JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Implementation of Reliable CRC Error Detection For Highly Flexible and Scalable Digit Serial / Parallel Finite Field Multiplier on FPGA for Cryptography Applications

[1]U. Hemanth, [2]U.V. Ratna Kumari,
[1]M. Tech Student, [2]Professor
[1]Electronics and Communication Engineering,
[1]University College of Engineering JNTUK, Kakinada, India

Abstract: The research into finite field multiplication has gained a significant amount of interest due to its prominent applications in cryptography and error detection codes. This cryptographic arithmetic technique is a laborious task that is not only costly but also takes a significant amount of time. It requires millions of gates. The research conducted for this particular investigation of a cryptographic algorithm suggests an effective hardware architecture for all cryptographic applications that is based on cyclic redundancy check (CRC) as an error detection method. This work introduces an architecture for digit serial multiplication in finite fields GF, with applications to cryptography. GF stands for Galios finite fields (2^m). The steps of multiplication and degree reduction are alternated within the framework of the suggested system, which is based on the representation of polynomial bases. A multiplier with M bits may be used for calculations in any binary field of order, and it is compatible with any irreducible polynomial. Therefore, the Versatile and Scalable Digit Serial/Parallel Multiplier Architecture for Finite Field will be merged with the Reliable CRC implementation and design of GF(2^3) = 8-Bit, GF(2^4) = 16-Bit, and GF(2^5) = 32-Bit based CRC Error detection. This work was created using Verilog HDL, and it was then synthesized using Vertex-5 FPGA. After that, all of the parameters like area, delay, and power are obtained.

Index Terms **- Cyclic redundancy check (CRC), Finite Field Multiplier, Galois Field (GF).**

I. Introduction:

Among the many finite-field operations that are used in the designs of many current applications and systems, finite-field multiplication has garnered a significant amount of attention due to its prominence. The multiplication modulo is an irreducible polynomial that is used to define the finite field. Finite-field multipliers conduct multiplication using this modulo [1]. In post quantum cryptography, also known as PQC, the inputs might be quite huge, and the finite-field multipliers can need millions of logic gates to function properly. Therefore, it is a difficult challenge to construct such systems that are robust to natural and deliberate defects. As a result, research has concentrated on methods to minimize mistakes and get better reliability while maintaining an acceptable level of overhead. In addition, there has been past work done on defending against fault assaults and providing dependability. This study used error-detection systems of number theoretic transform (NTT) in order to identify both permanent and transient faults [2]. One of the tasks that it carried out was defect detection for stateless hash-based PQC signatures. In addition, error-detection hash trees for stateless hash-based signatures are being suggested as a means of making such schemes more dependable against natural faults and aiding in the protection of them against intentional faults. Through re-computing with swapped cipher text and additional authenticated blocks, algorithm-oblivious constructions are proposed. These constructions, which can be applied to Galois counter mode (GCM) architectures by employing various finite-field multipliers in GF(2^{128}), can be applied to the algorithm. Several error-detection checksum codes and spatial and temporal redundancies are included in these countermeasures as part of the NTRU encryption technique. The suggested error-detection structures that were developed are customized for the Luov cryptographic algorithm; nevertheless, they are versatile enough to be used with a variety of PQC algorithms that make use of finite-field multipliers. Our suggested architectures make use of cyclic redundancy check (CRC) error-detection algorithms to ensure that they are aware of overhead and have high error coverage [3], [4].

These finite-field multiplications are very difficult to perform and need a footprint covering a large region. Because of this, the implementation of such designs that are robust to both natural and purposeful defects is a difficult undertaking. The purpose of this research is to develop error-detection schemes that can be adapted to other applications and cryptographic algorithms whose building blocks require finite-field multiplications. These error-detection schemes will protect finite-field multipliers used in cryptosystems from both natural faults and fault injections. One example of such a system is the Luov algorithm [5]. It is important to note that the proposed error-detection schemes are adaptable. In this work, Section II presents the Finite Field Multiplier with Proposed Error Detection Schemes Based on CRC, Section III presents the Versatile and Scalable Digit Serial Parallel Multiplier, and Section IV describes the Novelty of CRC Integration. In the next section, "Section V," we will examine the results and the implementation, and Section VI will provide the conclusion of this effort.

