The extended journal version of this paper can be found at: http://www.marcelokaihara.com/papers/BMM_Method.pdf

Bipartite Modular Multiplication

Marcelo E. Kaihara and Naofumi Takagi

Department of Information Engineering, Nagoya University,
Nagoya, 464-8603, Japan
{mkaihara, ntakagi}@takagi.nuie.nagoya-u.ac.jp

Abstract. This paper proposes a new fast method for calculating modular multiplication. The calculation is performed using a new representation of residue classes modulo M that enables the splitting of the multiplier into two parts. These two parts are then processed separately, in parallel, potentially doubling the calculation speed. The upper part and the lower part of the multiplier are processed using the interleaved modular multiplication algorithm and the Montgomery algorithm respectively. Conversions back and forth between the original integer set and the new residue system can be performed at speeds up to twice that of the Montgomery method without the need for precomputed constants. This new method is suitable for both hardware implementation; and software implementation in a multiprocessor environment. Although this paper is focusing on the application of the new method in the integer field, the technique used to speed up the calculation can also easily be adapted for operation in the binary extended field $GF(2^m)$.

1 Introduction

Modular multiplication is one of the basic arithmetic operations that are extensively used in many public-key cryptographic applications, such as RSA [10], ElGamal [5], Diffie-Hellman key exchange [4], DSA [1], and others. Because of its computational intensity, implementation in dedicated hardware is required for high-performance systems. Various techniques for speeding up modular multiplication have been reported in literature. Among them, two major approaches stand out: One is based on the interleaved modular multiplication algorithm where the multiplier is processed from the most significant position [2,3,6,8,11,12]. The other one is based on the Montgomery algorithm where the multiplier is processed from the least significant position [7,9,13,15]. The techniques for speeding up these two approaches have been developed separately.

This paper proposes a method that takes advantage of these techniques and the ones that may eventually be devised in the future, to further boost speed. The key that enables the linking of these two approaches is a new representation of residue classes modulo M. Assuming M is an n-word odd integer, where the radix of each word is $r = 2^k$, this new representation maps an integer U in the range $[0, M-1]$ to the number $U \cdot R \mod M$ in the same range. R is a constant of value $r^{\alpha n}$, coprime to M, where α is a rational number such that $0 < \alpha < 1$,

J.R. Rao and B. Sunar (Eds.): CHES 2005, LNCS 3659, pp. 201–210, 2005.

and, αn is an integer. The novelty in this representation is that the transformation constant R has a value less than the modulus M, a condition not allowed by the Montgomery representation. Modular multiplication is then performed in this new residue system. The new values for the transformation constant enable the splitting of the multiplier into two parts which can then be processed separately, in parallel. The upper part and the lower part of the multiplier can be processed using the interleaved modular multiplication algorithm and the Montgomery algorithm, respectively. The possibility of selecting the parameter α between the values 0 and 1, encompasses the application of this method to all combinations of algorithms of different performance derived from the interleaved modular multiplication algorithm and the Montgomery algorithm. If applied to algorithms with similar performance and the multiplier is split into two equal parts, it is theoretically possible to achieve the maximum speed of twice that of these two algorithms when performed individually. The latter condition is represented with the value of the parameter α so that $\alpha n = \lceil \frac{n}{2} \rceil$.

Two other advantages of this new method are: Firstly, compared to the Montgomery method, conversion speed between the original integer set and the new residue system is potentially doubled; and secondly, precomputation of constants is no longer necessary.

Due to the parallel processing, the proposed method is suitable for hardware implementation and also for software implementation in a multiprocessor environment.

The remainder of this paper is organized as follows: Section 2 reviews the interleaved modular multiplication algorithm and the Montgomery algorithm. The new computation method is introduced in Section 3. Section 4 explains hardware implementation of the method. Section 5 contains our concluding remarks.

