Galois Field Operational unit For Elliptic Curve Cryptography Digital Signature

Mohammed Kadhim Rahma

Computer Engineering Department, Lviv Polytechnic National University, UKRAINE, Lviv, S. Bandery Street 28a,
E-mail: muhamed_kadhem@yahoo.com

Abstract – Cryptography is the most standard and efficient way to protect the security of data transactions. An efficient cryptosystem must be one that is strong enough to ensure a high level of security for reliable transmission of information. Elliptic curve cryptography is one such type of public key and private key cryptosystem based on small key size with high efficient speed up of cryptography process. Elliptic curve cryptography is an alternative to traditional techniques for public key cryptography. It can be called the future generation of public key systems since it involves less number of bits suitable for resource constrained and wireless applications without compromising on the security level. The proposed architecture for elliptic curve scalar is based on Point multiplication algorithm. It was also generated (Extension Field) assimilation by EF(3^{87}) where GF(2^{173})& EF(3^{87}) fields have approximately the same number of elements, and results were compared and implemented.

Key words – Galois field *GF*(2^n), *Extension Field EF*(p^m) , Elliptic Curve Cryptography, Digital Signature, Operational unit For scalar multiplication .

I. Introduction

This work introduces elliptic curve Cryptography Digital Signature over Operational Unit architectures designed to take advantage of the features offered by reprogrammable hardware, in particular field programmable gate array (FPGA) hardware.

The first elliptic curve cryptosystems were independently proposed in 1985 by Neil Koblitz and Victor Miller. Since its inception, elliptic curve cryptography has been the subject of extensive cryptanalysis.

Today, elliptic curve cryptosystems are being for deemed secure for commercial as well as Government use. Based on today's cryptanalytic knowledge, elliptic curve cryptosystems offer security comparable to that of traditional public-key cryptosystems.

II. Design HW Operational unit ECDSA

This section specifies an arithmetic unit architecture for GF(2^{173}) arithmetic and GF(p^m) arithmetic. The main components of the arithmetic unit are a multiplier, inversion and adder, also a register file, and a zero test circuit. The multiplier is used to compute multiplications and squares. The adder is used to compute additions and subtractions over GF(2^{173}).The inversion is used to compute division. The GF(p^m) arithmetic unit introduced here is based on a new range level of 173 bits where used prime (3) and m (87) for architecture arithmetic unit introduced here as work develops to GF(2^{173})architecture.

The elliptic curve processor architectures is in fig.1 introduced in this work. In this figure, the elliptic curve processor is composed by an arithmetic unit and two programmable processors the main controller and the arithmetic unit controller.

Main Controller (MC): The MC is a reduced instruction set processor, Fig.2. The MC is responsible for orchestrating the point multiplication process. In the computation of a point multiplication is using mixed or projective coordinates.

