J. KSIAM **Vol.6**, No.2, 75-84, 2002

EFFICIENT BIT SERIAL MULTIPLIERS OF BERLEKAMP TYPE IN $\mathbb{F}_2^m$

SOONHAK KWON

Abstract. Using good properties of an optimal normal basis of type I in a finite field $\mathbb{F}_{2^m}$, we present a design of a bit serial multiplier of Berlekamp type, which is very effective in computing xy^2. It is shown that our multiplier does not need a basis conversion process and a squaring operation is a simple permutation in our basis. Therefore our multiplier provides a fast and an efficient hardware architecture for a bit serial multiplication of two elements in $\mathbb{F}_{2^m}$.

1. Introduction

Arithmetic of finite fields, especially finite field multiplication, found various applications in coding theory, cryptography and digital signal processing. Therefore efficient design of finite field multipliers is needed. A good multiplication algorithm depends on the choice of basis for a given finite field. One of the most widely used finite field multipliers is the Berlekamp's bit serial multiplier [1],[2],[3]. Because of it's low hardware complexity, it has been used in Reed-Solomon encoders which have been utilized in various practical applications such as a deep space probe and a compact disc technology. Let us briefly explain Berlekamp's multiplier over a finite field. We are mainly interested in a finite field $\mathbb{F}_{2^m}$ with characteristic two because it is suitable for a hardware implementation, but the basic theory can be easily extended to an arbitrary finite field. A finite field $\mathbb{F}_{2^m}$ with 2^m elements is regarded as a m-dimensional vector space over $\mathbb{F}_2$. Therefore it has a basis over $\mathbb{F}_2$. One may choose a standard polynomial basis but there exist other types of basis which are useful for their specific purposes.

Definition 1. *Two bases $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ and $\{\beta_1, \beta_2, \cdots, \beta_m\}$ of $GF(2^m)$ are said to be dual if the trace map, $Tr : GF(2^m) \to GF(2)$, with $Tr(\alpha) = \alpha + \alpha^2 + \cdots + \alpha^{2^{m-1}}$, satisfies $Tr(\alpha_i\beta_j) = \delta_{ij}$ for all $1 \le i, j \le m$, where $\delta_{ij} = 1$ if $i = j$, zero if $i \ne j$. A basis $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ is said to be self dual if $Tr(\alpha_i\alpha_j) = \delta_{ij}$.*

2000 *Mathematics Subject Classification.* Primary 11T71, 94A55.

Key words and phrases. normal basis, finite field, feedback shift register, multiplier.

This work is supported in part by KRF 2000-015-DP0005.

Let α be an element of $\mathbb{F}_{2^m}$ be such that $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$ is a basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. Let $\{\beta_0, \beta_1, \cdots, \beta_{m-1}\}$ be the dual basis of $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$. For any $x \in \mathbb{F}_{2^m}$, by considering both polynomial basis and it's dual basis expression of x, we may express x as

$$x = \sum_{i=0}^{m-1} x_i \alpha^i = \sum_{i=0}^{m-1} [x]_i \beta_i.$$

Let $y = \sum_{i=0}^{m-1} y_i \alpha^i$ be another element in $\mathbb{F}_{2^m}$. Then we have the dual basis expression

$$xy = \sum_{k=0}^{m-1} [xy]_k \beta_k,$$

where

$$\begin{aligned} [xy]_k &= Tr(\alpha^k xy) = Tr(\alpha^k x \sum_{i=0}^{m-1} y_i \alpha^i) \\ &= \sum_{i=0}^{m-1} y_i Tr(\alpha^{i+k} x) = \sum_{i=0}^{m-1} y_i [\alpha^k x]_i. \end{aligned}$$

Note that $[\alpha^k x]_i$ is the ith coefficient of the dual basis expression of $\alpha^k x$. On the other hand, we have

$$[\alpha x]_i = Tr(\alpha^i \alpha x) = [x]_{i+1}, \quad i = 0, 1, 2, \cdots, m-2.$$

