# Design and Implementation of High Performance Parallel CRC Architecture for Advanced Data Communication

Md. Mashrur Arifin*, Md. Tariq Hasan, Md. Tarikul Islam, Md. Almahmud Hasan and Himadri Shekhar Mondal
Electronics and Communication Engineering Discipline, Khulna University, Khulna, Bangladesh.
*arifinmash@gmail.com

***Abstract*—Cyclic Redundancy Check (CRC) leverages to detect the error of digital data throughout generation, transmission, storage or processing. CRCs are widely used for being simple to execute in binary appliances, crafty to mathematical simulation as well as exclusively performance oriented at identifying generic deviation occured due to intrusion in communication channels. Commonly, hardware implementation of Cyclic Redundancy Check (CRC) computations rely on the Linear Feedback Shift Registers (LFSRs). LFSR framework processes bits serially that is one message bit per clock cycle but while considering high-speed data communications, serial implementation speed is significantly inadequate which causes delay. In this research, a hardware architecture is proposed for parallel computation. Its architecture is not polynomial dependent. After testing its functionality using ModelSim, it is implemented in Altera DE1 FPGA (Field Programmable Gate Array) board and analyzed using Quartus II, TimeQuest Timing Analyzer and Power Play Power Analyzer tools. It is found that the designed took 2771 LEs (Logical Elements), it has 102 pins and consumed 120.68 mW power. Functionality test and FPGA implementation showed that CRC was computed in single clock pulse of frequency of 23.71 MHz and its throughput is 1.656 Gbps. It can be configured for a different polynomial at any time externally. The focus of the research is to represent an efficient, better throughput along with compact systematic interpretation for parallel CRC hardware which will alleviate the flaws including the challenges of the existing CRC checker which will be prominent for next generation high speed communication.**

***Index Terms*—CRC, FPGA, Implementation.**

## I. Introduction

In Next Generation communication systems, high-speed and high-reliability are the two most important parameters, so it is conceivable that faster speed and capability of deploying broad segments of the information in future parallel CRC computation circuits is a must. Such cases, a technique for parallel CRC computation, where consecutive units of bits use to process concurrently are highly desired. But the existing parallel CRC are applicable for fixed polynomial and data width. That's why a generalized parallel CRC formulation is highly necessary which can be incorporated to any polynomial and computable in one clock cycle.

For faster data communication, devices should have the competency to convey data from one appliance to another with tolerable deviation. For most appliances, a system must persuade the integrity of data received along with transmitted data. Data are conveyed from one module to the next can be corrupted in the midst. For Data at rest or Data in motion, there are many aspects such as Gaussian noise, malfunctions of hard drive and physical connections with faults can modify one or more bits of message [1]. Such cases desire a system for identifying errors. For detecting error Cyclic Redundancy Check (CRC) is the one of the widely functioned method. In comparison with traditional CRC while synchronizing with high speed data communication, objective is to increase CRC generation speed and check the integrity.

Data communicating device should have the capabilities to transport data from transmitter to receiver with adequate accuracy. But message wholly or partially can be distorted in the passage. So, it is required to detect the errors. CRC is an effective error diagnosis tool widely utilized for various communication and data infrastructures. Linear Feedback Shift Register (LFSR),one of the most traditional hardware solutions for CRC containing flip-flops and logic gates [2], [3]. Bits processing in LFSR architecture execute serially while in case of high-speed data communications serial implementation speed is obviously inadequate which generates delay. Such cases, a CRC parallel computation technique, where successive units of bits are processed concurrently is highly desired. But the existing parallel CRC are applicable for fixed polynomial and data width. Here, a generalized parallel CRC formulation is presented which can be incorporated to any polynomial and data width for speedy performance.

This paper has been well ordered as section II, related works, in which previous related papers are presented which are considered as case study for developing this research, section III, methodology where the main methods of analyzations and developments are presented. Section IV is based on results of this research. Finally conclusion and future aspects are presented in section V.

