ERROR DETECTION & CORRECTION

- CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel.

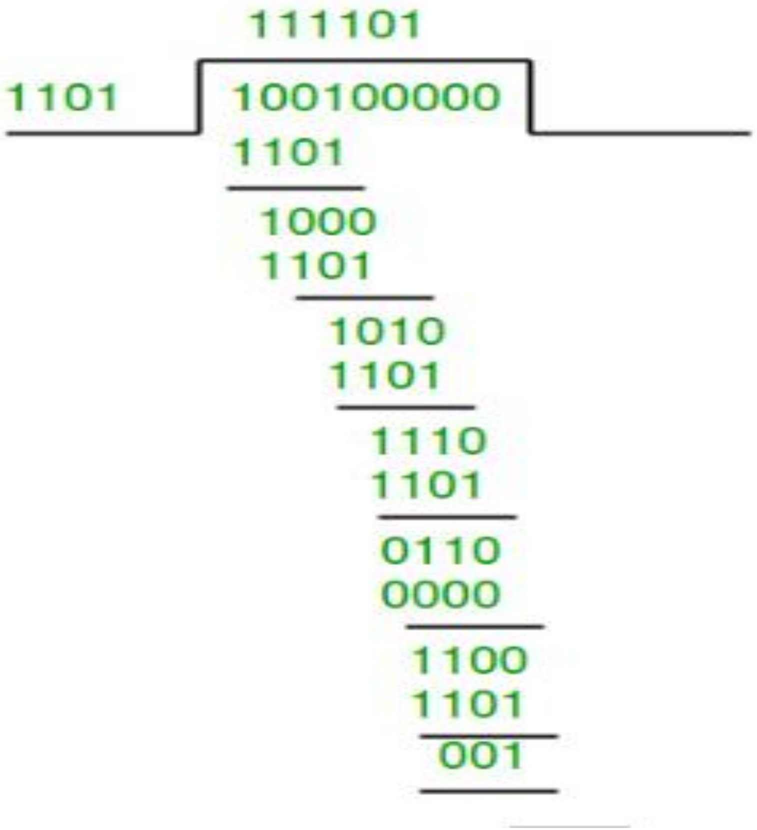
- CRC uses **Generator Polynomial** which is available on both sender and receiver side. An example generator polynomial is of the form like $x^3 + x + 1$.
- This generator polynomial represents key 1011. Another example is $x^2 + 1$ that represents key 101.

n : Number of bits in data to be sent from sender side.

k : Number of bits in the key obtained from generator polynomial.

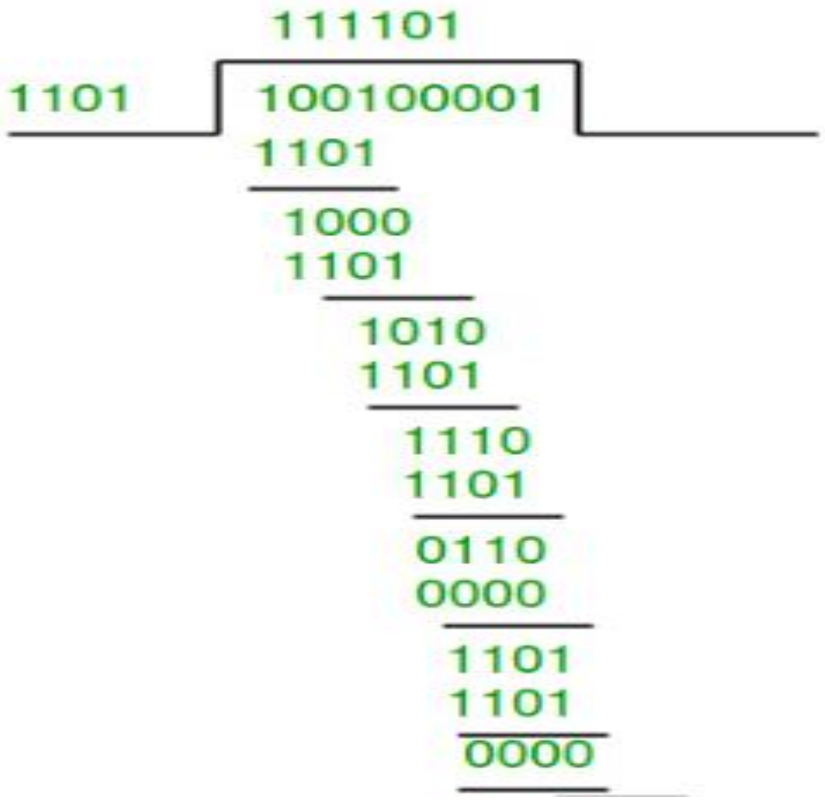
- **Example 1 :**
- Data word to be sent - 100100
- Key - 1101 [Or generator polynomial x3 + x2 + 1]

```
Sender Side:
```



