Unit-1 : Digital Concepts, Number systems, Boolean switching algebra. (10 hrs)

Introduction to Number systems -
1. positional Number systems, 1
2. Number system Conversion,

Binary Codes - 2
1. Binary arithmetic - 1
2. Binary logic functions. - 1

Switching algebra - 2

Functionally Complete Operation sets;
1. Reduction of switching equations using Boolean algebra. - 2
2. Realization of switching function. - 1

Introduction to Number system :

Analog system:

System which are capable of processing Continuous range of values which varies with respect to time.

Example: 1) tuning sections of radio.
2) V + I measured in meters.

Digital systems:

Digit refers to the discrete Counting unit systems which processes discrete values are digital systems.

Example: 1) Digital calculators.
2) Digital watches.

Advantages of digital systems:

1. Easier to design.
2. Storage of information is easier.
3. high accuracy and precision.
4. Less affected to noise.

Disadvantage :

1. Most of the signals available in real world are analog. so Conversion is necessary.

Number systems:

There are four main number systems.

1. Decimal number system.
2. Binary number system.
3. Octal number system.
4. Hexadecimal number system.

General Number representation:

A number N is represented generally as

$$N_r = a_n r^n + a_{n-1} r^{n-1} + \cdots a_2 r^2 + a_1 r^1 + a_0 + a_{-1} r^{-1} \cdots\cdots a_{-m} r^{-m}.$$

where

N = Number.

r = radix or base.

$a^n, a_{n-1} \cdots a_{-1}$ = Values of the n^{th} digit

n-1 from the point.

a_n ranges from 0 to r-1

n positional weightage

n increases from 0 to n to the left of the decimal point.

* All the number system follow the Principle of positional weighting.

1. If the number to the left of the decimal point is taken, as the position to the left increases the weightage increases

2. If the digits to the right of the

decimal point is taken, as the position of the right increases the weightage decreases.

3. So the position of the digit with reference to the decimal point determines its weight. this is called positional weighting or positional number system.

Decimal Number system:

The base or radix is 10

i.e., $r = 10$.

The a_n - - - - a_{n-1} co-efficients ranges from 0 to 9.

Example:

$(7395.362)_{10}$ ⇒ decimal number.

It can be written as

$7\times10^3 + 3\times10^2 + 9\times10^1 + 5\times10^0 + 3\times10^{-1} + 6\times10^{-2} + 2\times10^{-3}$

Here

$a_0 = 5$ 7 ⇒ MSD

$a_1 = 9$ 5 ⇒ LSD

$a_2 = 3$

$a_3 = 7$

So, as the digit to the left of decimal increases, the weightage increases and as the digits to the right, the weightage decreases.

7 has 10^3 - has more weightage than below term.

3 has 10^2

So all the digits are given weightage depending on the position.

Binary Number system:

* The base or radix is 2 i.e., $r = 2$
* The $a_n, a_{n-1}, - a_1 - a_0$
* The Co-efficient ranges from 0 to 1

Eight bit = 1 byte ; 4 bit = nibble.

Example: 10110.0110.

This can be written as

$$1\times2^4+0\times2^3+1\times2^2+1\times2^1+0\times2^0+0\times2^{-1}+1\times2^{-2}$$
$$+1\times2^{-3}+0\times2^{-4}.$$

Here

$a_0 = 0$ - LSB

$a_1 = 1$

$a_2 = 1$

$a_3 = 0$

$a_4 = 1$ - MSB.

Octal Number system:

* Base or radix is 8
* $a_n, a_{n-1}, - a, - a_0$ Co-efficient ranges from 0 to 7
* Advantage is it is Compact and occupies less space per data.

Example: 2346.12

$$= 2\times8^3+3\times8^2+4\times8^1+6\times8^0+1\times8^{-1}+2\times8^{-2}.$$

Here

$a_0 = 6$ - LSB

$a_1 = 4$ $r = 8$

$a_2 = 3$ $n = 3$

$a_3 = 2$ - MSB

Hexadecimal number system:

* Base or radix is 16
* $a_n, a_{n-1} --- a_0$, Co-efficients ranges from 0 to F.

i.e, (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F)

* Advantage is that it occupies less space than binary + there is a straight forward conversion possible.

Example:

52A6.2B5

Binary to octal.

Method: Group the binary in groups of three from the decimal point and write the octal equivalent.

(1) $(11\,111\,110)_2$

110 → 6
111 → 7 $= (376)_8$
011 → 3

(2) $(101011 \cdot 1010)_2$

101 = 5
011 = 3 $= (53 \cdot 5)_8$
101 = 5

Binary to hexadecimal.

Method: Group the binary in groups of four starting from the decimal point both the left and right and write the hexadecimal equivalent.

(1) $(110111110110)_2$

0110 ⇒ 6
1111 ⇒ F $(DF6)_{16}$
1101 ⇒ D

1010 ⇒ A
1011 ⇒ B
1100 ⇒ C
1101 ⇒ D
1110 ⇒ E
1111 ⇒ F

(2) $(10111 \cdot 1101)_2$

1101 ⇒ D
0111 ⇒ 7 $(17 \cdot D)_{16}$
0001 ⇒ 1

(3) $(1110111 \cdot 1101101)_2$

0111 ⇒ 7 $(77 \cdot DA)_{16}$
0111 ⇒ 7
1101 ⇒ D
1010 ⇒ A

$r = 16$

$n = 3$

$$5\times16^3 + 2\times16^2 + A\times16^1 + 6\times16^0 + 2\times16^{-1} + B\times16^{-2} + 5\times16^{-3}$$

$a_0 = 6$

$a_1 = A$

$a_2 = 2$

$a_3 = 5.$

Number system Conversion:

Binary to other Number system:

Binary to decimal.

$(10111)_2$ to decimal.

1 0 1 1 1

$1\times2^0 = 1$

$1\times2^1 = 2$

$1\times2^2 = 4$

$0\times2^3 = 0$

$1\times2^4 = 16$

$\underline{\quad 23 \quad}$

$(23)_{10}$

$(1011.101)_2$ to decimal.

1 0 1 1 . 1 0 1

$1\times2^{-3} = 1/8 = 0.125$

$0\times2^{-2} = 0$

$1\times2^{-1} = 1/2 = 0.5$

$1\times2^0 = 1$

$1\times2^1 = 2$

$0\times2^2 = 0$

$1\times2^3 = 8$

$\underline{\quad 11.625 \quad}$

$(11.625)_{10}$

Table showing the equivalent decimal, Octal and hexadecimal for Binary.

Binary	Decimal	octal	hexadecimal
0	0	0	0
1	1	1	1
10	2	2	2
11	3	3	3
100	4	4	4
101	5	5	5
110	6	6	6
111	7	7	7
1000	8	10	8
1001	9	11	9
1010	10	12	A
1011	11	13	B
1100	12	14	C
1101	13	15	D
1110	14	16	E
1111	15	17	F
10000	16	20	10

Decimal to other number system:

Decimal to Binary.

Successive division method.

1. Divide the decimal number by 2.
2. After one time division write the remainder to the side. Repeat the procedure till the remainder becomes 1 or 0.
3. The final result is obtained by assembling all the remainders with the last remainder becoming the MSB (Most significant bit).

(1) $(43)_{10}$ to binary:

2 | 43
2 | 21 − 1 → LSB
2 | 10 − 1
2 | 5 − 0
2 | 2 − 1

$= (101011)_2$

2. Convert $(21.6875)_{10}$ to binary.

Procedure for decimal point numbers.

For 21 same method is followed.
For fraction, it is multiplied by 2 to given integer and a fraction.
The new fraction is multiplied by 2 to give a new integer and a fraction.
The process is continued till the fraction is zero or the no. of digit is larger or the sufficient accuracy.

$$\begin{array}{r|l} 2 & 21 \\ 2 & 10 - 1 \\ 2 & 5 - 0 \\ 2 & 2 - 1 \\ & 1 - 0 \end{array}$$

$(10101)_2$

$0.6875 \times 2 = 1.3750$
$= 1 + 0.3750$

$0.3750 \times 2 = 0.7500$
$= 0 + 0.7500$

$0.7500 \times 2 = 1.5000$
$= 1 + 0.5000$

$0.5000 \times 2 = 1.0000$
$= 1 + 0.0000$

$(1011)_2$

$\Rightarrow (10101.1011)_2$

Decimal to octal.

(1) $(153)_{10}$ to octal.

$$\begin{array}{r|l} 8 & 153 \\ 8 & 19 - 1 \\ & 2 - 3 \end{array}$$

$= (231)_8$

(2) $(645.926)_{10}$ to octal.

$$\begin{array}{r|l} 8 & 645 \\ 8 & 80 - 5 \\ 8 & 10 - 0 \\ & 1 - 2 \end{array}$$

$0.926 \times 8 = 7 + 0.4080$
$0.4080 \times 8 = 3 + 0.2640$
$0.264 \times 8 = 2 + 0.112$
$0.112 \times 8 = 0 + 0.869$
$0.869 \times 8 = 7 + 0.168$
$0.168 \times 8 = 1 + 0.334$
$0.334 \times 8 = 2 + 0.752$

Ans: $(1205.7320712\ldots)_8$

Decimal to hexadecimal.

1. $(464)_{10}$ to hexa.

$$\begin{array}{r|l} 16 & 464 \\ \hline 16 & 29 - 0 \\ \hline & 1 - D \end{array}$$

$(1D0)_{16}$

2. $(121.650)_{10}$ to hexa.

$$\begin{array}{r|l} 16 & 121 \\ \hline & 7 - 9 \end{array}$$

$0.650 \times 16 = A + .400$
$0.400 \times 16 = 6 + 0.400$
$0.400 \times 16 = 6 + 0.400$
$0.400 \times 16 = 6 + 0.400.$

$(79.A666\ldots)_{16}$

Octal to other Number system.

Octal to Binary:

write the Binary equivalent (3 digit) for each of the octal no. that gives the binary equivalent for the Octal.

(1) $(7612)_8$ to Binary.

7 - 111
6 - 110
1 - 001
2 - 010

$(111110001010)_2$

(2) $(536.62)_8$ to Binary.

5 - 101
3 - 011
6 - 110
6 - 110
2 - 010

$(101011110.110010)_2$

Octal to Decimal.

1. $(37365)_8$ to decimal.

3 7 3 6 5

$5 \times 8^0 = 5$

$6 \times 8^1 = 48$

$3 \times 8^2 = 192$

$7 \times 8^3 = 3584$

$3 \times 8^4 = 12288$

16117

Answer : $(16117)_{10}$

2. $(63174 \cdot 216)_8$ to decimal.

6 3 1 7 4 · 2 1 6

$6 \times 8^{-3} = 0{\cdot}0117$

$1 \times 8^{-2} = 0{\cdot}0156$

$2 \times 8^{-1} = 0{\cdot}25$

$4 \times 8^0 = 4$

$7 \times 8^1 = 56$

$1 \times 8^2 = 64$

$3 \times 8^3 = 1536$

$6 \times 8^4 = 24576$

$26236{\cdot}277$

Answer : $(26236 \cdot 277)_{10}$

octal to hexadecimal.

(1) Convert octal to binary then Convert to hexadecimal.

$(721)_8$ to hexa

$= (\underline{1}11\ \underline{1010}\ \underline{0001})$

$1101 - D$

$0001 - 1$

$= (1\,D\,1)_{16}$

(2) $(63174 \cdot 216)_8$ to hexa.

110 011 00111 0100 · 0100 01110

0110 - 6
0110 - 6
0111 - 7
0100 - 4
0111 - 7

$(6674 \cdot 47)_{16}$

Hexadecimal to other Number system.

Hexa - binary.

