COUPLED FPGA/ASIC IMPLEMENTATION OF ELLIPTIC CURVE CRYPTO-PROCESSOR

Mohsen Machhout[1], Zied Guitouni[1], Kholdoun Torki [2], Lazhar Khriji[3] and Rached Tourki[1]

[1]Electronics and Micro-Electronics Laboratory (E. μ .E. L)
Faculty of Sciences of Monastir, Tunisia
{Machhout, ziedguitouni}@yahoo.fr, rached.tourki@planet.tn)
[2] TIMA – CMP Laboratory, Grenoble, France
kholdoun.torki@imag.fr
[3]Information Engineering Department, Sultane Qaboos University, Oman.
lazhar@squ.edu.om

ABSTRACT

In this paper, we propose an elliptic curve key generation processor over GF(2^{163}) scheme based on the Montgomery scalar multiplication algorithm. The new architecture is performed using polynomial basis. The Finite Field operations use a cellular automata multiplier and Fermat algorithm for inversion. For real time implementation, the architecture has been tested on an ISE 9.1 Software using Xilinx Virtex II Pro FPGA and on an ASIC CMOS 45 nm technology as well. The proposed implementation provides a time of 2.07 ms and 38 percent of Slices in Xilinx Virtex II Pro FPGA. Such features reveal the high efficiently of this implementation design.

KEYWORDS

Elliptic curve cryptography, cellular automata, finite fields & Montgomery algorithm.

1. INTRODUCTION

Elliptic Curve Cryptography (ECC) has been proposed by Miller [25] and Koblitz [17] in the mid 1980s. Recently, ECC has gained much attention in industry and academia. The main reason is that for a properly chosen elliptic curve, the sub-exponential algorithm that can be used, to break the system through the solution of the discrete logarithm problem, is not easy to be defined. Elliptic Curve Cryptography is one among the most powerful public keys [25]. The security of 160-bit and 224-bit of ECC has been shown to be equivalent to 1024-bit and 2048-bit of RSA [29]. Different security organizations like ISO, ANSI, IEEE and NIST, have been working to standardize the use of ECC.

To implement ECC, finite fields GF(p) and GF(2^m) have been used, where p is prime and m is positive integer. In particular, GF(2^m), which is an m-dimensional extension field of GF(2), is suitable for hardware implementation because there is no carry propagation in arithmetic operations. The function used for this purpose is the scalar multiplication K.P, where K is an integer and P is a point on an elliptic curve.

Recently, Hardware and firmware implementation of ECC over different fields GF(2^m) have been reported in numerous works. Leung et al. [26] presented a microcoded FPGA-based elliptic curve processor. This design is parameterized for arbitrary key sizes and allows the rapid development of different control flows. They have used a normal basis for the Galois field operations, and the point multiplication can be computed in 14.3 ms for GF(2^{281}). Morales-

Sandoval and Feregrino-Uribe [27] proposed a hardware architecture that can perform three different ECC algorithms. The main functional units in their cryptosystem are coprocessor for scalar multiplication, random number generator, algorithm units, and main controller. Its scalar multiplication can be computed in 4.7 ms for GF(2^{191}). Orland and Paar [28] designed a reconfigurable elliptic curve processor over GF(2^{167}), the processor consists of main controller and arithmetic units.

For the implementation of ECC in GF(2^{163}), Chang Hoon et al [4] described an FPGA implementation of high performance ECC processor over GF(2^{163}). The proposed architecture is based on Lopez- Dahab elliptic curve point multiplication algorithm and Gaussian normal basis for GF(2^{163}) and drive parallelized elliptic curve in point doubling and *point addition* algorithms with uniform addressing. Dan Young-ping et al, proposed a parallel hardware processor to compute elliptic curve scalar multiplication in polynomial basis representation over GF(2^{163}) [7]. Bednara et al [3] designed an FPGA-based cryptographic processor architecture that allows to use multiple squares, adders and multipliers. They are looking for an hybrid coordinate representation in affine projective Jacobian and Lopez-Dahab form. Two prototypes were synthesized for GF(2^{191}). In Ref [5], Cheung et al proposed an ECC design for various field operations, which is, however, not optimized for fixed field. The implementations of ECC in an integrated circuit (ASIC), are presented in works [18] and [19].

