논문 2010-47SC-1-6

시스템 상수의 효과적인 사용을 통한 Galois 필드에서의 고성능 지수제곱 연산 VLSI 설계

(Design of a High Performance Exponentiation VLSI in Galois Field through Effective Use of Systems Constants)

한 영 모*

(Youngmo Han)

요 약

정보보안을 위한 암호화는 종종 Galois Field 상에서 산술 연산의 형태로 이루어진다. 본 논문은 Galois Field 상에서 산술 정보의 지수 연산 처리를 효과적으로 수행하는 방법을 제안한다. 특히 기존의 비트별 병렬 처리 지수 연산기에서 게이트 카운트가 큰 요소를 제거하고, 시스템 상수를 효과적으로 사용하도록 개량함으로써, m 값이 큰 경우에도 고성능인 VLSI 시스템을 설계한다.

Abstract

Encapsulation for information security is often carried out in Galois field in the form of arithmetic operations. This paper proposes how to efficiently perform exponentiation of arithmetic information on Galois field. Espeically, by improving an existing bit-parallel exponentiator to exclude elements with heavy gate counts and to take advantage of system constants, this paper proposes how to implement a VLSI architecture with high performance even for large m.

Keywords: 지수 연산, Gaolis 필드, VLSI, 시스템 상수

Ⅰ. 서 론

정보보안을 위한 암호화는 종종 Galois Field 상에서 산술 연산의 형태로 이루어진다. 이 연산 중 특히 지수 연산의 계산량이 많기 때문에, 효율적인 계산법의 연구가 필요하다.

이러한 취지에서 많은 지수 연산 알고리즘이 제안(square-and-multiply 방법[1~2], pattern-matching 방법[3~4] 등) 되었는데, 특히 참고문헌 [5]에서 비트별 병렬 처리 방법을 제안함으로써 하드웨어 비용과 latency 모두에서 성능향상을 이루었다. 본 논문에서는 참고문헌 [5]에서 소개된 비트별 병렬처리 알고리즘을 시스템 상수를 효과적으로 사용하도록 개량하여 VLSI로 구현시 성능을 더욱 향상하는 방법을 제안한다. 제안된 VLSI 시스템은 특히 차수가 큰 Galois Field에서 효과적인데, 이 특징은 암호화가 차수가 큰 필드에서 수행되는 경우가 많다는 점에서 매우 유용하다.

Ⅱ. 기존의 비트별 병렬 처리 알고리즘 및 구현

본 절에서는 참고문헌 [5]에서 제안한 제안한 $2m$ 차수의 Galois 필드 상에서의 비트별 병렬 처리 알고리즘에 대해 살펴보겠다.

$GF(2^m)$의 임의의 원소 A의 다항식 표현이 $A=(a_{m-1},\cdots,a_1,a_0)$, $a_i\in GF(2)$이고, 지수 E의 이진 표현이 $E=(e_{m-1},\cdots,e_1,e_0)$, $e_i\in\{0,1\}$라고 하

* 정회원, 한양사이버대학교 컴퓨터공학과
(Dept. of Computer Engineering, Hanyang Cyber University)
접수일자: 2009년8월12일, 수정완료일: 2010년1월4일

자. 그러면 $GF(2^m)$의 원시다항식 $F(x)$의 해 α를 사용하여 A의 E제곱을 다음과 같이 곱셈으로 표현할 수 있다.

$$A^E = \prod_{i=1}^{m} pt_{m-i} \tag{1}$$

여기서, $pt_{m-i} = A^{e_{m-i}2^{m-i}}$. 이식을 살펴보면, A^E는 m 회의 곱셈에 의해 계산될 수 있다. 가 이진 디지트이므로, pt_{m-i}는 다음과 같이 단순화 될 수 있다.

