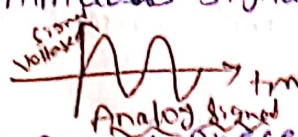
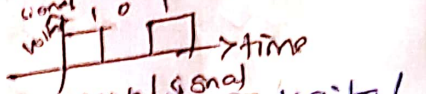
Digital electronics

Digital electronics: Digital electronics is the electronic circuit that process and control the digital signal.

Analog signal	Digital signal
① Analog signals are continuous signals 	Digital signal are not continuous, they are discrete signals 
② We can represent analog signals in the form of sign waves	We can represent digital signals in the form of square waves
③ The values of voltage will be in a continuous range.	The values of voltage will be dis continuous
④ Records the information as it is	Converts the Information into binary form.
⑤ Easily affected by the noise	These are stable and less affected by the noise
⑥ Analog signals may be affected during data transmission	Digital signals are not affected during data transmission
⑦ Analog signals use more power	Digital signals are use less power.
⑧ Example temperature, Presure flow measurements.	Examples Value feeback, motor start, Trip etc.
⑨ Components like resistors, capacitors, Inductors, Diodes are used in analog circuits	Components like transistors, logic gates, Microcontrollers, are used in Digital circuits.
⑩ Low cost	Cost is high.

* Digital Electronics *

UNIT-I

Boolean algebra and logic gates

Digital system :

A Digital system is an interconnection of digital components that manipulates and process information that is represented internally in the binary form.

Now a days digital systems are used everywhere like digital computers, digital watches, TV games, micro processors, smart watches, automated industrial machinery, consumer products and so on.

We use digital logic to operate digital electronic systems, this logic is rooted in binary code, i.e combination of zeros and ones. Digital system uses physical quantities called signals to represent discrete elements. Digital electronic system contains logic gates such as 'AND' 'OR' and 'NOT'. This system translates input signals into specific output with the presence of digital logic.

So, the concept of digital logic design is foundational to the fields of electrical engineering and computer engineering.

Advantages of Digital systems :

* Ease of programmability.
* Reduction in hardware
* High speed
* High reliability.
* Designing a circuit is easy.

Number System:

Number system is basis for counting items. Generally computers communicate and operate with binary numbers which uses only '0' & '1'. We, humans use Decimal number system.

Suppose let us consider a decimal number, '18' this number is represented in binary as 10010.

We observe that binary number system take more digits to represent the decimal number. for larger numbers we have to deal with very bigger strings. so this fact give rise to three new number systems.

i) octal number system
ii) Hexa decimal number system
iii) Binary coded Decimal number system.

To define any number system we have to specify Base of the number system such as 2, 8, 10 (or) 16.

The Decimal Number systems:

The Decimal number system contains ten unique symbols 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Since it contains ten symbols it's base (or) radix is ten. This number system is positional weighted system. ie the value attached to a symbol depends on its location with respect to the decimal point. Each symbol is called a digit.

The left most digit, which has the greatest positional weight out of all the digits present in the number is called the 'MSD'

(Most significant digit) and the right most digit which has least positional weight out of all the digits present in that number is called 'LSD' (least significant digit)

The leftmost digit, which has the greatest weight is called the most significant digit and the right most digit, which has the least weight, is called the least significant digit.

	10^3	10^2	10^1	10^0		10^{-1}
	5	6	7	8	.	9
In power of 10	5×10^3	6×10^2	7×10^1	8×10^0	.	9×10^{-1}

MSD LSD

$= 5\times10^3+6\times10^2+7\times10^1+8\times10^0+9\times10^{-1} = (5678.9)_{10}$

Representation of a decimal number.

Binary Number system:

Binary number system:

Binary system with it's two digits is a base-two system. The two binary digits (bits) are '1' & '0'. In binary systems, weight is expressed as a power of 2. By adding each digit of a binary number in a power of 2 we can find the decimal equivalent of the given binary numbers

	2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
	1	1	0	1	.	1	0	1
	1×2^3	1×2^2	0×2^1	1×2^0	.	1×2^{-1}	0×2^{-2}	1×2^{-3}

MSB LSB

$1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3} = (13.625)_{10}$

Octal Number System:

The octal number system uses first eight digit of ~~decimal~~ decimal number system : 0, 1, 2, 3, 4, 5, 6, & 7. As it uses 8 digits, it's base is 8.

By adding each digit of an octal number in a power of '8', we can find the decimal equivalent of the given octal number.

8^3	8^2	8^1	8^0		8^{-1}	8^{-2}	8^{-3}
5	6	3	2	.	4	7	1
5×8^3	6×8^2	3×8^1	2×8^0	.	4×8^{-1}	7×8^{-2}	1×8^{-3}

The octal number 5632.471 can be represented in power of '8', we can find the decimal equivalent of the given octal number.

$$= 5\times8^3 + 6\times8^2 + 3\times8^1 + 2\times8^0 + 4\times8^{-1} + 7\times8^{-2} + 1\times8^{-3}$$

$$= 5\times512 + 6\times64 + 3\times8 + 2\times1 + 4\times0.125 + 7\times0.015625 + 1\times0.001953125$$

$$= (2970.611328)_{10}$$

Hexadecimal Number System:

The hexadecimal number system has a base of '16' having '16' characters 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

By adding each digit of a hexadecimal number in a power of '16', we can find decimal equivalent of the given hexadecimal number.

16^2	16^1	16^0		16^{-1}	16^{-2}
3	F	D	.	8	4
3×16^2	$F\times16^1$	$D\times16^0$	.	8×16^{-1}	4×16^{-2}

$$= 3\times16^2 + F\times16^1 + D\times16^0 + 8\times16^{-1} + 4\times16^{-2}$$

(F = 15, D = 13)

$$= (1021.515625)_{10}$$

Format of a Binary number:

A single digit in the binary number is called bit. four binary digits form a nibble, eight binary digits form a byte, sixteen binary digits form a word and thirty-two binary digits form a double-word.

Bit →								
$b_{31}b_{30}b_{29}b_{28}$	$b_{27}b_{26}b_{25}b_{24}$	$b_{23}b_{22}b_{21}b_{20}$	$b_{19}b_{18}b_{17}b_{16}$	$b_{15}b_{14}b_{13}b_{12}$	$b_{11}b_{10}b_9b_8$	$b_7b_6b_5b_4$	$b_3b_2b_1(b_0)$
Nibble 7	Nibble 6	Nibble 5	Nibble 4	Nibble 3	Nibble 2	Nibble 1	Nibble 0
Byte 3		Byte 2		Byte 1		Byte 0	
Word 1				Word 0			
Double word							

Number base conversions:

* **Binary to Decimal number conversion:**

Convert $(1101.101)_2$ to decimal number and explain the process of conversion

$(1000)_2 = (?)_{10}$

Sol By adding each digit of a binary number in a power of 2 we can find the decimal equivalent of the given binary number.

