mathematics MDPI

Article

Compact Word-Serial Modular Multiplier Accelerator Structure for Cryptographic Processors in IoT Edge Nodes with Limited Resources

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: aa.mohamed@psau.edu.sa

Abstract: IoT is extensively used in many infrastructure applications, including telehealth, smart homes, smart grids, and smart cities. However, IoT has the weakest link in system security since it often has low processing and power resources. It is important to implement the necessary cryptographic primitives in these devices using extremely efficient finite field hardware structures. Modular multiplication is the core of cryptographic operators. Therefore, we present, in this work, a word-serial modular multiplier accelerator structure that provides the system designer with the ability to manage areas, delays, and energy consumption through selecting the appropriate embedded processor word size l. The modularity and regularity of the suggested multiplier structure makes it more suitable for implementation in ASIC technology. The ASIC implementation results indicates that the offered multiplier structure achieves area reduction compared to the competitive existing multiplier structures that vary from 76.2% to 98.5% for $l = 8$, from 73.1% to 98.1% for $l = 16$, and from 82.9% to 98.3% for $l = 32$. Moreover, the energy reduction varies from 61.2% to 98.8% for $l = 8$, from 67.7% to 98.3% for $l = 16$, and from 76.1% to 98.8% for $l = 32$. These results indicate that the proposed modular multiplier structure significantly outperforms the competitive ones, in terms of area and consumed energy, making it more suitable for utilization in resource-constrained IoT edge devices.

Keywords: modular multipliers; embedded security; IoT network; hardware security; parallel computing; cryptography

MSC: 11T06

check for updates

Citation: Ibrahim, A.; Gebali, F. Compact Word-Serial Modular Multiplier Accelerator Structure for Cryptographic Processors in IoT Edge Nodes with Limited Resources. *Mathematics* **2022**, *10*, 848. https://doi.org/10.3390/math10050848

Academic Editors: Luis Hernández Encinas and Víctor Gayoso Martínez

Received: 24 February 2022
Accepted: 6 March 2022
Published: 7 March 2022

1. Introduction and Related Work

The Internet of Things (IoT) is a broad network of physical devices that are equipped with sensors, software, electronics, and a network that allows them to share data and execute tasks. IoT is a promising technology that will shape our future by providing intelligent solutions in different applications, such as smart homes, smart cities, self-driven cars, smart farming, smart grids, and telehealth. To better understand how this system works, let us look at the IoT network topology in an IoT application, such as telehealth. It is one of the many emerging infrastructure applications that relay on IoT technology, to provide services to remote users, such as stay-at-home patients, and providing quality healthcare to remote communities [1,2]. Figure 1 shows a telehealth system that relies on IoT edge devices to deliver healthcare to remote locations.

The main entities of a telehealth system are: (a) A server that could be a hospital or medical center, and naturally considered a hardware root-of-trust (HRoT) due to the layered security measures implemented. (b) Internet cloud, which is, in general, an insecure communication medium. (c) A gateway that provides an interface between the IoT edge devices and the internet. (d) Edge IoT devices that comprise sensors and actuators to measure and deliver medications to remote patients. (e) Mobile devices that allow healthcare practitioners (doctors/nurses) to remotely connect to the telehealth system. It is clear from the figure that there are many opportunities for attacks due to the use of diverse hardware platforms, diverse operating systems, insecure wireless communication media, limited processing power for many system entities, and the sheer number of people involved in the operation of the system [3,4].

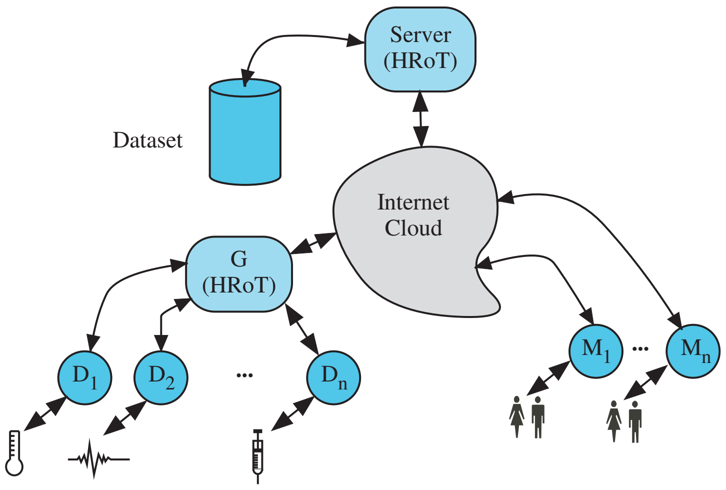

Figure 1. Telehealth network model.

Securing any system (telehealth or otherwise) implies many features that include integrity, confidentiality, authentication, non-repudiation, and availability. These security features are implemented using fundamental encryption algorithms, such as elliptic curve cryptography (ECC) and Rivest—Shamir—Adleman (RSA) algorithms. Given the power and delay restrictions for most of the devices used in most IoT systems, elliptic curve cryptography (ECC) is the encryption technique of choice due to its high level of security with shorter key lengths compared to common approaches, such as RSA [5]. An essential operation in ECC arithmetic is modular multiplication. There is an extensive body of literature covering modular multiplications in both prime fields $GF(p)$ and binary extension fields $GF(2^m)$. Most of the proposed multipliers possess high area and delay complexities, which make them unsuitable for resource-constrained IoT edge devices [6–8]. To overcome these limitations, several authors developed word-serial modular multipliers [9–12]. Systolic approaches were reported in [9,13–15] and non-systolic designs were report in [16–19]. Other authors attempted to save power and area by merging the modular multiplication and modular squaring operations [7,8,20]. However, the resulting structures were not suitable for resource-constrained IoT devices due to their high area and power requirements.

