Lecture No 04

Simplification of Boolean Function

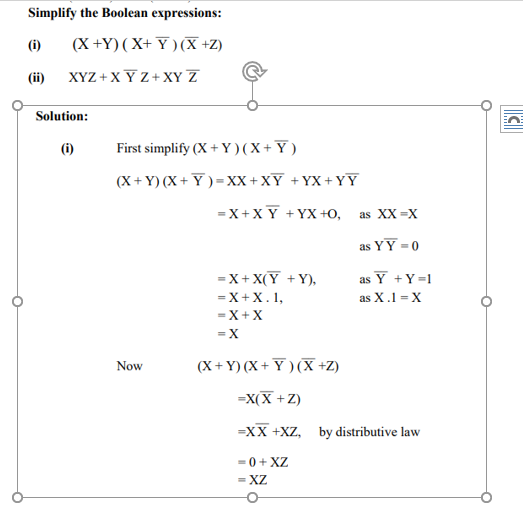
Simplification of Boolean functions is mainly used to reduce the gate count of a design. Less number of gates means less power consumption, sometimes the circuit works faster and also when number of gates is reduced, cost also comes down.

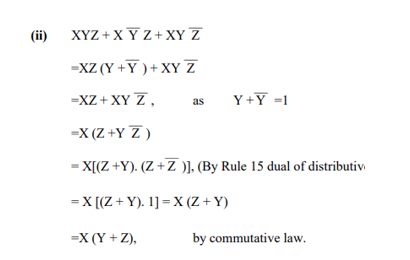
|  |
| --- |
|  |
| * Algebraic Simplification.   + Simplify symbolically using theorems/postulates.   + Requires good skills * Karnaugh Maps.   + Diagrammatic technique using 'Venn-like diagram' |

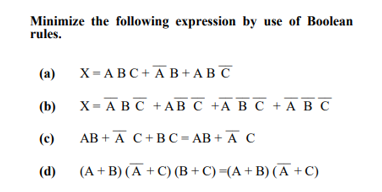
Simplification Using Algebraic Functions

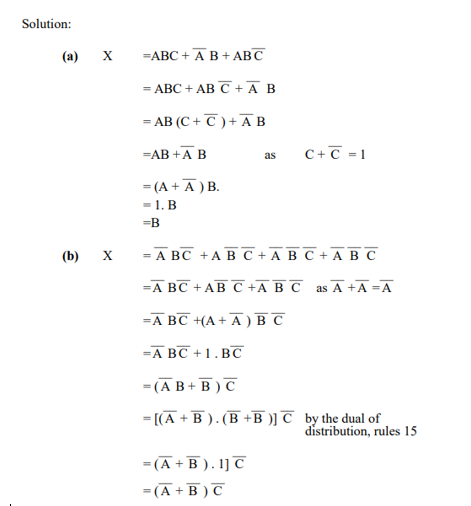
In this approach, one Boolean expression is minimized into an equivalent expression by applying Boolean identities.

