A New Bit-Serial Architecture for Field Multiplication Using Polynomial Bases

Arash Reyhani-Masoleh

Department of Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada
areyhani@uwo.ca

Abstract. Multiplication is the main finite field arithmetic operation in elliptic curve cryptography and its bit-serial hardware implementation is attractive in resource constrained environments such as smart cards, where the chip area is limited. In this paper, a new serial-output bit-serial multiplier using polynomial bases over binary extension fields is proposed. It generates a bit of the multiplication in each clock cycle with the latency of one cycle. To the best of our knowledge, this is the first time that such a serial-output bit-serial multiplier architecture using polynomial bases for general irreducible polynomials is proposed.

Key words: Finite or Galois field, Mastrovito multiplier, polynomial basis, bit-serial multiplier

1 Introduction

The multiplication over finite (or Galois) field $GF(2^m)$ is the main arithmetic operation in the elliptic curve cryptography [7, 11] and choosing a suitable basis plays an important role in efficient implementation [6]. A field element can be represented using different bases, such as polynomial basis (PB), normal basis, and dual basis. Among them, representation of field elements using a polynomial basis is simpler and has received more attention for hardware implementation.

A hardware implementation of a finite field multiplier can be categorized either as a bit-parallel or bit-serial type. In a bit-parallel multiplier over $GF(2^m)$, once $2m$ bits of two inputs are received, m bits of the product are obtained together at the output after a propagation delay through various logic gates. Such a parallel type multiplier (see for example [16, 10, 15, 5, 18, 13, 12]) requires $O(m^2)$ number of gates. On the other hand, a bit-serial multiplier takes m clock cycles for one multiplication using $O(m)$ number of gates.

Bit-serial multipliers can be categorized into two types of either parallel or serial output. In the parallel-output bit-serial (POBS) multipliers, all m output bits of the product are available at the end of the m-th cycle, whereas serial-output bit-serial (SOBS) multipliers generate one bit of the product in each of these m cycles. Examples of the former type includes the well known LSB- and MSB-first bit-serial polynomial basis multipliers [14, 3] and the normal basis

multiplier due to Agnew et al. [1] while those of the latter type are Berlekamp's bit-serial dual basis multiplier [2] and Massey-Omura's original bit-serial normal basis multiplier [8]. Usually, POBS multipliers run at a much higher clock rate than their SOBS counterparts. However, the latency to generate the first bit of the product in the SOBS multipliers is one clock cycle as compared to m clock cycles for the POBS ones. Therefore, in applications that require implementation on resource constrained environment such as smart cards, SOBS multipliers result in faster overall computation than POBS multipliers since such a system is usually running at low operating clock frequency. In this paper, we propose a new SOBS PB multiplier for a general irreducible polynomial. To the best of our knowledge, this is the first time that a SOBS PB multiplier is proposed for general polynomials.

The organization of this article is as follows. In Section 2, the traditional bit-serial architectures for PB multiplication over $GF(2^m)$ are introduced. In Section 3, the matrix formulations for the PB multiplication is revisited. Then, we derive formulations for the proposed multiplier structure. A new serial-output bit-serial multiplier is proposed in Section 4. Finally, conclusions are given in Section 5.

2 Traditional Bit-Serial Multipliers over $GF(2^m)$

The finite field $GF(2^m)$ consists of 2^m field elements and is constructed by the polynomial basis $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$, where α is a root of the irreducible polynomial

$$P(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1. \tag{1}$$

In (1), $1 \leq t_0 < t_1 < \cdots < t_{\omega-2}$, and ω is the number of non-zero terms. Then, each field element $B \in GF(2^m)$ can be written with respect to this basis as

$$B = (b_{m-1}, \cdots, b_1, b_0) = \sum_{i=0}^{m-1} b_i \alpha^i,\ b_i \in \{0, 1\}, \tag{2}$$

where b_is are the coordinates of B. For convenience, these coordinates will be denoted in vector notation as

$$\mathbf{b} = [b_0, b_1, \cdots, b_{m-1}]^T, \tag{3}$$

where T denotes the transposition of a vector or a matrix.

There are two types of bit-serial, namely LSB-first and MSB-first, multipliers [3]. The LSB-first bit-serial multiplier is shown in Figure 1(a). In this multiplier structure, both $X = \langle x_{m-1}, \cdots, x_1, x_0 \rangle$ and $Y = \langle y_{m-1}, \cdots, y_1, y_0 \rangle$ are m bit registers. Let $X(n)$ and $Y(n)$ denote the contents of X and Y at the n-th, $0 \leq n \leq m$, clock cycle, respectively. Suppose the X register in Figure 1(a) is initialized with A, i.e., $X(0) = A$, then the output of this register at the n-th

clock cycle is $X(n) = X^{(n)} \in GF(2^m)$, which is calculated from the input of this register, i.e., $X^{(n-1)}$, using the α module shown in Figure 1(a) as

$$X^{(n)} = \alpha \cdot X^{(n-1)} \mod P(\alpha), \quad 1 \le n \le m-1, \tag{4}$$

where $X^{(0)} = A$. Also, suppose that the register Y is initially cleared, i.e., $Y(0) = 0$. Then, one can obtain the content of Y at the first clock cycle as $Y(1) = b_0 A$ and in general at the n-th clock cycle as $Y(n) = b_0 A + \sum_{i=1}^{n-1} b_i X(i)$, $1 < n \le m$. Let C denote the PB multiplication of A and B, i.e., $C = AB \mod P(\alpha)$. Then, using (2) and (4) recursively, one can obtain

$$C = \sum_{i=0}^{m-1} b_i \cdot ((A\alpha^i) \mod P(\alpha)) \tag{5}$$

$$= \sum_{i=0}^{m-1} b_i \cdot X^{(i)}, \tag{6}$$

and noting the fact that $X(n) = X^{(n)}$, one can determine that after m clock cycles Y contains $C = AB \mod P(\alpha) \in GF(2^m)$, i.e., $Y(m) = C$. The implementation of $b_i \cdot X^{(i)}$ in (6) is done using m 2-input AND gates. This is shown with the double circle module with a dot inside in Figure 1(a). Also, the sum operation in (6) is implemented with m 2-input XOR gates which is shown with a double circle module with a plus inside. Since the coordinates of B enter the multiplier from the least significant bit (LSB), i.e., b_0, this multiplier is referred to as the LSB first bit-serial multiplier.

