CSS2C08

COMPUTER NETWORKS

<u>MODULE 4</u>

ERROR DETECTION AND CORRECTION

1. Error
2. Redundancy
3. Coding
   - ❖ Block coding
4. Modular Arithmetic
5. Error Detection Techniques
   - a) Parity check
   - b) Checksum
   - c) CRC
6. Error Correction Techniques
   - a) Forward Error Correction
   - b) Retransmission
   - c) Hamming code

Error

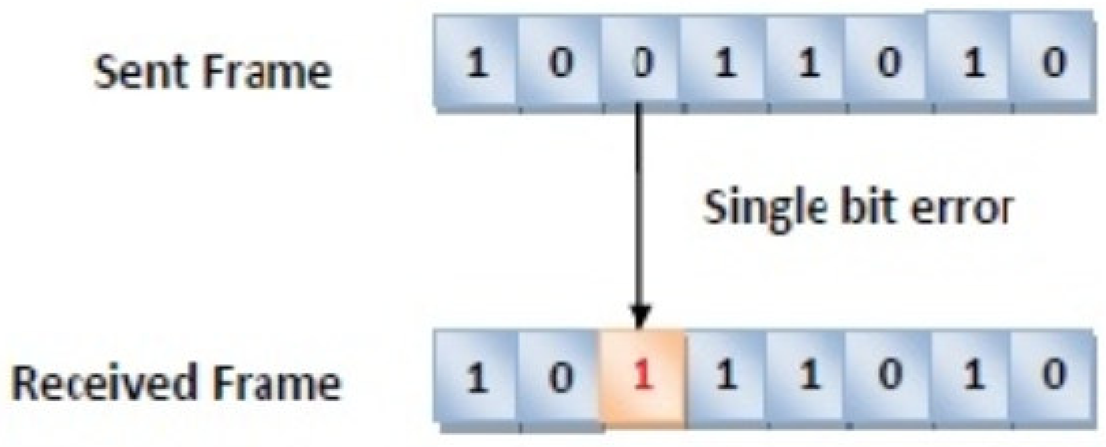
- When bits are transmitted over the computer network, they are subject to get corrupted due to interference and network problems. The corrupted bits leads to spurious data being received by the destination and are called errors.
- **Types of Errors:**

  Errors can be of three types, namely:

  1. Single bit errors
  2. Multiple bit errors, and
  3. Burst errors.

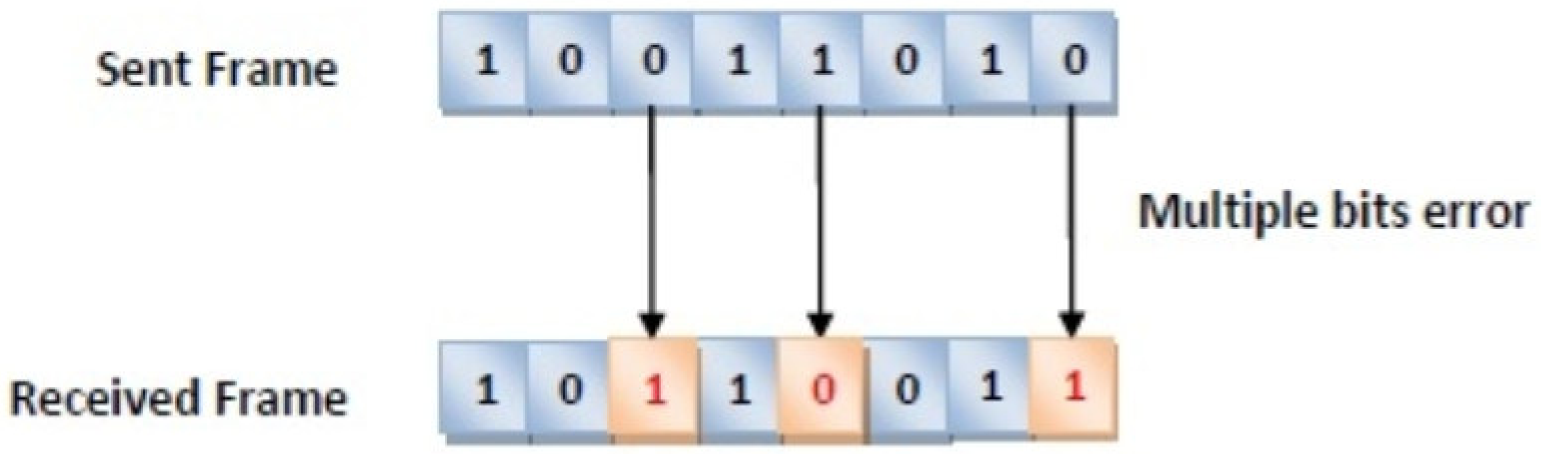
1. **Single bit error**:

   In the received frame, only one bit has been corrupted, i.e. either changed from 0 to 1 or from 1 to 0.



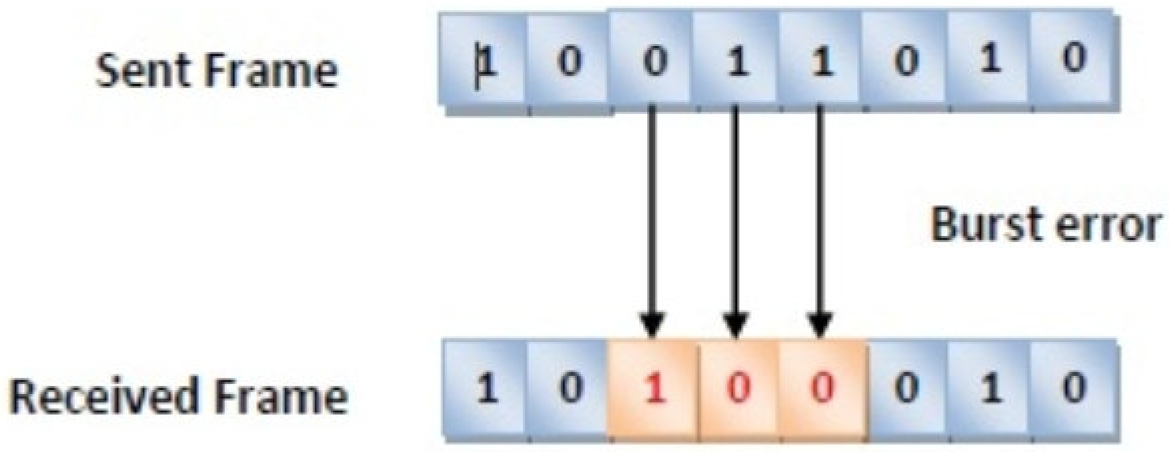
2. **Multiple bits error** :

In the received frame, more than one bits are corrupted.



3. **Burst error**:

In the received frame, more than one consecutive bits are corrupted.



