LFSR-based Bit-Serial $GF(2^m)$ Multipliers using Irreducible Trinomials

José L. Imaña

Abstract—In this paper, a new architecture of bit-serial polynomial basis (PB) multipliers over the binary extension field $GF(2^m)$ generated by irreducible trinomials is presented. Bit-serial $GF(2^m)$ PB multiplication offers a performance/area trade-off that is very useful in resource constrained applications. The architecture here proposed is based on LFSR (*Linear-Feedback Shift Register*) and can perform a multiplication in m clock cycles with a constant propagation delay of $T_A + T_X$. These values match the best time results found in the literature for bit-serial PB multipliers with a slight reduction of the space complexity. Furthermore, the proposed architecture can perform the multiplication of two operands for t different finite fields $GF(2^m)$ generated by t irreducible trinomials simultaneously in m clock cycles with the inclusion of $t(m-1)$ flipflops and tm XOR gates.

Index Terms—Multipliers, LFSR, bit-serial, $GF(2^m)$, polynomial basis, trinomials.

1 Introduction

Binary extension fields $GF(2^m)$ play a fundamental role in several important applications, such as cryptography, coding theory and digital signal processing. These applications require efficient hardware implementations of $GF(2^m)$ arithmetic operations, especially for *multiplication*. This operation is considered the most important and complex one because exponentiation, inversion and division can be performed by means of repeated multiplications. Different representation bases can be used to perform $GF(2^m)$ arithmetic operations, although *polynomial basis* (PB) is the most widely used. Efficient methods and architectures for $GF(2^m)$ multiplication have been proposed for PB where the complexity depends on the generating irreducible polynomial used for the finite field. Special irreducible polynomials such as trinomials or pentanomials can provide important optimizations in terms of both area and speed. Different architectures for $GF(2^m)$ PB multiplication have also been proposed, like bit-parallel, bit-serial and digit-serial multipliers. Bit-parallel multipliers [1] present a high area complexity, but can perform the multiplication in only one clock cycle. Bit-serial multipliers [2], [3] are restricted in area, but need m clock cycles. Digit-serial architectures [4] represent a speed-area trade-off achieved by processing several coefficients at the same time. Polynomial basis multiplication requires a polynomial multiplication followed by a reduction modulo the irreducible polynomial $f(y)$ selected for the field [5]. Mastrovito proposed a new method to combine the above two steps together [1], [6]. A new PB multiplication approach applied to five types of irreducible trinomials was proposed in [7], where functions $\mathbf{S_i}$ and $\mathbf{T_i}$ were obtained from the decomposition of a product matrix. These functions are given by the sum of product terms and their addition is used for the computation of the product of two $GF(2^m)$ operands.

In this paper, a new bit-serial polynomial basis multiplier over the binary extension field $GF(2^m)$ generated by irreducible trinomials is presented. Bit-serial PB multiplication offers a performance/area trade-off that is very useful in resource constrained applications such as smart cards. The architecture here proposed is based on LFSR (*Linear-Feedback Shift Register*) and can perform a multiplication in m clock cycles with a constant propagation delay of $T_A + T_X$ (where T_A and T_X represent the delay of 2-input AND and XOR gates, respectively). These values match the best time results found in the literature for bit-serial PB multipliers with a slight reduction of the space complexity. A characteristic of the proposed architecture is that it can perform the multiplication of two operands for t different finite fields $GF(2^m)$ generated by t irreducible trinomials simultaneously in m clock cycles with the inclusion of $t(m-1)$ flipflops and tm XOR gates. Based on [7], a new general multiplication algorithm over irreducible trinomials $f(y) = y^m + y^n + 1$, with $1 \leq n \leq m-1$, is also proposed.

The paper is organized as follows. Section 2 provides notation and mathematical background. Irreducible trinomials are introduced in Section 3, where a new general algorithm for multiplication is also given. Section 4 describes the new LFSR-based multiplier architecture, gives an example of multiplication and analyses the theoretical complexity. Comparisons with other similar multipliers and hardware implementation results on Xilinx FPGAs are given in Section 5. Finally, conclusions are given in Section 6.

2 Background

Any element $A \in GF(2^m)$ can be represented in the polynomial basis $\{1, x, \ldots, x^{m-1}\}$ as $A = \sum_{i=0}^{m-1} a_i x^i$, with $a_i \in GF(2)$, where x is a root of the irreducible polynomial $f(y) = \sum_{i=0}^{m} f_i y^i$. Two-step classic PB multiplication in $GF(2^m)$ requires a polynomial multiplication followed by a reduction modulo the irreducible polynomial. Mastrovito proposed an efficient bit-parallel multiplication method in which a *product matrix* combines the above two steps together. In [7], a new $GF(2^m)$ PB multiplication method was given. In order to compute the product $C = A \cdot B$, this approach defined functions $\mathbf{S_i}$ ($1 \leq i \leq m$) and $\mathbf{T_i}$ ($0 \leq i \leq m-2$) given by the addition of terms $x_k = (a_k b_k)$ and $z_i^j = (a_i b_j + a_j b_i)$, with $a_i, b_i \in GF(2)$. These functions were given in [7] as $\mathbf{S_i} = x_p + \sum_{h=0}^{p-1} z_h^{i-h-1}$ and $\mathbf{T_i} = x_q + \sum_{j=1}^{r-(i+1)} z_{i+j}^{m-j}$, where $p = \lfloor i/2 \rfloor$, $q = (\lceil m/2 \rceil + \lfloor i/2 \rfloor)$, the term $x_p = a_p b_p$ only appears for i *odd* and x_q only appears for (m and i *even*) or for (m and i *odd*). In this case, $r = q$. Otherwise, i.e., for (m *even* and i *odd*) or for (m *odd* and i *even*), the term x_q does not appear and $r = (\lceil m/2 \rceil + \lceil i/2 \rceil)$. For example, using the above expressions, the terms $\mathbf{S_i}$ and $\mathbf{T_i}$ for $GF(2^6)$ are: $\mathbf{S_1} = x_0 = a_0 b_0$, $\mathbf{S_2} = z_0^1 = (a_0 b_1 + a_1 b_0)$, $\mathbf{S_3} = x_1 + z_0^2 = a_1 b_1 + (a_0 b_2 + a_2 b_0)$, $\mathbf{S_4} = z_0^3 + z_1^2 = (a_0 b_3 + a_3 b_0) + (a_1 b_2 + a_2 b_1)$, $\mathbf{S_5} = x_2 + z_0^4 + z_1^3 = a_2 b_2 + (a_0 b_4 + a_4 b_0) + (a_1 b_3 + a_3 b_1)$, $\mathbf{S_6} = z_0^5 + z_1^4 + z_2^3 = (a_0 b_5 + a_5 b_0) + (a_1 b_4 + a_4 b_1) + (a_2 b_3 + a_3 b_2)$, $\mathbf{T_0} = x_3 + z_1^5 + z_2^4 = a_3 b_3 + (a_1 b_5 + a_5 b_1) + (a_2 b_4 + a_4 b_2)$, $\mathbf{T_1}$

• *J.L. Imaña is with the Department of Computer Architecture and Automation, Faculty of Physics, Complutense University, 28040 Madrid, Spain. E-mail: jluimana@ucm.es*

TABLE 1
Coordinates c_i of the product for the trinomials $f(y) = y^6 + y^n + 1$, with $1 \leq n \leq 5$.

