Unit – II

Objectives:

- To familiarize with the concepts of Boolean algebra and its minimization.

Syllabus:

MINIMIZATION TECHNIQUES: Boolean Theorems, & duality, De-Morgan Theorems , Minimization of logic functions using Boolean theorems, Minimizations of switching Functions using K-Maps up to 6 variables, Tabular minimization, problem solving (Code-converters using K-Map etc..).

Outcomes:

Students will be able to

- understand basic theorems and properties of Boolean algebra.
- minimize logic functions using Boolean theorems.
- determine the minimized Boolean function using K-maps and Tabular methods..

Learning Material

Boolean Theorems:

Boolean algebra:

Switching circuits called Logic circuits, gate circuits & digital circuits. Switching algebra called Boolean Algebra. Boolean algebra is a system of mathematical logic. It is an algebraic system consisting of the set of element (0,1) two binary operators called OR & AND & One unary operator NOT.

A+A=A , A.A=A because variable has only a logicvalue.

Complementation Laws:

Complement means invert(0' as 1 & 1' as0)

Law1:0'=1

Law2:1'=0

Law3:If A=0 then A' =1

Law4:If A=1 then A' =0

Law5: (A')' =A(double complementation law)

AND laws:

Law 1: A.0=0(Null law)

Law 2:A.1=A(Identity law)

Law 3:A.A=A

Law 4:A.A' =0

OR laws:

Law 1: A+0=A(Null law)

Law 2:A+1=1

Law 3:A+A=A

Law 4:A+ =0

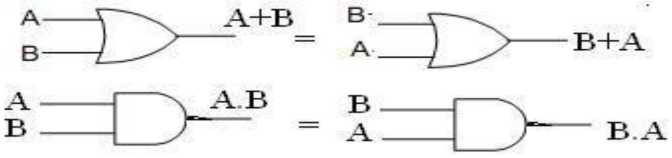
Commutative laws: allow change in position of AND or OR variables.2 commutative laws

Law 1: A+B=B+A

Law 2: A.B=B.A

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

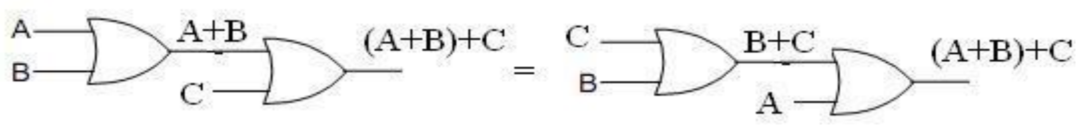
A.B	B.A
0	0
0	0
0	0
1	1



Associative laws: This allows grouping of variables. It has 2 laws.

Law 1: (A+B)+C=A+(B+C) =A OR B ORed with C

This law can be extended to any no. of variables
(A+B+C)+D=(A+B+C)+D=(A+B)+(C+D)

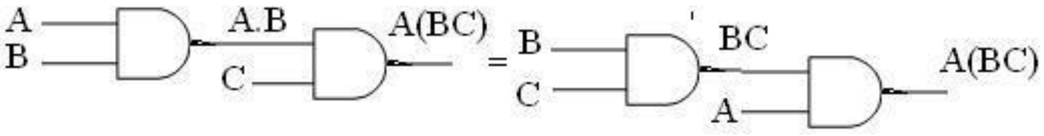


A B C	A+B	(A+B)+C
0 0 0	0	0
0 0 1	0	1
0 1 0	1	1
0 1 1	1	1
1 0 0	1	1
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

=

A B C	B+C	A+(B+C)
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	1	1
1 0 0	0	1
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

Law 2: (A.B).C=A(B.C)
This law can be extended to any no. of variables
(A.B.C).D=(A.B.C).D



A B C	AB	(AB)C
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	0	0
1 0 0	0	0
1 0 1	0	0
1 1 0	1	0
1 1 1	1	1

=

A B C	BC	A(BC)
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	0
1 0 0	0	0
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

Distributive Laws:

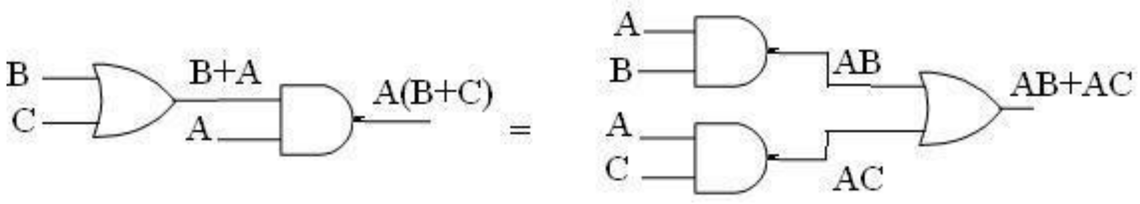
This has 2 laws

Law 1. A(B+C)=AB+AC

This law applies to single variables.

Ex:ABC(D+E)=ABCD+ABCE

AB(CD+EF)=ABCD+ABEF



A B C	B+C	A(B+C)
0 0 0	0	0
0 0 1	1	0
0 1 0	1	0
0 1 1	1	0
1 0 0	0	0
1 0 1	1	1
1 1 0	1	1
1 1 1	1	1

=

A B C	AB AC	AB+AC
0 0 0	0 0	0
0 0 1	0 0	0
0 1 0	0 0	0
0 1 1	0 0	0
1 0 0	0 0	0
1 0 1	0 1	1
1 1 0	1 0	1
1 1 1	1 1	1

Law 2. A+BC=(A+B)(A+C) RHF=(A+B)(A+C)

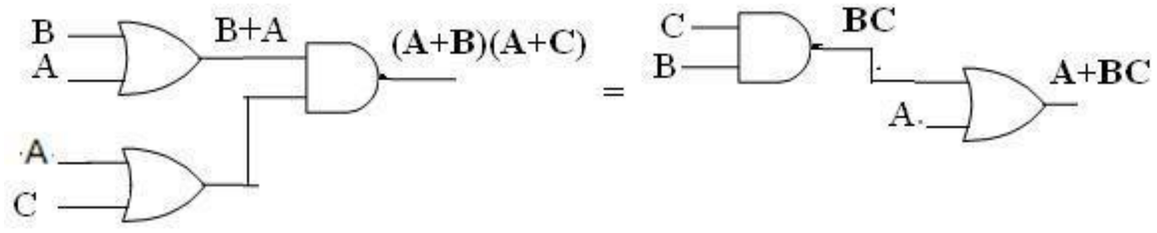
=AA+AC+BA+BC

=A+AC+AB+BC

=A(1+C+B)+BC

=A.1+BC

=A+BC LHF



A B C	BC	A+BC
0 0 0	0	0
0 0 1	0	0
0 1 0	0	0
0 1 1	1	1
1 0 0	0	1
1 0 1	0	1
1 1 0	0	1
1 1 1	1	1

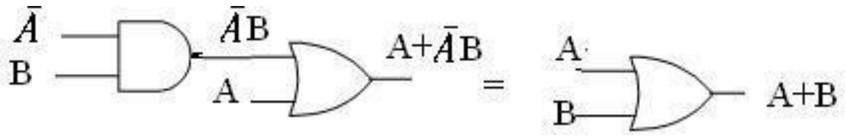
A B C	A+B	A+C	(A+B)(A+C)
0 0 0	0	0	0
0 0 1	1	1	0
0 1 0	1	0	0
0 1 1	1	1	0
1 0 0	0	1	0
1 0 1	1	1	1
1 1 0	1	1	1
1 1 1	1	1	1

Redundant Literal Rule(RLR):

Law 1: A+ A'B=A+B

LHF = (A+A')(A+B)
=1.(A+B)
=A+B RHF

Performing OR operation of a variable with the AND of the compliment of that variable with another variable, is equal to the Performing OR operation of the two variables.



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

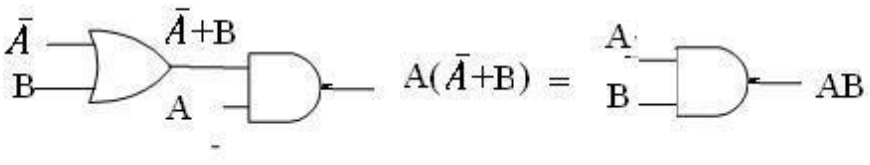
=

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

Law 2: A (A'+B) = AB

LHF = A.A' + AB
= 0+AB
=AB RHF

Performing AND operation of a variable with the OR of the complement of that variable with another variable, is equal to the performing AND operation of the two variables.



