

# MODULE- 1: NUMBER SYSTEMS & CODES

## Syllabus:

- Introduction about digital system
- Philosophy of number systems
- Complement representation of negative numbers
- Binary arithmetic
- Binary codes
- Error detecting & error correcting codes
- Hamming codes

## INTRODUCTION ABOUT DIGITAL SYSTEM

A Digital system is an interconnection of digital modules and it is a system that manipulates discrete elements of information that is represented internally in the binary form.

Now a day's digital systems are used in wide variety of industrial and consumer products such as automated industrial machinery, pocket calculators, microprocessors, digital computers, digital watches, TV games and signal processing and so on.

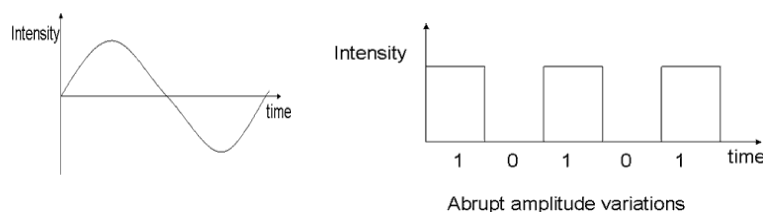
### Characteristics of Digital systems

- Digital systems manipulate discrete elements of information.
- Discrete elements are nothing but the digits such as 10 decimal digits or 26 letters of alphabets and so on.
- Digital systems use physical quantities called signals to represent discrete elements.
- In digital systems, the signals have two discrete values and are therefore said to be binary.
- A signal in digital system represents one binary digit called a bit. The bit has a value either 0 or 1.

### Analog systems vs Digital systems

Analog system process information that varies continuously i.e; they process time varying signals that can take on any values across a continuous range of voltage, current or any physical parameter.

Digital systems use digital circuits that can process digital signals which can take either 0 or 1 for binary system.



## Advantages of Digital system over Analog system

### 1. Ease of programmability

The digital systems can be used for different applications by simply changing the program without additional changes in hardware.

### 2. Reduction in cost of hardware

The cost of hardware gets reduced by use of digital components and this has been possible due to advances in IC technology. With ICs the number of components that can be placed in a given area of Silicon are increased which helps in cost reduction.

### 3. High speed

Digital processing of data ensures high speed of operation which is possible due to advances in Digital Signal Processing.

### 4. High Reliability

Digital systems are highly reliable one of the reasons for that is use of error correction codes.

### 5. Design is easy

The design of digital systems which require use of Boolean algebra and other digital techniques is easier compared to analog designing.

### 6. Result can be reproduced easily

Since the output of digital systems unlike analog systems is independent of temperature, noise, humidity and other characteristics of components the reproducibility of results is higher in digital systems than in analog systems.

## Disadvantages of Digital Systems

- Use more energy than analog circuits to accomplish the same tasks, thus producing more heat as well.
- Digital circuits are often fragile, in that if a single piece of digital data is lost or misinterpreted the meaning of large blocks of related data can completely change.
- Digital computer manipulates discrete elements of information by means of a binary code.
- Quantization error during analog signal sampling.

## NUMBER SYSTEM

Number system is a basis for counting various items. Modern computers communicate and operate with binary numbers which use only the digits 0 & 1. Basic number system used by humans is Decimal number system.

For Ex: Let us consider decimal number 18. This number is represented in binary as 10010.

We observe that binary number system takes more digits to represent the decimal number. For large numbers we have to deal with very large binary strings. So this fact gave rise to three new number systems.

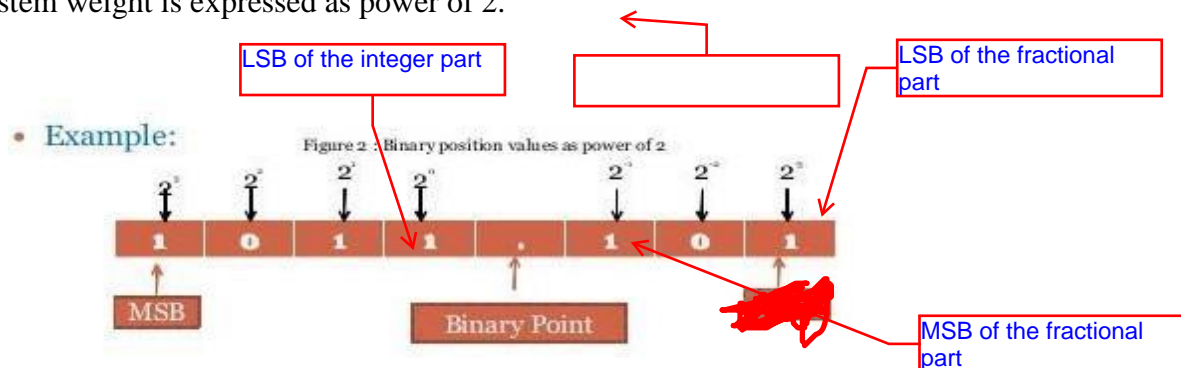
- i) Octal number systems
- ii) Hexa Decimal number system
- iii) Binary Coded Decimal number (BCD) system

To define any number system we have to specify

- Base of the number system such as 2, 8, 10 or 16.
- The base decides the total number of digits available in that number system.
- First digit in the number system is always zero and last digit in the number system is always base-1.

