ISSN 2348 – 9928
IJAICT Volume 4, Issue 2, February 2017 Doi:01.0401/ijaict.2014.07.23 Published on 05 (2) 2017

DESIGN OF PARALLEL CRC GENERATION FOR HIGH SPEED APPLICATION

Mr. Chaitali Tohgaonkar, *Prof. Sanjay Tembhurne* *Prof. Vipin Bhure*

Electronics and Communication Engineering,
GHREAT, RTMN University,
Nagpur, India.

Abstract*— *Error detection is important whenever there is a non-zero chance of data getting corrupted. A Cyclic Redundancy Check (CRC) is the remainder, or residue, of binary division of a potentially long message, by a CRC polynomial. This technique is ubiquitously employed in communication and storage applications due to its effectiveness at detecting errors and malicious tampering. The hardware implementation of a bit-wise CRC is a simple linear feedback shift register.This means that 'n' clock cycles will be required to calculate the CRC values for an n-bit data stream. Parallel CRC calculation can significantly increase the throughput of CRC computations. In this paper CRC-32 is design for Ethernet application. This paper presents implementation of parallel Cyclic Redundancy Check (CRC) based upon DSP algorithms of pipelining, retiming and unfolding. The architectures are first pipelined to reduce the iteration bound by using novel look-ahead techniques and then unfolded and retimed to design high speed parallel circuits.The methodology to be employed with VHDL, Xilinx ISE for simulation and test benchverification.

Keywords*— *Cyclic Redundancy Check (CRC), Pipelining, Retiming, Unfolding, VHDL Code.

I. INTRODUCTION

A CRC (Cyclic Redundancy Check) is a popular error-detecting code computed through binary polynomial division. To generate a CRC, the sender treats binary data as a binary polynomial and performs the modulo-2 division of the polynomial by a standard generator (e.g., CRC-32). The remainder of this division becomes the CRC of the data, and it is attached to the original data and transmitted to the receiver. Receiving the data and CRC, the receiver also performs the modulo-2 division with the received data and the same generator polynomial. Errors are detected by comparing the computed CRC with the received one. The CRC algorithm only adds a small number of bits (32 bits in the case of CRC-32) to the message regardless of the length of the original data, and shows good performance in detecting a single error as well as an error burst.

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. 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; 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 X MOD.

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 "⊕" in this paper) and multiplication is equivalent to AND. Figure 1 illustrates the basic architecture of LFSRs for serial CRC calculation [3].

Parallel CRC

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

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

ISSN 2348 – 9928
Doi:01.0401/ijaict.2014.07.23 Published on 05 (2) 2017

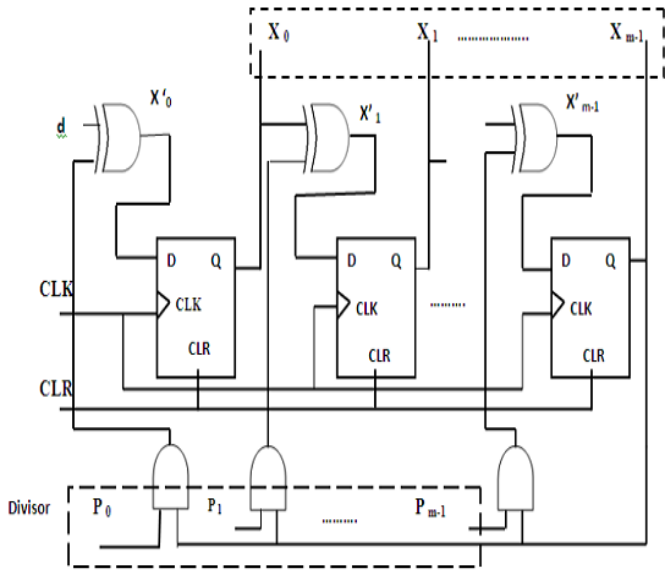

Fig 1 : Block diagram of LFSR

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. But it increases the no. of LUTs [5].

II. LITERATURE REVIEW

Hitesh H. Mathukiya; Naresh M. Patel; "A Novel Approach for Parallel CRC generation for high speed application," .In this paper, the parallel CRC generation deal with 64bit parallel processing based on built in F matrix with order of generator polynomial is 32. This gives CRC with half number of cycles. It drastically reduces computation time to 50% and same time increases the throughput. The no. of LUT getincreased, so area also get increase.

Yan Sun; Min Sik Kim; , "A Pipelined CRC Calculation Using Lookup Tables," In this paper, they present a fast cyclic redundancy check (CRC) algorithm that performs CRC computation for an arbitrary length of message. This paper proposes a table-based hardware architecture for calculating CRCby taking advantage of CRC's properties and pipelining the feedback loop. It achieves considerably higher throughput than existing serial or byte-wise lookup CRC algorithms.With delay increase in the critical path.

Weidong Lu and Stephan Wong, "A Fast CRC Update Implementation", IEEE Workshop on High Performance Switching and Routing Oct. 2003.In this paper, they presented a novel method to update theCRC code when packets are passing through interconnectingdevices. Andfocus on the CRC calculation that is performed during the routing of the Ethernet packets be encapsulating the packets into Ethernet frames, adding a frame header and adding a frame trailer.Itcalculates the intermediate results of the changed fields based on the parallel CRC calculation and performs a single step update afterwards. And the number of cycles is dramatically reduced.The fast CRC update only calculates the changed portion of a frame.

Campobello, G.; Patane, G.; Russo, M.; "Parallel CRC realization". This paper presents a theoretical result in the context of realizing high speed hardware for parallel CRC checksums.The number of bits processed in parallel can be different from the degree of the polynomial generator.Presented Pre-calculated F-matrix based 32 bit parallel processing. Which isfaster and more compact and is independent of the technology used in its realization. But it doesn't work if polynomial change.

III. PROPOSED TECHNIQUE

The Pipelining, Retiming and Unfolding algorithm is used in proposed paper which is used to reduced critical path delay and clock cycle.

Pipelining: It reduces the effective critical path by introducing pipelining latches along the criticaldata path either to increase the clock frequency orsample speed or to reduce power consumption at the same speed.

Retiming: 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.

Unfolding: It's a transformation technique that can be applied to DSP program to create a new program describing more than one iteration of the original program.

IV. CONCLUSION

This paper shows the use of Parallel CRC for high speed application such as Ethernet. This paper presents the implementation of Parallel CRC-32 with the use of Pipelining, Unfolding and Retiming algorithm. Pipelining

has decreased the iteration bound of the architecture effectively. Applying unfolding technique to pipelined architecture increased the throughput of the circuit and thereby applying retiming to the architecture reduced the critical path delay. So applying pipelining, unfolding and retiming to the CRC has increased the throughput to achieve high speed design.

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.