Hardware Implementation of SM9 Fast Algorithm Based on FPGA

Shuai Jing[1,a], XiaoTing Yang[2,b], YeJi Feng[3,c], XiaoDong Liu[4,d], FuQing Hao[5,e] and ZiHeng Yang[6,f*]

[123456]*School of Electronic Engineering, Heilongjiang University, Nangang District, Harbin City, Heilongjiang Province, China*

[a]*15315778557@163.com*, [b]*1217077150@qq.com*, [c]*3288618181@qq.com*, [d]*1354342202@qq.com*, [e]*1824534850@qq.com*, [*f]*yzh@hlju.edu.cn*

ABSTRACT

SM9 algorithm plays a very important role in the field of information security in our country, it can effectively solve the problem of certificate management of PKI. The dot multiplication operation in the SM9 algorithm takes up a lot of computing time, which seriously affects the operation efficiency. The traditional modular multiplication algorithm uses a simple shift and addition method. To shorten the time consumed by the operation, this paper proposes an improved modular multiplication algorithm that supports a four-stage pipeline. In addition, this paper also designs and improves algorithms such as point addition, point doubling, and point multiplication. Through FPGA simulation and verification, it only takes 0.848ms to realize one point multiplication operation, and the occupied resources are 18526 look-up tables (LUTs) and 13982 flip-flops (FFs). This research has great significance for the development of SM9, making it have wider application value.

Keywords: *FPGA, Dot multiplication, SM9, Identification encryption algorithm*

1. INTRODUCTION

The Israeli cryptographer Shamir proposed identity-based cryptography in 1984 [8], namely IBC (Identity-Based Cryptography). The identity cryptographic algorithm does not need to generate a public-private key pair through a third party (such as a CA) center to ensure the security of the key, nor does it need to use a certificate to transmit the public key, but use user identifiers such as name, IP address, email address The identification information representing the user such as address and mobile phone number is used as the public key, and the key center (Key Generate Center, KGC for short) calculates the private key according to the system master key and the user ID. This simplification saves the expenditure of traditional public-key cryptosystems in key generation, certificate issuance, key management, etc.

The SM9 identification cryptographic algorithm [10] is the national cryptographic industry-standard (GM/T 0044-2016) promulgated by the State Cryptography Administration of my country in March 2016. The algorithm inherits many advantages in the identification cryptography system, greatly simplifies the complexity of key management in the traditional certificate system and is easy to manage and use. However, the mathematical calculation involved in the algorithm is relatively complex, and the implementation performance is low, which affects the popularization and use of the cryptographic algorithm. Therefore, effectively improving the computing performance of the SM9 cryptographic algorithm is the focus of research in recent years.

The main factors that affect the performance of the SM9 algorithm are the point multiplication operation and the bilinear pairing operation. Dot multiplication operations include modular addition and subtraction, modular multiplication, modular inversion, point addition, and multiplication. Each point multiplication operation requires a lot of time. This paper mainly improves the operation speed of point multiplication by improving the above algorithms. In the literature [4][11], a point multiplication operation method combining software and hardware is proposed, the modular

A. El-Hashash et al. (Eds.): IEIT 2022, ACSR 100, pp. 797–803, 2023.
https://doi.org/10.2991/978-94-6463-058-9_125

multiplication unit is implemented in hardware, and the point multiplication operation is implemented in software, but the software and hardware need to constantly exchange information during the operation, and waste a lot of time on information transmission. In the literature [9], the point multiplication operation is implemented in pure hardware, and the performance is improved by improving the modular multiplication operation. For 256-bit point multiplication, it can be operated 94 times per second, that is, it takes 10.4ms to calculate one point multiplication. Literature [1][6] designed a high-speed modular multiplication algorithm, which improved the computing performance by rationally arranging the modules and optimizing the algorithm. The literature [3] is aimed at improving the modular inverse algorithm and reducing the modular inverse operation time by optimizing the algorithm. The above methods are all algorithm improvements for modular multiplication or modular inversion alone, and the final point multiplication algorithm is still inefficient. In this paper, the modular multiplication, modular inverse, point addition, point multiplication, and point multiplication algorithms have undergone a series of improvements, and the operation time is shortened by optimizing the algorithm and using parallel computing skills. Finally, the point multiplication operation achieves the best performance.