II. Finite Field Multiplier with Proposed Error Detection Scheme based on CRC:

Code-based, hash-based, isogeny-based, lattice-based, and multivariate-quadratic equation-based cryptosystems are the five most common types of PQC algorithm classes. The difficulty of decoding in a linear error-correcting code is what sets code-based cryptography apart from other forms of encryption; this is how its security is maintained. Hash-based cryptography is a kind of cryptography that generates signature algorithms by basing those algorithms on the safety of a particular cryptographic hash function. The difficulty of solving the task of finding an isogeny between two super singular elliptic curves serves as the foundation for the security provided by cryptography that is based on isogeny. The creation of a public-key cryptosystem that is based on lattices is within the capabilities of the cryptography that is based on lattices. Last but not least, the safety of cryptography that is based on multivariate-quadratic equations is determined by how difficult it is to solve a system of multivariate polynomials over a finite field [6]. These types of cryptographic algorithms make advantage of high field sizes in order to deliver the required levels of security. There is a constraint placed on the coefficients of the public key in the Luov cryptosystem, which is a multivariate public key encryption method and an adaption of the unbalanced oil and vinegar (UOV) signature scheme. The strategy, on the other hand, makes use of two finite fields: the first is a binary field with two elements, and the second is that field's extension with degree m. The binary field is denoted by F_2, and its extension to degree m. The central map F:$F_{2^m}^n \longrightarrow F_{2^m}^o$ is a quadratic map, where o and v full fill the equation n = o + v, $\alpha_{i,j,k}$, $\beta_{i,k}$ and γ_k are selected from the base field F_2, and whose components f_1,..., f_o are in the form of Equation1.

$$fk(x) = \sum_{i=1}^{v} \sum_{j=1}^{n} \propto_{i,j,k} x_i x_j + \sum_{i=1}^{n} \beta_{i,k} x1 + \gamma_k \quad (1)$$

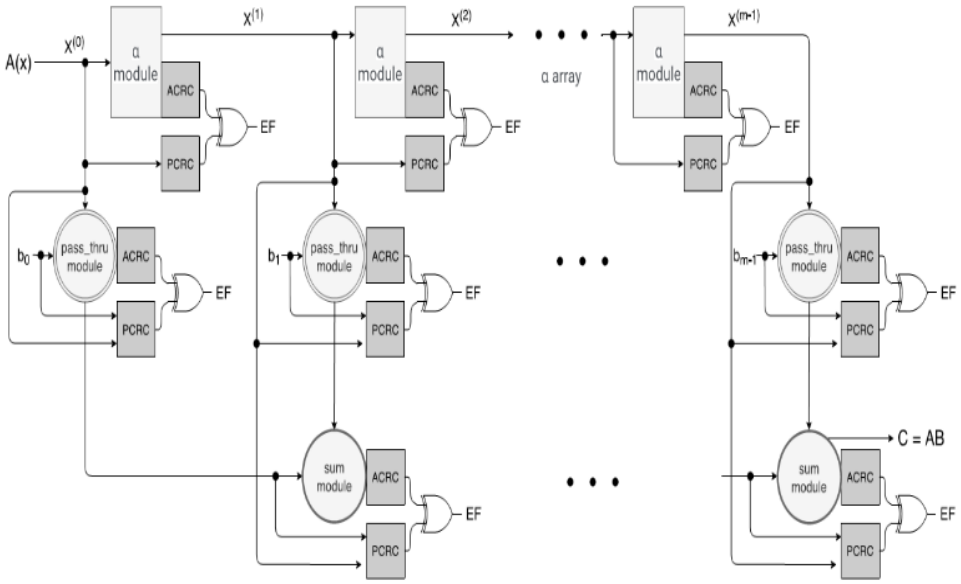

Fig.1: Finite Field Multiplier with the proposed error detection schemes based on CRC

