FPGA Based Crypto Processor for Elliptic Curve Point Multiplication (ECPM) Over $GF(2^{233})$

Malik Imran

Department of Electrical Engineering
Abasyn University
Islamabad, Pakistan

Faisal Shehzad

Department of Electrical Engineering
University of Bremen
Bremen, Germany

Abstract

Elliptic Curve Cryptography (ECC), provides all public key cryptographic primitives like digital signatures and key agreement algorithms/protocols in a constrained applications such as wireless sensor networks and radio frequency identification networks (RFIDs). In order to achieve digital signatures and key agreements, point/scalar multiplication is necessary to perform. However, we demonstrate the hardware architecture of elliptic curve point multiplication for low area constrained applications over binary $GF(2^m)$ field with $m = 233$ bit field size. The lower area is achieved, by using single hybrid karatsuba multiplier for both squarer and multiplication computations. The novel architecture is modeled in Verilog (HDL) using Xilinx (ISE) design tool and synthesized for Virtex 7 field-programmable-gate-array (FPGA). Moreover, it achieves a maximum operational frequency of 157MHz and utilizes only 11849 FPGA slices.

1. Introduction

Elliptic Curve Cryptography (ECC) is proposed by Neal Koblitz [1] and Victor Miller [2] in 1985. ECC has been commercially accepted and adopted by many standardized organizations such as American National Standards Institute (ANSI) [3], Institute of Electrical and Electronics Engineering (IEEE) [4], International Organization for Standardization (ISO) [5] and National Institute of Standards and Technology (NIST) [6]. It gains popularity, due to provision of shorter key lengths as compared with other public key cryptosystems such as Rivest-Shamir-Adleman (RSA) [7]. A system, where cryptographic operations are involved, called cryptosystem [8].

Point multiplication (PM) is the basic building block in all ECC primitives (for digital signatures and for key exchange and agreement protocols) and is computationally more intensive part of any asymmetric curve based cryptosystem [9]. In order to perform PM, two types of fields are involved i.e., prime field $GF(p)$ and 2) binary field $GF(2^m)$ [9].

However, binary field is particularly more attractive for hardware implementations [9]. Furthermore, each of these fields (prime and binary) can be implemented either by adopting affine coordinates or projective coordinates in polynomial basis (PB) or with normal basis (NB) representations. However, the projective coordinates are well suited to improve latency of the architecture [10] and PB representations provide efficient finite field (FF) multiplication [9].

For different cryptographic applications such as wireless sensor nodes, radio frequency identification networks (RFID), cloud computing and for high performances etc, various architectural approaches has been proposed to implement PM. Most of the commonly used architectural approaches are crypto processor [10]-[12], crypto coprocessor [13] and multi-core crypto processor [14]. A crypto processor is a programmable hardware, with a dedicated instruction set and it contains memory unit, control unit and an arithmetic unit [15]. Coprocessor, contains a host processor coupled with the crypto unit [16]. The host processor is used to control the crypto unit. In multi-core architectural approach, various cryptographic operations can be considered in parallel [15]. For further crypto architectural (crypto processors, coprocessors and multi-core crypto processors) details, interested readers can consult [15] and [16].

The performance of the entire crypto architecture is mainly depends upon its FF multiplier and inversion [17]. In recent research practices, the most commonly implemented multipliers are bit parallel [12], [17]-[19] and digit serial [10] and [20]. The bit parallel multipliers are used to achieve higher performances [18] whereas digit serial multipliers are used to achieve optimal throughput/area designs [20]. The bit parallel multipliers include, hybrid karatsuba multiplier, implemented in [12] and [18] and karatsuba Ofman multiplier, implemented in [17] and [19]. Moreover, digit serial multipliers can be implemented by varying digit sizes as implemented in [10] and [20]. Longer digit sizes results longer critical path delay and higher hardware cost [10]. Moreover, the cost of inversion can be reduced by

adopting projective coordinates with Itoh Tsujii algorithm [20].

