Design and Implementation of a Sequential Polynomial Basis Multiplier over GF(2^m)

Sudha Ellison Mathe[1*] and Lakshmi Boppana[1]
[1] Department of Electronics and Communication Engineering, National Institute of Technology-Warangal
Warangal, Telangana 506004 - India
[e-mail: ellison@nitw.ac.in, lakshmi@nitw.ac.in]
*Corresponding author: Sudha Ellison Mathe

Received August 29, 2016; revised January 17, 2017; accepted February 22, 2017; published May 31, 2017

Abstract

Finite field arithmetic over GF(2^m) is used in a variety of applications such as cryptography, coding theory, computer algebra. It is mainly used in various cryptographic algorithms such as the Elliptic Curve Cryptography (ECC), Advanced Encryption Standard (AES), Twofish etc. The multiplication in a finite field is considered as highly complex and resource consuming operation in such applications. Many algorithms and architectures are proposed in the literature to obtain efficient multiplication operation in both hardware and software. In this paper, a modified serial multiplication algorithm with interleaved modular reduction is proposed, which allows for an efficient realization of a sequential polynomial basis multiplier. The proposed sequential multiplier supports multiplication of any two arbitrary finite field elements over GF(2^m) for generic irreducible polynomials, therefore made versatile. Estimation of area and time complexities of the proposed sequential multiplier is performed and comparison with existing sequential multipliers is presented. The proposed sequential multiplier achieves 50% reduction in area-delay product over the best of existing sequential multipliers for m = 163, indicating an efficient design in terms of both area and delay. The Application Specific Integrated Circuit (ASIC) and the Field Programmable Gate Array (FPGA) implementation results indicate a significantly less power-delay and area-delay products of the proposed sequential multiplier over existing multipliers.

Keywords: Finite field, Cryptography, Polynomial basis, Application Specific Integrated Circuit, Area-delay product, Field Programmable Gate Arrays.

https://doi.org/10.3837/tiis.2017.05.021 **ISSN : 1976-7277**

1. Introduction

Security of information has become a crucial issue now-a-days because of vast technological developments in the field of computers and internet. Cryptography is used in information security to prevent unauthorized or accidental disclosure of information while in transit across an insecure medium. Modern cryptography came into existence with the advent of computer technology, and mainly deals with development of algorithms to hide information whose strength lies in the mathematical theory and computational hardness assumptions. It can be classified into two types: symmetric cryptosystem and asymmetric cryptosystem [1]. Symmetric cryptosystem uses same secret key for both the encryption and decryption process, whereas the asymmetric cryptosystem uses different keys. Data encryption standard (DES) [2] and advanced encryption standard (AES) [3] are some examples for symmetric cryptosystem, and elliptic curve cryptography (ECC) [4-5] and RSA [6] are some examples for asymmetric cryptosystem.

The complexity of many cryptographic schemes, when implemented in hardware, depends mainly on arithmetic operations in finite fields. The basic finite field operations are addition, subtraction, multiplication, division, exponentiation and inversion. A simple exclusive-OR (XOR) can realize the addition and subtraction operations. More complex operations such as division, exponentiation and inversion can be realized using repeated multiplication operations [7-9]. Therefore, the multiplication operation is the basic unit for all the arithmetic operations involved in a finite field. Many practical applications exist for finite field arithmetic – cryptography [10], error correcting codes [11], pseudo random number generation [12] and Reed-Solomon codes [13]. Particularly in cryptography, elliptic curve cryptosystems and some symmetric cryptosystems utilize the arithmetic operations involved in binary extension fields GF(2^m) and prime fields GF(p) [14-16]. However, binary extension fields offer efficient hardware realizations compared to prime fields due to their carry-free addition property.

Three popular basis representations exist for finite fields: Normal Basis (NB), Polynomial Basis (PB) and Dual Basis (DB) [17], and each basis has its own distinct advantages. The hardware implementations of NB multipliers typically consume less power compared to other bases and it is attractive for cryptosystems that utilize frequent squaring. Multiplications in PB are relatively easy and less complex whereas the hardware implementations consume more power compared to NB multipliers. The DB multipliers require lesser area than the other two bases. However, PB multipliers are the most popular among the three because they can be matched to the input or output of any system, whereas the NB and DB multipliers require 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. There are various types of irreducible polynomials which characterize the finite field multipliers. They are general/generic polynomials [18], trinomials [19], pentanomials [20],

all-one polynomials (AOP) [21], equally-spaced polynomials (ESP) [18] etc., which are recommended for cryptographic applications. The ESPs are polynomials whose terms are equally spaced (in degree) by *r*. The trinomials, pentanomials and AOPs are special cases of ESPs and they are distinguished based on the spacing of the terms. The generic polynomials can have a random number of terms and 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 due to less number of terms and consequently lower computational complexity. On the other hand, these multipliers cannot be utilized in all the applications due to the limitation of fixed number of terms. The multipliers based on AOPs require more hardware complexity compared to the multipliers based on trinomials or pentanomials due to more number of terms and consequently higher computational complexity. On the contrary, the ease of representation of AOP facilitates efficient implementations and simpler structures in hardware. The ESPs are well structured and have higher hardware complexity compared to AOP multipliers. The hardware complexity of the multipliers based on generic polynomials cannot be determined beforehand due to the randomness in the number of terms of the polynomials. Therefore, the multipliers for generic polynomials are highly generic and operate on any type of polynomials.

Various algorithms are proposed in the literature to perform finite field multiplications such as the Karatsuba-Ofman algorithm [22], Montgomery multiplication algorithm [23], Mastrovito multiplier [24], Cantor multiplier [25] and the FFT multipliers [26]. These algorithms were proposed to reduce the complexity of multiplication operation and to achieve well-suited hardware implementations. Consequently, a variety of hardware architectures have been reported in the literature for polynomial basis multiplication such as: parallel [18][27], sequential [28-29] and pipelined [30-32] architectures. The parallel architectures can implement the multiplication operation in less clock cycles while increasing the area, whereas the sequential architectures require more clock cycles while reducing the area. Pipelined architectures achieve a balance of both time and area complexities. Bit-serial [27], [30-33], digit-serial [27], [29], [31], [34], [35], and bit-parallel [18], [19], [27-28], [36] designs have also been proposed by many researchers. Bit-serial architectures require more clock cycles utilizing lesser area, whereas the bit-parallel architectures utilize more area and require less clock cycles to perform the multiplication operation. The digit-serial architectures offer trade-off between area and speed. Several systolic [27], [31], [34], [37], [38], semi-systolic [30], [39], [40] and non-systolic [27], [41], [42] bit-parallel architectures are proposed in the literature. Systolic architectures have several replicas of the same structure to perform fast computations with large area overheads. The non-systolic architectures have a single structure offering less area overhead and low speed when compared to the systolic designs. The semi-systolic architectures achieve a trade-off between speed and area.

In this paper, an optimized algorithm is derived based on a method of serial interleaved multiplication to achieve better trade-off between area complexity and delay of the multiplier architecture. The proposed algorithm performs multiplication with interleaved modular reduction of two arbitrary elements for a generic, field defining irreducible

polynomial. Subsequently, a versatile sequential polynomial basis multiplier is realized for multiplication of two elements over GF(2^m). The proposed sequential multiplier achieves low area-delay product and results in a latency of *m* clock cycles. The proposed sequential multiplier and other sequential polynomial basis multipliers available in the literature are implemented in ASIC and FPGA and the proposed architecture achieves low area-delay product and less power consumption when compared to existing multipliers. Moreover, the AES and Twofish algorithms, incorporating the proposed multiplier and existing multipliers, are implemented in FPGA and the proposed multiplier achieves significantly low power-delay and area-delay products.