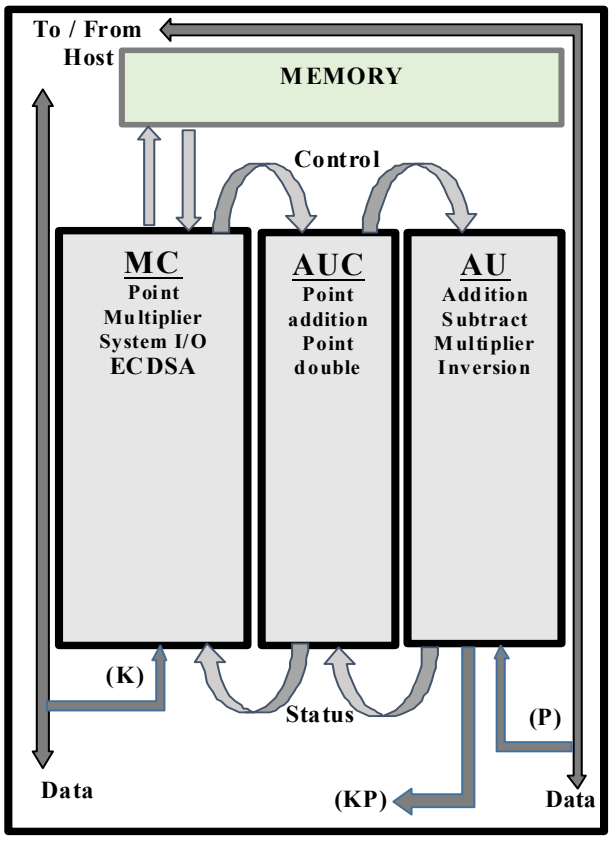

Fig. 1. Operation unit for ECDSA architecture

Arithmetic Unit Controller (AUC): The AUC is a reduced instruction set processor, fig.3. The AUC is responsible for processing MC commands. The AUC processes MC commands by guiding the AU through the computation of the necessary field arithmetic operations. In the computation of point multiplications, the AUC is responsible for guiding the AU in the computation of point additions, point subtractions, point doubles, coordinate conversions, multiplication and field inversions.

Arithmetic Unit (AU): The AU is the main processing engine of an elliptic curve processor. The AU performance dictates the performance of the elliptic curve processor. For high performance elliptic curve processors, the AU complexity also dictates the complexity of the elliptic curve processor, because for these implementations the aggregate complexity of the MC and the AUC is low in comparison with the complexity of the AU. The AU is responsible for performing field additions, subtractions, multiplications, inversion, and comparisons of finite field elements. The AU is also responsible for storing elliptic curve parameters, precomputed values, and temporary values. Fig. 4 shows a functional diagram of an AU. The adder, subtracted, multiplier, and inversion functional blocks represent the arithmetic functions performed by the AU. The zero test function (Reset) is using in the comparison of finite field elements.

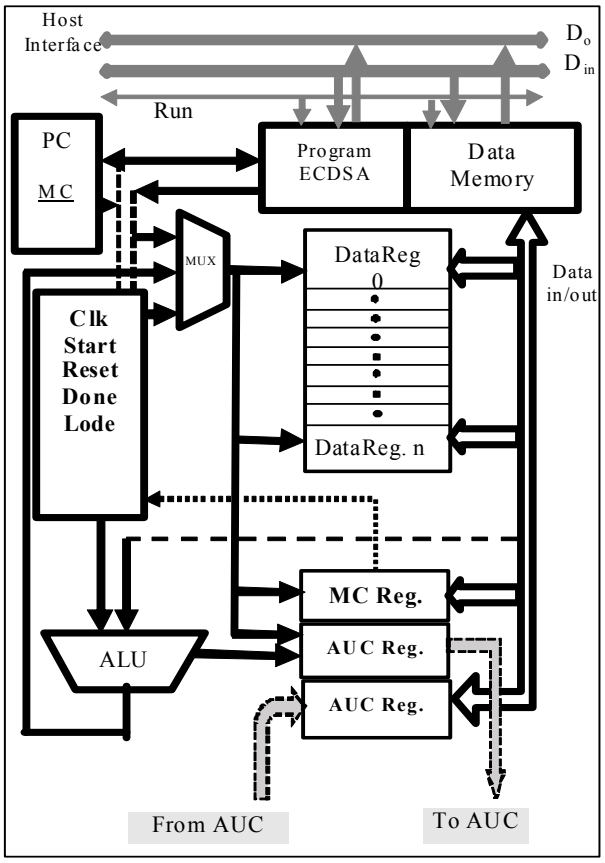

Fig. 2. Main controller (MC)