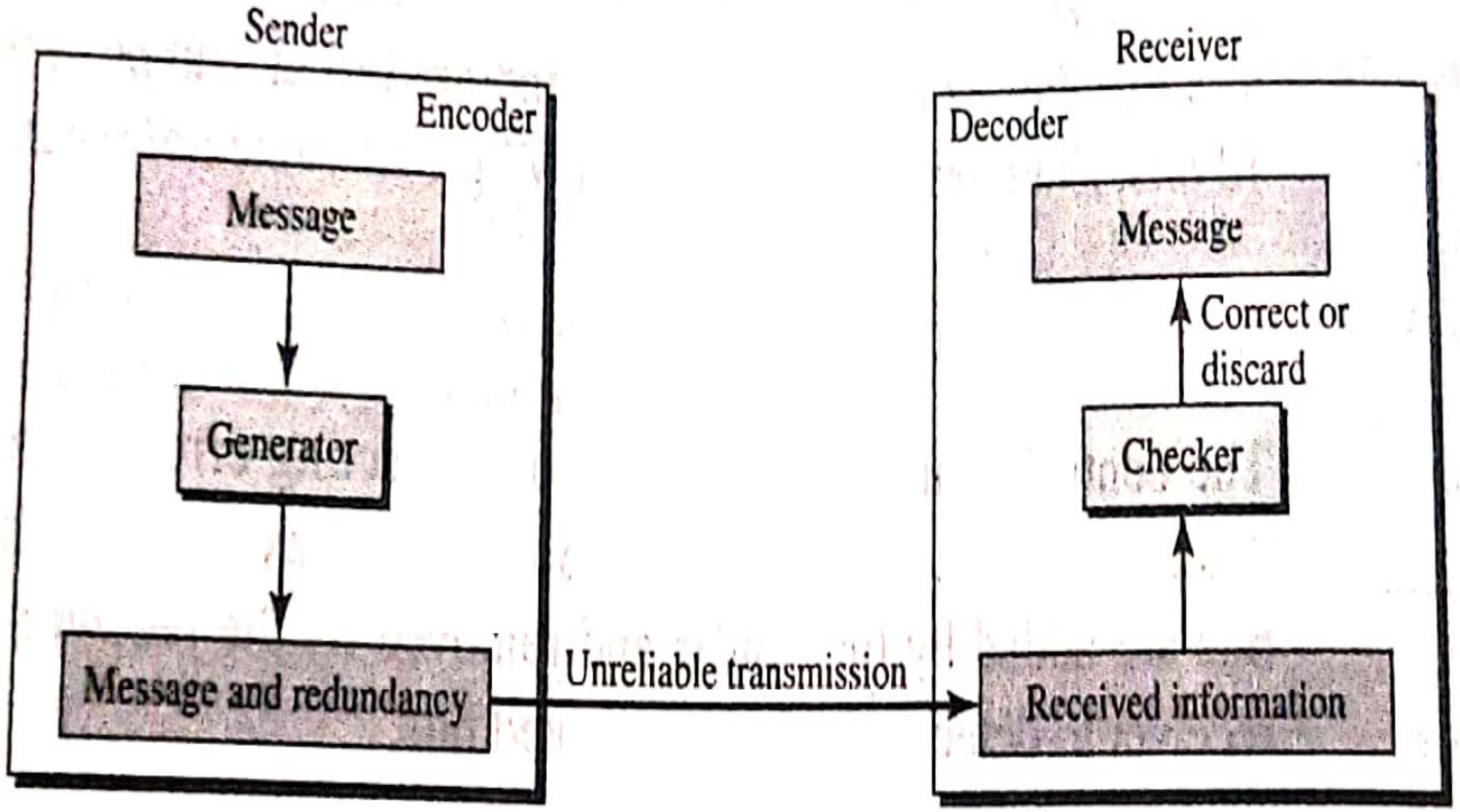
Redundancy

- The central concept in detecting or correcting errors is redundancy.
- To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

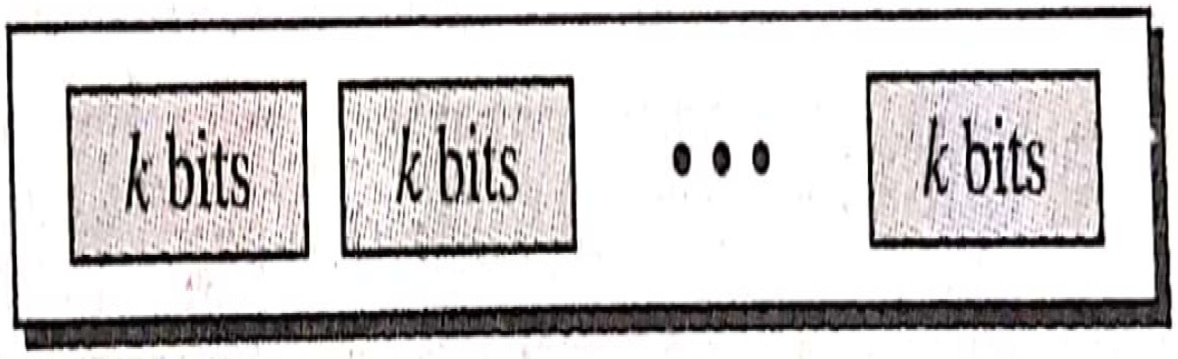
Coding

- Redundancy is achieved through various coding schemes.
- The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits.
- The receiver checks the relationships between the two sets of bits to detect or correct the errors.
- The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme.

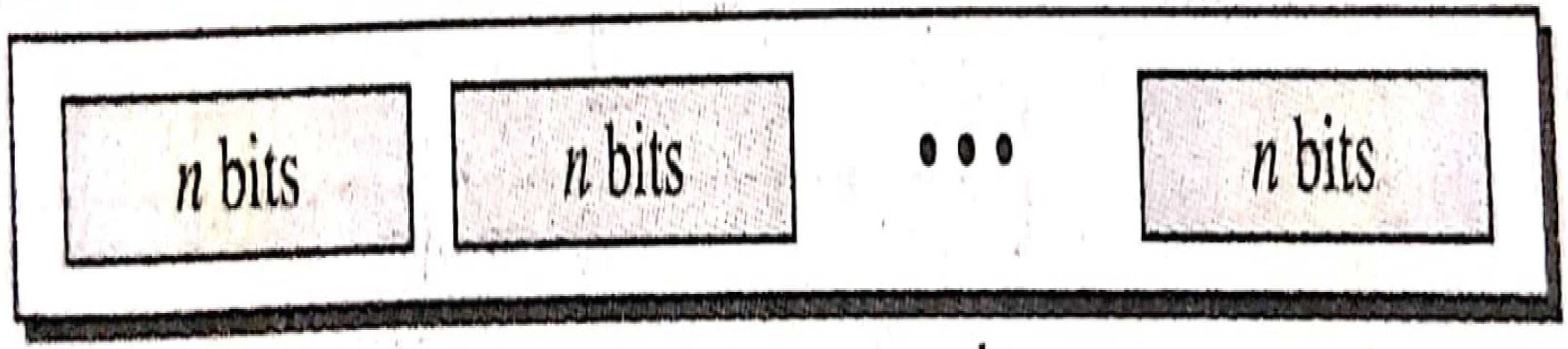
➢ General idea of coding:



- **Coding scheme: block coding**
  - In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length $n = k + r$. The resulting n-bit blocks are called code words.
  - The block coding process is one-to-one; the same data word is always encoded as the same code word.



