EFFICIENT SYSTOLIC MULTIPLICATIONS IN COMPOSITE FIELDS FOR CRYPTOGRAPHIC SYSTEMS

HAIBO YI

School of Computer Engineering Shenzhen Polytechnic
Shenzhen 518055, China

ABSTRACT. Multiplications in finite fields are playing a key role in areas of cryptography and mathematic. We present approaches to exploit systolic architecture for multiplications in composite fields, which are expected to reduce the time-area product substantially. We design a pipelined architecture for multiplications in composite fields \( GF((2^n)^2) \), where \( n \) is a positive integer. Besides, we design systolic architectures for multiplications and additions in finite fields \( GF(2^n) \). By integrating main improvements and other minor optimizations for multiplications in \( GF((2^n)^2) \), the non-pipelined versions of our design takes \( 8n + 4 \) AND gates and \( 8n \) XOR gates to compute multiplications with the executing time of \( nT_{AND} + 4nT_{XOR} \), where \( T_{AND} \) and \( T_{XOR} \) are delays of AND and XOR gates respectively; with the aid of pipelining, the pipelined version of our design has a throughput rate of one result per \( 2nT_{XOR} \). Other words, the time complexity and area complexity of our design are \( O(n) \). Thus, the complexity of time-area product of our design is \( O(n^2) \). Experimental results and comparisons show that our design provides significant reductions in executing time and area of multiplications.

1. Introduction. Finite field operations have gained increasing importance due to the fact that they are fundamental operations frequently encountered in areas of mathematic [1, 5, 23, 26, 27, 29, 36–38] and engineering [4, 7, 16, 28]. Generally, operations are computed via using a specific basis [2,3,6,9,10,13,19–22,25,32,34,35], i.e. polynomial basis, normal basis, triangular basis, dual basis and other bases. Besides, systolic architectures are used widely in finite field operations since they increase the efficiency of operations by combining the concept of parallel processing and pipelining into a single concept [8,11–15,17,18,24,30,31,33].

Among operations in finite fields, multiplications are crucial to many cryptographic systems, e.g. Multivariate Public Key Cryptography (MPKC) [23], AES [5] and CLEFIA [27]. AES and CLEFIA use a Substitution-Box (S-Box) [5], which is generated by using multiplications and inversions in a finite field; MPKC uses a great many multiplications in a finite field during encrypting, decrypting, signature generation and verification. Besides, multiplications are widely used in solving systems of linear equations [7]. Thus, it is desirable to improve multiplications in finite fields due to the fact that they are playing an importance role in the implementations of many cryptographic systems and other engineering systems.

2010 Mathematics Subject Classification. 12Y05.

Key words and phrases. Multiplication, multiplier, composite field, finite field, systolic.

* Corresponding author: Haibo Yi.
Among finite fields, composite fields are popular choices for implementations of cryptographic systems since they allow efficient hardware implementation in terms of the silicon area as well as the execution time. We present approaches to exploit systolic architecture for multiplications in composite fields, which are expected to reduce the time-area product substantially in finite fields.

Main improvements of this paper with known results are presented as follows. First, we design a systolic architecture for multiplications in finite fields \( GF(2^n) \), where \( n \) is a positive integer. Second, we design a systolic architecture for additions in \( GF(2^n) \). Third, we design a pipelined architecture for multiplications in composite fields \( GF((2^n)^2) \).

By integrating above improvements and other minor optimizations, non-pipelined versions and pipelined versions of multiplications in \( GF((2^n)^2) \) are designed. The non-pipelined versions of our design have the executing time of \( nT_{\text{AND}} + 4nT_{\text{XOR}} \), where \( T_{\text{AND}} \) and \( T_{\text{XOR}} \) are delays of AND and XOR gates respectively; the pipelined version of our design has a throughput rate of one result per \( 2nT_{\text{XOR}} \). Besides, it takes \( 8n + 4 \) AND gates and \( 8n \) XOR gates to compute a multiplication. Other words, the time complexity and area complexity of our design are \( O(n) \). Thus, the complexity of time-area product of our design is \( O(n^2) \).

Our design is well suited for Application Specific Integrated Circuit (ASIC), Altera and Xilinx Field Programmable Logic Arrays (FPGAs). We back up the claims with implementations of our design on TSMC-0.18\( \mu \)m standard cell CMOS ASIC and Altera, Xilinx FPGAs respectively. Experimental results and comparisons with other multiplications in \([9,20,22,34]\) show that our design provides significant reductions in executing time and area.