## II. Related works

During data transmission, the system must be capable to transport message from one appliance to other with sufficient efficiency. For most applications, certain mechanism must ensure the received data are identical to the transmitted data. Raw data can become corrupted during transmitted from one node to another in intermediate passage. One or more bits of a message can be altered by many factors. Applications want a component for identifying and remedying errors [4], [5].

978-1-7281-6040-5/19/$31.00 ©2019 IEEE

Considering the scenario of data communication and data storage ,data received may not be equivalent to the data transmitted due to the noise and the intrusion, that prompts the data transmission flaw and this research is completed by Zhang and his team [6]. Error detection coding is a strategy for recognizing and revising these errors to guarantee data is moved unblemished from its origin to its target. In Different forms of Digital Data Communication such as network communications,cellular telephone networks, optical data storage media, fault-tolerant computing in computer memory , satellite, and deep-space communication etc Error detection coding is extensively utilized. According to Klove's [7] investigation error coding applies mathematical equations to encode information bits into longer bit words at the source end during transmission. The codeword can be decoded at the destination to recover that was initially sent. The additional bits in the codeword give recurrence that, as indicated by the coding scheme applied, will approve the destination to use the decoding method to determine whether the correspondence channel introduced flaw and at intervals correct them facilitate data termination from retransmission.

According to Singh and his research group [8] different error detecting coding plans are picked relying upon the errors types anticipated, the correspondence channel's normal mistake rate, and the possibility of data retransmission. Considering progressive communications, high frequency processors and sophisticated communications technology allows increasingly complex coding plans, with more efficient error detecting and correcting capabilities which is feasible for smaller embedded systems.

To realize how errors can be dealt with, it is important to have a close look at what error truly is. Basically, a data frame can be structured with the m-data bits (message bits) and along with r-redundant bits (or check bits). Considering the absolute number of bits is n (m + r). A unit of n-bit contain data and check-bits generally indicates to as an n-bit code word. For two code-words, given 10010101 and 11010100, the number of differing corresponding bits can be determined just by functioning XOR operation of given code words and computing the number of 1's in the answer. The count of bits place where code words vary known as the Hamming distance. Defining when two code words having d Hamming distance, d single-bit errors will be required for the conversion one code word into other. According to a lots of research, the properties of error detection and correction depend on its Hamming distance. [9], [12], [13].

CRC is one of the most powerful error-detecting codes. The cyclic redundancy check is the brainchild of W. Peterson, who described the operation in 1961 [12]. Since its invention, it has been used in network communications and data storage solutions in order to detect errors.For message block with k-bit, n-bit sequence is generated by transmitter, which is called remainder. So according to a research by Stallings ,consisted resulting frame (k+n)-bits, is completely divisible by the polynomial. [11]. Then the incoming frame at Receiver end is divided by that polynomial and if no remainder found ascertain zero error occurred.

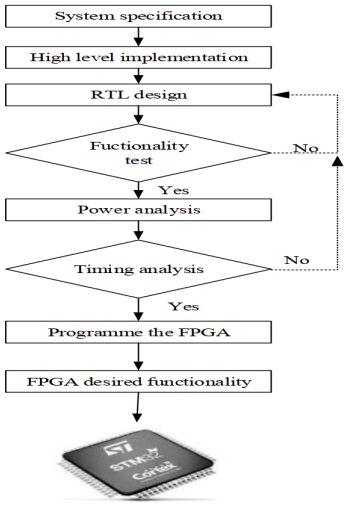


Fig. 1. Steps of system implementation in FPGA.

## III. Methodology

This section contains methods which will be used to design, simulate and check the functionality of CRC generation. Different EDA tools are used to simulate the methods and to observe the output.

### A. Flow Chart of System Implementation in FPGA

System specification: The first step of this research is to implement a Field Programmable Gate Array (FPGA). In this step the quantity of input ports, output ports and functionality of the module is indicated in figure 1.

High level implementation of the system: It is very simple to implement a module by high level programming language such as MATLAB, C++. So its desired functionality is implemented in high level platform.

RTL (Register Transfer Level) design: In this step of implementation the module is designed in ModelSim using Verilog.

Functionality check: After designed in ModelSim its functionality is checked. If it works properly then it goes to the next step, otherwise it is redesigned in ModelSim.

