UVicSPACE: Research & Learning Repository

University of Victoria

Faculty of Engineering

Faculty Publications

Compact hardware accelerator for field multipliers suitable for use in ultra-low power IoT edge devices

Ibrahim, A. & Gebali, F.

2022

© 2022 Atef Ibrahim et al. This is an open access article distributed under the terms of the Creative Commons Attribution License. http://creativecommons.org/licenses/by-nc-nd/4.0/

This article was originally published at:
https://doi.org/10.1016/j.aej.2022.07.013

Citation for this paper:

Ibrahim, A. & Gebali, F. (2022). "Compact hardware accelerator for field multipliers suitable for use in ultra-low power IoT edge devices." *Alexandria Engineering Journal*, *61*(12), 13079-13087. https://doi.org/10.1016/j.aej.2022.07.013

Alexandria Engineering Journal (2022) 61, 13079–13087

HOSTED BY

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

Compact hardware accelerator for field multipliers suitable for use in ultra-low power IoT edge devices

Atef Ibrahim [a,b,*], Fayez Gebali [b]

[a] Computer Engineering Department, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University Alkharj, Saudi Arabia
[b] ECE Department, University of Victoria, Victoria, BC, Canada

Received 23 February 2022; revised 18 June 2022; accepted 4 July 2022

KEYWORDS

Finite field multiplication;
IoT security;
Cryptography;
IoT-edge devices;
Parallel processing;
Processor array

Abstract Adoption of IoT technology without considering its security implications may expose network systems to a variety of security breaches. In network systems, IoT edge devices are a major source of security risks. Implementing cryptographic algorithms on most IoT edge devices can be difficult due to their limited resources. As a result, compact implementations of these algorithms on these devices are required. Because the field multiplication operation is at the heart of most cryptographic algorithms, its implementation will have a significant impact on the entire cryptographic algorithm implementation. As a result, in this paper, we propose a small hardware accelerator for performing field multiplication on edge devices. The hardware accelerator is primarily composed of a processor array with a regular structure and local interconnection among its processing elements. The main advantage of the proposed hardware structure is the ability to manage its area, delay, and consumed energy by choosing the appropriate word size *l*. We implemented the proposed structure using ASIC technology and the obtained results attain average savings in the area of 95.9%. Also, we obtained significant average savings in energy of 63.2%. The acquired results reveal that the offered hardware accelerator is appropriate for usage in resource-constrained IoT edge devices.

© 2022 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

1.1. Work Motivation

The advent of the internet of things (IoT) technology allowed solving many issues in different domains of our life. Most IoT applications utilize smart-edge devices to sense, manipulate, and transmit the captured data [1,2]. In the past, most research efforts have been devoted to the development of IoT systems without considering their security. Currently, there are many solutions suggested to overcome the security issues in IoT networks [3–7]. However, most of the used approaches mainly concentrate on software solutions at the higher layers of the IoT framework. Also, they do not provide compact and low-power hardware security solutions at the edge devices. It is known that compact and energy-efficient hardware implementation of the cryptographic protocols on the edge devices provides more robust and effective security than software

* Corresponding author.
E-mail addresses: attif_ali2002@yahoo.com, aa.mohamed@psau.edu.sa, atef@ece.uvic.ca (A. Ibrahim), fayez@ece.uvic.ca (F. Gebali).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2022.07.013
1110-0168 © 2022 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Engineering, Alexandria University
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

implementations. Regular IoT edge devices have limited hardware resources with limited memory and processing power. Also, they are deployed in remote places and rely on the environment or small batteries for power [8]. Therefore, there is an urgent need to optimize their power consumption. The limited resources of IoT edge devices make the implementation of cryptographic algorithms a challenging task. The lack of implementation of cryptographic algorithms on these devices makes the IoT network vulnerable to several security breaches [9].

The security features of integrity, secrecy, authentication, non-repudiation, and availability need to be implied to secure the IoT network. Implementation of these security features primarily depends on cryptographic algorithms that are mainly based on the fundamental finite field arithmetic operations. The finite field multiplication operation is at the heart of these operations. Given the power and delay constraints of most IoT edge devices, Elliptic Curve Cryptography (ECC) became the encryption technique of choice. This is due to its high level of security and shorter key lengths when compared to common approaches such as RSA [10–12].

1.2. Related Work

Finite field multiplication is the most basic operation in ECC arithmetic. In both prime fields $GF(p)$ and binary extension fields $GF(2^n)$, there is a large amount of work on finite field multiplication. Much of the reported multipliers exhibited large space and delay complexity, making them impractical for IoT edge devices with limited resources [13–16]. Numerous publications suggested word-serial finite field multipliers to overcome these constraints. The systolic methods were described in [17–22] and non-systolic methods were described in [23–26]. Other publications combined the finite field multiplication and finite field squaring operations to save power and space [14,15,27]. However, because of their large area and power costs, the resulting structures were unsuitable for resource-constrained IoT devices.

1.3. Main Contributions

The majority of the finite field multiplier structures that have been disclosed are one-of-a-kind structures. Ad hoc procedures are used with no thought given to how the structure might be changed to improve system performance parameters of latency, throughput, power, and area. Based on the algebraic technique primarily suggested by the second author [28], the authors of the article developed a systematic methodology for implementing the finite field multiplication algorithm. To construct the finite field multiplier structures, the systematic methodology employed linear mappings. On the other hand, linear mappings, have restricted capabilities in terms of both the number of parallel processing elements (PE) and the timing approaches that can be implemented.

This article recommends using nonlinear procedures to map the iterative field multiplication algorithm onto parallel PEs and to obtain more accommodating timing methodologies. The purpose of this paper is to create a word-serial processor accelerator for finite field multiplication over GF(2^n) that is based on Irreducible All-One Polynomials (AOP) [29]. The resulting architecture enables the developer to handle both the PE workload and the algorithm delay. In terms of area and consumed energy, the empirical results show that the proposed multiplier outperforms the effective word-serial ones originally discussed in the literature for various embedded word sizes. It achieves significant savings in the area and energy by factors up to 95.9% and 63.2%, respectively.

These design aspects make the provided architecture quite appropriate for resource-constrained IoT edge devices.

1.4. Work Organization

The arrangement of the article should be as follows. Section 2 shortly explains the mathematical background of the assumed finite field multiplication algorithm and its representation in the bit-level form. Also, it shows the details of the algorithm dependency graph. Section 3 displays the used methodology to explore the word-serial accelerator of the finite-field multiplier and provides the details of its logic structure. Section 4 exhibits the acquired implementation results. Section 5 concludes the recommended work.

