UVicSPACE: Research & Learning Repository

University of Victoria

Faculty of Engineering

Faculty Publications

"Energy-Efficient Word-Serial Processor for Field Multiplication and Squaring Suitable for Lightweight Authentication Schemes in RFID-Based IoT Applications"

Atef Ibrahim and Fayez Gebali

2021

© 2021 Ibrahim et al. This is an open access article distributed under the terms of the Creative Commons Attribution License. http://creativecommons.org/licenses/by/4.0

This article was originally published at:
https://doi.org/10.3390/app11156938

Citation for this paper:

Ibrahim, A., & Gebali, F. (2021). Energy-efficient word-serial processor for field multiplication and squaring suitable for lightweight authentication schemes in rfid-based iot applications. *applied sciences*, *11*(6938), 1-23.
https://doi.org/10.3390/app11156938

applied sciences

MDPI

Article

Energy-Efficient Word-Serial Processor for Field Multiplication and Squaring Suitable for Lightweight Authentication Schemes in RFID-Based IoT Applications

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; fayez@uvic.ca
* Correspondence: attif_ali2002@yahoo.com or aa.mohamed@psau.edu.sa or atef@ece.uvic.ca

Abstract: Radio-Frequency Identification (RFID) technology is a crucial technology used in many IoT applications such as healthcare, asset tracking, logistics, supply chain management, assembly, manufacturing, and payment systems. Nonetheless, RFID-based IoT applications have many security and privacy issues restricting their use on a large scale. Many authors have proposed lightweight RFID authentication schemes based on Elliptic Curve Cryptography (ECC) with a low-cost implementation to solve these issues. Finite-field multiplication are at the heart of these schemes, and their implementation significantly affects the system's overall performance. This article presents a formal methodology for developing a word-based serial-in/serial-out semisystolic processor that shares hardware resources for multiplication and squaring operations in GF(2^n). The processor concurrently executes both operations and hence reduces the execution time. Furthermore, sharing the hardware resources provides savings in the area and consumed energy. The acquired implementation results for the field size $n = 409$ indicate that the proposed structure achieves a significant reduction in the area–time product and consumed energy over the previously published designs by at least 32.3% and 70%, respectively. The achieved results make the proposed design more suitable to realize cryptographic primitives in resource-constrained RFID devices.

Keywords: energy-efficient IoT; radio-frequency identification; lightweight authentication schemes; embedded security; cryptographic processors; finite-field arithmetic; energy-efficient edge devices; parallel processing

check for updates

Citation: Ibrahim, A.; Gebali, F. Energy-Efficient Word-Serial Processor for Field Multiplication and Squaring Suitable for Lightweight Authentication Schemes in RFID-Based IoT Applications. *Appl. Sci.* **2021**, *11*, 6938. https://doi.org/10.3390/app11156938

Academic Editor: Manuel Armada

Received: 07 June 2021
Accepted: 26 July 2021
Published: 28 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Things (IoT) is a new paradigm that connects a significant number of objects—such as wearable devices, sensors, smartphones, smart meters, and vehicles—to the Internet [1]. The IoT is a promising technology that provides services and practical solutions in several fields, including smart cities, healthcare, smart grids, smart homes, logistics, industrial manufacturing, intelligent transportation, and business management [2,3].

Radio-Frequency Identification (RFID) is an essential technology in most IoT applications, especially healthcare applications [4]. RFID systems have a small chip called an RFID tag implanted in an object. The tag contains data that a reader can read via short-range radio waves. The data retrieved by the reader are transmitted to the server for further processing. RFID systems increase productivity in applications by providing contactless and automatic object identification [5].

Based on the cost, memory size, and battery requirements, RFID tags are categorized into three classes: passive tags, semipassive tags, and active tags [6]. Passive tags acquire their energy from the reader and hence do not require a battery to operate. They use the reader to trigger a response from the tag. They have a limited storage memory, a lower cost,

a longer lifespan, and a shorter communication range than the other classes. Semipassive tags have an internal chip battery to power a sensor and run the circuitry on the chip. They work similarly to passive tags by using the reader to trigger a response from the tag. They have a more extensive memory storage, a higher cost, a lower lifespan, and a longer communication range than passive tags. Active tags are powered by batteries and use two-way communication between the reader and the tag. They have a larger size and greater computing capabilities compared to the passive and semipassive tags. They have a larger storage capacity, a higher cost, a limited lifespan, and are more expensive than the other types of tags.

Due to the distinctive feature of contactless and automatic object identification, RFID technology is widely used in many IoT applications for delivering intelligent services. For example, in healthcare applications, RFID technology provides intelligent services such as patient monitoring, medical asset tracking, and drug administration [7,8]. In the field of logistics, RFID helps in delivering products. With the RFID tag attached to a package, we can track the process from pick up to delivery by tracking the current location of the products, which helps minimize the rate of incorrect deliveries due to human error [9]. In the field of assembly and manufacturing, we use RFID tags to improve throughput and reduce defects. Furthermore, we can use them in the manufacturing process to determine the product being assembled and the component that will be mounted. It is then possible to perform an instant check during assembly to select the parts that must be installed in the product and whether the components installed are correct. Thus, RFID plays a role in ensuring the quality of the final product. For payment systems, RFID can be used to pay for tolls without having to stop. These systems are widely used to make electronic toll collection more convenient, especially on highways and in parking lots [9]. The RFID toll system allows vehicles to automatically check in and out in a contactless, fast, secure, and convenient way [9]. Restaurants are also experimenting with paying for meals at drive-through windows using RFID tags [10]. For security and access control applications, may organizations use RFID cards to control access levels based on the security level granted to the card owners. For example, at the entrance to a building, readers are deployed to allow only authorized people access [9]. In supply chain management, RFID tags can be attached to products or every other part of the supply chain, and RFID readers can track all items from the manufacturing to the sales point [9].

Despite the considerable benefits of RFID-based IoT applications, they suffer from many security and privacy issues, which may hinder the future of this technology [11]. RFID tag privacy and protection concerns may have a significant effect on people or entities. Tags that are not adequately secured are always vulnerable to eavesdropping, Denial of Service (DoS), traffic analysis, and other threats. Therefore, we should apply essential and effective security solutions to protect RFID-based IoT applications.

1.1. Motivation

A direct way to provide adequate security solutions to secure the RFID system is to encrypt all the communications among the reader, the tag, and the server. However, the tags are resource-constrained devices with limited computation power, storage capacity, and energy, which makes implementing the standard cryptographic algorithms on these tags a challenging task. Therefore, over the last several years, researchers have developed numerous lightweight authentication schemes that satisfy the security requirements of the RFID systems. These schemes are based on elliptic curve cryptography (ECC) [12–22], hash functions [23–25], and rotation functions [26–29] to secure communications in IoT environments.

Most of the cryptographic primitives (such as ECC and hash functions) used in the proposed lightweight authentication protocols are mainly based on finite-field arithmetic operations of addition, multiplication, squaring, exponentiation, and division/inversion. Most of these operations are at the heart of the cryptographic primitives; hence, their performance has a significant effect on the overall performance of the whole authentication

protocol. Since the finite-field multiplication operation is the basic building block of the other field operations of division/inversion and exponentiation, it is considered the fundamental building block of most cryptographic processors. Therefore, any slight improvement in its implementation leads to a significant increase in the cryptographic processor's whole performance.

1.2. Contribution

The main contribution of this work is using a systematic methodology to obtain the Dependence Graph (DG) for the modular multiply/square algorithm. A nonlinear scheduling function was applied to the DG and a nonlinear projection function was used to implement a unified processor core to perform both modular multiplication and squaring operations. We followed a systematic approach previously reported by the authors of this paper in [30–36] to extract the processor core of one of the efficient multiplier and squaring algorithms previously reported in the literature, the bipartite multiplication and squaring algorithm suggested by Kim in [37]. We can briefly summarize the approach as follows:

1. We developed the bit-level version of the bipartite multiplication and squaring algorithm to have its regular iterative form;
2. We obtained the DG of the developed algorithm to help in extracting the unified hardware module;
3. We obtained a nonlinear scheduling function to allocate a time value to each node of the DG;
4. We developed a nonlinear projection function to map the DG nodes to the corresponding Processing Element (PE) in the extracted processor core.

The derived nonlinear scheduling and projection functions provide more flexibility to manage the utilization of the processor besides its execution time.

After applying the adopted approach to the suggested algorithm, we obtained a compact and efficient word-based Serial-In/Serial-Out (SISO) semisystolic processor core. The processor architecture simultaneously computes both the multiplication and squaring operations and accordingly reduces the total execution time. The shared processor hardware resources lead to savings in the hardware resource area and the consumed energy, making it more suitable for implementing cryptographic processors in resource-constrained RFID devices.

1.3. Paper Organization

The rest of the article is structured as follows. Section 3 briefly reviews the bipartite multiplication and squaring algorithm over GF(2^n) and its developed bit-level form. Section 4 explains the procedure followed to explore the expected word-based SISO semisystolic processor. Section 5 provides an analysis of the estimated area and time complexities besides the ASIC implementation results. Section 6 draws the summary and conclusion of the work.

2. Related Work

The binary extension field GF(2^n) is exceptionally efficient in hardware implementation due to the removal of the carry arithmetic. One more advantage of GF(2^n) over the other fields is the availability of various representations of the field elements such as the normal, polynomial, and dual basis. There are three main categories of polynomial basis multipliers in GF(2^n): bit-serial, word-serial, and bit-parallel. Most of the bit-serial and bit-parallel multiplier structures have a large hardware resource area and large time complexities, making them unsuitable for implementation in resource-constrained embedded devices such as RFID devices [38–41]. Thus, there is a need to possess multiplier architectures with limited hardware resources and a reasonable execution time suitable for these embedded devices. Word-serial multipliers are the potential candidate for solving this problem because they have a trade-off between the hardware resource area and the time complexities. Therefore, they provide more adaptability to manage the design

space and speed. The word-serial multiplier structures can be divided into four categories: Serial-In/Serial-Output (SISO), Serial-In/Parallel-Output (SIPO), Parallel-In/Serial-Output (PISO), and scalable structures. There are several polynomial-basis word-serial SISO systolic multiplier structures, published in [42–46]. Other polynomial-basis word-serial SIPO systolic multipliers were exhibited in [47–50]. Namin et al. in [51] developed a type-T Gaussian normal-basis word-serial PISO multiplier structure. Different scalable systolic multiplier structures were published in [52–58].

Several authors in the literature [40,41,59] tried to increase the performance and hardware utilization of multiplier structures by merging both multiplication and squaring operations in a combined construction. Even though the presented structures perform both operations concurrently and perform well in high-speed applications, they are mainly not suitable for embedded devices due to the considerable amount of consumed hardware resources and meaningful wasted power. This is mainly attributed to their parallel systolic/semisystolic structures, which require a large amount of hardware resources, which are limited in resource-constrained embedded devices. Increasing the hardware resources necessary to implement these parallel structures leads to increasing their extracted parasitic capacitances and hence raising the circuit switching activities, resulting in a considerable amount of consumed power that cannot be afforded by the embedded devices.