The MSB-first bit-serial multiplier is shown in Figure 1(b). This structure implements

$$C = (((b_{m-1}A\alpha + b_{m-2}A)\alpha + b_{m-3}A) + \cdots + b_1 A)\alpha + b_0 A, \tag{7}$$

where the $\mod P(\alpha)$ operations after multiplications by α are omitted for simplicity. If the registers U and V are initialized with $A = (a_{m-1}, \cdots, a_1, a_0)$ and $0 = (0, \cdots, 0, 0)$, respectively, then one can verify that after the m-th clock cycle the register V contains the coordinates of C, i.e., $V(m) = C$. It is noted that for parallel load of inputs into the registers in Figure 1, multiplexers may be used. These are not shown in the figure for simplicity.

3 Matrix Formulations for PB Multiplication Revisited

In [10, 9], Mastrovito showed that the coordinates of $C = AB \mod P(\alpha)$ are obtained from the matrix-by-vector product of $\mathbf{c} = [c_0, c_1, \cdots, c_{m-1}]^T = \mathbf{M} \cdot \mathbf{b}$, where $\mathbf{M}$ is an $m \times m$ binary matrix whose entries depend on the coordinates of A and the entries of the *reduction matrix* $\mathbf{Q} = [q_{i,j}]$, $0 \le i \le m-2$, $0 \le j \le m-1$, defined by [9]

$$[\alpha^m, \alpha^{m+1}, \cdots, \alpha^{2m-2}]^T \equiv \mathbf{Q}[1, \alpha, \cdots, \alpha^{m-1}]^T \pmod{P(\alpha)}. \tag{8}$$

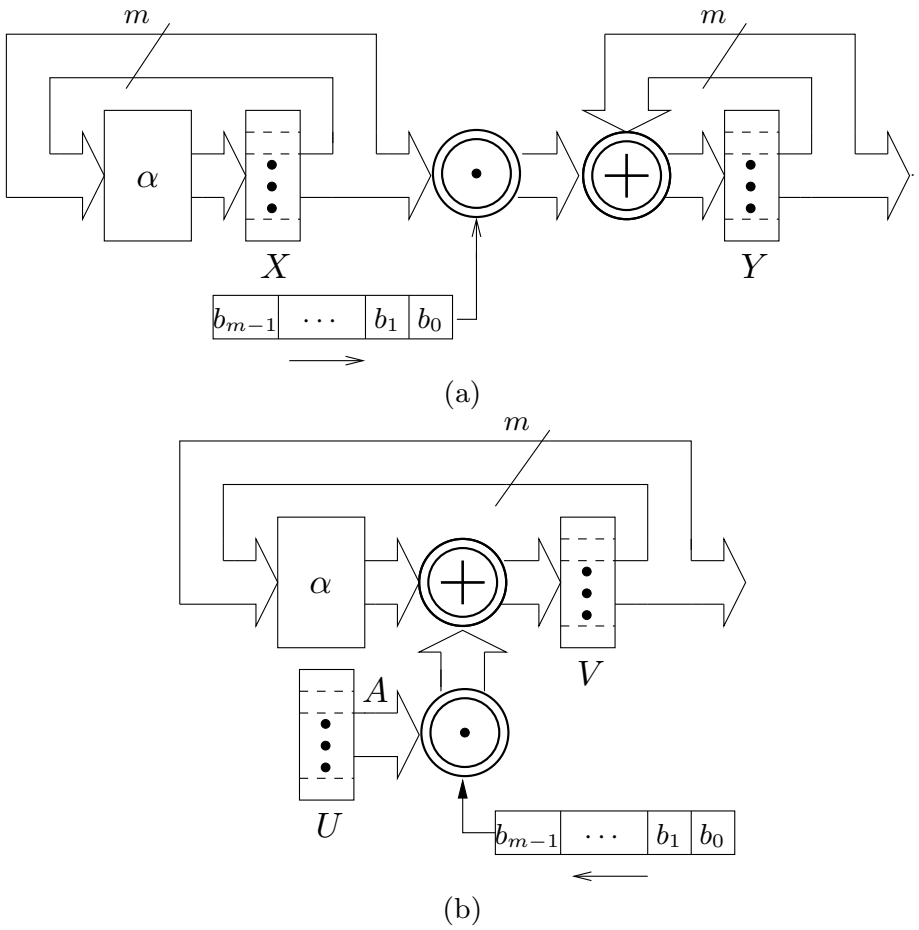

Fig. 1. (a) LSB first bit-serial multiplier. (b) MSB first bit-serial multiplier.

The Mastrovito matrix $\mathbf{M}$ has been studied in [15] and [5] for irreducible trinomials and arbitrary polynomials, respectively. Then, a systematic design to obtain the Mastrovito matrix $\mathbf{M}$ for general irreducible polynomials is presented in [18].

