情報保護學會論文誌
第18卷 第3號, 2008. 6

제약적인 환경에 적합한 유한체 연산기 구조 설계*

정 석 원†
목포대학교

Design of an Operator Architecture for Finite Fields in Constrained Environments*

Seok Won Jung
Mokpo National University

요 약

유한체 연산기는 생성 기약다항식과 원소의 표현 방법에 따라 효율성에 많은 영향을 받는다. 본 논문에서는 홀수 소수 p에 대한 확장체 $GF(p^n)$ 위의 곱셈에 대한 두 가지 직렬곱셈기를 제안한다. 기약 이항 다항식을 이용한 직렬 곱셈기는 (2n+5)개의 레지스터, 2개의 MUX, 2개의 GF(p) 곱셈기, 1개의 GF(p) 덧셈기를 사용하여 n^2+n 클럭 싸이클 이후에 곱셈결과를 얻는 구조이다. 기약 AOP를 이용한 직렬 곱셈기는 (2n+5)개의 레지스터, 1개의 MUX, 1개의 GF(p) 곱셈기, 1개의 GF(p) 덧셈기를 사용하여 n^2+3n+2 클럭 싸이클 이후에 곱셈결과를 얻는다.

ABSTRACT

The choice of an irreducible polynomial and the representation of elements have influence on the efficiency of operators for finite fields. This paper suggests two serial multiplier for the extention field $GF(p^n)$ where p is odd prime. A serial multiplier using an irreducible binomial consists of (2n+5) resisters, 2 MUXs, 2 multipliers of GF(p), and 1 adder of GF(p). It obtains the mulitplication result after n^2+n clock cycles. A serial multiplier using an AOP consists of (2n+5) resisters, 1 MUX, 1 multiplier of GF(p), and 1 adder of GF(p). It obtains the mulitplication result after n^2+3n+2 clock cycles.

Keywords : Finite Fields, Irreducible Binomials, All One Polynomials, Redundant Bases, Multiplier, Serial Architecture

Ⅰ. 서 론

1988년 제록스사의 Mark Weiser가 언제, 어디서나 사용자가 접속하여 원하는 정보와 서비스를 제공받을 수 있도록 컴퓨터를 실생활 환경에 편재시키는 유비쿼터스 컴퓨팅의 개념을 처음으로 소개하였다[14]. 최근에 유비쿼터스 환경에서의 컴퓨팅 기술이 급속도로 증가하고 있으며, RFID를 이용한 물류자동화, 셋톱박스를 이용한 홈네트워크, 차량자동항법장치를 이용한 텔레메틱스 등 여러 가지 서비스가 제안되어 실생활에 적용되고 있다.

유비쿼터스 환경이 도래함에 따라 다양한 디바이스의 개발과 이종 망간 데이터 통신이 빈번히 생기고 있어 이에 적합한 다양한 암호기술의 요구가 증대되고 있다. 그런데 기존의 유선 또는 무선 환경에서 인증과 전

접수일 : 2008년 4월 22일; 수정일 : 2008년 5월 16일;
채택일 : 2008년 5월 22일
* 이 논문은 2005년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임. (KRF-2005-003-C00028)
† 주저자, jsw@mokpo.ac.kr

자서명 등의 정보보호기술을 제공해 주는 인수분해와 이산대수문제 기반의 암호 프리미티브는 많은 수학적인 계산과 넓은 통신 대역폭을 요구하고 있다. 따라서 통신의 대역폭, 디바이스의 계산능력과 메모리 등이 제한되는 유비쿼터스 환경에 이러한 암호 프리미티브를 구현하는 것은 많은 어려움이 따른다.

유한체는 소프트웨어와 하드웨어적으로 간단하고 효율적인 연산 구조를 갖는다. 이러한 이유로 유한체는 암호화기법, 키 교환 프로토콜, 전자서명기법 등 많은 암호 알고리즘 및 프로토콜의 설계 기법에 사용되고 있다. 또한 유한체는 메모리와 시간의 교환을 고려한 다양한 구조로 구현할 수 있어 제약적인 환경의 디바이스에 소프트웨어와 하드웨어로 적용할 수 있는 가능성을 가지고 있다.

