Research Paper ENGINEERING

# Implementation of 'n' bit parallel CRC using Unfolding, Retiming, Pipelining for high speed applications

**C.Arun Kumar Chowdary** | Student, ANANTHA LAKSHMI INSTITUTE OF TECH & SCIENCE Affiliation to JNTUA,

**C. Kumara Narayana Swamy** | Associate Professor, ANANTHA LAKSHMI INSTITUTE OF TECH & SCIENCE Affiliation to JNTUA

**ABSTRACT**

Presently in networking environment high speed data transmission is very crucial. Cyclic Redundancy Check (CRC) is essential method for detecting error when the data is transmitted because the errors are introduced over physical links. In this paper, we present a fast Cyclic Redundancy Check (CRC) algorithm that performs CRC computation for an arbitrary length of input message bits. This paper purposes 'n' bit parallel CRC architecture based on LFSR, Unfolding, Retiming, Pipelining order of generator polynomial is 32 and showed CRC -'n' is having high throughput and less power consumption compared to CRC-64 parallel architecture through Xilinx simulator

**KEYWORDS** CRC, LFSR, Unfolding, Retiming, Pipelining

## INTRODUCTION

CRC is most widely used to detect errors in data communication, storage devices and data compression. Whenever high speed transmission is required, the general serial implementation cannot meet the speed requirement so on this basis we move on to parallel CRC computation.

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. However, that is still not enough for high speed application such as Ethernet networks. A possible solution is to process more bits in parallel.

Parallel processing is a very efficient way to increase the throughput rate. Although parallel processing increases the number of message bits that can be processed in one clock cycle, it can also lead to a long critical path (CP).

Thus, the increase of throughput rate that is achieved by parallel processing will be reduced by the decrease of circuit speed. Another issue is the increase of hardware cost caused by parallel processing, which needs to be controlled. The parallel CRC algorithm in processes an m -bit message in (m+k)/L clock cycles, where k is the order of the generator polynomial and L is the level of parallelism. However, m message bits can be processed in m/L clock cycles. High speed architectures for parallel long encoders are based on the multiplication and division computations on generator polynomial are efficient in terms of speeding up the parallel linear feedback shift register (LFSR) structures.

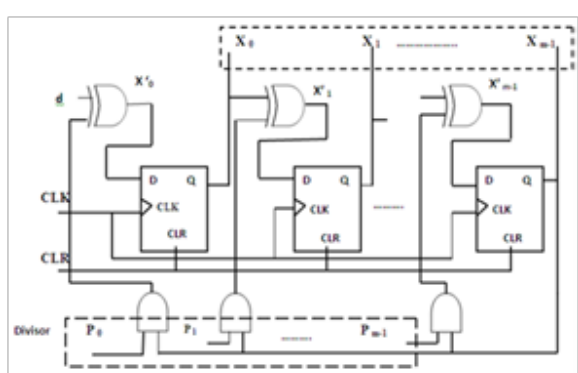
The proposed design achieves shorter critical path for parallel CRC circuits leading to high processing speed than commonly used generator polynomial. The proposed design starts from LFSR, which is generally used for serial CRC. An unfolding algorithm is used to realize parallel processing. However, direct application of unfolding may lead to a parallel CRC circuit with long iteration bound, which is the lowest achievable CP.

Two novel look-ahead pipelining methods are developed to reduce the iteration bound of the original serial LFSR CRC structures; then, the unfolding algorithm is applied to obtain a parallel CRC structure with low iteration bound. The retiming algorithm is then applied to obtain the achievable lowest CP.

## II. SERIAL CRC

The implementation of CRC check generation circuit can be done with the use of linear feedback circuit. Basic methodology for generating Serial CRC is based on the Linear Feedback Shift Register (LFSR). The main operation of Serial CRC is similar as the binary division. Generally binary division is executed by a sequence of shifts and substractions.

