Low-complexity bit-parallel dual basis multipliers using the modified Booth’s algorithm

Chiou-Yng Lee a,1, Che Wun Chiou b,*, Jim-Min Lin c,2

a Department of Computer Information and Network Engineering, Lunghwa University of Science and Technology, Taoyuan County 333, Taiwan, ROC
b Department of Computer Science and Information Engineering, Ching Yun University, 229 Chien-Hsin Road, Chung-Li 320, Taiwan, ROC
c Department of Information Engineering and Computer Science, Feng Chia University, Taichung City 407, Taiwan, ROC

Received 25 August 2004; received in revised form 1 July 2005; accepted 7 September 2005
Available online 2 December 2005

Abstract

New bit-parallel dual basis multipliers using the modified Booth’s algorithm are presented. Due to the advantage of the modified Booth’s algorithm, two bits are processed in parallel for reduction of both space and time complexities. A multiplexer-based structure has been proposed for realization of the proposed multiplication algorithm. We have shown that our multiplier saves about 9% space complexity as compared to other existing multipliers if the generating polynomial is trinomial or all one polynomial. Furthermore, the proposed multiplier is faster than existing multipliers.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Dual basis; Finite field multiplication; Booth’s algorithm; Trinomials; Polynomial basis; All one polynomial; Equally spaced polynomial

* Corresponding author. Tel.: +886 3 4581196x3700; fax: +886 3 4588924.
E-mail addresses: PP010@mail.lhu.edu.tw (C.-Y. Lee), cwchiou@cyu.edu.tw (C.W. Chiou), jimmy@fcu.edu.tw (J.-M. Lin).
1 Tel.: +886 3 4245215; fax: +886 3 4244168.
2 Tel.: +886 4 24517250x3754; fax: +886 4 24516101.

0045-7906/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
1. Introduction

In recent years, finite (Galois) fields GF($2^m$) have found many applications in areas of error correcting code [1], cryptography [2], digital signal processing [3,4], switching theory [5], pseudorandom number generation [6], encoding of Reed–Solomon codes [7], and solving the Wiener–Hopf equation [8]. Multiplication is the most critical operation in finite field arithmetic operations. Other time-consuming finite field arithmetic operations such as exponentiation, division, and multiplicative inversion can be carried out by repeated multiplications. Hence, there is a need to have fast multiplication architecture with low complexity. The representation of the field elements plays a crucial role in the efficiency of the architectures for the arithmetic operations. There are three popular bases, polynomial (canonical or standard) basis (PB), normal basis (NB), and dual basis (DB) for representation of finite fields. Different representation has its distinct advantage. The polynomial basis architectures [9–19] have low design complexities and their sizes are easily extended to meet various applications due to their simplicity, regularity, and modularity in architecture. The major advantage of the normal basis [20–27] is that the squaring of an element is computed by a cyclic shift of the binary representation. As compared with other two former bases, the dual basis multipliers [7,8,28–34] require less chip area. Berlekamp [7] firstly employed the dual basis of the polynomial basis for implementation of bit-serial field multiplication. Berlekamp’s bit-serial field multiplier efficiently realizes the multiplication operation by using a linear feedback shift register in the implementation of the Reed–Solomon encoder. Several serial systolic multipliers based on Berlekamp’s multiplication scheme have been proposed [33,34]. The complexity of the dual basis multipliers is heavily relied on the choice of the primitive irreducible polynomial which generates the field. Therefore, Wang and Blake [33] firstly derived a bit-serial dual basis multiplier with an irreducible trinomial to give low space and time complexities. Furthermore, Morii et al. [8] have found that the weakly dual basis is also available for bit-serial multiplication. A self-dual normal basis multiplier was presented by Wang [31] with very low complexity. Fenn et al. [30] developed a bit-parallel version of such dual basis multipliers using general dual basis. Wu et al. [28] further reduced the space complexity of the multiplier in [30] by reusing some partial sums.

The key idea of the “divide-and-conquer” algorithm was firstly introduced by Booth [35]. In this algorithm, intermediate results are always in a redundant form of two numbers. The modified Booth’s algorithm in [36] further reduces the partial products to the half. At each iteration cycle, two bits of the multiplier are processed in parallel. Pekmestzi [37] proposed a multiplexer-based array multiplier by using the modified Booth’s algorithm. Applying the modified Booth’s algorithm, we will present a dual basis finite field multiplier which is mainly constructed by multiplexers rather than conventional AND and XOR gates. The proposed multiplier saves both space and time complexities while comparing with other existing dual basis multipliers.

The organization of this paper is as follows. In Section 2, we introduce the mathematic background about the finite field arithmetic. In Section 3, the proposed multiplication algorithm using the modified Booth’s algorithm is derived. Examples are described in Section 4. Both space and time complexities of various bit-parallel multipliers over GF($2^m$) are discussed in Section 5. Finally, a brief conclusion is made in Section 6.
2. Preliminaries

Let \( \text{GF}(2^m) \) be a finite field of \( 2^m \) elements. \( \text{GF}(2^m) \) is an extension field of the ground field \( \text{GF}(2) \). Let \( x \) be a root of an irreducible primitive polynomial \( P(x) = p_0 + p_1x^1 + p_2x^2 + \cdots + p_{m-1}x^{m-1} + x^m \) of degree \( m \) over \( \text{GF}(2) \). Let \( \phi = \{1, x, x^2, \ldots, x^{m-1}\} \) be a polynomial basis of \( \text{GF}(2^m) \). An element \( B \in \text{GF}(2^m) \) is represented by

\[
B = b_0 + b_1x + b_2x^2 + \cdots + b_{m-1}x^{m-1},
\]

where \( b_i \in \{0, 1\} \) for all \( 0 \leq i \leq m - 1 \).

