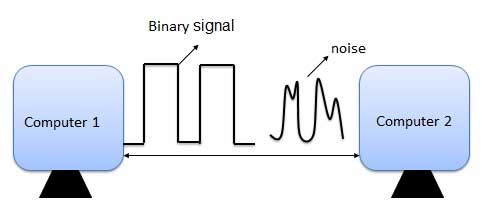
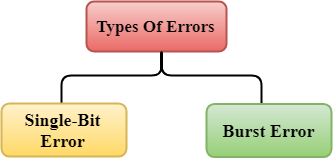
## ERROR DETECTION AND CORRECTION

## 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.



Types Of Errors

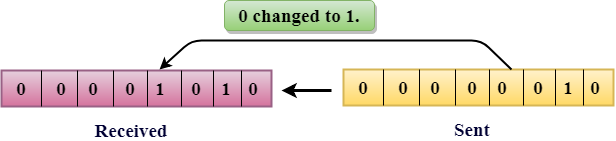


Errors can be classified into two categories:

* Single-Bit Error
* Burst Error

Single-Bit Error:

The only one bit of a given data unit is changed from 1 to 0 or from 0 to 1.



In the above figure, the message which is sent is corrupted as single-bit, i.e., 0 bit is changed to 1.

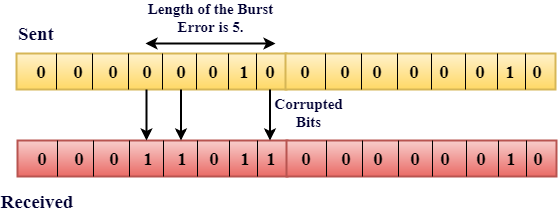
**Single-Bit Error** does not appear more likely in Serial Data Transmission. For example, Sender sends the data at 10 Mbps, this means that the bit lasts only for 1 ?s and for a single-bit error to occurred, a noise must be more than 1 ?s.

Single-Bit Error mainly occurs in Parallel Data Transmission. For example, if eight wires are used to send the eight bits of a byte, if one of the wire is noisy, then single-bit is corrupted per byte.

## Burst Error:

The two or more bits are changed from 0 to 1 or from 1 to 0 is known as Burst Error.

The Burst Error is determined from the first corrupted bit to the last corrupted bit.



The duration of noise in Burst Error is more than the duration of noise in Single-Bit.

Burst Errors are most likely to occurr in Serial Data Transmission.

The number of affected bits depends on the duration of the noise and data rate.

Error Detection

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.

Error Detecting Techniques:

Error detection is a technique that is used to check if any error occurred in the data during the transmission.

The most popular Error Detecting Techniques are:

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

**What are Error-Detecting Codes?**

Error-detecting codes are a sequence of numbers generated by specific procedures for detecting errors in data that has been transmitted over computer networks.

When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the receiver and are called errors.

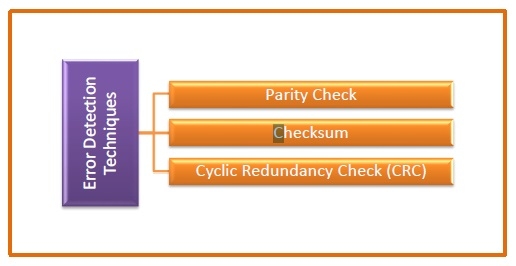
Error – detecting codes ensures messages to be encoded before they are sent over noisy channels. The encoding is done in a manner so that the decoder at the receiving end can detect whether there are errors in the incoming signal with high probability of success.

**Features of Error Detecting Codes**

* Error detecting codes are adopted when backward error correction techniques are used for reliable data transmission. In this method, the receiver sends a feedback message to the sender to inform whether an error-free message has been received or not. If there are errors, then the sender retransmits the message.
* Error-detecting codes are usually block codes, where the message is divided into fixed-sized blocks of bits, to which redundant bits are added for error detection.
* Error detection involves checking whether any error has occurred or not. The number of error bits and the type of error does not matter.

**Error Detection Techniques**

There are three main techniques for detecting errors



**Parity Check**

Parity check is done by adding an extra bit, called parity bit to the data to make number of 1s either even in case of even parity, or odd in case of odd parity.

While creating a frame, the sender counts the number of 1s in it and adds the parity bit in following way

* In case of even parity: If number of 1s is even then parity bit value is 0. If number of 1s is odd then parity bit value is 1.
* In case of odd parity: If number of 1s is odd then parity bit value is 0. If number of 1s is even then parity bit value is 1.

On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise it is rejected. Similar rule is adopted for odd parity check.

Parity check is suitable for single bit error detection only.

**Checksum**

In this error detection scheme, the following procedure is applied

* Data is divided into fixed sized frames or segments.
* The sender adds the segments using 1’s complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
* The receiver adds the incoming segments along with the checksum using 1’s complement arithmetic to get the sum and then complements it.
* If the result is zero, the received frames are accepted; otherwise they are discarded.

**Cyclic Redundancy Check (CRC)**

Cyclic Redundancy Check (CRC) involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system. The divisor is generated using polynomials.