Also letting $f_0 + f_1 X + f_2 X^2 + \cdots + f_{m-1} X^{m-1} + X^m \in \mathbb{F}_2[X]$ be the irreducible polynomial of α over $\mathbb{F}_2$,

$$[\alpha x]_{m-1} = Tr(\alpha^{m-1} \alpha x) = Tr(\sum_{i=0}^{m-1} f_i \alpha^i x) = \sum_{i=0}^{m-1} f_i Tr(\alpha^i x) = \sum_{i=0}^{m-1} f_i [x]_i.$$

Therefore for each k and i, $[\alpha^k x]_i$ can be computed by the feedback shifting process of previous expression $[\alpha^{k-1} x]_i, i = 0, 1, \cdots, m-1$. The multiplication $[xy]_k = \sum_{i=0}^{m-1} y_i [\alpha^k x]_i$ is realized by the following hardware design of a feedback shift register [1].

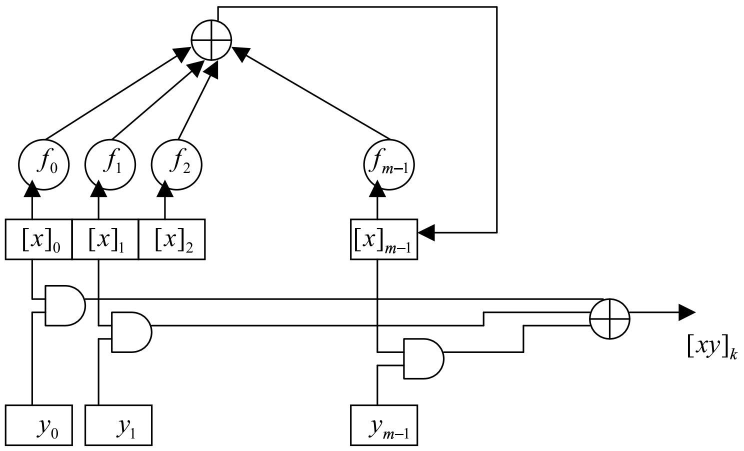

Figure 1. Bit serial arrangement of Berlekamp's dual basis multiplier.

After k-clock cycles, we get $[xy]_k$, the kth coefficient of the dual basis expression of xy. As is obvious from above algorithm, Berlekamp's bit serial multiplier uses a dual basis. That is, to multiply two elements in a finite field, one input y is expressed in terms of a standard polynomial basis, the other input x is expressed in terms of it's dual basis and the resulting output xy is expressed in terms of the dual basis. Therefore we need a basis conversion process in Berlekamp's algorithm, which necessarily increases the hardware complexity in it's implementation. This problem is largely solved in the cases when the corresponding finite field is generated by a root of an irreducible trinomial [2] or a pentanomial of a special kind [3]. Bit serial multipliers applicable for all types of irreducible polynomials are presented in [8]. But they have increased hardware complexity and the operations are relatively slow compared with Berlekamp's multipliers if the basis conversion is unnecessary. One more drawback of Berlekamp's multiplier is that exponentiation, inverse finding procedures are very time consuming when compared with bit parallel multipliers (especially with Massey-Omura type multipliers using normal bases [5],[6],[7]). Not to mention it's high speed because of parallel processing, bit parallel normal basis multipliers are quite effective in such operations as exponentiation, inverse finding since squaring operation is just a cyclic shift in normal basis expression. So various types of normal basis multipliers are being studied. Among them, so called optimal normal basis multipliers [6],[7] require least number of gates than other types of parallel multipliers. There are two types of optimal normal bases [4], namely, type I and type II. Our aim in this paper is to present a design of a bit serial multiplier using type I optimal normal basis which satisfies the following three properties. First, it is generally faster than Berlekamp's bit serial multiplier. Second, squaring operation is just a permutation in our basis. Therefore, exponentiation and inverse finding operations are much faster than Berlekamp's multiplier. Third, it needs no basis conversion process which is required in Berlekamp's algorithm.

