GF(2^m)상에서 2-디지트 시리얼 시스톨릭 곱셈기 설계 및 분석

김기원○ 이건직 유기영
경 북 대 학 교 컴 퓨 터 공 학 과
{kwkim}@purple.knu.ac.kr

Design and Analysis of a 2-digit-serial systolic multiplier for GF(2^m)

Kee-Won Kim○ Keon-Jik Lee Kee-Young Yoo
Dept. of Computer Engineering, Kyungpook University

요 약

본 논문에서는 유한 필드 $GF(2^m)$상에서 모듈러 곱셈 $A(x)B(x) \bmod P(x)$를 수행하는 2-디지트 시리얼(2-digit-serial) 시스톨릭 어레이 구조인 곱셈기를 제안하였다. LSB-first 곱셈 알고리즘을 분석한 후 2-디지트 시리얼 형태의 자료의존 그래프(data dependency graph, 이하 DG)를 생성하여 시스톨릭 어레이를 설계하였다. 제안한 구조는 정규적이고 서로 반대 방향으로 진행하는 에지들이 없다. 그래서 VLSI 구현에 적합하다. 제안한 2-디지트 시리얼 곱셈기는 비트-패러럴(bit-parallel) 곱셈기 보다는 적은 하드웨어를 사용하며 비트-시리얼(bit-serial) 곱셈기 보다는 빠르다. 본 논문에서 제안한 2-디지트 시리얼 시스톨릭 곱셈기는 기존의 같은 종류의 곱셈기 보다 처리기의 최대 지연 시간이 적다. 그러므로 전체 시스톨릭 곱셈기의 처리시간을 향상시킬 수 있다.

1. 서 론

유한 필드의 연산들은 암호학 등에서 매우 중요한 역할을 하고 있다. 특히, 최근에 많이 연구되고 있는 타원 곡선(elliptic curve) 암호시스템과 같은 암호학적 응용에 유한 필드의 연산이 중요하게 인식되었다[1].

$GF(2^m)$상의 연산중에, 덧셈은 비트별 배타적 논리합(exclusive or)이므로 비교적 빠르며, 비용이 작게드는 연산이다. 유한 필드상의 다른 연산들 중에 곱셈은 게이트 수나 지연시간에서 비용이 매우 많이 드는 연산중에 하나이다. 유한 필드상의 중요한 연산이 곱셈이다. 필드상의 곱셈 연산에 대한 효율적인 하드웨어 구현에 대해 많은 연구들이 이루어져 왔다[2,3,4,5].

유한 필드의 원소를 표준 다항식 기저 표현을 사용할 경우 곱셈 알고리즘은 LSB-first(least significant bit first)와 MSB-first (most significant bit first) 두 종류로 나눌 수 있다[3]. LSB-first 알고리즘은 두 번째 피연산자 $B(x)$의 LSB를 먼저 처리하고, MSB-first는 MSB를 먼저 처리한다. 일반적으로, 두 알고리즘의 공간 복잡도는 같으나, LSB-first 알고리즘을 사용한 경우가 MSB-first를 사용한 경우보다 더 작은 계산 지연 시간을 갖는다. 본 논문에서는 LSB-first 곱셈 알고리즘을 이용한다.

유한 필드의 곱셈기는 비트-패러럴 또는 비트-시리얼 구조를 가진다. 일반적으로, 전자는 후자에 비해 처리성능이 우수한 반면 공간 복잡도가 높다는 단점을 가지고 있다. 이런 두 구조는 공간-시간 상충관계(trade-off)의 전형적인 예이다. 공간과 시간 사이의 상충관계를 개선하기 위해서 디지트 시리얼 구조는 적절한 선택이다[5]. 본 논문에서는 이러한 디지트 시리얼 구조로 유한 필드상의 곱셈기를 설계한다.

