International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.02,Issue.06,
July-2013,
Pages:413-418

www.semargroups.org

A H/W Efficient 64-Bit Parallel CRC for High Speed Data Transactions

P.ABDUL RASOOL[1], N.MOHAN RAJU[2]

[1]Research Scholar, Kottam College of Engineering, Chinnatekur, Kurnool, AP-INDIA,
E-mail:abdul.rasool8@gmail.com.

[2]Asst Prof, Kottam College of Engineering, Chinnatekur, Kurnool, AP-INDIA,
E-mail:mohanrajunese@gmail.com.

Abstract: High speed data transmission is the current scenario in networking environment. Cyclic redundancy check (CRC) is essential method for detecting error when the data it is transmitted. Within the challenging speed of the transmitting data, and to the synchronize with speed, it's necessary to increase the speed of the CRC generation. Starting from the serial architecture to be identified a recursive formula from which parallel design is to be derived. In this paper we can presents 64 bits parallel CRC architecture based on F matrix within the order of generator polynomial is 32.When the Proposed design is hardware efficient and required 50% less cycles to generate CRC within the same order of generator polynomial. Then the whole design is functionally verified using Xilinx ISE Simulator.

Keywords: The Cyclic Redundancy Check, Parallel CRC and calculation, Linear Feedback Shift Register, The LFSR, and F matrix.

I. INTRODUCTION

Cyclic redundancy check is commonly used in data communication and other fields such as the data storage and the data compression, as a vital method for dealing with the data errors that is [6]. Usually, then the hardware implementation of CRC computations is based on the linear feedback shift registers and (LFSRs), which handle the data in the serial way. Though, the serial calculation of the CRC codes cannot be achieving a high throughput. In this the contrast is in the parallel CRC calculation can be significantly increase the throughput of CRC computations. For example, then the throughputs of the 32-bit parallel calculation of CRC-32 that can be achieve several gigabits per second [1.] However, that is still not enough for the high speed application such as Ethernet networks. When it is the possible solution is to process more bits in parallel; Variants of CRCs are used for the applications like CRC-16 BISYNC protocols, CRC32 in Ethernet frame for the error detection, CRC8 in the ATM, CRC-CCITT in the X-25 protocol and this disc storage, SDLC, and the XMODEM.

Albertengo and Sisto [2], has to be proposed z-transform based on the architecture for the parallel CRC, in which it's not possible to write synthesizable VHDL code. Braun et al [4] presented an this approach is more suitable for FPGA implementation, which has very complex and the analytical proof. Another approach is based on Galois field has been proposed by Shieh et al.[3]. Campobello [1], has presented pre-calculated F matrix based on the 32 bit is the parallel processing, which is doesn't working if polynomial can be change. In this paper, then the proposed architecture deal with 64bit parallel processing based on the built in F matrix generation; that it gives CRC with half number of cycles. When this paper starts with the introduction of serial CRC generation based on the LFSR.

II. SERIAL CRC

Traditional method for generating serial CRC is based on linear feedback shift registers (LFSR). The main operation of LFSR for CRC calculations is nothing more than the binary divisions.

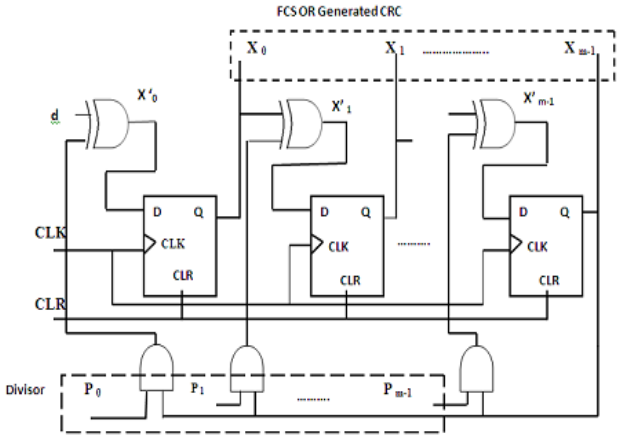

Fig1. Basic LFSR Architecture [2].

When these binary divisions are generally can be performed by the sequence of shifts and subtractions. In this the modulo 2 arithmetic the addition and subtraction are equivalent to bitwise XORs (denoted by "⊕" in this paper) and the multiplication is equivalent to AND (denoted by "⊗"in this paper). Figure 1shows the illustrates of the basic architecture of the LFSRs for serial CRC calculation.

