Towards Efficient FPGA Implementation of Elliptic Curve Crypto-Processor for Security in IoT and Embedded Devices

Shaimaa Abu Khadra
Al-Mahala High Institute of Engineering, Al-Mahala

Salah Eldin S. E. Abdulrahman
Dept. of Computer Sci. & Engineering, Faculty of Engineering, Menoufia University.

Nabil A. Ismail
Dept. of Computer Sci. & Engineering, Faculty of Engineering, Menoufia University.

***Abstract*—** An Elliptic Curve Crypto-Processor (ECCP) is a favorite public-key cryptosystem due to its small key size and its high security arithmetic unit. It is applied in constrained devices which often run on batteries and have limited processing, storage capabilities and low power. This research work presents an effective ECCP architecture for security in IoT and embedded devices. A finite field polynomial multiplier takes the most implementation effort of an ECCP because it is the most consuming operation for time and area. So, the objective is to implement the main operation of Point Multiplication (PM) $Q = kP$ using FPGA. The aim is to obtain the optimal registers number for an area optimization of ECCP architecture. Moreover, it proposes a time optimization of ECCP based on the liveness analysis and exploiting forward paths. Also, a comparison between sequential and parallel hardware design of PM based on Montgomery ladder algorithm is provided.

The developed ECCP design is implemented over Galois Fields GF (2^{163}) and GF (2^{409}) on Xilinx Integrated Synthesizes Environment (ISE) Virtex 6 FPGA. In case of GF (2^{163}), this work achieved an area saving that uses 2083 Flip Flops (FFs), 40876 Lookup Tables (LUTs) and 19824 occupied slices. The execution time is 1.963 μs runs at a frequency of 369.529 MHz and consumes 5237.00 mW. In case of GF (2^{409}), this work achieved an area saving that uses 8129 Flip Flops (FFs), 42300 Lookup Tables (LUTs) and 18807 occupied slices. The execution time is 29 μs runs at a frequency of 253.770 MHz and consumes 2 W. The obtained results are highly comparable with other state-of-the-art crypto-processor designs. The developed ECCP is applied as a case study of a cryptography protocol in ATMs.

Keywords:
Elliptic Curve Crypto-Processor, Public-key Cryptosystems, IoT and Embedded Systems Security, Optimal FPGA Implementation.

1. INTRODUCTION

With advanced technology, there is a necessity of secure multiple means of communications and data transmission between devices [1]. The insurance is importance for embedded systems that found everywhere and ranged from light devices to satellites around the earth [2]. Each embedded system composes many devices which connected with each other. Embedded systems contain microprocessors with modern devices technology like WIFI, GSM, GPS, Bluetooth or other devices that used in communication and remote control [3]. The core operation in a public key cryptosystem depends on encrypt data using a public key and decrypt data using a private key.The ECC is one of the public key schemes that use the finite field (Galois Field (GF)) arithmetic to do its operations. Fig. 1 illustrates the ECCP layers architecture. So, it is useful to optimize the finite field operations for ECCPs in order to reduce area and power consumption [8]. The efficiency of the ECC implementation relies on scalar multiplication or Point Multiplication (PM) which is built on group and Finite Field (FF) operations [4]. The field multiplication and field inversion have direct impact on speed and performance of the overall ECCP implementation.

The traditional way of implementing the ECC and the finite field algorithms is software only, running on general-purpose processors, microcontrollers, multicore/manycore (MC) or on digital-signal processors [4].In practice, the decisive encounter of software executions of ECC is the latency [9] or potential owing to the word close calculations essential and recurrent memory processes. Various ECC software implementation on MC architectures have been presented to enhance ECC performance by improving algorithms primary the PM methods. For example, Albahri et. al. [9] proposed an ECC PM over GF(2^{163}) on Xmos kit IDE founded on executing a vertical parallelization on finite field Point Doubling (PD) and Point Addition (PA) group steps. They also proposed a modification to the left-to-right double and add binary PM to eliminate data hazards. Nevertheless, the desired small key size, low area, lower memory requirements, faster encryption and decryption, less power consumption, and lower bandwidth necessities recommend ECC for hardware cryptosystem processor synthesizes. This work is concerned with the implementation of an efficient ECCP with a special scalar multiplier, fully pipelined, parallel, and self-controlled architecture. Its intention is to enhance the ECCP hardware design targeted for resource constrained, IoT, and embedded devices.

The proposed ECCP hardware design presented in this work is focused over binary Galois fields GF(2^{163}) and GF(2^{409}). There are numerous algorithms that have been reported to implement PM, some of them are designed to reduce area and others to save the ECCP time. The main contribution of this work may be stated in the following points:

- Implementing algorithms such as Montgomery [13-17], Itoh-Tsujii [12, 15] and Karatsuba [18-21] for optimizing EC and finite field operations to improve

speed, throughput, and/or computation time. The implementation is performed over $GF2^{163}$ and $GF2^{409}$

- Optimizing area requirements by minimizing the Register File (RF) size in ECCP.
- Presenting a low-power method to the proposal of embedded ECCP architecture that achieves the bypassing and substitutes the power cost of transfer short-lived variables to/from RF.
- Design and apply the technique for N time squarer $(x^2)^N$ for $GF2^{163}$ and $GF2^{409}$ and compared with other techniques which depend on using glitch free clock switch.
- Comparison between sequential and parallel architecture based on liveness analysis and forward path in the implementation of ECCP PM algorithm.

This research paper is planned as follows: the next section starts a mathematical preliminary that include arithmetic procedures for binary Galois fields. It also presents elliptic curve arithmetic algorithms, a brief introduction on scalar multiplication, projective representation and PM costs. The state-of-the-art is existing in section three. Section four provides area/time optimization approach. The implementation results and discussion are detailed in section five. Section six provides a case study of cryptography protocol. The conclusion and future direction are described in section seven.

2. ECC ARITHMETIC PRELIMINARY

2.1 BINARY GALOIS FIELD(2^m)ARITHMETIC

This section focuses on Galois fields of order 2^m arithmetic ($GF(2^m)$) which is endorsed byNISTfor EC Digital Signature Algorithm (ECDSA) application [2]. A generic standard binary ECcan be represented as:

$$\boldsymbol{E\left(GF(2^m)\right): y^2 + xy = x^3 + ax^2 + b} \quad \textbf{(1)}$$

where a, b are in $GF(2^m)$ and$b \neq 0$.

For special cases when **m** = 163 or 409, the hardware circuit for performing addition needs exactly 163 or 409 XOR gates respectively.

An ECCP is designed and build based on the main operation PM ($Q = k\,P = P + P + \cdots + P$) and it can be done using repeated Point Addition (PA) and Point Doubling (PD), for example $11P = 2(5P) + P = 2(2(2P) + P) + P$. The Lopez-Dahab (LD) projective coordinates are used to calculate the PM of the binary ECCP in Eq. (1).

