Design & Implementation of Cyclic Redundancy Check Generator

Kaluram Makwana[1], Prof. Madhvi Singh Bhanwar[2]
[1, 2] Dept of Electronics & Communication Engineering
[1, 2] SIRT Indore

Abstract- *Cyclic Redundancy Check is the most important method to detect the errors occurred during transmission of any data. Errors transpired in communication due to internal and external factors i.e. due to components, link, design, noise and interference etc. This paper focused on implementation and analysis of encoder and checker of CRC. The whole design is functionally verified using MATLAB & Xilinx ISE 14.1i.*

Keywords- CRC, Error detection, Communication, VLSI.

I. INTRODUCTION

In an advanced transmission system, an error occurred when a bit is adjusts in the middle of transmission and gathering in a channel. A binary 1 is moved and a binary 0 is gotten or a binary 0 is moved and a binary 1 is gotten. There are two kinds of errors that can happen: single-bit error and burst error [1]. A single bit error is a sequestered error that changes one bit however do not trouble neighboring bits. A burst error is a error in a bordering sequence of B bits in which the first and last bits and any number of halfway bits are gotten in error.

A single-bit error can arise in the occurrence of white noise, due to a slight arbitrary worsening of the signal-to-noise ratio, which is adequate to complicate the receiver's conclusion of a single bit. Spurt errors can be initiated by impulse noise or waning in a mobile wireless background and are more common and more problematic to contract with. The effects of rupture errors are larger at complex data rates. Bit errors occur in digital communication schemes due to inherent or extrinsic factors [2]. Intrinsic errors are due to the constituents, design and application of a link. They are produced due to internal noise causes, poor electrical influences, and sometimes receiver sampler error. In optical contacts the errors occur mainly because of the bodily components such as optical driver, optical receiver, connectors, optical fiber, etc. Errors are also caused due to optical attenuation and optical dispersion. CRCs have a long history of use for error detection in computing [3]. Error correction codes deliver a means to identify and correct errors presented by a transmission network.

Common Cyclic Redundancy Check polynomials are able to notice following types of errors:

a. Single bit error,
b. Double bit errors
c. All odd number of errors having sufficient constraint length
d. Any burst error for which the burst length is less than the polynomial length
e. Large burst errors [4].

Cyclic Redundancy Check uses binary symbols, 0 and 1. Arithmetic is based on GF (2) i.e. modulo-2 calculation (logical XOR) and multiplication (logical AND). The coding arrangement uses systematic codes. Suppose m(x) is the message polynomial, c(x) the code word polynomial and g(x) the generator polynomial. We have c(x)= m(x)g(x) which is also written using the systematic form as c(x)= m(x)xn-k + r(x) , where r(x) is the remainder of the division of m(x)xn-k by g(x) and r(x) represents the CRC bits. The transmitted message c(x) contain k-information bits followed by 'n-k' CRC bits.

If c'(x) is the received message, then no error or untraceable errors have occurred if c'(x) is multiple of g(x) , which is equivalent to determining that if c'(x)xn-k is a multiple of g(x), that is, if the remainder of the division from c'(x)xn-k by g(x) is 0**.**

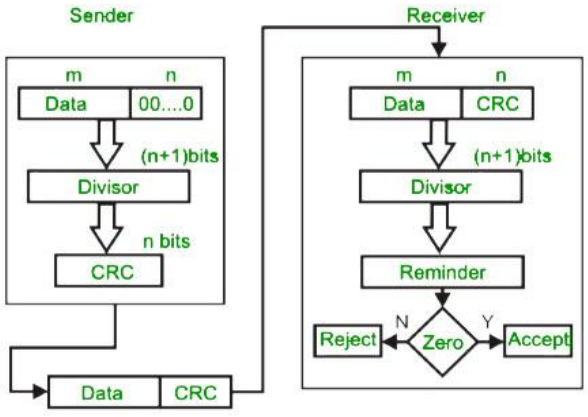

Figure 1: Basic structure of CRC

II. OPERATION OF COUNTERS

Counter Cyclic Redundancy Check (CRC) is an error detecting technique in which a transferred message is attached with a few redundant bits from the sender and then the codeword is plaid at the receiver using modulo-2 arithmetic for errors. The message is then communicated from the encoder and is received by the receiver where a CRC check is conceded out. This procedure supports to regulate any errors in communication over the channel.