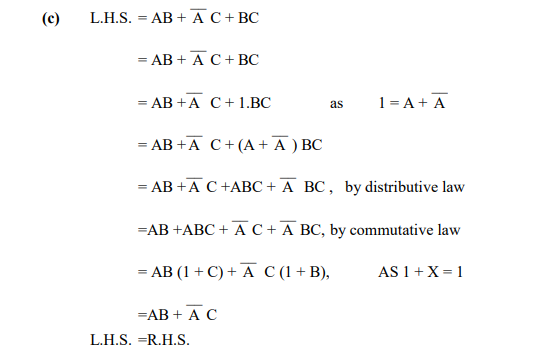
Problem; 1

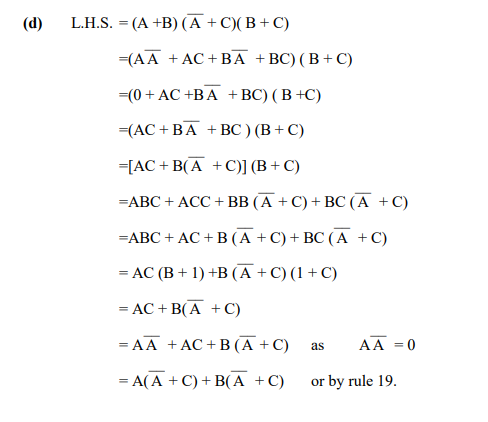






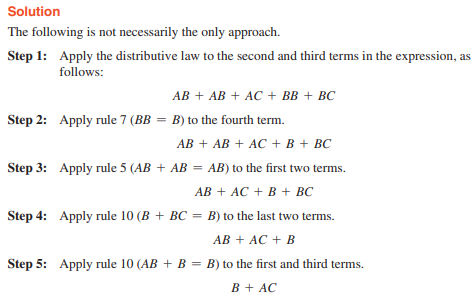


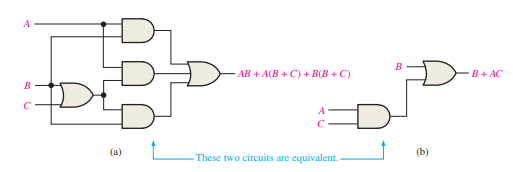




Problem 2

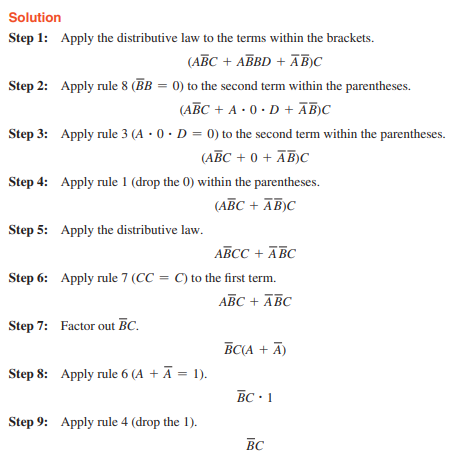




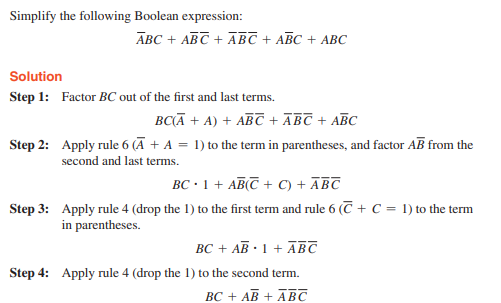


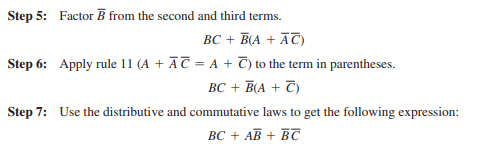
Problem 2



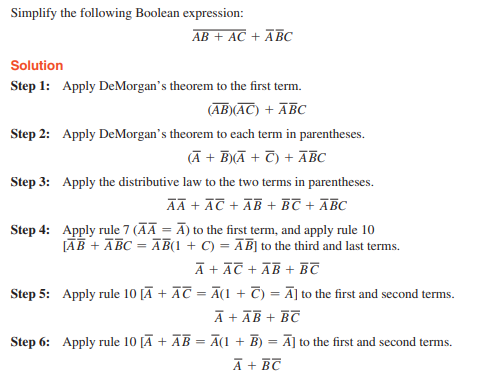


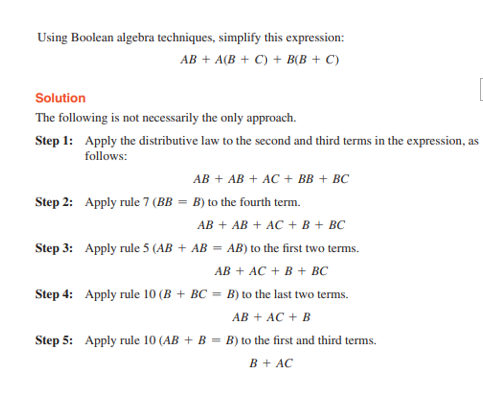
Problem 3

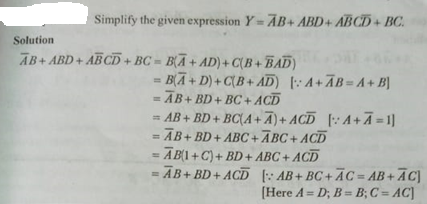




Problem 4

(



****

**SOP and POS representation for logic expression**

In any logic expression there may be two types of terms called sum term and product terms.