Most of the reported modular multiplier structures are classified as one-of-a kind structures. Ad hoc approaches are adopted with no consideration on how the structure can be modified to optimize system performance parameters, such as latency, throughput, power, and area requirements. The authors of this article presented a systematic methodology for implementing the modular multiplication algorithm based on the algebraic approach first proposed by the first author [21]. The systematic methodology applied linear mappings to obtain modular multiplier structures. However, linear mappings have limited abilities, both in terms of the number of parallel processing elements (PE) and also the timing strategies that could be developed.

This paper proposes using nonlinear techniques for mapping the algorithm onto parallel PEs and to obtain more flexible timing strategies. The goal of the paper is to obtain a word-serial processor accelerator for modular multiplication operations. The resulting structure gives the designer the ability to control the PE workload and the algorithm latency. The experimental results confirm that the proposed multiplier outperforms the efficient word-serial ones previously reported on in the literature, in terms of area and consumed energy for various embedded word-sizes. These design features make the proposed design more suitable for embedded applications and other resource-constrained IoT applications.

The outlining of the paper is as follows. Section 2 briefly describes the adopted modular multiplication algorithm and exhibits the details of its dependency graph. Section 3 presents the followed approach to extract the modular multiplier word-serial accelerator structure with its related logic details. Section 4 shows the realized implementation results. Section 5 concludes the recommended work.

2. Algorithm of Interleaved Modular Multiplication

We can perform modular multiplication over $GF(2^m)$ by multiplying two polynomials $P(\gamma)$ and $Q(\gamma)$ and reducing the result using the reduction polynomial $T(\gamma)$ as:

$$S(\gamma) \quad = \quad P(\gamma)Q(\gamma) \bmod T(\gamma) \tag{1}$$

the general polynomial format of $P(\gamma)$, $Q(\gamma)$, and $T(\gamma)$ can be given as:

$$P(\gamma) = \sum_{j=0}^{m-1} p_j\gamma^j = (p_0 + p_1\gamma^1 + \cdots + p_{m-1}\gamma^{m-1}) \tag{2}$$

$$Q(\gamma) = \sum_{j=0}^{m-1} q_j\gamma^j = (q_0 + q_1\gamma^1 + \cdots + q_{m-1}\gamma^{m-1}) \tag{3}$$

$$T(\gamma) = \sum_{j=0}^{m} t_j\gamma^j = (t_0 + t_1\gamma^1 + \cdots + t_m\gamma^m) \tag{4}$$

with $p_j, q_j, t_j \in GF(2)$.

By replacing $Q(\gamma)$ in Equation (1) with its polynomial format given in Equation (3), Equation (1) can be represented as follows:

$$S(\gamma) = P(\gamma)(q_0 + q_1\gamma^1 + \cdots + q_{m-1}\gamma^{m-1}) \bmod T(\gamma) \tag{5}$$

we can arrange Equation (1) in the interleaved form as:

$$\begin{aligned} S(\gamma) \quad = \quad & q_0[P(\gamma) \bmod T(\gamma)] + q_1[\gamma^1 P(\gamma) \bmod T(\gamma)] + \\ & + \cdots + q_{m-1}[\gamma^{m-1}P(\gamma) \bmod T(\gamma)] \end{aligned} \tag{6}$$

We choose to drop γ from polynomials $P(\gamma)$, $Q(\gamma)$, $S(\gamma)$, and $T(\gamma)$ to simplify the upcoming expressions. Investigating Equation (6), we notice that the multiplication product can be produced by accumulating the terms $q_i[P\gamma^i \bmod T]$, with $0 \le i \le m-1$.

Suppose $P^i = P\gamma^i \bmod T$, we can represent P^{i+1} in terms of P^i as $P^{i+1} = P\gamma^{i+1} \bmod T = [P\gamma^i]\gamma \bmod T = P^i\gamma \bmod T$. Thus, the recursive form of P^{i+1} can be represented as:

$$\begin{aligned}
P^{i+1} &= P^i\gamma \bmod T \\
\sum_{j=0}^{m-1} p_j^{i+1}\gamma^j &= (\sum_{j=0}^{m-1} p_j^i\gamma^j)\gamma \bmod T \\
&= \sum_{j=0}^{m-1}(p_j^i\gamma^{j+1}) \bmod T \\
&= p_{m-1}^i[\gamma^m \bmod T] + \sum_{j=0}^{m-2} p_j^i\gamma^{j+1} \\
&= p_{m-1}^i \sum_{j=0}^{m-1} t_j\gamma^j + \sum_{j=0}^{m-1} p_{j-1}^i\gamma^j \qquad (7)
\end{aligned}$$

with the initialization condition $P^0 = P$. The term $\gamma^m \bmod T$ in Equation (7) is equivalent to $\sum_{j=0}^{m-1} t_j\gamma^j$ as proved by [7]. Moreover, the term $\sum_{j=0}^{m-2} p_j^i\gamma^{j+1}$ represents a polynomial of order less than m.

The recursive Equation (7) can be expressed in the bit-level form as:

$$p_j^{i+1} = p_{j-1}^i + p_{m-1}^i t_j,\ 0 \le j \le m-1 \qquad (8)$$

with $p_{-1}^i = 0$ for $0 \le i \le m-1$.

The recursive form of partial product S^{i+1}, $0 \le i \le m-1$ can be given from accumulating P^i terms as:

$$\begin{aligned}
S^{i+1} &= S^i + q_iP^i \\
\sum_{j=0}^{m-1} s_j^{i+1}\gamma^j &= \sum_{j=0}^{m-1} s_j^i\gamma^j + q_i\sum_{j=0}^{m-1} p_j^i\gamma^j \qquad (9)
\end{aligned}$$

with $S^0 = 0$ and S^m represents the final result S. Recursive Equation (9) can be expressed in the bit-level form as:

$$s_j^{i+1} = s_j^i + q_ip_j^i \qquad (10)$$

with $0 \le i \le m-1$, $0 \le j \le m-1$, and $s_j^0 = 0$ for $0 \le j \le m-1$.

