Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An energy-efficient reconfigurable asymmetric modular cryptographic operation unit for RSA and ECC

Mengni BIE[†1], Wei LI[†‡1], Tao CHEN[1], Longmei NAN[2], Danyang YANG[1]

[1]*Information Engineering University, Zhengzhou 450001, China*

[2]*State Key Lab of ASIC and System, Fudan University, Shanghai 200000, China*

[†]E-mail: raspberry0213@126.com; liwei12@fudan.edu.cn

Received July 6, 2020; Revision accepted Dec. 22, 2020; Crosschecked Oct. 12, 2021

Abstract: RSA and ellipse curve cryptography (ECC) algorithms are widely used in authentication, data security, and access control. In this paper, we analyze the basic operation of the ECC and RSA algorithms and optimize their modular multiplication and modular inversion algorithms. We then propose a reconfigurable modular operation architecture, with a mix-memory unit and double multiply-accumulate structures, to realize our unified, asymmetric cryptosystem structure in an operational unit. Synthesized with 55-nm CMOS process, our design runs at 588 MHz and requires only 437 801 μm^2 of hardware resources. Our proposed design takes 21.92 and 23.36 mW for 2048-bit RSA modular multiplication and modular inversion respectively, as well as 16.16 and 15.88 mW to complete 512-bit ECC dual-field modular multiplication and modular inversion respectively. It is more energy-efficient and flexible than existing single algorithm units. Compared with existing multiple algorithm units, our proposed method shows better performance. The operation unit is embedded in a 64-bit RISC-V processor, realizing key generation, encryption and decryption, and digital signature functions of both RSA and ECC. Our proposed design takes 0.224 and 0.153 ms for 256-bit ECC point multiplication in $G(p)$ and $G(2^m)$ respectively, as well as 0.96 ms to complete 1024-bit RSA exponentiation, meeting the demand for high energy efficiency.

Key words: Modular operation unit; Reconfigurable; High energy efficiency
https://doi.org/10.1631/FITEE.2000325 **CLC number:** TP331.2; TN918.2

1 Introduction

Some well-known international communication organizations, such as 5G Infrastructure Public Private Partnership (5G PPP), Next Generation Mobile Networks (NGMN) Alliance, and Global System for Mobile Communications Assembly (GSMA) Intelligence, have published their respective 5G White Papers (GSMA Intelligence, 2014; NGMN, 2015; 5G PPP, 2016), indicating that "5G mobile communication needs to provide rapid response and ubiquitous network access for the development of the Internet of Things, which brings a lot of new security requirements and risks to 5G security." Protecting sensitive information when it is transmitted via insecure communication channels has become essential in this environment. Asymmetric cryptography, such as RSA and elliptic curve cryptography (ECC), has been investigated for this purpose. At present, the most mature and widely used asymmetric cryptography algorithm is RSA. This is simple in principle and is convenient to use. The new public-key cryptography algorithm ECC requires much shorter keys and has wider application prospects. Although the ECC algorithm has been widely used in the actual protocols for small devices, the RSA algorithm is still widely available in the market because of its wider application and greater compatibility. The above two cryptographic algorithms are the main popular

‡ Corresponding author
ORCID: Mengni BIE, https://orcid.org/0000-0003-2446-6692; Wei LI, https://orcid.org/0000-0002-6597-0142

secure socket layer (SSL) protocol encryption algorithms. Consequently, on one hand, the integration of dual-field (prime and binary fields) ECC using any width and RSA into a reconstruction framework would be important. On the other hand, there is a great need to balance resources and performance on resource-limited embedded devices. We have developed a reconfigurable modular operational unit for high energy efficiency, compatible with both RSA and dual-field ECC, supporting their applications in different encryption demand scenarios.

As a basic operation of public key cryptography, modular operation has been the subject of many studies, but it is relatively targeted. For modular multiplication of the RSA algorithm, Kuang et al. (2013, 2016) designed a modular multiplier based on an adder, which achieved better performance within a small area. Miyamoto et al. (2011) proposed three high-radix Montgomery algorithms with three different carry modes, and realized modular multipliers with different algorithms and different operating units. The above studies have completed the encryption and decryption function of the RSA cryptographic algorithm based on the modular multiplier, but the key generation function cannot be realized because of the lack of a modular inversion operation.

For an RSA modular inversion operation, research on the design is scant, and the research into its reconstruction with modular multiplication is even more deficient. This is mainly because the modular operations of RSA are in the even field, making its performance far worse than that of modular multiplication. Xia (2016) proposed a modular inverter suitable for RSA using a special formula. It converts modular inversion on the even field into two multiplications and a modular inversion on the prime field. In this way, parameter E in the key of the RSA algorithm is fixed to a special prime number, which reduces the applicability of the algorithm. The conventional extended Euclidean algorithm is still needed for the converted one-step modular inversion.

For the ECC algorithm, modular inversion is an operation in the binary field or the prime field. The particularity of the field optimizes the performance of the modular inversion algorithm. Therefore, research on the modular operation unit of the ECC algorithm is plentiful. Gu and Li (2019) improved the Montgomery modulator for the prime field and binary field, but it cannot compute the modular inversion. Ibrahim and Gebali (2017) and Li et al. (2019) proposed unified structures for modular multiplication and its inversion operation for the prime field and binary field operations of the ECC algorithm, but their solutions can be applied only to a single scenario. Also, Li et al. (2019)'s approach can be applied only to the 256-bit ECC algorithm and has poor scalability and limited application. Chen GH et al. (2010) proposed a unified structure for ECC dual-field modular multiplication and the inversion operation, but it is suitable only for fixed-width data. Liu et al. (2017) proposed a flexible structure of dual-field modular multiplication and its inversion based on adders, with a small area but long operation time. In summary, many researchers have explored single functional modular operational units that perform well in certain scenarios, but they are unable to combine the operations required by different public key algorithms into one unit.