$n=$	1	2	3	4	5
c_0	$\mathbf{S_1}$ $\mathbf{T_0}$	$\mathbf{T_4}$	$\mathbf{T_3}$	$\mathbf{T_2}$ $\mathbf{T_4}$	$\mathbf{T_1}$ $\mathbf{T_2}$ $\mathbf{T_3}$ $\mathbf{T_4}$
c_1	$\mathbf{S_2}$ $\mathbf{T_1}$ $\mathbf{T_0}$	$\mathbf{T_4}$	$\mathbf{T_3}$	$\mathbf{T_2}$ $\mathbf{T_3}$ $\mathbf{T_4}$	
c_2	$\mathbf{S_3}$ $\mathbf{T_2}$ $\mathbf{T_1}$ $\mathbf{T_0}$ $\mathbf{T_4}$	$\mathbf{T_4}$	$\mathbf{T_3}$ $\mathbf{T_4}$		
c_3	$\mathbf{S_4}$ $\mathbf{T_3}$ $\mathbf{T_2}$ $\mathbf{T_1}$	$\mathbf{T_0}$ $\mathbf{T_3}$	$\mathbf{T_4}$		
c_4	$\mathbf{S_5}$ $\mathbf{T_4}$ $\mathbf{T_3}$ $\mathbf{T_2}$	$\mathbf{T_1}$ $\mathbf{T_4}$	$\mathbf{T_0}$ $\mathbf{T_2}$ $\mathbf{T_4}$		
c_5	$\mathbf{S_6}$ $\mathbf{T_4}$ $\mathbf{T_3}$	$\mathbf{T_2}$	$\mathbf{T_1}$ $\mathbf{T_3}$	$\mathbf{T_0}$ $\mathbf{T_1}$ $\mathbf{T_2}$ $\mathbf{T_3}$ $\mathbf{T_4}$	

= $z_2^5 + z_3^4 = (a_2b_5 + a_5b_2) + (a_3b_4 + a_4b_3)$, $\mathbf{T_2} = x_4 + z_3^5 = a_4b_4 + (a_3b_5 + a_5b_3)$, $\mathbf{T_3} = z_4^5 = (a_4b_5 + a_5b_4)$, $\mathbf{T_4} = x_5 = a_5b_5$. The product $C = A \cdot B$ can then be computed as the addition of these terms.

An LFSR is a shift register whose feedback value is a linear function of its previous state. It is used in many important applications, such as cryptography, pseudo-random numbers generation or test pattern generation. The product $P = A \cdot x \bmod f(x)$, where x is a root of the irreducible polynomial $f(y) = y^m + f_{m-1}y^{m-1} + \ldots + f_1y + 1$ and $A \in GF(2^m)$ generated by $f(y)$, can also be performed using an m-tap LFSR. The product $P = A \cdot x = (a_{m-1}x^{m-1} + \ldots a_1x + a_0)x = a_{m-1}x^m + \ldots a_1x^2 + a_0x$ can be computed using the fact that x is a root of $f(y)$, so $x^m = f_{m-1}x^{m-1} + \ldots + f_1x + 1$. Substituting this expression of x^m into the expression for $P = A \cdot x$ it is obtained $P = p_{m-1}x^{m-1} + \ldots p_1x + p_0$, with $p_0 = a_{m-1}$ and $p_i = a_{i-1} + a_{m-1}f_i$ for $i = 1 \ldots m-1$ [5]. This operation can be implemented with an m-tap LFSR, where the registers are initially loaded with the coordinates of the element A, $(a_0, \ldots, a_{m-1})$, and the coefficients f_i, $i = 1 \ldots m-1$, of the irreducible polynomial are connected to AND gates together with the output of the last 1-bit register of the LFSR. After m clock cycles, the registers contents will be the coefficients $(p_0, \ldots, p_{m-1})$ of $P = A \cdot x \bmod f(x)$ [5].

Using LFSR, a new multiplier was proposed in [11]. The new multiplier works in the truncated polynomial ring modular q, $\mathbb{Z}_q[x]/(x^m - 1)$, where $\mathbb{Z}_q[x]$ is polynomial with integer coefficients modulo q and all polynomials are taken modulo $x^m - 1$. The product $C = A \times B \bmod (x^m - 1)$, with $A = \sum_{i=0}^{m-1} a_i x^i$ and $B = \sum_{i=0}^{m-1} b_i x^i$ (with $a_i, b_i \in \mathbb{Z}_q$), is computed as shown in Figure 1, where $\oplus$ refers to an adder, $\boxtimes$ refers to a multiplier and a $\square$ stands for a register. In Figure 1, the registers are initially loaded with 0. The coefficients of A, $(a_0, \ldots, a_{m-1})$, are connected to the inputs of each multiplier in parallel, while that the coefficients of the operand B, $(b_0, \ldots, b_{m-1})$, input to all the multipliers in a serial fashion, starting with the less significant coefficient b_0. After m clock cycles, the registers will store the coefficients of the product $C = A \times B \bmod (x^m - 1)$ [11].

3 Irreducible Trinomials

For hardware implementation of $GF(2^m)$ multiplication, low Hamming weight irreducible polynomials, such as trinomials and pentanomials [1], [9], [10], are normally used. *Irreducible trinomials* [8] $f(y) = y^m + y^n + 1$ are important because they are abundant and, for a given m, an irreducible trinomial can be found when irreducible pentanomials do not exist. PB multiplication for irreducible trinomials was studied in [7], where different expressions for the coefficients of the product were given in terms of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions for specific trinomials $f(y) = y^m + y^n + 1$ with $n = m - 1, m/2, (m+1)/2, (m-1)/2$ and $n = 1$. For example, the coefficients of the product for $GF(2^6)$ using trinomials $f(y) = y^6 + y^n + 1$, with $1 \leq n \leq 5$ are given in Table 1, where the expression of a coefficient c_i is computed by the addition (XOR) of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms given in its row. The first column in Table 1 includes common terms that appear in the coefficients for every trinomial $f(y) = y^6 + y^n + 1$, with $1 \leq n \leq 5$. The columns $n = 1$, $n = 2$, $n = 3$, $n = 4$ and $n = 5$ include specific terms for the coefficients of the different trinomials. For example, the coefficient c_5 for the trinomial $f(y) = y^6 + y + 1$ ($n = 1$) is $c_5 = \mathbf{S_6} + \mathbf{T_4}$, for $n = 2$ is $c_5 = \mathbf{S_6} + \mathbf{T_3}$, for $n = 3$ is $c_5 = \mathbf{S_6} + \mathbf{T_2}$, for $n = 4$ is $c_5 = \mathbf{S_6} + \mathbf{T_1} + \mathbf{T_2}$, and for $n = 5$ is $c_5 = \mathbf{S_6} + \mathbf{T_0} + \mathbf{T_1} + \mathbf{T_2} + \mathbf{T_3} + \mathbf{T_4}$. The method presented in [7] was based on the introduction of a *product matrix* that can be decomposed in a sum of matrices depending on the generating irreducible polynomial selected for the field. For example, for the trinomial $f(y) = y^6 + y^3 + 1$, the product $C = A \cdot B$ can be computed as $\underline{c} = \mathbf{M} \cdot \underline{b}$, where $\underline{c} = (c_0, \ldots, c_5)^T$ and $\underline{b} = (b_0, \ldots, b_5)^T$ are the coefficients of C and B, respectively, and where $\mathbf{M}$ is a 6×6 matrix whose elements are additions of the coefficients a_i of the operand A. In this case, the matrix $\mathbf{M}$ can be decomposed as (1):

$$\mathbf{M} = \mathbf{M_0} + \mathbf{M_1} + \mathbf{M_2} = \tag{1}$$
$$\begin{pmatrix} a_0 & a_5 & a_4 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_5 & a_4 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_5 & a_4 & a_3 \\ a_3 & a_2 & a_1 & a_0 & a_5 & a_4 \\ a_4 & a_3 & a_2 & a_1 & a_0 & a_5 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & a_5 & a_4 \\ \cdot & \cdot & \cdot & \cdot & \cdot & a_5 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & a_5 & a_4 & a_3 & a_2 & a_1 \\ \cdot & \cdot & a_5 & a_4 & a_3 & a_2 \\ \cdot & \cdot & \cdot & a_5 & a_4 & a_3 \end{pmatrix} + \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & a_5 & a_4 \\ \cdot & \cdot & \cdot & \cdot & \cdot & a_4 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

It can be observed that the product $\mathbf{M_0} \cdot \underline{b}$ corresponds with the addition of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms given in the first column of Table 1, while that the products $\mathbf{M_1} \cdot \underline{b}$ and $\mathbf{M_2} \cdot \underline{b}$ correspond with the two subcolumns included in the column labeled with $n = 3$ in Table 1, respectively. It must also be noted that the circulant matrix $\mathbf{M_0}$ (equivalent to $\mathbf{K_0}$ matrix in [7]) always appears in the decomposition of $\mathbf{M}$ for any field $GF(2^m)$, so the product $\mathbf{M_0} \cdot \underline{b}$ (and therefore the addition of the corresponding $\mathbf{S_i}$ and $\mathbf{T_i}$ terms) is common to any irreducible polynomial selected for $GF(2^m)$.