Dependency Graph

Using reference [21], the dependence graph DG describing the modular multiplication can be obtained from Equations (8) and (10). The indices i and j in the two equations designate that the DG can be defined in a two-dimensional integer domain. Index i denotes the rows, and index j denotes the columns. Figure 2 presents the DG for the field size $m = 5$. The circled nodes compute the operations depicted by Equations (8) and (10). The vertical lines represent the partial product signal s_j^i, the multiplier signal p_j^i, and the irreducible polynomial coefficient t_j. The horizontal lines represent the broadcast multiplicand signal q_i. The diagonal lines represent the signal p_{j-1}^i. Signals p_j^{i+1} and s_j^{i+1} requires signals p_{j-1}^i and p_j^i to be computed. The last column nodes produce signal p_{m-1}^i, which is used inside the column nodes, and broadcasted to the remaining row nodes as indicated in Figure 2.

As we observe from Figure 2, the input signals s_j^0, p_j^0 are fed at the upper row of the DG, and the output signals s_j^m are produced from the lower row.

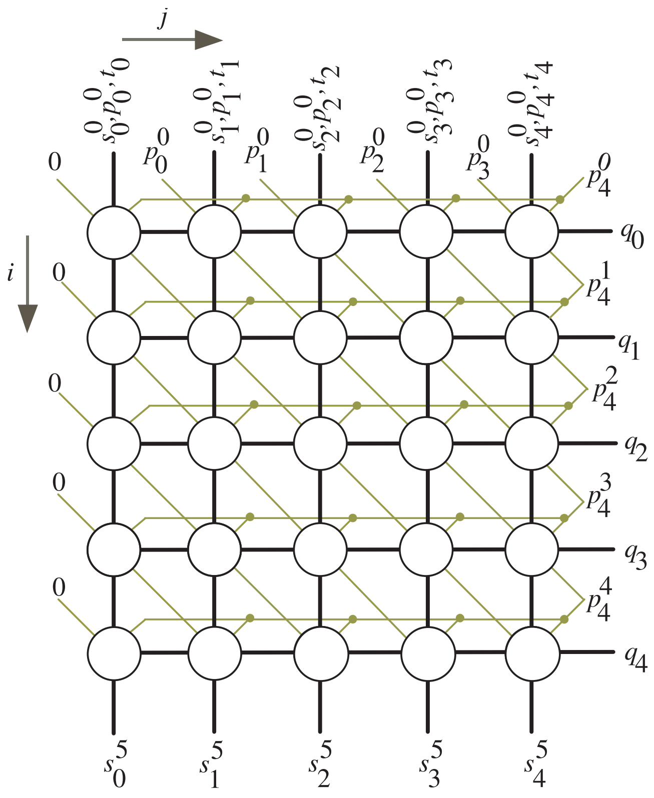

Figure 2. DG of the polynomial modular multiplication algorithm for the case $m = 5$.

3. Word-Serial Accelerator Structure Exploration

We will follow a formal and systematic methodology that we previously developed in [19,21–23] to map the recursive–iterative multiplier algorithm to a processor array and assign an execution schedule to each processing element (PE) in the resulting array.

3.1. Scheduling Function

Consider the two-dimensional dependence graph for the polynomial modular multiplication algorithm shown in Figure 2. We assume the processor array we would like to develop has l-bit digit or word size.

A valid nonlinear scheduling function assigns an execution time value to each node or point $\mathbf{P}$ in Figure 2 according to the nonlinear expression:

$$\Gamma(\mathbf{P}) \quad = \quad i\left\lceil \frac{m}{l} \right\rceil + \left\lfloor \frac{m-1-j}{l} \right\rfloor + 1 \tag{11}$$

$\Gamma(\mathbf{P})$ is the function that assigns a time instance to node $\mathbf{P}(i, j)$ in the DG.

Figure 3 shows the time index values after applying the nonlinear scheduling function in Equation (11) for the case when $m = 5$ and $l = 3$. The figure shows the DG points are being grouped horizontally in l-bit groups having the same execution time value. This ensures that all the bits in a single processor word are executed at the same time. An extra column of nodes is added at the left side of the DG to ensure that the number of columns in the DG is an integer multiple of l. For the general case, we need to add extra $\theta = l\lceil \frac{m}{l} \rceil - m$ columns, with zero inputs, at the left side of the DG. As we notice from Figure 3, the output of the multiplier will be available after $m\lceil \frac{m}{l} \rceil$ computation steps.

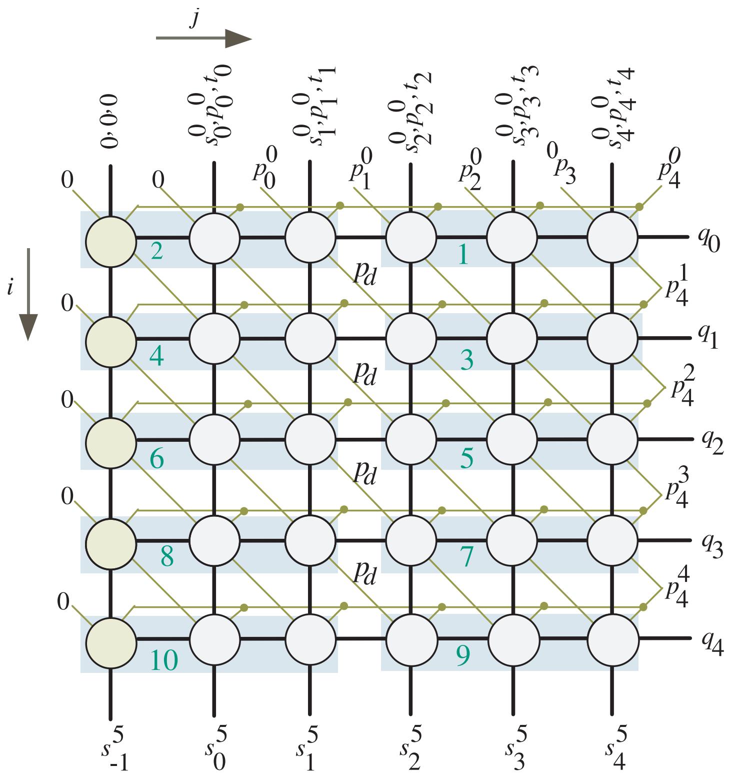

Figure 3. DG scheduling for the case $m = 5$ and $l = 3$.