Timing analysis: Design is analyzed to check the maximum operating frequency. If the timing requirement is met then it goes to the next step, otherwise it goes to the RTL design step.

Power analysis: In this step power analysis of the designed circuit is performed.

Program the FPGA: In this step the program is downloaded to the FPGA board.

FPGA with desired functionality: After downloading the program, the output is observed with the help of specified output pins (LEDs).

### B. CRC Polynomials

Some commonly used polynomial which used in different application discussed in Table 1.

| Common Name | Generator Polynomial | Hex |
|---|---|---|
| CRC-12 | $X^{12} + X^{11} + X^3 + X^2 + X + 1$ | 80F |
| CRC-16 | $X^{16} + X^{15} + X^2 + 1$ | 8005 |
| CRC-CCITT | $X^{16} + X^{12} + X^5 + 1$ | 1021 |
| CRC-32 | $X^{32}+ X^{26}+ X^{23} +X^{22} +X^{16} +X^{12}+ X^{11}+ X^{10}+X^8+ X^7+ X^5+ X^4+ X^2+ X+ 1$ | 04C11DB7 |

TABLE I
FEATURES OF VARIOUS CRC STANDARD [25].

Polynomial division procedure is fundamental for computing the CRC checksum executed in a hardware circuit . A Shift register is utilize in the procedure, which we indicate by CRC. The length of this is r (the degree of P) bits. At the phase of XOR where the subtractions are done, high-order bit need not to be represented as subtraction of high-order bit of P and the quantity is being executed from both 1. The division procedure can be defined as below,

1. Firstly, Set all bits 0 to the CRC register
2. Take first/next bit bit m (message).
3. If found 1 in the high-order bit of CRC, Shift both CRC and m left 1 position and XOR the result with low-order r bits of P with the r.
4. Else, shift left 1 position for CRC and m. Go back to get the next one If more message bits remain.

For this scenario, the subtraction should be done before shifting. It would be done in such steps that entire generator polynomial will be held by the CRC register. Rather to align things properly, only the low-order n bits of P held by the CRC register, so the shift is done first. An example is demonstrated for the Polynomial P = $X^3$ + X + 1 while the Message M = $X^7$ + $X^6$+ $X^5$+ $X^2$+ X. Binary expression be , P = 1011 and M = 11100110. Initially CRC contains 000 so the high-order bit is 0, now first message bit would be shifted, giving: 001. Then the steps are,

1. For 001; 0 is the High-order bit, so second message bit would be shifted which give: 011
2. For 011; 0 again High-order bit, so third message bit would be shifted which give: 111
3. For 111; 1 is the High-order bit, so after shift and XOR with 011.give:101
4. For 101; 1 is the High-order bit, so after shift and XOR with 011.give: 001
5. For 001; 0 is the High-order bit, so fifth message bit would be shifted which give: 011
6. For 011; 0 again High-order bit, so sixth message bit would be shifted which give: 111
7. For 111; 1 is the High-order bit, so after shift and XOR with 011.giving:101
8. For 101; No more message bits remain, so this find the remainder.

Following steps could be executed in the circuit defined in Figure 2.

In the figure 2 three bits of the CRC register are denoted by the boxes. When input a message bit, if 0 found in the high-order bit of C2 box, concurrently the message bit would be shifted into the C0 box, bit in X0 would be shifted to X1, bit in X1 would be shifted to to X2, and the bit in X2 would be abandoned. If 1 found in the high-order bit of the CRC register, then a 1 would remain at the lower input of each of the two XOR.

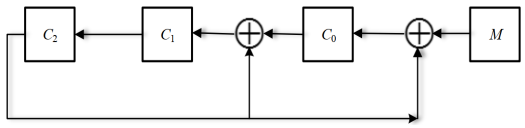


Fig. 2. Linear feedback shift register CRC module logic diagram.

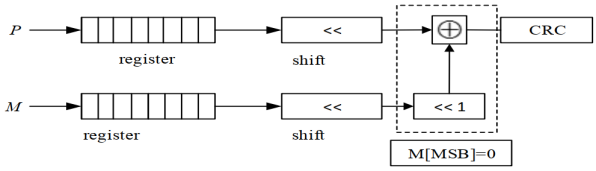


Fig. 3. Architecture of Proposed CRC module.

