Error Correction & Detection

→ Data can be corrupted during transmission For reliable communication errors must be detected and corrected.

→ Error Correction & Detection are either implemented at data link layer or transport layer of OSI model.

Types of Error

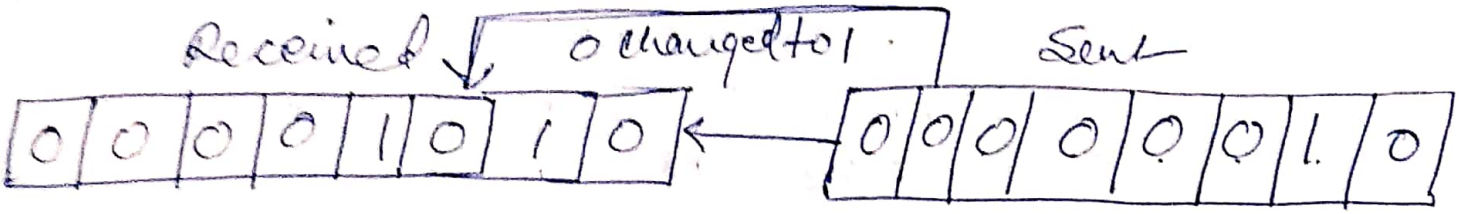
Single bit Error

→ only 1 bit in the data unit is changed.

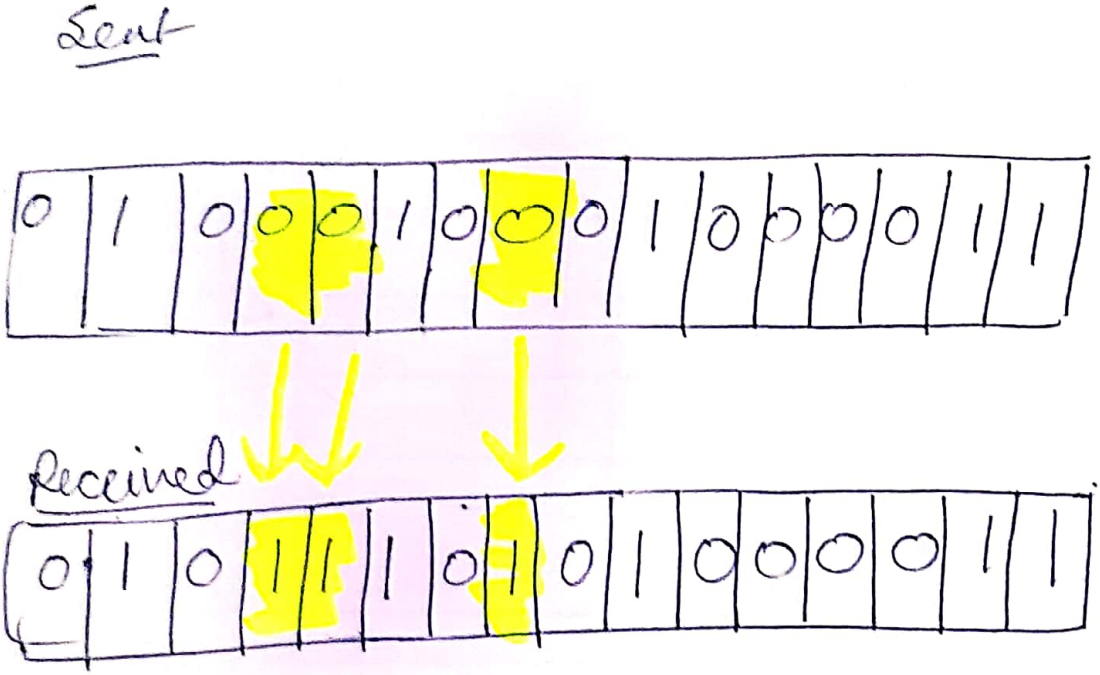
Burst Error

→ two or more bits in the data unit has changed.

