International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 11, No.1 (Mar-2022)

https://dx.doi.org/10.12785/ijcds/1101103

Compact and High Speed Point Multiplication Architecture for Elliptic Curve Diffie-Hellman Algorithm on Reconfigurable Computing

Abiy T. Abebe[1], Yalem zewd N. Shiferaw[2] and P.G.V. Suresh Kumar[3]

[1]*School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa, Ethiopia*
[2]*School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa, Ethiopia*
[3]*Ethiopian Technical University, Addis Ababa, Ethiopia*

Received 18 Mar. 2021, Revised 22 Dec. 2021, Accepted 12 Jan. 2022, Published 31 Mar. 2022

Abstract: Elliptic curve cryptography is popular for its efficiency and strong security as it provides equivalent security strength using smaller key sizes compared to other public key algorithms such as RSA that commonly use larger key sizes for the same level of security. Point multiplication is the core of elliptic curve cryptography. The development of reconfigurable devices enables researchers to exploit effective methods for implementation of efficient hardware based scalar multiplication. Modern FPGAs consist embedded hard-cores useful for design flexibility in addition to traditional generic fabrics. For cryptosystems implementation, several researchers used traditional logic elements, and only some have used embedded hard-cores including DSP slices and block RAMs. However complex cryptographic algorithms require large amount of generic logic and that in turn effect performance. Utilizing hard-cores entirely excludes flexibility of logic elements. Balanced utilization of these resources is considered in this research. Thus, for elliptic curve scalar multiplication required for implementing Elliptic Curve Diffie-Hellman algorithm, the FPGAs' hard-cores are used; while, simpler arithmetic and logical operations are flexibly implemented utilizing the generic FPGA fabrics. With this approach, a new architecture is proposed and implemented based on Montgomery algorithm with projective coordinates for point multiplication. Cascaded DSP48E1 slices are used with parallel-pipeline approach together with block RAMs for effective implementation. Compared to existing research outcomes reported in the literature, for implementation of the proposed architecture on Kintex-7 platform, smaller hardware resources (971 slices, 4BRAMs, and 32 DSP slices) are utilized with timing performance of 1.74 μs. Whereas, 1164 slices, 4 BRAMs, and 32 DSP slices are used with enhanced timing performance of 1.55 μs for implementation of the architecture on Virtex-7 platform.

Keywords: BRAMs, Cryptography, DSP slices, ECDH, embedded hard-cores, FPGA, Scalar Multiplication

1. Introduction

Key agreement protocols including Diffie-Hellman (DH) [1] and elliptic curve Diffie-Hellman (ECDH) algorithms enable senders and recipients of secret information to securely exchange key materials that can be used for secret key generations. The secret key can be generated without transferring it over insecure channels. Then, the independently computed and established key can be used for secret information exchange between the communicating parties.

ECDH algorithm is a public key method which is based on a technique combined from both the standard DH key agreement protocol and elliptic curve cryptography. The DH technique is based on discrete logarithm problem; whereas, elliptic curve cryptography (ECC) is constructed based on algebraic method of elliptic curves using finite fields. Therefore, ECDH gets its security strength from the computational hardness of both of these combined techniques. Scalar multiplication is considered as the core of elliptic curve cryptography [2].

In this paper, efficient architecture for EC point multiplication based on FPGA is proposed which is to be used for ECDH. The proposed method uses Montgomery algorithm to speed up the arithmetic processes for elliptic curve point multiplication in projective coordinates based on binary fields. Also, the modern FPGAs' dedicated hard-cores such as Digital Signal Processing (DSP) slices and block Random Access Memories (BRAMs) are used together to balance the speed of operation and the hardware resource utilization.

Most of existing FPGA based similar works focused on the security issues of the traditional Internet based information technology applications targeting mostly on high speed optimization.

E-mail address: abiy.tadesse@aait.edu.et, yalemzewdn@aait.edu.et, pendemsuresh@gmail.com

But, in the current IoT network paradigm, there are different platforms involved including high performance computing and constrained devices, each with different performance, resource, and security requirements. Direct mapping of existing cryptographic algorithms to secure the current IoT network may lead to inefficient performance and unacceptable security. Therefore, efficient hardware based cryptographic architectures with small area and high speed optimization targets tailored to constrained small sized IoT devices and high speed applications, respectively, are ongoing research since the size of constrained devices is getting smaller and smaller; and also, the speed of high performance platforms is increasing.

Thus, the compact and efficient architecture proposed in this work is intended to address these issues. More precisely, small footprint security is becoming increasing area of interest for researchers in both academia and industry. Thus, the small architecture proposed in this research enables to address security issues in IoT devices. Furthermore, the enhanced speed of performance of the proposed method is intended to balance area and speed trade-offs so as to meet the requirement of the IoT network security.

Existing methods for FPGA based ECC architectures also used either the generic FPGA fabrics entirely, or dedicated hard-cores completely, focusing on high throughput architectures or compact architectures. However, utilizing balanced amounts of both of these resources for suitable parts of the algorithm are important since complete use of the generic FPGA fabrics for complex cryptographic algorithms reduces performance, and requires large area. Similarly, though using only dedicated hard-cores may be helpful to achieve high throughput, but, the implementation is not flexible; and, if pipelining and cascading of the DSP slices are needed, it requires large amount of space.

Therefore, balancing utilization of both resources is useful depending on the intensity of the arithmetic computations in such a way that performing simple arithmetic and logical operations using the generic FPGA fabric; while, using dedicated embedded hardware resources for intensive arithmetic operations.

The contributions of this research work are described as follows:
In this research, a compact and efficient architecture for EC point multiplication is proposed that can be used for ECDH key exchange algorithm tailored to the security issues of constrained IoT devices. Parallel-pipelined approach is followed using cascaded DSP48E1 with BRAMs; and, Montgomery method is used to implement the architecture on 7-series FPGA platforms for binary field NIST curves based on projective coordinates system. Balanced utilization of the embedded hard-cores including DSP48E1 slices and BRAMs, and generic FPGA fabrics are used together to implement the proposed architecture. This work exploited the balanced use of these resources for reduced area and balanced speed of performance. Thus, for complex arithmetic operations, DSP slices are used; and, for simple logical, arithmetic, and control operations, the traditional FPGA fabrics are used. The proposed approach reduced the performance bottlenecks of the EC scalar multiplication that is used for ECDH algorithm based on FPGA. The target of optimization is small resource utilization with good performance considering the security of constrained IoT devices in IoT network. Moreover, the high speed performance is also useful for high performance platforms in IoT network. The proposed architecture could address both of these requirements. Therefore, smaller hardware resources are used and good performance metrics have been achieved. The main contributions of this research can be summarized as:

- Effective, compact, and high speed architecture suitable for elliptic curve point multiplication that can be used in ECDH algorithm is proposed based on FPGA hard-cores (DSP48E1 slices and BRAMs) together with balanced utilization of generic FPGA fabrics. Simple arithmetic and logical operations as well as control operations are done using generic FPGA fabrics; while, dedicated embedded hardware is used for intensive arithmetic operations.
- Parallel-pipelined method based on cascaded DSP slices and pipeline-able registers is employed. With this approach, the research exploited the modern FPGA resources, and balanced area and speed trade-offs for efficient performance, achieving lightweight and high speed architecture.
- Montgomery method with left-to-right approach is used for the parallel-pipelined based EC scalar multiplication architecture based on projective co-ordinates system for NIST binary curves so as to speed up the computations and reduce performance bottlenecks.

The organization of this paper is outlined as follows: Section 2 introduces ECC, ECDH, and Montgomery algorithms as background. In Section 3, related works are discussed followed by the proposed architecture which is presented in Section 4. Then, in Section 5, implementation approaches and comparisons of the achieved results against existing research outcomes found in the literature are followed. The paper is concluded in Section 6.

2. Background

A. *Elliptic Curve Cryptosystem*

ECC is a public key algorithm designed using algebraic constructs of elliptic curves based on finite fields [2].

Several researchers have stated and shown that Elliptic Curve Cryptography with smaller key length can provide equivalent security strength compared to RSA with longer key size [2], [3], [4], [5], [6].

