FPGA Realization of Low Register Systolic All-One-Polynomial Multipliers Over $GF(2^m)$ and Their Applications in Trinomial Multipliers

Pingxiuqi Chen, Shaik Nazeem Basha, Mehran Mozaffari-Kermani, *Member, IEEE*, Reza Azarderakhsh, *Member, IEEE*, and Jiafeng Xie, *Member, IEEE*

***Abstract*— Systolic all-one-polynomial (AOP) multipliers usually suffer from the problem of high register complexity, especially in field-programmable gate array (FPGA) platforms where the register resources are not that abundant. In this paper, we have shown that the AOP-based systolic multipliers can easily achieve low register-complexity implementations and the proposed architectures can be employed as computation cores to derive efficient implementations of systolic Montgomery multipliers based on trinomials. First, we propose a novel data broadcasting scheme in which the register complexity involved within existing AOP-based systolic multipliers is significantly reduced. We have found out that the modified AOP-based structure can be packed as a standard computation core. Next, we propose a novel Montgomery multiplication algorithm that can fully employ the proposed AOP-based computation core. The proposed Montgomery algorithm employs a novel precomputed-modular operation, and the systolic structures based on this algorithm fully inherit the advantages brought from the AOP-based core (low register complexity, low critical-path delay, and low latency) except some marginal hardware overhead brought by a precomputation unit. The proposed architectures are then implemented by Xilinx ISE 14.1 and it is shown that compared with the existing designs, the proposed designs achieve at least 61.8% and 47.6% less area-delay product and power-delay product than the best of competing designs, respectively.**

***Index Terms*— All one polynomial (AOP), finite field multiplication, irreducible trinomials, low register complexity, Montgomery algorithm, systolic structure.**

I. Introduction

FINITE field multiplication over $GF(2^m)$ is a major component of elliptic curve cryptography (ECC), which can be used in various applications, such as wearable devices and portable systems [1]–[4]. Basically, there are two bases, polynomial basis [5]–[13] and normal basis [14]–[17], which can be selected to represent the field operations. Nevertheless, in hardware realization, polynomial basis multipliers usually have simpler hardware structures than normal basis ones and hence are more widely used [8].

All-one-polynomials (AOPs) and trinomials are two of the important irreducible polynomials being used [7]–[11], [17]–[26]. The AOP-based multipliers can be used for the nearly AOP, which could be used for efficient realization of cryptosystems [24]. The AOP-based structures can be used as a kernel circuit for field exponentiation, inversion, and division architectures [24], while trinomial-based multipliers are more popular than AOP-based ones, as two trinomials have been recommended by the National Institute of Standards and Technology (NIST) for ECC implementation [5]. However, because of the complexity differences, AOPs and trinomials are not usually considered together in practical field multiplication implementations [18].

There are basically two kinds of structures for multipliers over $GF(2^m)$: systolic design and nonsystolic design. Systolic multipliers over $GF(2^m)$ based on irreducible polynomials are preferred in high-performance applications due to their features, such as modularity and regularity [5]–[11]. Systolic structures also have high register complexity, since all processing elements (PEs) in the systolic array need to use registers for pipelining [5], while nonsystolic designs usually have lower complexity with larger critical-path delay.

For practical applications, especially in field-programmable gate array (FPGA) platforms, where the register resources are not that abundant, low register-complexity systolic structures are required. Many efforts have been reported to reduce the register complexity in systolic multipliers based on irreducible AOPs and trinomials [7]–[10], [23]–[26]. A bit-parallel AOP-based systolic multiplier has been introduced in [23]. Furthermore, another efficient AOP-based design is presented in [24]. Moreover, one low-complexity systolic Montgomery AOP-based multiplier has been proposed in [7]. Lee *et al.* [7] presented a bit-parallel systolic trinomial multiplier. Meher [8] proposed efficient bit-parallel systolic and supersystolic designs. Xie *et al.* [9] introduced a low register-complexity systolic structure. Very recently, Montgomery systolic multipliers were presented where the register count was efficiently

Manuscript received March 31, 2016; revised June 22, 2016; accepted August 7, 2016. Date of publication September 8, 2016; date of current version January 19, 2017.

P. Chen, S. N. Basha, and J. Xie are with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435 USA (e-mail: chen.148@wright.edu; shaik.45@wright.edu; jiafeng.xie@wright.edu).

M. Mozaffari-Kermani is with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: m.mozaffari@rit.edu).

R. Azarderakhsh is with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA (e-mail: rxaeec@rit.edu).

Digital Object Identifier 10.1109/TVLSI.2016.2600568

reduced [10]. Several other works were reported for efficient realization of finite field Montgomery multiplication over $GF(2^m)$ [11], [17].

Based on the above discussion, in this paper, we introduce a novel strategy to design low register-complexity structures for multiplications over $GF(2^m)$. First, two low register-complexity AOP-based systolic multipliers are proposed. Then, the two designs are optimized as standard computation cores. After that, an efficient Montgomery multiplication algorithm (for trinomials) based on a novel precomputed-modular (PCM) operation for low register-complexity implementation is proposed. The proposed structures based on the proposed Montgomery algorithm can successfully employ the proposed AOP-based cores. Finally, FPGA implementation results are presented to confirm the efficiency of the proposed architectures.

The rest of this paper is organized as follows. The proposed AOP-based computation core is presented in Section II. The applications of the proposed AOP-based core, namely, the proposed Montgomery algorithm and systolic multipliers for trinomials, are described in Section III. In Section IV, we benchmark the hardware and time complexities of the proposed designs along with the corresponding existing works. Conclusions are given in Section V.

II. Low Register-Complexity AOP-Based Systolic Multipliers (AOP-Based Computation Core)

In this section, we briefly review the AOP-based multiplication algorithm first, and then present our proposed architectures based on the existing structures.

A. Review of AOP Multiplication Algorithm [24]

For simplicity of discussion, let $f(\alpha) = \alpha^k + \alpha^{k-1} + \cdots + \alpha + 1$ be an irreducible AOP of degree k over $GF(2)$ (where $k+1$ is prime and 2 is the primitive modulo $k+1$). For any $x \in GF(2)$, and x is a root of $f(\alpha) = 0$, we have

$$\begin{aligned} f(x) + xf(x) &= (x^k + x^{k-1} + \cdots + x + 1) \\ &\quad + x(x^k + x^{k-1} + \cdots + x + 1) \\ &= x^{k+1} + 1 = 0 \end{aligned} \tag{1}$$

and then we have

$$x^{k+1} = 1. \tag{2}$$

Then, let $\{x^{k+1}, x^k, \ldots, x, 1\}$ be the extended polynomial basis [27]. For any $A, B, C \in GF(2^m)$, these can be represented in the extended polynomial basis as

$$\begin{aligned} A &= \sum_{j=0}^{k} a_j x^j \\ B &= \sum_{j=0}^{k} b_j x^j \\ C &= \sum_{j=0}^{k} c_j x^j \end{aligned} \tag{3}$$

where $a_j, b_j, c_j \in GF(2)$, for $0 \le j \le k-1$, and $a_k = 0$, $b_k = 0$, and $c_k = 0$.

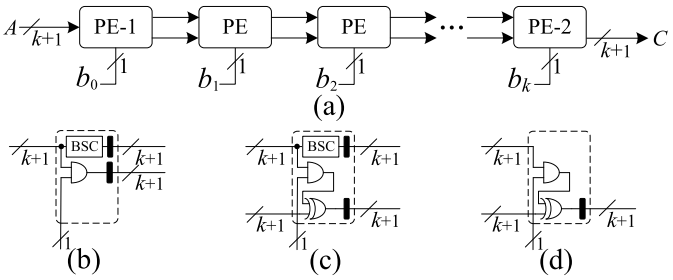


Fig. 1. Conventional systolic structure of AOP-based multiplication (structure-I: S-I), where BSC denotes the bit-shifting cell and the black box denotes the registers. (a) Structure. (b) Internal structure of PE-1. (c) Internal structure of regular PE. (d) Internal structure of PE-2.

