# 디지털 컨텐츠 보호를 위한 ECC 용 곱셈기 구현방법
# (ECC Multiplier over Galois Field for Digital Contents Protection)

김형중
강원대학교 제어계측공학과

**Hyoung Joong Kim**
**Kangwon National University**
**khj@cc.kangwon.ac.kr**

요약

디지털 컨텐츠의 정보보호는 근래 매우 중요한 기술로 등장했다. 애써 만든 디지털 컨텐츠가 무차별적으로 복제되어 배포되면 컨텐츠 제공자에게는 커다란 경제적 손실을 입히기 때문에 이를 보호하려는 기술이 개발되고 있다. 특별히 DVD 나 MP3, AAC 등 네트워크 환경에서 고급 품질의 영상이 품질의 손상 없이 복제되어 네트워크를 통해 클릭 한 번으로 배포될 수 있기 때문에 이에 대한 대처가 시급한 실정이다. 따라서, 이에 대한 해결책으로 타원곡선 암호시스템을 사용하는 상황에서 필요한 갱신가능 구조를 고려한 Massey-Omura 곱셈기를 제안한다.

## 1. 서론

디지털 컨텐츠의 정보보호는 근래 매우 중요한 기술로 등장했다. 애써 만든 디지털 컨텐츠가 무차별적으로 복제되어 배포되면 컨텐츠 제공자에게는 커다란 경제적 손실을 입히기 때문에 이를 보호하려는 기술이 개발되고 있다. 특별히 DVD 나 MP3, AAC 등 네트워크 환경에서 고급 품질의 영상이 품질의 손상 없이 복제되어 네트워크를 통해 클릭 한 번으로 배포될 수 있기 때문에 이에 대한 대처가 시급한 실정이다. 그런데 멀티미디어 환경에서는 고도로 정교한 암호알고리듬을 사용해서 어떠한 공격에도 견디도록 하는 방법은 사용하기 곤란하다. 그런 방법은 사용자에게 복잡한 절차를 요구할 수 있기 때문이다. 멀티미디어 사용자는 그저 수동적으로 컨텐츠를 이용할 뿐이므로 복잡한 절차를 요구하는 것을 싫어하게 된다. 그래서 암호 알고리듬도 성능 완화가 불가피하다. 그 결과 암호 알고리듬이 공격에 취약해질 가능성이 있다. 물론, 고도로 복잡한 알고리듬일지라도 공격에 강하다는 보장도 없다. 따라서 암호 알고리듬은 언제라도 공격받아 깨질 수 있다는 전제가 있어야 한다. 그러므로 이런 환경에서 사용할 암호 알고리듬은 갱신가능 (Renewable) 특성을 지녀야 한다. 즉 암호 알고리듬이나 키가 노출되면 즉시 갱신해서 새 것으로 대치해야 한다. 그러기 위해서는 회로도 이런 환경에 적합하도록 설계되어야 한다.

갈로와 필드 (Galois Field)는 정보통신분야에서 널리 사용되고 있다. 특별히 갈로와 필드에서의 연산은 ECC (Elliptic Curve Cryptosystem) 같은 암호이론이나 BCH 및 Reed-Solomon 코드 같은 코딩이론에서 매우 중요한 도구로 쓰이고 있다 [1,2,4,7]. 이들 응용에서는 덧셈, 곱셈, 나눗셈이 주로 이용된다. 그런데 이런 연산은 주로 표준기지 (Standard Basis) 표현에 의해 구현했다. 그렇지만, Massey-Omura 알고리듬 [6]이 소개되면서 정규기지 (Normal Basis) 표현도 각광을 받게 되었다. 정규기지에서는 제곱의 계산이 단순히 이진 표현의 순환이동 (Cyclic Shift) 만으로 구현될 수 있다는 장점이 있다.

지금까지 표준기지에서는 생성다항식 (Generaring Polynomial)로 트리노미얼 (Trinomial), 즉 $X^m + X^k + 1$이 주로 쓰였다. 그러나, 근래 AOP (All-One-Polynomial), 즉 모든 계수의 값이 1인 $X^m + X^{m-1} + \cdots + X + 1$도 유용하게 쓰이고 있다. 그런데 AOP 는 ESP (Equally Spaced Polynomial)의 특수한 형태로, AOP 나 ESP 모두 중요한 성질을 지니고 있다. 그래서, 이 둘의 성질도 소개하기로 한다. AOP 는 ONB (Optimal Normal Basis) 표현에서 중요한 역할을 담당한다.

