A Novel Approach for Parallel CRC generation for high speed application

Hitesh H. Mathukiya
Electronics and communication Department,
Sarvajanik College of Engineering and Technology,
Surat, India
hitesh.mathukiya@gmail.com

Naresh M. Patel
Electronics and communication Department,
Sarvjanik College of Engineering and Technology,
Surat, India
naresh.patel@scet.ac.in

***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 is transmitted. With challenging the speed of transmitting data, to synchronize with speed, it's necessary to increase speed of CRC generation. Starting from the serial architecture identified a recursive formula from which parallel design is derived. This paper presents 64 bits parallel CRC architecture based on F matrix with order of generator polynomial is 32. Proposed design is hardware efficient and required 50% less cycles to generate CRC with same order of generator polynomial. The whole design is functionally verified using Xilinx ISE Simulator.**

Keywords- Cyclic Redundancy Check, Parallel CRC calculation, Linear Feedback Shift Register, LFSR, F matrix

I. INTRODUCTION

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

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

This paper starts with the introduction of serial CRC generation based on LFSR. F matrix based parallel architecture for 32 bits and 64 bits are described in section 3 and 4. Finally, simulation results are shown in section 5 and concluded in section 6.

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. Binary divisions generally can be performed by a sequence of shifts and subtractions. In modulo 2 arithmetic the addition and subtraction are equivalent to bitwise XORs (denoted by "$\oplus$" in this paper) and multiplication is equivalent to AND (denoted by "$\otimes$" in this paper). Figure 1 illustrates the basic architecture of LFSRs for serial CRC calculation.

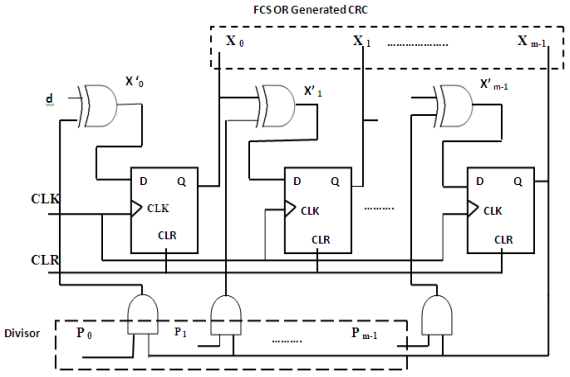

Figure 1. Basic LFSR Architecture [2]

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

$$\left.\begin{aligned} X_0' &= (P_0 \otimes X_{m-1}) \oplus d \\ X_i' &= (P_0 \otimes X_{m-1}) \oplus X_{i-1} \end{aligned}\right\} \quad (1)$$

The generator polynomial for CRC-32 is as follows

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

We can extract the coefficients of G(x) and represent it in binary form as

$P = \{p_{32}, p_{31}, \ldots\ldots\ldots, p_0\}$
$P = \{10000010011000001000111011011011 1\}$

978-0-7695-4692-6/12 $26.00 © 2012 IEEE
DOI 10.1109/CSNT.2012.131

IEEE computer society

Frame Check sequence (FCS) will be generated after (k+m) cycle, where k indicates number of data bit and m indicates the order of generator polynomial. For 32 bits serial CRC if order of generator polynomial is 32 then 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 input, LUT3, LUT2, and LUT1. LUT3 contains CRC values for the input followed by 12 bytes of zeros, LUT2 8 bytes, and LUT4 4 bytes. Basically this algorithm it can be obtain higher throughput. The main problem it with pre-calculating CRC and store it in LUT so, every time required to change LUT when changing the polynomial. Pipelining algorithm used to reducing critical path by adding the delay element.

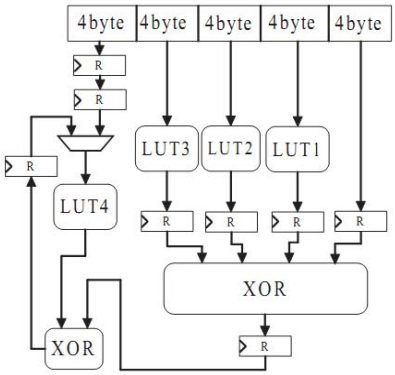

Figure .2 LUT based architecture [5]

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

In fast CRC update technique not required to calculate CRC each time for all the data bits, instead of that calculating CRC for only those bits that are change.

There are different approaches to generate the parallel CRC having advantages and disadvantages for each technique. Table based architecture required pre-calculated LUT, so, it will not used for generalized CRC, fast CRC update technique required buffer to store the old CRC and data. In unfolding architecture increases the no. of iteration bound. The F matrix based architecture more simple and low complex. Below algorithm and its' implementation is given.