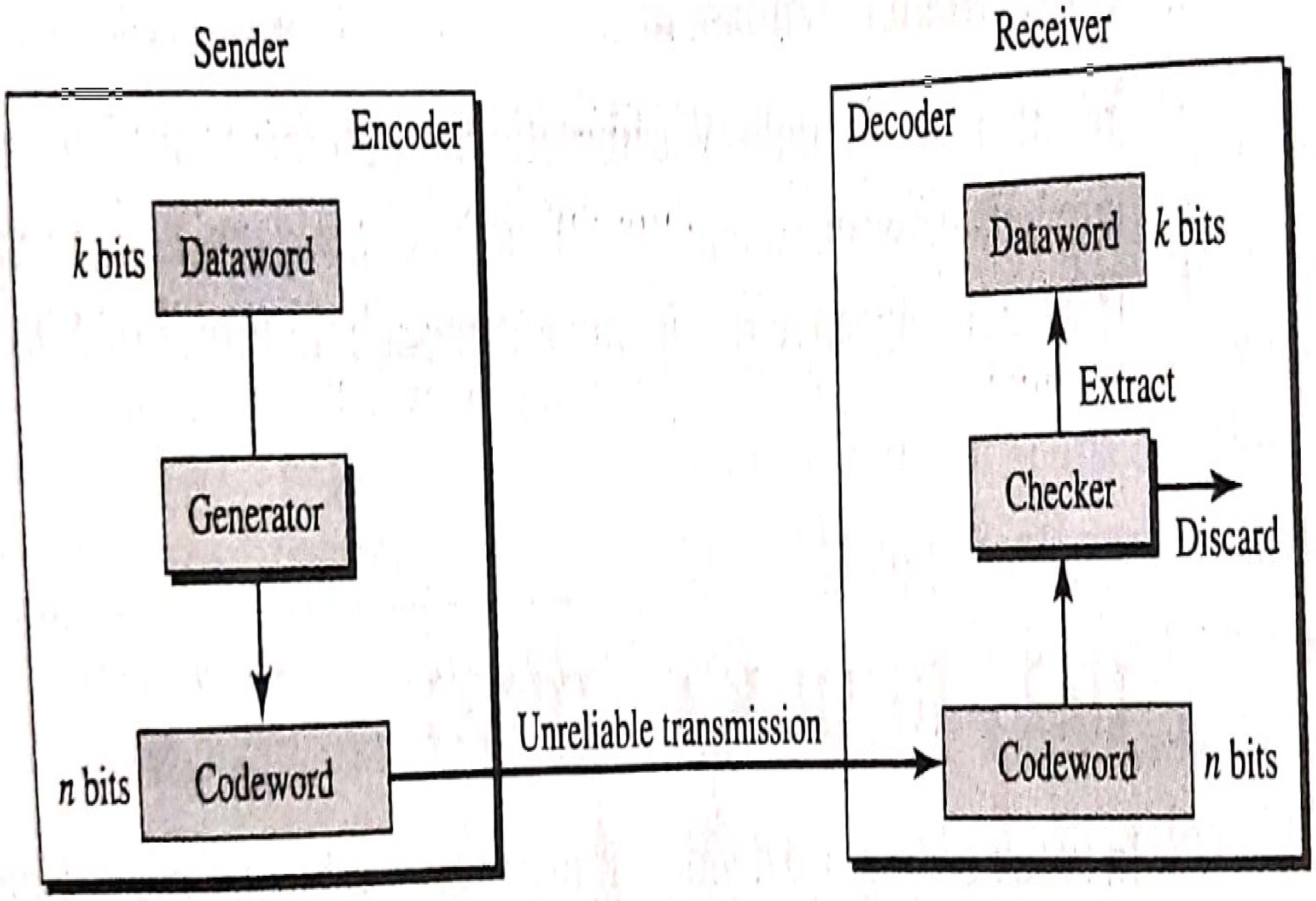
2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

❖ Error Detection:

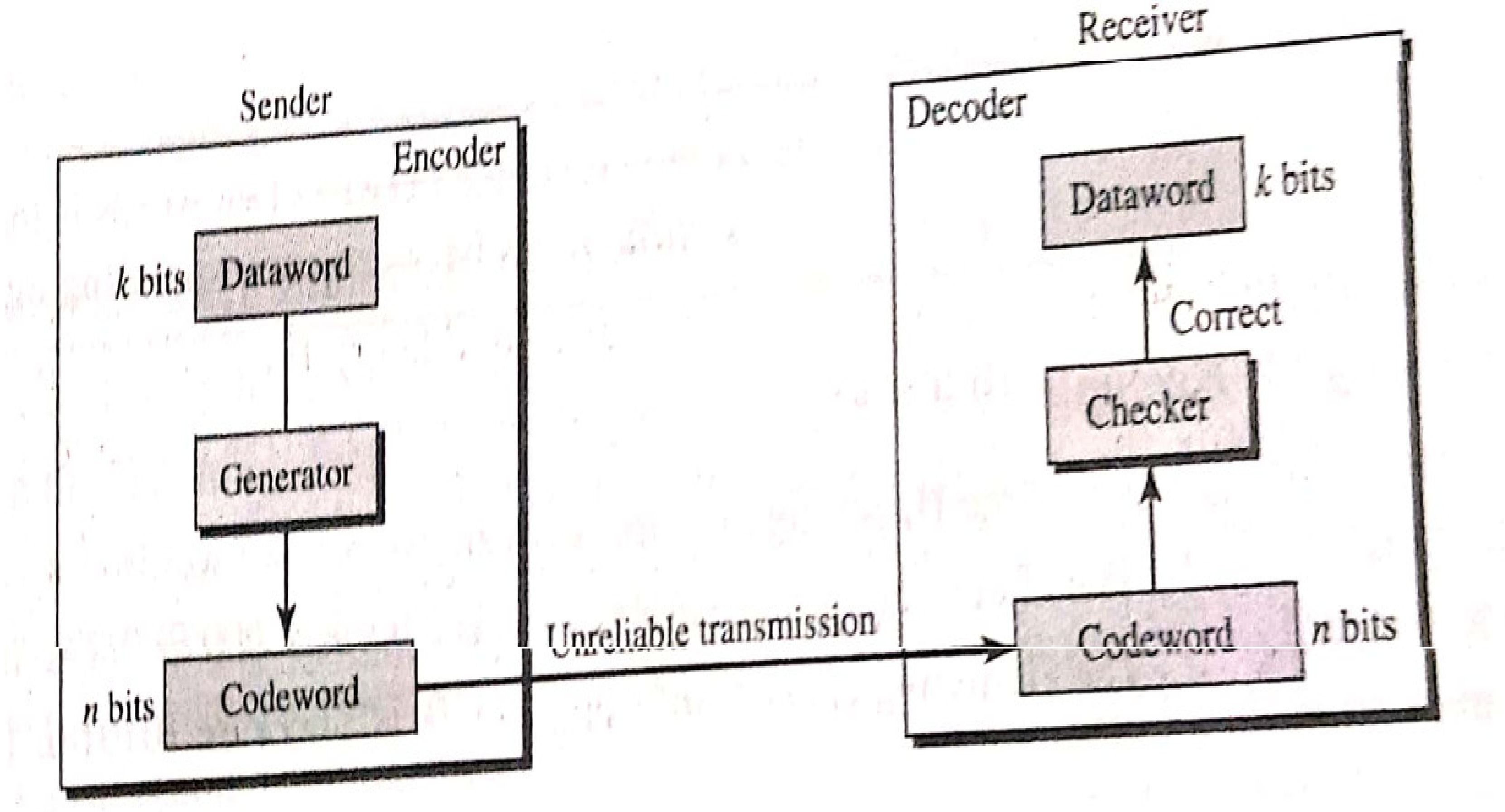
- An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.
- If the following two conditions are met, the receiver can detect a change in the original code word.
  1. The receiver has (or can find) a list of valid code words.
  2. The original code word has changed to an invalid one.



