[10] B. Calhoun and A. Chandrakasan, "Static noise margin variation for sub-threshold SRAM in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 41, no. 7, pp. 1673–1679, Jul. 2006.
[11] T. C. Hesterberg, "Advances in importance sampling," Ph.D. dissertation, Dept. Statistics, Stanford Univ., Stanford, CA, 1988.
[12] P. Smith, M. Shafi, and H. Gao, "Quick simulation: A review of importance sampling techniques in communications systems," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 4, pp. 597–613, May 1997.
[13] J. Wang, S. Yaldiz, X. Li, and L. Pileggi, "SRAM parametric failure analysis," in *Proc. DAC*, 2009, pp. 496–501.

Digit-Level Semi-Systolic and Systolic Structures for the Shifted Polynomial Basis Multiplication Over Binary Extension Fields

Arash Hariri and Arash Reyhani-Masoleh

***Abstract*—Finite field multiplication is one of the most important operations in the finite field arithmetic. In this paper, we study semi-systolic and systolic implementations of the shifted polynomial basis multiplication and propose low time complexity semi-systolic and systolic array structures. We show that our proposed semi-systolic multiplier is faster than its existing counterparts available in the literature. Our application-specified integrated circuit (ASIC) implementation of the proposed semi-systolic multiplier demonstrates that reduction in time complexity is achieved without imposing hardware overhead. Furthermore, our proposed systolic array shifted polynomial basis (SPB) multiplier has a low time complexity for general irreducible polynomials.**

***Index Terms*—Binary extension fields, digit-level, multiplication, semi-systolic, shifted polynomial basis, systolic.**

I. Introduction

Cryptographic algorithms such as elliptic curve cryptography (ECC) require different finite field arithmetic operations. Efficient design and implementation of these operations affects the performance of cryptosystems and consequently, has gained lots of interest in the literature, e.g., [1]–[3], and [4]. One of the main finite field arithmetic operations is the multiplication. The shifted polynomial basis (SPB), proposed in [5], is a variation of the polynomial basis (PB). The available works in the literature show that using the SPB results in efficient arithmetic units, e.g., [1], [6]–[9], and [10]. In [1], bit-parallel multipliers are designed for irreducible trinomials and type-II pentanomials, which are faster than the best known polynomial basis and dual basis multipliers. Similarly, it is shown in [6] that the SPB squarers are faster than their PB counterparts. Using the SPB, a new approach for designing sub-quadratic area complexity parallel multipliers is outlined in [7], where the reported multipliers are better than the other similar ones in terms of area and time complexities. Also using the SPB, different bit-parallel multipliers are designed for irreducible pentanomials and trinomials in [8] and [9], respectively. A parallel digit-serial SPB multiplication algorithm is proposed in [10] which has lower time complexity than the PB and Montgomery multiplication (MM) algorithms.

Manuscript received September 24, 2009; revised February 17, 2010 and July 01, 2010; accepted July 21, 2010. Date of publication September 13, 2010; date of current version September 14, 2011.The work of Arash Reyhani-Masoleh was supported in part by an NSERC Discovery grant.

The authors are with the Department of Electrical and Computer Engineering, The University of Western Ontario, London, ON N6A 5B9, Canada (e-mail: hariri@ieee.org, areyhani@uwo.ca).

Digital Object Identifier 10.1109/TVLSI.2010.2066994

A straightforward implementation of the projective Montgomery scalar multiplication requires up to $(m-1)(6M+3A+5S)+(10M+7A+4S+I)$ clock cycles, where M, A, S, and I represent the number of clock cycles for multiplication, addition, squaring, and inversion, respectively [11]. The inversion using Itoh-Tsujii algorithm requires $\lfloor \log_2(m-1) \rfloor + H(m-1) - 1$ multiplications and $m-1$ squarings, where $H(m-1)$ denotes the Hamming weight of $(m-1)$ [11]. As a result, accelerating multiplication significantly affects the performance of an elliptic-curve based crypto-system.

Semi-systolic array structures provide low latency in comparison to systolic array implementations and require fewer latches. Also, they can be pipelined to increase the throughput of the system. In the literature, semi-systolic array implementations have been presented for the finite field multiplication, see for example [12]–[15], and [16]. In the case of the PB, a classic multiplication structure is proposed in [12] which is studied in [13] comprehensively. For the Montgomery multiplication, [14] introduces a semi-systolic structure. Also, [15] and [16] introduce low-latency semi-systolic Montgomery multipliers.

In systolic array structures, the global lines are avoided and the connections are limited to local ones. This results in more efficient VLSI implementations. In case of the PB multiplication, [3] and [17] outline two structures for general irreducible polynomials, respectively. In [18] and [19], optimized structures are proposed for the PB multiplication using general irreducible polynomials and irreducible trinomials. A low latency systolic structure is proposed in [20] for all-one and equally spaced polynomials. Moreover, digit-serial systolic PB multipliers are proposed in [21], [22], and [23] for general irreducible polynomials. A systolic implementation of the PB multiplication is proposed in [24] for irreducible trinomials with a low latency. In case of the Montgomery multiplication, [25] proposes very low latency systolic multipliers for special irreducible polynomials. Also, two scalable structures are proposed in [26] and [27].