In this paper, we analyze a combined structure of RSA and ECC using modular operations to perform basic operations on finite fields along with other operations. Our contributions are highlighted here:

1. We propose an optimized dual-field high-radix Montgomery modular multiplication algorithm and dual-field extended Euclidean modular inversion algorithm that are more efficient for RSA and ECC.

2. We propose a reconfigurable public key operation unit, with a memory unit as the center and double multiply-accumulate structures. The unit can complete all of the operations needed by the ECC and RSA public key cryptography algorithms, including modular addition, modular subtraction, signed addition and subtraction, unsigned addition and subtraction, shift, data load, equality, modular multiplication, modular inversion, and other operations. Taking modular multiplication and modular inversion as examples, we discuss how to do efficient computation based on this unit.

3. We design the instruction set of efficient modular operation. This proves that the unit can be applied in the processor, and we realize the key generation, encryption and decryption, and digital signature functions of RSA and ECC by programming.

2 Dual-field high-radix Montgomery modular multiplication algorithm

The classical Montgomery algorithm transforms the modular multiplication of any modulus into a relatively simple modular operation suitable for hardware design. The operations of the algorithm on the prime and binary fields are similar, and they are suitable for our dual-field modular multiplication algorithm. However, this classical algorithm is based on multiplication, but the delay caused by the multiplier increases rapidly with the increasing data width. Consequently, this design will have greatly limited processing speed in an embedded system. Fortunately, a high-radix Montgomery algorithm has been derived from this algorithm to solve this problem. By dividing operations on large-width data into repeated operations on small-width data, this improved algorithm avoids a large critical path delay caused by large data widths. The representative algorithms include finely integrated operand scanning (FIOS) and coarsely integrated operand scanning (CIOS) algorithms (Kaya Koc et al., 1996).

Miyamoto et al. (2011) analyzed the delay distribution in the implementation of the high-radix Montgomery algorithm, and then successfully proposed a high-radix Montgomery algorithm suitable for semi-carry storage of the RSA algorithm. Miyamoto et al.'s algorithm splits the last level of the carry propagate adder (CPA) in the multiplier, separates carry from the results of the current operation, and cuts the operation width of CPA by half, as shown in Algorithm 1. However, this approach does not consider the compatibility with ECC modular multiplication. Although the modular multiplication of the ECC prime field is quite similar to that of RSA, differences exist in the modular multiplication of the binary field. We build on this algorithm, taking the compatibility of the two kinds of finite field operations in ECC into consideration. We simplify the steps for the ECC binary field and propose a dual-field high-radix Montgomery modular multiplication algorithm as shown in Algorithm 2.

Algorithm 1 High-radix Montgomery algorithm suitable for semi-carry storage

Input: $X=(x_{m-1},\dots,x_1,x_0)_{2^r}$, $Y=(y_{m-1},\dots,y_1,y_0)_{2^r}$, $N=(n_{m-1},\dots,n_1,n_0)_{2^r}$, $w=-N^{-1} \bmod 2^r$
Output: $Z=XY\cdot 2^{-rm} \bmod N$
$Z=0$, $v=0$
for i=0 to $m-1$ **do**
  $(\mathrm{cs1}_a+\mathrm{cs2}_a+\mathrm{ec}_a, z_0)=z_0+x_iy_0$
  $t_i=z_0w \bmod 2^r$
  $(\mathrm{cs1}_b+\mathrm{cs2}_b+\mathrm{ec}_b, z_0)=z_0+t_in_0$
  for j=1 to $m-1$ **do**
    $(\mathrm{cs1}_a+\mathrm{cs2}_a+\mathrm{ec}_a, z_j)=z_j+x_iy_j+\mathrm{cs1}_a+\mathrm{cs2}_a+\mathrm{ec}_a$
    $(\mathrm{cs1}_b+\mathrm{cs2}_b+\mathrm{ec}_b, z_{j-1})=z_j+t_in_j+\mathrm{cs1}_b+\mathrm{cs2}_b+\mathrm{ec}_b$
  end for
  $(v, z_{m-1})=\mathrm{cs1}_a+\mathrm{cs2}_a+\mathrm{ec}_a+\mathrm{cs1}_b+\mathrm{cs2}_b+\mathrm{ec}_b+v$
end for
if $Z>N$ **then**
  $Z=Z-N$
end if

Algorithm 2 Dual-field high-radix Montgomery modular multiplication algorithm

Input: $X=(x_{m-1},\dots,x_1,x_0)_{2^r}$, $Y=(y_{m-1},\dots,y_1,y_0)_{2^r}$, $N=(n_{m-1},\dots,n_1,n_0)_{2^r}$, $w=-N^{-1} \bmod 2^r$
Output: $Z=XY\cdot 2^{-rm} \bmod N$
$Z=0$, $v=0$
for i=0 to $m-1$ **do**
  $(\mathrm{cs1}_a\oplus\mathrm{cs2}_a\oplus\mathrm{ec}_a, z_0)=z_0\oplus x_i\otimes y_0$
  // "$\oplus$" means addition and XOR on the prime field
  // and binary field, respectively
  // "$\otimes$" means multiplication on the prime field and
  // binary field
  $t_i=z_0\otimes w \bmod 2^r$
  $(\mathrm{cs1}_b\oplus\mathrm{cs2}_b\oplus\mathrm{ec}_b, z_0)=z_0\oplus t_i\otimes n_0$
  for j=1 to $m-1$ **do**
    $(\mathrm{cs1}_a\oplus\mathrm{cs2}_a\oplus\mathrm{ec}_a, z_j)$
    $=z_j\oplus x_i\otimes y_j\oplus\mathrm{cs1}_a\oplus\mathrm{cs2}_a\oplus\mathrm{ec}_a$
    $(\mathrm{cs1}_b\oplus\mathrm{cs2}_b\oplus\mathrm{ec}_b, z_{j-1})$
    $=z_j\oplus t_i\otimes n_j\oplus\mathrm{cs1}_b\oplus\mathrm{cs2}_b\oplus\mathrm{ec}_b$
  end for
  $(v, z_{m-1})=\mathrm{cs1}_a\oplus\mathrm{cs2}_a\oplus\mathrm{ec}_a\oplus\mathrm{cs1}_b\oplus\mathrm{cs2}_b\oplus\mathrm{ec}_b\oplus v$
end for
if $Z>N$ & field = 1 **then**
  $Z=Z-N$
end if

