electronics

MDPI

Article

Word-Based Systolic Processor for Field Multiplication and Squaring Suitable for Cryptographic Processors in Resource-Constrained IoT Systems

Atef Ibrahim [1,2,*] and Fayez Gebali [2]

[1] Computer Engineering Department, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia
[2] Electrical and Computer Engineering Department, University of Victroia, Victoria, BC V8P 5C2, Canada; atef@ece.uvic.ca or fayez@uvic.ca
* Correspondence: aa.mohamed@psau.edu.sa or attif_ali2002@yahoo.com

Abstract: Internet of things (IoT) technology provides practical solutions for a wide range of applications, including but not limited to, smart homes, smart cities, intelligent grid, intelligent transportation, and healthcare. Security and privacy issues in IoT are considered significant challenges that prohibit its utilization in most of these applications, especially relative to healthcare applications. Cryptographic protocols should be applied at the different layers of IoT framework, especially edge devices, to solve all security concerns. Finite-field arithmetic, particularly field multiplication and squaring, represents the core of most cryptographic protocols and their implementation primarily affects protocol performance. In this paper, we present a compact and combined two-dimensional word-based serial-in/serial-out systolic processor for field multiplication and squaring over GF(2^m). The proposed structure features design flexibility to manage hardware utilization, execution time, and consumed energy. Application Specific Integrated Circuit (ASIC) Implementation results of the proposed word-serial design and the competitive ones at different embedded word-sizes show that the proposed structure realizes considerable saving in the area and consumed energy, up to 93.7% and 98.2%, respectively. The obtained results enable the implementation of restricted cryptographic primitives in resource-constrained IoT edge devices such as wearable and implantable medical devices, smart cards, and wireless sensor nodes.

Keywords: resource-constrained IoT systems; security in cyber physical systems; cryptographic processors; IoT edge security; finite-field arithmetic; low power design; systolic arrays

Citation: Ibrahim, A.; Gebali, F. Word-Based Systolic Processor for Field Multiplication and Squaring Suitable for Cryptographic Processors in Resource-Constrained IoT Systems. *Electronics* **2021**, *10*, 1777. https://doi.org/10.3390/electronics10151777

Academic Editors: Carsten Maple, Matthew Bradbury and Munam Ali Shah

Received: 28 May 2021
Accepted: 23 July 2021
Published: 25 July 2021

1. Introduction

Internet of Things (IoT) is a modern technology that connects a tremendous number of gadgets—such as smartphones, wearable devices, sensors, vehicles, and smart meters—to the internet [1,2]. It provides services and efficient solutions in numerous domains such as healthcare, smart cities, smart grid, industrial manufacturing, business management, logistics, smart homes, and intelligent transportation [3–8].

Security and privacy issues are the primary concern in most IoT-based systems. They prohibit its usage in most applications, especially healthcare applications. Accordingly, we should employ efficient and practical security solutions to protect the IoT-based systems. Therefore, cryptographic protocols should be applied at the different layers of the IoT framework, especially edge devices, to solve all security concerns. Most IoT edge devices have limited computing resources, which makes implementing traditional cryptographic algorithms, such as Rivest, Shamir, and Adleman (RSA) and Digital Signature Algorithm (DSA) [9], impractical. Due to its short-key sizes and enhanced computational efficiency, the Elliptic Curve Cryptographic (EEC) algorithm [10] becomes the cryptography choice for resource-constrained embedded devices such as mobile phones, smart cards, and

environmental sensors. ECC's essential operation is the point multiplication, which mainly depends on the basic finite field arithmetic operations of addition, multiplication, squaring, and division/inversion. The ECC processor's overall performance primarily relies on the efficient implementation of these operations. Since the finite field multiplier is the basic building block of the other field operations of division/inversion and exponentiation, it is considered the fundamental building block of the ECC processor. Therefore, any slight improvement in its implementation results in a significant increase in the ECC processor's whole performance.

1.1. Paper Motivation and Related Work

Field multiplication in GF(2^m) is very crucial in several field operations such as modular exponentiation and inversion/division as they are performed using a sequence of multiplications. Most of the previously reported multipliers over GF(2^m) have high area and time complexities that render their realization in resource-constrained IoT edge devices highly challenging [11–14]. Therefore, it becomes important to have multiplier architectures that target this type of applications. Word-serial multiplier architectures are reported in the literature to solve this problem. They have a trade-off between speed and area complexities and, thus, they provide the designer more flexibility to reach the desired design. The structures of word-serial multipliers are classified into four types: Serial-In/Serial-Output (SISO) structures, Serial-In/Parallel-Output (SIPO) structures, Parallel-In/ Serial-Output (PISO), and Scalable structures. The polynomial basis word-serial systolic multipliers using SISO structure are presented in [15–19]. The polynomial basis word-serial multipliers with the SIPO structure are reported in [20–23]. The word-serial type-T Gaussian normal basis (GNB) multipliers with PISO structure are reported in [24]. The scalable systolic multiplier structures are reported in [25–31].

Modular exponentiation is a fundamental part of cryptographic algorithms. There are two binary approaches used to compute modular exponentiation: the Most Significant Bit (MSB)-first approach and the Least Significant Bit (LSB)-first approach. In LSB-first approach, the modular multiplication and squaring operations can be executed concurrently to reduce the processing time. There are many attempts in the literature to combine the multiplication and squaring operations in a unified structure to increase performance and hardware utilization [13,14,32]. To the best of our knowledge, the suggested combined multiplier-squarer structures are dedicated for high-speed applications and do not target the resource-constrained applications.

1.2. Paper Contribution

In this paper, we propose a word-based two-dimensional SISO systolic processor for combined field multiplication and squaring over GF(2^m). The main difference between the proposed SISO architecture and the other types of word-serial multiplier architecture is that the proposed multiplier-squarer structure is extracted by using a systematic approach [33–39]. In contrast, the other word-serial multiplier structures are extracted conventionally without the use of specific methodology. The applied approach allows the designer to construct the design in the smallest size in order to fit all resource-constrained IoT edge devices that have more restrictions on area and power consumption. Moreover, it provides the flexibility in managing execution time and the consumed energy of these devices. Another advantage of the proposed SISO design over other word-serial conventional ones is that it provides a compact and unified structure that simultaneously performs multiplication and squaring operations. By contrast, the traditional design designs computes both operations sequentially. Moreover, it has a regular structure and local interconnections that render it more suitable for VLSI implementation.

1.3. Paper Organization

This paper can be organized as follows. Section 2 provides a brief explanation to the combined polynomial multiplication-squaring algorithm in $GF(2^m)$. Section 3 develops

its associated dependency graph (DG). Section 4 explains the explored two-dimensional word-based SISO systolic processor. Section 5 provides the area and delay complexities of the proposed design and the best of the existing word-serial designs. Section 6 concludes this study.

2. Combined Polynomial Multiplication-Squaring Algorithm in $GF(2^m)$

Let $C(x)$ and $D(x)$ be two polynomials in $GF(2^m)$ and $G(x)$ be the irreducible polynomial in standard basis representation. These polynomials can be represented as follows:

$$C(x) = \sum_{i=0}^{m-1} c_i x^i \quad (1)$$

$$D(x) = \sum_{i=0}^{m-1} d_i x^i \quad (2)$$

$$G(x) = \sum_{i=0}^{m} g_i x^i \quad (3)$$

where $c_i, d_i, g_i \in GF(2)$.

The polynomial multiplication and squaring over $GF(2^m)$ can be defined as follows.

$$R(x) = C(x)D(x) \bmod G(x) \quad (4)$$

$$Q(x) = C(x)C(x) \bmod G(x) \quad (5)$$

The products $R(x)$ and $Q(x)$ can be calculated using the combined algorithm, Algorithm 1, proposed by Choi in [13]. This algorithm calculates three partial polynomials $C(x)$, $R(x)$, and $Q(x)$. Variables C^i, R^i, and Q^i are used to indicate the values of $C(x)$, $R(x)$, and $Q(x)$ at iteration i. d_{i-1} and c_{i-1} represent the $(i-1)^{th}$ coefficients of input polynomials $D(x)$ and $C(x)$, respectively. The initial variables R^0 and Q^0 are assigned zero values and the initial variable C^0 is assigned the coefficients of input polynomial $C(x)$. In each i iteration of the for loop, the intermediate variables are updated as follows:

- Variable C^i is updated by shifting left C^{i-1} and by reducing using the irreducible polynomial G;
- Variable R^i is updated by multiplying C^{i-1} by coefficient d_{i-1} and by adding the obtained result to R^{i-1};
- Variable Q^i is updated by multiplying C^{i-1} by coefficient c_{i-1} and by adding the obtained result to Q^{i-1}.

Algorithm 2 is the bit-level representation of Algorithm 1. Variable c^i_{j+1} represents the $(j+1)^{th}$ bit of C at the i^{th} iteration. Moreover, r^i_j and q^i_j represent the i^{th} bit of R and Q at the i^{th} iteration, respectively. Notice that $\langle j+1 \rangle$ indicates that $j+1$ is to be reduced modulo m.

