A Sub-Quadratic Multiplier for ECC Processor in GF (2^{283})

Sumit Singh Dhanda[1, a)], Brahmjit Singh[1, b)] and Poonam Jindal[1, c)]
[1]*Department of Electronics and Communication Engineering,*
National Institute of technology, Kurukshetra, Haryana, India-136119

Author Emails
[a)] dhandasumit@gmail.com
[b)]brahmjit.s@gmail.com
[c)]poonamjindal81@nitkkr.ac.in

Abstract.
Scalar multiplication decides the performance of elliptic curve processors (ECP). Its efficiency is dependent upon the size and speed of finite-field multipliers. Sub-quadratic multipliers can be used to achieve higher performance of scalar multiplication ECP. In this work, we have proposed a hybrid Karatsuba multiplier for GF(2^{283}) which is coded in Verilog and synthesized on Xilinx PlanAhead software. It is implemented on different Xilinx FPGAs for the detailed analysis. Virtex-6 FPGA is found to be the best for the implementing this design. Implementation has achieved a delay of 8.764 ns and slice consumption of 7532 for this design. On comparison, with existing bit-parallel multipliers, it is verified that the space complexity of the multiplier is less. The design exhibits the minimum delay of 8.041 ns on Virtex-7.

Keywords: Hybrid-Karatsuba Multiplier; Elliptic Curve Cryptography (ECC); Field-Programmable Gate Array (FPGA); Sub-quadratic multiplier; Galois field.
DOINumber: 10.14704/NQ.2022.20.12.NQ77260 **NeuroQuantology2022;20(12): 2674-2681**

INTRODUCTION

In Internet of Things (IoT), massive data is conveyed by numerous of devices which are placed in the open. This information is utilized to take important decisions related to various business aspects. Hence the data is utmost valuable. It has made securing the information a top priority. Varied number of IoT use cases has provided new ways to conduct business. After transmission, it is stored, sorted and processed for extraction of knowledge inside Cloud, which has its own security requirements. It includes authentication, access-control etc. These can be easily provided by an asymmetric cipher. Elliptic curve cryptography (ECC) is one such primitive that can be used to provide more than one security services. It also serves as the base for the blockchain technology such as ethereum [1].

ECC is the most popular as it provides the security per bit is higher than other asymmetric ciphers. Elliptic curve discrete logarithm problem (ECDLP) serves as the base for the same and considered to be a hard problem. Large size of the finite field also increases the hardness of the problem. The field is fixed at 160-bit for the minimum level of security. National Institute of Standards and Technology (NIST) recommends five type of binary extension fields i.e. 163, 233-, 283-, 409- and 571-bit fields [2].

Elliptic curve cryptographic processor (ECCP) can be used alongside the main processor for he for securing information. Scalar multiplication is the most important among all the elliptic curve operations. It uses point doubling and addition and finite field arithmetic operations. Finite field multiplication and inversion consumes highest number of resources. Finite field multiplication is also required for the realization of inversion operation.

IoT is marked by the resource constrained devices. To adapt ECCP for the resource constrained operations, finite field multiplier (FFM) must be reduced in size. Delay is should be limited for such an operation as most IoT applications are real time. Hence, our primary concern is fast and compact design for the multiplier.

Bit-serial and bit-parallel multipliers are two important types of multipliers [3]. Arithmetic operations decide the performance of a multiplier. Bit-parallel multipliers can be categorized in:
Quadratic and Sub-quadratic multipliers based on arithmetic complexities. Space and time complexity provides the size and delay of the circuit. Sub-quadratic multipliers have low space complexity with small delay. A hybrid algorithm can be realized by combining two algorithms to achieve a trade-off in size and delay.

This work presents a hybrid algorithm to generate a fast and compact finite field multiplier for 283-bit

binary extension field. The design is synthesized on PlanAhead software from Xilinx and implemented on different Xilinx FPGAs for the sake of deeper insights. The best platform for the FPGA implementation is suggested after thorough analysis of these implementations.

