Binary codes

- When numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded.
- The group of symbols is called as a code. The digital data is represented, stored and transmitted as group of binary bits.
- This group is also called as **binary code**.

Need for coding

Information sent over a noisy channel is likely to be distorted

Information is coded to facilitate

- Efficient transmission
- Error detection
- Error correction

- Coding is the process of altering the characteristics of information to make it more suitable for intended application
- Coding schemes depend on
  - Security requirements
  - Complexity of the medium of transmission
  - Levels of error tolerated
  - Need for standardization

Bits → Binary digIT

Bit	- a binary digit 0 or 1
Nibble	- a group of four bits

4bit → nibble

1001 1100 1010

TB
Tera byte

Bit - a binary digit 0 or 1

Nibble - a group of four bits 1001 1100 1010

Byte - a group of eight bits → 1001 1100 → Byte

Word - a group of sixteen bits;

(Sometimes used to designate 32 bit or 64 bit groups of bits)

2 bytes → word

1 B Tera byte

Binary coding

n = no. of bits

Assign each item of information a unique combination of 1s and 0s

- n is the number of bits in the code word
- x be the number of unique words

If n = 1, then x = 2 (0, 1) ✓ $2^1 = 2 = (0,1)$

n = 2, then x = 4 (00, 01, 10, 11) $2^2 = 4$ → 00, 01, 10, 11

n = 3, then x = 8 (000,001,010 ...111) → $2^3 = 8$ → 000, 001, 010, 011, 100, 101, 110

n = j, then x = 2^j

n = 4 bits ⟹ $2^4 = 16$

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

Binary coded decimal codes

Simple Scheme

- Convert decimal number inputs into binary form
- Manipulate these binary numbers
- Convert resultant binary numbers back into decimal numbers

