Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Research on Elliptic Curve Cryptography Hardware Encryption System Based on FPGA

Huiwei Yang[1,✉]

[1] *Department of Information and Artificial Intelligence, Wuhu Institute of Technology, Wuhu, Anhui, 241000, China*

ABSTRACT

This paper presents a hardware encryption system based on FPGA (Field-Programmable Gate Array) implementing the elliptic curve cryptography algorithm. Using FPGA as the core control unit, IoT (Internet of Things) data transmission terminals are connected to FPGA-specific external interfaces via USB/SPI interfaces. Data collected into the FPGA undergoes encryption and decryption using the FPGA's internal hardware resources. The encrypted data is then converted into TCP/IP protocol packets and transmitted to a cloud server through the FPGA's internal Ethernet interface circuit module. A detailed analysis and design of the hardware implementation of the elliptic curve encryption algorithm are provided. Simulation validation of the point multiplication algorithm was conducted on a computer platform with a quad-core 3.2GHz processor and 8GB of memory, using the Xilinx 5vlx20tff323 chip. The simulation results indicate that the maximum execution frequency reached 372.686 MHz, with a single point multiplication operation completed in 3328 μs. This significantly enhances the processing speed of the algorithm, bearing significant theoretical value and practical implications for advancing the security of the IoT ecosystem.

Keywords: Elliptic Curve; FPGA; IoT; Security of the IoT ecosystem

1. Introduction

As the Internet of Things (IoT) technology is witnessing a boom, the privacy and security issues of IoT devices are of increasing concern. Since these devices have limited resources to spend too much on cryptographic computation, there is an urgent need for secure, efficient cryptographic processors with small resource consumption to meet the security requirements of resource-constrained IoT devices [1-4]. Among the several cryptographic algorithms currently in common use, Elliptic Curve Cryptography (ECC) algorithm provides a higher level of security with a smaller key size and is widely used in many resource-constrained devices.

Elliptic curve cryptography algorithm is an asymmetric public key cryptography algorithm which is based on the elliptic curve discrete logarithm puzzle [5]. Compared with the RSA algorithm based

✉ Corresponding author.
E-mail address: yanghuiwei66@whit.edu.cn (H. Yang).
Received 25 February 2024; Revised 10 June 2024; Accepted 05 November 2024; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-085

on the large integer factorization puzzle, the ECC algorithm has a number of advantages, including shorter key length, higher security, faster computing speed, and less storage space required [6-7]. In recent years, with the continuous improvement of supercomputing capability, the traditional, elliptic curve cryptography schemes that utilize CPUs with limited resources to implement the support of short bits have an increasingly high probability of being successfully cracked, and the security aspect of the information has been greatly threatened [8-11]. In order to guarantee the security of information, the elliptic curve cipher scheme supporting longer bits will inevitably lead to the slowdown of encryption and decryption operations, which can not meet the needs of today's network with high real-time requirements and large data throughput [12-15]. Field-programmable gate arrays (FPGAs) have the advantage of concurrent and streaming multitasking, and can obtain computational speeds several times higher than that of CPUs through the design of highly parallel computational architectures [16-17]. Existing FPGA-based elliptic curve cryptography schemes require long clock cycles to take serial operations during implementation, resulting in a large amount of time to complete an ECC dot product [18-19]. For application scenarios with small logic resources but high requirements for algorithmic security, this design implementation cannot be well adapted for use [20-21]. Therefore, how to implement efficient ECC encryption algorithms in FPGAs and consume lower logic resources has become an urgent problem to be solved.

Loi, K. C. et al. designed a unified ECC coprocessor based on DSP48E slices of Xilinx FPGAs, which efficiently evaluates a wide range of elliptic curve pointwise multiplication (ECPM) performances and combines with an extensible finite-domain arithmetic block to further improve the competitiveness of the proposed ECC coprocessor in terms of timing performance and hardware resource utilization [22]. Mehrabi, M. A. et al. present a low power, low area FPGA implementation of ED25519 and CURVE25519 scalar multiplication for IoT applications, designing a high base interleaved mode multiplication algorithm to reduce the complexity of the hardware implementation, which improves the efficiency of the modulo operations while reducing the dependence on the large integer multipliers of the FPGA DSP module [23]. Kashif, M. et al. developed an elliptic curve cryptoprocessor with high hardware utilization and low power consumption, which is capable of implementing scalar multiplication on standard NIST curves over Galois binary domains using polynomial bases, which is convenient for resource constrained low area and low power embedded system applications [24]. Venkataraman, K. et al. security and computational cost aspects, explored the improvement scheme of elliptic curve digital signature on FPGA, proposed Keccak-f1600 algorithm, which effectively reduces the computational effort of the improved ECDSA algorithm and solves the problem of signature forgery by utilizing the concept of hidden generating point, which brings the IoT devices with higher security and smaller computational resource consumption [25]. Hossain, M. R. et al. proposed a hardware architecture that can implement prime domain modulo operations on FPGAs, and experiments have shown that the proposed architecture achieves better performance in terms of throughput rate as well as resource utilization, and is able to significantly improve the efficiency of the ECC processor [26]. Islam, M. M. et al. investigated the implementation of a high-speed, low-area, side-channel attack-resistant ECC processor in the prime domain on FPGAs by introducing a digital signature scheme, EdDSA, and a hardware architecture that supports 256-bit dot product, which enables the proposed ECC processor to achieve fast scalar multiplication operations with high security at a low hardware utilization [27]. Wang, D. et al. established an efficient low-latency elliptic curve cryptographic architecture based on the NIST- p 256 prime number domain by incorporating the fast partial Montgomery reductive algorithm into the modulo-square unit of the architecture and letting it run in parallel with the modulo-multiplication operation, which greatly improves the speed of the dot-multiplication operation in the ECC processor [28].

