Compact Bit-Parallel Systolic Multiplier Over GF(2^m) Mulplicateur systolique compact à bit-parallèle sur GF(2^m)

Atef Ibrahim, *Member, IEEE*, Fayez Gebali, *Life Senior Member, IEEE*, Yassine Bouteraa, Usman Tariq, Tariq Ahanger, *Member, IEEE*, and Khaled Alnowaiser

***Abstract*—This article presents a compact and efficient bit-parallel systolic array structure for multiplication over the extended binary field, GF(2^m). The systolic array has a regular arrangement with local connections, making it more suitable for VLSI implementations. Also, it has the merits of having hardware complexity of order $\mathcal{O}(m)$ that distinguishes it from the previously reported bit-parallel designs having hardware complexity of order $\mathcal{O}(m^2)$. The achieved results exhibited that the suggested parallel architecture realizes a significant reduction in hardware complexity and the area-delay complexity over the competitor architectures previously published in the literature. Therefore, it is more suitable for usage in constrained hardware environments, having more restrictions on space, such as portable devices and smart cards.**

***Résumé*—Cet article présente une structure de tableau systolique à bit-parallèle compacte et efficace pour la mulplication sur le champ binaire étendu, GF (2^m). Le tableau systolique a une disposition régulière avec des connexions locales, ce qui le rend plus approprié pour les implémentations VLSI. De plus, il a le mérite d'avoir une complexité matérielle d'ordre $\mathcal{O}(m)$ qui le distingue des précédentes conceptions à bit-parallèle rapportées ayant une complexité matérielle d'ordre $\mathcal{O}(m^2)$. Les résultats obtenus ont montré que l'architecture parallèle suggérée permet une réduction significative de la complexité matérielle et de la complexité du délai de la zone par rapport aux architectures concurrentes précédemment publiées dans la liérature. Par conséquent, il est plus adapté à une utilisation dans des environnements matériels contraignants, ayant plus de restrictions d'espace, tels que les appareils portables et les cartes à puce.**

***Index Terms*—Cryptography, hardware security, modular multipliers, modular squares, parallel computing, systolic multipliers.**

I. Introduction

CRYPTOGRAPHIC algorithms provide security to the transmitted data by encrypting and decrypting it at the transmitter and receiver sides, respectively. Most of these algorithms heavily rely on finite-filed arithmetic operations, namely, the finite-field multiplication. The recursive multiplications are extensively used to perform the other field operations of inversion, division, and exponentiation [1]. Therefore, this operation has given more consideration in the literature to implement compact and efficient cryptographic algorithms. There are several bases over which the finite-field multiplication can be computed, precisely, polynomial basis (PB), dual basis (DB), and normal basis (NB) [2]. The hardware implementations of the multipliers based on the PB have less area complexity but consume more power compared with the NB multipliers. The NB and DB multipliers require base conversion, and this increases the hardware overhead of these multipliers. Therefore, PB multipliers are widely used to implement cryptographic algorithms.

A systolic array is an architecture having regular structures, which makes it very suitable for VLSI implementation. It consists of 1-D or 2-D processing elements (PEs) that perform a particular task. Most of the previously reported systolic multiplier structures over GF(2^m) are either bit-parallel [3]–[11] or bit-serial [12]–[14]. Bit-parallel systolic architectures realize very high speed but with considerable area complexity and high-power consumption. On the other hand, the bit-serial systolic structures offer a significant reduction in area

Manuscript received April 26, 2020; revised July 17, 2020 and September 12, 2020; accepted October 28, 2020. Date of current version June 15, 2021. This work was supported by the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University under research Project # 2020/01/16466. *(Corresponding author: Atef Ibrahim.)*

Atef Ibrahim is with the Computer Engineering Department, Prince Sattam Bin Abdulaziz University (PSAU), Al-Kharj 11942, Saudi Arabia, and also with the Department of Electrical and Computer Engineering (ECE), University of Victoria, Victoria, BC V8W 2Y2, Canada (e-mail: aa.mohamed@psau.edu.sa).

Fayez Gebali is with the Department of Electrical and Computer Engineering (ECE), University of Victoria, Victoria, BC V8W 2Y2, Canada (e-mail: fayez@ece.uvic.ca).

Yassine Bouteraa is with the Computer Engineering Department, Prince Sattam Bin Abdulaziz University (PSAU), Al-Kharj 11942, Saudi Arabia, also with the CEM Lab ENIS, University of Sfax, Sfax 3038, Tunisia, and also with the Digital Research Centre of Sfax, University of Sfax, Sfax 3038, Tunisia (e-mail: y.bouteraa@psau.edu.sa).

