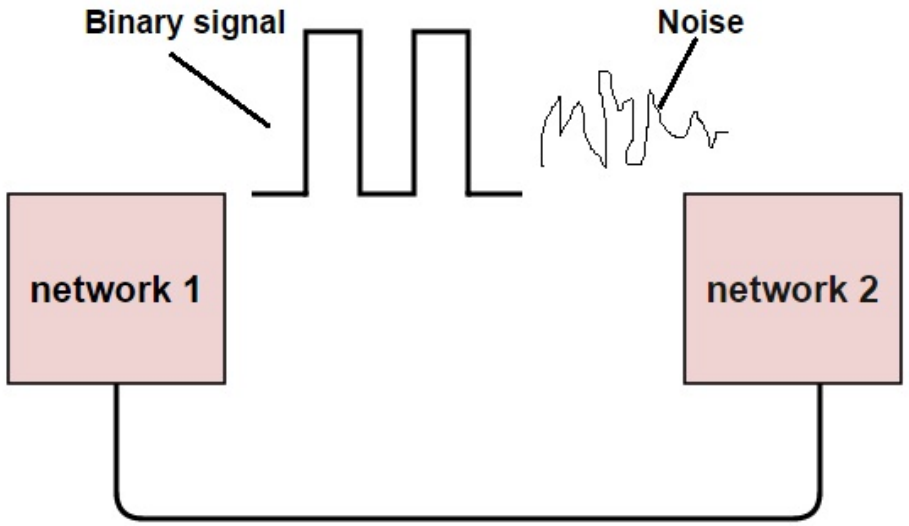
Lecture 6

Error Correction & Detection

What is an Error

- The data can be corrupted during transmission (from source to receiver). It may be affected by external noise or some other physical imperfections. In this case, the input data is not same as the received output data. This mismatched data is called “Error”.
- The data errors will cause loss of important / secured data. Even one bit of change in data may affect the whole system’s performance. Generally the data transfer in digital systems will be in the form of ‘Bit – transfer’. In this case, the data error is likely to be changed in positions of 0 and 1



Types Of Errors

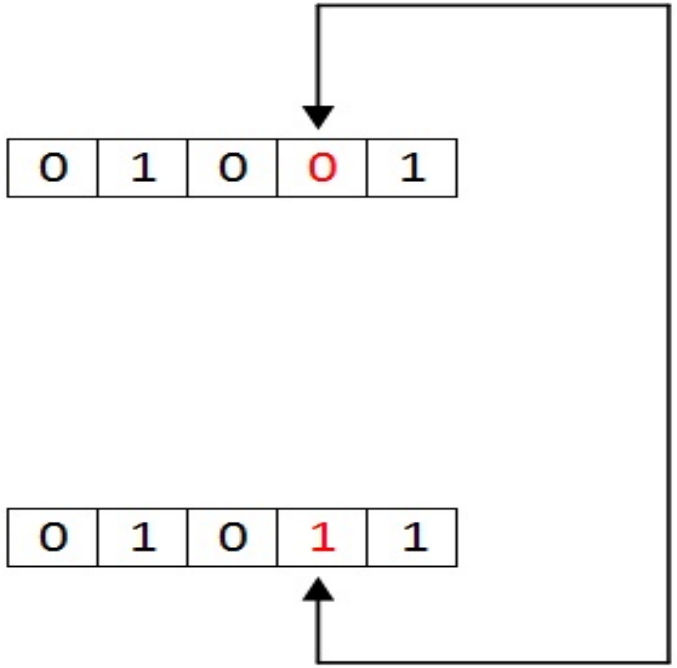
In a data sequence, if 1 is changed to zero or 0 is changed to 1, it is called “Bit error”.

There are generally 3 types of errors occur in data transmission from transmitter to receiver. They are

- Single bit errors
- Multiple bit errors
- Burst errors

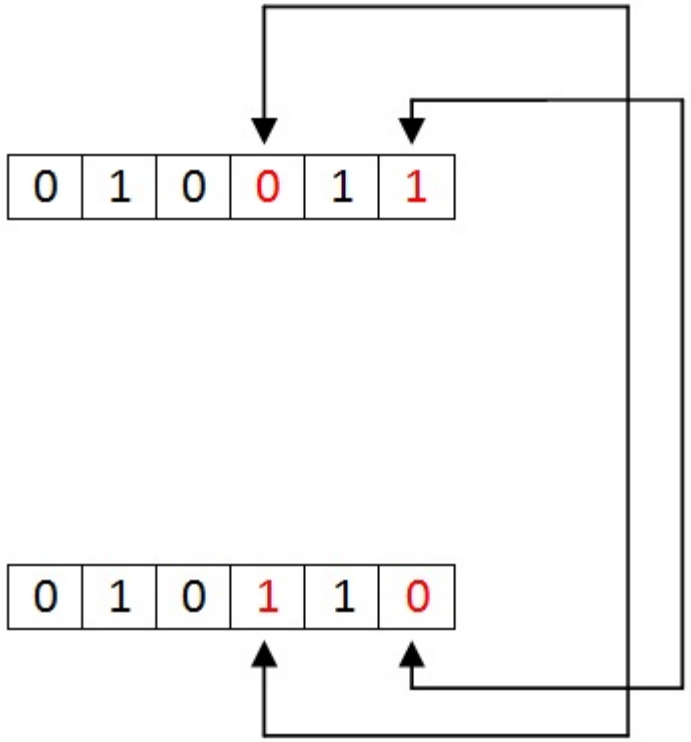
Single Bit Data Errors

The change in one bit in the whole data sequence , is called “Single bit error”. Occurrence of single bit error is very rare in serial communication system. This type of error occurs only in parallel communication system, as data is transferred bit wise in single line, there is chance that single line to be noisy



