UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 1

THE MAP METHOD

Introduction of K-Map (Karnaugh Map):-

- Karnaugh map or a K-map refers to a pictorial method that is utilized to minimize various Boolean expressions without using the Boolean algebra theorems along with the equation manipulations.
- A Karnaugh map can be a special version of the truth table. We can easily minimize various expressions that have 2 to 5 variables using a K-map.
- In numerous digital circuits and other practical problems, finding expressions that have minimum variables becomes a prerequisite.
- In such cases, minimization of Boolean expressions is possible that have 3, 4 variables. It can be done using the Karnaugh map without using any theorems of Boolean algebra.
- The K-map can easily take two forms, namely, Sum of Product or SOP and Product of Sum or POS, according to what we need in the problem.
- 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.
- Fill a grid of K-map with 1s and 0s, then solve it by creating various groups.

Solving an Expression Using K-Map:

Here are the steps that are used to solve an expression using the K-map method:

1. Select a K-map according to the total number of variables.

2. Identify maxterms or minterms as given in the problem.

3. For SOP, put the 1's in the blocks of the K-map with respect to the minterms (elsewhere 0's).

4. For POS, put the 0's in the blocks of the K-map with respect to the maxterms (elsewhere 1's).

5. Making rectangular groups that contain the total terms in the power of two, such as 2,4,8 ..(except 1) and trying to cover as many numbers of elements as we can in a single group.

6. From the groups that have been created in step 5, find the product terms and then sum them up for the SOP form.

7. From the groups that have been created in step 5, find the sum terms and then product them up for the POS form.

Mapping 2,3,4&5 variable functions On K-Map:

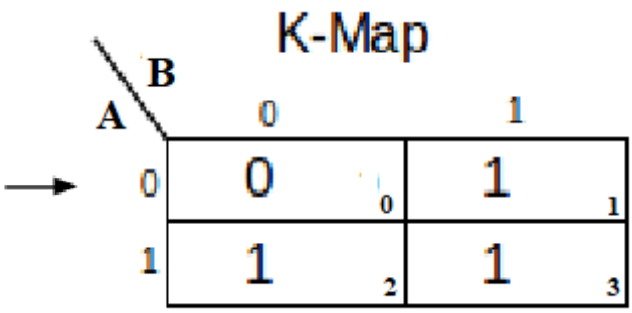
From the given Boolean function or truth table, fill in the K-map with 1s where the function is true (logic 1) and 0s where it's false (logic 0)

2 Variable K-Map:-

A logical specification is often created using a truth table. A truth table is a list of the inputs (A, B) on the left and the corresponding output (F) on the right.

2 Variable Truth Table

	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

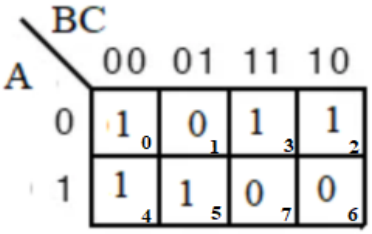


3 Variable K-Map:

A logical specification is often created using a truth table. A truth table is a list of the inputs (A, B, C) on the left and the corresponding output (F) on the right.

3 Variable Truth Table

	A	B	C	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

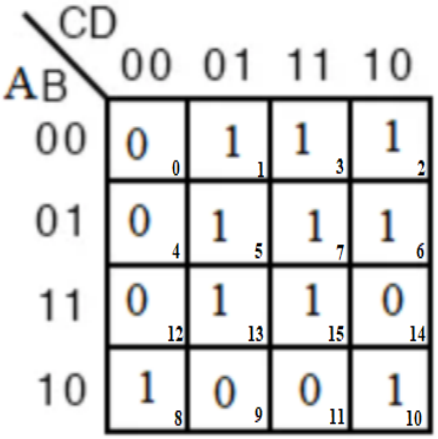


4-Variable Truth Table and K-Map:

A truth table is a list of the inputs (A, B, C,D) on the left and the corresponding output (F) on the right.

4 Variable Truth Table

	A	B	C	D	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
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	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	1

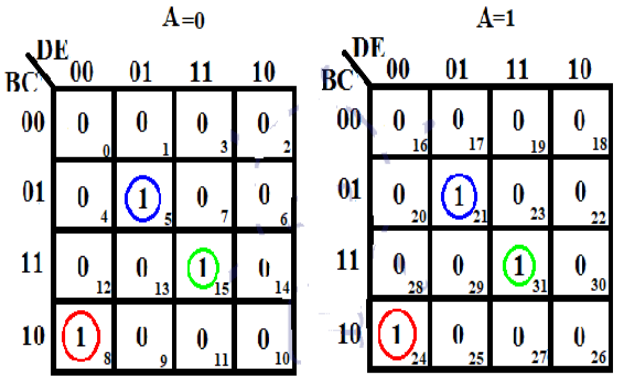


5-Variable Truth Table and K-Map:

A truth table is a list of the inputs (A, B, C,D,E) on the left and the corresponding output (F) on the right.

5 Varaible Truth table

	A	B	C	D	E	F
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	1	1
6	0	0	1	1	0	0
7	0	0	1	1	1	0
8	0	1	0	0	0	1
9	0	1	0	0	1	0
10	0	1	0	1	0	0
11	0	1	0	1	1	0
12	0	1	1	0	0	0
13	0	1	1	0	1	0
14	0	1	1	1	0	0
15	0	1	1	1	1	1
16	1	0	0	0	0	0
17	1	0	0	0	1	0
18	1	0	0	1	0	0
19	1	0	0	1	1	0
20	1	0	1	0	0	0
21	1	0	1	0	1	1
22	1	0	1	1	0	0
23	1	0	1	1	1	0
24	1	1	0	0	0	1
25	1	1	0	0	1	0
26	1	1	0	1	0	0
27	1	1	0	1	1	0
28	1	1	1	0	0	0
29	1	1	1	0	1	0
30	1	1	1	1	0	0
31	1	1	1	1	1	1



Need Of K-Map:-

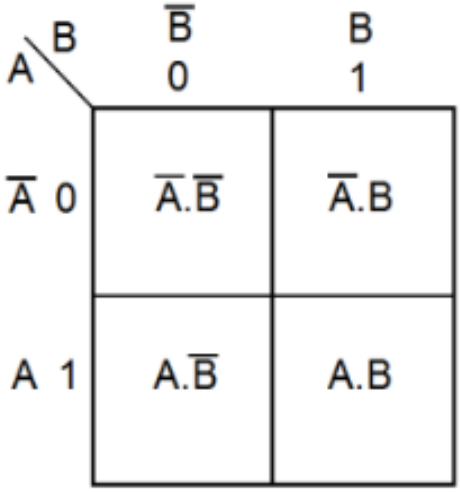
- Karnaugh Map (K-Map) is a graphical method for simplifying Boolean expressions.
- A grid-like diagram used to group and minimize terms.
- Simplification reduces the number of logic gates, minimizes circuit complexity, and improves efficiency.
- Achieve the simplest form of the Boolean expression.

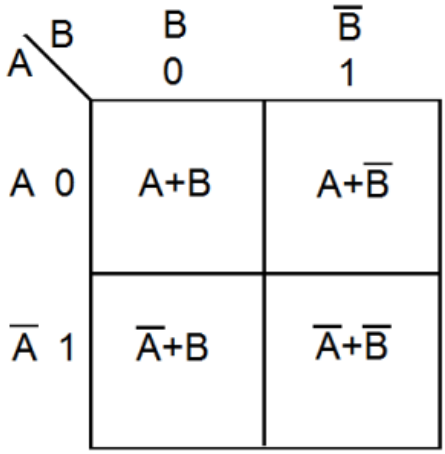
2 Variable K-Map:-

2 variables have $2^n = 2^2 = 4$ minterms. Therefore there are 4 cells (squares) in 2 variable K-map for each minterm.

Consider variable A & B as two variables. The rows of the columns will be represented by variable B.

The square facing the combination of the variable represents that min term as shown in fig below.

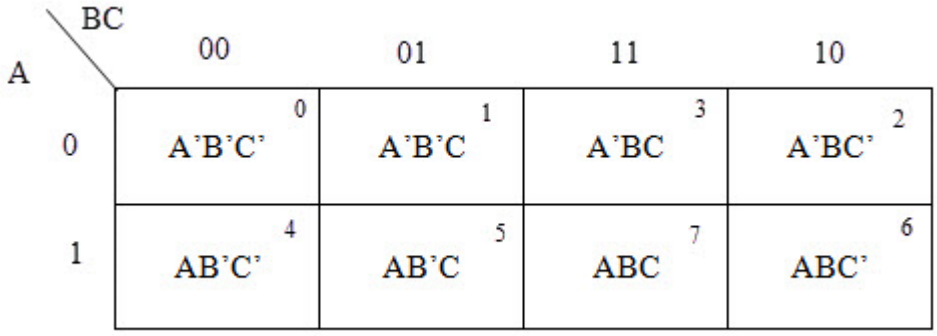




Grouping in 2 variables K-map is easy as there are few squares.

3 Variable K-Map:

3 variables make $2^n=2^3=8$ min terms, so the Karnaugh map of 3 variables will have 8 squares(cells) as shown in the figure given below.



3 variable K-map can be in both forms. Note the combination of two variables in either form is written in Gray code. So the min terms will not be in a decimal order.

The uppermost & lowermost cells are adjacent in the first form of K-map, the leftmost and rightmost cells are also adjacent in the second form of K-map. So they can be made into groups.

Some examples of grouping:

A \ BC	00	01	11	10
0	0 A'B'C'	1 A'B'C	3 A'BC	2 A'BC'
1	4 AB'C'	5 AB'C	7 ABC	6 ABC'

POS FORM:-

A \ B+C	B+C 0 0	B+$\overline{C}$ 0 1	$\overline{B}$+$\overline{C}$ 1 1	$\overline{B}$+C 1 0
A 0	A+B+C 0	A+B+$\overline{C}$ 1	A+$\overline{B}$+$\overline{C}$ 3	A+$\overline{B}$+C 2
$\overline{A}$ 1	$\overline{A}$+B+C 4	$\overline{A}$+B+$\overline{C}$ 5	$\overline{A}$+$\overline{B}$+$\overline{C}$ 7	$\overline{A}$+$\overline{B}$+C 6

4-variable K-Map:

4 variables have $2^n=2^4=16$ minterms. So a 4-variable k-map will have 16 cells as shown in the figure given below.

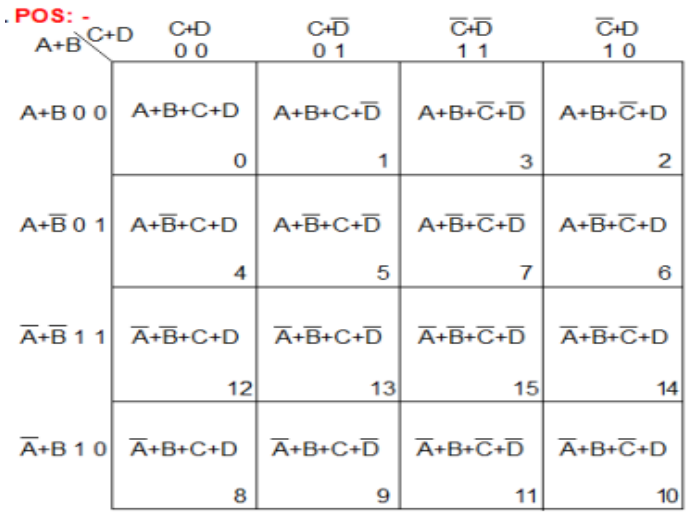
-

AB \ CD	00	01	11	10
00	0 A' B' C' D'	1 A' B' C' D	3 A' B' C D	2 A' B' C D'
01	4 A' B C' D'	5 A' B C' D	7 A' B C D	6 A' B C D'
11	12 A B C' D'	13 A B C' D	15 A B C D	14 A B C D'
10	8 A B' C' D'	9 A B' C' D	11 A B' C D	10 A B' C D'

Each cell (min term) represent the variables in front of the corresponding row & column.

The variables are in gray code (1-bit change). The four cells of the corner are adjacent to each other as there is a1-bit difference even if they are not touching physically. So they can be grouped together.

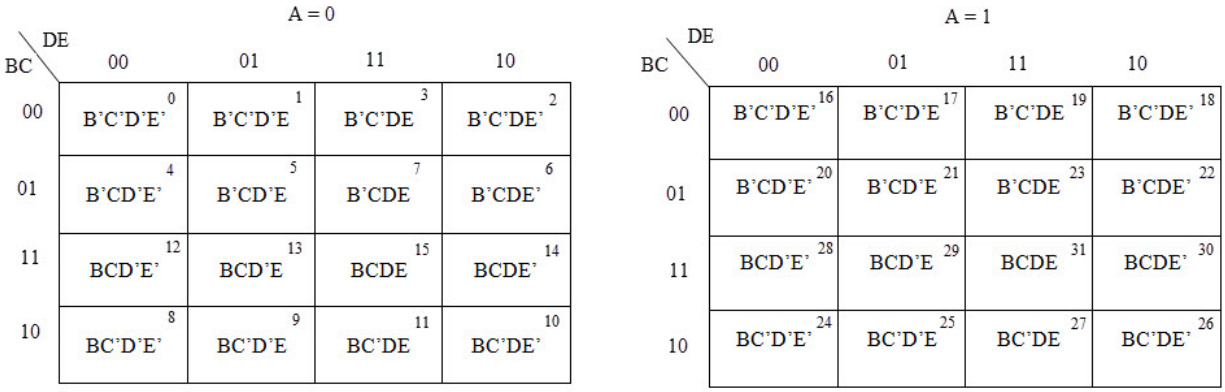
POS FORM:-



5 Variables K-Map:

5 variables have 32 min terms,which mean 5 variable karnaugh map has 32 squares (cells).

