Subquadratic Multiplication Using Optimal Normal Bases

H. Fan and M. A. Hasan

August 6, 2006

Abstract

Based on a recently proposed Toeplitz matrix-vector product approach, a subquadratic computational complexity scheme is presented for multiplications in binary extended finite fields using Type I and II optimal normal bases.

Index Terms

Finite field, subquadratic computational complexity multiplication, normal basis, optimal normal basis.

I. INTRODUCTION

Among different types of bases for representing elements of $GF(2^n)$, the normal basis has received considerable attention because squaring in normal bases is simply a cyclic shift of the coordinates of the element and, thus, it has found applications in computing multiplicative inverses and exponentiations. One of the most important advances in the normal basis multiplication is the discovery of the two types (Type I and Type II) of optimal normal bases (ONB) in 1987 [1]. The computational complexity (i.e., the number of arithmetic operations in the ground

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

field $GF(2)$) of a $GF(2^n)$ multiplication using an ONB is $O(n^2)$, while that using an arbitrary normal basis is usually greater than $O(n^2)$.

Recently, a new approach to subquadratic complexity multiplications in $GF(2^n)$ has been presented [2]. It takes advantage of the optimal Toeplitz matrix-vector product formulae, and can be used to design subquadratic space complexity multipliers using various bases, namely, polynomial, shifted polynomial, dual, weakly dual and triangular basis. In this work, we apply the Toeplitz matrix-vector product approach to design subquadratic computational complexity scheme for multiplications using Type I and II ONB. To the best of our knowledge, this is the first subquadratic scheme for normal basis multiplication.

Below, we first summarize the asymptotic complexities of Toeplitz matrix-vector product formulae for $n = 2^i$ and $n = 3^i$ $(i > 0)$. Then, we present our multiplication schemes using Type I and II ONB.

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 introduced [2]. A Toeplitz matrix is defined as follows:

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 $1 \leq i, k \leq n-1$.

Let $n = 2^i$ $(i > 0)$, T be an $n \times n$ Toeplitz matrix and V an $n \times 1$ column vector. Then the following noncommutative formula can be used to compute the Toeplitz matrix-vector product TV [3]:

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

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, $P_0 = (T_0 + T_1)V_1$, $P_1 = (T_1 + T_2)V_0$ and $P_2 = T_1(V_0 + V_1)$.

Similar to the case $n = 2^i$ $(i > 0)$, we may have a three-way split of matrix T and vector V for $n = 3^i$ $(i > 0)$, and obtain the following noncommutative formula which computes the Toeplitz matrix-vector product TV [3]:

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

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

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

Formulae (1) and (2) may be used recursively to compute the Toeplitz matrix-vector product TV. Their complexities are summarized in Table I for bit parallel implementations, where one AND and one XOR gate corresponds to one multiplication and one addition over $GF(2)$, respectively, and T_A and T_X are delays due to one AND and one XOR 2-input gate, respectively.

TABLE I

COMPLEXITIES OF TOEPLITZ MATRIX-VECTOR PRODUCT FOR $n = b^i$

b	#AND	#XOR	Gate delay
2	$n^{\log_2 3}$	$5.5n^{\log_2 3} - 6n + 0.5$	$(2\log_2 n)T_X + T_A$
3	$n^{\log_3 6}$	$\frac{24}{5}n^{\log_3 6} - 5n + \frac{1}{5}$	$(3\log_2 n)T_X + T_A$

III. NEW SUBQUADRATIC ONB MULTIPLIERS

We now apply the above Toeplitz matrix-vector product approach to design subquadratic complexity multiplication scheme using Type I and II ONB. It is well known that an ONB of $GF(2^n)$ over $GF(2)$ exists if and only if the following conditions are met [1], [4].

Theorem 1: Suppose $n+1$ is a prime and 2 is primitive in $\mathbb{Z}_{n+1}$. Then the n nonunit $(n+1)$th roots of unity form a Type I ONB of $GF(2^n)$ over $GF(2)$.

