International Journal of Scientific Research in Science and Technology
Available online at : www.ijsrst.com
Print ISSN: 2395-6011 | Online ISSN: 2395-602X
doi : https://doi.org/10.32628/IJSRST

A Novel Binary Polynomial Multiplier Based on M-Term Karatsuba for Finite Field Computation

Konduri Prasanna Lakshmi[1], Dr. S. Leela Lakshmi[2]

[1]PG Scholar, Department of ECE, VEMU Institute of Technology, P.Kothakota, Andhra Pradesh, India.

[2]Professor, Department of ECE, VEMU Institute of Technology, P.Kothakota, Andhra Pradesh, India.

ARTICLEINFO

Article History:

Accepted: 01 Sep 2023
Published: 10 Sep 2023

Publication Issue
Volume 10, Issue 5
September-October-2023
Page Number
91-98

ABSTRACT

This paper presents an optimized M-term Karatsuba-like binary, polynomial multiplier for finite field arithmetic. The proposed method is based on the traditional Karatsuba algorithm but incorporates modifications to handle binary polynomials of arbitrary degree. The proposed algorithm reduces the number of multiplications and additions required for multiplication of binary polynomials by using a divide-and-conquer approach. The algorithm also minimizes the number of temporary storage registers required during the multiplication process, thereby reducing the overall complexity of the algorithm. Experimental results show that the proposed algorithm outperforms existing algorithms in terms of speed and hardware complexity for polynomial multiplication in finite fields. The proposed algorithm is suitable for hardware implementation in applications such as error-correcting codes, cryptographic systems and digital signal processing.

Keywords : Karatsuba algorithm, Finite field arithmetic, Polynomial multiplier.

I. INTRODUCTION

With the ever-growing expansion of modern information technologies in almost every field, the number of threats and importance of information security are increasing day by day. Cryptography systems play a crucial role in ensuring safety and security of information .In these systems, a fundamental and frequently used operation that determines the overall speed and cost of systems is finite field multiplication. Therefore, the efficiency of the multiplier is of paramount importance . Among various polynomial multiplication algorithms, school-book multiplication (SBM) is the simplest form of multiplication. For two polynomials of n–1 degree, the SBM has complexity of O(n2). In order to improve the efficiency of multiplication, several algorithms have been proposed by researchers . One widely known algorithm is the Karatsuba–Ofman multiplier (KOM) . It is a recursive multiplicative approach that

has a lower space complexity [O(nlog2 3)] compared to conventional SBM . This approach splits the operands into lower and upper parts, and uses three sub multipliers to compute the product. Although KOM can reduce resource requirements, it has the disadvantage of higher delay compared to SBM because of its sub multiplier-based recursive structure. To overcome this problem, a series of Karatsuba modifications and various implementation strategies were proposed. Overlap-free Karatsuba was proposed to eliminate the higher combinational delay of general Karatsuba. A low-complexity Karatsuba multiplier introduced a new implementation strategy to eliminate the high register complexity in current systolic implementation, which leads to an increase in area and power consumption. Samanta et al. presented a modified Karatsuba implementation for 8-bit operands, in which terms are separated into different formats to reduce operational latency. Li et al. proposed a new type of non-recursive Mastrovito multiplier for GF(2m) using an n-term Karatsuba algorithm (KA). In Chiou-Yng et al.'s work, they presented an efficient digit-level parallel-in– serial-out (PISO) multiplier with sub quadratic space complexity using the overlap-free Karatsuba multiplication algorithm. Some of the relevant studies are and the M-term Karatsuba-like approach has received much attention in recent years. In these Karatsuba-like algorithms, the operand can be divided into a higher number of terms compared to only two terms in Karatsuba. This allows reducing the number of recurrent operations in the KOA and, hence, increasing the speed of multiplication. In the M-term Karatsuba-like multiplier, the more the number of terms, the larger the number of sub multipliers is needed. Montgomery introduced five-, six-, and seven-term Karatsuba-like algorithms that split each polynomial into five, six, and seven parts, and uses recursive construction to perform the multiplication. Based on Montgomery's work, Fan et al. proposed a method to obtain more Karatsuba-like formulas by generalizing the division algorithm. Find and Peralta gave a detailed account of M-term Karatsuba-like, where M =4, 5, 6, and 7, and their theoretical representation. Compared to previous works, they achieved smaller size and depth by optimizing the existing M-term Karatsuba-like algorithm. It has been theoretically proven that M-term Karatsuba multipliers, where M = 3, 4, 5, 6, and 7, have better performance for large degree polynomials since it reduces the number of recurrence stages and can be recommended as a viable alternative to the regular KOM . However, these Karatsuba-like formulas contain combinations of the sub multipliers, and the total delay and area are not linear, especially when physically implemented on the hardware. To optimize the combinational delay of these Karatsuba-like multipliers, a new methodology is proposed and verified with field-programmable gate array (FPGA) implementation. The main contributions of this article are given as follows.