Let us define C as the product of A and B, and then we have

$$C = A \cdot B \bmod f(x) \tag{4}$$

which can be written in this form

$$C = \sum_{j=0}^{k} b_j (x^j \cdot A \bmod f(x)). \tag{5}$$

Let us define $A^0 = A$ and $A^i = x^i \cdot A \bmod f(x)$, such that A^{i+1} can be obtained from A^i as

$$A^{i+1} = x \cdot A^i \bmod f(x). \tag{6}$$

Then, we have

$$A^{i+1} = \left(a_0^i x + a_1^i x^2 + \cdots + a_k^i \cdot x^{k+1}\right) \bmod f(x) \tag{7}$$

where

$$A^i = \sum_{j=0}^{k} a_j^i x^j. \tag{8}$$

Then, we have

$$A^{i+1} = a_0^{i+1} + a_1^{i+1} x + \cdots + a_k^{i+1} x^k \tag{9}$$

where

$$\begin{aligned} a_0^{i+1} &= a_k^i \\ a_j^{i+1} &= a_{j-1}^i, \quad \text{for } 1 \le j \le k-1. \end{aligned} \tag{10}$$

One can also extend to obtain A^{i+l} from A^i for $1 \le l \le k$, such that

$$\begin{aligned} a_j^{i+l} &= a_{k-l+j+1}^i \quad \text{for } 0 \le j \le l-1 \\ a_j^{i+l} &= a_{j-l}^i \qquad \text{otherwise.} \end{aligned} \tag{11}$$

B. Existing Systolic Structures

The conventional systolic structure based on the algorithm in Section II-A is shown in Fig. 1 (structure-I: S-I), where it consists of $(k+1)$ PEs (including three types of PEs: PE-1, PE-2, and regular PE). The internal structures of these PEs are shown in Fig. 1(b)–(d), respectively, where BSC denotes the bit-shifting cell. The latency of the structure in Fig. 1 is $(k+1)$ cycles, where the duration of each cycle period is $T_A + T_X$ (T_A and T_X refer to the delays of an AND gate and an XOR gate, respectively).

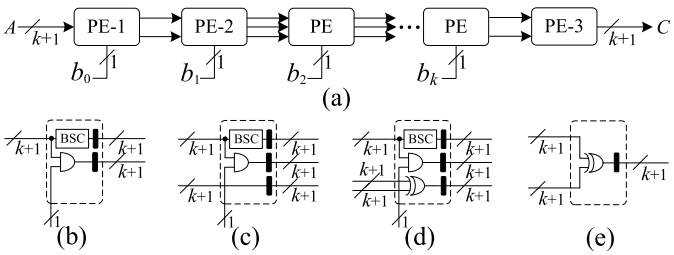


Fig. 2. Existing low critical-path structure of [24] for AOP-based multiplication (structure-II: S-II), where the black box denotes the registers. (a) Structure. (b) Internal structure of PE-1. (c) Internal structure of PE-2. (d) Internal structure of regular PE. (e) Internal structure of PE-3.

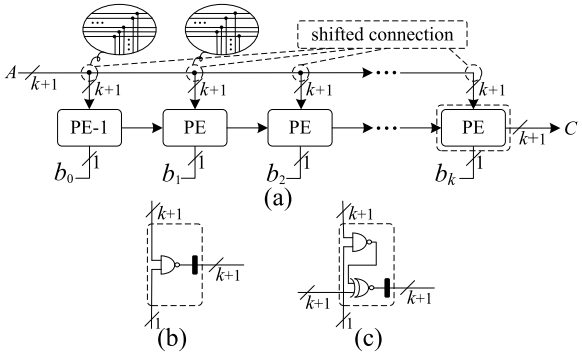


Fig. 3. Modified structure-I (MS-I), where the black box denotes the registers. For AOP implementation, we can remove the PE inside the dotted area since $b_k = 0$, but for the formation of standard computation core, this PE will be preserved. (a) MS-I. (b) Internal structure of PE-1. (c) Internal structure of regular PE.

A recent work has presented a low critical-path delay systolic structure (only T_X) [24], and it is shown in Fig. 2 (structure-II: S-II). The entire structure contains $(k + 2)$ PEs, where the internal structures of PEs are shown in Fig. 2(b)–(e), respectively. The latency of this structure is $(k + 2)$ cycles (critical-path delay: T_X).

C. Modified Low Register-Complexity Structures

For the structures of Figs. 1 and 2, we find that k^2 registers in the PEs pipeline identical data (in shifted order) to the neighboring PEs. These registers can be removed if we change the broadcasting strategy. As shown in Figs. 3 and 4, i.e., MS-I and MS-II, a shifted connection strategy is used in which the input A is directly fed to each PE and thus reduces the registers required. Moreover, the details of shifted connection are also shown in Figs. 3 and 4. To reduce the complexity further, we have used NAND and XNOR gates to replace the original AND and XOR gates, as depicted in [7] and [11] (the critical-path delay is then shortened to $T_{\rm NA} + T_{\rm XN}$, where $T_{\rm NA}$ and $T_{\rm XN}$ represent the delays of NAND and XNOR gates, respectively, as evidenced by the normalized area and delay comparison of various logic gates shown in Table I). It is noted that for AOP-based multiplication, the last PE (inside the dotted box) can be removed as $b_k = 0$. The modified structures involve nearly the same time complexity as the previous ones, but the register complexity is significantly reduced.

D. Low-Latency Implementations

For practical applications, we can further reduce the latencies of structures shown in Figs. 3 and 4, for $k+1 = pq + f$,

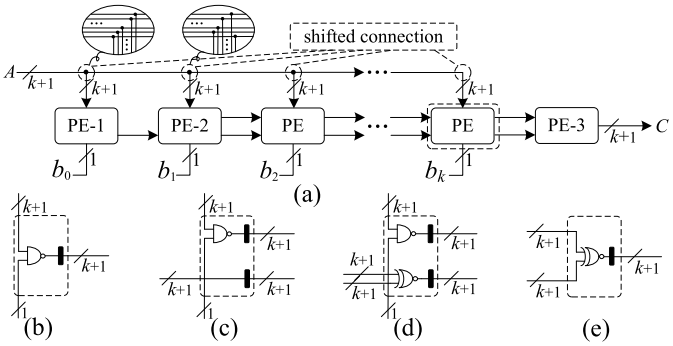


Fig. 4. Modified structure-II (MS-II), where the black box denotes the registers. For AOP implementation, we can remove the PE inside the dotted area since $b_k = 0$, but for the formation of standard computation core, this PE will be preserved. (a) MS-II. (b) Internal structure of PE-1. (c) Internal structure of PE-2. (d) Internal structure of regular PE. (e) Internal structure of PE-3.

TABLE I
NORMALIZED AREA AND DELAY COMPLEXITIES FOR VARIOUS LOGIC GATES BASED ON 45-nm TECHNOLOGY

Logic gate	Area	Delay
AND	1.064	0.02
NAND	0.792	0.01
XOR	1.596	0.04
XNOR	1.596	0.04

Unit for area: μm^2; Unit for delay: ns.
The library was generated using NanGate Library Creator based on the 45-nm FreePDK Base Kit of North Carolina State University [28] (Vdd = 1.10 V and Tj =25.0°C).

