Egyptian Informatics Journal 23 (2022) 239–245

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Combined two-dimensional word-based serial-in/serial-out systolic processor for multiplication and squaring over $GF(2^m)$

Atef Ibrahim [a,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

ARTICLE INFO

Article history:
Received 11 August 2020
Accepted 9 January 2022
Available online 28 January 2022

Keywords:
Word-serial systolic arrays
Green cryptoprocessors
Finite field multiplication
Finite field squaring
Ultra-low power devices
Embedded consumer electronics devices
ASIC

ABSTRACT

This paper presents a combined two-dimensional word-based serial-in/serial-out systolic processor for field multiplication and squaring over GF(2^m) to improve hardware utilization and power consumption. The proposed processor is extracted by applying non-linear scheduling and projection functions to the algorithm dependency graph. The extracted processor features scalability that gives the designer more flexibility to control the processor size and the execution time. ASIC Implementation results of the proposed combined two-dimensional word-serial design and the best existing designs show that the proposed structure realizes a considerable saving in the area and consumed energy up to 93.7% and 98.2%, respectively. This makes it more suitable for restricted implementations of cryptographic primitives in resource-constrained consumer electronics devices such as hand-held devices, wearable and implantable medical devices, smart cards, wireless sensor nodes, restricted nodes in the Internet of Things (IoT), and radio frequency identification (RFID) devices.

1. Introduction and related work

Finite field arithmetic operations are substantial in many applications such as RSA cryptography [1], epileptic curve cryptography (ECC) [2], error correction codes [3], and pairing-based cryptography (PBC) [4]. Field multiplication in GF(2^m) is very crucial in several field operations such as modular exponentiation and inversion/division as they performed using a sequence of multiplications.

Most of the previously reported multipliers over GF(2^m) have high area and time complexities that make their realization in resource-constrained consumer electronics devices are highly challenging [5–8]. Therefore, it becomes important to have multiplier architectures that target this type of applications. Word-serial multiplier architectures are reported in the literature to solve this problem. They have a trade-off between speed and area complexities and thus they give the designer more flexibility to reach the desired design. The structures of word-serial multipliers are classified into four types: Serial-In/Serial-Output (SISO) structures, Serial-In/Parallel-Output (SIPO) structures, Parallel-In/ Serial-Output (PISO), and Scalable structures. The polynomial basis word-serial systolic multipliers using SISO structure are presented in [9–13]. The polynomial basis word-serial multipliers with the SIPO structure is reported in [14–17]. The word-serial type-T Gaussian normal basis (GNB) multiplier with PISO structure is reported in [18]. The scalable systolic multiplier structures are reported in [19–25].

Modular exponentiation is a fundamental part of cryptographic algorithms. There are two binary approaches used to compute modular exponentiation: Most Significant Bit (MSB)-first approach and Least Significant Bit (LSB)-first approach. In LSB-first approach, the modular multiplication and squaring operations can be executed concurrently to reduce the processing time. There are many attempts in the literature to combine the multiplication and squaring operations in a unified structure to increase performance and hardware utilization [7,8,26]. To the best of our knowledge, the suggested combined multiplier-squarer structures are dedicated

* Corresponding author at: Computer Engineering Department, College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University Alkharj, 16278 Alkharj, Saudi Arabia.
E-mail addresses: aa.mohamed@psau.edu.sa, atef@ece.uvic.ca

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.eij.2022.01.001

for high-speed applications and do not target the resource-constrained applications.

In this paper, we propose word-based two-dimensional SISO systolic processor for the combined field multiplication and squaring over GF(2^m). It computes the multiplication and squaring operations concurrently using a unified hardware core. The proposed processor can be managed to have the smallest size to fit all the resource-constrained applications that have more restrictions on area and power consumption. Moreover, it has a regular structure and local interconnections that make it more suitable for VLSI implementation. The proposed processor is extracted by applying non-linear scheduling and projection functions to the algorithm dependency graph [27–32]. These non-linear functions provide the required flexibility to control the processor workload and the execution time.