Cellular automata (CAs) have been used in evolutionary computation for over a decade [15]. They have been used in variety of applications, such as parallel processing and number theory. Programmable cellular automata architecture has been used to design arithmetic computations by Zhang et al. [24]. Choulhury has designed an LSB multiplier based on CA [6]. As reported in Ref [25], Jun–Cheol Jeon has proposed an efficient division architecture using restricted irreducible polynomial on ECC based cellular automata.

In the present paper, we report a fast parallel architecture, for the Implementation of the elliptic curve scalar multiplication processor using programmable cellular automata, In our scheme, Montgomery algorithm for key generation is used. The key operations of scalar multiplication are GF(2^{163}). GF(2^{163}) multiplier is implemented using LSB multiplier based on CA. GF(2^{163}) inversion is implemented using Fermat algorithm.

The rest of the paper is organized as follows: the theoretical background, including ECC and CA is described in section 2; section 3 presents the design of elliptic curve crypto-processor using programmable cellular automata. The experimental results are discussed in section 4. Section 5 concludes the paper.

2. PRELIMINARY

In this section, we briefly discuss the mathematical background of the ECC, the characteristics and the properties of the programmable cellular automata.

2.1. Elliptic Curve Cryptosystem

Elliptic curves over GF(2^m) are particularly attractive because of the finite field operations that can be implemented efficiently in hardware. In this paper, the selected finite field GF(2^m) is an elliptic curve E(a,b). It is defined as a set of points satisfying eq (1).

$$y^2 + xy = x^3 + ax^2 + b \pmod{P(x)} \quad (1)$$

Where a, b $\in$ GF(2^m), b $\neq 0$ and P(x) is the irreducibly polynomial.

Let $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ be points in E(GF(2^m)) given in affine coordinates.
Assuming $P_1, P_2 \neq 0$ and $P_1 \neq -P_2$. The sum
$P_3(x_3, y_3) = P_1 + P_2$ is computed as follows:
If $P_1 \neq P_2$:

$$x_3=\lambda^2+\lambda+x_1+x_2+a$$
$$y_3=\lambda.(x_1+x_3)+y_1+x_3 \quad (2)$$
$$\lambda=(y_2+y_1)/(x_2+x_1)$$

If $P_1= P_2$:

$$x_3=\lambda^2+\lambda+a$$
$$y_3=x_1^2+(\lambda+1).x_3 \quad (3)$$
$$\lambda=x_1+y_1/x_1$$

ECC security is based on the discrete logarithm problem and called the Elliptic Curve Discrete Logarithm Problem (ECDLP). Thus, a cryptosystem could be built using this approach. The ECDLP consists of giving two points P, Q Є GF (2^m) to find the positive integer k such as Q = kP. On the contrary, knowing the scalar k and the point P, the operation kP is relatively easy to compute [14]. The hierarchy of an Elliptic Curve Point Multiplication is depicted in Fig.1.

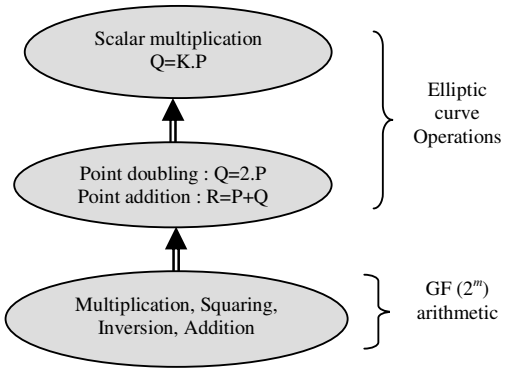

Figure 1: Elliptic Curve hierarchy

Compared with the field multiplication in affine coordinates, inversion is the most expensive basic arithmetic operation in GF(2^m). Inversion can be avoided by means of projective coordinate's representation [8]. A point P in projective coordinates is represented by three coordinates X, Y, and Z. This representation reduces the internal computational operations. It is expected to convert back the point P from projective coordinates to affine coordinates in the final step. This is because the affine coordinate's representation involves the usage of only two coordinates and therefore is more useful for external communication saving some valuable bandwidth. In standard projective coordinates, the projective point (X: Y: Z) with $Z \neq 0$ corresponds to the affine coordinates where, x = X / Z and y = Y / Z. (X: Y: Z). the projective equation of the elliptic curve is given by:

$$Y^2.Z+X.Y.Z=X^3+a.X^2.Z+b.Z^3 \quad (4)$$

where a, b ∈ GF(2^m).

