Cyclic Redundancy Checker

Submitted by:

Shanti Rayudu (154149)

Rivukanta Bhattacharya (154150)

Divya Sadashivpet (154151)

Submitted to

Sri P. Muralidhar

Associate Professor

Department of Electronics and Communication Engineering

National Institute of Technology, Warangal

2017

Certificate

This is to certify that the undersigned students of II/IV B. Tech Section-A Electronics and Communication Engineering Branch, have completed their mini project titled "Cyclic Redundancy Checker and Generator" for the partial fulfilment of Digital System Design (DSD) Laboratory.

Student's name	Roll no.	Signature
Shanti Rayudu	154149	
Rivukanta Bhattacharya	154150	
Divya Sadashivpet	154151	

Signature of the faculty

(Sri P. Muralidhar)

Associate Professor
Department of ECE
NIT Warangal

Contents

Abstract

Error detection is an important part of communication systems when there is a chance of data getting corrupted. Cyclic redundancy checking is a robust error-checking algorithm, which is commonly used to detect errors either in data transmission or data storage. In integer division dividing A by B will result in a quotient Q, and a remainder R. Polynomial division is similar except that when A and B are polynomials, the remainder is a polynomial, whose degree is less than B. The key point here is that any change to the polynomial A causes a change to the remainder R. This behavior forms the basis of the cyclic redundancy checking.

In terms of cyclic redundancy calculations, the polynomial A would be the binary message string or data and polynomial B would be the generator polynomial. The remainder R would be the cyclic redundancy checksum. If the data changed or became corrupt, then a different remainder would be calculated. Cyclic redundancy calculations can therefore be efficiently implemented in hardware, using a shift register modified with XOR gates. The shift register should have the same number of bits as the degree of the generator polynomial and an XOR gate at each bit, where the generator polynomial coefficient is one. Augmentation is a technique used to produce a null CRC result, while preserving both the original data and the CRC checksum. In communication systems using cyclic redundancy checking, it would be desirable to obtain a null CRC result for each transmission, as the simplified verification will help to speed up the data handling. Traditionally, a null CRC result is generated by adding the cyclic redundancy checksum to the data, and calculating the CRC on the new data.

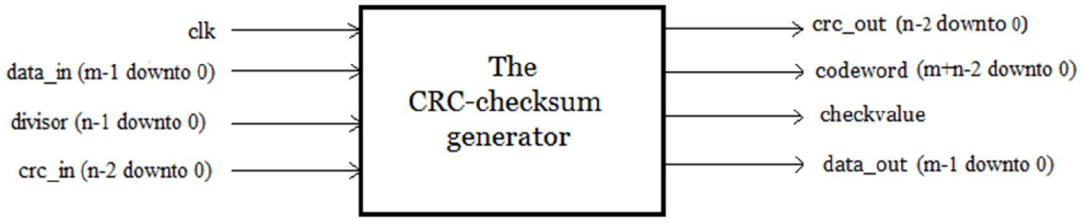
Theory

A cyclic redundancy check (CRC) is a non-secure hash function designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives. A CRC-enabled device calculates a short, fixed-length binary sequence, known as the CRC code or just CRC, for each block of data and sends or stores them both together. When a block is read or received the device repeats the calculation; if the new CRC does not match the one calculated earlier, then the block contains a data error and the device may take corrective action such as rereading or requesting the block be sent again. CRCs are so called because the check (data verification) code is a redundancy (it adds zero information) and the algorithm is based on cyclic codes. The term CRC may refer to the check code or to the function that calculates it, which accepts data streams of any length as input but always outputs a fixed-length code.

A technique known as augmentation is used to produce a null CRC result, while preserving both the original data and the CRC checksum. Augmentation allows the data to be transmitted along with its checksum, and still obtain a null CRC result. As explained before when obtain a null CRC result, the data changes, when the checksum is added. Augmentation avoids this by shifting the data left or augmenting it with a number of zeros, equivalent to the degree of the generator polynomial. When the CRC result for the shifted data is added, both the original data and the checksum are preserved. The degree of the remainder or cyclic redundancy checksum is always less than the degree of the generator polynomial. By augmenting the data with a number of zeros equivalent to the degree of the generator polynomial, we ensure that the addition of the checksum does not affect the augmented data.