All operations of the PM require finite field operations like inversion, squarer, polynomial multiplier and addition [4]. This work adopted with $GF(2^{163})$ and $GF(2^{409})$ binary fields. The $GF(2^m)$ is more suitable for hardware design in which the addition operation requires only XOR unit and eliminates the need for carry propagation. The square operation is done with no area and it is meant by inserting zero between bits. A polynomial multiplier may be implemented in a bit serial or a bit parallel multiplier [11]. A bit serial multiplier is a good choice for area but a bit parallel is a good choice for time. Both the polynomial squarer and the polynomial multiplier are needed to follow with irreducible polynomial. The irreducible polynomial for a(z) mod p(z) was meant by the reminder of a long division of a(z) by p(z).The irreducible polynomial was implemented in hardware by using shift and XOR operation. Finally, the inversion operation is the most complicated unit as it takes a large area and has a slow implementation. The Itoh-Tsujii [12] algorithm is one of the inversion algorithms that convert the inversion operation to run based on two finite field units which are multiplier and squarer units. The inversion operation takes only nine multiplications for $GF(2^{163})$ or ten multiplications for $GF(2^{409})$ and (m-1) repeated squaring operations.

FIELD ADDITION OPERATION

In a binary Galois Field $GF(2^m)$ or ($\mathbb{F}_{2^m}$) the polynomial addition operation is executed bitwise with no carry propagation (Exclusive-OR or XOR). For example, let A and B are represented as:

$$\boldsymbol{A(x) = \sum_{i=0}^{m-1} a_i x^i, B(x) = \sum_{i=0}^{m-1} b_i x^i.}$$

Then,

$$\boldsymbol{C(x) = \sum_{i=0}^{m-1} c_i x^i = A(x) + B(x) = \sum_{i=0}^{m-1} \left((a_i + b_i) mod\ 2\right) x^i} \quad \textbf{(2)}$$

FIELD SQUARING OPERATION

Squaring a field element in $\mathbb{F}_{2^m}$ represented via a polynomial basis, $\{1, x, x^2, \ldots, x^{m-1}\}$, is ruled by the following equation.

$$\boldsymbol{A^2(x) = \left(\sum_{i=0}^{m-1} A_i x^i\right)^2 = \sum_{i=0}^{m-1} A_i x^{2i}\ mod\ F(x)}$$

Hence, $A^2(x)$in $\mathbb{F}_{2^{163}}$, for example, is

$$\boldsymbol{A^2(x) = \underbrace{(a_{162}x^{160} + \cdots + a_{83}x^2 + a_{82})x^{164} mod\ F(x)}_{A_H} + \underbrace{(a_{81}x^{162} + \cdots + a_1x^2 + a_0)}_{A_L}} \quad \textbf{(3)}$$

comprises three mathematical stages: (i) expand A_L part with interleaved 0's; (ii) reduce the A_H part with the reduction polynomial

$$\boldsymbol{F(x) = x^{163} + x^7 + x^6 + x^3 + 1;}$$

and (iii) add the two parts: A_H, A_L. However, for hardware implementations, the use of Eq. (3) enables these three steps to be combined in four level XOR gates if the reduction polynomial has a smaller second-highest degree, which is the case here.

FIELD MULTIPLICATION OPERATION

A few polynomial basis bit-serial, digit-serial, and bit-parallel finite field multipliers have been proposed in [4]. The multiplication of two binary polynomials A and B may be simplified as:

$$\boldsymbol{C(x) = A(x) \cdot B(x) mod\ F(x) = A(x)\left(\sum_{i=0}^{m-1} b_i x^i\right) mod\ F(x) = \left(\sum_{i=0}^{m-1} b_i A(x) x^i\right) mod\ F(x)}$$

where $F(x)$ is the reduction binary polynomial of degree m.Therefore,

$$\boldsymbol{C(x) = (b_0 A(x) + b_1 A(x)x + b_2 A(x)x^2 + \cdots}$$

$$+\boldsymbol{b_{m-1}A(x)x^{m-1})mod\ F(x)} \quad \textbf{(4)}$$

Algorithm 1 shows a well-suited procedure for hardware implementation of Eq. (4) when area is constrained. The bit-parallel type multipliers are best suited for applications that require fewer clock cycles, i.e. high speed but more space.

Algorithm 1 shows that field multiplication in $GF(2^m)$ is computed by mainly two operations; binary polynomial multiplication $\dot{C} = A(x) \cdot B(x)$ and $mod\ F(x)$ reduction polynomial $F(x)$. This work adopts binary Karatsuba-Ofman [18, 19].

REDUCTION OPERATION

The reduction process is required by the square and multiplication operations as depicted in Eq. (3) and Eq. (4) respectively. The outcome of the reduction process is to decrease the order of subsequent values from greater than m to fewer or equal to m. $\boldsymbol{C(x) = \dot{C}(x)\ mod\ F(x)}$

KARATSUBA-OFMAN MULTIPLIER [18 - 19]

Binary Karatsuba-Ofman multiplier was introduced in [18, 19] which is appropriate for $GF(2^m)$ where m is a predetermined number. Let A, B two elements in $\mathbb{F}_{2^m}$ where:

$$\boldsymbol{A = \sum_{i=0}^{m-1} a_i x^i = \sum_{i=0}^{\frac{m}{2}-1} a_i x^i}$$

$$\boldsymbol{+ \sum_{i=\frac{m}{2}}^{m-1} a_i x^i = x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} a_i x^i = x^{\frac{m}{2}} A^H + A^L}$$

and

$$\boldsymbol{B = \sum_{i=0}^{m-1} b_i x^i = \sum_{i=0}^{\frac{m}{2}-1} b_i x^i + \sum_{i=\frac{m}{2}}^{m-1} b_i x^i}$$

$$\boldsymbol{= x^{\frac{m}{2}} \sum_{i=0}^{\frac{m}{2}-1} b_{i+\frac{m}{2}} x^i + \sum_{i=0}^{\frac{m}{2}-1} b_i x^i = x^{\frac{m}{2}} B^H + B^L}$$

The polynomial product is given as

$$\boldsymbol{C = x^m A^H B^H + (A^H B^L + A^L B^H) x^{\frac{m}{2}} + A^L B^L} \quad \textbf{(5)}$$

$$\boldsymbol{C = x^m A^H B^H + A^L B^L + (A^H B^H + A^L B^L + (A^H + A^L)(B^L + B^H)) x^{\frac{m}{2}}} \quad \textbf{(6)}$$

$$\boldsymbol{C = x^m C^H + C^L}$$

Algorithm 2 shows a modified binary Karatsuba multiplier [19].

2.2 ELLIPTIC CURVE ARITHMETIC

Arithmetic of EC is defined in terms of underlying field operations [4] which are described in section 2.1. In its simplest form a point on an EC, in affine coordinate, is distinct as (x, y) of $GF(2^m)$ that satisfying Eq. (1) with $b = 1$. Since Point Addition (PA) algorithms used in this work are explicitly involve a without b, so we do not have to store b. However, Point Doubling (PD) is a special case of point addition. Point Multiplication (PM), e.g., $Q = kP$ is the accumulation of k duplicates of point P hence:

$$\boldsymbol{Q = kP = \underbrace{P + P + \cdots + P}_{k\ copies}}$$

For a large k, PM can be done using repetitive PAs and PDs.

POINT MULTIPLICATION (PM) [4, 20 - 23]

Elliptic curve points can be represented using various coordinate systems such as affine or projective representations [21, 22]. For each such system, the speed of PAs and PDs is different [22]. An EC point in projective coordinate is represented in $GF(2^m)$ as $P = (X, Y, Z)$. To convert the affine point (x, y) to projective coordinates, Z is simply set to 1, i.e., $(x, y, 1)$ [21]. Projective coordinates are efficient for inner computations, however, a conversion operation from projective to affine is needed before the transmission step. The costs of addition, squaring, multiplication, and inversion operations in $\mathbb{F}_{2^m}$ for various coordinates are listed in Table 1 [23].