2.2. Cellular Automata

A cellular automaton (CA) is defined as an uniform array of identical cells in a n-dimensional space. Each cell can exist in a finite discrete state space where, the state space is fairly small. The evolution of a cellular automaton occurs in a series of time steps (clock cycles). It can be characterized by four basic properties: the cellular geometry, the neighborhood specification, the number of states per cell, and the algorithm under which the cellular automaton computes its successor states.

A CA can be defined as a d-dimensional Euclidean space (with d =1, 2 or 3). In this paper, we use d = 1, i.e. one dimensional grids. The identical rule contained in each cell is essentially a finite state machine, usually specified in the form of a rule table, with an entry for every possible neighbourhood configuration of states.
A 1-D binary CA is an array of cells (registers) $[q_0(t),q_1(t),\ldots,q_n(t)]$ where each cell's state $q_i \in \{0,1\}$ and $i \in [0,n-1]$ is any of its permissible state [20]. At each discrete time step (clock cycle), each cell of the CA updates its state using a transition rule based on a Boolean function applied to the current states of each cell's state transition neighborhood $q_i(t+1) = f_i(q_1(t),q_2(t),\ldots)$. The conventional nearest three-cell state transition neighborhood, having a radius r = 1, consists of itself (q_i) and its left/right most neighbors (q_{i-1}/q_{i+1}). Cellular automata may be uniform, with the same set of state transition neighborhood/rules used for each cell, or hybrid.
The Programmable cellular automata (PCA) [13] is a CA whose the state transition rule for each cell is not fixed but controlled by a number of control signals where different functions can be generated.

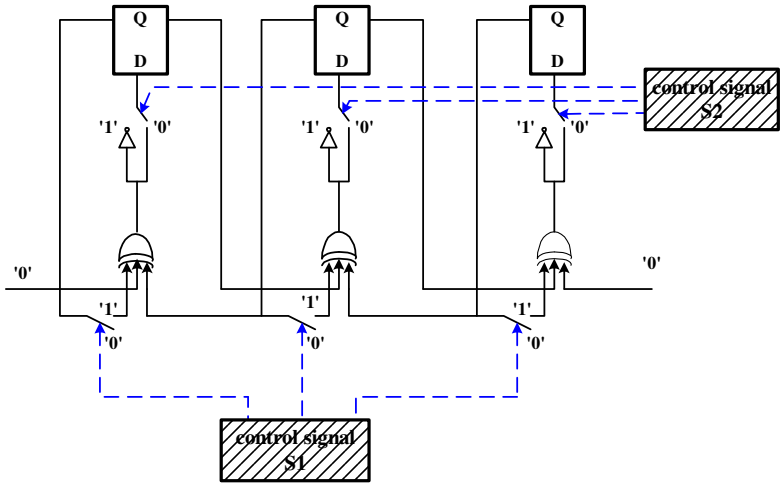

Figure 2: Programmable cellular automata

Figure.2 shows the PCA with four programmed state transition rules. It is denoted as PCA 90/105/150/165. When both control signals S1 and S2 are open, the rule 90 is applied to the cell. The cell configured with rule 150 when the control signals S1 and S2 are open and closed, alternatively. In table 1, we write down the different states of the control signals S1 and S2 and the corresponding rule.

Table1: The state transition rule for PCA 90/150/165/105

Control signal S1	Control signal S2	Rule name
0	0	90
0	1	150
1	0	165
1	1	105

3. DESIGN OF THE ELLIPTIC CURVE CRYPTO-PROCESSOR

3.1. Elliptic curve scalar multiplication algorithm

As mentioned above, we will study the scalar multiplication method based on Montgomery algorithm. The main advantages of this algorithm are: it does not have any extra storage requirements; the same operations are performed in every-iteration of the main loop, thereby potentially increasing resistance of timing attacks and power analysis attacks. The algorithm is shown below. In this algorithm, Madd, Mdouble and Mxy are three basic functions for point addition, point doubling and conversion of projective coordinates to affine coordinates.

Algorithm1: Montgomery point Multiplication Algorithm
Input: k = $(k_{n-1}, k_{n-2}.....k_1, k_0)2$ with
$k_{n-1} = 1$, $P(x, y) \in E(F2^m)$
Output: Q = kP
Procedure: MontPointMult (P, k)
1. Set X1←x, Z1←1, X2←x^4+b,Z2←x^2
2. For i from n − 2 downto 0 do
2.1 if (ki = 1) then
Madd(X1, Z1,X2, Z2), Mdouble(X2, Z2)
2.2 else
Madd(X2, Z2,X1, Z1), Mdouble(X1, Z1)
end if
End for
3. Q ← Mxy (X1, Z1, X2, Z2)
4. Return Q

According to this algorithm, the scalar multiplication is performed over three steps; (i) initialization operations, (ii) iteration of the point addition and point doubling; and (iii) the conversion to affine coordinates.

