BOOLEAN ALGEBRA

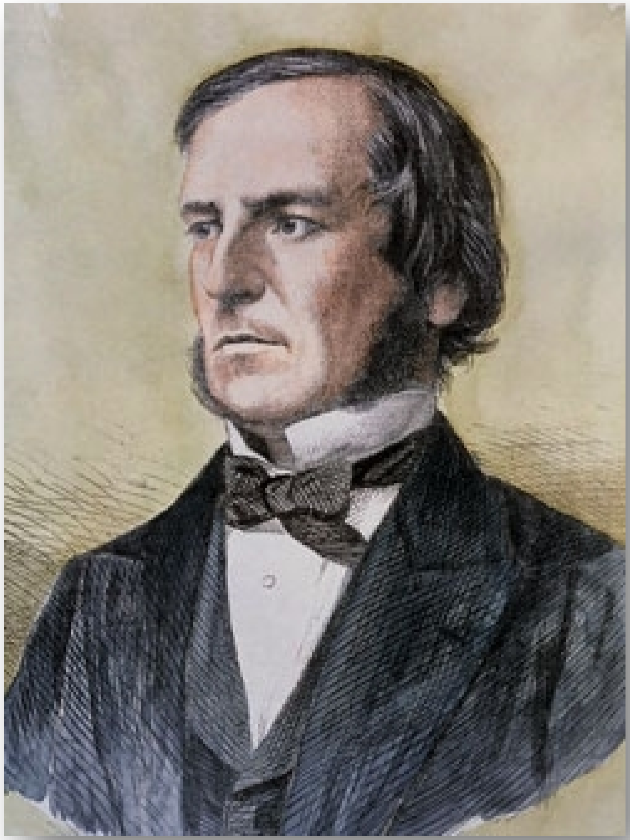
Compiled By: Afaq Alam Khan

Index

- Introduction
- Boolean Algebra Laws
- Boolean functions
- Operation Precedence
- Boolean Algebra Function
- Canonical Forms
  - SOP
  - POS
- Simplification of Boolean Functions
  - Algebric simplification
  - K-Map
  - Quine –McCluskey Method (Tabular Method)

Introduction

- Boolean Algebra is used to analyze and simplify the digital (logic) circuits.
- It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**.
- It is a convenient way and systematic way of expressing and analyzing the operation of logic circuits
- Boolean algebra was invented by **George Boole** in 1854.



Introduction

- Variable used in Boolean algebra can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- Complement of a variable is represented by an overbar (-). Thus, complement of variable B is represented as B' . Thus if B = 0 then B'= 1 and if B = 1 then B'= 0.
- ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
- Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Boolean Operations

AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Not

A	A’
0	1
1	0

Laws in Boolean Algebra

- **Commutative Law**

A.B = B.A

A+B = B+A

- **Associative Law**

(A.B).C = A.(B.C)

(A+B) + C = A+ (B+C)

- **Distributive Law**

A.(B+C)=A.B+A.C

A+(B.C)=(A+B).(A+C)

- **Absorption**

A+ (A.B)=A

A.(A+B)=A

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

- **AND Law**

A.0 = 0

A.1 =A

A.A = A

A.A' =0

- **OR law**

A+0 = A

A+1=1

A+A=A

A+A' = 1

- **Inversion Law(Involution)**

A'' = A

- **DeMorgan's Theorm**

(x.y)' = x' + y'

(x+y)' = x' . y'

Idempotent Law

Complement Law

Operator Presedence

- The operator Precedence for evaluating Boolean expression is:
  - 1. Parentheses
  - 2. NOT
  - 3. AND
  - 4. OR

Example

- Using the Theorems and Laws of Boolean algebra, Prove the following.

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

Boolean Algebric Function

- A Boolean function can be expressed algebraically with binary variables, the logic operation symbols, parentheses and equal sign.

- For a given combination of values of the variables, the Boolean function can be either 1 or 0.

- Consider for example, the Boolean Function:

$F1 = x + y'z$

The Function F1 is equal to 1 if x is 1 or if both y' and z are equal to 1; F1 is equal to 0 otherwise.

- The relationship between a function and its binary variables can be represented in a truth table. To represent a function in a truth table we need a list of the $\mathbf{2}^n$ combinations of the n binary variables.