이 논문에서는 곱셈기의 형태와 기본 개념을 소개하는데 주안점을 두고 있다. 현재 널리 쓰이고 있는 것들 가운데 Mastrovito 곱셈기 [5], Massey-Omura 곱셈기 [6], Itoh-Tsujii 곱셈기 [3]를 소개한다. 곱셈기의 연산성능은 기지 및 생성다항식의 선택에 따라 달라질 수 있다.

이 논문은 현재까지 이루어진 수학적 결과를 비교해서 정리한 것으로, 각 곱셈기의 수학적 배경과 VLIS로 설계할 때 얻을 수 있는 장점들에 초점을 맞추었다. 아울러 병렬계산을 통해 성능을 개선하고, Reconfigurable Mesh 구조를 채택해서 고속 구현이 가능함을 보인다.

## 2. 표준기지에서의 곱셈개념

$GF(2^m)$ 위의 $m$개의 선형독립인 원소들의 집합을 $GF(2^m)$의 기지 (Basis)라 한다. $\omega$를 차수 $m$인 원시다항식 (Primitive Polynomial) $P(X)$의 근이라고 할 때

$$\{1, \omega, \omega^2, \omega^3, \cdots, \omega^{m-1}\}$$

은 $GF(2^m)$의 기지라 하며, 특별히 이 기지를 $GF(2^m)$ 위의 표준기지라고 한다. 일반적으로 표준기지에서 곱셈을 어떻게 구현하는지 예 1을 살펴보자.

예 1. $GF(2^5)$ 상의 임의의 두 원소 $A$와 $B$의 곱셈 $C = AB$를 표준기저로 계산하는 방법을 살펴보자. 두 수 $A$와 $B$는 아래와 같다.

$$A = a_0\omega^0 + a_1\omega^1 + a_2\omega^2 + a_3\omega^3 + a_4\omega^4$$
$$B = b_0\omega^0 + b_1\omega^1 + b_2\omega^2 + b_3\omega^3 + b_4\omega^4$$

두 수의 곱 $C = AB$는 다음과 같이 쓸 수 있다.

$$C = AB = A\sum_{k=0}^{4} b_k\omega^k = \sum_{k=0}^{4} b_k(A\omega^k) \qquad (1)$$

가 된다. 식 (1)을 행렬식으로 표현하면 아래와 같이 된다.

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 & c_4 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 \end{bmatrix} \begin{bmatrix} A\omega^0 \\ A\omega^1 \\ A\omega^2 \\ A\omega^3 \\ A\omega^4 \end{bmatrix} \qquad (2)$$

만일 원시다항식이 $P(X) = X^5 + X^2 + 1$이라면, $A\omega^i$은 아래와 같이 계산된다.

$$\begin{aligned}
A\omega^0 &= a_0\omega^0 + a_1\omega^1 + a_2\omega^2 + a_3\omega^3 + a_4\omega^4 \\
A\omega^1 &= a_0\omega^1 + a_1\omega^2 + a_2\omega^3 + a_3\omega^4 + a_4\omega^5 \\
&= a_0\omega^1 + a_1\omega^2 + a_2\omega^3 + a_3\omega^4 + a_4(1+\omega^2) \\
&= a_4\omega^0 + a_0\omega^1 + (a_1 + a_4)\omega^2 + a_2\omega^3 + a_3\omega^4 \\
A\omega^2 &= a_3\omega^0 + a_4\omega^1 + (a_0 + a_3)\omega^2 + (a_1 + a_4)\omega^3 \\
&\quad + a_2\omega^4 \\
A\omega^3 &= a_2\omega^0 + a_3\omega^1 + (a_2 + a_4)\omega^2 + (a_0 + a_3)\omega^3 \\
&\quad + (a_1 + a_4)\omega^4 \\
A\omega^4 &= (a_1 + a_4)\omega^0 + a_2\omega^1 + (a_1 + a_3 + a_4)\omega^2 \\
&\quad + (a_2 + a_4)\omega^3 + (a_0 + a_3)\omega^4
\end{aligned}$$

따라서, 위의 결과를 식 (1) 또는 식 (2)에 대입하여 축약 (Reduction)하면 아래와 같이 된다.