The organization of this paper is as follows: preliminaries of conventional polynomial basis multiplication is presented in section 2, section 3 presents the proposed polynomial basis multiplier, section 4 gives the complexity analysis and implementation results, followed by conclusions in Section 5.

2. Preliminaries

The conventional polynomial basis (PB) representation and multiplication of the field elements over GF(2^m) is presented in this section [43]. GF(2^m) is an extension field of GF(2) having an *m*-dimensional vector space over it, where GF(2) is a binary field having only two elements {0, 1}. The addition and subtraction operations can be performed by the logical exculsive-OR (XOR) operation and the multiplication operation can be performed by the logical AND operation over GF(2). However, the multiplication over GF(2^m) is performed by multiplying the two polynomials and modular reducing the result by the irreducible polynomial.

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

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

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

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

$$\Omega = \left\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\right\} \tag{2}$$

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

$$GF(2^m) = \left\{g(x) \mid g(x) = g_{m-1}x^{m-1} + ... + g_1 x + g_0\right\} \tag{3}$$

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

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

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

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

$$D(x) = \left(A(x) \times B(x)\right) \bmod T(x) \tag{5}$$

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

$$\begin{aligned} C(x) &= A(x) \times B(x) \\ &= \left(a_0 + a_1 x + ... + a_{m-1} x^{m-1}\right) \times (b_0 + b_1 x + ... + b_{m-1} x^{m-1}) \\ &= c_0 + c_1 x + ... + c_{2m-2} x^{2m} \end{aligned} \tag{6}$$

Modular reduction: The intermediate polynomial $C(x)$ of degree at most 2m-2 is modular reduced by a degree m irreducible polynomial $T(x)$ resulting in the polynomial $D(x)$ of degree at most m-1 which is the final result of the multiplication operation.

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

Thus the multiplication of two polynomials of degree m-1 results in a polynomial of degree m-1 such that the resultant polynomial resides in the given field GF(2^m).

3. Proposed Sequential Polynomial Basis Multiplier

Various algorithms for fast computation of multiplications over finite fields are available in the literature. Among them, the multiplication algorithms with interleaved modular reduction provide a simple, fast and efficient multiplication technique wherein the multiplication and modular reduction are performed simultaneously. In this section, a serial interleaving multiplication algorithm [44] is analyzed and transformed into a modified serial multiplication algorithm with interleaved modular reduction over GF(2^m), efficient for realizing low hardware architectures. The proposed algorithm efficiently transforms the polynomial multiplications into much simpler logical exclusive-OR, AND and shift operations thus obtaining simpler hardware structures by distributing them over m iterations. A versatile sequential polynomial basis multiplier architecture realized from the proposed algorithm is also presented in this section.

3.1 Proposed Finite Field Multiplication Algorithm with Interleaved Modular Reduction

Let $a = (a_{m-1}, \dots, a_1, a_0)$ and $b = (b_{m-1}, \dots, b_1, b_0)$ be the binary representations of the two elements, $A(x)$ and $B(x)$, over GF(2^m), respectively. Let $t = (t_{m-1}, \dots, t_1, t_0)$ be the binary representation of the field defining irreducible polynomial $T(x)$ of degree at most m, and let $p = (p_{m-1}, \dots, p_1, p_0)$ be the accumulator of the intermediate calculations. The proposed algorithm is given in Algorithm1.

Algorithm1: Proposed multiplication algorithm with interleaved modular reduction over GF(2^m)	
1: INITIALIZATION: $p = 0$, *counter* = 0	
2: FOR *counter* = 0 TO m-1 DO	
3: $p = p \oplus (a \,\&\, b_0)$	&: Logical AND operation
4: $am = a_{m-1}$	$\oplus$: Logical XOR operation
5: $a = a << 1$	<<: Left shift operation
6: $a = a \oplus (t \,\&\, am)$	
7: $b = b >> 1$	>>: Right shift operation
8: END FOR	

Proof of Algorithm1: The two arbitrary elements $A(x)$ and $B(x)$ in GF(2^m) can be expressed as

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

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

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

It may be observed from (9) that $C(x)$ is the summation of the multiplication result of b_i and $A(x).x^i$; for all i = 0, 1, … , m-1 i.e. the entire summation is carried out in m iterations. $A(x).x^i$ is calculated by the modular reduction step which is then multiplied with b_i using AND operation; for all i = 0, 1, … , m-1. Contrary to the generic case of summation by addition, the exclusive-OR (XOR) operation is considered for the summation of each $bi.A(x).x^i$; for all i = 0, 1, … , m-1, since the addition is simply an XOR operation over GF(2). Hence, the calculation of $C(x)$ in (9) is transformed as Steps 3, 4, 8 in Algorithm1. Here, $p = (p_{m-1}, \dots, p_1, p_0)$ acts as the accumulator of $A(x)x^i$ and is initialized to zero at the beginning of each multiplication operation.

The modular reduction in the serial interleaving multiplication algorithm [44] is

performed as given below

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

(10) is evaluated for each i as follows
For $i = 0$:

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

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

$$\begin{aligned} A(x) &= \left(A(x) \times x\right) \bmod T(x) \\ &= \left(a_0 x + a_1 x^2 + \ldots + a_{m-1} x^m\right) \bmod \left(t_0 + t_1 x + \ldots t_{m-1} x^{m-1} + x^m\right) \\ &= a_{m-1} t_0 + \left(a_{m-1} t_1 + a_0\right) x + \left(a_{m-1} t_2 + a_1\right) x^2 + \ldots + \left(a_{m-1} t_{m-1} + a_{m-2}\right) x^{m-1} \end{aligned} \tag{12}$$

It is revealed from (12) that the modular reduction is reduced to the summation of $a_{m-1}.T(x)$ and $A(x).x^i$. The $A(x).x^i$ is computed by left shifting $A(x)$ by i times; for all $i = 0, 1, 2, \ldots, m-1$. This result is also used in the polynomial multiplication step given above. The $a_{m-1}.T(x)$ is computed by bit-wise AND operation of a_{m-1} with the binary representation of $T(x)$ i.e. $(t_{m-1}, \ldots, t_1, t_0)$. The summation is carried out in m iterations using the XOR operation. Therefore, the modular reduction step can be transformed as Steps 5, 6, 7 in Algorithm1.

Both the polynomial multiplication and modular reduction steps occur simultaneously and thereby resulting in an interleaved algorithm. The two transformed algorithms for polynomial multiplication and modular reduction work in cohesion resulting in the proposed algorithm – Algorithm1.

3.2 Proposed Sequential Polynomial Basis Multiplier Architecture

Fig. 1 shows the versatile sequential polynomial basis multiplier realized from the proposed algorithm. The top-level architecture is comprised of two main modules A and B and three m-bit registers. The multiplier takes one m-bit input – t; where, t denotes the binary representation of the irreducible polynomial. Module A computes the polynomial multiplication and module B computes the modular reduction. The logic diagrams of the modules A and B are shown in **Fig. 2** and **Fig. 3** respectively. The inputs of module A are a, b_0, p and output is p_{out}. In module A, an AND operation is performed on a with the LSB of b, b_0, to obtain an m-bit result, aa. aa is bit-wise XORed with the input p which results in the output of module A i.e. p_{out}. The inputs of module B are a, b, t and outputs are a_{new} and b_{new}. In module B, b is right shifted by one bit to obtain b_{new}. a is left shifted by one bit to obtain an intermediate result, al. An AND operation is performed on t with a_{m-1} to obtain tt.