- A Boolean function can be transformed from an algebraic expression into a logic diagram composed of different Gates

Boolean Algebric Function

- Consider the following Boolean function:

Canonical Form

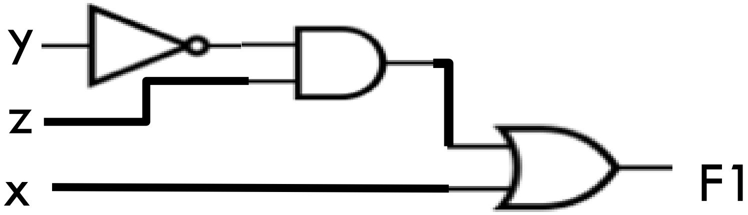
$F1 = x'y'z + xy'z' + xy'z + xyz' + xyz$

After Simplification

$F1 = x + y'z$

Non Canonical Form

- A Boolean function can be represented in a truth table.



Realization of Boolean Function using Gates

Truth Table

x	y	z	F1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits. It provides a convenient tool to:
  - Express in algebraic form a truth table relationship between binary variables.
  - Express in algebraic form the input-output relationship of logic diagrams.
  - Find simpler circuits for the same function.

- A Boolean function specified by a truth table can be expressed algebraically in many different ways. Two ways of forming Boolean expressions are **Canonical** and **Non-Canonical** forms.

Canonical Forms For Boolean Function

- **SOP Form:** The canonical SoP form for Boolean function of truth table are obtained by ORing the ANDed terms corresponding to the 1's in the output column of the truth table
- The product terms also known as **minterms** are formed by ANDing the complemented and un-complemented variables in such a way that the 0 in the truth table is represented by a complement of variable 1 in the truth table is represented by a variable itself.

Canonical Forms For Boolean Function

- SoP form – Example

F1= x'yz' + xy'z + xyz' + xyz

F1 = (m2+m5+m6+m7)

F1 =∑(m2,m5,m6,m7)

F1 = ∑ (2, 5,6,7)

Decimal numbers in the above expression indicate the subscript of the minterm notation

x	y	z	F1	Minterms	
0	0	0	0	x'y'z'	m0
0	0	1	0	x'y'z	m1
0	1	0	1	x'yz'	m2
0	1	1	0	x'yz	m3
1	0	0	0	xy'z'	m4
1	0	1	1	xy'z	m5
1	1	0	1	xyz'	m6
1	1	1	1	xyz	m7

Canonical Forms For Boolean Function

- **PoS Form:** The canonical PoS form for Boolean function of truth table are obtained by ANDing the ORed terms corresponding to the 0's in the output column of the truth table
- The product terms also known as **Maxterms** are formed by ORing the complemented and un-complemented variables in such a way that the 1 in the truth table is represented by a complement of variable 0 in the truth table is represented by a variable itself.

Canonical Forms For Boolean Function

- **PoS form – Example**

F2=(x+y+z).(x+y+z').(x+y'+z').(x'+y+z)

F2 = (M1.M2.M4.M5)

F2 =∏(M1,M2,M4,M5)

F2 = ∏(1, 2,4,5)

Decimal numbers in the above expression indicate the subscript of the Maxterm notation

x	y	z	F2	Maxterms	
0	0	0	**0**	**x + y+z**	M1
0	0	1	**0**	**x+y+z'**	**M2**
0	1	0	**1**	**x+y' + z**	M3
0	1	1	**0**	**x+y'+z'**	M4
1	0	0	0	x'+y+z	M5
1	0	1	1	x' +y+z'	M6
1	1	0	1	x'+y'+z	M7
1	1	1	1	x'+y'+z'	M8

Canonical Forms For Boolean Function

- **Example: Express the following in SoP form**

$F1 = x + y'z$

- **Solution:**

$=(y+y')x + y'z(x+x')$ [because $x+x'=1$]

$=xy + xy' + xy'z + x'y'z$

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

$=xyz + xyz' + xy'z + xy'z' + xy'z + x'y'z$

$=xyz + xyz' + (xy'z + xy'z) + xy'z' + x'y'z$

$= xyz + xyz' + xy'z + xy'z' + x'y'z$ [because $x+x =x$]

$= m7 + m6 + m5 + m4 + m1$

$= \sum(m7, m6, m5, m4, m1)$

$= \sum(1,4,5,6,7)$

Canonical Forms - Exercises

- Exercise 1: Express $G(A,B,C)=A.B.C + A'.B + B'.C$ in SoP form.

