A Study on Finite Field Multiplication over GF (2m) and its application on Elliptic Curve Cryptography

Ashutosh Narayan Mishra[1], Dr. R.K.Singh [1]
[1]*Kamla Nehru Institute of Technology (KNIT), Sultanpur, Uttar Pradesh, India*
Email- mishra.as123@gmail.com, singhrabinder57@gmail.com

ABSTRACT: The paper presents an extensive and careful study of finite field multiplication over GF (2^m) using polynomial basis as well as special polynomial like trinomials, pentanomial and All one polynomial (AOP). This multiplication is done by using montgomery multiplication scheme and application of it is also given. This paper focuses on different arithmetical operation on elliptic curve cryptography over GF (2^m). The parameter performance is also discussed in term of number of component, latency, space and time complexity.

Keywords: *Elliptic curve cryptography, Finite Field, polynomial basis, montgomery multiplication, LSD-First, MSD-First.*

1. Introduction

Finite field has received a lot of attention due to its widespread applications and computation in cryptography [1], coding theory, error control, digital signal processing [2,3]. Finite field arithmetic is used in Cryptography. Elliptic curve cryptography [4,5]and RSA[6] is two important Public Key cryptosystem. All the low-level operations are carried out in finite fields.

The mathematical model of finite field includes addition, subtraction, multiplication, divison, inversion and squaring etc. Finite field arithmetic operates in prime, binary and extension field which is significant tool of public key cryptography. Addition and multiplication are two basic operations in the binary finite field GF(2^m). Addition in GF(2^m) is easily realized using m two-input XOR gates while multiplication is complex operation and its performance is counted in terms of gate count and time delay. The other operations of finite fields, such as exponentiation, division, and inversion can be performed by repeated multiplications. For hardware implementation of cryptographic application binary field is more suitable.

Finite Field as vector space**,** polynomial basis [7-15] normal basis [16,17] dual basis [18,19] and redundant basis is used to represent the field. Each bases have their own advantage and disvantage, such as for square of element normal base is preferred than others. Squaring is performed in normal basis just by a cyclic right-shift, while it is performed by bit-extension through insertion of 0 between the consecutive bits followed by modular reductions to reduce the extended polynomial of degree2m-2 to degree m-1 in case of polynomial bases. Inversion is also requires less area and time-complexity in normal basis. But as our requirement is finite field multiplication, polynomial basis has better performance**.**

The polynomial basis multipliers have low design complexity, simplicity, modularity ,regularity, offer scalability for the fields of higher orders, and does not require a basis conversion in multiplication architecture, So, it is widely used in hardware realization of system. Systolic array architecture is prefferd in polynomial basis (PB) multiplier in which a basic cell is repeated in an array and signals flow unilaterally between neighbors.

Multiplication in PB contains two steps: partial multiplication and modular reduction. Modular multiplication is the most important arithmetic operation in ECC cryptosystem. For efficient implementation of modular multiplication, montgomery multiplication algorithm was proposed by Montgomery[20] .THe montgomery multiplication algorithm does not require division operations and it performs the reduction operation depending on the least significant digit rather than the most

significant digit. The modular multiplication using Montgomery multiplication algorithm is shown by Koc and Acar [21] in GF(2^m) fields. The Montgomery multiplication
is used to design an Elliptic Curve Cryptography (ECC) based crypto-processor in [22]. it is implemented with a semi-systolic array structure in [23]. Semi-systolic array structures provide low latency in comparison to systolic array implementations and require fewer latches. Also, they can be pipelined to increase the throughput of the system. An efficient architecture for bit parallel Montgomery multiplier and squarer in GF(2^m) fields generated with irreducible trinomials proposed by Wu [24]. Fournaris and Koufopavlou [25] presented both pipelined and folded architectures for the Montgomery multiplication in GF(2^m). Bajard et al. [26] provided a Montgomery multiplier over GF(pm) other than GF(2^m). An unified multiplier for GF(P) and GF(2^m) which uses Montgomery multiplication algorithm for both fields is introduced by E. Sava_ et al [27] in 2000. The multiplier based on word-size pipeline structure can handle operands of any size. But their pre-computed constants and transformations uses montgomery algorithm for multiplier. [A] Chin-Wun Chiou et all present time independent Montgomery multiplication scheme in GF (2^m). Two unified multipliers for both fields which are scalable and offer faster computation of multiplication is proposed by Savas et al. [28]. Their multipliers are also based on the Montgomery algorithm and provided high-radix design for low-power and high-performance applications. Satoh and Takano [29] introduced a scalable dual-field processor for Elliptic Curve Cryptograph by using the Montgomery algorithm.

In this paper, section 2 represents Fundamental of Finite Field and Elliptic curve cryptography, section 3 represents traditional bit –level Montgomery Multiplication and Time-independent Montgomery Multiplication, Section 4 represents Polynomial multiplication, this section further divided [A],[B] and [C], and this section contain both polynomial multiplication as well as montgomery multiplication scheme using trinomial, pentanomial and all-one-polynomial, further [A],[B] and [C] is divided in to 1,2,3 part according to requirement and last section of this paper 5 is conclusion of this paper.

2. Fundamental of Finite Field and Elliptic curve cryptography:

Field is mathematical place where we can do addition, multiplication, divison, inverse, square etc. Basically, A field is a set F with a multiplication and addition operation which satisfy given rules i.e. associativity and commutativity of both addition and multiplication, the distributive law, existence of an additive identity 0 and a multiplicative identity 1, additive inverses, and multiplicative inverses for everything except 0. The finite field F_2^m is the characteristic 2 finite field containing 2^m elements. Although there is only one characteristic 2 finite field F_2^m for each power 2^m of 2 with m≥1, there are many different ways to represent the elements of F F_2^m. Elements of F_2^m should be represented by the set of binary polynomials of degree m-1 or less i.e.

$$a_{m-1}x^{m-1} + \cdots\cdots + a_1x + a_0$$

Addition: The addition is quite simple in F_2^m , C=A+B where $A = a_{m-1}x^{m-1} + \cdots\cdots + a_1x + a_0$,
$B = b_{m-1}x^{m-1} + \cdots\cdots + b_1x + b_0$. It is logical XOR operation i.e. module 2 addition. So $c_i=a_i+b_i$, where + denote bit-wise XOR operation, a_i and b_i represent the field element of F_2^m where i varies from 0 to m-1.To represent it, polynomial, shifted polynomial and normal basis is used.

Multiplication: Multiplication is different and complicated than addition in F_2^m. In polynomial and shifted polynomial basis C= $AB \bmod F(x)$, in normal basis C=A.B represent the multiplication. Koc and Acar [21] introduced montgomery multiplication algorithm for fast modular integer multiplication.

