New systolic modular multiplication architecture for efficient Montgomery multiplication

Se-Hyu Choi and Keon-Jik Lee
School of Architectural, Civil, Environmental and Energy Engineering, Kyungpook National University, Daegu, 702–701, Korea
a) LeeMeeKael@gmail.com

Abstract: We propose a new low complexity Montgomery algorithm enabling the efficient selection of the quotient value necessary for an exact division in Montgomery multiplication. We also present two new systolic multipliers which use similar data flows as described in the most significant bit (MSB)-first $GF(2^m)$ multiplier in [1]. The proposed parallel and serial multipliers have less hardware and time complexities compared to related multiplier. The serial multiplier can be well applied to space-limited hardware. Furthermore, our proposed systolic multipliers include regularity, modularity, local interconnection, and unidirectional data flow features.

Keywords: Montgomery, modular multiplication, systolic array

Classification: Integrated circuits

References

[1] C. L. Wang and J. L. Lin: IEEE Trans. Circuits Syst. 38 (1991) 796. DOI:10.1109/31.135751
[2] P. L. Montgomery: Math. Comput. 44 (1985) 519. DOI:10.1090/S0025-5718-1985-0777282-X
[3] C. D. Walter: IEEE Trans. Comput. 42 (1993) 376. DOI:10.1109/12.210181
[4] C. McIvor, M. McLoone and J. V. McCanny: IEE Proc. Comput. Digit. Tech. 151 (2004) 402. DOI:10.1049/ip-cdt:20040791
[5] K. Manochehri, B. Sadeghian and S. Pourmozafari: IEICE Electron. Express 7 (2010) 513. DOI:10.1587/elex.7.513
[6] S. H. Choi and K. J. Lee: IEICE Electron. Express 11 (2014) 20140782. DOI:10.1587/elex.11.20140782

1 Introduction

The core operation of a public key cryptosystem (PKC) is modular exponentiation (ME) with a large number, which is conducted through a sequence of modular multiplications (MMs). The throughput of a PKC is dependent on the speed and number of MMs. To accelerate the MM process, Montgomery’s multiplication (MMM) is recognized as the best solution [2]. An architecture based on MMM is
the best-studied architecture in hardware. Differences exist because of various methods for avoiding long carry chains. The most common way to do so is using a systolic array. Therefore, an efficient MMM architecture under limited hardware resources is important from a system performance point of view. Several MMM architectures have been proposed [3, 4, 5, 6]. Although the processing element (PE) structure proposed in previous researches is adequate for designing a systolic multiplier, it is insufficient for obtaining optimality.

In this letter, we propose a new MMM for an efficient quotient determination when performing MMM. Also, we propose two new systolic multipliers having similar data flows as shown in the MSB-first $GF(2^n)$ multiplication of [1]. The proposed scheme can be used as a kernel circuit for both MM and ME.

2 MMM algorithm

2.1 Bit-parallel systolic MMM

MMM employs only additions and shift operations instead of trial divisions. The price paid is to convert the input and output operands into and out of Montgomery domain, which is negligible in applications such as a PKC. Walter proposed the parallel systolic MMM $^1(A, B, N) = ABR^{-1} \mod N$, where $N$ is an $n$-bit odd integer, $R = 2^{n+2}$, and $A (= aR \mod N)$ and $B (= bR \mod N)$ are the $N$-residue of integer $a$ and $b$, respectively, for $A, B, T \in [0, 2N)$. In this algorithm, $T(i)$ denotes the result generated at the $i$-th iteration, and $t_{i,j}$ denotes the $j$-th bit of $T(i)$, where $T(0) = 0$. The notation $a_i$ represents the $i$-th bit of $A$; thus, we have $A = \sum_{i=0}^{n+1} a_i2^i$, where $a_{n+1} = 0$.