3.2. Architecture of the processor

The Elliptic Curve Point Multiplication processor is depicted in Figure 3. The scalar number k represents the number of time P is added to itself. It is performed by expressing k in binary form K=KiKi-1…K1K0 and applying double and add method according to:

$$KP = 2(..2((K_i P) + K_{i-1} P) + ...) + K_0 P \quad (5)$$

The main units of the proposed Elliptic Curve Point Multiplication processor are shown in figure.3, including the input and the output interfaces for storing the input and the output data. The control module consists of a finite state machine. It generates the control signals for the initialization operations of finite field, the point addition and point doubling operations, and the conversion to affine coordinates operations, relying on the key values by the Montgomery algorithm. The elliptic curve operator is formed by the point addition and the point doubling modules. Finally the arithmetic and logical unit (ALU) allow parallel execution of finite field addition, inversion and multiplications, which are controlled by the control unit.

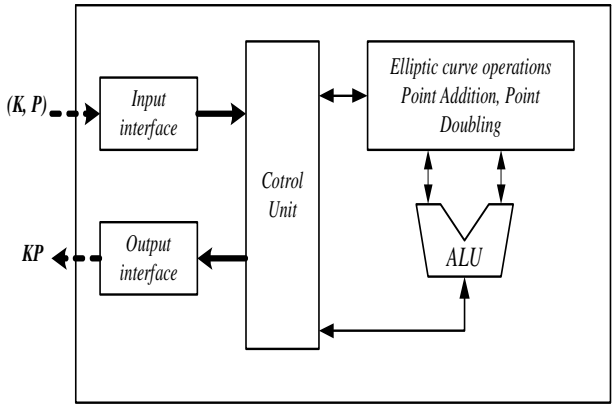

Figure 3: Elliptic Curve Point Multiplication Processor

3.3 Arithmetic and logic unit (ALU)

The arithmetic and logic unit is an important module for elliptical processor. It might complete the arithmetic operations, such as the finite field addition, multiplication and inversion operations. In the present work, the finite field GF (2^m) is the underlying field used as a basis for the elliptic curves. It can be viewed as a vector space of dimension m over the field GF (2). In hardware field, elements can be easily implemented as a bit vector, which makes this kind of finite fields interesting for hardware implementations [16]. In this section, we present the hardware implementation of the finite field operations using PCA in GF (2^m); the representation treated is a polynomial basis.

3.3.1. Addition

The addition of two elements in GF(2^m) is performed by adding the coefficients modulo 2, which is a simple bit-wise XOR-ing the coefficients of equal powers of x, that is :
$A = (a_{m-1}a_{m-2} \ldots a_2a_1a_0)$; $B = (b_{m-1}b_{m-2}\ldots b_2b_1b_0)$ and $C= (c_{m-1}c_{m-2}\ldots c_2c_1c_0)$, where

$$C = \sum_{i=0}^{m-1} a_i + b_i \bmod 2 \tag{7}$$

3.3.2. Multiplication

The multiplication of two field elements C(x) = A(x). B(x), where $A(x)=\sum_{i=0}^{m-1} A_i x^i$; $B(x)=\sum_{i=0}^{m-1} b_i x^i$ and $C(x)=\sum_{i=0}^{m-1} c_i x^i$, finite field multiplication can be carried out by multiplying A(x) and B(x) and then performing reduction modulo P(x) or alternatively by interleaving multiplication and reduction, the multiplication is shown as:

$$\left(b(x)a_{m-1}x^{m-1}+\ldots+b(x)a_2x^2+b(x)a_1x+b(x)a_0\right) \bmod P(x) \tag{8}$$

In Ref [15], H. Li and C.N Zhang presented a low complexity programmable cellular automata based versatile modular multiplier in GF(2^m). The PCA rules are shown in table 1, where Cm are configured as coefficients of B(x) and Cr are configured as coefficients of P(x), Xs are configured as coefficients of A(x), Xl and Xs are partial results of neighborhood PCA.
The algorithm of the multiplier based on PCA rules is shown in algorithm.2