In this research article, we present a competent word-based SISO semisystolic processor to perform the bipartite multiplication and squaring algorithm suggested by Kim in [37]. We developed the bit-level version of the suggested algorithm to be able to extract the hardware structure of the semisystolic multiplier and squarer processor. As mentioned in Section 1.2, the processor architecture simultaneously computes both operations and shares the hardware resources, leading to a reduction of the total processing time, hardware resources, and consumed energy. Moreover, the semisystolic array core of the proposed processor has a regular architecture and local interconnections between the processing elements constituting it, making it highly practical for VLSI implementation. The obtained implementation results for the recommended field size $n = 409$ indicate that the proposed multiplier-squarer structure realizes a reduction in the area–time product and consumed energy over the compared competing multiplier structures at specific embedded word sizes, as discussed in the Results Section. The results also confirm that the recommended multiplier-squarer structure is suitable to realize cryptographic primitives in RFID tags, especially semipassive and active tags, that have modest limitations in area and delay and significant restrictions in wasted energy.

3. Polynomial-Based Bipartite Multiplication-Squaring Algorithm over GF(2^n)

Let $G(\beta)$ denote the irreducible polynomial of order n over GF(2) that defines GF(2^n). Furthermore, let $E(\beta)$ and $F(\beta)$ denote any two polynomial elements of order $n-1$ over GF(2^n). The polynomials $G(\beta)$, $E(\beta)$ and $F(\beta)$ can be respectively expressed as:

$$G(\beta) = \sum_{i=0}^{n} g_i \beta^i \quad (1)$$

$$E(\beta) = \sum_{i=0}^{n-1} e_i \beta^i \quad (2)$$

$$F(\beta) = \sum_{i=0}^{n-1} f_i \beta^i \quad (3)$$

where $g_i, e_i, f_i \in$ GF(2).

Since β is a root of $G(\beta)$, $\beta^n \bmod G(\beta)$ and $\beta^{n+1} \bmod G(\beta)$ can be represented as follows:

$$\beta^n \bmod G(\beta) = \sum_{j=0}^{n-1} g_j \beta^j \quad (4)$$

$$\begin{aligned}\beta^{n+1} \bmod G(\beta) &= \sum_{j=1}^{n-1}(g_{n-1}g_j + g_{j-1})\beta^j + g_{n-1}g_0 \\ &\cong G'(\beta) = \sum_{j=0}^{n-1} g_j'\beta^j \end{aligned} \tag{5}$$

Suppose $c = \lfloor n/2 \rfloor$, $d = \lceil n/2 \rceil$, and assume that $G'(\beta)$ is available in advance. We can express polynomial multiplication and squaring over $GF(2^n)$ as:

$$\begin{aligned} Q(\beta) &= E(\beta)F(\beta) \bmod G(\beta) \\ &= \sum_{i=0}^{n-1} f_i E(\beta)\beta^i \bmod G(\beta) \\ &= [\sum_{i=0}^{d-1} f_{2i}E(\beta)\beta^{2i} + \\ &\quad \beta \sum_{i=0}^{c-1} f_{2i+1}E(\beta)\beta^{2i}] \bmod G(\beta) \end{aligned} \tag{6}$$

$$\begin{aligned} R(\beta) &= E(\beta)E(\beta) \bmod G(\beta) \\ &= \sum_{i=0}^{n-1} e_i E(\beta)\beta^i \bmod G(\beta) \\ &= [\sum_{i=0}^{d-1} e_{2i}E(\beta)\beta^{2i} + \\ &\quad \beta \sum_{i=0}^{c-1} e_{2i+1}E(\beta)\beta^{2i}] \bmod G(\beta) \end{aligned} \tag{7}$$

We can divide $Q(\beta)$ and $R(\beta)$ into two parts as:

$$Q(\beta) = [X(\beta) + \beta Y(\beta)] \bmod G(\beta) \tag{8}$$

$$R(\beta) = [P(\beta) + \beta S(\beta)] \bmod G(\beta) \tag{9}$$

where,

$$X(\beta) = \sum_{i=0}^{d-1} f_{2i}E(\beta)\beta^{2i} \bmod G(\beta) \tag{10}$$

$$Y(\beta) = \sum_{i=0}^{c-1} f_{2i+1}E(\beta)\beta^{2i} \bmod G(\beta) \tag{11}$$

$$P(\beta) = \sum_{i=0}^{d-1} e_{2i}E(\beta)\beta^{2i} \bmod G(\beta) \tag{12}$$

$$S(\beta) = \sum_{i=0}^{c-1} f_{2i+1}E(\beta)\beta^{2i} \bmod G(\beta) \tag{13}$$

Employing the above equations, Kim [37,41] proposed the unified bipartite algorithm, Algorithm 1, to simultaneously compute the products $Q(\beta)$ and $R(\beta)$. In Algorithm 1, we denote partial values of polynomials $E(\beta)$, $X(\beta)$, $Y(\beta)$, $P(\beta)$, and $S(\beta)$ as E^i, X^i, Y^i, P^i, and S^i, where i represents the iteration index. The symbols f_{2i-2}, f_{2i-1}, e_{2i-2}, and e_{2i-1} indicate the $(2i-2)^{th}$ and $(2i-1)^{th}$ coefficients of input polynomials $F(\beta)$ and $E(\beta)$, respectively. In the initialization step of the algorithm, we notice that the coefficients of input polynomial $E(\beta)$ are allocated to the initial variable E^0 and zero values are allocated to the initial variables X^0, Y^0, P^0, and S^0. The for-loop terminates after $d = \lceil n/2 \rceil$ iterations, and the products Q and R can be concurrently computed as shown in Steps 8 and 9 of Algorithm 1.

Algorithm 1 Bipartite multiplication and squaring algorithm in GF(2^n).

Input: $E(\beta), F(\beta) \in \text{GF}(2^n)$, $G(\beta)$, $G'(\beta)$, $d = \lceil n/2 \rceil$, and $c = \lfloor n/2 \rfloor$
Mult. Output: $Q(\beta) = E(\beta).\ F(\beta) \mod G(\beta)$
Square Output: $R(\beta) = E(\beta).\ E(\beta)) \mod G(\beta)$
Initialization:
$E^{(0)} \leftarrow E(\beta)$, $F \leftarrow F(\beta)$, $X^0 \leftarrow 0$, $Y^0 \leftarrow 0$, $P^0 \leftarrow 0$, $S^0 \leftarrow 0,$, $G \leftarrow G(\beta)$, $G' \leftarrow G'(\beta)$
Algorithm:
1: **for** $1 \leq i \leq d$ **do**
2: $E^i = E^{i-1}.\ \beta^2 \mod G$
3: $X^i \leftarrow X^{i-1} + f_{2i-2}E^{i-1}$
4: $Y^i \leftarrow Y^{i-1} + f_{2i-1}E^{i-1}$
5: $P^i \leftarrow P^{i-1} + e_{2i-2}E^{i-1}$
6: $S^i \leftarrow S^{i-1} + e_{2i-1}E^{i-1}$
7: **end for**
8: $Q \leftarrow (X^d + \beta Y^c) \mod G$
9: $R \leftarrow (P^d + \beta S^c) \mod G$

For the hardware implementation of Algorithm 1, we developed the corresponding bit-level form as shown in Algorithm 2. The algorithm has two nested loops to compute Steps 2 to 6 of Algorithm 1, the outer loop with the i index and the inner loop with the j index. Furthermore, there is a for-loop at the end of the algorithm to compute the postprocessing Steps 8 and 9 in Algorithm 1. e^i_j designates the (j)th bit of E at the ith iteration. Furthermore, x^i_j, y^i_j, p^i_j, and s^i_j designate the jth bit of X, Y, P, and S at the ith iteration, respectively. As we notice, the total number of iterations required to execute the algorithm is $(d+1)n$. The explored word-based SISO semisystolic processor will significantly reduce the total number of iterations, as will be explained later, to be $(\left\lceil \frac{d}{k} \right\rceil + 1)\lceil \frac{n}{k} \rceil$, where k is the processor bus width.

Algorithm 2 Bit-level form of the bipartite multiplication and squaring algorithm.

Input: $E(\beta), F(\beta) \in \mathrm{GF}(2^n)$, $G(\beta)$, $G'(\beta)$, $d = \lceil n/2 \rceil$, and $c = \lfloor n/2 \rfloor$
Mult. Output: $Q(\beta) = E(\beta).\, F(\beta) \mod G(\beta)$
Square Output: $R(\beta) = E(\beta).\, E(\beta)) \mod G(\beta)$
Initialization:
$E^0 = (e^0_{n-1} \cdots e^0_1 e^0_0 e^0_{-1} e^0_{-2}) \leftarrow (e_{n-1} \cdots e_1 e_0 00)$
$F \leftarrow (f_{n-1} \cdots f_1 f_0)$
$X^0 = (x^0_{n-1} \cdots x^0_1 x^0_0) \leftarrow (0 \cdots 00)$
$Y^0 = (y^0_{n-1} \cdots y^0_1 y^0_0 y^0_{-1}) \leftarrow (0 \cdots 000)$
$P^0 = (p^0_{n-1} \cdots p^0_1 p^0_0) \leftarrow (0 \cdots 00)$
$S^0 = (s^0_{n-1} \cdots s^0_1 s^0_0 s^0_{-1}) \leftarrow (0 \cdots 000)$
$G \leftarrow (g_{n-1} \cdots g_1 g_0)$
$G' \leftarrow (g'_{n-1} \cdots g'_1 g'_0)$
Algorithm:
1: **for** $1 \le i \le d$ **do**
2: $e^{i-1}_{-1} \leftarrow 0, e^{i-1}_{-2} \leftarrow 0$
3: **for** $0 \le j \le n-1$ **do**
4: $e^i_j = e^{i-1}_{j-2} + e^{i-1}_{n-2} g_j + e^{i-1}_{n-1} g'_j$
5: $x^i_j = x^{i-1}_j + f_{2i-2} e^{i-1}_j$
6: $y^i_j = y^{i-1}_j + f_{2i-1} e^{i-1}_j$
7: $p^i_j = p^{i-1}_j + e_{2i-2} e^{i-1}_j$
8: $s^i_j = s^{i-1}_j + c e_{2i-1} e^{i-1}_j$
9: **end for**
10: **end for**
11: $y^c_{-1} \leftarrow 0, s^c_{-1} \leftarrow 0$
12: **for** $0 \le j \le n-1$ **do**
13: $q_j = x^d_j + y^c_{n-1} g_j + y^c_{j-1}$
14: $r_j = p^d_j + s^c_{n-1} g_j + s^c_{j-1}$
15: **end for**

4. Extraction of the Word-Based SISO Semisystolic Processor

The approach used to explore the intended word-based SISO semisystolic processor starts by extracting the DG of the adopted algorithm. As we notice, Algorithm 2 performs regular iterations with two indices i and j. Therefore, it can be expressed in the two-dimensional (2D) domain, as shown in Figure 1. The DG consists of an array of node operations. Each node can be allocated using the corresponding row index i and the corresponding column index j. The DG has two types of nodes: the upper light orange nodes and the lower light blue nodes. The light orange nodes, arranged in the upper d rows, perform the operations of Steps 4 to 8 of Algorithm 2, while the light blue nodes perform the operations of Steps 13 and 14 of the same algorithm.

