**Title:**

**Computer Networks Lab Assignment-1**

**Name: Rahul Kundu**

**Class: BCSE-III**

**Roll No: 001810501085**

**Problem statement:** Design and implement an error detection module which has four schemes namely LRC, VRC, Checksum and CRC. Please note that you may need to use these schemes separately for other applications (assignments). You can write the program in any language. The Sender program should accept the name of a test file (contains a sequence of 0, 1) from the command line. Then it will prepare the data frame (decide the size of the frame) from the input. Based on the schemes, codeword will be prepared. Sender will send the codeword to the Receiver. Receiver will extract the dataword from codeword and show if there is any error detected. Test the same program to produce a PASS/FAIL result for following cases. (a) Error is detected by all four schemes. Use a suitable CRC polynomial (list is given in next page). (b) Error is detected by checksum but not by CRC. (c) Error is detected by VRC but not by CRC. [Note: Inject error in random positions in the input data frame. Write a separate method for that.]

**Submission date: 5/11/20**

**Submission deadline: 5/11/20**

**Design**

**PURPOSE**

This is an error detection module which has four schemes namely LRC, VRC, Checksum and CRC. These programs accepts a test file from the command line which contains a sequence of 0’s and 1’s. This sequence of 0’s and 1’s acts as the codeword for the test. The codeword is then divided into data frames of size n and the value of n is taken as an input from command line. Then the schemes (VRC,LRC,CheckSum and CRC) are applied and a new codeword is generated. The data is then transferred to the receiver. Before the data is received there is a facility to inject errors to the data. After the data is received the program test the data to produce PASS/FAIL result.

**STRUCTURE DIAGRAM**

Error Detection Methods

**Checksum**

**CRC**

**LRC**

**VRC**

**RECEIVER**

**SENDER**

**HELPER**

1.Accepts codeword

2.Modifies codeword by injecting errors

3.Sends erroneous codeword to receiver

1 . Accepts erroneous codeword

2 . Detects errors according to the error detection scheme being used

3.Returns: Passed/Failed

1.Reads dataword

2.Adds redundant bit(s) according to the error detection scheme being used

3.Sends codeword to helper function

Flowchart

Support to inject error in the codeword to be sent to the receiver. And the data is received by receiver

The receiver checks the data using proper scheme to generate PASS/FAIL results

Accept test file from command line (contains a sequence of 0s and 1s which is the codeword)

One of the Schemes (VRC / LRC / Checksum / CRC) is applied to generate new codeword

The codeword is broken into k data frames of size n (value is taken as input from command line)

**Input:** A test file is taken as input from the command line which contains a sequence of 0s and 1s.

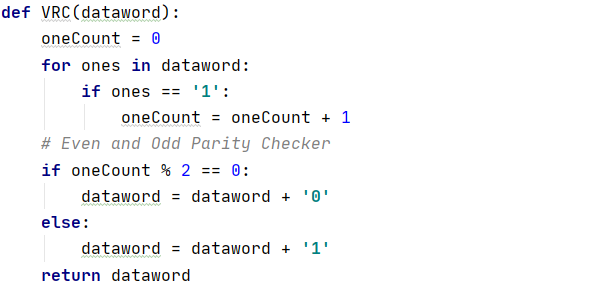
**Output:** PASS (if no error is detected) or FAIL (if error is detected).

**Implementation:**

I’ve implemented these error detection methods in python. The code snippet are as follows.

**VRC(VERTICAL REDUNDANCY CHECK):**

**Code:**



**Explaination:**

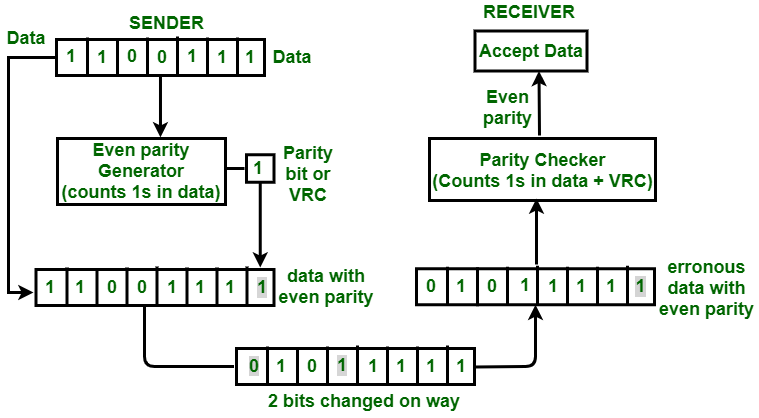
This function checks the parity of the dataword and adds either 0 or 1 at the end of the dataword according to the parity. (0 for even parity and 1 for odd parity)

To check the number of one, Count the number of 1’s in the codeword and check count %2 to get the parity.

