AAECC (2010) 21:249–255
DOI 10.1007/s00200-010-0124-2

Highly parallel modular multiplication in the residue number system using sum of residues reduction

Braden J. Phillips · Yinan Kong · Zhining Lim

Received: 21 June 2008 / Revised: 13 February 2009 / Published online: 7 March 2010

Abstract A new algorithm for modular multiplication in the residue number system (RNS) is presented. Modular reduction is performed using a sum of residues. As all of the residues can be evaluated simultaneously, the algorithm permits a highly parallel implementation and is suitable for performing public-key cryptography operations with very low latency.

Keywords Modular multiplication · Public-key cryptography · Residue number systems · RNS

1 Introduction

We consider the multiplication $C = A \times B$ followed by the modular reduction $C \bmod N = \langle C \rangle_N$ where A, B and N are all long integers. This operation is critical for public-key cryptosystems including RSA [10] and Elliptic Curve Cryptography (ECC) over a prime finite field [4]. Operands larger than 1024-bits are common for RSA; for ECC operands larger than 128-bits are common. In this paper we describe an algorithm for modular multiplication within the Residue Number System (RNS) [11]. Our goal is a highly parallel algorithm which can be exploited by a massively parallel implementation to perform public-key operations with exceptionally low latency.

This research was supported under the Australian Research Council's Discovery Projects funding scheme (project number DP0559582).

B. J. Phillips (✉) · Y. Kong · Z. Lim
School of Electrical & Electronic Engineering, The University of Adelaide, North Terrace, Adelaide, SA 5005, Australia
e-mail: phillips@eleceng.adelaide.edu.au

Existing algorithms for RNS modular multiplication [1,3,5,8] are based on Montgomery reduction [7]. Montgomery reduction of C modulo N proceeds by adding a multiple of N according to $\langle C\rangle_R$. This approach works well in the RNS as $\langle C\rangle_R$ can be found easily for some values of R. It can also be carefully arranged so that most operations only occur in half of the RNS channels [1,5].

Outside of the RNS, in a positional number system, Montgomery's algorithm is only one of the published alternatives. Early publications reduced C by finding a sum of residues modulo N [2,6,9]. If $C = \sum_i C_i$ then we can write $\sum_i \langle C_i\rangle_N \equiv C \pmod{N}$. This approach has the advantage that all of the residues $\langle C_i\rangle_N$ can be evaluated in parallel.

1.1 Contribution

Section 3 presents a new algorithm for modular multiplication in the RNS. Unlike previously published work, the new algorithm uses a sum of modular residues to perform modular reduction. Comparisons with existing algorithms presented in Sect. 4.2 show that the new algorithm has fewer dependent sequential steps, but more steps in total. Hence it is appropriate for very-high performance implementations based on parallel hardware.

2 Residue number systems

A Residue Number System is characterized by a set of n co-prime moduli $\{m_1, m_2, \ldots, m_n\}$. A number X is represented in the RNS by its residues with respect to the co-prime moduli:

$$X = [x_1, x_2, \ldots, x_n] = [\langle X\rangle_{m_1}, \langle X\rangle_{m_2}, \ldots, \langle X\rangle_{m_n}].$$

Within the RNS there is a unique representation for each integer in the range $0 \leq X < M$ where $M = \prod_{i=1}^{n} m_i$. M is known as the dynamic range of the RNS.

An advantage of RNS is that addition, subtraction and multiplication can be concurrently performed on the n residues within n parallel channels. For example, given RNS representations of A and B,

$$\begin{aligned} A &= [a_1, a_2, \ldots, a_n] = [\langle A\rangle_{m_1}, \langle A\rangle_{m_2}, \ldots, \langle A\rangle_{m_n}] \\ B &= [b_1, b_2, \ldots, b_n] = [\langle B\rangle_{m_1}, \langle B\rangle_{m_2}, \ldots, \langle B\rangle_{m_n}], \end{aligned}$$

the product $C = A \times B$ can be performed as

$$\begin{aligned} C &= [\langle a_1 \times b_1\rangle_{m_1}, \langle a_2 \times b_2\rangle_{m_2}, \ldots, \langle a_n \times b_n\rangle_{m_n}] \\ &= [c_1, c_2, \ldots, c_n] = [\langle C\rangle_{m_1}, \langle C\rangle_{m_2}, \ldots, \langle C\rangle_{m_n}]. \end{aligned}$$