A B	A’+B	A(A’+B)
0 0	1	0
0 1	1	0
1 0	0	0
1 1	1	1

=

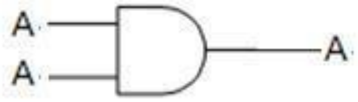
A	B	A+B
0	0	0
0	1	0
1	1	0
1	1	1

Idempotent Laws:

Idempotent means same value. It has 2 laws.

Law 1=A.A=A

This law states performing AND operation of a variable with itself is equal to that variable only

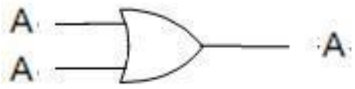


If A=0, then A.A=0.0=0=A

If A=1, then A.A=1.1=1=A

Law 2: A+A=A

This law states that performing OR operation of a variable with itself is equal to that variable only.



If A=0, then A+A=0+0=0=A

If A=1, then A+A=1+1=1=A

Absorption Laws:

Law 1=A+A.B=A

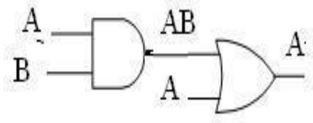
= A(1+B)

=A.1

=A

i.e., A+A. any term=A

A B	A.B	A+(A.B)
0 0	0	0
0 1	0	0
1 0	0	1
1 1	1	1



Law 2=A(A+B)=A

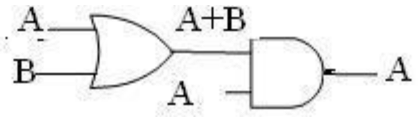
A(A+B)=A.A+A.B

= A+AB

=A(1+B)

= A.1 =A

A B	+	A(A+B)
0 0	0	0
0 1	1	0
1 0	1	1
1 1	1	1



Transposition Theorem:

$AB+ A'C=$

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

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

$=AA' +CA'+AB+CB$

$=0+ A'C+AB+BC$

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

$=AB+ABC+ C+ BC =AB+ C$ LHS

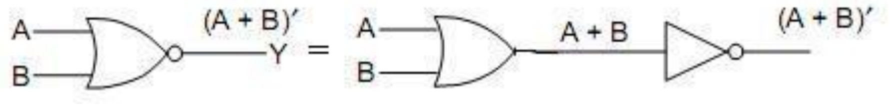
DeMorgan’s Theorem:

It represents two of the most powerful laws in Boolean algebra

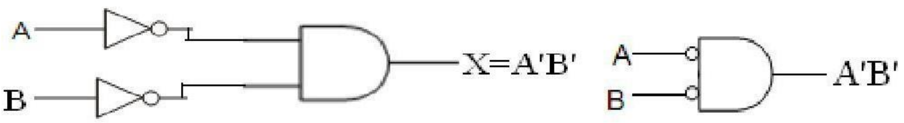
Law 1: $(A+B)' = A'.B'$

This law states that the compliment of a sum of variables is equal to the product of their individual complements.

LHS



RHS



NOR gate= Bubbled AND gate

This can be extended to any variables. $(A+B+C+D+-----)' =A'B'C'D'----$

Law 2: $(AB)'=A'+B'$

Complement of the product of variables is equal to the sum of their individual components.

Duality:

In a positive Logic system the more positive of the two voltage levels is represented by a 1 & the more negative by a 0. In a negative logic system the more positive of the two voltage levels is represented by a 0 & more negative by a 1. This distinction between positive &negative logic systems is important because an OR gate in the positive logic system becomes an AND gate in the negative logic system &vice versa. Positive & Negative logics give a basic duality in Boolean identities. Procedure dual identity by changing all + (OR) to. (AND) & complementing all 0„s &1„s. Once a theorem or statement is proved, the dual also thus stands proved called Principle of duality.

Relations between complement (A+B+C+….)

‘=A’.B’.C’ ….

(A.B.C.….) ‘= A’ + B’ + C’ + ….

Duals:

Expression	Dual
0=1	1=0
0.1=0	1+0=1
0.0=0	1+1=1
1.1=1	0+0=0
A.0=0	A+1=1
A.1=A	A+0=A
A.A=A	A+A=A
A.A’ =0	A+A’ =1
A.B=B.A	A+B=B+A
A.(B.C)=(A.B).C	A+(B+C)=(A+B)+C
A.(B+C)=(AB+AC)	A+BC=(A+B)(A+C)
A(A+B)=A	A+AB=A
A.(A.B)=A.B	A+A+B=A+B
(A+B)(A’+C)(B+C)=(A+B)(A’+C)	AB+ A'C+BC=AB+ A'C

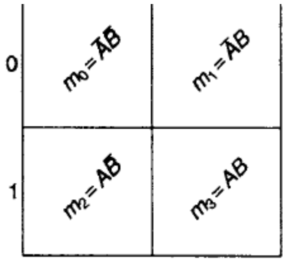
Minimization of switching functions using K (Karnaugh) -Maps

- The K-map is a diagram made up of squares.
- Each square represents one minterm. Since any function can be expressed as a sum of minterms, it follows that a Boolean function can be recognized from a map by the area enclosed by those squares, whose minterms are included in the operation.
- By various patterns, we can derive alternative algebraic expression for the same operation, from which we can select the simplest one. (One that has minimum number of literals).
- Construct the K-map as discussed above. Enter 1 in those squares corresponding to the minterms for which function value is 1. Leave empty the remaining squares. Now in following steps the square means the square with a value 1.
- Examine the map for squares that cannot be combined with any other squares and form group of such single squares.
- Now, look for squares which are adjacent to only one other square and form groups containing only two squares and which are not part of any group of 4 or 8 squares. A group of two squares is called a pair.
- Next, group the squares which result in groups of 4 squares but are not part of an 8-squares group. A group of 4 squares is called a quad.
- Group the squares which result in groups of 8 squares. A group of 8 squares is called octet.
- Form more pairs, quads and outlets to include those squares that have not yet been grouped, and use only a minimum no. of groups. There can be overlapping of groups if they include common squares.

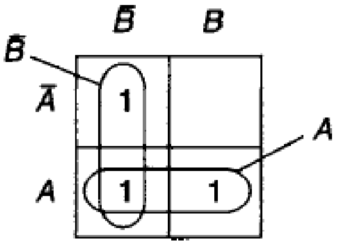
- Omit any redundant group.
- Form the logical sum of all the terms generated by each group.
- Using Logic Adjacency Theorem we can conclude that,
  - a group of two squares eliminates one variable,
  - a group of four squares eliminates two variable and a group of eight squares eliminates three variables.
- There are two, three and four variable K maps.

Two Variable K-Map:

- For two variables there are four minterms and these can be conveniently placed on a 'map' as shown in figure below



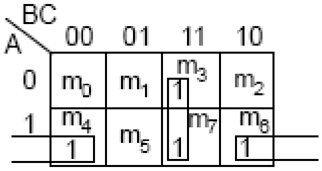
-
-
- The map consists of a square divided into four cells, one for each of the minterms.
- The possible values of the variable A are written down the left hand side of the map, labeling the corresponding rows of the map, while the possible values of the variable B are written along the top of the map, labeling the corresponding columns of the map.
- Hence, the top left-hand cell represents the minterm where A=0 and B=0, i.e. the minterm AB.
- The bottom right-hand cell represents the minterm AB where A=1and B=1.
- The process of simplifying a Boolean function with the aid of a K-map is simplya process of finding adjacencies on the function plot.
- This is best explained with the aid of a very simple example.
- Suppose that it is required to simplify the Boolean function f = A’B’+ AB’ + AB.
- Using Boolean algebra alone, it can be readily found that F=B(‘ + A) + AB = AB + B’
- However, suppose that F is plotted on a 2-variable K-map, as in Figure below.



- The next stage of the simplification process is to group together adjacent cells containing 1‟s. (In this context, note carefully that 'adjacent' means 'horizontally or vertically', *not* 'diagonally'.)
- Therefore, the bottom two cells, corresponding to A alone, may be grouped together.
- Similarly, the two left-hand cells, corresponding to B alone, may also be grouped together, as indicated in the figure above.
- The final stage is to write down the final simplified expression for the function obtained from the groupings thus identified. In this case, therefore, f = A + B’.