2. Mathematical Background to Develop the Finite Field Multiplication Algorithm

Assume that the finite field over $GF(2^n)$ is specified by the irreducible polynomial $R(w)$ of degree n. The polynomial representation of $R(w)$ is as follows:

$$R(w) = 1 + r_1 w^1 + \cdots + r_i w^i + \cdots + r_{n-1} w^{n-1} + w^n \tag{1}$$

with $r_i \in GF(2)$. Assume α is the root of the irreducible polynomial $R(w)$. As a result, the field elements can be represented by the set of polynomial basis $\{1, \alpha, \alpha^2, \alpha^3, \cdots, \alpha^{n-1}\}$.

Presume that E and H are any two field elements in $GF(2^n)$. They can be written in the polynomial form of degree $n-1$ as follows:

$$E = e_0 + e_1\alpha^1 + \cdots + e_i\alpha^i + \cdots + +e_{n-1}\alpha^{n-1} \tag{2}$$

$$H = h_0 + h_1\alpha^1 + \cdots + h_i\alpha^i + \cdots + +h_{n-1}\alpha^{n-1} \tag{3}$$

where $e_i, h_i \in GF(2)$.

The following formula can be used to multiply E and H over $GF(2^n)$.

$$D = E \cdot H \bmod R(w) \tag{4}$$

We could extend Eq. (4) to have a multiplication recurrence relation as follows:

$$D = h_0 \cdot E + \left[\sum_{i=1}^{n-1} h_i \cdot \alpha^{i-1} \cdot K\right] \bmod R(w) \tag{5}$$

Where $K = \alpha E$ is a degree n polynomial that could be expressed as:

$$K = \sum_{i=0}^{n} k_i \cdot \alpha^i \tag{6}$$

with $k_0 = 0$ and $k_i = e_{i-1}$ for $i = 1, 2, \cdots, n$.

By extending the polynomial of (6) and multiplying by α, we can get

$$\alpha K = k_0\alpha + k_1\alpha^2 + \cdots + k_{n-1}\alpha^n + k_n\alpha^{n+1} \tag{7}$$

Because α is a root of $R(w)$, $R(\alpha)$ equals 0. As a result of Eq. (1), we can get

$$\alpha^n = 1 + r_1\alpha + r_2\alpha^2 + \cdots + r_{n-1}\alpha^{n-1} \tag{8}$$

Since the polynomial $R(w)$ is an AOP, Eq. (8) can be rewritten as:

$$\alpha^n = 1 + \alpha + \alpha^2 + \cdots + \alpha^{n-1} \tag{9}$$

We can get the following result by multiplying both sides of Eq. (9) by α:

$$\alpha^{n+1} = 1 \tag{10}$$

We could reduce αK to a polynomial (K^1) of degree n by substituting from (10) in (7) as follows:

$$K^1 = k_n + k_0\alpha + k_1\alpha^2 + \cdots + k_{n-1}\alpha^n \tag{11}$$

The cyclic-shift-left of polynomial K generates the partially-reduced polynomial K^1 of polynomial αK, as shown in Eq. (11). The partially-reduced polynomial K^2 of polynomial $\alpha^2 K$ is generated by cyclic-shift-left of polynomial K^1. In general, the partially-reduced polynomial K^i of polynomial $\alpha^i K$ is formed by cyclic-shift-left of polynomial K^{i-1}. This cyclic-shift-left procedure could be mathematically represented as:

$$K^i = CSL\left(K^{i-1}\right), \quad 0 \leqslant i \leqslant n-1 \tag{12}$$

where $K^{-1} = (0\&E)$ and CSL identify the cyclic-shift-left operation. We could use Eq. (12) to formulate Eq. (13) as:

$$D = h_0 \cdot E + \left[\sum_{i=1}^{n-1} h_i \cdot K^{i-1}\right] \bmod R(w) \tag{13}$$

with $K^0 = K = \alpha E$.

Alternatively, we could write Eq. (13) as:

$$D = V \bmod R(w) \tag{14}$$

where V is the sum of degree n polynomials that could be defined as:

$$V = \sum_{i=0}^{n-1} h_i \cdot K^{i-1} \tag{15}$$

with $K^{-1} = (0\&E)$.

The polynomial in Eq. (15) can be written in the following form:

$$V = v_0 + v_1\alpha^1 + v_2\alpha^2 + \cdots + v_{n-1}\alpha^{n-1} + v_n\alpha^n \tag{16}$$

We can get the reduced form of polynomial $V \bmod R(w)$ (polynomial of degree $n-1$) by replacing α^n in Eq. (16) with the expansion given in Eq. (9).

$$\begin{aligned} D = V \bmod R(w) = (v_0 \oplus v_n) + (v_1 \oplus v_n)\alpha^1 \\ +(v_2 \oplus v_n)\alpha^2 + \cdots + (v_{n-1} \oplus v_n)\alpha^{n-1} \end{aligned} \tag{17}$$

Assuming j is the bit position in a binary sequence representing any polynomial, we can express Eqs. (12) and (15) in bit-level form, as illustrated in Eqs. (18) and (19), respectively:

$$\begin{aligned} k_{j+1}^i &= k_j^{i-1} \\ k_0^i &= k_{n+1}^i \end{aligned} \tag{18}$$

$$v_j^i = v_j^{i-1} + h_i \cdot k_j^{i-1} \tag{19}$$

with $k_n^{-1} = 0, v_j^{-1} = 0, 0 \leqslant i \leqslant n-1$, and $0 \leqslant j \leqslant n$.

In addition, the reduced form of the product polynomial D, provided in Eq. (17), can be expressed in the bit-level form as:

$$d_j = v_j^{n-1} + v_n^{n-1} \tag{20}$$

with $0 \leqslant j \leqslant n-1$.

2.1. Exploration of the Dependency Graph

The iterative phase of the finite-field multiplication algorithm is described by the two iterative Eqs. (18) and (19). The iterations are defined by the two indices i and j. It is feasible to create a dependence graph (DG) in the two-dimensional integer domain $\mathbb{D}$ using the approach described in reference [28]. The DG for the situation $n = 5$ is shown in Fig. 1. The operations indicated by Eqs. (18) and (19) are represented by the nodes of the DG. Signals of v_j^i are depicted by vertical lines, according to the construction criteria of reference [28]. The horizontal lines indicate the signals h_i. The diagonal lines are used to represent the signals k_j^i. k_{n+1}^i signal is produced from the nodes in the final column and allocated to the nodes in the first column. The resultant signals $v_j^{n-1}, 0 \leqslant j \leqslant n-1$, from the bottom row are added, using XOR gates, with the most significant signal v_n^{n-1} to form the final product bits $d_j, 0 \leqslant j \leqslant n-1$ as described in Eq. (20). In the DG, the algorithm inputs $v_j^{-1}, k_j^{-1} = e_j$ are depicted as vertical and diagonal inputs to the top row nodes. The reduced product output $d_j, 0 \leqslant j \leqslant n-1$, on the other hand, is obtained by combining the vertical outputs of the bottom nodes with the output of the most right bottom node as displayed in Fig. 1.