Algorithm 2: PCA based modular multiplication algorithm

Input: A(x), B(x), P(x) ∈ GF(2^m)
Output C = AB mod P(x)
Reset PCA
Configure Coefficients of B(x) as Cm, and Coefficients of P(x) as Cr
Run PCA m clock cycle

In figure.4, we present the general architecture of the serial multiplier based on PCA in GF(2^m). According to figure.4, we notice that the architecture of the serial multiplier is consisted of a logical block (LB) of m combination logic (CL) and m bascules. The execution time to compute the complete modular multiplication in GF(2^m) with this architecture is equal to (m.T), where T is the critical time of this architecture.

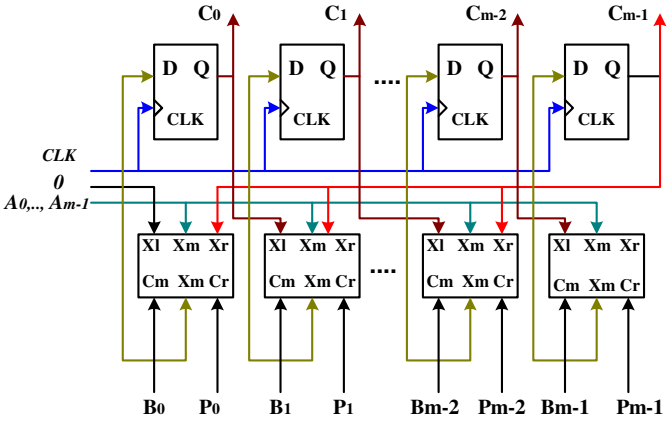

Figure.4 Serial Multiplier architecture in GF(2^m)

3.3.3. Inversion

The inversion is a complex operation that is computed only once in a k.P operation. In this subsection, we select the Fermat's theorem to perform inversion in a finite field GF(2^m). To apply this theorem, the multiplicative inverse of the field element A can be obtained by recursive squaring and multiplication, since the field element A is expressed as:

$$C=A^{-1}=A^{2^n-2}=(A(A(A...A(A(A)^2)^2...)^2)^2)^2 \quad (9)$$

The above equation can be generalized as follows.

Algorithm 3: Fermat's algorithm
Input: A(x), P(x) ∈ GF(2^m)
Output C = A^{-1} mod P(x)
1. $C_0 = A$
2. For i in 1 to n-2 do
   $C_i = A\ (C_{i-1})^2 = A^{2i-1},\ (1 \le i \le n-2)$
   End for.
3. $C_{n-1} = (C_{n-2})^2 = A^{-1}$
4. Return C

According to algorithm 3, the inversion operation is divided in three steps: initialization operation, iteration of the squaring and multiplication, and finally the final result. In our scheme, we select the multiplier based on cellular automate to complete multiplication and squaring operations.

3.4 The elliptic curve operator

The elliptic curve operator is defined by the point addition module and the point doubling module. According to our scheme, we select the Montgomery method in projective coordinate to perform the elliptic curve operations.

3.4.1. Point addition module

In standard projective coordinates, the projective point (X: Y: Z) with $Z \neq 0$ corresponds to the affine coordinates x = X / Z and y = Y / Z. The projective equation of the elliptic curve E is written as:

$$Y^2.Z+X.Y.Z=X^3+a.X^2.Z+b.Z^3 \quad (10)$$

Let P = (X_1, Z_1) and Q = (X_2, Z_2) be two points in projective coordinates, belong to the curve of (10). The coordinates of P + Q = (x_3, y_1) in projective coordinates is given by eq (11):

$$X_3 = x.Z_3+(X_1.Z_2)(X_2.Z_1) \quad (11)$$
$$Z_3 = (X_1.X_2+X_2.Z_1)^2$$

The point addition module is shown in Fig.5. The required field operations for point addition are as follows: three general multiplications, one multiplication by x, one squaring and two additions.