유한체는 크게 양의 정수 n과 홀수 소수 p에 대해 이진 확장체 $GF(2^n)$과 소수 확장체 $GF(p^n)$으로 나눌 수 있다. 유한체 $GF(2^n)$의 연산기는 덧셈이 독립적인 비트들의 XOR 연산으로 구현되는 장점으로 인해 병렬구조[1,9,13], 시스톨릭 어레이 구조[2], 직렬-병렬 구조[3], 직렬구조[4] 등에 대한 연구가 많이 이루어졌다. 최근 Bailey와 Paar[5]가 RISC 프로세서에서 소프트웨어로 고속으로 ECC(Elliptic Curve Cryptosystem)를 구현할 수 있는 최적 소수 확장체(OEF, Optimal Extention Fields)의 연산 방법을 제안하였다. 또한 소수 확장체 위의 패어링 함수를 이용한 암호 프리미티브에 대한 연구가 활발해지고 있다. 홀수 소수 p에 대한 유한체 $GF(p^n)$의 연산 효율성은 직접적으로 패어링 기반의 암호 프리미티브의 효율성에 영향을 미치기 때문에 이의 하드웨어 구현에 대한 연구가 활발하다[6-8].

소수 확장체 $GF(p^n)$에 대한 연구는 Bailey와 Paar[5]의 소프트웨어 기반에 이항 기약 다항식을 이용한 최적 소수 확장체에 대한 연구가 있었다. 패어링 기반의 연구는 GF(3) 위의 기약 삼항 다항식을 이용한 소수 확장체 $GF(3^m)$의 병렬 하드웨어 구조에 대한 연구가 활발하다. 본 논문에서는 제약적인 환경에 적합하도록 소수체 GF(p)의 연산기를 최소한으로 사용하여 면적의 효율성을 갖는 직렬구조의 소수 확장체의 연산 하드웨어 설계 구조를 두 가지 제안한다. 하나는 기약다항식 중에서 항의 개수가 가장 작은 기약 이항다항식과 다항식 기저를 이용한 직렬곱셈기 구조이고, 다른 하나는 모든 계수가 1인 기약 다항식(AOP, All One Polynomial)과 잉여기저를 이용한 직렬곱셈기 구조이다.

Ⅱ. 기약 이항다항식을 이용한 직렬 곱셈기

2.1 기약 이항다항식과 다항식 기저를 이용한 두 원소의 곱셈

홀수 소수 p와 양의 정수 n에 대해 유한체 GF(p)의 원소를 계수로 갖는 n차 기약다항식 q(x)의 한 근을 α라 하자. 그러면 GF(p) 위의 n차 확장체 $GF(p^n) = \{ a_0 + a_1\alpha^1 + \cdots + a_{n-1}\alpha^{n-1} \mid a_0, a_1, \cdots, a_{n-1} \in GF(p) \}$ 이다. 이때 어떤 n차 기약다항식을 선택하더라도 이들로 만든 유한체의 수학적 구조는 동형이다. 그러나 유한체 $GF(p^n)$의 원소들에 대한 연산의 효율성은 어떤 기약다항식을 선택했느냐에 많은 영향을 받는다.

유한체 GF(p) 위의 n차 기약 이항다항식(binomial)은 항의 개수가 2개이고 계수가 GF(p)의 원소인 기약다항식을 말한다. 즉, 기약 이항다항식은 적당한 $w \in GF(p)$에 대해 $f(x) = x^n - w$ 꼴로 표현된다. 기약 이항다항식의 존재성에 대한 다음의 정리는 [12]에 나타나 있다.

(정리 1) 정수 n이 2보다 크거나 같고 유한체 GF(p)의 원소 w에 대해 $x^n - w$가 기약 이항다항식인 것은 다음 두 가지 조건을 만족하는 것과 동치이다.

(1) n의 각 소인수는 GF(p)에서 w의 위수(oder) e를 나누지만 (p-1)/e를 나누지는 못한다.

(2) $n \equiv 0 \bmod 4$ 이면 $p \equiv 1 \bmod 4$ 이다.

$p=2^8-5$, 2^8-15, $2^{16}-165$일 때 $x^{25}-6$, $x^{20}-7$, $x^{10}-2$ 등이 각각 기약 이항다항식이 된다.[5]

유한체 $GF(p^n)$의 한 원소 $a = a_0 + a_1\alpha^1 + \cdots + a_{n-1}\alpha^{n-1}$으로 표현되고 이것은 집합 $\{1, \alpha^1, \cdots, \alpha^{n-1}\}$의 원소와 GF(p) 원소의 일차결합으로 볼 수 있다. 또한 유한체 $GF(p^n)$은 유한체 GF(p) 위의 n차원 벡터공간으로 볼 수 있으며, 집합 $A= \{1, \alpha^1, \cdots, \alpha^{n-1}\}$은 $GF(p^n)$의 기저(basis)가 되며, 이 기저를 다항식 기저라고 부른다. 원소 a는 기저를 생략하고 계수만 사용하여 $a = (a_0, a_1, \cdots, a_{n-1})$으로 표현할 수 있으며 이를 기저 A에 대한 벡터표현이라고 한다.