3. Developing the Word-Serial Multiplier Accelerator

We will use a formalized technique defined earlier in [26,28,30–35] to transfer the recursive-iterative algorithm to a processor

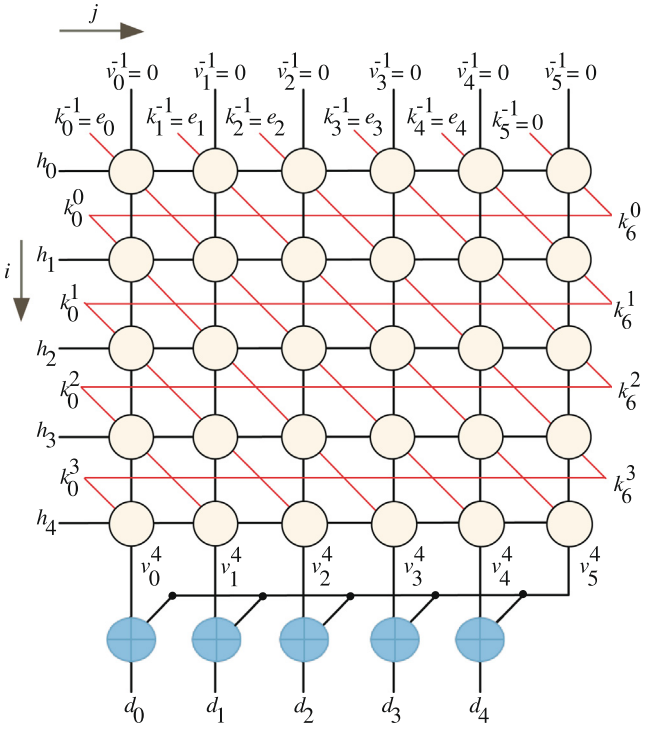

Fig. 1 DG of the adopted finite-field multiplication algorithm for $n = 5$.

array and assign an execution schedule to each processing element (PE) in the resulting array.

3.1. Scheduling Function

Suppose that the processor array we are working on has l-bit word size. By Considering the two-dimensional dependency graph in Fig. 1, we can derive the following non-linear scheduling function to allocate a proper execution time value to each node or point $\mathbf{P}$ in Fig. 1.

$$\Gamma(\mathbf{P}) = i\left\lceil \frac{n}{l} \right\rceil + \left\lfloor \frac{j}{l} \right\rfloor \tag{21}$$

The function $\Gamma(\mathbf{P})$ allocates a time instance to node $\mathbf{P}(i,j)$ in the DG. The time index values after applying the non-linear scheduling function of Eq. (21), for the case when $n = 5$ and $l = 3$, are shown in Fig. 2. As shown in the figure, the DG points are organized horizontally in l-bit groups with the same execution time value. This guarantees that every bit in a single processor word gets executed at the same time. If the number of columns is not a multiple of l, extra $\theta = l\lceil \frac{n}{l} \rceil - n$ columns with zero inputs should be added to the right side of the DG. The output of the multiplier will be accessible after $(n+1)\lceil \frac{n+1}{l} \rceil$ processing steps.

One of the most essential features of the proposed scheduling function is that it allows the system designer to manage the workload of the entire processor array system. Just one set of l bits is active at any particular time instance in the non-linear scheduling formula of Eq. (21). As a result, the PE workload is equivalent to the system workload.

3.2. Projection Function

Numerous nodes in the DG of Fig. 2 are projected to a particular node using the projection function technique explained in [28]. This is due to the fact that each group of nodes in Fig. 2 only operates once. As a result, we can map numerous groups into one PE in order to reuse the processing elements. To map a node $\mathbf{P}(i,j)$ to a new node $\overline{\mathbf{P}}(x,y)$, we recommend the following non-linear projection function:

$$\overline{\mathbf{P}}(x,y) = \mathbf{P}_{serial}\ \mathbf{P}(i,j) \tag{22}$$

$$x = i \tag{23}$$

$$y = j \bmod l \tag{24}$$

$$\mathbf{P}_{serial} = [1 \quad .\bmod l] \tag{25}$$

where "·" is a place holder for the argument as described in [28].

After applying the expected projection function to Fig. 2, the resulting word-serial accelerator structure is shown in Fig. 3. The following components make up the system:

1. Processor array block with l bit word size
2. One input register K of l bit size
3. One output register D of l bit size
4. One input D flip-flop to serialize the input h_i.

There is no need to introduce an input register for supplying input V because it contains zero values. We'll talk about how to set this input variable to zero values later. Due to the difference in feeding time to the first PE, the signal k_j^i should be separated into two signals, k_{n+1}^i and k_d^i, in the last PE of the processor array. As shown in Fig. 2, the k_d^i signal is passed diagonally to the next node after $r+1$ time steps, where $r = \lceil n/l \rceil$, whereas the k_{n+1}^i signal is fed to the next node after only one time step delay.

The elements of the processor array block for the scenario $l = 3$ bits are shown in Fig. 4. There are three different types of PEs in the processor array. The design features of the PEs are depicted in Figs. 5–7. The regular PEs, shown in Fig. 6, are located in the central part of the processor array and feature fewer logic components. Each PE has two shift registers, SR-K and SH-V. Before transmitting signal k_{j+1}^i to the next PE, SR-K pauses it for r time steps. Also, before returning v_j^i to

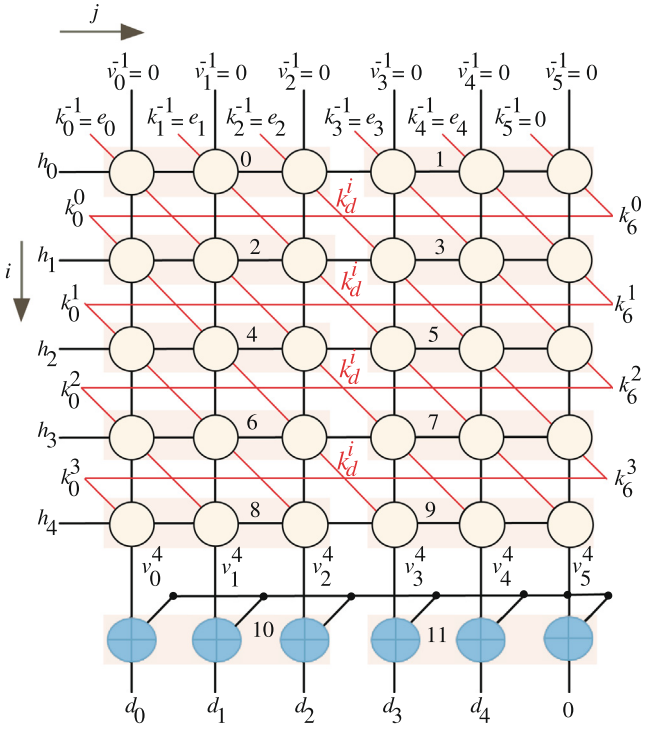

Fig. 2 DG scheduling for the case $n = 5$ and $l = 3$..

