2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 7-9 February, 2019

Efficient FPGA Implementation of Modular Arithmetic for Elliptic Curve Cryptography

Md. Rownak Hossain and Md. Selim Hossain
Department of Electrical and Electronic Engineering (EEE), Khulna University of Engineering & Technology (KUET)
Khulna-9203, Bangladesh
rownok.eee@gmail.com , selim@eee.kuet.ac.bd

Abstract— ***High throughput and low resource are the crucial design parameters of elliptic curve cryptographic (ECC) processor in many applications. The efficiency of ECC processor mainly depends on modular arithmetic operations such as modular addition, subtraction and multiplication. In this brief, hardware architectures of modular arithmetic over the prime field are presented. The novelty of this work is to execute modular addition and modular subtraction in a combined architecture. In addition, multiplication and modular operations are executed in a separate module instead of doing modular multiplication in a single operation which has significantly improved circuit latency and area optimization. The proposed architectures have been synthesized on Virtex-5 field-programmable gate array (FPGA) technology and achieved 0.575µs and 2.04µs computational time, as well as 4% and 17% of available slice-LUTs for 256-bit, combined modular addition and subtraction and modular multiplication respectively. The design offers almost 50-80% area minimization and reduces nearly 10-90% of the required time in comparison with the related designs. To our best knowledge, our architectures provide a better throughput/area performance on FPGA than the recent related work which is very impressive for the design of ECC processor.***

Keywords-Elliptic curve cryptography (ECC), modular multiplication, combined modular addition and subtraction, field-programmable gate array (FPGA)

I. INTRODUCTION

Secure transactions over the network have become the burning issue in today's world with the advancement of digital systems, where the transfer of information takes place within a reachable environment like the Internet and wireless network. Moreover, the sharp growth of IOT based system is connecting various types of operators such as computing devices, mechanical and digital machines, objects or people in a common platform. Therefore, ensuring security has moved to the forefront of digital systems. Strong and efficient cryptography can play a vital role to mollify these risks as well as ensures authentication, authorization, data confidentiality and data integrity. Private-key cryptography and public-key cryptography (PKC) are the two main schemes of cryptography which are used for protection. Though private-key cryptography is computationally less expensive than PKC, it has a key distribution problem. In contrast, PKC provides flexible key management but their huge computational complexity is an obstacle to implement it in hardware. With the technological advancement, the motif of the designer should be an efficient hardware implementation of PKC which requires fewer hardware resources with a high throughput rate. The Efficient design of modular arithmetic unit provides the platform to reach the ultimate goal of a high-performance cryptographic processor [1-3]. RSA [4] and ECC [5, 6] are considered as the two most powerful and popular methods of PKCs. But ECC is chosen among these two because of its mathematical hardness, shorter key length and less computational complexity [1]. ECC algorithms are mainly implemented in three platforms: software, FPGA and ASIC. Software approach is very time-consuming to fulfill the requirements for real-time and embedded applications. ASIC implementation provides more speed but costs much to build a complete cryptosystem. Considering these, FPGA is a wonderful implementation environment for prototype design because they can drastically reduce the hardware development, test cost and time since they do not require any fabrication cost.

The modular arithmetic unit is regarded as the heart of an ECC processor. Modular addition, subtraction, multiplication and inversion are the frequently encountered operations in these types of processor, where modular inversion can be avoided using Jacobian coordinates. As a cryptosystem has to perform a lot of encryption and decryption processes, the efficiency mostly depends on the performance of modular arithmetic operations. The challenge of implementing low resourced high-performance modular arithmetic unit has attracted researchers to contrive the best possible model for the world. There is a variety of work performed aiming at high-speed as well as minimum hardware resources. Modular multiplication architectures over the 256-bit prime field for ECC were modeled by Ghosh [11], Duan [12], Fan [13] and Daly [14]. Modular multiplication based on radix-4 Montgomery method over the prime field was implemented by Lee [10]. However, most of the work offer high-speed devices sacrificing the goal of minimum resources or low resourced devices without considering the intent of a high throughput rate. For this reason, efficient design of modular arithmetic is a very demanding job for the implementation of high-performance ECC processor [15].

