Name - Kshitij Chandrakar
SAP - 500124827

14

Page No :

Set-B

Q1 What is purpose of CRC in Error Detection Explain with Ex.

Q2 Write a Program to calculate CRC for the Bitstream 100100 with poly 1011

Q1 => CRC (Cyclic Redundancy check) is a Method Designed to transmit Data over the Internet. It does so by making the BitStream Exactly Divisible by the Generator Polynomial by padding the Remainder of the Division to the Bitstream.

=> the layout is BitStream CRC

=> the length of the CRC is Equal to the Degree of the Polynomial

=> It can only Detect Single Bit Errors.

for Ex. the CRC of 100100 with Generator Poly 1011 is

Generator Poly = $x^3 + x + 1$

∴ length of CRC = 3

Padded Message = 100100 000

Page No:

Binary Division

```
1011 | 100100.000
       1011
       ----
       001000
        1011
        ----
        001100
         1011
         ----
         01110
          1011
          ----
          0101
```

Remainder is 101
∴ CRC is 101

Transmitted Message = 100100.101

Consider this Message is Recieved with an Error (Single Bit)

Let Recieved Message = 100101 101
↑
Erroneous Bit

To check if Error is present
we do xor Division with the Generator polynomial
If the Remainder is Non-zero then the Data is Erroneous.

Page No :

XOR Division :-

```
1011 | 100 10ϕ :101
       101  1
       ----
       0010 0ϕ
            1011
            ----
            00 1010
               1011
               ----
               00 01 1   ← Non zero Remainder
```

∴ the Recieved Bit is Errored.

Code

```
def xor(a,b):
    result = []
    for i in range(len(a)):
        if a[i] == b[i]:
            result.append("0")
        else
            result.append("1")
    return "".join(result)

def div(divd, divs):
    c = len(divs)
    tmp = divd[0:c]

    While (c < len(divd)):
        if temp[0] == '1':
            temp = xor(temp, divs) + divd[
```

```
        Else:
            temp = xor('0'*c, temp) + divd[c:]
        c += 1
    if temp[0] == '1':
        temp = xor(temp, divs)
    Else
        temp = xor('0'*c, divs)

    return temp

def Encoding(data, Poly):

    Padded_Data = data + '0'*(len(Poly)-1)
    Remainder = div(data, Poly)
    Encoded = data + Remainder
    Print("Data is", Data)
    Print("CRC is", Remainder)
    Print("Encoded is", Encoded)

data = "100100"
Poly = "1011"
Encoding(data, Poly)
```