Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 675161, 8 pages
http://dx.doi.org/10.1155/2013/675161

Research Article

Reconfigurable Architecture for Elliptic Curve Cryptography Using FPGA

A. Kaleel Rahuman and G. Athisha

Department of Electronics and Communication Engineering, PSNA College of Engineering and Technology, Dindigul-624 622, Tamil Nadu, India

Correspondence should be addressed to A. Kaleel Rahuman; kaleel23@gmail.com

Received 5 April 2013; Accepted 4 July 2013

Academic Editor: William Guo

The high performance of an elliptic curve (EC) crypto system depends efficiently on the arithmetic in the underlying finite field. We have to propose and compare three levels of Galois Field GF(2^{163}), GF(2^{193}), and GF(2^{256}). The proposed architecture is based on Lopez-Dahab elliptic curve point multiplication algorithm, which uses Gaussian normal basis for GF(2^{163}) field arithmetic. The proposed GF(2^{193}) is based on an efficient Montgomery add and double algorithm, also the Karatsuba-Ofman multiplier and Itoh-Tsujii algorithm are used as the inverse component. The hardware design is based on optimized finite state machine (FSM), with a single cycle 193 bits multiplier, field adder, and field squarer. The another proposed architecture GF(2^{256}) is based on applications for which compactness is more important than speed. The FPGA's dedicated multipliers and carry-chain logic are used to obtain the small data path. The different optimization at the hardware level improves the acceleration of the ECC scalar multiplication, increases frequency and the speed of operation such as key generation, encryption, and decryption. Finally, we have to implement our design using Xilinx XC4VLX200 FPGA device.

1. Introduction

Many hardware designs of elliptic curve cryptography have been developed, aiming to accelerate the scalar multiplication processes, mainly those based on the field programmable gate arrays (FPGAs). The application field of FPGAs has clearly outgrown the prototype-only use. More and more FPGA implementations are in an environment which used to be ASIC-only territory. When these applications are implemented on an FPGA, they need secure data communication. In this rapidly changing environment, the reconfigurability of an FPGA is a very useful feature which is not available on an ASIC. Secure public key authentication and digital signatures are increasingly important for electronic communications and coerce, and they are required not only on high powered desktop computers, but also on smart cards and wireless devices with severely constrained memory and processing capabilities. Cryptography offers a robust solution for IT security services in terms of confidentiality, data integrity, authenticity, and nonrepudiation. In fact, security deals mainly with the ability to face counterattacks [1], while speed and area, which represent the eternal trade-off, concern the ability to make intensive cryptographic processes, while keeping used hardware as low as possible. In other words, it is the ability to embed a strategic and strong algorithm in very few hardware, that is, finding an optimal solution to the one to many problem: portability against power consumption, speed against area, but the main issue in cryptography is security.

In the last decade, the approach of hardware implementing elliptic curve cryptography (ECC) algorithm knew a very concentrated contest, due essentially to the requirements of security, speed, and area constraints. Cryptography has become one of the most important fields in our life, due essentially to two factors: increase in secrecy and increase in breaking code or hackers in the other side. Organizations tend to increase their benefits by keeping their information system as transparent as possible. On the other hand, hackers and code or key breakers are being organized in kind of unofficial groups; this leads to being a step ahead before getting the codes breakdown. Scientists are tending to complicate the

Input: $A, B \in \mathrm{GF}(2^m)$
Output: $D = (D_0, D_1, D_2, \ldots, D_{m-1}) \in \mathrm{GF}(2^m)$, $D = A \cdot B$
(1) For $0 \le t \le m-1$
(2) For $0 \le s \le m-1$
(3) $D_{s+t+1} \leftarrow y_{s,s+1}$
(4) end for
(5) end for
(6) return D

Algorithm 1: Bit-level multiplication algorithm for $\mathrm{GF}(2^m)$.

reverse engineering process of the encryption system, at the same time, keeping encryption keys as low as possible. This issue is being tackled by many mathematics, mainly those working on elliptic curves [2].

Elliptic curve cryptosystems possess a number of degrees of freedom like Galois field characteristic, extension degree, elliptic curve parameters, or the fixed point generating the working subgroup on the curve. The beauty of this new field is potentially related to the simplicity of the operators used in the encryption process, to the nonsecure transmission constraints used in the exchange of the keys and to the enhanced complexity that might face hackers when unwanted information goes out of the organization. This paper focuses on the high-performance comparisons of hardware design of ECC over $\mathrm{GF}(2^{163})$, $\mathrm{GF}(2^{193})$, and $\mathrm{GF}(2^{256})$ (Table 4).

2. Elliptic Curve Cryptography

In 1985, Koblitz and Miller introduced the use of elliptic curves in public key cryptography called Elliptic curve Cryptography (ECC). Basically, the main operation of elliptic curves consists of multiplying a point by a scalar in order to get a second point; the complexity arises from the fact that, given the initial point and the final point, the scalar could not be deduced, leading to a very difficult problem of reversibility, or cryptoanalysis, called also the elliptic curve discrete logarithm problem [1].