Algorithm 1 Algorithm for multiplication and squaring over GF(2^m) [13].

Input: $C(x)$, $D(x)$, and $G(x)$
Mult. Output: $R(x) = (C(x).\, D(x)) \bmod G(x)$
Square Output: $Q(x) = (C(x).\, C(x)) \bmod G(x)$
Initialization:
$R^0 \leftarrow 0$, $Q^0 \leftarrow 0$, $C^{(0)} \leftarrow C(x)$, $D \leftarrow D(x)$, $G \leftarrow G(x)$
Algorithm:
1: **for** $1 \leq i \leq m$ **do**
2: $C^i = C^{i-1}.\, x \bmod G(x)$
3: $R^i \leftarrow R^{i-1} + d_{i-1}C^{i-1}$
4: $Q^i \leftarrow Q^{i-1} + c_{i-1}C^{i-1}$
5: **end for**

Algorithm 2 Bit-level algorithm for multiplication and squaring over GF(2^m).

Input: $C(x)$, $D(x)$, and $G(x)$
Mult. Output: $R(x) = (C(x).\ D(x)) \mod G(x)$
Square Output: $Q(x) = (C(x).\ C(x)) \mod G(x)$
Initialization:
$R^0 = (r^0_{m-1} \cdots r^0_1 r^0_0) \leftarrow (0 \cdots 00)$
$Q^0 = (q^0_{m-1} \cdots q^0_1 q^0_0) \leftarrow (0 \cdots 00)$
$C^0 = (c^0_m \cdots c^0_1 c^0_0) \leftarrow (0 c_{m-1} \cdots c_1 c_0)$
$D \leftarrow (d_{m-1} \cdots d_1 d_0)$
$G \leftarrow (g_{m-1} \cdots g_1 g_0)$
Algorithm:
1: **for** $1 \le i \le m$ **do**
2: **for** $0 \le j \le m-1$ **do**
3: $c^i_{j+1} = c^{i-1}_j + c^{i-1}_{m-1} g_{\langle j+1 \rangle}$
4: $r^i_j = r^{i-1}_j + d_{i-1} c^{i-1}_j$
5: $q^i_j = q^{i-1}_j + c_{i-1} c^{i-1}_j$
6: **end for**
7: **end for**

3. Algorithm Dependency Graph

Algorithm 1 is an example of a Regular Iterative Algorithm (RIA). The authors of [33] showed how to obtain the dependency graph (DG) of an RIA algorithm. Figure 1 shows the DG based on Algorithm 2 for combined polynomial multiplication-squaring in $GF(2^m)$. The nodes in Figure 1 represent points in the two-dimensional integer domain $\mathbb{D}$, with indices i and j indicating the rows and columns, respectively, and they possess the following ranges:

$$1 \le i \le m, \qquad 0 \le j \le m-1 \tag{6}$$

The figure is for the case when $m = 5$ bits. The algorithm has three input variables C, D, and G and two output variables R and Q. Variables R, Q, and G are represented by the vertical lines. Variable C is represented by the slanted lines (red lines). Input bits c_{i-1} and d_{i-1} along with the resulting intermediate bits c^{i-1}_{m-1} are broadcasted horizontally. The initial bits r^0_j, q^0_j, c^0_j, and $g_{\langle j+1 \rangle}$ are inputs to the DG as shown at the top of Figure 1. The DG nodes (circles) execute the main operations of Algorithm 2 from steps 3 to 5. Output bits r^m_j and q^m_j are produced from the bottom of the DG as indicated in Figure 1.

The DG in Figure 1 can be used for design space exploration of the combined multiplication and squaring operations. The design exploration involves finding valid node scheduling functions and mapping or projecting the graph nodes to processing elements (PEs). Reference [33] explains how design space exploration could be performed by using affine and non-linear scheduling and projection functions.

The affine scheduling and projection functions cannot be used to explore word-serial systolic processors. Thus, our goal is to apply the non-linear scheduling and projection techniques discussed in [33] to the developed algorithm, Algorithm 2, in order to explore the most efficient two-dimension word-serial systolic processor that is able to satisfy any Input/Output (I/O) limitations/restrictions.

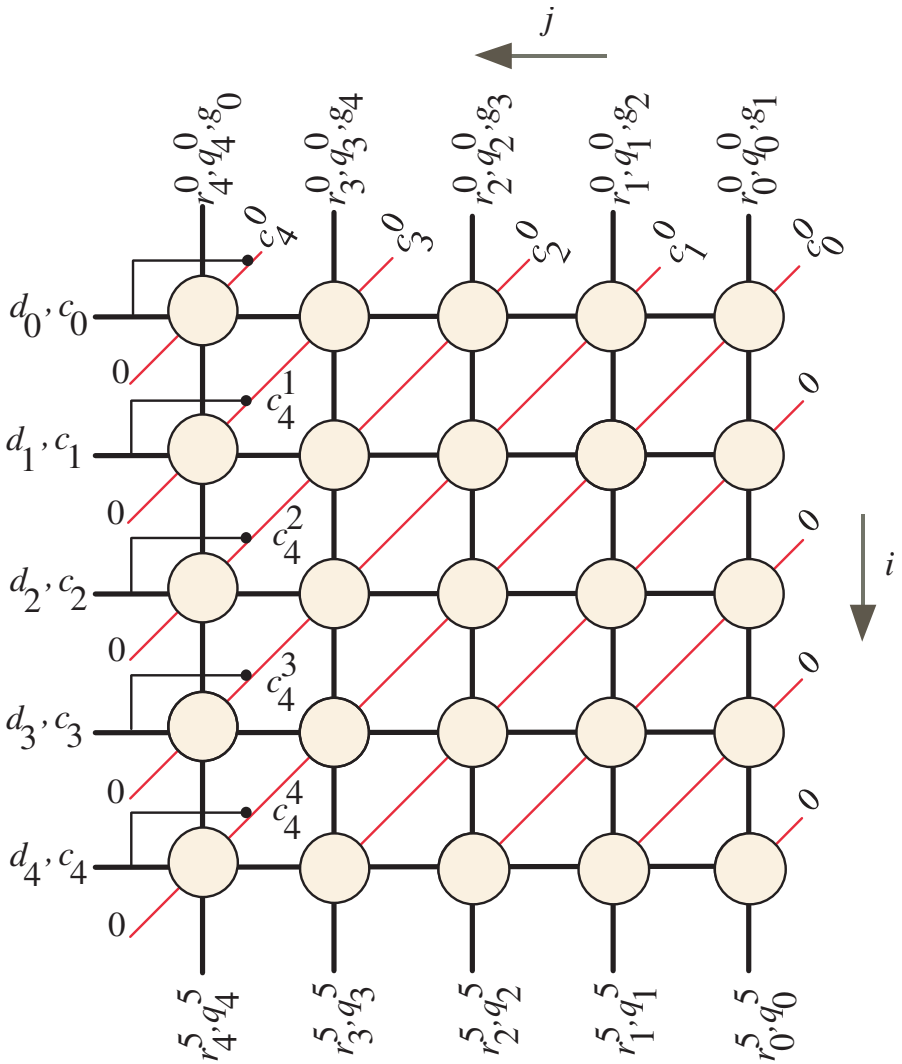

Figure 1. Dependence graph for the combined multiplication-squaring algorithm for $m = 5$.

4. Combined Two-Dimensional SISO Multiplier-Squarer

A SISO combined multiplier-squarer requires feeding in polynomials C, D, and G in a word-serial fashion at the start of iterations and then obtaining the Q and R polynomials in a word-serial fashion. Let us assume that we would like to perform w iterations at the same time; i.e., we would like to feed in w bits of the polynomial inputs and obtain w bits of the partial results. There are several nonlinear task scheduling and projection functions that can be used to obtain different two-dimensional SISO combined multiplier-squarer. The most efficient ones are discussed in the following sections.

4.1. Two-Dimensional SISO Task Scheduling

Following the scheduling methodology explained in [33], we can extract the following nonlinear scheduling function to partition $\mathbb{D}$ into $w \times w$ equitemporal zones:

$$n(\mathbf{p}) = \left\lceil \frac{m}{w} \right\rceil \left\lfloor \frac{i-1}{w} \right\rfloor + \left\lfloor \frac{m-1-j}{w} \right\rfloor + 1 \tag{7}$$

where $1 \leq i \leq m + \mu$ and $-\mu \leq j < m - 1$.

Figure 2 shows the node timing (scheduling time) for the case when $m = 5$ and $w = 2$.