1) We investigated the gate-level space and time complexity analyses for a larger number of terms, such as five-, six-, and seven-term Karatsuba-like algorithms.

2) We designed a road map to achieve an efficient finite field multiplier using the M-term Karatsuba-like algorithm.

3) We experimentally evaluated the proposed hardware on FPGA where the result indicated the average of 26% delay reduction and 15% lower area–delay product compared to standard KOM.

In this article, first, the performance of M-term Karatsuba like polynomial multipliers is evaluated both theoretically and based on implementation results. Furthermore, the hardware design space is explored for the various operand sizes in terms of area [number of look up tables (LUTs)], delay, and ADP. We have six degrees of freedom to select the most efficient M-term Karatsuba-like multiplier for each operand size in this step. Second, an improved composite model based on M-term Karatsuba-like and SBM is proposed. This composite solution uses the M-term Karatsuba-like at the higher level of recurrence

and utilizes single-step conventional SBM in a lower recurrence stage. The design has three degrees of freedom for selecting the most efficient composite model. Overall, there are 18 degrees of freedom in the proposed road map to select the number of terms in M-term Karatsuba-like and combination levels. The proposed design was first validated against Python results, described in VHDL, synthesized, and implemented on various FPGA devices, and furthermore, performance metrics and implementation costs were determined. The proposed multiplier demonstrated roughly 26% reduction in combinational delay and collectively has 15% lower ADP compared to the standard KOM method. Comparison with state of the art also indicated the effectiveness of the design. The rest of this article is organized as follows. The generalization of M-term Karatsuba-like algorithms is presented in Section II. The implementation of the M-term Karatsuba like model on FPGAs has been introduced in Section III. Discussions and significance of the proposed method are given in Section IV. Finally, Section V concludes this article.

II. EARLIER WORK

In the finite field arithmetic, multiplication requires a binary polynomial multiplier followed by the modular reduction operation with an irreducible polynomial. The multiplicative cost and additive cost determine the space complexity of implementing this multiplier. The number of combinational gates required to implement the multiplier is used to assess space complexity, while delay complexity is determined by the linear addition of standard gate delays. In this section, the space and delay complexities of the SBM and M-term Karatsuba-like are assessed.

SBM Algorithm Considering A and B as two degree one binomial, the polynomial multiplier using SBM could be realized as A(x) = A1x + A0 ,B(x) = B1x +B0 where A and B is split into two parts of (A0, A1) and (B0, B1).

A(x).B(x)= A1.B1x2 +(A1.B0 + A0.B1)x + A0.B0.

The polynomial multiplication of two degrees one binomial could be calculated, as described in trinomial, by employing four point multiplications and three point additions. The theoretical complexities for hardware implementation of this multiplier are given by the number of AND and XOR gates utilized. The maximum delay is the total number of gates in the critical path of the multiplier. The theoretical complexities are given as follows:

SXOR = (n–1)2

SAND = (n)2 ,STotal = SXOR +SAND ,TTotal = TAND +TXOR.SXOR and SAND gives the space complexity in terms of number of XOR and AND gates for operand size of n. STotal provides overall space complexity of the SBM multiplier. TXOR and TAND denote the linear addition of the combinational delay of XOR and AND gates. Total represents the total delay in the critical path of the multiplier.

