

Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields

Ç. K. Koç and B. Sunar

Abstract— We present a new low-complexity bit-parallel canonical basis multiplier for the field $\text{GF}(2^m)$ generated by an all-one-polynomial. The proposed canonical basis multiplier requires $m^2 - 1$ XOR gates and m^2 AND gates. We also extend this canonical basis multiplier to obtain a new bit-parallel normal basis multiplier.

Keywords— Finite fields, multiplication, normal basis, canonical basis, all-one-polynomial.

I. INTRODUCTION

The arithmetic operations in the Galois field $\text{GF}(2^m)$ have several applications in coding theory, computer algebra, and cryptography [6], [4]. In these applications, time and area efficient algorithms and hardware structures are desired for addition, multiplication, squaring, and exponentiation operations. The performance of these operations is closely related to the representation of the field elements. An important advance in this area has been the introduction of the Massey-Omura algorithm [7], which is based on the normal basis representation of the field elements. One advantage of the normal basis is that the squaring of an element is computed by a cyclic shift of the binary representation. Efficient algorithms for the multiplication operation in the canonical basis have also been proposed [5], [3]. The space and time complexities of these bit-parallel canonical basis multipliers are much less than those of the Massey-Omura multiplier.

In this paper, we present an alternative design for multiplication in the canonical basis for the field $\text{GF}(2^m)$ generated by an all-one-polynomial (AOP). The time complexity of our design is significantly less than similar bit-parallel multiplier designs for the canonical basis [5], [3], [1]. Furthermore, we use the proposed canonical basis multiplier to design a normal basis multiplier, whose space and time complexities are nearly the same as those of the modified Massey-Omura multiplier [2] given for the field $\text{GF}(2^m)$ with an AOP. Nevertheless, the proposed

normal basis multiplier is based on a different construction from the ones already known, and it has certain advantages.

II. CANONICAL BASIS MULTIPLIER

It is customary to view the field $\text{GF}(2^m)$ as an m -dimensional vector space defined over the ground field $\text{GF}(2)$. We need a set of m linearly independent elements from $\text{GF}(2^m)$ in order to represent the elements of $\text{GF}(2^m)$. This set serves as the basis of the vector space. A basis of the form $S = \{1, \alpha, \alpha^2, \dots, \alpha^{m-1}\}$, where $\alpha \in \text{GF}(2^m)$ is a root of the generating polynomial of degree m , is called a canonical basis. In order to reduce the complexity of the field multiplication, special classes of irreducible polynomials have been suggested [3], [5]. In particular, the AOP $p(x) = 1 + x + x^2 + \dots + x^m$ has been shown to be very useful. This polynomial is irreducible, and thus, generates the field $\text{GF}(2^m)$, if and only if $m + 1$ is prime and 2 is primitive modulo $m + 1$ [6]. For $m \leq 100$, the AOP is irreducible for the following values of m : 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82, and 100.

We now briefly describe the Mastrovito multiplier [5] for computing the products of two elements A and B in $\text{GF}(2^m)$ expressed in the canonical basis, which are respectively represented as

$$A = \sum_{i=0}^{m-1} a_i x^i = [a_0 a_1 \dots a_{m-1}]^T$$

$$B = \sum_{i=0}^{m-1} b_i x^i = [b_0 b_1 \dots b_{m-1}]^T.$$

The Mastrovito multiplier uses two matrices in the design process. The $(m - 1) \times m$ basis reduction matrix $Q = [q_{ij}]$ satisfies the equality

$$\begin{bmatrix} x^m \\ x^{m+1} \\ \vdots \\ x^{2m-2} \end{bmatrix} = Q \begin{bmatrix} 1 \\ x \\ \vdots \\ x^{m-1} \end{bmatrix}$$

The authors are with the Department of Electrical & Computer Engineering, ECE Building 220, Oregon State University, Corvallis, OR 97331. E-mail:{koc,sunar}@ece.orst.edu.