The ECC algorithms with their small key sizes present nowadays the best challenge for cryptanalysis problems compared to RSA or AES, thus dealing with ECC will lead to smaller area hardware, less bandwidth use, and more secure transactions.

The attractiveness of ECC algorithms is that they operate on a Galois Field (GF), by means of two simple operations, known as the field addition and field multiplication, which define a ring over $\mathrm{GF}(P^m)$ where P and m are primes. In the particular case, where we deal with hardware designs, a binary field is preferred, where the couple, (P, m), defines the set of elliptic curves. In this work, $P = 2$ and $m = 163, 193$, and 256.

In this paper, we propose a high-performance elliptic curve cryptographic processor over $\mathrm{GF}(2^m)$, that is, $\mathrm{GF}(2^{163})$, $\mathrm{GF}(2^{193})$, and $\mathrm{GF}(2^{256})$. The proposed architecture is based on a modified Lopez-Dahab elliptic curve point multiplication algorithm and uses GNB for $\mathrm{GF}(2^m)$ field arithmetic. Three major characteristics of the proposed architecture use fast arithmetic units based on a word-level multiplier which adopts a parallelized point doubling and point addition unit with uniform addressing mode to utilize the benefits of GNB representation. Therefore, the proposed architecture leads to a considerable reduction of computational delay. The proposed architecture has the feature of modularity and a simple control structure; it is well suited to VLSI implementations (see Algorithm 1).

In this research, we present a hardware design of the elliptic curve cryptography scheme, using Montgomery scalar multiplication based on the "add and double" algorithm, targeting as a primary goal of an increase in the speed of the hardware and an optimization in the ensuing inverse component.

3. Materials and Methods

3.1. Hardware Design. The strategy of hardware executing the ECC algorithms reposes on the ability of making the scalar multiplication in the $\mathrm{GF}(2^m)$ in a very few clock [1]. While increasing m, implementations become very time and resource consuming. Most of the known architectures concern the acceleration of the multiplication process by modifying the elliptic equations by changing the Z coordinate term [3], or by multiplication scalability [4], or by using many serial and parallel Arithmetic units [5], or using High parallel Karatsuba Multiplier [6], those based on the Massy-Omura multipliers, or the work based on a hybrid multipliers approach, also some parallel approach approaches, or the new word level structure, or through the systolic architecture, or by using the half and add method, or by parallelizing both the add and double Montgomery algorithms [7].

The second problem concerns the inversion based on the Fermat little theorem, or the almost inverse algorithm based on Kali ski's research [8]. In order to concentrate on one of the problems, some modifications have been done on the ECC equations in order to postpone inversion to the last stage, while dealing only with the multiplication process.

3.2. Elliptic Curve Mathematical Background. ECC is based on the discrete logarithm problem applied to elliptic curves over a finite field. In particular, for an elliptic curve E that relies on the fact, it is computationally easy to find that

$$Q = kx \square P, \tag{1}$$

Input: Elliptic curve domain parameters (p, E, P, n)
Output: Public key Q and private key k
(1) Select $k\varepsilon_R[1, n-1]$
(2) Compute $Q = kP$
(3) Return(Q, k)

Algorithm 2: Elliptic curve key pair generation.

where P and Q = Points of the elliptic curve E and their coordinates belong to the underlying GF(2^m), k = A scalar that belongs to the set of numbers $\{1 \cdots \#G-1\}$, G is the order of the curve E.

Elliptic Curve Key Generation. Let E be an elliptic curve defined over a finite field GF(2^m). Let P be a point in E(GF(2^m)), and suppose that P has prime order n. Then, the cyclic subgroup of E(GF(2^m)) generated by P is $P' = \{\infty, P, 2P, 3P, \ldots, (n-1)P\}$. The prime p, the equation of the elliptic curve E, and the point P and its order n are the public domain parameters. A private key is an integer k that is selected uniformly at random from the interval $[1, n-1]$, and the corresponding public key is $Q = kP$ (see Algorithm 2).

An encryption is described in what follows.

Encryption. (1) User A-Alice first selects a random generator point (x, y) lying on the elliptic curve.

(2) Message (M) to be encrypted is coded on to an elliptic curve point $Pm = (xm, ym)$.

(3) Alice selects a random private key "nA" and then computes the public key as

$$PA = nA\,(x, y)\,. \tag{2}$$

(4) To encrypt her message, Alice uses her private key and Bob's (user B) public key.

(5) The encrypted message denoted by Cm is created as follows:

$$Cm = \{PA, (Pm + nA \cdot PB)\}$$
$$\text{PB is the public key of Bob—user B.} \tag{3}$$

The sender transmits the points $\{PA, (Pm + nA \cdot PB)\}$ to the recipient who uses her private key k to compute $kPA = k(nA \cdot (x, y)) = nA(k \cdot (x, y)) = nA \cdot R$, where R is the public key of the recipient.

