mathematics

Article

Low-Complexity One-Dimensional Parallel Semi-Systolic Structure for Field Montgomery Multiplication Algorithm Perfect for Small IoT Edge Nodes

Atef Ibrahim [1,2,*], Usman Tariq [3], Tariq Ahamed Ahanger [3] 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 Victoria, Victoria, BC V8P 5C2, Canada
[3] College of Business Administration, Prince Sattam bin Abdulaziz University, Al-Kharj 16278, Saudi Arabia
* Correspondence: aa.mohamed@psau.edu.sa

Abstract: The use of IoT technology in several applications is hampered by security and privacy concerns with IoT edge nodes. Security flaws can only be resolved by implementing cryptographic protocols on these nodes. The resource constraints of the edge nodes make it extremely difficult to implement these protocols. The majority of cryptographic protocols' fundamental operation is finite-field multiplication, and their performance is significantly impacted by their effective implementation. Therefore, this work mainly focuses on implementing low-area with low-energy and high-speed one-dimensional bit-parallel semi-systolic multiplier for the Montgomery multiplication algorithm. The space and delay complexity analysis of the proposed multiplier structure reveals that the proposed design has a significant reduction in delay and a marginal reduction in the area when compared to the competitive one-dimensional multipliers. The obtained ASIC synthesis report demonstrates that the suggested multiplier architecture saves a marginal amount of space as well as a significant amount of time, area–delay product (ADP), and power–delay product (PDP) when compared to the competitive ones. The obtained results indicate that the proposed multiplier layout is very appropriate for use in devices with limited resources such as IoT edge nodes and tiny embedded devices.

Keywords: modular arithmetic; cryptography; ubiquitous computing; security; IoT applications; systolic and semi-systolic multipliers; cyber-physical systems; embedded systems

MSC: 11T71

Citation: Ibrahim, A.; Tariq, U.; Ahanger, T.A.; Gebali, F. Low-Complexity One-Dimensional Parallel Semi-Systolic Structure for Field Montgomery Multiplication Algorithm Perfect for Small IoT Edge Nodes. *Mathematics* **2023**, *11*, 111. https://doi.org/10.3390/math11010111

Academic Editors: Zijian Zhang, Liehuang Zhu and Meng Li

Received: 11 November 2022
Revised: 13 December 2022
Accepted: 14 December 2022
Published: 26 December 2022

1. Introduction

Our daily lives are currently greatly impacted by the Internet of Things. They can be applied to a variety of industries, including healthcare, transportation, entertainment, commercial appliances, agriculture, and housing. The primary objective of the IoT network is to gather data and send it to the cloud for additional analysis and decision-making. Since the majority of IoT applications are sensitive, the data gathered by IoT devices need to be secured at all IoT network layers. Implementing security mechanisms on most IoT edge nodes is difficult because of their constrained resource availability. As a result, numerous attempts have been made to find a solution to this difficult issue. There are many security protocols available that can be implemented on IoT edge nodes with limited resources. Elliptic Curve-Cryptography (ECC), among other cryptographic algorithms, are optimized for use on these nodes. Finite-field arithmetic operations, specifically finite-field multiplication, are the main foundation of the majority of optimized algorithms. In addition, finite-field multiplication is the fundamental operation for all other field operations, including inversion, division, and exponentiation [1]. As a result, it has given a great interest in supporting the implementation of small and extremely effective cryptographic algorithms [2–12].

1.1. Literature Review

The basis representations chosen for elements in GF(2^m) have a great effect on the efficiency of finite field multiplications. There are various basis representations, including polynomial basis (PB), normal basis (NB), dual basis (DB), and redundant basis (RB) [13]. Each foundation has unique advantages of its own. Among these bases, polynomial basis arithmetic is the most straightforward, regular, and scalable in hardware implementation [14–18]. In addition, it does not require a basis conversion like other ones. Therefore, it has wide employment in a variety of cryptographic protocols. The irreducible polynomial that is selected also affects the finite field multiplier's efficiency. Generic polynomials, All-One polynomials (AOP), trinomials, and pentanomials are different types of irreducible polynomials used in cryptographic algorithms. Generic polynomial-based multipliers are appropriate for a wider range of applications, but trinomial and pentanomial-based multipliers are more efficient. Although irreducible AOPs are less common than irreducible trinomials or pentanomials, they can be used to implement efficient multipliers [19–21].

Different multipliers can be created depending on the method of implementation. Bit-serial multipliers are space-efficient and have significant savings in power consumption, but they are slow and require m clock cycles to multiply two elements [2,22,23]. In contrast, bit-parallel multipliers require excessive hardware cost and high power consumption, but produce the result in one clock cycle [4,5,8,24–29]. Systolic/semi-systolic serial or parallel multiplier architectures are more suitable for VLSI implementation than the other types of conventional implementations. This is attributed to their defined characteristics of regularity, modularity, local relatively homogenous interconnection, and concurrency. In addition, the pipeline's inherent characteristics of the systolic/semi-systolic arrays allow for the use of a high clock frequency even when a lot of resources are being used.

Many authors in the literature have attempted to provide effective implementations of the systolic/semi-systolic multipliers over the binary extension field GF(2^m). Most of them made an effort to construct their structures using a particular irreducible polynomial. An error-detecting semi-systolic array multiplier was presented by Lee et al. [30] and Chiou et al. [2]. An effective semi-systolic array multiplier was suggested by Huang et al. [3] to reduce the time and space costs. For unified multiplication and squaring with minimal hardware overhead, Choi and Lee [5] created a highly area-time efficient serial and parallel systolic array. This makes fast modular exponentiation possible as it has the feature of computing multiplication and squaring simultaneously. The proposed LSB-first multiplication and exponentiation algorithms reinforce the performance of this architecture. A Chiou et al. [31] semi-systolic array multiplier was presented to reduce the time complexity. In a recent work, Lee [32,33] proposed new semi-systolic Montgomery Modular multipliers with two levels of systolic computation. The proposed multiplier structures are efficient in area and delay. A parallel and serial input are both available for the multiplier introduced by Mathe and Boppana [34]. Ibrahim [12] introduced an efficient one-dimensional bit-serial and bit-parallel systolic array structures to perform both multiplication and squaring operations over GF(2^m). The suggested trinomials for field sizes $m = 233$ and $m = 409$ are included in a recent GF(2^m) polynomial basis systolic multiplier proposed by Pillutla and Boppana [16]. GF(2^m)-based multipliers have been developed for a number of applications, but their high hardware complexity and lengthy delay times are significant drawbacks for security applications. Thus, there is a need for additional study on effective multiplication architectures with minimal space and time requirements.

The equally spaced and AOP polynomials were used as the foundation for bit-parallel's systolic multiplier structure proposed by Lee et al. [24,25]. In 2005, Lee et al. [26] suggested a mapping approach in order to reduce the complexity of the AOP-based bit-parallel systolic multiplier. The mapping approach changed the multiplier's foundation from AOP to trinomials. The stated Montgomery-based bit-parallel multiplier's complexity was lowered by applying the Toeplitz matrix-vector representation suggested by Lee et al. [27]. Sarmadi [28] offered a low space with high performance two-dimensional parallel systolic multiplier that is based on Montgomery algorithm. Mathe [29] adopted an interleaving

multiplication algorithm over GF(2^m) to implement a two-dimensional parallel systolic multiplier layout with low space complexity.