B. M-term Karatsuba-Like Multipliers An M-term Karatsuba-like multiplier breaks the operands into smaller size operands and uses a number of sub multipliers to recursively calculate the product. M-term Karatsuba uses a similar concept as KOM but splits to a higher number of equal parts. For the rest of this article, we assume that each operand is split into M number of polynomials with equal length. Table I presents recursive products and reconstruction steps required for M-term Karatsuba-like multipliers from M =2 to M =7 for different operand sizes (n). Here, for simplicity, we chose n and M to be equal. Each operand polynomials are first split into M equal parts denoted as A0, A1,...,AM and B0, B1,...,BM. Partial products are represented as P0, P1,...,Pk, where “k” is the number of partial products. The recursive products and reconstruction steps required for each M-term are tabulated in this table. In each recursion, the partial products are calculated using sub multipliers followed by reconstruction step. The coefficients of the multiplication result polynomial

are presented as R0, R1,...,RC, where C =2M –2. Using the equations in Table I, the theoretical boundaries for XOR and AND gates space and time complexities were determined and presented where the following holds.

1) SXOR: Number of XOR gates required/additive cost.

2) SAND: Number of AND gates required/multiplicative cost.

3) TTotal: Combinational delay required to multiply the given operands.

4) TG: Assume that both the combinational delay of AND and XOR gates are the same.

5) Rec: Number of recurrent stage required. The total space complexity of the multiplier could be calculated using the following equation:

STotal = SXOR +SAND.

Rec denotes the number of recurrence stages required. The main advantage of employing M-term Karatsuba-like functions is that it reduces the number of recurrence stages compared to KOM.

C. Discussion of Theoretical Complexities The time and space complexities for M-term Karatsuba-like multipliers dependon the number of stages and sub multipliers.

It is obvious that, for each operand size of "n," the number of recurrence stages required for larger values of "M" is smaller than the number of recurrent stages necessary for a two-term Karatsuba-like multiplier. Since each stage adds an extra delay to the total delay of the multiplier, it is expected that the delay of the two-term Karatsuba-like would be larger than multipliers with larger M. On the other hand, the size of the multiplier is smaller in two-term as it only has three sub multipliers. For example, in the seven-term multiplier, 22 sub multipliers have been employed, and as a result, it has a larger area than that of the two-term. The total number of gates (XOR and AND) and their corresponding delay are considered for theoretical evaluation of complexity. It is evident that the performance of each M-term Karatsuba-like varies depends on the size of the operands. However, the total number of gate utilization significantly increases when the number of recurrence iterations drops. Every term M-term Karatsuba-like multipliers were accessed for different operand sizes, and the performance metrics, such as space and time complexities, were examined with a detailed comparison of theoretical data and FPGA implementation results.

III. PROPOSED WORK

This work takes advantage of the low time complexity in SBM and low-space complexity in M-term Karatsuba by combining these two methods. Furthermore, the area–delay tradeoff was investigated.

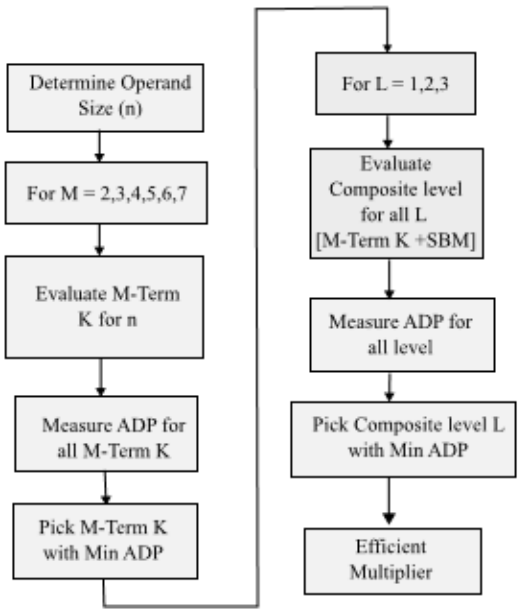

Fig.1: Proposed methodology using M-term Karatsuba

A. Efficient M-term Karatsuba-Like Functions First, hardware description language (HDL) codes were developed for each multiplier according to the recursive products and reconstruction functions represented in Table I. A set of python frameworks for each split function was created where the frameworks automate the generation of the VHDL codes for operand size n for all the M-term Karatsuba-like

functions. Thereafter, HDL codes were synthesized on Xilinx Artix 7 FPGA (XC7A35TIC)device.To measure