ECC is based on points on carefully selected elliptic

curves. The points can be denoted as x and y coordinates and are obtained by using suitable arithmetic field operations. Then, they can be used to represent secret messages for performing various cryptographic processes [2].

For cryptographic applications, elliptic curves have been used based on prime $GF(p)$ or binary $GF(2^m)$ field operations [4], [5], [6]. In case of prime fields, the prime number $p > 3$ is used, and computations are performed over the set of $0, ..., p-1$ modulo p. But, for binary fields, field operations are performed over $GF(2^m)$ and all the computations are done in $GF(2^m)$. For hardware implementations of elliptic curve cryptosystem, elliptic curves based on binary field representation are preferred since processing of binary elements in hardware can efficiently and easily be performed [7]. Elliptic Curves that are based on prime fields comprised of points x and y are shown in Eq. 1 [2]:

$$y^2 = x^3 + ax + b \ mod \ p \tag{1}$$

, where, x and $y \in GF(p)$, and $(a, b \in GF(p))$ are selected from the prime field $GF(p)$ such that $4a^3 + 27b^2 \neq 0 \ mod \ p$, $p \neq 0$ to satisfy non-singular elliptic curve and avoid repeated factors. All points satisfying the elliptic curve equation are considered as sets of solutions and points on the curve. Point at infinity, defined by O is considered as additive identity for additive group operation on the elliptic curve. If an elliptic curve is defined over a prime field and represented by E, then, E is an elliptic curve over $G(p)$. So, there can be finitely many points on E that can be computed using different techniques. The number of points that can be obtained in the curve $E(G(p))$, which can be denoted by $\#E(G(p))$ is known as the order of the elliptic curve E over $G(p)$ [2].

1) Elliptic curve point addition over $GF(p)$

If $P(x_1, y_1)$ and $Q(x_2, y_2)$ are points on the curve, then another point denoted by $R(x_3, y_3))$ can be found on the same curve by performing addition operation on P and Q provided $P \neq O$ and $P \neq \pm Q$, as $R(x_3, y_3) = P + Q$ by processing the following steps [2]:

let $\ell = \frac{(y_2 - y_1)}{(x_2 - x_1)} \ mod \ p$, then
$x_3 = \ell^2 - x_1 - x_2) \ mod \ p$
$y_3 = ((x_1 - x_3) \cdot \ell - y_1 \ mod \ p)$

2)Elliptic curve point doubling over $GF(p)$

Point doubling can also be expressed algebraically as follows:

If $y_1 \neq 0$, then
$R = (x_3, y_3) = 2P$
$\ell = (3 \cdot x_1^2 + a)/(2 \cdot y_1) \ mod \ p$
$x_3 = (\ell^2 - 2x_1) \ mod \ p$
$y_3 = ((x_1 - x_3) \cdot \ell - y_1) \ mod \ p$

Elliptic curves defined over binary fields $GF(2^m)$ can also be specified as shown in Eq. 2 [2]:

$$y^2 + xy \ mod \ p = x3 + ax^2 + b \ mod \ p \tag{2}$$

, where $a, b \in GF(2^m)$ and $b \neq 0$. Points in binary field which satisfy the elliptic curve equation (Eq. 2) can be obtained with similar approaches used for point addition and doubling processes of prime field $GF(p)$. Point addition over binary fields can be done as follows:

if $R = (x_3, y_3) = P + Q, P \neq -Q$, then
$\ell = \frac{(y_2 + y_1)}{(x_2 + x_1)}$
$x_3 = \ell^2 + \ell + x_1 + x_2$
$y_3 = y_1 + x_3 + \ell(x_1 + x_3)$

Similarly, point doubling can be performed as follows:
If $x_1 = 0$ then $P + P = 2P = O$
If $x_1 \neq 0$, then
Take
$R = (x_3, y_3 = 2P)$, then
$\ell = \frac{x_1 + y_1}{x_1}$
$x_3 = \ell^2 + \ell + a$
$y_3 = x_1 + (\ell + 1) \cdot x_3$

The process of computation of points on the curve continues until the required number of points are obtained by performing such procedures.

The security strength of ECC generally relies on the computational hardness of Elliptic Curve Discrete Logarithm Problem (ECDLP) [2]. If an elliptic curve is defined over $GF(p)$ or $GF(2^m)$, and P and Q are points on the curve having order r, then, by selecting a scalar $k \in [1, r-1]$, computation of a scalar multiplication $Q = kP$ is the core of elliptic curve cryptography. It is a discrete logarithm problem to be solved, but considered to be hard mathematical problem [7].

By repeated point addition and doubling steps, the elliptic curve scalar multiplication can be performed since the equation $Q = kP$ can be computed by adding the point P k-times as:

$$k \cdot P = \underbrace{P + P + \cdots + P}_{\text{k terms}}$$

And, this repeated addition of point P can be expressed based on point additions and point doublings as $kP = P + \ldots (2(2(\ldots P + 2(P + 2(P + 2P)))$. For example, if the selected value of $k = 31$, then $kP = 31P = P + 2(P + 2(P + 2(P + 2P))$, based on both point addition and doubling processes.

B. Elliptic Curve Double and Add Algorithms

There are different algorithms that have been proposed by various researchers for efficiently performing the elliptic curve scalar multiplications [8], [9], [10], [11], [12]. Double and Add algorithm is the commonly used algorithm for point multiplication in elliptic curve cryptography [7]. There are two forms of algorithms in this regard. The first one is right-to-left approach and the other one is left-to-right approach. The left-to-right approach [6] is presented in Algorithm 1, and the right- to-left method is presented in Algorithm 2.

Algorithm 1: Point Addition and Point Doubling (Left-to-right approach)

Input : $k = (k_{m-1}, k_{m-2}, \cdots, k_1, k_0)_2,\ P \in E(GF(2^m))$
Ensure: $Q = kP$
1: $Q = P$;
2: for $i = m-2$ downto 0 do
3: $Q = 2Q$
4: if $k_i = 1$ then
5: $Q = Q + P$;
6: end if
7: end for
8: Return Q

Algorithm 2: Point Addition and Point Doubling (Right-to-left approach)

Input: $k = (k_{m-1}, k_{m-2}, \cdots, k_1, k_0)_2,\ P \in E(GF(2^m))$
Ensure: $Q = kP$
1: $Q = 1; R = P$
2: for $i = 0$ to $m-1$ do
3: if $k_i = 1$ then
4: $Q = Q + R$
5: end if
6: $R = 2R$;
7: end for
8: Return Q

The left-to-right approach allows scanning each k_i bits starting at the most significant bit (MSB) down to its list significant bit (LSB). Then, it is used to perform point doubling operation whenever the k_i bit are zero (0). But, when the k_i bits are one (1), it performs point addition operation as presented in Algorithm 1. Similarly, the right-to-left method allows to scan each k_i bits starting at the LSB up to its MSB. Then, point addition could be done whenever the k_i bit are one (1). But, when the ki bits are zero (0), it performs point doubling operation as shown in Algorithm 2.

The representations of the double and add steps in both Algorithm 1 and Algorithm 2 have followed the affine x and y coordinates system of an elliptic curve. Many inversion processes are needed to perform scalar multiplication in affine coordinates [12]. The inversion - operations in GF arithmetic, generally, are slow and expensive processes which can lead to low performance of the cryptosystem. Therefore, researchers proposed projective coordinates system which can transform the affine system to another co-ordinate system that can avoid the inversion operations while providing better performance [2], [7], [13]. The standard way of transforming affine coordinates to projective is done by representing the elliptic curve point P by three notations: X, Y, and Z, and then expressing x and y affine co-ordinates as $x = \frac{X}{Z}$ and $y = \frac{Y}{Z}$, where $Z \neq 0$. Thus, the elliptic curve binary field equation will be changed as in Eq. 3 [2], [7], [10], [13]:

$$Y^2Z + XYZ = X^3 + aX^2Z + bZ^3. \tag{3}$$

After completing the scalar multiplication (point addition and point doubling) operations in projective system, then point P must be re-converted back into affine co-ordinates system [7], [13].