This research was supported in part by Intel Corporation.

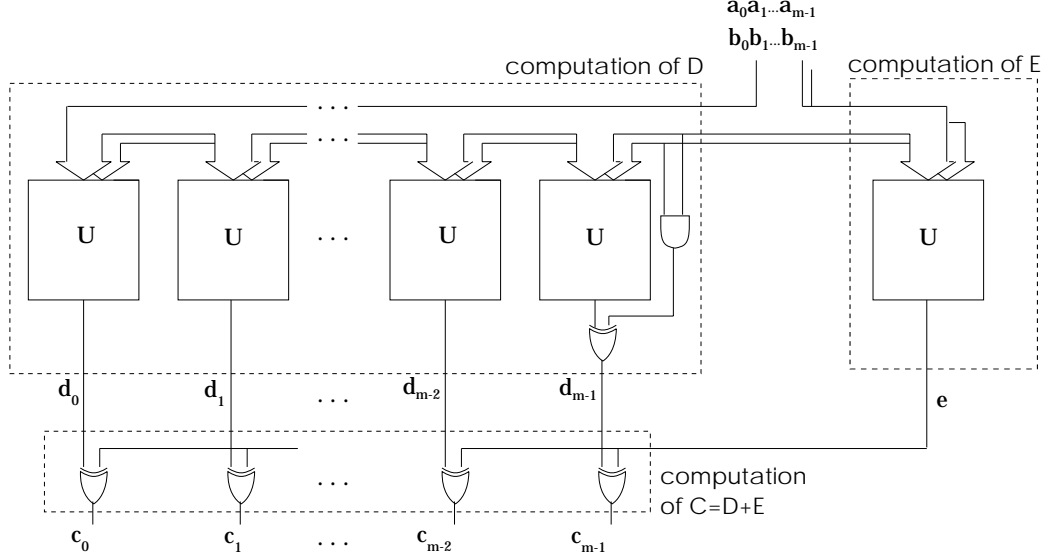


Fig.1. The proposed canonical basis multiplier.

The $m \times m$ product matrix $Z = f(A, Q) = [z_{ij}]$ is defined as

$$z_{ij} = \begin{cases} a_i, & \text{for } j = 0; i = 0, \dots, m-1, \\ u(i-j)a_{i-j} + \sum_{t=0}^{j-1} q_{j-1-t,i} a_{m-1-t}, & \text{for } j = 1, \dots, m-1; i = 0, \dots, m-1, \end{cases} \quad (1)$$

where the step function $u(t)$ is defined as

$$u(t) = \begin{cases} 1 & \text{for } t \geq 0, \\ 0 & \text{else.} \end{cases}$$

The product $C = AB$ is found by multiplying the matrix Z by the vector B in the ground field $\text{GF}(2)$. The Mastrovito algorithm directly computes this product $C = ZB$.

We introduce a new canonical basis multiplication algorithm for the field $\text{GF}(2^m)$ generated using an AOP by decomposing the matrix Z into the matrices Z_1 and Z_2 as $Z = Z_1 + Z_2$. The idea of decomposing a matrix has proved to be useful in many similar designs [2]. In order to construct these matrices, we first write the matrix equality (II) for the matrix Q in the field $\text{GF}(2^m)$ with an AOP using the identity $x^{m+1} = 1$ as

$$\begin{bmatrix} x^m \\ x^{m+1} \\ x^{m+2} \\ \vdots \\ x^{2m-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^{m-1} \end{bmatrix} \quad (2)$$

Using the definition (2) of Q and the definition (1) of Z , we construct the product matrix Z for the field $\text{GF}(2^m)$ with an AOP as the sum of two matrices Z_1 and Z_2 which are given as follows:

$$Z_1 = \begin{bmatrix} a_0 & 0 & a_{m-1} & a_{m-2} & \cdots & a_2 \\ a_1 & a_0 & 0 & a_{m-1} & \cdots & a_3 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m-2} & a_{m-3} & a_{m-4} & a_{m-5} & \cdots & 0 \\ a_{m-1} & a_{m-2} & a_{m-3} & a_{m-4} & \cdots & a_0 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 \\ 0 & a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_1 \end{bmatrix}.$$

