Contents

DIGITAL SYSTEMS (CS-BRANCH)

UNIT 1

REVIEW OF NUMBER SYSTEMS AND NUMBER BASE CONVERSIONS

Q.1. What do you understand by digital circuit and systems ?
(R.G.P.V., June 2009)

Ans. A ***digital circuit*** is one in which the voltage levels assume a finite number of distinct values.

Digital circuits are often called switching circuits, because the voltage levels in a digital circuit are assumed to be switched from one value to another instantaneously, that is the transition time is assumed to be zero.

Digital circuits are also called logic circuits, because each type of digital circuit obeys a certain set of logic rules. The manner in which a logic circuit responds to an input is referred to the circuit's logic.

All of us are familiar with the impact of modern digital computers, communication systems, digital display systems, internet, e-mail etc on society. One of the main causes of this revolution is the advent of integrated circuits (ICs), which became possible because of the tremendous progress in semiconductor technology in recent years. Most of us may not be familiar with the principles of working of computers, communication systems, internet, e-mail, etc. even though these have become an important part of our daily life. The operation of these systems, and many other systems, is based on the principles of digital techniques and these systems are referred to as ***digital systems***.

Q.2. What is meant by number systems ? Discuss its types.

Ans. Generally in any number system there is an ordered set of symbols called digits, which are used to specify any number. The digits are defined for performing arithmetic operations, such as addition, subtraction, multiplication, etc. A collection of these digits forms a number, which in general has two parts, namely integer and fractional. These two parts are set apart by a radix point (.).

In any number representation, the left most digit, which has the large positional weight out of all the digits shown in that number is known as the

most significant bit (MSB) and the right most digit, which has least positional weight out of all the digits present in that number is known as the least significant bit (LSB).

Generally, four types of number systems are used in digital electronics as –

*(i) **Binary Number System*** – It is a positional weighted system. The base of this number system is 2. Only two symbols, namely 0 and 1 are used in this system. These are called ***bits***. The binary number consists of a sequence of bits, each of which is either 0 or 1. In the binary number system, a group of four bits is called ***nibble*** and a group of 8-bits is called ***byte***.

*(ii) **Octal Number System*** – The octal number system is also positional weighted system. The octal number system uses the digits 0, 1, 2, 3, 4, 5, 6 and 7. The base of this system is eight (8). Since its base $8 = 2^3$, every 3-bit group of binary can be represented by an octal digit. The least significant position has weight of 8^0, i.e., 1, the higher significant positions are given weights in the ascending powers of eight (8), i.e., 8^1, 8^2, 8^3, etc., respectively.

*(iii) **Decimal Number System*** – It is a positional weighted system, which means that the value attached to a symbol depends on its location with respect to the decimal point.

The decimal number system contains ten unique symbols, 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9. Since counting in decimal involves ten symbols, its base is ten. The digits to the right of decimal point have weights, which are negative powers of 10 and forms fractional part. The digits to the left of the decimal point have weights, which are positive powers of 10 and forms integer part. The value of decimal number is the sum of the products of the digits of that number with their respective column weights.

Let us consider a mixed decimal number $(N)_b = d_n\, d_{n-1}\, d_{n-2} \ldots d_1 d_0\, d_{-1}\, d_{-2}\, d_{-3} \ldots d_{-k}$. The value of this mixed number is given by –

$$(N)_b = (d_n \times 10^n) + (d_{n-1} \times 10^{n-1}) + \ldots + (d_1 \times 10^1) + (d_0 \times 10^0) + (d_{-1} \times 10^{-1}) + (d_{-2} \times 10^{-2}) + (d_{-3} \times 10^{-3}) + \ldots + (d_{-k} \times 10^{-k}).$$

where $(N)_b$ denotes the value of entire number.

*(iv) **Hexadecimal Number System*** – The base of hexadecimal number system is 16, i.e., it has 16 independent symbols. These 16 symbols are namely 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F. Since its base is 16 $= 2^4$, every 4 binary digit combination can be represented by one hexadecimal digit. Each significant position in an hexadecimal number has a positional weight. The least significant position has a weight of 16^0, i.e., 1, the higher significant positions are given weights in the ascending powers of 16, i.e., 16^1, 16^2, 16^3, etc., respectively.

Q.3. What do you mean by radix 7 ?

Ans. As we know that the decimal number system contains ten unique symbols 0, 1, 2 3, 4, 5, 6, 7, 8 and 9. Since counting in decimal involves ten symbols, we say that its base or radix is ten. Similarly the radix 7 means that it contains 7 symbols and its base or radix is seven.

The principle of positional weighting can be extended to any number system. Any number can be represented by the equation

$$Y = d_n r^n + d_{n-1} r^{n-1} + \ldots + d_1 r^1 + d_0 r^0$$

where Y is the value of the entire number, d_n is the value of the n^{th} digit from the point and r is the radix or base. This equation has already been applied to the different number system. Similarly we can apply it to the radix 7 systems.

Q.4. Do the conversions with the help of example –
(i) Binary to octal (ii) Octal to binary.

Ans. (i) Binary to Octal – The binary numbers can be converted into equivalent octal numbers by making groups of 3-bits starting from least significant bit and moving towards most significant bit for integer part of the number. For fractional part, the 3 bit grouping are made from the starting of binary (radix) point.

Example – $(110101.101010)_2 = (\ )_8$

Group of three bits are $\underbrace{110}_{6}\ \underbrace{101}_{5}\ .\ \underbrace{101}_{5}\ \underbrace{010}_{2}$

Convert each group to octal

The result is $(65.52)_8$ **Ans.**

*(ii) **Octal to Binary*** – To convert an octal number to its equivalent binary number each digit of the given octal number is converted to its 3-bit binary equivalent.

Example – Convert the $(56.34)_8$ to its equivalent binary number.

$$(56.34)_8 = (101)\ (110).(011)\ (100)$$
$$= (101110.011100)_2$$

Q.5. Explain the following conversions with the help of example –
(i) Binary to hexadecimal (ii) Hexadecimal to binary.

Ans. (i) Binary to Hexadecimal – Binary number can be converted into the equivalent hexadecimal numbers by making groups of four bits starting from LSB and moving towards MSB for integer part and then replacing each group of four bits by its hexadecimal representation. Consider an example to convert binary number $(1101\ 0010\ 110\ 111\ 0101)_2$ to hexadecimal number.

0011	0100	1011	0111	0101
3	4	B	7	5

$$= (34B75)_{16}$$

(ii) Hexadecimal to Binary – Hexadecimal numbers can be converted into equivalent binary numbers by replacing each next digit by its equivalent 4 bit binary number. This is represented by the example given below –

$(A352.B1)_{16}$ = 1010 0011 0101
A 3 5
0010 1011 0001
2 B 1

Thus, $(A352.B1)_{16} = (1010001101010010.10110001)_2$

Q.6. Explain the hexadecimal to decimal conversion and vice-versa with the help of example.

***Ans.* Hexadecimal to Decimal** – The hexadecimal number to decimal number conversion is done by multiply each digit in the hexadecimal number with their weight of its position and add all the product terms.

Example – $(5E2C.7B)_{16} = (\ )_{10}$

$$(5E2C.7B)_{16} = 5 \times 16^3 + 14 \times 16^2 + 2 \times 16^1 + 12 \times 16^0 + 7 \times 16^{-1} + 11 \times 16^{-2}$$
$$= 20480 + 3584 + 32 + 12 + 0.4375 + 0.04297$$
$$= (24108.4805)_{10}$$

Decimal to Hexadecimal – For the conversion of a decimal number to an equivalent hexadecimal number, the decimal number is divided by 16 successively. For the conversion of decimal fraction to its equivalent hexadecimal fraction the technique of repeated multiplication by 16 is used. The integer part is note down after each multiplication and the new remainder fraction is used for multiplication at the next stage.

Example – Convert $(2586)_{10}$ to its hexadecimal equivalent.

	Quotient	Remainder
$2586 \div 16$	161	10 = A
$161 \div 16$	10	1
$10 \div 16$	0	10 = A

Thus $(2586)_{10} = (A1A)_{16}$

It may be noted that the remainder of the division processes from the digits of the hexadecimal number and the remainders that are greater than are represented by the letters A through F.

Q.7. Explain the following conversions with the help of example –
(i) Octal to hexadecimal (ii) Hexadecimal to octal.

Ans. (i) Octal to Hexadecimal – To convert an octal number to hexadecimal, the steps are as follows –

(a) Convert the given octal number to its binary equivalent

(b) Form groups of 4-bits, starting from the LSB

(c) Write the equivalent hexadecimal number for each group of 4-bits.

Example – Convert $(46.57)_8$ to its hexadecimal equivalent.

Converting $(46.57)_8$ first to its binary equivalent, we get

$$(46.57)_8 = (100)\ (110).\ (101)\ (111) = (100\ 110.\ 101111)_2$$

Now, forming the groups of 4 binary bits to obtain its hexadecimal equivalent we have

$$(100\ 110.101111) = (10)\ (0110).(1011)\ (11)$$
$$= (0010)\ (0110).(1011)\ (1100) = (26.BC)_{16}$$

(ii) Hexadecimal to Octal – To convert a hexadecimal number to octal, the following steps can be applied –

(a) Convert the given hexadecimal number to its binary equivalent.

(b) Form groups of 3-bits, starting from the LSB.

(c) Write the equivalent octal number for each group of 3-bits.

Example – Convert $(47)_{16}$ to its octal equivalent.

$$(47)_{16} = (0100\ 0111)_2 = (01000111)_2 = (107)_8$$

Thus, 47 in hexadecimal is equivalent to 107 in the octal number system.

Q.8. Explain the binary to decimal conversion and vice-versa.

***Ans.* Binary to Decimal** – To convert a binary number to its decimal equivalent we use the following expression –

The weight of the nth bit of the number from the right hand side = nth bit $\times$ $(Base)^{n-1}$.

First we mark the bit position and then we give the weight of each bit of the number depending on its position. The sum of the weights of all bits gives the equivalent number.

Example – $(11101.110)_2 = (\ )_{10}$

Positional weights –	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
Binary number –	1	1	1	0	1 .	1	1	0

$$(11101.110)_2 = (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2}) + (0 \times 2^{-3})$$
$$= 16 + 8 + 4 + 0 + 1 + 0.5 + 0.25 + 0 = (29.75)_{10}$$

Decimal to Binary – There are two methods, which are used to convert a binary number to a decimal number, namely, sum of weights method and double dabble method.

In sum of weights method, the set of binary weight values whose sum is equal to the decimal number is determined.

In the double dabble method, the decimal integer number is converted to binary integer number by successive division of 2 and the decimal fraction is converted to binary fraction by successive multiplication of 2. In this method the given decimal number is successively divided by 2 till the quotient is zero. The last remainder is the MSB. Thus, the integer numbers read from top to bottom give the equivalent binary fraction. To convert a mixed number to binary, convert the integer and fraction parts individually to binary and then combine them.

Example – Convert the decimal number 15.85 into binary.

Integer Part	Quotient	Remainder
$15 \div 2$	7	1
$7 \div 2$	3	1
$3 \div 2$	1	1
$1 \div 2$	0	1

Read

15 (Decimal number) = 1111 (Binary number)

Fractional Part –

Fraction	*Fraction × 2*	*Remainder New Fraction*	*Integer*
0.85	1.7	0.7	1 (MSB)
0.7	1.4	0.4	1
0.4	0.8	0.8	0
0.8	1.6	0.6	1
0.6	1.2	0.2	1
0.2	0.4	0.4	0 (LSB)

Thus $(0.85)_{10} = (0.110110)_2$

Therefore, $(15.85)_{10} = (1111.110110)_2$

Q.9. Explain the following conversions with the help of examples –
(i) Octal to decimal (ii) Decimal to octal.

Ans. (i) Octal Number into Decimal Number – The octal number to decimal number conversion is done by multiply each digit in the octal number with their weight of its position and add all the product terms.

Example – $(4154.24)_8 = (\ )_{10}$

$$(4154.24)_8 = (4 \times 8^3) + (1 \times 8^2) + (5 \times 8^1) + (4 \times 8^0) + (2 \times 8^{-1}) + (4 \times 8^{-2})$$

$$= (2048) + (64) + (40) + (4) + (0.25) + (0.0625)$$

Thus, $(4154.24)_8 = (2156.3125)_{10}$

(ii) Decimal Number into Octal Number – To convert the given decimal integer number to octal number, successively divide the given number by factor 8 till the quotient is 0. The last remainder is the MSB. The remainder read from bottom to top give the equivalent octal integer number. To convert the given decimal fractional number to octal fractional number, successively multiply the decimal fractional number by factor 8 till the product is 0 or till the required accuracy is obtained. The first integer from the top is the MSB. The integer read downward to give the octal fractional number.

Example – Convert $(444.96)_{10}$

Integer Part	Quotient	Remainder
$444 \div 8$	55	4
$55 \div 8$	6	7
$6 \div 8$	0	6

Reading the remainders from bottom to top, the decimal number $(444)_{10}$ is equivalent to octal $(674)_8$

Fractional Part –

Fraction	*Fraction × 8*	*Remainder New Fraction*	*Integer*
0.96	7.68	0.68	7 (MSB)
0.68	5.44	0.44	5
0.44	3.52	0.52	3
0.52	4.16	0.16	4
0.16	1.28	0.28	1 (LSB)

This process will continue further, so may take the result upto 5 places of octal point

$$(0.96)_{10} = (0.75341)_8$$

Hence the result $(444.96)_{10} = (674.75341)_8$

Q.10. What do you mean by signed binary numbers ?

Ans. In case of signed binary number, positive integer including zero can be represented as unsigned number. To represent negative integer, we need a notation for negative values. In ordinary arithmetic a negative number is indicated by minus sign and a positive number by a plus sign. Because of computer hardware limitation, computers must represent every thing with binary digits. It is usually to represent the sign with a bit placed in the leftmost position of the number. The convention is to make the sign bit 0 for positive and 1 for negative.

For example, the string of bits 00101 can be considered as 5 (unsigned binary) or + 5 (signed binary) because the left most bit is 0. The string of bit 10101 represents the binary equivalent of 21 when considered as an unsigned number or

as – 5 when considered as a signed number. This is because the 1 that is in the leftmost position designates a negative and the other 4-bits represent binary 5.

The signed magnitude, – 5 is obtained from + 5 by changing the sign bit in the leftmost position from 0 to 1. In signed – 1's complement, – 5 is obtained by complementing all the bits of + 5, including the sign bit. The signed – 2's complement representation of – 5 is obtained by taking the 2's complement of the positive number including the sign bit.

The signed-2's complement system has only one representation for 0, which is always positive. The signed magnitude system is used in ordinary arithmetic, but is awkward if employed in computer arithmetic because of separate handling of the sign and the magnitude.

Q.11. What do you mean by 1's complement representation ? Explain with example.

Ans. The 1's complement in the binary number system is similar to 9's complement in the decimal system. To obtain 1's complement of a binary number each bit of the binary number is subtracted from 1. Thus 1's complement of a binary number may be formed by simply changing each 1 to a 0 and each 0 to a 1.

Example – The 1's complement of the binary number 010 is 101.

Q.12. What do you mean by 2's complement representation ? Explain with example.

Ans. The 2's complement in the binary number system is similar to 10's complement in the decimal number system. The 2's complement of a binary number is equal to the 1's complement of the number plus one.

The 2's complement of a binary number = Its 1's complement + 1.

Example – The 2's complement of 0101 = 1010 + 1 = 1011.

Numerical Problems

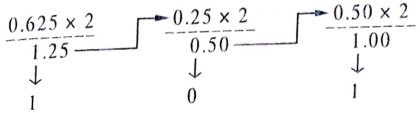
Prob.1. Convert the following (i) $(48.625)_{10}$ = ( )$_2$ (ii) Divide $(1EC87)_{16}$ by $(A5)_{16}$. ***(R.G.P.V., June 2014***

Sol. (i) Conversion of integer part $(48)_{10}$ –

Successive Division		Remainder
2 | 48		0 (LSB)
2 | 24		0
2 | 12		0
2 | 6		0
2 | 3		1
	1	1 (MSB)

i.e., $(48)_{10} = (110000)_2$

Conversion of fractional part $(0.625)_{10}$ –



i.e., $(0.625)_{10} = (0.101)_2$

Hence, $(48.625)_{10} = \mathbf{(110000.101)_2}$ **Ans.**

(ii) Divide $(1EC87)_{16}$ by $(A5)_{16}$

$$
\begin{array}{r}
2\ F\ C \\
A5\overline{)1\ E\ C\ 8\ 7} \\
1\ 4\ A \\
\hline
A\ 2\ 8 \\
9\ A\ B \\
\hline
7\ D\ 7 \\
7\ B\ C \\
\hline
1\ B
\end{array}
$$

Remainder **Ans.**

Prob.2. Convert the following –

(i) $(0.513)_{10}$ to octal

(ii) $(673.124)_8$ to binary

(iii) $(1010.01101)_2$ to decimal.

(R.G.P.V., Dec. 2010)

Sol. (i) $(0.513)_{10} = 0.513 \times 8 = 4.104$

$0.104 \times 8 = 0.832$

$0.832 \times 8 = 6.656$

$0.656 \times 8 = 5.248$

$0.248 \times 8 = 1.984$

$0.984 \times 8 = 7.872$

$(0.513)_{10} = \mathbf{(0.406517\ .....)_8}$ **Ans.**

(ii) $(673.124)_8 = (6\ \ 7\ \ 3\ .\ 1\ \ 2\ \ 4)_8$

$= (110\ \ 111\ \ 011\ .\ 001\ \ 010\ \ 100)_8$

$(673.124)_8 = \mathbf{(110111011.001010100)_2}$ **Ans.**

(iii) $(1010.01101)_2 = (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0) + (0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + 1 \times 2^{-5})$

$= 8 + 2 + 0.25 + 0.125 + 0.03125$

$= \mathbf{(10.40625)_{10}}$ **Ans.**

Prob.3. Convert $(412)_{10}$ to –

(i) Binary (ii) Octal (iii) Hexadecimal.