3 Sum of residues reduction

To derive an RNS algorithm for sum of residues reduction we begin with the Chinese remainder Theorem [11]:

$$\begin{aligned} C &= \sum_{i=1}^{n} \langle c_i M_i^{-1} \rangle_{m_i} M_i \bmod M \\ &= \sum_{i=1}^{n} \gamma_i M_i \bmod M \\ &= \sum_{i=1}^{n} \gamma_i M_i - \alpha M \end{aligned} \tag{1}$$

where $M_i = M/m_i$, $\gamma_i = \langle c_i M_i^{-1} \rangle_{m_i}$ and α is an unknown correction term.

Reducing Eq. 1 modulo N gives

$$\begin{aligned} Z &= \sum_{i=1}^{n} \gamma_i \langle M_i \rangle_N - \langle \alpha M \rangle_N \\ &= \sum_{i=1}^{n} \mathcal{C}_i - \langle \alpha M \rangle_N \\ &\equiv C \pmod{N} \end{aligned} \tag{2}$$

where we have defined $\mathcal{C}_i = \gamma_i \langle M_i \rangle_N$. Thus C can be reduced modulo N using a sum of the residues $\mathcal{C}_i$ and a correction factor $\langle \alpha M \rangle_N$. Note that if we define $\Gamma = [\gamma_1, \ldots, \gamma_n]$ and $\hat{M} = [\langle M_1^{-1} \rangle_{m_1}, \ldots, \langle M_n^{-1} \rangle_{m_n}]$ then $\Gamma = C \times \hat{M} \bmod M$. Hence we can find the γ_i terms using a single RNS multiplication by a precomputed constant.

3.1 Finding α

A solution for the unknown α in Eq. 2 is provided by [5]. This assumes the RNS moduli are ordered such that $m_i < m_j$ for all $i < j$ and that the moduli are each k bits long such that $2^{k-1} \le m_i < 2^k$.

Equation 6 from [5] gives an estimate for α as

$$\hat{\alpha} = \left\lfloor \sum_{i=1}^{n} \frac{\left\lfloor \frac{\gamma_i}{2^{k-q}} \right\rfloor}{2^q} + \Delta \right\rfloor . \tag{3}$$

In this equation Δ is a fixed-point correction term and q is an integer parameter which defines the number of bits truncated from the γ_i terms in the sum. Theorem 1 from [5] states that this estimate will be exact, so that $\alpha = \hat{\alpha}$, when we choose Δ and q to satisfy

$$0 \le n(\epsilon + \Delta) \le \Delta < 1$$
$$0 \le C < (1 - \Delta)M$$

given

$$\epsilon = \max\left(\frac{2^k - m_i}{2^k}\right) = \frac{2^k - m_1}{2^k}$$
$$\delta = \max\left(\frac{\gamma_i - \left\lfloor \frac{\gamma_i}{2^{k-1}} \right\rfloor \times 2^{k-q}}{m_i}\right) = \frac{2^{k-q} - 1}{m_1}.$$

3.2 Bounds

Note that the result Z in Eq. 2 may be greater than the modulus N and would require subtraction of a multiple of N to be fully reduced. Instead we can ensure that the dynamic range M of the RNS is large enough that the results of modular multiplications can be used as operands for subsequent modular multiplications without overflow.

Given that $\gamma_i < m_i < 2^k$, $\langle M_i \rangle_N < N$ and $\langle \alpha M \rangle_N \ge 0$ we know

$$Z = \sum_{i=1}^{n} \gamma_i \langle M_i \rangle_N - \langle \alpha M \rangle_N < n 2^k N.$$

So we take operands $A < n2^k N$ and $B < n2^k N$ such that $C = A \times B < n^2 2^{2k} N^2$.