* Here, the sender performs binary division of the data segment by the divisor. It then appends the remainder called CRC bits to the end of data segment. This makes the resulting data unit exactly divisible by the divisor.
* The receiver divides the incoming data unit by the divisor. If there is no remainder, the data unit is assumed to be correct and is accepted. Otherwise, it is understood that the data is corrupted and is therefore rejected.

**Errors and Error Detection**

When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the receiver and are called errors.

Error detection techniques are responsible for checking whether an error has occurred or not in the frame that has been transmitted via the network. It does not take into account the number of error bits and the type of error.

For error detection, the sender needs to send some additional bits along with the data bits. The receiver performs necessary checks based upon the additional redundant bits. If it finds that the data is free from errors, it removes the redundant bits before passing the message to the upper layers.

There are three main techniques for detecting errors in data frames: Parity Check, Checksum and Cyclic Redundancy Check (CRC).

**Parity Bits**

The parity check is done by adding an extra bit, called parity bit, to the data to make the number of 1s either even or odd depending upon the type of parity. The parity check is suitable for single bit error detection only.

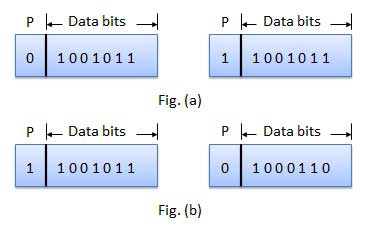
The two types of parity checking are

* **Even Parity** − Here the total number of bits in the message is made even.
* **Odd Parity** − Here the total number of bits in the message is made odd.

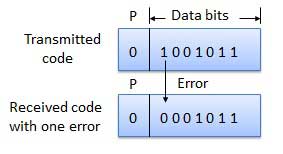
Use of Parity Bit

The parity bit can be set to 0 and 1 depending on the type of the parity required.

* For even parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is even. Shown in fig. (a).
* For odd parity, this bit is set to 1 or 0 such that the no. of "1 bits" in the entire word is odd. Shown in fig. (b).

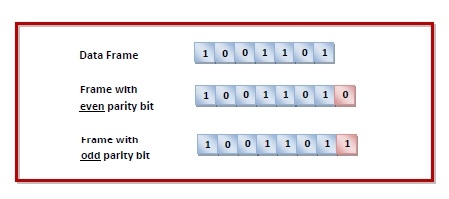


Parity checking at the receiver can detect the presence of an error if the parity of the receiver signal is different from the expected parity. That means, if it is known that the parity of the transmitted signal is always going to be "even" and if the received signal has an odd parity, then the receiver can conclude that the received signal is not correct. If an error is detected, then the receiver will ignore the received byte and request for retransmission of the same byte to the transmitter.



**Error Detection by Parity Check**

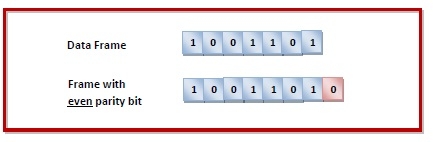
**Sender’s End** − While creating a frame, the sender counts the number of 1s in it and adds the parity bit the value of which is determined as follows -

* In the case of even parity: If a number of 1s is even, the parity bit value is 0. If a number of 1s is odd, the parity bit value is 1.
* In case of odd parity: If a number of 1s is odd, the parity bit value is 0. If a number of 1s is even, the parity bit value is 1.  
  

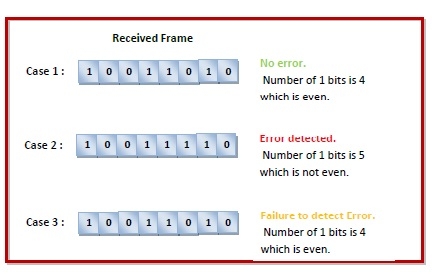
**Receiver’s End** − On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. In case of odd parity check, if the count of 1s is odd, the frame is accepted, otherwise, it is rejected.

**Example**

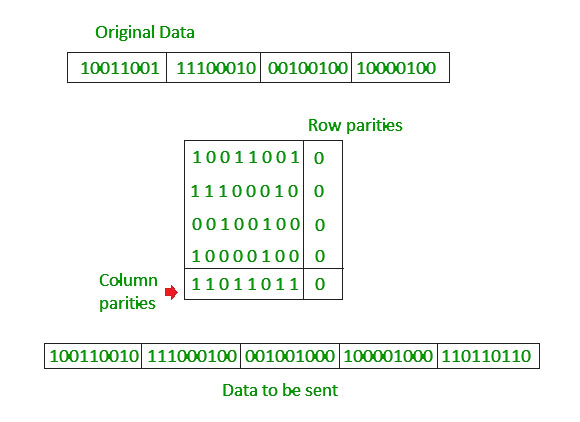
Suppose that a sender wants to send the data 1001101 using even parity check method. It will add the parity bit as shown below.



The receiver will decide whether an error has occurred by counting whether the total number of 1s is even. When the above frame is received, three cases may occur namely, no error, single bit error detection and failure to detect multiple bits error. This is illustrated as follows -



**Two-dimensional Parity check**  
Parity check bits are calculated for each row, which is equivalent to a simple parity check bit. Parity check bits are also calculated for all columns, then both are sent along with the data. At the receiving end these are compared with the parity bits calculated on the received data.