Our objective is to derive error-detection strategies that are superior than parity signatures in terms of error coverage and breadth, and then investigate how such strategies might be used to the Luov algorithm. As a result, we develop CRC signatures for the finite-field multipliers that are utilized in the Luov method and apply them. This would be a step forward toward detecting natural and malicious intelligent faults, particularly and as discussed in this brief, considering both primitive and standardized CRCs with different fault multiplicity coverage [7]. This would be a step forward toward detecting natural and malicious intelligent faults. CRC is based on the idea of cyclic error-correcting codes, which was initially introduced in 1961 when CRC was first suggested. It is necessary to have a generator polynomial g(x) in order to construct CRC. The message is what is used as the dividend, the quotient is thrown away, and the remainder is what is used to make the result. In the case of CRC, the data is supplemented with a

predetermined quantity of check bits, and these check bits are analyzed when the output is obtained in order to identify any mistakes. Figure 1 displays the whole of the finite-field multiplier along with our various error-detection algorithms [8]. The actual CRC (ACRC) and predicted CRC (PCRC) are abbreviations for the ACRC signatures and PCRC signatures, respectively. In Figure 1, only one EF is displayed for clarity's sake; however, for CRC-5, which is the case study recommended in this short, five EFs are calculated on each module. This is because CRC-5 is a complex case study. Figure 2 provides a more in-depth look at the module in order to provide light on the functioning of the proposed CRC signatures in each finite-field multiplier.

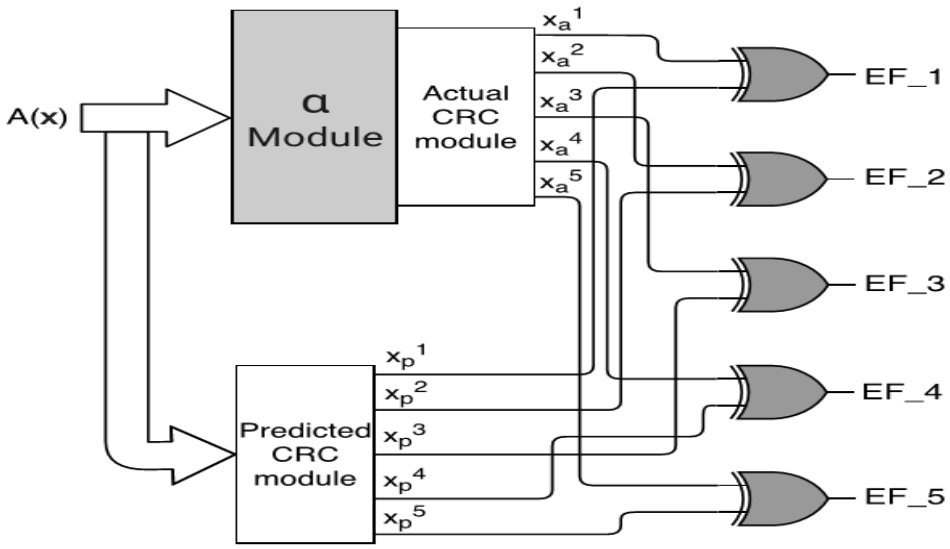

Fig.2 Proposed Error Detection Construction for α module

The procedure for the sum and pass-thru modules is the same as the one outlined for the parity signatures procedure. When it comes to the sum module of CRC, p_X is equivalent to the sum of the parity bits of the input components A and B in GF(2m). Specifically, $p_X = p_A + p_B$ describes this relationship. In addition, for the pass-thru module in CRC, $p_X = b \bullet p_A$, where b is a component in GF(2^m).From Equation2 α-module is obtained. Instead of adding up all of the bits, it examines n bits at a time in the modules that are responsible for the sum and the pass-thru for any other CRC-n method. The α module is given by the Equation2 for which certain number of derivations are required in order to include CRC-n into the module [9].