2^3	2^2	2^1	2^0		2^{-1}	2^{-2}	2^{-3}
1	1	0	1	.	1	0	1
$1X2^3$	$1X2^2$	$0X2^1$	$1X2^0$	.	$1X2^{-1}$	$0X2^{-2}$	$1X2^{-3}$

$= 1X2^3 + 1X2^2 + 0X2^1 + 1X2^0 + 1X2^{-1} + 0X2^{-2} + 1X2^{-3}$

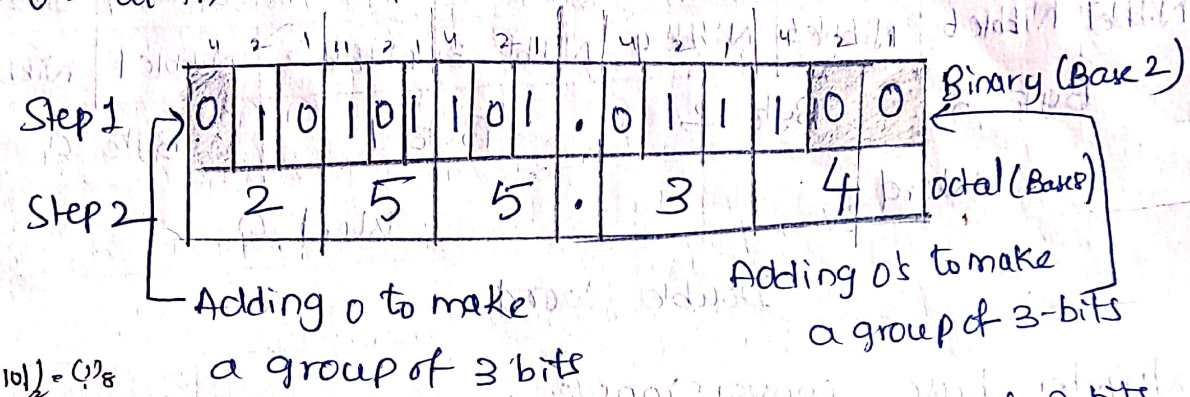
$= 8 + 4 + 1 + 0.5 + 0 + 0.166$

$= (13.666)_{10}$

* Binary to octal conversion:

Convert 10101101.0111 to octal equivalent and explain the conversion process.

Sol Step 1: Make group of 3 bits starting from LSB for integer part and MSB for fractional part, by adding 0's at the end, if required



Convert $(100110101)_2 = (?)_8$

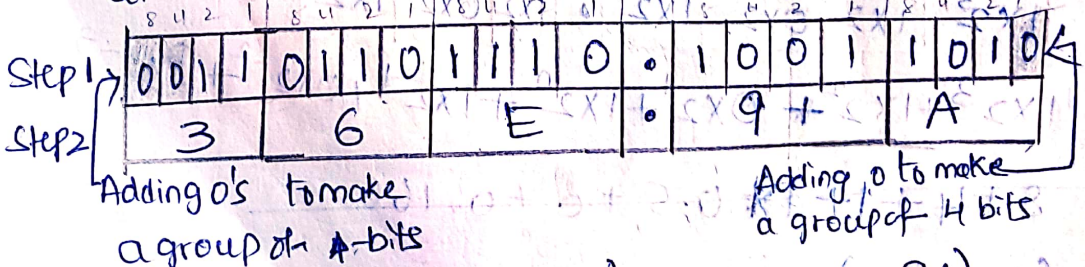
Equivalent octal number for each group of 3 bits

$(10101101.0111)_2 = (255.34)_8$

* Binary to Hexadecimal:

Convert 110110111 0.1001101 to hexadecimal equivalent and explain the conversion process.

A) → Make group of 4-bits starting from LSB for integer part and MSB for fractional part, by adding 0's at the end, if required.



A-10
B-11
C-12
D-13
E-14
F-15

$(1001\ 0110\ 1001\ 1100)_2 = (?)_{16}$

$(110110111.1001101)_2 = (36E.9A)_{16}$

969C16

Octal to binary Conversion:

Conversion from octal to binary is a reversal of the process of binary to octal conversion, each digit of the octal number is individually converted to its binary equivalent to get octal to binary conversion of the number

Convert $(125.62)_8$ to binary

Sol:- 1) Write equivalent 3-bit binary number for each octal digit

2) Remove any leading (or) trailing zeros

$(254)_8 = (?)_2$

Leading zeros → 1 2 5 . 6 2 ← trailing zeros

001 010 101 . 110 010

$(125.62)_8 = (1010101.110001)_2$

$x^0 = 1$
$x^1 = x$

$(543)_8 = (?)_{10}$
$= (643)_{10}$

Octal to hexa decimal conversion:

The easiest way to convert octal number to hexadecimal number is

1. Convert octal number to its binary equivalent
2. Convert binary number to its hexadecimal equivalent

Ex:- convert $(615.25)_8$ to hexa decimal.

1. Write equivalent 3-bit binary number for each digit
2. Make group of 4 bits starting from LSB for integer part and MSB for fractional part by adding 0's at the end, if required.
3. Write equivalent octal number for each group of 4 bits.

$(615.25)_8$ = 6 1 5 . 2 5

110 001 101 . 010 101

0001 1000 1101 . 0101 0100

1 8 D . 5 4

$(615.25)_8 = (18D.54)_{16}$

Hexadecimal to binary conversion :-

Conversion from hexadecimal to binary is a reverse of the process of conversion of binary to hexadecimal. Each digit of the hexadecimal number is individually converted to it's binary equivalent to get hexadecimal to binary conversion of the number

Convert $(8A9.B4)_{16}$ to binary

8 A 9 . B 4

1000 1010 1001 . 1011 0100

$(8A9.B4)_{16} = (1000101010011011010)_2$ $(A69.8)_{16} = (?)_{10}$

Hexadecimal to octal conversion :-

The easiest way to convert hexadecimal number to octal number

1. Convert hexadecimal number to it's binary equivalent
2. Convert binary number to it's octal equivalent

Convert $(BC66.AF)_{16}$ to Octal

B C 6 6 . A F

1011 1100 0110 0110 . 1010 1111 00

added 0's to make group.

added '0' to make group.

3 6 1 4 6 . 5 3 6

$(BC66.AF)_{16} = (136146.536)_8$

A-10
B-11
C-12
D-13
E-14
F-15

Converting any radix to decimal:-

To convert any radix to decimal expand the given number where each digit is multiplied by its weight (or) value.

In general numbers can be represented as.

$$N = A_{n-1}r^{n-1} + A_{n-2}r^{n-2} + \dots + A_1 r^1 + A_0 r^0 + A_{-1}r^{-1} + A_{-2}r^{-2} + \dots + C_{-m}r^{-m}$$

where N = Number in decimal

A = digit

r = Radix (or) base of a number system

n = The no. of digits in the integer portion of number

m = The no. of digits in the fractional portion of no.

Ex:- Convert $(3102.12)_4$ to decimal equivalent

$$N = 3\times4^3 + 1\times4^2 + 0\times4^1 + 2\times4^0 + 1\times4^{-1} + 2\times4^{-2}$$

$$= 192 + 16 + 0 + 2 + 0.25 + 0.125$$

$$N = (210.375)_{10}$$

→ Conversion of decimal number to any radix number:

Step 1: Convertion of integer part

Step 2: Conversion of fractional part

The conversion of integer part is accomplished by successive division method and the conversion of fractional part is accomplished by successive multiplication method.

Steps in successive division

① Divide the integer part of decimal number by desired base number, store Quotient (Q), & remainder (R)

② Consider quotient as a new decimal number and repeat step 1, until quotient becomes '0'

③ List the remainders in reverse order.

Steps in Successive multiplication method:-

(1) Multiply the fractional part of decimal number by desired base.

(2) Store integer part of product as carry and fractional part as new fractional part.