The main purpose of this work is to develop an elliptic curve point multiplication (ECPM) crypto processor for low area applications. For proposed ECPM processor, we used PB representation as it performs efficient FF multiplications than NB. The NB representation is useful where frequent squaring's are involved [20]. Furthermore, for polynomial multiplication we use bit parallel FF multiplier based on hybrid karatsuba multiplier. The hybrid karatsuba multiplier is also used in this work to compute squarer and inversion operations for ECPM over $GF(2^m)$ with $m = 233$ bit field size. Finally, the proposed crypto processor (ECPM) is modeled in Verilog (HDL) using Xilinx ISE 14.2 and synthesized over newer Virtex 7 (xc7vx690t-3ffg1930) field-programmable-gate-array (FPGA) for results.

The rest of this paper is organized as follows: Section 2, provides elliptic curve cryptography. Security strength of elliptic curves are further discussed in Section 3. Section 4, presents the point multiplication operation. Section 5, describes the finite field arithmetic operations over $GF(2^m)$ field. The proposed hardware architecture is further discussed in Section 6 whereas implementation results are provided in Section 7. Finally, Section 8 concludes the article.

2. Elliptic curve cryptography (ECC)

As shown in introductory part of this article that, two types of fields are generally involved to implement elliptic curves i-e., prime field $GF(p)$ and binary fields $GF(2^m)$ [9]. Prime field is best suited for software implementations whereas binary field is suitable for hardware implementations [9]. However, binary field is selected in this work and it can be implemented either by using:

- supersingular elliptic curves
- or non-supersingular elliptic curves

Supersingular elliptic curves define a special class of curves with some special properties which makes it unstable for cryptography [9]. However, the non-supersingular elliptic curves are considered as more secure and is defined by the curve constant parameters $a, b \in GF(2^m)$ with $b \neq 0$, consists of the set of points $P = (x, y)$, where $x, y \in GF(2^m)$, satisfying the Equation (1) [9]:

$$E: y^2 + xy = x^3 + ax^2 + b \quad (1)$$

For $GF(2^m)$, a Lopez Dahab projective form of non-singular elliptic curve (presented in Equation 1)

is defined as a set of points $P(X:Y:Z)$, satisfying the following Equation (2) [9]:

$$E: Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4 \quad (2)$$

In Equation (2), the variables ' X ', ' Y ' and ' Z ' are the Lopez Dahab projective elements of point $P(X:Y:Z)$ where $Z \neq 0$, ' a ' and ' b ' are the curve constants with $b \neq 0$. For, complete mathematical formulations and constructions of Equation (1) and Equation (2), interested readers can consult [9].

2.1 Hierarchical model of ECC

The typical hierarchy of ECC is organized into four different layers as shown in Figure 1. The top most layer is commonly known as protocol layer and it consists of elliptic curve digital signature algorithm (ECDSA) [21], elliptic curve diffie hellman (ECDH) [22] and elliptic curve integrated encryption scheme (ECIES) [23]. The ECDSA algorithm is used for key authentication whereas ECDH protocol is used for key establishment and agreement between server and client over the network. Moreover, ECIES protocol is used for message encryption and decryption. Simply, protocol defines the set of rules to achieve security.

Figure 1: Hierarchical model of elliptic curves [9]

The protocol layer is completely rely on the ECPM layer. ECPM defines the $Q = k \text{ times } P$, where ' P ' and ' Q ' are the initial and final points on the curve whereas ' k ' is the size of underlying field and is mainly rely on the elliptic curve point addition (ECPA) and doubling (ECPD) operations. If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are two distinct points on the defined elliptic curve then ECPA will be $R = (x_r, y_r) = P + Q$ with $P \neq Q$. Similarly, ECPD will be, $R = (x_r, y_r) = 2P = P + P$. Moreover, ECPA and ECPD layer operations are computed using the most bottom arithmetic layer of ECC. The arithmetic layer consists of FF arithmetic operations (i-e., addition, multiplication, squarer and inversion).

These arithmetic layer operations will be discussed later in this article (Section 5).

