World Academy of Science, Engineering and Technology
International Journal of Cognitive and Language Sciences
Vol:17, No:11, 2023

A Design of Elliptic Curve Cryptography Processor Based on SM2 over GF(p)

Shiji Hu, Lei Li, Wanting Zhou, Daohong Yang

Abstract—The data encryption is the foundation of today's communication. On this basis, to improve the speed of data encryption and decryption is always an important goal for high-speed applications. This paper proposed an elliptic curve crypto processor architecture based on SM2 prime field. Regarding hardware implementation, we optimized the algorithms in different stages of the structure. For modulo operation on finite field, we proposed an optimized improvement of the Karatsuba-Ofman multiplication algorithm and shortened the critical path through the pipeline structure in the algorithm implementation. Based on SM2 recommended prime field, a fast modular reduction algorithm is used to reduce 512-bit data obtained from the multiplication unit. The radix-4 extended Euclidean algorithm was used to realize the conversion between the affine coordinate system and the Jacobi projective coordinate system. In the parallel scheduling point operations on elliptic curves, we proposed a three-level parallel structure of point addition and point double based on the Jacobian projective coordinate system. Combined with the scalar multiplication algorithm, we added mutual pre-operation to the point addition and double point operation to improve the efficiency of the scalar point multiplication. The proposed ECC hardware architecture was verified and implemented on Xilinx Virtex-7 and ZYNQ-7 platforms, and each 256-bit scalar multiplication operation took 0.275ms. The performance for handling scalar multiplication is 32 times that of CPU (dual-core ARM Cortex-A9).

Keywords—Elliptic curve cryptosystems, SM2, modular multiplication, point multiplication.

I. Introduction

ELLIPTIC Curve Cryptography (ECC) is an asymmetric encryption algorithm based on the mathematical theory of elliptic curves. ECC has a higher security level when using the same length key than the RSA public key cryptography algorithm, and it was first proposed by Koblitz [1] and Miller [2] in 1985. As a public key cryptography research focus, the ECC algorithm has been proven to have great performance and security advantages and has been widely used in many fields. SM2 Elliptic Curve public key Cryptography algorithm [3] is an ECC-based public key cryptography algorithm published by China's State Cryptography Administration (SCA) in 2010. It is an ECC algorithm that defines a pseudo-Mersenne prime field with a special prime number P_{256}. The SM2 protocol provides functions such as public key encryption and decryption, signature verification, and key exchange based on the SM2 elliptic curve public key cryptography algorithm. As a national standard algorithm, SM2 has been applied to electronic authentication systems, key management systems, and other fields [4], [5].

The implementation of the elliptic curve cryptosystem consists of four levels from top to bottom: protocol layer, scalar point multiplication (PM), point add (PA) and point double (PD), and finite field operation. The finite field operation layer includes four basic algorithms: modular addition (MA), modular subtraction (MS), modular multiplication (MM), and modular inverse (MI). The protocol implementation of the elliptic curve cryptosystem depends on the mixed call of each level operation on the elliptic curve group. As a top layer of the elliptic curve algorithm, PM directly determines the computational speed of the whole elliptic curve cryptographic architecture. PM depends on PA and PD, and PA and PD depend on operations at the finite field layer. Therefore, the point operation has problems, such as large computation, complex structure, low algorithm efficiency, and high resource consumption.

In the four modular operations of the finite field layer, the computational complexity of modular addition and subtraction can be ignored, while the calculation of modular inversion can be avoided by the transformation from an affine coordinate system to a projective coordinate system. Therefore, modular multiplication is the key of the finite field operation layer. In addition, the implementation of PA and PD is based on continuous calls to finite field operations, so parallel processing of their computations without data association is the key to their acceleration. For PM, the scalar multiplication algorithm complexity directly affects the performance of the whole ECC processor. Therefore, many scholars have researched the ECC algorithm to solve the above problems [6], [7].

The implementation of the modular multiplication algorithm can be divided into two types: interleaved modular multiplication and the multiplication-then-reduce method. The Montgomery algorithm based on interleaved modular multiplication is often used in lightweight ECC processors [8], [9]. Montgomery algorithm uses the bit shift method instead of the costly division method, which can achieve good resource utilization, but its disadvantage is high cycle delay. So, the full-word multiply-then-reduce method is often an attractive option for high-speed design. However, as the length of the key increases, the resources required by the full-word multiplier also increase. So, some optimized algorithms, such as the Karatsuba algorithm, have also appeared [10]-[12]. There are two methods for Instruction

Shiji Hu is with the Integrated Circuit Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China (e-mail: shiji_hu@163.com).
Lei Li is with the Integrated Circuit Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China (e-mail: lilei_uestc@uestc.edu.cn).
Wanting Zhou and Daohong Yang are with University of Electronic Science and Technology of China.

