Indian Journal of Science and Technology, *Vol 8(34), DOI: 10.17485/ijst/2015/v8i34/86663, December 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Design of Fast Finite Field Multiplier based on Noramal Basis

Jae Yeon Choi*

Department of Information and Communication, Namseoul University, Cheonan City, Chungnam - 31020, South Korea; cjy@nsu.ac.kr

Abstract

Additional cost is not required to squaring operation; the finite field algebraic calculation using normal basis field arithmetic has advantages. Using this property, more simplified schematic of multiplier which needs a half of latency than the other existing multipliers is proposed and designed. The performance of the proposed multipliers is presented. The various types of multipliers are suggested to be selected according to the parameters such as the circuit size and the waiting time. The result shows the proposed structure has better performance with simpler complexity and shorter waiting time compared with the existing multipliers.

Keywords: Cryptography, Fast Multiplier, Finite Field, Non-Binary Field Arithmetic, Normal Basis

1. Introduction

Arithmetic operations in GF(2^m) play a vital role in encoding and decoding certain error-correcting codes. It is also used importantly in data encryption and decryption. Computing addition, multiplication, division and square roots in GF(2^m) is already described in the various branches of communication system. A number of additions, multiplications, and multiplicative inversions in GF(2^m) are required to implement logical devices such as decoders for error correcting codes like BCH codes as well as Reed Solomon codes[1-3]. Some algorithm using exchanging key method had suggested by[4] that is one of the examples of information-protecting applications. VLSI fabrication makes possible some arrays of logical devices to perform such functions. Any reduction in either the cost or the computation time of the logical devices required to perform these operations may result in considerable improvement of the design of decoders. Recent developments in VLSI technology are becoming increasingly attractive because of their easy fabrication. The work described in this paper presents mainly a number of circuits for computing the product of two arbitrary elements of GF(2^m) because an error control decoder deals with multipliers to a considerable extent. To improve the performance of a cryptographic communication system or data file system, an error correcting code can be concatenated between the enciphering and the deciphering.

A complete implementation of multipliers for a non-binary finite field is presented; these decoders are typically implemented using time-domain techniques. The decoding of a received non-binary word requires that three successive computational processes performed over GF(2^m) be executed. These processes are the syndrome calculation, error-locator polynomial determination, and the Chien search with error-value computation for non-binary codes. However, the details of implementing these functions may be modified somewhat depending on the code and the required data rate. To achieve a very efficient design, the designer must carefully match the architecture of the central processing unit to the hardware requirement of the computational subroutines.

In a low-data-rate implementation, the decoder architecture is most efficiently organized as a special-purpose

*Author for correspondence

processor having a processor which may require one or more GF(2^m) multipliers depending on the required data rate, random-access memory for buffering and temporary storage of intermediate outcomes, and the read only memory for storage of computational subroutines. When the syndrome computation is 0, no error correction is necessary. Accordingly, all of the operations associated with the Berlekamp or Peterson algorithms and Chien's search may be skipped. The single error case is easily accommodated through special calculations, thus, in this case, the average computational requirement is substantially reduced.

Non binary code's decoders for implementing the syndrome computation and the Chien search are basically shift registers with GF(2^m) multipliers and adders attached. The iterative algorithm for solving the error locator polynomial equation provides a technique that can be implemented in the central processor with an associated GF(2^m) arithmetic unit and memory at data rates in the several gigabits per second range. If the multiplication is done using a general GF(2^m) multiplier, then a custom integrated circuit performing this function could be used in the syndrome computation, the Chien search, and the processor. Such a circuit would substantially reduce the implementation complexity of very high speed decoders.

We discuss an extremely important and practical class of codes, the Reed Solomon codes. A Reed Solomon code is a cyclic symbol error correcting code and the typical non binary code for burst error correction. Theses codes are coming into widespread use in many communications and computer storage systems. In particular, a concatenated Reed Solomon convolutional encoding system has been adopted for the deep space downlink. A convolutional code could be used as the inner code and achieve excellent performance with relatively high overall code rates. Since the convolutional code would operate at a moderately high error rate, the use of short constraint length codes with Viterbi decoding is the most promising approach. Also, secure communications systems commonly use a non-binary code as one method for protection against jamming.

(61, 50) shortened code with symbols from GF(2^6) is used for error control. The code used in another disk storage system is a shortened non-binary code with symbols from the field GF(2^8). Each data block consists of 512 bytes. When a data block is recorded, nine parity check bytes are appended to it for error control during the read operation.

The importance of this non-binary codes is partly due to their superior error correcting capabilities, but it is equally due to the availability of an efficient decoding algorithm. The Berlekamp iterative algorithm can be used for efficient decoding of this non-binary codes if a slight modification is made that requires the calculation of the error values at the error locations.