C. Specification of Elliptic Curve Diffie-Hellman Algorithm

ECDH key exchange algorithm [2], [14] differs from the DH key exchange protocol [1] in that the former is based on scalar multiplication of elliptic curve points for generation of secret keys using discrete logarithm problem; whereas, the latter uses exponentiation based on number in prime field. When two communicating parties (a sender and recipient) use ECDH for key exchange, they first agree on the elliptic curve and related parameters including the base point [2], [14]. Then, the two communicating parties perform the following activities to calculate the shared secret:

1) The sender selects a random number k_s and the recipient also selects a random number k_r.

2) The sender performs computation: $Q_s = k_s \bullet P$, and, the recipient also computes $Q_r = k_r \bullet P$.

3) The sender and the recipient exchange their computed public keys (Q_s and Q_r).

4) After receiving the public key of the other end, then, the sender and the recipient compute the shared secret independently. The sender computes the shared secret as: $(SS_s) = Q_r \bullet k_s = (k_r \bullet P \cdot ks)$. Similarly, the recipient computes the same value as: $(SSr) = Qs \bullet kr = (k_s \bullet P \cdot kr)$. Since SSs and SSr are equal, they can use it as a key material.

D. The Montgomery Technique for Efficient Scalar Multiplication

If there are two points on an elliptic curve denoted by P and Q such that: $P(x_1, y_1)$ and $Q(x_2, y_2)$, then the third and fourth points on the same curve can be obtained based on the curve equation shown in Eq. 3 as $(x_3, y_3) = P + Q$ and $(x_4, y_4) = P - Q$ on the same curve [2], [7], [10], [13]. Therefore, x_3 can be computed by Eq. 4:

$$x_3 = x_4 + \frac{x_1}{x_1 + x_2} + \left(\frac{x_1}{x_1 + x_2}\right)^2 \quad (4)$$

As it is observed in Eq. 4, only x-coordinate of the points P, and Q, and also the point $P - Q$ are used to find the x-coordinate of $P + Q$. If the x-coordinate representation of point P in the projective coordinates system is expressed as $x = \frac{X}{Z}$, then points X_2 and Z_2 in the projective coordinates system (if $(X_2, -, Z_2)$) will be given by Eq. 5 [2], [7], [10], [13].

$$X_2 = X^4 + bZ^4; \; Z_2 = X^2.Z^2 \quad (5)$$

Similarly, point addition can be done as: $Z_3 = [(X_1 \bullet Z_2 + X_2 \bullet Z_1)]^2$; $X_3 = x \bullet Z_3 + (X_1 \bullet Z_2) \bullet (X_2 \bullet Z_1)$.

To implement scalar multiplication, one of the efficient methods for point addition and doubling operations based on projective co-ordinate system is the Montgomery algorithm [2], [7], [15]. The Montgomery algorithm [15] for point additions and point doublings [2], [4] are presented in Algorithms 3 ($Mdouble()$), and Algorithm 4 ($Madd()$), respectively.

Algorithm 3: Montgomery Point Doubling method $Mdouble(X_1, Z_1)$

Input: $P = (X_1, -, Z_1) \in E(GF(2^m))$, c such that $c^2 = b$
Ensure: $P = 2P$
1: $T = X_1^2$
2: $M = c \cdot Z_1^2$
3: $Z_2 = T \cdot Z_1^2$
4: $M = M^2$
5: $T = T^2$
6: $X_2 = T + M$
7: Return (X_2, Z_2)

Similarly, point addition can be expressed as:
$Z_3 = (X_1 \cdot Z_2 + X_2 \cdot Z_1)^2$
$X_3 = x \cdot Z_3 + (X_1 \cdot Z_2) \cdot (X_2 \cdot Z_1)$

Algorithm 4: Montgomery Method for Point Addition $Madd(X_1, Z_1, X_2, Z_2)$

Input: $P = (X_1, -, Z_1), Q = (X_2, -, Z_2) \in E(GF(2^m)$
Ensure: $P = P + Q$
1: $M = (X_1 \cdot Z_2) + (Z_1 \cdot X_2)$
2: $Z_3 = M^2$
3: $N = (X_1 \cdot Z_2) \cdot (Z_1 \cdot X_2)$
4: $M = x \cdot Z_3$
5: $X_3 = M + N$
6: Return (X_3, Z_3)

Based on Algorithms 3 and 4, Montgomery point multiplication algorithm has been constructed as presented by Algorithm 5. This algorithm composes both $Mdouble()$ (Algorithm 3) and $Madd()$ (Algorithm 4) methods and, it can be used to compute them concurrently [2], [4].

Algorithm 5: Montgomery Point Multiplication based on both Point Additions and Point Doublings

Require: $k = (k_{n-1}, k_{n-2}, \cdots, k_1, k_0)_2$ with $k_{n-1} = 1, P \in E(GF(2^m)$
Ensure: $Q = kP$
1: $X_1 = x, Z_1 = 1$
2: $X_2 = x^4 + b, Z_2 = x^2$
3: for $i = n - 2$ downto 0 do
4: if $k_i = 1$ then
5: $Madd(X_1, Z_1, X_2, Z_2)$
6: $Mdouble(X_2, Z_2)$
7: else
8: $Madd(X_2, Z_2, X_1, Z_1)$
9: $Mdouble(X_1, Z_1)$
10: end if
11: end for
12: $x_3 = \frac{X_1}{Z_1}$
13: $y_3 = (x + \frac{X_1}{Z_1})[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)](xZ_1Z_2)^{-1} + y$
14: Return (x_3, y_3)

In this algorithm, $Mdouble()$ and $Madd()$ are the operations presented in *Algorithm* 4.4 and *Algorithm* 4.5 that are executed at every iteration of the execution of the algorithm. Thus, when k_i is one (1), then the multiplication for $Madd(X_1, Z_1, X_2, Z_2)$ and $Mdouble(X_2, Z_2)$ can be done. Otherwise, (when the bits are zero (0), multiplications for $Madd(X_2, Z_2, X_1, Z_1)$ and $Mdouble(X_1, Z_1)$ will be done.

Now, conversion from standard projective coordinates back to affine coordinates is required. Therefore, for the x- coordinate, the conversion is done as $x_3 = \frac{X_1}{Z_1}$ as shown in *Algorithm* 5 (number 12). Similarly, the conversion to the affine y-coordinate can be done as $y_3 = (x + \frac{X_1}{Z_1})[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1Z_2)](xZ_1Z_2)^{-1} + y$, which is shown in *Algorithm* 5 (number 13). In this case, conversions to affine coordinates in both x_3 and y_3 have one step of inversion computation, though they are at the

final stage of the algorithm.

To further minimize number of inversion steps, sequential reduction steps are provided in Algorithm 6 [2], [4] which contains only one inversion step.

Algorithm 6: Standard Projective to Affine Coordinates

Input: $P = (X_1, Z_1), Q = (X_2, Z_2), P(x, y) \in E(GF(2^m))$
Ensure: (x_3, y_3)
1: $\ell_1 = Z_1 \times Z_2$
2: $\ell_2 = Z_1 \times x$
3: $\ell_3 = \ell_2 + X_1$
4: $\ell_4 = Z_2 \times x$
5: $\ell_5 = \ell_4 + X_1$
6: $\ell_6 = \ell_4 + \ell_2$
7: $\ell_7 = \ell_3 \times \ell_6$
8: $\ell_8 = x^2 + y$
9: $\ell_9 = \ell_1 \times \ell_8$
10: $\ell_{10} = \ell_7 + \ell_9$
11: $\ell_{11} = x \times \ell_1$
12: $\ell_{12} = inverse(\ell_{11})$
13: $\ell_{13} = \ell_{12} \times \ell_{10}$
14: $x_3 = \ell_{14} = \ell_5 \times \ell_{12}$
15: $\ell_{15} = \ell_{14} + x$
16: $\ell_{16} = \ell_{15} \times \ell_{13}$
17: $y_3 = \ell_{16} + y$
18: Return (x_3, y_3)

The single inversion step can be performed using inversion algorithms such as Fermat's Little Theorem [2], or Extended Euclidean Algorithm [12], or by method of precomputation. Based on the sequence of conversion steps in Algorithm 6, only one inversion operation is needed.