An important feature of the proposed scheduling function is to provide the system designer with the ability to control the workload of the entire processor array system. For the nonlinear scheduling formula in Equation (11), we note that only one group of l bits is active at any given time instance. Therefore, the PE workload is equal in that case to the system workload. Of course, the system designer could choose another scheduling function to choose a different system workload.

3.2. Projection Function

The projection function approach discussed in [21] projects several nodes in the DG of Figure 3 to a single node. This operation is necessary since each l group of nodes in Figure 3 operates only once. Therefore, to reuse the processing elements, we map several groups into one PE. The system workload in Figure 3 implies that we need to map all the nodes of the DG into one PE only. We propose the following nonlinear projection function to map a node $\mathbf{P}(i,j)$ to a new node $\overline{\mathbf{P}}(x,y)$:

$$\overline{\mathbf{P}}(x,y) = \mathbf{P}_{serial}\ \mathbf{P}(i,j) \quad (12)$$

$$x = i \quad (13)$$

$$y = m - 1 - j \bmod l \quad (14)$$

$$\mathbf{P}_{serial} = [\ 1 \quad . \bmod l\ ] \quad (15)$$

where "·" is a place holder for the argument [21].

Figure 4 shows the resulting word-serial accelerator structure after applying the assumed projection function to Figure 3. The system consists of the following components:

1. A processor array block whose word size is l;
2. Three input registers T, P and PL;
3. One output register S;
4. Three shift-right registers *SHR-S*, *SHR-pd* and *SHR-P* (which is inside the processor array block);
5. Rotate-right register *ROR-T*;

6. Four three-input MUXes (two of them inside the processor array block) to select between the inputs and partial results of variables P and T.

Register P passes bit values starting from bit p^0_{m-1}, while register PL passes the bit values of the word variable starting from bit p^0_{m-2}.

The partial results stored in S and P are cycled through the shift registers *SHR-S* and *SHR-P*, respectively. The fixed words of T are rotated through the rotate-right register *ROR-T*.

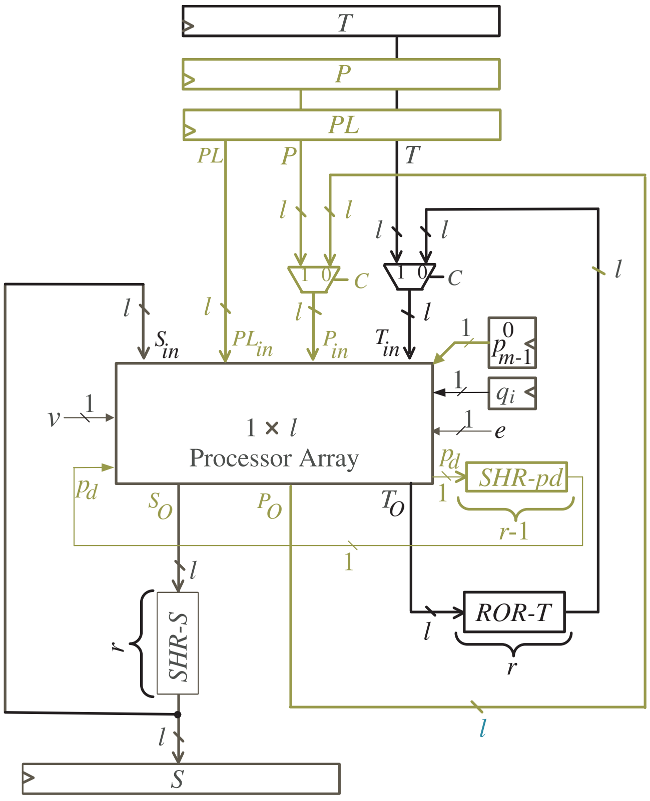

Figure 4. word-serial accelerator structure.

Observation of Figures 3 and 4 indicates that the rightmost bit p_d of register P is transferred diagonally to the following node after delayed by $r-1$ time steps, where $r = \lceil m/l \rceil$, while the remaining vertical and slanted word bits of P are transferred to the following bottom nodes after being blocked by r time steps. Therefore, bit P_d should be passed through the shift-right register, *SHR-pd*, that has depth size $r-1$ as shown in Figure 4. Shift-right registers *SHR-P* and *SHR-S*, and rotate-register *RoT-T* all have the same width and depth sizes of l and r, respectively.

Figure 5 displays the details of the processor array block for the case $l = 3$ bits. The processor array contains two types of PEs. Figures 6 and 7 depict the design details of the PEs. All the PEs are interconnected in a pipeline structure to perform computing at the same time. The yellow PE in Figure 3 has two more tri-state buffers managed by the control signal e. We will discuss below the role of these extra buffers.

The operation details of the explored multiplier accelerator can be summarized for generic filed size m and word size l as follows:

1. Control signal C, controlling the selection of all MUXes, activates ($C = 1$) during the the first $\lceil m/l \rceil$ clock cycles to feed the input words of operands T, P, and PL to all PEs of the processor array block. The words are fed starting with the most significant words. Moreover, the most significant bit p^0_{m-1} is passed to the last PE, PE_l, and broadcasted to the remaining PEs. At the first clock cycle, SHR-S is cleared to initialize the S variable with zero values.