A 5-variable K-map is made using two 4-variable K-maps. Consider 5 variables A,B,C,D,E. their 5 variable K-map is given below.



These both 4-variable Karnaugh map together represents a 5-variable K-map for variable A,B,C,D,E.

Notice variable A over the top of each 4-variable K-map. For A=0, the left K-map is selected and right map for A = 1.

Each corresponding squares (cells) of these 2 4-variable K-maps are adjacent. Visualize these both K-maps on top of each other. m_0 is adjacent to m_{16}, so is m_1 to m_{17} so on until the last square.

The rule (method) of grouping is same for each of the 4-variable k-maps. However, you also need to check the corresponding cells in both K-maps as well

POS FORM:-

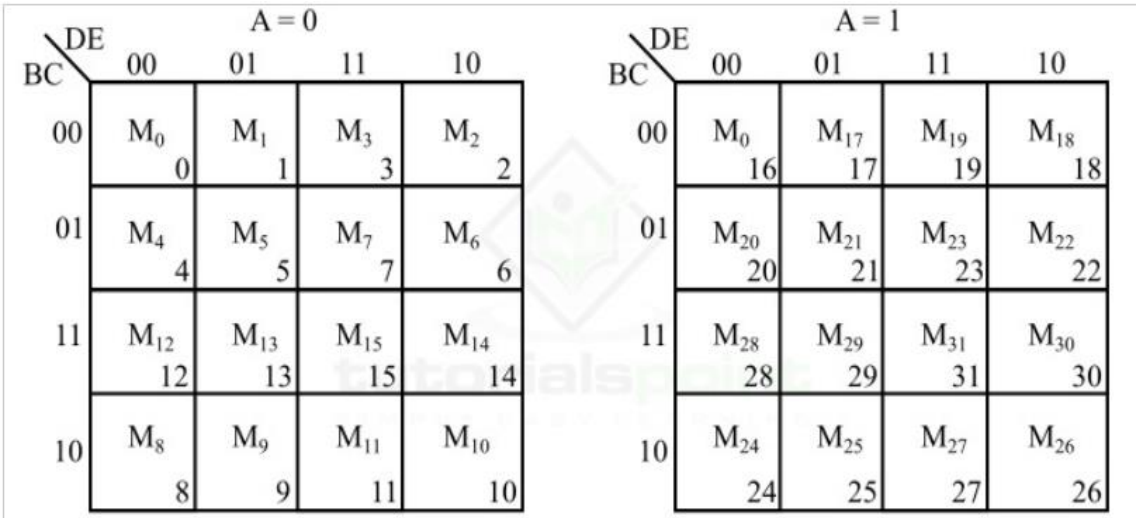


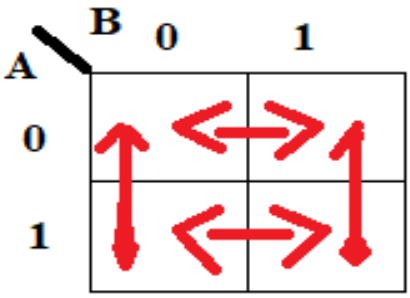
Figure 2 - 5 Variable POS K-Map

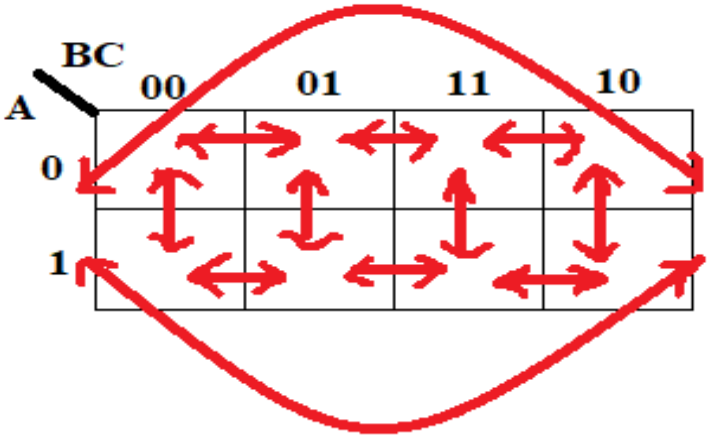
Cell Adjacency:

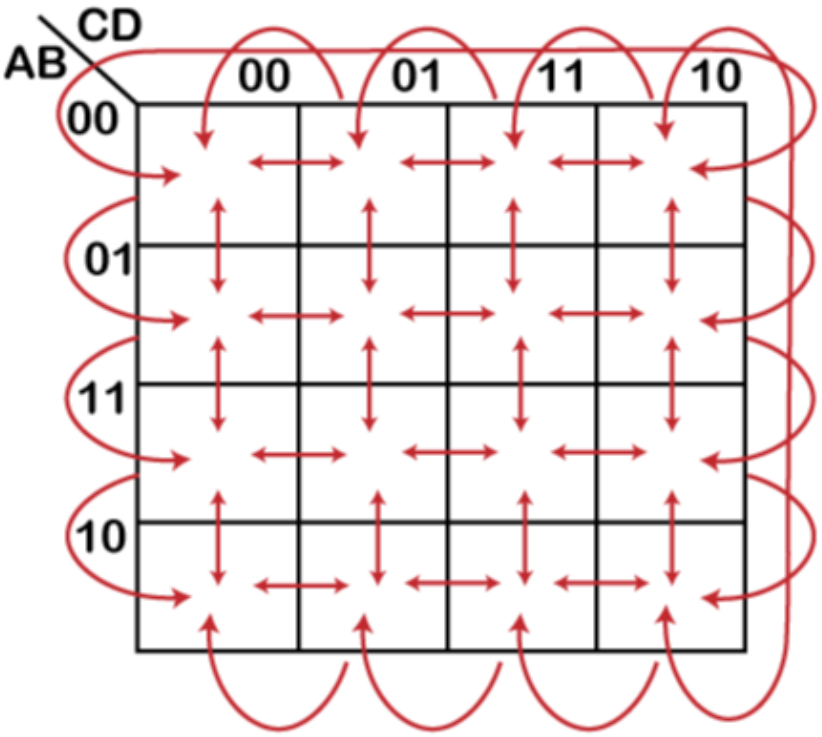
The cells in a k-map are arranged so that there is only a single- variable change between adjacency cells.

Adjacency is defined by a single variable change.

Cells with values that differ by more than one variable are not adjacent.

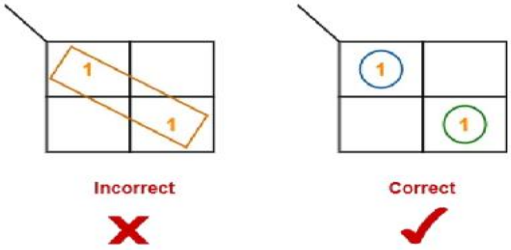




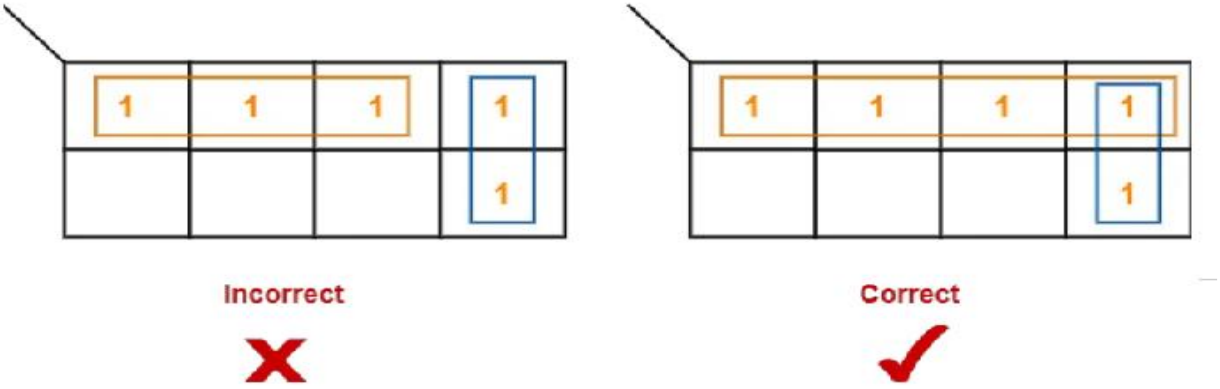


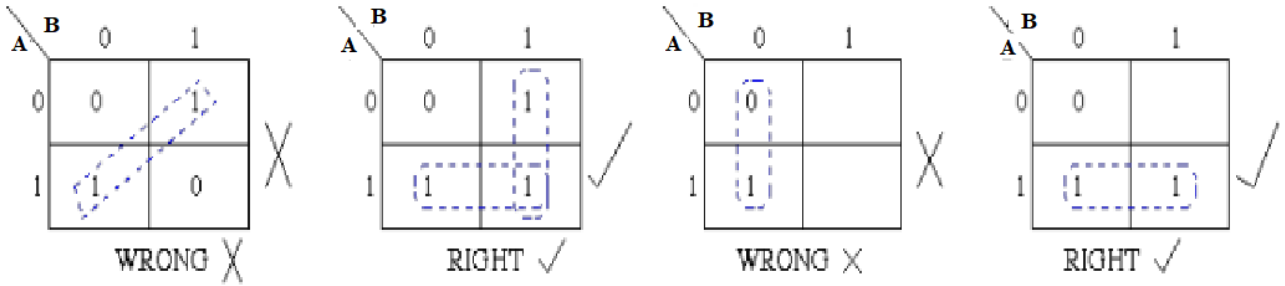
Do's and Dont's of K-Map:

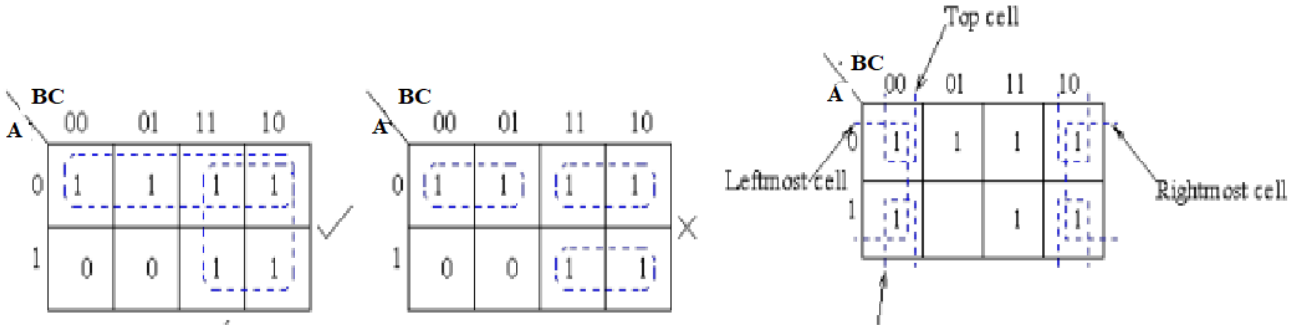
- We can not create groups of diagonal or any other shape.

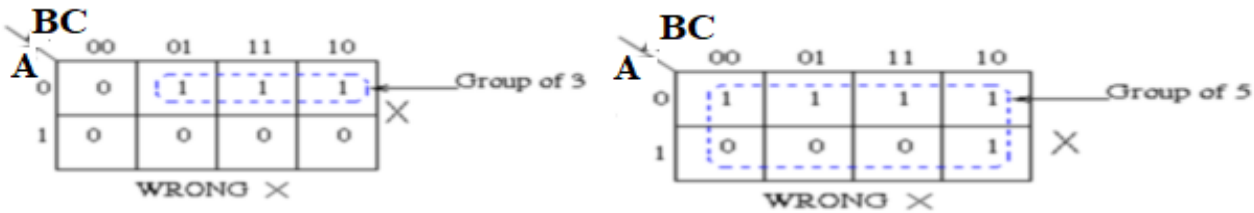


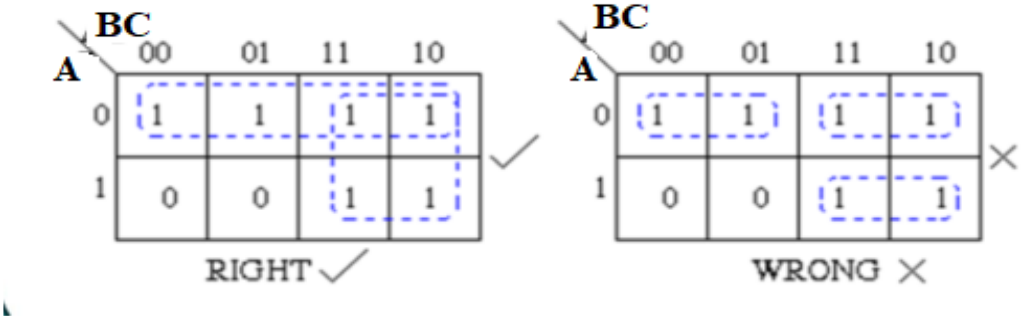
- We can only create a group whose number of cells can be represented in the power of 2.
- In other words a group can only contain 2n i.e,1,2,4,8,16 and so on number of cells.