Square and inversion: Square is special form of multiplication. For square normal basis is preferred over polynomial, as it is circular left shift in normal basis while it is much complicated in polynomial basis. If A Є F_2^m then to find A^{-1}, such that $A^{-1}.A=1$. Itoh and Tsuji [B] proposed inversion algorithm that is used for hardware implementation.

Let F_2^m be characteristic 2 finite field and let a,b Є F_2^m satisfy b≠0 in F_2^m. Then elliptic curve E (F_2^m) over F_2^m defined by the parameters a,;b& F_2^m consists of the set of solutions or points P= (x;,y) for x,y & F_2^m to the equation : $y^2+xy=x^3+ax^2+b$ with a point at infinity *o.*
If the number of point in elliptic curve is denoted by #E(F_2^m) then according to Hasse theorem

$$2^m+1-2\sqrt{2^m} \le \#E(F_2^m) \le 2^m+1+2\sqrt{2^m}$$

1. Rule to add the point at infinity to itself:
   $O+O=O$
2. Rule to add the point at infinity to any other point:
   $(x,y)+O=O+(x,y)=(x,y)$ for all (x,y) ϵ E (F_2^m)
3. Rule to add two points with the same *x*-coordinates when the points are either distinct or have X-coordinate 0: $(x,y)+(x,x+y)=O$ for all (x,y) Є E (F_2^m), the negative of the point (x,y) is -(x,y) = (x,x+y).
4. Rule to add two points with different *x*-coordinates: Let (x_1,y_1) ϵ $E(F_2^m)$ and (x_2,y_2) Є $E(F_2^m)$ be two points such that $x_1 \neq x_2$. Then $(x_1,y_1)+(x_2,y_2)=(x_3,y_3)$, where:
   $x_3=\lambda^2+\lambda+x_1+x_2+a$ and $y3=\lambda(x_1+x_3)+x_3+y_1$ in E (F_2^m) and $\lambda=y_1+y_2/x_1+x_2$ in E (F_2^m)
5. Rule to add a point to itself (double a point): Let (x_1,y_1) ϵ $E(F_2^m)$ be a point with $x_1 \neq 0$. Then
   $(x_1,y_1)+(x_1,y_1)=(x3;y3)$, where: $x_3=\lambda^2+\lambda+a$, $y3=x_1^2+(\lambda+1)x_3$ and $\lambda=x_1+y_1/x_1$

Here elliptic curve E represented by the affine coordinates (x1, y1) and (x2, y2), respectively. From above it is clear that inversion is required to represent rule no. 4 and 5 i.e. point add and double a point. As inversion is an expensive operation in finite field. So,we can use projective-coordinate approach proposed by Lopez and Dahab [30] to improve their performance . Scalar multiplication of elliptic curve points is the main operation in ECC, known as point multiplication. For given an integer *k* and a point *P* ϵ $E(F_2^m)$,scalar multiplication is the process of adding *P* to itself *k* times. The result of this scalar multiplication is denoted kxP or kP. It may be achieved by Montgomery ladder scalar multiplication scheme using Lopez and Dahab coordinates.

3. Montgomery Multiplication:

The use of modular multiplication is the most important arithmetic tool in Finite field. Montgomery Multiplication may be applied for both GF(p) and GF(2^m). But the requirement of montgomery algorithm is pre-computation constant and the basis conversion. The modular multiplication using Montgomery multiplication algorithm is shown by Koc and Acar [21] in GF(2^m) fields. Basically, bit-level and word-level are two type of multiplication algorithm.

.

Time independent Montgomery algorithm [42] shows time and space complexity reduction over bit-level time independent Montgomery multiplication algorithm.

Let A(x*)* and B(x*)* be elements in GF(2^m) generated by an irreducible polynomial F(x) of degree m. The set {x^0, x^1, x^2, …, x^{m-1}} is called the polynomial basis. Three terms A(x*)*, B(x*)* and F(x) are expressed as follows:

$$A(x)=a_{m-1}x^{m-1}+\cdots\cdots+a_1x+a_0 \quad (1)$$

$$B(x)=b_{m-1}x^{m-1}+\cdots\cdots\cdots+b_1x+b_0 \quad (2)$$

$$F(x)=x^m+f_{m-1}x^{m-1}+\cdots\cdots+f_1x+f_0 \quad (3)$$

According to conventional multiplication scheme, C(x) is given by

$$C(x)=A(x)B(x)\bmod F(x) \quad (4)$$

While according to Montgomery multiplication algorithm computes C(x) as given by

$$C(x)=A(x)B(x)R^{-1}(x) \bmod F(x) \quad (5)$$

here $R^{-1}(x)$ is the inverse of R(x) and chosen independently element of GF(2^m) such that GCD (F(x), R(x))=1. Efficient hardware implementation and complexity depend on the value of R(x), Let $R(x)=x^m$ for simplicity, because using $R(x)=x^m$ only requires ignoring the terms whose powers of x are greater than or equal to m. Montgomery scalar multiplication using projective coordinates requires up to
(m-1)(6M +3A+5S)+(10M +7A+4S +I) clock cycles, where M, A, S, and I represent the number of clock cycles for multiplication, addition, squaring, and inversion, respectively [31]. Montgomery multiplication/squaring with low delay can reduce the overall time complexity of the scalar point multiplication and hence, increase the speed of the elliptic curve processor.

$C(x)=A(x)B(x)\ X^{-m} \bmod F(x),$ (6)

$C(x)= (a_{m-1}x^{m-1} + \cdots\cdots + a_1 x + a_0)X\ B(x)\ X^{-m} \bmod F(x)$ (7)

$C(x)= (\ldots.(((a_0b(x)X^{-1}+ a_1b(x)X^{-1}+ a_2b(x)X^{-1}+\ldots\ldots+ a_{m-1}b(x)X^{-1}$ (8)

In the time dependent montgomery multiplication algorithm,(equation 8 is used for computation) the latency of this semi-systolic multiplication array requires m clock cycles for the mxm multiplication, Each clock cycle takes delays of two 2-input AND gates, two 2-input XOR gates, and two 1-bit latches.Figure 1 shows detail architecture of time dependent montgomery multiplication algorithm [42].