The assortment of producer polynomial is the most important part of implementing the CRC algorithm. The polynomial essential be chosen to maximize the error-detecting capabilities while minimizing overall collision probabilities. CRC is divided into the following types:

Table 1: Standard CRC and their polynomials

S.No	CRC	Polynomial	Application
1	CRC4	$x^4 + x^3 + 1$	Telephone
2	CRC8	$x^8 + x^2 + x + 1$	ATM header
3	CRC8 CCIT	$x^8 + x^7 + x^2 + x + 1$	1 wire bus
4	CRC10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	ATM AAL
5	CRC16	$x^{16} + x^{15} + x^2 + 1$	HDLC/USB
6	CRC16 CCIT	$x^{16} + x^{15} + x^5 + 1$	X.25/Modem
7	CRC32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$	Ethernet

A polynomial termed generator polynomial must be designated before the user calculates the CRC of a transferred message. The generator polynomial must have a degree greater than zero and a non-zero coefficient in the MSB and LSB positions.

III. PROPOSED DESIGN & SOLUTION

There are four major CRC implementation solutions [5]. CRC implementation can use either hardware or software solutions. In the traditional hardware implementation, a simple shift register circuit performs the computations by handling the data one bit at a time and parallel implementation by handling data in one word (n-bit) at a time [6, 7]. Software implementations of CRC encoding/ decoding do not resort to dedicated hardware requirements; their applicability, however, is limited to lower encoding rates.

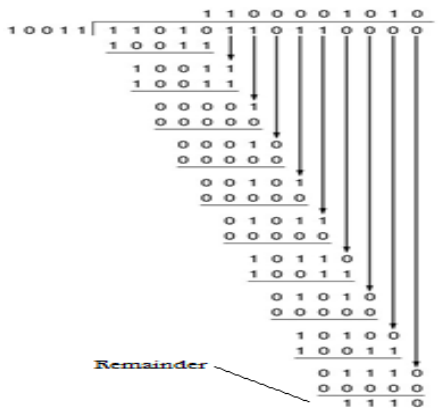

Figure 2: Division in CRC

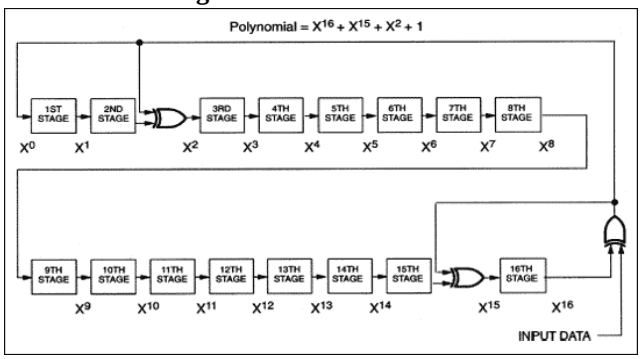

Figure 3: Simple hardware implementation of 16 bit CRC

Division operation of CRC is modulo-2 operation. The divider circuit of CRC can be made up of shift registers and mod-2 adders. The calculation of CRC can be done by the Xilinx to reduce the cost.

ENCODER ALGORITHM

a. Read the message vector.
b. Take the generator polynomial order 'k'.
c. Shift the message vector 'k' times and store it order as 'n'. Compute (n-k).
d. Shift generator polynomial (n-k) times and store the result 'h'.
e. XOR the generator polynomial and message vector and store the result in 'x'.
f. Determine the highest polynomial index position of 'x' where 1 is occurred and take it as 'n'.
g. If n >=k go to step 4.
h. Concatenate the check bits with the message bits.

CHECKER ALGORITHM

a. Read the received data and take its order as 'n' and store it in 'h'.

b. Take the order of generator polynomial as 'k'. Compute (n-k).
c. Shift the generator polynomial left (n-k) times and store the result in 'e'.
d. XOR the generator polynomial and received vector and store the results in 'x'.
e. Determine the highest index of 'x' where 1 is occurred and take it as a 'n'.
f. If n >= k go to step 3.
g. Store 'x' in rem.
h. If rem = 0 then the received data is error free else data contains error.

IV. SIMULATION AND RESULTS

In this section we have showed the result generated from MATLAB then whole design is implemented into Xilinx to check the power dissipation, output delay and memory used.

Case 1: In this Cyclic Redundancy check, polynomial length is 5 and input data has a length of 8 bits.

Input Data is 10010011 and 11001001
CRC polynomial is (x^4+x^3+1) i.e. 11001
Remainder generated is 0001 and 1001