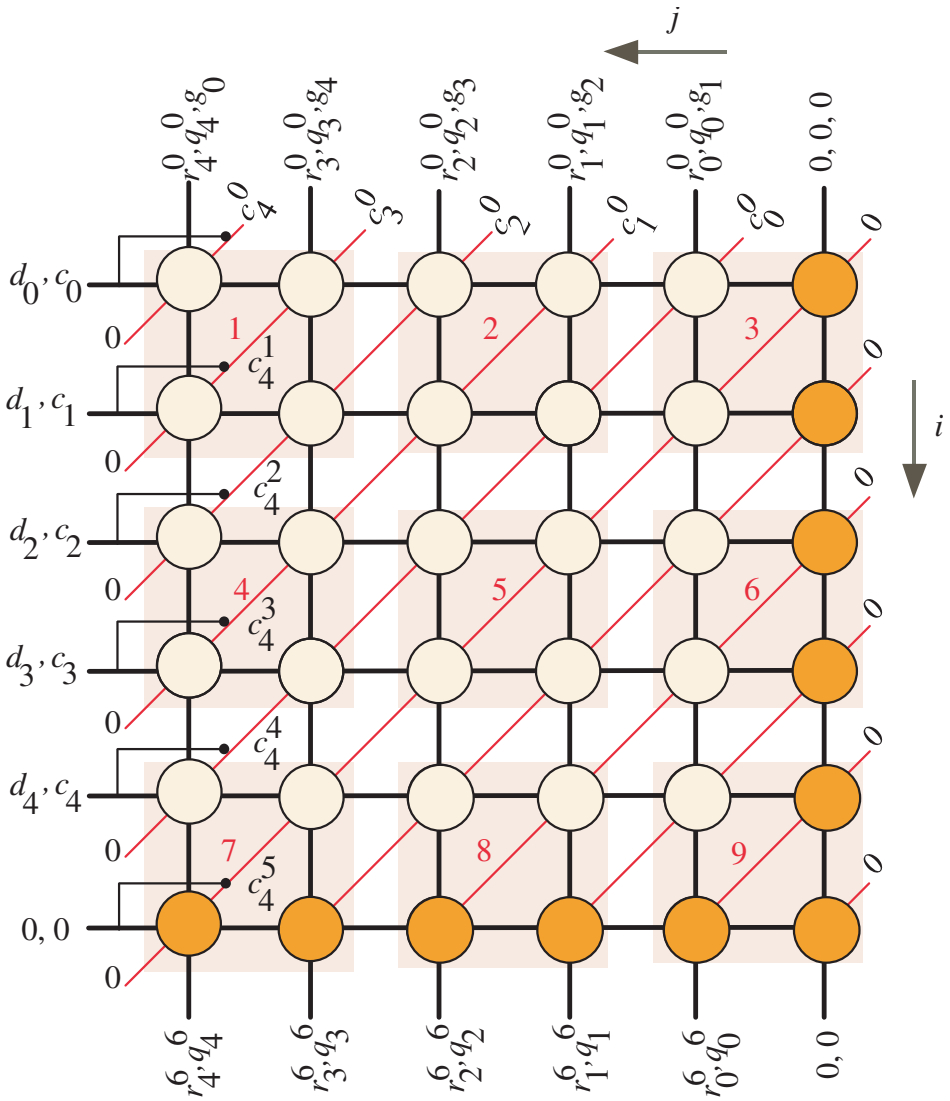

Figure 2. Scheduling time for the case when $m = 5$ and $w = 2$.

Notice that we added an extra column on the right and extra row at the bottom to render the number of columns and rows integer multiples of w. In general, we should add μ extra columns and μ extra rows to render the number of columns and rows an integer multiple of w, where $\mu = w\lceil \frac{m}{w} \rceil - m$.

Figure 3 shows the node timing (scheduling time) for the case when $m = 5$, and $w = 4$.

In this case $\mu = 3$, thus we had to add three extra columns on the right and three extra rows at the bottom to render the number of columns and rows an integer multiple of w. Therefore, the LSBs of inputs C and G and LSBs of the initial values of intermediate variables R and Q should be padded by μ zeros on the right as shown in Figure 3. Furthermore, the MSBs of inputs D and C should be padded by μ zeros at the bottom, as shown in the same figure.

The equitemporal zones are shown as light red boxes with the associated time index values indicated in red numerals within each zone. Notice that the bits of c_{m-1}^{i-1} are computed at the nodes of column $m-2$, as shown in Figure 3 and broadcasted horizontally along with the bits of d_{i-1} and c_{i-1} to the nodes of row $i-1$.

One last detail needs to be mentioned here and is best explained with reference to two adjacent equitemproal zones executing at times n and $n+1$. Figure 4 illustrates this situation.

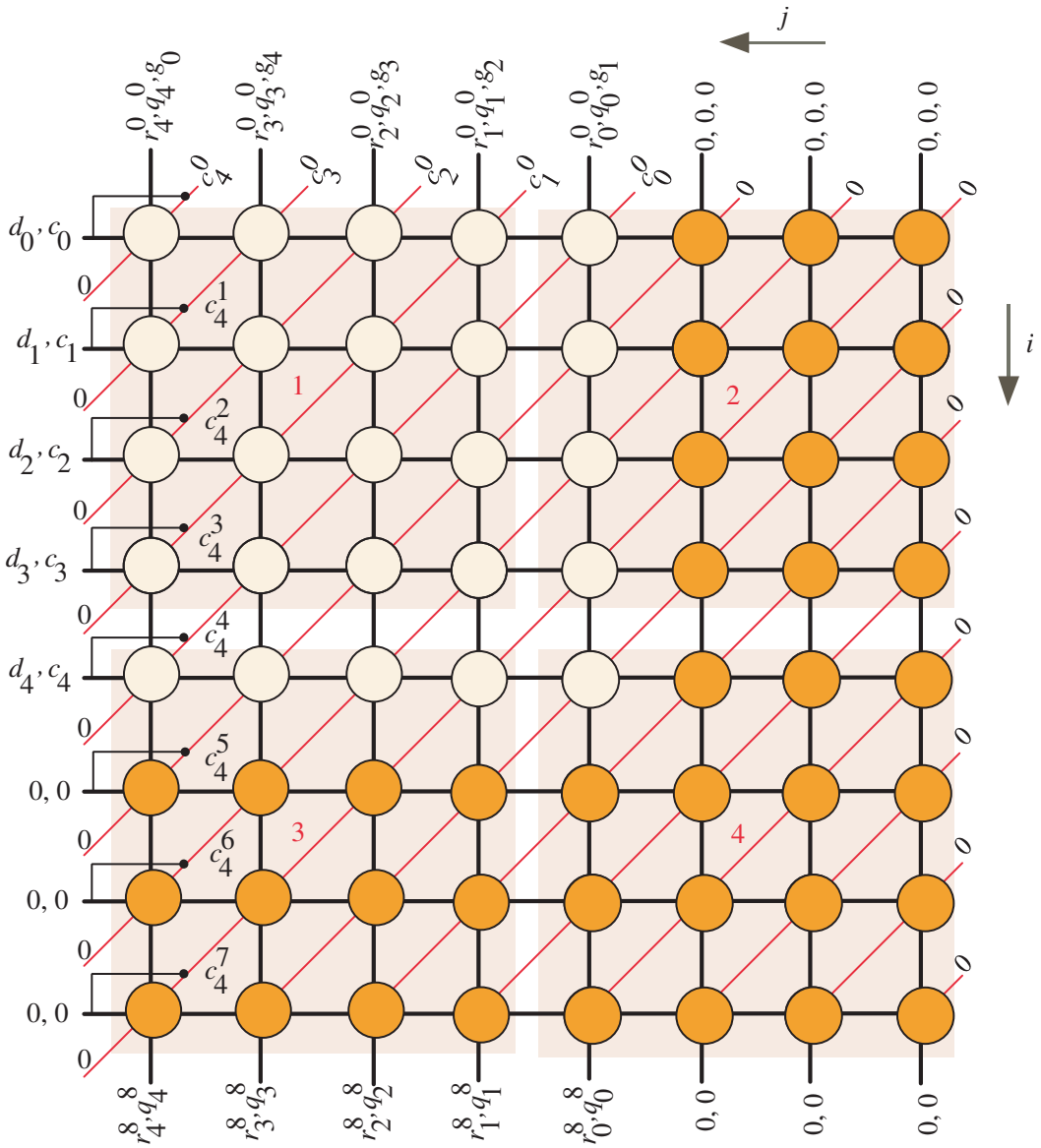

Figure 3. Scheduling time for the case when $m = 5$ and $w = 4$.

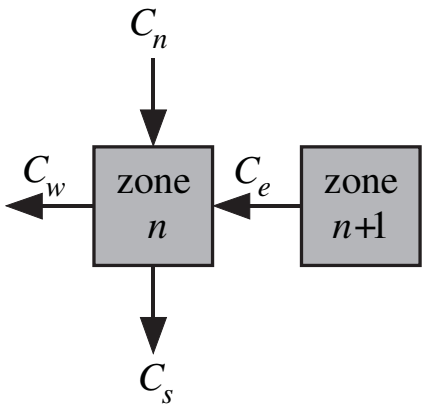

Figure 4. Data dependencies of two adjacent equitemporal zones from Figure 3.