Usman Tariq and Tariq Ahanger are with the Department of Information Systems, Prince Sattam Bin Abdulaziz University (PSAU), Al-Kharj 11942, Saudi Arabia (e-mail: u.tariq@psau.edu.sa; t.ahanger@psau.edu.sa).

Khaled Alnowaiser is with the Computer Engineering Department, Prince Sattam Bin Abdulaziz University (PSAU), Al-Kharj 11942, Saudi Arabia (e-mail: k.alnowaiser@psau.edu.sa).

Associate Editor managing this article's review: Peng Hu.

Digital Object Identifier 10.1109/ICJECE.2020.3035182

complexity and power consumption but with a substantial decrease in speed.

There are many literature attempts to design hardware-efficient bit-parallel systolic multiplier structures over GF(2^m). Most of these trials are based on using particular types of irreducible polynomials. In 2001, Lee *et al.* [6], [7] proposed a low-complexity bit-parallel systolic multiplier based on all-one polynomials (AOPs) and equally spaced polynomials. In 2005, Lee and Chiou [8] proposed a transformation method that allows converting the AOP-based bit-parallel systolic multiplier of [6] into a trinomial-based low-complexity bit-parallel systolic Montgomery multiplier. In 2008, Lee [9] presented a low-complexity bit-parallel systolic multiplier based on the Montgomery algorithm using the Toeplitz matrix-vector representation. In 2015, Bayat-Sarmadi and Farmani [10] developed a low-complexity with a high throughput trinomial-based bit-parallel systolic Montgomery multiplier. In 2018, Mathe and Boppana [11] proposed a bit-parallel systolic multiplier based on a modified interleaved multiplication algorithm over GF(2^m). The modified algorithm allows reducing the area overhead of the proposed systolic structure.

This article presents a bit-parallel 1-D systolic multiplier with significantly less hardware overhead and comparable speed to most of the previously reported competitor multiplier structures. The systolic structure is built based on the general irreducible polynomial and can be modified to any recommended types of special polynomials, namely, trinomials and penitentials. We explored this systolic structure by developing the dependence graph (DG) of the interleaved multiplication algorithm and applying the proper scheduling and projection vectors to map each DG node to the corresponding PE. The structure is very regular and has neighbor-to-neighbor connections, making it very suitable for VLSI implementations.

Section II briefly explains the interleaving multiplication algorithm over GF(2^m) besides the extraction of the corresponding DG. Section III discusses the used approach to develop the suggested bit-parallel multiplier structure and its logic details. Section IV provides the results and discussions. Section V summarizes and concludes this article.

II. Finite Field Interleaved Multiplication

Suppose that we have two polynomials $A(y)$ and $B(y)$ defined over GF(2^m) and $F(y)$ is an irreducible polynomial of degree m used to generate this field. We can represent these polynomials as follows:

$$A(y) = \sum_{j=0}^{m-1} a_j y^j = \left(a_0 + a_1 y^1 + \cdots + a_{m-1} y^{m-1}\right) \quad (1)$$

$$B(y) = \sum_{j=0}^{m-1} b_j y^j = \left(b_0 + b_1 y^1 + \cdots + b_{m-1} y^{m-1}\right) \quad (2)$$

$$F(y) = \sum_{j=0}^{m} f_j y^j = \left(f_0 + f_1 y^1 + \cdots + f_{m-1} y^{m-1}\right) \quad (3)$$

where $a_j, b_j, f_j \in$ GF(2).

The finite field multiplication can be performed by multiplying $A(y)$ and $B(y)$ and then reducing the result using $F(y)$ as follows:

$$\begin{aligned} C(y) &= A(y)B(y) \bmod F(y) \\ &= A(y)\left(b_0 + b_1 y^1 + \cdots + b_{m-1} y^{m-1}\right) \bmod F(y). \end{aligned} \quad (4)$$

We can rearrange (4) to have the following interleaved form:

$$\begin{aligned} C(y) = b_0[A(y) \bmod F(y)] + b_1\left[y^1 A(y) \bmod F(y)\right] \\ + \cdots + b_{m-1}\left[y^{m-1} A(y) \bmod F(y)\right]. \end{aligned} \quad (5)$$

For simplicity, we will refer to polynomials $A(y)$, $B(y)$, $C(y)$, and $F(y)$ as A, B, C, and F, respectively. From (5), we notice that the product operation can be performed as the accumulation of the terms $b_i[Ay^i \bmod F]$, where $0 \le i \le m-1$. Let $A^i = Ay^i \bmod F$, and we can represent A^{i+1} in terms of A^i as $A^{i+1} = Ay^{i+1} \bmod F = [Ay^i]y \bmod F = A^i y \bmod F$. Thus, we can recursively obtain A^{i+1} from A^i as follows:

$$\begin{aligned} A^{i+1} &= A^i y \bmod F \\ \sum_{j=0}^{m-1} a_j^{i+1} y^j &= \left(\sum_{j=0}^{m-1} a_j^i y^j\right) y \bmod F \\ &= \sum_{j=0}^{m-1} \left(a_j^i y^{j+1}\right) \bmod F \\ &= a_{m-1}^i \left[y^m \bmod F\right] + \sum_{j=0}^{m-2} a_j^i y^{j+1} \\ &= a_{m-1}^i \sum_{j=0}^{m-1} f_j y^j + \sum_{j=0}^{m-1} a_{j-1}^i y^j \end{aligned} \quad (6)$$

where $A^0 = A$. Notice that the term $y^m \bmod F$ of (6) is equivalent to $\sum_{j=0}^{m-1} f_j y^j$ in GF(2^m) [4]. Also, the term $\sum_{j=0}^{m-2} a_j^i y^{j+1}$ represents a polynomial of degree less than m. Thus, it does not need to be reduced modulo F.

We can express (6) in the bit-level form at step i as follows:

$$a_j^{i+1} = a_{j-1}^i + a_{m-1}^i f_j, \;\; 0 \le j \le m-1 \quad (7)$$

where $a_{-1}^i = 0$ for $0 \le i \le m-1$.

The accumulation of A^i terms to produce the partial product C^{i+1}, $0 \le i \le m-1$, can be performed using the following recursive equation:

$$\begin{aligned} C^{i+1} &= C^i + b_i A^i \\ \sum_{j=0}^{m-1} c_j^{i+1} y^j &= \sum_{j=0}^{m-1} c_j^i y^j + b_i \sum_{j=0}^{m-1} a_j^i y^j \end{aligned} \quad (8)$$

where $C^0 = 0$ and C^m represents the final product C. We can express (8) in the bit-level form at step i as follows:

$$c_j^{i+1} = c_j^i + b_i a_j^i \quad (9)$$

where $0 \le i \le m-1$, $0 \le j \le m-1$, and $c_j^0 = 0$ for $0 \le j \le m-1$.

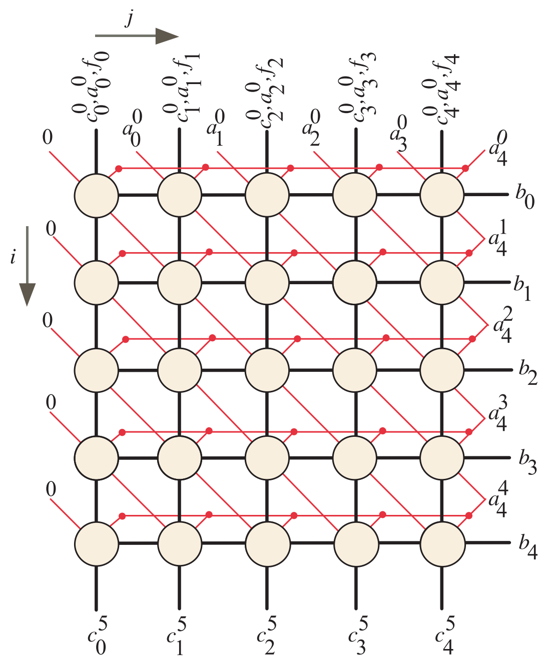


Fig. 1. Detailed DG of the algorithm for $m = 5$.

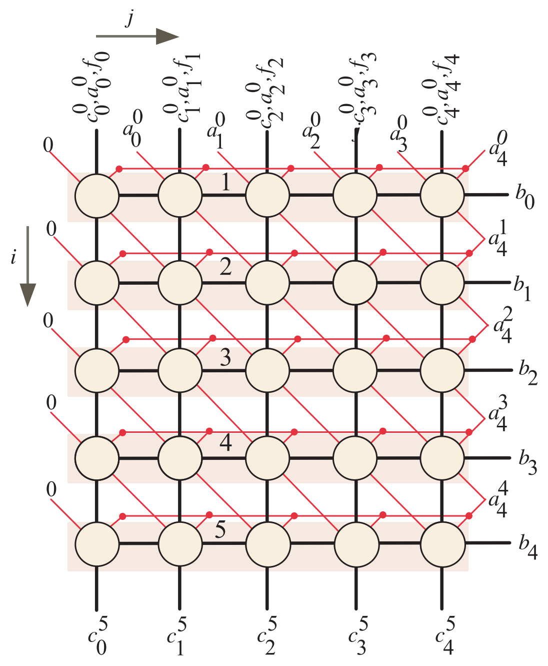


Fig. 2. Node timing for $m = 5$.

