Low-Power and Low-Hardware Bit-Parallel Polynomial Basis Systolic Multiplier over GF(2^m) for Irreducible Polynomials

Sudha Ellison Mathe and Lakshmi Boppana

Multiplication in finite fields is used in many applications, especially in cryptography. It is a basic and the most computationally intensive operation from among all such operations. Several systolic multipliers are proposed in the literature that offer low hardware complexity or high speed. In this paper, a bit-parallel polynomial basis systolic multiplier for generic irreducible polynomials is proposed based on a modified interleaved multiplication method. The hardware complexity and delay of the proposed multiplier are estimated, and a comparison with the corresponding multipliers available in the literature is presented. Of the corresponding multipliers, the proposed multiplier achieves a reduction in the hardware complexity of up to 20% when compared to the best multiplier for m = 163. The synthesis results of application-specific integrated circuit and field-programmable gate array implementations of the proposed multiplier are also presented. From the synthesis results, it is inferred that the proposed multiplier achieves low power consumption and low area complexitywhen compared to the best of the corresponding multipliers.

Keywords: Finite field, Cryptography, Systolic, Polynomial basis, Application-specific integrated circuit, Field programmable gate arrays.

1. Introduction

Cryptography is the art of efficiently hiding data and transmitting it over any insecure channel to prevent unauthorized access. Modern cryptography mainly depends on the foundation principles of mathematical theory and computer science. It mainly deals with the development of cryptographic algorithms that are designed around assumptions regarding computational hardness. Cryptography can be broadly categorized as symmetric-key cryptography and asymmetric-key cryptography [1]. Symmetric-key cryptography performs the encryption and decryption processes using the same secret key. The techniques developed using this principle are the data encryption standard [2] and the advanced encryption standard (AES) [3]. Asymmetric-key cryptography performs the encryption and decryption processes using different keys. Elliptic curve cryptography (ECC) [4], [5] and Rivest and others [6] are some techniques that have been developed using this principle.

Many cryptographic algorithms utilize the multiplication operation in finite fields. Basic operations in a finite field, such as addition and subtraction, are realized using the logical exclusive-OR (XOR), whereas complex operations such as division, exponentiation, and inversion are realized using recursive multiplications [7]. Therefore, the basic unit for all of these arithmetic operations is the multiplication operation. Multiplication in a finite field can be performed over three basis representations, namely Normal Basis (NB), Polynomial Basis (PB), and Dual

Manuscript received Oct. 28, 2016; revised Mar. 11, 2017; accepted Mar. 28, 2017.

Sudha Ellison Mathe (corresponding author, ellison@nitw.ac.in) and Lakshmi Boppana (lakshmi@nitw.ac.in) are with the Department of Electronics and Communication Engineering, National Institute of Technology-Warangal, India.

Basis (DB) [8], each of which has its distinct advantages. Hardware implementations of NB multipliers typically consume less power compared to other bases, and they are mainly used to realize squaring and exponentiation operations. Multiplications in PB are less complex, but the hardware implementations consume more power compared to NB multipliers. DB multipliers require a low area compared to the other two bases. However, PB multipliers are extensively used because they are less complex and can be matched to any system, whereas NB and DB multipliers require additional hardware for basis conversion.

The finite field GF(2^m) is characterized by an irreducible polynomial. An irreducible polynomial (T) is said to be irreducible over GF(2^m) if T cannot be factored into two product polynomials, where the product polynomials should belong to the above said field. Various types of irreducible polynomials characterize finite-field multipliers. These are general/generic polynomials, trinomials, pentanomials, all-one polynomials (AOP), and equally spaced polynomials (ESP), which are recommended for cryptographic applications. ESPs are polynomials whose terms are equally spaced (in degree) by r. Trinomials, pentanomials, and AOPs are special cases of ESPs, and they are distinguished based on the spacing of the terms. Generic polynomials may have a random number of terms, with random spacing between the terms. The multipliers based on trinomials and pentanomials have relatively lower hardware complexity when compared to other classes of irreducible polynomials, and this is due to a smaller number of terms, and consequently, a lower computational complexity. On the other hand, these multipliers cannot be utilized in all applications owing to the limitation of a fixed number of terms. Multipliers that are based on AOPs require additional hardware complexity compared to multipliers that are based on trinomials or pentanomials, and this is because of a larger number of terms, and consequently, a higher computational complexity. On the contrary, the ease of representation of AOP facilitates efficient implementations and simpler structures in hardware. ESPs are well structured and have higher hardware complexity compared to AOP multipliers. The hardware complexity of multipliers that are based on generic polynomials cannot be determined beforehand owing to the randomness in the number of terms of the polynomials. Therefore, multipliers for generic polynomials are highly generic and operate on any type of polynomial.

Various types of architectures have been proposed in the literature for PB multiplication, such as bit-serial [9], bit-parallel [10], digit-serial [11], systolic [12], semi-systolic [13], sequential [14], and pipelined [15]. Each type of architecture has its advantages and shortcomings. Systolic architectures require several replicas of the same structure to perform fast computations with high hardware overhead, thereby consuming more power. Hence, low-hardware and low- power systolic multipliers for finite fields are necessary to implement the encryption methods efficiently in portable power-constrained devices. Several systolic multipliers are proposed in the literature for PB multiplication for irreducible polynomials over GF(2^m). Yeh and others [16] proposed a systolic PB multiplier in 1984. An algorithm was derived and a systolic multiplier was realized to perform product-sum computations using a parallel-in/parallel-out two-dimensional (2D) structure. Wang and Lin [17] reconsidered an existing algorithm and proposed a parallel-in/parallel-out systolic array in 1991. When compared to [16], this multiplier provided better cascadability, fault tolerance, and the potential for wafer-scale integration, while providing comparable hardware complexity, throughput, and latency. In 1995, Wu and Chang [18] proposed a multiplication algorithm and realized a systolic multiplier from it. When compared with [16], the multiplier achieved low delay, high hardware complexity, and allowed well-ordered inputs into the system, while the latter did not. Jain and Parhi [19] developed a pipelined parallel-in/parallel-out structure in 1995. Their multiplier offered low hardware complexity and low delay when compared to [16] and [17]. Jain and others [20] introduced a pipelined semi-systolic structure using the least significant bit (LSB)-first algorithm in 1998. This multiplier achieved low hardware complexity, low delay, and also achieved substructure sharing among multiplier operations. Koc and Acar [21] proposed a Montgomery multiplication algorithm in 1998. Kwon and others [22] presented a bit-parallel systolic structure using an LSB-first algorithm in 2003. When compared to [16] and [17], this multiplier had the same hardware complexity and delay, but achieved a better critical path delay and had unidirectional data flow. Lee and others [23] proposed a multiplexer-based bit-parallel multiplier that was based on the modified Booth's algorithm in 2006. When compared to [17], this multiplier achieved low hardware complexity and low latency. In 2006, Kwon and others [24] proposed a systolic multiplier using an LSB-first algorithm. When compared with [16] and [17], this multiplier had the same hardware complexity and the same delay, but had a lower critical path and had unidirectional data flows.

