KARNAUGH MAPS

- ❖ Karnaugh Maps (K-Maps) with 2, 3, 4, and 5 variables
- ❖ Neighborhood in K-Maps
- ❖ Creating K-Maps of expressions in standard SoP form
- ❖ Simplification of expressions in standard SoP form
- ❖ Grouping cells and deriving simplified terms in K-Maps
- ❖ Creating K-Maps of expressions in standard PoS form
- ❖ Simplification of expressions in standard PoS form using K-Maps
- ❖ Don’t Care Situations

Karnaugh Maps (K-Maps)

Karnaugh-Map, or K-Map in short, is a systematic approach to simplify logic expressions in SoP or PoS form.

K-Maps contain all the combinations of input values and the outputs just like truth tables. In truth tables, we place the binary values of inputs in rows, and we place the output values on the right column of the inputs.

In K-Maps, we place the binary values of inputs in row and column headers, and we place the output values in the cells just like a matrix. Then, we do the simplification by grouping these cells.

Maurice Karnaugh

Karnaugh Maps (K-Maps)

K-Maps are especially suitable for expressions with less variables. We use Quine-McCluskey method to simplify expressions with many variables.

The number of cells in a K-Map is equal to the number of combinations of input variables.

Karnaugh Maps with 2 and 3 Variables

A K-Map with 2 variables has $2^2 = 4$ cells. Each cell denotes a minterm or a maxterm.

A B	0	1
0	A’B’	A’B
1	AB’	AB

A B	0	1
0	m_0	m_1
1	m_2	m_3

B A	0	1
0	m_0	m_2
1	m_1	m_3

A K-Map with 3 variables has $2^3 = 8$ cells. Such a K-Map can be created in 2×4 or 4×2 matrix form.

C AB	0	1
00	A’B’C’	A’B’C
01	A’BC’	A’BC
11	ABC’	ABC
10	AB’C’	AB’C

AB C	00	01	11	10
0	A’B’C’	A’BC’	ABC’	AB’C’
1	A’B’C	A’BC	ABC	AB’C

We place the values of variables AB in 00–01–11–10 order. The reason for this is, we use Gray code numbers where sequential numbers differ only one bit from each other.

Karnaugh Maps with 4 Variables

A K-Map with 4 variables has $2^4 = 16$ cells. Such a K-Map can be created in 4×4 matrix form.

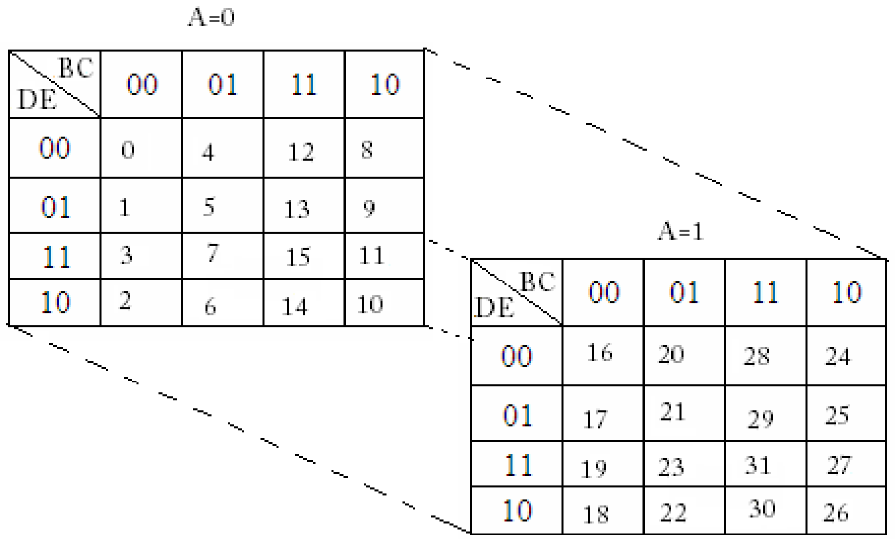
AB \ CD	00	01	11	10
00	A’B’C’D’	A’B’C’D	A’B’CD	A’B’CD’
01	A’BC’D’	A’BC’D	A’BCD	A’BCD’
11	ABC’D’	ABC’D	ABCD	ABCD’
10	AB’C’D’	AB’C’D	AB’CD	AB’CD’