Rest of the paper is organized as follows: Various contemporary implementations are discussed along with the older ones in Section 2. Basic Karatsuba algorithm has been discussed and Binary, Simple, general and Hybrid-Karatsuba multiplier has been presented in Section 3. Results of hardware implementations are presented and discussed in detail under Section 4. A comparison with existing designs is also carried out in this section. Finally, conclusions are drawn following the future work directions in Section 5.

RELATED WORK

Elliptic curve cryptography provides better security per key-bit when compared to other asymmetric versions. Scalar multiplication in elliptic curves is a very important operation in deciding the performance of this technique. Hence, there have been number of efforts in this direction to improve the algorithm. In [2], authors have purposed a design to use two different algorithms to calculate scalar multiplication in elliptic curve cryptography (ECC). One of the algorithms was binary karatsuba multiplication while the other was classical algorithm. Authors obtained a faster algorithm by combining these two algorithms. In [3], another faster hybrid Karatsuba multiplier was designed using the same concept but replaced the classical algorithm with different version of Karatsuba multiplication. In [4], a digit serial multiplier with segmented pipeline was presented for ECC multiplication. It also presented a combined algorithm for addition and doubling and with proper scheduling reduced the latency in multiplier. A modified Montgomery ladder was used to present a new architecture for the point multiplication in [5]. It is designed for generic irreducible polynomials.

Approximate computing- based design of a Karatsuba multiplier was presented in [6] that used low error rate to improve the speed and minimize the hardware requirement of the multiplier. In [7], completion detection logic was used to remove hardware registers from multiplier structure. It helped in implementing with only two multipliers. Inversion is done with Fermat's little theorem. The asynchronous design proposed by the authors is aimed at lower power consumption. In [8], authors used pipelining and parallelized the multiplication and squaring operation to reduce the critical path delay of ECPM. It uses modified Lopez-Dahab Montgomery algorithm for multiplication. The effort covers three fields 163, 233, 283.

A concurrent reconfigurable system for ECC is proposed in [9] that utilizes mapping on FPGA. The design implements ElGamal cryptosytem over elliptic curves using isomorphic transformation. A residue number system based ECC core for twisted Edwards and Weierstrass curves is designed in [10]. In [11], authors converted montgomery algorithm to a four clock-cycle multiplication and also proposed a six clock-cycle dual-field point multiplier for ECC. The main aim of this dual-field architecture is to optimally utilized the Karatsuba multiplier. Super-singular isogeny key encapsulation (SIKE) protocol is a strong candidate for the post quantum cryptography but latency make less competitive. A new SIKE algorithm that is built with a compact control logic is proposed in [12], it uses an extremely low-latency modular multiplier, and an efficient memory access method to minimize latency and optimize speed. In [13], authors have proposed a new design for ECPM that is suitable for variable operand size and exhibit lower combinational path delay. Authors in [14] proposed a ECC based DSA that uses ED448 curves. A refined Karatsuba multiplier is used with accurate scheduling to provide improved performance. Its implementation uses the DSPs available on the FPGA to produce higher throughput.

A lightweight implementation of ECC over prime fields is proposed in [15]. It uses an efficient and small modular Montgomery multiplier. Authors proposed a modified radix-2 interleaved algorithm for modular multiplication in [16]. It achieves high efficiency. In [17], authors presented an n-term Karatsuba multiplier for the arbitrary irreducible polynomial. An efficient way to calculate the Montgomery modular multiplication is presented in [18]. It implemented 256-bit and 1024-bit modular multiplier and achieved better area and speed results. In [19], authors have presented a detailed survey on the lightweight cryptographic techniques to secure IoT networks where the details of asymmetric cryptography have also been discussed. To ensure the safety of patient's collected data authors of [20] have proposed a secure lightweight anonymous authentication protocol which is based on elliptic curve cryptography. Mutual authentication can help establishing a safe communication link between service provider and consumer in smart grid network. It removes many cyber threats. Authors in [21] have proposed a lightweight authentication scheme for the smart -grids to ensure safe and secure communication.

