Subquadratic space complexity multiplier for a class of binary fields using Toeplitz matrix approach

M. A. Hasan[1] and C. Negre[2]
[1]ECE Department and CACR, University of Waterloo, Ontario, Canada.
[2]Team DALI/ELIAUS, Université de Perpignan, France.

Abstract

In the recent past, subquadratic space complexity multipliers have been proposed for binary fields defined by irreducible trinomials and some specific pentanomials. For such multipliers, alternative irreducible polynomials can also be used, in particular, nearly all one polynomials (NAOPs) seem to be better than pentanomials (see [7]). For improved efficiency, multiplication modulo an NAOP is performed via modulo a quadrinomial whose degree is one more than that of the original NAOP. In this paper, we present a Toeplitz matrix-vector product based approach for multiplication modulo a quadrinomial. We obtain a fully parallel (non-sequential) multiplier with a subquadratic space complexity, which has the same order of space complexity as that of Fan and Hasan [4].

The Toeplitz matrix-vector product based approach is also interesting in the design of sequential multipliers. In this paper, we present two such multipliers: one with bit serial output and the other bit parallel output.

Index Terms

Subquadratic complexity, binary field, multiplication, double basis.

I. Introduction

For hardware implementation of certain cryptosystems [2], [9], [8], a finite field multiplier can be one of the most circuit or space demanding blocks. In order to make such a multiplier circuit-efficient, low weight irreducible polynomials are used for defining the representation of the field elements. For an irreducible polynomial, with coefficients being 0 and 1 only, the least weight is three. Such irreducible binary trinomials however do not exist for all degrees. The second least weight for an irreducible binary polynomial is five and such pentanomials appear to exist for all practical purpose.

A slightly different approach is to use a low weight composite, instead of irreducible, polynomial to define the representation of the field elements [7]. This leads to redundancy in the representation, but can reduce the circuit requirement of the multiplier. To this end, in the past composite binomials of the form $X^n + 1$ have been suggested. For reduced redundancy, such a binomial is chosen to be the product of $X + 1$ and an irreducible all-one polynomial (AOP) [6]. The latter is however not so abundant and the use of nearly AOP (NAOP) has been suggested. Irreducible NAOPs appear to be abundant. The multiplication of $X + 1$ and an NAOP results in a polynomial of weight four. Such a composite quadrinomial may be a better choice than an irreducible pentanomial when an irreducible trinomial is not available.

In this paper, we use quadrinomials. In addition, for multiplication we represent the two inputs with respect to two different bases. The main motivation of using two bases is to be able to formulate the field multiplication as a Toeplitz matrix-vector product (TMVP). It is well known that such a product can be obtained in a bit parallel fashion with subquadratic circuit/space complexity [4]. In the recent past there has been a considerable amount of interest to design sub-quadratic space complexity field multipliers, especially for cryptographic applications where the dimension of the field is of intermediate sizes, i.e., in the range of hundreds.

The main contributions of this work are as follows. For the modified polynomial basis $\mathcal{B}$ introduced in [10], we have formulated the problem of binary field multiplication as a TMVP. The TMVP approach has been known for other bases, but to the best of our knowledge this is the first time that a direct TMVP approach has been derived for the modified polynomial basis. The TMVP formulation for the modified polynomial basis has been achieved by expressing one of the inputs with respect to another basis $\mathcal{B}'$. We are not aware of the prior use of $\mathcal{B}'$ in finite field arithmetic and in our work, we have given explicit formulas for basis conversions– from $\mathcal{B}$ to $\mathcal{B}'$ and vice versa. We have also proposed bit serial multiplier structures involving bases $\mathcal{B}$ and $\mathcal{B}'$. Although bit serial multipliers are available for various bases, to the best of our knowledge the proposed structures are the first for these $\mathcal{B}$ and $\mathcal{B}'$ bases.

II. Binary field multiplication

A binary field $\mathbb{F}_{2^{n-1}}$ is the set of binary polynomials modulo an irreducible polynomial P of degree $n-1$

$$\mathbb{F}_{2^{n-1}} = \mathbb{F}_2[X]/(P).$$

Each element can be seen as a polynomial of degree at most $n-2$ and they are often expressed in the polynomial basis $\{1, X, X^2 \ldots, X^{n-2}\}$. In $\mathbb{F}_{2^{n-1}}$, for such representation, field operations like multiplication and inversion are done modulo P. Multiplication is widely used in practice, in this paper we focus on this operation. Let $A = \sum_{i=0}^{n-2} a_i X^i$ and $B = \sum_{i=0}^{n-2} b_i X^i$ be two elements expressed in $\mathcal{B}$. We can compute $C = A \times B \mod P$ as

$$\begin{array}{rcl} C &=& A \times B = \sum_{i=0}^{n-2} (AX^i) b_i \mod P \\ &=& \sum_{i=0}^{n-2} A^{(i)} b_i, \end{array}$$

after expanding the expression of B in $\mathcal{B}$ and noting that $A^{(i)} = AX^i \mod P$. This can be written through a matrix vector product

$$C = \left[\begin{array}{cccc} A^{(0)} & A^{(1)} & \cdots & A^{(n-2)} \end{array}\right] \cdot B.$$

The two most used strategies to design an efficient hardware multiplier via the above matrix vector product are the following:

1) The choice of the polynomial P must provide an efficient computation of the column $A^{(i)}$. Until now the all one polynomials (AOP) and trinomials seems to be the best possible choice. However, neither of them exists for all degree. Consequently other type of irreducible polynomials have been considered. A class of pentanomials of the form

$$P = X^{n-1} + X^{k+2} + X^{k+1} + X^k + 1$$

seems to be very interesting for the computation of the columns $A^{(i)}$ [4], [11] . Almost irreducible trinomials have also been proposed [7], [1], [3]; these trinomials $X^{n-1+\delta} + X^k + 1$ have an irreducible factor of degree $n-1$. For a large set of field δ has really small value.

2) The second strategy consists of expressing the matrix

$$\left[\begin{array}{cccc} A^{(0)} & A^{(1)} & \cdots & A^{(n-2)} \end{array}\right]$$

to a Toeplitz form. We will see in Subsection II-B that a Toeplitz matrix vector product can be done efficiently through a subquadratic complexity circuit. Generally we can obtain the Toeplitz form of the matrix by performing some row operations or column operations, or in other words, by using different bases of representation.

A. Nearly all one polynomials

Here we will design a multiplier modulo the irreducible polynomial introduced by Katti and Brennan in [7]. We call such polynomials nearly all one polynomials (NAOPs). They have the following form $P = \sum_{i=0}^{k_2-1} X^i + \sum_{i=k_1}^{n-1} X^i$ with $k_2 < k_1$. In other words, for a NAOP all the coefficients are equal to 1 unless they are in an interval $[k_2, k_1 - 1]$. If P is irreducible, we can define $\mathbb{F}_{2^{n-1}}$ as $\mathbb{F}_2[X]/(P)$.

