Systolic-Arrays for Modular Exponentiation Using Montgomery Method

— Extended Abstract —

Keiichi Iwamura†, Tsutomu Matsumoto††, and Hideki Imai††

Abstract — This paper proposes two ideas for modular exponentiation using Montgomery method. (1) A novel algorithm for modular exponentiation without operation of subtracting N for every Montgomery's modular multiplication (MMM). (2) Two types of systolic-array for MMM which can realize more efficient and flexible chip implementation than the array in [1].

1 Introduction

We have proposed a systolic-array for modular multiplication [1], which we will refer to as Array-A in the following. Array-A is practical for modular exponentiation and is very suitable for chip implementation. However Array-A does not achieve the ultimate efficiency in the wide range of processing speed, when the efficiency is defined as (processing speed)/(circuit scale). In this paper, we propose a novel algorithm for modular exponentiation using simple repetition of Montgomery's modular multiplication (MMM) [2] in Section 2, and in Section 3 we propose the structures and actions of more efficient systolic-arrays (Array-B, Array-C) suitable for MMM in the wide range of processing speed than Array-A. Combining these proposals, we have efficient and fast hardware algorithms for modular exponentiation. In Section 4, we show that compared with the use of Array-A the use of Array-B and that of Array-C respectively achieves more efficient high-speed processing and more flexible implementation for modular exponentiation.

2 A Novel Algorithm for Modular Exponentiation

Definition 1 For integers N, R, A, and B, let $\mathrm{Mont}_{N,R}(A,B)$ denote the rational number

$$\mathrm{Mont}_{N,R}(A,B) = \frac{A \cdot B + ((A \cdot B \cdot (-N^{-1} \bmod R)) \bmod R) \cdot N}{R}.$$

We have the following fact due to Montgomery [2].

Proposition 2 For relatively prime integers N and R, and for integers A and B, $\mathrm{Mont}_{N,R}(A,B)$ is an integer such that

$$\mathrm{Mont}_{N,R}(A,B) \equiv ABR^{-1} \bmod N \pmod N.$$

We call $\mathrm{Mont}_{N,R}(A,B)$ the Montgomery's Modular Multiplication (MMM).

The condition that the maximum number of bits in A, in B and in $\mathrm{Mont}_{N,R}(A,B)$ are the same is described as follows:

† Canon Research Center, 21 Laboratory,
5-1 Wakamiya, Morinosato, Atsugi-shi, Kanagawa 243-01, Japan,
Fax +81-462-48-0306, Tel +81-462-47-2111
†† Yokohama National University, Division of Electrical & Computer Engineering,
156 Tokiwadai, Hodogaya, Yokohama, 240 Japan,
Fax +81-45-338-1157, Tel +81-45-335-1451, Internet tsutomu@mlab.dnj.ynu.ac.jp

R.A. Rueppel (Ed.): Advances in Cryptology - EUROCRYPT '92, LNCS 658, pp. 477-481, 1993.

Theorem 3 Let N, R, A, B, n, r, and l be integers and assume

$$0 < N < 2^n, \quad 0 < R \leq 2^r, \quad \text{and} \quad \gcd(N, R) = 1.$$

The necessary and sufficient condition for that

$$0 \leq A < 2^l, \quad 0 \leq B < 2^l, \quad \text{and} \quad 0 \leq \text{Mont}_{N,R}(A, B) < 2^l$$

is

$$n + 1 \leq l \leq r - 1. \quad (1)$$

(Proof) Let $0 \leq A < 2^l$ and $0 \leq B < 2^l$. Since it is straightforward that

$$0 \leq \text{Mont}_{N,R}(A, B) < \max\{2^{2l-r+1}, 2^{n+1}\}$$

the necessary and sufficient condition for

$$0 \leq \text{Mont}_{N,R}(A, B) < 2^l$$

is that

$$\max\{2^{2l-r+1}, 2^{n+1}\} \leq 2^l$$

which is equivalent to

$$2^{2l-r+1} \leq 2^l \quad \text{and} \quad 2^{n+1} \leq 2^l$$

which is equivalent to (1).