Three Variable K-Map:

- If the following Boolean function F (A, B, C) = Σ (3, 4, 6, 7).Then it is represented in k-map as shown in figure below:

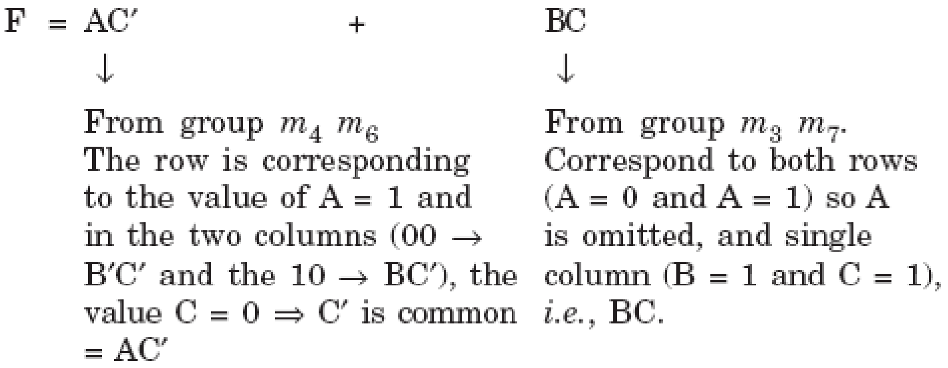


Step 1. *m*3 is adjacent to *m*7. It forms a group of two squares and is not a part of any group of 4 or 8 squares. Similarly *m*6 is adjacent to *m*7. So this is second group (pair) that is not a part of any group of 4 or 8 squares. Now according to new definition of adjacency *m*4 and *m*6 are also adjacent and form apair. Moreover, this pair (group) is not a part of any group of 4 or 8 squares.

Step 2. All the 1"s have already been grouped.

Step 3. The pair formed by *m*6 *m*7 is redundant because *m*6 is already covered in pair*m*4 *m*6 and *m*7 in pair *m*3 *m*7. Therefore, the pair *m*6 *m*7 is discarded.

Step 8. The terms generated by the remaining two groups are „OR" operated together to obtain the expression for F as follows:



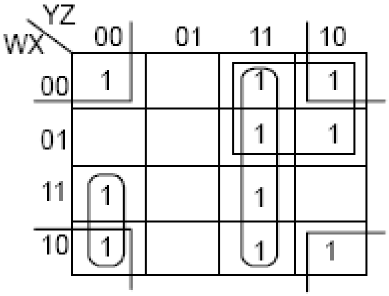
$$F = AC' + BC$$

From group $m_4\ m_6$
The row is corresponding to the value of A = 1 and in the two columns (00 → B′C′ and the 10 → BC′), the value C = 0 ⇒ C′ is common = AC′

From group $m_3\ m_7$.
Correspond to both rows (A = 0 and A = 1) so A is omitted, and single column (B = 1 and C = 1), *i.e.*, BC.

Four Variable K-Map:

- If the following Boolean function F(*w*, *x y*, *z*) = Σ (0, 2, 3, 6, 7, 8, 10, 11, 12, 15), then the K- map is given in the figure below

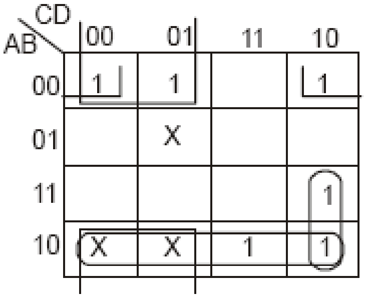


- Minterm 8 and 12. From a pair.
- Minterms 0, 2, 8 and 10 form I quad.
- Minterms 3, 7, 11, 15 form II quad.
- Minterms 2, 3, 6, 7 form III quad.
- Therefore the final minimized expression is given by

$$F = \underset{\substack{\downarrow \\ \text{Due to} \\ \text{I quad.}}}{x'z'} + \underset{\substack{\downarrow \\ \text{II quad}}}{yz} + \underset{\substack{\downarrow \\ \text{III quad}}}{w'y} + \underset{\substack{\downarrow \\ \text{Due to pair}}}{wy'z'}$$

Don’t care map entries:

- The occurrence of particular input combinations will have no effect on the output, then those inputs are known as don’t cares.
- That is a d or a × (cross) is entered into each square to signify “don'tcare” MIN/MAX terms.
- Simplify the following Boolean function. F (A, B, C, D) = Σ (0, 1, 2, 10, 11, 14)&d (5, 8, 9)

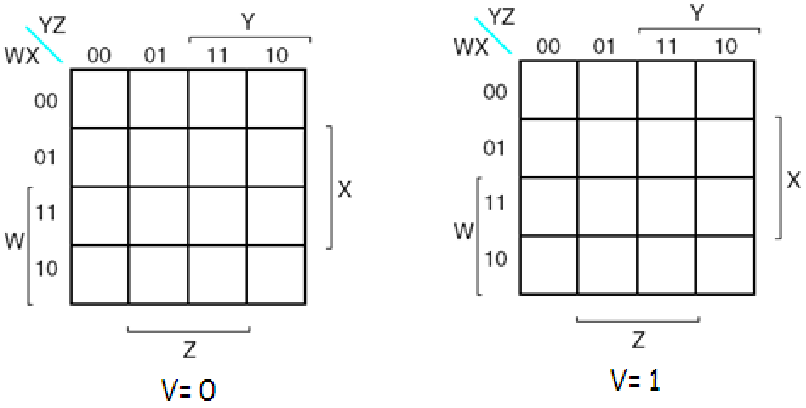


AB \ CD	00	01	11	10
00	1	1		1
01		X		
11				1
10	X	X	1	1

- As shown in K-map in Figure above, by combining 1‟s and d‟s(Xs), three quads can be obtained.
- The X in square 5 is left free since it does not contribute in increasing the size of any group. Therefore, the
  - I Quad covers minterms 0, 2, 10 and d8
  - II Quad covers minterms 10, 11 and d8, d9.
  - III Quad covers minterms 0, 1 and d8, d9.
  - A pair covers minterms 10 and 14.
- Therefore the final expression is

$$F = \underset{\substack{\text{Due} \\ \text{to I} \\ \text{quad.}}}{B'D'} + \underset{\substack{\text{II} \\ \text{quad}}}{AB'} + \underset{\substack{\text{III} \\ \text{Quad}}}{B'C'} + \underset{\substack{\text{Due} \\ \text{to} \\ \text{Pair.}}}{ACD'}$$

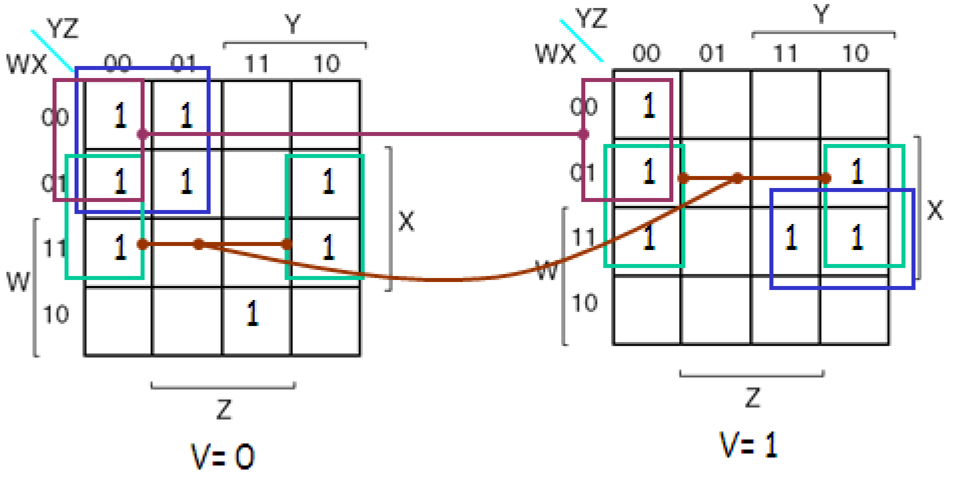
Five-variable K-maps – f(V,W,X,Y,Z)



			Y		
	m_0	m_1	m_3	m_2	
	m_4	m_5	m_7	m_6	X
W	m_{12}	m_{13}	m_{15}	m_{14}	
	m_8	m_9	m_{11}	m_{10}	
		Z			

			Y		
	m_{16}	m_{17}	m_{19}	m_8	
	m_{20}	m_{21}	m_{23}	m_{22}	X
W	m_{28}	m_{29}	m_{31}	m_{30}	
	m_{24}	m_{25}	m_{27}	m_{26}	
		Z			