2. Control signal C of all MUXes deactivates ($C = 0$) during the remaining clock cycles to feed the resulted intermediate words of P and fixed words of T to all PEs of the processor array block. These words are passed through shift-registers SHR-P, SHR-pd, and RoT-T, respectively. Moreover, the resulted intermediate words of S are fed to all PEs of the processor array block through the shift-register SHR-S.
3. Control signal e activates ($e = 1$) at the clock cycles $T = (i)\lceil \frac{m}{l} \rceil + 1$, $0 \le i \le m$, to enable the tri-state buffer Tr_1 shown in Figure 6 to horizontally feed the bits of p^i_{m-1}, $0 \le i \le m$, to the remaining PEs. Moreover, q_i input bits are broadcasted during the same clock cycles to all PEs in the processor array block. The control signal e deactivates ($e = 0$) during the remaining clock cycles to enable the tri-state buffer Tr_2, displayed in Figure 6, to feed the bits of p_d through the shift-register SHR-pd to the input of the processor array block as shown in Figure 4.
4. Control signal v, shown in Figure 5, deactivates ($v = 0$) at clock cycles $T = (i+1)\lceil \frac{m}{l} \rceil$, $0 \le i \le m$, to force zero bit values to the P words shown at the leftmost side of the DG, Figure 3. Control signal v activates ($v = 1$) at the remaining clock cycles to feed the P_d signal through the leftmost MUX of the processor array shown in Figure 5.
5. The resulting output words S are available at the output bus, through register S shown in Figure 4, during clock cycles $T \ge (m-1)\lceil \frac{m}{l} \rceil + 1$.

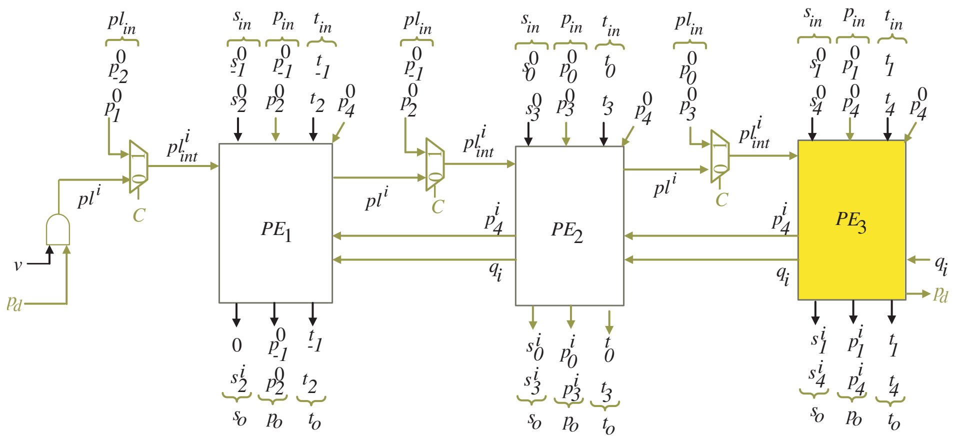

Figure 5. Details of the processor array block for word-size $l = 3$.

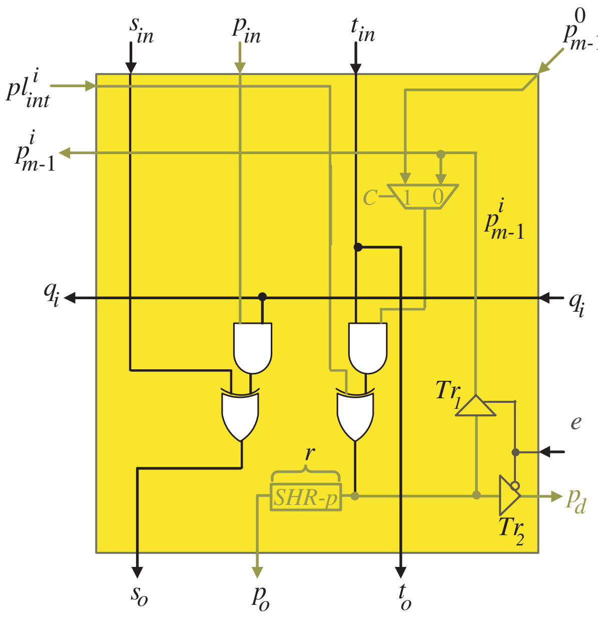

Figure 6. Yellow PE details.

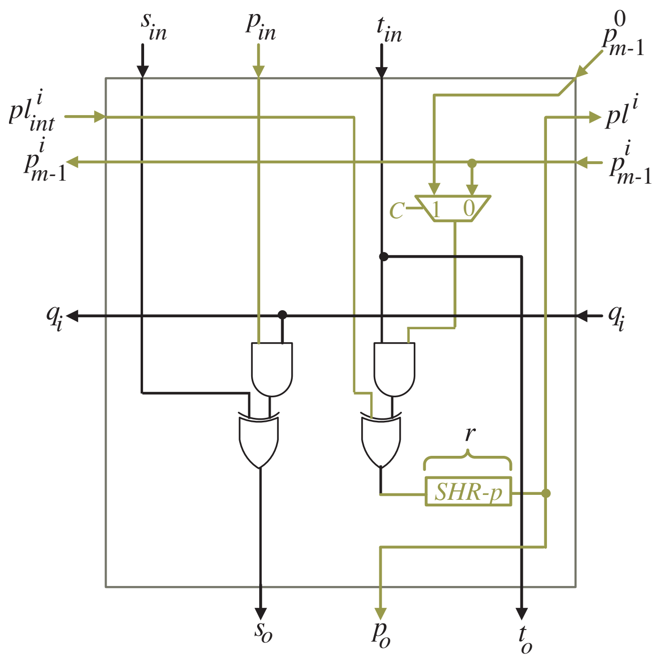

Figure 7. White PE details.