Theorem 2: Let $2n+1$ be a prime and assume that either

(1) 2 is primitive in $\mathbb{Z}_{2n+1}$, or

(2) $2n+1 \equiv 3$ (mod 4) and 2 generates the quadratic residues in $\mathbb{Z}_{2n+1}$.

Then $x = y + y^{-1}$ generates a Type II ONB of $GF(2^n)$ over $GF(2)$, where y is a primitive $(2n+1)$st root of unity in $GF(2^{2n})$.

In some cryptosystems, Type I ONB are avoided for security reasons. For practical purposes, e.g., $n < 2000$, Type II ONB are more abundant than Type I ONB [5]. Properties of Type I and II ONB can be found in various references, e.g., [1], [5], [6], [7], [8], [9], [10], [11] and [12]. Based on some of these properties, below we present two subquadratic computational complexity schemes for multiplications in $GF(2^n)$ using Type I and II ONB.

A. Formulation for Type I ONB

Let $\hat{X} = \{x^{2^0}, x^{2^1}, \cdots, x^{2^{n-1}}\}$ be a Type I ONB of $GF(2^n)$ over $GF(2)$. In the following, we will also use symbol $\hat{X}$ to denote the column vector $\hat{X} = (x^{2^0}, x^{2^1}, \cdots, x^{2^{n-1}})^T$. Since 2 is a primitive root of prime $n+1$, we know that

$$\{2^0, 2^1, \cdots, 2^{n-1}\} = \{1, 2, \cdots, n\}. \tag{3}$$

Therefore, $X = \{x^1, x^2, \cdots, x^n\}$ is also a basis of $GF(2^n)$ over $GF(2)$. Similarly, we will use symbol X to denote the column vector $X = (x^1, x^2, \cdots, x^n)^T$.

Given a field element a represented in the above two bases, i.e., $a = \hat{A}^T\hat{X} = \sum_{i=0}^{n-1} \hat{a}_i x^{2^i}$ and $a = A^T X = \sum_{i=1}^{n} a_i x^i$, where $\hat{A} = (\hat{a}_0, \hat{a}_1, \cdots, \hat{a}_{n-1})^T$ and $A = (a_1, a_2, \cdots, a_n)^T$, it is easy to obtain the following coordinate transformation formula:

$$a_{2^i} = \hat{a}_i, \tag{4}$$

where $0 \leq i \leq n-1$ and the subscript 2^i is to be reduced modulo $n+1$. From (3) and (4), we know that A is a permutation of $\hat{A}$. Therefore, the basis conversion operation between X and $\hat{X}$ may be performed in VLSI without using any logic gates. Now we use basis X to design a subquadratic complexity multiplication scheme.

Similar to a, define field element b with respect to basis X. Then the multiplication ab may be performed as follows.

$$\begin{aligned} ab &= \sum_{i=1}^{n} a_i x^i b = (x^1 b, x^2 b, \cdots, x^n b) A \\ &= X^T (Z_1, \cdots, Z_n) A \\ &= X^T Z A, \end{aligned} \tag{5}$$

where Z_i ($1 \leq i \leq n$) is the column vector corresponding to the coordinates of field element $x^i b$ with respect to basis X, and Z is an $n \times n$ matrix. Using the identity $x^{n+1} = 1 = \sum_{j=1}^{n} x^j$, we obtain the following explicit expression of Z_i:

$$\begin{aligned} Z_i &= x^i \sum_{j=1}^{n} b_j x^j = \sum_{j=1}^{n} b_j x^{i+j} = \sum_{k=i+1}^{n+i} b_{k-i} x^k \\ &= \sum_{k=i+1}^{n} b_{k-i} x^k + \sum_{k=n+1}^{n+i} b_{k-i} x^k \\ &= \sum_{k=i+1}^{n} b_{k-i} x^k + \sum_{k=0}^{i-1} b_{k+n+1-i} x^k \\ &= \left(\sum_{k=1}^{i-1} b_{k+n+1-i} x^k + \sum_{k=i+1}^{n} b_{k-i} x^k \right) + b_{n+1-i} \sum_{j=1}^{n} x^j. \end{aligned} \tag{6}$$

