

A semi-systolic Montgomery multiplier over $GF(2^m)$

Kee-Won Kim¹ and Jun-Cheol Jeon^{2a)}

¹ College of Convergence Technology, Dankook University, Cheonan, Korea

² Department of Computer Engineering, Kumoh National Institute of Technology, Gumi, Korea

a) jcjeon@kumoh.ac.kr, Corresponding Author

Abstract: In this study, we present an efficient finite field arithmetic architecture based on systolic array for multiplication which is a core algorithm for division and exponentiation operations. In order to obtain dedicated area-efficient circuits, we adopt Montgomery multiplication algorithm and systolic array. First of all we induce an efficient arithmetic algorithm from typical Montgomery multiplication using an effective factor, then we design an efficient semi-systolic array based multiplication architecture which is highly suitable for pipelined operations. The proposed multiplier saves at least 40% area complexity as compared to the corresponding existing structures.

Keywords: finite field, Montgomery multiplication, systolic array, cryptography

Classification: Integrated circuits

References

- [1] R. E. Blahut: *Theory and Practice of Error Control Codes* (Addison-Wesley, Reading, MA, 1983).
- [2] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone: *Handbook of Applied Cryptography* (CRC Press, Boca Raton, FL, 1996).
- [3] P. Montgomery: *Math. Comput.* **44** (1985) 519. DOI:10.1090/S0025-5718-1985-0777282-X
- [4] C. Koc and T. Acar: *Des. Codes Cryptogr.* **14** (1998) 57. DOI:10.1023/A:1008208521515
- [5] Z. Wang and S. Fan: *IEEE Trans. Comput.* **61** (2012) 415. DOI:10.1109/TC.2010.272
- [6] A. Hariri and A. Reyhani-Masoleh: *IEEE Trans. Comput.* **60** (2011) 1341. DOI:10.1109/TC.2010.258
- [7] W.-T. Huang, C. H. Chang, C. W. Chiou and F. H. Chou: *IET Inf. Secur.* **4** (2010) 111. DOI:10.1049/iet-ifs.2009.0160
- [8] K. W. Kim and J. C. Jeon: *Int. J. Cyber-Security Digital Forensics* **1** [2] (2012) 122.
- [9] K. W. Kim and S. H. Kim: *IEICE Electron. Express* **10** (2013) 20130354. DOI:10.1587/elex.10.20130354
- [10] S. H. Choi and K. J. Lee: *IEICE Electron. Express* **11** (2014) 20140713. DOI:10.1587/elex.11.20140713

- [11] N. Weste and K. Eshraghian: *Principles of CMOS VLSI Design: A System Perspective* (Addison-Wesley, Reading, MA, 1985).
- [12] STMicroelectronics: STMicroelectronics datasheet (2013) <http://www.st.com>.

1 Introduction

Finite field arithmetic operations have recently been applied in a variety of fields, including cryptography and error-correcting codes [1, 2]. The multiplication among these operations is the most important arithmetic operation. This is because the time-consuming operations such as exponentiation, division, and multiplicative inversion can be decomposed into repeated multiplications. Thus, the fast multiplication architecture with low complexity is needed to design dedicated high-speed circuits.

Montgomery multiplication (MM) algorithm, introduced by Montgomery is one of most interesting and useful advances in this realm for fast modular integer multiplication [3]. The MM was successfully adapted to finite field $GF(2^m)$ by Koc and Acar [4]. The MM over $GF(2^m)$ is a very efficient solution for the design of a fast architecture and VLSI implementation [5, 6]. Hariri and Reyhani-Masoleh have considered concurrent error detection for MM over $GF(2^m)$ [6]. Three different multipliers, namely the bit-serial, digit-serial, and bit-parallel multipliers, have been considered and the concurrent error detection scheme has been derived and implemented for each of them.

Many semi-systolic multiplier over $GF(2^m)$ have been developed [7, 8, 9, 10]. Huang et al. [7] proposed the semi-systolic polynomial basis multiplier over $GF(2^m)$ to reduce both area and time complexities. Also they proposed the semi-systolic polynomial basis multipliers with concurrent error detection and correction capability. Kim and Jeon [8] proposed much faster Montgomery multiplier than the architecture proposed in [7]. They proposed a two-fold architecture so that two different architectures are operated at the same time. Kim and Kim [9] proposed an area-efficient multiplier than multipliers proposed in [7, 8]. Recently, Choi and Lee [10] proposed a low complexity semi-systolic multiplier based on the redundant basis representation of the finite field elements.

In this paper, we induce an efficient multiplication algorithm for reduction of hardware complexity of typical architectures. The proposed algorithm enables multiplication to operate in pipelined computation so that two different operands can be computed in the same hardware architecture.

