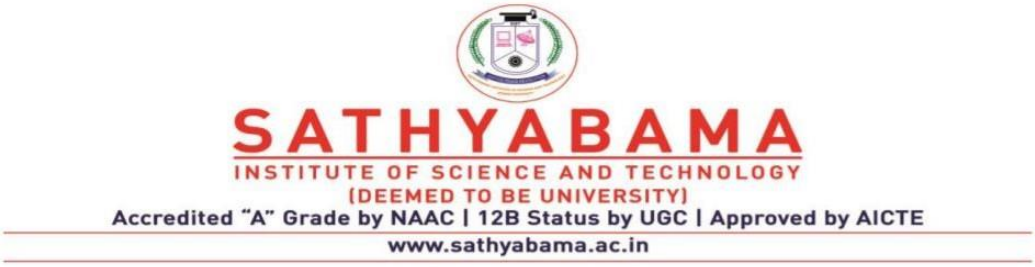


SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT – I – SBS1203 – COMPUTER ARCHITECTURE

1.1 NUMBER SYSTEM

Number systems are the technique to represent numbers in the computer system architecture. Whatever value we are store and getting from computer memory has defined in the number system. Computers can understand the following types of number.

1. Decimal Number - base 10
2. Binary Number – Base 2
3. Octal Number - Base 8
4. Hexadecimal Number – base 16
5. Decimal Number System

Decimal number are from 0 to 9, binary number are 0's and 1's. Octal number system starts from 0 to 7. Hexadecimal Number System starts from 0 to 15. In this hexadecimal system from 10 to 15 represented as A to F.

Decimal Number System

Decimal number system is base 10 number system. The digits from 0 to 9.

Decimal to Binary Conversion

Steps to follow for conversion:

- Divide each digit by 2, keep track of the reminder
- Find the reminder reaches 0 LSB(Least Significant Bit)
- E.g. $10_{10} = 1010_2$

Decimal to Octal Conversion

Steps to follow for conversion:

- Divide each digit by 8, keep track of the reminder
- Find the reminder reaches 0 LSB(Least Significant Bit)
- E.g. $10_{10} = 12_8$

Decimal to Hexadecimal Conversion

Steps to follow for conversion:

- Divide each digit by 16, keep track of the reminder
- Find the reminder reaches 0 LSB(Least Significant Bit)
- E.g. $20_{10} = A4_{16}$

Binary Number System

Digital Computers represent all kinds of data and information in form the binary number system.

Binary Number System consists of two digits 0 and 1. Base is 2.

Binary to Decimal Conversion

Steps to follow for conversion:

- Multiply each digit by 2^n, where n is "weight" of the bit.
- N is the position of the bit. Starting from 0.
- Add the result.
- E.g $1010_2 = 10_{10}$

Binary to Octal Conversion

Steps to follow for conversion:

- Group binary digits in a 3 bits from on right side (Most Significant Bit)
- Convert to octal digits.
- E.g $101011100_2 = 534_8$

Binary to Hexadecimal Conversion

Steps to follow for conversion:

- Group binary digits in a 4 bits from on right side (Most Significant Bit)
- Convert to hexadecimal digits.
- E.g $101011100_2 = 15C_{16}$

Octal Number System

Octal Number system is the base 8 number system. Starting from 0 to 7. The number after 7 is 10.

Octal to Decimal Conversion

Steps to follow for conversion:

-Multiply each digit by 8^n, where n is "weight" of the bit.
- N is the position of the bit. Starting from 0.
- Add the result.
- E.g $534_8 = 348_{10}$

Octal to Binary Conversion

Steps to follow for conversion:

- Group binary digits in a 3 bits from on right side (Most Significant Bit)
- Convert to octal digits.
- E.g $534_8 = 101011100_2$

Octal to Hexadecimal Conversion

Steps to follow for conversion:

- Convert the octal to binary.
- Group binary digits in a 4 bits from on right side (Most Significant Bit)
- Convert to octal digits.
- E.g $534_8 = 15C_{16}$

Hexadecimal Number System

Octal Number system is the base 16 number system. Starting from 0 to 9. The number is 10 - 15 represents as A - F.

Hexadecimal to Decimal Conversion

Steps to follow for conversion:

-Multiply each digit by 16^n , where n is "weight" of the bit.

- N is the position of the bit. Starting from 0.
- Add the result.
- E.g $15C_{16} = 348_{10}$

Hexadecimal to Binary Conversion

Steps to follow for conversion:

- Group binary digits in a 4 bits from on right side (Most significant Bit)
- Convert to octal digits.
- E.g $15C_{16} = 101011100_2$

Hexadecimal to Octal Conversion

Steps to follow for conversion:

- Group binary digits in a 4 bits from on right side (Most Significant Bit)
- Convert to octal digits.
- E.g $15C_{16} = 534_8$

2. COMPLEMENTS

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation. There are two types of complements, i.e.

a) r's-complement
b) (r – 1)' s-complement

1's-Complement

To find 1's-complement of a number replace all 0's with 1's and all 1's with 0's. The 1's complement of a number is always 1 less than the 2's-complement of a number.

E.g. 1. 1's-complement of 1011010.

- Replacing all 1's with 0's and all 0's with 1's.
- The 1's-complement of 1011010 is 0100101.

E.g. 2. 1's-complement of 0.0101

- Replace all 1's with 0's and all 0's with 1's.
- The 1's-complement of 0.0101 is 0.1010.

E.g.3. Find the subtraction $(1110101 - 1001101)_2$ using the 2's-complement method.

Minuend = 1110101
Subtrahend = 1001101
Minuend = 1110101
1's Complement of Subtrahend = 0110010
2's-complement of subtrahend = 0110010 +1= 0110011
1110101 + 0110011 = 1 0101000

Here, an end carry occurs, hence discard it. The result of $(1110101 - 1001101)_2$ is $(0101000)_2$.