$e_{m-i} = 0$인 경우,

$$pt_{m-i} \tag{2}$$

$e_{m-i} = 1$인 경우,

$$pt_{m-i} = A^{2^{m-i}}$$
$$= \left(a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0\right)^{2^{m-i}}$$

관계식:

$(\alpha_1 + \alpha_2 + \cdots + \alpha_t)^{p^r} = \alpha_1^{p^r} + \alpha_2^{p^r} + \cdots \alpha_t^{p^r}$, $r = 1, 2, 3, \cdots$

을 사용하면, 식 (2)는 다음과 같이 고쳐 쓸 수 있다.

$$pt_{m-i} = A^{2^{m-i}} \tag{3}$$
$$= a_{m-1}\alpha^{2^{m-i}(m-1)} + a_{m-2}\alpha^{2^{m-i}(m-2)} + \cdots + a_1\alpha^{2^{m-i}} + a_0$$

m이 짝수이면, 식 (3)의 두 항씩 묶어서 다음과 같이 표현할 수 있다.

$$pt_{m-i} = A^{2^{m-i}}$$
$$= \sum_{k=1}^{m/2}\left(a_{m-2k+1}\alpha^{(m-2k+1)2^{m-i}} + a_{m-2k}\alpha^{(m-2k)2^{m-i}}\right) \tag{4}$$

유사한 방법으로, m이 홀수이면 a_0를 제외한 두 항씩 묶을 수 있다.

순환 회전 (circle rotation) 함수 $\rho()$를 사용하여 식 (4)를 다음과 같이 고쳐 쓸 수 있다.

$$pt_{m-i} = A^{2^{m-i}} \tag{5}$$
$$= \sum_{k=1}^{m/2}\rho^{(m-2k)2^{m-i}}\left(a_{m-2k+1}\alpha^{2^{m-i}} + a_{m-2k}\right)$$

여기서,

$$\left(a_{m-2k+1}\alpha^{2^{m-i}} + a_{m-2k}\right) = \begin{cases} a_{m-2k} & if \quad a_{m-2k+1} = 0 \\ \alpha^{2^{m-i}} + a_{m-2k} & if \quad a_{m-2k+1} = 1 \end{cases}$$

이고, 순환 회전 함수는 $\rho(\beta) = \alpha\beta(\mathrm{mod}\, uloF(\alpha))$로 정의되는데, 임의의 $\beta \in GF(2^m)$에 대해 $\rho^i(\beta) = \alpha^i\beta$의 관계를 만족하는 특성이 있다. 여기서, $\alpha^{2^{m-i}}$는 주어진 $(m-i)$에 대해 상수이다. 식 (5)를 사용하여 pt_{m-i} for $i = 1, \cdots, m-1$를 계산할 수 있다.

반면, pt_0는 다음과 같이 간단한 방법으로 계산될 수 있다.

$$pt_0 = \begin{cases} 1 & if \quad e_0 = 0 \\ A & if \quad e_0 = 1 \end{cases} \tag{6}$$

본 알고리즘을 VLSI로 구현하는 방법을 보이기 위해서, $m = 4$의 경우를 예로 들어 보자. 이 경우, $GF(2^4)$의 원시 다항식으로 $F(x) = x^4 + x + 1$를 선택할 수 있다. 그리고 식 (1)은 다음과 같이 표현된다.

$$A^E = pt_3 \times pt_2 \times pt_1 \times pt_0 \tag{7}$$

식 (7)은 4 bit, pt_k, $k = 0, 1, 2, 3$가 병렬로 계산될 수 있음을 보여주고 있다. 이러한 특성은 그림 1에서 보여주는 것과 같이 비트별 병렬처리 구조를 가능하게 한다. 이 그림에서, E[0:3]는 E의 4 비트 이진수 표현이고, A[0:4]는 A의 4 비트 다항식 표현이다. 출력 OUT[0:3]은 A^E의 4 비트 다항식 표현이다. 그리고 DPT3, DPT2, DPT1, DPT0는 각각 pt_3, pt_2, pt_1 and pt_0를 계산하는 회로이다.