The nodes in the first row of the DG receive the initial bits e^0_j, x^0_j, y^0_j, p^0_j, s^0_j, g_j, and g'_j of variables E, X, Y, P, S, G, and G', respectively. In the upper d rows of DG, the updated partial bits e^i_j, x^i_j, y^i_j, p^i_j, and s^i_j besides the bits g_j and g'_j are designated by the vertical lines. The updated partial bits of e^i_j are also passed diagonally as designated by the slanted red lines. The input bits of e_{2i-2}, e_{2i-1}, f_{2i-2}, and f_{2i-1}, as well as the updated partial bits of e^{i-1}_{n-2}, e^{i-1}_{n-1}, and e^{i-1}_{j-2} are broadcast horizontally to all nodes in the same row of the upper d rows in DG. The resulting bits x^d_j, p^d_j from the dth row, as well as the broadcast bits g_j are passed vertically to the nodes in the last row of the DG. Furthermore, the resulting bits of y^c_j and s^c_j are passed diagonally to the same nodes as shown in Figure 1. These bits are used to compute the final bits of field multiplication q_j and field squaring r_j, as indicated in Figure 1.

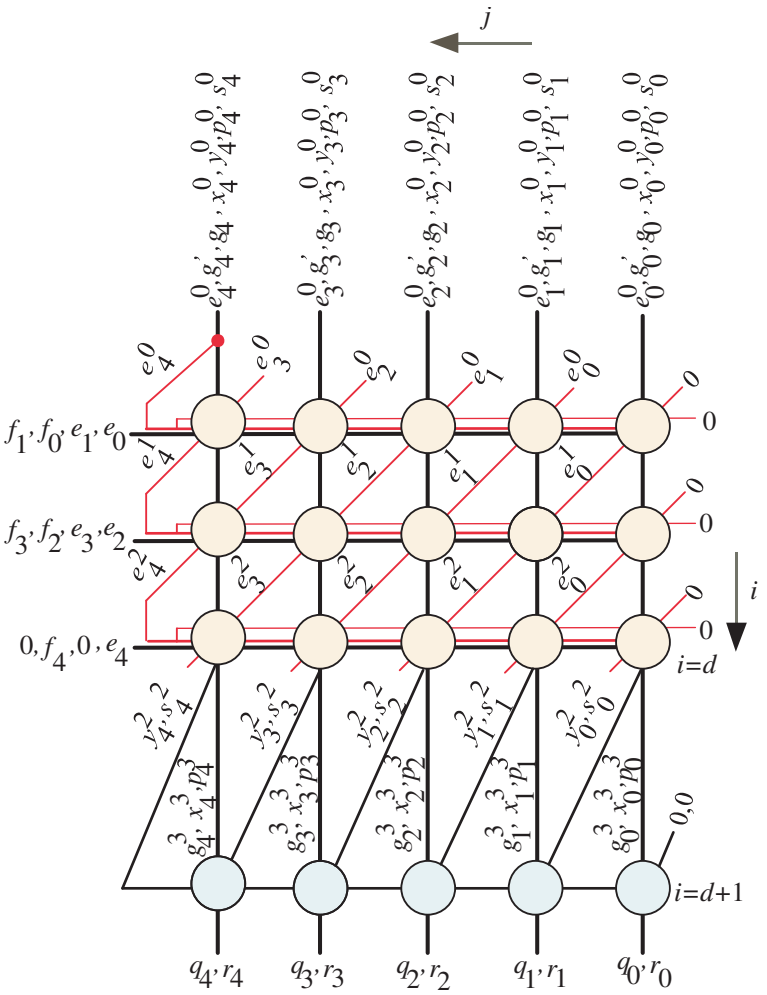

Figure 1. DG of the bipartite multiplication-squaring algorithm for $n = 5$.

The second step in the utilized approach, as discussed in [30], is to find a valid scheduling function to allocate a time value to each node of the DG. The last step of the procedure is to find a projection function to map the DG nodes to a corresponding Processing Element (PE) in the extracted systolic/semisystolic array. There are two types of scheduling and projection functions, as explained in [30]: affine (linear) scheduling and projection functions and nonlinear scheduling and projection functions. The affine scheduling and projection functions have the following limitations:

1. The inputs and outputs can only be accessed as bit-serial (one bit at a time) or bit-parallel (all bits at the same time instance) based on our choice of the scheduling function. That means that the linear functions cannot satisfy any restrictions on the processor bus size;
2. The designer cannot manage the number of accessed input or output samples at a particular time step;
3. The number of active PEs cannot be managed at a specific time step;
4. The designer cannot manage the PE's workload.

The nonlinear scheduling and projection functions avoid all of these limitations and provide the designer with more flexibility to control the number of accessed inputs and outputs and the number of utilized PEs, as well as the PE's workload. In our case, we require that the unified SISO multiplier-squarer feeds in variables E, F, G, and G' in a word-serial fashion at the start of the iteration and produces the output products Q and R in a word-serial fashion at the end of the iteration. Assuming the processor word size is k, we need to feed in k bits of input variables at the first clock cycle and produce k bits of output products at the last clock cycle. The following subsections display the selected nonlinear scheduling

and projection functions that resulted in the exploration of the intended word-based SISO semisystolic processor.

4.1. Word-Based SISO Scheduling Function

Based on the scheduling methodology discussed in [30], we can partition the DG into $k \times k$ equitemporal zones using the following scheduling function:

$$t(\mathbf{p}) = \left\lceil \frac{n}{k} \right\rceil \left\lfloor \frac{i-1}{k} \right\rfloor + \left\lfloor \frac{n-1-j}{k} \right\rfloor + 1 \tag{14}$$

where $t(\mathbf{p})$ represents the time allocated to each node $\mathbf{p}$ of the DG, $1 \leq i \leq n+\omega$, $-\zeta \leq j < n-1$, $\omega = k\left\lceil \frac{d}{k} \right\rceil - d$, $\zeta = k\lceil \frac{n}{k} \rceil - n$.

By applying the chosen scheduling function to the DG, we obtain the node timing (scheduling time) allocated to each zone, as indicated in Figure 2. This figure is developed for the case when $n = 7$ and $k = 3$. The green zones have $k \times k$ nodes that are executed at the time index (blue numerals) allocated to each zone. The yellow zones have $1 \times k$ postprocessing nodes that are executed also at the time index assigned to each of the yellow zones.

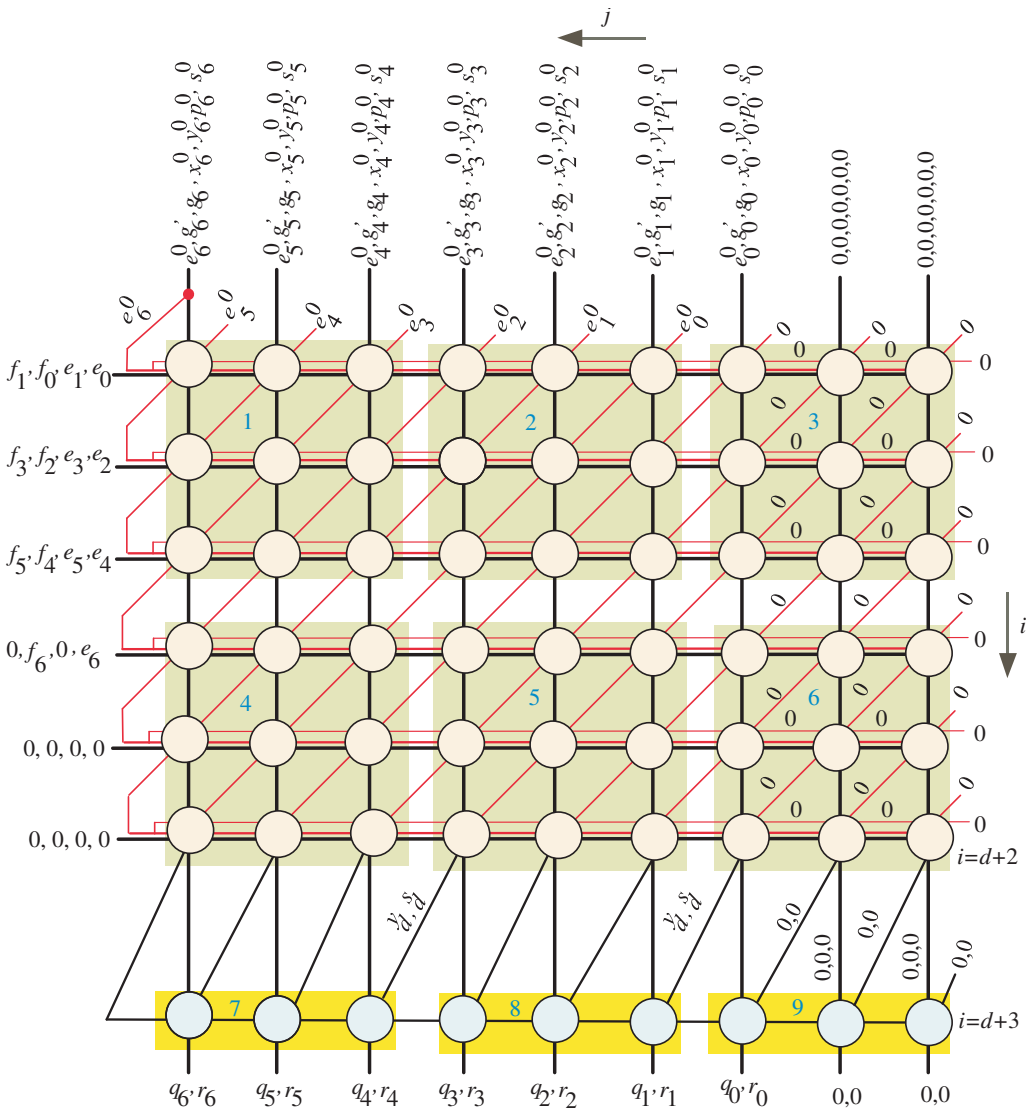

Figure 2. Scheduling time for the case when $n = 7$ and $k = 3$.

For the upper d rows of the DG, it is important to make the number of rows and columns multiple integers of k. Therefore, ω more rows and ζ more columns should be added to the DG. For our case, when $n = 7$ and $k = 3$, both ζ and ω will equal two, and hence, we should add two more rows and two more columns to the DG as indicated in Figure 2. This leads to the least significant two bits of the input variables E, G, and G', and the initial values of the intermediate variables X, Y, P, and S should be assigned zero values, as shown at the rightmost edge of the DG in Figure 2. Furthermore, the most significant two bits of input variables E and F should be assigned zero values at the two rows before the last row of the DG, as shown in Figure 2. It is worth noticing that the e_{n-1}^{i-1} and e_{n-2}^{i-1} signals should be updated at the nodes of the leftmost column, as displayed in Figure 2, and then broadcast horizontally and collectively with the signals e_{2i-2}, e_{2i-1}, f_{2i-2}, and f_{2i-1} to the nodes of row i.

The chosen scheduling function reduces the total number of iterations required to execute the adopted bipartite multiplication/squaring algorithm to be $(\lceil \frac{d}{k} \rceil + 1)\lceil \frac{n}{k} \rceil$ instead of $(d+1)n$, and hence, this significantly reduces its time complexity.