1. $(1A53)_{16}$ to Binary

$(0001\ 1010\ 0101\ 0011)_2$

2. $(AB12 \cdot CD)_{16}$ to Binary

$(1010\ 1011\ 0001\ 0010 \cdot 1100\ 1101)_2$

Hexa - decimal.

(1) $(67F2)_{16}$ to decimal.

6 7 F 2

$2 \times 16^0 = 2$
$F \times 16^1 = 240$
$7 \times 16^2 = 1792$
$6 \times 16^3 = 24576$

26,610

Answer : $(26610)_{10}$

(2) $(AB \cdot A2)_{16}$ to decimal.

A B · A 2

$2 \times 16^{-2} = 0 \cdot 0078$
$A \times 16^{-1} = 0 \cdot 625$
$B \times 16^0 = 11$
$A \times 16^1 = 160$

171·63281

Answer $(171 \cdot 63281)_{10}$

Hexadecimal to octal:

Convert the hexa to binary, group into 3's and write the octal equivalent for the grouped binary.

1. $(1A5B)_{16}$ to octal.

0001 1010 0101 1011

001 - 1
101 - 5
011 - 3

$(15133)_8$

2. $(1F67 \cdot E1)_{16}$ to octal.

0001 1111 0110 0111 · 1110 0001

001 - 1
111 - 7
101 - 5
100 - 4
111 - 7
111 - 7
000 - 0
010 - 2

$(17547 \cdot 702)_8$

Binary Codes:

Introduction:

→ The original circuits and Computers process data in the binary form because of the bistable nature of digital electronic circuits.

→ This may be true to internal operation but external world is decimal in nature.

→ Hence, the circuits are required to handle data which may be numeric, alphabets or special characters.

→ So, the conversion between decimal and binary are performed.

⇒ The decimal value becomes large and it becomes tedious to do conversions, since they become long and complicated.

⇒ For this reason the encoding decimal numbers and that combines some features of both the decimal and binary system.

⇒ The process of doing this is "ENCODING"

⇒ when numbers, letters or words are represented by a special group of symbols we say that they are being encoded and the group of symbols is called a "CODE."

⇒ The decimal numbers can be represented by an equivalent binary number. i.e., in 0's and 1's, code representing the decimal number called "STRAIGHT BINARY CODING". (BCD)

⇒ The numbers in a digital system or computer are used in coded form and this is done to acheive two things.

- To represent numeric (0-9), alphabet (A-Z, a-z) and special characters (+, *) with binary digit 0, 1 alone.
- To check whether a character transmitted in the coded form is correctly received and if not go for correction i.e., detection and correction of errors.

Classification

1. Weighted code
2. Non-weighted code.

Weighted code:

For each position or bit, there is a specific weight attached.

Example:

$$1000 = 1\times 2^3 + 0\times 2^2 + 0\times 2^1 + 0\times 2^0$$
$$= 8$$

Binary weighted code.

Non-weighted code:

No specific weight attached or if a particular representation is for a number, then it is non-weighted code.

Weighted Code:

→ There are many weighted Code, 8421 Code is the most popular code.

→ The bits are assigned the weights represented in the code name.

→ The main advantage of these codes is their easy convertibility to decimal system.

→ These codes represent each digit in a decimal number to its binary equivalent.

Example:

426 in 8421 code.

0100 0010 0110.

→ Because of easy convertibility this type of number system is used in digital system.

BCD [Binary Coded decimal] CODE:

The decimal number is coded straightly into binary so called BCD code.

Example:

$(5679)_{10}$ to BCD.

5 ⇒ 0101
6 ⇒ 0110
7 ⇒ 0111
9 ⇒ 1001

$(5679)_{10} = (0101011001111001)_{BCD}$.

$(4689)_{10}$ to BCD.

4 ⇒ 0100
6 ⇒ 0110
8 ⇒ 1000
9 ⇒ 1001

$(4689)_{10} = (0100011010001001)_{BCD}$

$(2346)_{10}$ to BCD.

2 ⇒ 0010
3 ⇒ 0011
4 ⇒ 0100
6 ⇒ 0110

$(2346)_{10} = (0010001101000110)_{BCD}$.

BCD to decimal equivalent.

1. $(\overset{4}{\overline{0100}}\,\underline{0111}\,\overset{8}{\overline{1000}}\,\underline{1000})_{BCD}$.

$\Downarrow$ 7 $\Downarrow$ 8

$= (4788)_{10}$.

2. $(\underline{0010}\ \underline{0100}\ \underline{0110} \cdot \underline{1001}\ \overset{3}{\overline{0011}})_{BCD}$.

$\Downarrow$ 2 $\Downarrow$ 4 $\Downarrow$ 6 9

$= (246 \cdot 93)_{10}$

BCD Arithemetic:

1. Addition. 2. subtraction 3. Multiplication.
4. Division.

Addition:

(1) Add 242 and 531 in BCD code.

242	0010	0100	0010
531	0101	0011	0001
	0111	0111	0011

decimal equivalent $(773)_{BCD}$.

(2) 649 and 418.

(1)
9 - 1001
8 - 1000
1 0001
0110
1 0111
Carry $\Downarrow$ 7.

1 0001 => illegal BCD Number so add 0110

(2)
0100 - 4
0001 - 1
0101 - 5
1
0110
$\Downarrow$ 6

previous Carry is to be added.

(3)
6 - 0110
4 - 0100
1010
0110
1 0000
Carry $\Downarrow$ 0

1010 => illegal BCD Number so add 0110.

Sum = 1 0 6 7

$(1067)_{10}$

Codes:

1. Numeric
2. Alphanumeric

Numeric

1. Weighted. → 1. positively weighted Code
   2. Negatively weighted Code

   Example:

   (1) 8421, 5211, 2421, 3321

   (2) 642-3, 631-1, 84-2-1, 74-2-1.
2. Non-weighted
3. Self Complementing → The code word obtained from the code word by interchanging 1 - 0 & 0 - 1.
4. Sequential → each succeeding code word is one binary number greater than its preceeding code word.
5. Error detecting and Correcting.
6. Reflective → Mirror image.
7. Cyclic. → sucessive code word differs from the preceeding one in only one bit position.

Alphanumeric

1. ASCII
2. EBCDIC
3. Hollerith.

BCD: 0 - 9, Coded with 4 bit.
Weighted Code
Sequential Code

Excess-3: Non-weighted Code
Sequential Code
Self complementing Code

Gray: non-weighted Code
Reflective Code
unit distance code [cyclic code].

Comparison of BCD and Binary.

1. It is important to realise that BCD is not a number system, it is a code.

2. It is a easy way of representing a decimal system.

3. It is not same as a straight binary number.

Advantages and disadvantages:

1. Easy Convertibility to decimal number system.

2. Though this is a decided advantage, we use a few digits as possible in natural binary encoding whereas we lose this advantage in going for BCD representation.

Non-Weighted Codes:

8421 Code is a weighted Code as each bit Position has been assigned a definite weight.

On the other hand, Gray code, excess 3 Code, ASCII Code, EBCDIC Code are classified as non-weighted code.

Gray Code, Excess 3 Code, parity Code Hamming Code are applied for error detection as well as error Correction.

ASCII, EBCDIC are applied for transmission of alphanumeric data.

Gray code:

It belongs to minimum change Codes in which only one bit in a code group changes when going from one step to the next. Called "mirror reflecting Code".

Decimal	Building Gray bits	Gray Code.
0	0	0000
1	1	0001
2	11	0011
3	10	0010
4	110	0110
5	111	0111
6	101	0101
7	100	0100
8	1100	1100
9	1101	1101
10	1111	1111
11	1110	1110
12	1010	1010
13	1011	1011
14	1001	1001
15	1000	1000

Advantages and Disadvantages.

* Since the bits are built up by just reversing the previous Combination no weight can be attached to the bits.

* hence unsuitable for arithemetic Operations.

Conversion:

Gray to Binary:

Steps for Conversion:

1. The MSB in Gray code is the same as in the binary number, hence record the MSB in the output.
2. Add the MSB in the O/P to the bit immediately on its right in the input and record the sum. If there is carry, it should be ignored.
3. Continue adding bits in the output to bits immediately to their right in the input until all bits have been added and the LSB is reached.
4. The final sum will be the binary equivalent which will have the same number of bits as the gray code.

four rules:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$

Example:

1. Convert 10101111 to Binary.

Solution:

1 0 1 0 1 1 1 1

1 1 0 0 1 0 1 0

$(10101111)_{gray} = (11001010)_2$

2. Convert 11011110 to binary.

Solution:

1 1 0 1 1 1 1 0

1 0 0 1 0 1 0 0

$(11011110)_{gray} = (10010100)_2$

Binary to Gray:

Step for Conversion:

1. The MSB in binary is same in Gray.
2. Add the MSB in binary (input) to the bit immediately to its right in binary and record the sum in the o/p. ignore the carry.
3. Repeat step 2 until all the bits in the binary number have been added.

Example:

1011 is a binary number.

$$1 \xrightarrow{+} 0 \xrightarrow{+} 1 \xrightarrow{+} 1$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 1 \quad 1 \quad 0$$

1. Convert 11001011 to Gray Code.

Solution:

$$1 \xrightarrow{(+)} 1 \xrightarrow{(+)} 0 \xrightarrow{(+)} 0 \xrightarrow{(+)} 1 \xrightarrow{(+)} 0 \xrightarrow{(+)} 1 \xrightarrow{+} 1$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0$$

$(11001011)_2 = (10101110)_{gray}$

---

2. Convert 10110101 to Gray

$$1 \xrightarrow{+} 0 \xrightarrow{+} 1 \xrightarrow{+} 1 \xrightarrow{+} 0 \xrightarrow{+} 1 \xrightarrow{+} 0 \xrightarrow{+} 1$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1$$

$(10110101)_2 = (11101111)_{gray}$.

Excess - 3 - Code:

It is also an non-weighted code and is generally used with BCD numbers.

Decimal digit	BCD Code	Excess-3 Code	Complement of Excess-3
0	0000	0011	1100
1	0001	0100	1011
2	0010	0101	1010
3	0011	0110	1001
4	0100	0111	1000
5	0101	1000	0111
6	0110	1001	0100
7	0111	1010	0101
8	1000	1011	0100
9	1001	1100	0011

1. Express 452 as Excess 3 number.

$$\begin{array}{r} 4\ \ 5\ \ 2 \\ +\ 3\ \ 3\ \ 3 \\ \hline 7\ \ 8\ \ 5 \\ \hline \end{array}$$

785 = (0111 1000 0101)

2. Convert 392 to Excess 3 Code.

$$\begin{array}{r} 3\ \ 9\ \ 2 \\ (+)\ 3\ \ 3\ \ 3 \\ \hline 6\ \ 12\ \ 5 \\ \hline \end{array}$$

6 12 5 = (0110 1100 0101)

3. Convert Excess 3 number 100111001100 to decimal equivalent.

$$\begin{array}{lr} & 1001\ 1100\ 1100 \\ & -\ 0011\ 0011\ 0011 \\ \hline \text{BCD.} & 0110\ 1001\ 1001 \\ \hline \text{Decimal} & 6\qquad 9\qquad 9 \end{array}$$

decimal equivalent $= (699)_{10}$

Advantage:

* when we try to add 8421 number whose decimal sum exceeds 9, obstacle arises, This is overrid by excess-3.

* This code is Very useful in digital system as this requires very simple electronic circuit for subtraction Operation.

1. perform the following additions in XS-3 code.

a. 37 + 28. b. 247.6 + 359.4.

$$\begin{array}{lcll} & & 1 & \\ 37 & \Rightarrow & 0110 & 1010 \\ 28 & \Rightarrow & 0101 & 1011 \\ \hline 65 & & 1100 & 0101 \\ \hline & - & 0011 & +0011 \\ \hline & & 1001 & 1000 \\ \hline \end{array}$$

add 0011 to correct 0101

subtract 0011 to correct 1100

65 in Excess 3 = 1001 1000

<u>Note:</u> Excess 3 or XS-3 is a non-weighted BCD code.