The rest of this paper is organized as follows: in Section 2, we introduce the background information; in Section 3, we propose systolic multiplications in composite fields; in Section 4, we present timing and area analysis of our design; in Section 5, we present implementations of our design; in Section 6, we compare our implementations with related methods; in Section 7, we present conclusions of this paper.

2. Preliminaries. In mathematics, a finite field is a field that contains a finite number of elements. As with any field, it is a set on which the basic operations of addition, multiplication and inversion have been defined.

Common, the prime field \( GF(p) \) of order and characteristic \( p \) is constructed as the integers modulo \( p \), where \( p \) is a prime number. Thus, the elements are represented by integers in the range \( 0, \ldots, p-1 \). Given a prime power \( q = 2^n \) with \( n > 1 \), the field \( GF(q) \) can be explicitly constructed. One chooses first an irreducible polynomial \( f \) in \( GF(2)[X] \) of degree \( n \). Then the quotient ring \( GF(q) = GF(2)[X]/f \) of the polynomial ring \( GF(2)[X] \) by the ideal generated by \( f \) is a field of order \( q \).

Composite field is a special case of finite field. The elements of composite fields \( GF((2^n)^m) \) can be represented in the standard base as polynomials with a maximum degree of \( m-1 \) in \( GF(2^n) \). The two pairs \( \{GF(2^n), p(x)\} \) and \( \{GF((2^n)^m), q(y)\} \) constitute a composite field if \( GF(2^n) \) is constructed from \( GF(2) \) by \( p(x) \) and \( GF((2^n)^m) \) is constructed from \( GF(2^n) \) by \( q(y) \), where \( p(x) \) and \( q(y) \) are field polynomials of degree \( n \) and \( m \) respectively. Composite fields \( GF((2^n)^m) \) are isomorphic to fields \( GF(2^l) \) if \( l = n \times m \). \( GF((2^n)^2) \) is a special case of composite fields, where \( m = 2 \). All finite fields \( GF(2^l) \) can be expressed as the forms \( GF((2^n)^2) \) if \( l \) is even.
3. Efficient systolic multiplications in composite fields.

3.1. Pipelined architecture for multiplications in composite fields. We propose a pipelined architecture for multiplications in composite fields, which is depicted in Fig. 1. It computes multiplications in $GF((2^n)^2)$, which is illustrated as follows.

(1) field elements $a_h, a_l, b_h, b_l$ and $e$ in $GF(2^n)$ are the inputs of the architecture;
(2) $a(x) = a_h x + a_l$ and $b(x) = b_h x + b_l$ are two operands of multiplication in $GF((2^n)^2)$;
(3) $q(x) = x^2 + x + e$ is the irreducible polynomial in $GF((2^n)^2)$;
(4) field elements $c_h$ and $c_l$ in $GF(2^n)$ are the outputs of the architecture;
(5) $c(x) = c_h x + c_l$ is the expecting multiplication results of $a(x)$ and $b(x)$ in $GF((2^n)^2)$;
(6) the architecture includes three stages, i.e. Stage0, Stage1 and Stage2;
(7) $a_h, a_l, b_h, b_l$ are input to Stage0, $e$ is input to Stage1;
(8) $c_h, c_l$ are the output of Stage2;
(9) the architecture uses four MUL components and four ADD components, where MUL and ADD are multiplication and addition components in $GF(2^n)$;
(10) components in the architectures are designed with AND gates and XOR gates;
(11) elements are sent to the architecture and the multiplications are computed with the aid of pipelining.

Based on the architecture, the multiplications are computed via pipelining as follows.

(1) period 0: $a_h, a_l, b_h, b_l$ are sent to Stage0;
(2) period 1: $a_h', a_l', b_h', b_l'$ are sent to Stage0, $e$ is sent to Stage1;
(3) period 2: $a_h'', a_l'', b_h'', b_l''$ are sent to Stage0, $e'$ is sent to Stage1, $c_h, c_l$ are generated via Stage2;
(4) period 3: $a''_h$, $a''_i$, $b''_h$, $b''_i$ are sent to Stage0, $e''$ is sent to Stage1, $c'''_h$, $c'''_i$ are generated via Stage2;
(5) period 4: $a''''_h$, $a''''_i$, $b''''_h$, $b''''_i$ are sent to Stage0, $e''''$ is sent to Stage1, $c''''_h$, $c''''_i$ are generated via Stage2;
(6) ...