Hardware implementations of modular arithmetic for 256-bit ECC processor over the prime field on field-programmable gate array (FPGA) are the main focus of this work. In this work, for doing this a model has been established which can perform both modular addition and modular subtraction in a single module instead of two separate modules. Then, a variety of multiplication algorithms such as Normal multiplication, radix-2 Booth multiplication, radix-4 Booth multiplication and Moore multiplication have been tried which are later integrated with the modular reduction module to obtain the optimized result both in terms of area usage and delay. Finally, the best design among our works has been chosen which can be used for high-performance ECC processor implementation.

This paper is structured as follows. Section II demonstrates the preliminaries of finite field arithmetic and algorithms related to our works. Section III covers the proposed hardware architectures of our work. Section IV deals with the implementation results and performance analysis which are done in two stages – firstly, a comparison has been made among our architectures and then the best of our architecture has been compared with other recent work, followed by conclusions in Section V.

II. MATHEMATICAL BACKGROUND

A. Finite field arithmetic

Fields, denoted by F, are abstractions of number systems and the properties associated with them. A field is considered to be finite, when the set F is finite. Mainly, addition and multiplication operations are involved in field arithmetic. Subtraction in field arithmetic can be defined in the form of addition for $(a, b) \in F$, $a - b = a + (-b)$ where $-b$ is the unique element in F such that $b + (-b) = 0$. In the same way, inversion or division in field arithmetic can be defined in the form of multiplication for $(a, b) \in F$ with $b \neq 0$, $a/b = a.b^{-1}$ where b^{-1} is the unique element in F such that $b.b^{-1} = 1$. Finite field order (q) is determined by the total number of elements contained in a field. A finite field called prime field when the order q can be represented by a prime power $(q = p^m)$, where p is a prime number and $m = 1$ [1].

B. Modular addition

Modular addition over GF(p) is a basic operation for cryptosystem. Mathematically, it can be written as $Z = (x + y) \bmod p$ where x and y are the given number and p is the prime number. Here, Output Z is obtained by adding x and y and then subtracting p from $(x + y)$ until the result is less than p. While performing modular addition, less concern is given to modular reduction operation because the inputs x and y are in the range 0 to $p-1$ and therefore Z must be $\leq 2p$.

C. Modular subtraction

Modular subtraction over GF(p) is another basic and essential operation for cryptosystem. Mathematically, modular subtraction can be written as $Z = (x - y) \bmod p$ where x and y are the given number and p is the prime number. In modular subtraction, if $x \geq y$ then it can be easily computed by simple subtraction or 2's complement addition. But if $x < y$ then p should be added to the input x before starting the subtraction.

D. Modular multiplication

Modular multiplication is one of the most expensive and time consuming operation over GF(p). Efficient implementation of modular multiplication is a must to develop a high performance cryptosystem. Mathematically, output of modular multiplication can be expressed as $Z = (M \times R) \bmod p$. In this research, two separate modules are established for modular multiplication; one for multiplication operation and another for modular reduction operation.

Different algorithms used for modular addition, modular subtraction, multiplication and modular reduction are mentioned below:

***Algorithm 1: Addition in* GF(*p*) [1]**

```
Input:   Modulus $p$ and integers $x, y \in [0, p-1]$
Output: $Z = (x + y) \bmod p$

1: Set $A = x$, $B = y$
2: Set $V = A + B$
3: if $V \geq p$  then $Z = V - p$
   Else $Z = V$
4: return $Z$
```

***Algorithm 2: Subtraction in* GF(*p*) [1]**

```
Input:   Modulus $p$ and integers $x, y \in [0, p-1]$
Output: $Z = (x - y) \bmod p$

1: Set $A = x$, $B = y$
2: if $A \geq B$ then $Z = A - B$
   Else $Z = (A + p) - B$
3: return $Z$
```

