A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields

Haining Fan and M. Anwar Hasan *Senior Member, IEEE*

July 19, 2006

Abstract

Based on Toeplitz matrix-vector products and coordinate transformation techniques, we present a new scheme for subquadratic space complexity parallel multiplication in $GF(2^n)$ using the shifted polynomial basis. Both the space complexity and the asymptotic gate delay of the proposed multiplier are better than those of the best existing subquadratic space complexity parallel multipliers. For example, with n being a power of 2, the space complexity is about 8% better, while the asymptotic gate delay is about 33% better, respectively. Another advantage of the proposed matrix-vector product approach is that it can also be used to design subquadratic space complexity polynomial, dual, weakly dual and triangular basis parallel multipliers. To the best of our knowledge, this is the first time that subquadratic space complexity parallel multipliers are proposed for dual, weakly dual and triangular bases. A recursive design algorithm is also proposed for efficient construction of the proposed subquadratic space complexity multipliers. This design algorithm can be modified for the construction of most of the subquadratic space complexity multipliers previously reported in the literature.

Index Terms

Finite field, subquadratic space complexity multiplier, coordinate transformation, shifted polynomial basis, Toeplitz matrix.

Haining Fan and M. Anwar Hasan are with the Department of Electrical and Computer Engineering University of Waterloo, Waterloo, Canada. E-mails: hfan@vlsi.uwaterloo.ca and ahasan@ece.uwaterloo.ca

I. INTRODUCTION

When the extended binary field $GF(2^n)$ is viewed as a vector space over $GF(2)$, the elements of the field are often represented with respect to a basis. For the purpose of efficient field arithmetic, a number of bases have been proposed in the literature, for example, polynomial, normal, dual, weakly dual, triangular and shifted polynomial bases. Between the two basic field arithmetic operations, namely, addition and multiplication, it is easier to implement the former using these representations. Multiplication is inherently complex and the existing bit parallel multipliers may be classified into the following two categories on the basis of the space complexity, which is often measured in terms of the number of 2-input AND and XOR gates: quadratic and subquadratic space complexity multipliers. Multipliers of [5]-[20] belong to the former category and those of [21]-[32] the latter category. The subquadratic space complexity multiplier provides a practical solution for large values of n due to its low space complexity. To this end, the polynomial version of the integer Karatsuba multiplication algorithm [1] has been widely used, e.g., [21]-[30]. These multipliers first perform a multiplication of two binary polynomials, each of degree $n-1$ or less, and then a modulo reduction operation using the field generating irreducible polynomial. For practical applications, n may not be an even integer. In such cases, in order to apply the Karatsuba algorithm some slight modifications are often made to the inputs, e.g., padding zeroes. The asymptotic gate delay of the Karatsuba algorithm is $(3\log_2 n - 1)T_X + T_A$ for $n = 2^i$ $(i > 1)$, where T_A and T_X are the delay of one 2-input AND and XOR gates.

The Karatsuba algorithm based multiplier has a low asymptotic space complexity, but its gate delay is three times more than that of the existing fast parallel multipliers [31]. In order to reduce this delay, a hybrid structure has been developed in [25]. Although this method does not improve the asymptotic gate delay, it is effective for some finite fields, e.g., $GF(2^{233})$. The Winograd short convolution algorithm may be viewed as a generalization of the original Karatsuba-based algorithm [2], [31] and [33]. For $n = 3^i$ $(i > 0)$, the algorithm improves the asymptotic gate delay of the Karatsuba algorithm in [21], and results in an asymptotic gate delay of $(4\log_3 n - 1)T_X + T_A \approx (2.52\log_2 n - 1)T_X + T_A$. But the asymptotic space complexity of this method is slightly worse than that of the Karatsuba-based multiplier. The parallel multiplier presented in [32] is based on the Chinese Remainder Theorem (CRT) and Montgomery's algorithm. It allows

the use of the extension field for which an irreducible trinomial or special pentanomial does not exist.

In summary, all the above subquadratic space complexity $GF(2^n)$ parallel multipliers are based on the improved polynomial multiplication algorithms, and will be referred to as the $polynomial\text{-}based$ multipliers in the following.

In this article, we follow a different approach using Toeplitz matrix-vector products and propose a new scheme for the subquadratic space complexity $GF(2^n)$ parallel multiplier. Our scheme takes advantage of a shifted polynomial basis (SPB) and applies the coordinate transformation technique of [3] and [4]. The end result is that not only the space complexity of the new multiplier is lower than that of the existing $polynomial\text{-}based$ multipliers, e.g., [21] and [31], its asymptotic gate delay is also better, in fact considerably better, than that of these $polynomial\text{-}based$ designs. Another advantage of the proposed matrix-vector product approach is that it can also be used to design subquadratic space complexity parallel multipliers using polynomial, dual, weakly dual or triangular basis.

Because of lack of apparent regularity, hardware implementations of subquadratic space complexity parallel multipliers requires considerable efforts. For large n, efficient design of subquadratic space complexity multipliers is often based on recursive application of the divide-and-conquer technique, and is not straightforward. Therefore, a systematic design approach is desirable. In this article, a recursive design algorithm is also proposed for an efficient construction of the proposed subquadratic space complexity multipliers. We have implemented the algorithm using ANSI C. The program generates a set of explicit boolean statements involving only $assignment$, AND and XOR operations. Therefore, it essentially provides a lower level of abstraction, e.g., the gate level description as in VHDL and Verilog. The proposed design algorithm may also be modified for the construction the Karatsuba-based subquadratic space complexity multipliers.

The remainder of this article is organized as follows: In Section II, we consider asymptotic complexities for computing Toeplitz matrix-vector products based on two and three-way splits. The proposed multipliers are presented in Section III. The recursive algorithm for the efficient construction of the proposed subquadratic space complexity multiplier is introduced in Section IV. Finally, concluding remarks are made in Section V.

II. Asymptotic Complexities of Toeplitz Matrix-Vector Product

In this section, some basic noncommutative matrix-vector multiplication schemes and their asymptotic space and gate delay complexities are presented. Elements of the matrix are in $GF(2)$. These schemes will be used to design the proposed parallel multiplier in the next section.

Definition 1: An $n \times n$ Toeplitz matrix is a matrix $(m_{k,i})$, where $0 \leq i, k \leq n-1$, with the property that $m_{k,i} = m_{k-1,i-1}$, where $1 \leq i, k \leq n-1$.

$Remark$ 1: An $n \times n$ Toeplitz matrix is determined by the $2n-1$ elements of the first row and the first column, and adding two $n \times n$ Toeplitz matrices requires $2n-1$ addition operations.

A. Two-way Split or $n = 2^i$ $(i > 0)$

Assume that T is an $n \times n$ Toeplitz matrix and V an $n \times 1$ column vector. Matrix T and vector V can be split as follows:

$$T = \begin{pmatrix} T_1 & T_0 \\ T_2 & T_1 \end{pmatrix} \text{ and } V = \begin{pmatrix} V_0 \\ V_1 \end{pmatrix},$$

where T_0, T_1 and T_2 are $(n/2) \times (n/2)$ matrices and are individually in Toeplitz form, and V_0 and V_1 are $(n/2) \times 1$ column vectors.