the delay,the input/output(i/o) buffer was neglected during the experimental study. This FPGA family contains six input LUTs as building blocks. The data reported in this section are only for implementation on Xilinx Artix 7 FPGA. It is worth mentioning that these results could change according to FPGA device or synthesis tools since each tool has a set of setting, which can optimize results for the area, speed, power, and other criteria. Besides, different vendors have various algorithms for optimization with different boundary conditions. The total number of LUTs, combinational delay, and ADP for hardware implementation of M-term Karatsuba like multiplier for different operand sizes are presented in Figs. 3–5(a)–(f). It needs to be acknowledged that the implementation results presented in this section do not include the delay and area for modular reduction operation. The reduction module adds a constant value of delay and later would be added to construct a finite field multiplier. The two-term function has been employed at all the last recurrent stages to reduce the zero paddings on some split functions. For example, if the operand size is 2, and if it needs to be reduced by a seven-term Karatsuba-like , there is a need to pad five zeros with the given operand, and hence, it increases the computing complexities.

1) Area Complexity Comparison: Trend variations of area complexity, such as LUTs and theoretical gates to implement the all M-term Karatsuba-like multipliers, are illustrated. LUT complexities are recorded from the circuit implemented on the Xilinx Artix 7 FPGA, whereas the gate complexities were evaluated using it is visible that the number of gates and LUT utilizations increase when the operand size increases. Our implementation results confirm that the area complexity mainly depends on the number of recursive products and recurrent stages (Rec) used to break down the given operands. It is worth noting that the area required to compute the two term Karatsuba-like multiplier is smaller than the seven-term Karatsuba-like for the given operand size. This is because the two-term uses only three recursive products, whereas the seven-term uses 22 recursive products. The area complexity of other M-term Karatsuba-like lies between two-term and seven-term, and it mainly depends on the number of recurrent stages for the operand size. From Table III, it is evident that there is a hike in LUT utilization whenever there is an increase in the number of recurrent stages (Rec). For example, for M =2, the difference in LUT consumption is not huge between n = 232 and 282 since it requires eight recurrent stages to compute the product. However, when the operand size increased to 409, there is a 185% increase in LUT since it requires nine recurrent steps. As the operand size grows, the same trend was experienced up to the operand size n =750.

2) Time Complexity Comparison: In Fig. 3, the trend for time complexity, such as combinational delay and the theoretical gate path for all M-term Karatsuba-like multipliers, is plotted. Delay complexities are also recorded on the Xilinx Artix 7 FPGA, whereas the gate path complexities were analyzed using. The graph shows that the FPGA data recorded from the implementation match the theoretical evaluation. As previously mentioned in the area complexity analysis, the number of recurrent stages (Rec) still plays a significant

role in determining the curvature of the combinational delay. It is obvious that the number of the recurrent stages (Rec) and the critical path delay of each stage are considered important factors that determine the speed of the multiplication. The two-term has a higher number of Rec, whereas each Rec stage has a lower critical path. Though seven-term has lower Rec stages for N = 232, 282, and 409, the FPGA delays are not lesser than three-term because of the higher critical path delay of each Rec stage. The two-term function has less area utilization; on the other hand; it requires more delay because of the extensive recurrent stage (Rec). For n = 232 multiplication, two-term requires 14.4ns,and seven-termrequiresonly9.7ns, hence seven-term, which is 67% faster for given operand size. To understand the

performance behavior of the M-term Karatsuba-like for higher operand sizes, area and delay complexity analyses were extended up to n = 2500. Then, LUTs and combinational delays were recorded, and the ADP was computed to pick the efficient M-term Karatsuba-like functions for each operand size.