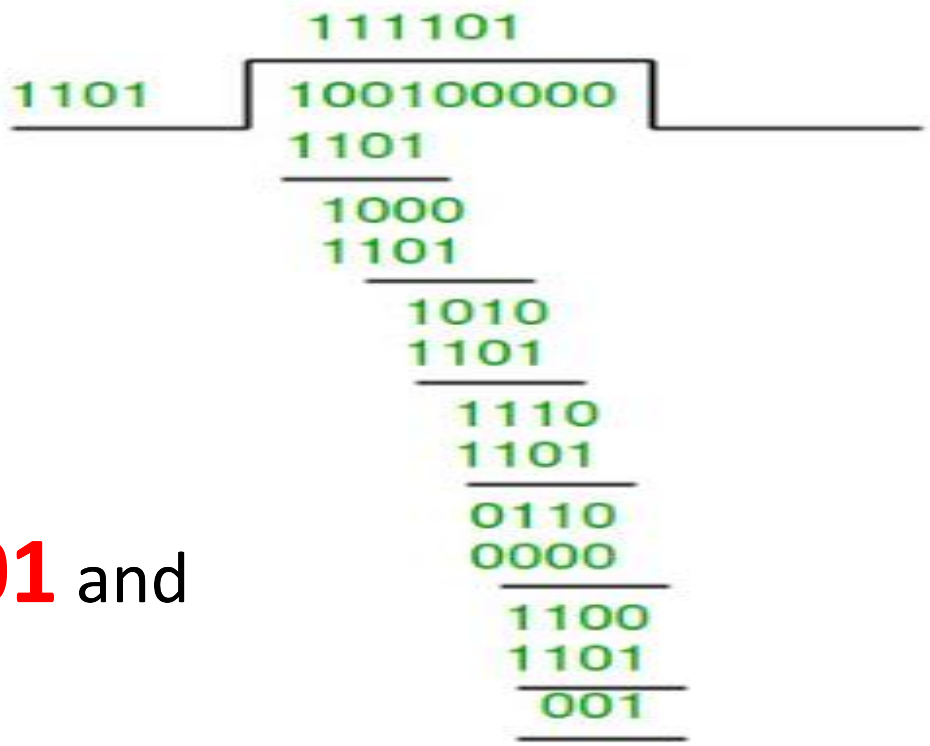
- Therefore, the remainder is 001 and hence the encoded
- data sent is 100100001.

```
Receiver Side:
Code word received at the receiver side
100100001
```



