ISSN: 2366-1313

An Efficient and High-Speed Overlap-Free Karatsuba-Based Finite-Field Multiplier for FGPA Implementation

[1]Dr.D. Vijaya Saradhi, [2]Ms.SK.Gousiya Begum, [3]Motakatla Divya Teja, [4]Yadlapati Aashritha, [5]Katikam Venkat Eswar, [6]Pulugu Gowtham

[1]Professor, Dept. of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

[2]Assistant Professor, Dept. of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

[3]BTech Student, Dept. Of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

[4]BTech Student, Dept. Of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

[5]BTech Student, Dept. Of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

[6]BTech Student, Dept. Of ECE, Malineni Perumallu Educational Society's Group of Institutions, Guntur (A.P)

***Abstract**: Finite field arithmetic is becoming a prominent calculation solution in many applications increasingly. This paper presents the complexity and delay of six defined multiples (the Mastrovito multiplier, the Paar-Roelse multiplier, the Massey-Omura multiplier, the Hasan-Masoleh multiplier, the Berlekamp multiplier, and the Karatsuba multiplier) are compared. Furthermore, this document provides a modified multiplier based on the Karatsuba multiplication algorithm. The product phrases are broken down into chance forms, and all terms are expressed recursively to improve Karatsuba's multiplication rules. This modified architecture saves 14.9% of computation time and consumes 45.5% fewer chips than the current Karatsuba multiplier. The Xilinx ISE design suite simulated and synthesized the proposed architecture for the Spartan & Vertex toolkit. The new structure is simple and clean. The proposed modified Karatsuba multiplier (MKM) is also implemented to calculate the circular convolution of the DSP. In the same Spartan3E FPGA hardware family, 8-bit circular convolution arithmetic using the modified Karatsuba algorithm (MKA) is 26.5% faster than the Karatsuba algorithm (KA). It also consumes 61.7% less chipset than the current KA Convolution-based chipset.*

***Keywords**: Karatsuba Algorithm; Finite fields; FPGA; VLSI; polynomial multiplication; Cicular Convolution*

I. INTRODUCTION

Cryptography is used to provide confidentiality, statistical security, and authentication in many packages, such as communication devices, self-driving cars, the Internet of Things (IoT), and healthcare. There are general encryption strategies: symmetric encryption and public key encryption. Public key cryptography allows all conversation events to communicate robustly without any mysterious statistics previously shared between them. For this reason, it is necessary to introduce the main status quo and default signature for comfortable communication. Examples of public key cryptography[1].

They consist of Diffie-Hellman, RSA, ElGamal, and elliptic curve coding (ECC). ECC has become the most popular public-key cryptosystem among these algorithms, especially because of its short key length concerning security strength and implementation efficiency. Compared with software applications, hardware coding results in better speed and lower price while enjoying digital devices' coffee power and performance requirements. There are some VLSI, and discipline programmable gate set (FPGA), implementations of ECC encryption algorithms.

The computation of elliptic curve factors can be subdivided into finite discipline operations where the finite subject multiplier becomes the main component in implementing ECC devices, where multiplier length and delay dominate. General location and device performance. Hence, many investigations have specifically focused on designing fast and efficient finite-area multipliers for encoder systems[2].

One of the famous multiplication algorithms is the Karatsuba (KA) rule set. This approach aims to reduce the diversity of multiples by replacing them with additions. Whereas in the traditional set of bases (CA) [28], multiplying two n two-digit numbers requires n2 single-digit outputs; this range for KA is nlog32_n1.58. Reducing the region's complexity will reduce the entire site required for the hardware implementation of the multiplier. On the flip side, KA is an iterative set of rules with better time complexity and, as a result, lower speed and overall performance. Therefore, there is a trade-off between the area and delay requirements at the same time that KA is determined via the CA rule set. Karatsuba's method can be changed according to the hardware implementation limit by increasing speed or decreasing its area. Hardware implementation techniques (for example, pipeline) are also used to improve the performance of multipliers.