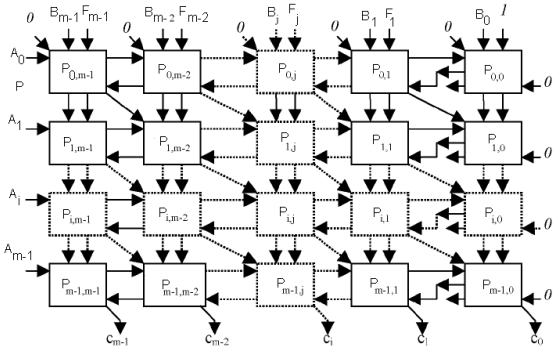
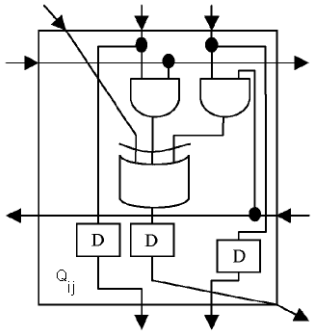


Figure 1: Semi-systolic array of m bit level Montgomery multiplication algorithm



Detail circuit of $P_{i,j}$

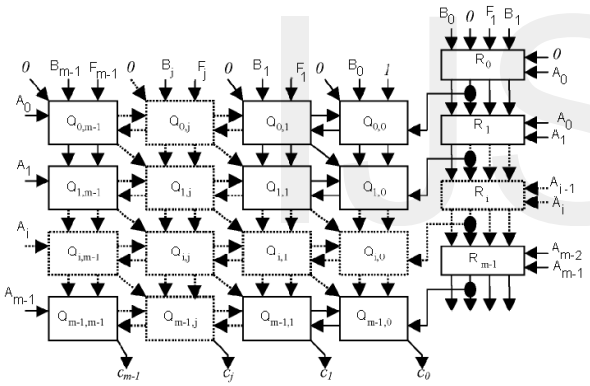
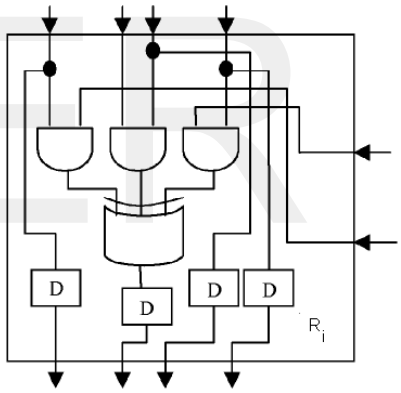


Figure 2: Semi-systolic array of Time independent Montgomery multiplication algorithm



Detail circuit of $Q_{i,j}$

In the Time independent montgomery multiplication algorithm, latency of this semi-systolic multiplication array with mxm size also needs m + 1 clock cycles, but each clock cycle only takes one 2-input AND gate delay, one 3-input XOR gate delay, and one 1-bit latch delay.Figure 2 shows detail architecture of time independent montgomery multiplication algorithm [42].Comparisons of time and space complexities is given in Table 1.

Table 1: Time and space complexities comparison for Time independent and Time-independent MM

Multipliers		Time-dependent Montgomery Multiplier	Time-Independent Montgomery Multiplier
Time Complexity	2-input AND gate delay	2m	m + 1
	2-input XOR gate delay	2m	0
	3-input XOR gate delay	0	m + 1
	1-bit Latch	2m	m + 1
	Total Delays (unit delays)	16mΔ	(8m + 8)Δ
Space Complexity	2-input AND gate	$2m^2$	$2m^2 + 3m$
	2-input XOR gate	$2m^2$	0
	3-input XOR gate	0	$m^2 + m$
	1-bit Latch	$4m^2$	$3m^2 + 4m$
	Total Transistors	$72m^2\Omega$	$(64m^2 + 78m)\Omega$

4. Multiplication Scheme:

The multiplier based on polynomial basis systolic array can be divided in to four category namely bit serial, bit parallel, hybrid or super serial, digit serial and combination of them. The performance parameter of multiplier is latency, space, power and time complexity. All one polynomial (AOP), trinomials and pentanomial are popular polynomial reported yet for multiplier. Each type of multiplier has own advantage and specialty.

[A] **Bit-Serial Multiplier**: Only one bit of operand is proceed in bit-serial multiplier and is applicable for small systems, size , cost and bandwidth are major restriction of such system. It has minimum area and minimum throughput. Generally, bit-serial multipliers are slow. The bit-serial polynomial basis multipliers proposed in [32] are the classic bit-serial multipliers. Systolic array implementation of the polynomial basis multiplication has started in [33] and [34].

Basically there are two type of bit-serial polynomial basis multipliers i.e. LSB-First and the MSB-First bit-serial polynomial basis multipliers [32].The use of Montgomery algorithm optimize the parameter performance. So, procedure of LSB-First and MSB-First bit-serial multiplier is also given.

According to conventional multiplication scheme, C(x) is given by equation (4)

$C(x)=A(x)B(x)\text{Mod } F(x)$

So we can write C(x) or simply C

$C=b_{m-1} A x^{m-1}+\dots +b_1Ax+Ab_0 \text{mod } F(x)$ (9)

$C=b_{m-1} (A x^{m-1} \text{ mod } F(x)) + +\dots +b_1 (Ax \text{ mod } F(x)) +Ab_0 \text{ mod } F(x)$ (10)

If x is root of irreducible polynomial (f_0 and f_m=1*)* of F(z) that satisfy given below equation

$$f_m x^m + f_{m-1}x^{m-1} + \cdots\cdots + f_1 x + f_0 = 0 \quad (11)$$

$$x^m = f_{m-1}x^{m-1} + \cdots\cdots + f_1 x + 1 \quad (12)$$

In this scheme LSB coordinate of B i.e. b_0 is first proceed. Architecture of this scheme is shown in figure 3. Here $A^{'}$ and $C^{'}$ are two m-bit latches, which store values of $A^{(i)}$ and $C^{(i)}$, respectively

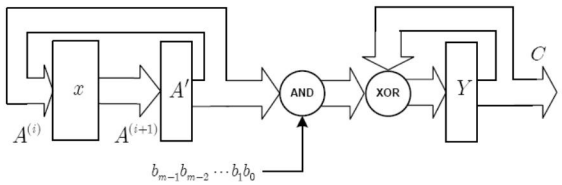


Figure 3: LSB-First bit serial polynomial basis multiplier

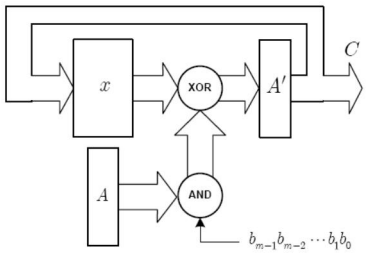


Figure 4: MSB-First bit serial polynomial basis multiplier

So

$A^{(i+1)}= A^{(i)}. x^1 \text{ mod } F(x)$

$$= (a_{m-1}^{(i)} x^m + ... + a_1^{(i)} x^2 + \cdots\cdots + a_0^{(i)} x) \text{ mod } F(x) \quad (13)$$

With the help of (12) and (13)

$$A^{(i+1)} = (a_{m-2}^{(i)} + a_{m-1}^{(i)} f_{m-1}) x^{m-1} + (a_{m-3}^{(i)} + a_{m-1}^{(i)} f_{m-2}) x^{m-2} ... + (a_0^{(i)} + a_{m-1}^{(i)} f_1) x^1 + a_{m-1}^{(i)}) \quad (14)$$

this module multiplies $A^{(i)}$ by x and reduces the results by F(x).

