ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Novel Approach for Parallel CRC Generation for High Speed Application

K.Lakshmi
M.Tech Student,
Department of ECE,
KITS for Women's, kodad, T.S, India.

MS.B.Jyothirmayee
Associate Professor,
Department of ECE,
KITS for Women's, kodad, T.S, India.

ABSTRACT:

CRC is playing a main role in the networking environment to detect the errors. With challenging the speed of transmitting data to synchronize with speed, it is necessary to increase speed of CRC generation. Most engineers are familiar with the cyclic redundancy check (CRC). Many know that it is used in communication protocols to detect bit errors and that it is essentially a remainder of the modulo-2long division operation. As a vital method for dealing with data errors usually the hardware implementation of CRC computations is based on the linear feedback shift registers (LFSRs), which handle the data in a serial way. The serial calculation of the CRC codes cannot achieve a high throughput. In constant parallel CRC calculation can significantly increase the throughput of CRC computations. Types of CRCs are used in applications like CRC-16BISYNC protocols, CRC32 in Ethernet for error detection, CRC8 in ATM, CRC-CCITT in X-25 set of rule, disk storage, XMODEM and SDLC. This paper presents 64 bits parallel CRC architecture. The whole design is functionally verified using Xilinx ISE Simulator.

Keywords:

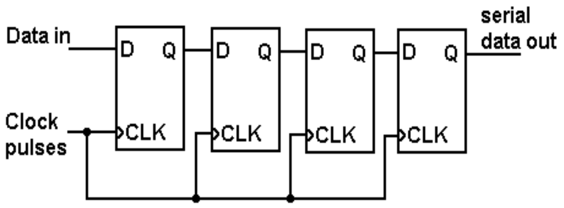
Redundancy check, Parallel CRC calculation, Shift register, Error control coding.

1. INTRODUCTION:

Cyclic Redundancy Check (CRC) technology can be used to test the integrity of the process in data transmission or data compression. It has been widely used in the communication network and the data storage technology, etc. The receiver calculates the data of received from transmitter with the same CRC algorithm. If the value calculated equals to checking code in frame, it shows that the process of transmission or compression is successful. Classic CRC algorithm is realized through the structure of linear feedback registers. Because it is a serial coding method, it is intuitive and easy to realize. But the speed of calculate is very low, according to the classic CRC algorithm. It is not suitable to apply in high speed data transmission. Parallel CRC algorithm can meet the demand of the high-speed data transmission. The general method of the parallel CRC is realized by software or hardware. Because speed of the software calculation is slow, the algorithm has been used in much communication equipment by hardware for high speed According to the thoughts of traditional serial CRC algorithm, parallel CRC PIPELINE ALGORITHM based on MATRIX is referred for high-speed coding.

The algorithm can reduce the delay time of circuit and improve the throughput rate of data. CRC can generate in two ways: A. Serial CRC generation. B. Parallel CRC generation. 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. So, to over that problem we are moving to Parallel CRC generation. In this we are discussing about 64-bit parallel CRC generation.32-bit parallel CRC generates gigabits per second, but it will not suited for high speed applications like Ethernet. A. SERIAL CRC GENERATION In this CRC checking is done serially. The data input will be single (binary) and every clock pulse the data input will be one. There will some delay present between the consecutive data inputs and the output will be zero if the data will be encoded with same CRC value otherwise it shows non-zero value.

By this form we will conclude where the data is accurate or corrupted. The data is serially processed and the polynomial is XORed with the data input, that will given to the D flipflop (output same as the input) final CRC is generated as serially.