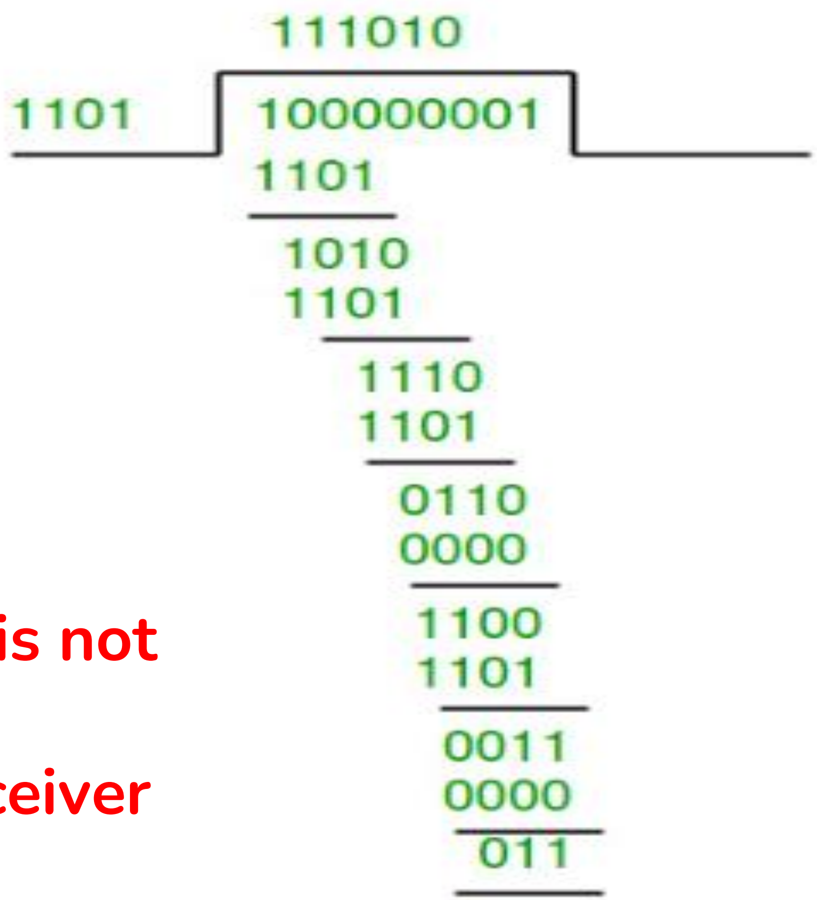
- Data word to be sent - 100100
- Key - 1101

```
Sender Side:
```



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**



Since the remainder is not all zeroes, the error is detected at the receiver side.

- A bit stream 10011101 is transmitted using the standard CRC method. The generator polynomial is x^3+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?

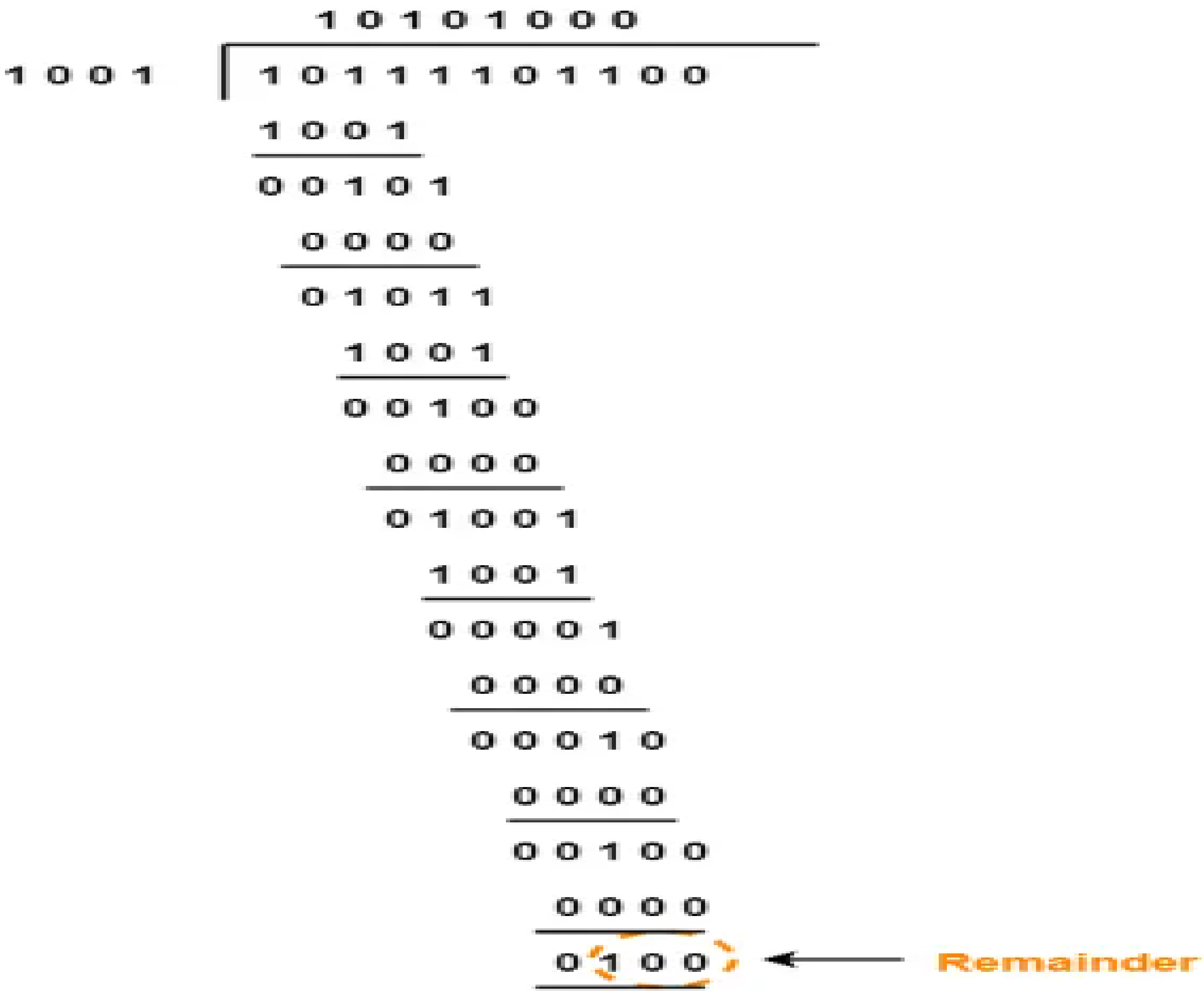
- Solution :
- The generator polynomial $G(x) = x^3 + 1$ is encoded as 1001.
- Clearly, the generator polynomial consists of 4 bits.
- So, a string of 3 zeroes is appended to the bit stream to be transmitted.
- The resulting bit stream is 10011101**000**.

```
             10001100
1001 | 100111101000
       1001
       00001
        0000
        00011
         0000
         00110
          0000
          01101
           1001
           01000
            1001
            00010
             0000
             00100
              0000
              0100   <-- CRC
```