The two contributions of this paper are stated as follows. The first contribution of this paper is introducing a new low time-complexity digit-level semi-systolic array structure for the SPB multiplication. The proposed structure is based on a similar technique used in [15], [16], [28], and [10]. In our proposed structure, the parallel operations are balanced and have the same critical path delay. The Montgomery multipliers presented in [29] include non-pipelined structures for special cases of irreducible polynomials. The semi-systolic structure presented in this paper is a low-latency pipelined multiplier with low critical path delay. We show that our proposed semi-systolic multiplier has the least time complexity among the existing ones available in the literature including [12]–[16], and [30]. The second contribution is to propose a digit-level systolic array SPB multiplier which offers a better time complexity, in terms of the combination of the critical path delay and latency, than the existing counterparts for general irreducible polynomials, such as [3], [17], [19], [23], [26], and [27].

The rest of this paper is organized as follows. In Section II, we present our semi-systolic array implementation of the SPB multiplication. In Section III, we propose a digit-level systolic array structure for the SPB multiplication. In Section IV, we provide our implementation results and comparisons. Finally, we conclude this paper in Section V.

II. Semi-Systolic SPB Multiplication

The binary extension field $GF(2^m)$ includes 2^m elements and is associated with an irreducible polynomial defined as

$$F(z) = z^m + f_{m-1}z^{m-1} + \cdots + f_1 z + 1, f_i \in \{0,1\} \quad (1)$$

for $i = 1$ to $m-1$. If x is a root of $F(z)$, i.e., $F(x) = 0$, the set $\{1, x, x^2, \ldots, x^{m-1}\}$ is known as the polynomial basis (PB). Assuming $0 < v \leq m$ is an integer, the Shifted Polynomial Basis (SPB) for $GF(2^m)$ is defined as the set $\{x^{-v}, x^{-v+1}, \ldots, x^{m-v-1}\}$ [5]. Assuming $A, B \in GF(2^m)$, one can write $A = \sum_{i=0}^{m-1} a_i x^{i-v}$ and $B = \sum_{i=0}^{m-1} b_i x^{i-v}$, where $a_i, b_i \in \{0,1\}$. The multiplication in the SPB is defined as

$$C = \sum_{i=0}^{m-1} c_i x^{i-v} = A \cdot B \bmod F(x). \tag{2}$$

Note that C is also a field element of degree $m-1-v$.

By expanding B, the SPB multiplication shown in (2) can be written as

$$\begin{aligned} C = b_0 A x^{-v} + b_1 A x^{-v+1} + \cdots + b_{v-1} A x^{-1} \\ + b_v A + \cdots + b_{m-1} A x^{m-v-1} \bmod F(x). \end{aligned} \tag{3}$$

Now, we split C in (3) into two polynomials as follows:

$$C' = b_0 A x^{-v} + b_1 A x^{-v+1} + \cdots + b_{v-1} A x^{-1} \bmod F(x) \tag{4}$$

$$C'' = b_v A + b_{v+1} A x + \cdots + b_{m-1} A x^{m-v-1} \bmod F(x). \tag{5}$$

Our objective is to implement (4) and (5) independently and in parallel. In this regard, for (4) we define a recursive equation as

$$A'^{(i+1)} = A'^{(i)} \cdot x^{-1} \bmod F(x) \tag{6}$$

where $A'^{(0)} = A$, and $i = 1, \cdots, v-1$ and we write (4) as $C' = b_0 A^{(v)} + b_1 A^{(v-1)} + \cdots + b_{v-1} A^{(1)}$. Now, another recursive equation is defined as

$$C'^{(i+1)} = b_{v-i} A'^{(i)} + C'^{(i)}, \qquad i = 1, \cdots, v \tag{7}$$

where $C'^{(0)} = C'^{(1)} = 0$ and $C' = C'^{(v+1)}$. As a result, C' is obtained after $v+1$ iterations (i.e., clock cycles). This is because one extra iteration is required to compute $A'^{(1)}$. Let $A'^{(i)}$ be represented as $a'^{(i)}_{m-1} x^{m-1} + \cdots + a'^{(i)}_1 x + a'^{(i)}_0$. Now, (6) is written as

$$A'^{(i+1)} = \left(a'^{(i)}_{m-1} x^{m-2} + \cdots + a'^{(i)}_1 + a'^{(i)}_0 x^{-1}\right) \cdot x^{-v} \bmod F(x). \tag{8}$$

Using the fact that $F(x) = 0$, it follows $x^{-1} = x^{m-1} + f_{m-1} x^{m-2} + \cdots + f_1$ [29]. Thus, (8) is rewritten as

$$\begin{aligned} A'^{(i+1)} = \left(a'^{(i)}_0 x^{m-1-v} + (a'^{(i)}_0 f_{m-1} + a'^{(i)}_{m-1}\right) x^{m-2-v} \\ + \cdots + \left(a'^{(i)}_0 f_1 + a'^{(i)}_1\right) x^{-v}. \end{aligned} \tag{9}$$