(R.G.P.V., Dec. 2017)

Sol. (i) $(412)_{10} = (?)_2$

Successive Division		Remainder
2	412	0 (MSB)
2	206	0
2	103	1
2	51	1
2	25	1
2	12	0
2	6	0
2	3	1
2	1	1 (LSB)
	0	

Hence, $(412)_{10} = \mathbf{(110011100)_2}$ **Ans**

(ii) $(412)_{10} = (?)_8$

Successive Division		Remainder
8	412	4 (MSB)
8	51	3
8	6	6 (LSB)
	0	

Hence, $(412)_{10} = \mathbf{(634)_8}$ **Ans**

(iii) $(412)_{10} = (?)_{16}$

Successive Division		Remainder
16	412	12 = C
16	25	9 = 9
	1	1 = 1

Hence, $(412)_{10} = \mathbf{(19C)_{16}}$ **Ans**

Prob.4. Convert the following –

(i) Decimal 225.225 to binary, octal and hexadecimal.

(ii) Binary 11010111.110 to decimal, octal and hexadecimal.

(R.G.P.V., June 2017)

Sol. (i) (a) $(225.225)_{10} = (?)_2$

Conversion of integer part $(225)_{10}$ –

Successive Division		Remainder
2	225	1 (MSB)
2	112	0
2	56	0
2	28	0
2	14	0
2	7	1
2	3	1
	1	1 (LSB)

i.e., $(225)_{10} = (11100001)_2$

Conversion of fractional part $(0.225)_{10}$ –

$0.225 \times 2 = 0.45$ → 0; $0.45 \times 2 = 0.90$ → 0; $0.90 \times 2 = 1.80$ → 1; $0.80 \times 2 = 1.60$ → 1; $0.60 \times 2 = 1.20$ → 1

i.e., $(0.225)_{10} = (0.0011)_2$

Hence, $(225.225)_{10} = \mathbf{(11100001.0011.....)_2}$ **Ans.**

(b) $(225.225)_{10} = (?)_8$

Conversion of integer part $(225)_{10}$ –

Successive Division		Remainder
8	225	→ 1
8	28	→ 4
8	3	→ 3
	0	

i.e., $(225)_{10} = (341)_8$.

Conversion of fractional part $(0.225)_{10}$ –

Multiplication	Generated Integer
$0.225 \times 8 = 1.8$	1
$0.8 \times 8 = 6.4$	6
$0.4 \times 8 = 3.2$	3
$0.2 \times 8 = 1.6$	1
$0.6 \times 8 = 4.8$	4

Hence, $(225.225)_{10} = \mathbf{(341.16314.....)_8}$ **Ans.**

(c) $(225.225)_{10} = (?)_{16}$

Conversion of integer part $(225)_{10}$ –

Successive Division		Remainder
16	225	1 = 1
16	14	14 = E
8	0	

i.e., $(225)_{10} = (E1)_{16}$

Conversion of fractional part $(0.225)_{10}$ –

0.225×16	3.6
0.600×16	9.6
0.600×16	9.6

i.e. $(0.225)_{10} = (0.399.....)_{16}$

Hence,

$(225.225)_{10}$ = **$(E1.399.....)_{16}$** **Ans.**

(ii) (a) $(11010111.110)_2 = (\ )_{10}$

$= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}$

$= 128 + 64 + 0 + 16 + 0 + 4 + 2 + 1 + 0.5 + 0.25 + 0$

$=$ **$(215.75)_{10}$** **Ans.**

(b) $(11010111.110)_2 = (\ )_8$ – To convert given binary number into octal number, grouping this binary number into 3-bit equivalent octal digit. Therefore,

$= (\underbrace{011}_{3}\ \underbrace{010}_{2}\ \underbrace{111}_{7} \cdot \underbrace{110}_{6})_2$ = **$(327.6)_8$** **Ans.**

(c) $(11010111.110)_2 = (\ )_{16}$

$= (\underbrace{1101}_{D}\ \underbrace{0111}_{7} \cdot \underbrace{1100}_{C})_2$

Hence,

$(11010111.10)_2$ = **$(D7.C)_{16}$** **Ans.**

Prob.5. Convert the number $(210.25)_{10}$ to base 2, 8. ***(R.G.P.V., June 2015)***

Sol. (i) Given, $(210.25)_{10} = (\ )_2$

Conversion of integer part $(210)_{10}$ –

Successive Division		Remainder	
2	210	0	(LSB)
2	105	1	
2	52	0	
2	26	0	
2	13	1	
2	6	0	
2	3	1	
2	1	1	(MSB)
	0		

i.e., $(210)_{10} = (11010010)_2$

Conversion of fractional part $(0.25)_{10}$ –

$0.25 \times 2 = 0.50$ (0)

$0.50 \times 2 = 1.00$ (1)

i.e., $(0.25)_{10} = (0.01)_2$

Hence, $(210.25)_{10}$ = **$(11010010.01)_2$** **Ans.**

(ii) $(210.25)_{10} = (\ )_8$

Conversion of integer part $(210)_{10}$ –

Successive Division		Remainder	
8	210	2	(LSB)
8	26	2	
8	3	3	(MSB)
	0		

i.e., $(210)_{10} = (322)_8$

Conversion of fractional part $(0.25)_{10}$ –

$0.25 \times 8 = 2.00$ (2)

i.e., $(0.25)_{10} = (0.2)_8$

Hence, $(210.25)_{10}$ = **$(322.2)_8$** **Ans.**

Prob.6. Convert the following –

(i) $(B65F)_{16} = (\ )_{10}$

(ii) $(153.25)_{10} = (\ )_2$

(iii) $(10110001101011)_2 = (\ )_8$

(iv) $(153)_{10} = (\ )_8$.

(R.G.P.V., June 2010)

Sol. (i) $(B65F)_{16} = B \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + F \times 16^0$

$= 11 \times 16^3 + 6 \times 16^2 + 5 \times 16 + 15 \times 1$

$=$ **$(46687)_{10}$** **Ans.**

(ii) $(153.25)_{10} = (\ )_2$

Successive Division		Remainder	
2	153	1	LSB
2	76	0	
2	38	0	
2	19	1	
2	9	1	
2	4	0	
2	2	0	
2	1	1	MSB
	0		

$(153)_{10} = (10011001)_2$

Conversion of fractional part $(.25)_{10}$ –

$.25 \times 2 = 0.50$ → $.50 \times 2 = 1.00$

0 1

$(.25)_{10} = (.01)_2$

Thus $(153.25)_{10} = \mathbf{(10011001.01)_2}$ **Ans.**

(iii) $(10110001101011)_2 = \underbrace{010}_{2}\ \underbrace{110}_{6}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{011}_{3}$

$= \mathbf{(26153)_8}$ **Ans.**

(iv) $(153)_{10} = (\ )_8$

		Remainder
8	153	1
8	19	3
8	2	2
	0	

$(153)_{10} = \mathbf{(231)_8}$ **Ans.**

Prob.7. Convert –

(i) 0.1011 → Decimal (Binary) ***(ii) $23_{(8)}$ → Decimal***

(iii) $(9AF)_{(16)}$ → Binary. ***(R.G.P.V., Dec. 2013)***

Sol. (i) $(0.1011)_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}$

$= 0.5 + 0.125 + 0.0625 = \mathbf{(0.6875)_{10}}$ **Ans.**

(ii) $23_{(8)} = 2 \times 8^1 + 3 \times 8^0 = 16 + 3 = \mathbf{(19)_{10}}$ **Ans.**

(iii) $(9AF)_{16} = (\underset{\downarrow}{9}\ \ \underset{\downarrow}{A}\ \ \underset{\downarrow}{F})$

$(1001\ \ 1010\ \ 1111)_2$

$= \mathbf{(100110101111)_2}$ **Ans.**

Prob.8. Convert the following –

(i) $(4213.321)_5 = (\ )_{10}$

(ii) $(10101101.110101001) = (\ )_8$

(iii) $(715230.3214)_8 = (\ )_{16}$

(iv) $(7524.4251)_{10} = (\ )_2$.

(R.G.P.V., June 2012)

Sol. (i) $(4213.321)_5 = 4 \times 5^3 + 2 \times 5^2 + 1 \times 5^1 + 3 \times 5^0 + 3 \times 5^{-1} + 2 \times 5^{-2} + 1 \times 5^{-3}$

$= 500 + 50 + 5 + 3 + \frac{3}{5} + \frac{2}{25} + \frac{1}{125}$

$= \mathbf{(558.688)_{10}}$ **Ans.**

(ii) $(10101101.110101001)_2 = (\underbrace{010}_{2}\ \underbrace{101}_{5}\ \underbrace{101}_{5}\ .\ \underbrace{110}_{6}\ \underbrace{101}_{5}\ \underbrace{001}_{1})$

$= \mathbf{(255.651)_8}$ **Ans.**

(iii) $(715230.3214)_8 = 111\ 001\ 101\ 010\ 011\ 000\ .\ 011\ 010\ 001\ 100$

$= \underbrace{0011}_{3}\ \underbrace{1001}_{9}\ \underbrace{1010}_{A}\ \underbrace{1001}_{9}\ \underbrace{1000}_{8}\ .\ \underbrace{0110}_{6}\ \underbrace{1000}_{8}\ \underbrace{1100}_{C}$

$= \mathbf{(39A98.68C)_{16}}$ **Ans.**

(iv) $(7524.4251)_{10} = (\ )_2$

Conversion of integer $(7524)_{10}$ –

Successive Division		Remainder
2	7524	0 (LSB)
2	3762	0
2	1881	1
2	940	0
2	470	0
2	235	1
2	117	1
2	58	0
2	29	1
2	14	0
2	7	1
2	3	1
2	1	1 (MSB)
	0	

i.e., $(7524)_{10} = (1110101100100)_2$

Conversion of fraction $(0.4251)_{10}$ –

$0.4251 \times 2 = 0.8502$ → $0.8502 \times 2 = 1.7004$ → $0.7004 \times 2 = 1.4008$ → $0.4008 \times 2 = 0.8016$ → $0.8016 \times 2 = 1.6032$

0 1 1 0 1

i.e., $(0.4251)_{10} = (0.01101\ \ldots\ldots\ )_2$

Therefore the final result is

$(7524.4251)_{10} = \mathbf{(1110101100100.011011\ \ldots\ldots\ )_2}$ **Ans.**

Prob.9. Do as directed –

(i) $(56)_{16} = (?)_{10}$

(ii) $(32)_{10} = (?)_2$

(iii) Bubbled OR gate is also called ..... .

(R.G.P.V., Dec. 2016)

Sol. (i) $(56)_{16} = 5 \times 16^1 + 6 \times 16^0$
$= (86)_{10}$ Ans.

(ii) $(32)_{10} = (\ )_2$

Successive Division

2	32
2	16
2	8
2	4
2	2
2	1
	0

Remainder

0 LSB
0
0
0
1 MSB

Hence $(32)_{10} = (100000)_2$ Ans.

(iii) Bubbled OR gate is also called **NOR gate.** Ans.

Prob.10. Multiply $(1AB)_{16}$ by $(89)_{16}$. *(R.G.P.V., Feb. 2010)*

Sol. First change given hexadecimal number into binary then multiply it

$(1\ AB)_{16} = (0001\ 1010\ 1011)_2$
$(89)_{16} = (1000\ 1001)_2$

$000110101011 \times 10001001$

$$
\begin{array}{r}
000110101011 \\
000000000000\times \\
000000000000\times \\
000110101011\times \\
000000000000\times \\
000000000000\times \\
000000000000\times \\
000110101011\times \\
\hline
000111001001000000011
\end{array}
$$

$(111001001000000011)_2$

Now change this binary equivalent into hexadecimal.

1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1 1

E 4 8 3

Hence, Ans

$(1AB)_{16} \times (89)_{16} = (E483)_{16}$

Prob.11. Obtain M – N using 1's complement and 2's complement if –
(i) M = 10110101, N = 00101101
(ii) M = 00101101, N = 11101011.

(R.G.P.V., Feb. 2010)

Sol. Given, $M = 10110101$
$N = 00101101$

Using 1's Complement –

M → 1 0 1 1 0 1 0 1
1's complement of N → 1 1 0 1 0 0 1 0
Carry → 1 1 0 0 0 0 1 1 1
Add carry → 1
1 0 0 0 1 0 0 0

Using 2's Complement –

1's complement of N = 1 1 0 1 0 0 1 0
= 1 1 0 1 0 0 1 0
+ 1
2's complement of N = 1 1 0 1 0 0 1 1
M → 1 0 1 1 0 1 0 1
2's complement of N → 1 1 0 1 0 0 1 1
1 0 0 0 1 0 0 0
↑

Discard carry = 10001000 Ans.

(ii) M = 00101101
N = 11101011

Using 1's Complement –

M → 0 0 1 0 1 1 0 1
1's complement of N → 0 0 0 1 0 1 0 0
0 1 0 0 0 0 0 1

No carry is obtained answer is the 1's complement of 01000001 and it is negative.

= – 10111110 Ans.

Using 2's Complement –

1's complement of N = 0 0 0 1 0 1 0 0
0 0 0 1 0 1 0 0
+ 1
2's complement of N = 0 0 0 1 0 1 0 1
M → 0 0 1 0 1 1 0 1
2's complement of N → 0 0 0 1 0 1 0 1
0 1 0 0 0 0 1 0

There is no carry. The result is in 2's complement form and is negative.
2's complement of 01000010

= 1 0 1 1 1 1 0 1
+ 1
1 0 1 1 1 1 1 0

= – 10111110 Ans.

Prob.12. Subtract using 1's and 2's complement –
(i) 10110101 – 01101101 (ii) 0010101 – 1101001.
(R.G.P.V., June 2012)

Sol. (i) Subtraction using 1's complement –

```
  1 0 1 1 0 1 0 1
+ 1 0 0 1 0 0 1 0   (1's complement form)
Carry (1) 0 1 0 0 0 1 1 1
Add carry         + 1
  0 1 0 0 1 0 0 0
```

Ans.

Subtraction using 2's Complement –

```
Minuend = 1 0 1 1 0 1 0 1
Subtrahend = 0 1 1 0 1 1 0 1
             ↓
  1 0 0 1 0 0 1 0 (1's complement form)
            + 1
  1 0 0 1 0 0 1 1 (2's complement form)
  1 0 1 1 0 1 0 1
+ 1 0 0 1 0 0 1 1 (2's complement form)
Carry (1) 0 1 0 0 1 0 0 0
```

The carry is discarded. The MSB is 0, so the result is positive and is in true binary form

$= \mathbf{01001000}$ **Ans.**

(ii) Subtraction using 1's complement –

```
  0 0 1 0 1 0 1
+ 0 0 1 0 1 1 0 (1's complement form)
  0 1 0 1 0 1 1
```

No carry is obtained. The answer is 1's complement of 0101011 and is opposite in sign i.e. – **1010100** **Ans.**

Subtraction using 2's complement –

```
Minuend = 0 0 1 0 1 0 1
Subtrahend = 1 1 0 1 0 0 1
             ↓
  0 0 1 0 1 1 0 (1's complement form)
            + 1
  0 0 1 0 1 1 1 (2's complement form)
  0 0 1 0 1 0 1
+ 0 0 1 0 1 1 1 (2's complement form)
  0 1 0 1 1 0 0
```

The MSB is 0, so the result is positive and is in true binary form

$= \mathbf{0101100}$ **Ans.**

Prob.13. Subtract $(1010)_2$ from $(1000)_2$ using 1's and 2's complement method. ***(R.G.P.V., Dec. 2014)***

Sol. Using 1's complement method –

```
                1 0 0 0
1's Complement → 0 1 0 1
                1 1 0 1
```

No carry is obtained. Then the answer is 1's complement of 1101 and is opposite in sign i.e. **– 0010.** **Ans.**

Using 2's complement method –

```
                1 0 0 0
2's Complement → 0 1 1 0
                1 1 1 0
```

No carry is obtained. Thus, the difference is negative and the true answer is the 2's complement of $(1110)_2$ i.e. $\mathbf{(0010)_2}$. **Ans.**

Prob.14. Subtract $(595)_{10}$ from $(378)_{10}$. ***(R.G.P.V., June 2009)***

Sol. The binary equivalent of the given decimal numbers are –

$$(378)_{10} = (000101111010)_2$$
$$(595)_{10} = (001001010011)_2$$

The subtraction of two given decimal numbers are given below –

```
(378)10          000101111010 → Minuend
- (595)10      + 110110101101 → 2's complement of subtrahend
  - 217         111100100111
```

Here the final carry is 0, the result is negative (the minuend is smaller than the subtrahend) and is in 2's complement form. The 2's complement of 111100100111 = 000011011001, which is equal to 217. Therefore, the result is $\mathbf{(-217)_{10}}$ **Ans.**

Prob.15. Perform subtraction using 2's complement method – 1100 – 110001. ***(R.G.P.V., June 2009)***

Sol. Subtraction using 2's complement method.

1's complement of 110001 is 001110.