In order to compute $C = ZB = (Z_1 + Z_2)B$, we first compute $D = Z_1B$ and $E = Z_2B$ in parallel, and then compute the result $C = D + E$. The product of the last row of Z_1 and B is computed using the rightmost U circuit with two additional gates which take care of the nonzero element of the last row of Z_1 . The architecture of the canonical basis multiplier is shown in Fig. 1. The module which computes the vector $D = Z_1B$ consists of m identical U circuits, an AND, and an XOR gate. The circuit U computes the innerproduct of two vectors of length $m-1$. Since one element in each row of Z_1 is zero, except in the last row, the innerproduct operation needs to be of length $m-1$. The vector A is shifted according to the place of the zero element in each row of Z_1 , while

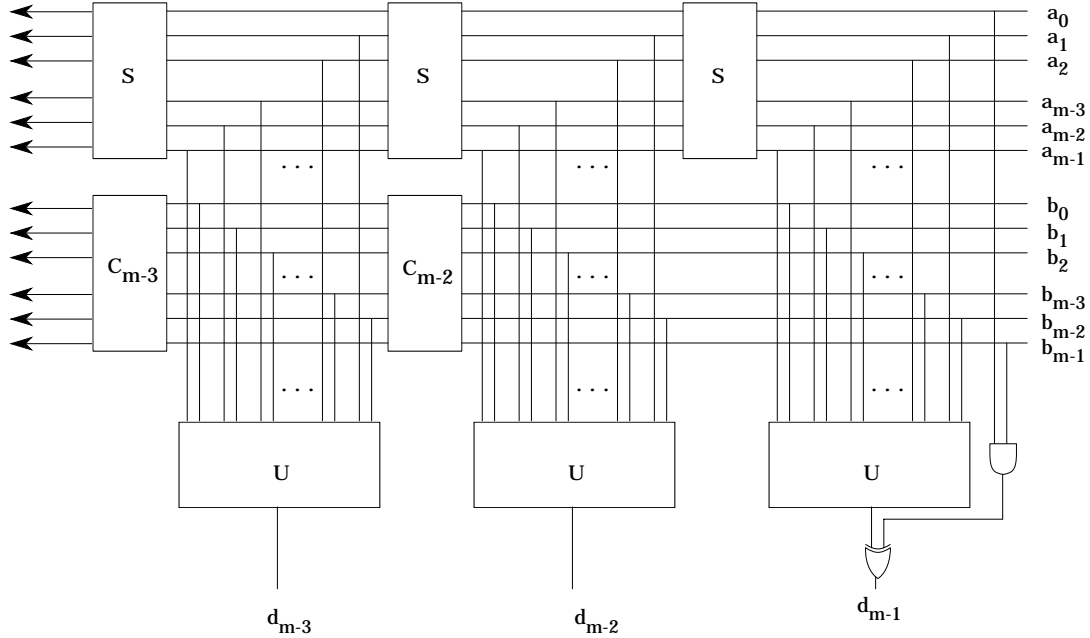


Fig. 2. The connection diagram for computing D .

the vector B is fed to the i th U module by skipping the i th bit. The connection diagram of the part of the multiplier computing D is shown in Fig. 2. The basic rewiring modules used in the connection diagram are defined in Fig. 3.

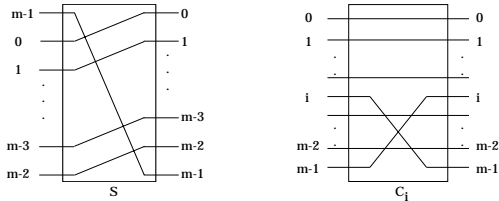


Fig. 3. The rewiring modules used in the connection diagram.