We also need to ensure that C does not overflow M. If we assume that N can be represented in h channels so that $N < 2^{kh}$, then $C < n^2 2^{2hk+2k}$. We require $C < M$ for $M > 2^{n(k-1)}$ which will be satisfied if

$$n > 2h + 2 + \frac{2\log_2 n + n}{k}. \tag{4}$$

For example, for $k \ge 32$ and $n < 128$, it will be sufficient to choose $n \ge 2h + 7$. Note that this bound is conservative and fewer channels may be required for a particular RNS.

3.3 The algorithm

The preceding results are expressed as an algorithm for modular multiplication in Algorithm 1.

Algorithm 1 Sum of residues RNS modular multiplication

```
Require: $k, N, n, h, \Delta, q, [m_1, \ldots, m_n], \hat{M}$ as described above.
Require: precomputed table $\langle M_i \rangle_N$ for $i = 1 \ldots n$
Require: precomputed table $\langle \alpha M \rangle_N$ for $\alpha = 1 \ldots n$
Require: $A < n2^k N$, $B < n2^k N$
Ensure: $Z \equiv A \times B \bmod N$, $Z < n2^k N$
1: $C := A \times B$                                    ▷ 1 RNS multiplication in $n$ channels
2: $\Gamma := C \times \hat{M}$                          ▷ 1 RNS multiplication in $n$ channels
3: loop for $i = 1$ to $n$
4:     $\mathcal{C}_i := \gamma_i \times \langle M_i \rangle_N$   ▷ 1 RNS multiplication in $n$ channels
5: end loop
6: $Z := \sum_{i=1}^{n} \mathcal{C}_i$                   ▷ $(n-1)$ RNS additions in $n$ channels
7: $\alpha := \left\lfloor \sum_{i=1}^{n} \left\lfloor \frac{\gamma_i}{2^{k-q}} \right\rfloor / 2^q + \Delta \right\rfloor$   ▷ sum of $n$ $q$-bit terms
8: $Z := Z - \langle \alpha M \rangle_N$                 ▷ 1 RNS addition in $n$ channels
```

4 Discussion

4.1 An example

To demonstrate the applicability of this scheme to, for example, 1024-bit RSA cryptography [10], consider the set of consecutive 32-bit prime moduli $[m_1, m_2, \ldots, m_{69}] = [4294965131, 4294965137, \ldots, 4294966427]$. For $h = 33$, $\left\lfloor \log_2 \left(\prod_{i=1}^{h} m_i \right) \right\rfloor = 1055$ and the RNS has over 1024 bits of dynamic range for the modulus N. Choosing $n = 69$ gives $\lfloor \log_2(M) \rfloor = 2207$ bits of dynamic range. The maximum value of the intermediate product C is $\left(\sum_{i=1}^{n} (m_i - 1)(\prod_{i=1}^{h} m_i - 1) \right)^2$ such that $\lfloor \log_2(C) \rfloor \leq 2188$ and $C < M$ as required.

Selecting $q = 7$ and $\Delta = 0.75$ ensures $0 \leq n(\epsilon + \Delta) \leq \Delta < 1$ and $0 \leq C < (1 - \Delta)M$ as required for exact determination of α.

4.2 Evaluation

If there are sufficient hardware resources, the residues $\mathcal{C}_i$ in the algorithm above can all be computed in parallel. Also, the sum $Z := \sum_{i=1}^{n} \mathcal{C}_i$ can occur in $(n - 1)$ sequential steps as indicated above, or an accumulation tree can be used to reduce the number of steps. For example, a binary tree would require only $\lceil \log_2 n \rceil$ sequential steps. The correction term α is not required until the last step and can be computed in parallel with the sum of the residues.

Precomputed tables are required for $\langle M_i \rangle_N$ and $\langle \alpha M \rangle_N$ but the memory for these tables is not excessive. Each table contains n entries, where each entry is an RNS constant in n channels.