2. INTRODUCTION TO SM9 ALGORITHM

The operation of the National Secret SM9 is based on the elliptic curve algorithm, and the most complicated content in the algorithm process is the point multiplication operation and the bilinear pairing operation. The SM9 algorithm can be divided into five levels, and there is a calling relationship between each level. The highest layer is the protocol layer including digital signature and key exchange, followed by bilinear pairing, followed by the group operation layer including point multiplication operation, followed by the point operation layer including point addition and doubling point operation, and finally the finite field modulo operation layer. Including modular addition and subtraction, modular multiplication and modular inverse operations. This paper proposes an optimization scheme for the point multiplication algorithm, which improves the overall performance of SM9 through efficient parallel scheduling. The structure of the SM9 algorithm is shown in Figure 1.

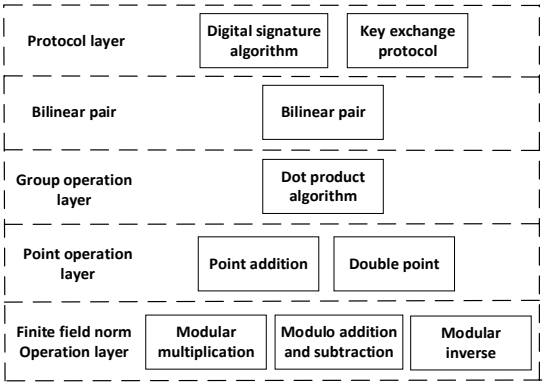

Figure 1: This caption has one line so it is centered.

2.1. Modular Multiplication Algorithm

In the elliptic curve algorithm, the modular multiplication operation is one of the most critical basic operations. Multiplication requires more resources and time than addition, so try to reduce the use of multiplication as much as possible. The traditional modular multiplication algorithm uses a low-cost shift-add method instead of multiplication. This algorithm has less complexity and is very easy to implement in hardware.

Algorithm 1: Modular multiplication of traditional shift-add

Input: a,b,p ,0⩽a,b<p
Output: c=a*b mod p
1. initialization c=0
2. for i from n-1 to 0, repeat the execution
2.1 c=2c+a_i*b
2.2 if c⩾p then c=c-p
3. **return** c

The improved parallel modular multiplication algorithm supports a four-stage pipeline, in which steps 2.1, 2.2, 2.3 and 2.4 are parallel operations. This method divides the multiplier equally into four segments of data, and judges the value of ai in each segment. Taking the first paragraph as an example, the value of ai1 is judged from left to right. If it is 0, this bit can be skipped until it encounters 1, so that b is shifted to the left by the i1 bit and assigned to b1. Such an improved method can make the shift operations of the four segments of data to be performed simultaneously, fully utilize the advantages of parallel operations, and greatly improve the operation speed of modular multiplication. The multiplication operation is replaced by the shift operation, which simplifies the operation process and occupies fewer resources. Finally, the simulation test shows that the calculation time of the improved parallel modular multiplication algorithm is shortened by about 75% compared with the traditional algorithm.

Algorithm 2: Improved Parallel Modular Multiplication

Input:a,b,p ,0≤a,b<p
Output:c=a*b mod p
1. initialization c=0
2. for j from (n/4)-1 to 0, repeat the execution
2.1. repeat for i_1 from n-1 to 3n/4
if a_{i1}=1 then b_1=b<<i_1
else then b_1=0
2.2. repeat for i_2 from (3n/4)-1 to n/2
if a_{i2}=1 then b_2=b<<i_2
else then b_2=0
2.3. repeat for i_3 from (n/2)-1 to n/4
if a_{i3}=1 then b_3=b<<i_3
else then b_3=0
2.4. repeat for i_4 from (n/4)-1 to 0
if a_{i4}=1 then b_4=b<<i_4
else then b_4=0
2.5. c=b_1+b_2+b_3+b_4+c
2.6. if c≥p then c=c-p
3. **return c**

2.2. *Modular inverse algorithm*

Common inversion methods are based on Euclidean calculations or Fermat's little theorem. Among them, Fermat's little theorem is suitable for the modular inverse operation of the binary extended field, but not suitable for the prime number field. The extended Euclidean algorithm completes the modulo inverse operation by turning and dividing and can replace the division with a binary shift. This method occupies fewer hardware resources and is convenient for hardware implementation. Based on the above analysis, this paper adopts the extended Euclidean algorithm.

