

Digital Systems- ITC131

Dr. Alaa Eldeen S. Mohamed

Lecture 05: Gate Level Minimization(K-Map)

Contents

➤ Circuit Optimization

- The Map Method
- Two-Variable Map
- Three-Variable Map
- Four-Variable Map
- Don't Care Conditions
- Product of Sums Simplification

Gate Level Minimization(K-Map)

Circuit Optimization

- **Complexity** of a logic circuit is directly related to the algebraic expression.
- Simplification using algebraic manipulation (in part1) has issues:
 - How to know succeeding steps in the manipulative process
 - Difficult to determine whether the simplest expression has been achieved.
- To obtain the simplest implementation for a given function, we can use a graphical technique using Karnaugh maps (K-maps)

Gate Level Minimization(K-Map)

Karnaugh Maps (K-map)

- A K-map is a collection of squares
 - Each square represents a **minterm** (or a row in the truth table)
 - Adjacent squares differ in the value of one variable
- A K-map is a graphical representation of a Boolean function

Karnaugh Maps (K-map)

- A K-map is a reorganized version of the truth table
- In K-map, the number of cells is similar to the total number of variable input combinations. For example, if the number of variables is two, the number of cells is $2^2 = 4$. if the number of variables is three, the number of cells is $2^3 = 8$, and if the number of variables is four, the number of cells is 2^4 .
- K-map provide a means for finding optimum or near optimum
 - SOP and POS standard forms, and

Gate Level Minimization(K-Map)

Karnaugh Maps (K-map)

Two Variable K-Maps

Every 2 adjacent cells differ in the value of 1 variable only.

- m_0 and m_1 are adjacent and differ in the value of the y variable.
- m_0 and m_2 differ in the x variable.
- m_1 and m_3 differ in the x variable.
- m_2 and m_3 differ in the value of the variable y .

	$y = 0$	$y = 1$
$x = 0$	m_0 $x'y'$	m_1 $x'y$
$x = 1$	m_2 xy'	m_3 xy

Gate Level Minimization(K-Map)

K-Map Function Representation

Example:

$$F(x,y) = \sum m_2, m_3$$

X	Y	F
0	0	0
0	1	0
1	0	1
1	1	1

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

		Y
	0	0
X	1	1

Gate Level Minimization(K-Map)

K-Map Function Representation

Example:

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

$$G(x,y) = \sum m1, m2, m3$$

X	Y	G
0	0	0
0	1	1
1	0	1
1	1	1

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

		Y
	0	1
X	1	1

Gate Level Minimization(K-Map)

Karnaugh Maps (K-map)

A three-variable K-map:

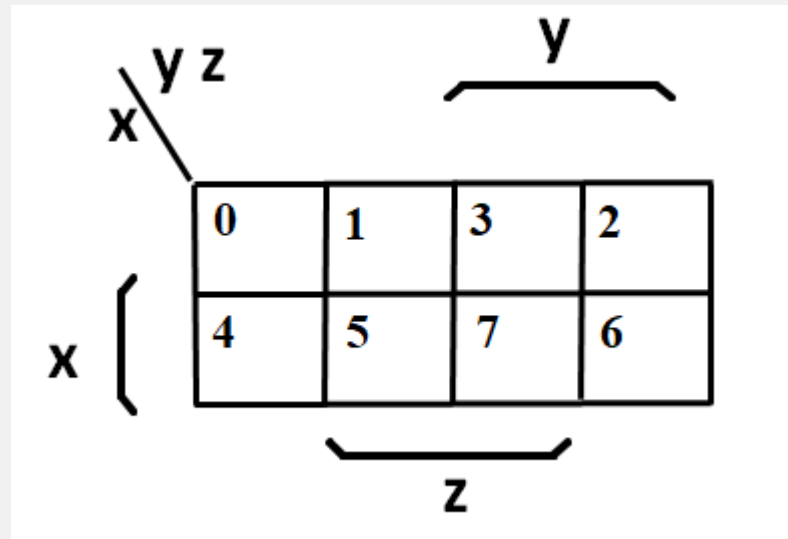
- Where each minterm corresponds to the product terms:
- Note that if the binary value for an index differs in one bit position, the minterms are adjacent on the K-Map

	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

		Y			
		00	01	11	10
X	0	$\bar{X}\bar{Y}\bar{Z}$	$\bar{X}\bar{Y}Z$	$\bar{X}YZ$	$\bar{X}Y\bar{Z}$
	1	$X\bar{Y}\bar{Z}$	$X\bar{Y}Z$	XYZ	$XY\bar{Z}$
		Z			