For \( \gamma \in \text{GF}(2^m) \), the trace \( \text{Tr}(\gamma) \) of \( \gamma \) over \( \text{GF}(2) \) is given by

\[
\text{Tr}(\gamma) = \sum_{i=0}^{m-1} \gamma^{2^i}.
\]

The following properties for trace function are hold:

(a) \( \text{Tr}(\gamma + \lambda) = \text{Tr}(\gamma) + \text{Tr}(\lambda) \) for all \( \gamma, \lambda \in \text{GF}(2^m) \),

(b) \( \text{Tr}(e^\gamma) = e\text{Tr}(\gamma) \) for \( e \in \text{GF}(2) \) and \( \gamma \in \text{GF}(2^m) \).

The trace function is a linear function from \( \text{GF}(2^m) \) to \( \text{GF}(2) \). Let \( \phi = \{1, x, x^2, \ldots, x^{m-1}\} \) and \( \psi = \{\psi_0, \psi_1, \psi_2, \psi_3, \ldots, \psi_{m-1}\} \) be bases for \( \text{GF}(2^m) \), and \( \gamma \in \text{GF}(2^m), \gamma \neq 0 \). Then, both \( \phi \) and \( \psi \) bases are said be weakly dual to each other with respect to \( \gamma \) if

\[
\text{Tr}(\gamma x^i \psi_j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}
\]

Fenn et al. [30] have showed that any polynomial basis has at least one dual basis with respect to the trace function. For example, one dual basis of the polynomial basis \( \{1, x, x^2, x^3, \ldots\} \) with the irreducible polynomial \( P(x) = 1 + x^2 + x^5 \) is \( \{x, 1, x^4, x^3, x^2\} \).

For an element \( A \in \text{GF}(2^m) \), \( A \) is represented in both \( \phi \) and \( \psi \) bases by

\[
A = \sum_{i=0}^{m-1} a_i x^i = \sum_{j=0}^{m-1} a_j^\psi \psi_j,
\]

where \( a_i \) and \( a_j^\psi \) are the coordinates of \( A \) with respect to the bases \( \phi \) and \( \psi \), respectively.

It may be necessary to perform a basis conversion between the polynomial basis and its dual basis. As the above representations for an element \( A \in \text{GF}(2^m) \), the \( j \)th coordinate with respect to the basis \( \psi \) is given by

\[
a_j^\psi = \text{Tr}(\gamma x^j A) = \text{Tr} \left( \gamma x^j \sum_{i=0}^{m-1} a_i x^i \right) = \sum_{i=0}^{m-1} a_i \text{Tr}(\gamma x^{i+j}).
\]
3. The proposed multiplexed-based multiplier

Assume that $m$ is even in this study and the result can be easily extended to an odd $m$. Let two elements $A$ and $B$ in GF(2$^m$), and $B, A$ are represented in the polynomial basis $\phi$ and its dual basis $\psi$, respectively, as follows:

$$A = \sum_{j=0}^{m-1} a_j^* \psi_j,$$

$$B = \sum_{i=0}^{m-1} b_i \chi^i,$$

where $a_j^*$ and $b_i \in$ GF(2) for $0 \leq i, j \leq m - 1$.

The product $C$ of $A$ and $B$ is computed as follows:

$$A \times B = C,$$

where $C$ is represented in the dual basis $\psi$ by

$$C = \sum_{j=0}^{m-1} c_j^* \psi_j,$$

for all $0 \leq j \leq m - 1$, $c_j^* \in$ GF(2$^m$).

The coefficient $c_j^*$ of $C$ is computed by

$$c_j^* = \text{Tr}(\gamma \chi^j C) = \text{Tr}(\gamma \chi^j AB) = \text{Tr} \left( \gamma \chi^j A \sum_{i=0}^{m-1} b_i \chi^i \right) = \sum_{i=0}^{m-1} (b_i \text{Tr}(\gamma \chi^{i+j} A)).$$

(2)

Based on Eq. (2), the product $C$ can be computed in the following form

$$\begin{bmatrix}
\text{Tr}(\gamma \chi^0 A) & \text{Tr}(\gamma \chi^1 A) & \text{Tr}(\gamma \chi^2 A) & \ldots & \text{Tr}(\gamma \chi^{m-1} A) \\
\text{Tr}(\gamma \chi^1 A) & \text{Tr}(\gamma \chi^2 A) & \text{Tr}(\gamma \chi^3 A) & \ldots & \text{Tr}(\gamma \chi^{m} A) \\
\text{Tr}(\gamma \chi^2 A) & \text{Tr}(\gamma \chi^3 A) & \text{Tr}(\gamma \chi^4 A) & \ldots & \text{Tr}(\gamma \chi^{m+1} A) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\text{Tr}(\gamma \chi^{m-1} A) & \text{Tr}(\gamma \chi^{m} A) & \text{Tr}(\gamma \chi^{m+1} A) & \ldots & \text{Tr}(\gamma \chi^{2m-2} A)
\end{bmatrix}
\begin{bmatrix}
b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-1}
\end{bmatrix}
= 
\begin{bmatrix}
c_0^* \\ c_1^* \\ c_2^* \\ \vdots \\ c_{m-1}^*
\end{bmatrix}.$$  

(3)