In polynomial multiplication, the Karatsuba algorithm makes multiplication efficient because this algorithm provides multiplication at the cost of extra addition. Due to the cost of multiplication being higher than the cost of an addition. The sum of the scalar m-bit requires m no. from XOR gates. Koge et al. [3] proposed a set of recursive rules for the rapid multiplication of large integers with a precision of 2k computer sentences, where ok is an integer. Its grammar was derived from the Karatsuba-of-guy grammar and had the same asymptotic complexity. They claimed their algorithm's walk time is slightly better than 1/3 of the repeated calls you make. Murat Senek et al. [4] have given advanced formulas for multiplying polynomials of small degrees on F2 using the Chinese remainder theorem (CRT) that improves the multiplication complexity. Zhang Zhou et al. Green FPGA (Field Programmable Gate Array) and complexity assessment applications for embedded Karatsuba multipliers in bit-parallel are presented in [5].

By introducing co-expression sharing and evaluating the complexity in polynomials of particular time intervals, they achieved a less secure gate than previous ASIC implementations. They extended the analysis using 4-input/6-input lookup tables (LUTs) in FPGAs. They evaluated the complexity of local lookup tables and the advantages and disadvantages of the time zone product in FPGAs with extraordinary computer-aided design (CAD) teams. They claim that their parallel bit multipliers consume the least resources among recognized FPGA implementations.

This paper proposes a modified multiplier based on Karatsuba a multiplication algorithm is proposed. The product phrases are divided into two surrogate documents, and all terms in the recurrence pattern are calculated to improve the set of multiplication rules in Karatsuba. This modified architecture saves 14.9% of computation time and consumes 45.5% fewer slices than the current Karatsuba multiplier. The proposed design was simulated and synthesized using the Xilinx FPGA-based Spartan and Vertex toolkit. The new structure is simple and smooth. It is also implemented to calculate circular torsion. In the same family of Spartan3E FPGA tools, 8-bit circular convolution arithmetic using MKA is 26.5% faster than KA. It also consumes 61.7% fewer chipsets than a KA-based convolution current.

II. RELATED WORK

Xiaoyang Xiao et al. [6] Polynomial multiplication is widely used in verbal exchange, signature, and image processing. In a real-world tool, the device often needs to process a large amount of information successfully and then display the results. However, because polynomials are constantly and widely multivariate, more than software algorithms are needed to meet actual demands. Also, the important thing in polynomial multiplication is the matrix multiplication process. However, due to the increasing number of complex forms of matrix multiplication, it isn't easy to understand green processing and to calculate large matrices. With the improvement in FPGA manufacturing, LSI, memory interface, and EDA equipment, it is possible to use hardware in polynomial multiplication. To address the above problems, we designed and implemented a fully FPGA-based polynomial multiplication platform, combining software and hardware to compensate for modern software algorithms' shortcomings. At the same time, it can run efficiently and effectively—polynomial multiplication function.

Milos Ercegovac et al. [7] A much nicer first-digit function evaluation technique (approach E) allows efficient evaluation of polynomials and certain Boolean features in custom devices. The time required to compute is the order of additions without fetching, where m is the number of digits in the final result. We discuss this technology's parallel-number and serial-number implementation on a DecPeRLe-1 board composed of a Xilinx FPGA. Following the introduction of electronic technology, we present a schematic diagram of the Dec-PeRLe-1 board architecture, present our designs, and discuss their performance.

Ring learning with errors (Ring-LWE) is the basis of many network-based cryptographic systems. The multiplication of the ring polynomials is the most important and computationally comprehensive operation for fully Ring-LWE-based cryptographic systems. This article introduces several optimization techniques to construct an efficient polynomial using number theoretical transformation (NTT). We support a way to improve bit inversion for NTT and NTT inversion. With further improvements, the polynomial multiplier reduces the required clock cycles from (8n + 1.5n lg n) to (2n + 1.5n lg n). By exploiting the connection of stable elements, the polynomial multiplier can reduce the set of constant factors from 4n to 2.5n, which saves about 37.5% of the ROM storage space. In addition, we support a new memory for schematic access to make maximum use of the butterfly operator. Using these techniques, the polynomial multiplier has successfully implemented polynomial multiples

57304/26913 according to 2d of measure 256/512 on a Spartan-6 FPGA.

Hanzaleh Akbari Nodehi et al. [8] There are a few sources on a comfortable MPC, and each has access to private input. Resources want to offload the computation of a polynomial function for inputs to a few processing nodes or employees. The processors are not trustworthy, i.e., a limited group can also conspire to take advantage of the data on the input. The goal is to limit the range of personnel required to compute the polynomial while the complicit persons do not benefit from the registers on the inputs. In this work, we expect the inputs to be large arrays while the employees have limited computation and storage for each factor. As a proxy for this, we expect the correlation between each resource and each worker to support a finite connection load. We recommend a plan for personal record sharing, called nested polynomial sharing, and it's been shown to support basic operations like addition, multiplication, and transposition while respecting the constraints of the problem. Thus, it allows the computation of arbitrary polynomials from input matrices while greatly reducing the number of servers required compared to the traditional scheme. Moreover, it generalizes the currently proposed scheme for sharing polynomials.