3 Dual-field extended Euclidean modular inversion algorithm

Existing methods of modular inversion in finite fields incorporate Fermat's little theorem, the extended Euclidean algorithm, or Montgomery's algorithm. Modular inversion algorithms using Fermat's little theorem and Montgomery's algorithm require the modulus to be prime, making them unsuitable for the RSA algorithm when the modulus is even. In contrast, the extended Euclidean algorithm has no requirements for the modulus and can determine the existence of the inverse modulus and return an error if it does not exist. This feature makes it possible to integrate the greatest common divisor (GCD) operation in RSA into the modular inversion

operation, saving hardware resources. Furthermore, the extended Euclidean modular inversion algorithm itself is compatible with binary field operations, making it more suitable for designing a dual-field modular inversion algorithm for arbitrary modulus values.

The original extended Euclidean algorithm is based on the multiplication of large integers, requiring greater hardware resources. Because of this, the binary system extended Euclidean algorithm has been proposed, replacing division with simple shift, addition, and subtraction operations. However, in the calculation process, the values of A, B, C, and D may be greater than the modulus value due to accumulation, or even exceed the limit of data bit width, resulting in data loss. The addition of carry storage will lead to an increase in area overhead. Thus, we use this extended Euclidean algorithm and increase the modular operation after each round of accumulation to ensure that its value is less than the modulus value. The improved algorithm is shown in Algorithm 3, in which the individual operations have also been optimized.

Algorithm 3 Dual-field extended Euclidean modular inversion algorithm

1: **Input:** $X = (x_{m-1}, \ldots, x_1, x_0)_{2^r}$, $Y = (y_{m-1}, \ldots, y_1, y_0)_{2^r}$
2: **Output:** $Z = Y^{-1} \bmod X$ or error
3: **if** X is even and Y is even **then**
4: return error
5: **end if**
6: $u = x$, $v = y$, $A = 1$, $B = 0$, $C = 0$, $D = 1$
7: **while** $u \neq 0$ **do**
8: **if** u is even **then**
9: $u = u \gg 1$
10: **if** A is even and B is even **then**
11: $A = A \gg 1$, $B = B \gg 1$
12: **else**
13: $A = (A \oplus y) \gg 1$, $B = (B \ominus x) \gg 1$
// "$\oplus$" and "$\ominus$" mean addition and subtraction
// on the prime field respectively, and both of
// them mean XOR on the binary field
14: **end if**
15: **end if**
16: **if** field = 0 and u is even **then** go to line 8
17: **else** go to line 19
18: **end if**
19: **if** v is even **then**
20: $v = v \gg 1$
21: **if** C is even and D is even **then**
22: $C = C \gg 1$, $D = D \gg 1$
23: **else**
24: $C = (C \oplus y) \gg 1$, $D = (D \ominus x) \gg 1$
25: **end if**
26: **end if**
27: **if** field = 0 and v is even **then** go to line 19
28: **else** go to line 30
29: **end if**
30: **if** $u \geq v$ **then**
31: $u = u \ominus v$, $A = A \ominus C$, $B = B \ominus D$
32: **if** field = 1 **then**
33: $A = A \bmod x$, $B = B \bmod x$
34: **end if**
35: **else**
36: $v = v \ominus u$, $C = C \ominus A$, $D = D \ominus B$
37: **if** field = 1 **then**
38: $C = C \bmod x$, $D = D \bmod x$
39: **end if**
40: **end if**
41: **end while**
42: **if** $v \neq 1$ **then** return error
43: **else if** $D \leq 0$ & field = 1 **then** $z = x + D$
44: **else** $z = D$
45: **end if**

4 Reconfigurable modular unit

4.1 Arithmetic and logic unit

Further analysis of the modular multiplication and modular inversion algorithm shows that the basic operation in Algorithm 2 is a multiply-accumulate operation with a word length fixed, while the basic operations of Algorithm 3 are addition, subtraction, and shift operations. By adopting complementary operations, addition and subtraction can be unified as an addition operation, so an operation unit with multiply-accumulate and shift operations can perform modular multiplication and modular inversion operations at the same time. However, given that the modular multiplication operation uses unsigned ones for its addition and subtraction and that the corresponding operations in the modular inversion algorithm use signed ones, we select several groups of operations involved in Algorithm 3 to study the relationship between the addition and subtraction of signed and unsigned quantities. The results are shown in Table 1.

According to Table 1, the numerical part of all signed number operations can be completed by unsigned numeric addition and subtraction operations, while the symbolic part depends only on the original symbol and carry.

When lines 8 and 19 in Algorithm 3 are executed, the symbols of A, B, C, and D can be expressed as $A_{\text{sym}} = A_{\text{sym}} \& (\sim J)$, $B_{\text{sym}} = B_{\text{sym}} | (\sim J)$, $C_{\text{sym}} = C_{\text{sym}} \& (\sim J)$, $D_{\text{sym}} = D_{\text{sym}} | (\sim J)$ (here, J represents the carry value of the

Table 1 Sign relationships between addition and subtraction of signed number operations

Operation in symbolic form	Operations in Algorithm 3	Result sign	Result carry	Result value
Positive+positive	$A+y$, $C+y$	Positive	J	Z
Positive−positive	$B-x$, $D-x$, $A-C$, $C-A$, $B-D$, $D-B$	J	0	Z
Positive+negative	None	J	0	Z
Positive−negative	$A-C$, $C-A$, $B-D$, $D-B$	Positive	J	Z
Negative+negative	None	Negative	J	Z
Negative−negative	$A-C$, $C-A$, $B-D$, $D-B$	J	0	Z
Negative+positive	$A+y$, $C+y$	J	0	Z
Negative−positive	$B-x$, $D-x$, $A-C$, $C-A$, $B-D$, $D-B$	Negative	J	Z

J represents the carry value of the unsigned number operation with two complements. Z represents the result of the unsigned number operation with two complements

unsigned number operation with two complements). When line 30 is executed, the form of the operation is similar, and its symbolic relationship can be expressed as $S_{\text{sym}} = (\sim (J|P)) \,|\, (A_{\text{sym}} \& P)$. The symbols of S_{sym}, A_{sym}, B_{sym}, C_{sym}, and D_{sym} represent the sign of S, A, B, C, and D, respectively. If it is a positive number, its logic is 0; if it is negative, its logic is 1.