$$\begin{aligned}
c_0 &= a_0b_0 + a_4b_1 + a_3b_2 + a_2b_3 + (a_1 + a_4)b_4 \\
c_1 &= a_1b_0 + a_0b_1 + a_4b_2 + a_3b_3 + a_2b_4 \\
c_2 &= a_2b_0 + (a_1 + a_4)b_1 + (a_0 + a_3)b_2 + (a_2 + a_4)b_3 \\
&\quad + (a_1 + a_3 + a_4)b_4 \\
c_3 &= a_3b_0 + a_2b_1 + (a_1 + a_4)b_2 + (a_0 + a_3)b_3 \\
&\quad + (a_2 + a_4)b_4 \\
c_4 &= a_4b_0 + a_3b_1 + a_2b_2 + (a_1 + a_4)b_3 + (a_0 + a_3)b_4
\end{aligned}$$

이상에서 볼 수 있듯이 곱셈은 매우 복잡하게 이루어진다. 만일 $A = \omega^{15}$, $B = \omega^{20}$일 때 위의 식을 이용할 수도 있고, 아래와 같이 계산할 수도 있다.

$$\begin{aligned}
C = AB &= \omega^{15}\omega^{20} \\
&= (1 + \omega + \omega^2 + \omega^3 + \omega^4)(\omega^2 + \omega^3) \\
&= (\omega^2 + \omega^3 + \omega^4 + \omega^5 + \omega^6) \\
&\quad + (\omega^3 + \omega^4 + \omega^5 + \omega^6 + \omega^7) \\
&= \omega^2 + \omega^7 \\
&= \omega^2 + (\omega^2 + \omega^4) \\
&= \omega^4
\end{aligned}$$

✡

## 3. Mastrovito 곱셈기의 개념

두 수 $A$와 $B$를 벡터 형식으로 표현하면

$$A = \sum_{i=0}^{m-1} a_i\omega^i = [a_0 a_1 \cdots a_{m-1}]$$

와

$$B=\sum_{i=0}^{m-1} b_i \omega^i = [b_0 b_1 \cdots b_{m-1}]$$

의 곱을 $C$라 하면 그 결과는 다음

$$C=\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \omega^i \omega^j$$

가 된다. 그런데, 곱셈 도중 기저에 포함되지 않는 수들이 나타나므로 그런 수들을 기저의 선형 결합으로 표시하기 위해 $Q$라는 기저축약행렬 (Basis Reduction Matrix)을 다음과 같이 정의한다.

$$\begin{bmatrix} \omega^m \\ \omega^{m+1} \\ \vdots \\ \omega^{2m-2} \end{bmatrix} = Q \begin{bmatrix} \omega^0 \\ \omega^1 \\ \vdots \\ \omega^{m-1} \end{bmatrix}$$

Mastrovito 곱셈기에서 $m \times m$ 곱셈행렬 $Z$는 $A$와 $Q$의 함수로

$$z_{ij} = \begin{cases} a_i & j=0; \\ & i=0,\cdots,m-1 \\ u(i-j)a_{i-j} & \\ +\sum_{t=0}^{j-1} q_{j-1-t,i}\, a_{m-1-t} & j \neq 0; \\ & i=0,\cdots,m-1 \end{cases} \quad (3)$$

과 같이 정의되며, 스텝함수 $u(k)$는

$$u(k)=\begin{cases} 1 & k \geq 0 \\ 0 & k < 0 \end{cases}$$

가 된다. $C=AB$를 Mastrovito 곱셈기는 $GF(2)$에서 $C=ZB$의 곱셈으로 구현한다.

예 **2.** 예 1의 곱셈을 식 (3)을 이용해서 다시 정리하라.

$P(X)=X^5+X^2+1$을 이용해서 $4\times5$ 기저축약행렬 $Q$를

$$\begin{bmatrix} \omega^5 \\ \omega^6 \\ \omega^7 \\ \omega^8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega^0 \\ \omega^1 \\ \omega^2 \\ \omega^3 \\ \omega^4 \end{bmatrix}$$

와 같이 정의할 수 있다. 식 (3)을 이용해서 곱셈행렬 $Z$는

$$Z=\begin{bmatrix} a_0 & a_4 & a_3 & a_2 & a_1+a_4 \\ a_1 & a_0 & a_4 & a_3 & a_2 \\ a_2 & a_1+a_4 & a_0+a_3 & a_2+a_4 & a_1+a_3+a_4 \\ a_3 & a_2 & a_1+a_4 & a_0+a_3 & a_2+a_4 \\ a_4 & a_3 & a_2 & a_1+a_4 & a_0+a_3 \end{bmatrix}$$

와 같다. ✡

예 1과 2에서는 트리노미얼 가운데 하나인 $P(X)=X^5+X^2+1$을 사용했는데 이제는 널리 쓰이는 AOP와 ESP를 소개하기로 한다.