To find the PB multiplication, another approach is proposed in [17] and [12] for irreducible trinomials and arbitrary polynomials, respectively. The multiplication operation in this approach consists of two parts of the *product* of two field elements $A = (a_{m-1}, \cdots, a_1, a_0), B \in GF(2^m)$, i.e., AB, followed by the *modular reduction*, i.e., $C = AB \mod P(\alpha)$. Let us denote the result of the product of two polynomials

$$AB = \sum_{j=0}^{m-1} b_j \alpha^j A = D + \alpha^m E, \tag{9}$$

where $D = (d_{m-1}, \cdots, d_1, d_0)$ and $E = (0, e_{m-2}, \cdots, e_1, e_0)$ are the field elements in $GF(2^m)$. It is shown in [12] that the coordinates of E and D can be obtained from the following:

$$\mathbf{d} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \end{bmatrix} = \mathbf{L}\mathbf{b} = \begin{bmatrix} a_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & \cdots & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix}, \tag{10}$$

$$\mathbf{e} = \begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{m-2} \end{bmatrix} = \mathbf{U}\mathbf{b} = \begin{bmatrix} 0 & a_{m-1} & \cdots & a_2 & a_1 \\ 0 & 0 & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{m-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix}. \tag{11}$$

Then, one can calculate the coordinates of $C = (c_{m-1}, \cdots, c_1, c_0)$ from the following reduction equation [12]

$$\mathbf{c} = [c_0, c_1, \cdots, c_{m-1}]^T = \mathbf{d} + \mathbf{Q}^T\mathbf{e}. \tag{12}$$

Let us define the down shift of the matrix $\mathbf{S}$ by j rows as $\mathbf{S}[\downarrow j]$ and the right shift of $\mathbf{S}$ by i columns as $\mathbf{S}[\rightarrow \mathbf{i}]$, where the emptied positions after the shifts are filled by zeros. Then, it is shown in [4] that the $\mathbf{Q}^T$ matrix in (12) can be represented as

$$\mathbf{Q}^T = \sum_{i \in N} \sum_{j \in \mathcal{T}} \mathbf{I}_{\mathbf{m \times (m-1)}} [\downarrow \mathbf{j}][\rightarrow \mathbf{i}], \tag{13}$$

where the sets $N \subset \{0, 1, \cdots, m-1\}$, $\mathcal{T} = \{0, t_1, \cdots, t_{\omega-2}\}$ (see $P(x)$ in (1)) and

$$\mathbf{I}_{\mathbf{m \times (m-1)}} = \begin{bmatrix} \mathbf{I_{m-1 \times m-1}} \\ \hline \mathbf{0_{1 \times m-1}} \end{bmatrix}. \tag{14}$$

In (14), $\mathbf{I_{m-1\times m-1}}$ is an $m-1\times m-1$ unity matrix and $\mathbf{0_{1\times m-1}}$ is a zero row vector with $m-1$ zero entries. Then, using (13), the matrix reduction equation of (12) is simplified in [4] to

$$\mathbf{c} = \mathbf{d} + \sum_{j\in\mathcal{T}} \mathbf{e}'[\downarrow j], \tag{15}$$

where

$$\mathbf{e}'[\downarrow j] = [\underbrace{0, \cdots, 0}_{j}, e'_0, \cdots, e'_{m-1-j}]^T \text{ for } j > 0, \tag{16}$$

and

$$\mathbf{e}' = \mathbf{e}'[\downarrow 0] = [e'_0, \cdots, e'_{m-2}, 0]^T = \sum_{i\in N} \mathbf{I}_{\mathbf{m\times(m-1)}}\,[\rightarrow \mathbf{i}]\mathbf{e}. \tag{17}$$

It is noted that to obtain the set $N \subset \{0, 1, \cdots, m-1\}$ in (17), one can use the algorithm proposed in [18]. For the irreducible polynomial $P(x)$ with the second highest degree $t_{\omega-2} \leq (m+1)/2$, it is proved in [4] that $N = \{0, m - t_{\omega-2}, \cdots, m - t_1\}$. In the following, we show another approach to find this set for arbitrary irreducible polynomial.

For a given irreducible polynomial $P(x)$ stated in (1), the reduction matrix defined in (8) is fixed. Thus, the entries of $\mathbf{Q}$ are constant, i.e., $q_{i,j} \in \{0, 1\}$, and can be found from (8) for the underlying polynomial $P(x)$. Let us assume the entries of column 0 of $\mathbf{Q}$, i.e., $q_{i,0}$, $0 \leq i \leq m-2$, are given. Let n and r_j $(0 \leq j \leq n-1)$ be the number of nonzero entries and their row positions of the column 0 in this matrix, respectively, i.e.,

$$q_{i,0} = 1, \text{ for } i \in \mathcal{R}, \tag{18}$$

where

$$\mathcal{R} = \{r_0, r_1, \cdots, r_{n-1}\}.$$

This column is equal to the row 0 of $\mathbf{Q}^T$ and is obtained from (13) for $j = 0$. Then, one can easily see that $\mathcal{R} = N$, i.e., the elements of N are the locations of non-zero entries of column 0 of the reduction matrix.

Remark 1. Using (8) and $x^m = \sum_{i=1}^{\omega-2} x^{t_i} + 1$ which is obtained from (1), one can easily see that $r_0 = 0$ for any irreducible polynomial [4].

Remark 2. It is noted that for the irreducible trinomial $P(x) = x^m + x + 1$, i.e., $t_{\omega-2} = 1$, $\omega = 3$, the column 0 of $\mathbf{Q}$ has only one nonzero entry, i.e., $n = 1$, which is in the row $r_0 = 0$.

Remark 3. If $t_{\omega-2} > 1$, then the second nonzero entry in the column 0 of $\mathbf{Q}$ is $r_1 = m - t_{\omega-2}$.