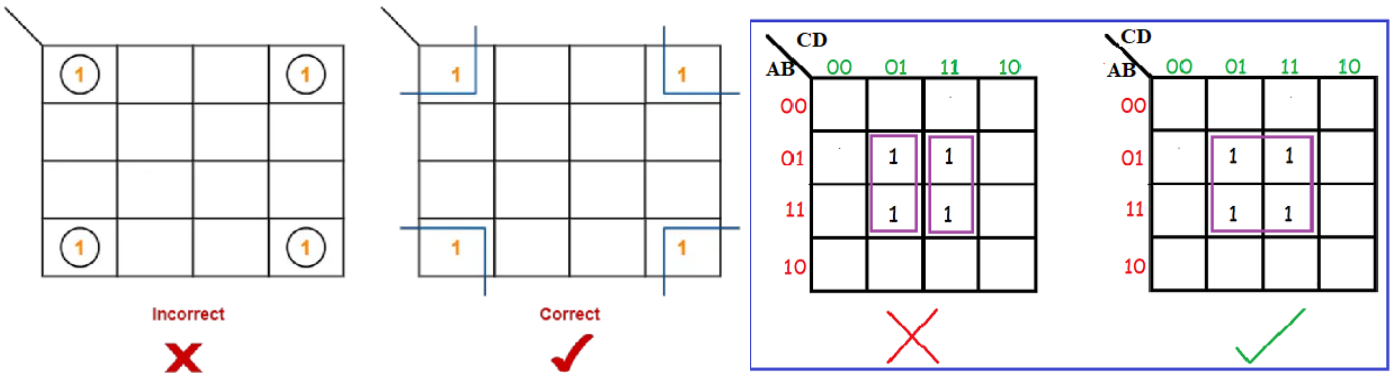








- Opposite grouping and corner grouping are allowed.



UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 3

TWO AND THREE -VARIABLE FUNCTIONS MINIMIZATION USING K-MAP METHOD

Steps to minimize functions using K-Map :

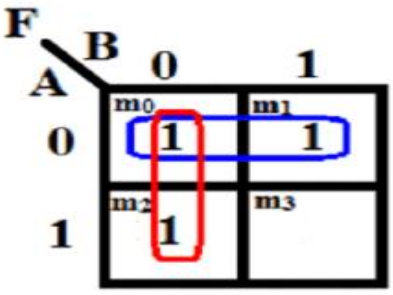
- Select the K-map according to the number of variables.
- Identify minterms or maxterms as given in the problem.
- For SSOP put 1's in blocks of K-map respective to the minterms .
- For SPOS put 0's in blocks of K-map respective to the max terms .
- Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
- Group adjacent 1's / 0's into pairs, quads, or octets.
- Derive the simplified Boolean expression.

Simplification Of 2-Varible SSOP expressions using K-MAP method:

Example 1 (Two variables):

Simplify the following function using K-Map method F = ∑ (m0, m1, m2)

Sol:



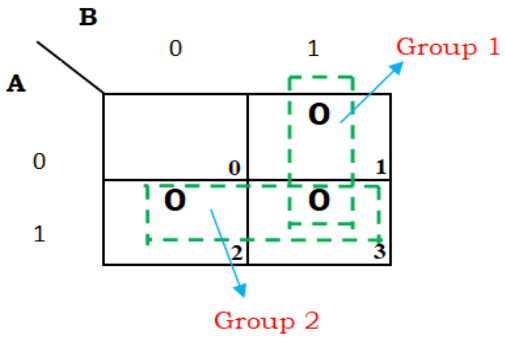
- We made 2 groups of 1's. each group contains 2 minterms.
- In the first group, variable A is changing & B remains unchanged. So the first term of the output expression will be $\overline{B}$ (because B = 0 in this group).
- In the 2nd group, Variable B is changing and variable A remains unchanged. So the second term of the output expression will be $\overline{A}$ (because A=0 in this group).
- Now the simplified expression will be the sum of these two terms as given below,
- **$F = \overline{A} + \overline{B}$**

Simplification of 2-variable SPOS expressions using K-MAP method:

Example 2 (Two variables):

Simplify the following function using K-Map method f(A,B) =πM(1,2,3)

Sol:



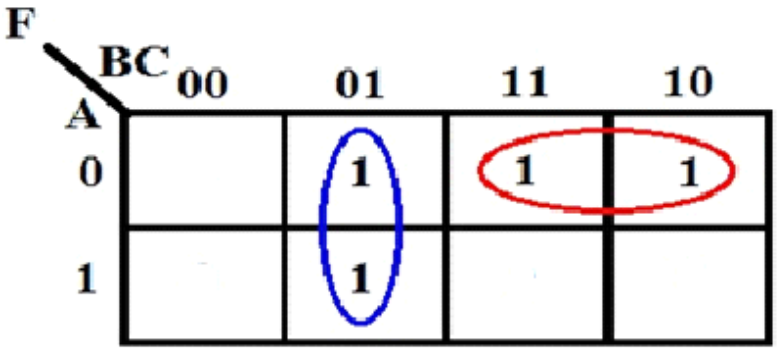
- In group 1, the cells 1 and 3 are grouped and the common term is B.
- In group 2, cells 2 and 3 are grouped and the common term is A.
- So the minimized POS expression is **f(A,B) =B'.A'**

Simplification of 3-variable SSOP expressions using K-MAP method:

Example 1 (Three variables):

Simplify the given 3-variable Boolean equation by using k-map F1 = ∑ (m1, m2,m3,m5)

Sol:

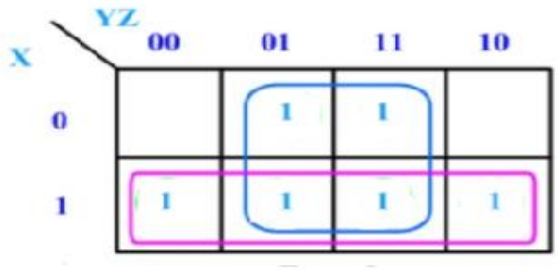


- We made 2 groups of 1's. each group contains 2 minterms.
- In the first group, variable A is changing & B, C remains unchanged. So the first term of the output expression will be B'C(because B = 0 and C=1 in this group).
- In the 2nd group, Variable C is changing and variable A & B remains unchanged. So the second term of the output expression will be A'B (because A=0 & B=1 in this group).
- Now the simplified expression will be the sum of these two product terms as given below,
- **F = B'C+A'B**

Example 2 (Three variables):

Simplify the given 3-variable Boolean equation by using k-map F2(X,Y,Z) = m1 + m3 + m4+m5+m6 + m7

Sol:



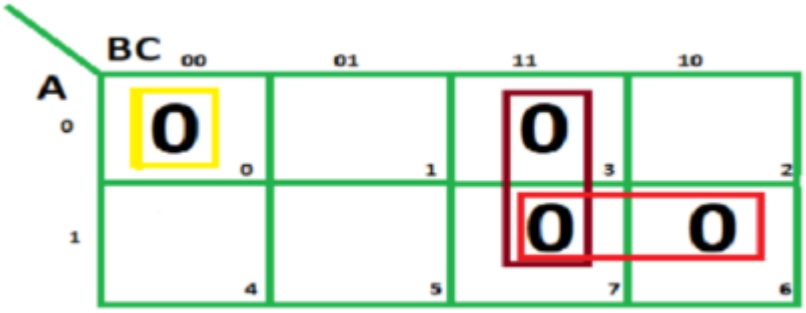
- We made 2 groups of 1's. each group contains 4 minterms.
- In the first group, variable X & Y are changing & Z remains unchanged. So the first term of the output expression will be Z (Z=1 in this group).
- In the 2nd group, Y & Z are changing & X remains unchanged. So the second term of the output expression will be X (because X=1 in this group).
- Now the simplified expression will be the sum of these two product terms as given below,
- **F = Z+X**

Simplification of 3-variable SPOS expressions using K-MAP method:

Example 1 (Three variables):

Simplify the given 3-variable Boolean equation by using k-map F1(A,B,C) =πM(0,3,6,7)

Sol:



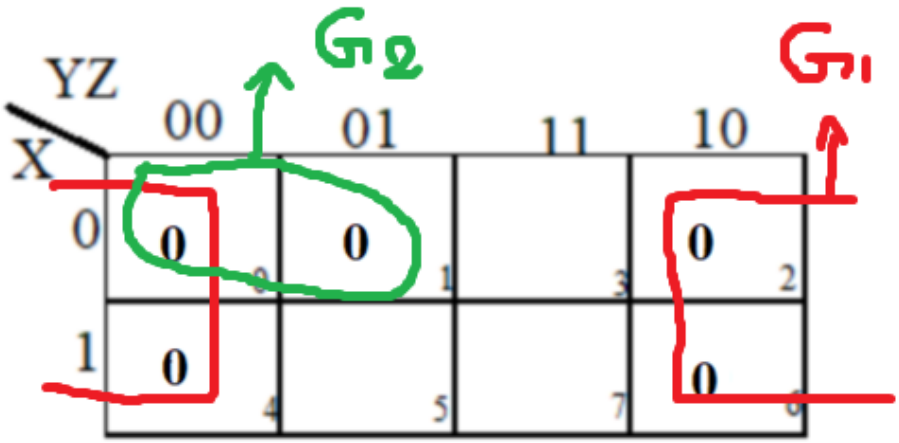
- We made 3 groups of 0's. two groups contain 2 maxterms and the third group is a isolated maxterm
- In the first group, variable A is changing & B,C remains unchanged. So the first sum term of the output expression will be (B' + C') (B=1 &C=1 in this group).
- In the second group, variable C is changing & A,B remains unchanged. So the second sum term of the output expression will be (A' + B') (A=1 & B=1 in this group).

- In third group, maxterm 0 (M_0) does not have adjacency so it is treated as isolated maxterm . So the third sum term of the output expression will be (A + B+C) (A=0,B=0 & C=0 in this group).
- Now the simplified expression will be the product of these three sum terms as given below,
- **F = (B' + C')(A' + B') (A + B + C)**

Example 2 (Three variables):

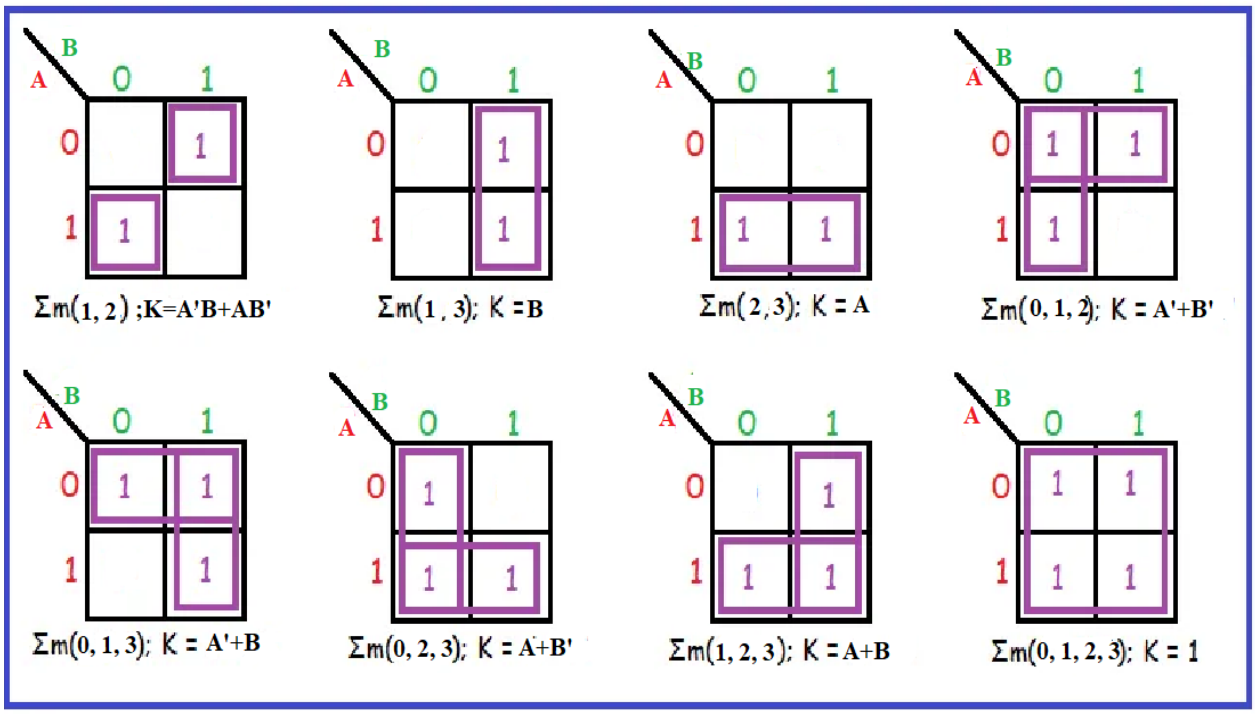
Simplify the given 3-variable Boolean equation by using k-map F2(X,Y,Z) =πM(0,1,2,4,6)

Sol:

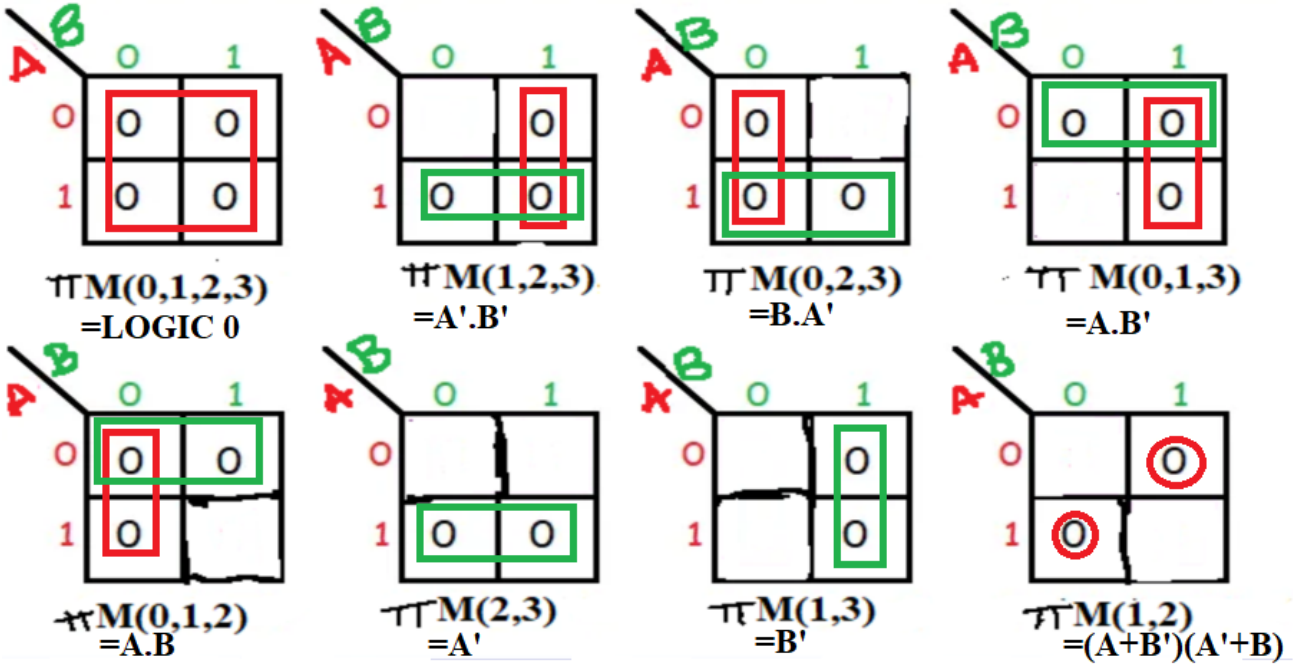


- We made 2 groups of 0's. one group contain 4 maxterms and the second group is ontain 2 maxterms .
- In the first group, variable X &Y are changing & Z remains unchanged. So the first sum term of the output expression will be (Z) (Z=0 in this group).
- In the second group, variable Z is changing & X,Y remains unchanged. So the second sum term of the output expression will be (X+ Y) (X=0 & Y=0 in this group).
- Now the simplified expression will be the product of these two sum terms as given below,
- **F = (Z)(X+Y)**

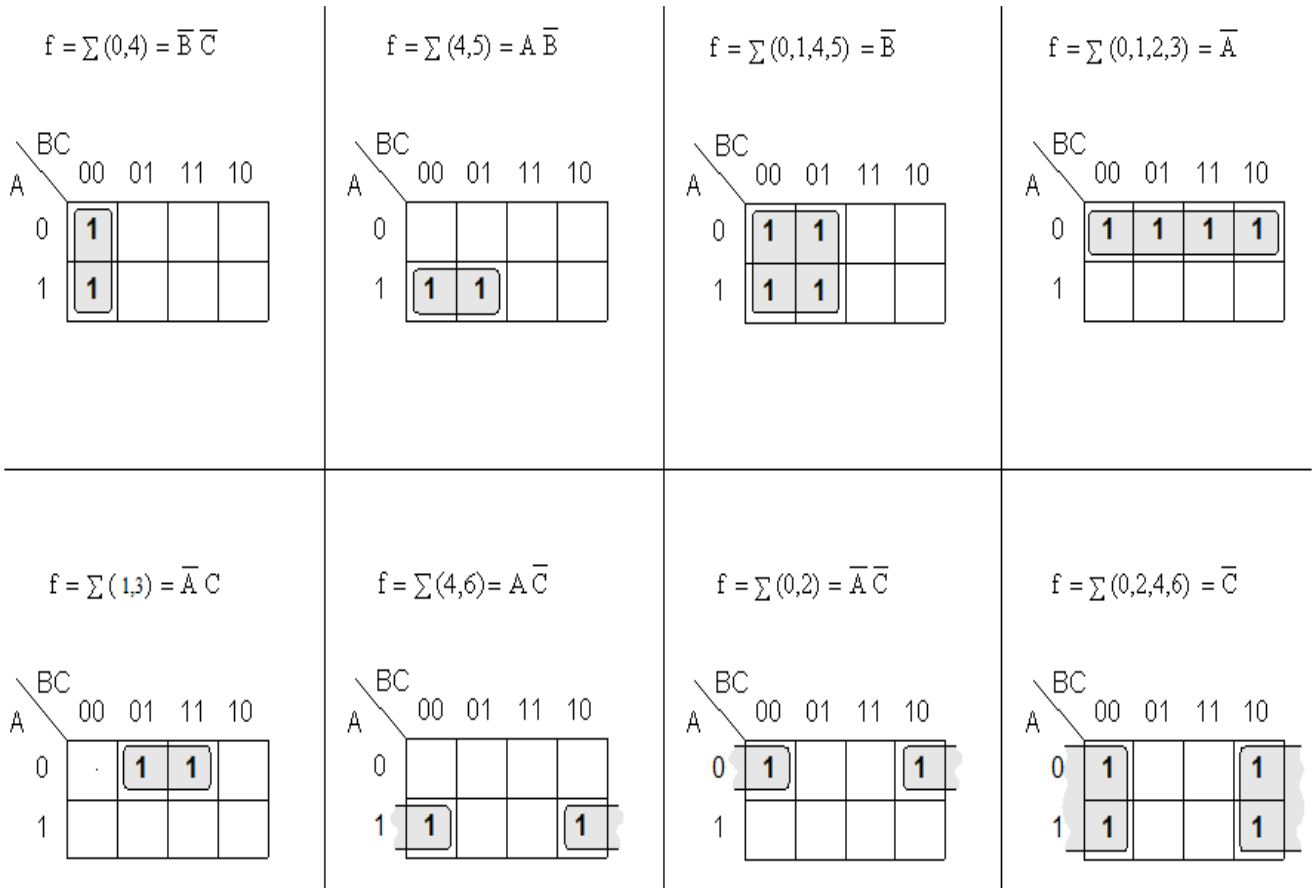
Practice Examples Of 2-Varible SSOP Functions using K-Map method:



Practice Examples Of 2-Varible SPOS Functions using K-Map method:



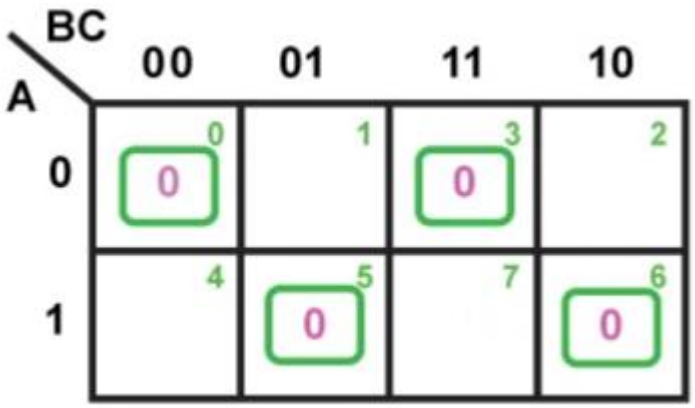
Practice Examples Of 3-Varible SSOP Functions using K-Map method:



Practice Examples Of 3-Varible SPOS Functions using K-Map method:

1. F1(A,B,C) =πM(0,3,5,6)

Sol:

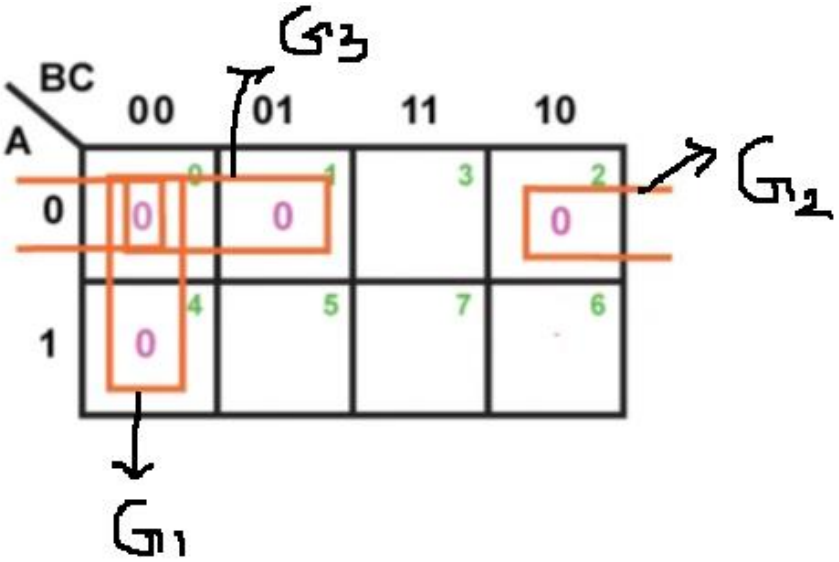


No cell adjacency since they are mapped diagonally hence are maxterms are treated as **Isolated Maxterms.**

F1=(A+B+C)(A+B'+C')(A'+B+C')(A'+B'+C)

2. F2(A,B,C) =πM(0,1,2,4)

Sol:



F2(A,B,C)=(B+C)(A+C)(A+B)

UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 4

FOUR -VARIABLE SSOP SIMPLIFICATION USING K-MAP

Steps for K-Map Simplification:

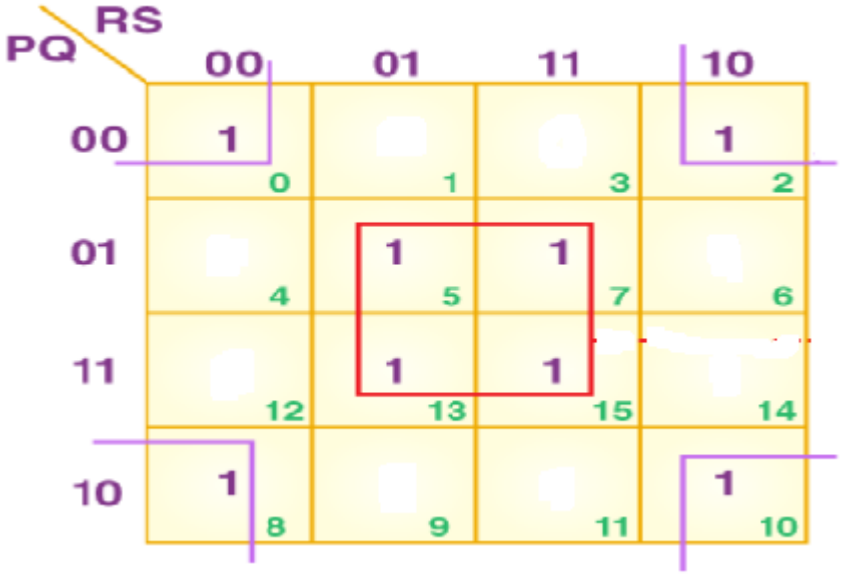
- Select the K-map according to the number of variables.
- Identify minterms as given in the problem.
- For SOP put 1's in blocks of K-map respective to the minterms .
- Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
- Group adjacent 1's into pairs, quads, or octets.
- Derive the simplified Boolean expression.

