Department of
Computer Science and Engineering

Title: Implementing Cyclic Redundancy Check and Parity Checker.

Data Communication Lab

CSE 208

Green University of Bangladesh

1 Objective(s)

- To attain knowledge on the parity checking and how it works.
- To attain knowledge on the cyclic redundancy check (CRC) and how it works.
- To implement parity checker and cyclic redundancy check (CRC) error detection algorithms.

2 Problem Analysis

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 we will study today:

- Parity Checking
- Cyclic Redundancy Check (CRC)

2.1 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 1. Even parity and 2. 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 and 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 and Even number of 1's, the parity bit is 1. Ex: data is 10010101 -> parity bit 1

2.1.1 Algorithm for Even Parity Checking:

1. Start.
2. Take the data from the user.
3. Count the number of 1's.
4. If the number of 1's is even then append a '1' at the MSB position of the data and print the data. Otherwise, append a '0' at the MSB position of the data and print the data.
5. End.

2.1.2 Implementation of Even Parity Checking:

```
#include<iostream>
using namespace std;
//function for measuring array length with data
int a_length(char array[])
{
    int count = 0;
    for(int i = 0; array[i] != NULL; i++)
    {
```

```
        count++;
    }
    return count;
}

int main()
{

    char data[100];
    cout<<"This is a program for even parity checking."<<endl;
    cout<<"Enter the data: "<<endl;
    //taking user data here
    cin>>data;
    //finding the user data length
    int length= a_length(data);

    int count=0;
    for(int i=0; i<length; i++)
    {
        //checking even parity
        if(data[i]=='1')
        {
            count++;
        }
    }
    //increasing the array for adding the parity bit.
    int c=length+1;
    //new array

    if(count%2 == 0)
    {

        for(int i=c,j=c-1; i>0; i--,j--)
        {
            //copying the data to the new array
            data[i] = data[j];
        }
        //initializing the parity
        data[0]='1';
        //displaying the new array
        cout<<"After adding '1' at the front of the data: "<<endl;
        cout<<data<<endl;
    }
    else
    {
        for(int i=c,j=c-1; i>0; i--,j--)
        {
            //copying the data to the new array
            data[i] = data[j];
        }
        //initializing the parity
        data[0]='0';
        //displaying the new array
        cout<<"After adding '0' at the front of the data: "<<endl;
        cout<<data<<endl;
    }
}
```

2.1.3 Output for the code of Even Parity Checking:

```
This is a program for even parity checking.
Enter the data:
1110001
After adding '1' at the front of the data:
11110001
```

2.2 Cyclic Redundancy Check (CRC)

A cyclic code is a linear (n, k) block code with the property that every cyclic shift of a codeword results in another code word. Here k indicates the length of the message at transmitter (the number of information bits). n is the total length of the message after adding check bits. (actual data and the check bits). n, k is the number of check bits. The codes used for cyclic redundancy check there by error detection are known as CRC codes (Cyclic redundancy check codes).Cyclic redundancy-check codes are shortened cyclic codes. These types of codes are used for error detection and encoding. They are easily implemented using shift-registers with feedback connections. That is why they are widely used for error detection on digital communication. CRC codes will provide effective and high level of protection.

2.2.1 CRC Code Generation

Based on the desired number of bit checks, we will add some zeros (0) to the actual data. This new binary data sequence is divided by a new word of length n + 1, where n is the number of check bits to be added . The reminder obtained as a result of this modulo 2- division is added to the dividend bit sequence to form the cyclic code. The generated code word is completely divisible by the divisor that is used in generation of code. This is transmitted through the transmitter. If the data stream is "01111010" and the polynomial bit stream is '11011' then we can see inf fig 1 the remainder will be "0001".

```
        01011 ← Quotient
       ------------------
11011 | 01111010
      - 00000
      ----------
         1111010
       - 11011
       ---------
            010110
          - 00000
          --------
             10110
           - 11011
           -------
              11010
            - 11011
            -------
              0001 ← Remainder (mod-2)
```

Figure 1: Division of the data bit stream by the polynomial to get the remainder.