정의 **1.** $P(X)=X^m+X^{tr}+\cdots+X^{2r}+X^r+1$, $m=(t+1)r$를 차수 $m$인 $r$-ESP라고 부른다. 1-ESP를 AOP라고 부른다.

AOP $P(X)=X^m+\cdots+X^2+X+1$에서 $m+1$이 소수이고, 동시에 2가 $Z_{m+1}$ 위에서 생성자(Generator)이면 그 AOP는 $GF(2)$에서 기약다항식 (irreducible Polynomial)이며, 따라서 필드 $GF(2^m)$을 생성할 수 있다. 현재 인수분해가 불가능한 기약다항식인 $m$으로 알려진 값은 2, 4, 10, 12 , 18, 28, 36, 52, 58, 등이 있다. AOP를 이용하서 곱셈을 구현할 경우

$$\begin{bmatrix} \omega^m \\ \omega^{m+1} \\ \omega^{m+2} \\ \vdots \\ \omega^{2m-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega^0 \\ \omega^1 \\ \omega^2 \\ \vdots \\ \omega^{m-1} \end{bmatrix}$$

$$Z=\begin{bmatrix} a_0 & a_{m-1} & a_{m-1}+a_{m-2} & \cdots & a_2+a_1 \\ a_1 & a_0+a_{m-1} & a_{m-2} & \cdots & a_3+a_1 \\ a_2 & a_1+a_{m-1} & a_0+a_{m-2} & \cdots & a_4+a_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2}+a_{m-1} & a_{m-3}+a_{m-2} & \cdots & a_0+a_1 \end{bmatrix}$$

과 같이 된다. 그런데 이때의 $Z$가 특별한 형태를 지니고 있어 $Z=Z_1+Z_2$로

$$Z_1=\begin{bmatrix} a_0 & 0 & a_{m-1} & \cdots & a_2 \\ a_1 & a_0 & 0 & \cdots & a_3 \\ a_2 & a_1 & a_0 & \cdots & a_4 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m-1} & a_{m-2} & a_{m-3} & \cdots & a_0 \end{bmatrix},$$

$$Z_2 = \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_1 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_1 \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{m-1} & a_{m-2} & \cdots & a_1 \end{bmatrix}$$

분해가 가능하다. 이런 분해가 곱셈기 성능향상에 도움이 될 수 있다.

## 3. Itoh-Tsujii 곱셈기

한편, AOP 인 $P(X) = X^4 + X^3 + X^2 + X + 1$ 에 대해 $P(X)$ 의 근인 $\omega = X$ 의 멱승으로는 $\omega^0$ 부터 $\omega^{14}$ 까지 15 개의 원소를 생성할 수 없다. 그 이유는 $P(X)$ 가 기약다항식이는 하지만 원시다항식은 아니기 때문이다. $P(X)$ 의 근인 $\omega = X$ 의 멱승으로 생성할 수 있는 수는

$$\begin{aligned} \omega^0 &= 1 \\ \omega^1 &= X \\ \omega^2 &= X^2 \\ \omega^3 &= X^3 \\ \omega^4 &= 1 + X + X^2 + X^3 \\ \omega^5 &= 1 \\ \omega^6 &= X \end{aligned}$$

와 같이 $\omega^0$ 부터 $\omega^4$ 까지 5 개에 불과하다. 이 점은 이미 예제 2 에서도 언급한 바 있다. 그런데, 덧셈에서 한 가지 문제가 더 발생한다. 예를 들어 $\omega^0 + \omega^1 = 1 + X$ 가 되는데 $X + 1$ 이라는 원소가 $\{1, X, X^2, X^3, X^3 + X^2 + X + 1\}$ 에는 존재하지 않는다. 다시 말해, 덧셈이 정의되지 않는다.

