IEICE Electronics Express, Vol.16, No.23, 1–6

LETTER

Novel bit-serial semi-systolic array structure for simultaneously computing field multiplication and squaring

Atef Ibrahim[1,2,3a)]

Abstract This paper presents a novel bit-serial semi-systolic array structure to simultaneously execute modular multiplication and squaring operations in GF(2^m). The architecture is explored by using a systematic methodology based on the proper choice of the scheduling and projection vectors applied to the algorithm dependency graph. The explored architecture has the advantage of sharing the data-path between the two operations, and hence it leads to saving more space compared to the case of using a separate data-path for each operation. Also, the simultaneous calculation of both operations significantly decreases the execution time required to perform modular exponentiation operation, as it mainly depends on these two core operations. Complexity analysis indicates that the developed bit-serial semi-systolic array structure outperforms the latest exiting competitor bit-serial systolic and non-systolic structures in terms of area-time (AT) by at least 24%. This makes the proposed structure more appropriate for use in resource-constrained cryptographic processors.

Keywords: systolic arrays, cryptoprocessor, field exponentiation, field multiplication, field squaring, parallel computing
Classification: Integrated circuits

1. Introduction and related work

Modular multiplication and squaring operations are at the heart of modular exponentiation. Thus, the performance of the modular exponentiation operation is mainly affected by the performance of these two operations. There are various hardware structures, in GF(2^m), developed to increase the performance of these crucial operations [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Unluckily, these hardware implementations mainly concentrate on increasing performance of multiplication operation and do not support the unified structures used to simultaneously compute both operations. Thus, they limit the use of modular exponentiation in several resource-constrained cryptographic and error-correcting codes applications due to their considerable area and time overhead.

There are several unified systolic and semi-systolic array structures presented in the literature to concurrently execute both modular multiplication and squaring operations in GF(2^m). Choi *et al.* [13] presented an approach for merging both operations in a combined systolic array structure. The proposed approach has the merit of reducing the space overhead of the systolic array as well as improving its utilization. They developed bit-serial and bit-parallel systolic array structures based on the proposed approach. Kim *et al.* [14], presented a bit-parallel systolic array structure based on the unified algorithm reported in [15]. This algorithm is based on the bipartite method discussed in [16]. In this method, the operand multiplier is divided into two parts that can be executed in parallel leading to a significant reduction in algorithm latency. Also, Kim *et al.* [17] presented a unified bit-parallel semi-systolic array structure to concurrently execute modular multiplication and squaring in GF(2^m). The developed structure is based on the Montgomery multiplication algorithm.

On the other hand, there are conventional (Non-systolic) architectures used to separately perform both operations based on the Mastrovito multiplier algorithm. The efficient serial architectures that are suitable for the targeted resource-constrained applications are the architectures of [18, 19]. These architectures are extracted based on the irreducible ω-nomial (polynomials with ω non-zero terms) and trinomials and have lower critical pass delay compared to the previously reported results.

In this paper, we present a novel bit-serial semi-systolic array structure to concurrently execute multiplication and squaring in GF(2^m) based on the bipartite multiplication-squaring algorithm reported in [15]. The structure is explored by using a systematic methodology consists of the following three steps: 1) extracting the algorithm dependency graph (DG); 2) assigning time values to each node of the DG based on a chosen scheduling vector; 3) Proper projection of several nodes of the DG to a specific processing element (PE) cell based on a chosen projection vector. The developed bit-serial semi-systolic array structure has lower area and AT complexities compared to the existing most recent bit-serial systolic and non-systolic structures of [13, 18, 19]. This enables the use of the proposed bit-serial array structure in different resource-constrained cryptographic and error-correcting code applications.

The paper is arranged as follows: Section 2 briefly explains the adopted bipartite multiplication-squaring algorithm. Section 3 gives the hardware details of the developed bit-serial semi-systolic array. Section 4 provides the complexity analysis of the developed and related bit-serial structures. Section 5 provides the work conclusion.

[1]Depatment of Computer Engineering, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[2]Microelectronics Department, Electronics Research Institute, Cairo, Egypt
[3]ECE Department, University of Victoria, Victoria, BC, Canada
a) attif_ali2002@yahoo.com

DOI: 10.1587/elex.16.20190600
Received September 26, 2019
Accepted October 21, 2019
Publicized November 11, 2019
Copyedited December 10, 2019

2. Bipartite multiplication and squaring algorithm in GF(2^m)

The details of the bipartite multiplication and squaring algorithm in GF(2^m) are stated in [14, 15]. In this section, we only provide a brief discussion about this algorithm to help understand the developed design.

Let $C(x)$ and $D(x)$ represent any two polynomial elements in GF(2^m). Also, let $F(x)$ be the irreducible polynomial used to produce the filed elements of this field. These polynomials can be expressed as:

$$C(x) = \sum_{j=0}^{m-1} c_j x^j \tag{1}$$

$$D(x) = \sum_{j=0}^{m-1} d_j x^j \tag{2}$$

$$F(x) = \sum_{j=0}^{m} f_j x^j \tag{3}$$

where coefficients $c_j, d_j, f_j \in GF(2)$.