Process of error detection in block coding

❖ Error Correction

- In error detection, the receiver needs to know only that the received code word is invalid; in error correction the receiver needs to find (or guess) the original code word sent. We can say that we need more redundant bits for error correction than for error detection.
- The idea is the same as error detection but the checker functions are much more complex.



Process of error correction in block coding

- **Hamming Distance:**
  - The Hamming distance between two words is the number of differences between corresponding bits.
  - We show the Hamming distance between two words x and y as d(x, y).
  - The Hamming distance can easily be found if we apply the XOR operation on the two words and count the number of 1s in the result.
  - Note that the Hamming distance is a value greater than zero.

- The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.
- We use d_{min} to define the minimum Hamming distance in a coding scheme.
- To find this value, we find the Hamming distances between all words and select the smallest one.
- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min}=s+1$
- To guarantee correction of up to t errors in all cases, the minimum Hamming distance in a block code must be $d_{min}=2t+1$

Modular Arithmetic

- In modular arithmetic, we use only a limited range of integers.
- We define an upper limit, called a modulus N.
- We then use only the integers 0 to N - I, inclusive. This is modulo-N arithmetic.
- For example, if the modulus is 12, we use only the integers 0 to 11, inclusive.
- In a modulo-N system, if a number is greater than N, it is divided by N and the remainder is the result. If it is negative, as many Ns as needed are added to make it positive.

Error Detection Techniques

- There are three main techniques for detecting errors in frames:

1. Parity Check
2. Checksum and
3. Cyclic Redundancy Check (CRC).

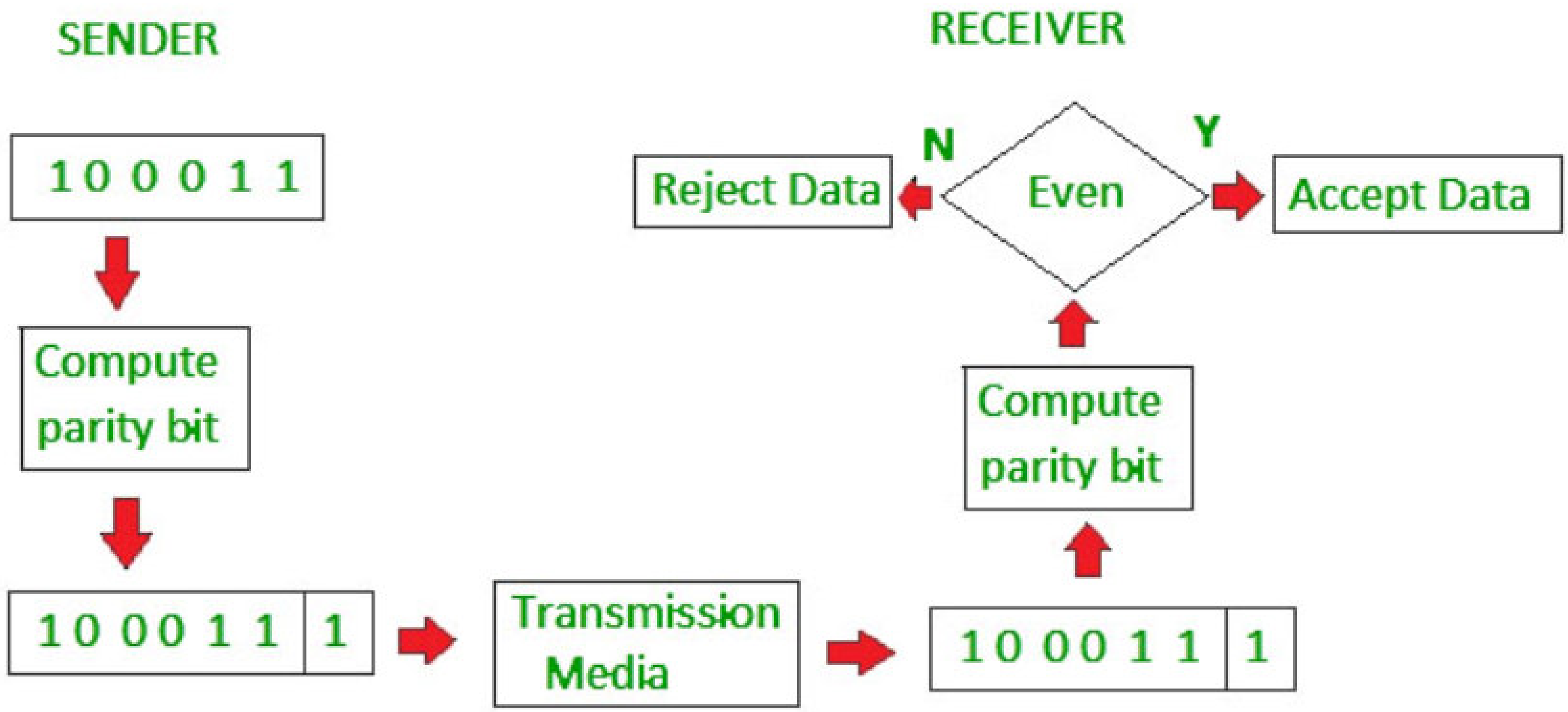
1. Parity Check

- The parity check is done by adding an extra bit, called parity bit to the data to make a number of 1s either even in case of even parity or odd in case of odd parity.
- While creating a frame, the sender counts the number of 1s in it and adds the parity bit in the following way:
  - **In case of even parity:** If a number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1.
  - **In case of odd parity:** If a number of 1s is odd then parity bit value is 0. If a number of 1s is even then parity bit value is 1.