Let $a_{m+d}^* = \text{Tr}(\gamma \chi^{m+d} A)$ for $0 \leq d \leq m - 2$, Eq. (3) based on Eq. (1) can be rewritten by

$$\begin{bmatrix}
a_0^* & a_1^* & a_2^* & a_3^* & \ldots & a_{m-2}^* & a_{m-1}^* \\
a_1^* & a_2^* & a_3^* & a_4 & \ldots & a_{m-1}^* & a_m^* \\
a_2^* & a_3^* & a_4^* & a_5^* & \ldots & a_m^* & a_{m+1}^* \\
a_3^* & a_4^* & a_5^* & a_6^* & \ldots & a_{m+1}^* & a_{m+2}^* \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
a_{m-2}^* & a_{m-1}^* & a_m^* & a_{m+1}^* & \ldots & a_{2m-4}^* & a_{2m-3}^* \\
a_{m-1}^* & a_m^* & a_{m+1}^* & a_{m+2}^* & \ldots & a_{2m-3}^* & a_{2m-2}^*
\end{bmatrix}
\begin{bmatrix}
b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{m-1}
\end{bmatrix}
= 
\begin{bmatrix}
c_0^* \\ c_1^* \\ c_2^* \\ \vdots \\ c_{m-2}^* \\ c_{m-1}^*
\end{bmatrix}.$$  

(4)
Assume that the function \( f_{i,j,u,v} \) is defined as follows:
\[
f_{i,j,u,v} = a_i^* b_u \oplus a_j^* b_v.
\]
The function \( f_{i,j,u,v} \) can also be represented by
\[
f_{i,j,u,v} = \overline{a_i} \cdot \overline{a_j} \cdot 0 \lor \overline{a_i} \cdot a_j^* \cdot b_v \lor a_i^* \cdot \overline{a_j} \cdot b_u \lor a_i^* \cdot a_j^* \cdot (b_u \oplus b_v).
\]
Assuming that the results of \( f_{i,j,u,v} \) can be determined depending on the values of the logical variables \( a_i^* \) and \( a_j^* \). The case of \( f_{i,j,u,v} \) requires computation when \( a_i^* \) and \( a_j^* \) have a value 1, i.e., \( f_{i,j,u,v} = b_u \oplus b_v \). Hence, the computed result of \( f_{i,j,u,v} \) can be selected from the set \( \{0, b_u, b_v, b_u \oplus b_v\} \) by the values of \( a_i^* \) and \( a_j^* \). Notably, an 4-to-1 MUX is used to realize the function \( f_{i,j,u,v} \) and is shown in Fig. 1. Using \( f_{i,j,u,v} \)'s, the above equation is rewritten as
\[
\begin{bmatrix}
    f_{0,1,0,1} & f_{2,3,2,3} & \cdots & f_{m-2,m-1,2,m-1} \\
    f_{1,2,0,1} & f_{3,4,2,3} & \cdots & f_{m-1,m,m-2,m-1} \\
    f_{2,3,0,1} & f_{4,5,2,3} & \cdots & f_{m,m+1,m-2,m-1} \\
    f_{3,4,0,1} & f_{5,6,2,3} & \cdots & f_{m+1,m+2,m-2,m-1} \\
    \vdots & \vdots & \ddots & \vdots \\
    f_{m-2,m-1,0,1} & f_{m,m+1,2,3} & \cdots & f_{2m-4,2m-3,m-2,m-1} \\
    f_{m-1,m,0,1} & f_{m+1,m+2,2,3} & \cdots & f_{2m-3,2m-2,m-2,m-1}
\end{bmatrix}
\begin{bmatrix}
    1 \\
    1 \\
    1 \\
    1 \\
    \vdots \\
    1 \\
\end{bmatrix}
= \begin{bmatrix}
    c_0^* \\
    c_1^* \\
    c_2^* \\
    c_3^* \\
    \vdots \\
    c_{m-2}^* \\
    c_{m-1}^*
\end{bmatrix}.
\]
In other words, the coefficient of \( C \) is given by
\[
c_i^* = \sum_{j=0}^{[\frac{m-1}{2}]} f_{2j+i,2j+i+1,2j+1+1}, \quad \text{for all } 0 \leq i \leq m-1.\tag{6}
\]
As stated above, Fig. 2 shows the function block for the proposed multiplication algorithm. The function block “TRACE” performs the following trace functions to do pre-computing basis conversion:
\[
a_{m+d}^* = \text{Tr}(\gamma^d A), \quad \text{for } 0 \leq d \leq m-2.
\]
The “Pre-XOR” function responses for the XOR functions as follows:
\[
b_i \oplus b_{i+1}, \quad \text{for } i = 0, 2, 4, \ldots, m-2.
\]
\[\begin{array}{c}
  b_u \\
  b_v \\
  b_u \\
  b_v \\
  0
\end{array}\]
\[\begin{array}{c}
  a_i^* \\
  a_j^* \\
  \text{4-to-1 MUX} \\
  f_{i,j,u,v}
\end{array}\]
Fig. 1. An 4-to-1 MUX for realizing \( f_{i,j,u,v} \).
The function block “MUX ARRAY” computes the function at the left side of Eq. (5) and its detail circuit is shown in Fig. 3. The function block “SUM” consists of binary XOR trees and performs final result \( C \) based on Eq. (6).