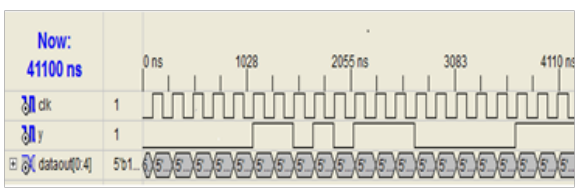
**Figure 2.1.(a) shows the Serial CRC generation using LFSR.**




In this CRC checking is done Serially. The input data will be single given in binary and every clock pulse the data input will be one. We observe here some delay present between the consecutive data inputs and output will be Zero if the data will be encoded with same crc value otherwise it shows non-zero value. By all these we conclude where the data is accurate or corrupted.

The data is Serially processed, the polynomial is XORed with the input data,that will given to the D flip-flop where we observe that output is same as input and final CRC is generated as Serially.


**RTL schematic of LFSR**

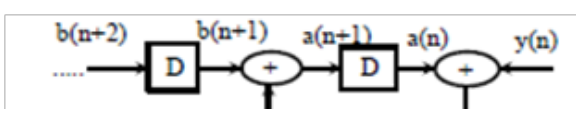



**Waveform of LFSR for 'n' bit input**

**III. Parallel CRC**

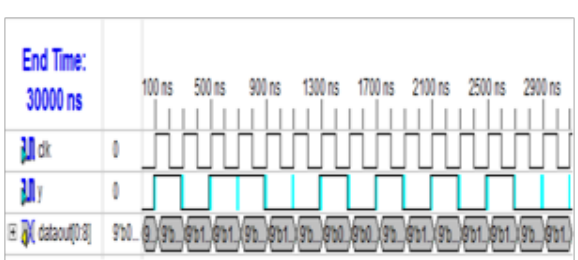
**III.1 Pipelining algorithm**

It reduce the extra critical path either to increase the clock frequency or sample speed or to reduce power consumption at the same speed. It is done using a look-ahead pipelining algorithm to reduce the iteration bound. 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.




**Pipelining to reduce iteration bound**


**RTL schematic**

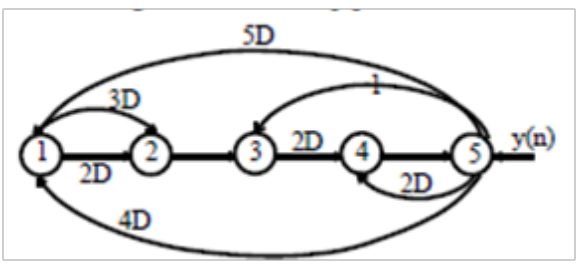



**Waveform of pipelining for 'n'bit input**

**3.2 Retiming algorithm**

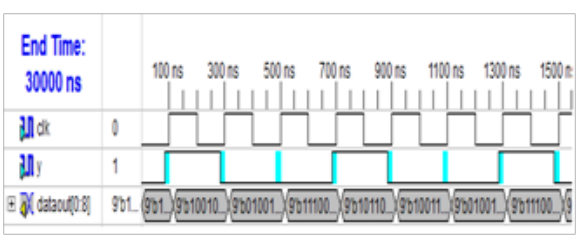
Retiming is used to change the locations of delay elements in a circuit without affecting the input/output characteristics of the circuit. It reduces the critical path of the system by not altering the latency of the system. Retiming has many applications in synchronous circuit design. These applications include reducing the clock period of the circuit, reducing the number of registers in the circuit, decreasing the power consumption of the circuit and logic synthesis.

It can be used to increase the clock rate of a circuit by reducing the computation time of the critical path. Critical path is the path with the longest computation time among all paths that contain zero delays, and its computation time is the lower bound on the clock period of the circuit. The two factors affecting the frequency of operation is critical path and iteration bound. Retiming is done by applying the algorithm presented in and using Floyd Warshall algorithm.