Thus we let $n + 1 \leq l \leq r - 1$ so that the output of MMM can be directly used as the next input to MMM. The smallest possible value of l is $l = n + 1$ and of r is $r = n + 2$. Using this condition and introducing $R_R = R^2 \bmod N$, we propose the following algorithm which evaluates modular exponentiation $C = M^e \bmod N$ with repeated MMMs and a final modular reduction $\bmod N$.

Algorithm 4

(Input : $M, e = (e_k, \cdots, e_1)_2, N, R, R_R = R^2 \bmod N$)
(Output : $C = M^e \bmod N$)
$M_R = \text{Mont}_{N,R}(M, R_R)$
$C_R = \text{Mont}_{N,R}(1, R_R)$
FOR $i = k$ TO 1
  IF $e_i = 1$ THEN $C_R = \text{Mont}_{N,R}(C_R, M_R)$
  IF $i > 1$ THEN $C_R = \text{Mont}_{N,R}(C_R, C_R)$
NEXT
$C_R = \text{Mont}_{N,R}(1, C_R)$
$C = C_R \bmod N$

Algorithm 4 has the following useful features:

(1) Algorithm 4 employs ordinary modular multiplication only at the precomputation of R_R which can be done independently with M. All the rest of modular multiplications are MMM. This feature helps to obtain simple structure.

(2) Except for the last step, the maximum length of an output of each MMM used in Algorithm 4 is not greater than that of each of the inputs to the MMM. Thus, the output can be directly fed into the next MMM without compensation like Fig.1, which is to obtain $ABR^{-1} \bmod N$ from $\text{Mont}_{N,R}(A, B)$ and often used in conventional algorithms. This feature greatly simplifies the control structure.

(2') For implementation by systolic-array, feature (2) can be effectively used to avoid idle processing elements. If such compensations are required, any bit of the input to a systolic-array for MMM cannot been fed before getting all the bits of the previous value of the output from the MMM. However such a loss does not emerge in Algorithm 4 since the previous output can be directly input to the next MMM without delay.

In contrast, any of the previously known methods for modular exponentiation using MMM (see [3][4][5]) does not simultaneously satisfy the above features.

3 Systolic-Arrays for Montgomery's Modular Multiplication

In the following we propose systolic arrays for computing $\mathrm{Mont}_{N,R}(A,B)$ under the condition that N is odd, $n = \lfloor \log_2 N \rfloor + 1$, $n+1 \le l \le r-1$, $0 \le A < 2^l$, $0 \le B < 2^l$, and $R = 2^r$.

We express A in radix $Y = 2^v$ and B, N and $T_R = \mathrm{Mont}_{N,R}(A,B)$ in radix $X = 2^d$ as follows,

$$\begin{array}{lllll} A & = A_{k-1}\cdot Y^{k-1} & +A_{k-2}\cdot Y^{k-2} & +\cdots & +A_1\cdot Y \quad +A_0 \\ B & = B_{m-1}\cdot X^{m-1} & +B_{m-2}\cdot X^{m-2} & +\cdots & +B_1\cdot X \quad +B_0 \\ N & = N_{m-1}\cdot X^{m-1} & +N_{m-2}\cdot X^{m-2} & +\cdots & +N_1\cdot X \quad +N_0 \\ T_R & = T_{m-1}\cdot X^{m-1} & +T_{m-2}\cdot X^{m-2} & +\cdots & +T_1\cdot X \quad +T_0 \end{array}$$

where $A_i \in \{0,1\}^v$ $(i = 0,\cdots,k-1)$, and B_j, N_j and $T_j \in \{0,1\}^d$ $(j = 0,\cdots,m-1)$, and $v \le d$.

$T_R = \mathrm{Mont}_{N,R}(A,B)$ can be calculated by the consecutive execution of the following operation from $i = 0$ to $i = k$.

$$\begin{array}{ll} T(i) & = (T(i-1) + A_i\cdot B\cdot Y + M_{i-1}\cdot N)/Y \quad (2) \\ \text{where} & M_{i-1} = (T(i-1) \bmod Y)\cdot N'_0 \bmod Y, \\ & T(-1) = 0,\ N'_0 = N' \bmod Y \end{array}$$

3.1 Array-B

Expression (2) can be realized by the processing element (PE) described in Fig.2, when B and N are synchronously fed into the port $[B_{in}, N_{in}]$ of the PE as $[B_0, N_0], [B_1, N_1], [B_2, N_2], \ldots, [B_{m-1}, N_{m-1}]$. The multiplication of $Y = 2^v$ is realized by the v-bit shift. For example, $A_i \cdot B_j \cdot Y$ is obtained by shifting the value $A_i \cdot B_j$ v bits into the direction of more significant bits.