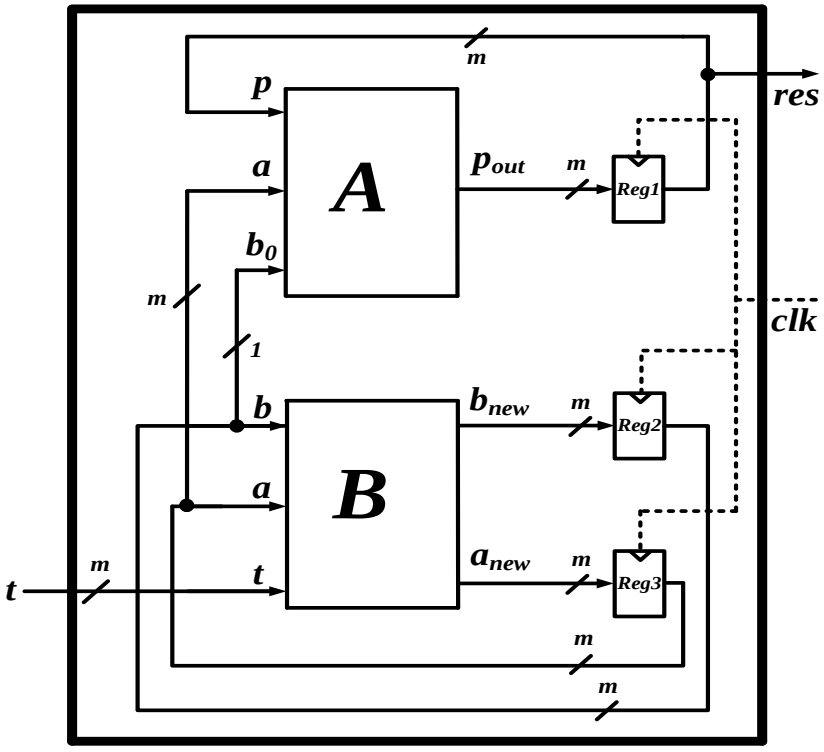

Fig. 1. Top level architecture of the proposed sequential polynomial basis multiplier

tt is XORed with *al* to obtain a_{new}. Before the multiplication operation begins, the registers *Reg2* and *Reg3* are initialized with the multiplicands *b* and *a* respectively and *Reg1* is cleared. For every clock cycle, module A computes the new *p* value given by p_{out} and module B computes the new *a* and *b* values given by a_{new} and b_{new} respectively. The final multiplication result is given by *res* after *m* clock cycles. The architecture is made versatile since it can perform multiplication of any two arbitrary elements *a* and *b* in GF(2^m) for any generic, field defining irreducible polynomial, *t*.

4. Complexity Analysis and Implementation Results

In this section, estimation of area complexity and delay of the proposed sequential multiplier is performed, and comparison with other sequential polynomial basis multipliers available in the literature is presented. The comparison is performed in terms of area, delay and area- delay product; where area is expressed in terms of total transistor count of the multiplier and delay is computed as a product of critical path and latency. The application specific integrated circuit (ASIC) and the field programmable gate array (FPGA) implementation of the proposed multiplier and existing multipliers is also performed and the area, delay, power consumption, power-delay product and area-delay product results

are also presented in this section. The implementation results of the AES and Twofish algorithms, incorporating the proposed sequential multiplier and existing multipliers, are also presented in this section; since these algorithms utilize the finite field multiplications extensively.

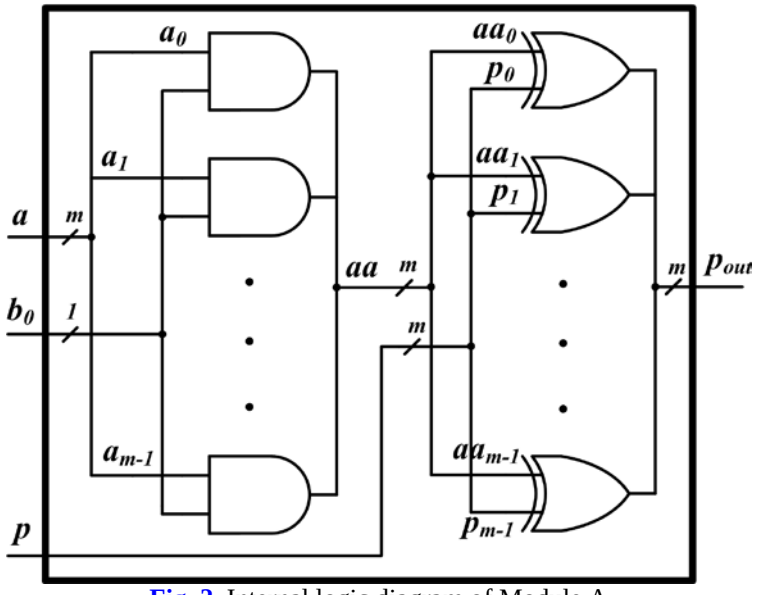

Fig. 2. Internal logic diagram of Module A

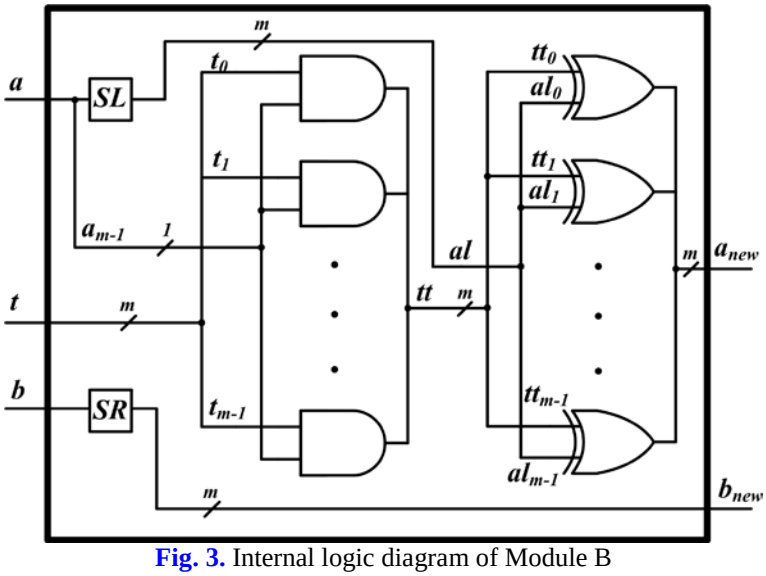

Fig. 3. Internal logic diagram of Module B

4.1 Estimation and Comparison of Area and Time Complexities

As discussed in section 3.2, the proposed sequential multiplier is comprised of three m-bit registers, one module A and one module B. Both module A and module B consists of m 2-input XOR gates and m 2-input AND gates each. An m-bit register can be realized using m 1-bit latches. Since the proposed architecture requires three m-bit registers, the total 1-bit latches required is $3m$. The shifting blocks, SL and SR, are comprised of only re-wiring and hence do not contribute to any complexity. Therefore, the total area complexity of the proposed architecture is $2m$ 2-input XOR gates, $2m$ 2-input AND gates and $3m$ 1-bit latches. The critical path delay of the proposed multiplier is the maximum of delays of either module A or module B. It can be observed from the architecture that the delays of module A and module B are equal i.e. $T_X + T_A$, where T_X and T_A are the delays of a 2- input XOR gate and a 2-input AND gate, respectively. Hence, the critical path delay of the multiplier is computed as $T_X + T_A$. As established by the proposed algorithm, the multiplication of two m-bit elements is computed over m iterations. Hence, the resultant latency is m clock cycles.