In 2006, Chiou and others [25] proposed a time-independent Montgomery multiplication algorithm and realized a systolic array Montgomery multiplier. The hardware complexity and delay of the multiplication

algorithm in [21] was analyzed and compared with the multiplier in [25]. It was observed that the multiplier in [25] achieved low hardware complexity and low delay when compared to [21]. In 2006, Lee and others [26] presented two algorithms using an interleaved multiplication and a folded technique. Two bit-parallel systolic multipliers were realized from these algorithms, where the two multipliers achieved a lower hardware complexity than [16] and [17]. Moreover, both of the multipliers achieved a low delay when compared to [17] and same delay when compared to [16]. A 2D systolic array multiplier using a linear feedback shift register was proposed by Chiou and others [27] in 2007. When compared to [16] and [17], this multiplier achieved low hardware complexity and low delay. Lee [28] presented a time-dependent and time-independent algorithm in 2008, and two bit-parallel systolic multipliers were realized by employing these algorithms. The two multipliers achieved low hardware complexity when compared to [16] and [17]. The multiplier in [28]*a* achieved the same delay as [16], and achieved low delay when compared to [17]. The multiplier in [28]*b* achieved high and low delay when compared to [16] and [17], respectively. In 2008, Lee [29] proposed a multiplexer-based systolic multiplier from a multiplication algorithm that used the cut-set systolization method and modified Booth's recoding. This multiplier achieved a comparable hardware complexity when compared to [16] and [17]. It also achieved high and low delays when compared to [16] and [17], respectively. Fournaris and Koufopavlou [30] proposed a Montgomery multiplication element from an optimized Montgomery multiplication algorithm in 2008. When compared to [17] and [20], this multiplier achieved low hardware complexity. However, it achieved low and high delays when compared to [17] and [20], respectively. In 2009, Kwon and others [31] proposed a systolic multiplier from an LSB-first algorithm. This multiplier achieved high hardware complexity when compared to [16], [17], and [31]*b*, but achieved low delay when compared to [17] and [31]*b*. In 2014, Kim and Jeon [32] proposed a cellular semi-systolic Montgomery structure from an optimized algorithm to achieve low time complexity. In 2016, a systolic multiplier was proposed by Mathe and Boppana [33] over GF(2^8), and achieved better hardware complexity and power consumption when compared to other GF(2^8) multipliers.

In this paper, to reduce hardware complexity, a modified interleaving multiplication method is proposed based on a conventional interleaving multiplication method. The proposed method performs multiplication with interleaved modular reduction of two arbitrary elements for a generic, field-defining, irreducible polynomial. Subsequently, a scalable bit-parallel systolic polynomial basis multiplier over GF(2^m) is realized based on the multiplier proposed in [33]. The proposed multiplier achieves low hardware complexity and low power consumption when compared to corresponding multipliers available in the literature. The multiplier achieves a latency of m clock cycles, which is comparable to corresponding multipliers.

The organization of this paper is as follows: the algorithm formulation is presented in Section II, and the proposed bit-parallel polynomial basis systolic multiplier is explained in Section III. In Section IV, the results and discussion are presented, while the conclusions are made in Section V.

II. Algorithm Formulation

The conventional polynomial basis (PB) representation and multiplication of the field elements over GF(2^m) are presented in this section [34].

Definition 1: Let $T(x)$ be the irreducible polynomial of degree m over the field GF(2^m). Then,

$$T(x) = x^m + t_{m-1}x_{m-1} + \cdots + t_1 x + t_0, \tag{1}$$

where $t_0, t_1, \ldots, t_{m-1} \in$ GF(2).

Definition 2: Let $\gamma \in$ GF(2^m) be a root of $T(x)$. Then, the following set constitutes the polynomial basis in GF(2^m)

$$\boldsymbol{\Omega} = \{1, \gamma, \gamma^2, \ldots, \gamma^{m-1}\}. \tag{2}$$

Definition 3: In Ω, the elements of GF(2^m) are polynomials with a degree of at most $m - 1$. Then, the set of all polynomials in GF(2^m) is

$$\mathrm{GF}(2^m) = \{g(x) \mid g(x) = g_{m-1}x_{m-1} + \cdots + g_1 x + g_0\}, \tag{3}$$

where $g_i \in$ GF(2); for $i = 0, 1, 2, \ldots, m - 1$.

Definition 4: Let $g(x) = g_{m-1}.x_{m-1} + \cdots + g_1.x + g_0$ be a polynomial in GF(2^m); then, the binary representation of this polynomial is

$$g = (g_{m-1}, \ldots, g_1, g_0), \tag{4}$$

where $g_i \in$ GF(2), and g_{m-1} is the most significant bit, while g_0 is the LSB; for $i = 0, 1, 2, \ldots, m - 1$. Let the polynomials $A(x)$ and $B(x)$ be the two field elements, $T(x)$ be the field-defining irreducible polynomial, and $C(x)$ be the final product polynomial. Then,

$$C(x) = (A(x) \times B(x)) \bmod T(x). \tag{5}$$

The product of $A(x)$ and $B(x)$, each with degree of at most $m - 1$, results in an intermediate polynomial given by

https://doi.org/10.4218/etrij.17.0116.0770

$$\begin{aligned} D(x) &= A(x) \times B(x) \\ &= a_0 + a_1 x + \cdots + a_{m-1} x^{m-1} \times b_0 + b_1 x + \cdots + b_{m-1} x^{m-1} \\ &= d_0 + d_1 x + \cdots + d_{2m-2} x^{2m-2}. \end{aligned} \tag{6}$$

The polynomial $D(x)$ with a degree of at most $2m - 2$ is modular reduced by a degree m irreducible polynomial $T(x)$ to obtain $C(x)$ with a degree of at most $m - 1$, which is the final result of the multiplication operation.

$$\begin{aligned} C(x) &= D(x) \bmod T(x) \\ &= d_0 + d_1 x + \cdots + d_{2m-2} x^{2m-2} \\ &\quad \times t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m \\ &= c_0 + c_1 x + \cdots + c_{m-1} x^{m-1}. \end{aligned} \tag{7}$$

Many multiplication methods are available in the literature to perform efficient multiplication operations over the polynomial basis for a finite field GF(2^m). In this study, a conventional interleaving multiplication method [35] is modified in an efficient manner to obtain a modified interleaving multiplication method, which is presented below.