Since x is a root of $F(x)$, $x^m \bmod F(x)$ and $x^{m+1} \bmod F(x)$ can be expressed as follows:

$$x^m \bmod F(x) = \sum_{j=0}^{m-1} f_j x^j \tag{4}$$

$$x^{m+1} \bmod F(x) = \sum_{j=1}^{m-1} (f_{m-1} f_j + f_{j-1}) x^j + f_{m-1} f_0$$

$$\cong F'(x) = \sum_{j=0}^{m-1} f'_j x^j \tag{5}$$

Assume $F'(x)$ is available in advance and Let $l = \lceil m/2 \rceil$, $k = \lfloor m/2 \rfloor$. We can express the modular multiplication and squaring as:

$$\begin{aligned} P(x) &= C(x)D(x) \bmod F(x) \\ &= \sum_{i=0}^{m-1} d_i C(x) x^i \bmod F(x) \\ &= \left(\sum_{i=0}^{l-1} d_{2i} C(x) x^{2i} + x \sum_{i=0}^{k-1} d_{2i+1} C(x) x^{2i} \right) \bmod F(x) \end{aligned} \tag{6}$$

$$\begin{aligned} S(x) &= C(x)C(x) \bmod F(x) \\ &= \sum_{i=0}^{m-1} c_i C(x) x^i \bmod F(x) \\ &= \left(\sum_{i=0}^{l-1} c_{2i} C(x) x^{2i} + x \sum_{i=0}^{k-1} c_{2i+1} C(x) x^{2i} \right) \bmod F(x) \end{aligned} \tag{7}$$

We can split $P(x)$ and $S(x)$ into two portions as:

$$P(x) = (H(x) + xG(x)) \bmod F(x) \tag{8}$$

$$S(x) = (V(x) + xU(x)) \bmod F(x) \tag{9}$$

where,

$$H(x) = \sum_{i=0}^{l-1} d_{2i} C(x) x^{2i} \bmod F(x) \tag{10}$$

$$G(x) = \sum_{i=0}^{k-1} d_{2i+1} C(x) x^{2i} \bmod F(x) \tag{11}$$

$$V(x) = \sum_{i=0}^{l-1} c_{2i} C(x) x^{2i} \bmod F(x) \tag{12}$$

$$U(x) = \sum_{i=0}^{k-1} c_{2i+1} C(x) x^{2i} \bmod F(x) \tag{13}$$

The term $C(x)x^{2i} \bmod F(x)$ is common in Eqs. (10), (11), (12), and (13) and can be defined as $C^i(x) = C^{i-1}(x)x^2 \bmod F(x)$, where $C^0(x) = C(x)$ and $0 \le i \le l-1$. Using Eqs. (4) and (5), we can formulate $C^i(x)$ as:

$$\begin{aligned} C^i(x) &= C^{i-1}(x)x^2 \bmod F(x) \\ &= \sum_{j=0}^{m-1} c_j^{i-1} x^{j+2} \bmod F(x) \\ &= \sum_{j=0}^{m-1} (c_{j-2}^{i-1} + c_{m-2}^{i-1} f_j + c_{m-1}^{i-1} f') x^j \end{aligned} \tag{14}$$

where $C^0 = C$, $c_{-2}^{i-1} = c_{-1}^{i-1} = 0$, and $1 \le i \le l-1$.

We can formulate the coefficients of $C^i(x), c_j^i$, as:

$$c_j^i = c_{j-2}^{i-1} + c_{m-2}^{i-1} f_j + c_{m-1}^{i-1} f' \tag{15}$$

where $c_j^0 = c_j$, $c_{-2}^{i-1} = c_{-1}^{i-1} = 0$, and $1 \le i \le l-1$.

We can express $H(x), G(x), V(x), U(x)$ based on (14) as:

$$H(x) = \sum_{i=1}^{l} d_{2(i-1)} C^{i-1}(x) \tag{16}$$

$$G(x) = \sum_{i=1}^{k} d_{2i-1} C^{i-1}(x) \tag{17}$$

$$V(x) = \sum_{i=1}^{l} c_{2(i-1)} C^{i-1}(x) \tag{18}$$

$$U(x) = \sum_{i=1}^{k} c_{2i-1} C^{i-1}(x) \tag{19}$$

The recurrence equations of $H(x), G(x), V(x), U(x)$ can be expressed as:

$$H^i(x) = H^{i-1}(x) + d_{2(i-1)} C^{i-1}(x) \tag{20}$$

$$G^i(x) = G^{i-1}(x) + d_{2i-1} C^{i-1}(x) \tag{21}$$

$$V^i(x) = V^{i-1}(x) + c_{2(i-1)} C^{i-1}(x) \tag{22}$$

$$U^i(x) = U^{i-1}(x) + c_{2i-1} C^{i-1}(x) \tag{23}$$

where $H^0(x) = G^0(x) = V^0(x) = U^0(x) = 0$, $H^i(x) = \sum_{j=0}^{m-1} h_j^i x^j$, $G^i(x) = \sum_{j=0}^{m-1} g_j^i x^j$, $V^i(x) = \sum_{j=0}^{m-1} v_j^i x^j$, $U^i(x) = \sum_{j=0}^{m-1} u_j^i x^j$ are the i^{th} intermediate results.

