electronics

MDPI

Article

FPGA Implementation of High-Efficiency ECC Point Multiplication Circuit

Xia Zhao [1], Bing Li [1,2,*], Lin Zhang [1], Yazhou Wang [1], Yan Zhang [2] and Rui Chen [3]

[1] School of Integrated Circuits, Southeast University, Nanjing 210096, China; eunicezhao@seu.edu.cn (X.Z.); 230169728@seu.edu.cn (L.Z.); 230179339@seu.edu.cn (Y.W.)
[2] School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China; yanzhang930807@seu.edu.cn
[3] School of Computer and Software Engineering, Nanjing Vocational University of Industry Technology, Nanjing 210023, China; chenrui@niit.edu.cn
* Correspondence: bernie_seu@seu.edu.cn; Tel.: +86-1536-504-5432

Abstract: The authentication of Internet of Things (IoT) devices based on the Physical Unclonable Function (PUF) is widely adopted in the information security domain. However, the leakage of PUF responses in an authentication system reduces its privacy and security. To improve its security, we can utilize the Elliptic Curve Cryptography (ECC) algorithm with different key lengths to encrypt the PUF response arbitrarily. Point multiplication is the most time-consuming operation in ECC because of its complex calculation process, which seriously affects the efficiency of the PUF response encryption. In order to solve this problem, a point multiplier based on binary field with reconfigurable key lengths of 233, 283, 409 and 571 is designed in this paper. In our method, by reusing the underlying computing units, the resources needed for point multiplication are effectively reduced. What it is more innovative is that double point multiplication operations with a key length of less than 283 bits can be performed simultaneously in the elaborate designed point multiplication circuit, which can effectively speed up the encryption process of ECC. The circuit is implemented on Xilinx Virtex-6 FPGA. The experiment results show the single point multiplication times of 233, 283, 409 and 571 key lengths are 19.33, 22.36, 41.36 and 56.5 µs, respectively, under the clock frequency of 135 MHz. In addition, it only needs 19.33 µs to perform two-point multiplication operations when the key length is 233 bits at the same time. When the key length is 283 bits, the point multiplication operation can be performed twice in 22.36 µs.

Keywords: PUF; elliptic curve encryption algorithm; binary field; point multiplication; FPGA

Citation: Zhao, X.; Li, B.; Zhang, L.; Wang, Y.; Zhang, Y.; Chen, R. FPGA Implementation of High-Efficiency ECC Point Multiplication Circuit. *Electronics* **2021**, *10*, 1252. https://doi.org/10.3390/electronics10111252

Academic Editor: Luis Gomes

Received: 14 April 2021
Accepted: 21 May 2021
Published: 24 May 2021

1. Introduction

IoT devices based on PUF are widely adopted in people's daily lives, such as intelligent security, smart home and other application scenarios [1,2]. The security of IoT terminal hardware design should be considered adequately; otherwise, an attacker can obtain the key through a side channel attack or other technical means [3]. An embedded PUF can be used as the physical fingerprint for lightweight authentication, which can effectively improve the security of the device [4–7]. However, once the PUF response is leaked, it will lead to serious risk of security for the authentication system [8]. The authentication of traditional IoT devices based on PUF is shown in Figure 1a. In the registration stage, a PUF-based IoT device can generate response *Rp*, which is directly stored in the server without any protection, and in the authentication stage, a PUF-based IoT device can generate response *Rp'*. If *Rp* is tampered with, *Rp* does not match *Rp'* correctly. Then, the query of the PUF-based IoT device cannot access the server in the authentication stage. Therefore, it is necessary to consider the encryption protection of *Rp*, as shown in Figure 1b. First, the *Rp* is encrypted into ciphertext *Rc* by using the ECC module, and *Rc* is stored in the server. Then, the *Rc* is decrypted via the EEC module to get the original PUF response

Rp during authentication stage. This can effectively protect the PUF response and ensure secure authentication of the IoT device.

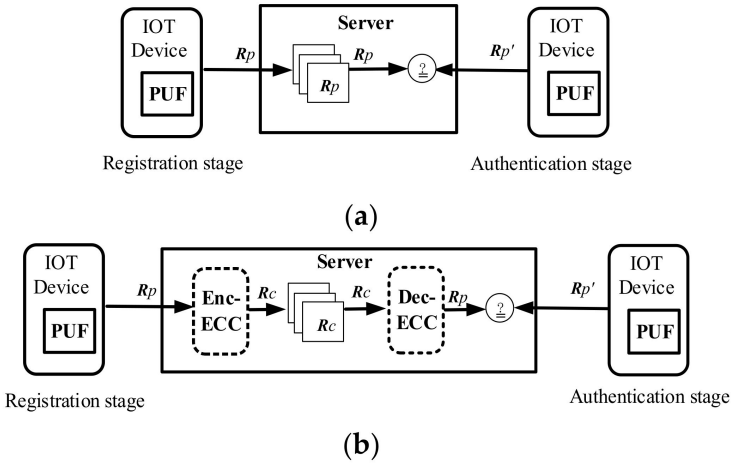

Figure 1. Authentication system for IoT device based on PUF: (**a**) Traditional authentication system; (**b**) authentication system for ECC encryption of PUF response.

Nowadays, ECC has been widely adopted in information encryption systems, as it can provide much higher security strength than other cryptographic algorithms. To protect the PUF responses used in PUF-based authentication protocols, ECC is used to encrypt the PUF response in our lab project. In the application of PUF response encryption, ECC encryption with different key lengths should be selected arbitrarily to increase the computation complexity of the system and improve the security of the system.

ECC encryption can be performed in binary field $F_2{}^m$ or prime field F_p. The elements in $F_2{}^m$ can be quickly calculated in a dedicated processor. In addition, the carry propagation in the prime field needs more area than the polynomial addition in the binary field. Therefore, the binary field $F_2{}^m$ is more suitable for hardware implementation [7]. For this reason, all the calculations in this paper are carried out on binary field $F_2{}^m$. In the implementation of the Elliptic Curve Encryption Scheme (ECES), the calculation of public key, encryption and decryption are all realized by point multiplication (PM). Compared with other operations, PM is the most time-consuming operation in ECC [9]. To accelerate ECC, it is an urgent problem to improve the calculation efficiency of PM. The existing ECC hardware system can only perform the calculation of a single key length. When ECC with different key lengths needs to be operated flexibly, it can only integrate multiple ECC modules with a single key length, so the hardware resource consumption is large, and the flexibility is poor.

In this paper, a reconfigurable ECC system based on FPGA is designed, which can perform the encryption operation by arbitrarily selecting four key lengths to meet the requirements of the PUF response encryption. It can be expanded to arbitrarily select one key from four key lengths for the encryption operation, thus greatly improving the complexity of cracking. The contributions of this paper are as follows.

1. A field operation module compatible with four key lengths is designed to meet the requirements of system compatibility and improved flexibility of the system.
2. The data path of group operation is optimized, the operation steps are reduced, and the speed of the system is improved.
3. The field multiplier designed by combining partial product multiplication with Karatsuba–Offman (KO) algorithm multiplication improves the operation speed of the system, and can execute two m = 233 or 283 at the same time, which further improves the operation speed and reconfiguration efficiency of the system.
4. By reusing the field operation module of the bottom layer, only two adders, two multipliers and two squarers are used to complete the requirements of the bottom layer operation of the system and save the system resources.

This paper is structured as follows. Section 2 lists the related work of hardware circuit research of PM. Section 3 gives a brief overview on the mathematical background of the ECC encryption algorithm and the structure of the original system. Section 4 shows the design of the PM computing hardware structure in detail. Section 5 discusses the experimental results and analysis. Section 6 summarizes the work of this paper.

2. Related Works

For lightweight authentication of an IoT device, we should consider the time consumption of the authentication system. Therefore, the computation complexity of ECC should be reduced as much as possible when the ECC module is embedded to improve the security of the system. As mentioned earlier, current research on ECC hardware implementation mainly focuses on improving the efficiency and speed of PM [9]. Numerous studies have been proposed during the last decade.

Professor Sutter from University of Outenoma studied all the five key length values recommended by NIST. To obtain the best performance, they conducted the point product of different key lengths separately and designed three digital multipliers and a divider on Xilinx Virtex-5 devices. The results showed that the time consumption was 5.5, 17.8, 33.6, 102.6, and 348 µs for the key lengths of 163, 233, 283, 409, and 571 bits in PM [10]. Although the above research realized the PM under different key lengths, the PM consisted of five hardware units, which means that the PM operations of different key lengths could not be achieved on the same hardware unit.

Khan, a doctoral student at the University of Sheffield, put forward two ECC processor architectures based on one or three multipliers, and verified the results on Virtex-5 FPGA. On the one hand, for an ECC processor with one multiplier using 16,090 LUTs, it took 4.91 µs to complete a 163 bit PM. On the other hand, for an ECC processor with three multipliers consisting of 42,192 LUTs, it required 3.99 µs to complete a 163 bit PM. Moreover, Khan also built an ECC processor on the basis of three multipliers on Xilinx Virtex-7 FPGA, and used 141,078 LUTs to achieve a 571 bit PM that took 34.05 µs [11]. The system designed by Khan could complete a PM operation of 163 and 571 key lengths. However, two different systems must be achieved, and the compatibility of multiple key lengths to manage the point multiplication in one system was not solved.