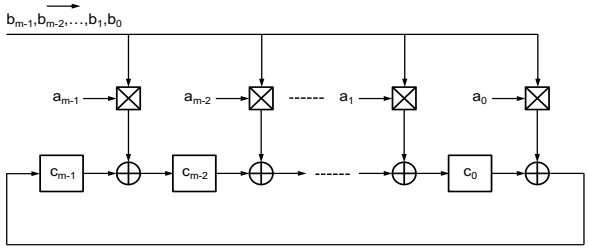

Fig. 1. LFSR for the implementation of $A \times B$ mod $(x^m - 1)$ [11].

3.1 New General Expressions for the Multiplier

The expressions given in [7] for $GF(2^m)$ PB multiplication are only valid for the five specific trinomials $f(y) = y^m +$

$y^n + 1$ with $n = m-1, m/2, (m+1)/2, (m-1)/2$ and $n = 1$. However, using Table 1, a new general multiplication algorithm for trinomials valid for any $1 \leq n \leq m-1$ can be given. Figure 2 shows these new expressions, where the coefficients of the product can be computed by the addition of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms. In Figure 2, the parameters $z = m-n$, $\delta = \lfloor \frac{m-2-i}{z} \rfloor$ and $\gamma = \lfloor \frac{m-2-i+n}{z} \rfloor$.

```
for i = 0 ... n − 2:
    c_i = (S_{i+1} + T_i) + Σ_{h=1}^{δ} T_{i+h·z};
c_{n−1} = (S_n + T_{n−1});
for i = n ... m − 1:
    if i = m − 1 then:
        c_i = (S_{i+1}) + Σ_{h=0}^{γ} T_{i−n+h·z};
    else:
        c_i = (S_{i+1} + T_i) + Σ_{h=0}^{γ} T_{i−n+h·z};
```

Fig. 2. New general multiplication algorithm for $1 \leq n \leq m-1$.

3.1.1 Proof of the new multiplication algorithm

The new algorithm given in Figure 2 can be deduced from [7] and from Table 1. From [7], the matrix $\mathbf{M}$ is decomposed as $\mathbf{M} = \mathbf{M_0} + \sum_{i=1}^{\tau} \mathbf{M_i}$, with $\tau = \lceil \frac{m-1}{z} \rceil$, in such a way that τ determines the number of subcolumns included in the columns for a given n in Table 1. For a given trinomial $f(y) = y^m + y^n + 1$, the coefficients c_i of the product include the addition $(\mathbf{S_{i+1}} + \mathbf{T_i})$ corresponding with $\mathbf{M_0} \cdot \underline{b}$. It can be observed that the coefficient c_{n-1} only includes this addition $(\mathbf{S_n} + \mathbf{T_{n-1}})$, while that the remaining coefficients also include the sum of additional $\mathbf{T_j}$ terms corresponding with the products $\mathbf{M_i} \cdot \underline{b}$, with $i = 1 \ldots \tau$. From [7], the first additional $\mathbf{T_j}$ terms (first subcolumn for a given n in Figure 2) for the coefficients c_i are $\mathbf{T_{i+z}}$ for $0 \leq i \leq n-2$ and $\mathbf{T_{i-n}}$ for $n \leq i \leq m-1$. These terms correspond with $\mathbf{M_1} \cdot \underline{b}$. It can be observed that if additional subcolumns (corresponding with additional products $\mathbf{M_h} \cdot \underline{b}$, with $h = 2, 3, \ldots$) are included, then additional terms $\mathbf{T_{i+hz}}$ ($0 \leq i \leq n-2$) and $\mathbf{T_{i-n+hz}}$ ($n \leq i \leq m-1$) are added to the coefficients c_i. As the maximum available term $\mathbf{T_i}$ for a given field $GF(2^m)$ is $\mathbf{T_{m-2}}$, then the maximum additional term $\mathbf{T_{i+hz}}$ (for $0 \leq i \leq n-2$) should be $i + hz \leq m-2$, so the maximum value of h should be $h = \lfloor \frac{m-2-i}{z} \rfloor$, corresponding with the value δ in Figure 2. Similarly, the maximum additional term $\mathbf{T_{i-n+hz}}$ (for $n \leq i \leq m-1$) should be $i - n + hz \leq m-2$, so the maximum value of h should be $h = \lfloor \frac{m-2-i+n}{z} \rfloor$, corresponding with the value γ in Figure 2. Furthermore, the coefficient with the maximum number of additional $\mathbf{T_j}$ terms is c_n [7], so the maximum value of γ (for $i = n$) will be $\gamma = \lfloor \frac{m-2}{z} \rfloor$. As the addition of terms given in Figure 2 ranges from 0 to γ, then the number of additional terms $\mathbf{T_j}$ (and therefore the maximum number of additional matrices $\mathbf{M_i}$) will be $\gamma + 1 = \lfloor \frac{m-2}{z} \rfloor + 1 = \lceil \frac{m-1}{z} \rceil$, therefore matching the value τ given in [7].

4 New LFSR-based Multiplier Architecture for Irreducible Trinomials

The LFSR multiplication approach [11] given in Section 2 can be easily adapted to the computation of the product $C = A \cdot B \bmod (x^m + 1)$, where $A = \sum_{i=0}^{m-1} a_i x^i$ and $B = \sum_{i=0}^{m-1} b_i x^i$, with $a_i, b_i \in GF(2)$. In this case, the same architecture given in Figure 1 can be used to perform the product $C = A \cdot B \bmod (x^m + 1)$ if $\oplus$ refers to XORs, $\boxtimes$ refers AND gate and $\square$ stands for 1-bit registers. For example, the product $C = A \cdot B \bmod (x^6 + 1)$ can be implemented with Figure 1 using the fact that x is a root of the polynomial $f(y) = y^6 + 1$, so $x^6 = 1$, $x^7 = x$, $x^8 = x^2$, $x^9 = x^3$ and $x^{10} = x^4$. Substituting these expressions into those obtained from the product of polynomials $(a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x^1 + a_0) \cdot (b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0)$ we get $c_0 = a_0b_0 + a_5b_1 + a_4b_2 + a_3b_3 + a_2b_4 + a_1b_5$, $c_1 = a_1b_0 + a_0b_1 + a_5b_2 + a_4b_3 + a_3b_4 + a_2b_5$, $c_2 = a_2b_0 + a_1b_1 + a_0b_2 + a_5b_3 + a_4b_4 + a_3b_5$, $c_3 = a_3b_0 + a_2b_1 + a_1b_2 + a_0b_3 + a_5b_4 + a_4b_5$, $c_4 = a_4b_0 + a_3b_1 + a_2b_2 + a_1b_3 + a_0b_4 + a_5b_5$ and $c_5 = a_5b_0 + a_4b_1 + a_3b_2 + a_2b_3 + a_1b_4 + a_0b_5$. It can be observed that after 6 clock cycles, the 1-bit registers c_0, c_1, c_2, c_3, c_4 and c_5 in Figure 1 contain these expressions of the coefficients of the product. In the above example, it is important to note that the expressions of the coefficients c_i, $i = 0 \ldots 5$, correspond with the addition of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms given in Section 2 for $GF(2^6)$ using the approach in [7]. In this case, the coefficients can be written as $c_0 = \mathbf{S_1} + \mathbf{T_0}$, $c_1 = \mathbf{S_2} + \mathbf{T_1}$, $c_2 = \mathbf{S_3} + \mathbf{T_2}$, $c_3 = \mathbf{S_4} + \mathbf{T_3}$, $c_4 = \mathbf{S_5} + \mathbf{T_4}$ and $c_5 = \mathbf{S_6}$. In fact, these coefficients correspond with the result of the product $\mathbf{M_0} \cdot \underline{b}$ as given in equation (1). This product was stated in Section 3 to be common to any irreducible polynomial selected for $GF(2^6)$. Furthermore, the content of the register c_5 in this example is $\mathbf{S_1}$ in the first clock cycle of the computation, $\mathbf{S_2}$ in the second cycle, $\mathbf{S_3}$ in the third one, $\mathbf{S_4}$ in the fourth cycle, $\mathbf{S_5}$ in the fifth one and $\mathbf{S_6}$ in the sixth clock cycle. As conclusion, the LFSR architecture given in Figure 1 can be used for the computation of the product $\mathbf{M_0} \cdot \underline{b}$ common to any polynomial selected for $GF(2^m)$, where the 1-bit registers c_i store the coefficients of $\mathbf{M_0} \cdot \underline{b}$ and where the register c_{m-1} contains the term $\mathbf{S_i}$ in the i-th clock cycle, $i = 1 \ldots m-1$.