As noticed in [7], multiplication in these fields can be done efficiently modulo $Q = (X+1) \times P$ since Q is a quadrinomial

$$Q = 1 + X^{k_2} + X^{k_1} + X^n.$$

In Table I we give several irreducible NAOPs with degrees suitable for cryptographic applications. Irreducible NAOPs are abundant, so we do not list all of them for each degree. However it appears to be an open question that there exists a NAOP for each degree. As we will see later, irreducible NAOPs which satisfy $k_1 = k_2 + 1$ give a more efficient multiplier. As much as possible we give such NAOPs for each degree $(n-1)$ in Table I (they are marked by †). When there are no irreducible NAOPs with $k_1 = k_2 + 1$ for a certain degree, Table I lists the NAOP with minimum k_1 for the smallest possible k_2.

TABLE I
IRREDUCIBLE NAOP OF DEGREE $n-1$

$n-1$	k_1, k_2	$n-1$	k_1, k_2	$n-1$	k_1, k_2
$161^\dagger$	$66, 65$	$247^\dagger$	$11, 10$	$503^\dagger$	$11, 10$
163	$33, 22$	248	$26, 6$	504	$26, 14$
164	$7, 5$	249	$12, 9$	$505^\dagger$	$17, 16$
165	$15, 8$	250	$16, 14$	506	$13, 3$
166	$22, 12$	251	$33, 2$	507	$25, 2$
$167^\dagger$	$7, 6$	252	$25, 17$	508	$17, 1$
$\vdots$	$\vdots$	$253^\dagger$	$43, 42$	$509^\dagger$	$75, 76$
$189^\dagger$	$35, 34$	254	$18, 14$	510	$34, 8$
190	$10, 2$	$255^\dagger$	$53, 52$	$511^\dagger$	$11, 10$
$191^\dagger$	$24, 23$	256	$34, 28$	512	$24, 22$
192	$20, 2$	$257^\dagger$	$69, 68$	513	$38, 29$
$193^\dagger$	$22, 21$	258	$11, 1$	514	$18, 6$
194	$17, 13$	259	$55, 22$	515	$315, 38$
$195^\dagger$	$3, 2$	$260^\dagger$	$9, 1$	516	$22, 2$
$\vdots$	$\vdots$	261	$35, 34$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

B. Asymptotic Complexities of Toeplitz Matrix Vector Product

As previously mentioned, multiplication in the binary field is often expressed as a Toeplitz matrix vector product. In this section we recall the method used to build a subquadratic circuit which computes a Toeplitz-matrix-vector multiplication. Recall that a Toeplitz matrix is defined as

Definition 1: An $n \times n$ Toeplitz matrix is a matrix $[t_{i,j}]_{0 \leq i,j \leq n-1}$ such that $t_{i,j} = t_{i-1,j-1}$ for $1 \leq i, j$.

If $2|n$ we can use a *two way approach* presented in Table II to compute a matrix vector product $T \cdot V$, where T is an $n \times n$ Toeplitz matrix. If $3|n$ we can use the *three way* approach which is also presented in Table II.

If n is a power of 2 or a power of 3 the formulas of Table II can be used recursively to perform $T \cdot V$. Using these recursive processes through parallel computation, the resulting multipliers [4] have the complexity given in Table III. In this table D_A represents the delay of an AND gate and D_X the delay of an XOR gate. It is also possible to design subquadratic TMVP multipliers for size $n = 3^i 2^j$ by combining two-way and three-way splits in the recursive computations.

III. Double basis representation

Let $Q = 1 + X^{k_2} + X^{k_1} + X^n$ be a quadrinomial with $k_2 < k_1 < n$ in $\mathbb{F}_2[X]$. Let $l_1 = n - k_1$ and $l_2 = n - k_2$. We present now our contribution on multiplication modulo Q. Our main goal is to get a subquadratic complexity multiplier modulo such a quadrinomial. To reach this goal, we attempt to express the product of two elements modulo Q as a Toeplitz matrix vector product.

First, we represent the elements of $\mathbb{F}_2[X]$ using the basis $\mathcal{B} = \{e_0, e_1, \ldots, e_{n-1}\}$ given in equation (1). This basis was introduced in [10]. In this basis the matrix involved in the multiplication is easier to

TABLE II

SUBQUADRATIC COMPLEXITY TOEPLITZ MATRIX VECTOR PRODUCT

Matrix decomposition	
Two-way	Three-way
$T = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}$	$T = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix}$
Recursive formulas	
$T \cdot V = \begin{bmatrix} P_0 + P_2 \\ P_1 + P_2 \end{bmatrix}$ where $P_0 = (T_0 + T_1) \cdot V_1,$ $P_1 = (T_1 + T_2) \cdot V_0,$ $P_2 = T_1 \cdot (V_0 + V_1),$	$T \cdot V = \begin{bmatrix} P_0 + P_3 + P_4 \\ P_1 + P_3 + P_5 \\ P_2 + P_4 + P_5 \end{bmatrix}$ where $P_0 = (T_0 + T_1 + T_2) \cdot V_2,$ $P_1 = (T_0 + T_1 + T_3) \cdot V_1,$ $P_2 = (T_2 + T_3 + T_4) \cdot V_0,$ $P_3 = T_1 \cdot (V_1 + V_2),$ $P_4 = T_2 \cdot (V_0 + V_2),$ $P_5 = T_3 \cdot (V_0 + V_1),$

TABLE III

ASYMPTOTIC COMPLEXITY

	Two-way split method	Three-way split method
# AND	$n^{\log_2(3)}$	$n^{\log_3(6)}$
# XOR	$5.5n^{\log_2(3)} - 6n + 0.5$	$\frac{24}{5}n^{\log_3(6)} - 5n + \frac{1}{5}$
Delay	$D_A + 2\log_2(n)D_X$	$D_A + 3\log_3(n)D_X$

compute than in the polynomial basis. However the matrix of [10] does not have the Toeplitz form. Here we obtain the Toeplitz form by performing some column operations on this matrix. These column operations should also be performed on the entries of B which is thus expressed in different or second basis $\mathcal{B}'$ and is given below.

$$\begin{array}{rcl|rcl|l} & \mathcal{B} & & & \mathcal{B}' & & \text{index} \\ e_i &=& X^i & e'_i &=& X^i & \text{for } i \in [0, l_1[, \\ e_i &=& X^i + X^{i-l_1} & e'_i &=& X^i & \text{for } i \in [l_1, l_2[, \\ e_i &=& X^i + X^{i-l_1} + X^{i-l_2} & e'_i &=& e_i & \text{for } i \in [l_2, n[. \end{array} \quad (1)$$