Just add 0011 to 8421 code we call as Excess-3 code. so called sequential code.

* Sequential code used for arithemetic operations.

* Also called as self-complementing code, Subtraction is performed by the method of complement addition.

* The Excess 3 code has six invalid states. i.e., 0000, 0001, 0010, 1101, 1110, 1111.

<u>To perform addition:</u>

A bit groups are added seperately if there is no carry ~~add~~ subtract 0011

Reason: when two decimal digits are added in XS-3 and there is no-carry, the result is in XS-6.

* If there is carry out, add 0011 to the sum term.

<u>Reason:</u> when there is a carry, the invalid states are skipped and the result is in normal binary.

b. 247.6 + 359.4.

		No carry	Carry	Carry		Carry
247.6	⇒	0101	0111	1010	.	1001
359.4	⇒	0110	1000	1100	.	0111
607.0		1100	0000	0111	.	0000
	−	0011	(+)0011	(+)0011		(+)0011
		1001	0011	1010	.	0011

Excess-3 Sum is $(607.0)_{10}$.

<u>Error detecting code:</u>

when binary data is transmitted and processed, it is susceptible to noise that can alter or distort its contents.

i.e) 1's to 0's or 0's to 1's.

Digital system must be accurate to the digits, errors can pose a serious problem.

We have many schemes to detect the error i.e., when any single bit error is devised or detected the binary word can be corrected and retransmitted.

Parity: The simplest technique for detecting errors is that of adding an extra bit, known as parity bit.

1. odd parity.
2. even parity.

Odd parity:

the parity bit is set to a 0 or 1 at the transmitter such that total number of 1 bits in the word including the parity bit is an odd number.

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.

* when digital data is received, a parity checking circuit generates an error signal if the total No of 1's is even in odd parity or odd in even parity. system.

* This parity checks can detect only single bit error and not more than one bit.

* Most often used parity check is odd parity, because even parity does not detect the situation where all 0's are created by a short circuit or some other fault condition.

Example:

In an even parity scheme, which of the following words Contain an error.

a) 10101010 ⇓ no. of 1's = 4 = even ⇓ No. error

b) 11110110 ⇓ 6 = even ⇓ No error.

c 10111001. ⇓ 5 = odd ⇓ This word has an error.

In an odd parity scheme, which of the following words Contain an error.

a) 10110111 ⇓ No. of 1's = 6 = even ⇓ has an error.

b) 10011010. ⇓ 4 = even ⇓ has an error

c) 11101010. ⇓ 5 = odd ⇓ has no error.

Error Correcting Code:

* The parity bit indicates only whether the error exists or not.

* But it will not tell which bit is incorrect

* For a code to be single bit error-correcting Code, the minimum distance of that code must be three.

* A Code with minimum distance of three not only be able to detect error or Correct two bit errors.

* If the Erroneous bit is detected it is easy to Correct it by Complementing the bit

The other code available to correct the code is hamming code.

Format:

We have 7-bit, 12-bit, 15-bit hamming code.

If four bit data to be transmitted, three parity bits located at positions 2^0, 2^1 and 2^2 from the left are added to make it 7-bit code word which is then transmitted.

$$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7.$$

D - data bits
P - parity bits.

P_1 is set to 0 or 1 to establish even parity bit. [1, 3, 5 and 7]

P_2 is set to 0 or 1 to establish even bit [2, 3, 6, 7]

P_4 is set to 0 or 1 to establish even bit [4 5, 6, 7]

* Hamming code is called as "Self Correcting Code"

Example:
Encode data bits 1101 into the 7-bit even Parity hamming code.

$$P_1 \quad P_2 \quad D_3 \quad P_4 \quad D_5 \quad D_6 \quad D_7.$$
$$1 \qquad 1 \quad 0 \quad 1.$$

Bits 1 3 5 7. i.e. (P_1 1 1 1) if $P_1 = 1$ only it will be even parity.

so $P_1 = 1$

Bits 2 3 6 7 ($P_2$101) = even parity if $P_2 = 0$

so $P_2 = 0$.

Bits 4 5 6 7 ($P_4$101) = even parity only if P_4 0

so $P_4 = 0$.

The final Code is

(1 0 1 0 1 0 1)

Example 2.

Construct a even parity seven bit hamming code for the word 1001

P_1	P_2	D_3	P_4	D_5	D_6	D_7
0	0	1	1	0	0	1

Bits 1 3 5 7 i.e) (P_1 101) if $P_1 = 0$ only Code will be even parity so $P_1 = 0$

Bits 2 3 6 7 i.e) (P_2 101) if $P_2 = 0$ only Code will be even parity so $P_2 = 0$

Bits 4 5 6 7 i.e) (P_4 001) if $P_4 = 1$ only Code will be even parity so $P_4 = 1$.

The final Code is

(0011001)

Alphanumeric Code :

* Alphanumeric Codes are codes used to encode the Characters of alphabet in addition to the decimal digits.

* They are used primarily for transmitting data between Computers and its I/O devices such as Printers, Keyboard + video display terminals.

* alphanumeric code can encode 10 decimal digits and 26 alphabetic characters.

1. ASCII Code.
2. EBCDIC Code.

ASCII Code:

American standard Code for Information Interchange.

* Basically a seven-bit Code.
* It can Oncode both uppercase and lower case letters.
* ASCII is very easy for a computer to alphabetize and sort.

EBCDIC code :

* Extended Binary coded Decimal interchange Code.
* Used to encode the symbols and Control Characters found in ASCII
* used in most large Computers for Communicating alphanumeric data.
* This Code uses binary - Coded decimal as the basis of binary assignment.
* closely related to Punched Card Codes.

Hamming Code.

Hamming code not only provides the detection of a bit error, but also identifies which bit is in error so that it can be corrected. Thus Hamming code is called error detecting and correcting code.

Encode the binary word 1011 into seven bit even parity hamming code.

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	0	1	0	1

(1, 3, 5, 7) have 3 1's, so to make it even parity add 1 to P_1, so $P_1 = 1$

(2, 3, 6, 7) have 2 1's so to make it even parity add 0 to P_2, so $P_2 = 0$

(4, 5, 6, 7) have 2 1's so to make it even parity add 0 to P_4, so $P_4 = 0$.

The generated hamming code is 1010101

Determine the single error-correcting code for the information code 10111 for odd parity.

No. of parity bits $\Rightarrow 2^p = x + p + 1$

if $p = 3$ $\quad 2^3 = 8 = 5 + 3 + 1 \neq 9$

if $p = 4$ $\quad 2^4 = 16 = 5 + 4 + 1 = 10$

4 parity bits are sufficient.

D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	0	1	1	1	1	1	0

(1, 3, 5, 7, 9) (2, 3, 6, 7) (4, 5, 6, 7) (8, 9)

1, 3, 5, 7, 9.

$P_1 = ?$

no. of 1's in (3, 5, 7, 9) = 3 so $P_1 = 0$ for odd parity.

2, 3, 6, 7.

$P_2 = ?$

no. of 1's in (7, 3, 6) = 2 so $P_2 = 1$ for odd parity.

4, 5, 6, 7 ; $P_4 = ?$

no. of 1's in (5, 6, 7) = 2 so $P_4 = 1$ for odd parity

8, 9 ; $P_8 = ?$

no. of 1's in (9) = 1 so $P_8 = 0$ for odd parity

the data bit is

(100111110)

Detecting and Correcting an Error.

(1) Assume that the even parity Hamming code in example (0110011) is transmitted and that 0100011 is received the receiver does not know about what was transmitted Determine bit location where error has occurred using received code.

Ans:

D_7	D_6	D_5	P_4	D_3	P_2	P_1
0	1	0	0	0	1	1

Check for parity bits:

(1) P_1 checks 1, 3, 5, 7.

No. of 1's = 1 (odd) so $P_1 = 1$ is wrong for even parity 1 (LSB)

P_2 checks for 2, 3, 6, 7.

There are 2 No. of 1's so check for parity is correct so '0'

P_4 Checks for 4, 5, 6, 7.

There are 1 No of 1 so check for even parity is wrong so '1'

The resultant is $\underset{LSB}{101} = 5$, go to 5^{th} location and change '0' to 1, Therefore the correct code is (0110011)

The Hamming code 101101101 is received. correct it if any errors. There are four parity bits and odd parity is used.

Ans:

D_9	P_8	D_7	D_6	D_5	P_4	P_3	P_2	P_1
1	0	1	1	0	1	1	0	1

Check for parity bit.

P_1 Checks for (1, 3, 5, 7, 9)

No. of 1's is 4 → check for odd parity is wrong so P_1 = 1 (LSB)

P_2 checks for (2, 3, 6, 7)

No. of 1's is 3 ⇒ Check for odd parity is correct so P_2 = 0

P_4 Checks for (4, 5, 6, 7)

No. of 1's is 3 → check for odd parity is correct so P_4 = 0.

P_8 checks for (8, 9)

No. of 1's is 1 → check for odd parity is correct so = 0

The resultant is 0001 ⇒ 1 the bit is in the Number 1 location has to be corrected and it contains error.

so

1 ⇒ 0

the transmitted data is

(1011 01100)

Extra:

Excess 3 addition.

(a) 8 + 6 (b) 1 + 2.

$$\begin{array}{r} \text{carry } 1011 \\ 1001 \\ \hline 1\,0100 \\ (+)\ 0011\ 0011 \\ \hline 0100\ 0111 \\ \hline \Downarrow \quad \Downarrow \\ 1 \quad 4 \end{array}$$

Excess 3 value is 14

$$\begin{array}{r} 0100 \\ 0101 \\ \hline \text{No carry } 1001 \\ \hline -\ 0011 \\ \hline 0110 \\ \hline \Downarrow \\ 3. \end{array}$$

Excess 3 subtraction.

(1) 8 − 5

1011 → Excess 3 of 8
0111 → complement of 5 in Excess-3

$$\begin{array}{r} 1011 \\ 0111 \\ \hline \text{Carry } 1\,0010 \\ 0011 \\ \hline 0101 \\ 1 \\ \hline 0110 \\ \hline \end{array}$$

0110 ⇒ Excess 3 for 3.

(b) 5 − 8

Excess 3 for 5   1000

Complement of Excess 3 for 8   0100

```
            1000
            0100
           ------
            1100
  No Carry  0011
           ------
            1001    Excess 3 to -3.
```

Perform the operation in excess 3 code.

(a) 16 + 29   (b) 21 − 12

a) 16 + 29

```
          No carry    carry
              1
 16 => 0100  1001
 29 => 0101  1100
      ----------------
       1010  0101  (+)
 (−)   0011  0011
      ----------------
       0111  1000
        4      5
```

Excess 3 → 45.

```
(b) 21 => 0101  0100
    12 => 1011  1010     Complement of Excess 3.
          ----- No carry
         10000  1110
          0011 (−) 0011
         -----------------
          0011  1011
                   1    end around carry
         -----------------
          0011  1100

           0     9   → Excess 3 for 9.
```

Binary Arithematic:

Addition:

0 + 0 = 0 ; 1 + 0 = 1

0 + 1 = 1 ; 1 + 1 = 0 with a carry 1

Example:

1101.101 and 111.011

$$\begin{array}{r} 1101.101 \\ (+)\quad 111.011 \\ \hline 10101.000 \\ \hline \end{array}$$

Subtraction:

0 − 0 = 0 ; 1 − 1 = 0

1 − 0 = 1 ; 0 − 1 = 1 with borrow 1.

