RESEARCH OF FAST MODULAR MULTIPLIER FOR A CLASS OF FINITE FIELDS[1]

Jin Yi'er Shen Haibin Chen Huafeng Yan Xiaolang
(*Institute of VLSI design, Zhejiang University, Hangzhou 310027, China*)

Abstract A new structure of bit-parallel Polynomial Basis (PB) multiplier is proposed, which is based on a fast modular reduction method. The method was recommended by the National Institute of Standards and Technology (NIST). It takes advantage of the characteristics of irreducible polynomial, *i.e.*, the degree of the second item of irreducible polynomial is far less than the degree of the polynomial in the finite fields GF(2^m). Deductions are made for a class of finite field in which trinomials are chosen as irreducible polynomials. Let the trinomial be $x^m + x^k + 1$, where $1 \le k \le \lceil m/2 \rceil$. The proposed structure has shorter critical path than the best known one up to date, while the space requirement keeps the same. The structure is practical, especially in real time cryptographic applications.

Key words Fast modular multiplication; Bit-parallel multiplier; Trinomials

CLC index TN918.4; TN402

DOI 10.1007/s11767-006-0257-4

I. Introduction

Finite field operations have been applied widely in cryptography, such as Elliptic Curve Cryptography (ECC). The most time-consuming step of ECC operation is the point multiplication. To speed the operation, some algorithms have been proposed. Montgomery scalar method is one of them, which divides point multiplication into modular multiplication, modular square and modular addition in GE(2^m)[1]. Among them, modular multiplication is found to be the one that has the most significant influence on computation speed in GF(2^m) for ECC.

Many multipliers over GF(2) have been proposed[2–10]. They can be categorized into three classes according to different bases: Polynomial Basis (PB) multiplier[6,10], Normal Basis (NB) multiplier[9] and Dual Basis (DB) (or sometimes weakly dual basis) multiplier[8]. An efficient modular multiplication algorithm was introduced in Ref.[11], which fits for the fields generated by polynomials such as $f(x) = x^m + x^k + 1$ with $\deg[T(x)] << m$. Based on Ref.[11], let $T(x) = x^k + 1$ a modified modular multiplier in GF(2^m) is presented in this paper. The results show that the proposed multiplier can achieve faster computation speed than any other multipliers using polynomial basis for fields generated by trinomials $f(x) = x^m + x^k + 1$, where $k \le \lceil m/2 \rceil$, within the same area cost.

II. An Efficient Modular Multiplication

There are many algorithms implementing modular multiplication. Besides the widely used methods such as Montgomery algorithm, some fast approaches are developed for certain types of irreducible polynomials. The National Institute of Standards and Technology (NIST) recommended a fast modular multiplication which is as follows[11]:

Algorithm 1 Fast modular multiplication over GF(2^m)

Input
$$\begin{cases} A(x) = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0); \\ B(x) = (b_{m-1}, b_{m-2}, \cdots, b_1, b_0); \\ f(x) = x^m + T(x), \text{ where } \deg[T(x)] = k; \end{cases}$$

Output $C(x) = A(x)B(x) \bmod f(x)$;
$C'(x) = A(x)B(x) = C_h(x)x^m + C_l(x)$;
while $(C_h(x) \neq 0)\{$
$C'(x) = C_h(x)T(x) + C_l(x)$

[1] Manuscript received date: Novembet 24, 2006; revised date: March 29, 2007.
Supported by the Hi-Tech Research and Development Program of China (863) (No.2003AA1Z1060).
Communication author: Shen Haibin, born in 1967, male, Ph.D., associate professor. The Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China.
Email: shb@vlsi.zju.edu.cn.

$$= C_h'(x)x^m + C_l'(x);$$
$$C_h(x) = C_h'(x),\ C_l(x) = C_l'(x)\}$$
return $C(x) = C'(x)$

The iteration time L can be computed under the fact that the degree of most significant bit changed from n to $n-m+k$ after an iteration, *i.e.*, the reduced degree is $\Delta P = m-k$. To get $C_h = 0$, L should fulfill the inequalities:

$$n - L\Delta P \le m-1 \tag{1}$$

$$n - (L-1)\Delta P \ge m \tag{2}$$

According to Eq.(1) and Eq.(2), the integer L's range is acquired:

$$(n-m+1)/(m-k) \le L \le (n-k)/(m-k) \tag{3}$$

It can be proved easily that only one integer is contained among the range of $[(n-m+1)/(m-k), (n-k)/(m-k)]$. The integer was

$$L = \left\lfloor \frac{n-m}{m-k} \right\rfloor + 1 \tag{4}$$

III. Modular Multiplier in GF(2^m)

According to Section II, the fast modular algorithm consisted of two computation parts: multiplication and modular reduction. Algorithm 1 is decomposed into partial multiplication and integrated modular reduction for the following deduction work. In the rest of the paper, we concentrate on the irreducible trinomials $f(x) = x^m + x^k + 1$.