Single bit Error

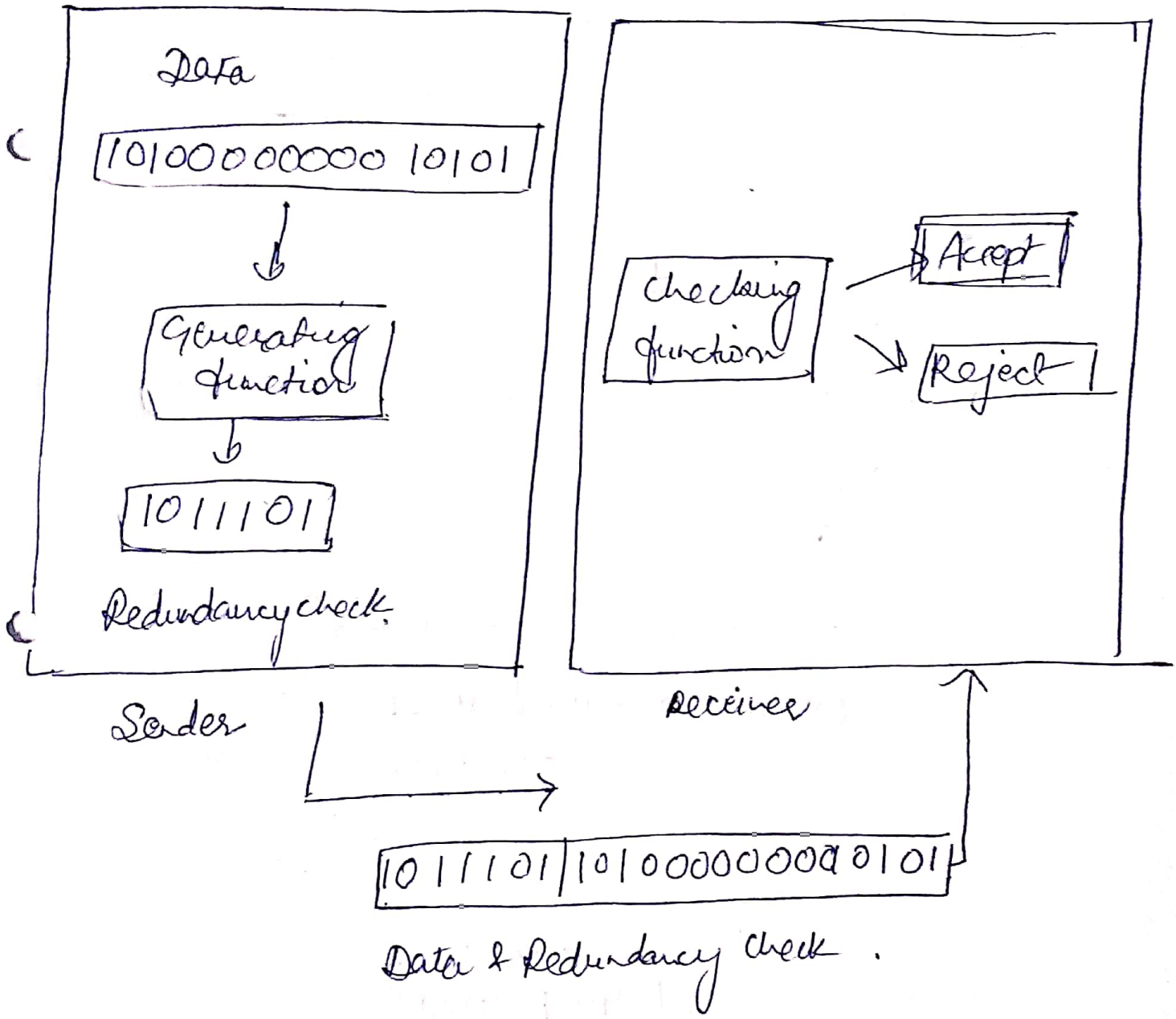


Burst Error



Error Detection

→ Error Detection uses the concept of redundancy, which means adding extra bits for for detecting error at destination



There are basically four types of redundancy checks. They are:-

1. VRC (Vertical Redundancy Check)
2. LRC (Longitudinal Redundancy Check)
3. CRC (Cyclical Redundancy check)

Vertical Redundancy check (VRC)

→ It is also known as parity check.

→ It is least expensive mechanism for error detection.

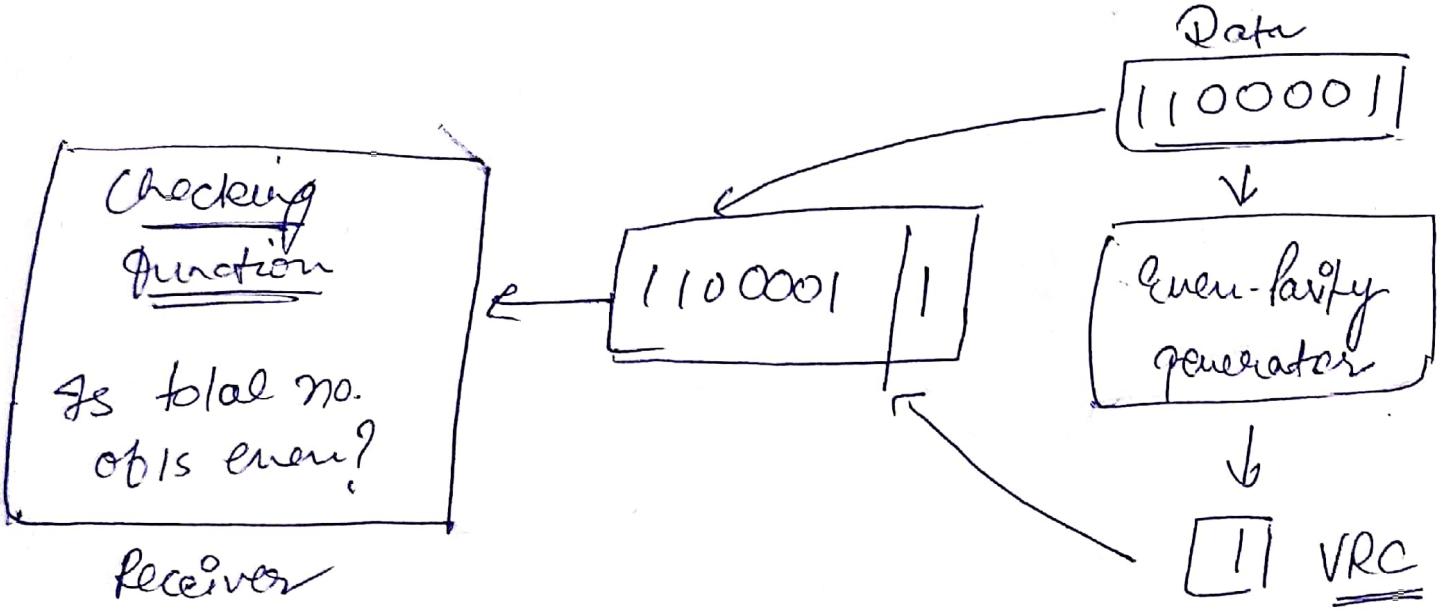
→ In this technique the redundant bit also called as parity bit is appended to every data unit so that total no. of 1's in the unit becomes 1. (including Parity bit)

Eg.

1110110 1101111 1110010

- After adding parity bit.

11101101 11011110 11100100



Advantages

1. VRC can detect all single bit errors.
2. It can detect burst error if total no. of errors in each data unit is odd

Eg: original data 1100111

Sent → 11001111

→ Received → 10111111

Odd bits are changed

Odd no of bits changed so total 1's are now (7) which is odd no.

So Receiver will detect that some bits are changed or there is error.

original Data → 11001111

→ Received → 11101111.