The north and east inputs to zone i are available at times n and $n + 1$, respectively. However, we notice that input C_n only affects the west output C_w and C_e only affects the south output C_s. Hence, at time n output C_w is valid while output C_s is not valid since

we were required to add C_e to it. This will result in an increase in the total number of iterations needed to produce the final result by one time step. Therefore, the total number of iterations needed to complete the combined multiplication/squaring computation will be provided by:

$$\#\,\text{Iterations} = \left\lceil \frac{m}{w} \right\rceil^2 + 1 \tag{8}$$

Two-Dimentional SISO Task Projection

Given the scheduling time in Figure 3, we note that only $w \times w$ nodes are active at a given time. Following the projection technique explained in [33], we can extract the following nonlinear projection function that maps a point $\mathbf{p}(i,j) \in \mathbb{D}$ of Figure 3 to a point $\overline{\mathbf{p}}$ in the PE space:

$$\overline{\mathbf{p}}(o,l) = \mathbf{P}_{siso}\ \mathbf{p}(i,j) \tag{9}$$

$$o = i \bmod w \tag{10}$$

$$l = j \bmod w \tag{11}$$

$$\mathbf{P}_{siso} = [\ .\bmod w \quad .\bmod w\ ] \tag{12}$$

where "dot" is a place holder for the argument.

Our systolic array will now consist of w^2 PEs arranged in w rows and w columns in addition to the necessary registers. Figure 5 shows the word-based two-dimensional SISO systolic processor.

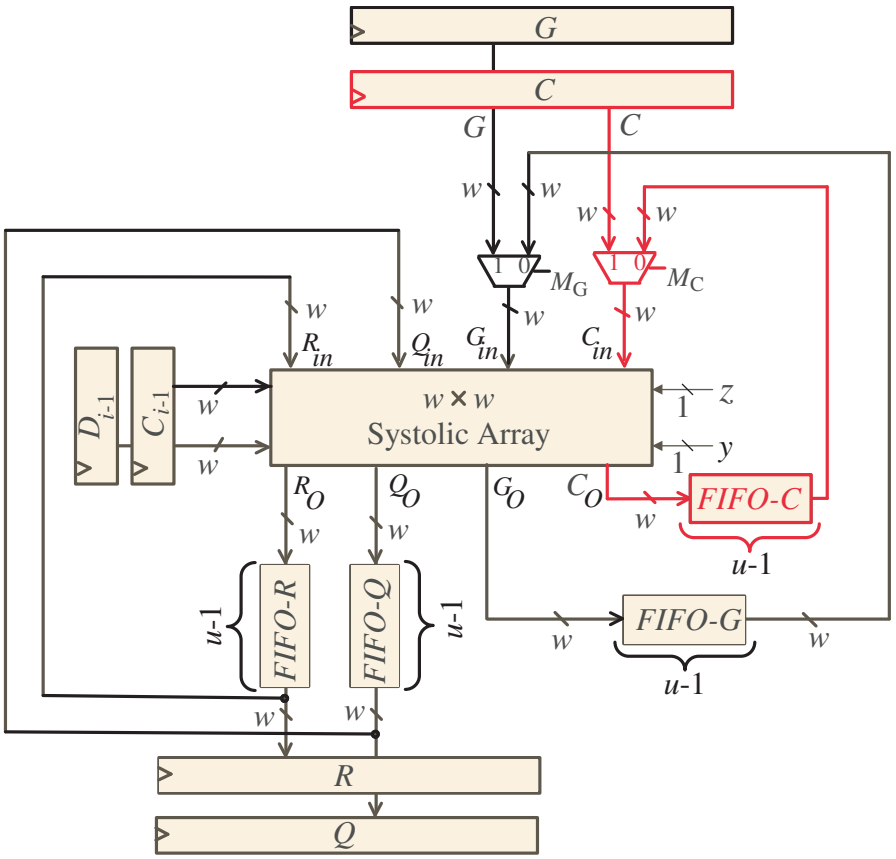

Figure 5. Word-based two-dimensional SISO systolic processor.

The Word-based two-dimensional SISO systolic processor consists of $w \times w$ systolic array block as well as input/output registers and FIFO buffers in addition to two 2-input MUXes for selecting between the inputs of C and G and their intermediate values. The resulting intermediate words of R, Q, and C and the words of G are pipelined through

the FIFO buffers of R, Q, C, and G, respectively. These FIFOs have a width size of w bits and a depth size of $u-1$, where $u = \lceil \frac{m}{w} \rceil$. The word update block ensures that the proper number of bits are extracted from the bottom outputs of the systolic array block shown in Figure 5. Notice that we added two registers for the input C: The north C register feeds the words of operand C to the systolic array starting from the most significant words, while the east register C_{i-1} feeds the words of operand C to the systolic array starting from the least significant words.

Figure 6 shows the details of the two-dimensional word-based SISO systolic array for the case when $m = 5$ and $w = 4$. The PEs of the systolic array are divided into two different types with each type possessing a different color. The logical details of the PEs are shown in Figures 7 and 8. The light blue PEs have an extra tri-state buffer that is enabled ($y = 0$) at time steps $n = k\lceil \frac{m}{w} \rceil + 1$ and $0 \leq k < \lceil \frac{m}{w} \rceil$ to pass the computed bits of C_{m-1}^{i-1}. These bits are broadcasted along with the input bits of D_{i-1} and C_{i-1} to the remaining nodes of the systolic array to compute the intermediate words of R, Q, and C. Moreover, they have an extra AND gate to enforce the partial results of the MSBs of C and c_m^i to be zero at the same time instances. This is controlled by the control signal y shown in Figure 7.

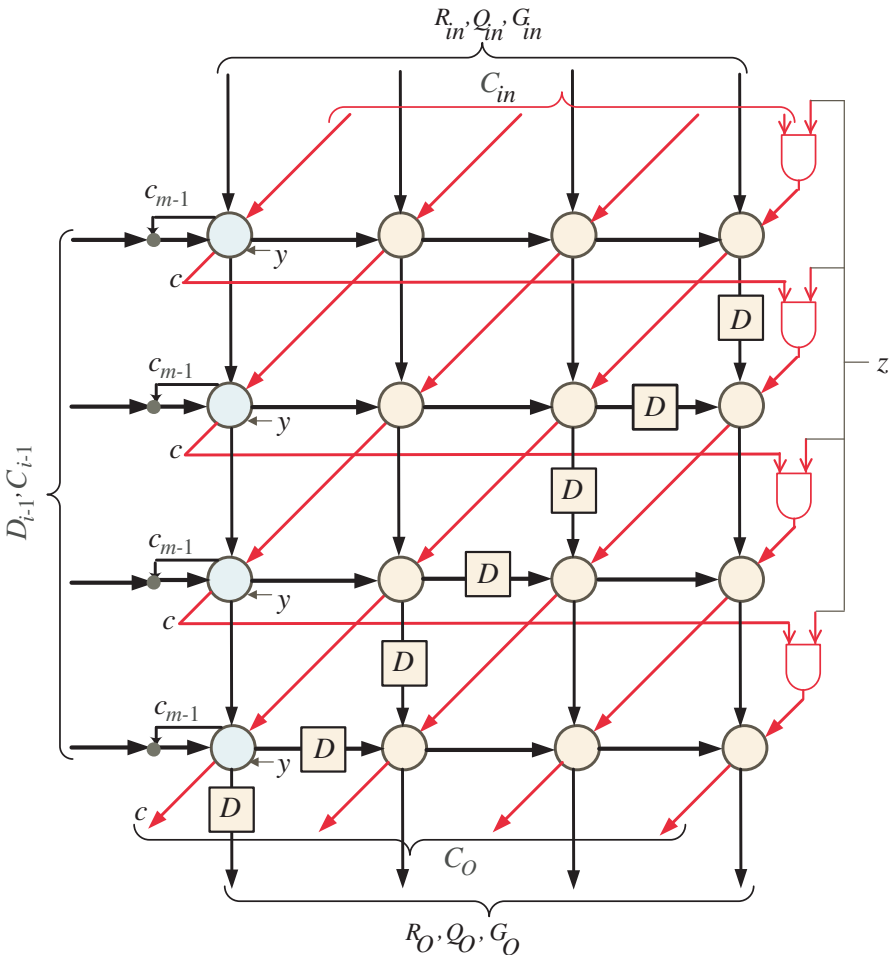

Figure 6. Word-based two-dimensional SISO systolic array.

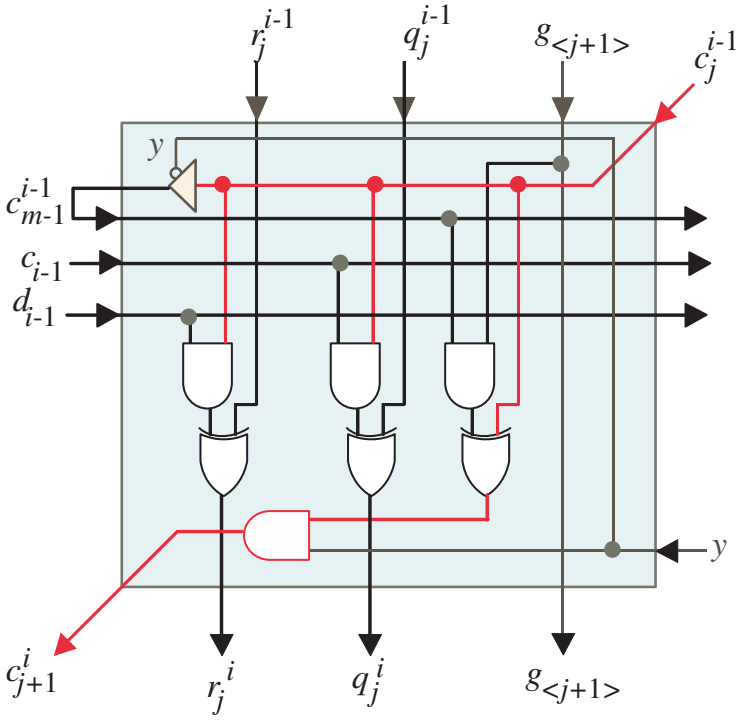

Figure 7. Light blue PE details of SISO two-dimensional systolic array.

