

DESIGN OF AN ELLIPTIC CURVE CRYPTOPROCESSOR OVER $GF(2^{163})$

*Vladimir Trujillo-Olaya, Jaime Velasco-Medina, Julio C. López-Hernández**

Grupo de Bionanoelectrónica, Escuela EIEE, Universidad del Valle, Cali, Colombia

* Instituto de computacao, UNICAMP, Campinas, Brasil

E-mail: vlatruo@univalle.edu.co, jvelasco@univalle.edu.co, jlopez@ic.unicamp.br

ABSTRACT

This article presents the design of a cryptoprocessor to implement the elliptic curve point multiplication over $GF(2^{163})$. The arithmetic operations were carried out using gaussian normal basis, and the elliptic curve point multiplication kP was implemented using Lopez-Dahab's algorithm. The processor was designed using schematic capture and VHDL description, the synthesis and the simulation was carried out using Quartus II of Altera, and the design was synthesized on the device EP2A70B724C7. The simulation results show that the processor presents a very good performance using low area. In this case, 0.819 ms to calculate kP and 110K gates.

DESIGN OF AN ELLIPTIC CURVE CRYPTOPROCESSOR OVER $GF(2^{163})$

*Vladimir Trujillo-Olaya, Jaime Velasco-Medina, Julio C. López-Hernández**

Grupo de Bionanoelectrónica, Escuela EIEE, Universidad del Valle, Cali, Colombia

* Instituto de computacao, UNICAMP, Campinas, Brasil

E-mail: vlatruo@univalle.edu.co, jvelasco@univalle.edu.co, jlopez@ic.unicamp.br

ABSTRACT

This article presents the design of a cryptoprocessor to implement the elliptic curve point multiplication over $GF(2^{163})$. The arithmetic operations were carried out using gaussian normal basis, and the elliptic curve point multiplication kP was implemented using Lopez-Dahab's algorithm. The processor was designed using schematic capture and VHDL description, the synthesis and the simulation was carried out using Quartus II of Altera, and the design was synthesized on the device EP2A70B724C7. The simulation results show that the processor presents a very good performance using low area. In this case, 0.819 ms to calculate kP and 110K gates.

1. INTRODUCTION

New techniques and efficient cryptographic algorithms are used to exchange confidential data through internet or protect confidential information. However, the process of encryption and decryption of data must be carried out by considering the minor possible time. Then taking into account this consideration and other ones, the cryptographic solutions implemented in hardware are more efficient than software implementations. Hardware implementations present high speed for data processing, high physical security and low cost. Cryptographic algorithms implemented in hardware are physically more secure due to these cannot be easily read or modified by an external agent. In order to protect confidential information, the public key cryptography is used and their two main applications are the private key exchange and the digital signature. In this context, public key cryptography based on elliptic curves is wide used because it presents higher security per key bit. Additionally, the elliptic curve cryptosystems (ECC) are systems of public key, which can be used in applications where the computation resources are limited such as smart cards and cellular telephones. The ECC systems are included in the NIST and ANSI standards, and the principal advantage over other systems of public key like RSA is the size of the parameters, which are very small,

however the ECC systems provide the same level of computational security.

On other hand, the algorithms used for Elliptic Curve Cryptosystems can be divided hierarchically into three levels. The arithmetic level, the group operation level and the encryption level. Therefore, in order to achieve efficient algorithm implementations it is evident to reach the best optimization for each level.

This article presents the design of a cryptoprocessor for elliptic curves over $GF(2^{163})$ using Gaussian Normal Basis, which have not been considered in the cryptoprocessors presented in the literature. In this case, the processor designed presents a good performance using low area.

This work is organized as follows. Initially, section 2 presents a summary of previous works, sections 3 and 4 present the basic concepts on $GF(2^m)$ arithmetic and the basic theory on elliptic curves. In section 5, the architecture design of the cryptoprocessor is described. In section 6, the simulation results are presented. Finally, section 7 presents the conclusions and the future work.

2. PREVIOUS WORK

In the literature are presented several designs of elliptic curve cryptoprocessors. Orlando and Paar present a cryptographic processor in [1], the processor performs the point multiplication based on Montgomery Scalar Multiplication in projective space over $GF(2^{167})$ using polynomial basis. In this case, the processor presents a performance of 0.21 ms per point multiplication and it is implemented on FPGA XCV400E and works at a frequency clock of 76.7 MHz and uses 570K gates.

N. Gura, S. Chang and H. Eberle present a cryptographic accelerator in [2], the hardware performs the point multiplication based on López-Dahab algorithm over $GF(2^{163})$ using polynomial basis. In this case, the processor presents a performance of 1.554 ms per point multiplication and it is implemented on FPGA XCV2000E-FG680-7 and works at a frequency clock of 66.4Mhz.