### Binary number system:

The binary number has a radix of 2. As  $r = 2$ , only two digits are needed, and these are 0 and 1. In binary system weight is expressed as power of 2.



The left most bit, which has the greatest weight is called the Most Significant Bit (MSB). And the right most bit which has the least weight is called Least Significant Bit (LSB).

For Ex:  $1001.01_2 = [(1) \times 2^3] + [(0) \times 2^2] + [(0) \times 2^1] + [(1) \times 2^0] + [(0) \times 2^{-1}] + [(1) \times 2^{-2}]$

$$1001.01_2 = [1 \times 8] + [0 \times 4] + [0 \times 2] + [1 \times 1] + [0 \times 0.5] + [1 \times 0.25]$$

$$1001.01_2 = 9.25_{10}$$

## Decimal Number system

The decimal system has ten symbols: 0,1,2,3,4,5,6,7,8,9. In other words, it has a base of 10.

## Octal Number System

Digital systems operate only on binary numbers. Since binary numbers are often very long, two shorthand notations, octal and hexadecimal, are used for representing large binary numbers. Octal systems use a base or radix of 8. It uses first eight digits of decimal number system. Thus it has digits from 0 to 7.

## Hexa Decimal Number System

The hexadecimal numbering system has a base of 16. There are 16 symbols. The decimal digits 0 to 9 are used as the first ten digits as in the decimal system, followed by the letters A, B, C, D, E and F, which represent the values 10, 11, 12, 13, 14 and 15 respectively.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

## Number Base conversions

The human beings use decimal number system while computer uses binary number system. Therefore it is necessary to convert decimal number system into its equivalent binary.

- i) Binary to octal number conversion
- ii) Binary to hexa decimal number conversion

The binary number: 001 010 011 000 100 101 110 111  
The octal number: 1 2 3 0 4 5 6 7

The binary number: 0001 0010 0100 1000 1001 1010 1101 1111  
The hexadecimal number: 1 2 5 8 9 A D F

- iii) Octal to binary Conversion

Each octal number converts to 3 binary digits

Code
0 - 000
1 - 001
2 - 010
3 - 011
4 - 100
5 - 101
6 - 110
7 - 111

To convert  $653_8$  to binary, just substitute code:

6 5 3  
↓ ↓ ↓  
110 101 011

4 F D 7  
↓ ↓ ↓ ↓  
0100 1111 1101 0111

- iv) Hexa to binary conversion
- v) Octal to Decimal conversion

Ex: convert  $4057.06_8$  to octal

$$\begin{aligned} &= 4 \times 8^3 + 0 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 + 0 \times 8^{-1} + 6 \times 8^{-2} \\ &= 2048 + 0 + 40 + 7 + 0 + 0.0937 \end{aligned}$$

$$=2095.0937_{10}$$

vi) Decimal to Octal Conversion

Ex: convert  $378.93_{10}$  to octal

**$378_{10}$  to octal:** Successive division:

$$\begin{array}{r} 8 \overline{) 378} \\ \underline{8 \mid 47} \quad \text{---} \quad 2 \\ \underline{8 \mid 5} \quad \text{---} \quad 7 \quad \uparrow \\ 0 \quad \text{---} \quad 5 \end{array}$$

$$=572_8$$

$0.93_{10}$  to octal :

$$0.93 \times 8 = 7.44$$

$$0.44 \times 8 = 3.52$$

$$0.53 \times 8 = 4.16$$

$$0.16 \times 8 = 1.28$$

$$=0.7341_8$$

$$378.93_{10} = 572.7341_8$$

vii) Hexadecimal to Decimal Conversion

Ex:  $5C7_{16}$  to decimal

$$= (5 \times 16^2) + (C \times 16^1) + (7 \times 16^0)$$

$$= 1280 + 192 + 7$$

$$= 147_{10}$$

viii) Decimal to Hexadecimal Conversion

Ex:  $2598.6751_{10}$

$$\begin{array}{r} 16 \overline{) 2598} \\ \underline{16 \mid 162} \quad -6 \\ 10 \quad \quad -2 \end{array}$$

$$= A26_{(16)}$$

$$0.675_{10} = 0.675 \times 16 \rightarrow 10.8$$

$$= 0.800 \times 16 \rightarrow 12.8 \quad \downarrow$$

$$= 0.800 \times 16 \rightarrow 12.8$$

$$= 0.800 \times 16 \rightarrow 12.8$$

$$= 0.ACCC_{16}$$

$$2598.675_{10} = A26.ACCC_{16}$$

ix) Octal to hexadecimal conversion:

The simplest way is to first convert the given octal no. to binary & then the binary no. to hexadecimal.

Ex:  $756.603_8$

7	5	6	.	6	0	3
111	101	110	.	110	000	011
0001	1110	1110	.	1100	0001	1000
1	E	E	.	C	1	8

x) Hexadecimal to octal conversion:

First convert the given hexadecimal no. to binary & then the binary no. to octal.

Ex:  $B9F.AE_{16}$

B	9	F	.	A	E		
1011	1001	1111	.	1010	1110		
101	110	011	111	.	101	011	100
5	6	3	7	.	5	3	4

$$= 5637.534$$

### Complements:

In digital computers to simplify the subtraction operation & for logical manipulation complements are used. There are two types of complements used in each radix system.

