International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE) Vol 3, Issue 4, April 2016

Implementation of High Speed 64-bit Parallel Cyclic Redundancy Check Generation for Ethernet Application

[1]Sinam Ajitkumar Singh, [2]Patrick Y, [3]L. Surajkumar Singh
Department of Electronics and Communication Engineering
National Institute of Technology Manipur, India
[1]ajit_sinam@yahoo.com, [2]Patrick.yumnam07@gmail.com, [3]surajloi@nitmanipur.ac.in

***Abstract*: CRC is the most powerful tools for detecting and correcting error in data transmission. CRC can be implemented in hardware using Linear Feedback Shift Register. This paper emphasizes mainly on data transmission using Ethernet. Serial CRC can be implemented easily but it cannot achieve high speed. Using F matrix algorithm, Parallel CRC can be implemented, in which speed is drastically increased. While transmitting 64-bit data using serial CRC-32, an output is obtained after 64 clock cycle. However, using Parallel CRC-32 output is obtained after 9 clock cycle, which required 50% less cycles to generate CRC compared to serial CRC generator. The design method is employed with Xilinx ISE 9.2 Tool**

***Index Terms*—Cyclic Redundancy Check, F matrix, Linear Feedback Shift Register, Parallel CRC.**

I. INTRODUCTION

CRCs are a popular technique for detecting errors that is based on cyclic error correcting codes theory. The message is transmitted through different transmitted media it is subjected to attenuation, distortion and noise. As a result, some of the bits of data gets corrupted (error). A CRC is a popular error detecting code computed through binary polynomial division. In 1961, W. W. Peterson proposed CRC [6].

Cyclic Redundancy Check is widely used in digital data communication and other areas such as data compression and data storage, as a powerful method for dealing with data errors. Usually, linear feedback shift registers (LFSRs) is used to implement CRC algorithm in hardware, which handles the data serially using flip-flop and logic gates. As digital data usage increases day by day, high speed data transmission becomes eminent and the increase in data transmission speed means a subsequent increase in error checking, In this case, a parallel computation of the CRC which can handle n bits data simultaneously, which is necessary [1,2].

II. CYCLIC REDUNDANCY CHECK

A. Basic approach

To generate a CRC, the transmitter will transmit a data and performed modulo 2 division operation by a predefined standard generator (e.g., CRC-32). The remainder of this operation becomes CRC of the data, it will append to the original data and transmitted to the receiver. The receiver also performs modulo 2 division operation with the received data and same predefined CRC generator. The operation of CRC performed at transmitter and receiver is shown in fig. 1below.

Operation is shown below, given an m bit block of a bit sequence, the sender generates an n bit sequence known as frame check sequence (FCS), thus the resulting frame consisting of m+n bits, which is exactly divisible by the same predetermined number. The receiver divides the incoming frame by that number and, if there is no remainder, assumes there was no error.

B. Polynomial representation

Some commonly used CRC generator polynomials are:
CRC-32:
$x32+x26+x23+x22+x16+x12+x11+x10+x8+x7+x5+x4+x2+x+1$
CRC-16: $x16+x15+x2+1$
CRC-CCIT: $x16+x12+x5+1$

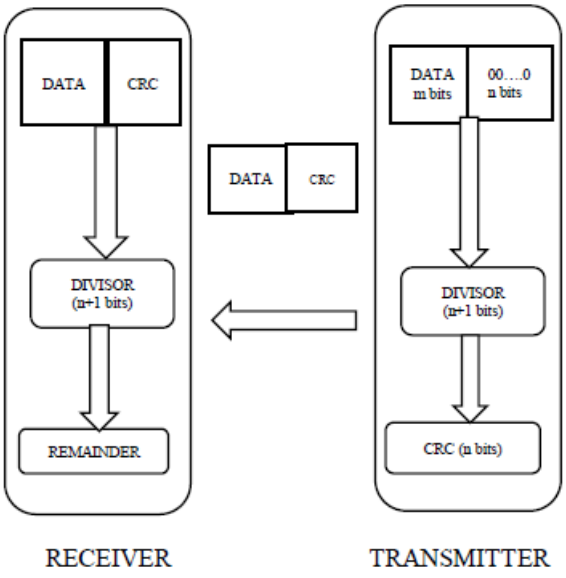

Figure 1. Operation of CRC performed at transmitter and receiver side [6]

III. SERIAL CRC

Serial CRC is designed by using Linear Feedback Shift Register (LFSR) [3] as shown in fig. 2 below.

