저면적 암호프로세서를 위한 고속직렬유한체 승산기설계

*김영훈,*이광엽,**김원종,**배영환,**조한진
*서경대학교, **한국전자통신연구원
전화 02-940-7240/핸드폰 019-225-2870

Design of a fast-serial finite field multiplier for Low cost Crypto-processors

*Young Hoon Kim,*Kwang Youb Lee,**Won Jong Kim,**Young Hwan Bae,**Han Jin Cho
*Dept. of Computer Eng. Seokyeong University, **ETRI
E-mail : kylee@skuniv.ac.kr

ABSTRACT

In this paper, an efficient architecture for the finite field multiplier is proposed. This architecture is faster and smaller than any other LFSR architectures. The traditional LFSR architecture needs t x m registers for achieving the t times speed. But, we desinged the multiplier using a novel fast architecture without increasing the number of registers. The proposed multiplier is verified with a VHDL description using SYNOPSYS simulator. The measured results show that the proposed multiplier is 2 times faster than the serial LFSR multiplier. The proposed multiplier is expected to become even more advantageous in the smart card cryptography processors.

I. 서론

Bit-serial 승산기는 주로 LFSR(Linear Feedback Shift Register) 구조를 사용하여 설계되는데 암호bit수에 비례하여 지연시간이 증가하지만 bit-parallel 보다 게이트의 수를 감소시키는 장점이 있다.

LFSR구조의 직렬형 유한체 승산기는 승산 데이터의 bit수가 각각 mbit일 때 m클럭 사이클의 계산시간과 3·m의 register가 소요된다. 계산 지연시간을 줄이기 위해서는 LFSR에서 한 클럭 사이클에 1bit씩 쉬프트하지 않고 여러bit를 쉬프트하는 기술이 필요하다. 즉, 한 클럭 사이클마다 여러 bit가 쉬프트하면 쉬프되는 bit 수만큼 clock cycle이 줄어든다. 예를 들어 한 clock cycle에 2bit씩 쉬프트하면 계산시간은 m/2만큼 줄어든다. 이때 한 clock cycle에 여러 bit를 shift하기 위해서는 register 및 gate count가 증가하게되는 단점이 있는데 기존의 구조에서와 같이 한 cycle에 2 bit씩 shift하면 4·m의 register가 소요되며 shift되는 bit수를 t로 하면 t·m의 register가 필요하다.

본 논문에서는 기본적인 LFSR구조의 유한체 승산기에서 회로의 크기를 크게 증가시키지 않고 속도를 개선하는 기술을 제안한다.

본 논문은 최근 스마트카드 등의 휴대형 정보단말 장치에서 널리 사용되고 있는 공개키 암호방식 가운데 타원곡선 암호알고리즘을 회로로 구현하는데 큰 장점을 갖는다.

II. 기존의 직렬유한체 승산기

직렬 유한체 승산기는 Mastrovito에 의하여 제안되어 유한체 승산기의 가장 기본적인 구조로 자리를 잡고 있다. 직렬 유한체 승산의 표현식은

$$Z = A \cdot B = \sum_{i=0}^{m-1} b_i(A\alpha_i)$$
$$= [\cdots[[b_0(A) + b_1(A\alpha)] + b_2(A\alpha^2)] + \cdots] + b_{m-1}(A\alpha^{m-1}) \quad (1)$$

이다. α에 관한 식을 정리하면
$\alpha^m = p_0 + p_1\alpha + \cdots + p_{m-1}\alpha^{m-1}$ 이므로

$$A\alpha = a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^m$$
$$= a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}(p_0\alpha + p_1\alpha_2 + \cdots + p_{m-1}\alpha^{m-1})$$
$$= a_{m-1}p_0 + (a_0 + a_{m-1}p_1)\alpha + (a_1 + a_{m-1}p_2)\alpha^2 + \cdots + (a_{m-2} + a_{m-1}p_{m-1})\alpha^{m-1} \quad (2)$$

(2)식과 함께 (1) 식을 회로로 구현하면 그림 1과 같은 bit-serial 승산기를 얻을 수 있다. 그림 1은 일반적으로 LFSR(Linear Feedback Shift Register) 구조라고 부른다.[1][2] 그림 1과 같은 구조의 기본적인 LFSR구조의 직렬 유한체 승산기는 m비트의 데이터를 승산하는데 3·m개의 레지스터가 필요하고 m개의 클럭사이클수의 계산속도를 갖는다. 직렬 유한체 승산기는 단순한 구조로 하드웨어 자원을 많이 소모하지 않지만 m이 커질수록 계산 결과가 늦어지는 단점이 있다.