**Advantages:**

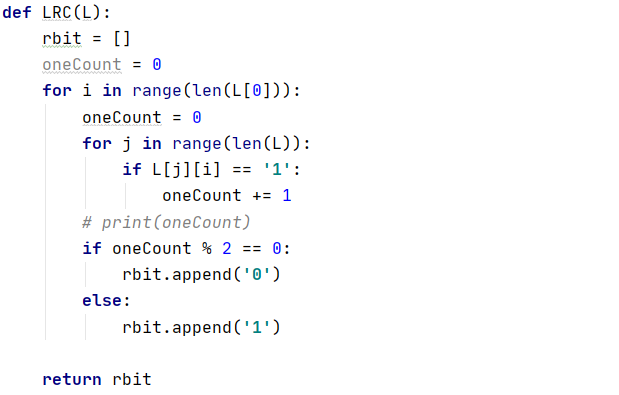
* VRC can detect all single bit error.
* It can also detect burst errors but only in those cases where number of bits changed is odd, i.e. 1, 3, 5, 7…….etc.

**Disadvantages :**  
The major disadvantage of using this method for error detection is that it is not able to detect burst error if the number of bits changed is even, i.e. 2, 4, 6, 8, …….etc.



**LRC(Longitudinal Redundancy Check):**

**Code:**



**Explaination:**

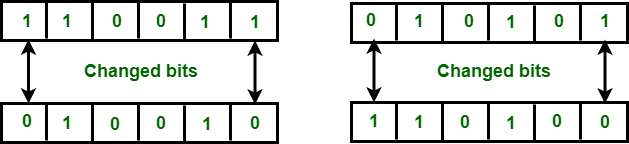
This function checks the parity of the datawords(in columns). Here I took an empty list and appended either 1 or 0 according to the parity of each column and then joined them to create a string to append it to the codeword.

**Advantage:**  
LRC is used to detect burst errors.

**Disadvantage:**  
The main problem with LRC is that, it is not able to detect error if two bits in a data unit are damaged and two bits in exactly the same position in other data unit are also damaged

**Example :**

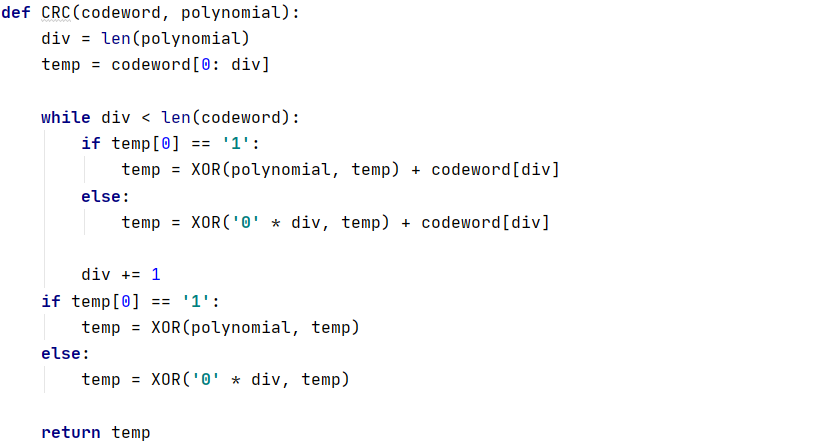
 If data 110011 010101 is changed to 010010110100.



In this example 1st and 6th bit in one data unit is changed. Also the 1st and 6th bit in second unit is changed.

**CRC(Circular Redundancy Check):**

**Code:**



**Explaination:**

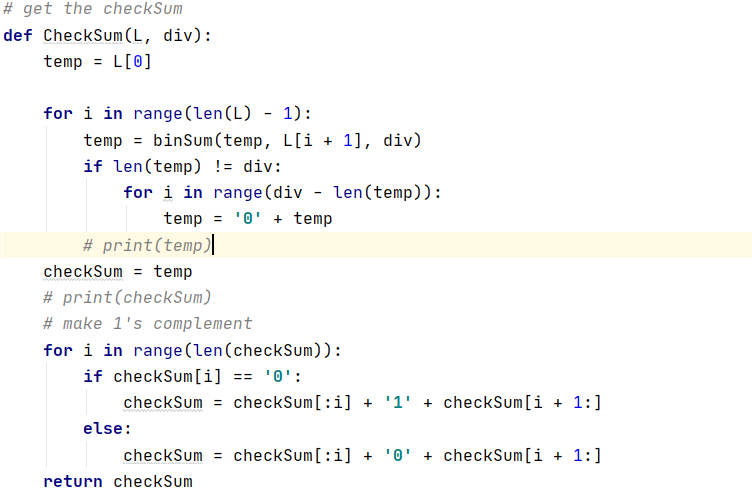
This function performs CRC division and returns the remainder. For XOR –ing there is a separate function defined. To get the remainder a loop is initiated until the end of the string is reached. It updates the string with each iteration depending upon the first element of the XOR –ed string.

