High-performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF(2^{163})

Md Selim Hossain, Ehsan Saeedi and Yinan Kong
Department of Engineering, Macquarie University, Sydney, NSW-2109, Australia

Keywords: Elliptic Curve Cryptography (ECC), Finite/Galois Field Arithmetic, Elliptic Curve Point Multiplication (ECPM), Field-programmable Gate Array (FPGA).

Abstract: Elliptic curve cryptography (ECC) plays a vital role in passing secure information among different wireless devices. This paper presents a fast, high-performance hardware implementation of an ECC processor over binary field GF(2^m) using a polynomial basis. A high-performance elliptic curve point multiplier (ECPM) is designed using an efficient finite-field arithmetic unit in affine coordinates, where ECPM is the key operation of an ECC processor. It has been implemented using the National Institute of Standards and Technology (NIST) recommended curves over the field GF(2^{163}). The proposed design is synthesized in field-programmable gate array (FPGA) technology with the VHDL. The delay of ECPM in a modern Xilinx Kintex-7 (28-nm) technology is 1.06 ms at 306.48 MHz. The proposed ECC processor takes a small amount of resources on the FPGA and needs only 2253 slices without using any DSP slices. The proposed design provides nearly 50% better delay performance than recent implementations.

1 INTRODUCTION

With the swift growth of mobile devices and computer applications, cryptography has become a vital tool to ensure the security of data communications and network services. Secret-key cryptography and public-key cryptography (PKC) are two main families of cryptography used for different data-security purposes. ECC (Miller, 1986; Koblitz, 1987) and the RSA cryptosystem (Rivest et al., 1978) are the most popular PKCs. The elliptic curve system as applied to cryptography was first proposed in the mid 80s by Koblitz and Miller. This cryptosystem became popular because it offers equivalent security to the traditional RSA with significantly smaller keys. For instance, 163-bit ECC provides equivalent security to 1024-bit RSA (Koblitz et al., 2000; SEC2, 2000). This feature makes ECC very popular for resource-constrained environments such as pagers, PDAs, cellular phones, smart cards and so on (Sutter et al., 2013). The IEEE (IEEE, 2000) and National Institute of Standards and Technology (NIST) (NIST, 2000) have standardized elliptic curve (EC) parameters for GF(p) and GF(2^m). Certicom has provided NIST-recommended EC domain parameters, standard for efficient cryptography in SEC2 (SEC2, 2000).

Several FPGA-based efficient ECC hardware architectures and elliptic curve cryptographic processors have been presented in the literature (Sutter et al., 2013; Chelton and Benaissa, 2008; Reaz et al., 2012; Hassan and Benaissa, 2010; Machhout et al., 2010; Ghanmy et al., 2014; Shieh et al., 2009; Smyth et al., 2006; Park and Hwang, 2005). In (Ghanmy et al., 2014), Ghanmy proposed ECC processor over GF(2^{163}) on a FPGA platform for wireless sensor networks (WSN). Reaz's design (Reaz et al., 2012) can perform ECC over GF(2^{131}) and GF(2^{163}) on Altera FPGAs. Hasan and Benaissa (Hassan and Benaissa, 2010) implemented their ECC processor using the μ-coding technique on Xilinx Spartan-3 FPGAs over GF(2^{131}), GF(2^{163}), GF(2^{283}) and GF(2^{571}). A coupled FPGA/ASIC implementation of an elliptic curve crypto-processor over GF(2^{163}) is presented in (Machhout et al., 2010), and they used Xilinx Virtex Pro FPGAs and ASIC CMOS 45 nm technology as a hardware platform. Shieh (Shieh et al., 2009), Park et al. (Park and Hwang, 2005) also proposed their ECC processor over a binary field using Xilinx FPGAs. An ASIC-based ECC processor is presented over GF(2^m) in (Smyth et al., 2006).

The optimization aim is generally to reduce the latency of an ECPM in terms of the number of required clock cycles. For this, we have concentrated on efficient algorithms and mathematical reformulations for improving finite-field arithmetic operations which are required for ECPM (Sutter et al., 2013;

Hossain, M., Saeedi, E. and Kong, Y.
High-performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF(2^{163}).
DOI: 10.5220/0005741604150422
In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016)*, pages 415-422
ISBN: 978-989-758-167-0

Chelton and Benaissa, 2008; Kong and Phillips, 2009; Phillips et al., 2010). The arithmetic includes operations defined in finite (Galois) fields, namely GF(p) and GF(2^m) (Hankerson et al., 2003). To the best of the authors' knowledge, there have been few high-speed hardware implementations of an ECC processor in the literature. Thus an efficient design of an ECC processor is still mandatory for modern cryptographic applications.