4.2. Word-Based SISO Projection Function

By applying the nonlinear projection approach, discussed in [30], to the DG nodes shown in Figure 2, we can develop the following nonlinear projection function that maps any node $\mathbf{p}(i,j)$ of Figure 2 to a processing element $\mathbf{PE}(o,u)$ in the resulting systolic/semisystolic array space:

$$\mathbf{PE}(o,u) = \mathbf{P}_s\ \mathbf{p}(i,j) \quad (15)$$

$$o = i \bmod k \quad (16)$$

$$u = j \bmod k \quad (17)$$

$$\mathbf{P}_s = [\ .\bmod k \quad .\bmod k\ ] \quad (18)$$

where "dot" is a place holder for the argument, as explained in [30].

The selected nonlinear projection function results in all the nodes of each green zone being mapped to a two-dimensional $k \times k$ semisystolic array. The semisystolic array is composed of k rows and k columns, as shown in Figure 3. Furthermore, the selected nonlinear projection function will map all the nodes of the yellow zones (the postprocessing nodes) to a one-dimensional $1 \times k$ postprocessing array, shown in Figure 4.

Figure 5 displays the block architecture of the word-based SISO semisystolic processor. It is composed of the semisystolic array block, the postprocessing array block, I/O registers, FIFO buffers, and two k-bit two-input MUXes, as well as $(k+2)$-bit two-input MUX. The k-bit two-input MUXes select between the input words of G and G' and their stored word values in FIFOs G and G'. On the other hand, the $(k+2)$-bit two-input MUX selects between the input words of E and its updated intermediate partial words stored in FIFO-E. The FIFO buffers FIFO-X, FIFO-Y, FIFO-P, FIFO-S, FIFO-G, FIFO-G', and FIFO-E sequentially feedback the resulting output words X, Y, P, S, G, G', and E of the semisystolic array block to the corresponding input words of the same array block. FIFO-X, FIFO-Y, FIFO-P, FIFO-S, FIFO-G, and FIFO-G' have a width size of k bits and a depth size of $W-1$, where $W = \lceil \frac{n}{k} \rceil$. FIFO-E has a width size of $k+2$ bits and the same depth size as the previous FIFOs. It is worth noticing that the upper registers of G, G', and E feed their word values to the corresponding inputs of the semisystolic array block starting from the most significant words. On the other hand, the registers of E_{2i-2} and E_{2i-1}, F_{2i-2} and F_{2i-1} at the left of the semisystolic array block supply their word values starting from the least significant words.

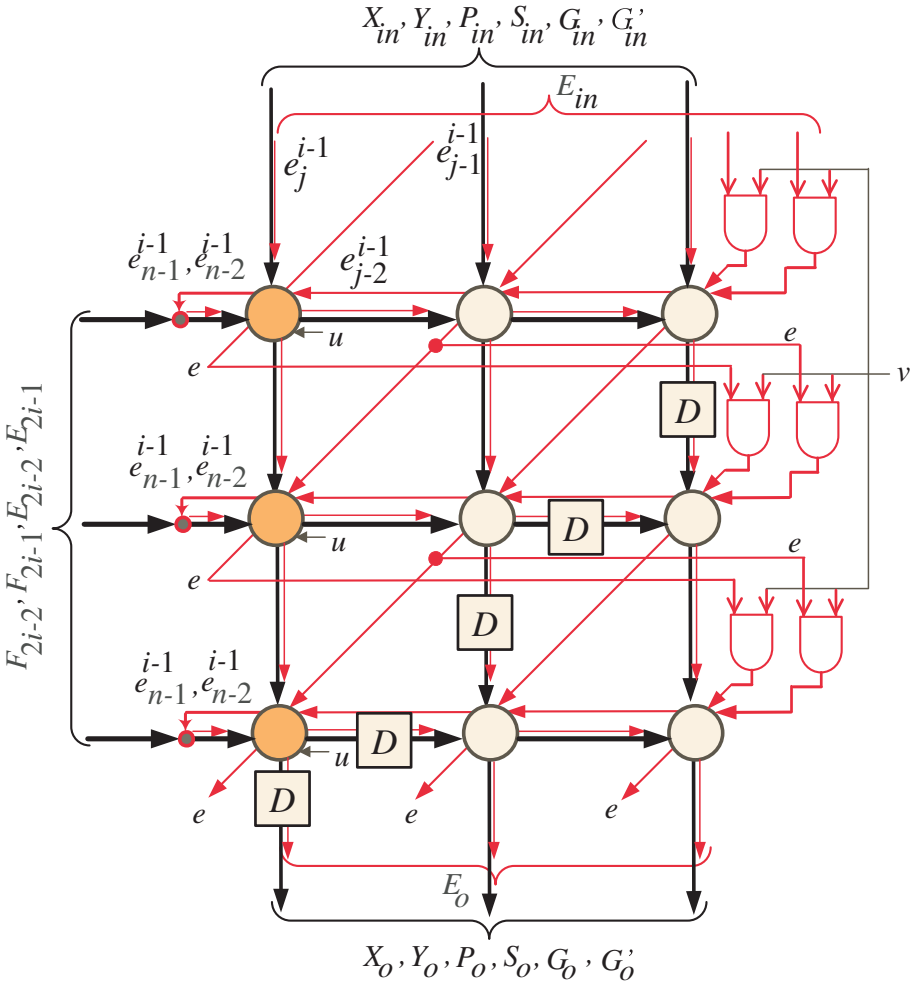

Figure 3. SISO semisystolic array for $k = 3$.

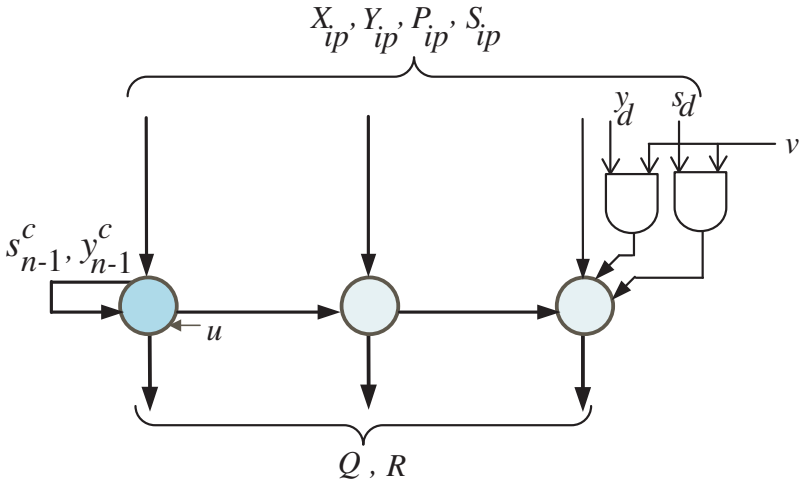

Figure 4. SISO postprocessing array for $k = 3$.

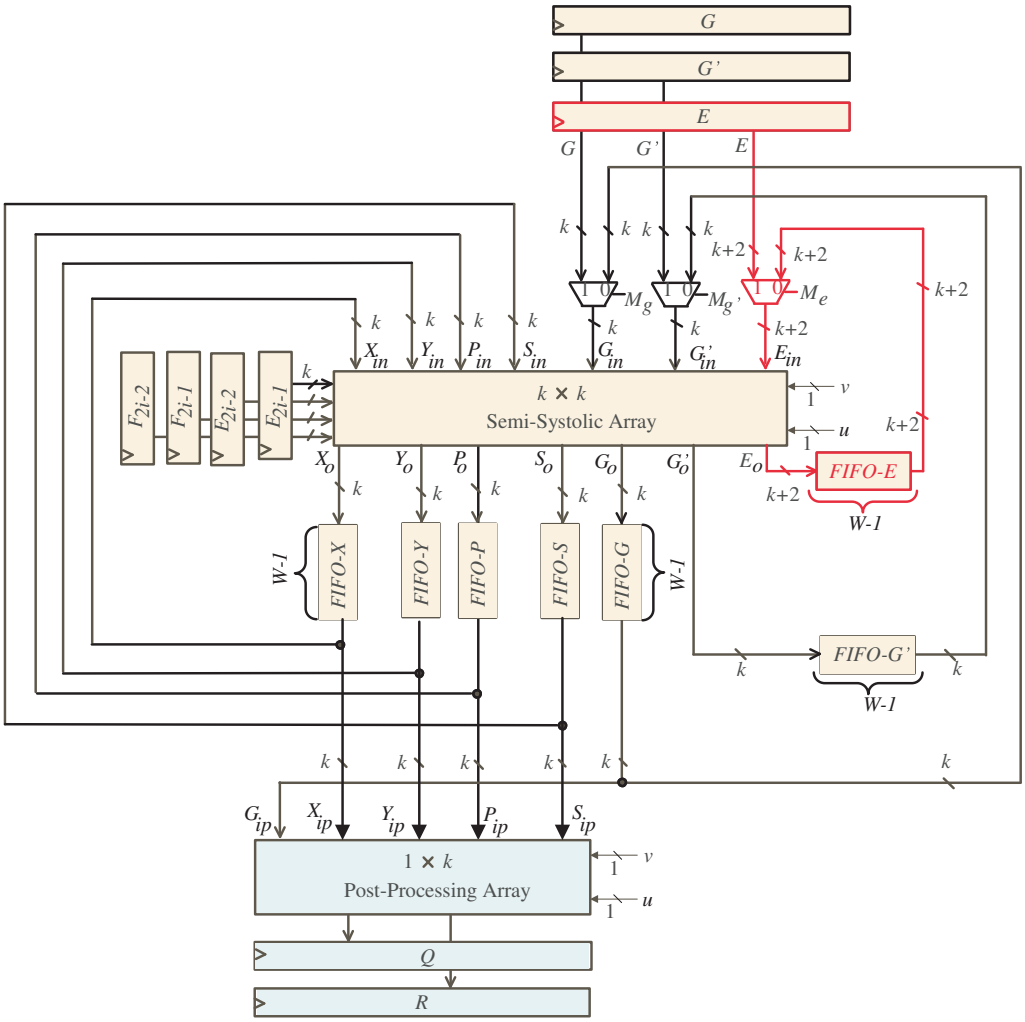

Figure 5. Word-based SISO semisystolic processor.

Perceiving the semisystolic array of Figure 3, we notice that it consists of two types of PEs: dark orange and light orange PEs. The logic details of the dark and light orange PEs are displayed in Figures 6 and 7, respectively. The two PEs almost have the same logic structure except that the dark orange PE has an extra tristate buffer that is enabled ($u = 0$) to pass the updated words of E_{n-1}^{i-1} and E_{n-2}^{i-1} at the proper time instances of $t = (i-1)\lceil \frac{n}{k} \rceil + 1, 1 \leq i < \lceil \frac{n}{k} \rceil$. Besides the input words E_{2i-2}, E_{2i-1}, F_{2i-2}, and F_{2i-1}, these words are horizontally transferred to the remaining PEs in the semisystolic array to update the intermediate partial words of X, Y, P, S, and E.

Similarly, observing the postprocessing array of Figure 4, we notice that it consists of two types of PEs: dark blue and light blue PEs. The logic details of the dark and light blue PEs are displayed in Figures 8 and 9. The two PEs almost have the same logic structure except that the dark blue PE has an extra tristate buffer that is enabled ($u = 0$) to pass the updated signals s_{n-1}^{c} and y_{n-1}^{c} at the proper time of $t = \left\lceil \frac{d}{k} \right\rceil \lceil \frac{n}{k} \rceil + 1$ to the remaining light blue PEs of the postprocessor array. These signals are used to calculate the words of the final products Q and R.