Given the backdrop above, and leveraging the parallel execution characteristics of FPGA (Field-Programmable Gate Array), there is a pressing need to research and design a hardware

encryption transmission system based on FPGA for elliptic curve cryptographic algorithms. Such research holds significant theoretical value and practical implications for advancing the security of the IoT ecosystem.

2. System Analysis and Design Approach

IoT (Internet of Things) systems typically comprise sensing units, network transmission units, platform management units, and higher-level application units. Data security is often compromised in the network transmission unit, either through theft or damage. By analyzing the structure of the IoT system in conjunction with the characteristics of data encryption algorithms, innovative research can be conducted on some of the key technologies within the encryption algorithm implementation process, introducing a unique architectural approach. Utilizing FPGA programming, the system design can be realized.

With the FPGA serving as the central control unit, SPI, UART, and other interfaces are designed in the Verilog language based on the I/O requirements of the IoT terminal devices. This completes the design of the interface modules, making them easily upgradable, modifiable, and highly flexible. Data from the IoT sensing unit's transmission terminal is connected to the FPGA's dedicated external interface through USB/SPI interfaces. Once data is collected inside the FPGA, the elliptic curve encryption and decryption module, also designed in the Verilog hardware description language, handles the data encryption and decryption processes. The encryption module, decryption module, key expansion module, and interface module are integrated according to encryption rules and solidified into the FPGA chip, culminating in the hardware design of the encryption system. By utilizing the FPGA's embedded processor core, relevant IP core resources, and integrating with the μCOS-II operating system, an embedded programmable on-chip system is constructed. The TCP/IP protocol is ported onto the μCOS-II platform, transforming encrypted data into TCP/IP protocol data packets. These packets are then sent to the FPGA's internal Ethernet interface circuit module and subsequently transmitted to the cloud server. The overall architecture is illustrated in Figure 1.

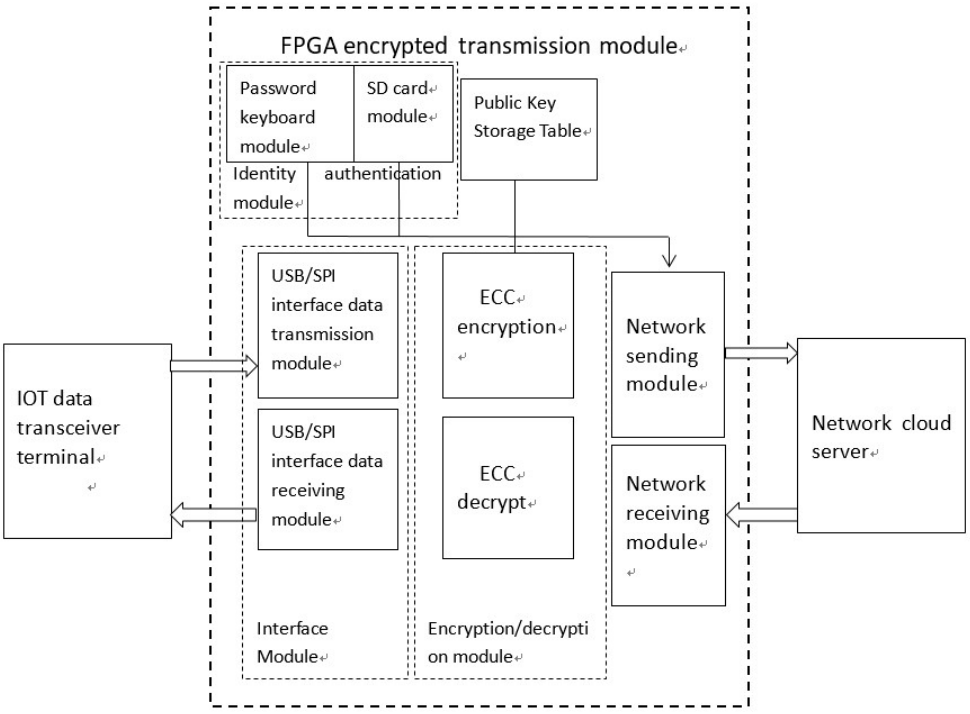

Fig. 1. Architecture of Elliptic Curve Cryptography Algorithm Hardware Encryption Transmission System Based on FPGA

3. Binary Finite Field Elliptic Curve Encryption Algorithm

The elliptic curve equation evolved from Weierstrass elliptic functions. The aforementioned elliptic curve is continuous and cannot be used directly for encryption. To make it applicable, the curve must be converted into discrete points, defining these points on a finite field K. The elliptic curve equation on field K is represented as:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{1}$$

Where coefficients $a_1, a_2, a_3, a_4,$ and a_6 are all elements of field K. The finite field K mainly includes two scenarios: the characteristic of the field being equal to 2 or 3, and the characteristic of the field being different from 2 or 3.