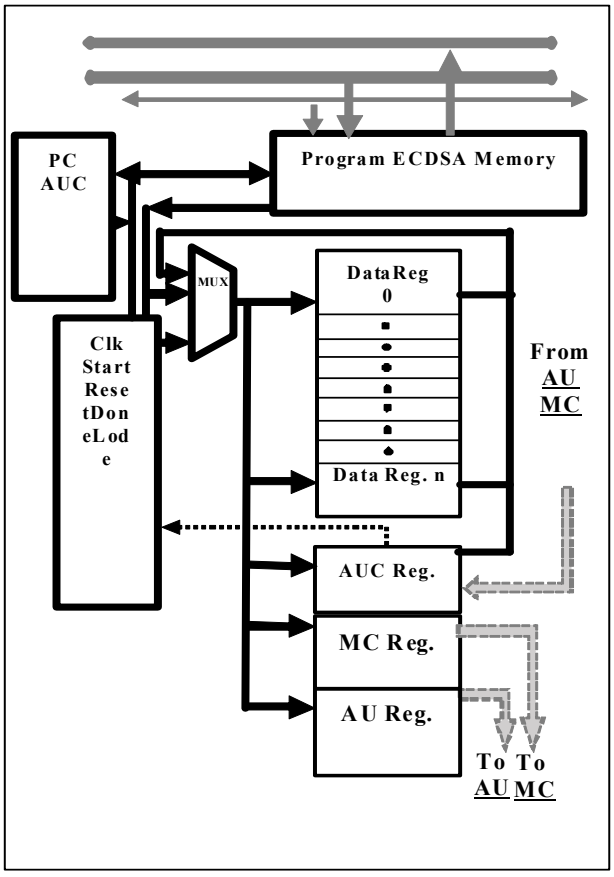

Fig. 3. Arithmetic unit controller (AUC)

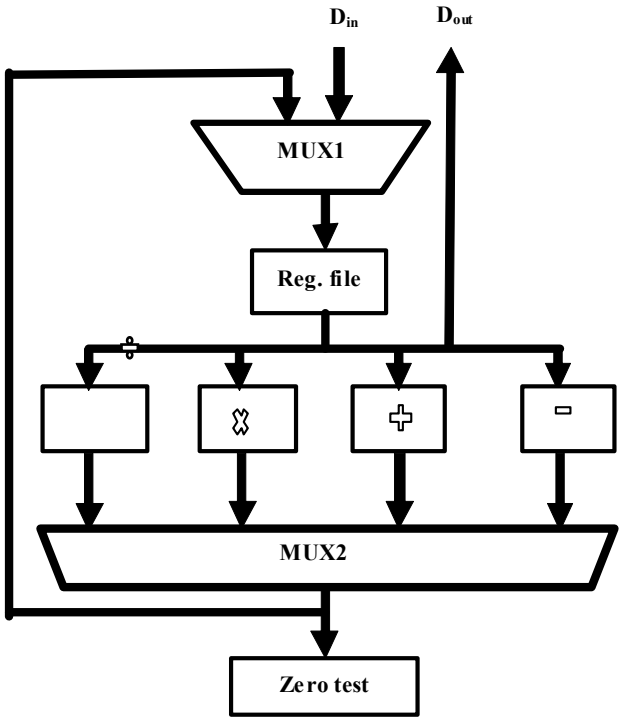

Fig. 4. Block diagram of the arithmetic unit

III. ECC Operations Hierarchy

Protocol Elliptic Curve Digital Signature Algorithm (ECDSA) is the number of mathematical operations hierarchy associated with the control unit according to the rules of the algorithm ECDSA, fig.5.

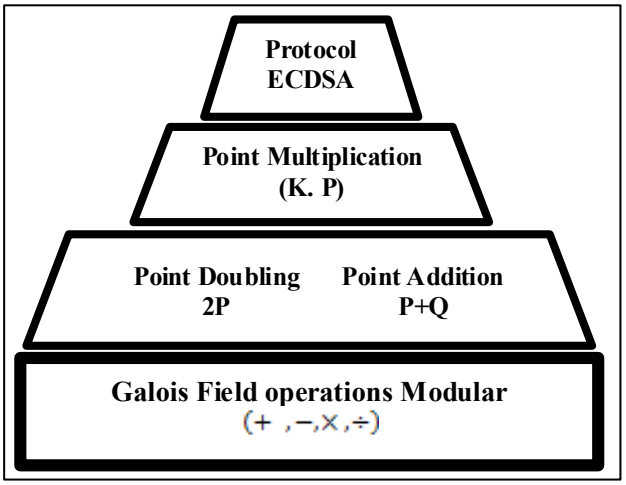

Fig. 5. ECC Operations Hierarchy

There are 4 levels in Fig.5. First level is basic Galois Field operations.GF addition, GF subtraction, GF multiplication, GF inversion. Second level is Elliptic Curve point operations or Scalar Point Multiplication. Add, Point Double, Arithmetic unit control level. Third Level is Elliptic Curve point operation level for the fundamental and most time-consuming operation, also it is Main control unit. Fourth Level is ECC protocol ECDSA level.