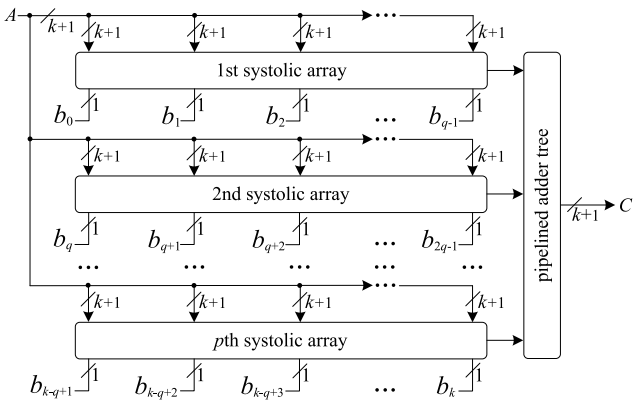


Fig. 5. Low-latency implementation of systolic structure, where the internal PEs can be those of MS-I or MS-II.

where $0 \le f \le q$. Without loss of generality, we assume $f = 0$, and then, we can decompose the original systolic array of $k+1$ PEs into p parallel arrays to achieve low-latency implementations, as shown in Fig. 5. An extra pipelined adder tree consisting of XNOR gates and registers is needed to add the results from p arrays together to yield the final result C.

E. Digit-Parallel Structures

We can combine neighboring PEs in a systolic array into one PE to reduce the register usage further. Fig. 6 shows an example of combining two neighboring PEs into one PE

TABLE II
COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS SYSTOLIC AOP-BASED MULTIPLIERS

Design	AND	NAND	XOR	XNOR	Register	Critical-path delay	Latency
Fig. 1 (S-I)	$(k+1)^2$	0	k^2+k	-	$2k^2+3k+1$	T_A+T_X	$k+1$
Fig. 2 (S-II)	$(k+1)^2$	0	k^2+k	-	$3k^2+6k+3$	T_X	$k+2$
Fig. 3 (MS-I)	–	k^2+k	–	k^2-1	k^2+2k	$T_{NA}+T_{XN}$	k
Fig. 4 (MS-II)	–	k^2+k	–	k^2-1	$2k^2+2k$	T_{XN}	$k+1$
Fig. 5[1]	–	k^2+k	–	$\simeq k^2-1$	$\simeq k^2+2k$	$T_{NA}+T_{XN}$	$q+\lceil\log_2 p\rceil$
Digit-parallel[2] ($d=2$)	–	k^2+k	–	$\simeq k^2-1$	$\simeq \lceil k^2/2\rceil+k$	$T_{NA}+T_{XN}$	$(q+\lceil\log_2 p)/2\rceil$

[1]: Based on the structure of Fig. 3 (MS-I).
[2]: Based on the low-latency structure of Fig. 5 (MS-I).

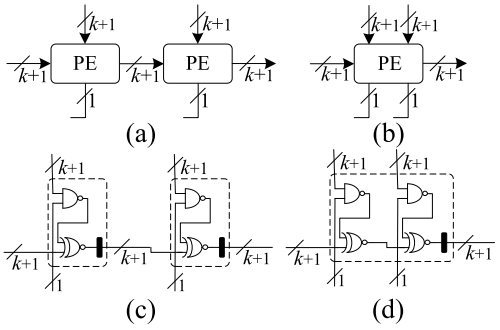


Fig. 6. PE design for digit-parallel implementation ($d=2$, based on the PEs from MS-I). (a) Original two neighboring PEs. (b) Combined PE. (c) Internal structure of previous two PEs. (d) Internal structure of combined PE.

(based on the PEs from MS-I). The critical-path delay of the new PE thus turns into ($T_{\rm NA}+2T_{\rm XN}$). For simplicity, we define the structure based on new PE in Fig. 6(b) as a digit-level parallel structure with digit size $d=2$. If we choose the value of d appropriately, the proposed architecture can achieve the optimal area-time complexity for specific applications.

F. Area-Time Complexities

The area-time complexities of the proposed designs in Figs. 3–6 are shown in Table II, along with the existing and conventional designs of Figs. 1 and 2. It can be seen that the proposed designs involve significantly less area-time complexity when compared with competing ones, especially in terms of the register complexity.

G. FPGA Implementation of Various AOP-Based Structures

We have also implemented these AOP-based systolic structures to confirm the efficacy of the proposed structures. We have synthesized these designs using Xilinx ISE 14.1 on Virtex 6 FPGA family with $k=162$. The results in terms of area-time-power complexity are shown in Table III.

It can be seen that the proposed structures outperform the existing ones, especially for area complexity. Since there is only a minor difference between the critical-path delays of $T_{\rm NA}+T_{\rm XN}$ and $T_{\rm XN}$ on FPGA platforms, the proposed MS-II does not have a significant advantage over existing ones. Therefore, the proposed MS-I can be used more widely than MS-II in practical applications.

H. AOP-Based Computation Core

To fully utilize the special property of the proposed AOP-based multipliers, we pack the structure of Fig. 5 (or combine with the structure of Fig. 6) as a standard computation core. The standard computation core is shown in Fig. 7, which consists of $k+1$ input bits from A, $k+1$ bits from input B, and $k+1$ bits of output C. For practical applications of this standard computation core, we can replace $k+1$ with any other integer. It is noted that both PEs of MS-I and MS-II can be used as internal structures for this computation core.

TABLE III
FPGA IMPLEMENTATION RESULTS OF VARIOUS AOP-BASED MULTIPLIERS FOR $k=162$

Design	Area	Delay[1]	Power	ADP[2]	PDP[3]
Fig. 1 (S-I)	53,300	154.035	2.236	8,210,066	344.42
Fig. 2 (S-II)	79,868	154.98	2.407	12,377,943	373.04
Fig. 3 (MS-I)	26,569	141.102	2.159	3,748,939	304.64
Fig. 4 (MS-II)	53,138	154.035	2.23	8,185,112	343.50

Unit for area: number of slice register; Unit for delay: ns; Unit for power: W (Power is estimated at 100MHz).
[1]: Delay = Latency.
[2]: ADP: Area-delay product = Area×Delay.
[3]: PDP: Power-delay product = Power×Delay.

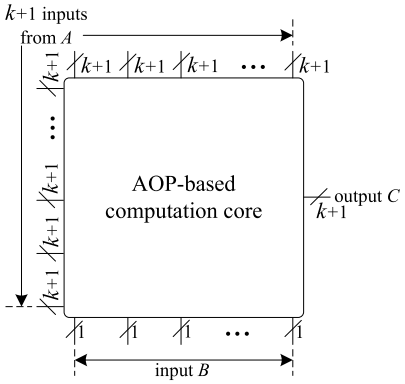


Fig. 7. AOP-based standard computation core, where the internal PEs can be those of MS-I or MS-II (the internal structure can be as that of Fig. 5, based on specific application environment).

III. APPLICATION OF THE PROPOSED AOP-BASED COMPUTATION CORE

In this section, we focus on the application of the AOP-based computation core to obtain a low register-complexity Montgomery multiplication based on trinomials.

A. Montgomery Multiplication Algorithm

Let $f(x)$ be a degree m irreducible trinomial over $GF(2)$ as

$$f(x) = x^m + x^n + 1 \tag{12}$$

where $1 \le n \le m-1$, such that we can have the Montgomery multiplication as [12]

$$C = A \cdot B\ r^{-1} \bmod\ f(x) \tag{13}$$

where A and B and their product C are elements in $GF(2^m)$ as $A = \sum_{i=0}^{m-1} a_i x^i$, $B = \sum_{i=0}^{m-1} b_i x^i$, and $C = \sum_{i=0}^{m-1} c_i x^i$, for $\{a_j, b_j, \text{and } c_j\} \in GF(2)$; r is the Montgomery factor that satisfies $gcd(r, f(x)) = 1$ (gcd refers to the greatest common divisor). Different algorithms have different selections of r to have the corresponding structures, as shown in [10]–[12].