K. H. Leung, W. K Wong and P. H. W Leong present a cryptoprocessor in [3], the processor performs point multiplication based on double and add method over

$GF(2^{113})$ using optimal normal basis. In this case, the processor presents a performance of 3.7 ms per point multiplication and it is implemented on FPGA XCV300 and works at a frequency clock of 45 MHz and uses 320K gates.

M. Ernst and S. Klupsch, present a processor in [4], the processor performs point multiplication based on double and add method over $GF(2^{270})$ using optimal normal basis. In this case, the processor presents a performance of 1.3 ms per point multiplication and it is implemented on FPGA XC4085XLA and works at a frequency clock of 37 MHz and uses 180K gates.

The cryptoprocessor presented in this article uses Gaussian Normal Basis, while the cryptoprocessors presented in the literature use polynomial basis. The processor is implemented on the FPGA EP2A70B724C7 and works at a frequency clock of 82.43MHz; the performance is 0.819 ms per point multiplication over $GF(2^{163})$ and uses 110K gates. In this case, the processor designed presents a good performance and uses low area, which is very suitable for applications that require low area, good speed and low consumption power, such as smart cards and cellular telephones.

3. $GF(2^m)$ ARITHMETIC FOR GAUSSIAN NORMAL BASIS

3.1 Concepts on gaussian normal basis

Normal Basis representations of an element in the finite field $GF(2^{163})$ have the computational advantage that squaring an element can be done very efficiently. However multiplying distinct elements, can be cumbersome in general for this reason, ANSI X9.62 specifies that *Gaussian Normal Basis* be used, for which multiplication is both simpler and more efficient [5].

A normal basis for $GF(2^m)$ is as follows:

$\{\beta, \beta^2, \beta^{2^2}, \dots, \beta^{2^{m-1}}\}$, where $\beta \in GF(2^m)$

where, any element $\alpha \in GF(2^m)$ can be written as follows:

$$\alpha = \sum_{i=0}^{m-1} a_i \beta^{2^i} \text{ where } a_i \in \{0,1\}.$$

The *type T* of a Gaussian Normal Basis (GNB) is a positive integer, measuring the complexity of the multiplication operation with respect to that basis. Generally, the type *T* of smaller value allows to make a more efficient multiplication. For a given *m* and *T*, the field $GF(2^m)$ can have at most one GNB of type *T*.

A GNB exists whenever *m* is not divisible by 8. Let *m* be a positive integer and let *T* be a positive integer. Then the type *T* of a GNB for $GF(2^m)$ exists if and only if $p = Tm + 1$ is prime.

If $\{\beta, \beta^2, \beta^{2^2}, \dots, \beta^{2^{m-1}}\}$ is a Gaussian Normal Basis in the finite field $GF(2^m)$, then the element

$\alpha = \sum_{i=0}^{m-1} a_i \beta^{2^i}$ is represented by the binary string

$(a_0 a_1 a_2 \dots a_{m-1})$ where $a_i \in \{0,1\}$

In this case, the multiplicative identity element is represented by the bit string of all 1's. While the additive identity element is represented by the bit string of all 0's.

An important result for the arithmetic of the Gaussian Normal Basis is the Fermat's Theorem. For all $\beta \in GF(2^m)$ so that:

$$\beta^{2^m} = \beta$$

This theorem is important to carry out the squaring of an element in the finite field $GF(2^m)$.

3.2 $GF(2^m)$ arithmetic for gaussian normal basis

The following arithmetic operations are defined on the elements of $GF(2^m)$ when using a GNB of type *T*:

□ *Addition:*

If $a = (a_0 a_1 a_2 \dots a_{m-1})$ and $b = (b_0 b_1 b_2 \dots b_{m-1})$ are elements of $GF(2^m)$, then $a + b = c = (c_0 c_1 c_2 \dots c_{m-1})$ where $c_i = (a_i + b_i) \bmod 2$.

□ *Squaring:*

Let $a = (a_0 a_1 a_2 \dots a_{m-1}) \in GF(2^m)$, then

$$\begin{aligned} a^2 &= \left(\sum_{i=0}^{m-1} \alpha_i \beta^{2^i} \right)^2 \\ &= \sum_{i=0}^{m-1} \alpha_i \beta^{2^{i+1}} = \sum_{i=0}^{m-1} \alpha_{i-1} \beta^{2^i} \text{ due to Fermat's} \end{aligned}$$

Theorem; $\beta^{2^m} = \beta$, then

$$a^2 = (a_{m-1} a_0 a_1 a_2 \dots a_{m-2})$$

In this case, squaring is a simple rotation of the vector representation.

□ *Multiplication:*

Let $p = Tm + 1$ and let $u \in GF(p)$ be an element of order *T*. Define the sequence $F(1), F(2), \dots, F(p-1)$ by: $F(2^i u^j \bmod p) = i$, for *i* from 0 until *m-1* and *j* from 0 until *T-1*.