2 Preliminaries

2.1 Interleaved Modular Multiplication Algorithm

Given a modulus M, and two elements of the residue class ring of integers modulo M, X and Y, we define the ordinary modular multiplication as:

$$X \times Y \triangleq X \cdot Y \mod M \tag{1}$$

Let the modulus M be an n-word number, where the radix of each word is $r = 2^k$. The i-th word ($i = 0, 1, \cdots, n-1$) of Y is denoted by y_i. Namely, $Y = \sum_{i=0}^{n-1} y_i \cdot r^i$. The interleaved modular multiplication algorithm for calculating the ordinary modular multiplication is shown below [2,3,11].

[Interleaved Modular Multiplication Algorithm]
Input: $M : r^{n-1} < M < r^n$
$X, Y : 0 \leq X, Y < M$
Output: $Z = X \cdot Y \bmod M$

Algorithm:
$Z := 0;$
for $i := n-1$ **downto** 0 **do**
$Z := r \cdot Z + y_i \cdot X;$
$q_C := \lfloor \frac{Z}{M} \rfloor;$
$Z := Z - q_C \cdot M;$
endfor

In this algorithm, the words of the multiplier are processed from the most significant position first.

2.2 Montgomery Multiplication Algorithm

Montgomery introduced a powerful algorithm for calculating modular multiplication where the multiplier is processed from the least significant position first [7]. Given an n-word odd modulus M and an integer U in the range $[0, M-1]$, the image, or the M-residue of U is defined as $X = U \cdot R_M \mod M$ where R_M is a constant coprime to M and $R_M > M$. In order to reduce computation effort, this constant is usually set to the value of r^n. If X and Y are the images of U and V respectively, the Montgomery multiplication of these two images, $X * Y$, is defined as:

$$X * Y \triangleq X \cdot Y \cdot R_M^{-1} \mod M \tag{2}$$

The result is the image of $U \cdot V \bmod M$. If the i-th word of M is denoted as m_i, then $M = \sum_{i=0}^{n-1} m_i \cdot r^i$. In a similar way, if the number that represents the partial products is denoted as $Z = \sum_{i=0}^{n-1} z_i \cdot r^i$, the resulting Montgomery algorithm is described below.

[Montgomery Multiplication Algorithm]
Input: $M : r^{n-1} < M < r^n$ and $\gcd(M, 2) = 1$
$X, Y : 0 \leq X, Y < M$
Output: $Z = X \cdot Y \cdot r^{-n} \bmod M$
Algorithm:
$Z := 0;$
for $i := 0$ **to** $n-1$ **do**
$Z := Z + y_i \cdot X;$
$q_M := (-z_0 \cdot m_0^{-1}) \bmod r;$
$Z := (Z + q_M \cdot M)/r;$
endfor
if $Z \geq M$ **then** $Z := Z - M$

The transformations back and forth between the ordinary representation and the M-residue representation can be performed using the same algorithm provided that the constant $R_M^2 \mod M$ is precomputed. An integer U can be transformed to the M-residue representation by applying the Montgomery algorithm

to this integer and the constant $R_M^2 \mod M$. Transformation of an image X back into the original integer set can be done by applying the Montgomery multiplication algorithm to this image and the number 1.

3 A New Modular Multiplication Method

In this section, a new fast method for calculating modular multiplication is presented. The calculation is performed using a new representation of residue classes modulo M. In contrast to the M-residue representation introduced by Montgomery which requires the constant R_M to be coprime to M and greater in value than M, we have changed the condition of $R_M > M$ and defined a new residue class representation using a new constant $R = r^{\alpha n}$, where R is coprime to M, and, α is a rational number so that $0 < \alpha < 1$ and αn is an integer. The resulting image of an integer U is $X = U \cdot r^{\alpha n} \mod M$. Given X and Y, two images of integers U and V respectively, multiplication modulo M in the new residue system is defined as:

$$X \circledast Y \triangleq X \cdot Y \cdot r^{-\alpha n} \mod M \tag{3}$$

The existence of $r^{-\alpha n} \mod M$ is assured by the relative primalty condition between $r^{\alpha n}$ and M. Since M is odd for cryptographic applications, by utilising $r = 2^k$, the primalty condition is satisfied.