This paper is organized as follows. Section 2 gives a brief explanation to the combined polynomial multiplication-squaring algorithm in $GF(2^m)$. Section 3 develops its associated dependency graph (DG). Section 4 explains the explored two-dimensional word-based SISO systolic processor. Section 5 provides the area and delay complexities of the proposed design and the best of the existing word-serial designs. Section 6 concludes this work.

2. Combined polynomial multiplication-squaring algorithm in $GF(2^m)$

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

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

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

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

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

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

$$R(x) = C(x)D(x) \bmod G(x) \tag{4}$$

$$Q(x) = C(x)C(x) \bmod G(x) \tag{5}$$

The products $R(x)$ and $Q(x)$ can be calculated using the combined algorithm, Algorithm 1, proposed by Choi in [7]. This algorithm calculates three partial polynomials $C(x), R(x)$, and $Q(x)$. Variables C^i, R^i, and Q^i are used to indicate the values of $C(x), R(x)$ and $Q(x)$ at iteration i. d_{i-1} and c_{i-1} represent the $(i-1)^{th}$ coefficients of input polynomials $D(x)$ and $C(x)$, respectively. The initial variables R^0, and Q^0 are assigned zero values and the initial variable C^0 is assigned the coefficients of input polynomial $C(x)$. In each i iteration of the for loop, the intermediate variables are updated as follows:

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

Algorithm 1 Algorithm for multiplication and squaring over GF(2^m) [7].

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

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

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

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

3. Algorithm dependency graph

Algorithm 1 is an example of a Regular Iterative Algorithm (IRA). Reference [27] showed how to obtain the dependency graph (DG) of an RIA algorithm. Fig. 1 shows the DG based on Algorithm 2 for combined polynomial multiplication-squaring in $GF(2^m)$. The nodes in Fig. 1 represent points in the two-dimensional integer domain $\mathbb{D}$ with indices i and j indicates the rows and columns, respectively, and have the ranges:

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

The figure is for the case when $m = 5$ bits. The algorithm has three input variables C, D, and G; and two output variables R and Q. Variables R, Q and G are represented by the vertical lines. Variable C is represented by the slanted lines (red lines). Input bits

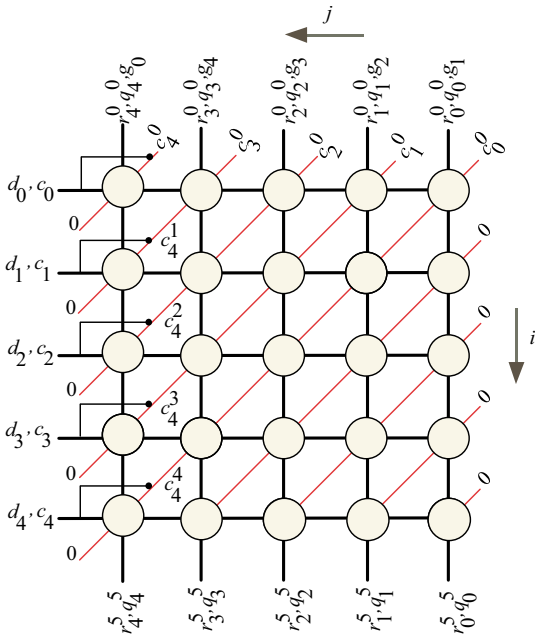

Fig. 1. Dependence graph for the combined multiplication-squaring algorithm for $m = 5$.

c_{i-1}, d_{i-1} along with the resulted intermediate bits c_{m-1}^{i-1} are broadcasted horizontally. The initial bits r_j^0, q_j^0, c_j^0, and $g_{\langle j+1 \rangle}$ are inputs to the DG as shown at the top of Fig. 1. The DG nodes (circles) execute Algorithm 2 main operations, steps 3–5. Output bits r_j^m, q_j^m are produced from the bottom of the DG as indicated in Fig. 1.

The DG in Fig. 1 can be used for design space exploration of the combined multiplication and squaring operations. The design exploration involves finding valid node scheduling functions and mapping or projecting the graph nodes to processing elements (PEs). Reference [27] explains how design space exploration could be performed using affine and non-linear scheduling and projection functions.

