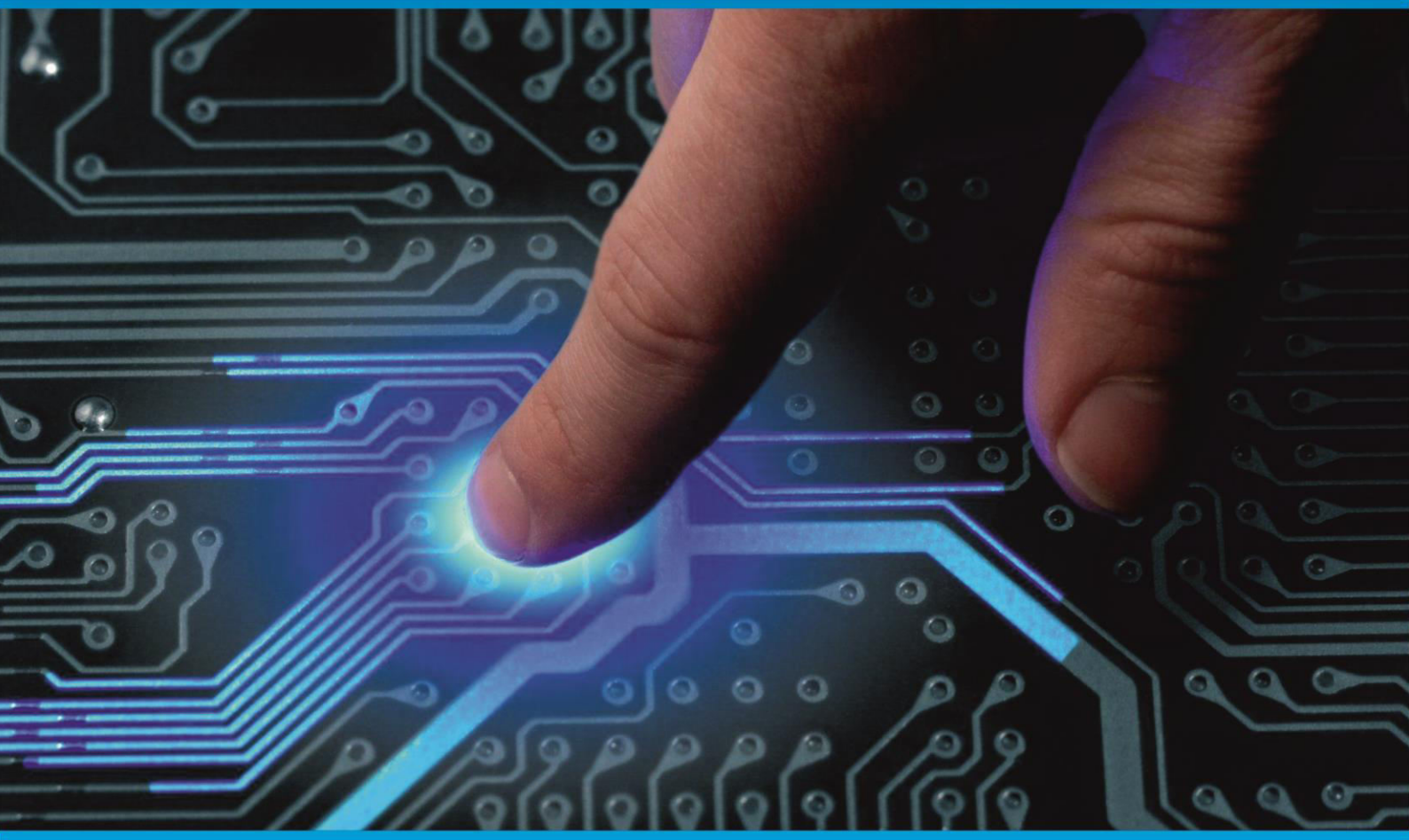
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 3, March 2021

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.488

9940 572 462 | 6381 907 438 | ijircce@gmail.com | www.ijircce.com

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

FPGA Implementation of CRC Using Parallel LFSR

Juthika Mohan M, Dr. Jobymol Jacob

PG Student, Dept. of ECE., Govt. MEC, Kerala Technological University, Kerala, India

Assistant Professor, Dept. of I.T., Govt. MEC, Kerala Technological University, Kerala, India

ABSTRACT: Cyclic redundancy check (CRC) codes are one of the most frequently used methods for error detection in digital data communication and storage. Linear Feedback shift registers are used to implement cyclic redundancy check and BCH encoders. This paper shows the implementation of various CRC generators by adopting the architecture of LFSR and thereby computing the area time product for each CRC generators. The implementation of CRC-3 is carried out using serial and parallel LFSR, where the parallel LFSR will reduce power in computation of CRC.

