유한 필드 GF(2^m)상의 모듈러 곱셈기 및 제곱기 특성 분석

(Characteristic Analysis of Modular Multipliers and Squarers for GF(2^m))

한 상 덕* (Sang-Duk Han) 김 창 훈* (Chang-Hoon Kim) 홍 춘 표** (Chun-Pyo Hong)

요 약 본 논문에서는 타원 곡선 암호화 시스템 등에 응용되는 유한 필드 GF(2^m)상의 모듈러 곱셈기 및 제곱기에 대한 처리 시간과 공간 복잡도를 비교 분석하였다. 이를 위하여 기존에 제시된 모듈러 곱셈기 및 제곱기를 설계하였으며, 이들을 VHDL로 기술한 후 회로를 합성하였다. 합성된 회로에 대한 기능 및 timing 시뮬레이션 결과 모두 정확한 결과 값을 얻었다. 합성된 모듈러 곱셈기 및 제곱기를 FPGA로 구현한 결과 한 클럭당 처리 시간은 시스톨릭 구조가 가장 빠르지만 지연 시간을 고려한 전체 처리 시간은 CA 구조가 가장 빠르다는 결과를 얻었다. 또한 공간 복잡도 특성에 있어서는 LFSR 구조가 가장 우수하다는 결과를 얻었다.

Abstract This paper analyzes the characteristics of three multipliers and squarers in finite fields GF(2^m) from the point of view of processing time and area complexity. First, we analyze structure of three multipliers and squarers: 1) Systolic array structure, 2) LFSR structure, and 3) CA structure. To make performance analysis, each multiplier and squarer was modeled in VHDL and was synthesized for FPGA implementation. The simulation results show that CA structure is the best from the point of view of processing time, and LFSR structure is the best from the point of view of area complexity.

1. 서 론

유한 필드 GF(2^m)상의 연산들은 오류 제어 코딩, 암호화 등 여러 분야에서 널리 응용되고 있다. 특히 타원곡선 암호 시스템과 같은 암호학적 응용분야에서 유한 필드상의 곱셈 연산은 중요하게 인식되고 있다. 유한 필드상에서 덧셈은 비트별 배타적 논리합으로 간단히 연산된다. 그러나 곱셈, 역원, 지수 연산은 덧셈에 비해 상당히 복잡한 연산을 요구한다. 이들 곱셈, 역원, 지수 연산에서 가장 기본이 되는 연산은 모듈러 곱셈 연산으로서 곱셈 연산은 덧셈 연산에 비해 처리 시간이 길뿐만 아니라 연산 빈도가 높기 때문에 효율적인 곱셈 연산에 관한 관련 연구가 많이 이루어지고 있다[1-7].

본 논문에서는 유한 필드 GF(2^m)상에서 모듈러 곱셈 $A(x)B(x)$ mod $P(x)$을 수행하는 기존의 곱셈기와 곱셈과 제곱을 동시에 수행하는 제곱기를 구현하여 이들의 성능을 처리 시간과 공간 복잡도 측면에서 비교 분석한다. 본 논문에서는 세 가지 형태의 곱셈기 및 제곱기를 분석하였으며, 시스톨릭 곱셈기 및 제곱기[4], Linear Feedback Shift Register (LFSR) 곱셈기[5] 및 제곱기, Celluar Automata (CA) 곱셈기[6] 및 제곱기를 대상으로 했다. 곱셈기 및 제곱기에 적용된 모듈러 곱셈 연산방식은 피연산자 $B(x)$가 Least Significant Bit (LSB) 부터 계산되는 LSB 우선 방식이다.

시스톨릭 곱셈기 및 제곱기 구조를 분석한 결과 입력 값이 연속적으로 입력될 경우 곱셈기 및 제곱기의 결과 값은 초기 지연이 $2m$, LFSR 곱셈기 및 제곱기 구조의 초기 지연은 m-1, CA 곱셈기 및 제곱기 구조는 초기 지연이 생기지 않는다. 3가지 구조 모두 초기 지연 후에 $1/m$ 클럭 사이클 비율로 곱셈과 제곱의 결과 값이 출력된다. 시스톨릭 구조는 기존의 시스톨릭 곱셈기 및 제곱기를 m 비트로 확장 가능하게 구현한 것이며, LFSR 구조에서 곱셈기는 기존의 LFSR 곱셈기를 m 비트로 확장 가능하게 구현한 것이며 곱셈과 제곱을 동시에 처리할 수 있는 제곱기는 새로이 추가하여 m 비트로 확장 가능하게 설계 및 구현하였다. CA 구조에서 곱셈기는 기존의 CA 곱셈기를 구현한 것이며 CA 제곱기는 기존에 제곱기에서 입력값 $A(x)$가 중복으

* 대구대학교 컴퓨터정보공학과 석사과정
** 대구대학교 정보통신공학부 교수

로 입력되어 사용되는 것을 한번의 입력 값 $A(x)$로 곱셈과 제곱의 결과를 얻을 수 있게 변형한 후 구현하였다.