Now the following noncommutative formula can be used to compute the Toeplitz matrix-vector product TV [2]:

$$TV = \begin{pmatrix} T_1 & T_0 \\ T_2 & T_1 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \end{pmatrix} = \begin{pmatrix} P_0 + P_2 \\ P_1 + P_2 \end{pmatrix}, \tag{1}$$

where $P_0 = (T_0 + T_1)V_1$, $P_1 = (T_1 + T_2)V_0$ and $P_2 = T_1(V_0 + V_1)$. Please note that the addition and subtraction are the same in fields of characteristic two. One important implication of (1) is that the product of an $n \times n$ Toeplitz matrix and an $n \times 1$ vector is primarily reduced to $three$ products of matrix and vector of sizes $(n/2) \times (n/2)$ and $(n/2) \times 1$.

$Remark$ 2: A straightforward matrix addition to obtain $T_0 + T_1$ and $T_1 + T_2$ requires a total of $2(n-1)$ XOR gates. Owing to the special structure of the Toeplitz matrix, some terms may be reused in computing $T_0 + T_1$ and $T_1 + T_2$. Suppose that we have obtained the summation $T_0 + T_1$. Then we only need to compute the first column of $T_1 + T_2$ since the last $n/2 - 1$ elements in the first row of $T_1 + T_2$ also appear in the first column of $T_0 + T_1$. Therefore, a total of $3n/2 - 1$ XOR gates are required to compute $T_0 + T_1$ and $T_1 + T_2$.

Let symbols $\mathcal{S}$ and $\mathcal{D}$ stand for "Space" and "Delay", respectively. We will use $\mathcal{S}_b^{\otimes}(n)$, $\mathcal{S}_b^{\oplus}(n), \mathcal{D}_b^{\otimes}(n)$ and $\mathcal{D}_b^{\oplus}(n)$ to denote the number of multiplication and addition operations, the time delays introduced by multiplication and addition operations for the case of $n = b^i$ $(i > 0)$, respectively. The following recurrence relations, which describe the algorithm complexities, can be established when this formula is used recursively to compute TV in the case of $n = 2^i$.

$$\begin{cases} \mathcal{S}_2^{\otimes}(2) = 3, \\ \mathcal{S}_2^{\otimes}(n) = 3\mathcal{S}_2^{\otimes}(n/2); \end{cases} \qquad \begin{cases} \mathcal{D}_2^{\otimes}(2) = 1, \\ \mathcal{D}_2^{\otimes}(n) = \mathcal{D}_2^{\otimes}(n/2); \end{cases}$$

$$\begin{cases} \mathcal{S}_2^{\oplus}(2) = 5, \\ \mathcal{S}_2^{\oplus}(n) = 3\mathcal{S}_2^{\oplus}(n/2) + 3n - 1; \end{cases} \text{ and } \begin{cases} \mathcal{D}_2^{\oplus}(2) = 2, \\ \mathcal{D}_2^{\oplus}(n) = \mathcal{D}_2^{\oplus}(n/2) + 2. \end{cases}$$

In order to obtain the explicit complexities of the above recurrence relations, we need the following lemmas. Proofs of these lemmas are simple and not given here.

Lemma 1: Let a, b and i be positive integers. Let $n = b^i$, $a \neq b$, and $a \neq 1$. The solution to the recurrence relations

$$\begin{cases} R_1 = e, \\ R_n = aR_{n/b} + cn + d, \end{cases}$$

is

$$R_n = a^i e + \frac{bc\,(a^i - b^i)}{a - b} + \frac{d\,(a^i - 1)}{a - 1}.$$

Lemma 2: Let b and i be positive integers, and $n = b^i$. The solution to the recurrence relations

$$\begin{cases} R_1 = 0, \\ R_n = R_{n/b} + d, \end{cases}$$

is $R_n = di = d\log_b n$.

Now it is easy to obtain the following complexity results for computing TV in the case of $n = 2^i$ $(i > 0)$:

$$\begin{cases} \mathcal{S}_2^{\otimes}(n) = n^{\log_2 3}, \\ \mathcal{S}_2^{\oplus}(n) = 5.5n^{\log_2 3} - 6n + 0.5, \\ \mathcal{D}_2^{\otimes}(n) = 1, \\ \mathcal{D}_2^{\oplus}(n) = 2\log_2 n. \end{cases}$$

B. Three-way Split or $n = 3^i \ (i > 0)$

As stated earlier, assume that T is an $n \times n$ Toeplitz matrix and V an $n \times 1$ column vector. Similar to the case $n = 2^i \ (i > 0)$, we may have a three-way split of matrix T and vector V, and obtain the following noncommutative formula which computes the Toeplitz matrix-vector product TV [2]:

$$TV = \begin{pmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \end{pmatrix} = \begin{pmatrix} P_0 + P_3 + P_4 \\ P_1 + P_3 + P_5 \\ P_2 + P_4 + P_5 \end{pmatrix},$$

where T_i $(0 \leq i \leq 4)$ are $(n/3) \times (n/3)$ Toeplitz matrices,

$$\begin{cases} P_0 = (T_0 + T_1 + T_2)V_2, \\ P_1 = (T_1 + T_2 + T_3)V_1, \\ P_2 = (T_2 + T_3 + T_4)V_0, \end{cases} \tag{2}$$

and

$$\begin{cases} P_3 = T_1(V_1 + V_2), \\ P_4 = T_2(V_0 + V_2), \\ P_5 = T_3(V_0 + V_1). \end{cases}$$

Based on Remark 1, additions of matrices in (2) may require as many as $6(\frac{2n}{3} - 1)$ XOR gates. However, by reusing repeated terms, the number of XOR gates can be considerably reduced. To this effect, we state the following lemma.

Lemma 3: Matrix additions $(T_0 + T_1 + T_2)$, $(T_1 + T_2 + T_3)$ and $(T_2 + T_3 + T_4)$ in (2) can be performed using a total of $2n - 1$ two-input XOR gates.

Proof: Let $n = 3m$ and the first row and the first column of Toeplitz matrix T be $(t_{3m-1}, t_{3m-2}, \cdots, t_0)$ and $(t_{3m-1}, t_{3m}, \cdots, t_{6m-2})^T$, respectively. There is a one-to-one correspondence between Toeplitz matrix T and polynomial $\sum_{i=0}^{6m-2} t_i x^i$. Adding two $n \times n$ Toeplitz matrices requires the same number of XOR gates as adding the corresponding polynomials of degree $2n - 2$. Therefore, we have the following polynomials corresponding to Toeplitz matrices T_0, T_1, T_2, T_3 and T_4: $q_0 = \sum_{i=0}^{2m-2} t_i x^i$, $q_1 = \sum_{i=0}^{2m-2} t_{i+m} x^i$, $q_2 = \sum_{i=0}^{2m-2} t_{i+2m} x^i$,

