each 1 is placed in a cell corresponding to the value of a product term.
  - Example: for the product term , a 1 goes in the 101 cell on a 3-variable map. $A\overline{B}C$

AB \ C	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$
01	$\overline{A}B\overline{C}$	$\overline{A}BC$
11	$AB\overline{C}$	ABC
10	$A\overline{B}\,\overline{C}$	$A\overline{B}C$

Mapping a Standard SOP Expression

The expression:

$$\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + AB\overline{C} + A\overline{B}\,\overline{C}$$

000	001	110	100

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

Practice:

$\overline{A}\,\overline{B}C + \overline{A}B\overline{C} + AB\overline{C} + ABC$

$\overline{A}BC + A\overline{B}C + A\overline{B}\,\overline{C}$

$\overline{A}\,\overline{B}CD + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + ABCD + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}C\overline{D}$

Three-Variable K-Maps

$f = \sum(0,4) = \overline{B}\,\overline{C}$

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

$f = \sum(4,5) = A\,\overline{B}$

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

$f = \sum(0,1,4,5) = \overline{B}$

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

$f = \sum(0,1,2,3) = \overline{A}$

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

$f = \sum(0,4) = \overline{A}\,C$

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

$f = \sum(4,6) = A\,\overline{C}$

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

$f = \sum(0,2) = \overline{A}\,\overline{C}$

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

$f = \sum(0,2,4,6) = \overline{C}$

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

Four-Variable K-Maps

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

$$f = \sum(0,8) = \overline{B} \bullet \overline{C} \bullet \overline{D}$$

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

$$f = \sum(5,13) = B \bullet \overline{C} \bullet D$$

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

$$f = \sum(13,15) = A \bullet B \bullet D$$

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

$$f = \sum(4,6) = \overline{A} \bullet B \bullet \overline{D}$$

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

$$f = \sum(2,3,6,7) = \overline{A} \bullet C$$

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

$$f = \sum(4,6,12,14) = B \bullet \overline{D}$$

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

$$f = \sum(2,3,10,11) = \overline{B} \bullet C$$

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

$$f = \sum(0,2,8,10) = \overline{B} \bullet \overline{D}$$

Four-Variable K-Maps

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

$f = \sum(4,5,6,7) = \overline{A} \bullet B$

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

$f = \sum(3,7,11,15) = C \bullet D$

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

$f = \sum(0,3,5,6,9,10,12,15)$

$f = A \otimes B \otimes C \otimes D$

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

$f = \sum(1,2,4,7,8,11,13,14)$

$f = A \oplus B \oplus C \oplus D$

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

$f = \sum(1,3,5,7,9,11,13,15)$

$f = D$

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

$f = \sum(0,2,4,6,8,10,12,14)$

$f = \overline{D}$

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

$f = \sum(4,5,6,7,12,13,14,15)$

$f = B$

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

$f = \sum(0,1,2,3,8,9,10,11)$

$f = \overline{B}$

Determining the Minimum SOP Expression from the Map

$$B + \overline{A}C + A\overline{C}D$$

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

$\overline{A}C$

B

$A\overline{C}D$

Determining the Minimum SOP Expression from the Map

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

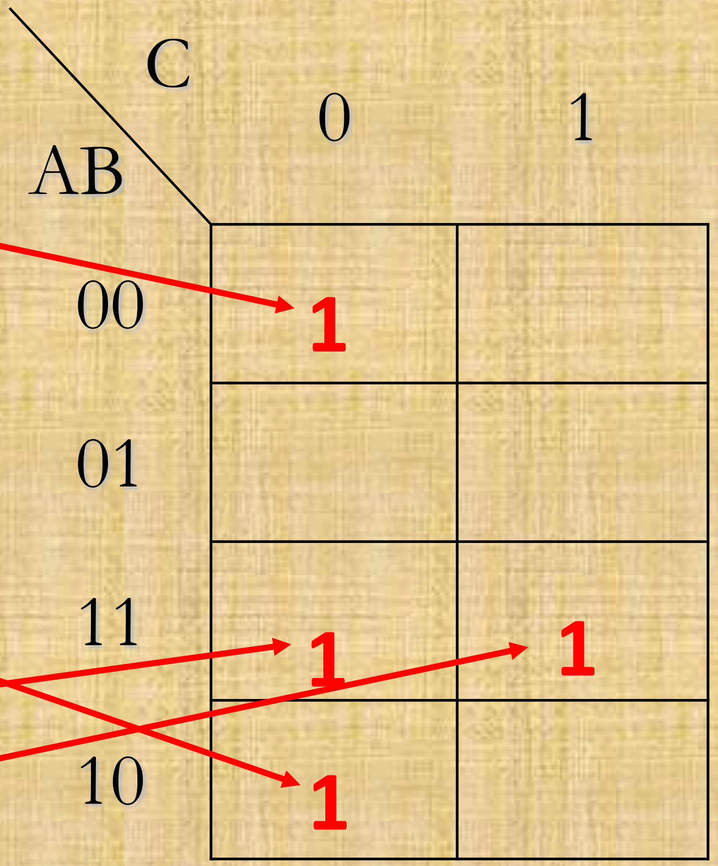
$$AB + BC + \overline{A}\,\overline{B}\,\overline{C}$$