디지트 시리얼 구조는 전체 데이터 비트를 각각 몇 비트들의 디지트들로 나눈다. 데이터들은 디지트 단위로 처리되고 전송된다. 만약 데이터의 크기가 m 비트이고, 디지트 크기가 L 비트이면, 디지트 수 $N=\lceil m/L \rceil$ 이다. 디지트 시리얼 구조는 N 시간 스텝마다 하나의 결과를 출력한다. 그러나, 비트-시리얼과 비트-패러럴 구조는 각각 m과 1시간스텝마다 하나의 결과를 출력한다. Guo[5] 등은 유한 필드상의 디지트 시리얼 시스톨릭 곱셈기를 제안하였다. 이 곱셈기는 MSB-first 알고리즘을 사용하였다.

본 논문에서는 유한 필드 $GF(2^m)$상에서 표준 다항식 기저 표현으로 곱셈 연산을 LSB-first 알고리즘을 사용하여 보다 빠른 2-디지트 시리얼 시스톨릭 곱셈기를 제안한다. 먼저 유한 필드 $GF(2^m)$상에서 LSB-first 곱셈 알고리즘을 분석한 뒤 이를 자료의존 그래프(data dependancy graph, 이하 DG)로 나타낸다. 이 DG를 2-디지트 시리얼로 만들기 위해 적절하게 DG를 수정한 뒤 이를 2-디지트 시리얼 시스톨릭 곱셈기로 설계한다. 제안한 어레이는 매우 정규적이며 규칙적이어서 VLSI 구현에 적절하다.

본 논문의 구성은 다음과 같다. 2장에서는 LSB-first 곱셈 알고리즘을 분석하여 DG를 생성한다. 3장에서는 이러한 DG를 2-디지트 시리얼로 만들기 위해 새로운 DG를 생성하여 2-디지트 시리얼 시스톨릭 곱셈기를 설계한다. 제안한 2-디지트 시리얼 시스톨릭 곱셈기를 4장에서는 기존의 곱셈기와 비교한다. 마지막으로, 5장에서는 결론을 내린다.

2. 곱셈 알고리즘

$A(x)$와 $B(x)$는 $GF(2^m)$의 원소이고, $G(x)$는 차수 m인 원시 다항식이다. 그리고, $P(x)$는 $A(x)B(x) \bmod G(x)$이면 다항식 $A(x)$, $B(x)$, $G(x)$ 및 $P(x)$는 다음과 같이 표현된다.

$$A(x) = \sum_{i=0}^{m-1} a_i x^i$$
$$B(x) = \sum_{i=0}^{m-1} b_i x^i$$
$$G(x) = x^m + \sum_{i=0}^{m-1} g_i x^i \quad (1)$$
$$P(x) = \sum_{i=0}^{m-1} p_i x^i$$

여기서 다항식들의 각 계수들은 $GF(2)$의 원소이다. 유한 필드상의 곱셈은 차수 m의 원시 다항식에 의해서 정의된다. 두 원소의 곱셈은 단순히 두 다항식을 곱한 뒤에 $G(x)$로 모듈러 연산을 취해주면 된다. 본 논문에서는 LSB-first 곱셈방법을 사용하여 2-디지트 시리얼 곱셈기를 설계한다.

2.1 LSB-first 곱셈 알고리즘

본 논문에서는 [3,4]에서 제안된 LBS-first 곱셈 알고리즘을 사용한다.

Input : $A(x)$, $B(x)$, $G(x)$
Output : $P(x) = A(x)B(x) \bmod G(x)$
1. $a_j^{(0)} = a_j$, for $0 \le j \le m-1$
2. $a_{-1}^{(i)} = 0$, for $0 \le i \le m$
3. $p_j^{(0)} = 0$, for $0 \le j \le m-1$
4. for i = 1 to m do
5. for $j = m-1$ to 0 do
6. $a_j^{(i)} = a_{j-1}^{(i-1)} + a_{m-1}^{(i-1)} g_j$
7. $p_j^{(i)} = a_j^{(i-1)} b_{i-1} + p_j^{(i-1)}$

그림 1. LSB-first 곱셈 알고리즘

그림 1의 알고리즘에서 $a_j^{(i)}$, $p_j^{(i)}$는 각각 $A^{(i)}$와 $P^{(i)}$의 j번째 계수를 나타내고, a_i, b_i는 각각 A와 B의 i번째 계수를 나타내고, g_j는 $G(x)$의 j번째 계수를 나타낸다.

