Low Complexity Bit-Parallel Finite Field Arithmetic Using Polynomial Basis

Huapeng Wu

Dept of ECE, IIT, Chicago IL 60616
hpwu@ece.iit.edu

Abstract. Bit-parallel finite field multiplication in $\mathbb{F}_{2^m}$ using polynomial basis can be realized in two steps: polynomial multiplication and reduction modulo the irreducible polynomial. In this article, we prove that the modular polynomial reduction can be done with $(r-1)(m-1)$ bit additions, where r is the Hamming weight of the irreducible polynomial. We also show that a bit-parallel squaring operation using polynomial basis costs not more than $\left\lfloor \frac{m+k-1}{2} \right\rfloor$ bit operations if an irreducible trinomial of form $x^m + x^k + 1$ over $\mathbb{F}_2$ is used. Consequently, it is argued that to solve multiplicative inverse in $\mathbb{F}_{2^m}$ using polynomial basis can be as good as using normal basis.

1 Introduction

The increasing use of cryptographic techniques in computer and communication network systems has inspired many researchers to find ways to perform fast or bit-parallel algorithms and architectures over finite fields of characteristic two. Besides the discrete logarithm cryptosystems over $\mathbb{F}_{2^n}$, the elliptic curve cryptosystems, which utilize the group of points on an elliptic curve over a field, can also be realized using finite fields of characteristic two. These groups are generally used to take advantage of their efficiency over multiprecision arithmetic for large prime fields. The elliptic curve cryptosystems also have the advantage of their high cryptographic strength relative to the key size, and thus they are especially attractive in applications such as the financial industry, smart cards and wireless areas where power and bandwidth are limited.

There are generally three types of basis in finite field, namely, normal basis (NB), polynomial basis (PB) and dual basis (DB). Normal basis is often chosen in cryptographic application, since squaring operation is only a cyclic shift of the lines and thus inversion and exponentiation can be efficiently performed. Massey and Omura first found a regular architecture for normal basis multiplication [13], while the use of the optimal normal basis further reduces the complexity of multiplication [16]. Polynomial basis has long been used for finite field arithmetic. Polynomial basis multiplication based on the irreducible trinomial $x^m + x^k + 1$ with $1 \le k \le \left\lfloor \frac{m}{2} \right\rfloor$ are attractive because they require fewer bit operations for modular reduction. Mastrovito has proposed a bit-parallel multiplication algorithm and architecture when $f(x)$ is a trinomial [14]. He has shown that the number of both bit multiplications and bit additions needed is proportional to $2m^2$ when the degree of $f(x)$ is no greater than 15 and not equal to 8. The Karatsuba-Ofman (KOA) algorithm has also been considered for building bit-parallel finite field multipliers [1,17]. An implementation of KOA [17] has shown that bit-parallel multiplication architectures in certain composite fields can have significantly lower complexity, compared to those proposed in [14]. However, the time delay of the architectures using the KOA can be longer. Polynomial dual basis and normal dual basis have also been considered for efficient multiplication [21,19].

In this article, we prove that bit-parallel reduction modulo the irreducible polynomial costs only $(r-1)(m-1)$ when the irreducible polynomial $f(x)$ has the Hamming weight of r. Consequent work can be shown that a bit-parallel multiplier in $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ can be built with at most m^2 AND gates and m^2-1 XOR gates for any integer m when an irreducible trinomial of degree m exists. Polynomial basis bit-parallel squaring is also discussed. When the irreducible polynomial is chosen as a trinomial of form $x^m + x^k + 1$, then bit-parallel squaring operation can realized with no more than $\left\lfloor \frac{m+k-1}{2} \right\rfloor$ bit additions. Consequently, it is argued that to solve multiplicative inverse in $\mathbb{F}_{2^m}$ using polynomial basis can be as good as using normal basis.

The organization of this paper is as follows: Polynomial basis bit-parallel multiplication and squaring are discussed in Section 2 and Section 3, respectively. In Section 4, we argue that to solve multiplicative inverse using polynomial basis can be as good as using normal basis. Finally, a few concluding remarks are given in Section 5.