1.2. Paper Contribution

This work develops a one-dimensional bit-parallel semi-systolic implementation of the field Montgomery multiplication algorithm suggested by [33]. The offered algorithm performs the multiplication operation over GF(2^m) and uses as a base the general irreducible polynomial. In contrast to many other algorithms, the adopted Montgomery algorithm has the advantage of reducing latency. In addition, it has the advantage of reducing the time and area overhead by using the same architecture to execute its two iterative parts [33]. Previous works in the literature used ad hoc approaches to extract the hardware structure with no thought given to how the structure might be altered to improve system performance factors such as latency, throughput, power, and area. In this work, we offer a mathematical approach for obtaining the proposed multiplier structure. The proposed approach has the advantage of selecting appropriate scheduling and projection functions to extract the optimal architecture that suits the required application.

To be able to extract the dependency graph (DG) for the adopted multiplication algorithm, we presented it in the bit-level form. By selecting the appropriate time-scheduling and node-projection functions, the DG will assist us in extracting the proposed low complexity one-dimensional bit-parallel semi-systolic multiplier structure. The proposed multiplier structure differs from the majority of those previously reported two-dimensional ones in that it exhibits area complexity of order $\mathcal{O}(m)$ as opposed to order $\mathcal{O}(m^2)$ for the majority of those structures. As a result, the suggested multiplier structure significantly reduces both the complexity of the physical space and the amount of consumed power. The performance of the suggested multiplier structure is unaffected by the area reduction because it exhibits the same timing delays as the two-dimensional ones. Furthermore, the modular structure and local connectivity between the constituting PEs of the proposed multiplier structure make it more suitable for VLSI implementation. In addition, local interconnection between the PEs improves the multiplier structure's overall performance by reducing wire delays. The suggested multiplier structure is better suited for use in tiny embedded devices or IoT edge nodes due to the significant space and power savings it offers.

1.3. Paper Organization

Following is a summary of the paper's arrangement: The mathematical modeling of the chosen Montgomery multiplication algorithm is presented, along with its bit-level representation, in Section 2. The produced DG of the adopted algorithm is described in Section 3. The process for obtaining the suggested one-dimensional bit-parallel semi-systolic multiplier layout is described in Section 4. The analysis of space and time complexities of the suggested multiplier and the currently used effective multipliers is shown in Section 5. Additionally, this section offers a real assessment of the suggested multiplier design's performance and competing one-dimensional multiplier designs based on ASIC synthesis. Section 6 provides conclusions for the offered work.

2. Montgomery Multiplication in GF(2^m)

Suppose the irreducible polynomial creating the finite field GF(2^m) is $F = \sum_{j=0}^{m} f_j \cdot \alpha^j$, where $f_m = f_0 = 1$ for $1 \leq j \leq m-1$. Each component of GF(2^m) is a distinct linear combination made up of polynomials with degrees less than m. Adding two polynomials in GF(2^m) can easily be performed using bitwise exclusive-OR (XOR). On the other hand, multiplying two polynomials in GF(2^m) is a little more challenging because the intermediate result requires additional modular reduction by $\alpha^m = \sum_{j=0}^{m-1} f_i \cdot \alpha^i$.

Assume ζ and γ are two of the GF(2^m) elements that will be multiplied. In addition, assume C and D are the Montgomery residues of ζ and γ, respectively, and R is a special element satisfying $gcd(R, G) = 1$. The Montgomery Modular Multiplication (MMM) of $C = \zeta R \bmod F = \sum_{j=0}^{m-1} c_j \cdot \alpha^j$ and $D = \gamma\ R \bmod F = \sum_{j=0}^{m-1} d_j \cdot \alpha^j$ is computed by

$P = CDR^{-1} \bmod F = \gamma R \bmod F = \sum_{j=0}^{m-1} p_j \cdot \alpha^j$. The final result T is then obtained by computing Montgomery multiplication using inputs P and 1, i.e., $T = PR^{-1} \bmod F = \zeta\ \gamma \bmod F$. Because of the requirements for pre- and post-transformation, Montgomery multiplication is advantageous in many applications utilizing repeated multiplications, such as inversion, exponentiation, and elliptic curve point multiplication [32].

The Montgomery multiplication, $P = CDR^{-1} \bmod F$, can be expressed as follows using $R = \alpha^{(m-1)/2}$ [33]:

$$\begin{aligned} P =& C(d_0 + d_1\alpha + \cdots + d_{(m-1)/2}\alpha^{(m-1)/2} + \\ & \cdots + d_{m-1}\alpha^{m-1})\alpha^{-(m-1)/2} \bmod F \end{aligned}$$

We can arrange Equation (1) to be as follows:

$$\begin{aligned} P =& C(d_{(m-1)/2} + d_{(m+1)/2}\alpha^1 + \cdots + d_{(m-1)}\alpha^{(m-1)/2}) \\ & + C(d_0\alpha^{-(m-1)/2} + d_1\alpha^{-(m-3)/2} + \\ & \cdots + d_{(m-3)/2}\alpha^{-1}) \bmod F \end{aligned} \tag{1}$$

It is worth noting that the most practical applications employ odd m. As a result, we will focus on using odd m when designing the multiplier.

Equation (1) can be represented as the summation of two polynomials A and B as:

$$\begin{aligned} A =& Cd_{(m-1)}\alpha^{(m-1)/2} + Cd_{(m-2)}\alpha^{(m-3)/2} + \cdots \\ & + Cd_{(m+1)/2}\alpha^1 + Cd_{(m-1)/2} \bmod F \end{aligned} \tag{2}$$

$$\begin{aligned} B =& Cd_0\alpha^{-(m-1)/2} + Cd_1\alpha^{-(m-3)/2} + \cdots \\ & + Cd_{(m-5)/2}\alpha^{-2} + Cd_{(m-3)/2}\alpha^{-1} \bmod F \end{aligned} \tag{3}$$

We can arrange Equations (2) and (3) as follows to be able to derive their recurrence forms:

$$\begin{aligned} A =& (\cdots((Cd_{(m-1)})\alpha \bmod F + Cd_{(m-2)})\alpha \bmod F + \cdots + \\ & Cd_{(m+1)/2})\alpha \bmod F + Cd_{(m-1)/2} \end{aligned} \tag{4}$$

$$\begin{aligned} B =& (\cdots(((Cd_0)\alpha^{-1} \bmod F + Cd_1)\alpha^{-1} \bmod F + \cdots + \\ & Cd_{(m-5)/2}\alpha^{-1} \bmod F + Cd_{(m-3)/2})\alpha^{-1} \bmod F \end{aligned} \tag{5}$$

Suppose A^i and B^i are the outcomes of the (i)th iteration of Equations (4) and (5), respectively, that can be calculated iteratively from the outcomes of $(i-1)$th pair of iterators. The iterative equation of (4) at step i for $1 \leq i \leq (m+1)/2$ can be written as:

$$A^i = A^{i-1}\alpha \bmod F + Cd_{(m-i)} \tag{6}$$

where $A^0 = 0$.

