Sri SAI RAM ENGINEERING COLLEGE

INSTITUTE OF TECHNOLOGY

West Tambaram, Chennai - 44

YEAR	SEM
II	III

CS8351

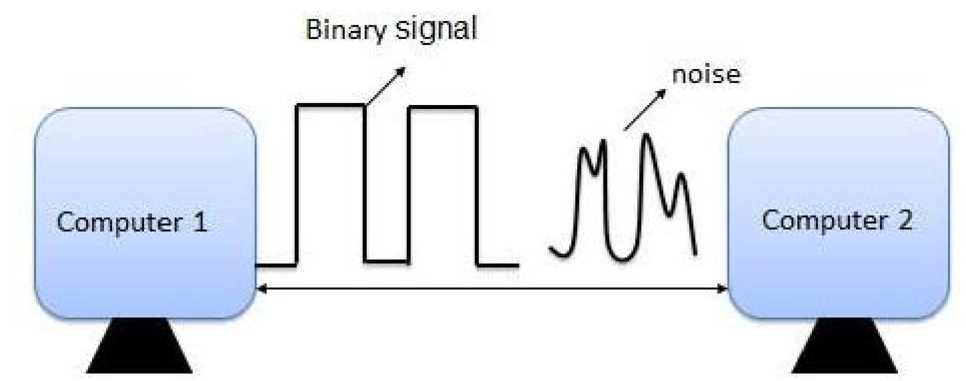
Digital Principles and System Design
(Common to CSE and IT)

UNIT V MEMORY AND PROGRAMMABLE LOGIC

5.3 ERROR DETECTION AND CORRECTION

What is Error?

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.



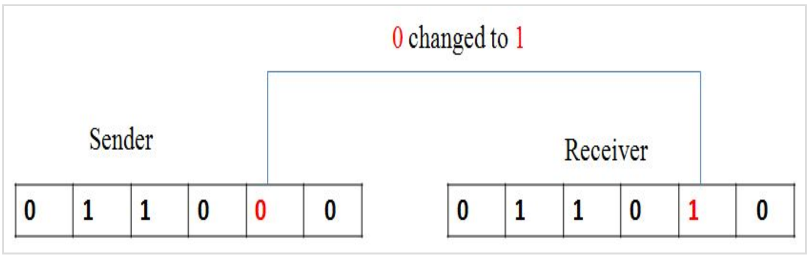
How to Detect and Correct Errors?

To detect and correct the errors, additional bits are added to the data bits at the time of transmission.

- The additional bits are called parity bits. They allow detection or correction of the errors.
- The data bits along with the parity bits form a code word.

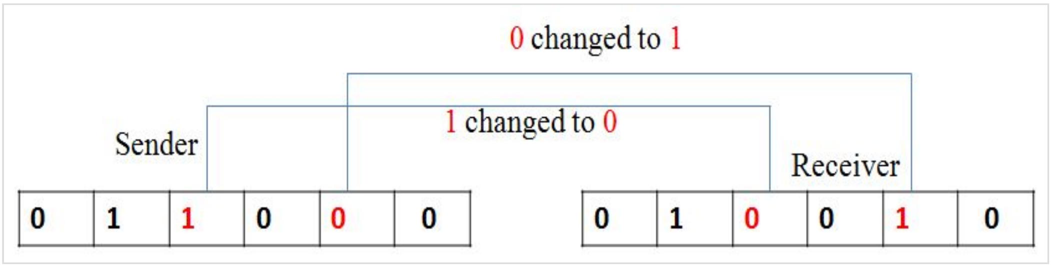
Types of errors

Single Bit error: If a bit is differed from sending and receiving information that is called single bit error. In the below figure the 0 bit is changed to 1 in the receiver side due to this noise, this results in single bit error



Single bit errors are very rare in the communication system because the noise must not a short duration that it can only disturb only one bit. So these errors in the communication system are less however this type of errors can happen in parallel communication where the noise will distribute in all the communication lines.

Multiple bit error: If two or more bits are differed between sending and receiving data stream, then the error is called multiple bit error. The position of the bits may differ i.e. the bits may not be consecutive.