The affine scheduling and projection functions can not be used to explore word-serial systolic processors. Thus, our goal is to apply the non-linear scheduling and projection techniques discussed in [27] to the developed algorithm, Algorithm 2, to explore the most efficient two-dimension word-serial systolic processor that is able to satisfy any Input/Output (I/O) limitations/restrictions.

4. Combined two-dimensional SISO multiplier-squarer

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

4.1. Two-dimensional SISO task scheduling

Following the scheduling methodology explained in [27], we can extract the following nonlinear scheduling function to partition $\mathbb{D}$ into $w \times w$ equitemporal zones:

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

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

Fig. 2 shows the node timing (scheduling time) for the case when $m = 5$ and $w = 2$. Notice that we added an extra column on the right and extra row at the bottom to make the number of columns and rows integer multiple of w. In general, we should add μ extra columns and μ extra rows to make the number of columns and rows an integer multiple of w, where $\mu = w\lceil \frac{m}{w} \rceil - m$.

Fig. 3 shows the node timing (scheduling time) for the case when $m = 5$, and $w = 4$. In this case $\mu = 3$, thus we had to add three extra columns on the right and three extra rows at the bottom to make the number of columns and rows an integer multiple of w. Therefore, the LSBs of inputs C and G and LSBs of the initial values of intermediate variables R, Q should be padded by μ zeros on the right as shown in Fig. 3. Also, the MSBs of inputs D and C should be padded by μ zeros at the bottom as shown in the same figure.

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

One last detail needs to be mentioned here and is best explained with reference to two adjacent equitemproal zones executing at times n and $n+1$. Fig. 4 illustrates this situation. The north and east inputs to zone i are available at times n and $n+1$, respectively. However, we notice that input C_n affects only west output C_w and C_e affects only south output C_s. Hence at time n output C_w is valid while output C_s is not since we need to add to it C_e. This will result in increasing the total number of iterations needed to produce the final result by one time step. Therefore, The total number of iterations needed to complete the combined multiplication/squaring computation will be given by:

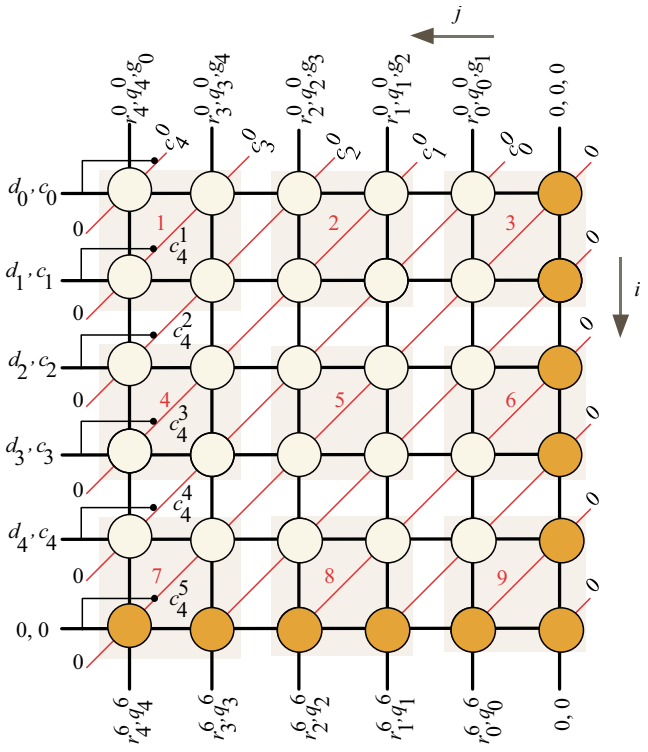

Fig. 2. Scheduling time for the case when $m = 5$ and $w = 2$.