As we can shows in fig.1 d is the serial data input, and X is the present state (generated CRC), when the ' X is the next state and p is the generator polynomial. Working of the basic LFSR architecture that can be expressed in terms of the following equations.

$$X_{0'} = (P_0 \otimes X_{m-1}) \otimes d$$

$$X_{i'} = (P_0 \otimes X_{m-1}) \otimes X_{i-1} \quad (1)$$

Then the generator polynomial for the CRC-32 is as follows

$$G(X) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8 + x^7+x^5+x^4+x^2+x^1+x^0; \quad (1.1)$$

We can extract these coefficients of G(x) and we can represent in binary form as

$$P = \{p_{32}, p_{31}, \ldots\ldots\ldots\ldots, p_0\}$$

$$P = \{10000010011000001000111011011 0111\} \quad (1.2)$$

Then the Frame Check sequence (FCS) will be generated after (k+m) the cycle, where k indicates the number of data bit and m indicates the order of the generator polynomial. When the 32 bits serial CRC if in order of generator polynomial is 32 then these serial CRC will be generated after 64 cycles.

III. PARALLEL CRC

There are different techniques for parallel CRC generation given as follow.

1. A Table-Based Algorithm for Pipelined CRC Calculation.
2. Fast CRC Update.
3. F matrix based parallel CRC generation.
4. Unfolding, Retiming and pipelining Algorithm.

LUT base architecture provides lower memory LUT and by the high pipelining Table base architecture has inputs, LUT3, LUT2, and LUT1. Then the LUT3 contains CRC values for the input that can be followed by 12 bytes of zeros, LUT2 8 bytes, and LUT4 4 bytes. When basically this algorithm it can be obtain for the higher throughput. Then the main problem can be with the pre-calculating CRC and store it in LUT so, that the every time will be required to change the LUT when this changing is the polynomial. When this pipelining algorithm is used to reducing the critical path by adding the delay elements.

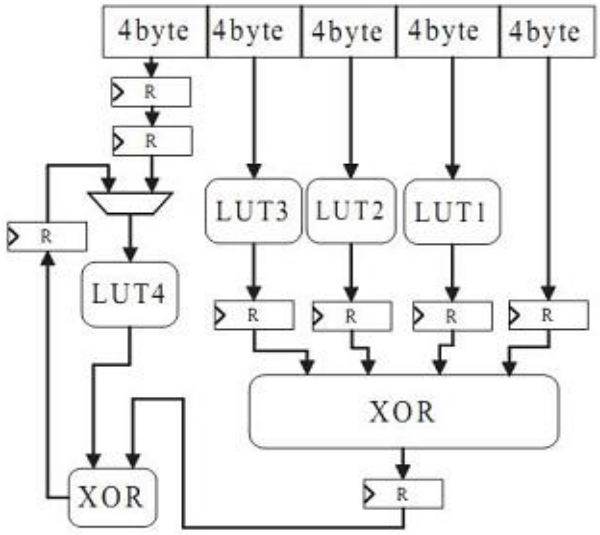

Fig2. LUT based architecture [5]

Parallel processing is used to increasing the throughput by producing the no. of output at the same time. Retiming used to be increasing the clock rate of the circuit by reducing the computation time of the critical path.

In this fast CRC update technique is not required to calculate the CRC each time for all the data bits, instead of that the calculating CRC for only those bits that are change. There are different approaches to generate the parallel CRC having there the advantages and the disadvantages for each technique. For this the table is based on the architecture can be required to be pre-calculated LUT, so, that it will not used for the generalized CRC, fast CRC update techniques are used to required buffer to store the old CRC and data. In unfolding the architecture can be increases the no. of iteration bound. Then the F matrix based on the architecture can be more simple and then the low complex. Below this algorithm and its' implementation is given.