3. SIGNED MAGNITUDE OF BINARY NUMBERS

A signed binary number consists of a sign, either positive or negative and magnitude. In a signed magnitude representation of binary numbers, the most significant digit is zero for the representation of positive binary number and one for the representation of negative binary numbers. This most significant digits (0 or 1) represents whether the number is positive or negative and the magnitude is the value of the numbers**.**

3.1. DECIMAL SIGNED NUMBERS

Decimal values of the positive and negative signed magnitude numbers can be determined by the summation of the weights of all the magnitude bits, where there are 1's and ignoring all other bits, where there are zeros (0).

E.g. Express the decimal equivalent of signed binary number 10011100 expressed in its sign magnitude form.

Solution: There are seven magnitude bits and one sign bit. Separating sign bits and magnitude bits sign bit = 1, which means that the magnitude of the number is negative.

Magnitude bits = 0011100, assigning the weights to the bits, we get

2^6 2^5 2^4 2^3 2^2 2^1 2^0

0 0 1 1 1 0 0

Summing the weights together where 1 exists and ignoring where 0 exists, we get,

$2^4 + 2^3 + 2^2 = 16 + 8 + 4 = 28.$

Adding sign magnitude bit to the solution for the signed magnitude binary number (1 0011100) is (–28).

3.2 . BINARY CODE

The digits 1 and 0 used in binary reflect the on and off states of a transistor. Each instruction is translated into machine code - simple binary codes that activate the CPU.

Programmers write computer code and this is converted by a translator into binary instructions that the processor can execute.

Different Types of Binary Code:

1. Binary Coded Decimal or 8421 Code
2. 2421 Code
3. 5211 Code
4. Gray Code (Reflected Code)
5. Error - Detection Code

Convert to BCD to Excess-3

0000 + 0011 = 0011
01 0011 = 0100

Convert Binary to Gray Code

BC GC
0011 0010
Most Significant Bit
Keep MSB , like 0 means 0
Both are same, like 0 and 0 means 0
like 1 and 1 means 0
Difference, like 0 and 1 means 1

Binary codes for the decimal digits

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Fig.1.1 Binary codes for the decimal numbers

4. Error Correction Code

Error Correction codes detect the error, if it is occurred during transmission of the original data bit stream.

E,g. Parity code, Hamming code.

Error correction codes − are used to correct the errors present in the received data bit stream so that, we will get the original data.

4.1.Parity Code

One of the common way to detect the error is Parity bit. A parity bit is extra bit included with a message to make the total number of one's transmitted either odd or Even.

Two Types in the Parity Code:

1. Odd Parity
2. Even Parity

If an odd parity selected, the total number of 1's in the message bit and parity bit is odd , the P bit.

Parity bit

Odd parity		Even parity	
Message	P	Message	P
0000	1	0000	0
0001	0	0001	1
0010	0	0010	1

Fig.1.2 . Parity bit

5. BINARY LOGIC

Binary Logic Deals with binary that on two discrete values and with operations that assume logical meaning. The two variables take may be called by different names.

e.g. True or false , yes or no

There are basic three logical operations : AND, OR and NOT

AND = X.Y

OR = X+ Y

NOT = X′

Truth Tables of Logical Operations

AND			OR			NOT	
x	y	x y	x	y	x + y	x	x′
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

Fig 1.3. Truth Tables of Logical operations

5.1 LOGIC GATES- TRUTH TABLE

AND GATE

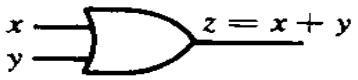
(a) Two-input AND gate

Fig. 1.4. Logical Diagram of AND gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 1.5. Truth Table of AND gate

OR GATE



(b) Two-input OR gate

Fig.1.6. Logical Diagram of the OR gate

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Fig.1.7. Truth Table of the OR gate

NOT GATE

NOT gate or inverter

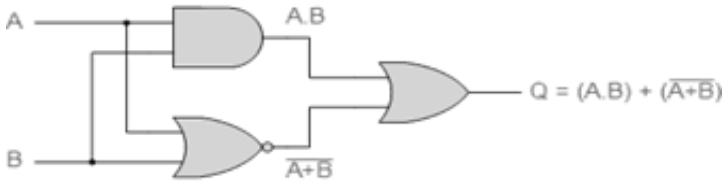
Fig.1.8 Logical Diagram of the NOT gate

NOT gate	
A	$\overline{A}$
0	1
1	0

Fig.1.9. Truth Table of the NOT gate

5.2.BOOLEAN ALGEBRA

The system consists of an AND Gate, a NOR Gate and finally an OR Gate. The expression for the AND gate is A.B, and the expression for the NOR gate is A+B. Both these expressions are also separate inputs to the OR gate which is defined as A+B. Thus the final output expression is given as:



The output of the system is given as Q = (A.B) + (A+B), but the notation A+B is the same as the De Morgan´s notation A.B, Then substituting A.B into the output expression gives us a final output notation of Q = (A.B)+(A.B), which is the Boolean notation for an Exclusive-NOR Gate as seen in the previous section.

Inputs		Intermediates		Output
B	A	A.B	A + B	Q
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

6. Simplification of Boolean function with Map method

2.6.3 Converting Expressions in Standard SOP or POS Forms

Sum of products form can be converted to standard sum of products by ANDing the terms in the expression with terms formed by ORing the literal and its complement which are not present in that term. For example for a three literal expression with literals A, B and C, if there is a term AB, where C is missing, then we form term $(C + \bar{C})$ and AND it with AB. Therefore, we get $AB(C + \bar{C}) = ABC + AB\bar{C}$

Steps to convert SOP to standard SOP form

Step 1 : Find the missing literal in each product term if any.

Step 2 : AND each product term having missing literal/s with term/s form by ORing the literal and its complement.

Step 3 : Expand the terms by applying distributive law and reorder the literals in the product terms.