Decimal → 7 = 111 ← binary ✓

However, it

7 = 0111 → 4 bits

However, it

- requires more hardware
- slows down the system

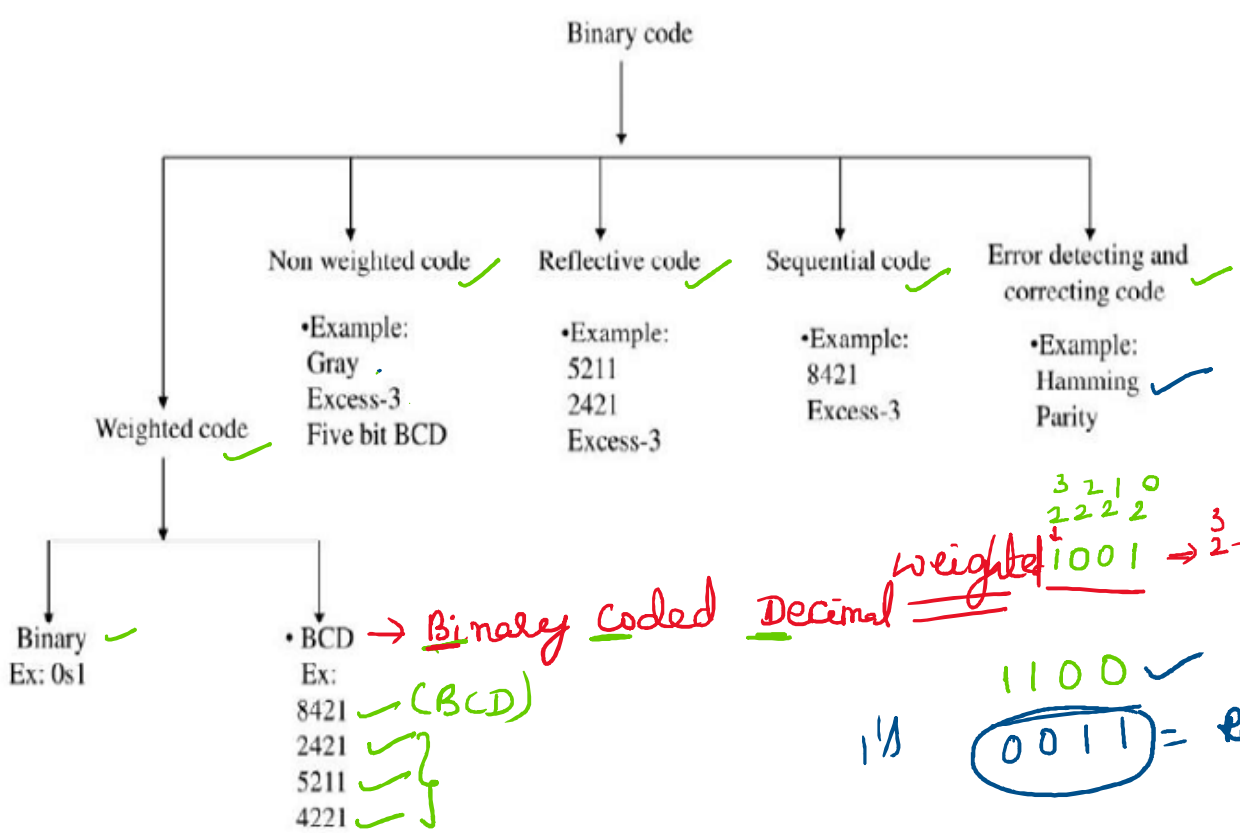
→ 7 = 111 binary

7 = 0111 → 4 bits → Binary Code

Advantages of Binary Code

- Binary codes are suitable for the computer applications.
- Binary codes are suitable for the digital communications.
- Binary codes make the analysis and designing of digital circuits if we use the binary codes.
- Since only 0 & 1 are being used, implementation becomes easy.

Classification of binary codes



weighted 1001 → $2^3+1 = 9$

1100

1's 0011 = Reflective

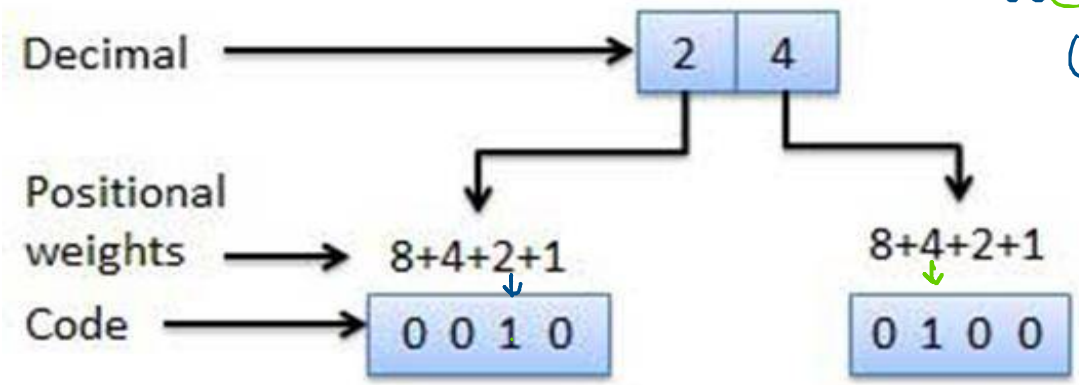
Weighted Codes

- Weighted binary codes are those binary codes which obey the positional weight principle.
- Each position of the number represents a specific weight. Several systems of the codes are used to express the decimal digits 0 through 9.
- Each decimal digit is represented by a group of four bits.

$(2)_{10}$ = Decimal

$00(10)_2$ = Binary

$(0010)_{BCD}$ = BCD



8 4 2 1

$(0010)_{BCD} = (2)_{10}$

Binary Coded Decimal (BCD) code

(8) 8421

- Each decimal digit is represented by a 4-bit binary number.
- BCD is a way to express each of the decimal digits with a binary code.
- In the BCD, with four bits we can represent sixteen numbers (0000 to 1111).
- But in BCD code only first ten of these are used (0000 to 1001).

The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

$n = 4 \Rightarrow 2^4 = 16$

0000 to 1001

(0 to 9)

$(46)_{10} = (\ )_{BCD}$

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

8421 Code (or) BCD Code

DECIMAL NUMBER	BINARY NUMBER	4 BIT EXPRESSION (8421)
0	0	0000
1	1	0001

Decimal Binary

$(1)_{10} = (01)_2$

$(0001)_{BCD}$

0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	0101
6	110	0110
7	111	0111
8	1000	1000
9	1001	1001

