Parallel Modular Multiplication Algorithm in Residue Number System

Hyun-Sung Kim[1], Hee-Joo Park[1], and Sung-Ho Hwang[2]

[1] Kyungil University, Computer Engineering,
712-701, Kyungsansi, Kyungpook Province, Korea
kim@kiu.ac.kr
[2] Pohang University of Sci. and Tech., Dept. of Computer Eng. and Sci.,
790-784, Pohangsi, Kyungpook Province, Korea

Abstract. This paper presents a novel method for the parallelization of the modular multiplication algorithm in the Residue Number System (RNS). The proposed algorithm executes modular reductions using a new lookup table along with the Mixed Radix number System (MRS) and RNS. MRS is used because algebraic comparison is difficult in RNS, which has a non-weighted number representation. Compared with the previous algorithm, the proposed algorithm only requires L moduli which is half the number needed in the previous algorithm. Furthermore, the proposed algorithm reduces the number of MUL operations by 25 %.

1 Introduction

Many cryptosystems employ modular multiplications with very large numbers of more than 512 bits [1]. Various algorithms have been proposed for such operations in previous literature [4] through [14]. Some of these algorithms use a weighted number system to perform the multiplications [12][13]. However, the Residue Number System (RNS) is of particular interest here because of its inherent property that both addition and multiplication can be executed extremely fast without the need for any carry propagation [2-11][14]. A variety of RNS multipliers are already available including pure table lookup multipliers, quarter square multipliers, index transform multipliers, and array multipliers [4][6][7][10].

Since RNS is not a weighted number system where each digit corresponds to a certain weight, magnitude comparison and division are both hard problems. Accordingly, an efficient method for modular reduction is needed for the performance of modular multiplication. Because such a method will require magnitude comparison operations, the RNS would not seem to be well suited for this algorithm. However, if the Mixed Radix number System (MRS) is used in combination with RNS, an efficient modular reduction can be obtained. Furthermore, this association of MRS with RNS can be defined using the same moduli base. A RNS Montgomery modular multiplication algorithm (MMM) was previously proposed in [5] where Montgomery's method applied to MRS. An auxiliary residue system is also used to solve the data loss problem that occurs at each

R. Wyrzykowski et al. (Eds.): PPAM 2003, LNCS 3019, pp. 1028–1033, 2004.

loop. As a result, this algorithm requires $2L$ moduli because of the additional L moduli needed for the auxiliary base.

This paper proposes a parallel modular multiplication algorithm in RNS, which uses a new lookup table in conjunction with MRS and RNS for the modular reduction. MRS is employed because algebraic comparison is difficult in RNS, which has a non-weighted number representation. The proposed algorithm only requires L moduli, which is half the number of moduli used previously, because there is no need for an additional L moduli for the auxiliary base.

2 Number System

RNS is a carry free system and is potentially very fast even though the advantages of the weighted number system do not carry over. However, algebraic comparison, overflow detection, and sign detection are all difficult, and division is awkward. RNS is defined as follows : The vector $m_1, m_2, ..., m_L$ forms a set of moduli where the m_i's are relatively prime; M is the value of the product of $m_1 \times m_2 \times ... \times m_L$; The vector $m_1, m_2, ..., m_L$ is the RNS representation of X, an integer less than M, where $x_i = X \bmod m_i$.

Let the bits for a prime number N be k, denoted by $\text{bit}(N) = k$. Integer A and B are represented using k-1 bits. In order to simplify the analysis, let $k = lw$. For efficiency reasons, each m_i is selected so that each residue x_i is represented using at most w bits or 1 word, i.e., $\text{bit}(x_i) = d_i < w$. The word size w depends on the computer and is usually taken as 8, 16, or 32. Therefore, a RNS representation is constructed by identifying the L pairwise relatively prime m_i, each of which consists of w bits, such that $\text{bit}(M) = Lw \geq 2k$. As $k = sw, L \geq 2s$. The reason for selecting the RNS range as twice the size of the input is related to the need to represent the product result of the two operands uniquely.