2's complement of 110001 is

```
  001110
+     1
  001111
```

Now

```
    1100
+ 001111
  0 11011
```

Here, end carry is 0, so result is negative and is in 2's complement form. Thus the actual result is

– (2's complement of 011011) = – **100101** **Ans.**

Prob.16. Add and subtract octal numbers 360 and 715. ***(R.G.P.V., Dec. 2015)***

Sol. (i) Add –

$$\begin{array}{rcl} 360 & = & 011110000 \\ +\ 715 & = & 111001101 \\ \hline (1275)_8 & & 1010111101 \end{array}$$

(ii) Subtract –

$$\begin{array}{rcl} (360)_8 & = & 011110000 \\ -(715)_8 & = & 000110011 \leftarrow \text{2's complement of } (715)_8 \\ \hline -(335)_8 & & 100100011 \end{array}$$

2's complement of 100100011 = 011011101 = $(335)_8$

Prob.17. Using 9's complement, subtract (63458 – 3354). ***(R.G.P.V., Dec. 2010)***

Sol. Regular subtraction

$$\begin{array}{r} 63458 \\ -\ 3354 \\ \hline 60104 \end{array}$$

9's complement subtraction

$$\begin{array}{rl} & 63458 \\ +\ & 6645 \quad \text{9's complement of 3354} \\ \hline & 70103 \\ & \downarrow \\ \hline -\ & 29896 \quad \text{9's complement of 70103} \end{array}$$

Therefore the ans. is – **29896** **Ans.**

Prob.18. Subtract the following number using 10's complement method – 786 – 427. ***(R.G.P.V., Nov./Dec. 2007)***

Sol. Given, Minuend = 786

Subtrahend = 427

$$\begin{array}{rl} & 999 \\ \text{9's complement of subtrahend} = & -\ 427 \\ \hline & 572 \\ & 572 \\ \text{10's complement of subtrahend} = & +\quad 1 \\ \hline & 573 \end{array}$$

Add this to minuend –

$$\begin{array}{rl} & 786 \\ & +\ 573 \\ \hline \text{End-around Carry (EAC)} \leftarrow & \boxed{1}\,359 \\ & \uparrow \\ & \text{Ignore} \end{array}$$

Therefore the result is **359** **Ans.**

BINARY CODES

Q.13. Explain number systems and codes. ***(R.G.P.V., June 2011)***

Ans. **Number Systems** – Refer the ans. of Q.2.

Codes – Code is a symbolic representation of discrete information, which may be present in the form of numbers, letters or physical quantities. The symbols used are the binary digits 0 and 1 which are arranged according to the rules of codes. These codes are used to communicate information to a digital computer and to retrieve messages from it. A code is used to enable an operator to feed data into a computer directly, in the form of decimal numbers, alphabets and special characters. The computer converts these data into binary codes and after, computation, transforms the data into its original format. When numbers, letters or words are represented by a special group of symbols, this is called encoding and the group of symbols is called a ***code***.

Some important groups of codes are given below –

(i) Weighted binary codes (ii) Non-weighted codes

(iii) Error-detecting codes (iv) Error-correcting codes

(v) Alphanumeric codes.

Q.14. Explain the difference between a weighted and non-weighted code with example. ***(R.G.P.V., June 2015)***

Ans. **Weighted Codes** – The codes which follow the positional weighting principles are known as weighted binary codes. Each position of a number represents a specific weight. In a weighted binary code the bits are multiplied by the weights indicated, the sum of these weighted bits give the equivalent decimal digit. Some weighted 4-bit binary codes with their decimal numbers is shown in table 1.1.

Table 1.1 Some Weighted 4-bit Binary Codes

Decimal Numbers	*BCD or 8421 Code*	*2421 Code*	*5421 Code*
0	0000	0000	0000
1	0001	0001	0001
2	0010	0010	0010
3	0011	0011	0011
4	0100	0100	0100
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

(i) BCD Code – The binary coded decimal uses the binary number system to specify the decimal numbers 0 to 9. The weights are assigned according to the positions occupied by these digits. The weights of the right most (or first) position is 2^0 (or 1), the second position is 2^1 (or 2), the third position is 2^2 (or 4) and the fourth position is 2^3 (or 8). The weights are 8-4-2-1 as reading from left to right and hence it is known as 8421 code.

(ii) 2421 Code – The 2421 code is weighted binary code and its weights are 2,4,2 and 1. A decimal number is represented in 4-bit form and the total weight of the four-bit = 2 + 4 + 2 + 1 = 9. Thus, the 2421 code shows decimal number from 0 to 9. This code is also a self complementing code, i.e., the 9's complement of number 'n' is obtained by complementing the 1's and 0's in the code word.

Non-weighted Codes – The codes which are not positionally weighted is called non-weighted codes. Means of that each position within a binary number is not assigned a fixed values. The Gray codes and Excess-3 codes are non-weighted codes.

(i) Gray Code – The Gray code is very useful code in which a decimal number is represented in binary form in such a manner, so each Gray code number differs from the preceding and the succeeding number by a one-bit. The Gray code is a non-weighted code and therefore it is not suitable for arithmetic operations. The Gray code is applicable in input/output devices and in some types of analog to digital converters. The Gray code is a reflective digital code which contains a special property of containing two adjacent code numbers that differ by only one-bit. The Gray code is also known as ***unit distance code***. The Gray code is used where the normal sequence of binary numbers may produce an error during the transition from one number to the next number.

The Gray code illustration for the decimal number together with binary code is depicted in table 1.2.

Table 1.2 Gray Code Illustration for Decimal Numbers with Binary Code

Decimal Numbers	*Binary Code* $B_3B_2B_1B_0$	*Gray Code* $G_3G_2G_1G_0$
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

(ii) Excess-3 Code – This code itself is complementing code which means that the 1's complement of the coded number gives 9's complement of the number itself. An Excess-3 code is a reflective code and is obtained by adding 3 to a decimal number. For example, Excess-3 code of decimal 2 is 0101, its 1's complement is 1010 which is Excess-3 for decimal 7. This is 9's complement of decimal 2. The self complementary property of Excess-3 code helps in performing subtraction operation in digital systems. Table 1.3 illustrates the BCD code, Excess-3 code for decimal numbers.

Table 1.3 Illustration of Excess-3 and BCD Codes for Decimal Numbers

Decimal Numbers	*BCD Code*				*Excess-3 Code*			
	D	*C*	*B*	*A*	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Q.15. What do you mean by radix 7 ? Also explain reflected code. ***(R.G.P.V., June 2009)***

Ans. **Radix 7** – Refer the ans. of Q.3.

Reflected Code – When the code for 9 is the complement of the code for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4 then a code is said to be reflective.

Q.16. Explain the self complementing code with example. ***(R.G.P.V., Dec. 2006)***

Ans. A code is said to be self complementing, if the code word of 9's complement of N, i.e., of 9-N can be obtained from code word of N by interchanging all the 0's and 1's. Therefore in self complementing code, the code for 9 is the complement of the code for 0, 8 for 1, 7 for 2, 6 for 3, and 5 for 4. The 2421, 5211, 642-3 and XS-3 are self complementing codes.

For a code to be self complementing sum of all its weight must be 9. This is because whatever be the weight, 0 is to be represented by 0000 and since in a self complementing code, the code for 9 is the complement of the code for 0, 9 has to be represented by 1111. There are only four (2421, 5211, 3321, 4311) positively-weighted, self complementing codes. There are 13 negatively weighted self complementing codes.

Q.17. What are the alphanumeric codes ? ***(R.G.P.V., June 2007)***

Ans. An alphanumeric (sometimes abbreviated alphameric) code is a binary code of a group of elements consisting of ten decimal digits, the 26 letters of the alphabet and a certain number of special symbols such as $. The total number of elements in an alphanumeric group is greater than 36. Therefore it must be coded with minimum of six bits ($2^6 = 64$).

One possible arrangement of a six bit alphanumeric code is shown in table 1.4 under the name "internal code". The need to represent more than 64 characters gave rise to seven and eight bit alphanumeric codes. A such type code is known as ASCII (American Standard Code for Information Interchange), another is known as EBCDIC (Extended BCD Interchange Code).

Table 1.4 Alphanumeric Character Codes

Character	*6-bit*		*7-bit*		*8-bit*	
	Internal	*Code*	*ASCII*	*Code*	*EBCDIC*	*Code*
A	010	001	100	0001	1100	0001
B	010	010	100	0010	1100	0010
C	010	011	100	0011	1100	0011
D	010	100	100	0100	1100	0100
E	010	101	100	0101	1100	0101
F	010	110	100	0110	1100	0110
G	010	111	100	0111	1100	0111
H	011	000	100	1000	1100	1000
I	011	001	100	1001	1100	1001
J	100	001	100	1010	1101	0001
K	100	010	100	1011	1101	0010
L	100	011	100	1100	1101	0011
M	100	100	100	1101	1101	0100
N	100	101	100	1110	1101	0101
O	100	110	100	1111	1101	0110
P	100	111	101	0000	1101	0111
Q	101	000	101	0001	1101	1000
R	101	001	101	0010	1101	1001
S	110	010	101	0011	1110	0010
T	110	011	101	0100	1110	0011
U	110	100	101	0101	1110	0100
V	110	101	101	0110	1110	0101
W	110	110	101	0111	1110	0110
X	110	111	101	1000	1110	0111
Y	111	000	101	1001	1110	1000
Z	111	001	101	1010	1110	1001
0	000	000	011	0000	1111	0000
1	000	001	011	0001	1111	0001
2	000	010	011	0010	1111	0010
3	000	011	011	0011	1111	0011
4	000	100	011	0100	1111	0100
5	000	101	011	0101	1111	0101
6	000	110	011	0110	1111	0110
7	000	111	011	0111	1111	0111
8	001	000	011	1000	1111	1000
9	001	001	011	1001	1111	1001
Blank	110	000	010	0000	0100	0000
.	011	011	010	1110	0100	1011
(	111	100	010	1000	0100	1101
+	010	000	010	1011	0100	1110
$	101	011	010	0100	0101	1011
*	101	100	010	1010	0101	1100
)	011	100	010	1001	0101	1101
–	100	000	010	1101	0110	0000
/	110	001	010	1111	0110	0001
,	111	011	010	1100	0110	1011
=	001	011	011	1101	0111	1110

Q.18. Explain Hamming and block codes. ***(R.G.P.V., June 2011)***

Ans. **Hamming Code –** The Hamming code is a type of error correcting code. R.W. Hamming developed a system that provides a mathematical way to add one or more parity bits to a data character in order to detect and correct errors. The Hamming distance between two code words is defined as the number of bits changed from one code word to another.

Consider C_i and C_j to be any two code words in a particular block code. The Hamming distance d_{ij} between the two vectors C_i and C_j is defined by the number of components in which they differ. Assuming that d_{ij} is determined for each pair of code words, the minimum value of the d_{ij} can be called the Hamming distance d_{min}. For linear block codes, minimum weight is equal to minimum distance.

For example,

$$C_i = 1\ 0\ 0\ 0\ 1\ 1\ 1$$
$$\downarrow \quad\quad \downarrow\ \downarrow$$
$$C_j = 0\ 0\ 0\ 1\ 0\ 1\ 1$$

Here, these code words differ in the leftmost bit position and in the fourth and fifth bit positions from the left. Accordingly, $d_{ij} = d_{min} = 3$.

The 7-bit Hamming (7,4) code word $h_1\ h_2\ h_3\ h_4\ h_5\ h_6\ h_7$ associated with a 4-bit binary number $b_3\ b_2\ b_1\ b_0$ is given as –

$$h_1 = b_3 \oplus b_2 \oplus b_0$$
$$h_2 = b_3 \oplus b_1 \oplus b_0,\ h_3 = b_3$$
$$h_4 = b_2 \oplus b_1 \oplus b_0,\ h_5 = b_2$$
$$h_6 = b_1,\ h_7 = b_1$$

where, $\oplus$ denotes the EX-OR operation. Note that bits h_1, h_2 and h_4 are even parity bits for the bit fields $b_3\ b_2\ b_0$, $b_3\ b_1\ b_0$ and $b_2\ b_1\ b_0$, respectively. In general, the parity bits (h_1, h_2, h_4, h_8, ...) are located in the positions corresponding to ascending powers of two (i.e., 2^0, 2^1, 2^2, 2^3, ... = 1, 2, 4, 8,..).

The h_1 parity bit has a 1 in the LSB of its binary representation. Therefore, it checks all bit positions, including it, that have 1's in the same location (i.e., LSB) in the binary representation (i.e., h_1, h_3, h_5 and h_7). The binary representation of h_2 has a 1 in the middle bit. Therefore, it checks all bit positions, including it, that have 1's in the same location (i.e., middle bit) in the binary representation (i.e., h_2, h_3, h_6 and h_7). The binary representation of h_4 has a 1 in the MSB. Therefore, it checks all bit positions, including it, that have 1's in the same location (i.e., MSB) in the binary representation (i.e., h_4, h_5, h_6 and h_7).

To decode a Hamming code, one must check for odd parity over the bit fields in which even parity was previously established. For example, a single bit error is indicated by a non-zero parity word $c_4\ c_2\ c_1$.

where,
$$c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7$$
$$c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7$$
$$c_4 = h_4 \oplus h_5 \oplus h_6 \oplus h_7$$

If $c_4\ c_2\ c_1 = 000$, there is no error in the Hamming code. If it has a non-zero value, it indicates the bit position in error. For example, if $c_4\ c_2\ c_1 = 101$, bit 5 is in error. To correct this error, bit 5 has to be complemented.

Block Codes – Block codes are also known as ***arithmetic codes,*** or ***group codes.*** In block codes, each block of k message bits is encoded into a block of n bits (n > k), as shown in fig. 1.1. The check bits are derived from the message bits and are added to them. The n-bit block of a channel encoder output is called a codeword and the codes or coding schemes in which the message bits appear at the beginning of a codeword, are called ***systematic codes***.

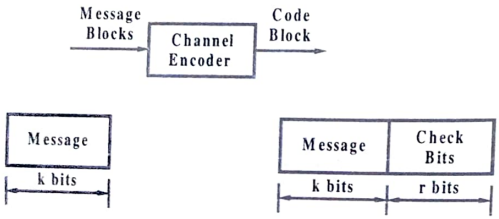


Fig. 1.1

Since there are k bits of information per block, there are 2^k possible distinct messages, out of the 2^n words that may be generated with n bits. This 2^k word set is called a ***block code*** . Every codeword has a code vector from the vector space V_n of all n-tuples. Also, a linear code is defined as a set of 2^k n-tuples, which is a sub-space of the vector space V_n of all n-tuples.

Q.19. Write briefly about error detecting and error correcting codes. ***(R.G.P.V., Dec. 2014)***

Ans. **Error Detecting –** Various methods of error detecting are as follows–

Parity – The simplest technique for detecting errors is that of adding an extra bit, known as the parity bit, to each word being transmitted. There are two types of parity – odd parity and even parity. For odd parity, the parity bit is set to a 0 or a 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an odd number. For even parity, the parity bit is set to a 0 or 1 at the transmitter such that the total number of 1 bits in the word including the parity bit is an even number.

Check-sums – The parity method can detect only a single error within a word. If there are two errors within the same word, then simple parity will not detect them. A type of two-dimensional parity is used to overcome this disadvantage. Every transmitted word is added to the previously sent word and sum retained at the transmitter. The example is given below –

Word A	1101 0011
Word B	0001 1011
Sum	1110 1110

Each successive word is added to the previous sum in the same manner. The sum (called ***check sum***) is sent at the end of the transmission up to that time. The received check can be checked to the transmitted sum by the receiver. If both are the same, then there were no errors detected.

Check sum transmission is used in teleprocessing (TP) systems.

Parity Data Codes – Parity can be contained within each character by choice of codes carefully. Only two 1's are contained in each character in both the systems. The receiver can check this property in each received character. The biquinary code is used in the abacus and 2 out of 5 codes, which is a unweighted code, is used in communication systems.

Error Correcting Codes – Refer the ans. of Q.18.

Q.20. Write the difference between error-detecting code and error-correcting code. ***(R.G.P.V., Nov./Dec. 2007)***

Ans. During the process of data transmission, errors may occur. To detect and correct, we use two types of codes, namely

(i) Error-detecting code (ii) Error-correcting code.

Error-correcting codes are more sophisticated than error detection codes and require more redundancy bits. The number of bits required to correct a multiple-bit or burst error is so high that in most cases it is inefficient to do so. For this reason, most error correction is limited to one, two, or three bit errors.

The Hamming code is a single-bit error correcting method using redundant bits.

Numerical Problems

Prob.19. Convert the following as directed –

(i) $(10111.101)_2 = (\ )_{10}$ (ii) $(10110101)_2 = (\ )_{Gray}$.

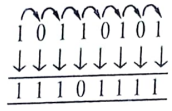
(R.G.P.V., Dec. 2014)

Sol. (i) $(101111.101)_2 = (1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) + (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})$

$= 47 + 0.5 + 0.125 = \mathbf{(47.625)_{10}}$ **Ans.**

(ii) $(10110101)_2 = (\ )_{Gray}$

To convert the given binary number into Gray, the procedure as follows-



$(10110101)_2 = \mathbf{(11101111)_{Gray}}$ **Ans.**

Prob.20. Convert the Gray code 110101 to binary form.

(R.G.P.V., Nov./Dec. 2007, Feb. 2010)

Sol. To convert Gray code into binary form following method is used-

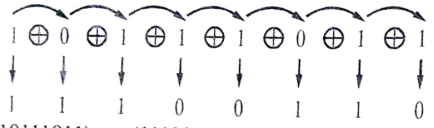
Hence, the binary form of given Gray code is **100110** **Ans**

Prob.21. Convert 10111011 in binary into its equivalent gray code.

(R.G.P.V., June 2015)

Sol. Given, $(10111011)_2 = (\ )_{Gray}$

To convert the given binary number into Gray, the procedure as follows-



Hence, $(10111011)_2 = \mathbf{(11100110)_{Gray}}$ **Ans**

Prob.22. Convert the following decimal numbers into Gray code –

(i) $(87)_{10} \rightarrow (?)$ Gray code (ii) $(1100)_{10} \rightarrow (?)$ Gray code.

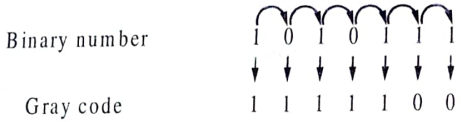
(R.G.P.V., June 2009)

Sol. (i) First convert the given decimal number into its equivalent binary number, then convert this binary number into Gray code.

Successive Division		Remainder
2	87	1 (LSB)
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1 (MSB)
	0	

Thus, $(87)_{10} = (1010111)_2$

In Gray code, a decimal number is coded in binary form in such a way that each Gray code number differs from the preceding and the succeeding number by a single bit.