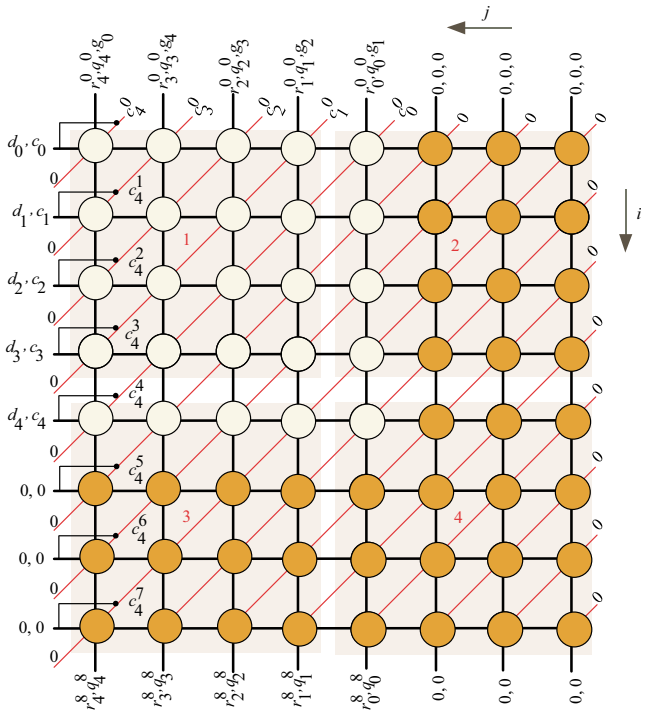

Fig. 3. Scheduling time for the case when $m = 5$ and $w = 4$.

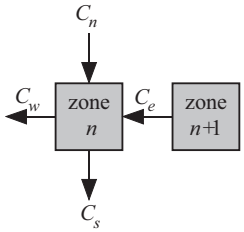

Fig. 4. Data dependencies of two adjacent equitemporal zones from Fig. 3.

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

4.1.1. Two-dimentional SISO task projection

Given the scheduling time in Fig. 3, we note that only $w \times w$ nodes are active at a given time. Following the projection technique explained in [27], we can extract the following nonlinear projection function that maps a point $\mathbf{p}(i,j) \in \mathbb{D}$ of Fig. 3 to a point $\overline{\mathbf{p}}$ in the PE space:

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

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

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

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

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

Our systolic array will now consist of w^2 PEs arranged in w rows and w columns plus the necessary registers. Fig. 5 shows the word-based two-dimensional SISO systolic processor. It consists of $w \times w$ systolic array block as well as input/output registers and FIFO buffers besides two 2-input MUXes to select between the inputs of C and G and their intermediate values. The resulted intermediate words of R, Q, C and the words of G are pipelined through the FIFO buffers of R, Q, C, and G, respectively. These FIFOs have a width size

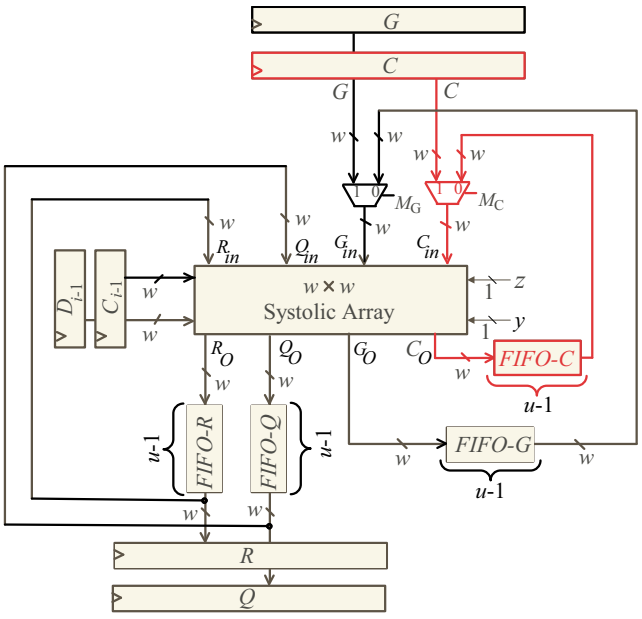

Fig. 5. Word-based two-dimensional SISO systolic processor.

of w bits and a depth size of $u - 1$, where $u = \left\lceil \frac{m}{w} \right\rceil$. The word update block ensures the proper number of bits are extracted from the bottom outputs of the systolic array block shown in Fig. 5. Notic that we added two registers for the input C, the north C register feeds the words of operand C to the systolic array starting from the most significant words, while the east register C_{i-1} feeds the words of operand C to the systolic array starting from the least significant words.