If \( m \) is an odd number, one zero-column is added to make the matrix \( A \) in Eq. (4) with an even number of columns and one zero-row is inserted to the matrix \( B \). Thus, Eq. (4) for an odd \( m \) is rewritten as

\[
\begin{bmatrix}
    a_0^* & a_1^* & a_2^* & a_3^* & \ldots & a_{m-2}^* & a_{m-1}^* & 0 \\
    a_1^* & a_2^* & a_3^* & a_4^* & \ldots & a_{m-1}^* & a_m^* & 0 \\
    a_2^* & a_3^* & a_4^* & a_5^* & \ldots & a_m^* & a_{m+1}^* & 0 \\
    a_3^* & a_4^* & a_5^* & a_6^* & \ldots & a_{m+1}^* & a_{m+2}^* & 0 \\
    \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\
    a_{m-2}^* & a_{m-1}^* & a_m^* & a_{m+1}^* & \ldots & a_{2m-4}^* & a_{2m-3}^* & 0 \\
    a_{m-1}^* & a_m^* & a_{m+1}^* & a_{m+2}^* & \ldots & a_{2m-3}^* & a_{2m-2}^* & 0
\end{bmatrix}
\begin{bmatrix}
    b_0 \\
    b_1 \\
    b_2 \\
    b_3 \\
    \ldots \\
    b_{m-2} \\
    b_{m-1} \\
    0
\end{bmatrix}
= \begin{bmatrix}
    c_0^* \\
    c_1^* \\
    c_2^* \\
    c_3^* \\
    \ldots \\
    c_{m-2}^* \\
    c_{m-1}^*
\end{bmatrix}.
\]

Based on Eq. (7), Eq. (5) becomes

\[
\begin{bmatrix}
    f_{0,1,0,1} & f_{2,3,2,3} & \ldots & f_{m-3,m-2,m-3,m-2} & f_{m-1,-1,m-1,-1} \\
    f_{1,2,0,1} & f_{3,4,2,3} & \ldots & f_{m-2,m-1,m-3,m-2} & f_{m,-1,-1,m-1} \\
    f_{2,3,0,1} & f_{4,5,2,3} & \ldots & f_{m-1,m,m-3,m-2} & f_{m+1,-1,-1,m-1} \\
    f_{3,4,0,1} & f_{5,6,2,3} & \ldots & f_{m,m+1,m-3,m-2} & f_{m+2,-1,-1,m-1} \\
    \vdots & \vdots & \vdots & \vdots & \vdots \\
    f_{m-2,m-1,0,1} & f_{m,m+1,2,3} & \ldots & f_{2m-5,2m-4,3,m-2} & f_{2m-3,-1,-1,m-1} \\
    f_{m-1,m,0,1} & f_{m+1,m+2,2,3} & \ldots & f_{2m-4,2m-3,3,m-2} & f_{2m-2,-1,-1,m-1}
\end{bmatrix}
\begin{bmatrix}
    1 \\
    1 \\
    1 \\
    1 \\
    \vdots \\
    1 \\
    1
\end{bmatrix}
= \begin{bmatrix}
    c_0^* \\
    c_1^* \\
    c_2^* \\
    c_3^* \\
    \ldots \\
    c_{m-2}^* \\
    c_{m-1}^*
\end{bmatrix}.
\]
The symbol ‘‘–1’’ in the subscripts of \( f_{i,j,u,v} \)’s indicates a logical value ‘‘0’’ applied to the corresponding input.

4. Examples

Two examples are used to describe our algorithm and will be depicted in the following paragraphs.

4.1. Trinomial

An irreducible polynomial \( P(x) \) of degree \( m \) is termed as a trinomial polynomial if it can be formed as \( 1 + x^n + x^m \) with \( m > n > 0 \). A trinomial polynomial \( P(x) = 1 + x + x^4 \) is as an example to illustrate our algorithm.

Let \( \alpha \) be a root of \( P(x) \). The basis \( \phi = \{1, \alpha, \alpha^2, \alpha^3\} \) is a polynomial basis and is assumed in this example. Let the basis \( \psi = \{\psi_0, \psi_1, \psi_2, \psi_3\} \) be its dual basis. Suppose \( A \) and \( B \) are represented in the basis \( \psi \) and \( \phi \), respectively, and are expressed by

\[
A = a^*_0 \psi_0 + a^*_1 \psi_1 + a^*_2 \psi_2 + a^*_3 \psi_3,
\]


Fig. 3. The detail circuit of the function block ‘‘MUX ARRAY’’ in Fig. 2.
and
\[ B = b_0 + b_1x + b_2x^2 + b_3x^3. \]

The product \( C = A \times B \) is computed by

\[
\begin{bmatrix}
  a_0^* & a_1^* & a_2^* & a_3^* \\
  a_1^* & a_2^* & a_3^* & \text{Tr}(\gamma x^4 A) \\
  a_2^* & a_3^* & \text{Tr}(\gamma x^4 A) & \text{Tr}(\gamma x^5 A) \\
  a_3^* & \text{Tr}(\gamma x^4 A) & \text{Tr}(\gamma x^5 A) & \text{Tr}(\gamma x^6 A)
\end{bmatrix}
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}
= \begin{bmatrix}
  c_0^* \\
  c_1^* \\
  c_2^* \\
  c_3^*
\end{bmatrix}. \tag{9}
\]

Trace values in the above equation are given as follows:

\[ a_4^* = \text{Tr}(\gamma x^4 A) = \text{Tr}(\gamma (x + 1)A) = \text{Tr}(\gamma xA) + \text{Tr}(\gamma A) = a_1^* + a_0^*, \]
\[ a_5^* = \text{Tr}(\gamma x^5 A) = \text{Tr}(\gamma x^4 A) = \text{Tr}(\gamma (x + 1)A) = \text{Tr}(\gamma x^2 A) + \text{Tr}(\gamma xA) = a_2^* + a_1^*, \]
\[ a_6^* = \text{Tr}(\gamma x^6 A) = \text{Tr}(\gamma x^4 A) = \text{Tr}(\gamma x^2 A) + \text{Tr}(\gamma x^3 A + \gamma x^2 A) = a_3^* + a_2^*. \]