In the following, we slightly simplify $\mathbf{e}'$ in (17) to present the key formulation for the proposed SOBS multiplier. Since

$$\mathbf{I_{m\times(m-1)}}[\rightarrow \mathbf{i}] = \left[\begin{array}{c|c} \mathbf{0_{m\times i}} & \begin{array}{c} \mathbf{I_{m-1-i\times m-1-i}} \\ \hline \mathbf{0_{i+1\times m-1-i}} \end{array} \end{array}\right], \tag{19}$$

one can see that $\mathbf{I_{m\times(m-1)}}\,[\rightarrow \mathbf{i}]\mathbf{e}$ is equal to the up shift of the vector $[e_0, \cdots, e_{m-2}, 0]^T$ by i rows, i.e.,

$$\mathbf{e}[\uparrow i] = [e_i, \cdots, e_{m-2}, \underbrace{0, \cdots, 0}_{i+1}]^T. \tag{20}$$

Therefore, we conclude the above discussion to state the following.

Lemma 1. *Let the finite field $GF(2^m)$ be constructed by the general irreducible polynomial $P(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$, then the coordinates of the PB multiplication of $C = AB \mod P(\alpha)$ can be obtained from two steps of*

$$\mathbf{e}' = [e'_0, \cdots, e'_{m-2}, 0]^T = \sum_{i\in\mathcal{R}} \mathbf{e}[\uparrow i] \tag{21}$$

followed by

$$\mathbf{c} = \mathbf{d} + \sum_{j\in\mathcal{T}} \mathbf{e}'[\downarrow j], \tag{22}$$

where $\mathbf{d}$, $\mathbf{e}$, $\mathbf{e}[\uparrow i]$ *and* $\mathbf{e}'[\downarrow j]$ *are obtained from (10), (11), (20) and (16), respectively.*

Proposition 1. *The reduction matrix method stated by (21) and (22) in Lemma 1 requires*

$$(m-1)(n+\omega-2) - \sum_{i=1}^{n-1} r_i - \sum_{j=1}^{\omega-2} t_j \tag{23}$$

number of two-input XOR gates with the critical path delay of at most

$$(\lceil \log_2 n \rceil + \lceil \log_2 \omega \rceil)\, T_X, \tag{24}$$

where T_X is the time delay of an XOR gate.

Proof. The number of bit-wise addition (XOR gates) required for (21) is

$$\sum_{i=1}^{n-1}(m-1-r_i) = (m-1)(n-1) - \sum_{i=1}^{n-1} r_i. \tag{25}$$

Similarly, implementation of (22) requires

$$m-1+\sum_{j=1}^{\omega-2}(m-1-t_j) = (m-1)(\omega-1) - \sum_{j=1}^{\omega-2} t_j. \tag{26}$$

Thus, by adding (25) and (26), the proof of (23) is complete. The time delay of (24) is obtained if we add the delay of (21), i.e., $\lceil \log_2 n \rceil\, T_X$, with the delay of (22), i.e., $\lceil \log_2 \omega \rceil\, T_X$.

4 New Serial-Output Bit-Serial Multiplier

Unlike the bit-serial multipliers presented in Section 2, this multiplier generates one bit of the multiplication in each clock cycle with the latency of one clock cycle.

4.1 Architecture

In order to develop a bit-serial multiplier, Lemma 1 is used to generate the coordinates of C in the order of c_0, followed by c_1, $\cdots$, and c_{m-1}. The new architecture, which is referred to as serial-output bit-serial (SOBS) multiplier, is shown in Figure 2(a). It consists of one register $B = \langle b_0, b_1, \cdots, b_{m-1} \rangle$ which contains the coordinates of the field element $B = (b_{m-1}, \cdots, b_1, b_0)$ as well as three shift registers $L = \langle l_{m-1}, \cdots, l_1 \rangle$, $U = \langle u_{m-1}, \cdots, u_1, u_0 \rangle$, and $X = \langle x_1, x_2, \cdots, x_{t_{\omega-2}} \rangle$.

As seen in this figure, the output of shift register L are connected to $n-1$ right shift (RS) blocks as well as the BTX array. The $\mathrm{RS}(r_i)$, $1 \le i \le n-1$, block shifts the $m-1-r_i$ left most input lines to the right by r_i positions. Let the input of the re-wiring $\mathrm{RS}(r_i)$ block be $L = \langle l_{m-1}, \cdots, l_1 \rangle$, the output of the $\mathrm{RS}(r_i)$ block is $L_{\rightarrow r_i} = \langle \underbrace{-, \cdots, -}_{r_i}, l_{m-1}, \cdots, l_{r_i+1} \rangle$, where $-$ denotes nothing is connected to those r_i left-most coordinates. The outputs of $\mathrm{RS}(r_1)$ and $\mathrm{RS}(r_{n-1})$ blocks, i.e., $L_{\rightarrow r_1}$ and $L_{\rightarrow r_{n-1}}$, respectively, are shown in Figure 2(b). This figure also shows how the outputs of the BTX array, i.e., $\langle v_{m-1}, \cdots, v_1 \rangle$, are obtained. As seen in Figure 2(b), the BTX array requires $m-1-r_1$ BTXs whose number of inputs vary from 2 to n. Specifically, it consists of $m-1-r_{n-1}$ BTXs with n inputs, $r_{n-1}-r_{n-2}$ BTXs with $n-1$ inputs, $\cdots$, and r_2-r_1 BTXs with 2 inputs, i.e., 2-input XOR gates. In general, the BTX array includes $r_{i+1}-r_i$ BTXs with $i+1$ inputs for $1 \le i \le n-1$ (assume $r_n = m-1$). Therefore, as seen in Figure 2(b), the outputs of the BTX array, i.e., v_is, are obtain as follows:

$$v_i = \begin{cases} l_i, & \text{if } m-r_1 \le i \le m-1 \\ l_i + l_{i+r_1}, & \text{if } m-r_2 \le i \le m-1-r_1 \\ l_i + l_{i+r_1} + l_{i+r_2}, & \text{if } m-r_3 \le i \le m-1-r_2 \\ \vdots & \vdots \\ l_i + \sum_{j=1}^{n-1} l_{i+r_j}, & \text{if } 1 \le i \le m-1-r_{n-1}. \end{cases} \tag{27}$$