Li Lijuan from Tsinghua University studied a PM accelerator on Koblitz curves. It mainly consisted of a mixed-form, double-digit, τ-adic Non-Adjacent Form (τNAF) converter and a bit-parallel polynomial basis finite field (FF) multiplier accumulator. A fully parallel, three-stage pipeline architecture was used to acquire a PM of 163, 233, 283, 409, and 571 bits. They are 2.5, 4.09, 5.81, 9.5, and 18.51 µs in the Xilinx Virtex-5 platform [12]. According to reference [12], the speed of point multiplication could be improved through optimized scheduling and the parallel scheme of PM. However, the point multiplication operation with different key lengths was still implemented by different modules.

Therefore, it is necessary to design a point multiplier with compatible key lengths of 233, 283, 409 and 571 bits in the same hardware structure. With the popularity of cloud computing and edge computing, the requirements for power consumption, design area and run speed on embedded devices, such as mobile phones, are growing higher. By integrating a hardware accelerator for ECC in IoT devices, not only can the central processing unit (CPU) resources be released, but the energy consumption can be reduced, the security can be improved, and real-time data transmission can be effectively guaranteed as well, which is another motivation of this paper.

3. Background

Point operations in elliptic curves are a series of a combination of addition, multiplication, square, reduction and inversion in $F_2{}^m$. In this section, the implementation of fast arithmetic operations in the binary field is firstly explained, and then description of the related operations in ECC is given.

3.1. Binary Field Arithmetic Operations

Bulleted lists look like the following: the elements $a(z)$ and $b(z)$ of $F_2{}^m$ are represented by binary polynomials of the degree, at most, $m-1$. $a(z)$ and $b(z)$, as written in Formula (1) are as follows:

$$\begin{aligned} a(z) &= a_{m-1}z^{m-1} + \cdots + a_2z^2 + a_1z^1 + a_0 \\ b(z) &= b_{m-1}z^{m-1} + \cdots + b_2z^2 + b_1z^1 + b_0 \end{aligned} \quad (1)$$

3.1.1. Field Addition/Subtraction

The addition on $F_2{}^m$ is the addition of polynomials, with coefficients arithmetic performed modulo 2, and can be expressed by Algorithm 1. Obviously, addition can be achieved by XOR. Through analysis, it is concluded that subtraction is the same as addition [13].

Algorithm 1 Addition in $F_2{}^m$

INPUT: Binary polynomials $a(z)$ and $b(z)$ of degree, at most, $m-1$.
OUTPUT: $c(z) = a(z) + b(z)$.

1. For i from 0 to $m-1$ do
1.1 $c_i \leftarrow a_i \oplus b_i$.
2. Return $c(z)$.

3.1.2. Polynomial Multiplication

Binary field multiplication is divided into two steps: polynomial multiplication and modular reduction. The elements of $F_2{}^m$ are represented by binary polynomials. Multiplication on $F_2{}^m$ starts with polynomial multiplication. Binary polynomial $a(z) = a_{m-1}z^{m-1} + \cdots + a_2z^2 + a_1z^1 + a_0$ is represented by a two-dimensional array of W columns. Polynomial multiplication is obtained with each bit of the word processed [13]. The implementation of the polynomial multiplication is shown in Algorithm 2.

Algorithm 2 Processing columns for polynomial multiplication

INPUT: Binary polynomials $a(z)$ and $b(z)$ of degree, at most, $m-1$.
OUTPUT: $c(z) = a(z) \cdot b(z)$.

1. $C \leftarrow 0$.
2. For k from 0 to $W-1$ do
2.1 For j from 0 to t-1 do
If the kth bit of $A[j]$ is 1 then add B to $C\{j\}$.
2.2 If $k \neq (W-1)$ then $B \leftarrow B \cdot z$.
3. Return(C).

3.1.3. Modular Square

Multiplication is equivalent to the square operation when $a(z) = b(z)$. For the items whose square result reaches m value, the reduction polynomial recommended by NIST is used for modular subtraction. Through the modular reduction of each term of the square result, the result has a certain rule. In the same case, the reduction in the adjacent two modules is the translation position relationship. As a whole, non-zero terms are diagonally related. According to these rules, a simple algorithm for reducing the modulus after the square is obtained. It can be observed that the modular reduction between two adjacent terms is translational. In addition, non-zero terms are diagonal if observed from top to bottom. For these rules, a simple algorithm of modular reduction after the square (Algorithm 3) is worked out. For different key length values, their reduced polynomials are different but their methods are the same, so they must be designed separately.

Algorithm 3 Fast modular square $f(z) = z^{233} + z^{74} + 1$ (when $m = 233$)

INPUT: A binary polynomial $a(z)$ of degree, at most, 232.
OUTPUT: $c(z) = a(z)^2 \bmod f(z)$.

1. For i from 1 to 231 do
1.1 $c[i] \leftarrow a[117 + (i-1)/2]$.
1.2 $i \leftarrow i + 2$.
2. For i from 0 to 232 do
2.1 $c[i] \leftarrow a[i/2]$.
2.2 $i \leftarrow i + 2$.
3. For i from 0 to 72 do
3.1 $c[i] \leftarrow c[i] \oplus a[196+i/2]$.
3.2 $i \leftarrow i + 2$.
4. For i from 74 to 146 do
4.1 $c[i] \leftarrow c[i] \oplus a[196 + (i-74)/2]$.
4.2 $i \leftarrow i + 2$
5. For i from 75 to 231 do
5.1 $c[i] \leftarrow c[i] \oplus a[117 + (i-75)/2]$.
5.2 $i \leftarrow i + 2$
6. Return $c(z)$.

3.1.4. Modular Reduction

The object of modular reduction studied in this paper is the polynomial multiplication result $c(z)$. The maximum degree of $c(z)$ is $2^m - 2$. NIST recommends reduced polynomials for fast reduction. At same time, these polynomials are fixed. Hence, the reduction algorithm is given by the polynomial reduction on $F_2{}^{571}$ [14] as demonstrated in Algorithm 4.

Algorithm 4 Fast reduction modulo $f(z) = z^{571} + z^{10} + z^5 + z^2 + 1$ (with $W = 32$)

INPUT: A binary polynomial $c(z)$ of degree, at most, 1140.
OUTPUT: $c(z) \bmod f(z)$.

1. For i from 35 down to 18 do {Reduce C[i]z32i modulo f (z)}
1.1 T$\leftarrow$ C[i].
1.2 C[i$-$18]$\leftarrow$C[i-18]$\oplus$(T << 5)$\oplus$(T<<7)$\oplus$(T << 10)$\oplus$(T << 15).
1.3 C[i$-$17]$\leftarrow$C[i-17]$\oplus$(T >> 27)$\oplus$(T>>25)$\oplus$(T >> 22)$\oplus$(T >> 17).
2. T$\leftarrow$ C[17] >> 27. {Extract bits 27–31 of C[17]}
3. C[0]$\leftarrow$C[0]$\oplus$T$\oplus$(T << 2)$\oplus$(T << 5)$\oplus$(T << 10).
4. C[17]$\leftarrow$C[17] & 0$\times$7FFFFF. {Clear the reduced bits of C[17]}
5. Return(C[17], C[16], . . . , C[1], C[0]).

3.1.5. Modular Inversion

The inverse element of the field element $a(z)$ on the finite field is $a(z)^{-1} \bmod f(z)$, which can be directly expressed as $a(z)^{-1}$ and satisfies $a(z) \cdot a(z)^{-1} \equiv 1 (mod\, f(z))$ on the finite field. Finite field inversion operations can be divided into two categories according to the Euclidean Algorithm, its extended algorithm and Fermat's Little Theorem [15]. Generally, the inverse algorithm of Fermat's Little Theorem is used in hardware systems. Based on Fermat's Little Theorem, the inversion of the binary field can be realized in the improved way as described by Algorithm 5 [16]. The improved method is presented in Algorithm 5. The inversion operation in different code lengths must be implemented separately.

Algorithm 5 Inversion in $F_2{}^m$ using Fermat's Little Theorem (m odd) (Itoh–Tsujii inversion algorithm)

INPUT: $a \in F_2{}^m (a \neq 0)$.
OUTPUT: a^{-1}.

1. Set $A \leftarrow a^2$, $B \leftarrow 1$, $x \leftarrow (m-1)/2$.
2. While $x \neq 0$ do
2.1 $A \leftarrow A \cdot A^{2x}$.
2.2 If x is even then $x \leftarrow x/2$.
Else $B \leftarrow B \cdot A$, $A \leftarrow A^2$, $x \leftarrow (x-1)/2$.
3. Return (B).

3.2. Elliptic Curve Overview

ECC is based on the elliptic curve discrete logarithm problem (ECDLP), whose core operation is built upon finite field arithmetic operations. The elliptic curve is actually a special kind of plane algebraic curve [13]. The research summarized in this paper is done on the K-curve recommended by NIST. Any randomly generated elliptic curve does not affect the ECDLP.