a와 b를 $GF(p^n)$의 두 원소라 하고 이를 다항식 기저 A로 표현하면

$$a = a_0 + a_1\alpha^1 + \cdots + a_{n-1}\alpha^{n-1} ,$$

$$b = b_0 + b_1\alpha^1 + \cdots + b_{n-1}\alpha^{n-1}$$

과 같이 나타낼 수 있다. a를 기저 A에 대한 벡터 표현이라고 하면, $a = (a_0, a_1, \cdots, a_{n-1})$ 으로 표현되고, $b = (b_0, b_1, \cdots, b_{n-1})$로 나타난다. 유한체의 두 원소 a와 b의 곱은

$$\begin{aligned} a \cdot b &= (a_0, a_1, \cdots, a_{n-1})(b_0, b_1, \cdots, b_{n-1}) \\ &= (a_0, a_1, \cdots, a_{n-1})b_0 + \\ &\quad (a_0, a_1, \cdots, a_{n-1})b_1\alpha^1 + \cdots + \\ &\quad (a_0, a_1, \cdots, a_{n-1})b_{n-1}\alpha^{n-1} \\ &= (a_0, a_1, \cdots, a_{n-1})b_0 + \\ &\quad [(a_0, a_1, \cdots, a_{n-1})\alpha^1]b_1 + \cdots + \\ &\quad [(a_0, a_1, \cdots, a_{n-1})\alpha^{n-1}]b_{n-1} \end{aligned} \quad (2)$$

이다. $\alpha^n = w$인 사실을 이용하면 i = 1, 2, ⋯, n-1에 대해 $a \cdot \alpha^i$을 다음 식 (3)과 같이 구할 수 있다.

$$\begin{aligned} a \cdot \alpha &= (a_0 + a_1\alpha^1 + \cdots + a_{n-1}\alpha^{n-1}) \cdot \alpha \\ &= a_0\alpha + a_1\alpha^2 + \cdots + a_{n-1}\alpha^n \\ &= a_{n-1}w + a_0\alpha + a_1\alpha^2 + \cdots + a_{n-1}\alpha^{n-1} \\ &= (\, a_{n-1}w,\ a_0,\ a_1,\ \cdots,\ a_{n-2}\,), \\ a \cdot \alpha^2 &= (a \cdot \alpha) \cdot \alpha \\ &= (\, a_{n-1}w,\ a_0,\ a_1,\ \cdots,\ a_{n-2}\,) \cdot \alpha \\ &= (\, a_{n-2}w,\ a_{n-1}w,\ a_0,\ \cdots,\ a_{n-3}\,), \\ &\cdots, \\ a \cdot \alpha^{n-1} &= (a \cdot \alpha^{n-2}) \cdot \alpha \\ &= (\, a_2w,\ a_3w,\ \cdots,\ a_0,\ a_1) \cdot \alpha \\ &= (\, a_1w,\ a_2w,\ \cdots,\ a_{n-1}w,\ a_0). \end{aligned} \quad (3)$$

식(3)의 i번째 $a \cdot \alpha^i$는 i-1번째 $a \cdot \alpha^{i-1}$의 벡터 표현을 오른쪽으로 순환 이동하여 벡터를 얻은 후 첫 번째 원소에 w를 곱해서 얻을 수 있다. a와 b의 곱은 i=0, 1, ⋯, n-1에 대해 식 (3)의 i번째 수식에 b_i를 곱한 후 모든 행을 더한 것이다. 따라서 식 (3)의 행벡터를 열벡터로 바꾸어 식 (4)와 같이 행렬 M을 구성하면, a와 b의 곱을 c라 할 때 c는 행렬 M과 벡터 b를 전치시켜 행렬로 표현한 $[b]^T$의 곱인 $c = M[b]^T$로 나타낼 수 있다.

$$M = \begin{pmatrix} a_0 & a_{n-1}w & a_{n-2}w & \cdots & a_2w & a_1w \\ a_1 & a_0 & a_{n-1}w & \cdots & a_3w & a_2w \\ a_2 & a_1 & a_0 & \cdots & a_4w & a_3w \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-2} & a_{n-3} & a_{n-4} & \cdots & a_0 & a_{n-1}w \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_1 & a_0 \end{pmatrix} \quad (4)$$