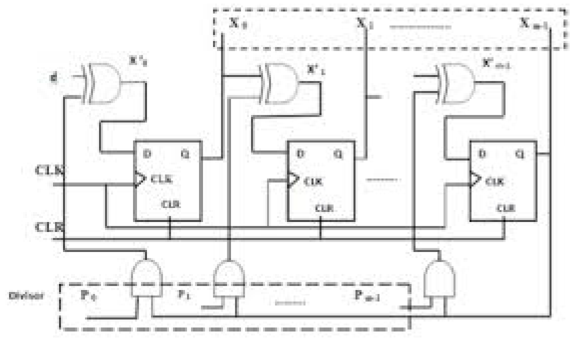
The diagram shows four flip-flops connected to form a input output shift register. At arrival of a clock pulse, data at the D input of each flip-flop is transferred to its Q output. First, the contents of the register can be set to zero by means of the CLEAR line. If 1 is applied to the input of the first flip-flop, then the arrival of the first clock pulse, this 1 is transferred to the output of flip-flop 1 (input of flip-flop 2). After 4 clock pulses 1 will be at the output of flip-flop 4. In this way, a four bit number can be stored in the register. After 4 more clock pulses, this data will be shifted out of the register.

2. METHODOLOGY:

CRC can generate in two ways: 1. Serial CRC generation 2. Parallel CRC generation 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. So, to over that problem we are moving to Parallel CRC generation. In this we are discussing about 64-bit parallel CRC generation.32-bit parallel CRC generates gigabits per second, but it will not suited for high speed applications like Ethernet [2].

3. IMPLEMENTATION:

Serial CRC Here„d“ represents the data, {Pm-1--------P1, P0} represents the generated polynomial, X is the present state and X ′ is the next state.



Operation: Here CRC checking is done serially. The data input will be single (binary) and every clock pulse the data input will be one. There will some delay present between the consecutive data inputs and the output will be zero if the data will be encoded with same CRC value otherwise it shows non-zero value. By this form we will conclude where the data is accurate or corrupted [3]. The data is serially processed; the polynomial is XORed with the data input, that will given to the D flip-flop (output same as the input) final CRC is generated as serially. Working of basic LFSR architecture is expressed in terms of following equations.

$$X_0' = (P_0 \otimes X_{m-1}) \oplus d$$
$$X_i' = (P_0 \otimes X_{m-1}) \oplus X_{i-1}$$

The generator polynomial for CRC-32 is as follows
G(x) = x32+ x28+ x23+ x20+ x18+ x15+ x11+ x10+ x8 + x6 + x3 + x2 + x1 + x0 We can extract the coefficients of G(x) and represent it in binary form as
P = {p31, p30………………. p0}
P= {10000010011000001000111011011011}.
Parallel CRC Every modern communication protocol uses one or more error-detection algorithms. CRC is by far the most popular. CRC properties are defined by the generator polynomial length and coefficients. The protocol specification usually defines CRC in hex or polynomial notation [1].

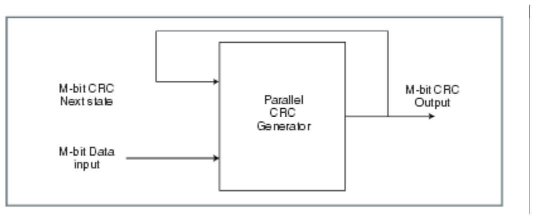
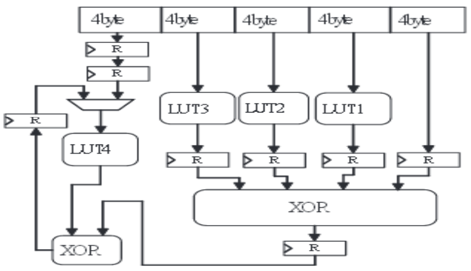


Figure3: This is a parallel CRC block. The next state CRC output is a function of the current state CRC and the data. There different techniques are:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

1. Table based algorithm
2. Fast CRC Update
3. F matrix parallel CRC generation
4. Unfolding, retiming and pipelining algorithm