$$MMM(A, B, N)$$

1: for $i = 0$ to $n + 1$
2: $q_i = (t_{i,0} + a_i \times b_0) \mod 2$;
3: $T(i+1) = (T(i) + a_i \times B + q_i \times N) \div 2$;

$MMM^1$ retains the same range for the input and output. Thus, the output $T$ can be used as the input for the repeated MM in ME. Note that if $b_0 = 0$, $q_i$ is equal to the 0-th bit of the previous result $T(0)$. This condition is satisfied when $B$ is even. The modified MMM using this property is formulated as follows, where $T(0) = 0$ [6].

$$MMM^2(A, B, N)$$

1: if ($b_0 = 1$) then $B = B - 1$; $T(0) = A$;
2: for $i = 0$ to $n + 1$
3: $T(i+1) = (T(i) + a_i \times B + t_{i,0} \times N) \div 2$;

The convergence range of $T$ is analyzed as follows. When $B$ is even, $T < N/2 + 3N(1/4 + 1/8 + \ldots + 1/2^{n+2})$, i.e., $T \in [0, 2N)$, which guarantees that $T$ can be fed back as an input for the next MM when implementing ME. When $B$ is odd, Step 1 sets the least significant bit (LSB) of $B$ to zero, and the initial value of $T$ is reset to $A$. After $n + 2$ iterations, $T < N/2 + N/2^{n+1} + 3N(1/4 + 1/8 + \ldots + 1/2^{n+2})$, i.e., $T \in [0, 2N)$ satisfied. The proof of the correctness of this algorithm is as follows: If
B is even, \( T = AB2^{-(n+2)} \mod N \). If B is odd, \( T = (A + A(B - 1))2^{-(n+2)} \mod N = AB2^{-(n+2)} \mod N \), as desired. Note that the critical operation of MMM² is a three-operand addition within a loop and there exists a long carry propagation delay. One way of solving this problem is to employ a systolic array instead of a ripple-carry adder. The recursive equations of MMM² are described as follows, where \( T^{(0)} = 0 \), and whenever inner for loop starts, carry C0 and C1 are reset to zero.

**MMM³(A, B, N)**

1: if \( (b_0 = 1) \) then \( B = B - 1 \); \( T^{(0)} = A \);
2: for \( i = 0 \) to \( n + 1 \)
3: for \( j = 0 \) to \( n + 2 \)
4: \( t' = (t_{i,j} + a_i \times b_j + C0_{i,j}) \mod 2; \)
5: \( C0_{i,j+1} = (t_{i,j} + a_i \times b_j + C0_{i,j}) \div 2; \)
6: \( t_{i+1,j-1} = (t' + t_{i,0} \times n_j + C1_{i,j}) \mod 2; \)
7: \( C1_{i,j+1} = (t' + t_{i,0} \times n_j + C1_{i,j}) \div 2; \)

where the result \( T = [t_{n+2,n}, \ldots, t_{n+2,0}] \) is an \((n + 1)\)-bit number and \( t_{n+2,-1} = 0 \). To construct a PE structure that has the smallest input/output delay, a critical path optimization using Boolean operators is conducted. As a result, the critical path delay is reduced from \( 3T_{\text{XOR}} + 2T_{\text{AND}} + T_{\text{OR}} \) to \( 3T_{\text{XOR}} \), where \( T_{\text{AND}}, T_{\text{OR}} \), and \( T_{\text{XOR}} \) denote the delay of the 2-input AND gate, 2-input OR gate, and 2-input XOR gate, respectively. Steps 4–7 of MMM³ are reformulated as follows, where \( T^{(0)} = 0 \), and whenever inner for loop starts, carry C0, D0, and D1 are reset to zero.

