Received October 1, 2016, accepted November 13, 2016, date of publication December 13, 2016, date of current version January 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2016.2639039

Efficient Scalable Digit-Serial Inverter Over GF(2^m) for Ultra-Low Power Devices

ATEF IBRAHIM[1,2], (Member, IEEE), TURKI F. AL-SOMANI[3], (Senior Member, IEEE), AND FAYEZ GEBALI[4], (Senior Member, IEEE)
[1]Department of Computer Engineering, Prince Sattam Bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[2]Department of Microelectronics, Electronics Research Institute, Cairo 12622, Egypt
[3]Department of Computer Engineering, Umm Al-Qura University, Mecca 21955, Saudi Arabia
[4]Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 2Y2, Canada

Corresponding author: A. Ibrahim (attif_ali2002@yahoo.com)

This work was supported in part by the NSTIP (MAARIFAH) Strategic Technologies Program under Grant 13-INF2541-10 and in part by the Science and Technology Unit, Umm Al-Qura University, Saudi Arabia.

ABSTRACT This paper proposes a novel scalable digit-serial inverter structure with low space complexity to perform inversion operation in $GF(2^m)$ based on a previously modified extended Euclidean algorithm. This structure is suitable for fixed size processor that only reuse the core and does not require to modulate the core size when m modified. This structure is extracted by applying a nonlinear methodology that gives the designer more flexibility to control the processing element workload and also reduces the overhead of communication between processing elements. Implementation results of the proposed scalable design and previously reported efficient designs show that the proposed scalable structure achieves a significant reduction in the area ranging from 83.0% to 88.3% and also achieves a significant saving in energy ranging from 75.0% to 85.0% over them, but it has lower throughput compared to them. This makes the proposed design more suitable for constrained implementations of cryptographic primitives in ultra-low power devices such as wireless sensor nodes and radio frequency identification (RFID) devices.

INDEX TERMS Scalable systolic arrays, hardware security, finite field inversion, ultra-low power devices, ASIC.

I. INTRODUCTION AND RELATED WORK

In resource-constrained platforms, the implementation of public key cryptosystems (PKC) is a challenge due to the limitations of area and power consumption [1]. Compared to other PKC algorithms, elliptic curve cryptography (ECC) algorithms have the merit of giving the same level of security using smaller key sizes and this leads to using these algorithms in resource constrained applications. Recently, there are a lot of hardware implementations of ECC that meet the area, energy and timing limitations of these applications [2]–[5]. These implementations are mainly concentrated on the efficient implementation of the operations of field multiplication and field inversion as they are the most costly operations in ECC cryptography. The field inversion is much slower and more expensive in power consumption than the field multiplication. Thus, improving the performance of the inversion operation will lead to a total improvement in the performance of the ECC system.

The systolic architectures for binary field inversion can be classified into three basic types. The first type of the systolic architectures composed of two-dimensional arrays of processing elements (PEs) and have area complexity of O(m^2) [6], [7]. These architectures are more suitable for high-throughput applications that require small values of m, but they are not suitable for applications that require large values of m due to the high area complexity that makes implementing these architectures on a single chip infeasible. The second type of systolic architectures are consists of one-dimensional arrays of PEs and have low area complexity of O(m). These architectures include folded bit-parallel architectures [8], bit-serial architectures [8], [9], and in place bit parallel architectures [8], [10]. The throughput of these architectures may be very slow for some real-time applications. The third type of systolic architectures is the Digit serial architectures [9], [11], [12] that consider the tradeoffs between area complexity and throughput in its circuit implementation. A digit-serial architecture with a digit size of d bits has area complexity of O(dm). For different sizes of d, we can easily obtain different throughputs.

In this paper, we propose a scalable digit-serial architecture that is suitable for resource-constrained devices to perform inversion operation in $GF(2^m)$ based on a previously modified extended Euclidean algorithm. This architecture is composed of a one-dimensional array of PEs and has low area complexity of O(T), where T is the number of PEs in the systolic array. This architecture is explored by applying a nonlinear technique, proposed by authors in [13] and [14], to the inversion algorithm.