- Exercise 2: Express $F(A,B,C)=A.B' + B'.C$ in PoS form

Simplification of Boolean functions

- Algebric simplification
- K-Map simplification
- Quine-McLusky Method of simplification

Algebraic Simplification

- Using Boolean algebra techniques, simplify this expression: AB + A(B + C) + B(B + C)
- Solution

=AB + AB + AC + BB + BC	(Distributive law)
=AB + AB + AC + B + BC	(B.B=B)
= AB + AC + B + BC	(AB+AB=AB)
= AB + AC + B	(B+BC =B)
=B+AC	(AB+B =B)

Algebric Simplification

- Minimize the following Boolean expression using Algebric Simplification

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

- Solution

$=A'B+(BC'+BC')+BC+AB'C'$ [indeponent law]

$= A'B+(BC'+BC)+(BC'+AB'C')$

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

$=A'B + B.1+ c' (B+A)$

$= B(A'+1)+C'(B+A)$

$=B + C'(B+A)$ [A'+1=1]

$= B+BC'+AC'$

$= B(1+C')+AC'$

$= B+AC'$ [1+C' = 1]

Algebric Simplification

- Simplify: C + (BC)'

$=C + (BC)'$	Original Expression
$=C + (B' + C')$	DeMorgan's Law.
$=(C + C') + B'$	Commutative, Associative Laws.
$=1 + B'$	Complement Law.
$=1$	Identity Law.

Algebric Simplification

- **Exercise 3:** Using the theorems and laws of Boolean Algebra, reduce the following functions

$F1(A,B,C,D) = \sum(0,1,2,3,6,7,14,15)$

- <u>Solution:</u>

= A'B'C'D' + A'B'C'D + A'B'CD' + A'B'CD +A'BCD' + A'BCD + ABCD' + ABCD

= ?

- **Exercise 4:** Using the theorems and laws of Boolean Algebra, reduce the following functions

$F1(X,Y,Z) = \prod(0,1,4,5,7)$

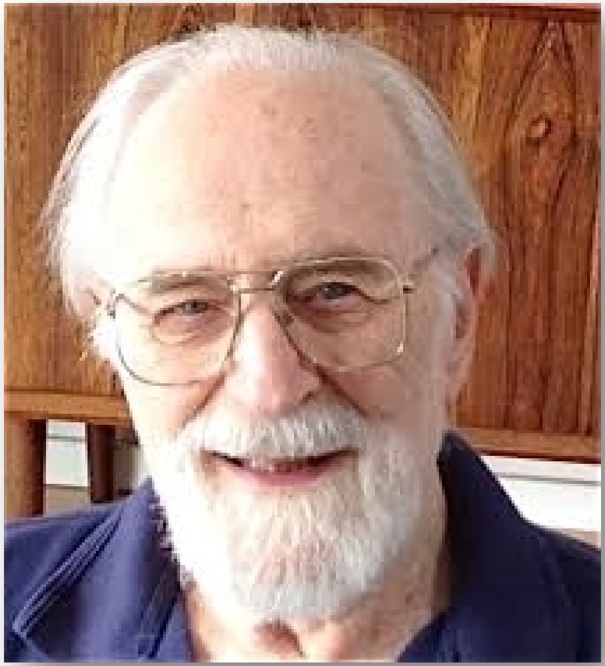
- <u>Solution:</u>

=(X+Y+Z) (X+Y+Z') (X'+Y+Z) (X'+Y+Z') (X'+Y'+Z')

= ?

Simplification Using K-Map

- **Karnaugh Maps**
- The Karnaugh map (K–map), introduced by Maurice Karnaugh in 1953, is a grid-like representation of a truth table which is used to simplify boolean algebra expressions.
- A Karnaugh map has zero and one entries at different positions. It provides grouping together Boolean expressions with common factors and eliminates unwanted variables from the expression.
- In a K-map, crossing a vertical or horizontal cell boundary is always a change of only one variable.



K-Map Simplification

- A Karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest expression possible, known as the minimum expression.
- Karnaugh maps can be used for expressions with two, three, four. and five variables. Another method, called the Quine-McClusky method can be used for higher numbers of variables.
- The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations as is the number of rows in a truth table. For three variables, the number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.