Fig. 6 shows the details of the two-dimensional word-based SISO systolic array for the case when $m = 5$ and $w = 4$. The PEs of the systolic array are divided into two different types with each type has different color. The logic details of the PEs are shown in

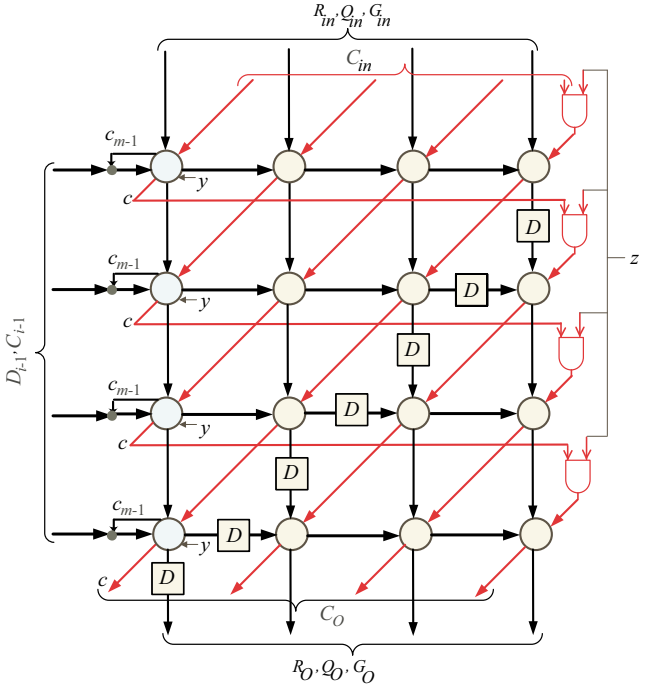

Fig. 6. Word-based two-dimensional SISO systolic array.

Figs. 7 and 8. The light blue PEs have an extra tri-state buffer that is enabled ($y = 0$) at time steps $n = k\lceil\frac{m}{w}\rceil + 1, 0 \leqslant k < \lceil\frac{m}{w}\rceil$, to pass the computed bits of C^{i-1}_{m-1}. These bits are broadcasted along with the input bits of D_{i-1} and C_{i-1} to the remaining nodes of the systolic array to compute the intermediate words of R, Q, and C. Also, they have an extra AND gate to enforce the partial results of the MSBs of C, c^i_m, to be zero at the same time instances. This is controlled by the control signal y shown in Fig. 7. The operation of the two-dimensional SISO systolic processor can be summarized for generic values of m and w as follows:

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

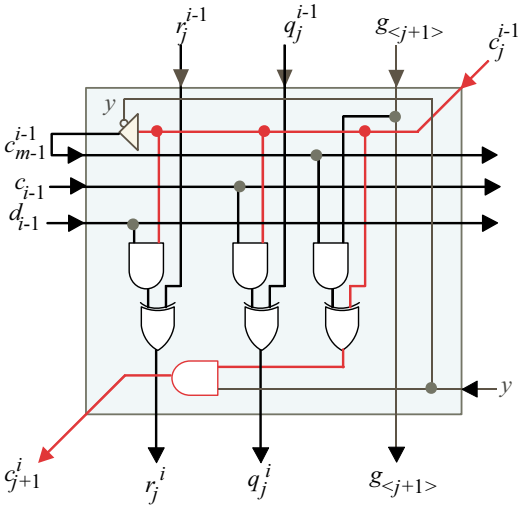

Fig. 7. Light blue PE details of SISO two-dimensional systolic array.

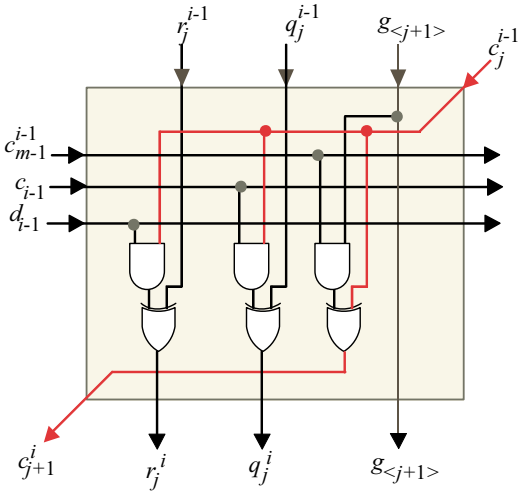

Fig. 8. Light Orange PE details of SISO two-dimensional systolic array.