Final codeword transmitted is **100100110001** and **110010011001**

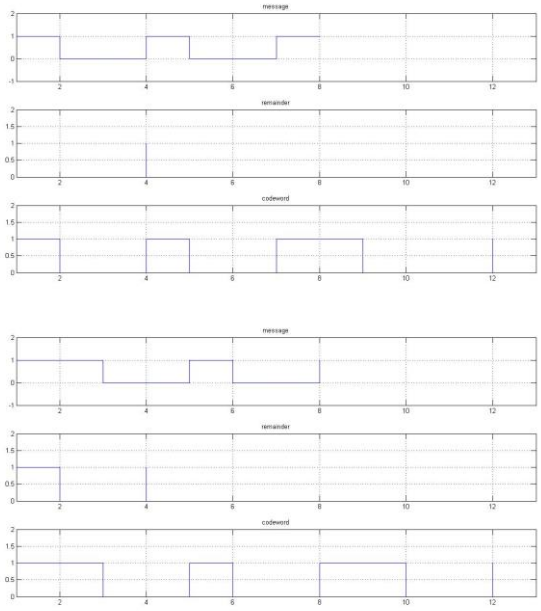

Figure 4: Output waveform for CRC 4 (MATLAB)

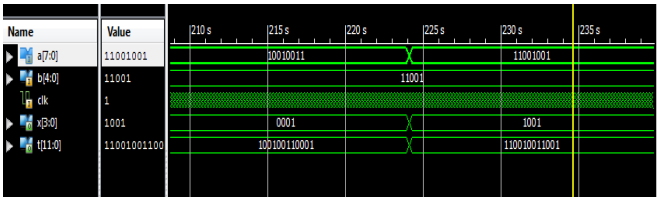

Figure 5: Output waveform for CRC 4 (Xilinx)

Case 2: In this Cyclic Redundancy check, polynomial length is 9 and input data has a length of 8 bits.

Input Data is 10010011 and 11001001
CRC polynomial is (x^8+x^2+x+1) i.e. 100000111
Remainder generated is 11110000 and 01110001

Final codeword transmitted is **1001001111110000** and **1100100101110001**

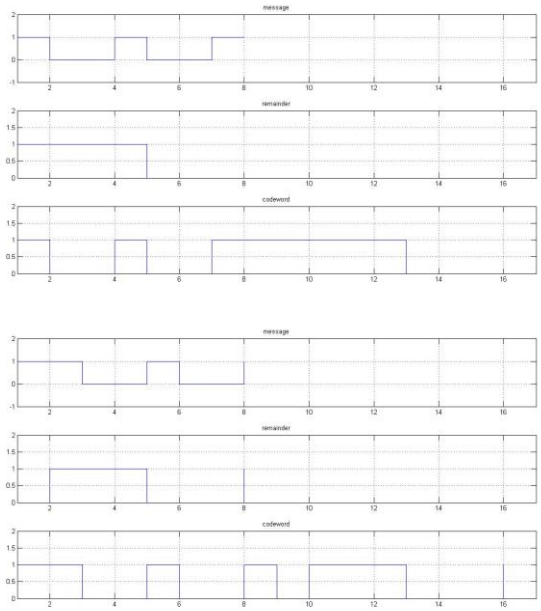

Figure 6: Output waveform for CRC 8 (MATLAB)

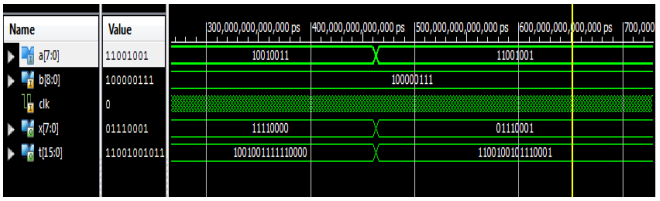

Figure 7: Output waveform for CRC 8 (Xilinx)

Case 3: In this Cyclic Redundancy check, polynomial length is 9 and input data has a length of 8 bits.

Input Data is 10010011 and 11001001
CRC polynomial is ($x^8+ x^7+x^2+x+1$) i.e. 110000111
Remainder generated is 00100111 and 10100100

Final codeword transmitted is **1001001100100111** and **1100100110100100**

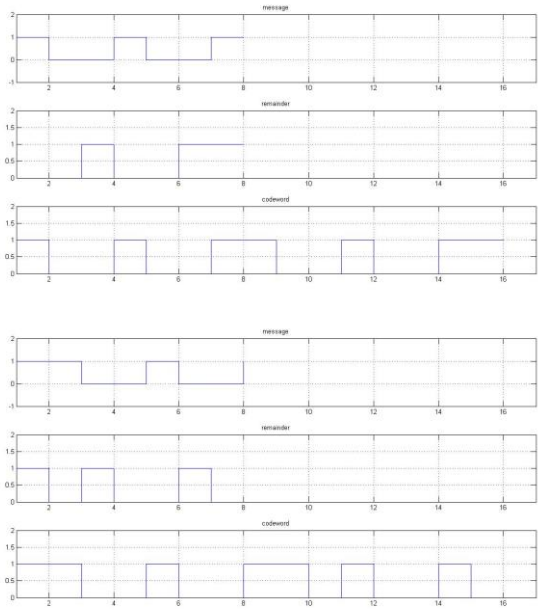

Figure 8: Output waveform for CRC 8 CCIT (MATLAB)