Alternative Map Labeling

- Alternate labeling are useful:



Gate Level Minimization(K-Map)

Example Functions

- By convention, we represent the minterms of F by a "1" in the map and leave the minterms of F' blank

Example:

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

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

Example:

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

				<i>y</i>
	0	1	3 <i>1</i>	2
<i>x</i>	4 <i>1</i>	5	7 <i>1</i>	6 <i>1</i>
			<i>z</i>	

Gate Level Minimization(K-Map)

Combining Squares

- By combining squares, we reduce number of literals in a product term
- On a 2-variable K-Map:

# of adjacent squares	Represents
1	A minterm with two variables
2	A product term with one variables
4	The function of all ones (no variables) = 1

Gate Level Minimization(K-Map)

Combining Squares

- By combining squares, we reduce number of literals in a product term
- On a 3-variable K-Map:

# of adjacent squares	Represents
1	A minterm with three variables
2	A product term with two variables
4	A product term with one variable
8	The function of all ones (no variables) = 1

Gate Level Minimization(K-Map)

Combining Squares

Groupings to select product-terms must be:

1. Rectangular in shape
2. In powers of twos (1, 2, 4, 8, etc.)
3. Always select largest possible groupings of minterms (i.e. prime implicants)
4. Eliminate redundant groupings

Combining Squares

- Groups may not include any cell containing a **zero**

A \ B	0	1
0	0	
1	1	

WRONG X

A \ B	0	1
0	0	
1	1	1

RIGHT ✓

- Groups may be horizontal or vertical, but not diagonal.

A \ B	0	1
0	0	1
1	1	0

WRONG X

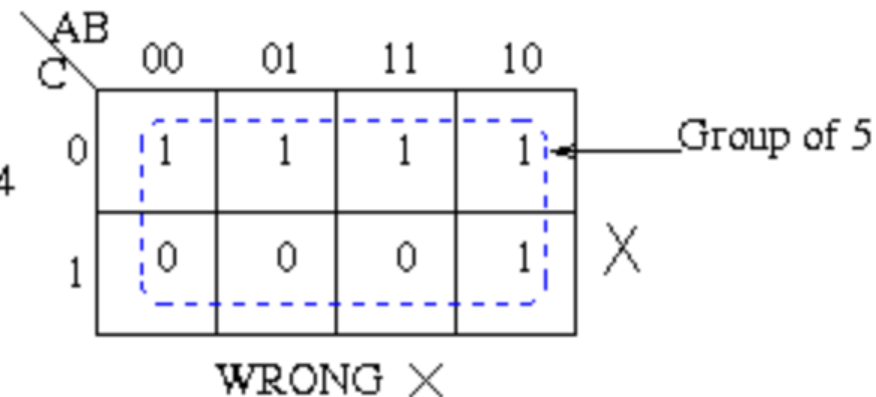
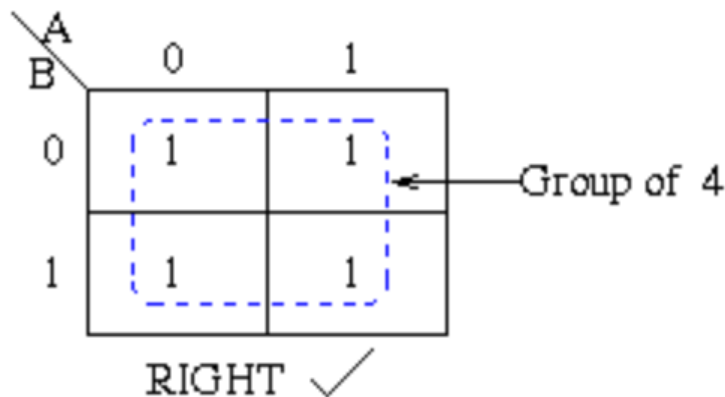
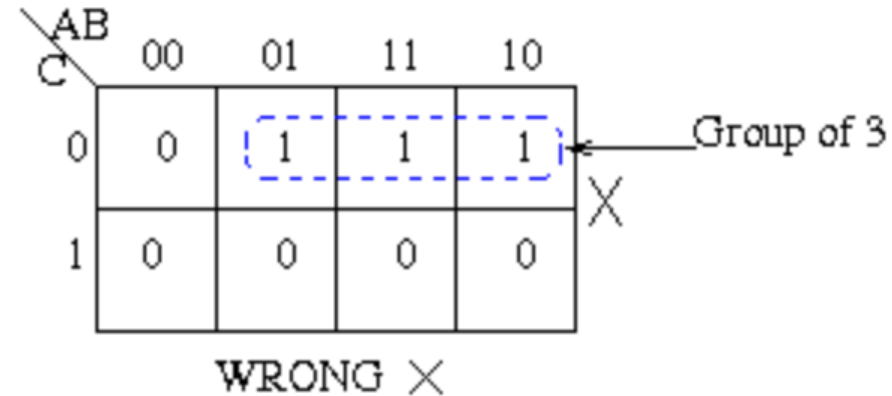
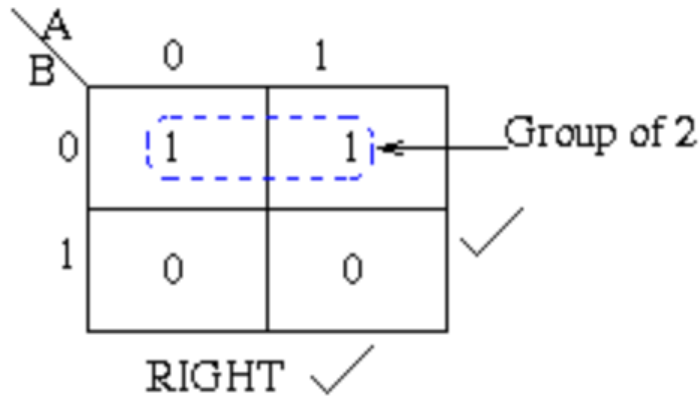
A \ B	0	1
0	0	1
1	1	1