Using Figure 2(b) or (27), one can obtain the number of XOR gates required for realizing the BTX array in Figure 2(a) as

$$\#\,\mathrm{XOR}_{BTX\ array} = \sum_{i=1}^{n-1}(m-1-r_i) = (n-1)(m-1) - \sum_{i=1}^{n-1} r_i. \tag{28}$$

Also, the time delay of the longest path between the inputs and outputs of the BTX array is $\lceil \log_2 n \rceil\, T_X$.

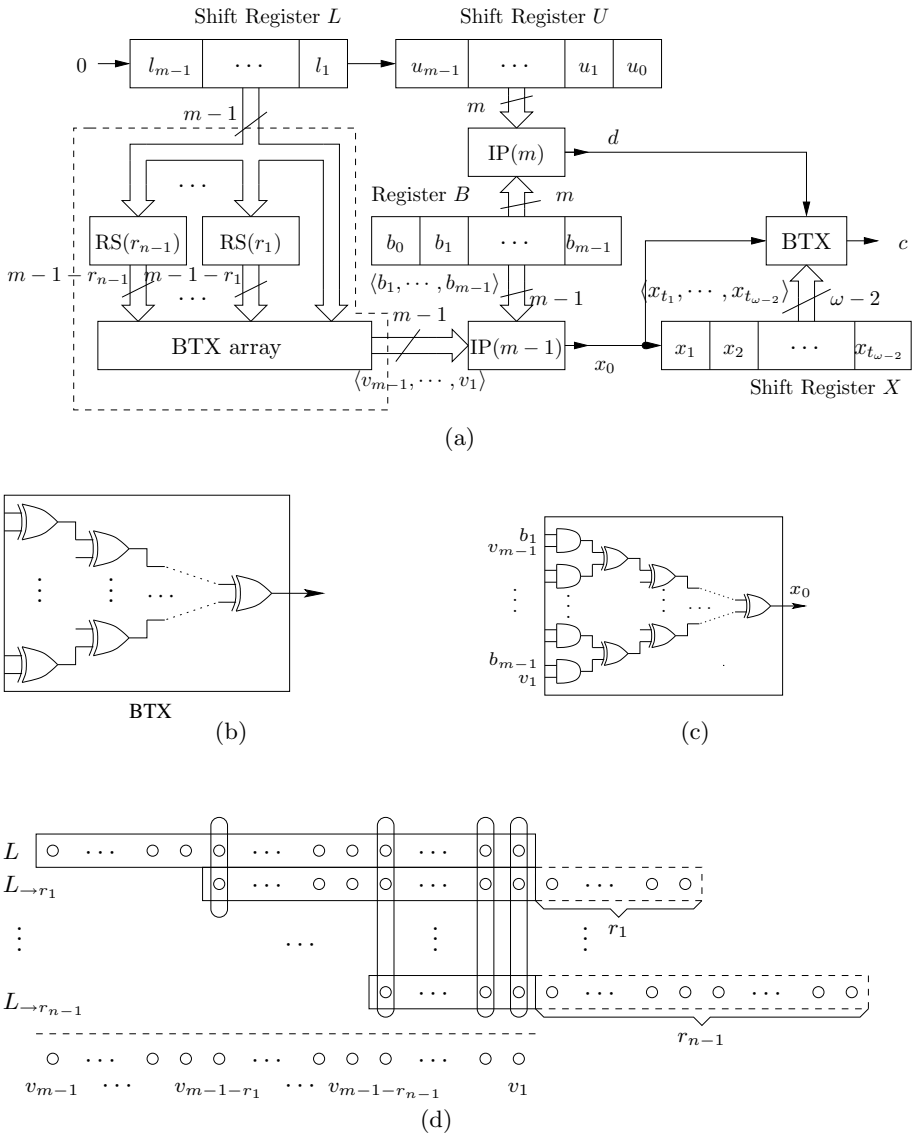

Fig. 2. (a) The architecture of serial output bit-serial (SOBS) PB multiplier over $GF(2^m)$. (b) The details of binary tree of XOR (BTX) gates. (c) The architecture of IP$(m-1)$, i.e., inner product with $m-1$ AND gates. (d) The BTX array output which requires $m-1-r_1$ BTXs.

Figure 2(a) also consists of two inner product (IP) blocks as denoted by IP(m) and IP($m-1$). Figure 2(c) shows the architecture of IP($m-1$) which implements

$$x_0 = \sum_{i=1}^{m-1} b_i v_{m-i} = [v_{m-1}, \cdots, v_1][b_1, \cdots, b_{m-1}]^T \tag{29}$$

using $m-1$ AND gates and $m-2$ XOR gates with $T_A + \lceil \log_2(m-1) \rceil T_X$ time delay. Similarly, the output of IP(m) generates

$$d = \sum_{i=0}^{m-1} b_i u_{m-i-1}, \tag{30}$$

which requires m AND gates and $m-1$ XOR gates with $T_A + \lceil \log_2 m \rceil T_X$ time delay.

4.2 Initialization and Multiplication Operation

In this section we show that by properly initialization of the shift registers, the bit-serial multiplier generates the coordinates of C in such a way that c_0 and c_{m-1} are the first and last bits output from c, respectively.