In this paper an efficient ECC processor is developed in which ECPM operations are achieved in a very low area (around 2.25K slices without using any DSP slices) and latency (almost 50% less than recent implementations). For this, efficient algorithmic reformulations underlying binary finite field and architectural optimization schemes are explored to improve the operating speed. We propose a data-flow architecture of elliptic curve point doubling (ECPD) and elliptic curve point addition (ECPA) that are required for the ECC processor. An efficient field inversion and multiplication algorithms over GF(2^m) are employed to implement high-performance ECPD and ECPA. Finally, an FPGA-based high-performance hardware implementation over GF(2^{163}) is proposed, which is the fastest implementation in an affine coordinate system.

The rest of this paper is organized as follows. Section II introduces a background of groups and fields, Galois finite fields (GF(p) and GF(2^m)), and ECC theories related to this work. Section III describes an efficient finite-field algorithm over GF(2^m), elliptic curve group operations (PD and PA) and hardware architectures. An elliptic curve point multiplication algorithm and cryptographic processor are given in Section IV. FPGA implementation results and comparisons with related designs are given in Section V. Finally this paper is summarized in Section VI.

2 BACKGROUND

In this section, a brief introduction to abstract algebra, field and group theories relevant to ECC designs used in our hardware implementation is presented.

2.1 Groups and Fields

An abelian group $(G,*)$ consists of a set of elements together with a binary operation $*$ which satisfies the following properties :
1. (Associativity) $a*(b*c) = (a*b)*c$ for all $a,b,c \in G$.
2. (Identity) There is an element $e \in G$ such that $a*e = e*a$ for all $a \in G$.
3. (Inverse) for each $a \in G$, there is an element $b \in G$, called the inverse of a, such that $a*b = b*a = e$.
The group operation is generally called addition (+) or multiplication (.). The group is finite if G is a finite set, in which case the number of elements in G is called the order of G.
Fields are abstractions of familiar number systems and their essential properties. A field ($\mathbb{F}$, $+$, $\times$) is a set of numbers $\mathbb{F}$ with two operations, addition and multiplication, satisfying the following properties:
1. ($\mathbb{F}$, +) is an abelian group with (additive) identity 0.
2. ($\mathbb{F} \setminus \{0\}$) is an abelian group with (mult.) identity 1.
Division of field elements is represented in terms of multiplication (mult.): for $a,b \in F$ with $b \neq 0$, $a/b = a.b^{-1}$ where $b.b^{-1} = 1$. (b^{-1} is called the inverse of b) (Hankerson et al., 2003).

2.2 Elliptic Curve Cryptography (ECC)

ECC is the most popular public-key encryption technique. To encrypt data in ECC, it is denoted as a point on an elliptic curve (EC) over a Galois field. A Galois field denoted normally as GF($q = p^m$) is said to be a binary field or characteristic-two finite field if $q = 2^m$. A elliptic curve defined over a Galois field provides a group structure that is used to implement cryptographic systems. The group operations are EC point addition (ECPA) and EC point doubling (ECPD).
There are various coordinate systems to represent elliptic curve points. They vary in the number and type of field operations required to implement PA/PD. In our work, we implement all elliptic curve operations in an affine coordinate system. A non-supersingular elliptic curve E over GF(2^m) in affine coordinates is the set of solutions to the equation