If $a = (a_0 a_1 a_2 \dots a_{m-1})$ and $b = (b_0 b_1 b_2 \dots b_{m-1})$ are elements of $GF(2^m)$, then

$a \bullet b = c = (c_0 c_1 c_2 \dots c_{m-1})$, where

$$c_i = \sum_{k=1}^{p-2} a_{F(k+1)} \bullet b_{F(p-k)}$$

□ *Inversion:*

If $a \neq 0$ and $a \in GF(2^m)$, the inverse of a , is the unique element $c \in GF(2^m)$ for which $a \bullet c = 1$ ($c = a^{-1}$).

The algorithm used for inversion is based on the identity:

$$a^{-1} = a^{2^n - 2} = (a^{2^{n-1} - 1})^2$$

In [6], Itoh and Tsujii proposed a method that minimizes the number of multiplications to calculate the inversion, which is based on the following identities:

$$a^{2^{n-1} - 1} = \begin{cases} (a^{2^{\frac{n-1}{2}} - 1})^{2^{\frac{n-1}{2}}} \bullet a^{2^{\frac{n-1}{2}} - 1}, & \text{odd} \\ a(a^{2^{n-2} - 1})^2, & \text{even} \end{cases}$$

4. ELLIPTIC CURVE ARITHMETIC

4.1 Concepts on elliptic curves

An elliptic curve is defined by an equation of the form:

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_5 \quad (1)$$

Where a_1 until a_5 are constant elements of a field F , the equation (1) is valid for any field; however, for cryptographic purposes only the finite fields are important. Two finite fields are of special interest: elliptic curves over Prime Field ($F = GF(p)$), and elliptic curves over Galois Field of characteristic 2 ($F = GF(2^m)$), this is important to hardware implementations.

The group of elliptic points can be represented as:

$$E(F) = \{O, P_1, P_2, P_3, \dots, P_n\}$$

Where O is a special point (called point at infinity). It is possible to define an operation of addition of two points over $E(F)$ so that $(E(F), +)$ forms an abelian group with identity O .

4.2 Elliptic curves over $GF(2^m)$

In this work, the non-supersingular curves are considered, defined by the following equation:

(2)

Where $a, b \in GF(2^m)$, $b \neq 0$. The set $E(GF(2^m))$ consists of all points (x, y) , which satisfy the equation (2), together with a special point O called point at infinity.

If $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points on the elliptic curve and $R = (x_3, y_3) = P + Q$, then there are two cases: in the first case, $P \neq Q$ (point addition) and in the second case $P = Q$ (point doubling), then the coordinates of R can be computed as follows:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad (3)$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \quad (4)$$

$$\lambda = \begin{cases} \frac{y_2 + y_1}{x_2 + x_1}, & P \neq Q \\ \frac{y_1}{x_1} + x_1, & P = Q \end{cases}$$

Equations (3) and (4) require inversions for point addition and doubling operations. In situations where inversion in $GF(2^m)$ is expensive relative to multiplication, it may be advantageous to represent points using projective coordinates of which several types have been proposed.

In standard projective coordinates, the projective point $(X : Y : Z)$, $Z \neq 0$ corresponds to the affine point $(X/Z, Y/Z)$, the projective equation of the elliptic curve is the equation (5).

$$Y^2 Z + XYZ = X^3 + aX^2 Z + bZ^3 \quad (5)$$

Other set of projective coordinates was introduced by López-Dahab in [7]. In this case, the projective point $(X : Y : Z)$, $Z \neq 0$ corresponds to the affine point $(X/Z, X/Z^2)$, and the projective equation of the curve is the equation (6).

$$Y^2 + XYZ = X^3 Z + aX^2 Z^2 + bZ^4 \quad (6)$$

The point at infinity corresponds to $(1 : 0 : 0)$, while $-P = (X : X+Y : Z)$, the equations (7), (8) and (9) are used to doubling an elliptic point.

$2(X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ where,

$$Z_3 \leftarrow X_1^2 * Z_1^2, \quad (7)$$

$$X_3 \leftarrow X_1^4 + bZ_1^4, \quad (8)$$

$$Y_3 \leftarrow bZ_1^4 * Z_3 + X_3 * (aZ_3 + Y_1^2 + bZ_1^4) \quad (9)$$

The equations to point addition $P = (X_1 : Y_1 : Z_1)$ and $Q = (X_2 : Y_2 : 1)$ are as follows:

$$A \leftarrow Y_2 * Z_1^2 + Y_1 \quad (10)$$

$$B \leftarrow X_2 * Z_1 + X_1 \quad (11)$$

$$C \leftarrow Z_1 * B \quad (12)$$

$$D \leftarrow B^2 * (C + a * Z_1^2) \quad (13)$$

$$Z_3 \leftarrow C^2 \quad (14)$$

$$E \leftarrow A * C \quad (15)$$

$$X_3 \leftarrow A^2 + D + E \quad (16)$$

$$F \leftarrow X_3 + X_2 * Z_3 \quad (17)$$

$$G \leftarrow (X_2 + Y_2) * Z_3^2 \quad (18)$$

$$Y_3 \leftarrow (E + Z_3) * F + G \quad (19)$$

The doubling of an elliptic curve point requires 4 multiplications, while the addition of two elliptic curve points requires 8 multiplications.