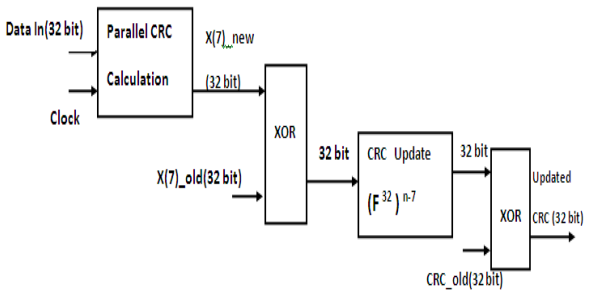

Figure .3 Fast CRC update architecture [7]

A. *Algorithm for F matrix based architecture.*

Algorithm and Parallel architecture for CRC generation based on F matrix is discussed in this section. As shown in fig. 2 it is basic algorithm for F matrix based parallel CRC generation

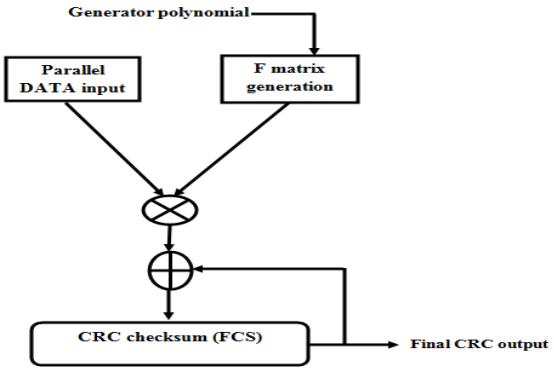

Figure 4: Algorithms for F matrix based architecture

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

B. *F Matrix Generation*

F matrix is generated from 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, $\{p_0 \ldots\ldots p_{m-1}\}$ is generator polynomial. For example, 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 based on F matrix illustrated in fig. 2. As shown in fig. 2, d is data that is parallel processed (i.e 32bit), X' is next state, X is current state (generated CRC), F(i)(j) is the i^{th} row and j^{th} column of F^w matrix. If $X = [x_{m-1} \ldots x_1 \ x_0]^T$ is utilized to denote the state of the shift registers, in linear system theory, the state equation for LFSRs can be expressed in modular 2 arithmetic as follow.

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

Where, X(i) represents the i^{th} state of the registers, X(i + 1) 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.

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

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

$$\begin{bmatrix} X'_{m-1} \\ X'_{m-2} \\ . \\ X'_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_{m-1} \\ X_{m-2} \\ . \\ X_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ . \\ 1 \end{bmatrix} .d \quad (7)$$

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

$$X' = F^W \otimes X \oplus d \quad (8)$$

Equation (7) is illustrated in fig. 2. If w bits are parallel processed, then CRC will be generated after $(k+m)/w$.

Equation (8) can be expanded for CRC4 given below.

$$\left.\begin{aligned} X_3' &= X_2 \oplus X_1 \oplus X_0 \oplus d_3 \\ X_2' &= X_3 \oplus X_2 \oplus d_2 \end{aligned}\right\} \quad (9)$$

$$\left.\begin{aligned} X_1' &= X_3 \oplus X_2 \oplus X_1 \oplus d_1 \\ X_0' &= X_3 \oplus X_2 \oplus X_1 \oplus X_0 \oplus d_0 \end{aligned}\right\}$$

Fig.5.demonstrates an example of parallel CRC calculation with multiple input bits w = m = 4. The dividend is divided into three 4-bit fields, acting as the parallel input vectors D(0),D(1),D(2), respectively. The initial state is $X(0) = [0\ 0\ 0\ 0]^T$. From Equation (8), we have,

$$\left.\begin{aligned} X(4) &= F^4 \otimes X(0) \oplus D(0) \\ X(8) &= F^4 \otimes X(4) \oplus D(1) \\ X(12) &= F^4 \otimes X(8) \oplus D(2) \end{aligned}\right\} \quad (10)$$

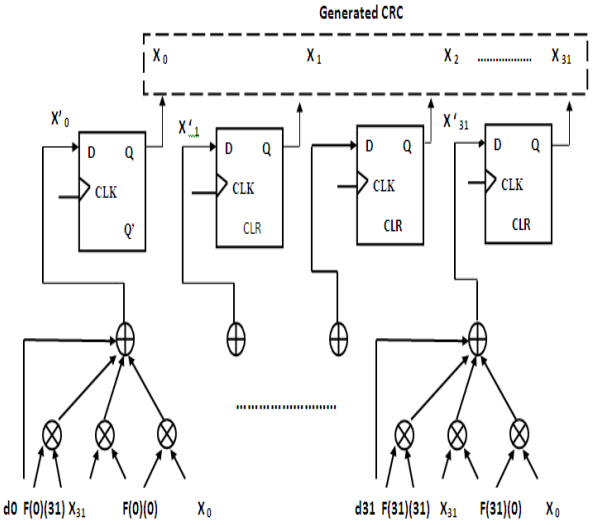

Figure .5 Parallel calculation of CRC-32 for 32bit [5].

