GF(2^m)상에서 병렬 승산기에 대한 기약다항식의 새로운 구성

문 경 재, , *황 종 학, **박 승 용, 김 흥 수
인하대학교 전자공학과, *체육과학 연구원, **제능대학
전화 : 032-860-7413 / 핸드폰 : 016-655-5567

A New Construction of the Irreducible Polynomial for parallel multiplier over GF(2^m)

Kyung Jae Moon, *Jong Hak Hwang, **Seung Young Park, Heung Soo Kim
Dept. of Electronic Engineering, Inha University
*Dept. of system Engineering, Korea Sport Science Institute, **Dept. of Jaenung University
E-mail : satanmoon@hanmail.net

Abstract

This paper presents the construction algorithm of the irreducible polynomial which needs to multiply over GF(2^m) and the flow chart representing the proposed algorithm has been proposed. And also, we get the degree from the value of xm+k formation to the value of k = 7 using the proposed flow chart. The multiplier circuit has been implemented by using the proposed irreducible polynomial generation(IPG) algorithm in this paper, and we compared the proposed circuit with the conventional one. In the case of k = 7, one AND gate and five Ex-or gates are needed as the delay time for the irreducible polynomial in the proposed algorithm, but seven AND gates and seven Ex-or gates in the conventional one. As a result, the proposed algorithm shows the improved performance on the delay time.

I. 서론

유한체(Galois field)는 오류정정부호, 디지털 통신의 암호화 및 해독화를 요하는 보안 등에 많이 응용되고 있다. 이들 중 오류정정부호의 경우 유한체 $GF(2^m)$ 상의 연산에서 실제로 부호기 및 복호기 설계시 전체 시스템의 규모와 성능에 절대적인 영향을 미치므로 회로경로의 연결, 시스템 구조의 복잡성과 동시성 등의 문제점을 개선하기 위한 연구가 진행되어 왔다.[1] 유한체 연산은 가산, 승산, 멱산, 제산 등인데, 가산은 매우 간단하여 유한체의 원소(field elements)들이 다항식 형태로 표현되는 경우 매우 간단한 회로로 수행될 수 있다.

1984년 Yet등[2]은 유한체 $GF(2^m)$상에서 $AB+C$ 연산을 수행하는 병렬 입-출력 시스토릭(systolic)구조를 갖는 승산기를 개발하였으며, 1차원 승산기 및 2차원 승산기의 셀 마다 기약다항식을 처리하기 위하여 추가적으로 1개의 AND와 1개의 XOR게이트를 사용하여 그 만큼 시간 지연을 갖는다. Scott 등[3]이 제시한 승산기는 직렬 입-출력 형태의 비트슬라이스(bit-slice) 구조로 되어 있으며, 각 셀에서 기약다항식을 처리하기 위하여 1개의 AND와 1개의 XOR에 해당하는 소자를 사용하고 있기 때문에 그 만큼 시간 지연을 갖는다. 그 외에 Mastrovito[3,4]에 의해서 기약 3항식 x^m+x+1에 대한 승산 알고리즘이 제안되었으나 기약다항식이 3항식으로 제안되었어 기약다항식을 선택하는 범위가 적어진다. 또한 Sunar[5]에 의하여 기약 3항식

x^m+x^n+1을 이용한 승산기를 제안하였으나 n을 정하여 행렬식으로 수식을 전개하는 과정을 필요로 하고 기약다항식이 3항식으로 제안되어야 한다.

본 논문에서는 유한체 $GF(2^m)$상에서 모든 기약 다항식에 적용이 가능한 알고리즘을 제안하였다. 제시된 기약다항식 생성 알고리즘의 회로의 복잡도는 증가하나 연산 지연시간이 적게 걸리는 장점이 있다. 이는 집적회로의 기술 발달에 따라 결과를 얻기 위한 실행 시간이 더욱 중요하기 때문이다. x^{m+k}에서 $k=7$인 값을 구하기 위하여 필요한 시간 지연은 한 개의 AND 게이트 시간 지연과 5개의 XOR 게이트 시간 지연이 필요하다.