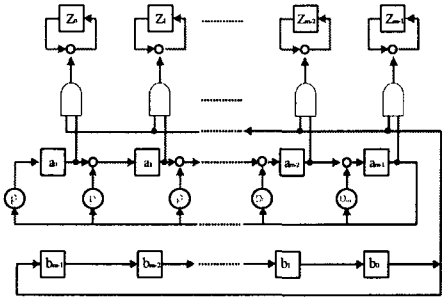

그림 1 LFSR구조를 이용한 직렬유한체승산기

기본적인 LFSR구조의 직렬 유한체 승산기의 처리속도를 개선하기 위하여 여러가지 연구가[3] 진행되었고 최근에는 그림 2와 같은 구조가 발표되었다.[4] 이 구조는 식(2)를 2부분으로 나누어 2배의 속도향상을 얻는방법으로 식(2)를 식(3)과 같이 계수가 짝수인 부분과 홀수인 부분으로 나눈다. 식(4)의 관계식을 이용하여 modulo reduction을 수행하면 low hamming weight에서는 p_{m-1}은 0 으로 대치할 수 있다. 결과적으로 x^2에 의하여 modulo reduction을 수행하면 식(5)를 얻는다. 식(5)의 표현을 이용하면 x^2의 reduction으로 Zeven(x)와 Zodd(x)를 동시에 수행하여 throughput을 2배 증가시킬 수 있다. 이 방법을 이용하여 회로를 구현하면 그림 3과 같이 Zeven(x)회로에는 m개의 AND게이트, m개의 XOR게이트, m개의 짝수 차수에 대한 결과 레지스터가 있고, 이와 비슷하게 Zodd(x)회로에는 m개의 AND게이트, m개의 XOR게이트, m개의 홀수 차수에 대한 결과 레지스터가 존재한다. 따라서, 그림 2의 방법으로 기본적인 LFSR 승산기보다 속도를 2배 개선하면 m개의 레지스터가 더 소요되는 단점이 있다.

$$Z_{even}(x) = [b_0A(x) + \ldots + b_{m-3}x^{m-3}A(x) + b_{m-1}A(x)] \bmod P(x)$$

$$Z_{odd}(x) = [b_1xA(x) + b_3x^3A(x) + \ldots + b_{m-2}x^{m-2}A(x)] \bmod P(x) = x[b_1A(x) + b_3x^2A(x) + \ldots + b_{m-2}x^{m-3}A(x)] \bmod P(x) \quad (3)$$

$$x^m \equiv p_0 + p_1x + \ldots + p_{m-1}x^{m-1} \bmod P(x)$$
$$x^{m-1} = p_0x + p_1x^2 + \ldots + p_{m-2}x^{m-1} + p_{m-1}x^m \bmod P(x) \quad (4)$$

$$x^2A(x) = a_0x^2 + a_1x^3 + a_2x^4 + \ldots + a_{m-2}x^m + a_{m-1}x^{m+1} = a_{m-2}p_0 + (a_{m-2}p_1 + a_{m-1}p_0)x + (a_{m-2}p_2 + a_{m-1}p_0 + a_0)x^2 + \ldots + (a_{m-2}p_{m-1} + a_{m-1}p_{m-2} + a_{m-3})x^{m-1} \quad (5)$$