$q_3 = \sum_{i=0}^{2m-2} t_{i+3m} x^i$ and $q_4 = \sum_{i=0}^{2m-2} t_{i+4m} x^i$, respectively. Now we can write

$$\begin{aligned} q_0 + q_1 + q_2 &= \sum_{i=0}^{m-2} (t_i + [t_{m+i} + t_{2m+i}])x^i + (t_{m-1} + [t_{2m-1} + t_{3m-1}])x^{m-1} \\ &\quad + \sum_{i=m}^{2m-2} ([t_i + t_{m+i}] + t_{2m+i})x^i; \end{aligned} \tag{3}$$

$$\begin{aligned} q_1 + q_2 + q_3 &= \sum_{i=0}^{m-2} \{t_{m+i} + t_{2m+i} + t_{3m+i}\}x^i + ([t_{2m-1} + t_{3m-1}] + t_{4m-1})x^{m-1} \\ &\quad + \sum_{i=m}^{2m-2} (t_{m+i} + [t_{2m+i} + t_{3m+i}])x^i; \end{aligned} \tag{4}$$

$$\begin{aligned} q_2 + q_3 + q_4 &= \sum_{i=0}^{m-2} \{t_{2m+i} + t_{3m+i} + t_{4m+i}\}x^i + (t_{3m-1} + t_{4m-1} + t_{5m-1})x^{m-1} \\ &\quad + \sum_{i=m}^{2m-2} ([t_{2m+i} + t_{3m+i}] + t_{4m+i})x^i. \end{aligned} \tag{5}$$

In (3), (4) and (5), reuses of terms occur in the following five cases:

1) term $[t_{2m-1} + t_{3m-1}]$ in $q_0 + q_1 + q_2$ also appears in $q_1 + q_2 + q_3$;

2) in $q_0+q_1+q_2$, summations $[t_{m+i} + t_{2m+i}]$ $(0 \leq i \leq m-2)$ and $[t_i + t_{m+i}]$ $(m \leq i \leq 2m-2)$ are the same;

3) summations $\{t_{m+i} + t_{2m+i} + t_{3m+i}\}$ $(0 \leq i \leq m-2)$ in $q_1 + q_2 + q_3$ are the same as summations $(t_i + t_{m+i} + t_{2m+i})$ $(m \leq i \leq 2m-2)$ in $q_0 + q_1 + q_2$;

4) summations $\{t_{2m+i} + t_{3m+i} + t_{4m+i}\}$ $(0 \leq i \leq m-2)$ in $q_2 + q_3 + q_4$ are the same as summations $(t_{m+i} + t_{2m+i} + t_{3m+i})$ $(m \leq i \leq 2m-2)$ in $q_1 + q_2 + q_3$;

5) summations $[t_{2m+i} + t_{3m+i}]$ $(m \leq i \leq 2m-2)$ appear in both $q_1 + q_2 + q_3$ and $q_2 + q_3 + q_4$.

Thus, $q_0 + q_1 + q_2$, $q_1 + q_2 + q_3$ and $q_2 + q_3 + q_4$ can be computed using $2n-1$ XOR gates. □

This lemma is useful in determining the space complexity of the matrix-vector product TV. In order to determine the gate delay, we can consider any of the three summation terms on the right hand side of (2), i.e.,

$$P_2 + P_4 + P_5 = [(T_2 + T_3 + T_4)V_0] + [T_2(V_0 + V_2) + T_3(V_0 + V_1)]\,.$$

For $n = 3$, it is easy to see that computing the terms in the square brackets requires a gate delay of $T_A + 2T_X$. Therefore, $P_2 + P_4 + P_5$ may be obtained with a gate delay of $T_A + 3T_X$.

When this result and Lemma 3 are used recursively to compute TV, the following recurrence relations, which describe the algorithm complexities, can be established:

$$\begin{cases} \mathcal{S}_3^{\otimes}(3) = 6, \\ \mathcal{S}_3^{\otimes}(n) = 6\mathcal{S}_3^{\otimes}(n/3); \end{cases} \qquad \begin{cases} \mathcal{D}_3^{\otimes}(3) = 1, \\ \mathcal{D}_3^{\otimes}(n) = \mathcal{D}_3^{\otimes}(n/3); \end{cases}$$

$$\begin{cases} \mathcal{S}_3^{\oplus}(3) = 14, \\ \mathcal{S}_3^{\oplus}(n) = 6\mathcal{S}_3^{\oplus}(n/3) + 5n - 1; \end{cases} \text{ and } \begin{cases} \mathcal{D}_3^{\oplus}(3) = 3, \\ \mathcal{D}_3^{\oplus}(n) = \mathcal{D}_3^{\oplus}(n/3) + 3. \end{cases}$$

After solving these recurrence relations, we obtain the following complexities for computing TV in the case of $n = 3^i$ $(i > 0)$:

$$\begin{cases} \mathcal{S}_3^{\otimes}(n) = n^{\log_3 6}, \\ \mathcal{S}_3^{\oplus}(n) = \frac{24}{5} n^{\log_3 6} - 5n + \frac{1}{5}, \\ \mathcal{D}_3^{\otimes}(n) = 1, \\ \mathcal{D}_3^{\oplus}(n) = 3\log_3 n. \end{cases}$$

C. Dealing with Arbitrary n

In the previous two subsections, complexity results for Toeplitz matrix-vector product are given for $b = 2$ and 3. Let us denote these two primes as $p_1 = 2$ and $p_2 = 3$. It is possible to find corresponding complexities for other small primes, say $p_3 = 5, p_4 = 7, \cdots, p_w$, by transposing [2, Th6, p.17] the polynomial multiplication algorithms in [35] or [26]. Since we have complexity results for more than one prime, the following two questions arise while dealing with an arbitrary n:

(1) If $p_i p_j | n$, where $1 \leq i < j \leq w$, and complexity results for both p_i and p_j are available, how to choose a sequence of these to obtain a lower complexity scheme?

(2) If none of the p_i's $(1 \leq i \leq w)$ are factors of n, how can these complexity results be applied?

We first discuss question (1). Let primes p_j and p_i divide n and $\Omega(n, p_j, p_i)$ denote the Toeplitz matrix-vector product algorithm of size n that first applies the formula corresponding to p_i and then that corresponding to p_j. As for the XOR gate complexity, we assume that the following recurrence relations have been established for p_i- and p_j-way splits:

$$\begin{cases} \mathcal{S}^{\oplus}_{p_i}(n) = h_i \mathcal{S}^{\oplus}_{p_i}(n/p_i) + k_i n + l_i, \\ \mathcal{S}^{\oplus}_{p_j}(n) = h_j \mathcal{S}^{\oplus}_{p_j}(n/p_j) + k_j n + l_j. \end{cases}$$

Let $n = p_i p_j t$ ($t > 0$). Then the XOR gate complexities of algorithms $\Omega(n, p_j, p_i)$ and $\Omega(n, p_i, p_j)$ are described as follows:

$$\begin{cases} \mathcal{S}^{\oplus}_{\Omega(n,p_j,p_i)} = h_i[h_j \mathcal{S}^{\oplus}(t) + k_j p_j t + l_j] + k_i n + l_i, \\ \mathcal{S}^{\oplus}_{\Omega(n,p_i,p_j)} = h_j[h_i \mathcal{S}^{\oplus}(t) + k_i p_i t + l_i] + k_j n + l_j, \end{cases} \tag{6}$$

where $\mathcal{S}^{\oplus}(t)$ denotes the number of XOR gates required to compute the Toeplitz matrix-vector product of size t.

Similar to the above discussion on the XOR gate complexity, it can be easily shown that time and AND gate complexities of algorithms $\Omega(n, p_j, p_i)$ and $\Omega(n, p_i, p_j)$ are the same.