Let us specify the general construction of such a double basis multiplier. Let $A = \sum_{i=0}^{n-1} a_i e_i$ be expressed relative to $\mathcal{B}$ and $B = \sum_{j=0}^{n-1} b'_j e'_j$ be expressed relative to $\mathcal{B}'$. The product $C = AB$ can be written as

$$C = A\left(\sum_{j=0}^{n-1} b'_j e'_j\right) = \sum_{j=0}^{n-1} b'_j \left(Ae'_j\right). \quad (2)$$

So if we denote $A^{(j)} = Ae'_j$ we obtain $C = \sum_{j=0}^{n-1} b'_j A^{(j)}$. Then if $A^{(j)}$ is expressed with respect to $\mathcal{B}$, using vector notations we obtain the $\mathcal{B}$ representation of C as

$$\begin{array}{rcl} C &=& \left[A^{(0)}, A^{(1)}, \cdots, A^{(n-1)}\right] \cdot B \\ &=& M_A \cdot B. \end{array} \quad (3)$$

We will show that a simple permutation of the columns of M_A results in a Toeplitz matrix, and thus the previous product can be performed using subquadratic approach given in Subsection II-B.

To have a complete multiplier, one would expect to use the same basis to represent the elements A, B, C. Here we use the basis $\mathcal{B}$. Thus, there is a preliminary step which performs a conversion of B from $\mathcal{B}$ to $\mathcal{B}'$. Below we present formulas to perform this conversion.

A. Conversion from $\mathcal{B}$ to $\mathcal{B}'$

We assume here that $l_1 > l_2 - l_1$. Let $B = \sum_{i=0}^{n-1} b_i e_i$ be expressed relative to $\mathcal{B}$. To convert the coefficient of B from $\mathcal{B}$ to $\mathcal{B}'$ we use the relation

$$\begin{array}{rcl} e_i &=& e'_i \text{ if } i \in [0, l_1[\cup[l_2, n[, \\ e_i &=& X^i + X^{i-l_1} = e'_i + e'_{i-l_1} \text{ if } i \in [l_1, l_2[. \end{array}$$

For B, we can write

$$\begin{aligned} B &= \textstyle\sum_{i=0}^{l_1-1} b_i e_i + \sum_{i=l_1}^{l_2-1} b_i e_i + \sum_{i=l_2}^{n-1} b_i e_i \\ &= \textstyle\sum_{i=0}^{l_1-1} b_i e'_i + \sum_{i=l_1}^{l_2-1} b_i (e'_i + e'_{i-l_1}) + \sum_{i=l_2}^{n-1} b_i e'_i \\ &= \textstyle\sum_{i=0}^{l_2-l_1-1} (b_i + b_{i+l_1}) e'_i + \sum_{i=l_2-l_1}^{n-1} b_i e'_i \end{aligned}$$

In other words $b'_i = b_i + b_{i+l_1}$ if $i < l_2 - l_1$ and $b'_i = b_i$ if $i \geq l_2 - l_1$.

B. Conversion from $\mathcal{B}'$ to $\mathcal{B}$

The reverse conversion is not required in our double basis multiplier, but for completeness we present it here. We need to express b_i in terms of b'_j. From the previous conversion from $\mathcal{B}$ to $\mathcal{B}'$ we know that

$$\begin{array}{rcl} b_i &=& b'_i \text{ if } i \geq l_2 - l_1, \\ b_i &=& b'_i + b_{i+l_1} \text{ if } i < l_2 - l_1. \end{array}$$

Thus we need to replace b_{i+l_1} by its own expression in terms of b'_i

$$\begin{array}{rcll} b_i &=& b'_i + b'_{i+l_1} & \text{if } i + l_1 \geq l_2 - l_1, \\ b_i &=& b'_i + b'_{i+l_1} + b_{i+2l_1} & \text{if } i + l_1 < l_2 - l_1. \end{array}$$

For the last case, we have to repeat the process with b_{i+2l_1}. If we expand the recursion we obtain the following formula

$$b_i = b'_i + b'_{i+l_1} + \ldots + b'_{i+\alpha l_1} \text{ with } \alpha = \left\lceil \frac{l_2 - l_1 - i}{l_1} \right\rceil.$$

Consequently if such conversion is needed, it is better to take quadrinomial such that $l_1 > l_2 - l_1$ since in this case α is at most equal to 1 and conversion requires at most $l_1 - l_2$ XOR operations. Most of the NAOP given in Table I satisfy $l_1 > l_2 - l_1$ and thus conversions between $\mathcal{B}$ and $\mathcal{B}'$ can be done using the formulas of the current section.

IV. CONSTRUCTION OF THE MATRIX M_A

Let $M_A = \left[A^{(0)}, A^{(1)}, \cdots, A^{(n-1)}\right]$ be the resulting matrix of the double basis multiplier associated to bases $\mathcal{B}$ and $\mathcal{B}'$ defined in (1). The $n \times n$ matrix M_A can be generated column by column starting from the left end as follows.

A. Generating the first l_2 columns of M_A.

First, note that $A^{(0)} = A$ and for $1 \leq i \leq l_2 - 1$ one can write

$$A^{(i)} = Ae'_i = AX^i = AX^{i-1}X = A^{(i-1)}X \bmod Q.$$

If we call $a_j^{(i)}$ the j-th coefficient in $\mathcal{B}$ of $A^{(i)}$, the previous equation becomes

$$A^{(i)} \;=\; \textstyle\sum_{j=0}^{n-1} a_j^{(i-1)} e_j X.$$

We need to express $e_j X$ in $\mathcal{B}$ and this is the goal of the following lemma.

Lemma 1: Let $\mathcal{B} = \{e_0, \ldots, e_{n-1}\}$ as defined in (1). *Then the following equation holds*

$$e_i X = \begin{cases} e_{i+1} & \text{if } i \neq l_1 - 1, l_2 - 1, n-1, \\ e_{i+1} + e_0 & \text{if } i = l_1 - 1, l_2 - 1, \\ e_0 & \text{if } i = n-1. \end{cases}$$

Proof: Suppose that $i \neq n-1, l_1 - 1, l_2 - 1$, thus $i + 1 \neq n, l_1, l_2$. We recall that

$$e_i = \begin{cases} X^i & \text{if } 0 \le i < l_1 - 1, \\ X^i + X^{i-l_1} & \text{if } l_1 \le i < l_2 - 2, \\ X^i + X^{i-l_1} + X^{i-l_2} & \text{if } l_2 < i < n-1. \end{cases}$$

Thus if we multiply these expressions with X, we get

$$e_i X = \begin{cases} X^{i+1} = e_{i+1} & \text{if } 0 \le i < l_1 - 1, \\ X^{i+1} + X^{i+1-l_1} = e_{i+1} & \text{if } l_1 \le i < l_2 - 1, \\ X^{i+1} + X^{i+1-l_1} + X^{i+1-l_2} & \\ \qquad = e_{i+1} & \text{if } l_2 \le i < n-1. \end{cases}$$