(0001) BCD

4bit binary

$(2)_{10} = (10)_2 \rightarrow$ Binary

8421

(0010) BCD

$(3)_{10} = (11)_2 \rightarrow$ Binary

8421

= (0011) BCD

$(4)_{10} = (100)_2 \rightarrow$ binary

8421

(0100) BCD $\Longrightarrow 4 \times 1 = (4)_{10}$

$(9)_{10} = (1001)_2 \rightarrow$ binary

$= (1001)_{BCD} \rightarrow$ BCD

$(57)_{10} = (\quad)_{BCD}$

0101 0111

$(98)_{10} =$ (8421 1001 1000) BCD

9 8421

$(73)_{10} = (\quad)_{BCD}$

(8421 0111 8421 0011) BCD

$\boxed{2421 \quad 1001}$ X

2421 Code

DECIMAL NUMBER	BINARY NUMBER	2421 CODE
0	0	0000

0000 = 0 is complement

NUMBER	NUMBER	CODE
0	0	0000
1	1	0001
2	10	0010
3	11	0011
4	100	0100
5	101	1011
6	110	1100
7	111	1101
8	1000	1110
9	1001	1111

0000 = 0 is complement 9

0001

0010

0011

0100

1011

1100

1101

1110

1111 = 9

Reflective

9 is complement 0

8 for 1

7 for 2

6 for 3

5 for 4

5211 Code

DECIMAL NUMBER	BINARY NUMBER	5211 CODE
0	0	0000
1	1	0001
2	10	0011
3	11	0101
4	100	0111
5	101	1000
6	110	1010
7	111	1100
8	1000	1110
9	1001	1111

0000

0001

0011

0101

0111

1000

1010

1100

1110

1111

Reflective code as well as weighted

Reflective codes

DECIMAL NUMBER	DECIMAL NUMBER	2421 CODE	5211 CODE
0	0	0000	0000
1	1	0001	0001
2	10	0010	0011
3	11	0011	0101
4	100	0100	0111
5	101	1011	1000

weighted codes

8421 → Not

2421 Reflect

5211

Reflective

4	100	0100	0111
5	101	1011	1000
6	110	1100	1010
7	111	1101	1100
8	1000	1110	1110
9	1001	1111	1111

Reflective Code

2421
5211

Advantages of BCD Codes

$(1234)_{10} = (0001\ 0010\ 0011\ 0100)_{BCD}$

- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

Non-Weighted Codes

- In this type of binary codes, the positional weights are not assigned.
- The examples of non-weighted codes are Excess-3 code and Gray code.

• Excess-3 code

- The Excess-3 code is also called as XS-3 code. Ex-3
- It is non-weighted code used to express decimal numbers.
- The Excess-3 code words are derived from the 8421 BCD code words adding (0011)2 or (3)10 to each code word in 8421.
- The excess-3 codes are obtained as follows –

$(6)_{10} = (\quad)$ Excess-3

Add

$(6)_{10} = (\quad)$ Excess-3

Decimal Number ⟶ 8421 BCD ⟶ Excess-3 (Add 0011)

$(0110)_{BCD} + (0011) \longrightarrow$ Excess-3 $\longrightarrow$ 1001 $= (9)$ XS-3

$$\begin{array}{r} 0110 \\ 0011 \\ \hline 1001 \end{array}$$

9 in excess -3

$(1001)_{BCD} + (0011) \Rightarrow 1100$

$$\begin{array}{r} 1001 \\ 0011 \\ \hline 1100 \end{array}$$

7 in excess -3

$(7)_{10} = (0111)_{BCD}$

$$\begin{array}{r} 0111 \\ +\ 0011 \\ \hline 1010 \end{array}$$

XS-3 for $(7)_{10}$

Example

Decimal	BCD 8 4 2 1	Excess-3 BCD + 0011
0	0 0 0 0	0 0 1 1 → 3
1	0 0 0 1	0 1 0 0 → 4
2	0 0 1 0	0 1 0 1 → 5
3	0 0 1 1	0 1 1 0 → 6
4	0 1 0 0	0 1 1 1 → 7
5	0 1 0 1	1 0 0 0 → 8
6	0 1 1 0	1 0 0 1 → 9
7	0 1 1 1	1 0 1 0 → 10
8	1 0 0 0	1 0 1 1 → 11

Sequential

$n = 4$

$2^4 = 16$

10 BCD

Sequential

Reflective

7	0 1 1 1	1 0 1 0 → 10
8	1 0 0 0	1 0 1 1 → 11
9	1 0 0 1	1 1 0 0 → 12

BCD and Ex-3 called as sequential codes

6011 0100
1100 1011

- # Gray Code

- It is the non-weighted code and it is not arithmetic codes. That means there are no specific weights assigned to the bit position.
- It has a very special feature that, only one bit will change each time the decimal number is incremented as shown in fig.
- As only one bit changes at a time, the gray code is called as a unit distance code. The gray code is a cyclic code.
- Gray code cannot be used for arithmetic operation.