IV. Operations unit of GF(2^{173})

Generally, there are important arithmetic operations over the binary Galois Field *GF(2^173)*, which includes addition, multiplication and inversion with represent polynomial base with irreducible binary polynomials $f(x) = x^{173} + x^8 + x^5 + x^2 + 1$ over $GF(2^{173})$.

Addition and Subtraction: The addition and subtraction operations in finite field binary arithmetic are simple XOR operation.

Multiplication: The multiplication of $GF(2^{173})$ finite fields elements *A(x), B(x)* can be performed in many ways as [1],[2] also we can use two type polynomial and normal base as [3] that implemented $GF(2^{521})$. Montgomery Multiplier implemented as shown [4] but in this work we use LSB-first algorithm which computes c(x) = a(x).b(x) mod *f(x)* where *f(x)* is the irreducible polynomial of the field, as shown Fig.6.

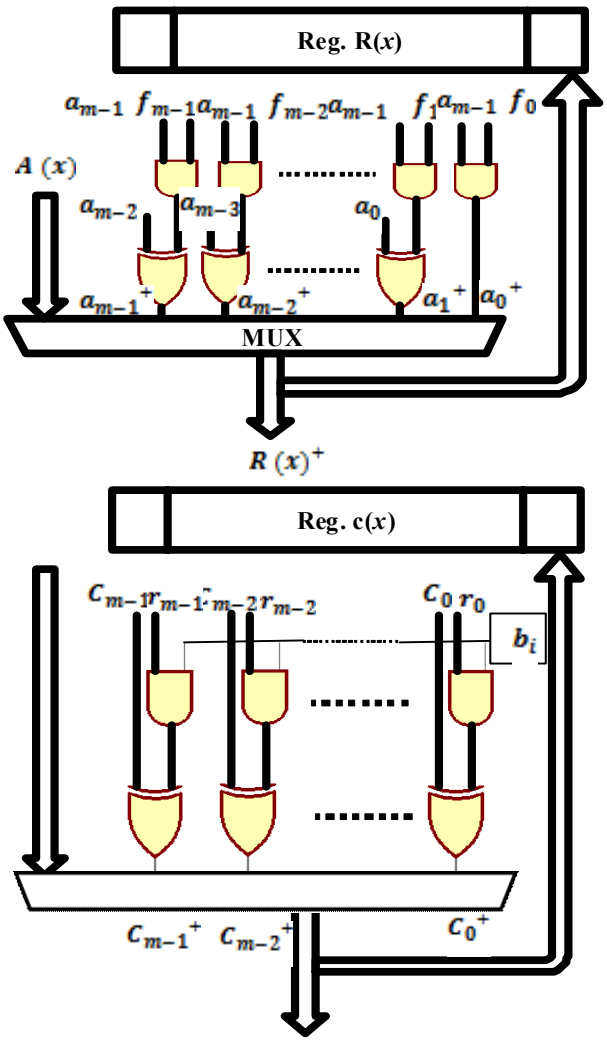

Fig. 6. Interleaved multiplier, computation of c(x) +bi . R(x); R(x).x mod f(x)

Division: it is the most time consuming operation in Galois field, Fig.7 because it has a multiplicative inverse as following: z(x) =g(x). $h(x)^{-1}$ mod f(x); **Where** $h(x)^{-1}$ = 1/h(x) **is** a multiplicative inverse. If g(x)=1 then z(x) inversion to h(x).

EC point operations over $GF(2^{173})$: Denote the elliptic curve over $GF(2^{173})$ such that $y^2 + xy = x^3 + ax^2 + b$ over $GF(2^{173})$ together with a point *O*, called the point infinity.

Point addition and doubling: Point doubling can be substituted by multiplication, over the finite field is constituted by the multiplication that done in the field arithmetic. Point addition is carried out in the field arithmetic. Let $f(x) = x^{173} + x^8 + x^5 + x^2 + 1$ irreducible polynomial and $P(x_1,y_1)$; $Q(x_2,y_2)$ are points in $GF(2^{173})$ as shown [5].