For the special cases $i \in \{l_1 - 1, l_2 - 1, n-1\}$, we have

$$\begin{array}{rcl} e_{l_1-1}X & = & X^{l_1-1}X = (X^{l_1} + 1) + 1 = e_{l_1} + e_0, \\ e_{l_2-1}X & = & (X^{l_2-1} + X^{l_2-1-l_1})X \\ & = & (X^{l_2} + X^{l_2-l_1} + 1) + 1 = e_{l_2} + e_0, \\ e_{n-1}X & = & X^n + X^{k_2} + X^{k_1} = 1 \mod Q = e_0. \end{array}$$

This completes the proof. ■

Now we can write

$$\begin{array}{rcl} A^{(i)} & = & \sum_{j=0}^{n-1} a_j^{(i-1)} e_j X \\ & = & \left(\sum_{j=0}^{n-2} a_j^{(i-1)} e_{j+1}\right) + a_{n-1}^{(i-1)} e_0 + a_{l_1-1}^{(i-1)} e_0 \\ & & +a_{l_2-1}^{(i-1)} e_0, \end{array}$$

This previous expression enables us to compute $A^{(i)}$ as illustrated below:

$$\begin{array}{cccc} A^{(0)} & A^{(1)} & A^{(2)} & \cdots \\ \downarrow & \downarrow & \downarrow & \\ a_0 & a_{n-1} + a_{l_1-1} + a_{l_2-1} & a_{n-2} + a_{l_1-2} + a_{l_2-2} & \cdots \\ a_1 & a_0 & a_{n-1} + a_{l_1-1} + a_{l_2-1} & \cdots \\ a_2 & a_1 & a_0 & \cdots \\ \vdots & \vdots & \vdots & \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots \end{array}$$

This gives the column $A^{(i)}$ of M_A for $i < l_2$. Now consider how to express $A^{(i)}$ for $l_2 \le i \le n-1$.

B. Generating the last k_2 columns of M_A

We remark that $A^{(i)} = A(X^i + X^{i-l_1} + X^{i-l_2})$ for $i = l_2, l_2 + 1, \ldots, n-1$. Thus, if we factorize X out from the right side we obtain for $i = l_2, l_2 + 1, \ldots, n-1$

$$A^{(i)} = A(X^{i-1} + X^{i-1-l_1} + X^{i-1-l_2})X = A^{(i-1)}X,$$

which can be rewritten as

$$A^{(i-1)} = A^{(i)}X^{-1}.$$

Again if we replace $A^{(i)}$ by its expression in $\mathcal{B}$ we get

$$A^{(i-1)} = \sum_{j=0}^{n-1} a_j^{(i)} e_j X^{-1}.$$

To proceed, we need to compute $e_j X^{-1}$ and this is done in the following lemma.

Lemma 2: *Let $\mathcal{B} = \{e_0, \dots, e_{n-1}\}$ be the basis given in of $\mathbb{F}_2[X]/(Q(X))$. Then we have*

$$e_i X^{-1} = \begin{cases} e_{i-1} & \text{if } i \neq 0, l_1, l_2, \\ e_{i-1} + e_{n-1} & \text{if } i = l_1, l_2, \\ e_{n-1} & \text{if } i = 0. \end{cases} \tag{4}$$

Proof: We first deal with the cases $i \neq 0, l_1, l_2$. We have

$$e_i = \begin{cases} X^i & \text{if } 0 < i < l_1, \\ X^i + X^{i-l_1} & \text{if } l_1 < i < l_2, \\ X^i + X^{i-l_1} + X^{i-l_2} & \text{if } l_2 < i < n. \end{cases}$$

If we multiply e_i with X^{-1} we get

$$e_i X^{-1} = \begin{cases} X^{i-1} = e_{i-1} & \text{if } 0 < i < l_1, \\ X^{i-1} + X^{i-l_1-1} = e_{i-1} & \text{if } l_1 < i < l_2, \\ X^{i-1} + X^{i-l_1-1} + X^{i-l_2-1} & \\ \qquad = e_{i-1} & \text{if } l_2 < i < n, \end{cases}$$

as required.

Now we assume that $i = 0, l_1$ or l_2. To prove these three cases we use $e_{n-1} = X^{-1} \mod Q$, indeed

$$\begin{aligned} e_{n-1}X &= (X^{n-1} + X^{n-l_1} + X^{n-l_2})X \\ &= X^n + X^{k_1} + X^{k_2} \\ &= 1 \mod Q \end{aligned}$$

Thus $e_0 X^{-1} = X^{-1} = e_{n-1} \mod Q$. Similarly, we have $e_{l_1}X^{-1} = X^{l_1-1} + X^{-1} = e_{l_1-1} + e_{n-1}$, and $e_{l_2}X^{-1} = e_{l_2-1} + e_{n-1}$. ■

For $A^{(i-1)}$,

$$\begin{aligned} A^{(i-1)} &= \textstyle\sum_{j=0}^{n-1} a_j^{(i)} e_j X^{-1} \\ &= \textstyle\sum_{j=1}^{n-2} a_j^{(i)} e_{j-1} + a_0^{(i)} e_{n-1} + a_{l_2}^{(i)} e_{n-1} \\ &\qquad + a_{l_1}^{(i)} e_{n-1}. \end{aligned}$$

We can thus compute $A^{(i)}$ for $i = n-1, \dots, l_2$ recursively, beginning with $A^{(n)}$ and multiply it by X^{-1}. For $A^{(n)}$ we have

$$\begin{aligned} A^{(n)} &= A \times (X^n + X^{n-l_1} + X^{n-l_2}) \\ &= A \times (X^n + X^{k_1} + X^{k_2}) \\ &= A \end{aligned}$$

Since $X^n + X^{k_1} + X^{k_2} = 1 \mod (X^n + X^{k_1} + X^{k_2})$. We can then compute $A^{(n-1)} = AX^{-1}, A^{(n-2)} = A^{(n-1)}X^{-1}, \dots$ as shown below:

$$\begin{array}{cccc} \cdots & A^{(n-2)} & A^{(n-1)} & A \\ & \downarrow & \downarrow & \downarrow \\ \cdots & a_2 & a_1 & a_0 \\ \cdots & a_3 & a_2 & a_1 \\ \cdots & a_4 & a_3 & a_2 \\ & \vdots & \vdots & \vdots \\ \cdots & a_0 + a_{l_1} + a_{l_2} & a_{n-1} & a_{n-2} \\ \cdots & a_1 + a_{l_1+1} + a_{l_2+1} & a_0 + a_{l_1} + a_{l_2} & a_{n-1} \end{array}$$