**Advantages:** CRC is simple to implement in binary hardware, Mathematical analysis of CRC is very simple, and it is good at detecting common errors caused by noise in transmission.

**Disadvantages:** CRC is not suitable for protecting against intentional alteration of data, and overflow of data is possible in CRC.

**CheckSum:**

**Code:**



**Explaination:**

This function performs the Check sum operation. The dataword generated are added using binary addition. A loop is initiated till it reach iteration number equal to the number of datawords as two datawords are added in each iteration. After getting the binary sum, we make it ones complement. To perform this dataword strings are sliced and readed to generate the checksum.

**Advantages:** 1.The CheckSum detects all errors involving an odd number of bits.

2.It detects most errors involving an even number of bits.

**Disadvantages:** If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sums of those columns will not change and receiver will not detect the error(s).

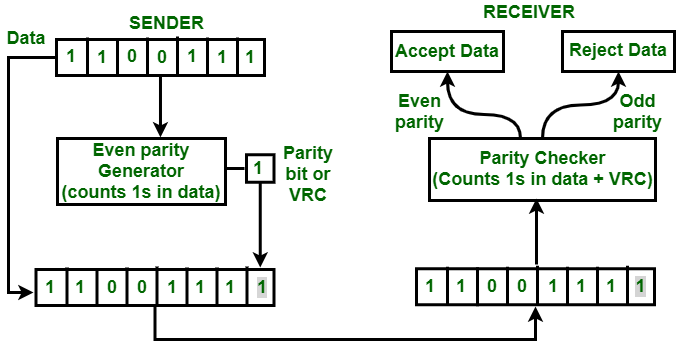
**Explantation with Testcase:**

**VRC:**

**Vertical Redundancy Check**is also known as Parity Check. In this method, a redundant bit also called parity bit is added to each data unit. This method includes even parity and odd parity. Even parity means the total number of 1s in data is to be even and odd parity means the total number of 1s in data is to be odd.

**Original data🡪 1100111**

**Received data (along with VRC)🡪 11001111**



If the original data is 1100111. After adding VRC, data unit that will be transmitted is 11001111. Suppose on the way 2 bits are 01011111. When this data will reach the destination, parity checker will count number of 1s in data and that comes out to be even i.e. 8. So, in this case, parity is not changed, it is still even.

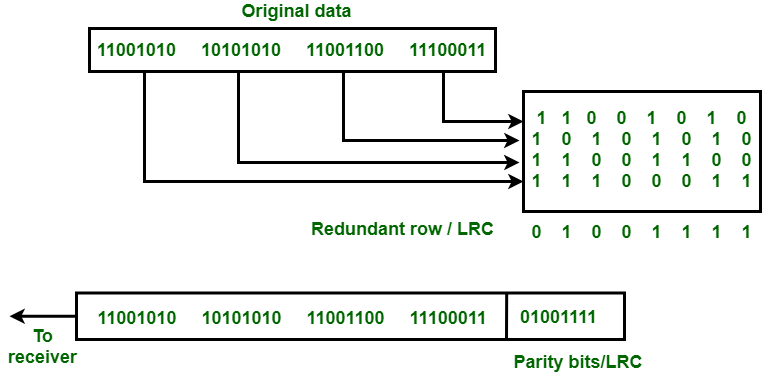
**LRC:**

[**Longitudinal Redundancy Check (LRC)**](https://practice.geeksforgeeks.org/problems/what-is-lrc) is also known as 2-D parity check. In this method, data which the user want to send is organised into tables of rows and columns. A block of bit is divided into table or matrix of rows and columns. In order to detect an error, a redundant bit is added to the whole block and this block is transmitted to receiver. The receiver uses this redundant row to detect error. After checking the data for errors, receiver accepts the data and discards the redundant row of bits.

**Original data🡪11001010 10101010 11001100 11100011**

**Received data (along with LRC)🡪 11001010 10101010 11001100 11100011 01001111**

If a block of 32 bits is to be transmitted, it is divided into matrix of four rows and eight columns which as shown in the following figure:



In this matrix of bits, a parity bit (odd or even) is calculated for each column. It means 32 bits data plus 8 redundant bits are transmitted to receiver. Whenever data reaches at the destination, receiver uses LRC to detect error in data.

