KILAT
Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

Analysis and Design of CRC-32 IEEE 802.3 Generator for 8 Bit Data Using VHDL

Aprilia Putri Dewanty [1*]; Bheta Agus Wardijono [2]

1. Department of Electrical Engineering, Faculty of Industrial Technology
   Universitas Gunadarma, Cimanggis, Depok, Jawa Barat 16451 Indonesia
2. STMIK Jakarta STI&K, Jakarta Selatan, DKI Jakarta 12140 Indonesia

[*)]Email: apriliaputridewanty@gmail.com

Received: 7 Juni 2022 | Accepted: 24 November 2022 | Published: 5 Desember 2022

ABSTRACT

In the communication system to achieve better quality data transmission required a method that can detect errors and correct errors. Cyclic Redundancy Check (CRC) is one of the methods used to perform data transmission on data link layer that can detect errors. CRC-32 is used to error-checking on Ethernet or implemented to IEEE 802.3. CRC generator in this research use CRC-32 IEEE 802.3 with 8 bit data width. This research can be implemented in the field of Electrical Engineering, especially in the telecommunications section, namely Ethernet which functions for transfer files and data via a computer network. CRC here has a role to prevent data changes caused by noise during the transmission process. The methods used in this design is modulo-2 division parallel circuit. This design is expected to use a simple schematic circuit, less noise and less resources. Testing is done by matching result of simulation using Xilinx ISE Simulator with implementation on Spartan 3E XC3S500E device with result of count . This research requires a resource of 223 4-input LUTs, 114 Occupied slice, 72 IOB flip flops, 114 bonded IOBs and 1 BUFGMUXs, where this research obtained resources is fewer than with previous research.

Keywords: *CRC Design, VHDL, Xilinx ISE 8.1*

ABSTRAK

Dalam sistem komunikasi untuk mencapai kualitas pengiriman data yang lebih baik diperlukan suatu metode yang dapat mendeteksi kesalahan dan mengoreksi kesalahan. Cyclic Redundancy Check (CRC) merupakan salah satu metode yang digunakan untuk melakukan transmisi data pada lapisan data link yang dapat mendeteksi kesalahan. CRC-32 digunakan untuk memeriksa kesalahan pada Ethernet atau diimplementasikan ke IEEE 802.3. Generator CRC dalam penelitian ini menggunakan CRC-32 IEEE 802.3 dengan lebar data 8 bit. Penelitian ini dapat diimplementasikan pada bidang Teknik Elektro khususnya pada bidang telekomunikasi yaitu Ethernet yang berfungsi untuk mentransfer file dan data melalui jaringan komputer. CRC disini berperan untuk mencegah perubahan data yang disebabkan oleh noise selama proses transmisi. Metode yang digunakan dalam perancangan ini adalah rangkaian paralel modulo-2 division. Desain ini diharapkan menggunakan skema rangkaian sederhana, lebih sedikit kebisingan dan sumber daya yang lebih sedikit. Pengujian dilakukan dengan mencocokkan hasil simulasi menggunakan Xilinx ISE Simulator dengan implementasi pada perangkat Spartan 3E XC3S500E dengan hasil penghitungan. Penelitian ini membutuhkan resource 223 4-input LUT, 114 Occupied slice, 72 IOB flip flop, 114 IOB berikat dan 1 BUFGMUX, dimana penelitian ini memperoleh resource yang lebih sedikit dibandingkan dengan penelitian sebelumnya.

Kata kunci: *CRC Design, VHDL, Xilinx ISE 8.1*

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

1. INTRODUCTION

The development of communication and digital technology at this time is very fast. In digital communications, data received may not be same as the transmitted data due to noise and interference, which causes errors during the process of data transmission and storage. To validate the data received is correct, many methods that can be used on a computer to detect that the received data is valid or invalid. CRC method is very simple and efficient to detect and correct errors.

Cyclic Redundancy Check (CRC) is an error-checking block code that has been used for error detection, while the received word has to be divided by a predetermined number called the generator number. If the remainder is zero, this means that there is no error detected, whereas nonzero remainder this means that there is an error detected. The error detection is done by counting the remaining bit on the message that needs to be transmitted. The remaining bit results will connect to the message to generate the codeword.