Here two bits are changed from 0 to 1 and 1 to 0 which leads to two bit difference between the sending and receiving information which is called multiple Bit Error. This type are errors are also common in parallel and serial communication and it leads to some difficulty in detecting and correcting.

Error-Detecting codes

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is parity check.

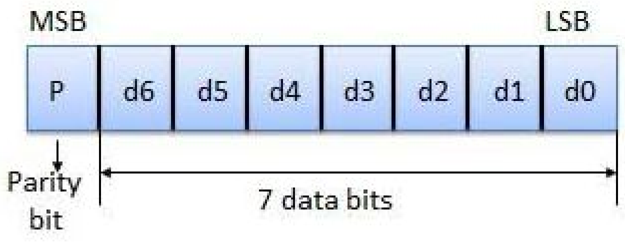
Error-Correcting codes

Along with error-detecting code, we can also pass some data to figure out the original message from the corrupt message that we received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit.

In error-correcting codes, parity check has a simple way to detect errors along with a sophisticated mechanism to determine the corrupt bit location. Once the corrupt bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message.

Parity Checking of Error Detection

It is the simplest technique for detecting and correcting errors. The MSB of an 8-bits word is used as the parity bit and the remaining 7 bits are used as data or message bits. The parity of 8-bits transmitted word can be either even parity or odd parity.



Even parity -- Even parity means the number of 1's in the given word including the parity bit should be even (2,4,6,....).

Odd parity -- Odd parity means the number of 1's in the given word including the parity bit should be odd (1,3,5,....).

Even Parity Code

The value of even parity bit should be zero, if even number of ones present in the binary code. Otherwise, it should be one. So that, even number of ones present in even parity code. Even parity code contains the data bits and even parity bit.

The following table shows the even parity codes corresponding to each 3-bit binary code. Here, the even parity bit is included to the right of LSB of binary code.

Binary Code	Even Parity bit	Even Parity Code
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

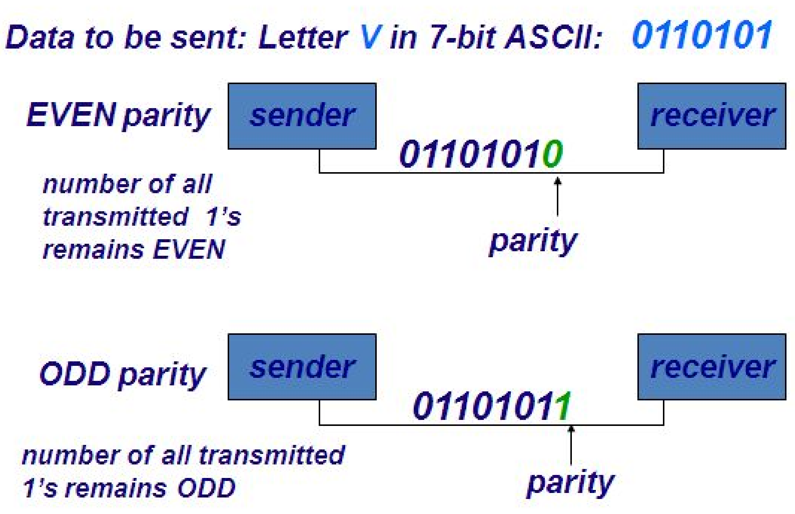
Odd Parity Code

The value of odd parity bit should be zero, if odd number of ones present in the binary code. Otherwise, it should be one. So that, odd number of ones present in odd parity code. Odd parity code contains the data bits and odd parity bit.

The following table shows the odd parity codes corresponding to each 3-bit binary code. Here, the odd parity bit is included to the right of LSB of binary code.

Binary Code	Odd Parity bit	Odd Parity Code
000	1	0001
001	0	0010
010	0	0100
011	1	0111
100	0	1000
101	1	1011
110	1	1101
111	0	1110

Example of Using Parity Bits


Data to be sent: Letter V in 7-bit ASCII: 0110101
EVEN parity
sender
receiver
01101010
number of all transmitted 1's remains EVEN
parity
ODD parity
sender
receiver
01101011
number of all transmitted 1's remains ODD
parity

<u>Simple Parity Check-An Example</u>

Data:

w o r l d

1110111 1101111 1110010 1101100 1100100

Sent As:

11101110 11011110 11100100 11011000 11001001

Corrupted:

11111110 11011110 11101100 11011000 11001001

<u>Simple Parity Check – Performance</u>

- Can detect all single-bit errors, Detects about 50% of errors
- Can also detect burst errors if the total number of bits changed is odd (1,3,5,..)
- Cannot detect errors where the total number of bits changed is even

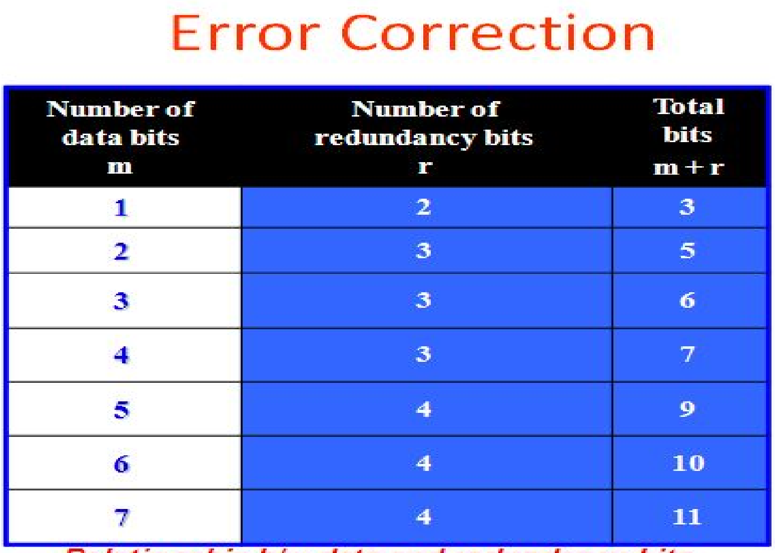
ERROR CORRECTION

HAMMING CODE

In telecommunication, Hamming codes are a family of linear error correcting. Hamming codes can detect up to two bit or one bit errors without detection of uncorrected error.

- Single-bit errors:
  - Can be detected by the addition of parity bit which helps to find "error" or "no error" which is sufficient to detect errors