Parallel operations are possible on the steps of Montgomery algorithms. For example, Montgomery point doubling operation, $2(X_1 : - : Z_1) = (X_2 : - : Z_2)$ can be executed in a single clock cycle (CC) as follows:
$CC_1 = T = X_1^2; M = c.Z_1^2; Z_2 = T.Z_1^2; CC_1' = X_2 = T^2 + M^2$
Similarly, Montgomery point addition operation $(X_1 : - : Z_1) = (X_1 : - : Z_1) + (X_2 : - : Z_2)$ can be executed just in two clock cycles: $CC_1 : t_1 = (X_1.Z_2); t_2 = (Z_1.X_2);$
$CC_1' : M = t_1 + t_2; Z_1 = M^2$
$CC_2 : N = t_1.t_2; M = x.Z_1$
$CC_2' = X_1 = M + N$

Thus, the computations of clock cycles CC_1' and CC_2' can be performed when CC_1 and CC_2 are executed respectively, due to their easiness. In this work, implementation of the binary field sizes recommended by NIST including 163, 233, and 283 have been considered.

3. Related Works

Most of the proposed works for ECC point multiplication have considered high speed processing [16], [17], [18], [19]. Only few existing works have considered small area architectures and architectures for balancing speed and area [13], [20], [21]. Moreover, the majority of existing methods used the common FPGA resources such as traditional logic elements for point multiplication [13], [21], [22], [23].

Basically, bit-parallel and digit-serial architectures have commonly been implemented as multipliers for high performance [24], [25], [26].

Thus, Karatsuba multiplier has been used as multiplier for bit-parallel architectures since it enabled to reduce resource utilization [27], [28]. However, since area reduction impacts performance speed, several pipelining stages were needed to increase performance at cost of latency.

For the trade-offs of higher performance speed with large area, digit-serial architectures were used based on full-precision multipliers. However, even though the Karatsuba multiplier has better complexity of multiplication than full-precision based digit serial approach, the latter has reduced number of critical logic levels [8], [29].

Very few research works demonstrated the use of DSP blocks for implementation of elliptic curve point multiplication [30], [31].
Summary of selected related works are presented in Table I.

The method proposed in the current work differs from the approaches followed in [30] and [31] that used DSPs and RAMs in many aspects. This work is not serial-to-parallel architecture for standard double and add algorithm implementation as the approach presented in [30], or it is not a sequential approach as used in [31]. The former used Montgomery method for standard double and add algorithm for serial-to-parallel architecture, and the latter used the Montgomery ladder with sequential approach using micro-code. They also used distributed RAMs for storage. Both [30] and [31] also used prime fields for their implementations.

In this work, implementation of the binary field sizes recommended by NIST including 163, 233, and 283 have been considered. The $GF(2^m)$ elements are structured in polynomial representation and the reduction polynomials used are:

$f(x) = x^{163} + x^7 + x^6 + x^3 + 1$; $f(x) = x^{233} + x^{74} + 1$; $f(x) = x^{283} + x^{12} + x^7 + x^5 + 1$; for 163, 233, and 283 field sizes, respectively.

4. The Proposed Architecture

To implement ECDH algorithm on FPGA, the core of elliptic curve cryptosystem that is, the scalar multiplication, as presented in the hierarchical structure shown in Fig. 1,

TABLE I. Summary of Related Works

Reference	Architecture	Method /Multiplier	Implementation Platform	Field	Optimization target	Other (Features)
[32]	Compact	Carry-Chain logic	FPGA -used generic logic	Prime	Small area	Countermeasure for SPA attack
[33]	Pipelining (Two stage pipelining)	-Karatsuba multiplier	FPGA -used generic logic	Binary	High speed	
[34]	-High performance (Two stage pipelining) -Low-latency	Quadratic full-precision -a single multiplier Quadratic full-precision Three multipliers	FPGA -used generic logic	Binary	High speed Low-latency But, consumed large resources	
[31]	Compact	Montgomery Ladder Algorithm -Sequential with micro-coding DSP slices were used for Arithmetic unit -Used distributed RAMs for local storages	FPGA	Prime	Small area Low-latency But, consumed large resources	Counter measures for Side channel and fault injection attack
[30]	Serial to parallel	Montgomery method for standard Double and Add algorithm -Used DSP48 and dual-port BRAMs	FPGA	Prime	High speed	

is considered. The Montgomery method that is based on projective co-ordinates system for scalar multiplication is used. Thus, Algorithm 5 which composes Algorithm 3 and Algorithm 4 is implemented. These algorithms are implemented in parallel for efficiency, while using $left-to-right$ approach for point addition and point doubling in order to balance the resource consumption. For conversion from projective back to affine coordinates system, Algorithm 6, which is the standard method is used. But, the inversion step is performed using pre-computation. Furthermore, pipelining method is employed in this work based on pipelining registers and the control logic. DSP slices are cascaded for the process of parallelization and to speed up the computation. In this work, binary field is considered instead of prime field.

Moreover, in the proposed method, inverses are pre-computed and stored in BRAMs. Since the approach used in [30] is intended for high speed processing, and the main purpose of the work in [31] is to produce compact architecture, the proposed architecture in the current work is used to balance speed and area trade-offs for reasonable resource utilization and good performance by using dedicated DSP slices and BRAMs for the complex computation part, and the traditional FPGA logic elements for flexible controlling and simpler logical and arithmetic operations.

DSP48E1 slices found in Xilinx Seven Series FPGA platforms [35] are used to implement the Montgomery

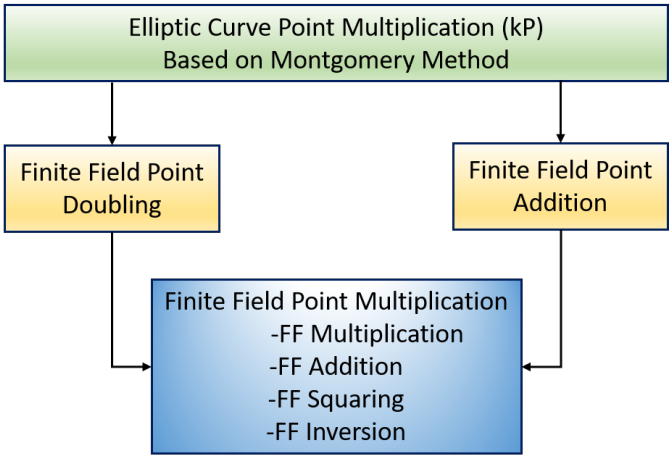

Figure 1. Hierarchy of Finite Field Point Multiplication

method for performing scalar multiplication based on both point doubling and point addition operations. The structure of the Montgomery method for point doubling with projective co-ordinates based on DSP48E1 construction is represented as shown in Fig. 2. In this case, to generate the X_2 and Z_2 outputs, only two DSP48E1 slices are required which can be processed in parallel. To produce Z_2, a DSP slice with only one multiplier component is used. Whereas,

to obtain the X_2 output, one DSP slice which consists of one multiplier and one adder is used.

But, based on the same approach, six DSP slices are needed to perform one point addition operation as shown in Fig. 3. Four of them require only a multiplier, and only two DSP slices use a multiplier together with an adder component. All these six DSP slices can be executed in parallel. The hardware required for implementation of the squaring operations shown in both Fig. 2 and Fig. 3 are not significant and can be negligible as the squaring operation over $GF(2^m)$ is in binary. The inverse steps found in the projective coordinate system are implemented by pre-computing and storing them in BRAMs.

A control block is used as shown in both Fig. 2 and Fig. 3. The control logic is used to effectively manage the activities of the major components of the implemented architecture including DSP slices, BRAMs, and registers; and, to synchronize the parallel operations of the replicated DSP slices as well as the pipeline registers. Operations including read and write activities and processes of registers and block RAMs are also synchronized based on the control logic. The control logic consists of signals that are used to manage the synchronization by activating and de-activating them for the working of the BRAMs, DSP slices, and the pipeline-able registers. It is implemented based on simple finite state machine (FSM) using the FPGA logic elements.

To implement these architectures for large number of point additions and doublings, the DSP slices are cascaded and then executed in parallel as shown in Fig. 4 and Fig. 5 for point doublings and point additions, respectively. The replicated structures shown in Fig. 4 and Fig. 5 that are constructed from the basic point addition and point doubling structures shown in Fig. 2 and Fig. 3, respectively, are useful for pipelining processes so as to increase the throughput. However, this works with area cost.

