New systolic array architecture for finite field division

Atef Ibrahim^{1,2a)}, Hamed Elsimary¹, and Fayez Gebali³

¹ Computer Engineering Department, Prince Sattam Bin Abdulaziz University, Alkharj 11942, Riyadh, Saudi Arabia

² Microelectronics Department, Electronics Research Institute, El-Tahrir St., Dokki 12622, Cairo, Egypt

³ Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, V8W 2Y2, Canada

a) attif_ali2002@yahoo.com

Abstract: This paper proposes a new systolic array architecture to perform division operations over $GF(2^m)$ based on the modified Stein's algorithm. The systolic structure is extracted by applying a regular approach to the division algorithm. This approach starts by obtaining the dependency graph for the intended algorithm and assigning a time value to each node in the dependency graph using a scheduling function and ends by projecting several nodes of the dependency graph to a processing element to constitute the systolic array. The obtained design structure has the advantage of reducing the number of flip-flops required to store the intermediate variables of the algorithm and hence reduces the total gate counts to a large extent compared to the other related designs. The analytical results show that the proposed design outperforms the related designs in terms of area (at least 32% reduction in area) and speed (at least 60% reduction in the total computation time) and has the lowest AT complexity that ranges from 80% to 94%.

Keywords: finite field division, systolic arrays, parallel processing, hardware security, cryptosystems, hardware accelerators

Classification: Integrated circuits

References

- [1] N. Takagi, *et al.*: "A fast algorithm for multiplicative inversion in $gf(2^m)$ using normal basis," IEEE Trans. Comput. **50** (2001) 394 (DOI: [10.1109/12.926155](https://doi.org/10.1109/12.926155)).
- [2] B. Rashidi, *et al.*: "High-performance and high-speed implementation of polynomial basis Itoh-Tsujii inversion algorithm over $GF(2^m)$," IET Inf. Secur. **11** (2017) 66 (DOI: [10.1049/iet-ifs.2015.0461](https://doi.org/10.1049/iet-ifs.2015.0461)).
- [3] R. P. Brent and H. T. Kung: "Systolic VLSI arrays for polynomial GCD computation," IEEE Trans. Comput. **C-33** (1984) 731 (DOI: [10.1109/TC.1984.5009358](https://doi.org/10.1109/TC.1984.5009358)).
- [4] Z. Yan, *et al.*: "Hardware-efficient systolic architectures for inversions in $GF(2^m)$," IEE Proc. Inf. Secur. **152** (2005) 31 (DOI: [10.1049/ip-ifs:20055017](https://doi.org/10.1049/ip-ifs:20055017)).
- [5] A. K. Daneshbeh and M. A. Hasan: "A class of unidirectional bit serial systolic

