논문 2005-42SD-6-9

재구성 가능한 타원 곡선 암호화 프로세서 설계

(Design of Programmable and Configurable Elliptic Curve Cryptosystem Coprocessor)

이 지 명*, 이 찬 호**, 권 우 석*

(Lee Jee-Myong, Chanho Lee, and Kwon Woo-Suk)

요 약

암호화 시스템은 다양한 표준으로 인해 하드웨어 구성에 많은 어려움이 있다. 본 논문에서는 다양한 암호화 규격을 수용할 수 있는 재구성 가능한 타원 곡선 암호화 프로세서 구조를 제안한다. 제안된 프로세서 구조는 32bit 크기의 입출력 포트와 내부 버스를 가지며 유한체 연산 장치(AU), 입력/출력 장치(IOU), 레지스터 파일 그리고 프로그램이 가능한 제어 장치(CU)로 이루어져 있다. 제어 장치의 ROM에 저장되어 있는 마이크로 코드에 의하여 프로세서에서 사용할 키의 길이와 원시 다항식이 결정된다. 마이크로 코드는 사용자가 프로세서 내부 ROM에 프로그래밍을 통해 저장할 수 있다. 프로세서 내부의 각 장치는 32 bit 크기의 버스로 연결되어 있어 타원 곡선 암호 규격에 무관하게 동작이 가능하므로 32bit 규격의 입출력 포트만 가지고 있으면 새로운 장치로 교체가 가능한 모듈 구조를 갖고 있다. 따라서 소프트웨어적으로 새로운 마이크로 코드를 프로그래밍하고 하드웨어적으로는 필요한 연산 장치의 교체를 통하여 다양한 타원 곡선 암호 체계에 응용될 수 있다. 본 논문에서는 제안된 프로세서 구조를 이용하여 타원곡선 암호화 프로세서를 구현하였으며 그 결과를 기존의 암호화 프로세서와 비교하였다.

Abstract

Crypto-systems have difficulties in designing hardware due to the various standards. We propose a programmable and configurable architecture for cryptography coprocessors to accommodate various crypto-systems. The proposed architecture has a 32 bit I/O interface and internal bus width, and consists of a programmable finite field arithmetic unit, an input/output unit, a register file, and a control unit. The crypto-system is determined by the micro-codes in memory of the control unit, and is configured by programming the micro-codes. The coprocessor has a modular structure so that the arithmetic unit can be replaced if a substitute has an appropriate 32 bit I/O interface. It can be used in many crypto-systems by re-programming the micro-codes for corresponding crypto-system, or by replacing operation units. We implement an elliptic curve crypto-processor using the proposed architecture and compare it with other crypto-processors

Keywords: 타원 곡선, 암호화 프로세서, 유한체 연산, 가변 비트 곱셈기, 마이크로 코드

I. 서 론

최근 유무선 네트워크를 통한 데이터 전송의 증가로 인해 정보를 보호하기 위한 암호화 기술에 대한 중요성을 인식하여 다양한 암호화 알고리즘이 나오게 되었다. 그런 암호화 알고리즘은 분류방식에 따라서 다양하게 나뉘게 되는데 그중 암호화키의 공개 여부에 따라 대칭키/비 대칭키 암호화 알고리즘으로 나뉘게 된다. 대칭키 암호화 알고리즘은 개인 비밀 키를 이용하여 암호화를 수행하며 대표적인 예로 DES(Data Encryption Standard)가 있다. 반면 비 대칭키 암호화 알고리즘은 공개키와 개인키의 두가지를 통하여 동작하게 되며 대표적인 비 대칭키 암호화 알고리즘으로는 RSA(Rivest Shamir Adleman)와 타원곡선암호체계(ECC: Elliptic Curve Cryptography)가 있다. 이러한 암호화 알고리즘에 필요한 키들은 프로세서 기술의 개발로 인한 더 높은 안전도를 보장하기 위해 더 많은 키 길이를 요구되는데 키 길이가 늘어감에 따라 암호화 알고리즘용 프로세서의 연산 시간이 크게 증가하거나 비슷한 성능을 유

* 학생회원, ** 정회원, 숭실대학교 정보통신전자공학부
(School of Electrinic Engineering, Soongsil University)
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
접수일자:2005년2월21일, 수정완료일: 2005년5월23일

지하기 위해서는 하드웨어적으로 전체 면적이 크게 증가하게 되었다. 따라서 짧은 키 길이를 가지면서도 높은 안전도를 갖는 암호화 알고리즘의 중요성이 대두되고 있다.

여러 가지 암호화 알고리즘 중 ECC 알고리즘은 구현된 공개 키 암호 알고리즘 중 비트 당 가장 높은 안전도를 갖는다[8]. 예를 들어, 1,024bit 키를 갖는 RSA 알고리즘은 160bit 키를 갖는 ECC 알고리즘은 같은 안전도를 보인다[2]. 작은 키 길이는 시스템의 구현에 있어서 보다 작은 메모리 공간과 처리 전력을 필요로 하므로 시스템을 적용하는데 유리하게 한다. 기존의 여러 타원곡선 암호와 관련된 표준들에서 안전도를 고려한 160 비트 이상의 권장 타원곡선들을 제시하고 있으며 이러한 타원곡선에서의 연산은 그 바탕을 유한체 연산에 두고 있다.