K-Map Simplification

- The 4-Variable Karnaugh Map
- The 4-variable Karnaugh map is an array of sixteen cells,
- Binary values of A and B are along the left side and the values of C and D are across the top.
- The value of a given cell is the binary values of A and B at the left in the same row combined with the binary values of C and D at the top in the same column.
- For example, the cell in the upper right corner has a binary value of 0010 and the cell in the lower right corner has a binary value of 1010.

The 4-Variable Karnaugh Map

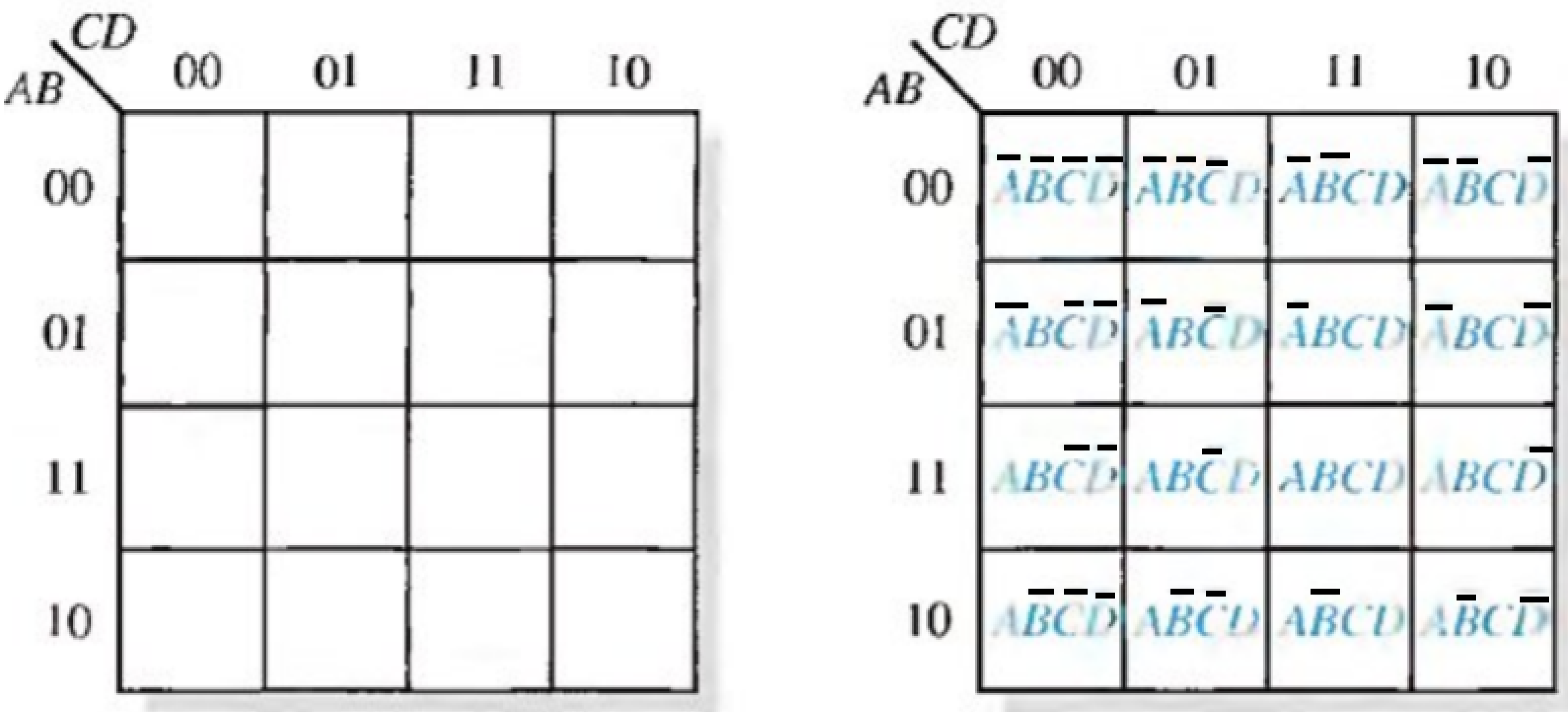
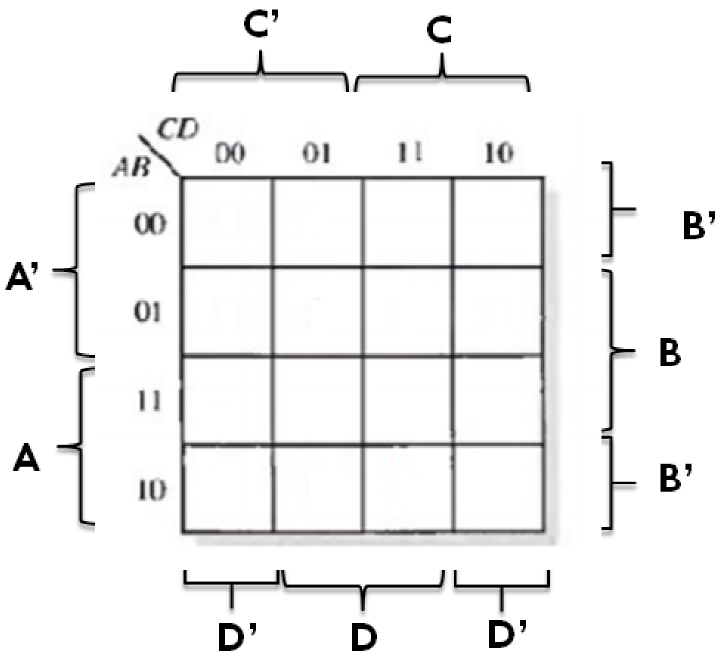
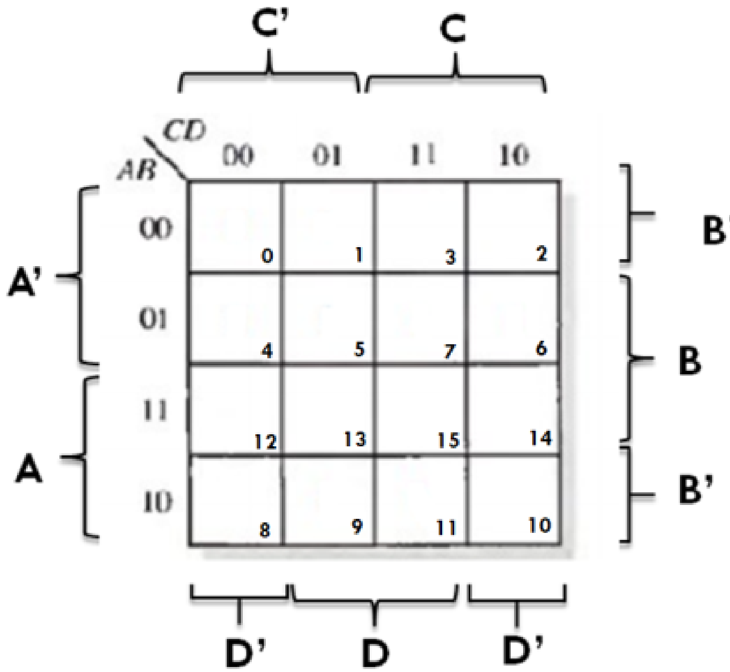