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

where $x,y,a,b \in GF(2^m), b \neq 0$. The coefficients $a,b \in \mathbb{F}_2{}^m$ specifying an elliptic curve $E(\mathbb{F}_2{}^m)$ are defined by the NIST standard and then the elliptic curve is defined by (1). The number of points on an elliptic curve E is represented by $\#E(\mathbb{F}_2{}^m)$. It is defined over $\mathbb{F}_2{}^m$ as nh, where n is the prime order of the curve, and h is an integer called the co-factor.
If $P = (x_1,y_1) \in E$ and $Q = (x_2,y_2) \in E$ (points on the EC), then summing PA and PD can be respectively derived as

$$\begin{aligned} R(x_3,y_3) = P(x_1,y_1) + Q(x_2,y_2) \in E, \\ x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \\ y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \\ \text{where } \lambda = (y_2 + y_1)/(x_2 + x_1) \text{ and } \mathrm{P} \neq \mathrm{Q}; \end{aligned} \tag{2}$$

$$\begin{aligned} R(x_3,y_3) &= 2P(x_1,y_1) \in E, \\ x_3 &= \lambda^2+\lambda+a = x_1^2 + b/x_1^2, \\ y_3 &= x_1^2+\lambda x_3 + x_3, \\ \text{where } \lambda &= x_1+y_1/x_1 \text{ and } \mathrm{P}=\mathrm{Q}; \end{aligned} \tag{3}$$

where $R = 0$ when $x_1 = x_2$ and $y_2 \neq y_1$, or $x_1 = x_2 = 0$. Hence, when $P \neq Q$ we have the PA operation in (2) and when $P = Q$ we have the PD operation in (3). Using these operations, EC point multiplication kP will be implemented using an ECC- based algorithm (Hankerson et al., 2003; Sutter et al., 2013; Miller, 1986; Koblitz, 1987).

Table 1: Comparison of Key length for equivalent security of Symmetric-key and public-key Cryptography (Hankerson et al., 2003).

Symmetric key	Example-algorithm	RSA and DH	ECC in GF(p)	ECC in GF(2^m)
80	SKIPJACK	1024	160	163
112	Triple-DES	2048	224	233
128	AES Small	3072	256	283
192	AES Medium	8192	384	409
256	AES Large	15360	521	571

In 2000, FIPS-2 was recommended with 10 finite fields: 5 prime fields, and 5 binary fields. The binary fields are $\mathbb{F}_2{}^{163}, \mathbb{F}_2{}^{233}, \mathbb{F}_2{}^{283}, \mathbb{F}_2{}^{409}$ and $\mathbb{F}_2{}^{571}$(NIST, 2000). Prime fields GF(p) and binary fields GF(2^m) of similar size are considered to provide almost the same level of security (Koblitz et al., 2000). Table 1 compares symmetric cipher key length, and key lengths for PKC such as RSA, Diffie-Hellman (DH), and ECC (both prime and binary fields). It demonstrates that smaller field sizes can be used in ECC than in RSA and DH systems at a given security level. ECC is many times more efficient than RSA and DH for either private-key operations (such as signature generation and decryption) or public-key operations (such as signature verification and encryption). This makes ECC a promising branch of public-key cryptography (Hankerson et al., 2003).

3 HARDWARE IMPLEMENTATION FOR FINITE FIELD

This section presents all arithmetic algorithms and operations for hardware implementation which are important for ECC. All parameters for NIST elliptic curves over GF(2^{163}) are listed in Table 2. The irreducible polynomial is $f(x) = x^{163}+x^7+x^6+x^3+1$ given for the field GF(2^{163}). A modern Xilinx Kintex-7 (XC7K325T-2FFG900) FPGA with VHDL (VHSIC Hardware Description Language) is used for this hardware implementation. The main components in this ECC design are: polynomial-basis modular addition or field addition, field multiplication, field squaring, field inversion, and elliptic curve group operations (PD and PA).

3.1 Polynomial Basis Representation

A polynomial basis (or standard basis) is an extension field used to represent field elements and is very popular. PB is used in our hardware design for the representation of numbers. For the PB representation, the elements $\mathbb{F}_2{}^m$ are the binary polynomials of degree at most $m-1$, i.e.

$$\begin{aligned} \mathbb{F}_2{}^m &= u_{m-1}.x^{m-1} + u_{m-2}.x^{m-2} + \cdots + u_1.x + u_0 \\ &= \sum_{i=0}^{m-1} u_i x^i : u_i \in \{0,1\} \end{aligned}$$

For instance, x^3+x+1 is a polynomial-basis representation for the 4-bit number 1011_2. For a reduction polynomial or irreducible polynomial, ($f(x)$ be an irreducible binary polynomials of degree m), and $f(x) = x^m + G(x) = x^m + \sum_{i=0}^{m-1} g_i x^i$ where $g_i \in \{0,1\}$ for $i = 1,\ldots,m-1$ and $g_0 = 1$ (Hankerson et al., 2003). For example, $f(x) = x^4+x+1 = 10011_2$ is an irreducible polynomial of the finite field GF(2^4).