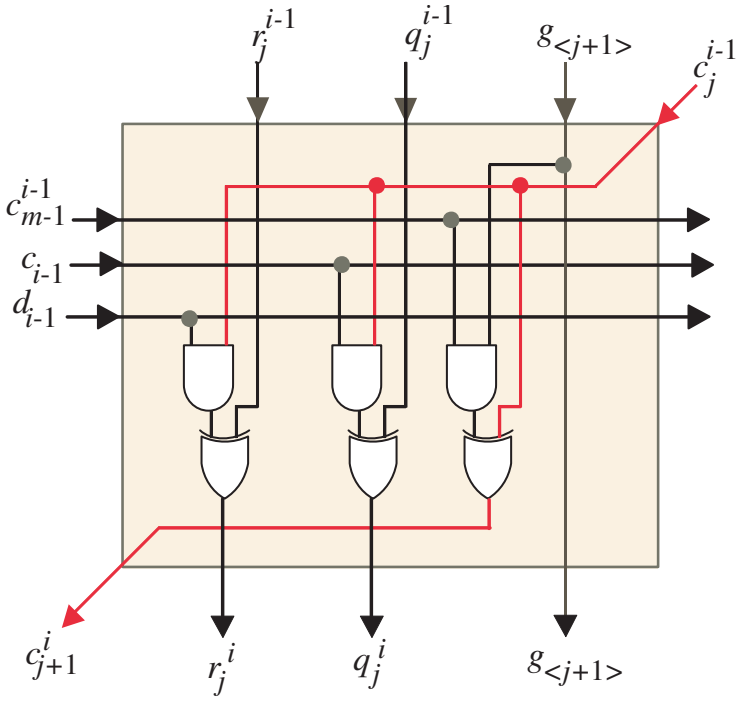

Figure 8. Light Orange PE details of SISO two-dimensional systolic array.

The operation of the two-dimensional SISO systolic processor can be summarized for the generic values of m and w as follows:

1. At time $n = 1$, MUXes M_C and M_G shown in Figure 5, are set to pass the w MSBs of operands C and G, respectively, to the systolic array block. Moreover, FIFO buffers of R and Q are reset at the same time to pass zero inputs to the systolic array block since the initial values of R and Q are zeros as indicated in Algorithm 1. Notice that, the control signals y and z are set to 0 and 1, respectively, through this time step. The

control signal $y = 0$ enables the tristate buffer shown in Figure 7 for all the light blue PEs of the systolic array, Figure 6, to pass the computed w bits of C_{m-1}^{i-1} and $1 \leq i \leq w$. The computed word of C_{m-1}^{i-1} along with the w LSBs of D_{i-1} and C_{i-1}, $1 \leq i \leq w$, are passed horizontally to the remaining PEs nodes of the systolic array. Moreover, the control signal $y = 0$ forces the bits of C_m^i and $1 \leq i \leq w$ through the AND gate shown in Figure 7 to have zero values as shown at the left edge of the DG, Figure 3.

2. At time instances $1 < n \leq \lceil \frac{m}{w} \rceil$, MUXs M_C and M_G are still set to pass the remaining words of inputs C and G, one word at each time step, to the systolic array. These operand words are used with the horizontally passed words of C_{m-1}^{i-1}, D_{i-1}, and C_{i-1}, $1 \leq i \leq w$, to compute the intermediate words of R, Q, and C in a word serial fashion. The resulting words of R, Q, and C are pipelined through the FIFOs of R, Q, and C shown in Figure 5, respectively. These FIFOs have a width size of w bits and a depth size of $u - 1$, where $u = \lceil \frac{m}{w} \rceil$. Notice that the depth of R and Q FIFOs ensures keeping the initial values of R and Q equal to zero through these time instances.
3. At time instances $n > \lceil \frac{m}{w} \rceil$, MUXs M_C and M_G passes the computed C words stored in FIFO-C and the G words stored in FIFO-G to the systolic array, one word at each time step. These words, along with the computed R and Q words that are stored in FIFO-R and FIFO-Q and the broadcasted words of C_{m-1}^{i-1}, D_{i-1}, and C_{i-1}, $kw < i \leq (k+1)w$, $1 \leq k \leq \lceil \frac{m}{w} \rceil - 1$, are used to update the intermediate partial results of R, Q, and C in a word serial fashion, one word at each time step.
4. At time instances $n = k\lceil \frac{m}{w} \rceil + 1$, $0 \leq k \leq \lceil \frac{m}{w} \rceil - 1$ the tri-sate buffer shown in Figure 7 is enabled ($y = 0$) in all the light blue PEs of the systolic array, Figure 6, to pass horizontally through the computed w bits of C_{m-1}^{i-1}, $kw < i \leq (k+1)w$, along with the w bits of inputs D_{i-1} and C_{i-1}, $kw < i \leq (k+1)w$, to the remaining PEs nodes of the systolic array. Notice that the D_{i-1} and C_{i-1} registers, shown in Figure 5, feeds the systolic array with the input words of D_{i-1} and C_{i-1} through these time instances. Furthermore, through these time instances, the control signal ($y = 0$) forces the bits of C_m^i and $kw < i \leq (k+1)w$ through the AND gate shown in Figure 7 to have zero values as shown at the left edge of the DG of Figure 3. At the remaining time instances, this control signal is equal to one.
5. Through time instances $n = k\lceil \frac{m}{w} \rceil + 1$, $1 \leq k \leq \lceil \frac{m}{w} \rceil$, the control signal z shown at the right side of Figure 6 is equal to zero to feed the zero values of C, shown at the right edge of DG of Figure 3, to the systolic array. At the remaining time instances, this control signal is equal to one.
6. Through time instances $n \geq \lceil \frac{m}{w} \rceil \left\lfloor \frac{m+\mu-1}{w} \right\rfloor$, the resulting output words of R and Q will be loaded in a word serial fashion, one word at each time step, in registers R and Q shown in Figure 5, respectively.

An important note that should be considered here is that the vertical w bit words of R, Q, and G and the horizontal w bit words of C_{m-1}^{i-1}, D_{i-1}, and C_{i-1} are delayed one time step inside the systolic array as shown in Figure 6. This is represented by the D registers (squares) shown in this figure. This renders a one time step difference between the PEs above the D registers (squares) and the PEs below of them. This time difference is attributed to the intermediate words of C, resulting from the left column (blue cells) of the systolic array shown in Figure 6, that are produced starting from the second time step and the words of R, Q, G, C_{m-1}^{i-1}, D_{i-1}, and C_{i-1} should be delayed, as shown in Figure 6, to synchronize the operation. This resulted in the extra time step needed to complete the the combined multiplication/squaring computation as explained before in Equation (8).

5. Experimental Results and Discussion

In this section, we compare the proposed two-dimensional word-serial combined multiplier-squarer structure and the best of the existing word-serial multiplier structures [18,23,40,41] in terms of area and time complexities. The area is estimated in terms of numbers of Tri-State buffers, 2-input AND gate, 2-input XOR gate, 2-input Multiplexers, and Flip-Flops. The time is represented by latency and Critical Path Delay (CPD).

The estimated area and time complexities of the compared structures are given in Table 1. In this Table, the field size and word size are represented by m and w, respectively. T_A represents the delay of 2-input AND gate. T_X represents the delay of 2-input XOR gate. T_{MUX} represents the delay of 2-to-1 MUX. The notations F_1, F_2, F_3, L_1, τ_1, τ_2, τ_3, and τ_4 are described by the following equations.

$$F_1 = 7m + m(\lceil \log m \rceil) + w + 3 \tag{13}$$

F_1 represents the number of Flip-Flops in Pan et al. [18] design.

$$F_2 = 2w^2 + 2w(\lceil m/w \rceil) + 4w + 1 \tag{14}$$

F_2 represents the number of Flip-Flops in Hua et al. [40] design.

$$F_3 = 2w^2 + 3w(\lceil m/w \rceil) + 2w \tag{15}$$

F_3 represents the number of Flip-Flops in Chen et al. [41] design.

$$L_1 = w + \lceil m/w \rceil^2 + \lceil m/w \rceil \tag{16}$$

L_1 represents the latency of Chen et al. [41] design.

$$\tau_1 = T_A + (\lceil \log_2 w \rceil + 1)T_X \tag{17}$$

τ_1 represents the critical path delay of Pan et al. [18] design.

$$\tau_2 = T_A + 2T_X \tag{18}$$

τ_2 represents the critical path delay of Hua et al. [40] design.

$$\tau_3 = T_A + T_X \tag{19}$$

τ_3 represents the critical path delay of Chen et al. [41] design.

$$\tau_4 = (w+1)T_A + wT_X + T_{MUX} \tag{20}$$

τ_4 represents the critical path delay of the proposed design.