We suppose that multiplications of $c(x) = a(x) \times b(x)$, $c(x)' = a(x)' \times b(x)'$, $c(x)'' = a(x)'' \times b(x)''$ are required to compute, where $a(x) = a_0x + a_1$, $b(x) = b_0x + b_1$, $a(x)' = a'_0x + a'_1$, $b(x)' = b'_0x + b'_1$, $a(x)'' = a''_0x + a''_1$, $b(x)'' = b''_0x + b''_1$ are operands of multiplications. It can be observed that using the architecture with the pipelining, the multiplications are computed within a period, e.g. $c(x) = c_hx + c_i$, $c(x)' = c'_hx + c'_i$, $c(x)'' = c''_hx + c''_i$ are generated in period 2, period 3, period 4, respectively.

3.2. Systolic Component MUL: Multiplications in $GF(2^n)$. As described in Fig. 1, the pipelined architecture for multiplications in $GF((2^n)^2)$ includes MUL components, which is use to compute multiplications in $GF(2^n)$. We design MUL in Fig. 2, which is illustrated as follows.

(1) it uses three different kinds of cells, i.e. A, B and C;
(2) it includes $n$ cells of A, i.e. $A_0, A_1, \ldots, A_{n-1}$;
(3) it includes a cell of B;
(4) it includes $n$ cells of C, i.e. $C_0, C_1, \ldots, C_{n-1}$;
(5) $f(x)$ and $g(x)$ are elements in $GF(2^n)$ and the expecting multiplication result of $f(x)$ and $g(x)$ is $h(x)$, which is an element in $GF(2^n)$;
(6) $f(x)$, $g(x)$ and $h(x)$ are represented as $f_0, f_1, \ldots, f_{n-1}$, $g_0, g_1, \ldots, g_{n-1}$ and $h_0, h_1, \ldots, h_{n-1}$, respectively, where $f_i, g_i, h_i$ are elements in $GF(2)$, i.e. 0 or 1, $i = 0, 1, \ldots, n-1$;
(7) $f_0, f_1, \ldots, f_{n-1}$ are stored in cells of $A_0, A_1, \ldots, A_{n-1}$, respectively;
(8) $g_0, g_1, \ldots, g_{n-1}$ are sent to cells of $A_0, A_1, \ldots, A_{n-1}$ continuously;
(9) the computation results of cells of $C_0, C_1, \ldots, C_{n-1}$ are stored in $h_0, h_1, \ldots, h_{n-1}$, respectively.

**Figure 2.** Systolic Component MUL in $GF(2^n)$
It can be observed from Fig. 2, cell $A$ has three ports, i.e. $a_0$, $a_1$ and $a_2$, where $a_0$ is input and $a_1$, $a_2$ are outputs. In cell $A_i$, the computation is illustrated as follows.

(1) if $a_0 = g_j$, $a_1 = g_j$, $g_j$ is passed to the next cell (if it exists), i.e. $A_{i+1}$;
(2) if $a_0 = g_j$, $a_2 = g_j \times f_i$, $g_j \times f_i$ is passed to the port $b_i$ of cell $B$;
(3) $g_j \times f_i$ is a multiplication in $GF(2)$ via using an AND gate.

It can be observed from Fig. 2, cell $B$ has $2n$ ports, i.e. $b_0, b_1, \ldots, b_{n-1}$ and $d_0, d_1, \ldots, d_{n-1}$, where $b_0, b_1, \ldots, b_{n-1}$ are inputs and $d_0, d_1, \ldots, d_{n-1}$ are outputs. In cell $B$, the computation is illustrated as follows.

(1) $p(x) = x^n + p_{n-1}x^{n-1} + \cdots + p_1x + 1$ is the irreducible polynomial in $GF(2^n)$, where $p_{n-1}, p_{n-2}, \ldots, p_1$ are elements in $GF(2)$, i.e. 0 or 1;
(2) for $i = 0, 1, \ldots, 2(n-1)$, $x^i \mod p(x) = \sum_{j=0}^{n-1} v_{ij}x^j$ is computed in advance, where $\mod$ is a modular operation;
(3) it uses an accumulator and its initial value is $k = 0$, if a new value is received via the input port, $k = k + 1$;
(4) when a new $b_i$ is received, for $t = 0, 1, \ldots, n-1$, if $v_{(k+i)t} = 1$, $b_i$ is sent to $d_k$ and $d_k$ is sent to cell $C_k$.