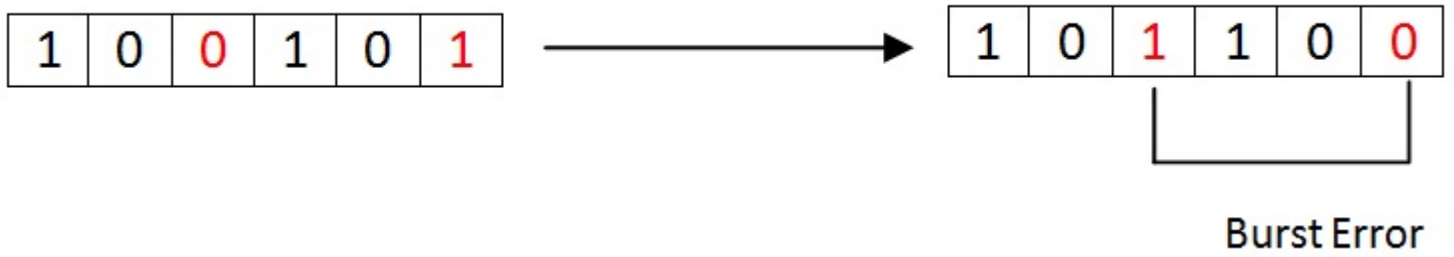
Multiple Bit Data Errors

If there is change in two or more bits of data sequence of transmitter to receiver, it is called “Multiple bit error”. This type of error occurs in both serial type and parallel type data communication networks



Burst Errors

The change of set of bits in data sequence is called “Burst error”. The burst error is calculated in from the first bit change to last bit change.



Here we identify the error form 3rd bit to 6th bit. The numbers between 3rd and 6th bits are also considered as error. These set of bits are called “Burst error”. These burst bits changes from transmitter to receiver, which may cause a major error in data sequence. This type of errors occurs in serial communication and they are difficult to solve.

Error Codes

We know that the bits 0 and 1 corresponding to two different range of analog voltages. So, during transmission of binary data from one system to the other, the noise may also be added. Due to this, there may be errors in the received data at other system.

That means a bit 0 may change to 1 or a bit 1 may change to 0. We can't avoid the interference of noise. But, we can get back the original data first by detecting whether any error[Math Processing Error]

present and then correcting those errors. For this purpose, we can use the following codes.

- **Error detection codes**
- **Error correction codes**

Error detection codes

These are used to detect the error present in the received data. These codes contain some bit, which are included to the original bit stream. These codes detect the error, if it is occurred during transmission of the original data.

- An error detection code is a binary code that detects digital errors during transmission. To detect error in the received message, we add some extra bits to the actual data.

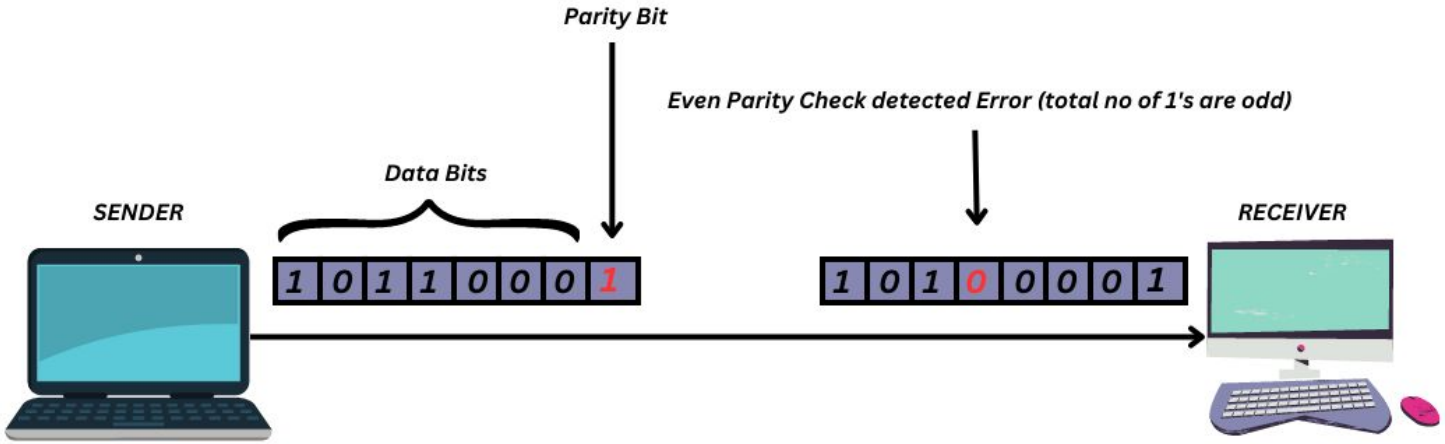
- Without addition of redundant bits, it is not possible to detect errors in the received message. There are 4 ways in which we can detect errors in the received message :

1. **Parity Bit**
2. **CheckSum**
3. **Cyclic Redundancy Check (CRC)**
4. **Longitudinal Redundancy Check**