We easily remark that the matrix

$$T_A = [A^{(l_2)}, A^{(l_2+1)}, \dots, A^{(n-1)}, A^{(0)}, A^{(1)}, \dots, A^{(l_2-1)}] \tag{5}$$

is a Toeplitz matrix. In addition, the first k_2 columns (resp. the last $l_2 = n - k_2$ columns) of T_A are the last k_2 columns (resp. the first $n - k_2$ columns) of $M_A = [A^{(0)}, A^{(1)}, \ldots, A^{(n-1)}]$ from (3). Similarly we denote

$$\widetilde{B} = \left[\begin{array}{cccccccc} b'_{l_2} & b'_{l_2+1} & \cdots & b'_{n-1} & b'_0 & b'_1 & \cdots & b'_{l_2-1} \end{array}\right]$$

such that the first k_2 (resp. the last $n - k_2$) coefficients are equal to the last k_2 (resp. the first $n - k_2$) coefficients of B. In this situation, the coordinates in $\mathcal{B}$ of $C = AB$ are given by

$$C = T_A \cdot \widetilde{B}. \tag{6}$$

V. Example

We use the irreducible NAOP $P = X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$ to define the field $\mathbb{F}_{2^8}$. We construct the matrix M_A of an element A modulo $Q = P \times (X + 1) = X^9 + X^6 + X^4 + 1$. The basis $\mathcal{B}$ of $\mathbb{F}_2[X]/(Q)$ is defined by the 9 elements

$$\begin{array}{l} e_0 = 1, e_1 = X, e_2 = X^2, \\ e_3 = X^3 + 1, e_4 = X^4 + X, \\ e_5 = X^5 + X^2 + 1, e_6 = X^6 + X^3 + X, \\ e_7 = X^7 + X^4 + X^2, e_8 = X^8 + X^5 + X^3. \end{array}$$

In this situation, we have

$$\begin{array}{ll} l_1 = 3, & l_2 = 5, \\ k_1 = 6, & k_2 = 4. \end{array}$$

Let $A = \sum_{i=0}^{8} a_i e_i$ be expressed in $\mathcal{B}$. We first construct column $A^{(i)}$ for $i = 0, \ldots, l_2 - 1$ as in subsection IV-A.

$$\begin{array}{ccccc}
A^{(0)} & A^{(1)} & A^{(2)} & A^{(3)} & A^{(4)} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
a_0 & a_8 + a_4 + a_2 & a_7 + a_3 + a_1 & a_6 + a_2 + a_0 & a_5 + a_1 + a_8 + a_4 + a_2 \\
a_1 & a_0 & a_8 + a_4 + a_2 & a_7 + a_3 + a_1 & a_6 + a_2 + a_0 \\
a_2 & a_1 & a_0 & a_8 + a_4 + a_2 & a_7 + a_3 + a_1 \\
a_3 & a_2 & a_1 & a_0 & a_8 + a_4 + a_2 \\
a_4 & a_3 & a_2 & a_1 & a_0 \\
a_5 & a_4 & a_3 & a_2 & a_1 \\
a_6 & a_5 & a_4 & a_3 & a_2 \\
a_7 & a_6 & a_5 & a_4 & a_3 \\
a_8 & a_7 & a_6 & a_5 & a_4
\end{array}$$

Similarly we compute the other part as explained in subsection IV-B.

$$\begin{array}{ccccc}
A^{(5)} & A^{(6)} & A^{(7)} & A^{(8)} & A^{(0)} \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
a_4 & a_3 & a_2 & a_1 & a_0 \\
a_5 & a_4 & a_3 & a_2 & a_1 \\
a_6 & a_5 & a_4 & a_3 & a_2 \\
a_7 & a_6 & a_5 & a_4 & a_3 \\
a_8 & a_7 & a_6 & a_5 & a_4 \\
a_0 + a_3 + a_5 & a_8 & a_7 & a_6 & a_5 \\
a_1 + a_4 + a_6 & a_0 + a_3 + a_5 & a_8 & a_7 & a_6 \\
a_2 + a_5 + a_7 & a_1 + a_4 + a_6 & a_0 + a_3 + a_5 & a_8 & a_7 \\
a_3 + a_6 + a_8 & a_2 + a_5 + a_7 & a_1 + a_4 + a_6 & a_0 + a_3 + a_5 & a_8
\end{array}$$

VI. Multiplier architecture

In this section, we present several multiplier architectures associated to the Toeplitz matrix vector expression in (6). Multiplier architectures can be classified into two types:

- *Parallel architecture.* Computations are done with no reuse of circuits and all the bits of the product $C = AB$ are output at each clock cycle.
- *Sequential architecture.* Computations are done with reuse of circuits and the result is obtained after n clock cycles. There are mainly two types of sequential multipliers, one which output one bit per clock cycle, and the other which output all n bits (in parallel) only at the end of n clock cycles. In the paper the former will be referred to as sequential multiplier with bit serial output (SMSO) and the latter as sequential multiplier with bit parallel output (SMPO).

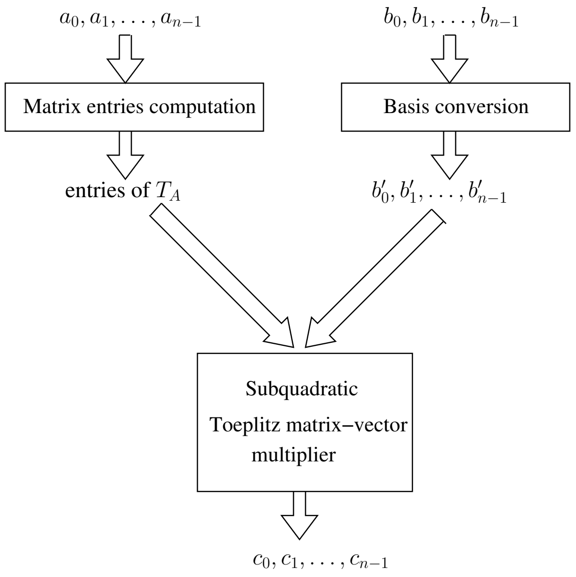
A. Parallel architecture

Here we present a parallel architecture associated to the double basis approach. This architecture is sketched in Figure 1. It consists of two preliminary parallel computations followed by a Toeplitz matrix vector product. The preliminary step computes the entries of the matrix T_A from the coordinates of A and in parallel performs the conversion of B from $\mathcal{B}$ to $\mathcal{B}'$. For the former, we use the expression of the columns $A^{(i)}$ given in Section IV. Let $a_j^{(i)}$ denote the j-th entry of $A^{(i)}$. Then we have

$$a_1^{(i)} = a_{n-i} + a_{l_2-i} + a_{l_1-i}, 1 \leq i \leq l_1, \quad (7)$$
$$a_1^{(i)} = a_{l_1-1}^{(i-1)} + a_{n-i} + a_{l_2-i}, l_1 < i < l_2, \quad (8)$$
$$a_{n-1}^{(n-i)} = a_{i-1} + a_{i-1+l_2} + a_{i-1+l_1}, 1 \leq i \leq k_2. \quad (9)$$