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

5. Complexities comparison

In this section, we compare the proposed two-dimensional word-serial combined multiplier-squarer structure and the best

of the existing word-serial multiplier structures [12,17,33,34] in terms of area and time complexities. The area is estimated in terms of numbers of Tri-State buffers, 2-input AND gate, 2-input XOR gate, 2-input Multiplexers, and Flip-Flops. The time is represented by latency and Critical Path Delay (CPD).

The estimated area and time complexities of the compared structures are given in Table 1. The notations used in this table can be defined as follows:

1. w is the word size
2. T_A is the delay of 2-input AND gate.
3. T_X is the delay of 2-input XOR gate.
4. T_{MUX} is the delay of 2-to-1 MUX.
5. $F_1 = 7m + m(\lceil \log m \rceil) + w + 3$
6. $F_2 = 2w^2 + 2w(\lceil m/w \rceil) + 4w + 1$
7. $F_3 = 2w^2 + 3w(\lceil m/w \rceil) + 2w$
8. $L_1 = w + \lceil m/w \rceil^2 + \lceil m/w \rceil$
9. $\tau_1 = T_A + (\lceil \log_2 w \rceil + 1)T_X$
10. $\tau_2 = T_A + 2T_X$
11. $\tau_3 = T_A + T_X$
12. $\tau_4 = (w+1)T_A + wT_X + T_{MUX}$

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

The designs in Table 1 are described using VHDL code and synthesized for $m = 409$ and different values of w $(8, 16, 32)$ to obtain real implementation results. We used for synthesizing the NanGate (15 nm, 0.8 V) Open Cell Library and Synopsys tools version 2005.09-SP2. We used the typical corner (V_{DD} = 0.8 V and $T_j = 25°C$) and unit drive strength for all the utilized primitives. The obtained results are listed in Table 2. The design metrics used to compare the proposed and the existing word-serial designs can be defined as follows:

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

For a fair comparison, the compared multiplier structures of [12,17,33,34] should perform multiplication and squaring operations in sequence and this doubles the obtained synthesis results of the time and consumed power/energy of these designs as indicated in Table 2. By observing the results in Table 2, we can conclude the following: 1) The proposed design structure saves area at the different values of w (by percentages ranging from 9.1% to 92.6% at $w = 8$, 11.6% to 93.7% at $w = 16$, and 20.7% to 91.9% at $w = 32$) over the existing designs. 2) The design of Pan [12] saves 40% time, at $w = 8$, over the best of the other designs including the proposed one. 3) The design of Xie [17] saves 0.6% and 26.5% time (at $w = 16$ and $w = 32$, respectively) over the best of the other designs including the proposed one. 4) The proposed design structure saves energy at the different values of w (by percentages ranging from 9.6% to 97.3% at $w = 8$, 17.1% to 97.5% at $w = 16$, and 29.0% to 98.2% at $w = 32$) over the existing designs.

As we notice, the proposed designs have the lower area and consumed energy at most of the embedded word sizes and this makes the proposed designs are suitable for application in resource-constrained consumer electronics devices such as handheld devices, wearable and implantable medical devices, wireless sensor nodes, smart cards, restricted nodes in IoT, and radio frequency identification (RFID) devices.

Table 1
Comparison between different word-serial field multipliers.