2.2 기약 이항다항식을 이용한 직렬곱셈기 구조

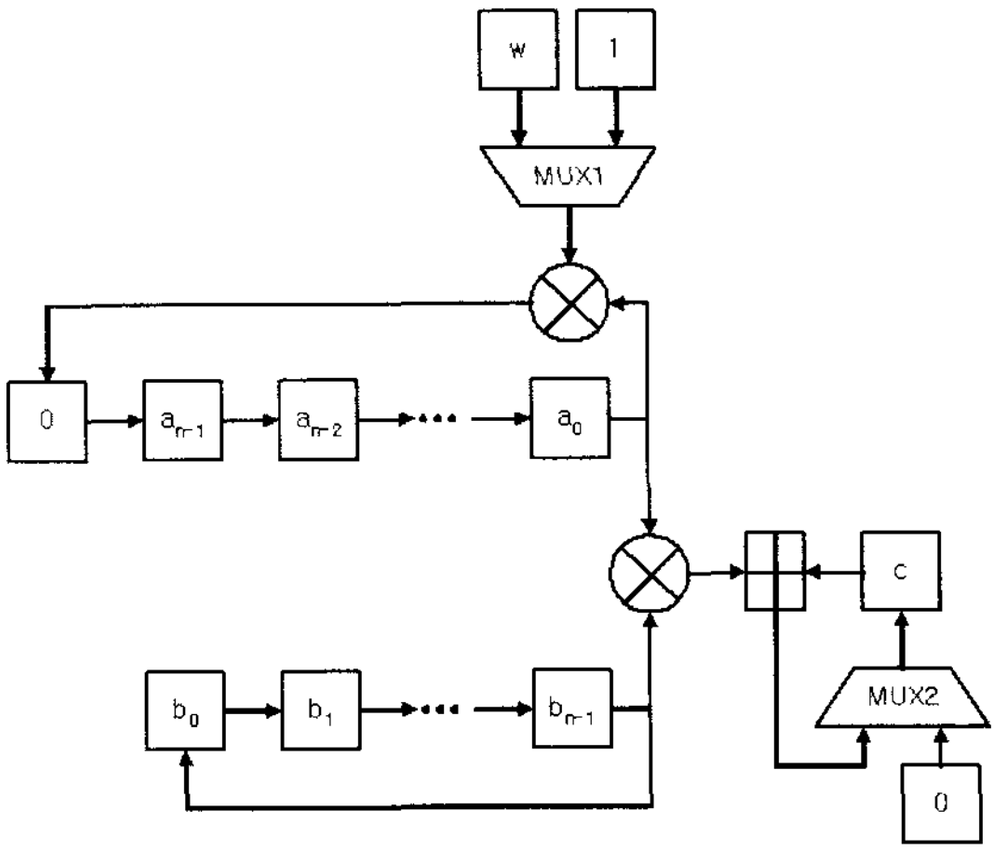
2.1절에서 $GF(p^n)$의 두 원소 a와 b의 곱을 원소 a에 기약 다항식의 근인 α를 곱하여 행렬 M을 구하고 행렬 M과 전치행렬 $[b]^T$의 곱으로 구할 수 있음을 보였다. 두 원소의 곱 c를 순차적으로 구하기 위해서 행렬 M의 마지막 행부터 $[b]^T$를 곱하는 것을 생각해보자. 그러면

$$\begin{aligned} c_{n-1} &= [a_{n-1}, a_{n-2}, \cdots, a_1, a_0][b]^T, \\ c_{n-2} &= [a_{n-2}, a_{n-3}, \cdots, a_0, a_{n-1}w][b]^T, \\ c_{n-3} &= [a_{n-3}, a_{n-4}, \cdots, a_{n-1}w, a_{n-2}w][b]^T, \\ &\cdots, \\ c_0 &= [a_{n-1}, a_{n-2}w, \cdots, a_1w, a_0w][b]^T \end{aligned} \quad (5)$$

이다. 즉, c_{i-1}은 c_i의 $[a_i, a_{i-1}, \cdots, a_{i+2}w, a_{i+1}w]$의 첫 번째 원소에 w를 곱한 후 왼쪽으로 한 칸씩 순환 이동시킨 결과인 $[a_{i-1}, a_{i-2}, \cdots, a_{i+1}w, a_iw]$와 $[b]^T$를 곱한 것이다. 따라서

$$c_{i-1} = a_{i-1}b_0 + a_{i-2}b_1 + \cdots + (a_{i+1}w)b_{n-2} + (a_iw)b_{n-1} \quad (6)$$