(3) Repeat step1 & step2 until fractional part of product becomes '0'(or) until you have many digits as necessary for your application

(4) Read carries downwards to get desired base number

Ex: Convert $(12.125)_{10}$ into binary

Sol

12. 125

integer | fraction

Integer part :-

2	12
2	6 — 0
2	3 — 0
	1 — 1

$(12)_{10} = (1100)_2$

Fraction part

$0.125 \times 2 = 0.25 \rightarrow 0$

$0.25 \times 2 = 0.50 \rightarrow 0$

$0.50 \times 2 = 1.00 \rightarrow 1$

$(0.125)_{10} = (0.001)_2$

$(12.125)_{10} = (1100.001)_2$

Ex: Convert $(658.825)_{10}$ into octal:-

Sol

Integer

8	658
8	82 — 2
8	10 — 2
	1 — 2

$(658)_{10} = (1222)_8$

Fractional part:

$0.825 \times 8 = 6.6 \rightarrow 6$

$0.6 \times 8 = 4.8 \rightarrow 4$

$0.8 \times 8 = 6.4 \rightarrow 6$

we have restricted fractional part upto 3-digits this is an approximate answer, to get accurate answer, we have to continue multiplying by '8'

$(658.825)_{10} = (1222.646)_8$

Convert $(5386.345)_{10}$ into hexadecimal

Sol integer part

16	5386
16	336 - 10 (A)
16	21 - 0
	1 - 5

$(5386)_{10} = (150A)_{16}$

fractional part

$0.345 \times 16 = 5.52 \rightarrow 5$

$0.52 \times 16 = 8.32 \rightarrow 8$

$0.32 \times 16 = 5.12 \rightarrow 5$

$(0.345)_{10} = (0.585)_{16}$

$(5386.345)_{10} = (150A.585)_{16}$.

* Complements *

Complement is the opposite of the something given

i's complements representation:

i's complement of binary number is the number that results when we change all i's to 0's and all 0's to i's.

Ex1- find the the i's complement of $(11010100)_2$

1 1 0 1 0 1 0 0

0 0 1 0 1 0 1 1

i's complement means the addition of negative integer to the 'no' & this eliminates the necessity of a separate subtracting processor.

i's definition

We can find the i's complement of the binary number by simply inverting the given number

* 2's complement representation *

The 2's complement is the binary number that results when we add '1' to the 1's complement

2's complement = 1's complement + 1

The 2's complement form is used to represent -ve no. (negative number)

* Advantages of 2's complement representation *

* There are distinct +0 and -0 representations in both the sign magnitude and 1's complement systems, but the 2's complement systems has only a +0 representation.
* for 4-bit numbers, the value -8 is representable only in the 2's complement system & not in other systems.
* It is more efficient for logic circuits implementation and often used in computers, for addition and subtraction operations.

Ex: find the 2's complement of $(11011011)_2$

Sol: find 1's complement of $(11011011)_2$

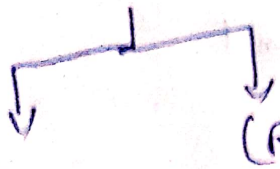
00100100

+1

00100101

→ for subtraction we use complements

complements are two types



R's

2, 8, 10, 16

(R-1)'s

1, 7, 9, 15

→ Most used complements are 1's and 2's

10's & 9's complement

→ Only for decimal number system

9's complements

Subtract each and every decimal no. from 9

① 4 5 3

9 9 9

(-) 4 5 3

5 4 6

10's complements

Step1: perform 9's complement

Step2: Add 1 to the LSD of 9's complement

Step1 ① 4 5 3

9 9 9

(-) 4 5 3

5 4 6

Step2

5 4 6

+ 1

5 4 7

8's and 7's complements

→ Apply for octal number system.

→ 7's complement

Subtract each and every octal from 7

① $(372)_8$

$$\begin{array}{r} 7\ 7\ 7 \\ -\ 3\ 7\ 2 \\ \hline 4\ 0\ 5 \end{array} \rightarrow 7\text{'s}$$

8's complement

Step1: Perform 7's complement

Step2: Add 1 to the LSD of 7's complements

Step1:

$$\begin{array}{r} 7\ 7\ 7 \\ 3\ 7\ 2 \\ \hline 4\ 0\ 5 \end{array} - 7\text{'s}$$

Step2:

$$\begin{array}{r} 405 \\ +\ 1 \\ \hline 406 \end{array} - 8\text{'s}$$

15's and 16's complements Apply for Hexadecimal number system

Subtract each and every Hexa decimal no. from F

A - 10
B - 11
C - 12
D - 13
E - 14
F - 15

① $(BAD.BA)_{16}$

$$\begin{array}{r} F\ F\ F.F\ F \\ -\ 3\ A\ D.B\ A \\ \hline C\ 5\ 2.4\ 5 \end{array} - 15\text{'s}$$

15 15 15 15
10 13 11 10
5 2 4 5

16's complement

Step1: perform 15's complement

Step2: Add 1 to LSD of 15's complements.

$$\begin{array}{r} C\ 5\ 2.4\ 5 \\ +\ 1 \\ \hline C\ 5\ 2.4\ 6 \end{array} - 16\text{'s}$$

Ex: perform addition $(11001100)_2$ and $(11011010)_2$.

```
  +1  +1  +1
   1   1   0   0   1   1   0   0
       1   1   0   1   1   0   1   0
+1 ---------------------------------
   1   1   0   1   0   0   1   1   0
```

Ex:- perform $(11011100)_2 - (00110010)_2$

```
   -1  -1          -1 Barrow
    1   1   0   1   1   0   0
    0   0   1   1   0   0   1   0
-------------------------------------
    1   0   1   1   1   0   1   0   Result
```

* Binary subtraction using 1's complement method *

* In 1's complement subtraction -ve number is represented in the 1's complement form. and actual addition is perform to get the desired result.
* for example, operation A-B is performed using following steps.

1. Take 1's complement of B.
2. Perform addition of A with 1's complement of B.
3. If carry is generated then the result is positive. and it is in the true form. and add carry to the result to get the final result.
4. If the carry is not generated then the result is negative. and it is in 1's complement form.

Ex:

ex: Perform 1's complement subtraction $(11011)_2 - (11011)_2$

Sol A = 11011 B = 11011

1's complement of $(11011)_2 = (000100)_2$

Perform addition with A and 1's complement of B.

$$\begin{array}{r} 11011 \\ +000100 \\ \hline 11101 \end{array}$$

+1

11101.0 - Result

* Binary subtraction using 2's complement method *

* In a 2's complement subtraction, negative number is represented in the 2's complement form and actual addition is perform to get the desired result.
* for example, the operation A-B is performed using following steps.

1. Take 2's complement of B.
2. Result ← Add A with 2's complement of B.
3. If carry is generated then the result is +ve and in the true form. In this case, carry is ~~ignored~~ ignored
4. If carry is not generated then the result is negative (-ve) & in 2's complement form.

* Binary Arithmetic *

→ Binary addition :-

The addition consists of four possible elementary operation. The first three operations produce a sum whose length is one digit, but when the last operations performed sum is two digits.

The higher significant bit of this result is called a carry and lower significant bit is called sum.

Such an operation is known as half - addition

The operation which performs addition of three bits is called a full addition

S.NO.	operations			
	A	B	sum	carry
1	0	0	0	0
2	0	1	1	0
3	1	0	1	0
4	1	1	0	1

→ Binary subtraction :-

The subtraction consists of four possible elementary operations, In all operations, each subtrahend bit is subtracted from the minuend bit. In case of second operation the minend bit is smaller than the subtrahend bit, hence '1' is borrowed.