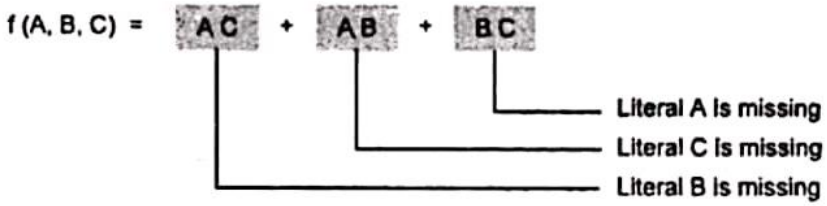
Step 4 : Reduce the expression by omitting repeated product terms if any. Because $A + A = A$.

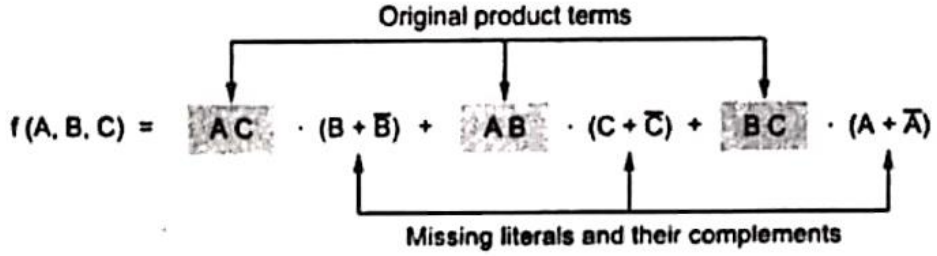
➠ **Example 2.3 :** *Convert the given expression in standard SOP form.*

$$f(A, B, C) = AC + AB + BC$$

Solution :

Step 1 : Find the missing literal/s in each product term





2.6.4 M-Notations : Minterms and Maxterms

Each individual term in standard SOP form is called **minterm** and each individual term in standard POS form is called **maxterm**. The concept of minterms and maxterms allows us to introduce a very convenient shorthand notations to express logical functions. Table 2.10 gives the minterms and maxterms for a three literal/variable logical function where the number of minterms as well as maxterms is $2^3 = 8$. In general, for an n-variable logical function there are 2^n minterms and an equal number of maxterms.

Variables			Minterms	Maxterms
A	B	C	m_i	M_i
0	0	0	$\bar{A}\,\bar{B}\,\bar{C} = m_0$	$A + B + C = M_0$
0	0	1	$\bar{A}\,\bar{B}\,C = m_1$	$A + B + \bar{C} = M_1$
0	1	0	$\bar{A}\,B\,\bar{C} = m_2$	$A + \bar{B} + C = M_2$
0	1	1	$\bar{A}\,B\,C = m_3$	$A + \bar{B} + \bar{C} = M_3$
1	0	0	$A\,\bar{B}\,\bar{C} = m_4$	$\bar{A} + B + C = M_4$
1	0	1	$A\,\bar{B}\,C = m_5$	$\bar{A} + B + \bar{C} = M_5$
1	1	0	$A\,B\,\bar{C} = m_6$	$\bar{A} + \bar{B} + C = M_6$
1	1	1	$A\,B\,C = m_7$	$\bar{A} + \bar{B} + \bar{C} = M_7$

Table 2.10 Minterms and maxterms for three variables

2.8 Karnaugh Map Minimization

We have seen that for simplification of Boolean expressions by Boolean algebra we need better understanding of Boolean laws, rules and theorems. During the process of simplification we have to predict each successive step. For these reasons, we can never be absolutely certain that an expression simplified by Boolean algebra alone is the simplest possible expression. On the other hand, the map method gives us a systematic approach for simplifying a Boolean expression. The map method, first proposed by Veitch and modified by Karnaugh, hence it is known as the **Veitch diagram** or the **Karnaugh map**.

2.8.1 One-Variable, Two-Variable, Three-Variable and Four-Variable Maps

The basis of this method is a graphical chart known as Karnaugh map (K-map). It contains boxes called cells. Each of the cell represents one of the 2^n possible products that can be formed from n variables. Thus, a 2-variable map contains $2^2 = 4$ cells, a 3-variable map contains $2^3 = 8$ cells, and so forth. Fig. 2.3 shows outlines of 1, 2, 3 and 4 variable maps.

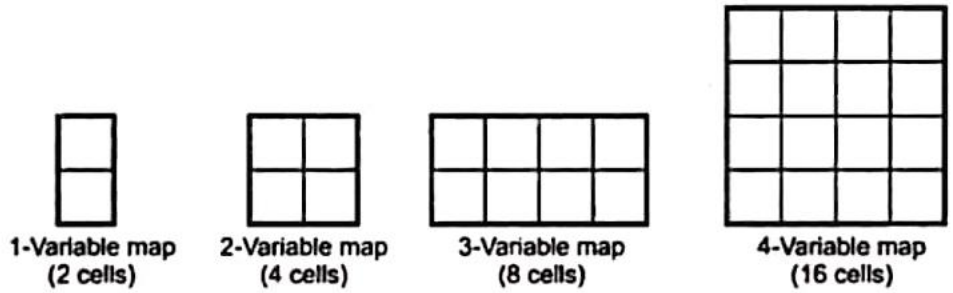


Fig. 2.3 Outlines of 1, 2, 3 and 4 variable Karnaugh maps

2.8.2 Plotting a Karnaugh Map

We know that logic function can be represented in various forms such as truth table, SOP Boolean expression and POS boolean expression. In this section we will see the procedures to plot the given logic function in any form on the Karnaugh map.

2.8.2.1 Representation of Truth Table on Karnaugh Map

Fig. 2.6 shows K-maps plotted from truth tables with 2, 3 and 4-variables. Looking a the Fig. 2.6 we can easily notice that the terms which are having output 1, have th corresponding cells marked with 1s. The other cells are marked with zeros.