KEYWORDS: Cyclic redundancy check, Linear feedback shift register

I. INTRODUCTION

Security and high speed data transmission have become a great concernin the field of communication. In order to guard data from unintendedusers and to realize a desirable level of security, several cryptography algorithms supported technologies have been proposed. Linear FeedbackShift Register (LFSR) plays a vital role in the design of such cryptographic algorithms. Due to the simple structure and implementation of theLFSR, it is used for generating random sequences. Cyclic redundancy checkis mostly employed in the field of communication for detecting errors when data istransmitted. CRC generators are necessary for increasing the speed of datatransmission. This paper deals with the implementation of CRC generatorsusing LFSR.

LFSR is a shift registerused to generate binary sequences which are referred toas pseudo random number sequences. The input of LFSR is a linear functionof its previous state. The general form of an LFSR is a shift register with 2or more flip-flop output XOR ed together and feedback as the input to theselective flip-flop of the LFSR. The term linear comes from the fact that thefeedback is completed through an XOR operation which is equivalent to modulo-2addition, where addition is linear operation[1].The main parts of LFSR are the shift register and the feedback function.The main function of shift register is to shift the contents in it to the adjacent position or else out of the register. D flip-flop are used as shift register and Q is the output of each shift register. The feedback path is nothing but the XOR function or the modulo 2 sum operation. The maximum length of Pseudo random numbers generated by LFSR is equivalent to $2^n - 1$ where n is the number of shift register. LFSR sequence depends upon the initial value, tap position and feedback. The initial value of LFSR is called seed value. The polynomial which produces maximum length of sequences is called primitive polynomial. The necessary and not sufficient conditions to be primitive polynomial are number of taps should be even and tap number should be co-prime.

Cyclic redundancy check codes (CRC) are one of the foremost widely used codes for error detection during the generation, transmission, processing or storage of digital data [2]. There are many CRC standards used which differ in their generator polynomial. The algorithms for computing CRC code treat the input data stream as a binary polynomial and compute the CRC for this input by dividing the input polynomial by generator polynomial. The remainder of this division is called the CRC code for the input data and it is transmitted along with this data. At the receiving end, the same CRC algorithm verifies the CRC of the transmitted data. For this purpose, the complete transmitted data including its CRC is divided by the same generator polynomial, and it is verified to be correct if the remainder is zero [3]. Some of the most widely used CRC standards are CRC-8, CRC -16, CRC-32, and CRC-64.CRC generators are implemented in hardware using linear feedback shift registers (LFSRs) and Exclusive OR gates. To implement the CRC generators, the number of flip-flops required is up to the degree of the generator polynomial. These generators handle only one input data bit at a time in each clock cycle. The number of clock cycles to calculate CRC of a message varies linearly with the length of the message in bits. Hence this design isn't so suitable for current high-speed applications. so parallel CRC generators are implemented to process a greater number of inputs at a time.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

II. RELATED WORK

Linear feedback shift register is conventionally used to compute a remainder polynomial by dividing a message polynomial by a generator polynomial, and widely employed in implementing BCH and CRC encoders [6].

A. Serial LFSR Architecture

A basic LFSR architecture for order K_{th} generating polynomial is shown in Figure.1. K denotes the length of the LFSR, i.e., the number of delay elements and $g_0, g_1, g_2, \ldots\ldots, g_K$ represent the coefficients of the characteristic polynomial [7]. The characteristic polynomial of this LFSR is

$$g(x)= g_0 + g_1(x) + g_2(x^2) +...+ g_K (x^K) \qquad (1)$$

For binary BCH codes and CRC, the coefficients are binary [2]. When the input u(x) is given to the most significant tap, this LFSR implements the division of $u(x)x^{nk}$ by g(x). Only the remainder r(x) is of interest to CRC. The coefficients of u(x) are input serially starting with the most significant bit, and r(x) is located in the registers after the last coefficient of u(x) is sent in. The register states at clock cycle t by $r(t) = [r_{n-k-1}(t), r_{n-k-2}(t)\ldots\ldots r_0(t)]'$, where' represents transpose [6]. Let u(t) be the input at clock cycle. Then $r(t + 1) = A \times r(t) + b \times u(t)$ where A is the companion matrix and $b = [g_{n-k-1}, \ldots g_1, g_0]'$.