**Sum Term** the term in which the variables are ORed i.e. A + . The sum term contains both complemented or uncomplemented variables.

**Product Term** the term in which the variables are ANDed i.e. A.. The product term contains both complemented or uncomplemented variables.

There are two standard forms

1. Sum-of-product form (SOP)
2. Product-of-sum form (POS)

**Sum-of-product (SOP) form**

It is the logical sum of two or more logical product terms, is called a Sum-of-product expression. It is basically the OR operation of AND operated variables. for example, X = AB+BC+ A, Y=ABC+ A

**Product-of-sum (POS) form**

It is the logical product of two or more logical sum terms, is called a Sum-of-product expression. It is basically the AND operation of OR operated variables. for example, X = (A+B+BC) (A+, Y=(A+B+C) (A+

**Standard or Canonical SOP and POS forms**

**Standard SOP forms**

Consider the following expression X (A, B, C) = AB+BC+ A. The following function is the three variable function and contains three product terms AB, BC and A.In each term one of the variables is missing, the missing variable can be added by using Boolean rule (B+)=1.

Thus X (A, B, C) = AB+BC+ A = AB.1+1.BC+ A.1

= AB(C+)+ (A+)BC+ A(C+)

= ABC+AB+ABC++ A+A

As each term contain all the variables, so called standard SOP or canonical SOP form and each term is called Minterm represented by m.

**Standard POS forms**

Consider the following expression X (A, B, C) = (A+B)(B+C)(A+. The following is the three variable function and contains three Sum terms A+B, B+C and A+.In each term one of the variables is missing, the missing variable can be added by using Boolean rule (B)=0.

Thus X (A, B, C) = (A+B) (B+C) (A+

X (A, B, C) = (A+B+0) (0+B+C) (A+

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

As each term contain all the variables, so called standard POS or canonical POS form and each term is called Maxterm represented by M.

**Problems**

Convert the following expression into their standard SOP and POS form.

1. X= A+BC+ABC
2. X= (A+B) (B+)
3. X = A+C

**Minterm and Maxterm for three variables**

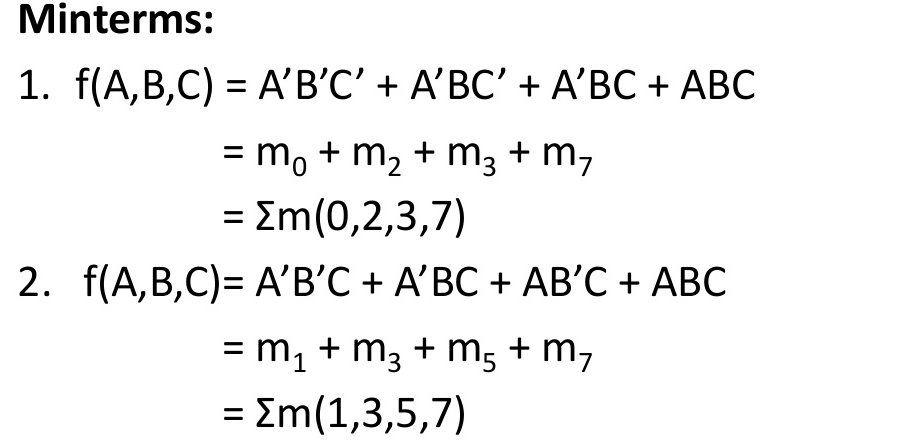
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| inputs | | | Minterm | Designation | Maxterm | Designation |
|  |  |  |
| 0 | **0** | **0** |  | **m0** |  | **M0** |
| 0 | **0** | **1** |  | **m1** |  | **M1** |
| 0 | **1** | **0** |  | **m2** |  | **M2** |
| 0 | **1** | **1** |  | **m3** |  | **M3** |
| 1 | **0** | **0** |  | **m4** |  | **M4** |
| 1 | **0** | **1** |  | **m5** |  | **M5** |
| 1 | **1** | **0** |  | **m6** |  | **M6** |
| 1 | **1** | **1** |  | **m7** |  | **M7** |

Above table show that the complemented variables are designated by 0 and uncomplemented variables are designated by 1for Minterm.