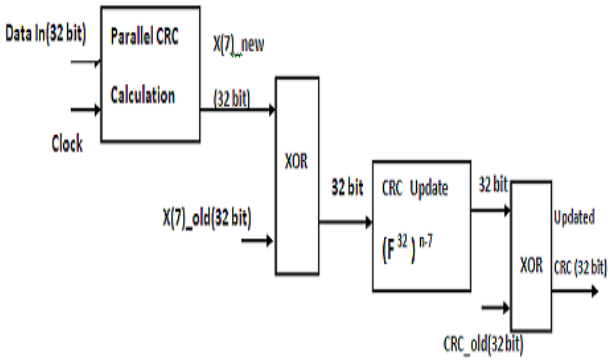

Fig3. Fast CRC update architecture [7]

A. Algorithm for F matrix based architecture.

Algorithm and Parallel architecture for the CRC generation based on the F matrix is discussed in this section. As we can be shown in fig2. It is the basic algorithm for F matrix is based on the parallel CRC Generation

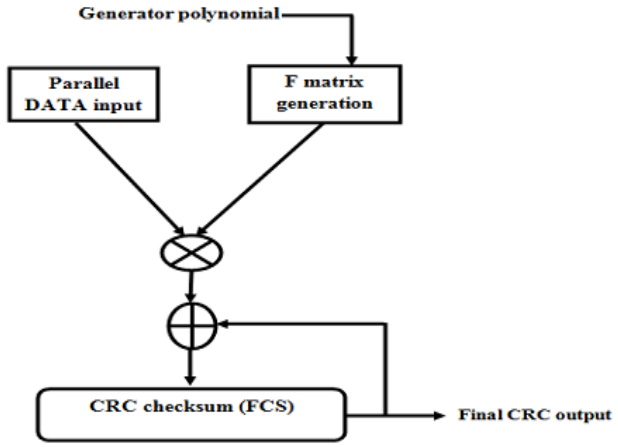

Fig4. The Algorithms for F matrix based architecture

Parallel data input and each element of the F matrix, which is generated and from given generator polynomial is the anded, result of that will be xoring with present state of the CRC checksum. Then the final result are generated after (k+ m) /w cycle.

B. F Matrix Generation

The F matrix is generated from the generator polynomial as per (2).

$$F = \begin{bmatrix} P_{m-1} & 1 & 0 & 0 & 0 \\ P_{m-2} & 0 & 1 & 0 & 0 \\ P_{m-3} & 0 & 0 & 1 & 0 \\ P_{m-4} & 0 & 0 & 0 & 1 \\ .. & .. & .. & .. & .. \\ P_0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2)$$

Where,{p0……pm-1}is the generator polynomial. For example, then the generator polynomial for CRC4 is {1, 0, 0, 1, 1} and w bits are parallely processed.

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

Here w=m=4, for that F w matrix calculated as follow.

$$F^4 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (4)$$

C. Parallel architecture

Parallel architecture is based on the F matrix that can be illustrated in fig. 2. As shown in fig.2, d is the data that is parallel processed (i.e 32bit), ' X is next state, X is current state (generated CRC), F(i)(j) is the Ith row and jth column of F w matrix. If X = [xm-1…..x1x0]T is utilized to denote the state of the shift registers, in this linear system theory, the state equation for the LFSRs can be expressed in modular 2 arithmetic as follow.

$$X_{I'} = (P_0 \otimes X_{m-1}) \otimes X_{i-1} \quad (5)$$

Where, X(i) represents the ith state of the registers, X(i +) denotes the (i + 1) th state of the registers, d denotes the one-bit shift-in serial input. F is an (m x m) matrix and G is a (1 x m) matrix

$$G\ [0\,0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 0\,1]T \quad (6)$$

Furthermore, if the F and G are substituted by this Equations (4) and (5), we can rewrite this equation (4) in the matrix form as:

$$\begin{bmatrix} X'm_{-1} \\ X'm_{-2} \\ \vdots \\ X'0 \end{bmatrix} = \begin{bmatrix} 1\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \\ 1\,0\,0\,0 \end{bmatrix} \cdot \begin{bmatrix} X_{m-1} \\ X_{m-2} \\ \vdots \\ X_0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \cdot d \quad (7)$$

Finally, equation (6) can be written in matrix form as

$$X' = F^w \otimes X (+) d \quad (8)$$

Equation (7) is can be illustrated in fig. 2. If the w bits are parallel and then the processed, then CRC will be generated after (k+ m)/w Equation (8) can be expanded for CRC4 given below.