Let the two arbitrary elements $A(x)$ and $B(x)$ in GF(2^m) be expressed as

$$A = \sum_{i=0}^{m-1} a_i x^i \quad B = \sum_{i=0}^{m-1} b_i x^i. \tag{8}$$

Let $D(x) \in$ GF(2^m) be the product polynomial of the two elements $A(x)$ and $B(x)$.

$$\begin{aligned} D(x) &= A(x) \times B(x) \\ &= A(x) \times \sum_{i=0}^{m-1} b_i x^i \\ &= b_0 A(x) + b_1 x A(x) + b_2 x^2 A(x) + \cdots + b_{m-1} x^{m-1} A(x). \end{aligned} \tag{9}$$

Here, $D(x)$ can be reduced to a summation of $A(x)x^i$, as can be observed from (9) if $b_i = 1$, for all $i = 0, 1, \ldots, m - 1$. The XOR operation is considered for the summation of $A(x)x^i$ because the addition is simply an XOR operation in polynomial representation. This summation runs from $b_0 A(x)$ to $b_{m-1} A(x) x^{m-1}$ in m iterations. The modular reduction in the conventional interleaving multiplication method [35] is performed by

$$A(x) = A(x) \times x^i \bmod T(x). \tag{10}$$

Equation (10) is evaluated for each i as follows

For $i = 0$:

$$\begin{aligned} A(x) &= A(x) \bmod T(x) \\ &= (a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}) \\ &\quad \bmod (t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m). \end{aligned} \tag{11}$$

A degree m polynomial cannot modulo divide a degree $m - 1$ polynomial. Hence, this step can be skipped. For $i = 1$:

$$\begin{aligned} A(x) &= (A(x) \times x) \bmod T(x) \\ &= (a_0 x + a_1 x^2 + \cdots + a_{m-1} x^m) \\ &\quad \bmod (t_0 + t_1 x + \cdots + t_{m-1} x^{m-1} + x^m) \\ &= a_{m-1} t_0 + (a_{m-1} t_1 + a_0) x + (a_{m-1} t_2 + a_1) x_2 + \cdots \\ &\quad + (a_{m-1} t_{m-1} + a_{m-2}) x_{m-1}. \end{aligned} \tag{12}$$

From (12), it can be observed that if $a_{m-1} = 1$, the modular reduction step is reduced to the summation of the irreducible polynomial, (t_{m-1}, ... , t_1, t_0), and the left-shifted value of a, (a_{m-2}, ... , a_1, a_0, 0), as depicted in the following equation

$$A(x) = t_0 + (t_1 + a_0) x + (t_2 + a_1) x^2 + \cdots + (t_{m-1} + a_{m-2}) x^{m-1}. \tag{13}$$

If $a_{m-1} = 0$, the result of the modular reduction step is simply the shifted value of a, that is, (a_{m-2}, ... , a_1, a_0, 0), as depicted in the following equation

$$A(x) = a_0 x + a_1 x^2 + \cdots + a_{m-2} x^{m-1}. \tag{14}$$

The above interpretations hold true in a similar manner for all $i = 2, \ldots, m - 1$. For each i, a is left-shifted. If $a_{m-1} = 0$, a is forwarded to the output. If $a_{m-1} = 1$, a is XORed with t and forwarded to the output. The summation is computed using the XOR operation, as discussed earlier. Therefore, the multiplication operation in a finite field can be performed in an interleaved manner using (9) and (12).

III. Proposed Bit-Parallel Polynomial Basis Systolic Multiplier

In this section, a bit-parallel systolic multiplier for irreducible polynomials over GF(2^m) is proposed based on the modified interleaving multiplication method presented in the previous section.

The operations in (9) and (12) can be performed recursively over m iterations to obtain the multiplication operation over GF(2^m). A signal flow graph (SFG) is realized from (9) and (12), as shown in Fig. 1(a). The SFG

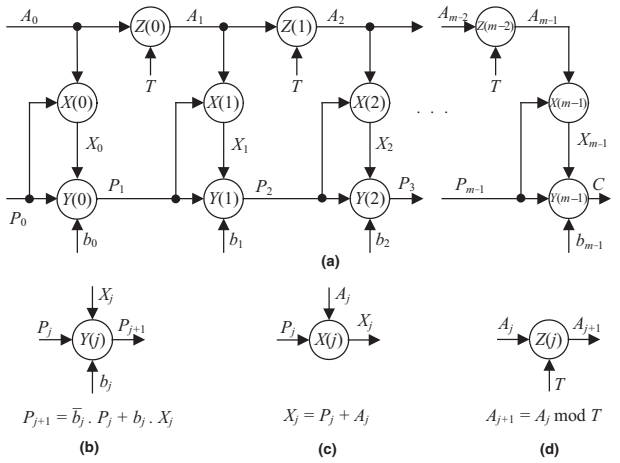

Fig. 1. SFG derived from the proposed multiplication method: (a) SFG, (b) logic of the $Y(j)$ node, (c) logic of the $X(j)$ node, and (d) logic of the $Z(j)$ node.

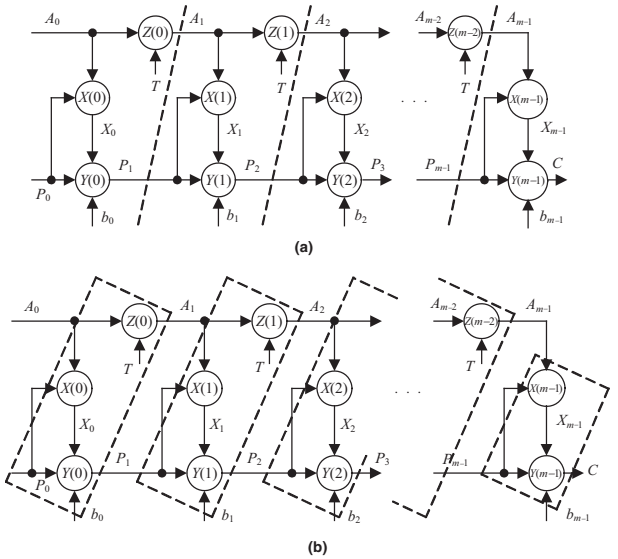

Fig. 2. Cut-set retiming of the SFG: (a) proposed cut-set retiming and (b) realization of PE.

consists of m addition nodes $X(j)$, m decision nodes $Y(j)$, and $m - 1$ modular reduction nodes $Z(j)$. The logic of these nodes is shown in Figs. 1(b)–(d). Here, A_0 is the binary representation of $A(x)$, A_{j+1} is the result of the modular reduction of A_j for the jth iteration, P_{j+1} is the result of the decision node for the jth iteration, X_j is the result of the addition node $X(j)$ for the jth iteration, T is the binary representation of the irreducible polynomial $T(x)$, b_i is the ith coefficient of $B(x)$, and C is the binary representation of the final product $C(x)$. Node $X(j)$ performs a bit-addition operation according to (9) and (12). It performs addition on the partial results P_j and A_j using the XOR operation. Node $Y(j)$ performs the decision (or selection) operation according to (9) and (12). It selects between the partial results P_j and X_j using the selection input b_i. Node $Z(j)$ performs the modular reduction of the degree by one according to (12). It reduces the degree of A_j by one, and A_{j+1} gives the result. Here, i denotes the index of the coefficient of the polynomial under consideration, and j denotes the iteration count.