It can be observed from Fig. 2, cell $C$ has a port, i.e. $c$, where $c$ is input. In cell $C_i$, the computation is illustrated as follows.

(1) when a new $c$ is received, $h_i = h_i + c$ is computed, where $+$ is an addition in $GF(2)$ via using a XOR gate.

Based on our design, we depict the systolic multiplication in $GF(2^n)$ in Fig. 3. Cell $A$ uses AND gates to compute multiplication in $GF(2)$, cell $B$ is a selector and cell $C$ use XOR gates to compute addition in $GF(2)$. Thus, the architecture uses $n$ AND gates and $n$ XOR gates, and it takes $2n$ clock cycles to perform a multiplication.

3.3. Systolic components $ADD$: Additions in $GF(2^n)$. As described in Fig. 1, the pipelined architecture for multiplications in $GF((2^n)^2)$ includes $ADD$ components, which is use to compute additions in $GF(2^n)$. We design $ADD$ in Fig. 4, which is illustrated as follows.
(1) it uses a cell, i.e. $A$;
(2) $f(x)$ and $g(x)$ are elements in $GF(2^n)$ and the expecting addition result of $f(x)$ and $g(x)$ is $h(x)$, which is an element in $GF(2^n)$;
(3) $f(x)$, $g(x)$ and $h(x)$ are represented as $f_0, f_1, \ldots, f_{n-1}$, $g_0, g_1, \ldots, g_{n-1}$ and $h_0, h_1, \ldots, h_{n-1}$, respectively, where $f_i, g_i, h_i$ are elements in $GF(2)$, i.e. 0 or 1, $i = 0, 1, \ldots, n-1$;
(4) $f_0, f_1, \ldots, f_{n-1}$ and $g_0, g_1, \ldots, g_{n-1}$ are sent to cell $A$ continuously;
(5) the computation results of cells of $A$ stored in $h_0, h_1, \ldots, h_{n-1}$, respectively.

Cell $A$ uses a XOR gate to compute additions in $GF(2)$, e.g. $h_0 = f_0 + g_0$, $h_1 = f_1 + g_1$, $h_2 = f_2 + g_2$. Thus, the architecture uses a XOR gate, and it takes $n$ clock cycles to perform an addition.

4. Timing and area analysis. According to our design, the architecture for multiplications can be designed with AND and XOR gates. Thus, in the following, we analyze the timing and area of our design in terms of AND and XOR gates.

Based on the illustration of our architecture, we analyze and summarize the executing time and area for a multiplication in $GF((2^n)^2)$ in Table 1, which shows that it takes $8n + 4$ AND gates and $8n$ XOR gates to compute a multiplication with the executing time of $nT_{AND} + 4nT_{XOR}$. Thus, the executing time and area of non-pipelined version of our design is logarithmic in the field size. Other words, the time complexity and area complexity of non-pipelined multiplications are $O(n)$.

In addition, we can use pipelining in our design to accelerate multiplications in composite fields. Table 2 shows that multiplications in $GF((2^n)^2)$ are computed with a throughput rate of one result per $2nT_{XOR}$ by using pipelining. Thus, the executing time of multiplications is reduced by more than 50% by using pipelining.

5. Implementation. According to the analysis in Section 4, our design takes 4 cells, including $8n + 4$ AND gates and $8n$ XOR gates, and 5n clock cycles to compute a multiplication with the executing time of $nT_{AND} + 4nT_{XOR}$ in $GF((2^n)^2)$. Besides, it computes multiplications with a throughput rate of one result per $2nT_{XOR}$

**Table 1. Executing Time and Area for Non-Pipelined Multiplication in $GF((2^n)^2)$**

| Stage | Clock Cycle | Executing Time | Area (Logic Gates) |
|-------|-------------|----------------|--------------------|
| 0     | 2n          | $2nT_{XOR}$    | $4n + 2$ AND gates, $4n$ XOR gates |
| 1     | 2n          | $2nT_{XOR}$    | $4n$ AND gates, $4n$ XOR gates |
| 2     | $n$         | $nT_{AND}$     | 2 AND gates |
| Total | 5n          | $nT_{AND} + 4nT_{XOR}$ | $8n + 4$ AND gates, $8n$ XOR gates |
by using pipelining. We evaluate and summary the performance of our design in Table 3.