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

$$\overline{B} + \overline{A}\,\overline{C} + AC$$

Mapping Directly from a Truth Table

I/P			O/P
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Don’t Care Conditions

- A *don’t care* condition, marked by (X) in the truth table, indicates a condition where the design doesn’t care if the output is a (0) or a (1).
- A *don’t care* condition can be treated as a (0) or a (1) in a K-Map.
- Treating a *don’t care* as a (0) means that you do not need to group it.
- Treating a *don’t care* as a (1) allows you to make a grouping larger, resulting in a simpler term in the SOP equation.

Some You Group, Some You Don’t

	$\overline{C}$	C
$\overline{A}\,\overline{B}$	X	0
$\overline{A}B$	1	0
AB	0	0
$A\overline{B}$	X	0

$\overline{A}\,\overline{C}$

This *don’t care* condition was treated as a (1). This allowed the grouping of a single one to become a grouping of two, resulting in a simpler term.

There was no advantage in treating this *don’t care* condition as a (1), thus it was treated as a (0) and not grouped.

Example

Solution:

R	S	T	U	F_4
0	0	0	0	X
0	0	0	1	0
0	0	1	0	1
0	0	1	1	X
0	1	0	0	0
0	1	0	1	X
0	1	1	0	X
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	X
1	1	0	0	X
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$\overline{R}\,T$

	$\overline{T}\,\overline{U}$	$\overline{T}\,U$	$T\,U$	$T\,\overline{U}$
$\overline{R}\,\overline{S}$	X	0	X	1
$\overline{R}\,S$	0	X	1	X
$R\,S$	X	0	0	0
$R\,\overline{S}$	1	1	X	1

$R\,\overline{S}$

$$F_4 = \overline{R}\,T + R\,\overline{S}$$

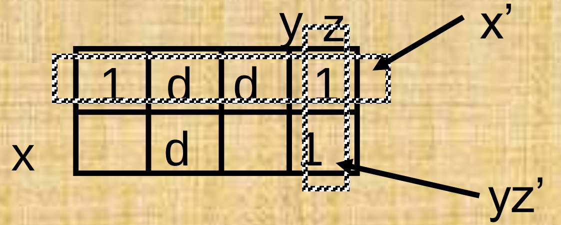
IMPLEMENTATION OF K-MAPS

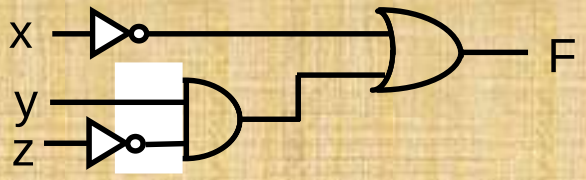
- In some logic circuits, the output responses for some input conditions are don't care whether they are 1 or 0.

- In K-maps, don't-care conditions are represented by d's 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





K-Map POS Minimization

- The approaches are much the same (as SOP) except that with POS expression, 0s representing the standard sum terms are placed on the K-map instead of 1s.