This work adopts the projective coordinates presented by Lopez-Dahab (LD) [14]. According to Lopez-Dahab projective point representation $P = (X, Y, Z), Z \neq 0$ adapts the affine representation $P = (X/Z, Y/Z^2)$. Hence, in projective coordinates, the binary EC curve in Eq. (1) is attained by substituting x and y with X/Z and Y/Z^2 [14]. Algorithm 3 is the right-to-left type of the elementary repetitive double/add method for point multiplication [4] with the aid of LD method for PA (Algorithm 4) and PD (Algorithm 5).

LOBEZ-DAHAB (LD) EXPLICIT FORMULAS [17, 22]

In Lopez-Dahab (LD) projective point [17, 22] coordinates Algorithm 4 and Algorithm 5 provide ECC explicit intermediate variables specified formulas for both PA and PD group operations, respectively.

As a means of avoiding an expensive field inversion operation, it is convenient to work with the projective coordinates presented by LD. Therefore, for a given k and m the overheads of implementing a scalar multiplication using Algorithm 3 [21] is approximated as:

$$\boldsymbol{N_{binary} = mN_{double} + (m/2)N_{add}}$$

BASIC POINT MULTIPLICATION COSTS

Approximations for point multiplication costs are existing in terms of curve operations (PAs and PDs) [4], and the equivalent field operations (MUL, SQR, ADD and INV). Convert from affine to projective costs 2SQR + 1ADD. Point addition and doubling in projective coordinates cost 14MUL + 4SQR + 9ADD and 5MUL + 4SQR + 5ADD operation counts respectively. The overall overhead of coordinates conversion from projective to its corresponding affine [21] may be 2MUL + 1INV + 1SQR.

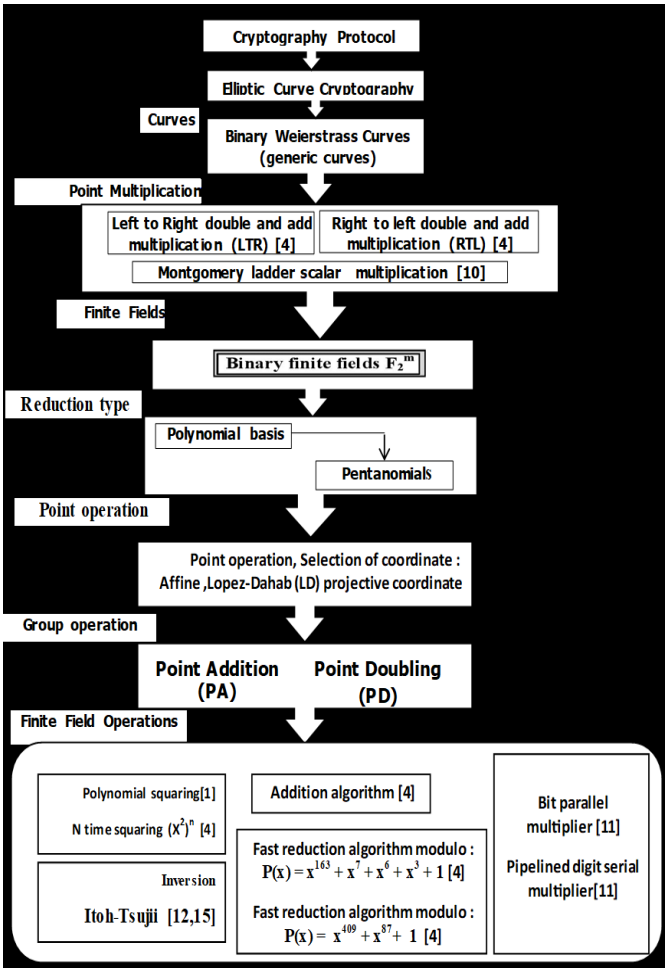

Fig. 1: The PM of ECCP Layers architecture.

Algorithm 1 bit-serial binary polynomial multiplication (low to high) in $\mathbb{F}_{2^m}$ [4], [20]

Input: Two elements $A(x)$ *and* $B(x) \in \mathbb{F}_{2^m}$, a reduction polynomial $F(x)$ of m degree,

Output: $C(x) \in \mathbb{F}_{2^m}$ where $C(x) = A(x) \cdot B(x) \bmod F(x)$.

$$C(x) = 0$$

for $i = 0$ to $m - 1$ do

$$C(x) = b_i A(x) + C(x)$$
$$A(x) = A(x) \cdot x \; mod \; F(x)$$

end-for

return $C(x) \; mod \; F(x)$ //Reduction operation

Algorithm 2 Modified binary Karatsuba multiplier in $GF(2^m)$ [18]

Input: Two elements $A, B \in \mathbb{F}_{2^m}$ with an arbitrary m, where $A = x^{\frac{m}{2}} A^H + A^L$, $B = x^{\frac{m}{2}} B^H + B^L$

Output: $C = AB$ a polynomial of2m-1 coordinates, and $C = x^m C^H + C^L$

procedure BK (C, A, B)

begin

$\ell = [\log_2 m]$

$h = m - 2^{\ell}$;

if $(h == 0)$ then C = Kmul 2^{ℓ} (A, B) return;

for i from 0 to h -1 do

$$M_{A_i} = A_i^L + A_i^H$$
$$M_{B_i} = B_i^L + B_i^H$$

end;

mul$2^{\ell}(C^L, A^L, B^L)$;

mul$2^{\ell}(M, M_A, M_B)$;

BK(C^H, A^H, B^H);

for i from 0 to 2ℓ -2 do$M_i = M_i + C_i^L + C_i^H$ end;

for i from 0 to 2ℓ -2 do$C_{\ell+1} = C_{\ell+1} + M_i$ end;

end;

Table 1: Cost of PM operations [23]

Operation	Coordinates	Cost			
		Add(A)	Mul(M)	Inv (I)	Sqr (S)
PA	$\mathcal{A}$	8	2	1	1
PD	$\mathcal{A}$	6	2	1	1
Negation	$\mathcal{A}$	1	0	0	0
PA	$\mathcal{P}$	8	16	0	2
PD	$\mathcal{P}$	5	8	0	4
Negation	$\mathcal{P}$	1	0	0	0
PA	$\mathcal{LD}$	9	13	0	4
PD	$\mathcal{LD}$	5	5	0	4
Negation	$\mathcal{LD}$	1	1	0	0
Montgomery	$\mathcal{P}$	3	6	0	4
Montgomery	$\mathcal{P} \mapsto \mathcal{A}$	6	10	1	6
Mapping	$\mathcal{P} \mapsto \mathcal{A}$	0	2	1	0
Mapping	$\mathcal{LD} \mapsto \mathcal{A}$	0	2	1	0

Algorithm 3 Bit-serial scalar multiplication [4] using LD coordinates for PA and PD [14].

Input: $P \in E\big(GF(2^m)\big), k = \sum_{i=0}^{t} k_i 2^i$
Output: kP
1: $Q \leftarrow \infty$
2: **for** i = 0 to t - 1 do
  2.1 **if** $k_i = 1$ then
  2.2 $Q \leftarrow Q + P$
  2.3 **end if**
  2.4 $P \leftarrow 2P$
3: **end for**
4: return $Q = kP$

Algorithm 4 Lobez-Dahab projective coordinate for point addition (PA) in $GF(2^m)$.

