CS335

Assignment II

Course Instructor :

Dr. Amitangshu Pal

Submitted By :

Nishant Roshan (200643)

Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

February 19, 2023

Question 1 [20 points]

Write a program (in any language) that generates an n-bit frame for transmission from a k-bit data block D and a (n − k + 1) bit CRC pattern P. Compile and run the program with at least two set of inputs to confirm that this program is generating CRC patterns correctly.
Now, modify the program that performs the following steps:

(a) Generates a message of k =10 bits.

(b) Uses the previous code with P=110101 to generate the corresponding 15-bit frame T for transmission.

(c) Generates transmission errors at any bit positions of T.

(d) Applies CRC to the received frame (i.e. frame T after introducing errors) to determine if the frame should be accepted or discarded.

Attach your code with your report, and also put comments stating the instructions of how to run it.

SOLUTION:

The C++ code is present in the same zip file as this report.
Instructions to run the code:
run: *g++ CRC.cpp -o CRC*
then run: *./CRC*
Then give the message to be transmitted as input. Next the code will ask for the generating polynomial in bit format.
After this the code outputs the data which is going to be transmitted and will ask for the received data.
Enter the received data to check if it is accepted or rejected by the CRC checker.
Here is the main function of my code:

```
}
int main(){
    string data,polynomial;
    //take the data string as input
    cout << "\nEnter the data to be transmitted: ";
    cin >> data;
    //take the generating polynomial as input
    cout<< "\nEnter the generating polynomial in bit format: ";
    cin >> polynomial;
    while(polynomial[0] == '0'){
        polynomial.erase(polynomial.begin());
        if(polynomial.size() == 0){
            cout<< "\nThe generating polynomial must be non zero\n";
            cout<< "\nExiting the code\n";
            return 0;
        }
    }
    //append n-k zeros at end of the data (where length of gen_poly is n-k+1)
    int n = polynomial.size();
    string data1=data;
    for(int i=0;i<n-1;i++){
        data1.push_back('0');
    }
    cout<< "---------------------------------------\n";
    cout<< "Data padded with n-1 zeros: " << data1 << "\n";
    cout << "---------------------------------------\n";

    string crc_bits = crc(data1,polynomial);

    cout<< "The CRC bits are: " << crc_bits << "\n";
    cout << "---------------------------------------\n";

    data+=crc_bits;

    cout<< "\nThe final data to be sent: "<<data;
    cout<<"\n---------------------------------------";

    string rece;
    cout<<"\nEnter the received data: ";
    cin>>rece;
    if(receive(rece,polynomial))
    cout<<"\nNo Error detected, hence frame accepted\n\n";
    else cout<<"\nError detected, hence frame rejected\n\n";
    return 0;
}
```

The helper functions for CRC calculation and checking received data are:

CRC.cpp > crc(string, string)

```cpp
#include <iostream>
#include <string>

using namespace std;
string crc(string data, string poly){
    int i = 0;
    while(i != data.size() - poly.size() +1){
        if(data[i] == '0'){
            i++;
        }
        else{
            for(int j=0;j<poly.size();j++){
                if(data[i+j] == poly[j])
                data[i+j]='0';
                else
                data[i+j]='1';
            }

        }
    }
    string crc_bits;
    for(int j=i;j<data.size();j++){
        crc_bits.push_back(data[j]);
    }
    return crc_bits;
}
bool receive(string recv,string poly){
    string rem = crc(recv,poly);
    for(int i=0;i<rem.size();i++){
        if(rem[i]=='1')
        return false;
    }
    return true;
}
```

While following is the output:

```
PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

NishantRoshan@MacBook-Air-2 CS425 % g++ CRC.cpp -o CRC
NishantRoshan@MacBook-Air-2 CS425 % ./CRC

Enter the data to be transmitted: 1110010111

Enter the generating polynomial in bit format: 110101
----------------------------------------
Data padded with n-1 zeros: 111001011100000
----------------------------------------
The CRC bits are: 10011
----------------------------------------

The final data to be sent: 111001011110011
----------------------------------------
Enter the received data: 111001011110010

Error detected, hence frame rejected
```

Question 2 [10 points]

In the Go-back-N ARQ mechanism using k-bit sequence numbers, why is the window size limited to $2^k - 1$ and not 2^k ?