- 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 = 10**1**11101100.

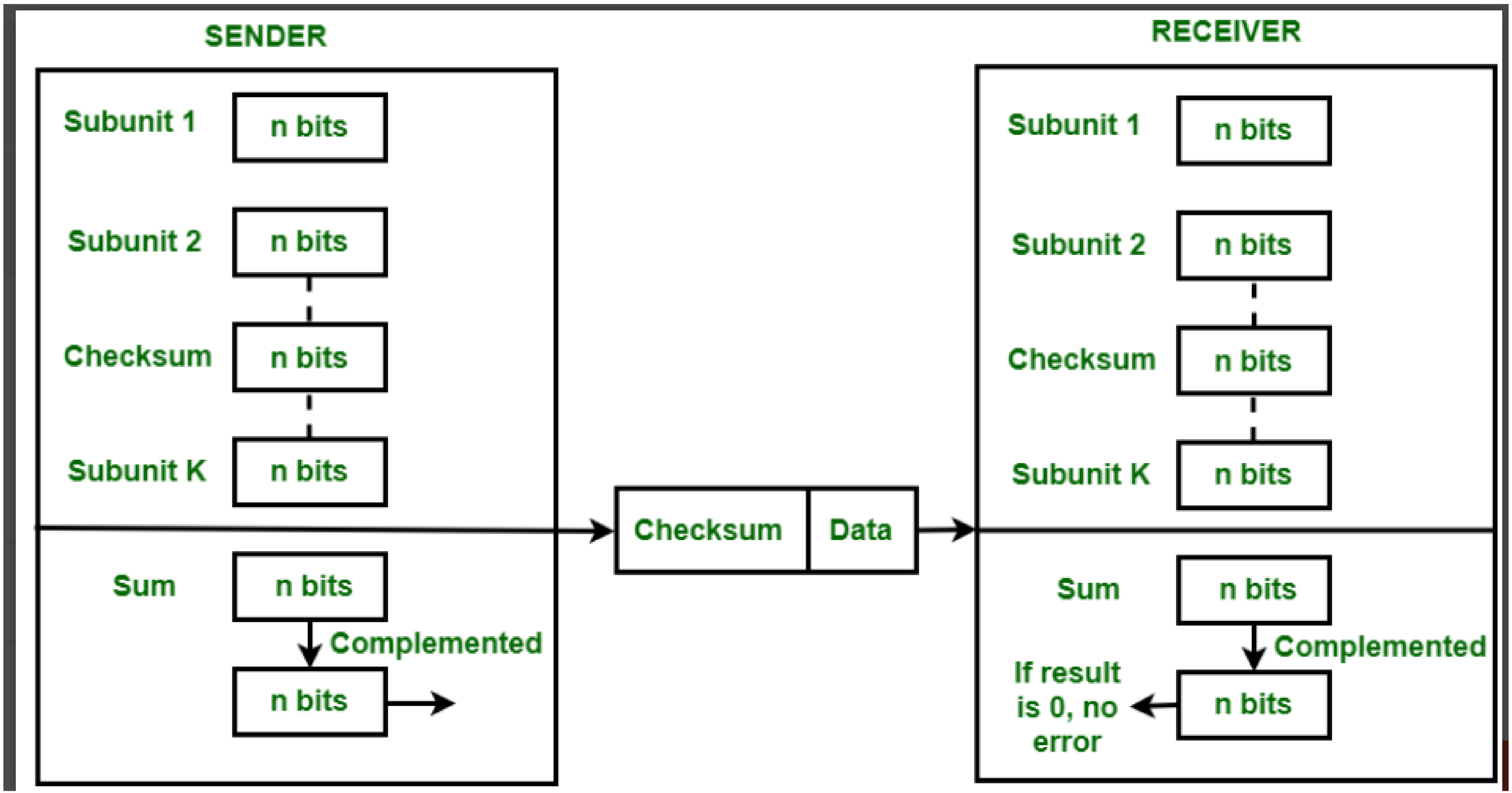
- Receiver receives the bit stream = 10111101100.
- Receiver performs the binary division with the same generator polynomial as-
-