2 Montgomery multiplication on $GF(2^m)$

$GF(2^m)$ is a kind of finite field that contains 2^m different elements. This finite field is an extension of $GF(2)$ and any $A \in GF(2^m)$ can be represented as a polynomial of degree $m - 1$ over $GF(2)$, such as $A = \sum_{i=0}^{m-1} a_i x^i$, where $a_i \in \{0, 1\}$, $0 \leq i \leq m - 1$. Let x be a root of the polynomial, then the irreducible polynomial G is represented as $G(x) = \sum_{i=0}^m g_i x^i$, where $g_i \in GF(2)$, $0 \leq i \leq m - 1$.

Let α and β be two elements of $GF(2^m)$, then we define $\gamma = \alpha \cdot \beta \bmod G$, where G denotes $G(x)$. Also, let A and B be two Montgomery residues, then they are

defined as $A = \alpha \cdot R \bmod G = \sum_{i=0}^{m-1} a_i x^i$ and $B = \beta \cdot R \bmod G = \sum_{i=0}^{m-1} b_i x^i$, where α and β are Montgomery factors, R and an irreducible polynomial, G are relatively prime, and $\gcd(R, G) = 1$. Then, the MM algorithm over $GF(2^m)$ can be formulated as $P = A \cdot B \cdot R^{-1} \bmod G$, where R^{-1} is the inverse of R modulo G , and $R \cdot R^{-1} + G \cdot G' = 1$ [2].

Then, P can be expressed as $P = (\alpha \cdot R) \cdot (\beta \cdot R) \cdot R^{-1} \bmod G = \gamma \cdot R \bmod G$ by the definition of the Montgomery residue. It means that P is the Montgomery residue of γ .

3 Proposed architecture

Based on the property of parallel architecture, we choose the Montgomery factor, $R = x^{\lfloor m/2 \rfloor}$. Then, the MM over $GF(2^m)$ can be formulated as

$$P = A \cdot B \cdot R^{-1} \bmod G = A \cdot B \cdot x^{-\lfloor m/2 \rfloor} \bmod G. \quad (1)$$

We know that x is a root of $G(x)$, i.e., $G(x) = 0$ and $g_m = g_0 = 1$ over all irreducible polynomials. Thus, x^m and x^{-1} are as follows:

$$x^m \bmod G = \sum_{i=0}^{m-1} g_i x^i \text{ and } x^{-1} \bmod G = x^{m-1} + \sum_{j=1}^{m-1} g_j x^{j-1}. \quad (2)$$

Meanwhile, $P = A \cdot B \cdot x^{-\lfloor m/2 \rfloor} \bmod G$ can be expressed as follows:

$$P = [b_{\lfloor m/2 \rfloor - 1} A x^{-1} + \dots + b_1 A x^{-\lfloor m/2 \rfloor + 1} + b_0 A x^{-\lfloor m/2 \rfloor} + b_{\lfloor m/2 \rfloor} A + \dots + b_{m-2} A x^{\lfloor m/2 \rfloor - 2} + b_{m-1} A x^{\lfloor m/2 \rfloor - 1}] \bmod G. \quad (3)$$

Now, it expresses that P can be divided into two parts. One is based on the negative powers of x and the other is based on the positive powers of x . Let $l = \lceil m/2 \rceil$ and $k = \lfloor m/2 \rfloor$. P can be denoted by $P = C + D$, where $C = \sum_{j=0}^{k-1} b_j A x^{-k+j} \bmod G$ and $D = \sum_{j=k}^{m-1} b_j A x^{-k+j} \bmod G$.

Meanwhile, let $\tilde{A}^{(i)} = A x^{-i} \bmod G$ and $A^{(i)} = A x^i \bmod G$. Then the equations can be expressed as $\tilde{A}^{(i)} = \sum_{j=0}^{m-1} \tilde{a}_j^{(i)} x^j$ and $A^{(i)} = \sum_{j=0}^{m-1} a_j^{(i)} x^j$, where $\tilde{A}^{(0)} = A^{(0)} = A$. By using (2), $\tilde{A}^{(i)}$ and $A^{(i)}$ are rewritten as

$$\tilde{A}^{(i)} = \tilde{A}^{(i-1)} x^{-1} \bmod G = \sum_{j=0}^{m-1} (\tilde{a}_{j+1}^{(i-1)} + \tilde{a}_0^{(i-1)} g_{j+1}) x^j \quad (4)$$

$$A^{(i)} = A^{(i-1)} x \bmod G = \sum_{j=0}^{m-1} (a_{j-1}^{(i-1)} + a_{m-1}^{(i-1)} g_j) x^j. \quad (5)$$

From (4) and (5), we can obtain the coefficient of $\tilde{A}^{(i)}$ and $A^{(i)}$ as follows:

$$\tilde{a}_j^{(i)} = \tilde{a}_{j+1}^{(i-1)} + \tilde{a}_0^{(i-1)} g_{j+1} \text{ and } a_j^{(i)} = a_{j-1}^{(i-1)} + a_{m-1}^{(i-1)} g_j, \quad (6)$$

where $\tilde{a}_m^{(i-1)} = a_{-1}^{(i-1)} = 0$, $g_m = g_0 = 1$ and $0 \leq j \leq m-1$.

Also, using the formulas of $\tilde{A}^{(i)}$ and $A^{(i)}$, the terms C and D are represented by the following equations. For deriving the identical structure, we add $z\tilde{A}^{(0)}$ to C , where $z = 0$.

$$C = \sum_{j=0}^{k-1} b_j A x^{-k+j} \bmod G = \sum_{j=0}^{k-1} b_j \tilde{A}^{(k-j)} + z\tilde{A}^{(0)} \quad (7)$$

and

$$D = \sum_{j=k}^{m-1} b_j A x^{-k+j} \bmod G = \sum_{j=0}^{l-1} b_{k+j} A^{(j)}. \quad (8)$$

From (7) and (8), the recurrence equations of C and D can be formulated by the following equations, where $C^{(0)} = D^{(0)} = 0$.

$$C^{(i)} = \begin{cases} C^{(i-1)} + z\bar{A}^{(i-1)}, & \text{for } i = 1 \\ C^{(i-1)} + b_{k-i+1}\bar{A}^{(i-1)}, & \text{for } 2 \leq i \leq k+1, \end{cases} \quad (9)$$

$$D^{(i)} = D^{(i-1)} + b_{k+i-1}a_j^{(i-1)}, \text{ for } 1 \leq i \leq l, \quad (10)$$

From (9) and (10), we can obtain the coefficient of $C^{(i)}$ and $D^{(i)}$ as follows:

$$c_j^{(i)} = \begin{cases} c_j^{(i-1)} + z\bar{a}_j^{(i-1)}, & \text{for } i = 1 \\ c_j^{(i-1)} + b_{k-i+1}\bar{a}_j^{(i-1)}, & \text{for } 2 \leq i \leq k+1, \end{cases} \quad (11)$$

$$d_j^{(i)} = d_j^{(i-1)} + b_{k+i-1}a_j^{(i-1)}, \text{ for } 1 \leq i \leq l, \quad (12)$$

where $c_j^{(0)} = d_j^{(0)} = 0 (0 \leq j \leq m-1)$ and $z = 0$.

In order to derive an efficient architecture from (6), (11) and (12), we perform index-transformation in the proposed formulas and eliminate unnecessary computations. We circularly shift the coefficient computation of $C^{(i)}$ to the right by one bit, reverse the computation order of coefficient of $D^{(i)}$ and then circularly shift it to the right by one bit, and reverse the computation order of coefficients of $\bar{A}^{(i-1)}$. In doing so, we obtain the hardware architecture of the proposed Montgomery multipliers over $GF(2^8)$ and $GF(2^9)$, as shown in Fig. 1, where the boxed D denotes 1-bit latch(flip-flop). The proposed multiplier over $GF(2^m)$ is composed of $k+1$ $\hat{W}_0^{(i)}$ cells, $(m-1) \times (k+1)$ $W_j^{(i)}$ cells, and one $\hat{S}$ cell. At the first clock cycle, the input values are submitted for the computation of C and the next input values at the second clock cycle are for the computation of D . At the i th clock cycle, $\hat{W}_0^{(i)}$ and $W_j^{(i)}$ cells compute the following equations

$$c_{\langle m-j \rangle}^{(i)} = \begin{cases} c_{\langle m-j \rangle}^{(i-1)} + z\bar{a}_{\langle m-j \rangle}^{(i-1)}, & \text{for } i = 1 \\ c_{\langle m-j \rangle}^{(i-1)} + b_{k-i+1}\bar{a}_{\langle m-j \rangle}^{(i-1)}, & \text{for } 2 \leq i \leq k+1, \end{cases} \quad (13)$$

$$\bar{a}_{m-1-j}^{(i)} = \bar{a}_{\langle m-j \rangle}^{(i-1)} + \bar{a}_0^{(i-1)}g_{m-j}, \text{ for } 1 \leq i \leq k, \quad (14)$$

where $0 \leq j \leq m-1$, $z = 0$, $\bar{a}_m^{(i-1)} = 0$, and $g_m = 1$. At the $(i+1)$ th clock cycle, $\hat{W}_0^{(i)}$ and $W_j^{(i)}$ cells compute the following equations

$$d_{\langle j-1 \rangle}^{(i)} = d_{\langle j-1 \rangle}^{(i-1)} + b_{k+i-1}a_{\langle j-1 \rangle}^{(i-1)}, \text{ for } 1 \leq i \leq l, \quad (15)$$

$$a_j^{(i)} = a_{\langle j-1 \rangle}^{(i-1)} + a_{m-1}^{(i-1)}g_j, \text{ for } 1 \leq i \leq l-1, \quad (16)$$

where $0 \leq j \leq m-1$, $a_{-1}^{(i-1)} = 0$, $g_0 = 1$, and $\langle p \rangle$ denotes p modulo m .

When m is even, the results of C and D are produced at the $(k+1)$ -th and k -th row, respectively after $k+1$ clock cycles. However, when m is odd, the results of C and D are produced at the $(k+1)$ -th row after $k+1$ and $k+2$ clock cycles, respectively. It means that the 1-bit latches are demanded at the output lines of c_i , where $0 \leq i \leq m-1$ when m is only odd. The detailed circuits of the cells in Fig. 1 are illustrated in Fig. 2.

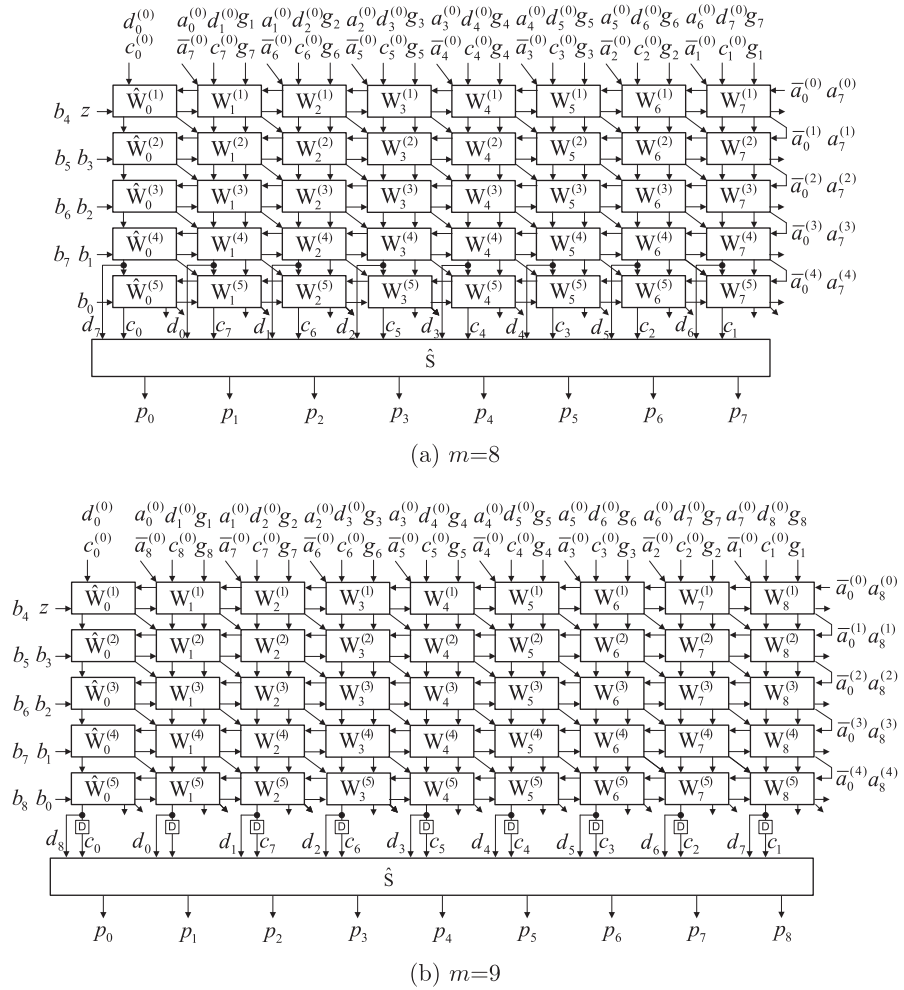


Fig. 1. The proposed multipliers in $GF(2^m)$

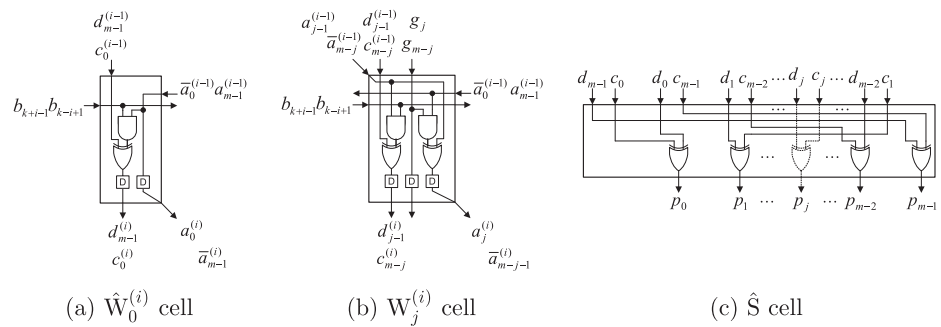


Fig. 2. Detailed circuit of $\hat{W}_0^{(i)}$, $W_j^{(i)}$, and $\hat{S}$ cell

4 Complexity analysis

In CMOS VLSI technology, each gate is composed of several transistors [11]. We adopt that $A_{AND2} = 6$, $A_{XOR2} = 6$, and $A_{LATCH1} = 8$, where A_{GATE_n} denotes transistor count of an n -input gate, respectively. Also, for a further comparison of time complexity, we adopt the practical integrated circuits in [12] and the following assumptions, as discussed in detail in [7], are made: $T_{AND2} = 7$, $T_{XOR2} = 12$, and $T_{LATCH1} = 13$, where T_{GATE_n} denotes the propagation delay of an n -input gate, respectively.

Table I. Comparison of semi-systolic architectures.

gate/delay	Huang et al. [7]	Kim-Jeon [8] even m /odd m	Kim-Kim [9] even m /odd m	Choi-Lee [10]	Fig. 1 even m /odd m
Area Complexity					
AND ₂	$2m^2$	$2m^2 + m - 1$	$\frac{2m^2 - m}{2m^2}$	$m^2 + 2m + 1$	$\frac{m^2 + 1.5m - 1}{m^2 + 0.5m - 0.5}$
XOR ₂	$2m^2$	$2m^2 + 2m - 1$	$\frac{m^2}{m^2 + 2m}$	$m^2 + 2m + 1$	$\frac{m^2 + 2.5m - 1}{m^2 + 1.5m - 0.5}$
XOR ₃	0	0	$\frac{0.5m^2}{0.5m^2 - 0.5m}$	0	0
Latch	$3m^2$	$\frac{3m^2 + 3m - 1}{3m^2 + 2m - 1}$	$\frac{2.5m^2 - m}{2.5m^2 - 1.5m}$	$3.5m^2 + 3.5m + 2$	$\frac{1.5m^2 + 2.5m - 1}{1.5m^2 + 2m - 0.5}$
Transistors	$48m^2$	$\frac{48m^2 + 42m - 20}{48m^2 + 34m - 20}$	$\frac{44m^2 - 14m}{44m^2 - 6m}$	$40m^2 + 52m + 28$	$\frac{24m^2 + 44m - 20}{24m^2 + 28m - 10}$
Time Complexity					
Cell delay	32	32	44	25	32
Latency	m	$\frac{0.5m + 1}{0.5m + 0.5}$	$\frac{0.5m + 1}{0.5m + 0.5}$	$m + 1$	$\frac{0.5m + 1}{0.5m + 1.5}$
Total delay	$32m$	$\frac{16m + 32}{16m + 16}$	$\frac{22m + 44}{22m + 22}$	$25m + 25$	$\frac{16m + 32}{16m + 48}$

A circuit comparison between the proposed multiplier and the related multipliers is given in Table I. In detail, the results show that the proposed semi-systolic multiplier saves about 50, 50, 45, and 40% area complexities as compared to the existing multipliers by Huang [7], Kim-Jeon [8], Kim-Kim [9], and Choi-Lee [10], respectively. The time complexity of our multiplier is the same with Kim-Jeon [8]. But our multiplier saves about 50, 27, and 36% time complexities as compared to Huang [7], Kim-Kim [9] and Choi-Lee [10], respectively.

5 Conclusion

In this paper, we propose a semi-systolic architecture for MM over finite fields. We induced an efficient algorithm which is highly suitable for the design of parallel pipelined structures. Our algorithm enabled the computation to share the hardware architecture so that we reduced not only time complexity but also hardware complexity by nearly 40% compared to the recent study. We expect that our architecture can be efficiently used for various applications including crypto coprocessor design, which demand high-speed computation, for security purposes.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments in improving our manuscript. The present research was conducted by the research fund of Dankook University in 2014.