3. Security strength of ECC

The security strength of ECC mainly depends upon its discrete logarithmic problem (DLP) [9]. The elliptic curve discrete logarithmic problem (ECDLP) is solving the inverse operation to determine 'k' when initial point 'P' and PM ($Q = k.P$) are known. PM is based on solving the exponentiations whereas its inverse i.e., solving logarithmic is harder than exponentiations [24].

According to author's best knowledge, there is no known common method (sub-exponential algorithm) to solve the ECDLP. This makes ECC, a most promising branch of public key cryptography (PKC) [25]. Moreover, ECC offers same level of security, compared with others such as traditional DLP based schemes [26]. In addition with security, ECC offers smaller key sizes and storage memory which results lower power consumptions and channel bandwidths over the network as compared with other PKC based cryptosystems such as RSA [9]. However, different security levels for symmetric and asymmetric (ECC, Diffie Hellman (DH) and RSA) cryptographic schemes are illustrated in Table 1.

Table 1: Different security levels for symmetric and asymmetric cryptographic schemes [26]

Symmetric key size (bits)	Asymmetric key size (bits)	
	ECC	RSA & DH
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	521	15360

As shown in Table 1, column 1 presents the symmetric key size (different security levels) in terms of number of bits whereas column 2 shows the asymmetric key size (in bits). Additionally, column 2 is sub portioned into two more columns which presents the required key bits for corresponding ECC and RSA & DH based cryptosystems. To use RSA and DH to protect 128 bit symmetric keys, one should use 3072 bit parameters which is three times the size in use throughout the internet today [26]. For 128 bit equivalent security, ECC requires only 256 bits.

4. Point multiplication (PM) over $GF(2^m)$

In asymmetric curve based cryptosystems (such as ECC), the core operation is a PM. Multiple algorithms e.g., Double and Add [27], Montgomery

[28] and Lopez Dahab [29] have been used in literature to implement PM. In order to compute PM, we have used the Lopez Dahab algorithm [29] and is presented here in Algorithm 1.

PM is defined as consider a base point 'P' and a large integer 'k' of the size of underlying field, then the PM (i.e., $Q = k.P$) will be the addition of 'k' copies of point 'P' as in Equation (3):

$$Q = k.P = k(P + P + P + \dots + P) \quad (3)$$

Algorithm 1: Lopez Dahab Algorithm [29]

Inputs: $P = (x_p, y_p) \in GF(2^m)$,

$k \leftarrow (k_{i-j}, \dots, k_1, k_0)$ where, k is j bit integer

Output: $k.P = (x_q, y_q)$

Step 1: Affine to Lopez Dahab Conversion

- 1) $X_1 \leftarrow (x_p)$ 2) $Z_1 \leftarrow 1$
- 3) $Z_2 \leftarrow (X_1)^2$ 4) $X_2 \leftarrow (Z_2)^2$
- 5) $X_2 \leftarrow (X_2 + b)$

Step 2: Point Multiplication (PM)

for int $i = j - 2$ down to 0 do

- 1) $V_1 \leftarrow (X_1 Z_2)$ 2) $V_2 \leftarrow (X_2 Z_1)$
- 3) $V_3 \leftarrow (X_1 Z_1)$ 4) $R_3 \leftarrow (Z_1)^2$
- 5) $R_3 \leftarrow (R_3)^2$ 6) $Z_2 \leftarrow (V_1 + V_2)$
- 7) $Z_2 \leftarrow (Z_2)^2$ 8) $Z_1 \leftarrow (V_3)^2$
- 9) $V_1 \leftarrow (V_1 V_2)$ 10) $V_2 \leftarrow (x_p Z_2)$
- 11) $V_3 \leftarrow (b R_3)$ 12) $R_3 \leftarrow (X_1)^2$
- 13) $R_3 \leftarrow (R_3)^2$ 14) $X_2 \leftarrow (V_1 + V_2)$
- 15) $X_1 \leftarrow (V_3 + R_3)$

if ($i = 0$ and $k_i = 1$)

swap(X_1, X_2), swap(Z_1, Z_2)