Decimal	~~BCD~~ Binary	Gray
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

MSB
Binary Code → 1011

neglect
1+1 = 10

1+0 = 01

Binary Code → 1 0 1 1

Go to

Gray Code → 1 1 1 0

MSB

1+1 = 1 0

Carry Sum

1+0 = 0 1

Carry Sum

1+1 = 1 0

Carry Sum

Binary → 1 1 1 0

Gray Code → 1 0 0 1

MSB

Binary Code → 1 0 1 1 0

Gray code → 1 1 1 0 1

	Binary	Gray	
0	0 0 0 0	0 0 0 0	→ 0
1	0 0 0 1	0 0 0 1	→ 1
2	0 0 1 0	0 0 1 1	→ 3
3	0 0 1 1	0 0 1 0	→ 2
4	0 1 0 0	0 1 1 0	6
5	0 1 0 1	0 1 1 1	7
6	0 1 1 0	0 1 0 1	5
7	0 1 1 1	0 1 0 0	4
8	1 0 0 0	1 1 0 0	12

7	1000
8	1001
9	1010
10	1011
11	1100
12	1101
13	1110
14	1111
15	

1100	12
1101	13
1111	15
1110	14
1010	10
1011	11
1001	9
1000	8

unit distance code

Gray code

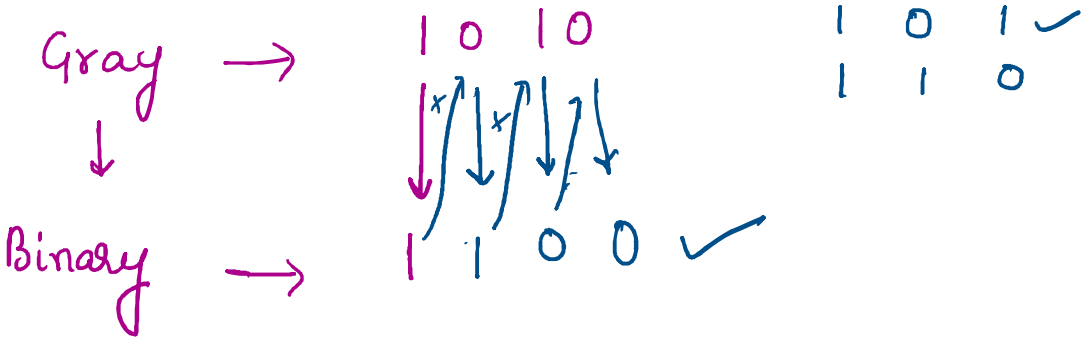
0000	→ 0	
0001	→ 1	mirror
0011	3	
0010	2	mirror
0110	6	
0111	7	
0101	5	
0100	4	mirror
1100	12	
1101	13	
1111	15	
1110	14	
1010	10	
1011	11	
1001	9	
1000	8	