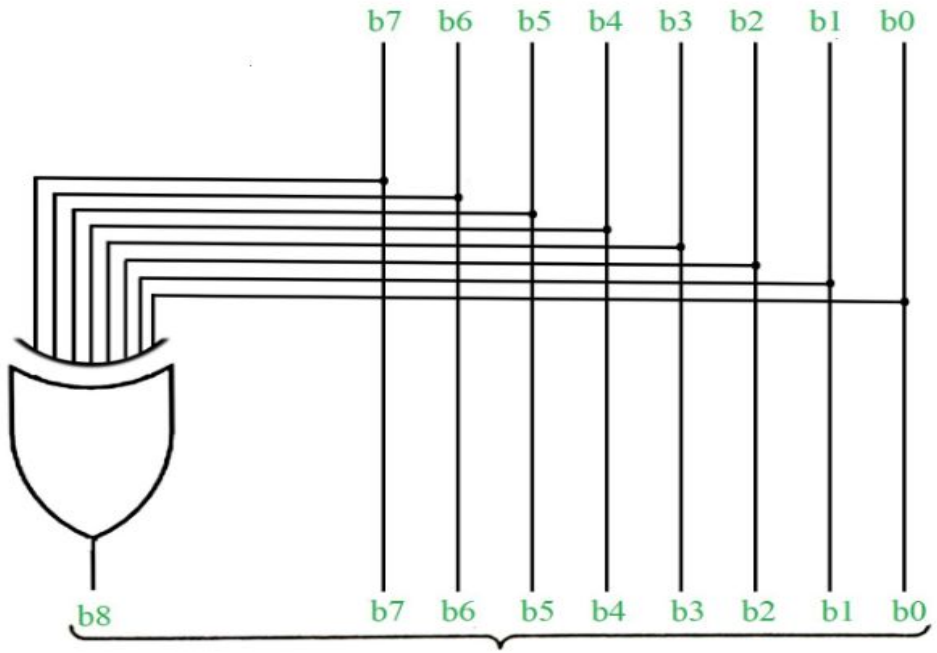
Parity Bit Method

A parity bit is an extra bit included in binary message to make total number of 1's either odd or even. Parity word denotes number of 1's in a binary string. There are two parity system – even and odd parity checks.

1. **Even Parity Check:** Total number of 1's in the given data bit should be even. So if the total number of 1's in the data bit is odd then a single 1 will be appended to make total number of 1's even else 0 will be appended(if total number of 1's are already even). Hence, if any error occurs, the parity check circuit will detect it at the receiver's end. Let's understand this with example, see the below diagram :



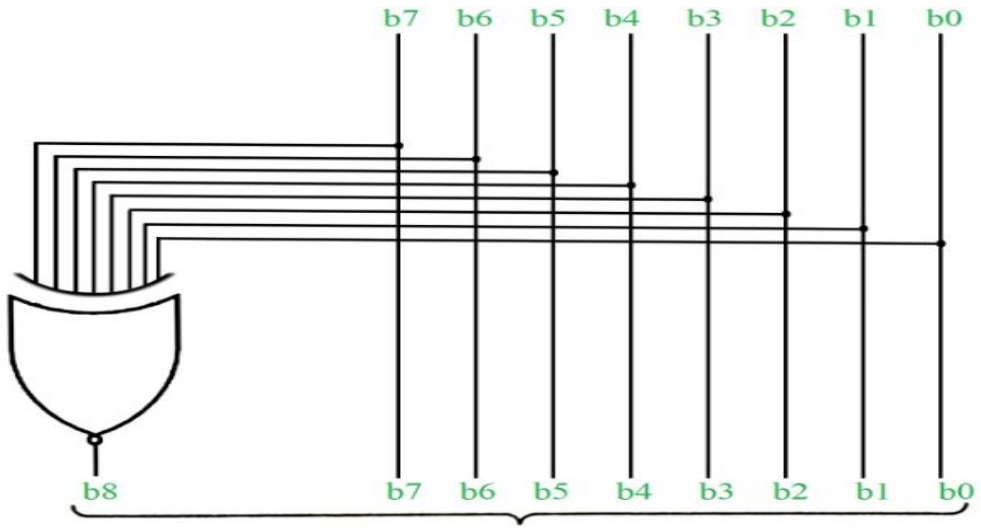
In the above image, as we can see the data bits are ***'1011000'*** and since this is even parity check that we're talking about, 1 will be appended as the parity bit (highlighted in red) to make total count of 1's even in the data sent. So here, our parity bit is 1. If the total count of 1 in the given data bits were already even, then 0 would've been appended.