We can express the coefficients of $H^i(x), G^i(x), V^i(x), U^i(x)$ in the recursive form as:

$$h_j^i = h_j^{i-1} + d_{2(i-1)}c_j^{i-1},\ for\ 1 \le i \le l \quad (24)$$

$$g_j^i = g_j^{i-1} + d_{2i-1}c_j^{i-1},\ for\ 1 \le i \le k \quad (25)$$

$$v_j^i = v_j^{i-1} + c_{2(i-1)}c_j^{i-1},\ for\ 1 \le i \le l \quad (26)$$

$$u_j^i = u_j^{i-1} + c_{2i-1}c_j^{i-1},\ for\ 1 \le i \le k \quad (27)$$

where $h_j^0 = g_j^0 = v_j^0 = u_j^0 = 0$ and $0 \le j \le m-1$. Eqs. (24) to (27) can be computed concurrently as there is no data dependency between them.

$P(x)$ and $S(x)$ can be calculated based on Eqs. (8) and (9) as follows:

$$P(x) = (H^l(x) + xG^k(x)) \bmod F(x)$$
$$= \sum_{j=0}^{m-1}(h_j^l + g_{m-1}^k f_j + g_{j-1}^k)x^j \quad (28)$$

$$S(x) = (V^l(x) + xU^k(x)) \bmod F(x)$$
$$= \sum_{j=0}^{m-1}(v_j^l + u_{m-1}^k f_j + u_{j-1}^k)x^j \quad (29)$$

where $g_{-1}^k = u_{-1}^k = 0$.

Based on Eqs. (28) and (29), we can calculate the coefficients of $P(x), S(x)$ as follows:

$$p_j = h_j^l + g_{m-1}^k f_j + g_{j-1}^k \quad (30)$$

$$s_j = v_j^l + u_{m-1}^k f_j + u_{j-1}^k \quad (31)$$

where $g_{-1}^k = u_{-1}^k = 0$ and $0 \le j \le m-1$.

3. Proposed bit-serial semi-systolic array

We used the recursive equations of (15), (24), (25), (26), (27), (30), and (31) to explore the dependency graph (DG) shown in Fig. 1. Since the equations have two indices i and j, the DG should be represented in the 2-D integer domain. The rows and columns of the DG are indicated by the indices of i and j, respectively. Circles in DG represent the operations performed by the recursive Eqs. (15), (24), (25), (26), (27), (30), and (31). Fig. 1 indicates that the DG is mainly consists of two sections: the upper section and lower section. The upper section represents the first l rows of DG and it calculates the coefficients of C, H, G, V, U based on Eqs. (15), (24), (25), (26), (27), respectively. The lower section represents the last row of the DG and it computes the coefficients of P and S based on equations (30) and (31), respectively. The initial bits c_j^0, h_j^0, g_j^0, v_j^0, u_j^0, c_j^0, f_j and f'_j are the inputs to the nodes of the upper section of the DG. In this section, the partial results of h_j^i, g_j^i, v_j^i and u_j^i alongside the transmitted input bits of f_j and f'_j are represented by the vertical lines. The diagonal red lines in this section indicates the computed partial results of c_j^i. Also, the input bits of $c_{2(i-1)}, c_{2i-1}, d_{2(i-1)}, d_{2i-1}$ alongside the resulted partial bits of $c_{m-2}^{i-1}, c_{m-1}^{i-1}$ are broadcasted horizontally to all nodes of this section. The output bits of $h_j^l, g_j^k, v_j^l, u_j^k$ produced from the upper section alongside the transmitted bits of f_j are fed as inputs to the lower section of the DG to compute the bits of modular multiplication p_j and squaring s_j as shown in Fig. 1.

We used the approach previously reported in [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30] to obtain the scheduling vector $\mathbf{S} = [2\ \ -1]$ and the projection vector $\mathbf{P} = [1\ 0]^T$. These vectors are used to assign different time values to each node in the DG and project several nodes of the DG to a specific PE cell. Fig. 1 also shows the resulted node timing. The adopted timing results in a serial feeding of the inputs and outputs of the DG.