2.2 자료의존 그래프

위 알고리즘의 수행을 2차원 평면에 그래프로 표현할 수 있다. 각 인덱스 점(index point) (i,j), $(1 \le i < m,\ 0 \le j < m-1)$은 계산이 수행되는 곳으로 계산점(computation point)이라고도 하는데, 그래프에서 한 개의 노드(node)로 표현되며, 각 인덱스 점에서 계산에 필요한 자료의 흐름은 그래프에서 에지(edge)로 표현된다. 에지는 각 계산에 필요한 자료의 의존관계를 의미하므로 이 그래프를 DG라 한다. 위의 알고리즘에 대한 DG는 그림 2와 같다. 시스톨릭 어레이의 설계에 대한 설명을 간단히 하기 위해서 $m=6$인 경우를 보겠다. 일반적인 $GF(2^m)$상의 시스톨릭 어레이의 설계는 같은 방법으로 확장하여 적용하면 된다.

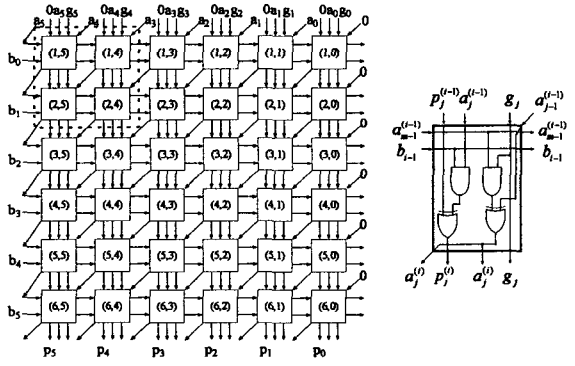

그림 2. LSB-first곱셈에 대한 DG와 (i, j) 계산점의 회로$(m=6)$

그림 2는 그림 1에 대한 $GF(2^m)$상의 LSB-first 곱셈의 DG이다. 여기서 $m=6$ 이다. 이것은 $m \times m$개의 계산점들로 구성된다. 이 DG에서 i 번째 행(row)에 있는 계산점들은 그림 1의 알고리즘에서 i 번째를 수행한다. (i,j)위치의 계산점은 그림 1에 있는 알고리즘의 6과 7단계의 $a_j^{(i)}$, $p_j^{(i)}$값을 계산한다. 결과인 $P^{(m)}(x)$의 계수들은 알고리즘에서 루프가 m번 반복 후 DG의 아래쪽 행에서 출력된다.

3. 2-디지트 시리얼 시스톨릭 곱셈기

여기서 디지트 크기를 2로 고정하고, 디지트 수 $N = m/2$을 정수라고 가정한다. 그러면 A_k, B_k, G_k 및 P_k들은$(0 \le k \le N-1)$ 각각 다항식 $A(x)$, $B(x)$, $G(x)$ 및 $P(x)$의 계수들의 디지트들이다. 예를 들면, $A(x)$는 다음과 같이 정의한다.

$$A(x) = (a_{m-1}, a_{m-2}, \ldots, a_1, a_0) = (A_{N-1}, A_{N-2}, \ldots, A_1, A_0) \quad (1)$$

여기서, 디지트 A_k는 $A_k = (a_{2k+1}, a_{2k})$이다. 그리고, 나머지 B_k, G_k 및 P_k도 같은 방법으로 정의된다.