Input: $P = (X_1, Y_1, Z_1)$ and $Q = (X_2, Y_2, Z_2)$ in LD coordinates on
$E\big(GF(2^m)\big): y^2 + xy = x^3 + ax^2 + b, b \neq 0$
Output: $P + Q = (X_3, Y_3, Z_3)$ in LD coordinates.
1: $A = X_1 * Z_2$
2: $B = X_2 * Z_1$
3: $C = A^2$
4: $D = B^2$
5: $E = A + B$
6: $F = C + D$
7: $G = Y_1 * Z_2^2$
8: $H = Y_1 * Z_1^2$
9: $I = G + H$
10: $J = I * E$
11: $Z_3 = F * Z_1 * Z_2$
12: $X_3 = A * (H + D) + B * (C + G)$
13: $Y_3 = (A * J + F * G) * F + (J + Z_3) * X_3$

Algorithm 5 Lobez-Dahab projective coordinate for point doubling (PD) in $GF(2^m)$.

Input: $P = (X_1, Y_1, Z_1)$ in LD coordinates on
$E\big(GF(2^m)\big): y^2 + xy = x^3 + ax^2 + b, b \neq 0$
Output: $2P = (X_3, Y_3, Z_3)$ in LD coordinates.
1: $A = Y_2 * Z_1^2 + Y_1$
2: $B = X_1 + X_2 * Z_1$
3: $C = B * Z_1$
4: $Z_3 = C^2$
5: $D = B^2(C + a * Z_1^2$
6: $E = A * C$
7: $X_3 = A^2 + D + E$
8: $F = X_3 + X_2 Z_3$
9: $G = (X_2 + Y_2) * Z_3^2$
10: $Y_3 = (E + Z_3) * F + G$

Algorithm 6 Point multiplication using Montgomery ladder [10, 17].

Input: $k = (k_{m-1}, k_{m-2}, \dots, k_2, k_1, k_0)_2$ with $k_{m-1} = 1,\ P = (x, y) \in E\left(GF(2^m)\right)$
Output: $Q = (x_k, y_k) = k \cdot P$
1- Set $X_1 = x, Z_1 = 1,\ X_2 = x^4 + b,\ Z_2 = x^2$
2- for $i = m - 2$ to 0 do
    2-1 If $k_i = 1$ then
        $(X_1, Z_1) =$ Add (X_1, Z_1, X_2, Z_2, x);
        $(X_2, Z_2) =$ Double (X_2, Z_2, b);
    2-2 else
    $(X_2, Z_2) =$ Add (X_2, Z_2, X_1, Z_1, x);
    $(X_1, Z_1) =$ Double (X_1, Z_1, b);
        end if;
end for;
3- Convert from projective to affine

$$x_k = \frac{X_1}{Z_1} = \frac{xZ_2X_1}{xZ_1Z_2} \quad (7)$$

$$y_k = \frac{(x+x_k)[(X_1+xZ_1)(X_2+xZ_2)+(x^2+y)Z_1Z_2]}{xZ_1Z_2} + y \quad (8)$$

4- Return Q

Algorithm 7 Inversion using Itoh-Tsujii algorithm (ITA) in $\mathbb{F}_{2^m}$

Input: $A = (xZ_1Z_2) \in GF(2^m)$
Output: A^{-1}
1: $r \leftarrow (2^m - 1)$
2: $A^r = A^{r-1}.A$
3: $A^{-1} = (A^r)^{-1}.A^{r-1}$
4: return A^{-1}

Table 2 PM state-of-the-art

Implementation	ECCP Optimization	Point (Scalar) Multiplication	FPGA
M. Imran *et. al*., 2018 [7]	Three architectures-based execution time vs reliability-security algorithm	Modified PA and PD for Montgomery PM	Virtex 5, Virtex 6, and Virtex 7.
B. Rashidi *et. al.* 2016 [18]	Parallel Low critical data path delay	Montgomery ladder GF(2^{163}) & GF(2^{233})	Virtex 5
M.S. Albahri*et. al.* 2016 [9]	Parallel on multi-core microcontroller	Modified Left-to-right binary PM in L-D	Xmos multi-core microcontroller
Z. Khan et.al. 2015 [25]	Efficient memory unit. Pipelined digit-serial multiplier Parallelization	LD modified Montgomery PM algorithm	1476 slices on Virtex 4 LX2512
Lijuan and Shuguo, 2016 [28]	Modified Montgomery ladder Pipeline	Three-stage pipeline	Virtex 4 XC4
B. Rashidi et. al. IET 2017 [12] P. Zode et. al. 2017 [29]	Tree structure		Virtex 4 ITA time 0.262 us for GF(2^{163})
Khan and Benaissa 2017 [25]	1 and 2 stages pipelined full-precision n	Modified L-D Montgomery ladder	Virtex 4, Virtex 5, Virtex 7
Loi&Ko 2016 [10]	Parallel FFAU and ECPM on Koblitz	Parallelization of L-D ECPM for Koblitz curves	Virtex 5