Specific interface chip will cause waste of resources and increased cost, particularly in the field of electronic design. This situation results in the requirement of realizing the whole system function in a single or a very few chips. Therefore the design will be designed using Very high speed Integrated Circuit Hardware Description Language (VHDL) which can be implemented on FPGA. The VHDL source code has been edited and synthesized using Xilinx ISE 8.1. It will be simulated and tested using ISim. By using this software can be known summary of the design (number of slices and logic gates used in the design) that have been made.

The device may take corrective action, such as rereading the block or requesting that it be sent again. Otherwise, the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking. Many engineers conducting research on Cyclic Redundancy Code (CRC) Generator, including :

In the research work of Pramod S P, Rajagopal A, and Akshay S Kotain with title “FPGA Implementation of Single Bit Error Correction using CRC”, in this research the designs are made using VHDL and generator polynomial use is CRC-16 and CRC-8. The algorithm has been implemented and verified on Xilinx Virtex-5 FPGA device. The device used for implementation is xc5vlx30 with speed grade 3. CRC generator is designed using method of modulo-2 division method. Purpose of this project is to focuses on effective implementation to detect the exact place of single bit error and correct them using minimum hardware. Experimental results demonstrate the validity of the proposed system. **(Pramod S P, Rajagopal A, and Akshay S Kotain, 2012)**

In the research of Debopam Ghosh, Arijit Mitra, Arijit Mukhopadhyay, Aniket Dawn and Devopam Ghosh with title “A GENERALIZED CODE FOR COMPUTING CYCLIC REDUNDANCY CHECK”, in this research the designs are made using VHDL and generator polynomial use is CRC-3 for simulation. This simulation uses software Xilinx ISE Design Suite. CRC generator is designed using method of modulo-2 division method. Purpose of this project is for developing a generalized CRC code where the user can vary the size of the generator polynomial. Experimental results demonstrate the validity of the proposed system. **(Debopam Ghosh, Arijit Mitra, Arijit Mukhopadhyay, Aniket Dawn and Devopam Ghosh, 2013)**

In the research work of P. Harika and B. V. V. Satyanarayana with title “FPGA Based High Speed Parallel Cyclic Redundancy Check”, in this research the designs are made using Verilog HDL and generator polynomial use is CRC-4. CRC generator is designed using method of modulo-2 division method and LFSR (linear feedback shift Register). Purpose of this project is to to design high-speed parallel circuits of cyclic redundancy check (CRC). Implementation of CRC based on unfolding, pipelining, and retiming algorithms. CRC architectures are first pipelined to reduce the

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

iteration bound by using novel look-ahead pipelining methods and then unfolded and retimed to design high-speed parallel circuits. the proposed design can increase the speed by up to 25% and control or even reduce hardware cost. **(P. Harika and B. V. V. Satyanarayana, 2013)**

In the research work of Chaitali Tohgaonkar , Prof. Sanjay B. Tembhurne and Prof. Vipin S. Bhure with title "Design of Parallel CRC Generation for High Speed Application", in this in this research the designs are made using VHDL and generator polynomial use is CRC-32. CRC generator is designed using method of modulo-2 division method and parallel pipelining methods . Proposed design (32 bits) reduces the computation time and also reduces the number of slices used. So applying pipelining to the CRC has increased the throughput to achieve high speed design. This paper presents implementation of parallel Cyclic Redundancy Check (CRC) based upon DSP algorithms of pipelining, retiming and unfolding. The design is simulated using Xilinx ISE. **(Chaitali Tohgaonkar , Prof. Sanjay B. Tembhurne and Prof. Vipin S. Bhure, 2015)**

In the research work of Deepali P. Atrawalkar and Manoj D. Bagde with title "Design and Simulation of Parallel CRC Generation Architecture for High Speed Application", in this in this research the designs are made using VHDL, simulated using Modelsim and synthesized by Altera Quartus II. The generator polynomial use is CRC-16 CCITT. CRC generator is designed using pipelined CRC method. Purpose of this project is to use pipelined CRC which can reduce clock cycle to achieve high speed design. The design can be implemented with DSP algorithms which improves the time further,increase speed in practice. **(Deepali P. Atrawalkar and Manoj D. Bagde, 2017)**

In the research work of Abdul Rehman Buzdar, Liguo Sun with title "Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths", in this research the designs are made using VHDL and generator polynomial use is CRC5, CRC8, CRC16 and CRC32 inside a CRC accelerator main block. Purpose of this project is to generate the performance of CRC accelerated Microblaze SoftCore embedded processor datapath in terms of execution time and energy efficiency. This acceleration is achieved at the cost of some area overhead. **(Abdul Rehman Buzdar and Liguo Sun, 2017)**.