Even bits are changed

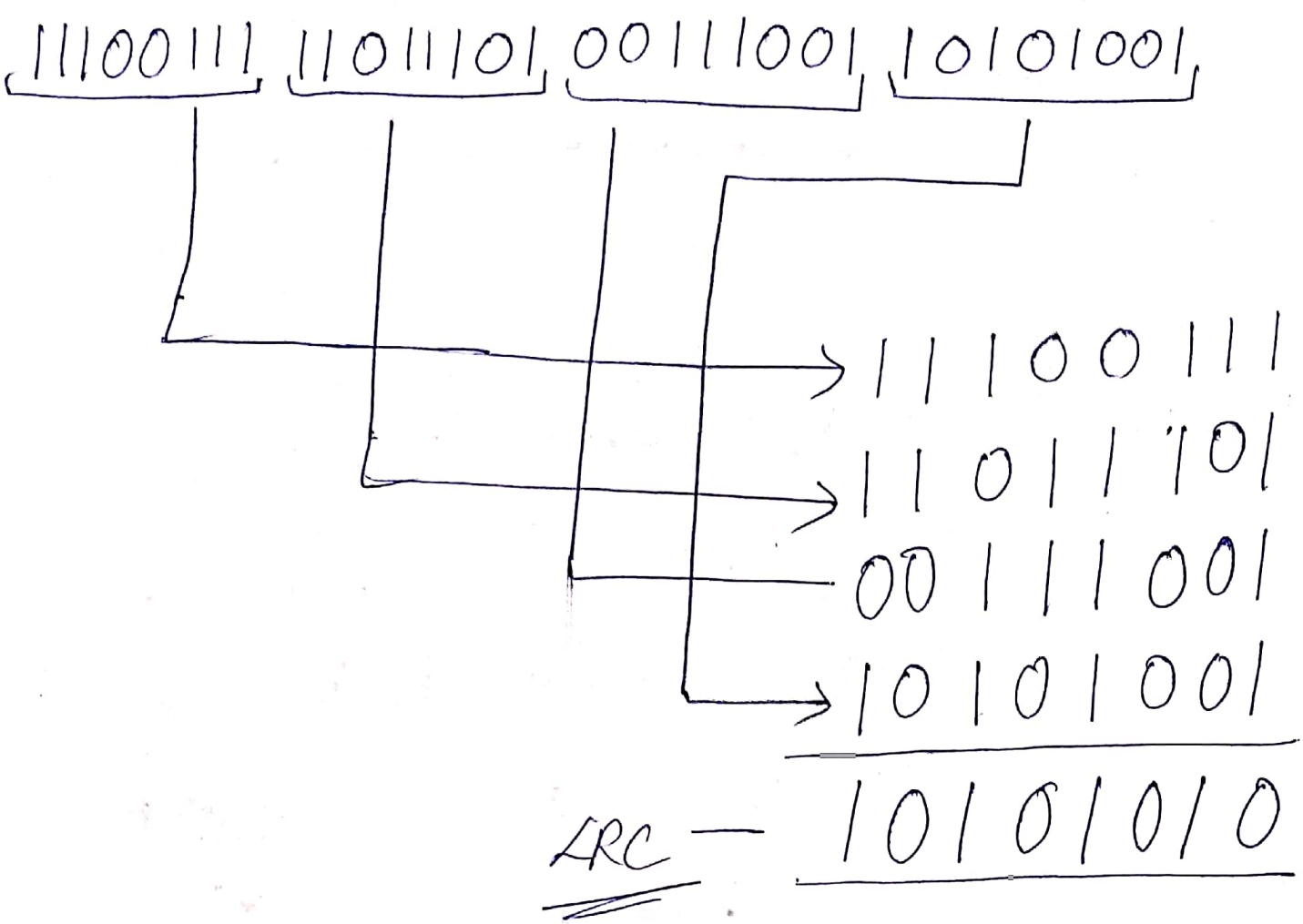
even no. of bits are changed so total 1's are (6) which is even no.

So receiver will not identify that there is an error.

Disadvantage :- VRC can not detect errors where total no. of bits changed is even.

Longitudinal Redundancy check → (LRC)

In this method a block of bits is organized in tables (rows and columns), Calculate the parity bit for each column. and the set of this parity bit is also sent with original data.



Sent →

11100111 11011101 00111001 10101001 10101010

Original data plus LRC.

Advantages -

An LRC of n bits can easily detect burst error of n bits.

10100111 00111001 11011101 11100111 10101010
LRC

10100111 10001001 11011101 11100111 10101010

↓

```
 1 0 1 0 0 0 1 1
 1 0 0 0 1 0 0 1
 1 1 0 1 1 1 0 1
 1 1 1 0 0 1 1 1
 1 0 1 0 1 0 1 0
-----------------
 1 0 1 1 1 0 1 0   → 5 no of 1's
```

Disadvantage : If two bits in one data unit are damaged and two bits in exactly same position in another data unit are damaged, then LRC checker will not detect the errors.