**MMM⁴(A, B, N)**

1: if \( (b_0 = 1) \) then \( B = B - 1 \); \( T^{(0)} = A \);
2: for \( i = 0 \) to \( n + 1 \)
3: for \( j = 0 \) to \( n + 2 \)
4: \( t' = (t_{i,j} \oplus C0_{i,j}) \oplus (a_i \land b_j); \)
5: \( C0_{i,j+1} = ((t_{i,j} \oplus C0_{i,j}) \land (a_i \land b_j)) \lor (t_{i,j} \land C0_{i,j}) ; \)
6: \( t_{i+1,j-1} = t' \oplus ((t_{i,0} \land n_j) \oplus (D0_{i,j} \lor D1_{i,j})); \)
7: \( D0_{i,j+1} = t' \lor ((t_{i,0} \land n_j) \oplus (D0_{i,j} \lor D1_{i,j})); \)
7: \( D1_{i,j+1} = (t_{i,0} \land n_j) \lor (D0_{i,j} \lor D1_{i,j}); \)

The logic expressions \((t_{i,j} \oplus C0_{i,j}) \) and \((a_i \land b_j) \) are shared in Steps 4 and 5; \((t_{i,0} \land n_j) \oplus (D0_{i,j} \lor D1_{i,j}) \) and \( t' \), in Steps 6 and 7.1; and \( (t_{i,0} \land n_j) \) and \( (D0_{i,j} \lor D1_{i,j}) \), in Steps 6–7.2. Based on the proposed algorithm, the bit-parallel systolic multiplier is shown in Fig. 1(a), where \( n = 3 \). This array consists of \((n + 2) \times (n + 2) \) basic cells of Fig. 1(b) and \((n + 3) \) AND gates. One 1-bit latch (denoted by “*”) is placed at each horizontal and slant link, and two 1-bit latches at each vertical link. The cells in the \( i \)-th row of the array conduct operations of the \( i \)-th iteration. Note that in Fig. 1(a), the 0-th column (i.e., the cells at position \((i, 0) \), \( 0 \leq i \leq n + 1 \)) was removed because \( b_0 \) is always zero and \( n_0 \) is always one. In Fig. 1(a), the cell at position \((i, j) \) receives \( t_{i,j} \) from the cell at position \((i - 1, j + 1) \) of the previous row, computes \( t_{i+1,j-1} \), and then passes it to the cell at position \((i + 1, j - 1) \) of the next row.
The initial positions and index points of each input are as follows. First, $a_i$ $(0 \leq i \leq n + 1)$ enters index $[i, 1]$ from the right side and flows in the direction of $[0, 1]$, where $a_{i+1} = 0$. The values $b_j$ and $n_j$ $(1 \leq j \leq n + 2)$ then enter index $[0, j]$ from the top and flow in the direction of $[1, 0]$, respectively, where $b_{n+1} = b_{n+2} = n_{n+1} = n_{n+2} = 0$. Next, $t_j$ $(1 \leq j \leq n + 2)$ enters index $[0, j]$ from the top, and is computed using the partial products generated by the previous row to give new partial products that are passed on to the next row, and then flows in the direction of $[1, -1]$, where $t_j = b_0 \land a_j$, for $1 \leq j \leq n$, and $t_{n+1} = t_{n+2} = 0$. Then, $t_{i, 0}$ $(0 \leq i \leq n + 1)$ enters index $[i, 1]$ from the right side and flows in the direction of $[0, 1]$, where $t_{0,0} = b_0 \land a_0$. Next, $C0_i$ and $D1_i$ $(0 \leq i \leq n + 1)$ enter index $[i, 1]$ from the right side and their computation results flow in the direction of $[0, 1]$, respectively, where $C0_i = D1_i = 0$. Then, $D0_i$ with the initial value $t_{i,0}$ enters index $[i, 1]$ from the right side and its computation result flows in the direction of $[0, 1]$. The final result, $T^{(n+2)}$, emerges in parallel from the bottom row of the array after $n + 2$ iterations.