end if

end for

Step 3: Lopez Dahab to Affine Conversion

- 1) $V_1 \leftarrow \text{Inv}(Z_1)$ 2) $V_2 \leftarrow \text{Inv}(Z_2)$
- 3) $V_3 \leftarrow \text{Inv}(x_p)$ 4) $R_1 \leftarrow (X_1 V_1)$
- 5) $V_2 \leftarrow (X_2 V_2)$ 6) $R_3 \leftarrow (x_p)^2$
- 7) $R_3 \leftarrow R_3 + y_p$ 8) $V_1 \leftarrow (x_p + R_1)$
- 9) $V_2 \leftarrow (x_p + V_2)$ 10) $V_1 \leftarrow (V_1 V_3)$
- 11) $V_2 \leftarrow (V_1 V_2)$ 12) $V_2 \leftarrow (V_2 + R_3)$
- 13) $V_2 \leftarrow (V_1 V_2)$ 14) $R_2 \leftarrow (V_2 + y_p)$

Return: $k.P = (x_q, y_q) = (R_1, R_2)$

In order to implement Lopez Dahab algorithm for PM, it requires a size of underlying field 'k' along with the initial point 'P' with its coordinates (x_p, y_p) as an input and produces (x_q, y_q) coordinates of the final point 'Q' as an output. It consists of three steps:

Step 1: Affine to projective (Lopez Dahab) conversions are performed in the first step.

Step 2: In PM step, ECPA ($P = P + Q$) and ECPD ($P = 2P$) will be computed, based on the inspected value of size of underlying field (k_i).

Step 3: Projective (Lopez Dahab) to Affine conversions are performed in the third step.

5. Finite field arithmetic over $GF(2^m)$

In order to implement PM operation over $GF(2^m)$, the required FF arithmetic operations are addition, multiplication, squarer and inversion. These arithmetic operations are discussed in the next subsequent sections (Section 5.1 to 5.5).

5.1 Addition

Two ' m ' bit polynomials addition is defined as, if $A(x), B(x) \in GF(2^m)$, with

$$A(x) = \sum_{i=0}^{m-1} a_i x^i$$

$$B(x) = \sum_{i=0}^{m-1} b_i x^i, \text{ then}$$

$$C(x) = \sum_{i=0}^{m-1} a_i x^i \oplus \sum_{i=0}^{m-1} b_i x^i \quad (4)$$

In Equation (4), $C(x)$ is the resulting polynomial, $A(x)$ and $B(x)$ are the input polynomials and ' m ' specifies the field length.

5.2 Multiplication

Polynomials multiplication of two ' m ' bit elements is defined as, if $A(x), B(x) \in GF(2^m)$, with

$$A(x) = \sum_{i=0}^{m-1} a_i x^i$$

$$B(x) = \sum_{i=0}^{m-1} b_i x^i, \text{ then}$$

$$C(x) = \sum_{i=0}^{2m-2} c_i x^i = A(x) \times B(x) \bmod P(x) \quad (5)$$

In Equation (5), $C(x)$ is the resulting polynomial, $A(x)$ and $B(x)$ are the input polynomials, $P(x)$ is the irreducible polynomial and ' m ' specifies the field length.

5.3 Squarer

The polynomial squaring of $A(x)$ is a linear operation and it can be defined as, if $A(x) \in GF(2^m)$, with

$$A(x) = a_{m-1}x^{m-1} + \dots + a_2x^2 + a_1x^1 + a_0x^0, \text{ then}$$

$$A(x)^2 = a_{m-1}x^{2m-2} + \dots + a_2x^4 + a_1x^2 + a_0x^0$$

The binary representation of polynomial $A(x)^2$ can be achieved by interleaving '0' between two successive data bits [9].

5.4 Reduction

After ' m ' bit polynomials multiplication and squaring, the resultant polynomial will be ' $2 \times m - 1$ ' bits. However, after each FF multiplication and squaring, FF reduction is required. In order to perform reduction operation, the irreducible polynomial ($P(x)$) recommended by NIST in the FIPS 186-2 Standard document [6] is presented in Equation (6). The NIST recommended reduction algorithm according to Equation (6) is presented here in Algorithm 2 and is also presented in [9] (Algorithm 2.42).

$$P(x) = x^{233} + x^{74} + 1 \quad (6)$$

Algorithm 2: NIST Reduction Algorithm [9]

Input: A binary polynomial $C(x)$ of degree at most 464.