With application of Horner's rule equation (9) can be written as

$$C=(\ldots\ldots(b_{m-1} A x \text{ Mod } F(x) + b_{m-2} A)x \text{ Mod } F(x) + \ldots\ldots + b_1 A)x \text{ Mod } F(x) + b_0 A \quad (15)$$

In MSB –First bit serial scheme b_{m-1} is first proceed (equation 15). Architecture of MSB –First bit serial is shown in Figure 4.

[1] Bit-Serial Montgomery Multiplier :As Montgomery multiplication is given as $C(x)=A(x)B(x)\ X^{-m} \text{ mod } F(x)$, but taking general consideration and choosing $R = x^u, 1 \le u \le m$

i.e. $C=A.B.\ x^{-u} \text{ Mod } F(x)$ (16)

$$C = b_0 Ax^{-u} + b_1 Ax^{-u+1} + \ldots\ldots\ldots + b_{m-1} Ax^{m-u-1} \text{ mod } F(x) \quad (17)$$

where x is root of polynomial F(z). With condition of irreducible polynomial (f_0 and f_m=1).

$$f_m x^m + f_{m-1} x^{m-1} + \cdots\cdots + f_1 x + f_0 = 0 \quad (18)$$

$$\text{So, } x^{-1} \text{ mod } F(x) = x^{m-1} + \ldots.. f_2 x + f_1 \quad (19)$$

Taking $R = x^u, 1 \le u \le m$ and $R = x^{m-1}$, and using equation (19) , we can design the Bit-Serial Montgomery Multiplier.

[2] MSB-First Bit-Serial MM: In this scheme MSB of B i.e. b_{m-1} is first considered with one bit at each cycle. So , C is given by

$$C = b_{m-1} Ax^{m-u-1} + \_\_\_ + b_1 Ax^{-u+1} + b_0 Ax^{-u} \text{ mod } F(x) \quad (20)$$

so, according to MM scheme, there is requirement of pre-computation i.e. $Ax^{m-u-1} \text{ mod } F(x)$ and complexity of scheme depends upon pre-computation complexity.

With use of Horner's rule and pre-computation factor i.e. $A^{(0)} = Ax^{m-u-1} \text{ mod } F(x)$,

$$A^{(i+1)} = A^{(i)}.\ x^{-1} \text{ mod } F(x)$$

$$= (a_{m-1}^{(i)} x^{m-2} + ... + a_1^{(i)} + \cdots\cdots + a_0^{(i)} x^{-1}) \text{ mod } F(x) \quad (21)$$

From equation (19 and 21)

$$A^{(i+1)} = = a_0^{(i)} x^{m-1} + (a_{m-1}^{(i)} + a_0^{(i)} f_{m-1}) x^{m-2} + ... + (a_2^{(i)} + a_0^{(i)} f_2) x^2 + (a_1^{(i)} + a_0^{(i)} f_1) \text{ mod } F(x)$$

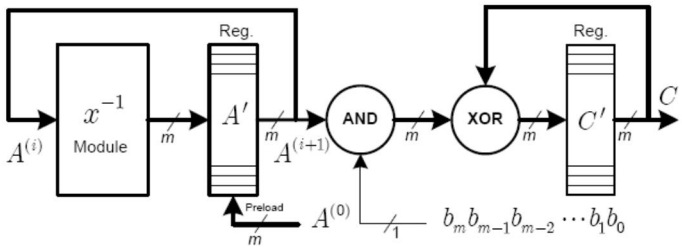


Figure 5: MSB-First bit serial Montgomery multiplication (MM) using $R = x^u$

LSB of B is First proceed in LSB –First MM scheme. Table 2 provides the comparison result.

Table 2: Parameter comparison of Bit-serial multiplier over GF (2^m)

For F(z) $= f_m z^m + f_{m-1} z^{m-1} + \cdots\cdots + f_1 z + f_0$

Type	Algorithm	#XOR	#AND	#Flip Flops	Latency	C.P.D
Polynomial basis	LSB-First	2m-1	2m-1	2m	m	T_A+T_X
	MSB-First	2m-1	2m-1	2m	m	T_A+2T_X
Montgomery multiplication	LSB-First	2m-1	2m-1	2m	m (u=m)	$2T_A+2T_X$
	MSB-First	2m-1	2m-1	2m	m (u=m-1)	T_A+T_X
	LSB-First	2m-1	2m-1	2m	m (u=m-1)	T_A+2T_X

[B] Bit-Parallel Polynomial Basis Multiplication: In Bit-Parallel Polynomial Basis Multiplication, operand are proceed in parallel. This polynomial multiplication scheme can be improve their performance using special irreducible polynomial such as all-one polynomials, trinomials, and pentanomial. In general bit-parallel polynomial basis multiplier is given by $C = b_0A + b_1Ax + ... + b_{m-1}Ax^{m-1} \bmod F(x)$. It is composed of some x-modules which multiply their inputs by x and reduce the results by F(x) [35]. Figure 6 shows architecture of Bit-Parallel polynomial basis multiplier.

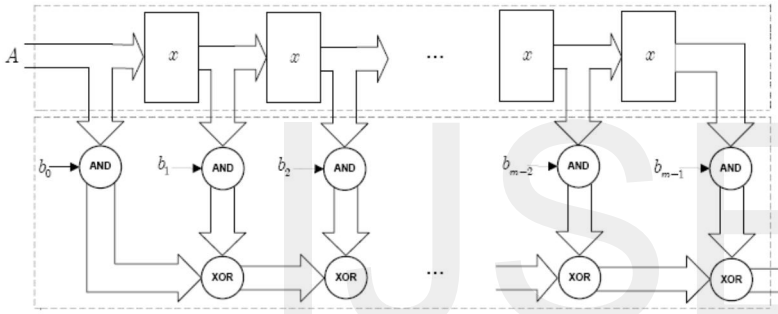


Figure 6: Conventional Bit-Parallel polynomial basis multiplier

[1] Bit-Parallel MM: In general MM can be formulated as $C = A .B R^{-1} \bmod F(x)$, where R can be chosen as $R= x^u$; $0 < u \leq m$. Using equation (16) and (17) we can get a new architecture of the bit-parallel Montgomery multiplier for u = m. In this architecture, the AND modules multiply a field element by a bit, whereas the XOR modules add two field elements. The architecture shown in figure 7.