4.3 Point multiplication on elliptic curves

The fundamental and most expensive operation underlying ECC is the point multiplication kP where k is an integer and P is an elliptic point. A single point multiplication requires multiple computations of point additions ($P \neq Q$) and point doubling ($P = Q$) which were described in Section 4.2. Calculating kP where P is a point on the curve, will yield a new point on the curve. This procedure forms the basic for public key cryptography using EC. Point multiplication is defined by repeated addition as follows:

$$kP = \underbrace{P + P + \dots + P}_{k \text{ times}}$$

To calculate kP of an efficient way, the López-Dahab's algorithm is used, which is based on an idea of Montgomery. Let $Q_1 = (x_1, y_1)$ and $Q_2 = (x_2, y_2)$ with $Q_1 \neq \pm Q_2$. Let $Q_1 + Q_2 = (x_3, y_3)$ and $Q_1 - Q_2 = (x_4, y_4)$. Then using the addition equations (3) and (4) it can be verified that:

$$x_3 = x_4 + \frac{x_1}{x_1 + x_2} + \left(\frac{x_1}{x_1 + x_2} \right)^2$$

Thus, the x -coordinate of $Q_1 + Q_2$ can be computed from the x -coordinates of Q_1 , Q_2 and $Q_1 - Q_2$. The iteration i of the López-Dahab's algorithm for determining kP computes $T_i = (lP, (l+1)P)$, where l is the integer given by the i leftmost bits of k . Then $T_{i+1} = (2lP, (2l+1)P)$ or $((2l+1)P, (2l+2)P)$ if the $(j+1)$ st leftmost bit of k is 0 or 1, respectively. Each iteration requires one doubling and one addition. After the last iteration, having computed the x -coordinates of $kP = (x_1, y_1)$ and $(k+1)P = (x_2, y_2)$, the y -coordinate of kP can be recovered as:

$$y_1 = x^{-1} \left((x_1 + x) \left[(x_1 + x)(x_2 + x) + x^2 + y \right] + y \right) \quad (20)$$

Equation (20) is derived using the addition equations (3) and (4) for computing the x -coordinate x_2 of $(k+1)P$ from $kP = (x_1, y_1)$ and $P = (x, y)$. The López-Dahab's algorithm uses standard projective coordinates and it is shown in Figure 1.

ALGORITHM: LÓPEZ-DAHAB POINT MULTIPLICATION.

Input: $k = (k_{t-1}, \dots, k_1, k_0)_2$, $k_{t-1} = 1$, $P(x, y) \in E(GF(2^{163}))$.

Output: kP .

1. If $k = 0$, or $x = 0$ then $kP = (0,0)$ and stop
2. $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.
{compute $x(P)$, $x(2P)$ }
3. **For** i from $t-2$ downto 0 do:
4. if $k_i = 1$ then:
5. $T \leftarrow Z_1$, $Z_1 \leftarrow (X_1 Z_2 + X_2 Z_1)^2$,
 $X_1 \leftarrow x Z_1 + X_1 X_2 T Z_2$
6. $T \leftarrow X_2$, $X_2 \leftarrow X_2^4 + b Z_2^4$, $Z_2 \leftarrow T^2 Z_2^2$
7. else:
8. $T \leftarrow Z_2$, $Z_2 \leftarrow (X_1 Z_2 + X_2 Z_1)^2$,
 $X_2 \leftarrow x Z_2 + X_1 X_2 T Z_1$
9. $T \leftarrow X_1$, $X_1 \leftarrow X_1^4 + b Z_1^4$, $Z_1 \leftarrow T^2 Z_1^2$
10. if $Z_1 = 0$, then $kP = (0,0) = 0_\infty$
11. $x_3 \leftarrow X_1 / Z_1$
12. if $Z_2 = 0$, then $kP = -P$ and stop
11. $y_3 \leftarrow (x + X_1 / Z_1) [(X_1 + x Z_1)(X_2 + x Z_2) + (x^2 + y) (Z_1 Z_2)] (x Z_1 Z_2)^{-1} + y$

Figure 1: López-Dahab point multiplication

4.4 Diffie-Hellman key exchange protocol

An application of elliptic curves is the Diffie and Hellman (D-H) key exchange protocol. In the cryptographic standards like ANSI, ISO, IEEE, NIST appear other cryptographic applications (digital signatures, encryption), which can be implemented on elliptic curves. The above EC applications use point multiplication as the main operation, then a cryptoprocessor could be used for all EC protocols.