Binary subtraction

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Binary addition

Example 1

$$\begin{array}{r} {\scriptstyle 1}\ \ {\scriptstyle 1}\ \ \ \ \\ 1\ \ 1\ \ 0\ \ 0 \\ +\ \ 1\ \ 1\ \ 0\ \ 1 \\ \hline 1\ \ 1\ \ 0\ \ 0\ \ 1 \end{array}$$

Binary Subtraction

Example :

$$\begin{array}{r} 1\ \ 1\ \ 1\ \ 0 \\ -\ \ 1\ \ 0\ \ 1\ \ 0 \\ \hline 0\ \ 1\ \ 0\ \ 0 \end{array}$$

(E 4 2 1)
1110 → 14
1010 → 10
0100
4

Binary multiplication :

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

Example

$$\begin{array}{r} 1\ \ 1\ \ 0 \\ 1\ \ 0\ \ 1 \\ \hline 1\ \ 1\ \ 0 \\ 0\ \ 0\ \ 0\ \ X \\ 1\ \ 1\ \ 0\ \ X\ \ X \\ \hline 1\ \ 1\ \ 1\ \ 1\ \ 0 \end{array}$$

Norm

X

6
X5
30

Example

$$
\begin{array}{r}
1010 \\
\times\ 101 \\
\hline
1010 \\
0000\times \\
1010\times\times \\
\hline
1100110
\end{array}
$$

Binary division:

$0 \div 0 = 0$ meaningless.

$0 \div 1 = 0$

$1 \div 0 = \infty$ (meaningless)

$1 \div 1 = 1$

Example

$$
\begin{array}{r|l}
101 & 101101 \ (1001 \\
\hline
101 & \\
\hline
0 & 101 \\
 & 101 \\
\hline
 & 0
\end{array}
$$

Dividend = 101101

Divisor = 101

Quotient = 1001

Remainder = 0

$(110110)_2 \div (1001)_2$

$$
\begin{array}{r|l}
1001 & 110110 \ (110 \\
\hline
1001 & \\
\hline
01001 & \\
1001 & \\
\hline
00000 &
\end{array}
$$

$10101 \div 11 = ?$

111

$(110110)_2 \div (1001)_2 = (110)_2$

Binary subtraction :

S.NO	A B	Operations	
		Difference	Borrow
1	0 0	0	0
2	0 1	1	1
3	1 0	1	0
4	1 1	0	0

Binary code :

It is a way of representing letters characters and numbers using group of 1's & 0's

→ *Binary codes *

The digital data is represented, stored and transmitted as group of binary (or) bits, the group of bits, also known as Binary code, represent both numbers and letters of the alphabets as well as many special characters and control functions. They are classified as numeric (or) alphanumeric

* Numeric codes are used to represent numbers
* Alphanumeric codes are used to represent characters: alphabets, letters and numerals.

* Classification of Binary codes *

The codes are mainly classified as

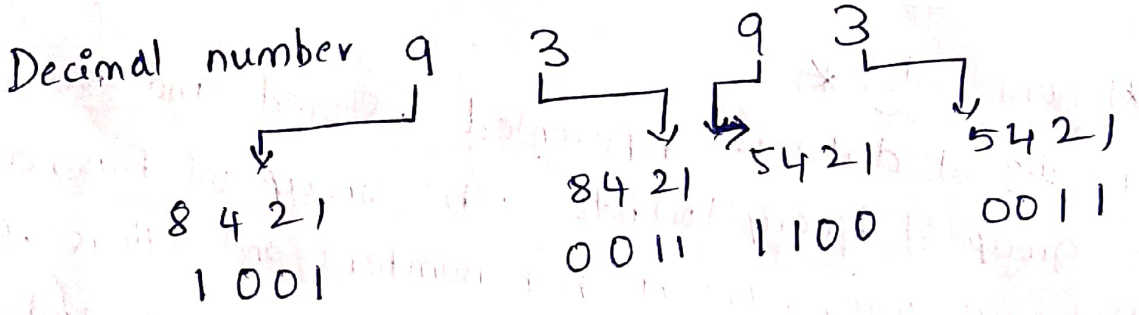
1. Weighted codes
2. Non-weighted codes
3. Reflective codes
4. Sequential codes
5. Alphanumeric codes
6. Error detecting and correcting codes.

* Weighted codes *

* In weighted codes, each digit position of the number represent a specific weight.

* In weighted binary code each bit has a weight 8,4,2(or) 1 and each decimal digit is represented by a group of four bits

Ex:- BCD, 8421, 5421, 2421, 5211 etc.



* Non-weighted codes *

Non-weighted codes are not assigned with any weight to each digit position i,e, each digit position within the number is not assigned fixed value.

Ex:- Excess-3, Gray code.

* Reflective codes *

A code is said to be, reflective when the code for '9' is the complement for '0', the code for '8' is complement for 1, 7 for 2, 6 for 3, 5 for 4.

Ex:- 2421, 5211, Excess-3 codes.

* Sequential codes *

In Sequential codes each succeding code is one binary number greater than its preceeding code.

Ex:- 8421, Excess-3 codes.

Alphanumeric codes:

The codes which consists of both numbers and alphabetics characters are called alphanumeric codes. Most of the codes, however, also represent symbol and various instructions necessary for converging intelligible information.

* The most commonly used alphanumeric codes are

ASCII - (American Standard code for Information Interchange

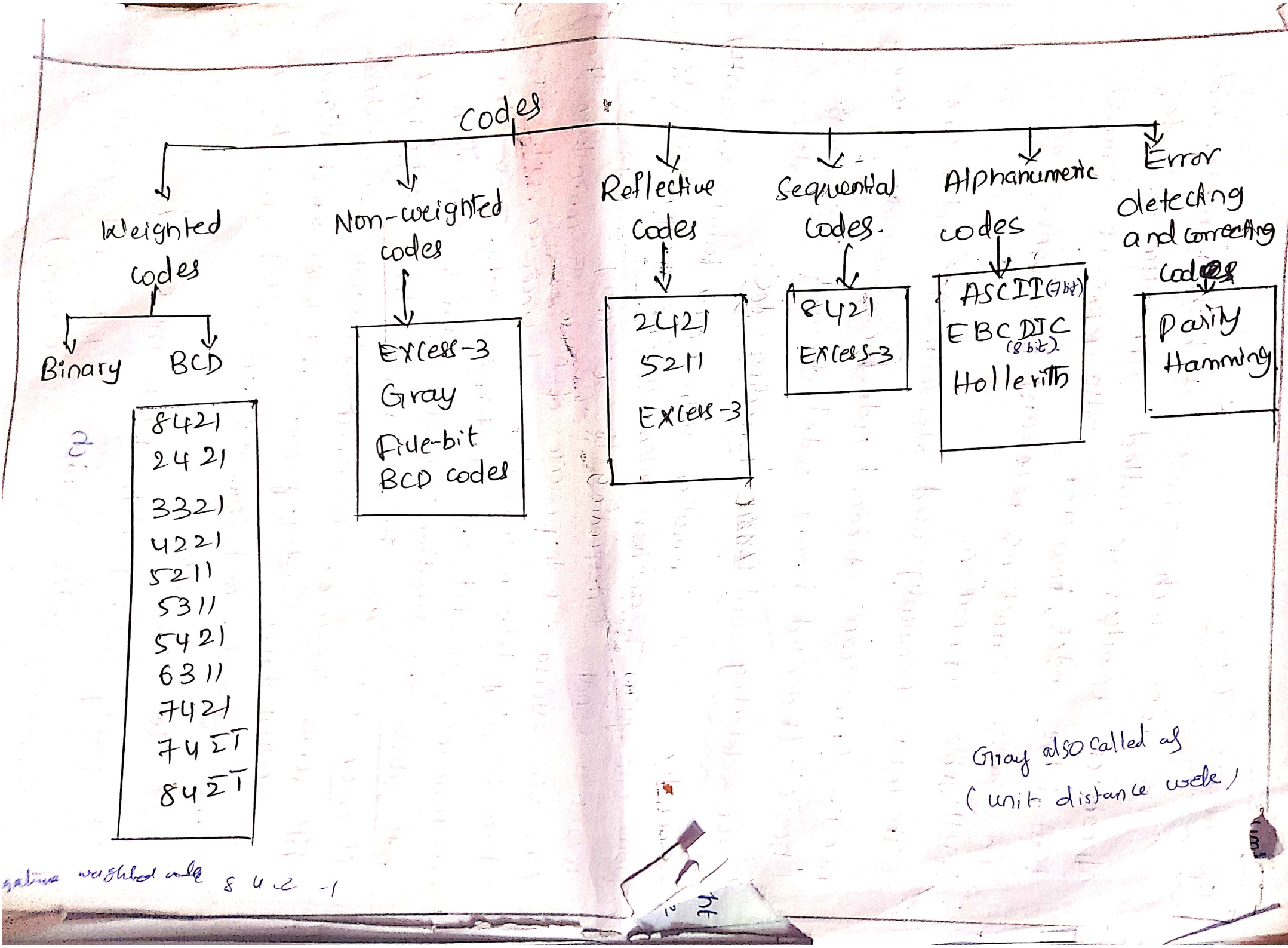
EBCDIC - (Extended binary coded decimal Interchange code