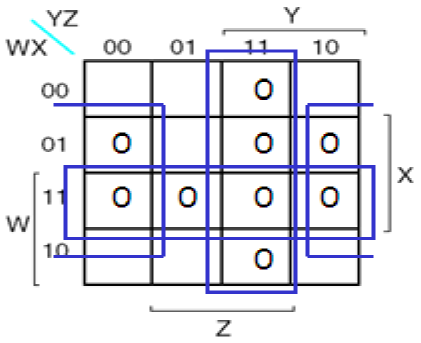
Simplify f(V,W,X,Y,Z)=Σm(0,1,4,5,6,11,12,14,16,20,22,28,30,31)



f = XZ'	Σm(4,6,12,14,20,22,28,30)
+ V'W'Y'	Σm(0,1,4,5)
+ W'Y'Z'	Σm(0,4,16,20)
+ VWXY	Σm(30,31)
+ V'WX'YZ	m11

F(W,X,Y,Z)= Σm(0,1,2,5,8,9,10)

= ∏ M(3,4,6,7,11,12,13,14,15)



WX \ YZ	00	01	11	10
00			0	
01	0		0	0
11	0	0	0	0
10			0	

F(W,X,Y,Z)= (W' + X')(Y' + Z')(X' + Z)

Or,

F(W,X,Y,Z)= X'Y' + X'Z' + W'Y'Z

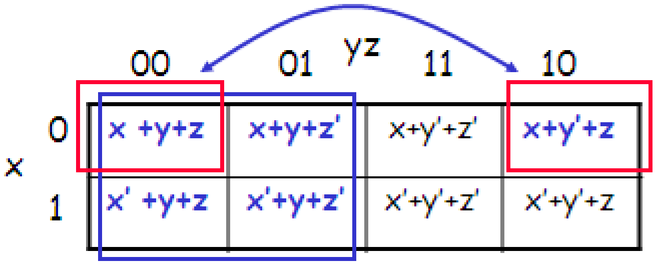
Which one is the minimal one?

PoS Optimization

- Maxterms are grouped to find minimal PoS expression

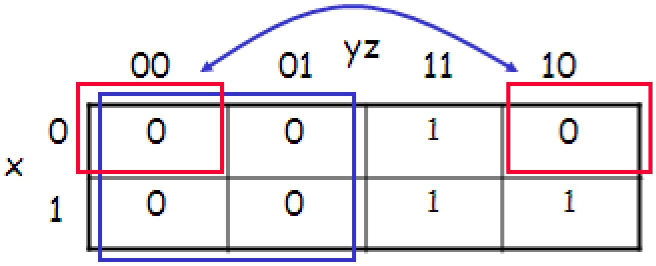
x \ yz	00	01	11	10
0	x +y+z	x+y+z'	x+y'+z'	x+y'+z
1	x' +y+z	x'+y+z'	x'+y'+z'	x'+y'+z

- F(W,X,Y,Z)= ∏ M(0,1,2,4,5)



x \ yz	00	01	11	10
0	x +y+z	x+y+z'	x+y'+z'	x+y'+z
1	x' +y+z	x'+y+z'	x'+y'+z'	x'+y'+z

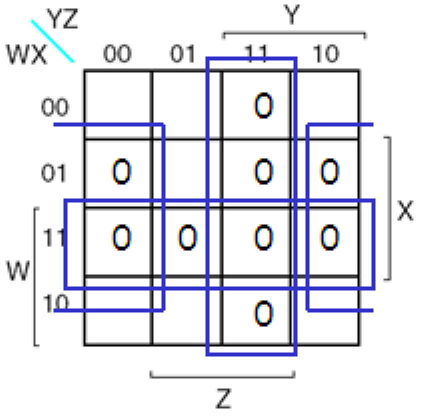
F(W,X,Y,Z)= Y . (X + Z)



x \ yz	00	01	11	10
0	0	0	1	0
1	0	0	1	1

PoS Optimization from SoP

$$F(W,X,Y,Z)=\Sigma m(0,1,2,5,8,9,10)$$
$$=\prod M(3,4,6,7,11,12,13,14,15)$$



WX \ YZ	00	01	11	10
00			0	
01	0		0	0
11	0	0	0	0
10			0	

$F(W,X,Y,Z)=(W'+X')(Y'+Z')(X'+Z)$

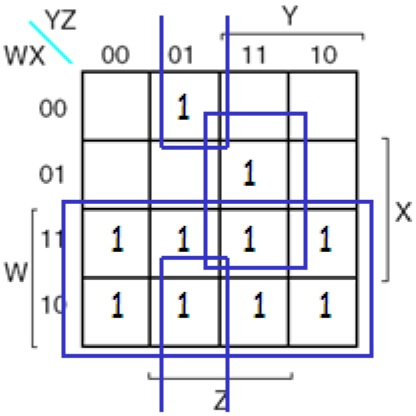
Or,

$F(W,X,Y,Z)=X'Y'+X'Z'+W'Y'Z$

Which one is the minimal one?

SoP Optimization from PoS

$$F(W,X,Y,Z)=\prod M(0,2,3,4,5,6)$$
$$=\Sigma m(1,7,8,9,10,11,12,13,14,15)$$



WX \ YZ	00	01	11	10
00		1		
01			1	
11	1	1	1	1
10	1	1	1	1

$F(W,X,Y,Z)=W+XYZ+X'Y'Z$

Don’t care

- You don't always need all 2^n input combinations in an n-variable function
  - If you can guarantee that certain input combinations never occur
  - If some outputs aren't used in the rest of the circuit

- We mark don't-care outputs in truth tables and K-maps with Xs.

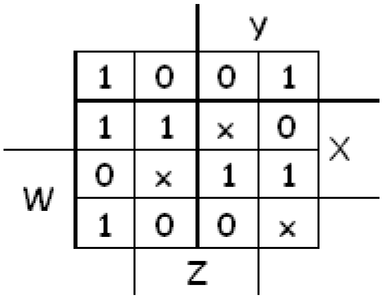
x	y	z	f(x,y,z)
0	0	0	0
0	0	1	1
0	1	0	X
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	X
1	1	1	1

- Within a K-map, each X can be considered as either 0 or 1. You should pick the interpretation that allows for the most simplification.

  - Find a MSP for

$$f(w,x,y,z) = \Sigma m(0,2,4,5,8,14,15), d(w,x,y,z) = \Sigma m(7,10,13)$$

This notation means that input combinations wxyz = 0111, 1010 and 1101 (corresponding to minterms m_7, m_{10} and m_{13}) are unused.



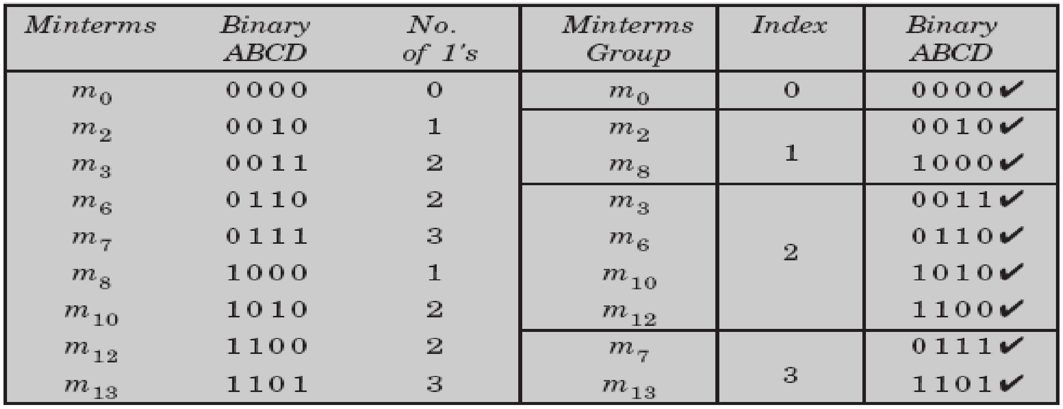
1	0	0	1
1	1	x	0
0	x	1	1
1	0	0	x

Minimization using Quine-Mccluskey (Tabular) Method:

- The K-map method is suitable for simplification of Boolean functions up to 5 or 6 variables.
- As the number of variables increases beyond this, the visualization of adjacent squares is difficult as the geometry is more involved.
- The „Quine-McCluskey‟ or „Tabular‟ method is employed in such case which is given below:
  1. Arrange all minterms in groups, such that all terms in the same group have same number of 1‟s in their binary representation. Start with the least number of 1‟s and continue with grouping of increasing number of 1‟s, the number of 1‟s in each term is called the index of that term *i.e.*, all the minterms of same index are placed in a same group. The lowest value of index is zero. Separate each group by a thick line. This constitutes the I stage.
  2. Compare every term of the lowest index (say *i*) group with each term in the successive group of index (say, *i* + 1). If two minterms differ in only one variable, that variable should be removed and a dash (–) is placed at the position, thus a new term with one less literal is formed. If such a situation occurs, a check mark (✔) is placed next to both minterms. After all pairs of terms with indices *i* and (*i* + 1) have been considered, a thick line is drawn under the last terms. When the above process has been repeated for all the groups of I stage, one stage of elimination have been completed. This constitutes the II stage.
  3. The III stage of elimination should be repeated of the newly formed groups of second stage. In this stage, two terms can be compared only when they have dashes in same positions. The process continues to next higher stages until no further comparisons are possible. (i.e., no further elimination of literals).
  4. All terms which remain unchecked (No ✔ sign) during the process are considered to be prime implicants (PIs). Thus, a set of all PIs of the function is obtained.
  5. From the set of all prime implicates, a set of essential prime implicants (EPIs) must be determined by preparing prime implicant chart as follow.
     1. The PIs should be represented in rows and each minterm of the function in a column.
     2. Crosses should be placed in each row corresponding to minterms that makesthe PIs.
     3. A complete PIs chart should be inspected for columns containing only a singlecross. PIs that cover minterms with a single cross in their column are called EPIs.
  6. The minterms which are not covered by the EPIs are taken into consideration and
     1. a minimum cover is obtained form the remaining PIs.

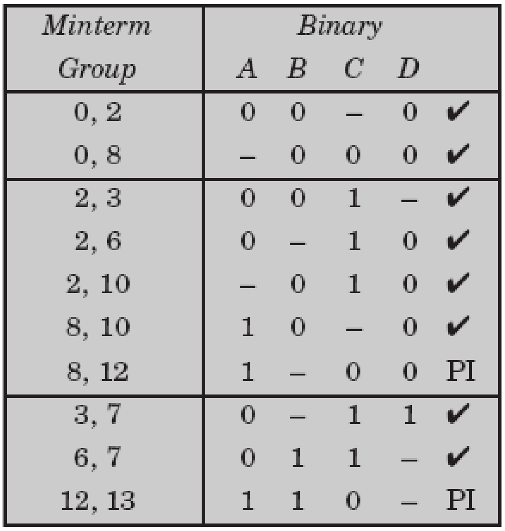
Ex: Simplify the function using tabular method. F (A, B, C, D) = Σ (0, 2, 3, 6, 7, 8, 10, 12, 13)

1. The minterms of the function are represented in binary form. The binary represented are grouped into a number of sections interms of the number of 1‟s index as shown in table

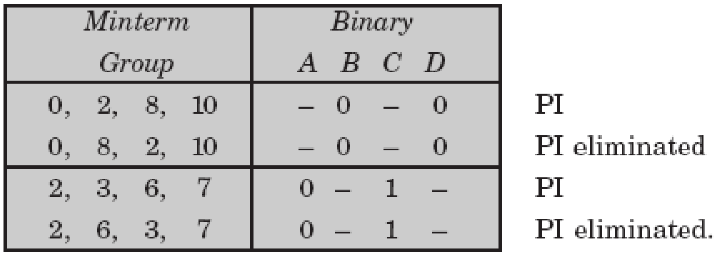


Minterms	*Binary ABCD*	*No. of 1's*	*Minterms Group*	*Index*	*Binary ABCD*
m_0	0000	0	m_0	0	0000✔
m_2	0010	1	m_2	1	0010✔
m_3	0011	2	m_8		1000✔
m_6	0110	2	m_3	2	0011✔
m_7	0111	3	m_6		0110✔
m_8	1000	1	m_{10}		1010✔
m_{10}	1010	2	m_{12}		1100✔
m_{12}	1100	2	m_7	3	0111✔
m_{13}	1101	3	m_{13}		1101✔

2. Compare each binary term with every term in the adjacent next higher category.If theydiffer only by one position put a check mark and copy the term into the next column with (–) in the place where the variable is unmatched, which is shown in Table



Minterm	*Binary*				
Group	*A*	*B*	*C*	*D*	
0, 2	0	0	–	0	✔
0, 8	–	0	0	0	✔
2, 3	0	0	1	–	✔
2, 6	0	–	1	0	✔
2, 10	–	0	1	0	✔
8, 10	1	0	–	0	✔
8, 12	1	–	0	0	PI
3, 7	0	–	1	1	✔
6, 7	0	1	1	–	✔
12, 13	1	1	0	–	PI

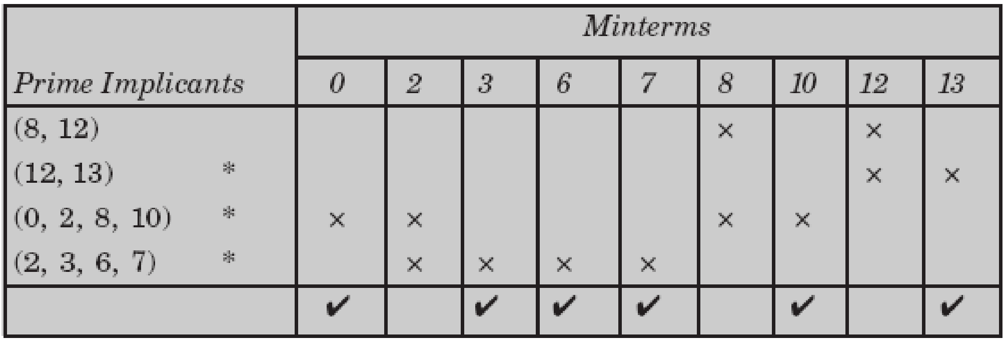


Minterm	*Binary*				
Group	*A*	*B*	*C*	*D*	
0, 2, 8, 10	–	0	–	0	PI
0, 8, 2, 10	–	0	–	0	PI eliminated
2, 3, 6, 7	0	–	1	–	PI
2, 6, 3, 7	0	–	1	–	PI eliminated.

3. Apply same process to the resultant column of Table and continue until no further elimination of literals. This is shown in Table above.

4. All terms which remain unchecked are the PIs. However note that the minterms combination (0, 2) and (8, 10) form the same combination (0, 2, 8, 10) as the combination (0, 8) and (2, 10). The order in which these combinations are placed does not prove any effect. Moreover, as we know that *x* + *x* = *x*, thus, we can eliminate one of these combinations. The same occur with combination (2, 3) and (6, 7).

5. Now we prepare a PI chart to determine EPIs as follows shown in Table



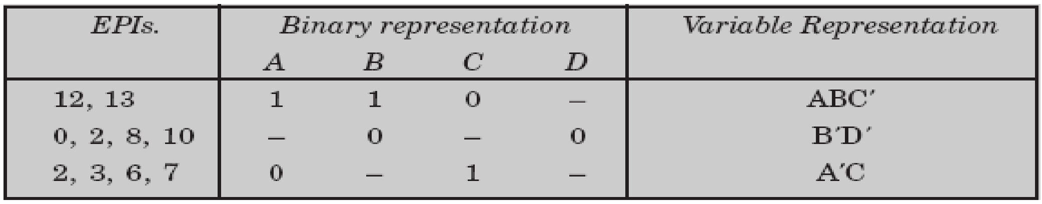
Prime Implicants		Minterms 0	2	3	6	7	8	10	12	13
(8, 12)							×		×	
(12, 13)	*								×	×
(0, 2, 8, 10)	*	×	×				×	×		
(2, 3, 6, 7)	*		×	×	×	×				
		✔		✔	✔	✔		✔		✔

a. All the PIs are represented in rows and each minterm of the function in acolumn.

b. Crosses are placed in each row to show the composition of minterms that make PIs.

c. The column that contains just a single cross, the PI corresponding to the row in which the cross(x) appears is Essential Prime Implicant. A tick mark is part against each column which has only one cross mark. A star (*) mark is placed against each EPI.