The algorithm for an encryption is described in the following.

As it can be seen from the previous algorithm, point multiplication plays a major role during the encryption process. The same hold during decryption too. The encrypted message is then communicated to the receiver. The receiver, Bob, then decrypts the message using the decryption mechanism [9].

A decryption at the receiver end is as follows.

Decryption. When Bob receives the encrypted message, he first multiplies the public key of Alice, which happens to be the first point in the encrypted message with his private key NB.

(1) When Bob receives the encrypted message, he first multiplies the public key of Alice, which happens to be the first point in the encrypted message with his private key nB.

(2) The result of this is then subtracted from the second point, the cipher text.

(3) This gives him the original message Pm.

Importance of Elliptic Curve Cryptography. There are several criteria that need to be considered when selecting a family of public key schemes for a specific application. The principal ones are as follows:

(1) functionality,

(2) security,

(3) performance.

Measuring the Efficiency of Algorithms. The efficiency of an algorithm is measured by the scarce resources it consumes. Typically, the measure used is time, but sometimes other measures such as space and number of processors are also considered. It is reasonable to expect that an algorithm consumes greater resources for larger inputs, and the efficiency of an algorithm is therefore described as a function of the input size. Here, the size is defined to be the number of bits needed to represent the input using a reasonable encoding.

In the affine coordinate representation, a finite field point on GF(2^m) is specified by two coordinates x and y both belonging to GF(2^m) satisfying (4). The point at infinity has no affine coordinates.

In most ECC hardware designs, the choice of using three coordinates responds on avoiding the periodic division of (5), which consumes a lot of resources in terms of execution cycles, as well as memory and power consumption.

The advantage here is, Bob's private key is only known to him and not to anyone else and therefore only Bob can extract the original message by subtracting the product of his private key and Alice's public key with the second point [7].

Nowadays, there is no known algorithm able to compute k given P and Q in a subexponential time. The equation of a nonsupersingular elliptic curve with the underlying field GF(2^m) is presented in (4). It is formed by choosing the elements "a" and "b" within GF(2^m) with

$$y^2 + xy = x^3 + ax^2 + b. \tag{4}$$

A point is converted from a couple of coordinates to a triple system of coordinates using one of the transforms of

$$x3 = \left(\frac{y1 + y2}{x1 + x2}\right)^2 + \left(\frac{y1 + y2}{x1 + x2}\right) + x1 + x2 + a, \quad P \neq Q,$$

$$x3 = x1^2 + \left(\frac{b}{x1^2}\right), \quad P = Q,$$

$$y3 = \left(\frac{y1 + y2}{x1 + x2}\right)(x1 + x3) + x3 + y1, \quad P \neq Q,$$
$$y3 = x1^2 + \left(x1 + \frac{y1}{x1}\right)x3, \quad P = Q. \tag{5}$$

Thus, a point $P(x, y)$ is mapped into $P(x, y, z)$, that is; a third projective coordinate is introduced in order to "flatten" the equations and avoid the division. Projective coordinates allow us to eliminate the need for performing inversion. The startup transformation required for the design is simply done by initializing X, Y, and Z as follows: [10]

$$\{X = \square x,\ Y = \square y,\ Z = \square 1 \square\square\square\square\square\square\square\}. \tag{6}$$

Introducing the new tricoordinates into (4) becomes

$$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4. \tag{7}$$

The VHDL implementation will be based now on (7). After completion of the successive operations of addition and multiplication, back to two affine coordinates as follows:

$$\left\{x = \frac{X}{z},\ y = \frac{Y}{z^2}\right\}. \tag{8}$$

In order to make the different computations, the Montgomery point doubling and Montgomery point addition algorithms are used, mainly through the ingenious observation of Montgomery, which states that the Y coordinate does not participate into the computations and can be delayed to the first stage and it working with two projective coordinates [11] (see Algorithm 3).

In the Decryption, *Madd*() function is the point addition operation on the elliptic curve, *Mdouble*() is the point doubling computation, and *Mxy*() is the conversion of projective coordinates to affine coordinates. The reader is referred to (Lopez and Dahab 1999) [12] for detailed explanation. Function *Madd*(), *Mdouble*(), and *Mxy*() in the Decryption are defined as follows:

Madd(x_1, y_1, x_2, y_2, x)
$\{X \leftarrow x_1z_2x_2z_1 + x(x_1z_2 + x_2z_1)^2;$
$Z \leftarrow (x_1z_2 + x_2z_1)^2;$
return$(X, Z);$
$\}.$

Requiring 1 field squaring operations, 4 field multiplications, One has and two simple field additions.