- The radix complement or  $r$ 's complement
- The diminished radix complement or  $(r-1)$ 's complement

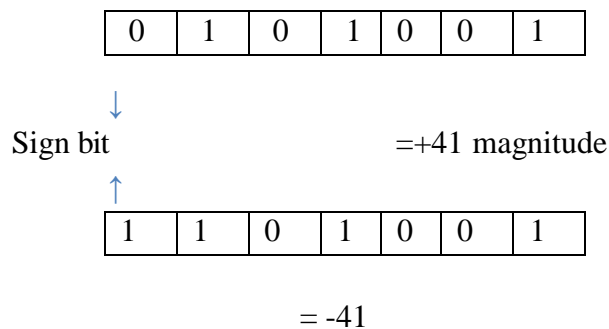
## Representation of signed no.s binary arithmetic in computers:

- Two ways of rep signed no.s
  1. Sign Magnitude form
  2. Complement form
- Two complimented forms
  1. 1's complement form
  2. 2's complement form

Advantage of performing subtraction by the complement method is reduction in the hardware. (instead of addition & subtraction only adding ckt's are needed.)  
i.e, subtraction is also performed by adders only.

Instead of subtracting one no. from other the compliment of the subtrahend is added to minuend. In sign magnitude form, an additional bit called the sign bit is placed in front of the no. If the sign bit is 0, the no. is +ve, If it is a 1, the no is \_ve.

Ex:



Note: manipulation is necessary to add a +ve no to a -ve no

### Representation of signed no.s using 2's or 1's complement method:

If the no. is +ve, the magnitude is rep in its true binary form & a sign bit 0 is placed in front of the MSB. If the no is \_ve , the magnitude is rep in its 2's or 1's compliment form & a sign bit 1 is placed in front of the MSB.

Ex:

Given no.	Sign mag form	2's comp form	1's comp form
01101	+13	+13	+13
010111	+23	+23	+23
10111	-7	-7	-8
1101010	-42	-22	-21



### Special case in 2's comp representation:

Whenever a signed no. has a 1 in the sign bit & all 0's for the magnitude bits, the decimal equivalent is  $-2^n$ , where n is the no of bits in the magnitude .

Ex: 1000= -8 & 10000=-16

### Characteristics of 2's compliment no.s:

Properties:

1. There is one unique zero
2. 2's comp of 0 is 0
3. The leftmost bit can't be used to express a quantity . it is a 0 no. is +ve.
4. For an n-bit word which includes the sign bit there are  $(2^{n-1}-1)$  +ve integers,  $2^{n-1}$  -ve integers & one 0 , for a total of  $2^n$  unique states.
5. Significant information is contained in the 1's of the +ve no.s & 0's of the -ve no.s
6. A -ve no. may be converted into a +ve no. by finding its 2's comp.

### Signed binary numbers:

Decimal	Sign 2's comp form	Sign 1's comp form	Sign mag form
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0011	0011	0011
+0	0000	0000	0000

-0	--	1111	1000
-1	1111	1110	1001
-2	1110	1101	1010
-3	1101	1100	1011
-4	1100	1011	1100
-5	1011	1010	1101
-6	1010	1001	1110
-7	1001	1000	1111
8	1000	--	--

## Methods of obtaining 2's comp of a no:

- In 3 ways
    1. By obtaining the 1's comp of the given no. (by changing all 0's to 1's & 1's to 0's) & then adding 1.
    2. By subtracting the given n bit no N from  $2^n$
    3. Starting at the LSB , copying down each bit upto & including the first 1 bit encountered , and complimenting the remaining bits.
- Ex: Express -45 in 8 bit 2's comp form

+45 in 8 bit form is 00101101

### I method:

1's comp of 00101101 & the add 1

00101101

11010010

+1

— — — — — — — — — —

11010011

is 2's comp form

### II method:

Subtract the given no. N from  $2^n$

$2^n = 100000000$

Subtract 45 = -00101101

+1

— — — —

11010011

is 2's comp

### III method:

Original no: 00101101

Copy up to First 1 bit 1

Compliment remaining : 1101001

—————

bits

11010011

**Ex:**

-73.75 in 12 bit 2's comp form

I method

```
01001001.1100
10110110.0011
+1
```

10110110.0100 is 2's

II method:

$2^8 = 100000000.0000$

Sub 73.75 = -01001001.1100

10110110.0100 is 2's comp

III method :

Original no : 01001001.1100

Copy up to 1'st bit 100

Comp the remaining bits: 10110110.0

10110110.0100

### 2's compliment Arithmetic:

- The 2's comp system is used to rep -ve no.s using modulus arithmetic . The word length of a computer is fixed. i.e, if a 4 bit no. is added to another 4 bit no . the result will be only of 4 bits. Carry if any , from the fourth bit will overflow called the Modulus arithmetic.

Ex: 1100 + 1111 = 1011

- In the 2's compl subtraction, add the 2's comp of the subtrahend to the minuend . If there is a carry out , ignore it , look at the sign bit I.e, MSB of the sum term .If the MSB is a 0, the result is positive.& it is in true binary form. If the MSB is a 1 ( carry in or no carry at all) the result is negative.& is in its 2's comp form. Take its 2's comp to find its magnitude in binary.

