ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.11, No.19, 1–5

Enhancement of a modified radix-2 Montgomery modular multiplication

Se-Hyu Choi and Keon-Jik Lee[a)]

School of Architectural, Civil, Environmental and Energy Engineering, Kyungpook National University, Daegu, 702–701, Korea

a) *LeeMeeKael@gmail.com*

Abstract: Recently, Manochehri et al. proposed a modified radix-2 Montgomery modular multiplication with a new recording method. In this letter, we present an improvement to their scheme that makes it simpler and faster. Manochehri et al.'s algorithm requires $n + 2$ iterations, whereas the proposed (non-pipelined) algorithm requires $n + 2$ iterations. Moreover, there is no need for post-processing to obtain the correct output, nor for a non-standard operation such as bitwise subtraction. The area/time complexity of our pipelined multiplier is reduced by approximately 24.36% compared to Manochehri et al.'s multiplier. The proposed architecture is simple, modular, and regular. Moreover, it exhibits low complexity and propagation delay. Accordingly, it is well suited for VLSI implementation.

Keywords: Montgomery, modular multiplication, radix-2, carry save adder

Classification: Integrated circuits

References

[1] P. L. Montgomery: Math. Comput. **44** (1985) 519. DOI:10.1090/S0025-5718-1985-0777282-X

[2] C. D. Walter: Electron. Lett. **35** (1999) 1831. DOI:10.1049/el:19991230

[3] K. J. Lee and K. Y. Yoo: Inf. Process. Lett. **76** (2000) 105. DOI:10.1016/S0020-0190(00)00131-9

[4] K. J. Lee, K. W. Kim and K. Y. Yoo: Inf. Process. Lett. **82** (2002) 65. DOI:10.1016/S0020-0190(01)00249-6

[5] K. J. Lee and K. Y. Yoo: Integr. VLSI J. **32** (2002) 99. DOI:10.1016/S0167-9260(02)00044-5

[6] C. McIvor, M. McLoone and J. V. McCanny: IEE Proc. Comput. Digit. Tech. **151** (2004) 402. DOI:10.1049/ip-cdt:20040791

[7] D. S. Lim, N. S. Chang, S. Y. Ji, C. H. Kim, S. J. Lee and Y. H. Park: J. Syst. Archit. **55** (2009) 355. DOI:10.1016/j.sysarc.2009.04.001

[8] K. Manochehri, B. Sadeghian and S. Pourmozafari: IEICE Electron. Express **7** (2010) 513. DOI:10.1587/elex.7.513

[9] H. Orup: Proc. Int. Symp. on Computer Arithmetic (1995) 193. DOI:10.1109/ARITH.1995.465359

DOI: 10.1587/elex.11.20140782
Received August 13, 2014
Accepted August 18, 2014
Publicized September 10, 2014
Copyedited October 11, 2014

1 Introduction

In a public key cryptosystem (PKC), the core operation is modular exponentiation (ME), which is accomplished by repeated modular multiplications (MM). Because the modulus is at least 1024 bits in size, high throughput is difficult to achieve without the use of hardware acceleration. The majority of the contributions are based on the Montgomery's modular multiplication (MMM), which replaces trial division by the modulus with a series of additions and divisions by a power of two [1, 2, 3, 4, 5, 6, 7, 8, 9]. Thus, it is well suited to hardware implementation. However, a potential difficulty in its computation may come from addition of long operands. If addition is computationally costly, all other related operations will be affected accordingly. If a redundant number system or systolic array system is employed, then the carry-propagation chain can be eliminated [3, 4, 5, 7]. Recently, Manochehri et al. presented MMM with a new recording method in which one step of the main loop is removed [8]. However, after the main loop, post-processing is necessary to obtain the correct result. In this letter, we propose an improved MMM algorithm that does not require post-processing.

2 MMM algorithm

2.1 Montgomery's algorithm

Using an ingenious representation of the residue class modulo N, the Montgomery multiplication of A ($= aR \bmod N$) and B ($= bR \bmod N$) is given by $\mathrm{MMM}^1(A, B) = ABR^{-1} \bmod N$, where N is an n-bit odd integer, $R = 2^n$, and A (B) is the N-residue of an integer a (b), for $a, b, A, B \in [0, N)$.