Transformation from the original representation to the new residue system is accomplished by performing conventional modular multiplication between the integer value and the constant $r^{\alpha n}$. The inverse transformation from the new residue system back to the original representation can be performed by multiplying either of the images with the constant $r^{-\alpha n}$ in modulo M, which can be done using the Montgomery algorithm as explained at the end of this section. That the new multiplication modulo M over the images of U and V results in a image of $U \cdot V \mod M$ can easily be demonstrated as follows.

$$\begin{aligned} & X \cdot Y \cdot r^{-\alpha n} \mod M \\ = & (U \cdot r^{\alpha n}) \cdot (V \cdot r^{\alpha n}) \cdot r^{-\alpha n} \mod M \\ = & (U \cdot V) \cdot r^{\alpha n} \mod M \end{aligned} \tag{4}$$

Isomorphism between the original integer set $\mathcal{Z}_{\mathcal{M}}$ with the operation $\times$, and the new residue system $\mathcal{Z}'_{\mathcal{M}}$ with the operation $\circledast$, holds as illustrated in Fig. 1.

As we will now show, modular multiplication can be efficiently computed using this new representation of residue classes. Let us consider the multiplier Y split into two parts Y_H and Y_L, i.e. $Y = Y_H \cdot r^{\alpha n} + Y_L$. Then, the multiplication modulo M of the images X and Y can be computed as follows:

$$X \circledast Y = (X \times Y_H + X \circledast Y_L) \mod M \tag{5}$$

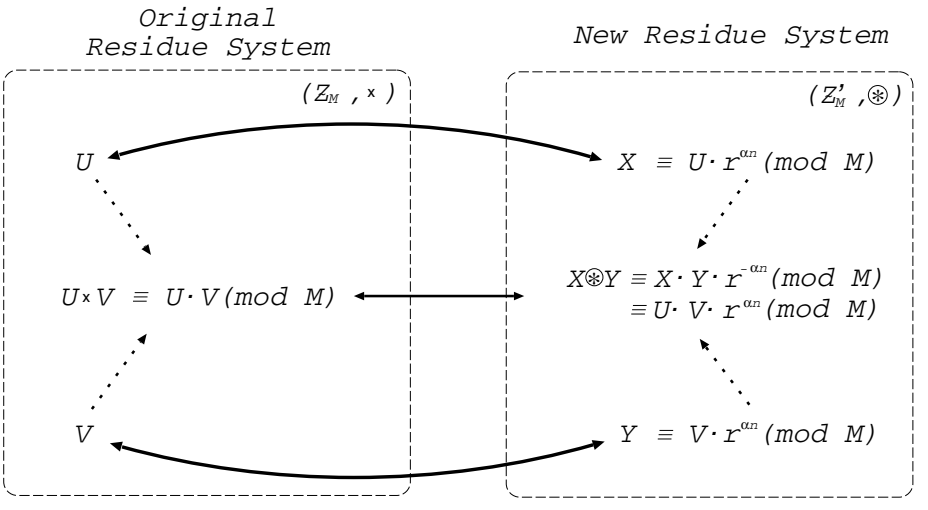

Fig. 1. Mapping between the original residue system and the new residue system

The left term, $X \times Y_H$, is calculated using the interleaved modular multiplication algorithm that processes Y_H from the most significant position first, while the second term, $X \circledast Y_L$, is calculated using the Montgomery algorithm which processes Y_L from the least significant position first. These two calculations are performed in parallel. Since the split operands Y_H and Y_L are shorter in length than Y, the calculations $X \times Y_H$ and $X \circledast Y_L$ are performed faster than the individual execution of the interleaved modular multiplication algorithm and the Montgomery algorithm with the unsplit operands.