본 논문에서는 각 곱셈기 및 제곱기에 대한 구조 해석을 한 다음, 곱셈기 및 제곱기들을 VHDL 로 기술하였으며, timing 시뮬레이션을 통하여 결과 값이 이론 값과 일치함을 확인하였다. FPGA 구현을 위해 Altera 사의 Quartus II 를 사용하여 speed 최적화한 결과를 처리 시간과 공간 복잡도로 비교 분석하였다.

2. 곱셈 및 제곱 알고리즘

유한 필드 GF(q^m)에서 q는 소수이고, m은 양의 정수이다. 본 논문에서 사용된 필드는 0과 1을 원소로 가지는 GF(2)의 확장 필드인 이진 유한 필드 GF(2^m)이며, 2^m 개의 원소를 가진다. 본 논문에서 구현된 곱셈기 및 제곱기들은 모두 다항식 기저 표현으로 필드 상의 원소들을 표현하였다. 유한 필드 GF(2^m)은 원시 기약 다항식 $P(x)$를 포함하며 GF(2^m)상의 모든 다항식들은 원시 기약 다항식 $P(x)$로 모듈러 연산을 수행하여 차수가 m 보다 작은 다항식으로 표현된다.

$A(x)$와 $B(x)$는 GF(2^m)의 원소이고, $P(x)$는 차수 m인 원시 기약 다항식이다. 곱셈 결과값 $M(x)$는 원시 기약다항식 $P(x)$에 의해 $A(x)B(x) \bmod P(x)$로 정의되며, 제곱 결과값 $S(x)$는 $A(x)A(x) \bmod P(x)$로 정의된다. 이때 다항식 $A(x)$, $B(x)$, $P(x)$, $M(x)$, $S(x)$는 아래의 식 (1)과 같이 표현된다.

$$\begin{aligned} A(x) &= a_{m-1}x^{m-1}+a_{m-2}x^{m-2}+\cdots+a_1x+a_0 \\ B(x) &= b_{m-1}x^{m-1}+b_{m-2}x^{m-2}+\cdots+b_1x+b_0 \\ P(x) &= x^m+p_{m-1}x^{m-1}+p_{m-2}x^{m-2}+\cdots+p_1x+p_0 \\ M(x) &= A(x)B(x) \bmod P(x) \\ S(x) &= A(x)A(x) \bmod P(x) \end{aligned} \quad (1)$$

본 논문에서 구현한 곱셈기 및 제곱기들은 모두 $B(x)$의 LSB부터 곱셈하는 LSB 우선 곱셈 방식이다. 식 (1)에서 곱셈의 결과값 $M(x) = A(x)B(x) \bmod P(x)$ 와 제곱의 결과값 $S(x) = A(x)A(x) \bmod P(x)$의 계산 과정을 전개하면 식 (2) 및 식 (3)과 같다.

$$\begin{aligned} M(x) &= A(x)B(x) \bmod P(x) \\ &= b_0A(x)+b_1[A(x)x \bmod P(x)] \\ &\quad +b_2[A(x)x^2 \bmod P(x)] \\ &\quad +\cdots+b_{m-1}[A(x)x^{m-1} \bmod P(x)] \end{aligned} \quad (2)$$

$$\begin{aligned} S(x) &= A(x)A(x) \bmod P(x) \\ &= a_0A(x)+a_1[A(x)x \bmod P(x)] \\ &\quad +a_2[A(x)x^2 \bmod P(x)] \\ &\quad +\cdots+a_{m-1}[A(x)x^{m-1} \bmod P(x)] \end{aligned} \quad (3)$$

식 (2) 및 식 (3)에서 곱셈과 제곱에 사용되는 $P(x)$ 모듈러 연산부분인 [ ]안에 있는 계산을 3가지 구조의 제곱기에서는 한번만 계산하여 모듈러 제곱기를 구현하였다.

3. 곱셈기 및 제곱기 구조

이 장에서는 본 논문에서 구현한 모듈러 곱셈기 및 제곱기 구조에 대해 설명한다.

3.1 시스톨릭 곱셈기 및 제곱기 구조

3.1.1 시스톨릭 곱셈기

그림 1은 유한 필드 GF(2^m)상에서의 시스톨릭 곱셈기를 나타낸 것이다[4]. $A(x)$, $P(x)$는 Most Significant Bit (MSB) 부터 순차적으로 입력되며, $B(x)$는 LSB 부터 순차적으로 입력된다. $M(x)$의 초기값은 0 이며 $P(x)$와 $M(x)$는 지연 시간을 고려하여 한 클럭 지연 후 값을 입력한다. 연속적으로 입력 값을 주었을 경우 곱셈 결과 값은 초기지연 $2m$ 클럭 후 $1/m$ 클럭 사이클 비율로 결과 값이 한 비트씩 $M^{out}(x)$로 출력된다. 그림 2는 그림 1에 있는 Processing Element (PE) 의 구조를 나타낸 것이며 그림 2에서 사용된 '●' 는 시간 지연 소자로 한 클럭 지연을 나타낸다. 제어 신호 ctl은 PE에서 계산시점이 다른 것을 제어하기 위해 사용되며 Multiplexer (MUX) 의 입력 값을 결정하는데 사용한다. GF(2^m)상에서 시스톨릭 곱셈기는 $2m$ 개의 2 입력 AND 게이트, $2m$ 개의 2 입력 XOR 게이트, 그리고 $2m$ 개의 MUX 로 구성된다.