3.1 DG의 새로운 변형

그림 2의 DG를 2-디지트 시리얼로 만들기 위해서는 2×2로 계산점들을 묶어서 하나의 계산점으로 만든다. 예를 들면, 그림 1의 점선과 같이 2×2로 계산점들을 묶어서 하나의 계산점으로 만들면, 디지트 크기는 2 이고 디지트 수는 $N=m/2=3$ 이다. 결과의 DG와 계산점은 그림 3과 같다. 이렇게 변형된 DG는 그림 3에 있는 계산점들의 $N \times N$으로 구성된다. 이러한 DG에서 디지트 A_k, G_k는 $(1,k)$ 계산점에 입력되고, 디지트 B_k는 $(k+1,N-1)$ 계산점에 입력된다. 그리고, 결과인 디지트 P_k는 (N,k) 계산점으로부터 출력된다. (단, $0 \le k \le N-1$).

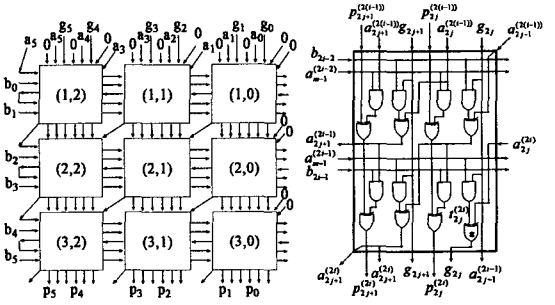

그림 3. 그림 2의 변형

그림 3의 DG에는 $(i, j-1)$ 계산점에서 (i, j) 계산점으로의 역방향 자료 흐름이 있다. 그래서, 수평으로 양방향의 자료 흐름이 생기게 된다. 이러한 사실 때문에 이 DG는 오른쪽으로 투영할 수 없다.

그림 3에서, (i,j) 계산점은 $(i,j-1)$ 계산점에서 받은 $a_{2j-1}^{(2i-1)}$값과 $t_{2j}^{(2i)}$를 '*' 표시가 있는 XOR 게이트를 통해 연산한다. 이러한 연산은 $(i+1,j)$ 계산점으로 옮길 수가 있다(단, $1 \le i \le N-1$, $0 \le j \le N$). 그러면 (i,j) 계산점에 있는 '*' 표시가 있는 XOR 게이트를 $(i+1,j)$ 계산점으로 옮기면 $t_{2j}^{(2i)}$는 (i,j) 계산점의 출력이 되며, $a_{2j-1}^{(2i-1)}$는 $(i+1,j)$ 계산점의 입력이 된다. 이런 변형과정을 거치면 그림 4와 같은 DG를 얻을 수 있다.

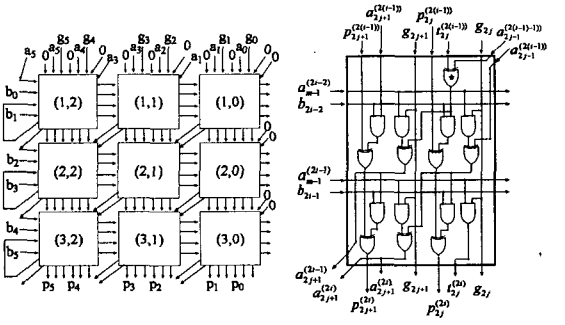

그림 4. 2-디지트 시리얼 곱셈 DG와 (i, j) 계산점의 회로$(m=6)$

그림 4에 있는 DG의 첫 행의 계산점들의 위에서 입력되는 $t_{2j}^{(2(i-1))}$은 a_{2j+1}값을 입력한다. 이것은 대각에서 입력되는 값 $a_{2j-1}^{(2(i-1)-1)}$에 0값을 입력함으로써 첫 행의 계산점들은 정상적으로 계산을 시작할 수 있다. 이 DG는 그림 4에 있는 계산점들의 3×3으로 구성된다. 그림 4의 계산점은 8개의 2-input AND 게이트들과 8개의 2-input XOR 게이트들로 이루어진다. 이 DG는 수평으로 양방향의 자료 흐름이 없어서 오른쪽으로 투영을 시킬 수 있다.