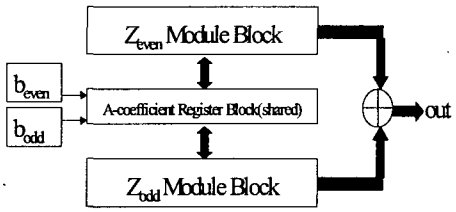

그림 4 LFSR구조를 개선한 2배속 승산기구조

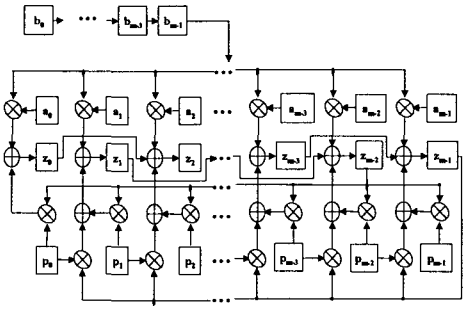

그림 5 개선된 2배속 승산기의 Zeven 모듈의 회로도

Ⅲ. 제안된 고속 직렬유한체 승산기

3-1. register 수의 증가가 없는 회로구조

기존의 개선된 2배속 승산기 구조에서는 유한체 승산기의 계산속도를 증가시키기 위해서 register의 수를 승산 데이터의 bit 수만큼 증가시켜야 한다.
또한, exclusive 게이트의 수가 증가하고 연결선이 복잡해져 회로의 크기를 크게 증가시키게 된다.

본 논문에서 제안하는 유한체 승산기의 구조는 그림 4에 수록한 것과 같이 register나 exclusive 게이트의 수를 크게 증가시키지 않고 그림 1에 나타낸 종래의 유한체

승산기 보다 계산속도를 2배 증가시킬 수 있다.

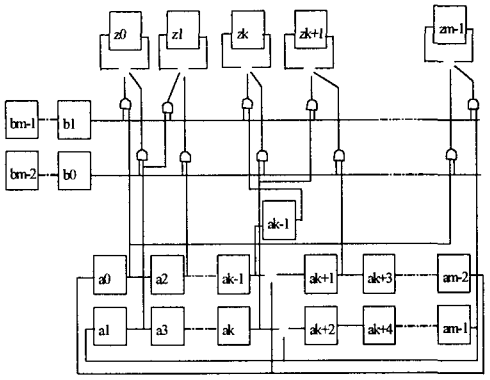

그림 6 제안된 유한체승산기 구조

그림 2의 방법은 그림 1의 기본적인 형태의 유한체 승산기와 비교하여 승산속도를 2배 증가하기 위해 승산 결과를 저장하는 Z register가 2배 크기로 증가하고 exclusive OR 게이트가 2배 증가하며 이를 연결하는 연결선의 복잡 도가 크게 증가한다.

그러나 그림 3에 있는 본 논문의 유한체 승산기는 그림 1의 기본적인 형태의 유한체 승산기와 비교하여 register 수에 변화가 없고 단지 궤환(feedback)이 발생하는 지점에서 register가 1개씩만 추가된다. 또한, exclusive 게이트 수에 변화가 없고 단지 AND 게이트 수가 2배로 증가한다.

본 논문에서 제안하는 그림 4의 회로 구성내용은 다음과 같다. 각각 m bit의 크기를 갖는 입력 데이터 a, b를 승산하고 그 결과를 m bit의 z 라고 표시한다. 입력 데이터와 승산결과는 1개의 플립플롭(이하, register라고 칭함)에 1 bit씩 저장된다. 따라서, 입력 데이터 a는 a0 부터 am-1 까지 m개의 register에 저장되고, 입력 데이터 b는 b0 부터 bm-1 까지 m개의 register에 저장된다. 승산 결과 데이터 z 는 z0 부터 zm-1까지 m개의 register에 저장된다.

유한체 승산에서는 기약다항식(irreducible polynomial)에 의하여 승산 결과가 나누어지기 때문에 승산 결과 데이터의 길이가 2m이 되지 않고 m이 된다. 따라서 유한체 승산기에서는 승산과정과 기약다항식에 의한 나누기 과정이 함께 실행된다. 기약다항식에 의한 나누기 과정은 기약다항식의 각 항의 계수가 1 인 부분에서 궤환(feed back)이 이루어지도록 하면 된다. 그림 3에서는 기약다항식이 p(x) = xm-1 + xk+1 일때 회로 구조를 나타낸다. 그림 3에서 보듯이 기약다항식의 항의 계수가 1인 부분은 k 번째 항과 0 번째 항이다. 따라서, k 번째 항과 0번째 항에서 궤환이 발생하였다. 만일 기약다항식의 항의 개수가 늘어나면 궤환의 개수도 늘어나면 된다.