A. Dependence Graph

We can extract the DG used to perform the finite-field multiplication from the regular iterative equations (7) and (9). As we notice, the two equations have two indices i and j, which helps us represent the DG in the 2-D integer domain $\mathbb{D}$ with index i designating the rows and index j the columns. Fig. 1 shows the details of the DG for $m = 5$. The nodes (circles) perform the operations represented by (7) and (9). The vertical lines represent the updated signals of c_j^i and a_j^i and the located signal f_j. The horizontal lines represent the broadcasted signals of b_i. The diagonal lines (red lines) represent the updated signals of a_{j-1}^i. Signals a_{j-1}^i and a_j^i are required to compute a_j^{i+1} and c_j^{i+1}, respectively. a_{m-1}^i signal is generated from last column nodes and broadcasted to the remaining nodes in the same row.

As we notice from Fig. 1, the input signals c_j^0, a_j^0, and f_j are at the top of the DG, while the output signals c_j^m come out from the bottom row.

III. Bit-Parallel Systolic Array Exploration

This section briefly describes the used approach to develop the compact bit-parallel systolic array performing the finite-field multiplication. It mainly describes the scheduling and projection functions used to assign each DG node to the corresponding PE [15]–[22].

A. Scheduling Function

We will refer to the nodes of the DG as a point $\mathbf{p}(i, j) = [i \ \ j]$. The methodology uses a scheduling vector $\mathbf{s}(w, z) = [w \ \ z]$ to allocate time value $t(\mathbf{p})$ to each point according to the following scheduling function:

$$t(\mathbf{p}) = \mathbf{s}\ \mathbf{p} - u = iw + jz - u \tag{10}$$

where u is an offset used to avoid assigning negative time values to any node of DG. For the given DG, it is equal to zero.

The valid scheduling vector is determined based on the timing limitations imposed by the DG. For instance, point allocated at $\mathbf{p}(i+1, m-1)$ should be processed after point allocated at $\mathbf{p}(i, m-1)$. This means that $t(\mathbf{p}(i+1, m-1)) > t(\mathbf{p}(i, m-1))$. This leads to the following inequality:

$$(i+1)w + (m-1)z - u > iw + (m-1)z - u \tag{11}$$

$$w > 0. \tag{12}$$

At each row, we notice another timing limitation on $\mathbf{s}$ that the point $(i, m-1)$ should be processed at the same time or before the other points allocated at (i, j), where $j < m-1$. This leads to the following inequality:

$$iw + \text{jz-u} \geq iw + (m-1)z - u \tag{13}$$

$$z \leq 0. \tag{14}$$

From inequalities (12) and (14), we can choose the valid scheduling vector leading to the compact bit-parallel systolic array structure. This vector should be as follows:

$$\mathbf{s} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{15}$$

The resultant node timing from applying the $\mathbf{s}$ vector to the DG is exhibited in Fig. 2. We notice that input signals c_j^0, a_j^0, a_{j-1}^0, and f_{j+1} can be applied in parallel, and the output signals c_j^m can be produced in parallel after m clock cycles.

B. Projection Function

As discussed in [14], we can use the following projection function to map any point $\mathbf{p}(i, j) \in \mathbb{D}$ to point $\overline{\mathbf{p}(i, j)}$ in the systolic array. Later, we will call $\overline{\mathbf{p}(i, j)}$ the PE

$$\overline{\mathbf{p}} = \mathbf{K}\mathbf{p} \tag{16}$$

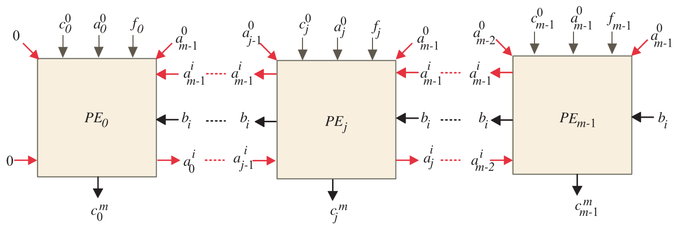


Fig. 3. Proposed bit-parallel systolic multiplier.

where **K** represents a projection matrix. This matrix is defined using the projection vector **v**, which is the null space of **K**. As discussed in [15], the following restriction is the only constraint imposed on the projection direction:

$$\mathbf{s}\ \mathbf{v} \neq 0. \tag{17}$$

This restriction assures that each processing node performs the assigned tasks at different time steps. Also, it guarantees a better utilization of all the processing nodes at each time step.