유한체 연산은 다양한 암호화 알고리즘이 이용되고 있으나 같은 암호화 알고리즘이라도 체 연산을 위한 키의 크기에 따라 하드웨어의 구조가 바뀌어야하므로 암호화의 키 크기가 바뀔 때마다 암호화를 위한 하드웨어가 바뀌어야하는 어려움이 있다. 이는 소프트웨어 방식으로 처리하면 해결이 가능하지만 휴대용 기기 등에서는 고성능 프로세서를 사용하기 어렵고 하드웨어 방식에 비해 효율과 전력 소모 면에서 크게 불리하다. 반면에 전용 프로세서를 사용하면 그러한 문제는 어느 정도 해결되지만 암호 키의 증가나 암호 체계의 변화에 대응할 수 없다는 문제가 있다. 따라서 전용 하드웨어의 장점에 암호 체계의 변화에 대응할 수 있는 해결 방법이 필요하다.

본 논문에서는 이러한 문제점을 해결하기 위한 새로운 암호 프로세서의 구조를 제안한다. 제안하는 프로세서 구조의 특징은 프로그램 가능한 마이크로 코드 방식의 연산 제어 구조와 임의의 크기의 유한체 연산이 가능한 가변 곱셈기, 그리고 모듈화 된 프로세서 구조이다.

제안된 프로세서는 마이크로 코드 방식을 적용하여 내부 ROM에 저장되는 코드의 제어 동작에 따라 유한체 연산이 필요한 다양한 분야에 응용가능하다. 암호프로세서는 유한체 연산에 기반을 두고 있으므로 기본적인 유한체 연산기가 있으면 적절한 데이터를 공급하고 연산 방법을 제어하여 서로 다른 종류의 암호화 과정도 동일한 프로세서에서 수행이 가능하다. 이를 위해 제어 회로를 하드웨어가 아닌 소프트웨어 방식의 마이크로 코드를 이용하여 구성하여 프로세서의 구조 변경 없이 마이크로 코드의 재 프로그래밍으로 기능의 변경이 가능하다. 한편, 기존의 유한체 연산을 위한 곱셈기들을 살펴보면 연산을 수행할 유한체의 크기가 바뀌는 경우 새로운 곱셈기로 교환해주고 제어회로를 수정해야 했다. 이는 하드웨어의 재설계를 요구한다. 이러한 문제를 해결하기 위해 유한체 연산 중 다항식 기저 상에서 비트 수에 의존하지 않고 어떤 비트 수에 해당하는 원시 다항식을 이용하여도 연산이 가능한 가변 비트 곱셈기를 제안한다. 제안된 곱셈기는 32bit 기본 곱셈기를 기반으로 하여 유한체의 크기와 원시다항식에 따라 32bit 단위로 연산을 수행하여 최종적으로 요구되는 비트 수의 유한체 연산이 가능하다. 위에서 언급한 두 가지 사항이 가능하도록 프로세서 내부의 각 모듈은 32bit 버스에 연결되어 데이터를 교환하도록 하였으며 입출력 포트 규격만 지키면 다른 모듈로 대체가 가능하도록 하였다. 이를 통해 다른 암호화 체계에서 다른 종류의 연산기가 필요할 경우 쉽게 하드웨어 구조 변경이 가능하다.

본 논문에서는 제안된 프로세서 구조에 따라 유한체의 키 길이가 1bit부터 1,024bit까지 연산이 가능한 ECC 프로세서를 설계하였다. 각종 유한체 연산과 타원곡선 상의 연산은 마이크로 코드를 이용하여 연산방법이 결정되고 제어 신호가 생성되어 가변 비트 곱셈기를 동작시켜 암호화 및 복호화 과정이 진행된다. 암호화 키 길이가 바뀌는 경우 마이크로 코드만을 수정하여 필요한 연산이 가능하다. 데이터를 저장하는 내부 메모리의 전체 저장 공간의 크기는 13kbit (32bit * 416)로 최대 1,024비트의 크기를 갖는 데이터를 13개까지 저장할 수 있다. 모든 데이터는 32bit 단위로 나뉘어 저장된다. 이 메모리의 크기가 키의 길이를 1,024bit로 제한하므로 메모리 크기를 늘리면 키의 길이도 증가시킬 수 있다.

Ⅱ. 가변 비트 곱셈기

1. 곱셈기 구조

기존의 유한체 곱셈기는 하나의 체가 정해지면 그 안에서만 곱셈 연산이 가능하게 설계되었다. 그중 하나인 digit-serial 곱셈기는 시스톨릭 배열에 기반을 두고 있다[4]. 이 곱셈기는 digit 곱셈을 위한 셀로 구성되어 있으며, 입력 값을 digit 단위로 입력 받는다. 유한체의 크기를 f, digit 크기를 d라 하면 곱셈 셀 내에서의 면적은 f/d가 되며, 최장 지연 시간은 f/d에 파이프라인의 개수를 더한 값이 된다. 이 곱셈기는 기존의 다른 곱셈기에

비하여 면적을 줄이는 장점을 갖게 된다.