3.2 $GF(2^m)$상에서의 새로운 디지트 시리얼 시스톨릭 곱셈기

이 절에서는 그림 4의 DG로부터 디지트 시리얼 시스톨릭 곱셈기를 설계한다. 그림 4의 DG를 [6]에 있는 투영절차에 따라 오른쪽으로 투영시킨다. 그리고, 컷-셋 시스톨릭화 테크닉(cut-set systolisation techniques)[7]를 적용하면 그림 5와 같은 시스톨릭 곱셈기를 얻을 수 있다.(단, '•'는 1-비트 래치). 이 시스톨릭 곱셈기는 그림 6에 있는 처리기 $m/2$개로 구성된다.

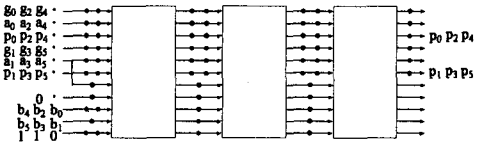

그림 5. $GF(2^6)$상에서의 2-디지트 시리얼 시스톨릭 곱셈기

DG를 투영함에 따라, 이 어레이에 디지트들 A_k, G_k, B_k 그리고 P_k는 왼쪽으로 시리얼 형태로 들어간다. 결과 $P(x)$의 계수들은 2 비트씩 오른쪽에서 출력된다. 입력 디지트들 A_k, G_k, 그리고 P_k들은 MSD(most significant digit)부터 입력되고, B_k는 MSD(most significant digit)부터 입력된다. 결과 digit P_k들은 MSD부터 시리얼 형태로 어레이의 오른쪽에서 출력된다.

각 처리기의 제어를 위해서 제어 신호가 필요하다. 길이 $(m/2)-1$인 일련의 제어신호 ctrl 011...1가 처리기를 제어한다. 그림 4의 DG에서, i번째 행에 있는 모든 계산점들은 값 $a_{m-1}^{(2i-2)}$, $a_{m-1}^{(2i-1)}$과 b_{2i-2}, b_{2i-1}이 필요하다. 그래서, 이 값들은 버퍼에 저장할 필요가 있다. 시스톨릭 곱셈기의 각 처리기에 $2L$개의 멀티플렉서들과 $2L$개의 1-비트 래치들을 사용하여 각 값들을 저장한다. 제어 신호가 0값일 때 이 값들은 저장된다.

이 시스톨릭 어레이의 자료가 연속적으로 들어오면 $3m/2$ 시간 스텝의 초기 지연 후에 $m/2$ 시간 스텝마다 하나의 결과를 얻을 수 있다.

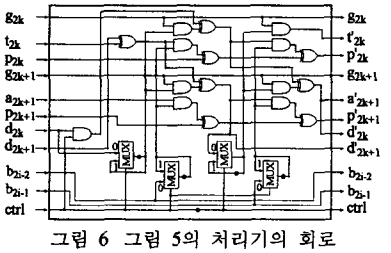

그림 6 그림 5의 처리기의 회로

표 1. $GF(2^m)$상의 digit-serial 시스톨릭 곱셈기 성능 비교

	Guo등[5]	제안한 곱셈기
전체 곱셈기의 복잡도	N 처리기 1 2-입력 XOR 1 2-입력 AND	N 처리기
처리기 복잡도	10 2-입력 AND 1 2-입력 XOR 2 3-입력 XOR 1 4-입력 XOR 20 1-비트 래치 4 MUX	9 2-입력 AND 8 2-입력 XOR 22 1-비트 래치 4 MUX
처리기 최대 지연 시간	$2T_{AND2}+T_{MUX2}$ $+T_{XOR3}+T_{XOR4}$	$T_{AND2}+T_{MUX2}$ $+3T_{XOR2}$

