레지스터수의 증가가 없는 고속 직렬 유한체 승산기

정회원 이 광 엽*, 김 원 종**, 장 준 영**, 배 영 환**, 조 한 진**

Fast-Serial Finite Field Multiplier without increasing the number of registers

Kwang-Youb Lee*,Won-Jong Kim**, June-Young Chang**, Young-Hwan Bae**,Han-Jin Cho**

Regular Members

요 약

본 논문에서는 LFSR구조를 개선하여 기존의 LFSR구조 보다 면적과 속도면에서 효율적인 새로운 구조의 유한체 승산기를 제안한다. 기존의 LFSR구조에서는 t배 속도를 개선하기 위하여 레지스터의 수를 t x m 만큼의 레지스터 수가 증가하였다. 그러나, 본 논문에서는 레지스터의 수를 증가하지 않고 속도를 개선하는 구조를 이용하여 직렬 유한체 승산기를 설계하였다. 설계된 회로는 SYNOPSYS 시뮬레이션을 이용하여 LFSR구조에 비하여 2배 속도가 개선된 성능을 검증하였으며 또한, 본 논문의 고속, 저면적 승산기는 스마트카드와 같은 휴대형 단말기의 암호처리장치에 효과적으로 사용될 수 있음이 검증되었다.

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. 서 론

디지털 서명, 인증 및 전자상거래 등의 정보 보안 분야에서 정보 보호기술이 필수적이다. 정보 보호기술은 암호학적인 알고리즘에 의해 복잡한 연산을 거쳐 이루어지는데, 암호 알고리즘은 정수 소인수 분해의 어려움에 기반을 두거나 유한체 위에서 정의된 이산대수 문제에 기초를 두는 경우가 널리 사용되고 있다.

유한체(finite field)는 보통 Galois field(GF)라고도 불리우는데, GF 상에서 연산을 수행하는 연산 구조들은 기저표현방식(basis representation)에 따라서 분류된다. 가장 많이 사용되는 표현 방식들은 표준(standard)기저, 정규(normal) 기저, 이중(dual)기저인데, 이 가운데 표준기저와 정규기저가 많이 사용된다. 정규기저 표현방식은 유한체내의 거듭제곱 연산(exponentiation) 의 수행에서 매우 효율적인 면을 보이지만, 하드웨어로 구현하였을 때의 규칙성(regularity)과 확장성(extensibility)이 표준 기저 표현 방식보다 떨어진다. 표준 기저 표현방식은 대부분의 유한체 연산에서 효율적인 구조를 보이고 또한 사용하기에 가장 익숙한 구조이다.

표준기저의 유한체 연산기 가운데 승산기는 가장 사용빈도수가 높고 중요한 역할을 하기 때문에 많

* 서경대학교 컴퓨터공학과 (kylee@skuniv.ac.kr), ** 한국전자통신연구원

논문번호 : 020195-0423, 접수일자 : 2002년 4월 23일

※ 본 논문은 한국전자통신연구원의 지원으로 작성되었으며 회로구현에는 system2010과 IDEC의 지원장비를 사용하였습니다.

은 연구가 진행되어오고 있다. 유한체 승산기에 대한 기존 연구 결과들은 직렬 승산기(serial multiplier), 시스톨릭 병렬 승산기(systolic parallel multiplier), 하이브리드 승산기(hybrid multiplier) 등이 존재한다. 이 중에서 하이브리드 승산기는 가격 대 성능비가 가장 우수하지만, 유한체의 크기가 같더라도 암호학적인 복잡도가 감소되는 합성 유한체(GF($2n^k$); n^k는 합성수) 에서만 사용이 가능하고, 암호학적인 특성이 우수한 소수 유한체(GF(2p); p는 소수) 에서는 사용할 수 없는 단점이 있다. 시스톨릭 병렬 승산기는 2차원 배열(array) 유한체 승산기로 직렬 승산기에 비하여 m배의 하드웨어를 사용하여 m배의 계산 속도를 얻어내는 방식으로 가격 대 성능비에서 효율이 떨어진다.