Eg Original data →

10101010, 10100100, 00111001, 11011101, 11100111

LRC

Data Received →

10101010, 10100011, 00110011, 11011101, 11100111

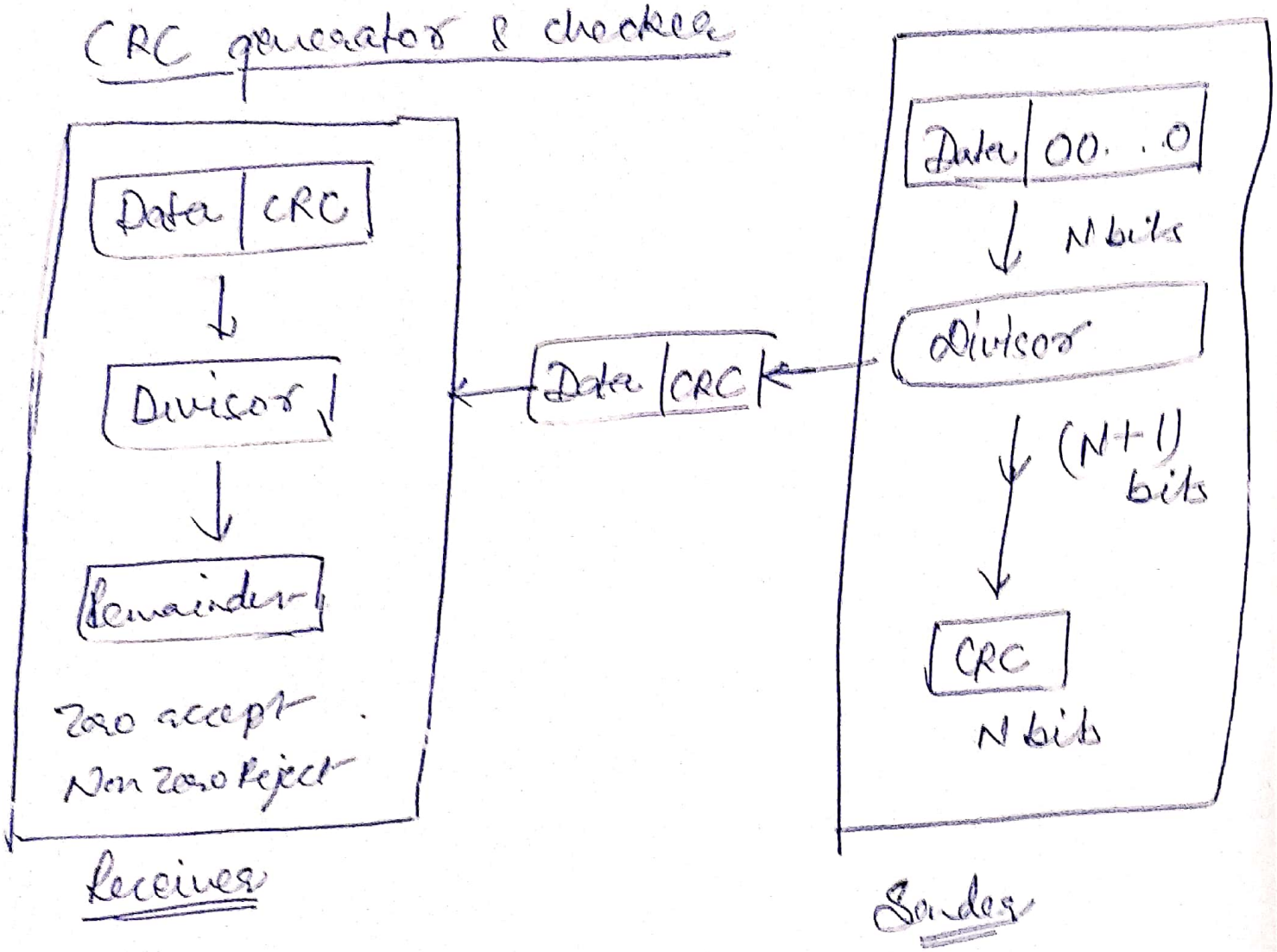
LRC

```
 1 0 1 0 0 0 1 1
 0 0 1 1 0 0 1 1
 1 1 0 1 1 1 0 1
 1 1 1 0 0 1 1 1
---------------
 1 0 1 0 1 0 1 0
```

→ Same LRC as original

Cyclic Redundancy Checks (CRC)