Simplification Of 4-Varible SSOP expressions using K-MAP method:

Example 1 (Four variables):

Simplify the following function using K-Map method F (P,Q,R,S) = ∑m(0, 2, 5, 7, 8, 10, 13, 15)

Sol:

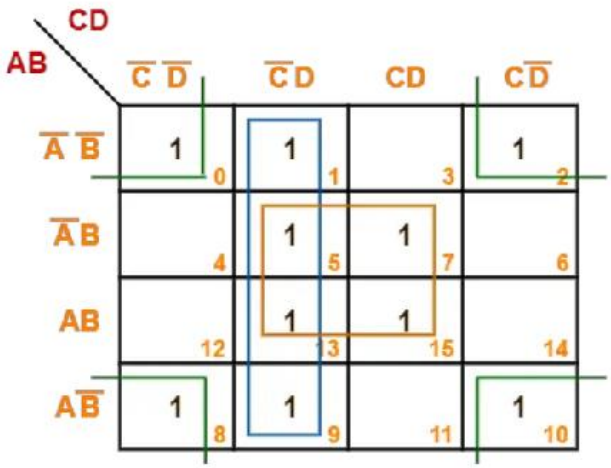


- From the red group, the product term would be —QS
- From group2, the product term would be —Q'S'
- If we sum these product terms, then we will get the final expression **F= (QS +Q'S')**

Example 2 (Four variables):

Simplify the following function using K-Map method F (A, B, C, D) = ∑m(0,1, 2, 5, 7, 8, 9,10, 13, 15)

Sol:



From the orange group, the product term would be —BD

From the green group, the product term would be —B'D'

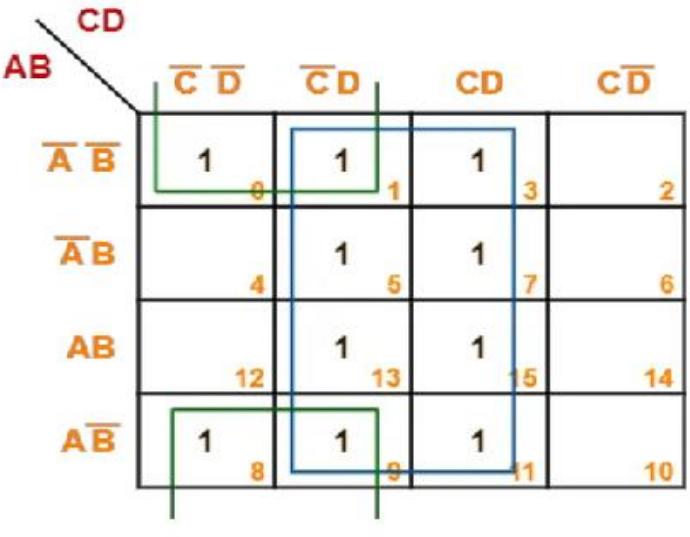
From the blue group, the product term would be —C'D

If we sum these product terms, then we will get this final expression **(BD +B'D'+C'D)**

Example 3 (Four variables):

Simplify the following function using K-Map method F (A, B, C, D) = ∑m(0,1, 3,5,7,8,9,11,13, 15)

Sol:



From the blue group, the product term would be —D

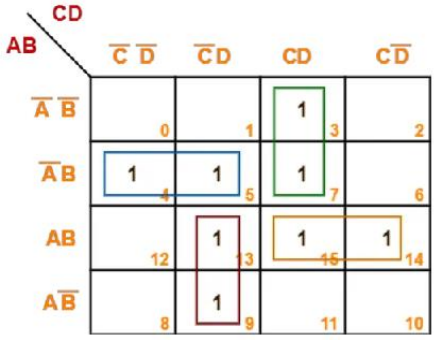
From the green group, the product term would be —B'C'

If we sum these product terms, then we will get this final expression **(D+B'C')**

Example 4 (Four variables):

Simplify the following function using K-Map method F (A, B, C, D) = ∑m(3,4,5,7,9,13,14, 15)

Sol:



From the blue group, the product term would be —A'BC'

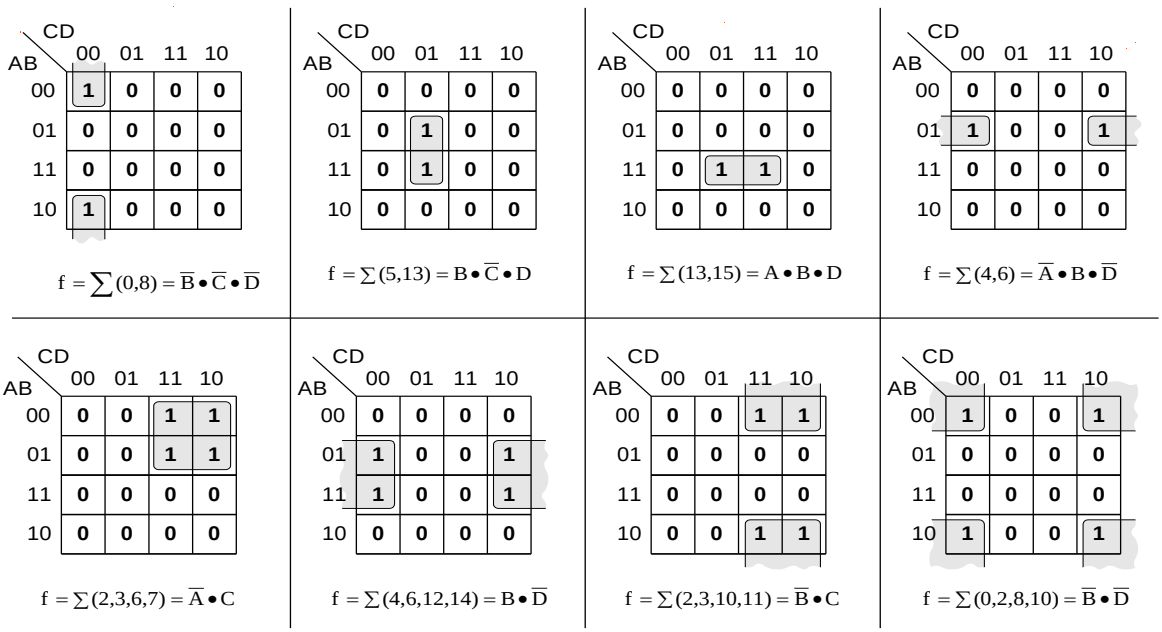
From the green group, the product term would be —A'CD

From the orange group, the product term would be —ABC

From the purple group, the product term would be —AC'D

If we sum these product terms, then we will get this final expression
F(A,B,C,D)=A'BC'+A'CD+ABC+AC'D

Practice Examples on 4-Variable SSOP functions:



UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 6

FIVE -VARIABLE SSOP SIMPLIFICATION

Five Variable K-Map:

The following are the important characteristics of a 5 variable K-map

- A five variable K-map have 32 (2^5) cells or squares, and each cell of the K-map represents either a minterm or a maxterm of the Boolean expression.
- If the given Boolean function is expressed in SOP (Sum of Products) form, then the minterms of five variables Boolean function are designated as $m_0, m_1, m_2, m_3 \ldots m_{31}$. Where, m_0 is corresponding to $(\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E})$, m_1 is corresponding to $(\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}E)$,... and m_{31} is corresponding to $(ABCDE)$.
- On the other hand, if the 5 variable Boolean function is expressed in POS (Product of Sums) form, then the maxterms of the function are designated as M_0, M_1, M_2,... M_{31}. Where, M0 represents $(A+B+C+D+E)$, M_1 represents $(A+B+C+D+\overline{E})$,... and M_{31} represents $(\overline{A}+\overline{B}+\overline{C}+\overline{D}+\overline{E})$.

Example 1:

Reduce the following 5 variable Boolean function in SOP form using the five variable K map.
F(A, B, C, D, E□=∑m(0,1,2,4,7,8,12,14,15,16,17,18,20,24,28,30,31□

Solution

The SOP K-map representation of the given SOP Boolean function is shown in Figure.

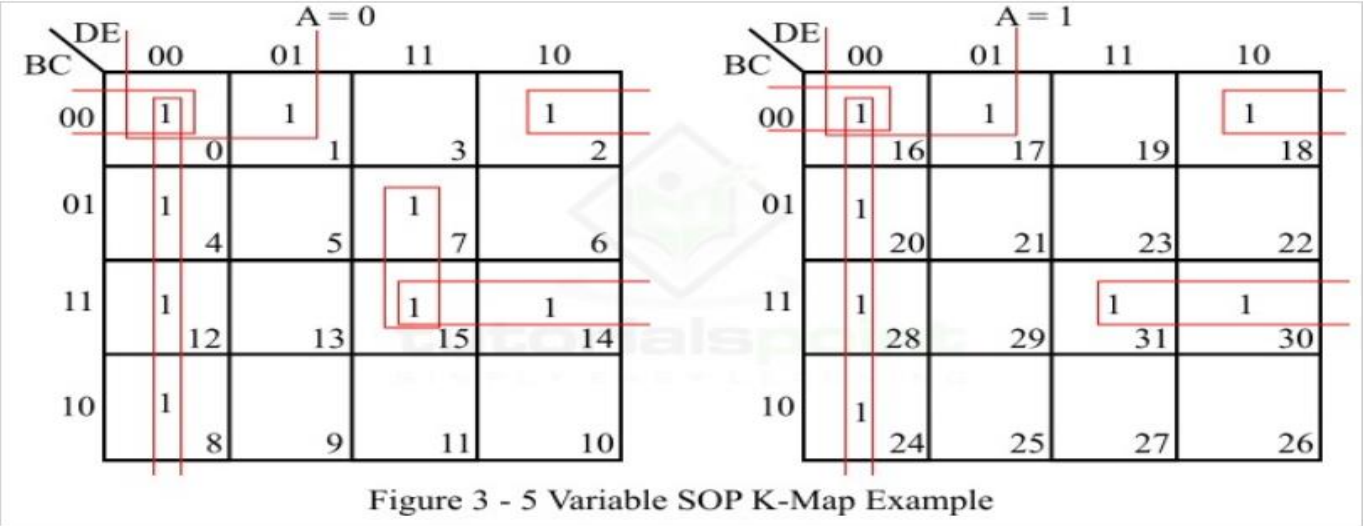


Figure 3 - 5 Variable SOP K-Map Example

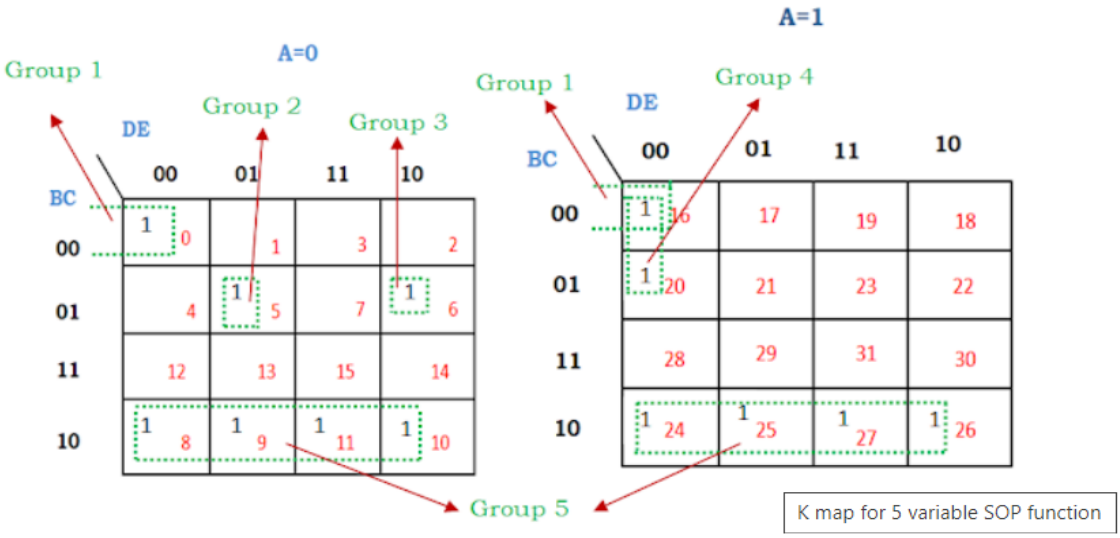
- There are no isolated 1s in the K-map.
- The minterm m_0 can form an 8-square with m_4, m_8, m_{12}, m_{16}, m_{20}, m_{24}, and m_{28}. So make it and read it as $(\overline{D}\,\overline{E})$.
- The minterms m_0, m_1, m_{16}, and m_{17} form a 4-square. Make it and read it as $(\overline{B}\,\overline{C}\,\overline{D})$.
- The minterms m_0, m_2, m_{16}, and m_{18} form a 4-square. Make it and read it as $(\overline{B}\,\overline{C}\,\overline{E})$.
- The minterms m_7 and m_{15} form a 2-square. Make it and read it as $(\overline{A}CDE)$.
- The minterms m_{14}, m_{15}, m_{30} and m_{31} form a 4-square. Make it and read it as (BCD).
- Finally, write all the product terms in SOP form.

Hence, the minimal SOP expression of the given 5 variable Boolean function is,

$$f(A,B,C,D,E) = \overline{A}CDE + \overline{B}\,\overline{C}\,\overline{D} + \overline{B}\,\overline{C}\,\overline{E} + BCD + \overline{D}\,\overline{E}$$

Example 2:

Minimize the following 5 variable SOP function using K map:
F(A,B,C,D,E)= ∑m(0,5,6,8,9,10,11,16,20,24,25,26,27)
Sol:



K map for 5 variable SOP function

- First fill the cells which are given in the SOP function with '1'.

- Group the adjacent cells in group of 1, 2 and 4.
- There are five groups in this K map.

Group 1: Cells 0 ,8,16 and 24 are grouped. The common term is C'D'E'.

Group 2: Cell number 5 is in group 2 and the Minterm for this group is A'B’CD’E.

Group 3: Cell number 6 is in group 3 and the Minterm for this group is A'B’CDE’.

Group 4: Cells 16 and 20 are grouped. The common term is AB'D'E'.

Group 5: Cells 8, 9, 10, 11, 24, 25, 26 and 27 are grouped. The common term is BC'

So the minimized SOP expression is

F(A,B,C,D,E)= C'D'E'+ A'B’CD’E+ A'B’CDE’+ AB'D'E' +BC'

UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 7

FIVE -VARIABLE SPOS SIMPLIFICATION

Five Variable K-Map:

The following are the important characteristics of a 5 variable K-map

- A five variable K-map have 32 (2^5) cells or squares, and each cell of the K-map represents either a minterm or a maxterm of the Boolean expression.
- If the given Boolean function is expressed in SOP (Sum of Products) form, then the minterms of five variables Boolean function are designated as $m_0, m_1, m_2, m_3 \ldots m_{31}$. Where, m_0 is corresponding to $(\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E})$, m_1 is corresponding to $(\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}E)$,... and m_{31} is corresponding to $(ABCDE)$.
- On the other hand, if the 5 variable Boolean function is expressed in POS (Product of Sums) form, then the maxterms of the function are designated as M_0, M_1, M_2,... M_{31}. Where, M0 represents $(A+B+C+D+E)$, M_1 represents $(A+B+C+D+\overline{E})$,... and M_{31} represents $(\overline{A}+\overline{B}+\overline{C}+\overline{D}+\overline{E})$.