Fig. 2 shows the resulted semi-systolic array structure after applying the projection vector $\mathbf{P} = [1\ 0]^T$ to the DG. It consists of $l+1$ PEs, where $l = \lceil m/2 \rceil$. PEs are classified into three types as follows: PE_i, PE_l, and PE_{l+1}. PE_i represents the first $l-1$ PEs, while PE_l, and PE_{l+1} represent the l^{th} and $(l+1)^{th}$ PEs, perspectively. PE_l is a simplified version of PE_i where there is no need to compute c_j^i in this PE. PE_{l+1} is the last PE used to compute the bits of modular multiplication (p_j) and squaring (s_j) according to Eqs. (30) and (31), respectively. As we notice from Fig. 2, bits of u_j^i, v_j^i, g_j^i, h_j^i, and f_j are pipelined between all the PEs. Bits of c_j^i are pipelined between the first l PEs. Bits of c_{j-1}^i, c_{j-2}^i, f'_j, $0 \le j \le m-1$, are pipelined between only the first $l-1$ PEs. Bits of $c_{2(i-1)}$, $c_{2i-1}, d_{2(i-1)}, d_{2i-1}$ are located at the first l PEs.

Fig. 3 shows the logic details of each PE. The last two bits $c_{m-2}^{i-1}, c_{m-1}^{i-1}$ are held in PE_i at clock cycles $3(i-1)+2$, $1 \le i \le l-1$, using the MUX-Latch combinations shown in Fig. 3(a). Also, the last two bits u_{m-1}^k and g_{m-1}^k are held in PE_{l+1} at clock cycles $3l+2$ using the MUX-Latch combinations shown in Fig. 3(c). Select signal S_{in} is pipelined between the PEs through the D_s latches to control the MUXes to synchronize the latching process.

The following briefly describes the operation of the developed semi-systolic array.

1) At the first clock cycle $t = 1$, the select input of the two MUXes, shown in Fig. 3(a), sets (i.e., $S_{in} = 1$) in PE_1 to pass the last two input bits c_{m-1}^0 and c_{m-2}^0 to be used alongside the input bits $c_0, c_1, d_0, d_1, c_{m-3}^0$, $u_{m-1}^0, v_{m-1}^0, g_{m-1}^0, h_{m-1}^0, f_{m-1}, f'_{m-1}$ to compute the intermediate bits $c_{m-1}^1, u_{m-1}^1, v_{m-1}^1, g_{m-1}^1, h_{m-1}^1$. Notice

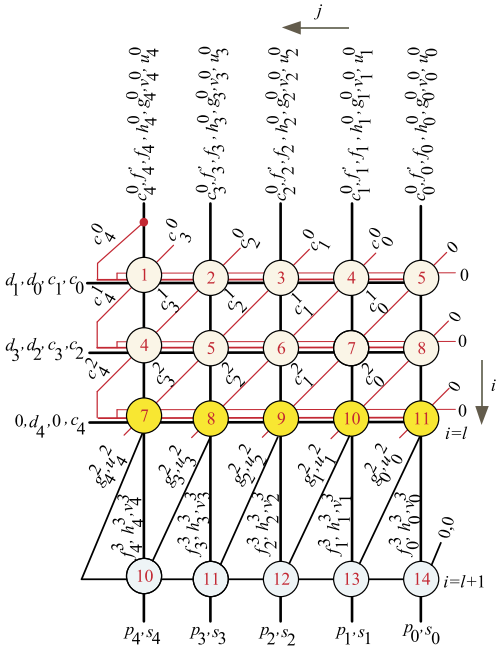

Fig. 1. DG and node timing of the unified multiplication-squaring algorithm for $m = 5$. Each node contains a different time value.

that input signals $c_j^{i-1}, c_{j-1}^{i-1}, c_{j-2}^{i-1}$ shown in PE$_i$, Fig. 3(a), will be assigned the values of c_{m-1}^0, c_{m-2}^0, and c_{m-3}^0 at this clock cycle.