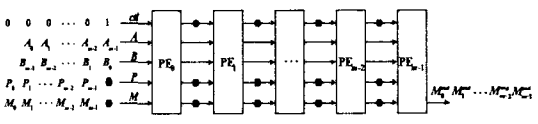


그림 1. GF(2^m)상의 시스톨릭 곱셈기

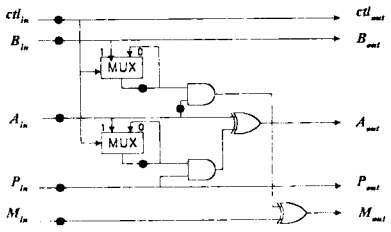


그림 2. 그림 1의 각 PE 구조

3.1.2 시스톨릭 제곱기

그림 3은 m 비트일 때 유한 필드 GF(2^m)상에서의 시스톨릭 제곱기를 나타낸 것이다[4]. 곱셈기에 제곱 기능을 추가하였기에 기본적인 기능은 곱셈기와 동일하다. $M(x)$와 $S(x)$는 각각 곱셈과 제곱의 결과 값을 저장하기 위해 모두 0 을 초기 입력 값으로 가지며 $P(x)$, $M(x)$, $S(x)$는 지연 시간을 고려하여 한 클럭 지연 후 값을 입력한다. 연속적으로 입력 값을 주었을 경우 초기지연 $2m$ 클럭 후 곱셈 결과값 $M^{out}(x)$와 제곱 결과값 $S^{out}(x)$는 $1/m$ 클럭 사이클 비율로 결과 값이 한 비트씩 출력된다. 그림 4는 그림 3에 있는 PE 의 구조를 나타낸 것이며 그림 4에서 사용된 '●' 는 시간 지연 소자로 한 클럭 지연을 나타낸다. 제어 신호 ctl은 곱셈기에서 사용된 것과 동일한 기능을 수행한다. GF(2^m)상에서 시스톨릭 제곱기는 $3m$ 개의 2 입력 AND 게이트, $3m$ 개의 2 입력 XOR 게이트, 그리고 $3m$ 개의 MUX 로 구성된다.

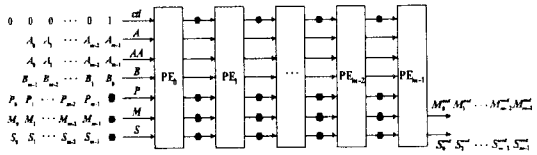


그림 3. GF(2^m)상의 시스톨릭 제곱기

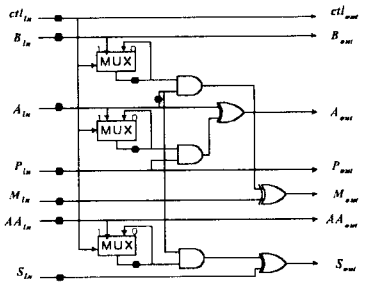


그림 4. 그림 3의 각 PE 구조

3.2 LFSR 곱셈기 및 제곱기 구조

3.2.1 LFSR 곱셈기

그림 5는 m 비트일 때 유한 필드 GF(2^m)상에서의 LFSR 곱셈기 구조를 나타낸 것이다[5]. 입력값 $a(x)$, $p(x)$는 MSB 부터 순차적으로 입력한다. $p(x)$ 모듈러 연산 결과 값이 m-1 클럭 지연 후부터 출력되기 때문에 곱셈 연산을 수행하기 위해서는 입력값 $b(x)$를 m-1 클럭 지연 후 LSB 부터 순차적으로 입력한다. $M(x)$에는 0 을 초기 값으로 입력한다. MUX 의 입력 값은 제어 신호에 의해 결정된다. 입력 값을 연속적으로 주었을 경우 제어 신호가 0 일 때는 $a(x)$의 입력 값을 순차적으로 받아들이고, $a(x)$의 입력 값이 모두 입력되면 제어 신호를 1 로 하여 계산된 값을 계속 활용하여 연산한다. m 번째 클럭부터 $a(x)$의 값은 $b(x)$의 LSB 와 연산되며, 또한 m 번째 클럭에서 $M(x)$는 모두 0 으로 초기화된다. $a(x)$와 $b(x)$의 연산 값에 모듈러 연산을 취한 곱셈 결과 값이 $M(x)$에 저장된다. 곱셈의 결과 값은 m-1 클럭 지연 후 $1/m$ 클럭 비율로 출력되기 시작하여 $2m$-1 클럭 때 최종 곱셈의 결과 값이 출력된다. LFSR 곱셈기는 GF(2^m)상에서 $2m$ 개의 2입력 AND 게이트, $2m$ 개의 2입력 XOR 게이트, 그리고 $2m$ 개의 MUX 로 구성된다.