The partial signals x_j^d, y_{j-1}^c, p_j^d, and s_{j-1}^c resulting from the semisystolic array block are provided at the proper time to the corresponding inputs of the postprocessing array using the tristate buffers T_x, T_y, T_p, and T_s, as shown in Figures 8 and 9. For an odd field size n, the value of d will be less than the value of c ($d \neq c$), and in this case, the partial signals of y_{j-1}^c, s_{j-1}^c should be provided one clock cycle earlier than the signals x_j^d, p_j^d.

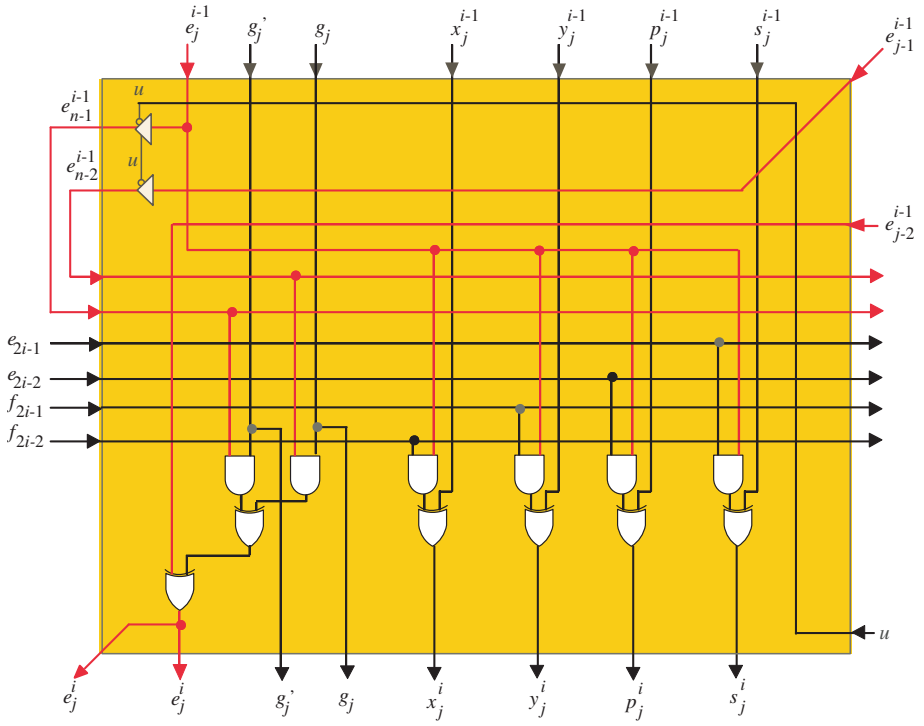

Figure 6. Logic details of the dark orange PEl.

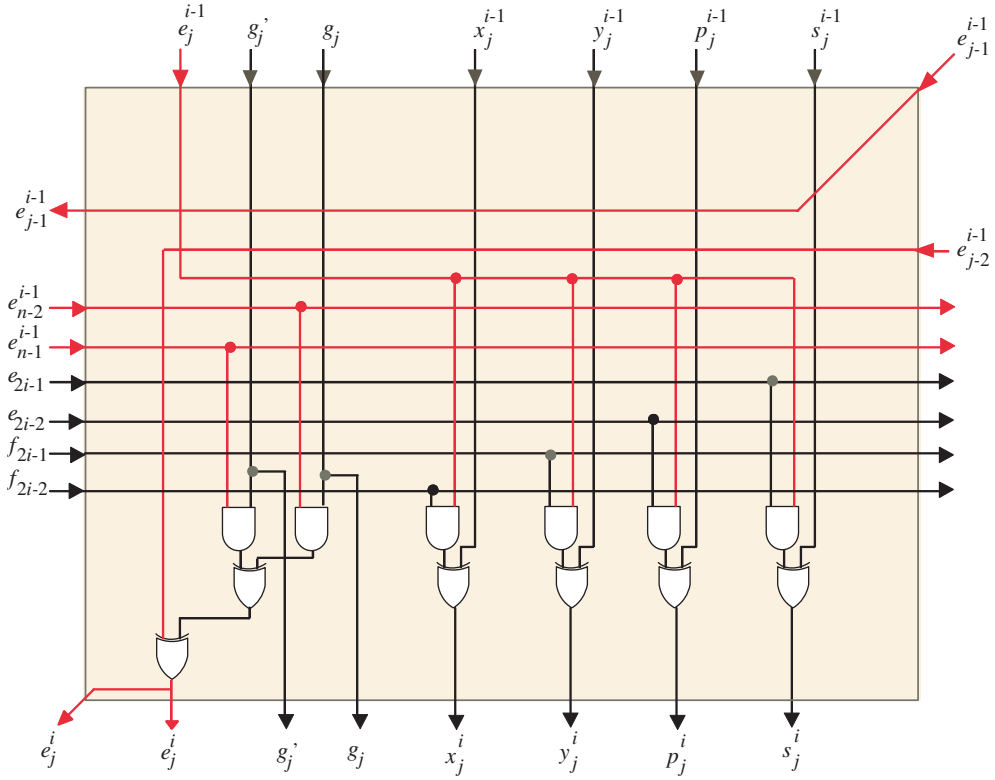

Figure 7. Logic details of the light orange PE.

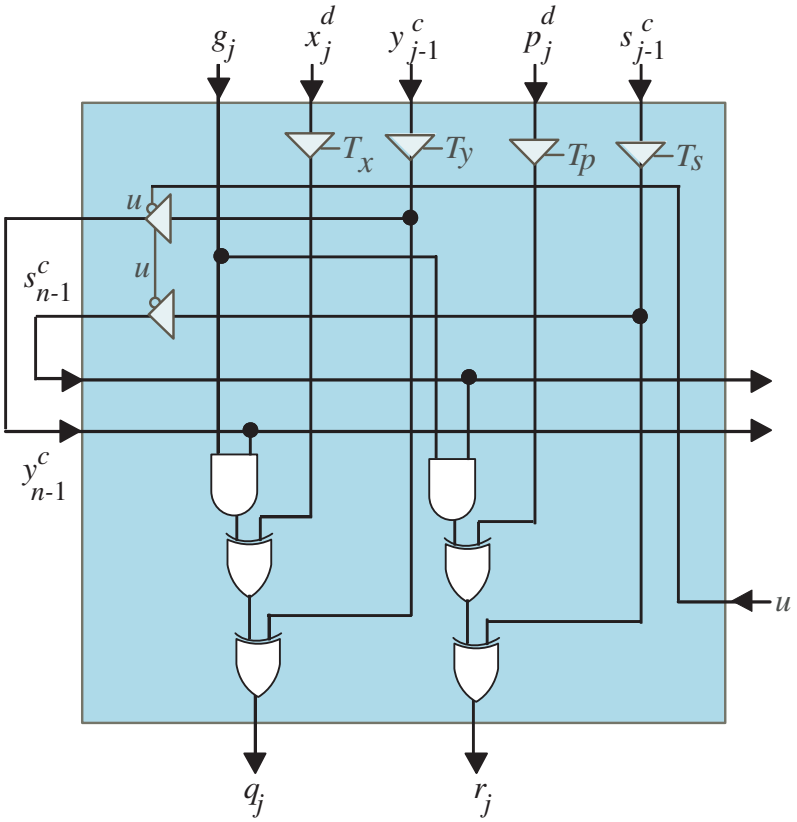

Figure 8. Logic details of the dark blue PE.

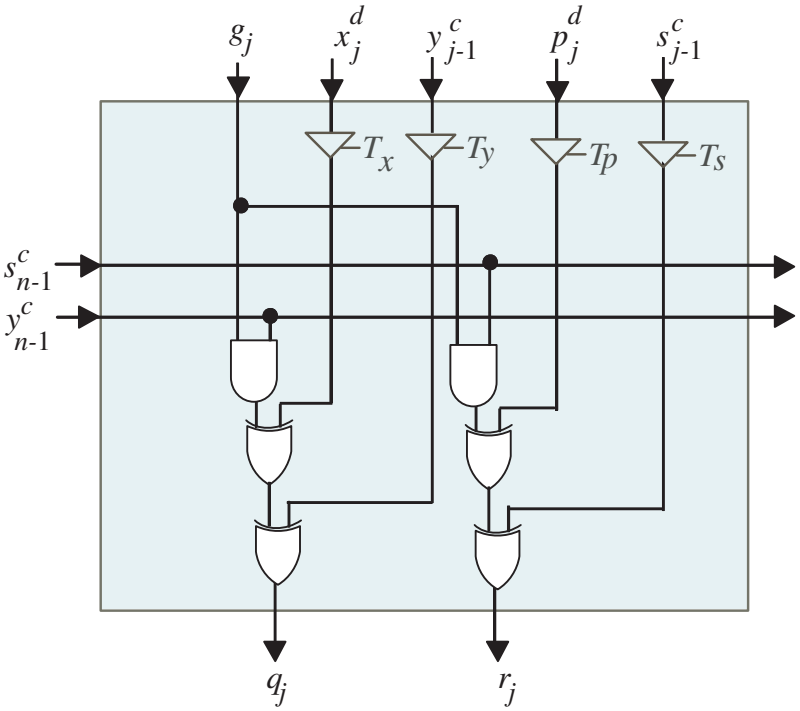

Figure 9. Logic details of the light blue PE.