Example:

Subtract 111.111_2 from 1010.01_2.

$$\begin{array}{r} 10\,10\,10\,\;10\,10\,10 \\ \cancel{1}\cancel{0}\cancel{1}\cancel{0}.\cancel{0}\cancel{1}\cancel{0} \\ (-)\quad 111.111 \\ \hline 0010.011 \\ \hline \end{array}$$

Multiplication:

0 × 0 = 0 ; 0 × 1 = 0.

1 × 0 = 0 ; 1 × 1 = 1.

Example:

multiply 1101_2 by 110_2

$$\begin{array}{r} 1101 \times 110 \\ \hline 000 \\ 1101 \\ 1101 \\ \hline 1001110 \end{array}$$

2. multiply 1011.101_2 by 101.01_2.

$$\begin{array}{r} 1011.101 \times 101.01 \\ \hline 1011101 \\ 0000000 \\ 1011101 \\ 0000000 \\ 1011101 \\ \hline 111101.00001 \\ \hline \end{array}$$

Division:

Divide 101101_2 by 110_2.

$$\begin{array}{r} 110\ )\ 101101\ (111.1 \\ 110 \\ \hline 1010 \\ 110 \\ \hline 1001 \\ 110 \\ \hline 110 \\ 110 \\ \hline 0 \\ \hline \end{array}$$

$101101 \div 110 = 111.1$.

Divide 110101.11_2 by 101_2

$$\begin{array}{r} 101\ )\ 110101.11\ (1010.11 \\ 101 \\ \hline 0110 \\ 101 \\ \hline 111 \\ 101 \\ \hline 101 \\ 101 \\ \hline 0 \end{array}$$

$(110101.11)_2 \div (101)_2 =$

$(1010.11)_2$

Representation of signed numbers.

There are two ways of representing signed numbers

1. Sign magnitude format 2. Complement form.

Complement form.

1. 1's Complement 2. 2's Complement.

* Most of the digital computers do subtraction by the 2's Complement or 1's Complement.

Advantage: reduction in hardware.

* Instead of performing subtraction we can perform addition by 1's Complement or 2's Complement.

* Instead of subtracting. Complement the subtrahend is added to minuend.

Sign - Magnitude form:

An additional bit called the sign bit is placed in front of the number

If a sign bit is 0, number is positive.
If it is 1, number is negative.

0	1	0	1	0	0	1

Sign bit Magnitude

= +41

1	1	0	1	0	0	1

Sign bit Magnitude

= -41

Representation of signed numbers using 2's or 1's Complement method.

1. If the number is positive, the magnitude is represented in its binary form and a sign bit 0 is placed in front of the MSB.

Example.

Represent +51 and -51 in 2's Complement

and 1's Complement.

51 = 33

+51	0	1	1	0	0	1	1	
−51	1	1	1	0	0	1	1	

2's Complement of −51

1	0	0	1	1	0	1	

1's Complement of −51

1	0	0	1	1	0	0	

Binary Arithmetic for Signed Numbers.

Complements :

The main advantage of this representation, is that we can use single electronic circuit for addition and subtraction.

Types of Complement representation.

1. 1's Complement
2. 2's Complement

} Binary System.

3. 9's Complement
4. 10's Complement

} Decimal system.

Complement representation in Binary :

(1) 2's Complement.

2's complement of $x = 2^n - x$.

Example

(i) 0100 in 2's Complement form.

$$
\begin{array}{lcr}
2^n & = & 1\ 0000\ 0000 \\
(0100)\ 'A' & = & 0000\ 0100 \\
\hline
 & & 1111\ 1100 \\
\hline
\end{array}
\quad (-)
$$

(ii) Find 2's Complement of 1011.

$$
\begin{array}{lcr}
2^8 & = & 1\ 0000\ 0000 \\
 & = & 0000\ 1011 \\
\hline
 & & 1111\ 0101 \\
\hline
\end{array}
\quad (-)
$$

1's Complement

The given binary number is subtracted from $2^n - 1$

$2^n - 1 = 11111111$

Example:

(i) find 1's Complement of 0100.

$$
\begin{array}{rl}
2^8 - 1 = & 1111\,1111 \\
= & 0000\,0100 \quad (-) \\
\hline
& 1111\,1011 \\
\hline
\end{array}
$$

Another easier way of converting binary to 1's Complement form is changing the 1's to zero to 0's to Ones.

(ii) find 1's Complement of 0100.

(1011) 1's Complement form.

Method 2:

2's Complement from 1's Complement.

* find 1's Complement from the given binary and add 1 to the 1's Complement Value.

i.e., Example.

Binary = 0100

1's complement of 0100 = 1011

$$
\begin{array}{rr}
\text{2's Complement of } 0100 = & 1011 \\
& 1 \\
\hline
& 1100 \\
\hline
\end{array}
$$

Example: 2

Express −19 in 2's complement form.

19 = 00010011

2^8 = 1 00000000

$$2^8 - x = \begin{array}{r} 1\ 0000\ 0000 \\ 0001\ 0011 \\ \hline 1110\ 1101 \end{array}$$

Method 2:

19 ⇒ 0001 0011

1's complement of 19 ⇒ 1110 1100

Add 1   1

2's complement of 19.   1110 1101

Method 3

19 = 0001 0011

1110 1101   2's complement of 19

Binary addition by 1's complement:

1. For adding two positive numbers of n bits sum should not exceed $2n$. If it exceeds, the addition gives an errornous result.

2. Add the two numbers with sign bit, including the sign bit in the addition.

Example:

Add +7 and +9

addition of these two digits gives more than five bits so take $n = 8$

0000 0111	+7
0000 1001	+9
0001 0000	+16.

Subtraction of numbers by 1's Complement.

1. Convert the negative numbers to their 1's Complement form, leaving the sign as 1.
2. Add to produce sum.
3. If there is a carry generated bring it round and add it to LSB of the sum. The sum is positive.
4. If there is no carry, the answer is a negative number in 1's complement form.
5. Reconversion is to be done to get the original answer.

Example:

Subtract +2 from +9.

Binary equivalent of 9 = 0 1001

Binary equivalent of -2 = 1 0010

1's Complement of -2 = 1, 1101

0 1001

Add 9, -2 = 1 0 0 1 1 0

Carry 1

0,0111

Answer is +7

2. Find the solution of -9, +4

Binary equivalent of -9 ⇒ 1, 1001

Binary equivalent of +4 ⇒ 0 0100

1's Complement of -9 ⇒ 1 0110

Add 4 ⇒ 0 0100

1, 1010

No carry is generated, Answer has to Complemented

1, 0101 = -5

3. Add −8 and −9.

Sum may exceed the no. of bits so choose

n = 5.

Binary equivalent of −8 = 101000

Binary equivalent of −9 = 101001

1's Complement of −8 = 110111

1's Complement of −9 = 110110 (+)

Add ①,10110 1

Carry 1

1,01110

1's complement of the Answer gives

= 1, 10001

= −17.

2's Complement arithemetic

The disadvantage of 1's complement is say +4 and −4. In decimal it is equal to zero.

In 1's complement

+4	00100
−4	10100
1's Complement of −4	11011
Add	00100
	1,1111

Answer is 'zero'

1's complement ⇒ there is two representation for zeros, this is a decided disadvantage.

(2) The second one is the necessity of end around carry which needs another addition

Same addition in 2's complement

Binary equivalent of +4 = 0,0100
1's complement of −4 = 1 1011

1,1111

Add '1'   1

1 0,0000

Addition and subtraction using 2's complement.

* The addition of two positive numbers is the same as in 1's complement.
* One positive number and one negative numbers are added the Result may be positive or negative.
* such operation ignores the overflow.
* Sign bit is treated as the part of the number.

Example:

(1) Add +9 and −7 using 2's complement.

Let n = 8, 8^{th} bit shows the sign.

Binary equivalent of +9 = 00001001

Binary equivalent of −7 = 10000111

2's complement of −7 = 11111001

Add +9 = 00001001

1 0.0000010 ← discard the carry

1 0000111
1111000
1
1 1111001

sum is $(00000010)_2$

= +2

(2) Add −8 and +10 using 2's complement.

Binary equivalent of −8 = 10001000

2's complement of −8 is = 11111000

Binary of +10 = 00001010

1 00000010 ← discard carry

1's complement → 1 1110111
1
1 1111000

$(00000010)_2$

= 2

(3) Add +16 and -19.

Binary equivalent of 16 = 0001 0000.

Binary equivalent of -19 = 1001 0011

1's complement of -19 = 1 1 1 0 1 1 0 0 + 1

2's complement of -19 = 1 1 1 0 1 1 0 1

Add 16 = 0 0 0 1 0 0 0 0

1, 1 1 1 1 1 0 1

Sign bit is 1 result is -ve

* To get the original answer result has to be converted to 2's complement form.

i.e.,

1, 1 1 1 1 1 0 1

1's complement 1 0 0 0 0 0 1 0
1

2's complement 1, 0 0 0 0 0 1 1 ⇒ -3

(A) Add -10 with -20.

Binary equivalent of -10 = 1 000 1010

Binary equivalent of -20 = 1 0010100

2's complement of -10.

-10 → 1's complement = 1 1 1 1 0 1 0 1
1

① ← 2's complement of -10 = 1 1 1 1 0 1 1 0

2's complement of -20

1's complement of -20 = 1 1 1 0 1 0 1 1

Add 1   1

② ← 2's complement of -20 1 1 1 0 1 1 0 0

Add ① + ②

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

discard it ← carry 1 1 1 1 0 0 0 1 0

To get Correct Answer find the 2's Complement of the Answer.

i.e, 1100010

$$\Rightarrow \begin{array}{r} 1001110\,1 \\ 1 \\ \hline 10011110 \\ \hline \end{array}$$

result is −30

Points to Remember:

1) Add 2 positive numbers, carry is obtained and ignored.

2) One Positive and One negative numbers are added. +ve results Carry is obtained and ignored.

-ve result → no carry, the result is Converted to 2's complement form.

3) two negative numbers are added, the result is negative, carry is generated, Carry is ignored and the result Converted to 2's Complement form.

9's and 10's Complement:

* Subtraction of decimal numbers accomplished by 9's and 10's complement.

* 9's complement → subtracting each digit by 9
* 10's complement → add 1 to the 9's Complement

Example:

Find 9's Complement of a) 3465 b) 782.54 c) 4526.075.

a)
$$\begin{array}{r} 9999 \\ 3465 \\ \hline 6534 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 999.99 \\ 782.54 \\ \hline 217.45 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 9999.999 \\ 4526.075 \\ \hline 5473.924 \\ \hline \end{array}$$

Find 10's complement of the following decimal number

a) 4069 b) 1056·074.

a)

$$\begin{array}{r} 9999 \\ 4069 \\ \hline 5930 \end{array} \rightarrow \text{9's complement}$$

$$\begin{array}{r} (+)\ 1 \\ \hline 5931 \end{array} \rightarrow \text{10's Complement}$$

b)

$$\begin{array}{r} 9999.999 \\ 1056.074 \\ \hline 8943.925 \end{array} \rightarrow \text{9's Comp}$$

$$\begin{array}{r} 1 \\ \hline 8943.926 \end{array} \rightarrow \text{10's Comp}$$

9's complement subtraction:

The negative number is converted to 9's Comp and added to the other number.

If carry exists, added to the number and the result is +ve.

If carry not-exists, the answer is -ve and the result is converted to 9's complement to get the correct result.

Example!

Subtract the following numbers using the 9's complement method.

a) 745·81 − 436·62, b) 436·62 − 745·81.

a) 745·81 − 436·62,

9's complement of − 436·62

$$\begin{array}{r} 999.99 \\ 436.62 \\ \hline 563.37 \end{array}$$

$$\begin{array}{r} 745.81 \\ 563.37 \\ \hline 1309.18 \\ 1 \\ \hline 309.19 \end{array}$$

Carry

If carry exists the Answer is positive.

b) 436·62 − 745·81.