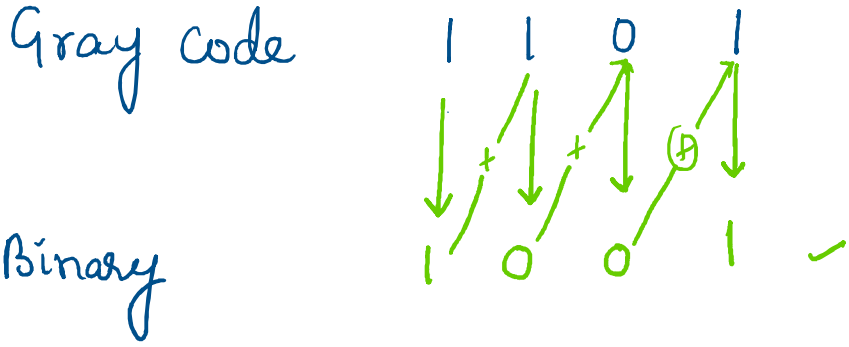
2^2 2^3

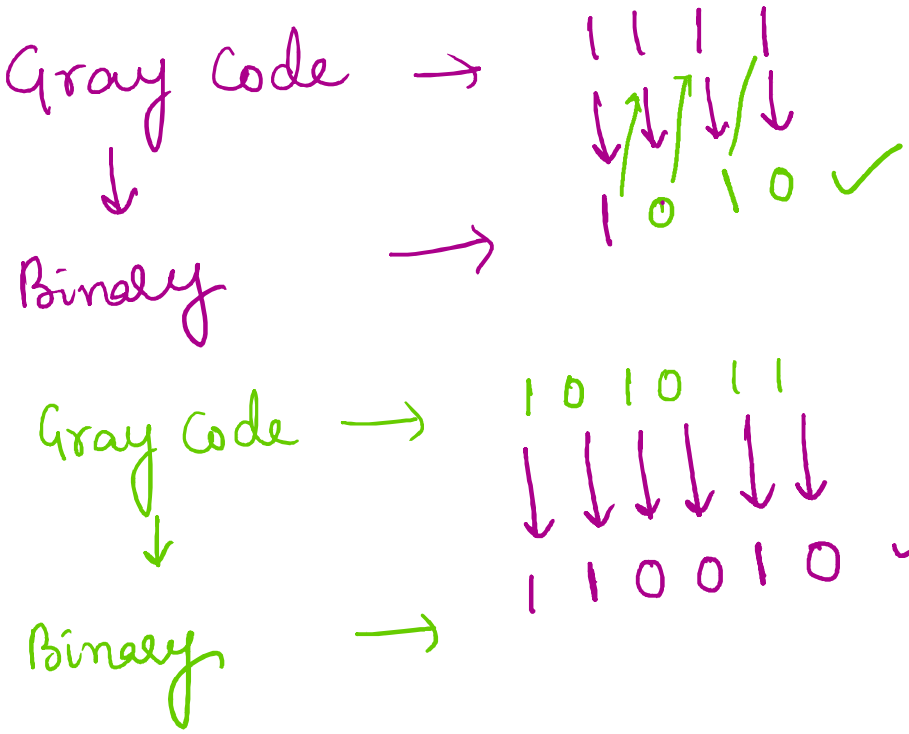
EX-OR

A	B	A⊕B
0	0	0
0	1	1 ✓
1	0	1 ✓
1	1	0

Gray → 1010







Application of Gray code

- Karnaugh map representation
- Gray code is popularly used in the shaft position encoders.
- A shaft position encoder produces a code word which represents the angular position of the shaft

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.

Alphanumeric codes

- A binary digit or bit can represent only two symbols as it has only two states '0' or '1'.
- But this is not enough for communication between two computers because there we need many more symbols for communication.
- These symbols are required to represent 26 alphabets with capital and small letters, numbers from 0 to 9, punctuation marks and other symbols.

  The alphanumeric codes are the codes that represent numbers and alphabetic characters.
- American Standard Code for Information Interchange (ASCII).
- Extended Binary Coded Decimal Interchange Code (EBCDIC).
- Five bit Baudot Code.

ASCII code is a 7-bit code whereas EBCDIC is an 8-bit code.

ASCII code is more commonly used worldwide while EBCDIC is used primarily in large IBM computers.

Error Codes

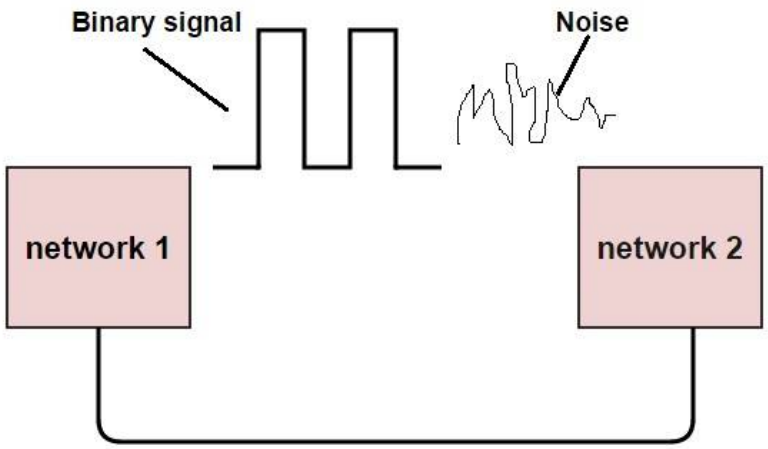
There are binary code techniques available to detect and correct data during data transmission.

- In digital systems, the analog signals will change into digital sequence (in the form of bits).
- This sequence of bits is called as “Data stream”. The change in position of single bit also leads to error in data output.