The clock cycles (CC_s) required to perform Montgomery point addition are two CC_s. Whereas for Montgomery point doubling, one CC is needed. To compute the total time consumed, the execution time required for point doubling and point addition processes and the number of bits of the corresponding binary curve are used. If the number of bits used for a selected curve is represented by m (where m can be 163, 233, or 283), and the total execution time achieved for implementation of both point doubling and point addition operations on the specific platform is represented by t, then: $m \cdot t$ is used to obtain the total consumed time, excluding the time required for inversion operation since inversion is pre-computed in this work. For example, if m=163, and the total execution time for both point additions and point doublings are t=100 ns, then, the total consumed time for the specified total clock cycles becomes 16300μs. In case of Kintex-7, for m=163, the total execution time is 10.75 ns, for m=233, the total execution time is 11.68 ns and, for m=283, the total execution time is 12.46 ns. In case of Virtex-7, for m=163, the total execution time is 9.5 ns.

5. Implementation Approaches and Result Comparisons

The compact and efficient architecture proposed in this research work is a new approach as it uses a parallel-pipelined method based on cascaded DSP48E1 slices with BRAMs in combination with traditional FPGA logic elements for scalar multiplication in binary field in projective coordinates system using Montgomery method so as to implement ECDH algorithm and speed up the process.

Existing similar works used either traditional FPGA generic logic only, or the FPGA hard-cores exclusively to implement their respective architectures for either compact or high speed architectures, respectively. This work exploits the balanced use of both of these resource types for reduced area and balanced speed of performance. For complex arithmetic operations DSP slices are used, and for simple logical and arithmetic operations and control operations, the traditional FPGA fabrics are used. The optimization target is small resource utilization with good performance considering the security of constrained IoT devices. Moreover, the high speed performance is also useful for high performance platforms in IoT network. The proposed architecture achieved both requirements.

The proposed architecture has been implemented on the modern Xilinx 7-series FPGA platforms including Kintex-7, and Virtex-7 devices. These modern platforms consist DSP48E1 hardcore which can flexibly perform complex arithmetic computations and logical processes. They are useful to construct pipelining architectures, and cascading of several DSP slices for parallel operation.

The Xilinx 7-series platforms support DSP blocks with 25×18 multiplier, and 48 bit for addition. However, since cascaded DSP slices have been used in this work so as to provide wider word widths, 16 bits for multiplication and 32 bits for addition have been used. Then, cascading has been applied from the same DSP column blocks for efficiency.

The BRAMs used are dual-port, and made to store pre-computed inverse values and input data for DSP slices. Since the 7-series FPGAs are used, Vivado HLS is used for verification and synthesis. This state-of-the-art tool provides different optimization directives to flexibly implement the required resources. The C-based pragmas enabled to implement the preferred resources as required. The default synthesis and implementation processes and technology mapping used by the synthesis tool (by default), are managed and changed based on the optimization targets using the HLS optimization directives available in Vivado HLS tool and the coding styles followed by using the HLS pragmas. Therefore, using this approach, specific hard-cores could be utilized such as DSPs and BRAMs, and it also enables to use even other different available resources as needed.

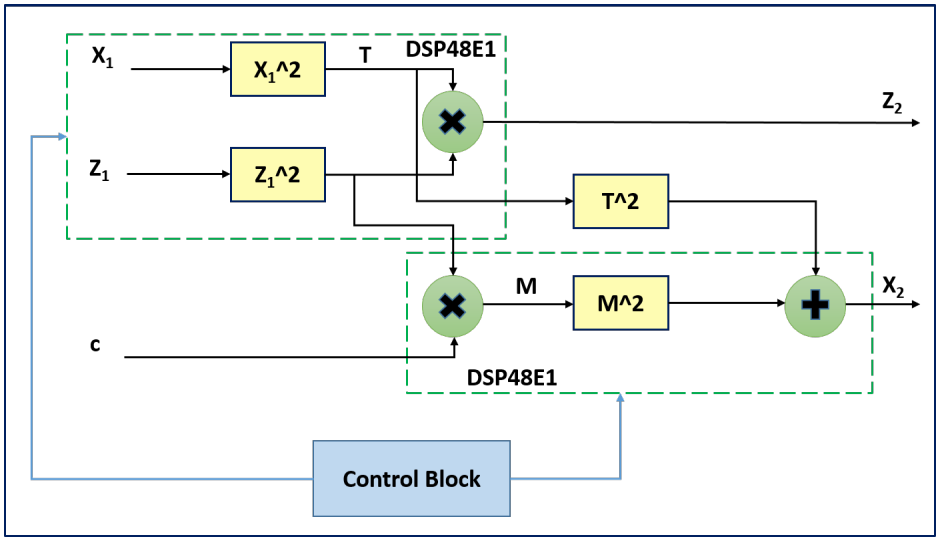

Figure 2. Single Cycle Point doubling using DSP slices based on Montgomery Method

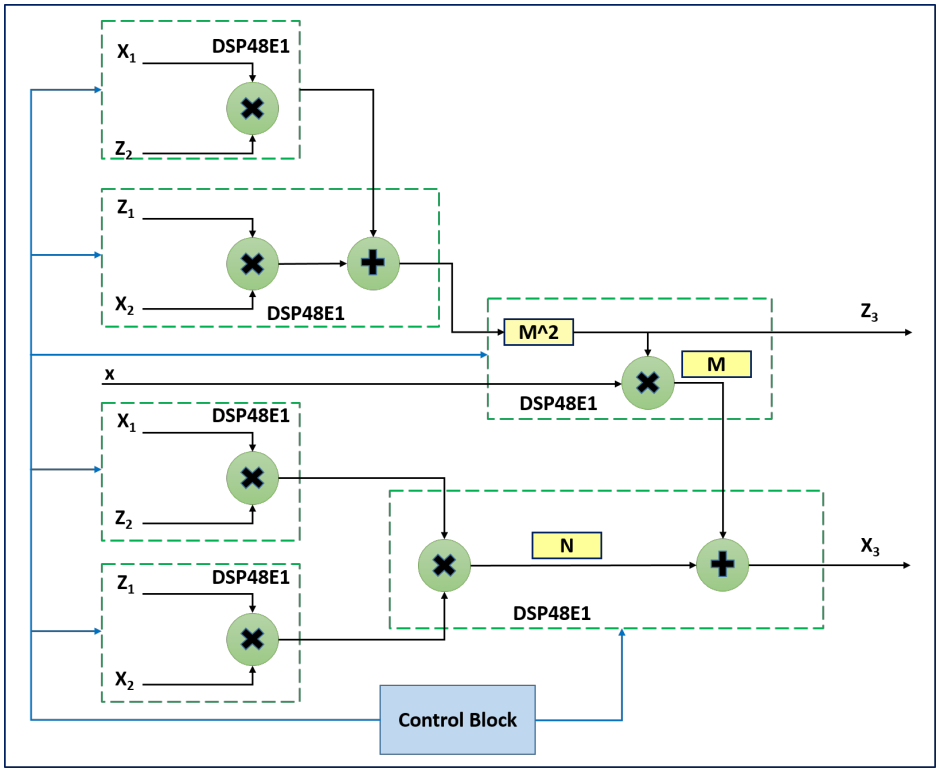

Figure 3. Single Cycle Point addition using DSP48E1 based on Montgomery Technique

The architectures proposed for elliptic curve point multiplications in binary field use NIST recommended curves including 163, 233, and 283 sizes for implementation. Only x-coordinate values are enough to be computed.

The functional verification of the elliptic curve point multiplication algorithm is first performed using Vivado high-level synthesis (HLS) 2018.3 tool. The generated x-coordinate value and the computed shared secret value by ECDH algorithm are shown in Fig. 6 and 7, respectively. Table II shows the performance comparisons of the proposed work against existing research outcome found in the open literature.