Point addition: $P+Q=(x_3, y_3)$;

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad (1)$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \quad (2)$$

$$\lambda = (y_1 + y_2) / (x_1 + x_2) \quad (3)$$

and point doubling: 2P=(x3, y3);

$$x_3 = \lambda^2 + \lambda + a \quad (4)$$

$$y_3 = x_1^2 + \lambda x_3 + x_3 \quad (5)$$

$$\lambda = x_1 + y_1 / x_1 \quad (6)$$

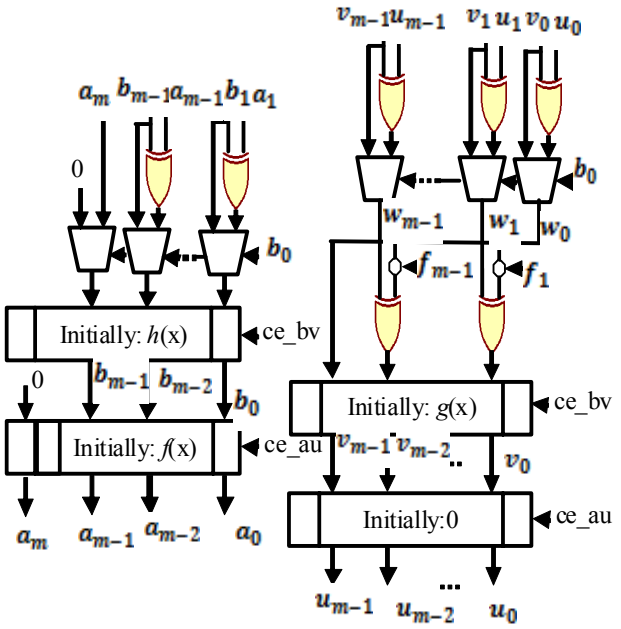

Fig. 7. Binary algorithm for polynomials division

V. Digit Size

After design and implementation Elliptic Curve Galois Field over $GF(2^{173})$ we can develop this work from the binary fields to extension fields to compare results and choose the best as Fig.8.

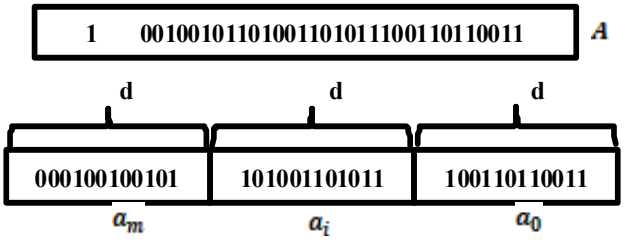

Fig. 8. Representation of large integers

VI. Operations unit over GF(p^m)

This operation is similar to Galois Field over *GF* (2^{173}), but different only in addition operation because the addition and subtraction have now different results. Therefore we need to implement subtraction into Arithmetic unit AU. Table 1 shows irreducible polynomial over *GF*(p^m).

TABLE 1

IRREDUCIBLE POLYNOMIAL OVER (p^m)

GF(p^m)	Irreducible Polynomial over *GF*(p^m)
GF(2^{173})	$f(x)= x^{173}+x^8+x^5+x^2+1$
GF(3^{87})	$f(x)= x^{87}+x^{26}+1$

Addition and Subtraction over *GF*(p^m): The addition of two elements *a* and *b* in *GF*(p^m) is performed by adding the polynomials *a*(*x*) and *b*(*x*) mod *f(x)*. The coefficients are added in the field *GF*(*p*). The subtraction of two elements *a* and *b* in *GF*(p^m) is performed by subtracting the polynomials *a*(*x*) and *b*(*x*), where the coefficients are subtracted in the field *GF*(*p*).Hardware finite field addition module p is the most important part because all finite field depending on it. The addition field consist of : (3*m) register of d bits (*a reg., b reg.*)= (*s reg.*), MUX, adder and comparator with p. Hardware finite field subtraction is similar to addition with b_i complement or a_i complement.

Multiplication over *GF*(p^m): This work will use array type LSE-first serial multiplier .Therefore, two registers c(α), *R*(α) are needed to store intermediate values. Since the degrees of (α_i. *B*(α) *mod f*(α)) are *m*-1 and *m* respectively, the modular reduction can be computed by only one subtraction. In general, the reduction is achieved by interleaved methods because non-interleaved methods require a lot of hardware resource. A basic method to implement a multiplication for *GF*(p^m) is a shift-and-add method.