$$A(x).x = a_{m-1}.x^m + a_{m-2}.x^{m-1} + \cdots + a_0.x \quad (2)$$

III. Versatile and Scalable Digit Serial Parallel multiplier:

Recent years have seen a rise in the significance of finite fields due to the many domains where they have found utility, most notably combinatorics, coding theory, and cryptography. In particular, the "carry-free" arithmetic of binary extension fields GF(2^m) is highly appealing from an implementation perspective. Polynomial bases, normal bases, and dual bases are all equivalent representations of the field elements, which is another benefit of GF(2^m). There is considerable leeway in the degree m of the binary extension field, which is determined by the specific uses. Finite fields of relatively high order are employed in elliptic curve cryptography; m is typically a prime between 160 and 200, but may be more than 500. The efficient hardware and software implementation of finite field arithmetic is the subject of a large body of literature. In this study, we restrict ourselves to polynomial bases and solely think about binary fields of GF(2^m). General-purpose microprocessors lack polynomial arithmetic instructions [10]. A missing instruction for multiplying two binary polynomials would considerably speed up large-order GF(2^m) multiplication. Polynomial base multiplication in GF(2^m) uses shift and XOR instructions or look-up tables. The bit-level technique is inefficient, whereas the lookup table method needs memory. Both strategies are impractical for smart cards and other low-power devices. A smart card co-processor for field arithmetic is a superior approach. A finite field multiplier for smart cards must be adaptable and scalable.

3.1 Versatility: For a GF(2^m) multiplier to be versatile if it operates over a broad variety of finite fields. This means that a multiplier that was initially sized for a data route precision of M bits should also be useable for fields of a lower order. The choice of field order has an impact on both the total number of points in the elliptic curve group as well as the degree of complexity of the discrete logarithm problem that corresponds to it. An adaptable multiplier for GF(2^m) makes it possible to choose the degree m of the field in accordance with the specific safety standards that are wanted. For instance, a multiplier that was developed for M = 256 bits is capable of performing multiplications in GF(2^{233}), GF(2^{193}), as well as GF (2^{163}).

3.2 Scalability: The performance of digit-serial multiplier designs is scalable due to the fact that the digit size, denoted by d, may be chosen according to the performance/area trade-off that the user desires [11]. The area-complexity of a digit serial multiplier is O(d •m), and it can calculate a multiplication in GF(2^m) in m clock cycles. Since the critical path of the mod p(t) operation varies

linearly on the digit-size d, the reduction modulo the irreducible polynomial p(t) becomes a performance barrier for high digit-sizes d. This is because the reduction is an irreducible polynomial.

Following is a concise summary of the binary method for performing multiplication in GF(2^m) with a polynomial base representation. First, we go through some of the notations that will be used in both this part and the ones that will follow it. Let's say that a binary extension field is denoted by GF(2^m). Any element of GF(2^m) may be stated as a polynomial a(t) of degree m-1 with coefficients in GF(2), which can be written as "a(t) of degree m-1 with coefficients in GF(2)."

$$a(t) = \sum_{i=0}^{m-1} a_i . t^i = a_{m-1} . t^{m-1} + \ldots + a_2 . t^2 + a_1 . t + a_0 \quad (3)$$

The value of a_i is either 0 or 1. Given that each coefficient of a_i may take on the value 0 or 1, it is possible for us to describe the field element a(t) using the notation for bit strings as follows: ($a_{m-1},\ldots a_2,a_1,a_0$). When employing the polynomial basis representation, the multiplication of field elements a(t), b(t) GF(2^m) is performed modulo an irreducible polynomial p(t) of degree m over GF(2) [12]. This is done in order to maintain the integrity of the polynomial basis representation in Equation3 and the architecture of bit serial multiplier shown in Figure 3 and it is synthesized in Xilinx Vertex-5 FPGA with Mod px, Reduced Polynomial and generation partial products.