Equation (5) can be expressed recursively in the same way as Equation (4):

$$B^i = B^{i-1}\alpha^{-1} \bmod F + Cd_{(i-1)} \tag{7}$$

where $B^0 = d_{(m-1)/2} = 0$.

It should be noted that, in order to compute the final $B^{(m+1)/2}$, the value $d_{(m-1)/2} = 0$ is necessary. There is no data dependency between A^i and B^i, so they can be computed concurrently, as shown by Equations (6) and (7). The reduced form of A^i for $1 \leq i \leq$

$(m+1)/2$ can be obtained using the bit-level representation by replacing the expansion of α^m in Equation (6). The resulting bit-level expression of A^i can be given as follows:

$$\begin{aligned} A^i =& a_{m-2}^{i-1}\alpha^{m-1} + \cdots + a_1^{i-1}\alpha^2 + a_0^{i-1}\alpha + \\ & a_{m-1}^{i-1}(f_{m-1}\alpha^{m-1} + \cdots + f_1\alpha + f_0) + \\ & d_{m-i}(c_{m-1}\alpha^{m-1} + \cdots + c_1\alpha + c_0) \end{aligned} \tag{8}$$

The recursive representation of A at step i can be expressed as follows:

$$a_{m-1-j}^i = a_{m-2-j}^{i-1} + a_{m-1}^{i-1} f_{m-1-j} + d_{m-i}c_{m-1-j} \tag{9}$$

where $a_j^0 = a_{-1}^{i-1} = 0$ and $0 \le j \le m-1$.

Since α is a root of F and $f_0 = f_m = 1$ for any irreducible polynomial, we can obtain $\alpha^{-1} = \sum_{j=1}^{m} f_j \alpha^{j-1}$ by multiplying each side of F by α^{-1}.

By replacing the expansion of α^{-1} on Equation (7), B_i can be rewritten similarly to Equation (8) as follows:

$$\begin{aligned} B^i =& b_{m-1}^{i-1}\alpha^{m-2} + \cdots + b_1^{i-1} + b_0^{i-1}(f_m\alpha^{m-1} + \cdots + \\ & f_2\alpha + f_1) + d_{i-1}(c_{m-1}\alpha^{m-1} + \cdots + c_1\alpha + c_0) \end{aligned} \tag{10}$$

The recursive representation of B at step i can be expressed as follows:

$$b_j^i = b_{j+1}^{i-1} + b_0^{i-1} f_{j+1} + d_{i-1}c_j \tag{11}$$

where $b_j^0 = b_m^{i-1} = d_{(m-1)/2} = 0$ for $0 \le j \le m-1$.

Finally, m 2-input XOR gates should be used to add $A^{(m+1)/2}$ and $B^{(m+1)/2}$ to obtain the final product P.

The algorithm structure of the previously described formulas is represented by Algorithms 1 and 2. Algorithm 2 is the bit-level variant of Algorithm 1.

Algorithm 1 Montgomery Multiplication Algorithm in GF(2^m)

Input: C, D, $R^{-1} = \alpha^{-(m-1)/2}$, and F
Output: P
Initialization:
$A^0 \leftarrow 0,\ B^0 \leftarrow 0$
Algorithm:
1: **for** $1 \le i \le (m+1)/2$ **do**
2: $\quad A^i = A^{i-1}\alpha \bmod F + Cd_{(m-i)}$
3: $\quad B^i = B^{i-1}\alpha^{-1} \bmod F + Cd_{(i-1)}$
4: **end for**
5: $P = A^{(m+1)/2} + B^{(m+1)/2}$

Algorithm 2 Montgomery Multiplication Algorithm in the bit-level formate

Input: $C = (c_{m-1}c_{m-2}\cdots c_0)$, $D = (d_{m-1}d_{m-2}\cdots d_0 0)$, $F = (f_m f_{m-1}\cdots f_0)$
Output: $P = (p_{m-1}p_{m-2}\cdots p_0)$
Initialization:
$A^0 = (a^0_{m-1}a^0_{m-2}\cdots a^0_0) \leftarrow (00\cdots 0)$
$B^0 = (b^0_0 b^0_1 \cdots b^0_{m-1}b^0_m) \leftarrow (00\cdots 00)$
Algorithm:
1: **for** $1 \le i \le (m+1)/2$ **do**
2: $a^{i-1}_{-1} = 0$
3: $b^{i-1}_m = 0$
4: **for** $0 \le j \le m-1$ **do**
5: $a^i_{m-1-j} = a^{i-1}_{m-2-j} + a^{i-1}_{m-1} f_{m-1-j} + d_{m-i}c_{m-1-j}$
6: $b^i_j = b^{i-1}_{j+1} + b^{i-1}_0 f_{j+1} + d_{i-1}c_j$
7: **end for**
8: **end for**
9: **for** $0 \le j \le m-1$ **do**
10: $p_j = a^{(m+1)/2}_{m-1-j} + b^{(m+1)/2}_j$
11: **end for**

3. Dependency Graph

The iterative portion of the Montgomery multiplication algorithm is described by the two recursive Equations (9) and (11). As we notice, the two equations have an identical and independent computation structure. Therefore, they can be represented using a unified dependency graph (DG). The extracted dependency graph is shown in Figure 1 for $m = 5$. The DG is represented in a two-dimensional integer domain $\mathbb{D}$ with indices i and j. The DG has $m \times (m+1)/2$ nodes that compute the operations represented by the recursive Equations (9) and (11). As we notice, the coefficients of A and B are computed in sequence starting with the coefficients of A. We arranged the coefficients of D, C, and F to be able to use the same processing node when computing the coefficients of B.

The initial locations of all inputs are as follows: Input signals d_{m-i} and d_i, $1 \le i \le (m+1)/2$ are entered in sequence from the left direction. The input signals c_{m-1-j} and c_j are entered in sequence from the top of the DG. In addition, f_{m-1-j} and f_{j+1} are entered in sequence from the top of the DG. The initial values of input signals a^0_{m-2-j} and b^0_{j+1} are equal to 0 and entered in sequence using the slanted red lines shown at the right corners of the input nodes. These signals are computed in each node and passed to the nodes of the next row to compute the intermediate partial products. The final result P is the summation of the coefficients of $A^{(m+1)/2}$ and $B^{(m+1)/2}$, where $A^{(m+1)/2}$ is delayed by one clock cycle relative to $B^{(m+1)/2}$. The summation is implemented using 2-input XOR gates, and delay is implemented by using two Latches, indicated by the red boxes, at the bottom of the DG as shown in Figure 1.