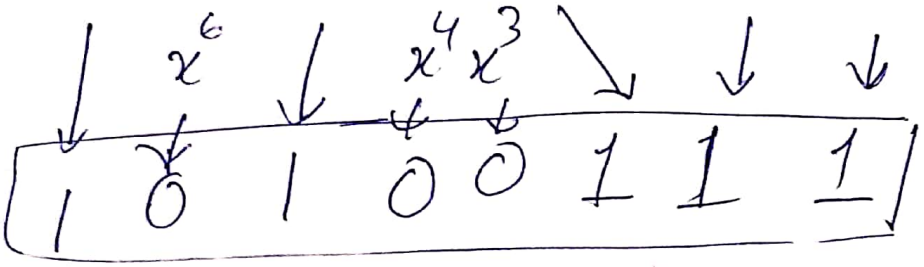
CRC generator & checker



Divisor →

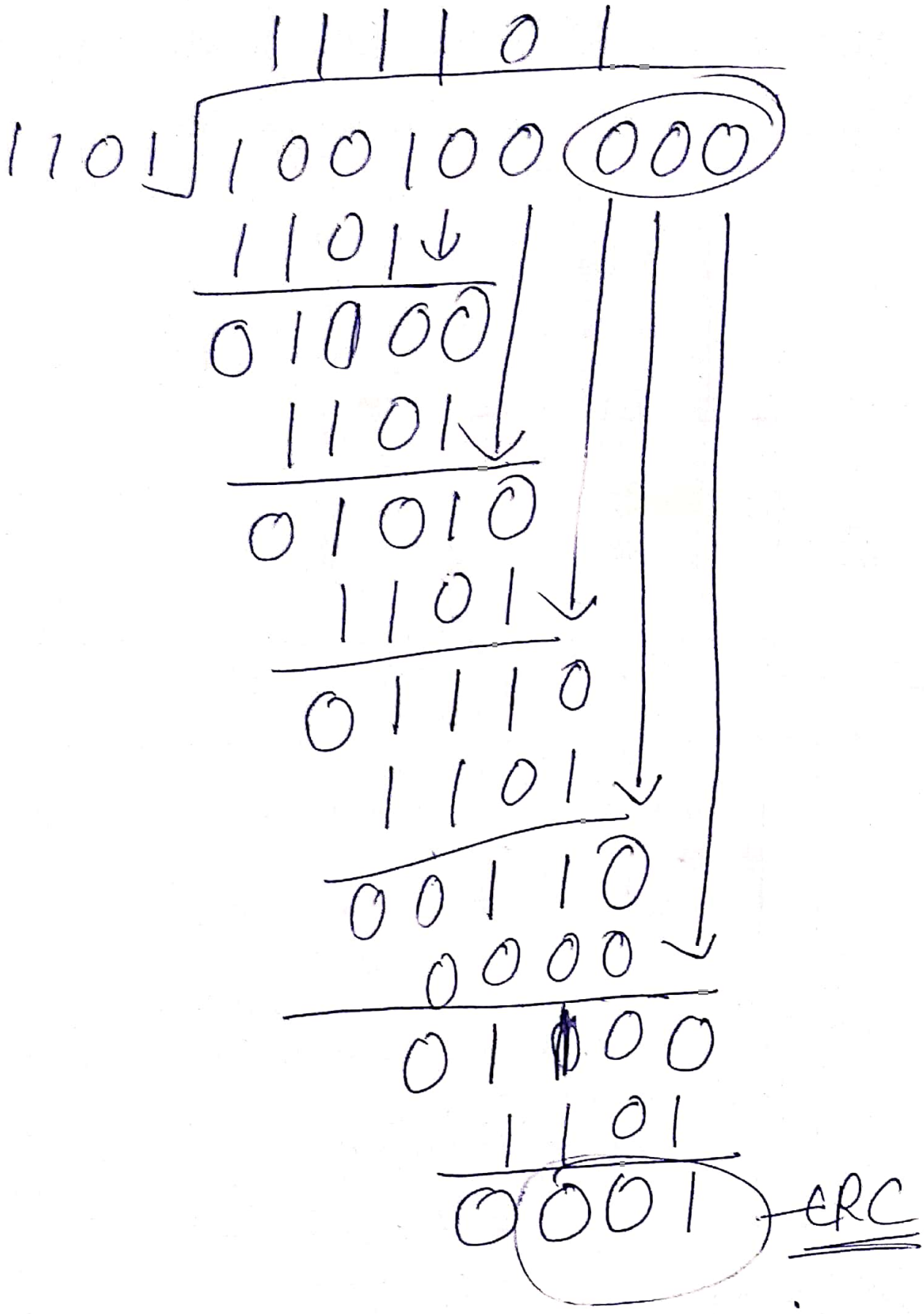
A polynomial is —

$$x^7 + x^5 + x^2 + x + 1$$

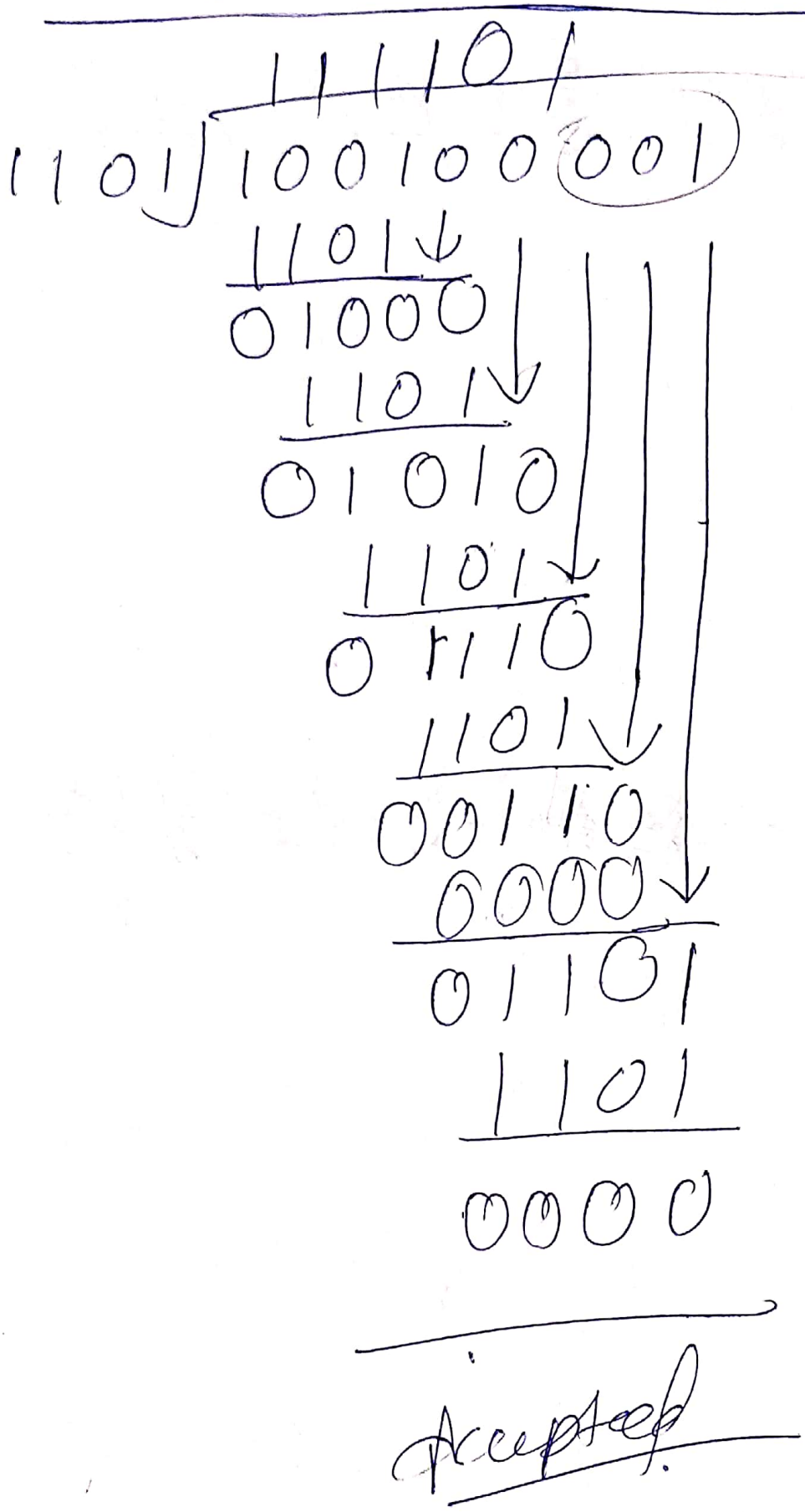


so the divisor is → 10100111

→ CRC generator at sender end



→ CRC checker at receiver end



CHECKSUM

For Error Detection by checksums,

Data is divided into fixed sized frames or segments.

