2005년 8월

교육학석사(전기・전자・통신교육전공)학위논문

합성체를 이용한 유한체의 역원 계산 알고리즘 구현

조선대학교 교육대학원

전기・전자・통신교육전공

신 자 영

합성체를 이용한 유한체의 역원 계산 알고리즘 구현

Implementation of Inverse Algorithm in Finite Fields using Composite Fields

2005년 8월

조선대학교 교육대학원

전기·전자·통신교육전공

신 자 영

합성체를 이용한 유한체의 역원 계산 알고리즘 구현

지도교수 이 강 현

이 논문을 교육학석사(전기·전자·통신 교육전공)학위 청구논문으로 제출합니다.

2005년 4월

조선대학교 교육대학원

전기·전자·통신교육전공

신 자 영

신자영의 교육학 석사학위 논문을 인준합니다.

심사위원장 조선대학교 교수 朴 種 伯 인

심사위원 조선대학교 교수 朴 暢 均 인

심사위원 조선대학교 교수 李 康 鉉 인

2005년 6월

조선대학교 교육대학원

Table of Content

List of Table

List of Figures

ABSTRACT

Implementation of Inverse Algorithm in Finite Fields using Composite Fields

SHIN, Ja-young
Advisor : Prof. RHEE, Kang Hyeon, Ph.D.
Major in Electricity, Electronics and Communication Education
Graduate School of Education, Chosun University

유한체(Finite field or Galois field)는 스위칭 이론, 디지털 신호처리 및 화상처리, 디지털 통신의 암호화 및 해독화를 요하는 보안 통신 등에서 많이 응용되고 있다.

특히 오류정정부호 중 BCH 부호나 Reed-Solomon 부호와 같은 블록부호는 유한체 상에서 정의되며, CD(Compact Disc), DAT(Digital Audio Tape), 타원곡선 알고리즘 등의 부호화 및 복호화에 $GF(2^N)$의 산술연산이 적용되어진다. 현재 많은 응용분야에서 유한체 연산의 실시간 처리를 요하므로 유한체 연산을 위한 전용 하드웨어 설계가 필요하게 되었고 이에 대한 많은 연구가 행하여지고 있다.

유한체는 사칙연산이 정의되는 유한개의 연소를 갖는 필드이며 모든 소수 P와 양수 N에 대해서 P^N개의 원소를 갖는 하나의 필드만이 존재한다. 이 필드를 유한체 필드라고 부르며 $GF(P^N)$으로 나타낸다. 유한체 상에서의 연산은 가·감산과 승산, 그리고 제산이 있으나 디지털 시스템은 유한체의 개수가 2의 승수인 $GF(2^N)$에서 이루어진다. 유한체의 역원의 계산은 크게 유한체 제산기를 이용하는 방법과 승산기를 이용하는 방법으로 나누어진다. 제산기를 이용하는 방법은 빠른 동작 속도를 가지나 하드웨어의 면적이 커지며, 승산기를 이용한 방법은 하드웨어의 면적은

작아지나 계산에 많은 시간이 소모된다. 본 논문에서는 합성체(Composite Fields)를 이용하여 *GF(256)*의 유한체의 역원을 계산할 수 있는 알고리즘을 제시하고 이를 하드웨어로 구현하여 현재 사용되어지는 'Itoh and Tsujii' 하드웨어 구조와 하드웨어 면적 및 계산 속도의 성능을 비교 하였다. 또한 AES의 SubBytes 블록에 이를 삽입하여 Altera FLEX10K FPGA 에뮬레이터 보드에 구현하여 제시된 알고리즘의 회로가 정상적으로 동작함을 확인하였다.

Chapter 1. Introduction

Finite fields (or Galois fields) are applied frequently in security communication needing digital communication coding and interpretation such as switching theorem, digital signal processing and image processing.

Especially, block codes such as BCH code and Reed-Solomon code among error correcting codes are defined with finite fields. Arithmetic operation of $GF(2^n)$ is used for the encoding and decoding of CD (compact disc), DAT(digital audio tape), and elliptical curve cryptography. In application fields, many researches are underway for designing hardwares for using finite field operation is needed since processing finite field operation is needed in real time[1,2].

A finite field is a field having a finite number of elements with four fundamental rules of arithmetics and exists in the field having only P^N elements for prime number P and positive number N of all elements. This finite field is expressed as $GF(P^N)$. Operations on a finite field include addition, subtraction, multiplication and division. However, a digital system is achieved at $GF(2^N)$ where the number of finite fields is 2, i.e., within multiplier. Calculation of multiplicative inverse of finite field is divided largely into two methods, i.e., the method using a finite field divider and the method using a multiplier. The former method is fast but needs a large area of hardware. The latter method needs a smaller area of hardware but needs much time. In this paper, we proposed an algorithm that could calculate $GF(256)$ multiplicative inverse of finite field using composite fields, implemented this algorithm in a hardware, and compare this hardware with the currently used 'Itoh and Tsujii' hardware in respect to structure, area and computation time. Furthermore, this hardware was inserted into the AES SubBytes block to show on FPGA

emulator board to confirm that the circuit of proposed algorithm was normally functioning. In this paper, the calculation of algorithm multiplicative inverse is examined in Chapter 3 and the calculation of algorithm multiplicative inverse was implemented using the proposed composite field in Chapter 4. In Chapter 5, we compare the circuit implementation, activity characteristic, and 'Itoh and Tsujii' function. The conclusion is drawn in Chapter 6.

Chapter 2. Theoretical Background

Finite fields are the general starting point for the constructions of many combinatorial structures. It will be important to know the fundamentals concerning these fields in order to investigate combinatorial structures and related areas of combinatorial interest. Unfortunately, the area of field theory is rather large and it would be impossible for us to cover it in detail and still have time to work with the results. In the interest of conserving time, we will present the elements of general field theory without proofs and only prove statements when we turn our attention specifically to finite fields.