9's complement of 745·81 ⇒

```
  999·99
  745·81
  ------
  254·18
```

```
  436·62
  254·18
  ------
  690·80
```

→ No carry exist, Answer is negative.

9's complement of the Answer is

```
  999·99
  690·80
  ------
  309·19
```

The Result is − 309·19.

10's Complement method of subtraction.

Example:

Subtract the following number by using 10's complement method.

a) 2928·54 − 416·73   b) 416·73 − 2928·54.

(a) 2928·54 − 416·73.

10's complement of 416·73

```
  9999·99
  0416·73
  -------
  9583·26
        1
  -------
  9583·27
```

```
   2928·54
   9583·27
   -------
  12511·81
```

carry ignored

Answer is 2511·81.

---

b) 416·73 − 2928·54

10's complement of 2928·54 is

```
  9999·99
  2928·54
  -------
  7071·45
        1
  -------
  7071·46
```

```
   416·73
  7071·46
  -------
  7488·19
```

→ No Carry Answer is −ve

Result is −2511·81.

find 10's Complement

```
  9999·99
  7488·19
  -------
```

```
  2511·80
        1
  -------
  2511·81
```

Binary logic functions:

Binary logic deals with variables that take place on two discrete values with operations that assume logical meaning. The two values is terms of bits are 0 and 1.

Binary logic is used to describe in a mathematical way, the manipulation and processing of binary information.

Binary logic consists of binary variable and logic operators. The variables are described by letters of alphabet such as A, B, C and x, y, z with each variables assigned 0 or 1

The logic operations are

1. AND 2. OR 3. NOT.

AND: This operator is represented by a dot or by the absence of an operator.

Example:

$$x \cdot y = z \quad (\text{or}) \quad xy = z$$

* It must be read as x and y is equal to z
* It means that $z = 1$ if $x = 1$ & $y = 1$

otherwise $z = 0$

OR: This operator is represented by a plus sign.

Example: $x + y = z$ is read as x or y is equal to z.

* meaning is that $z = 1$ if $x = 1$ or $y = 1$ (or) if both $x = 1$ and $y = 1$.

* If both $x = 0$ & $y = 0$ then $z = 0$.

NOT: This operator is represented by a bar.

Example: $x' = z$ (or) $\bar{x} = z$

* It is read as "not x is equal to z" meaning that z is what x is not.

i.e., if $x = 1$; $z = 0$
$x = 0$; $z = 1$.

Truth table :

A truth table is a table of all possible combinations of the variables showing the relation between the values that the variables may take and the result of the operation.

AND

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

OR

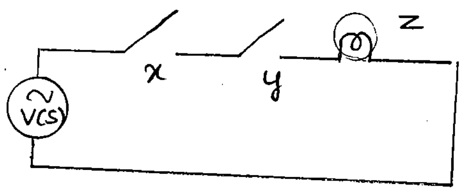
x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

NOT

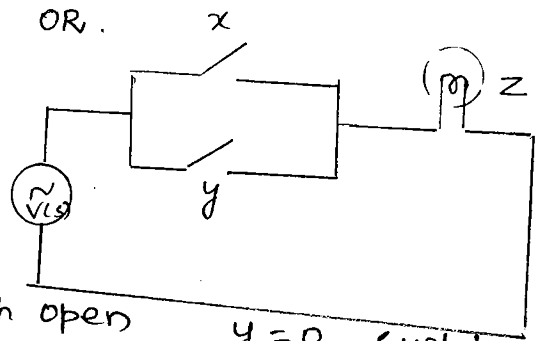
x	z
0	1
1	0

Switching Circuits :

AND.



OR.



$x \rightarrow$ switch
$y \rightarrow$ switch
$z \rightarrow$ lamp

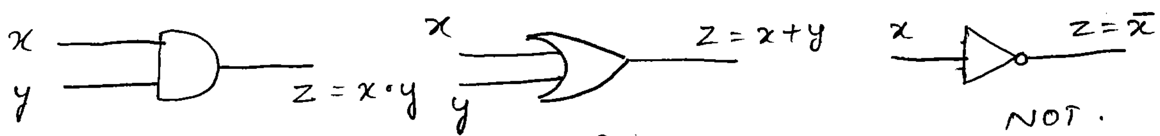
$x = 0$ switch open
$x = 1$ switch close
$z = 1$ lamp glows

$y = 0$ switch open
$y = 1$ switch close.
$z = 0$ lamp not glow.

Logic gates :

Digital circuits are called as Logic circuits because with proper i/p they establish manipulation path. These circuits are called as gate or logic gate. They are blocks of hardware that produce a logic 1 or logic 0 o/p signal if input logic requirements are satisfied.

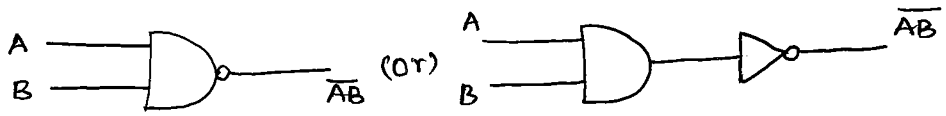
Symbols :



AND OR NOT.

Derived gates:

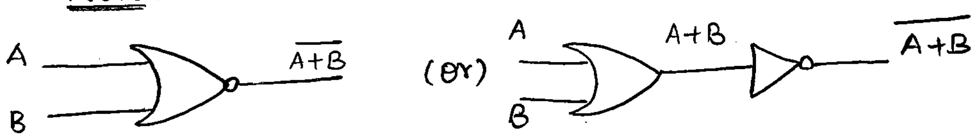
NAND:



Truth table:

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

NOR:

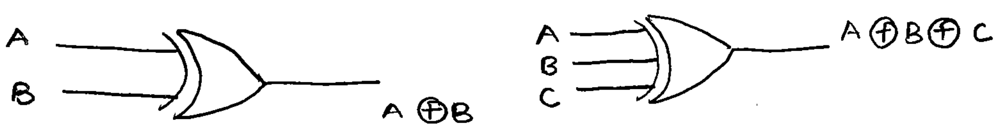


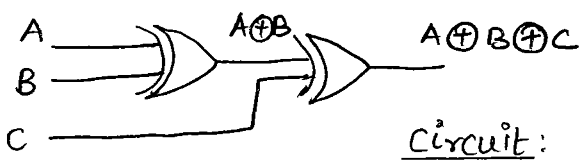
Truth table:

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

Exclusive - OR (XOR) gate:

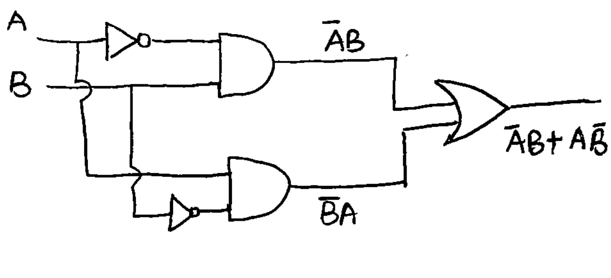
In the XOR operation either A or B but not both should be high to produce high o/p.





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

Circuit:



Truth table:

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Proof.

1. $A=0, B=0$

$\bar{A}B = 1.0 = 0$; $\bar{B}A = 1.0 = 0$

$\bar{A}B + \bar{B}A = 0+0 = 0$

2. $A=0, B=1$

$\bar{A}B = 1.1 = 1$ $\bar{B}.A = 0.1 = 0$

$\bar{A}B + \bar{B}A = 1+0 = 1.$

3. $A=1, B=0$

$\bar{A}B = 0.0 = 0$ $\bar{B}A = 1.1 = 1$

$\bar{A}B + \bar{B}A = 1+0 = 1.$

4. $A=1 ; B=1$

$\bar{A}B = 0.1 = 0$ $\bar{B}A = 0.1 = 0$

$\bar{A}B + \bar{B}A = 0+0 = 0.$

Application:

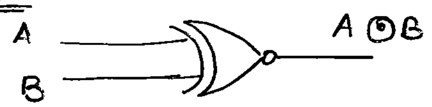
used for parity generation and checking.

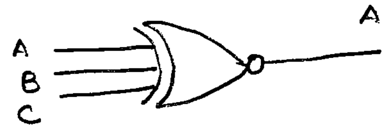
Exclusive NOR or XNOR.

This can be regarded as XOR followed by an inverter. Here either A or B should be low to produce high o/p.

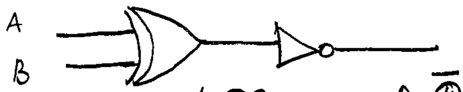
operator symbol $\odot$ or $\overline{\oplus}$

Symbol:





Logic circuit:



$A \oplus B$ $A \overline{\oplus} B$ or $A \odot B$

Truth table:

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Proof:

1. $A=0 ; B=0$; XOR $= 0$ XNOR $= 1$
2. $A=0 ; B=1$; XOR $= 1$ XNOR $= 0$
3. $A=1 ; B=0$; XOR $= 1$ XNOR $= 0$.
4. $A=1 ; B=1$ XOR $= 0$ XNOR $= 1$.

Application: Error detection of data during transmission and distribution.

Switching algebra: * algebra is conveniently used to describe the operation of complex networks of digital circuits.

Boolean law:

* Boolean laws have made it possible to design and analyse logic circuit mathematically.

* It has set of elements, operators and a number of axioms or postulates.

Definitions:

Binary operator for the set of elements is a rule which produces a fixed output from the given element. The condition is that the output should be an element of the set.

* Designers of digital systems use Boolean algebra to transform circuit diagram to expressions

Postulates:

These are the basic assumptions from which it is possible to deduce the rules, theorems and properties of the system.

1. Closure:

A set S is said to be closed if all the outputs of the elements of the set with respect to a binary operator are element of S.

Example: If $A, B \in S$

$C = A + B$, $C \in S$ is true.

2. Associative:

A binary operator '+' or '*' on a set S is said to be associative if

$$(x * y) * z = x * (y * z)$$

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

3. Commutative:

$$x + y = y + x$$

$$xy = y \cdot x$$

4. Identity element:

$$I * x = x * I = x.$$
$$I + x = x + I = x.$$

here $I = 1$ for operator $*$

$I = 0$ for operator $+$.

5. Inverse:

$$x * y = E$$
$$x + (-x) = E.$$

6. Distributive:

$$x + (y \cdot z) = (x+y) \cdot (\cancel{x} + z)$$
$$x \cdot (y+z) = xy + xz.$$
$$x + (\bar{x} \cdot y) = x + y.$$

Boolean theorems:

1. Single variable theorem.
2. multivariable theorem.

Single variable theorem:

These include AND, OR and NOT operations with a single input variable.

AND laws.

Law of intersection.

$x \cdot 0 = 0$	$x = 1 = 1 \cdot 0 = 0$
$x \cdot 1 = x$	$x = 0 = 0 \cdot 0 = 0$
$x \cdot x = x$	$x = 1 = 1 \cdot 1 = 1$
$x \cdot \bar{x} = 0$	$x = 0 = 1 \cdot 0 = 0$
	$x = 1 = 1 \cdot 1 = 1$
	$x = 0 = 0 \cdot 0 = 0.$
	$x = 1 = 1 \cdot 1 = 1$
	$x = 0 = 0 \cdot 1 = 0.$
	$x = 1 = 1 \cdot 0 = 0.$
	$x = 0 = 0 \cdot 1 = 0.$

OR Laws

Law of Union:

$$x + 0 = x$$
$$x + 1 = 1$$
$$x + x = x$$
$$x + \bar{x} = 1$$

Multivariable theorem:

These are theorems involving more than one variable.