***Algorithm 3: Normal multiplication* [1]**

```
Input:   Integers $M, R$
Output: $Z = M*R$

1: Set $A = M$, $B = R$, $P = Zeros$, $Q = Zeros$
2: for $i$ in 0 to 255 do
   $sel = A[i]$
            if $sel == 1$ then  $Q = B$
            else  $Q = Zeros$
   $P[i] = Q + P[i-1]$
3: $Z = P$
4: return $Z$
```

***Algorithm 4: Booth Radix-2 multiplication* [7]**

```
Input:   Integers $M, R$
Output: $Z = M*R$

1: Set $A = M$ & $Zeros$, $S = -M$ & $Zeros$
2: Set $P = Zeros$ & $R$ & $zero$
3: for $i$ in 0 to 255 do
   $sel = A[i+1]A[i]$
            case $sel$ is
                        when $b$ "01" => $P = P+A$
                        when $b$ "10" => $P = P+S$
                        when others  => $P = P$
            end case
   $P = P / 2$
4: $Z = P$
5: return $Z$
```

***Algorithm 5: Booth Radix-4 multiplication* [8]**

```
Input:   Integers $M, R$
Output: $Z = M*R$

1: Set $A = M$, $B = R$, $P = Zeros$
2: Set $S = 2*R$
3: for $i$ in 0 to 127 do
   $sel = A[i+2]A[i+1]A[i]$
            case $sel$ is
                        when $b$ "001" | "010"  => $P = P+B$
                        when $b$ "011"   => $P = P + 2*B$
                        when $b$ "100"   => $P = P - 2*B$
                        when $b$ "101" | "110"   => $P = P - B$
                        when others => $P = P$
            end case
   $P = P / 2$
4: $Z = P$
5: return $Z$
```

Algorithm 6: Moore multiplication **[7]**

Input: Integers *M, R*
Output: *Z = M*R*

```
1: Set $A = Zeros$, $B = M$, $C = R$, $AQ = Zeros \& R$
2: for $i$ in 0 to 255 do
    $sel = AQ[0]$
        if $sel = 1$ then $A = A + M$
        else $A = Zeros$
    $AQ = AQ / 2$
3: $Z = A$
4: return $Z$
```

Algorithm 7: Fast reduction modulo $p_{256} = 2^{256}-2^{224}+2^{192}+2^{96}-1$ **[1, 9]**

Input: An integer $c = (c_{15}, \dots \dots, c_1, c_0)$ in base 2^{32} with $0 \le c < p2_{256}$
Output: $c \bmod p_{256}$

```
1: Define 256-bit integers:
        $S_1 = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
        $S_2 = (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0)$,
        $S_3 = (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0)$,
        $S_4 = (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8)$,
        $S_5 = (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9)$,
        $S_6 = (c_{10}, c_8, 0, 0, 0, c_{13}, c_{12}, c_{11})$,
        $S_7 = (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12})$,
        $S_8 = (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13})$,
        $S_9 = (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})$,
2: return $(S_1 + 2S_2 + 2S_3 + S_4 + S_5 - S_6 - S_7 - S_8 - S_9 \bmod p_{256})$
```

The above mentioned algorithms for modular arithmetic are explained in the next section.

III. Hardware architecture

High speed and flexible hardware design of modular arithmetic units significantly improve the performance of ECC processor. In our research, six architectures have been formed in total for modular addition, subtraction and multiplication.

The architecture showed in Fig. 1 operates according to the selection of operation (addition or subtraction). One of two different types of values will be stored in registers on the basis of choice of operation. Then using adder, another value will be reserved in the register which will be passed through the comparator to convert that value in the suitable range. Finally, 256-bit modular addition or subtraction result will be achieved.

The architecture displayed in Fig. 2 represents normal multiplication that performs like the following description. At first, a specific bit of multiplier will be picked in accordance with the value of the counter. Then, on the basis of selected bit, a value will be loaded in the appropriate position of *partial-prod* register with the help of *zero-container* register and *mcand-reg* register. After this, an addition operation takes place, the value of which will be stored in another register. This addition will be continued until it reaches the last bit. At last, the 512-bit multiplication result will be collected after finishing this loop.