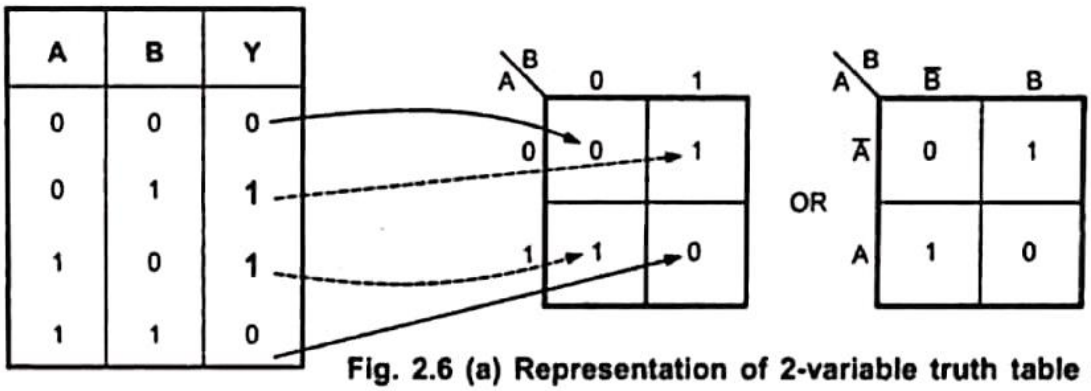


Fig. 2.6 (a) Representation of 2-variable truth table on K-map

Note : The student can varify the data in each cell by checking the data in the column Y for particular row number and the data in the same cell number in the K-map.

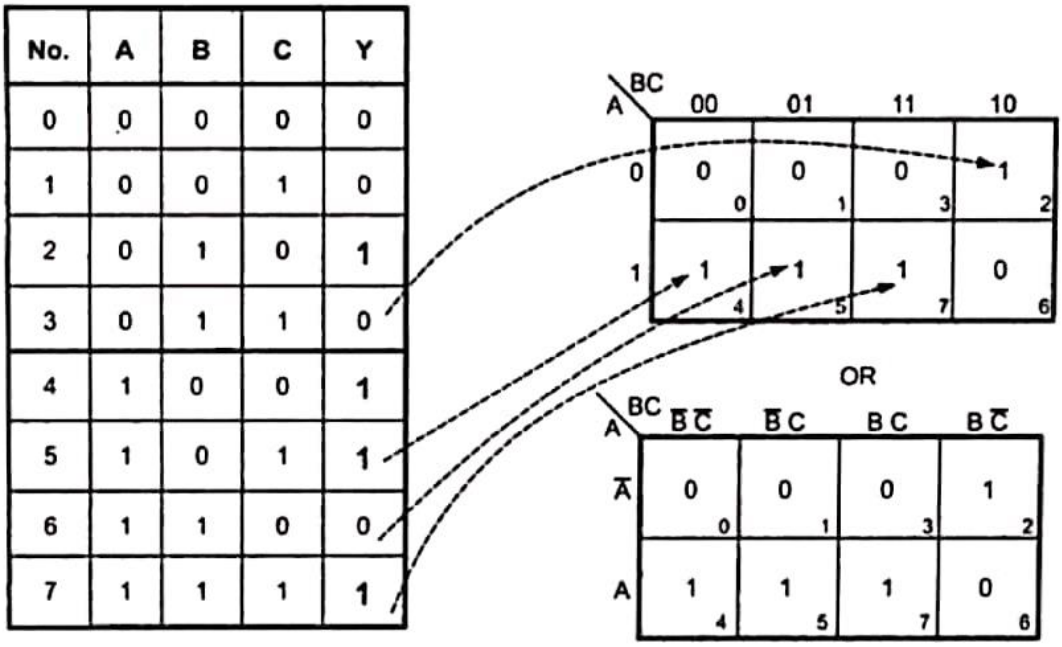


Fig. 2.6 (b) Representation of 3-variable truth table on K-map

No.	A	B	C	D	Y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	1

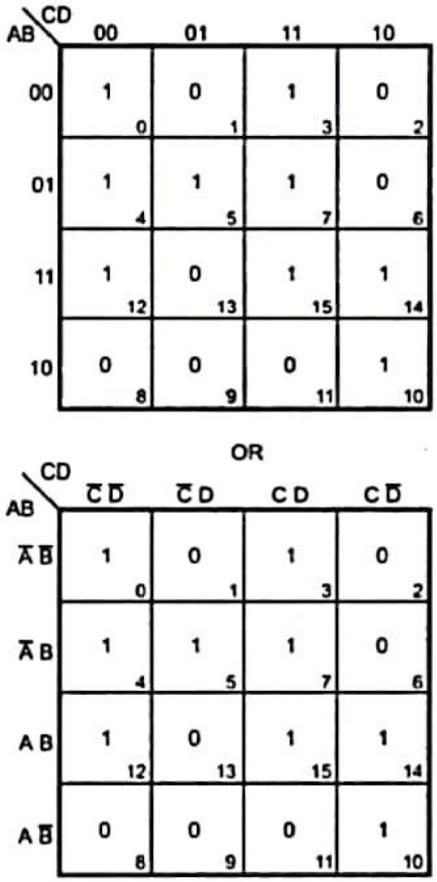


Fig. 2.6(c) Representation of 4-variable truth table on K-map
Fig. 2.6 Plotting truth table on K-map

2.8.2.2 Representing Standard SOP on K-map

A Boolean expression in the sum of products form can be plotted on the Karnaugh map by placing a 1 in each cell corresponding to a term (minterm) in the sum of products expression. Remaining cells are filled with zeros. This is illustrated in the following examples.

Example 2.10 : *Plot Boolean expression* $Y = A\bar{B}\bar{C} + ABC + \bar{A}\bar{B}C$ *on the Karnaugh map.*

Solution : The expression has 3 variables and hence it can be plotted using 3-variable map as shown below.

