Boolean Algebra

Gagan Deep
Founder & Director
Rozy Computech Services, Kurukshetra-136119
rozygag@yahoo.com
www.rozyph.com

Boolean Algebra

- BOOLEAN Algebra was developed by George Boole (1815- 1864), an English mathematician and logician.
- The credit for applying the laws of Boolean algebra goes to Claude. E. Shannon, an electrical engineer. Claude. E. Shannon in the year 1938, suggested that Boolean algebra can be applied to problems arising in telephone switching circuits and for this reason Boolean algebra is also known as Switching Algebra.
- The other noteworthy persons to realise the significance of this algebra were *August De- Morgan, Alfred North -White Head* and *Beltrand Russell*..

- Perhaps no one at that time might have realised that this abstract algebra was going to result in development of modern high speed digital computers, which have revolutionized the whole world today.
- In fact Boolean algebra placed the theory of switching circuits on firm mathematical footings, which resulted in simplification of these circuits by simple mathematical methods.
- Before discussing the Boolean algebra and its applications to switching circuits, it is important to have the knowledge of the logical statements and various operations that can be performed on these statements, because Boolean algebra deals with these statements

Logical Statements

- A computer may be programmed to make decisions based on certain statement.
- The truth and falsity of statement is know its truth value.
- A statement is either true or false, but not both.
- A statement having a truth value is called **Logical Statement**.
- The truth-values of a logical statement namely, True(T) or False(F) are called **Logical Constants**.

- To understand, the meaning of a Logical Statement. Let us explain it with the following statements.
- a) Please, go to the school.
- b) May God fulfil your desires!
- c) What are you doing?
- d) 2+2=5 e) 2+2=4
- f) Roses are red.
- g) Violets are blue.
- All the above stated statements are meaningful as each one of these conveys a particular meaning. However these statements differ in one respect.

- For example, the statements a), b), c) can not be classified as true or false that means we can not say whether these statements are true or false.
- On the other hand the statements d) is false, and e), f) & g) are true
- In other words, the statements d), e), f) and g) have a truth value which may be true or false.
- Hence the statements d), e), f) and g) are logical statements whereas the statements a), b) and c) are not logical statements. Thus, a logical statement may be defined as a meaningful statement, which has truth value as TRUE or FALSE.
- An exclamatory statement Is not a logical statement, because such a statement has no truth value.

Binary Valued Variables

- In mathematics, the quantities that take different values during the process of manipulation are called variables.
- These variables in mathematics are usually denoted by the letters a, b, c etc. and are capable of taking any numerical values in the irrespective domains.
- *However, the quantities those are capable of taking only two values are called Binary Valued Variables or Binary Variables.*
- The two values taken by these variables are True (T) or False (F).

- Let us consider the statement **a** given by ***a*** : *Is the door open?* The variable a, which stands for the logical statement namely, *Is the door open?*, can have either of the two truth-values namely Open(true) or not open (false). Some more examples
- Is Mohan Sohan's brother.
- Are you married?
- These statements are also binary valued both have output either true or false.

Compound Logical Statement

- Compound Logical Statements are those statements which are composed of sub-statements with the help of logical operator or connectives. e.g.

  a) Mohan is Sohan's brother or Devender's brother.

  b) Roses are red and Violets are blue.

- Statement a) is a compound statement with substatements "Mohan is Sohan's brother" and "He is Devender's brother". These statements are connected with OR logical operator.
- Statement b) is a compound statement with substatements "Roses are Red" and "Violets are blue". These statements are connected with AND logical operator.
- **Now What do you mean by Logical Operator?**

Operators(Connectives)

- The Logical Operators, used to form compound logical statements alongwith the symbols to represent these are listed in the table below:

Logical Operators	Name	Symbol
NOT	Negation or Complementation	' OR ~
AND	Conjunction	^
OR	Disjunction	v
Implication		$\rightarrow$ or $\Rightarrow$
Equivalence		$\leftrightarrow$or $\Leftrightarrow$

- We will discuss only three operators namely NOT, AND and OR because these are the operators to be used in the Boolean algebra.

NOT Operator (Negation or Complement operation)

- If the letter **x** stands for a logical statement then
- The statement **~x** or **x'** (pronounced as ***not x***) means negation of **x**.
- If the statement **x** is true, then the statement **x'** is false.
- Similarly, if the statement **x** is false, then **x'** is a true statement.
- We can say that the logical statements x and x' are logically opposite of each other. e.g., if
- x : Mohan is a Doctor
- then, x' : Mohan is not a Doctor

Truth Tables

- A truth table is a visual aid to represent the truth values of a logical statement.

Truth Table NOT operator

X	X’
T	F
F	T

Truth Tables of AND & OR Operators

AND Operator

X	Y	X∧Y
F	F	F
F	T	F
T	F	F
T	T	T

OR Operator

X	Y	X∨Y
F	F	F
F	T	T
T	F	T
T	T	T

AND Operator(Conjunction Operation)

- If x and y are two logical statements, then
- The statement **x AND y**, symbolically denoted by **x^y** or x.y is called Conjunction of **x** and **y**. e.g. if. x : Roses are red. and y : Violets are blue.
- Then the compound statement :
- **x^y** : Roses are red and Violets are blue.
- Here both the statement **x** and **y** are true, so **x^y** is also true.
- It must be noted that the statement **x^y** is a compound statement because it involves a connective(^). Moreover the truth value of a compound statement depends upon the truth values of its substatements (or constituent statements),also called prime statements, namely x and y.
- **If** x : Roses are red. and y : Violets are green.
- Then the compound statement : **x^y** : Roses are red and Violets are green.
- The statement **x** is true and **y** is false. Therefore, **x^y** is false.
- If both the statements are false then **x^y** are false.
- Thus, it may be concluded from the table that the compound statement x^y is true, if and only if both the substatements x and y are true.