2) At the second clock cycle $t = 2$, the select input of the two MUXes resets (i.e., $S_{in} = 0$) in PE$_1$ to hold the last two bits of c_{m-1}^0 and c_{m-2}^0 to be used alongside input bits c_0, c_1, d_0, d_1, c_{m-3}^0, c_{m-4}^0, u_{m-2}^0, v_{m-2}^0, g_{m-2}^0, h_{m-2}^0, f_{m-2} and f'_{m-2} to update the intermediate bits $c_{m-2}^1, u_{m-2}^1, v_{m-2}^1, g_{m-2}^1, h_{m-2}^1$.
3) Through clock cycles $3 \le t \le m$, the remaining input bits of PE$_1$, c_j^0, c_{j-2}^0, u_j^0, v_j^0, g_j^0, h_j^0, f_j and f'_j, $0 \le j \le m-3$, besides the located bits c_0, c_1, d_0, d_1 and kept bits c_{m-1}^0, c_{m-2}^0 are used to compute the intermediate bits $c_j^1, u_j^1, v_j^1, g_j^1, h_j^1$, $0 \le j \le m-3$.
4) At clock cycles $t = 3(i-1)+1$, $2 \le i \le k$, PE$_i$ starts running precisely like PE$_1$ and sets the MUXes (i.e., $S_{in} = 1$) to pass the last two input bits c_{m-1}^{i-1} and c_{m-2}^{i-1} to be used alongside bits $c_{2(i-1)}, c_{2i-1}, d_{2(i-1)}, d_{2i-1}$, $c_{m-3}^{i-1}, u_{m-1}^{i-1}, v_{m-1}^{i-1}, g_{m-1}^{i-1}, h_{m-1}^{i-1}, f_{m-1}, f'_{m-1}$ to compute the intermediate bits $c_{m-1}^i, u_{m-1}^i, v_{m-1}^i, g_{m-1}^i, h_{m-1}^i$.
5) At clock cycles $t = 3(i-1)+2$, $2 \le i \le k$, the select input of the two MUXes resets (i.e., $S_{in} = 0$) in PE$_i$ to hold the last two bits of c_{m-1}^{i-1} and c_{m-2}^{i-1} to be used alongside input bits $c_{2(i-1)}$, c_{2i-1}, $d_{2(i-1)}$, d_{2i-1}, c_{m-3}^{i-1}, c_{m-4}^{i-1}, u_{m-2}^{i-1}, v_{m-2}^{i-1}, g_{m-2}^{i-1}, h_{m-2}^{i-1}, f_{m-2} and f'_{m-2} to update the intermediate bits $c_{m-2}^i, u_{m-2}^i, v_{m-2}^i, g_{m-2}^i$, h_{m-2}^i.
6) Through clock cycles $t \le 3(i-1)+(m-j)$, $2 \le i \le k$ and $0 \le j \le m-3$, the remaining input bits of PE$_i$, c_j^{i-1}, c_{j-2}^{i-1}, u_j^{i-1}, v_j^{i-1}, g_j^{i-1}, h_j^{i-1}, f_j and f'_j, $2 \le i \le k$ and $0 \le j \le m-3$, besides the located bits $c_{2(i-1)}, c_{2i-1}, d_{2(i-1)}, d_{2i-1}$ and kept bits $c_{m-1}^{i-1}, c_{m-2}^{i-1}$ are used to compute the intermediate bits c_j^i, u_j^i, v_j^i, g_j^i, h_j^i, $2 \le i \le k$ and $0 \le j \le m-3$.
7) Through clock cycles $t \ge 3(l-1)+(m-j)$, $0 \le j \le m-1$, PE$_l$ runs exactly like PE$_i$ to update the intermediate bits, $u_j^k, v_j^l, g_j^k, h_j^l$, $0 \le j \le m-1$. Notice that there is no need to update c_j^l signal in PE$_l$. Also notice that when $l \ne k$ (this means that l is odd) the located bits of c_{2i-1} and d_{2i-1} will be assigned zero values resulting in the updated values of u_j^l and g_j^l remains the same as the previous values of u_j^k and g_j^k resulted from PE$_k$ (i.e., $u_j^l = u_j^k$ and $g_j^l = g_j^k$). On the other hand, when $l = k$ (this means that l is even) all the located bits $c_{2(l-1)}, c_{2l-1}, d_{2(l-1)}, d_{2l-1}$ will be assigned zero values leading to $u_j^l = u_j^k$, $v_j^l = v_j^k$, $g_j^l = g_j^k$, $h_j^l = h_j^k$, $0 \le j \le m-1$.
8) Through clock cycles $t \ge 3(l)+(m-j)$, $0 \le j \le m-1$, PE$_{l+1}$ runs to compute serially the bits of modular multiplication p_j and squaring s_j. It is worth noting that the control signal S_{in} is set ($S_{in} = 1$) at clock cycle $t = 3(l)+1$ to pass the last bits u_{m-1}^k and g_{m-1}^k through the MUXes of PE$_{l+1}$ and it resets ($S_{in} = 0$) at the beginning of clock cycle $t = 3(l)+2$ to hold these bits inside PE$_{l+1}$ to be used through the remaining cock cycles. Also, it is worth noting that some D-latches are added before the XOR gate inputs in PE_i and PE_{l+1} to decrease the critical path delay (CPD) and hence increasing the clock frequency. This leads to obtaining the final result after $3l+m+1$ clock cycles instead of $3l+m$ clock cycles.

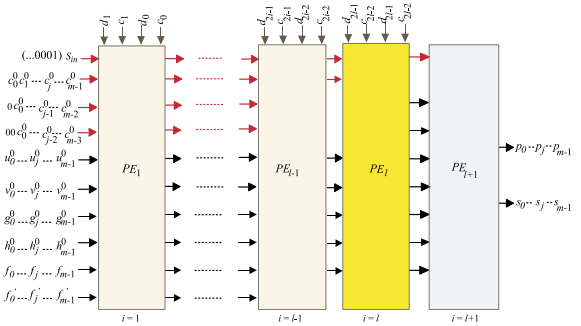

Fig. 2. Proposed serial semi-systolic array structure.