이고 이것은 GF(p) 원소의 곱셈기와 덧셈기 하나씩만 이용하여 n 클럭 싸이클 이후에 결과를 얻는 구조로 설계할 수 있다. 그러나 다음 원소인 c_{i-2}를 구하기 위해서는 다시 첫 번째 원소에 w를 곱한 후 왼쪽으로 한 칸씩 순환 이동시키는 연산이 한 번 더 필요하다. 이를 [그림 1]과 같은 구조로 설계하였다. 벡터 a를 담는 레지스터를 n+1개를 사용하고 마지막 레지스터에 0 값을 저장하여 n+1 클럭 싸이클 때 0 값이 b의 값과 곱해지고 b



〔그림 1〕 기약 이항다항식을 이용한 직렬곱셈기

가 순환 이동되도록 하였다. [그림 1]에서 ⊗는 GF(p)의 곱셈을 의미한다. MUX1은 n 클럭 싸이클 마다 w가 선택되어 벡터 a의 값과 곱해지도록 하고 다른 때는 1과 곱하여 그 값을 그대로 유지하게 하였다. n+1 클럭 싸이클 뒤에 벡터 b와 곱해지는 벡터 a의 레지스터 값이 0이므로 계산 결과에 영향을 미치지 않지만 벡터 b의 레지스터가 순환 이동하여 행렬 M의 다음 행의 값과 곱해지는 구조이다. 레지스터 c는 곱셈 결과를 저장하는 레지스터로 행렬 M의 원소와 벡터 b의 곱셈 결과를 축적하여 저장한다. [그림 1]에서 ⊞는 GF(p)의 덧셈을 의미한다. MUX2는 행렬 M의 한 행의 결과가 다 끝나는 n 클럭 뒤에 c의 원소를 0으로 초기화하기 위해 0 값을 선택하도록 구성하였다. n+1 클럭 싸이클 때 c에 저장된 값이 계산 결과이다. 이 직렬 곱셈기는 레지스터를 2n+5개, MUX를 두개, GF(p) 원소의 곱셈기를 두 개, GF(p) 원소의 덧셈기를 한 개 사용하여 (n+1)n 클럭 싸이클 이후에 결과를 얻는 구조이다.

Ⅲ. 기약 AOP를 이용한 직렬 곱셈기

3.1 기약 AOP와 잉여기저를 이용한 두 원소의 곱셈

유한체 GF(p) 위의 n차 기약 AOP(All One Polynomial)는 항의 개수가 n+1개이고 계수가 모두 1인 기약다항식을 말한다. 즉, 기약 AOP는 $f(x) = x^n + x^{n-1} + \cdots + x^2 + x + 1$ 꼴로 표현된다. 기약 이항다항식의 존재성에 대한 다음의 정리는 [11]에 나타나있다.

(정리 2) 유한체 $GF(p^n)$이 (n+1)승 단위 근으로 이루어진 최적 정규기저(optimal normal basis)를 갖기 위한 필요충분조건은 n+1이 소수이고 p가 $\mathbb{Z}_{n+1}^*$의 생성원인 것이다.

β를 최적 정규기저를 이루는 (n+1)승 단위 근(root of unity)이라면 β의 최소다항식이 $f(x) = x^n + x^{n-1} + \cdots + x^2 + x + 1$이 되므로 f(x)는 기약 AOP 다항식이 되고 $GF(p^n)$를 생성한다.[11] 따라서 n+1이 소수이고 $\mathbb{Z}_{n+1}$의 생성원 중 소수 p가 있으면 AOP는 기약다항식이 된다. 예를 들면 7은 $\mathbb{Z}_{11}^*$의 생성원이므로 $f(x) = x^{10} + x^9 + \cdots + 1$은 기약 AOP이고 $GF(7^{10})$을 만든다.

f(x)를 GF(p) 위의 기약 AOP 다항식이라 하자. 그리고 β를 f(x)의 한 근이라 하면 집합 $B=\{1, \beta, \beta^2, \cdots, \beta^{n-1}, \beta^n\}$은 GF(p) 위의 n차 확창체 $GF(p^n)$의 잉여기저(redundant basis)가 된다.[15]

$GF(p^n)$의 두 원소 a와 b는

$$a = a_0 + a_1\beta^1 + \cdots + a_{n-1}\beta^{n-1} + a_n\beta^n ,$$
$$b = b_0 + b_1\beta^1 + \cdots + b_{n-1}\beta^{n-1} + b_n\beta^n \quad (7)$$

으로 잉여 한 디지트를 더 사용하여 표현할 수 있다. 유한체의 생성 기약 다항식 $f(x) = 1+x+x^2+...+x^n$이므로 $\beta^{n+1} = 1$임은 잘 알려진 사실이다. 이 사실을 이용하면 i=1, 2, ⋯, n-1에 대해 $a \cdot \beta^i$를 다음 식 (8)과 같이 구할 수 있다.