It will be assumed that you are familiar with the definition of a field, the definition and basic properties of vector spaces and the fact that the integers modulo a prime p form a field (a finite one), which we will denote by $GF(p)$ (standing for the Galois Field of order p).

2.1 General Field Theory

Let L be a field containing a subset K, which is itself a field under the operations inherited from L. Then L is called an extension of K, and K a subfield of L. Every field has a smallest subfield, called the prime subfield, which is isomorphic to either the field of rationals Q, in which case we say that it has characteristic zero, or to a $GF(p)$ for some prime p, in which case we say that it has characteristic p. We shall denote the characteristic of an arbitrary field K by char K.

If L is an extension of K (denoted $L > K$), and p is an element of L but not an element of K, then the smallest field containing both K and p will be denoted by $K(p)$ and is a subfield of L. Similarly, the smallest field containing

K and the elements $p1, p2, \ldots, pn$ in $L > K$ will be written as $K(p1, p2, \ldots, pn)$. Any extension field L of K can be viewed as a vector space over K and the dimension of this vector space is called the degree of L over K, and is denoted by $[L:K]$. If the vector space is finite dimensional we say that L is a finite extension of K. If L is a finite extension of K and M is a finite extension of L, then $[M:K] = [M:L][L:K]$.

Denote by $K[x]$ the ring of polynomials over K in the variable x. $K[x]$ is a principal ideal domain (an ideal is a subring that is closed under multiplication, a principal ideal is an ideal generated by a single element, and a principal ideal domain is a commutative ring with unity all of whose ideals are principal). A polynomial in $K[x]$ is said to be monic if the coefficient of the highest power in x is unity. It is irreducible if it is not the product of two nonscalar polynomials in $K[x]$. If $f(x)$ is an irreducible polynomial in $K[x]$, then any zero (i.e., root) of $f(x)$ is not in K and so there is a smallest extension field L of K that contains it. Furthermore, L is isomorphic to the quotient field $K[x]/\langle f(x)\rangle$, where $\langle f(x)\rangle$ denotes the principal ideal of $K[x]$ generated by $f(x)$. If the irreducible polynomial $f(x)$ is of degree n, then $[L:K] = n$. Furthermore, if p is the zero of $f(x)$ in question, then $L = K(p)$ and the elements $1, p, p2, \ldots, pn-1$ form a basis of L over K.

Before we continue, let us illustrate these ideas with a well-known example. The field C of complex numbers is usually described in one of two ways, either as the set of numbers $(a + bi)$ where a and b are real numbers and $i = \sqrt{-1}$, or if you prefer not to mention the imaginary number i, C can be described as the set of all pairs (a, b) of real numbers where addition of two pairs is the usual component wise addition and multiplication of two pairs is defined by $(a,b)(c,d) = (ac - bd, ad + bc)$. The second version is of course viewing C as a vector space of dimension 2 over the real numbers R. Let us

see how we would construct C starting with the subfield R. Now $K = R$ and $K[x]$ is the ring of all polynomials with real coefficients. The polynomial $x2 + 1$ is monic since the coefficient of x^2 is 1 and it is irreducible over R since it cannot be factored into polynomials with only real coefficients. The principal ideal $<x^2+1>$ consists of all polynomials that have $x2 + 1$ as a factor. The quotient field structure, $R[x]/<x^2+1>$ is obtained by taking each polynomial in $R[x]$ and dividing it by $x2 + 1$. The polynomials that have the same remainder after division form equivalence classes, which are the elements of the quotient field. The different possible remainders are the polynomials $a + bx$, where a, b in R, and we identify the equivalence classes with these remainders. The association $a + bx$ iff $a + bi$ clearly shows that the quotient field and C are isomorphic. Now, let p be a zero of $x2 + 1$, i.e. $p2 + 1 = 0$, so $p = \sqrt{-1} = i$ which is not an element of R. $\{1, i\}$ forms a basis for C over R since every element of C can be written as $a(1) + b(i)$ with a, b in R and 1 and i are easily seen to be linearly independent. Using this basis, we can identify the elements of C with their coefficient vectors, i.e. $a + bi$ iff (a, b) to get the second representation as a vector space of dimension 2 (notice the highest power of x in the irreducible polynomial).

2.2 Finite Fields

Fields are abstractions of familiar number systems (such as the rational numbers Q, the real numbers R, and the complex numbers C) and their essential properties. They consist of a set F together with two operations, addition (denoted by $+$) and multiplication (denoted by $\cdot$), that satisfy the usual arithmetic properties:

(i) $(F, +)$ is an abelian group with (additive) identity denoted by 0.

(ii) $(F\backslash\{0\}, \cdot)$ is an abelian group with (multiplicative) identity denoted by 1.

(iii) The distributive law holds: $(a+b)\cdot c = a\cdot c + b\cdot c$ for all $a, b, c \in F$.

If the set F is finite, then the field is said to be finite.

This section presents basic facts about finite fields. Other properties will be presented throughout the book as needed.

Field operations

A field F is equipped with two operations, addition and multiplication. Subtraction of field elements is defined in terms of addition: for $a, b \in F$, $a - b = a + (-b)$ where $-b$ is the unique element in F such that $b+(-b) = 0$ ($-b$ is called the negative of b.) Similarly, division of field elements is defined in terms of multiplication: for $a, b \in F$ with $b \neq 0$, $a/b = a \cdot b^{-1}$ where b^{-1} is the unique element in F such that $b \cdot b^{-1} = 1$. (b^{-1} is called the inverse of b.)

Existence and uniqueness

The order of a finite field is the number of elements in the field. There exists a finite field F of order q if and only if q is a prime power, i.e., $q = p^m$ where p is a prime number called the characteristic of F, and m is a positive integer. If $m = 1$, then F is called a prime field. If $m \geq 2$, then F is called an extension field. For any prime power q, there is essentially only one finite field of order q informally, this means that any two finite fields of order q are structurally the same except that the labeling used to represent the field elements may be different. We say that any two finite fields of order q are isomorphic and denote such a field by F_q.