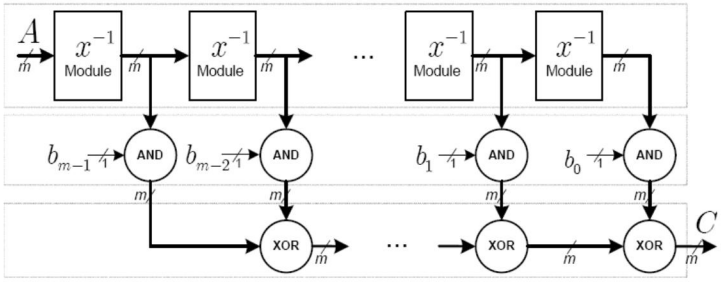


figure 7: Bit-parallel Montgomery multiplier for $R=x^m$

This architecture of figure 7 is similar to figure 6.(bit-parallel polynomial basis multiplier. But in this case i of x^{-1}-modules is used. Here important thing is that order of processing the coordinates of B is reverse. If u have the range of [1,m-1], we can rewrite the Montgomery multiplication as

$C = b_0Ax^{-u} + b_1Ax^{-u+1} + \_ \_ \_ + b_{u-1} Ax^{-1} + b_u A + b_{u+1} Ax + \_ \_ \_ + b_{m-1}Ax^{m-u-1} \bmod F(x)$ (22)

In this case, the main difference is that we multiply A by negative and positive powers of x to calculate the terms in (22). We can rewrite (22) as C = C1 + C2,

Where $C1 = b_0Ax^{-u} + b_1Ax^{-u+1} + \_ \_ \_ + b_{u-1} Ax^{-1} \bmod F(x)$ and $C2 = + b_u A + b_{u+1} Ax + \_ \_ \_ + b_{m-1}Ax^{m-u-1} \bmod F(x)$. Now,

we can design the new architecture of the general case of the MM with $r = x^u$ as depicted in Figure 8.

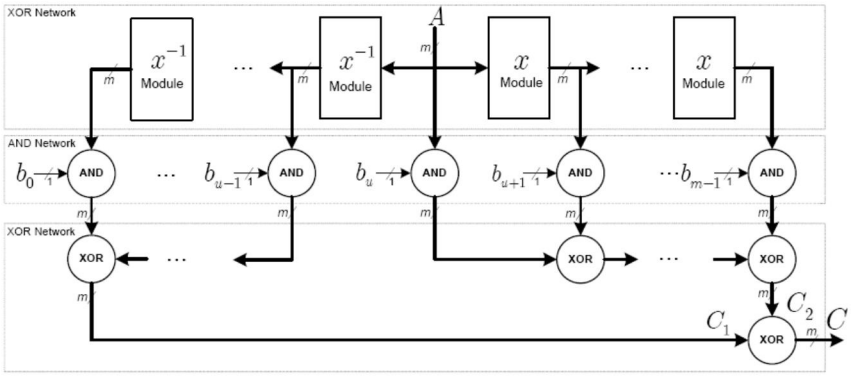


Figure 8: bit-parallel Montgomery multiplier over GF(2^m) with $R=x^u$, $1 \le u \le m-1$

Note that for $1 \le u \le m-1$, the number of the x and x^{-1}-modules is m - 1, as $b_u A$ is obtained directly from A.

Based on the architecture depicted in Fig. 8, the first step of the multiplication is to compute the terms $Ax^i \bmod F(x)$, for $i \in [-u, m-u-1]$. Here, we use $A^{'} i$ to represent $Ax^i \bmod F(x)$. This can be done by using the matrix M, whose columns show the PB representation of $A^{'} i$ for $i \in [-u, m-u-1]$. So, the matrix M has m rows and m columns. Then, the MM over GF(2^m) can be formulated as $[C_0, C_1, \ldots C_{m-1}]^T = M.[b_0, b_1, \ldots b_{m-1}]^T$ (23)

It is similar to the as Mastrovito multiplication [36].

[2] Bit-Parallel Montgomery Multiplier for Irreducible Trinomials

Let $F(z) = z^m + z^k + 1$ be an irreducible trinomial and x be the root of F(z). Then, the Montgomery factor $r = x^u$ is obtained from the following in order to design a fast Montgomery multiplier.

$$U = \begin{cases} 1, & k = 1 \\ k \text{ or } k-1, & k > 1 \end{cases} \quad (24)$$

Entries of the matrix M will be the additions of at most two terms. that the elements of the matrix M are summations of at most two terms if $k - 1 \le u \le k$. This scheme requires m^2 AND gate, $m^2 - \frac{m}{2}$, $k = \frac{m}{2}$ (two-input XOR gate) and $m^2 - 1, k \neq \frac{m}{2}$ (two-input XOR gate). Delay is given by

$$\begin{cases} T_A + [\log_2(2m-u-1)]T_X, & u \le \dfrac{m-1}{2} \\ T_A + [\log_2(m+u)]T_X, & u > \dfrac{m-1}{2} \end{cases}$$

[3] Bit-Parallel Montgomery Multiplier for Irreducible Pentanomials

Irreducible pentanomials are special form of irreducible polynomial which are used in finite field arithmetic, e.g., [37], [38], [39], [40], and [41]. Generally, they can be formulated as

$$F(z) = z^m + z^{k3} + z^{k2} + z^{k1} + 1;\ 1 \le k1 < k2 < k3 < m \quad (25)$$

We assume that $R = x^u$ is the Montgomery factor. For design of bit-parallel Montgomery multipliers matrix M plays an important role. If each column of the matrix M is computed with one step of reduction, then the matrix M can be obtained faster. In this regard, we use a special type of irreducible pentanomial, known as type-II irreducible pentanomial, is defined [41]. as

$$F(z) = z^m + z^{n+2} + z^{n+1} + z^{n+1}, \text{ where } 2 \le n \le [\tfrac{m}{2}]-1 \quad (26)$$

Basically, this approach is lead to Fast Montgomery Multiplier (FMM) and Low Complexity Montgomery Multiplier

(LCMM) which requires less area. It can be obtained by application of Montgomery multiplication scheme as discussed above. The final result of this scheme is given below.

Let $R = x^n$ (u = n) be the Montgomery factor. The fast bit-parallel Montgomery Multiplier (FMM) using type-II irreducible pentanomial of degree m requires time complexity of $T_A + (1 + [\log_2 (m + n)]) T_X$, if $n \geq \frac{m-1}{2}$ and $T_A + (1 + [\log_2 (2m - n - 2)]) T_X$, $n < \frac{m-1}{2}$, also it requires m^2 two-input AND gates and $m^2 + 3m - 9$ two-input XOR gates.

Again let $R = x^n$ (u = n) is used as the Montgomery factor, the low complexity bit-parallel Montgomery multiplier (LCMM) using the type-II irreducible pentanomial F(z) requires time complexity of $T_A + (1 + [\log_2 ([\frac{m-u}{2}]+4u-5)] T_X$, if $u > \frac{m-1}{2}$ and $T_A + (1 + [\log_2 ([\frac{u+1}{2}]+4m-4u-9)] T_X$, if $u \leq \frac{m-1}{2}$. Also it requires m^2 two-input AND gates and m^2+2m-3 two-input XOR gates.