If $v = 1$ the PE in Fig.2 can be easily realized as follows. Each of the multipliers M1 and M2 is constructed with only d AND gates. Each of the registers R1 and R2 is a 1-bit register respectively holding A_i and M_{i-1}. Since $N'_0 = 1$, register R3 and multiplier M3 can be omitted, so that M_{i-1} can be the least significant bit of the output from adder A1. R4 and R5 are d-bit registers which respectively transfers B_j and N_j to the next PE with a delay of one clock cycle. A1 is a 5-input adder whose output is received by register R6 of $d+3$ bits. The most and the second significant bits of R6 are fed back to adder A1 as carry bits. The least significant bit of R6 is fed into terminal $L2_{in}$ of the next PE but one to this PE. The rest of the bits of R6 are transferred to the terminal T_{in} of the next PE.

To obtain T_R we construct Array-B shown in Fig.4 which consists of $k+1$ tandem PEs described in Fig.2 for the repetition of Expression (2) and of the last and the last second PEs described in Fig.3. The $k+1$ PEs described as Fig.2 are connected in tandem by respectively tying terminals B_{out}, $L2_{out}$, T_{out}, $L1_{out}$, M_{out}, N_{out} to the corresponding terminals B_{in}, $L2_{in}$, T_{in}, $L1_{in}$, M_{in}, N_{in} of the next PE. And for $i = 0,1,\ldots,k-1$, register R1 of the i-th PE in Fig.2 is preset by value A_i. Values of $L1_{in}$, T_{in}, $L2_{in}$ and M_{in} of the first PE are set to 0.

3.2 Array-C

T_R can be also evaluated by systolic-array Array-C described in Fig.6. For the sake of simplicity we assume in this section that $k = m$, i.e., $v = d$.

Array-C consists of m copies of the same PE shown in Fig.5. Each PE is connected in tandem by respectively tying terminals A_{out}, B_{out}, T_{out}, M_{out}, N_{out} to the corresponding terminals A_{in}, B_{in}, T_{in}, M_{in}, N_{in} of the next PE. Values for T_{in} and M_{in} in the first PE are set to 0.

The multiplication by Y is realized by timing shift. In Array-C A_i is input one-clock-cycle prior to B_j for $i, j = 0, 1, \ldots, m-1$. Therefore we can set register R1 in PE# 0 the value of A_0 before computing $A_0 \cdot B_j$. At each PE A_i delays one clock cycle while B_j delays two clock cycle. Thus if A_i can be set in PE# i before computing $A_i \cdot B_j$ $(j = 0, 1, \ldots, m-1)$ then A_{i+1} can be set in PE# $i+1$ before computing $A_{i+1} \cdot B_{j+1}$ $(j = 0, 1, \ldots, m-1)$. Therefore values of A_i can be input serially like as B_j and N_j. That is, we do not have to preset the values of A_i.

If $v = d = 1$, the PE in Fig.5 can be readily realized as follows. Each of multipliers M1 and M2 is constructed with only one AND gate, and multiplier M3 can be omitted. Each of registers R1 $\sim$ R8 is a 1-bit register and register R3 can be also omitted. A1 is a 4-input adder and R9 is a 3-bit register, and the least significant bit of R9 is fed into the terminal T_{in} of the next PE and the rest of the bits of R9 is fed back to adder A1 as carry bits.

4 Conclusion

Since Array-A, Array-B, and Array-C is respectively constructed with 3 types of PE, 2 types of PE, and 1 type of PE, Array-C is simpler than Array-B, which is simpler than Array-A.

For modular reduction, Array-A uses ROM tables while Array-B and Array-C use multipliers, which can be constructed with only AND gates and can be implemented faster than ROMs. Thus the circuit scale and the processing time of Array-B and Array-C are less than those of Array-A. While the size of operands acceptable by Array-A is bounded by the available ROM size, that of Array-B and that of Array-C have no such severe restriction. Therefore Array-B and Array-C can flexibly cope with changing sizes of operands.