Prime fields

Let p be a prime number. The integers modulo p, consisting of the integers $\{0, 1, 2, ..., p-1\}$ with addition and multiplication performed modulo p, is a finite field of order p. We shall denote this field by F_p and call p the modulus of F_p. For any integer a, $a \bmod p$ shall denote the unique integer remainder r, $0 \leq r \leq p-1$, obtained upon dividing a by p this operation is called reduction modulo p.

Binary fields

Finite fields of order 2^m are called binary fields or characteristic-two finite fields. One way to construct F_{2^m} is to use a polynomial basis representation. Here, the elements of F_{2^m} are the binary polynomials (polynomials whose coefficients are in the field $F2 = \{0,1\}$) of degree at most $m-1$:

$$F_{2^m} = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots a_2z^2 + a_1z + a_0 : a_i \in \{0,1\}\}$$

An irreducible binary polynomial $f(z)$ of degree m is chosen (such a polynomial exists for any m and can be efficiently found). Irreducibility of $f(z)$ means that $f(z)$ cannot be factored as a product of binary polynomials each of degree less than m. Addition of field elements is the usual addition of polynomials, with coefficient arithmetic performed modulo 2. Multiplication of field elements is performed modulo the reduction polynomial $f(z)$. For any binary polynomial $a(z)$, $a(z) \bmod f(z)$ shall denote the unique remainder polynomial $r(z)$ of degree less than m obtained upon long division of $a(z)$ by $f(z)$ this operation is called reduction modulo $f(z)$.

Extension fields

The polynomial basis representation for binary fields can be generalized to all extension fields as follows. Let p be a prime and $m \geq 2$. Let $F_p[z]$ denote the set of all polynomials in the variable z with coefficients from F_p. Let $f(z)$, the reduction polynomial, be an irreducible polynomial of degree m in $F_p[z]$–such a polynomial exists for any p and m and can be efficiently found. Irreducibility of $f(z)$ means that $f(z)$ cannot be factored as a product of polynomials in $F_p[z]$ each of degree less than m. The elements of F_{p^m} are the polynomials in $F_p[z]$ of degree at most $m-1$:

$$F_{p^m} = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots a_2z^2 + a_1z + a_0 : a_i \in F_p\}$$

Addition of field elements is the usual addition of polynomials, with coefficient arithmetic performed in F_p. Multiplication of field elements is performed modulo the polynomial $f(z)$.

Subfields of a finite field

A subset k of a field K is a subfield of K if k is itself a field with respect to the operations of K. In this instance, K is said to be an extension field of k. The subfields of a finite field can be easily characterized. A finite field F_{p^m} has precisely one subfield of order p^l for each positive divisor l of m the elements of this subfield are the elements $a \in F_{p^m}$ satisfying $a^{p^l} = a$. Conversely, every subfield of F_{p^m} has order p^l for some positive divisor l of m.

Bases of a finite field

The finite field F_{q^n} can be viewed as a vector space over its subfield F_q. Here, vectors are elements of F_{q^n}, scalars are elements of F_q, vector addition is the addition operation in F_{q^n}, and scalar multiplication is the multiplication in F_{q^n} of F_q-elements with F_{q^n}-elements. The vector space has dimension n and has many bases.

If $B = \{b_1, b_2, \cdots, b_n\}$ is a basis, then $a \in F_{q^n}$ can be uniquely represented by an n-tuple $(a_1, a_2, \cdots a_n)$ of F_q-elements where $a = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$. For example, in the polynomial basis representation of the field Fpm described above, F_{p^m} is an m-dimensional vector space over F_p and $\{z^{m-1}, z^{m-2}, \cdots, z, 1\}$ is a basis for F_{p^m} over F_p.

Multiplicative group of a finite field

The nonzero elements of a finite field F_q, denoted F_q^*, form a cyclic group under multiplication. Hence there exist elements $b \in F_q^*$ called generators such that:

$$F_q^* = \{b^i : 0 \le i \le q-2\}$$

The order of $a \in F_q^*$ is the smallest positive integer t such that $a^t = 1$. Since F_q^* is a cyclic group, it follows that t is a divisor of $q-1$.

2.3 Prime Field Arithmetic

This section presents algorithms for performing arithmetic in the prime field F_p. Algorithms for arbitrary primes p are presented. The reduction step can be accelerated considerably when the modulus p has a special form. Efficient reduction algorithms for the NIST prime.

The algorithms presented here are well suited for software implementation. We assume that the implementation platform has a W-bit architecture where W is a multiple of 8. Workstations are commonly 64 or 32-bit architectures. Low-power or inexpensive components may have smaller W, for example, some embedded systems are 16-bit and smartcards may have $W = 8$. The bits of a W-bit word U are numbered from 0 to $W-1$, with the rightmost bit of U designated as bit 0.

The elements of F_p are the integers from 0 to $p-1$. Let $m = \lceil log_2 p \rceil$ be the bit length of p, and $t = m/W$ be its wordlength. Fig. 1 illustrates the case where the binary representation of a field element a is stored in an array $A = (A[t-1], ..., A[2], A[1], A[0])$ of t W-bit words, where the rightmost bit of $A[0]$ is the least significant bit.

A[t-1]	···	A[2]	A[1]	A[0]

Fig. 1 Representation of $a \in F_p$ as an array A of W-bit words.

Hardware characteristics may favour approaches different from those of the algorithms and field element representation presented here.

Addition and subtraction

Algorithms for field addition and subtraction are given in terms of corresponding algorithms for multi-word integers. The following notation and terminology is used. An assignment of the form "$(\epsilon, z) \leftarrow w$" for an integer w is understood to mean

$z \leftarrow w \bmod 2^w$, and

$\epsilon \leftarrow 0$ if $w \in [0, 2^w)$, *otherwise* $\epsilon \leftarrow 1$

If $w = x + y + \epsilon'$ for $x, y \in [0, 2^w)$ and $\epsilon' \in \{0, 1\}$, then $w = \epsilon 2^w + z$ and ϵ is called the carry bit from single-word addition (with $\epsilon = 1$ if and only if $z < x + \epsilon'$).

Integer multiplication

Field multiplication of $a, b \in F_p$ can be accomplished by first multiplying a and b as integers, and then reducing the result modulo p. Algorithms are elementary integer multiplication routines which illustrate basic operand scanning and product scanning methods, respectively. In both algorithms, (UV) denotes a $(2W)$-bit quantity obtained by concatenation of W-bit words U and V.

The calculation $C[i+j] + A[i] \cdot B[j] + U$ is called the inner product operation. Since the operands are W-bit values, the inner product is bounded by $2(2^w - 1) + (2^w - 1)^2 = 2^{2w} - 1$ and can be represented by (UV).

Reduction

For modulo p that are not of special form, the reduction z mod p can be an

expensive part of modular multiplication. Since the performance of elliptic curve schemes depends heavily on the speed of field multiplication, there is considerable incentive to select modulo, such as the NIST-recommended primes, that permit fast reduction. In this section, we present only the reduction method of Barrett and an overview of Montgomery multiplication.

The methods of Barrett and Montgomery are similar in that expensive divisions in classical reduction methods are replaced by less-expensive operations. Barrett reduction can be regarded as a direct replacement for classical methods; however, an expensive modulus-dependent calculation is required, and hence the method is applicable when many reductions are performed with a single modulus. Montgomery's method, on the other hand, requires transformations of the data. The technique can be effective when the cost of the input and output conversions is offset by savings in many intermediate multiplications, as occurs in modular exponent.

Note that some modular operations are typically required in a larger framework such as the signature schemes, and the modulo involved need not be of special form.

2.4 Constructing Finite Fields

We will illustrate the above material by actually constructing some finite fields.

GF(9)

Since $9 = 3^2$, the prime field must be *GF(3)* whose elements we will represent by *0, 1* and *2*, and where addition and multiplication are done modulo *3*. We seek an extension of degree *2* over the prime field, so our first task is

to find a monic irreducible polynomial of degree *2* in *GF(3)[x]*. For large field this can be a difficult assignment, but for small fields it is not too bad. While there are some theorems that may help, the brute force procedure is effective if the prime field is small. We can in fact easily list all of the monic quadratics in this ring, they are:

x^2
$x^2 + 1$
$x^2 + 2$
$x^2 + x$
$x^2 + x + 1$
$x^2 + x + 2$
$x^2 + 2x$
$x^2 + 2x + 1$
$x^2 + 2x + 2$

Now the problem is to find the irreducible ones in this list. Clearly, any polynomial without a constant term is factorable (x is a factor), so the first, fourth and seventh can immediately be crossed out. For the remaining six polynomials, we may opt for one of two procedures. We could take each in turn and substitute all the field elements for x, if none of these substitutions evaluates to zero, the polynomial is irreducible (i.e., it has no root in the field). So, for example, substituting in $x^2 + 2$ gives the values $0^2 + 2 = 2$, $1^2 + 2 = 0$ and $2^2 + 2 = 0$, thus $x^2 + 2$ factors, in fact $x^2 + 2 = (x + 1)(x + 2)$. On the other hand, the same procedure for $x^2 + 1$ gives $0^2 + 1 = 1$, $1^2 + 1 = 2$ and $2^2 + 1 = 2$ and so $x^2 + 1$ is irreducible. The second possible procedure is to take all the linear factors (in this case, because we want quadratic products)

and multiply them in all possible pairs to get a list of all the factorable quadratics, removing these from our list leaves all the irreducible quadratics. So,

$$(x+1)(x+1) = x^2 + 2x + 1$$
$$(x+1)(x+2) = x^2 + 2$$
$$(x+2)(x+2) = x^2 + x + 1$$

Implying that, x^2+1, x^2+x+2 and x^2+2x+2 are the only irreducible monic quadratic polynomials in *GF(3)[x]*. We could now choose any one of these letting p be a zero of the chosen polynomial and write out the elements of *GF(9)* in its vector form representation using the basis *{1, p}*. This however does not give us the most useful representation of the field. Rather, we will use the fact that the multiplicative group of the field is cyclic, so if we can find a primitive element (i.e., a generator of the cyclic group) we will have a handy representation of the elements. Now the primitive elements are to be found among the roots of the irreducible polynomials (they cannot be elements of the prime field). The cyclic group we are after has order *8*, so not every root need be primitive. For example, letting p be a root of x^2+1, i.e., $p^2+1=0$, so $p^2=2$, we can write out the powers of p.

$$p^1 = p,\ p^2 = 2,\ p^3 = 2p,\ p^4 = 2p(p) = 2p^2 = 2(2) = 1$$

And so p has order *4* and does not generate the cyclic group of order *8*, i.e., p is not a primitive element. On the other hand, consider μ a root of the polynomial x^2+x+2, so that $u^2+u+2=0$ or $u^2=2u+1$. Now the powers of μ give us:

$u^1 = u$
$u^2 = 2u + 1$
$u^3 = u(2u + 1) = 2u^2 + u = 2(2u + 1) + u = 2u + 2$
$u^4 = 2u^2 + 2u = u + 2 + 2u = 2$
$u^5 = 2u$
$u^6 = 2u^2 = u + 2$
$u^7 = u^2 + 2u = 2u + 1 + 2u = u + 1$
$u^8 = u^2 + u = 2u + 1 + u = 1$

so μ is a primitive element and so we have represented the elements of *GF(9)* as the *8* powers of μ together with *0*. Notice also that the bolded terms on the right are all the possible terms that can be written as linear combinations of the basis {*1*, μ} over *GF(3)*. When working with finite fields it is convenient to have both of the above representations, since the terms on the left are easy to multiply and the terms on the right are easy to add. So for instance, if we wanted to calculate $(2u + 2)^3 + u + 2$, we would do so in this way, $(2u + 2)^3 = (u^3)^3 = u^9 = u$ and so $(2u + 2)^3 + u + 2 = u + u + 2 = 2u + 2 = u^3$

GF(8)

Since $8 = 2^3$, the prime field is *GF(2)* and we need to find a monic irreducible cubic polynomial over that field. Since the coefficients can only be *0* and *1*, the list of irreducible candidates is easily obtained.

$x^3 + 1$
$x^3 + x + 1$
$x^3 + x^2 + 1$
$x^3 + x^2 + x + 1$

Now substituting *0* gives *1* in all cases, and substituting *1* will give *0* only if there are an odd number of x terms, so the irreducible cubics are just x^3+x+1 and x^3+x^2+1. Now the multiplicative group of this field is a cyclic group of order *7* and so every nonidentity element is a generator. Letting μ be a root of the first polynomial, we have $u^3+u+1=0$, or $u^3=u+1$, so the powers of μ are:

$u^1=u$
$u^2=u^2$
$u^3=u+1$
$u^4=u^2+u$
$u^5=u^2+u+1$
$u^6=u^2+1$
$u^7=1$

Now suppose we had chosen a root of the second polynomial, say , p. We would then have $p^3=p^2+1$ and the representation would be given by

$p^1=p$
$p^2=p^2$
$p^3=p^2+1$
$p^4=p^2+p+1$
$p^5=p+1$
$p^6=p^2+p$
$p^7=1$

We know that these two representations must be isomorphic, show that the isomorphism is induced by $u \text{---}> p^6$.

Chapter 3. Calculation of multiplicative inverse in the algorithm

Typical methods of calculating inverses in finite fields include the Fermat theorem, the method using extended Euclid algorithms, and the method using finite field multipliers[3,4]. There is also the method of using composite fields as proposed in this study.

3.1 Fermat theorem

The Fermat theorem is expressed as the Equation (1) when the random numbers a and p have relation of relatively prime.

$$\begin{cases} a^p = a \mod p, a \in GF(p) \\ a^{-1} = a^{p-2} = a^{2^m-2} \mod 2^m \end{cases}, p = 2^m \qquad (1)$$

When an inverse is calculated with Fermat theorem, the multiplication of $m-1$ time and square of $m-1$ times are needed for operation on $GF(2^m)$.

3.2 Extended Euclid algorithm

Equation (2) shows the process of calculating the greatest common dividers of two polynomials, i.e., $a(x)$ and $b(x)$, $\{deg(a(x)) < deg(b(x))\}$ using Euclid algorithm.

$$
\begin{aligned}
& r_{-1}(x) = b(x), \qquad r_0(x) = a(x) \\
& r_{-1}(x) = q_0(x) r_0(x) + r_1(x) \\
& r_0(x) = q_1(x) r_1(x) + r_2(x) \\
& \quad \vdots \qquad\qquad \vdots \\
& r_{k-1}(x) = q_k(x) r_k(x) + r_{k+1}(x)
\end{aligned} \tag{2}
$$

when GCD is in Equation (2), the Equation (3) results according to extended Euclid algorithm.

$$
\begin{aligned}
& s(x) b(x) + t(x) a(x) = g(x), \\
& g(x) = GCD(a(x), b(x))
\end{aligned} \tag{3}
$$

In addition, $s(x)$ and $t(x)$ in Equation (3) can be calculated as the repetitive calculation as in Equation (4).

$$
\begin{aligned}
& s_{-1}(x) = 1,\ s_0(x) = 0 \\
& t_{-1}(x) = 0,\ t_0(x) = 1 \\
& s_i(x) = q_{i-1}(x) s_{i-1}(x) + s_{i-2}(x),\ (-1 \le i \le k+1) \\
& t_i(x) = q_{i-1}(x) t_{i-1}(x) + t_{i-2}(x)
\end{aligned} \tag{4}
$$

$q_i(x)$ in Equation (4) is calculated with Equation (2). $t_k(x)$ calculated repeatedly until $t(x)$ in Equation (2) reaches $r_{k+1} = 0$, becomes $b(x)$, and $t_{k-1}(x)$ becomes $a^{-1}(x)$.

Here, Equation (3) is rewritten to become

$$
t(x) a(x) = g(x) \bmod b(x) \tag{5}
$$

At this time, since $b(x)$ is an irreducible polynomial, it is always $g(x) = 1$.

Thus, the result becomes

$$t(x)a(x) = 1 \bmod b(x)$$
$$t(x) = \frac{1}{a(x)} \bmod b(x) \quad (6)$$

3.3 Calculation of the inverse using a multiplier of the $GF(2^8)$ finite field

When the finite filed element y is inputted in the $GF(2^8)$ field, the inverse of y is achieved as in Equation (7).

$$y^{2^8-1} = y^{255} = 1$$
$$y^{-1} = y^{-1} \cdot y^{255} = y^{254} \quad (7)$$

We compared the circuit area and computation time of proposed algorithm with 'Itoh and Tsujii' circuit applied in Equation (7).

3.4 Composite Fields

Using the second composite field, the algorithm calculating the multiplicative inverse of finite field was proved by Cristof Paar[5]. When two finite fields ζ and δ have $\delta=\zeta^{-1}$ relationship, we could calculate the multiplicative inverse of finite field according to Equation (8).

$$
\begin{aligned}
\zeta &= Z_1 r + Z_0 \\
\delta &= D_1 r + D_0 \\
D_0 &= (Z_1 A + Z_0) F^{-1} \qquad (8) \\
D_1 &= Z_1 F^{-1} \\
F &= Z_1^2 B + Z_1 Z_0 A + Z_0^2
\end{aligned}
$$

We implemented the inverse algorithm in finite field using Equation (8). For hardware design, the multiplicative inverses of Christof Paar[1] were referred and $GF(2^4)$ multiplicative inverses were designed. Refereeing to the Mastrovito structure[6], finite field multiplier was designed.

Chapter 4. Inverse algorithm using composite field

$GF(256)$ in finite fields can be $GF(2^8)$, $GF(4^4)$ or $GF(16^2)$. In this study, $GF(16^2)$ was used as the composite field of $GF(p^{mm})$[7,8].

4.1 Process of calculating finite field inverse

Equation (9) cane be used when $\zeta \in GF(2^4)^2)$.

$$\zeta = Z_1 r + Z_0 \qquad (9)$$

Here, r is the root of the irreducible polynomial $P(x)$ of $GF(16^2)$ and satisfies $Z_1 \cdot Z_2 \in GF(2^m)$. Equation (10) is the irreducible polynomial $P(x)$.

$$P(x) = x^2 + Ax + B \qquad (10)$$

From Equations (9) and (10), ζ^{-1} can be driven as Equation (11).

$$\begin{aligned} &\zeta^{-1} = \delta = D_1 r + D_0 \\ &\zeta \cdot \delta = 1 = D_1 Z_1 r^2 + (D_1 Z_0 + D_0 Z_1) r + D_0 Z_0 \\ &P(r) = 0,\ r^2 = Ar + B \end{aligned} \qquad (11)$$

Equations (12) and (13) show the process of calculating the multiplicative inverses of finite fields, i.e., D_1 and D_0 and driven from Equation (11).

$$(D_1Z_0 + D_0Z_1 + AD_1Z_1)r + (D_0Z_0 + BD_1Z_1) = 0r + 1 \quad (12)$$

$$\begin{aligned} &D_1Z_0 + D_0Z_1 + AD_1Z_1 = 0 \\ &D_0Z_0 + BD_1Z_1 = 1 \\ &D_1 = D_0Z_1(Z_0 + AZ_1)^{-1} \\ &D_0 = D_1Z_1^{-1}(Z_0 + AZ_1)^{-1}) = 1 \\ &D_1Z_1^{-1}(Z_0^2 + AZ_1Z_0 + BZ_1^2) = 1 \end{aligned} \quad (13)$$

As a result, the computation of multiplicative inverses in finite fields using composite fields is driven as Equation (14).

$$\begin{aligned} &D_1 = Z_1 \cdot F^{-1},\ \ D_0 = (Z_0 + AZ_1) \cdot F^{-1} \\ &F = Z_0^2 + AZ_1Z_0 + BZ_1^2)^{-1} \end{aligned} \quad (14)$$

4.2 Circuit design of the proposed inverse algorithm

From Equation (14), the computation algorithm of multiplicative inverse of finite field was implemented as in Fig. 2. Here, the irreducible polynomial of compute field used was $P(x) = x^2 + x + \beta^{14}$. β^{14} is calculated to be $\beta^3 + 1 = y^3 + 1$.

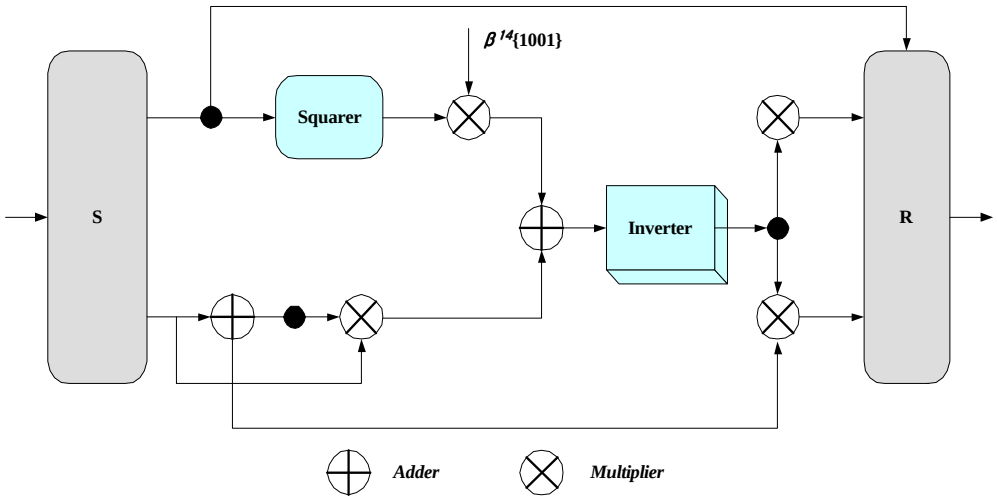


Fig. 2. Circuit design for inverse calculation using composite fields

In Fig. 2, S block is the arrange block changing the input extension field into composite field. R block is the arrange block changing the composite field into the extension field. in which $R = S^{-1}$[8]. Fig. 3 shows the values of S and R blocks.

$$S = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Fig. 3. Block coefficient of S and R

4.3 Simulation

When the results driven were coded using VHDL, the output values were

normal. Comparison with the resultant values was done using the inverse of finite field multiplier explained in the inverse table of AES S-box and Chapter. Fig. 4 show the simulation result, showing the same inverse value.

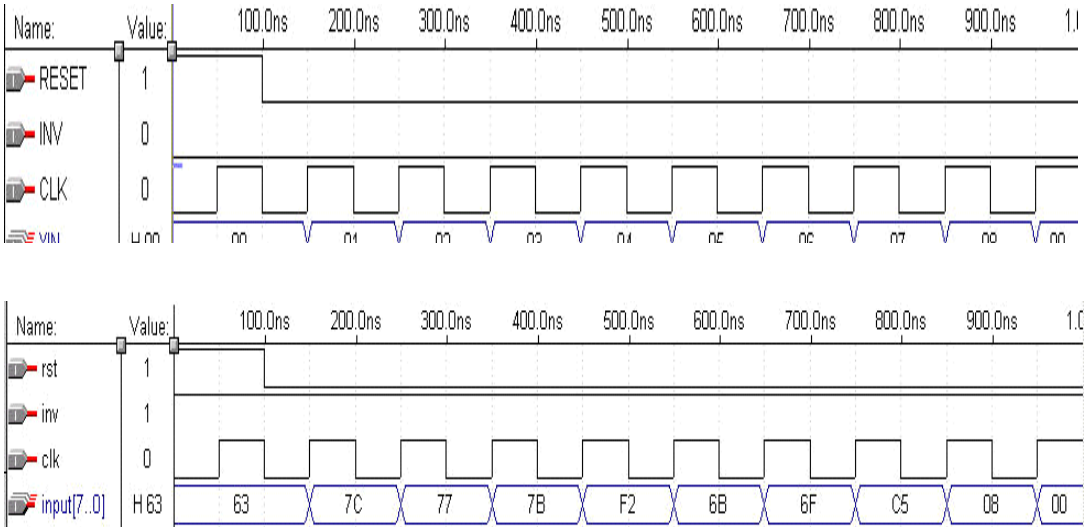


Fig. 4. Values from simulation

Fig. 5 is the gatelevel circuit synthesized using core_slow.db in Synopsys. The overall cell area was 4593.87.