The correctness of the above formula can be seen using the following equality:

$$\begin{aligned}
&(X \times Y_H + X \circledast Y_L) \mod M \\
&= (X \cdot Y_H + X \cdot Y_L \cdot r^{-\alpha n}) \mod M \\
&= X \cdot (Y_H \cdot r^{\alpha n} \cdot r^{-\alpha n} + Y_L \cdot r^{-\alpha n}) \mod M \\
&= X \cdot (Y_H \cdot r^{\alpha n} + Y_L) \cdot r^{-\alpha n} \mod M \\
&= X \cdot Y \cdot r^{-\alpha n} \mod M = X \circledast Y
\end{aligned} \tag{6}$$

Below is the algorithm that computes modular multiplication using the new representation. In this algorithm, A is a variable that stores the multiplicand; B_H and B_L are variables that store the upper and the lower parts of the multiplier respectively. $Interleaved\_modmul\ (A, B_H)$ is a function that calculates $A \times B_H$ by using the interleaved modular multiplication algorithm. $Montgomery\_modmul\ (A, B_L)$ is a function that calculates $A \circledast B_L$ by using the Montgomery algorithm. $\{C1; C2; \}$ means that two calculations, $C1$ and $C2$, are performed in parallel.

[Algorithm KT] (New Modular Multiplication)
Input: $M : r^{n-1} < M < r^n$, $\gcd(M, 2) = 1$
$X, Y \in \mathcal{Z}'_{\mathcal{M}}$
Output: $Z = X \cdot Y \cdot r^{-\alpha n} \bmod M \quad (Z \in \mathcal{Z}'_{\mathcal{M}})$
Algorithm:

Step 1: $A := X$; $M := M$; $S := 0$; $T := 0$;
$B_H := Y_H$; $B_L := Y_L \quad /* \, Y = Y_H \cdot r^{\alpha n} + Y_L \, */$

Step 2: $\{S := Interleaved\_modmul(A, B_H)$;
$T := Montgomery\_modmul(A, B_L); \}$

Step 3: $Z := (S + T) \bmod M$;

When using the interleaved modular multiplication algorithm and the Montgomery algorithm of similar performance, α can be set to the value so that $\alpha n = \lceil \frac{n}{2} \rceil$; the split two parts of the multiplier, Y_H and Y_L, are at most $\lceil \frac{n}{2} \rceil$-words wide. This means that, it is theoretically possible to obtain a maximum acceleration of twice the speed of the original algorithms performed individually, when these conditions are met. Fig. 2 shows the multiplication procedure with the parameter $\alpha = 1/2$.

Transformation of an integer U from the original integer set to the new residue system can be performed by executing $X = Interleaved\_modmul(U, r^{\alpha n})$. The inverse transformation of an image X from the new residue class representation back to the original integer set is accomplished by executing $U = Montgomery\_modmul(X, 1)$. When α is set to the value so that $\alpha n = \lceil \frac{n}{2} \rceil$, either of these transformations can be completed theoretically in half the time required by the Montgomery method without the need for precomputed constants.

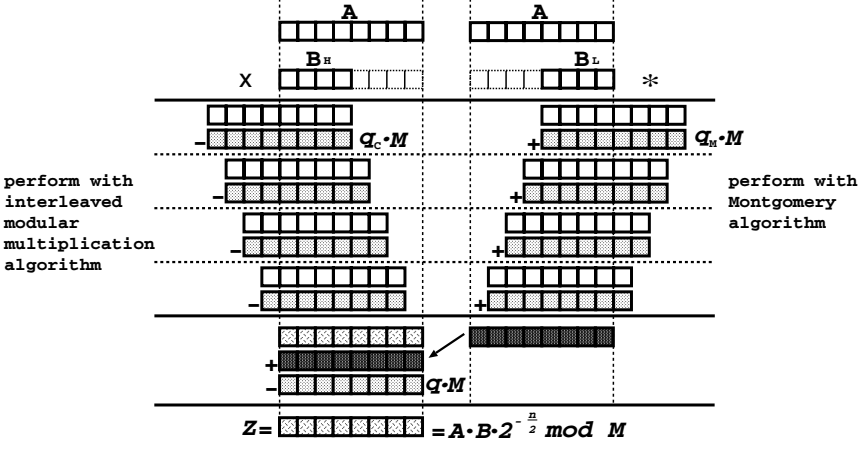

Fig. 2. New Modular Multiplication with $\alpha = 1/2$

4 Hardware Implementation

A modular multiplier based on the algorithm presented in the previous section consists of six registers, an interleaved modular multiplier, a digit-serial Montgomery multiplier, a modular adder, and a multiplexor. The registers are: A, which stores the multiplicand; B_H and B_L which are shift registers and store the upper and lower parts of the multiplier respectively; M, which stores the modulus M; and, S and T, which store the partial results. A block diagram of this hardware is shown in Fig. 3.