2 Bit-Parallel Polynomial Basis Multiplication in $\mathbb{F}_{2^m}$

Let the finite field $\mathbb{F}_{2^m}$ be generated with an irreducible r-term polynomial $f(x) = x^m + \sum_{i=0}^{r-2} x^{e_i}$, where $0 = e_0 < e_1 < \cdots < e_{r-2} < m$. Let $A(x) = \sum_{i=0}^{m-1} a_i x^i$ and $B(x) = \sum_{i=0}^{m-1} b_i x^i$ be any two elements in $\mathbb{F}_{2^m}$. Then, $C(x) = \sum_{i=0}^{m-1} c_i x^i \in \mathbb{F}_{2^m}$, the product of $A(x)$ and $B(x)$ can be obtained in two steps:

1. Polynomial multiplication:

$$S(x) = A(x)B(x), \tag{1}$$

where $S(x) = \sum_{k=0}^{2m-2} s_k x^k$, and s_k is given by

$$s_k \triangleq \sum_{\substack{i+j=k \\ 0 \leqslant i,j \leqslant m-1}} a_i b_j, \; k = 0, 1, 2, \ldots, 2m-2.$$

2. Reduction modulo the irreducible polynomial:

$$C(x) = S(x) \bmod f(x), \tag{2}$$

where $C(x) = \sum_{i=0}^{m-1} c_i x^i, \ c_i \in \mathbb{F}_2$.

Obviously, the complexities of polynomial basis bit-parallel multiplication in $\mathbb{F}_{2^m}$ are determined by these two parts. The complexity of the first step (polynomial multiplication) is independent of choice of the irreducible polynomial $f(x)$, and it has been shown to be $O(m \log m \log\log m)$ in bit operations [18]. We will show that the second step (modular reduction) requires at most $(r-1)(m-1)$ bit operations, where r is the Hamming weight of the irreducible polynomial $f(x)$.

2.1 Polynomial Multiplication

In the first step of PB multiplication (1), if $S(x)$ is computed from $A(x)$ and $B(x)$ by the conventional polynomial multiplication method, it requires m^2 multiplications and $(m-1)^2$ additions in the ground field and the time delay is $T_A + \lceil \log_2 m \rceil T_X$. However, there are some asymptotically faster methods for polynomial multiplication over finite fields [3], such as, the Fast Fourier Transform method [3,11] and the Karatsuba-Ofman algorithm [10,1,17]. They can result in asymptotically fewer bit operations at the expense of longer time delay and/or certain costly pre- and post-computations. Another technique for polynomial basis multiplication that can combine polynomial multiplication with modulo reduction into one single step is called the Montgomery method [15,12].

2.2 Reduction Modulo a Polynomial

For modular reduction $C(x) = S(x) \bmod f(x)$, where $\deg f = m, \deg S \leqslant 2m-2$ and $\deg C \leqslant m-1$, if the conventional polynomial division method is used, the complexity is $O(m^2)$ in ground field operations. Mastrovito [14] has found that if the irreducible polynomial is chosen properly for $m \leqslant 15, m \neq 8$, the complexity of modulo reduction can be greatly reduced by using some partial sums. Paar [17] has also discussed this issue for certain small values of m. However, their methods are based on computer based exhaustive search and available for only moderately small size fields. In the following, we will present a new algorithm that can perform modulo reduction in $(r-1)(m-1)$ ground field operations for any irreducible polynomial $f(x)$ with the Hamming weight r.

Theorem 1. If the Hamming weight of the irreducible polynomial $f(x)$ is r, then the modular polynomial reduction (2) can be done with $(r-1)(m-1)$ bit operations.

Proof: Define

$$\sum_{i=0}^{m+l} s_i x^i \ \bmod f(x) \triangleq \sum_{i=0}^{m-1} t_i^{(l)} x^i, \ l = -1, 0, 1, \ldots, m-2. \tag{3}$$

$t_i^{(l)}$'s have the initial values $t_i^{(-1)} = s_i$, and, we try to solve for the 'final' values $t_i^{(m-2)} = c_i$, $i = 0, 1, 2, \ldots, m-1$.

In the following, we shall prove by induction that the complexity of solving $t_i^{(m-2)}, i = 0, 1, 2, \ldots, m-1$, is $(r-1)(m-1)$ bit operations.

