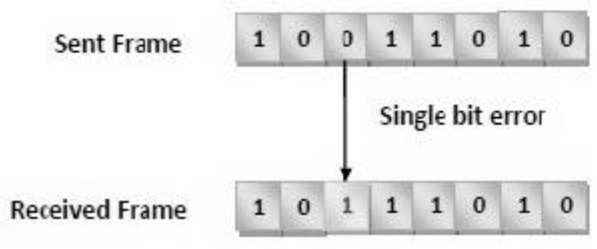
## Error:

When data is transmitted from one device to another device, the system does not guarantee whether the data received by the device is identical to the data transmitted by another device. An Error is a situation when the message received at the receiver end is not identical to the message transmitted.

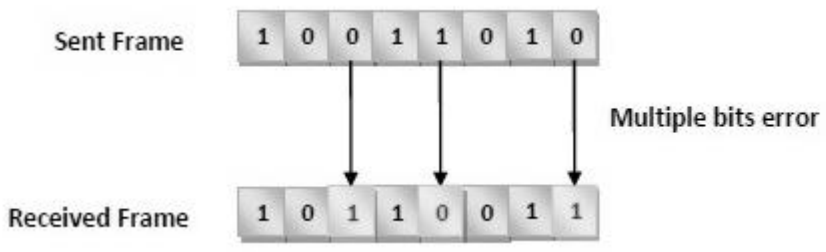
### Types of Errors

Errors can be of three types, namely single bit errors, multiple bit errors, and burst errors.

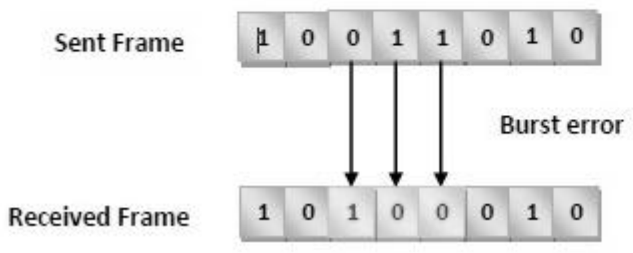
**Single bit error** − In the received frame, only one bit has been corrupted, i.e. either changed from 0 to 1 or from 1 to 0.



**Multiple bits error** − In the received frame, more than one bits are corrupted.



**Burst error** − In the received frame, more than one consecutive bits are corrupted.



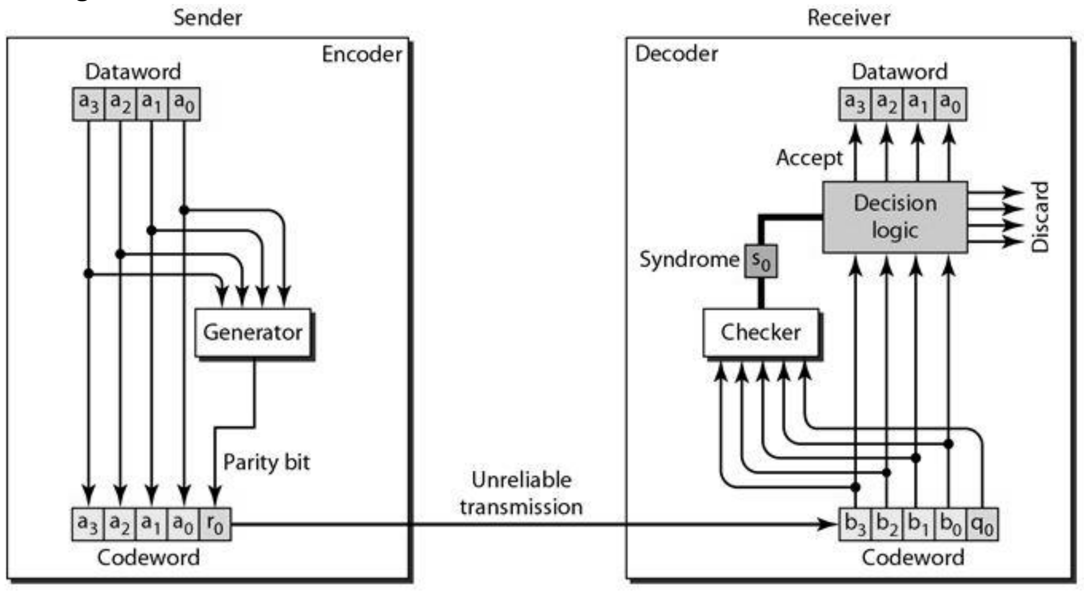
### Error Detecting Techniques:

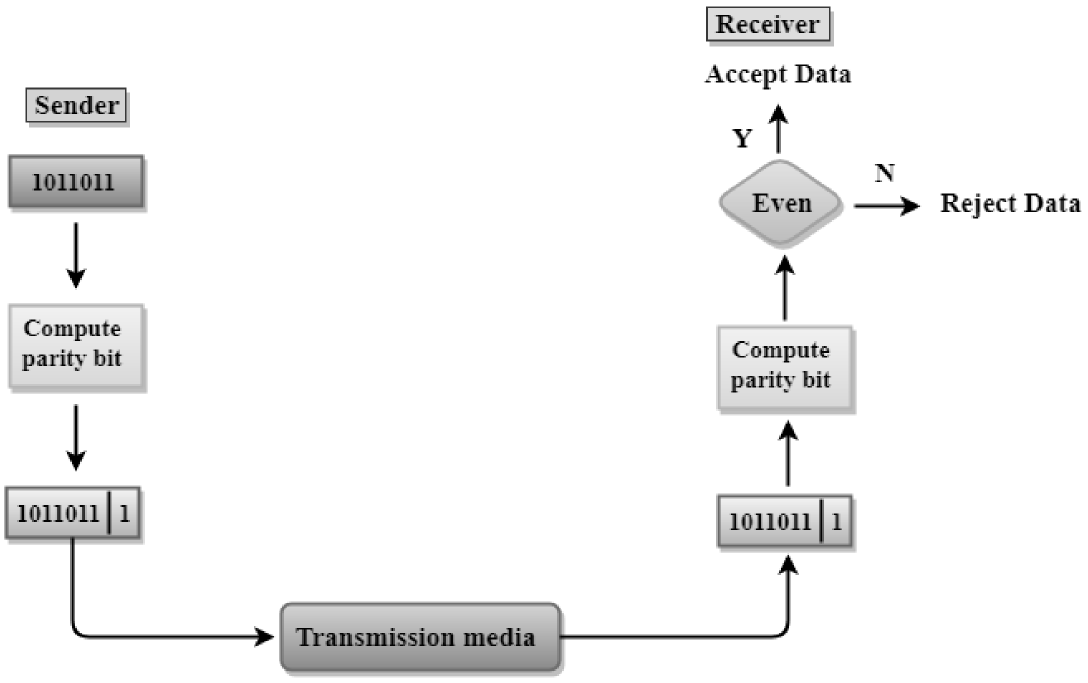
The most popular Error Detecting Techniques are:

- Single parity check
- Two-dimensional parity check
- Checksum
- Cyclic redundancy check

## Single Parity Check:

- Single Parity checking is the simple mechanism and inexpensive to detect the errors.
- In this technique, a redundant bit is also known as a parity bit which is appended at the end of the data unit so that the number of 1s becomes even.
- If the number of 1s bits is odd, then parity bit 1 is appended and if the number of 1s bits is even, then parity bit 0 is appended at the end of the data unit.
- At the receiving end, the parity bit is calculated from the received data bits and compared with the received parity bit.
- This technique generates the total number of 1s even, so it is known as even-parity checking.



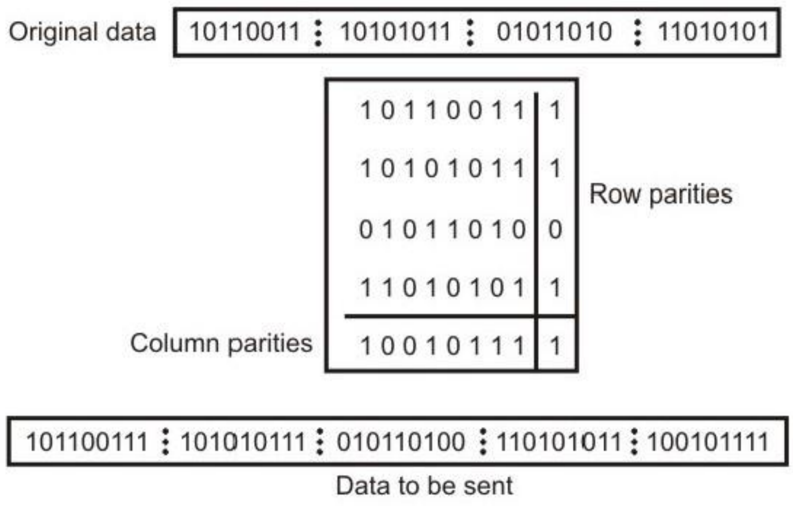


## Drawbacks of Single Parity Checking

- It can only detect single-bit errors which are very rare.
- If two bits are interchanged, then it cannot detect the errors.

## Two-Dimensional Parity Check

- Performance can be improved by using **Two-Dimensional Parity Check** which organizes the data in the form of a table.
- Parity check bits are computed for each row, which is equivalent to the single-parity check.
- In Two-Dimensional Parity check, a block of bits is divided into rows, and the redundant row of bits is added to the whole block.
- At the receiving end, the parity bits are compared with the parity bits computed from the received data.



## Drawbacks of 2D Parity Check

- If two bits in one data unit are corrupted and two bits exactly the same position in another data unit are also corrupted, then 2D Parity checker will not be able to detect the error.
- This technique cannot be used to detect the 4-bit errors or more in some cases.

## Checksum

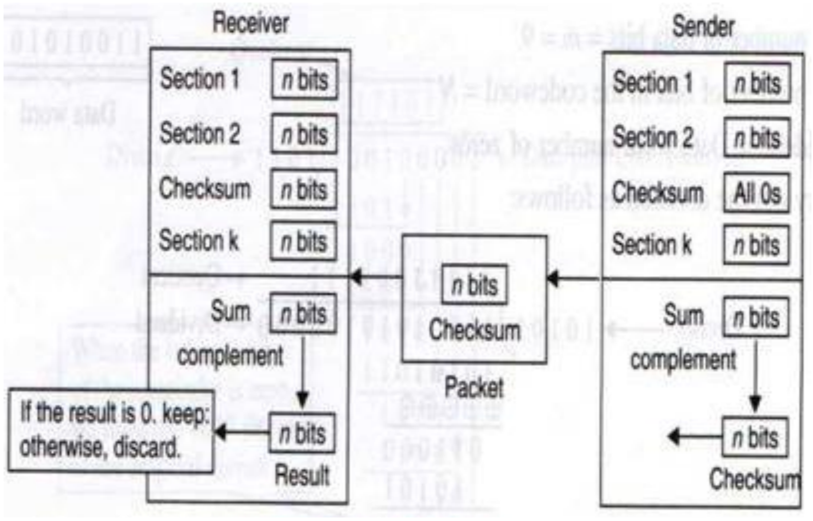
A Checksum is an error detection technique based on the concept of redundancy. It is divided into two parts:

### Checksum Generator

A Checksum is generated at the sending side. Checksum generator subdivides the data into equal segments of n bits each, and all these segments are added together. The sum is complemented and appended to the original data, known as checksum field. The extended data is transmitted across the network.

## Checksum Checker

A Checksum is verified at the receiving side. The receiver subdivides the incoming data into equal segments of n bits each, and all these segments are added together, and then this sum is complemented. If the complement of the sum is zero, then the data is accepted otherwise data is rejected.



**Example:**

Original Data

|  |  |  |  |
|---|---|---|---|
| 10011001 | 11100010 | 00100100 | 10000100 |
| 1 | 2 | 3 | 4 |

k=4, m=8

```
Sender

1          10011001
2          11100010
          ----------
          101111011
                  1
          ----------
           01111100
3          00100100
          ----------
           10100000
4          10000100
          ----------
          100100100
                  1
          ----------
Sum:       00100101
CheckSum:  11011010
```

```
Reciever

1             10011001
2             11100010
             ----------
             101111011
                     1
             ----------
              01111100
3             00100100
             ----------
              10100000
4             10000100
             ----------
             100100100
                     1
             ----------
              00100101
              11011010
             ----------
Sum:          11111111
Complement:   00000000
Conclusion: Accept Data
```

## Cyclic Redundancy Check (CRC)

CRC is a redundancy error technique used to determine the error.

**Following are the steps used in CRC for error detection:**

- In CRC technique, a string of n 0s is appended to the data unit, and this n number is less than the number of bits in a predetermined number, known as division which is n+1 bits.
- Secondly, the newly extended data is divided by a divisor using a process is known as binary division. The remainder generated from this division is known as CRC remainder.
- Thirdly, the CRC remainder replaces the appended 0s at the end of the original data. This newly generated unit is sent to the receiver.
- The receiver receives the data followed by the CRC remainder. The receiver will treat this whole unit as a single unit, and it is divided by the same divisor that was used to find the CRC remainder.

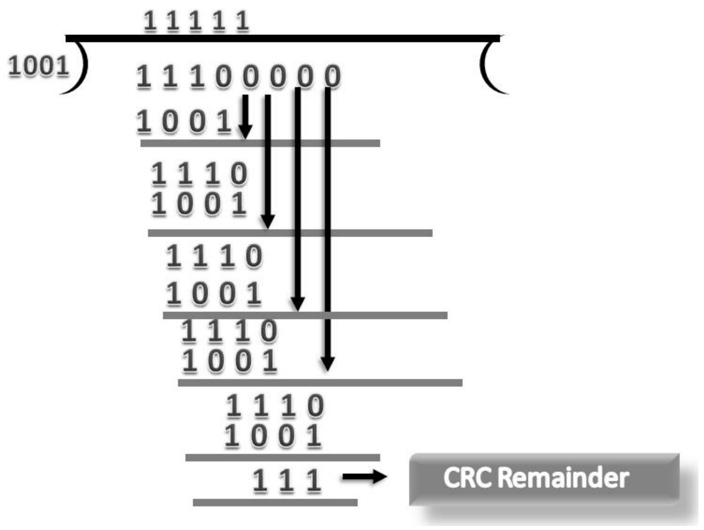
If the resultant of this division is zero which means that it has no error, and the data is accepted.
If the resultant of this division is not zero which means that the data consists of an error. Therefore, the data is discarded.

**Example:**

Suppose the original data is 11100 and divisor is 1001.

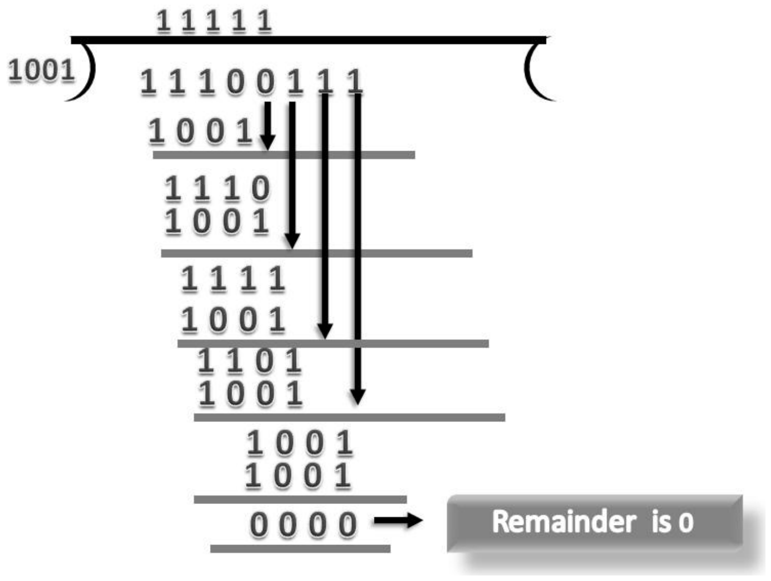
**CRC Generator**

- A CRC generator uses a modulo-2 division. Firstly, three zeroes are appended at the end of the data as the length of the divisor is 4 and we know that the length of the string 0s to be appended is always one less than the length of the divisor.
- Now, the string becomes 11100000, and the resultant string is divided by the divisor 1001.
- The remainder generated from the binary division is known as CRC remainder. The generated value of the CRC remainder is 111.
- CRC remainder replaces the appended string of 0s at the end of the data unit, and the final string would be 11100111 which is sent across the network.



**CRC Checker**

- The functionality of the CRC checker is similar to the CRC generator.
- When the string 11100111 is received at the receiving end, then CRC checker performs the modulo-2 division.
- A string is divided by the same divisor, i.e., 1001.
- In this case, CRC checker generates the remainder of zero. Therefore, the data is accepted.



## Hamming Code

Hamming code is a set of error-correction codes that can be used to **detect and correct the errors** that can occur when the data is moved or stored from the sender to the receiver. In this coding method, the source encodes the message by inserting redundant bits within the message. These redundant bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction. When the destination receives this message, it performs recalculations to detect errors and find the bit position that has error.

**Redundant bits –**

Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer.
The number of redundant bits can be calculated using the following formula:

$$2^r \geq m + r + 1 \qquad \text{where, r = redundant bit, m = data bit}$$

Suppose the number of data bits is 7, then the number of redundant bits can be calculated using:

$$2^4 \geq 7 + 4 + 1$$

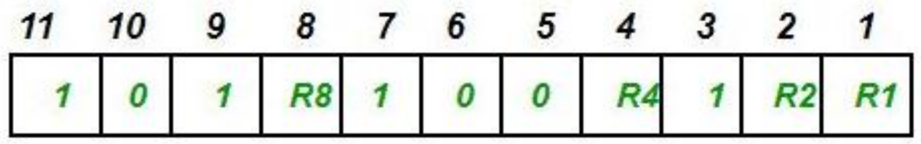
Thus, the number of redundant bits= 4

**Parity bits –**
A parity bit is a bit appended to a data of binary bits to ensure that the total number of 1’s in the data is even or odd. Parity bits are used for error detection. There are two types of parity bits:

1. **Even parity bit:**
   In the case of even parity, for a given set of bits, the number of 1’s are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1’s an even number. If the total number of 1’s in a given set of bits is already even, the parity bit’s value is 0.

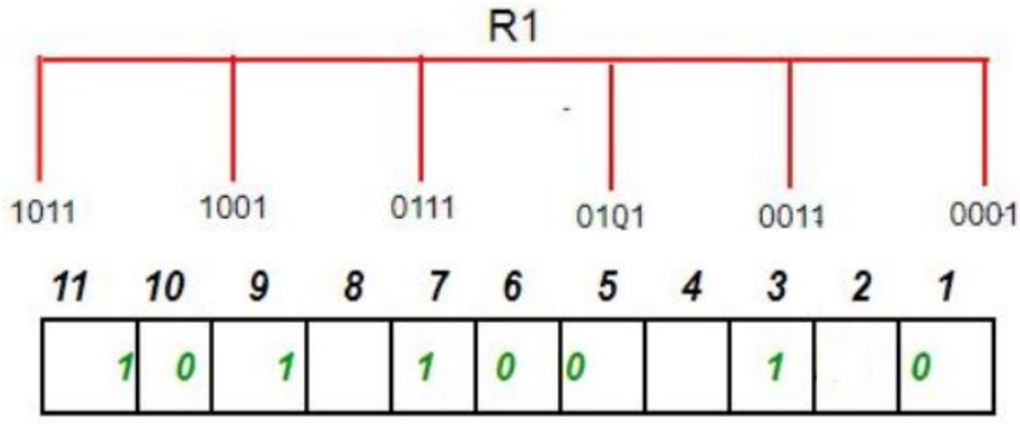
2. **Odd Parity bit –**
   In the case of odd parity, for a given set of bits, the number of 1’s are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1’s an odd number. If the total number of 1’s in a given set of bits is already odd, the parity bit’s value is 0.

**Example:**
Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



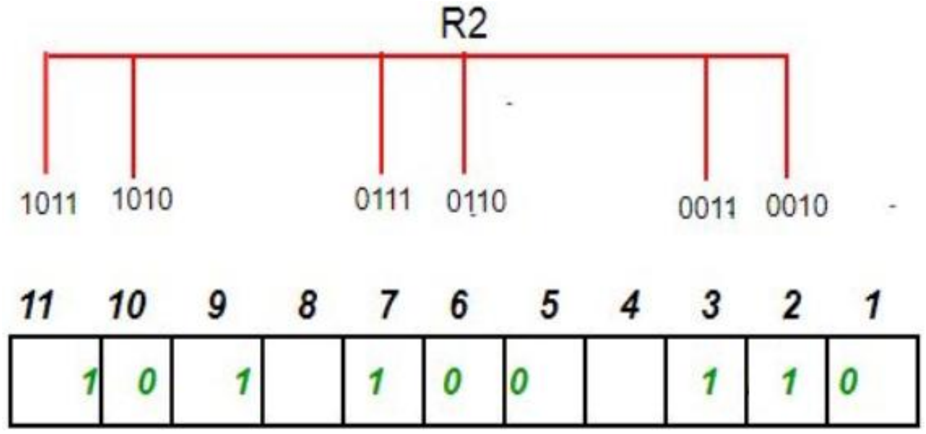
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | R8 | 1 | 0 | 0 | R4 | 1 | R2 | R1 |

**Determining the Parity bits:**
R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position. R1: bits 1, 3, 5, 7, 9, 11 . To find the redundant bit R1, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R1 is an even number the value of R1 (parity bit’s value) = 0

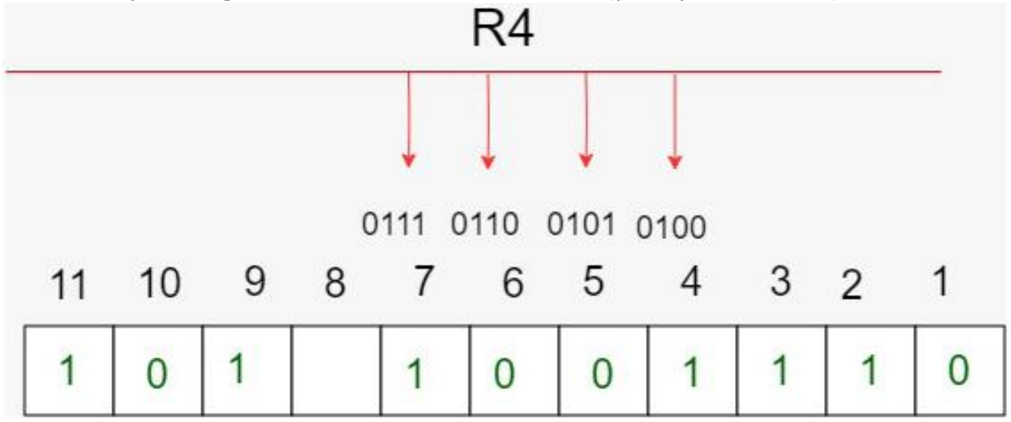


R2 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the second position from the least significant bit. R2: bits

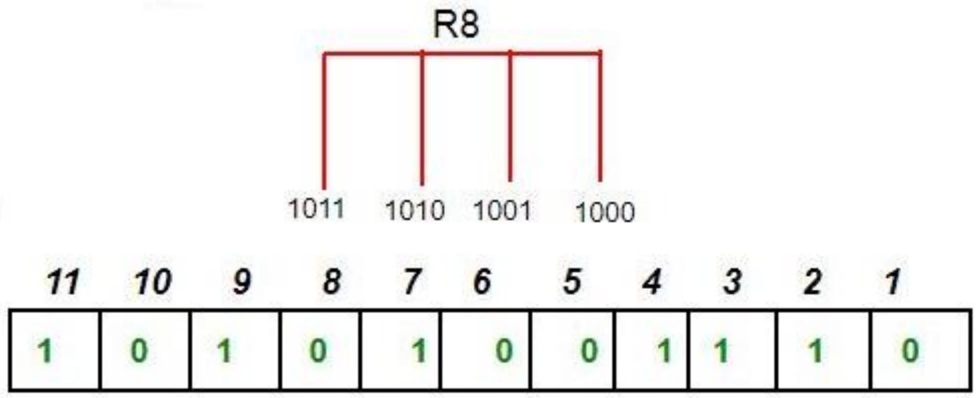
2,3,6,7,10,11 . To find the redundant bit R2, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R2 is odd the value of R2 (parity bit's value)=1