The bits for the result of the addition or subtraction cannot be larger than the maximum bits of the operands. However, in the multiplication $C = A \times B$ the bits required for the resulting integer increases. Therefore, the integer needs to be reduced modulo to a prime number N in order to obtain the product $C = A \times B$ in $\text{bit}(C) = k$. This, if residue arithmetic is to be used for the multiplication modulo N, a method needs to be devised to reduce the resulting integer modulo N. Accordingly, a table lookup reduction method is used to perform this reduction, which is described and analyzed in the following sections.

MRS associated with RNS can be defined using the same moduli base. Assuming that $< x_L, x_{L-1}, ..., x_1 >$, $0 \leq x_i < m_i$ is the MRS representation of X, an integer less than M, then $X_{MRS} = x_L m_{L-1} m_{L-2} ... m_1 + ... + x_3 m_2 m_1 + x_2 m_1 + x_1$ where m_i are the radices and x_i are the mixed radix digits and $0 \leq x_i < m_i$. The conversion from a RNS to a weighted representation is based on the mixed radix conversion procedure [2][3]. Given an RNS representation, the Mixed Radix Number Conversion (MRC) algorithm is used to compute X_{MRS}.

[Algorithm 1] Mixed Radix Number Conversion Algorithm
Input : $X_{RNS} = (x_1, x_2, ..., x_L)$
Output : $X_{MRS} =< x'_L, x'_{L-1}, ..., x'_1 \geq x'_L m_{L-1} m_{L-2} ... m_1 + ... + x'_2 m_1 + x'_1$
Auxiliary : $\overline{m_i} = \frac{1}{m_i} mod m_j, j = i+1, ..., L$
Step 1. for $i = 1$ to L-1 do
Step 2. $x'_i = x_i$
Step 3. $X_{RNS} = X_{RNS} - x_i \bmod m$
Step 4. $X_{RNS} = X_{RNS} * \overline{m_i} \bmod m$
Step 5. $x'_L = x_L$
Step 6. return $X_{MRS} =< x'_L, x'_{L-1}, ..., x'_1 >$

For efficient expression, the symbol $< >$ denotes that the enclosed digits are mixed radix digits, whereas ( ) denotes residue digits.

3 Table Lookup Method for Modular Reduction

To construct the reduction table, first, using a prime integer N, bit$(N) = k$, all multiples with bits less than $k+w$ are computed. Consider the set Q_w and I_w of all integers of length w as $Q_w = I_w = \{0, 1, 2, ..., 2^w - 1\}$. Let $q_i \in Q_w$ and $i \in I_w$. q_i is determined by integer i as $\sum_{j=0}^{w-1} i_j x^j$. The table T which contains 2^w rows can be constructed for $V_i = q_i N$ for $i \in I_w$. The most significant word of V_i is then used for the index of the table T. It is important that the most significant words of V_i for all $i \in I_w$ are all unique.

MRS is a weighted number system and hence a magnitude comparison can be easily performed. The conversion from a RNS to a particular MRS is relatively fast in residue computers. Table T, which stores the RNS and MRS forms of the multiples of N is used for reducing the integer modulo N. Let the multiplication result C be the number of length $lw + w$ denoted by $C =< c_l c_{l-1} ... c_1 c_0 >$ which is to be reduced. To reduce the length of the result to lw, the reduction algorithm computes C mod N. In order to reduce the result, the entry $< v_{i,l-1} v_{i,l-2} ... v_{i,1} v_{i,0} >$ from the table T is then selected as follows :

$C =< c_l c_{l-1} ... c_1 c_0 > - < v_{i,l-1} v_{i,l-2} ... v_{i,1} v_{i,0} >=< c'_{l-1} c'_{l-2} ... c'_1 c'_0 >$
where $c'_j = c_j - v_j$ for $0 \leq j < l - 1$.