예제 **3**: 다항식 $P(X) = X^4 + X^3 + X^2 + X + 1$ 에 대해 $\omega = X + 1$ 를 이용해서 $\omega^0$ 부터 $\omega^{14}$ 까지 15 개의 원소를 생성하라.

$$\begin{aligned} \omega^0 &= 1 \\ \omega^1 &= 1 + X \\ \omega^2 &= 1 + X^2 \\ \omega^3 &= 1 + X + X^2 + X^3 \\ \omega^4 &= X \\ \omega^5 &= X + X^2 \\ \omega^6 &= X + X^3 \\ \omega^7 &= 1 + X^2 + X^3 \\ \omega^8 &= X^2 \\ \omega^9 &= X^2 + X^3 \\ \omega^{10} &= 1 + X + X^2 \\ \omega^{11} &= 1 + X^3 \\ \omega^{12} &= X^3 \\ \omega^{13} &= 1 + X + X^3 \\ \omega^{14} &= X + X^2 + X^3 \end{aligned}$$

✡

예제 3 에서 본 $X^4 + X^3 + X^2 + X + 1$ 는 모든 다항식 항의 계수가 1 이기 때문에 AOP (All-One-Polynomial)이라고 부른다. 정의 1 에서 보듯 AOP 는 ESP (Equally Spaced Polynomial)의 특수한 형태가 된다.

Itoh-Tsujii 의 곱셈기는 $A$ 와 $B$ 에 대해

$$A = a_0\omega^0 + a_1\omega^1 + a_2\omega^2 + \cdots + a_m\omega^m$$
$$B = b_0\omega^0 + b_1\omega^1 + b_2\omega^2 + \cdots + b_m\omega^m$$

곱 $C = AB$ 를

$$C = c_0\omega^0 + c_1\omega^1 + c_2\omega^2 + \cdots + c_m\omega^m$$

로 표시할 때 $C$ 의 계수 $c_k$ 는 다음 식과 같이 계산한다.

$$c_k = \sum_{i+j=k \ (\mathrm{mod}\ m+1)} a_i b_j \ (\mathrm{mod}\ 2) \quad (0 \le k \le m)$$

Itoh-Tsujii 의 곱셈기는 $P(X)$ 가 AOP 인 경우를 전제로 한 것이다. 이 식이 의미하는 바를 예제 4 에서 살펴보기로 한다.

예제 **4**: 예제 3 에서 $\omega = X + 1$ 로 $\omega^0$ 부터 $\omega^{14}$ 까지 만든 15 개의 원소 가운데 $\omega^1 \cdot \omega^2$ 을 계산하기로 하자. $\omega^1 = 1 + X$, $\omega^2 = 1 + X^2$ 이므로

$$(a_0, a_1, a_2, a_3) = (1,\ 1,\ 0,\ 0),$$
$$(b_0, b_1, b_2, b_3) = (1,\ 0,\ 1,\ 0)$$

이다. 따라서

$$\begin{aligned} c_0 &= a_0b_0 + a_2b_3 + a_3b_2 = 1 + 0 + 0 = 1, \\ c_1 &= a_0b_1 + a_1b_0 + a_3b_3 = 0 + 1 + 0 = 1, \\ c_2 &= a_0b_2 + a_2b_0 + a_1b_1 = 1 + 0 + 0 = 1, \\ c_3 &= a_0b_3 + a_1b_2 + a_2b_1 + a_3b_0 = 0 + 1 + 0 + 0 = 1 \end{aligned}$$

이므로 $\omega^1 \cdot \omega^2 = 1 + X + X^2 + X^3 = \omega^3$ 이 된다.

✡

## 4. Massey-Omura 곱셈기

$GF(2^m)$ 상의 임의의 두 원소 $A$와 $B$의 곱셈 $C=AB$를 정규기저로 계산하는 방법을 살펴보자. 두 원소 $A$와 $B$를 수식으로 나타내면 아래와 같다.

$$A = a_0\omega^1 + a_1\omega^2 + a_2\omega^{2^2} + \cdots + a_{m-1}\omega^{2^{m-1}}$$
$$B = b_0\omega^1 + b_1\omega^2 + b_2\omega^{2^2} + \cdots + b_{m-1}\omega^{2^{m-1}}$$

$A=[a_0\ a_1\ a_2\ \cdots a_{m-1}]$, $W=[\omega\ \omega^2\ \omega^{2^2}\ \cdots \omega^{2^{m-1}}]$ 으로 표현할 때 두 원소의 곱은 다음과 같이 쓸 수 있다.