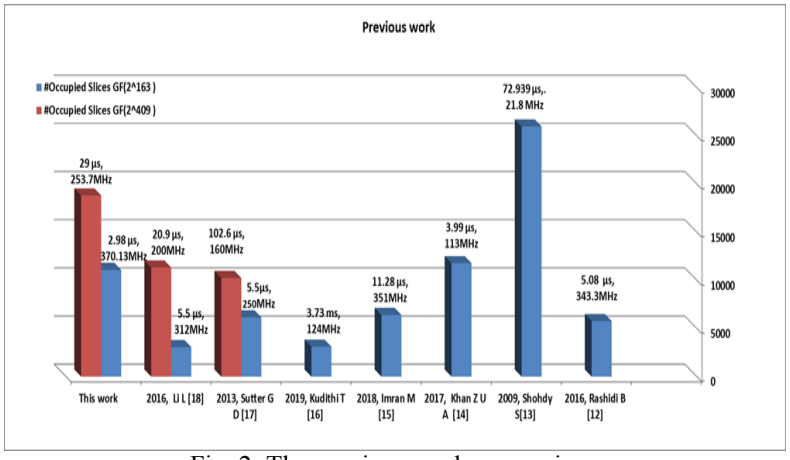

Fig. 2: The previous work comparison

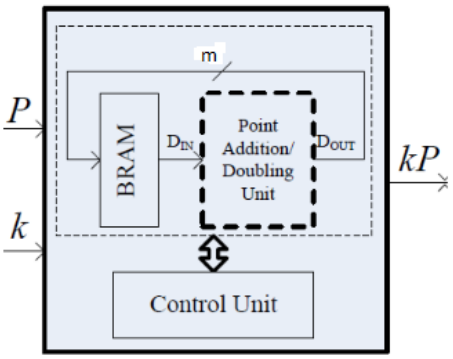

Fig. 3: A top-level architecture for ECCP

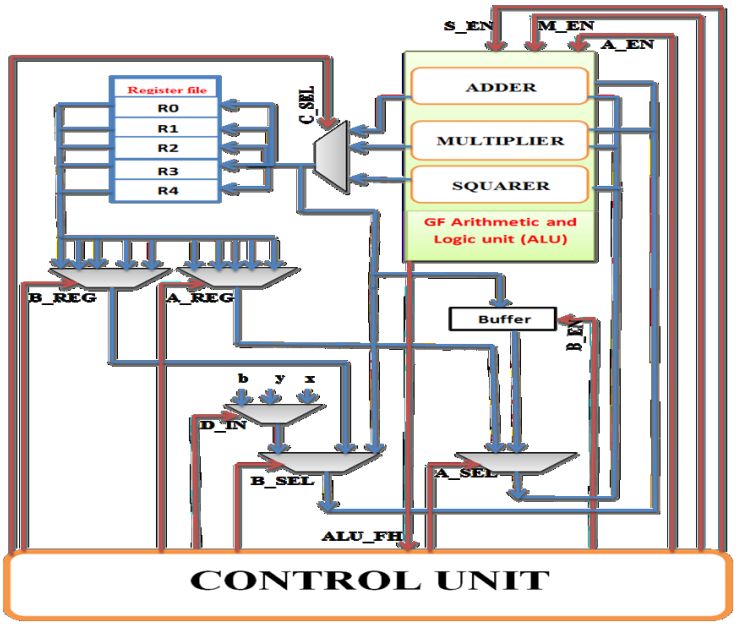

Fig. 4: The proposed ECCP architecture design

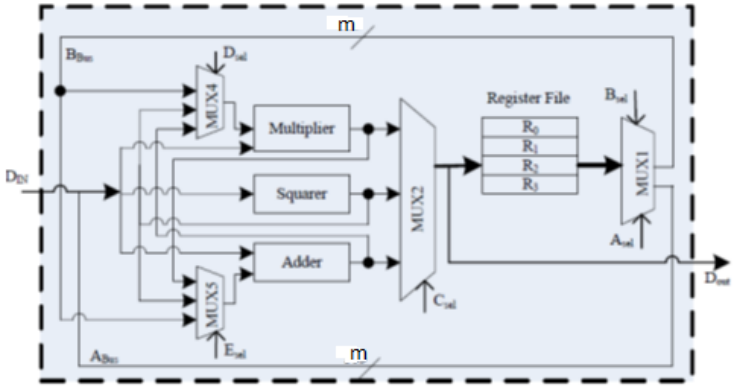

Fig. 5: Optimized Point Addition/Doubling unit

POINT MULTIPLICATIONUSING MONTGOMERY LADDER [10, 22]

One of the most efficient point multiplication which is based on LD projective PA and PD group operations is Montgomery ladder scalar multiplication [10, 22]. Algorithm 6 depicts an EC scalar multiplication method using Montgomery ladder where the curve point $P = (x, y)$, $Q = kP$ represented by (x_k, y_k) . PA $(Z_a, X_a) = \mathrm{Add}(X_1, Z_1, X_2, Z_2, x)$ is given by:

$\mathbf{Z_a = (X_1 Z_2 + X_2 Z_1)^2}$ and

$\mathbf{X_a = xZ_a + (X_1 Z_2)(X_2 Z_1)}$

and for P $\mathbf{(X_d, Z_d) = double(X_i, Z_i, b)}$

we have: $\mathbf{Z_d = X_i^2 Z_i^2}$ **and** $\mathbf{X_d = X_i^4 + bZ_i^4}$

Algorithm 6 shows an EC point multiplication using the Montgomery ladder method [17]. In algorithm 6 only one inversion operation, $\mathrm{INV}(xZ_1Z_2) = (xZ_1Z_2)^{-1}$ is required.

INVERSION ALGORITHMUSING ITOH-TSUJII (ITA)

In Eq. 7 and Eq. 8 the inversion $(xZ_1Z_2)^{-1}$ is employed using Itoh-Tsujii algorithm (ITA) reported in [12, 15, 17, 19]. The hardware implementation of the inversion inside conversion step in several research work has often use ITA for the point multiplication which is adopted in this work as well. ITA is summarized in Algorithm 7:

3. RELATED WORK

Several FPGA based hardware implementations for elliptic curve cryptosystem processor (ECCP) were suggested in literature [17 - 27] few of them aimed for low-end devices [22]. However, there is an equally important need for stand-alone ECCP engines in small constrained devices, like sensor networks and mobile devices [14]. Most implementations focus on minimizing speed and area of the PM on binary EC [17]. A pipeline digit-serial GF multiplier was used in Montgomery ladder GF(2^{163}) & GF(2^{233}) PM design by Rashidi et.al. [17]. El-Sisi et. al. [18] used a modified binary Karatsuba-OfmanGF(2^{191}) multiplier with Montgomery PM in projective coordinate. Ismail [22] implemented Left-to-right binary PM in Lopez-Dahab (LD) projective coordinate with digit-serial. Ansari & Hasan [25] implemented Montgomery ladder PM with pseudo-pipelined word-serial. Khan &Benaissa [26] proposed an ECC architecture with one and three pipelined multipliers based on a modified LD Montgomery PM. Imran et. al. [27] presented a modified PA and PD for PM based on Montgomery algorithm. Different FPGA based hardware implementations of the point multiplication on binary elliptic curve have been summarized as depicted in Table 2.

In [30], the parallel architecture depends on using the Montgomery ladder for field GF(2^{191}) but with Karatsuba-Ofman for the polynomial multiplier and Extended Euclidian Algorithm (EEA) for the inversion. However, the work in [31] presented an ECCP architecture design for IoT securityusing the Montgomery ladder algorithm running sequentially. All previous works results appeared in Fig.2.

4. AREA/TIME OPTIMIZATION

4.1 THE PROPOSED ECCP DESIGN

This work has implemented all the algorithms that have designated in the preceding sections to design the proposed ECCP. This can help to use only one functional unit of the GF(2^m) ALU of the ECCP as shown in Fig. 3; where **m** may be 163 or 409.

The top-level architecture consists of a control unit, Block RAM (BRAM), and a point addition/doubling unit as shown in Fig. 4. The control unit receives the EC parameters, reads a key (or a scalar), and controls the point addition/doubling unit according to the binary double and add point multiplication algorithm shown in 2.2. Design point addition/doubling unit, on the other hand, is responsible for computing all required field arithmetic operations. We assume that at the beginning of the scalar multiplication operation the BRAM contains the scalar and a projective EC point as it considers fast access FPGAs read/write memory. These values must be maintained during the iterations of EC scalar multiplication. The point addition/doubling unit designed for computing the scalar multiplication algorithms. Fig. 4 shows architecture for the proposed ECCP design.

4.2 OPTIMIZING ADDITION/DOUBLING UNITVIA FORWARDING PATHS

An optimal PA/PD unit is illustrated in Fig. 5. Its main three units which constitute the GF(2^m) addition, multiplication, and squaring operations.

In addition to the two multiplexers (MUX3) and (MUX4) used to accomplish the forward paths from one finite arithmetic unit to another. These multiplexers are controlled by two signals D_{Sel} and E_{Sel}. The operands are stored in the register file which consists of four to six registers with output being selected for A, and B using multiplexer (MUX1) with control signals (A_{sel}, and B_{sel}). These signals addressed from the control unit shown in Fig. 4. Moreover, the proposed ECCP design aims to obtain efficient registers number for various explicit formulas appearing in [22] for low-area ECCP design. The forward liveness analysis methodology [22] main emphasis is to expose a forward data-path entity that take place between the functional units (e.g. finite field squarer (S) multiplier (M) and adder (A)) and the register file. In theory, any path from root to leaf states gives a valid sequence requiring a minimum number of registers to complete execution [26]. Also, an effective execution order which achieves a maximum amount of short live variables to efficiently use forward path.