- architectures for multiplicative inversion and division over $GF(2^m)$,” IEEE Trans. Comput. **54** (2005) 370 (DOI: [10.1109/TC.2005.35](https://doi.org/10.1109/TC.2005.35)).
- [6] J. Fan, *et al.*: “Design and design methods for unified multiplier and inverter and its application for HECC,” Integration **44** (2011) 280 (DOI: [10.1016/j.vlsi.2011.04.001](https://doi.org/10.1016/j.vlsi.2011.04.001)).
- [7] M. Mozaffari-Kermani, *et al.*: “Reliable concurrent error detection architectures for extended euclidean-based division over $gf(2^m)$,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **22** (2014) 995 (DOI: [10.1109/TVLSI.2013.2260570](https://doi.org/10.1109/TVLSI.2013.2260570)).
- [8] W.-C. Lin, *et al.*: “A combined multiplication/division algorithm for efficient design of ECC over $GF(2^m)$,” TENCON 2007 - 2007 IEEE Region 10 Conference (2007) 1 (DOI: [10.1109/TENCON.2007.4429017](https://doi.org/10.1109/TENCON.2007.4429017)).
- [9] M. Shieh, *et al.*: “An efficient multiplier/divider design for elliptic curve cryptosystem over $GF(2^m)$,” J. Inf. Sci. Eng. **25** (2009) 1555.
- [10] W.-C. Lin, *et al.*: “Design of high-speed bit-serial divider in $GF(2^m)$,” IEEE International Symposium on Circuits and Systems (ISCAS’10) (2010) 713 (DOI: [10.1109/ISCAS.2010.5537479](https://doi.org/10.1109/ISCAS.2010.5537479)).
- [11] N. Takagi: “A vlsi algorithm for modular division based on the binary GCD algorithm,” IEICE Trans. Fundamentals **E81-A** (1998) 724.
- [12] C.-H. Wu, *et al.*: “High-speed, low-complexity systolic designs of novel iterative division algorithms in $GF(2^m)$,” IEEE Trans. Comput. **53** (2004) 375 (DOI: [10.1109/TC.2004.1261843](https://doi.org/10.1109/TC.2004.1261843)).
- [13] Z. Yan and D. V. Sarwate: “New systolic architectures for inversion and division in $GF(2^m)$,” IEEE Trans. Comput. **52** (2003) 1514 (DOI: [10.1109/TC.2003.1244950](https://doi.org/10.1109/TC.2003.1244950)).
- [14] F. Gebali: *Algorithms and Parallel Computers* (Wiley, New York, 2011).
- [15] A. Ibrahim, *et al.*: “Processor array architectures for scalable radix 4 montgomery modular multiplication algorithm,” IEEE Trans. Parallel Distrib. Syst. **22** (2011) 1142 (DOI: [10.1109/TPDS.2010.196](https://doi.org/10.1109/TPDS.2010.196)).
- [16] F. Gebali and A. Ibrahim: “Efficient scalable serial multiplier over $GF(2^m)$ based on trinomial,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **23** (2015) 2322 (DOI: [10.1109/TVLSI.2014.2359113](https://doi.org/10.1109/TVLSI.2014.2359113)).
- [17] A. Ibrahim and F. Gebali: “Low power semi-systolic architectures for polynomial-basis multiplication over $GF(2^m)$ using progressive multiplier reduction,” J. Signal Process. Syst. **82** (2016) 331 (DOI: [10.1007/s11265-015-1000-x](https://doi.org/10.1007/s11265-015-1000-x)).
- [18] F. Gebali and 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 (DOI: [10.1016/j.micpro.2015.11.016](https://doi.org/10.1016/j.micpro.2015.11.016)).
- [19] A. Ibrahim, *et al.*: “Low-power, high-speed unified and scalable word-based radix 8 architecture for montgomery modular multiplication in $GF(p)$ and $GF(2^n)$,” Arab. J. Sci. Eng. **39** (2014) 7847 (DOI: [10.1007/s13369-014-1363-5](https://doi.org/10.1007/s13369-014-1363-5)).
- [20] A. Ibrahim, *et al.*: “Systolic array architectures for Sunar-Koç optimal normal basis type ii multiplier,” IEEE Trans. VLSI Syst. **23** (2015) 2090 (DOI: [10.1109/TVLSI.2014.2358196](https://doi.org/10.1109/TVLSI.2014.2358196)).
- [21] A. Ibrahim, *et al.*: “High-performance, low power architecture for scalable radix 2 montgomery modular multiplication algorithm,” Can. J. Electr. Comp. Eng. **34** (2009) 152 (DOI: [10.1109/CJECE.2009.5599422](https://doi.org/10.1109/CJECE.2009.5599422)).
- [22] A. Ibrahim and F. Gebali: “Scalable and unified digit-serial processor array architecture for multiplication and inversion over $GF(2^m)$,” IEEE Trans. Circuits Syst. I, Reg. Papers **64** (2017) 2894 (DOI: [10.1109/TCSI.2017.2691353](https://doi.org/10.1109/TCSI.2017.2691353)).