Design	Tri-State	AND	XOR	MUXs	Flip-Flops	Latency	CPD
Xie [17]	0	$2mw$	$2mw + 6m - 6\frac{m}{w} + 6$	0	$4mw + 4m + 2w$	$2\lceil m/w \rceil + 2\lceil \log_2 w \rceil$	$2T_X$
Pan [12]	0	$m\sqrt{m}$	$\sqrt{mw}(2+m) + w$	0	F_1	$2\lceil \sqrt{m/w} \rceil$	τ_1
Hua [33]	0	w^2	$w^2 + 4 - 5w + 1^{(1)}$	0	F_2	$6w\lceil m/w \rceil^2$	τ_2
Chen [34]	0	$w^2 + w$	$w^2 + 2w$	$2w^{(2)}$	F_3	L_1	τ_3
Proposed	w	$3w^2 + 2w$	$3w^2$	$2w$	$4w(\lceil m/w \rceil - 1) + 8w$	$\lceil m/w \rceil^2 + 1$	τ_4

(1) Area of 3-input XOR gate as 1.5× a 2-input XOR gate.
(2) Multiplier of [34] uses switches that having same transistor count as 2-input MUX.

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

Multiplier	w	Latency	Area (A) [*Kgates*]	CPD [*ps*]	Time (T) [*ns*]	power (P) [*nW*]	Energy (E) [*fJ*]
Xie [17]	8	324	92.98	50.8	16.46	225.56	3.71
	16	172	146.96	50.8	8.74	375.5	3.28
	32	98	195.13	50.8	4.98	477.4	2.38
Pan [12]	8	48	97.46	206.3	9.90	252.91	2.50
	16	36	123.93	244.4	8.80	320.07	2.82
	32	24	164.34	282.5	6.78	425.09	2.88
Hua [33]	8	259584	7.99	73.4	19053.47	4.35	82.88
	16	129792	10.40	73.4	9526.73	5.85	55.73
	32	64896	19.91	73.4	4763.37	11.15	53.11
Chen [34]	8	11946	10.16	55.2	659.42	5.11	3.37
	16	3678	13.51	55.2	203.03	8.38	1.70
	32	1572	26.58	55.2	86.77	15.95	1.38
Proposed	8	2705	7.26	215.7	583.47	3.88	2.26
	16	677	9.19	407.7	276.01	5.12	1.41
	32	170	15.78	791.7	134.59	7.28	0.98

6. Summary and conclusion

This paper presented new efficient two-dimensional word-based SISO systolic processor to perform the multiplication and squaring operations concurrently over $GF(2^m)$. The proposed systolic processor structure shares the data-path and this leads to saving more area and power resources. We applied non-linear scheduling and projection functions to the algorithm dependency graph to explore the proposed systolic processor core. The applied non-linear scheduling and projection functions give the designer more flexibility to control the processor work load and the execution time. The size of the systolic array in the processor core does not depend on the field size and that makes the proposed design more suitable for implementation in embedded and ultra-low power devices. Implementation results of the proposed two-dimensional combined word-serial processor systolic structure and the best of the existing word-serial multiplication designs show that the proposed structure achieves a significant saving in area and consumed energy at different values of the embedded word sizes. This makes it more suitable for constrained implementations of cryptographic primitives in resource-constrained consumer electronics devices such as hand-held devices, wearable and implantable medical devices, wireless sensor nodes, smart cards, restricted nodes in IoT, and radio frequency identification (RFID) devices.

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 authors extend their appreciation to the Deanship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number (IF-PSAU-2021/01/17867)