Figure shows the standard product terms that are represented by each cell in the 4-variable Karnaugh map.

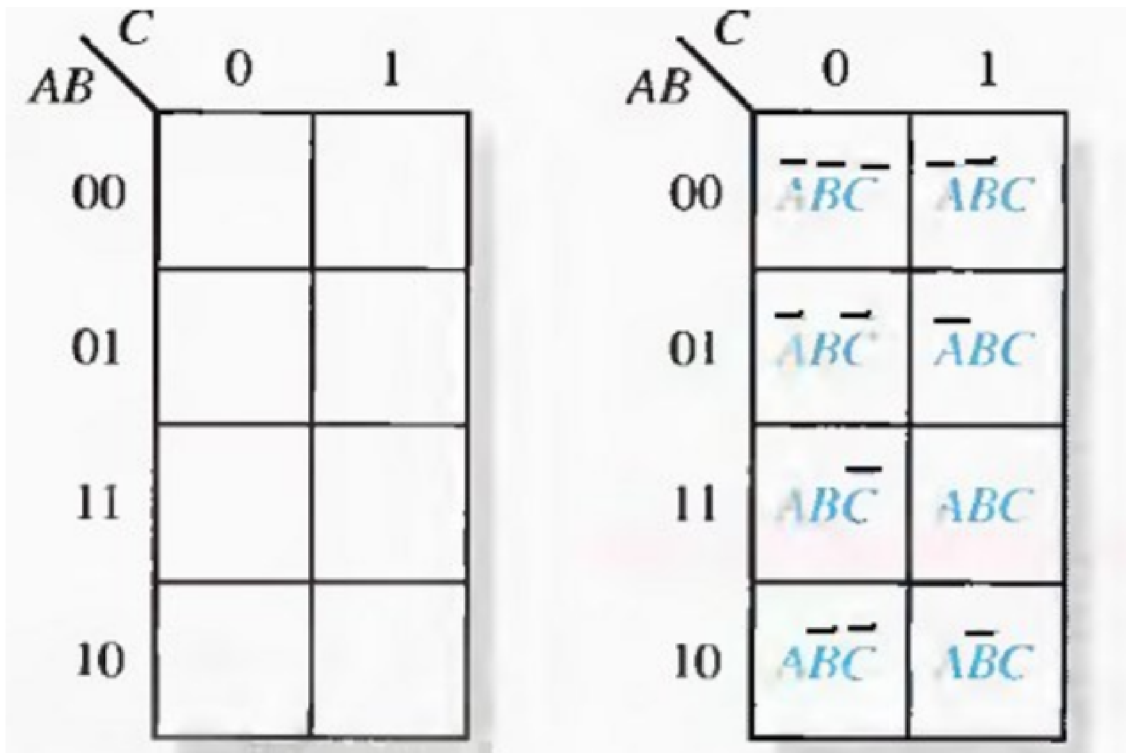
K-Map


C'
C
CD
AB
00
01
11
10
00
01
11
10
A'
A
B'
B
B'
D'
D
D'


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

The 3-Variable Karnaugh Map

- A 3-variable Karnaugh map showing product terms



K-Map Simplification

- **Procedure**
  - After forming the K-Map, enter 1s for the min terms that correspond to 1 in the truth table (or enter 1s for the min terms of the given function to be simplified). Enter 0s for the remaining minterms.
  - Encircle octets, quads and pairs taking in use adjecency, overlapping and rolling. Try to form the groups of maximum number of 1s
  - If any such 1s occur which are not used in any of the encircled groups, then these isolated 1s are encircled separately.
  - Review all the encircled groups and remove the redundant groups, if any.
  - Write the terms for each encircled group.
  - The final minimal Boolean expression corresponding to the K-Map will be obtained by ORing all the terms obtained above

K-Map Simplification – Example 1

- Simplify

F=A’B’C’D’ + A’B’C’D + A’BC’D’ + A’BC’D + A’BCD’ + A’BCD + AB’C’D + AB’CD

Solution:

Step 1: Draw the K-Map and label Properly

Step 2: Fill up the cells by 1s as per the given function which you want to simplify

Step 3: Encircle adjacent 1s making groups of 16, 8, 4 ,2 and single 1’s starting from big to small

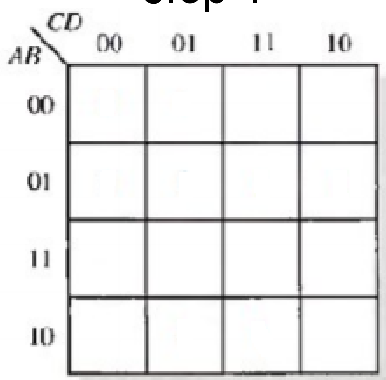
Step 4: write the terms representing the groups

Step 5: The final minimal Boolean expression corresponding to the K-Map will be obtained bu Oring all the terms obtained above