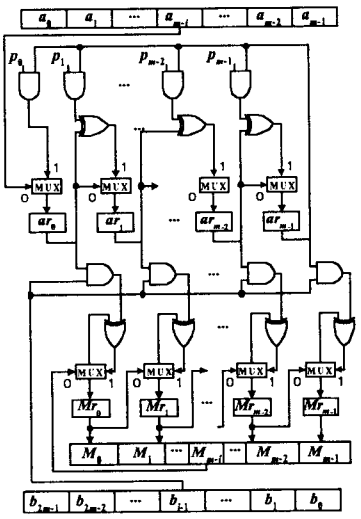


그림 5. GF(2^m)상의 LFSR 곱셈기

3.2.2 LFSR 제곱기

그림 6은 m 비트일 때 유한 필드 GF(2^m)상에서의 LFSR 제곱기를 나타낸 것이다. 제곱기의 기본 구조는 곱셈기와 동일하며 제곱에 이용되는 입력값 $\alpha a(x)$와 결과 값으로 사용될 $S(x)$가 추가되었다. 한번의

$p(x)$ 모듈러 연산으로 곱셈과 제곱을 동시에 처리하는 제곱기를 설계하기 위해 입력값 $aa(x)$를 m-1 클럭 지연 후 입력하였다. 곱셈에서 $p(x)$ 모듈러 연산 결과 값이 m-1 클럭 지연 후부터 출력되기 때문에 제곱 연산을 수행하기 위해서는 입력값 $aa(x)$를 m-1 클럭 지연 후 MSB 부터 순차적으로 입력한다. 곱셈기와 제곱기는 동일한 전체 지연 시간을 가지며, 본 논문의 제곱기에서 곱셈 결과 값과 제곱 결과 값은 동인한 시간에 출력된다. LFSR 제곱기에서 $M(x)$와 $S(x)$는 각각 0 을 초기 입력값으로 가지며 연산 후 곱셈과 제곱의 결과 값이 $M(x)$와 $S(x)$에 저장된다. 제어 신호의 기능은 곱셈기와 동일하다. 곱셈과 제곱의 결과 값은 m-1 클럭 지연 후 1/m 클럭 비율로 출력되기 시작하여 2m-1 클럭 때 최종 곱셈과 제곱의 결과 값이 출력된다. 그림 6을 자세히 분석하면 m 번째 클럭부터 $a(x)$의 입력 값은 $a(x)$의 레지스터 a_{m-i}에 모두 입력되어 $b(x)$의 LSB 와 연산되며 이와 동시에 $aa(x)$의 MSB 와 a_{m-i}의 레지스터 값들이 연산된다. 또한 m 번째 클럭에서 $M(x)$와 $S(x)$의 레지스터 M_{m-i}와 S_{m-i}는 모두 0 으로 초기화된다. $a(x)$와 $b(x)$의 연산 값에 모듈러 연산을 취한 곱셈 결과 값은 $M(x)$에 저장되고, $a(x)$와 $aa(x)$의 연산 결과 값에 모듈러 연산을 취한 제곱 결과 값은 $S(x)$에 저장된다. LFSR 제곱기는 GF(2^m)상에서 3m 개의 2입력 AND 게이트, 3m-1 개의 2입력 XOR 게이트, 그리고 3m 개의 MUX 로 구성된다.