1) Sender End → The sender add the segments using 1's complement arithmetic to get the sum. It then complements the sum to get checksum and sends it along with the data frames.

2) Receiver's End → The receiver add the incoming segments along with the checksum using 1's complement arithmatic to get the sum and then complements it.

If the result is zero, the received frames are accepted; otherwise they are discarded.

CHECKSUM

For Error Detection by checksums,
data is divided into fixed sized frames or segments.

1) Senders End → The sender add the segments using 1's complement arithmetic to get the sum. It then complements the sum to get checksum and sends it along with the data frames.

2) Receiver's End → The receiver add the incoming segments along with the checksum using 1's complement arithmetic to get the sum and then complements it.

If the result is zero, the received frames are accepted. Otherwise they are discarded.

Eg 11001100, 10101010, 1111 0000, 11000011

Sender End

Frame 1: 11001100

Frame 2: 10101010

1 0 0 1 1 0 1 1 0

+ 1

0 0 1 1 0 1 1 1

Frame 3: 11110000

1 0 1 1 0 0 1 1 1

+ 1

0 1 1 0 1 0 0 0

Frame 4: 11000011

Checksum:

1 0 0 1 0 1 0 1 1

+ 1

Sum: 0 0 1 0 1 1 0 0

Checksum: 1 1 0 1 0 0 1 1

Receiver End

Frame 1: 1 1 0 0 1 1 0 0

Frame 2: 1 0 1 0 1 0 1 0

1 0 1 1 1 0 0 1 0

+ 1

0 1 1 1 0 1 1 1

Frame 3 1 1 1 1 0 0 0 0

1 0 1 1 0 0 1 1 1

+ 1

0 1 1 0 1 0 0 0

Frame 4 1 1 0 0 0 0 1 1

1 0 0 0 1 0 1 1

+ 1

Sum 0 0 1 0 1 1 0 0

Checksum 1 1 0 1 0 0 1 1

+ 1 1 1 1 1 1 1 1

Complement 0 0 0 0 0 0 0 0

Hence the frames are accepted as the complement is 0.

Error Correction

1. Single bit Error Correction
2. Burst Error Correction

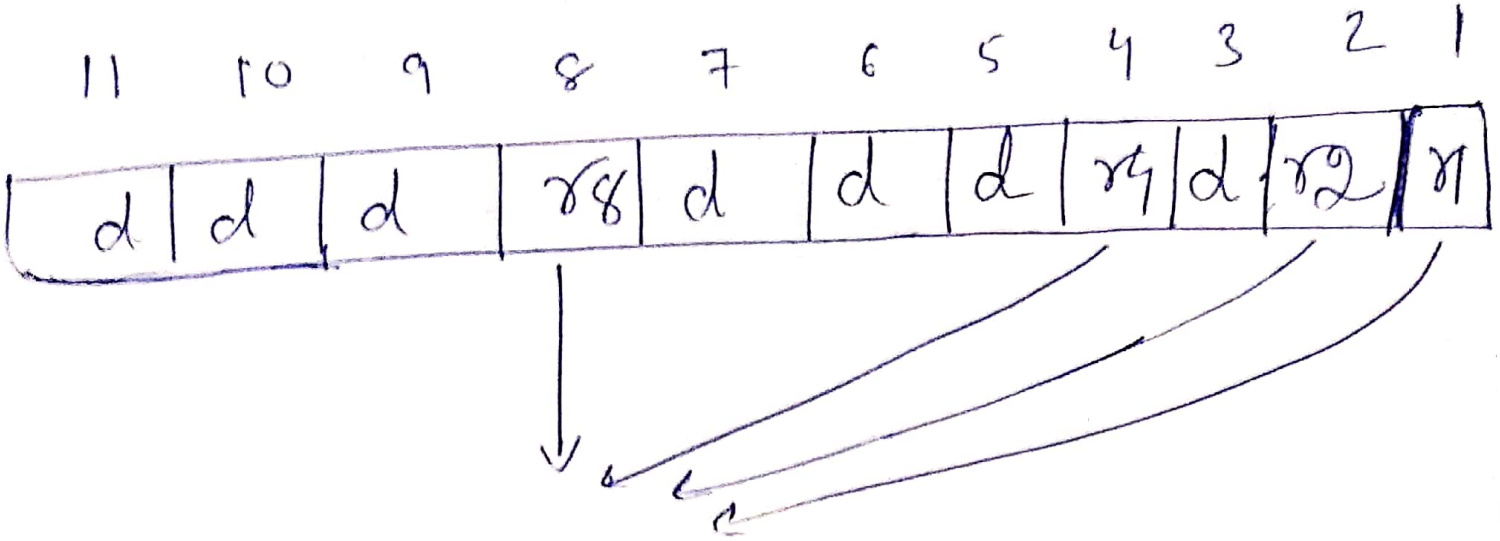
Hamming Code:- for single bit errors. Correction. It is a technique developed by R.W. Hamming.

This technique uses the relationship b/w data and redundancy bits.

Formula

1) A 7 bit ASCII code requires 4 Redundancy bits that can be added to the end of the data unit or in b/w the original bits.