The paper is organized as follows: Section II discusses the adopted extended Euclidean-based finite field inversion algorithm over GF(2^m). Section III shows how to parallelize this algorithm using nonlinear data scheduling and projection techniques. Section IV discusses the proposed scalable architecture. Section V discusses the proposed design complexity and compares it to the previous work. Finally, Section VI provides the conclusions of this work.

II. FINITE FIELD INVERSION

A finite field over GF(2^m) could be defined using the irreducible polynomial:

$$Q(x) = x^m + q_{m-1}x^{m-1} + \cdots + q_2x^2 + q_1x + 1 \qquad (1)$$

where $q_i \in GF(2)$ for $0 < i < m$.

A field element A in $GF(2^m)$ can be represented by the polynomial:

$$A(x) = \sum_{i=0}^{m-1} a_i x^i \qquad (2)$$

where $a_i \in GF(2)$ for $0 \le i < m$.

Suppose a polynomial $\hat{A}$ in GF(2) represents the multiplicative inverse of $A(x)$ such that $\hat{A}(x)A(x) \equiv 1 \bmod Q(x)$, where $\hat{A}(x)$ is denoted as $[A^{-1}(x) \bmod Q(x)]$. The most commonly used inversion algorithms are based on Fermat's little theorem, extended Euclidean algorithm (EEA), and Gaussian elimination. In practice, EEA is mostly used to carry out inversion.

Yan *et al.* [15] proposed a modified EEA-based inversion algorithm that solves the problem of long division needed in each iteration of the conventional EEA-based inversion algorithm by exchanging the degree comparison with a ring counter. This algorithm computes four intermediate polynomials, $R(x)$, $S(x)$, $Y(x)$, and $H(x)$ that are stored in $m+1$-bit registers with bits numbered as $m, m-1, \cdots, 1, 0$. The most significant bits of registers R and S are the highest degree terms of the $R(x)$ and $S(x)$ (i.e., $R(x) = r_m x^m + \cdots + r_1 x^1 + r_0 x^0$ and similarly for $S(x)$), while the least significant bits of registers Y and H are the highest degree terms of $Y(x)$ and $H(x)$ (i.e., $Y(x) = y_m x^0 + \cdots + y_1 x^{m-1} + y_0 x^m$ and similarly for $H(x)$). Yan *et al.* [15] proved that there is no need to store the m^{th} bit of $R(x)$, $S(x)$, $Y(x)$, and $H(x)$. Thus, the registers can be shortened from $m+1$-bit to m-bit. The ring counter D bits are ordered from right to left as $(d_{m-1}d_{m-2}\cdots d_0)$. The complement of control bit $c1$ is represented as $\overline{c1}$. In the initial step of the algorithm, the coefficients of the irreducible polynomial $Q(x)$ and the coefficients of polynomial $A(x)$ are assigned to the variables R^0 and S^0, respectively. Since the MSB of Q(x) is always equal to 1, this bit does not need to be computed or stored as mentioned in [15]. Thus, Q(x) coefficients can be stored in a register of size m.