- On receiving a frame, the receiver counts the number of 1s in it.
- In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. A similar rule is adopted for odd parity check.
- The parity check is suitable for single bit error detection only.
- Two categories:
  a) Simple Parity check
  b) Two-dimensional Parity check

a) **Simple Parity check:**

- Blocks of data from the source are subjected to a check bit or parity bit generator form, where a parity of :
  - 1 is added to the block if it contains odd number of 1's
  - 0 is added if it contains even number of 1's
- This scheme makes the total number of 1's even, that is why it is called even parity checking.


SENDER
1 0 0 0 1 1
Compute parity bit
1 0 0 0 1 1 | 1
Transmission Media
RECEIVER
1 0 0 0 1 1 | 1
Compute parity bit
Even
N
Y
Reject Data
Accept Data

b) Two-dimensional Parity check:

- Parity check bits are calculated for each row, which is equivalent to a simple parity check bit.
- Parity check bits are also calculated for all columns, then both are sent along with the data.
- At the receiving end these are compared with the parity bits calculated on the received data.

Original Data

10011001	11100010	00100100	10000100

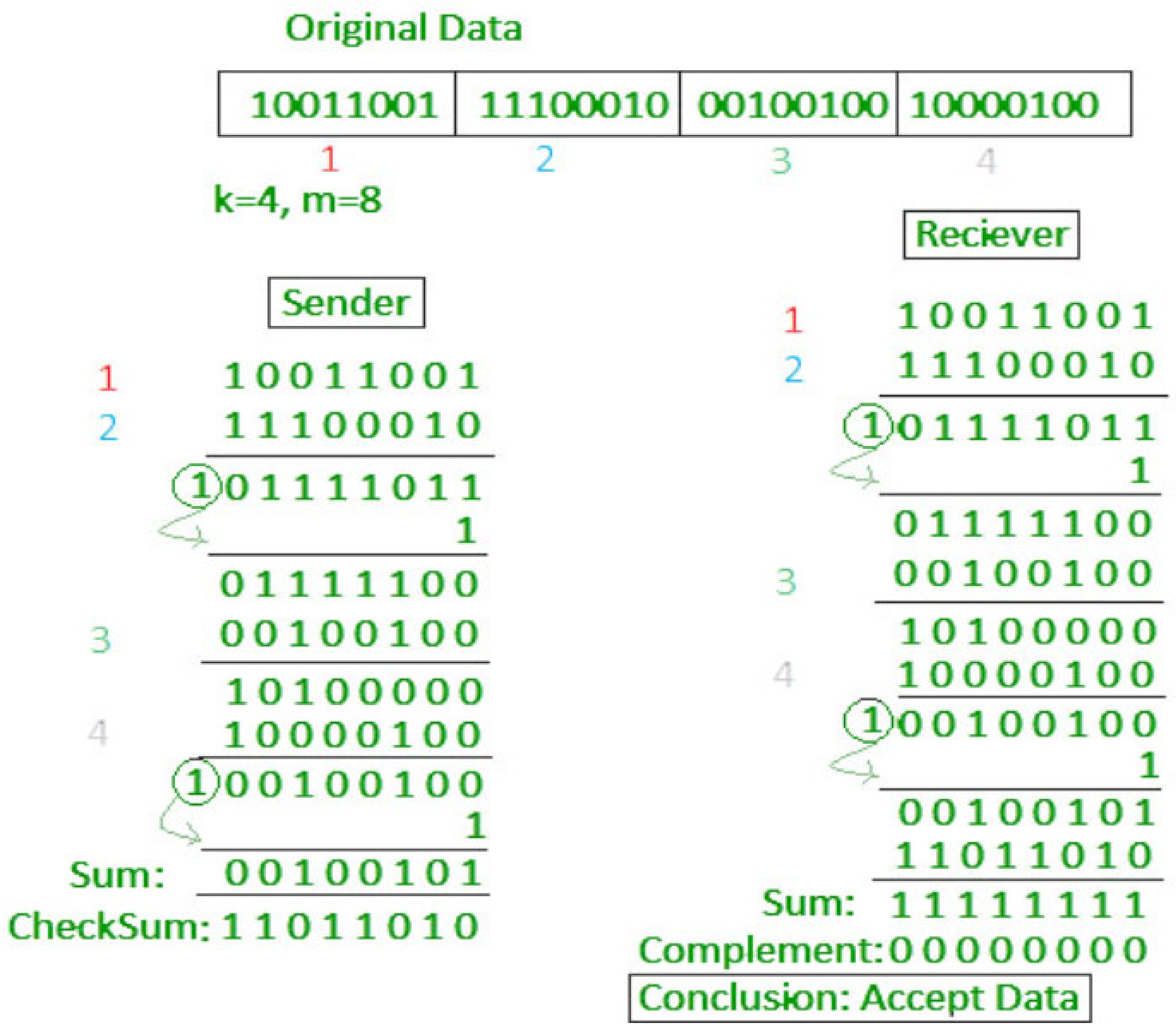
		Row parities
	1 0 0 1 1 0 0 1	0
	1 1 1 0 0 0 1 0	0
	0 0 1 0 0 1 0 0	0
	1 0 0 0 0 1 0 0	0
Column parities	1 1 0 1 1 0 1 1	0

100110010	111000100	001001000	100001000	110110110

Data to be sent

2. Checksum

- In checksum error detection scheme, the data is divided into k segments each of m bits.
- In the sender’s end the segments are added using 1’s complement arithmetic to get the sum. The sum is complemented to get the checksum.
- The checksum segment is sent along with the data segments.
- At the receiver’s end, all received segments are added using 1’s complement arithmetic to get the sum. The sum is complemented.
- If the result is zero, the received data is accepted; otherwise discarded.