Table 1. Comparison between different word-serial field multipliers.

Design	Tri-State	AND	XOR	MUXs	Flip-Flops	Latency	CPD
Xie [23]	0	$2mw$	$2mw + 6m - 6\frac{m}{w} + 6$	0	$4mw + 4m + 2w$	$2\lceil m/w \rceil + 2\lceil \log_2 w \rceil$	$2T_X$
Pan [18]	0	$m\sqrt{m}$	$\sqrt{mw}(2+m) + w$	0	F_1	$2\lceil \sqrt{m/w} \rceil$	τ_1
Hua [40]	0	w^2	$w^2 + 4 - 5w + 1$ (1)	0	F_2	$6w\lceil m/w \rceil^2$	τ_2
Chen [41]	0	$w^2 + w$	$w^2 + 2w$	$2w$ (2)	F_3	L_1	τ_3
Proposed	w	$3w^2 + 2w$	$3w^2$	$2w$	$4w(\lceil m/w \rceil + 1)$	$\lceil m/w \rceil^2 + 1$	τ_4

(1) Area of 3-input XOR gate as 1.5× a 2-input XOR gate. (2) Multiplier of [41] uses switches that having same transistor count as 2-input MUX.

For fair comparison, we added the area complexity of Input/Output registers for each design structure.

By inspecting Table 1, we observe that the expressions representing the estimated number of logic gates or components of the multiplier structures of Pan et al. [18] and Xie et al. [23] are approximate of order $\mathcal{O}(mw)$. On the other hand, it is of order $\mathcal{O}(w^2)$ for the other multiplier structures, except the MUXes and Flip-Flop components of the proposed design, which are of order $\mathcal{O}(w^2)$ and $\mathcal{O}(\lceil m/w \rceil)$, respectively. Since m is extremely larger than w, we can conclude that the area complexity of the multiplier structures of Pan et al. [18] and Xie et al. [23] will be higher than that of all the other multiplier

structures, including the suggested one. By examination of the gate counts' expressions of the developed multiplier-squarer and multipliers of Hua et al. [40] and Chen et al. [41], we recognize that the proposed design has a lower number of Flip-Flops compared to them. The flip-flop area expression is of order $\mathcal{O}(\lceil m/w \rceil)$ for the suggested multiplier-squarer and it is of order $\mathcal{O}(w^2)$ for the multipliers of Hua et al. [40] and Chen et al. [41]. Therefore, for large values of w, the number of flip-flops will be substantially decrease compared to the other multipliers. According to the standard CMOS libraries' data, the Flip-Flop consumes the largest area on the chip compared to the other gate types. Thus, reducing the number of flip-flops in the design structure will considerably reduce the overall area, which accounts for the insignificant increase in the area of the proposed design as w increases. It is interesting to notice that the suggested design performs multiplication and squaring operations simultaneously and the compared ones perform both operations in sequence and, hence, reducing the area of the developed design for different word sizes can be considered a considerable achievement.

By examining the latency expressions in Table 1, we can conclude that the multiplier structure of Pan et al. [18] has lower latency and that the multiplier structure of Hua et al. [40] has the most significant latency compared to the remaining multiplier structures, including the recommended one. The numerical results displayed in Table 2 show that the proposed design has lower latency than the designs of Hua et al. [40] and Chen et al. [41] and higher latency than the multiplier designs of Xie et al. [23] and Pan et al. [18] for the common field size $n = 409$. Furthermore, we can conclude from Table 1 that the latency of all multiplier structures typically decreases as word-size w increases as it is inversely proportional to w.

By analyzing the expressions of the CPD, for all word sizes of w, we can conclude that the multiplier structures of Xie et al. [23], Hua et al. [40], and Chen et al. [41] have a fixed and lower CPD. On the other hand, the CPD values of Pan et al. [18] and the developed design principally increases as w increases. We cannot accurately assume, from the estimated expressions, which design structure has the best execution time due to the difficulty in estimating the decreased amount of latency when w increases. The numerical results given in Table 2 will confirm the question of which multiplier structure presents better execution time complexity.

The designs in Table 1 are described using the VHDL code and synthesized for the common field size $m = 409$ and different values of w $(8, 16, 32)$ to obtain real implementation results. We used the NanGate (15 nm, 0.8 V) Open Cell Library and Synopsys tools version 2005.09-SP2 for synthesizing. We used the typical corner (V_{DD} = 0.8 V and $T_j = 25$ °C) and unit drive strength for all the utilized primitives.

Testing the proposed design starts by evaluating the wasted power at a frequency of 1 KHz for each multiplier structure. Then, through simulation using Mentor Graphics ModelSim SE 6.0a tools, we accumulated the switching activities in the Switching Activity Interchange Format (SAIF) file to obtain the power report. Next, we designed a testbench to simulate the suggested multiplier-squarer structure. The test bench has a single loop of 400 possible input combinations of 32-bits, each allowing the user to validate the correctness of the outputs. In order to regularly examine the resulting output's correctness, we used an error flag to designate if the implemented design is working accurately or not. If the error flag sets to '0' at the end of the simulation, then the multiplier-squarer structure works perfectly. On the other hand, if it sets to '1', the multiplier-squarer design is not operating correctly. In order to allow the examiner to examine the generated output from each input set, we utilized a "wait statement" to produce a delay of 50 ns between test vectors. Furthermore, we performed a post-layout simulation to include the additional pin cost and the propagation delay of all gates. Accordingly, we can achieve an accurate evaluation of the area, time, and consumed power.

Table 2. Implementation results of different word-serial field multipliers for $m = 409$ and different values of w.

Multiplier	w	Latency	Area (A) (*Kgates*)	CPD (*ps*)	Time (T) (*ns*)	Power (P) (*nW*)	Energy (E) (*fJ*)
Xie [23]	8	324	92.98	56.4	18.27	225.56	4.12
	16	172	146.96	56.4	9.70	375.5	3.64
	32	98	195.13	56.4	5.53	477.4	2.64
Pan [18]	8	48	97.46	206.3	9.90	252.91	2.50
	16	36	123.93	244.4	8.80	320.07	2.82
	32	24	164.34	282.5	6.78	425.09	2.88
Hua [40]	8	259,584	7.99	73.4	19053.47	4.35	82.88
	16	129,792	10.40	73.4	9526.73	5.85	55.73
	32	64896	19.91	73.4	4763.37	11.15	53.11
Chen [41]	8	11946	10.16	55.2	659.42	5.11	3.37
	16	3678	13.51	55.2	203.03	8.38	1.70
	32	1572	26.58	55.2	86.77	15.95	1.38
Proposed	8	2705	7.26	215.7	583.47	3.88	2.26
	16	677	9.19	407.7	276.01	5.12	1.41
	32	170	15.78	791.7	134.59	7.28	0.98

The obtained results are listed in Table 2. The design metrics used to compare the proposed and the existing word-serial designs can be defined as follows:

1. Latency: is the total number of clock cycles needed to complete a single operation;
2. Area (A): is the estimated design area in terms of the equivalent area of 2-input NAND gate;
3. CPD: is the synthesized critical path delay;
4. Time (T): is the total computation time required to complete a single operation;
5. Power (P): is the consumed power obtained at 1 KHZ;
6. Energy (E): is the consumed energy which obtained by multiplying power (P) by the total computation time (T).

For a fair comparison, the compared multiplier structures of [18,23,40,41] should perform multiplication and squaring operations in sequence and this doubles the obtained synthesis results of the time and consumed power/energy of these designs as indicated in Table 2. For a better explanation of the obtained results, we visualized area, time, power, and energy results by using the charts shown in Figures 9–12, respectively.

Figure 9 indicates that the proposed design structure saves area at the different values of w by percentages ranging from 9.1% to 92.6% at $w = 8$, 11.6% to 93.7% at $w = 16$, and 20.7% to 91.9% at $w = 32$ over the existing designs. The design of Pan [18] saves 45.8% and 9.3% time at $w = 8$ and $w = 16$, respectively, over the best of the other designs including the proposed one. The design of Xie [23] saves 18.4% time at $w = 32$ over the best of the other designs including the proposed one.

Figure 10 indicates that the multiplier of Pan [18] has the most reduced computation time at $w = 8$ over all the remaining designs, including the recommended one (at least 40% lower time than the multiplier of Xie [23]). At $w = 16$ and $w = 32$, the multiplier of Xie [23] has the cheapest computation time over the remaining designs (at least reduction by 0.6% at $w = 16$ and 6.5% at $w = 32$ over the multiplier of Pan [18]). The multiplier of Xie [23] outperforms the remaining designs at $w = 16$ and $w = 32$ due to the significant reduction in its latency compared to the other multiplier designs.