Odd Parity Check

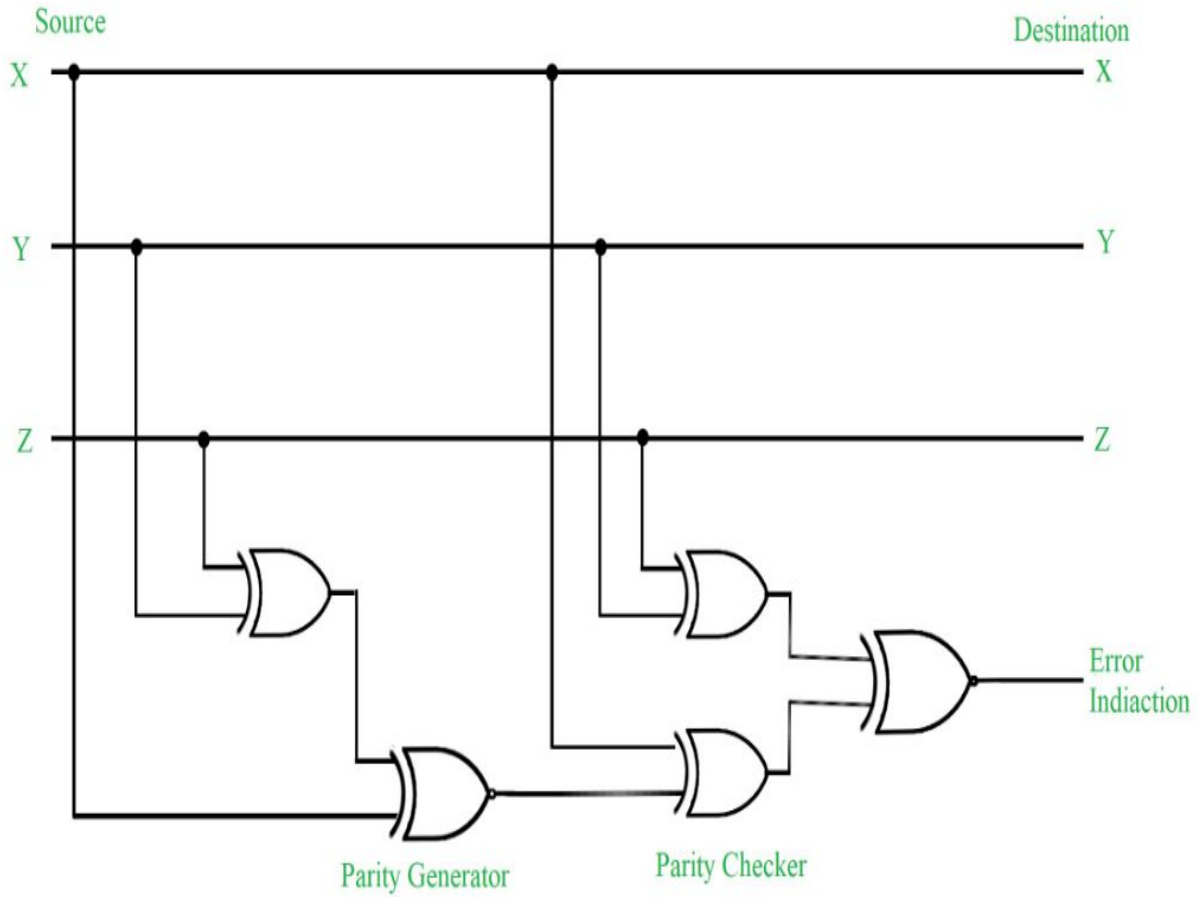
In odd parity system, if the total number of 1’s in the given binary string (or data bits) are even then 1 is appended to make the total count of 1’s as odd else 0 is appended. The receiver knows that whether sender is an odd parity generator or even parity generator. Suppose if sender is an odd parity generator then there must be an odd number of 1’s in received binary string. If an error occurs to a single bit that is either bit is changed to 1 to 0 or 0 to 1, received binary bit will have an even number of 1’s which will indicate an error.

- Take reference from and rather than appending the 1 as parity bit, append 0 because total number of 1’s are already odd.



Some More Examples

Message (XYZ)	P(Odd)	P(Even)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

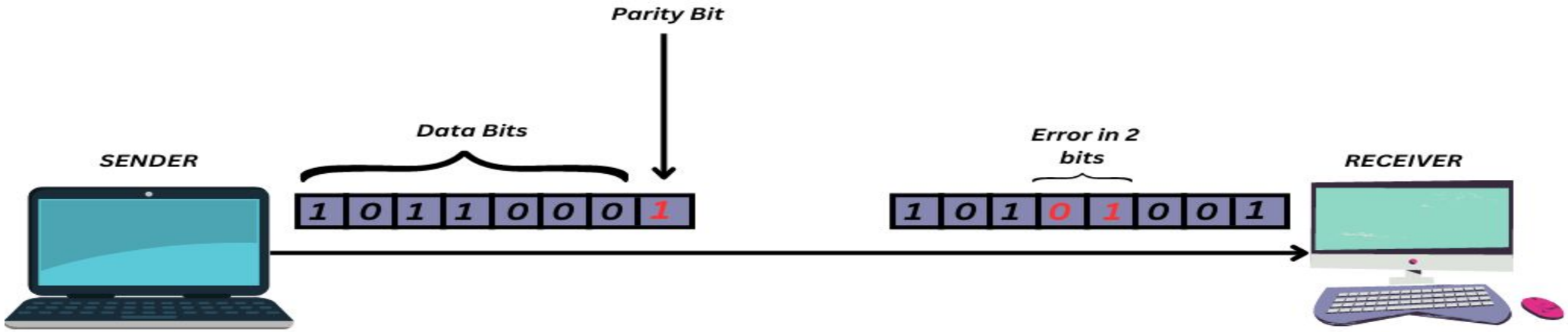


Limitations

1. The limitation of this method is that only error in a single bit would be identified and we also cannot determine the exact location of error in the data bit.

2. If the number of bits in even parity check increase or decrease (data changed) but remained to be even then it won't be able to detect error as the number of bits are still even and same goes for odd parity check.