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 parallel input D, make the 32-bit parallel input vector suitable for any length of messages besides the multiple of 32 bits. Remember that the length of the message is byte-based. If the length of message is not the multiple of 32,after a sequence of 32-bit parallel calculation, the final remaining number of bits of the message could be 8; 16,or 24. For all these situations, an additional parallel calculation w = 8; 16; 24 is needed by choosing the corresponding F^w. Since F^w can be easily derived from F^{32}, the calculation can be performed using Equation (8) within the same circuit as 32-bit parallel calculation, the only difference is the F^w matrix. If the length of the message is not the multiple of the number of parallel processing bits w = 4 i.e. data bit is 11011101011. Then last two more bits (D (3)) need to be calculated after getting X (12). Therefore, F^2 must be obtained from matrix F4, and the extra two bits are stored at the lower significant bits of the input vector D. Equation (8) can then 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 of w, the number of parallel processing bits. It is worth to notice that in CRC-32 algorithm, the initial state of the shift registers is preset to all `1's. Therefore, X(0) = 0xFFFF. However, the initial state X (0) does not affect the correctness of the design. In order for better understanding, the initial state X (0) is still set to 0x0000 when the circuit is implemented.

IV. PROPOSED PARALLEL ARCHITECTURE

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

$$\left.\begin{aligned} Xtemp &= F^{W} \otimes D(0to31) \oplus D(32to63) \\ X' &= F^{W} \otimes X \oplus Xtemp \end{aligned}\right\} \quad (11)$$

Where,

D (0 to 31) =first 32 bits of parallel data input
D (0 to 63) = next 32 bits of parallel data input
X'=next state
X=present state

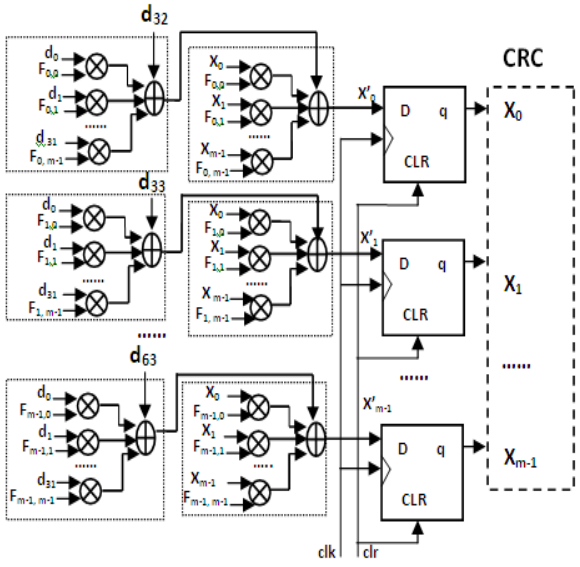

Figure 6. Block diagram of 64-bit parallel calculation of CRC-32.

In proposed architecture d_i is the parallel input and F(i)(j) is the element of F^{32} matrix located at i^{th} row and j^{th} column. As shown in figure 3 input data bits $d_0 \ldots d_{31}$ anded with each row of F^{W} matrix and result will be xored individually with $d_{32}, d_{33} \ldots\ldots d_{63}$. Then each xored result is then xored with the X' (i) term of CRC32. Finally X will be the CRC generated after $(k+m)/w$ cycle, where w=64.

V. RESULT AND ANALYSIS

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

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

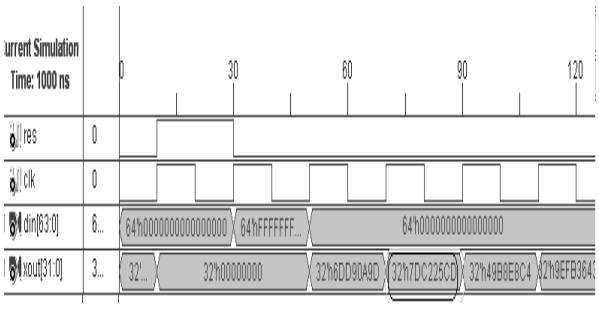

Figure 7. Generated waveform for proposed 64bit architecture

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

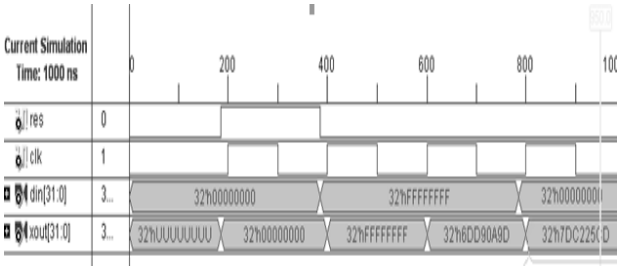

Figure 8. Generated waveform for proposed 32bit architecture

VI. CONCLUSION

32bit parallel architecture required 17 ((k + m)/w) clock cycles for 64 byte data[1][5]. Proposed design (64bit) required only 9 cycles to generate CRC with same order of generator polynomial. So, it drastically reduces computation time to 50% and same time increases the throughput. Pre-calculation of F matrix is not required in proposed architecture. Hence, this is compact and easy method for fast CRC generation.

REFERENCES

[1] Campobello, G.; Patane, G.; Russo, M.; "Parallel CRC realization," *Computers, IEEE Transactions on* , vol.52, no.10, pp. 1312- 1319, Oct.2003

[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