Commutative laws.

$$x + y = y + x$$
$$x \cdot y = y \cdot x.$$

Proof:

x	y	$x+y$	xy	$y \cdot x$	$y+x$
0	0	0	0	0	0
0	1	1	0	0	1
1	0	1	0	0	1
1	1	1	1	1	1

Associative law:

$$x + (y+z) = (x+y) + z = x + y + z$$
$$x(yz) = (xy)z = xyz.$$

Distributive law:

$$x(y+z) = xy + xz$$
$$(w+x)(y+z) = wy + wz + xy + xz.$$

Proof.

x	y	z	xy	xz	$y+z$	$x(y+z)$	$xy+xz$
0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	1	0	0
1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	1

x	y	z	xy	xz	$y+z$	$x(y+z)$	$xy+xz$
1	1	0	1	0	1	1	1
1	1	1	1	1	1	1	1

* $(w+x)(y+z) = wy + xy + wz + x\cdot z$
* $x + yz = (x+y)(x+z)$.

<u>Proof:</u>

$$= x\cdot x + y\cdot x + x\cdot z + y\cdot z.$$

$$= x(1+y+z) + y\cdot z \qquad \because y+1 = 1$$

$$= x + yz \qquad \because 1+z = 1.$$

<u>Law of absorption:</u>

1. $x(x+y) = x$.

<u>Proof</u>: $= x\cdot x + x\cdot y$

$= x + x\cdot y$

$= x(1+y)$

$= x$.

2. $x + xy = x$.

<u>Proof</u>: $= x(1+y)$

$= x\cdot 1$

$= x$.

(RL·R) 3. $x(\bar{x}+y) = x\cdot y$. (Redundant literal rule)

$= x\cdot\bar{x} + x\cdot y \qquad \because x\cdot\bar{x} = 0.$

$= x\cdot y$.

4. $xy + \bar{y} = x + \bar{y}$

$= x\cdot y + \bar{y}(x+1)$

$= xy + \bar{y}x + \bar{y} \qquad \because y+\bar{y} = 1.$

$= x(y+\bar{y}) + \bar{y}$

$= x + \bar{y}$.

5. $x\bar{y} + y = x + y$ (RLR) (Redundant literal rule)

$$= x\bar{y} + y(x+1)$$

$$= x\bar{y} + yx + y$$ $\because y + \bar{y} = 1$

$$= x(\bar{y} + y) + y$$ $\because x + 1 = 1$

$$= x + y$$

Law of involution:

$$\bar{\bar{x}} = x.$$

Consensus law:

Theorem 1: $xy + \bar{x}z + yz = xy + \bar{x}z.$

Proof:

$$xy + \bar{x}z + yz = xy + \bar{x}z + yz(x + \bar{x}).$$

$$= xy + \bar{x}z + xyz + \bar{x}yz.$$

$\because 1 + y = 1$

$1 + z = 1$

$x + \bar{x} = 1.$

$$= xy(1+z) + \bar{x}z(1+y).$$

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

Theorem 2:

$$(x+y)(\bar{x}+z)(y+z) = (x+y)(\bar{x}+z).$$

Proof:

$$(x+y)(\bar{x}+z)(y+z) = x\bar{x} + xz + \bar{x}y + yz \; (y+z)$$

$x \cdot \bar{x} = 0 \;\because$

$x + \bar{x} = 1$

$x + x = x$

$$= (xz + \bar{x}y + yz)(y+z)$$

$$= xyz + xz \cdot z + \bar{x}y \cdot y + \bar{x}y \cdot z + yz \cdot y + yz \cdot z.$$

$$= xyz + y\bar{x} + yz + \bar{x}yz + yz + x \cdot z.$$

$$= \bar{x}y + yz(x + \bar{x}) + yz + x \cdot z$$

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

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

Demorgan's law:

$$\overline{x+y} = \bar{x}\cdot\bar{y}$$

x	y	$x+y$	$\overline{x+y}$	$\bar{x}$	$\bar{y}$	$\bar{x}\cdot\bar{y}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0.

Thus

$$\overline{x+y} = \bar{x}\cdot\bar{y}$$

(or)

$$\overline{x\cdot y} = \bar{x}+\bar{y}.$$

Reduction of switching equation by Boolean algebra.

1. $\overline{(\bar{A}\cdot B)\,(B\cdot C)\,(C\cdot\bar{D})}$

Solution:

$\overline{A\cdot B} = \bar{A}+\bar{B}$

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

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

$= A+\bar{B}+\bar{C}+D.$

---

2. $\overline{(\bar{A}\cdot B\cdot\bar{C})} + \overline{(A\cdot\bar{B}\cdot C)}$

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

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

$= 1+1+1$

$= 1+1$

$= 1$

$\because A+1 = 1$

$1+1 = 1$

$A+\bar{A} = 1$

---

3. $(A\cdot B)+(B+C)$

$= (A+1)B + C$

$= B+C$

$\because A+1=1$

---

4. $\overline{A\cdot B}\,(A+B)$

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

$= A\cdot\bar{A} + B\cdot\bar{B} + A\cdot\bar{B} + \bar{A}\cdot B$

$= A\bar{B} + \bar{A}B.$

$\because A\cdot\bar{A} = 0$

$\because B\cdot\bar{B} = 0$

5. $C(B+C)(A+B+C)$

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

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

$= C(B+1)(A+B+C)$ $\quad B+1 = 1 \quad \because$

$= C(A+B+C)$ $\quad C \cdot C = C \quad \because$

$= AC + BC + C \cdot C$

$= AC + BC + C$

$= C(A+B+1)$

$= C(A+1)$

$= C$

---

6. $(A+\bar{B}+\bar{C})(A+\bar{B}+C)$

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

$= A + A\bar{B} + A(C+\bar{C}) + \bar{B}A + \bar{B} + \bar{B}C + \bar{C}\bar{B} + 0$ $\quad C \cdot \bar{C} = 0, \quad A \cdot A = A, \quad A \cdot \bar{A} = 0$

$= A(1 + \bar{B} + \bar{B}) + A + \bar{B}(C+1) + \bar{C}\bar{B}$ $\quad C + \bar{C} = 1$

$= A(1 + \bar{B} + \bar{B}) + A + \bar{B}(1 + \bar{C})$

$= A(1) + A + \bar{B}$

$= A + \bar{B}$

---

7. $A \cdot \bar{B} \cdot \bar{C} + \bar{A} \cdot B \cdot \bar{C} + \bar{A} \cdot B \cdot C$

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

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

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

8. $(B+BC)(B+\bar{B}C) * (B+D)$

$= (B \cdot B + B \cdot \bar{B}C + B \cdot BC + BC \cdot \bar{B}C) * (B+D)$

$= (B + BC)(B+D)$ $\quad D+1 = 1$

$= B \cdot B + B \cdot D + B \cdot B \cdot C + BC \cdot D$ $\quad A \cdot A = A, \quad C + \bar{C} = 1$

$= B + B \cdot D + BC + BCD$

$= BD[1+C] + B[1+C]$

$= BD + B$

$= B(1+D)$

$= B.$

Show that

$$A\cdot\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C = B + C.$$

LHS

$A\cdot\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C$

$= A\bar{B}C + B[1 + A\bar{D}] + B\bar{D} + \bar{A}C$

$= A\bar{B}C + B(1+\bar{D}) + AB\bar{D} + \bar{A}C.$

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

$= A\bar{B}C + B(1 + A\bar{D}) + \bar{A}C$

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

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

$= \bar{B}C[A+\bar{A}] + B[1+\bar{A}C]$

$= \bar{B}C + B$ $\quad\because x + \bar{x}y = x + y.$

$= B + C$

$= RHS.$

Simplify

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

$= (A\cdot\bar{A} + A\cdot C + \bar{A}\cdot B + B\cdot C)(\bar{B}+\bar{C})$ $\quad A\cdot\bar{A} = 0$

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

$= A\bar{B}C + \underbrace{\bar{A}B\cdot\bar{B}}_{0} + \underbrace{B\cdot\bar{B}C}_{0} + A\cdot \underbrace{C\cdot\bar{C}}_{0} +$
$\bar{A}B\bar{C} + B\cdot \underbrace{C\cdot\bar{C}}_{0}.$

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

Principle of Duality:

The duality theorem says that, starting with a Boolean relation, you can drive another boolean relation by.

1. changing each OR sign to an AND sign.
2. changing each AND sign to an OR sign.
3. Complementing any 0 or 1 appearing in the expression.

$$A+0 = A$$
$$A\cdot 1 = A$$

S.NO.	given expression	Dual.
1.	$\bar{0} = 1$	$\bar{1} = 0$
2.	$0\cdot 1 = 0$	$1+0 = 1$
3.	$0\cdot 0 = 0$	$1+1 = 1$
4.	$1\cdot 1 = 1$	$0+0 = 0$
5.	$A\cdot 0 = 0$	$A+1 = 1$
6.	$A\cdot 1 = A$	$A+0 = A$
7.	$A\cdot A = A$.	$A+A = A$.
8.	$A\cdot\bar{A} = 0$	$A+\bar{A} = 1$
9.	$A\cdot B = B\cdot A$	$A+B = B+A$.
10.	$A\cdot(B\cdot C) = (A\cdot B)\cdot C$	$A+(B+C) = (A+B)+C$
11.	$A\cdot(B+C) = AB+AC$	$A+(BC) = =(A+B)(A+C)$
12.	$A(A+B) = A$	$A+AB = A$
13.	$A\cdot(A\cdot B) = A\cdot B$	$A+A+B = A+B$.
14.	$\overline{AB} = \bar{A}+\bar{B}$	$\overline{A+B} = \bar{A}\bar{B}$
15.	$(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$	$AB+\bar{A}C+BC = AB+\bar{A}C$.
16.	$(A+C)(\bar{A}+B) = AB+\bar{A}C$	$AC+\bar{A}B = (A+B)(\bar{A}+C)$
17.	$A+\bar{B}C = (A+\bar{B})(A+C)$	$A(\bar{B}+C) = A\bar{B}+AC$
18.	$(A+B)(C+D) = AC+BC+AD+BD$	$(AB+CD) = (A+C)(B+C)(A+D)(B+D)$
19.	$A+B = AB+\bar{A}B+A\bar{B}$	$AB = (A+B)(\bar{A}+B)(A+\bar{B})$
20	$\overline{\overline{AB}+\bar{A}+AB} = 0$	$\overline{\overline{A+B}\cdot\bar{A}\cdot(A+B)} = 1$

Law of Transposition:

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

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

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

RHS

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

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

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

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

$= AB(1+C) + \bar{A}C(1+B)$

$= AB + \bar{A}C$

(ii) $(A+B)\cdot(\bar{A}+C) = AC + \bar{A}B$.

LHS

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

$= AC + B\bar{A} + BC(A+\bar{A})$

$= B\bar{A} + AC + ABC + \bar{A}BC$

$= AC(1+B) + \bar{A}B(1+C)$

$= AC + \bar{A}B$

$=$ RHS

Extension of Demorgan's law:

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

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

can be extended to complicated expressions.

1. Complement the entire given function.
2. Change all the AND's to OR's and all the OR's to AND's.
3. Complement each of the individual Variables.
4. Change all 0's to 1's and 1's to 0's.

This procedure is called demorganization or Complementation of switching expressions.

Example:

1. Demorganise $f = \overline{(A+\bar{B})(C+\bar{D})}$

Solution.

$= (A+\bar{B})(C+\bar{D})$ Complementing entire function.

$= A\bar{B} + C\bar{D}$ OR to AND, AND to OR.

$= \bar{A}\cdot B + \bar{C}\cdot D$ Complement the Variables.