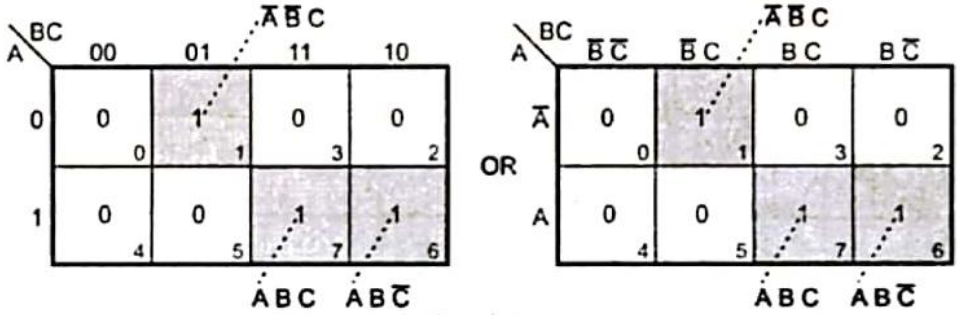


Fig. 2.7

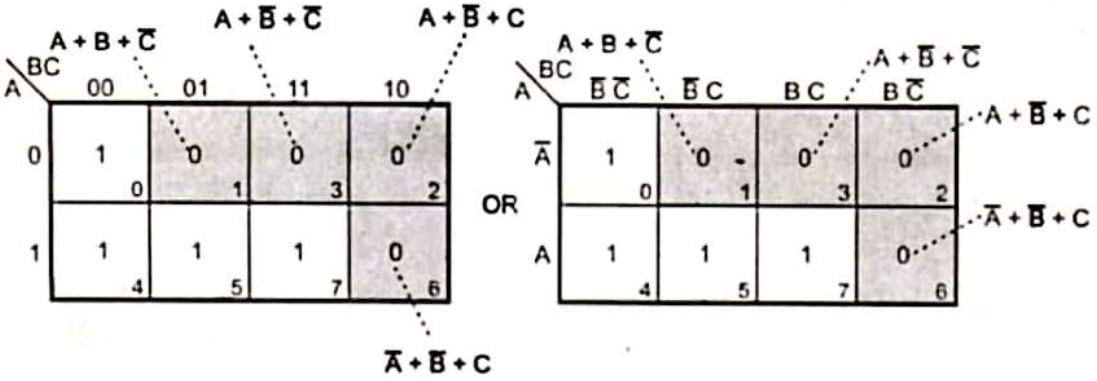
2.8.2.3 Representing Standard POS on K-map

A Boolean expression in the product of sums can be plotted on the Karnaugh map by placing a 0 in each cell corresponding to a term (maxterm) in the expression. Remaining cells are filled with ones. This is illustrated in the following examples.

Example 2.12 : *Plot Boolean expression* $Y = (A + \bar{B} + C)(A + \bar{B} + \bar{C})(\bar{A} + \bar{B} + C)(A + B + \bar{C})$ *on the Karnaugh map.*

Solution : The expression has 3 variables and hence it can be plotted using 3-variable map as shown below.

$(A + \bar{B} + C) = M_2, (A + \bar{B} + \bar{C}) = M_3, (\bar{A} + \bar{B} + C) = M_6, (A + B + \bar{C}) = M_1$



2.8.3 Grouping Cells for Simplification

In the last section we have seen representation of Boolean function on the Karnaugh map. We have also seen that minterms are marked by 1s and maxterms are marked by 0s. Once the Boolean function is plotted on the Karnaugh map we have to use grouping technique to simplify the Boolean function. The grouping is nothing but combining terms in adjacent cells. Two cells are said to be adjacent if they conform the single change rule. The simplification is achieved by grouping adjacent 1s or 0s in groups of 2^i, where $i = 1, 2, .., n$ and n is the number of variables. When adjacent 1s are grouped then we get result in the sum of products form; otherwise we get result in the product of sums form. Let us see the various grouping rules.

2.8.3.1 Grouping Two Adjacent Ones (Pair)

Fig. 2.11 (a) shows the Karnaugh map for a particular three variable truth table. This K-map contains a pair of 1s that are horizontally adjacent to each other; the first represents $\overline{A}\,\overline{B}C$ and the second represents $\overline{A}BC$. Note that in these two terms only the B variable

appears in both normal and complemented form ($\overline{A}$ and C remain unchanged). Thus these two terms can be combined to give a resultant that eliminates the B variable since it appears in both uncomplemented and complemented form. This is easily proved as follows :

$$Y = \overline{A}\,\overline{B}C + \overline{A}\,BC$$

$$= \overline{A}\,C(\overline{B} + B) \qquad \text{Rule 6 : } [A + \overline{A} = 1]$$

$$= \overline{A}\,C$$

This same principle holds true for any pair of vertically or horizontally adjacent 1s. Fig. 2.11 (b) shows an example of two vertically adjacent 1s. These two can be combined to eliminate A variable since it appears in both its uncomplemented and complemented forms. This gives result

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

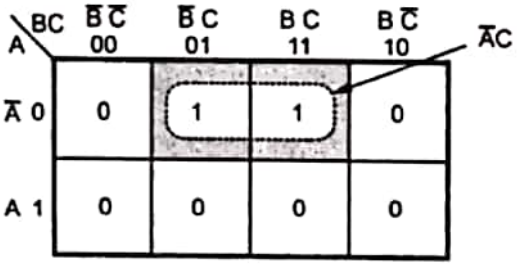


Fig. 2.11(a)

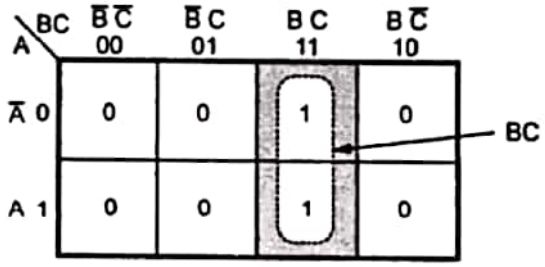


Fig. 2.11(b)