반면, Bit-serial 구조의 승산기는 LFSR(Linear Feedback Shift Register)[1] 를 주로 사용하여 설계되는데 암호비트수에 비례하여 지연시간이 증가하지만 bit-parallel 보다 게이트의 수를 감소시키는 방법이다.

LFSR구조의 직렬형 유한체 승산기는 승산 데이터의 비트수가 각각 m비트일 때 m클럭 사이클의 계산시간과 $3 \cdot m$의 레지스터가 소요된다. 계산 지연시간을 줄이기 위해서는 LFSR에서 한 클럭 사이클에 1비트씩 쉬프트하지 않고 여러 비트를 쉬프트하는 기술이 필요하다. 즉, 한 클럭 사이클마다 여러 비트가 쉬프트하면 쉬프트되는 비트 수만큼 클럭 사이클이 줄어든다. 예를 들어 한 클럭 사이클에 2비트씩 쉬프트하면 계산시간은 m/2만큼 줄어든다. 이때 한 클럭 사이클에 여러 비트를 쉬프트 하기 위해서는 레지스터 및 게이트수가 증가하게 되는 단점이 있는데 기존의 구조에서와 같이 한 사이클에 2 비트씩 쉬프트하면 $4 \cdot m$의 레지스터가 소요되며 쉬프트되는 비트수를 t로 하면 $t \cdot m$의 레지스터가 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 Mastrovito[2] 에 의하여 제안되어 유한체 승산기의 가장 기본적인 구조로 자리를 잡고 있다. 직렬 유한체 승산의 표현식은

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

이다. α에 관한 식을 정리하면

$$\alpha^m = p_0 + p_1\alpha + \cdots + p_{m-1}\alpha^{m-1} \text{ 이므로}$$

$$\begin{aligned} A\alpha &= a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}\alpha^m \\ &= a_0\alpha + a_1\alpha^2 + \cdots + a_{m-1}(p_0\alpha + p_1\alpha_2 + \cdots + p_{m-1}\alpha^{m-1}) \end{aligned}$$

$$\begin{aligned} &= a_{m-1}p_0 + (a_0 + a_{m-1}p_1)\alpha + (a_1 + \\ &a_{m-1}p_2)\alpha^2 + \cdots + (a_{m-2} + a_{m-1}p_{m-1})\alpha^{m-1} \end{aligned} \tag{2}$$

(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. 그림 1의 회로에서 ⊕ 표기는 exclusive OR 게이트를 표시하며 a 와 b로 표기된 사각형은 각각 승산에 사용되는 입력데이터를 한 비트씩 저장하는 레지스터를 표시한다. 또한, z로 표기된 사각형은 승산 결과 데이터를 저장하는 레지스터 1비트를 표시한다. 그림 1에서 m은 비트수로 a, b, z의 데이터가 각각 m 비트 수임을 나타낸다. 그림 1과 같은 구조의 기본적인 LFSR구조의 직렬 유한체 승산기는 m비트의 데이터를 승산 하는데 $3 \cdot m$ 개의 레지스터가 필요하고 m개의 클럭사이클수의 계산 속도를 갖는다. 직렬 유한체 승산기는 단순한 구조로 하드웨어 자원을 많이 소모하지 않지만 m이 커질수록 계산 결과가 늦어지는 단점이 있다.

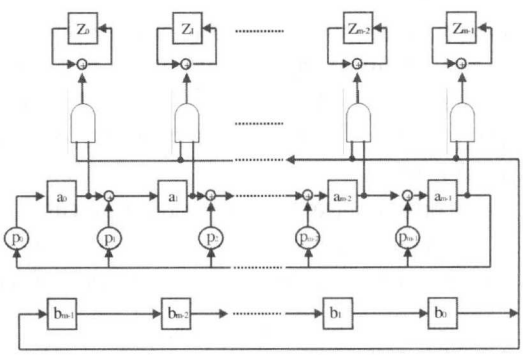


그림 1. LFSR구조를 이용한 직렬유한체승산기

기본적인 LFSR구조의 직렬 유한체 승산기의 처리속도를 개선하기 위하여 여러 가지 연구가[3] 진행되었고 최근에는 그림 2와 같은 구조가 발표되었다.[4] 이 구조는 식(2)를 2부분으로 나누어 2배의 속도향상을 얻는 방법으로 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.