### C. Proposed CRC Module Architecture

Proposed CRC module is designed to generate or check CRC in parallel computation method. Previous version of CRC needs huge time to generate or check CRC, it needs clock pulse as the length of the (m+r-1). Proposed CRC module needs only single clock pulse to compute total CRC.

Properties of the proposed module are as,

1. It takes single cycle count to generate reminder.
2. Polynomial is reconfigurable.
3. Parallel technique of CRC computation.
4. More registers needed than a serial CRC generator.
5. The polynomial length and message length can be varied.

### D. Algorithms of Proposed CRC Module

Algorithm of proposed CRC is presented in figure 4

```
Algorithm of proposed CRC
   1. M1<=0;
   2. M <= message ;
   3. P <= polynomial;
   4. Pos1P <= Value;
   5. Pos1M <= value;
   6. IF ( M[0] = 1 && P[0] = 1)
   7. FOR(LM)
   8. M1=M^P;
   9. END IF
   10.     END FOR
   11.     M=M+M1;
   12.     CRCOUT <= M;
```

Fig. 4. Algorithm of proposed CRC.

## IV. RESULTS

Serial computation technique of CRC and their analysis in different EDA tools such as MATLAB, ModelSim, and

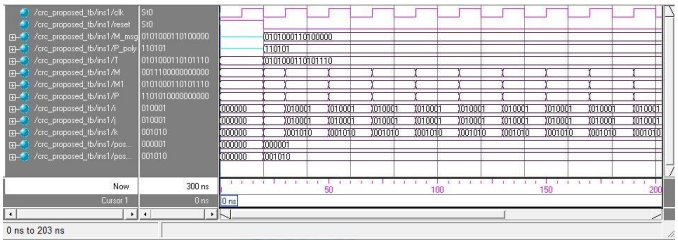

Fig. 5. Functionality test of proposed CRC module in ModelSim.

| | No. of element(s) | Total |
|---|---|---|
| Total logic elements | 2771 | 33216 < 15% |
| Total combinational functions | 2770 | 33216 < 15% |
| Dedicated logic register | 36 | 33216 < 1% |
| Total register | 36 | |
| Total virtual bits | 0 | 0% |
| Total number of pin | 102 | 322 < 32% |

TABLE II
NUMBER OF LOGIC ELEMENT ANALYSIS OF PROPOSED CRC.

Quartus II are analyzed and finally tested it's functionality in FPGA board. After simulation in different EDA tools information's are found such as timing information, power information and logical element information.

The proposed CRC computing technique is a unique one which can compute CRC in parallel with high speed performance. It also can compute CRC without any fixed polynomial which is very magnificent. This proposed technique can compute only in cycle for any length of message or any polynomial. Its overall performance is discussed below.

### A. RTL Functionality Analysis in ModelSim

The RTL code is developed using Verilog language and observed the behavior with ModelSim. To check its output here $2^n$M = 101000110100000, and P = 110101. And the CRC is shown in figure 5 as T = 0101000110101110.

### B. Compilation Report in Quartus II

The proposed CRC is simulated in ModelSim and analyzed in Quartus II to check logical element information. Simulated Verilog file in ModelSim is modified according to the Quartus II. Where every input and output ports are assigned according to the ports availability in Altera DE1 board. The logical elements that required for implementing this module are presented in table 2.

### C. Power Analysis Using PowerPlay Power Analyzer

Power analysis of proposed CRC module is analyzed by PowerPlay Power Analyzer in Quartus II which is presented in figure 6. Table 3 represents the summery of power status of this analyzation.

### D. Timing analysis in TimeQuest Timing Analyzer

Timing analysis is performed using TimeQuest Timing Analyzer tools in Quartus II. Timing analysis report shows two slack information and maximum operating frequency information. Figure 7 indicates the scenario.