[](https://media.geeksforgeeks.org/wp-content/uploads/detect11.jpg)

**Error-Detecting Codes - Checksum**

**Checksums**

This is a block code method where a checksum is created based on the data values in the data blocks to be transmitted using some algorithm and appended to the data. When the receiver gets this data, a new checksum is calculated and compared with the existing checksum. A non-match indicates an error.

**Error Detection by Checksums**

For error detection by checksums, data is divided into fixed sized frames or segments.

* **Sender’s End** − The sender adds the segments using 1’s complement arithmetic to get the sum. It then complements the sum to get the checksum and sends it along with the data frames.
* **Receiver’s End** − The receiver adds the incoming segments along with the checksum using 1’s complement arithmetic to get the sum and then complements it.

If the result is zero, the received frames are accepted; otherwise they are discarded.

**Example**

Suppose that the sender wants to send 4 frames each of 8 bits, where the frames are 11001100, 10101010, 11110000 and 11000011.

The sender adds the bits using 1s complement arithmetic. While adding two numbers using 1s complement arithmetic, if there is a carry over, it is added to the sum.

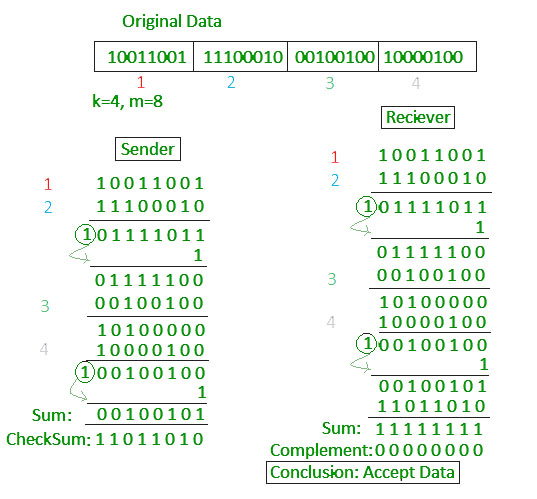
After adding all the 4 frames, the sender complements the sum to get the checksum, 11010011, and sends it along with the data frames.

The receiver performs 1s complement arithmetic sum of all the frames including the checksum. The result is complemented and found to be 0. Hence, the receiver assumes that no error has occurred.



**3. Checksum**

* In checksum error detection scheme, the data is divided into k segments each of m bits.
* In the sender’s end the segments are added using 1’s complement arithmetic to get the sum. The sum is complemented to get the checksum.
* The checksum segment is sent along with the data segments.
* At the receiver’s end, all received segments are added using 1’s complement arithmetic to get the sum. The sum is complemented.
* If the result is zero, the received data is accepted; otherwise discarded.

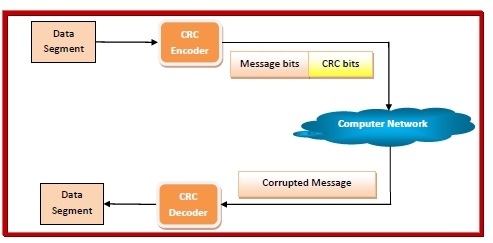
[](https://media.geeksforgeeks.org/wp-content/uploads/detect13.jpg)

## Cyclic Redundancy Check (CRC)

Cyclic Redundancy Check (CRC) is a block code invented by W. Wesley Peterson in 1961. It is commonly used to detect accidental changes to data transmitted via telecommunications networks and storage devices.

CRC involves binary division of the data bits being sent by a predetermined divisor agreed upon by the communicating system. The divisor is generated using polynomials. So, CRC is also called polynomial code checksum.

The process is illustrated as follows −



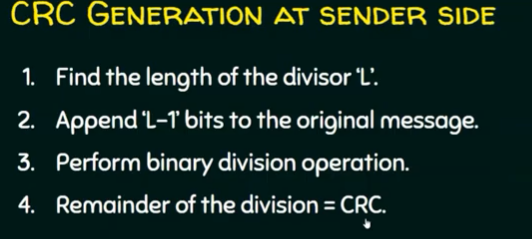
## Encoding using CRC

* The communicating parties agrees upon the size of message block and the CRC divisor. For example, the block chosen may be CRC (7, 4), where 7 is the total length of the block and 4 is the number of bits in the data segment. The divisor chosen may be 1011.
* The sender performs binary division of the data segment by the divisor.
* It then appends the remainder called CRC bits to the end of data segment. This makes the resulting data unit exactly divisible by the divisor.

## Decoding

* The receiver divides the incoming data unit by the divisor.
* If there is no remainder, the data unit is assumed to be correct and is accepted.
* Otherwise, it is understood that the data is corrupted and is therefore rejected. The receiver may then send an erroneous acknowledegment back to the sender for retransmission.