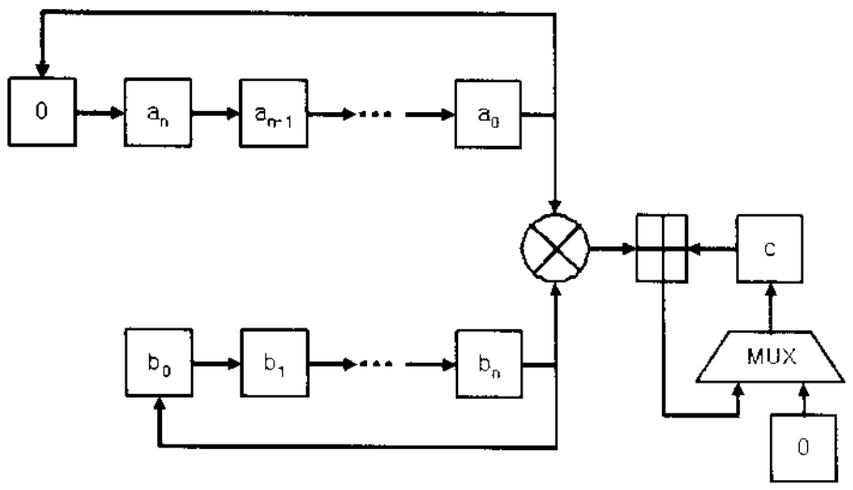
$$\begin{aligned} a \cdot \beta &= a_0\beta + a_1\beta^2 + \cdots + a_n\beta^{n+1} \\ &= a_n \cdot 1 + a_0\beta + a_1\beta^2 + \cdots + a_{n-1}\beta^n \\ &= (a_n, a_0, a_1, \cdots, a_{n-1}), \\ a \cdot \beta^2 &= (a \cdot \beta) \cdot \beta \\ &= (a_n \cdot 1 + a_0\beta + a_1\beta^2 + \cdots + a_{n-1}\beta^n) \cdot \beta \\ &= a_n\beta + a_0\beta^2 + a_1\beta^3 + \cdots + a_{n-1}\beta^{n+1} \\ &= a_{n-1} + a_n\beta + a_0\beta^2 + \cdots + a_{n-2}\beta^n \\ &= (a_{n-1}, a_n, a_0, \cdots, a_{n-2}), \\ &\cdots, \\ a \cdot \beta^n &= (a_n, a_{n-1}, \cdots, a_1, a_0). \end{aligned} \quad (8)$$

식 (8)의 i번째 행 $a \cdot \beta^i$는 i-1번째 $a \cdot \beta^{i-1}$의 벡터 표현을 오른쪽으로 순환 이동하여 얻을 수 있다. a와 b의 곱은 i=0, 1, ⋯, n-1에 대해 식 (8)의 i번째 행에 b_i를 곱한 후 모든 행을 더한 것이다. 따라서 식 (8)의 각 행 벡터를 열벡터로 바꾸어 식 (9)와 같이 행렬 N을 구성하면, a와 b의 곱을 c라 할 때 c는 행렬 N과 $[b]^T$의 곱인 $c = N[b]^T$로 나타낼 수 있다.

$$N = \begin{pmatrix} a_0 & a_n & a_{n-1} & \cdots & a_2 & a_1 \\ a_1 & a_0 & a_n & \cdots & a_3 & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_4 & a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \cdots & a_0 & a_n \\ a_n & a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \end{pmatrix} \quad (9)$$

3.2 기약 AOP를 이용한 직렬곱셈기 구조

3.1절에서 $GF(p^n)$의 두 원소 a와 b의 곱 c를 행렬 N과 전치행렬 $[b]^T$의 곱으로 볼 수 있음을 보였다. 두 원소의 곱 c를 순차적으로 구하기 위해서 행렬 N의 마지막 행부터 $[b]^T$를 곱하는 것을 생각해보자. 그러면



〔그림 2〕 기약 AOP를 이용한 직렬곱셈기

$$c_{n-1} = [a_n, a_{n-1}, \cdots, a_1, a_0][b]^T,$$
$$c_{n-2} = [a_{n-1}, a_{n-2}, \cdots, a_0, a_n][b]^T,$$
$$c_{n-3} = [a_{n-2}, a_{n-3}, \cdots, a_n, a_{n-1}][b]^T,$$
$$\cdots,$$
$$c_0 = [a_0, a_n, \cdots, a_2, a_1][b]^T \quad (10)$$