CD \ AB	00	01	11	10
00	A’B’C’D’	A’BC’D’	ABC’D’	AB’C’D’
01	A’B’C’D	A’BC’D	ABC’D	AB’C’D
11	A’B’CD	A’BCD	ABCD	AB’CD
10	A’B’CD’	A’BCD’	ABCD’	AB’CD’

Karnaugh Maps with 5 Variables

A K-Map with 5 variables has $2^5 = 32$ cells. It can e considered as double 4-variable K-Maps. We handle the 5th variable separately.

For F(A,B,C,D,E);



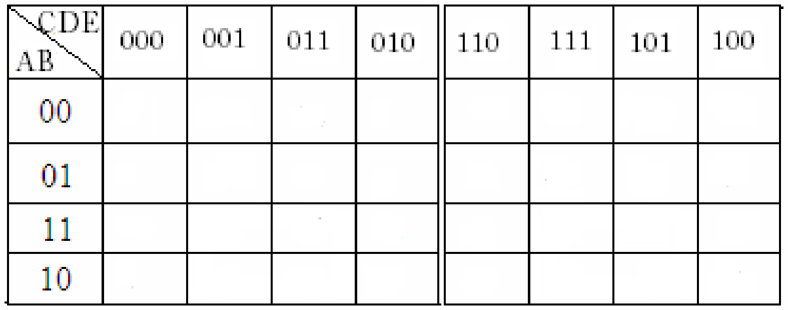
A=0

DE \ BC	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

A=1

DE \ BC	00	01	11	10
00	16	20	28	24
01	17	21	29	25
11	19	23	31	27
10	18	22	30	26

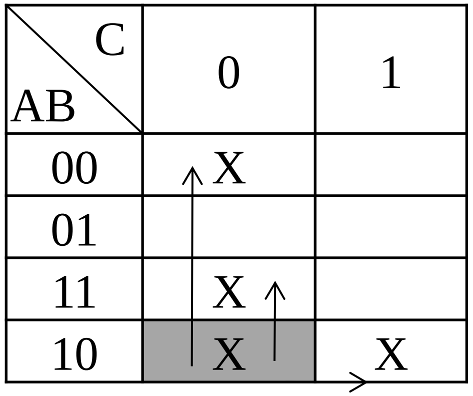
Alternatively, we can create a 5-variable K-Map as depicted below.



AB \ CDE	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Neighborhood in K-Maps

Cells correspond to input values. If input values of two cells differ only 1 bit, then these cells are considered to be neighbors. For example, in a 3-variable K-Map, a cell with input values 100 is a neighbor of the cells with input values 101, 000, and 110. If the difference is 2 bits (e.g. 010 and 001) these cells are not neighbors. Diagonal cells cannot be neighbors because there's no way that they can differ only one bit.



AB \ C	0	1
00	X	
01		
11	X	
10	▨ X	X

Each cell has a neighbor on all sides. Pay attention that, the cell on the bottom and the cell on the top are neighbors in the K-Map given above.

❖ The number of neighbors of a cell is equal to the number of variables.

Creating K-Maps of Expressions in Standard SoP Form

We put a 1 for each term in the logic expression in standard SoP form. We leave other cells empty. If there's no 1's on a K-Map, then the simplified logic expression is equal to 0. Likewise, if all the cells are 1, then the simplified logic expression is equal to 1. We can also create a K-Map from a truth table.

Example: Let's create the K-Map of the logic expression given below

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

The binary input values that make the logic expression's output 1 are;

A'B'C : 001

AB'C : 101

AB'C' : 100

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

Creating K-Maps of Expressions in Standard SoP Form

When we need to create the K-Map of an expression which is not in standard SoP form, then we need to convert it to Standard SoP form by expanding all the terms with missing variables. For example, if we have the term A’B in a 3-variable expression, then we need to expand this term with the missing variable C. The term A’B becomes A’B(C+C’) = A’BC+A’BC’. We learned how to do it in a practical way before.

