LETTER

Efficient parallel semi-systolic array structure for multiplication and squaring in GF(2^m)

Atef Ibrahim[1,2,3a)], **Usman Tariq**[1], **Tariq Ahmad**[1], **Ahmed Elmogy**[1,4], **Yassine Bouteraa**[1,5], **and Fayez Gebali**[3]

Abstract In this paper, we develop an efficient parallel semi-systolic array structure to concurrently compute multiplication and squaring operations in the binary extension field, GF(2^m), for efficient modular exponentiations. The proposed array is well suited to VLSI implementation that it has a regular structure as well as local communications between its processing elements. The obtained results show that the proposed array structure achieves a significant reduction in area-time (AT) complexity by at least 95.9% over the corresponding existing structures.

Keywords: semi-systolic arrays, modular multiplication, modular squaring, hardware security, parallel processing
Classification: Integrated circuits

1. Introduction and related work

Finite field Modular exponentiation in GF(2^m) is a critical operation in cryptographic and error-correcting codes applications [1, 2]. This operation is mainly performed using a sequence of finite field multiplication and squaring. Thus, field multiplication can be considered the core operation for the computation of modular exponentiation. As a result, several field multiplier structures in GF(2^m) are developed [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] to increase the performance of this crucial operation. Unfortunately, these structures impose a considerable area and time overhead, which restricts them to use in many cryptographic applications, especially the resource-constrained ones.

The authors in [17] proposed a unified algorithm to concurrently perform field multiplication and squaring in GF(2^m) based on the bipartite method. The proposed algorithm is a regular iterative algorithm and enables the parallel implementation of the two operations. The hardware implementation of this algorithm has significantly lower latency over the other hardware implementations to perform both operations. Therefore, we will adopt this algorithm in this research work to implement more efficient parallel semi-systolic hardware structure.

Many systolic and semi-systolic array structures are developed for simultaneously computing field multiplication and squaring in GF(2^m). In [6], authors developed a technique for combining both field multiplication and squaring in a unified systolic array structure. This technique has the advantage of increasing the utilization of the systolic array besides reducing its hardware overhead. In [18], authors developed a unified parallel semi-systolic array structure based on the Montgomery multiplication algorithm in GF(2^m) to concurrently compute field multiplication and squaring. In [19], authors developed a parallel systolic array structure based on the unified algorithm described in [17]. They proved that it has a lower latency and critical path delay over the related systolic structures.

In this paper, we develop an efficient parallel semi-systolic array structure to concurrently compute multiplication and squaring in GF(2^m) based on the bipartite multiplication method described in [17]. Comparing to the most recent related work of [6, 18, 19], the developed array structure has a significant reduction in both area and AT complexities. This recommends the developed array structure for use in various resource-constrained cryptographic applications.

The paper is organized as follows: Section 2 provides a brief discussion about the adopted unified multiplication-squaring algorithm. Section 3 describes the developed parallel semi-systolic array structure. Section 4 compares the area and time complexities of the developed array structure with the related structures. Section 5 concludes this work.

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

In this section, we briefly discuss the unified modular multiplication and squaring algorithm over GF(2^m) as the details of this algorithm are previously given in [17, 19].

Suppose that $F(x)$ be the irreducible polynomial used to generate the finite field over GF(2^m). Also, let $C(x)$ and $D(x)$ be two arbitrary polynomial elements in $GF(2^m)$. The polynomials of $F(x)$, $C(x)$ and $D(x)$ can be represented in the polynomial form as:

$$F(x) = \sum_{j=0}^{m} f_j x^j \qquad (1)$$

[1]College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia
[2]Microelectronics Department, Electronics Research Institute, Cairo, Egypt
[3]ECE Department, University of Victoria, Victoria, BC, Canada
[4]Computers & Control Engineering Department, Tanta University, Egypt
[5]CEM Lab, ENIS & Digital Research Centre of Sfax, University of Sfax, Tunisia
a) attif_ali2002@yahoo.com

DOI: 10.1587/elex.16.20190268
Received April 21, 2019
Accepted May 9, 2019
Publicized May 30, 2019
Copyedited June 25, 2019

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

$$D(x) = \sum_{j=0}^{m-1} d_j x^j \tag{3}$$

where coefficients $f_j, c_j, d_j \in \{0, 1\}$.