If the characteristic of field K is neither 2 nor 3, then through a compatible transformation, equation (1) can be converted to the curve equation as follows:

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

If the characteristic of field K equals 2, two situations need to be considered depending on whether a_1 is not equal to 0 or a_1 is equal to 0. Through compatible transformations, equation (1) can be respectively transformed into curve equations (3) and (4) as below:

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

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

If the characteristic of field K equals 3, two situations also need to be considered. By compatibly transforming equation (1), one can obtain the non-super singular curve equation (5) and the super singular curve equation analogous to equation (2).

$$y^2 = x^3 + ax^2 + b \tag{5}$$

From the Perspective of Hardware Implementation: Binary Finite Field Elliptic Curve Cryptography

Considering the ease of hardware implementation, the binary field is the most suitable approach, as the computation process only involves binary shifts and bitwise modulo-2 addition. Consequently, equation (3) was chosen as the elliptic curve encryption equation for this system. For users A and B engaged in network data communication, the encryption and decryption process using the elliptic curve algorithm is as follows:

1) User B, based on the binary finite field F_2^m melliptic curve equation $E(a,b)$, $y^2 + xy = x^3 + ax^2 + b$,selects appropriate parameters a and b within the binary finite field F_2^m. Furthermore, a point on the elliptic curve is chosen as the base point G. For security reasons, the coordinates of point G should be sufficiently large.

2) User B then randomly selects a large number k as a private key and generates the public key $P_B = kG$.

3) User B sends $E(a,b)$, the public key PB, and G to User A.

4) Upon receiving the public information, User A encodes the plaintext to point M on $E(a,b)$, generates a random integer r, and then calculates $C_1 = M + rP_B$ and $C_2 = rG$ to complete the encryption. Subsequently, the encrypted text C1 and C2 are sent to User B.

5) Upon receiving C_1 and C_2, User B retrieves the encoded effective data information M from User A by performing the operation $C_1 - kC_2$.

Throughout the encryption and decryption processes, the hardware implementation of the elliptic curve algorithm can be divided into two parts: the Finite Field Operation Module and the Elliptic Curve Operation Control Module. The Finite Field Operation Module primarily encompasses finite field multiplication, finite field addition, finite field squaring, and finite field modular inversion. The

Operation Control Module includes point addition, point doubling, and point multiplication.

4. Algorithm Design for Binary Finite Field Operation Module

The finite field operation module primarily consists of finite field multiplication, addition, squaring, and modular inversion. In binary finite fields, the addition operation is executed bit by bit without the need for carry forwarding. As a result, to obtain the sum, one only needs to perform a bitwise XOR operation on the two operand binary sequences. In hardware implementation, this operation requires only XOR gates, eliminating the need for adders. Multiple XOR gates can execute in parallel, enhancing the overall execution efficiency.

4.1. Binary Finite Field Multiplication

Finite field multiplication is a crucial unit in the entire design process. An efficient multiplication algorithm is imperative for enhancing the processing speed of elliptic curve encryption and decryption. The entire multiplication procedure primarily consists of two parts: multiplication and modular reduction. When implementing with hardware circuit design, it's considered that the elements of binary finite field

2 F_2^m are represented by polynomials. Let the reduction polynomial be represented as: $f(x)=x^m+r(x)$, where $\deg(r)\le m$-1, and $r(x)$ is the remainder polynomial. In the finite field F_2^m, elements are represented by binary polynomials up to a degree of m-1. A conventional approach to designing a multiplier involves directly multiplying two operands of width m. The product is then modulated by the reduction polynomial $f(x)$. This method requires a register of width 2m to store intermediate results, resulting in a relatively lower algorithmic efficiency and an excessive consumption of hardware resources. To improve the algorithm's execution efficiency, the multiplication operation is transformed into shift-add operations. This shift-add implementation for field multiplication is established based on the formula (6):

$$a(x).b(x)=a_{m-1}x^{m-1}b(x)+...a_ix^ib(x)+...+a_1xb(x)+a_0b(x) \tag{6}$$

While iterating over i in the range $i\in[0,m)$ to calculate $x^ib(x)$ mod $f(x)$, if $a_i=1$, the result is added to the accumulator. The left shift operation for $b(x)$ is achieved through $b(x)$. x mod $f(x)$. As the shift takes place, if the highest order bit b_{m-1} is 1, the remainder polynomial $r(x)$ is appended to $b(x)$. The specific algorithm is as follows:

Input: Binary polynomials $a(x)$, $b(x)$ of degree less than m.

Output: $c(x)=a(x).b(x)$ mod $f(x)$

Step 1: If $a_0=1$, then c=b; otherwise, c=0.

Step 2: Define a variable

i from 1 to m-1, and repeatedly execute the following operations:

$b=b.x \bmod f(x)$. If $a_i=1$, then $c=c+b$.

Step 3: Return c.

This method is also highly suitable for hardware implementation. A single shift of the vector can be accomplished within one clock cycle, without requiring any logic units.

4.2. Binary Finite Field Squaring

Binary finite field squaring is actually a special case of finite field multiplication, where $a(x)=b(x)$. Since −1=1 in the binary domain, for all a in F_2^m, we have $a=-a$. After squaring in the binary field, many intermediate terms with identical coefficients and powers of x cancel each other out, leaving

only terms with even powers of x. Hence, squaring a polynomial within the binary finite field is a linear operation and is faster than the multiplication of any two distinct polynomials.

For: $a(x) = a_{m-1}x^{m-1} + \ldots a_i x^i + \ldots a_1 x + a_0$,

Then: $a(x)^2 = a_{m-1}x^{2m-2} + \ldots a_i x^{2i} + \ldots a_1 x^2 + a_0$.