Table 2: NIST-recommended elliptic curves over $\mathbb{F}_2{}^{163}$.

K-163: $m = 163,\ f(x) = x^{163}+x^7+x^6+x^3+1,\ a,b = 1, h = 2$
`n=0x 4 00000000 00000000 00020108 A2E0CC0D 99F8A5EF`
`x=0x 2 FE13C053 7BBC11AC AA07D793 DE4E6D5E 5C94EEE8`
`y=0x 2 89070FB0 5D38FF58 321F2E80 0536D538 CCDAA3D9`

3.2 Addition in GF(2^m)

Addition is the simplest operation in GF(2^m). It is simply a bit-wise exclusive-or (xor ($\oplus$)) in either hardware or software. Addition in $\mathbb{F}_2{}^m$ can be achieved as shown in (4) (Wolkerstorfer, 2002):

$$\begin{aligned} Z(x) = U(x)+V(x) &= \sum_{i=0}^{m-1} u_i x^i + \sum_{i=0}^{m-1} v_i x^i \\ &= \sum_{i=0}^{m-1} (u_i+v_i) x^i = \sum_{i=0}^{m-1} z_i x^i \end{aligned} \tag{4}$$

where $z_i = (u_i + v_i) \bmod 2 = u_i \oplus v_i$. The subtraction operation in GF(2^m) is the same as addition because the additive inverse of an element is its identity : $U(x)+U(x) = 0$.

For example, if $U = 1100_2$ and $V = 0110_2$ over the finite field GF(2^4) then $Z = U + V = U \oplus V = (1100_2 \oplus 0110_2) = 1010_2$.

3.3 Multiplication in GF(2^m)

Polynomial multiplication or multiplication in GF(2^m) with the interleaved modular reduction algorithm is a well-known algorithm for hardware implementation (Wolkerstorfer, 2002). It computes the product of two polynomials then applies modular reduction, and its operation is different from simple integer multiplication. Multiplication in $\mathbb{F}_2{}^m$ can be achieved as shown in (5):

$$Z(x) = U(x).V(x) = U(x).\sum_{i=0}^{m-1} v_i.x^i = \sum_{i=0}^{m-1} (U(x).v_i).x^i \tag{5}$$

Multiplication by x^i can easily be calculated by the binary left-shift operation. From polynomial multiplication in algorithm 1, we check whether the result is an element of GF(2^m) with degree $< m$. A modular reduction step is only necessary if the polynomial multiplication result Z_v has degree m or higher. This condition is checked by the $Z_v(m) = 1$ command.

Algorithm 1: Mult. in GF(2^m) with interleaved modular reduction.

```
Input: $U(x), V(x) \in$ GF($2^m$), irreducible polynomials of degree m
Output: $Z(x)$ = $U(x)$ . $V(x)$ mod $f(x)$
    $Z_v$ = 0 ; $U_v$ = '0' & $U(x)$ ;
    for i = $m$ - 1 to 0 do
        if $V(i)$ = '1' then
            $Z_v = Z_v . x + U_v$ ; else $Z_v = Z_v . x$ ;
        end if
        if $Z_v(m)$ = '1' then
            $Z_v = Z_v + f(x)$ ;
        end if
    end for
    Return ($Z(x) = Z_v(m$-1 downto 0)) (At this instance,
     $Z(x)$ is the result of $U(x)$ . $V(x)$ mod $f(x)$)
```

The result of polynomial multiplication $Z(x) = U(x).V(x)$ mod $f(x)$, is achieved after m iterations. Algorithm 1 (Wolkerstorfer, 2002), named multiplication (Mult.) in GF (2^m) with interleaved modular reduction, takes just four steps to find the solution of polynomial multiplication over GF(2^4). The polynomial multiplication result should be reduced to a degree < 4 by irreducible polynomial $f(x) = x^4 + x + 1$.