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

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

$$\begin{aligned} x^{m+1} \bmod F(x) &= \sum_{j=1}^{m-1} (f_{m-1} f_j + f_{j-1}) x^j + f_{m-1} f_0 \\ &\cong F'(x) = \sum_{j=0}^{m-1} f'_j x^j \end{aligned} \tag{5}$$

Let $k = \lfloor m/2 \rfloor$, $l = \lceil m/2 \rceil$; and assume that $F'(x)$ is available in advance. The modular multiplication and squaring can be given as:

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

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

Equations (6) and (7) show that each of $P(x)$ and $S(x)$ can be separated into two parts and can be expressed as follows:

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

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

where,

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

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

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

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

From Eqs. (10), (11), (12), and (13), we can notice that the common term $C(x)x^{2i} \bmod F(x)$ is required to compute $H(x), G(x), V(x), U(x)$. We define $C^i(x) = C^{i-1}(x)x^2 \bmod F(x)$ where $C^0(x) = C(x)$ and $0 \le i \le l-1$. Then, based on Eqs. (4) and (5), $C^i(x)$ can be defined 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 $1 \le i \le l-1$.

From Eq. (14), The coefficient of $C^i(x)$, c_j^i, can be expressed 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 $1 \le i \le l-1$.

Using (15), we can represent $H(x), G(x), V(x), U(x)$ as follows:

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

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

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

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

We can formulate the recurrence equations of $H(x)$, $G(x), V(x), U(x)$ as follows:

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

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

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

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

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.

The coefficients of $H^i(x), G^i(x), V^i(x), U^i(x)$ can be represented recursively at step i as follows:

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

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

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

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

where $h_j^0 = g_j^0 = v_j^0 = u_j^0 = 0$ and $0 \le j \le m-1$. There is no data dependency between the equations from (24) to (27) and thus they can be executed simultaneously.

We still need to compute $P(x)$ and $S(x)$ to obtain the results of modular multiplication and squaring, recursively. Based on Eqs. (8) and (9), $P(x)$ and $S(x)$ can be computed as follows:

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

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

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

The coefficients of $P(x)$, $S(x)$ can be calculated as follows:

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

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

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

3. Proposed semi-systolic array architecture of the unified algorithm

We applied the methodology previously described by the first and last authors in [20, 21, 22, 23, 24] to extract the proposed semi-systolic array structure. The methodology can be applied in three steps as follows: 1) getting the data dependency graph (DG) for the specified algorithm. 2) allocating a time value to each node in the DG using a specific timing or scheduling function. 3) mapping several nodes of the DG to a processing element (PE) to form the systolic/semi-systolic array [20, 25, 26, 27, 28, 29, 30].

The DG of the unified multiplication and squaring algorithm over $GF(2^m)$ can be extracted from the recursive Eqs. (15), (24), (25), (26), (27), (30), and (31). The extracted DG based on these equations is shown in Fig. 1. The DG is represented in the two-dimensional integer domain $\mathbb{D}$ due to the equations have two indices i and j. The indices i, j indicates rows and columns, respectively. The algorithm operations are represented by the circle nodes. The DG is divided into two parts: the upper part and the lower part. The upper part consists of the upper l rows of the DG (rows with the light red nodes) and it computes the coefficients of C, H, G, V, U according to Eqs. (15), (24), (25), (26), (27), respectively. The lower part consists of the last row of the DG (row with the brown nodes) and it computes the coefficients of P and S according to Eqs. (30), and (31), respectively. In the upper part of the DG, the calculated intermediate results of h_j^i, g_j^i, v_j^i and u_j^i besides the broadcasted inputs of f_j and f'_j are indicated by the vertical lines. The calculated intermediate results of c_j^i are indicated by the red lines. The input bits $c_{2(i-1)}$, c_{2i-1}, $d_{2(i-1)}$, d_{2i-1}, along with the computed partial bits c_{m-2}^{i-1}, c_{m-1}^{i-1} are transmitted horizontally to all nodes in the upper part of the DG. The initial input 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 fed to the nodes at the top of the DG.

The outputs resulted from the upper part of the DG h_j^l, g_j^k, v_j^l, u_j^k beside the broadcasted bits of f_j are fed as input to the lower part (the last row), as shown in Fig. 1, to compute the output bits of p_j and s_j.

After applying the methodology formerly described in [20, 23, 24, 25, 26], we get the scheduling vector $\mathbf{S} = [1\ 0]$ and the projection vector $\mathbf{P} = [0\ 1]^T$ to assign node timing to the DG and map several nodes of the DG to a specific PE cell, respectively. The resulted node timing of the DG is shown in Fig. 2.