Example: Let’s create the K-Map of the expression given below.

F(A,B,C)= AB’+A’B’C+B

- The variable C is missing in term AB’.

So; AB’ = AB’C+AB’C’

- The term A’B’C has all the variables.
- The variables A and C are missing in term B.

So; B= A’BC’+A’BC+ABC’+ABC

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

F(A,B,C)= AB’C+AB’C’+A’B’C+A’BC’+A’BC+ABC’+ABC

Simplification of Expressions in Standard SoP form

We do the simplification in 3 steps: Creating groups of 1’s, deriving terms from these groups, and generating the expression using these terms.

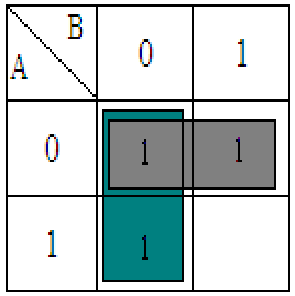
Grouping Rules

- We group only neighbor cells.
- Groups must contain as more cells as possible.
- The number of cells in a group must be the power of 2. Groups can have 1, 2, 4, 8, 16, ... cells.
- Groups must have a rectangular shape.
- All the 1’s on the K-Map must belong to at least 1 group.

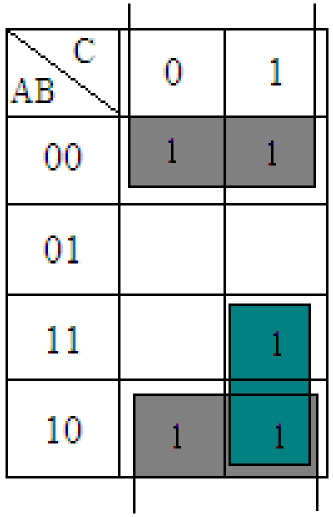
❖ The main goal is to form least, and largest groups.

Grouping Cells and Deriving Simplified Terms in K-Maps

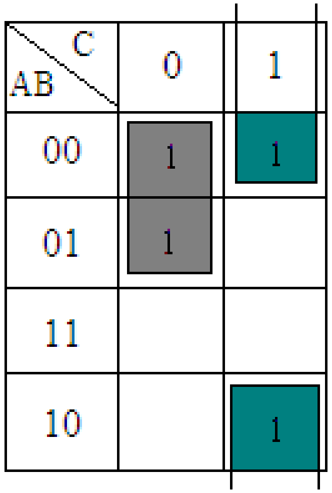
After creating groups, we derive a term from each group. If a group contains both 0 and 1 for a variable, we ignore that variable and use other variables to form the term.



B’+A’