See image for more details

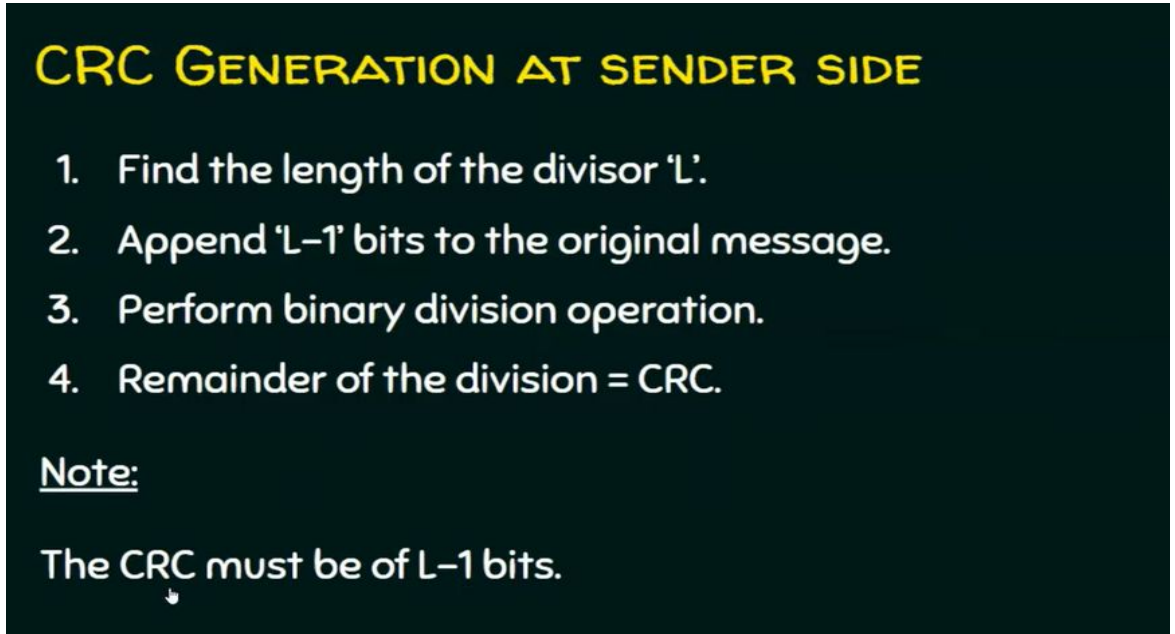


In the
be even, the error can't be detected even though the message's meaning has been changed. You can visualize the same for odd parity check the number of 1's will remain odd, even if the data bits have been changed and the odd parity check won't be able to detect error.

Cyclic Redundancy Check (CRC)

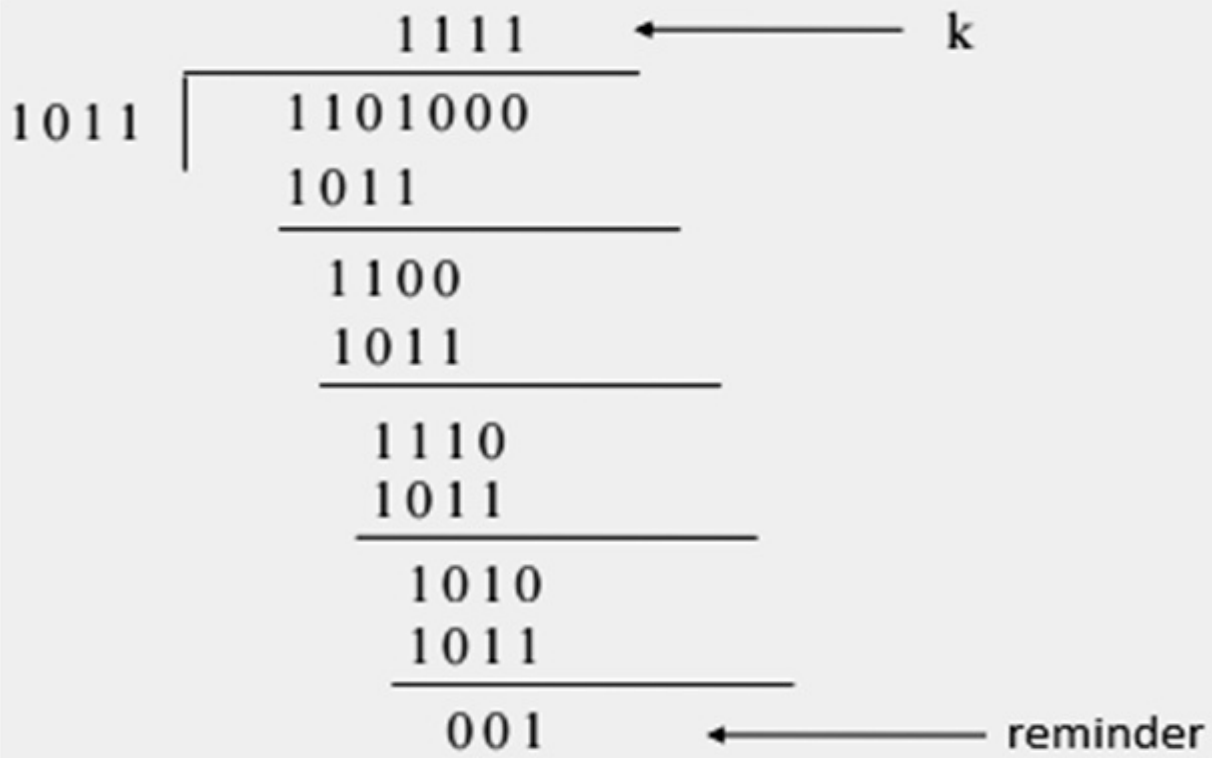
A cyclic code is a linear (n, k) block code with the property that every cyclic shift of a codeword results in another code word. Here k indicates the length of the message at transmitter (the number of information bits). n is the total length of the message after adding check bits. (actual data and the check bits). n, k is the number of check bits.