In order to prove that our architectures have high throughput of multiplications and low area on different devices, Hardware Description Language (Verilog HDL) code for modeling the design has been implemented on ASICS, Altera FPGAs and Xilinx FPGAs respectively. Since pipelining is used to gain a high throughput in our implementations, they consist of non-pipelined and pipelined versions.

5.1. Implementations on ASICS. We implement the non-pipelined and pipelined versions of our design in $GF((2^n)^2)$ on TSMC-0.18μm standard cell CMOS ASICs respectively. We use Synopsys Design Vision, which is a GUI for Synopsys Design Compiler tools. The map effort is set to medium. We report time (ns), throughput (ns) and area ($\mu m^2$) for implementations in composite fields.

We summary ASIC implementations of our design for different composite fields in Table 4, which clearly indicates that they achieve high throughput and low area of multiplications in $GF((2^n)^2)$.

5.2. Implementations on Altera FPGAs. In order to prove that our design is applicable to Altera FPGA devices, we implement the non-pipelined and pipelined versions in $GF((2^n)^2)$ on Altera FPGA (Stratix II EP2S180F1508C3) respectively. Synthesis, Fitting and Place & Route have been carried out by using Quartus II 64-bit version 8.0, which is a GUI for Altera synthesis software. ModelSim PE has been used to perform the circuit simulations. We report time (ns), throughput (ns), area (combinational ALUTs) and the utilization rate of combinational ALUTs for implementations in composite fields.

We summary Altera FPGA implementations of our design for different composite fields in Table 4, which clearly indicates that they achieves high throughput and low area of multiplications in $GF((2^n)^2)$.

5.3. Implementations on Xilinx FPGAs. In order to prove that our design is applicable to Xilinx FPGA devices, we implement the non-pipelined and pipelined versions in $GF((2^n)^2)$ on Xilinx FPGA (Virtex 5 XC5VLX110T) respectively. Synthesis, Fitting and Place & Route have been carried out by using ISE Design Suite version 14.4, which is a GUI for Xilinx synthesis software. ModelSim PE has been used to perform the circuit simulations. We report the time (ns), throughput (ns), area (Slice LUTs) and the utilization rate of slice LUTs for implementations in composite fields.

We summary Xilinx FPGA implementations of our design for different composite fields in Table 4, which clearly indicates that they achieves high throughput and low area of multiplications in $GF((2^n)^2)$.

Table 2. Executing Time for Pipelined Multiplications in $GF((2^n)^2)$

| Input | Starting Time | Ending Time       |
|-------|---------------|-------------------|
| $a, b$ | 0             | $nT_{AND} + 4nT_{XOR}$ |
| $a', b'$ | $2nT_{XOR}$ | $nT_{AND} + 6nT_{XOR}$ |
| $a'', b''$ | $4nT_{XOR}$ | $nT_{AND} + 8nT_{XOR}$ |
| $a''', b''''$ | $6nT_{XOR}$ | $nT_{AND} + 10nT_{XOR}$ |
| ... | ...           | ...               |
Comparison.

ASIC (TSMC-0.18 µm CMOS) implementations: T0 (ns) is the executing time of non-pipelined designs; T1 (ns) is the executing time of pipelined designs; A0 is area. 

Xilinx FPGA (Virtex 5 XC5VLX110T) implementations: T4 (ns) is the executing time of non-pipelined designs; T5 (ns) is the executing time of pipelined designs; A2 is slice LUTs; U1 is the utilization rate of combinational ALUTs.