From (6), we have the following decomposition of matrix $Z = Z_1 + Z_2$:

$$Z = \begin{pmatrix} 0 & b_n & b_{n-1} & \cdots & b_3 & b_2 \\ b_1 & 0 & b_n & \cdots & b_4 & b_3 \\ b_2 & b_1 & 0 & \cdots & b_5 & b_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n-2} & b_{n-3} & b_{n-4} & \cdots & 0 & b_n \\ b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_1 & 0 \end{pmatrix} + \begin{pmatrix} b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_2 & b_1 \end{pmatrix}.$$

Therefore, matrix-vector product ZA may be computed via $ZA = Z_1A+Z_2A$. Clearly, computing Z_2A requires only n multiplications and $n-1$ additions over the ground field $GF(2)$. The Toeplitz matrix-vector product Z_1A may be computed using the formulae in the previous section. The complexities of the resulting multiplication scheme are summarized in the upper half of Table II.

B. Formulation for Type II ONB

Following the notations in Theorem 2, let $\hat{X} = \{x^{2^0}, x^{2^1}, \cdots, x^{2^{n-1}}\}$ be a Type II ONB of $GF(2^n)$ over $GF(2)$. In the following we will also use symbol $\hat{X}$ to denote the column vector $\hat{X} = (x^{2^0}, x^{2^1}, \cdots, x^{2^{n-1}})^T$. Let $x_i = y^i + y^{-i}$ $(0 \leq i \leq n)$. From [10], we can write that

$$\{x^{2^0}, x^{2^1}, \cdots, x^{2^{n-1}}\} = \{x_1, x_2, \cdots, x_n\}. \tag{7}$$

Therefore, $X = \{x_1, x_2, \cdots, x_n\}$ is also a basis of $GF(2^n)$ over $GF(2)$. Similarly, we will use symbol X to denote the column vector $X = (x_1, x_2, \cdots, x_n)^T$.

Given a field element a represented in the above two bases, i.e., $a = \hat{A}^T\hat{X} = \sum_{i=0}^{n-1} \hat{a}_i x^{2^i}$ and $a = A^TX = \sum_{i=1}^{n} a_i x_i$, where $\hat{A} = (\hat{a}_0, \hat{a}_1, \cdots, \hat{a}_{n-1})^T$ and $A = (a_1, a_2, \cdots, a_n)^T$, the coordinate transformation formula between these two bases is given as follows [10]:

$$a_{s(2^i)} = \hat{a}_i, \tag{8}$$

where $0 \leq i \leq n-1$ and $s(j)$ is defined as the unique integer such that $0 \leq s(j) \leq n$ and $j \equiv s(j)$ (mod $2n+1$) or $j \equiv -s(j)$ (mod $2n+1$).

From (7) and (8), we know that A is a permutation of $\hat{A}$. Therefore, the basis conversion operation between X and $\hat{X}$ may be performed in VLSI without using any logic gates. Similar to the case of Type I ONB, we may compute ab via a matrix-vector product ZA using basis X. The matrix Z can be decomposed as the summation of two matrices i.e., $Z = Z_1 + Z_2$ [11]

[12]:

$$Z = \begin{pmatrix} b_2 & b_3 & b_4 & \cdots & b_{n-1} & b_n & b_n \\ b_3 & b_4 & b_5 & \cdots & b_n & b_n & b_{n-1} \\ b_4 & b_5 & b_6 & \cdots & b_n & b_{n-1} & b_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ b_{n-1} & b_n & b_n & \cdots & b_5 & b_4 & b_3 \\ b_n & b_n & b_{n-1} & \cdots & b_4 & b_3 & b_2 \\ b_n & b_{n-1} & b_{n-2} & \cdots & b_3 & b_2 & b_1 \end{pmatrix} + \begin{pmatrix} 0 & b_1 & b_2 & \cdots & b_{n-3} & b_{n-2} & b_{n-1} \\ b_1 & 0 & b_1 & \cdots & b_{n-4} & b_{n-3} & b_{n-2} \\ b_2 & b_1 & 0 & \cdots & b_{n-5} & b_{n-4} & b_{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ b_{n-3} & b_{n-4} & b_{n-5} & \cdots & 0 & b_1 & b_2 \\ b_{n-2} & b_{n-3} & b_{n-4} & \cdots & b_1 & 0 & b_1 \\ b_{n-1} & b_{n-2} & b_{n-3} & \cdots & b_2 & b_1 & 0 \end{pmatrix}.$$