Since XOR gate complexities presented in (6) involves parameters p_i and p_j, one can make a comparison of XOR gate complexities to select one of $\Omega(n, p_j, p_i)$ and $\Omega(n, p_i, p_j)$. As an example, we now consider the special case of $n = 6t$. The XOR gate complexities of algorithms $\Omega(t, 3, 2)$ and $\Omega(t, 2, 3)$ are $63t - 4 + 18 \cdot \mathcal{S}^{\oplus}(t)$ and $66t - 7 + 18 \cdot \mathcal{S}^{\oplus}(t)$, respectively. Since both algorithms $\Omega(n, 2, 3)$ and $\Omega(n, 3, 2)$ have the same AND gate complexities and gate delays, algorithm $\Omega(n, 3, 2)$ is preferable.

With regard to question (2), where none of the p_i's $(1 \leq i \leq w)$ divide n, one may choose one of the following two solutions. The first one is to pad one zero at the end of the vector, and to extend the Toeplitz matrix from $n \times n$ to $(n+1) \times (n+1)$ by inserting zeroes at positions $(0, n)$ and $(n, 0)$. Then apply the complexity results for $p_1 = 2$, $p_2 = 3$, etc. The second one is to delete the first row and the last column of the Toeplitz matrix and the last element of the vector, and then apply the complexity results for $p_1 = 2$, $p_2 = 3$, etc. The deleted elements are processed separately.

III. New Subquadratic Multipliers

In this section, we will use the above scheme of Toeplitz matrix-vector product to design subquadratic space complexity multipliers. For representing elements of the field $GF(2^n)$, we first consider a shifted polynomial basis, which can be viewed as a generalization of the standard

polynomial basis. Let x be a root of $f(u)$ and $GF(2^n) = GF(2)[u]/(f(u))$. A shifted polynomial basis (SPB) of $GF(2^n)$ over $GF(2)$ is defined as follows [20]:

Definition 2: Let v be an integer and the ordered set $M = \{x^i | 0 \leq i \leq n-1\}$ be a polynomial basis of $GF(2^n)$ over $GF(2)$. The ordered set $x^{-v}M := \{x^{i-v} | 0 \leq i \leq n-1\}$ is called a shifted polynomial basis with respect to M.

A. Formulation Using SPB

Let $X = (x^{-v}, x^{-v+1}, \cdots, x^{n-v-1})^T$ be the column vector of SPB basis elements, $A = (a_0, a_1, \cdots, a_{n-1})^T$ be the coordinate column vector of the field element $a = x^{-v}\sum_{i=0}^{n-1} a_i x^i$, and B, C and D are defined similarly. For hardware implementation of a $GF(2^n)$ SPB parallel multiplier, one method is to form a binary $n \times n$ matrix Z, which depends on b and $f(u)$, and then perform a matrix-vector product. Namely, the product $c = ab$ may be computed as follows

$$\begin{aligned} c &= \sum_{i=0}^{n-1} a_i x^{i-v} b = (x^{-v}b, \cdots, x^{-1}b, b, xb, \cdots, x^{n-v-1}b)A \\ &= X^T (Z_0, \cdots, Z_{n-1})\, A \\ &= X^T Z A, \end{aligned} \tag{7}$$

where Z_i is the coordinate column vector of $x^{i-v}b$ with respect to the SPB ($0 \leq i \leq n-1$), and Z is an $n \times n$ matrix.

From (7), we have the matrix-vector product $C = ZA$. However, Z is not generally a Toeplitz matrix. Therefore the subquadratic scheme presented in the previous section cannot be used directly. In [3] and [4] coordinate transformation techniques were proposed. Using this technique, one may first transform Z into a Toeplitz matrix T, i.e., $T = UZ$, where U is the transform matrix. Then use the subquadratic scheme to compute the Toeplitz matrix-vector product

$$D = TA. \tag{8}$$

Finally, the result C is obtained by

$$C = U^{-1}D. \tag{9}$$

In the following, we will apply this idea to an arbitrary irreducible trinomial $f(u) = u^n + u^k + 1$ ($1 \leq k \leq n-1$) and a special type of pentanomials $f(u) = u^n + u^{k+1} + u^k + u^{k-1} + 1$ ($1 < k < n-1$), and present exact expressions of C for $GF(2^n)$ generated by these special

types of irreducible polynomials. Please note that for all practical purposes, one may only need to consider irreducible trinomials and pentanomials, since at least one of the two types of irreducible polynomials is known to exist for every values of n in the range $1 < n < 10001$. In fact, there is no known value of n for which an irreducible polynomial of weight $w < 6$ does not exist [34]. Also note that NIST has recommended five finite fields of characteristic two for the ECDSA (Elliptic Curve Digital Signature Algorithm) applications: $GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$, but no irreducible trinomials exist for three degrees, viz., 163, 283 and 571. For these three fields, we have found all pairs of (n, k) for which $f(u) = u^n + u^{k+1} + u^k + u^{k-1} + 1$ is irreducible [36]: (163, 67), (163, 69), (163, 71), (163, 92), (163, 94), (163, 96), (283, 24), (283, 133), (283, 150), (283, 259), (571, 104), (571, 230), (571, 341) and (571, 467).

We assume that the value of v is equal to k in the definition of SPB for irreducible trinomials $f(u) = u^n+u^k+1$ ($1 \leq k \leq n-1$) and irreducible pentanomials $f(u) = u^n+u^{k+1}+u^k+u^{k-1}+1$ ($1 < k < n-1$).

B. New SPB Multipliers for General Irreducible Trinomials

For irreducible trinomials, we have formed a simple transformation matrix to be used with SPB. This matrix is much simpler than what can be obtained using [3] and [4] and is given as follows

$$U = \begin{pmatrix} 0 & I_{(n-v)\times(n-v)} \\ I_{v\times v} & 0 \end{pmatrix}, \tag{10}$$

where $I_{v\times v}$ is the $v \times v$ identity matrix.

Lemma 4: Let $f(u) = u^n + u^v + 1$ ($1 \leq v \leq n-1$) be an irreducible trinomial, and Z and U be matrices defined in (7) and (10). Then $T = UZ$ is a Toeplitz matrix.

Proof: Let $g = x^{j-v}b = \sum_{i=0}^{n-1} g_i x^{i-v}$ ($0 \leq j \leq n-2$). Thus the j-th column of Z in (7), i.e., Z_j, is the column vector consisting of the SPB coordinates of element g. Then column Z_{j+1} is the coordinate column vector of element

$$\begin{aligned} xg &= \sum_{i=0}^{n-1} g_i x^{i-v+1} = \sum_{i=0}^{n-2} g_i x^{i-v+1} + g_{n-1}(x^{-v}+1) \\ &= g_{n-1}x^{-v} + \sum_{i=1}^{v-1} g_{i-1}x^{i-v} + (g_{v-1}+g_{n-1}) + \sum_{i=v+1}^{n-1} g_{i-1}x^{i-v}. \end{aligned}$$