Error detecting and correcting codes:-

when the digital information in the binary form is transmitted from one system to another system an error may occur. This means a signal corresponding to '0' may change to '1' (or) vice-versa due to presence of noise. To maintain the data integrity between transmitter and receiver, extra bit (or) more than one bit are added in the data. These extra bits allow the detection and some times correction of error in the data. The data along with the extra bit/ bits forms the code. codes which allow only error detection are called Error detecting codes, and codes which allow error detection and correction are called error detecting and correcting codes.

Ex: parity codes (which detects the errors)

Hamming codes (which detects and corrects the error)



* Subtract 14 from 46 using the 8-bit 2's complement

Sol 46 - 14

14 → 128 64 32 16 8 4 2 1
0 0 0 0 1 1 1 0

1's complement → 1 1 1 1 0 0 0 1

2's complement → 1 1 1 1 0 0 0 1
+ 1
1 1 1 1 0 0 1 0

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
128 64 32 16 8 4 2 1
8-bit no

+46 → 0 0 1 0 1 1 1 0

2's complement of 2nd no. -14 → 1 1 1 1 0 0 1 0

1 0 0 1 0 0 0 0

If carry generated ignore it.

MSB is 0 so Result is +ve and is in Normal Binary form.

Result is 0 0 1 0 0 0 0 0

⇒ +32.

* Subtract -75 from +26 using the 8-bit 2's complement form

75 → 128 64 32 16 8 4 2 1
0 1 0 0 1 0 1 1

1's complement → 1 0 1 1 0 1 0 0

2's complement → 1 0 1 1 0 1 0 0
+ 1
1 0 1 1 0 1 0 1

1 → -ve
0 → +ve

26 → 0 0 0 1 1 0 1 0
-75 → 1 0 1 1 0 1 0 1
1 1 0 0 1 1 1 1

No carry then the result is -ve number and take 2's complement to obtained result

Subtract 14 from 25 using the 8-bit 1's complement?

Step 1:- 1's complement of second number (14)

Step 2:- Add first number to the 1's complement result of second number.

Step 3:- If carry is generated make it as end around carry.

25 - 14

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

128 64 32 16 8 4 2 1

14 → 0 0 0 0 1 1 1 0

1's complement of 14 → 1 1 1 1 0 0 0 1

Add (25) → 0 0 0 1 1 0 0 1

(+) 1 1 1 1

1 | 0 0 0 0 1 0 1 0

0 0 0 0 1 0 1 0

+ 1

0 0 0 0 1 0 1 1 (+11)

MSB is '0' so result is +ve and is in pure Binary form

→ Note: If No carry so result is -ve.

so we have to take 1's complement of result

Result 0 0 0 0 1 0 1 1

→ 1 1 1 1 0 1 0 0

14
-25
-11

* BCD code *

→ BCD is an abbreviation for binary coded decimal.

→ BCD is a numeric code in which each digit of a decimal number is represented by a separate group of 4-bits. The most common BCD code is 8-4-2-1 BCD

→ It is called 8-4-2-1 BCD because the weights associated with 4 bits are 8-4-2-1 from left to right. This means that, bit-3 has weight 8, bit-2 has weight 4 bit-1 has weight 2 and bit-0 has weight 1.

The table shows the 4-bit 8-4-2-1 BCD code used to represent a single decimal digit.

Decimal Digit	BCD code 8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

been

→ States 1010, 1011, 1100, 1101, 1110, 1111 are illegals of BCD.

→ In multidigit coding, each decimal digit is individually coded with 8-4-2-1 BCD code, as shown in the fig.

Total 8-bits are required to encode 58_{10} in 8-4-2-1 BCD.

Ex:

Decimal	5	8
8-4-2-1 code	0101	1000

Ex: Represent the following decimal number in BCD

$(13597)_{10} = (0001\ 0011\ 0101\ 1001\ 0111)_{BCD}$

$(93286)_{10} = (1001\ 0011\ 0010\ 1000\ 0110)_{BCD}$

* Excess-3 code *.

→ The Excess-3 code can be derived from the natural BCD code by adding 3 to each coded number

→ for example, decimal 12 can be represented in BCD as 0001 0010. Now adding 3 to each digit we get Excess-3 code as 0100 0101 (12 in decimal) it is a Non-weighted code.

Decimal

The table shows Excess-3 codes to represent single decimal digit. It is sequential code because we get any code word by adding binary 1 to its previous code word as shown in table.

Decimal digit	Excess-3 code
0	0 0 1 1
1	0 1 0 0
2	0 1 0 1
3	0 1 1 0
4	0 1 1 1
5	1 0 0 0
6	1 0 0 1
7	1 0 1 0
8	1 0 1 1
9	1 1 0 0

* Gray code *

→ The Gray code is a non-weighted code, it is not suitable for arithmetic operations. It is not a BCD code. It is a cyclic code, because successive code words in this code differ in one bit position only i.e it is a unit distance code.

* Gray to Binary conversion *

→ The Most significant Bit (MSB) of the binary number is the same as the MSB of the gray code number. So write down MSB as it is

→ To obtain the next binary digit, perform an exclusive-OR-operation between the bit just written down and the next gray code bit. Write down the result

→ Repeat step 2 until gray code bits have been exclusive-ORed with binary digits

Ex: Convert to binary 1001 to Gray code

Binary code: 1 ⊕ 0 ⊕ 0 ⊕ 1

Gray code: 1 1 0 1

* Binary to gray code conversion *

→ The MSB of the gray code is the same as the MSB of the binary number. So write down MSB as it is

→ To obtain the next gray digit, perform an exclusive-OR operation between previous and current binary bit. Write down the result.

→ Repeat step 2 until all binary bits have been ex-ORed with their previous ones.