Though there have been many efforts to improve the performance of the ECCP. Limited efforts can be seen in the direction of designing efficient multipliers which can help in improving the performance of

eISSN 1303-5150

ECCP. This was the main motivation to design a hybrid Karatsuba multiplier for GF(2^{283}) which will be utilized in ECCP of GF(2^{283}). The contribution of our work can be summarized as follows:

i) A hybrid karatsuba multiplier has been designed for GF(2^{283}).
ii) The design has been implemented on different FPGAs and detailed analysis has been carried out.
iii) The design is compared with the existing bit parallel designs in space complexities as well.
iv) State of the art in the field has also been presented.

MULTIPLIER'S DETAILS

Multiplication of two polynomials can be defined as:

$$W(x) = U(x).V(x) mod\ P(x) \quad (1)$$

Here, $U(x), V(x)\ and\ P(x) \in GF(2^m)$. *P(x)* is an irreducible polynomial which can generate the complete field *GF(2^m)*. This process is carried out in two steps, in the first $W(x) = U(x).V(x)$ is calculated, after that modulo of the *W'(x)* is calculated.

The Karatsuba algorithm is a sub-quadratic algorithm that is used to calculate a polynomial multiplication efficiently. This efficiency is achieved through splitting of multiplicand polynomial in two terms polynomial as shown in equation (2.1). Finally, the multiplication is completed by three multiplications of *m/2* each as depicted by equation (2.3).

$$W'(x) = (U_h x^{m/2} + U_l)(V_h x^{m/2} + V_l) \quad (2.1)$$

$$= U_h V_h x^m + (U_h V_l + U_l V_h) x^{m/2} + U_l V_l \quad (2.2)$$

$$= U_h V_h x^m + \left((U_h + U_l)(V_h + V_l) + U_h V_h + U_l V_l\right) x^{m/2} + U_l V_l \quad (2.3)$$

Algorithm 1 below presents Karatsuba-ofman method for the multiplication of polynomials. In the field *GF(2^m)*, recursions of basic Karatsuba multiplier will help in increasing the gains. A fully recursive Karatsuba multiplier can be used to get the most efficient multiplications but only in the field *GF(2^m)*. It can require *k* iterations of the algorithm. The number of gates required for such a multiplier are given as:

$$\#AND\ gates = m^{\log_2 3} \quad (3)$$

$$\#XOR\ gates = \sum_{r=0}^{\log_2 m} 3^r \left(\frac{4m}{r} - 4\right)$$

There are different types of Karatsuba multipliers that can be used for multiplication: Padded Karatsuba, binary Karatsuba, Simple Karatsuba multipliers, General Karatsuba multipliers. But we are going to discuss about binary-, simple-, general-, and hybrid Karatsuba algorithms in detail.

uses a method w It removes the extra arithmetic that will be introduced due to the. Arbitrary field sizes. First of all, in [5] a hybrid Karatsuba multiplier was designed. It combined classical algorithm with the binary-Karatsuba-Ofman multiplier. The logic behind hybrid Karatsuba algorithm was to truncate the multiplication at n-bit and perform rest of the multiplication by a different algorithm to achieve the lower time and space complexities.

Simple Karatsuba Multiplier

A small modification in basic karatsuba multiplier converts it to simple karatsuba multiplier. Dividing the polynomials, A and B in to two halves with $\lceil m/2 \rceil$ and $\lfloor m/2 \rfloor$ terms in A_l, B_l and A_h, B_h respectively, to complete an m-bit multiplication. The multiplication can be completed with a single $\lfloor m/2 \rfloor$ bit multiplication and two $\lceil m/2 \rceil$ bit multiplications. A $2^{\lceil \log_2 m \rceil}$ bit basic karatsuba multiplier and simple karatsuba multipliers have same higher bound for the required number of AND and XOR gates but the latter requires lesser gates for implementation. Simple karatsuba requires $\lceil \log_2 m \rceil$ recursions as compared to $\lfloor \log_2 m \rfloor$ recursions required for binary karatsuba multiplier. It results in larger delay for simple karatsuba multiplier. Equation (6) presents the number of gates for the implementation:

$$\#AND\ gates = 3^{\log_2 m} \quad (6)$$

$$\#XOR\ gates = \sum_{r=0}^{\log_2 m} 3^r (4\lceil m/2^r \rceil - 4)$$

General Karatsuba Multiplier

In case of general karatsuba multiplier, process of multiplying two m-bit polynomials is completed using three multipliers with m/2 bit each. The n-bit polynomial is divided into three terms each m/2 bit long.