Algorithm 3: Improved Euclidean Modular Inverse Algorithm

Input: prime p, integer a, where a belongs to (0,p)
Output: a^{-1} mod p
1. u=a, v=p, x_1=1, x_2=0
2. repeat if u≠0
2.1 if u is even, repeat the execution
2.1.1 u=u/2
2.1.2 if x_1 is even, then x_1=x_1/2
2.1.3 else x_1=(x_1+p)/2
2.2 repeat if v is even
2.2.1 v=v/2
2.2.2 if x_2 is even, then x_2=x_2/2
2.2.3 else x_2=(x_2+p)/2
2.3 if u≥v, then u=u-v.
2.3.1 if x_1≥x_2, then x_1=x_1-x_2
2.3.2 else x_1=x_1-x_2+p
2.4 if u<v, then v=v-u.
2.4.1 if x_2≥x_1, then x_2=x_2-x_1
2.4.2 else x_2=x_2-x_1+p
3. if u=1, a^{-1}mod p=x_1 mod p;
else a^{-1}mod p=x_2 mod p

The Euclidean inversion algorithm is reflected in improvements in hardware design. The division in steps 2.1.2 and 2.2.2 can be replaced by a binary right shift, which is convenient for hardware implementation. Since the variables in 2.1 and 2.2 do not affect each other, when writing the state machine, design steps 2.1 and 2.2 in the same state at the same time, and then jump after the next clock arrives, so that parallel operations can be realized and the operation speed can be improved. . Compared with the traditional modular inverse operation, the improved modular inverse operation shortens the time by about 22%.

2.3. *Point addition algorithm*

Point addition and point doubling algorithms belong to the point operation layer and are implemented by calling modulo addition/subtraction, modulo multiplication, and modulo inverse. The algorithm rule of point addition in the elliptic curve coordinate system is: set $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, the result is $P_3(x_3,y_3)=P_1+P_2$. Where $x_3=k_2-x_1-x_2$, $y_3=k(x_1-x_3)-y_1$, $k=(y_2-y_1)/(x_2-x_1)$. It can be seen that the point addition algorithm includes 6 modular additions and subtractions, 2 modular multiplications and 1 modular inverse algorithm.

Algorithm 4: Design of point addition algorithm

Input: $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, p
Output: $P_3(x_3,y_3)$
1. t_1=(x_2-x_1)mod p
2. t_2=(x_2+x_1)mod p
3. t_3=(y_2-y_1)mod p
4. t_1=(t_3/t_1)mod p
5. t_3=(t_1*t_1)mod p
6. t_3=(t_3-t_2)mod p
7. t_2=(x_1-t_3)mod p
8. t_1=(t_1*t_2)mod p
9. t_1=(t_1-y_1)mod p
10. **Return** x_3=t_3, y_3=t_1

Improvement in the hardware design of the point-add algorithm: Since the operations of steps 1, 2, and 3 do not affect each other, operations can be performed simultaneously to shorten the time. According to ordinary logic, seven registers should be used, but only three registers are needed after the improvement. This method reduces the use of resources. The improved point addition algorithm requires 1us for one operation, and the hardware occupies 9512 LUTs and 6454 FFs. Compared with the traditional operation method, the improved point-adding algorithm shortens the time by about 80%, and reduces the occupied resources by about 23%.

2.4. *Point doubling algorithm*

The rule of the point doubling algorithm is: set $P_1(x_1,y_1)$, the result is $P_3(x_3,y_3)=P_1+P_1$. Where $x_3=k_2-2x_1$, $y_3=k(x_1-x_3)-y_1$, $k=(3x_1^2+a)/2y_1$. It can be seen that the

point doubling algorithm includes 4 modular additions/subtractions, 6 modular multiplications and 1 modular inverse algorithm.