Original Data

10011001	11100010	00100100	10000100
1	2	3	4

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
```

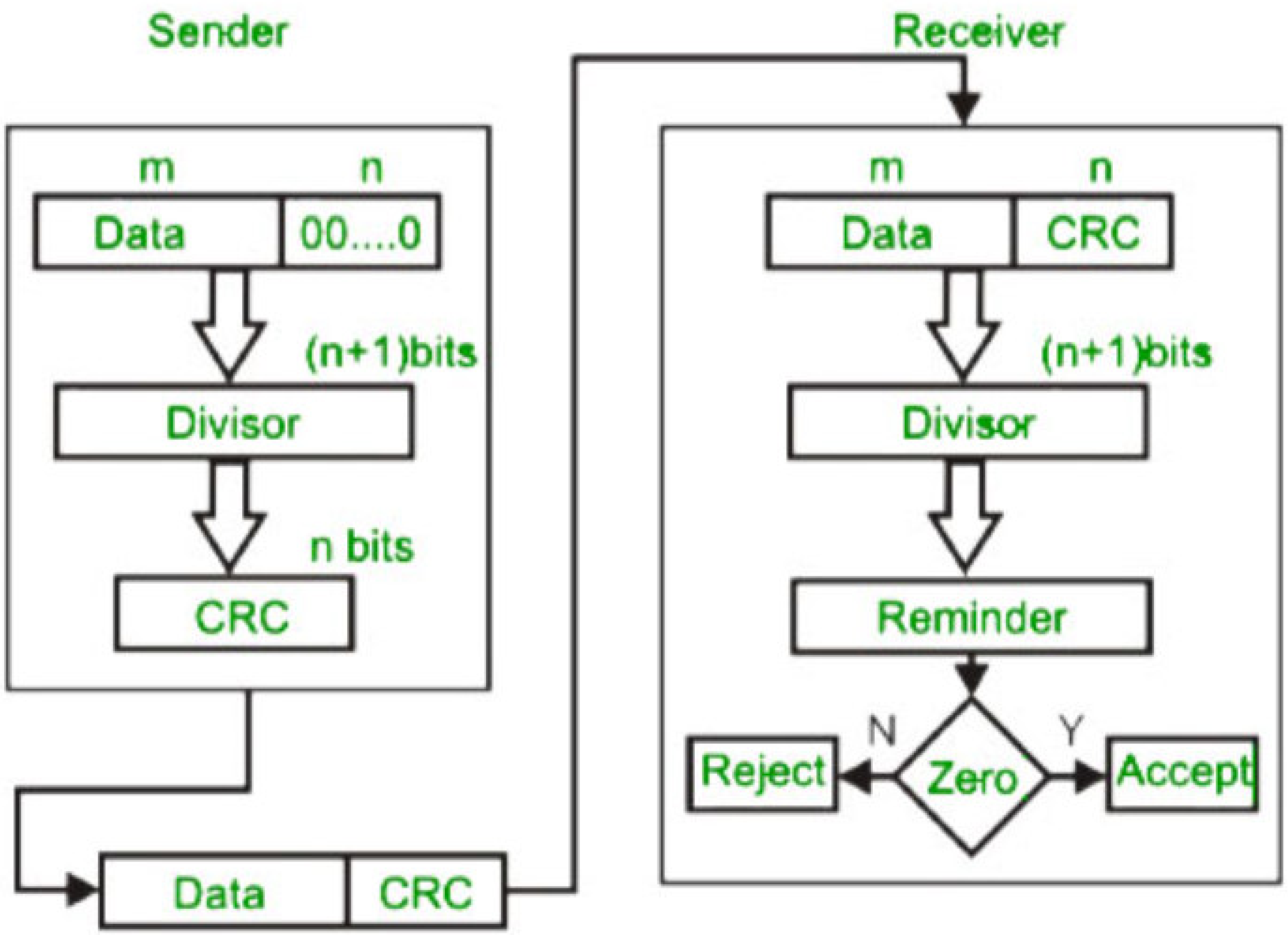
Reciever

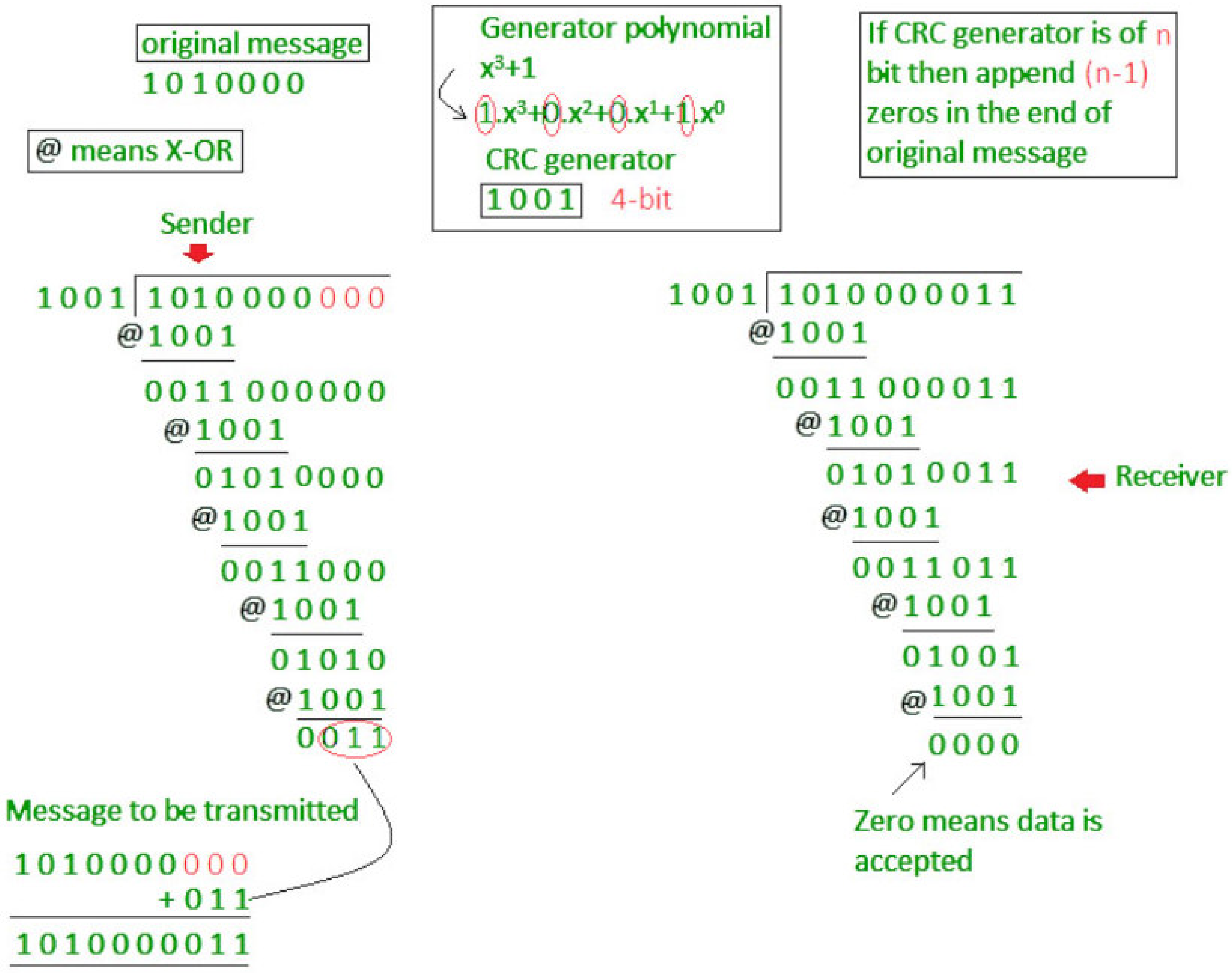
```
1              1 0 0 1 1 0 0 1
2              1 1 1 0 0 0 1 0
             -----------------
             ① 0 1 1 1 1 0 1 1
                             1
             -----------------
               0 1 1 1 1 1 0 0
3              0 0 1 0 0 1 0 0
             -----------------
               1 0 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

3. Cyclic Redundancy Check (CRC)

- CRC is based on binary division.
- In CRC, a sequence of redundant bits, called cyclic redundancy check bits, are appended to the end of data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number.
- At the destination, the incoming data unit is divided by the same number. If at this step there is no remainder, the data unit is assumed to be correct and is therefore accepted.
- A remainder indicates that the data unit has been damaged in transit and therefore must be rejected.