Thus, Eq. (9) is rewritten as

\[
\begin{bmatrix}
  a_0^* & a_1^* & a_2^* & a_3^* \\
  a_1^* & a_2^* & a_3^* & a_1^* + a_0^* \\
  a_2^* & a_3^* & a_1^* + a_0^* & a_2^* + a_1^* \\
  a_3^* & a_1^* + a_0^* & a_2^* + a_1^* & a_3^* + a_2^*
\end{bmatrix}
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}
= \begin{bmatrix}
  c_0^* \\
  c_1^* \\
  c_2^* \\
  c_3^*
\end{bmatrix}. \tag{10}
\]

Replacing with functions \( f_{i,j,a,v} \)'s, Eq. (9) or Eq. (10) is written as

\[
\begin{bmatrix}
  f_{0,1,0,1} & f_{2,3,2,3} \\
  f_{1,2,0,1} & f_{3,4,2,3} \\
  f_{2,3,0,1} & f_{4,5,2,3} \\
  f_{3,4,0,1} & f_{5,6,2,3}
\end{bmatrix}
\begin{bmatrix}
  1 \\
  1 \\
  1 \\
  1
\end{bmatrix}
= \begin{bmatrix}
  c_0^* \\
  c_1^* \\
  c_2^* \\
  c_3^*
\end{bmatrix}. \tag{11}
\]

The multiplexer-based structure for implementation of Eq. (11) is shown in Fig. 4.

4.2. All one polynomial

An irreducible polynomial \( P(x) = p_0 + p_1x + \cdots + p_{m-2}x^{m-2} + p_{m-1}x^{m-1} + x^m \) is called all one polynomial (AOP) of degree \( m \) if \( p_i = 1 \) for \( i = 0, 1, 2, \ldots, m - 1 \) [14]. We use the AOP \( P(x) = 1 + x + x^2 + x^3 + x^4 \) as an example to describe the proposed algorithm. Let \( x \) be the root of \( P(x) \). The polynomial basis is \( \phi = \{1, x, x^2, x^3\} \). Let the basis \( \psi = \{\psi_0, \psi_1, \psi_2, \psi_3\} \) be its dual basis. If \( A \) and \( B \) are elements in \( \text{GF}(2^4) \) generated by \( P(x) = 1 + x + x^2 + x^3 + x^4 \) and are represented by

\[ A = a_0^*\psi_0 + a_1^*\psi_1 + a_2^*\psi_2 + a_3^*\psi_3, \]

and

\[ B = b_0 + b_1x + b_2x^2 + b_3x^3. \]
The product $C$ of $A$ and $B$ is given by

$$
\begin{bmatrix}
  a_0^* & a_1^* & a_2^* & a_3^* \\
  a_1^* & a_2^* & a_3^* & a_4^* \\
  a_2^* & a_3^* & a_4^* & a_5^* \\
  a_3^* & a_4^* & a_5^* & a_6^*
\end{bmatrix}
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}
= 
\begin{bmatrix}
  c_0^* \\
  c_1^* \\
  c_2^* \\
  c_3^*
\end{bmatrix}.
$$

Based on the property of $x^5 = 1$, the trace values in the above equation are computed by

$$
a_4^* = \text{Tr}(\gamma x^4 A) = \text{Tr}(\gamma(1 + x + x^2 + x^3)A) = \text{Tr}(\gamma A) + \text{Tr}(\gamma x A) + \text{Tr}(\gamma x^2 A) + \text{Tr}(\gamma x^3 A)
= a_0^* + a_1^* + a_2^* + a_3^*,
$$

$$
a_5^* = \text{Tr}(\gamma x^5 A) = \text{Tr}(\gamma A) = a_0^*,
$$

$$
a_6^* = \text{Tr}(\gamma x^6 A) = \text{Tr}(\gamma xA) = a_1^*.
$$

Thus, we have the following equation:

$$
\begin{bmatrix}
  a_0^* & a_1^* & a_2^* & a_3^* \\
  a_1^* & a_2^* & a_3^* & a_4^* \\
  a_2^* & a_3^* & a_4^* & a_5^* \\
  a_3^* & a_4^* & a_5^* & a_6^*
\end{bmatrix}
\begin{bmatrix}
  b_0 \\
  b_1 \\
  b_2 \\
  b_3
\end{bmatrix}
= 
\begin{bmatrix}
  c_0^* \\
  c_1^* \\
  c_2^* \\
  c_3^*
\end{bmatrix}.
$$
Eq. (13) can be decomposed into
\[
\begin{bmatrix}
{a_0^*} & {a_1^*} & {a_2^*} & {a_3^*} \\
{a_1^*} & {a_2^*} & {a_3^*} & 0 \\
{a_2^*} & {a_3^*} & 0 & {a_0^*} \\
0 & {a_0^*} & {a_1^*} & {a_2^*}
\end{bmatrix}
\begin{bmatrix}
{b_0} \\
{b_1} \\
{b_2} \\
{b_3}
\end{bmatrix}
+
\begin{bmatrix}
0 \\
{a_3^*b_3} \\
{a_2^*b_2} \\
{a_1^*b_1}
\end{bmatrix}
=
\begin{bmatrix}
{c_0^*} \\
{c_1^*} \\
{c_2^*} \\
{c_3^*}
\end{bmatrix}.
\tag{14}
\]

Using \( f_{i,j,u,v} \) representation, the above equation is expressed by
\[
\begin{bmatrix}
{f_{0,1,0,1}} & {f_{2,3,2,3}} \\
{f_{1,2,0,1}} & {f_{3,-1,2,3}} \\
{f_{2,3,0,1}} & {f_{-1,0,2,3}} \\
{f_{3,-1,0,1}} & {f_{0,1,2,3}}
\end{bmatrix}
\begin{bmatrix}
1 \\
1 \\
1 \\
1
\end{bmatrix}
+
\begin{bmatrix}
0 \\
{a_4^*b_3} \\
{a_4^*b_2} \\
{a_4^*b_1}
\end{bmatrix}
=
\begin{bmatrix}
{c_0^*} \\
{c_1^*} \\
{c_2^*} \\
{c_3^*}
\end{bmatrix}.
\tag{15}
\]

The multiplexer-based implementation of Eq. (15) is shown in Fig. 5.