To compute the coordinates $b_i', i = 0, \ldots, n-1$ we only need to apply the formula given in subsection III-A. We evaluate the complexity of this architecture below:

Fig. 1. Parallel architecture for double basis multiplication modulo a quadrinomial


$a_0, a_1, \ldots, a_{n-1}$
$b_0, b_1, \ldots, b_{n-1}$
Matrix entries computation
Basis conversion
entries of T_A
$b_0', b_1', \ldots, b_{n-1}'$
Subquadratic Toeplitz matrix−vector multiplier
$c_0, c_1, \ldots, c_{n-1}$

- *Space complexity.* It includes the space (i.e., logic gates) needed for the precomputations as well as the matrix-vector product. Using (7), (8) and (9), we deduce that the precomputations require
$$\underbrace{(2(l_2-1)+2k_2)}_{\text{computation of } T_A} + \underbrace{(l_2-l_1)}_{\text{conversion of } B} \text{ XOR gates.}$$
We already know the space complexity of the matrix vector product which is given in Table III.
- *Time complexity.* The delay of the critical path is equal to the delay for the preliminary computations plus the delay of the Toeplitz matrix-vector product part. The critical path for the precomputations corresponds to the computation of Eq. (8) and the corresponding delay is equal to $3D_X$ to compute the coefficient of T_A.

Remark 1: The parallel architecture in Figure 1 has a small improvement in the complexity of the multiplier when $k_1 = k_2+1$. In this case, all the entries of T_A are computed with (7) and (9). Consequently the space complexity required for the computation of the entries of T_A is equal to $2(n-1)$ XOR gates and the time delay is $2D_X$.

For a fixed field $\mathbb{F}_{2^{n-1}}$ we give in Table IV the complexity results of the two-way and three-way approaches for the double basis multiplier. We also give the complexities of a two-way and three-way

TABLE IV

COMPLEXITY COMPARISON FOR FIELD $\mathbb{F}_{2^{n-1}}$

Method	Splitting	Space Complexity		Delay
		# XOR	# AND	
This paper	Two-way	$5.5n^{\log_2(3)} - 3n + 0.5$	$n^{\log_2(3)}$	$D_A + (2\log_2(n) + 3)D_X$
Pentanomial [4]	Two-way	$5.5(n-1)^{\log_2(3)} - 3n + 4$	$(n-1)^{\log_2(3)}$	$D_A + (2\log_2(n-1) + 4)D_X$
Redundant-trinomial [4], [1], [3]	Two-way	$5.5(n-1+\delta)^{\log_2(3)} - 6$ $+k+5$	$(n-1+\delta)^{\log_2(3)}$	$D_A + (2\log_2(n-1+\delta) + 1)D_X$
This paper	Three-way	$\frac{24}{5}n^{\log_3(6)} - 2n + \frac{1}{5}$	$n^{\log_3(6)}$	$D_A + (3\log_3(n) + 2)D_X$
Pentanomial [4]	Three-way	$\frac{24}{5}m^{\log_3(6)} - 2(n-1) + \frac{7}{10}$	$(n-1)^{\log_3(6)}$	$D_A + (3\log_3(n-1) + 3)D_X$
Redundant-trinomial [4], [1], [3]	Three-way	$\frac{72}{15}(n-1+\delta)^{\log_3(6)}$ $-4(n-1+\delta) - \frac{4}{5}$	$(n-1+\delta)^{\log_3(6)}$	$D_A + (3\log_3(m+\delta) + 1)D_X$
Sunar [12]	Winograd	$6(n-1)^{\log_2(3)} - 8n + 10$	$(n-1)^{\log_2(3)}$	$D_A + (2\log_2(n-1) + 1)D_X$
Sunar [12]	Winograd	$\frac{16}{3}(n-1)^{\log_3(6)} - \frac{22}{3}(n-1) + 2$	$(n-1)^{\log_3(6)}$	$D_A + (4\log_3(n-1) + 1)D_X$

splitting multipliers based on [4] for almost irreducible trinomial [1] $X^{n-1+\delta}+X^k+1$, also called redundant trinomials in [3]. These redundant trinomials admit an irreducible factor of degree $(n-1)$ which defines the field $\mathbb{F}_{2^{n-1}}$. The authors of [1] proposed severals algorithms to find almost irreducible trinomials with small δ.

In Table IV we also give the complexity of the multiplier of [4] for specific pentanomials and the complexity of the multiplier of Sunar [12]. The complexity results given in Table IV are valid if n (resp. $n-1$, $n-1+\delta$) is a power of 2 or 3. Some combination of two-way and three-way approaches could also be used, which extend the use of TMVP multiplication to size $n = 2^i3^j$.

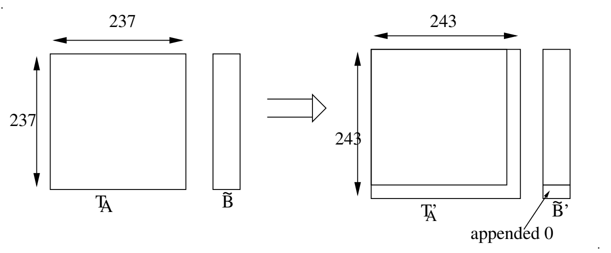
In this situation, we can see that quadrinomial with double basis approach gives a parallel multiplier with the same space complexity as the multiplier of Fan and Hasan for pentanomials, but with an improvement by D_X ($2D_X$ for special quadrinomials) in the delay of the multiplier. The redundant trinomial is better when δ is small, otherwise when δ is quite big the space complexity is higher.

For a general n, there are several strategies to obtain a subquadratic multiplier using the Toeplitz matrix. Using a small example, below we present two possible approaches : the first one enlarge the dimension of the Toeplitz matrix to the nearest 2^i3^j bigger than $(n-1)$, the second approach decomposes the matrix in a number Toeplitz blocks of size 2^i3^j.

Example 1: here we study here subquadratic space complexity multipliers for $n \neq 2^i3^j$. We do it for the field $\mathbb{F}_{2^{235}}$, first assuming double basis with quadrinomial and then redundant trinomials.

- *Quadrinomial approach.* We represent the field as a subring of $\mathbb{F}_2[X]/(X^{237} + X^2 + X + 1)$. In this situation, a multiplication of two elements is expressed as $T_A \cdot \widetilde{B}$ where T_A is a 237×237 matrix. Since 237 cannot be decomposed as power of 2 and 3, we expand the matrix T_A to a 243×243 Toeplitz matrix T'_A.