The goal of the D-H key exchange protocol is to establish a secret session key between two parties over an insecure channel. The two parties Alice and Bob, want to establish a secret key without Oscar (the adversary) being able to computing this key. During the setup stage Alice and Bob obtain the public parameters that include the coefficient of an elliptic curve and a point of this curve. The rest of the algorithm is the following:

- | | |
|--|--|
| 1a) Alice generates a random key k_a | 1b) Bob generates a random key k_b |
| 2a) Alice computes a new point $Q = k_a \cdot G$ | 2b) Bob computes a new point $R = k_b \cdot G$ |
| 3a) Alice sends Q to Bob | 3b) Bob sends R to Alice |
| 4a) Alice computes $T = k_a \cdot R = k_a (k_b \cdot G)$ | 4b) Bob computes $T = k_b \cdot Q = k_b (k_a \cdot G)$ |

After the final stage of the algorithm, Alice and Bob can compute the shared session key. Oscar cannot regenerate the session key from the public parameters P, Q and R because the two random integers k_a and k_b generated by Alice and Bob are private and never were transmitted over the insecure channel. The security of this scheme relies on the discrete logarithm problem for elliptic curves, which is believed to be intractable for EC recommended by standards [8].

5. PROCESSOR ARCHITECTURE AND DESIGN

An elliptic curve cryptosystem is based on a cryptoprocessor, which allows to calculate the point multiplication over $GF(2^m)$ of an efficient way. In this case, the processor designed is based on the implementation of algorithms to carry out arithmetic operations over $GF(2^{163})$ and the point multiplication kP . In order to design the cryptoprocessor, its complexity is divided into three levels, which are shown in Figure 2. The encryption level allows to carry out the López-Dahab's algorithm, the group operation level allows to realize the point addition and doubling operations and the arithmetic level allows to calculate the arithmetic over $GF(2^{163})$.

The processor architecture uses an embedded RAM memory and several functional blocks, which allow to calculate the addition, squaring, multiplication and inversion arithmetic over $GF(2^{163})$. In order to carry out the López-Dahab's algorithm a FSM is used, which uses two FSMs. The first FSM allows to realize the point addition and doubling operations and the second FSM allows to realize the key treatment. Additionally, a main controller is used, which allows to control the I/O registers, generate the control sequences and initialize the cryptoprocessor. The processor architecture is shown in Figure 3

5.1 Functional blocks for $GF(2^{163})$ arithmetic

Two functional blocks are used to implement the $GF(2^{163})$ arithmetic. The blocks are an adder and a multiplier-rotator.

5.1.1 Adder block

The adder block allows to calculate the addition arithmetic over $GF(2^{163})$ and it is implemented using a bit-wise exclusive OR (XOR) function.

5.1.2 Multiplier-rotator block

The multiplier-rotator block calculates the multiplication, squaring and inversion arithmetic over $GF(2^{163})$, this block is implemented using a multiplier and a rotator.

Multiplier

The multiplier block allows to calculate the multiplication over $GF(2^{163})$ using gaussian normal basis. The design of the multiplier uses 4 F(U,V) arrays, which allow to generate the subindexes to determine the order of combination of each bit of the two input data of the multiplier. Also, the multiplier uses one shift register and two barrel shifters. A FSM is used to generate the sequence of control for the multiplication. The multiplier is shown in Figure 4.

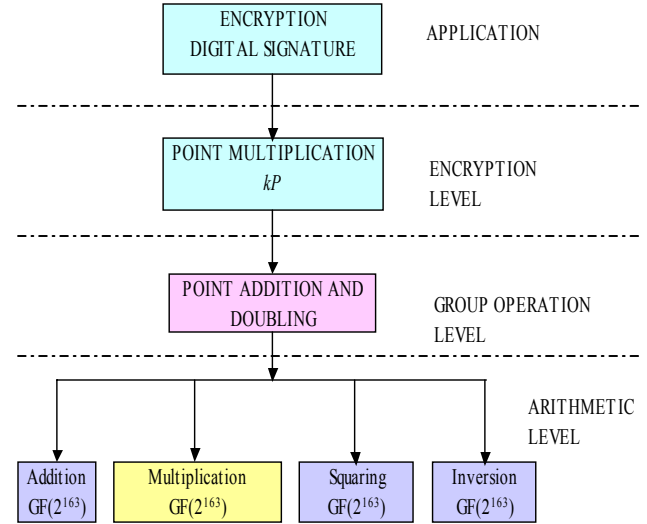


Figure 2: Hierarchical levels for the cryptoprocessor design

Rotator

Rotator allows to calculate the different powers of a normal basis, which will be inverted. In this case, the powers of the normal basis are: $2^1, 2^2, 2^5, 2^{10}, 2^{20}, 2^{40}$ and 2^{81} . The rotator block is a specialized block, which allows to rotate the normal basis 1, 2, 5, 10, 20, 40 and 81 bits.

5.2 RAM Memory

A RAM memory is used to storage the parameters of the elliptic curve, which are the coordinates of the points and the key. Also, the RAM memory stores the results of different arithmetic operations that are realized by each functional block of the cryptoprocessor.