The MOD unit in Fig. 9 is a part which computes $R(\alpha) = \alpha \,.\, R(\alpha) \bmod f(\alpha)$.

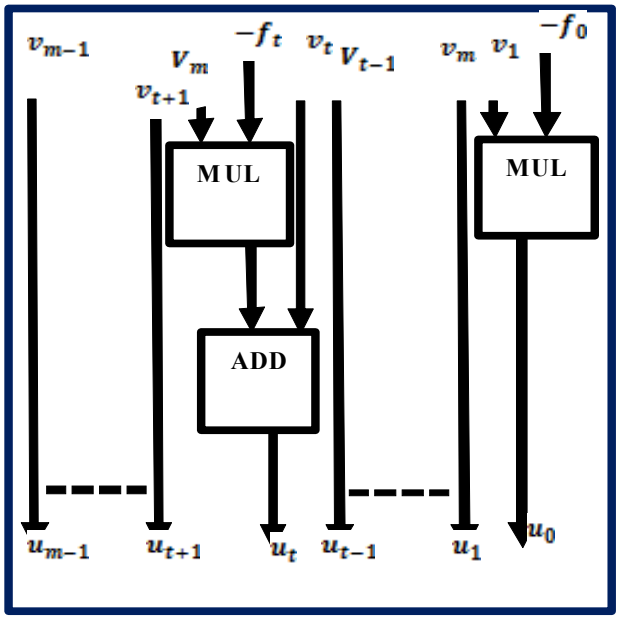

Fig. 9. MOD unit multiplier

Since the critical path of the MSE-first serial multiplier depends on the ADD unit and the MOD unit, the time delay is *m* MUL and *2m* ADD, where MUL and ADD refer to *GF* (*p*) multiplier and adder, respectively. For area complexity, *m* MUL at MUL unit, *m* ADD at ADD unit, and 2 MUL+1 ADD at MOD unite are required respectively. Therefore the total area complexity is (m + 2) MUL + (m + 1) ADD. As shown Fig.10.

Division: The inversion is a complex operation that is computed in (kP) operation. In this work we will select the square and multiply algorithm to perform inversion over *GF*(p^m). To apply this algorithm, the multiplicative inverse of the field element A can be obtain by recursive squaring and multiplication. The inversion operation is divided in three steps for one register: initialization operation, iteration of the squaring and multiplication, and finally the result. See Fig.11.

EC point operations over *GF*(p^m): An elliptic curve E that is define on a field p is expressed with the solutions of the Weierstrass equation and the point at infinity. General form of this equation given as follows:

$y^2+a_1xy+a_3y=x^3+a_2x^2+a_4x+a_6$. Let $P(x_1,y_1)$; $Q(x_2,y_2)$ are points in E*F*(p^m) as shown [6], point addition: $P+Q=(x_3, y_3)$;and point doubling: 2P=(x3, y3).

In this work, an elliptic curve over GF(pm) is given by $E:y^2=x^3+Ax+B$. Where A,B$\in$ GF(pm) and $4A^3+27B^2 \neq 0$.

The important operation is scaler multiplication as shown [7] where presented the total time to all operations (point adding, point doubling and arithmetic unit operations).