Formula for finding number of redundancy bit is **$2^r >= m + r + 1$**

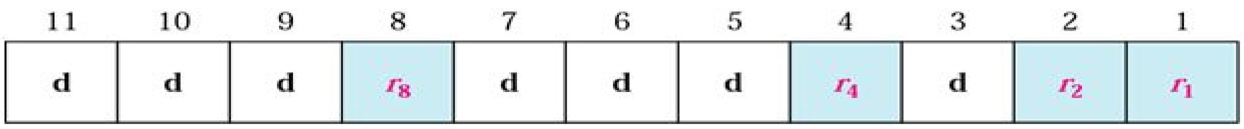


Error Correction

Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

Relationship b/w data and redundancy bits

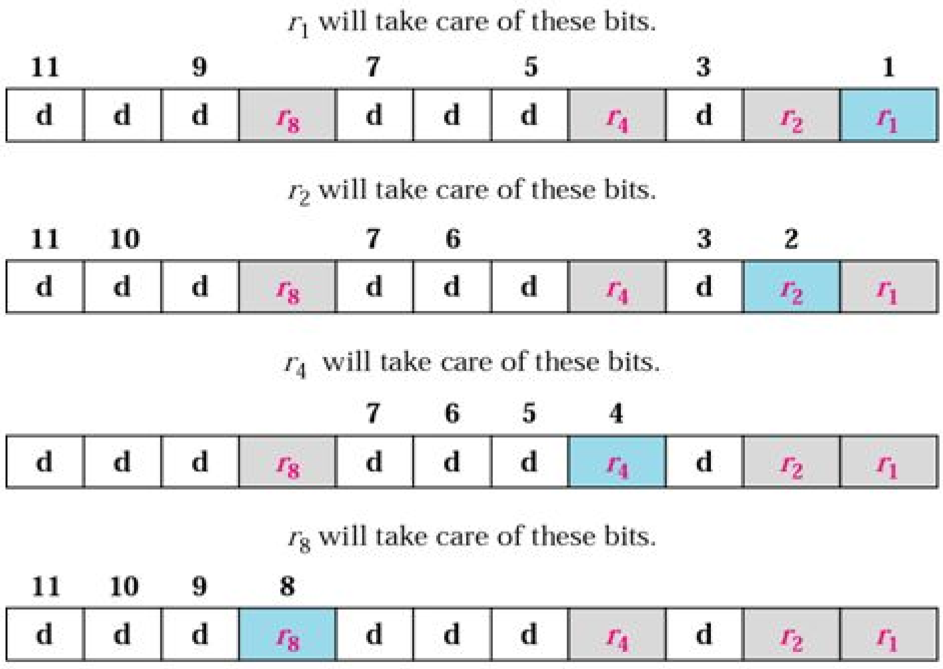
- Hamming Code can be applied to data units of any length and uses the relationship between data and redundancy bits
- For example: a 7-bit ASCII code requires 4 redundancy bits that can be added to the end of the data unit or mixed with the original data bits, which are placed in positions 1, 2, 4 and 8 i.e x^0,x^1,x^2,x^3 and so on.



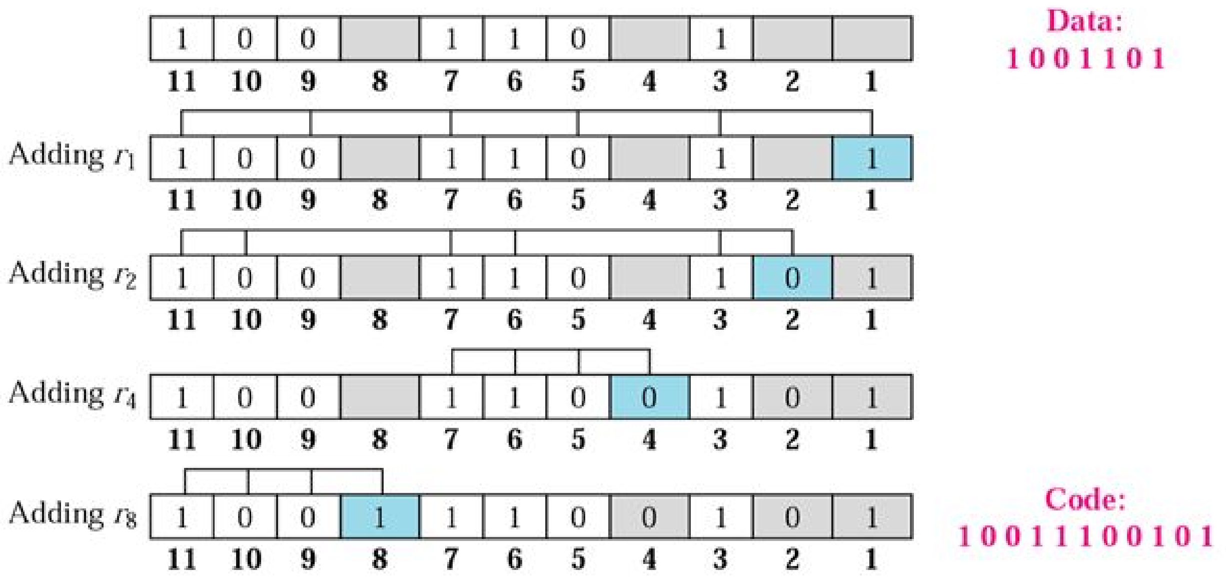
Positions of Redundancy bits in Hamming Code

- *In the Hamming Code, each "r" bit for one combination of data bits as below:*

r1:	*bits 1, 3, 5, 7, 9, 11*
r2:	*bits 2, 3, 6, 7, 10, 11*
r3:	*bits 4, 5, 6, 7*
r4:	*bits 8, 9, 10, 11*