$$\begin{aligned} Z_{even}(x) &= [b_0A(x) + \ldots + b_{m-3}x^{m-3}A(x) \\ &\quad + b_{m-1}A(x)] \bmod P(x) \\ Z_{odd}(x) &= [b_1xA(x) + b_3x^3A(x) + \ldots \\ &\quad + b_{m-2}x^{m-2}A(x)] \bmod P(x) \\ &= x[b_1A(x) + b_3x^2A(x) + \ldots \\ &\quad + b_{m-2}x^{m-3}A(x)] \bmod P(x) \end{aligned} \quad (3)$$

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

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

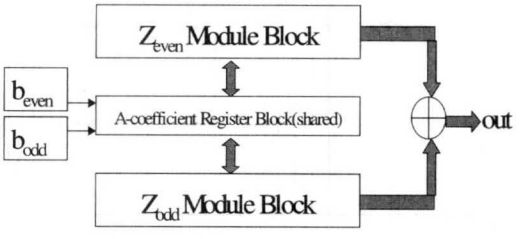


그림 2. LFSR구조를 개선한 2배속 승산기구조

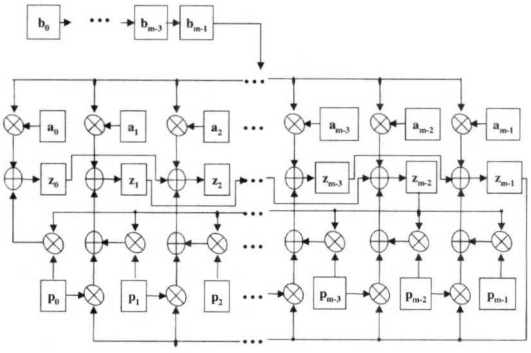


그림 3. 개선된 2배속 승산기의 Z_{even} 모듈의 회로도

본 논문에서는 도 2와 같이 속도를 2배 개선하여도 레지스터가 m개 증가하지 않는 방법을 제안하여 적은 면적의 고속 직렬 유한체 승산기를 구현에 효과적으로 사용되도록 한다.

Ⅲ. 제안된 고속 직렬유한체 승산기

3-1. 레지스터 수의 증가가 없는 회로구조

기존의 개선된 2배속 승산기 구조에서는 유한체 승산기의 계산 속도를 증가시키기 위해서 레지스터의 수를 승산 데이터의 비트 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 레지스터나 exclusive 게이트의 수를 크게 증가시키지 않고 그림 1에 나타낸 종래의 유한체 승산기 보다 계산 속도를 2배 증가 시킬 수 있다.

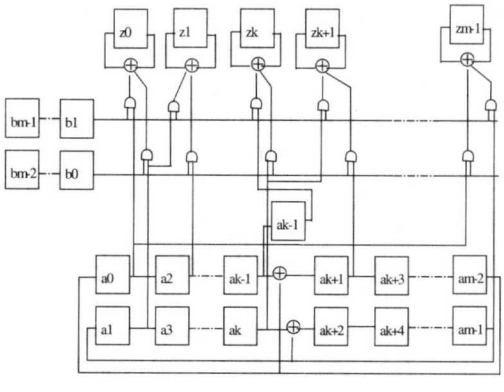


그림 4. 제안된 직렬유한체 승산기 구조

그림 3의 방법은 그림 1의 기본적인 형태의 유한체 승산기와 비교하여 승산속도를 2배 증가하기위

해 승산 결과를 저장하는 Z 레지스터가 2배 크기로 증가하고 exclusive OR 게이트가 2배 증가하며 이를 연결하는 연결선의 복잡도가 크게 증가한다.

그러나 그림 4에 있는 본 논문의 유한체 승산기는 그림 1의 기본적인 형태의 유한체 승산기와 비교하여 레지스터 수에 변화가 없고 단지 궤환(feedback)이 발생하는 지점에서 레지스터가 1개 씩만 추가된다. 또한, exclusive 게이트 수에 변화가 없고 단지 AND 게이트 수가 2배로 증가한다.