식 (5)는 다음과 같이 표현될 수 있다.

$$pt_3 = \left(a_3\alpha^{3(2^3)} + a_2\alpha^{2(2^3)}\right) + \left(a_1\alpha^{1(2^3)} + a_0\alpha^{0(2^3)}\right) \tag{8}$$

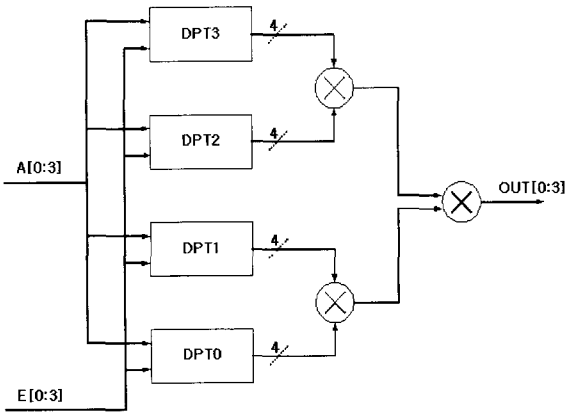


그림 1. $GF(2^m)$상에서의 지수제곱 연산을 위한 VLSI 시스템의 최고 상위 계층

Fig. 1. Top hierarchy of the VLSI system for exponentiation in $GF(2^m)$.

$$pt_2 = \left(a_3\alpha^{3(2^2)} + a_2\alpha^{2(2^2)}\right) + \left(a_1\alpha^{1(2^2)} + a_0\alpha^{0(2^2)}\right) \quad (9)$$

$$pt_1 = \left(a_3\alpha^{3(2^1)} + a_2\alpha^{2(21)}\right) + \left(a_1\alpha^{1(2^1)} + a_0\alpha^{0(21)}\right) \quad (10)$$

그림 2는 DPTk, k=1, 2, 3을 구현한 회로를 보여주고 있다. 이 그림에서 CONV_VECT 블록은 입력된1 비트

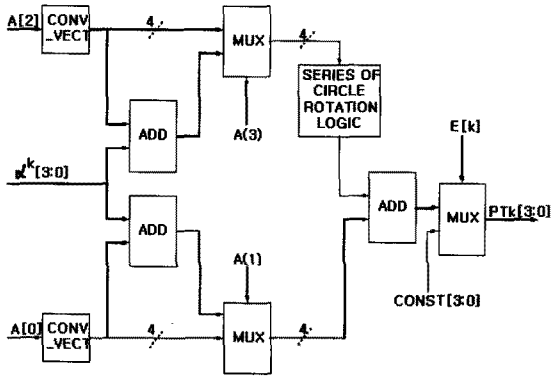


그림 2. k=1, 2, 3인 경우의 DPTk 회로
Fig. 2. DPTk circuit for k=1, 2, 3.

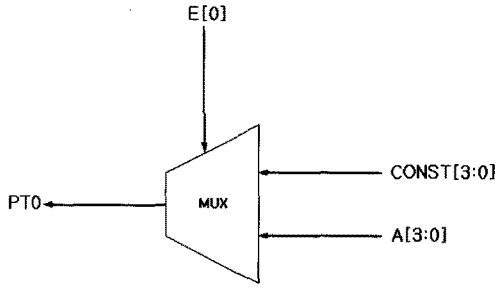


그림 3. k=0인 경우의 DTk 회로
Fig. 3. DPTk circuit for k=0.

표 1. $m=4$인 경우의 지수 제곱 연산을 위한 참고 문헌 [5]의 방법, square-and-multiply 방법, pattern-matching 방법의 latency와 하드웨어 기본 요소 비교

Table 1. Comparision of latency and primitive hardware elements for the method in [5], the square-and-multiply, and the pattern-matching systems for exponentiation in $GF(2^m)$ when $m=4$.

Primitive Element	Square-and -multiply	Pattern-matching (j=4)	참고 문헌 [5]의 방법
latency(clock)	63	24	2(18)
Flip-Flop	5	134	6
multiplier	1	0	3
adder	0	3	9
comparator	0	22	0
MUX	0	22	10
CRL	0	18	0

이진수를 LSB로 넣고, 상위 비트는 모두 0으로 설정함으로써 4비트 이진수로 바꾸는 역할을 한다.

이 VLSI 회로의 곱셈기로서, 표준 기저 (standard basis) 곱셈기, 정규 기저 (normal basis) 곱셈기[6~7] 또는 이중 기저 (dual basis) 곱셈기[8~10] 등의 기존 $GF(2^m)$곱셈기를 사용하는 것이 가능 하다.

$m=4$인 경우의 latency와 하드웨어 비용이 표 1에 정리되어 있다. 이 표에서, 이중 기저 곱셈기를 사용한 경우를 다루었고, 표준 기저와 정규 기저 곱셈기의 경우는 괄호 안에 표시하였다.