Sender
m
n
Data
00....0
(n+1)bits
Divisor
n bits
CRC
Data
CRC
Receiver
m
n
Data
CRC
(n+1)bits
Divisor
Reminder
N
Y
Reject
Zero
Accept


original message
1 0 1 0 0 0 0
@ means X-OR
Generator polynomial
x³+1
1.x³+0.x²+0.x¹+1.x⁰
CRC generator
1 0 0 1
4-bit
If CRC generator is of n bit then append (n-1) zeros in the end of original message
Sender
1 0 0 1 | 1 0 1 0 0 0 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0 0 0 0
@1 0 0 1
0 1 0 1 0 0 0 0
@1 0 0 1
0 0 1 1 0 0 0
@1 0 0 1
0 1 0 1 0
@1 0 0 1
0 0 1 1
Message to be transmitted
1 0 1 0 0 0 0 0 0 0
+ 0 1 1
1 0 1 0 0 0 0 0 1 1
1 0 0 1 | 1 0 1 0 0 0 0 0 1 1
@1 0 0 1
0 0 1 1 0 0 0 0 1 1
@1 0 0 1
0 1 0 1 0 0 1 1
@1 0 0 1
0 0 1 1 0 1 1
@1 0 0 1
0 1 0 0 1
@1 0 0 1
0 0 0 0
Receiver
Zero means data is accepted

Error Correction Techniques

- There are two main methods of error correction:

  a) **Forward error correction** is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small.

  b) **Correction by retransmission** is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free .

- The main error correction code is Hamming code.
- **Hamming code:**
  - It is used for detection and correction of error in digital transmission.
  - **Determine Single error correcting code:**
    - Step 1:Find the number of parity bit required.

$$2^P \geq m+p+1$$

where

m=no. of information bits

p=no. of parity bits.

- Step2 :Construct a bit position table and enter the information bits. The parity bits are located in the positions that are numbers corresponding to ascending powers of two(1,2,4,8,….)

  eg:P1,P2,M1,P3,M2,M3,M4

Bit designation	P1	P2	M1	P3	M2	M3	M4
Bit position							
Binary position no.							
Information bits							

- Step 3:Determine the parity bits as follows:

  P1 checks bit position:

  …………………………….
- Step 4:These parity bits are entered in the table

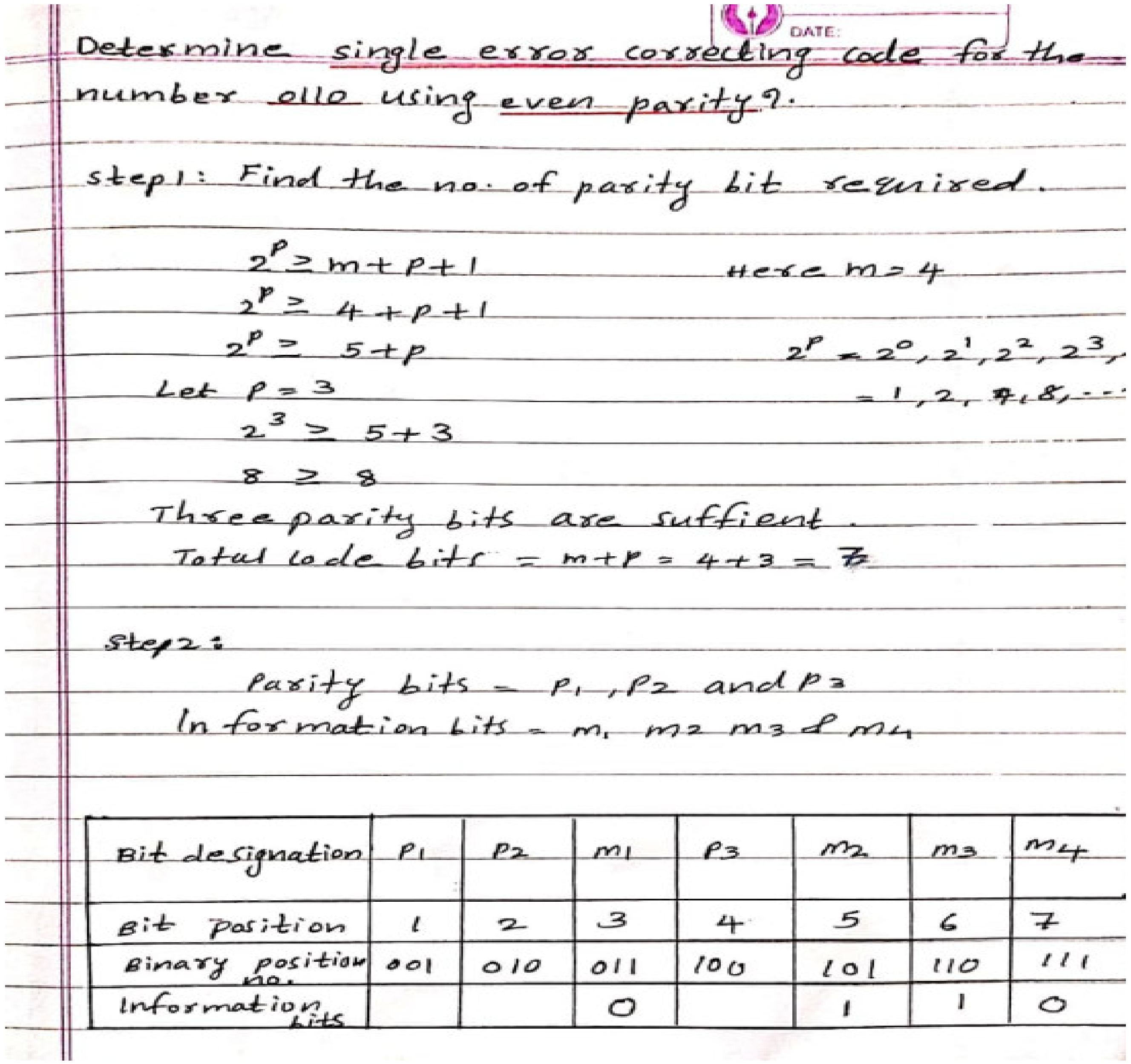
Bit designation	P1	P2	M1	P3	M2	M3	M4
Bit position							
Binary position no.							
Information bits							
Parity bits							

Resulting code=P1P2M1P3M2M3M4

❖Note:

- Using even parity
  - Even no. of 1’s=0
  - Odd no. of 1’s=1
- Using odd parity
  - Even no. of 1’s=1
  - Odd no. of 1’s=0

Example



Determine single error correcting code for the number 0110 using even parity?.

step1: Find the no. of parity bit required.