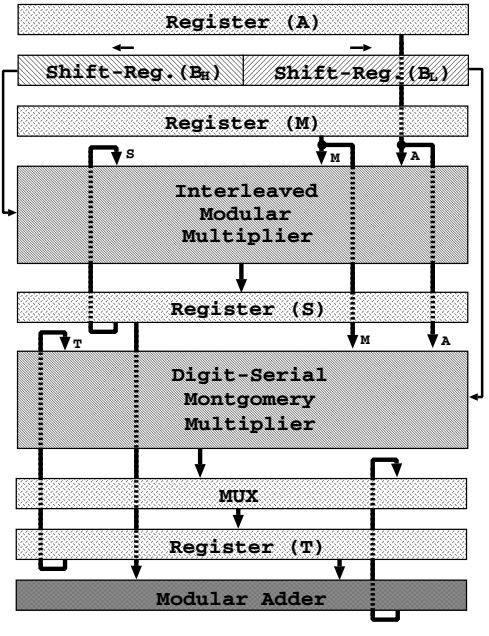

Fig. 3. Block diagram of a multiplier

Various implementations of the interleaved modular multiplier and the Montgomery multiplier are possible depending on the techniques used for speeding up the calculation. Most of these techniques use redundant representation and increase the radix, and the different combinations of the multipliers allow for a wide range of trade-offs between speed and hardware requirements.

When a radix-r interleaved modular multiplier is jointly used with a radix-r Montgomery multiplier with similar critical path delays, and n is even, the parameter α can be set to the value $1/2$ and registers of equal length can be used for B_H and B_L, thus halving the processing time compared to an individual execution of the interleaved modular multiplier or the Montgomery multiplier with the unsplit operands.

Transformation from the ordinary integer set to the new residue class representation can be performed with the same hardware provided that the hardware module which computes the interleaved modular multiplication iterates one extra cycle compared to that required for modular multiplication. Inverse transformation from the new residue class representation back to the original integer set can be performed using the hardware module that computes the Montgomery multiplication.

The value of the parameter α can be displaced around $1/2$ enabling the use of multipliers of different performance. In this case, the multiplier is then split into two unequal parts that can be stored in two registers B_H and B_L of different length. The value of α can be determined from the difference of performance between the multipliers. For example, if a radix-2 interleaved modular multiplier is used with a radix-4 Montgomery multiplier with similar critical path delays, then α can be set to a number where $\alpha n = \lceil \frac{2}{3} n \rceil$. Then, modular multiplication can be accomplished in about $\lceil \frac{n}{3} \rceil$ clock cycles. Transformation from the original integer set to the new residue system, can be performed with the same hardware by executing the interleaved modular multiplication module twice. In the first execution, an integer U is multiplied by $2^{\lceil \frac{n}{3} \rceil}$. In the second execution, the result of the first execution is multiplied by $2^{\lceil \frac{2}{3} n \rceil - \lceil \frac{n}{3} \rceil}$. Inverse transformation from the new residue system to the original integer set can be performed by the same hardware by executing the Montgomery multiplication module once.

The amount of hardware of the proposed multiplier is proportional to n. Compared to an individual interleaved modular multiplier or a Montgomery multiplier, the new modular multiplier requires an extra digit modular multiplier, an extra register, a modular adder and related multiplexors.

The space and time trade-offs for high radix modular multiplications based on the classical interleaved algorithm and the Montgomery algorithm are detailed in [14]. For both algorithms, increasing k, i.e. the number of bits of the radix, to values greater than $log(n)$, where n is the number of words, results in a penalty in time for producing the quotient bits q_C or q_M for the next modular reduction in time that makes this approach unattractive. By contrast, by using our algorithm, a speedup can be achieved for such values of radices, since the multiplication and the modular reduction for the split multiplier can be performed by two separate high-radix digit-serial multipliers processing in parallel. Thus, the number of iteration is reduced without increasing the time requirements for each cycle.