본 논문에서 제안하는 그림 4의 회로 구성내용은 다음과 같다. 각각 m 비트의 크기를 갖는 입력 데이터 a, b를 승산하고 그 결과를 m 비트의 z 라고 표시한다. 입력 데이터와 승산결과는 1개의 플립플롭(이하, 레지스터라고 칭함)에 1 비트씩 저장된다. 따라서, 입력 데이터 a는 a0 부터 am-1 까지 m개의 레지스터에 저장되고, 입력 데이터 b는 b0 부터 bm-1 까지 m개의 레지스터에 저장된다. 승산 결과 데이터 z 는 z0 부터 zm-1까지 m개의 레지스터에 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 4에서는 기약다항식이 $p(x) = x^{m-1} + x^k+1$ 일때 회로 구조를 나타낸다. 그림 4에서 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 레지스터에서 궤환이 발생하는데 이때 궤환선을 연결하는 방법은 m-1번째 a 레지스터의 출력을 k번째 a 레지스터의 출력과 exclusive OR의 연산을 한 다음 k+2번째 a 레지스터 입력에 연결한다. 또한, m-1 번째 a 레지스터의 출력을 1번째 a 레지스터의 입력에 연결한다. 특히, 본 논문에서 제안하는 것은 m-2번째 a 레지스터의 출력도 궤환을 시켜야 한다. 즉, m-2번째 a 레지스터의 출력을 k-1번째 a 레지스터의 출력과 exclusive OR연산을 한 후 k+1번째 a 레지스터의 입력으로 연결한다. 또한, m-2번째 출력은 0번째 a 레지스터의 입력으로 궤환한다.

본 논문에서 제안한 구조의 또 다른 특징은 궤환이 발생하는 a 레지스터 보다 한 차수가 낮은 a 레지스터의 비트를 별도의 레지스터를 두어 저장하는 것이다. 이렇게 별도로 저장된 a 레지스터의 값은 z 레지스터에서 승산결과를 연산하는데 사용된다.

3-2. Modular Cell 회로구조

본 논문의 승산기는 기존에 발표된 LFSR구조의 승산기에 비하여 모듈화가 용이한 회로구조를 갖는 것이 두번째 장점이다. 그림 5와 같이 전체 승산기 회로는 3종류의 셀로 모듈화가 가능하다. a레지스터를 2비트씩 묶어서 ACELL이라 하고 b레지스터를 2비트씩 묶어서 BCELL이라 한다. a와 b레지스터가 각각 m개(입력 데이터의 크기가 각각 m비트를 의미함)이면 ACELL과 BCELL의 수는 m/2개가 된다. m이 홀수 이면 m/2 +1개가 된다. 결과값을 저장하는 ZCELL은 m개이다. 각 ACELL과 BCELL을 연결할 때 a레지스터의 홀수번째 비트는 홀수비트에 연결하고 짝수비트는 짝수비트에 연결되도록 하며 b레지스터의 경우도 같은 방법으로 연결된다. 그 결과 a0는 a2에, a2는 a4에 연결되는 모양을 갖으며 a1은 a3에, a3은 a5에 연결되는 모양이 된다. b의 경우도 같은 방법이 적용된다.