3.4 Squaring in GF(2^m)

A PB squarer is simpler than and closely related to multiplication. But squaring in GF(2^m) has less difficulty than polynomial multiplication because $U(x)^2$ mod $f(x)$ is a linear operation. It can be computed as shown in (6):

$$Z(x) = U(x)^2 = u_{m-1}.x^{2m-2} + \cdots + u_2.x^4 + u_1.x^2 + u_0$$
$$= \sum_{i=0}^{m-1} u_i x^{2i} \tag{6}$$

The squaring operation in GF(2^m) of $Z(x) = U(x)^2$ is achieved by setting a 0 bit between consecutive bits of the binary representation of $U(x)$ as shown in Figure 1 (Hankerson et al., 2003; Wolkerstorfer, 2002).

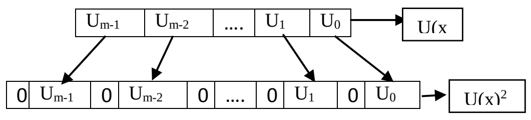

Figure 1: Squaring a binary polynomial $U(x)$.

3.5 Inversion in GF(2^m)

Inversion in GF(2^m) is the most expensive operation for implementing ECC over a binary field. Algorithm 2 computes the field inversion of a non-zero field element $U(x) \in \mathbb{F}_2{}^m$ using the modified Euclidean algorithm (Guo and Wang, 1998). We used this inversion algorithm for our hardware implementation because it is easy to implement on a FPGA.

Algorithm 2: Inversion in GF(2^m) with Modified Euclidean Algorithm.

```
Input: $U(x) \in$ GF($2^m$), irreducible polynomial of degree m
Output: $Z(x) = 1/U(x)$ mod $f(x)$
    $P_v$ = '0' & $U(x)$ ; $Q_v = f(x)$; $Z_v$ = 00001; $V$ = 0 ; $cnt$ = 0 ;
    for i = 1 to 2$m$ do
        if $P_v(m)$ = '0' then
            $P_v = x . P_v$ ; $Z_v = x . Z_v$ ;
            if $Z_v(m)$ = '1' then
                $Z_v = Z_v + f(x)$ ;
            end if
            $cnt = cnt + 1$ ;
        else
            if $Q_v(m)$ = '1' then
                $Q_v = Q_v + P_v$ ; $V = V + Z_v$ mod $f(x)$ ;
            end if
                $Q_v = x . Q_v$ ;
            if $cnt$ = 0 then
                $P_v = Q_v$ ; $Q_v = P_v$ ; ($P_v \leftrightarrow Q_v$)
                $Z_v = V$ ; $V = Z_v$ ; ($Z_v \leftrightarrow V$, exchange operations)
                $Z_v = x . Z_v$ mod $f(x)$; $cnt = cnt + 1$ ;
            else
                $Z_v = Z_v/x$ mod $f(x)$ ; $cnt = cnt - 1$ ;
            end if
        end if
    end for
    Return ($Z(x) = Z_v(m$-1 downto 0)) (At this case,
            $Z(x)$ is the result of $1/U(x)$ mod $f(x)$)
```

The result of field inversion $Z(x) = 1/U(x)$ mod $f(x)$ or multiplicative inversion of $U(x)$ is achieved after $2m$ iterations ($i = 1\,to\,2m$) and the value of cnt is always equal to zero at the end of the last iteration (Guo and Wang, 1998).

3.6 Proposed EC Group Operations

The elliptic curve group operations in GF(2^m) are the PD and PA operations. These are the building blocks of finite-field arithmetic operations such as addition, multiplication, squaring and inversion. Figures 2 and 3 show the data-flow architecture of the proposed ECPD and ECPA operations, corresponding to (2) and (3) respectively. The ECPD operation in affine coordinates requires one field inversion, five field additions, two field multiplications, and two field squarings. Similarly, the ECPA operation in affine coordinates requires one field inversion, eight field additions, two field multiplications, and one field squaring.

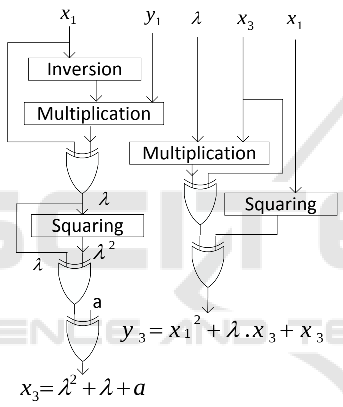

Figure 2: Hardware architecture of the elliptic curve point doubling (ECPD) operation.