Figure 2(a) shows the proposed cut-set retiming of the SFG, which is performed to obtain a pipelined structure with a reduced critical path. The proposed cut-set retiming allows one addition node, one decision node, and one modular reduction node in each iteration of the cut-set, thus enabling a reduced critical path. It also eliminates the data dependency between the addition node and the modular reduction node by performing them in a single iteration. The critical path after the proposed cut-set retiming amounts to $\max\{T_{XN}, T_{YN}, T_{ZN}\}$, where T_{XN}, T_{YN}, and T_{ZN} are the computation times of the addition node, decision node, and modular reduction node, respectively. Other variations of the cut-set retiming have been considered, and the cut-set retiming proposed in Fig. 2(a) is found to provide a good trade-off between the critical path and the latency. Further, the achieved trade-off is found to be comparable to, or better than similar structures that are reported in the literature. Each iteration of the proposed cut-set is wrapped into a single entity called a processing element (PE).

Figure 2(b) shows the grouping of the nodes of the retimed SFG into PEs based on the proposed cut-set. It can be observed that the PEs obtained from such a grouping of the nodes enables the realization of a regular and modular design consisting of one addition node, one decision node, and one modular reduction node in each PE. The addition node is realized using one XOR operation, the decision node is realized using one 2:1 multiplexing (MUX) operation, and the modular reduction node is realized using one XOR operation and one 2:1 MUX operation. Figure 3(a) shows the systolic design consisting of m PEs realized from the proposed cut-set retimed SFG given in Fig. 2(b). The $m - 1$ regular PEs (that is, PE[0] to PE[$m - 2$]) perform the addition, decision, and modular reduction operations concurrently, whereas the mth PE (that is PE[$m - 1$]) performs only the addition and decision operations concurrently, which is in accordance with the proposed cut-set retimed SFG. The functions of these two types of PEs are shown in Figs. 3(b)–(c). Each regular PE receives A_j, P_j, b_i, and T as inputs, and computes the new A_{j+1} and P_{j+1} values for the next iteration. The mth PE receives A_{m-1}, P_{m-1}, and b_{m-1} from the $(m - 1)$th regular PE, and produces the final

https://doi.org/10.4218/etrij.17.0116.0770

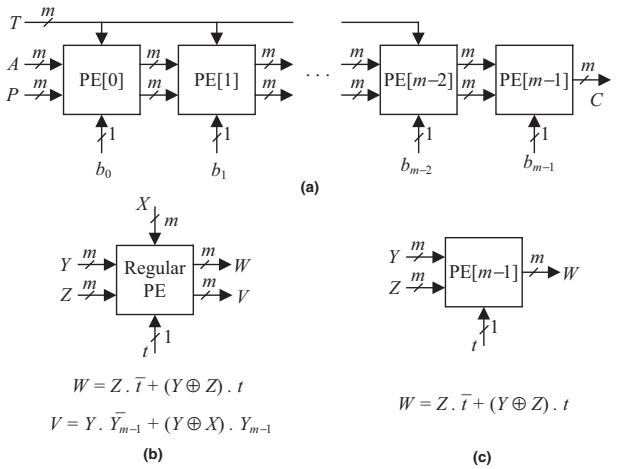

Fig. 3. Proposed systolic multiplier realized from the SFG using PEs: (a) systolic multiplier, (b) logic of the regular PE (PE[0] to PE[m – 2]), and (c) logic of PE[m – 1].

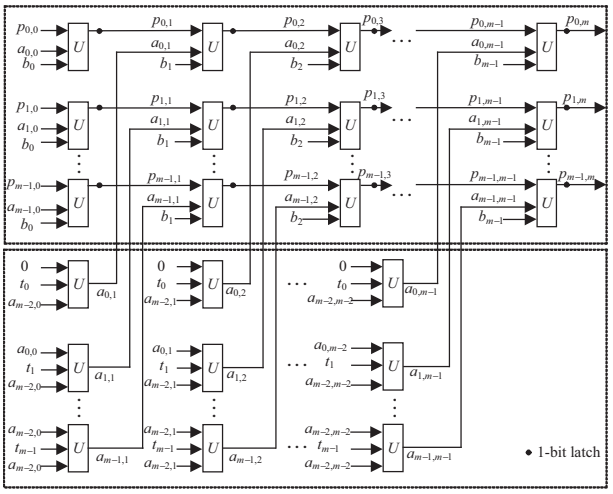

Fig. 4. Proposed systolic multiplier design using U-cells for irreducible polynomials over GF(2m).

result of the finite-field multiplication, C. The regular PE and PE[m – 1] are further decomposed into 2m U-cells and m U-cells, respectively, to derive a regular, scalable structure, and is much simpler for implementation and optimization.

The decomposed systolic structure realized using the U-cells is shown in Fig. 4. The first set of m U-cells (corresponding to the addition and decision nodes) of a regular PE are represented column-wise in the upper block, and the second set of m U-cells (corresponding to the modular reduction nodes) of the same PE are represented column-wise in the lower block. The m U-cells (corresponding to the addition and decision nodes) of the mth PE are represented in the upper block in the rightmost column. The first set of m^2 U-cells perform the polynomial multiplication operation corresponding to (9), and the second set of (m^2 – m) U-cells perform the modular reduction operation corresponding to (12). The inputs to each cell are $p_{i,j}$, $a_{i,j}$, b_i, and $a_{i,j}$, t_i, $a_{m-2,j}$ for the upper and lower blocks, respectively. The values $p_{i,j}$, $a_{i,j}$ and t_i are the ith bit values of the m-bit P_j, A_j and T, respectively, and $a_{m-2,j}$ is the (m – 2)th bit value of A_j. Here, i denotes the index of the coefficient of the polynomial under consideration, and j denotes the iteration count. It may be noted that the $a_{i,j}$ in the modular reduction block is right shifted by one bit according to (12).

The details of the circuit and the function of a U-cell are shown in Fig. 5. Each U-cell consists of one XOR gate and one 2:1 MUX. According to (9), the XOR and MUX in the U-cell for the upper block are derived from the addition node and the decision node, respectively. According to (12), the XOR and MUX in the U-cell for the lower block are derived from the modular reduction node. It can be observed that the pipelining registers applied for the systolic structure in Fig. 4 enable concurrent operations such that the critical path is minimized to $T_X + T_M$, where T_X and T_M are the delays of an XOR gate and a 2:1 MUX, respectively. The gate count of the structure is ($2m^2$ – m) XOR gates, ($2m^2$ – m) MUX gates, and m^2 latches. The multiplier produces the first output with an initial latency of m clock cycles followed by one output for every clock cycle. Hence, the throughput is one output per clock cycle, with an initial latency of m clock cycles.

IV. Results and Discussion

In this section, the analysis of the hardware complexity and delay of the proposed bit-parallel systolic multiplier are presented, and a comparison is made between the corresponding systolic multipliers for irreducible polynomials that are available in the literature. The comparisons of the application specific integrated circuit (ASIC) and field programmable gate array (FPGA) synthesis results of the proposed multiplier with the corresponding multipliers available in the literature are also presented.