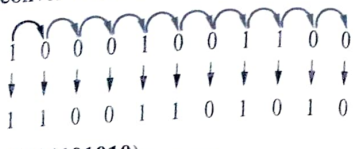
Thus $(87)_{10} = \mathbf{(1111100)_{Gray\ code}}$ **Ans.**

(ii) Similar to case (i) convert the given decimal number into its equivalent binary number, then convert this binary number into Gray code,

Successive Division		Remainder
2	1100	0 (LSB)
2	550	0
2	275	1
2	137	1
2	68	0
2	34	0
2	17	1
2	8	0
2	4	0
2	2	0
2	1	1 (MSB)
	0	

$(1100)_{10} = (10001001100)_2$

The binary to Gray code conversion is as follows –



Thus $(1100)_{10} = (11001101010)_{\text{Gray code}}$ **Ans.**

Prob.23. Express decimal 5280 in excess-3 code. (R.G.P.V., Dec. 2010)

Sol. Given decimal number 5 2 8 0

Add 3 to each bit = 5 +3, 2 +3, 8 +3, 0 +3

Sum → 8 5 11 3

Binary → 1000 0101 1011 0011

Thus,

$(5280)_{10} = (\mathbf{1000\ 0101\ 1011\ 0011})_{\text{EX-3}}$ **Ans.**

Prob.24. Convert the following codes as directed –

(i) $(785.B2)_{16} = (\ )_{10}$ ***(ii)*** $(1011011.1101)_2 = (\ )_8$

(iii) $(110101.101101)_2 = (\ )_{Gray}$ ***(iv)*** $(751.231)_8 = (\ )_{16}$.

(R.G.P.V., Dec. 2012)

Sol. (i) $(785.B2)_{16} = 7 \times 16^2 + 8 \times 16^1 + 5 \times 16^0 + B \times 16^{-1} + 2 \times 16^{-2}$

$= 1792 + 128 + 5 + \frac{11}{16} + \frac{2}{256}$

$= (\mathbf{1925.695313})_{10}$ **Ans.**

(ii) $(1011011.1101)_2 = (\ )_8$

$= \underbrace{001}_{1}\ \underbrace{011}_{3}\ \underbrace{011}_{3}.\underbrace{110}_{6}\ \underbrace{100}_{4} = (\mathbf{133.64})_8$ **Ans.**

(iii) $(110101.101101)_2 = (\ )_{Gray}$

The binary number to gray code formation is obtained as –

$= (\mathbf{101111.011011})_{\text{Gray}}$ **Ans.**

(iv) $(751.231)_8 = \underbrace{0001}_{1}\ \underbrace{1110}_{E}\ \underbrace{1001}_{9}.\underbrace{0100}_{4}\ \underbrace{1100}_{C}\ \underbrace{1000}_{8}$

$= (\mathbf{1E9.4C8})_{16}$ **Ans.**

Prob.25. (i) Convert binary $(10110)_2$ to Gray code.

(ii) The seven bit Hamming code as received is 0010001. Assuming that even parity has been used, check is it correct ? If not find correct code.

(R.G.P.V., June 2014)

Sol. (i) 1 0 1 1 0

↓ ↓ ↓ ↓ ↓

1 1 1 0 1

$(10110)_2 = (\mathbf{11101})_{\text{Gray}}$ **Ans.**

(ii)

d_7	d_6	d_5	r_4	d_3	r_2	r_1
0	0	1	0	0	0	1

Now, r_1 will take care of r_1, d_3, d_5, d_7 whose values are 1010 i.e., two 1's. Even parity is satisfied. Hence $r_1 = 0$.

r_2 will take care of r_2, d_3, d_6, d_7 whose values are 0000 i.e., No 1's, Hence $r_2 = 0$.

r_4 will take care of r_4, d_5, d_6 and d_7, whose values are 0100 i.e. one 1's. Hence $r_4 = 1$.

Hence error word $= (r_4, r_2, r_1) = (1\ 0\ 0) = (4)_{10}$

Hence the bit 4 of the transmission is in error. Since r_4 is received as 0, therefore correct codeword is –

$= \mathbf{0011001}$ **Ans.**

Prob.26. Encode the following binary digits into 7-bit even parity Hamming code –

(i) 0111 ***(ii) 1010.*** ***(R.G.P.V., June 2009)***

Sol. (i) The Hamming bit pattern is illustrated in the form as

D_7	D_6	D_5	P_4	D_3	P_2	P_1
0	1	1		1		

The parity bit P_1 is set to 0, so that it establishes even parity over bits 1, 3, 5 and 7 ($P_1D_3D_5D_7$). The parity bit P_2 is set to 0, so that it establishes even parity over bits 2, 3, 6 and 7 ($P_2D_3D_6D_7$). Parity bit P_4 is set to 0, therefore it establishes even parity over bits 4, 5, 6 and 7($P_4D_5D_6D_7$).

Therefore, the 7-bit even parity Hamming code is **0110100** **Ans.**

(ii) The Hamming bit pattern is given as –

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1		0		

Parity bit P_1 is set to 0, so that it establishes even partiy over the bit 1, 3, 5 and 7 ($P_1D_3D_5D_7$). Parity bit P_2 is set to 1, so that it establishes even parity over the bits 2, 3, 6 and 7 ($P_2D_3D_6D_7$). Parity bit P_4 is set to 0, therefore it establishes

even parity over bits 4, 5,6,7 ($P_4D_5D_6D_7$). Therefore the 7 bit 4, 5, 6, ($P_4D_5D_6D_7$). Therefore, the 7 bit even parity Hamming code is **1010010**. An

LOGIC GATES

Q.21. What are logic gates ? ***(R.G.P.V., June 201***

Ans. The logic gates are electronic circuits because they are made up o a number of electronic devices and components. The logic circuits that perfor the logical operations of AND, OR and NOT are called gates.

The inputs and outputs of logic gates can occur only in two levels, suc as HIGH (or 1) and LOW (or 0). The table that lists all the possible combinatio of input variables and the corresponding outputs is known as truth table. Th logic gates are the building blocks of hardware, which are available in th form of various IC families. Each gate has a distinct logic symbol and i operation can be described by means of an algebraic function. The level logi is defined as a logic in which the voltage levels represent logic-1 and logic-0 Level logic may be positive logic or negative logic.

Q.22. Describe the operation of NOT (inverter) logic gate using truth table and standard logic symbols.

Ans. The inverter circuit performs the operation called complementation The inverter changes one logic level to the opposite logic level. It has one inpu and one output. When a HIGH level is applied to an inverter, a LOW leve appears at its output. The standard logic symbol of NOT gate is shown in fig 1.2(a) and (b), respectively.

(a) *(b)*

Fig. 1.2 Standard Logic Symbols of NOT Gate

The logic equation of NOT gate is written as –

$$Y = \overline{A}$$

This expression read as Y equals complement of A. The truth table of a NOT gate is given in table 1.5.

Table 1.5 Truth Tabl

Input	*Output*
A	*Y = A*
0	1
1	0

The output of NOT gate is always complement of its input. The presence of a small circle called the bubble, always indicates inversion in digital circuits.

Q.23. Explain the operation of AND gate using truth table and standard logic symbol.

Ans. An AND gate performs logical multiplication, which is commonly known as AND function. The output of AND gate is 1 if and only if all the

inputs are 1. If A and B are two input variables of an AND gate and Y is its output, then we have –

$$Y = A.B = AB$$

The standard logic symbol of AND gate is shown in fig. 1.3.

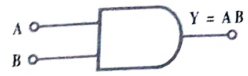


Fig. 1.3 Standard Logic Symbol of Two Input AND Gate

Table 1.6 Truth Table

Inputs		*Output*
A	*B*	*Y = AB*
0	0	0
0	1	0
1	0	0
1	1	1

The truth table of two input AND gate is given in table 1.6. The logical equation Y = AB is read as Y equals A and B, which means that Y will be 1 only when A and B are both 1.

Q.24. Describe the operation of OR gate using truth table and standard logic symbol.

Ans. An OR gate has two or more inputs and performs logical addition. An OR gate produces a 1 on the output when any of the inputs is 1. The output is 0 only when all of the inputs are 0. If A and B are the two input variables of an OR gate and Y is its output, then we get

$$Y = A + B$$

The logical equation Y = A + B is read as Y equals A OR B or A plus B. The standard logic symbol of OR gate is given in fig. 1.4.

The truth table of two input OR gate is given in table 1.7.

Fig. 1.4 Standard Logic Symbol of OR Gate

Table 1.7 Truth Table

Inputs		*Output*
A	*B*	*Y = A + B*
0	0	0
0	1	1
1	0	1
1	1	1

Q.25. How will you construct the following gates from a diode-resistor network –

(i) AND (ii) OR (iii) NOT.

(R.G.P.V., Dec. 2011)

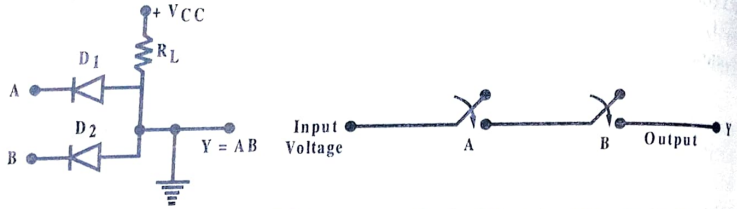
Ans.(i) AND Gate – A 2-inputs AND gate using diodes is shown in fig. 1.5, in which A and B represent the inputs and Y the output.

If A = 0 and B = 0, both the diodes conduct as they are forward biased, and hence the output is Y = 0.

If A = 0 and B = 1, the diode D_1 conducts and D_2 does not conduct, and hence the output is Y = 0.

If A = 1 and B = 0, the diode D_1 does not conduct and D_2 conducts, and hence the output is Y = 0.

If A = 1 and B = 1, both the diodes do not conducts as they are reverse biased, and hence the output is Y = 1.



(a) Circuit Diagram using Diodes *(b) Its Electrical Equivalent*

Fig. 1.5 2-input AND Gate

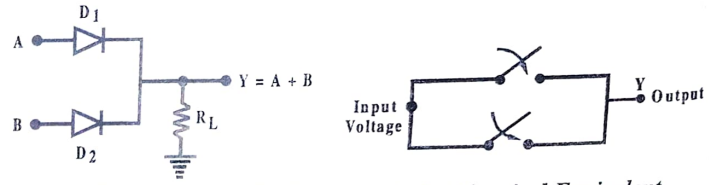
(ii) OR Gate – An OR gate using diodes is shown in fig. 1.6, in which A and B represent the inputs and Y the output. The resistance R_L is the load resistance.

If A = 0 and B = 0 both the diodes will not conduct and hence the output Y = 0.

If A = 1 and B = 0, diode D_1 conducts, then $V_0 \approx 5V$ and so Y = 1

If A = 0 and B = 1, diode D_2 conducts and hence Y = 1

If A = 1 and B = 1, both the diodes conduct and hence Y = 1.



(a) Circuit Diagram using Diodes *(b) Its Electrical Equivalent*

Fig. 1.6 2-input OR Gate

(iii) NOT Gate – A NOT gate using a transistor is shown in fig. 1.7, in which A represents the input and Y represent the output i.e., $Y = \overline{A}$. When the input is HIGH, the transistor is in the ON state and the output $V_C = V_{CE(sat)}$ is LOW. If the input is LOW, the transistor is in the OFF state and the output $V_C = V_{CC}$ is HIGH.

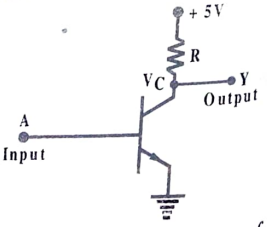


Fig. 1.7 Circuit Diagram of NOT Gate using Transistor

Q.26. Describe the operation of NAND gate with truth table and standard logic symbol.

Ans. The term NAND is a combination of the NOT-AND gates and implies an AND function with a complemented output. It has two or more inputs and only one output. A NAND gate produces a 0 (or LOW) output only when all the inputs are 1 (or HIGH). When any one of input is 0 (or LOW), the output will be 1 (or HIGH). Here we note that this operation is opposite that of the AND in terms of the output level. The logic symbol for the NAND gate is shown in fig. 1.8.

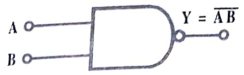


Fig. 1.8 Logic Symbol of NAND Gate

Table 1.8 Truth Table

Inputs		*Output*
A	***B***	***Y = AB***
0	0	1
0	1	1
1	0	1
1	1	0

The truth table of two input NAND gate is given in table 1.8.

The logical operation of a NAND gate is represented as –

$$Y = \overline{AB}$$

Q.27. Discuss NOR gate operation with the help of truth table and standard logic symbol.

Ans. NOR is a combination of NOT-OR gates. A NOR gate produces a 0 (or LOW) output when any of its input is 1 (or HIGH). It produces output 1 (or HIGH) only when all of its inputs are 0 (or LOW). The logical equation of the two input NOR gate is given as –

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

The standard logic symbol of NOR gate is given in fig. 1.9 and its truth table is shown in table 1.9.

Fig. 1.9 Standard Logic Symbol of NOR Gate

Table 1.9 Truth Table

Inputs		*Output*
A	***B***	***Y = A + B***
0	0	1
0	1	0
1	0	0
1	1	0

Q.28. Discuss AND, OR, X-OR and X-NOR gates. (R.G.P.V., June 2011)

Ans. **AND Gates** – Refer the ans. of Q.23.

OR Gates – Refer the ans. of Q.24.

X-OR Gates – The EX-OR gate is recognizes only words that have a odd number of 1's. It has two or more input and one output. The output of two input EX-OR gate assumes a HIGH (or 1) if one and only one inp assumes a HIGH (or 1). Because of their fundamental importance in mar applications, these gates are often used as basic logic elements with their ow unique symbols. The standard logic symbol for an EX-OR gate is given in fig 1.10 and the truth table for its operation is shown in table 1.10.

Fig. 1.10 Standard Logic Symbol of EX-OR Gate

Table 1.10 Truth Table

Inputs		*Output*
A	*B*	$Y = A\bar{B} + \bar{A}B = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

The logical equation of two input EX-OR gate is given as –

$$Y = A\bar{B} + \bar{A}B = A \oplus B$$

This expression is read as Y equals A EX-OR B.

X-NOR Gates – The EX-NOR gate is logically equivalent to an EX-O gate followed by an inverter (or NOT gate). The bubble on the output of t EX-NOR symbol denotes that its output is opposite that of EX-OR gate. A EX-NOR gate has two or more inputs and one output. The output of a tw input EX-NOR gate is HIGH if both the inputs A and B are HIGH or LOW. T output is LOW if the one input is LOW and other input is HIGH.

The standard logic symbol of EX-NOR gate is shown in fig. 1.11 and truth table for operation is given in table 1.11.

Fig. 1.11 Standard Logic Symbol of EX-OR Gate

Table 1.11 Truth Table

Inputs		*Output*
A	*B*	$Y = (\bar{A}\bar{B} + A\bar{B}) = (A \oplus B)$
0	0	1
0	1	0
1	0	0
1	1	1

Q.29. Draw the logic diagram of EX-NOR gate using only NOR gat (R.G.P.V., Dec. 20

Ans. The output of EX-NOR gate is given by

$$Y = A \oplus B = A\bar{B} + \bar{A}B = A\bar{B}.\overline{\bar{A}B} = (A + \bar{B}).(\bar{A} + B) = (A + \bar{B})(\bar{A} + B)$$

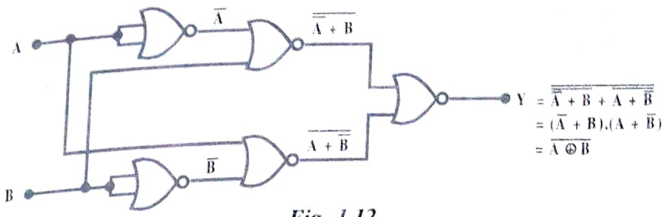


Fig. 1.12

The logic diagram of EX-NOR gate is shown in fig. 1.12.

Q.30. Compare truth table, excitation table and state table.
(R.G.P.V., Dec. 2006, June 2009)

Ans. The comparison between truth table, excitation table and state table are as follows –

S.No.	*Truth Table*	*Excitation Table*	*State Table*
(i)	It gives all possible combinations of input variables and the corresponding output.	It gives information about the excitation or input required to be applied to the memory element.	It is a tabular representation of relationship between the present state, the input, the next state, and the output.
(ii)	It shows how the logic circuit's output responds to various combination of logic level at the inputs.	It shows the information about the output of the machine after the application of the present input.	It shows the output that will be generated and the next state to which the machine will go.

Q.31. What do you understand by universal gate ? Design all logic gates using universal gates. ***(R.G.P.V., June 2009)***

Or

What is universal gate ? ***(R.G.P.V., June 2014)***

Or

What are universal gates ? Explain with example. (R.G.P.V., Dec. 2014)

Or

What are universal gates ? Why are they called so ? (R.G.P.V., Dec. 2015)

Or

Why NAND gate is known as universal gate ? ***(R.G.P.V., Dec. 2016)***

Or

What is universal gate ? Implement AND, OR and NOT gates using NAND gates and NOR gates. ***(R.G.P.V., Dec. 2017)***

Ans. NAND and NOR gates are called universal gates or universal buildi blocks because both can be used to implement any gate like AND, OR a NOT gates or any combination of these basic gates. Fig. 1.13 shows how NOR gate can be used to realize various logic gates while fig. 1.14 shows ho a NAND gate can be used for the same.