**Cyclic redundancy check (CRC)**

* Unlike checksum scheme, which is based on addition, CRC is based on binary division.
* In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
* At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
* A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.
* 

## **Example :**

## [4](https://media.geeksforgeeks.org/wp-content/uploads/detect15.jpg)

## 

## 

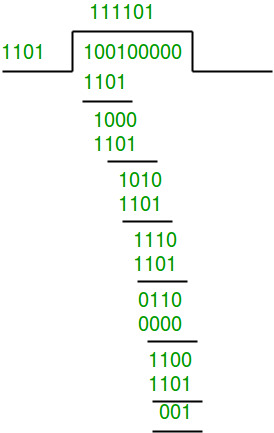
## 

**Example 1 (No error in transmission):**

Data word to be sent - 100100

Key - 1101 [ Or generator polynomial x3 + x2 + 1]

Sender Side:

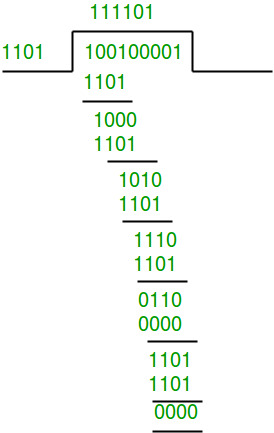
[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/rational1.jpg)

Therefore, the remainder is 001 and hence the encoded

data sent is 100100001.

Receiver Side:

Code word received at the receiver side 100100001

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/rational2.jpg)

Therefore, the remainder is all zeros. Hence, the

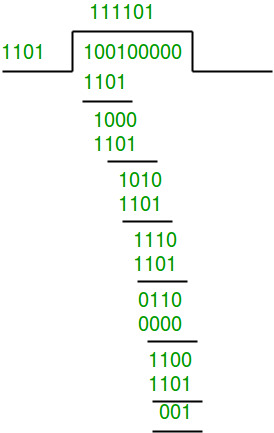
data received has no error.

**Example 2: (Error in transmission)**

Data word to be sent - 100100

Key - 1101

Sender Side:

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/rational1.jpg)

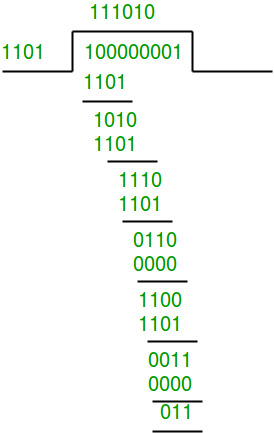
Therefore, the remainder is 001 and hence the

code word sent is 100100001.

Receiver Side

Let there be an error in transmission media

Code word received at the receiver side - 100000001

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/rational4.jpg)

Since the remainder is not all zeroes, the error

is detected at the receiver side.

## ****Cyclic Redundancy Check-****

* Cyclic Redundancy Check (CRC) is an error detection method.
* It is based on binary division.

## ****CRC Generator-****

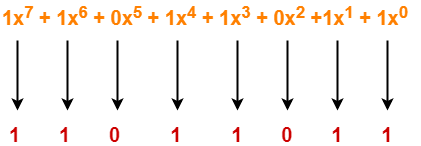
* CRC generator is an algebraic polynomial represented as a bit pattern.
* Bit pattern is obtained from the CRC generator using the following rule-

|  |
| --- |
| The power of each term gives the position of the bit and the coefficient gives the value of the bit. |

## ****Example-****

Consider the CRC generator is x7 + x6 + x4 + x3 + x + 1.

The corresponding binary pattern is obtained as-



Thus, for the given CRC generator, the corresponding binary pattern is 11011011.

## ****Properties Of CRC Generator-****

The algebraic polynomial chosen as a CRC generator should have at least the following properties-

### ****Rule-01:****

* It should not be divisible by x.
* This condition guarantees that all the burst errors of length equal to the length of polynomial are detected.

### ****Rule-02:****

* It should be divisible by x+1.
* This condition guarantees that all the burst errors affecting an odd number of bits are detected.

## ****Important Notes-****

If the CRC generator is chosen according to the above rules, then-

* CRC can detect all single-bit errors
* CRC can detect all double-bit errors provided the divisor contains at least three logic 1’s.
* CRC can detect any odd number of errors provided the divisor is a factor of x+1.
* CRC can detect all burst error of length less than the degree of the polynomial.
* CRC can detect most of the larger burst errors with a high probability.

## ****Steps Involved-****

Error detection using CRC technique involves the following steps-

### ****Step-01: Calculation Of CRC At Sender Side-****

At sender side,

* A string of n 0’s is appended to the data unit to be transmitted.
* Here, n is one less than the number of bits in CRC generator.
* Binary division is performed of the resultant string with the CRC generator.
* After division, the remainder so obtained is called as **CRC**.
* It may be noted that CRC also consists of n bits.

### ****Step-02: Appending CRC To Data Unit-****

At sender side,

* The CRC is obtained after the binary division.
* The string of n 0’s appended to the data unit earlier is replaced by the CRC remainder.

### ****Step-03: Transmission To Receiver-****

* The newly formed code word (Original data + CRC) is transmitted to the receiver.