II. 기약다항식 생성 알고리즘

$GF(2^m)$상에서 두 다항식을 승산하기 위하여 원시 기약다항식 $f(x)$이 필요하며 승산다항식 $A(x)$와 피 승산다항식 $B(x)$을 식 (1)과 같이 표현된다.

$$f(x)=x^m+\sum_{i=0}^{m-1}f_ix^i$$
$$A(x)=\sum_{i=0}^{m-1}a_ix^i \qquad (1)$$
$$B(x)=\sum_{i=0}^{m-1}b_ix^i$$

식 (1)에서 두 다항식 $A(x), B(x)$을 승산하면 최고차 항이 $2m-2$이 된다. 따라서 m차 이상의 차수 항을 원시기약다항식으로 처리하여 연산할 수 있다. 이를 처리하기 위하여 $f(\alpha)=0$을 만족하는 α^m은 다음 식 (2)과 같이 쓸 수 있다.

$$\alpha^m=\sum_{i=0}^{m-1}f_i\alpha^i \qquad (2)$$

식 (2)을 식 (3)으로 다시 표현할 수 있다.

$$\alpha^m=\sum_{i=1}^{m}f_{m-i}\alpha^{m-i} \qquad (3)$$

식 (3)을 이용하여 α^{m+1}을 구하면 식 (4)으로 구할 수 있다.

$$\alpha^{m+1}=\alpha^m\alpha=\sum_{i=1}^{m}f_{m-i}\alpha^{(m-i)+1}$$
$$=f_{m-1}\alpha^m+\sum_{i=2}^{m}f_{m-i}\alpha^{(m-i)+1} \qquad (4)$$

식 (4)에 식 (3)을 대입하면 식(5)과 같이 쓸 수 있다.

$$\alpha^{m+1}=f_{m-1}\sum_{i=1}^{m}f_{m-i}\alpha^{m-i}+\sum_{i=2}^{m}f_{m-i}\alpha^{(m-i)+1}$$
$$=\sum_{i=1}^{m}f_{m-1}f_{m-i}\alpha^{m-i}+\sum_{i=2}^{m}f_{m-i}\alpha^{(m-i)+1} \qquad (5)$$

식 (5)의 두 번째항에서 i가 1일 때를 첨가하여 첫 번째항과 연산할 수 있도록 한다. 이를 정리하여 식 (6)에 표현하였다.

$$\alpha^{m+1}=\sum_{i=1}^{m}f_{m-1}f_{m-i}\alpha^{m-i}+\sum_{i=2}^{m}f_{m-i}\alpha^{(m-i)+1}+f_{-1}\alpha^0$$
$$=\sum_{i=1}^{m}f_{m-1}f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m-1}f_{m-(i+1)}\alpha^{m-i}+f_{-1}\alpha^0 \qquad (6)$$

여기서 f_i에서 $i<0$ 이면 $f_i=0$ 이다.

식 (6)의 세 번째항을 두 번째항에 포함할 수 있도록 항의 형태를 변형하여 표현하면 식(7)과 같다.

$$f_{m-(m+1)}\alpha^{m-m} \qquad (7)$$

식 (7)을 식 (6)에 대입하여 정리하면 식 (8)과 같다.

$$\alpha^{m+1}=\sum_{i=1}^{m}f_{m-1}f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m-1}f_{m-(i+1)}\alpha^{m-i}+f_{m-(m+1)}\alpha^{m-m}$$
$$=\sum_{i=1}^{m}f_{m-1}f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m}f_{m-(i+1)}\alpha^{m-i}$$
$$=\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i} \qquad (8)$$

식 (8)을 이용하여 α^{m+2}을 구하면 식 (9)으로 구할 수 있다.

$$\alpha^{m+2}=\alpha^{m+1}\alpha$$
$$=\alpha\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i}$$
$$=\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i+1} \qquad (9)$$

식 (9)에서 $i=1$일 때 m차 항이 발생하므로 따로 분리하여 식 (10)으로 표현한다.