Chen et al. [9] The SAT is one of the most important fundamental problems in many computers science and technology areas. SAT solutions are software or hardware for processing the SAT example. This paper develops a specific example of an SAT test solution using FPGA, which applies the DPLL rule to our recent selection of random variables. In addition, we also introduced innovative hardware series from our SAT parser, which is organized along with software types, for example, Xilinx enterprise software, through our C++ parser and some scripts and FPGA hardware. Through experiments, our solution software has great solidity to get the best frequency (200MHz) for the Vertex-7 FPGA board; the top example under test has two hundred variables and 1200 sentences with sources much less than 3% consumed in the FPGA upgrade board.

Sinha et al [10] A homomorphic cipher is a hardware device that allows encrypted data to be computed and thus contains packages in cloud computing that maintain privacy. Although conceptually first-rate, implementing symmetric encryption can be very difficult, software implementations on commonly used computer systems are usually very slow. In this paper, we present our 12-month effort to design a micro-domain architecture on the heterogeneous Arm + FPGA platform to run homogeneous computing on encrypted data. We've designed a custom coprocessor for fancy computing for a symmetric coding system known as Fan-Vercauteren (FV) on an FPGA and turned an Arm processor into a server to run exceptional symmetric applications within the cloud, primarily using this FPGA-based coprocessor. We use the latest mathematical and algorithmic optimization strategies and explore the design space at different levels of the implementation hierarchy. In particular, we are pursuing pipeline strategies at the circuit and phase levels to increase clock frequency and enhance performance, respectively.

III. PROPOSED METHODOLOGY

Karatsuba Multiplier (KM)

In this section, we introduce the fundamental Karatsuba algorithm which can successfully be applied to polynomial multiplication. The Karatsuba Algorithm was introduced by Karatsuba in 1962. The fundamental Karatsuba multiplication for polynomial in *GF(2m)* is a recursive divide-and-conquer technique. It is considered as one of the fastest ways to multiply long numbers. For polynomial multiplication with original Karatsuba method both operands have to be divided into two equal parts. Then each sub operands are divided again into two parts. The process will continue until this become single. Figure1 shows the block diagram of Karatsuba multiplier for degree-3 polynomials. Then we get the followings by splitting the polynomials using KM: If *A(x)* and *B(x)* are field polynomials with degrees 3 over a field GF (2^4)

With the auxiliary variables

$$
\begin{aligned}
&D_0 = a_0b_0 \quad , D_1 = a_1b_1\\
&D_2 = a_2b_2 \quad , D_3 = a_3b_3\\
&D_{0,\,1} = (a_0 + a_1)\,(b_0 + b_1)\\
&D_{0,\,2} = (a_0 + a_2)\,(b_0 + b_2)\\
&D_{1,\,3} = (a_1 + a_3)\,(b_1 + b_3)\\
&D_{3,\,2} = (a_3 + a_2)\,(b_3 + b_2)\\
&D_{0,1,2,3} = (a_0 + a_1 + a_2 + a_3)\,(b_0 + b_1 + b_2 + b_3)
\end{aligned}
$$

Field multiplication can be performed into two steps. Firstly, we perform an ordinary polynomial multiplication of two field elements. Secondly, a reduction operation with an irreducible polynomial is need to be performed in order to obtain the (m-1) degree polynomial. It is noticed that once the irreducible polynomial *p(x) = x4+ x+1* has been selected, the reduction step can be accomplished by using XOR gates only. From the irreducible polynomial *p(x)* we can replace *x 4= x+1, x5= x2+ x* and *x6 = x 3+ x2* to obtain *C' (x)* as follows

$$
\begin{aligned}
&C'(x) = A(x)\,B(x) \bmod p(x)\\
&C'(x)=(D_{0,1,2,3}-D_{1,3}-D_{2,0}-D_{3,2}-D_{0,1}+D_0+D_1+D_2)x^3+\ (D_{0,2}+D\\
&+D_1-D_0)\,x^2+(D_{0,1}+D_{1,3}+D_{3,2}-D_0)x+(D_{1,3}-D_1-D_3+D_2+D_0)\ (
\end{aligned}
$$