When $l = 0$, from (3) we have

$$\begin{aligned}\sum_{i=0}^{m-1} t_i^{(0)} x^i = \sum_{i=0}^{m} s_i x^i &= \sum_{i=0}^{m-1} s_i x^i + s_m x^m \\ &= \sum_{i=0}^{m-1} t_i^{(-1)} x^i + s_m[1 + x^{e_1} + \cdots + x^{e_{r-2}}]\end{aligned}$$

Clearly, $t_i^{(0)} = \begin{cases} t_i^{(-1)} + s_m, & \text{if } i = 0, e_1, e_2, \ldots, e_{r-2}, \\ t_i^{(-1)}, & \text{if } 1 \le i \le m-1, \text{ and } i \ne e_1, e_2, \ldots, e_{r-2}. \end{cases}$

It can be seen that $r-1$ bit additions are required for obtaining $t_i^{(0)}$ from $t_i^{(-1)}, i = 0, 1, \ldots, m-1$.

Assume when $0 \le l < l'$, $r-1$ bit-additions are required for obtaining $t_i^{(l)}$ from $t_i^{(l-1)}, i = 0, 1, \ldots, m-1$. Then, when $l = l'$, we have

$$\begin{aligned}\sum_{i=0}^{m-1} t_i^{(l')} s^i = \sum_{i=0}^{m+l'} s_i x^i &= \sum_{i=0}^{m+l'-1} s_i x^i + s_{m+l'} x^{m+l'} \\ &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + s_{m+l'} x^{l'} x^m \\ &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + s_{m+l'} x^{l'} [1 + x^{e_1} + \cdots + x^{e_{r-2}}]\end{aligned}$$

If $m > l' + e_{r-2}$, then

$$t_i^{(l')} = \begin{cases} t_i^{(l'-1)} + s_{m+l'}, & \text{if } i = l', l' + e_1, \ldots, l' + e_{r-2}, \\ t_i^{(l'-1)}, & \text{otherwise.} \end{cases}$$

Obviously, $t_i^{(l')}$ can be computed from $t_i^{(l'-1)}$ using $r-1$ bit additions. Now suppose that $l'+e_{r'-1} < m \leqslant l'+e_{r'}, r' \in \{1,\dots,r-2\}$, thus it follows

$$\begin{aligned}\sum_{i=0}^{m+l'} s_i x^i = \{\sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + s_{m+l'} x^{l'}[1+x^{e_1}+\cdots+x^{e_{r'-1}}]\}& \\ +s_{m+l'}x^{l'}[x^{e_{r'}}+x^{e_{r'+1}}+\cdots+x^{e_{r-2}}]& \\ = \sum_{i=0}^{m-1} t_i^{(l',0)} x^i + s_{m+l'}x^{l'}[x^{e_{r'}}+x^{e_{r'+1}}+\cdots+x^{e_{r-2}}]& \\ = \sum_{i=0}^{m-1} t_i^{(l',0)} x^i + s_{m+l'}x^{l'+e_{r'}}+\cdots+s_{m+l'}x^{l'+e_{r-2}}& \qquad (4)\end{aligned}$$

where $\sum_{i=0}^{m-1} t_i^{(l',0)} \triangleq \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + s_{m+l'}x^{l'}[1+x^{e_1}+\cdots+x^{e_{r'-1}}]$. Since we have

$$t_i^{(l',0)} = \begin{cases} t_i^{(l'-1)} + s_{m+l'}, & \text{if } i = l', l'+e_1,\dots,l'+e_{r'-1}, \\ t_i^{(l'-1)}, & \text{otherwise,}\end{cases}$$

it can be seen that r' bit additions are required to obtain $t_i^{(l',0)}$ from $t_i^{(l'-1)}, i = 0,1,\dots,m-1$.