그러나 기존의 digit-serial 곱셈기는 두 개의 입력중 하나에 대해서만 digit 단위로 곱셈이 가능하여 기본적으로 하드웨어에 의해서 최대로 곱셈 가능한 크기가 결정되고 결정된 크기보다 작은 크기의 연산에서는 하드웨어 낭비가 심하게 된다. 또한 한 번에 연산하는 digit 수가 결정되면 변경할 수 없는 문제점도 가지고 있다. 제안된 가변 비트 곱셈기는 digit-serial 곱셈기의 구조를 개선하고 확장하여 $32b \times 32b$의 연산을 기본으로 하여 유한체의 크기가 변화하면 이를 32bit 단위로 분할하여 연산이 가능한 다항식 기저상의 곱셈기이다. 제안된 구조에서 곱셈의 단위는 32bit가 아닌 16bit나 64bit로 확장하거나 축소할 수도 있다.

앞에서 언급한 바와 같이 가변 비트 곱셈기에서 기본 곱셈 셀은 digit-serial 곱셈기의 구조를 사용한다[4]. 이 경우 최장 지연 시간이 길어지게 되는 단점을 보인다. d=32 일 경우의 셀 내부의 최장 지연 시간은 식 (1)과 같다.

$$t_p = t_{AND} + 64t_{XOR} \quad (1)$$

위의 식에서 t_{AND}와 t_{XOR}는 각각 AND 게이트와 XOR 게이트에서의 지연시간(propagation delay)이다. 이 문제를 해결하기 위해 제안된 곱셈기에서는 곱셈 셀을 두 부분으로 나누었다. 하나는 (A ×B) 연산을 하기 위한 부분이며, 다른 한 부분은 나머지(modulo) 연산을 위한 부분이다. 이 두 부분은 같은 구조를 가지고 있으므로, 곱셈 연산과 나머지 연산이 한 곱셈 셀에서 이루어지게 된다. 그림 1에 나타난 바와 같이 곱셈 셀은 XOR 게이트 트리 구조를 사용하여 지연시간을 줄이게 된다. 제안한 곱셈기의 최장 지연 시간은 식 (2)와 같다.

$$t_p = t_{AND} + 8t_{XOR} \quad (2)$$

이는 digit-serial 곱셈기에 비해 56개의 XOR 게이트를 통과하는 시간만큼 지연 시간이 감소한 것이다. 제안한 곱셈기는 $32b \times 32b$의 연산을 여러 번 진행하여 유한체의 크기가 바뀌더라도 재설계하여 합성을 다시 하지 않고 연산을 수행할 수 있다. 즉 키 길이가 56bit인 경우 데이터는 각각 32bit와 24bit로 나누어 입력되어 총 4번의 곱셈을 통해 $56b \times 56b$의 곱셈이 완료된다. 32bit보다 작은 값의 경우에는 남는 자리수를 0으로 채워 한 번에 연산이 가능하다.

제안한 곱셈기의 전체 구조를 그림 2에 나타내었다. 곱셈기는 그림 1의 기본 곱셈 셀과 입출력 레지스터,

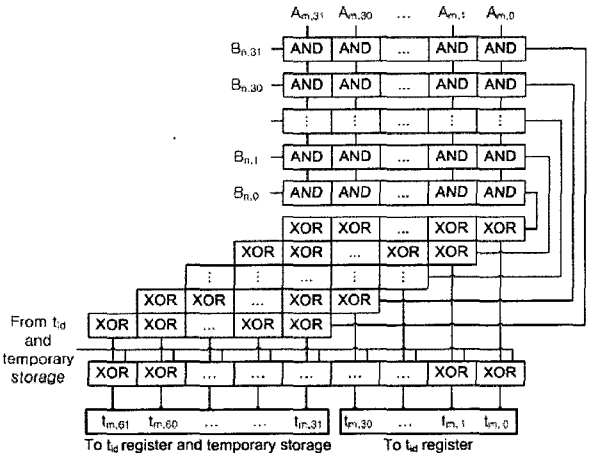

그림 1. 기본 곱셈 셀 구조
Fig. 1. Structure of basic multiplier cell.

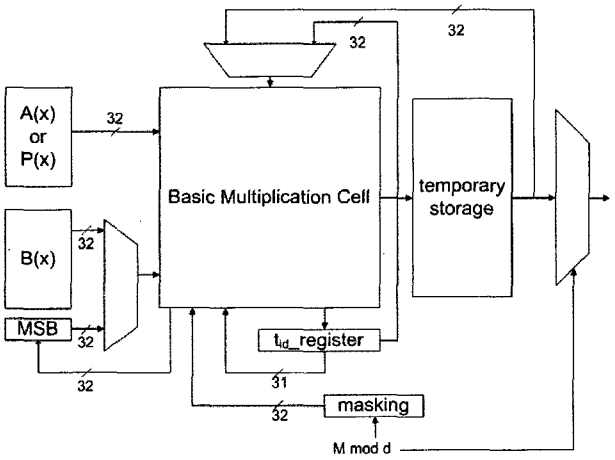

그림 2. 가변 비트 곱셈기의 전체 구조
Fig. 2. Block diagram of variable bit finite field multiplier.

중간 값을 저장하기 위한 임시 메모리(temporary storage), 그리고 두 번 이상의 기본 셀 연산이 필요한 경우 연산 제어를 위한 제어부로 구성되어 있다. 연산 가능한 비트 수에는 제한이 없으며 오직 내부 임시 메모리의 크기에 의해 제한을 받는다.

2. 곱셈기 동작

곱셈기는 외부로부터 연산이 되어야 할 데이터를 받는다. 이 데이터들은 32bit 단위로 입력되며 제어부에 의해 기본 곱셈 셀의 동작이 차례로 진행된다. 한 digit 단위($32b \times 32b$)의 연산이 끝나게 되면 이 값들은 t_{id} 레지스터와 임시 메모리에 저장이 된다. 셀에서 출력되는 결과 값 중 상위 32 비트는 임시 메모리에 저장이 되게 되고, 하위 31비트는 t_{id} 레지스터에 저장된다. 만약 A(x)의 첫 번째 digit이 곱셈 셀에 입력되는 경우에는, 셀에서 출력되는 상위 32bit가 MSD(Most

Significant Digit)에 저장이 된다. 마지막으로 모든 곱셈 연산이 완료가 되고 나면 순차적으로 임시 메모리에 있던 데이터들이 외부로 출력이 된다.

기본 곱셈 셀만으로 곱셈과 나머지 연산을 수행하기 때문에 나머지 연산을 하기 위해서 P(x)에 적어도 하나 이상의 '1'이 있는 지를 확인해야 한다. 여기서 P(x)는 원시 다항식의 digit 부분이다. 만약 P(x)에 적어도 하나 이상의 '1'이 있다면 카운터는 한 클럭 사이클 동안 동작을 중지하게 되며, 이 사이클에서 첫 번째 clock이 인가되는 순간에는 곱셈 셀은 (A ×B) 연산을 수행하게 된다. 이때 A, B와 임시 메모리의 데이터가 곱셈 셀로 입력된다. 이 곱셈 연산이 곱셈 셀에서 끝난 후에 출력되는 결과 값들은 t_{id} 에 저장되고, 바로 다음 clock에 곱셈 셀로 입력된다. 카운터의 동작이 중지된 2번째 clock에서는 바로 전 clock에서 받아들인 tid 레지스터의 데이터와 MSB, P(x)의 데이터로 나머지 연산을 수행한다.

제안한 곱셈기는 32bit 단위의 기본 곱셈 셀을 사용하고 있으므로 만약 유한체의 크기가 32bit의 크기로 나누어지지 않을 때 문제가 발생할 수 있다. 이 문제는 데이터의 하위 자리에 '0'을 채워 넣음으로써 해결을 할 수 있다.

3. 곱셈기 구현 결과

제안한 곱셈기는 GF(2^m)상에서 유한체 연산 중 다항식 기저상의 연산을 수행할 수 있도록 설계되었으며, Xilinx ISE와 Virtex-800 FPGA를 이용하여 합성하였다. 제안된 곱셈기와 동일한 유한체 상의 곱셈 연산을 하는 다른 곱셈기와의 성능 비교를 표 1에 나타내었다. 제안된 곱셈기는 1 - 1,024bit의 연산이 가능하고 한계값인 1,024는 내부 임시 메모리 크기에 의해 정해지는 값으로 메모리 크기를 늘리면 연산 비트 값은 더 증가한다. 곱셈 시간은 연산 비트 수에 따라 반복 연산 횟수가 달라지는데 표 1에 나타난 값은 연산 비트가 193 - 224bit인 경우의 값이다. 기존에 보고된 곱셈기와 비교해서 성능이나 면적에서 우수함을 알 수 있다. Digit-Serial 병렬 유한체 곱셈기의 경우[4] 성능과 면적이 제안된 구조보다 더 우수해 보이나 224bit까지 확장하면 면적은 제안된 곱셈기보다 더 커질 것이고 곱셈 시간도 표 1에 주어진 값보다 더 증가한다. 또한 곱셈기를 재설계해야하는 문제점이 있다.

표 1. 기존의 곱셈기와 제안된 곱셈기의 비교
Table 1. Comparison of implementation results of finite field multipliers.

	[1]	[5]	[4]	Proposed
Basis Type	Standard	Standard	Standard	Standard
Frequency	56 MHz	68 MHz	40 MHz	48 MHz
Multiplication Time[us]	3.45	2.85	0.625	1.62 (193-224bit)
Technology	Virtex 800	Virtex 1000	Virtex 800	Virtex 800
Operation bits	193	193	193	1-1,024
Area (# of Slice)	512	616	2,222	2,413

III. ECC 프로세서

1. ECCP의 동작

제안된 곱셈기를 이용하여 각 기능별로 블록화 되어 모듈화가 가능한 ECC 프로세서를 설계하였다. 제안한 프로세서는 마이크로 코드 방식을 써서 동작을 하게 되는데, 이는 ROM의 제어 연산을 바꿔줌으로써 새로운 형식의 좌표계 상에서도 동작이 가능하도록 되어있다. 또한 모듈화 방식으로 설계되어 연산 유닛(AU: arithmetic unit)과 메모리 등 각각의 블록화 된 부분에 새로운 유닛을 적용할 수 있어 유한체 연산이 포함된 여러 분야에서의 응용이 가능하다.

모듈화 방식의 제안된 암호 프로세서 구조가 그림 3에 나타나 있다. 각 유닛은 입출력 포트의 규격을 만족하면 더욱 개선된 또는 다른 연산을 하는 유닛으로 교체가 가능하며 이는 제안된 가변 비트 곱셈기를 포함하여 modified LFSR(linear feedback shift register) 곱셈기와 Massay-Omura 곱셈기를 사용하여 산술 연산기의 교체를 하였을 경우에도 정확한 결과가 출력되는 것을 확인하여 검증하였다.