6. Finally, the sum of all the EPIs gives the function in its minimal SOP form



EPIs.	Binary representation A	B	C	D	Variable Representation
12, 13	1	1	0	–	ABC′
0, 2, 8, 10	–	0	–	0	B′D′
2, 3, 6, 7	0	–	1	–	A′C

Therefore The “Quinine Mccluskey‟method for including don‟t cares, is given below:

- The don‟t care minterms are also included in the first table.
- However, that don‟t care minterms will not be listed as column headings in the PE table, as they do not have to be covered by the minimal (simplified) expression.

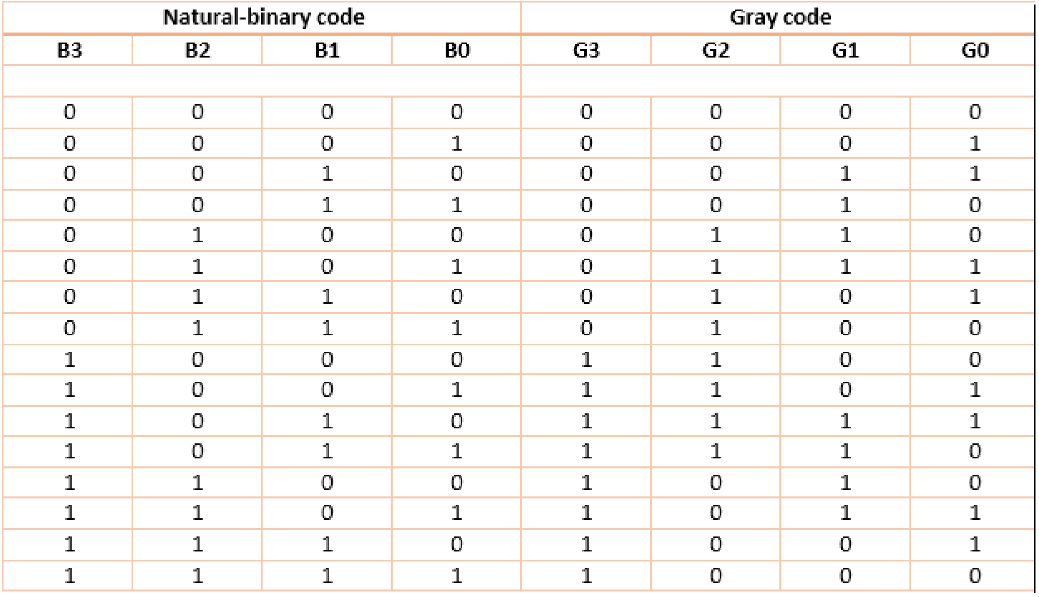
Code Converters

- Numbers are usually coded in one form or another so as to represent or use it as required. For instance, a number „nine‟ is coded in decimal using symbol $(9)_d$. Same is coded in natural-binary as$(1001)_b$.
- One of these other code is gray-code, in which any two numbers in sequence differ only by one bit change. This code is used in K-map reduction technique. The advantage is that when numbers are changing frequently, the logic gates are turning ON and OFF frequently and so are the transistors switching which characterizes power consumption of

the circuit; since only one bit is changing from number to number, switching is reduced and hence is the power consumption. Let’s discuss the conversion of various codes from one form to other.

Binary-to-Gray:

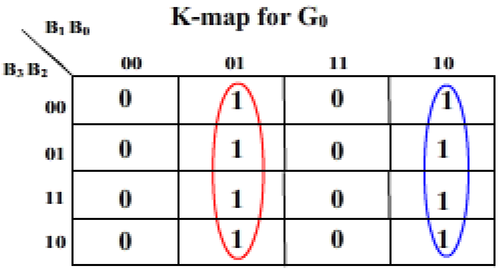
- The table that follows shows natural-binary numbers (upto 4-bit) and corresponding gray codes.



Natural-binary code				Gray code			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

- Looking at gray-code (G3G2G1G0), we find that any two subsequent numbers differ in only one bit-change.

- The same table is used as truth-table for designing a logic circuitry that converts a given 4-bit natural binary number into gray number. For this circuit, B3 B2 B1 B0 are inputs while G3 G2 G1 G0 are outputs.
  K-map for the outputs:



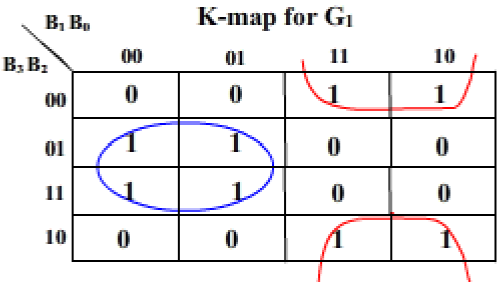
K-map for G_0

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$G_0 = B_1'B_0 + B_1B_0'$$

$$G_0 = B_0 \oplus B_1$$

, G3 = B3

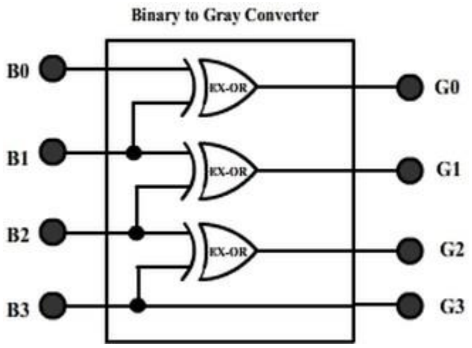


K-map for G_1

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$$G_1 = B_1'B_2 + B_1B_2'$$

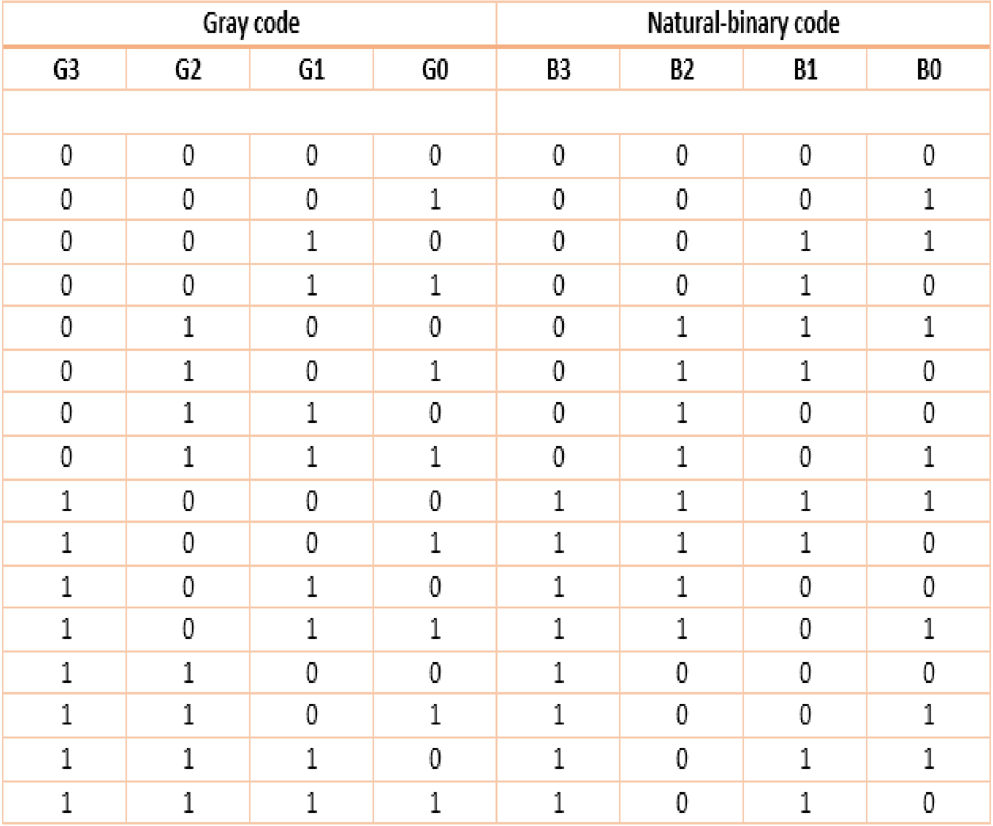
$$G_2 = B_1 \oplus B_2$$



- So that’s a simple three EX-OR gate circuit that converts a 4-bit input binary number into its equivalent 4-bit gray code. It can be extended to convert more than 4-bit binary numbers.

