A LOW COMPLEXITY AND A LOW LATENCY BIT PARALLEL SYSTOLIC MULTIPLIER OVER *GF*(2^m) USING AN OPTIMAL NORMAL BASIS OF TYPE II

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

Abstract

Using the self duality of an optimal normal basis (ONB) of type II, we present a bit parallel systolic multiplier over $GF(2^m)$ which has a low hardware complexity and a low latency. We show that our multiplier has a latency $m+1$ and the basic cell of our circuit design needs 5 latches (flip-flops). On the other hand, most of other multipliers of the same type have latency $3m$ and the basic cell of each multiplier needs 7 latches. Comparing the gates areas in each basic cell, we find that the hardware complexity of our multiplier is 25 percent reduced from the multipliers with 7 latches.

1. Introduction

Arithmetic of finite fields, especially finite field multiplication, found various applications in many cryptographic and coding theoretical areas. Therefore an efficient design of a finite field multiplier is needed. Though one may design a finite field multiplier in a software arrangement, a hardware implementation has a strong advantage when one wants a high speed multiplier. Moreover, arithmetic of $GF(2^m)$ is easily realized in a circuit design using a few logical gates. A good multiplication algorithm depends on the choice of a basis for a given finite field. In general, there are three types of basis being used, that is, polynomial, dual and normal basis. Some popular multipliers for cryptographical and coding theoretical purposes are Berlekamp's bit serial multiplier [1,2] which use a dual basis, and a bit parallel multiplier of Massey-Omura type [4,5,17,18] which use a normal basis. Above mentioned multipliers and other traditional multipliers have some unappealing characteristics. For example, they have irregular circuit designs. In other words, their hardware structures may be quite different for varying choices of m for $GF(2^m)$, though the multiplication algorithm is basically same for each m. Moreover as m gets large, the propagation delay also increases. So deterioration of the performance is inevitable. A systolic multiplier does not suffer from above problems. It has a regular structure consisting of a number of replicated basic cells, each of which has the same circuit design. So overall structures of systolic multipliers are same and not depending on a particular choice of m for $GF(2^m)$. Furthermore since each basic cell is only connected with its neighboring cells, signals can be propagated at a high clock speed. There are systolic multipliers using a polynomial basis [7,8,10,11,12,13,15] and a dual basis [9]. A bit parallel systolic multiplier in [8] has a comparable or better longest path delay than the multipliers in [7,9,10]. However, it has bidirectional data flows, whereas multipliers in [7,9,10] have unidirectional data flows. A bit parallel systolic multiplier proposed in [10] uses an all one polynomial (AOP) basis. This AOP multiplier has a low cell complexity and a high throughput when compared with other multipliers. However, it is applicable to relatively few finite fields. To be specific, the number of $m \leq 1000$ for which an AOP multiplier in $GF(2^m)$ exists is only 68. On the other hand, most of the multipliers using a standard polynomial basis are applicable to all finite fields. In this paper, we propose a design of a bit parallel systolic multiplier using a type II optimal normal basis (ONB) in $GF(2^m)$. Our multiplier has unidirectional data flows and does not broadcast signals. We show that our bit parallel systolic multiplier has a lower hardware complexity when compared with other systolic multipliers in [7,8,9,11]. A bit parallel systolic multiplier using an AOP basis in [10] has a lower hardware complexity than ours, but our multiplier is applicable to broader class of finite fields. We also show that our multiplier has a latency $m+1$ whereas other multipliers of the same type have latency $3m$ except for the multiplier in [10], where the latency is $m+1$. A low latency and a low hardware complexity multiplier is introduced in [12] but it broadcasts signals and has bidirectional

data flows. In practical situations such as VLSI implementation, broadcastings and bidirectional data flows should be avoided if one wants a reliable and a fault tolerant architecture. It should be mentioned that, though there exists [4,5] a bit parallel multiplier (of Massey-Omura type) using an ONB of type II, we have not yet found a systolic multiplier using the same basis. In fact, except for our multiplier and the multiplier in [10] using an AOP basis (which is not a normal basis, but roughly speaking, it may be viewed as a variant of a type I optimal normal basis), there is no other bit parallel systolic multiplier using a normal basis to our knowledge at this moment.