Booth Radix-2 multiplication is expressed in the architecture presented in Fig. 3. Initially, the value of multiplicand will be stocked in register *A* and negative value of multiplicand and the value of multiplier will be stored in register *S* and register *P* respectively with zeros. Next, an operation will be carried out according to the least two significant bits of *P* register. After that, the value will be gathered in *Ptemp* register which will again be sent to *P* register after executing a right shift operation. This process will be continued until the counter reaches its final value. Finally, 256-bit X 256-bit result will be collected at the end of the loop.

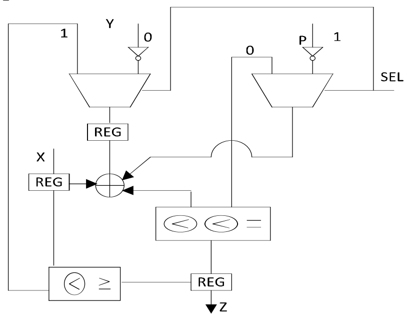

Fig. 1: Hardware Architecture of combined Modular addition and subtraction

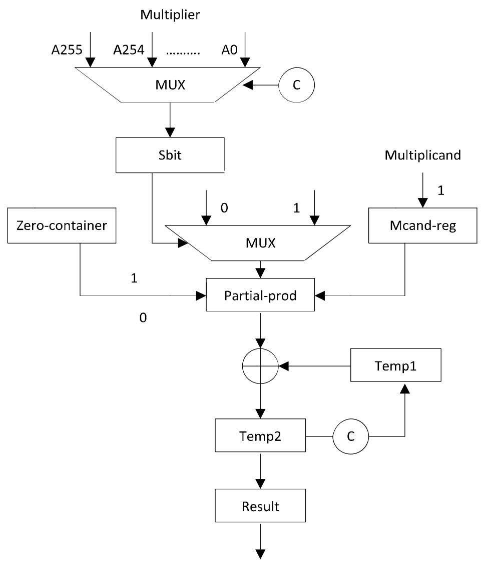

Fig. 2: Hardware Architecture of Normal Multiplication

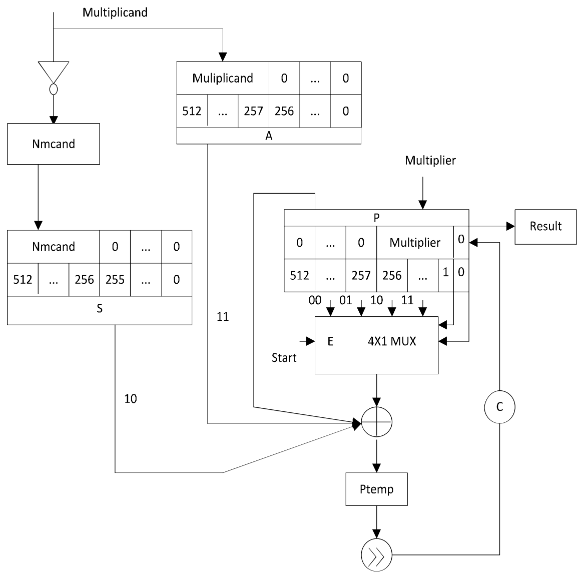

Fig. 3: Hardware Architecture of Booth Radix-2 Multiplication

The architecture exhibited in Fig. 4 depicts Booth Radix-4 multiplication. In the beginning, the values are stored in *state-reg*, *prod-reg*, *result-reg* and *Q-reg* in accordance with the value of reset. Then, the value will be updated in *state-next*, *prod-next* and *result-next*. Next, the value of multiplicand and multiplier will be taken in the *mcand-reg* and *prod-reg* register respectively during the period of the IDLE state. After that, one operation will be performed throughout the BUSY state on the basis of the least three significant bits of *result-next* register using an 8X1 multiplexer. This will be continued until the counter reaches the final value. At last, the 512-bit result will be collected.