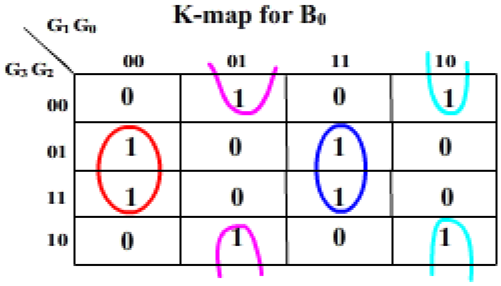
Gray-to-Binary:

Once the converted code (now in Gray form) is processed, we want the processed data back in binary representation. So we need a converter that would perform reverse operation to that of earlier converter. This we call a Gray-to-Binary converter. The design again starts from truth-table:



Gray code				Natural-binary code			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

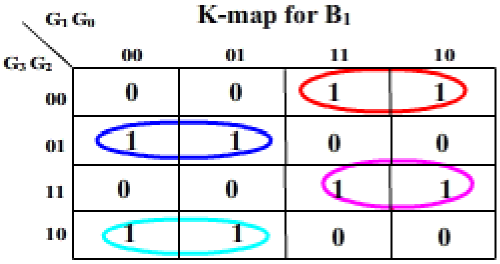
Then the K-maps:



K-map for B_0

G_3G_2 \ G_1G_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	1	0	1	0
10	0	1	0	1

$$B_0 = G_2G_1'G_0' + G_2'G_1G_0' + G_2'G_1'G_0 + G_2G_1G_0$$
$$= G_0'\,(G_1'G_2 + G_1G_2') + \;G_0\,(G_1G_2 + G_1'G_2')$$
$$= G_0'\,(G \oplus G_2) + \;G_0\,(G_1 \;\oplus G_2)' = G_0 \;\oplus G_1 \oplus G_2$$

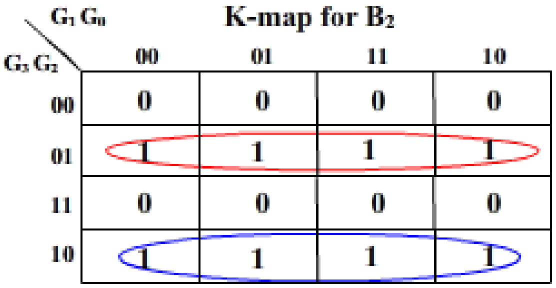


K-map for B_1

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

$$B_1 = G_3'G_2'G_1 + G_3'G_2G_1' + G_3G_2G_1 + G_3G_2'G_1'$$
$$= G_3'\,(G_2 \oplus G_1) + G_3\,(G_2 \oplus G_1)'$$
$$= \;G_1 \;\oplus G_2 \;\oplus G_3$$

B3=G3 and



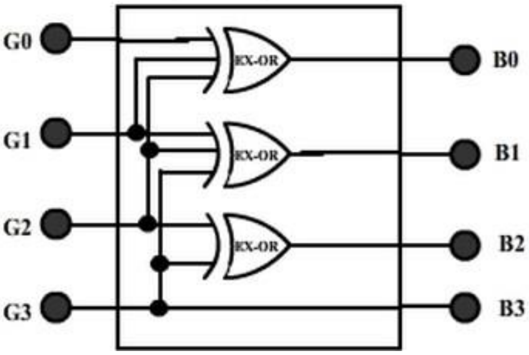
K-map for B_2

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$$B_2 = \;G_2 \;\oplus G_3$$

The realization of Gray-to-Binary converter is

Gray to Binary Converter



Assignment-Cum-Tutorial Questions

A. Questions testing the remembering / understanding level of students

I) Objective Questions

1) Every logical operation in a Boolean expression represents a ----------

2) A+AB+ABC+ABCD+ABCDE+ ----=

   (A) 1 (B) A (C) A+AB (D) AB

3) Boolean expression for the output of XNOR logic gate with inputs A and B is

   (A). AB“ + A“B (B) (AB)“ + AB (C).(A“ + B)(A + B“) (D) (A“ + B“)(A + B)

4) The number of cells in a 6- variable K-map is ----

   (A) 6 (B) 12 (C) 36 (D) 64

5) The bunch of 1“s on the K-map which from a 2-square, 4-square. Etc. is called a -

6) The number of adjacent cells each cell in an n variable K-map can have is

   (A) n-1 (B) n (C) n+1 (D) 2n

7) The NAND-NAND realization is equivalent to

   (A) AND –NOT realization (B) AND – OR realization

   (C) OR - AND realization (D) NOT – OR realization

8) If and only if all of the inputs are on, the output will be off. This is called ---------

   (A) NAND (B) NOR (C) X-OR (D) OR

9) Match the following

1)	DeMorgan“s law	A)	A (A + B) = A
2)	Distributive law	B)	A+1=1
3)	Identity law	C)	(A+B)“=A“.B“
4)	Redundancy Law	D)	A + (B C) = (A + B). (A + C)

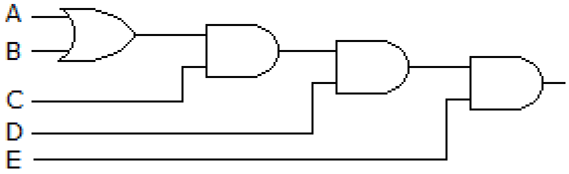
10) The Boolean expression A+BC in reduced form is

   (A) AB+BC (B) (A+B)(A+C) (C) (A+C)B (D) AB

11) Which of the following expressions is in the sum-of-products (SOP) form?

(A) (X + Y)(Z + W) (B) X+Y+ZW (C) XY+ZW (D) XY + Z+W

12) Derive the Boolean expression for the logic circuit shown below



A. C(A+B)DE B. [C(A+B)D+E“] C.[C(A+B)D]E“ D. ABCDE

II) Descriptive Questions

1) State and prove (a) Consensus theorem, (b) Transposition theorem and (c) Demorgan“s theorem.
2) Show that both NAND gate and NOR gate are universal gates.
3) Draw the logic circuits for the realization of basic operations using only NOR gates.
4) Simplify the following three variable Boolean expression using karnaugh map method.

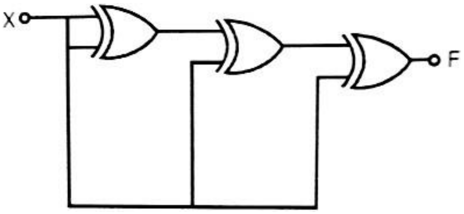
   Y= ABC“ +A“B“C“+ABC+AB“C“.
5) Using K-Map simplify the following Boolean function

   F=A“BC+ABC“+ABC+AB“C“
6) What is a code converter? Explain the principle of binary to gray and gray to binary.
7) Explain the process of minimization of Boolean function using tabular method.
8) Describe the procedure of reducingBoolean expression with don“t cares using tabular method

B. Question testing the ability of students in applying the concepts.

I) Objective Questions

1. For the circuit shown below, the output F is given by



(A) 1 (B) 0 (C) x (D) x‟

2. Minimum number of 2 input NAND gates required to implement the function given below is F=(X‟+Y‟)(Z+W)

(A)3 (B) 4 (C) 5 (D) 6

3. Which output expression might indicate a product-of-sums circuit construction?

(A) X=CH‟(D+E+F)

(B). X=CG(DE)

(C). X=(AC+BD+EF)‟

(D) X=(C+D)(E+G)

4. If f(A,B,C,D)=1 then the K-map contains__________number of logic 1‟s is

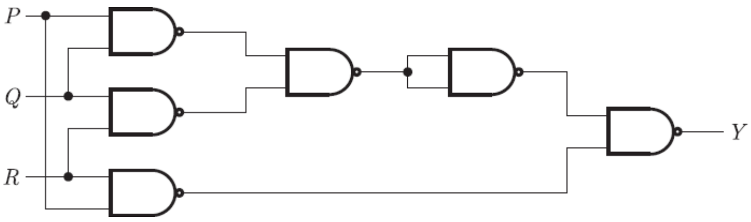
(A) 4 (B) 8(C) 16 (D) 32

5. The K – map for a Boolean function is shown in the figure. The number of essential prime implicants for this function is

ab \ cd	00	01	11	10
00	1	1	0	1
01	0	0	0	1
11	1	0	0	0
10	1	0	0	1

(A) 4 (B) 5 (C) 6 (D) 7

6.