2. CRC PROPOSED METHOD

2.1. Block Input Component Design

The CRC generator was designed in this research using a divisor polynomial of CRC-32 IEEE 802.3 with data width is 8 bits. CRC generator has 4 input port and 2 output port, which port are port of dataword, crc_32, clk and rst as input ports and the output ports are port of remainder and codeword. The dataword port in this design has an 8 bit data width. The clock port serves to generate a pulse signal (clock). Rst port is a port to enable or disable CRC generator. The crc_32 port is value of polynomial divisor (CRC-32) ,this port has a 33 bit data width. Remainder port is the result of combining dataword and augmented dataword then dividing it by divisor polynomial. Codeword port is the result of combining dataword with remainder value. Block diagram of this process can be seen in the figure 1.

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

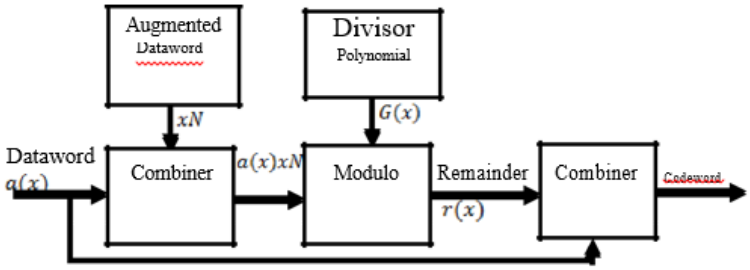


Figure 1. Block Diagram of CRC calculation

CRC generator design will be simulated to Xilinx ISE software using VHDL programming language.

2.2. Program Description

In this section will discuss about the CRC program scripts created using language VHDL. The draft design created using Xilinx ISE software. This program will initially initialize and then put the data processed with data width of 8 bits. Before the calculation, the program will check rst port. If the rst port is high logic then the remainder port will be 0 or return to initially condition but if the low logic then it will generate the CRC code with the specified polynomial divisor. How the program works can be seen as shown in figure 2.

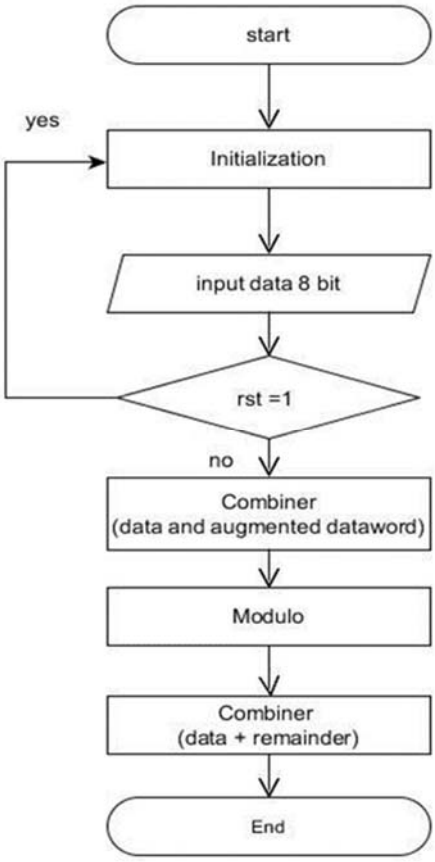


Figure 2. Program Flowchart

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

2.3. Implementation of Design using Software

In the process of design system using VHDL code will be implemented using Xilinx ISE 8.1 software as shown in the figure 3.