This code is a block sequence of finite field GF(2^m) of 2^m binary symbols, where m is the number of bits per symbol. This sequence of symbols can be viewed as the coefficients of a code polynomial where the field elements included GF(2^m).

Design of multiplier for finite field can be achieved by three different ways. Bit-serial conversion multiplier method was proposed[5-7]. It takes some clock cycles to complete a multiplication, but has some disadvantage of taking time and large sizes. Next method is bit-parallel, but it has also weak points, that is, large size and costs[8,9] and the third method is word-level multiplier which is of great utility[10-13].

2. The Non Binary Field Multiplier Over $GF(2^m)$

We can take a normal basis of $GF(2^m)$ over $GF(2)$ as

$$N = \{\beta, \beta^{2^1}, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\} \quad (1)$$

Where $\beta \in GF(2^m)$. For any positive integer m which is greater than 1, we can take normal basis $m > 1$, then any field element $A \in GF(2^m)$ can be expressed by combining the elements of N linearly, i.e.,

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, a_1, \cdots, a_{m-1}) \quad (2)$$

where $a_i \in GF(2)$, $0 \le i \le m-1$,are represented as coordinates of field element A for N. A^2 is costless parameter and can be performed with ease by carrying cyclic shifts to right in some implementation.

$$A^{2^i} = (a_{m-i}, a_{m-i+1}, \cdots, a_{m-i-1})$$

However, multiplication is more complicated than squaring. Recalling shortly about non binary bit-serial normal basis multiplier due to[6], let $A = (a_0, a_1, \ldots, a_{m-1})$ and $B = (b_0, b_1, \ldots, b_{m-1})$ be two elements of $GF(2^m)$, where a_i's and b_i's are each normal basis coordinates individually. Let $C = (c_0, c_1, \ldots, c_{m-1})$ be their product: $C = AB$ Then, any coordinates of C, is a function u of

A and B which can be obtained by a matrix multiplication[13], i.e.,

$$c_{m-1} = u(A, B) = \mathrm{a} \bullet \mathrm{M} \bullet \mathrm{b}^{\mathrm{T}}$$

where M is a binary m ′ m matrix[14], $a = [a_0, a_1, \ldots, a_{m-1}]$, $b = (b_0, b_1, \ldots, b_{m-1})$ and T indicates transposition of vector matrix. The numbers of 1's in M is known as the complexity of the normal basis N[15] and is denoted as C_N which determines the gate counts and hence time delay for a normal basis multiplier. C_N is greater than or equal to $2m-1$. If $C_N = 2m-1$, then the normal basis is known as an optimal normal basis.

Let field elements $A, B \in GF(2^{\mathrm{m}})$ be represented with respect to the normal basis N as

$$A = a_{m-1}\,\beta^{2^{m-1}} + a_{m-2}\,\beta^{2^{m-2}} + \cdots + a_0\beta \quad a_i \in GF(2)$$

$$B = b_{m-1}\,\beta^{2^{m-1}} + b_{m-2}\,\beta^{2^{m-2}} + \cdots + b_0\,\beta \quad b_i \in GF(2)$$

respectively. Then the product A and B can be given by

$$C = A \cdot B = A \cdot \sum b_{m-1}\,\beta^{2^{m-1}} + b_{m-2}\,\beta^{2^{m-2}} + \cdots + b_0\,\beta$$

Also, C can be rewritten like the following

$$C = C_0 + C_1$$

To accomplish equation (8), we can suggest the structure of multiplier like Figure 1. Before performing the multiplication, the register U and V must be reset to 0 with two input elements individually and the register Z also must be cleared, then after clocking m-times, the register Z contains the result of the product $C = AB$.

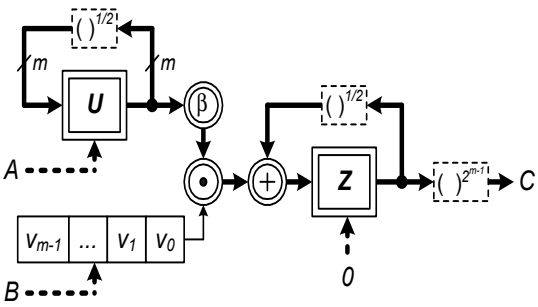

Figure 1. $GF(2^{\mathrm{m}})$ bit-serial multiplier structure.