Using the scheduling vector $\mathbf{s} = [1 \quad 0]$, the valid projection direction leading to the compact bit-parallel systolic array is given by

$$\mathbf{v} = \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{18}$$

Therefore, the projection matrix associated with **v** is given by

$$\mathbf{K} = \begin{bmatrix} 0 & 1 \end{bmatrix}. \tag{19}$$

C. Extraction of the Systolic Array

Using (10) and (16) for scheduling vector $\mathbf{s} = [1 \quad 0]$ and projection matrix $\mathbf{K} = [0 \quad 1]$, we can determine the scheduling and projection functions, respectively, for any point $\mathbf{p}(i, j)$ as follows:

$$t(\mathbf{p}) = i \tag{20}$$
$$\overline{\mathbf{p}}(\mathbf{p}) = j. \tag{21}$$

Fig. 3 displays the resulted bit-parallel 1-D systolic array after applying (10) and (16) to the DG points (nodes). It has m PEs and should complete m time steps to deliver the output result. Also, the hardware complexity of the proposed design is of order $\mathcal{O}(m)$. To the best of our knowledge, the reported bit-parallel systolic arrays implementing finite field multiplication have hardware complexity of order $\mathcal{O}(m^2)$. Therefore, the proposed bit-parallel systolic array significantly outperforms the preceding bit-parallel systolic ones in terms of the area besides almost having a comparable computation time.

We can describe the bit-parallel systolic array as follows. The initial bit values of input A, a_j^0, and its most significant bit (MSB), a_{m-1}^0, besides the input bits of F, f_j, are allocated to the PEs. The initial values of C, c_j^0, are generated by resetting the corresponding Latches, D_c Latches, inside the PEs due to having an initial value of zero. The intermediate bits of A, a_j^i, are pipelined between the PEs, while the intermediate bits of C, c_j^0, are updated locally inside the PEs. PE$_{m-1}$ produces the bits of a_{m-1}^i, which are distributed alongside the input bits of b_i to all PEs. All PEs produce the final resulted bits of the partial product C, c_j^m, in parallel at the last processing time step.

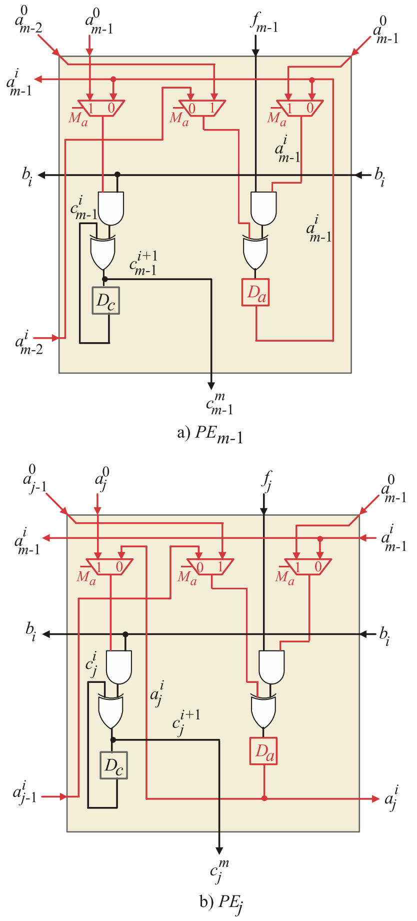


Fig. 4. Logic details for PEs. (a) PE$_{m-1}$. (b) PE$_j$. D_a and D_c boxes represent D-Latches.

The systolic array takes m clock cycles to produce the final result. Fig. 4 shows the logic details of the PEs. Fig. 4(a) provides the logic details of the last PE (PE$_{m-1}$), while Fig. 4(b) provides the logic details of the remaining PEs (PE$_j$). The last PE (PE$_{m-1}$) produces the MSB of A, a_{m-1}^i, after each processing step, and shares it between the remaining PEs (PE$_j$). The D_c Latches inside the PEs should be cleared before the systolic array starts its operation, to ensure that the input bits of C, c_j^0, are assigned zero values.

The following summarizes the operation of the developed systolic multiplier.

1) At the first clock cycle, M_a Muxes pass the initial bits of A, a_{j-1}^0, a_j^0, and a_{m-1}^0 to the PEs. Also, input bit b_0 is shared between all the PEs.
2) Through the remaining $m-1$ clock cycles, the last PE (PE$_{m-1}$) produces the MSBs of A, a_{m-1}^i, $1 \leq i < m$. These bits, alongside the input bits of b_i, $1 \leq i < m$, are broadcasted bit-by-bit (one bit at each clock cycle) to all PEs.
3) At the last clock cycle (clock cycle m), the PEs produces in parallel the final out bits of the partial product C, c_j^m, as indicated in Fig. 4.