이다. 즉, c_{i-1}은 c_i의 $[a_{i+1}, a_i, \cdots, a_{i+3}, a_{i+2}]$를 왼쪽으로 한 칸씩 순환 이동시킨 결과인 $[a_i, a_{i-1}, \cdots, a_{i+2}, a_{i+1}]$과 $[b]^T$를 곱한 것이다. 따라서

$$c_{i-1} = a_ib_0 + a_{i-1}b_1 + \cdots + a_{i+2}b_{n-2} + a_{i+2}b_{n-1}$$

는 GF(p) 원소의 곱셈과 덧셈을 하나씩만 이용하여 n+1 클럭 싸이클 이후에 결과를 얻을 수 있다. 이를 하드웨어로 구현하기 위해 벡터 a를 담는 레지스터를 n+2개를 사용하였고 맨 마지막 레지스터에 0 값을 저장하여 순환연산이 일어나도록 [그림 2]와 같이 설계하였다. 즉, n+2 클럭 싸이클 뒤에 벡터 b와 곱해지는 벡터 a의 레지스터 값을 0이 되도록 하여 벡터 b의 레지스터가 순환 이동하도록 하여 행렬 N의 다음 행의 값과 곱해지도록 하였다. [그림 2]에서 ⊗는 GF(p)의 곱셈을 의미한다. 레지스터 c는 곱셈 결과를 저장하는 레지스터로 행렬 N의 원소와 벡터 b의 곱셈 결과를 축적하여 저장한다. [그림 2]에서 ⊞는 GF(p)의 덧셈을 의미한다. MUX는 행렬 N의 한 행의 결과가 다 끝나는 n+1 클럭 뒤에 c의 원소를 0으로 초기화하기 위해 0 값을 선택하도록 구성하였다. 이 직렬 곱셈기는 레지스터를 2n+5개, MUX를 한 개, GF(p) 원소의 곱셈기를 한 개, GF(p) 원소의 덧셈기를 한 개 사용하여 (n+2) (n+1) 클럭 싸이클 이후에 결과를 얻는 구조이다.

Ⅳ. 결 론

최근 유비쿼터스 환경이 실현 가능해지면서 디바이스 인증, 개인정보보호 등의 정보보호 기술의 적용이 필요해지고 있다. 유비쿼터스 환경의 디바이스들은 컴퓨팅 능력, 대역폭, 면적 등의 제약을 가지고 있으므로 이러한 제약사항을 고려한 효율적인 정보보호 기술의 하드웨어 구현은 안전한 유비쿼터스 사회를 앞당기는데 큰 몫을 할 것임은 분명하다. 특히 소프트웨어와 하드웨어적으로 단순한 구조를 갖는 유한체의 연산기 구조에 대한 연구는 제약적인 환경의 디바이스 위에 다양한 정보보호 기술을 올릴 수 있는 가능성을 제공한다. 이진 확장체에 대한 연구는 주로 속도의 효율성을 고려하여 병렬구조에 대한 연구가 많았다. 이진 확장체의 직렬구조는 비트단위의 계산으로 면적의 효율성은 극대화 할 수 있으나 지연시간이 너무 길어지는 단점이 있다. 이런 측면에서 적절한 홀수 소수에 대한 소수 확장체는 디지트 단위의 연산을 수행하므로 면적의 효율성과 시간의 효율성을 교환하여 환경에 적합안 구조를 택할 수 있다.

본 논문에서는 정보보호 기술의 기반 프리미티브를 제공하는 홀수 소수 확장체의 두 원소의 곱셈을 제약적인 환경의 디바이스에 하드웨어로 구현 가능하도록 두 가지 구조를 제안하였다. 하나는 기약다항식 중에서 항의 개수가 가장 작은 기약 이항다항식과 다항식 기저를 이용한 곱셈기 구조이고, 다른 하나는 모든 계수가 1인 기약 AOP와 잉여기저를 이용한 곱셈기 구조이다. [표 1]은 이 두 개의 직렬 곱셈기에 소요되는 기본소자와 지연시간을 나타낸 것이다. 기약 AOP를 이용한 직렬곱셈기가 기약 이항다항식을 이용한 곱셈기보다 면적의 효율성이 좋음을 알 수 있다. 그러나 원소의 표현을 한 디지트 더 사용한 잉여기저를 사용함으로써 곱셈 결과를 얻기까지 필요한 클럭 수가 더 필요함을 알 수 있다.