2. Normal basis and optimal normal basis of type II

Let $GF(2^m)$ be a finite field of 2^m elements. $GF(2^m)$ is a vector space over $GF(2)$ of dimension m. We briefly explain basic finite field arithmetic.

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 $Tr(\alpha_i\alpha_j) = \delta_{ij}$.*

Definition 2. *A basis of $GF(2^m)$ over $GF(2)$ of the form $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is called a normal basis for $GF(2^m)$.*

It is well known [6] that normal bases in $GF(2^m)$ exist for all m. But our main interest is the following type of normal bases which is explained in detail in [6] and also in [4].

Theorem 1. *Let $GF(2^m)$ be a finite field of 2^m elements where $2m + 1 = p$ is a prime. Suppose that either ($\star$) 2 is a primitive root $\pmod p$ or ($\star\star$) -1 is a quadratic non residue $\pmod p$ and 2 generates the quadratic residues $\pmod p$. Then letting $\alpha = \beta + \beta^{-1}$ where β is a primitive pth root of unity in $GF(2^{2m})$, we have $\alpha \in GF(2^m)$ and $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ is a basis over $GF(2)$.*

Definition 3. *A normal basis in theorem 1 is called an optimal normal basis (ONB) of type II.*

Using the assumptions in the previous theorem, one finds easily (see [4].)

$$\alpha^{2^s} = (\beta + \beta^{-1})^{2^s} = \beta^{2^s} + \beta^{-2^s} = \beta^t + \beta^{-t},$$

where $0 < t < p = 2m + 1$ with $2^s \equiv t \pmod p$. Moreover, replacing t by $p - t$ if $m + 1 \le t \le 2m$, we find that $\{\alpha, \alpha^2, \cdots, \alpha^{2^{m-1}}\}$ and $\{\beta + \beta^{-1}, \beta^2 + \beta^{-2}, \cdots, \beta^m + \beta^{-m}\}$ are same sets. That is, $\alpha^{2^s}, 0 \le s \le m - 1$ is just a permutation of $\beta^s + \beta^{-s}, 1 \le s \le m$. In the design of our bit serial multiplier, we need the following self duality of a type II ONB. In fact, it is a special case of a well known fact from the theory of Gauss periods saying that a Gauss period of type (m, k) over $GF(2^m)$ is self dual if and only if k is even, which was proved in [3]. We present an elementary proof of the special case.

Lemma 1. *An optimal normal basis $\{\alpha^{2^s} | 0 \le s \le m-1\}$ of type II in $GF(2^m)$, if it exists, is self dual.*

Proof. After a permutation of basis elements, it can be written as $\{\beta^s + \beta^{-s} | 1 \le s \le m\}$. Note that $Tr((\beta^i + \beta^{-i})(\beta^j + \beta^{-j})) = Tr(\beta^{i-j} + \beta^{-(i-j)} + \beta^{i+j} + \beta^{-(i+j)})$. If $i = j$, then we have $Tr(\beta^{2i} + \beta^{-2i}) = Tr(\alpha^{2^s})$ for some $0 \le s \le m - 1$. Thus the trace value is $\alpha + \alpha^2 + \cdots + \alpha^{2^{m-1}} = 1$ because of the linear independence. Therefore we may assume $i \ne j$. Then replacing $i - j$ by $|i - j|$ if $i - j < 0$ and $i + j$ by $2m + 1 - (i + j)$ if $m + 1 \le i + j \le 2m$ (recall $\beta^{2m+1} = 1$.), we find $Tr((\beta^i + \beta^{-i})(\beta^j + \beta^{-j})) = Tr(\beta^u + \beta^{-u} + \beta^v + \beta^{-v}) = Tr(\beta^u + \beta^{-u}) + Tr(\beta^v + \beta^{-v})$ for some $1 \le u, v \le m$. Since $\beta^u + \beta^{-u}, \beta^v + \beta^{-v}$ are in $\{\alpha^{2^s} | 0 \le s \le m - 1\}$, we have $Tr(\beta^u + \beta^{-u}) + Tr(\beta^v + \beta^{-v}) = 1 + 1 = 0$. □

3. Multiplication algorithm

The proof of lemma 1 implies that many of the notations can be simplified if we define $\alpha_s = \beta^s + \beta^{-s}, 1 \le s \le m$. Therefore from now on, we assume $\{\alpha^{2^s} | 0 \le s \le m - 1\}$ is a type II ONB in $GF(2^m)$ and $\{\alpha_s | \alpha_s = \beta^s + \beta^{-s}, 1 \le s \le m\}$ is a basis obtained after a permutation of the basis elements of the normal basis. For a given $x = \sum_{i=1}^m x_i\alpha_i$ with $x_i \in GF(2)$, by using lemma 1, we have

$$x_s = \sum_{i=1}^m x_i Tr(\alpha_s\alpha_i) = Tr(\alpha_s \sum_{i=1}^m x_i\alpha_i) = Tr(\alpha_s x),$$

for all $1 \le s \le m$. We extend the definition of α_s and x_s for all integers s as follows.