The structure of the module U is very simple. The innerproduct of two vectors is computed by first generating the products in parallel using AND gates, and then by adding the partial products using a binary XOR tree. In order to generate the products $m - 1$ AND gates are needed, whereas $m - 2$ XOR gates are used to accumulate the products. The depth of the binary XOR tree is given as $\lceil \log_2(m-1) \rceil$. The total delay of the circuit U is equal to $T_A + \lceil \log_2(m-1) \rceil T_X$, where T_A and T_X are the delays of AND and XOR gates, respectively. The computation of d_{m-1} requires an additional XOR gate delay. Thus, the computation of D requires a total of $T_A + (1 + \lceil \log_2(m-1) \rceil) T_X$ delays.

In order to compute $E = Z_2B$, we need a single U module with inputs according to the definition of Z_2 given above. Since Z_2 has identical rows, the computation of Z_2B is accomplished by computing the innerproduct of a row of Z_2 and the vector B , and then replicating this resulting bit m times, i.e., $E = [eee \dots e]$, where e is repeated m times. After $E = Z_2B$ is computed, the result $C = Z_1B + Z_2B = D + E$ is obtained using m XOR gates, as shown in the bottom part of Fig. 1.

The proposed canonical basis multiplier architecture requires a total of $(m-2)(m+1)+1+m = m^2-1$ XOR gates and $(m-1)(m+1)+1 = m^2$ AND gates. The total delay of the circuit is found as $T_A + (2 + \lceil \log_2(m-1) \rceil) T_X$. These time and space complexity values are computed assuming only 2-input gates are available.

III. NORMAL BASIS MULTIPLIER

A basis of the form

$$N = \{\beta, \beta^2, \beta^3, \dots, \beta^{2^m-1}\},$$

is called a normal basis, where $\beta \in \text{GF}(2^m)$ and m is the degree of the generating polynomial. The root of an irreducible AOP has the following property that $\beta^{m+1} = 1$. Furthermore, if the generating polynomial is an AOP and also if 2 is primitive in Z_{m+1} , then we have

$$N = \{\beta, \beta^2, \beta^3, \dots, \beta^m\}. \quad (3)$$

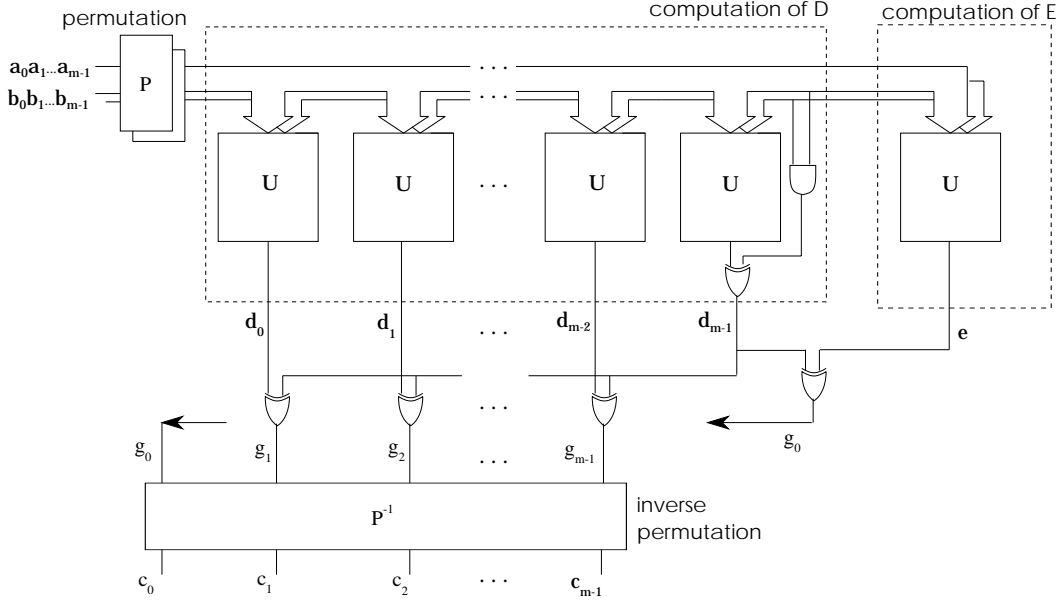


Fig. 4. The proposed normal basis multiplier.