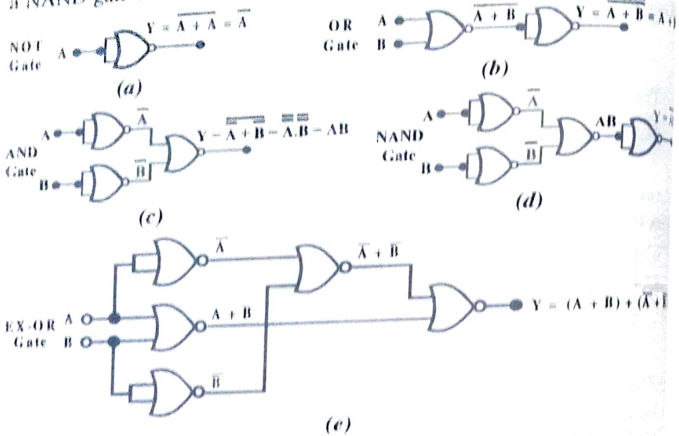
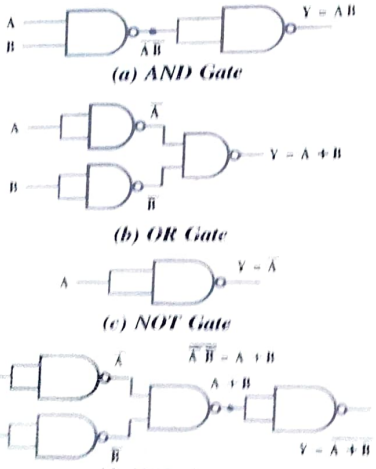


Fig. 1.13 Realization of Various Gates using NOR Gate



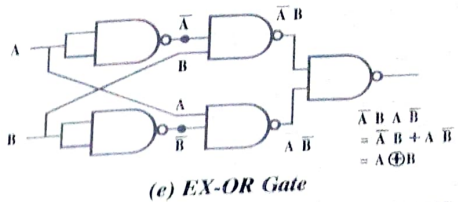


Fig. 1.14 Realization of Various Gates using NAND Gate

Q.32. Realize all logic gates using minimum number of NAND gates. (R.G.P.V., June 2009)

Ans. Refer the ans. of Q.31.

Q.33. Implement EX-OR gate using NOR gates.(R.G.P.V., June 2015)

Ans. Implement EX-OR gate using NOR gates is shown in fig. 1.13 (e).

$$Y = AB + AB$$
$$= AA + AB + AB + BB \qquad [AA = 0]$$
$$= A(A + B) + B(A + B) = (A + B)(A + B)$$
$$= (A + B)(A + B) = (A + B) + (A + B)$$

Numerical Problems

Prob.27. Implement y = AB + CD using only NAND gates. (R.G.P.V., June 2014)

Sol. Given function

$$y = AB + CD$$

The implementation of the given function is shown in fig. 1.15.

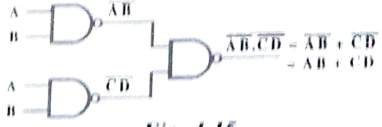


Fig. 1.15

Prob.28. Simplify the following function using NOR-gates –

(i) ABCD + ABD + ABC (ii) ABC + AB + AC. (R.G.P.V., Feb. 2010)

Sol. (i) $ABCD + ABD + ABC$

$$= BD(AC + A) + ABC = BD(A + C) + ABC$$
$$= ABD + BCD + ABC$$

Implementation of above reduced expression using NOR gate is shown in fig. 1.16.

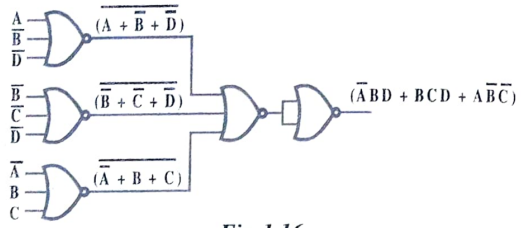


Fig 1.16

(ii) $ABC + AB + AC = AC(B + 1) + AB = AC + AB$

Implementation using NOR gate is shown in fig. 1.17.

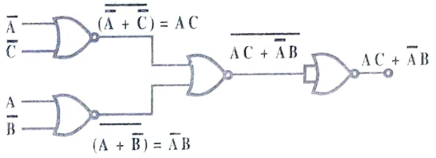


Fig. 1.17

Prob.29. Implement the function $F = A(B + CD) + BC'$ using NOR gate. ***(R.G.P.V., June 201***

Sol. Implementation of given function using NOR gate is shown in fig. 1.

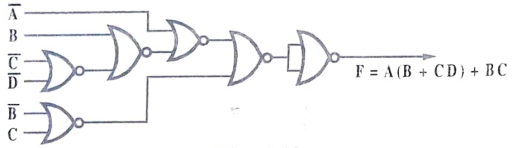


Fig. 1.18

BOOLEAN ALGEBRA, BOOLEAN FUNCTIONS, SIMPLIFICATION OF BOOLEAN FUNCTIONS

Q.34. Write short note on Boolean algebra.

Ans. A number system based on two digits 0 and 1 is known as binar number system. For manipulations of binary variables an english mathematici George Boolean developed laws, known as Boolean algebra. The Boolean algeb differs from both ordinary algebra and the binary number system. For examp in Boolean algebra the addition of two digits 1 + 1 = 1, while in binary arithmeti this results in 10.

Q.35. State duality theorem. ***(R.G.P.V., May/June 2006)***

Ans. The Huntington postulates have been listed in pair and designated by part (a) and part (b). One part may be obtained from the other if binary operator and the identity element are interchanged. This important property of Boolean algebra is called ***duality principle*** or ***duality theorem.*** It states that every algebraic expression deducible from the postulates of Boolean algebra remain valid if the operator and identity element are interchanged. In a two valued Boolean algebra, the identity element and the element of set B are same, i.e., 1 and 0. The duality principle have many applications. If the dual of an algebra expression is desired, we simply change the AND and OR operator and replace 1's by 0's and 0's by 1's.

Q.36. State and prove basic laws of Boolean algebra. ***(R.G.P.V., Dec. 2013)***

Ans. There are fundamental laws in Boolean algebra, which are used to build a workable, cohesive framework upon which are placed the theorems proceeding from these laws. These laws are as follows –

Laws of Complementation – The term complement simply means to invert or to change 1's to 0's and 0's to 1's. The five laws of complementation are given below –

Law 1 — $\bar{0} = 1$
Law 2 — $\bar{1} = 0$
Law 3 — If $A = 0$, then $\bar{A} = 1$
Law 4 — If $A = 1$, then $\bar{A} = 0$
Law 5 — $\bar{\bar{A}} = A$

AND Laws – There are four AND laws as given below –

Law 1 — $A \cdot 0 = 0$
Law 2 — $A \cdot 1 = A$
Law 3 — $A \cdot A = A$
Law 4 — $A \cdot \bar{A} = 0$

OR Laws – The four OR laws are as follows –

Law 1 — $A + 0 = A$
Law 2 — $A + 1 = 1$
Law 3 — $A + A = A$
Law 4 — $A + \bar{A} = 1$

Commutative Laws – The commutative laws allow the change in position of an AND or an OR variable –

Law 1 – $A + B = B + A$
Law 2 – $A \cdot B = B \cdot A$

Associative Laws – The associative laws allow the grouping of variables. There are two associative laws –

Law 1 – $A + (B + C) = (A + B) + C$

Law 2 – $A . (B . C) = (A . B) . C$

Distributive Laws – The distributive laws allow the factoring a multiplying out of expressions. Three distributive laws are as follows –

Law 1 – $A + BC = (A + B) (A + C)$

This law states that the AND operation of several variables and the O operation of the result with a single variable is equivalent to the OR operatio of the single variable with each of the several variables and then the AN operation of the sums.

Law 2 – $A(B + C) = AB + AC$

This law states that the OR operation of several variables and the AN operation of the result with a single variable is equivalent to the AND operatio of the single variable with each of the several variables and then the O operation of the products.

Law 3 – $A + AB + A + B$

This law states that the OR operation of a variable with the AND of th complement of that variable with another variable, is equal to the OR operatio of the two variables.

Absorption Laws – There are two laws of absorption which are a follows –

Law 1 — $A + A . B = A$

The law states that ORing of a variable, A with the AND of that variabl A and another variable, B is equal to that variable itself.

It is expressed algebraically as –

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

Law 2 — $A (A + B) = A$

The law states that ANDing of a variable, A with the OR of variable, and variable, B is equal to that variable itself. It is illustrates algebraically as

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

Q.37. What is Boolean algebra write any three theorems of Boolean algebra ? ***(R.G.P.V., Dec. 201***

Ans. Refer the ans. of Q.34 and Q.36.

Q.38. State the distributive property of Boolean algebra. ***(R.G.P.V., Dec. 201***

Ans. Refer the ans. of Q.36, under the heading Distributive law.

Q.39. What is minterm ? Write all minterm for three variables. ***(R.G.P.V., Nov./Dec. 200***

Ans. A product term containing all the K-variables of the function in either complemented or uncomplemented form is known as a ***minterm***. A 2-variables function has four possible combinations, which are $\bar{A}B$, $A\bar{B}$, $\bar{A}\bar{B}$ and AB. These product terms are referred to as minterms or standard product or fundamental products. There are 8 minterms for a 3-binary input variables function, as shown in table 1.12. Each minterm can be achieved by the AND operation of all the variables of the function. In the minterm, a variable appears either in uncomplemented form, if it has a value of 1 in the corresponding combination, or in complemented form. if it possesses the value of 0. The minterms of a 3-variable function can be denoted by m_0, m_1, m_2, m_3, m_4, m_5, m_6 and m_7.

Table 1.12 Minterm Table

Inputs			*Minterm*	*Designation*
A	*B*	*C*		
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0
0	0	1	$\bar{A}\bar{B}C$	m_1
0	1	0	$\bar{A}B\bar{C}$	m_2
0	1	1	$\bar{A}BC$	m_3
1	0	0	$A\bar{B}\bar{C}$	m_4
1	0	1	$A\bar{B}C$	m_5
1	1	0	$AB\bar{C}$	m_6
1	1	1	ABC	m_7

An important property of a minterm is that it contains the value 1 for only one combination of K input variables; i.e., for a K-variable function of the 2^K minterms, only one minterm will possess the value 1, while the remaining $2^K - 1$ minterms will have the value 0 for an arbitrary input combination.

Q.40. Write short note on maxterm.

Ans. A sum term containing all the K-variables of the function in either complemented or uncomplemented form is referred to as a ***maxterm***. A two-variable function has four possible combinations which are $A + B$, $A + \bar{B}$, $\bar{A} + B$ and $\bar{A} + \bar{B}$. These sums are known as ***maxterms.*** Therefore, a 3-binary input variable function has 8 maxterms as given in table 1.13. Each maxterm can be achieved by the OR operation of all the variables of the function. In a maxterm, a variable appears either in uncomplemented form if it contains the value 0 in the corresponding combination or in complemented form if it possesses the value 1. In a 3-variable function, the maxterms can be represented by M_0, M_1, M_2, M_3, M_4, M_5, M_6 and M_7.

Table 1.13 Maxterm Table

Inputs			Maxterm	Designation
A	B	C		
0	0	0	A + B + C	M_0
0	0	1	A + B + $\overline{C}$	M_1
0	1	0	A + $\overline{B}$ + C	M_2
0	1	1	A + $\overline{B}$ + $\overline{C}$	M_3
1	0	0	$\overline{A}$ + B + C	M_4
1	0	1	$\overline{A}$ + B + $\overline{C}$	M_5
1	1	0	$\overline{A}$ + $\overline{B}$ + C	M_6
1	1	1	$\overline{A}$ + $\overline{B}$ + $\overline{C}$	M_7

Important property of the maxterm is that it contains the value '0' for only one combination of K input variables, i.e., K-variable function of the 2^K maxterms, only one maxterm will possess the value 0, while all remaining $2^K - 1$ maxterms will have the value 1 for an arbitrary input combination.

Q.41. State and prove De-Morgan's theorem.

[R.G.P.V., Dec. 2002(CS), June 2010]

Ans. The De-Morgan's theorem represents two of the most powerful laws in Boolean algebra. De-Morgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted.

(i) Law 1 — $(A + B) = \overline{A}.\overline{B}$

The tables 1.14 (a) and (b) illustrate the truth table of NOR and bubbled AND gates, respectively. This law states that the complement of a sum of variables is equal to the product of their individual complements. It means that the complement of two or more variables ORed together, which is the same as the AND of the complements of each of the individual variables. Each side of law-1 can be represented schematically shown in figs. 1.19 (a) and (b), respectively.

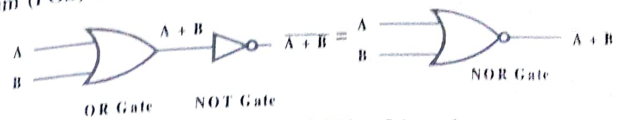
Table 1.14 (a) Truth Table of NOR Gate

Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

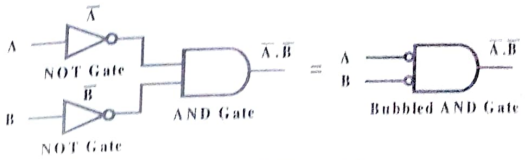
Table 1.14 (b) Truth Table of Bubbled AND Gate

Inputs		Output
A	B	$\overline{A}.\overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

It is observed that the NOR gate is equivalent to a bubbled AND gate. This law can also be extended to any number of variables. Thus, it may also be observed that this law allows removal of individual variables from under a NOT sign and transformation from a sum of product (SOP) form to a product of sum (POS) form.



(a) Left Hand Side of Law-1



(b) Right Hand Side of Law-1

Fig. 1.19

(ii) Law 2 — $\overline{AB} = \overline{A} + \overline{B}$

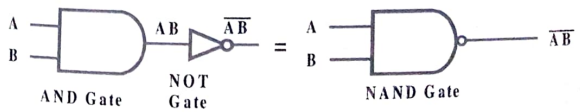
The truth table for NAND and bubbled OR gates are given in tables 1.15 (a) and (b), respectively. This law states that the complement of each of the product of variables is equal to the sum of their individual complements. Each side of this law can be represented schematically in figs. 1.20 (a) and (b), respectively.

Table 1.15 (a) Truth Table of NAND Gate

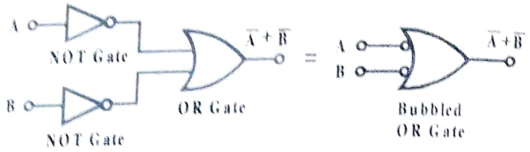
Inputs			Output
A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Table 1.15 (b) Truth Table of Bubbled OR Gate

Inputs				Output
A	B	$\overline{A}$	$\overline{B}$	$\overline{A} + \overline{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0



(a) Left Hand Side of Law-2



(b) Right Hand Side of Law-2

Fig. 1.20

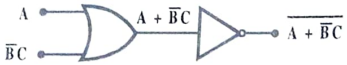
It is observed that the NAND gate is equivalent to a bubbled OR gate. This law can be extended to any number of variables. This law also permits removal of individual variables from under a NOT sign and transformation from a product of sum (POS) form to a sum of product (SOP) form.

Q.42. Define Boolean expressions and logic diagrams.

Ans. Boolean algebra is useless, unless it can be translated into hardware, in the form of AND gates, OR gates, and inverters. Similarly, Boolean algebra can be useful technique for analyzing existing circuits only if the hardware can be translated into a Boolean expression.

Algebra to Logic – The easiest way to convert an expression into a logic diagram is to start with the output and work toward the input. The expression $A + BC$ is to implemented in logic. As shown in fig. 1.21, start with the final expression. Since this is basically a noted function it must be the inverse of $A + BC$. Note that the circle is on the output, indicating when $A + BC$ is high, $A + BC$ is low. The circle should be drawn to indicate the location NOTed function.

Step 1. Output Inversion



Step 2. OR Function

Step 3. AND Function

Fig. 1.21 Conversion of the Boolean to Logic Circuit

Next consider the function on the input of the inverter, $A + BC$. This is an OR of the term A and $\overline{B}C$ so draw an OR gate with two inputs. One input is term A, and other is term $\overline{B}C$.

The BC term is an AND function of B and C. Therefore, draw an AND gate with two inputs. Label the inputs B and C. The $\overline{B}$ term can be left as shown, or another inverter can be shown with a B input.

Many input to logic system are similar to the term $\overline{B}$ and enter the system as a 0 V signal when an event occurs. In this case, a B signal never exists in hardware.

Consider fig. 1.22, and assume the switch is ON when it is closed and off when open. Then signal is true when the switch output is a ground signal. If a + 5V signal were required to indicate the switch is ON, the B signal would have to feed an inverter whose output would be B. Thus, the NOT over the B indicates two things –

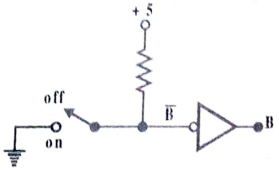


Fig. 1.22 Switching to the Ground Terminal

(i) The signal is true (high or 1) when a ground appears on the wire.

(ii) The signal must be inverted to obtain a high-level true signal.

Q.43. Write down the important rules for reduction the Boolean expressions.

Ans. Rules for reduction the Boolean expression are given below –

(i) First select the term that has the fewest number of adjacencies.

(ii) Expand this expression to include the largest number of square possible in a 2^n configuration. Squares may be covered several times if necessary for this expansion.

(iii) Continue to select uncovered terms that have the fewest number of adjacencies.

(iv) All terms must be covered in the final expression.

(v) Finally, read the map. A two-square will eliminate one variable, a four-square two variable, an eight-square three variables, and a 2^n square, n variables.

Q.44. How the Boolean expressions are reduced ?

Ans. As every element of hardware is represented by a logical operator, Boolean equation must be reduced to as simple a form as possible in order to reduce cost. The techniques, which are used in ordinary algebra, are also used here for these reductions. The following procedure is the general approach –

(i) Multiply all variables necessary to remove parantheses.