Let us initialize the shift register L and U with the coordinates of A as

$$L(0) = \langle a_{m-1}, \cdots, a_1 \rangle, \ U(0) = \langle a_0, 0, \cdots, 0 \rangle. \tag{31}$$

In fact, only one bit of U, i.e., u_{m-1}, is initialized with a_0 and other bits are cleared. Also, the register B is initialized with the coordinates of B as $B(0) = \langle b_0, b_1, \cdots, b_{m-1} \rangle$ and its contents remains unchanged during each clock cycle until the end of multiplication process. Thus, we can state that $B(\tau) = \langle b_0, b_1, \cdots, b_{m-1} \rangle$ for $0 \le \tau \le m-1$, where τ denotes the number of clock cycles applied after initialization ($\tau = 0$). Also, we assume that the contents of the shift register X are cleared initially, i.e., $X(0) = \langle x_1, x_2, \cdots, x_{t_{\omega-2}} \rangle = \langle \underbrace{0, 0, \cdots, 0}_{t_{\omega-2}} \rangle$.

It is noted that for parallel load of A and B into the registers L and B and the last bit of U, multiplexers may be used. Those are not shown in the figure for simplicity. However, for serial load such multiplexers are not needed.

Let $x_0(\tau)$ denote the output of IP($m-1$) in Figure 2(a) after the τ-th clock cycle. Then, by substituting (31) into (27) and using (29), one can obtain the initial value of the output of IP($m-1$) in Figure 2(a) as

$$x_0(0) = \left(\sum_{i \in \mathcal{R}} [\underbrace{0, \cdots, 0}_{i}, a_{m-1}, \cdots, a_{i+1}] \right) [b_1, \cdots, b_{m-1}]^T. \tag{32}$$

Using (11) and (21), one can simplify (32) to $x_0(0) = \sum_{i \in \mathcal{R}} e_i = e'_0$. Similarly, let $U(\tau)$ and $d(\tau)$ be the contents of the shift register U and signal d in Figure

2(a) after the τ-th, $0 \le \tau \le m-1$, clock cycle. Then, by using (10) and (30), one can see that

$$d(\tau) = \sum_{i=0}^{m-1} b_i u_{m-i-1}(\tau) = [a_\tau, \cdots, a_0, 0, \cdots, 0][b_0, b_1, \cdots, b_{m-1}]^T = d_\tau. \quad (33)$$

Thus, noting that the contents of register X are initially cleared, i.e., $x_j = 0$, $j \neq 0$, one can find that c in Figure 2(a) outputs c_0 after initialization, i.e.,

$$c(0) = \sum_{j \in \mathcal{T}} x_j(0) + d(0) = 0 + e'_0 + d_0 = c_0.$$

In the following, we show that the output c in Figure 2(a) generates c_τ after the τ-th clock cycle. At this time, the coordinates of register L is changed from the initial value of $L(0) = \langle a_{m-1}, \cdots, a_1 \rangle$ to

$$L(\tau) = \left\langle \underbrace{0, \cdots, 0}_{\tau}, a_{m-1}, \cdots, a_{\tau+1} \right\rangle. \quad (34)$$

Then, using (32) with the new value of L, the output of IP$(m-1)$ generates

$$x_0(\tau) = \left(\sum_{i \in \mathcal{R}} [\underbrace{0, \cdots, 0}_{i+\tau}, a_{m-1}, \cdots, a_{i+\tau+1}] \right) [b_1, \cdots, b_{m-1}]^T,$$

which simplifies to

$$x_0(\tau) = \sum_{i \in \mathcal{R}} e_{i+\tau} = e'_\tau \quad (35)$$

if (11) and (21) are used.

To obtain the output of c after the τ-th clock cycle, i.e., $c(\tau)$, we need to obtain the content of the shift register X, which are found as

$$x_i(\tau) = x_{i-1}(\tau - 1), \quad 1 \le i \le t_{\omega-2}. \quad (36)$$

By recursive using (36), one can find $x_i(\tau) = x_0(\tau - i)$ for $\tau \ge i$, which can be written to

$$x_i(\tau) = \begin{cases} e'_{\tau-i}, & \text{if } \tau \ge i, \\ 0 & \text{otherwise,} \end{cases} \quad (37)$$

if we use (35). Thus, the output of Figure 2(a) after the τ-th clock cycle is $c(\tau) = \sum_{j \in \mathcal{T}} x_j(\tau) + d(\tau)$. Therefore, by using (33), (35), (37) and Lemma 1, one can find $c(\tau) = c_\tau$.

4.3 An Example

We consider the field $GF(2^7)$ defined by the irreducible polynomial $P(x) = x^7 + x^5 + x^3 + x + 1$ for which the reduction matrix can be obtained as

$$\mathbf{Q} = \begin{bmatrix} 1\,1\,0\,1\,0\,1\,0 \\ 0\,1\,1\,0\,1\,0\,1 \\ 1\,1\,1\,0\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,0 \\ 0\,0\,1\,1\,1\,0\,0 \\ 0\,0\,0\,1\,1\,1\,0 \end{bmatrix}. \tag{38}$$

It is seen from the column 0 of (38) that $n = 2$, $r_0 = 0$, and $r_1 = 2$. For this example, $\mathcal{R} = \{0, 2\}$ and $\mathcal{T} = \{0, 1, 3, 5\}$. Table 1 shows how Figure 2(a) generates the coordinates of C at each clock cycle τ.