Let $\widehat{g}$ and $\widehat{xg}$ be the two elements of $GF(2^n)$ whose SPB coordinates form columns j and $j+1$ of matrix T, respectively. Because of premultiplication of U to Z, the lower $n-v$ rows (respectively, the upper v rows) of Z become the upper $n-v$ rows (respectively, the lower v rows) of the resultant matrix $T = UZ$. Thus we can write

$$\begin{aligned}\widehat{g} &= \left(\sum_{i=0}^{v-1} g_i x^{i-v}\right) x^{n-v} + \left(\sum_{i=v}^{n-1} g_i x^{i-v}\right) x^{-v} \\ &= \sum_{i=n-2v}^{n-v-1} g_{2v-n+i} x^i + \sum_{i=-v}^{n-2v-1} g_{2v+i} x^i,\end{aligned}$$

and

$$\begin{aligned}\widehat{xg} &= \left[g_{n-1}x^{-v} + \sum_{i=1}^{v-1} g_{i-1}x^{i-v}\right] x^{n-v} + \left[(g_{v-1}+g_{n-1}) + \sum_{i=v+1}^{n-1} g_{i-1}x^{i-v}\right] x^{-v} \\ &= g_{n-1}x^{n-2v} + \sum_{i=n-2v+1}^{n-v-1} g_{2v-n-1-i}x^i + \sum_{i=-v+1}^{n-2v-1} g_{2v-1+i}x^i + (g_{v-1}+g_{n-1})x^{-v} \\ &= \sum_{i=n-2v+1}^{n-v-1} g_{2v-n-1-i}x^i + \sum_{i=-v+1}^{n-2v} g_{2v-1+i}x^i + (g_{v-1}+g_{n-1})x^{-v}.\end{aligned}$$

Careful comparison shows that elements at (i,j) and $(i+1,j+1)$ of matrix T are the same for the case $0 \le i,j \le n-2$. Therefore T is a Toeplitz matrix. □

Now we present an example to illustrate the above transformation. Let $\{x^{i-4} | 0 \le i \le 6\}$ be the SPB for the irreducible trinomial $u^7 + u^4 + 1$. Matrices Z and T are as follows:

$$Z = \begin{pmatrix} b_4+b_0 & b_3 & b_2 & b_1 & b_0 & b_6 & b_5 \\ b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 & b_0 & b_6 \\ b_6+b_2 & b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 & b_0 \\ b_0+b_3 & b_6+b_2 & b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 \\ b_1 & b_0 & b_6 & b_5 & b_4 & b_3+b_6 & b_2+b_5 \\ b_2 & b_1 & b_0 & b_6 & b_5 & b_4 & b_3+b_6 \\ b_3 & b_2 & b_1 & b_0 & b_6 & b_5 & b_4 \end{pmatrix},$$

and

$$T = \begin{pmatrix} b_1 & b_0 & b_6 & b_5 & b_4 & b_3+b_6 & b_2+b_5 \\ b_2 & b_1 & b_0 & b_6 & b_5 & b_4 & b_3+b_6 \\ b_3 & b_2 & b_1 & b_0 & b_6 & b_5 & b_4 \\ b_4+b_0 & b_3 & b_2 & b_1 & b_0 & b_6 & b_5 \\ b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 & b_0 & b_6 \\ b_6+b_2 & b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 & b_0 \\ b_0+b_3 & b_6+b_2 & b_5+b_1 & b_4+b_0 & b_3 & b_2 & b_1 \end{pmatrix}.$$

It is clear that the transformation from Z to T requires no logic gates, and the complexity to form Z was presented in [20] as follows.

Gate delay = $1T_X$;

XOR gates = $\begin{cases} n-1 & 2k \neq n, \\ n/2 & 2k = n. \end{cases}$

For U as given in (10), it is clear that C is obtained from $D = TA$ with no additional logic gates.

Table I compares the asymptotic complexity of proposed constructions with those of the existing *polynomial-based* multipliers for the trinomial $f(u) = u^n + u^k + 1$ ($1 \leq k < \lceil n/2 \rceil$), where $n = 2^t$ or 3^t. Since no irreducible binary trinomial exists for the case $8|n$, we will assume that the multiplication operation is performed in the ring $GF(2)[u]/(f(u))$, where $f(u)$ is a trinomial and $n = 2^i$ $(i > 2)$, so that the discussion of the asymptotic complexity is meaningful. The parallel multiplier presented in [32] is based on the Chinese Remainder Theorem (CRT) and Montgomery's algorithm. It allows the use of extension fields for which an irreducible trinomial or special pentanomial does not exist. Please note that the gate delay value given in [21] and [31] is $(3\log_2 n)T_X + T_A$ for the case $n = 2^t$. But one T_X gate delay may be saved for the case $n = 2$, since the gate delay for computing expressions in the square brackets of

$$(a_1x + a_0)(b_1x + b_0) = a_1b_1x^2 + ([(a_1 + a_0)(b_1 + b_0)] + [a_1b_1 + a_0b_0])x + a_0b_0$$

is $T_A + T_X$. However, this does not improve the asymptotic gate delay since the overlapping occurs when $n > 2$. Similarly, one T_X gate delay may also be saved for the case of $n = 3$. We also note that for the case $\lceil n/2 \rceil < k < n$, multipliers presented in [21] and [31] requires more

XOR gates and delays than values listed in the table since more than two reduction operations are performed [6].

TABLE I

COMPARISONS OF SOME SELECTED SUBQUADRATIC MULTIPLIERS FOR $n = b^t$

b	Multipliers	#AND	#XOR	Gate delay
2	CRT [32]	$\frac{31}{6}n^{1.6} + 4n^{1.4} + O(n^{1.2})$	$\frac{31}{6}n^{1.6} + 7n^{1.4} + O(n^{1.2})$	$O(n)T_X + 4n^{0.4}T_A$
	Karastuba [21]	$n^{\log_2 3} \approx n^{1.58}$	$6n^{\log_2 3} - 6n + k - 1$	$(3\log_2 n + 1)T_X + T_A$
	Proposed	$n^{\log_2 3}$	$5.5n^{\log_2 3} - 5n - 0.5$	$(2\log_2 n + 1)T_X + T_A$
3	CRT [32]	$\frac{31}{6}n^{1.6} + 4n^{1.4} + O(n^{1.2})$	$\frac{31}{6}n^{1.6} + 7n^{1.4} + O(n^{1.2})$	$O(n)T_X + 4n^{0.4}T_A$
	Winograd [31]	$n^{\log_3 6} \approx n^{1.63}$	$\frac{80}{15}n^{\log_3 6} - \frac{16}{3}n + k - 1$	$(4\log_3 n + 1)T_X + T_A$
	Proposed	$n^{\log_3 6}$	$\frac{72}{15}n^{\log_3 6} - 4n - \frac{4}{5}$	$(3\log_2 n + 1)T_X + T_A$

C. New SPB Multipliers for Special Pentanomials $f(u) = u^n + u^{v+1} + u^v + u^{v-1} + 1$ ($1 < v < n-1$)

For this special type of pentanomials, we transform Z into Toeplitz matrix T via the following lemma.

Lemma 5: Let $f(u) = u^n + u^{v+1} + u^v + u^{v-1} + 1$ ($1 < v < n-1$) be an irreducible pentanomial, and Z be the matrix defined in (7). Let matrix

$$U = \begin{pmatrix} 0 & I_{(n-v)\times(n-v)} + J_{(n-v)\times(n-v)} \\ I_{v\times v} + J_{v\times v}^T & 0 \end{pmatrix},$$

where $J_{v\times v}$ is a $v \times v$ matrix with the single entry $(0, v-1) = 1$ and all remaining entries being 0. Then $T = UZ$ is a Toeplitz matrix.