Table based algorithm

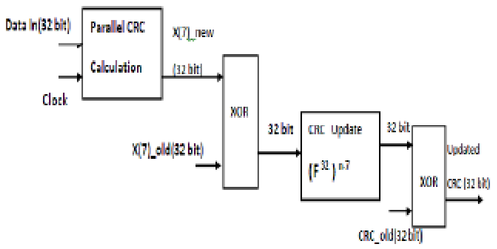


The pipelined architecture in Fig. 4 has five blocks as input; four of them are used to read four new blocks from the message in each iteration. They are converted into CRC using lookup tables: LUT3, LUT2, andLUT1.LUT3 contain CRC values for the input followed by 12bytes of zeros, LUT2 8 bytes, and LUT1 4 bytes. Note that the rightmost block does not need any lookup table. It is because this architecture assumes CRC-32, the most popular CRC, and 4-byte blocks. If the length of a binary string is smaller than the degree of the CRC generator, its CRC value is the string itself. Since the rightmost block corresponds to A4, it does not have any following zero and thus its CRC is the block itself.

The results are combined using XOR, and then it is combined with the output of LUT4, the CRC of the value from the previous iteration with 16 bytes of zeros concatenated. In order to shorten the critical path, we introduce another stage called the pre-XOR stage right before the fur-input XOR gate. This makes the algorithm more scalable because more blocks can be added without increasing the critical path of the pipeline. With the pre-XOR stage, the critical path is the delay of LUT4 and a two-input XOR gate and the throughput is 16 bytes per cycle [4]. Drawback: Table based architecture required pre-calculated LUT, so, it will not used for generalized CRC.

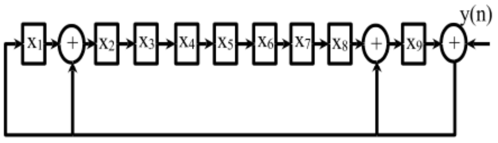
Fast CRC :

Our fast CRC update method is extended from the parallel CRC calculation and can adapt to a number of bits processed in parallel. The method can also reduce the data traffic and power consumption of the CRC calculation unit[6].



Drawback:Fast CRC update technique required buffer to store the old CRC and data.

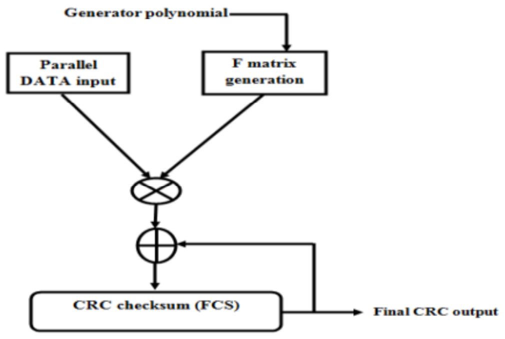
3. Unfolding, retiming and pipelining algorithm Iteration bound is defined as the maximum of all the loop bounds. Loop bound is defined as t/w, where„t"is the computation time of the loop and „w" is the no. of delay elements in the loop. The largest iteration bound of a general serial CRC architecture is also 2TXOR [5].



Drawback: In unfolding architecture increases the no. of iteration bound.

4. F-matrix parallel CRC generation:

F-matrix follows the algorithm as:



Parallely data is processed; it is ANDed with the F-matrix generation from the generated polynomial. Result of that will XORed with present state CRC checksum. The final result will obtained after (k+m)/w cycles.

Generation of F-matrix:

F-matrix generated from the generated polynomial, matrix form can be represented as:

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

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

$$F = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad F^4 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

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

Parallel architecture: Parallel architecture based on F-matrix"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 ith row and jth column of FW matrix. If X = [xm1 …..x1 x0]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

Xi′= (P0 Xm-1)d

Where, X(i) represents the state of the registers, X(i + 1) denotes the 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 --------0 1]T

This can be represented in the matrix form as

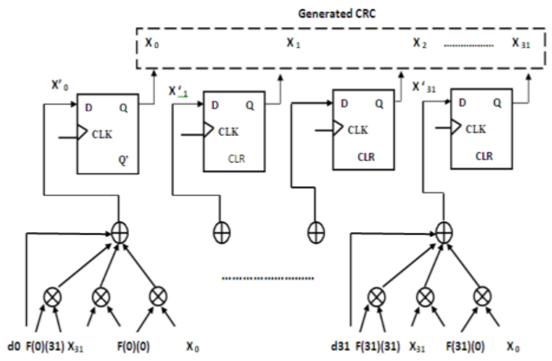
$$\begin{bmatrix} X'_{m-1} \\ X'_{m-2} \\ . \\ X'_0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} X_{m-1} \\ X_{m-2} \\ . \\ X_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ . \\ 1 \end{bmatrix} .d$$