5. Complexity analysis

Firstly, we use an example with the generating polynomial \( P(x) = 1 + x + x^m \) to describe how to derive the space and time complexities. Table 1 shows the space and time complexities of each function block in Fig. 2 for such a generating polynomial \( P(x) = 1 + x + x^m \).
In general, the space complexity of the proposed multiplication algorithm for the generating polynomial with \( k \) terms requires
\[
m \left\lfloor \frac{m}{2} \right. \right. + (k - 2)m + \left\lfloor \frac{m}{2} \right. \right. - k + 2 \quad 2 - \text{input XOR gates, and}
\[
m \left\lfloor \frac{m}{2} \right. \right. \quad \text{4-to-1 Multiplexers.}
\]

However, some multiplexers can be replaced by AND gates to reduce complexity for some special generating polynomials such as AOP. Comparisons of the space complexity for various bit-parallel multipliers over GF(\( 2^m \)) for trinomial and all one polynomial are listed in Tables 2A and 2B, respectively. We will take into the transistor count using a standard CMOS VLSI realization. In the CMOS VLSI technology, 2-input AND, 2-input XOR and 4-to-1 MUX are composed of 6, 6 and 16 transistors, respectively [37–40]. Tables 3A and 3B show the total number of transistors for various bit-parallel multipliers. As the trinomial of the form \( P(x) = 1 + x^n + x^m \) is employed for the generating polynomial, our multiplexer-based multiplication algorithm saves about 9% space complexity while comparing with other existing multipliers. When the generating polyn-

<table>
<thead>
<tr>
<th>Item</th>
<th>Operation</th>
<th>Space complexity</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TRACE</td>
<td>((m - 1)) XORs</td>
<td>1 XOR gate delay</td>
</tr>
<tr>
<td>2</td>
<td>Pre-XOR</td>
<td>(\left\lceil \frac{m}{2} \right\rceil) XORs</td>
<td>1 XOR gate delay</td>
</tr>
<tr>
<td>3</td>
<td>MUX ARRAY</td>
<td>(m \times \left\lceil \frac{m}{2} \right\rceil) 4-to-1 MUXs</td>
<td>(\left\lceil \frac{m}{2} \right\rceil) 4-to-1 MUX gate delays</td>
</tr>
<tr>
<td>4</td>
<td>SUM</td>
<td>(m \times \left\lceil \frac{m}{2} \right\rceil) XORs</td>
<td>(\left\lceil \log_2 \left\lceil \frac{m}{2} \right\rceil \right\rceil) XOR gate delays and (\left\lceil \frac{m}{2} \right\rceil) 4-to-1 MUX gate delays</td>
</tr>
</tbody>
</table>

In general, the space complexity of the proposed multiplication algorithm for the generating polynomial with \( k \) terms requires
\[
m \left\lfloor \frac{m}{2} \right. \right. + (k - 2)m + \left\lfloor \frac{m}{2} \right. \right. - k + 2 \quad 2 - \text{input XOR gates, and}
\[
m \left\lfloor \frac{m}{2} \right. \right. \quad \text{4-to-1 Multiplexers.}
\]

However, some multiplexers can be replaced by AND gates to reduce complexity for some special generating polynomials such as AOP. Comparisons of the space complexity for various bit-parallel multipliers over GF(\( 2^m \)) for trinomial and all one polynomial are listed in Tables 2A and 2B, respectively. We will take into the transistor count using a standard CMOS VLSI realization. In the CMOS VLSI technology, 2-input AND, 2-input XOR and 4-to-1 MUX are composed of 6, 6 and 16 transistors, respectively [37–40]. Tables 3A and 3B show the total number of transistors for various bit-parallel multipliers. As the trinomial of the form \( P(x) = 1 + x^n + x^m \) is employed for the generating polynomial, our multiplexer-based multiplication algorithm saves about 9% space complexity while comparing with other existing multipliers. When the generating polyn-

| Table 2A |
| Comparison of space-complexity of bit-parallel multipliers in GF(\( 2^m \)) using an irreducible trinomial |
| Multipliers | Basis | #AND | #XOR | #MUX |
| P(x) = 1 + x^n + x^m, \( m > 2 \) |
| Wu [19] | PB | \(m^2\) | \(m^2 - 1\) | 0 |
| Wu et al. [28] | DB | \(m^2\) | \(m^2 - 1\) | 0 |
| Presented here | DB | 0 | \(m \times \left\lceil \frac{m}{2} \right\rceil + 1 - 1\) | \(\left\lceil \frac{m}{2} \right\rceil \times m\) |
| P(x) = 1 + x^n + x^m, \( 1 < n < \frac{m}{2} \) |
| Wu [19] | PB | \(m^2\) | \(m^2 - 1\) | 0 |
| Wu et al. [28] | DB | \(m^2\) | \(m^2 - 1\) | 0 |
| Presented here | DB | 0 | \(m \times \left\lceil \frac{m}{2} \right\rceil + 1 - 1\) | \(\left\lceil \frac{m}{2} \right\rceil \times m\) |
| P(x) = 1 + x^n + x^m, \( n = \frac{m}{2} \) |
| Wu [19] | PB | \(m^2\) | \(m^2 - \frac{m}{2}\) | 0 |
| Wu et al. [28] | DB | \(m^2\) | \(m^2 - \frac{m}{2}\) | 0 |
| Presented here | DB | 0 | \(m \times \left\lceil \frac{m}{2} \right\rceil - 1\) | \(\left\lceil \frac{m}{2} \right\rceil \times m\) |
mial is all one polynomial, our proposed multiplier still saves about 9% space complexity while comparing with existing bit-parallel multipliers.

Tables 4A and 4B show comparisons of the time complexities of various bit-parallel multipliers over GF($2^m$). The symbols $T_A$, $T_X$ and $T_M$ denote the gate delays of 2-input AND, 2-input XOR and 4-to-1 MUX, respectively. Our proposed multiplexer-based multiplier is little faster than existing multipliers.

With different $m$ values, Tables 5A and 5B show the comparing results of space complexity of various bit-parallel multipliers with trinomials and AOPs, respectively. Tables 6A and 6B depict the comparing results of time complexity of various bit-parallel multipliers with trinomials and AOPs, respectively, if real circuits such as M74HC86 (STMicroelectronics, XOR gate, $t_{PD} = 12$ ns (TYP.)) [41], M74HC08 (STMicroelectronics, AND gate, $t_{PD} = 7$ ns (TYP.)) [42], and M74HC153 (STMicroelectronics, Multiplexer, $t_{PD} = 16$ ns (TYP.)) [43] are employed.

---

### Table 2B
Comparison of space-complexity of bit-parallel multipliers in GF($2^m$) using an irreducible AOP

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Basis</th>
<th>#AND</th>
<th>#XOR</th>
<th>#MUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itoh–Tsujii [14]</td>
<td>PB</td>
<td>$m^2 + 2m + 1$</td>
<td>$m^2 + 2m$</td>
<td>0</td>
</tr>
<tr>
<td>Wu–Hasan [29]</td>
<td>DB</td>
<td>$m^2$</td>
<td>$m^2 - 1$</td>
<td>0</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>$m - 1$</td>
<td>$(m + 3) \left\lceil \frac{m}{2} \right\rceil - 2$</td>
<td>$m \left\lfloor \frac{m}{2} \right\rfloor$</td>
</tr>
</tbody>
</table>

### Table 3A
Comparison of the total number of transistors of various bit-parallel multipliers in GF($2^m$) using an irreducible trinomial

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Basis</th>
<th>Generating polynomial</th>
<th>#Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(x) = 1 + x + x^m, \ m &gt; 2$</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>$22m \left\lceil \frac{m}{2} \right\rceil + 6m - 6$</td>
</tr>
<tr>
<td>$P(x) = 1 + x^n + x^m, \ 1 &lt; n &lt; \frac{m}{2}$</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>$22m \left\lceil \frac{m}{2} \right\rceil + 6m - 6$</td>
</tr>
<tr>
<td>$P(x) = 1 + x^n + x^m, \ n = \frac{m}{2}$</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>$12m^2 - 3m$</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>$22m \left\lceil \frac{m}{2} \right\rceil - 3m$</td>
</tr>
</tbody>
</table>

### Table 3B
Comparison of the total number of transistors of various bit-parallel multipliers in GF($2^m$) using an irreducible AOP

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Basis</th>
<th>Generating polynomial</th>
<th>#Transistors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itoh–Tsujii [14]</td>
<td>PB</td>
<td>AOP</td>
<td>$12m^2 + 24m + 6$</td>
</tr>
<tr>
<td>Wu–Hasan [29]</td>
<td>DB</td>
<td>AOP</td>
<td>$12m^2 - 6$</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>AOP</td>
<td>$22m \left\lceil \frac{m}{2} \right\rceil + 15m - 8$</td>
</tr>
</tbody>
</table>
Table 4A
Comparison of time complexity of various bit-parallel multipliers in \( \text{GF}(2^m) \) using an irreducible trinomial

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Basis</th>
<th>Generating polynomial</th>
<th>Propagation delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(x) = 1 + x + x^m ), ( m &gt; 2 )</td>
<td></td>
<td>Trinomial</td>
<td></td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2(m - 2) \rceil + 2)T_X )</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2 m \rceil + 1)T_X )</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_M + (\lceil \log_2(\frac{m}{2}) \rceil + 1)T_X )</td>
</tr>
<tr>
<td>( P(x) = 1 + x^n + x^m ), ( 1 &lt; n &lt; \frac{m}{2} )</td>
<td></td>
<td>Trinomial</td>
<td></td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2(m - 1) \rceil + 2)T_X )</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2(\frac{m+n-l}{2}) \rceil + 2)T_X )</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_M + (\lceil \log_2(\frac{m}{2}) \rceil + 1)T_X )</td>
</tr>
<tr>
<td>( P(x) = 1 + x^n + x^m ), ( n = \frac{m}{2} )</td>
<td></td>
<td>Trinomial</td>
<td></td>
</tr>
<tr>
<td>Wu [19]</td>
<td>PB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2(m - 1) \rceil + 1)T_X )</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_A + (\lceil \log_2(\frac{m+n+1}{2}) \rceil + 1)T_X )</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>Trinomial</td>
<td>( T_M + (\lceil \log_2(\frac{m}{2}) \rceil + 1)T_X )</td>
</tr>
</tbody>
</table>

Table 4B
Comparison of time complexity of various bit-parallel multipliers in \( \text{GF}(2^m) \) using an irreducible AOP

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>Basis</th>
<th>Generating polynomial</th>
<th>Propagation delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itoh–Tsujii [14]</td>
<td>PB</td>
<td>AOP</td>
<td>( T_A + \log_2 m + \log_2(m + 2)T_X )</td>
</tr>
<tr>
<td>Wu–Hasan [29]</td>
<td>DB</td>
<td>AOP</td>
<td>( T_A + (\lceil \log_2(m - 2) \rceil + 1)T_X )</td>
</tr>
<tr>
<td>Presented here</td>
<td>DB</td>
<td>AOP</td>
<td>( T_M + (\lceil \log_2(\frac{m}{2}) \rceil + 1)T_X )</td>
</tr>
</tbody>
</table>

Table 5A
Comparison of space complexity for various bit-parallel multipliers in \( \text{GF}(2^m) \) using an irreducible trinomial with different \( m \) values

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>( m = 21 )</th>
<th>( m = 41 )</th>
<th>( m = 81 )</th>
<th>( m = 161 )</th>
<th>( m = 321 )</th>
<th>( m = 641 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wu [19]</td>
<td>5286</td>
<td>20,166</td>
<td>78,726</td>
<td>311,046</td>
<td>1,236,486</td>
<td>4,930,566</td>
</tr>
<tr>
<td>Wu et al. [28]</td>
<td>5286</td>
<td>20,166</td>
<td>78,726</td>
<td>311,046</td>
<td>1,236,486</td>
<td>4,930,566</td>
</tr>
<tr>
<td>Presented here</td>
<td>5202</td>
<td>19,182</td>
<td>73,542</td>
<td>287,862</td>
<td>1,138,902</td>
<td>4,530,582</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.6%</td>
<td>5.1%</td>
<td>7.0%</td>
<td>8.1%</td>
<td>8.6%</td>
<td>8.8%</td>
</tr>
</tbody>
</table>

Note: Unit: transistor.

Table 5B
Comparison of space complexity for various bit-parallel multipliers in \( \text{GF}(2^m) \) using an irreducible AOP with different \( m \) values

<table>
<thead>
<tr>
<th>Multipliers</th>
<th>( m = 18 )</th>
<th>( m = 36 )</th>
<th>( m = 82 )</th>
<th>( m = 178 )</th>
<th>( m = 348 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itoh–Tsujii [14]</td>
<td>4326</td>
<td>16,422</td>
<td>82,662</td>
<td>384,486</td>
<td>1,461,606</td>
</tr>
<tr>
<td>Wu–Hasan [29]</td>
<td>3882</td>
<td>15,546</td>
<td>80,682</td>
<td>380,202</td>
<td>1,453,242</td>
</tr>
<tr>
<td>Presented here</td>
<td>3826</td>
<td>14,788</td>
<td>75,186</td>
<td>351,186</td>
<td>1,337,356</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.5%</td>
<td>5.1%</td>
<td>7.3%</td>
<td>8.3%</td>
<td>8.6%</td>
</tr>
</tbody>
</table>

Note: Unit: transistor.
6. Conclusions

We have presented new bit-parallel dual basis multipliers using the modified Booth’s algorithm. The proposed multiplier inherits the advantage of the modified Booth’s algorithm and then reduces both space and time complexities. A multiplexer-based structure has been proposed for realization of the proposed multiplication algorithm. We have shown that our multiplier saves about 9% space complexity as compared to other existing multipliers if the generating polynomial is trinomial or all one polynomial. Furthermore, the proposed multiplier is little faster than other existing multipliers.

Acknowledgments

The authors would like to thank anonymous referees and the editor for carefully reading the paper and for their great help in improving the paper.

References


Chiou-Yung Lee received the Bachelor's degree (1986) in Medical Engineering and the M.S. degree in Electronic Engineering (1992), both from the Chung Yuan University, Taiwan, and the Ph.D. degree in Electrical Engineering from Chang Gung University, Taiwan, in 2001. From 1988 to now, he was a research associate with Chunghwa Telecommunication Laboratory in Taiwan. He joined the department of project planning. He taught those related field courses at Ching-Yun Technology University. He is currently as an assistant professor of Department of Computer Information and Network Engineering in Lunghwa University of Science and Technology. His research interests include computations in finite fields, error-control coding, signal processing, and digital transmission system. Besides, he is a member of the IEEE and the IEEE Computer society. He is also an honor member of Phi Tao Phi in 2001.

Che Wun Chiou received his B.S. degree in Electronic Engineering from Chung Yuan Christian University in 1982, the M.S. degree and the Ph.D. degree in Electrical Engineering from National Cheng Kung University in 1984 and 1989, respectively. From 1990 to 2000, he was with the Chung Shan Institute of Science and Technology in Taiwan. He joined the Department of Electronic Engineering and the Department of Computer Science and Information Engineering, Ching Yun University in 2000 and 2005, respectively. He is currently as Dean of Division of Continuing Education in Ching Yun University. His current research interests include fault-tolerant computing, computer arithmetic, parallel processing, and cryptography.

Jim-Min Lin was born on March 5, 1963 in Taipei, Taiwan. He received the B.S. degree in Engineering Science and the M.S. and the Ph.D. degrees in Electrical Engineering, all from National Cheng Kung University, Tainan, Taiwan, in 1985, 1987, and 1992, respectively. Since February 1993, he has been an Associate Professor at the Department of Information Engineering and Computer Science, Feng Chia University, Taichung City, Taiwan. His research interests include Operating Systems, Software Integration/Reuse, Embedded Systems, Software Agent Technology, and Testable Design.