Based on the above characteristics, the squaring operation in a binary finite field can adopt the "alternate zero insertion" method. This involves inserting a zero every other bit, starting from the least significant bit, in the binary representation of $a(x)$. This process doubles the width of a_0 to $2m-2$ bits. The expanded result is then taken modulo the reduction polynomial $f(x)$ to produce the final outcome. This insertion method takes full advantage of the computational property $a = -a$ within the binary finite field. It quickly derives the result, denoted $c(x)$, and is more efficient than the binary finite field multiplication algorithm presented in section 4.1. However, it requires a storage unit with twice the width of) $a(x)$ to store the intermediate results.

By examining the modular arithmetic properties of finite field polynomials, we can deduce that the modulus of the polynomial equals the sum of the modulus of its subterms. Based on the five fast reduction polynomials $f(x)$ recommended by NIST in FIPS 186-2 and considering system features, we choose the maximum data width to be 233 bits. To reduce storage space, the trinomial $f(x) = x^{233} + x^{74} + 1$ is selected as the reduction polynomial. After iteratively calculating $c_i(x).x^i \bmod f(x)$, we observe that as i ranges from 0 to m−1, the effective data items after reduction exhibit a diagonal relationship. By further analysis, the algorithm can be optimized. The final algorithm is as follows

Selection of Reduction Trinomial: $f(x) = x^{233} + x^{74} + 1$

Input: DIN[232:0]with a data width of 233.

Output: DOUT[232:0] with a data width of 233, where $DOUT = DIN^2 \bmod f(x)$.

1) For i ranging from 0 to 74, incremented by 2, execute
$DOUT[i] = DIN[i]^{DIN[196+i>>1)]}$;

2) For i ranging from 1 to 74, incremented by 2, execute
$DOUT[i] = DIN[117 + (i\text{-}1) >> 1]$;

3) For i ranging from 74 to 146, incremented by 2, execute
$DOUT[i] = DIN\left[\frac{i}{2}\right]^{DIN[196+(i-74)]}$;

4) For i ranging from 75 to 232, incremented by 2, execute
$DOUT[i] = DIN[154 + (i-75) >> 1]^{DIN[196+i>>1]}$;

5) For i ranging from 148 to 232, $i = i + 2$, incremented by 2, execute $DOUT[i] = DIN\left[\frac{i}{2}\right]$;

6) Output DOUT[232:0].

Given the inherent parallelism of FPGA, steps 1 to 5 of the above algorithm can be executed in parallel through dataflow descriptive statements. They can be completed within a single clock cycle, and the resource consumption is at most m XOR gates. This results in minimal delay and significantly enhances execution efficiency.

4.3. Binary Finite Field Modular Inversion

In elliptic curve encryption algorithms, division operations are involved. Such operations can be transformed and implemented through multiplicative inverses. For the binary polynomial $a(x) \in F_2^m$, if $g(x)$ is the multiplicative inverse of $a(x)$ in F_2^m, then $a(x)g(x) = 1 \bmod f(x)$ The inverse of $a(x)$ can be directly expressed as $a^{-1}(x)$. The extended Euclidean algorithm for polynomials can efficiently compute the multiplicative inverse. The specific algorithm is as follows:

Input: A non-zero binary polynomial $a(x)$ with a degree of at most $-1m-1$.

Output: $a(x)^{-1} \bmod f(x)$;

1) Set $u=a,\ v=f$.

2) Initialize $g_1=1,\ g_2=0$;

3) While $\neq 1u=1$, repeat the following:

(1) $j=\deg(u)-\deg(v)$;

(2) If $<0j<0$ then $u<->v,\ g_1<->g_2,\ j=-j$;

(3) $u=u+x^j v$;

(4) $g_1=g_1+x^j g_2$;

4) Return 1 g_1, which is the computed inverse.

5. Elliptic Curve Arithmetic Control Module Algorithm Design

Elliptic curve-based encryption systems primarily depend on the arithmetic operations of points on the elliptic curve. The main operations include: point addition, point doubling, and point multiplication.

5.1. Elliptic Curve Point Addition and Doubling Operations

Based on the previously selected binary finite field F_2^m melliptic curve equation $E(a,b): y^2+xy=x^3+ax^2+b$, appropriate parameters a and b are chosen within the binary finite field F_2^m. For point addition, consider two points $P=(x_1,y_1)\in E\left(F_2^m\right)$ and Q=$(x_2,y_2)\in E\left(F_2^m\right)$, where $P\neq \pm Q$. The sum $P+Q$ is denoted as $K(x_3,y_3)\in E\left(F_2^m\right)$. The coordinates of $K(x_3,y_3)$ are calculated as follows: $x_3=\lambda^2+\lambda+x_1+x_2+a$ and $y_3=\lambda(x_1+x_3)+x_3+y_1$, where, $\lambda=\dfrac{(y_1+y_2)}{(x_1+x_2)}$.

For the point doubling operation, which is essentially when $P=Q$, for a point $P(x_1,y_1)\in E\left(F_2^m\right)$, where $P\neq -P$, the result of $2P=(x_3,y_3)\in E\left(F_2^m\right)$ is calculated as: $x_3=\lambda^2+\lambda+a$ and $y_3=x_1^2+\lambda x_3+x_3$,Where, $\lambda=\dfrac{(x_1+y_1)}{x_1}$.

While the definition of point doubling is different from point addition, they are fundamentally the same at their core, and they both utilize the same hardware resources and computation time. Moreover, these two operations will not appear simultaneously during the group operation process. Therefore, combining their functionalities can lead to efficient hardware resource sharing and time-division multiplexing.