Algorithm 1 Pseudo Code of the Bit-Level EEA-Based Inversion Algorithm

```
1:  INITIALIZE
2:  $R^0 = (r^0_{m-1} \cdots r^0_1 r^0_0) \leftarrow (q^0_{m-1} \cdots q^0_1 1)$
3:  $S^0 = (s^0_{m-1} \cdots s^0_1 s^0_0) \leftarrow (a^0_{m-1} \cdots a^0_1 a^0_0)$
4:  $H^0 = (h^0_{m-1} \cdots h^0_1 h^0_0) \leftarrow (0 \cdots 00)$
5:  $Y^0 = (y^0_{m-1} y^0_{m-2} \cdots y^0_1 y^0_0) \leftarrow (00 \cdots 001)$
6:  $D^0 = (d^0_{m-1} \cdots d^0_2 d^0_1 d^0_0) \leftarrow (0 \cdots 010)$
7:  $sign^0 \leftarrow 1$
8:  $d^{i-1}_{-1}, s^{i-1}_{-1}, h^{i-1}_{-1}, d^{i-1}_{m} = 0,$        for $1 \le i < 2m$
9:  for $1 \le i < 2m$ do
10:         $c1^i = s^{i-1}_{m-1}$
11:         $c2^i = c1^i AND\ sign^{i-1}$
12:   if $sign^{i-1} = 1$ then
13:       $sign^i = \overline{c1^i}$
14:   else
15:       $sign^i = d^{i-1}_0$
16:   end if
17:   for $m-1 \le j \le 0$ do
18:       if $c1^i = 1$ then
19:           $s^i_j = r^{i-1}_j + s^{i-1}_{j-1}$
20:           $y^i_j = h^{i-1}_{j-1} + y^{i-1}_j$
21:       else
22:           $s^i_j = s^{i-1}_{j-1}$
23:           $y^i_j = y^{i-1}_j$
24:       end if
25:       if $c2^i = 1$ then
26:           $r^i_j = s^{i-1}_{j-1}$
27:           $h^i_j = y^{i-1}_j$
28:       else
29:           $r^i_j = r^{i-1}_j$
30:           $h^i_j = h^{i-1}_{j-1}$
31:       end if
32:   end for
33:       $D^i \leftarrow \begin{cases} 2D^{i-1}, & \text{if } sign^i = 1 \\ D^{i-1}/2, & \text{if } sign^i = 0 \end{cases}$
34: end for
35: output: $\hat{a}_j = h^{2m-1}_{m-j-1},$        for $0 \le j \le m-1$
```

Algorithm 1 is the bit-level version of the modified EEA-based inversion algorithm of Yan [15]. In this algorithm, the terms r^i_j, s^i_j, y^i_j and h^i_j represent the j-th bit of R, S, Y and H at iteration i, respectively.

III. PARALLELIZATION OF THE INVERSION ALGORITHM

The ranges of i and j indices of Algorithm 1 define a set of points in a convex hull $\mathbb{D}$ in the 2-D integer space, i.e. $\mathbb{D} \subset \mathbb{Z}^2$ [13]. In this algorithm, their are two input variables A and Q; five intermediate m-bit variables R^i, S^i, Y^i, H^i, D^i

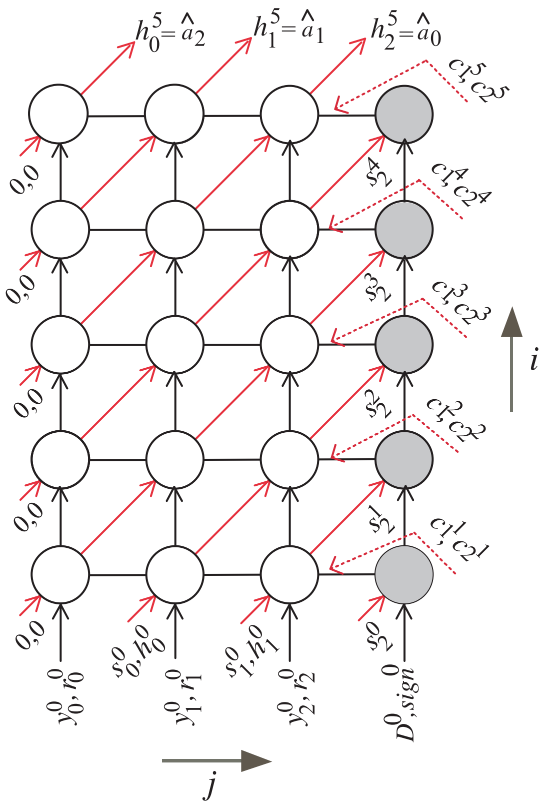

FIGURE 1. Dependence graph of the inversion algorithm for $m = 3$.