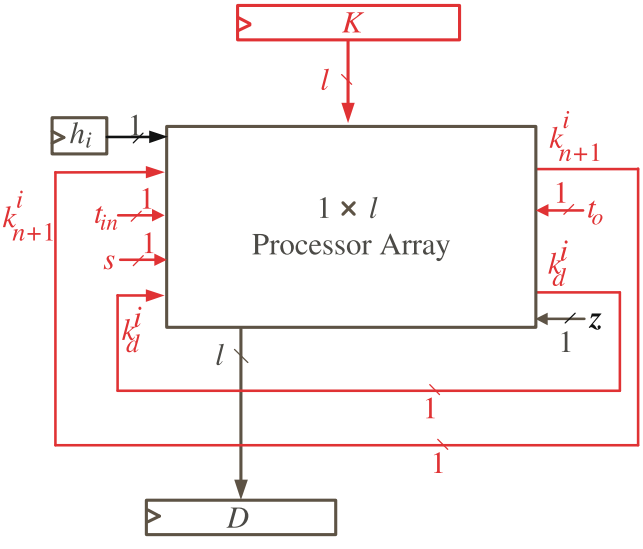

Fig. 3 word-serial accelerator structure.

the same PE, SH-V stops it for r time steps. The initial zero values of signal v_j^{i-1} are set by resetting the SH-V before the process begins. The control signals z and t_o manage three more tri-state buffers in the last PE (blue PE). The function of these buffers will be discussed further down. The last PE is also equipped with 2 extra D flip-flops. One is used to delay the k_{n+1}^i signal by a one-time step, while the other is used to extend the depth of the SR-K to have $r+1$ flip-flops. As illustrated in the node timing diagram, Fig. 2, increasing the depth of SR-K in the final PE causes it to transmit back the k_d^i signal to the first PE after $r+1$ time steps. The first PE (Green PE) only contains two additional tri-state buffers to choose between k_{n+1}^i and k_d^i fed back signals resulting from the last PE (Blue PE).

For generic filed size n and word size l, the operation features of the studied multiplier accelerator can be stated as follows:

1. During the first $\lceil n/l \rceil$ clock cycles, control signal s that control all MUXes should be set to zero ($s=0$) to pass the initial words of variable K to all PEs of the processor array block. The words are fed starting with the least significant words. To initialize the V variable with zero values, the SH-V shift register should be cleared on the first clock cycle.
2. During the remaining clock cycles, the control signal s of all MUXes activates ($s=1$) to feed the resultant intermediate bits of K to all PEs of the processor array block. These bits are piped between the PEs by passing through SH-K shift-registers. Due to the difference in its feeding time to the first PE, the intermediate signal k is separated into two signals in the last PE. The resultant intermediate words of V are also fed back into each PE of the processor array block through the shift-register SH-V during these clock cycles.
3. Input bits of h_i are propagated during the clock cycles $T=i\lceil \frac{n}{l} \rceil, 0 \leqslant i \leqslant n-1$, to all PEs in the processor array block.
4. Control signal t_o activates ($t_o=1$) at the clock cycles $T=(i+1)\lceil \frac{n}{l} \rceil - 1, 0 \leqslant i \leqslant n-1$, to enable the Tri-Sate buffer Tr_3, shown in Fig. 7, to feed back the bits of $k_{n+1}^i, 0 \leqslant i \leqslant n-1$, to the first PE after delayed by one time

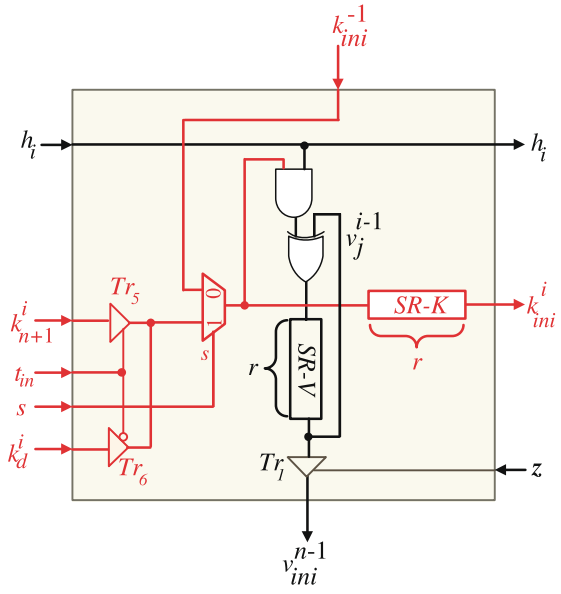

Fig. 5 First PE details.