References

[1] Rivest RL, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems. Mag Commun ACM 1978;21(2):120–6.
[2] Lidl R, Niederreiter H. Introduction to finite fields and their applications. Cambridge, UK: Cambridge University Press; 1994.
[3] Reed IS, Solmon G. Polynomial codes over certain finite fields. SIAM J Appl Math 1960;8:300–4.
[4] Boneh D, Franklin MK. Identity-based encryption from the Weil pairing. SIAM J Comput 2003;32(3):586–615.
[5] 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(2):566–74.
[6] Kim KW, Jeon JC. Polynomial basis multiplier using cellular systolic architecture. IETE J Res 2014;60(2):194–9.
[7] Choi S, Lee K. Efficient systolic modular multiplier/squarer for fast exponentiation over gf(2^m). IEICE Electron Express 2015;12(11):1–6.
[8] Kim KW, Kim SH. Efficient bit-parallel systolic architecture for multiplication and squaring over gf(2^m). IEICE Electron Express 2018;15(2):1–6.
[9] Kim CH, Hong CP, Kwon S. A digit-serial multiplier for finite field GF(2^m). IEEE Trans Very Large Scale Integr (VLSI) Syst 2005;13(4):476–83.
[10] 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(5):847–52.
[11] Guo JH, Wang CL. Hardware-efficient systolic architecture for inversion and division in gf(2^m). IEE Proc Comput Digital Techniques 1998;145(4):272–8.
[12] Pan JS, Lee CY, Meher PK. Low-latency digit-serial and digit-parallel systolic multipliers for large binary extension fields. IEEE Trans Circ Syst-I 2013;60(12):3195–204.
[13] Lee C-Y, Fan C-C, Yuan S-M. 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. p. 498–502.
[14] Hariri A, Reyhani-Masoleh A. Digit-serial structures for the shifted polynomial basis multiplication over binary extension fields. In: Proc. LNCS Intl Workshop Arithmetic of Finite Fields (WAIFI). p. 103–16.
[15] Kumar S, Wollinger T, Paar C. Optimum digit serial multipliers for curve-based cryptography. IEEE Trans Comput 2006;55(10):1306–11.
[16] Lee C-Y. Super digit-serial systolic multiplier over gf(2^m). In Proc. 6th Int. Conf. Genetic Evolutionary Computing, Kitakyushu, Japan; 2012. pp. 509–513..
[17] Xie J, Meher PK, Mao Z. Low-latency high-throughput systolic multipliers over gf(2^m) for NIST recommended pentanomials. IEEE Trans Circuits Syst 2015;62(3):881–90.
[18] Namin AH, Wu H, Ahmadi M. A word-level finite field multiplier using normal basis. IEEE Trans Comput 2011;60(6):890–5.
[19] Lee C-Y, Chiou CW, Lin JM, Chang CC. Scalable and systolic montgomery multiplier over generated by trinomials. IET Circuits Devices Syst 2007;1(6):477–84.
[20] Chen LH, Chang PL, Lee C-Y, Yang YK. Scalable and systolic dual basis multiplier over GF(2^m). Int J Innov Comput Inf Control 2011;7(3):1193–208.
[21] Orlando G, Paar C. A super-serial Galois fields multiplier for FPGAs and its application to public-key algorithms. In Proc. IEEE Symp. Field-Programm. Custom Comp.; 1999. pp. 232–239..
[22] Bayat-Sarmadi S, Kermani MM, Azarderakhsh R, Lee C-Y. Dual-basis superserial multipliers for secure applications and lightweight cryptographic architectures. IEEE Trans Circ Syst-II 2014;61(2):125–9.
[23] 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(10):2322–6.
[24] Ibrahim A, Gebali F, El-Simary H, Nassar A. High-performance, low-power architecture for scalable radix 2 Montgomery modular multiplication algorithm. Can J Electr Comput Eng 2009;34(4):152–7.
[25] 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(11):2894–906.
[26] Kim KW, Lee JD. Efficient unified semi-systolic arrays for multiplication and squaring over gf(2^m). IEICE Electron Express 2017;14(12):1–10.
[27] Gebali F. Algorithms and Parallel Computers. New York, USA: John Wiley; 2011.
[28] Ibrahim A, Elsimary H, Gebali F. New systolic array architecture for finite field division. IEICE Electron Express 2018;15(11):1–11.
[29] Ibrahim A, Elsimary H, Aljumah A, Gebali F. Reconfigurable hardware accelerator for profile hidden Markov models. Arab J Sci Eng 2016;41(8):3267–77.
[30] Ibrahim A. Scalable digit-serial processor array architecture for finite field division. Microelectron J 2019;85:83–91.
[31] 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;61:104–15.
[32] Ibrahim A, Alsomani T, Gebali F. New systolic array architecture for finite field inversion. Can J Electr Comput Eng 2017;40(1):23–30.
[33] Hua YY, Lin J-M, Chiou CW, Lee C-Y, Liu YH. Low space-complexity digit-serial dual basis systolic multiplier over gf(2^m) using hankel matrix and karatsuba algorithm. IET Inf Secur 2013;7(2):75–86.
[34] 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(7):1763–71.