A low latency systolic montgomery multiplier for Finite Field GF(2^m) based on Pentanomials is proposed by P.K.mehar[43] in which pre-computed addition technique is used and this design have low latency less area-delay and power-delay complexities.

[C] Digit-Serial Polynomial Basis Multiplication

In a digit-serial multiplier, the bits are grouped as digits and at each cycle, one digit is processed. In Digit-serial multiplication scheme m bit word is broken in to n = $\lceil m/D \rceil$ digits. By choosing different size of digit, it can be possible to compensate the gap between the speed and the amount of required hardware,with increment of digit size more hardware is required. In general we will start from the LSB of the operand B, i.e., b0, and group D consecutive bits as a digit, where D ≥ 2 to be the digit size. So

$$B=\sum_{i=0}^{n-1} B_i\, x^{iD} \quad \text{where } B_i \text{ is given as} \qquad (27)$$

$$B=\sum_{j=0}^{n-1} b_{Di+j}\, x^j, \quad 0\leq i\leq n-2 \qquad (28)$$

$$B=\sum_{j=0}^{m-1-D(n-1)} b_{Di+j}\, x^j, \quad i=n-1 \qquad (29)$$

So according to polynomial basis multiplication scheme C is given as

$$C=A.\sum_{i=0}^{n-1} B_i\, x^{iD} \bmod F(x) \qquad (30)$$

$$=AB_{n-1}\, x^{(n-1)D}+\ldots\ldots\ldots\ldots+AB_1\, x^D + AB_0 \bmod F(x) \qquad (31)$$

As above it can also divided into LSD-First digit-serial polynomial basis multiplication and MSD-First digit-serial polynomial basis multiplication. In first case B_0 and B_{n-1} is proceed in later case. The equation for LSD-First digit-serial polynomial basis multiplication and architecture (Figure 9) is given as

$$C= B_{n-1}(Ax^{(n-1)D} \bmod F(x)) +\ldots\ldots\ldots\ldots+B_1(Ax^D \bmod F(x)) + B_0 A \bmod F(x) \qquad (32)$$

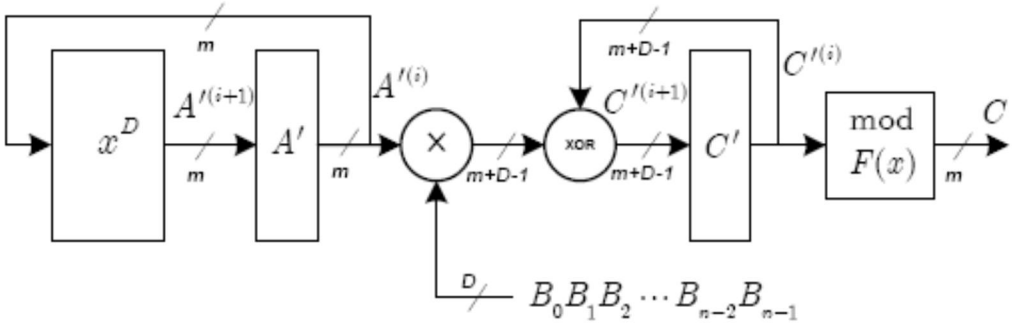


Figure 9: The LSD-first digit-serial polynomial basis multiplier

This multiplier requires latency of n + 1 clock cycles and critical path delay $D(T_A + T_X)$, Dx(3m-2)-m+1 two-input AND gates and Dx(3m-2)-m+1 two-input XOR gates and (2m+D-1) latches.

And for MSD-First digit-serial polynomial basis multiplication equation and architecture (Figure 10) is given as
$C=((B_{n-1} A \bmod F(x)) x^D + B_{n-2} A) \bmod F(x)) x^D \ldots) x^D + B_1 A) \bmod F(x)) \; x^D + B_0 A \bmod F(x)$ (33)

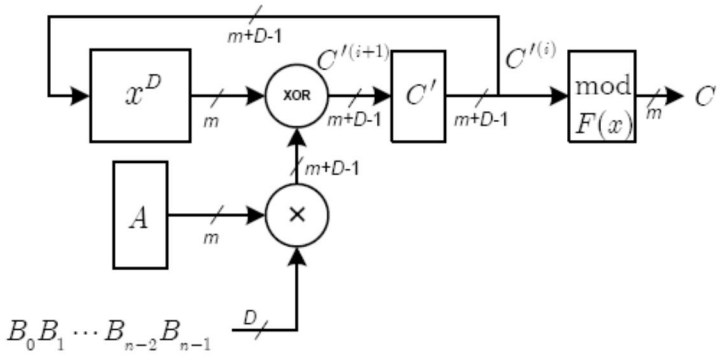


Figure 10: The MSD-first digit-serial polynomial basis multiplier

This multiplier requires latency of n + 1 clock cycles and critical path delay $D(T_A + T_X)+T_X$,D x(3m-2) two-input AND gates and D x(3m-2) two-input XOR gates and (2m + D - 1) latches.
We can improve the performance of Digit serial multiplication by using Shifted polynomial Basis (SPB) scheme.
Using Shifted polynomial basis scheme and using equation (4) SPB multiplication is given as,
$C= B_0 A x^{-v} + B_1 A X^{D-v} + \ldots\ldots\ldots\ldots + B_{n-1} A x^{(n-1)D-v} \bmod F(x)$ (34)
By choosing the appropriate value of v , we may lead different technique (algorithm). If we choose v=(n-1)D, above equation can be written as
$C= B_0 A x^{-(n-1)D} + B_1 A X^{-(n-2)D} + \ldots\ldots\ldots\ldots + B_{n-1} A \bmod F(x)$ (35)
It leads to MSD Digit-Serial multiplication scheme in which B_{n-1} is first proceed.

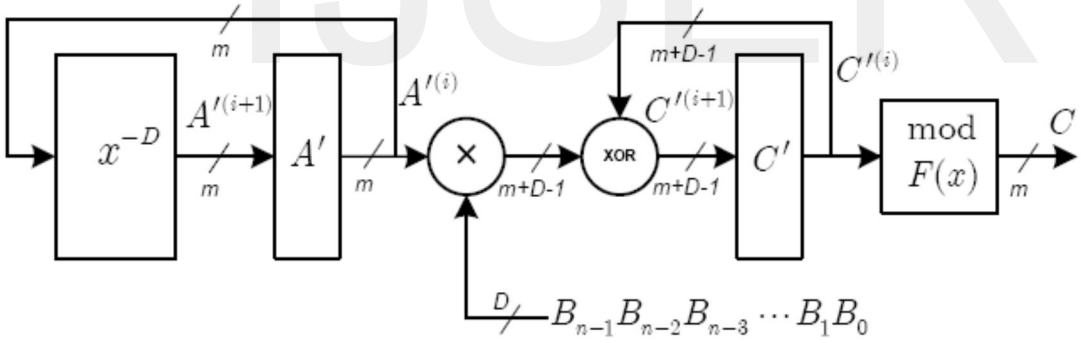


Figure 11: The MSD-first digit-serial SPB multiplier.