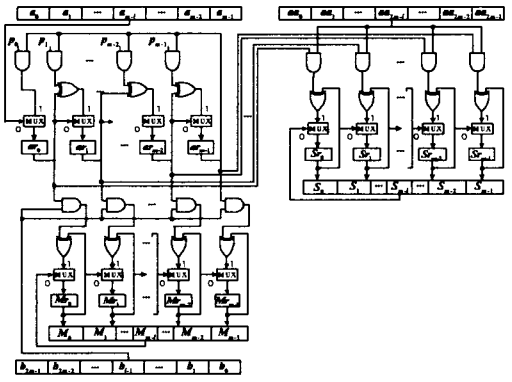


그림 6. GF(2^m)상의 LFSR 제곱기

3.3 CA 곱셈기 및 제곱기 구조

3.3.1 CA 곱셈기

본 논문에서 구현한 CA 곱셈기 구조는 순차적으로 $a(x)$, $b(x)$, $P(x)$, $M(x)$를 입력 값으로 받아 처리하는 위의 두 구조와는 달리 $b(x)$의 값을 제외한 나머지 값들을 한꺼번에 입력 값으로 받아 연산하는 방식이다. CA 구조는 위 두 구조의 연산 방식에 비해 지연 시간이 없어 전체 처리 속도는 빠르지만 입출력 값이 다른 구조에 비해 많아 비트 수를 크게 하여 FPGA로 구현할 경우 많은 수의 입출력 핀 수가 요구된다. 그림 7은 유한 필드 GF(2^m)상에서의 CA 곱셈기 구조를 나타낸 것이다[6]. 본 논문에서 구현한 CA 구조는 Periodic Boundary Cellular Automata (PBCA) 에 의해 매 클럭마다 $a(x)$의 입력 값이 한 비트씩 왼쪽 순환한다[6]. 즉 PBCA에서 $a(x)$의 MSB 는 왼쪽 순환하여 LSB 로 된다. PBCA를 통과한 $a(x)$의 LSB 와 $P(x)$를 연산한다. MUX 는 제어 신호가 0 일 경우 초기 입력값 $a(x)$을 입력으로 받아들이며 제어 신호가 1 일 경우 기존에 계산된 값을 계속 사용한다. MUX 의 출력 값은 $b(x)$와 연산하는 부분과 PBCA에 입력되는 부분으로 나누어진다. 그림 7에서 보는 바와 같이 $a(x)$, $P(x)$, $M(x)$는 동시에 병렬로 입력되고, $b(x)$는 LSB 부터 순차적으로 입력된다. 곱셈 결과 값은 초기 지연 없이 $b(x)$가 입력되는 첫 번째 클럭부터 1/m 클럭 사이클 비율로 출력되기 시작하여 m 클럭 때 최종 곱셈 결과 값이 출력된다. CA 곱셈기는 GF(2^m)상에서 2m 개의 2입력 AND 게이트, 2m-1 개의 2입력 XOR 게이트, 그리고 2m 개의 MUX 로 구성된다.

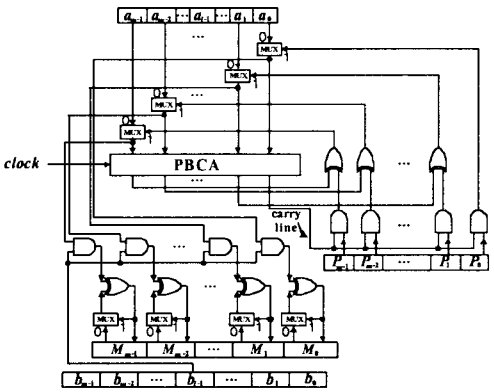


그림 7. GF(2^m)상의 CA 곱셈기

3.3.2 CA 제곱기

본 논문에서 구현한 그림 8의 CA 제곱기는 기존에 곱셈과 제곱을 계산하기 위해 입력값 $a(x)$를 각각 입력해주던 것을 한번의 입력값 $a(x)$로 곱셈과 제곱을 동시에 처리할 수 있게 변형한 것이며, 제곱 연산을 위해 $S(x)$가 추가되었다. 곱셈과 제곱 연산을 위해 $M(x)$와 $S(x)$에 초기값 0 을 입력한다. 곱셈은 CA 곱셈기와 동일하며, 제곱은 한번의 입력값 $a(x)$로 제곱 연산을 처리하기 위해 병렬로 입력되는 입력값 $a(x)$에서 매 클럭마다 MSB 부터 LSB 순서로 한 비트씩

추출하여 $P(x)$ 모듈러 연산 결과 값과 제곱 연산을 수행하였다. 제곱기에서 곱셈과 제곱 결과 값은 곱셈기와 같이 초기 지연 없이 첫 번째 클럭부터 $1/m$ 클럭 사이클 비율로 출력되기 시작하여 m 클럭 때 최종 곱셈과 제곱의 결과 값이 $M(x)$와 $S(x)$에 출력된다. CA 제곱기는 GF(2^m)상에서 $3m$ 개의 2입력 AND 게이트, $3m$-1 개의 2입력 XOR 게이트, 그리고 $3m$ 개의 MUX 로 구성된다.

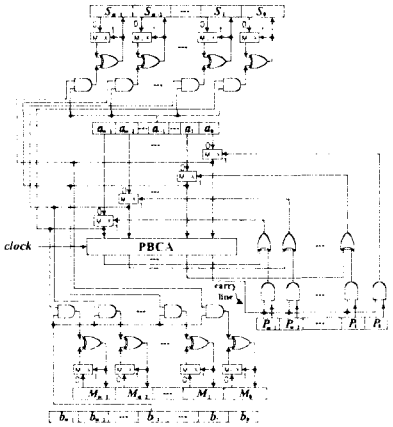


그림 8. GF(2^m)상의 CA 제곱기

4. FPGA 구현

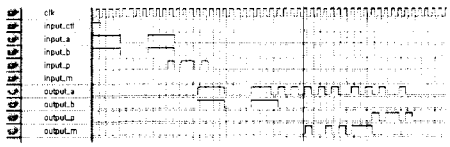
본 절에서는 3 절에서 해석한 각각의 곱셈기 및 제곱기들을 FPGA 로 구현하기 위해 VHDL 로 기술한 후 회로의 정확성을 검증하기 위해 Altera사의 Quartus II (Version 2.0)을 사용하여 funtional 시뮬레이션, place & routing, 그리고 timing 시뮬레이션을 수행하여 결과 값이 이론 치와 정확히 일치함을 확인하였다.

그림 9는 VHDL 로 기술한 파일을 Quartus II 에서 m=16 일 때 timing 시뮬레이션을 수행한 것이며, 그 결과 값이 일치함을 확인하였다. 시뮬레이션 과정에서 입력 값은 식 (4)와 같다.