From Fig. 2, we can notice that the initial inputs of h_j^0, g_j^0, v_j^0, u_j^0, c_j^0, and f, f' are fed in parallel at the first time step and the outputs of p_j, s_j are also available in parallel at the last time step $l+1 = \lceil m/2 \rceil + 1$. Fig. 3 displays the produced semi-systolic array architecture resulted from applying the projection vector $\mathbf{P} = [0\ 1]^T$ to the DG. It consists of m (PE$_j$) PE cells and m (PS$_j$) PE cells. PE$_j$ cells compute h_j^l, g_j^k, v_j^l, u_j^k, which are used along with f_j as inputs to the PS$_j$ cells to compute p_j and s_j as shown in Fig. 3.

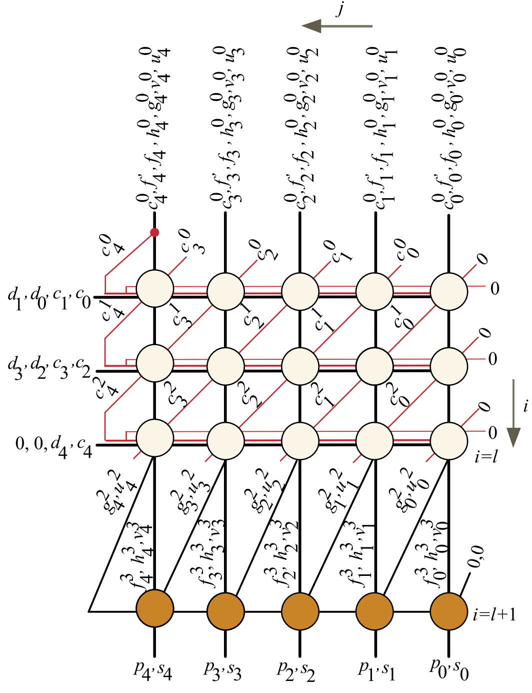
j
i
i=l
i=l+1
d_1, d_0, c_1, c_0
d_3, d_2, c_3, c_2
$0, 0, d_4, c_4$
p_4, s_4
p_3, s_3
p_2, s_2
p_1, s_1
p_0, s_0
$c_4^0, f'_4, f_4, h_4^0, g_4^0, v_4^0, u_4^0$
$c_3^0, f'_3, f_3, h_3^0, g_3^0, v_3^0, u_3^0$
$c_2^0, f'_2, f_2, h_2^0, g_2^0, v_2^0, u_2^0$
$c_1^0, f'_1, f_1, h_1^0, g_1^0, v_1^0, u_1^0$
$c_0^0, f'_0, f_0, h_0^0, g_0^0, v_0^0, u_0^0$
0
0,0

Fig. 1. DG of the unified multiplication-squaring algorithm for $m = 5$

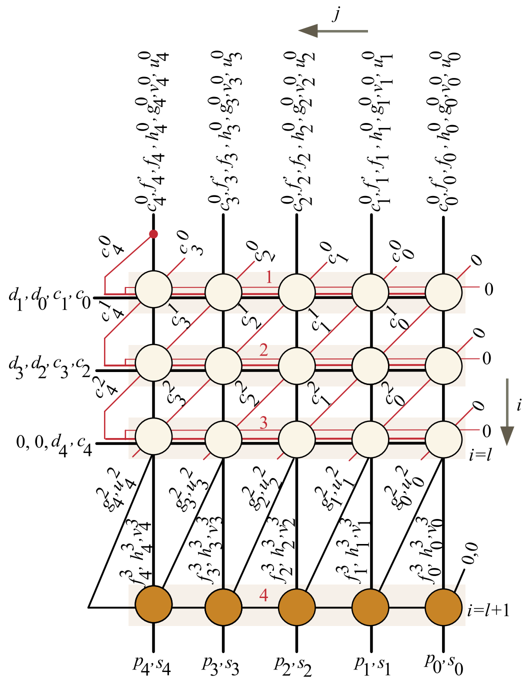

Fig. 2. DG Node Timing. Each row assigned different time value.

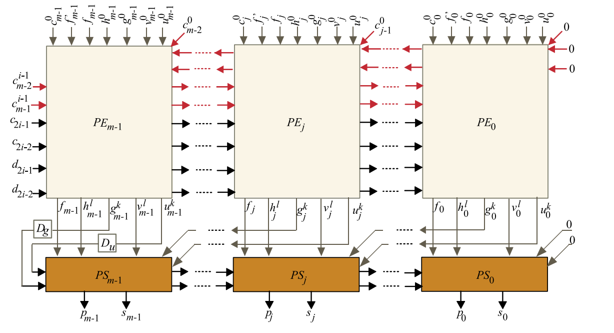

Fig. 3. Semi-systolic array architecture of the unified multiplier-squarer algorithm.