Mapping a Standard POS Expression

The expression:

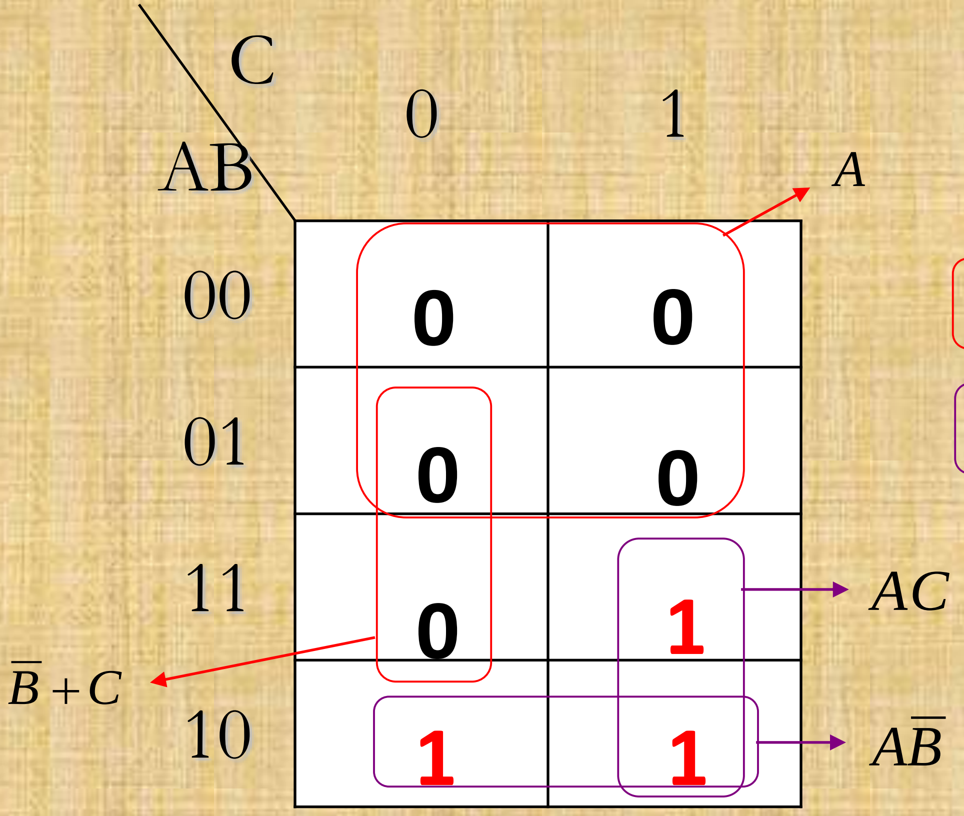
$$(A+B+C)(A+\overline{B}+C)(\overline{A}+\overline{B}+C)(\overline{A}+B+\overline{C})$$

000 010 110 101

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

K-map Simplification of POS Expression

$$(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+\overline{B}+C)$$



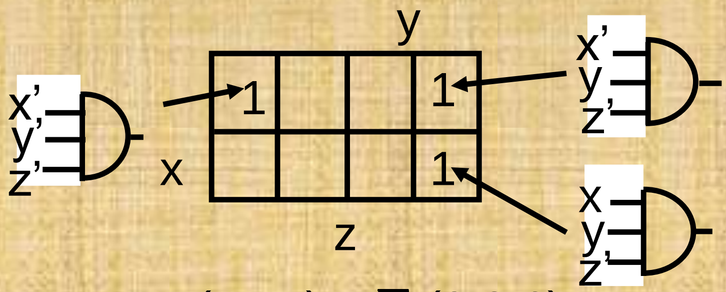
$A(\overline{B}+C)$

$A\overline{B}+AC$

IMPLEMENTATION OF K-MAPS - Sum-of-Products Form -

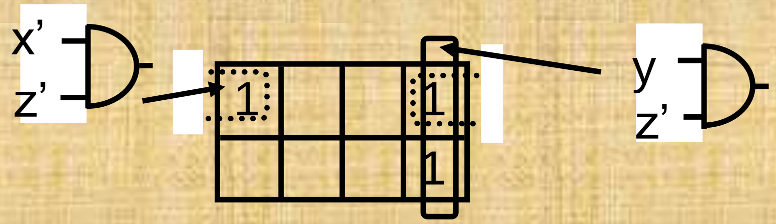
- Logic function represented by a Karnaugh map can be implemented in the form of not-AND-OR