MMM^1(A, B)
1: $T = 0$;
2: for $i = 0$ to $n - 1$ {
3: $\quad Q[i] = (T + A[i] \times B[0]) \bmod 2$;
4: $\quad T = (T + A[i] \times B + Q[i] \times N) \operatorname{div} 2$;
5: }
6: if $(T \geq N)$ $T = T - N$;

where $A[i]$ denotes the i-th bit of A. Because $T < 2N(1/2 + 1/4 + \cdots + 1/2^{n-1})$, i.e., $T \in [0, 2N)$, after executing the loop n times, an extra subtraction is needed if $T \geq N$.

2.2 Walter's MMM

Walter proposed the systolic MMM, where $A, B, T \in [0, 2N)$ and the number of iterations becomes $n + 2$ [2]. Walter's MMM retains the same range for input and output (i.e., there is no extra subtraction step). The following $\mathrm{MMM}^2(A, B)$ computes $ABR^{-1} \bmod N$, where $R = 2^{n+2}$.

MMM^2(A, B)
1: $T = 0$;
2: for $i = 0$ to $n + 1$ {
3: $\quad Q[i] = (T + A[i] \times B[0]) \bmod 2$;

DOI: 10.1587/elex.11.20140782
Received August 13, 2014
Accepted August 18, 2014
Publicized September 10, 2014
Copyedited October 11, 2014

4: $T = (T + A[i] \times B + Q[i] \times N) \operatorname{div} 2;$
5: }

where the sizes of A and B are extended to $n+2$ bits, and $A[n+1]$ and $B[n+1]$ are set to 0. After $n+2$ iterations, $T < (3N/2^{n+1} + \cdots + 3N/2 + 0 + N)/2$; i.e., $T \in [0, 2N)$ is satisfied. Thus, the output T can be directly used as the input for the repeated MM in applications such as ME. Note that a long carry propagation delay occurs in performing Step 4, which may be significant from the perspective of hardware implementation.

2.3 Manochehri et al.'s MMM

Manochehri et al. presented a modified MMM using a new operator Θ called bitwise subtraction. In MMM[2], if $B[0] = 0$, $Q[i]$ is equal to the 0-th bit of the previous result T. This condition is satisfied when B is an even number. To accomplish this, B is encoded as $B1\ \Theta\ B2$. If B is odd, then $B1 = B - 1$ and $B2 = -1$. If B is even, then $B1 = B$ and $B2 = 0$. The algorithm MMM$^3(A, B, a)$ consists of two phases: the main loop and post-processing.

MMM^3(A, B, a)
1: $T = 0$;
2: if$(B[0] = 1)$ then $\{B1 = B - 1; B2 = -1;\}$
3: else $\{B1 = B; B2 = 0;\}$
4: for $i = 0$ to $n - 1$ {
5: $T = (T + A[i] \times B1 + T[0] \times N) \operatorname{div} 2;$
6: }
7: $Q = (T[0]\,\Theta\,(B2 \times a[0])) \bmod 2;$
8: $T = (T\,\Theta\,(B2 \times a) + Q \times N) \operatorname{div} 2;$

where $A, B < N$ and a is the normal number of the N-residue A. Note that, when B is odd, post-processing of Steps 7 and 8 is necessary to obtain a correct result as follows: After n iterations, $T = AB2^{-n} \bmod N = A(B-1)2^{-n} \bmod N = AB2^{-n} - A2^{-n} \bmod N$. To obtain the desired result, a should be added to T. In Steps 7 and 8, if B is odd, then $T = (T + a + ((T[0] + a[0]) \bmod 2) \times N) \operatorname{div} 2$. When B is even, $T = (T + T[0] \times N) \operatorname{div} 2$. The convergence range of T is analyzed as follows: After n iterations, $T < 2N(1/2 + 1/4 + \ldots + 1/2^{n-1})$, i.e., T falls within the range $[0, 2N)$. In the following Steps 7 and 8, if B is odd, the range of T is $T < (2N + N + N)/2 < 2N$, as desired. If B is even, $T < (2N + N)/2 < 2N$ is also satisfied. Note that to keep the consistent range $[0, 2N)$ for both input and output, the number of iterations of MMM^3 should be increased from n to $n+2$.