Output: $C(x) \bmod P(x)$

1. For i from 15 downto 8 do
 2. $T \leftarrow C[i]$
 3. $C[i-8] \leftarrow C[i-8] \oplus (T \ll 23)$
 4. $C[i-7] \leftarrow C[i-7] \oplus (T \gg 9)$
 5. $C[i-5] \leftarrow C[i-5] \oplus (T \ll 1)$
 6. $C[i-4] \leftarrow C[i-4] \oplus (T \gg 31)$
 7. $T \leftarrow C[7] \gg 9$
 8. $C[0] \leftarrow C[0] \oplus T$
 9. $C[2] \leftarrow C[2] \oplus (T \ll 10)$
 10. $C[3] \leftarrow C[3] \oplus (T \gg 22)$
 11. $C[7] \leftarrow C[7] \& 0x1FF$
 12. **Return:**
 $(C[7], C[6], C[5], C[4], C[3], C[2], C[1], C[0])$
-

5.5 Inversion

The inverse of a polynomial $A(x)$ is defined as,

$$A(x)^{-1} = A(x) \times \frac{1}{A(x)} = 1 \quad (7)$$

In order to compute inversion i.e., $(1/A(x))$, square Itoh Tsujii algorithm (presented in Algorithm 3) is implemented in this work. It requires only field squaring's and multiplications [30], as shown in Algorithm 3.

For $GF(2^m)$, with $m = 233$ bit key length, inversion is computed by using ' $m - 1$ ' squarer operations followed with 10 field multiplications. The implemented addition chain for inversion computations is 1, 1, 3, 1, 7, 14, 1, 29, 58 and 116.

Algorithm 2: Square Itoh Tsujii Inversion**Input:** A**Output:** $A^{-1} = A \times \frac{1}{A} = 1$

Squares	Chain	Multiplications
1) $R_1 = A^2$	1	$R_2 = A \times R_1$
2) $R_1 = R_2^2$	1	$R_2 = A \times R_1$
3) $R_1 = R_2^2$	3	$R_2 = R_1 \times R_2$
4) $R_1 = R_2^2$	1	$R_2 = R_1 \times A$
5) $R_1 = R_2^2$	7	$R_2 = R_1 \times R_2$
6) $R_1 = R_2^2$	14	$R_2 = R_1 \times R_2$
7) $R_1 = R_2^2$	1	$R_2 = R_1 \times A$
8) $R_1 = R_2^2$	29	$R_2 = R_1 \times R_2$
9) $R_1 = R_2^2$	58	$R_2 = R_1 \times R_2$
10) $R_1 = R_2^2$	116	$R_1 = R_1 \times R_2$

6. Proposed ECPM processor

The proposed ECPM processor consists of multiplexers (M1, M2 and M3), demultiplexer (D1), adder unit, multiplier unit, intermediate registers and a finite state machine (FSM) based controller as shown in Figure 2.

Figure 2: Proposed elliptic curve point multiplication (ECPM) processor

6.1 Multiplexers and demultiplexer

The multiplexers (M1 and M2) are used to read register (Intermediate Registers) contents through FSM based generated control signals (S1 and S2). Moreover the multiplexer (M3) is act as a routing purpose which is used to select (using S3 control signal) an appropriate result produced by adder and multiplier units. Finally, the demultiplexer (D1) is used to modify the register contents using S4 control signal generated by FSM. All generated control signals (S1, S2, S3 and S4) are shown as with red color dotted lines in Figure 2.

6.2 Adder

The adder unit is implemented using ‘m’ bit XOR gates as presented in Section 5.1 (Equation (4)). It requires only a single clock cycle.