Table 1 presents the comparison of area complexity, latency and critical path of the proposed multiplier with other PB multipliers [32], [39], [37], [33], [38], [30], [28], [45], [40], [29] available in the literature. It may be noted that T_N, T_O and T_M denotes the delay of an inverter, 2-input OR gate and 2:1 multiplexer (MUX), respectively. In [32], a bit-serial pipelined multiplier is presented which allows variable field dimension m without any change in the hardware. The condition to be satisfied to achieve low circuit complexity is $1 < m \leq M$, where M is the maximum dimension that the multiplier supports. A non-versatile systolic design is given in [39]. [37] presents a low-complexity bit-parallel systolic multiplier for irreducible trinomials. A bit-serial polynomial basis multiplier for GF(2^m) is given in [33], where $1 < m \leq M$. M determines the maximum size the multiplier can support which allows for flexibility and ease of configuration. It can be noted that an additional demultiplexer (DMUX) and (m-1) additional OR gates are used in this multiplier. [38] gives a bit-parallel systolic multiplier for trinomials based on Hankel matrix-vector multiplications in which the standard basis multiplication is decomposed into k-term Hankel matrix-vector multiplications and hence the latency is obtained at $m+k$. A versatile bit-serial montgomery multiplier for generic irreducible polynomials is presented in [30]*a* and a systolic version is given in [30]*b*. A partially versatile, low latency sequential multiplier for general irreducible polynomials is given in [28] in which the condition $m \geq 2k_t$-1 must hold true, where k_t is the degree of the second leading term of the irreducible polynomial. A versatile bit-serial multiplier is proposed using MSB-first method in [45] having low area, flexibilty and simplicity. A partially versatile polynomial basis multiplier is given in [29] for any irreducible polynomial which satisfies the condition $m \geq k_t+1$, where k_t is the degree of the second leading term of the irreducible polynomial. In [40], a semi-systolic montgomery multiplier using cellular systolic architecture with reduced time complexity is presented.

In order to highlight the differences among various multiplier designs, the irreducible polynomial $f(x) = x^{163}+x^7+x^6+x^3+1$ recommended by National Institute of Standards and Technology (NIST) is considered as an example, where the order of the finite field is m =

163. Here, the second leading term of the irreducible polynomial, k_t, is 7 for the multiplier of [28]. The multiplication is decomposed into three Hankel matrix-vector multiplications, and hence k is 3 for the multiplier of [38]. The comparison of area complexity in terms of gate count cannot provide a clear difference among the multipliers considered. A better area complexity comparison can be achieved using a common parameter – transistor count. Moreover, latency alone cannot achieve a fair comparison of computation time; total delay as a product of latency and critical path must be considered.

Table 1. Comparison of area and time complexities of the proposed multiplier with existing multipliers over GF(2^m)

Design	#XOR	#AND	#MUX	#Latches	Latency	Critical Path
[32][2]	$2m$	$3m$	m	m^2	$3m$	$(T_A+T_N+T_O)log_2m+T_X$
[39][1,6]	$2m^2$	$2m^2$	m^2	$8m^2$	$2m$	T_X+T_A
[37][3,1,7]	m^2+m-1	m^2	0	$3m^2+2m-2$	$2m-1$	T_X+T_A
[33][4,2,*]	m	$2m$	m[a]	$3m$	m	$T_X+2T_A+T_N+(m+1)T_O$
[38][3,1,7]	m^2+m	m^2	0	$4m^2+m$	$m+k$[b]	T_X+T_A
[30]*a*[4,2]	m	m	$2m+1$	$3m$	$2m$	T_X+T_A
[30]*b*[4,1]	m^2-1	m^2-m+1	$2m^2+m-3$	$2m^2-m$	$2m$	T_X+T_A
[28][5,3]	$(m^2+m)/2$	$(m^2+m)/2$	$4m$	$5m-1$	$2k_t$[c]$+1$	$T_Xlog_2m+2T_M+T_A$
[45][2,4]	$2m$	$4m$	m[*]$+m$[#]	$3m$	m	T_X+T_A
[40][1]	$2m^2+3m$	$2m^2+2m$	0	$3m^2+4m$	$m/2+1$	$2T_X+T_A$
[29][5,3]	$6m+18$	0	$14m+26$	$6m+7$	$m/4$	$4T_X+T_M$
Proposed	$\mathbf{2m}$	$\mathbf{2m}$	**0**	$\mathbf{3m}$	$\mathbf{m}$	$\mathbf{T_X+T_A}$

[*]$(m-1)$ OR gates. [#]Inverters. [a] DMUX. [b] k-term Hankel matrix. [c] Degree of the second leading term of the irreducible polynomial. [1]systolic or semi-systolic. [2] bit-serial. [3] bit-parallel. [4] versatile. [5] partially versatile. [6] non-versatile. [7] for trinomials.

Table 2 provides the comparison of area complexity (in terms of total transistor count), total delay (latency × critical path) and area-delay product. In order to estimate the transistor count of individual gates, traditional CMOS logic transistor counts [28] are used: six transistors for a 2-input XOR gate, six for a 1-bit 2:1 MUX, six for a 1-bit 1:2 DMUX, six for a 2-input OR gate, six for a 2-input AND gate and eight for a 1-bit latch. Some real time circuits from STMicroelectronics are considered to estimate the critical path delay of the multipliers. The typical propagation delay (t_{PD}) of the respective gates is considered to ensure fair comparison. The circuits used are M74HC86 (XOR gate, t_{PD} = 12ns), M74HC257 (MUX, t_{PD} = 11ns), M74HC08 (AND gate, t_{PD} = 6ns), M74HC32 (OR gate, t_{PD} = 8ns) and M74HC04 (INVERTER, t_{PD} = 8ns) [28]. It can be seen from **Table 2** that the proposed multiplier has lower area complexity and lower area-delay product when compared to other multipliers available in the literature. The table also provides the percentage reduction in area complexity and area-delay product of the multipliers available in the literature when compared with the proposed multiplier. It can be noted that all the multipliers, except multipliers in [33], [30]*a*, [45], [29], have over 90% area complexity

and area-delay product when compared to the proposed multiplier. The multipliers in [33], [30]*a*, [45], [29], when compared with the proposed multiplier, have 11.05%, 0.08%, 29.41% and 71.76% more area complexity respectively. However, the multiplier in [33] has very high area-delay product due its high critical path (multiple of *m*). The multipliers in [30]*a*, [45], [29], when compared with the proposed multiplier, have 50%, 28.13% and 66.17% more area-delay product respectively. In general, the area-delay product is considered as a trade-off parameter to determine the efficiency of an architecture in terms of both area and total delay; lower the product value, the more efficient is the design. As already discussed above, the area-delay product of the proposed multiplier is considerably lower when compared to other multipliers resulting in a very efficient design in terms of both area and delay. The resultant total delay of the proposed multiplier is moderate when compared to the other multipliers.