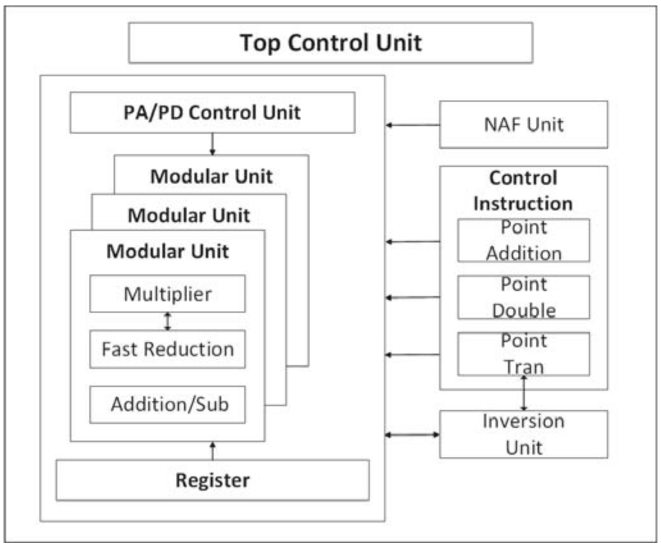


Fig. 1 Architecture of proposed ECC processor

Level Parallelism (ILP): on-the-fly ILP and Synthesis-ILP. The former is in the running stage by the scheduler to process the instruction to achieve parallel needs extra overhead to process the instruction. The other is implemented before the hardware synthesis is generated, which is less flexible but avoids extra computation. PM algorithms include the binary expansion scanning method, the Non-adjacent form (NAF) scanning method, the NAF sliding window method, and the Montgomery point multiplication algorithm. Although many efficient ECC designs have been proposed, the demand for data encryption speed also increases with the continuous development of communication technology. Therefore, more efficient design and better parallel schemes are necessary.

According to the above discussion, this paper presented an ECC processor, as shown in Fig. 1, which is based on the proposed algorithm implementation architecture. On the basis of realizing the modularization of four finite field operations, three groups of MM and MA/MD computing units are instantiated, and their interfaces are packaged with standardized instructions. Then, before hardware synthesis, a projective coordinate point operation instruction list is designed to call a three-layer parallel finite field operation unit, and the corresponding PA, PD, and coordinate system transformation operations are implemented according to the input preset instructions. Finally, based on the scalar multiplication algorithm, the PM operation is realized by calling the point addition and double point operation units in the way of the state machine.

In summary, we have made the following contributions in this paper:

- We proposed a FULL-WORD multiplier structure based on the KO algorithm. In this structure, we avoided the m+1 bit wide multiplication in the KO algorithm by introducing a single parallel addition to realize the pre-operation, furthermore reducing the bit width of addition by shifting the data in advance. By multiplexing three m-bit width multipliers, using twice ko algorithm, and introducing pipeline structure while optimizing the critical path, the realization of 4-m bit large number multiplication only after 6 clock cycles.
- A three-level parallel instruction structure is proposed to implement projective coordinate point operation. Based on the recommended P_{256} module, a synthesis-ILP approach was adopted to design a projective coordinate point operation parallel structure in advance. Combined with the scalar multiplication algorithm, based on the proposed parallel operation structure, the pre-operation of PA and PD is added to idle instructions so that one round of modular multiplication can be reduced in the alternating operation of PA and PD to improve computational efficiency.

II. Modular Operation Module over SM2 Prime Field

In finite field modulo operation, the computing unit includes modular multiplication, modular addition and subtraction, and modular inversion. The modular multiplication unit can be divided into two parts: multiply and reduce.

A. *Proposed Multiplication Method*

The idea for the traditional KOA algorithm [13] is to divide and then calculate, using a recursive way to decompose a complex multiplication operation into multiple simple multiplication operations, which is faster and more efficient than the traditional multiplier. For two m-bits, the complexity is $O(m^2)$ if you multiply them directly. If the KOA algorithm is used, the complexity can be reduced to $O(m^{lb3})$. Directly using the FULL-word multiplier to multiply two m digits is too costly and inefficient. However, Algorithm 1 can be used to realize the full word multiplier by calculating m/2 bit multipliers through half-byte multipliers.

Algorithm 1 Karatsuba-Ofman Multiplication Algorithm

```
Input: $A$: $2m$ bit integer, satisfy $A = a_1 \times 2^m + a_0$ $B$: $2m$
    bit integer, satisfy $B = b_1 \times 2^m + b_0$
Output: $C$: $4m$ bit product, satisfy $C = A \times B$.
1: $P_{00} \leftarrow a_0 \times b_0$; $a_{sum} \leftarrow a_0 + a_1$;
2: $P_{11} \leftarrow a_1 \times b_1$; $b_{sum} \leftarrow b_0 + b_1$;
3: $P_{ss} \leftarrow a_{sum} \times b_{sum}$;
4: $M \leftarrow P_{ss} - P_{00} - P_{11}$;
5: $C \leftarrow P_{11} \times 2^{2m} + M \times 2^{2m} + P_{00}$;
6: return C
```

Based on the original algorithm, in order to further optimize the critical path and reduce the consumption of hardware resources. The optimized Karatsuba-Ofman multiplication algorithm is presented in Algorithm 2.