In this algorithm, we have chosen $r = x^t = x^{(m-1)/2}$ (for the NIST recommended trinomials, m is an odd number). Then, (13) can be expressed as

$$\begin{aligned} C &= A \cdot B\ r^{-1} \bmod\ f(x) \\ &= \sum_{i=0}^{m-1} b_i (A \cdot x^i \cdot x^{-t}) \bmod\ f(x) = C_1 + C_2 \end{aligned} \tag{14}$$

where

$$\begin{aligned} C_1 &= \sum_{i=0}^{t-1} b_i \cdot A \cdot x^{i-t} \bmod\ f(x) \\ C_2 &= \sum_{i=t}^{m-1} b_i \cdot A \cdot x^{i-t} \bmod\ f(x). \end{aligned} \tag{15}$$

For C_1, we define $A_1^{(0)} = A$, $A_1^{(1)} = A \cdot x^{-1} \bmod\ f(x)$, …, $A_1^{(t)} = A \cdot x^{-t} \bmod\ f(x)$. Then, we have

$$A_1^{(i+1)} = A_1^{(i)} \cdot x^{-1} \bmod\ f(x) \tag{16}$$

where $0 \le i \le t-1$. C_1 can be expressed as

$$C_1 = \sum_{i=1}^{t} A_1^{(i)} b_{t-i}. \tag{17}$$

Let us define

$$A_1^{(i)} = a_{1,0}^{(i)} + a_{1,1}^{(i)} x + \cdots + a_{1,m-1}^{(i)} x^{m-1}. \tag{18}$$

Then, we have

$$\begin{aligned} A_1^{(i+1)} &= a_{1,0}^{(i)} x^{-1} + a_{1,1}^{(i)} + \cdots + a_{1,m-1}^{(i)} x^{m-2} \\ &= a_{1,0}^{(i+1)} + a_{1,1}^{(i+1)} x + \cdots + a_{1,m-1}^{(i+1)} x^{m-1}. \end{aligned} \tag{19}$$

Since x is the root of $f(x) = x^m + x^n + 1$, we can have $x^m + x^n = 1$ and $x^{m-1} + x^{n-1} = x^{-1}$. Substituting these into (19) yields

$$\begin{aligned} a_{1,m-1}^{(i+1)} &= a_{1,0}^{(i)} \\ a_{1,n-1}^{(i+1)} &= a_{1,n}^{(i)} \oplus a_{1,0}^{(i)} \\ a_{1,j}^{(i+1)} &= a_{1,j+1}^{i} \end{aligned} \tag{20}$$

for $0 \le j \le m-2$ and $j \ne n$.

Similarly, for C_2, we can define $A_2^{(0)} = A$, $A_2^{(1)} = A \cdot x \bmod\ f(x)$, …, $A_2^{(t)} = A \cdot x^t \bmod\ f(x)$ and $A_2^{(i+1)} = A_2^{(i)} \cdot x \bmod\ f(x)$ (where $0 \le i \le t-1$). With these definitions, C_2 can be expressed as

$$C_2 = \sum_{i=0}^{t} A_2^{(i)} b_{i+t}. \tag{21}$$

Let us define again $A_2^{(i)} = a_{2,0}^{(i)} + a_{2,1}^{(i)} x + \cdots + a_{2,m-1}^{(i)} x^{m-1}$. Similarly, we have

$$\begin{aligned} a_{2,0}^{(i+1)} &= a_{2,m-1}^{(i)} \\ a_{2,n}^{(i+1)} &= a_{2,n-1}^{(i)} \oplus a_{2,m-1}^{(i)} \\ a_{2,j}^{(i+1)} &= a_{2,j-1}^{(i)} \end{aligned} \tag{22}$$

for $1 \le j \le m-1$ and $j \ne n$.

B. Proposed Montgomery Multiplication Algorithm

Equations (12)–(22) represent the standard Montgomery multiplication process. To facilitate the Montgomery multiplication suitable for employing the proposed AOP-based computation core, we present the following proposed algorithm.

Let x^m be an extended polynomial basis. From (19), we define

$$A_U^{(1)} = \sum_{i=0}^{m} a_{U,i}^{(1)} x^i = a_{U,0}^{(1)} + a_{U,1}^{(1)} x + \cdots + a_{U,m}^{(1)} x^m \tag{23}$$

where

$$\begin{aligned} a_{U,0}^{(1)} + a_{U,1}^{(1)} x + \cdots + a_{U,m-1}^{(1)} x^{m-1} &= \sum_{i=0}^{m-1} a_i^{(0)} x^i \\ a_{U,m}^{(1)} = a_{1,n-1}^{(1)} &= a_n^{(0)} \oplus a_0^{(0)} \end{aligned} \tag{24}$$

such that $a_{U,i}^{(1)} x^i$ ($0 \le i \le m-1$) and $a_{U,i}^{(1)} x^i$ ($0 \le i \le n-1$, $n+1 \le i \le m$) can be selected to constitute $A_1^{(0)}$ and $A_1^{(1)}$, respectively, that is

$$\begin{aligned} \xi\left(A_U^{(1)}, 0\right) &= A_1^{(0)} \\ \xi\left(A_U^{(1)}, 1\right) &= A_1^{(1)} \end{aligned} \tag{25}$$

where $\xi(\cdot)$ represents the bit-selection operation.

We can similarly extend (23) to $A_U^{(2)} = \sum_{i=0}^{m+1} a_{U,i}^{(2)} x^i = a_{U,0}^{(2)} + \cdots + a_{U,m+1}^{(2)} x^{m+1}$, where x^{m+1} is an extended polynomial basis and

$$\begin{aligned} a_{U,0}^{(2)} + a_{U,1}^{(2)} x + \cdots + a_{U,m-1}^{(2)} x^{m-1} &= \sum_{i=0}^{m-1} a_i^{(0)} x^i \\ a_{U,m}^{(2)} = a_{1,n-1}^{(1)} &= a_n^{(0)} \oplus a_0^{(0)} \\ a_{U,m+1}^{(2)} = a_{1,n-1}^{(2)} = a_n^{(1)} \oplus a_0^{(1)} &= a_{n+1}^{(0)} \oplus a_1^{(0)}. \end{aligned} \tag{26}$$

Thus, $a_{U,i}^{(2)} x^i$ ($0 \le i \le m-1$), $a_{U,i}^{(2)} x^i$ ($0 \le i \le n-1$, $n+1 \le i \le m$), and $a_{U,i}^{(2)} x^i$ ($0 \le i \le n-2$, $n+1 \le i \le m+1$), respectively, can be chosen to construct $A_1^{(0)}$, $A_1^{(1)}$, and $A_1^{(2)}$, that is

$$\begin{aligned} \xi\left(A_U^{(2)}, 0\right) &= A_1^{(0)} \\ \xi\left(A_U^{(2)}, 1\right) &= A_1^{(1)} \\ \xi\left(A_U^{(2)}, 2\right) &= A_1^{(2)}. \end{aligned} \tag{27}$$

In conclusion, we can have

$$A_U^{(t)} = \sum_{i=0}^{m+t-1} a_{U,i}^{(t)} x^i = a_{U,0}^{(t)} + \cdots + a_{U,m+t-1}^{(t)} x^{m+t-1} \quad (28)$$

where $x^{m+1}, \ldots, x^{m+t-1}$ are defined as extended polynomial basis and (applicable to two trinomials recommended by NIST, where $m - n > t$)

$$\begin{aligned} &a_{U,0}^{(t)} + a_{U,1}^{(t)} x + \cdots + a_{U,m-1}^{(t)} x^{m-1} = \sum_{i=0}^{m-1} a_{1,i}^{(0)} x^i \\ &a_{U,m+j-1}^{(t)} = a_{1,n+j}^{(0)} \oplus a_{1,j-1}^{(0)} \quad (1 \le j \le n+1) \\ &\cdots \quad \cdots \quad \cdots \\ &a_{U,m+t-n-1+j}^{(t)} = a_{1,n+j}^{(0)} \oplus a_{1,j}^{(0)} \oplus a_{1,2n+j+1}^{(0)} \\ &\qquad (1 \le j \le t-n-1) \end{aligned} \quad (29)$$

where $a_{U,i}^{(t)} x^i$ $(0 \le i \le m-1)$, $a_{U,i}^{(t)} x^i$ $(0 \le i \le n-1, n+1 \le i \le m)$, …, $a_{U,i}^{(t)} x^i$ $(0 \le i \le n-t, n+1 \le i \le m+t-1)$ can be chosen, respectively, to construct $A_1^{(0)}$, $A_1^{(1)}$, …, and $A_1^{(t)}$, that is

$$\begin{aligned} &\xi(A_U^{(t)}, 0) = A_1^{(0)} \\ &\xi(A_U^{(t)}, 1) = A_1^{(1)} \\ &\cdots \quad \cdots \quad \cdots \\ &\xi(A_U^{(t)}, t) = A_1^{(t)}. \end{aligned} \quad (30)$$