In the following we shall prove that $t_i^{(l')}$, $i=0,1,\dots,m-1$, can be obtained from $t_i^{(l',0)}$ with $r-r'-1$ bit additions. Define

$$\sum_{i=0}^{m-1} t_i^{(l',1)} x^i \triangleq \sum_{i=0}^{m-1} t_i^{(l',0)} x^i + s_{m+l'}x^{l'+e_{r'}} \mod f(x),\ i=0,1,\dots,m-1. \quad (5)$$

Since $0 \leq l'+e_{r'}-m < l'$, we have

$$\sum_{i=0}^{m-1} t_i^{(l'+e_{r'}-m)} x^i = \sum_{i=0}^{m-1} t_i^{(l'+e_{r'}-m-1)} x^i + s_{l'+e_{r'}} x^{l'+e_{r'}} \mod f(x),\ i=0,1,\dots,m-1. \tag{6}$$

Since $t_i^{(l'+e_{r'}-m)}$ has been obtained from $t_i^{(l'+e_{r'}-m-1)}$ with $r-1$ bit additions as assumed, comparing (5) to (6), we can see that (5) and (6) can be combined together to save bit operations. That is, when $l = l'+e_{r'}-m$, instead of performing (6), we perform

$$\sum_{i=0}^{m-1} t_i^{(l'+e_{r'}-m-1)} x^i + (s_{l'+e_{r'}} + s_{m+l'}) x^{l'+e_{r'}} \mod f(x) = \sum_{i=0}^{m-1} t_i^{(l'+e_{r'}-m,\star)} x^i \tag{7}$$

with r bit additions, while (5) can be saved. In the sense of the count of bit operations, we may equivalently say that (5) requires one bit addition, while

(6) still needs $r-1$ bit operations. Similar arguments can be applied to the remaining $r-r'-2$ terms $s_{m+l'}x^{l'+e_j}, j = r'+1, r'+2, \ldots, r-2$ in (4). Thus for $l = l'$, $r-1$ bit additions are required for obtaining $t_i^{(l')}$ from $t_i^{(l'-1)}$, for $i = 0, 1, \ldots, m-1$. Therefore, to compute $t_i^{(l)}$ from $t_i^{(l-1)}, i = 0, 1, \ldots, m-1$, needs $r-1$ bit additions for any integer l. We conclude that computing $c_i = t_i^{(m-2)}$ from $s_i, i = 0, 1, \ldots, 2m-2$ requires $(m-1)(r-1)$ bit additions. □

Theorem 1 can be easily extended to $\mathbb{F}_{q^m}$ as it is stated in Theorem 2. A proof for Theorem 2 is analogous to that of Theorem 1.

Theorem 2. If the monic irreducible polynomial $f(x) \in \mathbb{F}_q[x]$ of degree m has the Hamming weight of r, then the modular polynomial reduction in polynomial basis multiplication can be done with $(r-1)(m-1)$ multiplications and $(r-1)(m-1)$ additions in $\mathbb{F}_q$.

If the conventional method for polynomial multiplication is used, some results of consequent work on finite field multiplier architecture are shown as follows:

If the finite field $\mathbb{F}_{2^m}$ is generated with an irreducible trinomial $f(x) = 1 + x^k + x^m$, $1 \leqslant k \leqslant \left\lfloor \frac{m}{2} \right\rfloor$, then a bit-parallel polynomial basis multiplier can be constructed with $\mathbf{C}_{SA} = m^2$,

(i) $\mathbf{C}_{SX} = m^2 - 1$ and $\mathbf{C}_T = T_A + (\lceil \log_2 m \rceil + 1) T_X$ for $k = 1$;
(ii) $\mathbf{C}_{SX} = m^2 - 1$ and $\mathbf{C}_T = T_A + (\lceil \log_2 m \rceil + 2) T_X$ for $1 < k < \frac{m}{2}$;
(iii) $\mathbf{C}_{SX} = m^2 - \frac{m}{2}$ and $\mathbf{C}_T = T_A + (\lceil \log_2 m \rceil + 1) T_X$ for $k = \frac{m}{2}$.

3 Polynomial Basis Bit-Parallel Squaring

3.1 Complexity of Polynomial Basis Bit-Parallel Squaring in $\mathbb{F}_{2^m}$

Let $f(x)$ be the irreducible polynomial over $\mathbb{F}_2$ generating the field $\mathbb{F}_{2^m}$. Let $A(x) = \sum_{i=0}^{m-1} a_i x^i$ be the polynomial representation of an arbitrary element of $\mathbb{F}_{2^m}$. The squaring operation of $A(x)$ is $C(x) \triangleq \sum_{i=0}^{m-1} c_i x^i = A^2(x) \bmod f(x) = a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_{\lceil \frac{m}{2} \rceil} x^{2\lceil \frac{m}{2} \rceil} + \ldots + a_{m-1} x^{2m-2} \bmod f(x)$. It can be seen that squaring in $\mathbb{F}_{2^m}$ is actually a case of polynomial modular reduction that has been discussed in the last section, where the degree of each squared terms in $A^2(x)$ is an even integer between 0 and $2m-2$. From the discussion in the last section, the following corollary is obvious.