and three intermediate 1-bit control variables $c1^i, c2^i, sign^i$; and one output variable H. The input bits $s_{j-1}^0, h_{j-1}^0, y_j^0, r_j^0$, $sign^0$ are shown in the dependency graph (DG) of Figure 1 at the bottom row. The m-bit input vector D^0 also shown in the DG of Figure 1 at the bottom of the gray nodes column. The bits $c1^i, c2^i$ are generated inside the right nodes (gray nodes) in each row i and broadcasted to the remaining nodes in the same row. This is pointed out by the horizontal lines in Figure 1. Also, $sign^i$ bit is generated inside the right nodes (gray nodes) using iteration steps 13 and 15. This is pointed out by the vertical line in the right-most column in Figure 1. The intermediate variables S^i and H^i are updated using iteration steps 19, 22, 27 and 30. This is pointed out by the anti-diagonal lines (red lines) in Figure 1. Since the control nodes (gray nodes) in the right column of the DG do not depend on H^i bit values resulted from the nodes in the previous column, therefore there is no need to connect them to the H^i bits. The iteration steps 20, 23, 26 and 29 of the algorithm updates the intermediate variables of Y^i and R^i. This is pointed out by the vertical lines (black lines) in Figure 1. The flow of data, at each time step, between the nodes is indicated by the arrows. The resulted bits of the multiplicative inverse are the last out bits of the variable H that produced at the top of the graph.

Each point in the DG of Figure 1 is assigned a time value $t(\mathbf{p})$ using timing function $\mathbf{S}$ and a parameter T, which represents the number of nodes to be computed at the same time step.

$$t(\mathbf{p}) = \mathbf{S}\mathbf{p} - \alpha = i\left\lceil \frac{m}{T} \right\rceil - \left\lfloor \frac{j+\mu}{T} \right\rfloor - \alpha \tag{3}$$

$$\mathbf{S} = \left[\left\lceil \tfrac{m}{T} \right\rceil \quad - \left\lfloor \tfrac{ph+\mu}{T} \right\rfloor \right] \tag{4}$$

$$\mu = T\left\lceil \frac{m}{T} \right\rceil - m \tag{5}$$

$$\alpha = -\left\lfloor \frac{m-1+\mu}{T} \right\rfloor \tag{6}$$

where terms $\lfloor ph/T \rfloor$ and $\lceil ph/T \rceil$ represent floor and ceiling functions, respectively, and the 'ph' represents a place holder for the argument.

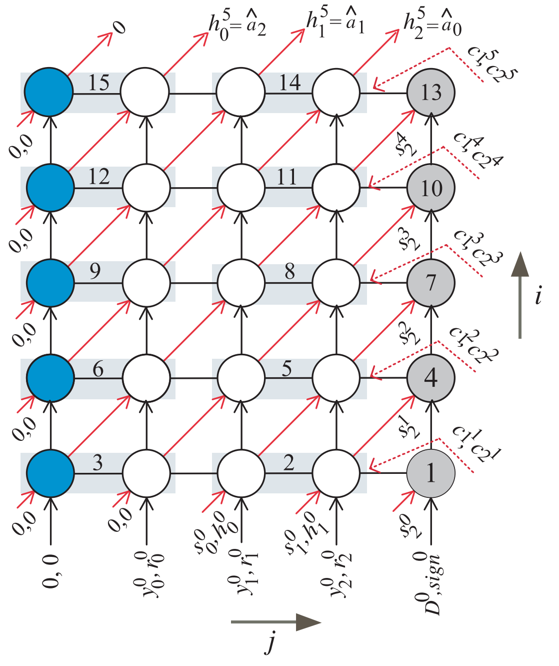

FIGURE 2. Node timing for the inversion algorithm for $m = 3$ and $T = 2$.

The node timing function $\mathbf{S}$ is shown in Figure 2 for $m = 3$ and $T = 2$. The nodes that have the same time index is indicated by the light blue areas. The values in each light blue area represent the time index. This time index depends on the values of both i and j indices. When m is not an integer multiple of T, we notice that the number of nodes processed at each time step are not the same. To have a constant number of nodes processed at each time step, m should be an integer multiple of T. Thus, the value of m should be increased to m' using the following relation:

$$m' = T\left\lceil \frac{m}{T} \right\rceil = m + \mu \tag{7}$$

For the case when $m = 3$ and $T = 2$, we get $m' = 4$ and $\mu = 1$ as shown in the figure. We chose to pad the LSB bits of R, S, H, Y with μ zeros to make m an integer multiple of T. This is indicated by the dark blue nodes shown in Figure 2.