The proposed algorithm is an improved version of Algorithm 1, and the structure is optimized and modified. The aim is to increase the maximum frequency and improve the efficiency. In Algorithm 1, half-word multiplications are reduced. However, two additional half-word additions and a full-word addition are introduced, and one of the m-bit

Algorithm 2 Proposed Multiplication Algorithm

```
Input: A: $2m$ bit integer, satisfy $A = a_1 \times 2^m + a_0$ B: $2m$ bit integer, satisfy $B = b_1 \times 2^m + b_0$
Output: C: $4m$ bit product, satisfy $C = A \times B$.
 1: $P_{00} = a_0 \times b_0$; $a_{sum} = a_0 + a_1$;
 2: $P_{11} = a_1 \times b_1$; $b_{sum} = b_0 + b_1$;
 3: $S = \{a_{sum}[0], b_{sum}[0]\}$
 4: if $S == 2'b00$ then
 5:     $Z = 0$;
 6: else if $S == 2'b01$ then
 7:     $Z = a_{sum}$;
 8: else if $S == 2'b10$ then
 9:     $Z = b_{sum}$;
10: else if $S == 2'b11$ then
11:     $Z = a_{sum} + b_{sum} - 1$;
12: end if
13: $P_{ss} = a_{sum}[m:1] \times b_{sum}[m:1]$;
14: $C = \{P_{11}, P_{00}[2m-1:m]\} - P_{00} - P_{11}$;
15: $C = C + \{P_{ss}, 2'b0\} + Z$;
16: $C = \{C, P_{00}[m-1:0]\}$;
17: return C
```

TABLE I
Z VALUES BASED ON $a_{sum}[0]$ AND $b_{sum}[0]$

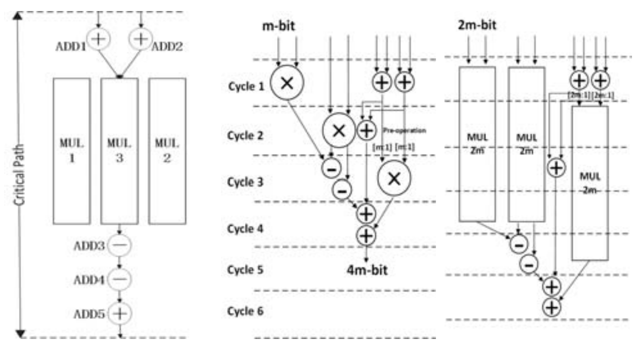
$a_{sum}[0]$	$b_{sum}[0]$	Z
0	0	0
0	1	a_{sum}
1	0	a_{sum}
1	1	$a_{sum} + b_{sum} - 1'b1$

multiplications is changed to (m+1) bit multiplications. In Algorithm 2, we convert the m+1 bit multiplication to m bit multiplication by pre-calculating the last two digits of the (m+1) bit multiplication in advance by adding an addition of parallel operations. It avoids resource waste and extra calculation delay when the (m+1) bit multiplier is reused. The (m+1) bit multiplier can be expressed as (1):

$$\begin{aligned} &a_{sum}[m:0] * b_{sum}[m:0] = \\ &(a_{sum}[m:1] * b_{sum}[m:1]) << 2 + a_{sum}[0] * b_{sum}[0] \\ &+ (a_{sum}[m:1] * b_{sum}[0]) << 1 \\ &+ (a_{sum}[0] * b_{sum}[m:1]) << 1; \end{aligned} \tag{1}$$

$$\begin{aligned} Z =& a_{sum}[0] * b_{sum}[0] + (a_{sum}[m:1] * b_{sum}[0]) << 1 + \\ & (a_{sum}[0] * b_{sum}[m:1]) << 1; \\ a_{sum}&[m:0] * b_{sum}[m:0] \\ &= (a_{sum}[m:1] * b_{sum}[m:1]) << 2 + Z; \end{aligned} \tag{2}$$

Then, the result influence of the corresponding $a_{sum}[0]$ and $b_{sum}[0]$ on Z is shown in Table I. Obviously, for the result of Z, only when the $Z = a_{sum} + b_{sum} - 1$, it requires additional parallel addition. Besides, the step15 of Algorithm 2 requires an additional addition. However, a reasonable pipeline structure can reduce the influence of such additions on the path.