TABLE I

HARDWARE AND DELAY COMPLEXITIES OF THE DIFFERENT MULTIPLIER DESIGNS

Design	AND	XOR	MUX	Latch	Latency	CPD	Throughput
Chiou [23]	m^2	m^2+m	m	$2m^2+3m$	$m+1$	$T_A+T_X+T_M$	1
Kim [3]	$2m^2+2m$	$2m^2+3m$	0	$3m^2+4m$	$\lfloor \frac{m}{2} \rfloor + 1$	T_A+T_X	1
Sarmadi [10]	$(m^2)^*$	$1.5m^2+0.5m$	$1.5m^2-2.5m+3$	$1.5m^2+2m-1$	$m+2$	T_N+T_X	1
Mathe [11]	m	m^2-1	m^2-m	m^2	m	T_M+2T_X	1
Mathe [24]	$2m$	$2m$	$2m$	$3m$	m	$T_A+T_X+T_M$	1
Proposed	$2m$	$2m$	$3m$	$2m$	m	$T_A+T_X+T_M$	1

($*$) 2-input NAND gates

TABLE II

PERFORMANCE COMPARISON OF AREA (A) AND DELAY (T) OF DIFFERENT MULTIPLIER DESIGNS FOR $m = 233$ AND $m = 409$

Design	Type	m	A [*Kgates*]	T [*ns*]	AT	% reduction in A	% AT improvement
Chiou [23]	Systolic	233	1,959.15	10.32	20,218.43	99.71%	99.73%
		409	4,731.13	17.73	83,882.93	99.78%	99.79%
Kim [3]	Systolic	233	3,320.23	4.41	14,642.21	99.83%	99.63%
		409	9,150.35	6.80	62,222.38	99.88%	99.72%
Sarmadi [10]	Systolic	233	2676.40	6.85	18,333.34	99.79%	99.76%
		409	7441.23	10.55	78,504.98	99.80%	99.81%
Mathe [11]	Systolic	233	1,954.10	11.61	22,687.10	99.71%	99.76%
		409	5,202.00	18.21	94,728.42	99.80%	99.81%
Mathe [24]	Sequential	233	6.92	10.52	72.79	19.36%	27.33%
		409	13.53	18.55	250.98	24.02%	31.27%
Proposed	Systolic	233	5.58	9.48	52.89	-	-
		409	10.28	16.78	172.49	-	-

IV. RESULTS AND DISCUSSION

In this section, we compare the hardware and delay complexities of the recommended parallel design and the previously reported competitive parallel designs of [3], [10], [11], and [23], as well as the competitive sequential design of [24]. Table I shows the estimated hardware complexity (number of logic gates and logic components) and delay complexity [latency and critical path delay (CPD)], as well as the throughput of the compared designs. T_A, T_N, T_X, and T_M represent the delays of the two-input AND gate, two-input NAND gate, two-input XOR gate, and the 2-to-1 multiplexer, respectively. As we notice from the Table, the hardware complexity of the offered design is of order $\mathcal{O}(m)$, while it is of order $\mathcal{O}(m^2)$ for the compared parallel structures of [3], [10], [11], and [23] and order $\mathcal{O}(m)$ for the sequential design of [24]. On the other hand, it has a similar latency as the designs of Mathe and Boppana [11], [24] and slightly lower latency than the designs of Bayat-Sarmadi and Farmani [10] and Chiou *et al.* [23], but it has significantly higher latency compared with the design of Kim and Jeon [3]. The CPD of the offered design is slightly larger than the systolic designs of Kim and Jeon [3] and Sarmadi and Boppana [10], but it has the same CPD as the designs of Mathe and Boppana [11], [24] and Chiou *et al.* [23].

It is worth considering the following regarding the design of Mathe and Boppana [24]: the authors ignored the MUXes at the inputs of the operand registers that select between the initial values of the input operands and the updated intermediate partial results. The ignored MUXes resulted in reducing the gate counts by $2m$ 2-to-1 MUXes and the CPD by the delay of a 1-bit 2-to-1 MUX. Table I shows the estimated hardware and delay complexities of the design of Mathe and Boppana [24] after adding the ignored MUXes. Table I also compares the throughput of the proposed design to all the remaining designs. They all have the same throughput of producing one output result for every clock cycle after different initial latencies.

Table II provides the real performance results of the compared designs for the most recommended NIST field sizes of elliptic-curve cryptography (ECC): $m = 233$ and $m = 409$. All designs are modeled using VHDL hardware description language and synthesized at their maximum operating frequency using Synopsys tools version 2005.09-SP2 and Nangate (1.5 nm, 0.8 V) open-cell library. We used the typical corner (VDD = 0.8 V and $T_j = 25°C$) and unit drive strength for all the utilized primitives. Table II shows the total hardware complexity (A) and the whole computation delay complexity (T). It also shows the improvement of area-delay complexity (AT) of the offered design over the compared ones.