RIGHT ✓

Gate Level Minimization(K-Map)

Combining Squares

- Groups must contain 1, 2, 4, 8, or in general 2^n cells.
That is if $n = 1$, a group will contain two 1's since $2^1 = 2$.
If $n = 2$, a group will contain four 1's since $2^2 = 4$.



Combining Squares

- Each group should be as large as possible.

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

RIGHT ✓

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

WRONG ✗

(Note that no Boolean laws broken, but not sufficiently minimal)

- Each cell containing a **one** must be in at least one group.

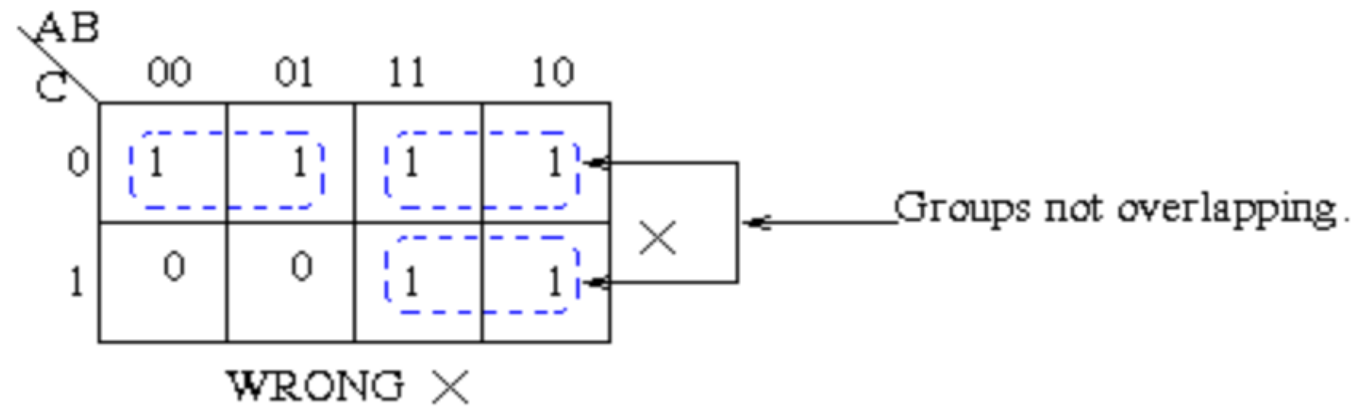
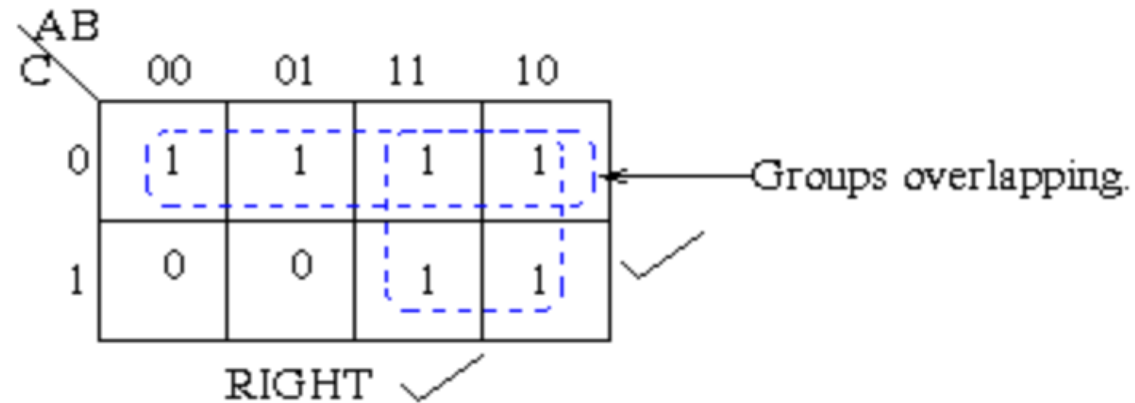
AB \ C	00	01	11	10
0	0	0	1	1
1	0	0	0	1