So the reveived bit stream will be "011110100001"

At the receiver side, we divide the received code word with the same divisor to get the actual code word. For an error free reception of data, the reminder is 0. If the reminder is a non – zero, that means there is an error in the received code / data sequence. The probability of error detection depends upon the number of check bits (n) used to construct the cyclic code. For single bit and two bit errors, the probability is 100 % .

2.2.2 Algorithm for CRC:

Algorithm for Encoding using CRC:

1. The communicating parties agrees upon the size of message,M(x) and the generator polynomial, G(x).
2. If r is the order of G(x),r, bits are appended to the low order end of M(x). This makes the block size bits, the value of which is xrM(x).
3. The block xrM(x) is divided by G(x) using modulo 2 division.
4. The remainder after division is added to xrM(x) using modulo 2 addition. The result is the frame to be transmitted, T(x). The encoding procedure makes exactly divisible by G(x).

Algorithm for Decoding using CRC:

1. The receiver divides the incoming data frame T(x) unit by G(x) using modulo 2 division. Mathematically, if E(x) is the error, then modulo 2 division of [M(x) + E(x)] by G(x) is done.
2. If there is no remainder, then it implies that E(x). The data frame is accepted.
3. A remainder indicates a non-zero value of E(x), or in other words presence of an error. So the data frame is rejected. The receiver may then send an erroneous acknowledgment back to the sender for re-transmission.

2.2.3 Implementation of CRC:

```
#include<stdio.h>
char data[20],div[20],temp[4],total[100];
int i,j,datalen,divlen,len,flag=1;
void check();
int main()
{
        printf("Enter the total bit of data:");
        scanf("%d",&datalen);
        printf("\nEnter the total bit of divisor");
        scanf("%d",&divlen);
        len=datalen+divlen-1;
        printf("\nEnter the data:");
        scanf("%s",&data);
    printf("\nEnter the divisor");
    scanf("%s",div);

        for(i=0;i<datalen;i++)
        {
                total[i]=data[i];
                temp[i]=data[i];
        }
        for(i=datalen;i<len;i++)
        total[i]='0';
        check();
        for(i=0;i<divlen;i++)
                temp[i+datalen]=data[i];
        printf("\ntransmitted Code Word:%s",temp);
        printf("\n\nEnter the received code word:");
    scanf("%s",total);
        check();
        for(i=0;i<divlen-1;i++)
                if(data[i]=='1')
                {
                        flag=0;
                        break;
```

```
                }
        if(flag==1)
        printf("\nsuccessful!!");
        else
        printf("\nreceived code word contains errors...\n");
}
void check()
{
    for(j=0;j<divlen;j++)
        data[j]=total[j];
    while(j<=len)
        {
        if(data[0]=='1')
            for(i = 1;i <divlen  ; i++)
                                    data[i] = (( data[i] == div[i])?'0':'1');
        for(i=0;i<divlen-1;i++)
            data[i]=data[i+1];
        data[i]=total[j++];
    }
}
```

2.2.4 Output for the code of CRC:

```
Enter the total bit of data:7

Enter the total bit of divisor4

Enter the data:1001010

Enter the divisor1011

transmitted Code Word:1001010111

Enter the received code word:1001010100

received code word contains errors...
```

3 Discussion & Conclusion

Based on the focused objective(s) to understand about the parity checker and cyclic redundancy check, the additional lab exercise made me more confident towards the fulfilment of the objectives(s).

4 Lab Task (Please implement yourself and show the output to the instructor)

1. Implement odd parity checker where user input will be the bit stream.

5 Lab Exercise (Submit as a report)

Implement cyclic redundancy verification where the polynomial bit stream and the data stream received by the receiver will be user input.

6 Policy

Copying from internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.