And since Array-B contains less number of registers than Array-C, the former is more efficient than the latter which is more efficient than Array-A.

In conclusion Array-B and Array-C are more useful than Array-A when we realize efficient and flexible implementations for modular exponentiation.

References

[1] K.Iwamura, T.Matsumoto and H.Imai, "High-speed implementation methods for RSA scheme," to appear in Advances in Cryptology — EUROCRYPT'92, Springer-Verlag (a primary version has appeared in EUROCRYPT'92 EXTENDED ABSTRACTS, pp.215-224).

[2] P.L.Montgomery, "Modular multiplication without trial division," Math.of Computation, Vol.44, pp.519-521, 1985.

[3] S.R.Dussé and B.S.Kaliski Jr., "A cryptographic library for Motorola DSP56000," Advances in Cryptology — EUROCRYPT'90, pp.230-244, Springer-Verlag.

[4] S.Even, "Systolic modular multiplication, " Advances in Cryptology — CRYPTO'90, pp.619-624, Springer-Verlag.

[5] B.Dixon and A.K.Lenstra, "Massively parallel elliptic curve factoring," EUROCRYPT'92 EXTENDED ABSTRACTS, pp.169-179.

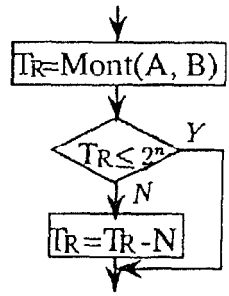

Fig.1: Compensation for MMM

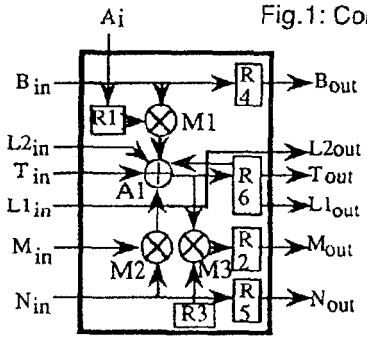

Fig.2: The PE used in Array-B

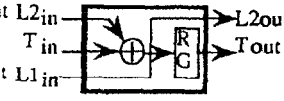

Fig3: Another type of PE used in the end of Array-B

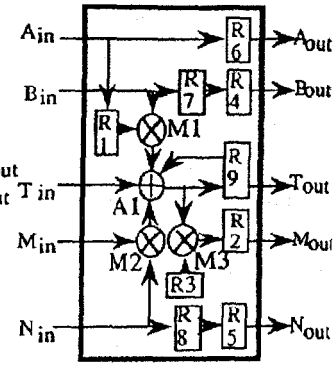

Fig.5: The PE used in Array-C

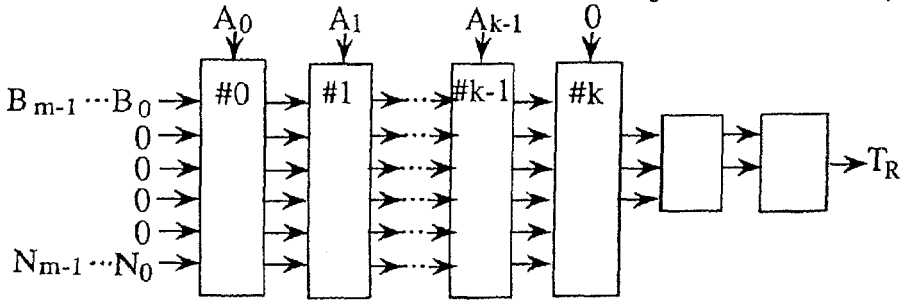

Fig.4: Array-B

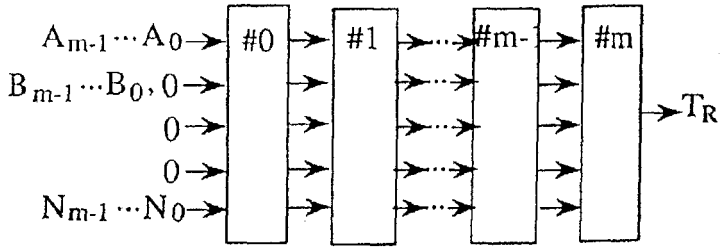

Fig.6: Array-C