The following provides the operation summary of the developed word-based SISO semisystolic array processor for different parameters of n and k as follows:

1. Through clock periods $1 < t \leq \lceil \frac{n}{k} \rceil$, the control signals of MUXes M_e, $M_{g'}$, and M_g, presented in Figure 5, are set to one to serially transfer the words of input operands E,

G, and G' to the corresponding inputs of the semisystolic array block (one word at each clock period) starting from the most significant words. Through the first clock period, the FIFO buffers of X, Y, P, and S are cleared to maintain zero initial values as denoted in Algorithm 1. As we notice from Figure 5, the FIFOs of X, Y, P, and S have a width size of k bits and a depth size of $W-1$, where $W = \lceil \frac{n}{k} \rceil$. The depth size of these buffers guarantees that the variables X, Y, P, and S have zero values through the intended clock periods, $1 < t \leq \lceil \frac{n}{k} \rceil$;

2. At clock periods $t = z\lceil \frac{n}{k} \rceil + 1, 0 \leq z < \lceil \frac{n}{k} \rceil$, the control signal u in all dark orange PEs of the semisystolic array shown in Figure 3 should be assigned a zero value ($u = 0$) to enable the tristate buffers designated in Figure 6 to serially transfer the updated words of E_{n-1}^{i-1} and E_{n-2}^{i-1} to the remaining light orange PEs in the array (one word at each clock period). Furthermore, through these clock periods, the words of E_{2i-2}, E_{2i-1}, F_{2i-2}, and F_{2i-1} should be serially transferred to all the PEs in the semisystolic array. The words of E_{2i-2}, E_{2i-1}, F_{2i-2}, and F_{2i-1} are transferred to the corresponding inputs in the semisystolic array block through the registers allocated at the left side of Figure 5;
3. At clock periods $t = z\lceil \frac{n}{k} \rceil, 1 \leq z < \lceil \frac{n}{k} \rceil$, the control signal v, shown in Figure 3, should be forced to have a zero value ($v = 0$) to force the least significant ζ bits of E, through the AND gates, to have zero values as indicated at the rightmost edge of the DG shown in Figure 2. Notice that in our case example of $n = 7$ and $k = 3$, ζ should equal two;
4. At clock periods $t > \lceil \frac{n}{k} \rceil$, the control signals of MUXes M_e, $M_{g'}$, and M_g are assigned zero values to transfer the following words to the corresponding inputs of the semisystolic array block: updated E words saved in FIFO-E, G' words saved in FIFO-G', and G words saved in FIFO-G. All of these words are transferred to the semisystolic array in a word-serial fashion (i.e., one word at each clock period). Through the same clock periods, the updated words X, Y, P, and S, saved in FIFO-X, FIFO-Y, FIFO-P, and FIFO-S, as well as the words of E_{n-1}^{i-1}, E_{n-2}^{i-1}, E_{2i-2}, E_{2i-1}, F_{2i-2}, and F_{2i-1} are serially passed (one word at each clock period) to the corresponding inputs of the semisystolic array block to compute serially (one word at each clock period) the intermediate partial results of X, Y, P, S, and E;
5. At clock period $t = \left\lceil \frac{d}{k} \right\rceil \lceil \frac{n}{k} \rceil + 1$, the control signal u in the dark blue PE of the postprocessing array, Figure 4, should be assigned a zero value ($u = 0$) to enable the tristate buffers designated in Figure 8 to serially transfer the updated bits of s_{n-1}^c and y_{n-1}^c to the remaining light blue nodes of the postprocessing array. The bits of s_{n-1}^c and y_{n-1}^c alongside the bits g_j, x_j^d, y_{j-1}^c, p_j^d, and s_{j-1}^c are used to determine the final words of the products Q and R;
6. At clock period $t = (\left\lceil \frac{d}{k} \right\rceil + 1)\lceil \frac{n}{k} \rceil$, the control signal v, indicated at the right side of the postprocessing array designated in Figure 4, should be forced to have a zero value ($v = 0$) to force the least significant bits of y_d and s_d, through the AND gates, to have zero values as shown at the rightmost edge of the DG shown in Figure 2;
7. Through clock periods $t \geq \left\lceil \frac{d}{k} \right\rceil \lceil \frac{n}{k} \rceil + 1$, the final output words Q and R produced from the postprocessing array will be loaded serially (one word at each clock period) in registers Q and R as presented in Figure 5.

It is worth noticing that the updated words of E are produced from the dark orange PEs of the semisystolic array, shown in Figure 3, starting from the second clock period. Therefore, the words of X, Y, P, S, E_{n-1}^{i-1}, E_{n-2}^{i-1}, E_{2i-2}, E_{2i-1}, F_{2i-2}, and F_{2i-1} should be delayed by one clock period to synchronize the operation. This is implemented by adding delay elements (represented by the D squares) to the semisystolic array, as shown in Figure 3.

5. Experimental Results and Discussion

In this section, we provide a qualitative and quantitative analysis of the proposed word-based SISO semisystolic multiplier-squarer structure and the efficient word-based serial multiplier structures available in the literature [45,50,60,61]. The qualitative analysis concentrates on developing analytical formulas for both the area and time complexities of the compared multiplier structures. On the other hand, the quantitative analysis focuses on providing the ASIC implementation results for all the compared designs to confirm the qualitative findings.

5.1. Analysis of the Estimated Area and Time Complexities

Table 1 provides the analytical formulas for both the hardware resource area and execution time complexities for the proposed structure, as well as the compared ones [45,50,60,61]. The hardware resource area was evaluated in terms of the counts of the logic gates/components (tristate buffers, 2-input AND gate, 2-input XOR gate, 1 bit 2-input multiplexers, and flip-flops). In contrast, the execution time complexity was evaluated in terms of the latency and Critical Path Delay (CPD). We notice from Table 1 that the formulas of the counted gates/components of the multiplier structures of Pan [45] and Xie [50] are almost of order $\mathcal{O}(nk)$, while being of order $\mathcal{O}(k^2)$, except for some gates/components of the proposed multiplier, in the case of the other multiplier structures. Therefore, the multiplier structures of Pan [45] and Xie [50] should have a higher area complexity over the other compared designs, including the proposed one. By further investigation of the gate count formulas of the proposed design and the designs of Hua [60] and Chen [61], we notice that the proposed design has a higher number of tristate buffers, AND gates, XOR gates, and MUXes. On the other hand, it has a lower number of flip-flops compared to them. The flip-flops formula of the proposed design is of order $\mathcal{O}(k\lceil n/k \rceil)$, while it is of order $\mathcal{O}(k^2)$ in the case of the designs of Hua [60] and Chen [61]. Therefore, for large values of k, the number of flip-flops of the proposed design will significantly decrease compared to the other designs. Based on the standard CMOS libraries' data, the area of the flop-flops, for most of the layout styles, is significantly higher than the area of the other basic gates. Thus, the significant reduction in the number of flip-flops in the design structure will lead to a considerable decrease in the design's overall area. The value of k at which the proposed design area outperforms the other design areas mainly depends on the amount of the field size n. For the recommend field size of $n = 409$, the word size of $k = 32$ leads to a significant decrease in the number of flip-flops of the proposed design and hence compensates for the increase in the area of the other gate counts. Further, it exceeds that by reducing the total area of the proposed design over the compared designs as proven by the real results in Table 2. Compared to the other designs, the proposed design performs both multiplication and squaring operations concurrently, while the other designs perform both operations in sequence; hence, reducing the area of the proposed design, for large word sizes, over that of the different designs is a notable achievement.

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

Design	Tristate	AND	XOR	MUXes	Flip-Flops	Latency	CPD
Xie [50]	0	$2nk$	$2nk + 6n - 6\frac{n}{k} + 6$	0	$4nk + 4n + 2k$	$2\lceil n/k \rceil + 2\lceil \log_2 k \rceil$	$2T_X$
Pan [45]	0	$n\sqrt{n}$	$\sqrt{nk}(2+n) + k$	0	$7n + n(\lceil \log n \rceil) + k + 3$	$2\lceil \sqrt{n/k} \rceil$	$T_A + (\lceil \log_2 k \rceil + 1)T_X$
Hua [60]	0	k^2	$k^2 + 4 - 5k + 1$ [(1)]	0	$2k^2 + 2k(\lceil n/k \rceil) + 4k + 1$	$6k\lceil n/k \rceil^2$	$T_A + 2T_X$
Chen [61]	0	$k^2 + k$	$k^2 + 2k$	$2k$ [(2)]	$2k^2 + 3k(\lceil n/k \rceil) + 2k$	$k + \lceil n/k \rceil^2 + \lceil n/k \rceil$	$T_A + T_X$
Proposed	$6k + 2$	$6k^2 + 4k + 2$	$6k^2 + 4k$	$3k + 2$	$(7k + 2)(\lceil n/k \rceil) - 4(k + 1)$	$(\lceil d/k \rceil + 1)\lceil n/k \rceil$	$\lceil \log_2(\frac{k}{2}) \rceil (T_A + 2T_X)$

[(1)] Area of the 3-input XOR gate as a 1.5×2-input XOR gate; [(2)] the multiplier of [61] uses switches that have the same transistor count as the 2-input MUX.

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

Multiplier	k	Latency	Area (A) [Kgates]	CPD [ps]	Time (T) [ns]	AT	Power (P) [nW]	Energy (E) [fJ]
Xie [50]	8	324	92.98	56.4	18.27	1699.0	225.56	4.12
	16	172	146.96	56.4	9.70	1425.5	375.5	3.64
	32	98	195.13	56.4	5.53	1078.5	477.4	2.64
Pan [45]	8	48	97.46	206.3	9.90	964.9	252.91	2.50
	16	36	123.93	244.4	8.80	1090.6	320.07	2.82
	32	24	164.34	282.5	6.78	1114.2	425.09	2.88
Hua [60]	8	259,584	7.99	73.4	19,053.47	152,237.2	4.35	82.88
	16	129,792	10.40	73.4	9526.73	99,077.9	5.85	55.73
	32	64,896	19.91	73.4	4763.37	94,838.7	11.15	53.11
Chen [61]	8	11,946	10.16	55.2	659.42	6699.7	5.11	3.37
	16	4678	13.51	55.2	203.03	2742.9	8.38	1.70
	32	1572	26.58	55.2	86.77	2306.4	15.95	1.38
Proposed	8	1325	12.87	96.0	127.2	1637.1	5.91	0.75
	16	364	13.98	145.0	52.8	737.9	8.95	0.47
	32	104	15.34	207.0	21.5	330.2	9.14	0.19

By observing the latency and CPD formulas in Table 1, we notice that the design of Pan [45] has lower latency and the design of Hua [60] has the most considerable latency compared to the remaining designs, including the proposed one. The quantitative analysis presented in the following subsection shows that the developed multiplier-squarer structure has lower latency than the multiplier structures of Hua [60] and Chen [61] and higher latency compared to the multiplier structures of Xie [50] and Pan [45] for field size $n = 409$. We notice also from Table 1 that as the word size k increases, the latency of all designs significantly decreases as the latency is inversely proportional to k.

By examining the formulas of the CPD, we notice that the multiplier structures of Xie [50], Hua [60], and Chen [61] have a constant and lower CPD for all values of word size k. In contrast, the CPD values of Pan [45] and the proposed design mainly increase as k increases. Since the latency of all designs significantly reduces as k increases and the CPD has constant values or increases as k increases, we cannot precisely conclude from the qualitative analysis which design should have the best execution time. The quantitative analysis provided in the following subsection will show which multiplier structure provides better execution time complexity.

5.2. ASIC Implementation Results

ASIC designs of the published works of Xie [50], Pan [45], Hua [60], Chen [61], as well as our proposed design were implemented using VHDL for field size $n = 409$ and embedded applications word sizes k of 8, 16, and 32. The developed code was synthesized using Synopsys design compiler 2005.09-SP2 with the NanGate (15 nm, 0.8 V) Open Cell Library. The simulations used Mentor Graphics ModelSim SE 6.0a tools and produced a Switching Activity Interchange Format (SAIF) file to obtain the power report. The simulations went through 300 possible 32 bit input combinations. Post-layout simulations were used to accurately model the delays, power, and area.

Table 2 summarizes the ASIC implementation results for the proposed design and the previously published designs. The design performance metrics were: (1) latency: the total time steps required to produce the final results; (2) Area (A): the required hardware resources estimated in terms of the two-input NAND gate; (3) Time (T): the execution time required to produce the output; (4) Area–Time product (AT): the product of A and T′

(5) Power (P): the power consumption; (6) Energy (E): the consumed energy obtained as the product of P and T.

It is important to note that the published word-serial multipliers of [45,50,60,61] only perform multiplication operation, while the proposed word-serial multiplier-squarer concurrently performs both multiplication and squaring operations. Therefore, to have a fair comparison, the word-serial multipliers of [45,50,60,61] should sequentially compute both operations. The execution of multiplication and squaring operations in sequence resulted in doubling the latency, T, P, and E.