It can be concluded from (9) that for general irreducible polynomials, this operation has the critical path delay of $T_A + T_X$, where T_A and T_X represent the delay of a two-input AND gate and a two-input XOR gate, respectively. The second recursive equation, (7), is written as $C'^{(i+1)} = (b_{v-i} a'^{(i)}_{m-1} + c'^{(i)}_{m-1}) x^{m-1-v} + \cdots + (b_{v-i} a'^{(i)}_0 + c'^{(i)}_0) x^{-v}$. One can notice that this operation has the critical path delay of $T_A + T_X$ for general irreducible polynomials as well. As (6) and (7) are computed in parallel, the computation of C' in (4) has the critical path delay of $T_A + T_X$ and requires $v+1$ clock cycles (i.e., iterations).

Next, we consider (5). The structure proposed in [13] can be used to implement this part. First, the following recursive equation is defined with the maximum degree of $m-v-1$ using the SPB

$$A''^{(i+1)} = A''^{(i)} \cdot x \bmod F(x) \tag{10}$$

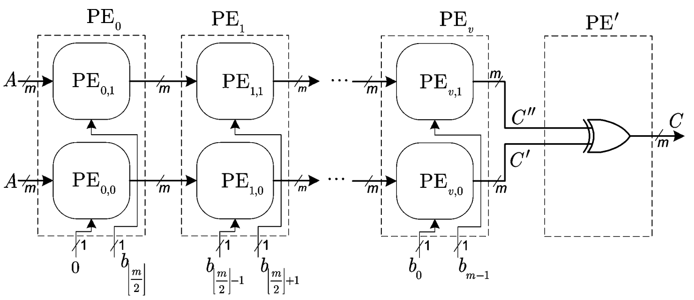

Fig. 1. One-dimensional semi-systolic SPB multiplier.

where $A''^{(0)} = A$ and $i = 0, \cdots, m-v-2$. By rewriting (5) and using (10), one obtains $C'' = b_v A^{(0)} + b_{v+1} A^{(1)} + \cdots + b_{m-1} A^{(m-v-1)}$, which results in the following recursive equation:

$$C''^{(i+1)} = b_{v+i} A''^{(i)} + C''^{(i)}, \qquad i = 0, \cdots, m-v-1 \tag{11}$$

where $C''^{(0)} = 0$ and $C'' = C''^{(m-v)}$. Therefore, C'' is obtained after $m-v$ iterations (i.e., clock cycles). Using the fact that $x^m = f_{m-1} x^{m-1} + \cdots + f_1 x + 1$ and similar to (9), (10) can be realized in hardware with the critical path delay of $T_A + T_X$. Similarly, (11) can be implemented with the critical path delay of $T_A + T_X$.

Similar to C', C'' is obtained by computing (10) and (11) in parallel. As a result, this operation has the critical path delay of $T_A + T_X$ and requires $m-v$ clock cycles (i.e., iterations). It is noted that (4) is a SPB multiplication which only processes v least significant bits of the operand B. Also, (5) is a PB multiplication which processes $m-v$ most significant bits of the operand B. As a result, the delay of obtaining C directly depends on the maximum delay of computing (4) and (5) which require $v+1$ and $m-v$ clock cycles with the critical path delay of $T_A + T_X$, respectively. As C' and C'' are computed in parallel, it is efficient to have equal latencies in computing (4) and (5). Thus, we are interested in the following:

$$v + 1 = m - v \Rightarrow v = \left\lfloor \frac{m}{2} \right\rfloor. \tag{12}$$

Note that (12) implies that m is an odd integer which is the common case in cryptographic applications [31]. Therefore, based on (12), the SPB multiplication can be performed efficiently if we choose $v = \lfloor (m)/(2) \rfloor$. The algorithm associated with this SPB multiplication is shown in Algorithm 1. In each cycle of this algorithm, two bits of B are processed. In Steps 4 and 5 of this algorithm, we compute $C' = \sum_{i=0}^{v} b_{v-1-i}(A \cdot x^{-(i+1)} \bmod F(x))$ and $C'' = \sum_{i=0}^{v} b_{v+i}(A \cdot x^i \bmod F(x))$ which are equal to (4) and (5), respectively. Note that in Algorithm 1, $b_{-1} = 0$.