Then, the algorithm has been synthesized and implemented on the aforementioned FPGA devices, and the

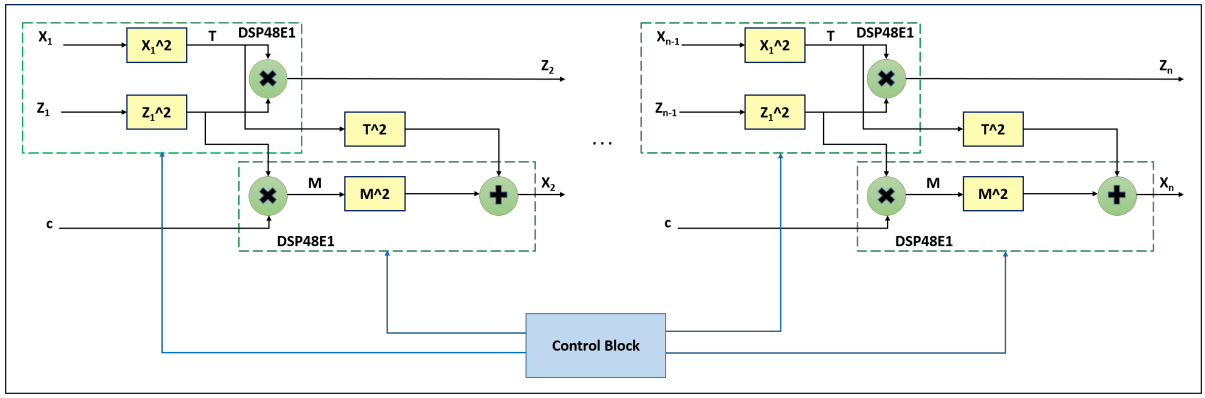

Figure 4. Point doubling using cascaded DSP48E1 slices based on Montgomery Method

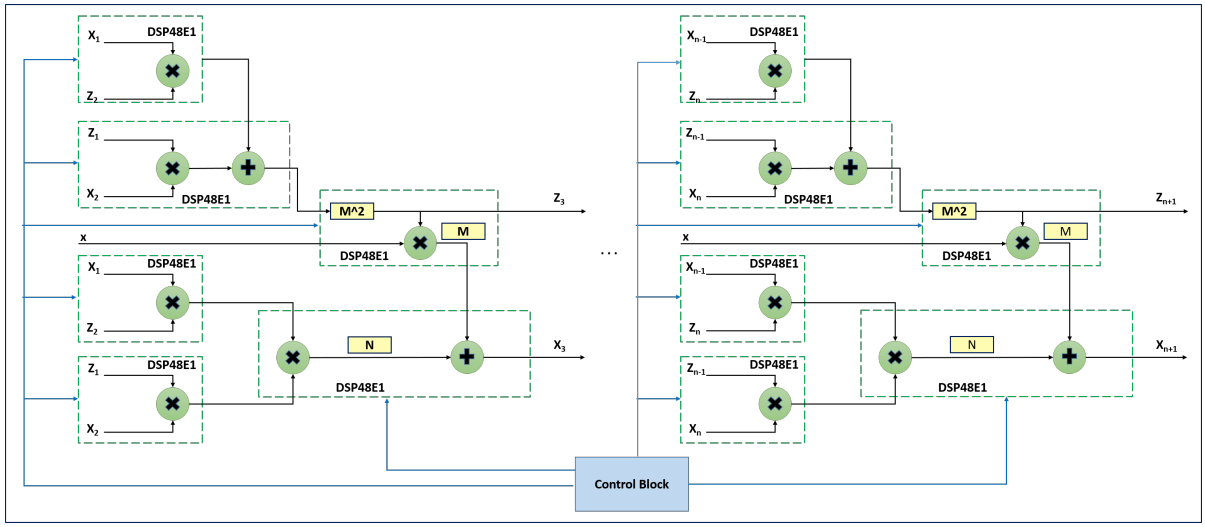

Figure 5. Point addition using cascaded DSP48E1 slices based on Montgomery Technique

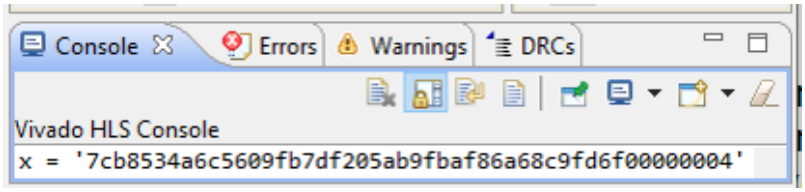

Figure 6. The generated *x*-coordinate value

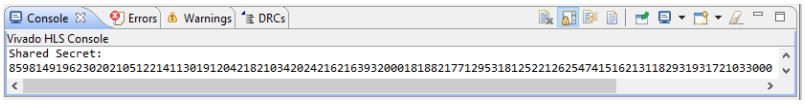

Figure 7. The computed shared secret value

generated VHDL codes were synthesized and implemented using Vivado 2018.3 IDE.

The inversion arithmetic required to convert from projective coordinates back to affine coordinates is pre-computed using Extended Euclidean algorithm. The NIST recommended elliptic curves over binary fields including 163, 233, and 283 bits are considered so that they could be used for ECDH key exchange based on users' preferences. Fair result comparisons could be done when the implementation platforms are similar and the used fields are the same. Thus, prime field implementations are not included in result comparisons. Moreover, existing implementation outcomes

that have used devices older than 7-series platforms are not included in result comparison for fair comparisons, since our architecture utilized DSP48E1 hard-cores which exist in such platforms. In addition, existing architectures which have used only pure hardware resources such as the generic FPGAs' logic elements and embedded hard-cores such as DSP slices and BRAMs are used for fair comparisons excluding design approaches such as hardware/software co-designs as well as software implementations.

Therefore, the 7-series FPGAs' dedicated DSP48E1 slices and BRAM blocks together with generic FPGA fabrics are utilized for the proposed architecture that consists Montgomery algorithm based on projective coordinates to speed up the performance of point multiplication. The approach enabled to achieve better performance of the scalar multiplication for ECDH in terms of performance time and resource utilization. Instead of using only the FPGA traditional logic elements that are required in large amount to implement such complex arithmetic operations, the dedicated blocks are used in addition, to efficiently implement the proposed architecture for the purpose of enhancing speed of performance and optimized resource utilization. The proposed architecture balanced the utilization of the traditional FPGA logic elements and that of the dedicated hard-cores as shown in Table II; and, the proposed method utilized fewer numbers of the generic elements compared to existing results, while outperforming in terms of the total time required for scalar multiplication for all the used field sizes. Thus, for implementations performed on Kintex-7 platform, existing similar works produced timing performance in *ms* which is very slower compared to the achieved results in μs of the proposed work.

Moreover, the results shown by existing works for hardware resources in terms of slices and look up tables (LUTs) is larger than the proposed work. Table II shows that existing works have not used dedicated hard-cores, but, the proposed work used DSP slices and BRAM blocks for implementation of the proposed architecture. For example, as shown in Table II, for the different field sizes on Kintex-7 platform, the proposed work utilized from 971 to 1214 FPGA slices, 4 to 6 BRAMs, and 32 to 64 DSP slices, as well as 1.75μs to 3.53 μs timing performance. Whereas, for the same platform, existing research outcomes showed from 3016 [7] to 6620 slices [36], 1.06 *ms* [36] to 2.66 *ms* [7] timing performance with no DSP slices and BRAMs. Similarly, on Virtex-7 platform, the present work utilized 1161 slices, 4 BRAMs and 32 DSP slices, while achieving 1.55μs timing performance for the field size of 163 bits. Whereas, the outcomes of the existing works range from 1476 [37] to 8736 [38] slices and 1.70μs [26] to 10.80μs [4] timing performance, with no dedicated hard-cores utilization. As sown in Table II, the performance of the proposed work balances the resource utilization of both the generic FPGA fabrics and the hard-cores while providing efficient performance.

The performance comparisons are also analyzed graphically as shown in Figs. 8 and 9. Fig. 8 shows comparisons based on consumed hardware resources on Kintex-7 and Virtex-7 platforms. Compared to the outcomes presented by existing works, smaller amounts of slices are utilized by the proposed work on both platforms.