Examining Table 2, we note the following:

(1) The multiplier structure of Pan [45] achieved the lowest latency for all word sizes compared to the other structures, including the proposed one. We also observed that the latency significantly decreased, for all multiplier structures, as the word size k increased;

(2) The multiplier structure of Chen [61] had the lowest CPD for all values of k.

For a better interpretation of the achieved results, we visualized the area, time, Area–Time product (AT), power, and energy results using the charts displayed in Figures 10–14, respectively.

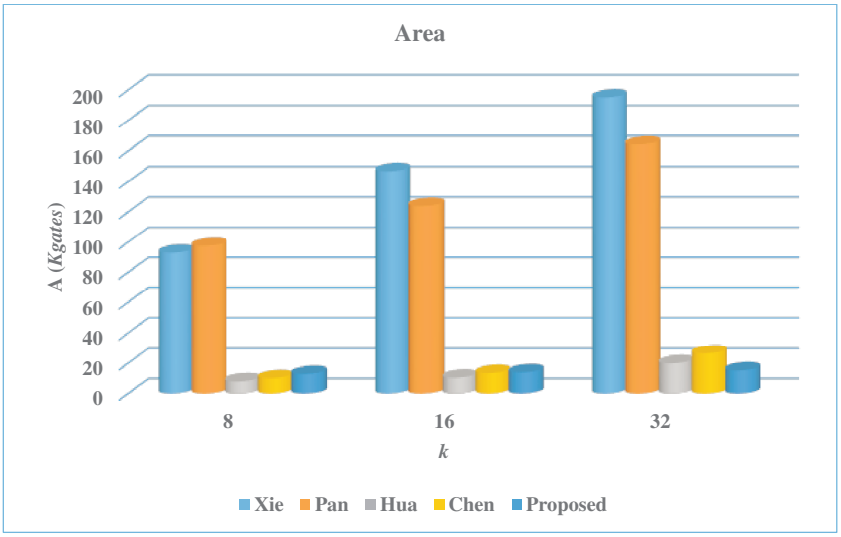

Figure 10. Area results.

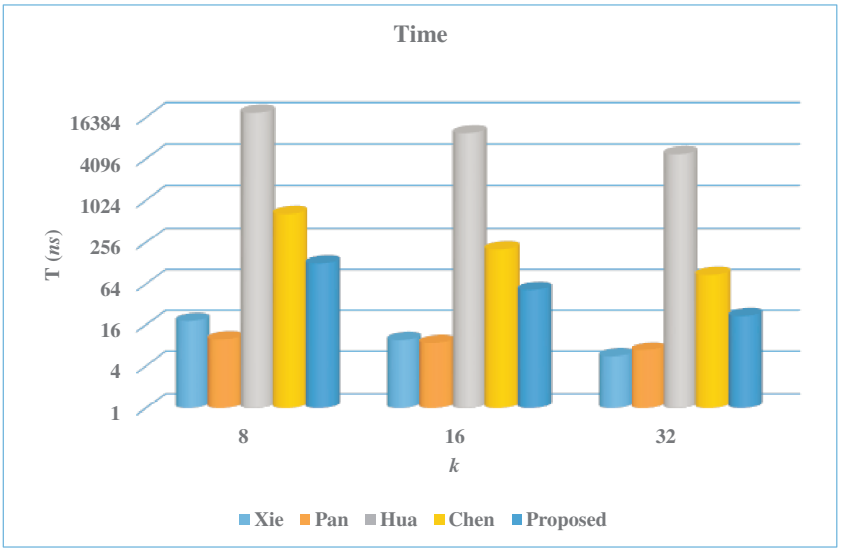

Figure 11. Time results.

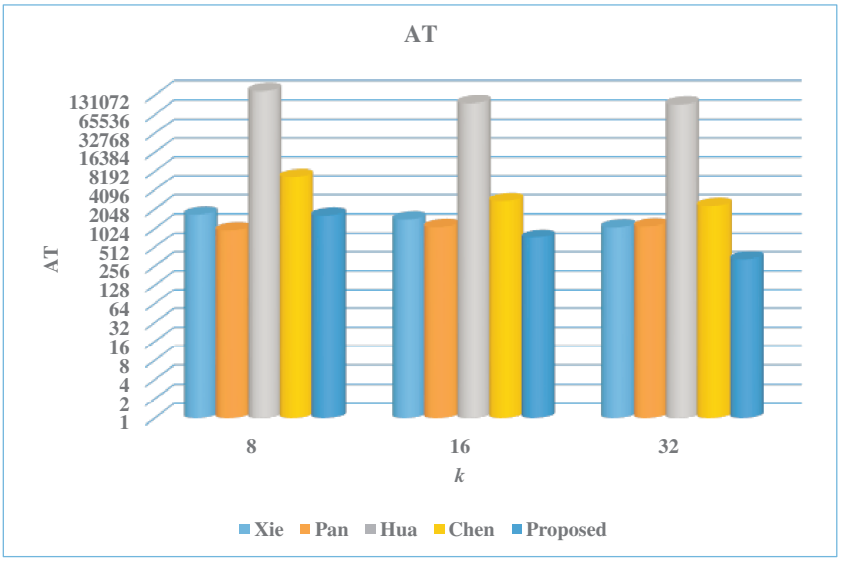

Figure 12. AT results.

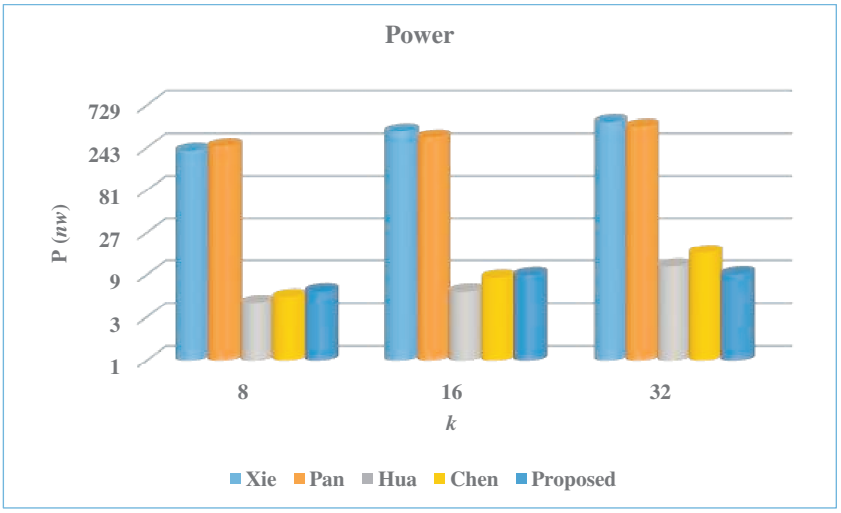

Figure 13. Consumed power results.

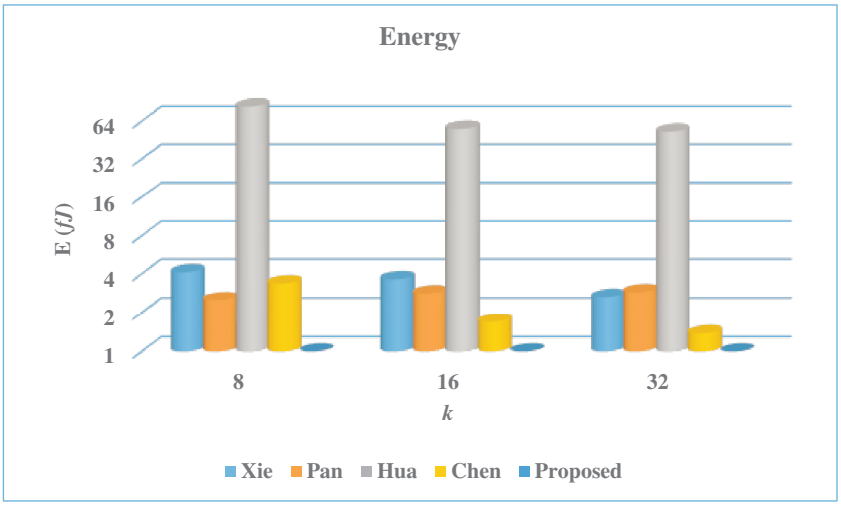

Figure 14. Consumed energy results.

From Figure 10, we notice that the multiplier structure of Hua [60] achieved a lower area (A) at word sizes $k = 8$ and $k = 16$, by at least 21.3% and 23.0%, respectively. In contrast, at the word size of $k = 32$, the proposed multiplier-squarer structure achieved the smallest area by at least 22.95%. The reduction of the proposed structure's area was mainly attributed to the significant decrease in the number of flip-flops as the word size increased, as we discussed before in the previous subsection.

From Figure 11, we notice that the multiplier structure of Pan [45] achieved a significant reduction in time (T) at word sizes $k = 8$ and $k = 16$ due to its significantly lower latency. On the other hand, the multiplier structure of Xie [50] realized a slight reduction in T, at word size $k = 32$, compared to that of Pan [45]. Due to the fluctuations in the obtained results of the latency, A, and T among the different multiplier structures at different word size, we added the Area–Time product (AT) metric to Table 2. The AT metric enabled us to discover the optimal multiplier structures in terms of area and time.

From Figure 12, we notice that the proposed multiplier-squarer structure achieved the lowest AT for word sizes $k = 16$ and $k = 32$. That was attributed to the modest amounts of its latency and area at these word sizes. The proposed multiplier-squarer fulfilled a decrease in AT at word sizes of $k = 16$ and $k = 32$ by at least 32.3% and 70.4%.

From Figure 13, we notice that the multiplier structure of Hua [60] had lower Power consumption (P) at word sizes $k = 8$ and $k = 16$ due mainly to the significant reduction in area. The decrease in area resulted in a reduction of the extracted parasitic capacitance and, hence, reduced the circuit's switching activities. The proposed multiplier-squarer structure had slightly higher power consumption over the multiplier structure of Hua [60] at the same word sizes of $k = 8$ and $k = 16$. In contrast, it had a slightly lower power consumption than the multiplier of Hua [60] at word size $k = 32$ due to its smaller area at this word size.

Despite having lower power consumption at word sizes $k = 8$, and $k = 16$, the multiplier of Hua [60] had significant values of consumed energy, as shown in Figure 14, due to the tremendous amounts of its execution Time (T). The proposed multiplier-squarer had the lowest consumed energy, as shown in Figure 14, at all word sizes due to the reasonable values of its execution Time (T), as well as the lower values of the consumed power. The proposed multiplier-squarer reduced the consumed energy at word sizes $k = 8$, $k = 16$, and $k = 32$ by at least 70%, 72.4%, and 86.2%, respectively.

From the previous discussion, we can conclude that the presented word-serial multiplier-squarer had a smaller AT for word sizes $k = 16$ and $k = 32$ and had less consumed energy for all word sizes. Thus, it is more suitable for realizing cryptographic primitives in RFID tags, especially semipassive and active tags with reasonable restrictions on area and delay and large restrictions on consumed energy. The design is more suitable for semipassive and active tags than battery-powered RFID tags (semipassive and active tags), which have restricted energy, contrary to energy-harvesting RFID tags (passive tags), which can accommodate endless energy with restricted power.