2. Normal basis and optimal normal basis of type I

Let α be an element of $\mathbb{F}_{2^m}$ of degree m and let $f(X) = f_0 + f_1 X + \cdots + f_{m-1}X^{m-1} + X^m$ be the irreducible polynomial of α over $\mathbb{F}_2$. Then we have

$$0 = f(\alpha) = f_0 + f_1\alpha + f_2\alpha^2 + \cdots + f_{m-1}\alpha^{m-1} + \alpha^m.$$

From this, it is clear that

$$0 = f(\alpha)^{2^i} = f(\alpha^{2^i}), \quad i = 0, 1, 2, \cdots, m-1,$$

since $f_0, f_1, \cdots, f_{m-1}$ are in $\mathbb{F}_2$ and the characteristic of $\mathbb{F}_{2^m}$ is two. In other words, all the zeros of $f(X)$ are $\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}$. If all the conjugates, $\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}$, of α are linearly independent over $\mathbb{F}_2$, then they form a basis for $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$.

Definition 2. *A basis of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ of the form $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is called a normal basis.*

It is a standard fact [4] that there is always a normal basis in $\mathbb{F}_{2^m}$ for any $m \geq 1$. If an element x in $\mathbb{F}_{2^m}$ is expressed with respect to a normal basis $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$, i.e. if $x = x_0\alpha + x_1\alpha^2 + \cdots + x_{m-1}\alpha^{2^{m-1}}$, then one easily notices

$$x^2 = x_{m-1}\alpha + x_0\alpha^2 + x_1\alpha^{2^2} + \cdots + x_{m-2}\alpha^{2^{m-1}}.$$

That is, x^2 is a right cyclic shift of x with respect the basis $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$. On the other hand, a normal basis expression of a product xy of two different elements x and y in $\mathbb{F}_{2^m}$ is not so simple. This is because the expression $\alpha^{2^i}\alpha^{2^j} = \sum_{k=0}^{m-1} a_k^{ij}\alpha^{2^k}$ may be quite complicated if one does not choose a normal basis properly. Therefore, to find an efficient bit serial multiplication using a normal basis, one has to choose a normal basis so that the coefficients a_k^{ij} in the expression $\alpha^{2^i}\alpha^{2^j}$ are zero for many indices i, j and k. There are not so many normal bases satisfying this condition, but we have one example of such normal basis and it is stated in the following theorem. A detailed proof can be found in [4].

Theorem 1. *Let $\mathbb{F}_{2^m}$ be a finite field of 2^m elements where $m+1 = p$ is a prime. Suppose that 2 is a primitive root $\pmod{p}$. Then letting α be a primitive pth root of unity in $\mathbb{F}_{2^m}$, $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is a basis over $\mathbb{F}_2$.*

Above theorem is based on the fact that the splitting field of the polynomial $x^s - 1 \in \mathbb{F}_2[x]$ over $\mathbb{F}_2$ is $\mathbb{F}_{2^t}$ where t is the least positive integer satisfying $2^t \equiv 1 \pmod{s}$. In the case when $s = p$ is a prime and 2 is a primitive root $\pmod{p}$, we have $\mathbb{F}_2(\alpha) = \mathbb{F}_{2^{p-1}} = \mathbb{F}_{2^m}$. Also since $\{2^i | i = 0, 1, \cdots, p-2\}$ is a reduced residue system $\pmod{p}$, we easily get $\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}\} = \{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^m\}$. Since α is a primitive pth root of unity where $m+1 = p$ is a prime, we have the irreducible polynomial $f(X)$ of α over $\mathbb{F}_2$ as

$$\begin{aligned} f(X) &= (X-\alpha)(X-\alpha^2)(X-\alpha^{2^2})\cdots(X-\alpha^{2^{m-1}}) \\ &= (X-\alpha)(X-\alpha^2)(X-\alpha^3)\cdots(X-\alpha^m) \\ &= \frac{X^{m+1}-1}{X-1} \\ &= 1+X+X^2+\cdots+X^m. \end{aligned}$$

Moreover since $\{\alpha,\alpha^2,\alpha^{2^2},\cdots,\alpha^{2^{m-1}}\}$ and $\{\alpha,\alpha^2,\alpha^3,\cdots,\alpha^m\}$ are same sets, we find

$$\alpha^{2^i}\alpha^{2^j} = \alpha^s\alpha^t = \alpha^{s+t},$$

where one may assume $1 \le s+t \le m$ because $\alpha^{m+1}=1$. Therefore we deduce that the multiplication of two different elements of $\mathbb{F}_{2^m}$ is not so complicated if one use above mentioned normal basis.