Definition 4. *Let β be a primitive pth ($p = 2m + 1$) root of unity in $GF(2^{2m})$ and let $x \in GF(2^m)$. For each integer s, define α_s and x_s as*

$$\alpha_s = \beta^s + \beta^{-s}, \qquad x_s = Tr(\alpha_s x).$$

Lemma 2. *We have $\alpha_s = 0 = x_s$ if $2m+1$ divides s. Also for all s,*

$$\alpha_{2m+1+s} = \alpha_s = \alpha_{2m+1-s} = \alpha_{-s}$$

and

$$x_{2m+1+s} = x_s = x_{2m+1-s} = x_{-s}.$$

Proof. We have $\alpha_s = \beta^s + \beta^{-s} = 0$ if and only $\beta^s = \beta^{-s}$, that is, $\beta^{2s} = 1$. And this happens whenever $2m+1 = p$ divides s since β is a primitive pth root of unity. Now $\alpha_{2m+1+s} = \beta^{2m+1+s} + \beta^{-(2m+1+s)} = \beta^s + \beta^{-s} = \alpha_s$ is obvious because $\beta^{2m+1} = 1$. Also $\alpha_{2m+1-s} = \beta^{2m+1-s} + \beta^{-(2m+1-s)} = \beta^{-s} + \beta^s = \alpha_s$. The result for x_s instantly follows from the result for α_s. □

Now for any integer s and t, we have

$$\alpha_s \alpha_t = (\beta^s + \beta^{-s})(\beta^t + \beta^{-t}) = \alpha_{s-t} + \alpha_{s+t}.$$

Using above relation and lemma 2, we are ready to give the following assertion.

Theorem 2. *Let $x = \sum_{i=1}^{m} x_i \alpha_i$ and $y = \sum_{i=1}^{m} y_i \alpha_i$ be elements in $GF(2^m)$. Then we have $xy = \sum_{i=1}^{m} (xy)_i \alpha_i$, where the kth coefficient $(xy)_k$ satisfies*

$$(xy)_k = \sum_{i=1}^{2m} y_i x_{i-k} = \sum_{i=1}^{2m+1} y_i x_{i-k}.$$

Proof. By the self duality of our basis,

$$\begin{aligned}
(xy)_k &= Tr(\alpha_k xy) \\
&= Tr(\alpha_k x \sum_{i=1}^{m} y_i \alpha_i) = \sum_{i=1}^{m} y_i Tr(\alpha_k \alpha_i x) \\
&= \sum_{i=1}^{m} y_i Tr(\alpha_{i-k} x + \alpha_{i+k} x) \\
&= \sum_{i=1}^{m} y_i (Tr(\alpha_{i-k} x) + Tr(\alpha_{i+k} x)) \\
&= \sum_{i=1}^{m} y_i (x_{i-k} + x_{i+k}) = \sum_{i=1}^{m} y_i x_{i-k} + \sum_{i=1}^{m} y_i x_{i+k}.
\end{aligned}$$

On the other hand, the second summation of above expression can be written as

$$\begin{aligned}
\sum_{i=1}^{m} y_i x_{i+k} &= \sum_{i=1}^{m} y_{m+1-i} x_{m+1-i+k} \\
&= \sum_{i=1}^{m} y_{m+i} x_{m+i-k} \\
&= \sum_{i=m+1}^{2m} y_i x_{i-k},
\end{aligned}$$

where the first equality follows by rearranging the summands and the second equality follows from lemma 2. Therefore we get

$$(xy)_k = \sum_{i=1}^{m} y_i x_{i-k} + \sum_{i=1}^{m} y_i x_{i+k} = \sum_{i=1}^{2m} y_i x_{i-k}.$$