### ****Step-04: Checking at Receiver Side-****

At receiver side,

* The transmitted code word is received.
* The received code word is divided with the same CRC generator.
* On division, the remainder so obtained is checked.

The following two cases are possible-

### ****Case-01: Remainder = 0****

If the remainder is zero,

* Receiver assumes that no error occurred in the data during the transmission.
* Receiver accepts the data.

### ****Case-02: Remainder ≠ 0****

If the remainder is non-zero,

* Receiver assumes that some error occurred in the data during the transmission.
* Receiver rejects the data and asks the sender for retransmission.

**Also Read-** [**Parity Check**](https://www.gatevidyalay.com/parity-check-parity-bit-error-detection/)

## ****PRACTICE PROBLEMS BASED ON CYCLIC REDUNDANCY CHECK (CRC)-****

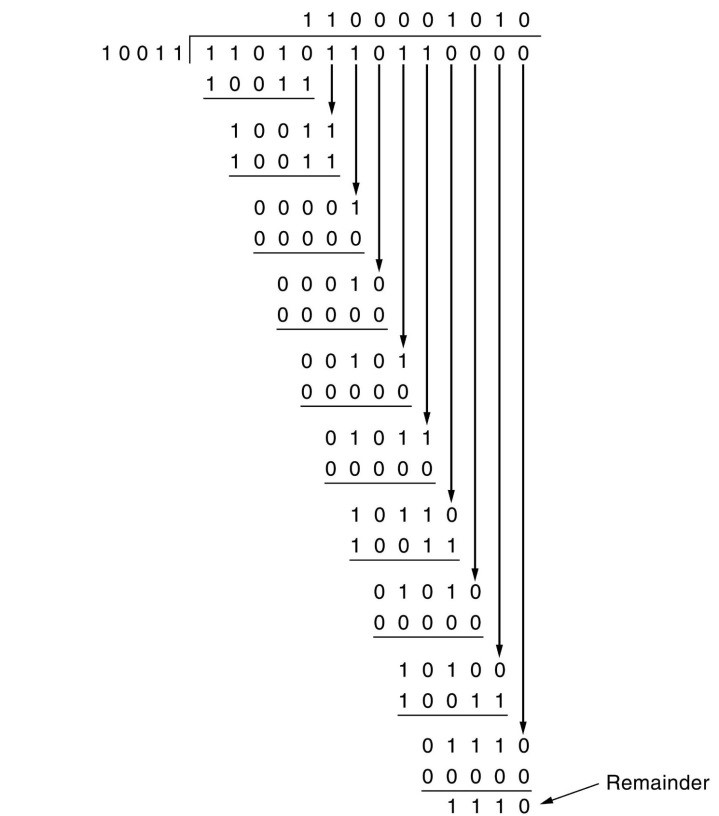
## ****Problem-01:****

A bit stream 1101011011 is transmitted using the standard CRC method. The generator polynomial is x4+x+1. What is the actual bit string transmitted?

## ****Solution-****

* The generator polynomial G(x) = x4 + x + 1 is encoded as 10011.
* Clearly, the generator polynomial consists of 5 bits.
* So, a string of 4 zeroes is appended to the bit stream to be transmitted.
* The resulting bit stream is 1101011011**0000**.

Now, the binary division is performed as-



From here, CRC = 1110.

Now,

* The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011011**0000**with the CRC.
* Thus, the code word transmitted to the receiver = 1101011011**1110**.

## ****Problem-02:****

A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x3+1.

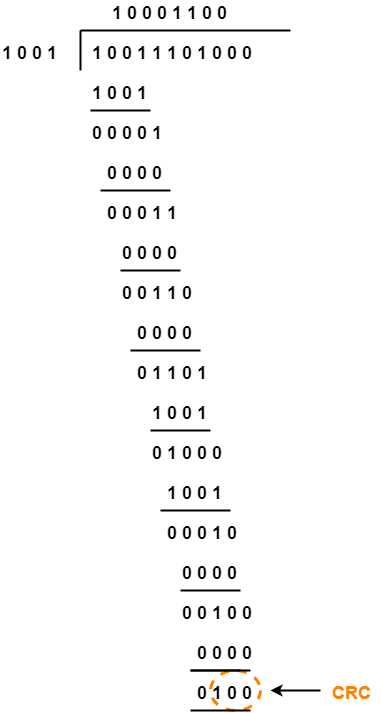
1. What is the actual bit string transmitted?
2. Suppose the third bit from the left is inverted during transmission. How will receiver detect this error?

## ****Solution-****

### ****Part-01:****

* The generator polynomial G(x) = x3 + 1 is encoded as 1001.
* Clearly, the generator polynomial consists of 4 bits.
* So, a string of 3 zeroes is appended to the bit stream to be transmitted.
* The resulting bit stream is 10011101**000**.

Now, the binary division is performed as-



From here, CRC = 100.

Now,

* The code word to be transmitted is obtained by replacing the last 3 zeroes of 10011101**000**with the CRC.
* Thus, the code word transmitted to the receiver = 10011101**100**.