Modified Karatsuba Multiplier (MKM)

In this section our Modified Karatsuba Algorithm (MKA) has been discussed. In MKA all techniques are same as fundamental basic Karatsuba multiplier except the splitting techniques. To optimize the Karatsuba Multiplication Algorithm, the product terms are splited into two alternative forms. This reduction technique requires small area and less delay than others existing multiplication algorithms. The results are compared by using Xilinx based synthesis tools on different FPGA device family like Spartan & Vertex. Our synthesis results are better than existing basic Karatsuba algorithm which is shown in the following section. Assume

A(x) and B(x) are two field polynomials with degree 3 in GF(24

$$A(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$
$$B(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

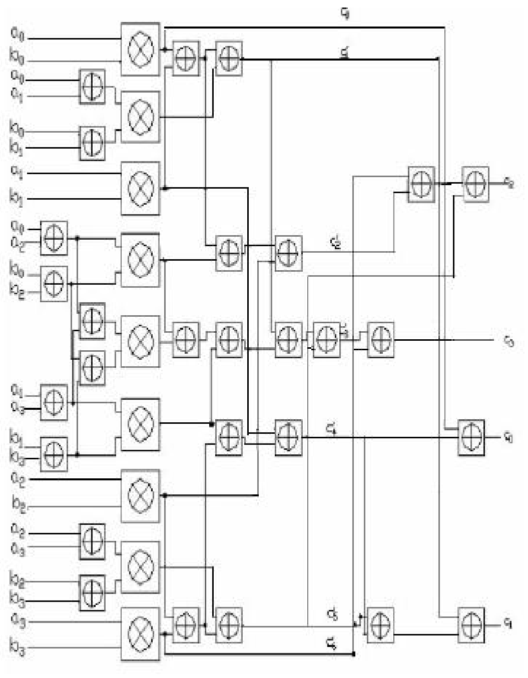

Fig.1 Block diagram of Karatsuba multiplier for degree-3 polynomials

Then we get the following expression by splitting the coefficients of C(x)= A(x)B(x) polynomial using MKA

$$D_0 = a_0 b_0 \ , D_1 = a_1 b_1$$
$$D_2 = a_2 b_2 \ , D_3 = a_3 b_3$$
$$D_{3,2} = (a_3 + a_2)(b_3 + b_2)$$
$$D_{3,1} = (a_3 + a_1)(b_3 + b_1)$$
$$D_{3,0} = (a_3 + a_0)(b_3 + b_0)$$
$$D_{1,2} = (a_1 + a_2)(b_1 + b_2)$$
$$D_{0,2} = (a_0 + a_2)(b_0 + b_2)$$
$$D_{0,1} = (a_0 + a_1)(b_0 + b_1)$$

Here operands are splited into two alternative terms. Employing auxiliary variables, we can obtain the following expression.

$C(x) = D_3x^6 + (D_{3,2} - D_2 - D_3)x^5 + (D_{1,3} - D_1 - D_3 + D_2)x$

$D_{1,2} - D_1 - D_2)x^3 + D_{0,2} - D_2 - D_0 + D_1)x^2 + (D_{0,1} - D_1 -$

Then *C'(x)* is computed by using the relationsl mod *p(x)*. Using the irreducible polynomial terms x^4, x^5 and x^6 in *C(x)* are replaced by (x^3+x^2) respectively. The simplified expression follows:

$C'(x)=(D_{0,3}-D_0+D_{1,2}-D_1-D_2)x^3+(D_{0,2}+D_{3,2}+D_1-D_0)$
$D_{3,2}-D_0)x +(D_{1,3}-D_1-D_3+D_2+D_0)$

Figure2 shows the block diagram of Modified Karatsuba multiplier for degree-3 polynomials