Since $y_{2m+1} = 0$ by lemma 2, $(xy)_k = \sum_{i=1}^{2m+1} y_i x_{i-k}$ is obvious from above result. □

4. Bit serial arrangement using type II optimal normal basis

Using theorem 2, we may express $(xy)_k$ as a matrix multiplication form of a row vector and a column vector,

$$\begin{aligned}
(xy)_k = (x_{1-k}, x_{2-k}, \cdots, x_{2m-k}, x_{2m+1-k}) \\
\times (y_1, y_2, \cdots, y_{2m}, y_{2m+1})^T,
\end{aligned}$$

where $(y_1, y_2, \cdots, y_{2m}, y_{2m+1})^T$ is a transposition of the row vector $(y_1, y_2, \cdots, y_{2m}, y_{2m+1})$. Then we have

$$\begin{aligned}
(xy)_{k+1} = (x_{-k}, x_{1-k}, \cdots, x_{2m-1-k}, x_{2m-k}) \\
\times (y_1, y_2, \cdots, y_{2m}, y_{2m+1})^T.
\end{aligned}$$

Since $x_{-k} = x_{2m+1-k}$ by lemma 2, we find that $(x_{-k}, x_{1-k}, \cdots, x_{2m-1-k}, x_{2m-k})$ is a right cyclic shift of $(x_{1-k}, x_{2-k}, \cdots, x_{2m-k}, x_{2m+1-k})$ by one position. From this observation, we may realize the multiplication algorithm in the shift register arrangement shown in Fig. 1. The shift register is initially loaded with $(x_0, x_1, \cdots, x_{2m})$ which is in fact $(0, x_1, \cdots, x_m, x_m, \cdots, x_1)$. After k clock cycles, we get $(xy)_k$, the kth coefficient of xy with respect to the basis $\{\alpha_1, \cdots, \alpha_m\}$.

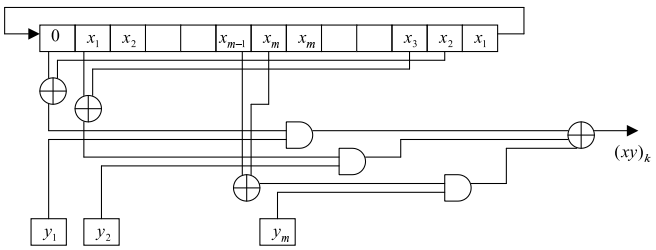

Figure 1. Bit serial multiplication using an optimal normal basis of type II.

Note that a bit parallel multiplier using an optimal normal basis of type II is discussed in [4], where it is suggested to find an efficient bit serial version of the construction in [4]. A hardware design of a bit serial multiplier using a type II ONB is proposed in a few papers [14,20,21]. The circuits in [14,20] and ours are basically the same design though the methods are slightly different. The construction in [21] is different from ours. The strong point of our approach is that our method can be applied to give a bit parallel systolic array which will be discussed in the next section and also a linear systolic array which follows directly from the bit parallel array via projection to vertical direction.

5. Bit parallel systolic architecture using an optimal normal basis of type II

A basis $\{\alpha_1, \alpha_2, \cdots, \alpha_m\}$ is used to derive an efficient bit serial multiplication of Berlekamp type in previous section. By rearranging the basis elements α_i, we may give a low latency and a low complexity bit parallel systolic multiplier. First, note that $\{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_m\}$ and $\{\alpha_1, \alpha_3, \alpha_5, \cdots, \alpha_{2m-1}\}$ are same sets. That is, if m is odd, then $\alpha_1, \alpha_3, \cdots, \alpha_{2m-1}$ are $\alpha_1, \alpha_3, \cdots, \alpha_m, \alpha_{m+2} = \alpha_{m-1}$, $\alpha_{m+4} = \alpha_{m-3}, \cdots, \alpha_{2m-1} = \alpha_2$. If m is even, $\alpha_1, \alpha_3, \cdots, \alpha_{m-1}, \alpha_{m+1} = \alpha_m, \alpha_{m+3} = \alpha_{m-2}, \cdots$, $\alpha_{2m-1} = \alpha_2$. Thus $\{\alpha_1, \alpha_3, \alpha_5, \cdots, \alpha_{2m-1}\}$ is also a basis for $GF(2^m)$ and a basis conversion from $\{\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_m\}$ to $\{\alpha_1, \alpha_3, \alpha_5, \cdots, \alpha_{2m-1}\}$ is as obvious as shown above. Let $x = \sum_{i=1}^{m} x_i \alpha_i$ be an element of $GF(2^m)$. Recall that we defined $x_s \in GF(2)$ as $x_s = Tr(\alpha_s x)$ for any integer s in definition 4. Let $y = \sum_{i=1}^{m} y_i \alpha_i$ be another element in $GF(2^m)$. For each integer j, we define row vectors X_j and Y_j as