Thus = m000 =m0 and  **=** m101 = m5

But for maxterm the complemented variables are designated by 1 and uncomplemented variables are designated by 0.

**= M000 = M0 and = M110 = M6**



Maxterms:
1. f(A,B,C) = (A+B+C).(A+B’+C).(A+B’+C’)+(A’+B’+C’)
= M0 + M2 + M3 + M7
= ΠM(0,2,3,7)
2. f(A,B,C)= (A+B+C’).(A+B...

**Example 7:**

**Express the following function in a sum of minterms and product of maxterms.**

1. **Sum of minterms**:

* Multiply, we get:

**m6 m7 m3 m5**

**In terms of truth table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Inputs | | | Minterm | Z |
|  |  |  |
| 0 | **0** | **0** | **m 0** | **0** |
| 0 | **0** | **1** | **m1** | **0** |
| 0 | **1** | **0** | **m2** | **0** |
| 0 | **1** | **1** | **m3** | **1** |
| 1 | **0** | **0** | **m4** | **0** |
| 1 | **0** | **1** | **m5** | **1** |
| 1 | **1** | **0** | **m6** | **1** |
| 1 | **1** | **1** | **m7** | **1** |

Sum of minterms

1. **Sum of maxterms**

* Using distributive law:

|  |
| --- |
|  |

* In the
* We get:

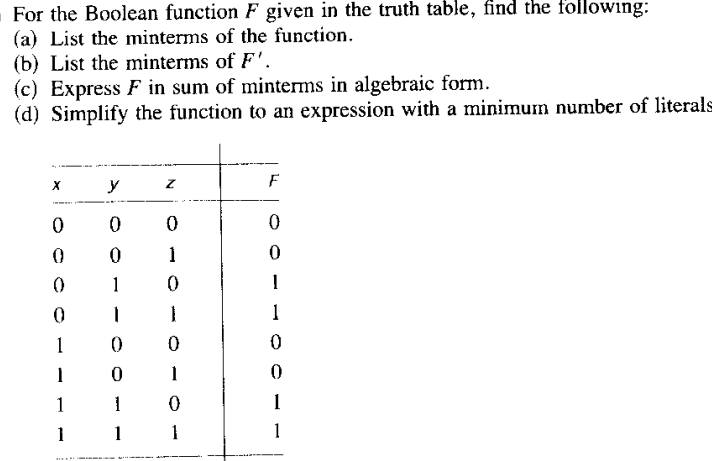
In we need , in we need we need

* We can write
* Substitute all of these term in ,we get:

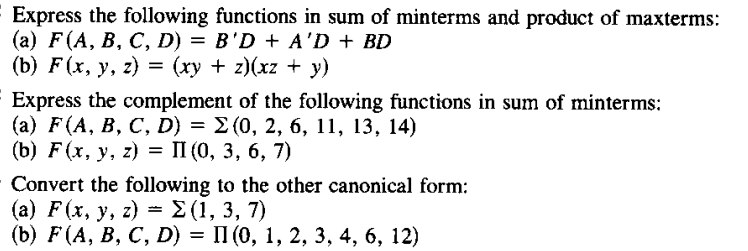
**= П(0,1,2,4)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| inputs | | |  | Maxterm |
|  |  |  |
| 0 | **0** | **0** | **0** | **M0** |
| 0 | **0** | **1** | **0** | **M1** |
| 0 | **1** | **0** | **0** | **M2** |
| 0 | **1** | **1** | **1** | **M3** |
| 1 | **0** | **0** | **0** | **M4** |
| 1 | **0** | **1** | **1** | **M5** |
| 1 | **1** | **0** | **1** | **M6** |
| 1 | **1** | **1** | **1** | **M7** |

**Problem**



**Problem**

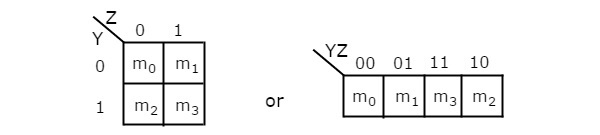


**Karnaugh map(k-map)**

K-map is a graphical method of simplifying the Boolean expression. K-map consist of cells or boxes. Each box contain the information about SOP or POS.