4. Complexity analysis

We used NanGate (15 nm, 0.8 V) Open Cell Library to estimate the area (A) and delay (T) of the basic cells (2-input AND gate, 2-input XOR gate, 2-to-1 MUX, and D-latch) in terms of 2-input NAND gate. Based on the estimated values of the basic cells, we evaluated the area and time complexities of the proposed and compared efficient serial structures of [13, 18, 19] as shown in Tables I and II, respectively. Choi design [13] is a systolic serial structure while Masoleh designs [18, 19] are non-systolic serial structures. Masoleh designs are extracted based on the irreducible ω-nomials (irreducible polynomials with ω non-zero terms) and trinomials. The trinomial-based designs have better performance and thus they are selected here for comparison. The estimated area and delay of the basic cells are as follows: $A_{AND} = 1.2$, $T_{AND} =$

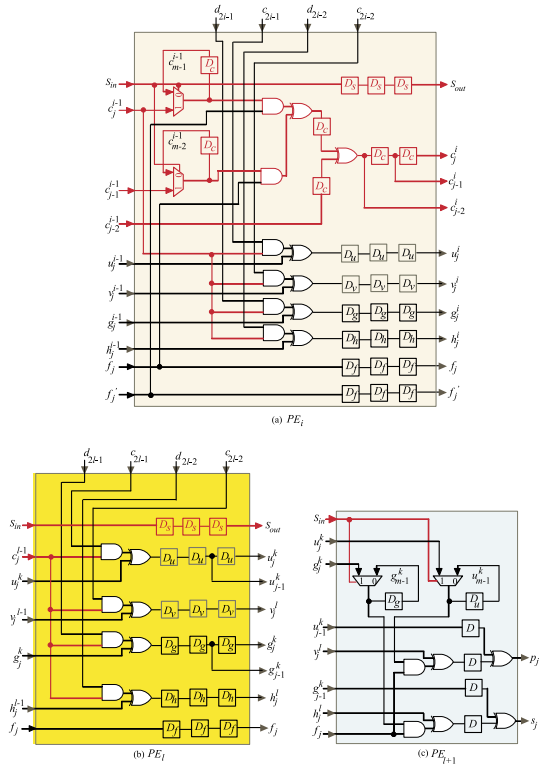

Fig. 3. Logic details of PEs. (a) PE$_i$. (b) PE$_l$. (c) PE$_{l+1}$. The square boxes inside PEs represent D-latches.

Table I. Area of various serial structures in terms of m.

Design	AND	XOR	MUX	Latch	TGC
Choi [13]	$3m$	$3m$	$3m$	$14m$	$78.8m$
Masoleh [18]	$H1^{(1)}$	$H2^{(1)}$	0	$H3^{(1)}$	$H4^{(1)}$
Masoleh [19]	$H5^{(2)}$	$H6^{(2)}$	0	$H7^{(2)}$	$H8^{(2)}$
Proposed	$6l^{(3)}$	$6l+2$	$2l$	$27l-3$	$H9^{(4)}$

(1) $H1 = 5m + 3$, $H2 = 5m + 2t - 4$, $H3 = 19m + 4t - 4$, $H4 = 176.4m + 8.1t - 12.1$, where t is the power of the second trinomial term, $(x^m + x^t + 1)$.
(2) $H5 = 7m + 6$, $H6 = 7m + 6$, $H7 = 18m - 2t - 2$, $H8 = 179.1m - 5.6t - 16.7$
(3) $l = \lceil m/2 \rceil$.
(4) $H9 = 71.7m - 7.9$.

Table II. Delay of various serial structures in terms of m.

Design	Latency	CPD	TD	AT	Th.
Choi [13]	$3m-1$	$\tau_2^{(4)}$	$TD1^{(2)}$	$AT1^{(1)}$	$\frac{1}{m}$
Masoleh [18]	m	$\tau_1^{(3)}$	$TD2^{(2)}$	$AT2^{(1)}$	$\frac{1}{m}$
Masoleh [19]	m	$\tau_3^{(5)}$	$TD3^{(2)}$	$AT3^{(1)}$	$\frac{1}{m}$
Proposed	$3l+m+1$	$\tau_2^{(4)}$	$TD4^{(2)}$	$AT4^{(1)}$	$\frac{1}{m}$

(1) $AT1 = 8{,}605m^2 - 2{,}868m$, $AT2 = (6{,}473.9 + 2{,}240.3\lceil \log_2(m) \rceil)m^2 + (297.3 + 102\lceil \log_2(m) \rceil)mt - (444.1 + 153.7\lceil \log_2(m) \rceil)m$, $AT3 = (6{,}573 + 2{,}274.6\lceil \log_2(m) \rceil)m^2 - (205.5 + 71.1\lceil \log_2(m) \rceil)mt - (612.9 + 212.1\lceil \log_2(m) \rceil)m$, $AT4 = 6{,}520m^2 - 2{,}464m - 287.56$
(2) $TD1 = 109.2m - 36.4$, $TD2 = 12.7m\lceil \log_2(m) \rceil + 36.7m$, $TD3 = 12.7m\lceil \log_2(m) \rceil + 11.3m$, $TD4 = 91m + 36.4$
(3) $\tau_1 = T_{AND} + (2 + \lceil \log_2(m) \rceil)T_{XOR}$
(4) $\tau_2 = T_{AND} + T_{XOR} + T_{MUX}$
(5) $\tau_3 = T_{AND} + \lceil \log_2(m) \rceil T_{XOR}$

Table III. Area and time values of serial structures for $m = 233$ and $t = 74$.