OR Operator (Disjunction Operation)

- If **x** and **y** are two, logical statements, then the statement **x OR y**. symbolically denoted by **x∨y** is called Disjunction of x and y.
- The statement x∨y is true if either of the sub statements x or y or both x and y are true.
- In other words. the compound statement x∨**y** is false, if and only if both the substatements x and y are false, in all other cases it is true.
- Now if, x : Roses are red. and y : Violets are green.
- Then the compound statement : **x∨y** : Roses are red OR Violets are green.
- The statement **x** is true and **y** is false.
- Therefore, **x∨y** is True.
- **If** x : Roses are red. and y : Violets are blue.
- Then the compound statement ; **x∨y** : Roses are red OR Violets are blue.
- Here both the statement **x** and **y** are true, so **x∨y** is also true.
- If both the statements are false than **x∨y** are false.
- It must be concluded from the above truth table that the compound statements x∨**y** is false, if and only if (iff) both the substatements x and y are false and it is true in all other cases.

Boolean Algebra – Switching Algebra

- It must be carefully noted that symbols l or o representing the truth-values of the Boolean variable, have nothing to do with numeric 1 and o respectively. In fact these symbols may be used to represent the active and passive states of a component say a switch or a transistor in an electric circuit.
- A closed switch or **High** or **ON** or **YES** is denoted by 1 and an open switch or **LOW** or **OFF** or **NO** by o. Since, initially Boolean algebra was applied to electric circuits containing switches, it is also known as Switching Algebra.
- Thus a Boolean variable is a logical statement which must either be true or false but not both at the same time. A Boolean variable is usually denoted by an English alphabet (a -z or A-Z).

SWITCHING CIRCUITS

- A Switch (more correctly an electrical switch) is a device that controls the flow of electric current in an electric circuit. The electric switches in the domestic appliances or in household electric circuit performs the same functions i.e. switching ON and switching OFF of an electric current in the circuit.
- Consider a switch, which is either 'closed' or 'open'. This is similar to the situation in Logical Variable where a proposition is either 'true' or 'false'. In Fig. 1(a) we have two switches A and B which are normally open. They are connected in series with a battery and a bulb. The bulb will light only when both A and B are closed. Notice the similarity between this and a compound proposition in which an AND connective is used.
- Now consider the arrangement of switches shown in Fig. 1(b) where two switches C and D are connected in parallel. The bulb will light when either C or D, or both, are closed. We recognize the similarity between this and the truth of a compound proposition when the OR connective is used.

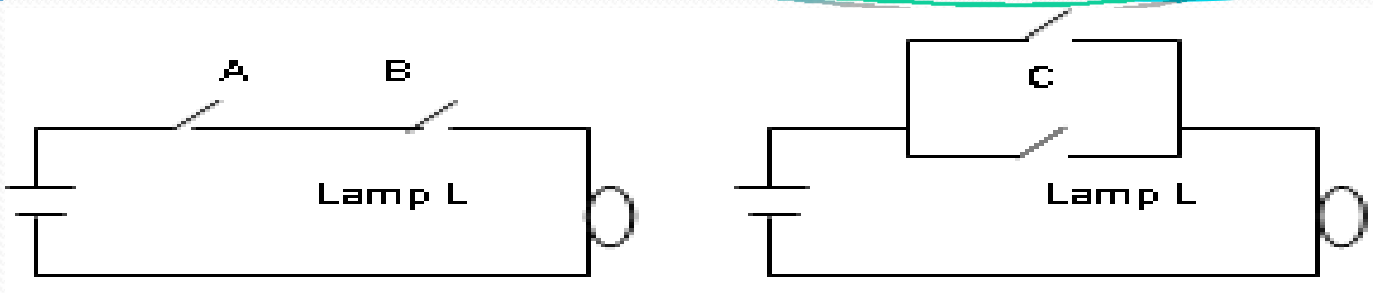


Fig 1 (a) Switches in series; (b) Switches in parallel.

- We can now see a one-to-one correspondence between 'closed' and 'true'. Similarly 'open' corresponds to 'false'. Because of this analog, we call these switching circuits as logic circuits.
- Thus we have switches which can be only open or closed and similarly propositions which are either of two values at one time i.e. 1 or 0.
- We have already observed a one-to-one correspondence between series and parallel connections of switches on the one hand and the connectives AND and OR and logical operator on the other. We can consider these connectives as performing operations on Boolean variables. Let us further represent **AND by the symbol (.) , OR by the symbol (+) and NOT by the symbol (BAR or ')** so that the condition for lighting the bulb in Fig. 1(a) can be written as A.B, and for Fig. 1(b) the condition can be expressed as C + D.

BOOLEAN EQUATION

Let, Y = A.B +C'D(E+F')

- This whole is known as Boolean Equation.

We can also write this as

- Y(A,B,C,D,E,F) = A.B +C'D(E+F')

or

- Y=f(A,B,C,D,E,F) = A.B +C'D(E+F')

Let's divide it in parts

Y=f(A,B,C,D,E,F)

This is read as Y is the Function of A, B, C, D, E, and F. Because in the equation these all variables exists and these variables are known as Boolean Variable.

A.B +C'D(E+F') This is known as expression.

- + . ' are Boolean operators.