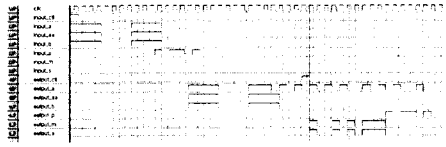
$$\begin{aligned} A(x) &= x^{15}+x^{14}+x^{13}+x^{12}+x^7+x^6+x^5+x^4 \\ B(x) &= x^{11}+x^{10}+x^9+x^8+x^3+x^2+x+1 \\ P(x) &= x^{16}+x^5+x^3+x^2+1 \\ M(x) &= 0 \\ S(x) &= 0 \end{aligned} \tag{4}$$

그림 9의 (a)와 (b)는 시스톨릭 곱셈기 및 제곱기에 대한 timing 시뮬레이션 결과이고, (c)와 (d)는 LFSR 곱셈기 및 제곱기에 대한 timing 시뮬레이션 결과이고, (e)와 (f)는 CA 곱셈기 및 제곱기에 대한 timing 시뮬레이션 결과를 나타낸 것이다. timing 시뮬레이션 결과 입력 값이 연속적으로 입력될 경우 (a)와 (b)는 2m-1 클럭 지연 후 1/m 클럭 사이클 비율로 곱셈과 제곱의 결과 값이 출력되어 3m-1 클럭 때 최종 곱셈과 제곱의 결과 값이 출력되며, (c)와 (d)는 m-2 클럭 지연 후 1/m 클럭 사이클 비율로 곱셈과 제곱의 결과 값이 출력되어 2m-2 클럭 때 최종 곱셈과 제곱의 결과 값이 출력된다. (e)와 (f)는 지연 없이 첫 클럭에 결과 값이 출력되기 시작하여 1/m 클럭 사이클 비율로 곱셈과 제곱의 결과 값이 출력되어 m 클럭 때 최종 곱셈과 제곱의 결과 값이 출력된다. 이때 곱셈 및 제곱 결과 값은 식 (5)와 같음을 확인하였다.

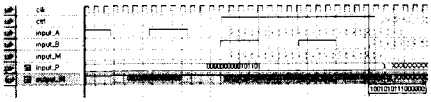
$$\begin{aligned} M(x) &= x^{15}+x^{12}+x^{10}+x^8+x^7+x^6 \\ S(x) &= x^{15}+x^{12}+x^{10}+x^8+x^7+x^6 \end{aligned} \tag{5}$$



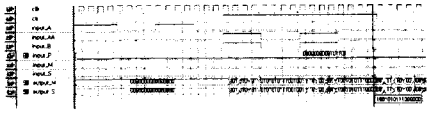
(a) 시스톨릭 곱셈기의 timing 시뮬레이션 결과



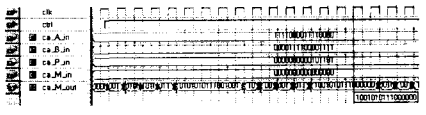
(b) 시스톨릭 제곱기의 timing 시뮬레이션 결과



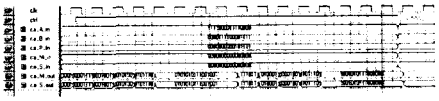
(c) LFSR 곱셈기의 timing 시뮬레이션 결과



(d) LFSR 제곱기의 timing 시뮬레이션 결과



(e) CA 곱셈기의 timing 시뮬레이션 결과



(f) CA 제곱기의 timing 시뮬레이션 결과

그림 9. GF(2^{16}) 상에서 timing 시뮬레이션 결과

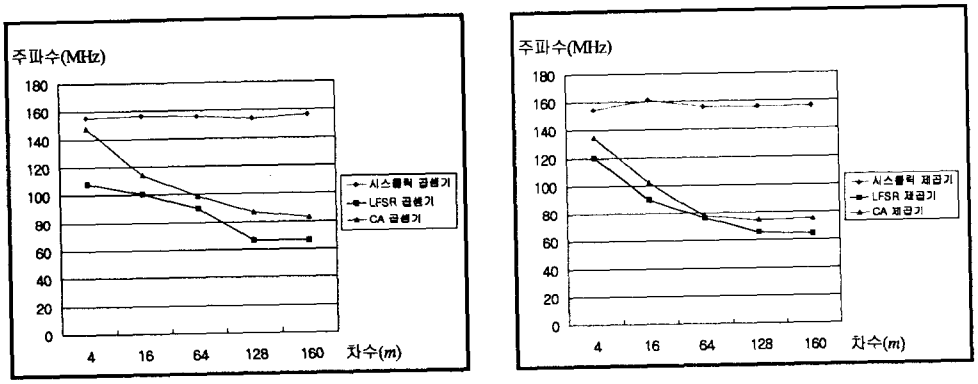
5. 성능분석

본 절에서는 세 가지 구조의 곱셈기 및 제곱기의 특성을 비교하고, 각각을 Quartus II 를 사용해 FPGA 로 구현한 후 Speed 최적화한 결과를 시간, 공간적으로 비교 분석하였다. 본 논문에서 구현된 결과는 입출력 핀으로부터의 지연을 포함한 결과이며, 입출력 핀으로부터의 지연을 고려하지 않고 구현 한 경우 처리 시간은 약 2배 정도 향상되었다. 그림 10 은 차수 (비트 수)의 변화에 따른 $GF(2^m)$상에서 곱셈기 및 제곱기의 시간적 특성을 그래프로 나타낸 것이며, 그림 11은 차수의 변화에 따른 $GF(2^m)$상에서 곱셈기 및 제곱기의 공간적 특성을 그래프로 나타낸 것이다.