Using equation (34) Shifted polynomial basis multiplication can be written as
$C= B_0 A x^{-v} + B_1 A X^{D-v} + B_{\lceil \frac{n}{2} \rceil} A x^{\lceil \frac{n}{2} \rceil D - V} \ldots\ldots\ldots\ldots + B_{n-1} A x^{(n-1)D-v} \bmod F(x)$ (36)
By choosing $v=\lceil n/2 \rceil D$
$C= B_0 A x^{-\lceil \frac{n}{2} \rceil D} + B_1 A \; x^{D-\lceil \frac{n}{2} \rceil D} + \ldots\ldots + B_{\lceil \frac{n}{2}-1 \rceil} A X^{-D} + B_{\lceil \frac{n}{2} \rceil} A + B_{\lceil \frac{n}{2}+1 \rceil} A X^{D} \ldots\ldots + B_{n-1} A x^{\lceil \frac{n-2}{2} \rceil D} \bmod F(x)$ (37)
It leads in to two parts i.e. positive part of x and negative part of x.
$C=C^-+C^+$, C^- is digit serial –SPB with $\lceil \frac{n}{2} \rceil$ of LSD of B, while other is n- $\lceil \frac{n}{2} \rceil$ MSD of operand of B.
$C^- = B_0 A x^{-\lceil \frac{n}{2} \rceil D} + B_1 A x^{D-\lceil \frac{n}{2} \rceil D} + \ldots\ldots + B_{\lceil \frac{n}{2}-1 \rceil} A X^{-D} \bmod F(x)$ (38)

$C^{+}=B_{\left[\frac{n}{2}\right]} A + B_{\left[\frac{n}{2}+1\right]} A X^{D} \ldots\ldots + B_{n-1} A x^{\left[\frac{n-2}{2}\right]D} \bmod F(x)$ (39)

Equation (37, 38, 39) represent Hybrid Digit-Serial SPB Multiplication scheme. By using different basis, we can get better result. Table 3 provides comparison result of Time complexity for Digit-Serial multiplier over GF (2^m)

Table 3: comparison of Time complexity for Digit-Serial multiplier over GF (2^m)

Algorithm	Type	Critical Path delay	Latency
$F(z) = z^m + f_w z^w + \sum_{i=l+1}^{w-1} f_i z^i + f_l z^l + 1,\ D > \min\{l, m-w\}$			
MSD-first	SPB	$D(T_A + T_X)$	$n+1$
Hybrid	SPB	$D(T_A + T_X)$	$\lfloor \frac{n}{2} \rfloor + 2$
$F(z) = z^m + f_w z^w + \sum_{i=l+1}^{w-1} f_i z^i + f_l z^l + 1,\ 2 \le D \le \min\{l, m-w\}$			
MSD-first	SPB	$T_A + \lceil \log_2(D+1) \rceil T_X$	$n+1$
Hybrid	SPB	$T_A + \lceil \log_2(D+1) \rceil T_X$	$[\frac{n}{2}] + 2$

This result can improve using systolic and semi-systolic architecture, using digit-serial systolic multiplication scheme(montgomery algorithm) over all-one –polynomial(AOP) over GF(2^m) have latency (2N-1) clock cycle, where N=$\left[\frac{m}{L}\right]$, m is the word size and L is the digit size[43].

5. Conclusion:

This paper covers architectures and multiplication scheme for binary finite field for implementing the elliptic curve cryptography. Here particularly, we focused on polynomial basis, but include trinomial, pentanomial and all-one-polynomial. We gave a comprehensive summary of finite field arithmetic in cryptography, covering all multiplication scheme and algorithm in order to create time and space efficient implementation of finite field operation. There is a lot of possibility to improve the performance of cryptographic system by using semi-systolic model, special polynomial, montgomery algorithm and different bases. It is also possible to create single hardware architecture that supports several different field and bases at a cost of improvement in multiplication algorithm.

Reference:

[1] Lidl, R. and Niederreiter, H., "*Introduction to Finite Fields and Their Applications"*, New York: Cambridge Univ. Press, U.S.A. (1994).

[2] Blahut, R. E., "*Fast Algorithms for Digital Signal Processing*, Reading, Mass*"*, Addison-Wesley (1985).

[3] Reed, I. S. and Truong, T. K., "The Use of Finite Fields to Compute Convolutions," *IEEE Trans. Information Theory*, Vol. IT-21, pp. 208-213, (1975).

[4] N. Koblitz, "Elliptic Curve Cryptosystems. Mathematics of Computation",48(177):203_209, 1987.

[5] Victor S Miller, "Use of Elliptic Curves in Cryptography. In Proceedings of Advances in Cryptology-CRYPTO 85", Lecture Notes in Computer Science, Vol. 218, pages 417_426, 1986.

[6] Rivest, R. L., Shamir, A. and Adleman, L., "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Comm. ACM*, Vol. 21, pp. 120-126 (1978).

[7] Bartee, T. C. and Schneider, D. J., "Computation with Finite Fields" , *Information and Computing*, Vol. 6, pp. 79-98 (1963).

[8] Mastrovito, E. D., "VLSI Architectures for Multiplication over Finite Field GF(2^m)," Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes, *Proc. Sixth Int'l Conf., AAECC-6, T. Mora, ed.*, Rome, pp. 297-309 (1988).

[9] Koç, Ç. K. and Sunar, B., "Low-complexity Bit-parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields", *IEEE Trans. Computers*, Vol. 47, pp. 353-356 (1998).

[10] Lee, C.-Y., "Low Complexity Bit-parallel Systolic Multiplier over GF(2^m) Using Irreducible Trinomials" , *IEE Proc.-Comput. Digit. Tech.*, Vol. 150, pp. 39-42 (2003).