Algorithm 5: Design of point doubling algorithm

Input: $P_1(x_1,y_1)$, $P_2(x_2,y_2)$, p, a
Output: $P_3(x_3,y_3)$
1. $t_1=(x_1*x_1)\bmod p$
2. $t_1=(t_1+t_1+t_1)\bmod p$
3. $t_2=(y_1+y_1)\bmod p$
4. $t_1=(t_1+a)\bmod p$
5. $t_1=(t_1/t_2)\bmod p$
6. $t_2=(t_1*t_1)\bmod p$
7. $t_2=(t_2-x_1)\bmod p$
8. $t_2=(t_2-x_1)\bmod p$
9. $t_3=(x_1-t_2)\bmod p$
10. $t_1=(t_1*t_3)\bmod p$
11. $t_1=(t_1-y_1)\bmod p$
12. **Return** $x_3=t_2$, $y_3=t_1$

Improvements in the hardware design of the point doubling algorithm: the calculation time of modular multiplication is much greater than the calculation time of modular addition and subtraction, so the modular multiplication algorithm in steps 2, 3, 7, and 8 is changed to the modular addition/subtraction algorithm to shorten the operation time. In addition, steps 2 and 3 do not affect each other, and parallel operations can be used. The improved doubling operation requires 8 modulo additions and subtractions, 3 modulo multiplications, and 1 modulo inverse algorithm. According to ordinary logic, nine registers should be used, and only three registers are needed after the improvement. This method reduces the use of resources. The improved point doubling algorithm requires 2.2us for one operation, and the hardware occupies 9330 LUTs and 6454 FFs. Compared with the traditional computing method, the improved multi-point algorithm shortens the time by about 77%, and reduces the resource consumption by about 30%.

2.5. Point multiplication algorithm

According to the nature of the elliptic curve operation, the point multiplication operation is the repeated addition of the same point on the elliptic curve. The dot multiplication algorithm is implemented by calling the dot addition and dot doubling operations multiple times. The common dot multiplication algorithm is the left-to-right binary dot multiplication algorithm.

Algorithm 6: Left-to-right binary dot multiplication algorithm

Input: P, t-bit integer $u=\sum_{j=0}^{t-1} u_j 2^j$, $u_j \in \{0,1\}$
Output: Q=[u]P
1. Q=0
2. Repeat for j from t-1 to 0
2.1 Q=[2]Q
2.2 If $u_j=1$, then Q=Q+P
3. **Return** Q

Algorithm 7: Improved dot multiplication

Input: P, t-bit integer $u=\sum_{j=0}^{t-1} u_j 2^j$, $u_j \in \{0,1\}$
Output: Q=[u]P
1. Repeat for j from t-1 to m
if uj=1, then Q=P, m=j;
else Q=0
2. Repeat for j from m-1 to 0
if $u_j=1$, then Q=[2]Q, Q=Q+P;
else Q=[2]Q
3. **Return** Q

The improved dot multiplication algorithm judges the value of the u_j from left to right. When the u_j is 0, this bit is skipped and the next bit is judged. This reduces t-m cycles, t-m left shifts and t-m addition operations compared to traditional dot multiplication. Such an improved method reduces a lot of time and takes up fewer resources.

3. EXPERIMENTAL RESULTS

Some improved algorithms proposed in this paper are tested on the ZYNQ development board of Xilinx Company, and the algorithms are completed by Verilog-HDL hardware description language. Figure 2 is a simulation diagram of the dot product algorithm. It can be seen from the figure that the clock frequency used is 200MHz, and it takes 0.848ms to operate a dot product. Table 1 shows the operation time and occupied resources of each algorithm after improvement.

Figure 2: Simulation diagram of dot product algorithm

Table 1: Algorithm operation time and occupied resources.

	Time(us)	LUT	FF
Modular multiplication	0.32	1820	782
Modular inverse	0.9	3922	1542
Point addition	1	9512	6454
Point doubling	2.2	9330	6454
Dot multiplication	848	18526	13982

The improved dot multiplication algorithm takes 0.848ms to operate once, and occupies 18526 LUTs and 13982 FFs. Compared with other literature, the dot product operation in this paper is much faster. Table 2 is a comprehensive comparison of the dot product algorithm in this paper and other literature. Since the clock frequencies used in each document are different, a unified clock frequency is required for comparison. Figure 2 shows the comparison of the point multiplication operation time with a clock of 200MHz.