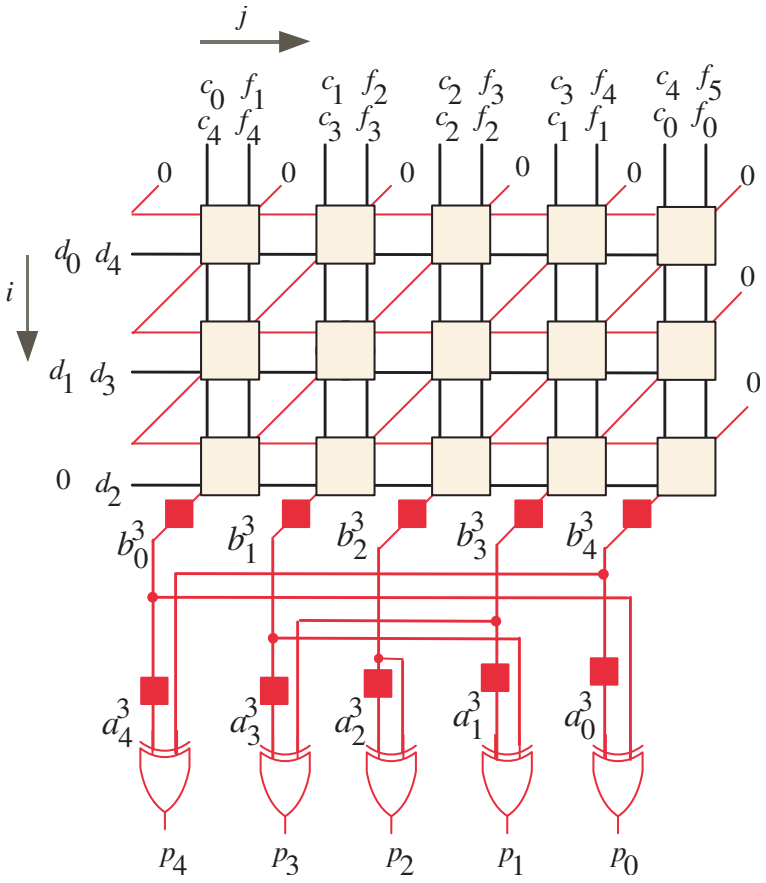

Figure 1. DG of the Montgomery algorithm for $m = 5$.

4. Exploration of the Semi-Systolic Multiplier Layout

This section discusses the methodology used to explore the one-dimensional parallel semi-systolic multiplier architecture. We will focus on the scheduling and node projection techniques offered in [35–37] and applied to the DG to develop the recommended parallel multiplier structure from the chosen Montgomery algorithm.

4.1. Scheduling Function

Suppose point $\mathbf{p}(i,j) = [i\ j]$ defines any DG node. In addition, assume scheduling vector $\mathbf{s} = [s_0\ s_1]$ is used to determine the time scheduling of each node by using the following scheduling function:

$$G(\mathbf{p}) = \mathbf{s}\ \mathbf{p} - v = is_0 + js_1 - v \tag{12}$$

where the scalar value v was incorporated in the previous equation to prevent assigning any node of the DG with negative time values. In our situation, selecting $v \equiv 0$ would ensure that only positive time values be allocated to the DG nodes depicted in Figure 1.

There are constraints on the scheduling vector, and it can only have a certain range of values. For instance, the nodes allocated at $\mathbf{p} = [i,j]$ must be executed after nodes allocated at $\mathbf{p} = [i-1,j]$, i.e.,

$$G(\mathbf{p} = [i,j]) > G(\mathbf{p} = [i-1,j]) \tag{13}$$

The equation above can be written as follows using the coordinate values of $\mathbf{s}$:

$$\begin{aligned} is_0 + js_1 &> (i-1)s_0 + js_1 \\ s_0 &> 0 \end{aligned} \tag{14}$$

Using the iterations in Equations (9) and (11), we can obtain further timing restriction. According to these equations, it is necessary to perform operations allocated at nodes $\mathbf{p} = [i, j+1]$ after operations allocated at nodes $\mathbf{p} = [i-1, j]$, i.e.,

$$G(\mathbf{p} = [i, j+1]) > G(\mathbf{p} = [i-1, j]) \tag{15}$$

The equation above can be written as follows using the coordinate values of $\mathbf{s}$:

$$\begin{aligned} is_0 + js_1 + s_1 &> is_0 - s_0 + js_1 \\ s_1 &> -s_0 \end{aligned} \tag{16}$$

We can select appropriate scheduling vectors using the inequalities (14) and (16). We could use the following scheduling vector $\mathbf{s}$ as one option for a legitimate scheduling vector:

$$\mathbf{s} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{17}$$

The accompanying DG for this scheduling vector is displayed in Figure 2. As we notice, the input signals c^{-1}_{m-1-j}, f_{m-1-j}, c_j, and f_{j+1} are fed in parallel. After $(m+1)/2+3$ clock cycles, the output signals p_{m-1-j}, $0 \le j \le m-1$ are gained in parallel.

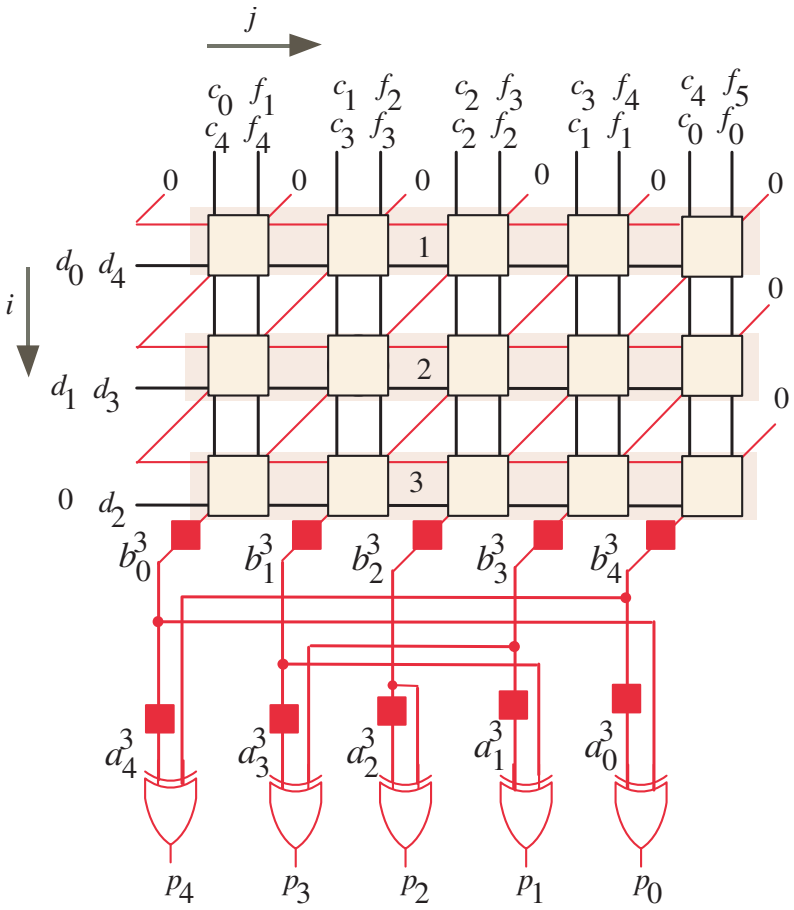

Figure 2. Node timing for $m = 5$.