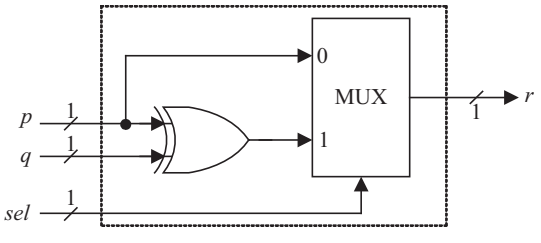

Fig. 5. Logic diagram of the U-cell.

1. Comparison of Hardware Complexity and Delay

As discussed in the previous section, the proposed systolic multiplier requires ($2m^2 - m$) XOR gates, ($2m^2 - m$) MUXs, and m^2 latches. The critical path and latency of the proposed systolic multiplier are obtained as $T_X + T_M$ and m clock cycles, respectively. The proposed structure gives one output in every clock cycle, with an initial latency of m clock cycles.

Table 1 shows a comparison of the hardware complexity, latency, and critical path of the proposed systolic multiplier with corresponding systolic multipliers [16]–[32] that are reported in the literature for generic irreducible polynomials. Here, T_A, T_X, T_M, T_{3X}, T_{4M} denote the delays of a 2-input AND gate, 2-input XOR gate, 2:1 MUX, 3-input XOR gate, and 4:1 MUX, respectively. The National Institute of Standards and Technology has proposed the use of five irreducible polynomials for cryptographic applications, that is, 163, 233, 283, 409, 571. The polynomial $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$ is taken as an example to compare the hardware complexity and delay of various systolic multipliers available in the literature. Traditional CMOS logic is used to estimate the hardware complexity, wherein the transistor counts are six transistors for a 2-input XOR gate, 2-input AND gate, and a 1-bit 2:1 MUX, 16 transistors for a 1-bit 4:1 MUX, and eight transistors for a 1-bit latch. To estimate the delay, real-time circuits from STMicroelectronics are considered, where the typical propagation delays of gates used in the various designs are: 2-input XOR gate (M74HC86) t_{PD} = 12 ns, 2-input AND gate (M74HC08) t_{PD} = 7 ns, 2:1 MUX (M74HC257) t_{PD} = 11 ns, and 4:1 MUX (M74HC153) t_{PD} = 16 ns. A 3-input XOR gate is realized using two 2-input XOR gates. Therefore, the hardware complexity and propagation delay of a 3-input XOR gate are estimated as twelve transistors and t_{PD} = 24 ns, respectively. Table 2 gives the hardware complexity (total transistor count), latency (number of clock cycles), critical path delay (CP), total delay, and percentage reduction in the hardware complexity of all the multipliers considered. Here, the total delay is obtained as the product of the latency and critical path delay. From Table 2, it is observed that the proposed multiplier achieves low hardware complexity

Table 1. Comparison of hardware complexity and delay of the proposed systolic multiplier with corresponding multipliers for irreducible polynomials over GF(2^m).

Design	#AND	#XOR	#MUX	#latches	#clock cycles	Critical path delay
[16]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[17]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_{3X}$
[18]	$2m^2 - m$	$2m^2$	0	$8m^2 - 7m$	$2m - 1$	$T_A + T_X$
[19]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[20]	$2m^2$	$2m^2$	0	$3m^2$	$m + 1$	$T_A + T_X$
[21]	$2m^2$	$2m^2$	0	$4m^2$	$2m$	$T_A + T_X$
[22]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[23]	m	$m^2 + 2m$	$(m^2/2)^a$	$6m^2 + 8m$	$3m/2$	$T_{4M} + T_X$
[24]	$2m^2$	$2m^2$	0	$7m^2$	$3m$	$T_A + T_X$
[25]	$2m^2 + 3m$	$(m^2 + m)^b$	0	$3m^2 + 4m$	$m + 1$	$T_A + T_{3X}$
[26]*a*	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[26]*b*	m^2	m^2	0	$5m^2$	$3m$	$T_A + T_X$
[27]	$2m^2$	$2m^2$	0	$3m^2$	m	$T_A + T_X$
[28]*a*	m^2	$m^2 + 2m$	0	$4m^2 + 3m$	$3m$	$T_A + T_X$
[28]*b*	m^2	m^2	0	$5m^2$	$4m$	$T_A + T_X$
[29]	m	$2m + (m^2/2)^b$	$(m^2 + m/2)^a$	$7m^2$	$3m/2$	$T_{4M} + T_{3X}$
[30]	$m^2 - m + 1$	$m^2 - 1$	$2m^2 + m - 3$	$2m^2 - m$	$2m$	$T_A + T_X$
[31]	$2m^2$	$2m^2$	$2m^2 + m - 3$	$7m^2$	$3m$	$T_A + T_X$
[32]	$2m^2 + 2m$	$2m^2 + 3m$	0	$3m^2 + 4m$	$\lfloor m/2 \rfloor + 1$	$T_A + T_X$
Fig. 4	**0**	$\boldsymbol{2m^2 - m}$	$\boldsymbol{2m^2 - m}$	$\boldsymbol{m^2}$	$\boldsymbol{m}$	$\boldsymbol{T_M + T_X}$

a: 4:1 MUX *b*: 3-input XOR gate

https://doi.org/10.4218/etrij.17.0116.0770

when compared to corresponding systolic structures that are available in the literature. Specifically, it achieves hardware reductions of up to 60%, 60%, 63%, 33%, 33%, 42%, 60%, 48%, 60%, 34%, 27%, 44%, 33%, 27%, 38%, 59%, 20%, 65%, and 33% in hardware for m = 163 when compared to corresponding multipliers [16]–[32], respectively. The latency, critical path, and total delay of the proposed systolic multiplier are comparable to corresponding multipliers.

Figure 6 shows the comparison of the hardware complexity of the proposed systolic multiplier with corresponding systolic multipliers over a range of m (specifically, m = 2 to 600). From Fig. 6, it is observed that the proposed multiplier achieves low hardware complexity compared to the best of the corresponding multipliers, that is, the multiplier in [30], and the difference in their hardware complexities increases as the order of the finite field increases. Hence, the proposed systolic multiplier achieves better performance in terms of hardware complexity for higher-order finite fields, which is suitable for asymmetric-key cryptographic schemes. The proposed multiplier and the multiplier in [30] were modeled in Verilog, and their functionalities were simulated using the Xilinx Vivado 2014.2 verification tool.

Table 2. Comparison of total transistor count, number of clock cycles, critical path, total delay, and % reduction in hardware complexity of the proposed systolic multiplier with corresponding multipliers for m = 163.