Table 2: The comparison of the point multiplication algorithm.

Literature	Clock frequency(MHz)	dot multiplication time(ms)
(Gao 2021)	150	4.24
(Marzouqi 2016)	160	2.26
(Loi 2015)	251	3.95
This article	200	0.848

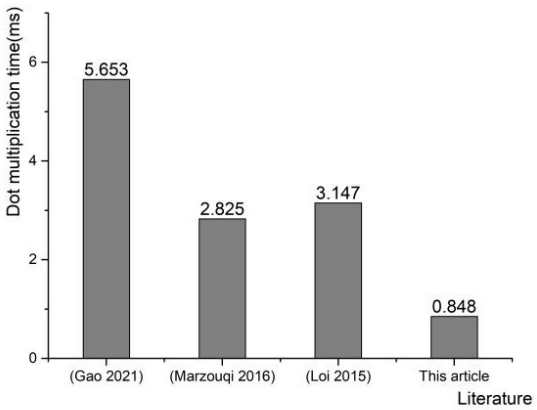

Figure 3. Comparison of the operation time of point multiplication

4. CONCLUSIONS

In order to improve the efficiency of point multiplication in the SM9 algorithm, this paper designs a modular multiplication algorithm that supports a four-stage pipeline. In this technology, this paper designs a parallel architecture and optimizes other underlying algorithms, thereby improving the operation speed of point multiplication. After a series of improvements, the efficiency of dot multiplication is greatly improved. Through the comparative analysis with related literature, the point multiplication operation in this paper is faster, takes less time, and is more suitable for hardware implementation. This research plays an important role in the application and development of SM9.

REFERENCES

[1] Guo Xiao, Jiang Anping, Zong Yu. Hardware Design of SM2 High Speed Dual Domain Montgomery Modular Multiplication [J]. Microelectronics and Computers, 2013, 30(9): 17-21.

[2] Gao Wei, Luo Yixuan, Li Jiakun, Wu Haixia. High-performance hardware implementation of elliptic curve cryptographic dot product in prime field[J]. Journal of Beijing Institute of Technology, 2021, 41(09): 977-984. DOI: 10.15918/j.tbit1001-0645.2020 .216.

[3] Hu Jin, Li Yongbin. An Improved Modular Inverse Algorithm and Hardware Implementation [J]. Journal of Hunan University (Natural Science Edition), 2022, 49(02): 101-105. DOI: 10.16339/j.cnki.hdxbzkb.2022264 .

[4] Janssens S , Thomas J , Borremans W , et al. Hardware/software Co-Design Of An Elliptic Curve Public-Key Cryptosystem[C]// Signal Processing Systems, 2001 IEEE Workshop on. IEEE, 2001.

[5] Loi K , Ko S B . Scalable Elliptic Curve Cryptosystem FPGA Processor for NIST Prime Curves[J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(11):2753-2756.

[6] Li Jiamin, Dai Zibin, Wang Yiwei. Research and Design of Programmable and Scalable Dual Domain Modular Multiplier-Adder [J]. Electronic Technology Application, 2018,44(01):28-32+36.DOI:10.16157/j.issn.0258 -7998.172194.

[7] Marzouqi H , Al-Qutayri M , Salah K , et al. A High-Speed FPGA Implementation of an RSD-Based ECC Processor[J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 24(1):151-164.

[8] Shamir A. Identity-based cryptosystems and signature schemes[C]//Workshop on the theory and application of cryptographic techniques. Springer, Berlin, Heidelberg, 1984: 47-53.

[9] Xie Tianyi, Huang Kai, Xiu Siwen, Tang Congxue, Yan Xiaolang. VLSI Implementation of Prime Field Elliptic Curve Cryptographic Accelerator [J]. Computer Engineering and Applications, 2016,52(01):89-94.

[10] Zhang Q, Wang A, Niu Y, et al. Side-channel attacks and countermeasures for identity-based cryptographic algorithm SM9[J]. Security and communication networks, 2018, 2018.

[11] Zhang Shengshi, Hu Xianghong, Xiong Xiaoming. FPGA Architecture Based on State Secret Algorithm SM2 Software-Hardware Collaborative System [J]. Microcontroller and Embedded System Application, 2019, 19(7): 15-19.