Table 3. Performance Evaluation of Our Design for Multiplications in GF((2^n)^2)

| Field | Clock Cycle | Executing Time | Throughput | Cells | Area (Logic Gates) |
|-------|-------------|----------------|------------|-------|--------------------|
| GF((2^n)^2) | 5n | nT_{AND} + 4nT_{XOR} | 2nT_{XOR} | 4 | 8n + 4 AND gates |

Table 4. ASIC, Altera FPGA and Xilinx FPGA Implementations

| Finite Field | T0 | T1 | A0 | T2 | T3 | A1 | U0 | T4 | T5 | A2 | U1 |
|--------------|----|----|----|----|----|----|----|----|----|----|----|
| GF((2^n)^2) | 1.4 | 0.6 | 478.8 | 16.8 | 7.1 | 48 | ≈1% | 17.8 | 7.2 | 45 | ≈1% |
| GF((2^n)^2) | 2.8 | 1.2 | 904.4 | 34.4 | 14.1 | 89 | ≈1% | 35.9 | 14.2 | 83 | ≈1% |
| GF((2^n)^2) | 9.1 | 3.7 | 2819.6 | 116.4 | 44.6 | 245 | <1% | 117.3 | 46.3 | 232 | <1% |
| GF((2^n)^2) | 11.9 | 4.8 | 3670.8 | 147.7 | 58.4 | 313 | <1% | 152.6 | 60.8 | 298 | <1% |
| GF((2^n)^2) | 21.7 | 8.8 | 6650.2 | 271.1 | 102.4 | 557 | <1% | 275.9 | 110.4 | 537 | <1% |
| GF((2^n)^2) | 25.9 | 10.3 | 7926.8 | 321.0 | 125.9 | 634 | <1% | 329.3 | 131.7 | 614 | <1% |
| GF((2^n)^2) | 42.7 | 17.1 | 13034.2 | 541.4 | 211.8 | 1023 | <1% | 542.9 | 217.2 | 998 | 1.44% |
| GF((2^n)^2) | 46.7 | 18.8 | 14310.8 | 574.2 | 233.1 | 1124 | <1% | 589.6 | 238.5 | 1097 | 1.58% |
| GF((2^n)^2) | 83.3 | 33.4 | 25376.4 | 1055.9 | 411.9 | 2012 | 1.41% | 1059.3 | 423.6 | 1927 | 2.79% |
| GF((2^n)^2) | 88.9 | 33.6 | 27078.8 | 1134.6 | 446.3 | 2119 | 1.47% | 1141.6 | 456.8 | 2078 | 3.01% |

Table 5. Comparison of Our Design with Other multiplications in GF((2^n)^2)

| O(1)   | O(2)   | O(3)   | O(4)   | O(5)   | O(n)   |
|--------|--------|--------|--------|--------|--------|
| Pan et al. [22] | Xie et al. [34] | Namin et al. [20] | Hariri et al. [9] | Ours |
| O(Time) | n^2   | n^2   | n^2   | n^2   | n^2   |
| O(Area) | n√2n  | 2n   | 2n   | n√2n  | 2n   |
| O(Time*Area) | n^3√2n | n^3  | n^3  | n^3√2n | n^3  |

6. Comparison. Our implementations are compared with related methods for multiplications in finite fields. To be fair, we use the non-pipelined versions in comparisons due to the fact that other multiplications are non-pipelined designs.

Table 5 lists the comparison of our design with the recent proposals of multiplications in [9, 20, 22, 34], which clearly demonstrates that our design is more efficient than other multiplications, e.g. the time-area product is reduced by 75% in GF((2^{61})^2) and the time-area product is reduced by 87% in GF((2^{127})^2). Thus, our design reduce the time-area product of multiplications in GF((2^n)^2) significantly.

7. Conclusion. Composite fields are popular choices for implementations of cryptographic systems since they allow efficient hardware implementation in terms of the silicon area as well as the execution time. We present approaches to exploit systolic architecture for multiplications in composite fields.

Main improvements of this paper with known results are presented as follows. First, we design a systolic architecture for multiplications in GF(2^n). Second, we design a systolic architecture for additions in GF(2^n). Third, we design a pipelined...
EFFICIENT SYSTOLIC MULTIPLICATIONS IN COMPOSITE FIELDS

architecture for multiplications in $GF((2^n)^2)$. By integrating above improvements and other minor optimizations, non-pipelined versions and pipelined versions of multiplications in $GF((2^n)^2)$ are designed. The non-pipelined versions of our design have the executing time of $nT_{\text{AND}} + 4nT_{\text{XOR}}$; the pipelined version of our design has a throughput rate of one result per $2nT_{\text{XOR}}$. Besides, it takes $8n + 4$ AND gates and $8n$ XOR gates to compute a multiplication. Other words, the time complexity and area complexity of our design are $O(n)$. Thus, the complexity of time-area product of our design is $O(n^2)$.