The method given in [7] for $GF(2^m)$ multiplication using irreducible trinomials computes the coefficients of the product by the addition of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms as given in Table 1 and Figure 2. It can be observed that different terms $\mathbf{T_i}$ are shared among the coefficients of the product while that $\mathbf{S_i}$ terms are only used in the c_{i-1} coefficients, for $i = 1 \ldots m$. Therefore, it would be useful that we could access to individual $\mathbf{T_i}$ terms in order to perform the product implementation. To do that, a modification of the LFSR architecture given in Figure 1 can be proposed in such a way that we can compute the matrix-vector product $\mathbf{M_0} \cdot \underline{b}$ and that the register c_{m-1} can contain the term $\mathbf{T_{m-1-i}}$ in the i-th clock cycle, $i = 1 \ldots m-1$. In the new proposed architecture, the coefficients of A, $(a_0, a_1, \ldots, a_{m-1})$, are connected to the inputs of each AND gate in reverse order than in Figure 1, and the coefficients of the operand B, $(b_0, b_1, \ldots, b_{m-1})$, input to all the AND gates in a serial fashion starting with the most significant coefficient b_{m-1}. Furthermore, the 1-bit registers (from left to right) are $c_{m-1}, c_0, c_1, \ldots, c_{m-2}$ that are in a different order than those in Figure 2. After m clock cycles, these registers will store the coefficients of $\mathbf{M_0} \cdot \underline{b}$ and the 1-bit register c_{m-1} will contain the term $\mathbf{T_{m-1-i}}$ in the i-th clock cycle, with $i = 1 \ldots m-1$.

For specific case $GF(2^6)$, Table 2 shows the evolution in time of the contents of the 1-bit registers $(c_0, c_1, c_2, c_3, c_4, c_5)$ for the new proposed architecture given above, where *Cycle*

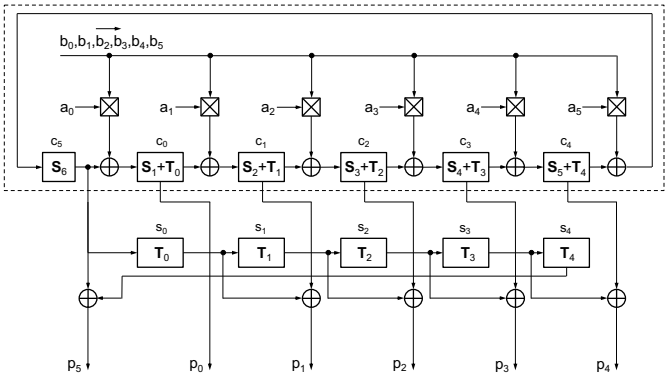

Fig. 3. New LFSR multiplier architecture for trinomial $f(y) = y^6 + y + 1$.

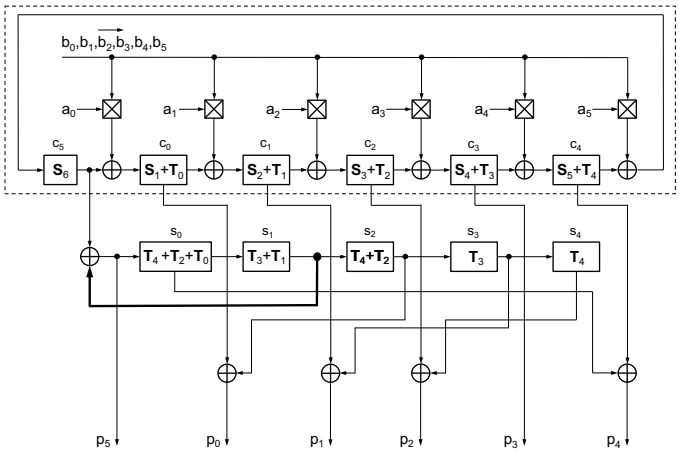

Fig. 4. New LFSR multiplier architecture for trinomial $f(y) = y^6 + y^4 + 1$.

denotes the clock cycles and *Serial* denotes the coefficient b_i that is valid in the serial input for each clock cycle. Coefficients $(a_0, a_1, a_2, a_3, a_4, a_5)$ are valid in parallel through all the process. In Table 2, during initialization (*Init.*) the most significant bit b_5 of operand B is valid in serial input and registers c_i, $i = 0 \ldots 5$ are loaded with 0. In first cycle t_1, the products a_5b_5, a_0b_5, a_1b_5, a_2b_5, a_3b_5 and a_4b_5 performed during initialization are loaded into c_5, c_0, c_1, c_2, c_3 and c_4, respectively. It must be noted that the term a_5b_5 loaded into c_5 correspond with the term $\mathbf{T_4}$ in $GF(2^6)$. In t_1, the bit b_4 of B is valid in serial input and multiplied (AND) with the coefficients of A. This products will be added (XOR) with the contents of the 1-bit registers in t_1 and loaded into registers in the second cycle t_2, and so on. It can be observed that after 6 clock cycles (in t_6), the registers contains the coefficients of $\mathbf{M_0} \cdot \underline{b}$. Furthermore, the contents of c_5 in cycles t_1, t_2, t_3, t_4 and t_5 (gray shadowed cells in Table 2) correspond with $\mathbf{T_4}$, $\mathbf{T_3}$, $\mathbf{T_2}$, $\mathbf{T_1}$ and $\mathbf{T_0}$, respectively.

4.1 New LFSR Multiplier Architecture

The new previously proposed architecture can be used to compute the product $P = A \cdot B$ over $GF(2^m)$ generated by irreducible trinomials. As shown in Table 1 and Figure 2, the product for trinomials can be computed with the addition of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms [7]. It has been previously stated that the new proposed architecture computes the coefficients of the product $\mathbf{M_0} \cdot \underline{b}$ (i.e., the addition of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the first column of Table 1) that is common to any irreducible polynomial selected for a given field size m, and all the terms $\mathbf{T_i}$, $0 \leq i \leq m-2$, that are successively stored in register c_{m-1} in the first $m-1$ cycles of the product computation. Therefore, if the contents of the 1-bit register c_{m-1} are loaded into a shift register, then the $\mathbf{T_i}$ terms can be accessed and XORed with the corresponding coefficients of $\mathbf{M_0} \cdot \underline{b}$ depending on the selected irreducible trinomial.

4.1.1 Multiplication for $f(y) = y^6 + y + 1$

For example, let us consider the multiplication for $GF(2^6)$ using the trinomial $f(y) = y^6 + y + 1$ given in Table 1 (column $n = 1$). Figure 3 shows the new multiplier architecture. At the top of Figure 3, the architecture for the computation of $\mathbf{M_0} \cdot \underline{b}$ is represented into a dotted rectangle where the 1-bit registers $(c_0, c_1, c_2, c_3, c_4, c_5)$ include their final content after the six clock cycles (cycle t_6 in Table 2). At the bottom of Figure 3, a shift register $(s_0, s_1, s_2, s_3, s_4)$ loads the successive contents of register c_5. Therefore, after six clock cycles, the contents of s_0, s_1, s_2, s_3 and s_4 are $\mathbf{T_0}, \mathbf{T_1}, \mathbf{T_2}, \mathbf{T_3}$ and $\mathbf{T_4}$, respectively. In Table 3, the evolution in time of the contents of registers c_5 and s_i, $0 \leq i \leq 4$, is given. In order to perform the multiplication, a final step of addition of the contents of registers $(c_0, c_1, c_2, c_3, c_4, c_5)$ and $(s_0, s_1, s_2, s_3, s_4)$ must be done in order to compute the product $P = A \cdot B$. It must be noted that in this case, only one level of 2-input XOR gates is needed for this final step.