The basic cell consists of six 2-input AND gates, four 2-input XOR gates, and two 2-input OR gates as shown in Fig. 1(b). The $(i, j)$ cell receives $t_{i,j}$ as its input from the $(i - 1, j + 1)$-th cell; $n_j$ and $b_j$, from the $(i - 1, j)$-th cell; and $a_i$, from the $(i, j - 1)$-th cell. The critical path delay of this structure is the total delay of the three 2-input XOR gates. In Fig. 1(a), $b_j$ and $n_j$ $(1 \leq j \leq n + 2)$ are staggered by one clock cycle relative to $b_{j+1}$ and $n_{j+1}$, respectively, and $a_{i+1}$ $(0 \leq i \leq n)$ is staggered by two clock cycles relative to $a_i$.

### 2.2 Bit-serial systolic MM

By projecting Fig. 1(a) in a westward direction (projection vector $[0, 1]$ and schedule vector $[2, 1]$) and retiming using the cut-set systolization techniques [1], a new one-dimensional serial systolic array in Fig. 2(a) is derived. This array consists of $n + 2$ basic cells and one 2-input AND gate; the functions of the cell are depicted in Fig. 2(b). This multiplier is controlled by a control sequence.
$\text{ctr} = 011\ldots1$ of length $n+2$. As shown in MMM$^4$, because the LSB of each partial result $T^{(i)}$ is required to execute Steps 6–7.2 and $D_0$, is initialized as $t_{i,0}$, two 2-to-1 MUXes and two 1-bit latches are added to each cell in Fig. 2(a). The zero bit of $\text{ctr}$ enters the array from the right side, one cycle ahead of the LSB of $B$ and $N$ and is synchronized with the LSB of $T^{(i)}$. When $\text{ctr}$ is in logic 0, the LSB of each $T^{(i)}$ is loaded into each cell. Furthermore, two 2-to-1 MUXes and two 1-bit latches are added to each cell in Fig. 2(a) owing to the fact that two zeros (initial values for $C_0$ and $D_1$) must be fed to each row of Fig. 1(a) from the rightmost cell. When $\text{ctr}$ is in logic 0, these MUXes generate two zeros and feed them into the cell.

Note that according to the projection, input values other than $A$ enter the right side of the array in a serial form, while each bit of $A$ should remain inside the systolic array, i.e., $a_i$ ($0 \leq i \leq n+1$) should remain at the $i$-th cell to be ready for the execution. It is possible to incorporate an additional one 2-to-1 MUX and one 1-bit latch into each cell in Fig. 2(a), so that $a_i$ may also enter the array serially with the LSB first at the same time as $\text{ctr}$. That is, when $\text{ctr} = 0$ enters the $i$-th cell, $a_i$ also enters that cell, and the loading operation of $a_i$ occurs.

![Fig. 2. (a) Proposed multiplier (b) Circuit of the basic cell](image)

The basic cell of Fig. 2(b) consists of six 2-input AND gates, four 2-input XOR gates, two 2-input OR gates, five 2-to-1 MUXes, and thirteen 1-bit latches, and its critical path delay is the total delay of the three 2-input XOR gates and one 2-to-1 MUX. The result $t_j$ ($0 \leq j \leq n$) emerges from the left side of the array in serial form with the LSB first.

### 3 Analysis and conclusion

In this letter, we presented two new systolic arrays for conducting an MMM, one with a parallel-form input/output and the other with a serial-form input/output. These two arrays involve a unidirectional data flow, and are highly regular and nearest-neighbor connected. They are thus well-suited for VLSI implementation. In [3], Walter described how to simplify the calculation of the quotient $q_i$ by shifting $B$ up by one place to make $b_0 = 0$. However, this can be performed at the cost of adding an extra row and column to the original parallel systolic array to obtain an output less than $2N$. The resulting parallel (resp., serial) array consists of $(n+3) \times$...
(n + 4) (resp., n + 3) cells, and the hardware complexity and latency (resp., throughput) are higher (resp., lower) than those of the original array.