Our design is well suited for ASIC, Altera and Xilinx FPGAs. We back up the claims with implementations of our design on TSMC-0.18µm standard cell CMOS ASIC and Altera (Stratix II EP2S180F1508C3), Xilinx (Virtex 5 XC5VLX110T) FPGAs respectively. Experimental results and comparisons with other multiplications show that our design provides significant reductions in executing time and area.

Acknowledgments. The authors acknowledge Natural Science Foundation of Guangdong Province, China (No. 2018A030310030), Foundation for Distinguished Young Talents in Higher Education of Guangdong, China (No. 2017GkQNCX059), Shenzhen Science and Technology Program under Grant (No. JCYJ20170306144219159), Special funds for Shenzhen Strategic Emerging Industries and Future Industrial Development (No. 20170502142224600), Science and Technology Program of Shenzhen Polytechnic (No. 601722K20018).

REFERENCES

[1] N. Ahmad and S. M. R. Hasan, Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, Integration the VLSI Journal, 46 (2013), 333–344.
[2] R. Azarderakhsh, Mozaffari-kermani M. high-performance two-dimensional finite field multiplication and exponentiation for cryptographic applications, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34 (2015), 1569–1576.
[3] S. Ballet and R. Rolland, Multiplication algorithm in a finite field and tensor rank of the multiplication, Journal of Algebra, 272 (2004), 173–185.
[4] C. Berrou and A. Glavieux, Near optimum error correcting coding and decoding: Turbo-codes, IEEE Transactions on Communications, 44 (1996), 1261–1271.
[5] D. Canright, A very compact S-box for AES, Cryptographic Hardware and Embedded Systems - CHES 2005, International Workshop, Edinburgh, Uk, August 29 - September 1, 2005, Proceedings. DBLP, 2005, 441–455.
[6] M. Cenk, C. K. Koc and F. Ozbudak, Polynomial multiplication over finite fields using field extensions and interpolation, IEEE Symposium on Computer Arithmetic, IEEE Computer Society, 2009, 84–91.
[7] A. Cichocki and R. Unbehauen, Neural networks for solving systems of linear equations and related problems, IEEE Transactions on Circuits and Systems I Fundamental Theory and Applications, 39 (1992), 124–138.
[8] M. Diab, Systolic architectures for multiplication over finite field $GF(2^m)$, International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, Springer-Verlag, 1991, 329–340.
[9] A. Hariri, Concurrent error detection in montgomery multiplication over binary extension fields, IEEE Transactions on Computers, 60 (2011), 1341–1353.
[10] M. A. Hasan, Look-up table based large finite field multiplication in memory constrained cryptosystems, Cryptography and Coding, IMA International Conference, Cirencester, Uk, December 20-22, 1999, Proceedings. DBLP, 1746 (1999), 213–221.
[11] Z. Huang, G. Q. Bai and H. Y. Chen, FPGA Implementation of Systolic Array for Modular Multiplication Using a Fine-grained Approach, Microelectronics and Computer, 2005.
[12] S. K. Jain, L. Song, K. K. Parhi, Efficient semisystolic architectures for finite-field arithmetic, IEEE Transactions on Very Large Scale Integration Systems, 6 (1998), 101–113.
1144  HAIBO YI