Here, Z_1 is a Hankel matrix, i.e., entries at (i, j) and $(i-1, j+1)$ are equal. In order to compute the Hankel matrix-vector product Z_1A, we may first exchange columns H_i and H_{n-1-i} for $0 \leq i < n/2$, and reverse the column vector $A = (a_1, a_2, \cdots, a_n)^T$. Then perform the Toeplitz matrix-vector product. Therefore, two Toeplitz matrix-vector products are used to obtain the matrix-vector product ZA. The complexities of the resulting multiplication scheme are summarized in the lower half of Table II.

IV. Conclusions

Taking advantage of the simple conversion relationship in (3) and (7), for which no logic gates is required to perform the basis conversions, we have presented a multiplication scheme of subquadratic computational complexity using ONB. However, it is still an open problem to design subquadratic computational complexity multiplication scheme for general normal bases.

TABLE II

COMPLEXITIES OF SUBQUADRATIC ONB MULTIPLIERS FOR $n = b^i$

ONB	b	#AND	#XOR	Gate delay
Type	2	$n^{\log_2 3} + n$	$5.5n^{\log_2 3} - 4n - 0.5$	$(2\log_2 n + 1)T_X + T_A$
I	3	$n^{\log_3 6} + n$	$\frac{24}{5}n^{\log_3 6} - 3n - \frac{4}{5}$	$(3\log_2 n + 1)T_X + T_A$
Type	2	$2n^{\log_2 3}$	$11n^{\log_2 3} - 12n + 1$	$(2\log_2 n + 1)T_X + T_A$
II	3	$2n^{\log_3 6}$	$\frac{48}{5}n^{\log_3 6} - 10n + \frac{2}{5}$	$(3\log_2 n + 1)T_X + T_A$

ACKNOWLEDGMENT

This research was supported by NSREC Discovery and Strategic grants awarded to Dr. Hasan.

REFERENCES

[1] R. Mullin, I. Onyszchuk, S.A. Vanstone, and R. Wilson, "Optimal Normal Bases in $GF(p^n)$," *Discrete Applied Math.*, vol. 22, pp. 149-161, 1988/1989.

[2] H. Fan and M. A. Hasan, "A New Approach to Subquadratic Space Complexity Parallel Multipliers for Extended Binary Fields," *IEEE Transactions on Computers*, accepted. A preliminary version is available at: http://www.cacr.math.uwaterloo.ca/techreports/2006/cacr2006-02.pdf

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

[4] S. Gao and Jr. H.W. Lenstra, "Optimal Normal Bases," *Design, Codes and Cryptography*, vol. 2, pp. 315-323, 1992.

[5] S. Gao, "Normal Bases over Finite Fields", PhD thesis, University of Waterloo, 1993.

[6] 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.

[7] 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.

[8] H. Fan, "Simple multiplication algorithm for a class of $GF(2^n)$," *IEE Electronics Letters*, vol. 32, no.7, pp.636-637, 1996.

[9] 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.

[10] S. Gao and S. Vanstone, "On Orders of Optimal Normal Basis Generators," *Math. Computation*, vol. 64, no. 2, pp. 1227-1233, 1995.

[11] B. Sunar and C. K. Koc, "An Efficient Optimal Normal Basis Type II Multiplier," *IEEE Transactions on Computers*, vol. 50, no. 1, pp. 83-87, Jan. 2001.

[12] C. Lee and C. Chang "Low-complexity Linear Array Multiplier for Normal Basis of Type-II," *Proc. IEEE International Conf. Multimedia and Expo*, pp. 1515-1518, 2004.