Algorithm 1: Low time-complexity SPB multiplication algorithm

```
Inputs: $A, B \in GF(2^m), F(x), v = \lfloor (m)/(2) \rfloor$

Output: $C = A \cdot B \bmod F(x)$
 Step 1: $A' := A, C' := 0, A'' := A, C'' := 0$
 Step 2: For $i := 0$ to $\lfloor (m)/(2) \rfloor$
 Step 3:     $A' := A' \cdot x^{-1} \bmod F(x)$
 Step 4:     $C' := b_{v-1-i} A' + C'$
 Step 5:     $C'' := b_{v+i} A'' + C''$
 Step 6:     $A'' := A'' \cdot x \bmod F(x)$
 Step 7: $C := C' + C'' \bmod F(x)$
```

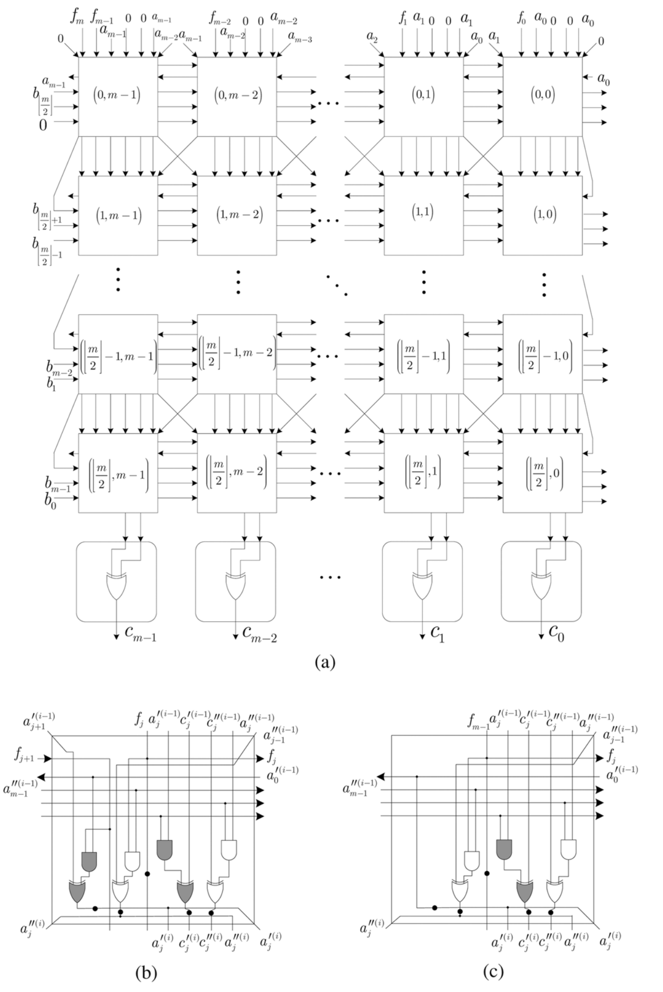

Fig. 2. (a) Two-dimensional semi-systolic SPB multiplier, (b) the cell (i, j), (c) the left-most column cells (the black dots represent latches).

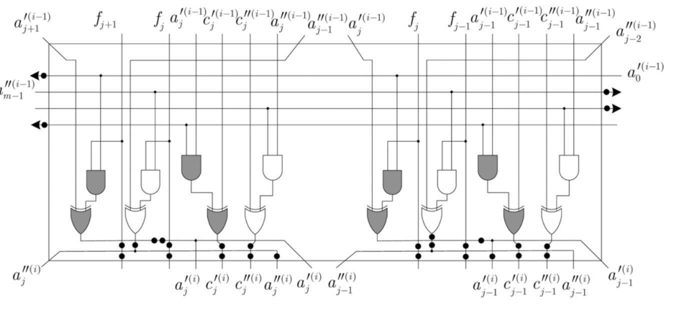

Fig. 3. Digit-level systolic array cell (the black dots represent latches).

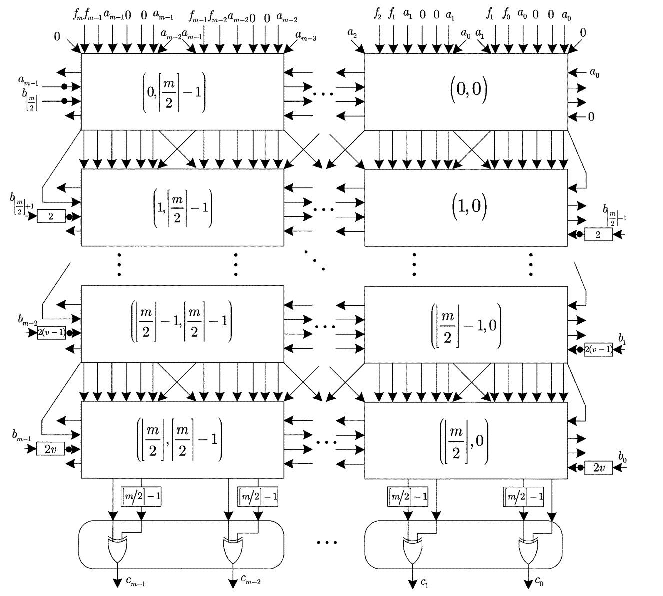

Fig. 4. Digit-level systolic array implementation of the SPB multiplication using general irreducible polynomials.