4.1.2 Multiplication for $f(y) = y^m + y^n + 1$

The computation of the product for $f(y) = y^6 + y + 1$ (and, in general for trinomials $f(y) = y^m + y^n + 1$, with $n = 1$) presents the simplest architecture because only one addition of $\mathbf{T_i}$ terms must be done with the results of the product $\mathbf{M_0} \cdot \underline{b}$. This fact can be observed in Table 1, where only one column with $\mathbf{T_i}$ terms appears for $n = 1$. However, for trinomials $f(y) = y^m + y^n + 1$, with $n > 1$, more subcolumns appear for each value of n. For $GF(2^6)$, Table 1 shows 2, 2, 3 and 5 subcolumns for $n = 2, 3, 4$ and 5, respectively. The appearance of h subcolumns in Table 1 implies, at least, one addition of h $\mathbf{T_i}$ terms and, therefore, the use of $\lceil log_2(h+1) \rceil$ levels of 2-input XOR gates in the final step of the multiplication process. In order to reduce the number of XOR levels in the final step of the product computation, a modification of the shift register used to store the $\mathbf{T_i}$ terms provided by the register c_{m-1} can be done. It can be observed in Table 1 that for $n > 1$, the addition of at least two $\mathbf{T_i}$ terms is needed for the computation. If a XOR gate is included in the input to the first register s_0 of the shift register used to store the $\mathbf{T_i}$ terms and some specific feedback to this XOR is done from a given s_i of the shift register, then the addition of specific $\mathbf{T_i}$ terms can be done.

For example, let us consider the multiplication for $GF(2^6)$ using the trinomial $f(y) = y^6 + y^4 + 1$ given in Table 1 (column $n = 4$). Figure 4 shows the new multiplier architecture where a XOR gate has been included to the input of the 1-bit register s_0 of the bottom shift register. One of the inputs is the output of c_5 register, while that the other one comes from the second register s_1 of the shift register. Using this architecture, Table 4 shows the evolution

TABLE 2
Computation of $\mathbf{M_0} \cdot \underline{b}$ for $GF(2^6)$ using the new LFSR architecture.

Cycle	Serial	c_5	c_0	c_1	c_2	c_3	c_4
Init.	b_5	0	0	0	0	0	0
t_1	b_4	a_5b_5	a_0b_5	a_1b_5	a_2b_5	a_3b_5	a_4b_5
t_2	b_3	$a_4b_5 + a_5b_4$	$a_5b_5 + a_0b_4$	$a_0b_5 + a_1b_4$	$a_1b_5 + a_2b_4$	$a_2b_5 + a_3b_4$	$a_3b_5 + a_4b_4$
t_3	b_2	$a_3b_5 + a_4b_4 + a_5b_3$	$a_4b_5 + a_5b_4 + a_0b_3$	$a_5b_5 + a_0b_4 + a_1b_3$	$a_0b_5 + a_1b_4 + a_2b_3$	$a_1b_5 + a_2b_4 + a_3b_3$	$a_2b_5 + a_3b_4 + a_4b_3$
t_4	b_1	$a_2b_5 + a_3b_4 + a_4b_3 + a_5b_2$	$a_3b_5 + a_4b_4 + a_5b_3 + a_0b_2$	$a_4b_5 + a_5b_4 + a_0b_3 + a_1b_2$	$a_5b_5 + a_0b_4 + a_1b_3 + a_2b_2$	$a_0b_5 + a_1b_4 + a_2b_3 + a_3b_2$	$a_1b_5 + a_2b_4 + a_3b_3 + a_4b_2$
t_5	b_0	$a_1b_5 + a_2b_4 + a_3b_3 + a_4b_2 + a_5b_1$	$a_2b_5 + a_3b_4 + a_4b_3 + a_5b_2 + a_0b_1$	$a_3b_5 + a_4b_4 + a_5b_3 + a_0b_2 + a_1b_1$	$a_4b_5 + a_5b_4 + a_0b_3 + a_1b_2 + a_2b_1$	$a_5b_5 + a_0b_4 + a_1b_3 + a_2b_2 + a_3b_1$	$a_0b_5 + a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1$
t_6	$-$	$a_0b_5 + a_1b_4 + a_2b_3 + a_3b_2 + a_4b_1 + a_5b_0$	$a_1b_5 + a_2b_4 + a_3b_3 + a_4b_2 + a_5b_1 + a_0b_0$	$a_2b_5 + a_3b_4 + a_4b_3 + a_5b_2 + a_0b_1 + a_1b_0$	$a_3b_5 + a_4b_4 + a_5b_3 + a_0b_2 + a_1b_1 + a_2b_0$	$a_4b_5 + a_5b_4 + a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0$	$a_5b_5 + a_0b_4 + a_1b_3 + a_2b_2 + a_3b_1 + a_4b_0$

TABLE 3
Computation of $A \cdot B$ over $GF(2^6)$ for $f(y) = y^6 + y + 1$ using the new LFSR architecture.

Cycle	c_5	s_0	s_1	s_2	s_3	s_4
Init.	0	0	0	0	0	0
t_1	$\mathbf{T_4}$	0	0	0	0	0
t_2	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0	0	0
t_3	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0	0
t_4	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0
t_5	$\mathbf{T_0}$	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0
t_6	$\mathbf{S_6}$	$\mathbf{T_0}$	$\mathbf{T_1}$	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$

TABLE 4
Computation of $A \cdot B$ over $GF(2^6)$ for $f(y) = y^6 + y^4 + 1$ using the new LFSR architecture.

Cycle	c_5	s_0	s_1	s_2	s_3	s_4
Init.	0	0	0	0	0	0
t_1	$\mathbf{T_4}$	0	0	0	0	0
t_2	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0	0	0
t_3	$\mathbf{T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0	0
t_4	$\mathbf{T_1}$	$\mathbf{T_4 + T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0	0
t_5	$\mathbf{T_0}$	$\mathbf{T_3 + T_1}$	$\mathbf{T_4 + T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$	0
t_6	$\mathbf{S_6}$	$\mathbf{T_4 + T_2 + T_0}$	$\mathbf{T_3 + T_1}$	$\mathbf{T_4 + T_2}$	$\mathbf{T_3}$	$\mathbf{T_4}$

in time of the contents of registers c_5 and s_i, $0 \leq i \leq 4$, where the contents of c_5 are shifted to the right until cycle t_3. Due to the feedback from the 1-bit register s_1 to the XOR in the input of s_0, the XOR of $\mathbf{T_2}$ (content of c_5 in the previous cycle t_3) and $\mathbf{T_4}$ (content of s_1 in t_3) is performed and loaded into s_0 in the next clock cycle t_4. In t_4, the previous contents of s_0 and s_1 are also loaded into s_1 and s_2, respectively. Similar situation occurs in t_5, where the new value $\mathbf{T_3} + \mathbf{T_1}$ is loaded into s_0 and the values $\mathbf{T_4} + \mathbf{T_2}$, $\mathbf{T_3}$ and $\mathbf{T_4}$ are shifted to s_1, s_2 and s_3, respectively. After six clock cycles (t_6), the values $\mathbf{T_4} + \mathbf{T_2} + \mathbf{T_0}$, $\mathbf{T_3} + \mathbf{T_1}$, $\mathbf{T_4} + \mathbf{T_2}$, $\mathbf{T_3}$ and $\mathbf{T_4}$ are loaded into s_0, s_1, s_2,s_3 and s_4, respectively. These final contents of the registers are shown in Figure 4. It can be observed that these values match with the additions of $\mathbf{T_i}$ terms given in Table 1 needed for the computation of the product $P = A \cdot B$ over $GF(2^6)$ when the trinomial $f(y) = y^6 + y^4 + 1$ is used. A final level of XOR gates performing the sums of the corresponding c_i ($i = 0 \ldots 4$) and s_i ($i = 0, 2, 3, 4$) registers completes the implementation of the product. After six clock cycles, the sum $c_5 \oplus s_1$ provides the most significant bit of the product. This sum feeds register s_0 and is already implemented, so no additional XOR is needed at the final level of XOR gates.