Table 2. Comparison of total transistor count, total delay and area-delay product for $m = 163$ recommended by NIST

Design	#Transis-tors	Latency (#Clock cycles)	Critical Path (ns)	Total Delay (ns)	ADP ($\times 10^6$)	% Reduction in Area	% Reducti-on in ADP
[32][2]	217442	489	174	85086	18501	96.40%	99.88%
[39][1,6]	2391210	326	18	5868	14032	99.67%	99.83%
[37][3,1,7]	906910	325	18	5850	5305	99.14%	99.56%
[33][4,2,*]	8796	163	1344	219072	1927	11.05%	98.81%
[38][3,1,7]	1118180	170	18	3060	3422	99.30%	99.32%
[30]*a*[4,2]	7830	326	18	5868	46	0.08%	50%
[30]*b*[4,1]	1008624	326	18	5868	5919	99.22%	99.61%
[28][5,3]	170816	15	106	1590	272	95.42%	91.54%
[45][2,4]	11084	163	18	2934	32	29.41%	28.13%
[40][1]	1285418	83	30	2490	3201	99.39%	99.28%
[29][5,3]	27704	41	60	2460	68	71.76%	66.17%
Proposed	**7824**	**163**	**18**	**2934**	**23**	**-**	**-**

[*] (*m*-1) OR gates. [#] Inverters. [a] DMUX. [b] *k*-term Hankel matrix. [c] Degree of the second leading term of the irreducible polynomial. [1] systolic or semi-systolic. [2] bit-serial. [3] bit-parallel. [4] versatile. [5] partially versatile. [6] non-versatile. [7] for trinomials.

Table 3 compares the total transistor count for $m = 163, 233, 283, 409$ and 571 recommended by NIST. It can be noted that the proposed multiplier exhibits a linear increase in area complexity similar to the multipliers in [33], [30]*a*, [45] and [29], whereas the remaining multipliers exhibit an exponential increase. This can be better illustrated in **Fig. 4** which depicts the above said property. It can be observed that, as the order of the finite fields (*m*) increases, the area (transistor count) of multipliers of [32], [39], [37], [38],

Table 3. Comparison of total transistor count for the five irreducible polynomials recommended by NIST

Design	m = 163	m = 233	m = 283	m = 409	m = 571
[32]	217442	441302	649202	1350518	2625458
[39]	2391210	4886010	7208010	15055290	29343690
[37]	906910	1850930	2729230	5696530	11097934
[33]	8796	12576	15276	22080	30828
[38]	1118180	2283400	3367700	7031528	13701716
[30]*a*	7830	11190	13590	19638	27414
[30]*b*	1008624	2061564	3041664	6354204	12386112
[28]	170816	342036	500336	1032308	1996208
[45]	11084	15844	19244	27812	38828
[40]	1285418	2620318	3861818	8054846	15685370
[29]	27704	39464	47864	69032	96248
Proposed	**7824**	**11184**	**13584**	**19632**	**27408**

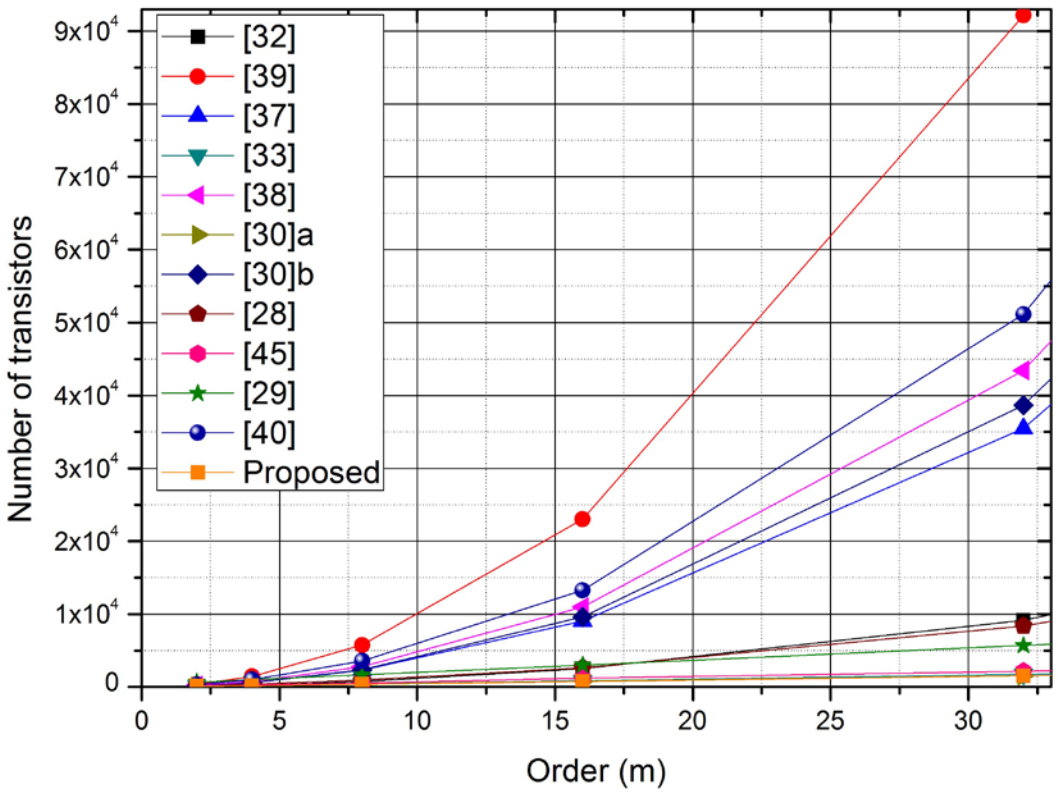

Fig. 4. Comparison of area complexity of the proposed sequential multiplier with existing multipliers

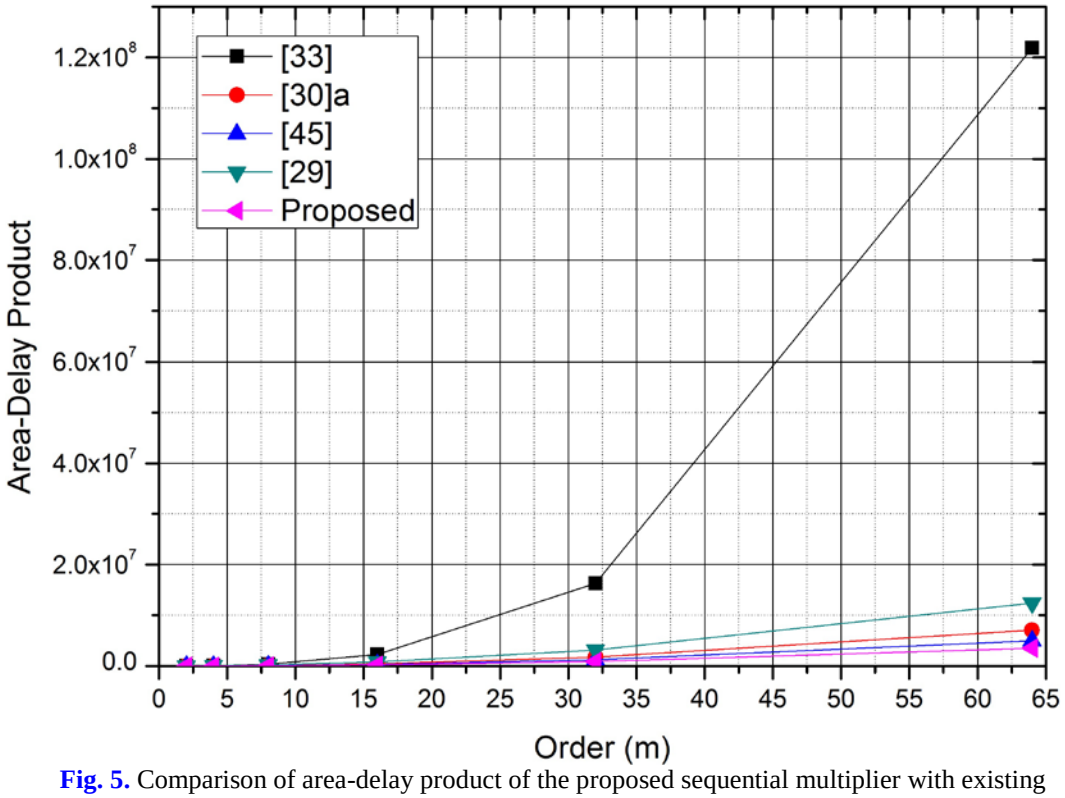

Fig. 5. Comparison of area-delay product of the proposed sequential multiplier with existing multipliers