A table for the modular reduction can then be constructed by taking a number q_i from Q_w, multiplying it by N to obtain $V_i = q_i N$, and then placing V_i in the table using the most significant word as the index. The relation table between an MRS and an RNS representation, T_{MRS} and T_{RNS}, is shown in Table 1. All of the most significant words are zeros of the T_{MRS} in Table 1. The T_{MRS} can then be reduced to T'_{MRS}, which uses the second word of T_{MRS} as the index.

4 Modular Multiplication Algorithm

This section presents a new parallel modular multiplication algorithm on RNS using the table lookup reduction method. The parallel modular multiplication

Table 1. Table T for modular reduction.

q_i, i	T_{MRS}	T_{RNS}	T_{MRS}
0	<0,0,0,0>	(0, 0, 0, 0)	< 0, 0 >
1	<0,1,0,6>	(6, 6, 9, 3)	< 0, 6 >
2	<0,2,0,12>	(12,12, 2, 6)	< 0,12 >
3	<0,3,1,5>	(5, 3,11, 9)	< 1, 5 >
4	<0,4,1,11>	(11, 9, 4, 1)	< 0,11 >
5	<0,5,2,4>	(4, 0,13, 4)	< 2, 4 >
6	<0,6,2,10>	(10, 6, 6, 7)	< 2,10 >
7	<0,7,3,3>	(3,12,15,10)	< 3, 3 >
8	<0,8,3,9>	(9, 3, 8, 2)	< 3, 9 >
9	<0,9,4,2>	(2, 9, 1, 5)	< 4, 2 >
10	<0,10,4,8>	(8, 0,10, 8)	< 4, 8 >
11	<0,11,5,1>	(1, 6, 3, 0)	< 5, 1 >
12	<0,12,5,7>	(7,12,12, 3)	< 5, 7 >
13	<0,13,6,0>	(0, 3, 5, 6)	< 6, 0 >
14	<0,14,6,6>	(6, 9,14, 9)	< 6, 6 >
15	<0,15,6,12>	(12, 0, 7, 1)	< 6,12 >

algorithm is given as following algorithm 2.

[Algorithm 2] Parallel Modular Multiplication Algorithm based on RNS
Input : $A_{RNS} = (a_1, a_2, ..., a_L), B_{RNS} = (b_1, b_2, ..., b_L)$
Output : $C_{RNS} = (c_1, c_2, ..., c_L)$
Auxiliary : $T, MAX[i]$
Step 1. $C_{RNS} = A_{RNS} * B_{RNS}$
Step 2. $M_{MRS} = MRC(C_{RNS})$
Step 3. for $i = 2l - 1$ downto l do
Step 4. $C_{RNS} = C_{RNS} - T[x'_i] * MAX[i]$
Step 5. $X_{MRS} = X_{MRS} - T[x'_i] \times MAX[i]$
Step 6. return C_{RNS}

Let A and B be two input with at most $k-1$ bits. In Step 1, each digit in a RNS representation is multiplied in parallel. After Step 1, C is represented at most $2k-2$ bits. The representation is still unique in RNS because the bits for required M are at least $2k$ and the MRC algorithm will yield a unique result. However, the result C cannot be used as an input for next multiplications. Thereby, a modular reduction operation is necessary. Steps 2-5 can achieve this operation. First the MRS representation, X_{MRS} for a weighted number system is computed from C using algorithm 1. Then the table lookup reduction makes bit(C) becomes less than k. The array element $MAX[2l-1]$ at Step 4 and 5 stores the maximaximum modulus, $\max(m_i)$, $i = 1, 2, ..., L$ and the others have