2) These bits are placed in positions 1, 2, 4, 8 We refer to these bits as r1, r2, r4, r8



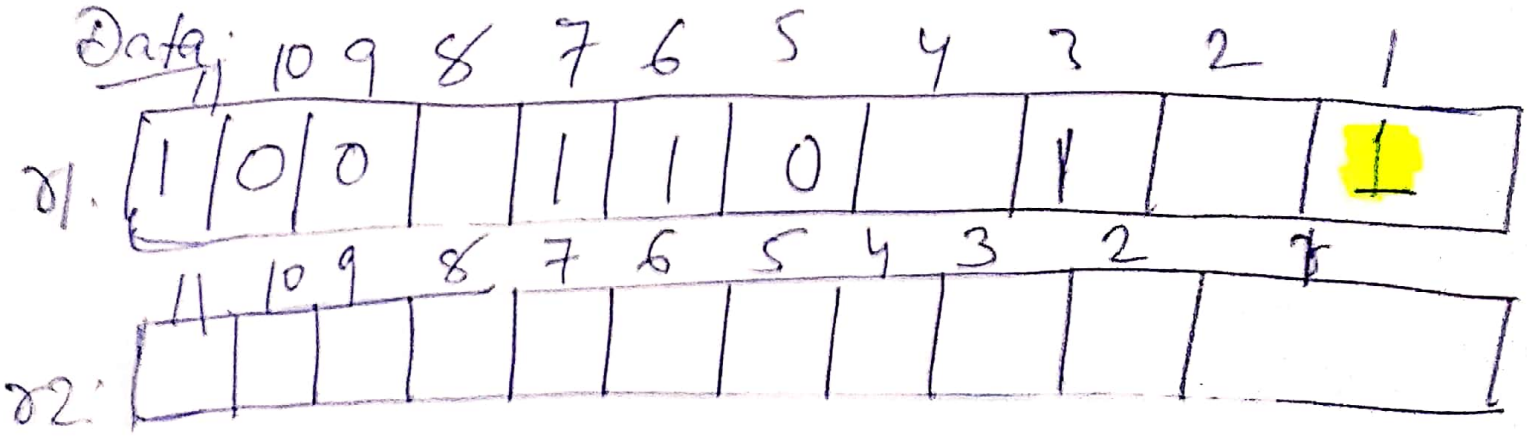
Redundancy Bits

r1 : bits 1, 3, 5, 7, 9, 11

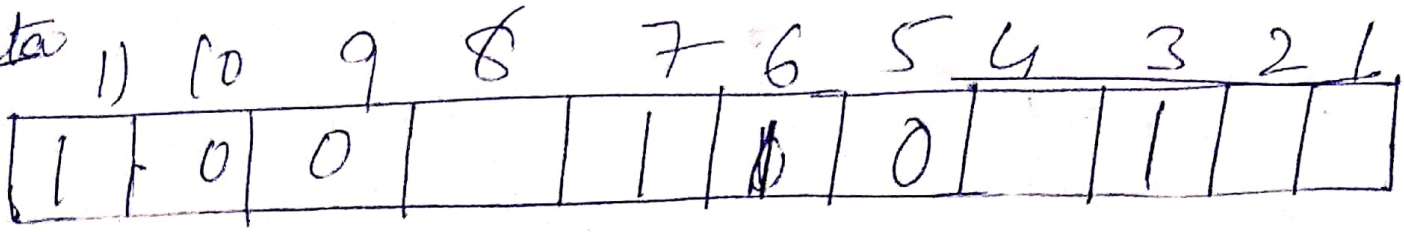
r2 : bits 2, 3, 6, 7, 10, 11

r3 : bits 4, 5, 6, 7

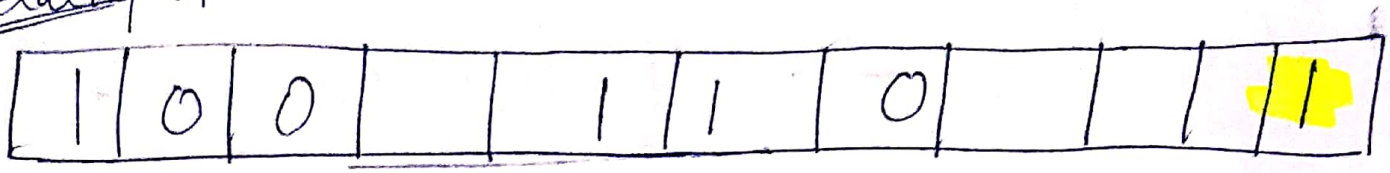
r8 : bits 8, 9, 10, 11