$$X3' = X_2 (+) X_1 (+) X_0 (+) d_3$$

$$X2' = X_3 (+) X_2 (+) d_2$$

$$X1' = X_3 (+) X_2 (+) X_1 (+) d_1$$

$$X0' = X_3 (+) X_2 (+) X_1 (+) X_0 (+) d_0 \quad (9)$$

Fig.5.demonstrates an example of parallel CRC calculation with multiple input bits w = m = 4. Then the dividend is divided into the three 4-bit fields, that can be acting as the parallel input vectors D(0),D(1),D(2), and

respectively. Then the initial state is X(0) = [0 0 0 0]T. From Equation (8), we have,

$$X(4) = F^4 \otimes X(0) \text{ (+) } D(0)$$

$$X(8) = F^4 \otimes X(4) \text{ (+) } D(1)$$

$$X(12) = F^4 \otimes X(8) \text{ (+) } D(2) \quad (10)$$

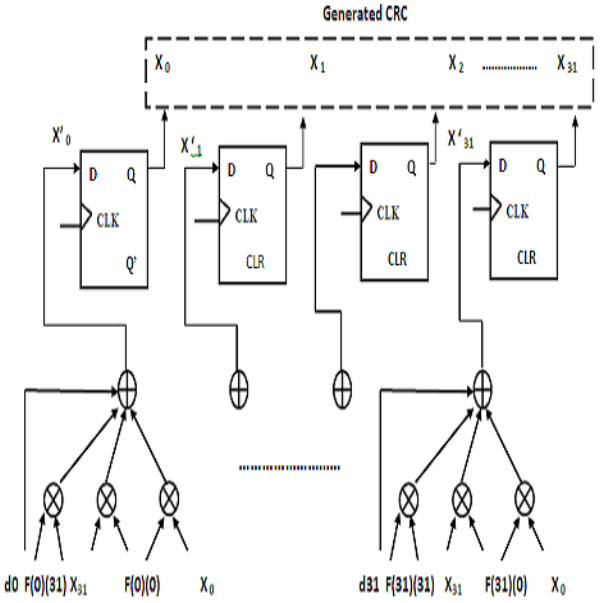

Fig5. Parallel calculation of CRC-32 for 32bit [5].

Property of the F w matrix and the previously mentioned fact that Equation (8) can be regarded as a recursive calculation of the next state X' by matrix F w, current state X and then the parallel input D, it make the 32-bit parallel input vector it is more suitable for any length of messages besides the multiples of the 32 bits. Remember that the length of this message is byte-based. If then the length of this message is not in then the multiple of 32, after that the sequence of 32-bit parallel calculation, and then the final remaining number of bits of the message could be 8; 16,or 24. For all these situations, and an additional parallel calculation w = 8; 16; 24 is needed by choosing the corresponding Fw. Since F w can be easily to derived from F 32, then the calculation can be performed by using this Equation (8) within the same circuit as 32-bit parallel calculation, then only difference is the F w matrix.

If the length of the message is not then the multiple number of the parallel processing bits w = 4 i.e. data bit is 11011101011. Then this the last two more bits (D (3)) need to be calculated after getting X (12). Therefore, F 2 must be obtained from the matrix F4, and the extra two bits are stored at the lower significant bits of the input vector D. Equation (8) that can be applied to calculate the final state X (14), which is the CRC code. Therefore, only an extra cycle is needed for calculating the extra bits if the data message length is not the multiple number of w, then the parallel processing bits. It is the worth to notice that in CRC-32 algorithm, then the initial state of the shift registers is preset to all `1's. That therefore, X(0) = 0xFFFF. However, the initial state X (0) does not affect that the correctness of this design. In order to obtain for the better understanding, the initial state X (0) is still set to 0x0000 when the circuit is implemented.

IV. PROPOSED PARALLEL ARCHITECTURE

In this the proposed architecture w= 64 bits are parallely processed and order of generator polynomial is m= 32 as shown in fig. 3. As we can discussed in this section 3, if 32 bits are processed by parallely then CRC-32 will be generated after m)/w (k+ cycles. If we can increase the number of bits to be processed parallely, and then the number of cycles required to calculate the CRC can be reduced. Then the Proposed architecture can be realized by below equation.