제안된 프로세서 구조에 따라 실제 프로세서의 구현을 위해 타원곡선 암호화 시스템을 채택하여 ECCP (ECC 프로세서)를 설계하였다. 기본적으로 이진 유한체 상에서의 타원곡선은 아핀(Affine) 좌표계와 사영(Projection) 좌표계 상에서 정의된다. 아핀 좌표계의 경우 나눗셈 연산을 위해 역원 값을 계산하는 동작이 필요로 하나 사영좌표계의 경우는 곱셈과 덧셈의 동작을 통하여 계산을 하기 때문에 역원을 계산하지 않아도 된다. 그러나 타원 곡선 상에서의 포인트(point) 연산은 두 좌표계 모두 동일하게 포인트 덧셈 연산과 포인트 이 배수(doubling) 연산의 반복을 통하여 이루어지게 된다. 설계된 ECCP는 사영좌표계에서 구현되어 역원 계산이 필요 없다.

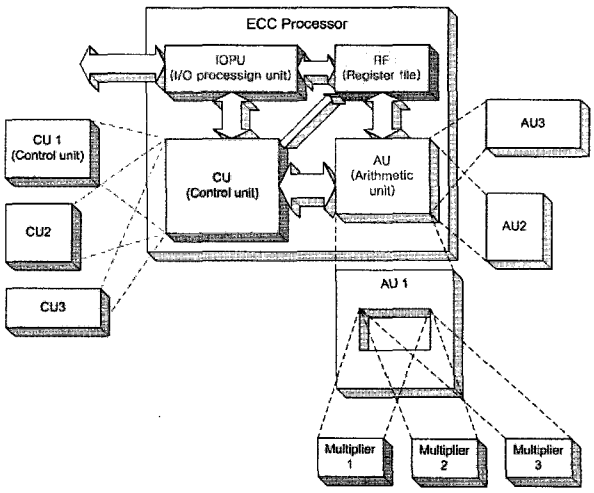

그림 3. 제안된 암호프로세서 구조
Fig. 3. Block diagram of the proposed ECC processor.

표 2는 타원곡선 연산을 위한 내부 제어 명령어를 보여 주고 있다. 이 명령어들은 제어부의 메모리에 마이크로 코드 형식으로 저장되며 이 코드에 따라 연산 방법이 달라진다. 마이크로 코드는 내/외부 명령을 실행하기 위한 제어 신호를 인코딩하여 모아 놓은 것으로 이 코드의 변화에 따라서 프로세서는 그 성능과 기능이 크게 달라질 수 있다. 마이크로 코드는 언제라도 소프트웨어적으로 변경이 가능하므로 성능 개선, 규격의 변경 등에 대해 쉽게 대응할 수 있다. 또한 명령어 자체를 바꾸는 것도 가능하다.

ECCP는 호스트로부터 받은 외부 명령어에 따라 ROM의 주소를 찾아가 동작을 하는데 외부 명령어는 $k \cdot P$ 연산뿐만 아니라 유한체상에서의 덧셈과 곱셈 연산 및 ECCP의 정확한 동작을 검증하기 위한 각각의 포인트 연산을 수행할 수 있게 되어 있다. 또한 일반 명령어인 shift, branch, move 명령어의 수행을 통해 새로운 어플리케이션에 응용이 가능하도록 하였으며 내부 명령어는 외부 명령어인 STR에 의하여 값을 변경할 수 있도록 설계하였다. 표 3에 외부 명령어의 종류와 각각의 이진 코드 값들을 나타내었다.

Host는 명령어와 데이터를 32bit 단위로 보낸다. 첫 입력 데이터의 상위 3bit는 프로세서 외부 명령어가 되고, 그 아래 11bit 상에 현재 연산을 하고자 하는 유한체의 키 길이가 포함되어 키의 길이에 따라 그에 해당하는 기저 다항식을 통하여 연산이 가능하다.

제안된 ECCP는 암호화/복호화를 위한 보조 프로세서로 사용되며 호스트에 대해 현재의 상태를 나타내는 3bit 크기의 상태 신호를 보내준다. 이들 상태 신호인

표 2. 프로세서 내부 명령어
Table 2. Internal instruction set of the proposed ECC processor.

명령어	동작
FADD FMUL FINV	유한체 연산을 위한 명령어. 덧셈/곱셈/역원
JCZ	전체 포인트 곱셈 연산을 한 bit 수에 따라 해당 주소로 이동
SHL	저장되어 있는 k 값을 1bit 왼쪽으로 쉬프트.
K_SCAN	k[MSB] 값에 따라 다음 수행할 동작을 지정
MOV_SHL	k 값을 쉬프트 레지스터로 이동시킴
Shift Left Shift Right	Logical Shift Left / Right with 32 bit data
Branch Conditional_ Branch	Main Rom의 주소값을 분기하는 명령어. 무조건 분기 명령어 / 비교하는 data의 두 값이 동일한 경우 분기하는 명령어.
(MOVE) mem to ctrl ctrl to mem mem to mem	내부 data의 값을 이동하는 명령어. CU과 메모리 사이에 data가 이동하는 동작과 메모리의 한 주소에서 다른 주소로 이동하는 동작에 따라 구분.
END	모든 동작이 끝났음을 호스트에 알림

표 3. 프로세서 외부 명령어
Table 3. External instruction set of the proposed ECC processor.