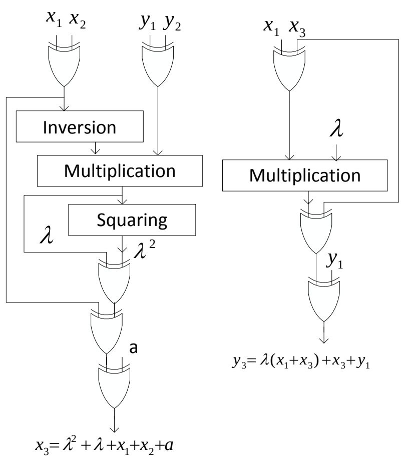

Figure 3: Hardware architecture of the elliptic curve point addition (ECPA) operation.

4 PROPOSED ECPM

Elliptic curve point multiplication (ECPM) is the main operation of an ECC processor; it is computationally the most expensive. However, we have designed a high-performance ECPM using efficient group operations and FFMA units. The building block of an elliptic curve cryptosystem contains ECC protocols such as ECDH (elliptic curve Diffie-Hellman) key exchange, ECDSA (EC digital signature algorithm) at the top level, point multiplication in the second level, group operations in the third level, and field arithmetic operations in the bottom level. The basic operation of ECPM is defined as kP, where k is a positive integer and P is a point on the elliptic curve E defined over a field $\mathbb{F}_{2^m}$. The proposed ECPM architecture over GF(2^m) is presented in Figure 4. Various methods exist for implementing ECPM: the binary method, the Non-adjacent form (NAF) method, and the Montgomery method. The easiest way to implement ECPM is the binary method (left to right) (Hankerson et al., 2003). Finally, we present the ECPM Algorithm 3 using the binary method. It is implemented using the "Double-and-Add" algorithm concept.

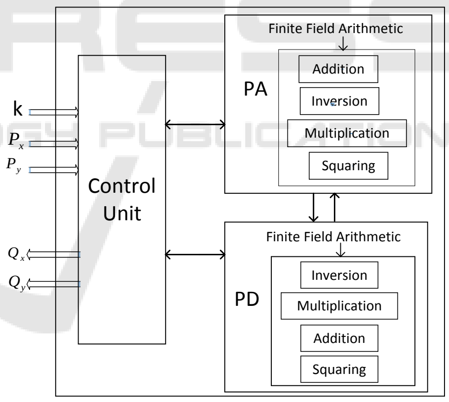

Figure 4: Hardware architecture of Elliptic Curve Point Multiplication (ECPM) processor.

Table 3: Synthesis Results of the finite-field arithmetic for GF(2^{163}) in Kintex-7.

Arithmetic Op*n*	FF	LUTS	LUT-FF (Pairs)	CC	Frequency (MHz)	Time (μs)
Mult./SQ	335	385	335	163	388.83	0.419
Inversion	1479	2007	1315	327	431.71	0.757

Table 4: Elliptic curve Group Operation Results for GF(2^m) in Kintex-7.

Group Operation	FF	LUTs	LUT-FF Pairs	CC	Frequency (MHz)	Time (μs)
PD	3484	4264	3037	1636	331.76	4.930
PA	6587	8347	5664	1636	331.58	4.934

Algorithm 3: Binary method (Left to right) for point multiplication.

```
Input: $k = (k_{m-1,...,k_1,k_0})_2$, $P(x,y) \in E(\mathbb{F}_{2^m})$
Output: $Q(x,y) = k.P(x,y)$, where $Q(x,y), P(x,y) \in E(\mathbb{F}_{2^m})$
    $Q = 0$ ;
    for i = $m$ - 1 to 0 do
        $Q = 2Q$;
        if $k(i)$ = '1' then
            $Q = Q + P$ ;
        end if
    end for
    Return $(Q(x,y))$
```

5 FPGA IMPLEMENTATION RESULTS AND PERFORMANCE ANALYSIS

This section presents the hardware implementation results of this design. We have implemented and tested our design on a modern 28-nm Xilinx Kintex-7 (XC7K325T-2FFG900) FPGA. All VHDL modules are extensively simulated using both Isim and ModelSim, and synthesized using Xilinx ISE 14.7 synthesis technologies.

