Efficient bit-parallel systolic architecture for multiplication and squaring over $GF(2^m)$

Kee-Won Kim$^a$ and Seung-Hoon Kim$^b$

Department of Applied Computer Engineering, Dankook University, Yongin, Korea

a) nirkim@dankook.ac.kr
b) edina@dankook.ac.kr, Corresponding Author

Abstract: In this letter, we propose a parallel-in-parallel-out systolic array for concurrently computing multiplication and squaring over $GF(2^m)$. For $m \geq 400$, the proposed bit-parallel systolic array can save about 50% time complexity as compared to the corresponding existing structure. The proposed array can be used as a core circuit for various applications. Also our architecture is well suited to VLSI implementation as well.

Keywords: finite field, modular multiplication, squaring, systolic array, cryptography

Classification: Integrated circuits

References

1 Introduction

Efficient design and implementation for finite field arithmetic is very important for many practical applications such as error-correcting codes and cryptography [1, 2]. The multiplication is one of the basic arithmetic operations over finite fields [3, 4, 5]. Especially, the modular exponentiation is an essential part of cryptographic algorithms. The modular multiplication and squaring can be performed simultaneously for computing fast exponentiation [6, 7, 8, 9, 10]. To compute modular multiplication and squaring concurrently over finite fields, Choi and Lee [8] proposed a systolic multiplier/squarer and Kim and Lee [9] proposed an efficient unified semi-systolic arrays. Recently, Kim et al. [10] proposed a combined algorithm for multiplication and squaring for fast exponentiation over finite fields.

In this letter, we modify Kim et al.’s algorithm [10] to reduce the cell delay and latency of array and propose a bit-parallel systolic array for concurrently performing multiplication and squaring over finite field \(GF(2^m)\). The proposed architecture is compared with related bit-parallel systolic array and has better features in cell delay, latency, and AT (area-time) complexity.

2 Proposed bit-parallel systolic array

2.1 The conventional algorithm

We briefly discuss Kim et al.’s algorithm [10] for concurrently performing multiplication and squaring over \(GF(2^m)\). Suppose that the finite field \(GF(2^m)\) is generated by an irreducible polynomial \(G = x^m + \sum_{j=0}^{m-1} g_j x^j\) over \(GF(2)\). The polynomial basis \(\{1, x, \cdots, x^{m-2}, x^{m-1}\}\) is used to represent the field elements. Any two arbitrary elements \(A\) and \(B\) over \(GF(2^m)\) can be represented as \(A = \sum_{j=0}^{m-1} a_j x^j\) and \(B = \sum_{j=0}^{m-1} b_j x^j\), where \(a_j\) and \(b_j\) \(\in\) \(\{0, 1\}\).

Since \(x\) is a root of \(G(x)\), one can have \(x^m = \sum_{j=0}^{m-1} g_j x^j\) and \(x^{m+1} = \sum_{j=1}^{m-1} (g_{m-1}g_j + g_{j-1})x^j + g_{m-1}g_0 \equiv G' = \sum_{j=0}^{m-1} g_j x^j\). Assume that \(x^{m+1} \mod G\) and \(x^m \mod G\) are given in advance. Let \(k = [m/2]\) and \(l = [m/2]\). Therefore, the \(P = AB \mod G\) and \(S = AA \mod G\) can be expressed as follows:

\[
P = \sum_{i=0}^{m-1} b_i A x^i \mod G = \sum_{i=0}^{l-1} b_{2i} A x^{2i} \mod G + x \sum_{i=0}^{k-1} b_{2i+1} A x^{2i} \mod G = Q + xR \mod G, \tag{1}
\]

\[
S = \sum_{i=0}^{m-1} a_i A x^i \mod G = \sum_{i=0}^{l-1} a_{2i} A x^{2i} \mod G + x \sum_{i=0}^{k-1} a_{2i+1} A x^{2i} \mod G = T + xU \mod G, \tag{2}
\]