POSTULATES

- A two state Boolean Algebra comprises of a set K={0,1}. On which two binary operations namely logical addition called OR operation denoted by the symbol (+), and logical multiplication called AND operation denoted by the symbol (.), alongwith unary operator called NOT operator denoted by the symbol (' or -) are defined and obeys the following

Postulates	Primary	Dual	Name
1	a = 0 iff a ≠ 1	a = 1 iff a ≠ 0	
2.	c=a+b	c=a.b	Closure
3	a +0 = a	a .1 = a	Identity
4	a + b = b + a	a . b = b . a	Commutative
5	a+(b+c)=(a+b)+c	a.(b.c)=(a.b).c	Associative
6	a . (b + c) = a.b + a.c	a + (b.c) = (a+b) . (a + c)	Distributive
7	a + a' = 1	a .a' = 0	Inverse

Theorems

Theorem1	a + a = a	a .a = a
Theorem2	a + 1 = 1	a .0.= 0
Theorem3	a + (a .b) = a	a . (a + b) = a
Theorem4	(a')' = a	
Theorem5	a + a'.b = a + b	a . (a' + b) = a.b
Theorem6	(a + b)' = a' . b'	(a . b)' = a' + b'

Demorgan's First Theorem

It states that the complement of a sum equals to the product of the complement i.e. (a+b+c+.....+z)' = a'.b'.c'.....z' or (a+b)'=a'.b'

Demorgan's Second Theorem

It states that the complement of a product equals to the sum of the complement i.e. (a.b.c.....z)' = a'+b'+c'+.....+z' or (a.b)'=a'+b'

Duality Principle

- The dual of any statement in Boolean algebra, is the statement obtained by interchanging the operation (+) and (.), and interchanging the elements 0 and 1. e.g.

  a.(b+c)=a.b + a.c and dual of the equation is

  a+(b.c)=(a+b).(a+c)

- Also, the dual of the equation a+1=1, is a.0=0
- Observe that we have listed postulates in two parts. One may be obtained from the other, if '+' is interchanged with '.' and '0' is interchanged with '1'. This important property is known as duality principle.
- This principle ensures that **if a theorem is proved** using the postulates, **then a dual theorem** obtained interchanging '+' with '.' and '0' with '1' **automatically holds and need not be proved separately**.

REPRESENTATION OF BOOLEAN EXRESSION

A boolean expression can be represented in either of the following forms.

- Sum of Products (SOP)
- Product of Sums (POS)

SOP FORM OF A BOOLEAN EXPRESSION.

- A Boolean expression which consists of Sum–of-products of various Boolean variables (x,v, z, a, b, c, etc.) either in direct or complemented form, is called Sum-of-products or SOP form of the expression. e.g., the following Boolean expression is in Sum -of- Products form. a. b + b. c
- Here, both the terms a.b and b.c are product terms of Boolean variables a, b and b,c respectively and are logically added. Thus, the given expression is in S-O-P form., Similarly a. b. c + a'. b.c + a. b'. c + a. b. c'

POS form of a Boolean Expression

- A Boolean expression which consists of the product-of-sums of various boolean variables, either in direct or complemented form, is called Product-of-Sums (POS) form of the expression. e.g., the following Boolean expression are in Product-of- Sums form

  (a + b). (b + c)

  (a + b + c).(a' + b + c). (a + b ' + c). (a + b + c')

- The terms within the parentheses are called **Factors of the expression**.
- Before discussing the canonical or normal forms of the SOP and POS forms of a Boolean expression, representing a boolean function, we must be familiar with the *Minterms & Maxterms*

Minterm

- A binary variable may appear either in its normal form (x) or in its complement form (x'). Now consider two binary variables x and y combined with an AND operation is known as *minterm* e.g. x'.y', x'.y, x.y', and x.y. If we have n variables can be form 2^n minterms.

  Consider, the following Boolean function

  f (x, y, z) = x. y.z. + x'. y. z + x. y. z + x. y. z
- The above stated function F (x, y, z) is a function of three Boolean variables, exact term on the right hand side is the product of all the three variables namely x, y, z either in direct or complemented form. Moreover, each of these variables appear only once in each of the product term. Hence, each product term on the right hand side is a minterm. On the other hand, the Boolean function or switching function

  f(x, y) = x + x. y

  is not represented by the minterms because the first term x is not a minterm as it does not contain both the literals.

Maxterms

- Now consider two binary variables x and y combined with an OR operation is known as *maxterm* e.g. x'+y', x'+y, x+y', and x+y. If we have n variables can be form 2^n maxterms.

  Consider the following boolean function,

  f(x, y, z) = (x + y + z). (x' + y + z). (x + y' + z). (x + y + z')
- The above stated function F (x, y, z) is a function of three Boolean variables, Each factor on the right hand side consists of the sum of all the three variables namely x, y, z and these variables appear only once in each factor. Hence each factor on the right hand side is a maxterm.
- On the other hand, the boolean function

  f *(x,y,*z, w) = (x + y'). *(x'* + y' + z + w'). (x + y + ')

  is represented by three factors, namely x + y', x' + y' + z + w' and x + y + z' out of which only the factor x' + y' + z + w', represents a maxterm, other factors are not maxterms.

The following properties of maxterm and minterm must be noted.

- A **minterm** can assume a value '1' for only one combination of variables. It has been stated earlier that a minterm in a SOP form of a Boolean expression is the product of all the variables either in direct or complemented form. The value of the minterm can be '1' only if all the variables in this term assume a true value represented by '1'. For all other values of these variables or variables, this term will have a value '0'.
- A maxterm can assume a value '0', for only one combination of variables. A maxterm in POS form of a Boolean expression is a sum of the variables either in direct or complemented form. This term can have a value '0' only if all the variables in this term assume a false value represented by '0'. For all other values of the variables this term will have a value '1 '.