$$X_j = (x_j, x_{j+1}, \cdots, x_{j+2m}),$$

and

$$Y_j = (y_j, y_{j+1}, \cdots, y_{j+2m}).$$

Note that for a nonnegative integer s, X_{j+s} and Y_{j+s} are left cyclic shifts of X_j and Y_j by s positions, respectively. Also note that X_{j-s} and Y_{j-s} are right cyclic shifts of X_j and Y_j by s positions. Moreover for any j and k, by using theorem 2, we get the following expression.

$$\begin{aligned}
&(xy)_k \\
&= \sum_{i=1}^{2m+1} y_i x_{i-k} = X_{1-k} Y_1^T \\
&= (x_{1-k}, x_{2-k}, \cdots, x_{2m+1-k})(y_1, y_2, \cdots, y_{2m+1})^T \\
&= (x_{j-k}, x_{j+1-k}, \cdots, x_{j+2m-k})(y_j, y_{j+1}, \cdots, y_{j+2m})^T \\
&= X_{j-k} Y_j^T.
\end{aligned}$$

Theorem 3. *Let $x = \sum_{i=1}^{m} x_i \alpha_i$ and $y = \sum_{i=1}^{m} y_i \alpha_i$ be elements of $GF(2^m)$. Then we have*

$$(xy)_{2k-1} = \sum_{i=1}^{m} (y_{i+k-1} x_{i-k} + y_{i-k} x_{i+k-1}) + y_{m+1-k} x_{m+1-k}.$$

Proof. By the remark just before the statement of this theorem,

$$\begin{aligned}
&(xy)_{2k-1} \\
&= X_{k-(2k-1)} Y_k^T = X_{1-k} Y_k^T \\
&= (x_{1-k}, x_{2-k}, \cdots, x_{2m+1-k})(y_k, y_{k+1}, \cdots, y_{k+2m})^T \\
&= \sum_{i=1}^{2m+1} y_{i+k-1} x_{i-k} \\
&= \sum_{i=1}^{m} y_{i+k-1} x_{i-k} + y_{m+k} x_{m+1-k} + \sum_{i=m+2}^{2m+1} y_{i+k-1} x_{i-k} \\
&= \sum_{i=1}^{m} y_{i+k-1} x_{i-k} + y_{m+1-k} x_{m+1-k} + \sum_{i=m+2}^{2m+1} y_{i+k-1} x_{i-k}.
\end{aligned}$$

On the other hand, the second summation of above expression can be written as

$$\begin{aligned}
\sum_{i=m+2}^{2m+1} y_{i+k-1} x_{i-k} &= \sum_{i=1}^{m} y_{2m+2-i+k-1} x_{2m+2-i-k} \\
&= \sum_{i=1}^{m} y_{i-k} x_{i+k-1},
\end{aligned}$$

where the first equality follows by rearranging the order of summation and the second equality follows from lemma 2. Therefore we have

$$\begin{aligned}
(xy)_{2k-1} = &\sum_{i=1}^{m} y_{i+k-1} x_{i-k} + y_{m+1-k} x_{m+1-k} \\
&+ \sum_{i=1}^{m} y_{i-k} x_{i+k-1},
\end{aligned}$$

which is the desired result. □

Now for each $(xy)_{2k-1}$, we define a column vector

$$\mathcal{W}_k = (w_{1k}, w_{2k}, \cdots, w_{mk}, w_{(m+1)k})^T,$$

where

$$\begin{aligned}
w_{ik} &= y_{i+k-1} x_{i-k} + y_{i-k} x_{i+k-1}, \ if \ 1 \le i \le m \\
w_{(m+1)k} &= y_{m+1-k} x_{m+1-k}, \qquad if \ i = m+1.
\end{aligned}$$