Similarly, for C_2, we have

$$A_V^{(t)} = \sum_{i=0}^{m+t-1} a_{V,i}^{(t)} x^i = a_{V,0}^{(t)} + \cdots + a_{V,m+t-1}^{(t)} x^{m+t-1} \quad (31)$$

and (applicable to two trinomials recommended by NIST, where $m - n > t$)

$$\begin{aligned} &a_{V,0}^{(t)} + a_{V,1}^{(t)} x + \cdots + a_{V,m-1}^{(t)} x^{m-1} = \sum_{i=0}^{m-1} a_{2,i}^{(0)} x^i \\ &a_{V,m+j-1}^{(t)} = a_{2,n-j}^{(0)} \oplus a_{2,m-j}^{(0)} \ (1 \le j \le t). \end{aligned} \quad (32)$$

Similar to (27), we can have

$$\begin{aligned} &\xi(A_V^{(t)}, 0) = A_2^{(0)} \\ &\xi(A_V^{(t)}, 1) = A_2^{(1)} \\ &\cdots \quad \cdots \quad \cdots \\ &\xi(A_V^{(t)}, t) = A_2^{(t)}. \end{aligned} \quad (33)$$

Based on the above, (14) can be rewritten as

$$C = C_1 + C_2 = Ab_t + \sum_{i=1}^{t} \left(\xi\left(A_U^{(t)}, i\right) b_{t-i} + \xi\left(A_V^{(t)}, i\right) b_{i+t}\right). \quad (34)$$

Algorithm 1 Proposed Montgomery Multiplication

Inputs: A and B are the pair of polynomials in $GF(2^m)$ to be multiplied.
Output: $C = A \cdot B\ r^{-1} \bmod f(x)$.
1. Initialization step
   1.1 $r^{-1} = x^{-t}$.
   1.2 $D = 0$, $E = 0$.
2. Multiplication step
   2.1-a. $A_U^{(t)} = \mathrm{PCM}(A, U)$.
   2.1-b. $A_V^{(t)} = \mathrm{PCM}(A, V)$.
   2.1-c. $D = Ab_t$.
   2.2. $E = D + \sum_{i=1}^{t} (\xi(A_U^{(t)}, i) b_{t-i} + \xi(A_V^{(t)}, i) b_{i+t})$.
3. Final step
   3.1. $C = E$.

From (28) and (29) and (32) and (33), we can derive $A_U^{(t)}$ and $A_V^{(t)}$ directly from operand A through XOR operations, and thus, we define this operation as PCM operation as

$$\begin{aligned} A_U^{(t)} &= \mathrm{PCM}(A, U) \\ A_V^{(t)} &= \mathrm{PCM}(A, V). \end{aligned} \quad (35)$$

Based on (23)–(35), the proposed Montgomery multiplication algorithm for employing the AOP-based computation core is thus given in Algorithm 1.

In Algorithm 1, Step 2.2 refers to the bit-parallel multiplication process. According to the proposed algorithm, we generate operands at the first cycle period, and then, they are distributed into t partial products to be accumulated in a systolic way, which greatly facilitates employing the proposed AOP-based computation core, since all involved bits are already generated from PCM (the details can be seen in Section III-C).

C. Proposed Low Register-Complexity Systolic Structure Employing the AOP-Based Computation Core

The proposed structure based on the proposed Algorithm 1 (employing the proposed AOP-based computation core) is shown in Fig. 8(a). It contains one AOP-based computation core and three extra PEs. PE-0 yields two outputs (each output with $m + t - 1$ bits) to the computation core to be selectively connected with $m - 1$ input ports. As shown in Fig. 8(c), PE-0 performs the PCM of operand A and yields two outputs ($A_U^{(t)}$ and $A_V^{(t)}$) to the computation core, respectively (m bits of operand A are shared). The PCM of (32) only takes one T_X delay, while the PCM of (29) takes $2T_X$. To lower the critical-path delay, we have used two-stage XOR operations to minimize the critical-path delay to one XOR delay [stage-I uses the least number of XOR gates required by (29), while stage-II realizes the rest of operations of (29) and (32)], as shown by an example design in Fig. 9. PE-1 calculates the multiplication of operand A and b_t according to Algorithm 1, while PE-2 functions as the final addition to produce the output C.

The internal structure of the AOP-based computation core is shown in Fig. 8(b), where we have used PEs from MS-I as internal PEs for $e = 2$ (one can extend the structure to

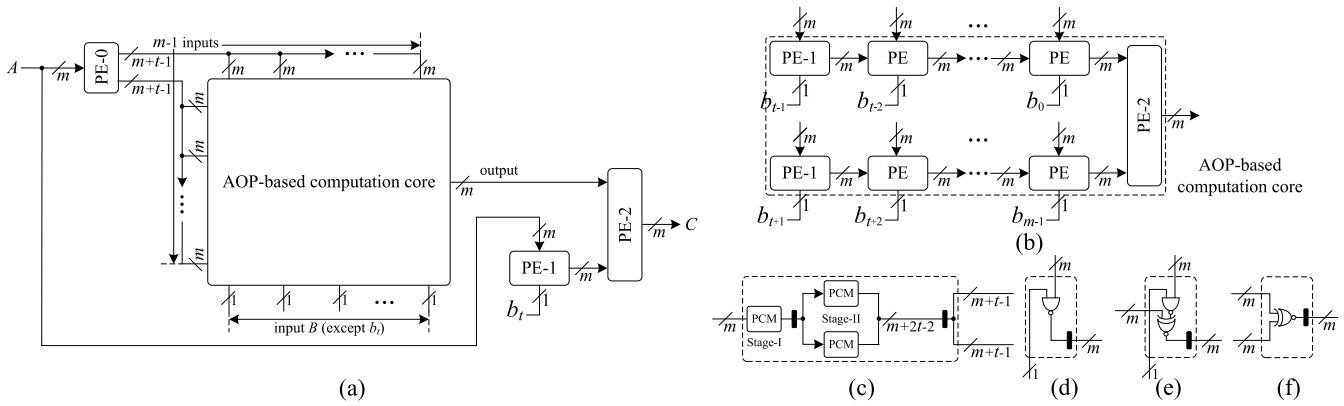


Fig. 8. Proposed low register-complexity systolic multiplier based on the AOP-based computation core (MS-I), where the black box denotes the registers. (a) Proposed structure. (b) Internal structure of the AOP-based computation core (MS-I, where $e = 2$). (c) Detailed design of PE-0. (d) Detailed design of PE-1. (e) Detailed design of regular PE. (f) Detailed design of PE-2.

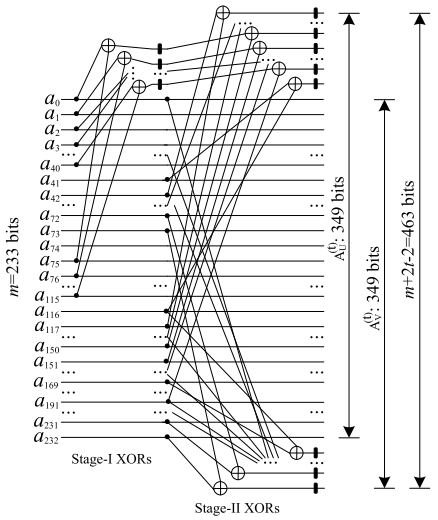


Fig. 9. Detailed design of two-stage XOR operations in PE-0 for trinomial $f(x) = x^{233} + x^{74} + 1$, where the black box denotes bit register.