Finally it can rewritten as

$$X' = F^W \otimes X \oplus d$$

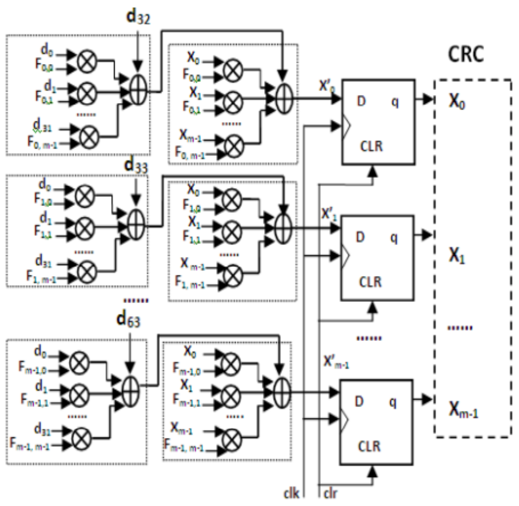
If W-bits are parallel processed,the result of the CRC will generated after (k+m)/w cycles.

64-bit PARALLEL ARCHITECTURE



In proposed architecture w= 64 bits are parallely processed and order of generator polynomial is m= 32 as shown in fig. 8, 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 realizedby below equation.

5. Concluding Remarks:

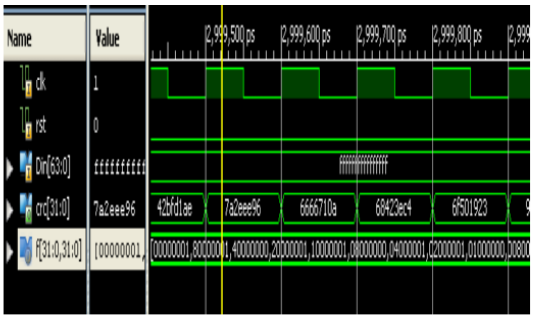


Generally when high-speed data transmission is required serial implementation is not preferred because of slow throughput. So parallel implementation is preferred which takes less time. CRC-32 requires 17 clock cycles to transmit 64bytes of data. But CRC-64 needs 9 clock cycles to transmit the same data

6. Experimental Results:

Xtemp = FWD (0to31) D(32to63) X' = FW X Xtemp

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



CRC-64

ACKNOWLEDGMENTS:

I am K.LAKSHMI and would like to thank the publishers, researchers for making their resources material available. I am greatly thankful to Assistant Prof: Miss. Miss.B.Jyothirmayee for their guidance. We also thank the college authorities, PG coordinator and Principal for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES:

[1] Hitesh H. Mathukiya; Naresh M. Patel; "A Novel Approach for Parallel CRC generation for high speed application," 2012 IEEE DOI.

[2] Y. Sun and M. S. Kim, "A table-based algorithm for pipelined CRC calculation," in Proceedings of IEEE International Conference on Communications, May 2010.'

[3] G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," IEEE Transactions on Computers, Oct. 2003.

[4] C. Cheng and K. K. Parhi, "High-speed parallel CRC implementation based on unfolding, pipelining, and retiming," IEEE Transactions on Circuits and Systems, Oct. 2006.

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

[6] W. Jiang and V. K. Prasanna, "A memory-balanced linear pipeline architecture for trie-based IP lookup," 2007.

Author's Detais:

Ms. K.Lakshmi. MTech student, in M.Tech Student, Dept of ECE in KITS for women's,kodad, T.S, India

Ms.B.Jyothirmayee working as a Assistant at ECE in KITS for women's, kodad, T.S, IndiaJNTUH Hyderabad. she has 3 years of UG/PG Teaching Experience