(ii) Look for identical terms. Only one of those terms be retained and all others dropped. For example,

$$AB + AB = AB$$

(iii) Look for a variable and its negation in the same term. This term can be reduced. For example,

$$A\bar{A}C = 0.C \qquad [\because A.\bar{A} = 0]$$
$$= 0$$

(iv) From the pairs of terms which are identical except for one variable, the larger term can be dropped. For example,

$$ABCD + ABD = ABD\,(C + 1)$$
$$= ABD.1$$
$$= ABD$$

If one variable is present in first term and its negation in the second term this can be reduced. For example,

$$ABCD + A\,B\,CD = (B + \bar{B})\,ACD$$
$$= 1.ACD$$
$$= ACD$$

In the most of the cases, this procedure can be used.

Numerical Problems

Prob.30. Reduce the following expression using Boolean algebra –

(i) $\overline{A\bar{B} + ABC} + A(B + A\bar{B})$ ***(ii)*** $AB + AB + AB + A\,B$.

(R.G.P.V., Feb. 2010

Sol. (i) $AB + ABC + A(B + AB)$

$= (AB.ABC) + A(B + A)(B + B)$ $[\because A + BC = (A+B)(A+C)]$

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

$= A\,A + A\,B + A\,C + BA + BB + BC + AB + A$

$= A + A\,B + A\,C + BA + BC + AB + A$ $[\because BB = B]$

$= (A + A) + A(B + B) + A\,C + BC + AB$

$= (1 + A) + A\,C + BC + AB$

$= (1 + A\,C) + BC + AB$ $[\because 1 + A = 1]$

$= (1 + BC) + AB$

$= 1 + AB = 1$ **Ans.**

(ii) $AB + AB + AB + A\,B$

$= B(A + A) + B(A + A)$

$= B + B = 1$ **Ans.**

Prob.31. Minimize the following function using Boolean algebra –

$f_1 = ABCD + BC + AD + C(A + BDC)$

$f_2 = (BCD + A)\,B + (AC + BC)\,D$.

(R.G.P.V., Dec. 2012

Sol. $f_1 = ABCD + BC + AD + C(A + BDC)$

$= ABCD + (BC)(AD) + C(A + B(D + C))$

$= ABCD + (B + C)(A + D) + CA + BCD + BCC$

$= ABCD + A\,B + B\,D + A\,C + C\,D + AC + BCD$ $(\because C\bar{C} = 0)$

$= A\,B + CD + B\,D + A\,C + C\,D + AC + BCD$

$= A\,B + B\,D + A\,C + C(D + D) + AC + BCD$

$= A\,B + B\,D + A\,C + C.1 + AC + BCD$

$= A\,B + B\,D + C(A + 1) + AC + BCD$

$= A\,B + B\,D + C.1 + AC + BCD$

$= A\,B + B\,D + C + A + BCD$

$= A + A\,B + B\,D + C + BCD$

$= A + B + B\,D + C + BD$

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

$= \mathbf{A + B + C + D}$ **Ans.**

$f_2 = (BCD + A)B + (AC + BC)\,D$

$= B.BCD + AB + ((A + C) + (B + C))\,D$

$= 0 + AB + (A + C)\,(B + C)\,D$ $[\because BB = 0]$

$= AB + A.C\,B\,C\,D$

$= AB + ABCD = A(B + BCD)$

$= A(B + CD) = \mathbf{AB + ACD}$ **Ans.**

Prob.32. Minimize using Boolean algebra –

(i) $ABC + BD + ABD + ABCD$

(ii) $A + BA + ABC + AC + ABC$.

(R.G.P.V., June 2012)

Sol. (i) $ABC + BD + ABD + ABCD$

$= ABC(1 + D) + BD + ABD$

$= ABC + BD + ABD$ $(\because 1 + D = 1)$

$= ABC + BD + A + B + D$

$= A + B + D + BD + ABC$ $(\because A + AB = A + B)$

$= A + B + D + D + ABC$

$= A + B + 1 + ABC$

$= A + 1 + ABC$

$= 1 + ABC = 1$ **Ans.**

(ii) $A + \bar{B}A + ABC + \bar{A}C + A\bar{B}C$

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

$= A + B + ABC + \bar{A} + C + A\bar{B}C$

$= A + \bar{A} + B + ABC + C + A\bar{B}C$

$= 1 + B + ABC + C + A\bar{B}C$

$= 1 + C + ABC + A\bar{B}C$

$= 1 + ABC + A\bar{B}C$

$= 1 + A\bar{B}C = 1$ Ans

Prob.33. Show that $(A + C)(A + D)(B + C)(B + D) = AB + CD$.
(R.G.P.V., Dec. 2016)

Sol. Taking L.H.S.

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

$= (AA + AD + AC + CD)(BB + BD + BC + CD)$

$= AB + ABD + ABC + ACD + ABD + ABD + ABCD + ACD$
$+ ABC + ABCD + ABC + ACD + BCD + BCD + BCD + CD$

$= AB + ABD + ABC + ACD + ABCD + BCD + CD$

$= AB(1 + D) + ABC + ACD(1 + B) + CD(B + 1)$

$= AB + ABC + ACD + CD$

$= AB(1 + C) + CD(A + 1)$

$= AB + CD$

L.H.S. = R.H.S. **Hence Proved**

Prob.34. Simplify the following Boolean function to minimum number of literals –

(i) $zx + zx'y$ ***(ii)*** $xy + xy'$ ***(iii)*** $y(wz' + wz) + xy$
(R.G.P.V., June 2017)

Sol. (i) $zx + zx'y = z(x + x'y)$ $[A + A'B = A + B]$

$= z(x + y)$ Ans

(ii) $xy + xy' = x(y + y')$ $[A + A' = 1]$

$= x.1$

$= x$ Ans

(iii) $y(wz' + wz) + xy = y[w(z + z')] + xy$

$= y[w.1] + xy$

$= yw + xy$

$= y(x + w)$ Ans

Prob.35. Solve the following expressions –

(i) $xy + \bar{x}z + x\bar{y}z(xy + z)$ ***(ii)*** $x.[y + z.\overline{(x.y + x.z)}]$.
(R.G.P.V., June 2011)

Sol. (i) $xy + \bar{x}z + x\bar{y}z(xy + z) = xy + \bar{x}z + xyz.xy + xyz.z$

$= xy + \bar{x}z + x\bar{y}z$ $[\because z.z = z \text{ and } y.y = 0]$

$= xy + \bar{x} + z + x\bar{y}z$

$= \bar{x} + xy + z + \bar{y}z$ $[\because x + xy = x + y]$

$= \bar{x} + xy + z + z\bar{y}$

$= \bar{x} + y + z + \bar{y}$

$= \bar{x} + z + 1$ $[y + \bar{y} = 1]$

$= x + 1$

$= 1$ **Ans.**

(ii) $x.[y + z.\overline{(x.y + x.z)}] = x[y + z\overline{(x.y)}.\overline{(x.z)}]$

$= [xy + xz.\overline{(x.y)}.\overline{(x.z)}]$

$= xy$ $[\because a.\bar{a} = 0]$

$= xy$ **Ans.**

Prob.36. Obtain the truth table of the function –
$F = xy + xy' + y'z$ ***(R.G.P.V., Dec. 2016)***

Sol. The truth table of the given function is shown in table 1.16.

Table 1.16

Input Variables							*Output*
x	y	z	y'	xy	xy'	$y'z$	$F = xy + xy' + y'z$
0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	1
0	1	0	0	0	0	0	0
0	1	1	0	0	0	0	0
1	0	0	1	0	1	0	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	1
1	1	1	0	1	0	0	1

Prob.37. Write the minterm of $ACD + AB$ and implement it.
(R.G.P.V., June 2014)

Sol. $y = ACD + AB$

$= ACD(B + \bar{B}) + AB(C + \bar{C})(D + \bar{D})$

$= ABCD + A\bar{B}CD + (ABC + AB\bar{C})(D + \bar{D})$

$= ABCD + A\bar{B}CD + ABCD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D}$

$= ABCD + A\bar{B}CD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D}$

The implementation of above equation is shown in fig. 1.23.

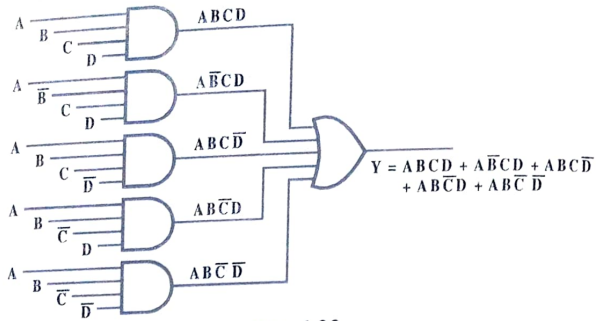


Fig. 1.23

Prob.38. Express the following in sum of minterms and product of maxterm form –

(i) $f(A, B, C) = (A + B)(BC)$

(ii) $f(x, y, z) = 1$. ***(R.G.P.V., June 2006)***

***Sol.* (i) (a) Sum of Minterm –**

$$f(A, B, C) = (A + B)(BC)$$
$$= ABC + BBC \quad \{BB = 0\}$$
$$= ABC \quad \{0.C = 0\}$$
$$f(A, B, C) = m_1$$
$$f(A, B, C) = \Sigma m(1)$$

(b) Product of Maxterm –

The expression is a three-variable function. The variable C is missing in the first term, so add CC to it. Therefore,

$$(A + B) = (A + B + CC) = (A + B + C)(A + B + C)$$

Similarly

$$B = B + AA + CC = (A + B + CC)(A + B + CC)$$
$$= (A + B + C)(A + B + C)(A + B + C)(A + B + C)$$
$$C = C + AA + BB = (A + C + BB)(A + C + BB)$$
$$= (A + B + C)(A + B + C)(A + B + C)(A + B + C)$$

Now,

$$(A + B)BC = (A + B + C)(A + B + C)(A + B + C)(A + B + C)$$
$$(A + B + C)(A + B + C)(A + B + C)$$
$$f(A, B, C) = M_7, M_6, M_5, M_4, M_3, M_2, M_0$$
$$= \Pi M (0, 2, 3, 4, 5, 6, 7)$$

***(ii)* $f(x, y, z) = 1$**

(a) Minterm – The expression is a three variable function. Function value is equal to 1 it means that all ($2^n = 8$) squares are adjacent and contain 1. Therefore minterm will be,

$$f(x, y, z) = x\bar{y}\bar{z} + xyz + xy\bar{z} + xyz + x\bar{y}z + xyz + x\bar{y}\bar{z} + xyz$$
$$= m_0 + m_1 + m_2 + m_3 + m_4 + m_5 + m_6 + m_7$$
$$= \Sigma m (0, 1, 2, 3, 4, 5, 6, 7)$$

(b) Maxterm – Taking the complement of minterms, we obtain the maxterm.

$$\text{Minterm} = x\bar{y}\bar{z} + xy\bar{z} + xyz + x\bar{y}z + x\bar{y}\bar{z} + xyz + x\bar{y}\bar{z} + xyz$$
$$\text{Maxterm} = (x + y + z)(x + y + z)(x + y + z)(x + y + z)$$
$$(x + y + z)(x + y + z)(x + y + z)(x + y + z)$$
$$= M_0 M_1 M_2 M_3 M_4 M_5 M_6 M_7$$
$$= \Pi M (0, 1, 2, 3, 4, 5, 6, 7)$$

Prob.39. Prove the following using De-Morgan's theorem –

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

and hence prove that an OR-AND configuration is equivalent to NOR-NOR configuration. Realize the logic equation given above using –

(i) OR and AND gates (ii) Only NOR gates.

(R.G.P.V., June 2008)

Sol. $(A + B)(C + D) = (A + B).(C + D)$

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

The left hand side is realizable by using two 2-input OR gates followed by a 2-input AND gate, while the right hand side is realizable by two 2-input NOR gates followed by another 2-input NOR gate. Hence, an OR-AND configuration is equivalent to a NOR-NOR configuration.

The realization equation when OR and AND gates.

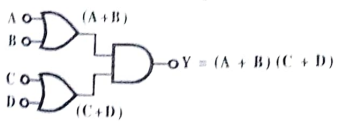


Fig. 1.24

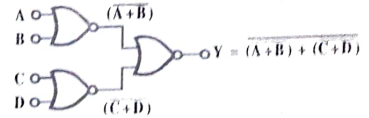


Fig. 1.25

The realization equation when only NOR gates.

Prob.40. Prove the following using De Morgan's theorem

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

and hence prove that an OR-AND configuration is equivalent to NOR-NOR configuration. Realize the logic equation given above using only NOR gates.

(R.G.P.V., June 2009)

Sol. Refer the sol. of Prob.39.

KARNAUGH MAP METHODS, SOP - POS SIMPLIFICATION, NAND - NOR IMPLEMENTATION

Q.45. Write short note on Karnaugh's map methods.

(R.G.P.V., Nov./Dec. 2007)

Or

What is Karnaugh Map ? ***(R.G.P.V., Dec. 2016)***

Ans. The Karnaugh's map technique provides a systematic method for simplifying and manipulating switching expressions.

In this technique, the information contained in a truth table or available in the POS or SOP form is represented on the Karnaugh map or K-map.

The K-map actually modified form of a truth table. Here, the combinations are conveniently arranged to aid the simplification process by applying the rule $Ax + Ax' = A$. In an n-variable K-map, there are 2^n cells.

Each cell corresponds to one combination of n-variables. Therefore, for each row of the truth table, i.e., for each minterm and for each maxterm, there is one specific cell in the K-map.

A two variable K-map has $2^2 = 4$ squares. These squares are called cells. Each square on the K-map represents a unique minterm. The minterm designations of the squares are shown in fig. 1.26. A 1 placed in any square indicates that the corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding minterm does not appear in the expression for output.

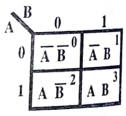


Fig. 1.26 Two-variable K-map

Q.46. Describe don't care condition with the aid of example.

Ans. Some logic circuits can be designed, so, there are certain input conditions for which there are no specified output levels, usually because these input conditions will never occur. In other words, there will be certain combinations of input levels where we "don't care" whether the output is high or low.

The K-maps are simplified using either 1's or 0's. Therefore, we make the entries in the K-map for either 1's or 0's. The cells, which do not contain 1 are assumed to contain 0 and vice-versa. This is not always true since there are cases in which certain combinations of input variables do not occur. Also, for some functions the outputs corresponding to certain combinations of input variables do not matter. In such situations the designer has a flexibility and it is left to him whether to assume a 0 or a 1 as output for each of these combinations. This condition is known as don't-care condition and can be represented on the K-map as a cross mark (×) in the corresponding cell. The cross mark (×) in a cell may be assumed to be a 1 or a 0 depending upon which one leads to a simpler expression. The function can be specified in one of the following manners –

(i) For example, consider the case of minterms with don't-care conditions as follows –

$$f(A, B, C, D) = \Sigma m(1, 3, 7, 11, 15) + d(0, 2, 5) \quad ...(i)$$

The K-map of equation (i) is shown in fig. 1.27.

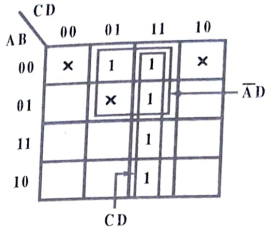


Fig. 1.27 K-map and Minimized Expression of Equation (i)

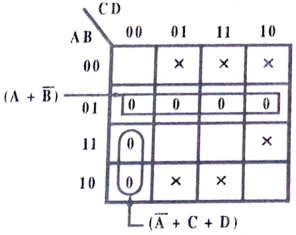


Fig. 1.28 K-map and Minimized Expression of Equation (ii)

(ii) The simplified expression of the above K-map is as follows of maxterms with don't-care conditions as follows –

$$f(A, B, C, D) = \Pi M(4, 5, 6, 7, 8, 12) + d(1, 2, 3, 9, 11, 14) \quad ...(ii)$$

The K-map of equation (ii) is shown in fig. 1.28.

Q.47. What do you understand by prime implicant ? Explain with the aid of an example.

Or

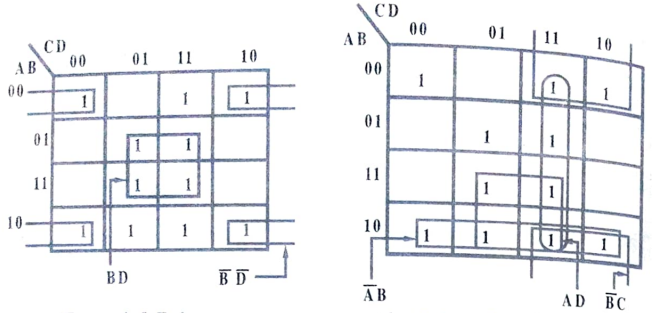
Write short note on the concept of prime implicant.

(R.G.P.V., June 2012)

Ans. A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map. When a minterm in a square is covered by only one prime implicant, then that implicant is said to be extremely important. The prime implicants of a function can be obtained from the map by combining all possible maximum numbers of squares. A single 1 on a K-map represents a prime implicant if it is not adjacent to any other 1's. Two adjacent 1's form a prime implicant provided that they are not within a group of four adjacent squares. Four adjacent 1's form a prime implicant if they are not within a group of eight adjacent squares and so on.

For example, consider the following four-variable Boolean functions as –

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$



(a) Essential Prime Implicants B.D and $\overline{B}.\overline{D}$

(b) Prime Implicants CD, $B\overline{C}$, AD and $A\overline{B}$

Fig. 1.29

The minterms of the function are marked with 1's in the K-maps of fig. 1.29. Two essential prime implicants are shown in fig. 1.29 (a). One term is essential because there is one way to include minterms m_0 within four adjacent squares. These four squares define the term $\overline{B}.\overline{D}$.

In a similar manner, there is only one way that minterms m_5, m_7, m_{13} and m_{15} forms a square and gives the term B.D. The two essential prime implicants cover eight minterms. Fig. 1.29 (b) shows all possible ways that minterms m_3, m_9 and m_{11} can be covered with prime implicants. The minterm m_3 can be covered with either prime implicant CD or $B\overline{C}$. Minterm m_9 can be covered with either AD or $A\overline{B}$. Minterm m_{11} is covered with any one of the four prime implicants.