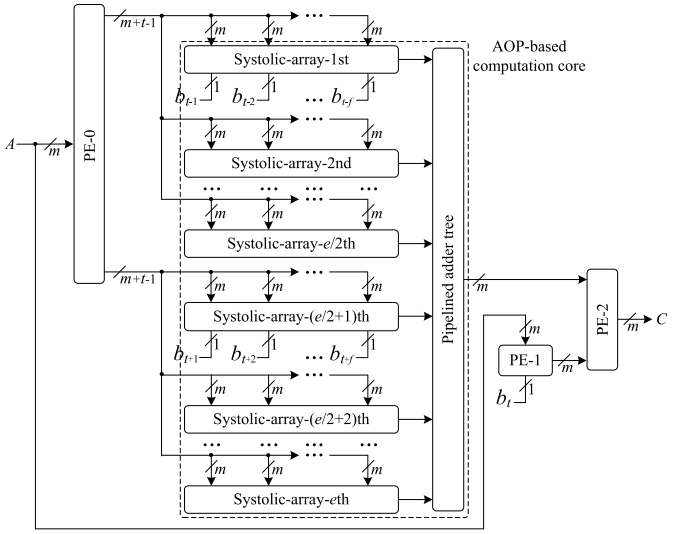


Fig. 10. Proposed low-latency systolic multiplier.

any value of e). The computation core contains $(2t + 1)$ PEs, where the detailed designs of PEs are shown in Fig. 8(d)–(f), respectively. PE-1 performs multiplication between one m-bit operand and one bit of operand B and then yields the result to their right. The regular PE performs multiplication between selected operand and one bit of operand B. The result of multiplication is added with the input from previous PE and then produces the result to the PE on its right. The last PE, PE-2, performs the addition of two systolic arrays and yields the final result.

The critical-path delay of the proposed multiplier of Fig. 8 is $(T_{\mathrm{NA}} + T_{\mathrm{XN}})$ (if we choose the PEs from MS-II, the critical-path delay will be T_{XN}). The proposed design gives the first output of desired product $(t + 3)$ cycles after the pair of operands are fed to the structure, while the successive outputs are produced in every cycle thereafter.

D. Low-Latency Structure

Let $2t = eh + l$, where $0 \le l \le h$. For simplicity, we can assume $l = 0$; however, it can be extended to $l \neq 0$. Then, we can rewrite (34) as

$$\begin{aligned} C = {} & Ab_t + \sum_{i=1}^{h} \left(\xi\left(A_U^{(t)}, i\right) b_{t-i} + \xi\left(A_V^{(t)}, i\right) b_{i+t}\right) \\ & + \sum_{i=h+1}^{2h} \left(\xi\left(A_U^{(t)}, i\right) b_{t-i} + \xi\left(A_V^{(t)}, i\right) b_{i+t}\right) \\ & + \cdots + \cdots + \cdots \\ & + \sum_{i=t-h+1}^{t} \left(\xi\left(A_U^{(t)}, i\right) b_{t-i} + \xi\left(A_V^{(t)}, i\right) b_{i+t}\right). \end{aligned} \tag{36}$$

where the original two systolic arrays in the computation core of Fig. 8 can be divided into e arrays (each array has h PEs), as shown in Fig. 10. The latency of the structure in Fig. 10 is only $(h + 3 + \lceil \log_2 e \rceil)$ cycles (PE-0 and PE-1 take two cycles to be processed in parallel, while the adder tree and PE-2 require $\lceil \log_2 e \rceil$ and one cycle, respectively), which is significantly shorter than the previous one in Fig. 1. A pipelined adder tree is used to add together the results of e systolic arrays of the computation core.

TABLE IV
COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS SYSTOLIC MULTIPLIERS BASED ON TRINOMIALS

Design	AND	NAND	XOR	XNOR	Register	Latency	Critical-path delay
Bit-parallel systolic structures							
[7]	m^2	-	m^2+m-1	-	$4m^2+2m-2$	$2m-1$	T_A+T_X
[8][1]	-	m^2	m^2-1	-	$2m^2-2m$	m	$T_{NA}+T_X$
[8][2]	-	m^2	$m^2+m-2\sqrt{\lceil m \rceil}$	-	$2m^2+m\sqrt{\lceil m \rceil}-2m$	$2\sqrt{\lceil m \rceil}$	$T_{NA}+T_X$
Fig. 2 [10]	-	m^2	$< 1.5m^2+0.5m+1$	-	$1.5m^2+0.5m$	$m+1$	$2T_X$
Fig. 3 [10]	-	m^2	$< 1.5m^2+0.5m+1$	0	$1.5m^2+2m$	$m+2$	$T_{NA}+T_X$
Fig. 8[3]*	-	m^2	-	m^2-1	m^2+3m-1	$\lceil (m+7)/2 \rceil$	$T_{NA}+T_{XN}$
Fig. 8[4]*	-	m^2	-	m^2-1	$2m^2+m$	$\lceil (m+9)/2 \rceil$	T_{XN}
Fig. 10[5]*	-	m^2	-	m^2-1	$\simeq m^2+3m-1$	$\lceil (m-1)/e \rceil+3+\lceil \log_2 e \rceil$	$T_{NA}+T_{XN}$
Digit-parallel systolic structures ($d=2$)							
[8]	-	m^2	m^2+m	-	m^2	$m/2$	$T_{NA}+2T_X$
[9][6]	-	m^2	$\simeq m^2+m$	-	$\simeq 1.5m^2+2m$	$< 2\sqrt{\lceil m \rceil}$	$T_{NA}+T_X$
Fig. 8[7]*	-	m^2	-	m^2-1	$\lceil m^2/2 \rceil+(m-1)/2$	$\lceil (m+11)/4 \rceil$	$T_{NA}+2T_{XN}$
Fig. 8[8]*	-	m^2	-	m^2-1	$m^2-m+(m-1)/2$	$\lceil (m+11)/4 \rceil$	$2T_{XN}$
Fig. 10[9]*	-	m^2	-	m^2-1	$\simeq \lceil m^2/2 \rceil+1.5m$	$\lceil (m-1)/(2e) \rceil$ $+2+\lceil \log_2(e/2) \rceil$	$T_{NA}+2T_{XN}$

*: The XOR gates in the PE-0 have been counted as XNOR gates.
[1]: The regular systolic structure.
[2]: The super-systolic structure.
[3]: The structure with $e=2$ and $d=1$ (PEs of the computation core are from MS-I).
[4]: The structure with $e=2$ and $d=1$ (PEs of the computation core are from MS-II).
[5]: The structure with $d=1$ (MS-I) (PEs of the computation core are from MS-I).
[6]: The structure here can be seen as digit-level structure of $d=2$.
[7]: The structure with $e=2$ and $d=2$ (PEs of the computation core are from MS-I).
[8]: The structure with $e=2$ and $d=2$ (PEs of the computation core are from MS-Ii).
[9]: The structure with $d=2$ (MS-I) (PEs of the computation core are from MS-I).

E. Digit-Parallel Structure

We can also employ the PEs from Fig. 6 to have digit-parallel structure to reduce the register complexity further. It is noted that the digit-parallel structure can be combined with the low latency one to achieve optimal implementation.

IV. AREA AND TIME COMPLEXITIES

In this section, we benchmark the hardware and time complexities of the proposed architectures.

A. Comparison

The area and time complexities in terms of logic gate count, register count, latency, and critical-path delay of the proposed and existing structures of [7]–[10] are listed in Table IV.