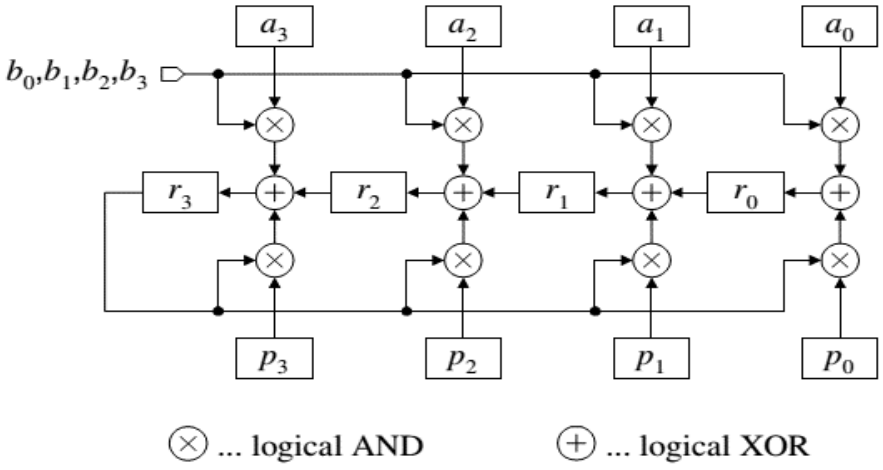

Fig.3 Bit Serial Multiplier for GF(2^4)

IV. Programmable CRC Implementation:

The mathematical concept of polynomial division, modulo two, provides the foundation for the computation of a cyclic redundancy check. In application, it is quite similar to long division of the binary message string by the "generator polynomial" string, with a set number of zeroes added; the only difference is that exclusive or operations substitute subtraction operations. This type of division can be efficiently realized in hardware by a modified shift register[1,] and in software by a series of equivalent algorithms, beginning with simple code that is close to the mathematics and becoming faster (and arguably more obfuscated) through byte-wise parallelism and space–time tradeoffs. The polynomial division method is expanded upon by a number of CRC standards, each of which specifies an initial shift register value, a final Exclusive-Or step, and, most importantly, a bit ordering. As a direct consequence of this, the code that is actually used in reality deviates in a manner that is inconsistent with "pure" division, and the register may move either left or right. Figure 4 depicts the novelty-based CRC N-Size architecture; the way in which it works at the iteration level is dependent upon the bit size [13]. Initial input given to left shift by 1, which means the input data multiplied by 2, similarly CRC polynomial code also left shift by 1, both input data and polynomial code data will be XOR'ed, AND'ed finally we get the sign bit comparisons value, if its equal 1 we will XOR the CRC code to CRC Polynomial for more encryptions, otherwise the original XOR'ed data directly will be shared, then it follows the next iteration The number of times we go through this process is determined by both the bit size and the encryption size. Due to the fact that this work proposes error-detection designs, such architectures are based on CRC-8 since there are 8 iterations of signatures,software implementations has been done for the purpose of verification. In addition, investigation and research has been done on both primitive and standardised generator polynomials for CRC-8 and compared the level of difficulty that each of them presented to us. The results shows high error coverage has been achieved while maintaining an acceptable level of cost by integrating the suggested error-detection techniques into the original finite-field multipliers of Luov's algorithm.