$2^p \geq m+p+1$ Here $m=4$

$2^p \geq 4+p+1$

$2^p \geq 5+p$ $2^p = 2^0, 2^1, 2^2, 2^3,$

Let $p=3$ $= 1, 2, 4, 8, \ldots$

$2^3 \geq 5+3$

$8 \geq 8$

Three parity bits are suffient.

Total code bits $= m+p = 4+3 = 7$

step2:

Parity bits = p_1, p_2 and p_3

Information bits = m_1 m_2 m_3 & m_4

Bit designation	p_1	p_2	m_1	p_3	m_2	m_3	m_4
Bit position	1	2	3	4	5	6	7
Binary position no.	001	010	011	100	101	110	111
Information bits			0		1	1	0

PAGE NO.
DATE

step 3:

Determine the parity bit as follows:

P_1 checks bit position : 1, 3, 5, 7 ⇒ 1
P_2 checks bit position : 2, 3, 6, 7 ⇒ 1
P_3 checks bit position : 4, 5, 6, 7 ⇒ 0

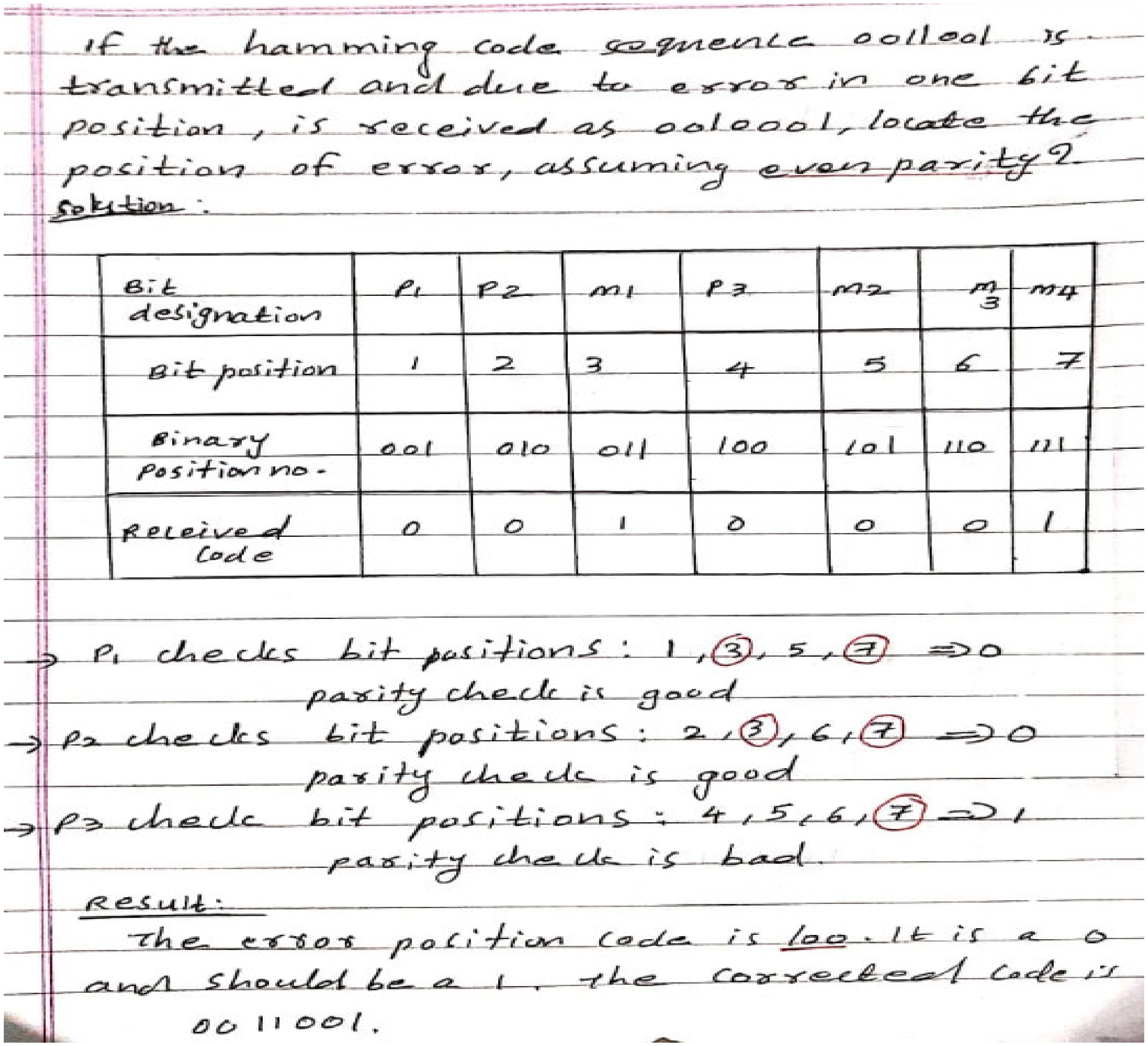
step 4: These parity bits are entered in the table.

Bit designation	P_1	P_2	m_1	P_3	m_2	M_3	M_4
Bit position	1	2	3	4	5	6	7
Binary position no.	001	010	011	100	101	110	111
Information bit			0		1	1	0
Parity bit	1	1		0			

Resulting combining code = <u>1100110</u>

- **Detecting and correcting Error:**
  - **Step1:**Start with the group checked by P1.
  - **Step2**:Check the group for proper parity.
    - 0 represent a good parity check
    - 1 represent a bad parity check
  - **Step 3:**Repeat step 2 for each parity group.
  - **Step 4:**The binary number formed by the results of all the parity checks designates the position of the code bit that is an error. This is the error position code. The first parity check generates the LSB.If all checks are good, there is no error.

Example:



If the hamming code sequence 0011001 is transmitted and due to error in one bit position, is received as 0010001, locate the position of error, assuming <u>even parity</u>?

<u>Solution</u>:

Bit designation	P_1	P_2	m_1	P_3	m_2	m_3	m_4
Bit position	1	2	3	4	5	6	7
Binary Position no.	001	010	011	100	101	110	111
Received Code	0	0	1	0	0	0	1

→ P_1 checks bit positions: 1, ③, 5, ⑦ ⇒ 0
parity check is good

→ P_2 checks bit positions: 2, ③, 6, ⑦ ⇒ 0
parity check is good

→ P_3 check bit positions: 4, 5, 6, ⑦ ⇒ 1
parity check is bad.

<u>Result</u>:

The error position code is <u>100</u>. It is a 0 and should be a 1. The corrected code is 0011001.