The proposed architectures outperform the existing ones, especially in the register count. The proposed architectures have lower area-time complexity than the design of [7]. When compared with the low-latency supersystolic structure of [8], the proposed architecture (Fig. 10) has shorter latency (if we choose $e=\sqrt{\lceil m \rceil}$) and less registers. Furthermore, when compared with the two architectures in [9] and [10], the proposed architectures not only have lower register count, but also constitute significantly lower latency. Among all the existing architectures, only the work of [8] and [9] has proposed the similar digit-parallel structures, as shown in Fig. 6. From Table IV, it is shown that the proposed digit-parallel structures have less register count and shorter latency than those of [8] and [9].

TABLE V
COMPARISON OF REGISTER COUNT AND LATENCY OF VARIOUS BIT-PARALLEL MULTIPLIERS

Architecture	Register[1]	Latency[1]	Register[2]	Latency[2]
[7]	217,620	465	669,940	817
[8][3]	111,840	32	342,333	42
Fig. 2 [10]	81,551	234	251,127	410
Fig. 3 [10]	81,900	235	251,740	411
Fig. 8 (MS-I)	54,987	120	168,507	208
Fig. 10 (MS-I)[4]	54,987	22	168,507	33

[1]: For trinomial $f(x)=x^{233}+x^{74}+1$.
[2]: For trinomial $f(x)=x^{409}+x^{87}+1$.
[3]: The super-systolic structure.
[4]: Proposed structure with $e=16$.

For a fair benchmark, we have also given the comparison of register count and latency of various architectures based on trinomials $f(x)=x^{233}+x^{74}+1$ and $f(x)=x^{409}+x^{87}+1$, as shown in Table V. It can be seen that the proposed architecture, especially Fig. 8 (MS-I), has the highest efficiencies in terms of both register count and latency.

B. FPGA Implementations

We have implemented the proposed architectures, including the structures of Fig. 8 ($e=2$) and Fig. 10 ($e=16$ and $d=2$),

TABLE VI
COMPARISON OF AREA-TIME COMPLEXITIES OF VARIOUS MULTIPLIERS BASED ON TRINOMIALS

Architecture	Area	Delay[1]	Power	ADP[2]	PDP[3]
Bit-parallel systolic structures ($f(x) = x^{233} + x^{74} + 1$)					
Fig. 3 ([10])	81,805	222.1	2.515	18,168,891	558.58
Fig. 8[4]	54,032	128.2	2.336	6,926,902	299.48
Fig. 10[5]	56,400	37.3	2.351	2,103,156	87.67
Bit-parallel systolic structures ($f(x) = x^{409} + x^{87} + 1$)					
Fig. 3 ([10])	251,740	436.1	3.970	109,783,814	1,731.3
Fig. 8[4]	168,505	220.7	3.433	37,189,054	757.66
Fig. 10[5]	177,503	55.9	3.472	9,922,418	194.08
Digit-parallel systolic structures ($d = 2$) ($f(x) = x^{233} + x^{74} + 1$)					
[9][6]	81,911	35.6	2.516	2,916,032	89.57
Fig. 10[7]	22,160	22.0	2.130	488,406	46.95
Digit-parallel systolic structures ($d = 2$) ($f(x) = x^{409} + x^{87} + 1$)					
[9][6]	251,740	57.6	3.970	14,500,224	228.67
Fig. 10[7]	94,067	30.5	2.952	2,869,044	90.04

Unit for Area: number of slice register; Unit for delay: ns; Unit for power: W (power is estimated at 100MHz).
[1]: Delay = Latency.
[2]: ADP: Area-delay product = Area×Delay.
[3]: PDP: Power-delay product = Power×Delay.
[4]: Based on the structure of MS-I with $e = 2$.
[5]: Based on the structure of MS-I with $e = 16$ and $d = 1$.
[6]: The structure here has 16 parallel systolic arrays ($d = 2$).
[7]: Based on the structure of MS-I with $e = 16$ and $d = 2$.

using Xilinx ISE 14.1 on the Virtex 6 FPGA family based on the trinomials $f(x) = x^{233} + x^{74} + 1$ and $f(x) = x^{409} + x^{87} + 1$. The area-time-power complexities of the best existing designs [9], [10, Fig. 3] are also obtained. The area-time-power complexities of all these designs are shown in Table VI.

As shown in Table VI, the proposed structures significantly outperform the existing designs. The proposed structures are found to have at least 61.8% and 47.6% less area-delay product (ADP) and power-delay product (PDP) than the state-of-the-art previous architectures, respectively. It is also noted that as field-size increases from 233 to 409, the proposed structures are found to be more efficient in area-time-power complexities, e.g., the proposed designs have at least 61.8% and 47.6% less ADP and PDP than the existing architectures at $GF(2^{233})$, while the proposed ones have at least 66.2% and 56.2% less ADP and PDP than the competing ones at $GF(2^{409})$.

C. Discussion

It is noted that from Table VI, the proposed architecture of Fig. 10 ($e = 16$ and $d = 2$) achieves the best area-time complexity among all the designs. The reduction of registers brought by digit-parallel implementation is significant. For practical applications, one can choose suitable values of d (coordinating with the selection of e) to obtain optimal realizations based on the usage models and performance and implementation objectives.

It is worth mentioning that after packing as a computation core, the AOP-based multipliers can be used as a regular component in practical cryptosystems usage though AOP-based designs are usually not preferable in such systems due to security issues [5]. In the future, we plan to extend the AOP-based cores to pentanomial-based cryptosystems.

V. CONCLUSION

An efficient, new scheme for low-complexity implementation of finite field multipliers over $GF(2^m)$ based on trinomials benchmarked on the FPGA platform has been proposed. We have proposed a modified data broadcasting technique to reduce the register complexity within the existing AOP-based multipliers. Then, the AOP-based multipliers have been packed as standard computation cores to be used for trinomial-based multipliers. Moreover, a novel low register-complexity Montgomery multiplication algorithm for systolic trinomial-based finite field multipliers is presented. The systolic multiplier based on the proposed algorithm can employ the AOP-based computation core to offer low register-complexity implementations. We have also introduced structures for low-latency and digit-parallel implementations. Both the theoretical analysis and the FPGA implementation results have confirmed the higher efficiency of the proposed architectures compared with the competing ones.

REFERENCES

[1] I. F. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography* (London Mathematical Society Lecture Note Series). Cambridge, U.K.: Cambridge Univ. Press, 1999.

[2] N. R. Murthy and M. N. S. Swamy, "Cryptographic applications of Brahmagupta–Bhăskara equation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 7, pp. 1565–1571, Jul. 2006.

[3] M. Sun *et al.*, "eButton: A wearable computer for health monitoring and personal assistance," in *Proc. 51st Design Autom. Conf.*, 2014, pp. 1–6.

[4] D. Schinianakis and T. Stouraitis, "Multifunction residue architectures for cryptography," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 4, pp. 1156–1169, Apr. 2014.

[5] *FIPS 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2*, Nat. Inst. Standards Technol., Gaithersburg, MD, USA, 2000.

[6] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York, NY, USA: Wiley, 1999.

[7] C.-Y. Lee, J.-S. Horng, I.-C. Jou, and E.-H. Lu, "Low-complexity bit-parallel systolic Montgomery multipliers for special classes of $GF(2^m)$," *IEEE Trans. Comput.*, vol. 54, no. 9, pp. 1061–1070, Sep. 2005.

[8] P. K. Meher, "Systolic and super-systolic multipliers for finite field $GF(2^m)$ based on irreducible trinomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 4, pp. 1031–1040, May 2008.

[9] J. Xie, P. K. Meher, and J. He, "Low-latency area-delay-efficient systolic multiplier over $GF(2^m)$ for a wider class of trinomials using parallel register sharing," in *Proc. IEEE Int. Sym. Circuits Syst.*, May 2012, pp. 89–92.