The above approach can be used to implement the $GF(2^m)$ multiplication when a trinomial is used. Figure 5 shows the architecture of the shift registers for the five trinomials $f(y) = y^6 + y^n + 1$, with $1 \leq n \leq 5$, where the final contents of the registers after six clock cycles are also included. At the top of Figure 5, the computation of $\mathbf{M_0} \cdot \underline{b}$ is represented into a dotted rectangle. At the bottom, the shift registers $(s_0, s_1, s_2, s_3, s_4)$ for the five trinomials are shown. It can be observed that for each trinomial $y^6 + y^n + 1$, the feedback from the register $s_{m-1-n} = s_{z-1}$ to the XOR in the input of the 1-bit register s_0 must be done. In fact, this feedback is also done for $f(y) = y^6 + y + 1$ (represented in Figure 5 with a dotted XOR and dotted feedback), although it is not applied because only six clock cycles are needed for the computation. For this reason, the input XOR and feedback can be removed for this trinomial. Figure 5 does not show the final step with one level of XOR gates performing the addition of the corresponding c_i ($i = 0 \ldots 5$) and s_i ($i = 0 \ldots 4$) registers in order to complete the multiplication.

4.1.3 Proof of the new architecture

It can be proven that the contents of s_i registers in Figure 5 match the terms that must be added in the coefficients c_i of the product given in Table 1 and in the new algorithm shown in Figure 2. From Table 1, single pairs of additions $(\mathbf{T_{m-i}} + \mathbf{T_{n-i}})$, $i = 2, 3, \ldots$ appear (when at least two subcolumns are included for a given n) as long as no repeated $\mathbf{T_j}$ terms are included in these pairs. For example, in Table 1 for $n = 4$, single pairs $(\mathbf{T_4}+\mathbf{T_2})$ and $(\mathbf{T_3}+\mathbf{T_1})$ are given for c_0 and c_5, respectively, but no single pair $(\mathbf{T_2}+\mathbf{T_0})$ is given ($\mathbf{T_2}$ is already included in the first pair). In this case, the addition of terms in c_4 comes from the addition of the single pair $(\mathbf{T_4} + \mathbf{T_2})$ and $\mathbf{T_0}$. From Table 4 and Figure 5, terms $\mathbf{T_{m-2}}$ and $\mathbf{T_{n-2}}$ are loaded into registers s_{z-1} and c_{m-1}, respectively, in the clock cycle t_{z+1}. Due to the feedback from register s_{z-1} to the XOR in the input of the register s_0, in cycle t_{z+2} the single pair $(\mathbf{T_{m-2}} + \mathbf{T_{n-2}})$ is loaded into s_0. The content of s_0 is shifted to s_1 in cycle t_{z+2}, when next pair $(\mathbf{T_{m-3}} + \mathbf{T_{n-3}})$ is loaded into s_0, and so on. After m clock cycles, the contents of s_i registers match with the additions of $\mathbf{T_i}$ terms given in Figure 2 and Table 1 (for $m = 6$) needed for the computation of the product $P = A \cdot B$ over $GF(2^m)$ for trinomials $f(y) = y^m + y^n + 1$.

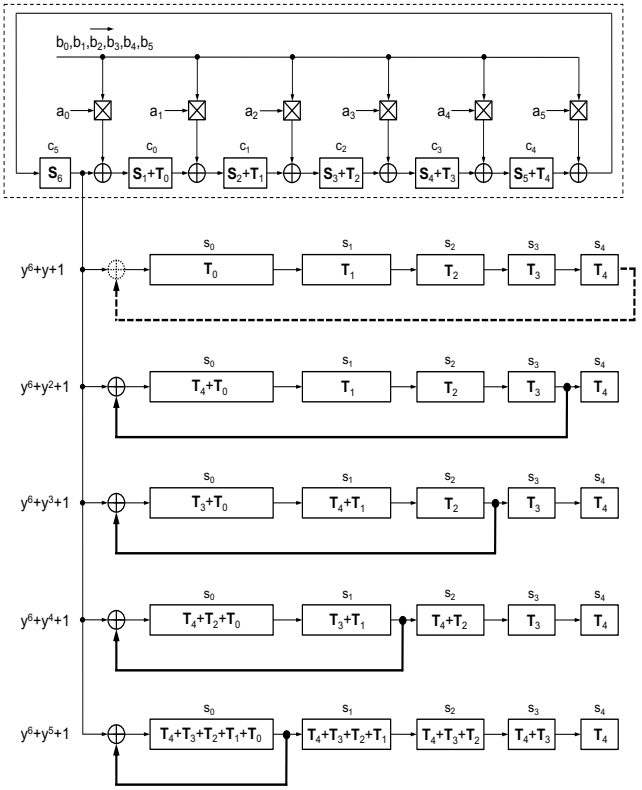

Fig. 5. Shift registers for trinomials $f(y) = y^6 + y^n + 1$, $1 \leq n \leq 5$.

4.1.4 Parallel multiplication

A characteristic of the proposed architecture is that it can perform the multiplication of two operands for n different fields $GF(2^m)$ using n trinomials simultaneously in m clock cycles only including $n(m-1)$ 1-bit registers and nm XOR gates. For example, Figure 5 would perform the parallel computation of $P = A \cdot B$ over $GF(2^6)$ for the five trinomials $f(y) = y^6 + y^n + 1$, with $1 \leq n \leq 5$, if the final step of XOR gates performing the addition of the corresponding c_i ($i = 0 \ldots 5$) and s_i ($i = 0 \ldots 4$) registers for each trinomial is included. The general architecture of the $GF(2^m)$ multiplier for an irreducible trinomial $f(y) = y^m + y^n + 1$, with $1 \leq n < m$, is given in Figure 6. Parallel multiplication of two operands for different irreducible polynomials could be applied in cryptography. In order to hinder cryptanalytic attacks, cryptosystems working over different irreducible polynomials could be proposed in such a way that different polynomials could be selected for different messages without modifying the $GF(2^m)$ arithmetic hardware [12], [13]. Pipelined architectures could also use simultaneous multiplication in such a way that different stages could use different polynomials, fulfilling that a given message uses the same irreducible polynomial through the pipeline.

4.2 Complexity Analysis

As shown in Figure 6, the three main modules of the new LFSR-based $GF(2^m)$ multiplier architecture for irreducible trinomials $f(y) = y^m + y^n + 1$, with $1 \leq n < m$, are the module for the computation of $\mathbf{M_0} \cdot \underline{b}$, the shift register for the loading/addition of $\mathbf{T_i}$ terms and the module that performs the XOR of the outputs of the above two modules.

The computation of $\mathbf{M_0} \cdot \underline{b}$ requires m AND gates, m XOR gates and m flipflops (1-bit registers). Furthermore, the critical path delay is $T_A + T_X$. The shift register for the loading/addition of $\mathbf{T_i}$ terms (see Figure 5) requires $m-1$ flipflops and one XOR gate when $2 \leq n < m$ (for $n = 1$ no XORs are needed). The critical path delay in this case is T_X. Finally, the module that performs the XOR of the outputs of the above two modules only requires $m-2$ XOR gates in parallel when $2 \leq n < m$ or $m-1$ XOR gates in parallel for $n = 1$, so the critical path delay is also T_X. Therefore, the overall area complexity of the new LFSR-based multiplier is m AND gates, $2m-1$ flipflops and $2m-1$ XOR gates. With respect to time complexity, the product is computed in m clock cycles with a period determined by the critical path delay. The critical propagation delay of the multiplier is given by the maximum path delay of the above three modules, i.e., $\max\{T_A + T_X, T_X\} = T_A + T_X$. This is a constant propagation delay, independent of the value of n. Furthermore, a characteristic of the proposed architecture is that it can perform the product of two operands for t different fields $GF(2^m)$ generated by t irreducible trinomials simultaneously in m clock cycles with the inclusion of $t(m-1)$ flipflops and tm XOR gates (see Figure 5).