$$\alpha^{m+2}=(f_{m-1}f_{m-1}+f_{m-2})\alpha^m+\sum_{i=2}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i+1}$$
$$=(f_{m-1}+f_{m-2})\alpha^m+\sum_{i=2}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i+1} \qquad (10)$$

식 (10)에서 식 (3)을 이용하여 α^m에 대입하여 정리하면 식 (11)과 같다.

$$\alpha^{m+2}=(f_{m-1}+f_{m-2})\sum_{i=1}^{m}f_{m-i}\alpha^{m-i}+\sum_{i=2}^{m}(f_{m-1}f_{m-i}+f_{m-(i+1)})\alpha^{m-i+1}$$
$$=\sum_{i=1}^{m}(f_{m-1}+f_{m-2})f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m-1}(f_{m-1}f_{m-(i+1)}+f_{m-(i+2)})\alpha^{m-i} \qquad (11)$$

N째항에 $i=m$ 을 추가하여 첫번째항의 형태의 수식을 갖도록 식 (12)에 표현하여 정리한다.

$$\alpha^{m+2}=\sum_{i=1}^{m}(f_{m-1}+f_{m-2})f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m-1}(f_{m-1}f_{m-(i+1)}+f_{m-(i+2)})\alpha^{m-i}+(f_{m-1}f_{-1}+f_{-2})\alpha$$

$$=\sum_{i=1}^{m}(f_{m-1}+f_{m-2})f_{m-i}\alpha^{m-i}+\sum_{i=1}^{m}(f_{m-1}f_{m-(i+1)}+f_{m-(i+2)})\alpha^{m-i}$$

$$=\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-2}f_{m-i}+f_{m-1}f_{m-(i+1)}+f_{m-(i+2)})\alpha^{m-i} \quad (12)$$

여기서 f_i에서 $i<0$이면 $f_i=0$ 이다.

앞에서 유도한 방법으로 α^{m+3}을 구하면 식(13)과 같다.

$$\alpha^{m+3}=\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-1}f_{m-(i+1)}+f_{m-1}f_{m-2}f_{m-i}+f_{m-1}f_{m-(i+2)}+f_{m-1}f_{m-2}f_{m-i}+f_{m-2}f_{m-(i+1)}+f_{m-3}f_{m-i}+f_{m-(i+3)})\alpha^{m-i} \quad (13)$$

$GF(2^m)$상에서 식 (13)에서 수식을 다시 정리하기 위하여 같은 항이 존재할 때 짝 수개 만큼 제거한다. 즉 식 (13)에서는 $f_{m-1}f_{m-2}f_{m-i}$가 2개 있으므로 2개 모두 제거하여 정리하면 식 (14)과 같다.

$$\alpha^{m+3}=\sum_{i=1}^{m}(f_{m-1}f_{m-i}+f_{m-1}f_{m-(i+1)}+f_{m-1}f_{m-(i+2)}+f_{m-2}f_{m-(i+1)}+f_{m-3}f_{m-i}+f_{m-(i+3)})\alpha^{m-i} \quad (14)$$

식 (1)부터 식 (14)을 정리하기 위하여 식 (15)과 같이 표현하고, 순서도는 그림 1과 같다.

$$\alpha^{m+k}=\sum_{i=1}^{m}(F_k)\alpha^{m-i} \quad (15)$$

여기서 $F_0=f_{m-i}$ 이다.

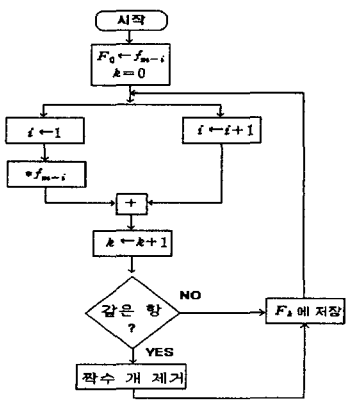


그림 1. 기약다항식 발생 알고리즘 순서도

III. 기약다항식 발생기 구성

본 장에서는 II장에서 제시한 기약다항식 발생 알고리즘을 이용하여 기약다항식 발생기를 구성할 것이다. 먼저 α^{m+1}의 다항식을 구하는 회로를 구현하기 위하여 그림 2와 그림 3에 표시한 1개의 2 입력 mod(2) 가산 게이트와 1개의 2입력 mod(2) 승산 게이트를 나타내었다.