The architecture appeared in Fig. 5 symbolizes Moore multiplication. The activity of this architecture is initialized with the accumulation of values of multiplicand and multiplier in *Q* and *M* registers respectively. The value of multiplier will also be taken along with zeros in another register *AQ*. According to the value of the least significant bit of *AQ* register, one operation will be performed and the value will be collected in the *accumulator* register. The value of *AQ* will also be right shifted at the same time. These operations will be continued until the counter reaches its final value. In the end, the 512-bit result will be found.

The architecture demonstrated in Fig. 6 delineates modular reduction operation. The functioning of this architecture starts with the creation of nine values according to the fast reduction modulo. Then, these values will be added with the help of left-shifter, not gate and adder to fulfill the required addition operation. Next, this result will be added with six pre-defined values. After that, according to the values of addition, proper bits will be chosen using multiplexers. Finally, 256-bit result will be collected.

The overall approach of our modular multiplication is presented in Fig. 7 where Booth Radix-4 multiplication unit is chosen for the multiplication operation. The selection is done by analyzing the performance and comparing with other models of multiplications which evidence is shown in section IV. The output of the multiplication module is 512-bit that is combined with the modular reduction module to achieve 256-bit output.

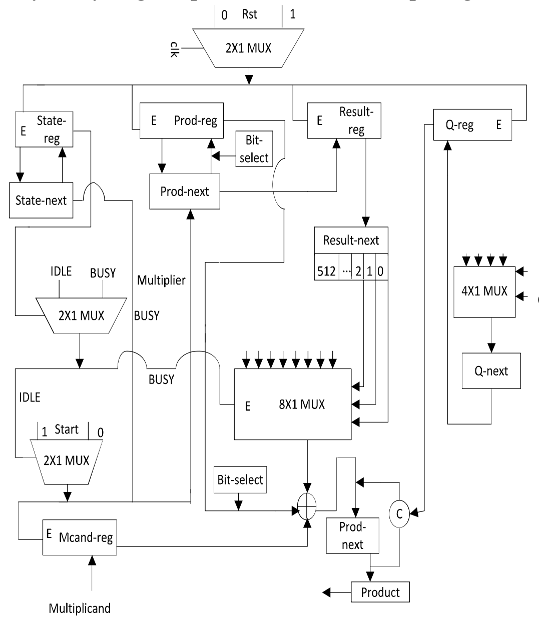

Fig. 4: Hardware Architecture of Booth Radix-4 Multiplication

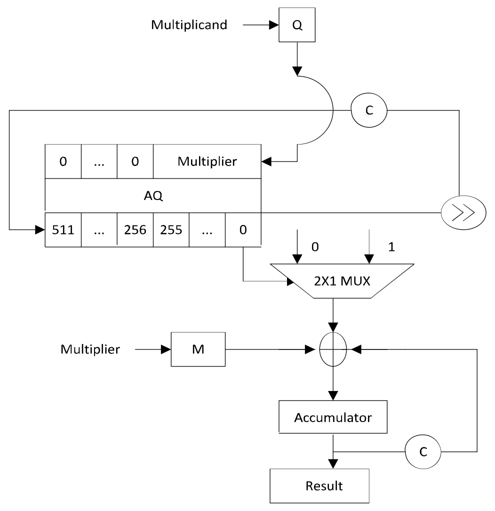

Fig. 5: Hardware Architecture of Moore Multiplication

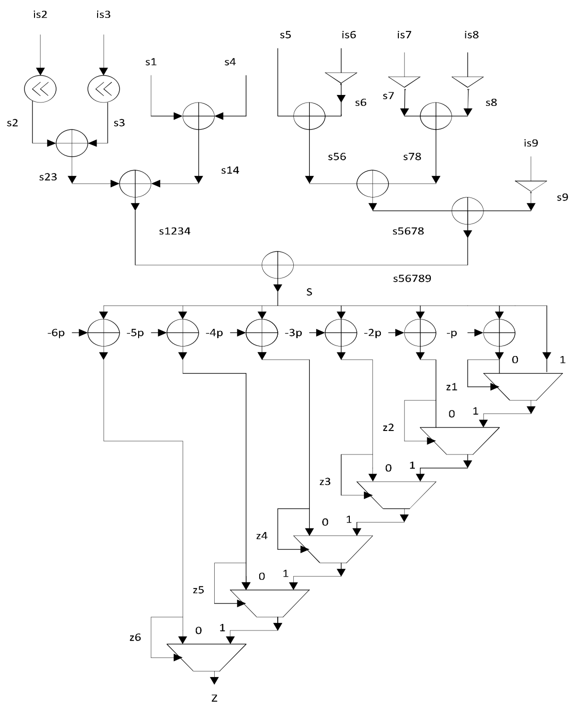

Fig. 6: Hardware Architecture of Modular Reduction Operation

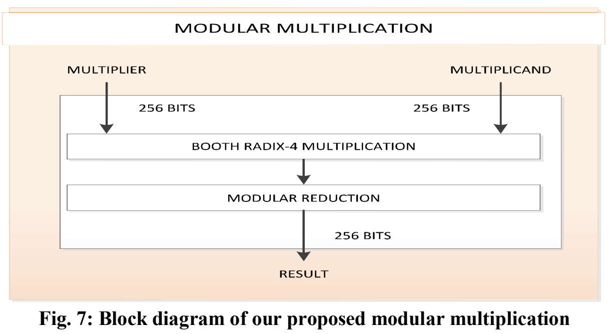

Fig. 7: Block diagram of our proposed modular multiplication architecture

IV. RESULTS AND PERFORMANCE COMPARISON

This part of this research deals with the hardware implementation results for our proposed modular arithmetic architectures over GF(*256*), on Virtex-5 (xc5vlx50t-2ff1136) FPGA technology. The designs are synthesized using Xilinx ISE 14.5 synthesis technology with the aim of high throughput rate and low resources. Both Modelsim and Isim are used for simulation purpose and the verification of these results are performed using high-level Maple software. Here, the comparison has been made in two stages- at first various architectures of us have been contrasted with one another to choose the best among our architectures. Then, another comparison has been made with related works in this field to signify the efficiency of our architecture.