6.3 Hybrid karatsuba multiplier

The basic principle for multiplication of two ‘m’ bit polynomials is defined in Section 5.2 (Equation 5). However, a bit parallel hybrid karatsuba multiplier is implemented in this work. In karatsuba multiplier, the splitting of two ‘m’ bit polynomial operands ‘a’ and ‘b’ are as follows:

$$a = a_h R^{\frac{m}{2}} + a_l \text{ and } b = b_h R^{\frac{m}{2}} + b_l$$

Where, ‘ a_h ’ and ‘ b_h ’ represents the higher order bits, ‘ a_l ’ and ‘ b_l ’ represents the lower order bits, ‘R’ is the radix and ‘m’ specifies the field length. If ‘m’ is odd, then ‘ a_h ’ and ‘ b_h ’ will be padded with a bit ‘0’ to make all terms with equal size. Based on aforementioned splitting, two ‘m’ bit polynomials multiplication can be accomplished as:

$$a \cdot b = (a_h \cdot b_h) R^m + a_l \cdot b_l + ((a_h + a_l)(b_h + b_l) + (a_h \cdot b_h) + (a_l \cdot b_l)) R^{\frac{m}{2}} \quad (8)$$

The karatsuba multiplier requires divide and conquer approach. The divide and conquer approach is performed in a chronological order i.e., from lower (14 and 15) bits to higher (233) bits as shown in Figure 3.

Figure 3: Hybrid karatsuba multiplier

A hybrid karatsuba approach is utilized in this work by implementing general karatsuba multiplier and simple karatsuba multiplier using Equation (8). The general karatsuba multiplier is suitable over longer bits for minimizing gate counts whereas simple karatsuba multiplier is suitable over smaller bits. Bit parallel hybrid karatsuba requires only a single clock cycle for polynomial multiplication. By taking advantage of bit parallel multiplier (as it requires one clock cycle) hardware resources can be

reduced. However, in order to save hardware resources, polynomial squaring is also performed by providing the same inputs to multiplier unit.

After each polynomial multiplication and squaring reduction is necessary to perform. However, NIST recommended algorithm as presented in Section 5.4 (Algorithm 2) is implemented in this work. Additionally, inversion is computed by implementing Itoh Tsujii algorithm as presented in Section 5.5 (Algorithm 3). Itoh Tsujii requires only frequent squaring's and multiplications as shown in Algorithm 3. However, only multiplier unit (for both squaring and multiplication) is used in this work for inversion computation.

6.4 Intermediate registers

In order to compute PM, a total of 12 registers (Reg_0 to Reg_11) are required. Each particular register contains 'm' bit width and these registers (Reg_0 to Reg_11) are used to hold the intermediate results while implementing Algorithm 1 for PM.

6.5 FSM controller

In order to implement Algorithm 1 for PM, FSM incorporates a total of 71 states. During each state, control signals (S1, S2, S3 and S4) for Algorithm 1 (Step 1, Step 2 and Step 3) are generated.

7. Implementations on FPGA

7.1 Input parameters

As we have discussed in introductory part of this article that, a PB representation we have used. However, for PB representation the input parameters for the proposed architecture is selected from the NIST recommended document [6] and is presented here in Table 2. The column 1 of Table 2 shows the curve parameter and its respective value (in hexadecimal) is presented in column 2.

Table 2: NIST recommended curve parameters for polynomial basis representation of pseudo random ($B - 233$) elliptic curves over binary $GF(2^{233})$ field

Curve parameter	Value
'b'	0x 00000066 647EDE6C 332C7F8C 0923BB58 213B333B 20E9CE42 81FE115F 7D8F90AD
'x' coordinate	0x 000000FA C9DFCBAC 8313BB21 39F1BB75 5FEF65BC 391F8B36 F8F8EB73 71FD558B
'y' coordinate	0x 00000100 6A08A419 03350678 E58528BE BF8A0BEF F867A7CA 36716F7E 01F81052

As shown in Table 2, 'b', 'x' and 'y' are the curve parameters which are required to implement PM operation.