As Fig. 3 displays, input bits f, f', and c_j^0 are allocated to each PE$_j$ cell and the resulted intermediate bits of C, are pipelined between the PE$_j$ cells. Since the inputs of H (h_j^0), G (g_j^0), V (v_j^0), U (u_j^0) are initialized with zero values, they can be created by resetting the corresponding D-Latches. The produced intermediate bits of h_j^i, g_j^i, v_j^i, u_j^i are updated internally within each PE$_j$ cell. Input bits c_{2i-2}, c_{2i-1}, d_{2i-2}, d_{2i-1}, in addition to the resulted intermediate bits c_{m-2}^{i-1} and c_{m-1}^{i-1} (produced from PE$_{m-2}$ and PE$_{m-2}$ cells, respectively) are transferred to all PE$_j$ cells. The output bits of H, h_j^l, and V, v_j^l, resulted from PE$_j$ cells are produced in parallel after $l = \lceil m/2 \rceil$ time steps, while the output bits of G, g_j^k, and U, u_j^k, are produced in parallel after $k = \lfloor m/2 \rfloor$ time steps. The Tri-State buffers shown in Fig. 4 will control this timing process. The output bits of P, p_j, and S, s_j, resulted from PS$_j$ cells are produced in parallel after $l + 1 = \lceil m/2 \rceil + 1$ time steps. Therefore, the total time required to get the final results is equal to $\lceil m/2 \rceil + 1$.

Fig. 4 displays the hardware details of each PE$_j$ cell. To insure that the initial inputs of H, h_j^0, G, g_j^0, V, v_j^0, and U, u_j^0, assigned zero values, the corresponding D-Latches (D_h, D_g, D_v, and D_u), in each PE$_j$ cell, should be cleared before the iterations begin. It is important to notice that signals c_j^{i-1} and c_{j-1}^{i-1} shown in Fig. 4, represents the c_{m-1}^{i-1} and c_{m-2}^{i-1} input in the last PE$_{m-1}$ cell as indicated in Fig. 3.

Fig. 5 displays the hardware details of each PS$_j$ cell. Inputs f_j, h_j^l, v_j^l, g_{j-1}^k, u_{j-1}^k, g_{m-1}^k, u_{m-1}^k are used to compute the multiplication and squaring outputs p_j and s_j, respectively. f_j, h_j^l, v_j^l, g_{j-1}^k, u_{j-1}^k inputs are delayed by one clock cycle through the corresponding D-Latches inside the PS$_j$ cell, while g_{m-1}^k and u_{m-1}^k inputs are previously delayed through the D-Latches (D_g and D_u FFs) indicated in Fig. 3.

The following summarizes the operation of the extracted semi-systolic array architecture.

1) At the first clock cycle $i = 1$, Muxes inside PE$_j$ cells are activated ($M_c = 1$) to transfer the input bits of C, c_{j-1}^0 and c_j^0. Also, input bits c_0, c_1, d_0, d_1 alongside the bits c_{m-2}^0 and c_{m-1}^0 are transferred to all PE$_j$ cells.
2) At clock cycles $1 < i \leq l$, Muxes inside PE$_j$ cells transfer the intermediate bits of c_{j-1}^{i-1} and c_j^{i-1} ($M_c = 0$) as pointed out in Fig. 3. c_{j-1}^{i-1} signal passes to the next PE$_j$ cell and c_j^{i-1} signal also passes to the next cell bedside used inside the current cell to compute the new intermediate values of h_j^i, g_j^i, v_j^i and u_j^i as shown in Fig. 3. Also, input bits c_{2i-2}, c_{2i-1}, d_{2i-2}, and d_{2i-1} alongside the bits c_{m-2}^{i-1}, c_{m-1}^{i-1} are transferred to all PE$_j$ cells, one bit at each clock cycle.
3) At clock cycle $i = k = \lfloor m/2 \rfloor$, The Tri-State buffers T$_g$ and T$_u$ passes the produced values of g_j^k and u_j^k.
4) At clock cycle $i = l = \lceil m/2 \rceil$, The Tri-State buffers T$_h$ and T$_v$ pass the resulted values of h_j^l and v_j^l.
5) At the last clock cycle $i = l + 1 = \lceil m/2 \rceil + 1$, the inputs f_j, h_j^l, v_j^l, g_{j-1}^k, u_{j-1}^k, g_{m-1}^k, u_{m-1}^k of the PS$_j$ cells will be used to compute the output bits of P, p_j, and S, s_j, in parallel as shown in Fig. 3.