Design	A [Kgates]	T [ns]	AT	%AT
Choi [13]	18	25	467	24
Masoleh [18]	42	32	1344	73
Masoleh [19]	41	26	1066	67
Proposed	16	21	357	-

11.3 ps, $A_{XOR} = 2.5$, $T_{XOR} = 12.7$ ps, $A_{MUX} = 2.5$, $T_{MUX} = 12.4$ ps, $A_{Latch} = 2.8$, $T_{Latch} = 16.6$ ps.

In Table I, we estimated the total gate count (TGC) of each array structure in terms of the field size m based on the area values of the basic cells. In Table II, we multiplied the estimated latency of each design by the corresponding critical path delay (CPD) to obtain the total delay (TD). The Area-Time (AT) complexity of each array structure is also given in this table and it is estimated by multiplying TGC of each array structure with the corresponding TD. By examining the expressions given in Tables I, we can deduce that the total area (TGC) of the basic cells of the proposed serial structure, $H9 = 71.7m - 7.9$, is lower than that of the other serial structures. Also, the expressions given in Table II show that the total delay of the proposed serial structure, $TD4 = 91m + 36.4$, is lower than that of the other serial structures. Moreover, the AT complexity of the proposed serial structure, $AT4 = 6{,}520m^2 - 2{,}464m - 287.56$, is significantly lower than that of the other serial structures by at least 24% as shown in Table III. The evaluated throughput of the compared structures is given in Table II and the given results show that all structures have same throughput. Based on the analytical analysis given in Tables I and II, we quantified the amount of area (A), total computation Time (T), and area-time (AT) for $m = 233$ and $t = 74$ as shown in Table III. The attained results indicate that the developed bit-serial array structure outperforms the compared ones in AT by at least 24%. This makes it more suitable for use in resource-constrained cryptographic processors.

5. Summary and conclusion

In this paper, we developed a unified bit-serial semi-systolic array structure that simultaneously computes both modular multiplication and squaring operations in GF(2^m). The shared data-path between the two operations acquires the advantage of reducing the area overhead compared to using a separate data path for each operation. Also, the concurrent computation of both operations leads to significantly increasing the execution speed of the modular exponentiation operation. The obtained results display that the developed bit-serial array structure outperforms the most recent exiting competitor serial structures in terms of area-time (AT). This makes it more suitable for use in resource-constrained cryptographic processors.

Acknowledgments

The author would like to acknowledge the support of the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University under the research project # 2019/01/10761.

References

[1] C. W. Chiou, *et al.*: "Concurrent error detection in montgomery multiplication over gf(2^m)," IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E89-A** (2006) 566 (DOI: 10.1093/ietfec/e89-a.2.566).
[2] W. T. Huang, *et al.*: "Concurrent error detection and correction in a polynomial basis multiplier over gf(2^m)," IET Inf. Secur. **4** (2010) 111 (DOI: 10.1049/iet-ifs.2009.0160).
[3] K.-W. Kim and J.-C. Jeon: "Polynomial basis multiplier using cellular systolic architecture," IETE J. Res. **60** (2014) 194 (DOI: 10.1080/03772063.2014.914699).
[4] K. W. Kim and J. C. Jeon: "A semi-systolic montgomery multiplier over gf(2^m)," IEICE Electron. Express **12** (2015) 20150769 (DOI: 10.1587/elex.12.20150769).
[5] K. J. Lee and K. Y. Yoo: "Linear systolic multiplier/squarer for fast exponentiation," Inf. Process. Lett. **76** (2000) 105 (DOI: 10.1016/S0020-0190(00)00131-9).
[6] J.-C. Ha and S.-J. Moon: "A common-multiplicand method to the montgomery algorithm for speeding up exponentiation," Inf. Process. Lett. **66** (1998) 105 (DOI: 10.1016/S0020-0190(98)00031-3).
[7] Y. Y. Hua, *et al.*: "Low space complexity digit-serial dual basis systolic multiplier over gf(2^m) using hankel matrix and karatsuba algorithm," IET Inf. Secur. **7** (2013) 75 (DOI: 10.1049/iet-ifs.2012.0227).
[8] C. C. Chen, *et al.*: "Scalable and systolic Montgomery multipliers over GF(2^m)," IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **E91-A** (2008) 1763 (DOI: 10.1093/ietfec/e91-a.7.1763).
[9] S. Kumar, *et al.*: "Optimum digit serial multipliers for curve-based