- Almost in all electronic devices, we find errors and we use error detection and correction techniques to get the exact or approximate output.

What is an Error

- The data can be corrupted during transmission (from source to receiver).
- It may be affected by external noise or some other physical imperfections. The input data is not same as the received output data.
- This mismatched data is called “Error”.
- The data errors will cause loss of important / secured data.
- Even one bit of change in data may affect the whole system’s performance.
- Generally the data transfer in digital systems will be in the form of ‘Bit – transfer’.
- The data error is likely to be changed in positions of 0 and 1 .



Types Of Errors

In a data sequence, if 1 is changed to zero or 0 is changed to 1, it is called “Bit error”.

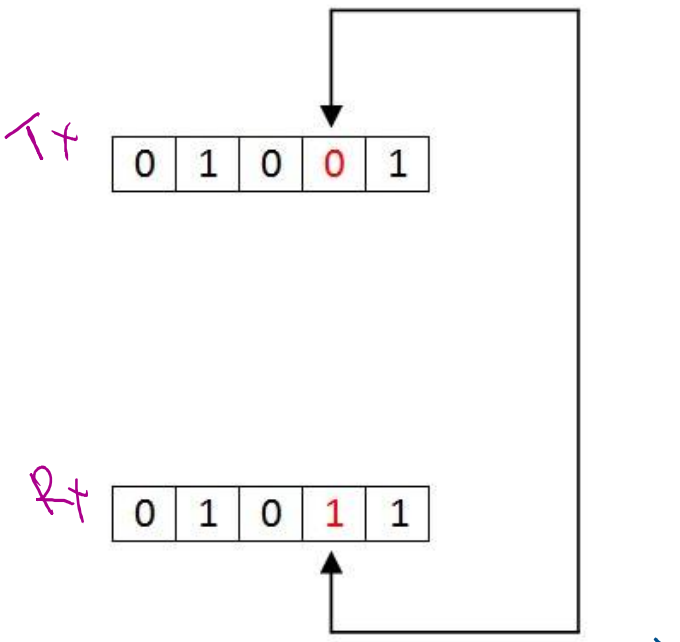
There are generally 3 types of errors occur in data transmission from transmitter to receiver. They are

- Single bit errors

- Multiple bit errors
- Burst errors

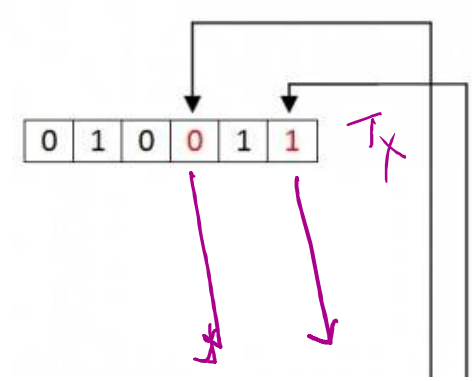
Single Bit Data Errors

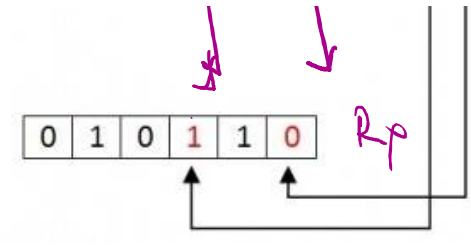
- *The change in one bit in the whole data sequence , is called "Single bit error".*
- *Occurrence of single bit error is very rare in serial communication system.*
- *This type of error occurs only in parallel communication system, as data is transferred bit wise in single line, there is chance that single line to be noisy.*



Multiple Bit Data Errors

- If there is change in two or more bits of data sequence of transmitter to receiver, it is called "Multiple bit error".
- This type of error occurs in both serial type and parallel type data communication networks.





Error Detecting Codes

- In digital communication system errors are transferred from one communication system to another, along with the data.
- If these errors are not detected and corrected, data will be lost .
- For effective communication, data should be transferred with high accuracy .This can be achieved by first detecting the errors and then correcting them.
- Error detection is the process of detecting the errors that are present in the data transmitted from transmitter to receiver, in a communication system.
- We use some redundancy codes to detect these errors, by adding to the data while it is transmitted from source (transmitter). These codes are called “Error detecting codes”.