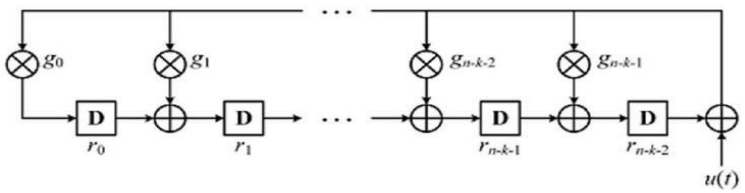


Figure 1: Serial LFSR

$$A = \begin{bmatrix} g_{n-k-1} & 1 & 0 & \cdots & 0 \\ g_{n-k-2} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ g_1 & 0 & 0 & \cdots & 1 \\ g_0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

B. Parallel LFSR

The throughput of the system is limited by the serial computation of the LFSR.We can increase the throughput by modifying the system to process some number of bits in parallel [3]. Many parallel implementations have been proposed based on mathematical deduction [8]-[9]. In these papers, recursive formulations were used to derive parallel CRC architectures. Substituting back to itself p times, it can be derived that $r(t+p) = A^p \times r(t) + B_p \times u_p(t)$ where $B_p = [A^{p-1}b, \ldots. Ab, b]$ and $u_p(t) = [u(t)...u(t + p - 2), u(t + p - 1)]'$. A p-parallel LFSR [7] that processes p bits in each clock cycle can be implemented according to the above equations. The multiplication of A^p is in a feedback loop, and its data path limits the achievable clock frequency of the overall LFSR. To address this issue, transform the state vector as $r(t) = T \times r_T(t)$, where T is non-singular and is referred to as the transformation matrix [6].

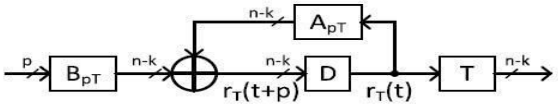


Figure 2: Parallel LFSR

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

Accordingly, the state equation for p-parallel processing becomes $r_T(t + p) = A_{pT} \times r_T(t) + B_{pT} \times u_p(t)$ where $A_{pT} = T^{-1} \times A^p \times T$ and $B_{pT} = T^{-1} \times B_p \times T$, B_{pT} and A_{pT} are also referred to as the prepossessing and feedback matrices, respectively[6]. A block diagram for implementing such a transformed p-parallel LFSR is shown in Figure 2. The T matrix considered in the format of $T = [b_1, A^p b_1, \ldots, A^{(n-k-1)P} b_1]$, so that A_{pT} is a companion matrix, which has only one XOR gate in the data path [2],[7]. It turns out that the overall gate count is also reduced by the transformation, and exhaustive search was done in to find the b_1 leading to minimal gate count. To further reduce the gate count, adopts an upper triangular matrix for T, which has all '1's in the diagonal, and the nonzero entries in row i+1 equal to those in row i shifted to the right by one bit with the last bit eliminated [7].

C. CRC background

Cyclic redundancy check codes (CRC) are one among the foremost widely used codes for error detection during the generation, transmission, processing or storage of digital data [1]. To enable error detection using CRCs, the input stream is split by a typical polynomial, called the "generator polynomial" of a particular CRC standard. The remainder obtained in this division process is called CRC code for the input data. Before transmission or storage of the information, this remainder is attached with the original data. At the receiver, the transmitted data (original data and its CRC) is again, divided by the identical generator polynomial. The remainder of this division should be zero if there were no errors introduced during transmission or storage. Cyclic redundancy codes are the foremost popular for detection of burst errors [2]. CRCs are used fairly often as they are easy to design and are capable of detecting a burst of errors. The type of errors that a particular CRC code will be able to detect depends on its generator polyno mial [2]. Cyclic redundancy check codes detect single bit errors and burst of errors which is adequate to the degree of generator polynomial [3].

III. METHODOLOGY

The section discusses the architectures to be used in implementing LFSR based CRC generators for detecting errors in the digital communication systems.