The output Y in the circuit below is always „1‟ when

(A) two or more of the inputs *P*,*Q*,*R* are „0‟
(B) two or more of the inputs *P*,*Q*,*R* are „1‟
(C) any odd number of the inputs *P*,*Q*,*R* is „0‟
(D) any odd number of the inputs *P*,*Q*,*R* is „1‟

7. The Boolean function *Y*=*AB*+*CD* is to be realized using only 2 -input NAND gates. The minimum number of gates required is

(A) 2 (B) 3 (C) 4 (D) 5

8. A function F(A,B,C) contains minterms 1,2,3,5,6,7, its complement contains

(A) Σm (0,4) (B) Σm(1,2,3,5,6,7) (C) ΠM (1,2,3,5,6,7) (D)ΠM (0,4)

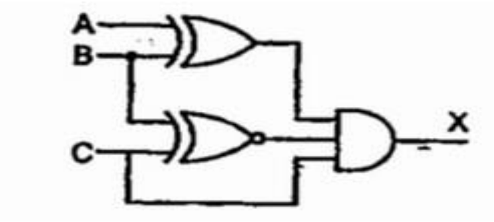
9. The Boolean expression for the truth table shown is

A	*B*	*C*	*D*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(A) B(A+C)(A‟+C‟) (B) B(A+C‟)(A‟+C)

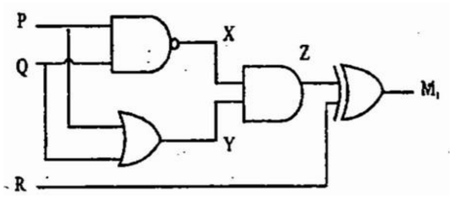
(C) B‟(A+C‟)(A‟+C) (D) B‟(A+C)(A‟+C‟)

10.For the logic circuit shown in the figure the required input(A,B,C) condition to make the output(X) = 1 is



(A) (1,0,1) (B) (0,0,1) (C) (1,1,1) (D)(0,1,1)

11. For the circuit shown below the expression for output is



(A) (PR OR Q)XOR R (B) (PR AND Q)XOR R

(C) (PR NOR Q)XOR R (D) (PR XOR Q) XOR R

II) Problems

1 Simplify the following Boolean expressions to a minimum number of literals

i) $x\ y + xy + x\ y$ ii) $x + y\ \ + (x + y)$ ii) $x\ y + xy + xy + x\ y$

2 Find the complement of the following Boolean expressions

i) $xy + x\ y$ ii) $AB + C\ \ D + E$

3 Using DeMorgan‟s theorem, convert the following Boolean expression to equivalent expression that have only OR and complement operation

F=$x\ y + x\ z + yz$

4 Given the following Boolean expression:

F= $x\quad + x\ yz + wxy + wx\ y + wxy$

a) Obtain the truth table of the function

b) Draw the logic diagram using the original Boolean expression.

c) Simplify the function to a minimum number of literals using Boolean algebra.

d) Obtain the truth table of the function from the simplified expression and show that it is the same as the one in part(a).

5 Express the following functions in sum of minterms and product of maxterms

i) F(A,B,C,D)=$BD + A\ D + BD$ ii)F(x,y,z)=(xy+z)(xz+y)

6 Find the complement of the function in sum of minterms F(A,B,C,D)=∑(0,2,6,11,13,14)

7 Draw the logic diagram of the following Boolean expressions without simplifying them:

i) $BC\ + AB + ACD$ ii) $A + B\ C + D\ (A + B + D)$

8 Show that the dual of the exclusive-OR is equal to its complement

9 Simplify the following Boolean expression and implement them in

a) Two level NAND gate circuit b)Two level NOR gate circuit

$$AB + ABD + ABD + A\ C\ D + A\ BC$$

10 Simplify the following Boolean expression K- maps

i) $XY + XYZ\ \ + XYZ$ ii) F(A,B,C,D)=∑(4,6,7,15)

1 Simplify the following Boolean expression by first finding the essential prime implicates F(W,X,Y,Z)=∑(0,2,4,5,6,7,8,10,13,15)

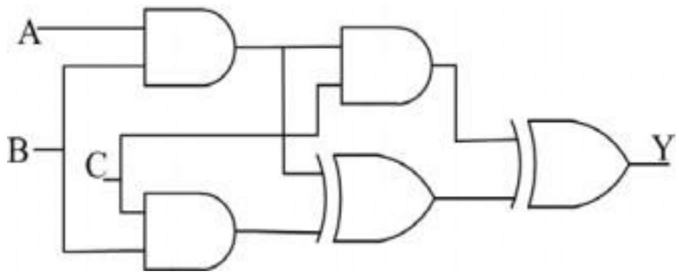
C. Questions testing the analyzing / evaluating ability of students

1. Obtain the minimal expression for Σm(2,3,5,7,9,11,12,13,14,15) and implement it in NOR logic
2. Obtain the minimal expression for ΠM(2,3,5,7,9,11,12,13,14,15) and implement it in NAND logic

3. With the use of maps, find the simplest sum-of-products form of the function F = fg where f = abc“ + c“d + a“cd“ + b“cz“ and g = (a + b + c“ + d“)(b“ + c“ + d)(a“ + c + d“)
4. Given AB“+ A“B=C, then prove that AC“+A“C=B.

D. GATE/IES Questions

1. The output of the combinational circuit given below is **GATE-16**



(A) A+B+C (B) A(B+C) (C) B(B+A) (D) C(A+B)

2. The Boolean expression F(X,Y,Z)=XYZ + XYZ + XYZ + XYZ converted into the canonical product of sum(POS) form is **GATE-15**

(a) $x+y+z\,x+y+z \quad x+y+z\ (x+y+z\,)$

(b) $x+y+z \;\; x \;\; +y+z \;\; x+y+z\,(x+y+z\,)$

(c) $x+y+z \;\; x \;\; +y+z \;\; x+y+z\,(x+y+z\,)$

(d) $x+y+z\,x+y+z\,x+y+z\ (x+y+z)$

3. For the given boolean function which one of the following is the complete set of essential Prime implicants F(W,X,Y,Z)=wy + xy + wxyz + wxy + xz + xyz
(A) w, y, xz, xz (B)W, Y, XZ (C) y, xyz (D)y, xz, xz **GATE-14**

4. In the sum of products function F(x,y,z)=∑m(2,3,4,5), the prime implicants are

(A) xy,xy (B)xy,xyz (C)xyz,xyz, xy (D)xyz,xyz,xyz,xyz **GATE- 12**

5. IF x=1,In the given logic equation [x+z{y+(z+xy)}] {x+z(x+y)}=1,then **GATE-09**

(A)y=z (B)y=z (C)z=1 (D)z=0

6. The following boolean expression Y=ABCD + ABCD + ABCD + ABCD can be minimized to

(A)ABCD + ABC + ACD (B)Y=ABCD + BCD + ABCD **GATE-07**

(C) ABCD + BCD + ABCD (D)ABCD + BCD + ABCD

7. The number of product terms in the minimized sum of product expression obtained through the following karnaugh map (where d indicates don‟t care conditions). **GATE-06**

1	0	0	1
0	D	0	0
0	0	D	1
0	0	0	1

(A)2 (B)3 (C)4 (D)5

8.The boolean expression for the truth table shown is **GATE-05**

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

(A)$B\ \ A + C\ \ (A + C)$ (B)$B\ \ A + C\ \ (A + C)$

(C)$B\ \ A + C\ \ (A + C)$ (D)$B\ \ A + C\ \ (A + C)$

9. The Boolean expression $AC + B\ C$ is equivalent to **GATE-04**

(A)$AC + BC + AC$ (B)$BC + AC + BC + ACB$

(C) $AC + B\ C + BC + ABC$ (D)$ABC + ABC + ABC + ABC$

IES Questions

1). The logic Function $ABD + ABD$ can be reduced to **IES-2015**

A) A‟B‟ B) AB‟ C) B‟D‟ D) AD‟

2)The minimum number of NAND gates required to implement A +AB+ABC is: **IES-2014**

(A) 0 (B) 1 (C) 4 (D) 7

3) The logic function f(A, B, C, D) = (A+BC) (B+ CD) can be expressed to **IES-2013**

A) AB+BC+AC‟D+BCD B) AB+AB‟+ACD+BCD‟

C) AB+A“B“+A“CD+BC“D D) AB“+AB“+ACD“+BCD

4) The logic function (A+ B) can be expressed in terms of min terms as: **IES-2012**

A) AB+BA B) A“B+B“A+A“B“ C) A“B+AB D) AB+BA“

5) A3- Variable truth table has a High output for the inputs: 010, 011 and 110. **IES-2011**

The Boolean expression for sum of product (SOP) can be written as:

A) AB+BC (B) AB“+BC“ C) A“B“+BC D) AB+B“C“