Proof: Let $g = x^{j-v}b = \sum_{i=0}^{n-1} g_i x^{i-v}$ ($0 \le j \le n-2$). Thus the j-th column of Z in (7), i.e., Z_j, is the column vector consisting of the SPB coordinates of element g. Then column Z_{j+1} is the coordinate column vector of element

$$xg = \sum_{i=0}^{n-1} g_i x^{i-v+1} = \sum_{i=0}^{n-2} g_i x^{i-v+1} + g_{n-1}(x^{-v} + x^{-1} + 1 + x).$$

Let $\widehat{g}$ and $\widehat{xg}$ be the two elements of $GF(2^n)$ whose SPB coordinates form columns j and $j+1$ of matrix T, respectively. Because of premultiplication of U to Z, we can write

$$\begin{aligned}\widehat{g} &= \left[\sum_{i=0}^{v-2} g_i x^{i-v} + (g_{v-1}+g_0)x^{-1}\right] x^{n-v} + \left[(g_v+g_{n-1}) + \sum_{i=v+1}^{n-1} g_i x^{i-v}\right] x^{-v} \\ &= (g_v+g_{n-1})x^{-v} + \sum_{i=v+1}^{n-1} g_i x^{i-2v} + \sum_{i=0}^{v-2} g_i x^{i+n-2v} + (g_0+g_{v-1})x^{n-v-1},\end{aligned}$$

and

$$\begin{aligned}\widehat{xg} &= \left[g_{n-1}x^{-v} + \sum_{i=0}^{v-3} g_i x^{i-v+1} + (g_{v-2}+g_{n-1}+g_{n-1})x^{-1}\right] x^{n-v} \\ &\quad + \left[(g_{v-1}+g_{n-1}+g_{n-2}) + (g_v+g_{n-1})x + \sum_{i=v+1}^{n-2} g_i x^{i-v+1}\right] x^{-v} \\ &= (g_{v-1}+g_{n-2}+g_{n-1})x^{-v} + (g_v+g_{n-1})x^{-v+1} + \sum_{i=v+1}^{n-1} g_i x^{i-2v+1} + \sum_{i=0}^{v-2} g_i x^{i+n-2v+1}.\end{aligned}$$

Careful comparison shows that elements at (i,j) and $(i+1,j+1)$ of matrix T are the same for the case $0 \le i,j \le n-2$. Therefore T is a Toeplitz matrix. □

Row operations described in the above lemma are as follows:

(1) XOR row 0 to row $v-1$;

(2) XOR row $n-1$ to row v;

(3) place the lower $n-v$ rows on the top of upper v rows.

Therefore, instead of computing $C=(c_0,c_1,\cdots,c_{n-1})^T = Z(a_0,a_1,\cdots,a_{n-1})^T$, we compute $D=(d_0,d_1,\cdots,d_{n-1})^T = T(a_0,a_1,\cdots,a_{n-1})^T$ first, where

$$d_i = \begin{cases} c_v + c_{n-1} & i=0, \\ c_{i+v} & 1 \le i \le n-v-1, \\ c_{i+v-n} & n-v \le i \le n-2, \\ c_{v-1}+c_0 & i=n-1. \end{cases}$$

Then we obtain coordinates of C from D as follows and it requires only 2 XOR gates.

$$c_i = \begin{cases} d_{n-v+i} & 0 \le i \le v-2, \\ d_{n-1}+d_{n-v} & i=v-1, \\ d_{n-v-1}+d_0 & i=v, \\ d_{i-v} & v+1 \le i \le n-1. \end{cases}$$

For the case $3 < v < (n-3)/2$, we summarize explicit expressions of the first row and the first column of matrix T below. They are obtained by applying the above transformations on the coordinate expressions of Z in [36].

$$T_{0,i} = \begin{cases} b_{2v-i} & 0 \le i \le v-1, \\ b_v + b_{n-1} & i = v, \\ b_{v-1} + b_{n-2} + b_{n-1} & i = v+1, \\ b_{2v-i} + b_{n+v-1-i} + b_{n+v-i} + b_{n+v+1-i} & v+2 \le i \le 2v, \\ b_{n+v-1-i} + b_{n+v-i} + b_{n+v+1-i} + b_{n+2v-i} & 2v+1 \le i \le n-2, \\ b_v + b_{v+1} + b_{v+2} + b_{2v+1} + b_{n-1} & i = n-1, \end{cases}$$

$$T_{i,0} = \begin{cases} b_{2v+i} & 0 \le i \le n-2v-1, \\ b_{2v-n+i} & n-2v \le i \le n-v-2, \\ b_0 + b_{v-1} & i = n-v-1, \\ b_0 + b_1 + b_v & i = n-v, \\ b_{2v-n+i} + b_{v-1-n+i} + b_{v-n+i} + b_{v+1-n+i} & n-v+1 \le i \le n-3, \\ b_0 + b_{v-3} + b_{v-2} + b_{v-1} + b_{2v-2} & i = n-2, \\ b_1 + b_{v-2} + b_{v-1} + b_v + b_{2v-1} & i = n-1. \end{cases}$$

Remark 3: Since some signals may be reused, a total of $3T_X$ delays and no more than $\lfloor \frac{5}{2}n \rfloor$ 2-input XOR gates are required to compute all elements of this Toeplitz matrix.

D. Use of Equally Spaced Pentanomials

By carefully choosing other types of irreducible pentanomials it is possible to reduce the space and time complexities for generating the Toeplitz matrix T and for obtaining the final product vector C. For example, consider a very special class of pentanomials of the form $f(u) = u^{4s} + u^{3s} + u^{2s} + u^s + 1$ of degree $n = 4s$ with $s > 0$ and $v = 2s$. Such an s-equally-spaced pentanomial is irreducible if $s = 5^i$ ($i \ge 0$) [12]. These equally-spaced irreducible pentanomials are not that abundant, however they can reduce the space and time complexities for obtaining T and C. For example, applying the following transformation matrix, the use of such an equally-spaced irreducible pentanomial of degree n requires $0.75n$ XOR gates and $1T_X$ delay for T and

$0.5n$ XOR gates and $1T_X$ delay for C

$$\begin{pmatrix} 0 & I_{\frac{n}{2}\times\frac{n}{2}} + K_{\frac{n}{2}\times\frac{n}{2}} \\ I_{\frac{n}{2}\times\frac{n}{2}} + K^T_{\frac{n}{2}\times\frac{n}{2}} & 0 \end{pmatrix},$$

where $K_{\frac{n}{2}\times\frac{n}{2}}$ is a $\frac{n}{2} \times \frac{n}{2}$ matrix with its entry at $(i, i + \frac{n}{4}) = 1$ for $0 \leq i \leq \frac{n}{4} - 1$ and all remaining entries being 0.