4. 비교분석

이 장에서는 제안한 시스톨릭 곱셈기와 기존의 곱셈기를 비교한다. Guo[5]는 MSB-first 곱셈 알고리즘을 이용하여 디지트 시리얼 시스톨릭 곱셈기를 설계하였다. 본 논문에서는 LSB-first 곱셈 알고리즘을 이용하여 디지트 시리얼 시스톨릭 곱셈기를 설계하였다. 두 곱셈기의 성능비교는 표 1과 같다(단, T_{GATE}는 GATE의 지연시간). 비교를 위해서 3-입력 XOR 게이트는 2개의 2-입력 XOR 게이트로 구성되고, 4-입력 XOR 게이트는 3개의 2-입력 XOR 게이트로 구성된다는 것을 가정한다[8]. 그러면 처리기 복잡도에서 XOR 게이트 수는 같고, MUX 수도 같다. 본 논문에서 제안한 디지트 시리얼 곱셈기가 Guo의 곱셈기에 비해 하나의 2-입력 AND 게이트가 적고, 2개의 1-비트 래치가 많다. 그러나 계산점의 최대 지연시간의 차이는 $T_{AND2}+T_{XOR3}+T_{XOR4}-3T_{XOR2}$이다. 그래서 제안한 곱셈기의 처리기가 Guo의 곱셈기의 처리기보다 $T_{AND2}+T_{XOR3}+T_{XOR4}-3T_{XOR2}$만큼 빨리 수행될 수 있다.

5. 결론

본 논문에서는 GF(2^m)상에서 LSB-first 곱셈 알고리즘을 이용하여 2-디지트 시리얼 시스톨릭 곱셈기를 제안하였다. LSB-first 곱셈 알고리즘으로부터 얻어진 DG를 2-디지트 시리얼로 만들기 위해 각 계산점들을 2×2로 합하여 DG를 만들었다. 이 DG는 양방향의 자료 흐름 때문에 오른쪽으로 투영이 불가능하였다. 이것을 가능하도록 하기 위해서 각 계산점을 수정하여 새로운 DG를 만들었다. 이러한 DG는 자료의 의존관계가 규칙적이고, 또한 서로 반대 방향으로 진행하는 에지들이 없어서 시스톨릭 어레이 구조의 하드웨어로 설계하기 좋다. 그리고 비트-시리얼보다는 빠르며 비트-패러럴보다는 적은 하드웨어를 사용함으로써, 시간과 공간 사이의 상충관계를 개선하기 위해서 디지트 시리얼 구조는 적절하다.

기존의 방식보다 하드웨어적인 측면에서 볼 때 게이트의 수는 약간 증가하지만, 각 처리기의 최대 지연 시간을 줄임으로써 전체적인 처리 시간을 줄일 수 있었다

참 고 문 헌

[1] A.J. Menezes, *Applications of Finite Fields*, Kluwer Academic Publishers, 1993.

[2] L. Song and K. K. Parhi, "Efficient finite field serial/parallel multiplication," *in Proc. Int. Conf. Application Specific Syst., Architectures and Processors*, Chicago, IL, Aug, 1996, pp. 72-82

[3] S. K. Jain, L. Song, and K. K., Parhi, "Efficient Semi-Systolic Architectures for Finite Field Arithmetic," *IEEE Trans. on VLSI Systems*, vol. 6, no. 1, pp. 101-113, March, 1998.

[4] 유기영, 김정준, "유한 필드 GF(2^n)상의 시스톨릭 곱셈기 및 곱셈/제곱기", 제 11회 정보보호와 암호에 관한 학술대회, WISC'99, pp. 375 - 389, 1999.

[5] J. H. Guo and C. L. Wang, "Digit-serial systolic multiplier for finite fields GF(2^n)," *IEE Proc.-Comput. Digit. Tech.*, Vol. 145, No. 2, pp. 143-148, March, 1998.

[6] S. Y. Kung, *VLSI Array Processors*, Englewood Cliffs, NJ: Prentice-Hall, 1988.

[7] S. Y. Kung, "On supercomputing with systolic/wavefront array processors," *Proc. IEEE*, pp. 867-884, 1984.

[8] N. Weste and K. Eshraghian, *Principles of CMOS VLSI design: a system perspective*, Addison Wesley, Reading, MA, 1985.