Corollary 1. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible r-term polynomial $f(x)$ of degree m. Then squaring a field element in parallel can be performed with at most $(r-1)(m-1)$ addition operations in $\mathbb{F}_2$.

When $f(x)$ is an irreducible trinomial, however, both the size complexity and time complexity can be further reduced.

Theorem 3. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible trinomial $f(x) = x^m + x^k + 1$, where m is even and k odd. Then squaring a field element in a bit-parallel fashion can be performed with at most $\frac{m+k-1}{2}$ bit operations. □

Proof: Let

$$A^2(x) = \sum_{i=0}^{m-1} a_i x^{2i} = \sum_{i=0}^{2m-2} a'_i x^i,$$

where $a'_i \overset{\triangle}{=} a_{\frac{i}{2}}$ if i even, and 0 if i odd. Define

$$\sum_{i=0}^{m+2l} a'_i x^i \bmod f(x) \overset{\triangle}{=} \sum_{i=0}^{m-1} t_i^{(l)} x^i,\ l = -1, 0, 1, \dots, \frac{m}{2} - 1.$$

The terms $t_i^{(l)}$'s have their initial values $t_i^{(-1)} = a'_i$, and we try to solve the final values $t_i^{(\frac{m}{2}-1)} = c_i, i = 0, 1, \dots, m-1$.

When $l = 0$,

$$\sum_{i=0}^{m-1} t_i^{(0)} x^i = \sum_{i=0}^{m} a'_i x^i = \sum_{i=0}^{m-1} a'_i x^i + a'_m x^m = \sum_{i=0}^{m-1} a'_i x^i + a'_m (1 + x^k).$$

$$\therefore\ t_i^{(0)} = \begin{cases} a'_i + a'_m, & i = 0; \\ a'_m, & i = k. \\ a'_i, & 0 < i \leqslant m-1, i \text{ even}; \\ 0, & i \text{ odd},\ i \neq k; \end{cases}$$

Clearly, one bit addition is needed to compute $t_i^{(0)}$ from $t_i^{(-1)}$, $i = 0, 1, \dots, m-1$. For $l > 0$, we have

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m+2l} a'_i x^i \\ &= \sum_{i=0}^{m+2(l-1)} a'_i x^i + a'_{m+2l} x^{m+2l} \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l} x^{2l} (1 + x^k) \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l} x^{2l} + a'_{m+2l} x^{k+2l}. \end{aligned}$$

If $k+2l<m$ or $l<\frac{m-k}{2}$, then

$$t_i^{(l)}=\begin{cases} t_i^{(l-1)}, & 0\leqslant i\leqslant m-1, i\neq 2l, i \text{ even; or } i=k,k+2,\dots,k+2(l-1);\\ 0, & i \text{ odd}, i\neq k,k+2,\dots,k+2l;\\ t_i^{(l-1)}+a'_{m+2l}, & i=2l;\\ a'_{m+2l}, & i=k+2l. \end{cases} \tag{8}$$

It can be seen that only one bit addition is required to compute $t_i^{(l)}$ from $t_i^{(l'-1)}$ for $0<l<\frac{m-k}{2}$ and $i=0,1,\dots,m-1$.

In the following we proceed with induction. When $l=\frac{m-k+1}{2}$ ($\because m-k$ is odd) and $l<\frac{m}{2}$, we have

$$\begin{aligned}\sum_{i=0}^{m-1} t_i^{(l)}x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)}x^i + a'_{m+2l}x^{m+2l}\\ &= \sum_{i=0}^{m-1} t_i^{(l-1)}x^i + a'_{m+2l}x^{2l} + a'_{m+2l}x^{k+2l}\\ &= \sum_{i=0}^{m-1} t_i^{(l-1)}x^i + a'_{m+2l}x^{2l} + a'_{m+2l}x + a'_{m+2l}x^{k+1}.\end{aligned}$$

Then,

$$t_i^{(l)}=\begin{cases} t_i^{(l-1)}+a'_{m+2l}, & i=2l \text{ or } i=k+1;\\ a'_{m+2l}, & i=1;\\ t_i^{(l-1)}, & i \text{ even}, i\neq 2l, k+1;\\ & \text{or } i=k,k+2,\dots,k+2(l-1);\\ 0, & i \text{ odd}, i\neq k,k+2,\dots,k+2(l-1) \text{ and } i\neq 1. \end{cases} \tag{9}$$

Obviously, two bit additions are required to compute $t_i^{(l)}$ from $t_i^{(l-1)}$, $i=0,1,\dots,m-1$.