1. The symbol * denotes that the multiplication operation is computed on RNS, whereas $\times$ denotes on MRS.

Table 2. Comparison

	J. C. Bajard in [5]	Proposed
TREAD	0	$2l$
MUL	$4l$	$3l+1$
XOR	$4l$	$4l$
DIV	$2l$	0

5 Analysis

An analysis is made of the proposed algorithm and its performance is compared with that of the MMM algorithm by J. C. Bajard in [5]. The MMM algorithm is based on Montgomery's method using MRS and RNS. Since each loop of the MMM algorithm requires division, the use of RNS alone is very difficult, as such, MRS is used for the division. However, division of the ith residue cannot be computed because m_i is not relatively prime to itself. Thus the ith residue is lost. An auxiliary residue system is therefore used to solve the loss of residue, however, it needs twice the number of moduli, $2L$ moduli, compared with the proposed approach because of the additional L moduli for the auxiliary base. Although the algorithm in [5] does not require table lookup operations, it still needs divisions. In contrast, the table lookup operation was easily implemented with a combinatorial logic of reasonable complexity. Accordingly, the operation counts indicated that the proposed parallel modular multiplication algorithm was computationally more efficient. For the perspective of the table size, table T have 2^w rows containing a number of length ks. Each entry in table T contained MRS and RNS numbers. So, the total table size is $2 \times 2^w \times k$ bits, which are mainly depending on the word size of the computer.

6 Conclusions

This paper proposed a parallel modular multiplication algorithm in RNS through the use of a new lookup table as a tool for modular reduction. MRS is used for the magnitude comparison since MRS is a weighted number system. The proposed algorithm only requires L moduli, which is half the number of moduli used in the MMM algorithm in [5]. From Table 2, the proposed parallel multiplication algorithm reduced the number of MUL and DIV operations by 25% and 100%, respectively. Further work is currently underway on improving the proposed algorithm and implementing hardware design of the proposed parallel modular multiplication algorithm.

References

1. W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Trans. on Info. Theory, vol. IT-22(6), pp. 644-654, 1976.
2. N.S. Szabo, R.I. Tanaka, Residue Arithmetic and Its Applications to Computer Technology, McGraw-Hill, New York, 1967.
3. F.J. Taylor, Residue Arithmetic: A Tutorial with Examples, Computer, pp. 50-62, May 1984.
4. K.M. Elleithy, M.A. Bayoumi, A Systolic Architecture for Modulo Multiplication, IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 42, no. 11, pp. 725-729, Nov. 1995.
5. J.C Bajard, L.S. Didier, P. Kornerup, An RNS Montgomery Modular Multiplication Algorithm, IEEE Trans. on Computers, vol. 47, no. 7, pp. 766-776, July 1998.
6. D. Radhakrishnan, Y. Yuan, Novel Approaches to the Design of VLSI RNS Multipliers, IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 39, no. 1, pp. 52-57, Jan. 1992.
7. G. Alia, E. Martinelli, A VLSI Modulo m Multiplier, IEEE Trans. on Computers, vol. 40, no. 7, pp. 873-878, July 1991.
8. F.J. Taylor, A VLSI Residue Arithmetic Multiplier, IEEE Trans. on Computers, vol. C-31, no. 6, pp. 540-546, June 1982.
9. G.A. Jullien, Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Theoretic Transforms, IEEE Trans. on Computers, vol. C-29, no. 10, pp. 899-905, Oct. 1980.
10. M. Soderstrand, W.K. Jenkins, G.A. Jullian, F.J. Taylor, Residue Number Systems: Modern Applications in Digital Signal Processing, New York, IEEE, 1986.
11. V.S. Dimitrov, G.A. Jullien, W.C. Miller, A Residue Number System Implementation of Real Orthogonal Transforms, IEEE Trans. on Signal Processing, vol. 46, no. 3, pp. 563-570, March 1998.
12. H.S. Kim, S.W. Lee, K.Y. Yoo, Partitioned Systolic Multiplier for $GF(2^m)$, Information Processing Letter, vol. 76, pp. 135-139, 2000.
13. H. S. Kim, Bit-Serial AOP Arithmetic Architecture for Modular Exponentiation, Ph. D. Thesis, Kyungpook National Univ., 2002.
14. A. Halbutogullari, C.K. Koc, Parallel Multiplication in $GF(2^k)$ using Polynomial Residue Arithmetic, Design, Codes and Cryptography, vol. 20, no. 2, pp. 155-173, 2000.