CRC Error Checking Chad Unterschultz - Feb 22, 2022

Why is the CRC operation useful?

The CRC operation is useful for error checking data. Errors can be randomly created in transmitted data due to electromagnetic interference in the transmission medium. These errors corrupt the transmitted data. We can't necessarily correct these errors. However, we can check to see if an error has occurred. To do this, CRC error checking is one such method.

CRC error checking is a popular method because of its effectiveness, and ease of implementation. A good CRC can catch literally billions of errors. Likewise, the operations necessary for CRC error checking can be implemented using simple state machines.

How does the CRC operation work?

The CRC operation works by *encoding* data before transmission, then *decoding* it after it is received. The *decoding* will reveal if an error has occurred.

The basic idea behind the CRC operation is essentially division using a special number. This number is known as a CRC Key or CRC Code.

To encode, you want to make your data divisible by the CRC Code:

1. Start with your Data, and your CRC Code
2. Divide your Data by your CRC Code to obtain a Remainder
3. Stuff your Remainder into your Data to create a Codeword
4. Transmit Codeword instead of the Data itself

To decode, you want to check to make sure your data is still divisible by the CRC code:

1. Start with a Codeword
2. Divide your Codeword by your CRC Code to obtain a Remainder
3. If your Remainder is not zero than an error has occurred

The CRC operation does not catch all errors. The length and nature of the code will dictate how many and what kind of errors you can catch, respectively. A good CRC can catch 99.999% of errors, or more. Essentially, the longer the CRC, the better the error catching.

CRC Operation Flow Diagram and Example

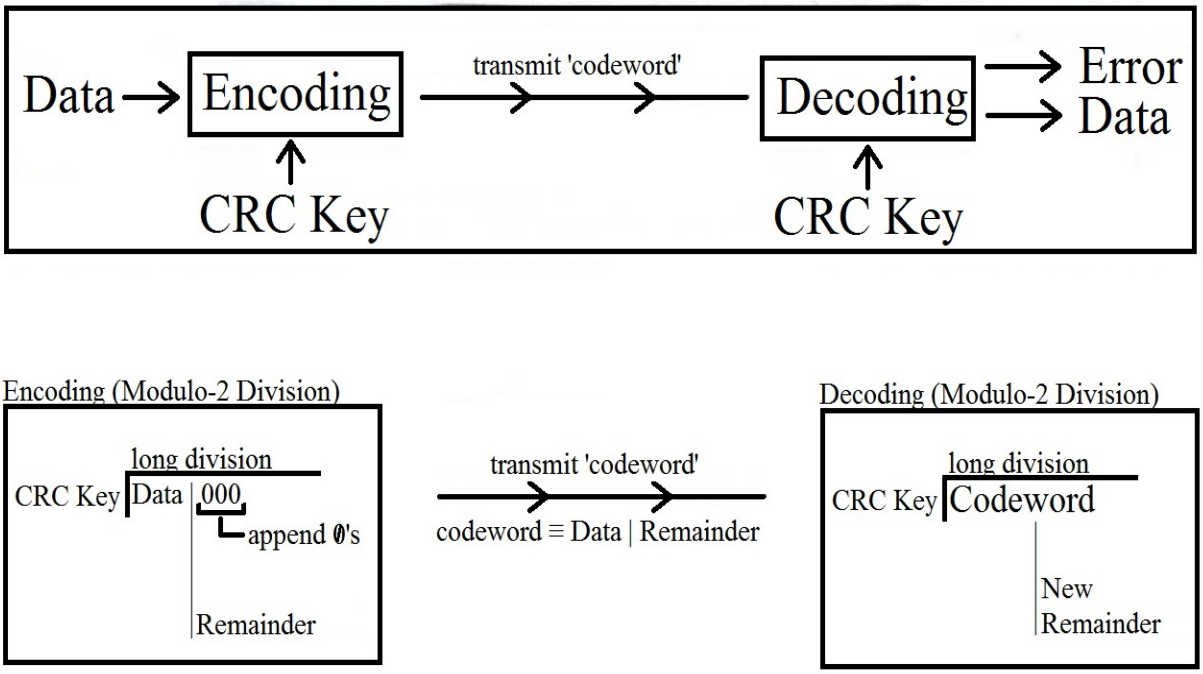

Figure 1: CRC Flow Diagram

Modulo-2 Division (XOR operation)

	0	1
0	0	1
1	1	0

Simple CRC Example

Data: 1001
Key: 1011

Codeword Length = Data Length + Key Length - 1
So append 3 zeros to the data when encoding

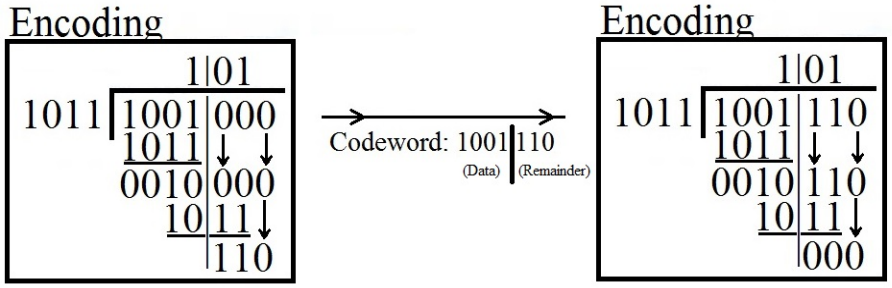

Figure 2: Simple CRC Encoding-Decoding Example

CRC Error Catching

The CRC operations allow you to detect most, but not all errors. The exact types of errors caught are dependent upon the nature of the CRC code itself. This is a high level topic not discussed in here. However, the number of errors that can be cause are based on the length of the CRC code. This is quantified as follows [1]:

The Setup
Given data length: k bits long
Given CRC key length: m bits long
Given Codeword Length: n bits long, where n = k + m - 1

Interpreting the Setup
2^n possible binary variations in a codeword "n" bits long
1 of those variations is your actual codeword
$2^n - 1$ of those variations, the rest of the possible combinations, are errors

Conclusion
$2^n - 2^k$ number of errors can be detected
$\frac{2^n - 2^k}{2^n - 1} * 100$ is the % of error sequences that can be detected.
the number of error sequences that can be detected, over the total number of sequences that are in error

Brief Example

Given 8 bits of data, and a 4 bit CRC: $\frac{2^{11} - 2^8}{2^{11} - 1} * 100 = 87.5$ percent of all errors caught
, and an 8 bit CRC: $\frac{2^{15} - 2^8}{2^{15} - 1} * 100 = 99.2$ percent of all errors caught
, and a 16 bit CRC: $\frac{2^{23} - 2^8}{2^{23} - 1} * 100 = 99.997$ percent of all errors caught

References:

[1] Dr. Ivan Fair, ECE 487, Topic: “Error Detection”, University of Alberta, Sept 16-23, 2014. CRC error detection rates, and general knowledge.
[2] Behrouz A. Forouzan, “Data Communications and Networking”, 4th edition, New York, McGraw-Hill Companies Inc, 2007