**Ex:** Subtract 14 from 46 using 8 bit 2's comp arithmetic:

```
+14   = 00001110
-14   = 11110010      2's comp

+46   = 00101110
-14   = +11110010     2's comp form of -14
```

$$\begin{array}{r} \text{-32} \quad \overline{(1)00100000} \quad \text{ignore carry} \end{array}$$

Ignore carry, The MSB is 0. so the result is +ve. & is in normal binary form. So the result is +00100000=+32.

**EX:** Add -75 to +26 using 8 bit 2's comp arithmetic

$$\begin{array}{r} +75 = 01001011 \\ -75 = 10110101 \quad \text{2's comp} \\ +26 = 00011010 \\ -75 = +10110101 \quad \text{2's comp form of -75} \\ \hline \text{-49} \quad \overline{11001111} \quad \text{No carry} \end{array}$$

No carry, MSB is a 1, result is -ve & is in 2's comp. The magnitude is 2's comp of 11001111. i.e, 00110001 = 49. so result is -49

**Ex:** add -45.75 to +87.5 using 12 bit arithmetic

$$\begin{array}{r} +87.5 = 01010111.1000 \\ -45.75 = +11010010.0100 \end{array}$$

$$\begin{array}{r} \text{-41.75} \quad (1)00101001.1100 \text{ ignore carry} \\ \text{MSB is 0, result is +ve.} = +41.75 \end{array}$$

### 1's compliment of n number:

- It is obtained by simply complimenting each bit of the no., & also, 1's comp of a no, is subtracting each bit of the no. from 1. This complemented value rep the -ve of the original no. One of the difficulties of using 1's comp is its rep of zero. Both 00000000 & its 1's comp 11111111 rep zero.
- The 00000000 called +ve zero & 11111111 called -ve zero.

Ex: -99 & -77.25 in 8 bit 1's comp

$$\begin{array}{r} +99 = 01100011 \\ -99 = 10011100 \end{array}$$

$$\begin{array}{r} +77.25 = 01001101.0100 \\ -77.25 = 10110010.1011 \end{array}$$

### 1's compliment arithmetic:

In 1's comp subtraction, add the 1's comp of the subtrahend to the minuend. If there is a carryout, bring the carry around & add it to the LSB called the **end around carry**. Look at the sign bit (MSB). If this is a 0, the result is +ve & is in true binary. If the MSB is a 1 (carry or no carry), the result is -ve & is in its 1's comp form. Take its 1's comp to get the magnitude in binary.

Ex: Subtract 14 from 25 using 8 bit 1's EX: ADD -25 to +14

$$\begin{array}{rcl}
 25 & = & 00011001 \\
 -45 & = & 11110001 \\
 \hline
 +11 & & (1)00001010 \\
 & & \hline
 & +1 & \\
 & & 00001011 \\
 & & \hline
 \end{array}
 \qquad
 \begin{array}{rcl}
 +14 & = & 00001110 \\
 -25 & = & +11100110 \\
 \hline
 -11 & & 11110100 \\
 & & \hline
 \end{array}$$

No carry MSB =1  
result=-ve=-11<sub>10</sub>

MSB is a 0 so result is +ve (binary )

=+11<sub>10</sub>

### Binary codes

Binary codes are codes which are represented in binary system with modification from the original ones.

- ☐ Weighted Binary codes
- ☐ Non Weighted Codes

Weighted binary codes are those which obey the positional weighting principles, each position of the number represents a specific weight. The binary counting sequence is an example.

Decimal	BCD 8421	Excess-3	84-2-1	2421	5211	Bi-Quinary 5043210			5	0	4	3	2	1	0
0	0000	0011	0000	0000	0000	0100001		0		X					X
1	0001	0100	0111	0001	0001	0100010		1		X				X	
2	0010	0101	0110	0010	0011	0100100		2		X			X		
3	0011	0110	0101	0011	0101	0101000		3		X		X			
4	0100	0111	0100	0100	0111	0110000		4		X	X				
5	0101	1000	1011	1011	1000	1000001		5	X						X
6	0110	1001	1010	1100	1010	1000010		6	X					X	
7	0111	1010	1001	1101	1100	1000100		7	X				X		
8	1000	1011	1000	1110	1110	1001000		8	X			X			
9	1001	1111	1111	1111	1111	1010000		9	X		X				

### Reflective Code

A code is said to be reflective when code for 9 is complement for the code for 0, and

so is for 8 and 1 codes, 7 and 2, 6 and 3, 5 and 4. Codes 2421, 5211, and excess-3 are reflective, whereas the 8421 code is not.

### Sequential Codes

A code is said to be sequential when two subsequent codes, seen as numbers in binary representation, differ by one. This greatly aids mathematical manipulation of data. The 8421 and Excess-3 codes are sequential, whereas the 2421 and 5211 codes are not.

### Non weighted codes

Non weighted codes are codes that are not positionally weighted. That is, each position within the binary number is not assigned a fixed value. Ex: Excess-3 code

### Excess-3 Code

Excess-3 is a non weighted code used to express decimal numbers. The code derives its name from the fact that each binary code is the corresponding 8421 code plus 0011(3).