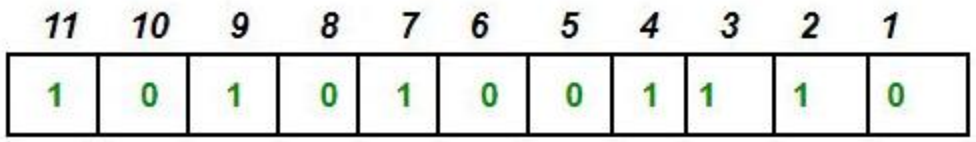
R4 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the third position from the least significant bit. R4: bits 4, 5, 6, 7 . To find the redundant bit R4, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R4 is odd the value of R4 (parity bit's value) = 1



R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit. R8: bit 8,9,10,11. To find the redundant bit R8, we check for even parity. Since the total number of 1's in all the bit positions corresponding to R8 is an even number the value of R8(parity bit's value)=0.



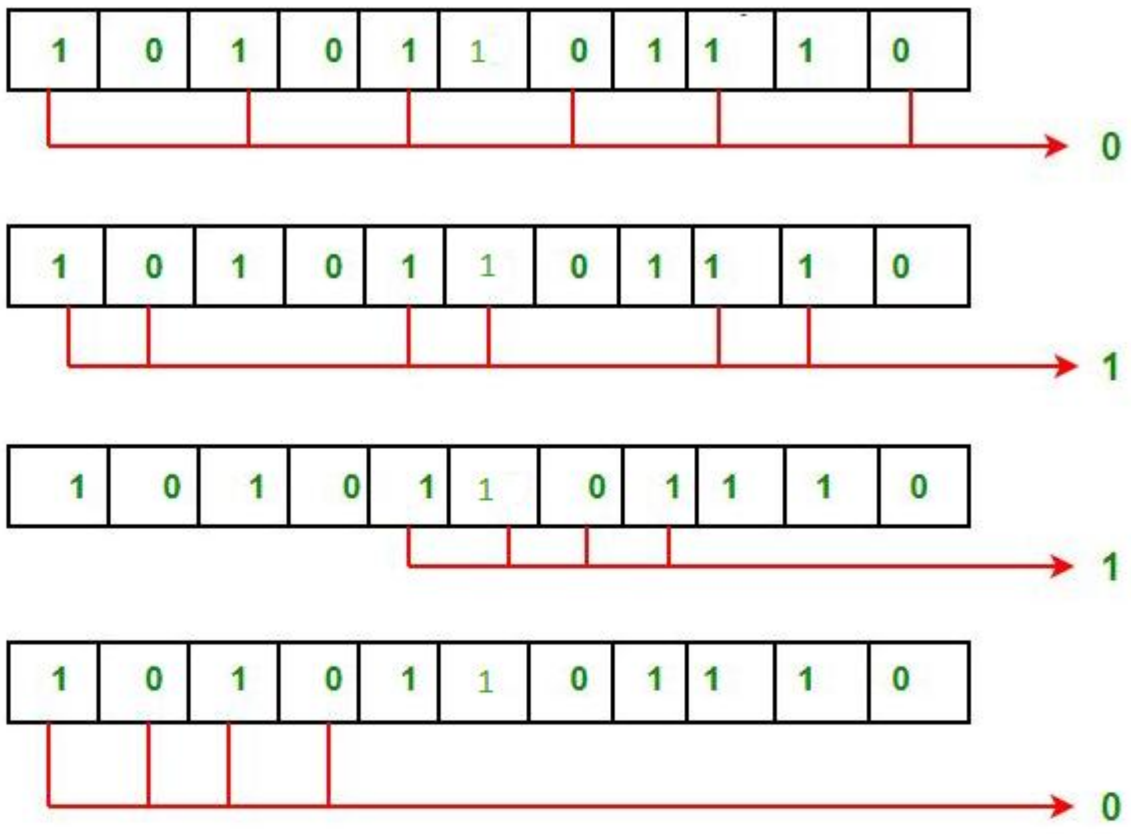
Thus, the data transferred is:



| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |

**Error detection and correction:**

Suppose in the above example the 6th bit is changed from 0 to 1 during data transmission, then it gives new parity values in the binary number:



The bits give the binary number 0110 whose decimal representation is 6. Thus, bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.