연산 동작	명령어	이진 코드
Field Arithmetic Instruction	Field Addition	001
	Field Multiplication	010
	Field Inversion	110
Point Instruction	Point Addition	011
	Point Doubling	100
kP Instruction	Point Multiplication	101
STR	Store Instruction	111
nop	No operation	000

'READY', 'BUSY', 'DONE'의 값에 따라 호스트에서는 ECCP의 상태를 판단한다. 'READY' 상태는 ECCP에서 아무런 동작을 하지 않는 상태로 외부 명령어와 데이터를 받아서 그에 해당하는 명령을 수행할 수 있는 준비 상태이고 'ecp_on' 신호가 인가되면 외부 명령어와 입력 데이터가 32bit 버스로 전달되고 'BUSY' 상태로 넘어가게 된다. 'BUSY' 상태는 ECCP에서 명령을 수행하고 있는 상태이다. 모든 연산이 종료되면 'OPDONE' 신호가 발생하며 'DONE' 상태로 넘어가게 된다. 'DONE' 상태는 모든 연산이 수행된 후에 그 결과 값을 외부로 출력하는 상태로 모든 결과 값이 출력된 후에는 다시 'READY' 상태로 넘어가게 된다.

2. ECCP의 구조

그림 4에 나타난 바와 같이 제안된 ECCP는 크게

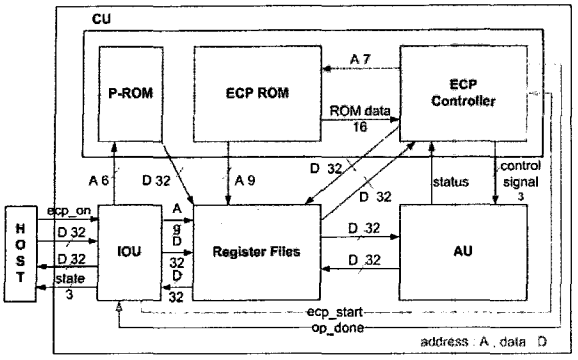

그림 4. 제안된 ECCP 전체 구조
Fig. 4. Block diagram of the proposed ECC processor.

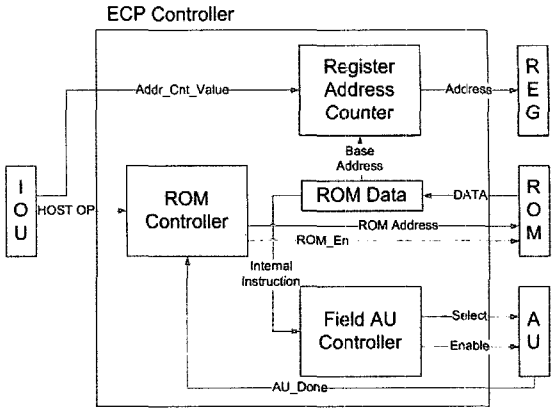

그림 5. CU의 내부 구조
Fig. 5. Block diagram of CU structure.

IOU(input/output unit), AU(field arithmetic unit), CU(control unit), Register Files 의 4개의 모듈로 이루어져 있으며 CU는 P-ROM, ECP MAIN ROM, ECP controller의 3 부분으로 이루어져 있다.

제안된 ECCP의 전체 동작을 살펴보면 IOU를 통하여 host에서 오는 명령어와 데이터를 받아들여 Register File에 저장한다. 이후 CU에서 명령어를 디코딩하고 해당 마이크로 코드를 실행시켜 연산에 필요한 값이 저장되어 있는 Register File의 주소와 제어신호를 생성하여 Register File에서 필요한 데이터를 이용해 유한체 연산을 통하여 나온 결과 값을 다시 host로 보내는 동작을 하게 된다.

내부 블록들을 살펴보면 IOU는 호스트와의 통신을 위한 유닛으로 연산에 사용되는 데이터와 외부 명령어를 호스트로부터 받는다. 모든 연산이 끝난 후에 호스트로 프로세서의 상태와 연산 결과 값을 보내주는 역할을 한다.

CU는 전체 연산을 위한 내부 명령어가 들어있는 ECP MAIN ROM과, 내부 명령어를 통하여 ECC 프로세서의 전체 동작을 제어하는 ECP Controller, 그리고 원시 다항식 값이 저장되어 있는 P-ROM으로 구성되어 있다. 설계된 프로세서에 사용 가능한 감산 다항식의 값들은 KC-ECDSA 표준 권고 타원 곡선인 163, 193, 233, 283, 409, 571bit를 기반으로 하여 P-ROM 에 저장되어 있으며, 이 다섯 가지의 키 길이와 원시 다항식에 대해서는 별다른 변경 없이 연산이 가능하고 이외의 다른 키 길이나 원시 다항식에 대해 연산이 필요한 경우에는 P-ROM 상의 원시다항식을 변경한 후 1,024 bit보다 작은 크기를 갖는 경우는 host에서 명령어를 전송 시 키 길이에 대한 data를 함께 전송해 줌으로 다른 m 값의 타원 곡선에도 이용할 수 있다. 또한 유한체 덧셈과 곱셈만으로 가능한 응용이라면 마이크로 코드의 재구성을 통해 적용이 가능하다. 마이크로 코드의 재 프로그램은 host에 의해 이루어진다. CU는 AU와 IOU의 동작을 제어하며 Register File의 주소를 지정하는 역할을 한다. 그림 5는 CU의 내부 구조를 보여주고 있다.