4.2. Projection Function

In accordance with [35], the projection function converts a large number of DG nodes or points $\mathbf{p}(i, j)$ into a single processing element $\overline{\mathbf{p}}$. The systolic/semi-systolic array is

developed by connecting the resulting processing elements together. One way to express the projection function is as follows:

$$\overline{\mathbf{p}} = \mathbf{H}\,\mathbf{p} \tag{18}$$

where **H** represents a projection matrix. In order to find the projection matrix, the projection vector **V** should be located first. As discussed [35], the projection vector **V** is the null space of projection matrix **H**. The restriction listed below ought to be applied to the projection vector, as per the discussion in [35]:

$$\mathbf{s}\mathbf{V} \neq \mathbf{0} \tag{19}$$

This restriction makes sure that each PE completes the assigned tasks at various times. In addition, a more effective utilization of PE is produced by this multiplexing.

Using the restrictions imposed on **V**, Equation (19)), the scheduling vector $\mathbf{s} = [1\ \ 0]$, and the projection vector that produces the bit-parallel semi-systolic structure is provided by:

$$\mathbf{V} = \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{20}$$

Because **V** is the null space of **H**, we can write the projection matrix **H** as follows:

$$\mathbf{H} = \begin{bmatrix} 0 & 1 \end{bmatrix} \tag{21}$$

4.3. Semi-Systolic Multiplier Architecture

For each DG node or point, $\mathbf{p}[i, j]$, we can acquire the $G(\mathbf{p})$ and $\overline{\mathbf{p}}(\mathbf{p})$ functions by including vectors $\mathbf{s} = [1\ \ 0]$ and $\mathbf{H} = [0\ \ 1]$ in Equations (12) and (18). The resultant functions can be defined as follows:

$$\begin{aligned} G(\mathbf{p}) &= i \\ \overline{\mathbf{p}}(\mathbf{p}) &= j \end{aligned} \tag{22}$$

By applying the derived functions of $G(\mathbf{p})$ and $\overline{\mathbf{p}}(\mathbf{p})$ to the DG points (nodes), we can extract the one-dimensional bit-parallel semi-systolic multiplier structure displayed in Figure 3. As depicted in Figure 3, the semi-systolic structure is composed of m regular PEs. The PEs' internal logic is displayed in Figure 4. We can reduce the area and time complexities by modifying the PE logic by replacing the MUX component with an AND gate as shown in Figure 5. This modification will reduce the critical pass delay of the semi-systolic array to be represented as the summation of the delays of the 2-input AND gate and the 3-input XOR gate instead of 2-input MUX and 3-input XOR.

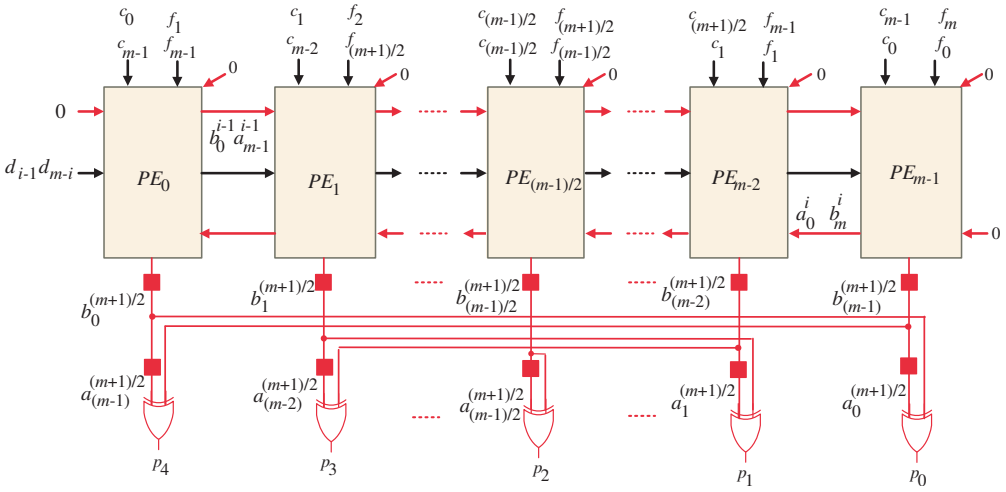

Figure 3. Semi-systolic bit-parallel multiplier structure.

The proposed semi-systolic multiplier differs from the previously published two-dimensional parallel systolic designs in that it has an area complexity of order $\mathcal{O}(m)$ as opposed to $\mathcal{O}(m^2)$. Additionally, the semi-systolic array's final output is accessible after a latency of $(m+1)/2+3$ clock cycles, just like with the Montgomery two-dimensional parallel semi-systolic structures that have been recently published [32,33]. The resulting semi-systolic multiplier structure is superior to them in terms of space complexity. Moreover, the proposed multiplier structure outperforms the parallel multipliers that are based on the conventional field multiplication, [3,4,28,29,31,38,39], in terms of area and latency, as will be presented in the Results section.

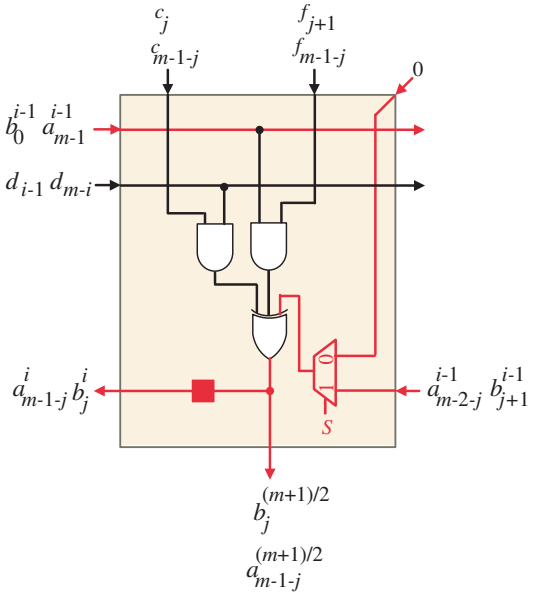

Figure 4. Logic diagram of PE_j.

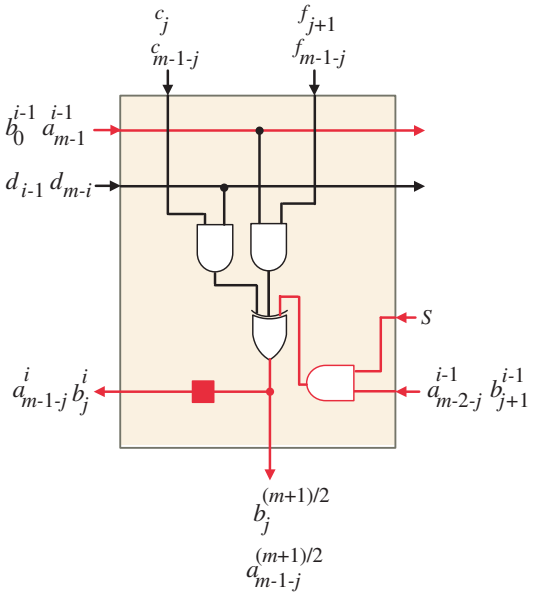

Figure 5. Simplified Logic diagram of PE_j.