**Retimed graph**

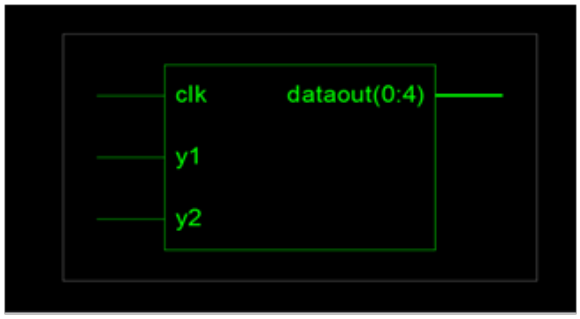

**RTL Schematic**



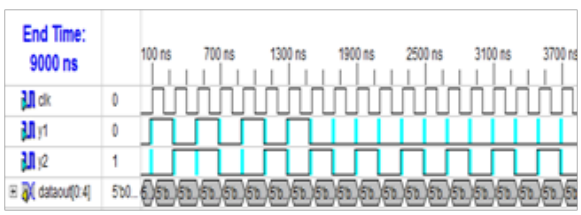

**Waveform of Retimed for 'n' bit input**

**III.3 Unfolding algorithm**

An unfolding algorithm is for parallel implementation of CRC. However, direct implementation of unfolding may lead to a parallel CRC circuit with long iteration bound, which is the lowest achievable CP. In Unfolding we have developed two novels look-ahead pipelining methods to reduce the iteration bound of the genuine serial LFSR CRC structures; after all this ,the unfolding algorithm is applied to obtain a parallel CRC structure with low iteration bound. The retiming algorithm is later applied to achieve the lowest CP.




**RTL Schematic**




**Waveform of Unfolding for 'n' bit**

This brief is organized as follows .Analyzes the effect of unfolding on the iteration bound of a parallel CRC circuit. Our proposed pipelining algorithms are discussed. Retimed and unfolded LFSR CRC circuits are compared.

## IV. RESULTS & ANALSYS

\The Proposed architecture is synthesized in Xilinx-8.2i and simulated in Xilinx ISE simulator, which required less than the half cycles compared to previous 64 bit design. In our programming in VHDL by specifying only generator polynomials,it directly gives F matrix useful for parallel CRC generation that is available in previous method but in present we are using the technology of pipelining, unfolding, retimed algorithm.

| METHOD | 4 INPUT LUT'S | | POWER | | DELAY | MEMORY |
|---|---|---|---|---|---|---|
| | USED | AVAILABLE | AVAILABLE | CONSUMES | | |
| PIPELINE | 4 | 1536 | 24 mw | 0.062 mw | 6.318 ns | 165352 kb |
| RETIMING | 5 | 1536 | 24 mw | 0.078 mw | 6.314 ns | 165352 kb |
| SERIALLFSR | 3 | 1536 | 24 mw | 0.046 mw | 6.314 ns | 165352 kb |
| UNFOLDING | 5 | 63 | 24 mw | 0.001 mw | 7.44 ns | 164780 kb |

From the table it is to observe that, the architecture proposed requires less than the half cycles and LUT compared to 64 bit and also area also reduced and power decreases, only disadvantage for the proposed architecture is delay time.

## CONCLUSIONS

In previous paper 32 bit parallel architecture required more clock cycles for 64 bit data. proposed design (n bits) required only few cycles to generate with same order of polynomial. So, it drastically reduces computation time to 30% and same increases the throughput. Pre-caluculation of F matrix is not required in this architecture. So,this is compact and easy method for fast CRC generation. CRC 'n' bit provides less latency and more throughput.

## ACKNOWLEDGEMENT

We would like to thank the Principal, HOD and all the teaching and non-teaching staffs of Anantha lakshmi Institute of Technology & sciences for helping to complete the work as mentioned in the paper.

## REFERENCES

[1] Campobello, G.; Patane, G.; Russo, M.; "Parallel CRC realization," Computers, IEEE Transactions on , vol.52, no.10, pp. 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, epartment of Electrical Engineering, Coimbatore Institute of Technology, Coimbatore- 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 |