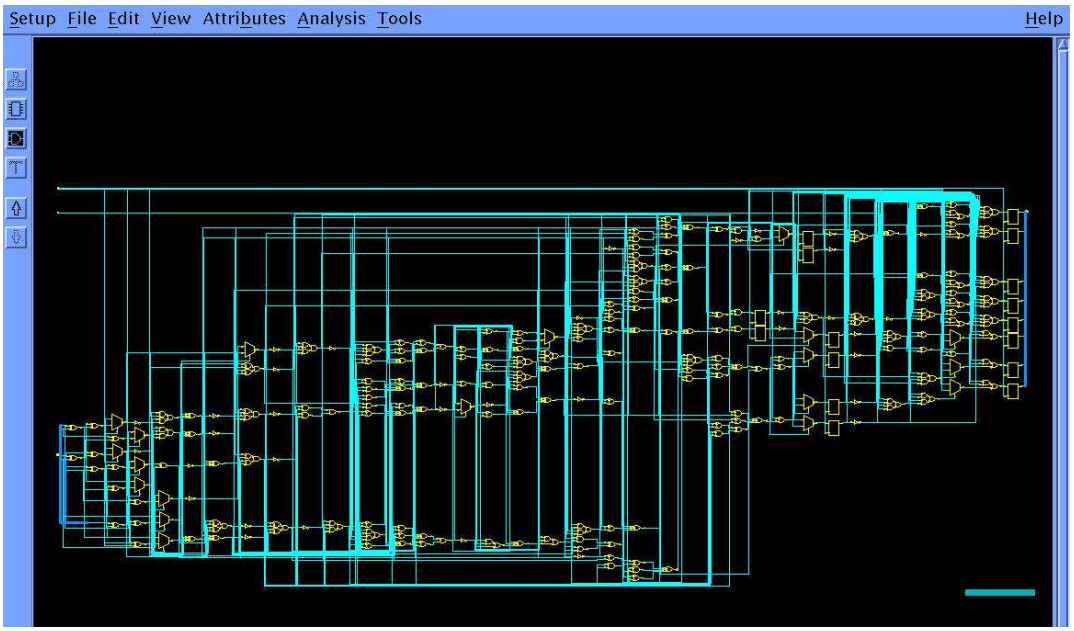


Fig. 5. Gate level circuit

4.4 Measuring function

The inverse computation algorithm proposed in this study and 'Itoh and Tsujii' algorithm are compared in Table 1.

Table 1. Results of circuit synthesis and comparison of function

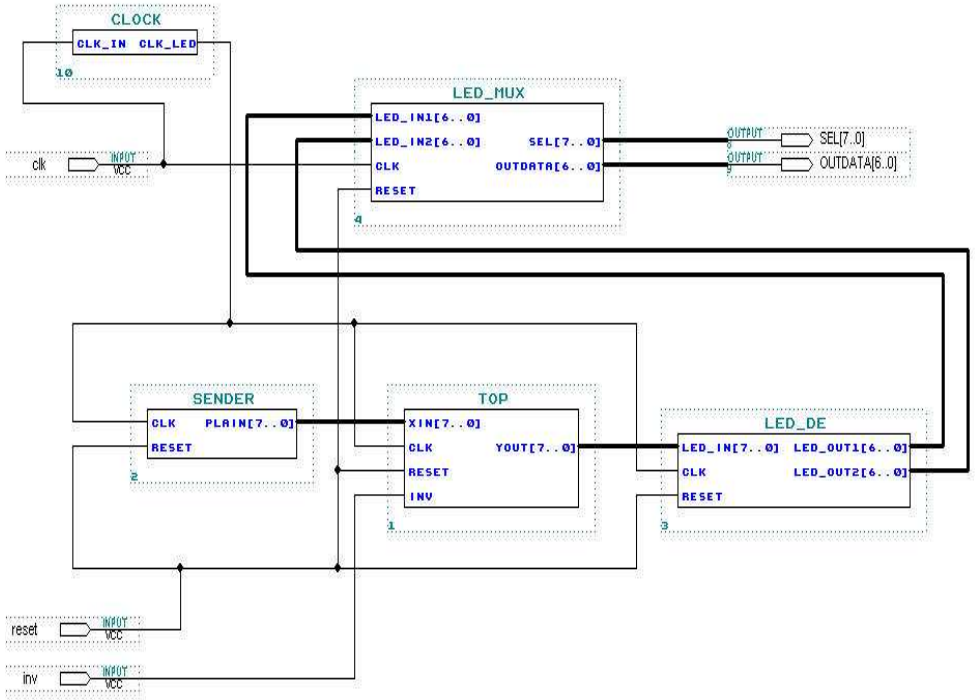
Design Architecture	Cell Area	Data Arriver Time
'Itoh and Tsujii'	12030.64	19.960[ns]
Proposed Design	4593.87	15.155[ns]

When the function was compared, the design proposed in this study showed a decrease of 61.6% in cell area and an increase of 24.97% in computation time

compared with 'Itoh and Tsujii' design.