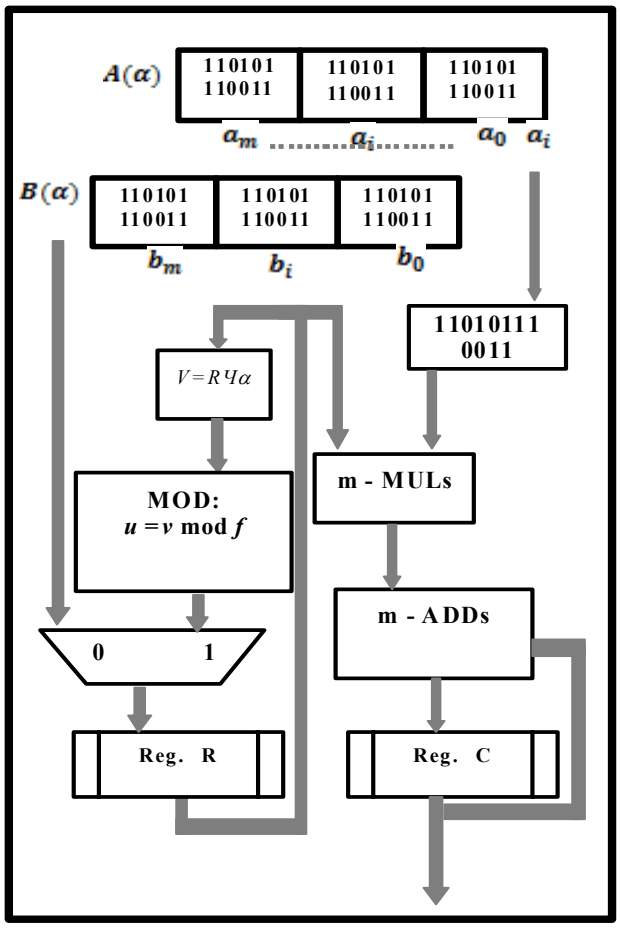

Fig.10. LSE-first serial multiplier for GF (p^m)

VII. Implementation and Results

In this work *GF(2^173)* numbers are 173 bits wide and checking of results is very bulky. In this matter, similar input/output have been written to the VHDL programming, in order to compare the execution steps, as well as the final results, timing is not taken into consideration in this specific stage for each field degree m, a pseudo-random curve is given, along with a Koblitz curve.

The pseudo-random curve has the form E: $y^2+xy = x^3+x^2+b$; a random curve over x^3 is denoted by _m. The Koblitz curve has the form E_a: $y^2+xy = x^3+ax^2+1$; where a= 0 or 1.For each pseudorandom curve, the cofactor is h = 2. The cofactor of each Koblitz curve is h = 2 if a = 1, and h = 4 if a = 0. A Koblitz curve over x^3 is denoted by k_m.

EC point doubling and EC point adding in Tables 2, 3. The results of arithmetic unit in Table 4. The results of compared to GF (2^{233}, 2^{173}, 2^{163}) in Table 5.The results of point multiplications in Table 6.

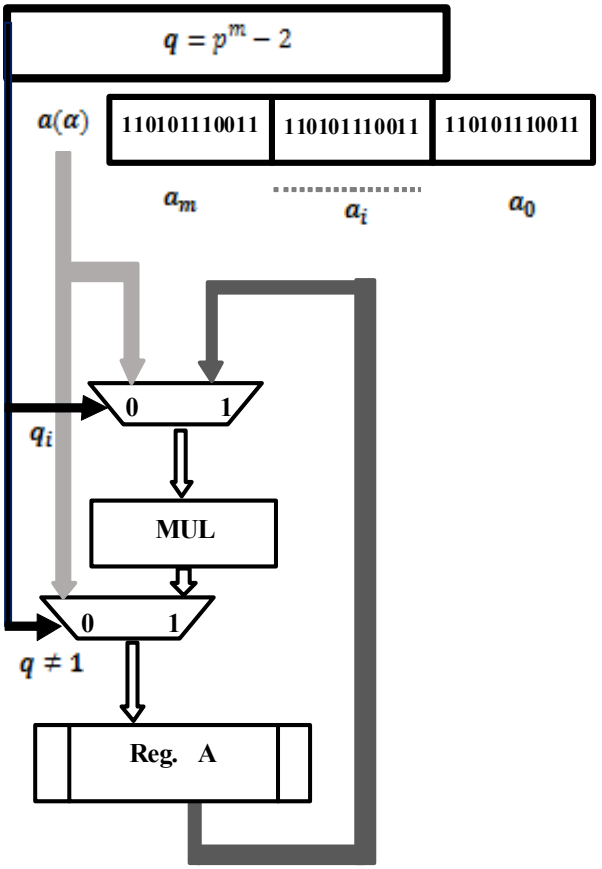

Fig. 11. Square and multiply for inversion GF(p^m)

TABLE 2

EC POINT DOUBLING

Operation name	Operation number	Clock Cycles
INV	1	346
MUL	3-1 synch.	346
ADD	3	3
----	----	695

Total Time EC point doubling = 69500 ns +delay.

TABLE 3

EC POINT ADDITION

Operation name	Operation number	Clock Cycles
INV	1	346
MUL	2	346
ADD	6-1 synch.	5
----	----	697

Total Time EC point adding = 69700 ns +delay.