τ	$v_6,\ v_5,\ v_4,\ v_3,\ v_2,\ v_1$	x_0	x_1, x_2, x_3, x_4, x_5	d	$c = x_0 + x_1 + x_3 + x_5 + d$
0	$a_6, a_5, a_6 + a_4, a_5 + a_3, a_4 + a_2, a_3 + a_1$	e'_0	$0, 0, 0, 0, 0$	d_0	$e'_0 + d_0 = c_0$
1	$0, a_6, a_5, a_6 + a_4, a_5 + a_3, a_4 + a_2$	e'_1	$e'_0, 0, 0, 0, 0$	d_1	$e'_1 + e'_0 + d_1 = c_1$
2	$0, 0, a_6, a_5, a_6 + a_4, a_5 + a_3$	e'_2	$e'_1, e'_0, 0, 0, 0$	d_2	$e'_2 + e'_1 + d_2 = c_2$
3	$0, 0, 0, a_6, a_5, a_6 + a_4$	e'_3	$e'_2, e'_1, e'_0, 0, 0$	d_3	$e'_3 + e'_2 + e'_0 + d_3 = c_3$
4	$0, 0, 0, 0, a_6, a_5$	e'_4	$e'_3, e'_2, e'_1, e'_0, 0$	d_4	$e'_4 + e'_3 + e'_1 + d_4 = c_4$
5	$0, 0, 0, 0, 0, a_6$	e'_5	$e'_4, e'_3, e'_2, e'_1, e'_0$	d_5	$e'_5 + e'_4 + e'_2 + e'_0 + d_5 = c_5$
6	$0, 0, 0, 0, 0, 0$	0	$e'_5, e'_4, e'_3, e'_2, e'_1$	d_6	$e'_5 + e'_3 + e'_1 + d_6 = c_6$

Table 1. The multiplication operation for $GF(2^7)$ generated by $x^7 + x^5 + x^3 + x + 1$.

4.4 Complexity Analysis

In this section, we obtain the space and time complexities of the proposed serial-output bit-serial (SOBS) multiplier.

Proposition 2. *For the finite field $GF(2^m)$ generated by the general irreducible ω-nomial $P(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1$, the SOBS PB multiplier (Figure 2(a)) requires $3m + t_{\omega-2} - 1$ 1-bit register, $2m - 1$ 2-input AND gates, and $(n+1)(m-1) + \omega - 2 - \sum_{i=1}^{n-1} r_i$ 2-input XOR gates.*

Proof. The number of 1-bit registers includes the ones in the L and U shift registers, i.e., $2m - 1$, the register B, i.e., m, and the shift register X, i.e., $t_{\omega-2}$, Thus, the multiplier requires $3m + t_{\omega-2} - 1$ 1-bit registers. The IP(m) and IP($m - 1$) blocks require m and $m - 1$ AND gates, respectively. Therefore, the multiplier requires $2m - 1$ 2-input AND gates. The number of XOR gates is obtained by adding those for the BTX array, the IP(m) and IP($m - 1$) as well as the BTX blocks, which are (28), $m - 1$, $m - 2$, and $\omega - 1$, respectively. As a

result, the number of XOR gates required in the multiplier is $(n-1)(m-1) - \sum_{i=1}^{n-1} r_i. + m - 1 + m - 2 + \omega - 1 = (n+1)(m-1) + \omega - 2 - \sum_{i=1}^{n-1} r_i$ and the proof is complete.

The time complexities of the multiplier are determined by three factors: latency, the number of clock cycles required for whole multiplication, and the critical path delay. Let us define the latency as the number of clock cycles needed that the first bit of the output be available. Based on this definition, one can see that the latency of the SOBS multiplier is one and the entire multiplication requires m clock cycles. The critical path delay, which is the longest path from the registers to the output c, determines the maximum operating frequency. By properly implementation of the BTX block in Figure 2(a), one can minimize this delay to obtain it as follows.

Proposition 3. *Let T_A and T_X be the delay of an AND gate and an XOR gate, respectively. Then, the critical path delay of the SOBS PB multiplier (Figure 2(a)) is at most $T_A + \max(T_1, T_2)$, where $T_1 = (1 + \lceil \log_2(\omega - 1) \rceil + \lceil \log_2 m \rceil) T_X$ and $T_2 = (1 + \lceil \log_2(m-1) \rceil + \lceil \log_2 n \rceil) T_X$.*

Proof. The critical path delay of the multiplier is determined by the maximum delay between the two paths from the shift registers of L and U to the output c. In order to minimize this delay, one can implement c in Figure 2(a) as $c = c' + x_0$, where

$$c' = \sum_{j \in \mathcal{T} - \{0\}} x_j + d. \tag{39}$$

Since the path delay from the shift register U to the output d is $T_A + \lceil \log_2 m \rceil T_X$ and (39) requires $\lceil \log_2(\omega - 1) \rceil T_X$ using a BTX, one can see that the delay to generate c' is at most $T' = T_A + (\lceil \log_2(\omega - 1) \rceil + \lceil \log_2 m \rceil) T_X$. Also, the delay to generate x_0 from the shift register L is $T'' = T_A + (\lceil \log_2(m-1) \rceil + \lceil \log_2 n \rceil) T_X$. Therefore, the total delay to generate c is $T_X + \max(T', T'')$ which is equal to $T_A + \max(T_1, T_2)$ and the proof is complete.

4.5 Comparison

Table 2 shows the comparison of the proposed SOBS PB multiplier with the traditional LSB-first and MSB-first ones presented in Section 2 in terms of time and space complexities for irreducible ω-nominal and trinomial. To illustrate the differences between the complexities of the proposed multiplier with the ones of other multipliers, the complexities for irreducible trinomials are also tabulated in this table. The number of XOR gates γ in this table is obtained for the irreducible trinomial $P(x) = x^m + x^k + 1$, $1 \leq k < \frac{m}{2}$. For the $GF(2^{233})$ field recommended by NIST, one can use $m = 233$, $k = 74$, and $T_3 = T_A + 10T_X$ in this table. As seen from this table, the proposed SOBS multiplier has the lowest latency at the expense of longer critical path and more area requirement.