$$\text{X temp} = F^W \otimes \text{D } (0 \text{ to } 31) \text{ (+) } D(32 \text{ to } 63)$$

$$X' = F^W \otimes \text{X temp} \quad (11)$$

Where,

D (0 to 31) =first 32 bits of parallel data inputs

D (0 to 63) = next 32 bits of parallel data inputs

X'=next state

X=present state

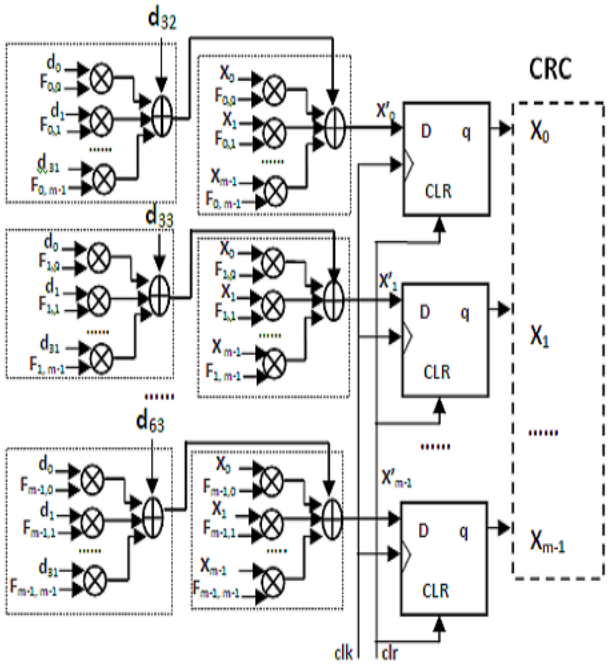

Fig6. Block diagram of 64-bit parallel calculation of CRC-32.

When this proposed architecture di is the parallel input and F(i) (j) is the element of F 32 matrix that can be located at ith row and jth column. As we can shown in figure then the 3 input data bits d0….d31anded with each row of F W matrix and result will be xored individually with d32, d33…….d63. Then each xored results is then xored with in the ' X(i) in terms of CRC32. Finally X will be the CRC generated after m)/w (k+ cycle, where w=64.

V. RESULT AND ANALYSIS

Then this proposed architecture is the synthesized in the Xilinx-9.2i and simulated in the Xilinx ISE Simulator, which is required half cycle then the previous 32bit design[1][5]. In this the programming in VHDL by specifying only generator polynomial, it is directly gives F matrix that can be useful for parallel CRC generation that is not available in this the previous methods [1][5][6]. Hardware utilization is compared in this table I for different approaches for the different parameter are like LUT, CPD and cycles.

TABLE I. Comparison of LUT, Clock cycle and CPD

CRC w parallel bits	Clock cycles	LUTs	CPD
CRC32 w=32bit[1]	17	162	7.3ns
CRC32 w=32bit[8]	17	220	30.5ns
CRC32 w=64bit (Proposed)	9	370	5.7ns

From this table we can observe that, the architecture proposed by [1][8] it can be require for these 17 clock cycles to generate CRC as per equation (k+ m)/w and for proposed architecture can be required only 9 cycles, CPD for this proposed architecture is less than the architecture [1][8],only the disadvantage for proposed architecture is the no. of LUT get increased, so that the area also get increase .

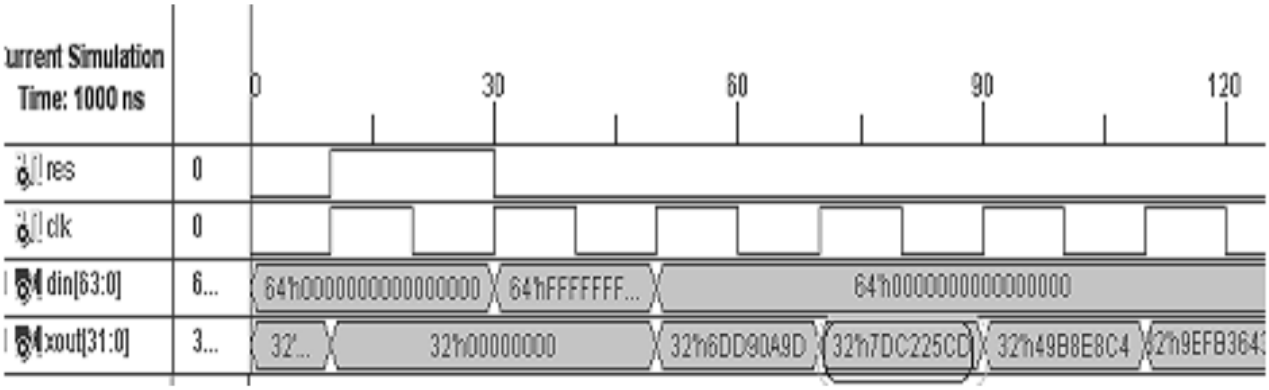

Fig7. Generated waveform for proposed 64bit architecture

When the proposed CRC-32 architecture with the 64bit parallel bit simulated in the Xilinx 9.2i ISE simulator. When the input data bit to system is FFFFFFFFFFFFFFFF (64 bit). The final result can be obtain after m)/w (k+ cycle for 32-bit residual will be 7DC225CD (hexadecimal form).

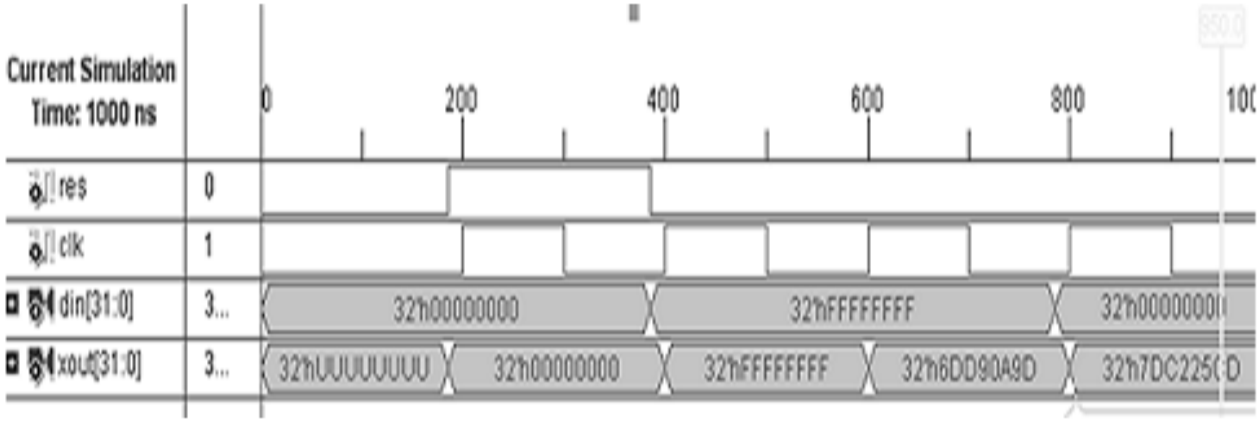

Fig8. Generated waveform for proposed 32bit architecture

VI. CONCLUSION

The32bit parallel architecture can be required 17 ((k + m)/w) clock cycles for 64 byte data [1] [5]. When this the Proposed design (64bit) is also can be required only 9 cycles to generate the CRC with the same order of generator polynomial. So, it is drastically reduces for the computation time to 50% and that the same time increases the throughput. Pre-calculation of F matrix is not required in proposed architecture. When this is the compact and easy method for fast CRC generation.

VII. REFERENCES

[1] Hitesh H. Mathukiya, Naresh M. Patel " A Novel Approach for Parallel CRC generation for high speed applications, International Conference on Communication Systems and Network Technologies, Oct.2012

[2] Albertengo, G.; Sisto, R.; , "Parallel CRC generation," Micro, IEEE , vol.10, no.5, pp.63-71,Oct1990.

[3] M.D.Shieh et al., "A Systematic Approach for Parallel CRC Computations," Journal of Information Science and Engineering, May 2001.

[4] Braun, F.; Waldvogel, M.;"Fast incremental CRC updates for IP over ATM networks," High Performance Switching and Routing, 2001 IEEE Workshop on , vol., no., pp.48-52, 2001

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

[6] S.R. Ruckmani, P. Anbalagan, " High Speed cyclic Redundancy Check for USB" Reasearch Scholar, Department of Electrical Engineering, Coimbatore Institute of Technology, Coimbatore-641014, DSP Journal, Volume 6, Issue 1, September, 2006.

[7] Yan Sun; Min Sik Kim; , "A Pipelined CRC Calculation Using Lookup Tables," Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE , vol., no., pp.1-2, 9-12 Jan. 2010

[8] Sprachmann, M.;, "Automatic generation of parallel CRC circuits," Design & Test of Computers, IEEE , vol.18, no.3, pp.108-114, May 2001.

Author's Profile:

P Abdul Rasool,
II Mtech,
Kottam College of Engineering,
Chinnatekur
Kurnool.
E-mail:abdul.rasool8@gmail.com

N Mohan Raju,
Assistant Professor,
Kottam College of Engineering,
Chinnatekur, Kurnool,
E-mail:Mohanrajunese@gmail.Com