Ex: Gray to Binary code conversion:

Convert 00100101111 .

Gray code: 0 0 1 0 0 1 0 1 1 1 1

Binary code: 0 0 1 1 1 0 0 1 0 1 0

$(00100101111)_{Gray} = (00111001010)_{Binary}$

Five Bit codes

→ Some 5-bit BCD codes that have parity contained within each code for ease of error detection are shown in table. The 63210 is a weighted code. It has the useful error-detecting property that there are exactly two 1's in each code group. This code is used for storing data on magnetic tapes.

→ The 2 out of 5 code is a non-weighted code. It also has exactly two 1's in each code group. This code is used in the telephone and communication industries. At the receiving end, the receiver can check the no of 1's in each character received. The shift counter code, also called the Johnson code, has the bit patterns by a 5-bit Johnson counter. The 5 1111 code is similar to the Johnson code but is weighted

Five bit BCD code

Decimal	63210	2 out of 5	Shift counter	5 1111
0	00110	00011	00000	00000
1	00011	00101	00001	00001
2	00101	00110	00011	00011
3	01001	01001	00111	00111
4	01010	01010	01111	01111
5	01100	01100	11111	10000
6	10001	10001	11110	11000
7	10010	10010	11100	11100
8	10100	10100	11000	11110
9	11000	11000	10000	11111

ASCII

→ Computer works on character data and this data is not alphabet but numeric values, punctuation, spaces etc. are also character data. In Common language whatever is on keyboard (except Shift, Caps lock key) are character data. As you know computer operates on binary values so these character are also represented in binary values. Most Common Character coding used in india are ASCII, ISCII, and UNICODE.

→ ASCII stands for American standard code for information interchange. Computers can only understand numbers. So an ASCII code is the numerical representation of a character such as 'A' has numerical value as 65. Some other characters and it's equivalent ASCII values are following

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J

ASCII	Hex	Symbol
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

EBCDIC
→ Extended binary coded decimal interchange code.

→ From the above table we can see, characters have equivalent ASCII values. Similarly, each character has ASCII value.

→ ASCII value for a to z is from 97 to 122 i.e a=97, z=122.

→ Error:

When the digital information in the binary form is transmitted from one system to another system an error may occur. This means a signal corresponding to '0' may change to '1' (or) vice-versa due to presence of noise. To maintain the data integrity between transmitter and receiver, extra bit (or) more than one bit are added in the data. These extra bit's allow the detection and some times correction of error in the data. The data along with the extra bit / bits forms the code. Codes which allow only error detection are called error detecting codes, and codes which allow error detection and correction are called

error detecting and correcting codes:-

Ex: parity codes (which only detects the errors)

Hamming codes (which detects & corrects the error)

Hamming codes

→ Hamming code is used to detecting two error and correct single error.

→ They are mainly 3 types of Hamming code.

① 7-Bit Hamming code:-

In 7 bit hamming code 3 is parity, 4 is data.

7 Bit → P_1 P_2 D_3 P_4 D_5 D_6 D_7

parity	2^2	2^1	2^0	
	4	2	1	→ parity places
0	0	0	0	
1	0	0	1	
2	0	1	0	
3	0	1	1	
4	1	0	0	
5	1	0	1	
6	1	1	0	
7	1	1	1	

$P_1 = \{1, 3, 5, 7\}$

$P_2 = \{2, 3, 6, 7\}$

$P_3 = \{4, 5, 6, 7\}$

② 12 Bit Hamming code:-

12 Bit P_1 P_2 D_3 P_4 D_5 D_6 D_7 P_8 D_9 D_{10} D_{11} D_{12}

parity	2^3	2^2	2^1	2^0
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0

	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

$P_1 = \{1, 3, 5, 7, 9, 11\}$

$P_2 = \{2, 3, 6, 7, 10, 11\}$

$P_4 = \{4, 5, 6, 7, 12\}$

$P_8 = \{8, 9, 10, 11, 12\}$

③ 15 Bit Hamming code :-

15 Bit P_1 P_2 D P_4 D_5 D_6 D_7 P_8 D_9 D_{10} D_{11} D_{12} D_{13} D_{14} D_{15}

Parity –

	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)
0 –	0	0	0	0
1 –	0	0	0	1
2 –	0	0	1	0
3 –	0	0	1	1
4 –	0	1	0	0
5 –	0	1	0	1
6 –	0	1	1	0
7 –	0	1	1	1
8 –	1	0	0	0
9 –	1	0	0	1
10 –	1	0	1	0
11 –	1	0	1	1
12 –	1	1	0	0
13 –	1	1	0	1
14 –	1	1	1	0
15 –	1	1	1	1

$P_1 = \{1, 3, 5, 7, 9, 11, 13, 15\}$

$P_2 = \{2, 3, 6, 7, 10, 11, 14, 15\}$

$P_4 = \{4, 5, 6, 7, 12, 13, 14, 15\}$

$P_8 = \{8, 9, 10, 11, 12, 13, 14, 15\}$

even parity scheme

· 7Bit

P_1	P_2	D_3	P_4	D_5	D_6	D_7
		1		1	0	1

Parity 2^2 2^1 2^0

$P_1 = \{1, 3, 5, 7\}$

$P_2 = \{2, 3, 6, 7\}$

$P_3 = \{4, 5, 6, 7\}$

② Encode the data 11100101 in 12 Bit Hamming code.
odd parity scheme.

12 Bit

P_1	P_2	D_3	P_4	D_5	D_6	D_7	P_8	D_9	D_{10}	D_{11}	D_{12}
		1		1	1	0		0	1	0	1

$P_1 = \{1, 3, 5, 7, 9, 11, 13, 15\}$

$P_2 = \{2, 3, 6, 7, 10, 11, 14, 15\}$

$P_4 = \{4, 5, 6, 7, 12, 13, 14, 15\}$

$P_8 = \{8, 9, 10, 11, 12, 13, 14, 15\}$

Eg: The message below coded in the 7 Bit hamming code is is transmitted to through a noisy channel decode the message Assuming atmost a single error occur in each code word

1001001

7Bit

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

$P_1 = \{1, 3, 5, 7\}$ → No error → $C_1 = 0$

$P_2 = \{2, 3, 6, 7\}$ → error → $C_2 = 1$

$P_3 = \{4, 5, 6, 7\}$ → No error → $C_3 = 0$

Take the values from bottom to top, so $(010)_2$ = error in 2nd place complement to 2nd Bit

Correct :- 1101001

* Binary storage AND Registers *

→ A storage technique in which the information is represented and stored in the binary form i.e using 0's and 1's is referred to as binary storage. Binary storage is an important concept in digital electronics. as all the digital systems operate using binary logic

→ Binary storage can be implemented using various types of storage devices like semiconductor circuits, magnetic tapes, optical disks etc. overall, binary storage forms the foundation for storing and manipulating digital information.

→ Depending on the characteristics & applications, there are various types of binary storage technologies

* Semiconductor Electronic circuits. eg:- transistors
* Magnetic Media. eg:- magnetic tape / harddisks
* Optical DISKS. eg :- light sensitive materials, plastic

* Binary storage register *

→ In a digital electronic system, the binary storage register is a fundamental storage unit used to store and process binary data.

→ A Binary storage register consists of a collection of flip flops, where each flip flop can store 1 bit of information in binary form.