Table 3 depicts the synthesis results of the finite-field arithmetic operations such as field multiplication/squaring and field inversion over GF(2^{163}). Multiplication or squaring over GF(2^{163}) takes the same area (FF and LUTS), the same number of clock cycles and the same computation time. On the other hand, the clock cycles, flip-flops (FFs), and LUTs (look-up tables) ratio of inversion to multiplications are about 2, 4.45, and 5.2 respectively. Only inversion consumes more clock cycles, area, and timing. The multiplication/squaring (SQ) over GF(2^{163}) is performed in Xilinx Kintex-7 in 419 ns but inversion takes 757 ns. From our implementation results, we notice that field inversion is the most time-consuming operation over the binary field because an inversion takes the same number of clock cycles as 2 multiplications.

The hardware implementation results of proposed elliptic curve group operations are presented in Table 4. The major building block of the elliptic curve group operations (PD and PA) contains addition, multiplication, squaring and inversion. These operations were defined over the binary finite field GF(2^m). The PA operation occupies almost double the area of the PD operation, but the number of clock cycles and the computation time are identical for both operations. The ECPM results for the NIST-recommended field (GF(2^{163}) is shown in Table 6. We achieve a point multiplication in 1.06 ms at a frequency of 306.48 MHz in Xilinx Kintex-7 (XC7K325T-2FFG900) FPGA.

Table 5: Synthesis results for elliptic curve point multiplication (ECPM) over $\mathbb{F}_{2^{163}}$.

Logic Utilization	Used	Available	Used (%)
Numbers of Slice Registers	6620	407600	1
Number of Slice LUTs	7963	203800	3
Numbers of Fully Used LUT-FF Pairs	5712	8871	66
Numbers of BUFG/BUFGCTRLs	2	32	6
Numbers of Bonded IOBs	330	500	66
Numbers of occupied Slices	2253	50950	4

Table 5 represents the summary of estimated values of device utilization. The implemented design over the binary field $\mathbb{F}_{2^{163}}$ takes a small amount of resources on the FPGA. The synthesis report shows that our design is area-efficient as it contains only 2253 slices (4% utilization of total available resources).

The hardware implementation results and performance comparisons with related cryptographic processors are listed in Table 6, which tries to give all the frequencies, number of clock cycles, and the computation time of the designs to make a fair comparison

Table 6: Comparison between our ECC design and related work over GF(2^{163}).

References	Technology	Frequency (MHz)	Clock cycles	Time (ms)
This work	Kintex-7	306.48	325564	1.06
Ghanmy (Ghanmy et al., 2014)	Virtex-II	24	54138	2.26
Reaz (Reaz et al., 2012)	FLEX10KE	43	640700	14.9
Hasan (Hassan and Benaissa, 2010)	Spartan-3	76	205200	2.7
Machhout (Machhout et al., 2010)	Virtex-II	167.84	347425	2.07
Shieh (Shieh et al., 2009)	V1000E	-	-	2.55
Park (Park and Hwang, 2005)	V1000E	44	134090	3.05
Smyth (Smyth et al., 2006)	0.13μmASIC	166	526280	3.17

on the performance between them. An ECC processor over GF(2^{163}) for wireless sensor networks (WSN) is proposed in (Ghanmy et al., 2014), and it requires 2.26 ms to achieve a point multiplication. The ECC processor proposed by Reaz (Reaz et al., 2012) provides a result for the field GF(2^{163}), and their design takes 14.9 ms to compute a point multiplication. Our implemented result is almost 14 times the speed of Reaz (Reaz et al., 2012) but our presented result is not in the same platform. Hasan (Hassan and Benaissa, 2010), Machhout (Machhout et al., 2010), and Sheih (Shieh et al., 2009) implemented ECC processors over GF(2^{163}), and their designs require 2.7 ms, 2.07 ms, and 2.55 ms respectively. Our implemented result is almost double the speed of that of Hasan, Maccout, and Sheih. Park (Park and Hwang, 2005) and Smyth (Smyth et al., 2006) developed ECC processors over GF(2^{163}) in different platforms but their cryptographic processors require more computation time than our design. Our ECC processor over GF(2^{163}) takes 1.06 ms to accomplish a point multiplication. We have also achieved a higher frequency than other cryptographic processors. From the comparison and performance analysis in Table 6, our ECC processor over GF(2^{163}) provides better performance than others.