5 Comparison with other multipliers and hardware implementations

In Table 5, the theoretical complexity of our proposed multiplier is compared with several $GF(2^m)$ bit-serial multipliers found in the literature. In this table, the number of 2-input AND, XOR and OR gates, the number of 1-bit registers, 2:1 MUXs, the number of clock cycles needed to complete the multiplication and the critical path delay are given. Furthermore, T_{N}, T_{O}, T_{M} and T_{NAND} represent the delay of an inverter, 2-input OR gate, 2:1 MUX and 2-input NAND gate, respectively. In [14], Least-significant (LSB) and Most-significant (MSB) bit-serial multipliers for general irreducible polynomials were given. Versatile $GF(2^m)$ bit-serial multipliers (that can also perform multiplications in all underlying $GF(2^n)$ fields with $1 \leq n \leq m$) for general irreducible polynomials were also given in [15] and [16]. In [17], [20] and [22], bit-serial multipliers for irreducible trinomials were proposed. A bit-serial multiplier for general irreducible polynomials was also given in [21]. In [18] and [19], bit-serial multipliers for all-one-polynomials (AOPs) were also presented. The bit-serial multiplier for irreducible trinomials here proposed is represented as $Prop$ in Table 5.

In order to highlight the differences between the various multiplier designs, specific results for $GF(2^{233})$ generated by the NIST irreducible trinomial $f(y) = y^{233} + y^{74} + 1$ are given in Table 6 ($m = 233$, $n = 74$). From the results, it can be observed that our proposed multiplier presents, together with the LSB/MSB bit-serial multipliers in [14], the lowest propagation delay. Regarding the number of clock cycles needed to perform the product, our multiplier needs m clock cycles. The multipliers in [16], [21] require less clock cycles, although both present highest propagation delays. With respect to area complexity, our multiplier matches the lowest number of AND gates in [14], although needs $2m-1$ XOR gates versus m gates given in [18], [19]. The multiplier here proposed also presents the lowest number of 1-bit

TABLE 5
Complexities of bit-serial PB multipliers.

	#AND	#XOR	#OR	#FF	#Mux	#Clk	Delay
LSB [14]	m	$m+1$	0	$2m$	m	m	$T_{\rm A}+T_{\rm X}$
MSB [14]	m	$m+1$	0	$2m$	m	m	$T_{\rm A}+T_{\rm X}$
[15]	$3m$	$2m$	0	m^2	m	$3m$	$T_{\rm X}+\lceil log_2(m)\rceil(T_{\rm A}+T_{\rm N}+T_{\rm O})$
[16]	$3m$	$2m$	0	$4m+2$	$2m$	$(0.664)m$	$2T_{\rm X}+T_{\rm M}$
[17]	$2m-1$	$2m+n-2$	0	$3m+n-1$	0	m	$T_{\rm A}+(2+\lceil log_2(m)\rceil)T_{\rm X}$
[18]	$m+1$	m	0	$2m+2$	$m+2$	$2m$	$T_{\rm A}+mT_{\rm X}$
[18]	$2m-1$	$2m-2$	0	$2m+2$	$m+2$	$2m-1$	$T_{\rm A}+(1+\lceil log_2(m-1)\rceil)T_{\rm X}$
[19]	$m+1$	m	0	$2m+2$	2	$2m-1$	$T_{\rm A}+mT_{\rm X}$
[20]	$m^{2\dagger}$	m^2-1	0	$2m^2-2m$	0	m	$T_{\rm NAND}+T_{\rm X}$
[21]	$\frac{m^2+m}{2}$	$\frac{m^2+m}{2}$	0	$5m-1$	$4m$	$2n+1$	$T_{\rm A}+\lceil log_2 m\rceil T_{\rm X}+2T_{\rm M}$
[22]	$2m-1$	$2m+1$	0	$3m-2$	0	m	$\max\{3T_{\rm X}, T_{\rm A}+2T_{\rm X}\}^{\ddagger}$
Prop.	m	$2m-1$	0	$2m-1$	0	m	$T_{\rm A}+T_{\rm X}$

† NAND, ‡ delay $T_{\rm A}+2T_{\rm X}$ for $n\neq m-1$ and $\max\{3T_{\rm X}, T_{\rm A}+2T_{\rm X}\}$ for $n=m-1$.

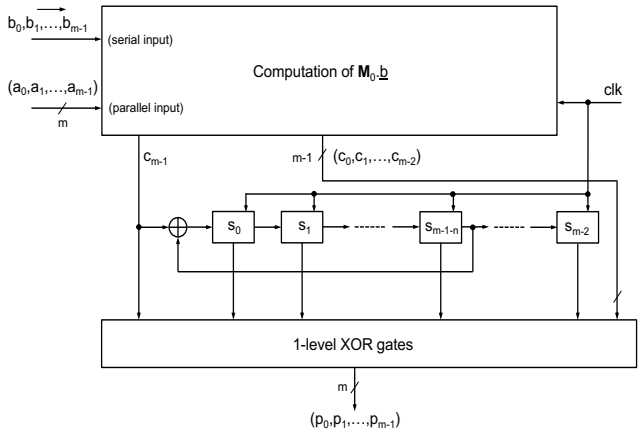

Fig. 6. LFSR-based $GF(2^m)$ multiplier architecture for trinomials.

registers ($2m-1$ versus $2m$ given in [14]) and it does not use multiplexers. In Table 6, the area (transistors count) and time estimates for the NIST $GF(2^{233})$ multipliers are also compared. In order to fairly compare time complexities, typical propagation-delays t_{PD} of the following STMicroelectronics real circuits have been used: M74HC08 (AND gate, t_{PD} = 6 ns.), M74HC86 (XOR, t_{PD} = 12 ns.), M74HC32 (OR, t_{PD} = 8 ns.), M74HC257 (MUX, t_{PD} = 11 ns.), M74HC04 (INV, t_{PD} = 8 ns.) and M74HC00 (NAND, t_{PD} = 6 ns.). In order to estimate the number of CMOS transistors used, the traditional counts (6 transistors for AND gate, 6 for XOR, 6 for OR, 8 for SR-latch, 6 for 2:1 MUX and 4 for NAND) have been used. In Table 6, the *Time* needed to perform the multiplication (given in nanoseconds) is computed as the product of the critical path *Delay* by the number of clock cycles used to complete the multiplication, and the *Area*×*Time* column is given in *transistors* × *miliseconds*. It must be noted that some of the results in Table 6 correspond to multipliers for general irreducible polynomials and AOPs. Although the comparison of different architectures of multipliers can not be fair, it can be observed that the proposed multiplier slightly reduces the best area complexity given in [14] and matches the best time complexities (critical path delay and time needed to perform multiplication) given in [14] and [20]. With respect to the area×time metric, our multiplier slightly reduces the best values obtained in [14].

5.1 FPGA implementations

In order to further compare the work here presented with other bit-serial multipliers, FPGA implementations of the proposed multiplier and the best one given in [14] have been performed for NIST $GF(2^{233})$ generated by irreducible trinomials (m = 233, n = 74) and (m = 233, n = 159). These multipliers have been described in VHDL, synthesized and implemented on Xilinx FPGA Artix-7 XC7A200T-FFG1156. Experimental results are those reported by Xilinx ISE 14.7 using XST synthesizer for *speed high* optimizations. Experimental post-place and route results are given in Table 7 for the multiplier here proposed ($Prop.$) and for LSB bit-serial multiplier given in [14], where $Slices$ gives the used number of slices, $Per.$ represents the minimum clock period (in nanoseconds), $Time$ is the time needed (in nanoseconds) for performing the multiplication after 233 clock cycles ($Time = Per.\times 233$), and $A\times T$ expresses area by time delay in $Slices \times ns$ to compare the area and delay (less is better). In Table 7, experimental results obtained by the proposed multiplier and the best multiplier [14] found in Table 5 are given for single multiplications with NIST trinomials (233,74) and (233,159). Furthermore, in order to check the property of the proposed multiplier for performing parallel multiplications for different fields $GF(2^m)$ generated by different irreducible trinomials simultaneously, experimental results are given in Table 7 (labeled as *Parallel*) for parallel multiplications using both NIST $GF(2^{233})$ trinomials. For the method given in [14], two multipliers for these trinomials were implemented and for the proposed multiplier, the architecture given in Figure 5 was used. In both cases, multiplexers were included in order to select the product of one of the trinomials depending on a control signal. From experimental results, it can be observed that, for single multiplications over (233,74) and (233,159), the multiplier here proposed can use 6.6% and 13.5% less slices and needs 3.8% and 1.6% less time to perform a multiplication than the multiplier given in [14], respectively. For the area×time metrics, the multiplier here proposed obtains a reduction of 10.2% for (233,74) and 14.9% for (233,159) with respect to [14]. Experimental results for parallel multiplications show that the proposed multiplier can use 17.7% less slices, needs 2.9% more time to perform a multiplication and presents a reduction of 15.4% in area×time metrics with respect to [14].