Mdouble(x_1, z_1, b)
$\{X \leftarrow {x_1}^4 + b{z_1}^4;$
$Z \leftarrow {x_1}^2{z_1}^4;$
return $(X, Z);$
$\}.$

Requiring 4 field squaring operations, 2 field multiplications, and one simple field addition.

Input: $k = (k_{n-1}, k_{n-2}, \ldots, k_1, k_0)2$ with $k_{n-1} = 1$,
$P(x, y)\ \varepsilon E\ (\mathrm{GF}(2^m))$.
Output: $Q = KP$.
(1) Set $x_1 = x$, $z_1 = 1$, $x_2 = x^4 + b$, $z_2 = x^2$;
(2) for $i = n - 2$ to 0 do
(3) if $k_i = 1$ then
(4) $(x_1, z_1) \leftarrow Madd\,(x_1, z_1, x_2, z_2, x)$;
$(x_2, y_2) \leftarrow Mdouble\,(x_2, z_2, b)$;
(5) else
(6) $(x_2, y_2) \leftarrow Madd\,(x_2, z_2, x_1, z_1, x)$;
$(x_1, y_1) \leftarrow Mdouble\,(x_1, z_1, b)$;
(7) end if
(8) end for
(9) $Q \leftarrow Mxy(x_1, z_1, x_2, z_2, x, y)$;
(10) return Q;

Algorithm 3: Montgomery scalar multiplication algorithm.

$Mxy(x_1, z_1, x_2, z_2, x, y)$
$\{x_k \leftarrow x_1/z_1;$
$yk \leftarrow [x^2 + y + (x + x_1/z_1)(x + x_2/z_2)](x + x_k)/x = y;$
return (x_k, y_k)
$\}.$

In these functions, (x, y) is the coordinate of the original point P, which is fixed during the calculation of kP; (x_k, y_k) is the coordinate of $kP \cdot k$ is represented on an m bits register. The three basic functions in turn rely on finite field operations such as addition, multiplication, and inversion.

The inversion in GF(2^{193}), required at the final stage, could be realized in one of the two known methods, either via the extended Euclidean algorithm, or by Fermat's theorem which states that knowing after proof that $A^{2^m} - 1 = 1$ leads to consider that $A^{-1} = A^{2^{m-2}}$ is also factual.

Thus, in order to compute the inverse of one element in GF(2^{193}), one needs to take the power of this element (2^{193}-2) times. By using the Itoh-Tsujii algorithm based on the add and multiply Method leads to realize the inverse as presented in Table 3 [13].

C. ECC Components. The ECC processor shown in Figure 1 consists of eight main components. Eight components are host interface (HI), data memory, register file, instruction memory, control-1, control-2, AU-1, and AU-2. The HI communicates with host processor. Processor transmits all parameters for kp to HI with start signal and receives "kp" results and end signal. In the proposed work, we have used Intel 32-bit processor as host processor. Therefore, HI is 32-bit interface. The HI transmits all parameters for computing elliptic curve point doubling and point addition to register file and receives "kp" results from data memory. The data memory consists of 8×163-bit dual port block RAM and the instruction memory contains 13 microcode sequences of 11-bits word, respectively. Thus, these two block memories are to compute coordinate conversion. For high-performance implementation of point doubling and addition, we add

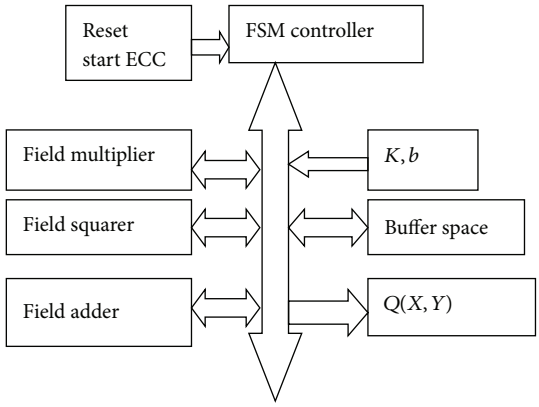

Figure 1: ECC components block diagram.

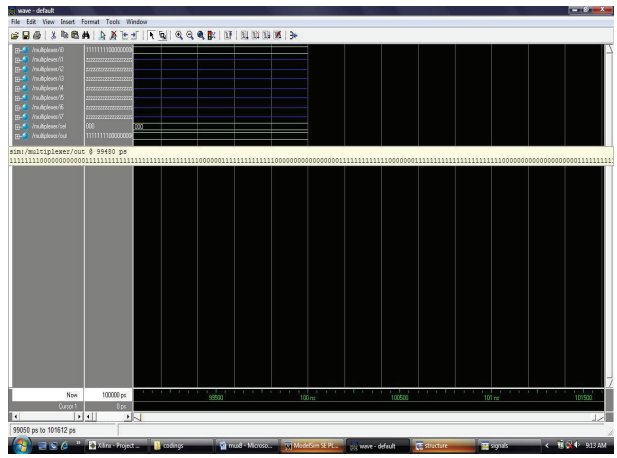

Figure 2: Simulation waveform of multiplexer.