For further information, the reader is referred to [6, page 99]. Since the set (3) is also a basis, it can be used to represent the elements of $GF(2^m)$. This basis is a shifted version of the canonical basis. An element of $GF(2^m)$ in the normal basis representation can easily be converted to the shifted canonical representation. This is accomplished using a permutation of the binary representation. With the help of the identity $\beta^{m+1} = 1$, we perform the conversion

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = \sum_{i=1}^m a'_i \beta^i,$$

using the permutation P given as

$$a'_{2^i \bmod (m+1)} = a_i \quad \text{for } i = 0, 1, \dots, m-1.$$

In order to perform a normal basis multiplication, we take the inputs A and B represented in the normal basis, convert them to the shifted canonical basis using the permutation P , and then perform a canonical basis multiplication. At the end of this computation, we obtain $F = AB/\beta^2$ represented in the canonical basis as

$$F = f_0 + f_1\beta + f_2\beta^2 + \dots + f_{m-1}\beta^{m-1}.$$

Note that the values f_i are the outputs of the canonical basis multiplier shown in Fig. 1, and therefore, we have $f_i = d_i + e$ for $i = 0, 1, \dots, m-1$. We then multiply F by β^2 , and obtain $G = F\beta^2$ as

$$G = (d_0 + e)\beta^2 + (d_1 + e)\beta^3 + \dots + (d_{m-1} + e)\beta^{m+1}.$$

We now need to represent this number in the shifted canonical basis. Since

$$\beta^{m+1} = \beta + \beta^2 + \dots + \beta^m$$

the coefficient $(d_{m-1} + e)$ is added to the coefficients of all the other terms. We can write the final expression as

$$\begin{aligned} G &= (d_{m-1} + e)\beta + (d_0 + e + d_{m-1} + e)\beta^2 + \\ &\quad (d_1 + e + d_{m-1} + e)\beta^3 + \dots \\ &\quad \dots + (d_{m-2} + e + d_{m-1} + e)\beta^m \\ &= (d_{m-1} + e)\beta + (d_0 + d_{m-1})\beta^2 + \\ &\quad (d_1 + d_{m-1})\beta^3 + \dots + (d_{m-2} + d_{m-1})\beta^m, \end{aligned}$$

which gives $g_0 = d_{m-1} + e$ and $g_i = d_{i-1} + d_{m-1}$ for $i = 1, 2, \dots, m-1$. Thus, we have obtained the representation of the number G in the shifted canonical basis. We now apply the inverse of the permutation P to G and obtain the bits of the number C in the normal basis. The architecture of the normal basis multiplier is given in Fig. 4. It is very similar to that of the canonical basis multiplier.

The implementation of the permutation and inverse permutation operations are accomplished by wiring. Therefore, the normal basis multiplier requires exactly the same number AND and XOR gates as that of the canonical basis multiplier in Fig. 1. Furthermore, the time complexity of the normal basis multiplier is equal to that of the canonical basis multiplier.

TABLE 1
COMPARING CANONICAL BASIS MULTIPLIERS WITH GENERATING AOPs.

	XOR Gates	AND Gates	Delay
Itoh-Tsujii [3]	$m^2 + 2m$	$m^2 + 2m + 1$	$T_A + \lceil \log_2 m + \log_2(m + 2) \rceil T_X$
Hasan-Wang-Bhargava [1]	$m^2 + m - 2$	m^2	$T_A + (m + \lceil \log_2(m - 1) \rceil) T_X$
Proposed design (Fig. 1)	$m^2 - 1$	m^2	$T_A + (2 + \lceil \log_2(m - 1) \rceil) T_X$

TABLE 2
COMPARING NORMAL BASIS MULTIPLIERS WITH GENERATING AOPs.