[13] R. Katti and J. Brennan, Low complexity multiplication in a finite field using ring representation, IEEE Transactions on Computers, 52 (2003), 418–427.
[14] C. H. Kim, C. P. Hong and S. Kwon, A digit-serial multiplier for finite field GF(2^m), IEEE Transactions on Very Large Scale Integration Systems, 13 (2005), 476–483.
[15] C. Y. Lee, W. C. Che, New bit-parallel systolic architectures for computing multiplication, multiplicative inversion and division in gf(2^m) under polynomial basis and normal basis representations, Journal of Signal Processing Systems, 52 (2008), 313–324.
[16] D. J. C. Mackay, Good error-correcting codes based on very sparse matrices, IEEE Transactions on Information Theory, 45 (1999), 399–431.
[17] P. K. Meher, Systolic formulation for low-complexity serial-parallel implementation of unified finite field multiplication over GF(2^m), IEEE International Conf. on Application -specific Systems, Architectures and Processors. 2007, 134–139.
[18] P. K. Meher, Systolic and super-systolic multipliers for finite field GF(2^m) based on irreducible trinomials, IEEE Transactions on Circuits and Systems, 55 (2008), 1031–1040.
[19] A. H. Namin, H. Wu and M. Ahmadi, Comb architectures for finite field multiplication in F(2^m), IEEE Transactions on Computers, 56 (2007), 909–916.
[20] S. H. Namin, H. Wu and M. Ahmadi, Low-power design for a digit-serial polynomial basis finite field multiplier using factoring technique, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 25 (2017), 441–449.
[21] P. Ning and Y. L. Yin, Efficient software implementation for finite field multiplication in normal basis, International Conference on Information and Communications Security, Springer-Verlag, 2001, 177–188.
[22] J. S. Pan, C. Y. Lee, P. K. Meher, Low-latency digit-serial and digit-parallel systolic multipliers for large binary extension fields, IEEE Transactions on Circuits & Systems I Regular Papers, 60 (2013), 3195–3204.
[23] A. Petzoldt, S. Bulygin and J. Buchmann, Selecting parameters for the rainbow signature scheme, Post-Quantum Cryptography, Third International Workshop, PQCrypto 2010, Darmstadt, Germany, May 25-28, 2010. Proceedings. DBLP, 2010, 218–240.
[24] A. Pincin, A new algorithm for multiplication in finite fields, IEEE Transactions on Computers, 38 (1989), 1045–1049.
[25] A. Reyhani-Masoleh and M. A. Hasan, Low complexity bit parallel architectures for polynomial basis multiplication over GF(2^m), IEEE Transactions on Computers, 53 (2004), 945–959.
[26] A. Satoh, S. Morioaka and K. Takano, et al., A compact rijndael hardware architecture with S-box optimization, Advances in Cryptology–ASIACRYPT 2001., Springer Berlin Heidelberg, 2248 (2001), 239–254.
[27] T. Shirai, K. Shibutani and T. Akishita, et al., The 128-bit blockcipher CLEFIA, Proceedings of the 14th International Conference on Fast Software Encryption, Springer-Verlag, 2007, 181–195.
[28] M. Sudan, Decoding of reed solomon codes beyond the error-correction bound, Journal of Complexity, 13 (1997), 180–193.
[29] S. Tang, H. Yi and J. Ding, et al., High-speed hardware implementation of rainbow signature on FPGAs, Post-Quantum Cryptography. Springer Berlin Heidelberg, 2011, 228–243.
[30] C. L. Wang and J. L. Lin, Systolic array implementation of multipliers for finite fields GF(2^m), IEEE Transactions on Circuits and Systems, 38 (1991), 796–800.
[31] C. W. Wu and M. K. Chang, Bit-level systolic arrays for finite-field multiplications, Journal of Signal Processing Systems, 10 (1995), 85–92.
[32] H. Wu, Bit-parallel finite field multiplier and squarer using polynomial basis, IEEE Transactions on Computers, 51 (2002), 750–758.
[33] J. Xie, J. J. He and P. K. Meher, Low latency systolic montgomery multiplier for finite field GF(2^m) based on pentanomials, IEEE Transactions on Very Large Scale Integration Systems, 21 (2013), 385–389.
[34] J. Xie, P. K. Meher and Z. H. Mao, Low-latency high-throughput systolic multipliers over GF(2^m) for NIST recommended pentanomials, IEEE Transactions on Circuits & Systems I Regular Papers, 62 (2015), 881–890.
[35] H. Yi and W. Li, Fast three-input multipliers over small composite fields for multivariate public key cryptography, International Journal of Security and Its Applications, 9 (2015), 165–178.
[36] H. Yi, W. Li and Z. Nie, Fast hardware implementations of inversions in small finite fields for special irreducible polynomials on FPGAs, *International Journal of Security and Its Applications*, 19 (2016), 109–C120.

[37] H. Yi and S. Tang, Very small FPGA processor for multivariate signatures, *Computer Journal*, 59 (2016), 1091–1101.

[38] H. Yi, S. Tang and R. Vemuri, Fast inversions in small finite fields by using binary trees, *The Computer Journal*, 59 (2016), 1102–1112.

Received July 2017; revised December 2017.

E-mail address: haiboyi@126.com