표 1에서 본 방식을 기존의 방식과 비교하기 위해 대표적인 두 개의 기존 방식과 비교하였다. 하나는 serial 시스템의 한 예인 square-and-multiply 방법[1~2]이고, 다른 하나는 systolic 시스템의 한 예인 pattern-matching 방법[3~4]이다.

III. 제안하는 비트별 병렬 처리 알고리즘 및 구현

이번 절에서는 참고 문헌 [5]에서 제안한 방법을 개량하여 성능을 향상하는 방법을 제안하고자 한다. 그 기본 아이디어는 그림 2에 주어진 DPTk 회로를 단순화 하는 것이다. 표 1을 살펴보면, 순환 회전 함수를 18개나 사용하고 있는 것을 볼 수 있다. 그런데 문제는 순환 회전 함수의 게이트 수 (gate count)가 2(m-1)+6 이며 (NAND 게이트 수를 기준), 이러한 순환 회전 함수가 $\sum_{i=1}^{m}\{(m-2)2^i\} \bmod (2^m-1)$개가 필요하다는 것이다. 따라서 m값이 큰 경우 게이트 수가 $m^2 2^m$에 비례하여 급속히 증가하게 되므로 성능이 떨어지게 된다. 이러한 특성 때문에 참고 문헌 [5]의 시스템이 m이 작을 때 hardware cost가 작고 latency가 작아 효과적이었던 것에 비해, m이 큰 경우에 hardware cost가 매우 커지게 되어 성능이 떨어지게 된다.

이를 보완하기 위해서 본 논문에서는 순환 회전 함수를 사용하지 않고, 대신 시스템 상수를 적극 활용하는 방법을 제안한다. 이를 위해 식 (4)와 식(5)를 각각 다음과 같이 고쳐 쓸 것을 제안한다.

$$pt_{m-i} = A^{2^{m-i}} \quad (11)$$
$$= \sum_{k=1}^{m/2} \left(a_{m-2k+1}\alpha^{(m-2k+1)2^{m-i}} + a_{m-2k}\alpha^{(m-2k)2^{m-i}}\right)$$

$$\left(a_{m-2k+1}\alpha^{(m-2k+1)2^{m-i}}+a_{m-2k}\alpha^{(m-2k)2^{m-i}}\right) \quad (12)$$

$$=\begin{cases}\alpha^{(m-2k+1)2^{m-i}}+\alpha^{(m-2k)2^{m-i}} & if \quad (a_{m-2k+1},a_{m-2k})=(1,1)\\ \alpha^{(m-2k+1)2^{m-i}} & if \quad (a_{m-2k+1},a_{m-2k})=(1,0)\\ \alpha^{(m-2k)2^{m-i}} & if \quad (a_{m-2k+1},a_{m-2k})=(0,1)\\ 0 & if \quad (a_{m-2k+1},a_{m-2k})=(0,0)\end{cases}$$

식 (11)과 식 (12)에 따라 그림 2의 DPTk 회로는 그림 4와 같이 바뀌게 된다. 여기서, $\alpha^{(m-2k+1)2^{m-i}}$와 $\alpha^{(m-2k+1)2^{m-i}}$는 주어진 $(m-i)$에 대해 상수이다. 이러한 상수 값들을 본 논문에서는 시스템 상수라고 부른다.

그림 4에서 제안한 DPTk 회로를 적용하면, 표 1의 값들이 표 2와 같이 바뀌게 된다.

표 2를 살펴보면, 순환 회전 회로를 사용하지 않고 있음을 알 수 있다.

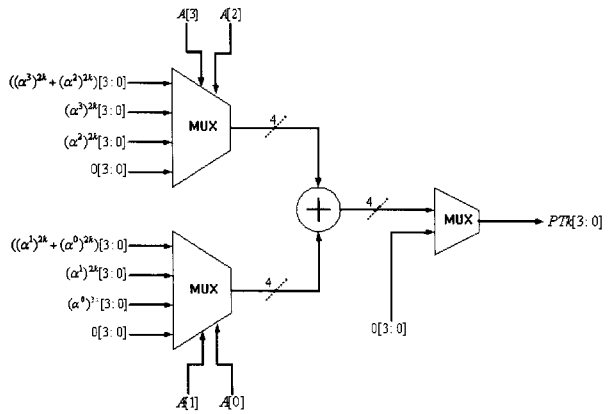


그림 4. k=1, 2, 3인 경우의 제안하는 DPTk 회로
Fig. 4. Proposed DPTk circuit for k=1, 2, 3.