1. Partial multiplication in Algorithm 1

The polynomial basis representations of $A(x)$ and $B(x)$ can be written as:

$$A(x) = \sum_{i=0}^{m-1} a_i x^i = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1 x + a_0$$
$$B(x) = \sum_{i=0}^{m-1} b_i x^i = b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1 x + b_0$$

Thus, $C'(x) = A(x)B(x)$ can be achieved as follows:

$$C'(x) = A(x)B(x) = \left(\sum_{i=0}^{m-1} a_i x^i\right)\left(\sum_{i=0}^{m-1} b_i x^i\right) = \sum_{i=0}^{2m-2} c_i'(x)\, x^i \tag{5}$$

where c_i' is in the form of

$$c_i'(x) = \begin{cases} \sum_{j=0}^{i} a_j b_{i-j},\ 0 \le i \le m-1 \\ \sum_{j=i-m+1}^{m-1} a_j b_{i-j},\ m \le i \le 2m-2 \end{cases} \tag{6}$$

For this partial multiplication, m^2bit multiplications and $(m-1)^2$bit additions are required, which is shown in Tab.1. Assume that AND gate is two-input with time delay T_a, XOR gate is two-input with time delay T_x. From Tab.1, the maximal time delay of AND gate network is T_a.

Tab.1 Complexity and time delay of AND gate network and XOR gate number required for partial multiplication

Coefficients of $C'(x)$	Number of AND gates used	Number of XOR gates required	AND network time delay
$c_0' = a_0 b_0$	1	0	T_a
⋮	⋮	⋮	⋮
$c_i' = \sum_{j=0}^{i} a_j b_{i-j}, 0 \le i \le m-1$	$i+1$	i	T_a
$c_i' = \sum_{j=i-m+1}^{m-1} a_j b_{i-j}, m \le i \le 2m-2$	$2m-(i+1)$	$2m-(i+2)$	T_a
⋮	⋮	⋮	⋮
$c_{2m-2}' = a_{m-1} b_{m-1}$	1	0	T_a
Total	m^2	$(m-1)^2$	T_a

2. Integrated modular reduction in Algorithm 1

According to Section II, to complete modula reduction, the iteration time L equals $\lfloor (n-m)/(m-k) \rfloor + 1$. Let n be its maximum case, *i.e.* $n = 2m-2$, then

$$L = \left\lfloor \frac{m-2}{m-k} \right\rfloor + 1 \tag{7}$$

While k ranges from 1 to $m-1$, L changes according to the following equations:

$$L = \begin{cases} 1, k = 1 \\ 2, 2 \le k \le \left\lceil \dfrac{m}{2} \right\rceil \\ \ge 3, \left\lceil \dfrac{m}{2} + 1 \right\rceil \le k \le m-1 \end{cases} \tag{8}$$

Three cases of Eq.(8) should be considered separately. However, the demand of irreducible polynomial for ECC determines that k is always far less than $\lceil m/2 \rceil$ [11]. So, only two cases are considered below.

Case 1 If $k = 1$, *i.e.*, only one iteration is needed to complete modular reduction. Let

$$C'(x) \equiv C_h(x)(x+1) + C_l(x) \bmod f(x) \tag{9}$$

We can acquire

$$\begin{aligned} C(x) &= c'_{2m-2}x^{m-2} + c'_{2m-3}x^{m-3} + \cdots + c'_{m+1}x \\ &\quad + c'_m + \left(c'_{2m-2}x^{m-2} + c'_{2m-3}x^{m-3} + \cdots \right. \\ &\quad \left. + c'_{m+1}x + c'_m\right)x + c'_{m-1}x^{m-1} + \cdots \\ &\quad + c'_1 x + c'_0 \\ &= \left(c'_{m-1} + c'_{2m-2}\right)x^{m-1} + \left(c'_{m-2} + c'_{2m-2} \right. \\ &\quad \left. + c'_{2m-3}\right)x^{m-2} + \cdots + \left(c'_1 + c'_{m+1} + c'_m\right)x \\ &\quad + \left(c'_0 + c'_m\right) \end{aligned} \tag{10}$$

Just like Tab.1, numbers of XOR gates and delays of XOR gate network are shown in Tab.2. Note that the XOR gate network includes XOR gates both in Tab.1 and Tab.2. Outputs of the AND gates can be used more than one time, so m^2 AND gate network can provide all inputs needed by XOR gate network under this method. An example is given to explain the organization of AND and XOR gates in the next section.