	XOR Gates	AND Gates	Delay
Massey-Omura [7]	$2m^2 - 2m$	m^2	$T_A + (1 + \lceil \log_2(m - 1) \rceil) T_X$
Hasan-Wang-Bhargava [2]	$m^2 - 1$	m^2	$T_A + (1 + \lceil \log_2(m - 1) \rceil) T_X$
Proposed design (Fig. 4)	$m^2 - 1$	m^2	$T_A + (2 + \lceil \log_2(m - 1) \rceil) T_X$

IV. CONCLUSIONS

The time complexity of the proposed canonical basis multiplier is significantly less than previously proposed similar multipliers for the field $\text{GF}(2^m)$ generated by an AOP. The structure of the canonical basis multiplier is very regular: it consists of $m + 1$ identical modules, and some additional XOR and AND gates. It is more regular than the Mastrovito multiplier, and requires significantly less gate delays. The proposed canonical basis multiplier requires m^2 AND gates and $m^2 - 1$ XOR gates. The Mastrovito multiplier requires $m^2 - 1$ XOR gates and m^2 AND gates, and has a delay less than $T_A + 2\lceil \log_2 m \rceil T_X$ if the generating polynomial is a primitive trinomial of the form $x^m + x + 1$ [5], [8]. The XOR and AND complexities of the Mastrovito multiplier for a general trinomial or an AOP are not known. However, the number of XOR gates for a general trinomial is conjectured to be $\geq m^2 - 1$ in [8].

The normal basis multiplier proposed here and the modified Massey-Omura multiplier [2] require the same number of XOR and AND gates, which is about half of the number of gates required by the Massey-Omura multiplier for the field $\text{GF}(2^m)$ with an AOP. The design proposed in this paper requires only 1 more XOR delay than the modified Massey-Omura multiplier. Nevertheless, it is an alternative design, and is based on an entirely different construction. Another advantage is that it is highly modular. Since the proposed normal basis multiplier is based on a canonical basis multiplier, any advances made in canonical basis multiplication using AOPs can be utilized in this design to further reduce the complexity or timing requirements.

REFERENCES

- [1] M. A. Hasan, M. Z. Wang, and V. K. Bhargava. "Modular construction of low complexity parallel multipliers for

- a class of finite fields $\text{GF}(2^m)$," *IEEE Transactions on Computers*, vol. 41, no. 8, pp. 962-971, August 1992.
- [2] 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, October 1993.
- [3] T. Itoh and S. Tsujii. "Structure of parallel multipliers for a class of finite fields $\text{GF}(2^m)$," *Information and Computation*, vol. 83, pp. 21-40, 1989.
- [4] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications*. New York, NY: Cambridge University Press, 1994.
- [5] E. D. Mastrovito. VLSI architectures for multiplication over finite field $\text{GF}(2^m)$. In T. Mora, editor, *Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, 6th International Conference, AAEC-6*, pp. 297-309, Rome, Italy, July 1988. New York, NY: Springer-Verlag.
- [6] A. J. Menezes, editor. *Applications of Finite Fields*. Boston, MA: Kluwer Academic Publishers, 1993.
- [7] J. Omura and J. Massey. "Computational method and apparatus for finite field arithmetic," U.S. Patent Number 4,587,627, May 1986.
- [8] C. Paar. *Efficient VLSI Architectures for Bit Parallel Computation in Galois Fields*. PhD thesis, Universität GH Essen, VDI Verlag, 1994.
- [9] C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed. "VLSI architecture for computing multiplications and inverses in $\text{GF}(2^m)$," *IEEE Transactions on Computers*, vol. 34, no. 8, pp. 709-717, August 1985.