표 2. $m=4$인 경우의 지수 제곱 연산을 위한 제안하는 방법, square-and-multiply 방법, pattern-matching 방법의 latency와 하드웨어 기본 요소 비교
Table 2. Comparison of latency and primitive hardware elements for the propsosed, the square-and-multiply, and the pattern-matching systems for exponentiation in $GF(2^m)$ when $m=4$.

Primitive Element	Square-and-multiply	Pattern-matching (j=4)	참고 문헌 [5]의 방법
latency(clock)	63	24	2(18)
Flip-Flop	5	134	6
multiplier	1	0	3
adder	0	3	9
comparator	0	22	0
MUX	0	22	10
CRL	0	18	0

IV. 결과 및 결론

본 절에서는 앞 절에서 제안한 VLSI 시스템을 기존의 방법과 비교해 본다. 특히 m 값이 증가함에 따른 변화에 주목한다. m 값의 변화에 따른 latency와 게이트 수의 변화는 각각 그림 5와 그림 6에 주어져 있다.

그림 5와 그림 6의 결과를 살펴보면, 기존의 두 방법인 square-and-multiply 방법과 pattern-matching 방법

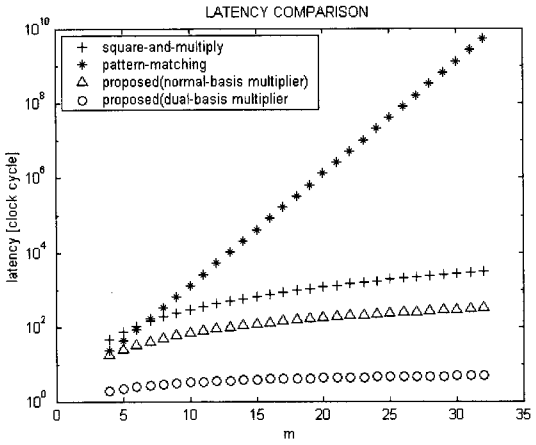


그림 5. 다양한 m값에 대한 제안하는 시스템, square-and-multiply 시스템, pattern-matching 시스템의 latency 비교 도표
Fig. 5. Plot of latency comparison for the proposed, the square-and-multiply, and the pattern-matching systems with respect to various m.

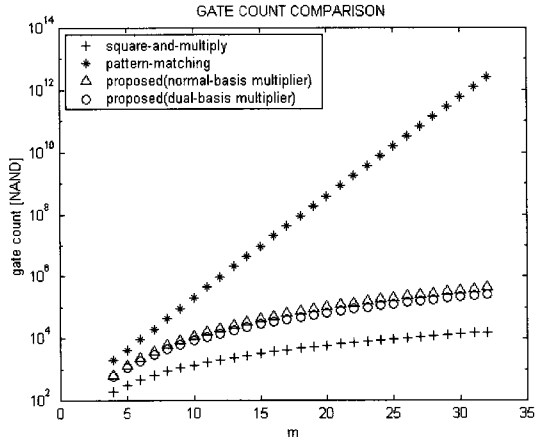


그림 6. 다양한 m값에 대한 제안하는 시스템, square-and-multiply 시스템, pattern-matching 시스템의 게이트 수 비교 도표
Fig. 6. Plot of gate count comparison for the proposed, the square-and-multiply, and the pattern-matching systems with respect to various m.

에 비해 제안하는 방법이 낮은 latency와 하드웨어 코스트를 가지며, 이 값들은 m이 증가해도 급속히 커지지 않는 것을 볼 수 있다. Pattern-matching 방법의 경우 m이 증가하면 하드웨어 코스가 급격히 증가하고, square-and-multiply 방법의 경우 latency가 급격히 증가하는 것을 볼 수 있다.

일반적으로 큰 하드웨어 코스트는 area의 증가를 가져와서, 실제적인 구현 자체가 어려워 질 수 있다. 따라서, pattern-matching 방법은 그 응용 범위가 m이 작은 경우로 제한된다.