4. Complexities Analysis

The area, delay, and consumed energy complexities of the proposed multiplier are reported and compared to other efficient word-serial multipliers reported in [9–12]. Table 1 summarizes the area and delay complexities of recommended multiplier and the previously reported efficient word-serial ones. The total count of logic gates/components in the accelerator structure estimates the area complexity. The entire number of clock cycles needed to produce the product represents the latency (L) of the multiplier. The whole gate delays in the longest path of the logic circuit represent the critical path delay (CPD) of the multiplier structure. The product of latency and critical path delay (CPD) estimates the delay complexity. We can represent the delays of the 2-input AND, 2-input XOR, and 2-to-1 MUX by τ_A, τ_X, and τ_{MUX} symbols, respectively.

Table 1. Estimation of area and delay for the adopted word-serial multipliers.

Multiplier	Tri-State	AND	XOR	MUXes	Flip-Flops	Latency	CPD
Xie [10]	0	$2ml$	$2ml + 6m - 6\frac{m}{l} + 6$	0	$4ml + 4m + 2l$	$2\lceil m/l \rceil + 2\lceil \log_2 l \rceil$	$2D_X$
Pan [9]	0	$m\sqrt{m}$	$\sqrt{ml}(2 + l) + l$	0	R_1	$2\lceil \sqrt{m/l} \rceil$	η_1
Hua [11]	0	l^2	$l^2 + 4 - 5l + 1^{(1)}$	0	R_2	$6l\lceil m/l \rceil^2$	η_2
Chen [12]	0	$l^2 + l$	$l^2 + 2l$	$2l^{(2)}$	R_3	D_1	η_3
Proposed	2	$3l + 2$	$3l$	$2l$	$4l(\lceil m/l \rceil + 1) - 1$	$m\lceil m/l \rceil$	η_4

(1) The three-input logic XOR area is estimated as 1.5× the area of the two-input logic XOR. (2) The switches in Multiplier of [12] have the same area as the 2-to-1 MUX as it has the same number of transistors.

The following formulas describe the remaining notations in Table 1:

- $R_1 = 7m + m(\lceil \log m \rceil) + l + 3$
- $R_2 = 2l^2 + 2l(\lceil m/l \rceil) + 4l + 1$
- $R_3 = 2l^2 + 3l(\lceil m/l \rceil) + 2l$
- $D_1 = l + \lceil m/l \rceil^2 + \lceil m/l \rceil$
- $\eta_1 = \tau_A + (\lceil \log_2 l \rceil + 1)\tau_X$
- $\eta_2 = \tau_A + 2\tau_X$
- $\eta_3 = \tau_A + \tau_X$

- $\eta_4 = 2\tau_A + \tau_X + \tau_{MUX}$

Input and output flip-flops of each multiplier structure are added to the total estimated number of its flop-flops. As we notice from Table 1, the proposed multiplier structure has a significant reduction in the area compared to other multiplier structures due to having area complexity of order $\mathcal{O}(l)$.

To quantify the results obtained in Table 1, we modeled the proposed multiplier structure and the adopted ones using VHDL hardware language and synthesized them for the recommended field size $n = 409$ and embedded word sizes of $l = 8$, $l = 16$, and $l = 32$. The synthesis was performed using NanGate Open Cell Library (15nm, 0.8V) and Synopsys tools version 2005.09-SP2. The following describes the synthesis design parameters obtained in Table 2:

1. Area (A) results are obtained in terms of the two-input NAND gate and are represented in units of kilo-gates *kgates*.
2. Total computation time (T) is represented in nano-second *ns* time unit.
3. Consumed power (P) is obtained at a frequency of 1 KHz in units of milliwatt *mW*.
4. Consumed energy (E) is obtained as the product of P and T in units of femtojoule fJ
5. Area–time product (AT) is obtained as the product of A and T in units of kgates–nanosecond *Kgates.ns*

Table 2. Performance parameters of the adopted word-serial multipliers for $n = 409$ and different values of l.

Multiplier	l	A [*Kgates*]	T [*ns*]	P [*mW*]	E [*fJ*]	AT	%A	%AT	%P	%E
Xie [10]	8	92.9	18.3	225.6	4.1	1698.7	97.9	31.8	99.5	76.5
	16	147	9.7	375.5	3.6	1425.9	98.1	15.0	99.4	75
	32	195.1	5.5	477.4	2.6	1078.9	98.3	31.8	99.4	76.1
Pan [9]	8	130.5	9.9	252.9	2.5	1291.6	98.5	−27.3	99.6	61.2
	16	153.9	8.8	320.1	2.8	1354.6	98.2	10.5	99.3	67.7
	32	194.3	6.8	425.1	2.9	1317.6	98.3	44.2	99.3	78.1
Hua [11]	8	7.9	19,053.5	4.4	82.9	152,237.2	76.2	98.9	74.3	98.8
	16	10.4	9526.7	5.9	55.7	99,077.9	73.1	98.8	64.1	98.4
	32	19.9	4763.3	11.2	53.1	94,838.7	82.9	99.2	73.9	98.8
Chen [12]	8	10.2	659.4	5.1	3.4	6699.7	81.3	75.5	78.1	71.2
	16	13.5	203.0	8.4	1.7	2742.9	79.3	55.8	74.9	46.5
	32	26.6	86.8	15.9	1.4	2306.3	87.2	68.1	81.8	54.3
Proposed	8	1.9	865.6	1.1	0.97	1644.6	-	-	-	-
	16	2.8	432.8	2.1	0.91	1211.8	-	-	-	-
	32	3.4	216.4	2.9	0.6	735.8	-	-	-	-

Charts of Figures 8–11 compare the obtained results of area (A), area–time product (AT), consumed power (P), and consumed energy (E), respectively, of the proposed multiplier structure with the adopted ones.