The simplified expression is obtained from the logical sum of the two essential prime implicants and any two prime implicants that cover minterms m_3, m_9 and m_{11}. There are four possible ways that the function can be expressed with four product terms of two literals each, which is given as –

$$F = BD + B\overline{D} + CD + AD$$
$$= BD + B\overline{D} + CD + A\overline{B}$$
$$= BD + B\overline{D} + B\overline{C} + AD$$
$$= BD + B\overline{D} + B\overline{C} + A\overline{B}$$

The identification of the prime implicants in the K-map helps in determining the alternatives that are available for obtaining a simplified expression.

Q.48. Define the followings –
(i) SOP (ii) POS.

Ans. (i) Sum of Products (SOP) – The logical sum of two or more logical product terms, is known as a sum of products (SOP). It is basically an

OR operation of AND operated variables such as –

$$Y = AB + B\overline{C} + AC$$

(ii) Product of Sums (POS) – A product of sums expression is a logical product of two or more logical sum terms. It is basically an AND operation of OR operated variables such as –

$$Y = (A + B)(B + C)(A + C)$$

Q.49. How do you convert an SOP form to a POS form and vice versa?

Ans. (i) Canonical (Sum of Products) SOP Form – In the given expression, the variables B and C are missing in the first term. The variable A is missing in the second term. Hence first term has to be multiplied by $(B + \overline{B})$ and $(C + \overline{C})$, the second term by $(A + \overline{A})$.

$$Y = A + \overline{B}C$$
$$= A(B + \overline{B})(C + \overline{C}) + BC(A + \overline{A})$$
$$= A(BC + B\overline{C} + \overline{B}C + \overline{B}\,\overline{C}) + ABC + \overline{A}BC$$
$$= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + ABC + \overline{A}BC$$
$$= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}BC$$

(ii) Canonical Product of Sum (POS) Form –

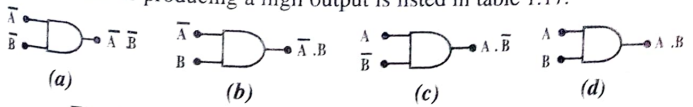
$$Y = A + BC$$
$$= (A + B)(A + C)$$
$$[\because (A + BC) = (A + B)(A + C)]$$

In this expression, the variable C is missing in the first term. The variable B is missing in the second term. Hence, the first term has to be added with $C\overline{C}$, the second term with $B\overline{B}$.

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

Q.50. Write the sum-of products (SOP) method with the help of truth table and logic circuit.

Ans. There are four possible way to AND two input signals which are in complemented and uncomplemented form, shown in fig. 1.30. These outputs are known as fundamental products. Each fundamental product next to the input conditions producing a high output is listed in table 1.17.



(a) *(b)* *(c)* *(d)*

Fig. 1.30 ANDing Two Variables and Their Complements

Table 1.17 Fundamental Products for Two Inputs

A	B	*Fundamental Product*
0	0	$\bar{A}\,\bar{B}$
0	1	$\bar{A}\,B$
1	0	$A\,\bar{B}$
1	1	$A\,B$

The fundamental products idea applies to three or more input variables. Let us take three input variables; A, B, C and their complements. To AND these three input variables and their complements, there are eight possible ways. It results in fundamental products of

$$ABC, AB\bar{C}, A\bar{B}C, A\bar{B}\bar{C}, \bar{A}BC, \bar{A}B\bar{C}, \bar{A}\bar{B}C, \bar{A}\bar{B}\bar{C}$$

Fig. 1.31 (a) shows the last fundamental product, fig. 1.31 (b) the second last and Fig. 1.31 (c) the third from the last.

Table 1.18 summarizes the fundamental products, listing each one next to the input condition which results in a high output. For example, if A = 1, B = 0 and C = 0, the fundamental product obtained in an output form of

$$y = A\,B\,\bar{C} = 1\,0\,0 = 1$$

Table 1.18 Fundamental Products for Three Inputs

A	*B*	*C*	*Fundamental Product*
0	0	0	$\bar{A}\,\bar{B}\,\bar{C}$
0	0	1	$\bar{A}\,\bar{B}\,C$
0	1	0	$\bar{A}\,B\,\bar{C}$
0	1	1	$\bar{A}\,B\,C$
1	0	0	$A\,\bar{B}\,\bar{C}$
1	0	1	$A\,\bar{B}\,C$
1	1	0	$A\,B\,\bar{C}$
1	1	1	$A\,B\,C$

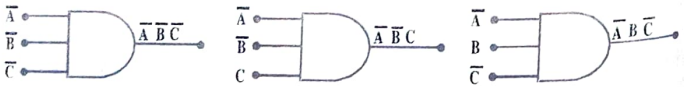


Fig. 1.31 Examples of ANDing Three Variables and Their Complements

Sum of Products Equation – To get the sum of products solution, a truth table is given like table 1.19. Fundamental product is written for each output 1 in the truth table. For example, the first output 1 appears for an input of A = 0, B = 1 and C = 1. Then the fundamental product will be $\bar{A}\,B\,C$. For A = 1, B = 0 and C = 1, the next output 1 appears. $A\,\bar{B}\,C$ is the fundamental product for this.

Logic Circuit – After getting sum-of-products equation, logic circuit can be drawn by an AND-OR network, or a NAND-NAND network. The AND-OR circuit is the one solution of the problem. The AND-OR circuit of fig. 1.32 has the truth table given by table 1.19.

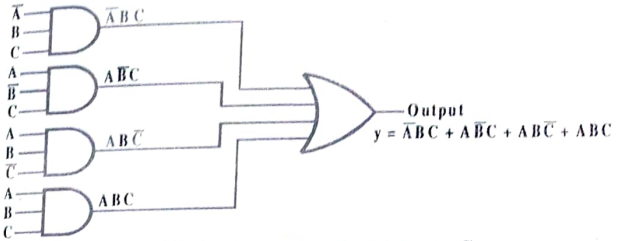


Fig. 1.32 Design of AND-OR Logic Gates

Table 1.19 Design Truth Table

A	*B*	*C*	*Output*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	$1 \Rightarrow \bar{A}BC$
1	0	0	0
1	0	1	$1 \Rightarrow A\bar{B}C$
1	1	0	$1 \Rightarrow AB\bar{C}$
1	1	1	$1 \Rightarrow ABC$

Q.51. Explain the product-of sum (POS) method with the help of truth table and logic circuit.

Ans. In the product-of-sums method, a truth table is given and the fundamental sums are required for a logic design. By ANDing these sums, the product-of-sums equation is obtained corresponding to the truth table. There are few differences between sum-of-product and product-of-sum approaches. In sum-of-products method, the fundamental product produces an output 1 for the corresponding input condition. On the other hand in the product-of-sums method, the fundamental sum produces an output 0 for the corresponding input condition.

Converting a Truth Table to an Equation – Suppose, a truth table is given in table. 1.20. To get the product-of-sums equation, we locate each output 0 in the truth table and write down its fundamental sum. The first output 0 appears for A = 0, B = 0 and C = 0 in table 1.20. The fundamental

sum for these inputs is A + B + C and this produces an output zero for th corresponding input condition,

Output = A + B + C = 0 + 0 + 0 = 0

Table 1.20

A	B	C	Y
0	0	0	0 ⇒ A + B + C
0	0	1	0 ⇒ A + B + C
0	1	0	0 ⇒ A + B + C
0	1	1	1
1	0	0	0 ⇒ A + B + C
1	0	1	1
1	1	0	1
1	1	1	1

The second output 0 appears for the input condition of A = 0, B = 0 an C = 1. A + B + $\overline{C}$ is the fundamental sum for this. To get a logical sum of 0 f the given input conditions, C is complemented

Output = A + B + $\overline{C}$ = 0 + 0 + $\overline{1}$

= 0 + 0 + 0 = 0

The third output 0 occurs for A = 0, B = 1 and C = 0. Its fundamen sum is A + $\overline{B}$ + C

Output = A + $\overline{B}$ + C = 0 + $\overline{1}$ + 0 = 0 + 0 + 0 = 0

Similarly, the fourth output 0 appears for the input condition of A = B = 0, C = 0. Hence its fundamental same is $\overline{A}$ + B + C

Output = $\overline{A}$ + B + C = $\overline{1}$ + 0 + 0 = 0 + 0 + 0 = 0

Table 1.20 shows all the fundamental sums required to implement t truth table. Notice that when input variable is 1, then corresponding variable complemented and when input variable is 0, the corresponding variable uncomplemented. By ANDing the fundamental sums, product-of-sums equati is obtained

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

This is the product-of-sums equation for table 1.20.

Logic Circuit – After getting a product-of-sums equation, the logic circ can be got by drawing an OR-AND network, or a NOR-NOR network. equation (i) each sum shows the output of a 3-input OR gate and the logical is the output of a 4-input AND gate. Hence, a circuit can be drawn as sho in fig. 1.33.

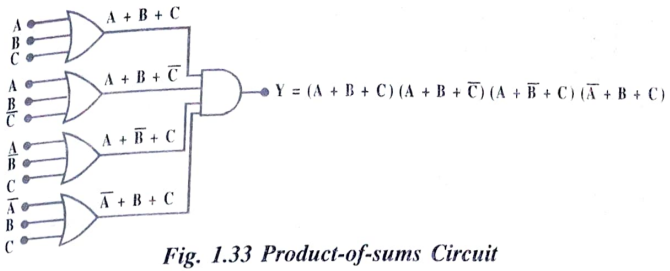


Fig. 1.33 Product-of-sums Circuit

Q.52. How can be converted to NOR-NOR and NAND-NAND gate networks ?

Ans. According to De-Morgan's theorem,

$$\overline{AB} = \overline{A} + \overline{B}$$

and

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

The above expressions may be written as

$$AB = \overline{\overline{A} + \overline{B}} \quad \text{...(i)}$$

and

$$A+B = \overline{\overline{A}\,\overline{B}} \quad \text{...(ii)}$$

Hence, an AND gate is equivalent to a NOR gate with bubbles at its inputs and an OR gate is equivalent to a NAND gate with bubbles at its inputs. Also, a NOR gate is equivalent to an AND gate with bubbles at its inputs and a NAND gate is equivalent to an OR gate with bubbles at its inputs, as discussed in Q.27 and Q.26. The above equations (i) and (ii) are implemented in fig. 1.34 (a) and (b).

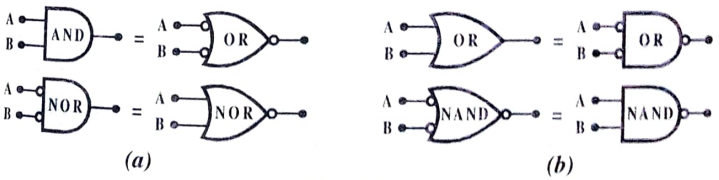


Fig. 1.34

Q.53. Write short note on AND-OR conversion to NOR-NOR gate networks.

Ans. Fig. 1.35 (a) shows a two level AND-OR gate network. A two level AND-OR gate network is converted into NOR-NOR gate network by replacing the AND gate with NOR gates and the OR gate with a NOR gate. But it is not logically equivalent.

At the inputs and output, it is corrected by using inverters as shown in fig. 1.35 (b). Fig. 1.35 (c) shows the modified circuit of fig. 1.35 (b).

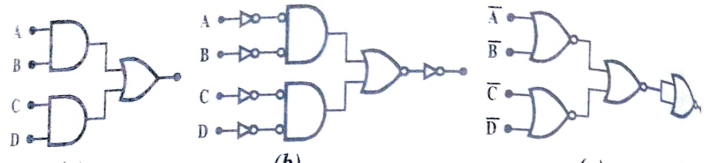


Fig. 1.35

Q.54. Explain in brief OR-AND conversion to NAND-NAND gate networks.

Ans. A two level OR-AND gate network is shown in fig. 1.36 (a). This is converted into NAND-NAND gate network by replacing the AND gate with a NAND gate and OR gate with NAND gates. But this is not logically equivalent at the inputs and output, it is corrected by using inverters as shown in fig. 1.36 (b). Fig. 1.36 (c) shows the modified circuit of fig. 1.36 (b).

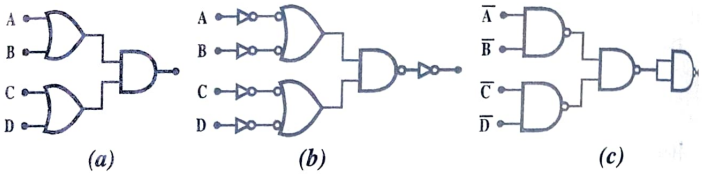
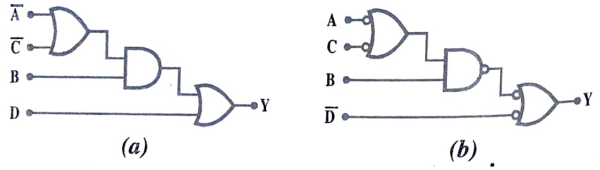


Fig. 1.36

Q.55. How can be converted to AND-OR into NAND-NAND gate networks ?

Ans. A multilevel AND-OR gate network is simply converted into a gate network with NAND-NAND gate network. This is logically equivalent to a gate with the uncomplemented switching variable and a bubble at the input of that gate to which the complemented switching variable can be connected



whenever a gate has a complemented switching variable at its input. Two bubbles are shown at both ends of a line connecting two gates since their is no logical effect. According to the above conditions, the circuit is depicted in fig. 1.37 (a). Fig. 1.37 (b) shows the modified circuit of fig. 1.37 (a).

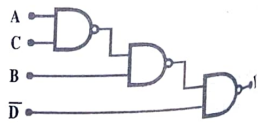


Fig. 1.37

Numerical Problems

Prob.41. Prove sum of equation $Y = ABCD + ABC\overline{D}$ using Karnaugh maps. ***(R.G.P.V., Dec. 2013)***

Sol. The given expression can be expressed as –

$$f(A, B, C, D) = \Sigma m (14, 15) \quad ...(i)$$

The K-map for the equation (i) is illustrated in fig. 1.38.

The minimized expression obtained from the K-map is given as under –

$$Y = ABC$$

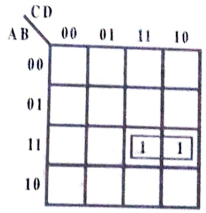


Fig. 1.38 K-map for f(A, B, C, D) = Σm (14, 15)

Prob.42. Simplify the following Boolean function with K-map – $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$. ***(R.G.P.V., June 2010, 2011, 2014)***

Or

Simplify the Boolean function using K-map. $F(A, B, C, D) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$ ***(R.G.P.V., Dec. 2017)***

Sol. Function F has four variables so a four variable map must be used.

K-map of given function F is shown in fig. 1.39.

The minimized expression is given below –

$$F = y' + w'z' + xz' \qquad \textbf{Ans.}$$

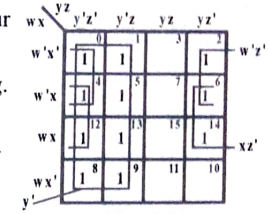


Fig. 1.39

Prob.43. Simplify the Boolean function – $f(w, x, y, z) = \Sigma m(1, 3, 7, 11, 15) + \Sigma d(0, 2, 5)$. ***(R.G.P.V., June 2014)***

Sol. The K-map for the given function with don't care condition is shown in fig. 1.40.

$$F = (w, x, y, z)$$
$$= \Sigma m(1, 3, 7, 11, 15) + \Sigma d(0, 2, 5)$$

The simplified expression is

$$F = \overline{w}\,\overline{x} + yz \qquad \textbf{Ans.}$$

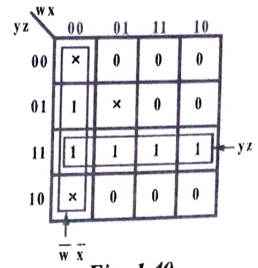


Fig. 1.40

Prob.44. Simplify the Boolean function
$F = B'C'D' + BCD' + ABCD'$
and the don't care condition
$d = B'CD' + A'BC'D$ *(R.G.P.V., June 201*

Sol. Given, $F = B'C'D' + BCD' + ABCD'$

The given function is of four variables. Hence,

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

$F(A, B, C, D) = \Sigma m(0, 6, 8, 14) + \Sigma d(2, 5, 10)$

The K-map for above function is shown in fig. 1.41.

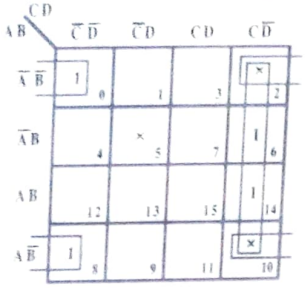


Fig. 1.41

The minimized expression is given below –

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

Prob.45. Minimize the following function using Karnaugh map method.
$F(a, b, c, d) = \Sigma m(1, 3, 7, 10, 12, 14) + \Sigma d(0, 9)$
(R.G.P.V., Dec. 201

Sol. The K-map for the above expression is shown in fig. 1.42.

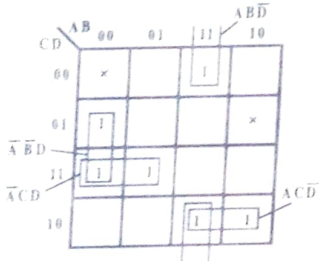


Fig. 1.42

$Y = \overline{A}\,\overline{B}\,D + \overline{A}CD + AC\overline{D} + AB\overline{D}$ An

Prob.46. Minimize the given Boolean function using K-map and implement the simplified function using only NAND gates.
$F(A, B, C, D) = \Sigma m(0, 1, 2, 9, 11, 15) + d(8, 10, 14).$
(R.G.P.V., Dec. 2015)

Sol. Function F has four variables so a four variable map must be used

$F(A, B, C, D) = \Sigma m(0, 1, 2, 9, 11, 15) + d(8, 10, 14)$

The K-map for the given function with don't care condition is shown in fig. 1.43.