$$W = UV \quad (7.1)$$

eISSN 1303-5150

$$= \left(U_2 x^{2m/3} + U_1 x^{m/3} + U_0\right)\left(V_2 x^{2m/3} + V_1 x^{m/3} + V_0\right)$$

(7.2)

$$= U_2V_2x^{4m/3} + (U_2V_1 + U_1V_2)x^m + (U_2V_0 + U_0V_2 + U_1V_1)x^{2m/3} + (U_0V_1 + U_1V_0)x + U_0V_0$$

(7.3)

$$= U_2V_2x^{4m/3} + \left((U_2 + U_1)(V_2 + V_1) + U_2V_2 + U_1V_1\right)x^m + ((U_2 + U_0)(V_2 + V_0) + U_2V_2 + U_1V_1 + U_0V_0)x^{2m/3} + ((U_1 + U_0)(V_1 + V_0) + U_1V_1 + U_0V_0)x^{m/3} + U_0V_0$$

(7.4)

Other formulas related to multiplication of two polynomials are as follows: the equations $Z_i = U_iV_i$ and $Z_{p,q} = \left(U_p + U_q\right)\left(V_p + V_q\right)$ and W_0, W_{2m-2}, W_i are coefficients of W.

$W_0 = Z_0$

(8.1)

$W_{2m-2} = Z_{m-1}$

(8.2)

$W_i = \sum_{\substack{p+q=i \\ q>p\geq 0}} Z_{p,q} + \sum_{\substack{p+q=i \\ n>q>p\geq 0}} \left(Z_p + Z_q\right)$

for odd i 0<i<2m-2

(8.3)

$W_i = \sum_{\substack{p+q=i \\ q>p\geq 0}} Z_{p,q} + \sum_{\substack{p+q=i \\ n>q>p\geq 0}} \left(Z_p + Z_q\right) + Z_{i/2}$

for even i 0<i<2m-2

(8.4)

Complete algorithm is shown in Algorithm 3.

Algorithm 3: General Karatsuba Multiplier (*gen_kar_mul*)

Input: U, V are m-bits in length
Output: W is 2m-1 bits in length
Define $W_p = U_pV_p$
Define $W_{(p,q)} = \left(U_p + U_q\right)\left(V_p + V_q\right)$

Begin

for $i = 0$ *to* $m-2$
  $W_i = W_{2m-2-i} = 0$
  for $j = 0$ *to* $\lfloor i/2 \rfloor$
    if $i = 2j$ *then*
      $W_i = W_i + M_j$
      $W_{2m-2-i} = W_{2m-2-i} + M_{m-1-j}$
    else
      $W_{2m-2-i} = W_{2m-2-i} + M_{m-1-j} + M_{m-1-i+j} + M_{m-1-j,m-1-i+j}$
    end
  end
end

$W_{n-1} = 0$
for $j = 0$ to $\lfloor (m-1)/2 \rfloor$ do
  if $m - 1 = 2j$ then
    $W_{m-1} = W_{m-1} + M_j$
  else
    $W_{m-1} = W_{m-1} + M_j + M_{m-1-j} + M_{(j,m-1-j)}$
  end
end

end

The space complexity of the general Karatsuba [6] is given by the following equation:

$$\#AND\ gates = m(m+1)/2$$
$$\#XOR\ gates = \frac{5}{2}m^2 - \frac{7}{2}m + 1 \quad (9)$$

Hybrid Karatsuba Multiplier

The logic behind the use of these two techniques is to maximize the use of LUTs that will result in minimization of the LUT consumption. As per [6], the number of under-utilized LUTs in m-bit general Karatsuba are lesser in comparison to m-bit simple Karatsuba for the values of m that are less than 29. Hence, a combination of these two multipliers to design a hybrid Karatsuba multiplier that uses general Karatsuba for 'm<29-bits' and for higher values of m uses simple Karatsuba will result in a much efficient multiplier in which number of under-utilized LUTs are quite less.