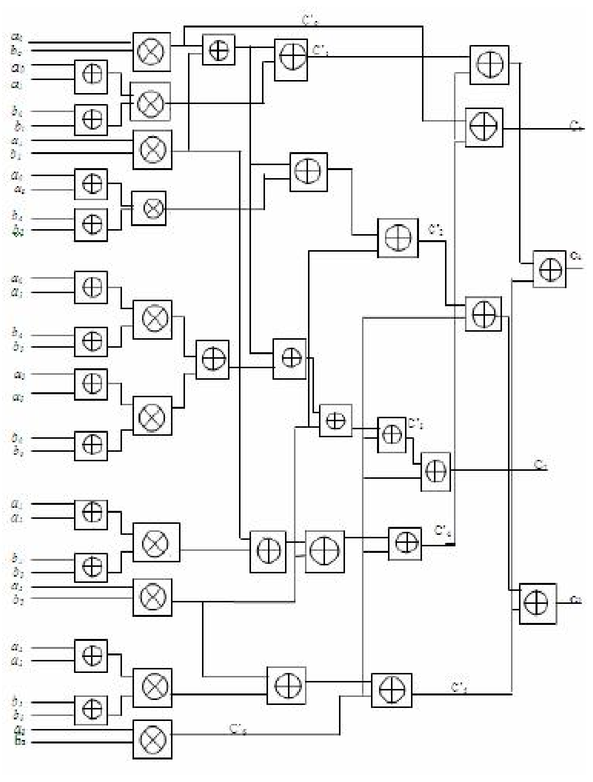

Fig.2 Block diagram of Modified Karatsuba multiplier for degree-3 polynomial

IV. RESULTS & DISCUSSION

We have studied the performance of each multiplier over GF(*24*) employing the Xilinx ISE simulation tool. Multipliers are implemented on Spartan3E xc3s100e-4 device. These multipliers are compared based on number of slices, number of 4-input LUTs, bonded I/O blocks and delay

TABLE 1: Comparison of different multipliers in *GF(24)* field

Different GF Multipliers	No. of slices (out of 960)	No. of 4 i/p LUTs (out of 1920)	No. of bonded IOBs (out of 66)	Max. combinational path delay (ns)
Mastrovito[2]	7	12	12	13.195
Paar – Roelse [3]	7	12	12	13.083
Massy Omura [4]	7	13	12	14.932
Hasan Masoleh [5]	7	12	12	13.271
Berlekamp [6]	8	15	12	12.985
Karatsuba Multiplier (KM) [7]	9	15	12	14.790
Modified Karatsuba Multiplier (MKM)	6	11	12	13.057

Table-1 shows the result of device utilization and combinational path delay of various types of GF(*24*) multipliers. Proposed multiplier has less hardware complexity than another GF multiplier. It is also faster than other multipliers except Berlekamp

Table.2 Comparison of device utilization and combinational path delay to compute circular convolution using KA and MKA

Length	Algorithm	# Slices (out of 960)	# 4-i/p LUT (out of 1920)	# Bonded IOB (out of 66)
	circular convolution using KA	10	17	12
4-bit	circular convolution using MKA	7	12	12
	circular convolution using KA	68	118	24
8-bit	circular convolution using MKA	26	45	25