1 Introduction

Polynomial division over finite fields is extensively used in several applications such as data coding, error detection, digital communication systems, cryptography, and signal processing. Among the arithmetic operations in finite fields, division has been considered as the most complicated and expensive finite-field operation. Therefore, there are many algorithms are presented in the literature to perform field division with their high performance realization in both software and hardware. These algorithms can be computed based on several schemes such as Fermat's little theorem [1, 2], Extended Euclid's algorithm (EEA) [3, 4, 5, 6, 7], and Extended Stein's algorithm (ESA) [8, 9, 10, 11, 12].

There are two hardware techniques used to implement the field division algorithms. The first technique is the conventional technique that is based on Lookup tables and is efficient for VLSI implementation of field division algorithms over $GF(2^m)$ for small field size m [3, 6]. When m gets larger, we can not easily use this technique in VLSI implementations due to the increasing overhead cost of area. The second technique is the systolic array technique that is considered the efficient hardware technique used to implement division algorithms over $GF(2^m)$ for large values of field size m [12, 13]. This is due to the distinguished features of the systolic architectures such as regularity, modularity, concurrency, and local communication between processing elements that makes them more suited to high performance VLSI system design.

In this paper, we propose new systolic array architecture to perform modular division over $GF(2^m)$ based on the ESA scheme [12]. The architecture consists of one-dimensional systolic array of processing elements (PEs) and has area and AT complexities of $O(m)$ and $O(m^2)$, respectively. The latency of the proposed design is $2m - 1$ that makes it outperforms the previously reported designs in terms of speed.

The paper is organized as follows: Section 2 provides a brief discussion about the adopted division algorithm over $GF(2^m)$ [12]. Section 3 explains the proposed divider systolic structure. Section 4 shows the complexities of the proposed design and compares it with the previously reported designs. Section 5 concludes this work.

2 Division over $GF(2^m)$

Let $G(x) = \sum_{i=0}^m g_i x^i$ be the irreducible polynomial used to define a finite field over $GF(2^m)$, where $g_i \in \{0, 1\}$, for $0 < i < m$, and $g^m = g^0 = 1$. We can represent any field element $Q \in GF(2^m)$ in the polynomial form as $Q(x) = \sum_{i=0}^{m-1} q_i x^i$, where $q_i \in \{0, 1\}$, for $0 \leq i < m$. Also, we can represent the reciprocal polynomial $Q^*(x)$ of $Q(x)$ as $Q^*(x) = \sum_{i=0}^{m-1} q_i x^{m-i-1}$, where $q_i \in \{0, 1\}$, for $0 \leq i < m$.

In [12], the authors proposed the following division algorithm, Algorithm 1, to obtain $V(x) = A(x)/B(x) \bmod G(x)$. In this algorithm, Δ represents the upper bounds on $\deg(R)$ and $\deg(S)$ of the intermediate polynomials R and S , where $\deg()$ indicates the polynomial degree.

Algorithm 1 Finite field division algorithm [12].

```

1: Input:  $A(x)$ ,  $B(x)$ ,  $G(x)$ , and  $\Delta$ 
2: Output:  $V(x) = (A(x)/B(x)) \bmod G(x)$ 
3: Initialization:  $R^0(x) \leftarrow B(x)$ ,  $S^0(x) \leftarrow G(x)$ ,  $U^0(x) \leftarrow A(x)$ ,  $V^0(x) \leftarrow 0$ ,
    $P(x) \leftarrow G(x)$ ,  $\Delta^0 \leftarrow -1$ 
4: Algorithm:
5: for  $1 \leq i \leq 2m - 1$  do
6:   if  $r_0^{i-1} = 1$  then
7:     if  $\Delta^{i-1} < 0$  then
8:        $R^i(x) \leftarrow (R^{i-1}(x) + S^{i-1}(x))/x$ 
9:        $S^i(x) \leftarrow R^{i-1}(x)$ 
10:       $U^i(x) \leftarrow ((U^{i-1}(x) + V^{i-1}(x))/x) \bmod P(x)$ 
11:       $V^i(x) \leftarrow U^{i-1}(x)$ 
12:       $\Delta^i \leftarrow -\Delta^{i-1}$ 
13:    else
14:       $R^i(x) \leftarrow (R^{i-1}(x) + S^{i-1}(x))/x$ 
15:       $S^i(x) \leftarrow S^{i-1}(x)$ 
16:       $U^i(x) \leftarrow ((U^{i-1}(x) + V^{i-1}(x))/x) \bmod P(x)$ 
17:       $V^i(x) \leftarrow V^{i-1}(x)$ 
18:       $\Delta^i \leftarrow \Delta^{i-1} - 1$ 
19:    end if
20:  else
21:     $R^i(x) \leftarrow R^{i-1}(x)/x$ 
22:     $S^i(x) \leftarrow S^{i-1}(x)$ 
23:     $U^i(x) \leftarrow (U^{i-1}(x)/x) \bmod P(x)$ 
24:     $V^i(x) \leftarrow V^{i-1}(x)$ 
25:     $\Delta^i \leftarrow \Delta^{i-1} - 1$ 
26:  end if
27: end for