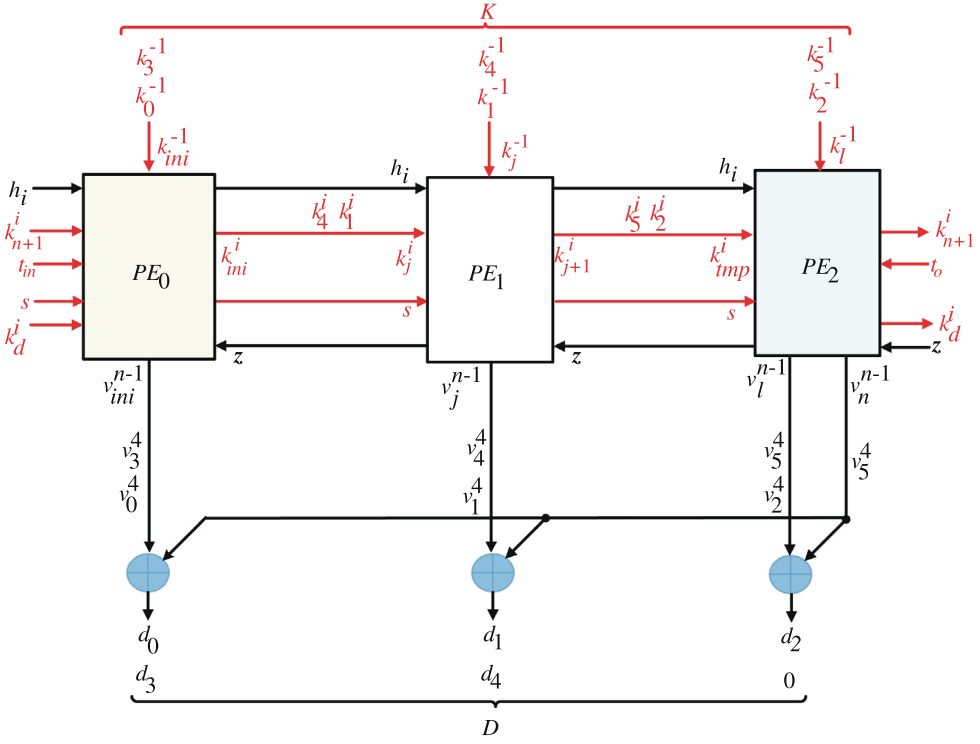

Fig. 4 The processor array block details for word-size $l=3$.

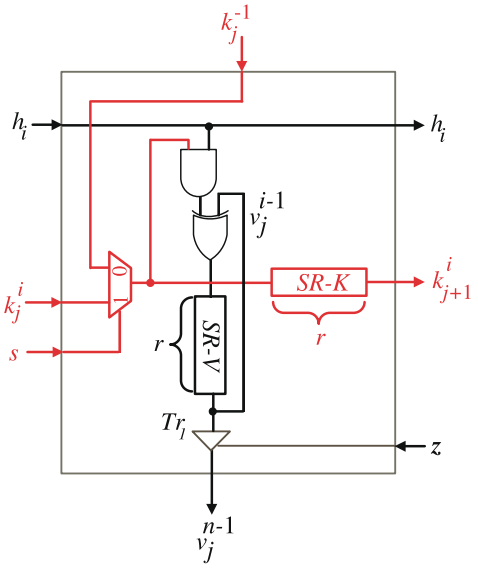

Fig. 6 Intermediate PE details.

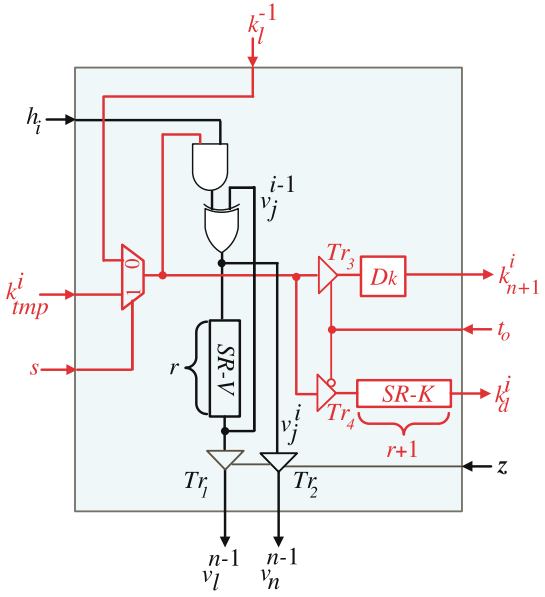

Fig. 7 Last PE details.

step. The control signal t_o deactivates ($t_o = 0$) during the remaining clock cycles to enable the Tri-Sate buffer Tr_4, displayed in Fig. 7, to feedback the bits of k_d^i to the first PE after delayed by $r + 1$ time steps by the shift-register SH-K as shown in Figs. 3 and 4.

5. Control signal t_{in} activates ($t_{in} = 1$) at the clock cycles $T = (i + 1)\lceil \frac{n}{l} \rceil, 0 \leqslant i \leqslant n - 2$, to enable the Tri-Sate buffer Tr_5, shown in Fig. 5, to load the bits of $k_{n+1}^i, 0 \leqslant i \leqslant n - 1$, to the first PE. The control signal t_{in} deactivates ($t_{in} = 0$) during the remaining clock cycles to enable the Tri-Sate buffer Tr_6, displayed in Fig. 5, to load the bits of k_d^i to the first PE.
6. Control signal z, shown in Fig. 4, activates ($z = 1$) starting from clock cycle $T = (n)\lceil \frac{n}{l} \rceil$, to pass the bits of variable V through tri-state buffers Tr_1 to be XORed with most significant bit v_n^{n-1} resulted from the last PE. As we notice from Fig. 7, it has extra tri-state buffer (Tr_2) to pass the resulted v_n^{n-1} bit at clock cycle $T = (n)\lceil \frac{n}{l} \rceil$ before loaded to the shift-register SH-V.
7. The resulted output words D are available at the output bus, through register D shown in Fig. 3, during clock cycles $T \geqslant n\lceil \frac{n}{l} \rceil$.

4. Complexities Analysis

In this section, the area, delay, and consumed energy complexities of the recommended finite field multiplier are described and compared to existing efficient word-serial multipliers presented in [19,36–38]. The area and delay complexities of the recommended multiplier and the earlier efficient word-serial multipliers are summarized in Table 1. The area complexity is evaluated using the total number of logic gates/components in the accelerator structure. The multiplier's latency (L) is the total number of clock cycles required to produce the product. The critical path delay (CPD) of the multiplier structure is the sum of all gate delays in the logic circuit's longest path. The delay complexity is quantified in terms of latency and critical path delay (CPD). The delays of the Tri-sate buffer, 2-input AND, 2-input XOR, and 2-to-1 MUX are represented by the symbols τ_T, τ_A, τ_X, and τ_{MUX}, respectively.

The further mathematical notation in Table 1 are detailed as follows:

1. $R_1 = 7n + n(\lceil \log n \rceil) + l + 3$
2. $R_2 = 2l^2 + 2l(\lceil n/l \rceil) + 4l + 1$
3. $R_3 = 2l^2 + 3l(\lceil n/l \rceil) + 2l$
4. $D_1 = l + \lceil n/l \rceil^2 + \lceil n/l \rceil$
5. $\eta_1 = \tau_A + (\lceil \log_2 l \rceil + 1)\tau_X$
6. $\eta_2 = \tau_A + 2\tau_X$

Table 1 Area and delay estimate for the chosen word-serial multipliers.