Based on the above analysis, we can design a set of symbols and carry registers so that unsigned addition and subtraction operations complete the signed addition and subtraction operations, with register values updated synchronously with each operation. In this way, the main structure of modular multiplication and modular inversion operations can be unified using only unsigned multiply accumulation.

The parallel multiplier consists of three parts: partial product generator (PPG), partial product adder (PPA), and carry propagation adder (CPA). Miyamoto et al. (2011) analyzed the delay of these three parts and proposed a half carry storage multiplication scheme. It also split the $2n$-bit result generated by the PPA, allowed the high n-bit result to go to the next round of addition without any processing, and obtained the current round result simply by adding the low n-bit data, reducing the CPA delay by half. However, this method makes the multiplication and accumulation of signed number operations impossible. If the high n-bit data is not processed, the positive or negative results of this round are not obtained, leading to chaos in the carry and borrow stage. To deal with this flaw, we use a carry select adder to calculate the carry sum synchronously, adding only one multiplexer delay. Fig. 1 shows the improved unsigned multiply-accumulate structure.

There are many PPG and PPA design schemes. In this study, we adopt a Radix-4 Booth code multiplier (Chen HM et al., 2012) and a 4-2 and 3-2 compression mixed Wallace tree design PPA.

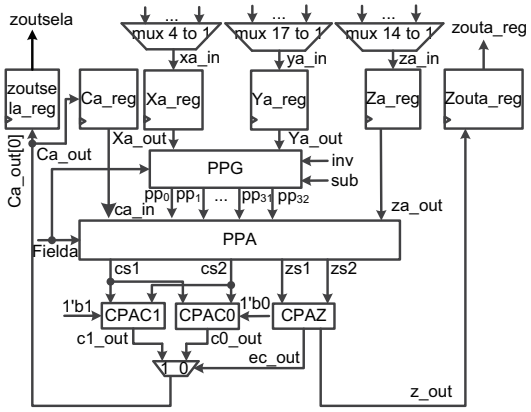

Fig. 1 Structure of the unsigned multiply-accumulate stage (PPG: partial product generator; PPA: partial product adder; CPA: carry propagation adder)

In our work, there are only two equality tests: U being equal to 0 and V being equal to 1 in the modular inversion operation. Therefore, the handling of 576-bit data can be achieved at one clock cycle. If the data width is greater than 576 bits, we keep the lowest 64-bit data unchanged and reconstruct and fill the high bits to complete the equality.

The preceding analysis proves that a single set of registers is essential for the addition and subtraction of signed number operations to store the sign and the carry temporarily. If there is a shift after carry, two cases may result: shift the sign or shift the carry, which requires another set of registers for temporarily storing the possible shift. In Algorithm 3, only one-bit data of the shift is needed, because modular operations would be performed for the results in each round. Algorithm 3 also requires the determination of parity and modular inversion. Thus, a set of parity state registers and modular inversion error registers must be used, forming a state register heap.

4.2 Block storage structure

In the ECC encryption algorithm, a small key meets common security requirements, while RSA uses larger keys. Static random-access memory (SRAM) is thus a better fit for RSA, having a smaller area than a comparable register, but with poorer performance. In contrast, registers are more suitable for ECC. For compatibility, we adopt a hybrid memory design of shift registers and SRAM.

The block storage structure includes 8×4096-bit SRAM and 8×576-bit shift registers. One SRAM corresponds to one shift register, and each SRAM interacts only with its corresponding SRAM. Fig. 2 shows the design.

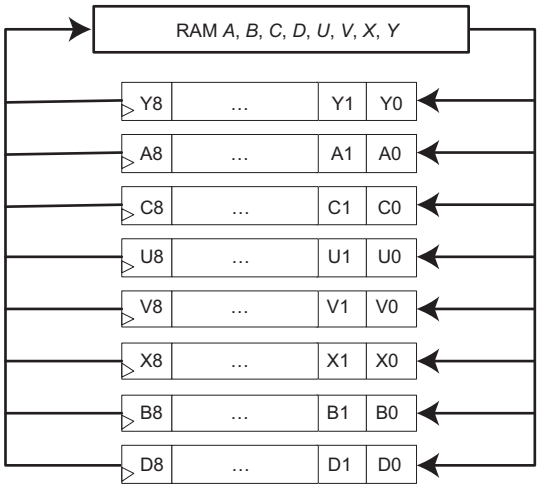

Fig. 2 Block storage structure

4.3 State control circuit

To improve the parallel degree of operation, we use a two-way multiply-accumulate structure to form the multiply-accumulate circuit. The difference between the two way structures lies in the X, Y, and Z data selection entries. Our design performs lines 9 and 10 of Algorithm 2 in parallel, and likewise the addition and subtraction steps of Algorithm 3.