Types of Error detection

1. Parity Checking
2. Cyclic Redundancy Check (CRC)
3. Longitudinal Redundancy Check (LRC)
4. Check Sum
5. Hamming code

Parity Checking

- Parity bit means nothing but an additional bit added to the data at the transmitter before transmitting the data.
- Before adding the parity bit, number of 1’s or zeros is calculated in the data.
- Based on this calculation of data an extra bit is added to the actual information / data.
- The addition of parity bit to the data will result in the change of data string size.
- This means if we have an 8 bit data, then after adding a parity bit to the data binary string it will become a 9 bit binary data string.
- Parity check is also called as “Vertical Redundancy Check (VRC)”.

- There is two types of parity bits in error detection, they are

  - Even parity
  - Odd parity

Even Parity

- If the data has even number of 1's, the parity bit is 0. Ex: data is 10000001 -> parity bit 0

- Odd number of 1's, the parity bit is 1. Ex: data is 10010001 -> parity bit 1

Odd Parity

- If the data has odd number of 1's, the parity bit is 0. Ex: data is 10011101 -> parity bit 0

- Even number of 1's, the parity bit is 1. Ex: data is 10010101 -> parity bit 1

NOTE: The counting of data bits will include the parity bit also.

- The circuit which adds a parity bit to the data at transmitter is called "Parity generator".

- The parity bits are transmitted and they are checked at the receiver.

- If the parity bits sent at the transmitter and the parity bits received at receiver are not equal then an error is detected.

- The circuit which checks the parity at receiver is called "Parity checker".

Messages with even parity and odd parity

3 bit data			Message with even parity		Message with odd parity	
A	B	C	Message	Parity	Message	Parity
0	0	0	000	0	000	1
0	0	1	001	1	001	0
0	1	0	010	1	010	0
0	1	1	011	0	011	1
1	0	0	100	1	100	0
1	0	1	101	0	101	1
1	1	0	110	0	110	1

1	0	1	101	0	101	1
1	1	0	110	0	110	1
1	1	1	111	1	111	0

Hamming Code

- This error detecting and correcting code technique is developed by R.W.Hamming .
- This code not only identifies the error bit, in the whole data sequence and it also corrects it.
- This code uses a number of parity bits located at certain positions in the codeword.
- The number of parity bits depends upon the number of information bits.
- The hamming code uses the relation between redundancy bits and the data bits and this code can be applied to any number of data bits.

What is a Redundancy Bit?

- Redundancy means "The difference between number of bits of the actual data sequence and the transmitted bits".
- These redundancy bits are used in communication system to detect and correct the errors, if any.

How the Hamming code actually corrects the errors?

- In Hamming code, the redundancy bits are placed at certain calculated positions in order to eliminate errors. The distance between the two redundancy bits is called "Hamming distance".
- To understand the working and the data error correction and detection mechanism of the hamming code, let's see to the following stages.

Number of parity bits

The number of parity bits to be added to a data string depends upon the number of information bits of the data string which is to be transmitted. Number of parity bits will be calculated by using the data bits. This relation is given below.

Here, n represents the number of bits in the data string.

P represents number of parity bits

Here, n represents the number of bits in the data string.

P represents number of parity bits.

$$2^P \geq n+P+1$$

If we have 4 bit data string, i.e. n = 4, then the number of parity bits to be added can be found by using trial and error method. Let's take P = 2, then

$$2^2 \geq 4+2+1$$

$$2^4 \geq 7 \Rightarrow \text{Not satisfied}$$

Let us $P = 3$

$$2^3 \geq 4+3+1$$

$$8 \geq 8 \quad \text{satisfied}$$

$n = 4$
$P = 3$
Total lengths $= 4+3 = 7$

So we can say that 3 parity bits are required to transfer the 4 bit data with single bit error correction.

Where to Place these Parity Bits?

After calculating the number of parity bits required, we should know the appropriate positions to place them in the information string, to provide single bit error correction.

In the above considered example, we have 4 data bits and 3 parity bits. So the total codeword to be transmitted is of 7 bits (4 + 3). We generally represent the data sequence from right to left, as shown below.

bit 7, bit 6, bit 5, bit 4, bit 3, bit 2, bit 1, bit 0

The parity bits have to be located at the positions of powers of 2. I.e. at 1, 2, 4, 8 and 16 etc. Therefore the codeword after including the parity bits will be like this

D7, D6, D5, P4, D3, P2, P1

Here P1, P2 and P3 are parity bits. D1 —- D7 are data bits.