of area-delay product of the proposed multiplier with the multipliers of [33], [30]*a*, [45] and [29] over a range of *m* is shown in **Fig. 5**. Only three multipliers available in the literature are considered here for clear graphical depiction. It can be seen that as the order of the finite fields (*m*) increases, the proposed multiplier has a lower area-delay product when compared to other multipliers available in the literature, indicating an efficient design in terms of both area and delay.

4.2 ASIC Implementation Results

The multipliers in [30]*a* and [29] are the best in area-delay product (ADP) among existing multipliers as discussed in Section 4.1. Therefore, the proposed multiplier along with the multipliers in [30]*a* and [29] are modeled in Verilog and synthesized by Cadence Encounter RTL Compiler Tool which uses UMC 0.18µm technology for $m = 8$ and $m = 163$. The finite field multipliers are mainly utilized in symmetric and asymmetric encryption techniques such as Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC), respectively. Therefore, the multipliers of order $m = 8$ and $m = 163$ are considered as an example for AES technique and ECC techniques, respectively. The delay, area and power consumption (at 100MHz frequency) obtained from the synthesis results are listed in **Table 4**. It can be concluded that the proposed multiplier, when compared to the multiplier

in [29], requires 45.95% & 76.46% less power for m = 8 and m =163, respectively and requires 48.12% & 76.50% less area for m = 8 and m = 163, respectively. When compared to the multiplier in [30]*a*, the proposed multiplier requires 68.22% & 88.262% less power for m = 8 and m =163, respectively, and requires 8.45% & 50.74% less area for m = 8 and m = 163, respectively. The proposed multiplier, when compared to the multiplier in [29], achieves 47.74% & 45.56% lower ADP and power-delay product (PDP), respectively, for m = 8 and 78.08% & 78.04% lower ADP and PDP, respectively, for m = 163. When compared to the multiplier in [30]*a*, the proposed multiplier achieves 13.59% & 70.00% lower ADP and PDP, respectively, for m = 8 and 53.67% & 88.96% lower ADP and PDP, respectively, m = 163. The proposed multiplier achieves better delay than that of the multipliers in [29] and [30]*a*.

Table 4. Comparison of ASIC implementation results of the proposed sequential multiplier with existing multipliers

***m* = 8**					
Design	**Delay (ns)**	**Power (μW)**	**Area (μm^2)**	**PDP (μW×ns)**	**ADP (μm^2×ns)**
Ho [29]	1.482	14.866	14.418	22.031	21.368
Fournaris [30]*a*	1.582	25.282	8.17	39.99	12.925
Proposed	**1.493**	**8.035**	**7.48**	**11.996**	**11.168**
***m* = 163**					
Design	**Delay (ns)**	**Power (μW)**	**Area (μm^2)**	**PDP (μW×ns)**	**ADP (μm^2×ns)**
Ho [29]	1.731	404.839	372.276	700.776	644.410
Fournaris [30]*a*	1.717	811.769	177.594	1393.807	304.929
Proposed	**1.615**	**95.282**	**87.48**	**153.880**	**141.280**

4.3 FPGA Implementation Results

The Verilog models of the proposed multiplier and the multipliers in [29] and [30]*a* are simulated and synthesized using Xilinx Vivado 2014.2 to verify the functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA protoype board. The delay, area, power consumption, ADP and PDP (at 66.67MHz frequency) obtained from the synthesis results are listed in **Table 5**. It can be concluded from the table that the proposed multiplier, when compared to the multiplier in [29], requires 14.12% & 15.98% less power for m = 8 and m = 163, respectively and requires 51.06% & 80.58% less area for m = 8 and m = 163, respectively. When compared to the multiplier in [30]*a*, the proposed multiplier requires 55.97% & 35.10% less power for m = 8 and m = 163, respectively, and requires same area for m = 8 & 49.39% less area for m = 163. The proposed multiplier, when compared to the multiplier in [29], achieves 73.39% & 53.28% lower ADP and PDP, respectively, for m = 8 and 93.18% & 70.48% lower ADP and PDP, respectively, for m = 163. When compared to the multiplier in [30]*a*, the proposed multiplier achieves 53.23% & 79.40% lower ADP and PDP, respectively, for m = 8 and 77.25% & 70.82% lower ADP and PDP, respectively, m = 163. The proposed multiplier achieves better delay than that of the multipliers in [29] and [30]*a*.

Table 5. Comparison of FPGA implementation results of the proposed sequential multiplier with existing multipliers

***m* = 8**					
Design	**Delay (ns)**	**Power (W)**	**Area (#LUTs)**	**PDP (W×ns)**	**ADP (#LUTs×ns)**
Ho [29]	1.455	0.262	47	0.381	68.385
Fournaris [30]*a*	1.691	0.511	23	0.864	38.893
Proposed	**0.791**	**0.225**	**23**	**0.178**	**18.193**
***m* = 163**					
Design	**Delay (ns)**	**Power (W)**	**Area (#LUTs)**	**PDP (W×ns)**	**ADP (#LUTs×ns)**
Ho [29]	11.686	0.713	850	8.33	9933.1
Fournaris [30]*a*	9.131	0.923	326	8.428	2976.706
Proposed	**4.105**	**0.599**	**165**	**2.459**	**677.325**

4.4 FPGA Implementation of AES with the Proposed Sequential Multiplier and Existing Multipliers

The AES algorithm is implemented on FPGA by incorporating the proposed sequential multiplier and the multipliers in [29] and [30]*a*. The Verilog models of the AES implementation of these multipliers are simulated and synthesized using Xilinx Vivado 2014.2 to verify the functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 FPGA protoype board (XC7VX1140TFLG1930-1). The delay, area, power consumption, ADP and PDP obtained from the synthesis results are listed in **Table 6**. It can be concluded from the table that the proposed sequential multiplier, when compared to the multiplier in [29], requires 42.70% less power and 36.07% less area. When compared to the multiplier in [30]*a*, the proposed sequential multiplier requires 71.32% less power and 22.22% less area. Moreover, the proposed sequential multiplier, when compared to the multiplier in [29], achieves 34.29% & 41.09% lower ADP and PDP, respectively. When compared to the multiplier in [30]*a*, the proposed sequential multiplier achieves 40.13% & 77.92% lower ADP and PDP, respectively. The proposed sequential multiplier achieves delay comparable to the multipliers in [29] and [30]*a*.