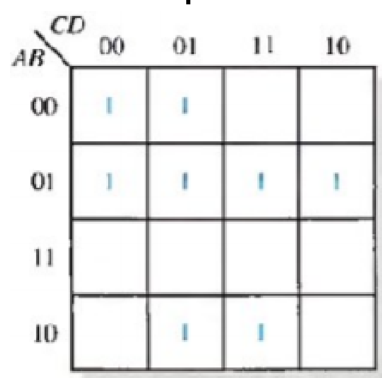
Simplify

F=A’B’C’D’ + A’B’C’D + A’BC’D’ + A’BC’D + A’BCD’ + A’BCD + AB’C’D + AB’CD

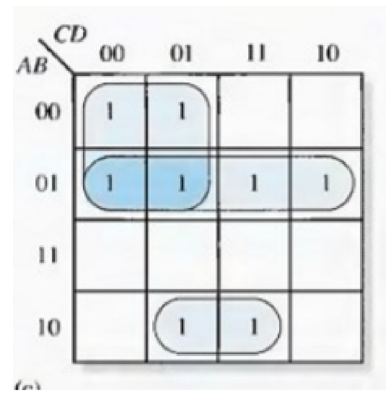
Step 1



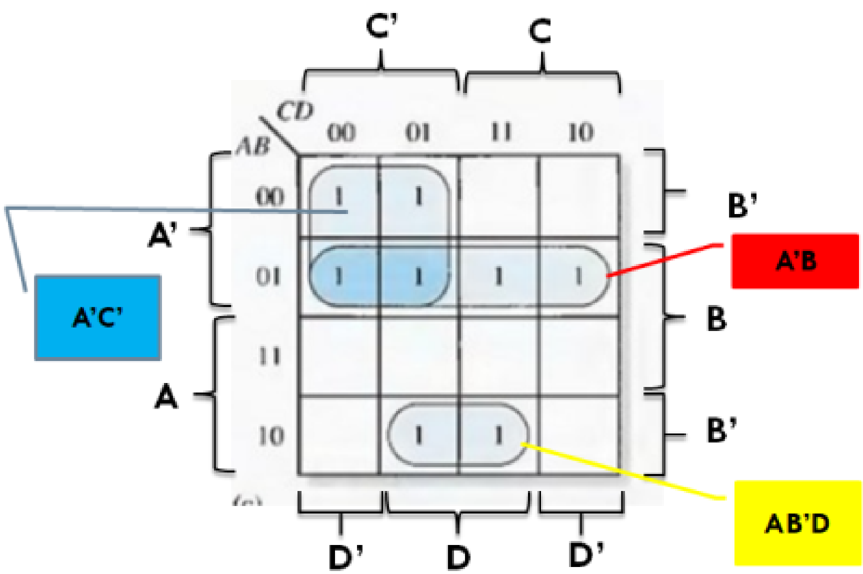
Step 2



Step 3



Step 4



Step 5:

F = A’C’ + A’B + AB’D

K-Map Example 2

- Simplify $F = \overline{B}\,\overline{C} + A\overline{B} + AB\overline{C} + A\overline{B}C\overline{D} + \overline{A}\,\overline{B}\,\overline{C}D + A\overline{B}CD$
- **Solution**

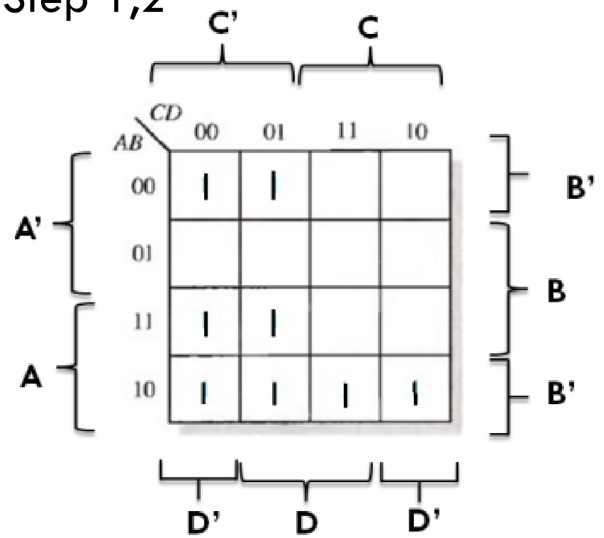
The given expression is obviously not in standard form because each product term does not have four variables.