The K-curve equation is displayed in Formula (2) [14]:

$$E: y^2 + xy = x^3 + 1 \tag{2}$$

There is a chord-and-tangent rule for adding two points in $E(K)$ to produce a third point in $E(K)$. The addition rule is best explained geometrically. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two distinct points on an elliptic curve E. Then the sum R, of P and Q is defined as shown below. As depicted in Figure 2a, the line passing through P and Q intersects with the curve and the third point is generated. The symmetric point of this intersection about the x-axis is R. The double R of P is illustrated in Figure 2b. It can be seen that the tangent passes through point P, intersects with the curve and produces the second point. Then, R is the negative of this point on the x-axis. In Figure 2, we can observe that the specific point addition and point doubling operations come from the following two graphs.

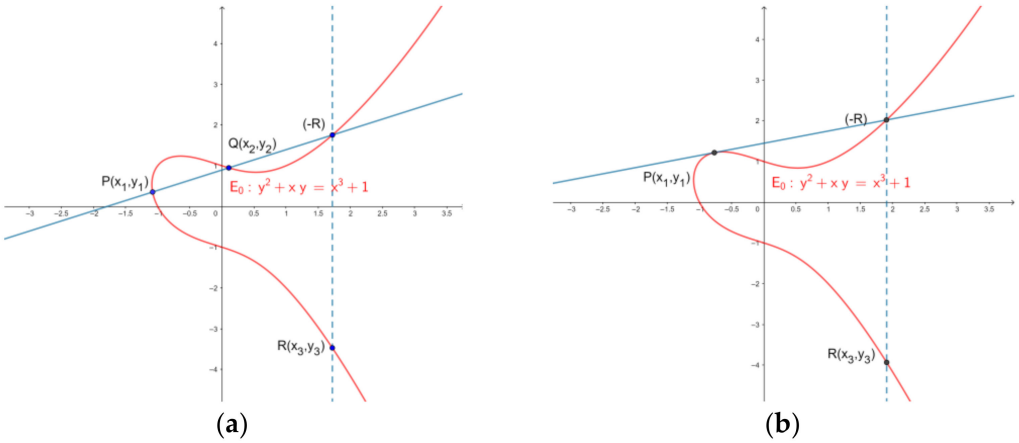

Figure 2. Geometric addition and doubling of elliptic curve point. If there are multiple panels, they should be listed as: (**a**) Addition: $P + Q = R$.; (**b**) doubling: $P + P = R$.

The coordinates of the point are firstly explained. There are many kinds of projection coordinates of points on the K-curve in the binary field, which include standard coordinates, Jacobian coordinates and LD coordinates. The PM of the system is achieved by the improved Montgomery method, where the affine coordinate points are mapped to the standard projection coordinates according to the research done by Lopez and Dahab [17]. The standard projection point (X:Y:Z), $Z \neq 0$ corresponds to the affine point (X/Z, Y/Z). With the addition and doubling of points, projective coordinates are obtained, which must be retransformed to affine coordinates subsequently.

3.2.1. Point Addition

The addition of points is performed in standard projective coordinates. Take the three points P, P_1 and P_2 on the curve so that $P_2 = P_1 + P$ [13]. Replace the x coordinate of the point P_1 with X/Z, where $i \in \{1,2\}$. The x coordinate of $P_1 + P_2$ is replaced by a standard projected coordinate. See Algorithm 6 for implementation.

Algorithm 6 Montgomery point addition (Madd) (standard projective coordinates)

INPUT: $P = (x,y)$ in affine coordinates, $P_1 = (X_1{:}Y_1{:}Z_1)$ and $P_2 = (X_2{:}Y_2{:}Z_2)$ in standard projective coordinates on $(E(F_2{}^m){:}y^2 + xy = x^3 + ax^2 + b, P \in E(F_2{}^m))$.
OUTPUT: $P_1 + P_2 = (X_1{:}Y_1{:}Z_1)$ in standard projective coordinates.

1. $T_1 \leftarrow x$
2. $X_1 \leftarrow X_1 \times Z_2$
3. $Z_1 \leftarrow Z_1 \times X_2$
4. $T_2 \leftarrow X_1 \times Z_1$
5. $Z_1 \leftarrow Z_1 + X_1$
6. $Z_1 \leftarrow Z_1{}^2$
7. $X_1 \leftarrow Z_1 \times T_1$
8. $X_1 \leftarrow X_1 + T_2$

3.2.2. Point Doubling

Similarly, the doubling of points is conducted in standard projective coordinates. Point doubling is a special case of point addition, and its operation can be simplified. Take a point P on the curve. Replace the x coordinate of the point P with X/Z. Point doubling is to transform x coordinate of $2P$ into standard projective coordinate [13]. Algorithm 7 describes the implementation of point doubling.

Algorithm 7 Montgomery point doubling (Mdouble) (standard projective coordinates)

INPUT: $P = (X{:}Y{:}Z)$ in standard projective coordinates on
$(E(F_{2^m})\ y^2 + xy = x^3 + ax^2 + b, c^2 = b, P \in E(F_{2^m}))$.
OUTPUT: $2P = (X{:}Y{:}Z)$ in standard projective coordinates.

1. $T_1 \leftarrow c$
2. $X \leftarrow X^2$
3. $Z \leftarrow Z^2$
4. $T_1 \leftarrow Z \times T_1$
5. $Z \leftarrow Z \times X$
6. $T1 \leftarrow T_1{}^2$
7. $X \leftarrow X^2$
8. $X \leftarrow X + T_1$

3.2.3. Coordinate Retransformation

The point addition and point doubling operations are carried out in the standard projection coordinate system, and the results are still in the projection coordinate system. The PM module uses affine coordinates, so it is necessary to transform the result from projection coordinates to affine coordinates.

The algorithm for coordinate retransformation is accomplished on the basis of the direct correspondence between standard projective coordinates and affine coordinates. See Algorithm 8 for the implementation of coordinate retransformation.

Algorithm 8 Montgomery point retransformation (Mxy) (standard projective coordinates)

INPUT: $P = (x,y)$ in affine coordinates, $P_1 = (X_1{:}Y_1{:}Z_1)$ and $P_2 = (X_2{:}Y_2{:}Z_2)$ in standard projective coordinates on $(E(F_2{}^m){:}y^2 + xy = x^3 + ax^2 + b, P \in E(F_2{}^m))$.
OUTPUT: $P_1 = (x_k,y_k) = (X_2,Z_2)$ in affine coordinates.

1. If $Z_1 = 0$ then output (0,0) and stop.
2. If $Z_2 = 0$ then output $(x, x+y)$ and stop.
3. $T_1 \leftarrow x$
4. $T_2 \leftarrow y$
5. $T_3 \leftarrow Z_1 \times Z_2$
6. $Z_1 \leftarrow Z_1 \times T_1$
7. $Z_1 \leftarrow Z_1 + X_1$
8. $Z_2 \leftarrow Z_2 \times T_1$
9. $X_1 \leftarrow Z_2 \times X_1$
10. $Z_2 \leftarrow Z_2 + X_2$
11. $Z_2 \leftarrow Z_2 \times Z_1$
12. $T_4 \leftarrow T_1{}^2$
13. $T_4 \leftarrow T_4 + T_2$
14. $T_4 \leftarrow T_4 \times T_3$
15. $T_4 \leftarrow T_4 + Z_2$
16. $T_3 \leftarrow T_3 \times T_1$
17. $T_3 \leftarrow inverse(T_3)$
18. $T_4 \leftarrow T_3 \times T_4$
19. $X_2 \leftarrow X_1 \times T_3$
20. $Z_2 \leftarrow X_2 + T_1$
21. $Z_2 \leftarrow Z_2 \times T_4$
22. $Z_2 \leftarrow Z_2 + T_2$

3.3. Point Multiplication

The core operation of the ECC algorithm is the multiplication of points, where $Q = kP$, k is the private key, Q is the public key, and P is the base point on the elliptic curve. The hierarchical structure of the point multiplication operation is shown in Figure 3. The point multiplication operation is converted into point addition, point doubling and coordinate retransformation operations. Point doubling operations are achieved through field addition, polynomial multiplication, modular reduction and modular square operations. It can be seen from the analysis in Section 3.2 that there is no need for inversion when performing point addition and point doubling under projected coordinates, which can greatly improve the efficiency of point multiplication. In order to carry out the point multiplication operation, the projective coordinates of points must be retransformed to affine coordinates. Modular inversion is demanded for coordinate retransformation [13]. The point multiplication under standard projective coordinates is to be gained by the following Algorithm 9.

Algorithm 9 Montgomery point multiplication (standard projective coordinates)

INPUT: $k = (k_{m-1}, \ldots, k_2,k_1,k_0)_2, k_{m-1} = 1, P = (x, y) \in E(\mathrm{F}_2{}^m)$.
OUTPUT: kP.

1. $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$. {Compute(P,2P)}
2. For i from $m-2$ downto 0 do
2.1 If $k_i = 1$ then
Madd(X_1, Z_1, X_2, Z_2), Mdouble(X_2, Z_2).
2.2 Else
Madd(X_2, Z_2, X_1, Z_1), Mdouble(X_1, Z_1).
3. Return(Q = Mxy(X_1, Z_1, X_2, Z_2)).