Using the nonlinear scheduling function **S** represented by Eq. (4), we will be able to control the workload per time step and the number of time steps required to complete the execution of the inversion operation. In this case, the workload is equal to $T = 2$ and the inversion operation will require $(2m-1)(\lceil m/T \rceil + 1)$ time steps to complete.

Now, we need to map several nodes of the DG onto a single node that forms the resulting systolic array. The projection technique discussed in [13], [14], [16], [17] can be used to perform this mapping operation. The third author of this paper explained in [13] how to carry out the projection operation using a projection Matrix **P**. Since our algorithm is two-dimensional, **P** will reduce to a row vector. The valid projection matrices associated with the scheduling function **S** are as follows:

$$\mathbf{p}_1 = \begin{bmatrix} 0 & (ph+\mu) \bmod T \end{bmatrix} \tag{8}$$

$$\mathbf{p}_2 = \begin{bmatrix} 0 & \lfloor [(ph+\mu) \bmod T]/W \rfloor \end{bmatrix} \tag{9}$$

$$\mu = T\lceil m/T \rceil - m \tag{10}$$

Each scalable systolic array configuration is associated with one projection matrix, therefore the processor design space allows for two scalable systolic designs. The scalable systolic array related to the projection matrix $\mathbf{P}_2$ has a high control complexity and is not suitable for VLSI implementation. Thus, this design will be ignored in this research paper. In the following section, we will investigate the scalable systolic array related to the projection matrix $\mathbf{P}_1$.

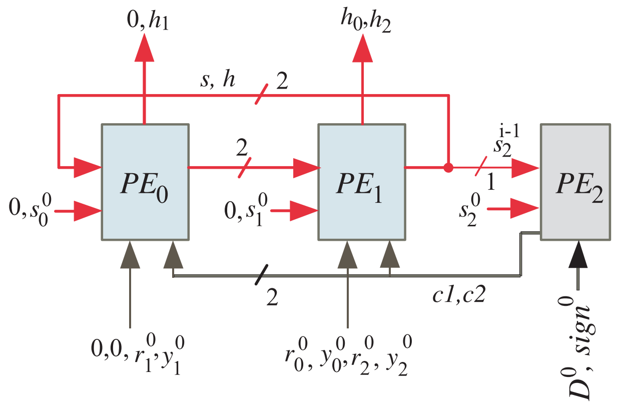

FIGURE 3. Proposed scalable systolic array when $m = 3$ and $T = 2$.

IV. THE PROPOSED SCALABLE DESIGN FOR THE INVERSION ALGORITHM

Using projection matrix $\mathbf{P}_1 = [\,0\ (ph+\mu) \bmod T\,]$, A point $\mathbf{p} = [i\quad j]^t \in \mathbb{D}$ will be mapped onto the point:

$$\overline{\mathbf{p}} = \mathbf{P}_1\mathbf{p} - \delta = (j+\mu) \bmod T \tag{11}$$

where

$$\mu = T\lceil m/T \rceil - m, \quad \text{and } \delta = 0 \tag{12}$$

The scalable systolic array design resulted from this mapping, when $m = 3$ and $T = 2$, is shown in Figure 3. The number of PE's is $T+1$. Thus, the required number of PEs depends

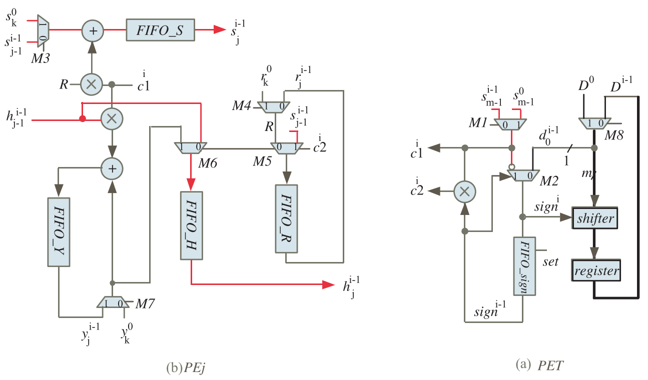

FIGURE 4. PEs details. (a) PE_T details. (b) PE_j details when $0 \le j \le T-1$. Boxes labeled *FIFOs* are $(m'/T)+1$-bit flip-flops with load and clear/set control inputs. Box labeled *register* is m-bit flip-flops with load and clear control inputs.

on the value of T and there is not any dependency on the field size m. Figure 4 (a) shows the details for the control processing element PE_T. Fig. 4 (b) shows the details for PE_j.