Group I (points to the top row of 1s)
Group II (points to the bottom-right 1)
1 present in at least one group.

Gate Level Minimization(K-Map)

Combining Squares

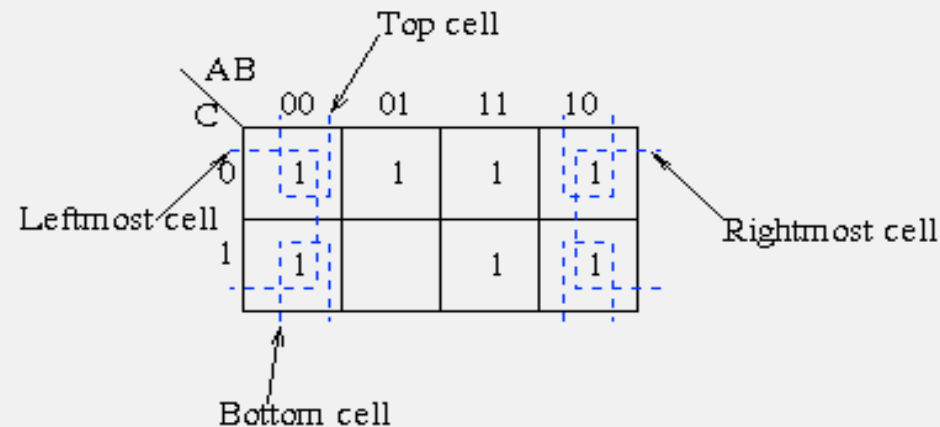
- Groups may overlap.



Gate Level Minimization(K-Map)

Combining Squares

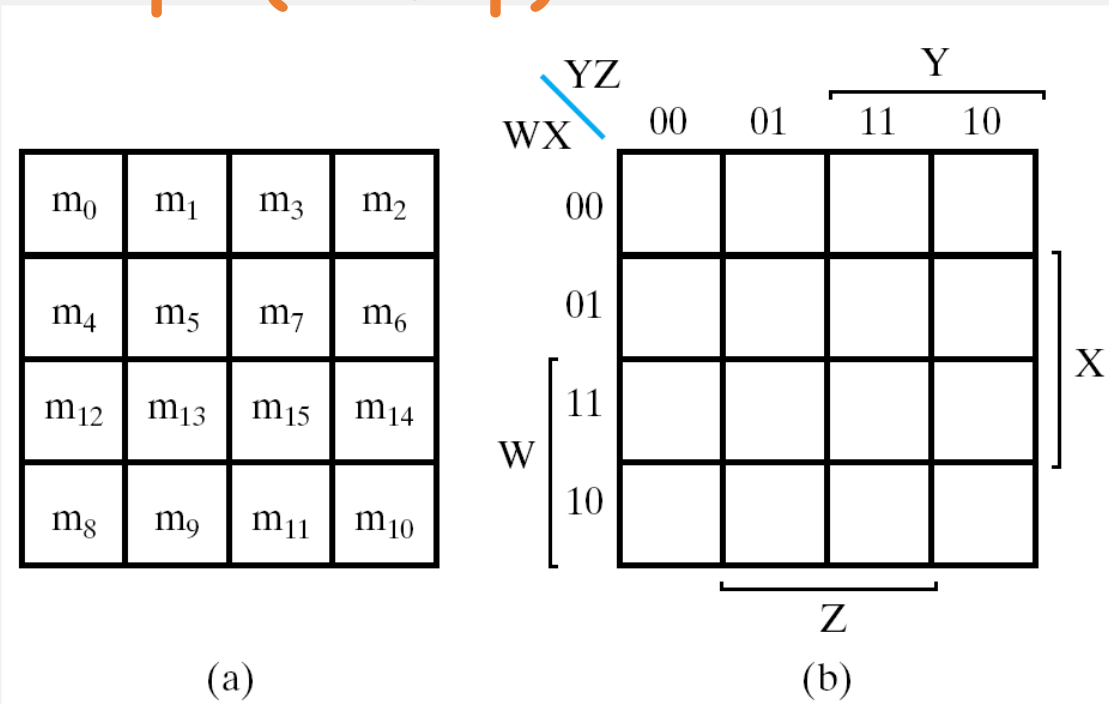
- Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



Karnaugh Maps (K-map)

Four variable maps can have rectangles corresponding to:

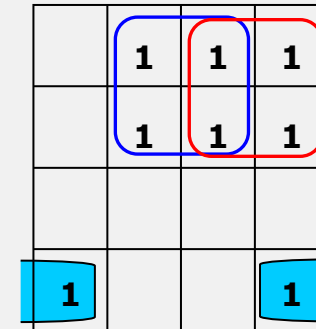
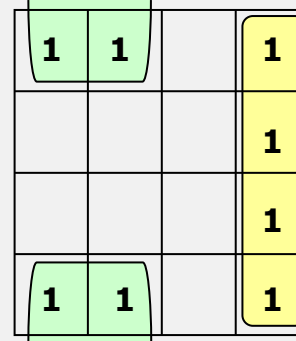
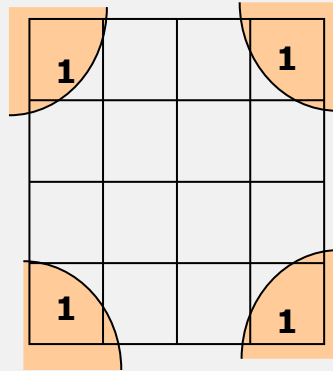
- A single 1 = 4 variables, (i.e. Minterm)
- Two 1s = 3 variables,
- Four 1s = 2 variables
- Eight 1s = 1 variable,
- Sixteen 1s = zero variables (i.e. Constant "1")



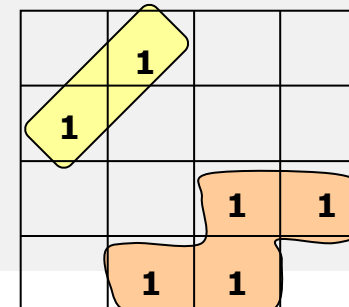
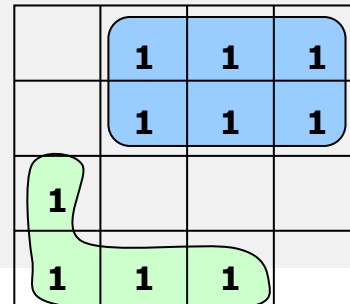
Four-Variable Maps

Karnaugh Maps (K-map)

- Possible valid groupings of a 4-variable K-map include: **Four-Variable Maps**



- Invalid groups examples:*



Gate Level Minimization(K-Map)

Combining Squares

- By combining squares, we reduce number of literals in a product term
- On a 4-variable K-Map:

# of adjacent squares	Represents
1	A minterm with four variables
2	A product term with three variables
4	A product term with two variable
8	A product term with one variable
16	The function of all ones (no variables) = 1

Gate Level Minimization(K-Map)

Simplest SOP Expressions

- To find the simplest possible **sum of products** (SoP) expression from **a K-map**, you need to do the following:
 1. define the given expression in its canonical form
 2. create the K-map by entering 1 to each product-term into the K-map cell and fill the remaining cells with zeros

3. Form the groups by considering each one in the K-map.

0	0	1	1
1	1	0	0

4. Group the number of ones in the decreasing order. First, we have to try to make the group of eight, then for four, after that two and lastly for 1.
5. find the boolean expression for each group. By looking at the common variables in cell-labeling, we define the groups in terms of input variables.