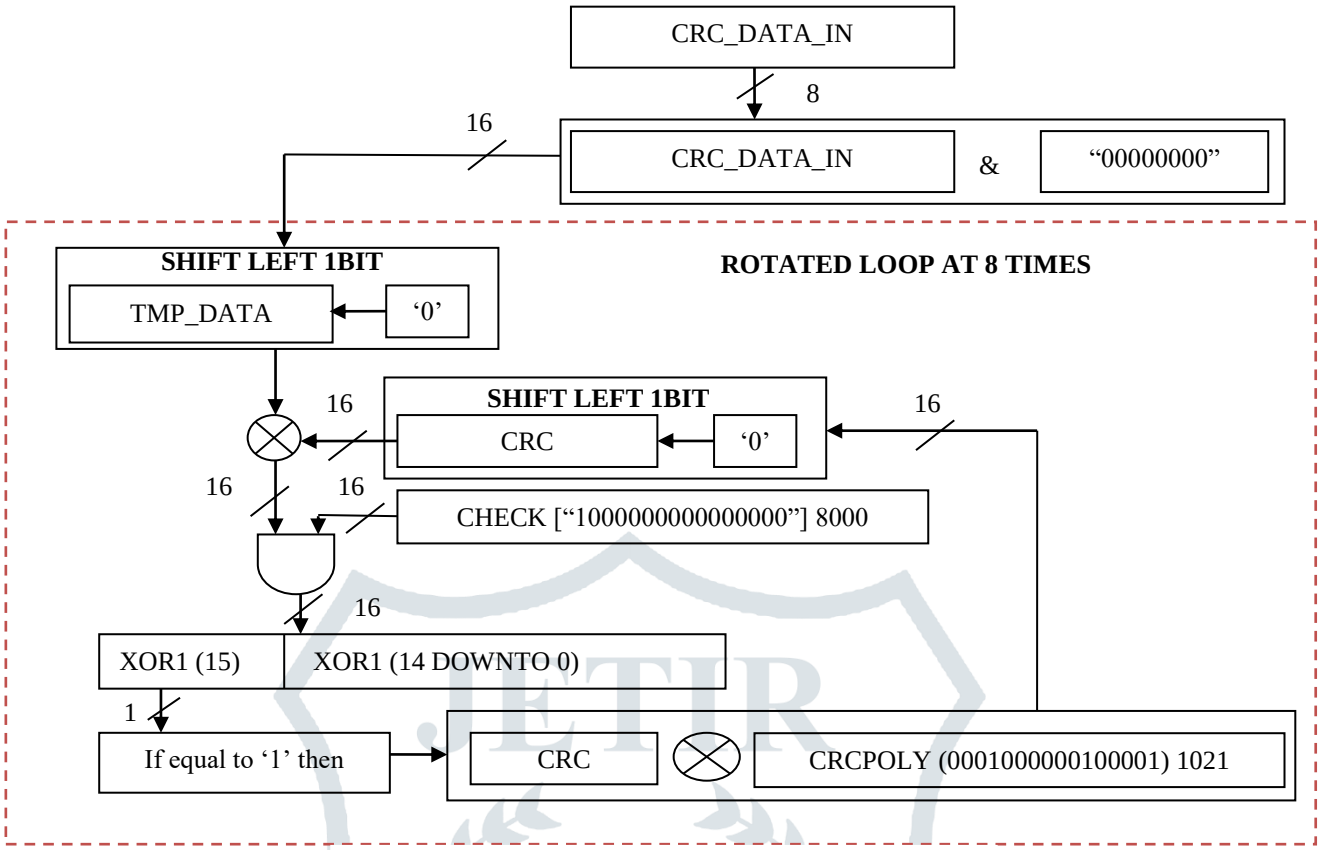

Fig.4 Novel Architecture of CRC Implementation

V. RESULTS AND DISCUSSION

The procedure known as "finite-field multiplication" is quite expensive and demands a big footprint. In order to demonstrate that the suggested error-detection systems give good error coverage with reasonable cost, developed the Luov polynomial generation algorithm. This kind of implementation results in the polynomial $p(x) = a_{m-1}\, x^{m-1} + ...+ a_1 x + a_0$, which necessitates m-1 finite-field multiplications as well as m-1 XOR operations. As was said before, each and every finite-field multiplication makes use of three distinct modules that are referred to as the α, sum, and pass-thru modules. In order to carry out each finite-field multiplication, a total of m-1 α modules, m-1 sum modules, and m pass-thru modules are required. In addition, an XOR operation requires a total of m-1 sum modules to be carried out successfully [14]. Here, we have developed an 8-bit, 16-bit, and 32-bit Reliable CRC Implementation using a Versatile and Scalable Bit Serial Multiplier. Figure 5 displays the RTL Schematic for the 32-bit architecture. Figure 6 shows the simulation results of 32-bit reliable CRC based error detection. Utilization of Resources in Xilinx Vertex-5 for architectures 8-bit,16-bit and 32-bit Reliable CRC Based Error Detection & Construction for Finite Field Multipliers were brought up in Table 1.