We use a finite state machine (FSM) as the control circuit to generate control signals and to drive shift registers and memory units to provide the corresponding data words for the operation unit. We employ shift registers to complete the pipeline structure. For modular multiplication, SRAMs X, Y, B, D, V, and A store data X, Y, N, W, Z, and T from Algorithm 2, respectively. For modular inversion, memory blocks X, Y, U, V, A, B, C, and D store data Z, Y, U, V, A, B, C, and D from Algorithm 3, respectively. For any modular operation whose width is fewer than 576 bits, the memory block uses the shift registers only and bypasses the SRAMs. For any modular operation whose width is within 4096 bits (2048×2048), it uses SRAMs to store big data and shift registers as operational cache. In the following paragraphs, we analyze the modular multiplication and inversion of 512-bit and 2048-bit data.

We first consider the modular multiplication of 512 bits, i.e., 8×64-bit words, using the structure proposed in this study. We calculate the first step of external cycle by taking the lowest 64 bits of the X, Y, and V shift registers. We take the lowest 64 bits of the D shift register and the results of this round as the input to the next operation. The inner layer of the modular multiplication operation is divided into two directions in parallel, known as a-way and b-way. The a-way operation takes the lowest 64 bits of the X shift register and the fourth 64 bits of Y and V shift registers (the lowest is the first 64 bits) and stores the results in the fourth 64 bits of the V shift register. The b-way operation takes the lowest 64 bits of the A shift register and the third 64 bits of the B and V shift registers (the lowest is the first 64 bits) and stores the results in the second 64 bits of the V shift register. After that, the V shift register is synchronized with the Y and B shift registers, moving 64-bit data to the right to enter the next cycle. At the end of each external cycle, the X shift register moves the 64-bit data to the right to enter the next cycle. When the external cycle ends, all data is rotated to the initial position. At this time, the lowest 64-bit data of the V and B registers is used for the subtraction operation. After each 64-bit data calculation, the data is rotated to the right. After eight cycles, the results are stored in the X shift register. If the subtraction result is negative, the value of the V shift register is output. Otherwise, the value of the X shift register is output. Fig. 3 shows the overall process.

Next, we consider modular multiplication of 2048-bit data, i.e., 32×64-bit words, and use the structure described in the previous example with RAMs added. RAMs remove the data needed for the next operation one clock cycle in advance and send it to the corresponding shift registers. Fig. 4 shows the flow of an example loop.

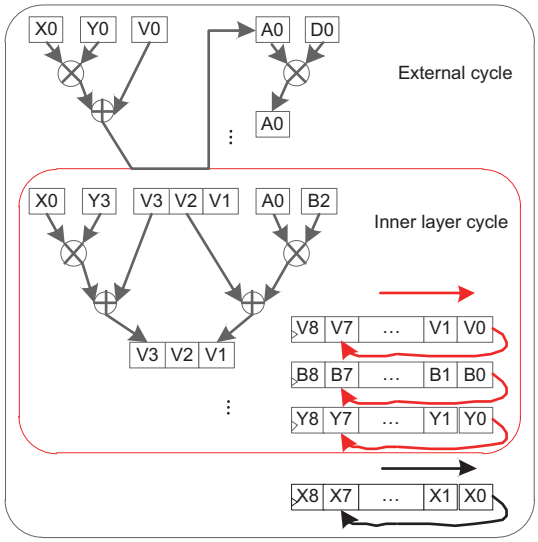

Fig. 3 Operation flow of the internal modular multiplication

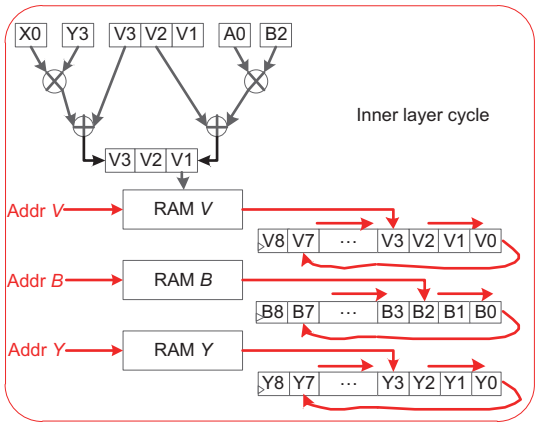

Fig. 4 Modular operation flow when the data length is over 576 bits

Now, we consider the modular inversion of 512-bit data, i.e., 8×64-bit words. Algorithm 3 successively performs equality, shift, addition, and subtraction operations. The operation logic has two 576-bit equality units and can determine data equality in one clock cycle. Operations on data within 576 bits are stored in the shift registers, enabling the data shift to end within one clock cycle. Addition and subtraction operations use 64 bits as the basic operation length, completing 512-bit data addition and subtraction in eight clock cycles.

Finally, we consider the modular inversion of 2048-bit data, i.e., 32×64-bit words. The modular operation unit can determine the equality of 8×64-bit data in nine clock cycles from low to high bits. After completing the operation on the low 9×64 bits, the lowest 64-bit data is kept unchanged, with 8×64-bit words written into the high shift register. This continues until the equality check of the 2048-bit data is finished. The shift operation proceeds from high to low bits and completes a 64-bit data shift in one clock cycle. The addition and subtraction operations are similar to those described in the previous paragraph, with the data in RAMs sent to the shift registers for calculation one clock cycle in advance.

4.4 Overall architecture

The overall architecture, with block storage as the center, is composed of three parts: operation, control, and block storage. The architecture is shown in Fig. 5. The operation unit includes a 64-bit double-multiply-accumulate module, a 576-bit data equality module, and a state register heap. The control unit can perform basic operations on 64-integer-multiple bit width, including multiplication, addition, subtraction, shift, and equality. Modular inversion can also be realized through the combination of these basic operations. The block storage unit is composed of eight 64×64-bit SRAMs and eight shift registers corresponding to block SRAMs. The shift registers are used as SRAM cache to balance the performance and area overhead of different bit width data requirements.