Definition 3. *A normal basis in theorem 1 is called an optimal normal basis of type I.*

Optimal normal bases of type I are widely used in the design of bit parallel multipliers. A table in [4, p. 100] says that there is an optimal normal basis of type I when $m = 2,4,10,12,18,28,36,52,58,60,66,82,100,106,\cdots$. In fact, the number of $m \le 2000$ for which there exists an optimal normal basis of type I is 118. This may not be sufficient enough for coding theoretical purposes. But for a cryptographical purpose where one fixes a large finite field $\mathbb{F}_{2^m}$ of a suitable size, we do have a sequence of a large finite field which has an optimal normal basis of type I.

3. Multiplication algorithm

To find an efficient bit serial multiplier with respect to a type I optimal normal basis, we express $x \in \mathbb{F}_{2^m}$ as $x = \sum_{i=0}^m x_i\alpha^i$. Note that the expression is not unique since $\{1,\alpha,\alpha^2,\cdots,\alpha^m\}$ is no longer a basis over $\mathbb{F}_2$. But the expression $x=\sum_{i=0}^m x_i\alpha^i$ can easily be represented using the basis $\{\alpha,\alpha^2,\alpha^3,\cdots,\alpha^m\}$ as follows. If $x_0=0$, x has already an expression with respect to the basis $\{\alpha,\alpha^2,\alpha^3,\cdots,\alpha^m\}$. If $x_0=1$ then,

$$x = \sum_{i=0}^m x_i\alpha^i = \sum_{i=0}^m x_i\alpha^i + \sum_{i=0}^m \alpha^i = \sum_{i=1}^m (x_i+1)\alpha^i,$$

since the minimal polynomial of α over $\mathbb{F}_2$ is

$$1+X+X^2+\cdots+X^m \in \mathbb{F}_2[X].$$

Theorem 2. *Let $x=\sum_{i=0}^m x_i\alpha^i$ and $y=\sum_{i=0}^m y_i\alpha^i$ be two elements in $\mathbb{F}_{2^m}$, where $\{\alpha,\alpha^2,\cdots,\alpha^{2^{m-1}}\}$ is a type I optimal normal basis. Then we have $xy=\sum_{i=0}^m (xy)_i\alpha^i$, where the kth coefficient $(xy)_k$ is*

$$y_0x_k + y_1x_{k-1} + \cdots + y_kx_0 + y_{k+1}x_m + y_{k+2}x_{m-1} + \cdots + y_mx_{k+1}.$$

Proof.

$$\begin{aligned}
xy &= \sum_{i=0}^{m} x_i\alpha^i \sum_{j=0}^{m} y_j\alpha^j \\
&= \sum_{i=0}^{m}\sum_{j=0}^{m} x_i y_j \alpha^{i+j} \\
&= \sum_{i+j\le m} x_i y_j \alpha^{i+j} + \sum_{i+j>m} x_i y_j \alpha^{i+j} \\
&= \sum_{k=0}^{m}\sum_{i+j=k} x_i y_j \alpha^{k} + \sum_{k=0}^{m-1}\sum_{i+j=m+1+k} x_i y_j \alpha^{k} \\
&= \sum_{k=0}^{m}(y_0x_k + y_1x_{k-1} + \cdots + y_kx_0)\alpha^k \\
&\quad + \sum_{k=0}^{m-1}(y_{k+1}x_m + y_{k+2}x_{m-1} + \cdots + y_mx_{k+1})\alpha^k \\
&= \sum_{k=0}^{m}(y_0x_k + \cdots + y_kx_0 + y_{k+1}x_m + \cdots + y_mx_{k+1})\alpha^k,
\end{aligned}$$

which completes the proof. □

Using above theorem, we have the matrix multiplication form of $(xy)_k$ as

$$(xy)_k = (x_k, x_{k-1}, \cdots, x_0, x_m, x_{m-1}, \cdots, x_{k+1})(y_0, y_1, \cdots, y_m)^T,$$