Multiplier	Tri-State	AND	XOR	MUXes	Flip-Flops	Latency	CPD
Xie [36]	0	$2nl$	$2nl + 6n - 6\frac{n}{l} + 6$	0	$4nl + 4n + 2l$	$2\lceil n/l \rceil + 2\lceil \log_2 l \rceil$	$2D_X$
Pan [19]	0	$n\sqrt{n}$	$\sqrt{nl}(2 + l) + l$	0	R_1	$2\lceil \sqrt{n/l} \rceil$	η_1
Hua [37]	0	l^2	$l^2 + 4 - 5l + 1^{(1)}$	0	R_2	$6l\lceil n/l \rceil^2$	η_2
Chen [38]	0	$l^2 + l$	$l^2 + 2l$	$2l^{(2)}$	R_3	D_1	η_3
Proposed	$l + 4$	l	$2l$	l	$2l + (l + 1)r + 3$	$(n + 1)\lceil (n + 1)/l \rceil$	η_4

(1) $r = \lceil n/l \rceil$.
(2) The 3-input logic XOR area is estimated as 1.5× the area of the 2-input logic XOR.
(2) The switches in Multiplier of [38] have the same area as the 2-to-1 MUX as it has the same number of transistors.

Table 2 Performance parameters of the chosen word-serial multipliers for $n = 508$ and various l values.

Multiplier	l	Latency	Area (A) [*Kgates*]	CPD [*ps*]	Time (T) [*ns*]	power (P) [*mW*]	Energy (E) [*fJ*]	AT	%A	%AT	%P	%E
Xie [36]	8	386	110.7	67.1	25.9	268.5	7	2866.4	99.2	10.8	99.6	55.7
	16	205	174.9	67.1	13.7	447.4	6.1	2396.5	99.4	34.8	99.7	65.6
	32	117	232.2	67.1	7.8	568.1	4.4	1810.9	99.3	33.3	99.7	70.5
Pan [19]	8	58	115.9	245.5	14	301.1	4.2	1622.7	99.2	−57.6	99.6	26.2
	16	43	147.6	290.8	12.5	380.9	4.8	1844.5	99.3	15.3	99.6	56.3
	32	29	195.5	336.2	9.6	505.9	4.9	1876.9	99.1	35.7	99.6	73.5
Hua [37]	8	308905	9.5	87.3	26981.6	5.2	141.3	256864.8	90.5	99.0	78.9	97.8
	16	154453	12.4	87.3	13490.8	7.0	94.7	166962.1	91.1	99.1	78.6	97.8
	32	77227	23.8	87.3	6745.4	13.2	89.1	160540.5	92.9	99.25	85.6	98.5
Chen [38]	8	14216	12.1	65.7	933.8	6.1	5.7	11334.5	92.6	77.4	81.9	45.6
	16	4377	16.1	65.7	287.5	9.9	2.9	4618.7	93.2	66.2	84.9	27.6
	32	1871	31.7	65.7	122.9	19.0	2.3	3890.3	94.6	68.9	90.0	43.5
Proposed	8	32576	0.9	87.4	2841.6	1.1	3.1	2557.5	-	-	-	-
	16	16288	1.1	87.4	1420.9	1.5	2.1	1562.9	-	-	-	-
	32	8144	1.7	87.4	710.5	1.9	1.3	1207.8	-	-	-	-

7. $\eta_3 = \tau_A + \tau_X$
8. $\eta_4 = \tau_T + \tau_A + \tau_X + \tau_{MUX}$.

For a fair comparison, the input and output flip-flops of each multiplier structure are added to the total expected number of flop-flops. By examining Table 1, we can notice that the suggested multiplier structure has a significantly smaller area than the prior multiplier structures due to its area complexity of order $\mathcal{O}(l)$.

To confirm the qualitative results obtained in Table 1, we used the VHDL hardware description language to describe the suggested and existing multiplier structures. We synthesized them for the field size $n = 508$ and embedded word sizes of $l = 8, l = 16$, and $l = 32$. The synthesis was carried out with Synopsys tools version 2005.09-SP2 and NanGate Open Cell Library (15 nm, 0.8 V).

The synthesis design parameters obtained in Table 2 are as follows:

- The area (A) findings are calculated using the 2-input NAND gate and are expressed in kilo-gates (*kgates*).
- The Critical Path Delay is measured in pico-second (*ps*) time unit.
- The total computation time (T) is measured in nano-second (*ns*) time unit.
- The consumed power (P) is measured in mili-watt (*mW*) units at a frequency of 1 kHz.
- The consumed energy (E) is calculated by multiplying P and T and obtained results are represented in femto-joule (*fJ*) units.
- The Area-Time design metric is calculated by multiplying A and T and obtained results are represented in (*kgates.ns*).

The savings in the area (%A), Area-Time (%AT), power (%P), and energy (%E) of the proposed design over the exiting designs are given in the last columns of Table 2. The charts given in Figs. 8–11 compare the achieved results of A, T, P, and E of the suggested multiplier structure with those of the adopted multiplier structure.

Fig. 8 depicts the area results for the different embedded word sizes on a bar chart for the proposed and compared word-serial designs. The proposed design is represented by a dark blue color. The vertical axis represents the area on a logarithmic scale, while the horizontal axis represents the embedded word sizes. The green line that crosses the bars represents the average savings in the area of the proposed design over the compared ones. By examining the chart, we can conclude that the suggested multiplier structure saves a substantial amount of area when compared to the adopted word-serial multipliers by an average value of 95.9%. As previously stated, the proposed

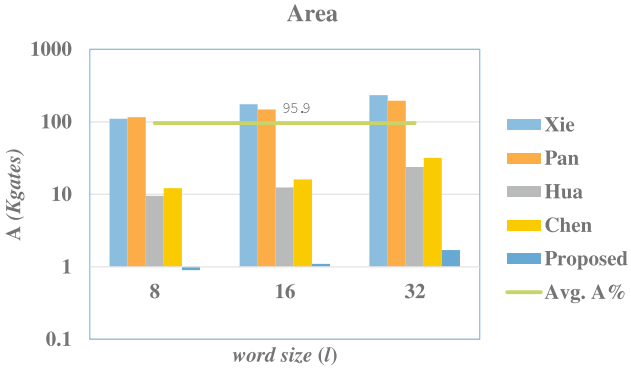

Fig. 8 Experimental findings of the area.

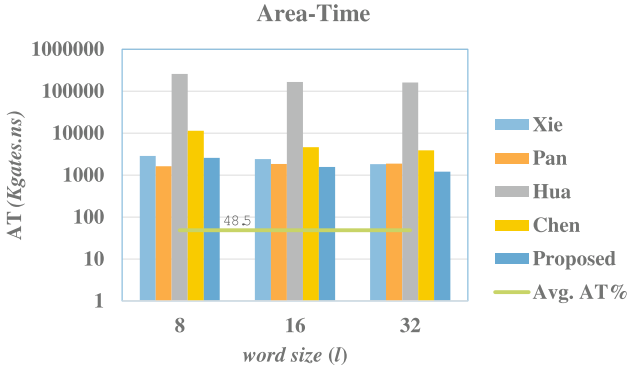

Fig. 9 Experimental findings of the Area-Time.