```

For large values of field size m , the counter used to compute the variable Δ will dominate the Critical Path Delay (CPD) and reduces the divider speed to a large extent. To reduce the effect of increasing CPD, Δ is replaced by two variables h and D such that [12]: h denotes the sign of Δ ($h = 0$ if $\Delta > 0$ and $h = 1$ if $\Delta < 0$) and D is a one-hot $m + 1$ -bit vector defined by $D = 2^{|\Delta|} = 2^{\Delta(-1)^h}$. This representation of Δ replaces the counter operation with a shift for the value of D and a bit inversion for the sign h as shown in Algorithm 2.

In Algorithm 2, variables $R^i(x)$, $S^i(x)$, $U^i(x)$, and $V^i(x)$ represent the computed values of polynomials $R(x)$, $S(x)$, $U(x)$, and $V(x)$, respectively, after iteration i . We can express the recurrence equations for updating $R^i(x)$, $U^i(x)$, $S^i(x)$, and $V^i(x)$ in bit-level form as:

$$r_{j-1}^i = r_j^{i-1} + c3^i s_j^{i-1} \quad (1)$$

$$u_{j-1}^i = u_j^{i-1} + c3^i v_j^{i-1} + c4^i p_j^{i-1} \quad (2)$$

$$s_j^i = c5^i r_j^{i-1} + \overline{c5^i} s_j^{i-1} \quad (3)$$

$$v_j^i = c5^i u_j^{i-1} + \overline{c5^i} v_j^{i-1} \quad (4)$$

Algorithm 2 Modified Finite field division algorithm [12].

```

1: Input:  $A(x)$ ,  $B(x)$ ,  $G(x)$ ,  $h$ ,  $D$ 
2: Output:  $V(x) = (A(x)/B(x)) \bmod G(x)$ 
3: Initialization:  $R^0(x) \leftarrow B(x)$ ,  $S^0(x) \leftarrow G(x)$ ,  $U^0(x) \leftarrow A(x)$ ,  $V^0(x) \leftarrow 0$ ,  $P(x) \leftarrow G(x)$ ,  $h^0 \leftarrow 1$ ,  $D^0 \leftarrow 2$ 
4: Algorithm:
5: for  $1 \leq i \leq 2m - 1$  do
6:   if  $r_0^{i-1} = 1$  then
7:     if  $h^{i-1} = 1$  and  $d_0^{i-1} = 0$  then
8:        $R^i(x) \leftarrow (R^{i-1}(x) + S^{i-1}(x))/x$ 
9:        $S^i(x) \leftarrow R^{i-1}(x)$ 
10:       $U^i(x) \leftarrow ((U^{i-1}(x) + V^{i-1}(x))/x) \bmod P(x)$ 
11:       $V^i(x) \leftarrow U^{i-1}(x)$ 
12:       $h^i \leftarrow 0$ 
13:    else
14:       $R^i(x) \leftarrow (R^{i-1}(x) + S^{i-1}(x))/x$ 
15:       $S^i(x) \leftarrow S^{i-1}(x)$ 
16:       $U^i(x) \leftarrow ((U^{i-1}(x) + V^{i-1}(x))/x) \bmod P(x)$ 
17:       $V^i(x) \leftarrow V^{i-1}(x)$ 
18:      if  $h^{i-1} = 0$  and  $d_0^{i-1} = 0$  then
19:         $D^i \leftarrow D^{i-1}/2$ 
20:      else
21:         $D^i \leftarrow 2D^{i-1}$ 
22:         $h^i \leftarrow 1$ 
23:      end if
24:    end if
25:  else
26:     $R^i(x) \leftarrow R^{i-1}(x)/x$ 
27:     $S^i(x) \leftarrow S^{i-1}(x)$ 
28:     $U^i(x) \leftarrow (U^{i-1}(x)/x) \bmod P(x)$ 
29:     $V^i(x) \leftarrow V^{i-1}(x)$ 
30:    if  $h^{i-1} = 0$  and  $d_0^{i-1} = 0$  then
31:       $D^i \leftarrow D^{i-1}/2$ 
32:    else
33:       $D^i \leftarrow 2D^{i-1}$ 
34:       $h^i \leftarrow 1$ 
35:    end if
36:  end if
37: end for