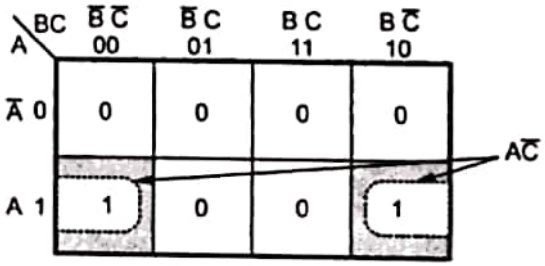


Fig. 2.11 (c)

In a Karnaugh map the leftmost column and rightmost column are considered to be adjacent. Thus, the two 1s in these columns with a common row can be combined to eliminate one variable. This is illustrated in Fig. 2.11 (c).

Here variable B has appeared in both its complemented and uncomplemented forms and hence eliminated as follows :

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

$$= A\,\overline{C}(\overline{B} + B) \qquad \text{Rule 6 : } [\overline{A} + A = 1]$$

$$= A\,\overline{C}$$

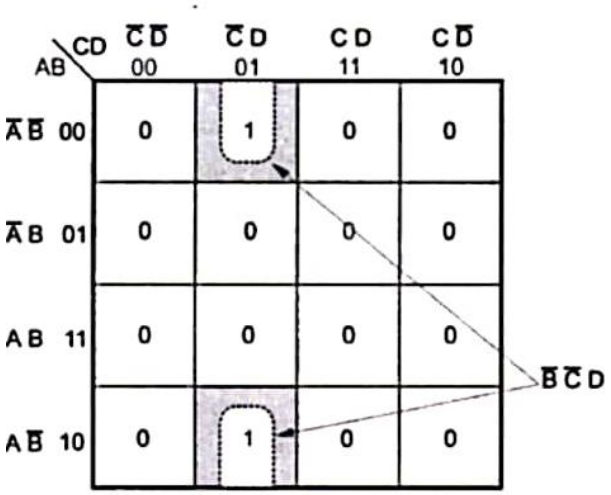


Fig. 2.11 (d)

Let us see another example shown in Fig. 2.11 (d). Here two 1s from top row and bottom row of some column are combined to eliminate variable A, since in a K-map the top row and bottom row are considered to be adjacent.

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

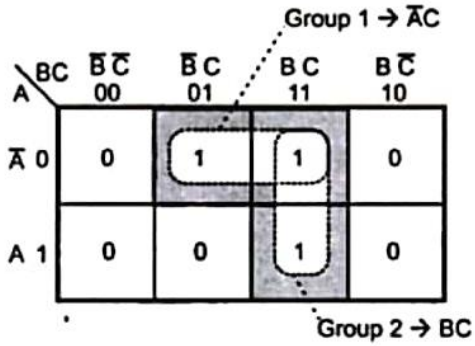


Fig. 2.11 (e)

Fig. 2.11 (e) shows a Karnaugh map that has two overlapping pairs of 1s. This shows that we can share one term between two pairs.

$$Y = \overline{A}\,\overline{B}C + \overline{A}\,BC + A\,BC$$
$$= \overline{A}\,\overline{B}C + \overline{A}\,BC + \overline{A}\,BC + A\,BC \qquad \text{Rule 5 : } [A + A = A]$$
$$= \overline{A}\,C\,(\overline{B} + B) + B\,C\,(\overline{A} + A)$$
$$= \overline{A}\,C + B\,C$$

Fig. 2.11 (f) shows a K-map where three group of pairs can be formed. But only two pairs are enough to include all 1s present in the K-map. In such cases third pair is not required.

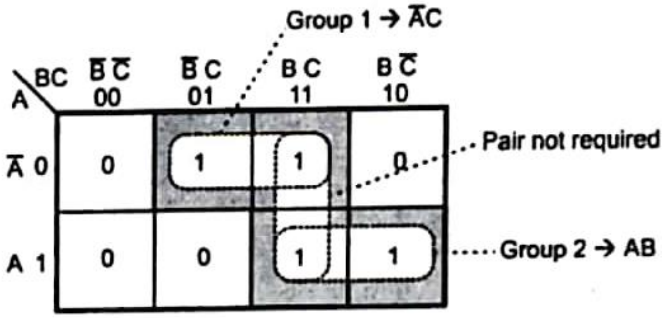


Fig. 2.11 (f) Examples of combining pairs of adjacent ones

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

$$= \overline{A}\,C\,(\overline{B} + B) + A\,B\,(C + \overline{C}) \qquad \text{Rule 6 : } [\overline{A} + A = 1]$$

$$= \overline{A}\,C + A\,B$$

2.8.3.2 Grouping Four Adjacent Ones (Quad)

In a Karnaugh map we can group four adjacent 1s. The resultant group is called Quad. Fig. 2.12 shows several examples of quads. Fig. 2.12 (a) shows the four 1s are horizontally adjacent and Fig. 2.12 (b) shows they are vertically adjacent.