Register File은 연산에 필요한 값들을 저장하기 위한 메모리이다. 총 11개의 블록으로 구성되어 있으며, 각 블록은 1,024bit(32 ×32)로 이루어져있다. 따라서 총 11,264bit(11 ×1,024)의 데이터를 저장할 수 있다. Register File과 CU의 메모리는 block RAM을 이용하여 설계하였으며 hex 형식으로 명령어 및 감산 다항식을 저장하도록 설계하였다. 따라서 MAIN ROM에 들어가는 hex 코드의 수정으로 ROM의 명령어를 간단하게 교체가 가능하다. 또한 동작 중에도 host에서 프로그램 모드로 변경하여 마이크로 코드를 변경할 수 있다. 따라서 제안된 프로세서는 타원곡선을 위한 암호화 프로세서로서의 기능뿐만 아니라 유한체 연산이 필요한 다른 어플리케이션에서도 사용할 수 있도록 설계되었다.

AU는 유한체 연산을 담당하며 내부에 유한체 덧셈기와 제안된 가변 비트 유한체 곱셈기가 포함되어 있어 CU와의 통신을 통해 동작한다. 다른 암호 체계 또는 응용에서 주어진 연산기로 할 수 없는 기능을 요구하는 경우 AU에 필요한 연산 기를 추가하면 CU의 제어를 통해 실행이 가능하다.

ECCP의 내부 버스는 32bit 크기로 설계하여 외부와 32bit 단위로 통신하고 AU는 인터페이스 규격만 맞추면 새로운 unit으로 교환해 줄 수 있어 다양한 유한체 상에서의 연산이 가능하도록 설계되어 있다. 표 4에서 각 모듈의 동작을 정리하고 있다.

표 4. ECCP 내부 구조
Table 4. Summary of the operation units of the proposed ECC processor.

Unit		설명
IOU		HOST와의 통신을 위한 unit.
Register File		data들이 저장되는 unit .
AU		유한체 연산용 Unit.
CU	ECP ROM	각 동작을 수행하기 위한 명령어와 데이터들의 주소가 저장된 ROM 기억 장치
	ECP controller	프로세서의 동작을 제어하기 위한 unit.
	P-ROM	Field Multiplication 연산을 위한 원시다항식이 저장된 ROM 기억 장치.

3. ECCP 구현 결과

제안된 프로세서는 기존 프로세서와의 비교를 위하여 Xilinx XCV800과 Atmel device를 이용하여 합성하였다. 비교 결과가 표5에 나타나 있다. 설계된 ECCP는 m=155, 36MHz의 동작 주파수에서 유한체 곱셈 연산 시 약 1.34us(48 clocks), 포인트 덧셈의 경우 7번의 유한체 덧셈 연산과 14번의 유한체 곱셈 연산을 통하여 약 23us(820 clocks), 그리고 포인트 이배수 연산 시에는 4번의 유한체 덧셈 연산과 12번의 유한체 곱셈 연산을 통하여 약 19.1us(682 clocks)가 소요된다. 또한 포인트 곱셈 연산 시에는 m=155일 때, k 값에 따라 한 번의 포인트 덧셈만 하는 경우부터 최대 154번의 포인트 덧셈 연산과 154번의 포인트 이배수 연산을 하게 되며 평균 6.05ms(216,048 clocks)의 시간이 소요된다. 이는 초당 평균 165번의 연산이 수행됨을 알 수 있다. 223bit 크기로 연산을 할 경우 동일한 프로세서에서 마이크로 코드의 재구성으로 가능하며 평균 9.9ms(351,179 clocks)의 시간이 소요됨을 확인할 수 있었다. 이 때 동일한 프로세서를 이용하므로 면적은 같다. 설계된 프로세서는 이런 방식으로 1,024bit까지 연산이 가능하고 비트수가 증가함에 따라 32bit 단위의 불연속적으로 연산 시간만 증가한다. 반면에 기존의 프로세서 구조는 연산 비트가 증가하면 연산시간과 함께 면적도 같이 증가하고 재설계를 해야 한다. 따라서 제안된 프로세서의 설계 결과가 기존의 결과에 비해 우수함을 알 수 있다. 유한체 크기에 따른 포인트 곱셈 시간의 변화가 그림 6에 나타나 있다. 1,024bit 이상의 연산이 필요한 경우 프로세서 내부의 Register file을 증가시키고 마이크로 코드를 재구성하면 연산이 가능하다.

표 5. 기존의 프로세서와의 비교
Table 5. Comparison of implementation results of ECC processors.