where \(Q = \sum_{i=0}^{l-1} b_{2i} A x^{2i} \mod G\), \(R = \sum_{i=0}^{k-1} b_{2i+1} A x^{2i} \mod G\), \(T = \sum_{i=0}^{l-1} a_{2i} A x^{2i} \mod G\), and \(U = \sum_{i=0}^{k-1} a_{2i+1} A x^{2i} \mod G\).
We can observe that the computations of $Q$, $R$, $T$, and $U$ require $Ax^2i$ in common. We define $A^0 = Ax^{2i}$, for $0 \leq i \leq l - 1$. Then, the equations can be expressed as $A^0 = \sum_{j=0}^{m-1} d_j^0 x^j \mod G$, where $A^0 = A$. Then, based on $x^m$ and $x^{m+1}$, $A^i$ can be expressed as $A^i = A^{(i-1)} x^2 \mod G = \sum_{j=0}^{m-1} (d_{j-2}^{(i-1)} + d_{j-1}^{(i-1)}) g_j + d_{m-1}^{(i-1)} g_m^i$, where $A^0 = A$, $d_{l-2}^{(i-1)} = d_{l-1}^{(i-1)} = 0$, and $1 \leq i \leq l - 1$.

The coefficient of $A^i$ is as follows:

$$a_j^i = a_j^{(i-1)} + d_{m-1}^{(i-1)} g_j + d_{m-1}^{(i-1)} g_j',$$

where $a_j^0 = a_j$, $a_j^{(i-1)} = 0$, and $1 \leq i \leq l - 1$.

Using $A^i$, $Q$, $R$, $T$, and $U$ are represented as $Q = \sum_{i=1}^{l} b_{2i-1} A^{(i-1)}$, $R = \sum_{i=1}^{k} b_{2i-1} A^{(i-1)}$, $T = \sum_{i=1}^{l} a_{2i-1} A^{(i-1)}$, and $U = \sum_{i=1}^{k} a_{2i-1} A^{(i-1)}$.

The recurrence equations of $Q$, $R$, $T$, and $U$ can be formulated as $Q^0 = Q^{(i-1)} + b_{2i-1} A^{(i-1)}$, $R^0 = R^{(i-1)} + b_{2i-1} A^{(i-1)}$, $T^0 = T^{(i-1)} + a_{2i-1} A^{(i-1)}$, and $U^0 = U^{(i-1)} + a_{2i-1} A^{(i-1)}$, where $Q^0 = R^0 = T^0 = U^0 = 0$, $Q^i = \sum_{i=0}^{m-1} q_j^i x^j$, $R^i = \sum_{i=0}^{m-1} r_j^i x^j$, $T^i = \sum_{i=0}^{m-1} t_j^i x^j$, and $U^i = \sum_{i=0}^{m-1} u_j^i x^j$ are intermediate results.

Therefore, the coefficients of $Q^0$, $R^0$, $T^0$, and $U^0$ are as follows:

$$q_j^0 = q_j^{(i-1)} + b_{2i-1} a_j^{(i-1)}, \quad \text{for } 1 \leq i \leq l,$$

$$r_j^0 = r_j^{(i-1)} + b_{2i-1} a_j^{(i-1)}, \quad \text{for } 1 \leq i \leq k,$$

$$t_j^0 = t_j^{(i-1)} + a_{2i-1} a_j^{(i-1)}, \quad \text{for } 1 \leq i \leq l,$$

$$u_j^0 = u_j^{(i-1)} + a_{2i-1} a_j^{(i-1)}, \quad \text{for } 1 \leq i \leq k,$$

where $q_j^0 = r_j^0 = t_j^0 = u_j^0 = 0$ and $0 \leq j \leq m - 1$. The equations from (4) to (7) can be simultaneously executed because there are no data dependency between computations of $Q$, $R$, $T$, and $U$.

After computing $Q$, $R$, $T$, and $U$, we finally require computing $P = Q + xR \mod G$ and $S = T + xU \mod G$ to obtain the result of multiplication and square. Therefore, $P$ and $S$ are $P = Q^0 + xR^0 \mod G = \sum_{i=0}^{m-1} q_j^0 + r_j^{(k)} + r_{j-1}^{(k)} x^j$, and $S = T^0 + xU^0 \mod G = \sum_{i=0}^{m-1} t_j^0 + u_j^{(k)} + u_{j-1}^{(k)} x^j$, where $r_{i-1} = 0$ and $a_{i-1}^0 = 0$.

Therefore, the coefficients of $P$ and $S$ can be computed as follows:

$$p_j = q_j^0 + r_j^{(k)} + r_{j-1}^{(k)},$$

$$s_j = t_j^0 + u_j^{(k)} + u_{j-1}^{(k)},$$

where $r_{i-1}^{(k)} = 0$ and $a_{i-1}^{(k)} = 0$.

For the more detailed algorithm, refer to [10]. Kim et al. have proposed only the algorithm without implementation. We now implement a signal flow graph (SFG) array for realizing their algorithm for computing multiplication and squaring in parallel. It is depicted in Fig. 1(a) and consists of $m \times m/2$ $W_j^0$ cells and $m V_j$ cells, where the detailed circuits of cells are presented in Fig. 1(b) and (c). Each $W_j^0$ cell employs six 2-input AND gates and six 2-input XOR gates in order to compute $d_j^0$, $q_j^0$, $r_j^{(k)}$, $t_j^0$, and $u_j^{(k)}$ from (3) to (7). Each $V_j$ cell is composed of two 2-input AND gates and four 2-input XOR gates to compute $p_j$ and $s_j$ in (8) and (9). The SFG array involves bi-directional data flow. If we implement a systolic array using the SFG, the required latency is about $3m$ clock cycles and a cycle of cells.
requires the delay of $T_A + 2T_X$, where $T_A$ and $T_X$ represent the delays of a two-input AND gate and a two-input XOR gate, respectively.

### 2.2 Proposed parallel-in parallel-out systolic array

We propose an efficient parallel-in parallel-out systolic array modifying the SFG array in Fig. 1(a). To get rid of the bi-directional data flow and reduce the latency of SFG array, we can merge two adjacent cells in the horizontal direction to design a new cell. For simplicity of derivation, we assume that $m$ is even. Each new $\tilde{W}_n^{(i)}$ cell computes following equations, where $0 \leq n \leq m/2 - 1$.

$$
\begin{align*}
    a_{2n+1}^{(i)} &= a_{2n-1}^{(i)} + a_{m-2}^{(i)}g_{2n+1} + a_{m-1}^{(i)}g_{2n}, \\
    q_{2n+1}^{(i)} &= q_{2n}^{(i)} + b_{2n+1}^{(i)}a_{m-1}^{(i)}g_{2n}, \\
    r_{2n+1}^{(i)} &= r_{2n}^{(i)} + b_{2n+1}^{(i)}a_{m-1}^{(i)}g_{2n}, \\
    u_{2n+1}^{(i)} &= u_{2n}^{(i)} + a_{2n}^{(i)}g_{2n+1} + a_{2n-1}^{(i)}g_{2n}.
\end{align*}
$$

In (10), $a_{2n+1}^{(i)}$, $q_{2n+1}^{(i)}$, $r_{2n+1}^{(i)}$, and $u_{2n+1}^{(i)}$ in (10), (11), and (12) and the critical path is $T_A + 2T_X$. In (10), $a_{2n+1}^{(i)}$, $a_{2n-1}^{(i)}$, and $a_{2n-2}^{(i)}$ are required to compute $a_{2n}^{(i)}$ and $a_{2n-1}^{(i)}$, respectively. Since $a_{2n-1}^{(i)}$ and $a_{2n-2}^{(i)}$ come from $\tilde{W}_{n-1}^{(i)}$ cell and is inputted to $\tilde{W}_n^{(i)}$, $q_{2n}^{(i)}$ and $r_{2n}^{(i)}$ can be used after one clock cycle than other input values in $\tilde{W}_n^{(i)}$ cell. Therefore, to reduce the cell delay...
of cells, we can insert two one-bit latches (denoted by ‘■’ in figures) after the computations of \( a_{m-2}^{(i-1)} s_{2n+1} + a_{m-1}^{(i-1)} s_{2n+1} \) and \( d_{m-2}^{(i-1)} s_{2n} + d_{m-1}^{(i-1)} s_{2n} \) in \( \hat{W}_n^{(i)} \) cell. Similarly, we can insert four one-bit latches after the computations of \( q_{2n+1}^{(l)} + r_{m-1}^{(k)} s_{2n+1} \), \( q_{2n}^{(l)} + r_{m-1}^{(k)} s_{2n} \), \( t_{2n+1}^{(l)} + u_{m-1}^{(k)} s_{2n+1} \), and \( t_{2n}^{(l)} + u_{m-1}^{(k)} s_{2n} \) in \( \hat{V}_n \) cell computing equations (11) and (12).

By applying the cut-set systolisation techniques and above our modification to the SFG array in Fig. 1(a), we can derive a parallel-in parallel-out systolic array for concurrently computing multiplication and squaring over \( GF(2^m) \). The result is given in Fig. 2(a), where the detailed circuits of cells are depicted in Fig. 2(b) and (c). The array involves unidirectional data flow and can provide the maximum throughput of one result per clock cycle after an initial delay of \( m + m/2 + 1 \) clock cycles.

### 3 Analysis and conclusion

For a comparison of the time and area complexity, we utilize the SAMSUNG STD 150 0.13 m 1.2 V CMOS Standard Cell Library. Based on this library, we estimated the time and area complexity of the proposed and the related arrays. We adopt that \( A_{AND} = 6.68 \), \( T_{AND} = 0.094 \) ns, \( A_{XOR} = 12.00 \), \( T_{XOR} = 0.167 \) ns, \( A_{LATCH} = 16.00 \), and \( T_{LATCH} = 0.157 \) ns, where \( A_{GATEn} \) denotes transistor count of an \( n \)-input gate and \( T_{GATEn} \) denotes the propagation delay of an \( n \)-input gate.

A comparison between the proposed and the related bit-parallel systolic array is given in Table I. For \( m \geq 400 \), the proposed bit-parallel systolic array at least can save not only 7.28% area complexity but also 49.92% time complexity as compared...
to the Choi-Lee’s bit-parallel systolic array [8]. Consequently, the proposed bit-parallel systolic array at least can save 53.56% AT complexity as compared to the Choi-Lee’s array [8].

In this letter, a bit-parallel systolic architecture for concurrently performing multiplication and squaring for fast modular exponentiation over finite fields has been presented. 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

The present research was conducted by the research fund of Dankook University in 2016.

Table 1. Comparison of the bit-parallel systolic arrays of multiplication and squaring

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Choi-Lee [8]</th>
<th>Fig. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND$_2$</td>
<td>$3m^2$</td>
<td>$3m^2 + 2m$</td>
</tr>
<tr>
<td>XOR$_2$</td>
<td>$3m^2$</td>
<td>$3m^2 + 4m$</td>
</tr>
<tr>
<td>Latch</td>
<td>$10m^2$</td>
<td>$9m^2 + 3m$</td>
</tr>
<tr>
<td>Total transistors</td>
<td>$216.04m^2$</td>
<td>$200.04m^2 + 109.36m$</td>
</tr>
</tbody>
</table>

| Time complexity |
|-----------------|----------------|
| Cell delay      | 0.418          |
| Latency         | $3m$           |
| Total delay     | $1.254m$       |

<table>
<thead>
<tr>
<th>Area-Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT complexity</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>