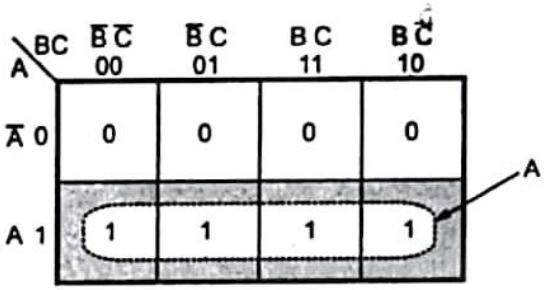


Fig. 2.12 (a) Y = A

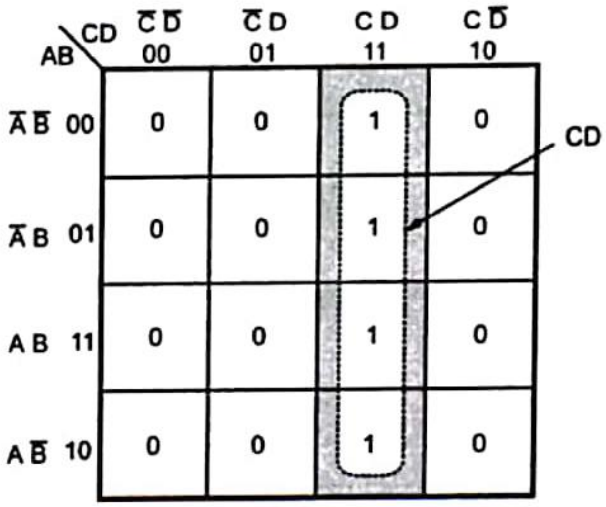


Fig. 2.12 (b) Y = CD

A K-map in Fig. 2.12 (c) contains four 1s in a square, and they are considered adjacent to each other. The four 1s in Fig. 2.12 (d) are also adjacent, as are those in Fig. 2.12 (e) because, as mentioned earlier, the top and bottom rows are considered to be adjacent to each other and the leftmost and rightmost columns are also adjacent to each other.

Simplification of Sum of Products Expressions (Minimal Sums)

From the above discussion we can outline generalized procedure to simplify Boolean expressions as follows :

1. Plot the K-map and place 1s in those cells corresponding to the 1s in the truth table or sum of product expression. Place 0s in other cells.
2. Check the K-map for adjacent 1s and encircle those 1s which are not adjacent to any other 1s. These are called isolated 1s .
3. Check for those 1s which are adjacent to only one other 1 and encircle such pairs.
4. Check for quads and octets of adjacent 1s even if it contains some 1s that have already been encircled. While doing this make sure that there are minimum number of groups.
5. Combine any pairs necessary to include any 1s that have not yet been grouped.
6. Form the simplified expression by summing product terms of all the groups.

To get familiar with these steps we will solve some examples.

Example 2.14 : *Minimize the expression* $Y = A\overline{B}C + \overline{A}\,\overline{B}C + \overline{A}BC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{C}$

Solution :

Step 1 : Fig 2.14 (a) shows the K-map for three variables and it is plotted according to the given expression.

Step 2 : There are no isolated 1s.

Step 3 : 1 in the cell 3 is adjacent only to 1 in the cell 1. This pair is combined and referred to as group 1.

Step 4 : There is no octet, but there is a quad. Cells 0, 1, 4 and 5 form a quad. This quad is combined and referred to as group 2.

Step 5 : All 1s have already been grouped.

Step 6 : Each group generates a term in the expression for Y. In group 1 B variable is eliminated and in group 2 variables A and C are eliminated and we get,

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

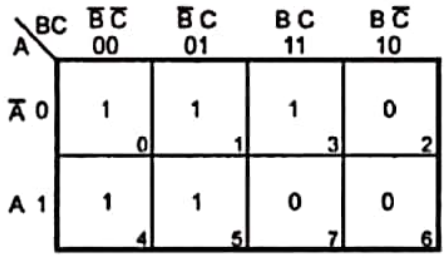


Fig. 2.14 (a)

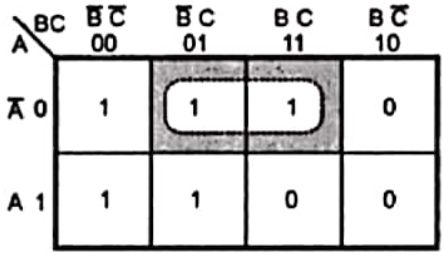


Fig. 2.14 (b)

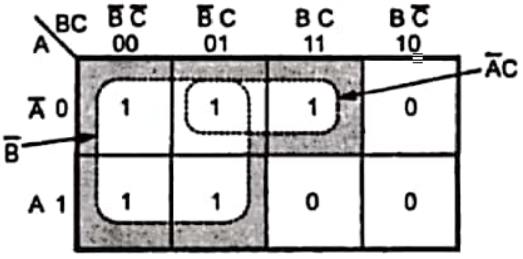


Fig. 2.14 (c)

2.8.6 Simplification of Product of Sums Expressions (Minimal Products)

In the above discussion, we have considered the Boolean expression in sum of products form and grouped 2, 4, and 8 adjacent ones to get the simplified Boolean expression in the same form. In practice, the designer should examine both the sum of products and product of sums reductions to ascertain which is more simplified. We have already seen the representation of product of sums on the Karnaugh map. Once the expression is plotted on the K-map instead of making the groups of ones, we have to make groups of zeros. Each group of zero results a sum term and it is nothing but the **prime implicate**. The technique for using maps for POS reductions is a simple step by step process and it is similar to the one used earlier.

1. Plot the K-map and place 0s in those cells corresponding to the 0s in the truth table or maxterms in the products of sum expression.
2. Check the K-map for adjacent 0s and encircle those 0s which are not adjacent to any other 0s. These are called isolated 0s.
3. Check for those 0s which are adjacent to only one other 0 and encircle such pairs.
4. Check for quads and octets of adjacent 0s even if it contains some 0s that have already been encircled. While doing this make sure that there are minimum number of groups.
5. Combine any pairs necessary to include any 0s that have not yet been grouped.
6. Form the simplified SOP expression for $\bar{F}$ by summing product terms of all the groups.

(**Note** : The simplified expression is in the complemented form because we have grouped 0s to simplify the expression.)

7. Use DeMorgan's theorem on $\bar{F}$ to produce the simplified expression in POS form.

To get familiar with these steps we will solve some examples.