cryptography," IEEE Trans. Comput. **55** (2006) 1306 (DOI: 10.1109/TC.2006.165).
[10] C. H. Kim, *et al.*: "A digit-serial multiplier for finite field GF(2^m)," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **13** (2005) 476 (DOI: 10.1109/TVLSI.2004.842923).
[11] S. Bayat-Sarmadi, *et al.*: "Dual basis super-serial multipliers for secure applications and lightweight cryptographic architectures," IEEE Trans. Circuits Syst. II, Exp. Briefs **61** (2014) 125 (DOI: 10.1109/TCSII.2013.2291075).
[12] C.-Y. Lee, *et al.*: "New digit-serial three-operand multiplier over binary extension fields for high-performance applications," 2nd IEEE International Conference on Computational Intelligence and Applications (ICCIA) (2017) 17415249 (DOI: 10.1109/CIAPP.2017.8167267).
[13] S. Choi and K. Lee: "Efficient systolic modular multiplier/squarer for fast exponentiation over gf(2^m)," IEICE Electron. Express **12** (2015) 20150222 (DOI: 10.1587/elex.12.20150222).
[14] K. W. Kim and S. H. Kim: "Efficient bit-parallel systolic architecture for multiplication and squaring over gf(2^m)," IEICE Electron. Express **15** (2018) 20171195 (DOI: 10.1587/elex.14.20171195).
[15] K.-W. Kim, *et al.*: "Efficient combined algorithm for multiplication and squaring for fast exponentiation over finite fields gf(2^m)," 7th International Conference on Emerging Databases (EDB) (2017) 50 (DOI: 10.1007/978-981-10-6520-0_6).
[16] E. M. Kaihara and N. Takagi: "Bipartite modular multiplication method," IEEE Trans. Comput. **57** (2008) 157 (DOI: 10.1109/TC.2007.70793).
[17] K. W. Kim and J. D. Lee: "Efficient unified semi-systolic arrays for multiplication and squaring over gf(2^m)," IEICE Electron. Express **14** (2017) 20170458 (DOI: 10.1587/elex.14.20170458).
[18] A. Reyhani-Masoleh: "A new bit-serial architecture for field multiplication using polynomial bases," 7th International Workshop Cryptographic Hardware Embedded Systems (2008) 300 (DOI: 10.1007/978-3-540-85053-3_19).
[19] E. A. H. Abdulrahman and A. Reyhani-Masoleh: "High-speed hybrid-double multiplication architectures using new serial-out bit-level Mastrovito multipliers," IEEE Trans. Comput. **65** (2016) 1734 (DOI: 10.1109/TC.2015.2456023).
[20] F. Gebali: *Algorithms and Parallel Computers* (John Wiley, New York, USA, 2011) 364.
[21] A. Ibrahim, *et al.*: "Processor array architectures for scalable radix 4 montgomery modular multiplication algorithm," IEEE Trans. Parallel Distrib. Syst. **22** (2011) 1142 (DOI: 10.1109/TPDS.2010.196).
[22] F. Gebali and A. Ibrahim: "Efficient scalable serial multiplier over gf(2^m) based on trinomial," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 2322 (DOI: 10.1109/TVLSI.2014.2359113).
[23] A. Ibrahim and F. Gebali: "Low power semi-systolic architectures for polynomial-basis multiplication over gf(2^m) using progressive multiplier reduction," J. Signal Process. Syst. **82** (2016) 331 (DOI: 10.1007/s11265-015-1000-x).
[24] A. Ibrahim, *et al.*: "Systolic array architectures for Sunar-Koc optimal normal basis type II multiplier," IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 2090 (DOI: 10.1109/TVLSI.2014.2358196).
[25] A. Ibrahim, *et al.*: "High-performance, low-power architecture for scalable radix 2 montgomery modular multiplication algorithm," Can. J. Electr. Comput. Eng. **34** (2009) 152 (DOI: 10.1109/CJECE.2009.5599422).
[26] A. Ibrahim and F. Gebali: "Scalable and unified digit-serial processor array architecture for multiplication and inversion over $gf(2^m)$," IEEE Trans. Circuits Syst. I, Reg. Papers **64** (2017) 2894 (DOI: 10.1109/TCSI.2017.2691353).
[27] A. Ibrahim, *et al.*: "New systolic array architecture for finite field division," IEICE Electron. Express **15** (2018) 20180255 (DOI: 10.1587/elex.15.20180255).
[28] A. Ibrahim: "Scalable digit-serial processor array architecture for finite field division," Microelectronics J. **85** (2019) 83 (DOI: 10.1016/j.mejo.2019.01.011).
[29] A. Ibrahim, *et al.*: "Unified systolic array architecture for field multiplication and inversion over gf(2^m)," Comput. Electr. Eng. **61** (2017) 104 (DOI: 10.1016/j.compeleceng.2017.06.014).
[30] A. Ibrahim, *et al.*: "New systolic array architecture for finite field inversion," Can. J. Electr. Comput. Eng. **40** (2017) 23 (DOI: 10.1109/CJECE.2016.2638962).