### Gray Code

The gray code belongs to a class of codes called minimum change codes, in which only one bit in the code changes when moving from one code to the next. The Gray code is non-weighted code, as the position of bit does not contain any weight. The gray code is a reflective digital code which has the special property that any two subsequent numbers codes differ by only one bit. This is also called a unit- distance code. In digital Gray code has got a special place.

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

## Binary to Gray Conversion

- ☐ Gray Code MSB is binary code MSB.
- ☐ Gray Code MSB-1 is the XOR of binary code MSB and MSB-1.
- ☐ MSB-2 bit of gray code is XOR of MSB-1 and MSB-2 bit of binary code.
- ☐ MSB-N bit of gray code is XOR of MSB-N-1 and MSB-N bit of binary code.

## 8421 BCD code ( Natural BCD code):

Each decimal digit 0 through 9 is coded by a 4 bit binary no. called natural binary codes. Because of the 8,4,2,1 weights attached to it. It is a weighted code & also sequential . it is useful for mathematical operations. The advantage of this code is its ease of conversion to & from decimal. It is less efficient than the pure binary, it requires more bits.

Ex: 14→1110 in binary

But as 0001 0100 in 8421 code.

The disadvantage of the BCD code is that , arithmetic operations are more complex than they are in pure binary . There are 6 illegal combinations 1010,1011,1100,1101,1110,1111 in these codes, they are not part of the 8421 BCD code system . The disadvantage of 8421 code is, the rules of binary addition 8421 no, but only to the individual 4 bit groups.

## BCD Addition:

It is individually adding the corresponding digits of the decimal no,s expressed in 4 bit binary groups starting from the LSD . If there is no carry & the sum term is not an illegal code , no correction is needed .If there is a carry out of one group to the next group or if the sum term is an illegal code then  $6_{10}(0100)$  is added to the sum term of that group & the resulting carry is added to the next group.

Ex: Perform decimal additions in 8421 code

(a)25+13

In BCD      25= 0010 0101

In BCD      +13 =+0001 0011

\_\_\_\_\_

38      0011 1000

No carry , no illegal code .This is the corrected sum

(b). 679.6 + 536.8

679.6 = 0110 0111 1001 .0110 in BCD  
 +536.8 = +0101 0011 0010 .1000 in BCD

-----  
 1216.4      1011      1010      0110      . 1110      illegal codes  
                  +0110      + 0011      +0110      . + 0110      add 0110 to each

(1)0001      (1)0000      (1)0101      . (1)0100      propagate carry  
 /                   /                   /                   /  
 +1                   +1                   +1                   +1  
 -----  
 0001              0010              0001              0110      .      0100  
  
 1                  2                  1                  6                  .      4

### BCD Subtraction:

Performed by subtracting the digits of each 4 bit group of the subtrahend the digits from the corresponding 4- bit group of the minuend in binary starting from the LSD . if there is no borrow from the next group , then  $6_{10}(0110)$  is subtracted from the difference term of this group.

(a)38-15

In BCD      38= 0011      1000  
 In BCD      -15 = -0001      0101

-----  
 23      0010      0011

No borrow, so correct difference.

(b) 206.7-147.8

206.7 = 0010 0000 0110 . 0111      in BCD  
 -147.8 = -0001 0100 0111 . 0110      in BCD

-----  
 58.9      0000      1011      1110      .      1111      borrows are present  
          -0110 -0110 .      -0110      subtract 0110  
 -----  
                  0101      1000      .      1001



### BCD Subtraction using 9's & 10's compliment methods:

Form the 9's & 10's compliment of the decimal subtrahend & encode that no. in the 8421 code . the resulting BCD no.s are then added.

EX: 305.5 – 168.8

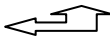
$$\begin{array}{r} 305.5 = 305.5 \\ -168.8 = +83.1 \quad \text{9's comp of -168.8} \\ \hline (1)136.6 \\ \quad +1 \quad \text{end around carry} \\ \quad \mathbf{136.7} \quad \text{corrected difference} \\ 305.5_{10} = 0011 \ 0000 \ 0101 \ . \ 0101 \\ +831.1_{10} = +1000 \ 0011 \ 0001 \ . \ 0001 \quad \text{9's comp of 168.8 in BCD} \\ \hline +1011 \ 0011 \ 0110 \ . \ 0110 \quad \text{1011 is illegal code} \\ +0110 \quad \text{add 0110} \\ \hline (1)0001 \ 0011 \ 0110 \ . \ 0110 \\ \quad +1 \quad \text{End around carry} \\ \hline 0001 \ 0011 \ 0110 \ . \ 0111 \\ = 136.7 \end{array}$$

### Excess three(xs-3)code:

It is a non-weighted BCD code .Each binary codeword is the corresponding 8421 codeword plus 0011(3).It is a sequential code & therefore , can be used for arithmetic operations..It is a self-complementing code.s o the subtraction by the method of compliment addition is more direct in xs-3 code than that in 8421 code. The xs-3 code has six invalid states 0000,0010,1101,1110,1111.. It has interesting properties when used in addition & subtraction.