B. Optimization Using Proposed Method After implementing the M-term Karatsuba-like for n = 232 to 2500, the efficient M-term multiplier is chosen for each operand size (n). The ADPs, along with the efficient M-term Karatsuba-like corresponding to the operand sizes, were tabulated in Table V. The main advantage of Karatsuba over SBM is having a lower area complexity. The tradeoff, however, is an increase in time complexity. The delay of M-term Karatsuba-like multipliers required is competitive to SBM, yet it does not suffer from the very high space complexity of the SBM. It is evident that the M-term Karatsuba-like multipliers have better performance in the higher bound of the recurrent stage (Rec), and SBM is more efficient in lower bounds. To make the multiplier more efficient, the composite model was presented by investigating the tradeoff between the area and time complexities. Phase B introduces the composite method with different levels by combining SBM with M-term Karatsuba-like multipliers. Combined designs were developed by using the M-term function on top levels and SBM on bottom levels. Fig. 4 demonstrates the different levels of composite level explorations for an operand size n. In level 1, the M-term Karatsuba function is implemented in the first recurrent stage for operand size n, and the single-step SBM is implemented on the next recurrent step (n/M). In level 2, two M-term functions have been employed on the top-two recurrent stages continued by SBM with a size of (n/M2). In level 3, SBM is implemented in the fourth recurrent stage with a size of (n/M3). All of these levels were tested on Xilinx Spartan 7, and performance metrics were recorded. Table VI summarizes the results for all three composite levels based on the ADP. As the data in Table VI show, after finding the efficient M-term Karatsuba-like, our proposed method was able to find the more efficient multiplier, which outperforms the current findings. It was confirmed with the experimental evaluation, and the highlighted data show the efficient composite levels for each operand size. two iterative steps of MTK. Area complexity trends are vice versa. When the delays decrease, area consumption becomes slightly larger since we use larger SBM in level I and smaller SBM in different levels. However, here, the proposed method still outperforms the efficient split method in terms of ADP. Fig. 5 demonstrates the hardware description blocks from the top level to the inner logic level. For the illustration of the proposed method, four-term composite level 3 was used with the operand size n. The top-level four-term Karatsuba-like has two input operand A and B of bit length n and has an output of M with[2n–2]bit length. The next two levels of four-term Karatsuba-like have n/4 and n/16 correspondingly. The SBM has been used at a lower level with the operand size n/64. In Fig. 6, the gate-level design of four-term Karatsuba-like is depicted for operand A and B with size n. The design uses nine recursive products and reconstruction, to generate the product M with bit length [2n–1]. Depiction of the data-flow graph (DFG) of gate-level design for hardware implementation of n-bit SBM. The input operands are A and B with the size of n and the output of M with the operand size of 2n–1. For the smaller operand sizes, such as n = 232 to 409, composite level I is more efficient than other levels. In contrast, levels II and III give better performance for operand sizes n =750 to 2500.

IV. EXPERIMENTAL RESULTS

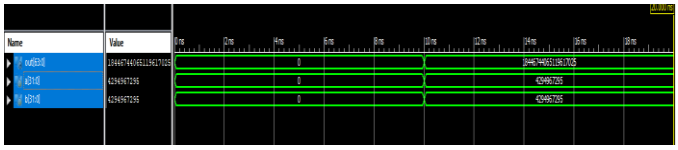

Fig.2: Simulation result

The above figure shows the simulated output of project.

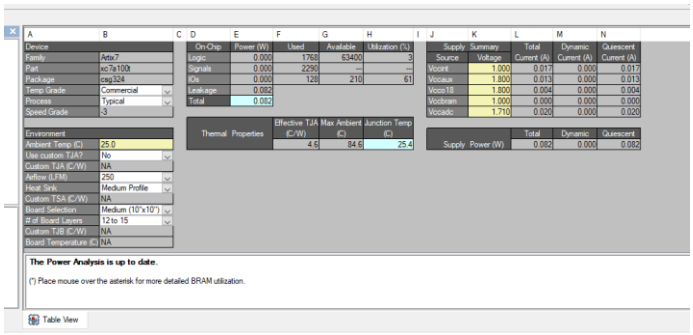

Fig.3:Power Analysis

The above figure represents the power analysis of our project.

```
Slice Logic Utilization:
 Number of Slice LUTs:                 2228  out of  63400     3%
    Number used as Logic:              2228  out of  63400     3%
```

Fig.4:Area

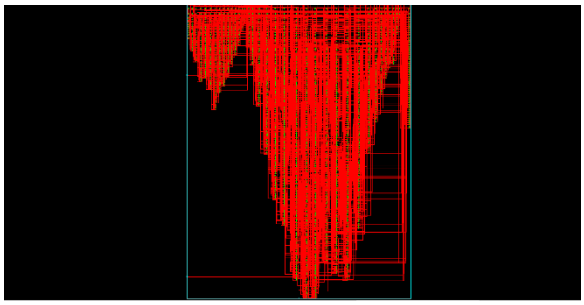

Fig.4:Technological Schematic

The above is the technological schematic view of our project.