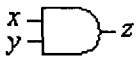


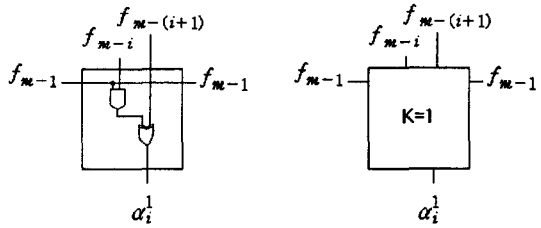
그림 2. mod(2) 승산 게이트

Fig. 2. The mod(2) multiplicative gate.

그림 3. mod(2) 가산게이트

Fig. 3. The mod(2) additional gate.

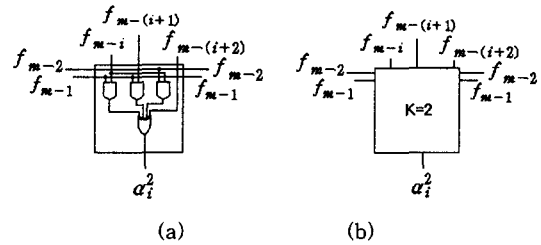
그림 2와 그림 3을 이용하여 식 (8)에서 $f_{m-1}f_{m-i}+f_{m-(i+1)}$ 의 회로와 기호를 그림 4 에 나타내었다.



(a) (b)

그림 4. $GF(2^m)$상에서 $k=1$인 기약다항식 발생회로 (a)회로도 (b)기호

그림 2, 3을 이용하여 식 (12)에서 $f_{m-1}f_{m-i}+f_{m-2}f_{m-i}+f_{m-1}f_{m-(i+1)}+f_{m-(i+2)}$의 회로도와 기호를 그림 5 에 나타내었다.



(a) (b)

그림 5. $GF(2^m)$상에서 $k=2$인 기약다항식 발생회로 (a)회로도 (b)기호

기존 기약다항식 발생기는 3항식을 이용하여 회로를

간략화하는 알고리즘을 사용하였으나 본 논문에서는 항수에 무관하게 성립하는 알고리즘이다. 따라서 기존에 항수에 무관하여 구하는 회로와 비교하기 위하여 그림 6,7에 기존 회로를 나타내었다.

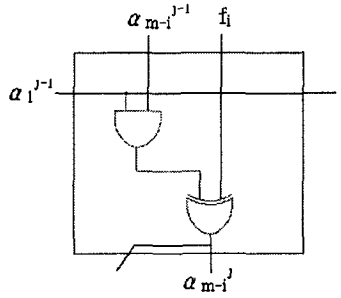


그림6. 기존 $GF(2^m)$상의 기약다항식 생성 기본셀

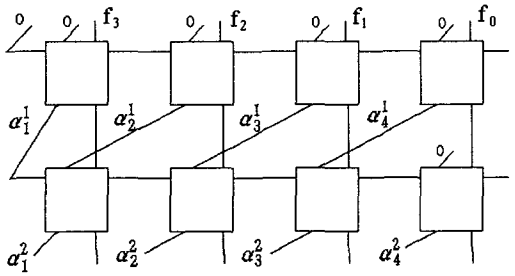


그림7. 기존 $GF(2^m)$상의 원시 기약다항식 생성 회로도

그림 6,7에서 α^6을 구하기 위하여 α^5에서 α^3의 계수가 1인지 아니지를 판단하여 α^6을 구할 수 있다. 따라서 α의 차수가 커지면 본 논문에서 제시한 알고리즘 수행 시간보다 결과를 얻기 위하여 많은 시간을 요한다.