그림 5에서 보듯이 ACELL 과 BCELL안에 있는 두개의 레지스터는 같은 클럭에 연결되어 있어 두

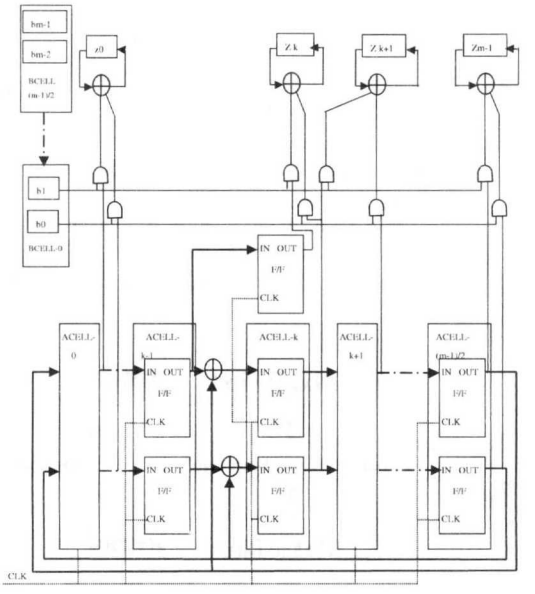


그림 5. 3종류의 셀로 구성된 회로

개의 레지스터가 동시에 데이터를 입력 받게 된다. 그 결과 a레지스터와 b레지스터는 2비트씩 이동(shift)하는 구조를 갖는다. 따라서 m/2 클럭에 a와 b레지스터가 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 레지스터의 구조에 대하여 설명하면 다음과 같다. 1비트씩 쉬프트하는 기존의 승산기와는 달리 b레지스터는 2비트씩 쉬프트(shift)한다. 그림 4와 그림 5에서 보듯이 b레지스터의 b0와 b1 등 2비트가 한 BCELL-0가 되고, b2와 b3가 BCELL-1이 된다. 그리고 클럭에 의하여 한번에 b2는 b0으로, b3는 b1으로 전달된다.

z레지스터의 구조와 승산과정은 다음과 같다. 기존의 승산기에서는 a레지스터의 한 비트와 b 레지스터의 한 비트가 AND연산하고 그 값은 z 레지스터의 내용과 exclusive OR를 하여 다시 z레지스터에 저장된다.

그러나, 본 논문에서는 두개의 b 레지스터 비트와 두개의 a레지스터의 비트가 동시에 AND 연산하고 그 값은 z레지스터의 내용과 exclusive OR를 하여 z레지스터에 저장한다. 두개의 비트끼리 AND연산을 하는 방법은 그림 4, 5에서 보듯이 b0레지스터에는 a1과 a2레지스터의 출력이 연결되고 b1레지스터에는 a0와 a1레지스터가 연결된다. 이와같은 방법을 일반기호로 표시하면 k-1번째 b레지스터는 k번째 a레지스터 및 k+1번째 a레지스터와 AND연산을 하고, k번째 b레지스터는 k-1번째 a 레지스터 및 k번째 a레지스터와 AND연산을 한다.

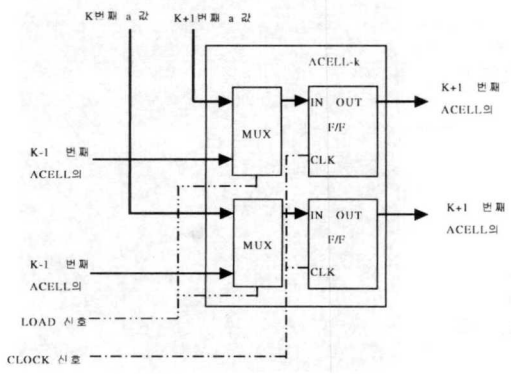


그림 6. A-Cell의 내부 회로구조

그림 6은 ACELL의 내부 회로구조를 나타낸다. 본 논문에서는 승산기의 속도를 개선하기 위하여 유한체 승산 입력 데이터의 하나인 a 레지스터를 2비트씩 묶고 이 를 ACELL이라 칭한다. ACELL은 그림 4, 5에서와 같이 직렬로 묶음으로써 a레지스터의 값이 2비트씩 쉬프트하는 효과를 갖는다. ACELL내부에 있는 멀티플렉서(MUX)는 앞 단의 ACELL로 부터 쉬프트되는 a값을 받거나 외부로부터 a의 초기값을 받을 때 선택하기 위하여 필요하다. BCELL의 회로도는 그림 6의 ACELL 회로도와 동일하게 사용할 수 있다.

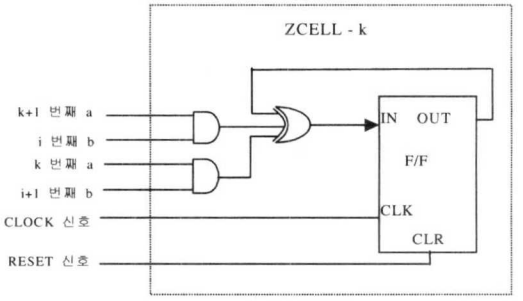


그림 7. k번째 Z-Cell의 내부 회로구조

그림 7과 그림 8은 ZCELL을 나타낸다. ZCELL은 2가지의 형태를 갖는다. ZCELL은 a레지스터의 값과 b레지스터의 값을 AND연산하고 그 결과값과 z레지스터의 값을 exclusive OR연산을 하여 z레지스터에 저장하는 연산을 수행한다. 이때, a레지스터의 비트순서에 따라, AND연산하는 b레지스터의 순서가 다르기 때문에 ZCELL이 2가지 형태를 갖는다.