5.2. Elliptic Curve Point Multiplication Operation

For a point $P(x,y)\in E\left(F_2^m\right)$, the multiplication of point P with an integer k is denoted as kP, which is equivalent to adding point P k times, termed as point multiplication. The elliptic curve point multiplication operation is based on finite field operations, point addition, and point doubling. Implementing point multiplication in hardware can significantly enhance the efficiency of the encryption processing system, requiring an effective combination of efficient algorithms and hardware structures. Currently, there are three main algorithms for point multiplication operation. The first algorithm is the traditional left-to-right binary method, where k is bitwise shifted, and point addition or point doubling is performed on P based on the value of each bit. For a k with a bit width of m, this algorithm approximately requires m/2 point additions and m point doubling

operations, making it inefficient and storage-intensive, hence rarely used.

The second algorithm is the Non-Adjacent Form (NAF) algorithm. The main idea behind this algorithm is as follows: for $P=(x,y)\in E\left(F_2^m\right)$, them $-P=(x,x+y)$, making subtraction as efficient as addition for elliptic curve points. This allows the use of signed digit representation $k=\sum_{i=0}^{l-1}k^i 2^i$, where k belongs to $k\in\{0\pm1\}$, with $k^{l-1}\neq 0$. This forms the Non-Adjacent Form (NAF) of a positive integer k, with no two consecutive digits being non-zero. The digits of NAF(k) can be obtained by continuously dividing k by 2, allowing for remainders of 0 or ±1±1. If k is odd, the remainder r belongs to $r\in\{-1,1\}$, ensuring the quotient $\frac{(k-r)}{2}$ is even, making the next NAF digit 0. Finally, the binary representation of k is replaced with NAF(k) to improve the left-to-right binary point multiplication algorithm. The NAF algorithm can reduce the expected runtime to /3m/3 point additions and m point doubling operations but increases the consumption of hardware resources for redundant encoding circuits.

The third algorithm is based on the Montgomery method, utilizing the LD projection coordinates to implement point multiplication operation on non-supersingular elliptic curves $y^2+xy=x^3+ax^2+b$ in the binary field. An advantage of this algorithm is that it doesn't require additional memory and operates the same way in each iteration of the main loop, enhancing resistance to timing analysis and power analysis attacks. After comparison, this method is selected for the system to implement point multiplication operation. The specific implementation of Montgomery point multiplication is as follows:

Input: $k=(k_{t\text{-}1},\ldots,k_1,k_0)_2$, where $k_{t\text{-}1}=1, P=(x,y)\in E\left(F_2^m\right), k<n$.

Output: kP.

1) Compute (P,2P), $X_1=x, Z_1=1, X_2=x^4+b, Z_2=x^2$;

2) For i from t-2 to 0, repeat the following steps:

(1) If $k_i=1$. then

$T=Z_1, Z_1=\left(X_1Z_2+X_2Z_1\right)^2, X_1=xZ_1+X_1X_2TZ_2$

$T=X_2, X_2=X_2^4+bZ_2^4, Z_2=T^2Z_2^2$

(2) Otherwise:

$T=Z_2, Z_2=\left(X_1Z_2+X_2Z_1\right)^2, X_2=xZ_2+X_1X_2Z_1T$

$T=X_1, X_1=X_1^4+bZ_1^4, Z_1=T^2Z_1^2$

3) $x_3=\frac{X_1}{Z_1}$;

4) $y_3=\left(x+\frac{X_1}{Z_1}\right)\left[\left(X_1+xZ_1\right)\left(X_2+xZ_2\right)+\left(x^2+y\right)\left(Z_1Z_2\right)\right]\left(xZ_1Z_2\right)^{-1}+y$

5) Return (x_3,y_3).

During the implementation, considering the slow speed of a fully serial structure multiplier and the large hardware overhead of a fully parallel structure, a hybrid serial-parallel structure is adopted within the FPGA, taking into account its internal resources. This approach performs N iterative operations of addition, left shift, and reduction in each clock cycle. This method not only achieves a manifold increase in speed compared to a serial structure but also addresses the hardware overhead issue.

6. Hardware Implementation and Simulation Verification

Through the design and analysis process of the elliptic curve encryption algorithm, it can be seen

that the entire process mainly involves point addition and point multiplication operations. Among them, point multiplication operation is at the highest level in the entire encryption and decryption operation hierarchy, and resource consumption and efficiency of the entire elliptic curve encryption and decryption are very important. On a computer platform with a quadcore 3.2GHz processor and 16GB of memory, the XC5vlx30t-2ff665 programmable FPGA chip of the Virtex 5 series from Serex is selected as the main controller based on the scale of the algorithm. In the ISE Design Suite 14.7 integrated open environment, a top-down modular design method is adopted and Verilog language is used to implement the elliptic curve encryption and decryption point multiplication algorithm design。As shown in Figure 2:

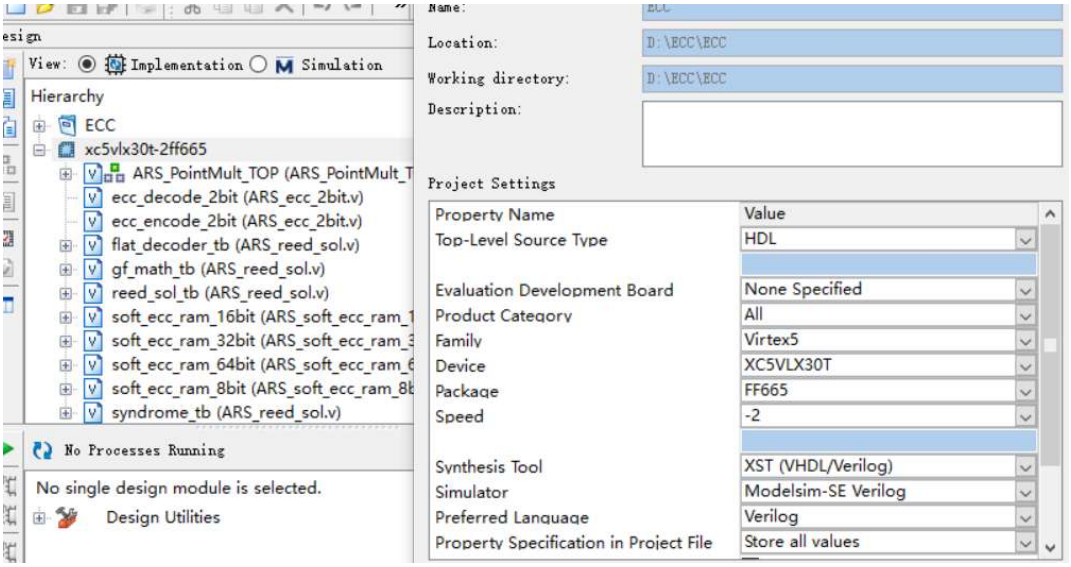

Fig. 2. The Project Settings

The ViewRTL Schematic result after Synthesize is show in Figure 3:

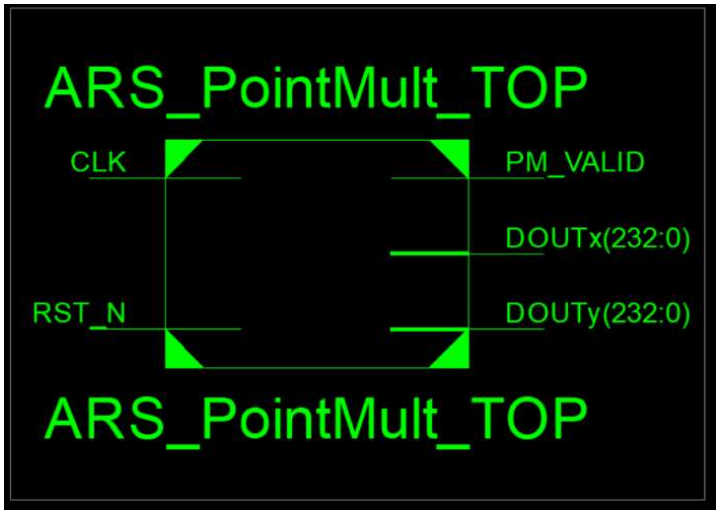

Fig. 3. The ViewRTL Schematic result after Synthesize

The analysis report on resource utilization is shown in Table 1:

Table 1. Statistical Table of Resource Occupation

Selected Device: xc5vlx30t-2ff665			
Number of Slice Registers	45 out of	19,200	1%
Number of Slice LUTs	50 out of	19,200	1%
Number used as logic	50 out of	19,200	1%
Number of LUT Flip Flop pairs used	66		
Number with an unused Flip Flop	21 out of	66	31%
Number with an unused LUT	16 out of	66	24%
Number of fully used LUT-FF pairs	29 out of	66	43%
Number of unique control sets	12		
Number of slice register sites lost to control set restrictions	31 out of	19,200	1%

Combined with the third-party emulator Modelsim to complete the point multiplication operation function for simulation verification, two groups 233 coordinates and k values were selected:

First Set:

$DINx = 233'h17232BA853A7E731AF129F22FF4149563A419C26BF50A4C9D6EEFAD6126$;

$DINy = 233'h1DB537DECE819B7F70F555A67C427A8CD9BF18AEB9B56E0C11056FAE6A3$;

$K = 233'h800000000000000000000000069D5BB915BCD46EFB1AD5F173ABDE$;

The resulting coordinates after obtaining point multiplication are:

$DOUTx: 233'h17232BA853A7E731AF129F22FF4149563A419C26BF50A4C9D6EEFAD6126$;

$DOUTy: 233'hA961C769D267C4EDFE7CA84830333DAE3FE848806E5CAC5C7EB9578785$;

The first simulation results are shown in the Figure 4:

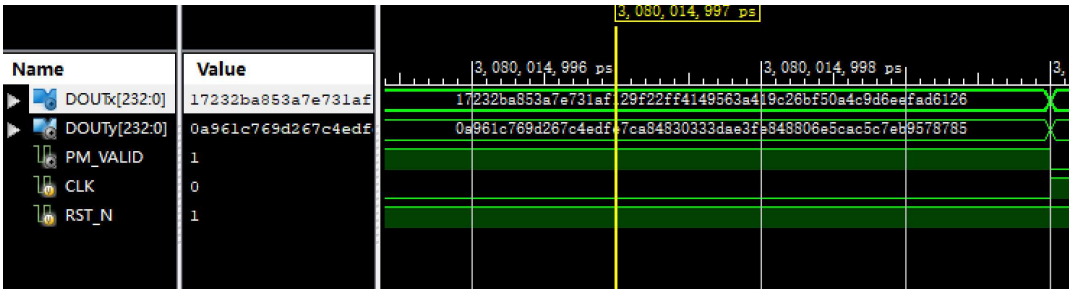

Fig. 4. The first simulation results

SecondSet:

$DINx = 233'h17232BA853A7E731AF129F22FF4149563A419C26BF50A4C9D6EEFAD6126$;