(a) Cascading KO Structure (b) Our 2m-bit mult pipeline structure (c) Our 4m-bit mult pipeline structure

Fig. 2 Mult pipeline structure

In addition, we can optimize the addition and subtraction at the end of algorithm 1. It can be noted that step5, there is no overlap between $P_{11} * 2^m$ and P_{00}, so this addition can be substituted by bit concatenation as shown in (3). At the same time, it can be found that the output low m bit data are only affected by the parameter P_{00}. Therefore, the calculation of a low m bit in the process of addition and subtraction can be ignored. Thus, the bit width of the adder can be reduced by m bits, further reducing the resource consumption. Finally, the low mb data is added to the output again by bit concatenation, as shown in (4). Finally, replacing P_{ss} in algorithm 1 with $P_{ss} + Z$ becomes step 15-16 in the proposed algorithm.

$$C = \{P_{11}, P_{00}\} - P_{00} * 2^m - P_{11} * 2^m + P_{ss} 2^m; \tag{3}$$

$$C = \{P_{11}, P_{00}\} >> 2^m - P_{00} - P_{11} + P_{ss}; \tag{4}$$

According to Algorithm 1, multiplication is preceded by two parallel addition operations and followed by three serial addition and subtraction operations. If the Cascading KO structure Fig. 2(a) is adopted in implementing the 4m-bit multiplier with two iterations of the KO algorithm based on the m-bit multiplier, it would result in huge critical path delays. Based on the existing pipeline structure [14], we analyze the critical path of Algorithm 2, so that multiplication and addition are executed in parallel in the operation process to reduce the critical path length. The optimized improvement of Algorithm 2 realizes 2m bit multiplication by multiplexing an m bit multiplier in pipeline structure with the delay of 4 cycles Fig. 2(b) . Then three proposed 2m bit multipliers are used for parallel multiplexing. After two additional addition cycles, it takes 6 cycles to realize the multiplication of 4m bit large numbers, Fig. 2(c) .

B. Fast Reduction Method

As a pseudo-Mersenne prime, SCA-256 prime (P_{256}) is shown in (5). Or P_{256} can be expressed in a fast reduction form such as (6). Thus, the modulo formula for the higher power of 2 can be derived as follows, shown as (7):

$$\begin{aligned} P_{256} =& 0xFFFFFFFE\_FFFFFFFF\_FFFFFFFF\_ \\ & FFFFFFFF\_FFFFFFFF\_00000000\_ \\ & FFFFFFFF\_FFFFFFFF \end{aligned} \tag{5}$$

$$P_{256} = 2^{256} - 2^{224} - 296 + 264 - 1 \tag{6}$$

$$\begin{aligned} 2^{256} \equiv& 2^{224} + 2^{96} - 264 + 1 (modP_{256}) \\ 2^{288} \equiv& 2^{224} + 2^{128} - 264 + 2^{32} + 1 (modP_{256}) \\ & ... \\ & ... \end{aligned} \tag{7}$$

$$\begin{aligned} C =& (c_{15}, ..., c_2, c_1, c_0) \\ C =& c_{15} * 2^{480} + c_{14} + 2^{448} + ... + c_1 * 2^{32} + c_0 \end{aligned} \tag{8}$$

The result of a 512-bit multiplication can be split 32 bits wide and divided into 16 parts with different weights, as shown in (8). Applying the modulo operation to C, the above modulo (7) for higher powers of 2 can be brought into (8). Then, the modulo reduction of 512-bits based on P_{256} can be converted into a fixed number of addition and subtraction.

Compared with traditional modular operation, the algorithm complexity can be greatly reduced by the way of fast modular reduction. In addition, on this basis, the computational complexity can be reduced again by pre-calculating repeated operations [15]. By combining and decomposing the reduction algorithm, we can find that many parameters are calculated repeatedly in the whole operation process, such as c[12]+c[13]+c[14]+c[15]. In addition, the critical path delay in the process of modulo reduction is reduced by inserting a pipeline. Finally, the 512-bit number is reduced by 4 cycles.

C. Optimized Modular ADD/SUB

In order to reduce resource consumption, the modular addition and modular subtraction units are implemented in one module. The module selects different initial values for calculation according to the modular addition and subtraction mode. The result of Modular addition may exceed P_{256}, so P_{256} needs to be subtracted from the result. Modular subtraction is probably going to be negative, so it needs to add P_{256}. Finally, the final output result is determined according to the overflow flag bit, as shown in Algorithm 3.

In Algorithm 3. SEL is the mode selection. According to the different modes of modular addition and subtraction, m and b are judged to be complement codes, and the calculation is completed through steps 6 to 7, where c_0 is the overflow flag bit, and the final calculation result can be determined according to the value of c_0 and SEL, its structure is shown in the Fig. 3.

Algorithm 3 Optimized Modular Add/Sub

```
Input: $a$, $b$, $SEL$
Output: $out = (a + / - b)modP_{256}$
1: if $SEL ==' ADD'$ then
2:     $m = P_{256} + 1'b1$;
3: else if $SEL ==' SUB'$ then
4:     $b = b + 1'b1; m = P_{256}$;
5: end if
6: $\{c_0, s_0\} = a + b$;
7: $\{c_1, s_1\} = \{c_0, s_0\} + m$;
8: if $((SEL ==' ADD')\&\&(c_0 == 1))||((SEL ==' SUB')\&\&(c_0 == 0))$ then
9:     $out = s_1$;
10: else
11:     $out = s_0$;
12: end if
13: return $out$;
```