$$C = AB = (AW^T)(BW^T)^T = AMB^T$$

가 되고, 곱셈행렬 $M$은 다음과 같이 정의된다.

$$M = W^TW = \begin{bmatrix} \omega^{2^0+2^0} & \omega^{2^0+2^1} & \cdots & \omega^{2^0+2^{m-1}} \\ \omega^{2^1+2^0} & \omega^{2^1+2^1} & \cdots & \omega^{2^1+2^{m-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{2^{m-1}+2^0} & \omega^{2^{m-1}+2^1} & \cdots & \omega^{2^{m-1}+2^{m-1}} \end{bmatrix}$$

그런데 $M$은 아래와 같이 다시 표현할 수 있다.

$$M = M_0\omega + M_1\omega^2 + \cdots + M_{m-1}\omega^{2^{m-1}}$$

여기서 행렬 $M_k$ $(0 \le k \le m-1)$의 원소는 $GF(2)$에 속한다. $M_k$에서 $0, 1, \cdots, m-1$로 각 행과 열을 번호로 표시할 수 있다. 그렇게 하면 $M_k$의 열 $i$와 행 $j$는 $\omega^{2^i+2^j}$를 정규기저로 표현했을 때 $\omega^{2^k}$의 계수로 표시된다. 따라서 곱셈의 결과는

$$c_{m-1-k} = AM_{m-1-k}B^T, \quad (4)$$
$$= A^{(k)}M_{m-1}B^{(k)T}, \quad k = 0, 1, \cdots, m-1 \quad (5)$$

로 표현할 수 있다. 여기서 $A^{(k)}$는 $A$를 오른쪽으로 $k$번 돌린 것을 의미한다. 이것이 Massey-Omura 곱셈기의 기본 원리이다. 이 곱셈기는 병렬계산이 가능하며, Reconfigurable Mesh 구현이 용이하다는 장점을 지니고 있다.

예제 **5:** AOP $p(X) = X^4 + X^3 + X^2 + X + 1$에 대해 곱셈행렬 $M$를 구성하는 방법을 조사해보자.

$M$의 정의에 의해

$$M = \begin{bmatrix} \omega^2 & \omega^3 & \omega^5 & \omega^9 \\ \omega^3 & \omega^4 & \omega^6 & \omega^{10} \\ \omega^5 & \omega^6 & \omega^8 & \omega^{12} \\ \omega^9 & \omega^{10} & \omega^{12} & \omega^{16} \end{bmatrix}$$

가 된다. 그런데 여기서 주어진 기약다항식 AOP $p(X) = X^4 + X^3 + X^2 + X + 1$의 성질에 따라

$$X^4 = X^3 + X^2 + X + 1$$
$$X^5 = X^4 + X^3 + X^2 + X = 1$$
$$X^6 = X$$
$$X^7 = X^2$$
$$X^8 = X^3$$
$$X^9 = X^4$$

인 점을 이용하면 $M$은 다음과 같이 표현된다.

$$M = \begin{bmatrix} \omega^2 & \omega^3 & 1 & \omega^4 \\ \omega^3 & \omega^4 & \omega & 1 \\ 1 & \omega & \omega^3 & \omega^2 \\ \omega^4 & 1 & \omega^2 & \omega \end{bmatrix}$$

위의 $M$은 정규기저로 표현되지 않았기 때문에 $\omega^3 = \omega^8$, $1 = \omega + \omega^2 + \omega^4 + \omega^8$으로 치환해서 모든 원소를 정규기저로 다시 정리하고 정리하면 다음과 같은 행렬을 구할 수 있다.

$$M_{m-1} = M_3 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

## 5. Reconfigurable Mesh 곱셈기

식 (4)를 보면 입력 $A$와 $B$는 변하지 않고 $M_k$만 변화시키면서 곱셈이 가능하다. 다시 말해

$$c_k = AM_kB^T$$

가 되므로, $c_k$는 $M_k$만 있으면 바로 계산이 병렬로 이루어질 수 있다. 일반적으로 식 (5)처럼 하나의 $M_k$에 대해 $A$와 $B$를 회전시키면서 곱셈하는 것이 관례였다. 그러나, 이 논문에서는 $M_k$를 각각 구현함으로써 병렬계산의 효과를 최대한 활용할 수 있음을 보인다. 그런데, 이럴 경우 VLSI로 구현하면 곱셈기를 무려 $k$개나 확보해야 하는 부담이 생긴다. 물론, 그런 상황이 발생한다면 실용적 가치가 전혀 없게 된다.