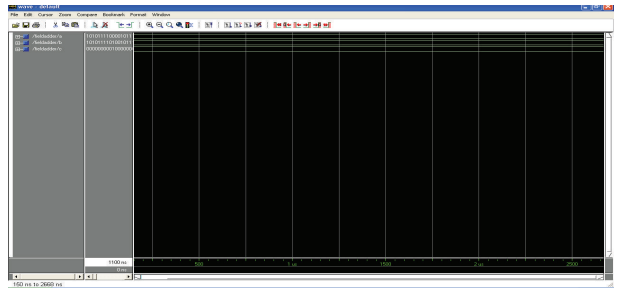

Figure 3: Simulation waveform of field adder.

7×163-bit register file, which receives data from HI and transmits temporary computation results (X_1, X_2, Z_1, Z_2) to data memory. The AU-1 is used for point doubling and addition and controlled by control-1. The AU-2 performs the coordinate conversion in algorithm. The control-2 receives operation code from instruction memory and generates control signals for AU-2, HI, and Data memory. "L" gives the number of required clock cycles to perform elliptic curve point multiplication over GF(2^{163}), where we assumed the digit size $\omega = 55$; that is, $L = 3$.

The 193 bits ECC components have been developed using the VHDL language. The different components forming the design are as follows: A 193 bits adder which is a simple 193 "2 bits" Xors. A 193 bits modulo which is an xor-array evaluated through a Matlab script as an input-output matrix, through polynomial reduction using the National Institute of Standards and Technology (NIST) proposed polynomial $P(x) = x^{193} + x^{15} + 1$. A 193 bits squarer that has also been generated from a Matlab script. A 193 bits modified version of the Karatsuba-Ofman multiplier circuit that is a based on splitting the operands into 3 identical operands (High (H), Middle (M), and Low (L) bits). A Galois inverter circuit requiring 21 power squaring 9 field multiplications.

The 256 bits ECC components have developed a system describes the architecture of the most crucial component in ECDSA, namely that the one that implements elliptic curve operations and modular operations over a finite field GF(P). For this purpose, a flexible, yet compact, elliptic curve processor is developed for applications where speed is of minor importance. The processor is optimized for FPGA families that were introduced to the market from the year 2003 on. But that are still used in many new products. These FPGAs also contain some Hard-IPs (HIPs). The presented processor uses Block RAM and multiplier HIPs, which are available on the majority of FPGA devices. The proposed system is the design of hardware that executes the ECC algorithm that reposes on the ability of making the modular operation over the GF(2^{256}). The research was based on using the efficient Montgomery ladder algorithm, ECDSA algorithm for EC point multiplication. In this system achieved compact architecture.

In Figure 2 the efficient implementation of finite field arithmetic in elliptic curve system. The implementation uses dedicated multiplexer on the FPGA.

In Figure 3 the elliptic curve system achieved high throughput rates. The implementation uses dedicated field adder on the FPGA.

In Figure 4 design of an efficient Field multiplexer over GF(2^{163}) using Gaussian Normal Basis.

In Figure 5 the elliptic curve system achieved high performance. The implementation uses dedicated field squarer on the FPGA.

4. Results and Performance Comparisons

We present the respective estimated number of cycles, required for each part of the algorithm at each stage of FSM controller (Table 1).

Working with 193 bits and 2^{193} order numbers or more is not a direct way and checking the results is very bulky. In this matter, different Matlab scripts with similar input/output behavior to the VHDL programming have been written, in order to compare the execution steps, as well the final results; timing is not taken into consideration in this specific stage.

Synthesis Result of Multiplexer

Speed Grade: −6

Minimum period: no path found

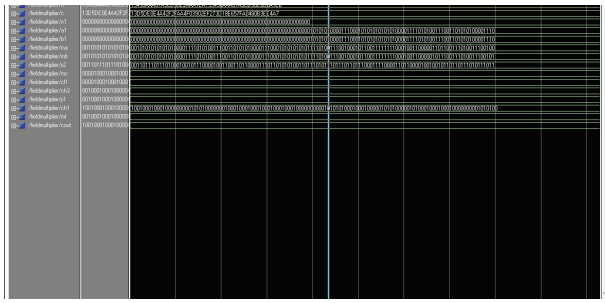

Figure 4: Simulation waveform of field multiplexer.

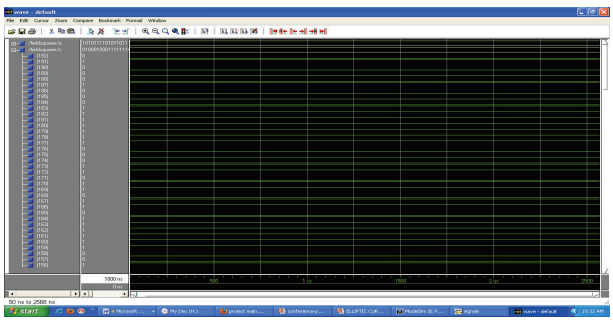

Figure 5: Simulation waveform of field squarer.