6 CONCLUSIONS

A high-performance ECC processor over GF(2^{163}) has been implemented using FPGA technology. The binary method (double-and-add) point-multiplication algorithm using an affine coordinate system was used for this hardware implementation. An efficient polynomial-basis multiplication and inversion algorithm was developed for performing elliptic curve PD and PA operations and hence ECC processor. The implemented design is optimized by using different optimization techniques such as balancing the PD and PA architecture, parallelization in operations, and pre-computations for obtaining high performance on an FPGA compared to other designs. In GF(2^{163}), we can achieve a point multiplication in 1.06 ms at 306.48 MHz in Kintex-7 (28-nm) devices, which is the fastest hardware implementation result. The proposed design provides nearly 50% better delay performance than recent implementations. Our implemented design is also area-efficient as it contains only 2253 slices without using any DSP slices. Based on the overall performance analysis and comparisons of different ECC processors over the binary field $\mathbb{F}_{163}$, it can be concluded that this design provides better performance than others in terms of the area and the timing.

REFERENCES

Chelton, W. and Benaissa, M. (2008). Fast elliptic curve cryptography on FPGA. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 16(2):198–205.

Ghanmy, N., Fourati, L. C., and Kamoun, L. (2014). Elliptic curve cryptography for WSN and SPA attacks method for energy evaluation. *Journal of Networks*, 9(11):2943–2950.

Guo, J.-H. and Wang, C.-L. (1998). Systolic array implementation of euclid's algorithm for inversion and division in GF(2^m). *IEEE Trans. Comput.*, 47(10):1161–1167.

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

Hassan, M. N. and Benaissa, M. (2010). Efficient time-area scalable ECC processor using μ-coding technique. In *Third International Workshop, WAIFI, Arithmetic of Finite Fields, LNCS 6087*, pages 250–268.

IEEE (2000). IEEE standard specifications for public-key cryptography. *IEEE Std 1363-2000*, pages 1–228.

Koblitz, N. (1987). Elliptic curve cryptosystems. In *Math. Computation*, volume 48 (177), pages 203–209.

Koblitz, N., Menezes, A., and Vanstone, S. (2000). The state of elliptic curve cryptography. *Des. Codes Cryptography*, 19(2-3):173–193.

Kong, Y. and Phillips, B. (2009). Fast scaling in the residue number system. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(3):443–447.

Machhout, M., Guitouni, Z., Torki, K., Khriji, L., and Tourki, R. (2010). Coupled FPGA/ASIC implementation of elliptic curve crypto-processor. *International Journal of Network Security & its Applications (IJNSA)*, 2(2):100–112.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Lecture Notes in Computer Sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA. Springer-Verlag New York, Inc.

NIST (2000). NIST- National Institute of Standards and Technology, Digital Signature Standard, FIPS Publication 186-2.

Park, J. and Hwang, J.-T. (2005). FPGA and ASIC implementation of ECC processor for security on medical embedded system. In *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) Volume 2 - Volume 02*, ICITA '05, pages 547–551, Washington, DC, USA. IEEE Computer Society.

Phillips, B., Kong, Y., and Lim, Z. (2010). Highly parallel modular multiplication in the residue number system using sum of residues reduction. *Applicable Algebra in Engineering, Communication and Computing*, 21(3):249–255.

Reaz, M. B. I., Jalil, J., Husian, H., and Hasim, F. H. (2012). FPGA implementation of elliptic curve cryptography engine for personal communication systems. *WSEAS Tran. on Circuits and Systems*, 11(3):82–91.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

SEC2 (2000). SEC 2: Recommended elliptic curve domain parameters, standards for efficient cryptography, Certicom Research.

Shieh, M.-D., Chen, J.-H., Lin, W.-C., and Wu, C.-M. (2009). An efficient multiplier/divider design for elliptic curve cryptosystem over GF(2^m). *Journal of Information Science and Engineering*, 25:1555–1553.

Smyth, N., McLoone, M., and McCanny, J. V. (2006). An adaptable and scalable asymmetric cryptographic processor. In *ASAP*, pages 341–346. IEEE Computer Society.

Sutter, G., Deschamps, J., and Imana, J. (2013). Efficient elliptic curve point multiplication using digit-serial binary field operations. *IEEE Transactions on Industrial Electronics*, 60(1):217–225.

Wolkerstorfer, J. (Springer, 2002). Dual-field arithmetic unit for GF(p) and GF(2^m). In *CHES, Lecture Notes in Computer Science*, pages 500–514.