TABLE I
COMPARISON OF THE SEMI-SYSTOLIC ARRAY FINITE FIELD MULTIPLIERS

Structure	#Cells	A cell	CPD	Latency
[12]: PB	m^2	XOR3: 1 AND: 2 Latch: 3	$T_A + T_{X3}$	m
[13]: PB	m^2	XOR: 2 AND: 2 Latch: 3	$T_A + T_X$	m
[14]: MM	$m \times (m+1)$	XOR3: 1 AND: 2 Latch: 3	$T_A + T_{X3}$	$m+1$
[30]: PB	$(\lfloor \frac{m}{2} \rfloor + 1) \times m$ plus m XOR gates	XOR: 6 AND: 4 Latch: 5	$T_A + T_{X3} + T_M$	$\lfloor \frac{m}{2} \rfloor + 2$
[15]: MM	$(\lfloor \frac{m}{2} \rfloor + 1) \times m$ plus m XOR gates	XOR: 4 AND: 4 Latch: 5	$T_A + 2T_X$	$\lfloor \frac{m}{2} \rfloor + 2$
[16]: MM	$(\lfloor \frac{m}{2} \rfloor + 1) \times m$ plus m XOR gates	XOR3: 2 AND: 4 Latch: 6	$T_A + T_{X3}$	$\lfloor \frac{m}{2} \rfloor + 2$
Fig. 2: SPB	$(\lfloor \frac{m}{2} \rfloor + 1) \times m$ plus m XOR gates	XOR: 4 AND: 4 Latch: 5	$T_A + T_X$	$\lfloor \frac{m}{2} \rfloor + 2$

TABLE II
ASIC IMPLEMENTATION OF SEMI-SYSTOLIC STRUCTURES

Multiplier	Critical path delay (ns)	Latency	Area (μm^2)	Power (mW)
		$m = 31$		
PB [13]	0.21	31	$46,767.75$	54.23
SPB	0.21	17	$41,307.23$	47.56
		$m = 91$		
PB [13]	0.28	91	$417,292.71$	485.66
SPB	0.28	47	$362,039.06$	405.99
		$m = 131$		
PB [13]	0.28	131	$863,966.36$	$1,002.43$
SPB	0.28	67	$752,009.93$	836.67

Now, we present a semi-systolic structure for Algorithm 1 in Fig. 1 using (6), (7), (10), and (11). The main loop in Algorithm 1 (Step 2) has $v+1$ iterations and consequently, the semi-systolic array structure requires $v+1$ processing elements (PEs). The PEs represented by PE_i for $i = 0$ to v in Fig. 1 implement Steps 3–6 of Algorithm 1. To show the parallel operations, these PEs are split into two smaller PEs. The PEs represented by $\mathrm{PE}_{i,0}$ implement Steps 3 and 4, and the ones represented by $\mathrm{PE}_{i,1}$ implement Steps 5 and 6 of the algorithm. This means that $\mathrm{PE}_{i,0}$ realizes (6) and (7), and $\mathrm{PE}_{i,1}$ realizes (10) and (11). Finally, the last step of the algorithm (Step 7) requires a different PE which is labeled as PE' in Fig. 1.

Now, we present this semi-systolic array structure in more details as shown in Fig. 2(a). The cells are shown with two indices represented by (i, j), where $0 \le i \le \lceil (m)/(2) \rceil$ is the row number starting from the top row, and $0 \le j \le m-1$ is the column number starting from the right-hand side column. The row i in Fig. 2(a) represents PE_i shown in Fig. 1. To explain this structure, we show the internal structure of the main cell (i, j) in Fig. 2(b). Note that $\mathrm{PE}_{i,0}$ and $\mathrm{PE}_{i,1}$ have been merged. However, to distinguish between different PEs, we have shown the internal structures of $\mathrm{PE}_{i,0}$ and $\mathrm{PE}_{i,1}$ with gray and white gates, respectively. Corresponding to Steps 3–4 and 5–6, these two sets of gates work in parallel without any interaction. The last row of the structure implements Step 7 of Algorithm 1 (i.e., PE' is Fig. 1) which includes m two-input XOR gates. Note that in the first row of Fig. 2(a),