6. Conclusions and Future Work

This paper introduced a capable word-based SISO semisystolic processor to simultaneously compute multiplication and squaring operations over GF(2^n). Both operations share the same hardware core, resulting in significant savings in the hardware resource area and the consumed energy, especially for large word sizes. We used a systematic approach to find valid nonlinear scheduling and projection functions that can be applied to the DG to produce the expected word-based SISO semisystolic processor. The chosen functions provide more adaptability to adjust the utilization of the processor besides its execution time. The paper also presented a qualitative and quantitative analysis of the proposed multiplier-squarer structure and the efficient multiplier structures in the literature. The acquired results for the recommended field size $n = 409$ proved that the proposed multiplier-squarer structure had the lowest Area–Time product (AT) for word sizes $k = 16$ and $k = 32$ and consumed less energy for all embedded word sizes. Therefore, it is more

suitable to realize cryptographic primitives in RFID-based applications with moderate restrictions on area and delay and significant restrictions on wasted energy. In future work, we will improve the hardware design to have lower area and delay complexities to be suitable for applications with high restrictions on these design parameters.

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. Both authors 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.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

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
ECC	Elliptic Curve Cryptography
SISO	Serial-In/Serial-Out
SIPO	Serial-In/Parallel-Out
PISO	Parallel-In/Serial-Out
CPD	Critical Path Delay
AT	Area–Time

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. 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]
3. Atzori, L.; Iera, A.; Morabito, G. The Internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805. [CrossRef]
4. Fan, K.; Gong, Y.; Liang, C.; Li, H.; Yang, Y. Lightweight and ultralightweight RFID mutual authentication protocol with cache in the reader for IoT in 5G. *Secur. Commun. Netw.* **2016**, *9*, 3095–3104. [CrossRef]
5. Juels, A. RFID security and privacy: A research survey. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 381–394. [CrossRef]
6. Baashirah, R.; Abuzneid, A. Survey on prominent RFID authentication protocols for passive tags. *Sensors* **2018**, *18*, 3584. [CrossRef]
7. Yao, W.; Chu, C.H.; Li, Z. The adoption and implementation of RFID technologies in healthcare: A literature review. *J. Med. Syst.* **2012**, *36*, 3507–3525. [CrossRef] [PubMed]
8. Zhu, F.; Li, P.; Xu, H.; Wang, R. A Novel Lightweight Authentication Scheme for RFID-Based Healthcare Systems. *Sensors* **2020**, *20*, 4846. [CrossRef]
9. Wu, D.L.; Wing, W.; Yeung, D.S.; Ding, H.L. A brief survey on current RFID applications. In Proceedings of the 2009 International Conference on Machine Learning and Cybernetics, Baoding, China, 12–15 July 2009; Volume 4, pp. 2330–2335.
10. Kaur, M.; Sandhu, M.; Mohan, N.; Sandhu P. S. RFID technology principles, advantages, limitations & its applications. *Int. J. Comput. Electr. Eng.* **2011**, *3*, 151.
11. Rahman, F.; Bhuiyan, M.Z.A.; Ahamed, S.I. A privacy preserving framework for RFID based healthcare systems. *Future Gener. Comput. Syst.* **2017**, *72*, 339–352. [CrossRef]
12. Fan, K.; Jiang, W.; Li, H.; Yang, Y. Lightweight RFID protocol for medical privacy protection in IoT. *IEEE Trans. Ind. Inform.* **2018**, *14*, 1656–1665. [CrossRef]

13. Fan, K.; Zhu, S.; Zhang, K.; Li, H.; Yang, Y. A lightweight authentication scheme for cloud-based RFID healthcare systems. *IEEE Netw.* **2019**, *33*, 44–49. [CrossRef]
14. Zhao, Z. A secure RFID authentication protocol for healthcare environments using elliptic curve cryptosystem. *J. Med. Syst.* **2014**, *38*, 46. [CrossRef] [PubMed]
15. Zhang, Z.; Qi, Q. An efficient RFID authentication protocol to enhance patient medication safety using elliptic curve cryptography. *J. Med. Syst.* **2014**, *38*, 47. [CrossRef] [PubMed]
16. Farash, M.S.; Nawaz, O.; Mahmood, K.; Chaudhry, S.A.; Khan, M.K. A provably secure RFID authentication protocol based on elliptic curve for healthcare environments. *J. Med. Syst.* **2016**, *40*, 165. [CrossRef] [PubMed]
17. Jin, C.; Xu, C.; Zhang, X.; Zhao, J. A secure RFID mutual authentication protocol for healthcare environments using elliptic curve cryptography. *J. Med. Syst.* **2015**, *39*, 24. [CrossRef]
18. Jin, C.; Xu, C.; Zhang, X.; Li, F. A secure ECC-based RFID mutual authentication protocol to enhance patient medication safety. *J. Med. Syst.* **2016**, *40*, 12. [CrossRef]
19. Qiu, S.; Xu, G.; Ahmad, H.; Wang, L. A robust mutual authentication scheme based on elliptic curve cryptography for telecare medical information systems. *IEEE Access* **2017**, *6*, 7452–7463. [CrossRef]
20. Abbasinezhad-Mood, D.; Nikooghadam, M. Efficient design of a novel ECC-based public key scheme for medical data protection by utilization of NanoPi fire. *IEEE Trans. Reliab.* **2018**, *67*, 1328–1339. [CrossRef]
21. Kumar, V.; Ahmad, M.; Kumari, A. A secure elliptic curve cryptography based mutual authentication protocol for cloud-assisted TMIS. *Telemat. Inform.* **2019**, *38*, 100–117. [CrossRef]
22. 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]
23. Srivastava, K.; Awasthi, A.K.; Kaul, S.D.; Mittal, R. A hash based mutual RFID tag authentication protocol in telecare medicine information system. *J. Med. Syst.* **2015**, *39*, 153. [CrossRef]
24. Li, C.T.; Weng, C.Y.; Lee, C.C. A secure RFID tag authentication protocol with privacy preserving in telecare medicine information system. *J. Med. Syst.* **2015**, *39*, 77. [CrossRef]
25. Aghili, S.F.; Mala, H.; Shojafar, M.; Peris-Lopez, P. Laco: Lightweight three-factor authentication, access control and ownership transfer scheme for e-health systems in IoT. *Future Gener. Comput. Syst.* **2019**, *96*, 410–424. [CrossRef]
26. Safkhani, M.; Vasilakos, A. A new secure authentication protocol for telecare medicine information system and smart campus. *IEEE Access* **2019**, *7*, 23514–23526. [CrossRef]
27. 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]
28. Safkhani, M.; Bendavid, Y.; Rostampour, S.; Bagheri, N. On Designing Lightweight RFID Security Protocols for Medical IoT. *IACR Cryptol. EPrint Arch.* **2019**, *2019*, 851.
29. Aghili, S.F.; Mala, H. Security analysis of an ultra-lightweight RFID authentication protocol for m-commerce. *Int. J. Commun. Syst.* **2019**, *32*, e3837. [CrossRef]
30. Gebali, F. *Algorithms and Parallel Computers*; Wiley Online Library: New York, NY, USA, 2011.
31. Ibrahim, A.; Elsimary, H.; Gebali, F. New systolic array architecture for finite field division. *IEICE Electron. Express* **2018**, *15*, 1–11. [CrossRef]
32. Ibrahim, A.; Elsimary, H.; Aljumah, A.; Gebali, F. Reconfigurable hardware accelerator for profile hidden Markov models. *Arab. J. Sci. Eng.* **2016**, *41*, 3267–3277. [CrossRef]
33. Ibrahim, A. Scalable digit-serial processor array architecture for finite field division. *Microelectron. J.* **2019**, *85*, 83–91. [CrossRef]
34. Ibrahim, A.; Alsomani, T.; Gebali, F. Unified Systolic Array Architecture for Field Multiplication and Inversion Over GF(2^m). *Comput. Electr. Eng. J.* **2017**, *61*, 104–115. [CrossRef]
35. Ibrahim, A.; Alsomani, T.; Gebali, F. New Systolic Array Architecture for Finite Field Inversion. *IEEE Can. J. Electr. Comput. Eng.* **2017**, *40*, 23–30.
36. Gebali, F.; Ibrahim, A. Low space-complexity and low power semisystolic multiplier architectures over GF(2^m) based on irreducible trinomial. *Microprocess. Microsyst.* **2016**, *40*, 45–52. [CrossRef]
37. Kim, K.W.; Lee, H.H.; Kim, S.H. Efficient combined algorithm for multiplication and squaring for fast exponentiation over finite fields GF(2^m). In Proceedings of the 7th International Conference on Emerging Databases, LNEE 461, Busan, Korea, 7–9 August 2017; pp. 50–57.
38. 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]
39. Kim, K.W.; Jeon, J.C. Polynomial Basis Multiplier Using Cellular Systolic Architecture. *IETE J. Res.* **2014**, *60*, 194–199. [CrossRef]
40. Choi, S.; Lee, K. Efficient ssystolic modular multiplier/squarer for fast exponentiation over GF(2^m). *IEICE Electron. Express* **2015**, *12*, 1–6. [CrossRef]
41. 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]
42. Kim, C.H.; Hong, C.P.; Kwon, S. A digit-serial multiplier for finite Field GF(2^m). *IEEE Trans. Very Large Scale Integr. Syst.* **2005**, *13*, 476–483.

43. 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. Syst.* **2010**, *18*, 847–852. [CrossRef]
44. 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]
45. 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]
46. 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.
47. Hariri, A.; Reyhani-Masoleh, A. Digit-serial structures for the shifted polynomial basis multiplication over binary extension fields. In *International Workshop on the Arithmetic of Finite Fields (WAIFI)*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 103–116.
48. Kumar, S.; Wollinger, T.; Paar, C. Optimum digit serial multipliers for curve-based cryptography. *IEEE Trans. Comput.* **2006**, *55*, 1306–1311. [CrossRef]
49. Lee, C.Y. Super digit-serial systolic multiplier over GF(2^m). In Proceedings of the 2012 Sixth International Conference on Genetic and Evolutionary Computing, Kitakyushu, Japan, 25–28 August 2012; pp. 509–513.
50. 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]
51. Namin, A.H.; Wu, H.; Ahmadi, M. A word-level finite field multiplier using normal basis. *IEEE Trans. Comput.* **2011**, *60*, 890–895. [CrossRef]
52. 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]
53. 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.
54. Orlando, G.; Paar, C. A super-serial galois fields multiplier for FPGAs and its application to public-key algorithms. In Proceedings of the Seventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines (Cat. No. PR00375), Napa Valley, CA, USA, 23 April 1999; pp. 232–239.
55. 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.
56. Gebali, F.; Ibrahim, A. Efficient Scalable Serial Multiplier Over GF(2^m) Based on Trinomial. *IEEE Trans. Very Large Scale Integr. Syst.* **2015**, *23*, 2322–2326. [CrossRef]
57. 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]
58. 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]
59. Kim, K.W.; Lee, J.D. Efficient unified semisystolic arrays for multiplication and squaring over GF(2^m). *IEICE Electron. Express* **2017**, *14*, 1–10. [CrossRef]
60. 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.
61. 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]