Mengni BIE, Wei LI, Tao CHEN, Longmei NAN, Danyang YANG, 2022. An energy-efficient reconfigurable asymmetric modular cryptographic operation unit for RSA and ECC. *Frontiers of Information Technology & Electronic Engineering*, 23(1):134-144. https://doi.org/10.1631/FITEE.2000325



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

**Key words:** Modular operation unit; Reconfigurable; High energy efficiency

Corresponding author: Wei LI
E-mail: liwei12@fudan.edu.cn
ORCID: https://orcid.org/0000-0002-6597-0142

# Motivation

- Although the ellipse curve cryptography (ECC) algorithm has been widely used in 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 secure socket layer (SSL) protocol encryption algorithms.
- As the basic operation of public key cryptography, the modular operation has been the subject of many studies, but it is relatively targeted. 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.

# Contents

- We propose an optimized dual-field high-radix Montgomery modular multiplication algorithm and a dual-field extended Euclidean modular inversion algorithm that are more efficient for RSA and ECC.
- 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. Taking modular multiplication and modular inversion as examples, we discuss how to do efficient computation based on this unit.
- 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.

# Dual-field high-radix Montgomery modular multiplication algorithm

## Modular multiplication for RSA

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

**Input:** $X = (x_{m-1}, \ldots, x_1, x_0)_{2^r}$, $Y = (y_{m-1}, \ldots, y_1, y_0)_{2^r}$, $N = (n_{m-1}, \ldots, n_1, n_0)_{2^r}$, $w = -N^{-1} \bmod 2^r$
**Output:** $Z = XY \cdot 2^{-rm} \bmod N$
$Z = 0$, $v = 0$
**for** $i$=0 to $m-1$ **do**
  $(\mathrm{cs1}_a + \mathrm{cs2}_a + \mathrm{ec}_a, z_0) = z_0 + x_i y_0$
  $t_i = z_0 w \bmod 2^r$
  $(\mathrm{cs1}_b + \mathrm{cs2}_b + \mathrm{ec}_b, z_0) = z_0 + t_i n_0$
  **for** $j$=1 to $m-1$ **do**
    $(\mathrm{cs1}_a + \mathrm{cs2}_a + \mathrm{ec}_a, z_j) = z_j + x_i y_j + \mathrm{cs1}_a + \mathrm{cs2}_a + \mathrm{ec}_a$
    $(\mathrm{cs1}_b + \mathrm{cs2}_b + \mathrm{ec}_b, z_{j-1}) = z_j + t_i n_j + \mathrm{cs1}_b + \mathrm{cs2}_b + \mathrm{ec}_b$
  **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**

## Modular multiplication for RSA and ECC

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

**Input:** $X = (x_{m-1}, \ldots, x_1, x_0)_{2^r}$, $Y = (y_{m-1}, \ldots, y_1, y_0)_{2^r}$, $N = (n_{m-1}, \ldots, n_1, n_0)_{2^r}$, $w = -N^{-1} \bmod 2^r$
**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**

# Dual-field extended Euclidean modular inversion algorithm

**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**

# Reconfigurable modular unit

**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

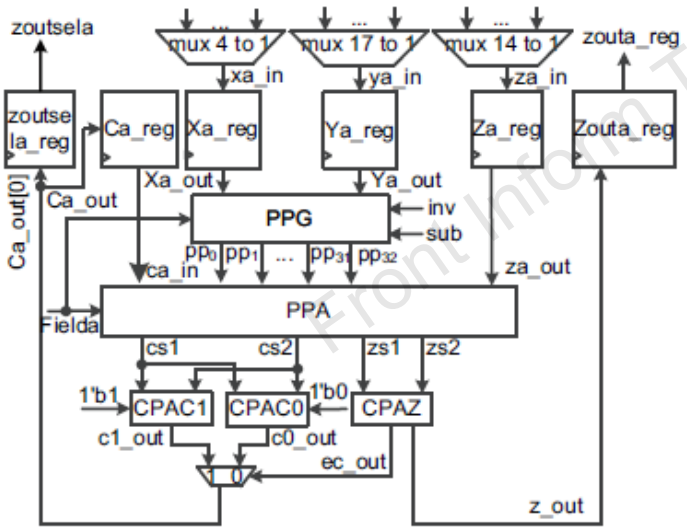


**Fig. 1 Structure of the unsigned multiply-accumulate stage (PPG: partial product generator; PPA: partial product adder; CPA: carry propagation adder)**