V. CONCLUSION

In this article, first M-term Karatsuba-like binary multipliers were analysed in terms of space and time complexities for different values of M and various operand sizes (n). Performance parameter's trends were pictured for the Xilinx Artix FPGA device. Later, a novel composite method is introduced to take advantage of the low-space complexity of M-term Karatsuba-like and low time complexity SBM. The proposed method was extensively tested on different FPGAs to attain the improvement graph over other similar works. The proposed work offers power efficient with 55.87mW. In FPGA devices, implementation results show that the composite method requires 11% additional resources and drops the delay complexity 26% lower, and it is 15% more efficient in ADP than standard KOM. This work achieved the suitable trade-off between space and time complexities, which minimizes the ADP requirement of the multiplier.

VI. REFERENCES

[1]. R. Abu-Salma, M. A. Sasse, J. Bonneau, A. Danilova, A. Naiakshina, and M. Smith, "Obstacles to the adoption of secure communication tools," in Proc. IEEE Symp. Secur. Privacy (SP), May 2017, pp. 137–153.

[2]. B. Vembu, A. Navale, and S. Sadhasivan, "Creating secure communication channels between processing elements," U.S. Patent 9 589 159, Mar. 7, 2017.

[3]. J. Yoo and J. H. Yi, "Code-based authentication scheme for light weight integrity checking of smart vehicles," IEEE Access, vol. 6, pp. 46731–46741, 2018.

[4]. P. Aparna and P. V. V. Kishore, "Biometric-based efficient medical image watermarking in E-healthcare application," IET Image Process., vol. 13, no. 3, pp. 421–428, 2019.

[5]. T. D. Premila Bai, K. M. Raj, and S. A. Rabara, "Elliptic curve cryptography based security framework for Internet of Things (IoT) enabled smart card," in Proc. World Congr. Comput. Commun. Technol. (WCCCT), Feb. 2017, pp. 43–46.

[6]. Z. U. A. Khan and M. Benaissa, "High-speed and low-latency ECC processor implementation over GF(2m) on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 1, pp. 165–176, Jan. 2017.

[7]. G. Chen, G. Bai, and H. Chen, "A high-performance elliptic curve cryptographic processor for general curves over GF(p) based on a systolic arithmetic unit," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 54, no. 5, pp. 412–416, May 2007.

[8]. H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, and T. Stouraitis, "A high-speed

FPGA implementation of an RSD-based ECC processor," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 1, pp. 151–164, Jan. 2016.

[9]. K. C. C. Loi and S. B. Ko, "Scalable elliptic curve cryptosystem FPGA processor for NIST prime curves," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 11, pp. 2753–2756, Jan. 2015.

[10]. N. Y. Goshwe, "Data encryption and decryption using RSA algorithm in a network environment," Int. J. Comput. Sci. Netw. Secur. (IJCSNS), vol. 13, no. 7, p. 9, 2013.

[11]. R. J. McEliece, Finite Fields for Computer Scientists and Engineers, vol. 23. Boston, MA, USA: Springer, 2012.

[12]. N. Homma, K. Saito, and T. Aoki, "Toward formal design of practical cryptographic hardware based on Galois field arithmetic," IEEE Trans. Comput., vol. 63, no. 10, pp. 2604–2613, Oct. 2014.

[13]. A. D. Piccoli, A. Visconti, and O. G. Rizzo, "Polynomial multiplication over binary finite fields: New upper bounds," J. Cryptograph. Eng., vol. 10, pp. 197–210, Apr. 2019.

[14]. F. Mallouli, A. Hellal, N. Sharief Saeed, and F. Abdulraheem Alzahrani, "A survey on cryptography: Comparative study between RSA vs ECC algorithms, and RSA vs el-gamal algorithms," in Proc. 6th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/5th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom), Jun. 2019, pp. 173–176.

[15]. B. Sunar, "A generalized method for constructing subquadratic complexity GF (2k) multipliers," IEEE Trans. Comput., vol. 53, no. 9, pp. 1097–1105, Sep. 2004.

Cite this article as :

Konduri Prasanna Lakshmi, Dr. S. Leela Lakshmi, "A Novel Binary Polynomial Multiplier Based on M-Term Karatsuba for Finite Field Computation", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 10 Issue 5, pp. 91-98, September-October 2023.

Journal URL : https://ijsrst.com/IJSRST52310511