Multiplier	Latency	Critical path	# AND	# XOR	# 1-bit Register
$P(x) = x^m + \sum_{i=1}^{\omega-2} x^{t_i} + 1,\ 1 \le t_0 < t_1 < \cdots < t_{\omega-2}$					
LSB-first	m	$T_A + T_X$	m	$m + \omega - 2$	$3m$
MSB-first	m	$T_A + T_X$	m	$m + \omega - 2$	$3m$
SOBS	1	$T_A + \max(T_1, T_2)$	$2m - 1$	γ	$3m + t_{\omega-2} - 1$
$P(x) = x^m + x^k + 1,\ 1 \le k < \frac{m}{2}$					
LSB-first	m	$T_A + T_X$	m	$m + 1$	$3m$
MSB-first	m	$T_A + T_X$	m	$m + 1$	$3m$
SOBS	1	T_3	$2m - 1$	$2m + k - 2$	$3m + k - 1$

Table 2. Comparison of multipliers in terms of time and space complexities for irreducible ω-nomial and trinomial, where $\gamma = (n+1)(m-1) + \omega - 2 - \sum_{i=1}^{n-1} r_i$, $T_1 = (1 + \lceil \log_2(\omega - 1) \rceil + \lceil \log_2 m \rceil)\, T_X$, $T_2 = (1 + \lceil \log_2(m-1) \rceil + \lceil \log_2 n \rceil)\, T_X$, and $T_3 = T_A + (2 + \lceil \log_2 m \rceil) T_X$.

5 Conclusions

A new serial-output bit-serial multiplier structure for general irreducible polynomials has been proposed. The proposed multiplier can be used for applications, such as, RFID tags, where the field size and irreducible polynomial are fixed. We have obtained the complexities of the proposed multiplier and compared them with the ones of the LSB-first and the MSB-first multipliers. Unlike the parallel-output multipliers which require m clock cycles for the latency, the proposed serial-output bit-serial multiplier has the latency of one clock cycle. This is achieved at the expense of longer critical path delay and more area requirement.

It is interesting to note that by connecting the output of the proposed multiplier to the serial-input of the LSB-first multiplier, one can obtain a hybrid structure which performs two multiplications together. The results of such a hybrid structure are available in parallel after m clock cycles and it has practical applications for fast cryptographic computations.

The proposed bit-serial multiplier can be extended to obtain a new serial-output digit-serial multiplier by replicating the BTX, IP(m), and IP($m-1$) blocks in Figure 2(a). The latency of such a digit-serial multiplier is one and it generates K bits of the multiplication in each clock cycles with the total $\lceil \frac{m}{K} \rceil$ clock cycles for the entire multiplication.

Acknowledgements

The author would like to thank the anonymous referees of CHES 2008 for their comments. This work has been supported in part by an NSERC Discovery grant awarded to the author.

References

1. G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone. "An Implementation for a Fast Public-Key Cryptosystem". *Journal of Cryptology*, 3:63–79, 1991.
2. E. R. Berlekamp. "Bit-Serial Reed-Solomon Encoders". *IEEE Transactions on Information Theory*, 28(6):869–874, Nov. 1982.
3. T. Beth and D. Gollman. "Algorithm Engineering for Public Key Algorithms". *IEEE J. Selected Areas in Communications*, 7(4):458–465, May 1989.
4. S. S. Erdem, T. Yanik, and C. K. Koç. "Polynomial basis multiplication over $GF(2^m)$". *Acta Applicandae Mathematicae*, 93(1-3):33–55, September 2006.
5. A. Halbutogullari and C. K. Koç. "Mastrovito Multiplier for General Irreducible Polynomials". *IEEE Transactions on Computers*, 49(5):503–518, May 2000.
6. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.
7. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.
8. J. L. Massey and J. K. Omura. "Computational Method and Apparatus for Finite Field Arithmetic". *US Patent No. 4,587,627*, 1986.
9. E. D. Mastrovito. "VLSI Designs for Multiplication over Finite Fields $GF(2^m)$". In *LNCS-357, Proc. AAECC-6*, pages 297–309, Rome, July 1988. Springer-Verlag.
10. E. D. Mastrovito. *VLSI Architectures for Computation in Galois Fields*. PhD thesis, Linkoping Univ., Linkoping Sweden, 1991.
11. V. S. Miller. "Use of Elliptic Curves in Cryptography". In *LNCS 218 as Proceedings of Crypto '85*, pages 417–426. Springer Verlag, 1986.
12. 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(8):945–959, August 2004.
13. F. Rodriguez-Henriquez and C. K. Koç. "Parallel Multipliers Based on Special Irreducible Pentanomials". *IEEE Transactions on Computers*, 52(12):1535–1542, December 2003.
14. P. A. Scott, S. E. Tavares, and L. E. Peppard. "A Fast VLSI Multiplier for $GF(2^m)$". *IEEE J. Selected Areas in Communications*, 4(1):62–66, Jan. 1986.
15. B. Sunar and C. K. Koç. "Mastrovito Multiplier for All Trinomials". *IEEE Transactions on Computers*, 48(5):522–527, May 1999.
16. C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed. "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$". *IEEE Transactions on Computers*, 34(8):709–716, Aug. 1985.
17. H. Wu. "Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis". *IEEE Transactions on Computers*, 51(7):750–758, July 2002.
18. T. Zhang and K. K. Parhi. "Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials". *IEEE Transactions on Computers*, 50(7):734–748, July 2001.