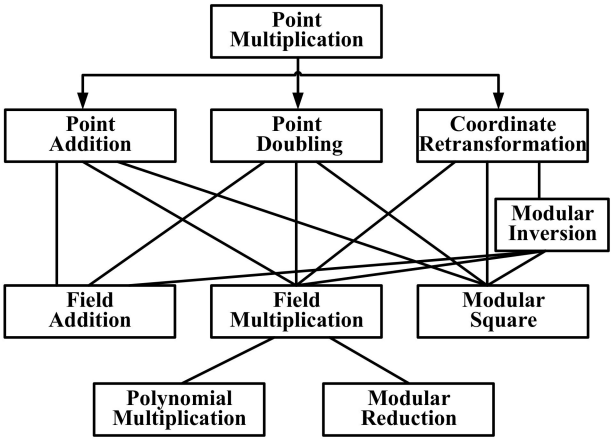

Figure 3. Hierarchical structure of the point multiplication operation.

4. Hardware Design Architecture

It can be concluded that PM is the core device of the ECC encryption system from the above section, and the implementation of PM is described below.

4.1. PM Module Overview

Figure 4 illustrates a structural diagram of the PM system with different key lengths. This arithmetic logic unit is called the PM module.

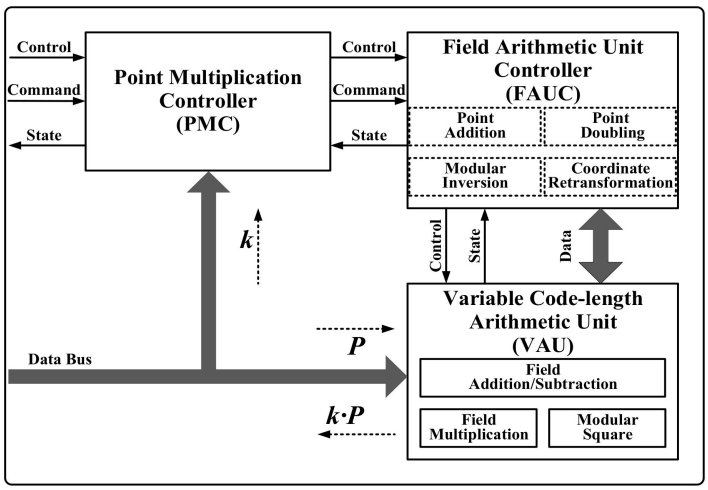

Figure 4. Structural diagram of point multiplication arithmetic unit with variable key lengths.

The PM module is divided into three parts: variable code length arithmetic unit (VAU), field arithmetic unit controller (FAUC) and PM controller (PMC). VAU, the basis of arithmetic operation, is the bottom operation module in the system. The module includes three field operations: field addition/subtraction, field multiplication and modular square. These field operations are compatible with diverse key lengths in the point multiplication system. FAUC is used to send control signals, which requires the submodules of VAU. FAUC is composed of four submodules: point addition, point doubling, inversion and coordinate retransformation. PMC, the top-level controller in the PM module, sends control signals and calls the FAUC submodule to complete the multiplication of points.

4.2. VAU

4.2.1. Field Addition/Subtraction Module

For $F_2{}^m$ of polynomial basis, addition is performed, bitwise. There is no carry propagation, and field addition can simply be realized through the XOR circuit. Similarly, it

is concluded that subtraction is the same as addition. The core of the hardware design of the field addition module is a simple XOR gate array, and the circuit diagram is given in Figure 5.

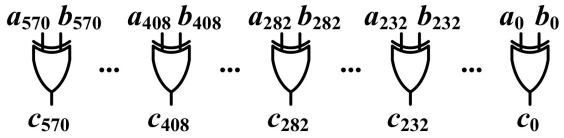

Figure 5. Circuit diagram of field addition operation.

4.2.2. Modular Square Module

For different key lengths, NIST recommends different polynomial reductions, but the fast modular square methods of Algorithm 3 have no difference at all. Therefore, the modular square operation module with different key lengths should be designed separately. To take the key length m = 233 as an example, it can be seen that each output bit is obtained by an XOR of, at most, two input bits. The actual circuit structure is shown in Figure 6.

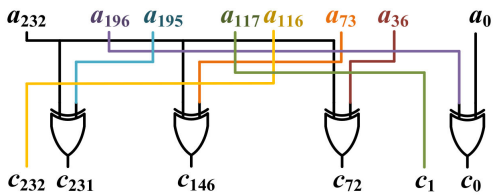

Figure 6. Circuit diagram of modular square operation (m = 233).

In this system, owing to the variable key length algorithm, four modular square operation modules need to be designed so as to perform the modular square operations under different key lengths, and the modular square operation module is created separately because of its simple structure and limited hardware resources. In this way, the multiplication of points is sped up. A set of square circuits are designed to construct the square module in this system, and the modular square operations with different key lengths are executed respectively. The modular square module is displayed in Figure 7.

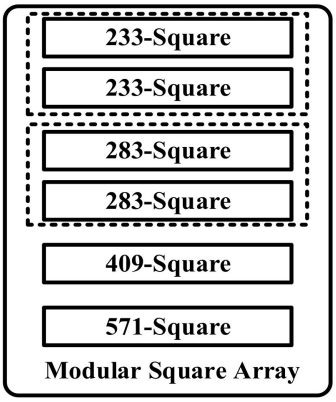

Figure 7. Modular square module.

4.2.3. Field Multiplication Module

Field multiplication operation plays the most important role in a single point multiplication operation. There are two ways to achieve a field multiplier: parallel and serial. The traditional field multiplier is a two-step operation that directly performs field multiplication, so it is also called the two-step traditional field multiplier. On the one hand, it

is a full parallel structure and occupies a larger area. On the other hand, the shift serial field multiplier can segment the multiplier into W bits according to the word width, which accelerates this operation but increases the circuit complexity. Therefore, if the key length is longer, these two methods are not suitable for the hardware system.

According to the above analysis, two field multipliers with different multiplexing structures are designed. One is based on a digital serial structure and segmented with the partial-product calculated. The multiplier consists of polynomial multiplication and modular reduction. The other is implemented by following the KO algorithm, where field multiplication is performed step by step. The polynomial multiplication is firstly performed by referring to the KO algorithm. Then, the modular reduction is conducted. The performance comparison of the two field multipliers is shown in Table 1. The first line of Table 1 is the multiplier circuit structure. The second line is the maximum clock frequency of the circuit. The third column is the FPGA devices used in the system implementation. The second to third lines are the hardware resource consumption. The fourth to ninth lines are the number of clocks required by the multiplier to complete a field multiplication for different code length values.

Table 1. Comparison between digital serial multiplier and KO multiplier @XC6VLX240T [18].

Circuits	Digital Serial Multiplier	KO Multiplier
Freq. (MHz)	141.920	188.253
LUTs	20,114	41,612
Registers	1757	3089
Delay (Clock Cycles) (m = 233)	16	1
Delay (Clock Cycles) (m = 283)	18	1
Delay (Clock Cycles) (m = 409)	26	2
Delay (Clock Cycles) (m = 571)	36	2
Delay (Clock Cycles) (m = 233 & 233)	-	1
Delay (Clock Cycles) (m = 283 & 283)	-	1

The performance test results of two different field multiplexers are compared. The field multiplier based on a digital serial structure takes less resources. The hardware resource of field multiplier based on KO algorithm is doubled. However, it can be calculated from Table 1 that when the key length is 233, the speed of performing a multiplication is increased by 95%. In particular, if we need to process two m = 233 or 283 key length modular multiplications at one time, they can be executed simultaneously in the field multiplier of the KO algorithm, and the speed can be doubled.

Field multiplication is done in two steps: polynomial multiplication and modular reduction. Each function is executed step by step. Hence, the field multiplication module is divided into two submodules: polynomial multiplication and modular reduction.

- Polynomial multiplication submodule.

The KO algorithm is an effective recursive algorithm for polynomial multiplication. The highest degree of elements of $F_2{}^m$ is $m-1$, and its operation process is gradually halved. The implementation of the KO algorithm can be represented by Formulas (3) and (4). Element A is divided into AH and AL. In the same way, B is split into two parts. By analogy, AH, AL, BH and BL are divided into more iterations. To take A as an example, it can be represented by the hierarchy diagram as observed in Figure 8.

$$\begin{aligned} a(z) &= z^{m/2}(z^{m/2\text{-}1}a_{m-1} + \cdots + a_{m/2}) + (z^{m/2\text{-}1}a_{m/2\text{-}1} + \cdots + a_0) = z^{m/2}A_H + A_L \\ b(z) &= z^{m/2}(z^{m/2\text{-}1}b_{m-1} + \cdots + b_{m/2}) + (z^{m/2\text{-}1}b_{m/2\text{-}1} + \cdots + b_0) = z^{m/2}B_H + B_L \end{aligned} \quad (3)$$

$$d(z) = a(z)\cdot b(z) = z^m A_H B_H + z^{m/2}(A_H B_L + A_L B_H) + A_L B_L \quad (4)$$

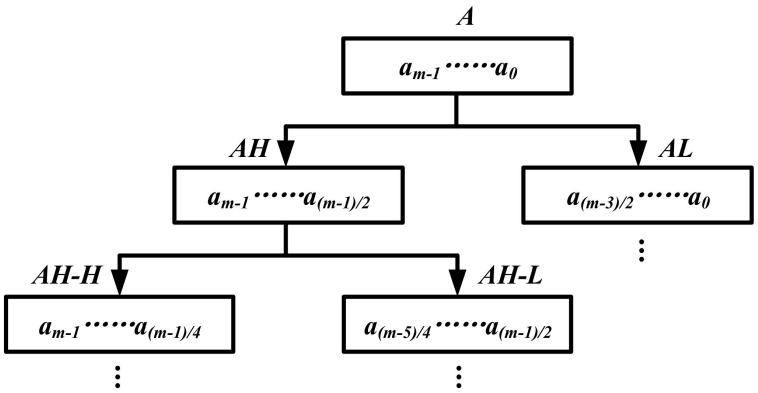

Figure 8. Schematic diagram of element half processing.

The system needs to be compatible with ECC operations, the key lengths of which are 233, 283, 409 and 571 bits respectively. Combined with Formula (4), the key lengths of 409 and 571 bits can be converted into the multiplication operations of four pairs of a 286-bit length on $F_2{}^m$ by using the partial product. The KO multiplier is designed on the basis of the partial product as shown in Figure 9. Take 571 as an example; split the operands by following the processing scheme presented in Figure 7, calculate *AH*·*BH* and *AL*·*BL* in the first step, *AH*·*BL* and *AL*·*BH* in the second step, and the result of 571 is accomplished according to Formula (4).

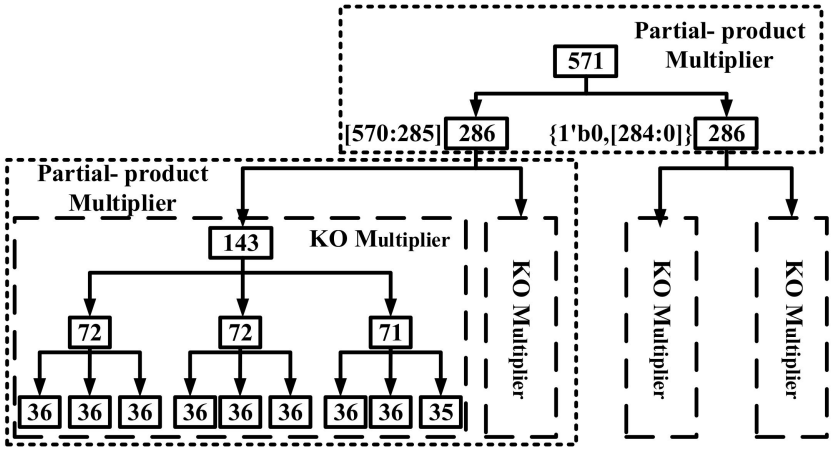

Figure 9. Data half path graph of KO Multiplier based on partial-product.

When the key length is less than 286, only one multiplier is needed for the ECC operation. Hence, two ECC operations with key lengths of less than 286 can be managed simultaneously. As only with one clock, the multiplication of m = 233 or 283 can be completed; therefore, the multiplication of one pair of key groups (m = 233 & 233, m = 283 & 283) is performed at the same time. It takes two clock cycles to execute a multiplication of m = 409 or 571. To sum up, this multiplier is not just energy efficient, but also ensures the maximum reuse of underlying resources. Therefore, it is able to fulfill the requirements of the encryption of multiple keys at the same time.

- Modular reduction submodule.

The corresponding reduction polynomials, advocated by NIST, of the four key length values are fixed and used for fast modular reduction in this design. Taking m = 571 for an example, the results of modular reduction are given in Table 2.

Table 2. The fast modular reduction result ($m = 571, f(z) = z^{571} + z^{10} + z^5 + z^2 + 1$).

The Range of Degree i	Modular Reduction Results
[0, 570]	z^i
[571, 1131]	$(z^{10} + z^5 + z^2 + 1)z^{i-571}$
[1132, 1136]	$(z^{566} + z^{563} + z^{561} + z^{10} + z^5 + z^2 + 1)z^{i-1132}$
[1137, 1139]	$(z^{568} + z^{566} + z^{15} + z^7 + z^2 + 1)z^{i-559}$
[1140]	$z^{569} + z^{18} + z^3 + z^2 + 1$

As seen from Table 2, when the reduction polynomial is a given and fixed value, the modular reduction can be achieved fast by XOR. Provided that the key length is varied, the reduction polynomials and the modular reduction results are different. The modular reduction module should be designed in a specific manner according to the different key lengths.

In order to realize the simultaneous calculation of two keys, the lengths of which are less than 286 in the multiplier, a modular reduction array that covers two 233-module reductions, two 286-module reductions, one 409-module reduction and one 571-module reduction is formed, as shown in Figure 10.

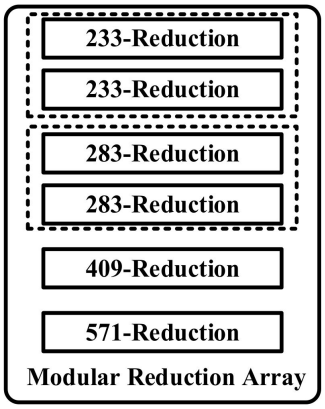

Figure 10. Modular reduction array.

4.3. FAUC

The FAUC module is used to achieve the function of elliptic curve group operations; it determines the computation performance of the ECC system. There are four modules in FAUC. The point addition module sends the control signal to VAU, and starts the bottom addition, multiplication and square arithmetic operations, so as to accomplish the point addition operation. Similarly, the bottom addition, multiplication and square operations are also implemented via the point doubling, modular inverse and coordinate retransformation modules, respectively.

4.3.1. Point Addition Module

This Montgomery point addition method is performed according to Algorithm 6, which is applied in this system. One square, four multiplications and two additions should be done if the point addition operation in Algorithm 6 is to be conducted. The optimal design diagram of the point addition is similar to point doubling, where the arithmetic operators are reused in Figure 11.

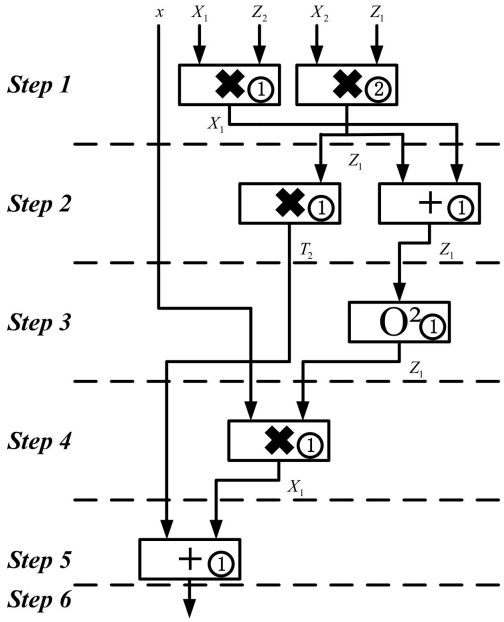

Figure 11. Data path diagram of Madd operation circuit.

As shown in Figure 10, it can be seen that there is one multiplier reused three times in the process of fulfilling the point addition operation. As a result, in the Madd operation, only one squarer, two multipliers and one adder are employed within six steps.

4.3.2. Point Doubling Module

The Montgomery point doubling method is used in the point doubling operation, and affine coordinates are transformed into standard projection coordinates according to Algorithm 7. Four squares, two multiplications, one addition and one intermediate variable are required to perform the point doubling operation. In order to save resources, the operators are reused by utilizing the pipeline way. As a result, at least two squares, two multipliers and one adder are needed to complete the point doubling operation. The data path of the optimized circuit is shown in Figure 12. It can be seen from the data path that the point doubling operation uses five steps where one squarer is reused twice in the actual circuit.

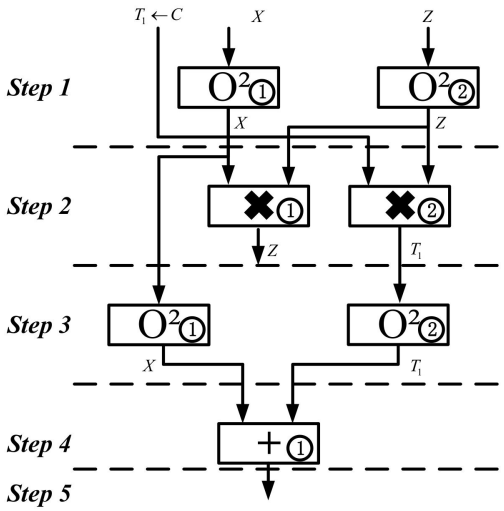

Figure 12. Data path diagram of the Mdouble operation circuit.

4.3.3. Modular Inversion Module

The modular inverse, an important part of point multiplication, is fairly complex in terms of calculation. This system needs to be compatible with ECC encryption of four different key lengths. We can utilize the proposed algorithm of Fermat's Little Theorem (see Algorithm 5) to design this module. Its hardware implementation is performed by reusing the field operation module. Therefore, this operation can effectively reduce the computational complexity of modular inversion. The modular inverse operation steps for four different key length values are given in Table 3.