Figure 8 depicts that the proposed multiplier structure saves a significant amount of area ranging from 76.2% to 98.5% at $l = 8$, 73.1% to 98.1% at $l = 16$, and 82.9% to 98.3% at $l = 32$ compared with the adopted word-serial multipliers. As we mentioned before, the saving in the area is due to the lower area complexity $\mathcal{O}(l)$ of the proposed design compared to the other designs. It is worth noting that the area of the proposed design has a slight difference at the different word sizes. This is due to the reverse relationship between the number of flip-flops and the word size l as indicated in Table 1. Therefore, as l increase, the number of flip-flops decrease and the number of basic logic components increase as it is directly proportional to the word-size l. The area complexity of the flip-flop is much higher than that of the other gates. As a result, the area reduction of flip-flops will have a significant impact on the overall area reduction of the proposed multiplier. Thus, the net result is a slight increase in the proposed design area as its word size increases. The proposed multiplier structure saves a significant amount of area ranging from 76.2% to 98.5% at $l = 8$, 73.1% to 98.1% at $l = 16$, and 82.9% to 98.3% at $l = 32$ compared with the adopted word-serial multipliers.

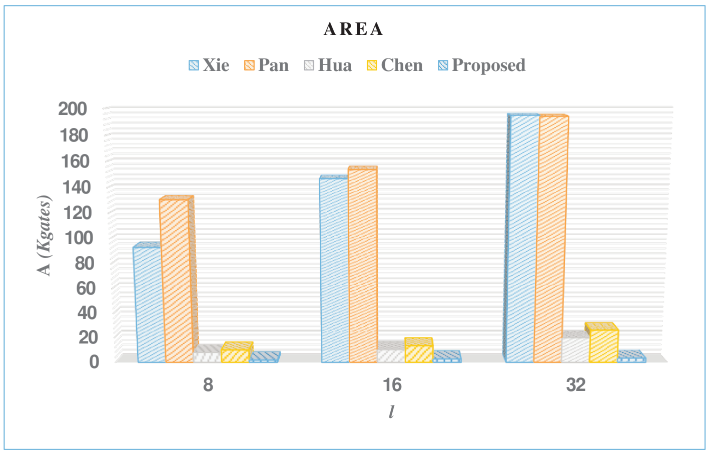

Figure 8. Area experimental results.

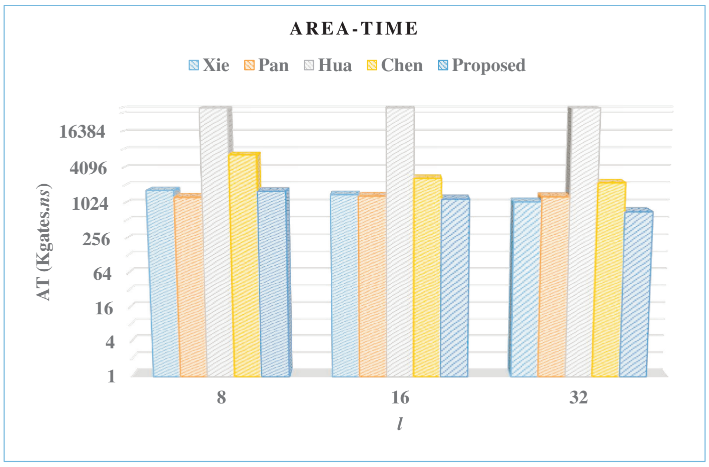

Figure 9. Area–time experimental results.

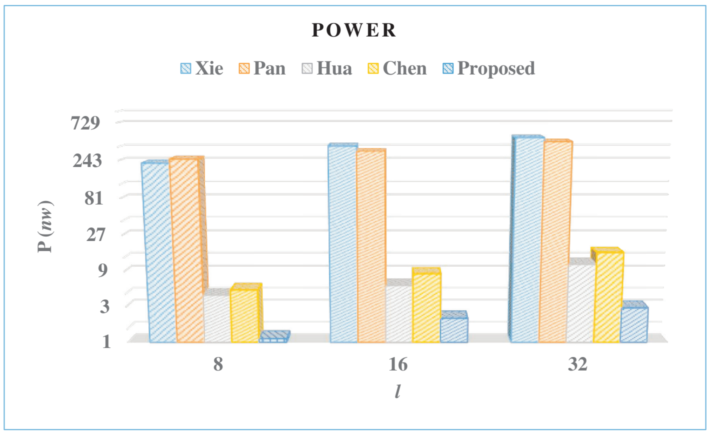

Figure 10. Consumed power experimental results.

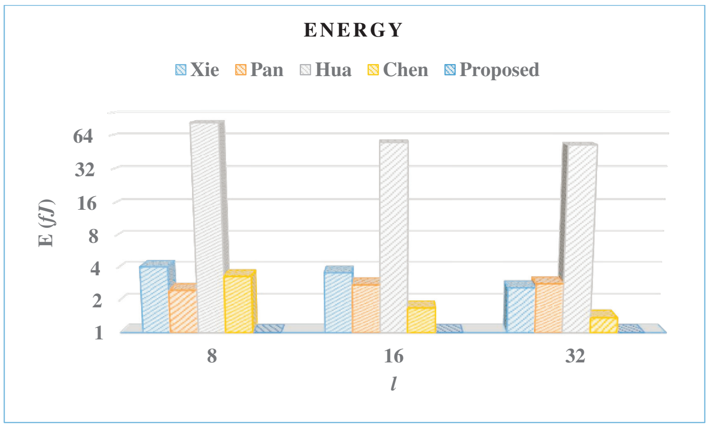

Figure 11. Consumed energy experimental results.

Figure 9 displays the obtained area–time (AT) results of the proposed design and the adopted word-serial ones. We can read the results based on the different word-sizes l as follows:

(i) At word-size $l = 8$, the multiplier that achieves the lowest AT is the multiplier of Pan [9]. It outperforms the proposed design by %27.8 at this word size. On the other hand, the proposed multiplier outperforms the other multipliers in AT at this word size, achieving a maximum reduction of 98.9% over the design of Hua [11].