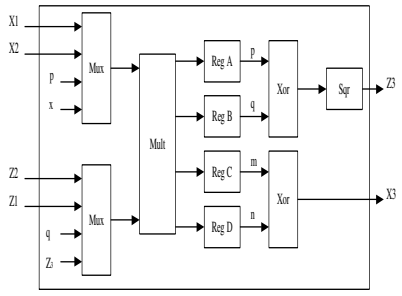

Figure 5: Montgomery point addition architecture

3.4.2. Point doubling module

Let P = (X_1, Z_1) be one point in projective coordinates, the point addition 2P = (X_3, Z_3) in projective coordinate can be deduced from equation.12:

$$X_3 = X_1^4 + b.Z_1^4$$
$$Z_3 = X_1^2.Z_1^2 \quad (12)$$

The computation of (6) requires a general multiplication, a multiplication by a constant b, four squaring and an addition (XOR). The architecture of the point doubling module is shown in Fig.6.

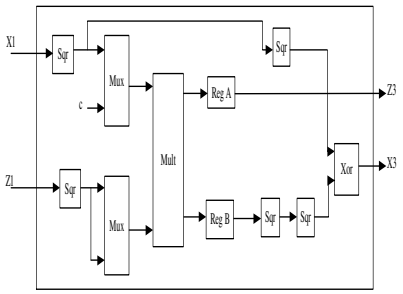

Figure 6: Montgomery point doubling architecture

4. EXPERIMENTAL RESULTS

In this section, we present a results of the implementation of the elliptic curve key generation processor on GF(2^{163}) using polynomial basis. In our scheme, two types of implementations are presented. The first implementation is on the ISE 9.1 software using Xilinx Virtex II pro FPGA, and the second is one the implementation of ECC on ASIC circuit using CMOS 45 nm technology

4.1. FPGA Implementation of ECC Processor

The ECC processor over GF(2^{163}) has been coded using VHDL language and synthesized using ISE Xilinx 9.1 software, in which Virtex II pro. $P(x)=x^{163}+x^8+x^7+x^3+1$ is the irreducible polynomial (see Fig.3).
In table 2, we present the synthesis results of the arithmetic unit. Three criteria are summarized: the occupied resources (number of slices), the frequency and the timing of the field operations.

Table 2: Synthesis of the finite field arithmetic

Operations	Frequency (Mhz)	Occupation (Slices)	Time (µs)
Multiplication	177.89	225	0.91
Inversion	179.08	1505	26

According to that reported in table 2, we notice the high speed of the multiplier based on cellular automata and the low occupation of the resources. The multiplication is performed in 915 ns. The total number of slices is equal to 225. The inversion occupies 1505 slices, and one field inversion is performed in 0.266 ms.
In table 3, we present the synthesis results of the elliptic curve operations. All operations were defined over the finite field GF(2^{163}).

Table 3: Synthesis of the elliptic curve operations

Operations	Frequency (Mhz)	Occupation (Slices)	Time (µs)
Point Addition	160.83	961	3.05×10^{-3}
Point Doubling	177.89	873	1.84×10^{-3}
Point multiplication	167.838	12861	2.07

Comparison with results in terms of frequency to recent proposed architectures is given in table 4. Results of the present work are not comparable to previous hardware implementations. This is because each design uses a different hardware platform, finite field and finite size.

Table.4 Performance comparison of Frequency results

	Device	**Frequency (MHz)**
Shu et al [21], GF(2^{163})	XCV2000E-7	68.9
Eberte et al [9], GF(2^{m}) <256	XCV2000E-7	66.4
Benaissa and Wim [2], GF(2^{160})	XCV2000E-7	150
Gura et al [11] , GF(2^{163})	XCV2000E-7	66.5
Saqib et al [22] , GF(2^{191})	XCV3200E	9.99
Grabbe et al [12], GF(2^{233})	XC2V6000	100
This work, GF(2^{163})	Virtex II pro	167.83
	XCV2000E-7	69.15

Table 5 shows a comparison of the performance of the scalar multiplication timing result that we have obtained with the same hardware implementations.

Table.5 Performance comparison of timing results

Reference	**m**	**Platform**	**Time (ms)**
[10]	113	Atmel AT94K40	10.9
[1]	160	Virtex-E	3.9
[23]	163	VirtexE XCV2600	2.618
This work	163	Virtex-II Pro	2.07

4.2.ASIC Implementation based on Platform

We designed an ECC IP using the VHDL language and synthesized using Synopsys Design Compiler. The resulting netlist is used as input to Cadence in order to perform mapping and routing with a 45 nm CMOS technology. The results obtained from these operations are reported in table 6 and figure.7. The final ASIC has been implemented using CMOS 45nm technology. The result in synthesis operating frequency is about 346 Mhz and data arrival time about 2.89 ns. The ECC processor areas are about 0.54 mm^2. The total Input/output is equal to 44.