기약다항식에서 k번째와 0번째 항에서 계수가 1을 갖기 때문에 k번째와 0번째 a register에서 궤환이 발생하는데 이때 궤환선을 연결하는 방법은 m-1번째 a register의 출력을 k번째 a register의 출력과 exclusive OR의 연산을 한 다음 k+2번째 aregister 입력에 연결한다. 또한, m-1 번째 a register의 출력을 1번째 a register의 입력에 연결한다. 특히, 본 발명에서 제안하는 것은 m-2번째 a register의 출력도 궤환을 시켜야 한다. 즉, m-2번째 a register의 출력을 k-1번째 a register의 출력과 exclusive OR연산을 한 후 k+1번째 a register의 입력으로 연결한다. 또한, m-2번째 출력은 0번째 a register의 입력으로 궤환한다.

본 발명에서 제안한 구조의 또 다른 특징은 궤환이 발생하는 a register 보다 한 차수가 낮은 a register의 bit를 별도의 register를 두어 저장하는 것이다. 이렇게 별도로 저장된 a register의 값은 z register에서 승산결과를 연산하는데 사용된다.

3-2. Modular Cell 회로구조

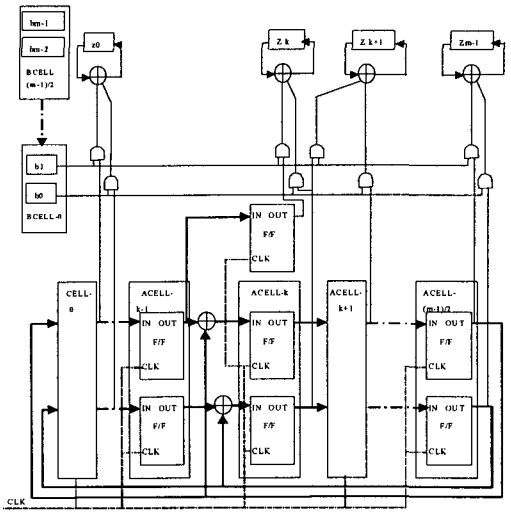

그림 7 3종류의 CELL로 구성된 회로

본 논문의 승산기는 기존에 발표된 LFSR구조의 승산기에 비하여 모듈화가 용이한 회로구조를 갖는 것이 두번째 장점이다. 그림 5와 같이 전체 승산기회로는 3종류의 셀로 모듈화가 가능하다. aregister를 2bit씩 묶어서 ACELL이라 하고 bregister를 2bit씩 묶어서 BCELL이라 한다. a와 bregister가 각각 m개(입력 데이터의 크기가 각각 mbit를 의미함)이면 ACELL과 BCELL의 수는 m/2 개가된다. m이 홀수이면 m/2 +1개가된다. 결과값을 저장하는 ZCELL은 m개이다. 각 ACELL과 BCELL을 연결할 때 aregister의 홀수번째 bit는 홀수bit에 연결하고 짝수 bit는 짝수bit에 연결되도록 하며 bregister의 경우도 같은 방법으로 연결된다. 그 결과 a0는 a2에, a2는 a4에 연

결되는 모양을 갖으며 a1은 a3에, a3은 a5에 연결되는 모양이 된다. b의 경우도 같은 방법이 적용된다. 그림 5에서 보듯이 ACELL 과 BCELL안에 있는 두개의 register는 같은 클럭에 연결되어 있어 두개의 register가 동시에 데이터를 입력 받게된다. 그 결과 aregister와 bregister는 2bit씩 이동(shift)하는 구조를 갖는다. 따라서 m/2 클럭에 a와 bregister가 이동을 완료하게 된다. 이것은 그림 1의 종래의 구조보다 승산속도가 2배 증가한 것을 입증한다. 회로의 구성방법은 ACELL과 BCELL을 그림 5와 같이 연결한 다음 기약다항식에서 항의 계수가 1이되는 위치에 궤환선(feedback line)을 연결하면 된다. b register의 구조는 1bit씩 쉬프트하는 기존의 승산기와는 달리 2bit씩 쉬프트(shift)한다. 그림 4와 그림 5에서 보듯이 bregister의 b0와 b1 등 2bit가 한 BCELL-0가 되고, b2와 b3가 BCELL-1이 된다. 그리고, 클럭에 의하여 한번에 b2는 b0으로, b3는 b1으로 전달된다. zregister의 구조와 승산과정은 기존의 승산기에서는 aregister의 한 bit와 b register의 한 bit가 AND연산하고 그 값은 z register의 내용과 exclusive OR를 하여 다시 zregister에 저장된다. 그러나, 본 논문에서는 두개의 bregister bit와 두개의 aregister의 bit가 동시에 AND 연산하고 그 값은 zregister의 내용과 exclusive OR를 하여 zregister에 저장한다. 두개의 bit끼리 AND연산을 하는 방법은 그림 4, 5에서 보듯이 b0register에는 a1과 a2register의 출력이 연결되고 b1register에는 a0와 a1register가 연결된다. 이와 같은 방법을 일반기호로 표시하면 k-1번째 bregister는 k번째 aregister 및 k+1번째 aregister와 AND연산을 하고, k번째 bregister는 k-1번째 aregister 및 k번째 aregister와 AND연산을 한다.