Fig. 9 shows the comparison of timing performance of existing research results with the current work on Kintex-7 and Virtex-7 platforms. The performance comparisons in Fig. 9 show that the proposed work has better timing performance than existing similar works found in the literature for both Kintex-7 and Virtex-7 platforms.

6. Conclusions

In this research work, smaller and efficient EC point multiplication architecture for binary field NIST curves is proposed and implemented on FPGA based on Montgomery algorithm with projective coordinates system, and used for implementation of ECDH algorithm. This compact and efficient architecture is aimed to address the IoT network security that involves constrained devices that require lightweight and efficient security mechanisms. Exploiting the advantages of modern FPGA resources is helpful to produce effective architecture for complex cryptographic algorithms, in order to efficiently implement them. Utilization of generic FPGA fabrics for implementation of complex algorithms consumes large amounts of such resources and leads to slow performance. On the other hand, utilization of the FPGAs' hard-cores for simple logical and arithmetic operations is not cost effective in terms of hardware resource utilization and is limited in terms of flexibility. Thus, the proposed work balanced the uses of these FPGA resources and utilized the traditional FPGA resources for simple arithmetic and logical processes while utilizing the embedded hard-cores for complex arithmetic and logical operations. As a result it balanced the utilization of both resource types and reduced the performance bottlenecks of elliptic curve scalar multiplication that is used for ECDH algorithm on FPGA. Using this approach, smaller hardware resources (971 slices, 4 BRAMS, and 32 DSP slices) and balanced timing performances (1.74μs) are achieved on Kintex-7 platform for the proposed architecture compared to existing research outcomes found in the literature. On Virtex-7, enhanced timing performance (1.55μs) is achieved and balanced resource (1164 slices, 4 BRAMs, and 32 DSP slices) are utilized. In the future, elliptic curve digital signature will be included to enable the hardware based cryptosystem can provide multiple cryptographic security services.

TABLE II. Performance Comparisons for Implementations on Xilinx 7-series FPGA Platforms

Work	Field size	Device	Slice	LUT	BRAM	DSP	Freq. (MHz)	Clock cycle	Time
[7]	233	Kintex-7	-3016	9151	-	-	255.66	679776	2.66 *ms*
	283		4625	14440	-	-	251.98	1395312	5.54 *ms*
[36]	163	Kintex-7	6620	7963	-	-	306.48	325564	1.06 *ms*
This work	163	Kintex-7	971	2877	4	32	324.52	1031	1.74 μs
	233		1053	3168	6	64	345.324	1098	2.72 μs
	283		1214	3225	6	64	375.745	1194	3.53 μs
[34]	163	Virtex-7	4150	14202	-	-	352	1119	3.18 μs
	163	Virtex-7	11657	41090	-	-	159	450	2.83 μs
[37]	163	Virtex-7	1476	4721	-	-	397	4168	10.51 μs
[38]	163	Virtex-7	8736	27105	-	-	223	780	3.50 μs
[26]	163	Virtex-7	na	28911	-	-	320.5	547	1.70 μs
[4]	163	Virtex-7	3657	10128	-	-	135	3426	10.80 μs
This work	163	Virtex-7	1161	-	4	32	381	1214	1.55 μs

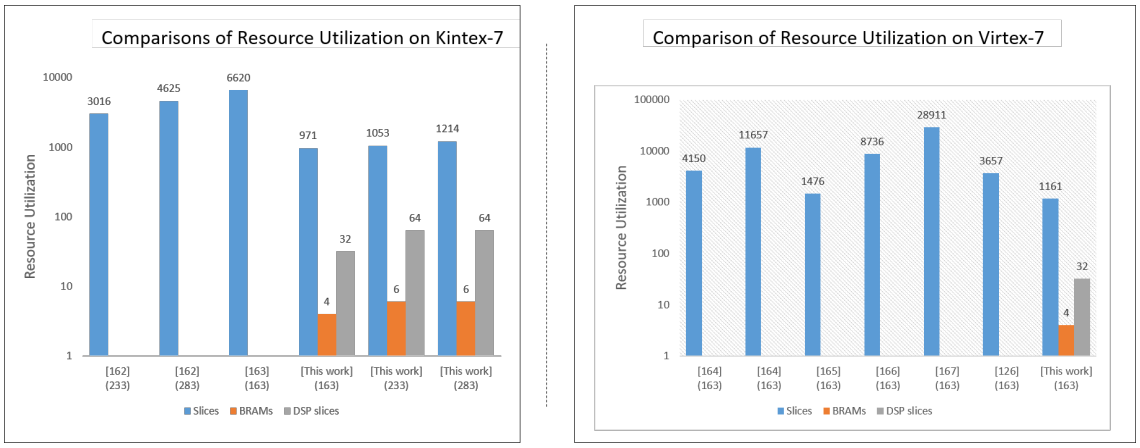

Figure 8. Comparison of resource utilization

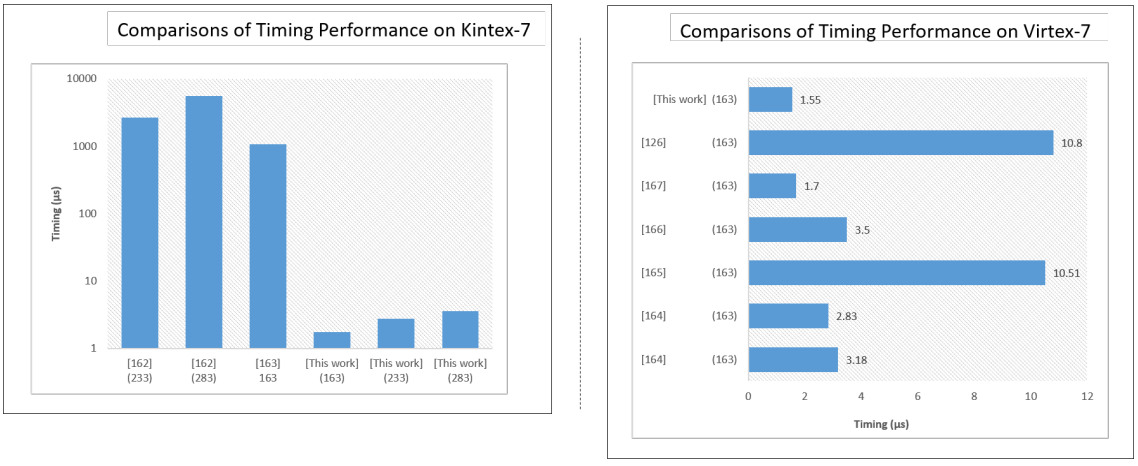

Figure 9. Result comparisons based on timing performance

References

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic*

Curve Cryptography. New York: Springer, 2004.

[3] K. Maletsky, "rsa vs ecc comparison for embedded systems," Atmel, White Paper, 2015.

[4] M. Imran, I. Shafi, A. R. Jafri, and M. Rashid, "Hardware design and implementation of ecc based crypto processor for low-area-applications on fpga," in *2017 International Conference on Open Source and Technologies (ICOSST)*. Lahore, Pakistan: IEEE, 2017, pp. 54–59.

[5] H. Asshidiq, A. Sasongko, and Y. Kurniawan, "Implementation of ecc on reconfigurable fpga using hard processor system," in *In 2018 International Symposium on Electronics and Smart Devices (ISESD)*. Bandung, Indonesia: IEEE, 2018, pp. 1–6.

[6] M. M. Islam, M. S. Hossain, M. K. Hasan, M. Shahjalal, and Y. M. Jang, "Fpga implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field," *IEEE Access*, vol. 7, p. 178811–178826, 2019.

[7] M. S. Hossain, E. Saeedi, and Y. Kong, "High-speed, area-efficient, fpga-based elliptic curve cryptographic processor over nist binary fields," in *In 2015 IEEE International Conference on Data Science and Data Intensive Systems*. Sydney, NSW, Australia: IEEE, 2015, pp. 175–181.

[8] G. D. Sutter, J.-P. Deschamps, and J. L. Imaña, "Efficient elliptic curve point multiplication using digit-serial binary field operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, p. 217–225, 2013.

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

[10] G. Orlando and C. Paar, "A high-performance reconfigurable elliptic curve processor for gf (2m)," in *In International Workshop on Cryptographic Hardware and Embedded Systems*. Berlin, Heidelberg: Springer, 2000, pp. 41–56.