The core dimension of the ECC processor is about 0.416mm x 0.416mm, and the core is about 0.173 mm^2.

Table 6. ASIC implementation of ECC processor

Core dimension	0.416mm x 0.416mm
Core area	0.173 mm^2
Circuit dimension	0.735mm x 0.735mm
Circuit area	0.54 mm^2
Data arrival time	2.89 ns
Core dimension	0.416mm x 0.416mm

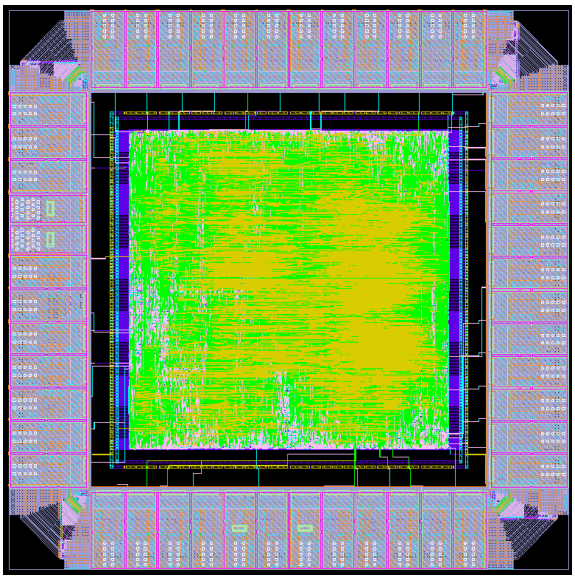

Figure.7: Layout of the design

5. CONCLUSIONS

This paper proposes a parallel and high performance architecture of elliptic curve key generation over GF(2^{163}) scheme based on the Montgomery scalar multiplication algorithm. In our scheme, we have developed the arithmetic unit over the finite field GF (2^{163}) and the elliptic curve operations. We have provided a comparison with some ECC hardware implementations in terms of occupation (Slices) and performances. The proposed ECC implementation outperforms all other implementations used for comparative purposes. It provides a time of 2.07 ms and 38 % slices in platform Xilinx Virtex II pro FPGA. The second part of the paper, present's the first design of the ECC processor using a 45 nm CMOS technology. The ASIC area is about 0.54 mm^2. This circuit operates with clock frequency of 346 Mhz.

REFERENCES

[1] L. Batina, G. Bruin-Muurling, and S. B. Ors, "Flexible Hardware Design for RSA and Elliptic Curve Cryptosystems", Proceedings of Topics in Cryptology – CTRSA 2004, Lecture Note in Computer Science, Springer-Verlag, Vol. 2271, 2004, pp. 250-263.

[2] M. Benaissa, W. M Lim, Design of flexible GF(2m) elliptic curve cryptography processors, IEEE Transactions on VLSA Systems 14 (6), 2006, pp. 659-662.

[3] M Bednara, M Daldrup, J von zur Gathen and J Shokrollahi, Reconfigurable implementation of elliptic curve crypto algorithms. Reconfigurable Architectures Workshop, 16th International Parallel and Distributed Processing Sympsium, April 2002.

[4] Chang Hoon Kim, Soonhak kown and Chun Pyo Hong, FPGA implementation of high performance ECC processor over GF(2163), Journal of Systems Architecture, Vol 54(, pp 893-900, 2008.

[5] Cheung R C C, Telle N J, Luk W, et al, Customizable elliptic curve cryptosystems, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol 13 (2), pp 1048-1059, 2005.

[6] P.P. Choudhury, R. Barua, "Cellular automata based VLSI architecture for computing multiplication and inverses in GF(2m), Proceedings of the IEEE 7th International Conference on VLSI Design, 1994, pp. 279-282.

[7] Dan Young-ping, Zou Xue-cheng, Han Yu and Yi Li-hua, Design of highly efficient elliptic curve crypto-processor with two multiplications over GF(2163), The journal of china Universities of Posts and Telecommunications, Vol 16(2), pp 72-79, 2009.

[8] M. Dion. "Implantation d'ECDSA sur une Carte à Puce".Université de Montréal, Département d'informatique et de Recherche Opérationnelle. Mai 1999.

[9] H. Elberte, N.Gura, S. Chang-Shantz, Vipul Gupta, A Cryptographic Processor for Arbitrary Elliptic Curves over GF(2m), Application-Specific Systems, Architectures, and Processors , 2003, pp. 444-454.