where $(y_0, y_1, \cdots, y_m)^T$ is the transposition of the row vector $(y_0, y_1, \cdots, y_m)$. From

$$(xy)_{k+1} = (x_{k+1}, x_k, \cdots, x_0, x_m, x_{m-1}, \cdots, x_{k+2})(y_0, y_1, \cdots, y_m)^T,$$

we notice that the row vector $(x_{k+1}, x_k, \cdots, x_0, x_m, x_{m-1}, \cdots, x_{k+2})$ is just a right cyclic shift of $(x_k, x_{k-1}, \cdots, x_0, x_m, x_{m-1}, \cdots, x_{k+1})$ by one position. Therefore the multiplication xy is easily realized by the shift register arrangement shown in Fig. 2 which is noticed in [9].

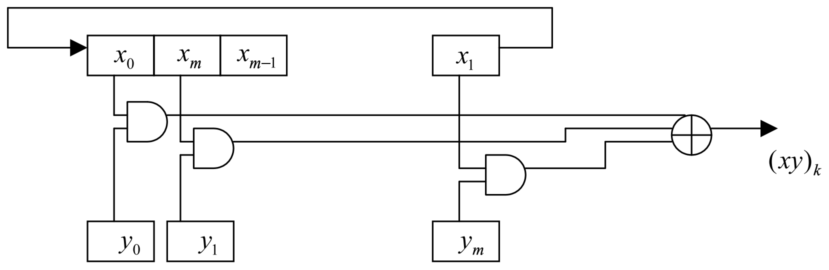

Figure 2. Bit serial arrangement using an optimal normal basis of type I.

The shift register is initially loaded with $(x_0, x_m, x_{m-1}, \cdots, x_1)$. After k clock cycles, we get $(xy)_k$, the kth coefficient of xy with respect to the extended basis $\{1, \alpha, \alpha^2, \cdots, \alpha^m\}$. One may further improve above result in [9] by using the property of a normal basis.

That is, we will present a bit serial multiplier computing xy^2 in $\mathbb{F}_2^m$. In the theory of error correcting codes, the following types of product sum operations in $GF(2^m)$, $xy+z$ and xy^2+z, are the most frequently used arithmetic operations. Those product sum operations are easily realized by using extra registers and XOR gates once you have the circuits which compute xy and xy^2. Since a squaring operation is a permutation in our basis, the operation xy^2 is computed by the following two steps. First, compute y^2 by permutating the coefficients of y. Second, compute xy^2 using the bit serial arrangement in Fig. 2. However, we do not have to actually permutate the coefficients of y to compute xy^2. We only need the second step by slightly adjusting the wiring in Fig. 2. The idea is derived in the following way. First, note that the coefficient $(xy)_k$ in theorem 2 can be reexpressed in the following form,

$$(xy)_k = \sum_{j=0}^{m} x_{k-j}y_j = (x_k, x_{k-1}, \cdots, x_{k-m})(y_0, y_1, \cdots, y_m)^T,$$

where it is defined, for all integers s and t, that $x_s = x_t$ if and only if $s \equiv t \pmod{m+1}$. Using this notation on x_s for all integers s, we get the following result.

Theorem 3. $(xy^2)_k = (x_k, x_{k-2}, x_{k-4}, \cdots, x_{k-2m})(y_0, y_1, y_2, \cdots, y_m)^T$.

Proof.

$$\begin{aligned} xy^2 &= \sum_{i=0}^{m} x_i\alpha^i \sum_{j=0}^{m} y_j\alpha^{2j} \\ &= \sum_{i=0}^{m}\sum_{j=0}^{m} x_i y_j \alpha^{i+2j} = \sum_{i=0}^{m}\sum_{j=0}^{m} x_{i-2j}y_j\alpha^i \qquad \square \end{aligned}$$

Now let $2i \equiv j \pmod{m+1}$. Then using $2m/2 \equiv -1 \pmod{m+1}$, we get the expression of $i \equiv 2^{-1}j \pmod{m+1}$ as

$$i \equiv \frac{j}{2} \quad if\ j = even, \qquad and \quad i \equiv \frac{j+1}{2} + \frac{m}{2} \quad if\ j = odd.$$