Let the irreducible polynomial $p(x) = 1 + x^2 + x^5$ which can produce the finite field $GF(2^5)$ and if α is the root of the polynomial $p(x)$ that is, $\mathrm{p}(\alpha) = 0$. We choose $\beta = \alpha^5$, then $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$ is a type 2 optimal normal basis. Using (7), C is computed as

$$C = A \cdot B = A \cdot \sum b_4\,\beta^{2^4} + b_3\,\beta^{2^3} + \cdots + b_0\,\beta$$

Figure 2 shows the structure of bit-serial normal basis multiplier corresponding $GF(2^{\mathrm{m}})$ which is two times faster than an ordinary multiplier.

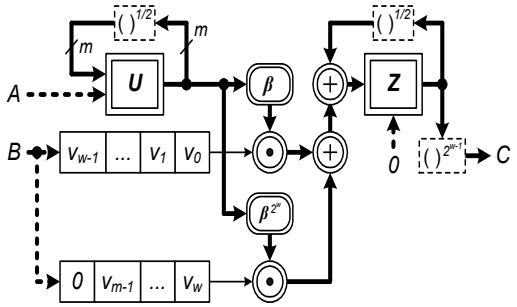

Figure 2. Two times fast bit-serial normal basis multiplier structure.

3. The Structure of the 2 Times Fast Multiplier

The ordinary multiplier shown as Figure 1 needs m times clocks to achieve one multiplying. To accelerate the speed of operation, B which is a element of $GF(2^{\mathrm{m}})$ should be divided into $d(=[m/2])$ words of 2 bits ; $(w = 2)$

$$C = A \cdot \sum b_{m-1}\,\beta^{2^{m-1}} + b_{m-2}\,\beta^{2^{m-2}} + \cdots + b_{m-w}\,\beta^{2^{m-w}}$$

Then, (10) can be written as

$$C = A \cdot B = A \cdot b_4\beta^{2^4} + b_3\beta^{2^3} + b_2\beta^{2^2} \quad A \cdot b_1\beta^2 + b_0\beta$$

$$\left[\left[\, b_4\, A^{1/2^2}\beta^{2^2}\,\right] + b_3\, A^{1/2}\,\beta^{2^2}\right]^2 + b_2 A\,\beta^{2^2}$$

$$+\left[\left[\, b_1\, A^{1/2^2}\beta^{1/2}\,\right] + b_3\, A^{1/2}\,\beta^{1/2}\right]^2 + 0A\,\beta^{1/2}$$

According to the equation (11), bit-serial multiplier structure which is 2 times faster than the ordinary multiplier can be represented. If we want to operate the implemented hardware circuit, the register Z should be cleared and the register U and V must load two input elements. The B register is demanded to be filled with the data reciprocally. If the input B is divided into d words of w bits, then, in the first clock cycle the inputs are the last bit of every word, $b_{m-1}, b_{m-w-1}, \ldots, b_{m-(d-1)w-1}$. In the second clock cycle, the inputs are $b_{m-2}, b_{m-w-2}, \ldots, b_{m-(d-1)w-2}$. Finally, in the w-th clock cycle the inputs are b_{m-w}, b_{m-2w},

Table 1. Number of gates and delay Comparison between various multipliers

Types of Multiplier	No. of AND	No. of XOR	Critical path delay	Basis
WLMO [5]	$d(2m-1)$	$d(2m-2)$	$T_A + [\log_2(2m-1)]\ T_X$	ONB II
AEDS [10]	$d(m-d/2+1/2)$	$d(3m-d-2)$	$T_A + (1+[\log_2 m])\ T_A$	ONB II
XEDS [10]	$d(2m-\mathrm{d})$	$d(2m-\mathrm{d}/2+3/2)$	$T_A + (1+[\log_2 m])\ T_A$	ONB II
w-SMPOI [11]	$d([m/2]+1)+m$	$d(2m-1)$	$2T_A + (3+[\log_2(d-1)])\ T_X$	ONB II
w-SMPOII [11]	$dm+m$	$d(m+[m/2])$	$2T_A + (3+[\log_2(d-1)])\ T_X$	ONB II
Comb Style [12]	dm	$2dm$	$T_A + (1+[\log_2(d+1)])\ T_X$	RNB
Proposed	dm	$d(2m-1)$	$T_A + (1+[\log_2(d+1)])\ T_X$	ONB II

... , b_0. Note that if the subscript of an input bit exceeds m then it is replaced by a zero bit. When the last clock is completed, the contents of the Z register represents the result of the product between A and B. And now, 2 times fast multiplier which was introduced in the previous case can be implemented like Figure 3. By substituting these parameters into (10), we have

$$C = A \cdot B = A \cdot \square b_4\beta^{2^4} + b_3\beta^{2^3} + b_2\beta^{2^2} \square\square A.\square b_1\beta^2 + b_0\beta\square$$
$$\square\left[\left[\ b_4\ A^{1/2^2}\beta^{2^2}\ \right] + b_3\ A^{1/2}\ \beta^{2^2}\ \right]^2 + b_2 A\ \beta^{2^2}$$
$$+\left[\left[\ b_1\ A^{1/2^2}\beta^{1/2}\ \right] + b_3\ A^{1/2}\ \beta^{1/2}\ \right]^2 + 0A\ \beta^{1/2}$$