Table 1: Field operations required for the ECC operation.

Occurrences in one Field operations	#cycles	#cycle of the FSM
Field multiplication (193 bits)	1	24
Field squaring $(A^2)^1$	1	15
Field squaring $(A^2)^6$	1	7
Field squaring $(A^2)^{15}$	1	8
Field addition	1	11
Field reduction (modulo)	1	24

Minimum input arrival time before clock: 11.217 ns

Maximum output required time after clock: 6.514 ns

Maximum combinational path delay: no path found

Logic utilization:

Number of slices: 466 out of 6144 7%

Number of 4 input LUTs: 932 out of 12288 7%

Number of bonded IOBs: 210 out of 240 87.5%

Number used as Route-thru: 254.

Figure 6 and Table 5 show the output results of the ECC scalar multiplication for a "193 bits" arbitrary value of k.

The frequency of encryption operation is 1930 MHZ and speed of operation also increases (Figure 7). It can be used in any application where security is needed but lacks the power, storage, and computational power that is necessary for our current cryptosystems.

The frequency of decryption operation is 1930 MHZ and the speed of operation also increases. It can be used in any application where security is needed but lacks the power, storage, and computational power that is necessary for our current cryptosystems (Figure 8).

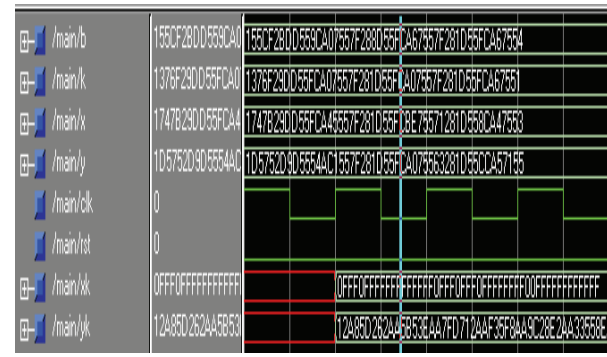

Figure 6: Final result of the scalar multiplication $K \square P$.

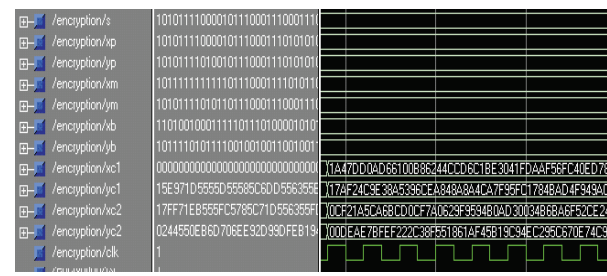

Figure 7: Simulation result of encryption operation.

The processor is optimized for compactness on an FPGA. The implementation uses dedicated modular addition on the FPGA (Figure 9).

In Figure 10, the processor is optimized for compactness on an FPGA. The implementation uses dedicated modular subtraction on the FPGA.

In Figure 11, the processor is optimized for compactness on an FPGA. The implementation uses dedicated multiplier on the FPGA.

In Figure 12, the research was based on using the efficient Montgomery ladder algorithm, ECDSA Algorithm for EC point multiplication. In this system architecture achieved less area.

5. Discussion

The main contribution of the present research concerned three major points: an optimal finite state machine (FSM) controlling the whole components, minimizing empty cycles; optimization of the inversion process, by reducing the number of different squaring from 192 to 21, leading to an inversion; separation of the data path routing from the control part, in order to modify only the multiplier, the squarer, the adder and the modulo components.

The results we have obtained are very encouraging and will impact our decision on the embedding of larger encryption schemes, mainly the extension to the NIST proposed curves (233, 283, 409, and 571), taking into account the use

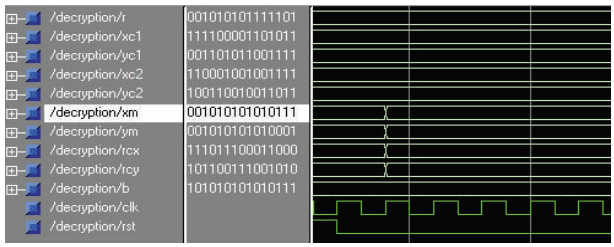

Figure 8: Simulation result of decryption operation.

Table 2: Performance comparisons of ECC (GF(2^{193})) with previous design.

Frequency design [MHz] (Max)	Performance [μs]
Chelton and Benaissa [10] 153.900	19.5500
Smyth et al. [13] 166.000	3720.0000
Sozzani et al. [12] 416.700	30.0000
Satoh and Takano [14] 510.200	190.0000
Sakiyama et al. [3] 555.600	12.0000
Mohamed Abdelkader 561.136	6.1799
This work (Fastest) 1930	1.625

Table 3: Estimation of the FSM stages and their respective execution number of cycles.