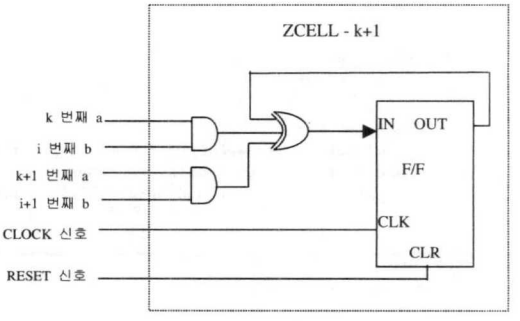


그림 8. k+1번째 Z-Cell의 내부 회로구조

Ⅳ. 회로 검증 및 성능결과

제안된 승산기 회로를 검증하기 위하여 m=193인 승산기를 VHDL로 모델링한 후 SYNOPSYS툴을

이용하여 시뮬레이션을 수행하였다. 피승수 a와 승수 b 그리고 곱의 결과 z는 모두 193비트이고 mode reduction을 위하여 $p(x)=x^{193}+x^{15}+1$인 원시 다항식을 사용하였다. 원시다항식은 다항식기저(polynomial basis) 를 사용하였고 타원곡선 암호기를 구현하는데 널리 사용되는 예제를 이용하였다. 시뮬레이션 결과는 그림 9에 수록하였다. 상단그림은 기존의 LFSR구조의 시뮬레이션 결과이고 하단그림은 본 논문에서 제안된 구조의 시뮬레이션 결과이다.

시뮬레이션에는 같은 승산 데이터를 사용하였고 output이 출력되는 시점은 각각 1.98us와 1.02us로 약 2배의 속도가 개선된 것을 볼 수 있다. 이것은 매 클럭사이클 마다 2비트 씩 승산이 이루어지기 때문에 가능하다. 검증에는 m=193을 이용하였지만 제안된 구조는 어떤 m값에 대해서도 동일한 결과를 얻을 수 있다. 표 1에는 유한체 승산기의 성능결과를 수록하였다.

표 1. 유한체 승산기의 성능결과(GF(2^m)).
(입력 a, b 레지스터, 출력 z 레지스터 포함)

	AND	XOR	REG	CLOCK
Mastrovito 직렬 승산기	m	m	3m	m
기존 승산기[4]	2m	3m	4m	m/2
제안된 승산기	2m	m	3m	m/2

V. 결 론

본 논문은 다항식기저의 유한체 승산기를 설계할 때 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. 승산기 회로의 설계를 용이하게 하기 위하여 3가지 유형의 셀 회로를 제안하고 이 셀을 이용할 경우 승산기 전체회로를 modular scheme으로 설계가 가능하다. 본 논문에서 제안된 승산기는 스마트카드와 같은 적은 크기의 회로를 필요로 하는 전자카드의 암호화 프로세서회로[5] 에서 핵심적인 역할을 할 것으로 기대된다.

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 andTechnology 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.