Table 1 compares the number of operations for the new algorithm with RNS Montgomery multiplication algorithms due to [1] and [5]. The table shows: the number of sequential RNS steps assuming sufficient hardware to execute steps in parallel wherever possible; the total number of RNS operations; and the total number of chan-

Table 1 Operation counts for the new algorithm compared with RNS Montgomery multiplication due to [1] and [5]

Operation	New	Bajard	Kawamura
Min. sequential RNS steps			
RNS multiplications	3	9	10
RNS additions	$2h + a$	$2h$	$2h + 3$
Total RNS operations			
Multiplications in $2h + a$ channels	$2h + a + 2$	0	0
Additions in $2h + a$ channels	$2h + a$	0	0
Multiplications in h channels	0	$2h + 8$	$2h + 9$
Additions in h channels	0	$2h$	$2h + 1$
Multiplications in 1 channel	0	$2h + 4$	0
Additions in 1 channel	0	$2h$	0
Total channel-width operations			
Modular multiplications	$(2h + a)(2h + a + 2)$	$2h^2 + 10h + 4$	$2h^2 + 9h$
Modular additions	$(2h + a)^2$	$2h^2 + 2h$	$2h^2 + h$
q-bit additions	$2h + a$	0	$2h$

The modulus is h channels long in every case such that $2^{(k-1)(h-1)} < N < 2^{kh}$. For the new algorithm we take $n = 2h + a$. The term a varies according to the RNS but in the 1024-bit example above $a = 3$

nel-width operations. The new algorithm can complete in fewer parallel RNS steps than these predecessors but involves more channel-width operations.

5 Conclusion

Using RNS for modular multiplication provides a significant degree of independence across the RNS channels; using sum of residues reduction makes many of the RNS operations down the channels independent. By combining these techniques we have developed an algorithm for modular multiplication with very few dependent sequential steps. Where sufficient hardware is available, it can complete in the time required for 3 channel-width modular multiplications and the modular accumulation of $2h + a$ channel-width terms. Here h is the minimum number of channels required to represent the modulus and a is a small integer term (around 3). It should be possible to exploit this algorithm with massively parallel hardware for an extremely low latency implementation of RSA cryptography.

References

1. Bajard, J.C., Didier, L.S., Kornerup, P.: Modular multiplication and base extensions in residue number systems. In: Proceedings of 15th IEEE Symposium on Computer Arithmetic, vol. **2**, pp. 59–65 (2001)
2. Findlay, P.A., Johnson, B.A.: Modular exponentiation using recursive sums of residues. In: Proceedings of Advances in Cryptology—Crypto 89, Lecture Notes in Computer Science, vol. **435**, pp. 371–386 (1990)

3. Freking, W.L., Parhi, K.K.: Modular multiplication in the residue number system with application to massively-parallel public-key cryptography systems. In: Proceedings of 34th Asilomar Conference on Signals, Systems and Computers, vol. **2**, pp. 1339–1343 (2000)
4. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer-Verlag, New York (2004)
5. Kawamura, S., Koike, M., Sano, F., Shimbo, A.: Cox-rower architecture for fast parallel montgomery multiplication. In: Advances in Cryptology—Eurocrypt 2000, Lecture Notes in Computer Science, vol. **1807**, pp. 523–538 (2000)
6. Kawamura, S.I., Hirano, K.: A fast modular arithmetic algorithm using a residue table. In: Advances in Cryptology—Eurocrypt 88, Lecture Notes in Computer Science, vol. **330**, pp. 245–250 (1988)
7. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**(170), 519–521 (1985)
8. Posch, K.C., Posch, R.: Modular reduction in residue number systems. IEEE Trans. Parall. Distrib. Syst. **PDS-6**(5), 449–453 (1995)
9. Quisquater, J.J., Couvreur, C.: Fast decipherment algorithm for RSA public-key cryptosystem. Electron. Lett. **18**(21), 905–906 (1982)
10. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
11. Szabo, N.S., Tanaka, R.I.: Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill, New York (1967)