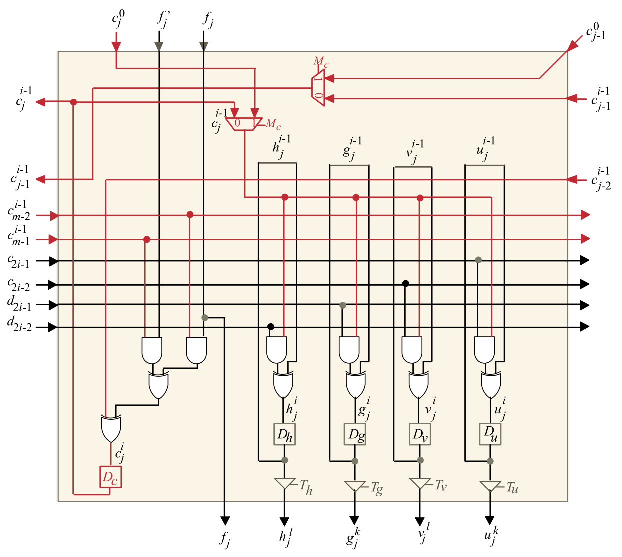

Fig. 4. PE$_j$ cell. D-Latches are indicated with the square boxes.

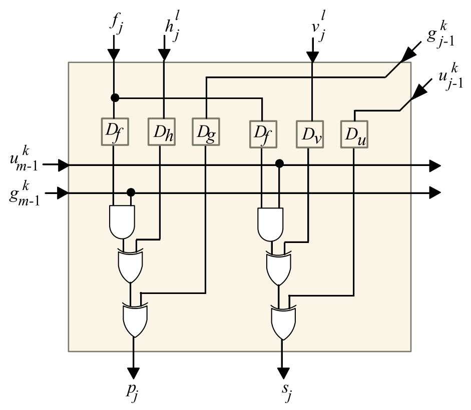

Fig. 5. PS$_j$ cell. D-Latches are indicated with the square boxes.

4. Complexity analysis

We used NanGate (15 nm, 0.8 V) Open Cell Library to get the area and worst-case intrinsic delay of the basic gates, 2-to-1 MUX, and Latch. The worst-case intrinsic delay is attained pertaining to the unit-derive strength of the open cell library. The estimated area and delay for the basic cells are given in Table I. The area of the basic logic cells is given in terms of the 2-input NAND gate. The area and delay complexities of the presented and related most recent parallel systolic/semi-systolic designs [6, 18, 19] are given in Tables II and III. We used the data in Table I to calculate the total gate count (TGC) and total delay (TD) of the developed and compared parallel systolic/semi-systolic array structures in terms of the field size m as shown in Tables II and III, respectively. TD is obtained from the product of latency and the critical path delay (CPD). T_A, T_X, and T_{MUX} in Table III denote the delay of the 2-input AND cell, the 2-input XOR cell, and the 2-to-1 MUX, respectively. Table III also shows the Area-Time (AT) complexity of each array structure. It is calculated for each array by multiplying the corresponding TGC and TD.

Table I. Area and delay of basic cells in terms of 2-input NAND gate.

	INV	TSB*	AND	XOR	MUX	Latch
Area (2-input NAND)	0.6	0.8	1.2	2.5	2.5	2.8
Delay (ps)	5.8	7.9	11.3	12.7	12.4	16.6

(∗) TSB represents the Tri-State buffer.

Table II. Area of various parallel systolic/semi-systolic array structures in terms of the field size m.

Design	TSB	AND	XOR	MUX	Latch	TGC
Choi [6]	0	$3m^2$	$3m^2$	0	$10m^2$	$104.1m^2$
Kim [18]	0	$F1^{(*)}$	$F2^{(*)}$	0	$F3^{(*)}$	$F4^{(*)}$
Kim [19]	0	$F5^{(*)}$	$F6^{(*)}$	0	$F7^{(*)}$	$F8^{(*)}$
Proposed	$4m$	$8m$	$10m$	$2m$	$11m$	$90.1m$

(∗) $F1 = 1.5m^2 + 1.5m$, $F2 = 1.5m^2 + 3.5m$, $F3 = 2m^2 + 4m$, $F4 = 14.15m^2 + 27.75m$, $F5 = 3m^2 + 2m$, $F6 = 3m^2 + 4m$, $F7 = 9m^2 + 3m$, $F8 = 49.8m^2 + 25.9m$

From Table II, we notice that the compared array structures have area complexity (TGC) of $\mathcal{O}(m^2)$, while the proposed array structure has area complexity of $\mathcal{O}(m)$. From Table III, we notice that the compared array structures have AT complexity of $\mathcal{O}(m^3)$, while the proposed array structure has AT complexity of $\mathcal{O}(m^2)$. This means that the proposed array structure has a significant reduction in area and AT complexities over the compared array structures.