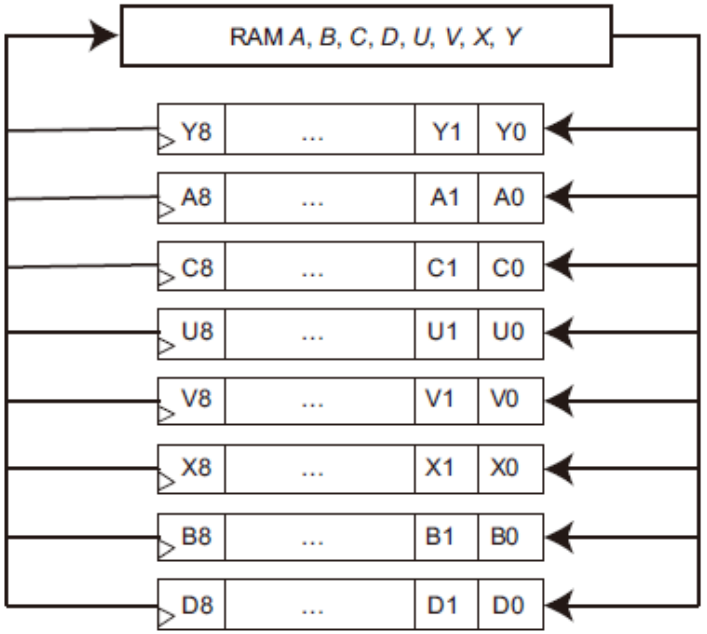


**Fig. 2 Block storage structure**

# Reconfigurable modular unit ——state control circuit

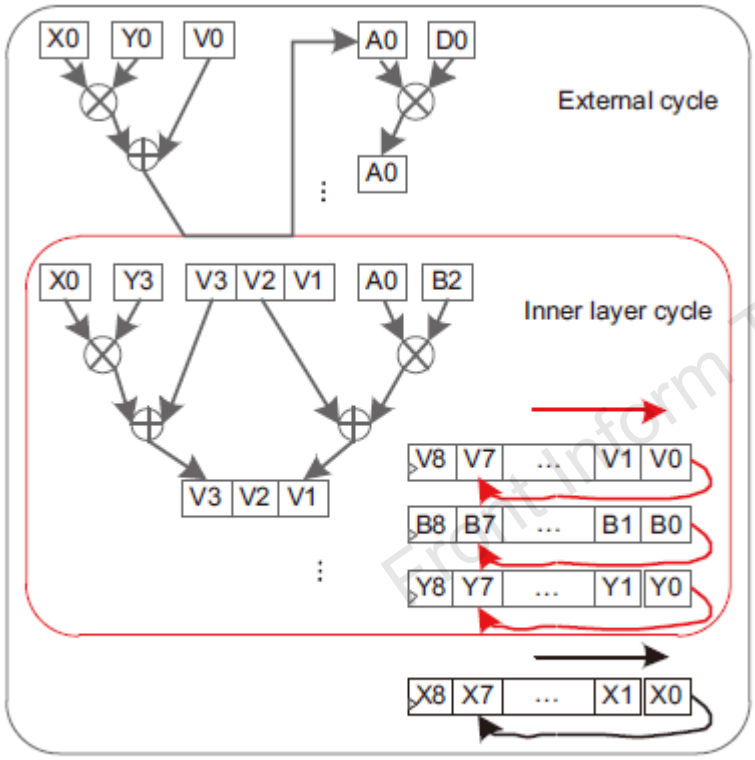


Fig. 3 Operation flow of the internal modular multiplication

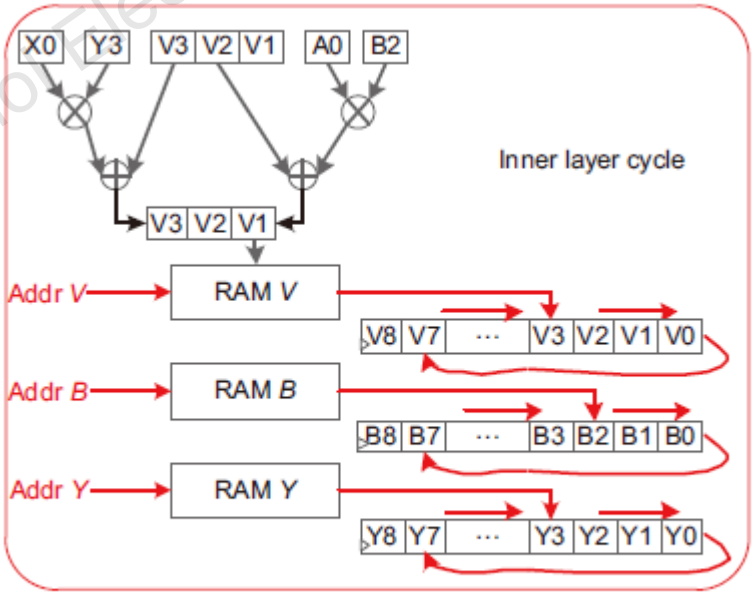


Fig. 4 Modular operation flow when the data length is over 576 bits

# General architecture of the reconfigurable modular units

## Architecture

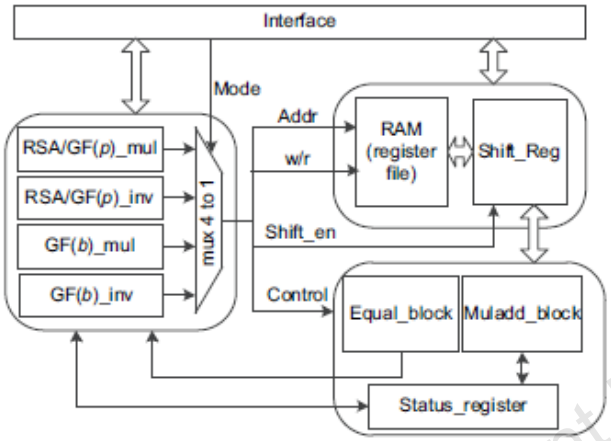


**Fig. 5** General architecture of the reconfigurable modular units

## Performance

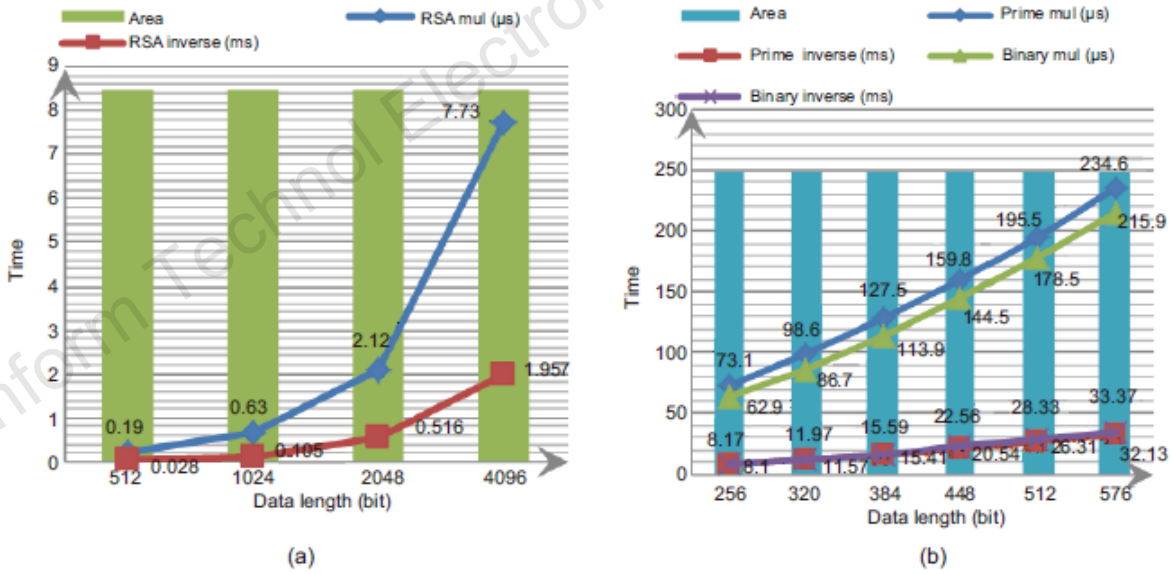


**Fig. 6** Performance of RSA (a) and ECC (b) modular operations

# General architecture of the reconfigurable modular units

## Instruction



**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 |

# 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 selection adder and semi-carry storage multiplier, and the structure of unsigned multiply-accumulate has been improved to be applied to the multiply accumulator of signed numbers.
- In memory processing, we have used static random-access memories (SRAMs) and shift registers to balance the performance area conflict caused by small-data-length ECC operations and large-data-length RSA operations.
- Comparison results showed that our modular unit is more efficient and flexible than existing designs.