2.4 Improved unified MMM

In this section, we propose a unified and efficient algorithm that does not compute Steps 7–8 in MMM^3. The inputs to the proposed MMM^4 are two $n+2$ bit integers and the output T is an integer satisfying $0 \le T < 2N$, where $A[n+1] = B[n+1] = 0$.

MMM^4(A, B)
1: $T = 0$;

DOI: 10.1587/elex.11.20140782
Received August 13, 2014
Accepted August 18, 2014
Publicized September 10, 2014
Copyedited October 11, 2014

2: if($B[0] = 1$) then $\{B = B - 1; T = A;\}$
3: for $i = 0$ to $n + 1$ {
4: $\quad T = (T + A[i] \times B + T[0] \times N) \operatorname{div} 2;$
5: }

When B is odd, Step 2 sets the least significant bit of B to zero to make B an even number. Then, to discard the post-processing phase, T is initialized as A. The unified MMM[4] is more efficient than MMM[3] in computing the Montgomery modular multiplication. Manochehri et al.'s MMM[3] requires $n + 2$ iterations, whereas the proposed MMM[4] also requires $n + 2$ iterations. However, there is no need for post-processing to obtain the correct output, nor is there a need for any non-standard operations such as bitwise subtraction. The convergence range of T is analyzed as follows: After $n + 2$ iterations, $T < N/2 + N/2^{n+1} + 3N(1/4 + 1/8 + \ldots + 1/2^{n+2})$, i.e., T is bounded by $2N$, which guarantees that it can be fed back as an input for the next multiplication when implementing ME. When B is even, $T < N/2 + 3N(1/4 + 1/8 + \ldots + 1/2^{n+2})$, i.e., T is also bounded by $2N$. The proof of the correctness of the algorithm is as follows: If B is even, $T = AB2^{-(n+2)} \bmod N$. If B is odd, $T = (A + A(B - 1))2^{-(n+2)} \bmod N = AB2^{-(n+2)} \bmod N$, as desired.

To eliminate the carry propagation chain encountered in binary adders, MMM[4] can be implemented using the 5-to-2 carry-save adder (CSA) logic as in reference [6, 8]. To accomplish this, Step 4 can be rewritten as the following two equations: $T0[i] = (T1 + T2) \bmod 2;$ $\quad (T1, T2) = (T1 + T2 + A[i] \times (B1 + B2) + T0[i] \times N) \operatorname{div} 2$, where $(T1, T2)$ and $(B1, B2)$ denote the carry-save representations of T and B, respectively. The computation of $T0[i]$ is simply an odd-even check. Note that to remove the data dependency between two equations, the use of $T0[i]$ computed at iteration i can be delayed until the next iteration $i + 1$ [9]. As a result, these two equations are performed in a pipelined fashion and thus the generation of $T0[i]$ is excluded from the critical path. The pipelined version can be easily constructed by increasing the number of iterations by one, modifying $T0[i] \times N$ to $T0[i - 1] \times (-N^{-1} \bmod 4) \times N$, and adding the initial values $T0[-1] = A[n + 2] = B[n + 2] = 0$. The proof of the correctness of the algorithm is as follows: For ease of explanation, we use the normal binary representation instead of carry-save representation. Assume that B is an odd number. Let T_i be the value of the partial product T at the start of the loop. So $T_0 = 0$. Let $A_i = \sum_{j=0}^{i-1} A[j]2^j$, $T0_{i-1} = \sum_{j=0}^{i-2} T0[j]2^j$, and $N' = (-N^{-1} \bmod 4) \times N$. So $A_0 = T0_0 = T0_{-1} = 0$. Then, by induction, $2^i \times T_i = A + A_i \times B + T0_{i-1} \times N'$. Thus, the final value T_{n+3} of T satisfies $2^{n+3} \times T = A + A \times B + T0 \times N'$, so that $T = A \times B \times 2^{-(n+3)} \bmod N$ is obtained. This is easily applied to the case that B is even. Similarly, by induction, $2^i \times T_i = A_i \times B + T0_{i-1} \times N'$ and T_{n+3} satisfies $2^{n+3} \times T = A \times B + T0 \times N'$, therefore $T = A \times B \times 2^{-(n+3)} \bmod N$. Next, the convergence range of the output T is examined. When B is odd, T is bounded by $0 \le T < N'/2 + N'/2^{n+2} + 3N'(1/4 + 1/8 + \ldots + 1/2^{n+3})$, i.e., $T \in [0, 2N')$, as desired. When B is even, T is bounded by $0 \le T < N'/2 + 3N'(1/4 + 1/8 + \ldots + 1/2^{n+3})$, i.e., T also falls in the range of $[0, 2N')$.