Block diagram

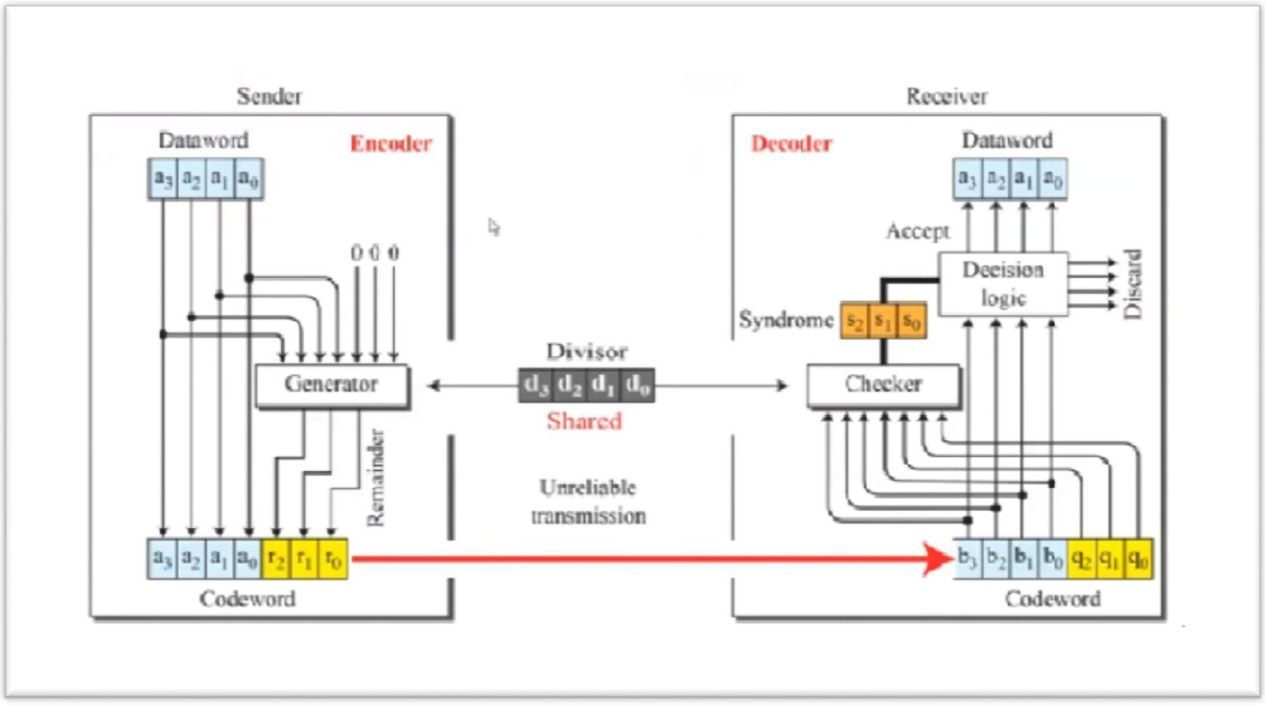
- ## Design entity:



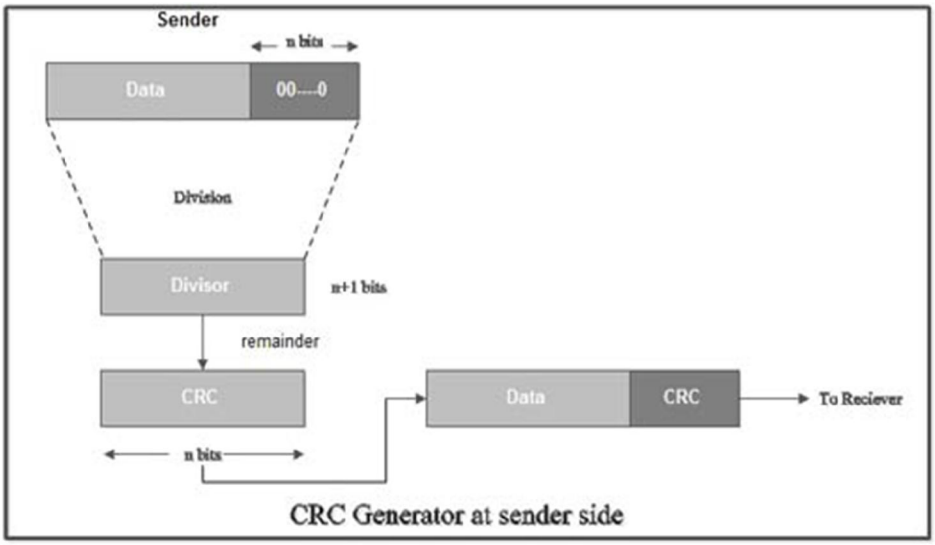
The above figure is the design entity of the CRC checksum generator. The inputs are clk, data_in (m bits), crc_in (n-1 bits), divisor (n bits). The outputs are crc_out (n-1 bits), codeword (m+n-1 bits), checkvalue, data_out (m bits).