Based on the analytical results gained in Tables II and III, we can quantify the amount of area (A), computation time (T), and AT for $m = 233$ and $m = 409$ as shown in Table IV. The attained results indicate that the proposed array structure has AT improvement over the compared ones by at least 95.9%.

5. Summary and conclusion

This study proposes an efficient parallel semi-systolic array structure to concurrently compute the unified multiplication and squaring algorithm in GF(2^m) for efficient modular exponentiations. The developed array structure has a regular structure and local communications between processing elements that make it more suited to VLSI implementation. The obtained results indicate that the proposed array structure has a significant improvement in the area and AT complexity over the most recent related work. This nominates the proposed design for use in various resource-constrained applications for security purposes.

Table III. Delay of various parallel systolic/semi-systolic array structures in terms of the field size m.

Design	Latency	CPD	TD	AT
Choi [6]	$3m$	$T_A + T_X$	$72m$	$7495.2m^3$
Kim [18]	$0.5m + 2.5$	$T_A + T_X$	$12m + 60$	$AT1^{(*)}$
Kim [19]	$1.5m + 1$	$T_A + 2T_X$	$TD1^{(**)}$	$AT2^{(*)}$
Proposed	$\lceil m/2 \rceil + 1$	$T_A + 2T_X$	$TD2^{(**)}$	$AT3^{(*)}$

(∗) $AT1 = 170m^3 + 1182m^2 + 1665m$, $AT2 = 2765m^3 + 3266m^2 + 951m$, $AT3 = 3306m(\lceil m/2 \rceil + 1)$
(∗∗) $TD1 = 55.53m + 36.7$, $TD2 = 36.7(\lceil m/2 \rceil + 1)$

Table IV. Calculated area and time of various systolic/semi-systolic arrays for $m = 233$ and $m = 409$.

Design	m	A [Kgates]	T [ns]	AT	% AT
Choi [6]	233	5,651.48	16.78	94,809.31	99.9
	409	17,413.95	29.45	512,806.06	99.9
Kim [18]	233	774.66	2.86	2,212.41	95.9
	409	2,378.38	4.97	11,815.77	97.6
Kim [19]	233	2,709.63	12.98	35,157.92	99.7
	409	8341.19	22.75	189,749.24	99.8
Proposed	233	20.9	4.3	90.9	-
	409	36.85	7.56	278.60	-

Acknowledgments

The authors 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] R. E. Blahut, *et al.*: *Handbook of Applied Cryptography* (CRC Press, Boca Raton, FL, 1996) 816.
[2] R. E. Blahut: *Theory and Practice of Error Control Codes* (Addison-Wesley, Reading, MA, 1983) 500.
[3] C. W. Chiou, *et al.*: "Concurrent error detection in Montgomery multiplication over gf(2^m)," IEICE Trans. Fundamentals **E89-A** (2006) 566 (DOI: 10.1093/ietfec/e89-a.2.566).
[4] 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).
[5] 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).
[6] 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).
[7] 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).
[8] P. A. Scott, *et al.*: "Architectures for exponentiation in gf(2^m)," IEEE J. Sel. Areas Commun. **6** (1988) 578 (DOI: 10.1109/49.1927).
[9] 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).
[10] 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).

[11] 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).

[12] C. C. Chen, *et al.*: "Scalable and systolic Montgomery multipliers over GF(2^m)," IEICE Trans. Fundamentals **E91-A** (2008) 1763 (DOI: 10.1093/ietfec/e91-a.7.1763).

[13] S. Kumar, *et al.*: "Optimum digit serial multipliers for curve-based cryptography," IEEE Trans. Comput. **55** (2006) 1306 (DOI: 10.1109/TC.2006.165).

[14] 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).

[15] 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).

[16] 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).

[17] 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).

[18] 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).

[19] 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).

[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," Microelectron. 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).