Chapter 5. Circuit implementation using FPGA

The inverse algorithm proposed in Chapter 4 was implemented on Altera FLEX10K. Activity of implemented algorithm was confirmed by implementing FPGA circuit rather than the rom table by designing AES SubBytes Transform circuit. The circuit design is shown in Fig. 6.



CLOCK : Clock division circuit(40:1)

Top : SubBytes(Inverse SubBytes) Transform circuit

LED_MUX : Seven segment control switch

LED_DE : Seven segment signal decode

SENDER : Input signal data

Fig. 6. FPGA Circuit design

Fig. 7 shows the picture actually realized on FPGA emulator board.

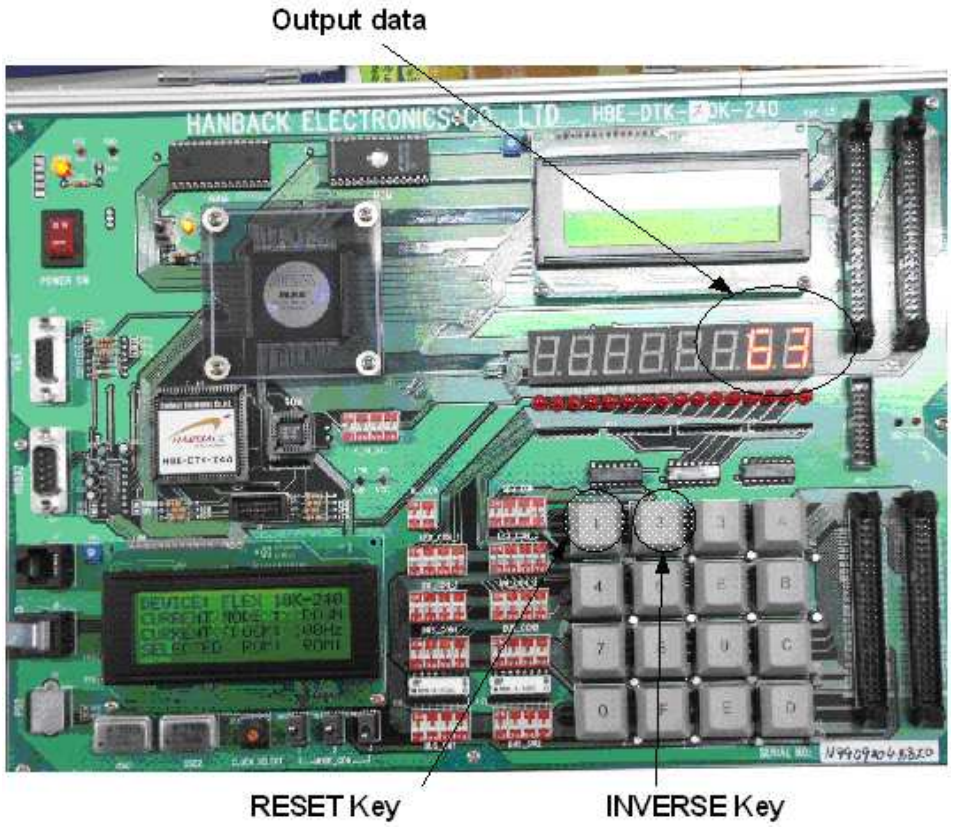


Fig. 7. FLEX10K FPGA verification circuit

In Fig. 7, the output data in the ouput data portion is outputted in 7 segments and the circuit initializes through the RESET Key. The INVERSE Key is the control preventing the circuit from inverse transformation.

Chapter 6. Conclusion

We proposed an optimal algorithm to calculate the multiplicative inverse of a finite field using composite fields and implement the result on Altera FLEX10K FPGA board. In this paper, we could implement the circuit with AND and XOR gate. The result of circuit synthesis showed that cell area was 4593.87, showing a decrease of 61.8% in cell area compared with 'Itoh and Tsujii' algorithm, which uses multipliers of finite fields, and the computation time was 15.155[ns], which improved by 24.07%. Furthermore, in order to confirm whether the multiplicative inverse of finite field was functioning normally, normal function was confirmed using AES SubBytes Transform designed to substitute for ROM table used for inverse calculation.

References

[1] C. Paar. Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields. Ph.D thesis,Institute for Experimental Mathematics, University of Essen, 1994

[2] Vincent Rijmen. Efficient implementation of the Rijndael S-box. http://www.esat.kuleuven.ac.be/ ~rijmen/rijndael/sbox.pdf, *2000.*

[3] Federal Information Processing Standards Publication 197, Nov. 26, 2001. Announcing the ADVANCED ENCRYPTION STANDARD.

[4] Michael Rosing, Implementing Elliptic Curve Cryptography, Manning, 1999.

[5] C. Paar, P. Fleischmann and S. Pedro, Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents, *IEEE Trans. on Computers.*, vol. 48, no. 10, pp. 1025-1034, Oct. 1999.

[6] E.D Mastrovito, "VLSI Architecture for Computation in Galois Field," PhD Thesis, PhD Dissertation, Linkping Univ., Linkping Sweden, 1991.

[7] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal basis", J. Society for Electronic Communication, pp. 31-36, 1986.

[8] Cillian O'Driscoll. Hardware implementation aspects of the Rijndael block cipher. Master's thesis, University Colleg.

[9] The original granddaddy in the area is: Dickson, Linear Groups (with an Exposition of the Galois Field Theory), Dover, 1958.

[10] Berlekamp, Algebraic Coding Theory, McGraw-Hill, New York 1968.

[11] Blake & Mullin, An Introduction to Algebraic and Combinatorial Coding Theory, Academic Press, 1976.

[12] Introduction to the Theory of Error-Correcting Codes, Wiley, 1982.

[13] Lidl & Niederreiter, Finite Fields, Vol. 20 in the Encyclopedia of

Mathematics and its Applications, Addison-Wesley, 1983.

[14] Lidl & Niederreiter, Introduction to Finite Fields and their Applications, Cambridge University Press, 1986.

[15] McEliece, Finite Fields for Computer Scientists and Engineers, Kluwer Academic Publishers, 1987.