Above relation combined with the expression, $(xy^2)_k = \sum_{i=0}^{m} x_{k-2i}y_i = \sum_{j=0}^{m} x_{k-j}y_{2^{-1}j}$, implies that we may express $(xy^2)_k$ as a matrix multiplication form

$$(xy^2)_k = (x_k, x_{k-1}, x_{k-2}, \cdots, x_{k-m+1}, x_{k-m})(y_0, y_{\frac{m}{2}+1}, y_1, \cdots, y_m, y_{\frac{m}{2}})^T.$$

Thus we have

$$(xy^2)_{k+1} = (x_{k+1}, x_k, x_{k-1}, \cdots, x_{k-m+2}, x_{k-m+1})(y_0, y_{\frac{m}{2}+1}, y_1, \cdots, y_m, y_{\frac{m}{2}})^T.$$

Since $k+1 \equiv k-m \pmod{m+1}$, we get $x_{k+1} = x_{k-m}$. From this, we easily notice that $(x_{k+1}, x_k, x_{k-1}, \cdots, x_{k-m+2}, x_{k-m+1})$ is a right cyclic shift of $(x_k, x_{k-1}, x_{k-2}, \cdots, x_{k-m+1}, x_{k-m})$. Above algorithm is realized in the shift register arrangement shown in Fig. 3. Unlike the case of Berlekamp's dual basis multiplier, our multiplier does not need XOR operations among the elements $x_i s$ to determine a shifting element.

Therefore our multiplier is generally faster than Berlekamp's multiplier. Also, since our multiplier is using a normal basis, such arithmetical operations as squaring and xy^2 are very efficiently computed as is explained above. Note that, in the case of the dual basis multiplier, we do not have a similar structure for the multiplication xy^2 and the computation of xy^2 is much more time consuming than the computation of xy.

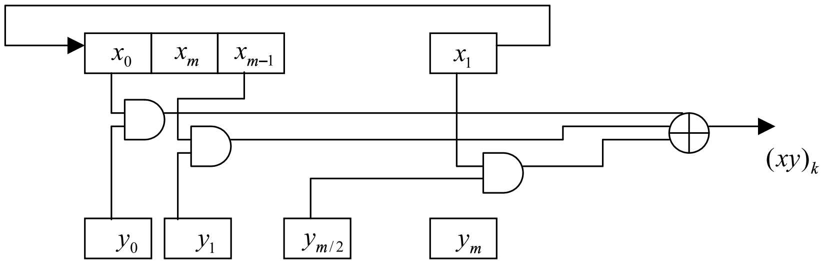

Figure 3. Bit serial arrangement for computing xy^2.

Moreover we do not need a basis conversion process which is required in dual basis multipliers. A similar design for the computation of xy is proposed in [10]. In [10], they used a dual basis argument on $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$. Letting $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ be the dual basis of $\{1, \alpha, \alpha^2, \cdots, \alpha^{m-1}\}$, they expressed the inputs x and y with respect to the dual basis as

$$x = \sum_{i=0}^{m-1} [x]_i \alpha_i, \quad y = \sum_{i=0}^{m-1} [y]_i \alpha_i.$$

They also used one extra memory (or a flip-flop) similar to ours, but the input to the extra memory is $[y]_0 + [y]_1 + \cdots + [y]_{m-1}$. This a great drawback when one implements it to a hardware arrangement, since one needs $m-1$ more XOR gates and a time delay of order $\log m$ is occurred during the initial loading of $[y]_0 + [y]_1 + \cdots + [y]_{m-1}$. That is why only the software implementation is presented in [10]. However, our multiplier has no such problem and our method is well suited to both hardware and software arrangements. The algorithm of software arrangement can be explained as follows. For each $x = \sum_{i=0}^{m} x_i \alpha^i \in \mathbb{F}_{2^m}$, we denote it as a vector form $x = (x_0, x_1, \cdots, x_m)$. Let $x(k) = (x_{m-k+1}, x_{m-k+2}, \cdots, x_m, x_0, x_1, \cdots, x_{m-k})$ denote the right cyclic shift of x by k positions. Now using the relation