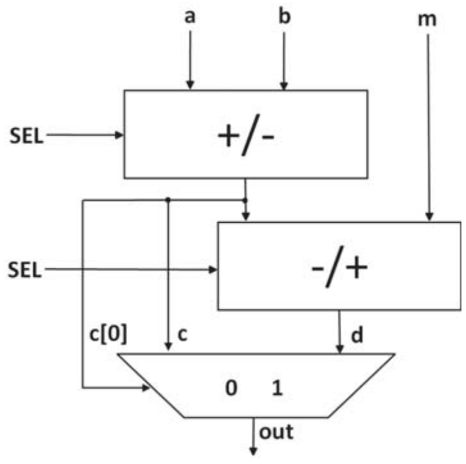


Fig. 3 Modular Add/Sub Structure

D. Modular Inversion and Division

Generally, the algorithms of modular inverse on the prime field include extended Euclid, Fermat's Little Theorem, and the Montgomery algorithm. The extended Euclid algorithm adopts the division method, which can be changed into addition and subtraction operation and binary shift according to the property of greatest common divisor (GCD) so as to be able to be implemented efficiently on the hardware. In addition, there is a more efficient extended Euclid algorithm [16] based on the Radix-4 binary GCD algorithm. Its implementation can be realized by bitwise shifting and modular operations. The upper limit of its operation steps is $2k$, and the maximum delay corresponding to each step corresponds to a $(k + 4)$ bit full adder delay. Therefore, the upper limit of its total computing time is $2k(k + 4)$, and its time complexity is $O((log(p))^2)$. Compared to the extended Euclidean algorithm with a time complexity of $O(p(log(p))^2)$, the Radix-4 binary GCD modular division algorithm is more efficient.

III. Scalar Point Operation Module

All scalar point operations are based on points on elliptic curves. A non-super singular elliptic curve over $GF(p)$ for $p > 3$, and the Weierstrass equation is defined as $y^2 = x^3 + ax + b$, and the SM2 is defined in a particular field of prime numbers, where a=-3.

The formula for adding the same points and points on an elliptic curve differs. They are called points double (PD) and point addition (PA). In addition, all operations in the formula are operations based on finite fields. The scalar point multiplication (PM) consists of repeated PD and PA and can be defined as follows, as shown in (9):

$$kP = \sum_{i=1}^{k} P = P + P + P + ... + P \tag{9}$$

A. Optimized Point Add Scheduling

Through analysis, we can find that the calculation of PA and PD involves modular inversion operations, which is tedious. Therefore, we can avoid modular inverse operation through coordinate system transformation. The PA and PD formulas in mixed affine-Jacobian coordinates are shown in (10) and (11). However, at the end of the calculation, an extra coordinate system transformation operation is needed.

$$\begin{aligned} X_3 =&(Y_2Z_1^3 - Y_1)^2 - (X_2Z_1^2 - X_1)^2(X_1 + X_2Z_1^2) \\ Y_3 =&(Y_2Z_1^3 - Y_1)(X_1(X_2Z_1^2 - X_1)^2 - X_3) \\ &- Y_1(X_2Z_1^2 - X_1)^2 \\ Z_3 =&(X_2Z_1^2 - X_1)Z_1 \end{aligned} \tag{10}$$

According to (10) and (11), PD and PA operations in the Jacobian projective coordinate system can be converted into a series of finite field operations without modular inversions. From the previous analysis in Section I, we know that the majority of the operation time is from the call to the modulo multiplication operation. By reusing the same parameters, (10) can at least be divided into 12 modular multiplication operations and several modular addition operations, among which modular addition operations can be implemented in parallel when calculating modular multiplication operations. Therefore, we will focus on the time-consuming of modular multiplication.

In order to realize PA and PD operation more efficiently, a variety of parallel architectures of point operation is proposed to reduce the running time. Point-operating-level parallel architectures deploy multiple modular operational units to perform computations in parallel, which have no data correlation with each other. Depending on the number of deployed modular computing units, the computational design of the parallel architecture is different. In this paper, we accelerate the point addition and double point operation by synthesizing three modular multiplication units.

Based on the hardware architecture of three-level parallel, according to the correlation between its data, it can be divided into 5 rounds by decomposing the 12 modular operations of PA. The Point Add Scheduling is shown in Table II. Where the $A = X_2Z_1^2 - X_1$, $B = Y_2Z_1^3 - Y_1$.

Based on the parallel design, the PA calculation time is optimized from 12*MM to 5*MM. Besides, combined with PM algorithm analysis, it can be found that in most PM algorithms, PA operations are generally inserted between PD operations, such as binary shift scalar multiplication algorithm, etc. The parallel design of PD is shown in Table IV. Obviously, the point-addition operation in the Jacobi projective coordinate system is more complex than the point-double operation, and the point-addition time is 5*MM, while the point-double time is 4*MM. This leads to the idle hardware unit in the computing process, resulting in a waste of resources. Therefore, we adopt an optimized Point Add Scheduling, as shown in Table III.