Example 2.21 : *Minimize the expression*

$$Y = (A + B + \bar{C}) (A + \bar{B} + \bar{C}) (\bar{A} + \bar{B} + \bar{C}) (\bar{A} + B + C)(A + B + C)$$

Solution : $(A + B + \bar{C}) = M_1$, $(A + \bar{B} + \bar{C}) = M_3$, $(\bar{A} + \bar{B} + \bar{C}) = M_7$,

$(\bar{A} + B + C) = M_4$, $(A + B + C) = M_0$

Step 1 : Fig. 2.21 (a) shows the K-map for three variable and it is plotted according to given maxterms.

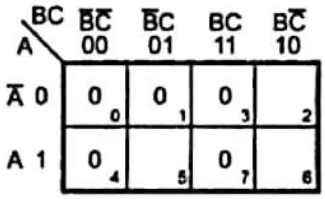


Fig. 2.21 (a)

Step 2 : There are no isolated 0s.

Step 3 : 0 in the cell 4 is adjacent only to 0 in the cell 0 and 0 in the cell 7 is adjacent only to 0 in the cell 3. These two pairs are combined and referred to as group 1 and group 2 respectively.

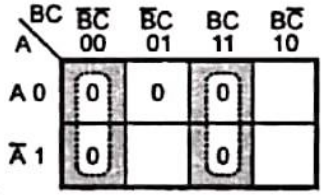


Fig. 2.21 (b)

Step 4 : There are no quads and octets.

Step 5 : The 0 in the cell 1 can be combined with 0 in the cell 3 to form a pair. This pair is referred to as group 3.

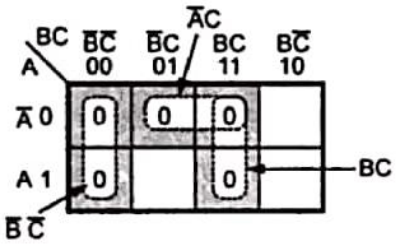


Fig. 2.21 (c)

Step 6 : In group 1 and in group 2, A is eliminated, where as in group 3 variable B is eliminated and we get

$$\bar{Y} = \bar{B}\bar{C} + BC + \bar{A}C$$

Step 7 :

$$Y = \bar{Y} = \overline{\bar{B}\bar{C} + BC + \bar{A}C}$$

$$= \left(\overline{\bar{B}\bar{C}}\right)\left(\overline{BC}\right)\left(\overline{\bar{A}C}\right)$$

$$= \left(\bar{\bar{B}} + \bar{\bar{C}}\right)\left(\bar{B} + \bar{C}\right)\left(\bar{\bar{A}} + \bar{C}\right)$$

$$= (B + C)\left(\bar{B} + \bar{C}\right)\left(A + \bar{C}\right)$$

It is possible to directly write the expression for Y by using DeMorgans theorem for each minterm as follows :

$$\bar{Y} = \bar{B}\bar{C} + BC + \bar{A}C \rightarrow Y = (B + C)\left(\bar{B} + \bar{C}\right)\left(A + \bar{C}\right)$$

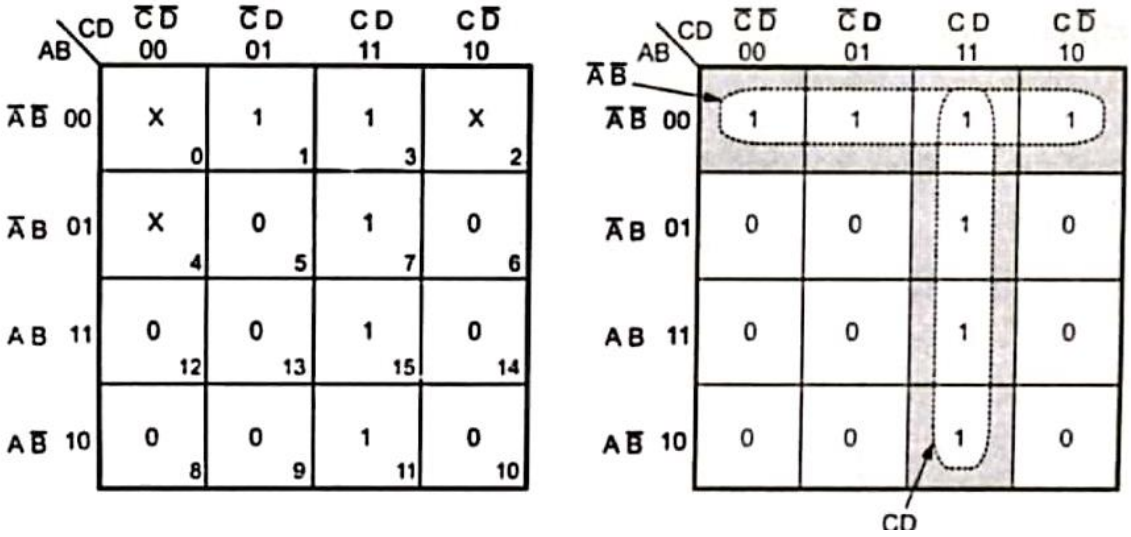
2.9.2 Don't Care Conditions in Logic Design

In this section, we see the example of incompletely specified Boolean function. Let us see the logic circuit for an even parity generator for 4-bit BCD number. The Table 2.15 shows the truth table for even-parity generator. The truth table shows that the output for last six input conditions cannot be specified, because such input conditions does not occur when input is in the BCD form.

Example 2.25 : *Find the reduced SOP form of the following function.*

$$f(A, B, C, D) = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 4).$$

Solution :



To form a quad of cells 0, 1, 2 and 3 the don't care conditions 0 and 2 are replaced by 1s.

The remaining don't care condition is replaced by 0 since it is not required to form any group. With these replacements we get the simplified equation as

$$f(A, B, C, D) = \underset{\text{Group 1}}{\bar{A}\bar{B}} + \underset{\text{Group 2}}{CD}$$