Algorithm 4: Hybrid Karatsuba Multiplier (*hy_kar_mul*)

Input: U, V are m-bits in length
Output: W is 2m-1 bits in length

begin

if $m < 29$ ***then***
  return $gen\_kar\_mul(U, V, m)$

else
  $l = \lceil m/2 \rceil$
  $U' = U_{[m-1\ldots.l]} + U_{[l-1\ldots.0]}$
  $V' = V_{[m-1\ldots.l]} + V_{[l-1\ldots.0]}$
  $W_{p1} = hy\_kar\_mul(U_{[l-1\ldots.0]}, V_{[l-1\ldots.0]}, l)$
  $W_{p2} = hy\_kar\_mul(U', V', l)$

  $W_{p3} = hy\_kar\_mul(U_{[m-1\ldots l]}, V_{[m-1\ldots.l]}, m-1)$
  return $\left(W_{p3} \ll 2l\right) + \left(W_{p1} + W_{p2} + W_{p3}\right) \ll l + W_{p1}$

end

end

Equation (10) provides the number of LUTs that are used in the *m*- bit recursive Karatsuba multiplication is given by

$$\#LUTs = 3 \times 3^{\log_2 m - 1} + \sum_{k=0}^{\log_2 m - 2} 3^k \left(2 \times 2^{\log_2 m - k} - 1\right)$$

eISSN 1303-5150

$$= \sum_{k=0}^{\log_2 m-1} 3^k (2^{\log_2 m-k+1} - 1)$$

(10)

Simple Karatsuba multiplier [21] consumes nearly same resources as given by equation (10). While, the total number of LUTs for general Karatsuba multiplier consumes is given by equation (11).

$$\#LUTs = 2\left(\sum_{i=0}^{m-2} LUT_{W_i}\right) + LUT_{W_{m-1}}$$

(11)

These LUTs for a particular value of Wi are calculated by the following formula:

LUTs required for complexity Wi

$$\#LUT_{W_i} = \begin{cases} 1 & if\ i = 0 \\ \lceil i/2 \rceil + LUT(\lceil i/2 \rceil)\ \ if\ i\ is\ odd \\ i/_2 + LUT(i/2+2)\ \ if\ i\ is\ even \end{cases}$$

(12)

Hybrid Karatsuba multiplier is a combination of two type of multipliers; simple and general karatsuba multipliers. Simple Karatsuba uses two multipliers one with $\lceil m/2 \rceil$ and another one of $\lfloor m/2 \rfloor$ bit in length. All the recursion of hybrid Karatsuba multiplier which are greater than or equal to 29-bit uses simple Karatsuba multiplier. On the other hand, general Karatsuba multiplier is used for any multiplication less than 29-bit. Hence, the total number of AND gates can be calculated as AND gates needed for last recursion times the number of $\lceil m/2^{r-1} \rceil$ multipliers present.

$$\#AND = 3^{r-1} \left\lceil \frac{m}{2^r} \right\rceil \left(\left\lceil \frac{m}{2^{r-1}} \right\rceil + 1 \right)$$

(13)

Again, *XOR* gates for i^{th} recursion is given as $4\left\lceil \frac{m}{2^i} \right\rceil - 4$. The final count *XOR* gates that are of 2-input *can* be calculated by adding the *XORs of* last recursion ($\#XOR_{g_{r-1}}$) with *XORs* used in rest of the recursions ($\#XOR_{S_i}$).

$$\#XOR = 3^{r-1}\#XOR_{g_{r-1}} + \sum_{i=0}^{r-2} 3^i \#XOR_{S_i}$$
$$= 3^{r-1}\left(10\left\lceil \frac{m}{2^r} \right\rceil^2 - 7\left\lceil \frac{m}{2^r} \right\rceil + 1\right) + \sum_{i=0}^{r-2} 3^i \left(4\left\lceil \frac{m}{2^i} \right\rceil - 4\right) \quad (14)$$

RESULTS AND DISCUSSION

The design of the 283-bit hybrid Karatsuba multiplier is discussed in previous section. This design is coded in Verilog language. The design has been synthesized using Xilinx PlanAhead 14.6 software. It is implemented on various Xilinx FPGAs for a detailed analysis.