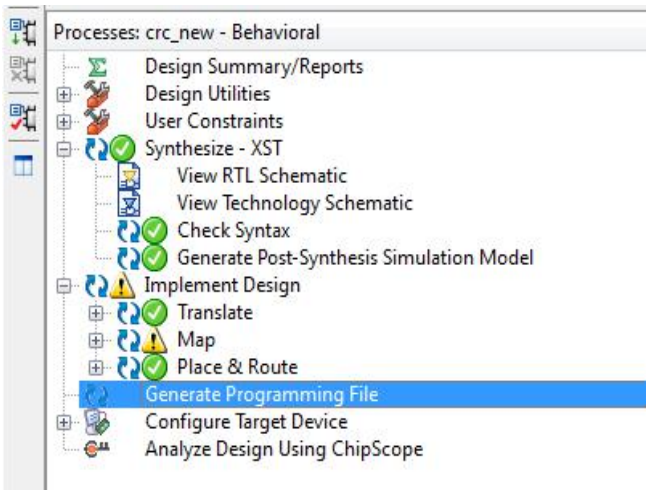


Figure 3. Process Panel

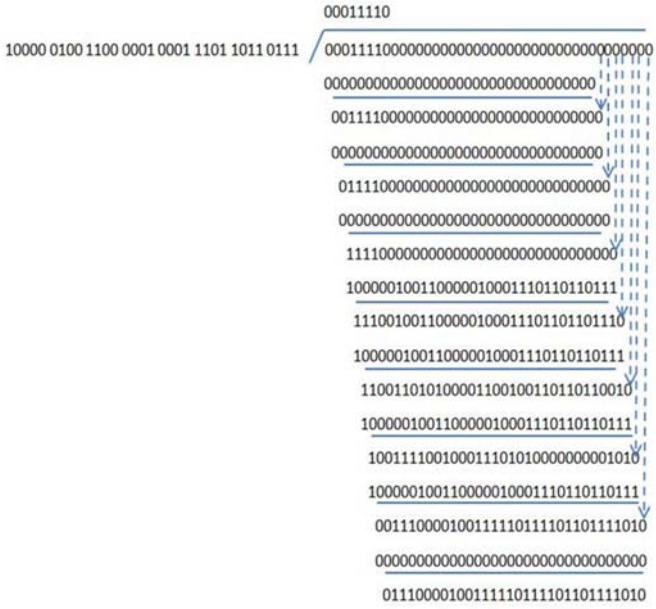
Synthesize is the process of generating a netlist for each source file. After conducting the synthesis process then implement design. In this section there are three steps: translate, map, place and route. Translate is the process of combining multiple files into a netlist. Map is a process to map a slice and I/O Blocks. Place and route is process placing the design on the chip and components connected. After implement design is completed, it can be seen a summary of the design and report. After this process, the RTL schematic can be seen. Register-Transfer Level (RTL) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals.

3. CRC TESTING AND ANALYSIS

3.1. System Testing

System testing is done based on the design that has been made in this research. Before performing a system simulation, first determine the data used in this research. There are 8 test data to be conducted on this research then do calculations using modulo 2 division method. The result of count with result of the design and simulation will be compared to determine the simulation result. The following is an example of a CRC calculation using one of the data attempted 00011110 (1E in hexadecimal).

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279



As calculated above, the dataword will be combined with augmented dataword and then it will be divided by the polynomial divisor to generate remainder bit. The result of remainder bit is 01110000100111110111011011111010. After the remainder bits are obtained then combined with the dataword and then it will generated the codeword.

Codeword = 00011110_01110000100111110111011011111010 (1E709F7B7A in hexadecimal). Codeword will be divided by the polynomial divisor for error checking. The following is a calculation for error checking.

On the above calculation, remainder bit is 00000000000000000000000000000000. It has been ensured that data received no error.

In Table 1 shows the trial data performed and also the results of calculations that have been obtained. If the codeword matched with the result of the calculation so it shows that there is no error detected.

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

Table 1. Table Trial Data

No.	Data (Binary)	Data (Hex)	Remainder (Binary)	Remainder (Hex)	Codeword (Hex)
1.	11010011	D3	0001 1100 1101 1000 0110 1101 0011 0000	1CD86D30	D31CD86D30
2.	00011110	1E	0111 0000 1001 1111 0111 1011 0111 1010	709F7B7A	1E709F7B7A
3.	11111111	FF	1011 0001 1111 0111 0100 0000 1011 0100	B1F740B4	FFB1F740B4
4.	00001111	0F	0011 1000 0100 1111 1011 1101 1011 1101	384FBDBD	0F384FBDBD
5.	11010100	D4	0000 0010 1001 1111 0011 1101 0011 0101	029F3D35	D4029F3D35
6.	11010101	D5	0000 0110 0101 1110 0010 0000 1000 0010	065E2082	D5065E2082
7.	11010110	D6	0000 1011 0001 1101 0000 0110 0101 1011	0B1D065B	D60B1D065B
8.	11010111	D7	0000 1111 1101 1100 0001 1011 1110 1100	0FDC1BEC	D70FDC1BEC

The result of trial data in hexadecimal is completely can be seen in the appendix. On the results obtained there is no error, because the codeword matched with the result of the calculation. Simulation results of the design can be seen in Figure 4. This simulation used Isim simulation.