Example 1:

Minimize the following 5 variable Boolean function in POS form using the five variable K map.
F(A,B,C,D,E□=∏M{ 3,5,6,9,10,11,13,19,21,22,23,25,26,27,29□Solution

The POS K-map representation of the given SPOS Boolean function is shown in Figure.

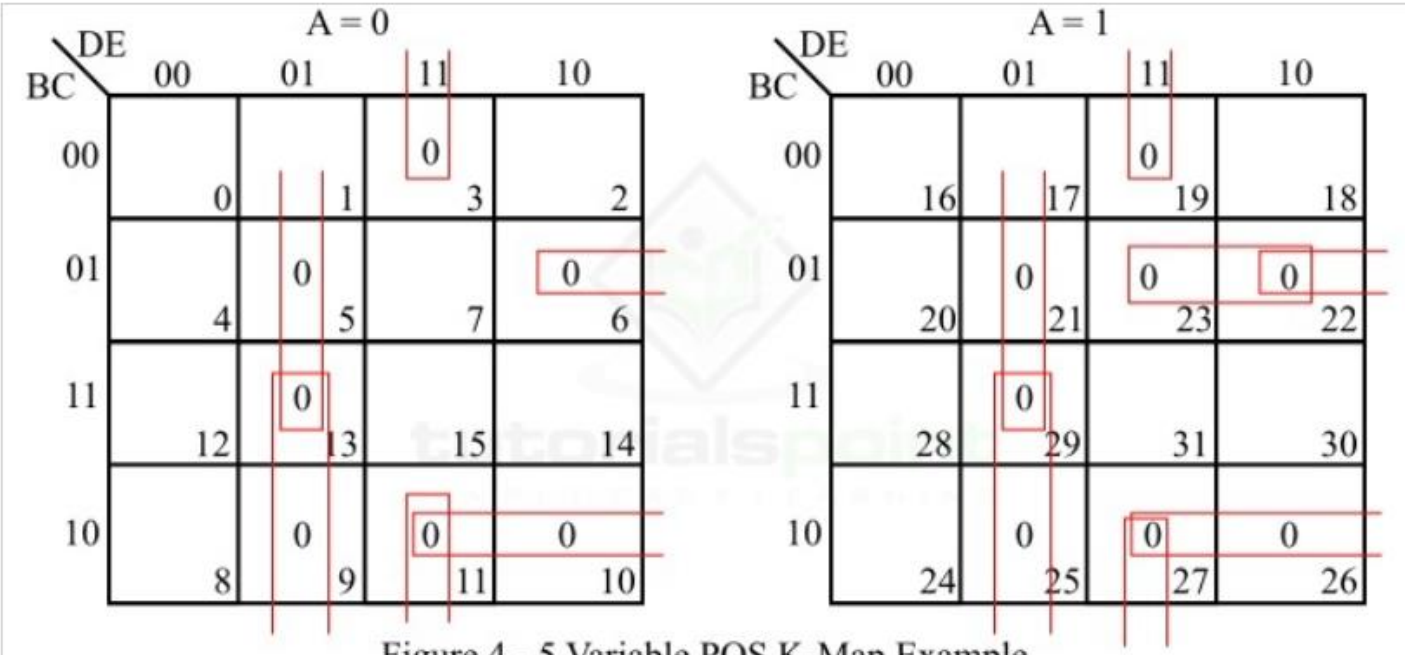


Figure 4 - 5 Variable POS K-Map Example

- There are no isolated zeros in the K-map.
- The maxterms M_9, M_{13}, M_{25}, and M_{29} form a 4 – square. Make it and read it as $(\overline{B} + D + \overline{E})$.
- The maxterms M_3, M_{11}, M_{19}, and M_{27} form a 4 – square. Make it and read it as $(C + \overline{D} + \overline{E})$.
- The maxterms M_5, M_{13}, M_{21}, and M_{29} form a 4 – square. Make it and read it as $(\overline{C} + D + \overline{E})$.
- The maxterms M_6 and M_{22} form a 2 – square. Make it and read it as $(B + \overline{C} + \overline{D} + E)$.
- The maxterms M_{10}, M_{11}, M_{26}, and M_{27} form 4 – square. Make it and read it as $(\overline{B} + C + \overline{D})$.
- The maxterms M_{22} and M_{23} form 2 – square. Make it and read it as $(\overline{A} + B + \overline{C} + \overline{D})$.
- Finally, write all the sum terms in POS form.

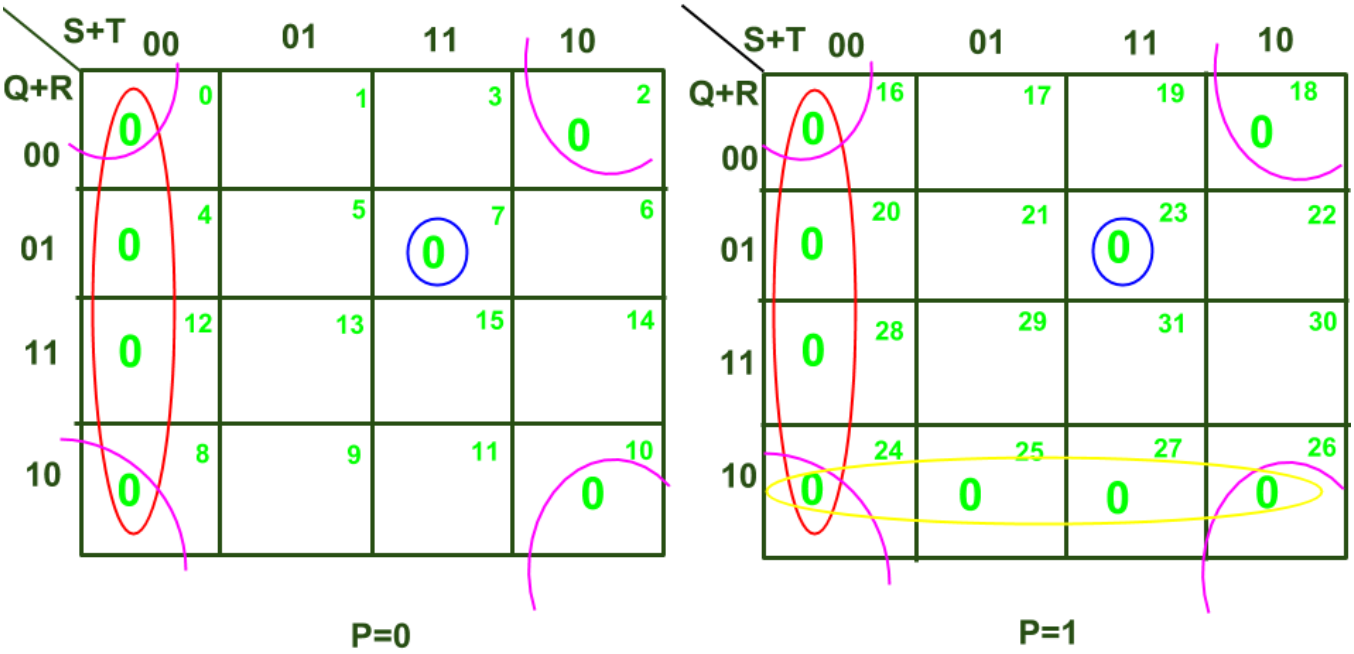
Therefore, the minimal POS expression of the given Boolean function in five variables is,

$$f(A, B, C, D, E) = (\overline{B} + D + \overline{E})(C + \overline{D} + \overline{E})(\overline{C} + D + \overline{E})(B + \overline{C} + \overline{D} + E)(\overline{B} + C + \overline{D}$$

Example 2:

Minimize the following 5 variable SPOS function using K map:
F(P,Q,R,S,T)=πM(0, 2, 4, 7, 8, 10, 12, 16, 18, 20, 23, 24, 25, 26, 27, 28)

Sol:



- First fill the cells which are given in the POS function with '0'.
- Group the adjacent cells in group of 1, 2 and 4 and 8.
- There are four groups in this K map.

Group 1: Cells 0, 4, 8, 12, 16, 20, 24 and 28 are grouped. The common term is S+T.

Group 2: Cell number 7 is in group 23 are grouped.The common term is Q+R+S'+T'.

Group 3: Cell number 0, 2, 8, 10, 16, 18, 24 and 26 are grouped.The common term is R+T.

Group 4: Cells 24, 25, 26 and 27 2 are grouped. The common term is P'+Q'+R.

So the minimized POS expression is

f(P,Q,R,S,T)= (S+T).(Q+R+S'+T').(R+T).(P'+Q'+R)

UNIT -2

GATE-LEVEL MINIMIZATION

TOPIC – 8

SSOP AND SPOS BOOLEAN FUNCTIONS SIMPLIFICATION WITH DON'T CARES

Don't Care Conditions:

- "Don't Care" conditions (represented by "X") are input conditions where the output doesn't affect the system.Such conditions are termed don't care conditions.
- We mark don't-care outputs in truth tables and K-maps with X's/d's/Φ.

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	X
1	1	1	1

Examples of where "Don't Cares" might occur:

- Unused combinations in BCD,only 10 combinations (0000 to 1001) are valid.The remaining combinations (1010 to 1111) are "Don't Care" because they are never used.
- Within a K-map, each "X" can be considered as either 0 or 1.
- We can achieve more simplified expressions, leads to simpler circuit designs.

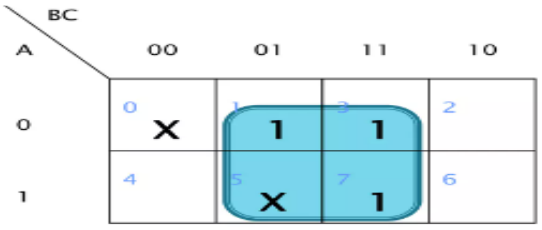
How Don't Cares Help in Simplification:

- Don't Cares can be used to form larger groups (quads, octets, or pairs) by treating them as either 1 or 0, whichever helps to minimize the Boolean expression.
- Large groups make Simpler expressions, since each group reduces the more number of variables .

Example 1:

Minimize the following SSOP function with don't cares using K map method F= m(1, 3, 7) + d(0,5)
Solution:

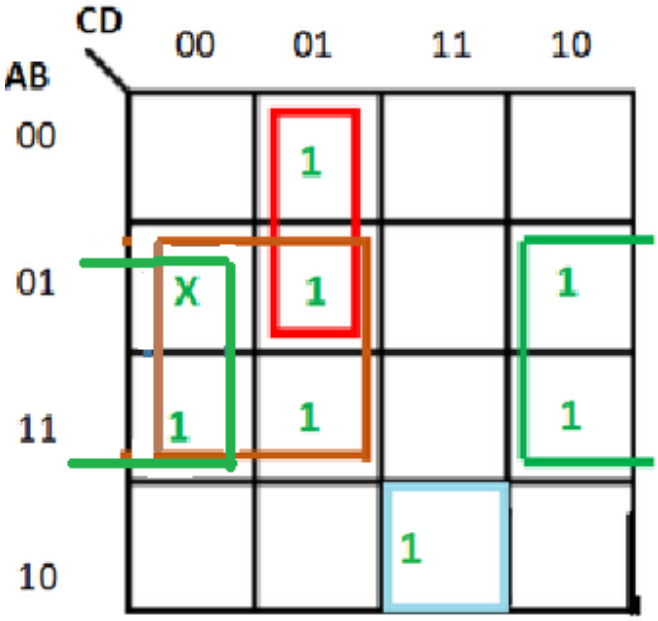
The simplified expression with don't cares is: **F=C** **(**A,B variables are changing,C remains unchanged **)**



Example 2:

F= m(1, 5, 6, 11, 12, 13, 14) + d(4)

Sol:



Group 1: d4,m5,m12 and m13 are grouped. The common term is **BC'**.

Group 2: d4,m6,m12 and m14 are grouped.The common term is **BD'**.

Group 3: m1,m5 are grouped.The common term is **A'C'D**.

Group 4: m11 don't have adjacency, treated as Isolated minterm. The common term is **AB'CD**.

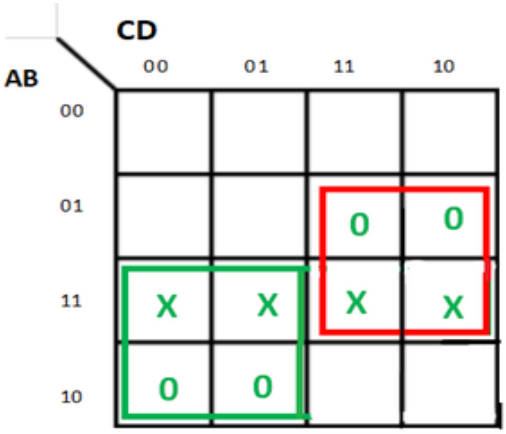
So the simplified expression with don't cares is:**F= BC' + BD' + A'C'D + AB'CD**

Example 3:

Minimize the following function in POS minimal form using K-Maps:

F(A, B, C, D) = M(6, 7, 8, 9) + d(12, 13, 14, 15)

Sol:



Group 1: M6,M7,d14, d15 are grouped. The common term is **(B' + C')**.