TABLE II
TRADITIONAL POINT ADD SCHEDULING

OP rand	Unit1	Unit2	Unit3
1	Z_1^2	Z_1Y_2	NONE
2	$Y_2Z_1^3$	$X_2Z_1^2$	NONE
3	AZ_1	A^2	$A(X_2 + X_3Z_2^2)$
4	X_1A^2	B^2	A^3
5	$B(X_1A^2 - X_2)$	Y_1A^3	NONE

TABLE III
OPTIMIZED POINT ADD SCHEDULING

OP rand	Unit1	Unit2	Unit3
if(pre_PD) skip 1	Z_1^2	Z_1Y_2	NONE
2	$Y_2Z_1^3$	$X_2Z_1^2$	NONE
3	AZ_1	A^2	$A(X_2 + X_3Z_2^2)$
4	X_1A^2	B^2	A^3
5	$B(X_1A^2 - X_2)$	Y_1A^3	Z_3^2

In Table II, the first round of MM operation in the original scheduling is put into the calculation of the last round of PD operation. Before the first round of the point-add operation, we determine whether the point-add operation follows the PD operation. If so, we skip the first round of the MM operation. In this way, the operation time of PA operation can be shortened from the previous 5*MM to 4*MM, effectively improving the operation speed of PA. In the last round of modular multiplication of PA, the pre-operation is introduced to optimize the double operation.

B. Optimized Proposed Point Double Scheduling

$$\begin{aligned} X_3 =&(3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \\ Y_3 =&(3X_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\ Z_3 =&2Y_1Z_1 \end{aligned} \tag{11}$$

In Jacobi affine coordinates, unlike the PA operation, the multiple-point operation has an additional parameter a=-3, which is based on the SM2 parameter. By replacing a with (-3), we can express (11) as (12):

$$(3X_1^2 - 3Z_1^4) = 3(X_1 - Z_1^2)(X_1 + Z_1^2) \tag{12}$$

Thus, the original 3 modular multiplication calculation is converted into two modular multiplication calculations. Thus, the whole multipoint operation can be divided into 8 modular multiplication operations. Similar to the previous parallel scheduling design of PA operation, based on the three-level parallel hardware architecture, PD's 8 modular operations can be decomposed. According to its data independence, PD can

TABLE IV
TRADITIONAL POINT DOUBLE SCHEDULING

OP rand	Unit1	Unit2	Unit3
1	Z_1^2	NONE	NONE
2	$(X_1+Z_1^2)(X_1-Z_1^2)$	Y_1^2	Y_1Z_1
3	$(3X_1^2-3Z_1^4)^2$	$2X_1Y_1^2$	$4Y_1^4$
4	$(3X_1^2-3Z_1^4)(4X_1Y_1^2-X_2)$	NONE	NONE

TABLE V
OPTIMIZED POINT DOUBLE SCHEDULING

OP rand	Unit1	Unit2	Unit3
if(pre_PA) skip 1	Z_1^2	NONE	NONE
2	$(X_1+Z_1^2)(X_1-Z_1^2)$	Y_1^2	Y_1Z_1
3	$(3X_1^2-3Z_1^4)^2$	$2X_1Y_1^2$	$4Y_1^4$
4	$(3X_1^2-3Z_1^4)(4X_1Y_1^2-X_2)$	Z_3^2	Z_3Y_2

be divided into 4 rounds of operation implementation, as shown in Table IV.

Based on this parallel method, the precalculation of PA operation and the judgment operation before the first round of calculation are introduced. We suppose the PD operation follows the PA operation. In that case, the first modular multiplication operation is skipped, as shown in Table V. Thus, the calculation delay of the PD operation is reduced from 4*MM to 3*MM. The speed of PD operation is improved effectively.

C. Coordinate System Transform

After we have done the calculation based on the elliptic curve, the result is based on the Jacobi projective coordinate system, so we need to convert it back to the affine coordinate system. The relationship between its Jacobian projective coordinates and affine coordinates can be described as (13). The operations are shown in Table VI.

$$X = X_J Z_J^{-2}, Y = Y_J Z_J^{-3} \tag{13}$$

D. Point Multiplication Module

The Montgomery point multiplication algorithm is the most efficient and widely used among the existing scalar multiplication algorithms. However, the Montgomery algorithm needs to realize the parallel of the point addition unit and point double unit, resulting in twice the resource consumption. Moreover, according to the Montgomery PM algorithm, PA and PD operations in the Jacobiaffine coordinate system need to introduce additional Z2 parameters. As a

TABLE VI
COORDINATE SYSTEM TRANSFORM

OP rand	Unit1	Unit2	Unit3
1	Z_3^2		
2	Z_3^3		
3	$INVZ_3^2$		
4	$INVZ_3^3$		
5	$X_3(INVZ_3^2)$	$Y_3(INVZ_3^3)$	

TABLE VII
CLOCK NUMBER AND DELAY TIME OF EACH UNIT

Operation	Cycles	Time per operation
MA/MS	1	20.1ns
MM	10	0.201us
MI	624	12.6us
NAF	120	2.42us
PA	50	1.01us
PD	40	0.81us
PM	13662	0.275ms

result, the complexity of the original PA and PD operations is greatly increased. So, we adopted the NAF algorithm [17] to implement a Point Multiplication Module.