[10] S. Bayat-Sarmadi and M. Farmani, "High-throughput low-complexity systolic Montgomery multiplication over $GF(2^m)$ based on trinomials," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 4, pp. 377–381, Apr. 2015.

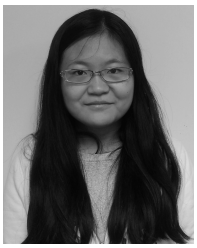
[11] J. Xie, J. J. He, and P. K. Meher, "Low latency systolic Montgomery multiplier for finite field $GF(2^m)$ based on pentanomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 2, pp. 385–389, Feb. 2013.

[12] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, 1985.

[13] R. Azarderakhsh, K. Järvinen, and V. Dimitrov, "Fast inversion in $GF(2^m)$ with normal basis using hybrid-double multipliers," *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 1041–1047, Apr. 2014.

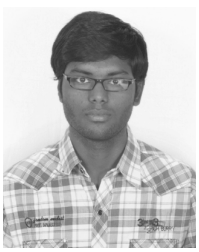
[14] R. Azarderakhsh, D. Jao, and H. Lee, "Common subexpression algorithms for space-complexity reduction of Gaussian normal basis multiplication," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2357–2369, May 2015.

[15] R. Azarderakhsh and M. Mozaffari-Kermani, "High-performance two-dimensional finite field multiplication and exponentiation for cryptographic applications," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1569–1576, Oct. 2015.
[16] C.-Y. Lee and P. K. Meher, "Area-efficient subquadratic space-complexity digit-serial multiplier for type-II optimal normal basis of $GF(2^m)$ using symmetric TMVP and block recombination techniques," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 12, pp. 2846–2855, Dec. 2015.
[17] S. Talapatra, H. Rahaman, and S. K. Saha, "Unified digit serial systolic Montgomery multiplication architecture for special classes of polynomials over $GF(2^m)$," in *Proc. 13th Euromicro Conf. Digit. Syst. Design, Archit., Methods Tools*, Sep. 2010, pp. 427–432.
[18] S. T. J. Fenn and M. G. Parker, "Bit-serial multiplication in $GF(2^m)$ using irreducible all-one polynomials," *IEE Proc.-Comput. Digit. Tech.*, vol. 144, no. 6, pp. 391–393, Nov. 1997.
[19] K.-Y. Chang, D. Hong, and H.-S. Cho, "Low complexity bit-parallel multiplier for $GF(2^m)$ defined by all-one polynomials using redundant representation," *IEEE Trans. Comput.*, vol. 54, no. 12, pp. 1628–1630, Dec. 2005.
[20] H.-S. Kim and S.-W. Lee, "LFSR multipliers over $GF(2^m)$ defined by all-one polynomial," *Integr., VLSI J.*, vol. 40, no. 4, pp. 473–478, 2007.
[21] P. K. Meher, Y. Ha, and C.-Y. Lee, "An optimized design for serial-parallel finite field multiplication over GF(2^m) based on all-one polynomials," in *Proc. ASP-DAC*, Jan. 2009, pp. 210–215.
[22] M. Morales-Sandoval, C. Feregrino-Uribe, and C. Kitsos, "Bit-serial and digit-serial $GF(2^m)$ Montgomery multipliers using linear feedback shift registers," *IET Comput. Digit. Tech.*, vol. 5, no. 2, pp. 86–94, Mar. 2011.
[23] C.-Y. Lee, E.-H. Lu, and J.-Y. Lee, "Bit-parallel systolic multipliers for $GF(2^m)$ fields defined by all-one and equally spaced polynomials," *IEEE Trans. Comput.*, vol. 50, no. 5, pp. 385–393, May 2001.
[24] J. Xie, P. K. Meher, and J. He, "Low-complexity multiplier for $GF(2^m)$ based on all-one polynomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 168–173, Jan. 2013.
[25] Y.-R. Ting, E.-H. Lu, and Y.-C. Lu, "Ringed bit-parallel systolic multipliers over a class of fields $GF(2^m)$," *Integr., VLSI J.*, vol. 38, no. 4, pp. 571–578, 2005.
[26] S. Talapatra, H. Rahaman, and J. Mathew, "Low complexity digit serial systolic Montgomery multipliers for special class of $GF(2^m)$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 5, pp. 847–852, May 2010.
[27] T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of fields $GF(2^m)$," *Inf. Comput.*, vol. 83, no. 1, pp. 21–40, 1989.
[28] *NanGate Standard Cell Library*. [Online]. Available: http://www.si2.org/



Pingxiuqi Chen received the B.S. degree in physics from Hainan Normal University, Haikou, China, in 2014. She is currently pursuing the Ph.D. Degree with the Department of Electrical Engineering, Wright State University, Dayton, OH, USA.

Her current research interests include VLSI cryptographic circuits design and finite field arithmetic design.



Shaik Nazeem Basha received the B.Tech. degree in electronics and communications engineering from the Rajiv Gandhi University of Knowledge Technologies, Hyderabad, India in 2014. He is currently pursuing the Master Degree with the Department of Electrical Engineering, Wright State University, Dayton, OH, USA.

His current research interests include VLSI cryptographic circuits design and VLSI signal processing systems.

Mr. Basha is a recipient of the Merit Scholarship of Wright State University from 2015 to 2017.

Mehran Mozaffari-Kermani (M'11) received the B.Sc. degree in electrical and computer engineering from the University of Tehran, Tehran, Iran, in 2005, and the M.E.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Western Ontario, London, ON, Canada, in 2007 and 2011, respectively.

He joined the Advanced Micro Devices, Sunnyvale, CA, USA, as a Senior ASIC/Layout Designer, where he was involved in integrating sophisticated security/cryptographic capabilities into accelerated processing. In 2012, he joined the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, as an NSERC Post-Doctoral Research Fellow. He is currently with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA.

Dr. Kermani was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada Post-Doctoral Research Fellowship in 2011, and the Texas Instruments Faculty Award (Douglas Harvey) in 2014. He is currently an Associate Editor of the IEEE TRANSACTIONS ON VLSI SYSTEMS, the *ACM Transactions on Embedded Computing Systems*, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I, and the Guest Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations in 2016 and 2017. He was the Lead Guest Editor of the IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING for special issues on security.

Reza Azarderakhsh (M'11) received the B.Sc. degree in electrical and electronic engineering and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2002 and 2005, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2011.

He joined the Department of Electrical and Computer Engineering, University of Western Ontario as a Limited Duties Instructor, in 2011. He has been an NSERC Post-Doctoral Research Fellow with the Center for Applied Cryptographic Research and the Department of Combinatorics and Optimization, University of Waterloo, Waterloo, ON, Canada. His current current research interests include finite field and its application, elliptic curve cryptography, and pairing based cryptography.

Dr. Azarderakhsh was a recipient of the prestigious Natural Sciences and Engineering Research Council of Canada (NSERC) Post-Doctoral Research Fellowship in 2012. He is currently an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I. He is the Guest Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING for the special issue of Emerging Embedded and Cyber Physical System Security Challenges and Innovations in 2016 and 2017. He is currently the Guest Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS for the special issue of Emerging Security Trends for Biomedical Computations, Devices, and Infrastructures in 2015 and 2016. He is currently with the Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY, USA.

Jiafeng Xie (M'15) received the B.E. degree in measurement and control technology and instrumentation from Yanshan University, Qinhuangdao, China, in 2006, the M.E. degree in control science and engineering from the Central South University, Changsha, China, in 2010, and the Ph.D. degree in electrical engineering from the University of Pittsburgh, Pittsburgh, PA, USA, in 2014.

He is currently an Assistant Professor with Department of Electrical Engineering, Wright State University, Dayton, OH, USA. His current research interests include VLSI cryptographic circuits design, intelligent system fault detection, hardware security, and VLSI signal/image processing systems.