5.3 Control units

The cryptoprocessor uses two control units, which are implemented using finite state machines. The first control unit allows to implement the López-Dahab algorithm using two FSMs. The first FSM allows to realize the point addition and doubling operations and the second FSM allows to realize the key treatment. This FSM shifts the bits of the key and evaluate each bit of the key from the most significant bit until the less significant bit. In this

case, depending of the value of the bit, the cryptoprocessor performs the point addition and doubling operations. The second control unit is used to carry out the

main control of the cryptoprocessor, which allows to control the I/O registers, generate the control sequences and initialize the cryptoprocessor.

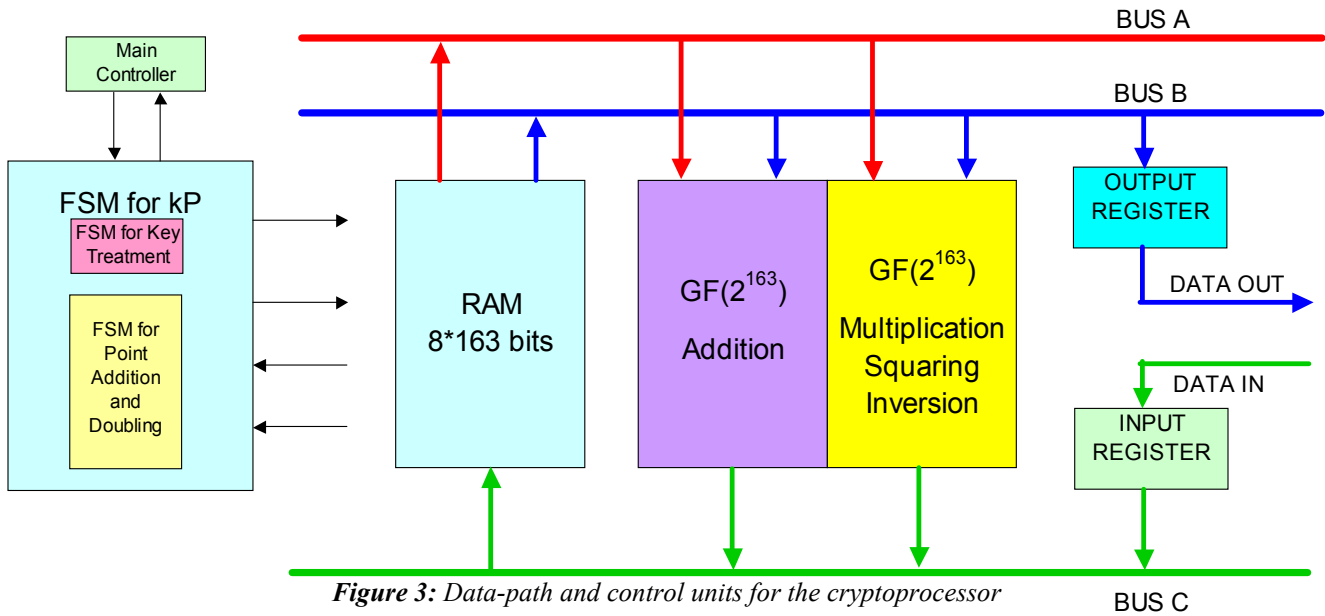


Figure 3: Data-path and control units for the cryptoprocessor

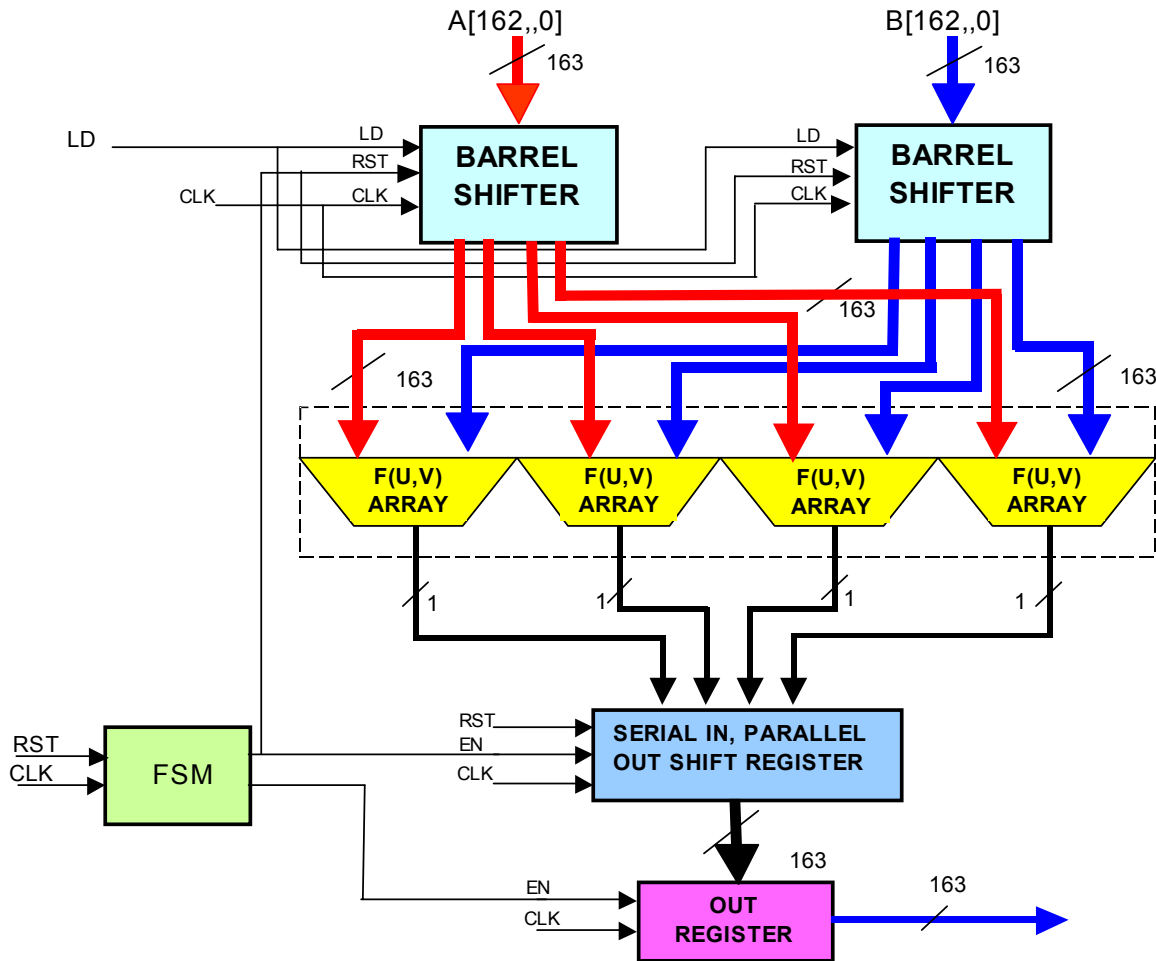


Figure 4: $GF(2^{163})$ multiplier architecture

6. SIMULATION RESULTS

In order to verify the performance of the cryptoprocessor, the Diffie-Hellman key exchange scheme is simulated. The parameters recommended by NIST for the elliptic curve over $GF(2^{163})$ using gaussian normal basis are:

$n = 0x85E25BFE5C86226CDB12016F7553FD0E693A268$
 $a = 1$
 $b = 0x6645F3CACF1638E139C6CD13EF61734FBC9E3D9FB$
 $P_x = 0x311103C17167564ACE77CCB0C681F886BA54EE8$
 $P_y = 0x333AC13C6447F2E67613BF7009DAF98C87BB50C7F$

NIST uses the P_x and P_y coordinates like generator point. The coordinates of the elliptic point $P = (x, y)$ that were taken as example are:

$x = 0x311FB9BE0651AB02E87211FDB571791440884144E$
 $y = 0x414C313E83210D531729381D314C9F0C56821A3DB$