Assume that for $\frac{m-k+1}{2} \leqslant l < l'$, (9) holds, then for $l = l' < \frac{m}{2}$, we have

$$\begin{aligned}\sum_{i=0}^{m-1} t_i^{(l')} x^i &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + a_{m+2l'} x^{m+2l'} \\ &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + a'_{m+2l'} x^{2l'} [1 + x^k] \\ &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + a'_{m+2l'} x^{2l'} + + a'_{m+2l'} x^{2l'+k} \\ &= \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + a'_{m+2l'} x^{2l'} + a'_{m+2l'} x^{k+2l'-m} + a'_{m+2l'} x^{2k+2l'-m} \\ &= \sum_{i=0}^{m-1} t_i^{(l',0)} x^i + a'_{m+2l'} x^{2k+2l'-m}, \end{aligned} \tag{10}$$

where $t_i^{(l',0)}$ is defined by $\sum_{i=0}^{m-1} t_i^{(l',0)} x^i \triangleq \sum_{i=0}^{m-1} t_i^{(l'-1)} x^i + a'_{m+2l'} x^{2l'} + a'_{m+2l'} x^{k+2l'-m}$. Since $2l' < m$, and $k + 2l' - m$ is odd and less than k,

$$t_i^{(l',0)} = \begin{cases} t_i^{(l'-1)} + a'_{m+2l'}, & i = 2l'; \\ a'_{m+2l'}, & i = k + 2l' - m; \\ t_i^{(l'-1)}, & 0 \leqslant i \leqslant m-1, i \neq 2l', i \text{ even; or } i = k, k+2, \ldots, k+2(l'-1); \\ & \text{or } i = 1, 3, \ldots, k+2(l'-1)-m; \\ 0, & i \text{ odd}, i \neq k, k+2, \ldots, k+2(l'-1), i \neq 1, 3, \ldots, \\ & k+2(l'-1)-m. \end{cases} \tag{11}$$

Thus it requires one bit addition to obtain $t_i^{(l',0)}$ from $t_i^{(l'-1)}$, $i = 0, 1, \ldots, m-1$.

When $2k + 2l' - m < m$, we have $t_i^{(l')} = t_i^{(l',0)}$ if $i \neq 2k + 2l' - m$, otherwise $t_i^{(l')} = t_i^{(l',0)} + a'_{m+2l'}$. In this case it is therefore two bit operations are required to compute $t_i^{(l')}$ from $t_i^{(l'-1)}$ for $i = 0, \ldots, m-1$.

When $2k + 2l' - m \geqslant m$, consider

$$\sum_{i=0}^{m-1} t_i^{(k+l'-m)} = \sum_{i=0}^{m-1} t_i^{(k+l'-m-1)} + a'_{2k+2l'-m} x^{2k+2l'-m}. \tag{12}$$

It can be seen that the last terms of the right hand side of (10) and (12) are the same except for the coefficient. At the step $l = k + l' - m$, instead of performing (12), if we perform

$$\sum_{i=0}^{m-1} t_i^{(k+l'-m,\star)} = \sum_{i=0}^{m-1} t_i^{(k+l'-m-1)} + (a'_{2k+2l'-m} + a'_{m+2l'}) x^{2k+2l'-m} \tag{13}$$

at the cost of one more bit operation, then at step $l = l'$, the term $t_i^{(l')}$ can be computed from $t_i^{(l'-1)}, i = 0, \ldots, m-1$ with only one bit operation. Equivalently, we might say that at step $l = l'$, term $t_i^{(l')}$ can be computed from $t_i^{(l'-1)}$, $i = 0, 1, \ldots, m-1$, at the cost of two bit operations. Thus for $\frac{m-k+1}{2} \leqslant l < \frac{m}{2} - 1$, it requires two bit additions at each step.