<표 1>은 $GF(2^m)$상에서 각 곱셈기 및 제곱기의 하드웨어 복잡도 및 처리 시간을 비교한 것이며, <표 2> 및 <표 3>은 $GF(2^m)$상에서 비트 사이즈 m 을 다르게 하여 곱셈기 및 제곱기를 FPGA 로 구현한 결과이다. FPGA 구현을 위해 Altera사의 Quartus II 2.0 을 사용하였으며, Stratix 군의 EP1S80F1508C-6 디바이스를 target 으로 하였다. CA 제곱기의 구조상 입출력 핀이 많이 사용되기 때문에 이에 적당한 입출력 핀 수를 가진 디바이스 EP1S80F1508C-6 을 선택하였다. $GF(2^m)$상에서 m=160으로 하였을 때 CA 제곱기의 입출력 핀 수는 1,122개이다.

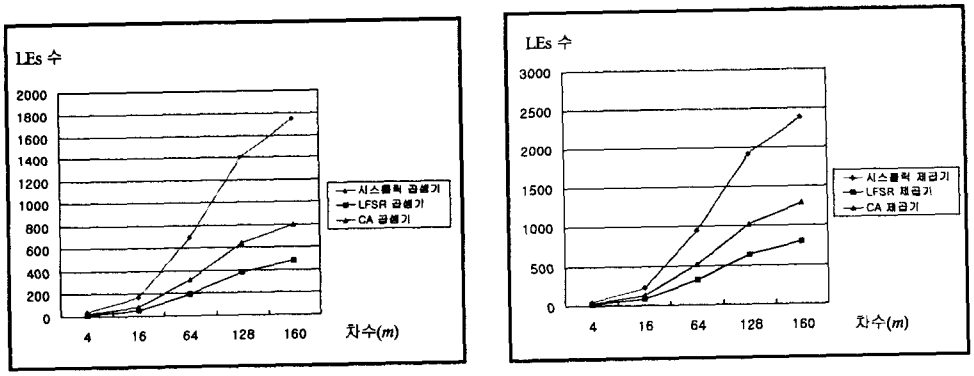
EP1S80F1508C-6 의 최대 사용자 입출력 핀 수는 1,211개로서 이 칩을 이용한 FPGA 구현에는 문제가 없음을 확인하였다.

<표 1>에서 하드웨어 복잡도는 LFSR 구조와 CA 구조가 시스톨릭 구조보다 XOR 게이트 한 개가 더 적다. 그러나 입출력 형식 차이로 인해 구현하였을 때 회로 복잡도는 상당한 차이를 보인다. <표 2> 및 <표 3>에서 FPGA 로 구현하였을 때 칩 이용율은 Stratix 군의 가장 작은 논리적 단위인 LEs 로 표시하였으며, 한 개의 LEs 는 한 개의 4 입력 LUT, 한 개의 programmable register, 한 개의 carry chain 으로



(a) 곱셈기 시간적 특성 (b) 제곱기 시간적 특성

그림 10. $GF(2^m)$ 상에서 곱셈기 및 제곱기의 시간적 특성



(a) 곱셈기 공간적 특성 (b) 제곱기 공간적 특성

그림 11. $GF(2^m)$ 상에서 곱셈기 및 제곱기의 공간적 특성

<표 1> GF(2^m)상에서의 곱셈기 및 제곱기 특성 비교

	시스톨릭 곱셈기	시스톨릭 제곱기	LFSR 곱셈기	LFSR 제곱기	CA 곱셈기	CA 제곱기
입출력 형식	Serial In Serial Out		Serial In Parallel Out		Parallel In Parallel Out	
AND	$2m$	$3m$	$2m$	$3m$	$2m$	$3m$
XOR	$2m$	$3m$	$2m-1$	$3m-1$	$2m-1$	$3m-1$
MUX	$2m$	$3m$	$2m$	$3m$	$2m$	$3m$
Latency(cycles)	$3m$		$2m-1$		m	
Throughput	$1/m$		m		m	
전체 처리 시간	Clock Period * $3m$		Clock Period * $(2m-1)$		Clock Period * m	

<표 2> GF(2^m)상에서의 곱셈기 FPGA 구현 결과

m	항목	시스톨릭 곱셈기	LFSR 곱셈기	CA 곱셈기
16	Area (LEs)	168	48	80
	Clock Period (ns)	6.387	9.932	8.773
	Frequency (MHz)	156.57	100.68	113.99
64	Area (LEs)	696	192	320
	Clock Period (ns)	6.406	11.076	10.101
	Frequency (MHz)	156.10	90.27	99.00
160	Area (LEs)	1752	480	800
	Clock Period (ns)	6.391	15.058	11.936
	Frequency (MHz)	156.47	66.41	83.78
LEs = Logic Elements				