Minterms and Maxterms for 2 variable

Variables		Minterms		Maxterms	
X	Y	Term	Designation	Term	Designation
0	0	X'.Y'	m_0	X+Y	M_0
0	1	X'.Y	m_1	X+Y'	M_1
1	0	X.Y'	m_2	X'+Y	M_2
1	1	X.Y	m_3	X'+Y'	M_3

Variables			Minterms		Maxterms	
X	Y	Z	Term	Designation	Term	Designation
0	0	0	X'.Y'Z'	m_0	X+Y+Z	M_0
0	0	1	X'.Y'Z	m_1	X+Y+Z'	M_1
0	1	0	X'.Y.Z'	m_2	X+Y'+Z	M_2
0	1	1	X'.Y.Z	m_3	X+Y'+Z'	M_3
1	0	0	X.Y'.Z'	m_4	X'+Y+Z	M_4
1	0	1	X.Y'.Z	m_5	X'+Y+Z'	M_5
1	1	0	X.Y.Z'	m_6	X'+Y'+Z	M_6
1	1	1	X.Y.Z	m_7	X'+Y'+Z'	M_7

Canonical (or Normal) Sum-of-Products form of a Boolean Expression

- A Boolean expression comprising of the sum of minterms is called a Canonical or Normal form of the Sum -of -Products form of the Boolean expression.
- The following Boolean expressions are in canonical SOP forms.

  (i) f (a,b) = a'.b + a.b

  (ii) f(a,b,c) = a.b.c + a.b'.c
- The process of expressing a Boolean expression in canonical SOP form, is called Maximization (**expansion**).

The following steps are followed to express a boolean function in its sum-of-products form :

- Construct a truth table for the given boolean function.
- Form a minterm for each combination of the variable, which produces a 1 in the function.
- The desired expression is the sum (OR) of all the minterms obtained in step 2.

Example

- Obtain canonical sum-of-products form for the Boolean expression A + B

Solution : The given expression is A+B

The truth table for this expression is shown below

- In this truth table the rows 2, 3 and 4 give an output 1. Hence the minterms are to be obtained from these rows.
- The minterm from row 2, is obtairied by replacing the variable A by A ' (because A has value 0, therefore, A' = 1) and taking its product (i.e. ANDing) with B. Thus the minterm is A'.B The minteml from row 3, is obtained by replacing the variable B by B ' (because B has value 0) and taking its product with A. Thus the minterm is A.B'.
- The minterm from row 4, is obtained by taking the product of A with B, because both , have values as 1. Thus the minterm is A.B.
- The required expression is obtained by adding (ORing) an the minterms.

Thus the given expression can be written in canonical SOP form as shown below.

A+B=A'.B+A.B'+A.B

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

Example Obtain the canonical POS forms of the following expression a.b + a'.b’

Solution. The given expression is a.b + a'.b’

The truth table for the given expression is presented below.

- The maxterms are obtained from those rows for which the output is zero i.e. the second and third rows give the maxterms.
- In the second row, the input a has a value 0 and the input b has a value 1. The maxterm is obtained by writing the input a in direct form and the input b in the complemented form and then adding these terms to obtain a+b'.
- Similarly the maxterm obtained from the third row is a' +b.
- The required canonical POS form of the given Boolean expression is
- (a +b’). (a' +b)

a	b	ab+a’b’
0	0	1
0	1	0 (a+b’)
1	0	0 (a’+b)
1	1	1

SIMPLIFICATION OF BOOLEAN FUNCTIONS *(MLNTMIZATION).*

- The cost of fabricating a switching circuit depends upon the number of components used in it. Sometimes It is possible to design switching circuits which require lesser number of components and hence are less expensive compared to the equivalent switching circuits.
- Switching circuits are generally represented by Boolean expressions also known as switching expressions or logical expressions. These expressions are expressed in concise form by Boolean functions. Each variable in a Boolean expression represents a switch.
- The purpose of minimization of a Boolean function is to reduce this function to such a form that it contains minimum number of literals. This form of the Boolean function containing minimum number of variables is called the minima1 form, of the function.

Obviously, a Boolean function in a minimal form when implemented in the form of a switching circuit needs least number of components. Such a circuit is the best one as it is least expensive compared to equivalent switching circuits. A Boolean function in SOP form or POS form is said to be in minimal form if switching expression representing this function possesses one or more of the following characteristics.

- The expression contains the minimum number of variables, a variable may be present in direct or complemented form.
- The expression has the minimum number of terms.
- The expression requires least number of logical units (logical Gates) in its circuit implementation.

- The last characteristic is the most important one and has to be met within practice, as it decides the cost of a switching circuit. While designing a switching circuit, one has to consider the constraints such as type of the logic function, number of inputs and operation speed etc. However, to start with the given function must be reduced to the form which contains least number of variables and also there must be few terms as possible. Some of the methods to reduce a given Boolean function to the minimal form are
- By Boolean Algebra
- By Karnaugh Maps
- By Quine's Method

MINIMIZATION USING BOOLEAN ALGEBRA

Using Postulates Minimise equation

A'.B + A.B+A'.B'

Solution :

=B(A'+A)+A'B' (a(b+c) = a.b +a.c)

=B.1+A'B' (a+a' = 1)

=B+A'B' (a.1=1)

=B+B'A' (a.b=b.a)

=B+A' (a+a'b=a+b)

=B+A'

Using laws of Boolean algebra, prove

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

Solution.

L.H.S. = A.B + A'.C + B.C.1 (A .1 =A)

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

= A.B + A'.C + B.C.A + B.C.A' (Distributlve law)

= A.B + A'.C + A.B.C + A'.B.C (Associatlve law)

= A.B + A.B.C + A'.C + A'.B.C (Commutatlve law)

= A.B. (1 + C) + A'.C (1 + B) (Distributlve law)

=A.B.l +A'.C. 1 (1 +B= 1. 1 +C= 1)

= A.B + A'.C = R.H.S. (A.1= A)

MINIMIZATION USING KARNAUGH MAP(K-MAP) METHOD.