TABLE 6
Complexities and Area-Time estimates for NIST $GF(2^{233})$ defined by trinomial $f(y) = y^{233} + y^{74} + 1$.

	Complexities							*Estimates*			
	#AND	#XOR	#OR	#FF	#Mux	#Clk	Delay	Transistors	Delay	Time	Area×Time
LSB [14]	233	234	0	466	233	233	$T_{\rm A} + T_{\rm X}$	7928	18	4194	33.25
MSB [14]	233	234	0	466	233	233	$T_{\rm A} + T_{\rm X}$	7928	18	4194	33.25
[15]	699	466	0	54289	233	699	$T_{\rm X} + 8(T_{\rm A} + T_{\rm N} + T_{\rm O})$	442700	188	131412	58176.09
[16]	699	466	0	934	466	154	$2T_{\rm X} + T_{\rm M}$	17258	35	5390	93.02
[17]	465	538	0	772	0	233	$T_{\rm A} + 10T_{\rm X}$	12194	126	29358	357.99
[18]	234	233	0	468	235	466	$T_{\rm A} + 233T_{\rm X}$	7956	2802	1305732	10388.40
[18]	465	464	0	468	235	465	$T_{\rm A} + 9T_{\rm X}$	10728	114	53010	568.69
[19]	234	233	0	468	2	465	$T_{\rm A} + 233T_{\rm X}$	6558	2802	1302930	8544.61
[20]	54289	54288	0	108112	0	233	$T_{\rm NAND} + T_{\rm X}$	1407780	18	4194	5904.23
[21]	27261	27261	0	1164	932	149	$T_{\rm A} + 8T_{\rm X} + 2T_{\rm M}$	342036	124	18476	6319.46
[22]	465	467	0	697	0	233	$T_{\rm A} + 2T_{\rm X}$	11168	30	6990	78.06
Prop.	233	465	0	465	0	233	$T_{\rm A} + T_{\rm X}$	7908	18	4194	33.17

TABLE 7
Comparison of FPGA implementations.

	Slices	*Per.*(ns)	*Time*(ns)	$A \times T$	NIST
[14]	120	1.84	428.72	51446.40	(233,74)
Prop.	112	1.77	412.41	46189.92	
[14]	118	1.87	435.71	51413.78	(233,159)
Prop.	102	1.84	428.72	43729.44	
[14]	338	1.75	407.75	137819.50	*Parallel*
Prop.	278	1.80	419.40	116593.20	

6 Conclusion

In this paper, a new bit-serial $GF(2^m)$ polynomial basis multiplier using irreducible trinomials is presented. The architecture here proposed is based on *Linear-Feedback Shift Register* and can perform a multiplication in m clock cycles with a constant propagation delay of $T_A + T_X$. These values match the best time results found in the literature for bit-serial PB multipliers with a slight reduction of the space complexity. Furthermore, an important characteristic of the proposed architecture is that it can perform the multiplication of two operands for t different finite fields $GF(2^m)$ generated by t irreducible trinomials simultaneously in m clock cycles with the inclusion of $t(m-1)$ flipflops and tm XOR gates. New general expressions for multiplication over irreducible trinomials $f(y) = y^m + y^n + 1$, with $1 \leq n \leq m-1$, have also been proposed in this work.

Acknowledgments

This work has been supported by the Spanish MINECO and CM under grants S2018/TCS-4423, TIN 2015-65277-R and RTI2018-093684-B-I00.

References

[1] A. Reyhani-Masoleh and M.A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over $GF(2^m)$", *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 945-959, Aug. 2004.

[2] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer, New York, 2004.

[3] H. El-Razouk and A. Reyhani-Masoleh, "New Bit-Level Serial $GF(2^m)$ Multiplication Using Polynomial Basis", *22nd IEEE Symposium on Computer Arithmetic, ARITH 22*, pp. 129-136, Lyon, France, June 22-24, 2015.

[4] L. Song and K.K. Parhi, "Low-Energy Digit-Serial/Parallel Finite Field Multipliers", *J. VLSI Signal Process.*, vol. 19, pp. 149-166, 1998.

[5] F. Rodríguez-Henríquez, N. Saquib, A. Díaz-Pérez and Ç.K. Koç, 'Cryptographic Algorithms on Reconfigurable Hardware', Springer, New York, 2006.

[6] T. Zhang and K.K. Parhi, "Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials", *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 734-749, July 2001.

[7] J.L. Imaña, "Bit-Parallel Finite Field Multipliers for Irreducible Trinomials", *IEEE Trans. Comput.*, vol. 55, no. 5, pp. 520-533, May 2006.

[8] H. Fan and Y. Dai, "Fast Bit Parallel $GF(2^m)$ Multiplier for All Trinomials", *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485-490, Apr. 2005.

[9] F. Rodríguez-Henríquez and Ç.K. Koç, "Parallel Multipliers Based on Special Irreducible Pentanomials", *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1535-1542, Dec. 2003.

[10] J.L. Imaña, "Fast Bit-Parallel Binary Multipliers Based on Type-I Pentanomials", *IEEE Trans. Comput.*, vol. 67, no. 6, pp. 898-904, June 2018.

[11] B. Liu and H. Wu, "Efficient Architecture and Implementation for NTRUEncrypt System", *Midwest Symposium on Circuits and Systems, MWSCAS*, pp. 1-4, Aug. 2-5, 2015.

[12] S. Gashkov and A. Frolov, "Comparative Analysis of Calculations in Cryptographic Protocols Using a Combination of Different Bases of Finite Fields", *12th Intl. Conf. Dependability and Complex Systems*, pp. 166-177, July 2-6, 2017.

[13] A. Seghier, J. Li and D.Z. Sun, "Advanced encryption standard based on key dependent S-Box cube", *IET Information Security*, vol. 13, no. 6, pp. 552-558, Nov. 2019.

[14] T. Beth and D. Gollman, "Algorithm Engineering for public Key Algorithms", *IEEE J. Sel. Areas Commun.*, vol. 7, no. 4, pp. 458-466, 1989.

[15] M.A. Hasan and M. Ebtedaei, "Efficient architectures for computations over variable dimensional Galois field", *IEEE Trans. Circuits Syst. I. Fundam. Theory Appl.*, vol. 45, no. 11, pp. 1205-1211, Nov. 1998.

[16] G.N. Selimis, A.P. Fournaris, H. Michail and O. Koufopavlou, "Improved throughput bit-serial multiplier for $GF(2^m)$ fields", *Integration*, vol. 42, pp. 217-226, 2009.

[17] A. Reyhani-Masoleh, "A New Bit-Serial Architecture for field Multiplication Using Polynomial Bases", *Cryptographic Hardware and Embedded Systems, CHES 2008*, LNCS 5154, pp. 300-314, 2008.

[18] S.T.J. Fenn, M.G. Parker, M. Benaissa and D.Taylor, "Bit-serial multiplication in $GF(2^m)$ using irreducible all-one polynomials", *IEE Proc. Comput. Digit. Tech*, vol. 144, no. 6, pp. 391-393, 1997.

[19] H.-S. Kim and K.-Y. Yoo, "AOP arithmetic architectures over $GF(2^m)$", *Appl. Mathem. and Computation*, vol. 158, pp. 7-18, 2004.

[20] P.K. Meher, "Systolic and Super-Systolic Multipliers for Finite Field $GF(2^m)$ Based on Irreducible Trinomials", *IEEE Trans. Circuits Syst. I Reg. Papers*, vol. 55, no. 4, pp. 1031-1040, May 2008.

[21] J.L. Imaña, "Low Latency $GF(2^m)$ Polynomial Basis Multiplier", *IEEE Trans. Circuits Syst. I Reg. Papers*, vol. 58, no. 5, pp. 935-946, May 2011.

[22] H. El-Razouk and A. Reyhani-Masoleh, "New Bit-Level Serial $GF(2^m)$ Multiplication Using Polynomial Basis", *IEEE 22nd Symposium on Computer Arithmetic*, pp. 129-136, 2015.