By examining Figures 3 and 4, we can describe the developed parallel semi-systolic multiplier's layout as follows: The input signals c_{m-1-j}, f_{m-1-j}, c_j, and f_{j+1} are assigned to each PE. The initial values of input signals a_{m-2-j} and b_{j+1}, $0 \leq j \leq m-1$, are equal to zero. Therefore, the inputs at the right corners of the PEs are assigned zero values as shown in Figure 3. In addition, the initial values of input signals a_{m-1} and b_0 are equal to

zero as indicated by the zero input shown on the left side of the semi-systolic array. The input signals d_{m-i} and d_{i-1}, $1 \leq i \leq (m+1)/2$ are fed in sequence and go through all the PEs. Each PE generates the intermediate signals of a^i_{m-1-j} and b^i_j, $1 \leq i \leq (m+1)/2$ and $0 \leq j \leq m-1$, in sequence and pipelined them through the D latch (the solid red box shown in Figures 4 and 5) to the next PE. After $(m+1)/2$ clock cycles, the resulting bits of $a^{(m+1)/2}_{m-1-j}$ and $b^{(m+1)/2}_j$, $0 \leq j \leq m-1$ will be available in parallel at the outputs of all the PEs. Signals of $a^{(m+1)/2}_{m-1-j}$ are delayed by one clock cycle relative to signals of $b^{(m+1)/2}_j$, $0 \leq j \leq m-1$. This is implemented by using the latches (red boxes) shown at the output of the semi-systolic array shown in Figure 3. The final product bits p_{m-1-j}, $0 \leq j \leq m-1$ is obtained at clock cycle $(m+1)/2+3$ from adding (using 2-input XOR gates) the bits of $a^{(m+1)/2}_{m-1-j}$ and $b^{(m+1)/2}_j$, $0 \leq j \leq m-1$.

Following is an explanation of how the investigated bit-parallel semi-systolic multiplier structure functions:

1. Through the first two clock periods, select signal S is deactivated ($S = 0$) to enforce the zero input bits of a_{m-2-j} and b_{j+1}, $0 \leq j \leq m-1$, to be localized in each PE. When S is equal to zero, the AND gate output will equal zero, which represents the initial values of a_{m-2-j} and b_{j+1}, $0 \leq j \leq m-1$. When S activates to one, the AND gate output will represent the intermediate values of the partial results of a^{i-1}_{m-2-j} and b^{i-1}_{j+1}, $0 \leq j \leq m-1$. At the same clock periods, the input bits of d_{m-1} and d_0 are fed in sequence to go through all the PEs.
2. The PEs produce the internal bit values a^i_{m-1-j} and b^i_j, $2 \leq i \leq (m+1)/2$ and $0 \leq j \leq m-1$, sequentially, over the forthcoming $(m+1)/2-1$ clock periods. Additionally, all PEs receive input bits in a bit sequence from d_{m-i} and d_{i-1}, $2 \leq i \leq (m+1)/2$.
3. The resultant output bits of the product P, p_{m-1-j}, $0 \leq j \leq m-1$ are produced in parallel at the outputs of XOR gates shown in Figure 3. They are generated at the last clock period $(m+1)/2+3$.

5. Results and Discussion

In this part, we compare the suggested one-dimensional bit-parallel semi-systolic multiplier to the previous impactful systolic/semi-systolic multiplier structures of [3,4,28,29,31–33,38,39] in addition to the competitive sequential multiplier of [34]. This section is divided into two subsections: The first one discusses in detail the area and time complexities of the proposed design and compares them to that of the competitive designs. The second subsection verifies the complexity analysis results using real implementation.

5.1. Complexity Analysis

As we notice from the offered semi-systolic structure depicted in Figure 3, it composed of regular m PEs having $3m$ AND gates, m 3-input XOR gates that are equivalent to $2m$ 2-input XOR gates, 0 MUXes, and $3m$ Latches. Using m 2-input XOR gates, we added the resulting bits of $a^{(m+1)/2}_{m-1-j}$ and $b^{(m+1)/2}_j$, $0 \leq j \leq m-1$, to produce the output bits of the final product p_j. As a result, the overall number of used 2-input XOR gates ought to be $3m$. As was previously mentioned, the suggested multiplier takes $(m+1)/2+3$ clock periods to generate the output results. By looking into the exact details of the PE logic, we can calculate the critical path delay (CPD) of the offered multiplier as the sum of the propagation delays of one 2-input AND gate (T_A) and two 2-input XOR gates ($2T_X$).

Table 1 estimates space in terms of the total number of utilized components (gates, MUXes, and latches), latency, and CPD of the recommended semi-systolic multiplier structure along with the currently available systolic/semi-systolic multiplier structures of [3,4,28,29,31–33,38,39] and the competitive sequential multiplier structure of [34]. Table 1 shows that the designs of [3,4,28,29,31–33,38] have an area complexity of order $\mathcal{O}(m^2)$, whereas the designs of [34,39], and the proposed one have an area complexity of order $\mathcal{O}(m)$. In addition, it indicates that all the designs have time complexity of order $\mathcal{O}(m)$.

For IoT and embedded applications, the designs of [34,39] and the proposed one are more suitable than the other designs. Therefore, we will concentrate on comparing the proposed design to the competitive designs of [34,39]. In terms of area, we notice that the proposed design has m more AND and XOR gates than the competitive ones, but it has zero MUXes compared to them. In addition, the proposed design has the same latches compared to the design of [34] and the design of [39] has less area by m latches than the proposed one. On the other hand, the proposed design has significant less latency compared to the competitive designs of [34,39] and almost the same critical path delay (CPD). The real implementation results provided in Table 2 show that the proposed design has slightly lower area complexity and significantly lower time complexity than the competitive ones as we will discuss in the upcoming sections.

Table 1. Complexity analyses in terms of area and time of the suggested and the existing multipliers.

Design	AND	XOR	MUX	Latch	Latency	CPD	Area Complexity	Time Complexity
Huang [3]	$2m^2$	$2m^2$	0	$2m^2+m+1$	$m+1$	T_A+T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Chiou [31]	m^2	$3m^2+2m$	0	$3m^2+4m$	$m+1$	T_A+3T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Lee [32]	m^2+m	m^2+2m	0	$1.6m^2+4m$	$(m+7)/2$	T_A+T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Lee [33]	m^2+m	$m^2+(7m+1)/2$	0	$2.1m^2+6.5m$	$(m+7)/2$	T_A+T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Chiou [38]	m^2	m^2+m	m	$2m^2+3m$	$m+1$	$T_A+T_X+T_M$	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Kim [4]	$2m^2+2m$	$2m^2+3m$	0	$3m^2+4m$	$\lfloor \frac{m}{2} \rfloor+1$	T_A+T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Sarmadi [28]	$(m^2)^*$	$1.5m^2+0.5m$	$1.5m^2-2.5m+3$	$1.5m^2+2m-1$	$m+2$	T_N+T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Mathe [29]	m	m^2-1	m^2-m	m^2	m	T_M+2T_X	$\mathcal{O}(m^2)$	$\mathcal{O}(m)$
Mathe [34]	$2m$	$2m$	$2m$	$3m$	m	$T_A+T_X+T_M$	$\mathcal{O}(m)$	$\mathcal{O}(m)$
Ibrahim [39]	$2m$	$2m$	$3m$	$2m$	m	$T_A+T_X+T_M$	$\mathcal{O}(m)$	$\mathcal{O}(m)$
Proposed	$3m$	$3m$	0	$3m$	$(m+7)/2$	T_A+2T_X	$\mathcal{O}(m)$	$\mathcal{O}(m)$

$(*)$ 2-input NAND gates.