정리 1과 정리 2에 의하면 모든 양의 정수 n에 대해 기약 이항다항식과 기약 AOP 다항식이 존재하는 것은 아니다. 즉, n에 대해 기약 이항다항식과 기약 AOP가 없을 수도 있고, 기약 이항다항식 또는 기약 AOP 둘 중 하나만 있을 수도 있으며 두 다항식 모두 있을 수도 있다. 제약적인 환경의 디바이스에 적용 가능한 유한체 차

〔표 1〕 $GF(p^n)$의 직렬곱셈기의 복잡도

종류	레지스터 수	MUX 수	GF(p) 곱셈기 수	GF(p) 덧셈기 수	필요 클럭 수
기약 이항 다항식	2n+5	2	2	1	n2+n
기약 AOP	2n+5	1	1	1	n2+3n+2

수의 범위가 주어지면 이 범위 내에서 기약 이항다항식 또는 기약 AOP가 존재하는 n을 찾은 후 본 논문에서 제안하는 곱셈기를 구현할 수 있다. 이들 곱셈기를 이용하면 GF(p^n) 위에서 응용되는 타원곡선 암호알고리즘, ElGamal 암호 알고리즘 등의 공개키 프리미티브를 제약적인 환경의 디바이스에 구현해 볼 수 있을 것이다. 향후 제안한 연산기의 시뮬레이션을 통한 검증과 이를 이용한 공개키 프리미티브의 효율적인 구현 구조에 대해 연구를 할 필요가 있다.

참고문헌

[1] 이옥석, 장남수, 김창한, 홍석희, "Equally Spaced 기약다항식 기반의 효율적인 이진체 비트-병렬 곱셈기", 정보보호학회 논문지 제18권 제2호, pp.3-9, 2008.

[2] 이진호, 김현성, "GF(2m) 상에서 디지트 단위 모듈러 곱셈/제곱을 위한 시스톨릭 구조", 정보보호학회 논문지 제18권 제1호, pp.41-47, 2008.

[3] 조용석, "유한체 상에서 고속연산을 위한 직렬 곱셈기의 병렬화 구조", 정보보호학회 논문지 제17권 제1호, pp.33-39, 2007.

[4] 조용석, "GF(2m) 상의 저복잡도 고속-직렬 곱셈기 구조", 정보보호학회 논문지 제17권 제4호, pp.97-102, 2008.

[5] D. V. Bailey and C. Paar, "Optimal Extention Fields for Fast Arithmetic in Public Key Algorithm", Advanced in Cryptology - CRYPTO'98, LNCS 1462, pp.472-458, 1998

[6] G. Bertoni, L. Breveglieri, P. Fragneto, and G. Pelosi, "Parallel Hardware Architectures for the Cryptographic Tate Pairing," Proceedings of the Third International Conferenceon Information Technology : New Generations (ITNG'06), pp.186-191, 2006.

[7] J. Beuchat, M. Shirase, T. Takagi, E. Oka- moto, "An Algorithm for the Eta_T Pairing Calculation in Characteristic Three and its Hardware Implementation", 18th IEEE International Symposium on Computer Arithmetic, ARITH-18, pp.97-104, 2007.

[8] P. Grabher and D. Page, "Hardware Acceleration of the Tate Pairing in Characteristic Three," CHES 2005, LNCS 3659, pp.398-411, Springer-Verlag, 2005.

[9] J. L. Massey and J, K. Omura, "Computational method and apparatus for finite field arithmetic", U. S. Patent No. 4587627, 1984

[10] A. J. Menezes, "Application of Finite Fields", Kluwer Academic Publishers, 1993

[11] R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone, "Optimal Normal Bases in GF(p^n)", Discrete Applied Mathematics 22, pp.149-161, 1987.

[12] R. Lidl and H. Neiderrieter, "Finite Fields", Encyclopedia of Mathematics and its Applications, vol. 20, Addison-Wesley, 1983.

[13] C. Paar, "A new architecture for a parallel finite field multiplier with low complexity based on composite fields", IEEE Trans. on computers, vol.45, No.7, pp.856-861, 1996.

[14] M. Weiser, "The Computer for the 21 Centry, Scientific American", Vol. 256, No. 3, pp.94-104, 1991.

[15] H. Wu, "Efficient Computations in Finite Fields with Cryptographic Significance", Ph.D Thesis in Computer Engineering, Univ. of Waterloo, Canada, 1999.

〈著者紹介〉

정 석 원 (Seok Won Jung) 종신회원

1991년 2월 : 고려대학교 수학과 졸업
1993년 2월 : 고려대학교 수학과 이학석사
1997년 2월 : 고려대학교 수학과 이학박사
2002년 3월~2004년 2월 : 고려대학교 정보보호기술연구센터 연구교수
2004년 3월~현재 : 목포대학교 정보보호전공 조교수
<관심분야> 암호 알고리즘 구현, 스마트카드 보안, 방송보안