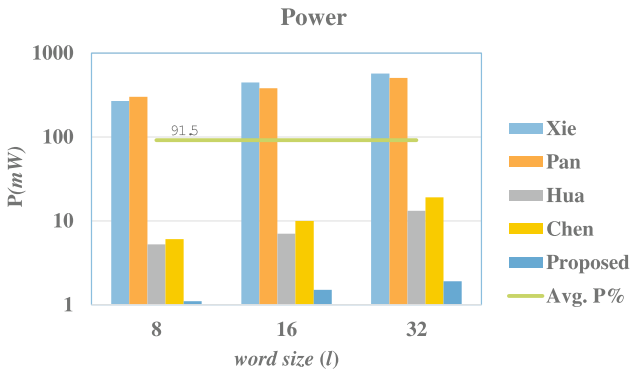

Fig. 10 Experimental findings of of the consumed power.

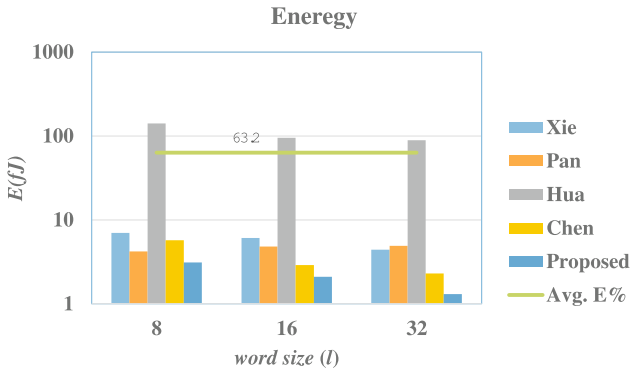

Fig. 11 Experimental findings of the consumed energy.

design saves space due to its lower area complexity $\mathcal{O}(l)$ when compared to the other designs. It's worth noting that the recommended design's area differs slightly depending on the word size. This is owing to the fact that the number of Flip-Flops and the word size l have the opposite relationship, as shown in Table 1. As a result, as l increases, the number of Flip-Flops decreases while the number of basic logic components increases because it is directly proportional to word-size l. As the area of the Flip-Flop is larger than the areas of the other logic components, reducing the Flip-Flop will have a significant impact on the total area of the design. As a result, the savings in the Flip-Flop area will partially offset the increase in logic gate components, accounting for the slight increase in the suggested design area as the word size l increases.

Fig. 9 depicts the proposed design's Area-Time (AT) complexity results and the word-serial ones for various embedded word sizes. A bar chart with two coordinates is depicted in the figure. The vertical coordinate represents the obtained AT values on a logarithmic scale, while the embedded word sizes are represented by the horizontal coordinate. The green horizontal line that runs through the bars represents the proposed multiplier's average savings in AT over the compared ones. We can see from the chart that the multiplier of Pan [19] is the one that gives the smallest AT at word size $l = 8$. For the remaining word sizes, the proposed design outperforms Pan's design in AT due to the significant reduction in its area and time at these word sizes when compared to the design of Pan. Also, the proposed design outperforms all the other designs in AT at all word sizes and achieves average savings in AT by 48.5% as shown in Fig. 9.

In comparison to the other multiplier designs, the suggested multiplier structure reduces power consumption and achieves average savings in power by 91.5% as shown in Fig. 10. This is indicated by the horizontal green line that goes through the bars of the chart. The vertical axis of the chart represents the power results on a logarithmic scale, while the horizontal axis represents the embedded word sizes. The savings in power is due to the reduction in area complexity of the suggested design over the other designs. Reducing design area reduces parasitic capacitance and hence reduces the total switching activities, which are one of the major sources of power consumption. Also, the systolic nature of the proposed design removes glitches that have a great impact on the consumed power.

In comparison to existing word-serial multipliers, Fig. 11 shows that the presented multiplier structure saves energy by an average of 63.2%. This is indicated by the horizontal green line that runs through the bars. As the figure indicates, the energy results are plotted on a logarithmic scale chart for the different embedded word sizes. The energy savings are due to the proposed design's consumed power being reduced by a significant margin when compared to the adopted designs.

According to the prior analysis, the recommended word-serial multiplier structure surpasses the other competing multiplier designs in terms of area and consumed energy for the various embedded word sizes. As a result, the suggested multiplier is appropriate for IoT edge devices utilized in resource-constrained IoT applications.

5. Summary and Conclusion

We offered in this article a word-serial accelerator structure that performs multiplication in GF(2^n). The proposed multiplier's key feature is its ability to manage the accelerator workload as well as the overall computation time steps required to produce the output results. The experimental results indicate that for various embedded word sizes, the proposed multiplier outperforms the efficient word-serial multipliers previously published in the literature in terms of area and consumed energy, making it more suitable for utilization in IoT edge devices used in resource-constrained IoT applications.

6. Future Work

In future work, we are planning to implement the entire ECC cryptographic processor based on the proposed multiplier accelerator structure to estimate the overall savings in the area and consumed energy of the whole system.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The author would like to acknowledge the support of the National Research Council of Canada (NRC) under Grant No. 51291–52200.

References

[1] B. Pourghebleh, V. Hayyolalam, A.A. Anvigh, Service discovery in the internet of things: review of current trends and research challenges, Wireless Netw. 26 (7) (2020) 5371–5391.

[2] J.H. Anajemba, C. Iwendi, M. Mittal, T. Yue, Improved advance encryption standard with a privacy database structure for iot nodes, in: 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), IEEE, 2020, pp. 201–206.

[3] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, B. Fang, A survey on access control in the age of internet of things, IEEE Internet Things J. 7 (6) (2020) 4682–4696.

[4] M. Shafiq, Z. Tian, A.K. Bashir, X. Du, M. Guizani, Iot malicious traffic identification using wrapper-based feature selection mechanisms, Comput. Secur. 94 (2020) 101863.

[5] S. Su, Z. Tian, S. Li, J. Deng, L. Yin, X. Du, M. Guizani, Iot root union: a decentralized name resolving system for iot based on blockchain, Inform. Process. Manage. 58 (3) (2021) 102553.

[6] Z. Gu, H. Li, S. Khan, L. Deng, X. Du, M. Guizani, Z. Tian, Iepsbp: A cost-efficient image encryption algorithm based on parallel chaotic system for green iot, IEEE Transactions on Green Communications and Networking.

[7] H. Wang, W. Zhang, H. He, P. Liu, D.X. Luo, Y. Liu, J. Jiang, Y. Li, X. Zhang, W. Liu, et al, An evolutionary study of iot malware, IEEE Internet Things J. 8 (20) (2021) 15422–15440.