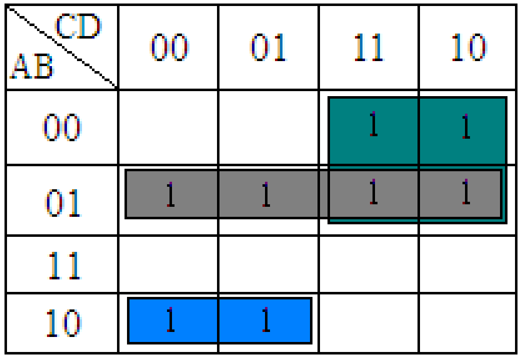


B’+AC



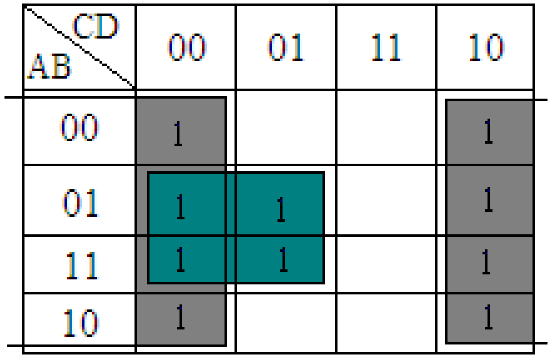
A’C’+B’C

Grouping Cells and Deriving Simplified Terms in K-Maps



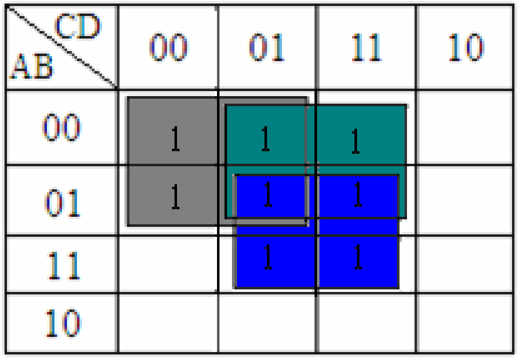
AB \ CD	00	01	11	10
00			1	1
01	1	1	1	1
11				
10	1	1		

A’B+ A’C +AB’C’



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

D’+BC’



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

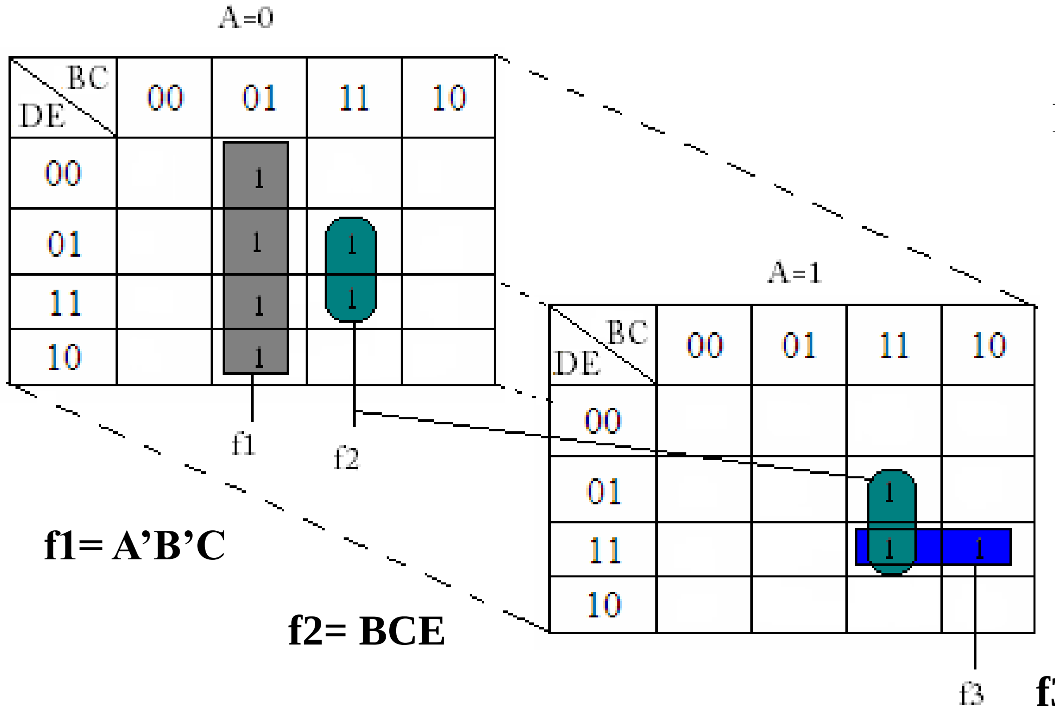
A’C’+A’D+BD

Grouping Cells and Deriving Simplified Terms in K-Maps

Example: Let's simplify the expression given below using K-Map.

F(A,B,C,D,E) = ∑ (4,5,6,7,13,15,27,29,31)

We'll handle the most significant variable (A) separately. First, we place the other variables on the headers of the maps. Then we place the minterms on the map. Overlapping cells are considered to be neighbors.