```

where,

$$c3^i = r_0^{i-1} \quad (5)$$

$$c4^i = u_0^{i-1} + r_0^{i-1}v_0^{i-1} \quad (6)$$

$$c5^i = r_0^{i-1}h^{i-1}\overline{d_0^{i-1}} \quad (7)$$

Converting variable D^i to bit-level from will increase the complexity of hardware design, thus we preferred to keep it in the word-level form. It can be updated along with h^i using the following recurrence equations:

$$D^i = c6^i(D^{i-1}/2) + \overline{c6^i}(2D^{i-1}) \quad (8)$$

$$h^i = \overline{c6^i} \quad (9)$$

where,

$$c6^i = \overline{(r_0^{i-1}h^{i-1})d_0^{i-1}} \quad (10)$$

The subscript j used in the previous equations refers to the j^{th} coefficient bit of polynomials $R(x)$, $U(x)$, $S(x)$, $V(x)$, and $P(x)$, where $0 \leq j \leq m$. u_0^{i-1} , r_0^{i-1} , v_0^{i-1} , and d_0^{i-1} indicate the Least Significant Bits (LSBs) of $U(x)$, $R(x)$, $V(x)$, and D , respectively, at iteration i .

3 Proposed systolic array design of the division algorithm

The proposed systolic array design is obtained by using an approach previously explained by the second author in [14]. This approach starts by obtaining the dependency graph (DG) for the intended algorithm and assigning a time value to each node in the DG using a scheduling function as explained by authors in [14, 15, 16, 17, 18, 20]. The approach ends by projecting several nodes of the DG to a processing element (PE) to constitute the systolic array [14, 17, 19, 20, 21, 22].

We can extract the DG from Equations (1)–(10) for $m = 5$ as shown in Fig. 1. The equations has two indexes (i, j) , thus the computation domain is the convex hull in the two-dimensional (2D) space and the circles in this domain define the operations. Inputs of the DG are s_j^0 , r_j^0 , u_j^0 , v_j^0 , p_j^0 , D^0 , and h^0 . The nodes in right most column (gray nodes) compute control signals $c3^i$, $c4^i$ and $c5^i$. These signals are broadcasted to all nodes in the same row i . The red lines (slanted lines) denote the updated bits u_{j-1}^i and r_{j-1}^i . p_j^{i-1} bit is assigned to all nodes in each column j . The resulted output bits at the top of DG are the last output bits of the variable V , v_j^{2m-1} .