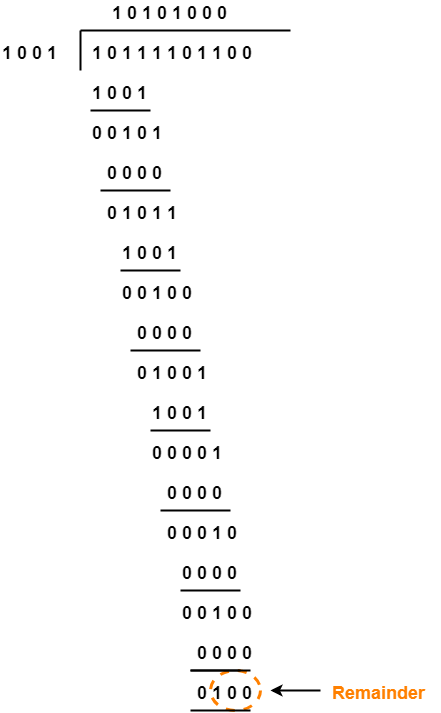
## ****Part-02:****

According to the question,

* Third bit from the left gets inverted during transmission.
* So, the bit stream received by the receiver = 10111101100.

Now,

* Receiver receives the bit stream = 10111101100.
* Receiver performs the binary division with the same generator polynomial as-



From here,

* The remainder obtained on division is a non-zero value.
* This indicates to the receiver that an error occurred in the data during the transmission.
* Therefore, receiver rejects the data and asks the sender for retransmission.

## Errors and Error Correcting Codes

When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the receiver and are called errors.

Error-correcting codes (ECC) are a sequence of numbers generated by specific algorithms for detecting and removing errors in data that has been transmitted over noisy channels. Error correcting codes ascertain the exact number of bits that has been corrupted and the location of the corrupted bits, within the limitations in algorithm.

ECCs can be broadly categorized into two types −

* **Block codes** − The message is divided into fixed-sized blocks of bits, to which redundant bits are added for error detection or correction.
* **Convolutional codes** − The message comprises of data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream.

## Hamming Code

Hamming code is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. It was developed by R.W. Hamming for error correction.

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.

## Encoding a message by Hamming Code

The procedure used by the sender to encode the message encompasses the following steps −

* **Step 1** − Calculation of the number of redundant bits.
* **Step 2** − Positioning the redundant bits.
* **Step 3** − Calculating the values of each redundant bit.

Once the redundant bits are embedded within the message, this is sent to the user.

## Step 1 − Calculation of the number of redundant bits.

If the message contains m𝑚number of data bits, r𝑟number of redundant bits are added to it so that m𝑟 is able to indicate at least (m + r+ 1) different states. Here, (m + r) indicates location of an error in each of (𝑚 + 𝑟) bit positions and one additional state indicates no error. Since, r𝑟 bits can indicate 2r𝑟 states, 2r𝑟 must be at least equal to (m + r + 1). Thus the following equation should hold  2r ≥ m+r+1

## Step 2 − Positioning the redundant bits.

The r redundant bits placed at bit positions of powers of 2, i.e. 1, 2, 4, 8, 16 etc. They are referred in the rest of this text as r1 (at position 1), r2 (at position 2), r3 (at position 4), r4 (at position 8) and so on.

## Step 3 − Calculating the values of each redundant bit.

The redundant bits are parity bits. A parity bit is an extra bit that makes the number of 1s either even or odd. The two types of parity are −

* **Even Parity** − Here the total number of bits in the message is made even.
* **Odd Parity** − Here the total number of bits in the message is made odd.

Each redundant bit, ri, is calculated as the parity, generally even parity, based upon its bit position. It covers all bit positions whose binary representation includes a 1 in the ith position except the position of ri. Thus −

* r1 is the parity bit for all data bits in positions whose binary representation includes a 1 in the least significant position excluding 1 (3, 5, 7, 9, 11 and so on)
* r2is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 2 from right except 2 (3, 6, 7, 10, 11 and so on)
* r3 is the parity bit for all data bits in positions whose binary representation includes a 1 in the position 3 from right except 4 (5-7, 12-15, 20-23 and so on)

## Decoding a message in Hamming Code

Once the receiver gets an incoming message, it performs recalculations to detect errors and correct them. The steps for recalculation are −

* **Step 1** − Calculation of the number of redundant bits.
* **Step 2** − Positioning the redundant bits.
* **Step 3** − Parity checking.
* **Step 4** − Error detection and correction

Hamming Code in Computer Network

Last Updated: 04-05-2020

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. It is **technique developed by R.W. Hamming for error correction**.

**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 ≥ m + r + 1

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 ≥ 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:

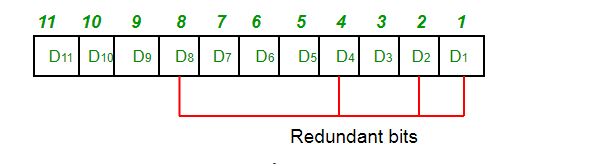
1. **Even parity bit:**  
   In the case of even parity, for a given set of bits, the number of 1’s are counted. If that count is odd, the parity bit value is set to 1, making the total count of occurrences of 1’s an even number. If the total number of 1’s in a given set of bits is already even, the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

**General Algorithm of Hamming code –**  
The Hamming Code is simply the use of extra parity bits to allow the identification of an error.