Tab.2 Complexity and time delay on computing $C(x) = A(x)B(x) \bmod (x^m + x + 1)$

Coefficients of $C(x)$	Number of extra XOR gates	XOR Time delay
$c_0 = c'_0 + c'_m$ $= a_0b_0 + a_1b_{m-1} + a_2b_{m-2} + \cdots + a_{m-1}b_1$	1	$\lceil \log_2 m \rceil T_x$
$c_1 = c'_1 + c'_{m+1} + c'_m$ $= a_0b_1 + a_1b_0 + a_2b_{m-1} + \cdots + a_{m-1}b_2 + a_1b_{m-1}$ $+ \cdots + a_{m-1}b_1$	2	$\lceil \log_2(2m-1) \rceil T_x$
$c_2 = c'_2 + c'_{m+2} + c'_{m+1}$ $= a_0b_2 + a_1b_1 + a_2b_0 + a_3b_{m-1} + \cdots + a_{m-1}b_3$ $+ a_2b_{m-1} + \cdots + a_{m-1}b_2$	2	$\lceil \log_2(2m-2) \rceil T_x$
$\vdots$	$\vdots$	$\vdots$
$c_{m-1} = c'_{m-1} + c'_{2m-2}$ $= a_0b_{m-1} + \cdots + a_{m-1}b_0 + a_{m-1}b_{m-1}$	1	$\lceil \log_2(m+1) \rceil T_x$
Total	$2m-2$	$\left\lceil \log_2(2m-1)\, T_x \right\rceil$

As a conclusion, implementation of $C(x) = A(x)\,B(x) \bmod (x^m + x + 1)$ needs extra $(2m-2)$ XOR gates besides those needed in Tab.1. The maximal time delay of XOR gate network was $\left\lceil \log_2(2m-1) \right\rceil T_x$. The time delay of modular multiplication of this multiplier is $T_a + \lceil \log_2(2m-1) \rceil T_x$.

Case 2 If $2 \le k \le \lceil m/2 \rceil$, according to Eq.(8), two iterations are needed to compute modular

reduction. They are deducted below to get the expressions of c_i.

Step 1

$$C'(x)=C_h(x)(x^k+1)+C_l(x) \bmod f(x) \tag{11}$$

$$\begin{aligned}C'(x) &= c'_{2m-2}x^{m-2}+c'_{2m-3}x^{m-3}+\cdots \\ &\quad + c'_{m+1}x+c'_m+\left(c'_{2m-2}x^{m-2}\right. \\ &\quad \left.+c'_{2m-3}x^{m-3}+\cdots+c'_{m+1}x+c'_m\right)x^k \\ &\quad + c'_{m-1}x^{m-1}+c'_{m-2}x^{m-2}+\cdots+c'_1x+c'_0 \\ &= \left(c'_{2m-2}x^{m+k-2}+c'_{2m-3}x^{m+k-3}+\cdots\right. \\ &\quad \left.+c'_{2m-k}x^m\right)\left(c'_{2m-k-1}+c'_{m-1}\right)x^{m-1}+\left(c'_{2m-k-2}\right. \\ &\quad \left.+c'_{2m-2}+c'_{m-2}\right)x^{m-2}+\cdots+\left(c'_{m+1}\right. \\ &\quad \left.+c'_{m+k+1}+c'_{k+1}\right)x^{k+1}+\left(c'_m+c'_{m+k}+c'_k\right)x^k \\ &\quad +\left(c'_{m+k-1}+c'_{k-1}\right)x^{k-1}+\cdots+\left(c'_{m+1}+c'_1\right)x \\ &\quad +\left(c'_m+c'_0\right)\end{aligned} \tag{12}$$