TABLE III
COMPARISON OF DIFFERENT SYSTOLIC ARRAY GF(2^m) MULTIPLIERS

Structure	Polynomial	#Cells	A cell	CPD	Latency
Bit-Parallel					
[3]: PB	General	m^2	XOR: 2 AND: 2 Latch: 7	$T_A + T_X$	$3m$
[17]: PB	General	m^2	XOR: 1 AND: 2 Latch: 7	$T_A + T_{X3}$	$3m$
[19]: PB	General	m^2	XOR: 2 AND: 2 Latch: 2	$T_A + T_X$	$3m$
[25]: MM	Trinomial	m^2	XOR: 1 AND: 1 Latch: 4	$T_A + T_X$	$m + 1$
Scalable with $d = 2$					
[26]: MM	Trinomial	Total: XOR: 9 AND: 4 SW: 2 Latch: $8\lceil \frac{m}{2} \rceil + 8$		$T_A + T_X$	$\lceil \frac{m}{2} \rceil^2 + 2$
[27]: MM	General	Total: XOR: 6 AND: 4 SW: 6 Latch: $8\lceil \frac{m}{2} \rceil + 12$		$T_A + T_X$	$2 + \lceil \frac{m}{2} \rceil (m + 2)$
Digit-level with $d = 2$					
[23]: PB	General	$(\lceil \frac{m}{2} \rceil)^2$	XOR: 8 AND: 10 MUX: 4 Latch: 21	$2(T_A + T_X) + T_{MUX}$	$3\lceil \frac{m}{2} \rceil$
Fig. 4: SPB	General	$(\lceil \frac{m}{2} \rceil)^2$ plus m XOR gates	XOR: 8 AND: 8 Latch: 28	$T_A + T_X$	$3\lceil \frac{m}{2} \rceil + 2$

one of the inputs b (the bottom horizontal line) is zero since in (4), $A^{(1)} = A \cdot x^{-1} \bmod F(x)$ should be obtained first.

Considering Fig. 2(b), it follows that the critical path delay of this structure is $T_A + T_X$ and its latency is $(\lfloor (m)/(2) \rfloor + 2)$ clock cycle. This structure has two types of cells. As mentioned before, $(\lfloor (m)/(2) \rfloor + 1)$ first rows perform the multiplication (PE_0 to PE_v) and the last row performs the final addition (PE'). As a result, one can state the following for the complexity of this multiplier.

Proposition 1: The semi-systolic implementation of the SPB multiplication includes $m \times (\lfloor (m)/(2) \rfloor + 1)$ cells of the first type (as shown in Fig. 2), each of which contains four two-input AND gates, four two-input XOR gates, and five latches for general irreducible polynomials. Also, the last row requires m two-input XOR gates.

The cells shown in Fig. 2(b) can be further simplified in some cases. Some of the inputs of the cells located on the top row are zero. This results in removing some of the XOR gates. The same case happens in the left-most and right-most columns as well, where $a'^{(i-1)}_{j+1} = 0$ and $a''^{(i-1)}_{j-1} = 0$, respectively. Also, all the cells located on the row labeled $\lfloor (m)/(2) \rfloor$ only produce C' and C''. Consequently, all the gates required to generate the other outputs can be removed. Fig. 2(c) shows a case where the cells located on the left-most column have been optimized based on $a'^{(i-1)}_{j+1} = 0$ and $f_m = 1$.

III. SYSTOLIC ARRAY IMPLEMENTATION OF THE SPB MULTIPLICATION USING GENERAL IRREDUCIBLE POLYNOMIALS

In this section, we design a digit-level systolic SPB multiplier using general irreducible polynomials. From Fig. 2(a), one can notice that the inputs b_is, $0 \leq i < m$, are connected to the cells using global lines which should be removed to achieve a systolic structure. Therefore, it is required to latch all the horizontal connections as well.

Without lack of generality and for sake of simplicity, we set the digit size to $d = 2$. Each basic cell in Fig. 2(a) processes two bits of the operand B. Here, we modify the basic cells shown in Fig. 2(a) to process two bits of the operand A as well. In this regard, we combine two neighboring cells to form a new cell which is shown in Fig. 3. The small dots on the interconnections show the necessary latches.

Fig. 4 shows the digit-level systolic SPB multiplier using general irreducible polynomials. The new cell (i', j') in Fig. 4 is formed by merging the cells $(i', 2j')$ and $(i', 2j' + 1)$ in Fig. 2(a). Since m is chosen to be an odd number in cryptographic applications, the cells in the left-most column have a simpler structure similar to the one shown in Fig. 2(b). The small rectangular blocks on inputs b_is, $0 \leq i < m$, and outputs of row $\lfloor (m)/(2) \rfloor$ represent the number of delay units required to be considered on the corresponding connection. The delay units for the vertical inputs of the first row of the digit-level systolic array have not been depicted in Fig. 4 for simplicity. Here again some of the cells shown in Fig. 4 can be further simplified. This includes all the cells of the first row, where some inputs are fixed to zero and the cells on the second last row where just C' and C'' should be computed.

From Figs. 3 and 4, the critical path delay of this structure is $T_A + T_X$ with the latency of $3\lceil (m)/(2) \rceil + 2$ and each cell requires 8 two-input AND gates, 8 two-input XOR gates, and 28 latches. The total number of the cells is $\lceil (m)/(2) \rceil^2$ plus m two-input XOR gates for the last row. The presented structure can be generalized for other even digit sizes as well. Assuming d is the digit size, the general structure is constructed by merging the cells of $d/2$ rows and d columns in Fig. 2(a). Then, the cut-set systolization technique should be applied. As a result, each cell will process d bits of both A and B.