Gate Level Minimization(K-Map)

Before Simplification

$$F(A,B,C) = \sum m2, m3, m4, m5$$

$$= A'.B.C' + A'.B.C + A.B'.C' + A.B'.C'$$

After simplification

$$F(A,B,C) = A'.B + A.B'$$

A \ BC		B	
		0	1
A	0	0	1
	1	1	0
		C	
		0	1

Gate Level Minimization(K-Map)

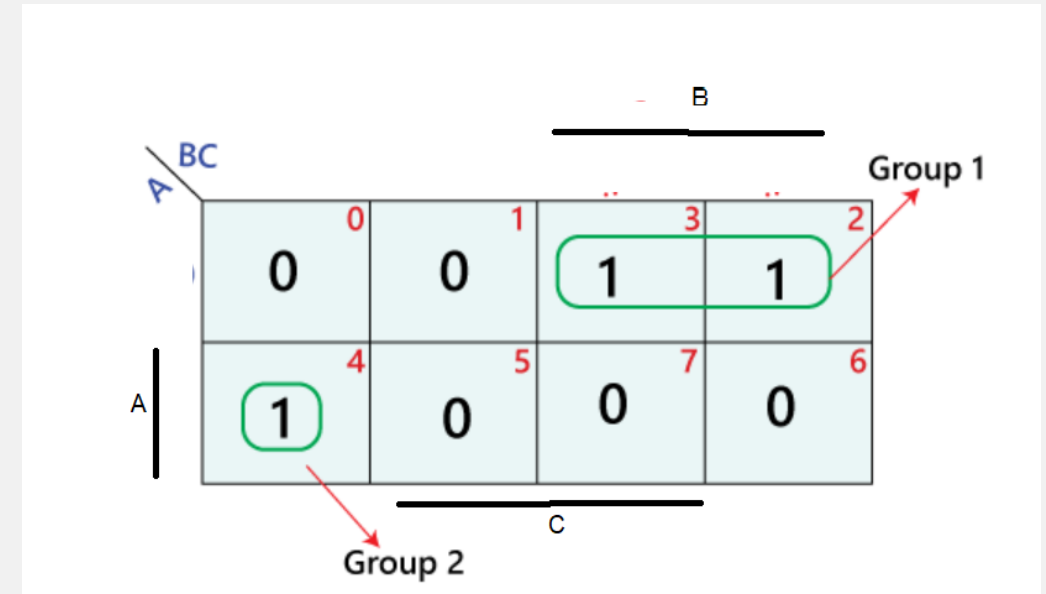
Before Simplification

$$F(A,B,C) = \sum m_2, m_3, m_4$$

$$= A'.B.C' + A'.B.C + A.B'.C'$$

After simplification

$$F(A,B,C) = A.B'.C' + A'.B$$



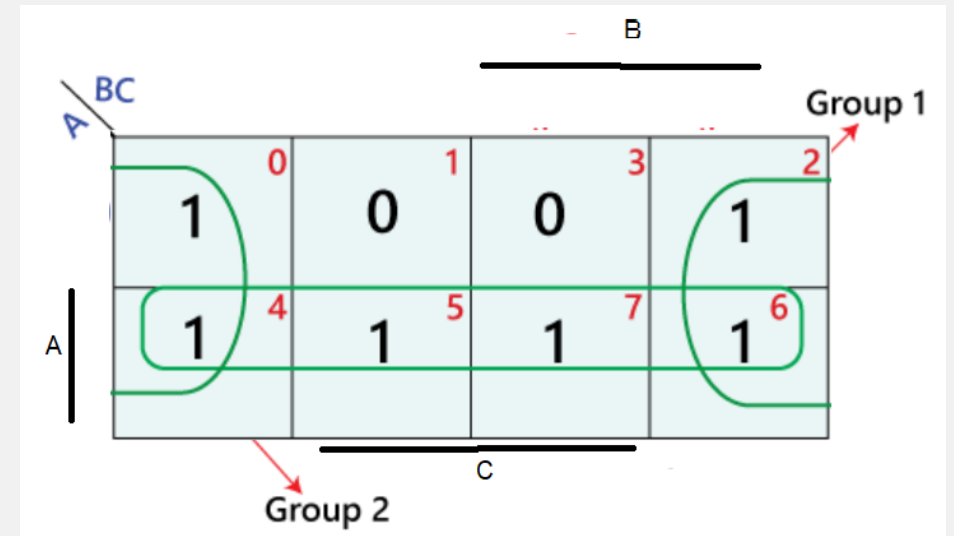
Gate Level Minimization(K-Map)