2 Variable K-Map

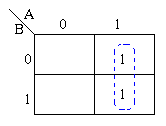
The number of cells in 2 variable K-map is four, since the number of variables is two. The following figure shows **2 variable K-Map**.



* There is only one possibility of grouping 4 adjacent min terms.
* The possible combinations of grouping 2 adjacent min terms are {(m0, m1), (m2, m3), (m0, m2) and (m1, m3)}.

**Example 1:**

Consider the following map. The function plotted is: Z = f(A,B) = A + AB



Using algebraic simplification,

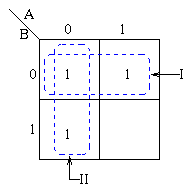
Z = A + AB

Z = A( + B)

Z = A

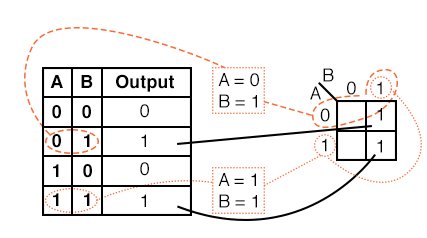
**Example 2:**

Consider the expression Z = f(A,B) =  + A  + B plotted on the Karnaugh map:



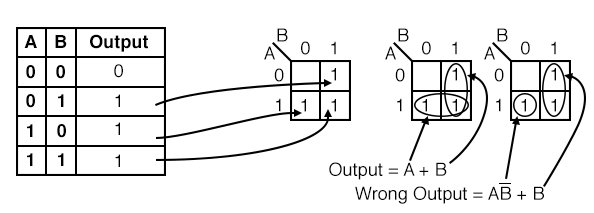
**Relation between truth table and K-map entries**

Transfer the contents of the truth table to the Karnaugh map above.

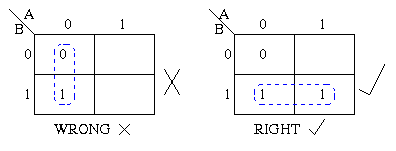
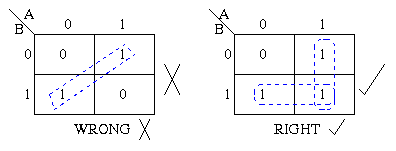
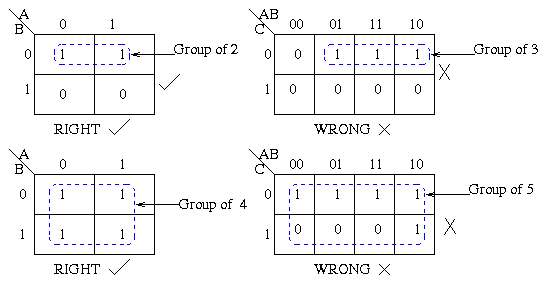


**Example:**

For the Truth table below, transfer the outputs to the Karnaugh, then write the Boolean expression for the result.

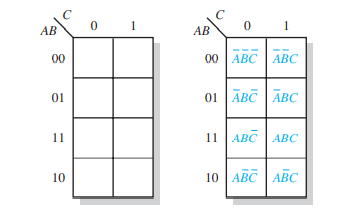


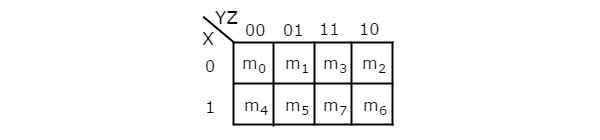
# The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together [adjacent](http://www.ee.surrey.ac.uk/Projects/Labview/common/glossary.html#Adj) cells containing *ones*