Then the sum of all entries of the column vector $\mathcal{W}_k$ is exactly $(xy)_{2k-1}$ and $\mathcal{W}_k$ appears as a kth column vector of the $m+1$ by m matrix $\mathcal{W} = (w_{ik})$ where

$$\mathcal{W} = \begin{pmatrix}
w_{11} & w_{12} & w_{13} & \cdot & \cdot & \cdot & w_{1m} \\
w_{21} & w_{22} & w_{23} & \cdot & \cdot & \cdot & w_{2m} \\
w_{31} & w_{32} & w_{33} & \cdot & \cdot & \cdot & w_{3m} \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
w_{m1} & w_{m2} & w_{m3} & \cdot & \cdot & \cdot & w_{mm} \\
w_{(m+1)1} & w_{(m+1)2} & w_{(m+1)3} & \cdot & \cdot & \cdot & w_{(m+1)m}
\end{pmatrix}$$

For each $1 \le i, k \le m$, using the relation

$$w_{ik} = y_{i+k-1} x_{i-k} + y_{i-k} x_{i+k-1},$$

we have

$$w_{(i-1)(k-1)} = y_{i+k-3}x_{i-k} + y_{i-k}x_{i+k-3}.$$

That is, the signals x_{i-k} and y_{i-k} in the expression of w_{ik} come from the signals in the expression of $w_{(i-1)(k-1)}$. Also since

$$w_{(i-1)(k+1)} = y_{i+k-1}x_{i-k-2} + y_{i-k-2}x_{i+k-1},$$

we deduce that the signals x_{i+k-1} and y_{i+k-1} in the expression of w_{ik} come from the signals in the expression of $w_{(i-1)(k+1)}$. Moreover the signals in the last row come from the signals in the mth row. That is, $w_{(m+1)1} = y_m x_m$ comes from the signals y_m and x_m in the expression $w_{m1} = y_m x_{m-1} + y_{m-1}x_m$. And for each $2 \leq k \leq m$, $w_{(m+1)k} = y_{m+1-k}x_{m+1-k}$ comes from the signals y_{m+1-k} and x_{m+1-k} in the expression $w_{m(k-1)} = y_{m+k-2}x_{m+1-k} + y_{m+1-k}x_{m+k-2}$. From this observation, we may construct a bit parallel systolic multiplier with respect to the basis $\{\alpha_1, \alpha_3, \cdots, \alpha_{2m-1}\}$. The circuit of basic cell is explained in Fig. 2, where $\bullet$ is one bit latch (flip-flop). An output of the vertical line produces partial sum of the product.

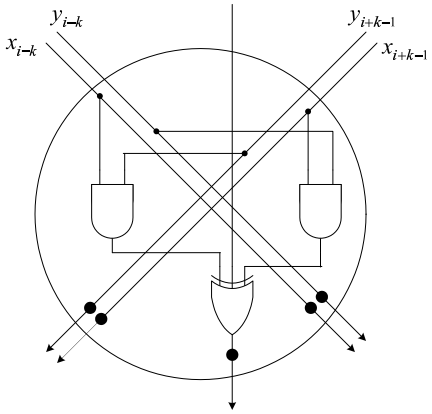

Figure 2. The circuit of (i, k) basic cell.

For convenience, assume $m = 5$ where the existence of type II ONB is well known. Then the matrix $\mathcal{W}$ is as follows;

$$\mathcal{W} = \begin{pmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} \\ y_5x_5 & y_4x_4 & y_3x_3 & y_2x_2 & y_1x_1 \end{pmatrix}$$

where $w_{ik} = y_{i+k-1}x_{i-k} + y_{i-k}x_{i+k-1}$ for $1 \leq i, k \leq m$. Letting $z = \sum_{i=1}^{m} z_i\alpha_i$ be another element in $GF(2^m)$, we may realize the product sum operation $u = xy + z$ in a bit parallel systolic arrangement shown in Fig. 3. Note that $x_0 = 0 = y_0$ in the arrangement and the output at the kth column is u_{2k-1}. We compare our multiplier with other bit parallel systolic multipliers in Table 1. None of the multipliers in the table broadcasts signals and all have unidirectional data flows except for the multipliers in [8] and [11], which have bidirectional data flows. Since the multiplier in [9] uses a dual basis, one needs a basis conversion process.