Group 2:M8,M9,d12,d13 are grouped.The common term is **(A'+ C)**.

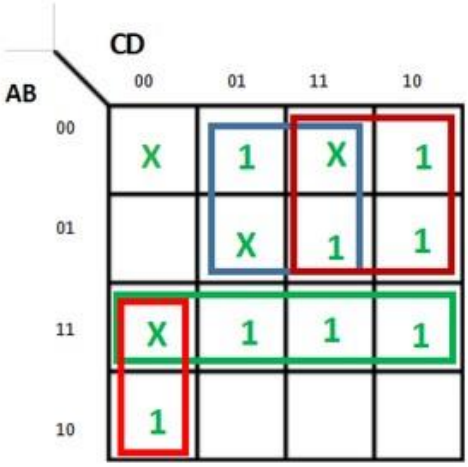
Therefore, simplified expression with don't cares is: **F = (A'+ C)(B' + C')**

Example 4:

Minimize the following function in SSOP to minimal form using K-Map method?

F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(0, 3, 5, 12)

Sol:



Group 1: d12 & m8 are grouped. The common term is **AC'D'**.

Group 2: m1,d3,d5 and m7 are grouped.The common term is **A'D**.

Group 3: d3,m2,m7 and m6 are grouped.The common term is **A'C**.

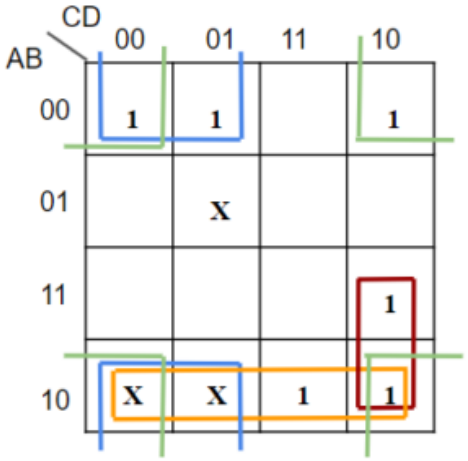
Group 4: d12,m13,m14 and m15 are grouped. The common term is **AB**.

So the simplified expression with don't cares is:**F= AC'D' + A'D + A'C + AB.**

Example-5:

Simplify F(A,B,C,D) = ∑(0,1,2,10,11,14) + d(5,8,9) in SOP form.

Sol:



I Quad covers minterms 0, 2, 10 and d8 -- B'D'

II Quad covers minterms 10, 11 and d8, d9-- AB'

III Quad covers minterms 0, 1 and d8, d9 -- B'C'

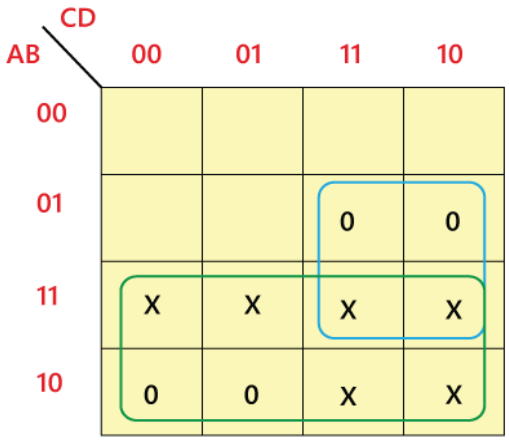
A pair covers minterms 10 and 14-- ACD'.

So the simplified expression with don't cares is:**F= AC'D' + A'D + A'C + AB.**

Example-6:

Simplify F(A,B,C,D) = M(6,7,8,9) + d(10,11,12,13,14,15)

Sol:



Group 1: M8,M9, d10,d11, d12,d13,d14 & d15 are grouped. The common term is **A'**.

Group 2: M6,M7,d14 &d15 are grouped.The common term is **(B'+C')**.

So, the minimized POS form of the function is:**F = A'(B' + C')**

UNIT -2

GATE-LEVEL MINIMIZATION

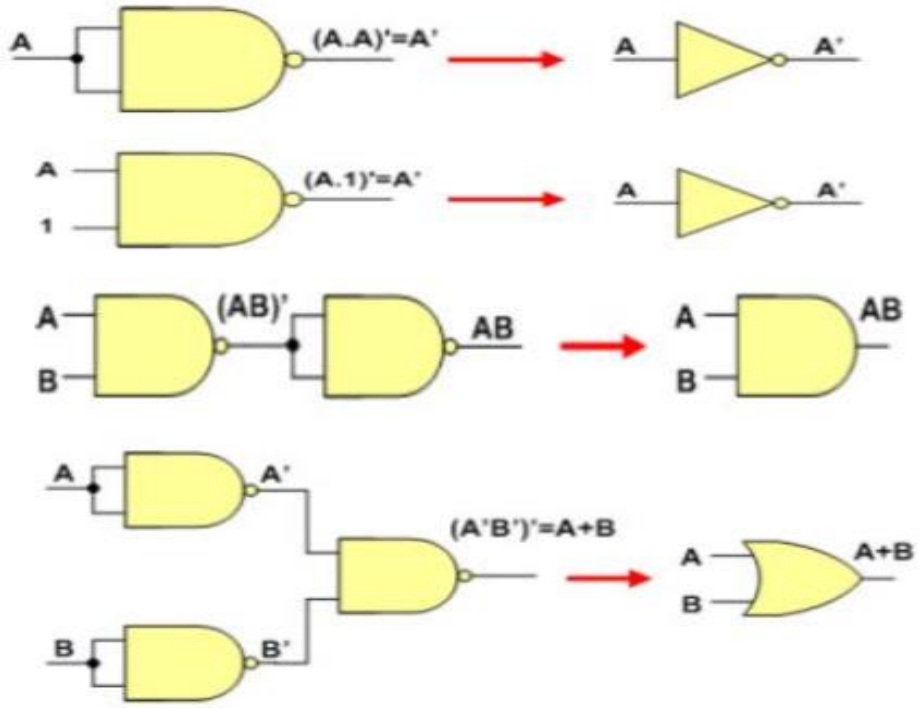
TOPIC – 9

NAND AND NOR IMPLEMENTATION

Nand and Nor Implementation:

- NAND & NOR gates are universal gates,since we can convert any circuit into a circuit consisting only of NAND gates or NOR gates.
- Digital circuit are frequently constructed with NAND or NOR gates rather than AND and OR gates.
- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families(covering AND,OR,NOT).
- NAND and NOR gates are easier and more cost-effective to manufacture.
- In practical circuits, using only one type of gate can simplify hardware design and reduce power consumption.

Realization of logic gates using NAND gates:



NOT Using NAND:

(AA)' = A'

AND Using NAND:

This is quite straightforward, we wish to obtain AB but the NAND gate gives an output (AB)'

so we complement the output of the NAND gate using another NAND gate to obtain ((AB)')'

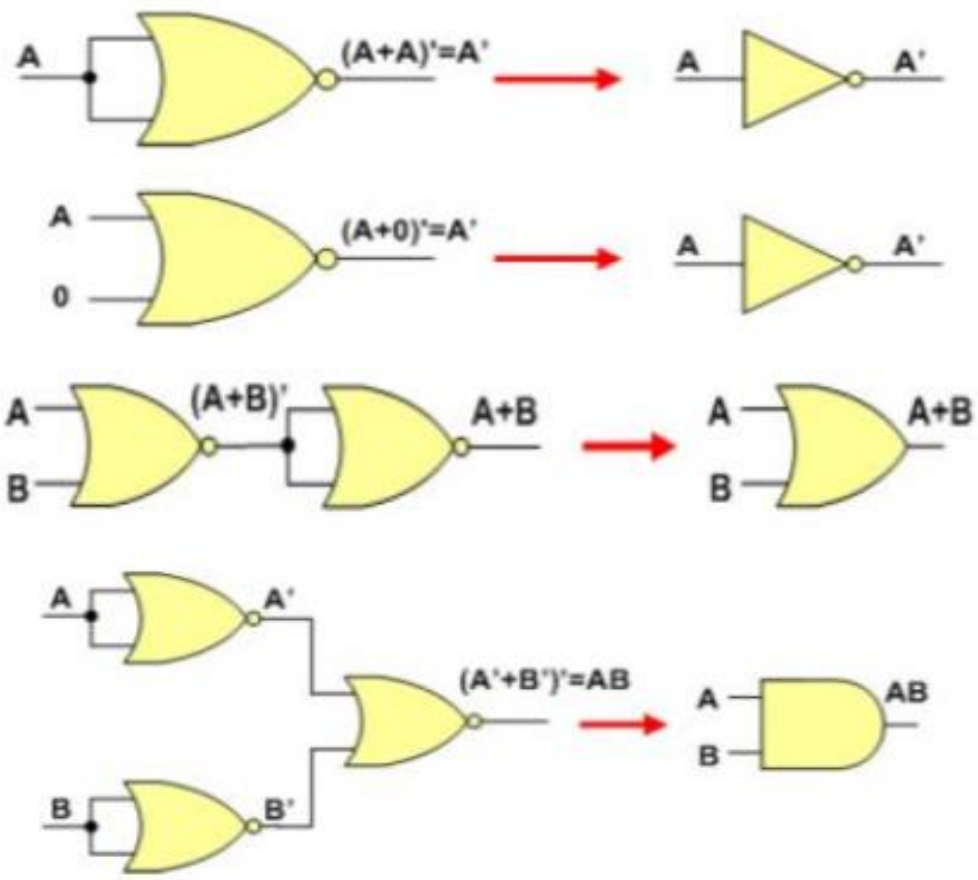
which is AB

OR Using NAND:

We first complement the inputs A and B. Then we perform the NAND operation on these complemented inputs. We get (A'B')'.

Using de Morgan's law we can show that (A'B')' = A + B.

Realization of logic gates using NOR gates:



NOT Using NOR:

(A+A)' = A'

AND Using NOR:

We first complement the inputs A and B. Then we perform the NOR operation on these complemented inputs. We get (A'+B')'.

Using de Morgan's law we can show that (A'+B')' = A . B.

OR Using NOR:

This is quite straightforward, we wish to obtain A+B but the NOR gate gives an output (A+B)',so we complement the output of the NOR gate using another NOR gate to obtain ((A+B)')' which is A+B

Examples of where "Don't Cares" might occur:

- Unused combinations in BCD,only 10 combinations (0000 to 1001) are valid.The remaining combinations (1010 to 1111) are "Don't Care" because they are never used.
- Within a K-map, each "X" can be considered as either 0 or 1.
- We can achieve more simplified expressions, leads to simpler circuit designs.

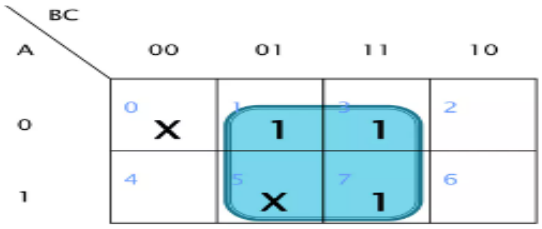
How Don't Cares Help in Simplification:

- Don't Cares can be used to form larger groups (quads, octets, or pairs) by treating them as either 1 or 0, whichever helps to minimize the Boolean expression.
- Large groups make Simpler expressions, since each group reduces the more number of variables .

Example 1:

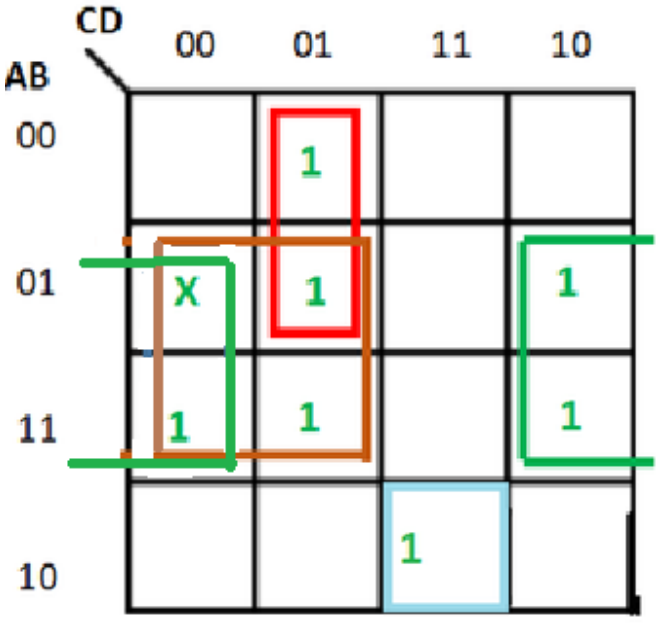
Minimize the following SSOP function with don't cares using K map method F= m(1, 3, 7) + d(0,5)
Solution:

The simplified expression with don't cares is: **F=C** **(**A,B variables are changing,C remains unchanged **)**



Example 2:

F= m(1, 5, 6, 11, 12, 13, 14) + d(4)
Sol:



Group 1: d4,m5,m12 and m13 are grouped. The common term is **BC'**.

Group 2: d4,m6,m12 and m14 are grouped.The common term is **BD'**.

Group 3: m1,m5 are grouped.The common term is **A'C'D**.

Group 4: m11 don't have adjacency, treated as Isolated minterm. The common term is **AB'CD**.