Design	#transistors	#clock cycles	CP (ns)	Total delay (ns)	% reduction in hardware complexity
[16]	2,125,520	489	19	9,291	60
[17]	2,125,520	489	31	15,159	60
[18]	2,326,988	325	19	6,175	63
[19]	1,275,312	164	19	3,116	33
[20]	1,275,312	164	19	3,116	33
[21]	1,487,864	326	19	6,194	42
[22]	2,125,520	489	19	9,291	60
[23]	1,660,644	245	28	6,860	48
[24]	2,125,520	489	19	9,291	60
[25]	1,285,418	164	31	5,084	34
[26]*a*	1,175,882	489	19	9,291	27
[26]*b*	1,541,002	489	19	9,291	44
[27]	1,275,312	163	19	3,097	33
[28]*a*	1,175,882	489	19	9,291	27
[28]*b*	1,382,566	652	19	12,388	38
[29]	2,076,620	245	40	9,800	59
[30]	1,061,438	326	19	6,194	20
[31]	2,445,308	489	19	9,291	65
[32]	1,284,114	82	19	1,558	33
Fig. 4	**848,252**	**163**	**23**	**3,749**	**N/A**

Figures 7(a) and (b) gives the simulation results of the proposed systolic multiplier and the multiplier in [30], respectively, by considering various random inputs over GF(2^8). The first set of inputs are $\{83\}_{hex}$ and $\{57\}_{hex}$, the irreducible polynomial is $\{1B\}_{hex}$, and the multiplication result obtained is $\{C1\}_{hex}$. This result corresponds to the result obtained in Section IV.2 of [3], thereby verifying the functionality of the proposed multiplier.

2. ASIC Implementation Results

The proposed systolic multiplier and the systolic multipliers in [30] and [28]*a* were modeled in Verilog for m = 8 and m = 163. These field values were selected because m = 8 is used in the AES algorithm, and m = 163 is recommended by NIST for use in ECC techniques. The models were synthesized at their respective maximum possible frequencies of operation using Synopsys Design Vision Compiler and Synopsys 90 nm Generic Library. The area cost, total delay, and power consumption are tabulated in Table 3. Here, the total delay is obtained as the product of the latency and clock period of the respective designs.

It is observed from the table that when compared to the multiplier in [30], the proposed systolic multiplier requires up to 62% and 43% less power for m = 8 and m = 163, respectively, and it requires up to 37% and 35% less area for m = 8 and m = 163, respectively. Compared to the multiplier in [28]*a*, the proposed multiplier requires up to 65% and 45% less power for m = 8 and m = 163, respectively, and requires up to 63% and 61% less area for m = 8 and m = 163, respectively. It is also observed that the proposed systolic multiplier achieves lower delay than the multipliers in [30] and [28]*a*. The proposed multiplier also achieves a better area-delay product (ADP) and power-delay product (PDP) than [30] and [28]*a*. When compared to [30], the proposed multiplier achieves up to 83% and 76% less PDP for m = 8 and m =163, respectively, and achieves up to 73% and 73% less ADP for m = 8 and m = 163, respectively. Compared to [28]*a*, it achieves up to 91% and 88% less PDP for m = 8 and m =163, respectively, and achieves up to 91% and 91% less ADP for m = 8 and m = 163, respectively.

3. FPGA Implementation Results

The proposed systolic multiplier and the systolic multipliers in [30] and [28]*a* were modeled in Verilog.

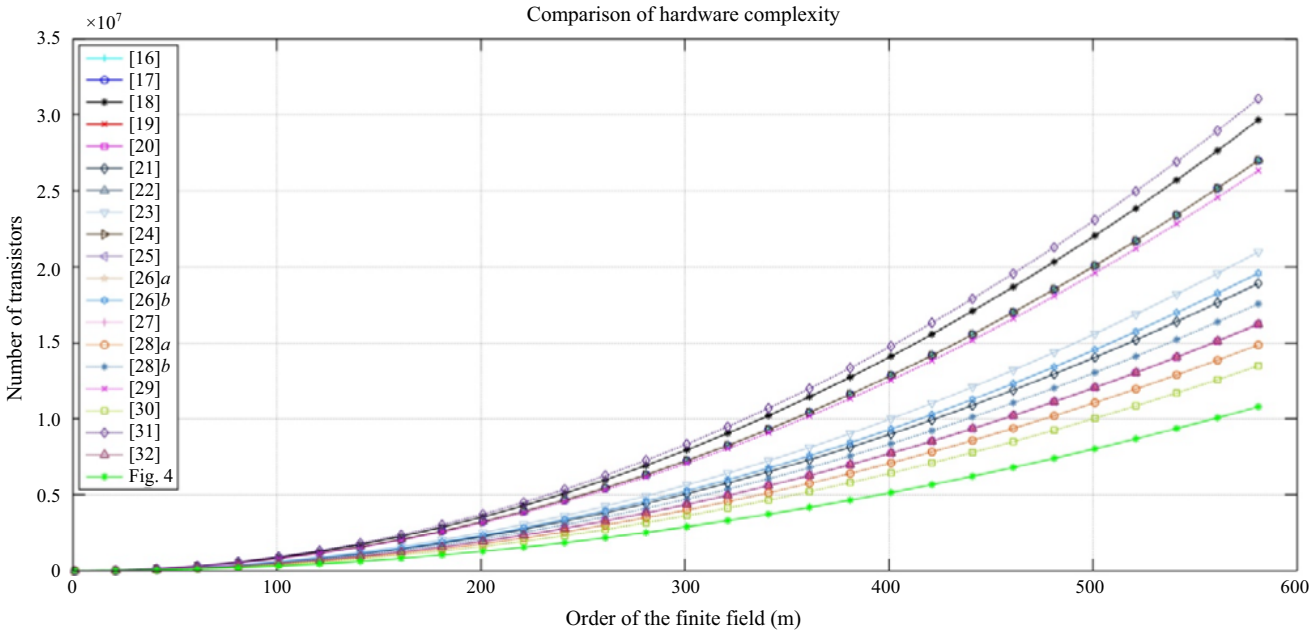

Fig. 6. Comparison of hardware complexity (in terms of transistor count) of the proposed systolic multiplier with corresponding multipliers over a range of m for irreducible polynomials over GF(2^m).