[10] M. Ernest et al, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over GF(2n) ", In Proc. of CHES'02, Vol. 2523 of LNCS, Redwood Shores, CA. Springer , August 2002, pp. 381-399.

[11] N. Gura, S.C. Shantz, H. Elberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, D. Stebila , An end-to-end systems approach to elliptic curve cryptography , CHES 2000, Lecture Notes in Computer Science, vol.2523, 2002, pp.349-365.

[12] C. Grabbe, M.Bednara, J. von zur Gathem, J. Shokrollahi, J. Teih, A high performance vliw processor for finite field arithmetic, Reconfigurable architecture Workshop (RAW), 2003.

[13] P.D. Hortensius, R.D.McLeod, and H.C. Card, Parallel Random Numbers VLSI Systems using Cellular Automata, IEE Transactions on Cmputers, vol. 38, no. 10, pp.1,466-1,473 October 1989.

[14] T. Izu1, B. Moller, and T. Takagi, "Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks", Progress in Cryptology – INDOCRYPT 2002,. Springer-Verlag LNCS 2551, 2002, pp. 296–313

[15] Jun-Cheol Jeon, Kee-Young Yoo, "Elliptic curve based hardware architecture using cellular automata", Mathematics and Computers In Simulation, Elsevier, 2007.

[16] M.Jung, F. Madlener, M. Ernst, S. Huss, "A Reconfigurable Coprocessor for Finite Field Multiplication in GF (2n) ", IEEE Workshop Heterogeneous reconfigurable Systems on Chip, Hamburg, April 2002.

[17] N. Koblitz, Elliptic curve cryptosystems, Mathematics of computations 48 (1987), pp 203-209.

[18] Sakyama K, Batina L, Preneel B, et al, Multicore curve-based cryptoprocessor with rconfigurable modular arithmetic logic units over GF(2^n), IEEE Transactions on Computers, vol 56 (9), pp 1269-1282, 2007.

[19] Sozzana F, Bertoni G, S Turcato, et al, A parallelized design for an elliptic curve cryptosystem coprocessor, Proceeding of the International Conference on Information Technology, IEEE Computer Society, pp 626-630, 2005.

[20] Sheng-Uei Guan and Syn Kiat Tan, "Pseudorandom Number Generation With Self-Programmable Cellular Automata", IEEE Transactions on computers,- Aided design of integrated circuits and systems, vol 23, pp 1095-1101,July 2004.

[21] C. Shu, K. Gaj, T. El-Ghazawin, Low latency elliptic curve cryptography accelerators for NIST curves over binary fields, FPT 2005, vol. 1965, 2000, pp. 41-56.

[22] N.A. Saqib, F. Rodriguez- Henriquez, A. Diaz-Pérezm, A parallel architecture for fast computation of elliptic curve scalar multiplication over GF(2m), Parallel and Distributed Processing Symposium (IPDPS), 2004.

[23] El hadj youssef WAJIH, Guitouni ZIED, Machhout MOHSEN and Tourki RACHED, "Design and Implementation of Elliptic Curve Point Multiplication Processor over GF (2m)", IJCSES

International Journal of Computer Sciences and Engineering Systems, ISSN 0973-4406, Vol.2, No2, pp 125-134,2008

[24] C.N. Zhang, M.Y. Deng, R.Mason, "A VLSI programmable cellular automata array for multiplication in GF(2^n), Proceedings of the PDPTA'99 International Conference, 1999.

[25] V.S. Miller, Use of elliptic curves in cryptography, in Proceeding of the Advances in Cryptography, CRYPTO'85, pp 417-426

[26] K.H. Leung et al., FPGA implementation of a microcoded elliptic curve cryptographic processor, IEEE Symposium on Field Programmable Custom Computing Machines, 2000, pp 68-76.

[27] M.Morales-Sandoval, C.Feregrino-Uribe, on the hardware design of an elliptic curve cryptosystem, Proceeding of the 5th Mexican International Conference in Computer Science, 2004, pp60-70.

[28] G.Orlando, C.Paar, A high performance reconfigurable elliptic curve processor for GF(2^m), Second International Wrkshop on Cryptographic Hardware and Embeded Systems (CHES 2000), pp 41-56.

[29] D. Hankerson, A. Menezes, and S. Vanstone, Guide to Elliptic Curves Cryptography, Springer-Verlag, 2003.