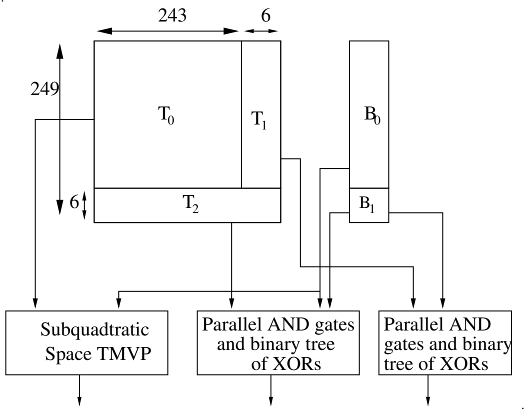
Fig. 2. Redundant quadrinomial architecture for $\mathbb{F}_{2^{235}}$



We also append 6 zeros to $\widetilde{B}$ to obtain $\widetilde{B}'$. The product of A and B is now given by the first 237 coefficients of $T'_A \cdot \widetilde{B}'$ and the three-way splitting approach can be applied since $243 = 3^5$. The resulting multiplier has a delay of $17D_X + D_A$ and a space complexity of 36586 XOR and 7776 AND gates.

- *Almost irreducible trinomial approach.* We represent the field $\mathbb{F}_{2^{235}}$ as a subring of $\mathbb{F}_2[X]/(X^{249} + X^7 + 1)$. The polynomial $X^{249} + X^7 + 1$ is the smallest degree trinomial which has a prime factor of a degree 235 polynomial. Following [4], the multiplication of two elements A and B can be expressed as a Toeplitz matrix vector product $T_A \cdot B$. In order to use the three-way splitting approach we propose to split the matrix T_A into three blocks (cf. Fig. 3).

Fig. 3. Redundant trinomial architecture for $\mathbb{F}_{2^{235}}$


243
6
249
6
T0
T1
T2
B0
B1
Subquadtratic Space TMVP
Parallel AND gates and binary tree of XORs
Parallel AND gates and binary tree of XORs

The computation $T_A \cdot B$ is decomposed in $T_0 \cdot B_0$ and $T_1 \cdot B_1$ and $T_2 \cdot \widetilde{B}$. We can perform these three TMVPs in parallel. This approach is depicted in Fig. 3. The computation of $T_0 \cdot B$ is done using three-way split subquadratic multiplication. For both $T_1 \cdot B_1$ and $T_2 \cdot B$ we use AND gates in parallel followed by XOR gates arranged in binary tree fashion. Other multiplication strategies could be applied for example for $T_2 \cdot B$, by splitting T_2 in several 6×6 Toeplitz matrices, computing each product in parallel and add the results with a binary tree of XORs. But this would not be advantageous considering both space and time complexities since the size of the Toeplitz matrices is really small. The resulting space complexity of this architecture is equal to 39061 XOR and 10728 AND gates and the delay is $17D_X + D_A$.

In conclusion of this example, we point out that the construction of the subquadratic multiplier for degree $n \neq 2^i 3^j$ involves some decomposition or expansion of the Toeplitz matrices. The consequence is that it requires some additional computation, and the best approach between quadrinomial or redundant trinomial depends deeply on the approach which gives the best decomposition or expansion.

B. Sequential multiplier with bit serial output (SMSO)

This multiplier is based on (6) where the i-th coordinate of C is obtained as the inner product of the i-th row of T_A and the vector $\widetilde{B}$. We remark that if we begin from the l_2-th row of T_A, we have

$$R_{l_2} = \left[\begin{array}{cccc} a_{n-1} & a_{n-2} & \cdots & a_0 \end{array} \right],$$

and R_{l_2+1} is generated from R_{l_2} by performing a right shift and placing $a_0 + a_{l_1} + a_{l_2}$ into the resulting empty position at the left most end. Then R_{l_2+2} is generated from R_{l_2+1} in the same way, and so on.

This process can be performed by applying right shifts to a bidirectional linear feedback shift register (bLFSR) as shown in the lower part of Figure 4. The bidirectional feature implies that the feedback shift register can be shifted to either left or right direction as desired.

Hardware structure for such an bLFSR has been presented in [5]. Specifically, the implementation of the modulo two addition of the linear relation for both left and right shifts requires an XOR gate and some additional hardware which manages the change of direction of the output (see Fig 5 where $L/\bar{L}$

Fig. 4. Sequential multiplier with bit serial output

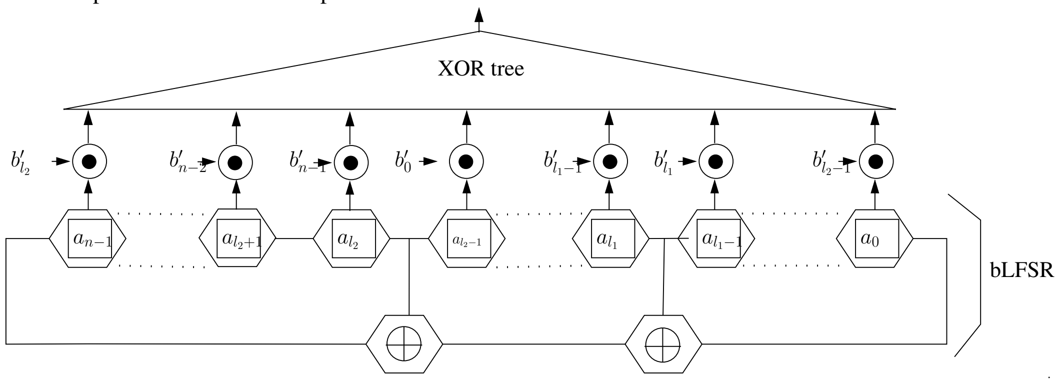
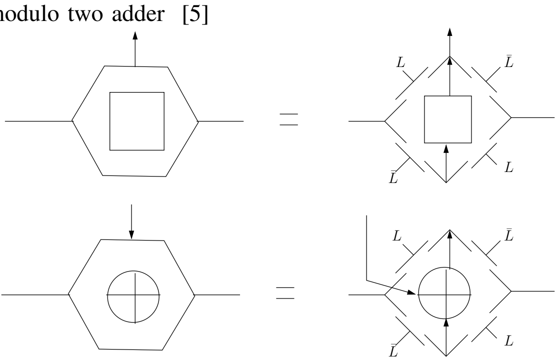


Fig. 5. Bi-directional storage cell and modulo two adder [5]



determines the direction of data flow). The same is true for the storage cells. More details about bLFSR can be found in [5].

We remark also that if we begin with R_{l_2}, we can shift the bLFSR contents to the left and place $a_{n-1} + a_{l_1-1} + a_{l_2-1}$ into the right most cell to have R_{l_2-1}. The row $R_{l_2-2}, \ldots, R_0$ can be computed in the same way. So using the left shift property of the bLFSR we can produce the first l_2 rows of T_A and thus compute the corresponding coefficients of C.