A. CRC Mathematics

In CRC, generator, input message and its CRC, all are represented in polynomials. Here g(x) is the generator polynomial for a CRC standard of degree k and m(x) is an m-bit message. To find CRC of this message m(x), first m(x) is multiplied by x^k, where it shifts the original message m(x) by k bits and appends k zeros at the end of m(x). This shifted version of message m(x) is then divided by generator polynomial which results in a quotient q(x) and remainder r(x). This remainder itself is taken as CRC of the message m(x). The CRC calculations are done in the Galois field (2) where both addition and subtraction operations are equivalent to exclusive-or (XOR) operation, and multiplication is equivalent to logical AND operation. The CRC calculation can be written in equation form as follows $x^k m(x)g(x) = q(x) \wedge r(x))/g(x)$. Where the degree of the remainder polynomial is always less than the degree of the generator polynomial, k. The term x^k m(x) XORed with r(x) is the original message with its CRC, appended at the end of the message. At the receiving end, the transmitted message $x^k m(x)$ XORed with r(x) is divided by g(x). If the remainder is non-zero, then it indicates that message is received with errors. And if remainder obtained is zero then, it is assumed that there is no error in the message received.

B. Serial CRC

A serial CRC contains shift register and a feedback path. The position of XOR gates is defined by a generator polynomial. The generator polynomial will be in the form, $P(x)=p_0+p_1(x^1) +.... +p_n (x^n)$ where n corresponds to the number of flip-flops in the CRC circuits. The coefficients $p_n = 1$ and $p_n = 0$ decides the connection and disconnection between the XOR and feedback path. The state of the flip flops is represented as $C_0(t)$, $C_1(t)$, ..., $C_n(t)$, where $C_i(t)$ represents the state of i_{th} flip-flop. The CRC, C(x) is calculated by dividing $x_n K(x)$ by generator polynomial P(x). CRC is appended with message sequence in order to spot the transmission errors [11]. Figure 3 shows the architecture of a serial CRC. A serial CRC processes one bit at a time. A k bit message is processed and CRC value is obtained after k clock cycles and data stream is serial.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

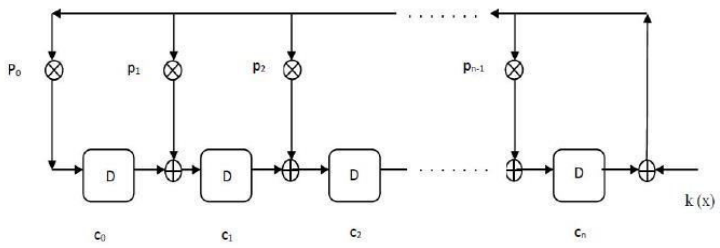


Figure 3: Serial CRC

C. Parallel CRC

In order to realize high speed communication [11], a parallel arrangement of serial CRC is employed during which the message is split into blocks of length k/m bits each. When comparing serial CRC, a parallel CRC can calculate the CRC in k/m clock cycles, and hence throughput may be increased. In Figure 4, a parallel CRC structure is shown which receives 64-bit message (k=16), which is split into four blocks (m=4). For processing the message, four execution units are there, into which each message block is given. After 16 clock cycles (k/m=64/4=16), four 16-bit CRC values (if degree n of generator polynomial is 16) are obtained. The final 16-bit CRC are going to be the XOR of those four CRC values. Parallel processing increases the throughput rate moreover because the number of bits which will be processed at a time.

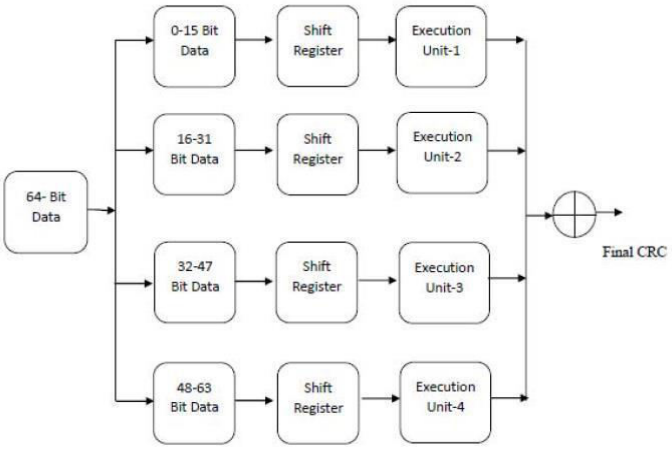


Figure 4: Parallel CRC

D. CRC Generators

This paper implements CRC-12, CRC-16 and CRC-32 using the polynomials provided in the table 1.