We introduced a new MMM that selects $q_i$ efficiently by manipulating the initial values of $B$ and $T$ to make LSB of $B$ zero when performing $T = MMM(A, B, N)$ and the associated hardware designs. The proposed MMM satisfies the demand that we maintain a consistent range of input and output in MMM and does not increase the size of the array compared to Walter’s method. The resulting parallel (resp., serial) array is composed of $(n + 2) \times (n + 2)$ (resp., $n + 2$) basic cells.

We obtained the area of the gates, multiplexer, and latch 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 and related work. Table I summarizes the time and area requirements for the cells used in our analysis, where the notation $G_n$ denotes the $n$-input logic cell $G$.

| Table I. Time and area evaluations for the cells used |
|------------------------------------------------------|
| Time (ns) | AND$_2$ | OR$_2$ | XOR$_2$ | MUX | Latch |
|-----------|---------|-------|--------|-----|-------|
| Area (transistor count) | 6.68 | 6.68 | 12.00 | 12.00 | 16.00 |
| Note: MUX denotes a 2-to-1 multiplexer. |

To demonstrate the efficiency of the proposed method, we measure the area-time (AT) complexity of each work and then calculate the improvement. From Table II, we can see that the proposed multipliers shown in Figs. 1 and 2 have better advantages in terms of the area, time, AT, throughput, and latency over Walter’s method. In detail, the AT complexity of the parallel (resp., serial) systolic

| Table II. Complexity comparison of systolic multipliers |
|-------------------------------------------------------|
| Multipliers | Walter’s method | Proposed |
|-------------|-----------------|----------|
| # cells | $(n + 3) \times (n + 4)$ | $n + 3$ | $(n + 2) \times (n + 2)$ | $n + 2$ |
| Throughput | $3n + 7$ | $3n + 7$ | $3n + 5$ | $3n + 5$ |
| Latency | $6n^2 + 42n + 72$ | $6n + 18$ | $6n^2 + 25n + 27$ | $6n + 12$ |
| OR$_2$ | $2n^2 + 14n + 36$ | $2n + 6$ | $2n^2 + 8n + 8$ | $2n + 4$ |
| XOR$_2$ | $4n^2 + 28n + 48$ | $4n + 12$ | $4n^2 + 16n + 16$ | $4n + 8$ |
| MUX | $0$ | $5n + 15$ | $0$ | $5n + 10$ |
| Latch | $10n^2 + 70n + 120$ | $13n + 39$ | $10n^2 + 40n + 40$ | $13n + 26$ |
| # transistors | $261.44n^2 + 1830.08n + 3217.44$ | $369.44n$ | $261.44n^2 + 1052.44n + 1108.32$ | $369.44n$ |
| Cell area | $+3217.44$ | $+1108.32$ | $+1065.8$ | $+738.88$ |
| Cell time | $0.64$ | $0.64$ | $0.50$ | $0.64$ |
| Delay | $1.50n + 3.51$ | $1.93n + 4.49$ | $1.50n + 2.51$ | $1.93n + 3.21$ |
| AT complexity | $392.94n^2 + 3667.48n^2 + 11253.90n + 11283.56$ | $711.54n^2 + 3794.89n + 4980.79$ | $392.94n^2 + 2236.72n^2 + 4238.26n + 2669.83$ | $711.54n^2 + 2608.99n + 2371.80$ |
multiplier is improved by approximately 0.18% (resp., 0.08%) compared to Walter’s method (for \( n = 2048 \)). The proposed parallel (resp., serial) multiplier produces the results at a rate of one per 1 (resp., \( n + 2 \)) cycles with a latency of \( 3n + 5 \) (resp., \( 3n + 5 \)) cycles.

Note that the parallel systolic architecture has a better throughput but much higher hardware cost than a serial systolic architecture. Thus, the serial one is attractive for resource-constrained applications. The proposed MMM can be used for PKC applications such as asymmetric watermarking for geographic information system vector map management.