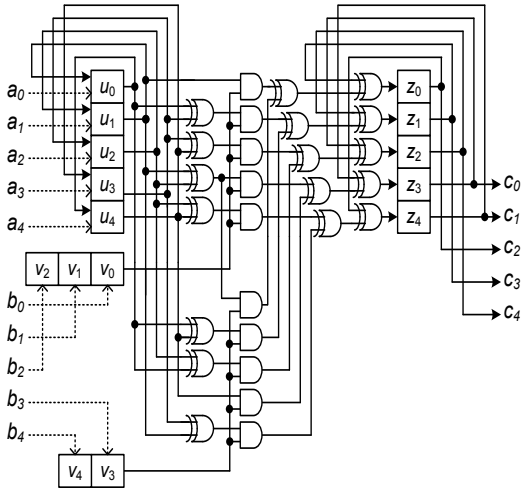

Figure 3. $GF(2^5)$ two times fast bit-serial multiplier schematic.

Table 1 shows the number of the AND gates and XOR gates used in the various type of multiplier. In the first row of Table 1, WLMO means Word-Level Massey-Omura multiplier[5]. In the next rows, AEDS and XEDS mean the AND-efficient digit-serial and XOR-efficient digit-serial multipliers suggested in[10]. w-SMPO I and w-SMPO II represents the word-level sequential multipliers with parallel output type I and type II discribed in[11], and the comb style structure was suggested in[12]. The proposed structure is presented in the last row. The proposed structure has some advantages in size, time delay and cost than any other structure. The other multipliers are more complicated than the proposed.

4. Conclusions

A new non-binary field multiplier which has the 2 times fast speed is proposed in this paper. While the other multipliers have m clock times for waiting time, the proposed multiplier need a half of time for waiting time, and it has also some advantages of the size and complexity. It needs simpler size and shorter time than the other compared multipliers, and its complexity is compared with the other multipliers by calculating the number of the gates needed to implement the multiplier. The concluded results represents that the proposed hardware need less number of digital gates and waiting time. The proposed structure will be helpful to make smaller VLSI fabrication.

5. Acknowledgements

Funding for this paper was provided by Namseoul University.

6. References

1. Rhee MY. Error-Correcting Coding Theory. McGraw-Hill; 1989.
2. Lidl R, Niederreiter H. Introduction to finite fields and their applications. Cambridge Univ Press; 1994.
3. Wicker SB, Bhargava VK. Editors. Reed-Solomon Codes and Their Applications. New York: IEEE Press; 1994.

4. Hankerson D, Menezes A, Vanstone S. Guide to Elliptic Curve Cryptography. Springer-Verlag; 2004.
5. Massey JL, Omura JK. Computational method and apparatus for finite field arithmetic. U.S. patent application; 1984.
6. Beth T, Gollmann D. Algorithm engineering for public key algorithms. IEEE Journal on Selected Areas in Communications. 1989; 7(4):458–66.
7. Agnew GB, Mullin RC, Onyszchuck IM, Vanstone SA. An implementation for a fast public-key cryptosystem. Journal of Cryptology. 1991; 3:63–79.
8. Reyhani-Masoleh A, Hasan MA. A new construction of Massey-Omura parallel multiplier over GF(2^m). IEEE Transactions on Computers. 2002; 51(5):511–20.
9. Koc CK, Sunar B. Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields. IEEE Transactions on Computers. 1998; 47(3):353–56.
10. Reyhani-Masoleh A, Hasan MA. Efficient digit-serial normal basis multipliers over binary extension fields ACM Transactions on Embedded Computing Systems. 2004; 3(3):575–92.
11. Reyhani-Masoleh A, Hasan MA. complexity word- level sequential normal basis Multipliers. IEEE Transactions on Computers. 2005; 54(2):98–110.
12. Namin AH, Wu H, Ahmadi M. Comb architectures for finite field multiplication in $F_2{}^m$. IEEE Transactions on Computers. 2007; 56(7):909–16.
13. Rhee MY. Cryptography and Secure Communications. McGraw-Hill; 1994.
14. IEEE Std 1363-2000. IEEE Standard Specifications for Public-Key Cryptography; 2000 Jan.
15. Mullin RC, Onyszchuk IM, Vanstone SA, Wilson RM. Optimal Normal Bases in $GF(p^n)$. Discrete Applied Mathematics. 1988/89; 22:149–61.