The systolic array architecture can be extracted by choosing the scheduling vector $\mathbf{S} = [1 \ 0]$ and the projection vector $\mathbf{P} = [0 \ 1]^T$ for the DG. These vectors are obtained from applying the approach previously reported by authors in [14, 17, 20, 21, 22].

Fig. 2 shows the resulted systolic array after applying the suggested mapping vectors on the DG. It consists of $m + 1$ PEs. PE_0 produces control signals $c3^i$, $c4^i$, and $c5^i$ through each i iteration. Also, it updates variables D^i , h^i , and the LSB of V^i , v_0^i . The bits s_0^i and r_0^i are not processed in PE_0 because s_0^i is always equal to ‘1’ and r_0^i is shifted out [9]. PE_j updates r_{j-1}^i , u_{j-1}^i , s_j^i , v_j^i based on Equations (1)–(4). PE_m is a simplified form of PE_j and computes $u_{m-1}^i = c3^i v_m^{i-1} + c4^i$, $r_{m-1}^i = c3^i s_m^{i-1}$, $s_m^i = \overline{c5^i} s_m^{i-1}$. This simplification is resulted from the constant input values $p_m^{i-1} = 1$, $r_m^{i-1} = 0$, $u_m^{i-1} = 0$ applied to PE_m .

The hardware details of the PEs (PE_0 , PE_j , and PE_m) are shown in Figs. 3–5. Multiplexers (Muxes) controlled by $c2$ control signal are used to select between the input signals and the intermediate results generated through different i iterations. $c2$ equal ‘1’ in the first clock cycle to pass the input signals and equal ‘0’ through the

Fig. 1. DG of the Division algorithm for $m = 5$, $0 \leq j \leq m$, and $1 \leq i \leq 2m - 1$.

Fig. 2. Proposed systolic array for different values of m . Superscript β is equal to $2m - 1$.

remaining clock cycles to latch the intermediate results. Variables D^i and h^i are updated based on Equations (8)–(9) and control signal $c6^i$. This signal controls the shifter shown in Fig. 3(a) to set the shift direction (right/left). The shifter hardware details are shown in Fig. 3(b).

The operation of the systolic array can be summarized as follows:

1. At time $n = 1$, the external control signal $c2$ is set to ‘1’ to pass the initial inputs to the D-FFs and registers located inside the PEs.
2. At time $n > 1$, control signals $c3^i$, $c4^i$, and $c5^i$ are produced inside PE_0 and transmitted to the remaining PEs. Also, $c2$ control signal is set to ‘0’ to latch the updated values of the intermediate variables.

Fig. 3. PE_0 hardware details. Squares are D flip-flops with load and clear control inputs. Circles with symbols “ $\times$ ” and “ $+$ ” represent 2-input AND and XOR gates, respectively.

Fig. 4. PE_j hardware details, $1 \leq j < m$.

- At the last time step $n = 2m - 1$, the systolic array produces the output bits of V , v_0^β and v_j^β , from PE_0 and PE_j , $1 \leq j \leq m - 1$, respectively.

4 Performance analysis and comparison

Table I shows a comparison between the proposed systolic design and the related work [5, 10, 12] in terms of area requirements and delay. In this table, T_A , T_X , and T_{MUX} denote the delay of 2-input AND gate, 2-input XOR gate, and 2-input Multiplexer, respectively. The total gate count is estimated in terms of 2-input NAND gate based on the NanGate (15 nm, 0.8 V) Open Cell Library that is based on the NCSU FreePDK15 process kit. The area and delay information of the standard cells is given in Table II.

We notice from Table I that the area requirements and latency of the proposed design are lower than that of the compared systolic designs. The systolic arrays

Fig. 5. PE_m hardware details.

presented in [5, 10] consist of $2m$ and $2m - 1$ PEs, respectively. Besides the large number of PEs used in each systolic array, each PE consists mainly of 2 parts: datapath part and control part. The control part is repeated in each PE and this increases the area complexity of each PE and hence increases the total area complexity of the systolic arrays. Also, more flip-flops (FFs) are added inside the PEs to reduce the critical path delay and this leads to increasing the area complexity and the latency of these designs. The systolic array presented in [12] consists of $m + 1$ PEs like the proposed systolic array, but it employed a pipeline interleaving technique to improve the utilization of the PEs. This resulted in increasing the number of FFs used in the systolic array to a large extent and hence increasing the latency of the resulted systolic array to be $5m - 2$. Due to this reason and the longer critical path delay of this design, the area and time complexities of this design exceeds the proposed one.