Step 2

$$\begin{aligned}C'_h(x) &= c'_{2m-2}x^{k-2}+c'_{2m-3}x^{k-3} \\ &\quad +\cdots+c'_{2m-k}\end{aligned} \tag{13}$$

$$\begin{aligned}C'_l(x) &= \left(c'_{2m-k-1}+c'_{m-1}\right)x^{m-1}+\left(c'_{2m-k-2}+c'_{2m-2}\right. \\ &\quad \left.+c'_{m-2}\right)x^{m-2}+\cdots+\left(c'_{m+1}+c'_{m+k+1}\right. \\ &\quad \left.+c'_{k+1}\right)x^{k+1}+\left(c'_m+c'_{m+k}+c'_k\right)x^k \\ &\quad +\left(c'_{m+k-1}+c'_{k-1}\right)x^{k-1} \\ &\quad +\cdots+\left(c'_{m+1}+c'_1\right)x \\ &\quad +\left(c'_m+c'_0\right)\end{aligned} \tag{14}$$

The final result $C(x)$ fulfills the expression below:

$$\begin{aligned}C(x) &= (c'_{2m-2}x^{k-2}+c'_{2m-3}x^{k-3}+\cdots+c'_{2m-k})(x^k+1) \\ &\quad +\left(c'_{2m-k-1}+c'_{m-1}\right)x^{m-1}+(c'_{2m-k-2}+c'_{2m-2} \\ &\quad +c'_{m-2})x^{m-2}+\cdots+(c'_{m+1}+c'{}_{m+k+1}+c'_{k+1})x^{k+1} \\ &\quad +(c'_m+c'_{m+k}+c'_k)x^k+(c'_{m+k-1}+c'_{k-1})x^{k-1} \\ &\quad +\cdots+(c'_{m+1}+c'_1)x+(c'_m+c'_0) \\ &= (c'_{2m-k-1}+c'_{m-1})x^{m-1}+(c'_{2m-k-2}+c'_{2m-2} \\ &\quad +c'_{m-2})x^{m-2}+\cdots+(c'_{2m-2}+c'_{m-k-2}+c'_{m+2k-2} \\ &\quad +c'_{2k-2})x^{2k-2}+(c'_{2m-3}+c'_{m+k-3}+c'_{m+2k-3} \\ &\quad +c'_{2k-3})x^{2k-3}+\cdots+(c'_{2m-k}+c'_m+c'_{m+k} \\ &\quad +c'_k)x^k+(c'_{m+k-1}+c'_{k-1})x^{k-1}+(c'_{2m-2} \\ &\quad +c'_{m+k-2}+c'_{k-2})x_{k-2}+\cdots+(c'_{2m-k+1} \\ &\quad +c'_{m+1}+c'_1)x+(c'_{2m-k}+c'_m+c'_0)\end{aligned} \tag{15}$$

As the same manner in Case 1, number of XOR gates and time delay of XOR gates network are shown in Tab.3. Since four addition items are included from $c_k=c'_{2m-k}+c'_m+c'_{m+k}+c'_k$ to $c_{2k-2}=c'_{2m-2}+c'_{m+k-2}+c'_{m+2k-2}+c'_{2k-2}$ and the first two of them are the same as those in $c_0=c'_{2m-k}+c'_m+c'_0$ to $c_{k-2}=c'_{2m-2}+c'_{m+k-2}+c'_{k-2}$, two more XOR gates are needed. Thus, the maximal time delay of XOR gate network is $\lceil\log_2(2m+k-2)\rceil T_x$, which occur at c_k. Because $k<m/2$, the time delay of the multiplier is less than $T_a+\lceil\log_2(5m/2-2)\rceil T_x$, *i.e.*, $T_a+(\lceil\log_2(5m/8-1/2)\rceil+2)T_x$.

IV. The Multiplier Architecture

From Tab.1, Tab.2 and Tab.3, a bit-parallel modular multiplier with AND gate network and XOR gate network are constructed. AND gate network consists of m^2 two-input AND gates. XOR gates network consists of m^2-1 XOR gates. To locate these XOR gates, some rules should be obeyed: (1) Items of c'_m to c'_{2m-2} should be added into one item before they were added to others in Case 1; (2) The coefficients pair fulfill $(c_i+c_{i+m-k}), i\in[m,m+k-2]$ should be added into one item first in Case 2.