5.2. Implementation Results

To verify and assess the performance of the compared designs, we used VHDL programming language to model the recommended semi-systolic multiplier structure and the competing multiplier structures of [34,39]. The resultant code is synthesized by using Synopsis design compiler with Nangate library (1.5 nm, 0.8 V). The multiplier structures were tested using Modelsim's functional verification tools before being synthesized. An evaluation of the power usage occurs at a frequency of 10 MHz.

For the recommended field sizes of $m = 409$ and $m = 571$, we obtained the synthesis results of area, delay, and power consumption shown in Table 2. From the obtained synthesis results, we computed the design metrics of area–delay product (ADP) and power–delay product (PDP). Table 2 also illustrates the savings between the recommended bit-parallel semi-systolic multiplier structure and its competitors [34,39] in terms of space, consumed power, ADP, and PDP. The following can be seen by reading the results found in Table 2:

- The suggested semi-systolic multiplier structure uses slightly less space and power than competing designs [34,39]. The average savings of area for $m = 409$ are ranging from 2% to 6.6% and ranging from 2.1% to 7.4% for $m = 571$. The achievable average reduction in power consumptions for $m = 409$ of the developed multiplier structure over the competitive multiplier structures are ranging from 4.9% to 13.4% and ranging from 6.8% to 11.8% for $m = 571$. The reduction in area and power is mainly due to the slightly lower gate counts and wire area of the proposed design over the competitive ones.

- The suggested semi-systolic multiplier structure has significant savings in delay over the competitive designs of [34,39]. This is attributed to the significant reduction in latency of the proposed designs over the competitive ones. The average savings of delay for $m = 409$ are ranging from 49.2% to 50.0% and ranging from 45.9% to 46.4% for $m = 571$;
- The ADP and PDP of the developed semi-systolic structure are significantly lower than those of the rival designs of [34,39]; The average reductions of ADP at $m = 409$ are ranging from 50.2% to 53.3% and ranging from 47.0% to 50.4% for $m = 571$; The achievable average reduction of PDP of the offered multiplier structure over the competitive ones for $m = 409$ is ranging from 51.7% to 56.7% and ranging from 49.5% to 52.8% for $m = 571$.

Table 2. Analyzing the performance of different multiplier structures for the values of $m = 409$ and $m = 571$.

Multiplier	Type	m	Area [*Kgates*]	Delay [*ns*]	Power [*mW*]	ADP	PDP	Area Saving (%)	Delay Saving (%)	Power Saving (%)	ADP Saving (%)	PDP Saving (%)
Mathe [34]	Sequential	409	10.6	13.4	6.7	142.0	89.8	6.6	50.0	13.4	53.3	56.7
		571	14.9	18.3	9.3	272.7	170.2	7.4	46.4	11.8	50.4	52.8
Ibrahim [39]	Systolic	409	10.1	13.2	6.1	133.3	80.5	2.0	49.2	4.9	50.2	51.7
		571	14.1	18.1	8.8	255.2	159.3	2.1	45.9	6.8	47.0	49.5
Proposed	Systolic	409	9.9	6.7	5.8	66.3	38.9	-	-	-	-	
		571	13.8	9.8	8.2	135.2	80.4	-	-	-	-	

As we notice from the acquired results, the suggested one-dimensional bit-parallel semi-systolic multiplier has marginal space and power savings. Additionally, it offers significant savings in terms of delay, ADP, and PDP results compared to the competitor ones. Therefore, the offered multiplier structure is more appropriate for use in devices with limited resources, such as IoT edge nodes and other tiny embedded devices.

6. Summary and Conclusions

In this article, we derived a low complexity one-dimensional bit-parallel semi-systolic array structure for polynomial-basis Montgomery multiplication in GF(2^m). A dependency graph can be used to depict the chosen algorithm, which is a standard iterative algorithm. We were able to acquire the feasible bit-parallel semi-systolic multiplier architecture by allocating the proper scheduling and node projection functions to each DG node. The developed one-dimensional parallel structure differs from the previously published two-dimensional parallel structures in that it has space complexity of order $\mathcal{O}(m)$ as opposed to order $\mathcal{O}(m^2)$ of later ones. The recommended multiplier structure, like other systolic structures, has a modular structure with local connectivity between its PEs. This feature makes the recommended multiplier design more appropriate for VLSI implementation. According to the space and time complexities analysis, the offered multiplier has a significant reduction in delay when compared to the majority of the parallel competitive multipliers, as well as a comparable number of logic components. To confirm the findings of the complexity analysis, we used the ASIC CMOS library to synthesize the suggested and the previously described one-dimensional multiplier architectures to assess their performance. According to the acquired results, the offered multiplier architecture has marginal space and power savings compared to the competitive multiplier structures. In addition, it has significant savings in delay, area–delay, and power–delay products. Therefore, we can conclude that the offered multiplier architecture is appropriate for use in devices with limited resources, such as IoT edge nodes and tiny embedded systems. We will merge the recommended multiplier architecture into the ECC crypto-processor in the forthcoming project to quantify the system's overall space, time, and energy savings.

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

Funding: Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, project number (IF2-PSAU-2022/01/21637).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IF2-PSAU-2022/01/21637).

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
ADP	Area–Delay Product
PDP	Power–Delay Product
ASIC	Application Specific Integrated Circuit
ECC	Elliptic Curve Cryptography
DG	Dependency Graph
CPD	Critical Path Delay