<표 3> GF(2^m)상에서의 제곱기 FPGA 구현 결과

m	항목	시스톨릭 제곱기	LFSR 제곱기	CA 제곱기
16	Area (LEs)	229	80	128
	Clock Period (ns)	6.194	11.074	9.798
	Frequency (MHz)	161.45	90.30	102.06
64	Area (LEs)	949	320	512
	Clock Period (ns)	6.401	13.122	12.764
	Frequency (MHz)	156.23	76.21	78.16
160	Area (LEs)	2389	800	1280
	Clock Period (ns)	6.406	15.566	13.325
	Frequency (MHz)	156.10	64.24	75.05

구성된다[10]. <표 2> 및 <표 3>에 나타낸 것처럼 처리 시간은 두 가지 측면으로 고려할 수 있는데, 첫째, 한 클럭 당 처리 시간은 시스톨릭 구조가 가장 빠르며 LFSR 구조가 가장 느리다. 둘째, 전체 지연 시간을 고려한 처리 시간은 CA 구조가 가장 빠르다는 것을 알 수 있다. 또한 공간 복잡도 측면에서는 LFSR 구조가 가장 간단하며, 시스톨릭 구조가 가장 복잡하다는 것을 알 수 있다.

6. 결론

본 논문에서는 타원 곡선 암호화 시스템 등에 널리 이용되는 유한 필드 GF(2^m)상의 모듈러 곱셈기 및 제곱기들에 대한 처리 시간과 공간 복잡도를 비교 분석하였다. 비교한 곱셈기 및 제곱기는 세 가지로서 시스톨릭 구조, LFSR 구조, 그리고 CA 구조이다. 각 곱셈기 및 제곱기들을 FPGA 로 구현한 다음 특성을 분석한 결과, 한 클럭 당 처리 시간은 시스톨릭 구조가 가장 빠르며 LFSR 구조가 가장 느리다는 결과를 얻었다. 또한 전체 지연 시간을 고려한 처리 시간은 CA 구조가 가장 빠르다는 결과를 얻었다. 마지막으로 공간 복잡도 측면에서는 LFSR 구조가 가장 우수하였다.

참 고 문 헌

[1] R. J. McEliece, Finite Fields for Computer Scientists and Engineers, New York: Kluwer-Academic, 1987.

[2] A. Menezes, Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers. 1993.

[3] S. Y Kung, *VLSI Array Processors*, *Prentice-Hall*, 1987.

[4] 유기영, 김정준, "유한 필드 GF(2^m)상의 시스톨릭 곱셈기 및 곱셈/제곱기", 제 11회 정보보호와 암호에 관한 학술대회 WISC'99 논문집, pp. 375-389, 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.

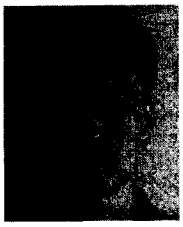
[6] 하경주, 구교민, "셀룰러오토마타를 이용한 LSB 곱셈기 설계", 한국산업정보학회 2001 추계공동학술대회, pp.850-859, November, 2001.

[7] P. Pal. Choudhury, "Cellular Automata Based VLSI Architecture for Computing Multiplication and Inverses In GF(2^m)", IEEE 7th International Conference on VLSI Design, January. 1994.

[8] W. F. Lee, VHDL coding and Logic Synthesis with Synopsis, Academic Press, 2000.

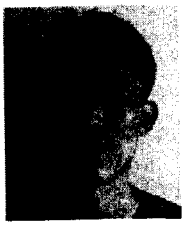
[9] R. K. Dueck, Digital Design with CPLD Applications and VHDL, Delmar, 2001.

[10] Altera, Stratix Programmable Logic Device Family Data Sheet, ver. 2.1, Aug. 2001.



한 상 덕 (Sang Duk Han)

e-mail : sdhan@dsp.taegu.ac.kr
2000년 대구대학교 컴퓨터정보공학과 학사
2000년 ~ 현재 대구대학교 컴퓨터정보공학과 석사과정
<관심분야> 마이크로 프로세서 설계, 암호 시스템, 재구성형 컴퓨팅



김 창 훈 (Chang Hoon Kim)

e-mail : chkim@dsp.taegu.ac.kr
2000년 대구대학교 컴퓨터정보공학부 학사
2000년 ~ 현재 : 대구대학교 컴퓨터정보공학과 석사과정
<관심분야> 암호 시스템, 내장형 시스템, 재구성형 컴퓨팅

홍 춘 표 (Chun Pyo Hong)

e-mail : cphong@daegu.ac.kr
1978년 경북대학교 전자공학과 학사
1986년 Georgia Institute of Technology, Electrical and Computer Engineering 석사
1991년 Georgia Institute of Technology, Electrical and Computer Engineering 박사
1994년 ~ 현재 : 대구대학교 정보통신공학부 교수
<관심분야> DSP 하드웨어 및 소프트웨어, 컴퓨터 구조, VLSI 신호처리, 내장형 시스템