2. Apply Demorgan's theorem to the expression.

$$f = \overline{\overline{AB}(CD+\overline{EF})(\overline{AB}+\overline{CD})}$$

$$= \overline{\overline{AB}} + \overline{CD+\overline{EF}} + \overline{\overline{AB}+\overline{CD}}$$

$$= AB + \overline{CD}\cdot\overline{\overline{EF}} + \overline{\overline{AB}}\cdot\overline{\overline{CD}}$$

$$= AB + (\bar{C}+\bar{D})(E+F) + AB\cdot CD$$

3. Reduce the expression.

$$f = \overline{\overline{AB} + \bar{A} + AB}$$

1^{st} step : Break the line and change the sign.

$$= \overline{\overline{AB}} \cdot \bar{\bar{A}} \cdot \overline{AB}$$

$$= AB \cdot A \cdot \overline{AB}$$

$$= 0$$

(or)

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

$$= \bar{\bar{A}} \cdot \bar{\bar{B}} \cdot \overline{AB}$$

$$= AB \cdot \overline{AB}$$

$$= 0.$$

---

4. Reduce the expression

$$F = A\left[B + \bar{C}\left(\overline{AB + A\bar{C}}\right)\right]$$

De morganize $\overline{AB + A\bar{C}}$ $= A\left[B + \bar{C}\left(\overline{AB} \cdot \overline{A\bar{C}}\right)\right]$

De morganize $\overline{AB} \cdot \overline{A\bar{C}}$ $= A\left[B + \bar{C}\ (\bar{A} + \bar{B})(\bar{A} + C)\right]$

$$= A\left[B + \bar{C}\left[\bar{A}\cdot\bar{A} + \bar{A}\cdot C + \bar{A}\bar{B} + \bar{B}C\right]\right]$$

$$= A\left[B + \bar{C}\left[\bar{A}\right] + \bar{C}\bar{A}C + \bar{A}\bar{B}\bar{C} + \bar{B}C\bar{C}\right]$$

$$= A\left[B + \bar{C}\bar{A} + \bar{A}\bar{B}\bar{C}\right]$$

$A \cdot \bar{A} = 0$
$1 + \bar{B} = 1$

$$= A\left[B + \bar{A}\bar{C}\left[1 + \bar{B}\right]\right]$$

$$= AB + A\bar{A}\bar{C}$$

$$= AB.$$

---

5. Redue the expression.

$$F = A + B\left[AC + (B + \bar{C})D\right]$$

$$= A + B\left[AC + BD + \bar{C}\bar{D}\right]$$

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

$$= A(1 + BC) + BD + B\bar{C}D$$

$$= A + BD(1 + \bar{C}).$$

$$= A + BD.$$

6. Reduce the expression.

$$F = (\overline{A + \overline{BC}})(A\overline{B} + ABC).$$

Demorganice $\overline{A+\overline{BC}} = (\overline{A} \cdot \overline{\overline{BC}})(A\overline{B} + ABC).$

$$= (\overline{A}BC)(A\overline{B} + ABC)$$

$$= \overline{A}AB\overline{B}C + A\overline{A}B \cdot B \cdot C \cdot C.$$

$$= 0 + 0$$

$$= 0.$$

---

7. Show that $AB + A\overline{B}C + B\overline{C} = AC + B\overline{C}.$

LHS

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

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

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

$$= AB(C+\overline{C}) + AC + B\overline{C}$$

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

$$= AC(1+B) + B\overline{C}(1+A)$$

$$= AC + B\overline{C}.$$

$B \cdot B + \overline{B} \cdot B + BC + \overline{B}C.$

$B(B+\overline{B}) + BC + \overline{B}C$

$B + BC + \overline{B}C$

$B(1+C) + \overline{B}C$

$B + \overline{B}C$

8. Simplify $F = (A+B)(A+\overline{C}) + \overline{A}\overline{B} + \overline{A}\overline{C}.$

$$= A \cdot A + A \cdot \overline{C} + A \cdot B + B \cdot \overline{C} + \overline{A}\overline{B} + \overline{A}\overline{C}.$$

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

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

$$= A(1+B) + \overline{C}(1+B) + \overline{A}\overline{B}$$

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

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

9) Simplify using Consensus theorem.

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

$AB + \overline{A}C = AB + \overline{A}C + BC$

Functionally Complete sets of Operations:

* OR, AND and NOT (+, •, –) form a functionally Complete set in the sense that any function Can be realised using these operators in SOP or POS form.

SOP → Sum of product.
POS → product of Sum.

* with the help of De-morgan's theorem it is possible to produce $A \cdot B$ using only the set of Operators (+, –).
* Likewise $A+B$ can be realised using only the operators (•, –).
* These sets, each Containing two operators only (•, –) or (+, –) are said to be functionally Complete sets.
* Further using only the NAND operator or the NOR operator, it is possible to produce all the Boolean operations.
* Hence each one of them forms a functionally Complete single element set.

Realisation of Switching function (or) Boolean function.

A function of n Boolean variables denoted by $f(x_1, x_2, x_3 - - x_n)$ is another variable of algebra and takes one of the two possible values, 0 and 1..

The various ways of representing the function is given as.

1. SOP (Sum of Product) form

2. POS (Product of sum) form.
3. Truth table.
4. Standard SOP form.
5. Standard POS form.
6. Venn diagram form.
7. Karnaugh map

1. Sum of product (SOP) form :

Definition

Property

Two's complement representation allows the use of binary arithmetic operations on signed integers, yielding the correct 2's complement results.

Positive Numbers

Positive 2's complement numbers are represented as the simple binary.

Negative Numbers

Negative 2's complement numbers are represented as the binary number that when added to a positive number of the same magnitude equals zero.

Integer		2's Complement
Signed	Unsigned	
5	5	0000 0101
4	4	0000 0100
3	3	0000 0011
2	2	0000 0010
1	1	0000 0001
0	0	0000 0000
-1	255	1111 1111
-2	254	1111 1110
-3	253	1111 1101
-4	252	1111 1100
-5	251	1111 1011

Note: The most significant (leftmost) bit indicates the sign of the integer; therefore it is sometimes called the sign bit.

If the sign bit is zero,
then the number is greater than or equal to zero, or positive.
If the sign bit is one,
then the number is less than zero, or negative.

Calculation of 2's Complement

To calculate the 2's complement of an integer, invert the binary equivalent of the number by changing all of the ones to zeroes and all of the zeroes to ones (also called **1's complement**), and then add one.

For example,

`0001 0001`(binary 17) ⇒ `1110 1111`(two's complement -17)

```
         NOT(0001 0001) = 1110 1110   (Invert bits)
1110 1110 + 0000 0001 = 1110 1111   (Add 1)
```

2's Complement Addition

Two's complement addition follows the same rules as **binary addition**.

For example,

5 + (-3) = 2

```
    0000 0101 = +5
  + 1111 1101 = -3
  -------------
    0000 0010 = +2
```

2's Complement Subtraction

Two's complement subtraction is the **binary addition** of the minuend to the 2's complement of the subtrahend (adding a negative number is the same as subtracting a positive one).

For example,

7 - 12 = (-5)

```
    0000 0111 = +7
  + 1111 0100 = -12
  -------------
    1111 1011 = -5
```

2's Complement Multiplication

Two's complement multiplication follows the same rules as **binary multiplication**.

For example,

$(-4) \times 4 = (-16)$:

```
  1111 1100 = -4
× 0000 0100 = +4
-------------
  1111 0000 = -16
```

2's Complement Division

Two's complement division is repeated **2's complement subtraction**. The 2's complement of the divisor is calculated, then added to the dividend. For the next subtraction cycle, the quotient replaces the dividend. This repeats until the quotient is too small for subtraction or is zero, then it becomes the remainder. The final answer is the total of subtraction cycles plus the remainder.

For example,

$7 \div 3 = 2$ remainder 1

```
  0000 0111 = +7        0000 0100 = +4
+ 1111 1101 = -3      + 1111 1101 = -3
-------------         -------------
  0000 0100 = +4        0000 0001 = +1 (remainder)
```

Sign Extension

To extend a signed integer from 8 bits to 16 bits or from 16 bits to 32 bits, append additional bits on the left side of the number. Fill each extra bit with the value of the smaller number's most significant bit (the sign bit).

For example,

Signed Integer	8-bit Representation	16-bit Representation
-1	1111 1111	1111 1111 1111 1111
+1	0000 0001	0000 0000 0000 0001

Other Representations of Signed Integers

Sign-Magnitude Representation

Another method of representing negative numbers is sign-magnitude. Sign-magnitude representation also uses the most significant bit of the number to indicate the sign. A negative number is the 7-bit binary representation of the positive number with the most significant bit set to one. The drawbacks to using this method for arithmetic computation are that a different set of rules are required and that zero can have two representations (+0, `0000 0000` and -0, `1000 0000`).

Offset Binary Representation

A third method for representing signed numbers is offset binary. Begin calculating a offset binary code by assigning half of the largest possible number as the zero value. A positive integer is the absolute value added to the zero number and a negative integer is subtracted. Offset binary is popular in A/D and D/A conversions, but it is still awkward for arithmetic computation.

For example,

Largest value for 8-bit integer $= 2^8 = 256$
Offset binary zero value $= 256 \div 2 = 128_{(decimal)} = 1000\ 0000_{(binary)}$

$1000\ 0000_{(\text{offset binary } 0)} + 0001\ 0110_{(\text{binary } 22)} = 1001\ 0110_{(\text{offset binary } +22)}$
$1000\ 0000_{(\text{offset binary } 0)} - 0000\ 0111_{(\text{binary } 7)} = 0111\ 1001_{(\text{offset binary } -7)}$

Signed Integer	Sign Magnitude	Offset Binary
+5	0000 0101	1000 0101
+4	0000 0100	1000 0100
+3	0000 0011	1000 0011
+2	0000 0010	1000 0010
+1	0000 0001	1000 0001
0	0000 0000 1000 0000	1000 0000
-1	1000 0001	0111 1111
-2	1000 0010	0111 1110
-3	1000 0011	0111 1101
-4	1000 0100	0111 1100
-5	1000 0101	0111 1011

Notes

Other Complements

1's Complement = $\mathrm{NOT}(n)$ = $\texttt{1111 1111} - n$
9's Complement = $\texttt{9999 9999} - n$
10's Complement = $(\texttt{9999 9999} - n) + 1$

BCD or **Binary Coded Decimal** is that number system or code which has the binary numbers or digits to represent a decimal number.

A decimal number contains 10 digits (0-9). Now the equivalent binary numbers can be found out of these 10 decimal numbers. In case of **BCD** the binary number formed by four binary digits, will be the equivalent code for the given decimal digits. In **BCD** we can use the binary number from 0000-1001 only, which are the decimal equivalent from 0-9 respectively. Suppose if a number have single decimal digit then it's equivalent **Binary Coded Decimal** will be the respective four binary digits of that decimal number and if the number contains two decimal digits then it's equivalent **BCD** will be the respective eight binary of the given decimal number. Four for the first decimal digit and next four for the second decimal digit. It may be cleared from an example.

Let, $(12)_{10}$ be the decimal number whose equivalent **Binary coded decimal** will be 00010010. Four bits from L.S.B is binary equivalent of 2 and next four is the binary equivalent of 1.

Table given below shows the binary and **BCD** codes for the decimal numbers 0 to 15.

From the table below, we can conclude that after 9 the decimal equivalent binary number is of four bit but in case of BCD it is an eight bit number. This is the main difference between Binary number and binary coded decimal. For 0 to 9 decimal numbers both binary and BCD is equal but when decimal number is more than one bit BCD differs from binary.