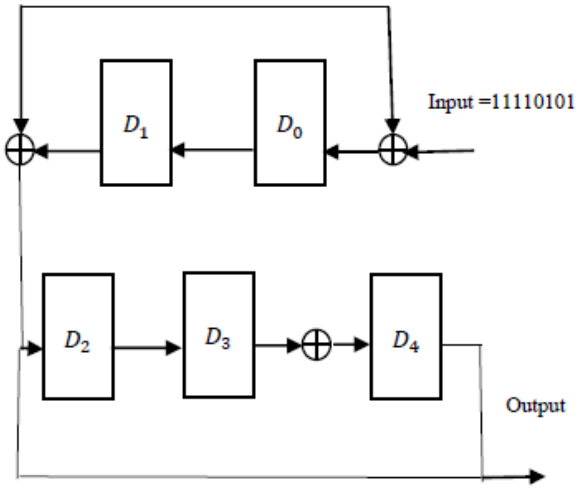

Figure 2. LFSR that divides serial input by using CRC5.

LFSR operation is performed by a series of a shift register, which stored temporal data for every bit wise operation. By XORs between two registers, as shown above fig.2, bit wise operation are performed. For example, in the above fig.2. CRC5 is implemented by using five D flip-flop and three XORs gates. Serial 8 bit input is passed and after eight clock cycle CRC is generated in which data to be sent from transmitter is 1111010110100 as shown in table 1below.

Clk	D_4	D_3	D_2	D_1	D_0	$D_4(+)$ In	$D_4(+)$ D_1	$D_4(+)$ D_2	In
	0	0	0	0	0	1	0	0	1
1	0	0	0	1	1	1	0	0	1
2	0	0	0	1	1	1	1	0	1
3	0	0	1	1	1	1	1	0	1
4	0	1	1	1	1	0	1	1	0
5	1	1	1	0	0	0	0	0	1
6	0	1	0	0	0	0	0	1	0
7	1	0	0	0	0	0	1	1	1
8	1	0	1	0	0				

Table 1. An LFSR computation of (7 + $x6$ + $x5$ + $x4$ + $x2$ + 1)/($x5$ + $x4$ + $x2$ + 1).

If P represents generator polynomial which can be written as P = {pm, $pm-1$, … … … , $p0$}, F is m × m matrix and G as 1× m [3].

The matrices of F and G can be denoted as follows.

$$F = \begin{bmatrix} P_{m-1} & 1 & 0 & 0 & 0 \\ P_{m-2} & 0 & 1 & 0 & 0 \\ P_{m-\beta} & 0 & 0 & 1 & 0 \\ P_{m-4} & 0 & 0 & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

$$G = [0\ 0\ \cdots 0\ 1]^T \quad (2)$$

The state equation for basic serial LFSR can be written as:

$Xi' = F.\ Xi$ G. d (3)

Where Xi and Xi'denotes present state and next state of the system respectively, which is given as Xi = [$xm-1$ ⋯ $x1$ $x0$] and Xi' = [$xm-1$ ′ ⋯ $x1$ ′ $x0$ ′]T . And d denotes serial input data.

IV. PARALLEL CRC

A. Parallel CRC Computation

Parallel CRC is implemented using F matrix algorithm as shown in fig. 3.

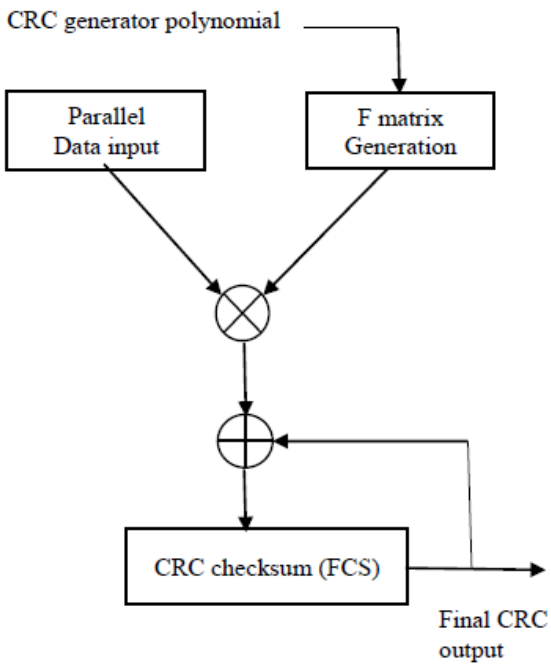

Figure 3: F matrix algorithm for generation of CRC.

Firstly, using CRC generator polynomial F matrix is generated. Each input of F matrix and parallel input data is anded. Finally, the result of anded operation will be exoring with current state of CRC checksum. [1]

B. F Matrix Generation

Let us assume m as a degree of polynomial CRC generator, w bits as a message to be transmitted and k as the length of the message to be processed. Then after $k+m$ w clock cycle, output CRC is generated [1]. F matrix is generated using equation (1) as follows.

$$F = \begin{bmatrix} P_{m-1} & 1 & 0 & 0 & 0 \\ P_{m-2} & 0 & 1 & 0 & 0 \\ P_{m-3} & 0 & 0 & 1 & 0 \\ P_{m-4} & 0 & 0 & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ P_0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Using equation (2) and (3) we can write final equation as
$Xi' = F\ Xi$ D (4)

Where Xi' and Xi indicates next state and current state respectively, and w dimensional data input D =$[dm-1 \cdots d1d0]T$.

Let us take CRC-4 generator polynomial, which is given as P(x) =$x4 + x3 + x1 + x0$. Then using equation (1), F is given as follows.

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$Xremain = Fw$ D(0 to 31) D(32 to 63)
$X' = Fw$ X $Xremain$
Where,
X = current state
X'= next state
D(0 to 31) = first 32 bits input data which processed in parallel manner
D(32 to 63) = next 32 bits input data which processed in parallel manner

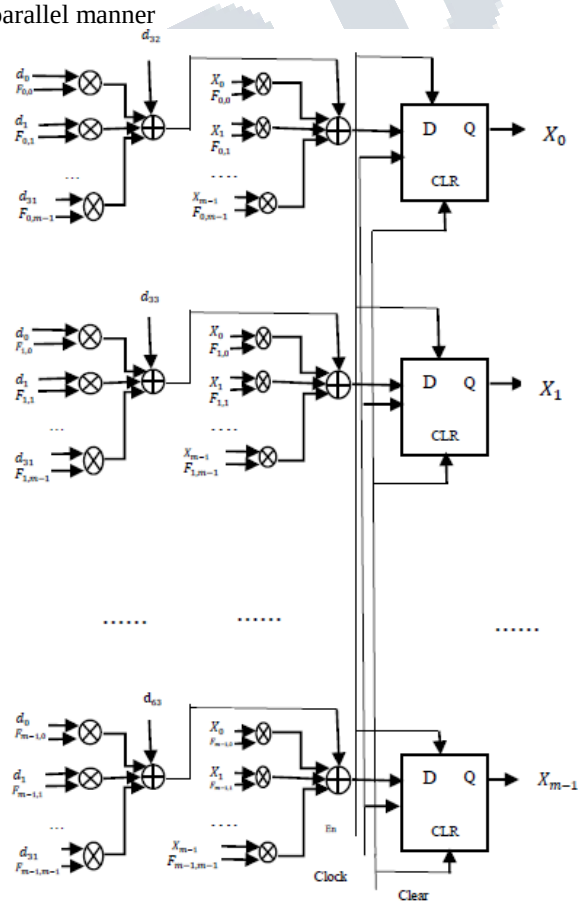

Figure 4: Proposed model of CRC32.

In the proposed model dp is the input data which is processed in a parallel manner. The element of matrix $F32$ with pth row and qthcolumn is represented as Fp, . As shown in fig.4 first 32 input bits data is anded with each row of Fw and result will be xored with next 32 bits input data. To get the final resultant X, each element of $Xremain$ is xored with Xp ′ term of CRC32. The output

CRC is generated after $k+m$ w cycles i.e. after 9 cycle, where w = 64.

IV. RESULT AND ANALYSIS

The design method is employed using Xilinx 9.2i and simulated using Xilinx ISE simulator. The output CRC is obtained after 9 clock cycles. By using Verilog code, CRC generator is designed and implemented. Comparative analysis of various parallel CRC is analyzed in Table.2 below.

Polynomial	LUTs	CPD	Clock cycles
CRC32 [1] W= 32bit	162	30.5 ns	17
CRC32 [7] W= 32bit	220	7.3 ns	17
CRC32 W=64bit (Proposed)	373	4.95 ns	9

Table. 2. Comparison table of a variety of parallel CRC circuits based on LUTs, CPD and Clock Cycles.

The simulation result of our proposed method required 9 clock cycle, CPD required is much less than other two methods whose architecture was proposed by [1][7]. The only disadvantages of this method is that, it required more area compared to other two architecture.

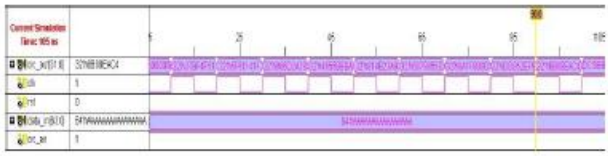

Figure 5: Simulation result for proposed 64bit CRC.

The simulation result for CRC32 with 64 bit parallel data is obtained by Xilinx ISE 9.2i. Input to the system is AAAAAAAAAAAAAAAA (64 bit). After 9 clock cycle, final CRC output is obtained i.e. 8BO8EAC4 (hex form).

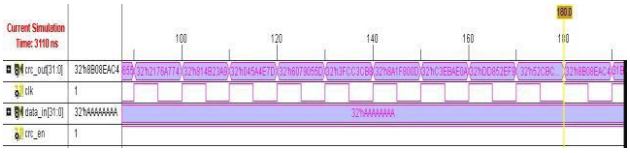

Figure 6: Simulation result for proposed 32 bit CRC.

V. CONCLUSION

32 bit parallel architecture proposed by [1] [7] required 17 clock cycle to generate CRC output**.** Proposed 64 bit parallel CRC32 architecture required only 9 clock cycle, which required 50% less cycles to generate CRC compared to serial CRC generator**.** In this proposed method, pre-calculation of F matrix is not required. Hence this method is simple and easy to implement for high speed CRC generation.

REFERENCES

[1] G. Campobello, G. Patane, Marco Russo, "Parallel CRC Realization," IEEE Transactions on Computers, Vol 52, No.10, pp.1312-1319,Oct.2003.

[2] Yan sun and Min Sik Kim, "High Performance Tle-Based Algorithm for Pipelined CRC Calculation," Consumer Communication and Networking Conference (CCNC), 7th IEEE, Vol. No. pp. 1-2, 9-12 Jan 2010.

[3] Weidong Lu and Stephan Wong, "A Fast CRC Update Implementation," IEEE Workshop on High Performance Switching and Routing, pp. 113-120, Oct 2003.

[4] F Barun and M. Waldvogel,"Fast Incremental CRC Updates for IP over ATM Networks," IEEE Workshop on High Performance Switching and Routing. Vl. No. pp. 48-52, 2001.

[5] Albertengo. G and Sisto. R, "Parallel CRC Generation," Micro, IEEE Vol. No.5 pp. 1312-1319, Oct 2003.

[6] W. W. Peterson and D.T. Brown, "Cyclic codes for error detection," Proceedings of the IRE, Vol. 49, No.1, pp. 228-235, Jan 1961.

[7] M. Spachmann, "Automatic Generation of Parallel CRC Circuits," IEEE Design and Test of Computers, May 2001