Table 1 provides the details of the resource utilization as obtained for different implementation. It also includes the delay of the multiplier which is a measure of performance. Fig 1 shows the comparison of resource usage on different FPGAs from Table 1. The maximum number of LUTs are consumed on Virtex-4 FPGA 24756 LUTs while Spartan-3 implementation is largest with 13808 slices. Resource consumption is minimum in terms of LUTs on Virtex-5 FPGAs. In terms of slice consumption Virtex-6 implementation is smallest with 7532 slices.

Table 1: HKMul-283

S. No.	HKMUl-283 on FPGA Board	LUTs	Slices	Delay (ns)
1.	Spartan-3	24608	13808	26.216
2.	Virtex-4	24756	12756	17.413
3.	Virtex-5	19253	10850	15.367
4.	Virtex-6	20150	7532	8.764
5.	Virtex-7	20473	9337	8.041

Fig 1 depicts that the resources consumed as per table 1. Virtex-5 FPGAs based implementation is smallest in terms of LUT consumption with 19253 LUTs. This is surprising when we compare this to other two advanced FPGAs i.e., Virtex-6 and -7. Similarly, Virtex-6 also performs better than any other FPGA as the design consumes 7532 slices on it. It outperforms Virtex-7 in terms of slice consumption in site of better CMOS technology used in it.

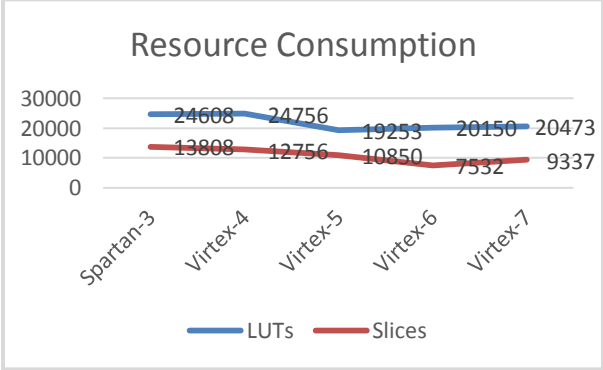

Fig 1: Comparison of design on different FPGAs on resource consumption

Fig 2 depicts the performance of the multipliers in terms of delay noted on different FPGAs. This trends as per technology advancement. As the design moves from oldest to newest FPGA it displays better results. Virtex-7 being the fastest with 8.041 ns and Spartan-3 with longest delay with 26.216 ns. As the levels of logic are 22 in design implementation on the Virtex-7. Moreover, the number of XoR gates used are 27312 in Virtex-7. In rest of the FPGAs, level of

logic stands at 22 while XOR gates used are 31836. This is due to the increase in the inputs of LUTs in Virtex-7.

Overall, Virtex-6 can be considered as the best suited FPGA for the design. It achieves second minimum LUT consumption while it has smallest slice consumption. In terms of delay it is the second best and outperforms Spartan-3, Virtex-4 and Virtex-5 by 66.57% , 49.67% and 42.97 percent respectively while it is 8.21 percent slower than Virtex-7.