IV. VERIFICATION AND HARDWARE IMPLEMENTATION RESULTS

The ECC processor architecture is described using VHDL-Verilog language. This architecture was verified and implemented on Xilinx FPGA boards Virtex-7 based on the Vivado 2020.2 EDA tool.

A. Analysis of Required Clock Cycles

A scalar multiplication operation can be decomposed into repeated calls of point addition and multiple points in the Jacobi projective coordinate system and two inversion operations required for coordinate system transformation. According to the NAF algorithm, the time of a PM can be expressed as (14):

$$\begin{aligned} & N+(L-1)D+(L/(r+1)-1)A+2I \\ =& D+(2^{r-1}-1)A+(D-1)D+(L/(r+1)-1)A+2I \\ =& LD+(2^{r-1}+L/(r+1)-2)A+2I \end{aligned} \tag{14}$$

where N is the clock cycle for calculating $NAF(k)$, D and A are the clock cycle to perform PD and PA operations, r is the width of NAF, l is the length of scalar k, and I is the clock cycle required for modular inversion.

The clock number of each unit and the time delay corresponding to a single operation are shown in Table VII. Because MI calculations are involved, the number of clocks required for the PM calculation results from multiple tests.

B. FPGA-Based Implementation

The ECC processor structure is realized on Virtex-7 by multiplexing three MM and MA/MS unit groups. The resource occupation of each module is shown in Table VIII.

According to Table VIII, no DSP resources are used in the whole implementation process of the ECC processor. This may lead to poor computing performance compared with the DSP design, but it has good portability and can be implemented on other platforms, such as ASIC.

The performance results on the FPGA platform and the comparison with other designs are shown in Table IX. Compared with [18], it adopts Montgomery modular

TABLE VIII
AREA OF EACH MODULE

module	LUT	REG	DSP
NAF	2352	1620	0
MA/MS	1205	612	0
Multiplior	20170	3289	0
Reduction	2218	817	0
Radix-4 Inversion	8625	1570	0

TABLE IX
COMPARISON WITH OTHER DESIGNS

Paper	Ours	[18]	[19]	[20]	[21]
FPGA	Virtex-7	Virtex-7	Virtex-7	Virtex-6	Virtex-6
Field	P_{256}	P_{256}	F_{256}	F_{256}	P_{256}
Freq(MHz)	49.6	244	124.2	327	38.045
Clock(K)	13.66	148.26	464.1	153.16	14.24
Area LUTs	76241	16982	6400(slices)	65600	27655
Area DSP	0	32	0	0	0
PM time(ms)	0.275	0.608	3.73	0.47	0.37

multiplication, which has resource consumption and frequency advantages. However, the high cycle delay brought by the serial Montgomery algorithm leads to its overall computing cycle being much higher than our design. Compared with [19], it is implemented based on the NIST-256 prime number field. Although it is much less than ours in resource consumption, its computing speed is also much higher. Compared with [20], although it is also implemented in a multilevel parallel way, the computing speed is slower than ours compared with its frequency and computing cycle. Compared with [21], we consume more resources to realize parallel and pipeline insertion to improve the frequency, which has advantages in the final operation cycle and frequency.

Although the proposed design consumes more area, it brings shorter running cycles and faster running speeds through multiplier optimization and three-level parallelism.

C. Comparison with CPU

We tested the application effect of the proposed ECC processor structure on an embedded platform. This structure is implemented on the Xilinx ZYNQ-7 platform. Its highest frequency is similar to that of virtex-7 and can complete 3636 PM operations in one second. In the PS (processing system) of zynq, we implemented PM test on the CPU of embedded devices based on openSSL1.1.1, as shown in Table X. Cpu-based (dual-core ARM Cortex-A9) can only perform 112 PM calculations per second. As shown in this test, more than 32 times speed improvements can be achieved by using the proposed ECC processor architecture.

TABLE X
COMPARISON WITH CPU

Divice	1/PM	Speed
CPU(dual-core ARM Cortex-9)	112.8	1
Proposed ECC structure	3636.7	32.24

V. CONCLUSION

This paper proposed a design of an elliptic curve cryptographic processor based on SM2 over GF(P). This paper optimized large number multipliers in finite field MM operations and proposed a structure of full word multipliers based on the KO algorithm. The (m+1) bit multiplication is avoided, and the adder is optimized. The 256-bit multiplication is realized in 6 cycles by pipeline structure and multiplexing three m-bit width multipliers. In addition, a three-level parallel PA and PD structure is proposed. By optimizing the parallel scheduling instruction and adding the pre-operation method, the operation complexity of PA and PD is reduced to 3*MM and 4*MM, respectively. The implementation results show that the proposed ECC processor structure can achieve 0.275ms per PM. For PM computing. The ECC processor improves performance by more than 32 times compared to the CPU (dual-core ARM Cortex-A9).

ACKNOWLEDGMENT

This work was supported by Stability Program of Science and Technology on Communication Security Laboratory (2022) and Sichuan Science and Technology Program (No.2021YJ0082).