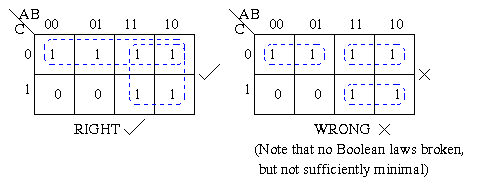
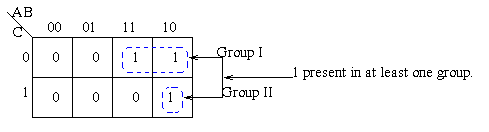
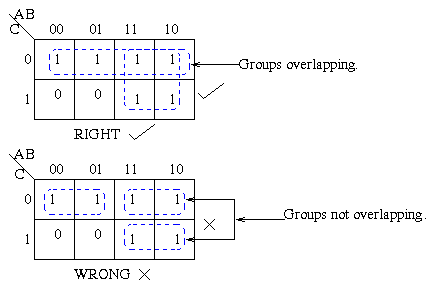
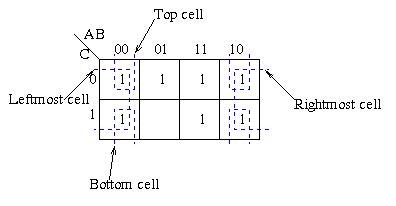
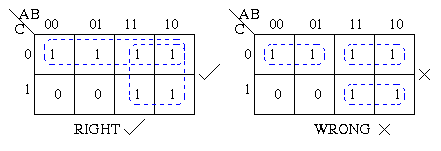
* **Groups may not include any cell containing a zero  
  **
* **Groups may be horizontal or vertical, but not diagonal.  
  **
* **Groups must contain 1, 2, 4, 8, or in general 2n cells.  
  That is if n = 1, a group will contain two 1's since 21 = 2.  
  If n = 2, a group will contain four 1's since 22 = 4.  
  **

**3- Variable K-Map**

The number of cells in 3 variable K-map is eight, since the number of variables is three. The following figure shows **3 variable K-Map**.



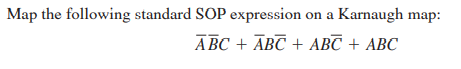


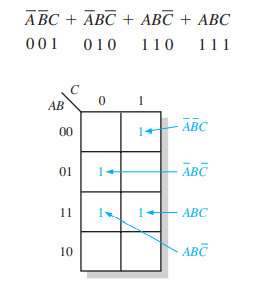
* **Each group should be as large as possible.  
  **
* **Each cell containing a *one* must be in at least one group.  
  **
* **Groups may overlap.  
  **
* **Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.  
  **
* **There should be as few groups as possible, as long as this does not contradict any of the previous rules.  
  **

**Summary:**

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible.

**Example**





**Example**

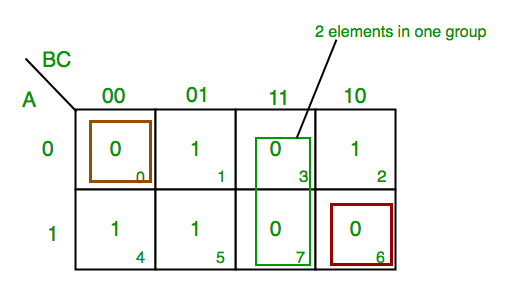


**Example**

**POS FORM**

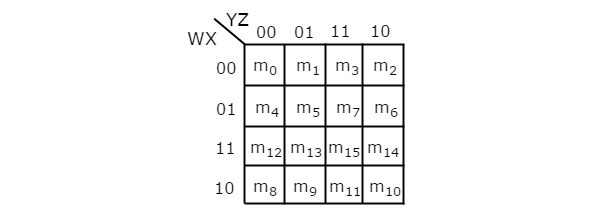
1. **K-map of 3 variables-**

F(A,B,C)=π(0,3,6,7)

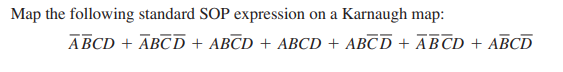
[](https://media.geeksforgeeks.org/wp-content/uploads/K-Map-Karnaugh-Map-4.png)

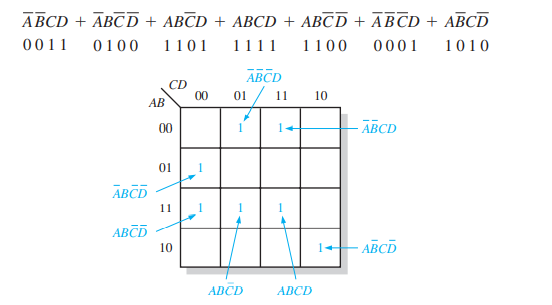
**4 Variable K-Map**

The number of cells in 4 variable K-map is sixteen, since the number of variables is four. The following figure shows **4 variable K-Map**.