The cryptoprocessor is implemented on the FPGA EP2A70B724C7, and the simulation and synthesis were carried out using Quartus II design tool of Altera Professional version 2.0.

In order to present the performance results for different cryptoprocessors presented in the literature, in Tables 1 and 2 are shown the performance results for arithmetic and encryption levels respectively. Also, in Table 3 are shown the performance and area results. From Table 3, it is possible to mention that the cryptoprocessor

designed presents a good performance and uses low area, which allows to integrate it into real applications, such as smart-cards and cellular telephones.

7. CONCLUSIONS AND FUTURE WORK

In this article the design of a cryptoprocessor is presented which can be used over elliptic curve cryptographic systems. The cryptoprocessor realizes arithmetic operations over $GF(2^{163})$ using gaussian normal basis, and uses the López-Dahab Algorithm for the point multiplication kP . Then, the main operation of the cryptoprocessor is the point multiplication whose performance depends on the multiplication over $GF(2^{163})$. In this case, the group operation and encryption levels are implemented using finite state machines.

The processor was designed using schematic capture and VHDL description, was synthesized by Quartus II design tool of Altera and was implemented on the FPGA EP2A70B724C7.

The cryptoprocessor presents a good performance and it uses low area, which is very suitable for applications such as smart cards and cellular telephones.

The future work, will be oriented to design a multiplier over $GF(2^{163})$ using polynomial basis and to implement the point halving algorithm in order to compare the performance of gaussian normal basis versus polynomial basis, and López-Dahab's algorithm versus point halving algorithm.