TABLE I depicts the logic utilization summary for the proposed architecture of combined modular addition and subtraction module. From implementation results, it takes only 0.575ns in Virtex-5 FPGA for a prime field of 256-bit. This table is showing that more than 100% of bonded IOBs are used. But it can easily be improved if we send bits serially in real time implementation.

Our proposed modular multiplication architecture is composed of two modules: one module performs the operation of multiplication where the other module executes the modular operation. The module of the multiplication operation is based on several multiplication algorithms and modular operation is based on the fast reduction modulo algorithm. We have used all multiplication architectures with modular reduction module. These architectures have been implemented on Virtex-5 FPGA.

TABLE II presents the logic utilization summary of modular multiplication which is based on the normal multiplication algorithm and fast reduction modulo algorithm. Circuit latency of this design is (15.832ns X 257) = 4.07 µs in Virtex-5 FPGA over GF(*256*).

Logic utilization summary of proposed modular multiplication architecture is mentioned in TABLE III. It is based on the Booth Radix-2 multiplication and fast reduction modulo algorithm where the implementation platform is Virtex-5 FPGA. Circuit latency of this design is (15.609ns X 257) = 4.01 µs in Vertex-5 FPGA.

TABLE IV enlists the logic utilization summary observed during the performance analysis of modular multiplication operation which is based on the Booth Radix-4 multiplication algorithm and fast reduction modulo algorithm. Total clock cycle required for multiplication is 128 because Booth Radix-4 multiplication algorithm takes two bits at a time. Thus, the total time needed for modular multiplication operation is (15.832ns X 129) = 2.04 µs in Virtex-5 FPGA over GF(*256*).

TABLE V contains the logic utilization summary of the proposed architecture of modular multiplication designed on the basis of the Moore model of multiplication and fast reduction modulo algorithm. Circuit latency of this design is (15.606ns X 257) = 4.01 µs in Virtex-5 FPGA over GF (*256*).