- 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.**

- CheckSum

- **Checksum** is the error detection method used by upper layer protocols and is more reliable than LRC, VRC and CRC. This method makes the use of **Checksum Generator** on Sender side and **Checksum Checker** on Receiver side.
- At the Sender side, the data is divided into equal subunits of n bit length by the checksum generator. This bit is generally of 16-bit length.
- These sub-units are then added together using one’s complement method. This sum is of n bits. **The resultant bit is then complemented. This complemented sum which is called checksum is appended to the end of original data unit** and is then transmitted to Receiver.


SENDER
Subunit 1
n bits
Subunit 2
n bits
Checksum
n bits
Subunit K
n bits
Sum
n bits
Complemented
n bits
Checksum
Data
RECEIVER
Subunit 1
n bits
Subunit 2
n bits
Checksum
n bits
Subunit K
n bits
Sum
n bits
Complemented
If result is 0, no error
n bits

- The Receiver after receiving **data + checksum** passes it to checksum checker.
- Checksum checker divides this data unit into various subunits of equal length and adds all these subunits.
- These subunits also contain checksum as one of the subunits.
- The resultant bit is then complemented. If the complemented result is zero, it means the data is error-free. If the **result is non-zero** it means the **data** contains an **error** and **Receiver rejects** it.

- **Example –**
  If the data unit to be transmitted is
- 10101001 00111001,
- the following procedure is used at Sender site and Receiver site.
- 10101001 **subunit 1**
- 00111001 **subunit 2**
- 11100010 **sum**
- 00011101 **checksum (complement of sum)**

1010001 00111001	00011101
Data	**Checksum**

- **Receiver Site :**

- 10101001 **subunit 1**
- 00111001 **subunit 2**
- 00011101 **checksum**
- 11111111 **sum**
- 00000000 **sum's complement**

- **Result is zero, it means no error**

- Example 2:

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

k=4, m=8

k=4, m=8

Sender

```
1              1 0 0 1 1 0 0 1
2              1 1 1 0 0 0 1 0
             ------------------
             1 0 1 1 1 1 0 1 1
                             1
             ------------------
               0 1 1 1 1 1 0 0
3              0 0 1 0 0 1 0 0
             ------------------
               1 0 1 0 0 0 0 0
4              1 0 0 0 0 1 0 0
             ------------------
             1 0 0 1 0 0 1 0 0
                             1
             ------------------
Sum:           0 0 1 0 0 1 0 1
             ------------------
CheckSum:      1 1 0 1 1 0 1 0
```