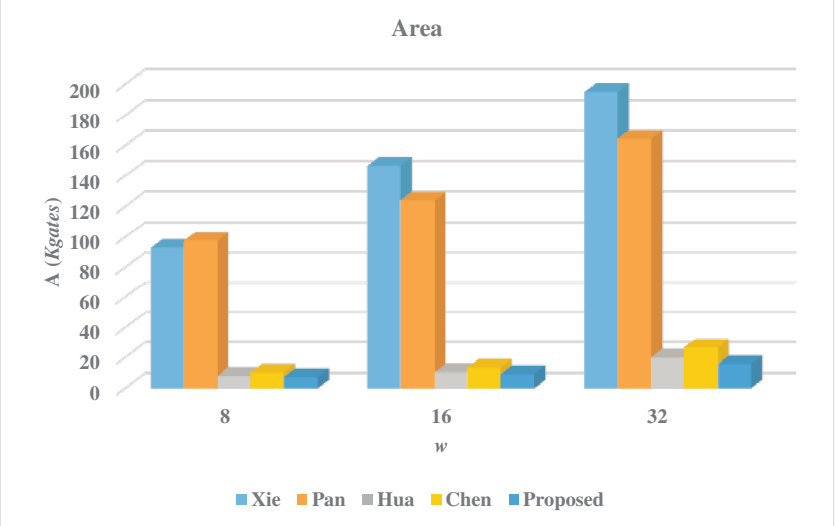

Figure 9. Comparison of area results.

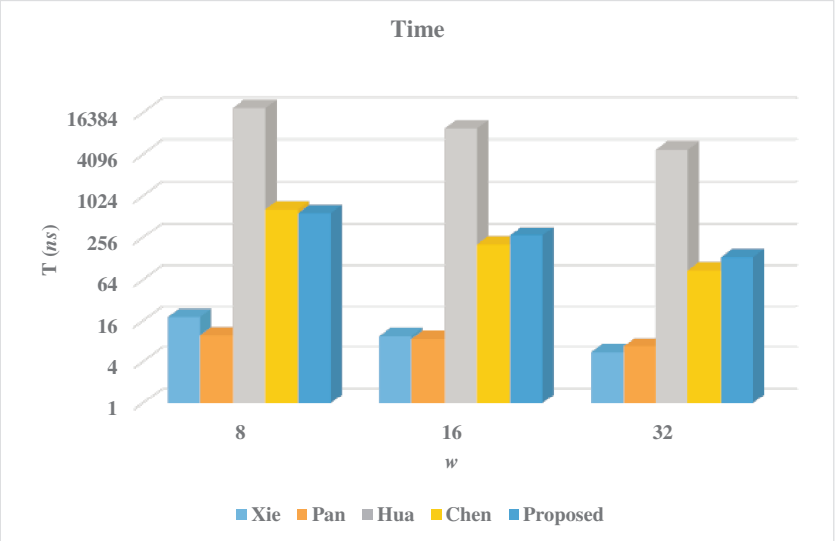

Figure 10. Comparison of time results.

Figure 11 indicates the developed design that has the lowest power consumption at all word sizes due to its significant reduction in area. The savings in the area will reduce the parasitic capacitance and, thus, reduces the switching activities in the entire circuit, which is one of the main contributors to power consumption. We noticed from Table 2 that the proposed design reduces power consumption at different word sizes by percentages ranging from 10.8% to 98.5% at $w = 8$, 12.5% to 98.4% at $w = 16$, and 34.7% to 98.5% at $w = 32$ over the existing designs.

Figure 12 indicates that the proposed design structure has the a significant reduction in energy over all the remaining multipliers at the different values of w. By observing the energy results in Table 2, we notice that the proposed design saves energy at the different values of w by percentages ranging from 9.6% to 97.3% at $w = 8$, 17.1% to 97.5% at $w = 16$, and 29.0% to 98.2% at $w = 32$ over the existing designs. The reduction in consumed energy of the proposed design, at all word sizes, is mainly attributed to the fair values of its execution time (T) and the lower values of the consumed power.

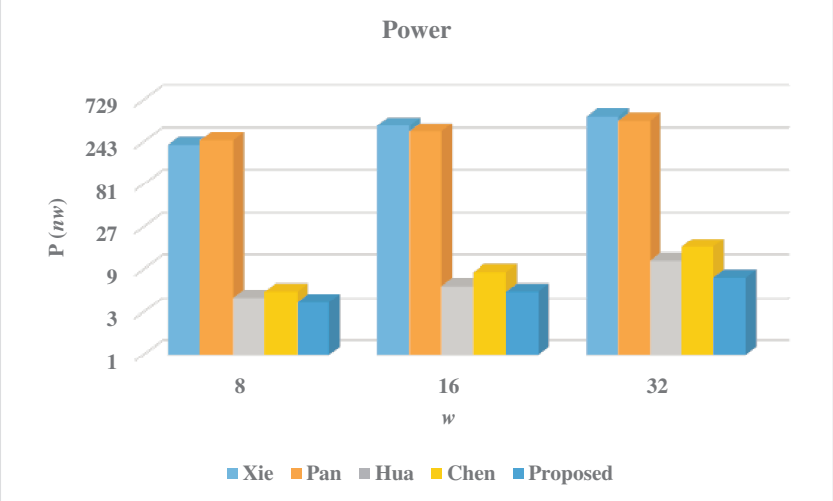

Figure 11. Comparison of consumed power results.

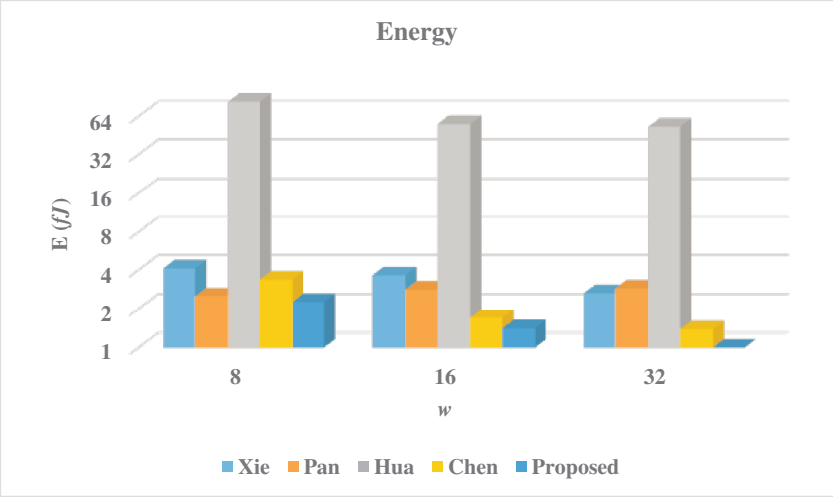

Figure 12. Comparison of consumed energy results.

As we notice, the proposed design has the lower area, power, and consumed energy at all of the embedded word sizes. Thus, it enables the implementation of cryptographic processors in resource-constrained IoT edge devices such as hand-held devices, wearable and implantable medical devices, wireless sensor nodes, smart cards, and radio frequency identification (RFID) devices.

6. Summary and Conclusions

This paper presented new efficient two-dimensional word-based SISO systolic processor for performing the multiplication and squaring operations concurrently over $GF(2^m)$. The proposed systolic processor structure shares the data-path and this results in saving more area and power resources. We applied non-linear scheduling and projection functions to the algorithm dependency graph to explore the proposed systolic processor core. The applied non-linear scheduling and projection functions provide the designer more flexibility to control the processor work load and the execution time. The size of the systolic array in the processor core does not depend on the field size and that renders the proposed design more suitable for implementation in embedded and ultra-low power devices. Implementation results of the proposed two-dimensional combined word-serial processor systolic structure and the best of the existing word-serial multiplication designs show that the proposed structure achieves significant savings in area and consumed energy at different values of the embedded word sizes. This renders it more suitable for constrained

implementations of cryptographic primitives in resource-constrained IoT edge devices such as hand-held devices, wearable and implantable medical devices, wireless sensor nodes, smart cards, and radio frequency identification (RFID) devices.

Author Contributions: Conceptualization, A.I. and F.G.; methodology, A.I. and F.G.; software, A.I.; validation, A.I. and F.G.; formal analysis, A.I.; investigation, A.I.; resources, A.I.; data curation, A.I.; writing—original draft preparation, A.I.; writing—review and editing, A.I. and F.G.; visualization, A.I. and F.G.; supervision, A.I.; project administration, A.I. and F.G.; funding acquisition, F.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the NATIONAL RESEARCH COUNCIL OF CANADA (NRC) grant number 51291-52200.

Acknowledgments: The authors would like to acknowledge the financial support of the National Research Council of Canada (NRC) grant under the Collaborative R&D Initiative.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things;
RFID	Radio Frequency Identification;
ASIC	Application Specific Integrated Circuit;
RSA	Rivest, Shamir, and Adleman;
DSA	Digital Signature Algorithm;
ECC	Elliptic Curve Cryptography;
SISO	Serial-In/Serial-Out;
SIPO	Serial-In/Parallel-Out;
PISO	Parallel-In/Serial-Out;
RIA	Regular Iterative Algorithm;
CPD	Critical Path Delay.