Table 3. Modular inverse operation steps for four different key lengths.

Key Length (m)	233		283		409		571	
	Product	**Cycles**	**Product**	**Cycles**	**Product**	**Cycles**	**Product**	**Cycles**
Step1	a	1	a	1	a	1	a	1
Step2	a	1	a^{2^2-1}	2	a^{2^2-1}	2	a^{2^2-1}	2
Step3	a^{2^3-1}	3	a^{2^4-1}	4	a^{2^3-1}	3	a^{2^4-1}	4
Step4	a	1	a^{2^8-1}	8	a^{2^6-1}	6	a^{2^8-1}	8
Step5	a^{2^7-1}	7	a	1	$a^{2^{12}-1}$	12	a	1
Step6	$a^{2^{14}-1}$	14	$a^{2^{17}-1}$	17	a	1	$a^{2^{17}-1}$	17
Step7	a	1	a	1	$a^{2^{25}-1}$	25	a	1
Step8	$a^{2^{29}-1}$	29	$a^{2^{35}-1}$	35	a	1	$a^{2^{35}-1}$	35
Step9	$a^{2^{58}-1}$	58	$a^{2^{70}-1}$	70	$a^{2^{51}-1}$	51	a	1
Step10	$a^{2^{116}-1}$	116	a	1	$a^{2^{102}-1}$	102	$a^{2^{71}-1}$	71
Step11	1	1	$a^{2^{141}-1}$	141	$a^{2^{408}-1}$	408	$a^{2^{142}-1}$	142
Step12	-	-	1	1	1	1	a	1
Step13	-	-	-	-	-	-	$a^{2^{285}-1}$	285
Step14	-	-	-	-	-	-	1	1

Table 3 shows that the steps for the modular inversion are numerous but not that complicated. In this design, four modular inverse operation step tables with different key lengths are fixed in the memory, and then a state machine is used to realize the inverse operation of the four code length values.

Figure 13 illustrates the steps of the inversion operations. The cycle number as well as the multiplication term of "MOD_INV_1_1" are all determined by the ECC key length value and the inversion steps in Table 3. "MOD_INV_1_3" offers a comparison of inversion steps to see whether the corresponding inversion step has been completed according to the ECC key length, so as to move to "MOD_INV_2". The inverse result is obtained in "MOD_INV_2". "T4" denotes the result of each inverse step.

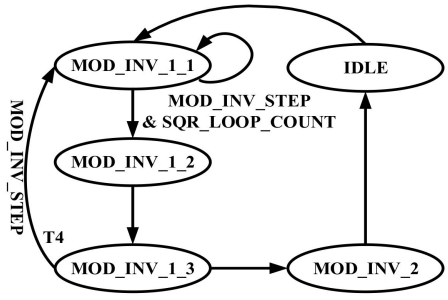

Figure 13. State chart of modular inversion module.

As seen in Table 3, modular inversion is implemented by multipliers and squarers. Therefore, two 233 or 283 point multiplication operations can be accomplished simultaneously by using the above two 286 length partial product multipliers and two squarers in parallel.

4.3.4. Coordinate Retransformation Module

Point addition and point doubling are calculated in standard projective coordinates, but point multiplication needs to be calculated under the affine coordinate values. Therefore, upon the completion of the iteration of the point operation, it is necessary to have the coordinates restored. In this system, coordinates have to be transformed from standard projective ones to affine ones, that is, from (X, Z) to (x, y). As mentioned above, the coordinate retransformation may be achieved by Algorithm 8.

The coordinate retransformation circuit is designed based on Algorithm 8. With the reused arithmetic unit, the data path can be optimized as shown in Figure 14. In this scheme, fully reusing the bottom field operation module, only two adders, two multipliers and one squarer are needed to complete the coordinate inversion.

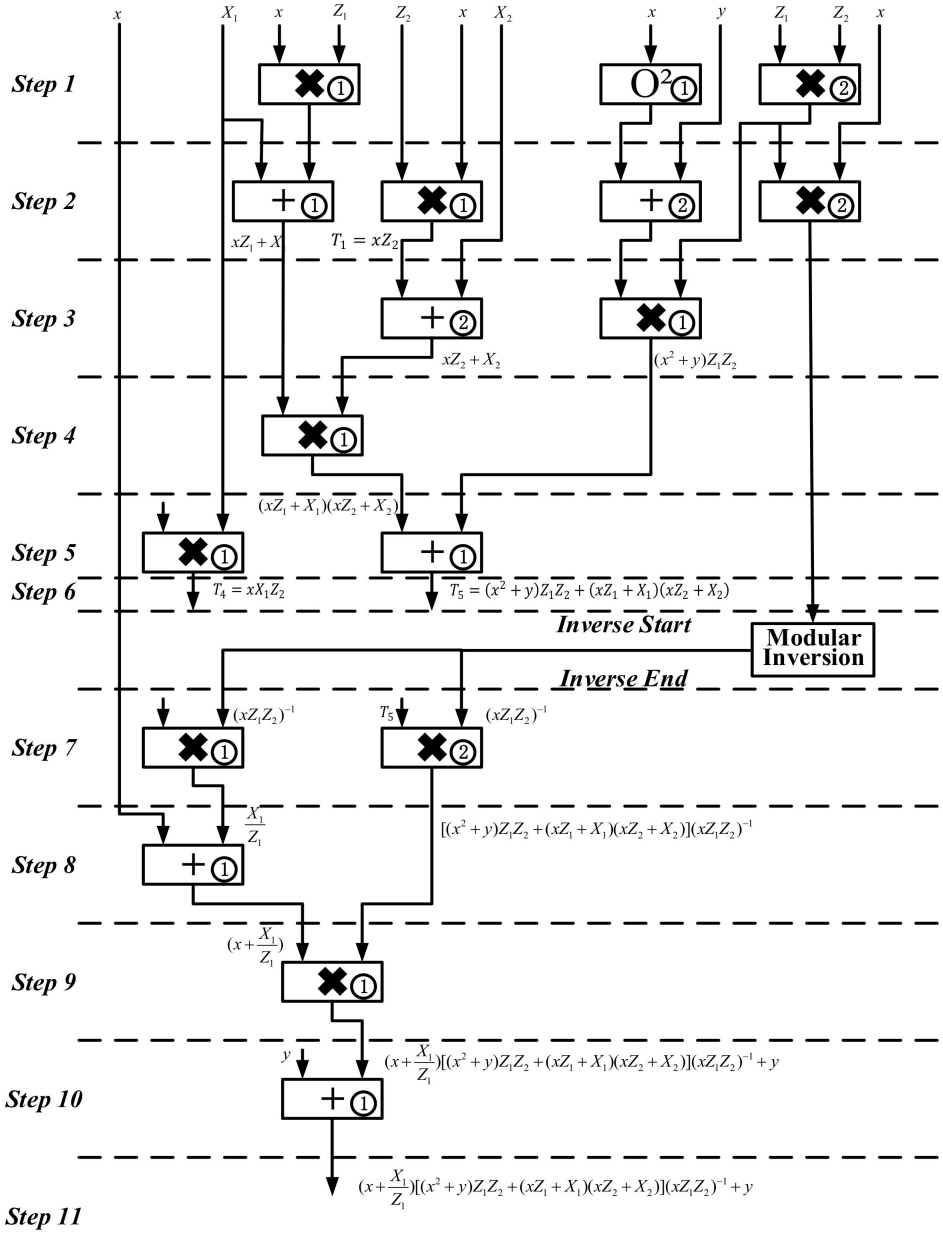

Figure 14. Data path diagram of Mxy operation circuit.

Considering the requirements of point addition, point multiplication and coordinate retransformation, this PM system needs at least two adders, two multipliers and two squarers to complete one point multiplication.

4.4. PMC

The major role of the PMC module is to integrate the operations of the scheduled point adding and doubling modules to work through the point multiplication operations. Based on the PM operations, an operation flow chart of PM is concluded in Figure 15. In terms of the point multiplication process, the specific functions of the PM control module are listed as follows:

1. The affine coordinates of base point P are transformed into standard projective coordinates, and the data initialization assignment of this module is completed.
2. The private key k is scanned bit by bit to determine the value of each bit, and the point addition and point doubling operations are subject to the scanning results.
3. According to the Montgomery method, the point adding and doubling modules are called upon to perform the iterative operation.
4. Upon completion of the point multiplication operation in projective coordinates, coordinate retransformation is called upon to convert the outcome into affine coordinates.