5. IMPLEMENTATION RESULTSAND COMPARISON

The developed ECCP over GF(2^{163}) and GF(2^{409}) system has been implemented entirely in RTL-level VHDL and Integrated Synthesizing Environment (ISE). For fair comparison with the other architectures presented in the literature [27], the code has been synthesized and implemented on FPGAs using Virtex-6 XC6VLX760.

5.1 Implementation Results

Squaringand Reduction Operations:
The squaring and reduction implementation result over GF(2^{163}) and over GF(2^{409}) are shown in Table 3.

Polynomial Multipliers: Bit Parallel
A GF(2^{163}) and a GF(2^{409}) binary polynomial multipliers based on bit parallel digit-serial have been proposed in this architecture, see Table 4.

ModifiedRight-to-Left GF(2^{163}) and GF(2^{409}) Multiplications
A parallel implementation using Karatsuba-Ofman to reduce the latency is shown in Table 5 (time optimization).
Pipeline technique in GF(2^{163}) and in GF(2^{409}) polynomial multipliers (area optimization) are illustrated in Table 6.

Inversion

Fig. 6 illustrates the implementation of the inversion ITA [17].

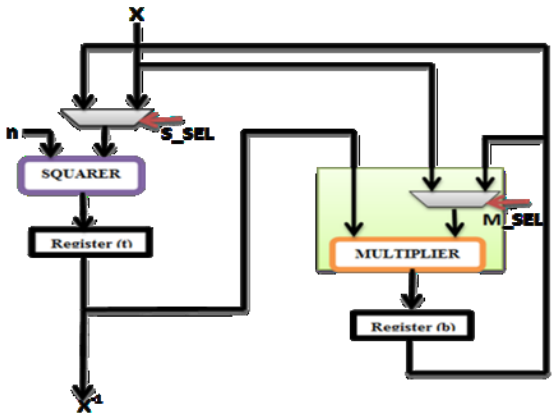

Fig. 6: Itoh-Tsujii architecture

PA & PD in ProjectiveCoordinate:

Fig. 7 shows PA & PD architecture.Architecture with one and three pipelined multipliers based on a modified LD Montgomery PM is illustrated in Table 7.

Scalar Multiplicationin Projective Coordinate:

Fig. 8 shows the implementation of scalar multiplication architecture.

Parallel Implementation of Montgomery Ladder

The proposed ECCP depend on the parallel implementation for the Montgomery scalar algorithm which uses 3 unit for polynomial multiplier. The operation of point addition and point doubling take two stage to complete in this proposed architecture. Table 8 shows the results of Montgomery parallel implementation.

5.2 ComparisonwithState-of-the-Art

Table 9 shows a relatively architecture enhancement results which has been obtained by this work compared with state-of-the-art PM implementations.

6. A Cryptography Protocol

The security of Automated Teller Machines (ATMs) is necessary because it has become one of the essential elements in human life. Also, securing banking transactions is one aspect that many researchers are interested in cryptography theory and mechanisms. Financial transactions need to insurance and make sure that hackers cannot change data and even steal its contents [32]. The security between two channels need some aspects like confidentiality, data authentication, data authorization as well as data integrity and nonrepudiation. So, the ATMs security and related smart cards can be investigated as a cryptography protocol.

As indicated in Fig. 9, each ATM has two security weakness; one attack is on the ATM itself and another attack is on the network existed between the ATM and bank computers. These weaknesses need to be strengthened using cryptography secure mechanisms to protect data from unauthorized use. An ECCP which is used to change message or data from one cipher form to another can be used with the basic key generation and exchange protocol [32]. This security key can be used in the authentication process. As well as a fast and small size key generation, several mechanisms can be added with elliptic curve cryptography like biometric to make a high assurance verification for ATM bank transactions.

ATMs allow customers to perform most banking services including withdrawing, depositing, transferring funds and checking account without bank employees. ATMs operate 24 hours for customer services and each of them has a small display and either touch screen or input devices for entering inputs. Steps of a secure financial transaction is showed in Fig. 10. A typical ECCP as an embedded system with ATMs is illustrated in Fig. 11.

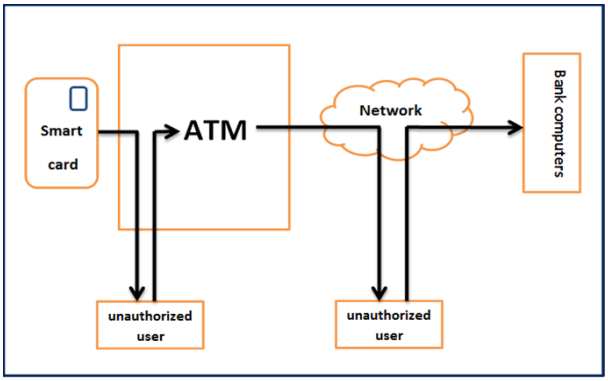

Fig. 9: The possibility attacks on smart cards and ATM machines

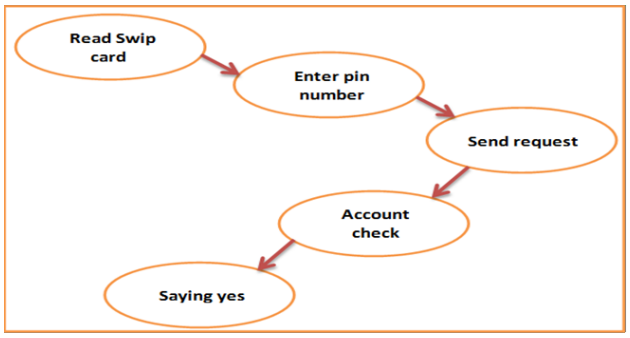

Fig. 10: ATM transaction steps taken from [33].

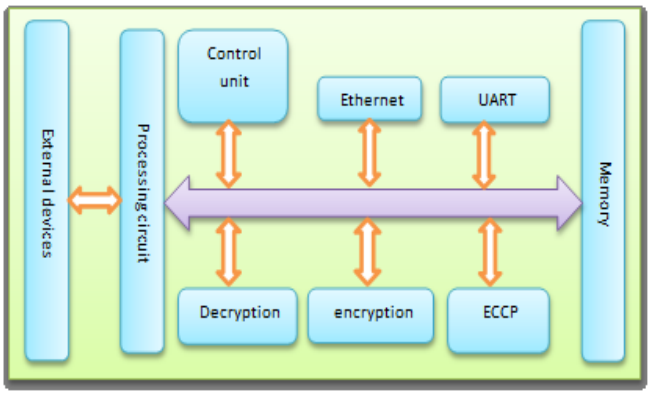

Fig. 11: ATM communication with other devices taken from [33].

The security of an ATM is based on using smart cards which may be contact, contactless, and hybrid cards. The hybrid cards can operate in either contact or contactless mode. Contact cards mean insert a smart card inside an ATM, and then read and store the account information recorded on magnetic strips on that card. The control unit starts the execution of the encryption unit to convert the card's information to a cipher text and send it to bank computers for verification. The same manner is occurred for contactless cards, but it is based on RFID sensors. The RFID tag was used to present the machine card reader which is read the magnetic strip on the card.