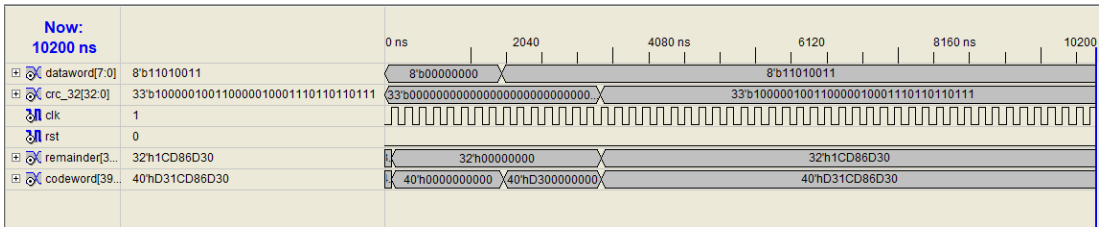


Figure 4. Result of Simulation

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

Figure 4. is the result of simulation data that included in the CRC generator. Input ports are dataword port, crc_32 port, clk and rst port. Output ports are remainder port and codeword port. If rst port is logic low then remainder port will release results of calculation between the data entered (dataword) and the divisor polynomial (crc_32). The value of the dataword used is the same as the table. 1. The codeword port is result of combination of dataword and remainder bits. If rst port is high logic then remainder port will be 0 or no results of calculations. One clock cycle (clk_period) = 10 ns. The calculation results in table 1 and the simulation results yield the same value.

3.2. Design Summary

Results of the resources used from this research can be seen in Figure 5. In Figure 5 has found the number of resources used in a 4-input LUTs is 223 or 1% of the resources available. the number of resources used from the Occupied slice is 114 or 1% of the resources available, the number of resources used on the IOB flip flops is 72 (32 IOB flip flops of remainder bit and 40 IOB flip flops of codeword), it corresponds to the theory of to the theory of linear shift register Method for encoding/decoding that can be seen on the page 12 , the number of resources used on the bonded IOBs is 114 or 45% of the resources available and the number of resources used on BUFGMUXs is 1 or 4% of the resources available.

ENCODERCRC32 Project Status			
Project File:	EncoderCRC32.ise	**Current State:**	Placed and Routed
Module Name:	crc_new	**• Errors:**	No Errors
Target Device:	xc3s1200e-5fg320	**• Warnings:**	1 Warning (0 new, 0 filtered)
Product Version:	ISE, 8.1.03i	**• Updated:**	Mon Mar 2 10:38:23 2020

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	223	17,344	1%	
Logic Distribution				
Number of occupied Slices	114	8,672	1%	
Number of Slices containing only related logic	114	114	100%	
Number of Slices containing unrelated logic	0	114	0%	
Total Number of 4 input LUTs	223	17,344	1%	
Number of bonded IOBs	114	250	45%	
IOB Flip Flops	72			
Number of GCLKs	1	24	4%	
Total equivalent gate count for design	1,917			
Additional JTAG gate count for IOBs	5,472			

Figure 5. Design Summary

3.3. Comparison with pervious research

In this section will discuss the results of this research with previous research. Here are some research journals related to CRC generator. In the table 2 shows the results of the comparison of some methods or previous research.

Vol. 11, No. 2, Oktober 2022, P-ISSN 2089-1245, E-ISSN 2655-4925
DOI: https://doi.org/10.33322/kilat.v11i2.1279

Table 2. Results of The Comparison

Research	Methode 1 (CRC-32)	Methode 2 (CRC-32)	Methode 3 (CRC-32)
Number of 4-input LUTs	340	300	223
Number of Slices	166	194	114

Method 1 is research work of Abdul Rehman Buzdar and Liguo Sun with title "Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths ". In the research journal, CRC generator is designed using modulo-2 division method, Number of LUT is 340 and number slice that used is 166 slices. The simulation used is Xilinx ISE design suit with implementation on Spartan-6 FPGA SP605 Evaluation Kit device.

Mothod 2 is research work of Chaitali Tohgaonkar , Prof. Sanjay B. Tembhurne and Prof. Vipin S. Bhure with title "Design of Parallel CRC Generation for High Speed Application". The research journal CRC generator is designed using modulo-2 division method, Number of LUT is 300 and number of slice that used is 194 slices. The simulation used is ISE Simulator with implementation on Spartan-3 FPGA device.

Method 3 is the result of this research that has been done and simulated using Xilinx ISE Simulator with implementation on Spartan 3E XC3S500E device. Method 1 uses a lot of resources specifically the number of 4-input LUT used is more than method 2 and method 3, whereas the number of slices used is less than method 2. Method 3 has fewer resources than method 1 and method 2. In this research, the same modulo-2 parallel circuit method is used, but in this research the determination of the clock begins with the search for the maximum of data processing, up to 16 bits (consisting of input data and CRC values) and the schematic circuit in this research is simpler than previous research.

4. CONCLUSION

Based on the design results of the CRC-32 generator in this study, it can be concluded that the design has been successfully made and is also in accordance with the expected research objectives. The CRC generator will work when there is incoming data with a data width of 8 bits and then performs CRC-32 IEEE 802.3 calculations. In this study, there are two conditions when the first port is in high condition, the output data for the remaining port will be 0, while if it is in low condition, the remaining port output will be proportional to the results of the calculation of the incoming data. The circuit scheme is simple, the resulting noise is less and the resources used are also less than previous studies with the same CRC method. This study requires 223 4-input LUT resources, 114 Occupied slices, 72 IOB flip flops, 114 bonded IOBs and 1 BUFGMUX, where this study obtained fewer resources than previous research.

REFERENCE

[1] B. Chris, IEEE 802.3 Cyclic Redundancy Check. Xilinx (2001).

[2] W. M. El-Medany (2012). FPGAImplementation of CRC with Error Correction. ICWMC 2012, The Eighth International Conference on Wireless and Mobile Communications.

[3] Pramod S P, Rajagopal A, and Akshay S Kotain (2012). FPGA Implementation of Single Bit Error Correction using CRC. International Journal of Computer Applications, vol. 52, no. 10, pp. 2-6.

[4] D. Ghosh, A. Mitra, A. Mukhopadhyay and A. Dawn (2013). A GENERALIZED CODE FOR COMPUTING CYCLIC REDUNDANCY. International Journal of Students Research in Technology & Management, vol. 1, pp. 192-202.

[5] P. Harika and B. V. V. Satyanarayana (2013). FPGA Based High Speed Parallel Cyclic Redundancy Check. International Journal of Engineering Research & Technology, vol. 2, no. 3.

[6] Chaitali Tohgaonkar , Prof. Sanjay B. Tembhurne and Prof. Vipin S. Bhure (2015). Design of Parallel CRC Generation for High Speed Application. International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 6.

[7] Deepali P. Atrawalkar and Manoj D. Bagde (2017). Design and Simulation of Parallel CRC Generation Architecture for High Speed Application. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 4, no. 7.

[8] Abdul Rehman Buzdar and Liguo Sun (2017). Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths. International Journal of Advanced Computer Science and Applications (IJACSA), vol. 8, no. 2.

[9] N. G. Augoestien and Ryan Aditya (2019). Implementasi Rangkaian CRC (Cyclic Redundancy Check) Generator pada FPGA (Field Programmable Gate Array). Indonesian Journal of Electronics and Instrumentation Systems (IJEIS), vol. 9, pp. 65-74.

[10] Mitra, J. dan Nayak, T., (2017). Reconfigurable very high throughput low latency VLSI (FPGA) design architecture of CRC 32, The VLSI Journal, 56, pp. 1-14.

[11] C. E. Kennedy and M. Mozaffari-Kermani (2015). Generalized parallel CRC computation on FPGA. IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), Halifax, NS, 2015, pp. 107-113.

[12] A. R. Buzdar, L. Sun, R. Kashif, M. W. Azhar and M. I. Khan (2017). Cyclic Redundancy Checking (CRC) Accelerator for Embedded Processor Datapaths. International J. of Advanced Computer Science and Applications, Vol 8, No. 2, pp 321- 325.

[13] Y. Jun, D. Jun, L. Na, G. Yixiong and D. Yin (2010). FPGA-based multi-channel CRC generator implementation. International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT), Shenzhen, 2010, pp. 81-84.

[14] M. F. Hasmi, dan A. G. Keskar (2017). An Optimized FPGA Implementation of CAN 2.0 Protocol Error Detection Circuitry. Indonesian Journal of Electrical Engineering and Computer Science, Vol. 6, No. 3, pp. 602-614.

[15] S.N.V.P.Kumar, S. B. Jyothi, G. K. S. Tejaswi (2017). FPGA Based Design Of Parallel CRCGeneration For High Speed Application. IJSRET (International Journal of Scientific Research Engineering & Technology, Vol 6,