### Excess-3 Addition:

Add the xs-3 no.s by adding the 4 bit groups in each column starting from the LSD. If there is no carry starting from the addition of any of the 4-bit groups , subtract 0011 from the sum term of those groups ( because when 2 decimal digits are added in xs-3 & there is no carry , result in xs-6). If there is a carry out, add 0011 to the sum term of those groups( because when there is a carry, the invalid states are skipped and the result is normal binary).

EX:	37	0110	1010	
	+28	+0101	1011	
	<hr/>			
	65	1011	(1)0101	carry generated
		+1		propagate carry
		<hr/>		
		1100	0101	add 0011 to correct 0101 &
		-0011	+0011	subtract 0011 to correct 1100
		<hr/>		
		1001	1000	=65 <sub>10</sub>

### Excess -3 (XS-3) Subtraction:

Subtract the xs-3 no.s by subtracting each 4 bit group of the subtrahend from the corresponding 4 bit group of the minuend starting from the LSD .if there is no borrow from the next 4-bit group add 0011 to the difference term of such groups (because when decimal digits are subtracted in xs-3 & there is no borrow , result is normal binary). If there is a borrow , subtract 0011 from the difference term (because taking a borrow is equivalent to adding six invalid states , result is in xs-6)

Ex: 267-175

267 =	0101	1001	1010	
-175 =	-0100	1010	1000	
	<hr/>			
	0000	1111	0010	
	+0011	-0011	+0011	
	<hr/>			
	0011	1100	+0011	=92 <sub>10</sub>

### Xs-3 subtraction using 9's & 10's compliment methods:

Subtraction is performed by the 9's compliment or 10's compliment

Ex: 687-348 The subtrahend (348) xs -3 code & its compliment are:

9's comp of 348 = 651

Xs-3 code of 348 = 0110 0111 1011

1's comp of 348 in xs-3 = 1001 1000 0100

Xs=3 code of 348 in xs=3 = 1001 1000 0100

$$\begin{array}{r} 687 \\ -348 \\ \hline \end{array} \rightarrow \begin{array}{r} 687 \\ +651 \text{ 9's compl of 348} \\ \hline \end{array}$$

$$\begin{array}{r} 339 \\ (1)338 \\ +1 \text{ end around carry} \\ \hline \end{array}$$

$$\begin{array}{r} - \\ 339 \end{array} \quad \text{corrected difference in decimal}$$

$$\begin{array}{r} 1001 \\ +1001 \\ \hline \end{array} \quad \begin{array}{r} 1011 \\ 1000 \\ \hline \end{array} \quad \begin{array}{r} 1010 \\ 0100 \\ \hline \end{array} \quad \begin{array}{l} 687 \text{ in xs-3} \\ 1's \text{ comp } 348 \text{ in xs-3} \end{array}$$

//

$$\begin{array}{r} +1 \\ \hline \end{array} \quad \begin{array}{r} +1 \\ \hline \end{array} \quad \text{propagate carry}$$

$$\begin{array}{r} (1)0011 \quad 0010 \quad 1110 \\ \hline \end{array}$$

+1 end around carry

$$\begin{array}{r} \hline \end{array}$$

$$\begin{array}{r} 0011 \\ +0011 \\ \hline \end{array} \quad \begin{array}{r} 0011 \\ +0011 \\ \hline \end{array} \quad \begin{array}{r} 1111 \\ +0011 \\ \hline \end{array} \quad \begin{array}{l} \text{(correct 1111 by sub0011 and} \\ \text{correct both groups of 0011 by} \\ \text{adding 0011)} \end{array}$$

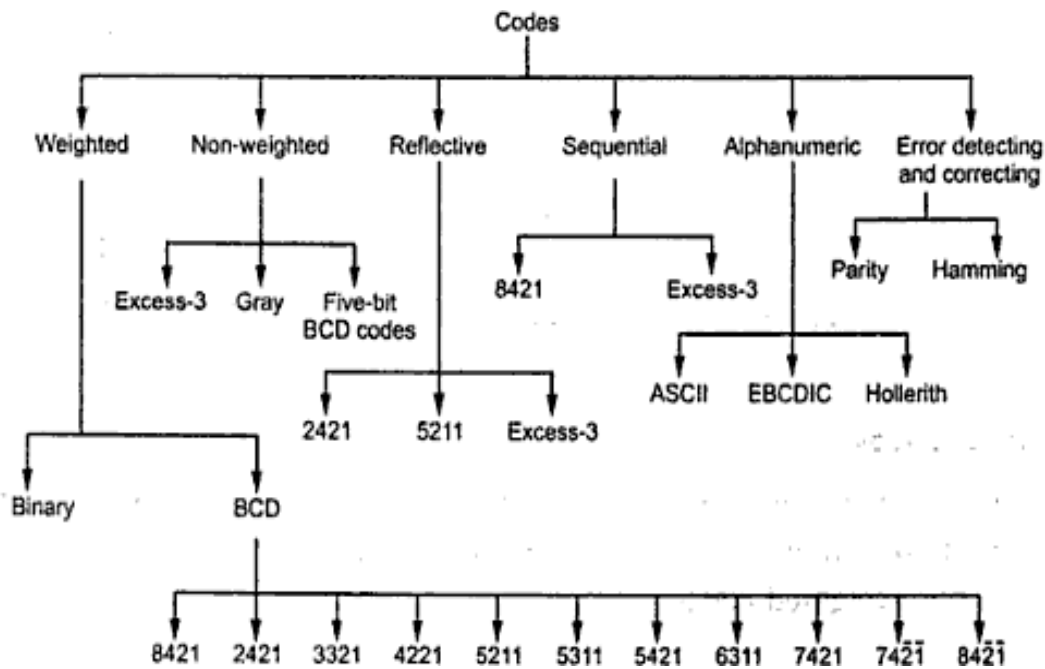
$$\begin{array}{r} 0110 \\ 0110 \\ 1100 \end{array} \quad \text{corrected diff in xs-3} = 330_{10}$$