Furthermore, as there is no need to increase the radix, the design can remain relatively simple compared to hardware designed using higher radix.

In cryptographic applications, such as in RSA, this approach is much more attractive than implementing two modular multiplier processors separately because it has the advantage of producing the outputs sequentially.

5 Concluding Remarks

In this paper, we have presented a fast method for computing modular multiplication. We have defined a new residue class representation which enables the

splitting of the multiplier into two parts which can be processed by using the interleaved modular multiplication algorithm and the Montgomery algorithm in parallel, potentially doubling the speed. Transformations back and forth between the original integer set and the new residue system can be performed at a maximum of twice the speed of the Montgomery method without the need for precomputed constants. The dual processing makes it suitable for software implementation in a multiprocessor environment as well as for hardware implementation as discussed in Section 4. Finally, although this paper is focused on application in the integer field, the technique used to speed up the calculation in the proposed method can also easily be adapted for accelerating modular multiplication in the binary extended field $GF(2^m)$.

Acknowledgments

The authors would like to thank Associate Professor Kazuyoshi Takagi for his valuable comments and discussions and to Miss Grith Christensen for her help in preparing this paper.

References

1. ANSI X9.30. Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA). American National Standard Institute. American Bankers Association. 1997.
2. G. R. Blakley, "A Computer Algorithm for Calculating the Product *AB* Modulo *M*," *IEEE Trans. Computers*, vol. C-32, no. 5, pp. 497–500, May 1983.
3. E. F. Brickell, "A fast modular multiplication algorithm with application to two key cryptography," in *Advances in Cryptology, Proc. CRYPTO'82*, D. Chaum *et al.*, Eds. New York: Plenum, 1983, pp. 51–60.
4. W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 11, pp. 644–654, Nov. 1976.
5. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Information Theory*, vol. IT-31, no. 4, pp. 469–472, July 1985.
6. P. Kornerup, "High-Radix Modular Multiplication for Cryptosystems," *Proc. 11th IEEE Symp. Computer Arithmetic*, G. Jullien, M. J. Irwin, and E. Swartzlander, eds., pp. 277-283, Windsor, Canada, 1993.
7. P. L. Montgomery, "Modular Multiplication without Trial Division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519–521, Apr. 1985.
8. H. Morita, "A fast modular multiplication algorithm with application to two key cryptography," in *Lecture Notes in Computer Science*, vol. 435, G. Brassard Ed., *Advances in Cryptology – CRYPTO'89 Proc.* Berlin, Germany: Springer-Verlag, 1990, pp. 387–399.
9. H. Orup, "Simplifying quotient determination in high-radix modular multiplication," *Proc. 12th IEEE Symp. Computer Arithmetic*, S. Knowles and W. H. McAllister, eds., pp. 193-199, Bath, England, 1995.
10. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.

11. K. R. Sloan, "Comments on 'A Computer Algorithm for Calculating the Product AB Modulo M'," *IEEE Trans. Computers*, vol. C-34, no. 3, pp. 290–292, March 1985.
12. N. Takagi, "A radix-4 modular multiplication hardware algorithm for modular exponentiation," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 949–956, Aug. 1990.
13. A. F. Tenca, G. Todorov, and Ç. K. Koç, "High-Radix Design of a Scalable Modular Multiplier," *Cryptographic Hardware and Embedded Systems - CHES 2001*, Ç. K. Koç, D. Naccache, C. Paar, eds., pp. 185–201, Paris, France, 2001.
14. C. D. Walter, "Space/Time Trade-offs for Higher Radix Modular Multiplication using Repeated Addition," *IEEE Trans. Computers*, vol. 46, no. 2, pp. 139–141, Feb. 1997.
15. C. D. Walter, "Systolic Modular Multiplication," *IEEE Trans. Computers*, vol. 42, no. 3, pp. 376–378, Mar. 1993.