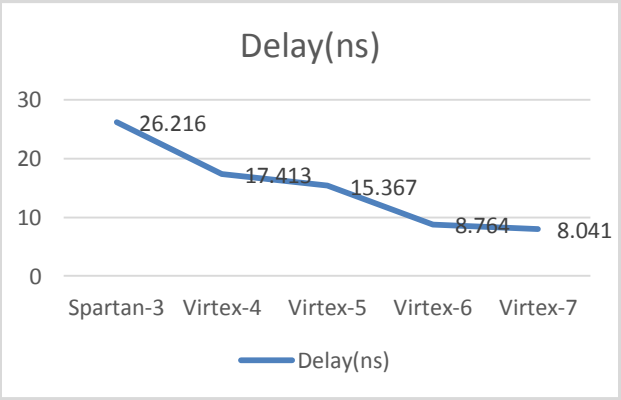

Fig 2: Delay comparison of design on different FPGAs

Table 2: Comparison of different design on Space Complexity

Multipliers	**Irreducible Polynomials**	**AND gates**	**XOR gates**
Yeh et al [22]	General Form	$2m^2$	$2m^2$
Wang-Lin [23]	General Form	$2m^2$	$2m^2$
Chiou et.al. [24]	General Form	$2m^2+2m$	$2m^2+2m$
Lee [25]	Trinomial	m^2	m^2+m
Lee [26]	Trinomial	m^2	
Park et al.[27]	Pentanomial	$\frac{3m^2+14m+23}{4}$	$\frac{3m^2+2m-1}{4}$
This work	General form	$3^{r-1}\left\lceil\frac{m}{2^r}\right\rceil\left(\left\lceil\frac{m}{2^{r-1}}\right\rceil+1\right)$	$3^{r-1}\left(10\left\lceil\frac{m}{2^r}\right\rceil^2-7\left\lceil\frac{m}{2^r}\right\rceil+1\right)+\sum_{i=0}^{r-2}3^i\left(4\left\lceil\frac{m}{2^i}\right\rceil-4\right)$

Table 2 presents the details of different designs. It compares all the designs on their space complexities. It presents the number of AND and XOR gates consumed by the respective designs. As can be seen the table all the design have quadratic complexities $\Theta(n^2)$ while the presented design exhibits sub-quadratic complexity which is $\Theta(n^{1.58})$. Hence, it can be clearly stated that the design achieves smaller size in comparison to above mentioned designs. It will help in minimizing size of ECCP when used in the design.

CONCLUSION AND FUTURE WORK

Multipliers plays an important role in determining the performance of elliptic curve processors. They form the base for a highly efficient scalar multiplication. Karatsuba multipliers are the sub-quadratic multipliers that can be used to enhance the performance of ECP. Efforts have been made to improve the elliptic curve processors with different strategies such as pipelining. In this work, we have proposed a hybrid karatsuba multiplier for GF(2283). The proposed multiplier is comparable to the available compact multipliers but is nearly two time faster. This multiplier can be used to improve the performance of an ECC processor for GF(2^{283}). In future, we will be trying to do the same.

ACKNOWLEDGMENTS

REFERENCES

1. S. S. Dhanda, B. Singh, P. Jindal **(2020),** "IoT Security: A Comprehensive View", S.-L. Peng et al. (eds.), Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm, Intelligent Systems Reference Library 174, page 467-493 https://doi.org/10.1007/978-3-030-33596-0_19
2. F. R. Henriquez, N.A. Saqib, A. Diaz-Perez, "A Fast parallel of elliptic curve point multiplication over GF(2m)", Microprocessors and Microsystems 28(2004) 329-339.
3. C. Rebeiro and D. Mukhopadhyay, "Power Attack Resistant Efficient FPGA Architecture for Karatsuba Multiplier," 21st International Conference on VLSI Design (VLSID 2008), 2008, pp. 706-711, DoI: 10.1109/VLSI.2008.65.

4. P.K.G. Nadikuda, L. Boppana **(2022)**, "An area-time efficient point-multiplication architecture for ECC over GF(2m) using polynomial basis", Microprocessors and Microsystems, Vol. 91 **(2022)** 104525