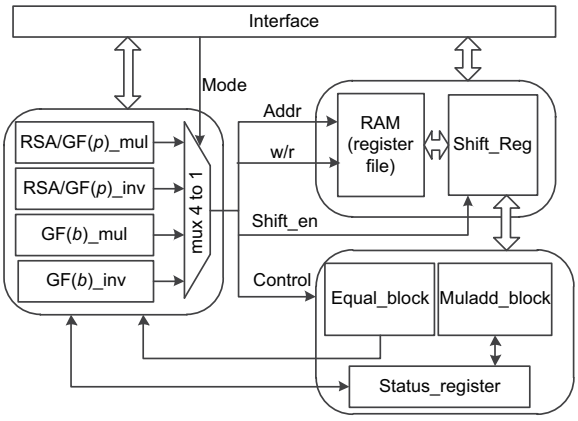

Fig. 5 General architecture of the reconfigurable modular units

5 Functional test and performance evaluation

5.1 Performance comparison and analysis

The architecture has been synthesized using a 55-nm CMOS process at TT corner (1.2 V, 25 °C).

The public key operation unit occupies 437 801 μm^2 of hardware resources with a maximum clock frequency of 588 MHz. We also perform power simulation with random input patterns. Table 2 provides the performance results for each operation.

Our design supports modular operations on any data width. Table 2 shows the performances of RSA modular multiplication, RSA modular inversion, ECC dual-field modular multiplication, and ECC modular inversion for different data widths.

Fig. 6 shows that the time consumed by the modular inversion operation does not increase significantly with increasing data width, showing its adaptability to large data width. The time required by the modular multiplication operation increases exponentially with the increase in the operation data width. However, the overall time is short, showing high performance with any data width.

Tables 3 and 4 compare our proposed unit with others. Not all the literature provides power consumption data, so we use speed area product as the power substitution parameter for comparison. We also select some literature that provides power consumption data for comparative analysis. The results are shown in Table 5. To facilitate comparison with other works, we introduce the speed area factor "AT" (Lee et al., 2014; Choi et al., 2018; Ding et al., 2019).

The speed area product of our proposed RSA modular multiplication operation is 1/5 that of Kuang et al. (2016). The performance of our design with RSA modular inversion is 1/20 that of Xia (2016), but our unit has more functions as the speed area product is six times that of Xia (2016).

Chen GH et al. (2010) proposed the reconstruction design of modular multiplication and inversion operation for dual-field ECC. Our design reduces the speed area product of the modular multiplication operation by about 50%, improves the performance of the modular inversion operation by 10%, and increases the area cost by about four times. The improvements are possible because the proposed operation unit completes not only the ECC algorithm but also the RSA key generation without additional precalculated parameters, making it more convenient

Table 2 Public key operation unit performance

Operation	Number of clock cycles	Operation time	Power (mW)
2048-bit RSA modular multiplication	1251	2.12 μs	21.92
2048-bit RSA modular inversion	303 502	0.50 ms	23.36
512-bit ECC prime-field modular multiplication	115	195.50 ns	16.16
512-bit ECC prime-field modular inversion	15 476	26.31 μs	15.88
512-bit ECC binary-field modular multiplication	105	178.50 ns	16.01
512-bit ECC binary-field modular inversion	16 665	28.33 μs	15.88

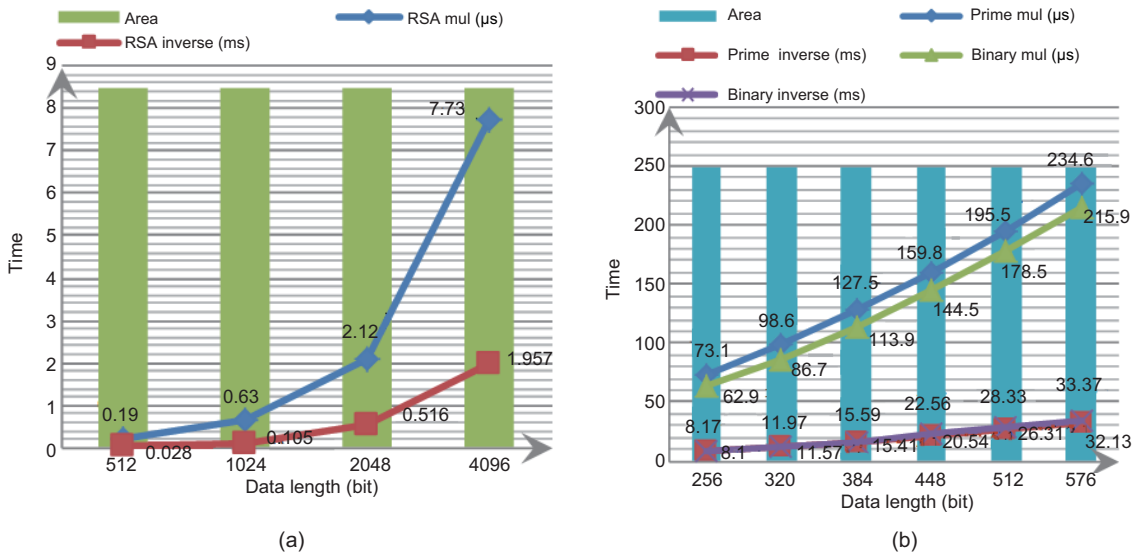

Fig. 6 Performance of RSA (a) and ECC (b) modular operations

Table 3 Performance comparison of 2048-bit RSA modular multiplication and modular inversion operations

Method	Process (nm)	Area	Speed		AT	
			mul	inv	mul	inv
Kuang et al. (2016)'s	90	950 184 μm^2	7.64 μs	None	14.443 s · μm^2	None
Xia (2016)'s	130	3770 gates	None	2.62 ms (1024-bit)	None	13.67 s · gates
Ours	55	137 801 μm^2 (2.36×10^5 gates)	2.12 μs	0.105 ms (1024-bit)	3.037 s · μm^2	81.1 s · gates

AT=speed×area×180/process with the unit of s · μm^2 or s · gates; mul: modular multiplication; inv: modular inversion

Table 4 Performance comparison of 256-bit ECC modular multiplication and modular inversion operations