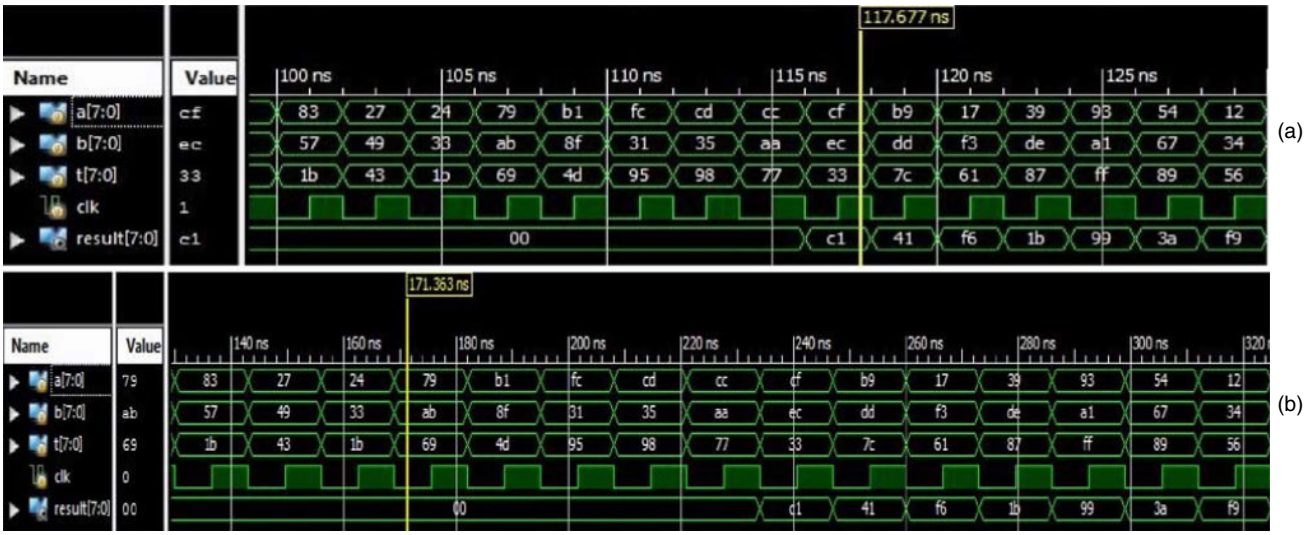

Fig. 7. Simulation results for various inputs: (a) proposed systolic multiplier and (b) multiplier design in [30].

The simulation and synthesis of these models were carried out at their respective maximum possible frequencies of operation using Xilinx Vivado 2014.2 to verify the functionality of the multipliers. The implementation of the synthesized netlist was performed on a Xilinx Virtex-7 (XC7V2000TFLG1925-2) FPGA prototype board. The area complexity (#Logic Elements), total delay, and power consumption are tabulated in Table 4. Here, the total delay is obtained as the product of the latency and clock period of the respective designs.

It is observed from the table that when compared to the multiplier in [30], the proposed multiplier requires up to 25% and 53% less power for m = 8 and m = 163, respectively, and requires up to 29% and 37% less area for m = 8 and m = 163, respectively. When compared to the multiplier in [28]a, the proposed multiplier requires up to 32% and 48% less power for m = 8 and m = 163,

https://doi.org/10.4218/etrij.17.0116.0770

Table 3. Comparison of the ASIC implementation results of the proposed multiplier using the best of the corresponding multipliers.

	Design	[28]*a*	[30]	Fig. 4
$m = 8$	Total delay (ns)	13.68	7.41	3.2
	Power (mW)	0.7277	0.6931	0.2527
	Area (μm^2) ($\times 10^3$)	12.055	7.057	4.381
	PDP (mW × ns)	9.955	5.136	0.823
	ADP (μm^2 × ns) ($\times 10^3$)	164.912	52.298	14.019
$m = 163$	Total delay (ns)	303.51	157.78	65.2
	Power (mW)	175.5645	167.3967	95.1454
	Area (μm^2) ($\times 10^3$)	4,953.948	2,985.735	1,917.372
	PDP (mW × ns)	53,285.581	26,411.851	6,203.48
	ADP (μm^2 × ns) ($\times 10^6$)	1,503.573	471.089	125.013

Table 4. Comparison of the FPGA implementation results of the proposed multiplier using the best of the corresponding multipliers.

	Design	[28]*a*	[30]	Fig. 4
$m = 8$	Total delay (ns)	363.36	96.672	10.552
	Power (W)	0.771	0.692	0.517
	Area (#LEs)	190	139	98
	PDP (W × ns)	280.151	66.897	5.455
	ADP (#LEs × ns ($\times 10^3$))	69.038	13.437	1.034
$m = 163$	Total delay (ms)	7.403	1.970	0.215
	Power (W)	3.6	6.187	2.848
	Area (#LEs)	154,635	105,787	66,434
	PDP (W × ms)	26.651	12.188	0.612
	ADP (#LEs × ms)	1,144.763	208.400	14.283

respectively, and requires up to 48% and 57% less area for $m = 8$ and $m = 163$, respectively. The proposed systolic multiplier also achieves a better delay than the multipliers in [30] and [28]*a*. Hence, the multiplier achieves a better ADP and PDP. When compared to [30], the proposed multiplier achieves up to 91% and 94% less PDP for $m = 8$ and $m = 163$, respectively, and achieves up to 92% and 93% less ADP for $m = 8$ and $m = 163$, respectively. When compared to [28]*a*, it achieves up to 98% and 97% less PDP for $m = 8$ and $m = 163$, respectively, and achieves up to 98% and 98% less ADP for $m = 8$ and $m = 163$, respectively.

V. Conclusions

In this paper, a modified interleaving multiplication method that performs the multiplication of any two arbitrary field elements for a generic irreducible polynomial was proposed over GF(2^m). Based on this method, an SFG was derived and an optimal cut-set retiming performed. Based on the retimed SFG, a systolic multiplier was proposed that performs multiplication of finite-field elements over generic irreducible polynomials. The hardware complexity and delay are estimated for the proposed systolic multiplier and for systolic multipliers reported in literature. From the estimated results, it may be concluded that the proposed systolic multiplier achieves low hardware complexity and comparable latency, critical path, and total delay when compared to corresponding multipliers. Based on the ASIC and FPGA synthesis results obtained, it can be concluded that the proposed systolic multiplier achieves a better performance in terms of power consumption, area cost, delay, power-delay product, and area-delay product when compared to the best of the corresponding multipliers. Moreover, the proposed systolic multiplier is scalable, modular, and regular, and it is hence well suited for VLSI implementations.

References

[1] B. Schneier, “Foundations”, in *Applied Cryptography*, London, UK: John Wiley & Sons Inc., 1996.

[2] FIPS PUB 46-3, *Data Encryption Standard (DES)*, NIST, Springfield, VA, USA, 1977.

[3] FIPS PUB 197, *Advanced Encryption Standard (AES)*, NIST, Springfield, VA, USA, 2001.

[4] N. Koblitz, “Elliptic Curve Cryptosystems,” *Math. Comput.*, vol. 48, 1987, pp. 203–209.

[5] V.S. Miller, “Use of Elliptic Curves in Cryptography,” *Proc. Adv. Cryptology-Crypto*, CA, USA, Aug. 18–22, 1986, pp. 417–426.

[6] R.L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” *Commun. ACM*, vol. 21, no. 2, Feb. 1978, pp. 120–126.

[7] T.C. Chen, S.W. Wei, and H.J. Tsai, “Arithmetic Unit for Finite Field GF(2^m),” *IEEE Trans. Circuits Syst. I*, vol. 55, no. 3, Apr. 2008, pp. 828–837.

[8] S. Roman, *Field Theory*, New York, USA: Springer, 2006.

[9] M. Wang and I.F. Blake, “Bit Serial Multiplication in Finite Fields,” *SIAM J. Discrete Math.*, vol. 3, no. 1, 1990, pp. 140–148.

[10] H. Wu, “Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis,” *Int. Workshop*

Cryptograph. Hardw. Embedded Syst., Worcester, MA, USA, Aug. 12–13, 1999, pp. 280–291.

[11] L. Song and K.K. Parhi, "Low-Energy Digit-Serial/Parallel Finite Field Multipliers," *J. VLSI Signal Process. Syst., Signal, Image Video Technol*., vol. 19, no. 2, July 1998, pp. 149–166.