We conclude that the total cost for computing $c_i = t_i^{(\frac{m}{2}-1)}$ from $t_i^{(-1)} = a'_i, i = 0, 1, \ldots, m-1$ is $\frac{m-k+1}{2} + 2(\frac{m}{2} - 1 - \frac{m-k-1}{2}) = \frac{m+k-1}{2}$ bit operations. □

Theorem 4. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible trinomial $f(x) = x^m + x^k + 1$, where m is odd and k even. Then bit-parallel squaring in $\mathbb{F}_{2^m}$ can be performed with at most $\frac{m+k-1}{2}$ bit additions.

Theorem 5. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible trinomial $f(x) = x^m + x^k + 1$, where both m and k are odd. Then bit-parallel squaring in $\mathbb{F}_{2^m}$ can be performed with at most $\frac{m-1}{2}$ bit additions. □

Proofs of Theorems 4 and 5 are similar to that of Theorem 3.

Some results of consequent work on implementation of bit-parallel squaring operation done by us is given below.

Theorem 6. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible trinomial $f(x) = x^m + x^k + 1$, where $m + k$ is odd. Then a bit-parallel squarer can be implemented with at most $\frac{m+k-1}{2}$ XOR gates. For $k = 1$ or 2, the incurred time delay is T_X, and for $2 < k \leqslant \frac{m}{2}$, it is $2T_X$.

Theorem 7. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible trinomial $f(x) = x^m + x^k + 1$, where both m and k are odd. Then a bit-parallel squarer can be implemented with at most $\frac{m-1}{2}$ XOR gates. The incurred time delay is T_X if $k = 1$, and $2T_X$ if $2 < k < \frac{m}{2}$.

4 Inversion

Inversion operation is required in elliptic curve cryptosystem when computing point multiples. This operation is usually performed with two methods. One is the extended Euclidean algorithm and the other is to exponentiate the element using the following identity

$$x^{-1} = x^{2^m - 2}. \tag{14}$$

The extended Euclid's algorithm usually requires the field element having a polynomial basis representation [5], where the most used operations are field addition, shifting and loading. Efficient algorithms have been proposed for the second method, for example, [8] and [2]. Both algorithms use about $\frac{3}{2} \log(m-1)$ field multiplications on average[1] and $m-1$ squaring operations. It has been generally accepted that normal basis should be used for this method since squaring in normal basis is only a cycle shift of the coefficients [8,2]. However, with the results presented in this paper, we argue that to solve inverse using (14) polynomial basis representation can be as efficient as normal basis representation.

It has been shown in [7] that a normal basis multiplication can be performed with $2m^2-1$ bit operations when a type I optimal normal basis is used. If a type II optimal normal basis or a non-optimal normal basis is used, it takes at least $3m^2$ bit operations to accomplish a field multiplication [16,6]. On the other hand, If the polynomial basis generated with an irreducible trinomial is used, a bit-parallel multiplication needs at most $2m^2-1$ bit operations while a bit-parallel squaring costs not greater than m bit operations. In this case, the complexity to solve the inverse using (14) in terms of bit operations with different bases is given in the following table.

Table 1. The complexity (in bit operations) of inversion using the algorithms in [8].

	Multiplications	Squarings
Type I optimal NB	$(2m^2-1)\,(\log_2(m-1)+H(m-1)-1)$	–
Type II optimal NB	$\geq 3m^2\,(\log_2(m-1)+H(m-1)-1)$	–
Trinomial-generated PB	$(2m^2-1)\,(\log_2(m-1)+H(m-1)-1)$	$< m(m-1)$

It can be seen from the table that the complexity using polynomial basis generated with an irreducible trinomial is comparable to that using type I optimal normal basis, while much smaller than that using type II optimal normal basis or non-optimal basis. Moreover, given finite field $\mathbb{F}_{2^m}$ there is more chance that an irreducible trinomial exists than that a type I optimal normal basis does. In fact, for $2 \leqslant m \leqslant 1000$, there is an irrducible trinomial in $\mathbb{F}_{2^m}$ for 545 values of m while there exists a type I optimal normal basis for only 67 values of m [4,9].