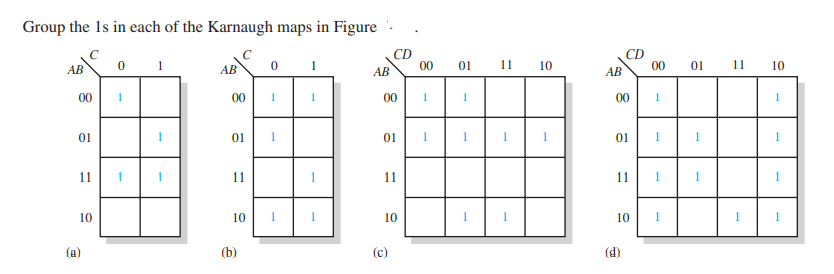
**Example**



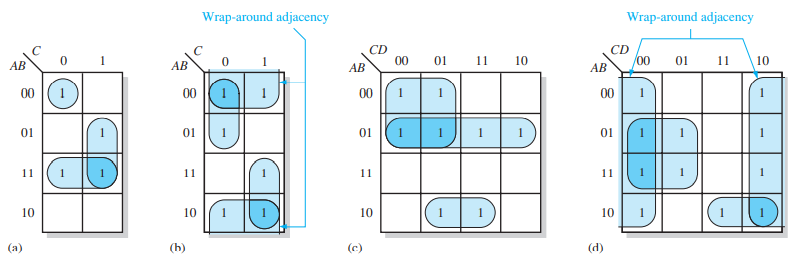


**Example**

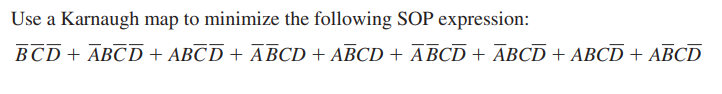


****

**Solution**

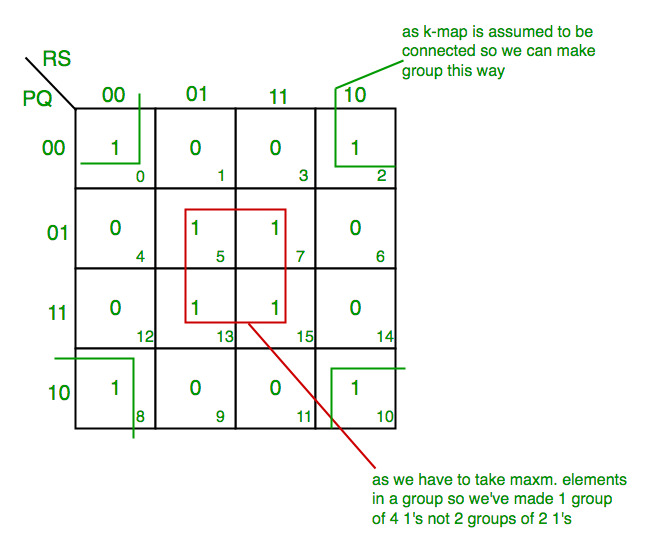


**Example**

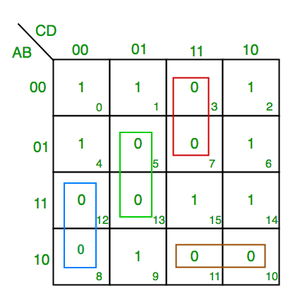


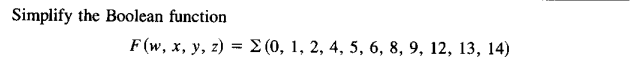
1. **K-map for 4 variables**

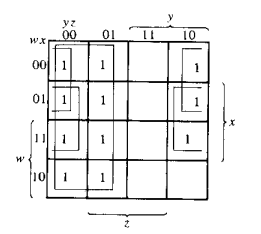
F(P,Q,R,S)=∑(0,2,5,7,8,10,13,15)   
 **Example**