Each PE processes $\lceil m/T \rceil = m'/T$ bits, where m' is given in Eq. (7), and works on one bit at each clock cycle. The processing elements PE_j, $0 \le j \le T-2$, store (m'/T) bits for S, R, Y, and H as shown by the four sets of FIFO buffers in Fig. 4 (b). Also, the processing element PE_T will need to store (m'/T) bits for $sign^0$ as shown by the FIFO buffer in Fig. 4 (a). The processing element PE_{T-1} will need to store only $(m'/T)-1$ bits for S, R, Y, and H.

The operation of each PE_j $(0 \le j < T)$ for the proposed scalable design can be summarized as follows:

1) For the first $(m'/T)+1$ time steps (i.e. $1 \le t \le (m'/T)+1$), all the D-FFs of the FIFO_*sign* will set to have the initial value of $sign^0$ equal to 1 through all these time steps. Also, through these time steps, MUX8 is set to accept input D^0. The register at the output of shifter is loaded every $(m'/T)+1$ time steps.
2) For the first $(m'/T)+1$ time steps (i.e. $1 \le t \le (m'/T)+1$), MUXs M_1, M_3, M_4, and M_7 are set to accept the inputs of s_{m-1}^0, s_k^0, r_k^0, and y_k^0 corresponding to the polynomials S, R and Y, respectively. Through these time steps, the flip-flops of FIFO_H is cleared to to have the input bits of $h_k'^0$ equal to zero. PE_j will accept bits s_k^0, r_k^0, and y_k^0 at time t such that:

$$j = k \bmod T \quad 0 \le k < m' \tag{13}$$

$$t = m'/T - \lfloor k/T \rfloor + 1 \tag{14}$$

These bits will be loaded in FIFO_S, FIFO_R, and FIFO_Y, respectively.
3) For times $t > (m'/T)+1$, MUXs M_1, M_3, M_4, M_6, M_7 and M_8 are set to accept the inputs of s_{m-1}^{i-1}, s_{j-1}^{i-1}, r_j^{i-1}, h_{j-1}^{i-1}, y_j^{i-1} and D^{i-1}, respectively.
4) Control bits $c1^i$ and $c2^i$ are broadcast to all PEs at iteration i where:

$$i = \left\lfloor \frac{t}{(m'/T)+2} \right\rfloor + 1 \tag{15}$$

TABLE 1. Comparison between different digit-serial finite field inverters.

Design	NOT	AND	XOR	MUXs	Flip-Flops	Latency	Critical Path delay
Yan [12]	$2m$	$2m(2d+2)$	$2m(2d-2)$	$2m(3d+1)$	F_1	L_1	τ_1
Fan [18]	0	$3d(m+1)$	$d(3m+2)$	$d(3m+4)$	$6(m+1)$	$\lceil \frac{2m-1}{d} \rceil$	τ_2
Proposed Design	1	$2T+1$	$2T$	$2m+5T+2$	F_2	L_2	τ_3

TABLE 2. Synthesis results of different digit-serial inverters for $m = 233$, $T = 64$, $d = 4$ and $S = 1$.

Inverter	Latency	Area [Kgates]	Maximum Frequency [GHz]	delay (D) [μs]	Power [μW]	energy [PJ]	Throughput (m_{bits}/D) [Mbps]
Yan [12]	409	15.30	2.45	0.167	16.7	2.8	1395.2
Fan [18]	117	10.54	1.10	0.106	14.9	1.6	2198.1
Proposed Design	2325	1.79	3.75	0.620	0.64	0.4	375.8

5) The output product H is available at time t, which satisfies the inequalities:

$$(2m-1)(m'/T+1)-(m'/T) \leq t < (2m-1)(m'/T+1) \quad (16)$$

We can conclude that the scalable design is suited to resource-constrained embedded applications due to the following reasons:

1) Ability to control the number of PEs in the systolic array.
2) Inter-processor communication is limited to one-bit data only.