5 Concluding Remarks

In this article, we have shown that a bit-parallel multiplication operation in $\mathbb{F}_{2^m}$ using polynomial basis can be performed in $2m^2+(r-3)m-(r-2)$ bit operations. We have also proven that a bit-parallel squaring operation using polynomial basis costs not more than $\left\lfloor \frac{m+k-1}{2} \right\rfloor$ bit operations if an irreducible trinomial x^m+x^k+1 over $\mathbb{F}_2$ is used. Consequently, it is argued that to solve multiplicative inverse in $\mathbb{F}_{2^m}$ using polynomial basis can be as good as using normal basis.

Consequent work on implementation has shown that the resultant bit-parallel multiplier and bit-parallel squarer also have low time delay.

[1] Assume that the Hamming weight of $m-1$ is $\frac{1}{2}\log(m-1)$ on average.

Acknowledgements

This work was done when the author worked for his Ph.D degree with the Dept of ECE, University of Waterloo. The author thanks Professor Hasan and Professor Blake for their encouragement and valuable comments.

References

1. Afanasyev, V.B.: On the complexity of finite field arithmetic. Proc 5th Joint Soviet-Swedish Intern. Workshop on IT, Moscow, USSR, 1991, 9-12
2. Agnew, G.B., Beth, R., Mullin, R.C., Vanstone, S.A.: Arithmetic operations in GF(2^m). J. Cryptology **6** (1993) 3-13
3. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley Publ. Co., Reading, MA, 1974
4. Blake, I.F., Gao, S., Lambert, R.: Constructive Problems for Irreducible Polynomials over Finite Fields. Canadian Workshop on IT, Springer-Verlag, 1993
5. Brunner, H., Curiger, A., Hofstetter, M.: On computing multiplicative inverse in GF(2^m). IEEE Trans. Comput. **42** (1993) 1010-1015
6. Gao, S., Vanstone, S.A.: On orders of optimal normal basis generators. Math. Comp. **64** (1995) 1227-1233
7. Hasan, M.A., Wang, M., Bhargava, V.K.: A modified Massey-Omura parallel multiplier for a class of finite fields. IEEE Trans. Comput. **42** (1993) 1278-1280
8. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverse in GF(2^m) using normal bases. Inform. and Comput. **78** (1988) 171-177
9. Itoh, T., Tsujii, S.: Structure of parallel multipliers for a class of fields GF(2^m). Inform. and Comput. **83** (1989) 21-40
10. Karatsuba, A., Ofman, Y.: Multiplication of multidigit numbers on automata. Sov. Phys.-Dokl. (English translation), **7** (1963) 595-596
11. Knuth, D.E.: The Art of Computer Programming: Seminumerical Algorithms. Addison-Wesley Publishing Company, Reading, MA, 1981
12. Koç, Ç. K., Acar, T.: Montgomery multiplication in GF(2^k). Designs, Codes and Cryptography, **14** (1998) 57-69
13. Massey, J.L., Omura, J.K.: Computational method and apparatus for finite field arithmetic. U.S. Patent No.4587627, 1984.
14. Mastrovito, E.D.: VLSI Architectures for Computations in Galois Fields. Ph.D Thesis, Linköping University, 1991, Linköping, Sweden
15. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. **44** (1985) 519-521
16. Mullin, R., Onyszchuk, I., Vanstone, S.A., Wilson, R.: Optimal normal bases in GF(p^n). Disc. Appl. Math. **22** (1988) 149-161
17. Paar, C.: Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. Ph.D Thesis, VDI-Verlag, Düsseldorf, 1994
18. Schönhage, A.: Schnelle Multiplikation von Polynomen uber Korpern der Charakteristik 2. Acta Inf. **7** (1977) 395-398
19. Wang, C.C.: An algorithm to design finite field multipliers using a self-dual normal basis. IEEE Trans. Comput. **38** (1989) 1457-1459
20. Wu, H.: Efficient Computations in Finite Fields with Cryptographic Significance. Ph.D Thesis, University of Waterloo, Waterloo, Canada, 1998
21. Wu, H., Hasan, M.A., Blake, I.F.: Low complexity weakly dual basis bit-parallel multiplier over finite fields. IEEE Trans. Comput. **47** (1998) 1223-1234