[12] S.W. Wei, "A Systolic Power-Sum Circuit for GF (2^m)," *IEEE Trans. Comput.*, vol. 43, no. 2, Feb. 1994, pp. 226–229.

[13] L. Song and K.K. Parhi, "Optimum Primitive Polynomials for Low-Area Low-Power Finite Field Semi-systolic Multipliers," *IEEE Workshop Signal Process. Syst. SIPS 97-Des. Implement*., Leicester, UK, Nov. 5, 1997, pp. 375–384.

[14] H. Ho, "Design and Implementation of a Polynomial Basis Multiplier Architecture Over GF(2^m)," *J. Signal Process. Syst.*, vol. 75, no. 3, 2014, pp. 203–208.

[15] J.H. Guo and C.L. Wang, "Digit-Serial Systolic Multiplier for Finite Fields GF(2^m)," *IEE Proc.-Comput. Digital Techn*., vol. 145, no. 2, Mar. 1998, pp. 143–148.

[16] C.S. Yeh, I.S. Reed, and T.K. Truong, "Systolic Multipliers for Finite Fields GF(2^m)," *IEEE Trans. Comput*., vol. C-33, no. 4, Apr. 1984, pp. 357–360.

[17] C.L. Wang and J.L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields GF(2^m)," *IEEE Trans. Circuits Syst.*, vol. 38, no. 7, 1991, pp. 796–800.

[18] C.-W. Wu and M.-K. Chang, "Bit-Level Systolic Arrays for Finite-Field Multiplications," *J. VLSI Signal Process. Syst*., vol. 10, no. 1, June 1995, pp. 85–92.

[19] S.K. Jain and K.K. Parhi, "Low Latency Standard Basis GF (2^m) Multiplier and Squarer Architectures," *Int. Conf. Acoustics, Speech, Signal Process*., Detroit, MI, USA, May 9–12, 1995, pp. 2747–2750.

[20] S.K. Jain, L. Song, and K.K. Parhi, "Efficient Semisystolic Architectures for Finite-Field Arithmetic," *IEEE Trans, Very Large Scale Integr. Syst.*, vol. 6, no. 1, Mar. 1998, pp. 101–113.

[21] C.K. Koc and T. Acar, "Montgomery Multiplication in GF (2^k)," *Des., Codes Cryptography*, vol. 14, no. 1, Apr. 1998, pp. 57–69.

[22] S. Kwon, C.H. Kim, and C.P. Hong, "A Systolic Multiplier with LSB First Algorithm Over GF(2^m) Which is as Efficient as the One with MSB First Algorithm," *Int. Symp. Circuits Syst*., Bangkok, Thailand, May 25–28, 2003, pp. V-633–V-636.

[23] C.Y. Lee, Y.H. Chiu, and C.W. Chiou, "New Bit-Parallel Systolic Multiplier over GF(2^m) Using the Modified Booth's Algorithm," *IEEE Asia Pacific Conf. Circuits Syst*., Singapore, Dec. 4–7, 2006, pp. 610–613.

[24] S. Kwon, C.H. Kim, and C.P. Hong, "Unidirectional Two Dimensional Systolic Array for Multiplication in GF(2^m) Using LSB First Algorithm," *Int. Workshop Fuzzy Logic Applicat*., Crema, Italy, Sept. 2005, pp. 420–426.

[25] C.W. Chiou et al., "Efficient VLSI Implementation for Montgomery Multiplication in GF(2^m)," *Tamkang J. Sci. Technol.*, vol. 9, no. 4, Apr. 2006, pp. 365–372.

[26] C.Y. Lee et al., "Low-Complexity Bit-Parallel Systolic Multipliers Over GF(2^m)," *IEEE Conf. Syst., Man, Cybernetics*, Taipei, Taiwan, Oct. 8–11, 2006, pp. 1–6.

[27] C.W. Chiou, C.Y. Lee, and J.M. Lin, "Finite Field Polynomial Multiplier with Linear Feedback Shift Register," *Tamkang J. Sci. Technol.*, vol. 10, no. 3, 2007, pp. 253–264.

[28] C.Y. Lee, "Low-Complexity Bit-Parallel Systolic Multipliers Over GF (2^m)," *Integr., VLSI J*., vol. 41, no. 1, Jan. 2008, pp. 106–112.

[29] C.Y. Lee, "Multiplexer-Based Bit-Parallel Systolic Multipliers Over GF(2^m)," *Comput. Electr. Eng.*, vol. 34, no. 5, Sept. 2008, pp. 392–405.

[30] A.P. Fournaris and O. Koufopavlou, "Versatile Multiplier Architectures in GF(2^k) Fields Using the MONTGOMERY Multiplication Algorithm," *Integr., VLSI J*., vol. 41, no. 3, May 2008, pp. 371–384.

[31] S. Kwon, C.H. Kim, and C.P. Hong, "More Efficient Systolic Arrays for Multiplication in GF(2^m) Using LSB First Algorithm with Irreducible Polynomials and Trinomials," *Comput. Electr. Eng.*, vol. 35, no. 1, Jan. 2009, pp. 159–167.

[32] K.W. Kim and J.C. Jeon, "Polynomial Basis Multiplier Using Cellular Systolic Architecture," *IETE J. Res.*, vol. 60, no. 2, 2014, pp. 194–199.

[33] S.E. Mathe and L. Boppana, "Efficient Bit-Parallel Systolic Polynomial Basis Multiplier over GF(2^8) based on Irreducible Polynomials." *Indian J. Sci. Technol*., vol. 9, no. S1, 2016, pp. 1–6.

[34] S.S. Erdem, T. Yanik, and C.K. Koc, "Polynomial Basis Multiplication oVer GF(2^m)," *Acta Applicandae Math.*, vol. 93, no. 1, Sept. 2006, pp. 33–55.

[35] F. Rodriguez-Henriquez et al., "Binary Finite Field Arithmetic", in *Cryptographic Algorithms on Reconfigurable Hardware*. New York, USA; Springer, 2007, pp. 139–188.

https://doi.org/10.4218/etrij.17.0116.0770

Sudha Ellison Mathe received his B.Tech degree in Electronics and Communications Engineering from Nagarjuna University, India, in 2011 and his ME degree in Embedded Systems from Birla Institute of Technology and Science-Pilani, India, in 2013. Currently, he is working towards his doctorate in the field of VLSI at the National Institute of Technology-Warangal, India. His research areas include VLSI architectures, cryptography, and finite fields.

Lakshmi Bopanna received her B.Tech degree in Electronics and Communications Engineering from Nagarjuna University, India, M.Tech degree in Electronics and Instrumentation Engineering from the National Institute of Technology-Warangal, India, in 1990, and PhD degree from the Indian Institute of Technology-Kharagpur, India in 2010. She is currently working as an Associate Professor at the National Institute of Technology-Warangal, India. Her areas of interests include digital system design, microprocessor systems, and VLSI architectures. She is also a peer-reviewer for Elsevier journals.