그림 6의 ZCELL은 aregister의 값과 bregister의 값을 AND연산하고 그 결과값과 zregister의 값을 exclusive OR연산을 하여 zregister에 저장하는 연산을 수행한다. 이때, aregister의 bit순서에 따라, AND연산하는 bregister의 순서가 다르기 때문에 ZCELL이 2가지 형태를 갖는다.

그림 7은 ACELL의 내부 회로구조를 나타낸다. 본 논문에서는 승산기의 속도를 개선하기 위하여 유한체 승산 입력 데이터의 하나인 a register를 2bit씩 묶고 이 를 ACELL이라 칭한다. BCELL의 회로도는 그림 6의 ACELL 회로도와 동일하게 사용할 수 있다.

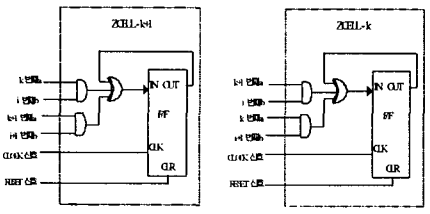

그림 8 두 가지 Z-CELL의 구조

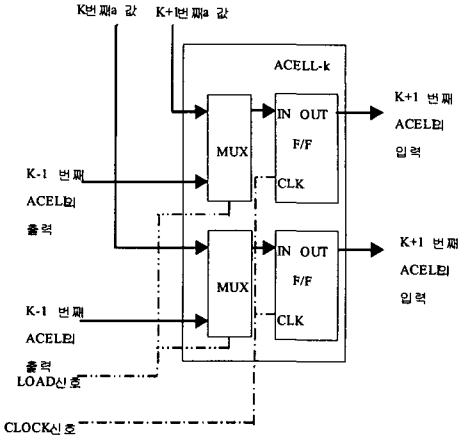

그림 7 A-CELL의 내부구조

V. 결론

본 논문은 다항식기저의 유한체 승산기를 설계할때 기존의 LFSR구조보다 2배 빠른 속도의 승산을 수행할 수 있는 구조를 제안하였다. 또한, 기존의 LFSR구조를 이용하여 속도를 개선한 승산기와는 달리 레지스터의 수를 크게 증가시키지 않으면서 처리속도를 개선하였다. 또한, 승산기 회로의 설계를 용이하게 하기 위하여 3가지 유형의 셀 회로를 제안하였다.

* *본 논문은 한국전자통신연구원의 지원을 받았으며 IDEC의 지원장비를 이용하였습니다*

참 고 문 헌

[1] 이만영, "BCH부호와 Reed-Solomon 부호", *민음사*, pp. 21-54

[2] Edoardo D. Mastrovito, "VLSI Architecture for computations in Galois Fields",Linkoping Studies in Science and Technology Dissertations, No.242,1991

[3] C. Paar and N.Lange,"A comparative VLSI synthesis of Finite Fields Multiplier", Proc. Third Intl Symp. Comm. Theory and Its Applications, Lake District, U.K. July, 1995.

[4] Sangook Moon, J. Park, Y. Lee,"Fast VLSI architecture algorithms for high-security elliptic curve cryptographic applications", IEEE Tr. on Consumer Electronics, Vol: 47, No. 3, August 2001

[5] David Naccache David M'Raïhi,"Cryptographic Smart Cards", *IEEE MICRO*, Vol 16, No 3, pp. 14-23, June, 1996