- A better and more elegant way to reduce a Boolean function to the minimal form is with the help of a graphical method called Karnaugh Map method.
- A Karnaugh map(K-Map) is a graphical representation of a boolean function.
- Once a Boolean function is represented by a K-Map, then this function can be reduced to the minimal form in a very easy manner.
- Moreover, this map ensures that further minimzation of the Boolean function is not possible.
- In short, a K-Map provides a systematic mathematical method to reduce a Boolean function to the minimal form.
- A K-Map can be used to simplify Boolean functions having any number of variables but **this map becomes unsuitable and cumbersome for the Boolean functions of more than four variable**

- The map is a diagram made up of squares also called cells.
- The number of cells in this map depends upon the number of variables in the switching function.
- A Karnaugh map for a Boolean function of n variables consists of 2^n squares or cells.
- Each square represents one minterms. Since any boolean function can be expressed as a sum of minterms, it follows that a boolean function is recognised graphically in the map from the area enclosed by those squares whose minterms are included in the function.
- By recognising various patterns, the user can derive alternative algebraic expressions for the same function, from which he can select the simplest one.
- We shall assume the simplest algebraic expression is any one in a sum of products or products of sums that has a minimum number of literals. (This expression is not necessarily unique.)

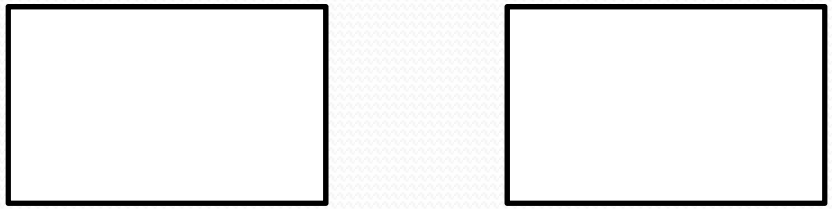
Rules for drawing and Simplification of K-Map

Before drawing a K-map for the given function, **the function must be converted to canonical form, preferably to the SOP form**.

- A K-map is to be drawn in such a manner that the adjacent cells remain logical neighbours. Two cells are called adjacent or neighbouring cells, if these have only one common side between them.
- When no. of variables in function are 2 then the number of cells in the K-map are 2; when variables are 3, cells are 8 and when variables are 4, cells are 16.
- It is permissible to interchange the variable along the horizontal and vertical axis.
- Enter a 1 on the K-map for each fundamental product that corresponds to 1 output in the truth table. Enter 0s elsewhere.
- Encircle the octets(A group of 8 adjacent 1s on a K-map), quads(A group of 4 adjacent 1s on a K-map), and pairs(A group of 2 adjacent 1s on a K-map, these 1s may be horizontally or vertically alligned). Remember to roll and overlap to get the largest group possible. If any 1s remains, encircle them.
- Eliminate redundant groups (A group of 1s on a Kmap all of which are overlapped by other groups) if they exist.
- Write the boolean equation by ORing the products corresponding to the encircled groups.

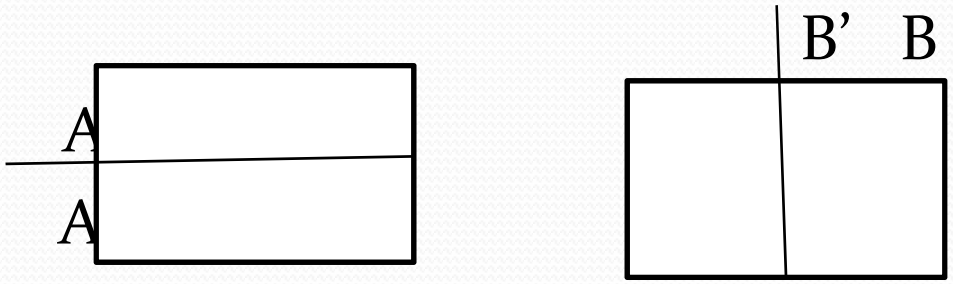
Two Variable K-Map

- As we read earlier that if we have two variables then we have 2^2 minterms i.e. 4 minterms, hence the map consists of four squares.
- Let take Variable A have two states A and A' (i.e. 1 and 0) and B also have two states B and B'. A have total One unit area and B also have total one unit area as shown in figure



A B

For two states these will looks like



After superimposing these two we get four squares for different minterms, as shown in figure.

	B'	B
A'	A'B' 00	A'B 01
A	AB' 10	AB 11

	B'	B
A'	m_0	m_1
A	m_2	m_3

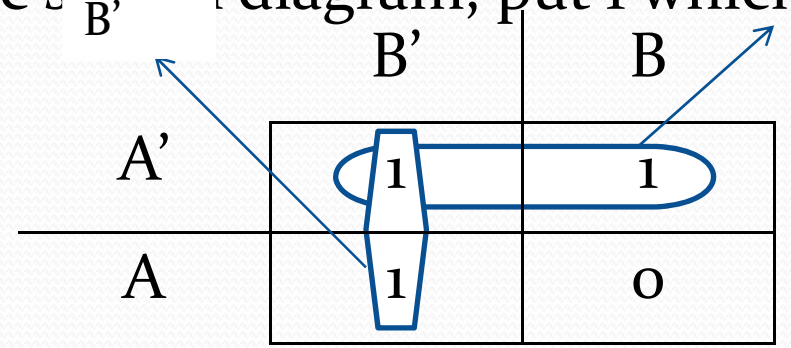
Example : Draw a K-Map to minmise the following boolean function

1. f(A,B)=A'B' +A.B' +A'.B, or = Σ(0,1,2), or = $m_0+m_1+m_2$

2. f(A,B) = A'B' +A.B, or = = Σ(0,3), or = m_0+m_3

Example 1.With the help of above s[illegible] diagram, put 1 which minterm is present

	B'	B
A'	1	1
A	1	0



In Table(where groups are made), We made two groups. Put each groups value and add them

Therefore, we get, =A'+B'

This is the minimisation of A'.B'+A'.B+A.B' = A'+B'

We can also prove it by Boolean algebric expression or by Truth table method.