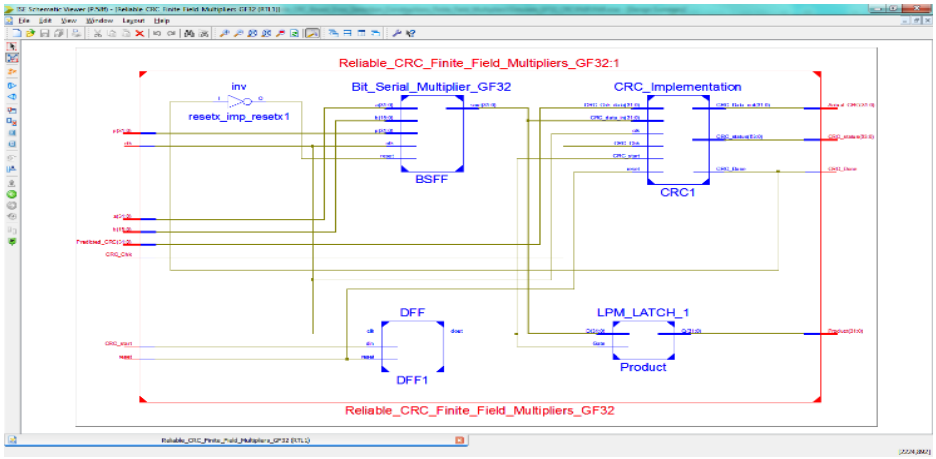

Fig.5 RTL Schematic of 32-bit Reliable CRC Based Error Detection

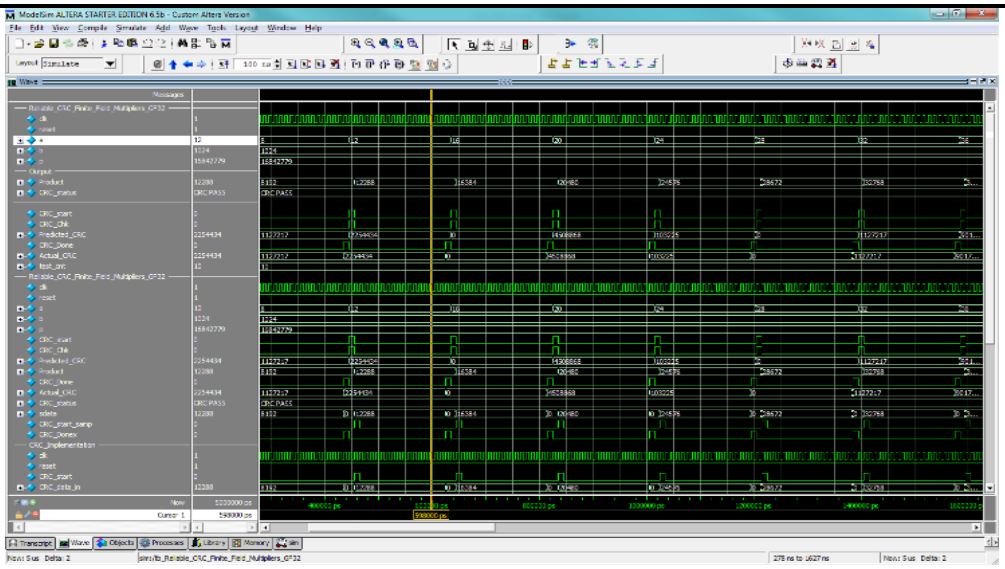

Fig.6 Simulation results of 32-bit Reliable CRC Based Error Detection

Table1: Utilization of resources for architectures 8-bit,16-bit and 32-bit Reliable CRC Based Error Detection & Construction for Finite Field Multipliers on Xilinx Vertex-5 FPGA

Parameter	Architecture		
	GF8 - CRC	***GF16 - CRC***	***GF32 - CRC***
Number of Slice Registers	86	134	214
Number of Slice LUTs	140	247	616
Number of Occupied Slices	43	85	229
IOBs	113	157	245
Delay (ns)	3.102	3.133	3.665
Power (mW)	0.3096	0.3199	0.3300