TABLE I
Implementation result of combined Modular Addition and Subtraction

Logic utilization	Used	Available	Utilization
No. of slice LUTs	1229	41000	4%
No. of fully used LUT-FF pairs	0	1929	0%
No. of bonded IOBs	771	300	257%
No. of BUFG/BUFGCTRL/BUFHCEs	1	128	0%

TABLE II
Implementation result of Modular Multiplication using Normal Multiplication

Logic utilization	Used	Available	Utilization
No. of slice registers	2055	28800	7%
No. of slice LUTs	4646	28800	16%
No. of fully used LUT-FF pairs	1033	5668	18%
No. of bonded IOBs	517	480	107%
No. of BUFG/BUFGCTRLs	1	32	3%

TABLE III
Implementation result of Modular Multiplication using Booth Radix-2 Multiplication

Logic utilization	Used	Available	Utilization
No. of slice registers	2046	28800	7%
No. of slice LUTs	4377	28800	15%
No. of fully used LUT-FF pairs	1087	5336	20%
No. of bonded IOBs	770	480	160%
No. of BUFG/BUFGCTRLs	1	32	3%

TABLE IV
Implementation result of Modular Multiplication using Booth Radix-4 Multiplication

Logic utilization	Used	Available	Utilization
No. of slice registers	1290	28800	4%
No. of slice LUTs	4915	28800	17%
No. of fully used LUT-FF pairs	584	5621	10%
No. of bonded IOBs	771	480	160%
No. of BUFG/BUFGCTRLs	2	32	6%

TABLE V
Implementation result of Modular Multiplication using Moore Multiplication

Logic utilization	Used	Available	Utilization
No. of slice registers	1280	28800	4%
No. of slice LUTs	3357	28800	11%
No. of fully used LUT-FF pairs	363	4274	8%
No. of bonded IOBs	514	480	107%
No. of BUFG/BUFGCTRLs	1	32	3%

TABLE VI
Overall comparison among our modular multiplication architectures

Multiplication approach	Area utilization		Time utilization(µs)
	Slices	LUTs, FFs	
Booth Radix-4	1290(4%)	4915(17%), 584(10%)	2.04
Booth Radix-2	2046(7%)	4377(15%), 1087(20%)	4.01
Normal approach	2055(7%)	4646(16%), 1033(18%)	4.07
Moore approach	1280(4%)	3357(11%), 363(8%)	4.01

TABLE VII
Comparative analysis of modular multiplication between our implemented design and other related work over GF(*256*)

Ref.	Technology	Slices	Freq.(MHz)	Time(µs)
Ours	Virtex-V	1290	63.16	2.04
Lee [10]	Virtex-II	4843	37	3.46
Ghosh [11]	Virtex-II	5379	34	7.53
Duan [12]	Virtex-II	2607	194.56	4.06
Fan [13]	Virtex-II	3873	93	2.3
Daly [14]	Virtex-II	5477	14	18.28

TABLE VI presents that Booth Radix-4 multiplication along with fast reduction modulo is by far the most efficient hardware implementation approach both in terms of optimized area and time having 1290(4%) slices, 4915(17%) LUTs, 584(10%) FFs and 2.03 µs delay.

Table VII shows comparisons with the relevant modular multiplier. Implementation result of radix-4 Montgomery modular multiplication unit given by Lee [10] offers 4843 slices and 3.46 µs computational time. The design implemented by Ghosh [11] requires 7.53 µs and 5379 slices to perform a 256-bit modular multiplication. Other architectures over GF(*256*) presented by [12], [13], [14] require 4.06 µs, 2.3 µs and 18.28 µs respectively. The proposed FPGA-based modular multiplication offers better performance both in terms of area and timing compared to other works which is highly optimized for 256-bit ECC processor.