As an example, the circuits to build the modular multiplier in GF(2^4) are shown in Fig.1, where $f(x)=x^4+x+1$. From Tab.1 and Tab.2, the AND gate network and XOR gate network are built, which consists of 16 AND gates and 15 XOR gates. Deductions are followed bellow.

$$\left.\begin{aligned}c_0 &= c'_0+c'_4=a_0b_0+(a_1b_3+a_2b_2+a_3b_1) \\ c_1 &= c'_1+c'_5+c'_4=a_0b_1+a_1b_0+a_2b_3 \\ &\quad +a_3b_2+a_1b_3+a_2b_2+a_3b_1 \\ c_2 &= c'_2+c'_6+c'_5=a_0b_2+a_1b_1+a_2b_0 \\ &\quad +a_3b_3+a_2b_3+a_3b_2 \\ c_3 &= c'_3+c'_6=a_0b_3+a_1b_2+a_2b_1+a_3b_0+a_3b_3\end{aligned}\right\}$$

It is checked that the total time delay is

$$T_a + \lceil \log_2(2m+k-2) \rceil T_x$$

$$= T_a + \lceil \log_2(8+1-2) \rceil T_x$$

$$= T_a + 3T_x$$

Tab.3 Complexity and time delay on computing $C(x) = A(x)B(x) \bmod (x^m + x^k + 1)$, where $2 \le k \le \lceil m/2 \rceil$ and $m \ge 4$

Coefficients of $C(x)$	Extra XOR gates	XOR time delay
$c_0 = c'_{m-k} + c'_m + c'_0$	2	$\lceil \log_2(m+k-1) \rceil T_x$
$c_1 = c'_{2m-k+1} + c'_{m+1} + c'_1$	2	$\lceil \log_2(m+k-2) \rceil T_x$
$\vdots$	$\vdots$	$\vdots$
$c_{k-1} = c'_{m+k-1} + c'_{k-1}$	1	$\lceil \log_2 m \rceil T_x$
$c_k = c'_{2m-k} + c'_m + c'_{m+k} + c'_k$	2	$\lceil \log_2(2m+k-2) \rceil T_x$
$c_{k+1} = c'_{2m-k+1} + c'_{m+1} + c'_{m+k+1} + c'_{k+1}$	2	$\lceil \log_2(2m+k-4) \rceil T_x$
$\vdots$	$\vdots$	$\vdots$
$c_{m-1} = c'_{2m-k+1} + c'_{m-1}$	1	$\lceil \log_2(m+k) \rceil T_x$
Total	$2m-2$	$\lceil \log_2(2m+k-2) \rceil T_x$

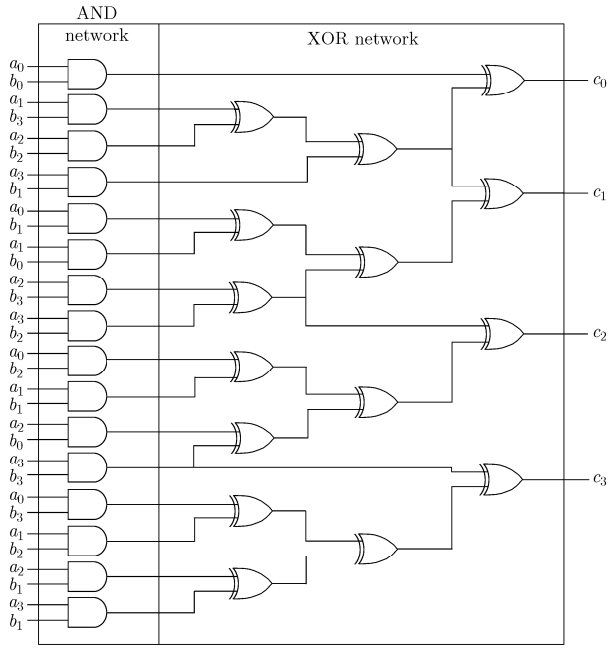

Fig.1 Bit-parallel modular multiplier in GF(2^4)

V. Comparisons

A comparison of different implementations of bit-parallel multiplier in the class of fields generated with irreducible trinomials in terms of gate number and the multiplication time delay is listed in Tab.4. These multipliers chosen represent the best performance to date using polynomial basis and weakly dual basis. Since the multiplier proposed is designed for cryptographic applications, we concentrated on the case $k < m/2$ in trinomial $f(x) = x^m + x^k + 1$. The results show that the time delay of multiplier presented is equal to other multipliers when $k = 1$, and less than that of other multipliers when $1 < k \le m/2$, with the same number of AND gates and XOR gates.