5. Riya Jain, Neeta Pandey **(2021)** "Approximate Karatsuba multiplier for error-resilient applications", Int. J. Electron. Commun. (AEÜ) 130 **(2021)** 153579
6. Mohamed Asan Basiri M, Sandeep K. Shukla, **(2019)** "Asynchronous hardware implementations for crypto primitives", Microprocessors and Microsystems 64 (2019) 221–236
7. N.P. Kumar, Shirisha C., **(2020)** "An area-efficient ECC architecture over GF(2m) for resource-constrained applications", Int. J. Electron. Commun. (AEÜ) 125 (2020) 153383
8. Rami Amiri and Omar Elkeelany, **(2021)** "FPGA Design of Elliptic Curve Cryptosystem (ECC) for Isomorphic Transformation and EC ElGamal Encryption", IEEE Embedded Systems Letters, Vol. 13, No. 2, June 2021 65-68
9. Mohamad Ali Mehrabi, Christophe Doche, and Alireza Jolfaei, **(2020)** "Elliptic Curve Cryptography Point Multiplication Core for Hardware Security Module", IEEE Transactions on Computers, Vol. 69, No. 11, November 2020 1707-1718
10. J. Li, W. Wang, J. Zhang, Y. Luo, and S. Ren, **(2021)** "Innovative Dual-Binary-Field Architecture for Point Multiplication of Elliptic Curve Cryptography", IEEE Access Vol. 9, 2021, *DOI: 10.1109/ACCESS.2021.3051282*
11. J.Tian, B.Wu, and Z. Wang, **(2021)** "High-Speed FPGA Implementation of SIKE Based on an Ultra-Low-Latency Modular Multiplier", IEEE Trans. on Circuits and Systems—I: Regular Papers, Vol. 68, No. 9, September 2021 3719-3731
12. M. Heidarpur and M. Mirhassani, **(2021)** "An Efficient and High-Speed Overlap-Free Karatsuba-Based Finite-Field Multiplier for FGPA Implementation", IEEE Transactions on Very Large-Scale Integration (VLSI) Systems, Vol. 29, No. 4, April 2021 pp-667
13. M.B.Niasar, R. Azarderakhsh , and M.M. Kermani **(2021)**, "Area-Time Efficient Hardware Architecture for Signature Based on Ed448", IEEE Trans. on Circuits and Systems—II: Express Briefs, Vol. 68, No. 8, August 2021, Pp2942
14. A.A.H. Abd-Elkader et.al. **(2021)** "FPGA-Based Optimized Design of Montgomery Modular Multiplier", IEEE Transactions on Circuits and Systems—II: Express Briefs, Vol. 68, No. 6, June 2021, 2137
15. Md. Mainul Islam et. al. **(2020)** "Area-Time Efficient Hardware Implementation of Modular Multiplication for Elliptic Curve Cryptography", *Digital Object Identifier 10.1109/ACCESS.2020.2988379*
16. Yin Li, et. al. **(2020)** "On the Complexity of Hybrid *n*-Term Karatsuba Multiplier for Trinomials", IEEE Trans on Circuits and Systems–I: Regular Papers, Vol. 67, No. 3, March 2020 Pp-852
17. A.A.H. Abd-Elkader et. al., "Efficient implementation of Montgomery modular multiplier on FPGA", Computers & Electrical Engineering, Volume 97, **2022**, 107585, https://doi.org/10.1016/j.compeleceng.2021.107585.
18. Muhammad Rana, Quazi Mamun, Rafiqul Islam, **(2022)** "Lightweight cryptography in IoT networks: A survey"' Future Generation Computer Systems 129 (2022) 77–89
19. K Sowjanya, M. Dasgupta and Sangram Ray, **(2021)** "Elliptic curve cryptography-based authentication scheme for Internet of Medical Things", Journal of Information Security and Applications 58 (2021) 102761.
20. D Sadhukan, S Ray, Mohd S Obaidat, Mou Dasgupta, **(2021)** "A secure and privacy preserving lightweight authentication scheme for smart-grid communication using elliptic curve cryptography", Journal of System Architecture 114 (2021) 101938.
21. C.S. Yeh, S. Reed, T.K. Truong, Systolic Multipliers for Finite Fields GF(2m), IEEE Trans. Comput. 33 (4) (1984) 357–360.
22. C.L. Wang, J.L. Lin, Systolic array implementation of multipliers for GF(2m), IEEE Trans. Circuits Systems II 38 (7) (1991) 796–800.
23. C.W. Chiou, C.Y. Lee et al., "Concurrent error detection in montgomery multiplication over GF(2m)", IEICE Trans. Fund. E89-A (2) (2006) 566–574.
24. C.Y. Lee, Low-complexity bit-parallel systolic multiplier over GF(2m) using irreducible trinomials, IEE Comput. Digital Techn. 144 (1), (2003) 39–42.

eISSN 1303-5150

25. C.Y. Lee et al. "Low-complexity bit-parallel systolic montgomery multipliers for special classes of GF(2m)", IEEE Trans. Comput. 54 (9) (2005) 1061–1070.

26. Sun-Mi Park, et al."New efficient bit-parallel polynomial basis multiplier for special pentanomials", INTEGRATION, the VLSI Journal, Vol. 47 (2014) Pages:130–139

eISSN 1303-5150