따라서, 곱셈기는 하나가 있지만 그 곱셈기가 상황에 따라 구조를 바꾸는 Reconfigurable Mesh의 성질을 활용해서 하드웨어의 구조는 단순화시키고, 계산성능을 향상시키자는 것이다. Systolic Array가 하드웨어의 규칙성 (Regularity) 측면을 강조하고, 입력이 연속적으로 공급되는 구조를 고려했다면, 이 방식에서는 입력이 연속적으로 공급되면서 하드웨어 구조를 예제 6처럼 가변적으로 변화시키면서 성능을 향상시키자는 것이다.

예제 **6:** 예제 5에 대해 4개의 $M_k$를 구성해보자.

$$M_0 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad M_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \quad M_3 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

✡

식 (5)를 바탕으로 회로를 구성한 것이 바로 Massey-Omura 곱셈기로 그 모양이 그림 1과 같다.

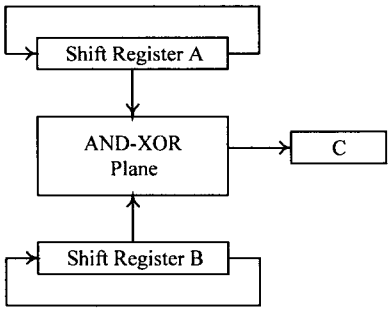


그림 **1. Massey-Omura** 곱셈기

Massey-Omura 곱셈기는 기저를 하나 (정규기지) 사용한다는 장점이 있다. 또 AOP를 사용하면 타입 1 최적정규기지가 되면서 회로가 간단해진다는 장점이 있다. 그러나, 다항식이 달라지면 회로를 다시 설계해야 하는 단점이 있다. 다시 말해 갱신가능 회로가 되지 못한다. 그래서 식 (5) 대신 식 (4)를 사용하자는 것이다.

그림 2가 식 (4)를 기반으로 설계하는 회로 전체의 개념을 보여주고 있다. 예제 6을 보면 행렬에 규칙성이 존재한다. 물론, 그 규칙성 때문에 식 (5)가 가능하다. 따라서 회로를 그림 2처럼 Reconfigurable Mesh로 대치하고, 행렬의 규칙성에 따라 회로를 변경시키면 된다.

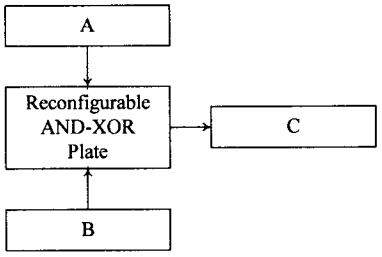


그림 **2. Massey-Omura** 곱셈기의 변형

## 6. 결론

갱신가능한 암호 알고리듬 회로를 VLSI로 설계할 수 있음을 보였다. Reconfigurable Mesh가 여기에 적합하며 기본 골격은 Massey-Omura 회로를 이용하면 됨을 보였다. 갱신가능 알고리듬의 중요성에 비추어 볼 때 이 분야의 연구활성화가 기대된다.

## 후기

이 연구는 한국과학재단 특정연구과제 97-01-01-02-01-3 연구비 지원에 의해 이루어졌다.

## 참고문헌

[1]. E. R. Berlekamp, "Bit-serial Reed-Solomon encoder," *IEEE Transaction on Information Theory*, vol. 28, pp. 869-874, 1982.

[2]. M. A. Hasan, V. K. Bhargava, and T. Le-Ngoc, "Algorithms and architectures for the design of a VLSI Reed-Solomon codec," *Reed-Solomon Codes and Their Applications*, ed. S. B. Wicker and V. K. Bhargava, IEEE Press, pp. 60-107, 1994.

[3]. T. Itoh, and S. Tsujii, "Structure of parallel multipliers for a class of fields $GF(2^m)$," *Information and Computations*, vol. 83, pp. 21-40, 1989.

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

[5]. E. Mastrovito, VLSI architectures for computation in Galois fields," Ph.D. Dissertation, Dept. Electrical Engineering, Linkoping University, Sweden, 1991.

[6]. J. Omura, and J. Massey, "Computational method and apparatus for finite field arithmetic," U.S. Patent No. 4,587,627, 1986.

[7]. I. S. Reed, and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the ACM*, vol. 8, pp. 300-304, 1960.