$\overline{B}\,\overline{C}$	$A\overline{B}$ +	$AB\overline{C}$ +	$A\overline{B}C\overline{D}$ +	$\overline{A}\,\overline{B}\,\overline{C}D$ +	$A\overline{B}CD$
0000	1000	1100	1010	0001	1011
0001	1001	1101			
1000	1010				
1001	1011				

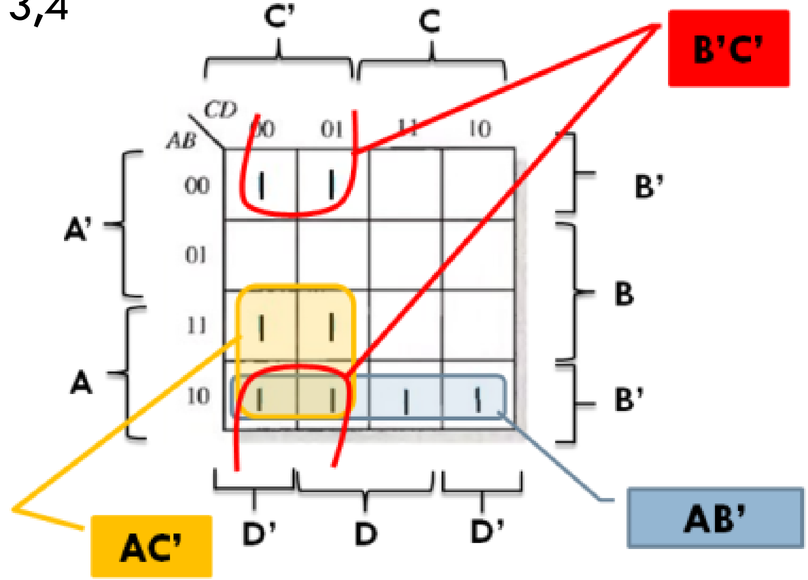
- Map each of the resulting binary values by placing a 1 in the appropriate cell of the 4- variable Karnaugh map.

Simplify: F= $\bar{B}\bar{C} + A\bar{B} + AB\bar{C} + A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}CD$

Step 1,2



Step 3,4



Step 5

F= AB' + AC' + B'C'

K-Map

- For a 4-variable map:
  - 1-cell group yields a 4-variable product term
  - 2-cell group yields a 3-variable product term
  - 4-cell group yields a 2-variable product term
  - 8-cell group yields a 1-variable term
  - 16-cell group yields a value of 1 for the expression
- For a 3-variable map:
  - l-cell group yields a 3-variable product term
  - 2-cell group yields a 2-variable product term
  - 4-cell group yields a 1-variable term
  - 8-cell group yields a value of 1 for the expression

K-Map Example 3

- Simplify the following three variable function

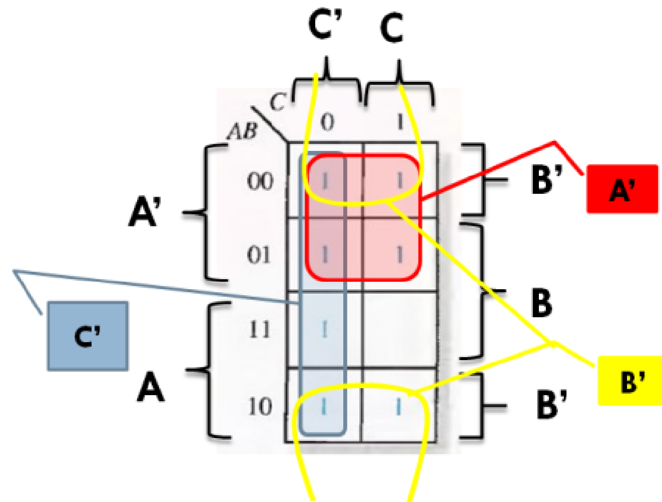
F = A' + AB' + ABC'

Solution:

The given function is not in standard SoP form, so the standard form will be

$\overline{A}$	$+ A\overline{B}$	$+ AB\overline{C}$
000	100	110
001	101	
010		
011		

F= $\sum(0,1,2,3,4,5,6)$



F = A' + B' + C'

K-Map Simplification - Exercise

- Minimize the following function using K-Map

i) $P(A,B,C,D) = \sum(0,1,2,5,8,10,11,14,15)$

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

iii) $S(a,b,c,d) = a'b'c' + b'cd' + a'bc'd + ab'c'd' + ab'cd + acbd' + abcd$

Quine- McCluskey Method

- K-Map Method is a useful tool for the simplification of Boolean function up to four variables. Although this method can be used for 5 or 6 variables but it is not simple to use.
- Another method developed by Quine and improved by McCluskey was found to be good for simplification of Boolean functions of any number of variables.

Self Study

Thankyou