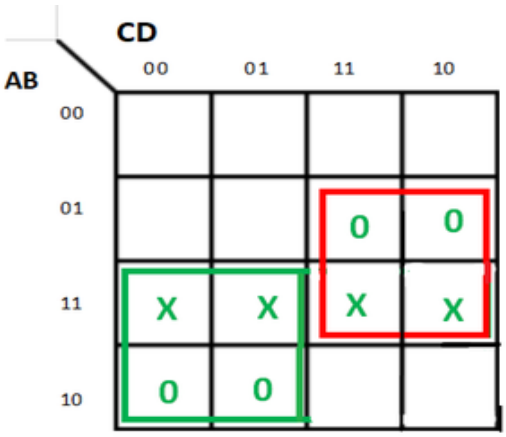
So the simplified expression with don't cares is:**F= BC' + BD' + A'C'D + AB'CD**

Example 3:

Minimize the following function in POS minimal form using K-Maps:

F(A, B, C, D) = M(6, 7, 8, 9) + d(12, 13, 14, 15)

Sol:



Group 1: M6,M7,d14, d15 are grouped. The common term is **(B' + C')**.

Group 2:M8,M9,d12,d13 are grouped.The common term is **(A'+ C)**.

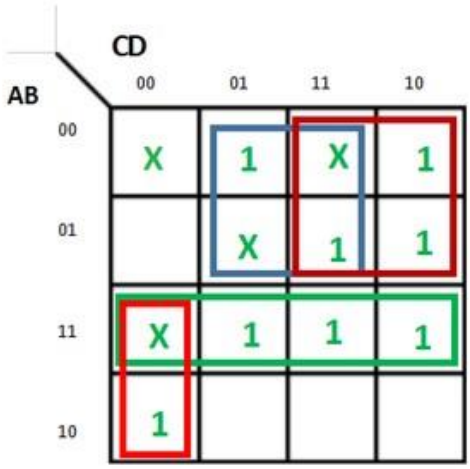
Therefore, simplified expression with don't cares is: **F = (A'+ C)(B' + C')**

Example 4:

Minimize the following function in SSOP to minimal form using K-Map method?

F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(0, 3, 5, 12)

Sol:



Group 1: d12 & m8 are grouped. The common term is **AC'D'**.

Group 2: m1,d3,d5 and m7 are grouped.The common term is **A'D**.

Group 3: d3,m2,m7 and m6 are grouped.The common term is **A'C**.

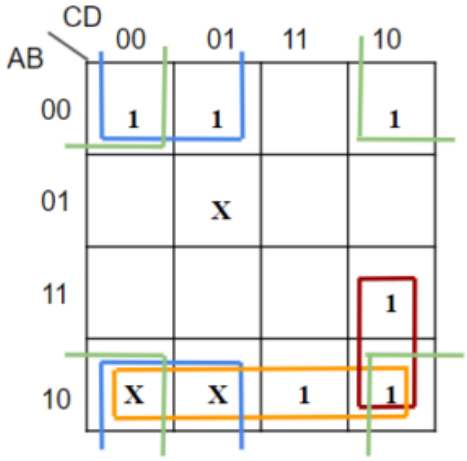
Group 4: d12,m13,m14 and m15 are grouped. The common term is **AB**.

So the simplified expression with don't cares is:**F= AC'D' + A'D + A'C + AB.**

Example-5:

Simplify F(A,B,C,D) = ∑(0,1,2,10,11,14) + d(5,8,9) in SOP form.

Sol:



I Quad covers minterms 0, 2, 10 and d8 -- B'D'

II Quad covers minterms 10, 11 and d8, d9-- AB'

III Quad covers minterms 0, 1 and d8, d9 -- B'C'

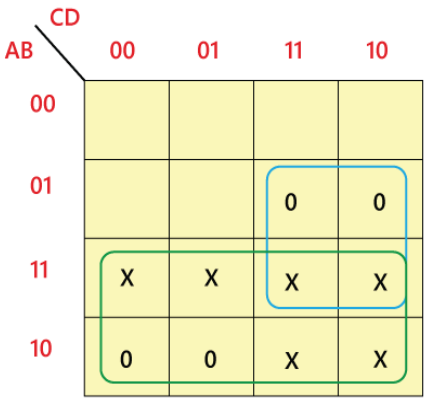
A pair covers minterms 10 and 14-- ACD'.

So the simplified expression with don't cares is:**F= AC'D' + A'D + A'C + AB.**

Example-6:

Simplify F(A,B,C,D) = M(6,7,8,9) + d(10,11,12,13,14,15)

Sol:



Group 1: M8,M9, d10,d11, d12,d13,d14 & d15 are grouped. The common term is **A'**.

Group 2: M6,M7,d14 &d15 are grouped.The common term is **(B'+C')**.

So, the minimized POS form of the function is:**F = A'(B' + C')**

UNIT-2
GATE-LEVEL MINIMIZATION
TOPIC – 10
OTHER TWO LEVEL IMPLEMENTATION
&
EXCLUSIVE-OR FUNCTION

Other Two Level Implementation:

- Two level logic means that the logic design uses maximum two logic gates between input and output.
- For two-level logic implementation, we consider four logic gates i.e. AND Gate, OR Gate, NAND Gate, and NOR Gate.
- If we use one of these four gates at first level and one at the second level then we get a total of 16 combinations of two-level logic.
- There are two types in 16 combinations.

  Degenerate Form

  Non-Degenerate Form

Degenerate Form:

If the output of two level logic realization can be obtained by using single Logic gate, then it is called as **degenerative form**.

Only **6 combinations** of two level logic realizations out of 16 combinations come under degenerative form. Those are AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-NOR, NOR-NAND.

Non-degenerative Form:

If the output of two level logic realization can't be obtained by using single logic gate, then it is called as **non-degenerative form**.

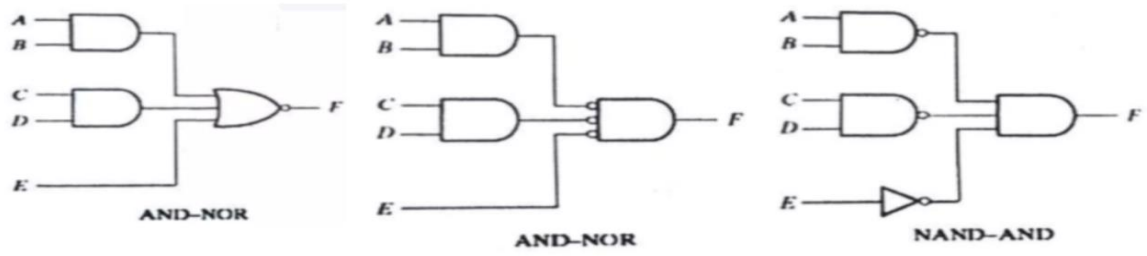
The remaining **10 combinations** of two level logic realizations come under non-degenerative form. Those are AND-OR, AND-NOR, OR-AND, OR-NAND, NAND-AND, NAND-OR, NAND-NAND, NOR-AND, NOR-OR, NOR-NOR.

AND-OR-Invert Implementation:

AND-NOR Logic:

In this logic realization, AND gates are present in first level and NOR gate(s) are present in second level. The following figure shows an example for **AND-NOR logic** realization.

F=(AB+CD+E)'



This Boolean function is in **AND-OR-Invert** form. Since, we can't implement it by using single logic gate, this AND-NOR logic realization is a **non-degenerative form.**

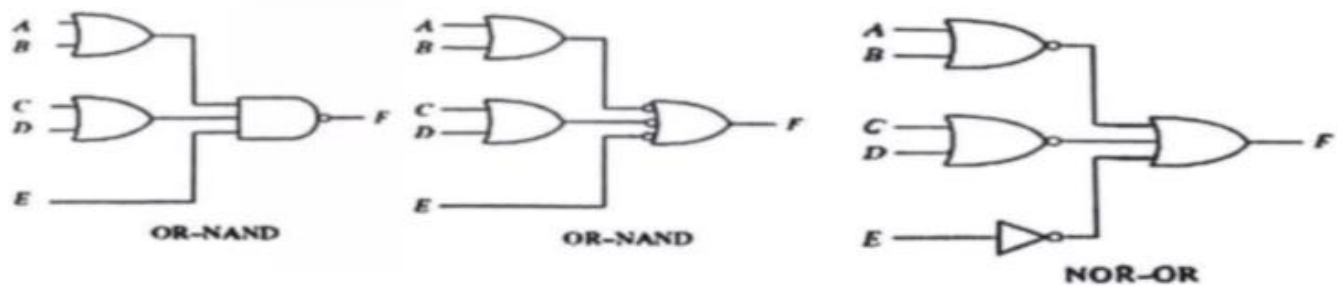
The AND-NOR form and NAND-AND forms are equal.

OR-AND-Invert Implementation:

OR-NAND Logic:

In this logic realization, OR gates are present in first level & NAND gate(s) are present in second level.

F=[(A+B)(C+D)E]'



This Boolean function is in **OR-AND-Invert** form. Since, we can't implement it by using single logic gate, this OR-NAND logic realization is a **non-degenerative form.**

The OR-NAND form and NOR-OR forms are equal.

EXCLUSIVE-OR FUNCTION:

XOR Gate: XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.

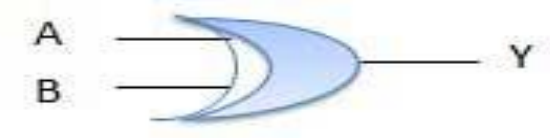
$$Y = A \text{ XOR } B \text{ XOR } C \ldots\ldots N$$

$$Y = A \oplus B \oplus C \ldots\ldots N$$

$$Y = \overline{A}B + A\overline{B}$$

XOR gate is also known as the Exclusive OR gate or Ex-OR gate. It gives the output **1** (High) if an odd number of inputs is high. This can be understood in the Truth Table. It can have an infinite number of inputs and only one output. In most cases, two-input or three-input XOR gates are used.

LOGIC DIAGRAM **TRUTH TABLE**



Inputs		Output
A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Properties of XOR:

1.Commutative: A⊕B=B⊕A

2.Associative: (A⊕B)⊕C=A⊕(B⊕C)

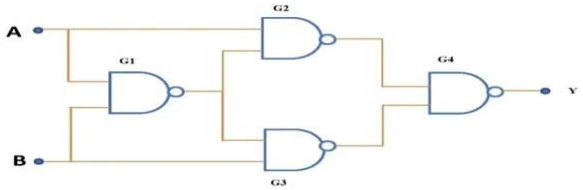
3.Identity: A ⊕0=A and A ⊕1=A'

4.Self-Inverse: A⊕A=0 (XORing a value with itself always results in 0)

5.A ⊕A' = 1

XOR gate using NAND gate:

One can construct an XOR gate by using **a minimum of four NAND gates**. However, It is also possible to design an XOR gate using more than four NAND gates. The following figure shows the circuit diagram for the implementation of a two-input XOR gate using four NAND gates.



The output for $G_1 = \overline{AB}$

The output of $G_2 = \overline{A.\overline{AB}} = \overline{A.(\overline{A} + \overline{B})} = \overline{A.\overline{B}}$

The output of $G_3 = \overline{B.\overline{AB}} = \overline{B.(\overline{A} + \overline{B})} = \overline{B.\overline{A}}$

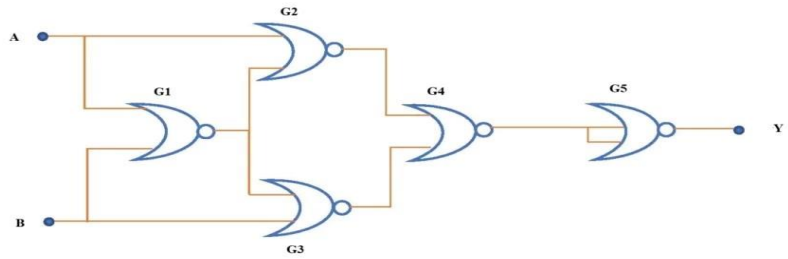
The output of G_4 gate = $Y = \overline{(A.\overline{B}).(B.\overline{A})} = \overline{A.\overline{B}} + \overline{B.\overline{A}} = A.\overline{B} + B.\overline{A}$

Thus, output of the overall circuit is $Y = A.\overline{B} + \overline{A}.B$

Thus the output of the above circuit is the same as the output of an XOR gate. Hence the above circuit represents the circuit diagram of Exclusive OR gate using NAND gates.

XOR gate using NOR gate:

To design the circuit diagram of an XOR gate using only NOR gates, a **minimum of five NOR gates** are required. XOR gate can be contained by more than five NOR gates as well. The following figure is the schematic diagram of XOR gate using five NOR gates.



Derivation of the output of NOR gate-based XOR circuit:

Here is how to obtain the output of XOR gate from the above circuit –

The output for $G_1 = \overline{A + B}$

The output of $G_2 = \overline{A + \overline{A + B}} = \overline{A + (\overline{A}.\overline{B})} = \overline{A + \overline{B}}$

The output of $G_3 = \overline{B + \overline{A + B}} = \overline{B + (\overline{A}.\overline{B})} = \overline{B + \overline{A}}$

The output of G_4 gate = $\overline{(A + \overline{B}) + (B + \overline{A})} = \overline{(\overline{A}.B) + (\overline{B}.A)}$

The output of $G_5 = \overline{\overline{(\overline{A}.B) + (\overline{B}.A)}} = \overline{A}B + A\overline{B}$

Thus, output of the overall circuit is $Y = A.\overline{B} + \overline{A}.B$