References

1. Chen, C.C.; Lee, C.Y.; Lu, E.H. Scalable and Systolic Montgomery Multipliers Over GF(2^m). *IEICE Trans. Fundam.* **2008**, *E91-A*, 1763–1771. [CrossRef]
2. 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]
3. Huang, W.T.; Chang, C.; Chiou, C.; Chou, F. Concurrent error detection and correction in a polynomial basis multiplier over GF(2^m). *IET Inf. Secur.* **2010**, *4*, 111–124. [CrossRef]
4. Kim, K.W.; Jeon, J.C. Polynomial Basis Multiplier Using Cellular Systolic Architecture. *IETE J. Res.* **2014**, *60*, 194–199. [CrossRef]
5. Choi, S.; Lee, K. Efficient systolic modular multiplier/squarer for fast exponentiation over GF(2^m). *IEICE Electron. Express* **2015**, *12*, 1–6. [CrossRef]
6. Reyhani-Masoleh, A. A new bit-serial architecture for field multiplication using polynomial bases. In *Cryptographic Hardware and Embedded Systems, Proceedings of the 7th International Workshop Cryptographic Hardware Embedded Systems (CHES 2008), Washington, DC, USA, 10–13 August 2008*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 330–314.
7. Abdulrahman, E.A.; Reyhani-Masoleh, A. High-Speed Hybrid-Double Multiplication Architectures Using New Serial-Out Bit-Level Mastrovito Multipliers. *IEEE Trans. Comput.* **2016**, *65*, 1734–1747. [CrossRef]
8. Kim, K.W.; Jeon, J.C. A semi-systolic Montgomery multiplier over GF(2^m). *IEICE Electron. Express* **2015**, *12*, 20150769. [CrossRef]
9. Ibrahim, A. Novel Bit-Serial Semi-Systolic Array Structure for Simultaneously Computing Field Multiplication and Squaring. *IEICE Electron. Express* **2019**, *16*, 20190600. [CrossRef]
10. Kim, K.W.; Lee, J.D. Efficient unified semi-systolic arrays for multiplication and squaring over GF(2^m). *Electron. Express* **2017**, *14*, 20170458. [CrossRef]
11. Kim, K.W.; Kim, S.H. Efficient bit-parallel systolic architecture for multiplication and squaring over GF(2^m). *IEICE Electron. Express* **2018**, *15*, 20171195. [CrossRef]
12. Ibrahim, A. Efficient Parallel and Serial Systolic Structures for Multiplication and Squaring Over GF(2^m). *Can. J. Electr. Comput. Eng.* **2019**, *42*, 114–120. [CrossRef]
13. Roman, S. *Field Theory*, 2nd ed.; Springer: New York, NY, USA, 1983.
14. Pillutla, S.R.; Boppana, L. Area-efficient low-latency polynomial basis finite field GF(2^m) systolic multiplier for a class of trinomials. *Microelectron. J.* **2020**, *97*, 104709. [CrossRef]
15. Imana, J.L. LFSR-Based Bit-Serial GF(2^m) Multipliers Using Irreducible Trinomials. *IEEE Trans. Comput.* **2020**, *70*, 156–162.

16. Pillutla, S.R.; Boppana, L. Low-latency area-efficient systolic bit-parallel GF(2^m) multiplier for a narrow class of trinomials. *Microelectron. J.* **2021**, *117*, 105275. [CrossRef]
17. Li, Y.; Cui, X.; Zhang, Y. An Efficient CRT-based Bit-parallel Multiplier for Special Pentanomials. *IEEE Trans. Comput.* **2021**, *71*, 736–742. [CrossRef]
18. Li, Y.; Zhang, Y.; He, W. Fast hybrid Karatsuba multiplier for type II pentanomials. *IEEE Trans. Very Large Scale Integr. (Vlsi) Syst.* **2020**, *28*, 2459–2463. [CrossRef]
19. Meher, P.K.; Lou, X. Low-Latency, Low-Area, and Scalable Systolic-Like Modular Multipliers for GF(2^m) Based on Irreducible All-One Polynomials. *IEEE Trans. Circuits Syst. Regul. Pap.* **2016**, *64*, 399–408. [CrossRef]
20. Mohaghegh, S.; Yemiscoglu, G.; Muhtaroglu, A. Low-Power and Area-Efficient Finite Field Multiplier Architecture Based on Irreducible All-One Polynomials. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
21. Zhang, Y.; Li, Y. Efficient Hybrid GF(2^m) Multiplier for All-One Polynomial Using Varied Karatsuba Algorithm. *IEICE Trans. Fundam. Electron. Comput. Sci.* **2021**, *104*, 636–639. [CrossRef]
22. Zhou, B.B. A New Bit Serial Systolic Multiplier over GF(2^m). *IEEE Trans. Comput.* **1988**, *37*, 749–751. [CrossRef]
23. Fenn, S.T.J.; Taylor, D.; Benaissa, M. A Dual Basis Bit Serial Systolic Multiplier for GF(2^m). *Integration* **1995**, *18*, 139–149. [CrossRef]
24. Lee, C.Y.; Lu, E.H.; Lee, J.Y. Bit-Parallel Systolic Multipliers for GF(2^m) Fields Defined by All-One and Equally-Spaced Polynomials. *IEEE Trans. Comput.* **2001**, *50*, 358–393.
25. Lee, C.Y.; Lu, E.H.; Sun, L.F. 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* **2001**, *50*, 519–523.
26. Lee, C.Y.; Chiou, C.W. 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.* **2005**, *E88-A*, 3169–3179. [CrossRef]
27. Lee, C.Y. Low-latency bit-pararallel systolic multiplier for irreducible $x^m + x^n + 1$ with $GCD(m, n) = 1$. *IEICE Trans. Fund. Elect. Commun. Comput. Sci.* **2008**, *55*, 828–837.
28. Bayat-Sarmadi, S.; Farmani, M. High-Throughput Low-Complexity Systolic Montgomery Multiplication Over GF(2^m) Based on Trinomials. *IEEE Trans. Circuits Syst. II* **2015**, *62*, 377–381. [CrossRef]
29. Mathe, S.E.; Boppana, L. Bit-parallel systolic multiplier over GF(2^m) for irreducible trinomials with ASIC and FPGA implementations. *IET Circuits Desvices Syst.* **2018**, *12*, 315–325. [CrossRef]
30. Lee, C.Y.; Chiou, C.W.; Lin, J.M. Concurrent error detection in a polynomial basis multiplier over GF(2^m). *J. Electron. Test.* **2006**, *22*, 143–150. [CrossRef]
31. Chiou, C.W.; Lee, C.M.; Sun, Y.S.; Lee, C.Y.; Lin, J.M. High-throughput Dickson basis multiplier with a trinomial for lightweight cryptosystems. *IET Comput. Digit. Tech.* **2018**, *12*, 187–191. [CrossRef]
32. Lee, K. Resource and Delay Efficient Polynomial Multiplier over Finite Fields GF(2^m). *J. Korea Soc. Digit. Ind. Inf. Manag.* **2020**, *16*, 1–9.
33. Lee, K. Low Complexity Systolic Montgomery Multiplication over Finite Fields GF(2^m). *J. Korea Soc. Digit. Ind. Inf. Manag.* **2022**, *18*, 1–9.
34. Mathe, S.E.; Boppana, L. Design and Implementation of a Sequential Polynomial Basis Multiplier over GF(2^m). *KSII Trans. Int. Inf. Syst.* **2017**, *11*, 2680–2700.
35. Gebali, F. *Algorithms and Parallel Computers*; John Wiley: New York, NY, USA, 2011.
36. 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]
37. Ibrahim, A.; Alsomani, T.; Gebali, F. New Systolic Array Architecture for Finite Field Inversion. *IEEE Can. J. Electr. Comput. Eng.* **2017**, *40*, 23–30. [CrossRef]
38. Chiou, C.W.; Lin, J.M.; Lee, C.Y.; Ma, C.T. Novel Mastrovito Multiplier over GF(2^m) Using Trinomial. In Proceedings of the 2011 5th International Conference on Genetic and Evolutionary Computing (ICGEC), Kitakyushu, Japan, 29 August–1 September 2011; pp. 237–242.
39. Ibrahim, A.; Gebali, F.; Bouteraa, Y.; Tariq, U.; Ahanger, T.; Alnowaiser, K. Compact Bit-Parallel Systolic Multiplier Over GF(2^m). *IEEE Can. J. Electr. Comput. Eng.* **2021**, *44*, 199–205. [CrossRef]