The obtained results show that the developed systolic parallel design realizes a meaningful reduction in the hardware complexity over the compared systolic parallel designs by at least 99.71% for $m = 233$ and 99.78% for $m = 409$, respectively. Also, it achieves a considerable reduction in hardware complexity over the sequential design of Mathe and Boppana [24] by 19.36% for $m = 233$ and 24.02% for $m = 409$, respectively. The results also display that the proposed design has a significant improvement in AT over the compared parallel designs by at least 99.72% for $m = 233$ and 99.76% for $m = 409$, respectively. Compared with the sequential design of Mathe and Boppana [24], the proposed systolic design has a considerable improvement in AT by 27.33% and 31.27%, respectively. Thus, the obtained results make the offered systolic design more efficient for use in constrained hardware environments, having more restrictions on space, such as portable devices and smart cards.

The real implementation results of the proposed systolic design show a significant reduction in hardware and delay complexities over that of the sequential design of Mathe and Boppana [24] even though it has a slightly less estimated number of gates/components and the same estimated latency and CPD, as shown in Table I. The remarkable increase in hardware and delay complexities of the sequential design over the proposed one is mainly attributed to the massive increase in area and delay of its interconnecting wires. The sequential design has long feedback wires between its registers and logic modules that significantly increase its hardware and delay complexities. On the other hand, the proposed design has a systolic structure with local interconnections (short wires) between its PEs resulting in less contribution to its total hardware and delay complexities.

V. Conclusion

In this article, we presented a compact and efficient bit-parallel systolic multiplier for multiplication over the binary extension field. It is developed based on the proper determination of the scheduling and projection vectors used to assign each DG node to the corresponding PE. The space complexity of the exhibited systolic multiplier is of order $\mathcal{O}(m)$, and this differentiates it from the rest of the previously reported parallel designs, which has a space complexity of order $\mathcal{O}(m^2)$. The achieved results disclosed that the developed architecture realizes a meaningful reduction in space complexity and the area-delay complexity over the previously published competing ones. Therefore, it is more efficient for use in applications imposing more restrictions on space.

References

[1] C.-C. Chen, C.-Y. Lee, and E.-H. Lu, "Scalable and systolic montgomery multipliers over GF(2^m)," *IEICE Trans. Fundam.*, vol. 91, no. 7, pp. 1763–1771, 2008.

[2] S. Roman, *Field Theory*. 2nd ed. New York, NY, USA: Springer-Verlag, 1983.

[3] K.-W. Kim and J.-C. Jeon, "Polynomial basis multiplier using cellular systolic architecture," *IETE J. Res.*, vol. 60, no. 2, pp. 194–199, Mar. 2014.

[4] S. Choi and K. Lee, "Efficient systolic modular multiplier/squarer for fast exponentiation over GF (2^m)," *IEICE Electron. Express*, vol. 12, no. 11, pp. 1–6, 2015.

[5] K. W. Kim and J. C. Jeon, "A semi-systolic Montgomery multiplier over GF (2^m)," *IEICE Electron. Express*, vol. 12, no. 21, pp. 1–6, 2015.

[6] C.-Y. Lee, E. H. Lu, and J. Y. Lee, "Bit-parallel systolic multipliers for GF(2^m) fields defined by all-one and equally spaced polynomials," *IEEE Trans. Comput.*, vol. 50, no. 5, pp. 385–393, May 2001.

[7] C.-Y. Lee, E. H. Lu, and L. F. Sun, "Low-complexity bit-parallel systolic architecture for computing AB^2+C in a class of finite field GF (2^m)," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 5, pp. 519–523, May 2001.

[8] C.-Y. Lee and C.-W. Chiou, "Efficient design of low-complexity bit-parallel systolic Hankel multipliers to implement multiplication in normal and dual bases of GF (2^m)," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E88, no. 11, pp. 3169–3179, 2005.

[9] C.-Y. Lee, "Low-latency bit-parallel systolic multiplier for irreducible $x^m + x^n + 1$ with GCD(m,n) = 1," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 55, no. 3, pp. 828–837, 2008.

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

[11] S. E. Mathe and L. Boppana, "Bit-parallel systolic multiplier over GF (2^m) for irreducible trinomials with ASIC and FPGA implementations," *IET Circuits, Devices Syst.*, vol. 12, no. 4, pp. 315–325, 2018.

[12] B. B. Zhou, "A new bit-serial systolic multiplier over GF (2^m)," *IEEE Trans. Comput.*, vol. 37, no. 6, pp. 749–751, Jun. 1988.

[13] S. T. J. Fenn, D. Taylor, and M. Benaissa, "A dual basis bit-serial systolic multiplier for GF (2^m)," *Integration*, vol. 18, nos. 2–3, pp. 139–149, 1995.

[14] C.-W. Chiou, C.-Y. Lee, A.-W. Deng, and J.-M. Lin, "Concurrent error detection in montgomery multiplication over GF(2^m)," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 89, no. 2, pp. 566–574, 2006.

[15] F. Gebali, *Algorithms and Parallel Computers*. New York, NY, USA: Wiley, 2011.