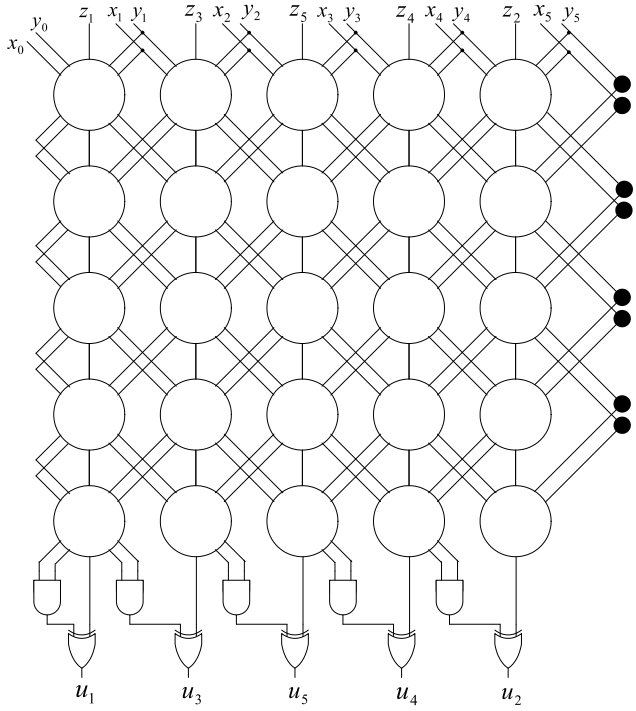

Figure 3. Systolic architecture for computing $u = xy + z$ in $GF(2^5)$.

Table 1. Comparison of our multiplier with other bit parallel systolic multipliers.

	Wang [7]	Yeh [8]	Fenn [9]	Wei [11]	Lee [10]	Fig. 3
basis	polynomial	polynomial	dual	polynomial	AOP	type II ONB
function	AB	AB+C	AB	AB^2+C	AB+C	AB+C
cell complexity						
AND	2	2	2	3	1	2
XOR	0	2	2	1	1	0
3XOR	1	0	0	1	0	1
Latch	7	7	7	10	3	5
number of cells	m^2	m^2	m^2	m^2	$(m+1)^2$	m^2
latency	$3m$	$3m$	$3m$	$3m$	m+1	m+1
critical path delay	D_A+D_{3X} +D_L	D_A+D_X +D_L	D_A+D_X +D_L	D_A+D_{3X} +D_L	D_A+D_X +D_L	D_A+D_{3X} +D_L

AND and XOR mean 2-input gates and 3XOR means a 3-input XOR gate. D_A, D_X, D_{3X} and D_L denote the delay time of an AND, a XOR, a 3XOR and a latch respectively. Note that the area complexity of a latch is higher than any other gate in the table.

Approximately, a latch takes 5 times more area than an AND gate and 2 times more area than a XOR gate.

It is shown in Table 1 that our multiplier has the best hardware complexity and latency except for the multiplier in [10]. The multiplier in [10] is applicable to a finite field $GF(2^m)$ when there is an all one polynomial (AOP) basis. A polynomial $1 + x + x^2 + \cdots + x^m \in GF(2)[x]$ is called an all one polynomial (AOP) of degree m. A finite field $GF(2^m)$ has an AOP basis whenever an AOP of degree m is irreducible over $GF(2)$. It is not difficult to show that an AOP basis exists in $GF(2^m)$ if and only if $m + 1 = p$ is a prime and 2 is a primitive root modulo p. A behavior of an AOP basis for moderately small values of m is well known. In fact, a table in [6, p. 100] shows that the number of $m \leq 2000$ for which an AOP basis exists is 118. For example, we have an AOP basis when $m = 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82, 100, 106, \cdots$. On the other hand, the same table says that the number of $m \leq 2000$ for which a type II optimal normal basis exists is 324. There is a type II optimal normal basis when $m = 2, 3, 5, 6, 9, 11, 14, 18, 23, 26, 29, 30, 33, 35, 39, 41, 50, 51, 53, 65, 69, 74, \cdots$. Therefore our multiplier in Fig. 3 is applicable to a broader class of m than the multiplier using an AOP basis. Moreover, since an AOP basis in [10] is a nonconventional basis having $m + 1$ basis elements in $GF(2^m)$, one needs extra logical operations to convert the basis to an ordinary basis. However, our multiplier has no such problem.