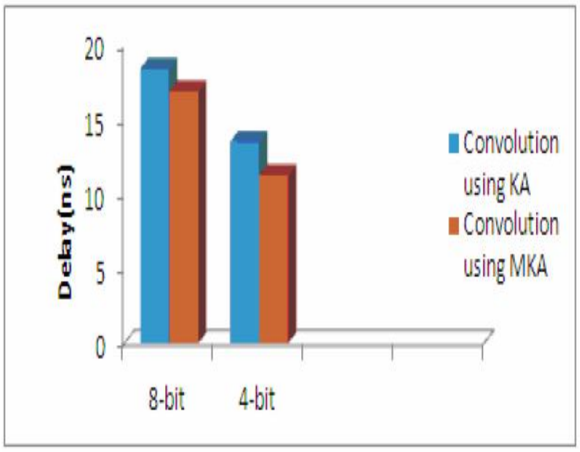

Fig.3 Delay for comparing circular convolution using KA and MKA

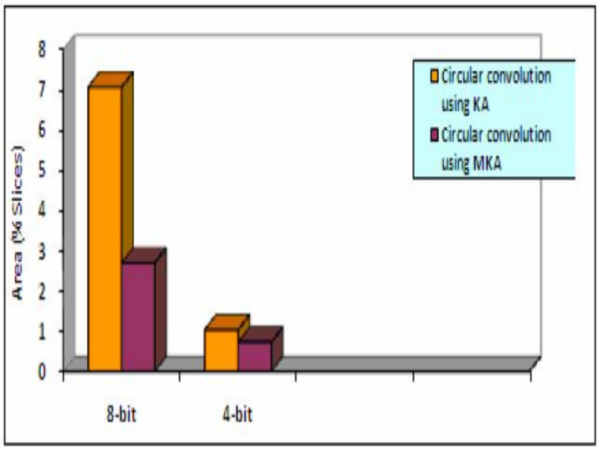

Fig.4 Area occupied (% slices) between circular Convolution using KA and MKA

The circular convolution algorithm is coded using Verilog HDL language. It is simulated and synthesized using Xilinx ISE 7.1i software tool. Table 5 shows the comparison of device utilization and combinational path delay to compute circular convolution using KA and MKA. It is observed that circular convolution based on MKA requires least amount of area and path delay. Figure 3 shows the delay in computing convolution using two different algorithms and Figure 4 shows the resource utilization in terms of % of slices necessary for the implementation. In Spartan3E FPGA device family, computation of 8-bit circular convolution based on MKA is 26.5% faster than KA. It also consumes 61.7% less slices than existing KA based convolution.

V. CONCLUSION

In this paper, modified Karatsuba multipliers for degree 3 and 7 polynomials has been implemented on FPGA platform. The device utilization and combinational path delay of MKM have been compared with standard 8×8 KM. It has been observed that the proposed multiplier has better timing performance than standard KM. In Spartan3E FPGA device, proposed multiplier needs 14.9% lesser delay than KM, and it also consumes 45.5% lesser slices compared to KM. The new architecture is very simple and easy. This feature is advantageous to have a suitable trade-off between area and speed for implementing circular convolution algorithm in VLSI. In FPGA device family, computation of 8- bit circular convolution using MKA is 26.5% faster than KA. It also consumes 61.7% less slices than existing KA based convolution. MKM may also be used to design cryptosystems. Proposed multiplier is faster and hardware efficient compared to existing Karatsuba multiplier

REFERENCES

[1] C Dharma Raj, D Vijaya Saradhi, P Hemambaradhara Rao, P Chandra Sekhar, 2011, “Optimization of Layer Thickness to Yield Predetermined Shielding Performance of Multilayer Conductor Electromagnetic Shield”, Proc. of the International Conference on Advanced Computing and Communication Technologies, pp.352-356.

[2] C D Raj, P H Rao, D V Saradhi, 2011, “Development of a Multilayer Conductor Electromagnetic Shield for Optimum Shielding Performance”, International Conference and Workshop on Emerging Trends in Technology, pp.1153-1156.

[3] Vijaya Saradhi Dommeti and Dharma Raj Cheruku, 2020, “Multi-layer composites shielding for electromagnetic radiation protection”, International Journal of Advanced Intelligence Paradigms, pp.139-158.

[4] Vijaya Saradhi, D., Katragadda, S. and Valiveti, H.B. (2023), "Hybrid filter detection network model for secondary user transmission in cognitive radio networks", International Journal of Intelligent Unmanned Systems, Vol. 11 No. 1, pp. 61-74. https://doi.org/10.1108/IJIUS-08-2021-0091.

[5] Vijaya Saradhi Dommetti and Dharma Raj Cheruku, 2017, “Review and Analysis of Contemporary Polymer Coats based Electromagnetic Shielding Strategies”, Indian Journal of Science and Technology, Vol 10(18), DOI: 10.17485/ijst/2017/v10i18/111939.

[6] H. A. Nodehi, S. R. H. Najarkolaei and M. A. Maddah-Ali, "Entangled Polynomial Coding in Limited-Sharing Multi-Party Computation", 2018 IEEE Information Theory Workshop (ITW), pp. 1-5, 2018.

[7] Z. Chen, J. Wu, H. Guo, J. Xiong and A. He, "A FPGA based SAT solver with random variable selection", 2016 International Conference on Integrated Circuits and Microsystems (ICICM), pp. 329-333, 2016.

[8] S. Sinha Roy, F. Turan, K. Jarvinen, F. Vercauteren and I. Verbauwhede, "FPGA-Based High-Performance Parallel Architecture for Homomorphic Computing on Encrypted Data", 2019 IEEE

International Symposium on High Performance Computer Architecture (HPCA), pp. 387-398, 2019.

[9] Z. J. Shi and H. Yun, " Software implementations of elliptic curve cryptography," *International Journal of Network Security*, vol. 7, no. 1, pp. 141-150, 2008.

[10] Murat Cenk and Ferruh O¨ zbudak,"Improved Polynomial Multiplication Formulas over F2 Using Chinese Remainder Theorem", *IEEE Transactions on Computers,* vol. 58, no. 4, pp. 572- 576, April 2009