The codes used for cyclic redundancy check there by error detection are known as CRC codes (Cyclic redundancy check codes).Cyclic redundancy-check codes are shortened cyclic codes. These types of codes are used for error detection and encoding. They are easily implemented using shift-registers with feedback connections. That is why they are widely used for error detection on digital communication. CRC codes will provide effective and high level of protection.



CRC Code Generation

Based on the desired number of bit checks, we will add some zeros (0) to the actual data. This new binary data sequence is divided by a new word of length n + 1, where n is the number of check bits to be added . The reminder obtained as a result of this modulo 2- division is added to the dividend bit sequence to form the cyclic code. The generated code word is completely divisible by the divisor that is used in generation of code. This is transmitted through the transmitter.



CRC

At the receiver side, we divide the received code word with the same divisor to get the actual code word. For an error free reception of data, the reminder is 0. If the reminder is a non – zero, that means there is an error in the received code / data sequence. The probability of error detection depends upon the number of check bits (n) used to construct the cyclic code. For single bit and two bit errors, the probability is 100 % .

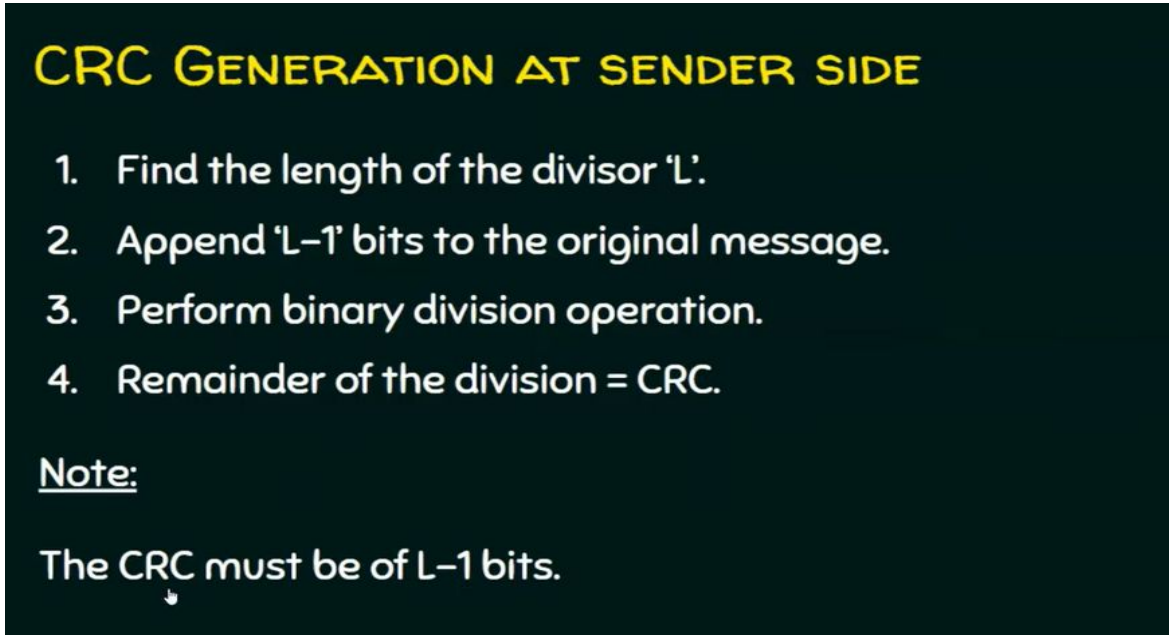
For a burst error of length n – 1, the probability of error detecting is 100 % .

A burst error of length equal to n + 1 , the probability of error detecting reduces to 1 – (1/2)^n-1 .

A burst error of length greater than n – 1 , the probability of error detecting is 1 – (1/2)^n