Method	Process (nm)	Area ($\times10^3$ gates)	Speed^1		AT^1 (s · gates)		Speed^2		AT^2 (s · gates)	
			mul	inv	mul	inv	mul	inv	mul	inv
Chen GH et al. (2010)'s	180	51.83	2110 ns	25.36 μs	0.11	1.32	2110 ns	25.36 μs	0.11	1.32
Li et al. (2019)'s	130	77.1	360 ns	None	0.038	None	None	None	None	None
Ibrahim and Gebali (2017)'s	45	2.23	None	None	None	None	299 ns (233-bit)	0.449 μs	0.002	0.004
Ours	55	236	73.1 ns	8.1 μs	0.056	6.20	62.9 ns	8.17 μs	0.048	6.25

AT = speed × area × 180/process with the unit of s · gates; mul: modular multiplication; inv: modular inversion. The superscript "1" means for $G(p)$ and superscript "2" means for $G(2^m)$

Table 5 Power comparison with other literature

Method	Process (nm)	Operation	Energy
Ours	55	1024-bit RSA modular multiplication	12.17 nJ
		1024-bit RSA modular inversion	2.28 μJ
		256-bit ECC prime-field modular multiplication	1.04 nJ
		256-bit ECC prime-field modular inversion	114 nJ
		256-bit ECC binary-field modular multiplication	0.88 nJ
		256-bit ECC binary-field modular inversion	113 nJ
		256-bit ECC prime-field point multiplication	3.36 μJ
		256-bit ECC binary-field point multiplication	2.29 μJ
Gu and Li (2019)'s	65	256-bit modular multiplication for both RSA and ECC	2.01 nJ*
	90	1024-bit modular multiplication for both RSA and ECC	18.54 nJ**
Liu et al. (2017)'s	55	256-bit ECC dual-field point multiplication	49.7 μJ

*2.01 nJ=1.95 nJ×$(55/65)^2\times1.2^2$; ** 18.54 nJ=34.48 nJ×$(55/90)^2\times1.2^2$

and complete to use. Li et al. (2019) put forward a single type of ECC operation design for the prime field, with better performance and area optimization. However, the data they provided does not contain storage resources and the structure they proposed is not scalable, while our approach has a multifunctional reconstruction design with greater flexibility and wider application. Ibrahim and Gebali (2017) proposed an ECC operation design on the binary field with small areas, causing the speed area product to be much less than ours. We offer three reasons for this. First, Ibrahim and Gebali's design cannot be applied in the prime field, nor could it perform the RSA operation. Second, the structure they proposed is a 233-bit array. It can only do operations smaller than 233 bits. Last but not the least, they just gave the area for the operation array and did not cover the storage area. However, the area that we are giving includes storage units.

Overall, the controllable area and performance cost of our proposed unit support the basic operations of RSA and ECC public key cryptography algorithms with arbitrary data widths. It satisfies the energy-efficiency requirements of various public key algorithms.

As shown in Table 5, for RSA modular multiplication, our work consumes 65% of the energy of Gu and Li (2019). When performing modular multiplication of ECC prime fields, our work consumes 60% of the energy of Gu and Li (2019). Since we test only the power consumption of a modular operation unit, we integrate the unit into the processor to

collect the time consumption of the point multiplication operation. We obtain the energy consumption through calculation and compare it with that of Liu et al. (2017). The comparison results show that the energy consumption of Liu et al. (2017) is more than 10 times that of our unit. To sum up, our proposed modular operation unit meets the requirement for high energy efficiency.

5.2 Verification and analysis on the system level

We use an open-source, single-core PULPino 64-bit RISC-V processor to extend the public key operation unit and construct a test platform. Based on the test platform, we extend the modular operation instructions using the original RISC-V instruction set and realize the key generation, encryption and decryption, digital signature, and other functions of RSA and ECC programmatically. The extended instructions are shown in Table 6.

Taking the ECC point multiplication and RSA encryption as examples, Table 7 summarizes the performance comparision of this design to others. We use the traditional RL power algorithm to achieve RSA encryption. The ECC point multiplication is calculated from left to right under the affine coordinate system.

According to the data in the preceding table, the energy efficiency of our proposed unit is twice as high as that of Kuang et al. (2013) for RSA modular exponentiation. Compared with Li et al. (2019), our design is faster but requires twice as much area, causing the speed area product to be lower. However, Li et al.'s work can complete the ECC prime field point multiplication operation only up to 256 bits, which needs fewer area resources. Compared with Liu et al. (2017), the speed area product of our unit has been increased by more than five times, meeting the demand for high energy efficiency.

6 Conclusions

We have proposed a reconfigurable public key operation unit, with a memory unit as the center and double-multiply-accumulate structures. The key delay has been reduced by the fusion structure of the condition select adder and semi-carry storage multiplier, and the structure of unsigned multiply-accumulate has been improved to apply to the multiply accumulator of signed number operations. In memory processing, we have used SRAMs and shift registers to balance the performance area conflict caused by small-data-length ECC operations and large-data-length RSA operations. In addition to

Table 6 Extended instructions for modular operation

Instruction	Operand A	Operand B	Result	Description
SETMMI	Length (6-bit)	Mode (2-bit)	None	The operation mode setting instruction specifies the operation length and operation mode. There are four modes: ordinary modular multiplication, ordinary modular inversion, modular multiplication in the binary field, and modular inversion in the binary field. The length is a multiple of 64.
STARTMMI	None	None	None	Starting operation instruction
WMMI	Datain (64-bit)	Addr (2-bit)	None	Indicating the address of the four parameters (X, Y, N, W)
RMMI	None	None	Dataout	Output instruction