V. CONCLUSIONS

A high-performance FPGA-based implementation of the modular arithmetic unit over the prime field GF(p) is developed which will pave the way of producing high-performance 256-bit ECC processor. Two novel architectures of modular multiplication and combined modular addition and subtraction are proposed aimed at reducing the area and decreasing the overall latency of modular arithmetic operations. The fastest design of our modular multiplication achieved takes 2.03 µs using only 1290 slices on a Xilinx Virtex-5 FPGA. The proposed architecture of combined modular addition and subtraction needs only 0.575ns for the 256-bit prime field on Xilinx Virtex-5 FPGA. In conclusion, the proposed design provides the best result both in terms of timing and area on the basis of overall performance and comparative analysis with various modular arithmetic units over the 256-bit prime field.

ACKNOWLEDGMENT

The authors would like to express their thanks to the Committee for Advanced Studies and Research (CASR), Khulna University of Engineering and Technology (KUET), Khulna-9203, Bangladesh

.

REFERENCES

[1] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[2] M. S. Hossain and Y. Kong,"FPGA-Based Efficient Modular Multiplication for Elliptic Curve Cryptography", *International Telecommunication Networks and Applications Conference (ITNAC)*, UNSW, Sydney, Australia, pp. 191-195, 18-20 November, 2015.

[3] M. S. Hossain, Y. Kong, E. Saeedi and N. C. Vayalil, "High-Performance Elliptic Curve Cryptography Processor over NIST Prime Field", *IET Computers & Digital Techniques* vol. 11, no. 1, pp. 33-42, 2016.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[5] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. CRYPTO 1985*, 1986, pp. 417–426.

[6] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, pp. 203–209, 1987.

[7] Deschamps, Jean-Pierre / Sutter, Gustavo D / Cantó, Enrique, *Guide to FPGA implementations of arithmetic functions* ISBN 978-94-007-2986-5 – Springer, 2012.

[8] Wai-Leong Pang, Kah-Yoong Chan, Sew-Kin Wong and Choon-Siang Tan, "VHDL Modeling of Booth Radix-4 Floating Point Multiplier for VLSI designer's library", *WSEAS transactions on systems,* Issue 12, Volume 12, December 2013.

[9] Jean-Pierre Deschamps, Jose Luis Imana and Gustavo D. Sutter, *Hardware implementation of finite-field arithmetic,* ISBN 978-0-0715-4581-5 – McGrawHill, March, 2009.

[10] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "Efficient power-analysisresistant dual-field elliptic curve cryptographic processor using heterogeneous dualprocessing-element architecture," *IEEE Trans. VLSI Syst.*, vol. 22, no. 1, pp. 49–61, Jan. 2014.

[11] S. Ghosh, D. Mukhopadhyay, and D. Roychowdhury, "Petrel: Power and timing attack resistant elliptic curve scalar multiplier based on programmable GF(p) arithmetic unit," *IEEE Trans. Circuits Syst. I, Reg. Papers,*, vol. 58, no. 8, pp. 1798–1812, Aug. 2011.

[12] D. Cheng-hua, L. Yi, and C. Yong-tao, "A 3-stage pipelined large integer modular arithmetic unit for ecc," in *International Symposium on Information Engineering and Electronic Commerce (IEEC '09)*, May 2009, pp. 519–523.

[13] J. Fan, K. Sakiyama, and I. Verbauwhede, "Montgomery modular multiplication algorithm on multi-core systems," in *IEEE Workshop on Signal Processing Systems*,Oct. 2007, pp. 261–266.

[14] A. Daly, W. Marnane, T. Kerins, and E. Popovici, "An FPGA implementation of a GF(p) ALU for encryption processors," *Microprocessors and Microsystems*, vol. 28,no. 5–6, pp. 253–260, 2004, special Issue on FPGAs: Applications and Designs.

[15] Lu Zhou, Kuo-Hui Yeh, Gerhard Hancke, Zhe Liu, Chunhua Su, "Security and Privacy for the Industrial Internet of Things: An Overview of Approaches to Safeguarding Endpoints", Signal Processing Magazine IEEE, vol. 35, no. 5, pp. 76-87, 2018.