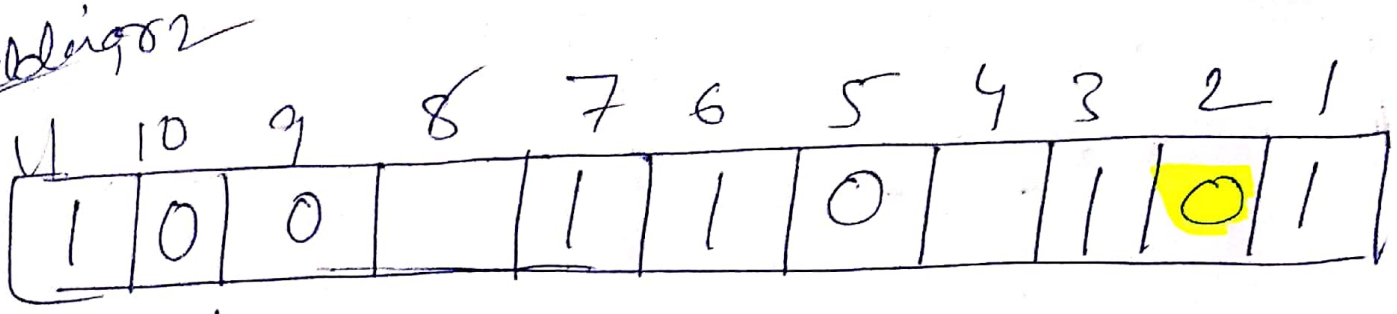
Data



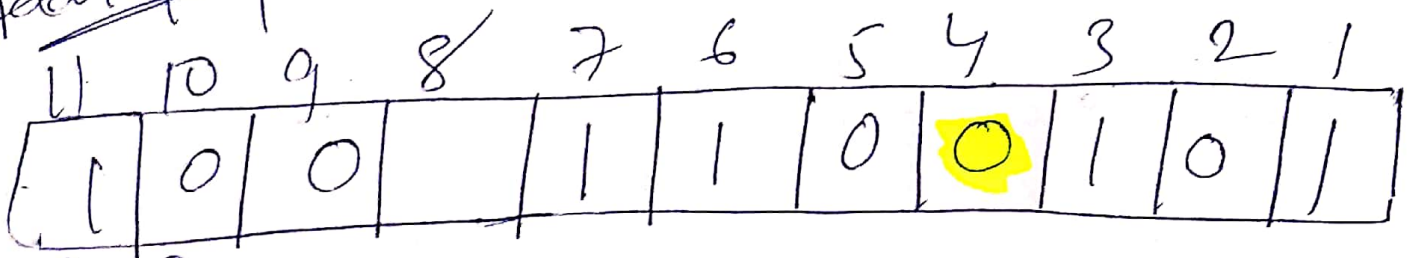
Adding r1



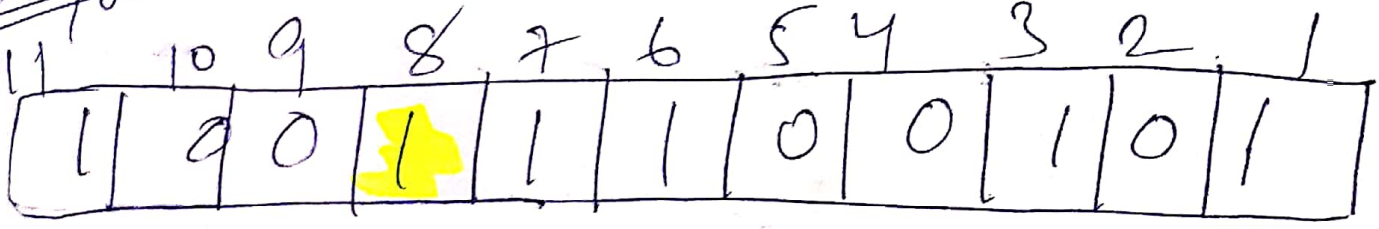
Adding r2



Adding r4



Adding r3

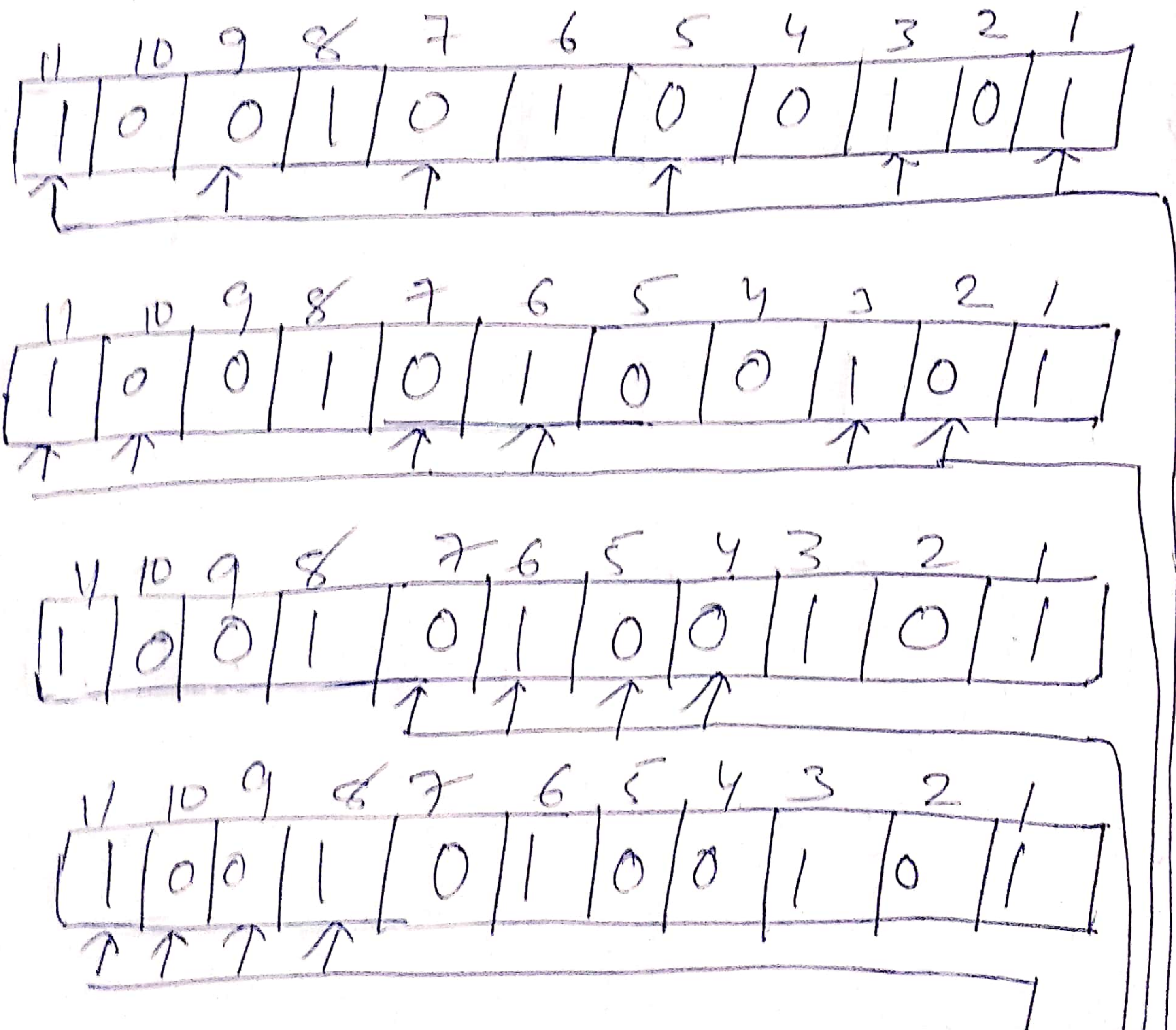


Code:

100111 00101

1 0 0 1 1 1 0 0 1 0 1 → Send

1 0 0 1 0 1 0 0 1 0 1 → Received



7th position is error → 0 1 1 1