### The Gray code (reflective –code):

Gray code is a non-weighted code & 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.Popular of the unit distance code.It is also a reflective code i.e,both reflective & unit distance. The n least significant bits for  $2^n$  through  $2^{n+1}-1$  are the mirror images of those for 0 through  $2^n-1$ .An N bit gray code can be obtained by reflecting an N-1 bit code about an axis at the end of the code, & putting the MSB of 0 above the axis & the MSB of 1 below the axis.

Reflection of gray codes:

Gray Code				Decimal	4 bit binary
1 bit	2 bit	3 bit	4 bit		
0	00	000	0000	0	0000
1	01	001	0001	1	0001
	11	011	0011	2	0010
	10	010	0010	3	0011
		110	0110	4	0100
		111	0111	5	0101
		101	0101	6	0110
		110	0100	7	0111
			1100	8	1000
			1101	9	1001
			1111	10	1010
			1110	11	1011
			1010	12	1100
			1011	13	1101
			1001	14	1110
			1000	15	1111



Binary codes block diagram

**Error – Detecting codes:** When binary data is transmitted & processed, it is susceptible to noise that can alter or distort its contents. The 1's may get changed to 0's & 1's. Because digital systems must be accurate to the digit, error can pose a problem. Several schemes have been devised to detect the occurrence of a single bit error in a binary word, so that whenever such an error occurs the concerned binary word can be corrected & retransmitted.

**Parity:** The simplest technique for detecting errors is that of adding an extra bit known as parity bit to each word being transmitted. Two types of parity: Odd parity, even parity. For odd parity, the parity bit is set to a 0 or a 1 at the transmitter such that the total no. of 1 bit in the word including the parity bit is an odd no. For even parity, the parity bit is set to a 0 or a 1 at the transmitter such that the parity bit is an even no.

Decimal	8421 code	Odd parity	Even parity
0	0000	1	0
1	0001	0	1
2	0010	0	1
3	0011	1	0
4	0100	0	1
5	0100	1	0
6	0110	1	0
7	0111	0	1
8	1000	0	1
9	1001	1	0

When the digit data is received . a parity checking circuit generates an error signal if the total no of 1's is even in an odd parity system or odd in an even parity system. This parity check can always detect a single bit error but cannot detect 2 or more errors with in the same word. Odd parity is used more often than even parity does not detect the situation. Where all 0's are created by a short ckt or some other fault condition.

Ex: Even parity scheme

(a) 10101010 (b) 11110110 (c) 10111001

Ans:

- (a) No. of 1's in the word is even is 4 so there is no error
- (b) No. of 1's in the word is even is 6 so there is no error
- (c) No. of 1's in the word is odd is 5 so there is error

Ex: odd parity

(a) 10110111 (b) 10011010 (c) 11101010

Ans:

- (a) No. of 1's in the word is even is 6 so word has error
- (b) No. of 1's in the word is even is 4 so word has error
- (c) No. of 1's in the word is odd is 5 so there is no error

### Checksums:

Simple parity can't detect two errors within the same word. To overcome this, use a sort of 2 dimensional parity. As each word is transmitted, it is added to the sum of the previously transmitted words, and the sum retained at the transmitter end. At the end of transmission, the sum called the check sum. Up to that time sent to the receiver. The receiver can check its sum with the transmitted sum. If the two sums are the same, then no errors were detected at the receiver end. If there is an error, the receiving location can ask for retransmission of the entire data, used in teleprocessing systems.

### Block parity:

Block of data shown is create the row & column parity bits for the data using odd parity. The parity bit 0 or 1 is added column wise & row wise such that the total no. of 1's in each column & row including the data bits & parity bit is odd as

Data	Parity bit	data
10110	0	10110
10001	1	10001
10101	0	10101
00010	0	00010
11000	1	11000
00000	1	00000
11010	0	11010

### Error –Correcting Codes:

A code is said to be an error –correcting code, if the code word can always be deduced from an erroneous word. For a code to be a single bit error correcting code, the minimum distance of that code must be three. The minimum distance of that code is the smallest no. of bits by which any two code words must differ. A code with minimum distance of 3 can't only correct single bit errors but also detect ( can't correct) two bit errors, The key to error correction is that it must be possible to detect & locate erroneous that it must be possible to detect & locate erroneous digits. If the location of an error has been determined. Then by complementing the erroneous digit, the message can be corrected , error correcting , code is the Hamming code , In this , to each group of m information or message or data bits, K parity checking bits denoted by P<sub>1</sub>,P<sub>2</sub>,-----p<sub>k</sub> located at positions  $2^{k-1}$  from left are added to form an (m+k) bit code word. To correct the error, k parity checks are performed on selected digits of each code word, & the position of the error bit is located by forming an error word, & the error bit is then complemented. The k bit error word is generated by putting a 0 or a 1 in the  $2^{k-1}$ th position depending upon whether the check for parity involving the parity bit P<sub>k</sub> is satisfied or not. Error positions & their corresponding values :