DOI: 10.1587/elex.11.20140782
Received August 13, 2014
Accepted August 18, 2014
Publicized September 10, 2014
Copyedited October 11, 2014

3 Analysis and conclusion

We obtained the area of the gates, multiplexer and flip flop along with their worst-case intrinsic delays pertaining to unit drive-strength from the "SAMSUNG STD 150 0.13 µm 1.2 V CMOS Standard Cell Library" databook. Using these data we estimated the time and area complexities of the proposed structure and the corresponding Manochehri et al.'s structure.

Table I. Cells used for evaluation of time and area

	AND	OR	XOR	INV	MUX	FF
Time (ns)	0.094	0.105	0.167	0.039	0.141	-
Area (transistor count)	6.68	6.68	12.00	2.68	12.00	24

Note: FF, INV, and MUX denote a flip-flop, an inverter, and a 2-to-1 multiplexer, respectively.

The notations T_G and A_G denote the delay and area of the two-input basic cell G, respectively. Table I summarizes the time and area requirements for the standard cells used in our analysis.

Table II. Complexity comparison of MMM

Multipliers	Clock cycle	Critical path delay	Transistor count	AT complexity	Improvement		
					Time	Area	AT
Manochehri et al. [8]	$n+2$	$1.169(3A_{FA}+T_{XOR})$	$453.52(3A_{FA}+5A_{AND}+A_{XOR}+3A_{MUX}+10A_{FF})$	530.16	14.29%	11.76%	24.36%
Proposed	$n+3$	$1.002(3A_{FA})$	$400.20(3A_{FA}+6A_{AND}+A_{XOR}+2A_{MUX}+8A_{FF})$	401	-	-	-

To demonstrate the efficiency of the proposed method, we measure the area/time (AT) complexity of each work and then calculate the improvement defined as $(AT_{OLD} - AT_{NEW})/AT_{OLD}$, where AT_{OLD} and AT_{NEW} denote the complexity of the previous work and ours, respectively. From Table II, we can see that the developed MMM obtains obvious time, area, and AT advantages over Manochehri et al.'s MMM, assuming n is large enough to ignore the constant term. This work presents an efficient Montgomery architecture for computing the modular multiplication, which is the core operation in PKC. Compared to Manochehri et al.'s MMM, the proposed multiplier has the following properties: 1) lower critical path delay and area; 2) no requirement for post-processing to correct the output; 3) no requirement for a non-standard operation such as bitwise subtraction; 4) better performance for software implementation. Furthermore, the proposed MMM satisfies the demand that we maintain a consistent range of input and output in MMM. Note that the techniques proposed in this letter can be applied to systolic MMM architecture. The proposed MMM can be used for PKC applications such as asymmetric watermarking for geographic information system vector map management.

DOI: 10.1587/elex.11.20140782
Received August 13, 2014
Accepted August 18, 2014
Publicized September 10, 2014
Copyedited October 11, 2014