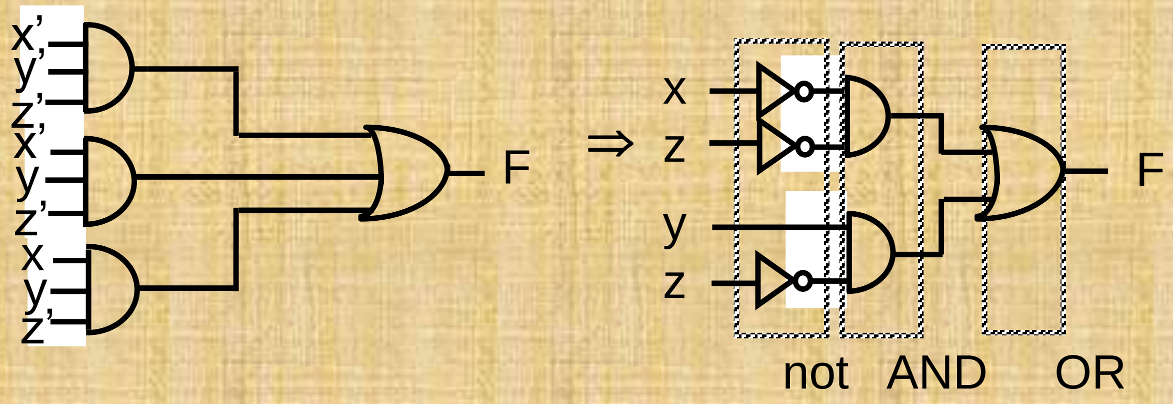
- A cell or a collection of the adjacent 1-cells can be realized by an AND gate, with some inversion of the input variables.



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

⇒



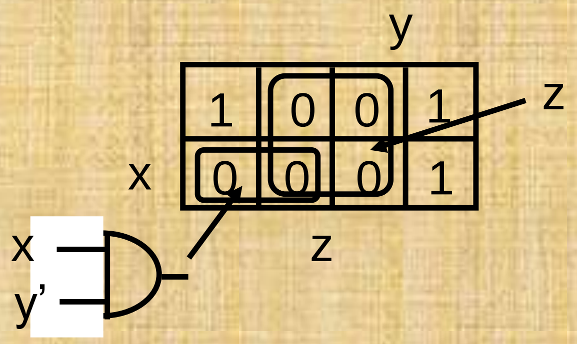


IMPLEMENTATION OF K-MAPS - Product-of-Sums Form -

- Logic function represented by a Karnaugh map can be implemented in the form of I-OR-AND

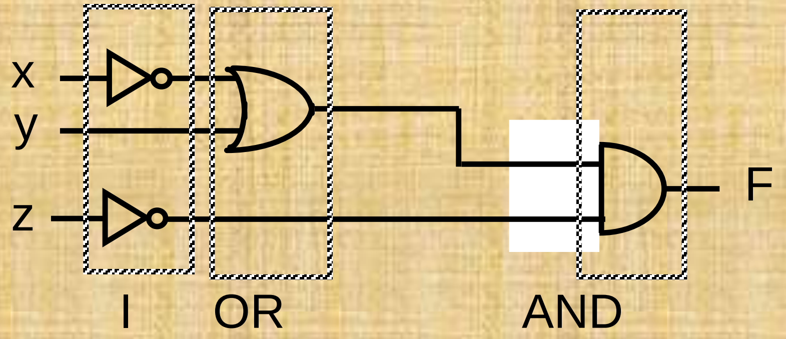
- If we implement a Karnaugh map using 0-cells, the complement of F, i.e., F', can be obtained. Thus, by complementing F' using DeMorgan's theorem F can be obtained

F(x,y,z) = (0,2,6)



F' = xy' + z

F = (xy')z'
  = (x' + y)z'



Design of combinational digital circuits

- Steps to design a combinational digital circuit:
  - From the problem statement derive the truth table
  - From the truth table derive the unsimplified logic expression
  - Simplify the logic expression
  - From the simplified expression draw the logic circuit

