
# AN-NAJAH NATIONAL UNIVERSITY

## FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

***Computer Engineering Department***

**Computer Communications**

Prepared by:

**Masa Masri** **11923986** section 1

# Cyclic Redundancy Check and Modulo-2 Division

A **cyclic redundancy check** (**CRC**) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to digital data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRCs can be used for error correction (see bitfilters).

CRCs are so called because the *check* (data verification) value is a *redundancy* (it expands the message without adding information) and the algorithm is based on *cyclic* codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function.

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.
```

**Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):**

1. The binary data is first augmented by adding k-1 zeros in the end of the data
2. Use ***modulo-2 binary division*** to divide binary data by the key and store remainder of division.
3. Append the remainder at the end of the data to form the encoded data and send the same

**Receiver Side (Check if there are errors introduced in transmission)**

Perform modulo-2 division again and if the remainder is 0, then there are no errors.

In this article we will focus only on finding the remainder i.e. check word and the code word.

**Modulo 2 Division:**

The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. Just that instead of subtraction, we use XOR here.

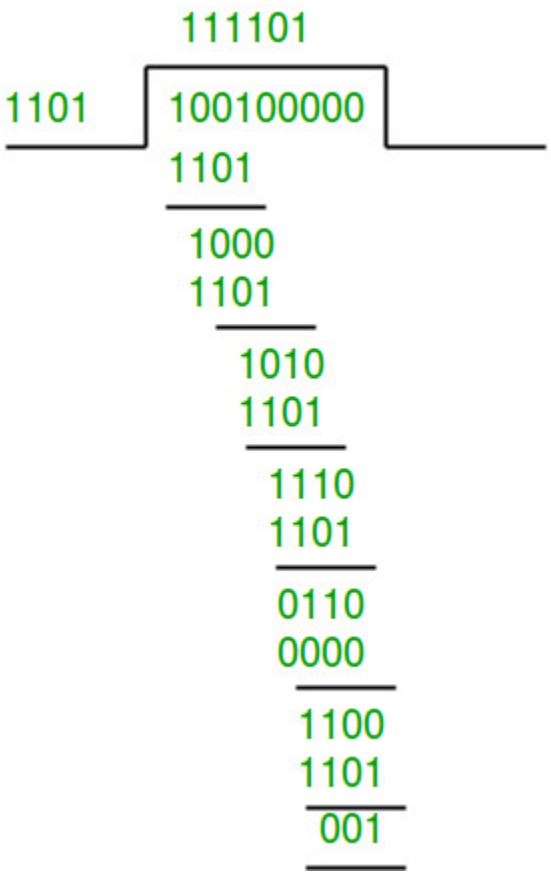
- In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit remainder which is appended at the sender side.

**Illustration:**

**Example 1 (No error in transmission):**

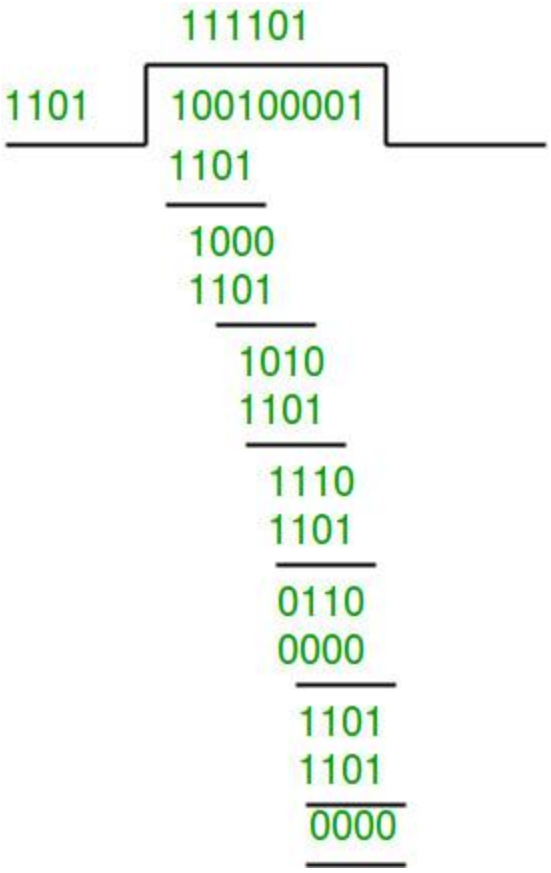
Data word to be sent - 100100
Key - 1101 [ Or generator polynomial $x^3 + x^2 + 1$]

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




Therefore, the remainder is all zeros. Hence, the data received has no error.