SOLUTION:

A k-bit sequence number provides a sequence number range of 2^k

However the maximum window size is limited to $2^k - 1$, and not 2^k

To the contrary, let us assume the case of a 3-bit sequence number and the window size of 2^3, i.e. 8 and not 7.

- Suppose sender sends frame 0 and gets back an RR 1

- It then sends frames 1, 2, 3, 4, 5, 6, 7, 0 and gets another RR 1
- This could mean that all eight frames were received correctly and the RR 1 is a cumulative acknowledgment
- It could also mean that all eight frames were damaged or lost in transit, and the receiving station is repeating its previous RR 1
- The problem is avoided if the maximum window size is limited to 7, i.e. $2^3 - 1$

Thus, we have a $\boxed{\text{max window size of } 2^k - 1}$

Question 3 [10 points]

What is the maximum window size that can be used in the Selective-Reject ARQ mechanism that uses k-bit sequence numbers? Explain your answer.

SOLUTION:
Consider the case of a 3-bit sequence number size for selective-reject:

- Sender sends frames 0 through 6
- Receiver receives all seven frames and cumulatively acknowledges with RR 7
- But, because of a noise burst, the RR 7 is lost
- Sender times out and retransmits frame 0
- Receiver has already advanced its receive window to accept frames 7, 0, 1, 2, 3, 4, and 5
- It assumes that frame 7 has been lost and that this is a new frame 0, which it accepts

The problem is that there is an overlap in between the sending and receiving window.
To overcome the problem, the maximum window size should be no more than half the range of sequence numbers.
Thus, $\boxed{\text{Max window size} = 2^{k-1}\text{, for a k-bit sequence number}}$

Question 4 [10 points]

A channel has a data rate of 4 kbps and a propagation delay of 20 ms. For what range of frame sizes does stop-and-wait give an efficiency of at least 50% ?

SOLUTION:
We know that efficiency of a stop-and-wait channel in flow control(U) is given by:

$$U = \frac{1}{1 + 2a} \tag{1}$$

where a is the ratio of propagation delay and the transmission time
Given: U $\geq \frac{1}{2}$

$\implies$ a $\leq \frac{1}{2}$ (From equation (1))
Now write,

$$a = \frac{PropagationDelay}{TransmissionTime}$$

Given a propagation delay of 20ms, we get a Transmission Time $\geq$ 40ms

$$TransmissionTime = \frac{FrameSize}{DataRate}$$

$\implies$ $\boxed{\text{Frame size} \geq 160 \text{ bits}}$

Question 5 [20 points]

Consider a frame consists of one character of 4 bits. Assume that the probability of bit error is 10^{-3} and that is independent in each bit.

(a) What is the probability that the received frame contains no errors?

(b) What is the probability that the received frame contains at least one error?

(c) Now assume that one parity bit is added. What is the probability that the frame is received with errors that are not detected?

SOLUTION:

Part (a): Consider any one particular bit first.
Probability that this bit contains error (P_b) = 10^{-3} (Given)
Probability that this bit is error free: 1 - P_b
Probability that the whole frame does not contain any error(P_1) = (1 - P_b)^F = (1 - 10^{-3})^4 Here, F is the number of bits in a frame.
$\implies$ $\boxed{P_1 = 0.996}$
Part (b): Probability that received frame contains atleast one error + Probability that it contains no error = 1
Thus, P_2 (that the received frame contains at least one error) = 1 - P_1 Hence P_2 = 1 - (1 - 10^{-3})^4
$\implies$ $\boxed{P_2 = 0.004}$
Part (c): Let E denote the event that the parity bit gets flipped due to some error. We have P(E) = 10^{-3} = 0.001
Now there can be two cases of error occuring:
1) The number of bits flipped is even OR 2) The number of bits flipped is odd.
For case 1), for the error to not be detected, the parity bit must not flip
While for case 2), for the error to not be detected, the parity bit must also be flipped
Case (1): The number of bits flipped is even and parity bit remains unchanged
(a) 2 bit error: Probability = P($\overline{E}$) $\times$ $\binom{4}{2}(1-0.001)^2 0.001^2$
(b) 4 bit error: Probability = P($\overline{E}$) $\times$ 0.001^4
Total probability of the two cases summed up: 0.000005982
Case (2): The number of bits flipped is odd and the parity bit also gets erroneous
(a) 1 bit error: Probability = P(E) $\times$ $\binom{4}{1}(1-0.001)^3 0.001$
(a) 3 bit error: Probability = P(E) $\times$ $\binom{4}{3}(1-0.001)^1 0.001^3$
Total probability of the two cases summed up: 0.000003988

Total probability of error occuring and not getting detected is the sum of the parobability of the two cases

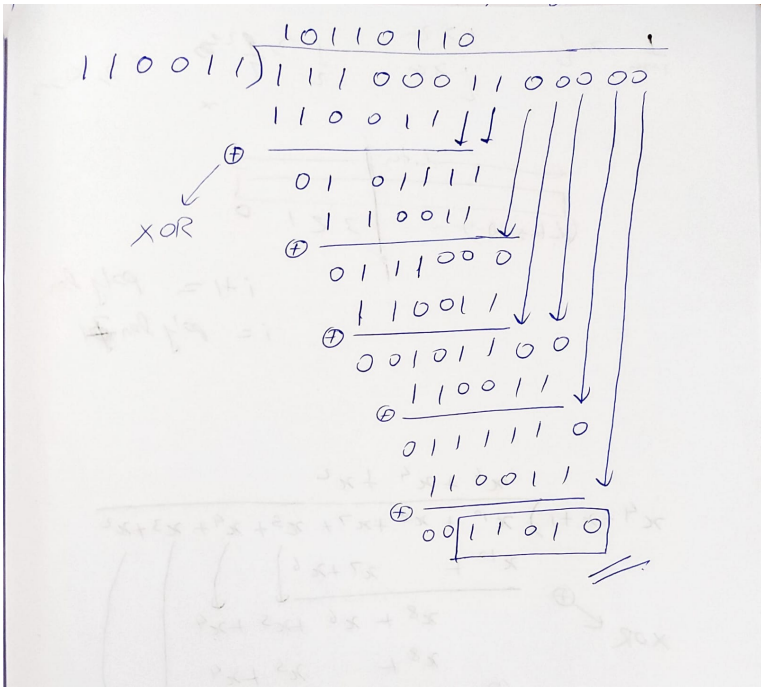
$\implies P_3 = 0.000005982 + 0.000003988 = 0.00000997 = \boxed{9.97 \times 10^{-6}}$

Question 6 [10 points]

For P = 110011 and M = 11100011, find the CRC

SOLUTION:

Given, P = 110011 (Generating polynomial in bit form)
Message data to be sent, M = 11100011
Number of bits in CRC generating polynomial = 6
Thus, we need to append 5 bits at the end of M and then perform the division.
Dividend = 1110001100000
Divisor = 110011
Performing the required division:



Thus we get $\boxed{\text{Remainder = CRC = 11010}}$

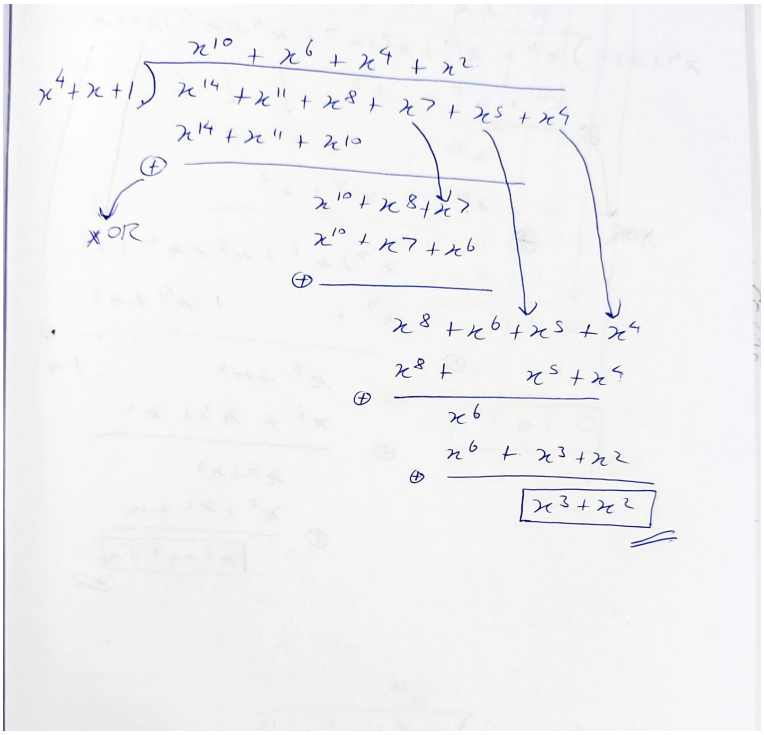
Question 7 [20 points]

(a) In a CRC error-detecting scheme, choose P(x) = X^4 + X + 1. Encode the bits 10010011011.
(b) Suppose the channel introduces an error pattern 100010000000000 (i.e., a flip from 1 to 0 or

from 0 to 1 in position 1 and 5). What is received? Can the error be detected?
(c). Repeat part (b) with error pattern 100110000000000

SOLUTION:

Part (a): Given generator polynomial = $X^4 + X + 1$.
Data to encode(D): 10010011011
Encode this data into polynomial form : D(x) = $(1).X^{10} + (0).X^9 + (0).X^8 + (1).X^7 + (0).X^6 + (0).X^5 + (1).X^4 + (1).X^3 + (0).X^2 + (1).X^1 + (1).X^0$
$\implies$ D(x) = $X^{10} + X^7 + X^4 + X^3 + X^1 + X^0$
Now, we know that T(X) = $X^{n-k}D(X) + R(X)$, where n - k + 1 is equal to the number of bits in (or the length of) the generator polynomial.
Converting the generator polynomial into bit form, we get, P = 10011. Number of bits in P = 5. Hence, $n - k + 1 = 5$. Therefore, $n - k = 4$.
Thus we have the dividend = $X^4(X^{10} + X^7 + X^4 + X^3 + X^1 + X^0) = X^{14} + X^{11} + X^8 + X^7 + X^5 + X^4$
Performing long division:

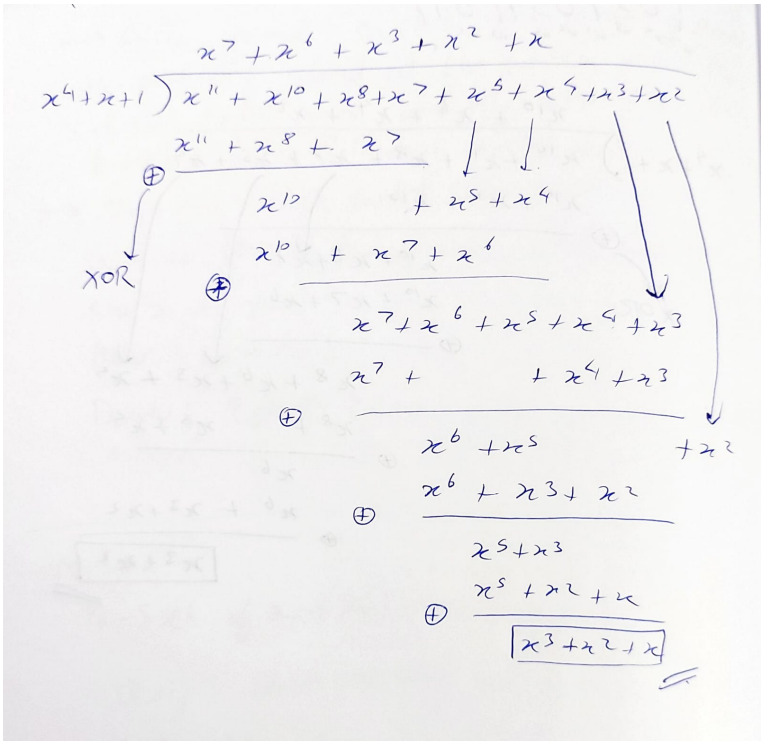


Hence we get, R(X) = $X^3 + X^2 = 1.X^3 + 1.X^2 + 0.X + 0.1$
In bit format, we have R = 1100 (CRC bits)
Thus the encoded data would be, original data appended with these bits.

Encoded data = 100100110111100

Part (b): Error mask : 10001000000000 0 (i.e., a flip from 1 to 0 or from 0 to 1 in position 1

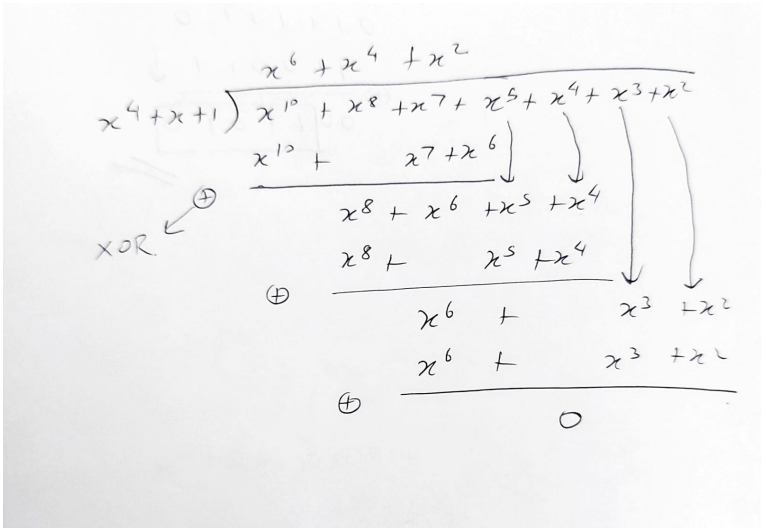
and 5)
Data to be filtered (Answer of question 1): 100100110111100
Filtered data (After flipping bits in position 1 and 5): 000110110111100
Convert this to polynomial form: $D(X) = 0.X^{14} + 0.X^{13} + 0.X^{12} + 1.X^{11} + 1.X^{10} + 0.X^9 + 1.X^8 + 1.X^7 + 0.X^6 + 1.X^5 + 1.X^4 + 1.X^3 + 1.X^2 + 0.X^1 + 0.X^1$
$\implies$ $D(X) = X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^3 + X^2$
Divisor (Generating polynomial): $X^4 + X + 1$
Performing the required division:



Remainder: $X^3 + X^2 + X$

$\implies$ Remainder/CRC bits = 1110 (thus the remainder is non zero)
Hence the errors are detected

Part (c): Given error pattern = 100110000000000 (i.e. the bits at position 1,4 and 5 get reversed)
Thus received data would be: 000010110111100 Converting it into polynomial form, $D(X) = X^{10} + X^8 + X^7 + X^5 + X^4 + X^3 + X^2$
Divisor (in polynomial form) $P(X) = X^4 + X + 1$



Performing the long division, we get: $\boxed{\text{Remainder R(X)} = 0}$. Since this erroneous data is still completely divisible by the divisor, this error remains undetected.

Thank You