TABLE 4

RESULTS AU

FF-Level Operation	Clock Cycles
FF-Mult.	173
FF-Add.	1
FF-inver.	346

TABLE 5

GF(2^{233},2^{173},2^{163})

GF Operation	MAX _Frequency
$GF(2^{233})$	136.323MHz
$GF(2^{173})$	142.857 MHz
$GF(2^{163})$	169.477MHz

TABLE 6

RESULTS SCALAR MULTIPLICATION

Operation	Clock Cycles
EC-Double	695
EC-Add	697
k·P	121,100
Time(k·P)	12,110,000 ns

Conclusions

This work introduces elliptic curve processor architectures suitable for the computation of point multiplications for curves defined over fields *GF(2^n)* and fields *GF*(p^m) where (n=173 bit and digits), (m= 87 digits) and (174 bits because p is prime and consist of 2 bits). The work presented here concentrated on the development of a high performance elliptic curve processor for the computation of point multiplications for curves defined over fields *GF*(p^m).This work compared prototyped implementations of elliptic curve processors suitable for the computation of point multiplications for curves defined over fields *GF(2^n)* and curves defined over fields *GF*(p^m).

This work compared prototyped implementations of elliptic curve processors suitable for the computation of point multiplications for curves defined over fields GF(2^n) and curves defined over fields GF(p^m).

In summary, this work defines elliptic curve processor architectures suitable for the computation of point multiplications for curves defined over fields *GF*(2^n) and

curves defined over fields $GF(p^m)$. These architectures are well suited for implementation in FPGAs, as it was proved with prototyped implementations: the fastest prototyped $GF(2^n)$ processor can compute an arbitrary point multiplication for curves defined over fields $GF(2^{173})$ is 12,110,000 nanoseconds and the maximum frequency F ≥100 MHz = 142.85714 MHz. The programmability of the processors allows them to incorporate new point multiplication algorithms without the need for reconfiguration. The wide range of time-area options presented for the different processors allows designers to tailor the presented architectures according to their cost-performance goals. The optimization of the processor architectures for FPGA technology allows implementations to evolve with advancements in FPGA technology; for example, as FPGAs become faster, processor implementations ported to faster FPGAs will achieve higher performance. In addition, as the densities of FPGAs increase the ability to develop faster processors increases. Finally, FPGA's re-configurability allows designers to use optimized point multiplication processors for different finite fields.

References

[1] V. Hlukhov, R. Elias and A. Melnyk., "Features of the FPGA-based Galois field GF(2^m) elements sectional multipliers with extra large exponent.," "Computer-Integrated Technologies: education, science and industry, vol. 12, p. 103 – 106, 2013.

[2] A. Hlukhova, "Results of Galois Field Elements Multipliers Structural Complexity Evaluation," Computer Science & Engineering 2013, vol. 12, p. 21–23, 2013.

[3] E. Rodrigue, "Design of an Elliptic Curve Cryptography Using A Finite Field Multiplier in GF(2^{521}).," Computer Systems and Networks, vol. 658, pp. 144 - 149, 2009.

[4] S. A. Karim, Reneesh C.Zacharia and Rijo Sebastian, "VLSI Implementation of Montgomery Multiplier in Finite Field Arithmetic for ECC over GF(2^{163})," nternational Journal of Scientific Engineering and Technology, vol. 3, no. 8, p. 4, 2014.

[5] K. H and P. Rosen, HANDBOOK OF DISCRETE MATHEMATICS and ITS APPLICATIONS, Boca Raton, FL 33487-2742: Taylor & Francis Group, LLC, 2013, p. 1068.

[6] W. Stallings, Cryptography and Network Security Principles and Practice, Sixth Edition ed., New Jersey, United States of America: Library of Congress Cataloging-in-Publication Data on file, 2014.

[7] Abdalhossein Rezai and Parviz Keshavarzi, "An Efficient Scalar Multiplication Algorithm for Elliptic Curve Cryptography Using a New Signed-Digit Representation," Isfahan University of Technology (IUT), rezaie@acecr.ac.ir, Semnan University, pkeshavarzi@semnan.ac.ir, Isfahan, Iran, 2014.