FSM steps	#stages	#execution cycles
Startup	1	1
Affine to Projective	1	1
Initial point Doubling	2	2
Counter increase	1	1
Counter compare	1	1
Montgomery point Addition	7	192
Montgomery point Doubling	7	192
Projective to affine	62	1

*The symbol # stands for "Number of."

Table 4: Performance comparisons.

GF(2^m)	Device/size	F (MHZ)/time
163-bit	XC4VLX200/24,363 Slices	143/10 μs
193-bit	XC4VLX200/6376 Slices	1930/1.625 μs
256-bit	XC4VLX200/2085 Slices	2140/1.035 μs

of two or more multipliers (tuned parallel design), the use of internal memories such as Block RAMs (optimized timing memory accesses), the speedup of the FSM, and using different ECC hardware algorithms; these optimization schemes are constrained to minimize the parallel inputs of the design and reduce routing circuitry that severely decrease efficiency, lower speed, and increase power consumption.

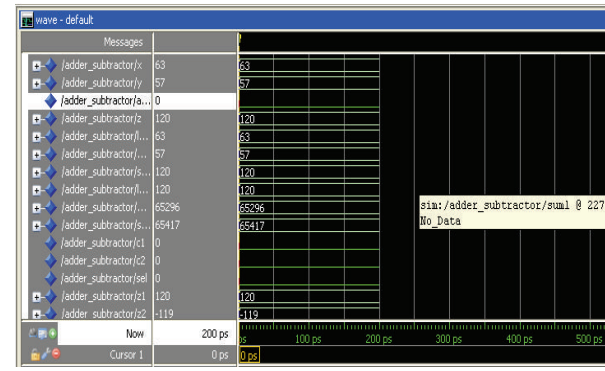

Figure 9: Simulation result of modular addition.

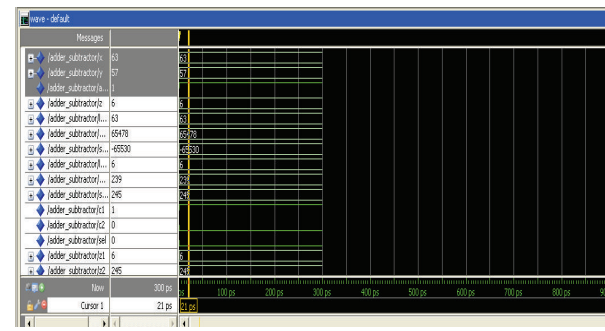

Figure 10: Simulation result of modular subtraction.

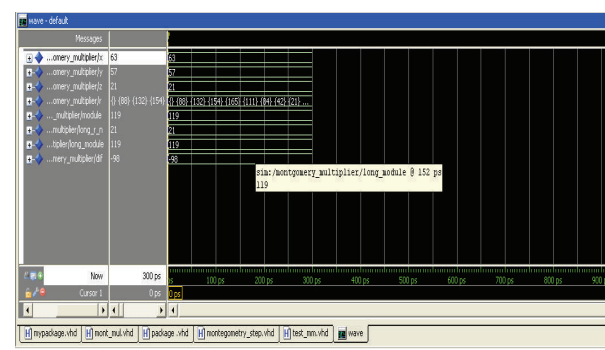

Figure 11: Simulation result of modular multiplication.

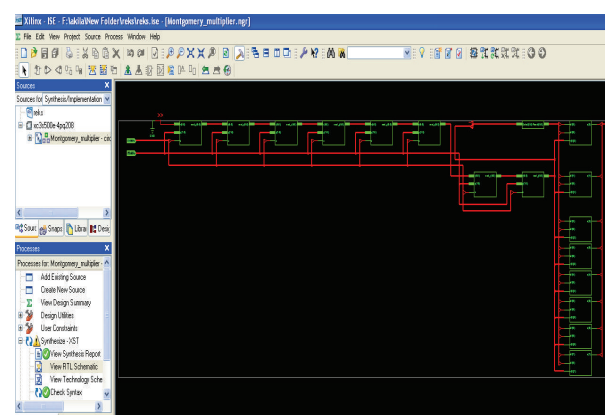

Figure 12: Synthesis waveform of Montgomery multiplexer.