[11] Itoh, T. and Tsujii, S., "Structure of Parallel Multipliers for a Class of Fields GF(2^m)," *Information and Computation*,

Vol. 83, pp. 21-40 (1989).
[12] Hasan, M. A.,Wang, M. and Bhargava, V. K., "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields GF(2^m)," *IEEE Trans Computers,* Vol. 41, pp. 962-971(1992).
[13] Lee, C. Y., Lu, E. H. and Lee, J. Y., "Bit-parallel Systolic Multipliers for GF(2m) Fields Defined by All-one and Equally-spaced Polynomials", *IEEE Trans. Computers,* Vol. 50, pp. 385-393 (2001).
[14] Paar, C., "A New Architecture for a Parallel Finite Field Multiplier with Low Complexity Based on Composite Fields", *IEEE Trans. Computers,* Vol. 45, pp. 856-861 (1996).
[15] Chiou, C. W., Lin, L. C., Chou, F. H. and Shu, S. F., "Low Complexity Finite Field Multiplier Using Irreducible Trinomials", *Electronics Letters,* Vol. 39, pp. 1709-1711 (2003).
[16] Massey, J. L. and Omura, J. K., "Computational Method and Apparatus for Finite Field Arithmetic", U.S. Patent Number 4,587,627, May (1986).
[17] Reyhani-Masoleh, A. and Hasan, M. A., "A New Construction of Massey-Omura Parallel Multiplier over GF(2^m)", *IEEE Trans. Computers,* Vol. 51, pp. 511-520 (2002).
[18] Berlekamp, E. R., "Bit-serial Reed-Solomon Encoders", *IEEE Trans. Information Theory,* Vol. IT-28, pp. 869-874 (1982).
[19] Wu, H. M., Hasan, A. and Blake, I. F., "New Lowcomplexity Bit-parallel Finite Field Multipliers Using Weakly Dual Bases", *IEEE Trans. Computers,* Vol. 47, pp. 1223-1234 (1998).
[20] Montgomery, P. L., "Modular Multiplication Without Trial Division", *Math. Computation,* Vol. 44, pp. 519- 521 (1985).
[21] Koc, C. K. and Acar, T., "Montgomery Multiplication in GF(2^k)" , *Designs, Codes, and Cryptography,* Vol. 14, pp. 57-69 (1998).
[22] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "High-Performance Public-key Cryptoprocessor for Wireless Mobile Applications. Mobile Networks and Applications", 12(4):245_258, 2007.
[23] N. Mentens, S. B. Ors, B. Preneel, and J. Vandewalle, "An FPGA Implementation of a Montgomery multiplier over GF(2m). In Proceedings of the 7th IEEE Work- shop on Design and Diagnostics of Electronic Circuits and Systems (DDECS) ", pages 121_128, 2004.
[24] Wu, H., "Montgomery Multiplier and Squarer in GF(2^m)," CHES 2000, LNCS 1965, pp. 264-276 (2000).
[25] Fournaris, A. P. and Koufopavlou, O., "GF(2^k) Multipliers Based on Montgomery Multiplication Algorithm" ,*Proc. of the IEEE International Symposium on Circuits and Systems,* Vol. II, pp. 849-852 (2004).
[26] Bajard, J. C., Imbert, L., Negre, C. and Plantard, T., "Efficient Multiplication in GF(p^k) for Elliptic Curve Cryptography" , *IEEE Symp. Computer Arithmetic,* pp. 181-187 (2003).
[27] Sava_, E., Tenca, A. F. and Koç, Ç. K., "A Scalable and Unified Multiplier Architecture for Finite Fields GF(p) and GF(2^m)" , CHES 2000, LNCS 1965, pp. 277-292 (2000).
[28] Savas, E., Tenca, A. F., Çiftçibasi, M. E. and Koç, Ç. K., "Multiplier Architectures for GF(p) and GF(2n)" , *IEE Proceedings-Computers and Digital Technology,* Vol. 151, pp.147-160 (2004).
[29] Satoh, A. and Takano, K., "A Scalable Dual-field Elliptic Curve Cryptographic Processor" , *IEEE Trans. Computers,* Vol. 52, pp. 449-460 (2003).
[30] J. Lopez and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation. Cryptographic Hardware and Embedded Systems", First International Workshop, CHES'99, Worcester, MA, USA, August 1999: Proceedings, pages 316_327, 1999.
[31] B. Ansari and MA Hasan, "High Performance Architecture of Elliptic Curve Scalar Multiplication" , IEEE Transactions on Computers, 57(11):1443_1453, 2008.
[32] T. Beth and D. Gollman, "Algorithm Engineering for Public Key Algorithms. IEEE Journal on Selected Areas in Communications" , 7(4):458_466, 1989.
[33] C.-S. Yeh, I. S. Reed, and T. K. Truong, "Systolic Multiplier for Finite Fields GF(2^m) " , IEEE Transactions on Computers, C-33:357_360, 1983.
[34] C. L. Wang and J. L. Lin, "Systolic Array Implementation of Multipliers for Finite Fields GF(2^m) " , IEEE Transactions on Circuits and Systems, 38(7):796_ 800, 1991.
[35] A. Reyhani-Masoleh and M.A. Hasan, "Fault Detection Architectures for Field Multiplication Using Polynomial Bases" , IEEE Transactions on Computers, 55(9):1089_1103, 2006.
[36]E. D. Mastrovito, "VLSI Architectures for Computation in Galois Fields" , PhD thesis, Linkoping Univ., Linkoping, Sweden, 1991.

[37] H. Fan and M.A. Hasan, "Fast Bit Parallel Shifted Polynomial Basis Multipliers in GF(2^n) " , IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 53(12):2606_2615, 2006.
[38] JL Imana, R. Hermida, and F. Tirado, "Low Complexity Bit-Parallel Multipliers Based on a Class of Irreducible Pentanomials" ,IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 14(12):1388_1393, 2006.
[39] S. M. Park, K. Y. Chang, and D. Hong, "Efficient Bit-Parallel Multiplier for Irreducible Pentanomials Using a Shifted Polynomial Basis" ,. IEEE Transactions on Computers, 55(9):1211_1215, 2006.
[40] A. Reyhani-Masoleh and MA Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2^m)", IEEE Transactions on Computers, 53(8):945_959, 2004.
[41] F. Rodriguez-Henriguez and CK Koc, "Parallel Multipliers Based on Special Irreducible Pentanomials", IEEE Transactions on Computers, 52(12),1535_1542, 2003.
[42] Che-Wun Chiou,Chiou-Yng Lee, An-Wen Deng and Jim-Min Lin, "Efficient VLSI Implementation for Montgomery Multiplication in GF(2^m) " , Tamkang Journal of Science and Engineering, Vol. 9, No 4, pp. 365_372 (2006)
[43] Somsubhra Talapatra, Hafizur Rahaman, and Jimson Mathew, "Low Complexity Digit Serial Systolic Montgomery Multipliers for Special Class of GF(2^m) " , IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 18, NO. 5, MAY 2010.
[44] Jiafeng Xie, Jian jun He, and Pramod Kumar Meher**,** "Low Latency Systolic Montgomery Multiplier for Finite Field GF(2^m) Based on Pentanomials", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 21, NO. 2, FEBRUARY 2013