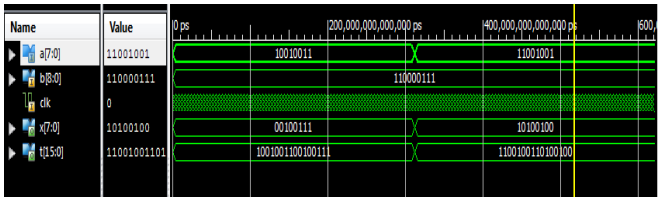

Figure 9: Output waveform for CRC 8 CCIT (Xilinx)

Table 2 describes the logic utilization summary for CRC4, CRC8 and CRC8 CCIT obtained during the analysis of cyclic redundancy check in Xilinx

Table 2: Device utilization summary

Slice Logic Utilization	CRC4	CRC8	CRC8 CCIT
Number of Slice LUTs	17	30	30
Number used as logic	17	30	30
Number of occupied Slices	7	15	16
Number of LUT Flip Flop pairs used	17	30	32
Number of bonded IOBs	25	40	42
Average Fanout of Non-Clock Nets	3.47	3.86	3.74

V. CONCLUSIONS

The method of calculating the remainder for CRC, creating codeword for a given message and generator is described here. Table 2 describes the overall parameter for cyclic redundancy generator. This CRC can be used in different application as per suggested in table 1.

REFERENCES

[1] T Stallings, William, "Data and computer communications", Upper Saddle River, N.J.: Pearson/Prentice Hall, 8th Edition, 2007.

[2] Palani Subbaiah, "Bit- Error Rate for High Speed Serial Data Communication", Data-communications Division, Cypress Semiconductor, November 2008.

[3] Peterson, W. & E. Weldon, "Error-Correcting Codes", Second Edition, MIT Press, 1972.

[4] Ulf Nordqvist, Thesis: "Protocol Processing in Network Terminals", Department of Electrical Engineering, Linkopings University, SE-581 83 Linkoping, Sweden 2004.

[5] U. Nordqvist, T. Henriksson, D. Liu, "CRC Generation for Protocol Processing", Norchip 2000, Turku, Finland, pp. 288-293.

[6] Ming-Der Shieh, Ming-Hwa Sheu, Chung-Ho Chen, Hsin-Fu Lo, "A Systematic Approach for Parallel CRC Computations", Journal of Information Science And Engineering, Vol. 17, 2001 pp. 445-461.

[7] Giuseppe Campobello, Giuseppe Patane, Marco Russo, "Parallel CRC Realization", IEEE Transactions On Computers, Vol. 52, No. 10, 2003, pp. 245-256.

[8] A. H. Saleh, K. M. Saleh and S. Al-Azawi, "Design and simulation of CRC encoder and decoder using VHDL," 2018 1st International Scientific Conference of Engineering Sciences - 3rd Scientific Conference of Engineering Science (ISCES), IEEE 2018, pp. 221-225.

[9] A. K. Panda, S. Sarik and A. Awasthi, "FPGA Implementation of Encoder for (15, k) Binary BCH Code Using VHDL and Performance Comparison for Multiple Error Correction Control," International Conference on Communication Systems and Network Technologies, Rajkot, 2012, pp. 780-784.

[10] Bajarangbali P., Aparna Anand, "Design of High Speed CRC Algorithm for Ethernet on FPGA using reduced lookup table algorithm", IEEE Annual India Conference (INDICON) 2016

[11] Zavodnik, T; Kekely, L.; and Pus, V., "CRC based hashing in FPGA using DSP blocks," in Design and Diagnostics of Electronic Circuits & Systems, 17th International Symposium on , vol., no., pp.179-182, 23-25 April 2014

[12] Kounavis, M.E. and Berry, F.L. "Novel Table Lookup-Based Algorithms for High-Performance CRC Generation," in Computers, IEEE Transactions. pp. vol.57, no.11, pp.1550-1560, Nov. 2008

[13] M.D. Shieh, M.H. Sheu, C.H. Chen, and H.F. Lo, "A Systematic Approach for Parallel CRC Computations," J. Information Science and Eng., vol. 17, pp. 445-461, 2001.

[14] Christopher E. Kennedy and Mehran Mozaffari-Kermani,"Generalized Parallel CRC Computation on FPGA", Proceeding of the IEEE 28th Canadian Conference on Electrical and Computer Engineering Halifax, Canada, May 3-6, 2015

[15] Jubin Mitra and Tapan K. Nayak, "Reconfigurable Concurrent VLSI (FPGA) Design Architecture of CRC-32 for high-speed data communication", 2015 IEEE International Symposium on Nanoelectronic and Information Systems