7. Conclusionand Future Work

In this work, there is a trade-off between the execution time, minimum number of registers (area overhead) and security over GF(2^{163}) and GF(2^{409}). Hence, the implementation is compared with state-of-the-art which shows a relatively improvement in the execution time for the same algorithmic architecture. These results show that, optimizing area/time of the ECCP requires the registers inside the multiplier and the number of modules of functional units be minimum. The high-performance demand and minimizing the FU power dissipation mandate the next step of this work must guarantee that the multipliers are, nearly, at no time left idle. On the other hand, this is an indication that the future work needs to address how to develop a methodology to optimize the amount of power dissipation by the finite field multiplier. There is a definite possibility that reducing the area requirements may significantly increase the reduction of total power consumption dissipated by the optimized design by fitting a smaller FPGA device. It is certain that this will cause more reduction specially in leakage power. The next step of this research is to impose an authentication scheme based on ECCP for IoT and embedded devices which satisfies all security requirements and is immune to various types of attacks [1].

Table 3: Squaring and reduction operations

Operation	# of occupied LUTs	# of occupied Slices	Latency (Virtex-6 XC6VLX760)
Squaring 163 bit	0	0	0.345 ns
reduction (one bit at a time)	164	89	1.616 ns
Fast reduction	165	146	1.267 ns
Squaring 409 bit	0	0	0.345 ns
Fast reduction	285	187	9.977 ns

Table 4: Results of the bit parallel Multipliers

Operation	Digit size in bits	Frequency in MHz	# of occupied Slices	Time in ns
Bit parallel	163	-	4,123	4.69
Pipeline Digit serial	82	380.12	2,718	5.26
Pipeline Digit serial	42	408.69	1,497	9.78
Bit parallel	409	-	17,560	5
Pipeline Digit serial	205	276.095	9,907	7.5
Pipeline Digit serial	102	281.597	4,813	15

Table 5: Karatsuba-Ofman for PM in $GF(2^{163})$ and for PM in $GF(2^{409})$

Operation	# of parallel operations	# of occupied Slices	Latency (Virtex-6 XC6VLX760)
Basic (82 bit)	2	4,715	18.711 ns
Basic (42 bit)	4	4,584	10.665 ns
Basic (22 bit)	8	4,600	6.845 ns
Basic (102 bit)	4	7,324	27.504 ns
Basic (52 bit)	8	7,275	13.517 ns
Basic (24 bit)	16	7,075	8.213 ns

Table 6: Pipelined digit serial K-O for PM in $GF(2^{163})$ and for PM in $GF(2^{409})$

Operation	# of parallel operations	# of occupied Slices	Latency (Virtex-6 XC6VLX760)
basic(163bit)	1	281	489 ns
basic (42 bit)	4	612	108.609ns
basic (22 bit)	8	1237	54.264ns
basic (12 bit)	16	2268	26.257ns
basic (6 bit)	32	4400	11.452ns
basic(409bit)	1	628	968 ns
basic (204 bit)	2	1325	658ns
basic (22 bit)	8	2843	369ns
basic (12 bit)	16	4228	184ns
basic (6 bit)	32	8764	95ns

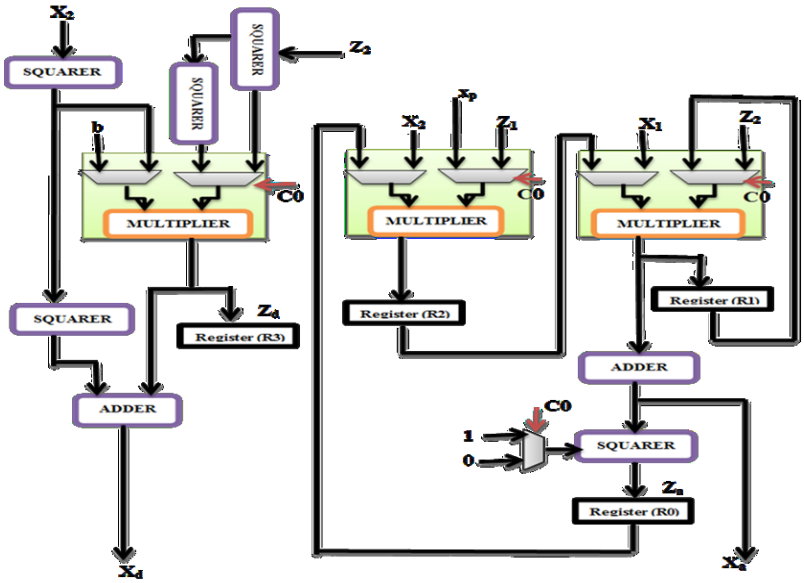

Fig. 7: The Point Addition and Point Doubling architectures

Table 7: The inversion results

Operation	PM algorithm	# of Slices	Frequency in MHz	Latency (Virtex-6 XC6VLX760)
Itoh-Tsujii	K-O on RTL	46198	----	219.398 ns
Itoh-Tsujii (pipeline)	K-O on RTL	4201	225.7	360 ns
Itoh-Tsujii (pipeline)	Bit parallel multiplier	4602	458.6	315 ns
Itoh-Tsujii (pipeline)	K-O on RTL	7806	257.3	608 ns
Itoh-Tsujii (pipeline)	Bit parallel multiplier (DS = 52)	5,657	296.845	504 ns

Table 8: Montgomery ladder algorithm for generic ECCP

Type of PM	Power in mW	Digit Size in bits	Frequency in MHz	# of Slices	Time (µs)
Bit parallel	5237.00	DS = 163	369.5	19,824	1.9
K-O shift& add	7440.74	DS = 163	151.9	14,068	3.583
Bit parallel	2115.00	DS = 52	253.770	18,807	29
Bit parallel	1824.25	DS = 26	317.140	12,630	59

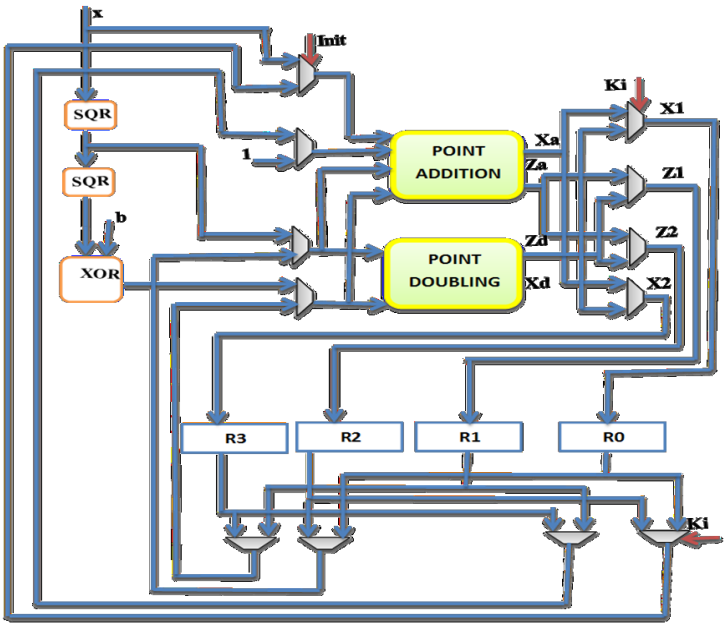

Fig. 8: The scalar multiplication architecture

Table 9: A comparison with state-of-the-art

Architecture	PM algorithm	FF multiplier	FPGA	Frequency in MHz	# of occupied slices	Time in µs
[23]	Montgomery Ladder	Bit parallel with K-O	Virtex 7	294	3041	4.6
[22]		Segmented pipeline	Virtex 5	113	11777	3.99
[8]		Digit parallel	Virtex 7	351	3107	11.28
This work		Bit parallel	Virtex 6	369.5	19,824	1.9

REFERENCES

[1] S. Kumari et. al., "A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers," *J. Supercomputer, Springer, Published online*, April 2017.