Constructing a Bit Location Table

In Hamming code, each parity bit checks and helps in finding the errors in the whole code word. So we must find the value of the parity bits to assign them a bit value.

Bit Designation	D7	D6	D5	P4	D3	P2	P1
Bit Location	7	6	5	4	3	2	1
Binary Location Number	111	110	101	100	011	010	001
Data Bits (Dn)				---		---	---
Parity Bits (Pn)	---	---	---		---		

By calculating and inserting the parity bits in to the data bits, we can achieve error correction through Hamming code.

Let's understand this clearly, by looking into an example.

Ex: Encode the data 1101 in even parity, by using Hamming code.

Step 1

Calculate the required number of parity bits.

Let P = 2, then

$n = 4$ (Information bits)

2 parity bits are not sufficient for 4 bit data.

So let's try P = 3, then

$2^P \geq n+P+1$

$4 \geq 4+2+1 \Rightarrow 4 \geq 7$ (Not satisfied)

$2^P = 2^2 = 4$ and $n+P+1 \Rightarrow 4+2+1 = 7$

Therefore 3 parity bits are sufficient for 4 bit data.

$2^p \geq n+p+1$

$8 \geq 8$ (satisfied)

Step 2

$2^p = 2^3 = \underline{\underline{8}}$ and $n+p+1 \Rightarrow 4+3+1 = 8$

Constructing bit location table

Bit Designation	D7	D6	D5	P4	D3	P2	P1
Bit Location	7	6	5	4	3	2	1
Binary Location Number	111	110	101	100	011	010	001
Data Bits (D_n)	1	1	0		1		
Parity Bits (P_n)	↓	↓	↓	0 0		~~0~~ 1	0
Hamming code	1	1	0	0	1	1	0

$P_1 \Rightarrow 3, 5, 7 \Rightarrow 1, 0, 1 \xRightarrow{\text{even}} P_1 = 0$

$P_2 \Rightarrow 3, 6, 7 \Rightarrow 1, 1, 1 \xRightarrow{\text{odd}} P_2 = 1$

$P_4 \Rightarrow 5, 6, 7 \Rightarrow 0, 1, 1 \xRightarrow{\text{even}} P_4 = 0$

Step 3

Determine the parity bits.

For P1 : 3, 5 and 7 bits are having two1's so for even parity, P1 = 0.

For P2 : 3, 6 and 7 bits are having three 1's so for even parity, P2 = 1.

For P3 : 5, 6 and 7 bits are having two 1's so for even parity, P3 = 0.

By entering / inserting the parity bits at their respective positions, codeword can be formed and is transmitted. 1101110 Hamming code

NOTE: If the codeword has all zeros (ex: 0000000), then there is no error in Hamming code.

To represent the binary data in alphabets and numbers, we use alphanumeric codes.

① Determine the single error correcting code (Hamming code) for the information code 10111 for the odd parity bit

$D_9 D_7 D_6 D_5 D_3$

Data bits (information bits) = 10111

no. of information bits (n) = 5

no. of parity bits (P) = ?

Let P = 3
$2^3 \geq 5+3+1$
$8 \geq 9$ Not satisfied

$2^P \geq n+P+1$

Let (P = 4) $2^4 \geq 5+4+1$

$16 \geq 10$

Condition satisfied

$n+P \rightarrow 2^0\ 2^1\ 2^2\ 2^3\ 2^4$

Hamming code length = $5+4 = 9$

$2^4 = 8$ P_8 $2^2 = 4$ P_4 $2^1 = 2$ P_2 $2^0 = 1$ P_1

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1

Bit designation	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	9	8	7	6	5	4	3	2	1
Binary location	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bits	1		0	1	1		1		
Parity bits		0				1		1	0
Hamming code	1	0	0	1	1	1	1	1	0

$P_1 \rightarrow 3, 5, 7, 9 \Rightarrow 1, 1, 0, 1 \xRightarrow{\text{odd}} P_1 = 0$

$P_2 \Rightarrow 3, 6, 7 \Rightarrow 1, 1, 0 \xRightarrow{\text{even}} P_2 = 1$

$P_4 \Rightarrow 5, 6, 7 \Rightarrow 1, 1, 0 \xRightarrow{\text{even}} P_4 = 1$

$P_8 \Rightarrow 9 \Rightarrow 1 \xRightarrow{\text{odd}} P_8 = 0$

Hamming code $\Rightarrow$ 10 0 1 1 1 1 0 ✓