- ## Block diagram:

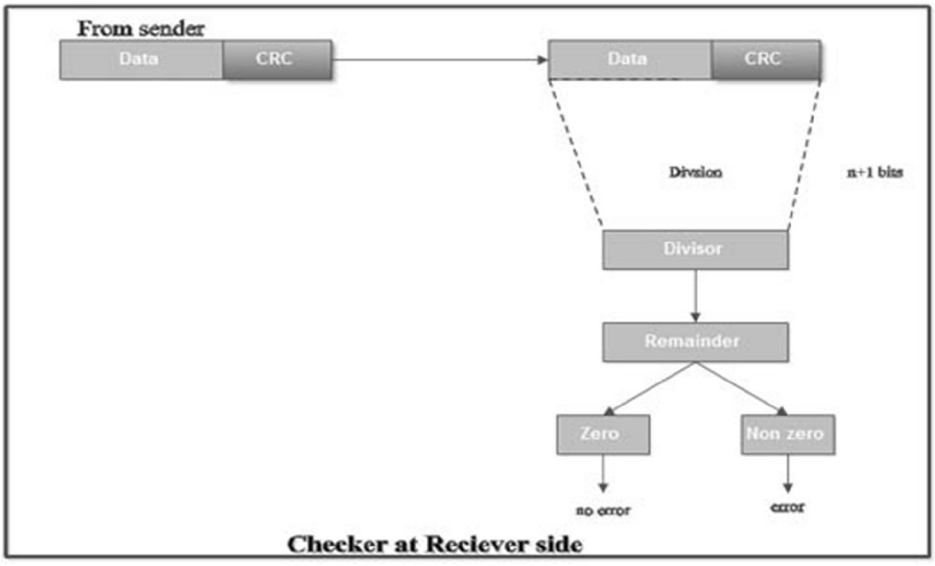
Below is the block diagram consisting of both *Sender* and *Receiver* parts.



- Flowchart:



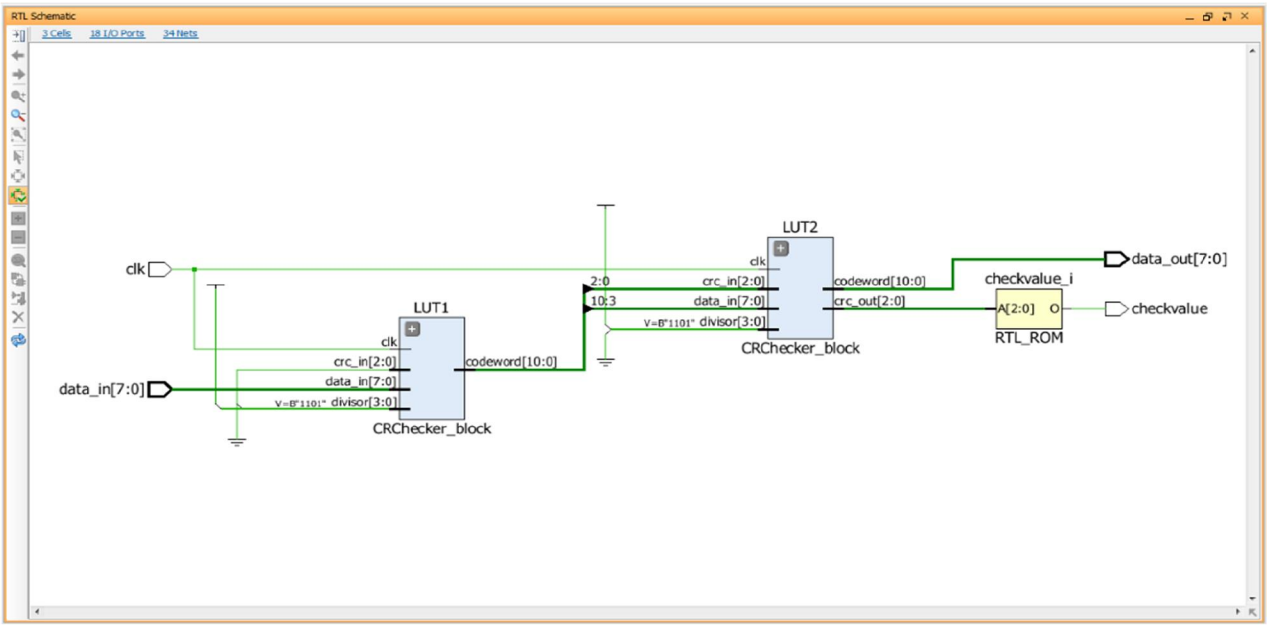
The above figure is the flowchart for the CRC generator at the sender side. The data is appended with 'n' 0 bits and is divided by a generator polynomial of n+1 bits to get the CRC bits. The zeroes are replaced with the CRC bits and the data is transmitted to the receiver.