Slightly different complexity results, namely $0.75n$ XOR gates and $1T_X$ delay for T and $0.5n$ XOR gates and $2T_X$ delay for C, are obtained using the following transformation matrix.

$$\begin{pmatrix} 0 & 0 & I_{\frac{n}{4}\times\frac{n}{4}} & 0 \\ I_{\frac{n}{4}\times\frac{n}{4}} & 0 & 0 & I_{\frac{n}{4}\times\frac{n}{4}} \\ 0 & I_{\frac{n}{4}\times\frac{n}{4}} & 0 & 0 \\ I_{\frac{n}{4}\times\frac{n}{4}} & 0 & I_{\frac{n}{4}\times\frac{n}{4}} & 0 \end{pmatrix}.$$

E. Considerations for Other Bases

While there appears to be no scheme known to directly use the well known Karatsuba algorithm to design the subquadratic space complexity dual, weakly dual and triangular basis parallel multipliers [3], [4], [18] and [19], the proposed matrix-vector product approach can be used for these bases. Namely, for a, $b \in GF(2^n)$ let a be represented with respect to a polynomial basis and b be by the dual, weakly dual or triangular basis of the polynomial basis. Then the product $c = ab$ can be written as a matrix-vector product $C = H(a_0, a_1, \cdots, a_{n-1})^T$, where H depends on b and the field generating irreducible polynomial. Explicit expressions of entries of H and the complexity to compute H can be found in the above corresponding references. In these cases, H is not a Toeplitz matrix, but a Hankel matrix, i.e., entries at (i, j) and $(i-1, j+1)$ are equal. We may first exchange columns H_i and H_{n-1-i} for $0 \leq i < n/2$, and reverse the column vector $(a_0, a_1, \cdots, a_{n-1})^T$. Then perform the Toeplitz matrix-vector product. An alternative is to obtain the Hankel matrix-vector product formulae, which are similar to those in Section II and have the same asymptotic complexities. It is noted that no coordinate transformation is required for these parallel multipliers, which use the polynomial basis to represent input a and use the corresponding dual, weakly dual, or triangular basis to represent the other input b and the product c.

IV. Algorithm for Designing Subquadratic Space Complexity Parallel $GF(2^n)$ Multipliers

In order to design application-specific circuits, different levels of abstraction may be used to describe the hardware. Normally, a higher abstraction level provides more flexibility, and a lower one provides a better performance.

Based on previous sections, we now present a recursive design algorithm for efficient construction of the proposed subquadratic space complexity multipliers. Toeplitz matrix T are constructed first in the main program, then the recursive procedures are invoked, which output a set of explicit boolean statements. These expressions involve only $assignment$, AND and XOR operations. For example,

$T[4][0][0][0] = T[3][2][0][0] \oplus T[3][2][0][1] \oplus T[3][2][0][2]$; and

$C[9] = C[9] \oplus T[4][1][0][0] \otimes V[4][1][0]$.

Therefore a lower level abstraction, e.g., the gate level in Verilog HDL, is provided for the design of the multiplier. We note that the proposed design algorithm may also be modified for the construction of the Karatsuba-based subquadratic space complexity multipliers.

Algorithm A1: Design algorithm for the subquadratic space complexity multipliers.

Input: Field generating irreducible polynomial $f(u)$.

Output: Program for computing $c = ab$ in $GF(2^n)$.

{

Clear the output vector C;

Construct Toeplitz matrix T from B;

Construct vector V from A;

Toeplitz_mvp(n, 0, 0, 0);

Perform the coordinate transformation.

}

Subprogram: Toeplitz_mvp(INT: $fsize$, $flvl$, $fblk$, pos)

{/*

$fsize$: The size of the input vector.

$flvl$: The calling level.

$fblk$: The block number of the submatrix.

pos: The final position that this mvp will XOR to. */

INT: $cblk = 0$, $clvl = flvl + 1$;

IF $(fsize = 1)$ THEN $\{$

Print the sentence "$C[pos] = C[pos] \oplus$
$(M[flvl][fblk][0][0] \otimes V[flvl][fblk][0])$;";

return;$\}$

IF ($0 = fsize$ mod 2) THEN$\{$

// Print sentences for computing P_0

Print the sentence for computing each entry of the
$\frac{fsize}{2} \times \frac{fsize}{2}$ submatrix $T_1 + T_0$, which looks like
"$T[clvl][cblk][i][j] = T[flvl][fblk][i][j] \oplus$
$T[flvl][fblk][i][j + fsize/2]$;",
where $0 \leq i, j \leq fsize/2$;

Print the sentence for computing each entry of the
$\frac{fsize}{2} \times 1$ subvector V_1, which looks like
"$V[clvl][cblk][i] = V[flvl][fblk][i + fsize/2]$;",
where $0 \leq i \leq fsize/2$;

Toeplitz_mvp($fsize/2, clvl, cblk, pos$);

$cblk$++;

// End of computing P_0

// Print sentences for computing P_1

Print the sentence for computing each entry of the
$\frac{fsize}{2} \times \frac{fsize}{2}$ submatrix $T_1 + T_2$;

Print the sentence for computing each entry of the
$\frac{fsize}{2} \times 1$ subvector V_0;

Toeplitz_mvp($fsize/2, clvl, cblk, pos + fsize/2$);

$cblk$++;

// End of computing P_1

// Print sentences for computing P_2

Print the sentence for computing each entry of the

$\frac{fsize}{2} \times \frac{fsize}{2}$ submatrix T_1;

Print the sentence for computing each entry of the

$\frac{fsize}{2} \times 1$ subvector $V_0 + V_1$;

Print sentences for operations

"PUSH vector P_0 on the stack";

Print sentences for operations "Set vector P_0 to 0";

Toeplitz_mvp($fsize/2, clvl, cblk, pos$);

$cblk$++;

Print sentences for operations "XOR P_2 to P_1 and P_0";

// Please note that vector P_0 is on the stack,

// and vector P_2 is in the position of P_0.;

Print sentences for operations

"POP modified vector P_0 from the stack";

// End of computing P_2

} ELSE IF ($0 = fsize$ mod 3) THEN{

// Sentences for $3|n$ are similar to those for $2|n$. Omitted.

} ELSE {// Padding

Print sentences for padding zeroes after the last elements

of matrix T's first row and first column, respectively;

Print the sentence for padding a zero after the last element

of vector V;

Print the sentence "PUSH($C[pos + fsize]$);";

Toeplitz_mvp($fsize + 1, flvl, fblk, pos$);

Print the sentence "POP($C[pos + fsize]$);";

}

}

V. Conclusions

A new scheme for the subquadratic space complexity parallel multiplier has been presented. Both the space complexity and the asymptotic gate delay of the proposed multiplier are lower than those of the best existing subquadratic space complexity parallel multipliers. For practical

applications, the hybrid structure developed in [25] may also be modified to reduce the gate delay of the proposed multipliers at the cost of a slight increase in the space complexity.

A recursive design algorithm has also been proposed for efficient construction of the proposed subquadratic space complexity multipliers. It may be modified for the construction of the Karatsuba-based subquadratic space complexity multipliers.

We note that although the proposed matrix-vector product approach may be used to design the subquadratic space complexity polynomial, shifted polynomial, dual, weakly dual and triangular basis parallel multipliers, the gate delay of the SPB multiplier is always equal to or lower than those of other multipliers. For example, if we consider the irreducible trinomial $f(u) = u^n + u^{n-1} + 1$, then generating matrix T requires $2T_X$ gate delays for the SPB, but at least $(\log_2 n)T_X$ for another bases.

Finally, although the work presented here are primarily for hardware implementations, our Toeplitz matrix-vector product based approach can also be applied to software implementation using general purpose processors [37].

ACKNOWLEDGMENT

The authors thank the reviewers for their careful reading and useful comments. The work was supported in part by NSERC through grants awarded to Dr. Hasan.

REFERENCES

[1] A. Karatsuba and Y. Ofman, "Multiplication of Multidigit Numbers on Automata," *Soviet Physics-Doklady (English translation)*, vol. 7, no. 7, pp. 595-596, 1963.

[2] S. Winograd, *Arithmetic Complexity of Computations*, SIAM, 1980.

[3] M. A. Hasan and V. K. Bhargava, "Division and Bit-serial Multiplication over $GF(q^m)$," *IEE Proceedings-E*, vol. 139, no. 3, pp. 230-236, May 1992.