6. Conclusion

In this paper, we proposed a bit parallel systolic multiplier using an optimal normal basis (ONB) of type II. We showed, in Table 1, that our multiplier has a lower hardware complexity and a latency than corresponding multipliers in Table 1 except for the multiplier in [10], which uses an AOP basis. However, an AOP basis appears quite less frequently than an optimal normal basis of type II. A table in [6, p. 100] implies that a type II ONB is three times more likely to occur than an AOP basis. Therefore our bit parallel systolic multiplier provides an hardware efficient architecture for many finite fields, where an AOP basis does not exist and no other low complexity systolic architecture is known yet. Using the remark on gates areas in Table 1, we find that the hardware complexity of our multiplier is reduced by 25 percent from the ones in [7,8,9]. Also, our construction of Fig. 3 can be easily modified via projection to vertical direction to give a linear (one dimensional) systolic array . In this case, our linear systolic array has parallel in parallel out architecture and gives an output after m clock cycles.

References

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

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

[3] S. Gao, J. von zur Gathen and D. Panario, "Gauss periods and fast exponentiation in finite fields," *Lecture Notes in Computer Science*, vol. 911, pp. 311–322, 1995.

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

[5] A. Reyhani-Masoleh and M.A. Hasan, "A new construction of Massey-Omura parallel multiplier over $GF(2^m)$," *IEEE Trans. Computers*, vol. 51, pp. 511–520, 2002.

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

[7] C.L. Wang and J.L. Lin, "Systolic array implementation of multipliers for finite fields $GF(2^m)$," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 796–800, 1991.

[8] C.S. Yeh, I.S. Reed and T.K. Troung, "Systolic multipliers for finite fields $GF(2^m)$," *IEEE Trans. Computers*, vol. C-33, pp. 357–360, 1984.

[9] S.T.J. Fenn, M. Benaissa and D. Taylor, "Dual basis systolic multipliers for $GF(2^m)$," *IEE Proc. Comput. Digit. Tech.*, vol. 144, pp. 43–46, 1997.

[10] C.Y. Lee, E.H. Lu and J.Y. Lee, "Bit parallel systolic multipliers for $GF(2^m)$ fields defined by all one and equally spaced polynomials," *IEEE Trans. Computers*, vol. 50, pp. 385–393, 2001.

[11] C.W. Wei, "A systolic power sum circuit for $GF(2^m)$," *IEEE Trans. Computers*, vol. 43, pp. 226–229, 1994.

[12] S.K. Jain, L. Song and K.K. Parhi, "Efficient semisystolic architectures for finite field arithmetic," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 101–113, 1998.

[13] J.H. Guo and C.L. Wang, "Systolic array implementation of Euclid's algorithm for inversion and division in $GF(2^m)$," *IEEE Trans. Computers*, vol. 47, pp. 1161–1167, 1998.

[14] S. Kwon and H. Ryu "Efficient bit serial multiplication using optimal normal bases of type II in $GF(2^m)$," *Lecture Notes in Computer Science*, vol. 2433, pp. 300–308, 2002.

[15] C.Y. Lee, E.H. Lu and L.F. Sun, "Low complexity bit parallel systolic architecture for computing $AB^2 + C$ in a class of finite field $GF(2^m)$," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 519–523, 2001.

[16] W.C. Tsai, C.B Shung and S.J. Wang, "Two systolic architectures for modular multiplication," *IEEE Trans. VLSI Syst.*, vol. 8, pp. 103–107, 2000.

[17] T. Itoh and S. Tsujii, "Structure of parallel multipliers for a class of finite fields $GF(2^m)$," *Information and computation*, vol. 83, pp. 21–40, 1989.

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

[19] C. Paar, P. Fleischmann and P. Roelse, "Efficient multiplier archtectures for Galois fields $GF(2^{4n})$," *IEEE Trans. Computers*, vol. 47, pp. 162–170, 1998.

[20] H. Wu, M.A. Hasan, I.F. Blake and S. Gao, "Finite field multiplier using redundant representation," *IEEE Trans. Computers*, vol. 51, pp. 1306–1316, 2002.

[21] G.B. Agnew, R.C. Mullin, I. Onyszchuk and S.A. Vanstone, "An implementation for a fast public key cryptosystem," *J. Cryptology*, vol. 3, pp. 63–79, 1991.