Example 2. With the help of above shown diagram, put 1 which minterm is present

	B'	B
A'	1	0
A	0	1

	B'	B
A'	1	0
A	0	1

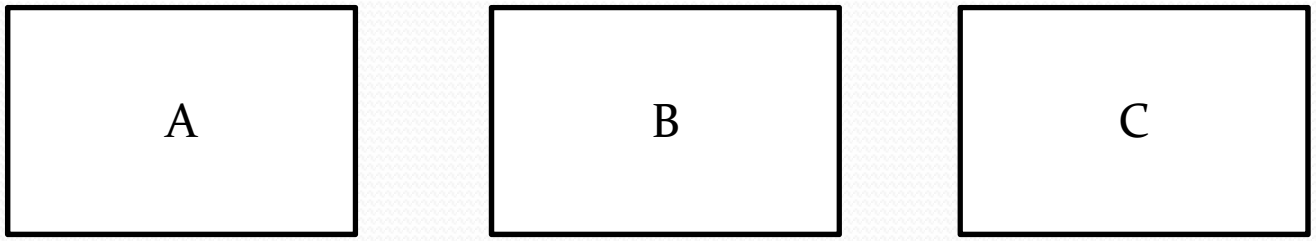
In this example we have only two minterms present, they both are not adjacent or you can say ***they are diagonal***. So, we can't make groups. Therefore, by putting each minterm's value and adding them we get minimised result. A'.B'+A.B

f(A,B) =A'.B'+A.B=A'.B'+A.B

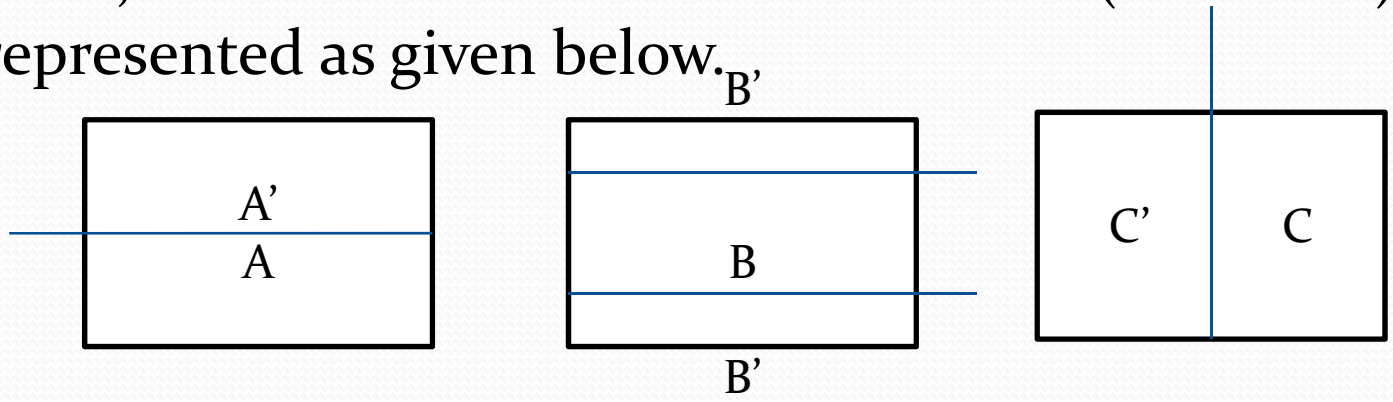
LHS is eqaul to RHS, This shows we can't minimise this expression or you can this is minimised expression.

Three Variable K Map

- Now consider the case of three variable K-Map. In three variable K-Map we have 2^3 i.e. 8 minterms, hence the map consists of eight squares.
- Let us take three variables as A, B & C. All these variables have two states i.e. A & A', B & B' and C & C'. Now All these variables have unit area as shown



- Now, in this unit area both the states (i.e. 1 & 0) can be represented as given below.



- By Superimposing, these three we get eight squares for different mintermes as shown below

	C’	C	
A’	A’B’C’ 000	A’B’C 001	B’
	A’BC’ 010	A’BC 011	B
A	ABC’ 110	ABC 111	
	AB’C’ 100	AB’C 101	B’

	C’	C	
A’	m_0	m_1	B’
	m_2	m_3	B
A	m_6	m_7	
	m_4	m_5	B’

Examples : Design a K-Map to represent the following switching functions.

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

2) F(X,Y,Z)=X'Y'Z' +X'Y'Z+XY'+XY

Solutions : F(A,B,C) = ABC' +AB First covert the expression into canonical SOP

By using above methods discussed in the above SOP section

We get, F(A,B,C)=ABC'+ABC'+ABC F(A,B,C)=ABC'+ABC (B'coz A+A=A)

or F(A,B,C)=$\Sigma(6,7)=m_6+m_7$

Put these in the K-Map by presenting 1 in the squares

	C'	C	
A'	m_0 0	m_1 0	B'
	m_2 0	m_3 0	B
A	m_6 1	m_7 1	
	m_4 0	m_5 0	B'

a

	C'	C	
A'	m_0 0	m_1 0	B'
	m_2 0	m_3 0	B
A	m_6 1	m_7 1	
	m_4 0	m_5 0	B'

AB

b

In Figure b, We made One group of minterm m_6 and m_7. Put each groups value and add them. Therefore, we get,

=AB

This is the minimization of F(A,B,C) = ABC' +AB = AB

- We can also prove it by Boolean algebric expression or by Truth table method.