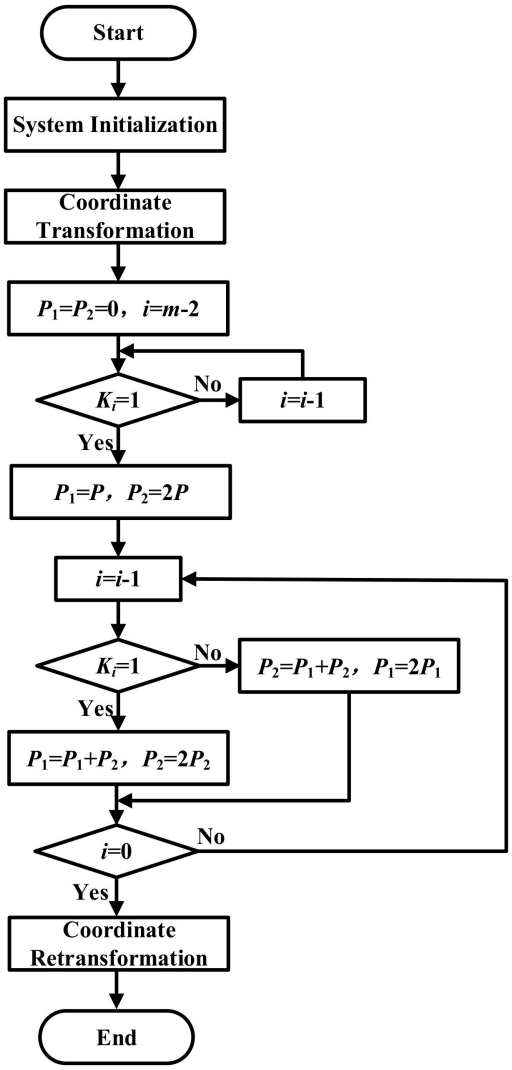

Figure 15. Flow-process diagram of Montgomery point multiplication in projective coordinates.

The system is compatible with point multiplication operations of four different key lengths, so the PM control module must be guaranteed to work in response to the selection signal of the key length value.

The following can be seen from the PM structure diagram (Figure 16): the PM system has two field multipliers, two modular squarers and two field adders. One field multiplication is made up of two 286-KO multipliers and one reduction array; one modular square consists of two 233-SQR, two 283-SQR, one 409-SQR and one 571-SQR; one modulo addition module includes one 571-bit adder.

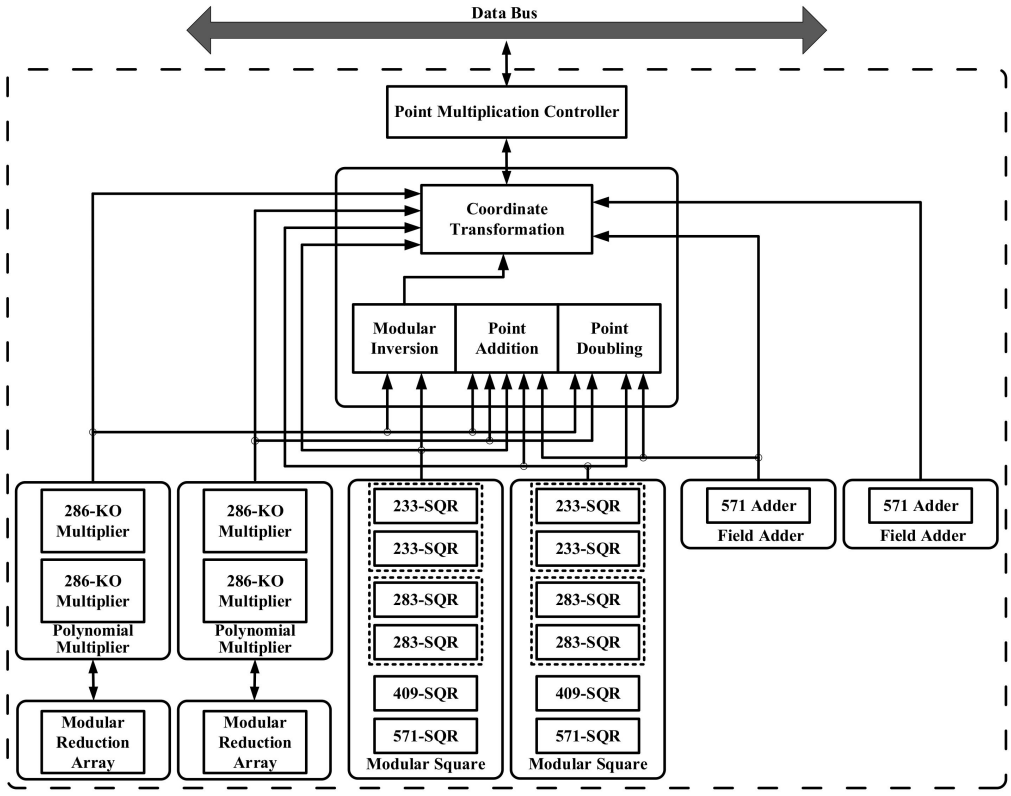

Figure 16. Point multiplication hardware structure diagram.

Four of the same 286 length partial product multipliers and corresponding squarers are utilized in the point multiplication when m = 409 and 571. On the one hand, this significant advantage is that the PMC module is compatible with point multiplication for four different key lengths. On the other hand, it can perform two-point multiplication with 233 or 283 key lengths at the same time.

In this way, the system can simultaneously execute two ECC algorithms with a key length of 233 or 283, or two combined key lengths of 233 and 283, respectively.

5. Experimental Results and Discussion

5.1. Experiment Setup

As shown in Figure 17, we design the PM of an ECC-based encryption algorithm on the FPGA platform. In order to test the PM performance, we establish a test platform by using the PC and ML605 evaluation board for evaluating this performance in Figure 18. The ML605 evaluation board uses a XC6VLX240T FPGA chip, and the I/O interface between the PC and the evaluation board adopts a serial port to achieve serial communication. In the PC terminal, we use serial debugging software to complete the data transmission and reception. In this experiment, we use the counter module of FPGA to obtain the number of clock cycles *Clk.* of point multiplication. Combined with the operating frequency *Freq.* (MHz), we can calculate the time T (μs) for one point multiplication. This formula can be expressed as Formula (5).

$$T = Clk./Freq.\ (\mu s) \tag{5}$$

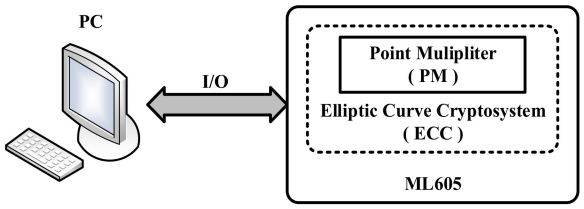

Figure 17. Typical application system of the FPGA PM for ECC.

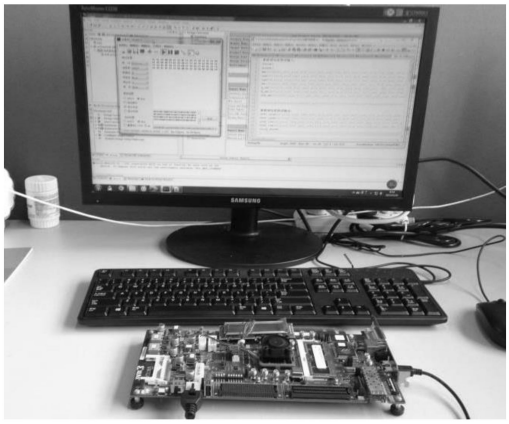

Figure 18. Experimental test platform.

We take a 571-bit key length as an instance to test the module of the point multiplication. First of all, the test data are transmitted into the module of the point multiplication of FPGA though serial communication. Then, the result is calculated via the point multiplication module. Finally, these calculated data are transmitted to serial debugging software of PC by using serial communication. As shown in Figure 19, the input data are the based point Px, Py coordinates and private key K, respectively. The output data are the public key coordinates Qx and Qy. Moreover, the 4 bytes after the flag bits (00FF) are the real clock cycles of the point multiplication. We can observe that this result of the point multiplication is equal to the theoretical result. In addition, the operating frequency is 135 Mhz, and the run time of point multiplication $T = 7628/135 = 56.50$ μs can be computed by Formula (5).

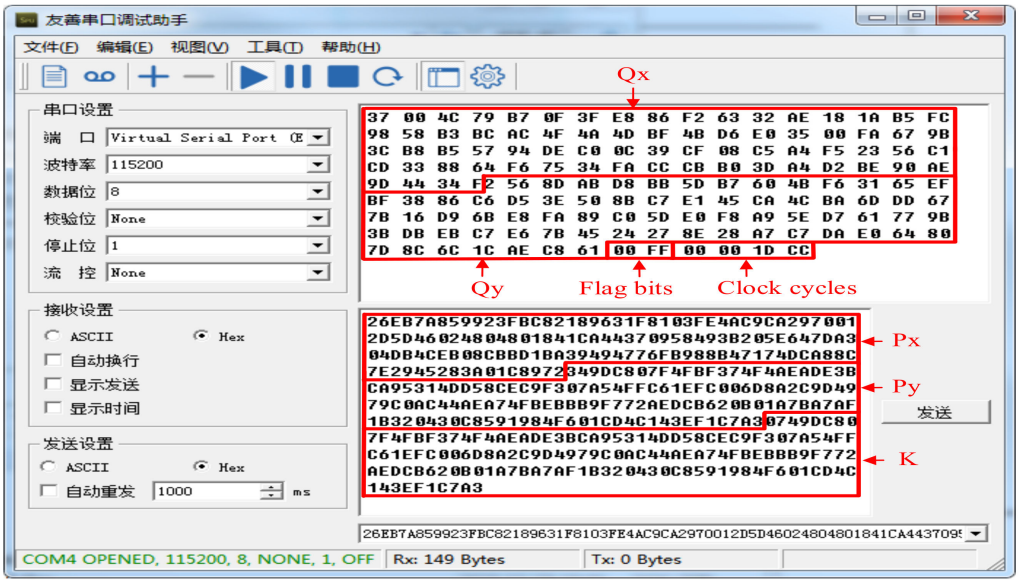

Figure 19. The test result of point multiplication ($m = 571$).