Table 5: The X and Y coordinates of the result $Q = KP$.

k = "1376F29DD55FCA07557F281055FCA07557F281D55FCA67551"
x_k = "0FFF0FFFFFFFFFFFFF0FFF0FFF0FFFFFFFF00FFFFFFFFFFF"
y_k = "12A85D262AA5B53EAA7FD712AAF35F8AA9C28E2AA33558EAA"

6. Results and Conclusion

We have presented the design of a fast version of EC crypto-hardware based on a Finite State Machine. A new ECC processor for GF(2^{163}) is proposed in this paper. The ECC processor consists of eight main components: host interface (HI), data memory, register file, instruction memory, control-1, control-2, AU-1, and AU-2. Secondly, GF(2^{193}) design introduces a better optimization at the level of multiplier and the squaring components, which utilizes the modular inverse circuit. The main characteristics of this design are concerned with the elimination of delays between the different internal components, the minimization of the global clocking resources, and a strategic separation of the data path from the control part.

We have proposed GF(2^{193}) result, indicating that using different optimization at the design of hardware level improves efficiency (Table 2), acceleration of the ECC scalar multiplication (615384 per seconds), and the frequency that is, the frequency of scalar multiplication, encryption, and decryption operations are 1930 MHZ and speed of operation such as key generation, encryption, and decryption are also increased. It can be used in any application where security is needed but lacks the power, storage, and computational power that is necessary for our current cryptosystem.

Thirdly, GF(2^{256}), an elliptic curve processor, which is used to perform the finite field operations in an elliptic curve digital signature unit. The processor is optimized for compactness on an FPGA. The implementation uses dedicated multipliers and block RAMs on the same generation on FPGAs. In this we have achieved the most compact solution. This makes our processor suitable for applications where a small overhead for security is desirable. Security issues will play an important role in the majority of communication and computer networks of the future. As the Internet becomes more and more accessible to the public, security measures have to be strengthened. Elliptic curve cryptosystems allow for shorter operand lengths than other public-key schemes based on the discrete logarithm in finite fields. We have implemented our design using Xilinx XC4VLX200 FPGA device which uses 16,209 slices and has a maximum frequency of 143 MHz. This design is roughly 4.8 times faster than two times increased hardware complexity compared with the existing methods.

References

[1] M. A. Bencherif, H. Bessalah, and A. Guessoum, "Reconfigurable elliptic curve crypto-hardware over the galois field GF(2^{163})," *American Journal of Applied Sciences*, vol. 6, no. 8, pp. 1596–1603, 2009.

[2] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGA: state of the art implementations and attacks," *ACM Transactions in Embedded Computing Systems*, vol. 3, pp. 53–59, 2004.

[3] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "High-performance public-key cryptoprocessor for wireless mobile applications," *Mobile Networks and Applications*, vol. 12, no. 4, pp. 245–258, 2007.

[4] C. Grabbe, M. Bednara, J. Teich, J. von zur Gathen, and J. Shokrollahi, "FPGA designs of parallel high performance GF(2^{233}) multipliers," in *Proceedings of the 2003 IEEE International Symposium on Circuits and Systems*, pp. 268–271, May 2003.

[5] S. Sutikno and A. Surya, "Architecture of $F_2(2N)$ multiplier for elliptic curves cryptosystem," in *Proceedings of the IEEE 2000 Internaitonal Symposium on Circuits and Systems*, pp. 279–282, May 2000.

[6] G. Quan, J. P. Davis, S. Devarkal, and D. A. Buell, "High-level synthesis for large bit-width multipliers on FPGAs: a case study," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis*, pp. 213–218, September 2005.

[7] F. Rodriguez-Henriquez, N. A. Saqib, A. Diaz-perez, and C. K. Koc, *Cryptographic Algorithms on Reconfigurable Hardware*, Springer, Berlin, Germany, 2006.

[8] Y. Dan, X. Zou, Z. Liu, Y. Han, and L. Yi, "High-performance hardware architecture of elliptic curve cryptography processor over GF(2^{163})," *Journal of Zhejiang University*, vol. 10, no. 2, pp. 301–310, 2009.

[9] W. N. Chelton and M. Benaissa, "A scalable GF(2^m) arithmetic unit for application in an ECC processor," in *Proceeding of the IEEE Workshop on Signal Processing Systems*, pp. 355–360, October 2004.

[10] W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on FPGA," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, no. 2, pp. 198–205, 2008.

[11] Y. F. Chung, K. H. Huang, F. Lai, and T. S. Chen, "ID-based digital signature scheme on the elliptic curve cryptosystem," *Computer Standards and Interfaces*, vol. 29, no. 6, pp. 601–604, 2007.

[12] F. Sozzani, G. Bertoni, S. Turcato, and L. Breveglieri, "A parallelized design for an elliptic curve cryptosystem coprocessor," in *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC '05)*, pp. 626–630, October 2005.

[13] N. Smyth, M. McLoone, and J. V. McCanny, "An adaptable and scalable asymmetric Cryptographic processor," in *Proceeding of the IEEE 17th International Conference on Application-specific Systems, Architectures and Processors (ASAP '06)*, pp. 341–346, September 2006.

[14] A. Satoh and K. Takano, "A scalable dualfield elliptic curve cryptographic processor," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 449–460, 2003.