F=f1+f2+f3= A'B'C+ BCE+ ABDE

f1= A'B'C

f2= BCE

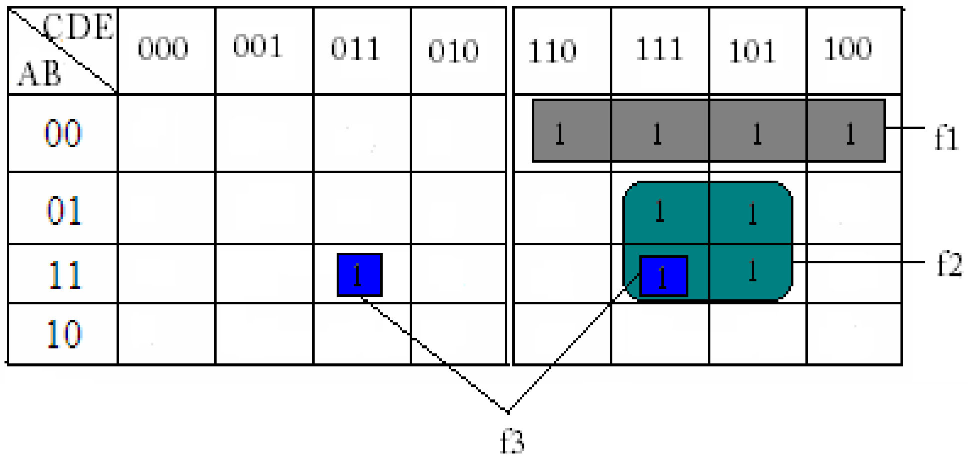
f3=ABDE

Grouping Cells and Deriving Simplified Terms in K-Maps

Example: Let’s simplify the expression given below using K-Map.

$F(A,B,C,D,E) = \sum(4,5,6,7,13,15,27,29,31)$

When grouping, we consider that we can bend the map vertically from the center. Overlapping cells are considered to be neighbors. In 6-variable K-Maps, we can bend the maps both vertically and horizontally. Likewise, overlapping cells are considered to be neighbors.



AB \ CDE	000	001	011	010	110	111	101	100
00					1	1	1	1
01						1	1	
11			1			1	1	
10								

f1 = A’B’C

f2 = BCE

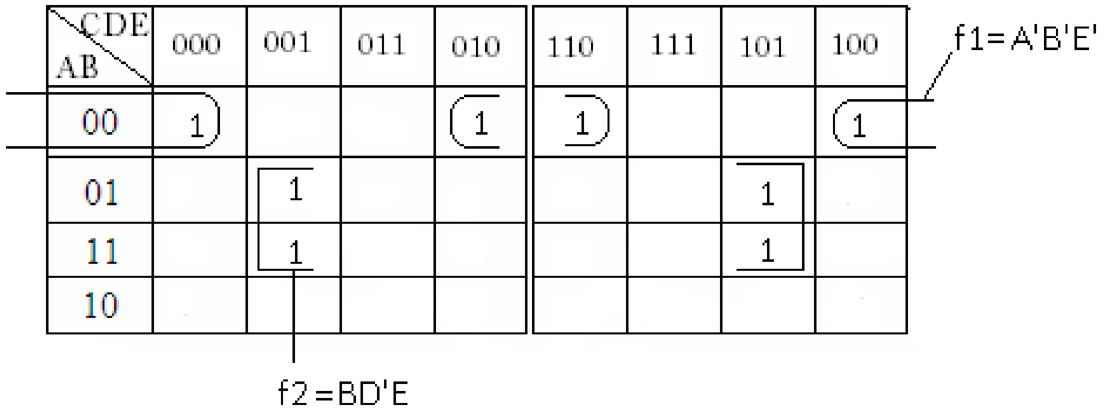
f3 =ABDE

F=f1+f2+f3

Grouping Cells and Deriving Simplified Terms in K-Maps

Example: Let’s simplify the expression given below using K-Map.