The simplified expression is,

$$F = AC + B\,C + B\,D$$

Implementation of above minimized function using only NAND gate is shown in fig. 1.44.

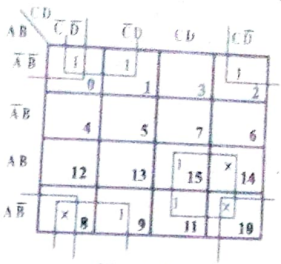


Fig. 1.43

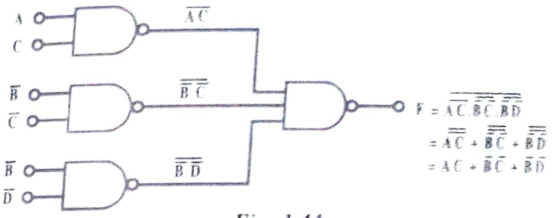


Fig. 1.44

Prob.47. Minimize the following switching functions using the Karnaugh map. List all prime implicants and essential prime implicants –
$F(x_1, x_2, x_3, x_4) = \Sigma(0, 1, 2, 3, 6, 7, 9, 13, 14, 15).$
(R.G.P.V., Dec. 2010)

Sol. The given function is

$F(x_1, x_2, x_3, x_4) = \Sigma(0, 1, 2, 3, 6, 7, 9, 13, 14, 15)$

K-map for given function is shown in fig. 1.45.

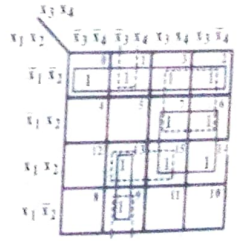


Fig. 1.45

The minimal form of the given function is

$$F = \overline{x}_1\overline{x}_2 + x_3x_2 + x_1\overline{x}_3x_4$$

The prime implicants of the function are

$x_2x_3x_4$, $\overline{x}_1x_2$, x_3x_2, $x_1x_3x_4$, $x_4x_1x_2$, x_1x_3

The essential prime implicants are

$\mathbf{x_1x_2}$, $\mathbf{x_3x_2}$, $\mathbf{x_1x_3x_4}$ **Ans.**

Prob.48. Determine the prime-implicants of the function and minimize function –

$$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15).$$

(R.G.P.V., Dec. 201

Or

Determine the prime-implicants of the function –

$$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$$

(R.G.P.V., June 201

Sol. Given function is

$F(w, x, y, z) = \Sigma(1, 4, 6, 7, 8, 9, 10, 11, 15)$

K-map for given function is shown in fig. 1.46.

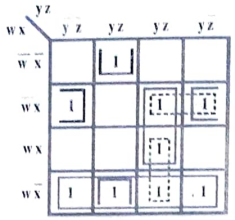


Fig. 1.46

From the K-map the prime implicants are given below –

$w\bar{x}$, yzx, $\bar{y}z\bar{x}$, $\bar{w}x\bar{z}$, $y\bar{w}x$, wyz

and minimized function is

$$F = wx + yzx + yzx + wxz$$

An

Prob.49. Reduce the expression $F = \Sigma_m(0, 2, 3, 4, 5, 6)$ using K-m and implement using NAND gates only. ***(R.G.P.V., Dec. 20***

Sol. The K-map for the given function is given in fig. 1.47.

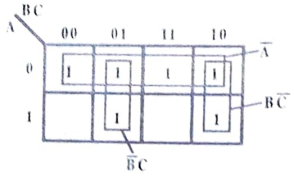


Fig. 1.47

The minimized expression is –

$$F = A + BC + BC$$

Implementation of above equation using NAND gate is shown in f 1.48.

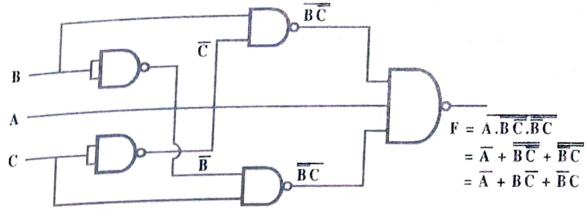


Fig. 1.48

Prob.50. Minimize the following function using K-map and implement the reduced expression using NAND gates only –

$$Y = \bar{A}\,\bar{B}\,\bar{C} + A\bar{C}\,\bar{D} + A\bar{B} + AB\bar{C}\,\bar{D} + \bar{A}\,\bar{B}C.$$

(R.G.P.V., Feb. 2010)

Sol. Given, $Y = \bar{A}\,\bar{B}\,\bar{C} + AC\,\bar{D} + A\bar{B} + AB\bar{C}\,\bar{D} + \bar{A}\,\bar{B}C$

Here, some terms are missing in first, second, third and fifth. By multiplying variables we get

$$Y = \bar{A}\,\bar{B}\,\bar{C}(D + D) + AC\,\bar{D}(B + B) + A\bar{B}(C + C)(D + D) + AB\bar{C}\,\bar{D} + \bar{A}\,\bar{B}C(D + D)$$

$$Y = \bar{A}\,\bar{B}\,\bar{C}D + \bar{A}\,\bar{B}C\,\bar{D} + AB\bar{C}\,\bar{D} + A\bar{B}\bar{C}\,\bar{D} + A\bar{B}(CD + C\bar{D} + \bar{C}D + \bar{C}\,\bar{D}) + AB\bar{C}\,\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D$$

$$Y = A\bar{B}\,\bar{C}D + A\bar{B}C\,\bar{D} + AB\bar{C}\,\bar{D} + A\bar{B}\bar{C}\,\bar{D} + AB\bar{C}D + AB\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}D$$

$$Y = A\bar{B}\,\bar{C}D + A\bar{B}C\,\bar{D} + AB\bar{C}\,\bar{D} + A\bar{B}\bar{C}\,\bar{D} + AB\bar{C}D + AB\bar{C}D + A\bar{B}CD + A\bar{B}\bar{C}D$$

$$Y = \Sigma m(0, 1, 2, 3, 8, 9, 10, 11, 12) \quad \ldots(i)$$

K-map for equation (i) is shown in fig. 1.49.

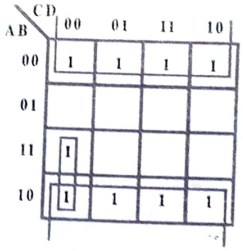


Fig. 1.49 K-map

Minimized expression obtained from K-map is

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

De-Morganize above equation then

$$Y = \overline{\overline{A}CD} + B = \overline{\overline{A}CD}.\overline{B}$$

Implementation of above equation using NAND gate is shown in fig. 1.

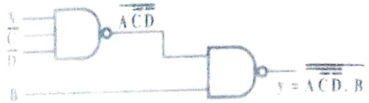


Fig. 1.50

Prob.51. With the aid of a K-map derive a minimal sum of product form of –

(i) $f(w, x, y, z) = \pi(1, 4, 5, 6, 11, 12, 13, 14, 15)$.
Is your answer unique ?

(ii) $f(w, x, y, z) = \Sigma(0, 2, 4, 9, 12, 15) + \Sigma\phi(1, 5, 7, 10)$ where d denotes don't care states.

(R.G.P.V., Dec. 20

Sol. (i) $f(w, x, y, z) = \pi(1, 4, 5, 6, 11, 12, 13, 14, 15)$

Fig. 1.51 shows the K map for given function.

From fig. 1.51, the minimal sum of products form is given as follow

$$f = x\,z + w\,yz$$

Fig. 1.51

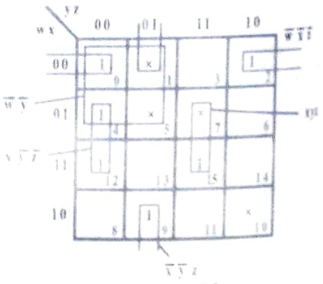


Fig. 1.52

(ii) The K-map for the given expression is shown in fig. 1.52

$$f = \overline{w}\,\overline{y} + x\overline{y}\,\overline{z} + \overline{x}\,\overline{y}z + xyz + \overline{w}\,\overline{x}\,\overline{z}$$

Prob.52. Minimize using Karnaugh map –

(i) $\Sigma_m(0, 1, 4, 7, 9, 12, 13) + \Sigma_\phi(2, 8, 14)$

(ii) $\Sigma_m(5, 7, 9, 12) + \Sigma_\phi(1, 2, 6, 15)$.

(R.G.P.V., June 20

Sol. ***(i) $\Sigma_m(0, 1, 4, 7, 9, 12, 13) + \Sigma_\phi(2, 8, 14)$***

The K-map for the above function is shown in fig. 1.53.

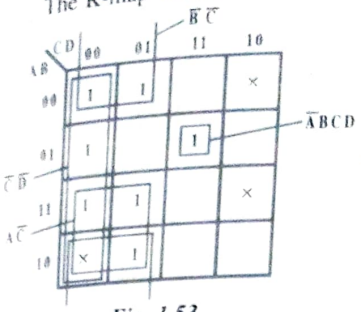


Fig. 1.53

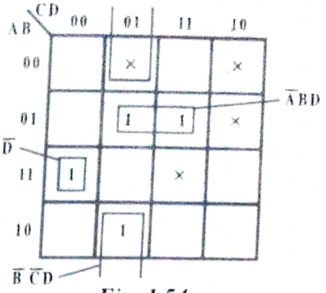


Fig. 1.54

The simplified expression is

$$Y = AC + B\,C + C\,D + ABCD \qquad \textbf{Ans.}$$

(ii) $\Sigma_m(5, 7, 9, 12) + \Sigma_\phi(1, 2, 6, 15)$

The K-map for the above function is shown in fig. 1.54.

The simplified expression is

$$Y = ABD + B\,CD + ABC\,D \qquad \textbf{Ans.}$$

Prob.53. Plot the following expressions in K-map and then minimize them

(i) $ABCD + ABC\,D + ABC + AB$

(ii) $Y = \Sigma m(7, 9, 10, 11, 12, 13, 14, 15)$.

(R.G.P.V., Dec. 2014)

Sol. (i) $Y = ABCD + ABCD + ABC + AB$

$= ABCD + ABCD + ABC(D + D) + AB(C + C)(D + D)$

$= ABCD + ABCD + ABCD + ABCD + (ABC + ABC)(D + D)$

$= ABCD + ABCD + ABCD + ABCD + ABCD$
$+ ABCD + ABCD + ABCD$

$= ABCD + ABCD + ABCD + ABCD + ABCD$
$+ ABCD + ABCD$

$= m_{15} + m_8 + m_{11} + m_{10} + m_{14} + m_{13} + m_{12}$

$\Sigma m(8, 10, 11, 12, 13, 14, 15)$ ...(i)

K-map for equation (i) is shown in fig. 1.55

Hence, simplified expression is

$$Y = AB + AC + AD \qquad \textbf{Ans.}$$

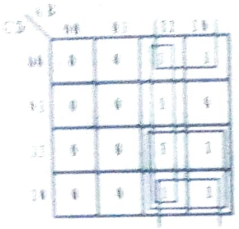


Fig. 1.55

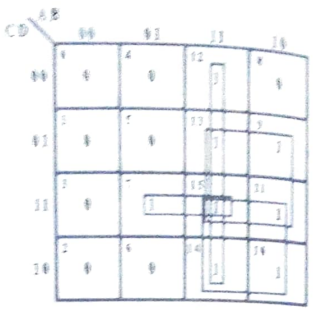


Fig. 1.56

(ii) $Y = \Sigma m(7, 9, 10, 11, 12, 13, 14, 15)$ ...(i)

K-map for equation (ii) is shown in fig. 1.56

Now simplified expression is

$$Y = AB + AD + AC + BCD$$
$$= A(B + D + C) + BCD$$ Ans

Prob.54. Given the logic equation –

$$F = A\bar{B}D + AB\bar{C}\,\bar{D} + \bar{A}BD + ABCD$$

(i) Make a truth table

(ii) Simplify using K-map.

(R.G.P.V., June 2015)

Sol. (i) The truth table for the given function is shown in table 1.21

Table 1.21

A	*B*	*C*	*D*	*A*	*C*	*D*	*AB*	*AB*	*ABD*	*ABC D*	*ABD*	*ABCD*	*Y*
0	0	0	0	1	1	1	0	1	0	0	0	0	0
0	0	0	1	1	1	0	0	1	1	0	0	0	1
0	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	1	1	1	0	0	0	1	1	0	0	0	1
0	1	0	0	1	1	1	0	1	0	0	0	0	0
0	1	0	1	1	1	0	0	1	1	0	1	0	1
0	1	1	0	1	0	1	0	1	0	0	0	0	0
0	1	1	1	1	0	0	0	1	1	0	1	0	1
1	0	0	0	0	1	1	0	1	0	0	0	0	0
1	0	0	1	0	1	0	0	1	1	0	0	0	1
1	0	1	0	0	0	1	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	1	1	0	0	0	1
1	1	0	0	0	1	1	1	0	0	1	0	0	1
1	1	0	1	0	1	0	1	0	0	0	0	0	0
1	1	1	0	0	0	1	1	0	0	0	0	1	1
1	1	1	1	0	0	0	1	0	0	0	0	0	0

(ii) Given function is simplified by using K-map as shown in fig. 1.57.

$$F = ABD + ABC\,D + A\bar{B}D + ABCD$$
$$= (A + B).D + ABC\,D + A\bar{B}D(C + C) + ABCD$$
$$= AD(B + B)(C + C) + BD(A + A)(C + C) + ABC\,D + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD$$
$$= (ABD + A\,BD)(C + C) + (ABD + A\,BD)(C + C) + ABC\,D + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD$$
$$= ABCD + ABC\bar{D} + A\,BCD + A\,BC\bar{D} + ABCD + AB\,CD + A\,BCD + A\,BC\bar{D} + ABCD + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD$$
$$= ABCD + ABC\bar{D} + A\,BCD + A\,BC\bar{D} + ABCD + A\bar{B}CD + A\bar{B}C\bar{D} + ABCD$$

$= \Sigma m = (1, 3, 5, 7, 9, 11, 12, 14)$

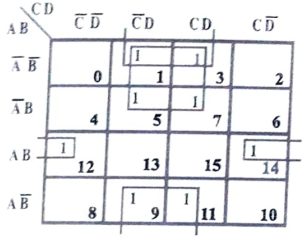


Fig. 1.57

The simplified expression is

$$F = \bar{A}D + \bar{B}D + AB\bar{D}$$ **Ans.**

Prob.55. Realise the following function as –

(i) Multilevel NAND-NAND gate network and

(ii) Multilevel NOR-NOR network.

$$F = ABC + B(C + D) + AD$$

Sol. The given function can be implemented as a four level AND-OR gate network as shown in fig. 1.58.

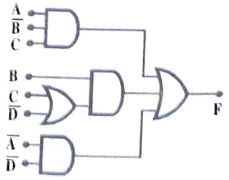
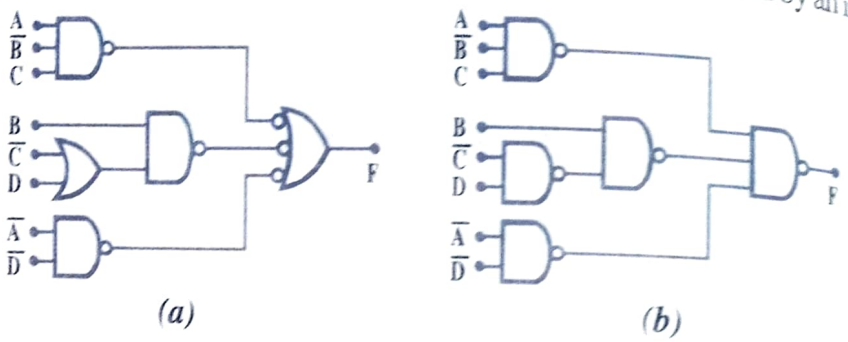


Fig. 1.58

(i) ***Multilevel NAND-NAND Implementation –***

(a) Each OR gate is replaced by OR gate with bubbles at its inputs

(b) Each AND gate is replaced by a NAND gate followed by an inverter



(a) *(b)*

Fig. 1.59

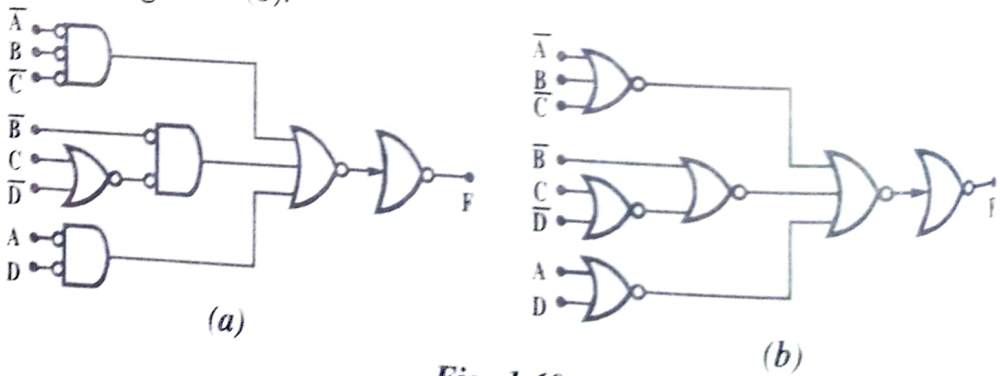
The corresponding input variables to bubbled OR gate is complemented so that the logically equivalent of the function is maintained as shown in fig. 1.59 (a). Now the logic circuit shown in fig. 1.59 (a) can be modified as shown in fig. 1.59 (b).

(ii) ***Multilevel NOR-NOR Implementation –***

(a) Each AND gate is replaced by AND gate with bubbles at its inputs

(b) Each OR gate is replaced by a NOR gate followed by an inverter

The corresponding input variables to bubbled AND gate is complemented so that the logically equivalent of the function is maintained as shown in fig. 1.60 (a). Now the logic circuit shown in fig. 1.60 (a) can be modified as shown in fig. 1.60 (b).



(a) *(b)*

Fig. 1.60