[4] M. A. Hasan and V. K. Bhargava, "Architecture for Low Complexity Rate-Adaptive Reed-Solomon Encoder," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 938-942, July 1995.

[5] E. D. Mastrovito, "VLSI schemes for Multiplication over Finite Field $GF(2^m)$," *Proc. 6th Int'l Conf. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC-6)*, pp. 297-309, 1988.

[6] B. Sunar and C. K. Koc, "Mastrovito Multiplier for All Trinomials," *IEEE Transactions on Computers*, vol. 48, no. 5, pp. 522-527, May 1999.

[7] C. K. Koc and B. Sunar, "Low-complexity Bit-parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields," *IEEE Transactions on Computers*, vol. 47, no. 3, pp. 353-356, Mar. 1998.

[8] A. Halbutogullari and C. K. Koc, "Mastrovito Multiplier for General Irreducible Polynomials," *IEEE Transactions on Computers*, vol. 49, no. 5, pp. 503-518, May 2000.

[9] S. O. Lee, S. W. Jung, Ch. H. Kim, J. Yoon, J. Koh, and D. Kim, "Design of Bit Parallel Multiplier with Lower time complexity," *Proc. 6th Int'l Conf. Information Security and Cryptology (ICICS 2003)*, LNCS 2971, pp. 127-139, 2004.

[10] E. Savas, A. F. Tenca, and C. K. Koc, "A Scalable and Unified Multiplier scheme for Finite Fields $GF(p)$ and $GF(2^m)$," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2000)*, C.K. Koc and C. Paar, eds., pp. 277-292, Aug. 2000.

[11] C.-C. Wang, T.-K. Truong, H.-M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed, "VLSI schemes for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE Transactions on Computers*, vol. 34, pp. 709-717, Aug. 1985.

[12] M. A. Hasan, M. Z. Wang, and V. K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields $GF(2^m)$," *IEEE Transactions on Computers*, vol. 41, no. 8, pp. 962-971, Aug. 1992.

[13] M. A. Hasan, M. Z. Wang, and V. K. Bhargava, "A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Transactions on Computers*, vol. 42, no. 10, pp. 1278-1280, Oct. 1993.

[14] A. Reyhani-Masoleh and M. A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$," *IEEE Transactions on Computers*, vol. 51, no. 5, pp. 511-520, May 2002.

[15] G. Drolet, "A New Representation of Elements of Finite Fields $GF(2^m)$ Yielding Small Complexity Arithmetic Circuits," *IEEE Transactions on Computers*, vol. 47, no. 9, pp. 938-946, Sept. 1998.

[16] R. Katti and J. Brennan, "Low Complexity Multiplication in Finite Field Using Ring Representation," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 418-427, Apr. 2003.

[17] C. Negre, "Quadrinomial Modular Arithmetic Using Modified Polynomial Basis," *Proc. Int'l Conf. Information Technology: Coding and Computing (ITCC 2005)*, vol. 1, pp. 550-555, 2005.

[18] S. T. J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ Multiplication and Division Over the Dual Basis," *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 319-327, Mar. 1996.

[19] H. Wu, M. A. Hasan, and I. F. Blake, "New Low-Complexity Bit-Parallel Finite Field Multipliers Using Weakly Dual Bases," *IEEE Transactions on Computers*, vol. 47, no. 11, pp. 1223-1234, Nov. 1998.

[20] H. Fan and Y. Dai, "Fast bit parallel $GF(2^n)$ Multiplier for All Trinomials," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 485-490, Apr. 2005.

[21] C. Paar, "A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields," *IEEE Transactions on Computers*, vol. 45, no. 7, pp. 856-861, July 1996.

[22] C. Paar, P. Fleischmann, and P. Roelse, "Efficient Multiplier schemes for Galois Fields $GF(2^{4n})$, " *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 162-170, Feb. 1998.

[23] M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over $GF(2^n)$," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2002)*, LNCS 2523, pp. 381-399, 2003.

[24] M. Jung, F. Madlener, M. Ernst, and S. Huss, "A Reconfigurable Coprocessor for Finite Field Multiplication in $GF(2^n)$," *Proc. IEEE Workshop Heterogeneous reconfigurable Systems on Chip*, 2002.

[25] C. Grabbe, M. Bednara, J. Shokrollahi, J. Teich and J. von zur Gathen, "FPGA Designs of parallel hign performance $GF(2^{233})$ Multipliers," *Proc. Int'l Symposium on Circuits and Systems (ISCAS 2003)*, vol. II, pp. 268-271, 2003.

[26] A. Weimerskirch and C. Paar, "Generalizations of the Karatsuba Algorithm for Efficient Implementations," 2003, $http://www.crypto.ruhr-uni-bochum.de/imperia/md/content/texte/kaweb.pdf$.

[27] F. Rodriguez-Henriquez and C. K. Koc, "On fully parallel Karatsuba multipliers for $GF(2^m)$," *Proc. Int'l Conf. Computer Science and Technology (CST 2003)*, pp. 405-410, 2003.

[28] A. E. Cohen and K. K. Parhi, "Implementation of scalable elliptic curve cryptosystem crypto-accelerators for $GF(2^m)$," *Proc. 13th Asilomar Conf. on Signals, Systems and Computers*, vol. 1, pp. 471 - 477, Nov. 2004.

[29] L. S. Cheng, A. M. and T. H. Yeap, "Improved FPGA Implementations of Parallel Karatsuba Multiplication over $GF(2^n)$," *Proc. 23rd Biennial Symposium on Communications*, 2006.

[30] J. von zur Gathen and J. Shokrollahi, "Efficient FPGA-based Karatsuba Multipliers for Polynomials over F_2," *Proc. 12th Workshop on Selected Areas in Cryptography (SAC 2005)*, 2005.

[31] B. Sunar, "A Generalized Method for Constructing Subquadratic Complexity $GF(2^k)$ Multipliers, " *IEEE Transactions on Computers*, vol. 53, no. 9, pp. 1097-1105, Sept. 2004.

[32] J.-C. Bajard, L. Imbert and G. A. Jullien, "Parallel Montgomery Multiplication in $GF(2^k)$ Using Trinomial Residue Arithmetic," *Proc. 17th IEEE Symposium on Computer Arithmetic (ARITH 2005)*, 2005.

[33] D. J. Bernstein. "Multidigit Multiplication for Mathematicians," Advances in Applied Mathematics, to appear.

[34] G. Seroussi, "Table of Low-Weight Binary Irreducible Polynomials," *Technical Report HPL*-98-135, *Hewlett-Packard Laboratories*, Palo Alto, Calif., Aug. 1998, Available at http://www.hpl.hp.com/techreports/98/HPL-98-135.html.

[35] P. L. Montgomery, "Five, Six, and Seven-Term Karatsuba-Like Formulae," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 362-369, Mar. 2005.

[36] H. Fan and M. A. Hasan, "Fast Bit Parallel Shifted Polynomial Basis Multipliers in $GF(2^n)$," *Technical Report, CACR 2005-34, University of Waterloo*, Waterloo, Canada, Sept. 2005.

[37] H. Fan and M. A. Hasan, "Alternative to the Karatsuba Algorithm for Software Implementation of $GF(2^n)$ Multiplication," *Technical Report, CACR 2006-13, University of Waterloo*, Waterloo, Canada, May 2006.