Table 6. Comparison of FPGA implementation results of AES with the proposed multiplier and existing multipliers

Design	**Delay (ns)**	**Power (W)**	**Area (#LUTs)**	**PDP (W×ns)**	**ADP (#LUTs×ns)**
Ho [29]	58.693	0.459	15793	26.94	926938.549
Fournaris [30]*a*	78.371	0.917	12981	71.87	1017333.951
Proposed	**60.327**	**0.263**	**10097**	**15.87**	**609121.719**

4.5 FPGA Implementation of Twofish with the Proposed Sequential Multiplier and Existing Multipliers

The Twofish algorithm is implemented on FPGA by incorporating the proposed sequential multiplier and the multipliers in [29] and [30]*a*. The Verilog models of the Twofish implementation of these multipliers are simulated and synthesized using Xilinx Vivado 2014.2 to verify the functionality. The synthesized netlist is implemented on a Xilinx Virtex-7 (XC7VX1140TFLG1930-1) FPGA protoype board. The delay, area, power consumption, ADP and PDP obtained from the synthesis results of the Twofish implementation are listed in **Table 7**. It can be concluded from the table that the proposed sequential multiplier, when compared to the multiplier in [29], requires 37.49% less power and 23.51% less area. When compared to the multiplier in [30]*a*, the proposed sequential multiplier requires 73.85% less power and 4.16% less area. Moreover, the proposed sequential multiplier, when compared to the multiplier in [29], achieves 17.68% & 32.84% lower ADP and PDP, respectively. When compared to the multiplier in [30]*a*, the proposed sequential multiplier achieves 16.94% & 77.34% lower ADP and PDP, respectively. The proposed sequential multiplier achieves delay comparable to the multipliers in [29] and [30]*a*.

Table 7. Comparison of FPGA implementation results of Twofish with the proposed multiplier and existing multipliers

Design	Delay (ns)	Power (W)	Area (#LUTs)	PDP (W×ns)	ADP (#LUTs×ns)
Ho [29]	85.359	0.274	19805	23.39	1690534.995
Fournaris [30]*a*	106.009	0.654	15805	69.33	1675472.245
Proposed	**91.870**	**0.171**	**15148**	**15.71**	**1391646.76**

5. Conclusions

In this paper, a multiplication algorithm with interleaved modular reduction is proposed, which performs multiplication of any two arbitrary field elements for any field defining irreducible polynomial over GF(2^m). A versatile sequential polynomial basis multiplier architecture over GF(2^m) is realized from the proposed algorithm. The area and delay complexities of the proposed multiplier are estimated and performance is compared with other sequential polynomial basis multipliers available in the literature. It may be concluded from the comparisons of the estimated results that the proposed multiplier achieves low area complexity for generic irreducible polynomials of degree *m*. The resultant area-delay product of the proposed multiplier is the lowest when compared to other multipliers, indicating an efficient multiplier design in terms of both area and delay. In addition, the area cost of the proposed multiplier increases linearly for a range of degrees of finite field, *m*. This is in contrast to the exponential increase in the area cost of majority of other multipliers available in the literature, thus indicating that the proposed sequential multiplier is more area-efficient for higher order finite fields. The proposed *m*-bit sequential multiplier architecture is scalable and modular and hence suitable for VLSI

implementations. From the ASIC and FPGA synthesis results of the multipliers, it can be concluded that the proposed sequential multiplier achieves significantly less power-delay product and area-delay product when compared to existing multipliers. In addition, the AES and Twofish algorithms are implemented in FPGA using the proposed sequential multiplier and existing multipliers. The proposed sequential multiplier achieves better area-delay product and power-delay product compared to existing multipliers. Moreover, the proposed design can be optimized for irreducible trinomials and pentanomials which are recommended by NIST.

References

[1] B. Schneier, "Foundations," in *Proc. of Applied Cryptography*, 2nd ed., John Wiley & Sons Inc., U.K., pp. 1-4, 1996. Article (CrossRef Link)

[2] "FIPS PUB 46-3 Data Encryption Standard (DES)," NIST - *Federal Information Processing Standard Publication*, October, 1999. Article (CrossRef Link)

[3] "FIPS PUB 197 Advanced Encryption Standard (AES)," NIST - *Federal Information Processing Standard Publication*, November, 2001. Article (CrossRef Link)

[4] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203-209, January, 1987. Article (CrossRef Link)

[5] V. Miller, "Use of elliptic curves in cryptography," in *Proc. of Advances in Cryptology-Crypto'85*, pp. 417-426, August 18-22, 1986. Article (CrossRef Link)

[6] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, February, 1978. Article (CrossRef Link)

[7] T.C. Chen, S.W. Wei, and H.J. Tsai, "Arithmetic unit for finite field GF(2^m)," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 3, pp. 828-837, May, 2008. Article (CrossRef Link)

[8] K. Kobayashi, and N. Takagi, "A combined circuit for multiplication and inversion in GF(2^m)," *IEEE Transactions on Circuits and Systems II*, vol. 55, no. 11, pp. 1144-1148, December, 2008. Article (CrossRef Link)

[9] K. Kobayashi, and N. Takagi, "Fast hardware algorithm for division in GF(2^m) based on the extended euclid's algorithm with parallelization of modular reductions," *IEEE Transactions on Circuits and Systems II*, vol. 56, no. 8, pp. 644-648, July, 2009. Article (CrossRef Link)

[10] R. Lidl, and H. Niederreiter, "Introduction to finite fields and their applications," *Revised edition, Cambridge University Press*, Cambridge, 1994. Article (CrossRef Link)

[11] F.J. MacWilliams, and N.J.A. Sloane, "The theory of error correcting codes," *1st ed., North-Holland Publishing Company*, New York, 1977. Article (CrossRef Link)

[12] C.C. Wang, and D. Pei, "A VLSI design for computing exponentiation in GF(2^m) and its application to generate pseudorandom number sequences," *IEEE Transactions on Computers*, vol. 39, no. 2, pp. 258-262, February, 1990. Article (CrossRef Link)

[13] E.R. Berlekamp, "Bit-serial Reed Solomon encoder," *IEEE Transactions on Information Theory*, vol. 28, no. 6, pp. 869-874, November, 1982. Article (CrossRef Link)

[14] R. Schroeppel, H. Orman, S. OMalley, *et al.*, "Fast key exchange with elliptic curve systems," in *Proc. of Advances in Cryptology-Crypto'95*, pp. 43-56, August 27-31, 1995. Article (CrossRef Link)

[15] E.D. Win, A. Bosselaers, S. Vandenberghe, *et al.*, "A fast software implementation for arithmetic operations in GF(2^n)," in *Proc. of Advances in Cryptology-ASIACRYPT '96*, pp. 65-76, November 3-7, 1996. Article (CrossRef Link)

[16] A.J. Menezes, *Applications of finite fields*, Kluwer Academic, Massachusetts, 1993. Article (CrossRef Link)

[17] S. Roman, *Field theory*, 2nd ed., Springer Verlag, New York, 2006. Article (CrossRef Link)

[18] A. Reyhani-Masoleh, and M.A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2^m)," *IEEE Transactions on Computers*, vol. 53, no. 8, pp. 945-959, June, 2004. Article (CrossRef Link)

[19] J.L. Imana, J.M. Sanchez, and F. Tirado, "Bit-parallel finite field multipliers for irreducible trinomials," *IEEE Transactions on Computers*, vol. 55, no. 5, pp. 520-533, April, 2006. Article (CrossRef Link)