Decimal number	**Binary number**	**Binary Coded Decimal(BCD)**
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD Addition

Like other number system in BCD arithmetical operation may be required. BCD is a numerical code which has several rules for addition. The rules are given below in three steps with an example to make the idea of **BCD Addition** clear.

a) At first the given number are to be added using the rule of binary. For example,

Case 1:

$$\begin{array}{r} 1010 \\ +\ 0101 \\ \hline 1111 \end{array}$$

Case 2:

$$\begin{array}{r} 0001 \\ +\ 0101 \\ \hline 0110 \end{array}$$

b) In second step we have to judge the result of addition. Here two cases are shown to describe the rules of **BCD Addition.** In case 1 the result of addition of two binary number is greater than 9, which is not valid for BCD number. But the result of addition in case 2 is less than 9, which is valid for BCD numbers.

c) If the four bit result of addition is greater than 9 and if a carry bit is present in the result then it is invalid and we have to add 6 whose binary equivalent is $(0110)_2$ to the result of addition. Then the resultant that we would get will be a valid binary coded number. In case 1 the result was $(1111)_2$, which is greater than 9 so we have to add 6 or $(0110)_2$ to it.

$(1111)_2 + (0110)_2 = 0001\ 0101 = 15$. As you can see the result is valid in BCD.

But in case 2 the result was already valid BCD, so there is no need to add 6. This is how BCD Addition could be.

Now a question may arrive that why 6 is being added to the addition result in case BCD Addition instead of any other numbers. It is done to skip the six invalid states of binary coded decimal i.e from 10 to 15 and again return to the BCD codes.

Now the idea of BCD Addition can be cleared from two more examples.

Example:1
Let 0101 is added with 0110.

$$
\begin{array}{r l}
 & 0101 \\
+ & \underline{0110} \\
 & 1011 \rightarrow \text{Invalid BCD number} \\
+ & \underline{0110} \rightarrow \text{Add 6} \\
 & 0001\ 0001 \rightarrow \text{Valid BCD number}
\end{array}
$$

Check your self. $(0101)_2 \rightarrow (5)_{10}$ & $(0110)_2 \rightarrow (6)_{10}$

$(5)_{10} + (6)_{10} = (11)_{10}$

Example:2

Now let 0001 0011 is added to 0010 0110.

$$
\begin{array}{r l}
 & 0001\ 0001 \\
+ & \underline{0010\ 0110} \\
 & 0011\ 0111 \longrightarrow \text{Valid BCD number}
\end{array}
$$

$(0001\ 0001)_{BCD} \rightarrow (11)_{10}$, $(0010\ 0110)_{BCD} \rightarrow (26)_{10}$ and $(0011\ 0111)_{BCD} \rightarrow (37)_{10}$
$(11)_{10}$
$+ (26)_{10} = (37)_{10}$
So no need to add 6 as because both $(0011)_2 = (3)_{10}$ and $(0111)_2 = (7)_{10}$ are less than $(9)_{10}$. This is the process of BCD Addition.

BCD Subtraction

There are several methods of **BCD Subtraction**. BCD subtraction can be done by 1's compliment method and 9's compliment method or 10's compliment method. Among all these methods 9's compliment method or 10's compliment method is the most easiest. We will clear our idea on both the methods of **BCD Subtraction**

Method of BCD Subtraction : 1

In 1st method we will do **BCD Subtraction** by 1's compliment method. There are several steps for this method shown below. They are:-

a) At first 1's compliment of the subtrahend is done.

b) Then the complimented subtrahend is added to the other number from which the subtraction is

Assignment - 1

1. Simplify $x+\bar{x}y$ Ans: $x+y$
2. Apply Demorgans theorem to simplify $\overline{A+B\bar{C}}$ Ans: $\bar{A}\bar{B}+\bar{A}C$
3. prove the following Boolean identities.

$$(x_1+x_2)(\bar{x}_1\bar{x}_3+x_3)(\overline{\bar{x}_2+x_1x_3}) = \bar{x}_1x_2$$

4. $\bar{A}BC\bar{D}+BC\bar{D}+B\bar{C}\bar{D}+B\bar{C}D$. Ans $B(\bar{D}+\bar{C})$
5. $AB+\overline{AC}+A\bar{B}C(AB+C)$. Ans: 1
6. Simplify $A+AB+\bar{A}+B$ Ans: 1
7. Apply Demorgan's theorem to the following expression $\overline{((A+B+C)D)}$ Ans: $\bar{A}\bar{B}\bar{C}+\bar{D}$
8. using Boolean laws and rules simplify the logic expression. $Z=(\bar{A}+B)(A+B)$ Ans: B
9. prove the following using Demorgan's theorem.
(1) $AB+CD = \overline{\overline{AB}\cdot\overline{CD}}$ (2) $(A+B)(C+D) = \overline{\overline{A+B}+\overline{C+D}}$
10. Apply Demorgans theorem for the function

repeated $\overline{(A+B+C)D}$ Ans $\bar{A}\overline{BC}+\bar{D}$

11. Find the complement of $A+BC+AB$ $\bar{A}\bar{B}+\bar{A}\bar{C}$
12. Simplify $(B\bar{C}+\bar{A}D)(A\bar{B}+C\bar{D})$ Ans: 0
13. Simplify $xy+\bar{x}z+yz$ Ans: $xy+\bar{x}z$
14. Simplify $a\bar{b}\bar{c}+a\bar{b}c+abc$ Ans: $a\bar{b}+ac$
15. Simplify $(a+b)(a+\bar{b})$ Ans: x.

Questions:

1. What are Boolean Variables?
2. Define the following terms: Boolean Variables, Complement, literals.
3. State the fundamental postulates of Boolean algebra.
4. State various laws of Boolean algebra.
5. State the associative law of Boolean algebra.
6. State and prove Demorgan's theorem.
7. Explain the principle of Duality with example.

* what are universal gates? Give examples.
* Why NAND and NOR gates are called universal gates?
* why digital circuits are more frequently constructed with NAND OR NOR gates than with AND and OR gates?
* write the logic symbol, expression and truth table for the following logic gates:

(i) EX-OR (ii) NOR (iii) NAND (iv) EX-NOR

* what are the basic digital logic gates?
* Give the Boolean expression used for following gates.

a) AND b) NOR c) EX-OR D) OR E) NOT.

part - A. Assignment - 2.

. Convert 9ABCD Hexa to Octal and decimal.
2. Show that $A(A+B) = A$.
3. Convert $FACE_{16}$ to Binary
4. Simplify $F = AB + \overline{AC} + A\overline{B}C(AB+C)$
5. Convert the hexadecimal 68BE to binary.
6. Perform the subtraction in the following unsigned binary number using 1's Complement method
   11011 - 11001
7. Determine the decimal value of the fractional binary number 0.1011.
8. State De-Morgan's theorem
9. What is an alphanumeric code?
10. Multiply $(1011)_2$ by $(101)_2$
11. Show that excess-3 Code is self Complementing.
12. 25_{10} to gray.
13. state and prove Consensus theorem

). prove that
   (i) $x + yz = (x+y)(x+z)$
   (ii) $x\overline{y} + y = x + y$
   (i) 427_8 to decimal, binary and hexa.
   (ii) $(1A53)_{16}$ to other systems.
1. Decimal to binary, Octal and hexa.
   1) 25 2) 58 3) 82 4 99 5) 112 6) 555.
12. a) Prove $(A+B)(C+D) = \overline{(\overline{A+B}) + (\overline{C+D})}$
b) Represent 396 and 4096 in Binary, BCD, Excess-3, Hexa, Octal.

part - B.

1. Perform the following arithmetic using 9's complemen[t] Compare them.

(i) 835 − 274 (ii) 429 − 476 using BCD and Excess 3 code

2. Simplify the following switching equation using Boolean algebra

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

(ii) $W = (X_1 + X_2)(\bar{X_1} + X_1X_2) + (\bar{X_2} + X_1\bar{X_2})$

3. a) Solve for X when $(137)_x = (5F)_{16}$ (4)

b) Simplify and implement the function using Basic gates. $F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + AB\bar{C} + ABC$ (5)

c) speed power product of logic gates (3).

4) a) Subtract 232 from 343 using 2's comp. meth[od] (4)

5) write a short Note on (i) Binary codes (ii) Arithemetic Codes.

6. a) Convert the following hexadecimal numbers to decimal.

(i) $1C_{16}$ (ii) $E5_{16}$ (iii) $B2F8_{16}$

b) List out any four basic rules that are used in Boolean algebra expression

7. state and prove the theorems of Boolean algebra with illustration.

8. Reduce a) $y = (A + (BC)')(AB' + (ABC)')$

b) $y = ((AB)' + A' + AB)'$

c) $y = C(B+C)(A+B+C)$.

9. perform the following.

a) $(1100 \cdot 101)_2 = (\ \ )_8 = (\ \ )_{10} = (\ \ )_{16}$

b) $(\cdot 98A)_{16} = (\ \ )_2 = (\ \ )_8 = (\ \ )_{10}$

c) $(674)_8 = (\ \ )_{10} = (\ \ )_2 = (\ \ )_{16}$.

part - A. Assignment - 2.

1. Convert 9ABCD Hexa to Octal and decimal.
2. Show that $A(A+B) = A$.
3. Convert $FACE_{16}$ to Binary
4. Simplify $F = AB + \overline{AC} + A\overline{B}C(AB+C)$
5. Convert the hexadecimal 68BE to binary.
6. Perform the subtraction in the following unsigned binary number using 1's Complement method
   11011 − 11001
7. Determine the decimal value of the fractional binary number 0.1011.
8. State De-Morgan's theorem
9. What is an alphanumeric code?
10. multiply $(1011)_2$ by $(101)_2$
11. Show that excess-3 Code is self Complementing.
12. 25_{10} to gray.
13. State and prove Consensus theorem

10. prove that
   (i) $x + yz = (x+y)(x+z)$
   (ii) $x\overline{y} + y = x + y$
   (i) 427_8 to decimal, binary and hexa.
   (ii) $(1A53)_{16}$ to other systems.
11. Decimal to binary, Octal and hexa.
   1) 25 2) 58 3) 82 4) 99 5) 112 6) 555.
12. a) Prove $(A+B)(C+D) = ((\overline{A+B}) + (C+D))$
   b) Represent 396 and 4096 in Binary, BCD, Excess-3, Hexa, Octal.

part - B .

1. Perform the following arithmetic using 9's complement Compare them.

(i) 835 − 274 (ii) 429 − 476 using BCD and Excess 3 codes

2. Simplify the following switching equation using Boolean algebra

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

(ii) $W = (X_1 + X_2)(\bar{X_1} + X_1X_2) + (\bar{X_2} + X_1\bar{X_2})$

3. a) Solve for X when $(137)_x = (5F)_{16}$ (4)

b) Simplify and implement the function using Basic gates. $F = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + AB\bar{C} + ABC$ (5)

c) Speed Power product of logic gates (3).

4) a) Subtract 232 from 343 using 2's comp. method (4)

5) write a short note on (i) Binary codes (ii) Arithemetic Codes.

6. a) Convert the following hexadecimal numbers to decimal.

(i) $1C_{16}$ (ii) $E5_{16}$ (iii) $B2F8_{16}$

b) List out any four basic rules that are used in Boolean algebra expression

7. state and prove the theorems of Boolean algebra with illustration.

8. Reduce a) $y = (A + (BC)')(AB' + (ABC)')$

b) $y = ((AB)' + A' + AB)'$

c) $y = C(B + C)(A + B + C)$.

9. perform the following.

a) $(1100.101)_2 = (\ )_8 = (\ )_{10} = (\ )_{16}$

b) $(98A)_{16} = (\ )_2 = (\ )_8 = (\ )_{10}$

c) $(674)_8 = (\ )_{10} = (\ )_2 = (\ )_{16}$.