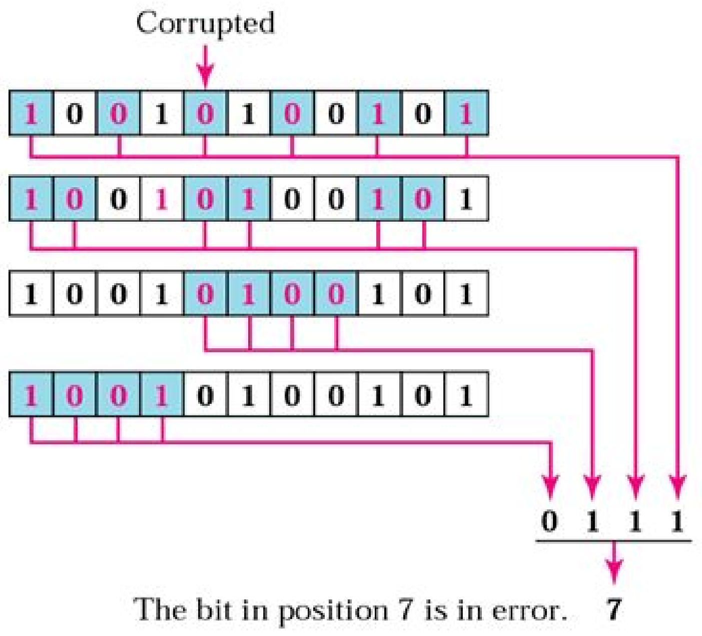

r_1 will take care of these bits.
11 9 7 5 3 1
d d d r_8 d d d r_4 d r_2 r_1
r_2 will take care of these bits.
11 10 7 6 3 2
d d d r_8 d d d r_4 d r_2 r_1
r_4 will take care of these bits.
7 6 5 4
d d d r_8 d d d r_4 d r_2 r_1
r_8 will take care of these bits.
11 10 9 8
d d d r_8 d d d r_4 d r_2 r_1

Redundancy bit Calculation



Example of Redundancy bit Calculation

Error Detection-Using Hamming Code



The bit in position 7 is in error.

Example 1

Let us find the Hamming code for binary code, d4 d3 d2 d1 = 1000. Consider even parity bits.

The number of bits in the given binary code is n=4.

We can find the required number of parity bits by using the following mathematical relation.

2k≥n+k+1

Substitute, n=4 in the above mathematical relation.

⇒2k≥4+k+1

⇒2k≥5+k

The minimum value of k that satisfied the above relation is 3. Hence, we require 3 parity bits p1, p2, and p3. Therefore, the number of bits in Hamming code will be 7, since there are 4 bits in binary code and 3 parity bits. We have to place the parity bits and bits of binary code in the Hamming code as shown below.

The 7-bit Hamming code is

b7 b6 b5 b4 b3 b2 b1 = d4 d3 d2 p3 d1 p2 p1

By substituting the bits of binary code, the Hamming code will be

b7 b6 b5 b4 b3 b2 b1=1 0 0 p3 0 p2 p1. Now, let us find the parity bits.

p1=b7⊕b5⊕b3=1⊕0⊕0=1

p2=b7⊕b6⊕b3=1⊕0⊕0=1

p3=b7⊕b6⊕b5=1⊕0⊕0=1

By substituting these parity bits, the Hamming code will be

b7 b6 b5 b4 b3 b2 b1=1001011.

Example 2

In the above example, we got the Hamming code as

b7 b6 b5 b4 b3 b2 b1 = 1001011.

Now, let us find the error position when the code received is

b7 b6 b5 b4 b3 b2 b1=1001111.

Now, let us find the check bits.

c1=b7⊕b5⊕b3⊕b1=1⊕0⊕1⊕1=1

c2=b7⊕b6⊕b3⊕b2=1⊕0⊕1⊕1=1

c3=b7⊕b6⊕b5⊕b4=1⊕0⊕0⊕1=0

The decimal value of check bits gives the position of error in received Hamming code.

c3c2c1=(011)2=(3)10

Therefore, the error present in third bit (b3) of Hamming code.