[](https://media.geeksforgeeks.org/wp-content/uploads/K-Map-Karnaugh-Map-2-1.png)

**Example**

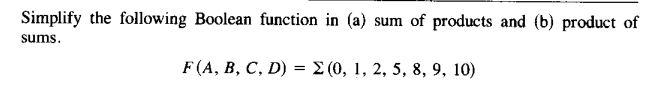
F(A,B,C,D)=π(3,5,7,8,10,11,12,13)



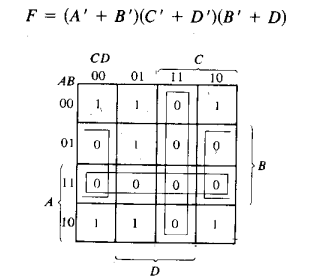




Problem



Solution



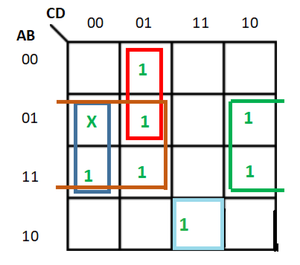
# **Don’t Care (X) Conditions in K-Maps**

One of the very significant and useful concepts in simplifying the output expression using K-Map is the concept of “Don’t Cares”. The “Don’t Care” conditions allow us to replace the empty cell of a [K-Map](https://www.geeksforgeeks.org/k-mapkarnaugh-map/) to form a grouping of the variables which is larger than that of forming groups without don’t cares. While forming groups of cells, we can consider a “Don’t Care” cell as 1 or 0 or we can also ignore that cell. Therefore, “Don’t Care” condition can help us to form a larger group of cells.

A Don’t Care cell can be represented by a cross(X) in K-Maps representing a invalid combination. For example, in Excess-3 code system, the states 0000, 0001, 0010, 1101, 1110 and 1111 are invalid or unspecified. These states are called don’t cares.

**Example-1:**   
Minimize the following function in SOP minimal form using K-Maps: 

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



Therefore, SOP minimal is, 

f = BC' + BD' + A'C'D + AB'CD

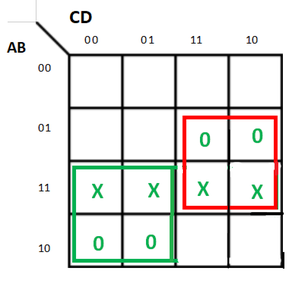
**Example-2:**   
Minimize the following function in POS minimal form using K-Maps: 

F (A, B, C, D) = m (0, 1, 2, 3, 4, 5) + d (10, 11, 12, 13, 14, 15)

**Explanation:**   
Writing the given expression in POS form: 

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

The POS K-map for the given expression is:

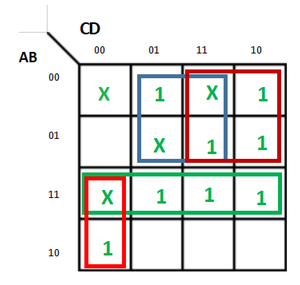


Therefore, POS minimal is,

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

**Example-3:**   
Minimize the following function in SOP minimal form using K-Maps:   
F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(0, 3, 5, 12)

**Explanation:**   
The SOP K-map for the given expression is: 



Therefore, 

**f = AC’D’ + A’D + A’C + AB**

**Significance of “Don’t Care” Conditions:**

Don’t Care conditions has the following significance in designing of the digital circuits: 

1. **Simplification of the output:**   
   These conditions denotes inputs that are invalid for a given digital circuit. Thus, they can used to further simplify the boolean output expression of a digital circuit.
2. **Reduction in number of gates required:**   
   Simplification of the expression reduces the number of gates to be used for implementing the given expression. Therefore, don’t cares make the digital circuit design more economical.
3. **Reduced Power Consumption:**   
   While grouping the terms long with don’t cares reduces switching of the states. This decreases the memory space that is required to represent a given digital circuit which in turn results in less power consumption.
4. **Represent Invalid States in Code Converters:**   
   These are used in code converters. For example- In design of 4-bit BCD-to-XS-3 code converter, the input combinations 1010, 1011, 1100, 1101, 1110, and 1111 are don’t cares.