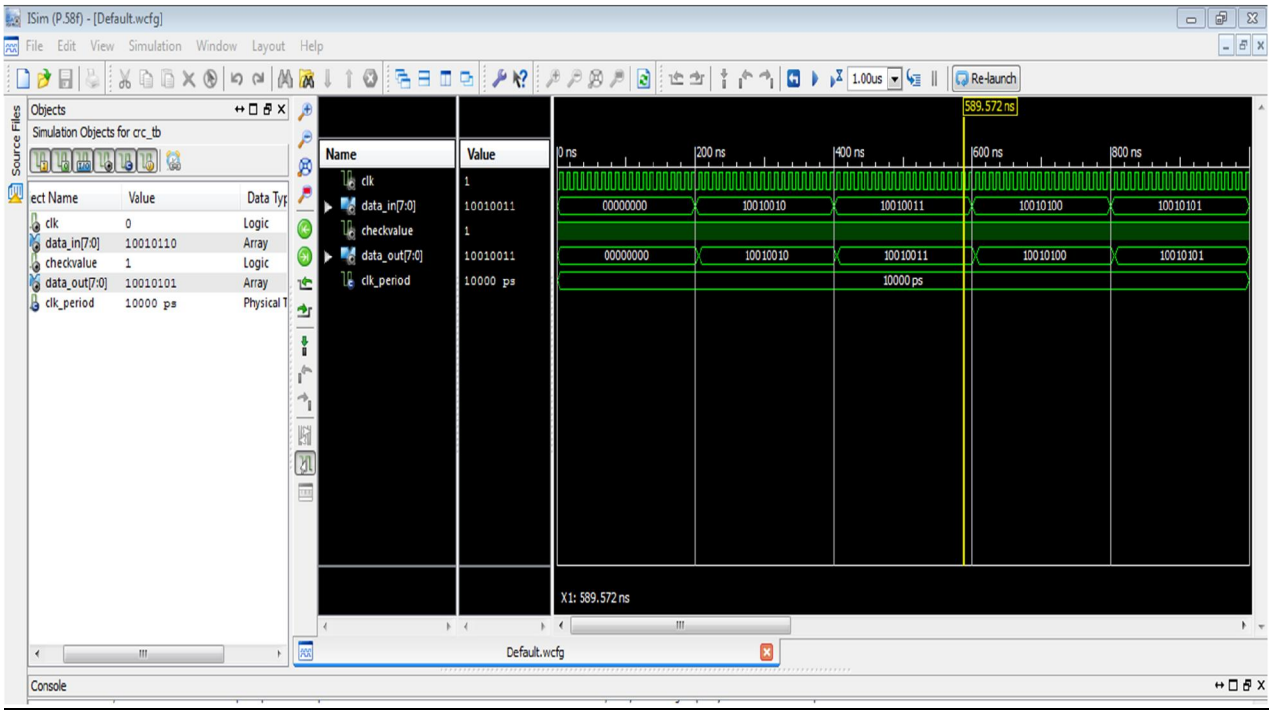
The above figure is the flowchart of the checker at the receiver side. The data with the CRC bits is received and is again divided by the generator polynomial. If the remainder is zero, then there is no error. If the remainder is not zero, then there is an error and the data is discarded.

RTL view & Simulation results

- RTL Schematic (of the top-level entity):



- Simulation results:



Conclusion

- This project gave us an insight about the Cyclic redundancy checker.
- The VHDL model for cyclic redundancy generator and checker was simulated using Xilinx Vivado 2016.4 and the corresponding results were obtained.

References

1. http://www.computing.dcu.ie/~humphrys/Notes/Networks/data.polynomial.html
2. http://ecomputernotes.com/computernetworkingnotes/communication-networks/cyclic-redundancy-check
3. https://www.macs.hw.ac.uk/~pjbk/nets/crc/

THE END