[11] J. Wolkerstorfer, "Dual-field arithmetic unit for gf(p) and gf(2m)," in *In CHES, Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 2002, pp. 500–514.

[12] J. H. Guo and C. L. Wang, "Systolic array implementation of euclid's algorithm for inversion and division in gf(2m)," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 10, p. 1161–1167, 1998.

[13] X. Hu, X. Zheng, S. Zhang, S. Cai, and X. Xiong, "A low hardware consumption elliptic curve cryptographic architecture over gf(p) in embedded application," *Electronics*, vol. 7, no. 7, p. P. 104, 2018.

[14] D. Boneh and I. E. Shparlinski, "On the unpredictability of bits of the elliptic curve diffie–hellman scheme," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, ser. Lecture Notes in Computer Science, vol. 2139. Springer, 2001, pp. 201–212.

[15] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of computation*, vol. 48, no. 177, pp. 243–264, 1987.

[16] Y. A. Shah, K. Javeed, S. Azmat, and X. Wang, "Redundant signed digit based high-speed elliptic curve cryptographic processor," *Journal of Circuits, Systems and Computers*, vol. 28, no. 5, p. p.1950081, 2019.

[17] K. Javeed and X. Wang, "Fpga based high-speed spa-resistant elliptic curve scalar multiplier architecture," *Int. J. Reconfigurable Comput.*, vol. 2016, no. 5, pp. 1–10, 2016.

[18] K. Javeed and X. Wang, "Low latency flexible fpga implementation of point multiplication on elliptic curves over gf(p)," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 214–228, 2017.

[19] M. S. Hossain, Y. Kong, E. Saeedi, and N. C. Vayalil, "High-performance elliptic curve cryptography processor over nist prime fields," *IET Computers and Digital Techniques*, vol. 11, no. 1, pp. 33–42, 2017.

[20] A. Daly, W. Marnane, T. Kerins, and E. Popovici, "An fpga implementation of a gf(p) alu for encryption processors," *Elsevier - Microprocessors and Microsystems*, vol. 28, no. 5-6, pp. 253–260, 2004.

[21] J. Vliegen, N. Mentens, J. Genoe, A. Braeken, S. Kubera, A. Touhafi, and I. Verbauwhede, "A compact fpga-based architecture for elliptic curve cryptography over prime fields," in *In Proc. IEEE Int. Conf. Appl.-Specific Syst. Archit. Process*. Rennes, France: IEEE, 2010, pp. 313–316.

[22] G. Orlando and C. Paar, "A scalable gf(p) elliptic curve processor architecture for programmable hardware," in *In International Workshop on Cryptographic Hardware and Embedded Systems*. Berlin, Heidelberg: Springer, 2010, pp. 348–363.

[23] A. Satoh and K. Takano, "A scalable dual-field elliptic curve cryptographic processor," *IEEE Transactions Computers*, vol. 52, no. 4, pp. 449–460, 2003.

[24] M. Imran and M. Rashid, "Architectural review of polynomial bases finite field multipliers over $gf(2^m)$," in *2017 International Conference on Communication, Computing and Digital Systems (C-CODE)*. Islamabad, Pakistan: IEEE, 2017, pp. 331–336.

[25] S. Khan, K. Javeed, and Y. A. Shah, "High-speed fpga implementation of full-word montgomery multiplier for ecc applications," *Microprocessors Microsyst.*, vol. 62, pp. 91–101, 2018.

[26] J. Li, Z. Li, S. Cao, J. Zhang, W. Wang *et al.*, "Speed-oriented architecture for binary field point multiplication on elliptic curves," *IEEE Access*, vol. 7, pp. 32 048 – 32 060, 2019.

[27] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of elliptic curve scalar multiplier on lut-based fpgas for area and speed," *IEEE*, vol. 21, no. 5, p. 901–909, 2013.

[28] S. Liu, L. Ju, X. Cai, Z. Jia, and Z. Zhang, "High performance fpga implementation of elliptic curve cryptography over binary fields," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*. Beijing, China: IEEE, 2014, pp. 148–155.

[29] H. Mahdizadeh and M. Masoumi, "Novel architecture for efficient fpga implementation of elliptic curve cryptographic processor over gf(2^{163})," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 21, no. 12, p. 2330–2333, 2013.

[30] T. Güneysu and C. Paar, "Ultra high performance ecc over nist primes on commercial fpgas," in *In International Workshop on Cryp-*

tographic Hardware and Embedded Systems. Berlin, Heidelberg: Springer, 2008, pp. 62–78.

[31] F. Turan and I. Verbauwhede, "Compact and flexible fpga implementation of ed25519 and x25519," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 3, pp. 1–21, 2019.

[32] C. McIvor, M. McLoone, and J. McCanny, "An fpga elliptic curve cryptographic accelerator over gf(p)," in *In Irish Signals and Systems Conference (ISSC)*. Belfast, Ireland: IET, 2004, p. 589–594.

[33] L. Li and S. Li, "High-performance pipelined architecture of elliptic curve scalar multiplication over gf((2^m))," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 24, no. 4, p. 1223–1232, 2016.

[34] Z. U. Khan and M. Benaissa, "High-speed and low-latency ecc processor implementation over gf(2m) on fpga," *IEEE Trans. Very Large Scale Integr. Syst*, vol. 25, no. 1, p. 165–176, 2016.

[35] "Xilinx, 7 series dsp48e1 slice. user guide ug479," *http://www. xilinx. com.*

[36] M. S. Hossain, E. Saeedi, and Y. Kong, "High-performance fpga implementation of elliptic curve cryptography processor over binary field $gf(2^{163})$," in *In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*. Rom, Italy: Springer, 2016, pp. 415–422.

[37] Z. U. Khan and M. Benaissa, "Throughput/area-efficient ecc processor using montgomery point multiplication on fpga," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 11, pp. 1078–1082, 2015.

[38] Z. U. Khan and M. Benaissa, "High speed ecc implementation on fpga over gf(2m)," in *In Proc.25th Int. Conf. on Field-programmable Logic and Applications (FPL)*. London, UK: IEEE, 2015, pp. 1–6.

P.G.V. Suresh Kumar Dr. P.G.V. Suresh Kumar born in India. He received his PhD in Computer Networks in 2007. Currently working as a Professor of Information Communication Technology at Ethiopian Technical University, Addis Ababa. His research interests include Computer Networking, Wireless networks, Network Security, Cryptography and A.I.

Yalemzewd Negash Shiferaw Yalemzewd Negash Shiferaw is Assistant Professor at Addis Ababa Institute of Technology (AAiT) in Addis Ababa University (AAU), Addis Ababa, Ethiopia. Currently, Dr. Yalemzewd is the dean of the School of Electrical and Computing Engineering (SECE). He earned his B.Sc. Degree in Electrical Engineering, his MSc and PhD degrees in Communication Engineering from Faculty of Technology and now AAiT, AAU. He has served the school for more than 12 years as Lecturer and Assistant Professor. He has taught a number of courses in electrical engineering, communication engineering and railway engineering. His research interests focus on wireless and optical communication, modeling traffic patterns, information theory and railway signals. He is actively involved in supervising M.Sc. Students and co-supervising PhD students. His articles are published in Value tools, IEEE sponsored conference proceedings and on the Journal of Ethiopian Engineers and Architects. He is also engaged in the industry sector as a General Manager of a private company.

Abiy Tadesse Abebe Abiy Tadesse Abebe is currently a PhD candidate at Addis Ababa University, Addis Ababa Institute of Technology, School of Electrical and Computer Engineering, Addis Ababa, Ethiopia. He obtained his MSc degree in Electrical and Computer Engineering from AAU in 2009. Abiy has published several research papers in the area of reconfigurable computing based cryptosystem implementations for applications such as IoT security, Fog-Cloud security, healthcare IoT, etc., giving main emphasis on optimization and performance enhancement of existing algorithms. His areas of interest include: IoT security, healthcare IoT, Cloud security, hybrid cryptosystems, reconfigurable computing based implementations, wireless network security, high level synthesis, Machine Learning, AI, etc.