Table 7 Overall performance comparison of RSA and ECC designs

Method	Process (nm)	Area	Speed (ms)			AT		
			$G(p)$	$G(2^m)$	RSA	$G(p)$	$G(2^m)$	RSA
Li et al. (2019)'s	130	7.71×10^4 gates	0.86	None	None	91.8 s · gates	None	None
Kuang et al. (2013)'s	130	714 676 μm^2	None	None	2.36	None	None	2335.15 s · μm^2
Liu et al. (2017)'s	55	350 000 μm^2	1.45	1.45	None	1660.90 s · μm^2	1660.90 s · μm^2	None
Ours	55	437 801 μm^2 (2.36×10^5 gates)	0.224	0.153	0.96	320.95 s · μm^2 (173.01 s · gates)	219.22 s · μm^2	1375.49 s · μm^2

AT=speed×area×180/process with the unit of s · μm^2 or s · gates; $G(p)$: 256-bit ECC ($G(p)$) point multiplication; $G(2^m)$: 256-bit ECC ($G(2^m)$) point multiplication; RSA: 1024-bit RSA exponentiation

modular multiplication and modular inversion, our proposed structure can complete modular addition and subtraction, signed addition and subtraction, unsigned addition and subtraction, shift, and equality, which can complete the RSA or ECC cryptographic algorithm by programming. Our proposed unit took 21.92 mW and 23.36 mW for 2048-bit RSA modular multiplication and modular inversion respectively, as well as 16.16 mW and 15.88 mW to complete 512-bit ECC dual-field modular multiplication and modular inversion respectively. It took 0.224 ms and 0.153 ms for 256-bit ECC point multiplication in $G(p)$ and $G(2^m)$ respectively, as well as 0.96 ms to complete 1024-bit RSA exponentiation. The results showed that our modular unit is more efficient and flexible than existing designs.

Contributors

Mengni BIE, Wei LI, and Tao CHEN performed the investigation, participated in the formulation of research plans, and designed the research. Mengni BIE and Danyang YANG processed the data. Mengni BIE drafted the paper. Wei LI and Longmei NAN helped organize the paper. Wei LI revised and finalized the paper.

Compliance with ethics guidelines

Mengni BIE, Wei LI, Tao CHEN, Longmei NAN, and Danyang YANG declare that they have no conflict of interest.

References

5G Infrastructure Public Private Partnership (5G PPP), 2016. View on 5G Architecture. White Paper. https://5g-ppp.eu/

Chen GH, Zhu JM, Liu M, et al., 2010. Dual-field modular multiplication algorithm and modular inversion algorithm with VLSI implementation. *J Electron Inform Technol*, 32(9):2095-2100 (in Chinese). https://doi.org/10.3724/SP.J.1146.2009.01258

Chen HM, Li Z, Xie TD, 2012. Optimal design of multiplier based on Radix-4 Booth encoding. *Comput Eng*, 38(1):233-235 (in Chinese). https://doi.org/10.3969/j.issn.1000-3428.2012.01.076

Choi P, Lee MK, Kim JH, et al., 2018. Low-complexity elliptic curve cryptography processor based on configurable partial modular reduction over NIST prime fields. *IEEE Trans Circ Syst II*, 65(11):1703-1707. https://doi.org/10.1109/TCSII.2017.2756680

Ding JN, Li SG, Gu Z, 2019. High-speed ECC processor over NIST prime fields applied with Toom–Cook multiplication. *IEEE Trans Circ Syst I*, 66(3):1003-1016. https://doi.org/10.1109/TCSI.2018.2878598

GSMA Intelligence, 2014. Understanding 5G: Perspectives on Future Technological Advancements in Mobile. White Paper. https://www.gsma.com/

Gu Z, Li SG, 2019. A division-free Toom–Cook multiplication-based Montgomery modular multiplication. *IEEE Trans Circ Syst II*, 66(8):1401-1405. https://doi.org/10.1109/TCSII.2018.2886962

Ibrahim A, Gebali F, 2017. Scalable and unified digit-serial processor array architecture for multiplication and inversion over GF(2^m). *IEEE Trans Circ Syst I*, 64(11):2894-2906. https://doi.org/10.1109/TCSI.2017.2691353

Kaya Koc C, Acar T, Kaliski BS, 1996. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26-33. https://doi.org/10.1109/40.502403

Kuang SR, Wang JP, Chang KC, et al., 2013. Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems. *IEEE Trans Very Large Scale Integr Syst*, 21(11):1999-2009. https://doi.org/10.1109/TVLSI.2012.2227846

Kuang SR, Wu KY, Lu RY, 2016. Low-cost high-performance VLSI architecture for Montgomery modular multiplication. *IEEE Trans Very Large Scale Integr Syst*, 24(2):434-443. https://doi.org/10.1109/TVLSI.2015.2409113

Lee JW, Chung SC, Chang HC, et al., 2014. Efficient power-analysis-resistant dual-field elliptic curve cryptographic processor using heterogeneous dual-processing-element architecture. *IEEE Trans Very Large Scale Integr Syst*, 22(1):49-61. https://doi.org/10.1109/TVLSI.2013.2237930

Li B, Lei BJ, Zhang YL, et al., 2019. A novel and high-performance modular square scheme for elliptic curve cryptography over GF(p). *IEEE Trans Circ Syst II*, 66(4):647-651. https://doi.org/10.1109/TCSII.2018.2867618

Liu ZL, Liu DS, Zou XC, 2017. An efficient and flexible hardware implementation of the dual-field elliptic curve cryptographic processor. *IEEE Trans Ind Electron*, 64(3):2353-2362. https://doi.org/10.1109/TIE.2016.2625241

Miyamoto A, Homma N, Aoki T, et al., 2011. Systematic design of RSA processors based on high-radix Montgomery multipliers. *IEEE Trans Very Large Scale Integr Syst*, 19(7):1136-1146. https://doi.org/10.1109/TVLSI.2010.2049037

Next Generation Mobile Networks (NGMN), 2015. NGMN 5G. White Paper. https://www.ngmn.org/

Xia JF, 2016. Design of RSA Key Pair Accelerating Circuit for Smart Card. MS Thesis, Huazhong University of Science and Technology, Wuhan, China (in Chinese).