V. COMPLEXITY COMPARISON

From Fig. 3, we can estimate the time and area complexities of the proposed scalable design. Table 1 compares the area , latency, and critical path delay of the proposed scalable digit-serial design to the closest digit-serial competitor designs in the literature [12], [18].

In Table 1 we have:

1) T_A is AND gate delay
2) T_{MUX} is MUX delay
3) T_X is XOR gate delay
4) $F_1 = 30m + (6d + 2)(\frac{2mS}{d})$, where S is the pipelined stages inserted in each PE and d is the digit size.
5) $F_2 = (4T + 1)(\lceil \frac{m}{T} \rceil + 1)$
6) $L_1 = \lceil \frac{2m-2}{d} \rceil (S + 2) + \lceil \frac{m}{d} \rceil - 1$
7) $L_2 = (2m - 1)(\lceil \frac{m}{T} \rceil + 1)$
8) $\tau_1 = \frac{d}{S+1}(T_A + T_X)$
9) $\tau_2 = 2dT_{MUX}$
10) $\tau_3 = 2T_{MUX}$

In order to verify the area and performance (delay and power) of the proposed scalable design, we used Synopsys synthesis tools package version 2005.09-SP2 for logic synthesis and power analysis of the proposed design as well as the most efficient digit-serial designs of [12] and [18]. The designs are first described using VHDL and then synthesized to obtain the gate level for field size of $m = 233$, digit size $d = 4$, $S = 1$, and $T = 64$ using (45 nm, 1.1 V) standard-cell CMOS technology. Table 2 shows the obtained synthesis results (area, delay, power) of the different digit-serial inverters. Also, it shows the calculated energy as well as the throughput rate that are used to measure the degree of the improvement achieved in each digit-serial inverter design. The power was estimated at a low frequency of 100kHz which is suitable for ultra-low power devices like RFID.

From this table, we notice that the proposed scalable design has a significant reduction in the area (ranging from 83.0% to 88.3%) and energy (ranging from 75.0% to 85.7%) over the compared efficient designs that make it very suitable for constrained implementations of cryptographic primitives in ultra-low power devices that have tight restrictions on area and power consumption. On the other hand, the proposed design has significantly lower throughput values compared to all other designs.

VI. SUMMARY AND CONCLUSION

This paper presented a new scalable systolic array structure to perform inversion operation in $GF(2^m)$ based on a previously modified extended Euclidean algorithm. This structure is suitable for fixed size processor that only reuse the core and does not require to modulate the core size when m is modified. This structure is extracted by applying a nonlinear methodology that gives the designer more flexibility to control the processing element work load and also reduces the overhead of communication between processing elements. Implementation results of the proposed scalable digit-serial design and the previously reported efficient digit-serial designs shows that the proposed scalable structure achieves a significant reduction in the area and power that makes it more suitable for constrained implementations of cryptographic primitives in ultra-low power devices such as wireless sensor nodes and radio frequency identification (RFID) devices.

REFERENCES

[1] J. Kaps, "Cryptography for ultra-low power devices," Ph.D. dissertation, Dept. ECE, Worcester Polytechnic Inst., Worcester, MA, USA, 2006.

[2] G. M. de Dormale, R. Ambroise, D. Bol, J.-J. Quisquater, and J.-D. Legat, "Low-cost elliptic curve digital signature coprocessor for smart cards," in *Proc. IEEE 17th Int. Conf. Appl.-Specific Syst., Archit. Process.*, Sep. 2006, pp. 347–353.

[3] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, and I. Verbauwhede, "Public-key cryptography on the top of a needle," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1831–1834.

[4] M. Feldhofer and J. Wolkerstorfer, "Strong crypto for RFID tags—A comparison of low-power hardware implementations," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1839–1842.