→ The size (or) storage capacity of a binary storage register is determined by the no. of

flip-flops used in the logic circuit.
→ For example, if a register consists of 8 flip flops then its storage capacity will be 8-bits.
→ The binary storage register is that it stores data temporarily.
→ Some common types of binary storage registers used in various digital systems are listed below.

* Serial-in Serial-out (SISO) Register
* Serial-in parallel-out (SIPO) Register
* parallel-in Serial-out (PISO) Register
* parallel-in parallel-out (PIPO) Register.

* Binary logic *

→ Binary logic is the basis of electronic systems such as computers and cell phones. It works on 0's and 1's. It involves addition, subtraction, multiplication, division of zeros and ones.
→ It includes logic gate functions, AND, OR, and NOT which translates input signals in to specific output.
→ It facilitates the computing, robotics and other electronic applications.
→ Computer stores & processes two basic types of data viz.. character and number. The character type data includes alphabets and some special symbols. can be done. Eg: Student name → character data.
roll no → numeric data.

Boolean algebra*

→ Boolean algebra is a mathe matical system that defines a series of logic operations (AND, OR, NOT) performed on sets of variables (a, b, c ...) when stated in this form. The expression is called a boolean equation (or) switching equation.

Terminology:-

→ Variable :- The symbol which represents an arbitary elements of an Boolean algebra is known as variable. Any single variable (or) a function of several variables can have either a '1' (or) '0' value

→ Constant :- In expression $Y = A + 1$, 'A' is variable and '1' is constant, which has a fixed value.

→ Literal :- Each occurance of a variable in Boolean function either in complement (or) Uncomplemented form is called a 'Literal'

→ Boolean function :- Boolean expressions are constructed by connecting the boolean constants and variables with the boolean operations.

Eg: $f(A, B, C) = (A + \bar{B})C$ (or) $f = (\bar{A} + \bar{B})C$

* Axiomatic definitions of Boolean algebra *

The postulates of mathematical system form the basic assumption from which it is possible to deduce the theorem, laws and properties of the system. Boolean algebra is formulated by a set of elements, together with two binary operators.

→ '+' and '.' provided that the given postulates are satisfied.

→ * Closure property *

When two binary elements are operated by operator either '+' or '.', the result must be a unique binary element.

→ * Identity property *

$$A \cdot 1 = 1 \cdot A = A$$

→ * Commutative property *

$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

→ * Distributive property *

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

→ * Associative property *

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

→ * Complement Property *

$$A \cdot \overline{A} = 0$$
$$A + \overline{A} = 1$$

→ * Demorgan's property *

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

→ * Idempotency property *

$$A + A = A$$
$$A \cdot A = A$$

→ * Absorption property *

$$A + A \cdot B = A$$
$$A \cdot (A+B) = A$$

→ * Involution property *

$$\overline{\overline{A}} = A$$

Basic theorems & properties of Boolean algebra

The following are the basic Huntingtons postulates and theorems of Boolean algebra.

Postulates

	a	b
Postulates 1	Result of each operation either '0' (or) '1' $\in B$	
postulates 2 (Identity)	$A+0=A$	$A.1=A$
postulates 3 (commutative)	$A+B=B+A$	$A.B=B.A$
postulates 4 (Distributive)	$A.(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$
postulates 5 (complement)	$A+\bar{A}=1$	$A.\bar{A}=0$

Theorems

Theorem 1 (Idempotency)	$A+A=A$	$A.A=A$
Theorem 2	$A+1=A$	$A.0=0$
Theorem 3 (Involution)	$(\bar{A})=A$	
Theorem 4 (Absorption)	$A+AB=A$	$A.(A+B)=A$
Theorem 5 DLR	$A+\bar{A}B=A+B$	$A.(\bar{A}+B)=AB$
Theorem 6 (Associative)	$A+(B+C)=(A+B)+C$	$A.(BC)=(A.B)$

* Boolean function (or) Switching function :

→ Boolean expressions are constructed by connecting the boolean constants and variables with the boolean operations. These Boolean expressions are also known as Boolean formulas

Eg: $f(A,B,C) = (A+\bar{B})C$ (or) $f = (A+\bar{B})C$

Based on the structure of Boolean expression, it can be categarized in different formulae.

Consider $f(A,B,C,D) = A+\bar{B}C+A\bar{C}\bar{D}$

Product terms.

a product is group of literals which are ANDed together. They appear in either complement (or) un-complemented form.

Consider another form.

$f(A,B,C,D) = (B+\bar{D}).(A+\bar{B}+C).(\bar{A}+C)$

Sum terms.

a sum term is group of literals which are ORed together they appear either in complement (or) uncomplemented form. Hence, these literals and terms are arranged in one of the two forms.

a) Sum of product form (SOP)

b) product of sum form (POS)

* SOP * Sum of product form:

The words sum and product are derived from the symbolic representations of the OR and AND functions by '+' and '.' respectively. A product term is any group of literals that are ANDed together

for example, ABC, xy and so on

Sum

Ex: $f(A,B,C) = ABC + A\bar{B}\bar{C}$

product terms

* POS * (product of Sum form)

A product of sum is any groups of sum terms ANDed together. sum terms are formed by ORing literals together.

product

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

Sum terms.

* (Canonical form (Standard SOP & POS form)) *

→ In normal SOP and POS form, all the terms donot involue all the literals; for example $AB + A\bar{B}C$, here in the first term 'c' is missing.

→ If each term in SOP form contains all the literals then the SOP form ~~contains all the literals then it~~ is known as standard (or) Canonical SOP form.

→ If each term in POS form contains all the literals then the POS form is known as standard POS (or) Canonical POS form.

Steps to convert SOP to Standard SOP*

1. Find the missing literal in each product term
2. Perform AND operation with a term, which is formed with missing literal and it's complement which are ORed together
3. Expand the terms
4. Reduce the expression by omitting repeated terms.

Steps to convert POS to standard POS form

1. find missing literals in each sum term
2. OR each sum term having missing literal with term formed by ANDing the literal and it's complement
3. Expand the terms.
4. Reduce the expression by omitting repeated Sumterms

*M Notations (Minterms and maxterms):

Each individual term in standard SOP form is called minterm and each individual term in standard POS form is called maxterm.

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

Note:- $f(A,B,C) = \Sigma m(0,1,3,4,6,7) = \pi M(2,5)$

Σm is used to represent SOP form

πM is used to represent POS form.

Every Boolean expression must be reduced to as simple as possible before realization, because every logic operation the expression represents a corresponding element of Hardware. Realization of digital circuit with the minimal expression results in reduction of cost and complexity.

The Boolean expression reduction procedure should follows following

i) Multiply all variables to remove paranthesis

ii) Look for identical terms only one of those terms retained and all other removed.

iii) Look for a variable and it's negation in the same term

i.e $BA + B\bar{A} = A$.

(iv) Look for pairs of terms that are identical except for one variable which may be missing in one of the terms. ie $AB\bar{C} + AB \Rightarrow AB(1+\bar{C}) = AB$

(v) Look for pairs of terms which have the same variables, with one (or) more variables complemented

ie $AB\bar{C} + ABC \Rightarrow AB(C+\bar{C}) = AB \cdot 1 = AB$

→ SOP:- In sum of product each term is known as minterm.

→ Σm is used to represent SOP form

→ $f(A,B,C) = \Sigma m(0,1,3,4,6,7) = \Pi M(2,5)$

→ POS :- In product of sum each term is known as maxterm.

→ ΠM is used to represent POS form.