그리고 높은 latency는 산출율의 저하를 가져와서, 실용적인 활용의 효율을 떨어뜨리게 된다. 따라서, square-and-multiply 방법을 m이 큰 경우에 적용할 경우 낮은 산출율로 인해 그 효율이 떨어지게 된다.

이에 대조적으로, 제안하는 방법의 경우는 하드웨어 코스트와 레이턴시가 m이 증가해도 급격히 커지지 않는다. 따라서, 본 방법은 m이 큰 경우에도 구현하기 용이하고, 높은 산출율을 유지하여 그 사용 효율이 상대적으로 높다고 결론지을 수 있다.

참 고 문 헌

[1] C. C. Wang and D. Pei, "A VLSI design for computing exponentiations in and its application to generate pseudorandom number sequences", IEEE Trans. Comput. vol. 39, pp. 258-262, 1990.

[2] M. A. G. Martinez, G. M. Luna, F. R. Henriquez, "Hardware implementation of the binary method for exponentiation in $GF(2^m)$", Proc. Fourth International Conferenceon Computer Science, pp. 131-134, 2003.

[3] K. Slind, "A verification of AES", http://www.cs.utah.edu.~slindpapersaes.html, 2003.

[4] M. Kovac, N. Rangnathan, "ACE: A VLSI chip for Galois field $GF(2^m)$", IEEE T. Circuits and Systems, vol. 43, pp. 289-297, 1996.

[5] 한영모, "GF(2^m) 상에서의 효율적인 지수제곱 연산을 위한 VLSI Architecture 설계", 전자공학회 논문지 제 41권 SC편 제 6호, pp. 27-35, 11월 2004년.

[6] B. Sunar, C. K. Koc, "An efficient optimal normal basis type II multiplier", IEEE T. Computers, vol. 50, pp. 83-87, 2001.

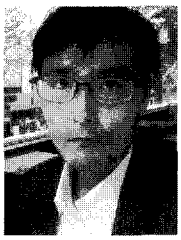
[7] C. K. Koc, B. Sunar, "Low complexity bit-parallel canonical and normal basis multipliers for a class of finite fields", IEEE T. Computers, vol. 47, pp. 353-356, 1998.

[8] F. R. Henriquez, C. K. Koc, "Parallel multipliers based on special irreducible pentanomials", IEEE T. Computers 52 (2003) 1-8.

[9] H. Fan, M. A. Hasan, "A new approach to sub-quadratic space complexity parallel multipliers for extended binary fields", IEEE T. Computers, vol. 56, pp. 224-233, 2007.

[10] S. T. J. Fenn, M. Benaissa, D. Taylor, "$GF(2^m)$ multiplication and division over the dual basis", IEEE T. Computers, vol. 45, pp. 319-327, 1996.

저 자 소 개



한 영 모(정회원)

1992년 서울대학교 물리교육학과 학사 졸업(준우등 졸업)

1995년 서울대학교 제어계측공학과 학사 졸업(차석 졸업)

1998년 서울대학교 대학원 전기공학부 석사 졸업(컴퓨터 기반 신호처리용 VLSI 설계 전공)

2002년 서울대학교 대학원 기계항공공학부 박사 졸업(컴퓨터비전 기반 인체 영상의 자세 인식, 로봇 제어 전공)

2002년~2003년 세종-록히드마틴 우주항공 연구소 전임연구원, 전임연구교수

2004년~2005년 이화여자대학교 정보통신공학과 연구전임강사, 연구교수

2006년~현재 한양사이버대학교 컴퓨터공학과 전임강사, 조교수

2007년~현재 International Biographical Centre (England), Deputy Director General 및 Honorary Director General

2007년~현재 World Congress of Arts, Science and Communications (England), Vice President

2009년~현재 American Biographical Institute Research Association (USA), Deputy Governor

2010년~현재 World Academy of Letters (USA), Vice Chancellor

<주관심분야 : 컴퓨터비전 응용 멀티미디어 및 생체 영상 인식, 모바일 및 로봇 임베디드 시스템, 인간과 컴퓨터의 시각적 인터페이스, 정보기술을 위한 통합 과학적 접근법 등>