7.2 FPGA

FPGAs are reprogrammable silicon chips which consists of a prebuilt logic blocks and a programmable routing resources [31]. FPGA offers higher flexibility, lower design time, easily available to everyone in the market, and lower non-recurring engineering costs as compared with application specific integrated circuits (ASICs) [11].

7.3 Synthesized results

The proposed architecture is modeled in Verilog (HDL) and synthesized by Xilinx ISE design suite tool (14.2) for newer Virtex 7 (xc7vx690t-3ffg1930) FPGA device. The area results of our proposed architecture is tabulated in Table 3. The column 1 of Table 3 presents the type of resources (Flip Flops, Look up Tables, Slices and Frequency) whereas column 2 and 3 shows the available and utilized resources.

Table 3: Implementation results over Virtex 7

Resource type	Available resources	Utilized resources
Slice Logic Utilization		
Flip Flops (FF)	866400	2886 (1%)
Look Up Tables (LUTs)	433200	21453 (4%)
Slices	108300	11849 (10%)
IO Utilization		
Number of IOBs	1000	703 (70%)
Maximum Frequency	-	157 MHz

As shown in Table 3, the proposed architecture utilizes only 2886 FFs, 21453 LUTs and 11849 slices respectively and it achieves a maximum operational frequency of 157 MHz.

8. Conclusions

Elliptic curve point multiplication architecture is proposed in this work for Lopez Dahab point multiplication algorithm over $GF(2^{233})$. The novel architecture performs both FF squaring and multiplication by using hybrid karatsuba multiplier. In order to perform finite field inversion, Itoh Tsujii algorithm is implemented. The proposed architecture is synthesized and tested over Virtex 7 FPGA by using Xilinx ISE (14.2) design suite tool and it

achieves a maximum operational frequency of 157 MHz with 11849 slices.

9. Acknowledgements

This work was supported by Abasyn University Islamabad, Pakistan and University of Bremen, Germany.