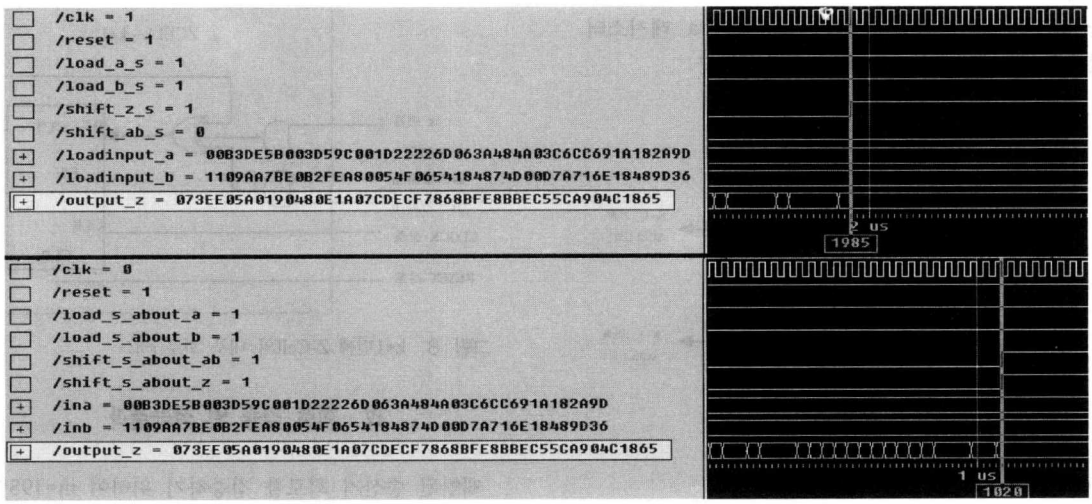


그림 9. 한 m=193 타원곡선 암호프로세서 시뮬레이션결과 (상단) 기존의 LFSR구조를 적용 (하단) 제안된 구조를 적용

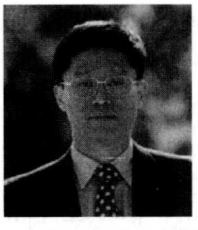
[4] Sangook Moon, J. Park, Y. Lee, "Fast VLSI architecture algorithms for high-security elliptic curve cryptographic applications", IEEE Tr. on Consumer Electronics, Vol; 47, No. 3, August 2001

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

김 원 종(Won-Jong Kim) 정회원

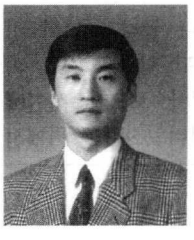
1989년 2월 : 전남대학교 전자공학과 공학사
1982년 2월 : 한양대학교 전자공학과 석사
1999년 2월 : 한양대학교 전자공학과 박사
1999년 3월~2000년 3월 : 한양대학교 공학기술 연구소 선임연구원
2000년 4월~현재 : 한국전자통신연구원 반도체원천기술연구소 선임연구원
<주관심 분야> VLSI CAD, Embedded System 설계, SoC 설계 등

이 광 엽(Kwang-Youb Lee) 정회원



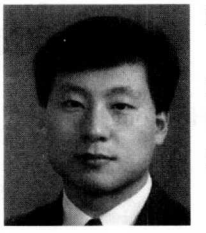
1985년 8월 : 서강대학교 전자공학과 학사
1987년 8월 : 연세대학교 전자공학과 석사
1994년 2월 : 연세대학교 전자공학과 박사
1989-1995년 : 현대전자 선임연구원, 1995년-현재 서경대학교 컴퓨터공학과 조교수
<주관심 분야> 마이크로 프로세서, 암호프로세서

장 준 영(June-Young Chang) 정회원



1985년 2월 : 전남대학교 전산학과 학사
1987년 2월 : 중앙대학교 전산학과 석사
1996년 3월 : 전남대학교 전산학과 박사
1998년 : 대불대학교 컴퓨터공학과 전임강사,
1997년~현재 : 한국전자통신연구원 반도체원천기술연구소 선임연구원
<주관심 분야> CAD/VLSI, 논리합성, Codesign, Embadded System, SOC 설계

배 영 환(Young-Hwan Bae) 정회원



1985년 2월 : 한양대학교 전자공학과 석사
1987년 2월 : 한양대학원 전자공학과 석사
1987년 2월~현재 : 한국전자통신연구원 반도체원천기술연구소 책임연구원
<주관심 분야> VLSI CAD, 하드웨어/소프트웨어 통합설계,Embedded System설계 등

조 한 진(Han-Jin Cho) 정회원

1982년 2월 : 한양대학교 전자공학과 졸업
1987년 5월 : New Jersey Institute of Technology 전기공학과 공학석사
1992년 5월 : University of Florida 전기공학과 공학박사
1992년 11월~현재 : 한국전자통신연구원 시스템IC설계팀 팀장
<주관심 분야> SOC 설계 방법론, 무선통신, 멀티미디어 설계