[2] C. Lee and H. Chien, "An Elliptic Curve Cryptography-Based RFID Authentication Securing E-Health System," *International Journal of Distributed Sensor Networks*, 2015.

[3] C. H. Gebotys, *Security in Embedded Devices*, New York, Springer, 2010.

[4] R. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography, Chapter 3 "Elliptic Curve Arithmetic"*, New York: Springer Professional Computing, 2004.

[5] Institute of Electrical and Electronics Engineers, *Standard Specifications for Public Key Cryptography*, (IEEE P1363: IEEE), 2000.

[6] National Institute of Standards and Technology (NIST), *Digital Signature Standard (DSS)*, (NIST: NIST FIPS 186-4), 2009. https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf

[7] M. Imran, M. Rashid, A. R. Jafri, and M. Najam ul Islam, "ACryp-Proc: Flexible Asymmetric Crypto Processor for Point Multiplication," *IEEE Access,* Vol. 6, pp. 22778-22793, 2018.

[8] Realpe P C and Velasco-medina J, *High-Performance Elliptic Curve Cryptoprocessors over GF(2^m) on Koblitz Curve*, Analog Integr Circ Sig Process. Springer. 85 129–138, 2015.

[9] M.S. Albahri, M. Benaissa, and Zia Uddin A. Khan, "Parallel Implementation of ECC Point Multiplication on a Homogeneous Multi-Core Microcontroller," *Proceeding of 12th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN)*, IEEE, 16-18 Dec., Hefei, China, pp. 386-389, 2016.

[10] Loi K C C and Ko S-B, "Parallelization of Scalable Elliptic Curve Cryptoprocessors in GF(2^m)," *Microprocessors and Microsystems*, Volume 45, Part A, Elsevier 45 Pages 10–22, August 2016.

[11] Rashidi B., Farashahi R. R. and Sayedi S. M., " High-Speed and Pipelined Finite Field Bit-Parallel Multiplier over GF(2^m) for Elliptic Curve Cryptosystems," *11th Int. ISC Conferene on Information Security and Cryptology,* Tehran 15-20, 2014.

[12] Rashidi B., Farashahi R. R. and Sayedi S. M., "High-Performance and High-Speed Implementation of Polynomial Basis Itoh-Tsujii Inversion Algorithm over GF(2^m)," *IET Information Security,* Vol. 11, Iss. 2, pp. 66-77, 2017.

[13] P. L. Montgomery, "Speeding the Pollard and Elliptic Curve Methods of Factorization," *Math. Computing,* Vol. 48 - pp. 243–264, 1987.

[14] J. López, and R. Dahab, "Improved Algorithms for Elliptic Curve Arithmetic in GF(2^n)," *Proc. Sel. Areas Cryptography,* pp. 201–212, 1999.

[15] T. Itoh, and S. Tsujii, "A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) using Normal Bases," *Information Computing,* Vol. 78, No. 3, pp. 171–177, 1988.

[16] A. P. Fournaris, C. Dimopoulos, and O. Koufopavlou, "A Design Strategy for Digit Serial Multiplier Based Binary Edwards Curve Scalar Multiplier Architectures," *Proc. Euromicro Conference on Digital System Design (DSD)*, 2017.

[17] Rashidi B, Farashahi R R and Sayedi S M, "High-Speed Hardware Architecture of Scalar Multiplication for Binary Elliptic Curve Cryptosystems," Microelectronics Journal, 2016.

[18] A. B. El-Sisi, S. M. M Shohdy, and N. Ismail, "Reconfigurable Implementation of Karatsuba Multiplier for Galois Field Elliptic Curves," *Tarek Sobh, Khalid Ellithy, and Ausif Mahmood (Editors)*, Novel Algorithms and Techniques in Telecommunications and Networking, Springer Nature America, pp 87-92 Inc, 2010.

[19] F. Rodriguez-Henriquez and Q. K. Kog. "On Fully Parallel Karatsuba Multipliers for GF (2m)". *Proc. International Conference on Computer Science and Technology (CST)*, pp. 405-410, 2003.

[20] H. Modares, Y. Salem, R. Salleh and M. T. Shahgoli "A Bit-Serial Multiplier Architecture for Finite Fields Over Galois Fields," *Journal of Computer Science*, 2010.

[21] M. Keller and W. Marnane "Low Power Elliptic Curve Cryptography," *Lecture Notes in Computer Science*, and *Proceeding of PATMOS'07*, pp. 310-319, 2007.

[22] M. N. Ismail, "Towards Efficient Hardware Implementation of Elliptic and Hyperelliptic Curve Cryptography," *Ph.D. Thesis, Dept. of Electrical and Computer Engineering*, University of Waterloo, Ontario, Canada, July 2012.

[23] H. Cohen and G. Frey, Eds. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, "Arithmetic of elliptic curves," D. Doche and T. Lange, Chapman & Hall/CRC, Boca Raton, FL, Chapter 13, pp. 267-302,2006.

[24] B. Ansari and M. A. Hasan, "High-Performance Architecture of Elliptic Curve Scalar Multiplication," *IEEE Transactions on Computers*, Vol. 57, No. 11, pp. 1443-1453, 2008.

[25] Zia U. A. Khan and M. Benaissa, "High-Speed and Low-Latency ECC Processor Implementation Over GF(2m) on FPGA," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, Vol. 25, no. 1, pp. 165-176, Jan. 2017.

[26] P. K. Mishra, P. Pal and P. Sarkar, "Towards Minimizing Memory Requirement for Implementation of Hyperelliptic Curve Cryptosystems," *Lecture Notes in Computer Science*, Vol. 4464, and ISPEC, pp. 269-283, 2007.

[27] W. N. Chelton and M. Benaissa "Fast Elliptic Curve Cryptography on FPGA," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol.16, Issue 2, pp. 198-205, 2008.

[28] L. Lijuan and L. Shuguo, "High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication over GF(2^m)," *IEE Transactions on VLSI Systems*, pp. 1223-1232, 2016.

[29] P. Zode et.al. "Fast Architecture of Modular Inversion Using Itoh-Tsujii Algorithm," *International Symposium on VLSI Design and Test VDAT*, pp. 48-55, 2017.

[30] Shohdy S, El-Sisi A and Ismail N., "FPGA Implementation of Elliptic Curve Point Multiplication over GF(2^{191})," *Advances in Information Security and Its Applications (ISA)*, *Lecture Notes in Computer Science*, Springer-Verlag Berlin. Germany, 5576 619–634, 2009.

[31] Kudithi T, Sakthivel R., "High-Performance ECC Processor Architecture Design for IoT Security Applications," *Supercomputer J. Springer*. 75 447–474, 2019.

[32] David Hutchison, and et., " Information Security and Cryptology", *ICISC 2004, 7th International Conference Seoul, Korea, Springer*, December 2-3, 2004.

[33] Y. W. Hau1 Mohamed Khalil-Hani and Muhammad N. Marsono, "Systemc-Based Hardware/Software Co-Design of Elliptic Curve Cryptographic System for Network Mutual Authentication", *Malaysian Journal of Computer Science*, Vol. 24(2), 2011.