$DINy = 233'hA961C769D267C4EDFE7CA84830333DAE3FE848806E5CAC5C7EB9578785$;

$K = 233'h\ 5232B4353A7E3DF2AF129F4D4314958FA2419DE26BF50A4ACEEFADE25$;

The resulting coordinates after obtaining point multiplication are:

$DOUTx: 233'h\ 60763B0C91DAC37F10E57C1E889B3B6D7DD25E5A6A3E9072160C52460F$;

$DOUTy: 233'h\ 2FB152CDB980F34747A03BA4A3BAE7D37A95BB4B73130B18F82747B127$;

The second simulation results are shown in the Figure 5:

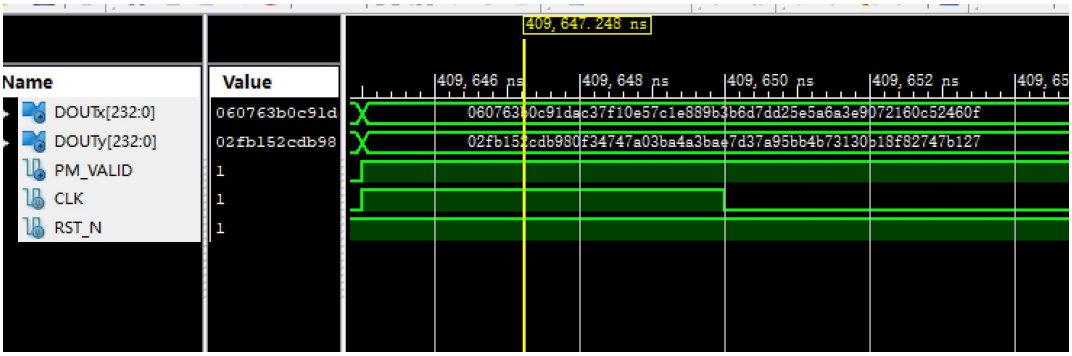

Fig. 5. The second simulation results

The simulation experiment results show that the maximum execution frequency is 372.686Mhz, and the time to complete a point multiplication operation is 3328 μs , Greatly improved algorithm implementation efficiency.

7. Conclusion

Combining the programmable and parallel running characteristics of FPGA, a hardware encryption system based on FPGA elliptic curve cryptography algorithm is studied and designed, and innovative research is conducted on some key technologies in the implementation steps of the encryption algorithm. (1) During the entire encry- ption and decryption process, the algorithm is divided into two parts: a finite field op- eration module and an operation control module. (2) Using binary sequences of two operands for bitwise XOR operations to implement addition operations on binary finite fields, multiple XOR gates are used for parallel execution to improve execution efficiency. (3) Transforming binary finite field multiplication operations into shift addition operations, achieved through iterative accumulation combined with left shift operations, improves algorithm execution efficiency. (4) The binary finite field square operation adopts an improved algorithm based on the interval zero interpolation algorithm, and selects the maximum data width of 233 bits according to the system characteristics. The trinomial $f(x) = x^{233} + x^{74} + 1$ is selected as the reduction polynomial. Parallel execution of data flow description statements in FPGA improves algorithm execution efficiency. (5) Convert division operations into multiplication inverse operations, using the extended Euclidean algorithm of polynomials for implementation. (6) Using the Montgomery algorithm, point multiplication is achieved using LD projection coordinates.

In the process of implementing point multiplication in FPGA hardware, a hybrid implementation scheme of serial and parallel is adopted to complete N iterations of accumulation, left shift, and reduction operations within a single clock cycle, which can greatly improve the computational speed and reduce hardware overhead. Through comprehensive simulation verification in ISE Design Suite 14.7+Modelsim, the results show that the algorithm system has high execution efficiency and moderate resource consumption. It can provide a basis for encrypted transmission of IoT data.

Funding

This work was supported by the Anhui University Natural Science Research Project (no. 2024AH052016; no. KJ2021A1332) and Anhui Province Quality Engineering Teaching Research Project (no. 2022jyxm1778).

References

[1] Lara-Nino, C. A., Diaz-Perez, A., & Morales-Sandoval, M. (2020). Lightweight elliptic curve cryptography accelerator for internet of things applications. Ad Hoc Networks, 103, 102159.

[2] Alhayani, B. S., Hamid, N., Almukhtar, F. H., Alkawak, O. A., Mahajan, H. B., Kwekha-Rashid, A. S., ... & Alkhayyat, A. (2022). Optimized video internet of things using elliptic curve cryptography based encryption and decryption. Computers and Electrical Engineering, 101, 108022.

[3] Abdaoui, A., Erbad, A., Al-Ali, A. K., Mohamed, A., & Guizani, M. (2021). Fuzzy elliptic curve cryptography for authentication in Internet of Things. IEEE Internet of Things Journal, 9(12), 9987-9998.

[4] Durairaj, M., & Muthuramalingam, K. (2018). A new authentication scheme with elliptical curve cryptography for internet of things (IoT) environments. Int. J. Eng. Technol, 7(2.26), 119-124.

[5] Ifrim, R., Loghin, D., & Popescu, D. (2024). A Systematic Review of Fast, Scalable, and Efficient Hardware Implementations of Elliptic Curve Cryptography for Blockchain. ACM Transactions on Reconfigurable Technology and Systems, 17(4), 1-33.