5.2. Results Analysis

Table 4 provides a performance comparison between the system designed in this paper and others proposed in existing studies. In order to compare the speed fairly, the time of point multiplication is converted to the number of working cycles required to complete the point multiplication (shown in the sixth column). The first column of Table 4 is the point multiplication circuit, the second column is the key length value of a point multiplication operation, the third column is the FPGA devices used in the system implementation, the fourth is the hardware resource consumption, the fifth column is the maximum clock frequency of the circuit, the sixth column is the number of clock cycles to perform a point multiplication operation, the seventh column is the time required for the system to complete a point multiplication operation, and the eighth column lists the compatible key length values of a single point multiplication operator.

Table 4. Comparison of the latency of one point multiplication operation in ECC systems with different structures.

Solutions	*m*	FPGA	LUTs	Freq. (MHz)	Cycles	T (μs)	Key Length
[10]	233	V5	25,129	250	1375	5.50	233
	283		25,030	189	6350	33.6	283
	409		28,503	161	16,519	102.6	409
	571		32,432	121	42,108	348	571
[11]	571	V7	141,078	111	3780	34.05	571
[12]	233	V5	-	173	708	4.09	233
	283		-	154	895	5.81	283
	409		-	143	1358	9.50	409
	571		-	121	2240	18.51	571
[19]	233	K7	95,799	120	2321	19.34	163,233,
	283				2819	23.49	283,367
[20]	163	DE10	-	50	1,179,500	23,590	163,233, 283,409
	233				2,546,000	50,920	
	283				3,772,000	75,440	
	409				9,287,000	185,740	
Ours	233	V6	116,241	135	2609	19.33	233,283, 409,571
	283				3018	22.36	
	409				5784	41.36	
	571				7628	56.50	
	233 & 233				2609	19.33	
	283 & 283				3018	22.36	

For effective comparison, we choose some point multiplication methods proposed in [10–12,19,20]. In comparison with diverse point multiplication structures, the circuit designed in this paper is found to process a single key faster than [10,19,20], but slightly slower than structures of [11,12]. Ref. [20] has a point multiplication operation with four key lengths implemented by using reconfigurable modules, but it is much slower than that in this paper. It also demands more power from the perspective of hardware resource consumption. The system's function is improved at the expense of certain hardware power. In comparison, the advantage of the system designed in this paper is that it can calculate the point multiplication of four different key lengths and is capable of processing two keys all together.

6. Conclusions

In order to meet the needs of encryption for the auxiliary data of fuzzy extractor in the lightweight authentication system based on a PUF IoT device, this paper uses FPGA technology to design a binary field ECC point multiplication circuit. In order

to further increase the computation complexity of decrypting the encrypted auxiliary data, this paper implements a point multiplication circuit with reconfigurable key length. Moreover, four key lengths recommended by NIST are taken as the research object, and a point multiplication circuit compatible with four different key lengths is designed. The implementation of the circuit makes a certain contribution to the research of selecting n key lengths from N different key lengths for ECC encryption.

The core unit of ECC is PM, which can reflect the influence of different field multipliers. The Montgomery method and standard projection coordinates are used to realize point multiplication. We consider reusing the field operation module and combining the point addition and point doubling operation to further improve the speed of PM.

The foremost progress made in this paper lies in the design of an ECC algorithm that is compatible with four different key lengths, and the parallel implementation of a pair of ECC algorithms in the case of key length m = 233 or 286, which extends the application scope and improves the efficiency of the system.

The future research exhibition will carry out the following work. In order to find the optimal point multiplication method, the advantages of the multiplication method in different projection coordinates are studied. In order to ensure the security of the algorithm, this study adopts the NIST recommended security parameters, and does not consider the anti-attack strategy of the system. Therefore, the next step is to improve the system performance at the same time, and further improvement of the system's anti-attack strategy is very worthy of in-depth study.

Author Contributions: Conceptualization, X.Z.; methodology, X.Z., L.Z. and Y.Z.; software, X.Z. and L.Z.; validation, X.Z.; formal analysis, X.Z.; resources, X.Z.; writing—original draft X.Z.; writing—review and editing, B.L., L.Z., Y.W. and R.C.; supervision, B.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the ShenZhen Science Technology and Innovation Commission (SZSTI): JCYJ20170817115500476 and JCYJ20170817115538543.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feki, M.A.; Kawsar, F.; Boussard, M.; Trappeniers, L. The Internet of Things: The Next Technological Revolution. *Computer* **2013**, *46*, 24–25. [CrossRef]
2. Kai, Z.; Ge, L. A Survey on the Internet of Things Security. In Proceedings of the 2013 International Conference on Computational Intelligence and Security, Emeishan, China, 14–15 December 2013; pp. 663–667.
3. Tsague, H.D.; Twala, B. Practical Techniques for Securing the Internet of Things (IoT) Against Side Channel Attacks. In *Internet of Things and Big Data Analytics Toward Next-Generation Intelligence*, 1st ed.; Springer: Cham, Germany, 2017; pp. 439–481.
4. Herder, C.; Yu, M.; Koushanfar, F.; Devadas, S. Physical Unclonable Functions and Applications: A Tutorial. *Proc. IEEE* **2014**, *102*, 1126–1141. [CrossRef]
5. Chen, S.; Li, B.; Cao, Y. Intrinsic Physical Unclonable Function (PUF) Sensors in Commodity Devices. *Sensors* **2019**, *19*, 2428. [CrossRef] [PubMed]
6. Chen, S.; Li, B.; Chen, Z.; Zhang, Y.; Wang, C.; Tao, C. Novel Strong-PUF-based Authentication Protocols Leveraging Shamir's Secret Sharing. *IEEE Internet Things J.* **2021**, in press. [CrossRef]
7. Zhang, Y.; Li, B.; Liu, B.; Hu, Y.; Zheng, H. A Privacy-Aware PUFs-Based Multi-Server Authentication Protocol in Cloud-Edge IoT Systems Using Blockchain. *IEEE Internet Things J.* **2021**, in press. [CrossRef]
8. Ulrich, R.; Jan, S.; Frank, S.; Xiaolin, X.; Ahmed, M.; Vera, S.; Gideon, D.; Jürgen, S.; Wayne, B.; Srinivas, D. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1876–1891.
9. Imran, M.; Rashid, M.; Jafri, A.R.; Kashif, M. Throughput/area optimized pipelined architecture for elliptic curve crypto processor. *IET Comput. Digit. Tech.* **2019**, *13*, 361–368. [CrossRef]
10. Sutter, G.; Deschamps, J.; Imana, J. Efficient elliptic curve point multiplication using digit-serial binary field operations. *IEEE Trans. Ind. Electron.* **2013**, *60*, 217–225. [CrossRef]
11. Khan, Z.U.A.; Benaissa, M. High-Speed and Low-Latency ECC Processor Implementation Over GF(2^m) on FPGA. *IEEE Trans. Very Large Scale Integr. Syst.* **2017**, *25*, 165–176. [CrossRef]
12. Li, L.; Li, S. High-performance pipelined architecture of point multiplication on Koblitz curves. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 1723–1727. [CrossRef]
13. Hankerson, D.; Menezes, A.; Springer, S.V. *Guide to Elliptic Curve Cryptography*, 1st ed.; Springer: New York, NY, USA, 2004.

14. Digital Signature Standard. FIPS Standard 186-4. 2013. Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf (accessed on 21 May 2021).
15. Morioka, S.; Katayama, Y. $O(\log_2 m)$ Iterative Algorithm for Multiplicative Inversion in $GF(2^m)$. In Proceedings of the 2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060), Sorrento, Italy, 25–30 June 2000; p. 449.
16. Itoh, T.; Tsujii, S. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Inf. Comput.* **1988**, *78*, 171–177. [CrossRef]
17. Meher, P.K.; Lou, X. Low-Latency, Low-Area, and Scalable Systolic-Like Modular Multipliers for GF(2m) Based on Irreducible All-One Polynomials. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *64*, 399–408. [CrossRef]
18. Renuka, G.; Shree, V.U.; Reddy, P.C. Comparison of AES and DES Algorithms Implemented on Virtex-6 FPGA and Microblaze Soft Core Processor. *Int. J. Electr. Comput. Eng.* **2018**, *8*, 3544–3549. [CrossRef]
19. Li, L. Research on Algorithms and Hardware Implementations for Elliptic Curve Cryptography over Binary Extension Fields. Ph.D. Thesis, Tsinghua University, Beijing, China, 2017.
20. Hasbi, A.; Arif, S.; Yusuf, K. Implementation of ECC on Reconfigurable FPGA Using Hard Processor System. In Proceedings of the 2018 International Symposium on Electronics and Smart Devices (ISESD), Bandung, Indonesia, 23–24 October 2018; pp. 1–6.