VI. Conclusion

In this work, error-detection schemes have been derived for the finite-field multipliers that are used in post quantum cryptographic algorithms. The error-detection architectures that have been presented in this study are based on CRC-8 signatures, and for the purpose of verification, we have carried out software implementations of these signatures. In addition, investigation and research has been done on both primitive and standardized generator polynomials for CRC-8, and compared the level of difficulty that each of them presented to us. Applications to cryptography are discussed throughout this paper, which presents an architecture for digit serial multiplication in finite fields GF. GF stands for Galois finite fields (2^m). Within the scope of the system that has been provided, the stages of multiplication and degree reduction are alternated with one another. This system is based on the representation of polynomial bases. Calculations may be performed in any binary field of order using a multiplier that has M bits, and it is compatible with any irreducible polynomials that may be utilized. As a result, the Reliable CRC implementation and design of GF(2^3) = 8-Bit, GF(2^4) = 16-Bit, and GF(2^5) = 32-Bit based CRC Error detection will be combined with the Versatile and Scalable Digit Serial/Parallel Multiplier Architecture for Finite Field. The HDL language known as Verilog was used in the creation of this work, and Vertex-5 FPGA was used for the synthesis. After then, each of the parameters was evaluated based on its area, the amount of time it required, and its power.

REFERENCES

[1] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted MOSFET's with very small physical dimensions," IEEE J. Solid-State Circuits, vol. 9, no. 5, pp. 256–268, Oct. 1974.

[2] M. E. Kounavis and F. L. Berry, "Novel Table Lookup-Based Algorithms for High-Performance CRC Generation," IEEE Trans. Comput., vol. 57, no. 11, pp. 1550–1560, Nov. 2008.

[3] A. Akagic and H. Amano, "High-speed fully-adaptable CRC accelerators," IEICE Trans. Inf.& Syst., vol. 96, no. 6, pp. 1299–1308, 2013.

[4] L. Kekely, J. Cabal, and J. Koˇrenek, "Effective FPGA Architecture for General CRC," in International Conference on Architecture of Computing Systems. Springer, 2019, pp. 211–223.

[5] C. Toal, K. McLaughlin, S. Sezer, and X. Yang, "Design and Implementation of a Field Programmable CRC Circuit Architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 8, pp. 1142–1147, 2009.

[6] The P4 Language Consortium. (Nov. 2018). The P4 Language Specification, Version 1.0.5. [Online]. Available: https://p4.org/p4-spec/p4- 14/v1.0.5/tex/p4.pdf.

[7] M. Grymel and S. B. Furber, "A Novel Programmable Parallel CRC Circuit," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 10, pp. 1898–1902, Oct. 2011.

[8] S. Gueron, "Speeding up CRC32C computations with Intel CRC32 instruction," Information Processing Letters, vol. 112, no. 5, pp. 179– 185, 2012.

[9] K. R Radhakrishnan, M. Amalasweena, T. Kesavamurthy, "FPGA Implementation of Area and Power Optimized and High Speed Seizure Detection System using ELM Classification Method", Neuro Quantology, October 2022, Volume 20, Issue 12.

[10] G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," IEEE Trans. Comput., vol. 52, no. 10, pp. 1312–1319, Oct. 2003.

[11] H. Liu, Z. Qiu, W. Pan, J. Li, L. Zheng, and Y. Gao, "Low Cost and Programmable CRC Implementation based on FPGA," 4 2020. [Online]. Available: https://www.techrxiv.org/articles/Low-Cost and Programmable CRC Implementation based on FPGA/12181494

[12] K. Vipin and S. A. Fahmy, "FPGA Dynamic and Partial Reconfiguration: A Survey of Architectures Methods and Applications," ACM Comput. Surv., vol. 51, no. 4, pp. 1–39, 2018.

[13] P. Orosz, T. Tothfalusi, and P. Varga, "FPGA-Assisted DPI Systems: ´ 100 Gbit/s and Beyond," IEEE Communications. Surveys Tuts., vol. 21, no. 2, pp. 2015–2040, 2019.

[14] M. Jubin and T. Nayak, "Reconfigurable very high throughput low latency VLSI (FPGA) design architecture of CRC 32," Integration, vol. 56, pp. 1–14, 2017.