References

- [1] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Communication*, vol. 48, pp. 203–209, 1987.
- [2] V. Miller, "Use of Elliptic Curves in Cryptography," In: Williams H.C. (Eds.): CRYPTO 1985, 218 of the series LNCS, pp. 417–426.
- [3] American National Standard X9.62-1998, "Public Key Cryptography for the Financial Services Industry," 1998. Available at: <https://www.security-audit.com/files/x9-62-09-20-98.pdf>.
- [4] 1363-2000 - IEEE Standard Specifications for Public-Key Cryptography, 2000. Available at: <http://ieeexplore.ieee.org/document/891000/>.
- [5] ISO/IEC 16022, Information Technology-Automatic Identification and data capture techniques-Data Matrix Bar Code Symbology Specification, 2009. Available at: <https://www.iso.org/standard/44230.html>.
- [6] National Institute of Standards and Technology (NIST): Recommended Elliptic Curves for Federal Government Use 1999. Available at: <http://csrc.nist.gov/CryptoToolkit/dss/ecdsa/NISTReCur.pdf>.
- [7] R. L. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, 1978.
- [8] S. M. Bellovin, What is a Cryptosystem, 2006. Available at: <https://www.cs.columbia.edu/~smb/classes/f06/103.pdf>.
- [9] D. Hankerson, A. Menezes, and S. Vanstone, "Guide to Elliptic Curve Cryptography," Springer-Verlag, New York, 2004.
- [10] G. D. Sutter, J. P. Deschamps, and J. L. Imana, "Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 217–225, 2013.
- [11] M. Amara and A. Siad, "Hardware Implementation of Elliptic Curve Point Multiplication over $GF(2^m)$ for ECC Protocol," *International Journal for Information Security Research (IJISR)*, vol. 2, no. 1, pp. 106–112, 2012.
- [12] M. Imran, M. Kashif and M. Rashid, "Hardware Design and Implementation of Scalar Multiplication in Elliptic Curve Cryptography (ECC) over $GF(2^{163})$ on FPGA," *6th IEEE International Conference on Information and Communication Technologies (ICICT)*, Karachi, Pakistan, 2015, pp. 1–4.
- [13] X. Guo, J. Fan, P. Schaumont and I. Verbauwhede, "Programmable and Parallel ECC Coprocessor Architecture: Tradeoffs between Area, Speed and Security," In: C. Clavier and K. Gaj (Eds.): CHES 2009, 5747 of the series LNCS, pp. 289–303.
- [14] J. Han, R. Dou, L. Zeng, S. Wang, Z. Yu and X. Zeng, "A Heterogeneous Multicore Crypto-Processor With Flexible Long-Word-Length Computation," *IEEE Transactions on Circuits and Systems*, vol. 62, no. 5, pp. 1372–1381, 2015.
- [15] M. Rashid, M. Imran and A. R. Jafri, "Comparative analysis of flexible cryptographic implementations," *11th IEEE International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2016, pp. 1–6.
- [16] L. Bossuet, M. Grand, L. Gaspar, V. Fischer and G. Gogniat, "Architectures of Flexible Symmetric Key Cryptoengines-A Survey: From Hardware Coprocessors to Multi-Crypto-Processor System on Chip," *ACM Computing Surveys*, vol. 45, no. 4, 2013.
- [17] A. R. Jafri, M. N. Islam, M. Imran and M. Rashid, "Towards an Optimized Architecture for Unified Binary Huff Curves," *Journal of Circuits, Systems and Architectures*, vol. 26, no. 9, 2017.
- [18] C. Rebeiro, S. S. Roy and D. Mukhopadhyay, "Pushing the limits of High-Speed $GF(2^m)$ Elliptic Curve Scalar Multiplication on FPGA," CHES'12 Proceedings of the 14th international conference on Cryptographic Hardware and Embedded Systems, pp. 494–511, Belgium, 2012.
- [19] Z. A. Benselama, M. A. Bencherif, N. Khorissi and M. A. Bencherchali, "Low Cost Reconfigurable Elliptic Crypto-Hardware", *11th International Conference on Computer Systems and Applications (AICCSA)*, IEEE/ACS, pp. 788–792, Doha, 2014.
- [20] Z. Khan and M. Benaissa, "Throughput/area-efficient ECC processor using Montgomery point multiplication on FPGA," *IEEE Transaction on Circuits and Systems-II*, vol. 62, pp. 1078–1082, 2015.
- [21] D. Johnson and A. Menezes, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," Technical Report CORR 99-34, Dept. of C&O, University of Waterloo, Canada. 2000.
- [22] L. Law, A. Menezes, M. Qu and J. Sloinas, "An Efficient Protocol for Authenticated Key Agreement," Technical Report CORR 98-05, Dept. of C&O, University of Waterloo, Canada. 1998.
- [23] V. Shoup, "A Proposal for an ISO Standard for Public Key Encryption (Version 2.1)," 2001.
- [24] Paul's Online Math Notes, Available at: <http://tutorial.math.lamar.edu/Classes/Alg/SolveExpEqns.aspx>.

- [25] W. Chou, "Elliptic Curve Cryptography and Its Applications to Mobile Devices," Available at: <http://honors.cs.umd.edu/reports/ECCpaper.pdf>.
- [26] National Security Agency-Central Security Service, "The Case for Elliptic Curve Cryptography," Available at: http://www.nsa.gov/business/programs/elliptic_curve.shtml.
- [27] Elliptic Curves–Double and Add Algorithm, Available at: <http://hyperelliptic.blogspot.com/2009/06/double-and-add-algorithm.html>.
- [28] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Mathematics of Computation*, vol. 48, no. 177, pp. 243–264, 1987.
- [29] Y. Zhang, D. Chen, Y. Choi, L. Chen and S. Ko, "A high performance ECC hardware implementation with instruction-level parallelism over GF(2163)," *Microprocessors and Microsystems*, vol. 34, no. 6, pp. 228–236, 2010.
- [30] [30]T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," *Information and Computation*, vol. 78, no. 3, pp. 171–177, 1988.
- [31] [31]National instruments, "Introduction to FPGA technology: Top 5 benefits," Available at: <http://www.ni.com/white-paper/6984/en/>.