The total gate count (area) of the proposed design is lower than the other compared designs because of the reduced number of FFs required to store the intermediate variables. The reduced number of FFs leads to reducing the latency of the proposed design to be $2m - 1$. The lower critical path delay besides the reduced latency of the proposed design lead to reducing the time complexity of the proposed design to be less than the compared designs.

To quantify the analytical results obtained in Table I, Table III shows the estimated values of total gate counts (A), latency (L), critical path delay (CPD) or clock period, total computation time (T), and AT complexity for $m = 233$. The AT complexity is defined as the area (A) multiplied by the total computation time (T). The sum of critical path delay, setup delay (T_{setup}) and propagation delay ($T_{Clk \rightarrow Q}$) of the D-FF represents the clock period. From Table III, we notice that the proposed design outperforms other compared designs in terms of area (at least 32%

reduction in area) and speed (at least 60% reduction in total computation time) and has the lowest AT complexity that ranges from 80% to 94%.

Table I. Complexity comparison of different systolic designs over $GF(2^m)$.

Design	Latency	Critical Path delay	Number of Flip-Flops	Total Gate Count
Daneshbeh [5]	5m-4	$2T_A + T_X + T_{MUX}$	36m	222m
Lin [10]	5m-3	$T_A + 2T_X$	34m-17	215.2m-73.1
Wu [12]	5m-2	$T_A + 2T_X + T_{MUX}$	10m+5	68.3m+96
Proposed	2m-1	$T_A + 2T_X$	5m+3	46.5m+25

5 Conclusion and future work

The details of a new systolic array architecture of the extended stein's division algorithm over $GF(2^m)$ are explained in this article. We used a previously reported systematic approach to extract the proposed design. The approach starts by obtaining the dependency graph for the intended algorithm and assigning a time value to each node in the DG using a scheduling function and ends by projecting several nodes of the DG to a processing element to constitute the systolic array. The obtained design structure has a reduced number of flip-flops required to store the intermediate variables of the algorithm and hence a significant reduction in the total gate counts compared to the related designs. The analytical results showed that the proposed design outperforms the previously reported related designs in terms of area and speed. As a future work, the proposed design and the other related designs will be described using VHDL programming language and will be implemented on FPAG and ASIC to obtain an accurate real implementation results. Due to the lower area and delay achieved by the proposed design, we expect that the total power consumption and energy of the proposed design will be lower than that of the previously reported related designs. This makes the proposed design more suitable for resource-constrained applications that have more restrictions on area and power.

Table II. Area and delay values in terms of 2-input NAND gate for standard cells.

	INV	AND/OR	XOR	MUX	Flip-Flops
Area (2-input NAND)	0.6	1.2	2.5	2.5	4.3
Delay (ps)	5.8	11.3	12.7	12.4	32.8*

(*) Delay of the flip-flop is equal to the sum of the propagation delay ($T_{CLK \rightarrow Q}$) and setup time (T_{setup}) of the flip-flop.

Table III. Estimated area and delay results of different systolic designs over GF(2^{233}).

Design	A [K gates]	L [cycles]	CPD [ps]	T [ns]	AT
Daneshbeh [5]	51.7	1161	47.7	55.4	2864
Lin [10]	50.1	1162	36.7	42.6	2134
Wu [12]	16.0	1163	49.1	57.1	914
Proposed	10.9	465	36.7	17.1	186

Acknowledgements

This project was supported by the Deanship of Scientific Research at Prince Sattam Bin Abdulaziz University under the research project # 2017/01/6958.