(ii) At word-sizes $l = 16$ and $l = 32$, the proposed multiplier achieves the lowest AT than the other multiplier structures due to the significant reduction of its latency and computation time at these word sizes. As we notice from Table 1, the latency of the proposed multiplier is inversely proportional to the word size l. As a result, the latency significantly decreases as the word size l increases.

Figure 10 shows that proposed multiplier structure achieves a significant reduction in power consumption ranging from 74.3% to 99.6% at $l = 8$, 64.1% to 99.44% at $l = 16$, and 73.9% to 99.4% at $l = 32$ compared to the other multiplier designs. The reduction of power is attributed to the lower area complexity of the proposed design over the other designs. The reduction in area reduces the total amount of parasitic capacitances, resulting in a significant reduction in switching activities, which is one of the primary sources of consuming power.

Figure 11 shows that the proposed multiplier structure achieves a magnified reduction in energy ranging from 61.2% to 98.8% at $l = 8$, 67.7% to 98.3% at $l = 16$, and 76.1% to 98.8% at $l = 32$ compared to the adopted word-serial multipliers. The energy reduction is mainly attributed to the magnified reduction of the consumed power of the proposed design over the adopted ones.

From the previous analysis, we can conclude that the recommended word-serial multiplier structure outperforms the other competitor multiplier structures in terms of area and consumed energy for the different embedded word sizes. This indicates that the proposed multiplier is suitable for IoT devices in resource-constrained IoT applications.

5. Summary and Conclusions

This paper proposes a word-serial accelerator multiplier structure that performs multiplication in GF(2^m). The multiplier was extracted based on a systematic methodology that uses non-linear scheduling and projection functions to map the nodes of the algorithm dependency graph on to parallel processing elements. The main features of the proposed multiplier involve its flexibility in managing the accelerator workload and the required total computation time steps to produce the output results. The regularity and modularity of the extracted processor array block of the multiplier accelerator make it more suitable for implementation using ASIC technology. The experimental results confirm that the proposed multiplier outperforms the efficient word-serial ones previously reported on in the literature, in terms of area and consumed energy for various embedded word sizes, making it more suitable for embedded applications and other resource-constrained IoT applications. In the future, we will use the obtained multiplier structure as a building block for the ECC cryptographic processor to evaluate the overall reduction of the cryptographic processor in terms of area and consumed energy.

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

Funding: Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia, project number (IF-PSAU-2021/01/17867).

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 (IF-PSAU-2021/01/17867).

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

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
ASIC	Application Specific Integrated Circuit
ECC	elliptic curve cryptography
DG	dependency graph
VLSI	very large scale integrated circuit
RSA	Rivest, Shamir, and Adleman
CPD	critical path delay

References

1. Pourghebleh, B.; Hayyolalam, V.; Anvigh, A.A. Service discovery in the Internet of Things: Review of current trends and research challenges. *Wirel. Netw.* **2020**, *26*, 5371–5391. [CrossRef]
2. Anajemba, J.H.; Iwendi, C.; Mittal, M.; Yue, T. Improved advance encryption standard with a privacy database structure for IoT nodes. In Proceedings of the 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), Gwalior, India, 10–12 April 2020; pp. 201–206.
3. Mittal, M.; Vijayal, S. Detection of attacks in IoT based on ontology using SPARQL. In Proceedings of the 2017 7th International Conference on Communication Systems and Network Technologies (CSNT), Nagpur, India, 11–13 November 2017; pp. 206–211.
4. Anajemba, J.H.; Yue, T.; Iwendi, C.; Alenezi, M.; Mittal, M. Optimal cooperative offloading scheme for energy efficient multi-access edge computation. *IEEE Access* **2020**, *8*, 53931–53941. [CrossRef]
5. NIST. Post-Quantum Cryptography, Round 2 Submissions. Available online: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions (accessed on 9 December 2020).
6. Kim, K.W.; Jeon, J.C. Polynomial Basis Multiplier Using Cellular Systolic Architecture. *IETE J. Res.* **2014**, *60*, 194–199. [CrossRef]
7. Choi, S.; Lee, K. Efficient systolic modular multiplier/squarer for fast exponentiation over GF(2^m). *IEICE Electron. Express* **2015**, *12*, 20171195. [CrossRef]
8. 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]
9. 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. Sys.-I* **2013**, *60*, 3195–3204. [CrossRef]
10. 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]
11. 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 GF(2^m) Using Hankel Matrix and Karatsuba Algorithm. *IET Inf. Secur.* **2013**, *7*, 75–86.

12. 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]
13. Talapatra, S.; Rahaman, H.; Mathew, J. Low complexity digit serial systolic montgomery multipliers for special class of GF(2^m). *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2010**, *18*, 847–852. [CrossRef]
14. 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]
15. 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.
16. 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.
17. Bayat-Sarmadi, S.; Kermani, M.M.; Azarderakhsh, R.; Lee, C.Y. Dual-Basis Superserial Multipliers for Secure Applications and Lightweight Cryptographic Architectures. *IEEE Trans. Circ. Sys.-II* **2014**, *61*, 125–129. [CrossRef]
18. Gebali, F.; Ibrahim, A. Efficient Scalable Serial Multiplier Over GF(2^m) Based on Trinomial. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2015**, *23*, 2322–2326. [CrossRef]
19. 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]
20. Kim, K.W.; Lee, J.D. Efficient unified semi-systolic arrays for multiplication and squaring over GF(2^m). *IEICE Electron. Express* **2017**, *14*, 1–10. [CrossRef]
21. Gebali, F. *Algorithms and Parallel Computers*; John Wiley: New York, NY, USA, 2011.
22. Ibrahim, A.; Alsomani, T.; Gebali, F. Unified Systolic Array Architecture for Field Multiplication and Inversion Over GF(2^m). *Comput. Electr. Eng. J.-Elsevier* **2017**, *61*, 104–115. [CrossRef]
23. 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]