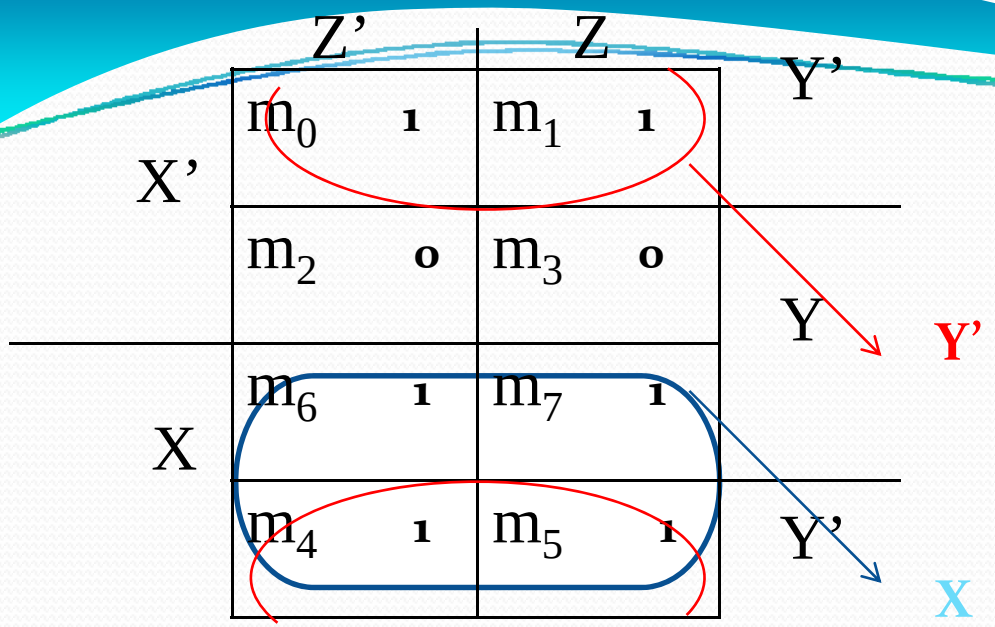
Solutions 2) F(X,Y,Z)=X'Y'Z' +X'Y'Z+XY'+XY

- First covert the expression into canonical SOP
- By using methods discussed in the above SOP section

We get, F(X,Y,Z)= X'Y'Z' +X'Y'Z+XY'Z+ XY'Z'+XYZ+ XYZ'

or F(X,Y,Z)= Σ(0,1,4,5,6,7)

Put these in the K-Map by presenting 1 in the squares



We made two groups Blue and Red of minterm

Put each groups value and add them

Red Line Group (m_1, m_2, m_4, m_5) is equal Y'

Blue Line Group (m_4, m_5, m_6, m_7) equal to the value X

Therefore, we get, =X+Y'

This is the minimisation of F(X,Y,Z)=X'Y'Z' +X'Y'Z+XY'+XY

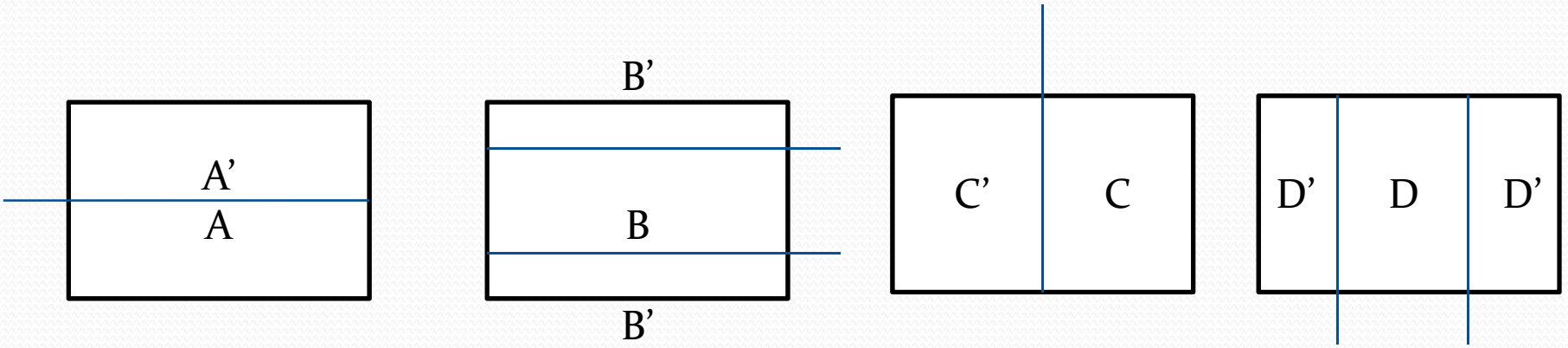
=X+Y'

We can also prove it by Boolean algebraic expression or by Truth table method.

Four Variable K-Map

Now consider the case of Four variable K-Map. In four variable K-Map we have 2^4 i.e. 16 minterms, hence the map consists of 16 squares.