F(A,B,C,D,E) = $\sum(0,2,4,6,9,13,25,29)$



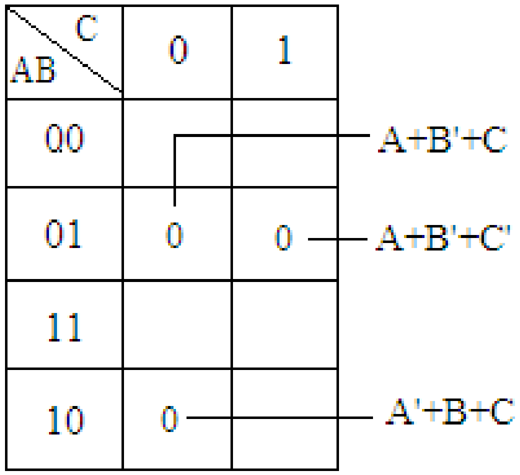
F=f1+f2=A’B’E’+BD’E

Creating K-Maps of Expressions in Standard PoS Form

In PoS form, we place 0's to the cells that correspond to each term. We leave the other cells empty.

For example, in a 3-variable expression, we put 0 to the cell 101 that corresponds to the term (A'+B+C').

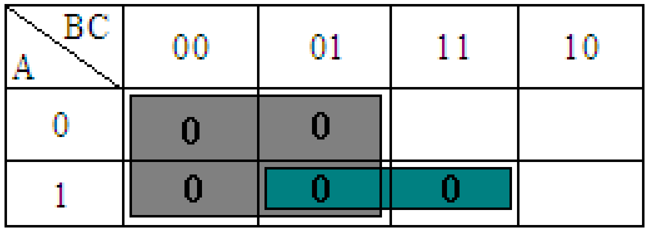
Example: F(A,B,C)= (A'+B+C)(A+B'+C)(A+B'+C')



Simplification of expressions in standard PoS form using K-Maps

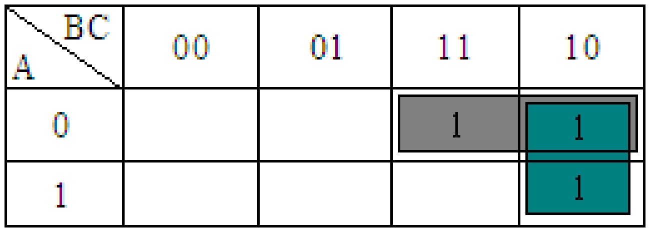
Grouping procedure in PoS form is exactly the same as SoP form, except that we group 0's instead of 1's. After grouping, we derive sum terms from the groups, then form the expression by multiplying all the terms.

Example: If we place SoP terms on a map, the empty cells are actually the PoS terms. If we simplify the PoS expression below, we can see that it will be equal to the SoP expression we derive from the K-Map.



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

B.(A'+C')



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

A'B+BC'

Don’t Care Situations

In some cases, variables may not be fully independent from each other. And in some other cases, it may not be possible for some combinations to occur. These situations are called “don’t care situations”.

For example, there are 6 don’t care situations in BCD code because 1010, 1011, … are invalid BCD codes. A don’t care situation is considered as 1 if it helps simplifying the expression. But if it doesn’t help simplifying the expression, then it is considered as 0.

Example: Consider a system that takes BCD code as input, and outputs 1 only when BCD input is 9 (1001). Let’s simplify this expression.

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

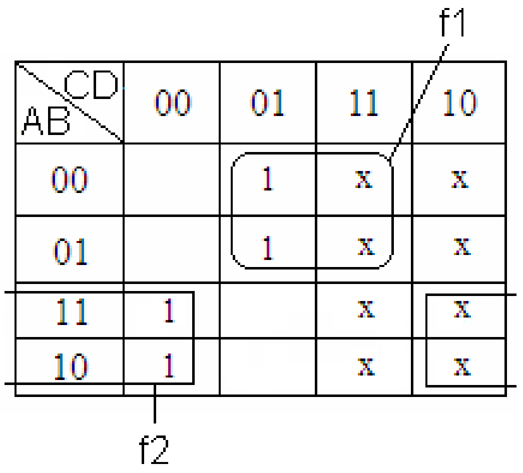
The simplified logic expression will be
f(A,B,C,D) = A.D

Don’t Care Situations

Example: F(A,B,C,D) = ∑(1,5,8,12)

Don’t care situations are F_x(A,B,C,D) = ∑(2,3,6,7,10,11,14,15)

Let’s simplify this expression.



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

f1 = A’.D

f2 = A.D’

F = f1+f2 = A’D+AD’= A⊕D