[6] Leelavathi, G., Shaila, K., & Venugopal, K. R. (2016, November). Elliptic Curve Cryptography implementation on FPGA using Montgomery multiplication for equal key and data size over GF (2 m) for Wireless Sensor Networks. In 2016 IEEE Region 10 Conference (TENCON) (pp. 468-471). IEEE.

[7] Kalaiarasi, M., Venkatasubramani, V. R., Vinoth Thyagarajan, V., & Rajaram, S. (2022). A parallel elliptic curve crypto-processor architecture with reduced clock cycle for FPGA platforms. The Journal of Supercomputing, 78(13), 15567-15597.

[8] Arunachalam, K., & Perumalsamy, M. (2022). FPGA implementation of time-area-efficient Elliptic Curve Cryptography for entity authentication. Informacije MIDEM, 52(2), 89-103.

[9] Abu Khadra, S., Abdulrahman, S. E. S., & Ismail, N. A. (2020). Towards efficient FPGA implementation of elliptic curve crypto-processor for security in IoT and embedded devices. Menoufia Journal of Electronic Engineering Research, 29(2), 106-118.

[10] Hossain, M. S., Saeedi, E., & Kong, Y. (2016, February). High-Performance FPGA Implementation of Elliptic Curve Cryptography Processor over Binary Field GF (2^ 163). In ICISSP (pp. 415-422).

[11] Morales-Sandoval, M., Flores, L. A. R., Cumplido, R., Garcia-Hernandez, J. J., Feregrino, C., & Algredo, I. (2021). A Compact FPGA‐Based Accelerator for Curve‐Based Cryptography in Wireless Sensor Networks. Journal of Sensors, 2021(1), 8860413.

[12] Wu, T., & Wang, R. (2019). Fast unified elliptic curve point multiplication for NIST prime curves on FPGAs. Journal of Cryptographic Engineering, 9(4), 401-410.

[13] Javeed, K., & Wang, X. (2017). Low latency flexible FPGA implementation of point multiplication on elliptic curves over GF (p). International Journal of Circuit Theory and Applications, 45(2), 214-228.

[14] Javeed, K., El-Moursy, A., & Gregg, D. (2023). EC-crypto: Highly efficient area-delay optimized elliptic curve cryptography processor. IEEE Access, 11, 56649-56662.

[15] Lara-Nino, C. A., Diaz-Perez, A., & Morales-Sandoval, M. (2018). Elliptic curve lightweight cryptography: A survey. IEEE Access, 6, 72514-72550.

[16] Hossain, M. S., Kong, Y., Saeedi, E., & Vayalil, N. C. (2017). High‐performance elliptic curve cryptography processor over NIST prime fields. IET Computers & Digital Techniques, 11(1), 33-42.

[17] Kamalakannan, V., & Tamilselvan, S. (2018). Fpga implementation of elliptic curve cryptoprocessor for perceptual layer of the internet of things. EAI Endorsed Transactions on Security and Safety, 5(15), e4-e4.

[18] Matutino, P. M., Araujo, J., Sousa, L., & Chaves, R. (2017, July). Pipelined FPGA coprocessor for elliptic curve cryptography based on residue number system. In 2017 international conference on embedded computer systems: architectures, modeling, and simulation (SAMOS) (pp. 261-268). IEEE.

[19] Verri Lucca, A., Mariano Sborz, G. A., Leithardt, V. R. Q., Beko, M., Albenes Zeferino, C., & Parreira, W. D. (2020). A review of techniques for implementing elliptic curve point multiplication on hardware. Journal of Sensor and Actuator Networks, 10(1), 3.

[20] Mukhtar, N., Mehrabi, M. A., Kong, Y., & Anjum, A. (2018). Machine-learning-based side-channel evaluation of elliptic-curve cryptographic FPGA processor. Applied Sciences, 9(1), 64.

[21] Venugopal, E., & Hailu, T. (2018, March). FPGA based architecture of elliptic curve scalar multiplication for IOT. In 2018 Conference on Emerging Devices and Smart Systems (ICEDSS) (pp. 178-182). IEEE.

[22] Loi, K. C., & Ko, S. B. (2018). Flexible elliptic curve cryptography coprocessor using scalable finite field arithmetic blocks on FPGAs. Microprocessors and Microsystems, 63, 182-189.

[23] Mehrabi, M. A., & Doche, C. (2019). Low-cost, low-power FPGA implementation of ED25519 and CURVE25519 point multiplication. Information, 10(9), 285.

[24] Kashif, M., Cicek, I., & Imran, M. (2019, November). A hardware efficient elliptic curve accelerator for FPGA based cryptographic applications. In 2019 11th International Conference on Electrical and Electronics Engineering (ELECO) (pp. 362-366). IEEE.

[25] Venkataraman, K., & Sadasivam, T. (2019). FPGA implementation of modified elliptic curve digital signature algorithm. Facta Universitatis, Series: Electronics and Energetics, 32(1), 129-145.

[26] Hossain, M. R., & Hossain, M. S. (2019, February). Efficient fpga implementation of modular arithmetic for elliptic curve cryptography. In 2019 International conference on electrical, computer and communication engineering (ECCE) (pp. 1-6). IEEE.

[27] Islam, M. M., Hossain, M. S., Hasan, M. K., Shahjalal, M., & Jang, Y. M. (2019). FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field. IEEE Access, 7, 178811-178826.

[28] Wang, D., Lin, Y., Hu, J., Zhang, C., & Zhong, Q. (2023). FPGA implementation for elliptic curve cryptography algorithm and circuit with high efficiency and low delay for IoT applications. Micromachines, 14(5), 1037.