VI. Conclusions

Based on the fast modular multiplication in GF(2^m) recommended by the NIST, a new structure of multiplier is proposed. It has less time delay than other multipliers with the same number of gates for the class of fields generated by irreducible trinomials, especially in ECC where the second lowest degree of the trinomials is often much less than the degree of the field. The structure is practical, especially in real time cryptographic applications.

Future work will involve the fields generated with irreducible pentanomials and improving performance when the second highest degree k is larger than half of the field degree, *i.e.*, $k > m/2$.

Tab.4 Gate number and multiplication time delay comparisons

Proposals	Trinomials	Number of AND gates	Number of XOR gates	Modular multiplication time delay
Weakly Dual Basis[8]	$f(x)= x^m + x + 1$	m^2	m^2-1	$T_a + \left(\lceil \log_2 m \rceil + 1\right) T_x$
PB Mastrovito[6]		m^2	m^2-1	$T_a + \left(\lceil \log_2 m \rceil + 1\right) T_x$
PB Montgomery[5]		m^2	m^2-1	$T_a + \left(\lceil \log_2 (m-2) \rceil + 2\right) T_x$
PB conventional[10]		m^2	m^2-1	$T_a + \left(\lceil \log_2 (m-2) \rceil + 2\right) T_x$
The proposed		m^2	m^2-1	$T_a + \left(\lceil \log_2 (2m-1) \rceil\right) T_x$
Weakly Dual Basis[8]	$f(x) = x^m + x^k + 1,$ $1 < k < m/2$	m^2	m^2-1	$T_a + \left(\left\lceil \log_2 \left\lceil \frac{m+k-1}{2} \right\rceil \right\rceil + 2\right) T_x$
PB Mastrovito[6]		m^2	m^2-1	$T_a + \left(\lceil \log_2 m \rceil + 2\right) T_x$
PB Montgomery[5]		m^2	m^2-1	$T_a + \left(\lceil \log_2 (m-2) \rceil + 2\right) T_x$
PB conventional[10]		m^2	m^2-1	$T_a + \left(\lceil \log_2 (m-1) \rceil + 2\right) T_x$
The proposed		m^2	m^2-1	$< T_a + \left(\left\lceil \log_2 \left(\frac{5}{8} m - \frac{1}{2} \right) \right\rceil + 2\right) T_x$

References

[1] J. Lopez and R. Dahab. Fast multiplication on elliptic curves over $\mathrm{GF}(2^m)$ without precomputation. Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), Worcester, Massachusetts, USA, August 12–13, 1999, LNCS 1717, 1999, 316–327

[2] J. Bajard, L. Imbert, and G. A. Jullien. Parallel Montgomery multiplication in $\mathrm{GF}\left(2^k\right)$ using trinomial residue arithmetic. 17th IEEE Symposium on Computer Arithmetic (ARITH'05), Cape Cod, Massachusetts, USA, June 27–29, 2005, 164–171.

[3] C. Grabbe, M. Bednara, J. Teich, *et al.* FPGA designs of parallel high performance $\mathrm{GF}\left(2^{333}\right)$ multipliers. The IEEE International Symposium on Circuits and Systems (ISCAS'03), Bangkok, Thailand, May 25, 2003, vol.2, 268–271.

[4] C. Koc and B. Sunar. Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields. *IEEE Trans. on Computers*, **47**(1998)3, 353–356.

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

[6] B. Sunar and C. Koc, Mastrovito multiplier for all trinomials. *IEEE Trans. on Computers*, **48**(1999)5, 522–527.

[7] M. Schimmler and V. Bunimov. Fast modular multiplication by operand changing. International Symposium on Information Technology: Coding and Computing (ITCC'2004), Las Vegas, NV, USA, April 05–07, 2004, vol.2, 518–524.

[8] H. Wu, M. A. Hasan, and I. F. Blake. New low-complexity bit-parallel finite field multipliers using weakly dual bases. *IEEE Trans. on Computers*, **47**(1998)12, 1223–1234.

[9] A. Reyhani-Masoleh and M. A. Hasan. A new construction of massey-omura parallel multiplier over $\mathrm{GF}(2^m)$. *IEEE Trans. on Computers*, **51**(2002)5, 511–520.

[10] H. Wu. Bit-parallel finite field multiplier and squarer using polynomial basis. *IEEE Trans. on Computers*, **51**(2002)7, 750–758.

[11] D. Hankerson, A. Menezes, and S. Vanstone. Guide to Elliptic Curve Cryptography. New York, Springer-Verlag, 2004, 2.3.2–2.3.5.