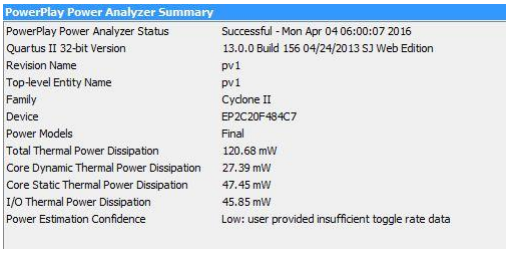


Fig. 6. Power analysis of proposed CRC module.

| | |
|---|---|
| Total thermal power dissipation | 120.68 mW |
| Core dynamic thermal power dissipation | 27.39 mW |
| Core static thermal power dissipation | 47.45 mW |
| Input output thermal power dissipation | 45.85mW |

TABLE III
POWER ANALYSIS OF PROPOSED CRC MODULE.

### E. Implementation in Altera DE1 Board

After simulation in Quartus software II its functionality is checked in Altera DE1 board. Working procedures for DE1 board implementation are as,

1. Verilog code is compiled in Quartus II software.
2. After compilation a (.sof) file is generated by the software.
3. This (.sof) file is then loaded to the Altera DE1 board.
4. The functionality is observed by using assigned I/O pins.

Maximum operating frequency of proposed CRC module is presented in figure 8. By following all the procedures, for message $2^n$M = 101000110100000 and polynomial P = 110101 the remainder is 01110. The desired output is marked as yellow box in the figure 9 where first LED (Light Emitting Diode) and last LED are off (from left side) and the remaining are on which represents the binary 01110. So it can be said that, the proposed CRC module worked properly in Altera DE1 board which is the theoretical remainder of the mentioned message and polynomial.

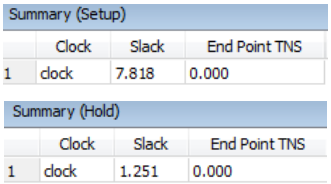


Fig. 7. Slack information of proposed CRC module.

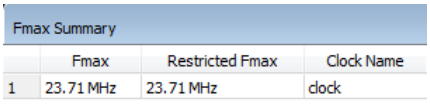


Fig. 8. Maximum operating frequency of proposed CRC module.

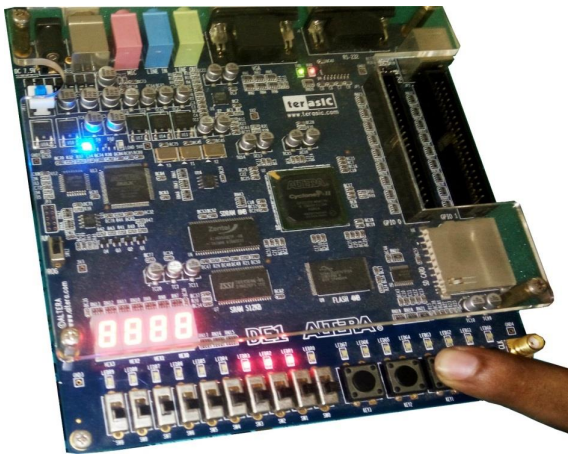

Fig. 9. Functionality test of proposed module in Altera® DE1 board.

| Parameters | Previous version of CRC-16 Module | Proposed CRC Module |
|---|---|---|
| Time | Time required depends on length of polynomial and message | Only one clock cycle. |
| Slack information | Setup slack 8.73. Hold slack 0.615 | Setup slack 7.81. Hold slack 1.25 |
| Maximum operating frequency | 808.41MHz | 23.71 MHz |
| Computation technique | Serial computation. | Parallel computation. |
| Polynomial | If polynomial is changed total circuitry would have to change. | Its polynomial is re-configurable. |
| No of Element required | 33 | 2771 |
| Total number of pins | 35 | 102 |
| Power | 150.36 mW | 120.68 mW |

TABLE IV
COMPARISON BETWEEN PREVIOUS MODULE AND PROPOSED CRC MODULE.