Reciever

1	1 0 0 1 1 0 0 1
2	1 1 1 0 0 0 1 0
	1 0 1 1 1 1 0 1 1
	1
	0 1 1 1 1 1 0 0
3	0 0 1 0 0 1 0 0
	1 0 1 0 0 0 0 0
4	1 0 0 0 0 1 0 0
	1 0 0 1 0 0 1 0 0
	1
	0 0 1 0 0 1 0 1
	1 1 0 1 1 0 1 0
Sum:	1 1 1 1 1 1 1 1
Complement:	0 0 0 0 0 0 0 0

Conclusion: Accept Data

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

Sender's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011

Receiver's End

Frame 1:	11001100
Frame 2:	+ 10101010
Partial Sum:	1 01110110
	+ 1
	01110111
Frame 3:	+ 11110000
Partial Sum:	1 01100111
	+ 1
	01101000
Frame 4:	+ 11000011
Partial Sum:	1 00101011
	+ 1
Sum:	00101100
Checksum:	11010011
Sum:	11111111
Complement:	00000000

Hence accept frames.

- Disadvantages :
- The **main problem is that the error goes undetected** if one or more bits of a subunit is damaged and the corresponding bit or bits of a subunit are damaged and the corresponding bit or bits of opposite value in second subunit are also damaged.
- This **is because the sum of those columns remains unchanged.**

- **Example –**
  If the data transmitted along with checksum is 10101001 00111001 00011101.
- But the data received at destination is **0**0101001 **1**0111001 00011101.

- **00101001 1st bit of subunit 1 is damaged**
- **10111001 1st bit of subunit 2 is damaged**
- **00011101 checksum**
- **11111111 sum**
- **00000000 Ok 1's complement**

Explain how collision handled in CSMA/CD? A 2 km long broadcast LAN uses CSMA has 10^7 bps bandwidth and uses CSMA/CD.

The signal travels along the wire at 2 × 10^8 m/s.

What is the minimum packet size that can be used on this network (MU QP)

Solution :

- Distance = 2 km
- Bandwidth = 10^7 bps
- Speed = 2 x 10^8 m/sec

- **Calculating Propagation Delay-**
-
- Propagation delay (T_p)
- Distance / Propagation speed
- = 2 km / (2 x 10^8 m/sec)
- = 2 x 10^3 m / (2 x 10^8 m/sec) = = 10^{-5} sec
- **Calculating Minimum Frame Size-**
- Minimum frame size
- = 2 x Propagation delay x Bandwidth
- = 2 x 10^{-5} sec x 10^7 bits per sec
- = **200 bits or 25 bytes**

Explain how collision handled in CSMA/CD? A 5 km long broadcast LAN uses CSMA has 10^7 bps bandwidth and uses CSMA/CD.

The signal travels along the wire at 5×10^8 m/s.

What is the minimum packet size that can be used on this network (MU QP DEC2023)

Solution : ??

•

A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200 Kbps bandwidth. Find the throughput of the system, if the system (all stations put together) produces 250 frames per second:

Concept:

In slotted aloha, there are time slots for the stations to transmit the data. Stations can transmit data in time slots. Stations have to wait for their time slots to transmit the data.

Formula:

Throughput of slotted aloha = $G \times e^{-G}$

Calculation:

Bandwidth = 200 kbps

It transmits 200 frames using shared channel

Bandwidth = 200 kbps

It transmits 200 frames using shared channel

Frame transmission rate $= \frac{200}{200} = 1\,ms$

System produces 250 frames per second

So, in 1 second frames are = 250

In 1 ms frames are = 250 × 1 ms = (250) × 10^{-3} = ¼

So, G= ¼

$$\mathbf{S} = \frac{1}{4} \times \mathrm{e}^{-\frac{1}{4}} = \mathbf{0.195}$$

Throughput of system = 250 × 0.195 = 49 frames

- THANK YOU

- END OF MODULE 3