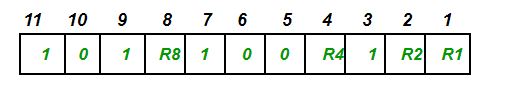
1. Write the bit positions starting from 1 in binary form (1, 10, 11, 100, etc).
2. All the bit positions that are a power of 2 are marked as parity bits (1, 2, 4, 8, etc).
3. All the other bit positions are marked as data bits.
4. Each data bit is included in a unique set of parity bits, as determined its bit position in binary form.  
   **a.** Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significant  
   position (1, 3, 5, 7, 9, 11, etc).  
   **b.** Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position from  
   the least significant bit (2, 3, 6, 7, 10, 11, etc).  
   **c.** Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position from  
   the least significant bit (4–7, 12–15, 20–23, etc).  
   **d.** Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position from  
   the least significant bit bits (8–15, 24–31, 40–47, etc).  
   **e.** In general, each parity bit covers all bits where the bitwise AND of the parity position and the bit position is  
   non-zero.
5. Since we check for even parity set a parity bit to 1 if the total number of ones in the positions it checks is  
   odd.
6. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

**Determining the position of redundant bits –**  
These redundancy bits are placed at the positions which correspond to the power of 2.  
As in the above example:

1. The number of data bits = 7
2. The number of redundant bits = 4
3. The total number of bits = 11
4. The redundant bits are placed at positions corresponding to power of 2- 1, 2, 4, and 8



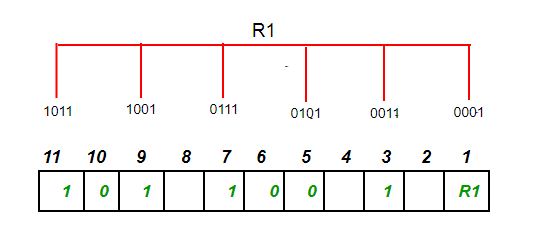
Suppose the data to be transmitted is 1011001, the bits will be placed as follows:



**Determining the Parity bits –**

1. R1 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the least significant position.

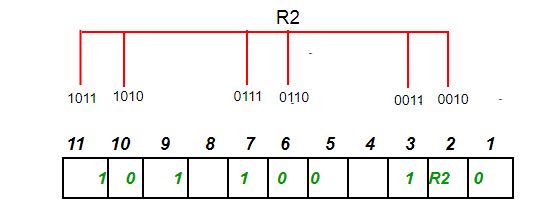
R1: bits 1, 3, 5, 7, 9, 11



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

1. 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.

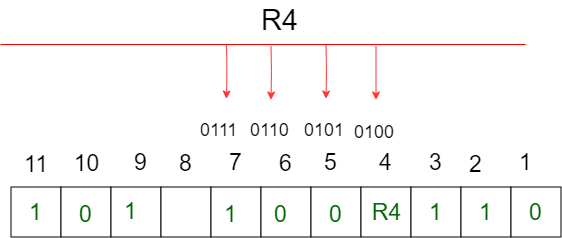
R2: bits 2,3,6,7,10,11



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

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.

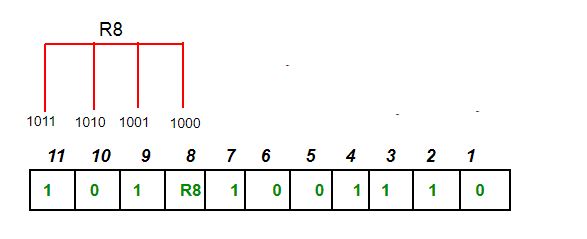
R4: bits 4, 5, 6, 7



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

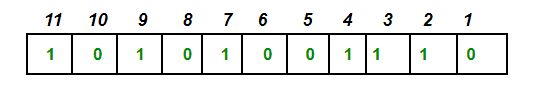
1. R8 bit is calculated using parity check at all the bits positions whose binary representation includes a 1 in the fourth position from the least significant bit.

R8: bit 8,9,10,11

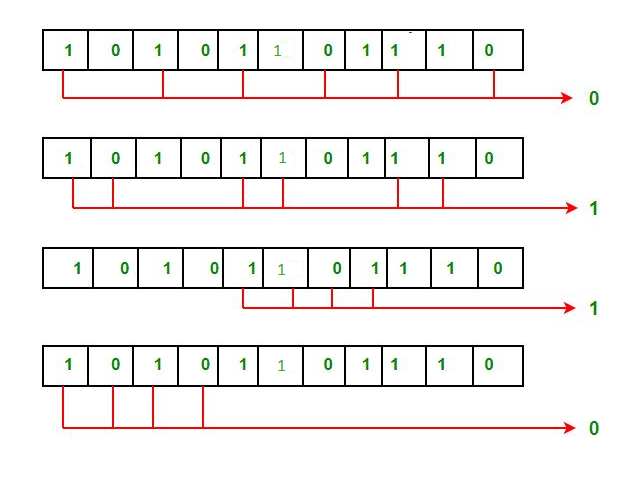


To find the redundant bit R8, we check for even parity. Since the total number of 1’s in all the bit positions corresponding to R8 is an even number the value of R8(parity bit’s value)=0.