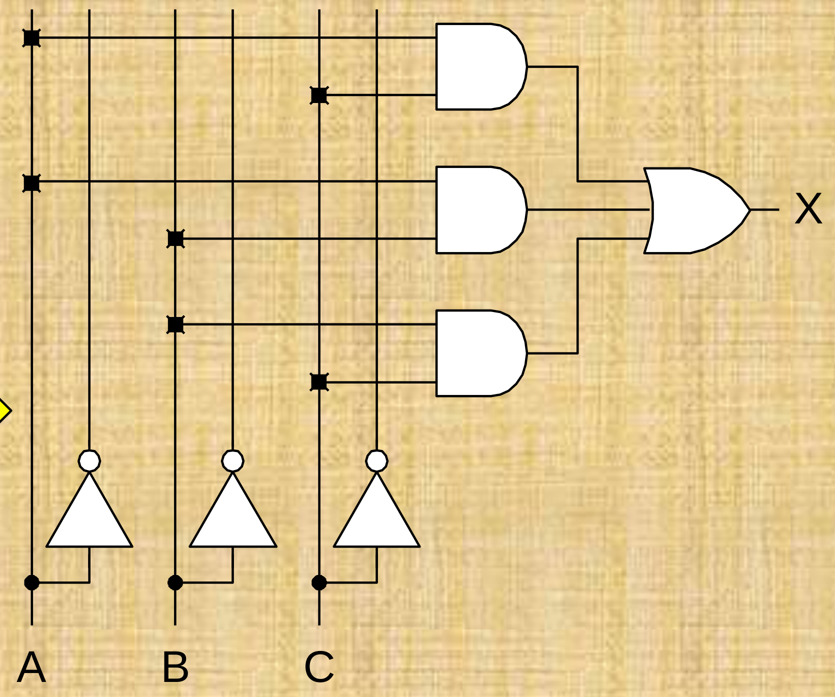
Example: Design a 3-input (A,B,C) digital circuit that will give at its output (X) a logic 1 only if the binary number formed at the input has more ones than zeros.

	Inputs			Output
	A	B	C	X
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$$X = \sum (3, 5, 6, 7)$$

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

$$X = AC + AB + BC$$



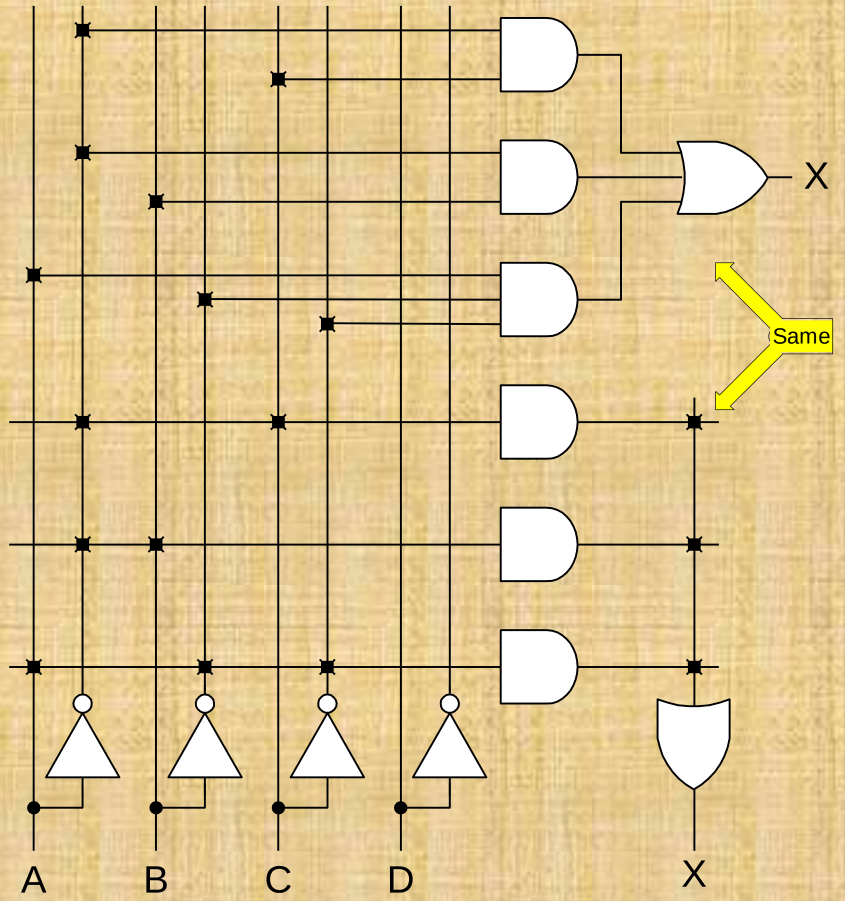
Example: Design a 4-input (A,B,C,D) digital circuit that will give at its output (X) a logic 1 only if the binary number formed at the input is between 2 and 9 (including).

	Inputs				Output
	A	B	C	D	X
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

$$X = \sum(2,3,4,5,6,7,8,9)$$

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

$$X = \overline{A}C + \overline{A}B + A\overline{B}\,\overline{C}$$



THANK-YOU