→ $f(A,B,C) = \Pi(2,5)$

Reduce the literals in given expression

① $A(\bar{A}+B)$

$= A\bar{A}+AB$

$= 0+AB$

$= AB$

② $(A+B)(\bar{A}+B)$

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

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

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

$= B(1)+B$

$= B+B$

$= B.$

$A+\bar{A}=1$

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

③ Proof

$A+A=A$

$= A+A(1)$

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

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

$= A\cdot A+0+A\cdot A+0 \quad (\because A\cdot A=A)$

$= A+A$

$= A$

④ Proof $A+1=1$

$= (A+1)\cdot 1$

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

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

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

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

$= 1$

Proof

$$A \cdot A = A$$

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

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

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

$$= A \cdot 1$$

$$= A.$$

* SOP (Sum of product) :-

→ It is expressed as (m)

→ The notation for expressing SOP is Σ (Sigma)

→ In SOP prime is equal to zero, unprime is equal to one

→ prime - complemented

→ unprime - uncomplemented

* max terms (POS) (product of sums) :-

→ The terms in POS is known as max terms

→ It is expressed as M.

→ Notation for POS is $\Pi(Pi)$

→ In POS prime is equal to one, unprime is equal to zero.

A	B		
0	0	$A+B$	M_0
0	1	$A+\bar{B}$	M_1
1	0	$\bar{A}+B$	M_2
1	1	$\bar{A}+\bar{B}$	M_3

A	B	C		
0	0	0	$A+B+C$	M_0
0	0	1	$A+B+\bar{C}$	M_1
0	1	0	$A+\bar{B}+C$	M_2
0	1	1	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$\bar{A}+B+C$	M_4
1	0	1	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$\bar{A}+\bar{B}+C$	M_6
1	1	1	$\bar{A}+\bar{B}+\bar{C}$	M_7

find the min terms for the given expression

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

001 010 011 100 111

$= m_1 \cdot m_2 \quad m_3 \quad m_4 \quad m_7$

$= \Sigma m(1,2,3,4,7)$

Max terms POS M Π

$A = 0$

$\bar{A} = 1$

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

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

0 1 0 — 1 1 1 — 0 0 0

M_2 — M_7 — M_0

$F(x,y,z) = \Pi M(0,2,7)$

→ Boolean expression can be expressed in SOP, POS.

→ These are divided into 2 types

(1) Canonical (2) standard

(1) Canonical form: In canonical form each and every term will consisting of all the variables given in the function

(2) standard form: In standard form some of the terms consisting of all the variables and some of the terms may miss one (or) more variables

Problems

(1) $f = AB + AC + AB\bar{C}$ write the expression in canonical form and also write the minterms, maxterms from the expression.

$f = AB + AC + AB\bar{C}$

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

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

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

1 1 1 — 1 1 0 — 1 1 1 — 1 0 1 — 1 1 0

m_7 — m_6 — m_7 — m_5 — m_6

$\Sigma m(5,6,7)$

$\Pi M(0,1,2,3,4)$

2^3 combinations

$(0,1,2,3,4,5,6,7)$

Ex: Reduce the expression $A[B+\bar{C}\,\overline{(AB+A\bar{C})}]$

Sol: The Given expression is $A[B+\bar{C}\,\overline{(AB+A\bar{C})}]$

$= A[B+\bar{C}(\overline{AB}\cdot\overline{A\bar{C}})]$ [De morgans law]

$= A[B+\bar{C}(\bar{A}+\bar{B})\cdot(\bar{A}+\overline{\bar{C}})]$ [Demorgans law]

$= A[B+\bar{C}(\bar{A}+\bar{B})\cdot(\bar{A}+C)]$ [Involution law]

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

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

Simplify

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

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

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

$= AB+0+0$

$= AB$

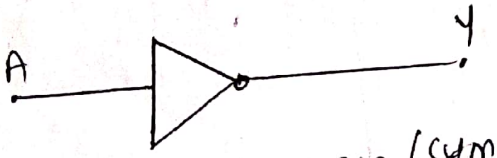
→ * Digital logic gates *

Logic gates are the basic elements that make up a digital system. The electronic gate is circuit that is able to operate on a no. of binary inputs in order to perform a particular logic functions. The types of gates available are

1. NOT
2. AND
3. OR
4. NAND
5. NOR
6. Exclusive-OR
7. Exclusive-NOR

→ NOT Gate (Inverter) :-

The output is complement of input



Logic diagram (symbol)

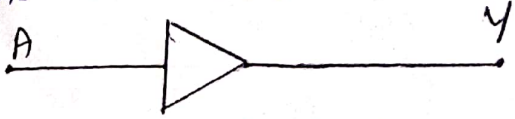
Boolean expression $Y = \overline{A}$

Truth table

Input (A)	output (Y)
0	1
1	0

→ Buffer :-

The output is same as input



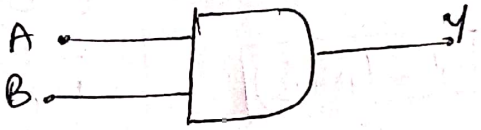
Logic diagram

Truth table

Input (A)	output (Y)
0	0
1	1

→ AND gate

The output is high only when all the inputs are high.



Logic diagram

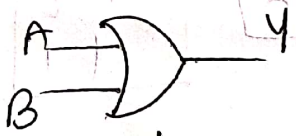
Boolean expression $Y = A \cdot B$

Truth table

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

→ OR gate :-

The output is high when any one of the inputs is high.



logic diagram

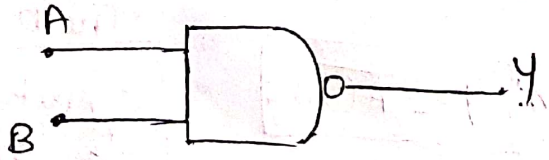
Boolean expression $Y = A + B$

Truth table

Inputs		outputs
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

→ NAND gate :-

The output is high only when one of the inputs is low.



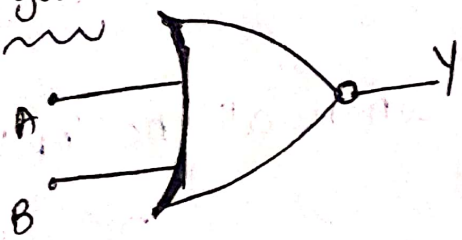
logic diagram

Boolean expression $Y = \overline{A \cdot B}$

Truth table

Inputs		output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate :



logic diagram

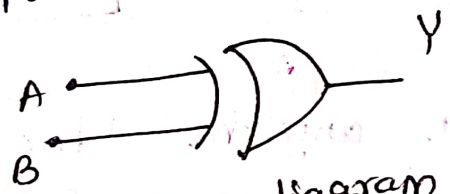
Boolean expression $\boxed{Y = \overline{(A+B)}}$

Truth table

Input		output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Exclusive OR (EX-OR) gate :-

The output is high only when odd number of inputs are high



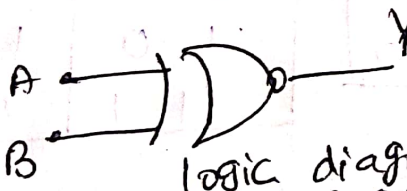
logic diagram

Boolean Expression $\boxed{Y = A \oplus B}$ $Y = \bar{A}B + A\bar{B}$

Truth table

Input		output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive NOR (EX-NOR) Gate :



logic diagram

The output is high only when even number of ones at the input are all inputs are high.

Boolean expression $\boxed{Y = \overline{A \oplus B}}$

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

Truth table

Input		output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1