Thus, the data transferred is:



**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 as 0110 whose decimal representation is 6. Thus, the bit 6 contains an error. To correct the error the 6th bit is changed from 1 to 0.

**What is a Hamming code?**

In [computer science](https://en.wikipedia.org/wiki/Computer_science) and [telecommunication](https://en.wikipedia.org/wiki/Telecommunication), **Hamming codes** are a family of [linear error-correcting codes](https://en.wikipedia.org/wiki/Linear_code). Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors. By contrast, the simple [parity code](https://en.wikipedia.org/wiki/Parity_bit) cannot correct errors, and can detect only an odd number of bits in error. Hamming codes are [perfect codes](https://en.wikipedia.org/wiki/Perfect_code), that is, they achieve the highest possible [rate](https://en.wikipedia.org/wiki/Block_code#The_rate_R) for codes with their [block length](https://en.wikipedia.org/wiki/Block_code#The_block_length_n) and [minimum distance](https://en.wikipedia.org/wiki/Block_code#The_distance_d) of three.[[1]](https://en.wikipedia.org/wiki/Hamming_code#cite_note-1) [Richard W. Hamming](https://en.wikipedia.org/wiki/Richard_Hamming) invented Hamming codes in 1950 .

Hamming code is a liner code that is useful for error detection up to two immediate bit errors. It is capable of single-bit errors.

In Hamming code, the source encodes the message by adding redundant bits in the message. These redundant bits are mostly inserted and generated at certain positions in the message to accomplish error detection and correction process.

**History of Hamming code**

* Hamming code is a technique build by R.W.Hamming to detect errors.
* Hamming code should be applied to data units of any length and uses the relationship between data and redundancy bits.
* He worked on the problem of the error-correction method and developed an increasingly powerful array of algorithms called Hamming code.
* In 1950, he published the Hamming Code, which widely used today in applications like ECC memory.

**Application of Hamming code**

Here are some common applications of using Hemming code:

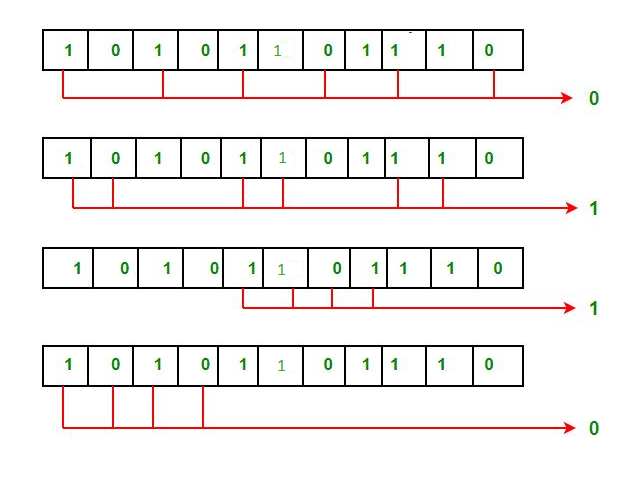
* Satellites
* Computer Memory
* Modems
* PlasmaCAM
* Open connectors
* Shielding wire
* Embedded Processor

**Advantages of Hamming code**

* Hamming code method is effective on networks where the data streams are given for the single-bit errors.
* Hamming code not only provides the detection of a bit error but also helps you to indent bit containing error so that it can be corrected.
* The ease of use of hamming codes makes it best them suitable for use in computer memory and single-error correction.

**Disadvantages of Hamming code**

* Single-bit error detection and correction code. However, if multiple bits are founded error, then the outcome may result in another bit which should be correct to be changed. This can cause the data to be further errored.
* Hamming code algorithm can solve only single bits issues.

‘

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

frame 1 : 10101010

frame 2: 11110000

---------------------------

110011010

1

----------------------------

10011011

frame 3: 11001100

--------------------------

101100111

1

-----------------------------

01101000

frame 4: 00011100

-----------------------------

SUM : 10000100

CHECK SUM : 01111011

----------------------------------

AT RECEIVER END , THE FOUR FRAMES AND CHECKSUM IS REXED. SUM IS CALCULATED, ADD CHECKSUM, TAKE COMPLEMENT , IF ALL ZEROS THEN NO ERROR

ASSUME ERROR AT FRAME 4 : 10011100 INSTEAD OF 00011100 THEN SUM WILL BE 00000101

checksum is 01111011. result is not all 1's 000001010

011111011

-------------------------

10000011 therefore error

Modulo 2 Arithmetic

Modulo 2 arithmetic is performed digit by digit on binary numbers.

Each digit is considered independently from its neighbours.

Numbers are not carried or borrowed.

Addition/Subtraction

Modulo 2 addition/subtraction is performed using an **exclusive OR (xor)** operation on the corresponding binary digits of each operand.

0 ± 0 = 0; 0 ± 1 = 1; 1 ± 0 = 1; 1 ± 1 = 0