$$(xy)_k = \sum_{k=0}^{m} (y_0 x_k + \cdots + y_k x_0 + y_{k+1} x_m + \cdots + y_m x_{k+1}) \alpha^k,$$

we find the following matrix form

$$\begin{pmatrix} (xy)_0 \\ (xy)_1 \\ (xy)_2 \\ \cdot \\ \cdot \\ \cdot \\ (xy)_m \end{pmatrix} = \begin{pmatrix} x_0 & x_m & x_{m-1} & \cdot & \cdot & x_1 \\ x_1 & x_0 & x_m & \cdot & \cdot & x_2 \\ x_2 & x_1 & x_0 & \cdot & \cdot & x_3 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ x_m & x_{m-1} & x_{m-2} & \cdot & \cdot & x_0 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_m \end{pmatrix}$$

$$= \sum_{k=0}^{m} y_k \begin{pmatrix} x_{m-k+1} \\ x_{m-k+2} \\ \vdots \\ x_m \\ x_0 \\ \vdots \\ x_{m-k} \end{pmatrix} = \sum_{k=0}^{m} y_k x(k)^t.$$

Thus we have the following.

Table 1. A multiplication algorithm.

```
Input: $x = (x_0, x_1, \cdots, x_m), y = (y_0, y_1, \cdots, y_m)$
Output: $z = ((xy)_0, (xy)_1, \cdots, (xy)_m)$
$z \leftarrow (0, 0, \cdots, 0)$
for $(k = 0; k \le m; k++)$
{
        if $(y_k \neq 0)$    $z \leftarrow z \oplus x(k)$
}
```

The notation $\oplus$ in above algorithm denotes a bitwise XOR operation of two vectors in $\mathbb{F}_2^{m+1}$. Though the algorithm for xy^2 is not described here in detail, it should be mentioned that it is a routine manner to write down the algorithm by using the property that $\{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^m\}$ and $\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}\}$ are same sets.

References

[1] E.R. Berlekamp, "Bit-serial Reed-Solomon encoders," *IEEE Trans. Inform. Theory*, **28**, pp. 869–874, 1982.

[2] M. Wang and I.F. Blake, "Bit serial multiplication in finite fields," *SIAM J. Disc. Math.*, **3**, pp. 140–148, 1990.

[3] M. Morii, M. Kasahara and D.L. Whiting, "Efficient bit-serial multiplication and the discrete-time Wiener-Hopf equation over finite fields," *IEEE Trans. Inform. Theory*, **35**, pp. 1177–1183, 1989.

[4] A.J. Menezes, *Applications of finite fields*, Kluwer Academic Publisher, 1993.

[5] Ç.K. Koç and B. Sunar, "Low complexity bit-parallel canonical and normal basis multipliers for a class of finite fields," *IEEE Trans. Computers,* **47**, pp. 353–356, 1998.

[6] B. Sunar and Ç.K. Koç, "An efficient optimal normal basis type II multiplier," *IEEE Trans. Computers*,**50**, pp. 83–87, 2001.

[7] M.A. Hasan, M.Z. Wang and V.K. Bhargava, "A modified Massey-Omura parallel multiplier for a class of finite fields," *IEEE Trans. Computers*,**42**, pp. 1278–1280, 1993.

[8] M.A. Hasan and V.K. Bhargava, "Division and bit-serial multiplication over $GF(q^m)$," *IEE Proc. E*, **139**, pp. 230–236, 1992.

[9] S.T.J Fenn, M.G. Parker, M. Benaissa and D. Taylor, "Bit serial multiplication in $GF(2^m)$ using irreducible all one polynomials," *IEE Proc. Comput. Digit. Tech.*,**144**, pp. 391–393, 1997.

[10] C.H. Lee and J.I. Lim, "A new aspect of dual basis for efficient field arithmetic," *Public Key Cryptography*, Lecture Notes in Computer Science **1560**, pp. 12–28, 1999.

Department of Mathematics
Sungkyunkwan University
Suwon 440-746, Korea
email: shkwon@math.skku.ac.kr