[8] M. Mittal, S. Vijayal, Detection of attacks in iot based on ontology using sparql, in: 2017 7th International Conference on Communication Systems and Network Technologies (CSNT), IEEE, 2017, pp. 206–211.

[9] J.H. Anajemba, T. Yue, C. Iwendi, M. Alenezi, M. Mittal, Optimal cooperative offloading scheme for energy efficient multi-access edge computation, IEEE Access 8 (2020) 53931–53941.

[10] NIST, Post-quantum cryptography, round 2 submissions, https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions (2020).

[11] D. s. s. NIST, National institute for standards and technology (nist), gaithersburg, md, usa, jan. fips pub 186–2. (2000). http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf.

[12] National Institute of Standards and Technology, FIPS 186-2, Digital Signature Standard (DSS), Federal Information Processing Standards Publication 186-2 (2000).

[13] K.W. Kim, J.C. Jeon, Polynomial basis multiplier using cellular systolic architecture, IETE J. Res. 60 (2) (2014) 194–199.

[14] S. Choi, K. Lee, Efficient systolic modular multiplier/squarer for fast exponentiation over gf(2^m), IEICE Electron. Exp. 12 (11) (2015) 1–6.

[15] K.W. Kim, S.H. Kim, Efficient bit-parallel systolic architecture for multiplication and squaring over gf(2^m), IEICE Electron. Exp. 15 (2) (2018) 1–6.

[16] S. Di Matteo, L. Baldanzi, L. Crocetti, P. Nannipieri, L. Fanucci, S. Saponara, Secure elliptic curve crypto-processor for real-time iot applications, Energies 14 (15) (2021) 4676.

[17] S. Talapatra, H. Rahaman, J. Mathew, Low complexity digit serial systolic montgomery multipliers for special class of gf(2^m), IEEE Trans. Very Large Scale Integr. (VLSI) Sys. 18 (5) (2010) 847–852.

[18] J.H. Guo, C.L. Wang, Hardware-efficient systolic architecture for inversion and division in gf(2^m), IEE Proc. Comput. Digital Tech. 145 (4) (1998) 272–278.

[19] J.S. Pan, C.Y. Lee, P.K. Meher, Low-latency digit-serial and digit-parallel systolic multipliers for large binary extension fields, IEEE Trans. on Circ. and Sys.-I 60 (12) (2013) 3195–3204.

[20] C.-Y. Lee, C.-C. Fan, S.-M. Yuan, New digit-serial three-operand multiplier over binary extension fields for high-performance applications, in: Proc. 2017 2nd IEEE International Conference on Computational Intelligence and Applications, 2017, pp. 498–502.

[21] K. Lee, Low complexity systolic montgomery multiplication over finite fields gf (2^m), J. Korea Soc. Digital Industry Inform. Manage. 18 (1) (2022) 1–9.

[22] P. Siva Ramakrishna, B. Lakshmi, Low-latency area-efficient systolic bit-parallel gf (2^m) multiplier for a narrow class of trinomials, Microelectron. J. 117 (2021) 105275.

[23] L.H. Chen, P.L. Chang, C.-Y. Lee, Y.K. Yang, Scalable and systolic dual basis multiplier over GF(2^m), Int. J. Innovat. Comput., Inform. Control 7 (3) (2011) 1193–1208.

[24] S. Bayat-Sarmadi, M.M. Kermani, R. Azarderakhsh, C.-Y. Lee, Dual-basis superserial multipliers for secure applications and lightweight cryptographic architectures, IEEE Trans. Circ. and Sys.-II 61 (2) (2014) 125–129.

[25] F. Gebali, A. Ibrahim, Efficient scalable serial multiplier over gf(2^m) based on trinomial, IEEE Trans. Very Large Scale Integr. VLSI Syst. 23 (10) (2015) 2322–2326.

[26] A. Ibrahim, F. Gebali, Scalable and unified digit-serial processor array architecture for multiplication and inversion over gf(2^m), IEEE Trans. Circuits Syst. I Regul. Pap. 22 (11) (2017) 2894–2906.

[27] K.W. Kim, J.D. Lee, Efficient unified semi-systolic arrays for multiplication and squaring over gf(2^m), IEICE Electronics Express 14 (12) (2017) 1–10.

[28] F. Gebali, Algorithms and Parallel Computers, John Wiley, New York, USA, 2011.

[29] P.K. Meher, X. Lou, Low-latency, low-area, and scalable systolic-like modular multipliers for gf(2^m) based on irreducible all-one polynomials, IEEE Trans. Circuits Syst. I Regul. Pap. 64 (2) (2016) 399–408.

[30] A. Ibrahim, T. Alsomani, F. Gebali, New systolic array architecture for finite field inversion, Canadian Journal of Electrical and Computer Engineering 40 (1) (2017) 23–30.

[31] A. Ibrahim, F. Gebali, H. El-Simary, A. Nassar, High-performance, low-power architecture for scalable radix 2 montgomery modular multiplication algorithm, Canadian Journal of Electrical and Computer Engineering 34 (4) (2009) 152–157.

[32] A. Ibrahim, T. Alsomani, F. Gebali, Unified systolic array architecture for field multiplication and inversion over gf(2^m), Computers and Electrical Engineering Journal-Elsevier 61 (2017) 104–115.

[33] F. Gebali, A. Ibrahim, Low space-complexity and low power semi-systolic multiplier architectures over gf (2^m) based on irreducible trinomial, Microprocess. Microsyst. 40 (2016) 45–52.

[34] A. Ibrahim, H. Elsimary, F. Gebali, New systolic array architecture for finite field division, IEICE Electronics Express 15 (11) (2018) 1–11.

[35] A. Ibrahim, Efficient parallel and serial systolic structures for multiplication and squaring over gf (2^m), Canadian Journal of Electrical and Computer Engineering 42 (2) (2019) 114–120.

[36] J. Xie, P.K. Meher, Z. Mao, Low-latency high-throughput systolic multipliers over gf(2^m) for nist recommended pentanomials, IEEE Trans. on Circuits and Systems 62 (3) (2015) 881–890.

[37] Y.Y. Hua, J.-M. Lin, C.W. Chiou, C.-Y. Lee, Y.H. Liu, Low space-complexity digit-serial dual basis systolic multiplier over gf(2^m) using hankel matrix and karatsuba algorithm, IET Inf. Secur. 7 (2) (2013) 75–86.

[38] C.-C. Chen, C.-Y. Lee, E.-H. Lu, Scalable and systolic Montgomery multipliers over GF(2^m), IEICE Trans. Fundam. Electron., Commun. Comput. Sci. E91-A (7) (2008) 1763–1771.