[5] F. Furbass and J. Wolkerstorfer, "ECC processor with low die size for RFID applications," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2007, pp. 1835–1838.

[6] J.-H. Guo and C. L. Wang, "Hardware-efficient systolic architecture for inversion and division in GF(2^m)," *IEE Proc. Comput. Digit. Techn.*, vol. 145, no. 4, pp. 272–278, Jul. 1998.

[7] Z. Yan and D. V. Sarwate, "New systolic architectures for inversion and division in GF(2^m)," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1514–1519, Nov. 2003.

[8] Z. Yan, D. V. Sarwate, and Z. Liu, "Hardware-efficient systolic architectures for inversion in GF(2^m)," *IEE Proc. Inf. Security*, vol. 152, no. 1, pp. 31–46, Oct. 2005.

[9] 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.*, vol. 54, no. 3, pp. 370–380, Mar. 2005.

[10] C.-H. Wu, C.-M. Wu, M.-D. Shieh, and Y.-T. Hwang, "An area-efficient systolic division circuit over GF(2^m) for secure communication," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2002, pp. 733–736.

[11] P. Corbett and R. Hartley, "Designing systolic arrays using digit-serial arithmetic," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 1, pp. 62–65, Jan. 1992.

[12] Z. Yan, "Digit-serial systolic architectures for inversions over GF(2^m)," in *Proc. IEEE Workshop Signal Process. Syst. Design Implement. (SIPS)*, Oct. 2006, pp. 77–82.

[13] F. Gebali, *Algorithms and Parallel Computers*. New York, NY, USA: Wiley, 2011.

[14] A. Ibrahim, F. Gebali, and T. Al-Somani, "Systolic array architectures for SunarŰKoç optimal normal basis type II multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2090–2102, Oct. 2015.

[15] Z. Yan, D. V. Sarwate, and Z. Liu, "High-speed systolic architectures for finite field inversion," *Integr., VLSI J.*, vol. 38, no. 3, pp. 383–389, Jan. 2005.

[16] 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.*, vol. 82, no. 3, pp. 331–343, Mar. 2015.

[17] F. Gebali and A. Ibrahim, "Efficient scalable serial multiplier over GF(2^m) based on trinomial," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 10, pp. 2322–2326, Oct. 2015.

[18] J. Fan, L. Batina, and I. Verbauwhede, "Design and design methods for unified multiplier and inverter and its application for HECC," *Integr., VLSI J.*, vol. 44, no. 4, pp. 280–289, Sep. 2011.

ATEF IBRAHIM (M'16) received the B.Sc. degree in electronics from Mansoura University, the M.Sc. and Ph.D. degree in electronics and electrical communications from Cairo University, and the. Ph.D. degree in electronics and electrical eommunications from the University of Victoria, Canada. Dr. Ibrahim is an Associate Professor of Computer Engineering at Prince Sattam Bin Abdulaziz University, KSA and an Associate Professor at Electronics Research Institute, Cairo, Egypt. Also, he is adjunct professor at University of Victoria. His research interests include computer arithmetic, cryptography, bio-computing, embedded systems design, and digital VLSI design.

TURKI F. AL-SOMANI (SM'11) received the B.Sc. and M.Sc. degrees in electrical and computer engineering from King Abdulaziz University, Saudi Arabia, and the Ph.D. degree in computer science and engineering from the King Fahd University of Petroleum and Minerals. He is currently an Associate Professor with the Department of Computer Engineering, Umm Al-Qura University, Saudi Arabia. His research interests include information theory, computer arithmetic, and cryptosystems.

FAYEZ GEBALI (SM'76) received the B.Sc. degree (Hons.) in electrical engineering from Cairo

University, the B.Sc. degree (Hons.) in mathematics from Ain Shams University, and the Ph.D. degree in electrical engineering from The University of British Columbia. He is a Professor of Computer Engineering in the Electrical and Computer Engineering Department, University of Victoria. His research interests include parallel algorithms, 3D IC design, hardware verification and security, and wireless communications.