References

1. Chen, D.; Zhang, N.; Qin, Z.; Mao, X.; Qin, Z.; Shen, X.; Li, X.Y. S2M: A lightweight acoustic fingerprints-based wireless device authentication protocol. *IEEE Internet Things J.* **2016**, *4*, 88–100. [CrossRef]
2. Sowjanya, K.; Dasgupta, M.; Ray, S. An elliptic curve cryptography based enhanced anonymous authentication protocol for wearable health monitoring systems. *Int. J. Inf. Secur.* **2020**, *19*, 129–146. [CrossRef]
3. Granjal, J.; Monteiro, E.; Silva, J.S. Security for the internet of things: a survey of existing protocols and open research issues. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1294–1312. [CrossRef]
4. Safkhani, M.; Vasilakos, A. A new secure authentication protocol for telecare medicine information system and smart campus. *IEEE Access* **2019**, *7*, 23514–23526. [CrossRef]
5. Aghili, S.F.; Mala, H.; Kaliyar, P.; Conti, M. Seclap: Secure and lightweight RFID authentication protocol for medical IoT. *Future Gener. Comput. Syst.* **2019**, *101*, 621–634. [CrossRef]
6. Anajemba, J.H.; Iwendi, C.; Mittal, M.; Yue, T. Improved advance encryption standard with a privacy database structure for IoT nodes. In Proceedings of the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 10–12 April 2020; pp. 201–206.
7. Anajemba, J.H.; Yue, T.; Iwendi, C.; Alenezi, M.; Mittal, M. Optimal cooperative offloading scheme for energy efficient multi-access edge computation. *IEEE Access* **2020**, *8*, 53931–53941. [CrossRef]
8. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
9. Rivest, R.L.; Shamir, A.; Adleman, L. A method for obtaining digital signatures and public-key cryptosystems. *Mag. Commun. ACM* **1978**, *21*, 120–126. [CrossRef]
10. Lidl, R.; Niederreiter, H. *Introduction to Finite Fields and Their Applications*; Cambridge University Press: Cambridge, UK, 1994.
11. Chiou, C.W.; Lee, C.Y.; Deng, A.W.; Lin, J.M. Concurrent error detection in Montgomery multiplication over GF(2^m). *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2006**, *E89-A*, 566–574. [CrossRef]
12. Kim, K.W.; Jeon, J.C. Polynomial Basis Multiplier Using Cellular Systolic Architecture. *IETE J. Res.* **2014**, *60*, 194–199. [CrossRef]
13. Choi, S.; Lee, K. Efficient ssystolic modular multiplier/squarer for fast exponentiation over GF(2^m). *IEICE Electron. Express* **2015**, *12*, 1–6. [CrossRef]
14. Kim, K.W.; Kim, S.H. Efficient bit-parallel systolic architecture for multiplication and squaring over GF(2^m). *IEICE Electron. Express* **2018**, *15*, 1–6. [CrossRef]

15. Kim, C.H.; Hong, C.P.; Kwon, S. A digit-serial multiplier for finite Field $GF(2^m)$. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2005**, *13*, 476–483.
16. Talapatra, S.; Rahaman, H.; Mathew, J. Low complexity digit serial systolic montgomery multipliers for special class of GF(2^m). *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2010**, *18*, 847–852. [CrossRef]
17. Guo, J. H.; Wang, C. L. Hardware-efficient Systolic Architecture for Inversion and Division in GF(2^m). *IEE Proc. Comput. Digit. Tech.* **1998**, *145*, 272–278. [CrossRef]
18. Pan, J.S.; Lee, C.Y.; Meher, P.K. Low-Latency Digit-Serial and Digit-Parallel Systolic Multipliers for Large Binary Extension Fields. *IEEE Trans. Circ. Syst.-I* **2013**, *60*, 3195–3204. [CrossRef]
19. Lee, C. Y.; Fan, C. C.; Yuan, S. M. New Digit-Serial Three-Operand Multiplier over Binary Extension Fields for High-Performance Applications. In Proceedings of the 2017 2nd IEEE International Conference on Computational Intelligence and Applications, Beijing, China, 8–11 September 2017; pp. 498–502
20. Hariri, A.; Reyhani-Masoleh, A. Digit-serial structures for the shifted polynomial basis multiplication over binary extension fields. In *Proc. LNCS Intl Workshop Arithmetic of Finite Fields (WAIFI)*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 103–116
21. Kumar, S.; Wollinger, T.; Paar, C. Optimum digit serial multipliers for curve-based cryptography. *IEEE Trans. Comput.* **2006**, *55*, 1306–1311. [CrossRef]
22. Lee, C.Y. Super digit-serial systolic multiplier over GF(2^m). In Proceedings of the 6th International Conference Genetic Evolutionary Computing, Kitakyushu, Japan, 25–28 August 2012; pp. 509–513
23. Xie, J.; Meher, P.K.; Mao, Z. Low-latency high-throughput systolic multipliers over GF(2^m) for NIST recommended pentanomials. *IEEE Trans. Circuits Syst.* **2015**, *62*, 881–890. [CrossRef]
24. Namin, A.H.; Wu, H.; Ahmadi, M. A word-level finite field multiplier using normal basis. *IEEE Trans. Comput.* **2011**, *60*, 890–895. [CrossRef]
25. Lee, C.Y.; Chiou, C.W.; Lin, J.M.; Chang, C.C. Scalable and systolic Montgomery multiplier over generated by trinomials. *IET Circuits Devices Syst.* **2007**, *1*, 477–484. [CrossRef]
26. Chen, L.H.; Chang, P.L.; Lee, C.Y.; Yang, Y.K. Scalable and systolic dual basis multiplier Over GF(2^m). *Int. J. Innov. Comput. Inf. Control* **2011**, *7*, 1193–1208.
27. Orlando, G.; Paar, C. A super-serial galois fields multiplier for FPGAs and its application to public-key algorithms. In Proceedings of the IEEE Symposium Field-Programmable Custom Computing, Napa Valley, CA, USA, 23 April 1999; pp. 232–239
28. Bayat-Sarmadi, S.; Kermani, M.M.; Azarderakhsh, R.; Lee, C.Y. Dual Basis Super-Serial Mult. for Secure Applications and Lightweight Cryptographic Arch. *IEEE Trans. Circ. Syst.-II* **2014**, *61*, 125–129.
29. Gebali, F.; Ibrahim, A. Efficient Scalable Serial Multiplier Over GF(2^m) Based on Trinomial. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 2322–2326. [CrossRef]
30. Ibrahim, A.; Gebali, F.; El-Simary, H.; Nassar, A. High-performance, low-power architecture for scalable radix 2 Montgomery modular multiplication algorithm. *IEEE Can. J. Electr. Comput. Eng.* **2009**, *34*, 152–157. [CrossRef]
31. Ibrahim, A.; Gebali, F. Scalable and Unified Digit-Serial Processor Array Architecture for Multiplication and Inversion over $GF(2^m)$. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *22*, 2894–2906. [CrossRef]
32. Kim, K.W.; Lee, J.D. Efficient unified semi-systolic arrays for multiplication and squaring over GF(2^m). *IEICE Electron. Express* **2017**, *14*, 1–10. [CrossRef]
33. Gebali, F. *Algorithms and Parallel Computers*; John Wiley: New York, NY, USA, 2011.
34. Ibrahim, A.; Elsimary, H.; Gebali, F. New systolic array architecture for finite field division. *IEICE Electronics Express* **2018**, *15*, 1–11. [CrossRef]
35. Ibrahim, A.; Elsimary, H.; Aljumah, A.; Gebali, F. Reconfigurable hardware accelerator for profile hidden Markov models. *Arabian J. Sci. Eng.* **2016**, *41*, 3267–3277. [CrossRef]
36. Ibrahim, A. Scalable digit-serial processor array architecture for finite field division. *Microelectron. J.* **2019**, *85*, 83–91. [CrossRef]
37. Ibrahim, A.; Alsomani, T.; Gebali, F. Unified Systolic Array Architecture for Field Multiplication and Inversion Over GF(2^m). *Comput. Electr. Eng. J.-Elsevier* **2017**, *61*, 104–115. [CrossRef]
38. Ibrahim, A.; Alsomani, T.; Gebali, F. New Systolic Array Architecture for Finite Field Inversion. *IEEE Can. J. Electr. Comput. Eng.* **2017**, *40*, 23–30.
39. Gebali, F.; Ibrahim, A. Low space-complexity and low power semi-systolic multiplier architectures over GF(2^m) based on irreducible trinomial. *Microprocess. Microsyst.* **2016**, *40*, 45–52. [CrossRef]
40. Hua, Y.Y.; Lin, J.M.; Chiou, C.W.; Lee, C.Y.; Liu, Y.H. Low space-complexity digit-serial dual basis systolic multiplier over Galois field GF (2^m) using Hankel matrix and Karatsuba algorithm. *IET Inf. Secur.* **2013**, *7*, 75–86.
41. Chen, C. C.; Lee, C. Y.; Lu, E. H. Scalable and Systolic Montgomery Multipliers Over GF(2^m). *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2008**, *E91-A*, 1763–1771. [CrossRef]