TABLE 1: Generator Polynomials used in CRC

CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
SDLC	$x^{16}+x^{12}+x^5+1$
CRC-16 REVERSE	$x^{16}+x^{14}+x+1$
SDLC REVERSE	$x^{16}+x^{11}+x^4+1$
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

The LFSR based serial CRC-3, CRC-12, CRC-16, CRC-32 and parallel CRC-3 generators are implemented in Verilog hardware description language. The functionality of the design is verified in Xilinx ISE 14.7 All the generators have the same ports such as

- clk - clock in which the entire CRC generator is synchronized simultaneously.
- rst- when this signal goes high all the flip-flops within the CRC generator is reset-ed.
- datain - message bits are sent one bit per clock cycle.
- crc out - crc output code for every generator is obtained at this port.

IV. RESULTS AND DISCUSSION

THIS SECTION DISCUSSES THE RESULTS AND SIMULATIONS OBTAINED IN IMPLEMENTING THE LFSR BASED CRC GENERATORS USING XILINX ISE.

A. *Simulation results of serial CRC-3*

A polynomial of degree 3 is used for the development of serial and parallelCRCs. Xilinx 14.7 was utilized for obtaining simulation results of the design. A serial CRC is designed for the polynomial, $P(x)=1+x+x^3$. The message inputted is 100111101 at each clock cycle. The seed value is given as 111. After nine clock cycles the CRC is obtained which is used as the error detecting code. Figure 5 shows the simulation results of CRC-3 implemented in method 1.

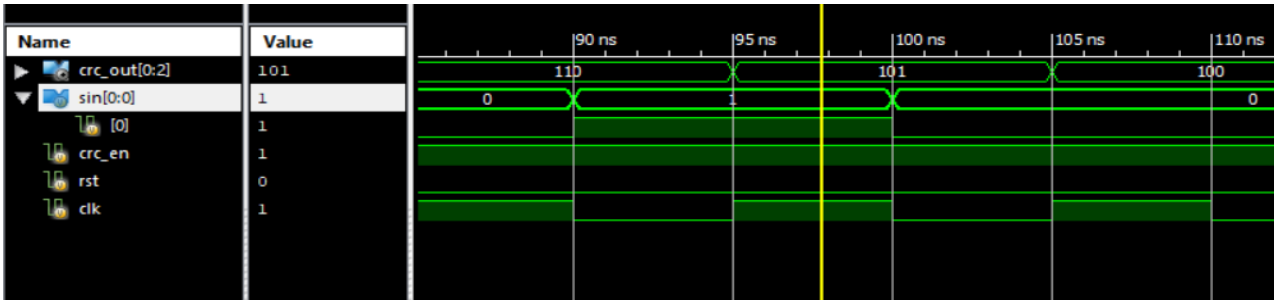


Figure 5 : Simulation results of serial CRC-3

B. *Simulation results of parallel CRC-3*

In the parallel CRC architecture, three serial CRCs are kept parallel. Each of the CRCs are having same generator polynomial $P(x) = 1+x+x^3$. A 9-bit message input message = 100111101 is given and each serial CRC circuit is initialized to a seed value 111. The message is divided as three blocks, 100, 111 and 101 and each CRC will receive each bit of message block on each clock cycle. After 3 clock cycles, four 3-bit CRC values will be obtained. The final CRC will be the XOR of those three CRC values. The four CRCvalues obtained are, c1=111, c2=010, and c3=100. The final 3-bit CRC value obtained is 001. Figure 6 shows the simulation results of CRC-3 implemented in parallel method.