[16] 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*, vol. 22, no. 11, pp. 2894–2906, Nov. 2017.

[17] A. Ibrahim, T. F. Al-Somani, and F. Gebali, "New systolic array architecture for finite field inversion," *Can. J. Electr. Comput. Eng.*, vol. 40, no. 1, pp. 23–30, win. 2017.

[18] A. Ibrahim, F. Gebali, H. El-Simary, and A. Nassar, "High-performance, low-power architecture for scalable radix 2 montgomery modular multiplication algorithm," *Can. J. Electr. Comput. Eng.*, vol. 34, no. 4, pp. 152–157, 2009.

[19] A. Ibrahim, T. Alsomani, and F. Gebali, "Unified systolic array architecture for finite field multiplication and inversion," *Comput. Elect. Eng.*, vol. 61, pp. 104–115, Jul. 2017.

[20] F. Gebali and A. Ibrahim, "Low space-complexity and low power semi-systolic multiplier architectures over GF (2^m) based on irreducible trinomial," *Microprocessors Microsyst.*, vol. 40, pp. 45–52, Feb. 2016.

[21] A. Ibrahim, H. Elsimary, and F. Gebali, "New systolic array architecture for finite field division," *IEICE Electron. Express*, vol. 15, no. 11, pp. 1–11, 2018.

[22] A. Ibrahim, "Efficient parallel and serial systolic structures for multiplication and squaring over GF(2^m)," *Can. J. Elect. Comput. Eng.*, vol. 42, no. 2, pp. 114–120, Spring 2019.

[23] C. W. Chiou, J.-M. Lin, C.-Y. Lee, and C.-T. Ma, "Novel Mastrovito multiplier over GF(2^m) using trinomial," in *Proc. 5th Int. Conf. Genetic Evol. Comput.*, Aug./Sep. 2011, pp. 237–242.

[24] S. E. Mathe and L. Boppana, "Design and implementation of a sequential polynomial basis multiplier over GF(2^m)," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 5, pp. 2680–2700, 2017.

Atef Ibrahim (Member, IEEE) received the B.Sc. degree in electronics from Mansoura University, Mansoura, Egypt, in 1998, the M.Sc. degree in electronics and electrical communications from Cairo University, Giza, Egypt, in 2004, and the Ph.D. degree in electronics and electrical communications from Cairo University and the University of Victoria, Victoria, BC, Canada, in 2010.

He is currently an Associate Professor of computer engineering with Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. He is an Adjunct Professor with the University of Victoria. His research interests include computer arithmetic, cryptography, biocomputing, embedded systems design, and digital VLSI design.

Fayez Gebali (Life Senior Member, IEEE) received the B.Sc. degree in electrical engineering from Cairo University, Giza, Egypt, in 1972, and the Ph.D. degree in electrical engineering from The University of British Columbia, Vancouver, BC, Canada, in 1979.

He is currently a Professor of computer engineering with the University of Victoria, Victoria, BC, Canada. His research interests include parallel algorithms, 3-D IC design, and wireless communications.

Yassine Bouteraa received the National Engineering degree in electrical engineering and the M.Sc. degree in control and computer science from the National School of Engineers of Sfax, Sfax, Tunisia, in June 2006 and 2007, respectively, the Ph.D. degree in electrical and computer engineering from the University of Orléans, Orléans, France, in 2012, and the HDR (accreditation to supervise research) degree in electrical and computer engineering from the University of Sfax, Sfax, in 2017.

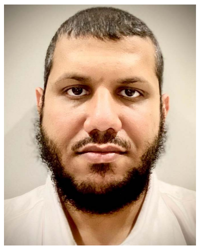
He is the author/coauthor of more than 50 scientific articles. His interest concerns robotics, embedded systems, and real-time implementation.

Dr. Bouteraa is also a TPC member of some international conferences and a reviewer of some indexed journals.

Usman Tariq received the Ph.D. degree in information and communication technology in computer science from Ajou University, Suwon, South Korea, in 2010.

He is having a strong background in *ad hoc* networks and network communications. He is a skilled research engineer. He is also experienced in managing and developing projects from conception to completion. He has worked on large international scale and long-term projects with multinational organizations. He is currently an Associate Professor with the College of Computer Engineering and Science, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. His research interests include networking and security fields.

Tariq Ahanger (Member, IEEE) is currently an Associate Professor with the Department of Information Systems, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. He has authored over 40 referred articles. His interests include the Internet of Things, cybersecurity, and artificial intelligence.



Khaled Alnowaiser received the B.Sc. degree in computer engineering from King Saud University, Riyadh, Saudi Arabia, in 2001, and the Ph.D. degree in computer science from the University of Glasgow, Glasgow, U.K., in 2016.

He is currently an Assistant Professor with Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia. His research interests include computer architecture and security.