IV. COMPLEXITY ANALYSIS AND COMPARISONS

The complexity results of the semi-systolic array implementation of the finite field multipliers are summarized in Table I. For all the designs, it is assumed that the input $F(x)$ is latched. Also, T_{X3} and T_M represent the delay of a three-input XOR gate and a multiplexer, respectively. Assuming m is an odd positive integer [31], this structure (without simplification) requires $2m$ more two-input XOR gates and $2m$ more AND gates in comparison with [13]. However, the proposed structure requires about $0.5m^2 - 6m$ less latches than [13] and its latency is almost a half of the latency of the other classic semi-systolic finite field multipliers (e.g., [12], [13], and [14]). In comparison to the existing parallel structures with $d = 2$ (i.e., [15], [16], and [30]), our proposed multiplier offers the least critical path delay with a similar latency and area complexity. Note that the multiplier of [30] requires some multiplexers and they have not been included in Table I. Also, the area complexity of the multiplier proposed in [16] is presented in Table I for general irreducible polynomials without any simplifying assumption.

To further evaluate the proposed semi-systolic SPB multiplier, it has been implemented on 65 nm complimentary metal-oxide-semiconductor (CMOS) ASIC technology using the Synopsys® Design Analyzer® and structural VHDL. We have also implemented the LSB-first semi-systolic PB multiplier of [13] as a good comparison benchmark. The Map Effort was set to medium with a target clock period of 1 ns. The results are presented in Table II for some values of m up to 131 based on our available resources in the laboratory (i.e., memory constraints of the Sun machines). As one can see from the table, both structures have the same critical path delay. Since the proposed structure requires less latches than the multiplier of [13] does, it has lower area and power consumption.

The proposed digit-level systolic multiplier is compared to the existing systolic multipliers in Table III. Note that we have included the multipliers which mostly have been designed for general irreducible polynomials to have a fair comparison. However, some of the multipliers included in this table are designed for trinomials which are expected to have better time and area complexities. It can be seen from the table that the proposed multiplier has the critical path delay of

$T_A + T_X$ and the latency of $3\lceil (m)/(2) \rceil + 2$. One can notice from Table III that the proposed digit-level systolic array SPB multiplier has a better time complexity while its area complexity is comparable to the existing structures reported in [3], [17], [19], and [23]. In comparison to [26] and [27], our proposed multiplier is faster however it has a higher area complexity. The multiplier reported in [25] for irreducible trinomials has a better time and area complexities (the lower bound of the area has been reported in Table III). It is noted that this is expected since having the restriction of the polynomial to be a trinomial simplifies the multiplication algorithm and the hardware implementation. However, our multiplier is designed for general irreducible polynomials without any assumptions. Comparing to the multipliers presented in this paper, the multipliers of [29] are in a different category of multipliers (non-pipelined multipliers). The MSB-first bit-serial Montgomery multiplier presented in [29] has the same critical path delay of $T_A + T_X$, however its latency is m clock cycles whereas the latency of the semi-systolic multiplier proposed in this paper is $\lfloor m/2 \rfloor + 2$ clock cycles and it is pipelined. It can be concluded from Table III that our proposed multiplier is faster than the existing ones designed for general irreducible polynomials.

V. Conclusion

In this paper, we have proposed a digit-level semi-systolic array SPB multiplier which has the critical path delay of $T_A + T_X$ with the latency of $\lfloor m/2 \rfloor + 2$. This structure outperforms the existing semi-systolic structures in terms of time complexity (combination of critical path delay and latency). Also, we have designed a digit-level systolic array SPB multiplier which has the critical path delay of $T_A + T_X$ and the latency of $3\lceil (m)/(2) \rceil + 2$. The complexity results show that our proposed systolic structure has a better time complexity (combination of critical path delay and latency) than the existing counterparts using general irreducible polynomials.

Acknowledgment

The authors would like to thank the reviewers for their constructive comments.

References

[1] H. Fan and M. Hasan, "Fast bit parallel shifted polynomial basis multipliers in $GF(2^n)$," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 53, no. 12, pp. 2606–2615, Dec. 2006.

[2] A. Reyhani-Masoleh and M. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

[3] C.-S. Yeh, I. S. Reed, and T. K. Truong, "Systolic multiplier for finite fields $GF(2^m)$," *IEEE Trans. Comput.*, vol. C-33, pp. 357–360, Apr. 1984.

[4] T. Beth and D. Gollman, "Algorithm engineering for public key algorithms," *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458–466, May 1989.

[5] H. Fan and Y. Dai, "Fast bit-parallel $GF(2^n)$ multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485–490, Apr. 2005.

[6] S. M. Park and K. Y. Chang, "Low complexity bit-parallel squarer for $GF(2^n)$ defined by irreducible trinomials," *IEICE Trans. Fundam. Electron., Commun. and Comput. Sci.*, vol. 89, pp. 2451–2452, 2006.

[7] H. Fan and M. Hasan, "A new approach to subquadratic space complexity parallel multipliers for extended binary fields," *IEEE Trans. Comput.*, vol. 56, no. 2, pp. 224–233, Feb. 2007.