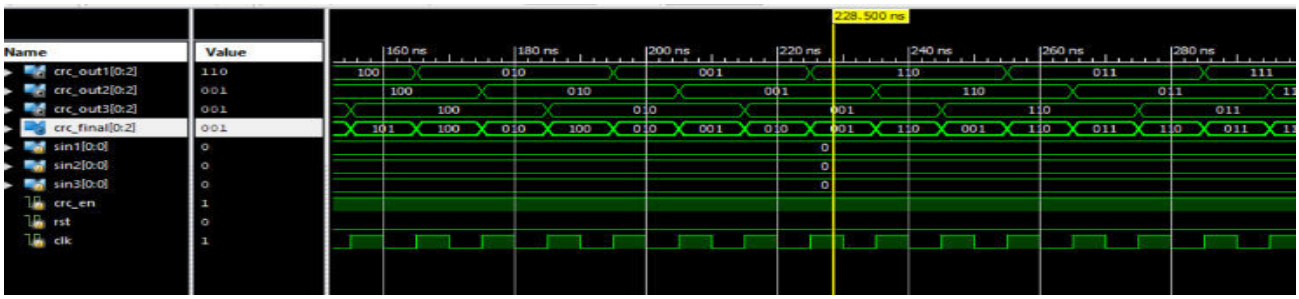


Figure 6: Simulation results of parallel CRC-3

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

C. *Schematic diagrams*

The Figure 7 and Figure 8 shows the schematic of serial and parallel CRC-3 architectures after synthesis using Xilinx. It gives a clear analysis of components involved in the architecture.

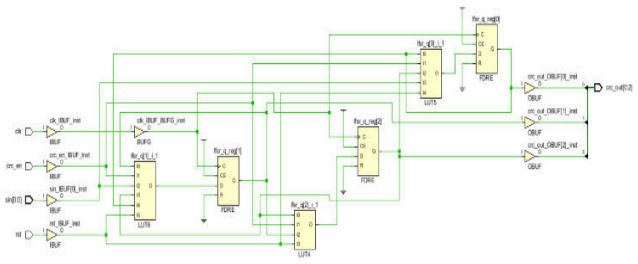


Figure 7: Schematic of synthesized serial CRC-3

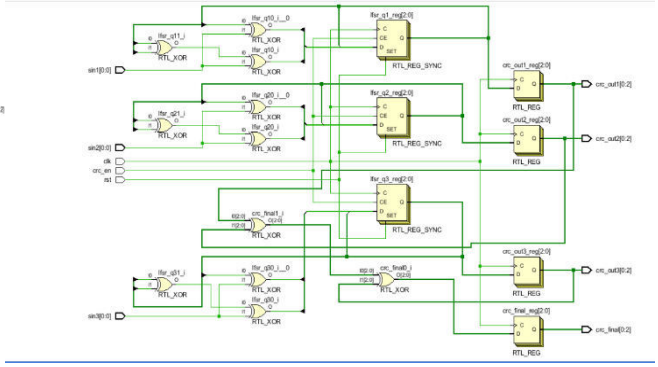


Figure 8: Schematic of synthesized parallel CRC-3

D. *Simulation results of LFSR based CRC generators*

Simulation results of CRC-12, CRC-16, CRC-32 obtained are shown in this section. A 32-degree generator polynomial is used for implementing the CRC-32. 1-bit data is inputted to the serial LFSR and based on the tap positions, feedback is provided and corresponding CRC is obtained at the output.

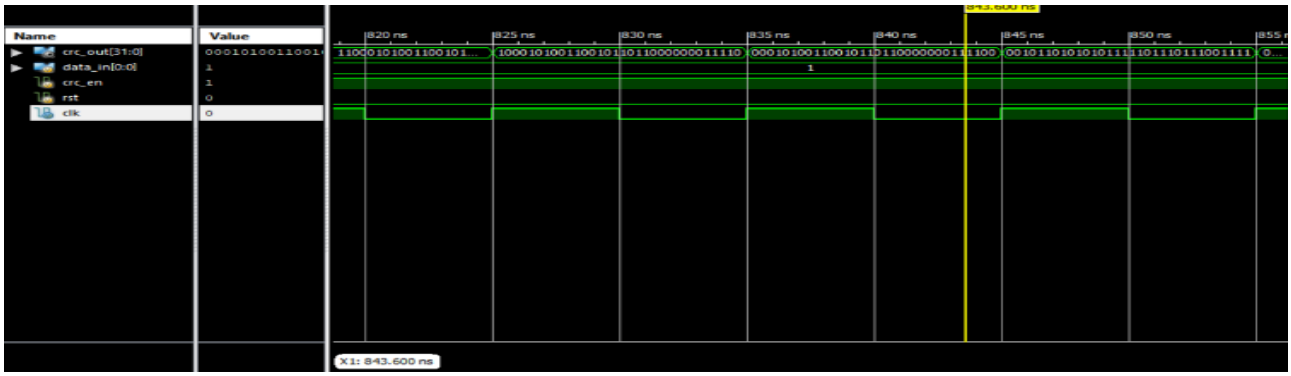


Figure 9: Simulation results of CRC-32

A 16-degree generator polynomial is used for implementing the CRC-16. 1-bit data is inputted to the serial LFSR and based on the tap positions, feedback is provided and corresponding CRC is obtained at the output.

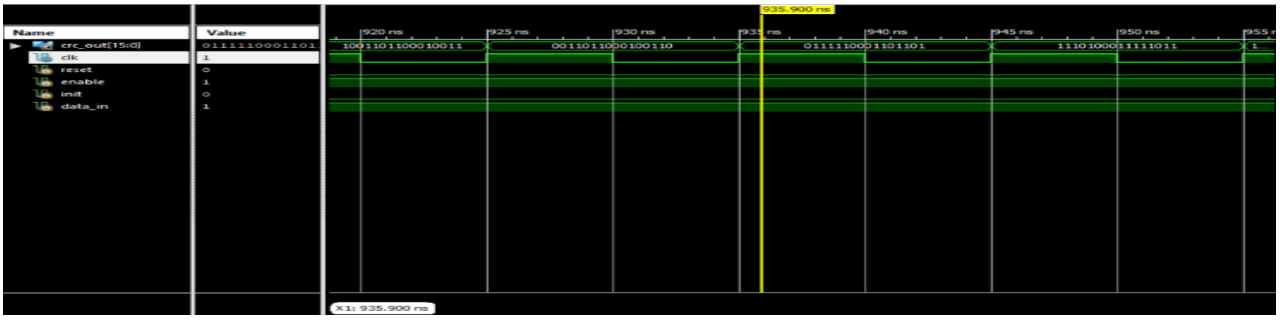


Figure 10: Simulation results of CRC-16

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

A 12-degree generator polynomial is used for implementing the CRC-12. 1-bit data is inputted to the serial LFSR and based on the tap positions, feedback is provided and corresponding CRC is obtained at the output.

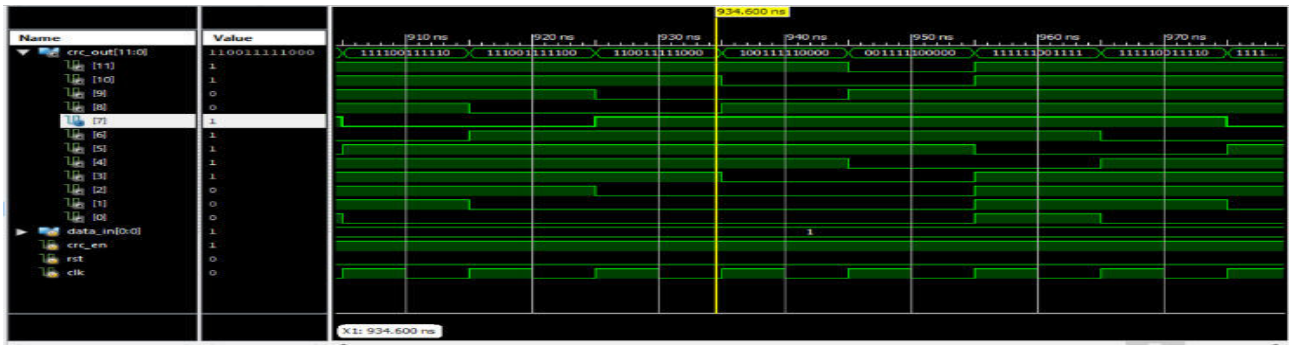


Figure 11: Simulation results of CRC-12

E. *Performance evaluation*

This section discusses the analysis of power, delay, resource utilization of serial and parallel LFSR based CRC generator and also calculate the area time product (ATP) of each serial CRC generators.

a) ATP calculation: Area time product (ATP) is calculated for each CRC generators like CRC12,CRC16, CRC32. The equation used for calculating ATP is ATP = (1.5*DE+XOR) CPD, DE is the number of delay elements in the CRC. XOR is the number of XOR gates used in CRC implementation. CPD is critical path delay [7].

TABLE 2

Area Time Product Calculation Table

CRC -polynomial	XOR	DE	CPD	ATP
crc 12	9	12	5	135
CRC-16	10	16	6	204
CRC-32	27	32	6	450

ATP calculation for each CRC generators its shown. As degree of the generator polynomial increases the area-time product alsoincreases and highest ATP is obtained for CRC-32.

b) Resource utilization:

Resource utilization determines the amount of device resources such as block RAMs, flip-flops, and LUTs an IP core uses when it is programmed into an FPGA device.

TABLE 3

Resource Utilization

	CRC-3	CRC-12	CRC-16	CRC-32
Registers	3	12	16	32
Flip-Flops	3	12	16	32
Delay	0.855ns	0.868ns	0.850ns	0.850ns
XOR	3	9	10	27
Slice LUT used	2	5	16	14

This table 3 list the number of registers, flip-flops, XOR gates used and LUT utilized for implementing CRC generators such as CRC-3, CRC-12, CRC-16 and CRC-32.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 9, Issue 3, March 2021 ||

| DOI: 10.15680/IJIRCCE.2021.0903009 |

c) Power Analysis:

Total on-chip-power is estimated with the help of Xilinx XPower Analyzer estimation tool. The two versions of LFSR based CRC-3 generators were implemented. The first version was serial CRC-3 generator. The estimated total on chip power was 0.82W and the second version was parallel CRC-3 generator implementation, its power estimated was 0.14W. When serial CRC is converted to parallel CRC, there is small reduction in the power consumption even though the area increases and circuit complexity increases.

V. CONCLUSION

In this paper, the successful design and implementation of LFSR based CRC generators was carried out. Two versions of CRC-3 generators were implemented. First version is made by simple serial architecture of linear feedback shift register in which 1-bit data input is given at every clockcycle. In second version, the parallel architecture of LFSR is utilized where the data messages are divided into small chunks of data and applied to each block of the parallel LFSR, each CRC out from the parallel LFSR are XOR-ed to get the final CRC. The implementation of both versions CRC-3 generators and CRC-12, CRC-16, CRC-32 were also carried out in Verilog HDL. The designs were synthesized in Xilinx ISE 14.7 platform. The performance, area-time product, resource utilization and power consumption were evaluated. There is a reduction in power consumption when parallel architecture of LFSR for implementing a CRC-3 generator is used when compared to serial CRC-3. Hence low power parallel LFSR is better for implementing CRC encoders for data transmission. State space transformation technique can be used in future to reduce the power for long LFSR.

REFERENCES

1. Sahithi, M., et al. "Implementation of random number generator using LFSR for high secured multi purpose applications." International Journal of Computer Science and Information Technologies, vol. 3, no.1, pp : 3287-3290, 2012.
2. Kounavis, Michael E., and Frank L. Berry. "Novel table lookup-based algorithms for high-performance CRC generation." IEEE transactions on Computers, vol. 57, no.11, pp : 1550-1560, 2008.
3. Perez, Aram. "Byte-wise CRC calculations." IEEE Micro, vol. 3, pp : 40-50, 1983.
4. Bose, Raj Chandra, and Dwijendra K. Ray-Chaudhuri. "On a class of error correcting binary group codes." Information and control, vol. 3, no. 1, pp : 68-79, 1968.
5. Burton, H. "Inversionless decoding of binary BCH codes." IEEE Transactions on Information Theory, vol. 17, no. 4, pp : 464-466, 1971.
6. Ayinala, Manohar, and Keshab K. Parhi. "High-speed parallel architectures for linear feedback shift registers." IEEE transactions on signal processing, vol. 59, no. 9, pp : 4459-4469, 2011.
7. Zhang, Xinmiao. "A low-power parallel architecture for linear feedback shift registers." IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 66, no. 3, pp : 412-416, 2018.
8. Kennedy, Christopher, and Arash Reyhani-Masoleh. "High-speed CRC computations using improved state-space transformations." 2009 IEEE International Conference on Electro/Information Technology, pp : 9-14, 2009.
9. Hu, Guanghui, Jin Sha, and Zhongfeng Wang. "High-speed parallel LFSR architectures based on improved state-space transformations." IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 3, pp : 1159-1163, 2016.
10. Pei, T-B., and Charles Zukowski. "High-speed parallel CRC circuits in VLSI." IEEE Transactions on Communications, vol. 40, no. 4, pp : 653-657, 1992.
11. Prabha, Lekshmi S., and R. S. Geethu. "Architecture of parallel CRC encoder using state space transformations." 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), pp : 1-7, 2018.

BIOGRAPHY

Juthika Mohan Misa PG Student, *Electronics and communication Engineering* Department,*Govt Model Engineering College,* KeralaTechnical University, Kerala, India

.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.488

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर
NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details