Problem

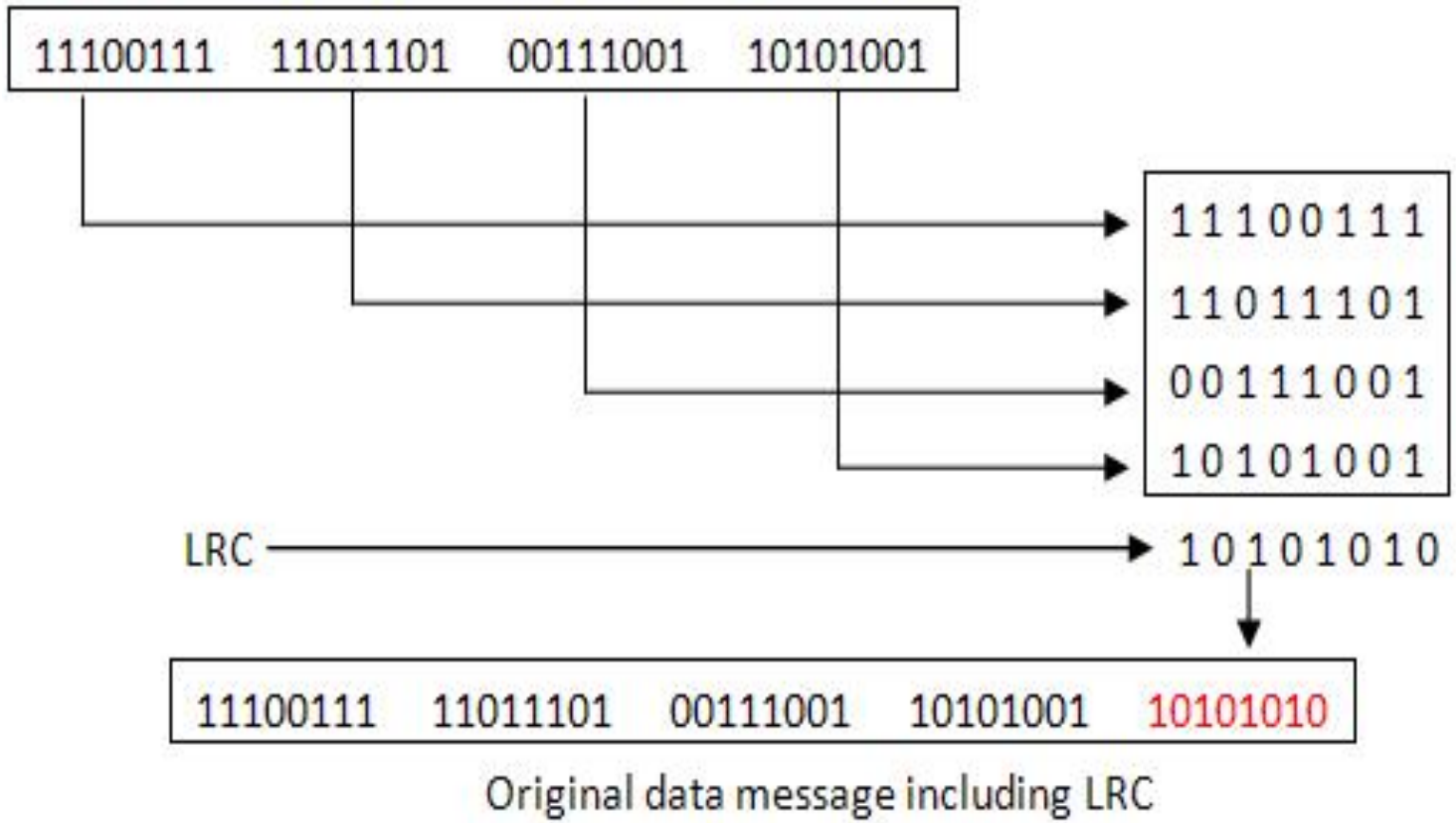
- Find CRC for (1010, 110101)



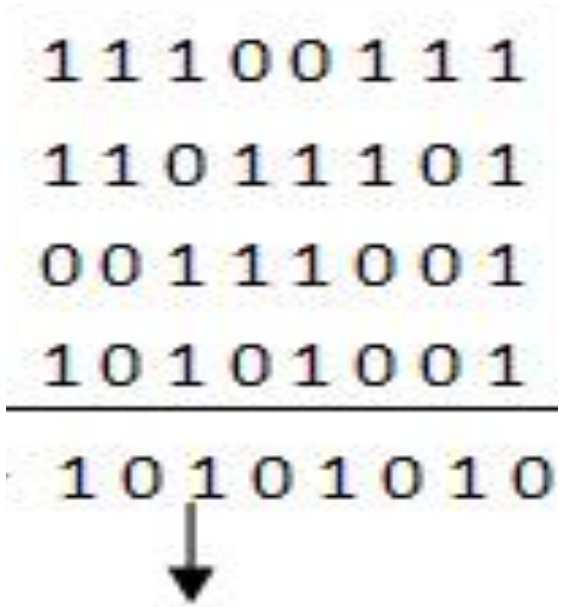
Longitudinal Redundancy Check (LRC)

- In longitudinal redundancy method, a BLOCK of bits are arranged in a table format (in rows and columns) and we will calculate the parity bit for each column separately. The set of these parity bits are also sent along with our original data bits.
- Longitudinal redundancy check is a bit by bit parity computation, as we calculate the parity of each column individually.
- This method can easily detect burst errors and single bit errors and it fails to detect the 2 bit errors occurred in same vertical slice

LRC


11100111 11011101 00111001 10101001
11100111
11011101
00111001
10101001
LRC
10101010
11100111 11011101 00111001 10101001 10101010
Original data message including LRC

LRC



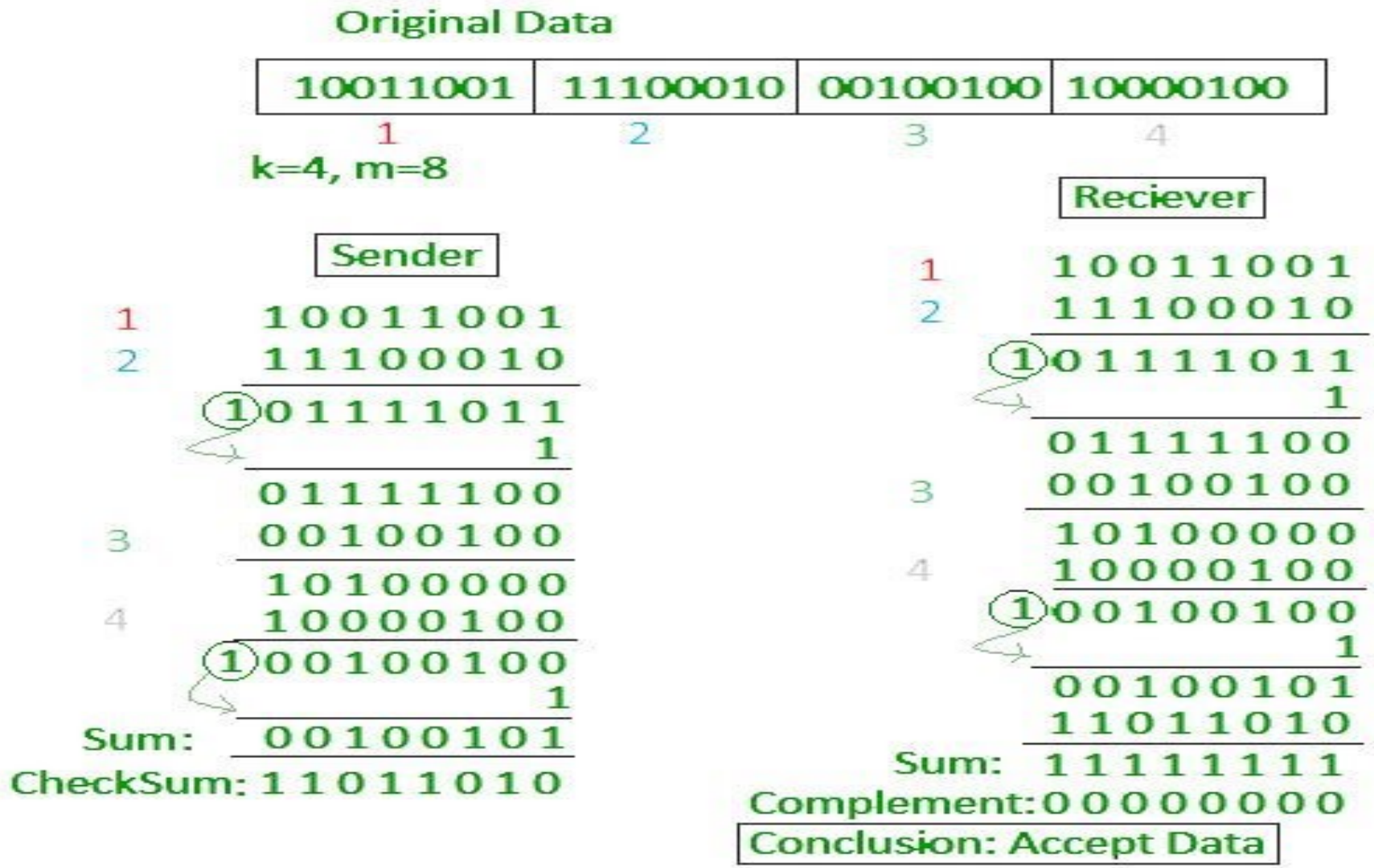
0 1 0 1 1 0 0 0 No Error Detected

CheckSum

- Checksums are similar to parity bits except, the number of bits in the sums is larger than parity and the result is always constrained to be zero. That means if the checksum is zero, error is detected. A checksum of a message is an arithmetic sum of code words of certain length. The sum is stated by means of 1’s compliment and stored or transferred as a code extension of actual code word. At receiver a new checksum is calculated by receiving the bit sequence from transmitter.

- The checksum method includes parity bits, check digits and longitudinal redundancy check (LRC). For example, if we have to transfer and detect errors for a long data sequence (also called as Data string) then we divide that into shorter words and we can store the data with a word of same width. For each another incoming bit we will add them to the already stored data. At every instance, the newly added word is called “Checksum”.

CheckSum


Original Data
10011001
11100010
00100100
10000100
1
2
3
4
k=4, m=8
Sender
1 10011001
2 11100010
1 01111011
1
01111100
3 00100100
10100000
4 10000100
1 00100100
1
Sum: 00100101
CheckSum: 11011010
Reciever
1 10011001
2 11100010
1 01111011
1
01111100
3 00100100
10100000
4 10000100
1 00100100
1
00100101
11011010
Sum: 11111111
Complement: 00000000
Conclusion: Accept Data

10011001	11100010	00100100	10000100

Carry	1	1	1	1	1			
	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	1	1	0	0	0	1	0
	1	0	0	1	1	0	0	1
	0	0	1	0	0	0	1	1
							1	0
	0	0	1	0	0	1	0	1
CHECKSUM	1	1	0	1	1	0	1	0

11011010	10011001	11100010	00100100	10000100

Carry	1	1	1	1	1	1		
	1	0	0	0	0	1	0	0
	0	0	1	0	0	1	0	0
	1	1	1	0	0	0	1	0
	1	0	0	1	1	0	0	1
	1	1	0	1	1	0	1	0
	1	1	1	1	1	1	0	1
							1	0
	1	1	1	1	1	1	1	1

CheckSum

In general, there are 5 types of checksum methods like

- Integer addition checksum
- One’s complement checksum
- Fletcher Checksum
- Adler Checksum
- ATN Checksum (AN/466)

- Using checksum method find if there’s any error at the receiving end or not

While transmitted data value was 10111010110100000111101 and received data value 10111010110100000111011011111111