IV. 비교 및 검토

본 장에서는 제시한 기약다항식 발생기를 기존기약다항식 발생기와 수행 결과 시간을 비교하여 표 1에 나타내었다.

표 1. 수행시간 비교표

k	기존 논문	본 논문
1	$1T_{AND}+1T_{XOR}$	$1T_{AND}+1T_{XOR}$
2	$2T_{AND}+2T_{XOR}$	$1T_{AND}+2T_{XOR}$
3	$3T_{AND}+3T_{XOR}$	$1T_{AND}+3T_{XOR}$
4	$4T_{AND}+4T_{XOR}$	$1T_{AND}+4T_{XOR}$
5	$5T_{AND}+5T_{XOR}$	$1T_{AND}+4T_{XOR}$
6	$6T_{AND}+6T_{XOR}$	$1T_{AND}+5T_{XOR}$
7	$7T_{AND}+7T_{XOR}$	$1T_{AND}+5T_{XOR}$

단 T_{XOR}는 2입력 XOR게이트만 사용하였다.

T_{AND}는 AND게이트 수행시간, T_{XOR}는 XOR게이트 수행시간

표 1에서 기존 논문과 본 논문에서 제시한 비교표에서 보는 바와 같이 k=1일 때, 즉 α^{m+1}의 값을 구하기 위하여 1개의 AND와 1개의 XOR 게이트 통과하여 결과를 얻을 수 있다. k=2일 때, 즉 α^{m+2}일 때, 기존 논문에서는 2개의 AND와 2개의 XOR 게이트를 통과하면 결과를 얻을 수 있는 반면에 본 논문에서는 1개의 AND와 2개의 XOR 게이트 통과 시간만을 요한다. 만약 k=7일 때 기존논문에서는 AND와 XOR가 각각 7개씩 통과해야 결과를 얻을 수 있지만 본 논문에서는 1개의 AND와 5개의 XOR 게이트만 통과하면 결과를 얻을 수 있다. 만약 2입력 게이트가 아니고 3 또는 4입력 게이트를 새로 설계한다면 더욱 회로가 간략화 되고 시간 지연도 적게 걸릴 수 있다.

V. 결론

본 논문에서는 유한체 $GF(2^m)$상에서 승산기에서 필요한 원시기약 다항식 발생기의 알고리즘을 제시하였으며, 제시된 기약다항식 발생 알고리즘을 이용하여 회로를 구성하였다. 제시된 승산알고리즘을 이용하여 흐름도로 나타내었으며, X^{m+k}라 표현할 때 k=1,2,3,4,5,6,7일 때의 기약다항식 발생기의 다항식을 구하였다. 그 결과를 얻기 위한 시간 차이가 없었으나 k의 값이 증가할수록 결과를 얻기 위한 시간이 짧게 걸리나 회로의 복잡성은 증가하는 단점이 있다. 드러나 집적회로의 발달로 인하여 회로의 복잡성보다는 수행시간을 더욱 중요시되고 있다.

참고문헌(또는 Reference)

[1] S.L. Hurst, "Multiple-valued logic-its future," *IEEE Trans. Computers*, vol 30, pp. 1161-1179, Dec. 1984.

[2] C.S. Yeh, I.S. Reed, and T.K. Truong, "Systolic Multipliers for Finite Fields $GF(2^m)$," *IEEE Trans. Computers*, vol. 33, no. 4, pp. 357-360, Apr. 1384.

[3] P.A. Scott, S.E. Tarvares and L.E. Peppard, "A fast multiplier for $GF(2^m)$," *IEEE J.Select. Areas Commun.*, vol. SAC-4, Jan. 1986.

[4] E.D. Mastrovito, "VLSI Architectures for Multiplication Over Finite Field $GF(2^m)$," *Applied Algebraic Algorithms, and Error-Correcting Code, Proc.* Sixth Int'l Conf., AAECC-6, T.Mora,ed., pp.297-309,Rome, July 1988.

[5] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," *PhD thesis*, Linkoping Univ., Dept. of Electrical Eng., Linkoping, Sweden, 1991.