	[8] (155bit)	Proposed (~1,024bit)	[6] (155bit)	Proposed (~1,024bit)	Proposed (~1,024bit)
Operation Frequency [MHz]	30	36	30	22.5	36
Point multiplication execution time [ms]	8.3	6.05 (155bit)	9.6	9.6 (155bit)	9.9 (223bit)
Operation Cycle [clocks]	N/A	217,857	N/A	217,857	353,179
Target Device	Xilinx XCV300	Xilinx XCV300	Atmel	Atmel	Xilinx XCV300
Area (slices)	1,868	3,473	8,122	5,316	3,473

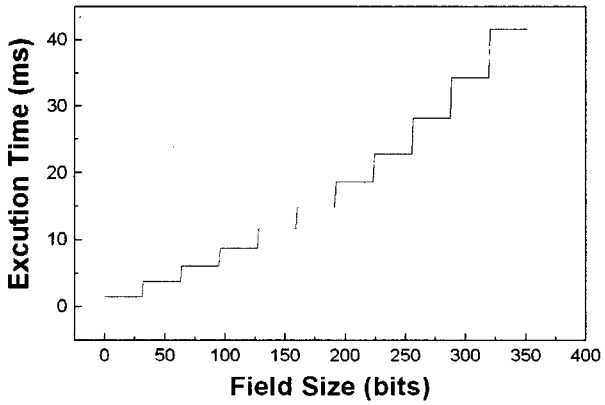

그림 6. 유한체 크기에 따른 포인트 곱셈 시간 비교
Fig. 6. Comparison of the point multiplication time with each finite field size.

Ⅳ. 결 론

본 논문에서는 가변 비트 곱셈기와 그 곱셈기를 이용한 타원 곡선 암호 프로세서 구조를 제안하고 그에 따라 타원곡선 암호 프로세서를 설계하여 동작을 검증하였다. 기존의 암호 프로세서는 키 길이가 바뀌거나 암호체계가 달라지면 하드웨어를 재설계해야 했으나 제안된 프로세서는 가변 비트 곱셈기와 마이크로 코드를 이용한 제어 방식을 채택하여 하드웨어 재설계 없이 실시간으로 프로그래밍을 통해서 소프트웨어적으로 프로세서를 재구성하는 것이 가능하다. 또한 프로세서의 내부 기능 블록을 모듈화 하여 하드웨어 변경의 경우에도 기능 블록의 교체가 쉽도록 하였다. 이를 통해 연산기가 지원하는 범위 내에서는 마이크로 코드의 설계를 통해 다양한 응용이 가능하다.

구현된 프로세서 및 곱셈기는 1,024bit 크기의 유한체까지 연산이 가능하지만 내부 메모리의 크기를 증가시

키면 1,024bit 이상의 키를 갖는 값도 연산이 가능하다. 설계된 가변 비트 곱셈기는 Xilinx Virtex FPGA를 이용하여 합성하였고 2,413 slice의 면적과 48MHz의 동작 주파수를 갖는다. 224bit 유한체 곱셈 연산의 경우 1.62us가 소요된다. ECCP는 3,473 slice의 면적을 차지하고 36MHz 동작 주파수에서 223bit 포인트 곱셈 연산 기준으로 평균 9.6ms의 연산 시간을 보였다.

참 고 문 헌

[1] P. Kitsos, G. Theodoridis, and O. Koufopavlou, "An efficient reconfigurable multiplier architecture for Galois field GF(2^m)," *Microelectronics Journal,* Vol. 34, pp. 975-980, 2003.

[2] "SEC2: Recommended Elliptic Curve Domain Parameters. v. 1.0," Certicom Corp, pp. 29-32, Sept. 2000.

[3] "부가형 전자 서명 방식 표준 - 제 3부 타원 곡선을 이용한 기반 전자서명 알고리즘," 한국 정보통신 기술 협회, pp. 8-61, Nov. 2001.

[4] L. Song, and K. K. Parhi, "Low-energy digit-serial parallel finite field multipliers," *Journal of VLSI signal processing,* Vol. 19, pp. 149-166, 1998.

[5] C. K. Koc, and T. Acar, "Montgomery multiplier in GF(2k) design," *Codes and Cryptography,* Vol. 14, pp.57-69, 1998.

[6] S. Janssens, J. Thomas, W. Borremans, P. Gijsels, I. Verbauwhede, F. Vercauteren, B. Preneel, and J. Vandewalle, "Hardware/Software co-design of an elliptic curve public-key ctyptosystem," *2001 IEEE Workshop on Signal Processing Systems,* pp.209-216, Antwerp, Belgium, Sept. 26-28, 2001.

[7] J. H. Guo, and C. L. Wang, "Digit-serial systolic multiplier for finite fields GF(2^m)," *Computers and Digital Techniques, IEE Proceedings,* Vol. 145, pp. 143-148, March 1998.

[8] K. H. Leung, K. W. Ma, W. K. Wong, and P. H. W. Leong, "FPGA implementation of a microcoded elliptic curve cryptographic processor," *2000 IEEE Symposium on Field - Programmable Custom Computing Machines,* pp. 68-76, Napa Valley California USA, April 17-19, 2000.

저 자 소 개

이 지 명(학생회원)
2005년 2월 숭실대학교 정보통신전자공학부 졸업.
2005년 3월~현재 숭실대학교 대학원 전자공학과 재학 중.
<주관심분야 : 3차원 그래픽 가속기 설계>

권 우 석(학생회원)
1997년 숭실대학교 정보통신전자공학부 입학
<주관심분야 : 디지털 회로 설계>

이 찬 호(정회원)
1987년 서울대학교 전자공학과 학사졸업
1989년 서울대학교 대학원 전자공학과 석사졸업
1994년 University of California, Los Angeles 전자공학과 박사졸업.
1994년 8월~1995년 2월 삼성전자 반도체연구소 선임연구원.
1995년 3월~현재 숭실대학교 정보통신전자공학부 부교수.
<주관심분야: 채널코덱의 VLSI 구현, SoC on-chip-network, 3차원 그래픽 가속기, SoC 설계 방법론>