Before Simplification

$$\begin{aligned}
 F(A,B,C) &= \sum m_0, m_2, m_4, m_5, m_6, m_7 \\
 &= A'.B'.C' + A'.B.C' + A.B'.C' + AB'C \\
 &\quad + A.B.C' + A.B.C
 \end{aligned}$$

After simplification

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

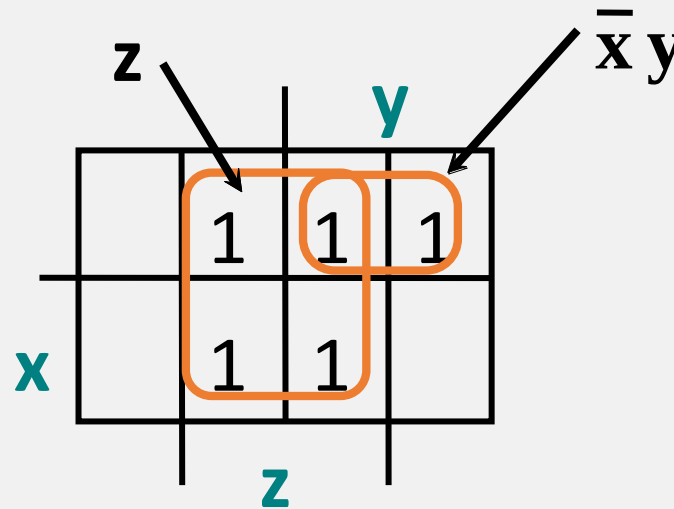


Gate Level Minimization(K-Map)

Karnaugh Maps (K-map)

Example:

Simplify $F(x,y,z)=\Sigma(1,2,3,5,7)$



$$F(x,y,z)=x'y+z$$

Gate Level Minimization(K-Map)

Karnaugh Maps (K-map)

Example:

- Simplify $F(x,y,z) = \Sigma(2,3,6,7)$

$$F=y$$

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

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

Gate Level Minimization(K-Map)

Product-of-Sums Optimization

- The optimized Boolean functions derived from the maps in all of the previous examples were expressed in **sum-of-products** form.
- With only minor modification, the **product-of-sums** form can be obtained.
- STEPS:
 1. Mark the **empty** squares with **0s**
 2. Combine them into **valid rectangles**
 3. Read the product terms to obtain **F'**
 4. Take the complement of F' to obtain F as a **product-of-sums**.

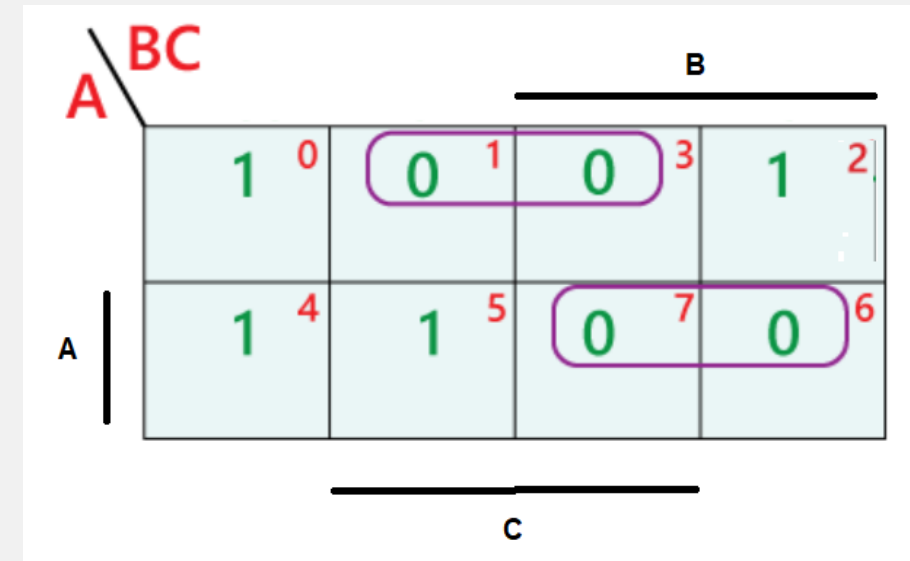
- Simplify the following function using K.MAP in the form of product of sum

$$F = \sum 0, 2, 4, 5$$

Solution:

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

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



Gate Level Minimization(K-Map)

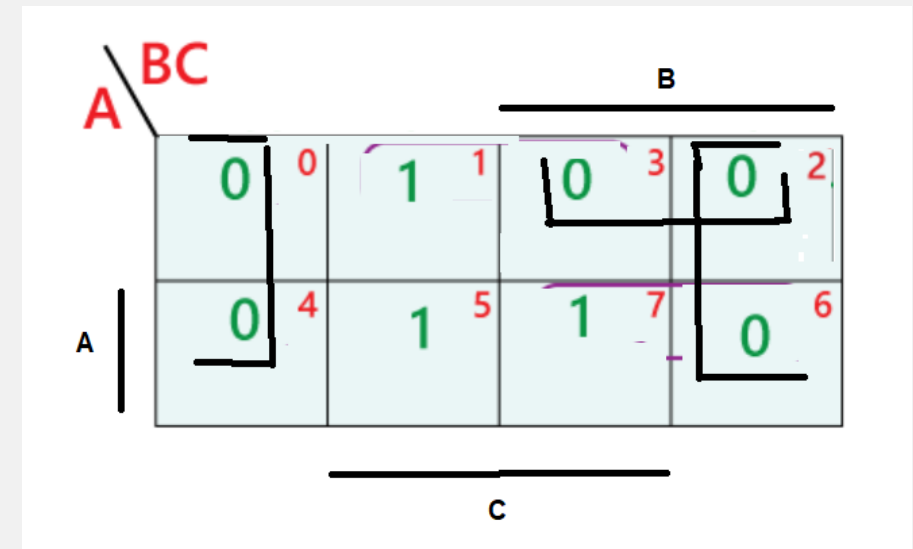
- Simplify the following function using K.MAP in the form of product of sum

$$F = \sum 1, 5, 7$$

Solution:

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

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



Gate Level Minimization(K-Map)

Don't Cares in K-Maps

- Sometimes a function table or map contains :
 - Input values for the minterm will never occur,
 - Output value for the minterm is not used
- In these cases, the output value is defined as a "don't care" (an "x" entry).
- By doing that. the cost of the logic circuit of that function may be lowered.

Gate Level Minimization(K-Map)

Don't Cares in K-Maps

Example :

- Assume A 3 variable logic function F is defined as follow:
- The output is equal to one of its input is <3 and ≥ 6
- The output is not specified when the input = 3
- The output is zero otherwise.
- Since the output @ 3 never occurs, it is represented as don't care..

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

Gate Level Minimization(K-Map)

Don't Cares in K-Maps

- Ultimately, each "x" entry may take on either a 0 or 1 value in resulting solutions.
- For example, an "x" may take on value "0" in an SOP solution and value "1" in a POS solution (If not covered), or vice-versa (If covered).
- Any minterm with value "x" need not be covered by a prime implicant.

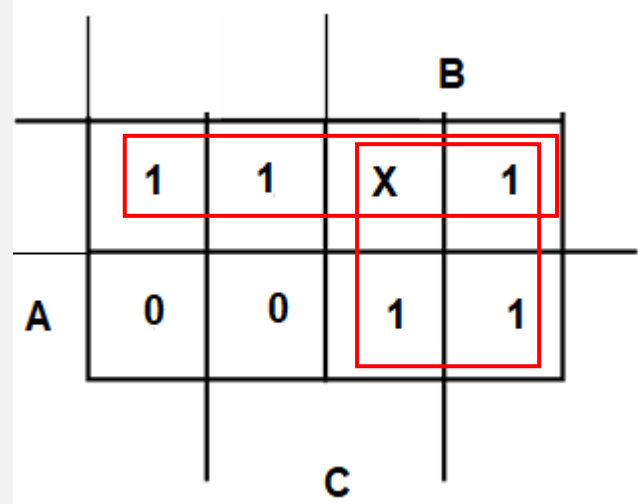
Product of Sums Example

Gate Level Minimization(K-Map)

Find the optimum **SoP** solution:

$$F(A,B,C) = \Sigma(m_0, m_1, m_2, m_6, m_7) + \Sigma_d(m_3)$$

$$F = A' + B$$



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

Find The simplified function in terms of PoS

$$F = (B' + C') \cdot (A + C)$$

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	X
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

			B	
	0	1	X	X
A	1	1	0	1
			C	

End of Chapter 5