Author	Arithmetic level (μs)				m bits	Platform
	Addition	Square	Multiplication	Inversion		
Gura et al [2]	10.03	0.090	0.9	4.95	163	XCV 2000E-G680- 66.4MHz
Fong et al [9]	0.032	0.185	1.058	10	163	C, Pentium III 1Ghz
Trujillo et al	0.013	0.013	0.673	6.14	163	EP2A70B724C7 82.43Mhz

Table 1: Performance results for the arithmetic level

Author	kP(ms)	m bits	Platform
Eberle et al [10]	0.30 Montgomery	163	XCV2000E-FG680-7 66.4Mhz
Kerins et al [12]	5.1 Double and add	151	VX2000Bg- 560 -6 40Mhz
	6.9 Double and add	176	
Fong et al [9]	1.203 López-Dahab	163	C, Pentium III 1Ghz
	1.057 Halving with NAF and sqrt		
	1.178 4-NAF		
Orlando et al [1]	0.21 Double and add	167	VX400E-8-BG432 76.7Mhz
Trujillo et al	0.819 López-Dahab	163	EP2A70B724C7 82.43Mhz

Table 2: Performance results for the encryption level

Author	Device	Bits	Kp (ms)	Gates
Leong et al [11]	FPGA (XCV300, 45 MHz)	113	3.7	320K
Ernst et al [4]	FPGA (XC4085XLA, 37 Mhz)	155	1.3	180K
Okada et al[13]	FPGA (EPF10K, 3 Mhz)	163	80.7	310K
Orlando et al [1]	FPGA (XCV400E, 76.7 Mhz)	167	0.21	570K
Trujillo et al	FPGA (EP2A70B724C7,82.43Mhz)	163	0.819	110K

Table 3: Performance and area results

8. ACKNOWLEDGMENT

This work was sponsored by Altera Corporation through the University Program. The authors give a special thanks to Mrs Ralene Marcoccia of Altera Corporation.

9. BIBLIOGRAPHY

- [1] G. Orlando, C. Paar, "A high performance reconfigurable elliptic curve for $GF(2^m)$ ", Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2000, Springer-Verlag, Lecture Notes in Computer Science 1965, 2000.
- [2] N. Gura, S. Chang Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, D. Stebila, "An End-to-End systems approach to elliptic curve cryptography.", http://www.research.sun.com/projects/crypto_paper_from_cheese.pdf
- [3] K.H. Leung, K.W. Ma, W.K. Wong and P.H.W. Leong, "FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor," *Proc. IEEE FCCM 2000*, pp. 68–76, Napa Valley.
- [4] M. Ernst, S. Klupsch, O. Hauck and S. A. Huss, "Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key Cryptosystems," *Proc. 12th IEEE Workshop on Rapid System Prototyping (RSP01)*, Monterey, CA, 2001.
- [5] D. Johnson and A. Menezes, "The elliptic curve digital signature algorithm (ECDSA)", Technical report CORR 99-34, University of Waterloo, 2000
- [6] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal bases", *Information and Computation*, 1988.
- [7] J. Lopez Hernández, R. Dahab, "Fast multiplication on elliptic curves over $GF(2^n)$ without precomputation", *Cryptographic hardware and embedded systems- CHES'99*, LNCS 1717, 1999, 316-327.
- [8] Martin C. Rosner, "Elliptic curve cryptosystem on reconfigurable hardware", Worcester Polytechnic Institute, May 1998. www.crypto.ruhr-uni-bochum.de/Abschlussarbeiten/texte/documents/ms_mrosner.pdf
- [9] Kenny Fong, Darrel Hankerson, Julio Lopez, Alfred Menezes, Matt Tucker, "Performance comparisons of elliptic curve systems in software", University of Waterloo. October 2001. <http://www.cacr.math.uwaterloo.ca/conferences/2001/ecc/hankerson.pdf>.
- [10] Hans Eberle, Nils Gura, Sheueling Chang Shantz, and Vipul Gupta "A Cryptographic Processor for Arbitrary Elliptic Curves over $GF(2^m)$ " http://research.sun.com/techrep/2003/sml_tr-2003-123.pdf
- [11] Philip H. W. Leong, and Ivan K. H. Leung, "A Microcoded Elliptic Curve Processor Using FPGA Technology", *IEEE transactions on very large scale integration (vlsi) systems*, vol. 10, no. 5, october 2002.
- [12] Tim Kerins, Emanuel Popovici, William Marnane, and Patrick Fitzpatrick, "Fully Parameterizable Elliptic Curve Cryptography Processor over $GF(2^m)$ ", Dept of Electrical and Electronic Engineering, University College Cork, College Rd., Cork City, Ireland.
- [13] S. Okada, N. Torii, K. Itoh and M. Takenaka, "Implementation of Elliptic Curve Cryptographic Coprocessor over $GF(2^m)$ on an FPGA," *Workshop on Cryptographic Hardware and Embedded Systems (CHES 2000)*, LNCS 1965, C.K. Koc and C. Paar Eds., Springer-Verlag, pp. 25–40, 2000.