### *F. Comparison between Traditional Module and Proposed Module*

As we studied previous CRC modules and proposed different types of CRC module, a comparison is given below in table 4. It can be identified from the table that, the proposed CRC module consumes less power as well as enriched with other unprecedented characteristics. It is found that the designed took 2771 LEs (Logical Elements), it has 102 pins and consumed 120.68 mW power. Functionality test and FPGA implementation showed that CRC was computed in single clock pulse of frequency of 23.71 MHz and its throughput is 1.656 Gbps. It can be configured for a different polynomial at any time externally.

## V. CONCLUSION

For alleviating the drawbacks of serial CRC in high-speed data communications our proposed parallel computation of the CRC offers successive units of bits which have been handled simultaneously. Moreover existing parallel CRCs are applicable for fixed polynomial and data width.So our research endeavour is a generalized parallel CRC formulation which can be incorporated to any polynomial and data width for fast operation and to minimize the delay. The overall performance of the proposed design is satisfactory but the operating frequency of the proposed module is quite low. So in future there is a prospect to optimize and increase the maximum operating frequency of the module to cope up with the present and future high speed technology.

## REFERENCES


[1] Huo, Yuanhong, Xiaoyang Li, Wei Wang, and Dake Liu. "High performance table-based architecture for parallel CRC calculation." In The 21st IEEE International Workshop on Local and Metropolitan Area Networks, pp. 1-6. IEEE, 2015.

[2] Campobello, Giuseppe, Giuseppe Patane, and Marco Russo. "Parallel CRC realization." IEEE Transactions on Computers 52, no. 10 (2003): 1312-1319.

[3] Morrison, Daniel, Dennis Delic, Mehmet Rasit Yuce, and Jean-Michel Redouté. "Multistage Linear Feedback Shift Register Counters With Reduced Decoding Logic in 130-nm CMOS for Large-Scale Array Applications." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27, no. 1 (2018): 103-115.

[4] Forouzan, A. Behrouz. Data communications & networking (sie). Tata McGraw-Hill Education, 2007.

[5] Bhardwaj, Kartik Krishna, Anirudh Khanna, Deepak Kumar Sharma, and Anshuman Chhabra. "Designing Energy-Efficient IoT-Based Intelligent Transport System: Need, Architecture, Characteristics, Challenges, and Applications." In Energy Conservation for IoT Devices, pp. 209-233. Springer, Singapore, 2019.

[6] Zhang, Yanbin, and Qi Yuan. "A multiple bits error correction method based on cyclic redundancy check codes." In 2008 9th International Conference on Signal Processing, pp. 1808-1810. IEEE, 2008.

[7] Klove, T. "Using codes for error correction and detection (Corresp.)." IEEE Transactions on Information Theory 30, no. 6 (1984): 868-870.

[8] Singh, Jatinder, and Jaget Singh. "A comparative study of error detection and correction coding techniques." In 2012 Second International Conference on Advanced Computing & Communication Technologies, pp. 187-189. IEEE, 2012.

[9] Hamming, Richard W. "Error detecting and error correcting codes." The Bell system technical journal 29, no. 2 (1950): 147-160.

[10] Schoeny, Clayton, Irina Alam, Mark Gottscho, Puneet Gupta, and Lara Dolecek. "Error Correction and Detection for Computing Memories Using System Side Information." In 2018 IEEE Information Theory Workshop (ITW), pp. 1-5. IEEE, 2018.

[11] Stallings, William. Data and computer communications. Pearson Education India, 2007.

[12] Peterson, Wesley. "Encoding and error-correction procedures for the Bose-Chaudhuri codes." IRE Transactions on Information Theory 6, no. 4 (1960): 459-470.

[13] Chervyakov, Nikolay I., Pavel A. Lyakhov, Mikhail G. Babenko, Irina N. Lavrinenko, Anton V. Lavrinenko, Maxim Anatolievich Deryabin, and A. S. Nazarov. "A new model to optimize the architecture of a fault-tolerant modular neurocomputer." Neurocomputing 303 (2018): 37-46.

[14] Poulos, Alexandra, Dylan Wallace, Robert Robey, Laura Monroe, Vanessa Job, Sean Blanchard, William Jones, and Nathan DeBardeleben. "Improving Application Resilience by Extending Error Correction with Contextual Information." In 2018 IEEE/ACM 8th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS), pp. 19-28. IEEE, 2018.