- Let us take Four variables as A, B, C & D. All these variables have two states i.e. A & A', B & B' , C & C' and D & D'. Now All these variables have unit area with both states as shown



	C'		C		
A'	A'B'C'D' 0000 m_0	A'B'C'D 0001 m_1	A'B'CD 0011 m_3	A'B'CD' 0010 m_2	B'
	A'BC'D' 0100 m_4	A'BC'D 0101 m_5	A'BCD 0111 m_7	A'BCD 0110 m_6	B
A	ABC'D' 1100 m_{12}	ABC'D 1101 m_{13}	ABCD 1111 m_{15}	ABCD' 1110 m_{14}	
	AB'C'D' 1000 m_8	AB'C'D 1001 m_9	AB'CD 1011 m_{11}	AB'CD' 1010 m_{10}	B'
	D'	D		D'	

Examples Develop K-map for minimisation of Boolean expression. or Obtain the minimal SOP form of the following by K-Map method.

1). F(ABCD) = A'B'C'D +A'BC'D + A'BCD' +A'BCD+ABCD' +ABCD

2) F(A,B,C,D) = Σ(0,1,2,4,5,8,9,10,12,13)

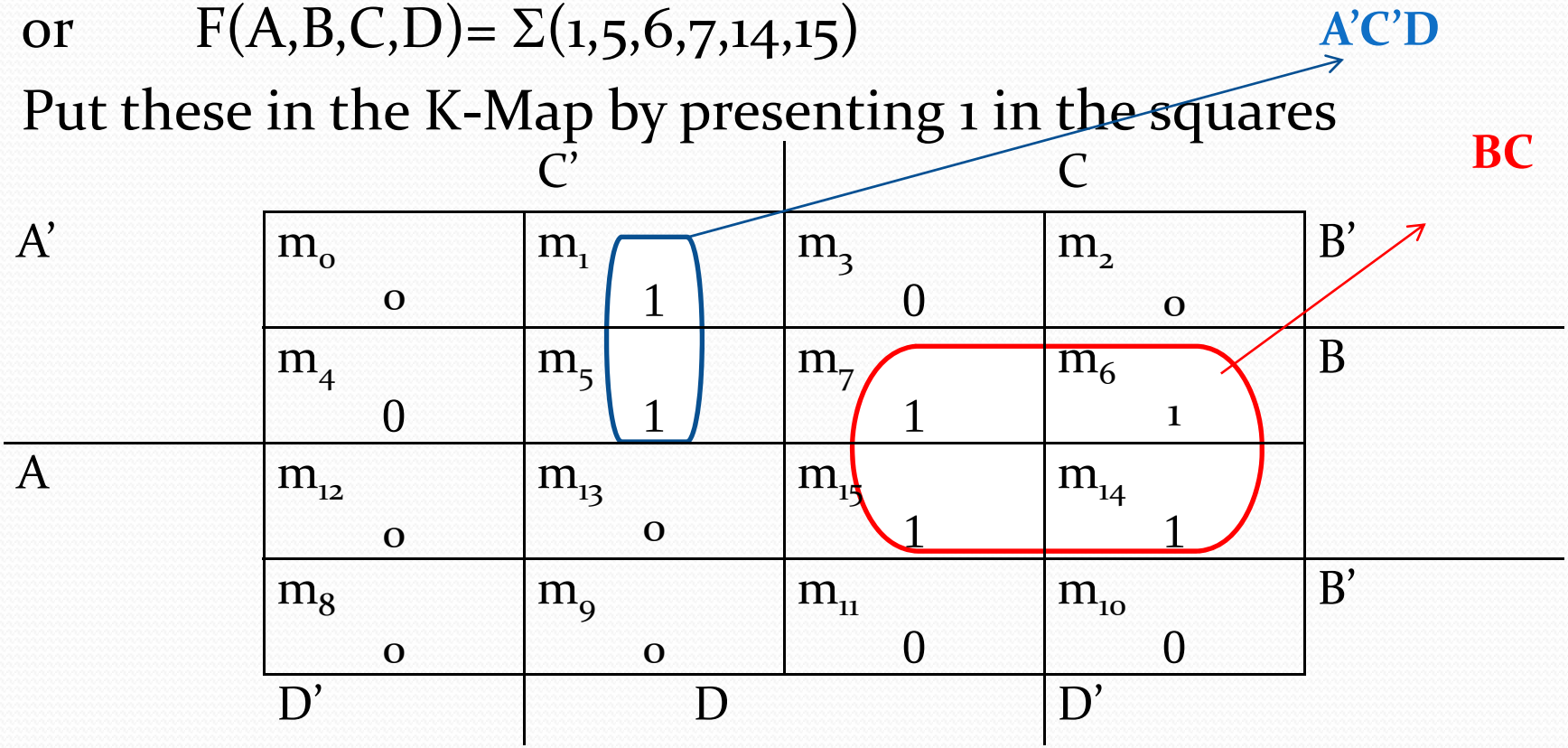
Solution 1

1) F(ABCD) = A'B'C'D +A'BC'D + A'BCD' +A'BCD+ABCD' +ABCD

First covert the expression into canonical SOP. This is already in the SOP

or F(A,B,C,D)= Σ(1,5,6,7,14,15)

Put these in the K-Map by presenting 1 in the squares



In Figure, We made two groups of minterm

Put each groups value and add them

Red Group (m_6, m_7, m_{14}, m_{15}) is equal BC

Blue Group (m_1, m_5) equal to the value A'C'D

Therefore, we get, =BC+A'C'D

This is the minimisation of

F(ABCD) = A'B'C'D +A'BC'D + A'BCD'

+A'BCD+ABCD' +ABCD = BC+A'C'D

We can also prove it by Boolean algebric expression or by Truth table method.

2nd Problem is for your home work

Thanks!

If you have any problem Write me at

rozygag@yahoo.com

www.rozyph.com