**CRC:**

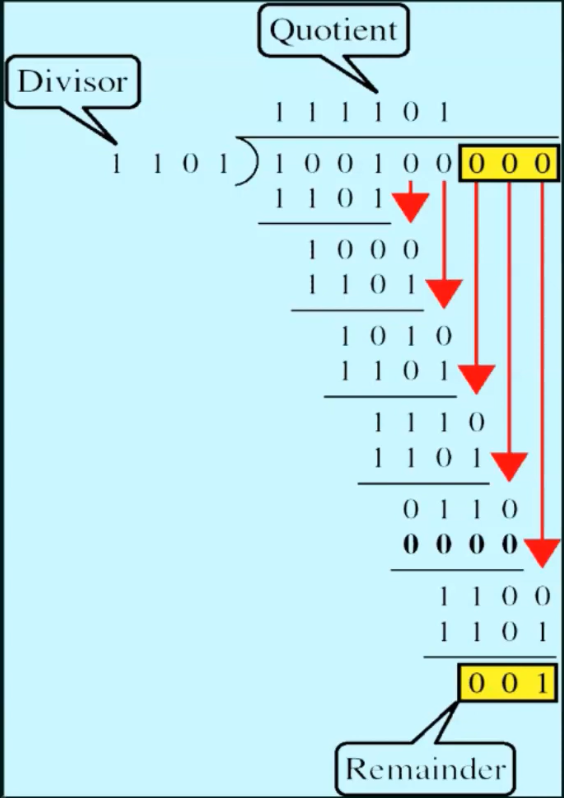
CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel.  
CRC uses Generator Polynomial which is available on both sender and receiver sides. An example generator polynomial is of the form like x^3 + 1. This generator polynomial represents key 1001. Another example is x^2 + x. that represents key 110.

**Original data 🡪 100100**

**CRC 🡪 001**

**Received data (along with CRC) 🡪100100001**

**Sender Side:**



**Receiver Side:**

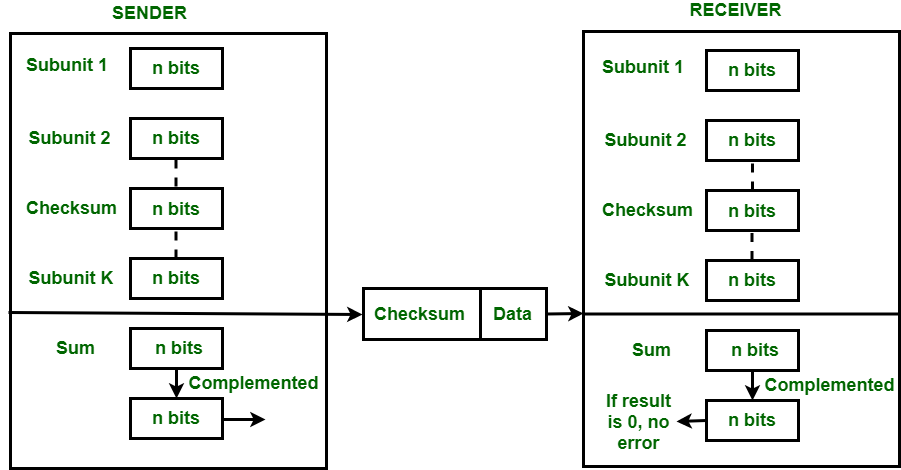


**CheckSum:**

**Original data 🡪11001100 10101010 11110000 11000011**

**Received data (along with CheckSum) 🡪1001100 10101010 11110000 11000011 11010011**

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 1. Hence, the receiver assumes that no error has occurred.



**ERROR IS NOT DETECTED BY CRC BUT BY VRC AND CHECKSUM:**

Here I use CRC-8-ATM which is given by **111010101** or whose 0th 1st 2nd 4th 6th and 8th bits are 1. If there is an error which changes the bits of the codeword at similar position then the CRC division can detect the error.

But on the contrary, it is detectable by VRC and Checksum. As odd number of changes takes place in the data which is easily detectable by even parity and for checksum the sum will not be zero at the receiver end as odd number of changes take place.

RESULTS AND ANALYSIS:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NAME | BIT ERROR | BURST ERROR | COMMENTS | EFFICIENCY |
| VRC | **Detected** | **Not Detected**  **Properly** | It can also detect burst errors but only in those cases where number of bits changed is odd but not for the even ones. | Poor |
| LRC | **Detected** | **Detect**  **Most of It** | It is not able to detect error if two bits in a data unit are damaged and two bits in exactly the same position in other data unit are also damaged | Better Than VRC |
| CHECKSUM | **Detected** | **Detect**  **Most of It** | If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sums of those columns will not change and receiver will not detect the error(s). | Better Than LRC |
| CRC | **Detected** | **Detected** | It can detect burst errors of length equal to maximum polynomial degree only | Mostly used in Real Life and Better Than CheckSum |

**Comments:**

**What did I learn from it?**

During the making of this assignment I got to know about 4 different error detection schemes (VRC,LRC,CRC,CheckSum)clearly.

**Was it too hard? (explain why?) Too easy? (explain why?)**

Actually I was facing many difficulties when I started implementing the Assignment . Later on I realized that it was not as hard as I thought .

So, I think it is neither too easy nor too hard for me.