[20] A. Cilardo, "Fast Parallel GF (2^m) Polynomial Multiplication for All Degrees," *IEEE Transactions on Computers*, vol. 62, no. 5, pp. 929-943, May, 2013. Article (CrossRef Link)

[21] M. Nikooghadam, and A. Zakerolhosseini, "Utilization of Pipeline Technique in AOP Based Multipliers with Parallel Inputs," *Journal of Signal Processing Systems*, vol. 72, no. 1, pp. 57-62, July, 2013. Article (CrossRef Link)

[22] A. Karatsuba, and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet physics doklady*, vol. 7, pp. 595-596, January, 1963. Article (CrossRef Link)

[23] M. Morales-Sandoval, C. Feregrino-Uribe, and P. Kitsos, "Bit-serial and digit-serial GF(2^m) Montgomery multipliers using linear feedback shift registers," *IET Computers & Digital Techniques*, vol. 5, no. 2, pp. 86-94, April, 2011. Article (CrossRef Link)

[24] E.D. Mastrovito, "VLSI Architectures for Computations in Galois Fields," *PhD thesis, Linkoping University*, Linkoping, Sweden, 1991. Article (CrossRef Link)

[25] D.G. Cantor, "On arithmetical algorithms over finite fields," *Journal of Combinatorial Theory*, vol. 50, no. 2, pp. 285-300, March, 1989. Article (CrossRef Link)

[26] J.V.Z. Gathen, and J. Gerhard, "Arithmetic and factorization of polynomial over F^2," in *Proc. of International Symposium on Symbolic and algebraic computation*, pp. 1-9, July 24-26, 1996. Article (CrossRef Link)

[27] P.K. Meher, "Systolic and non-systolic scalable modular designs of finite field multipliers for reed-solomon codec," *IEEE Transactions on Very Large Scale Integrated Systems*, vol. 17, no. 6, pp. 747-757, March, 2009. Article (CrossRef Link)

[28] J.L. Imana, "Low Latency GF(2^m) Polynomial Basis Multiplier," *IEEE Transactions on Circuits and Systems I*, vol. 58, no. 5, pp. 935-946, May, 2011. Article (CrossRef Link)

[29] H. Ho, "Design and Implementation of a Polynomial Basis Multiplier Architecture Over GF(2^m)," *Journal of Signal Processing Systems*, vol. 75, no. 3, pp. 203-208, June, 2014. Article (CrossRef Link)

[30] A.P. Fournaris, and O. Koufopavlou, "Versatile multiplier architectures in GF(2^k) fields using the Montgomery multiplication algorithm," *INTEGRATION the VLSI journal*, vol. 41, no. 3, pp. 371-384, May, 2008. Article (CrossRef Link)

[31] P.K. Meher, "Systolic and super-systolic multipliers for finite field based on irreducible trinomials," *IEEE Transactions on Circuits and Systems I*, vol. 55, no. 4, pp. 1031-1040, May, 2008. Article (CrossRef Link)

[32] M.A. Hasan, and M. Ebtedaei, "Efficient architectures for computations over variable dimensional Galois fields," *IEEE Transactions on Circuits and Systems I*, vol. 45, no. 11, pp. 1205-1210, November, 1998. Article (CrossRef Link)

[33] P. Kitsos, G. Theodoridis, and O. Koufopavlou, "An efficient reconfigurable multiplier architecture for Galois field GF(2^m)," *Microelectronics Journal*, vol. 34, no. 10, pp. 975-980, October, 2003. Article (CrossRef Link)

[34] J.H. Guo, and C.L. Wang, "Digit-serial systolic multiplier for finite fields GF(2^m)," *IEE Proceedings on Computers and Digital Techniques*, vol. 145, no. 2, pp. 143-148, April, 1998. Article (CrossRef Link)

[35] L. Song, and K.K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *The Journal of VLSI Signal Processing - Systems for Signal, Image, and Video Technology*, vol. 19, no. 2, pp. 149-166, July, 1998. Article (CrossRef Link)

[36] H. Fan, and M.A. Hasan, "Fast bit parallel-shifted polynomial basis multipliers in GF(2^n)," *IEEE Transactions on Circuits and Systems I*, vol. 53, no. 12, pp. 2606-2615, December, 2006. Article (CrossRef Link)

[37] C.Y. Lee, "Low complexity bit-parallel systolic multiplier over GF(2^m) using irreducible trinomials," *IEE Proceedings on Computers and Digital Techniques*, vol. 150, no. 1, pp. 39-42, February, 2003. Article (CrossRef Link)

[38] C.Y. Lee, Y.H. Chen, C.W. Chiou, and *et al.*, "Unified Parallel Systolic Multiplier Over GF(2^m)," *Journal of Computer Science and Technology*, vol. 22, no. 1, pp. 28-38, January, 2007. Article (CrossRef Link)

[39] W.C. Tsai, and S.J. Wang, "Two systolic architectures for multiplication in GF(2^m)," *IEEE Proceedings on Computers and Digital Techniques*, vol. 147, no. 6, pp. 375-382, December, 2000. Article (CrossRef Link)

[40] K.W. Kim, and J.C. Jeon, "Polynomial Basis Multiplier Using Cellular Systolic Architecture," *IETE Journal of Research*, vol. 60, no. 2, pp. 194-199, June 2014. Article (CrossRef Link)

[41] A. Reyhani-Masoleh, and M.A. Hasan, "A new construction of Massey-Omura parallel multiplier over GF(2^m)," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 511-520, June, 2002. Article (CrossRef Link)

[42] H. Wu, and M.A. Hasan, "Low complexity bit-parallel multipliers for a class of finite fields," *IEEE Transactions on Computers*, vol. 47, no. 8, pp. 883-887, August, 1998. Article (CrossRef Link)

[43] S.S. Erdem, T. Yanik, and C.K. Koc, "Polynomial Basis Multiplication over GF(2^m)," in *Proc. of Acta Applicandae Mathematicae*, vol. 93, pp. 33-55, September, 2006. Article (CrossRef Link)

[44] F. Rodriguez-Henriquez, N.A. Saqib, A.D. Perez, and *et al.*, "Binary Finite Field Arithmetic," in *Proc. of Cryptographic Algorithms on Reconfigurable Hardware*, 1st ed., Springer, New York, pp. 139-188, 2007. Article (CrossRef Link)

[45] A. Zakerolhosseini, and M. Nikooghadam, "Low-power and high-speed design of a versatile bit-serial multiplier in finite fields GF (2^m)," *INTEGRATION, the VLSI journal*, vol. 46, no. 2, pp. 211-217, March, 2013. Article (CrossRef Link)

Mr. M. Sudha Ellison received his B.Tech degree in Electronics and Communication Engineering from Nagarjuna University and M.E. degree in Embedded Systems from Birla Institute of Technology and Science-Pilani. Currently, he is working for his doctorate in the field of VLSI at National Institute of Technology-Warangal, India. His research areas include VLSI architectures, Crypography, Finite fields.

Dr. B. Lakshmi received her B.Tech degree in Electronics and Communication Engineering from Nagarjuana university, M.Tech degree in Electronic Instrumentation from National Institute of Technology, Warangal, and Ph.D degree in VLSI Architectures from I.I.T, Kharagpur. She is currently working as an Associate Professor in National Institute of Technology-Warangal. Her area of interests are Digital System Design, Microprocessor Systems and VLSI Architectures. She is also a reviewer for Elsevier journals.