Error Position	For 15 bit code C <sub>4</sub> C <sub>3</sub> C <sub>2</sub> C <sub>1</sub>	For 12 bit code C <sub>4</sub> C <sub>3</sub> C <sub>2</sub> C <sub>1</sub>	For 7 bit code C <sub>3</sub> C <sub>2</sub> C <sub>1</sub>
0	0 0 0 0	0 0 0 0	0 0 0
1	0 0 0 1	0 0 0 1	0 0 1
2	0 0 1 0	0 0 1 0	0 1 0
3	0 0 1 1	0 0 1 1	0 1 1
4	0 1 0 0	0 1 0 0	1 0 0
5	0 1 0 1	0 1 0 1	1 0 1
6	0 1 1 0	0 1 1 0	1 1 0
7	0 1 1 1	0 1 1 1	1 1 1
8	1 0 0 0	1 0 0 0	
9	1 0 0 1	1 0 0 1	
10	1 0 1 0	1 0 1 0	
11	1 0 1 1	1 0 1 1	
12	1 1 0 0	1 1 0 0	
13	1 1 0 1		
14	1 1 1 0		
15	1 1 1 1		

### 7-bit Hamming code:

To transmit four data bits, 3 parity bits located at positions  $2^0$ ,  $2^1$  &  $2^2$  from left are added to make a 7 bit codeword which is then transmitted.

The word format

P <sub>1</sub>	P <sub>2</sub>	D <sub>3</sub>	P <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------

D—Data bits P-

Parity bits

Decimal Digit	For BCD P <sub>1</sub> P <sub>2</sub> D <sub>3</sub> P <sub>4</sub> D <sub>5</sub> D <sub>6</sub> D <sub>7</sub>	For Excess-3 P <sub>1</sub> P <sub>2</sub> D <sub>3</sub> P <sub>4</sub> D <sub>5</sub> D <sub>6</sub> D <sub>7</sub>
0	0 0 0 0 0 0 0	1 0 0 0 0 1 1
1	1 1 0 1 0 0 1	1 0 0 1 1 0 0
2	0 1 0 1 0 1 1	0 1 0 0 1 0 1
3	1 0 0 0 0 1 1	1 1 0 0 1 1 0
4	1 0 0 1 1 0 0	0 0 0 1 1 1 1
5	0 1 0 0 1 0 1	1 1 1 0 0 0 0
6	1 1 0 0 1 1 0	0 0 1 1 0 0 1
7	0 0 0 1 1 1 1	1 0 1 1 0 1 0
8	1 1 1 0 0 0 0	0 1 1 0 0 1 1
9	0 0 1 1 0 0 1	0 1 1 1 1 0 0



Ex: Encode the data bits 1101 into the 7 bit even parity Hamming Code

The bit pattern is

P<sub>1</sub>P<sub>2</sub>D<sub>3</sub>P<sub>4</sub>D<sub>5</sub>D<sub>6</sub>D<sub>7</sub>

1        1    0    1

Bits 1,3,5,7 (P<sub>1</sub> 111) must have even parity, so P<sub>1</sub>=1

Bits 2, 3, 6, 7(P<sub>2</sub> 101) must have even parity, so P<sub>2</sub>=0

Bits 4,5,6,7 (P<sub>4</sub> 101) must have even parity, so P<sub>4</sub>=0

The final code is 1010101

EX: Code word is 1001001

Bits 1,3,5,7 (C<sub>1</sub> 1001) →no error →put a 0 in the 1's position→C<sub>1</sub>=0

Bits 2, 3, 6, 7(C<sub>2</sub> 0001)) → error →put a 1 in the 2's position→C<sub>2</sub>=1

Bits 4,5,6,7 (C<sub>4</sub> 1001)) →no error →put a 0 in the 4's position→C<sub>3</sub>=0

**15-bit Hamming Code:** It transmit 11 data bits, 4 parity bits located  $2^0 2^1 2^2 2^3$

Word format is

P <sub>1</sub>	P <sub>2</sub>	D <sub>3</sub>	P <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	P <sub>8</sub>	D <sub>9</sub>	D <sub>10</sub>	D <sub>11</sub>	D <sub>12</sub>	D <sub>13</sub>	D <sub>14</sub>	D <sub>15</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------	-----------------	-----------------	-----------------	-----------------	-----------------

**12-Bit Hamming Code:**It transmit 8 data bits, 4 parity bits located at position  $2^0 2^1 2^2 2^3$

Word format is

P <sub>1</sub>	P <sub>2</sub>	D <sub>3</sub>	P <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	P <sub>8</sub>	D <sub>9</sub>	D <sub>10</sub>	D <sub>11</sub>	D <sub>12</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----------------	-----------------	-----------------

### Alphanumeric Codes:

These codes are used to encode the characteristics of alphabet in addition to the decimal digits. It is used for transmitting data between computers & its I/O device such as printers, keyboards & video display terminals. Popular modern alphanumeric codes are ASCII code & EBCDIC code.