[8] S. M. Park, K. Y. Chang, and D. Hong, "Efficient bit-parallel multiplier for irreducible pentanomials using a shifted polynomial basis," *IEEE Trans. Comput.*, vol. 55, no. 9, pp. 1211–1215, Sep. 2006.

[9] C. Negre, "Efficient parallel multiplier in shifted polynomial basis," *J. Syst. Architecture*, vol. 53, no. 2–3, pp. 109–116, 2007.

[10] A. Hariri and A. Reyhani-Masoleh, "Digit-serial structures for the shifted polynomial basis multiplication over binary extension fields," in *Proc. 2nd Int. Workshop Arithmetic Finite Fields (WAIFI)*, 2008, vol. 5130, pp. 103–116, ser. Lecture Notes in Computer Science.

[11] B. Ansari and M. Hasan, "High performance architecture of elliptic curve scalar multiplication," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[12] B. A. Laws and C. K. Rushforth, "A cellular-array multiplier for GF (2^m)," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 1573–1578, Dec. 1971.

[13] S. K. Jain, L. Song, and K. K. Parhi, "Efficient semisystolic architectures for finite-field arithmetic," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 1, pp. 101–113, Mar. 1998.

[14] C. W. Chiou, C. Y. Lee, A. W. Deng, and J. M. Lin, "Concurrent error detection in montgomery multiplication over $GF(2^m)$," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E89-A, no. 2, pp. 566–574, Feb. 2006.

[15] L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, "Balanced point operations for side-channel protection of elliptic curve cryptography," *IEE Proc. Inf. Security*, vol. 152, no. 1, pp. 57–65, Oct. 2005.

[16] P. K. Meher, "Systolic formulation for low-complexity serial-parallel implementation of unified finite field multiplication over $GF(2^m)$," in *Proc. IEEE Int. Conf. Appl.-specific Syst., Architectures Processors*, Montreal, QC, 2007, pp. 134–139.

[17] C. L. Wang and J. L. Lin, "Systolic array implementation of multipliers for finite fields $GF(2^m)$," *IEEE Trans. Circuits Syst.*, vol. 38, no. 7, pp. 796–800, Jul. 1991.

[18] C. Y. Lee, "Low-complexity bit-parallel systolic multipliers over $GF(2^m)$," *Integr., VLSI J.*, vol. 41, no. 1, pp. 106–112, Jan. 2008.

[19] S. Kwon, C. H. Kim, and C. P. Hong, "More efficient systolic arrays for multiplication in $GF(2^m)$ using LSB first algorithm with irreducible polynomials and trinomials," *Comput. Electr. Eng.*, vol. 35, no. 1, pp. 159–167, Jan. 2009.

[20] 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, p. 385, May 2001.

[21] J. H. Guo and C. L. Wang, "Digit-serial systolic multiplier for finite fields $GF(2^m)$," *IEE Proc. Comput. Digital Techn.*, vol. 145, no. 2, pp. 143–148, Mar. 1998.

[22] C. H. Kim, S. D. Han, and C. P. Hong, "An efficient digit-serial systolic multiplier for finite fields $GF(2^m)$," in *Proc. 14th Annu. IEEE Int. ASIC/SOC Conf.*, Arlington, VA, 2001, pp. 361–365.

[23] C. H. Kim, C. P. Hong, and S. Kwon, "A digit-serial multiplier for finite field $GF(2^m)$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 4, pp. 476–483, Apr. 2005.

[24] C. Y. Lee, "Low complexity bit-parallel systolic multiplier over $GF(2^m)$ using irreducible trinomials," *IEE Proc. Comput. Digital Tech.*, vol. 150, no. 1, pp. 39–42, Jan. 2003.

[25] 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.

[26] C. Y. Lee, C. W. Chiou, J. M. Lin, and C. C. Chang, "Scalable and systolic montgomery multiplier over $GF(2^m)$ generated by trinomials," *IET Circuits, Devices Syst.*, vol. 1, no. 6, pp. 477–484, Dec. 2007.

[27] C. C. Chen, C. Y. Lee, and E. H. Lu, "Scalable and systolic montgomery multipliers over $GF(2^m)$," *IEICE Trans. Fund. Electron., Commun. Comput. Sci.*, vol. E91-A, no. 7, pp. 1763–1771, Jul. 2008.

[28] M. E. Kaihara and N. Takagi, "Bipartite modular multiplication method," *IEEE Trans. Comput.*, vol. 57, no. 2, pp. 157–164, Feb. 2008.

[29] A. Hariri and A. Reyhani-Masoleh, "Bit-serial and bit-parallel montgomery multiplication and squaring over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1332–1345, Oct. 2009.

[30] S. Moon, J. Park, and Y. Lee, "Fast VLSI arithmetic algorithms for high-security elliptic curve cryptographic applications," *IEEE Trans. Consum. Electron.*, vol. 47, no. 3, pp. 700–708, Aug. 2001.

[31] NIST, "Recommended Elliptic Curves for Federal Government Use," Jul. 1999 [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf