Available online at www.sciencedirect.com

ScienceDirect

Information Processing Letters 107 (2008) 211–215

Information Processing Letters

www.elsevier.com/locate/ipl

Low complexity bit parallel multiplier for GF(2^m) generated by equally-spaced trinomials

Haibin Shen [a,*], Yier Jin [a,b]

[a] *Institute of VLSI Design, Zhejiang University, Hangzhou, China*
[b] *Department of Electrical Engineering, Yale University, 10 Hillhouse Avenue, New Haven, CT 06520, USA*

Received 20 February 2007; received in revised form 14 May 2007; accepted 3 January 2008

Available online 16 March 2008

Communicated by D. Pointcheval

Abstract

Based on the shifted polynomial basis (SPB), a high efficient bit-parallel multiplier for the field GF(2^m) defined by an equally-spaced trinomial (EST) is proposed. The use of SPB significantly reduces time delay of the proposed multiplier and at the same time Karatsuba method is combined with SPB to decrease space complexity. As a result, with the same time complexity, approximately 3/4 gates of previous multipliers are used in the proposed multiplier.
© 2008 Published by Elsevier B.V.

Keywords: Shifted polynomial basis (SPB); Equally-spaced trinomial (EST); Karatsuba method; Bit-parallel multiplier; Cryptography

1. Introduction

Finite filed operations are used in many areas such as coding theory, computers algebra, combinatorial designs and cryptography [3,6,13]. Among these operations, multiplication is of the most importance because other complex operations such as exponentiation, division, etc. can be carried out through iterative multiplications. Based on the advanced design technology nowadays, more and more logic gates can be located on a single chip which makes the implementations of parallel architectures possible and reasonable. In order to improve the efficiency of cryptographic system and coding system, many bit-parallel multiplier architectures have been proposed recently to achieve high computation speed [2,5,8,10–12,14,15]. Also, various basis except for polynomial basis (PB), dual basis (DB), and normal basis (NB) are developed in order to further reduce the critical path of multipliers with even fewer logic gates. And certain types of irreducible polynomials are used to improve the performance of multipliers in which trinomial is one of the best choices [12].

Although different architectures can be evaluated from several points of view, time complexity and space complexity are often the two most important parameters. The former is defined as the elapsed time between input and output of the circuit implementing the multiplier, and it is usually expressed as the sum of T_A (the delay of a two input AND gate) and T_X (the delay of a two input XOR gate) with corresponding coefficients. The latter is weighed by the numbers of AND gates and XOR gates used in multiplier denoted as #AND

[*] Corresponding author.
E-mail addresses: shb@vlsi.zju.edu.cn (H. Shen), yier.jin@yale.edu (Y. Jin).

0020-0190/$ – see front matter © 2008 Published by Elsevier B.V.
doi:10.1016/j.ipl.2008.01.012

and #XOR. In the finite field generated by trinomials, the most efficient multiplier architecture nowadays contains m^2 AND gates and m^2-1 XOR gates with time delay of $T_A + (1 + \lceil \log_2 m \rceil)T_X$ [5]. If the trinomial is an equally-spaced trinomial (EST) in the form of $f(x) = x^m + x^{\frac{m}{2}} + 1$ (m is even), the best result is TimeDelay $= T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$, #AND $= m^2$, #XOR $= m^2 - \frac{m}{2}$.

As mentioned above, although we can put much more logic gates in a single chip than ever before, the $O(m^2)$ complexity of AND and XOR gates still costs considerable chip area. Many researchers are devoted to the reduction of multipliers' space complexity without increasing their time complexity. M. Elia et al. [1] use Karatsuba–Ofman multiplication and achieve even lower space complexity but their method requires two more T_X delays.

In this paper, we propose a new bit-parallel multiplier for GF(2^m) defined by EST $f(x) = x^m + x^{\frac{m}{2}} + 1$ using shifted polynomial basis which can significantly reduce the critical path. Also, we use the well-known Karatsuba–Ofman multiplication [4] to decrease the space complexity of the proposed bit-parallel multiplier. Based on these two methods, an high efficient architecture is constructed. The space complexity of the proposed multiplier is about 3/4 of the previous result while the time complexity matches the best efficient multipliers ever known of $T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$. The irreducible EST in the form of $x^m + x^{\frac{m}{2}} + 1$ exist when $m = 2 \times 3^i$ where i is a non-negative integer. Although the number of irreducible EST is not that redundant, they can be used in source critical area, e.g., smartcard where field polynomials are often fixed for the sake of lowering chip area.

The rest of paper is organized as follows: Section 2 introduces the representation of shift polynomial basis (SPB). Based on this representation a new bit-parallel multiplier architecture is proposed in Section 3. Section 4 presents the comparison between the proposed multiplier and some others. Finally the conclusions are drawn in Section 5.

2. Shifted polynomial basis (SPB)

Shifted polynomial basis (SPB) is first introduced by H. Fan and Y. Dai [5] which is derived from polynomial basis (PB) by adding a shift variable into each field element in order to improve the efficiency of multiplier. In PB representation, each element of GF(2^m) is represented by a different binary polynomial of degree less than m. More explicitly, a bit string $(a_{m-1}, a_{m-2}, \ldots, a_1, a_0)$ is taken to represent binary polynomial as

$$a(x) = \sum_{i=0}^{m-1} a_i x^i = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1 x + a_0,\ a_i \in \mathrm{GF}(2).$$

Here the set $M = \{x^{m-1}, x^{m-2}, \ldots, x, 1\}$ represents a polynomial basis.

The addition of bit strings corresponds to addition of binary polynomials. Multiplication is defined in terms of an irreducible binary polynomial $f(x)$ of degree m, called the *field polynomial* for the representation. The product of two elements is simply the product of the corresponding polynomials, reduced modulo $f(x)$.

Here goes the definition of SPB over GF(2^m) in GF(2).

Definition 1. (See [5].) Let v be an integer and the ordered set $M = \{x^i \mid 0 \leqslant i \leqslant m-1\}$ be a polynomial basis of GF(2^m) over GF(2). The ordered set $x^v M := \{x^{i+v} \mid 0 \leqslant i \leqslant m-1\}$ is called the shifted polynomial basis (SPB) with respect to M.

In reality, let $f(x) = x^m + x^k + 1$ be an irreducible trinomial over GF(2), $M = \{x^i \mid 0 \leqslant i \leqslant m-1\}$ be a PB and $x^{-v}M := \{x^{i-v} \mid 0 \leqslant i \leqslant m-1\}$ be an SPB, where $0 \leqslant v \leqslant m-1$ and x is a root of $f(x) = 0$. It has been proved that the best value of v is k or $k-1$ with which the multiplier has lowest complexities [5]. From now on, we denote that v equals to k and $x^{-k}M := \{x^{i-k} \mid 0 \leqslant i \leqslant m-1\}$ is the SPB. A field element $a(x)$ can be uniquely represented as $a(x) = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0) = x^{-k}\sum_{i=0}^{m-1} a_i x^i$ with respect to SPB.

It is easy to transform the elements between PB and SPB representations. Let $d(x) = \sum_{i=0}^{m-1} d_i x^i$ and $a(x) = \sum_{i=-v}^{m-1-v} a_{v+i} x^i$ be two elements represented in PB and SPB. The conversions between these two representations are showed by the following two formulae:

$$\begin{aligned} d(x) &= \sum_{i=0}^{m-1} d_i x^i \\ &= \sum_{i=0}^{m-1-v} d_i x^i + \sum_{i=m-v}^{m-1} d_i \left(x^{v+i-m} + x^{i-m}\right) \\ &= \left(\sum_{i=0}^{m-1-v} d_i x^i + \sum_{i=-v}^{-1} d_{m+i} x^i\right) + \sum_{i=0}^{v-1} d_{m+i-v} x^i, \end{aligned}$$

$$\begin{aligned} a(x) &= \sum_{i=-v}^{m-1-v} a_{v+i}x^i = \sum_{i=0}^{m-1-v} a_{v+i}x^i \\ &\quad + \sum_{i=-v}^{-1} a_{v+i}\left(x^{m+i} + x^{v+i}\right) \\ &= \left(\sum_{i=0}^{m-1-v} a_i x^i + \sum_{i=m-v}^{m-1} a_{v-m+i}x^i\right) + \sum_{i=0}^{v-1} a_i x^i. \end{aligned}$$

According to the above formulae, the conversion between these two representations only needs v XOR gates and $1T_X$ delay with parallel computing.

Multiplication on SPB is the same as that on PB except that reduction step abides by two formulae:

$$\begin{aligned} &x^i = x^{k+i-m} + x^{i-m}, \quad \text{where} \\ &\quad m - v \leqslant i \leqslant 2m - 2 - 2v, \\ &x^i = x^{m+i} + x^{k+i}, \quad \text{where } -2v \leqslant i \leqslant -(v+1). \end{aligned}$$

If the irreducible trinomial is $f(x) = x^m + x^{\frac{m}{2}} + 1$ where $k = v = \frac{m}{2}$, we have:

$$\begin{aligned} &x^i = x^{i-\frac{m}{2}} + x^{i-m}, \quad \text{where } \frac{m}{2} \leqslant i \leqslant m-2, \\ &x^i = x^{m+i} + x^{\frac{m}{2}+i}, \quad \text{where } -m \leqslant i \leqslant -\left(\frac{m}{2}+1\right). \end{aligned}$$

3. Multiplier based on SPB

Karatsuba method has been used to improve the efficiency of bit-parallel multiplier for GF(2^m) generated by an AOP (All-One Polynomial) and a trinomial in [1,7,9]. This method can reduce the space complexity by approximately a factor of 3/4 because it replaces the multiplication by three half-sized integers multiplications. This method, however, will increase the time delay which makes the decrease of space complexity less attractive. Here, by using SPB representation, we modify the Karatsuba method and propose a new multiplier architecture with significantly low space complexity and time delay in the fields generated by EST.

Assume that $a(x) = x^{-\frac{m}{2}} \sum_{i=0}^{m-1} a_i x^i$, $b(x) = x^{-\frac{m}{2}} \sum_{i=0}^{m-1} b_i x^i$ are two elements in SPB representation. We partition $a(x) = A \cdot x^{-\frac{m}{2}} + B$ and $b(x) = C \cdot x^{-\frac{m}{2}} + D$, where

$$\begin{aligned} A &= \sum_{i=0}^{\frac{m}{2}-1} a_i x^i, \qquad B = \sum_{i=0}^{\frac{m}{2}-1} a_{i+\frac{m}{2}} x^i, \\ C &= \sum_{i=0}^{\frac{m}{2}-1} b_i x^i, \qquad D = \sum_{i=0}^{\frac{m}{2}-1} b_{i+\frac{m}{2}} x^i. \end{aligned}$$

Then, we multiply $a(x)$ and $b(x)$ with Karatsuba method and do some transformations as follows:

$$\begin{aligned} S &= a(x) \cdot b(x) \\ &= \left(A \cdot x^{-\frac{m}{2}} + B\right)\left(C \cdot x^{-\frac{m}{2}} + D\right) \\ &= AC \cdot x^{-m} + BD + (AC + BD)x^{-\frac{m}{2}} \\ &\quad + (A+B)(C+D)x^{-\frac{m}{2}} \\ &= \left(AC \cdot x^{-\frac{m}{2}} + BD\right)x^{-\frac{m}{2}} \\ &\quad + \left(AC \cdot x^{-\frac{m}{2}} + BD\right) + (A+B)(C+D)x^{-\frac{m}{2}}. \end{aligned} \tag{1}$$

The right side of (1) can be divided into two parts: S_{re}, which needs further reductions modulo $f(x)$ and S_{nore}, which does not need any reductions because the exponents of all elements in S_{nore} are located in the interval of $[-\frac{m}{2}, \frac{m}{2} - 1]$. These two parts are listed separately as follow:

$$\begin{aligned} S_{\text{re}} &= \left(AC \cdot x^{-\frac{m}{2}} + BD\right)x^{-\frac{m}{2}} + \left(AC \cdot x^{-\frac{m}{2}} + BD\right), \\ S_{\text{nore}} &= (A+B)(C+D)x^{-\frac{m}{2}}. \end{aligned}$$

(i) We consider S_{re} in detail first. Let

$$AC = \left(\sum_{i=0}^{\frac{m}{2}-1} a_i x^i\right) \cdot \left(\sum_{i=0}^{\frac{m}{2}-1} b_i x^i\right) = \sum_{i=0}^{m-2} p_i x^i.$$

These p_is can be computed as follows:

$$p_i = \begin{cases} \sum_{j=0}^{i} a_j b_{i-j}, & 0 \leqslant i \leqslant \frac{m}{2} - 1, \\ \sum_{j=i-\frac{m}{2}+1}^{\frac{m}{2}-1} a_j b_{i-j}, & \frac{m}{2} \leqslant i \leqslant m-2. \end{cases} \tag{2}$$

Similarly, we get the coefficients q_is of $BD = \sum_{i=0}^{m-2} q_i x^i$ as:

$$q_i = \begin{cases} \sum_{j=0}^{i} a_{j+\frac{m}{2}} b_{i-j+\frac{m}{2}}, & 0 \leqslant i \leqslant \frac{m}{2} - 1, \\ \sum_{j=i-\frac{m}{2}+1}^{\frac{m}{2}-1} a_{j+\frac{m}{2}} b_{i-j+\frac{m}{2}}, & \frac{m}{2} \leqslant i \leqslant m-2. \end{cases} \tag{3}$$

According to (2) and (3), the result of the expression $AC \cdot x^{-\frac{m}{2}} + BD$ can be computed quickly.

$$\begin{aligned} &AC \cdot x^{-\frac{m}{2}} + BD \\ &\quad = \sum_{-\frac{m}{2}}^{m-2} z_i x^i = \begin{cases} p_{i+\frac{m}{2}}, & -\frac{m}{2} \leqslant i \leqslant -1, \\ p_{i+\frac{m}{2}} + q_i, & 0 \leqslant i \leqslant \frac{m}{2} - 1, \\ q_i, & \frac{m}{2} \leqslant i \leqslant m-2. \end{cases} \end{aligned} \tag{4}$$

From (4), we can find that, for $-\frac{m}{2} \leqslant i \leqslant -1$, z_i contains $(i + \frac{m}{2} + 1)$ elements, for $0 \leqslant i \leqslant \frac{m}{2} - 1$, z_i contains $\frac{m}{2}$ elements and for $\frac{m}{2} \leqslant i \leqslant m - 2$, z_i contains $(m - 1 - i)$ elements. Thus, circuit implementation of $(AC \cdot x^{-\frac{m}{2}} + BD)$ requires $\frac{m}{2} \cdot \frac{m}{2} - m + \frac{m}{2} \cdot \frac{m}{2} = \frac{m^2}{2} - m$ XOR gates. The calculations of AC and BD both need

Table 1
The space and time complexities of $S_{re} = (AC \cdot x^{-\frac{m}{2}} + BD) + (AC \cdot x^{-\frac{m}{2}} + BD)x^{-\frac{m}{2}} \bmod f(x)$

Operation	#AND	#XOR	Time delay
$(AC \cdot x^{-\frac{m}{2}} + BD)$	$\frac{m^2}{2}$	$\frac{m^2}{2} - m$	$T_A + \lceil \log_2 \frac{m}{2} \rceil T_X$
$(AC \cdot x^{-\frac{m}{2}} + BD) + (AC \cdot x^{-\frac{m}{2}} + BD)x^{-\frac{m}{2}}$		m	$1T_X$
Total	$\frac{m^2}{2}$	$\frac{m^2}{2}$	$T_A + \lceil \log_2 m \rceil T_X$

$\frac{m}{2} \cdot \frac{m}{2} = \frac{m^2}{4}$ AND gates. As a result, $(AC \cdot x^{-\frac{m}{2}} + BD)$ totally requires $\frac{m^2}{2}$ AND gates, $(\frac{m^2}{2} - m)$ XOR gates and the time delay is $(T_A + \lceil \log_2 \frac{m}{2} \rceil T_X)$.

Note that S_{re} needs to be reduced modulo $f(x)$ and we partition $(AC \cdot x^{-\frac{m}{2}} + BD)$ into three parts named r_1, r_2, r_3.

$$AC \cdot x^{-\frac{m}{2}} + BD \triangleq r_1 \cdot x^{-\frac{m}{2}} + r_2 + r_3 \cdot x^{\frac{m}{2}},$$

where $r_1 = \sum_{i=-\frac{m}{2}}^{-1} z_i x^{i+\frac{m}{2}}$, $r_2 = \sum_{i=0}^{\frac{m}{2}-1} z_i x^i$, $r_3 = \sum_{i=\frac{m}{2}}^{m-2} z_i x^{i-\frac{m}{2}}$. Sequentially we have

$$\begin{aligned}&\left(AC \cdot x^{-\frac{m}{2}} + BD\right) \cdot x^{-\frac{m}{2}} \\ &\quad = r_1 \cdot x^{-m} + r_2 \cdot x^{-\frac{m}{2}} + r_3.\end{aligned}$$

Because the exponents of $r_3 \cdot x^{\frac{m}{2}}$ in $(AC \cdot x^{-\frac{m}{2}} + BD)$ and $r_1 \cdot x^{-m}$ in $(AC \cdot x^{-\frac{m}{2}} + BD) \cdot x^{-\frac{m}{2}}$ are beyond the range $[-\frac{m}{2}, \frac{m}{2} - 1]$, they need to be reduced as follows:

$$\begin{aligned} S_{\text{re}} \bmod f(x) &= \big(r_1 \cdot x^{-\frac{m}{2}} + r_2 + r_3 \cdot x^{\frac{m}{2}} \\ &\quad + r_1 \cdot x^{-m} + r_2 \cdot x^{-\frac{m}{2}} + r_3\big) \bmod f(x) \\ &= r_1 \cdot x^{-\frac{m}{2}} + r_2 + r_3 + r_3 \cdot x^{-\frac{m}{2}} \\ &\quad + r_1 + r_1 \cdot x^{-\frac{m}{2}} + r_2 \cdot x^{-\frac{m}{2}} + r_3 \\ &= (r_1 + r_2) + (r_2 + r_3)x^{-\frac{m}{2}}. \end{aligned} \tag{5}$$

In (5), identical parts are removed under the addition law in GF(2). Therefore (5) needs m XOR gates and require $1T_X$ delay. In conclusion, the generation of S_{re} needs $\frac{m^2}{2}$ AND gates and $(\frac{m^2}{2} - m) + m = \frac{m^2}{2}$ XOR gates with time delay of $T_A + (1 + \lceil \log_2 \frac{m}{2} \rceil)T_X = T_A + \lceil \log_2 m \rceil T_X$. The space and time complexity on computing S_{re} are summarized in Table 1.

(ii) Here we consider S_{nore} in detail. Because $S_{\text{nore}} = (A + B)(C + D)x^{-\frac{m}{2}}$ needs no further reduction, it can be carried out by an $\frac{m}{2}$-fold left shift of $(A + B)(C + D)$. The shift operation can be realized by a simple rewiring without any logic gates. The space and time complexity on computing S_{nore} are summarized in Table 2.

From Tables 1 and 2, the computations of S_{re} and S_{nore} have the same time delay so they can be calculated in parallel simultaneously.

Since $C = S \bmod f(x) = S_{\text{re}} \bmod f(x) + S_{\text{nore}}$, another m XOR gates and $1T_X$ delay should be added when computing the final result. The total space complexity and time complexity of the proposed architecture can be calculated from Tables 1, 2 and extra gates on adding S_{re} and S_{nore} together:

Table 2
The space and time complexities of $S_{\text{nore}} = (A + B)(C + D)x^{-\frac{m}{2}}$

Operation	#AND	#XOR	Time delay
$(A + C), (B + D)$		m	$1T_X$
$(A + C)(B + D)x^{-\frac{m}{2}}$	$\frac{m^2}{4}$	$(\frac{m}{2} - 1)^2$	$T_A + \lceil \log_2 \frac{m}{2} \rceil T_X$
Total	$\frac{m^2}{4}$	$\frac{m^2}{4} + 1$	$T_A + \lceil \log_2 m \rceil T_X$

Table 3
Comparison of bit-parallel multipliers when $f(x) = x^m + x^{\frac{m}{2}} + 1$

Proposals	#AND	#XOR	Time delay
Wu [8]	m^2	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$
Sunar [12]	m^2	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$
Imana [15]	m^2	$m^2 - \frac{m}{2}$	$T_A + (1 + \lceil \log_2 m \rceil)T_X$
Our proposal	$\frac{3}{4}m^2$	$\frac{3}{4}m^2 + m + 1$	$T_A + (1 + \lceil \log_2(m-1) \rceil)T_X$

$$\#\text{AND} = \frac{m^2}{2} + \frac{m^2}{4} = \frac{3}{4}m^2,$$

$$\#\text{XOR} = \frac{m^2}{2} + \frac{m^2}{4} + 1 + m = \frac{3}{4}m^2 + m + 1,$$

$$\text{Time delay} = T_A + \big(1 + \lceil \log_2 m \rceil\big)T_X.$$

Because m is even, $\lceil \log_2 m \rceil = \lceil \log_2(m - 1) \rceil$, the time complexity can be rewritten as $T_A + (1 + \lceil \log_2(m - 1) \rceil)T_X$.

4. Comparison

In the fields generated by trinomials, low complexity multipliers mainly use polynomial basis. Table 3 gives a comparison of four different implementations of bit-parallel multipliers in the class of fields generated by an equally-spaced trinomial $x^m + x^{\frac{m}{2}} + 1$ according to space complexity and time complexity. From Table 3, the proposed multiplier requires about 25 percent fewer circuit gates than the previous best architectures while with the same time complexity of $T_A + (1 + \lceil \log_2(m - 1) \rceil)T_X$. This merit enables the proposed multiplier to be used in space critical area such as smartcard, RFID tags, etc.

5. Conclusion

In this paper, a new bit-parallel multiplier architecture is proposed. In this architecture, SPB and Karatsuba method are combined which can reduce the time complexity and space complexity, respectively. This multiplier can be used in area-critical occasion because of its low space complexity in GF(2^m) defined by EST. To find more efficient polynomials which can use the method proposed in this paper should be the future work.

References

[1] M. Elia, M. Leone, C. Visentin, Low complexity bit-parallel multipliers for GF(2^m) with generator polynomial $x^m + x^k + 1$, Electronic Letters 35 (7) (1999) 551–552.

[2] A. R-Masoleh, M.A. Hasan, A new construction of Massey–Omura parallel multiplier over GF(2^m), IEEE Transactions on Computers 51 (5) (2002) 511–520.

[3] R. Lidl, H. Niederreiter, Introduction to Finite Fields and Their Applications, Cambridge Univ. Press, New York, 1994.

[4] D.E. Knuth, The Art of Computer Programming, vol. 2, Addison-Wesley, 1998.

[5] H. Fan, Y. Dai, Fast bit-parallel GF(2^n) multiplier for all trinomials, IEEE Transactions on Computers 54 (4) (2005) 485–490.

[6] A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, T. Yaghoobian, Applications of Finite Fields, Kluwer Academic, 1993.

[7] M. Leone, A new low complexity parallel multiplier for a class of finite fields, in: Cryptographic Hardware and Embedded Systems (CHES 2001), 2001, pp. 160–170.

[8] H. Wu, Montgomery multiplier and squarer for a class of finite fields, IEEE Transactions on Computers 51 (5) (2002) 521–529.

[9] K.-Y. Chang, D. Hong, H.-S. Cho, Low complexity bit-parallel multiplier for GF(2^n) defined by all-one polynomials using redundant representation, IEEE Transactions on Computers 54 (12) (2005) 1628–1630.

[10] H. Wu, M.A. Hasan, I.F. Blake, S. Gao, Finite field multiplier using redundant representation, IEEE Transactions on Computers 51 (11) (2002) 1306–1315.

[11] Ç.K. Koç, B. Sunar, Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields, IEEE Transactions on Computers 47 (3) (1998) 353–356.

[12] B. Sunar, Ç.K. Koç, Mastrovito multiplier for all trinomials, IEEE Transactions on Computers 48 (5) (2002) 522–527.

[13] D. Hankerson, A. Menezes, S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.

[14] H. Wu, M.A. Hasan, I.F. Blake, New low-complexity bit-parallel finite field multipliers using weakly dual bases, IEEE Transactions on Computers 47 (11) (2002) 1223–1234.

[15] J.L. Imana, J.M. Sanchez, F. Tirado, Bit-parallel finite field multipliers for irreducible trinomials, IEEE Transactions on Computers 55 (5) (2006) 520–533.