Consequently, we propose an original bit serial multiplier which uses these properties. This bit serial multiplier is depicted in Figure 4 and its operation has the following four steps.

1) Load the register with A.
2) Apply right shifts to bLFSR to output the bits $c_{l_2}, c_{l_2+1}, \ldots, c_{n-1}$.
3) Reload the register with A.
4) Apply $l_2 + 1$ left shifts to bLFSR to output the bits $c_{l_2}, c_{l_2-1}, \ldots, c_0$.

We remark that the bit c_{l_2} is output twice, since the rows in the upper and the lower parts of T_A are generated from the same row R_{l_2}. One of these bits should be removed.

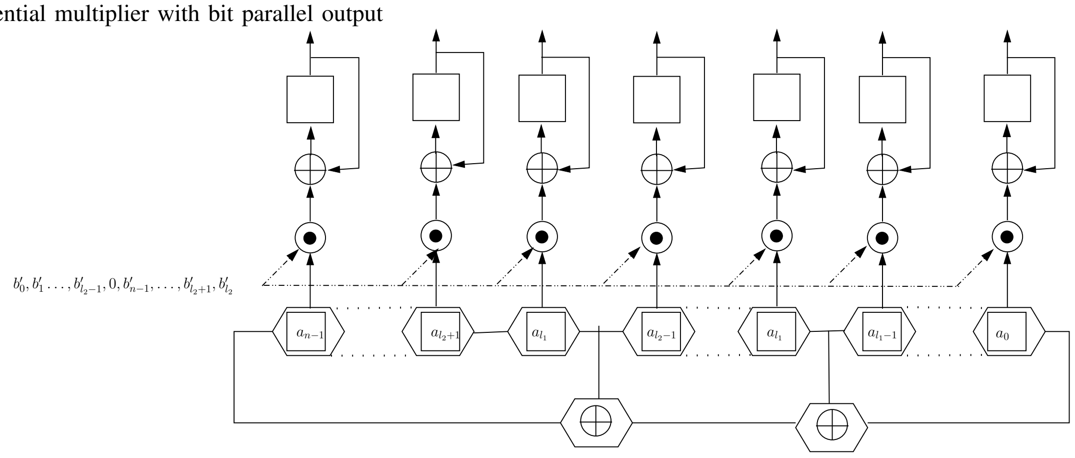
C. Sequential multiplier with bit parallel output (SMPO)

We have seen in Section IV that there are two recursive relations which allow us to generate the columns of T_A. By generating the columns $A^{(j)} = Ae'_j$ for $j = 0, 1, \ldots, n-1$ with a weight of b'_j we obtain C as in equation (6). Note that $A^{(j)}$'s are columns of T_A.

The expressions of $A^{(j)}$ in terms of $A^{(j-1)}$ or $A^{(j+1)}$ given in Section IV correspond to a left and right shifts of a bi-directional LFSR as described above to generate the rows of T_A. This leads to a multiplier with bit parallel output (Figure 6). The operation of the multiplier has the following steps

1) Load the register with A.
2) Apply k_2 right shifts to bLFSR.
3) Reload the register with A and apply a left shift with 0 as input in AND gates.
4) Apply l_2 left shifts to bLFSR to output the bits $c_{n-1}, c_{n-2}, \ldots, c_0$.

Fig. 6. Sequential multiplier with bit parallel output


$b'_0, b'_1 \ldots, b'_{l_2-1}, 0, b'_{n-1}, \ldots, b'_{l_2+1}, b'_{l_2}$
a_{n-1}
a_{l_2+1}
a_{l_1}
a_{l_2-1}
a_{l_1}
a_{l_1-1}
a_0

D. Comparison of SMSO and SMPO

Using Figures 4 and 6, we can easily evaluate the complexity of the two sequential multipliers. We can see that the two architectures have roughly the same number of XOR and AND gates. Compared to SMSO, SMPO requires twice more flip-flops but is faster by a factor of $\log_2(n+1)$.

TABLE V

COMPLEXITY OF SEQUENTIAL MULTIPLIERS

	SMSO	SMPO
# XOR	$n+1$	$n+2$
# AND	n	n
Flip-Flop	n cells	$2n$ cells
Delay	$(n+1)(D_A + \lceil \log_2(n+1) \rceil D_X)$	$(n+1)(D_A + D_X)$

VII. CONCLUSION

In this paper we have considered multipliers over binary fields defined by near all one polynomials. To this end, we have used multiplication modulo a quadrinomial. We have introduced a double basis approach which provides a Toeplitz matrix vector product expression for multiplication modulo the quadrinomial. This has resulted in a subquadratic space complexity parallel multiplier, which has lower delay than the Fan and Hasan multiplier modulo pentanomials [4]. We have also presented two sequential multipliers using a bidirectional LFSR.

REFERENCES

[1] Richard P. Brent and Paul Zimmermann. Algorithms for almost irreducible and almost primitive trinomials. In *in Primes and Misdemeanours: Lectures in Honour of the Sixtieth Birthday of Hugh Cowie Williams, Fields Institute*, 2004.

[2] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 24:644–654, 1976.

[3] Christophe Doche. Redundant trinomials for finite fields of characteristic 2. In *In ACISP 2005*, pages 122–133. SpringerVerlag, 2005.

[4] H. Fan and M. A. Hasan. A new approach to sub-quadratic space complexity parallel multipliers for extended binary fields. *IEEE Trans. Computers*, 56(2):224–233, September 2007.

[5] M. A. Hasan and V. K. Bhargava. Architecture for a Low Complexity Rate-Adaptive Reed-Solomon Encoder. *IEEE Trans. Computers*, pages 938–942, July 1995.

[6] T. Itoh and S. Tsujii. Structure of parallel multipliers for a class of fields GF(2^m). *Inform. Comp.*, 83:21–40, 1989.

[7] R. Katti and J. Brennan. Low complexity multiplication in a finite field using ring representation. *IEEE Trans. Comput.*, 52(4):418–427, 2003.

[8] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[9] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology, proceeding's of CRYPTO'85*, volume 218 of *LNCS*, pages 417–426. Springer-Verlag, 1986.

[10] C. Negre. Quadrinomial modular multiplication using modified polynomial basis. In *ITCC 2005, Las Vegas USA*, LNCS, april 2005.

[11] F. Rodriguez-Henriquez and C.K. Koç. Parallel multipliers based on special irreducible pentanomials. *IEEE Transaction on Computers*, 52(12):1535–1542, December 2003.

[12] B. Sunar. A generalized method for constructing subquadratic complexity GF(2^k) multipliers. *IEEE Transactions on Computers*, 53:1097–1105, 2004.