REFERENCES

[1] Victor S Miller. Use of elliptic curves in cryptography. In Advances in CryptologyłCRYPTO85 Proceedings, pages 417– 426. Springer, 1985.
[2] Neal Koblitz. Elliptic curve cryptosystems. Mathematics of computation, 48(177):203–209, 1987.
[3] Yang A, Nam J, Kim M, et al. Provably-secure (Chinese government) SM2 and simplified SM2 key exchange protocols[J]. The Scientific World Journal, 2014, 2014.
[4] Zhang Y, He D, Zhang M, et al. A provable-secure and practical two-party distributed signing protocol for SM2 signature algorithm[J]. Frontiers of Computer Science, 2020, 14: 1-14.
[5] Zhou L, Su C, Hu Z, et al. Lightweight implementations of NIST P-256 and SM2 ECC on 8-bit resource-constraint embedded device[J]. ACM Transactions on Embedded Computing Systems (TECS), 2019, 18(3): 1-13.
[6] Liu Z, Liu D, Zou X. An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor[J]. IEEE Transactions on Industrial Electronics, 2016, 64(3): 2353-2362.
[7] Roy D B, Mukhopadhyay D. High-speed implementation of ECC scalar multiplication in GF (p) for generic Montgomery curves[J]. IEEE transactions on very large scale integration (VLSI) systems, 2019, 27(7): 1587-1600.
[8] Seo S C, Kim T, Hong S. Accelerating elliptic curve scalar multiplication over GF (2m) on graphic hardwares[J]. Journal of Parallel and Distributed Computing, 2015, 75: 152-167.
[9] Salman A, Ferozpuri A, Homsirikamol E, et al. A scalable ECC processor implementation for high-speed and lightweight with side-channel countermeasures[C]//2017 international conference on ReConFigurable Computing and FPGAs (ReConFig). IEEE, 2017: 1-8.
[10] Awaludin A M, Larasati H T, Kim H. High-speed and unified ECC processor for generic Weierstrass curves over GF (p) on FPGA[J]. Sensors, 2021, 21(4): 1451.
[11] Ding J, Li S. A reconfigurable high-speed ECC processor over NIST primes[C]//2017 IEEE Trustcom/BigDataSE/ICESS. IEEE, 2017: 1064-1069.
[12] Ding J, Li S, Gu Z. High-speed ECC processor over NIST prime fields applied with Toom–Cook multiplication[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2018, 66(3): 1003-1016.
[13] Khan S, Javeed K, Shah Y A. High-speed FPGA implementation of full-word Montgomery multiplier for ECC applications[J]. Microprocessors and Microsystems, 2018, 62: 91-101.

[14] Feng X, Li S. A high performance FPGA implementation of 256-bit elliptic curve cryptography processor over GF (p)[J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2015, 98(3): 863-869.
[15] Zhang D, Bai G. High-performance implementation of SM2 based on FPGA[C]//2016 8th IEEE International Conference on Communication Software and Networks (ICCSN). IEEE, 2016: 718-722.
[16] Deschamps J P, Sutter G. Hardware implementation of finite-field division[J]. Acta Applicandae Mathematica, 2006, 93: 119-147.
[17] Anagreh M, Vainikko E, Laud P. Accelerate Performance for Elliptic Curve Scalar Multiplication based on NAF by Parallel Computing[C]//ICISSP. 2019: 238-245.
[18] Wu T, Ye J, Lu J. Hardware implementation of SM2 ECC protocols on FPGAs[C]//2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, 2021, 5: 33-37.
[19] Kudithi T, Sakthivel R. High-performance ECC processor architecture design for IoT security applications[J]. The Journal of Supercomputing, 2019, 75(1): 447-474.
[20] Shah Y A, Javeed K, Azmat S, et al. Redundant-signed-digit-based high speed elliptic curve cryptographic processor[J]. Journal of Circuits, Systems and Computers, 2019, 28(05): 1950081.
[21] Hu X, Zheng X, Zhang S, et al. A high-performance elliptic curve cryptographic processor of SM2 over GF (p)[J]. Electronics, 2019, 8(4): 431.

Shiji Hu received the bachelor of Microelectronics Science and Engineering from Chongqing University of Posts and Telecommunications in 2021. He is currently a master’s student at the University of Electronic Science and Technology of China. His research interest covers integrated circuit design and hardware safety.

Lei li received Ph.D. degree in communication engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2010. He is currently an associate researcher with University of Electronic Science and Technology of China. His current research interests include integrated circuits, machine learning, and hardware security

Wanting Zhou received the Ph.D. degree in communication and information systems from University of Electronic Science and Technology of China, Chengdu, China, in 2014. She is currently a research associate with University of Electronic Science and Technology of China. Her research interests include integrated circuits, machine Learning and hardware security

Daohong Yang received the bachelor of communication engineering from Donghua university in 2021. He is currently a master’s student at the University of Electronic Science and Technology of China. His research interest covers integrated circuit design and hardware safety.