All-One Polynomial에 의해 정의된 유한체 $GF(2^m)$ 상의 새로운 Low-Complexity Bit-Parallel 정규기저 곱셈기

(A New Low-complexity Bit-parallel Normal Basis Multiplier for $GF(2^m)$ Fields Defined by All-one Polynomials)

장 용 희 † 권 용 진 ††
(Yong-Hee Jang) (Yong-Jin Kwon)

요 약 대부분의 공개키 기반 암호시스템은 유한체 $GF(2^m)$ 상의 산술 연산들을 기반으로 구축된다. 이들 연산 중 덧셈을 제외한 다른 연산들은 곱셈 연산을 반복하여 계산되므로, 곱셈 연산의 효율적인 구현은 공개키 기반 암호시스템에서 매우 중요하다. 본 논문에서는 All-One Polynomial에 의해 정의된 $GF(2^m)$ 상의 효율적인 Bit-Parallel 정규기저 곱셈기를 제안한다. 게이트 및 시간적인 면에서 본 곱셈기의 복잡도(complexity)는 이전에 제안된 같은 종류의 곱셈기 보다 낮거나 동일하다. 또한, 본 논문의 곱셈기는 아키텍처가 규칙적(regular)이어서 VLSI 구현에 적합하다.

키워드 : 공개키 기반 암호시스템, 유한체 $GF(2^m)$, 곱셈 연산, All-One 다항식, 정규기저, 곱셈기, VLSI

Abstract Most of pubic-key cryptosystems are built on the basis of arithmetic operations defined over the finite field $GF(2^m)$. The other operations of finite fields except addition can be computed by repeated multiplications. Therefore, it is very important to implement the multiplication operation efficiently in public-key cryptosystems. We propose an efficient bit-parallel normal basis multiplier for $GF(2^m)$ fields defined by All-One Polynomials. The gate count and time complexities of our proposed multiplier are lower than or equal to those of the previously proposed multipliers of the same class. Also, since the architecture of our multiplier is regular, it is suitable for VLSI implementation.

Key words : Public-key cryptosystems, Finite fields $GF(2^m)$, Multiplication operation, All-One Polynomials, Normal basis, Multiplier, VLSI

1. 서 론

유한체 $GF(2^m)$ 상의 산술 연산, 즉 덧셈, 곱셈, 제곱(squaring) 및 곱셈역원 연산 등은 부호이론, 컴퓨터 대수, 암호이론과 같은 분야의 어플리케이션, 특히 공개키 기반 암호시스템에서 많이 사용된다[1]. 이들 어플리케이션이 효율적으로 구현되기 위해서는 이들 연산에 대해서 게이트 및 시간적인 복잡도(complexity) 면에서 효율적인 알고리즘과 하드웨어 구조가 필요하다[2]. 덧셈은 매우 간단히 구현되지만, 다른 연산들은 더 복잡하다. 덧셈 연산을 제외한 다른 연산들, 즉 exponentiation, 나눗셈 및 곱셈역원 연산은 곱셈 연산을 반복 계산하여 수행되므로 곱셈 연산을 효율적으로 구현하는 것은 매우 중요하다[3].

지금까지 일반적인 기약 다항식에 의해 정의된 유한체 $GF(2^m)$ 상의 Bit-Parallel 곱셈기가 많이 제안되어 왔다. 그러나 이들 곱셈기는 시스템 복잡도가 커서 공개키 기반 암호시스템과 같은 암호 분야의 application에 비효율적이다[3]. 1989년에 Itoh와 Tsujii는 시스템 복잡도를 감소시키기 위해 차수 m의 기약 All-One Polynomial(AOP)에 의해 정의된 유한체 $GF(2^m)$ 상의 low-complexity Bit-Parallel 곱셈기를 제안했다[4]. 이

· 본 논문은 과학기술부·한국과학재단지정 「한국항공대학교 인터넷정보검색연구센터」의 연구비 및 IDEC의 지원으로 수행되었음
† 학생회원 : 한국항공대학교 정보통신공학과
yhjang@mail.hangkong.ac.kr
†† 정 회 원 : 한국항공대학교 전자,정보통신,컴퓨터공학부 교수
yjkwon@tikwon.hangkong.ac.kr
논문접수 : 2003년 5월 9일
심사완료 : 2003년 9월 16일

이후로 AOP에 의해 정의된 $GF(2^m)$ 상의 Bit-Parallel 곱셈기가 많이 제안되어 왔다(예 [1-3,5-8]). 그리고 사용되는 기저의 종류에 따라, 유한체 $GF(2^m)$ 상의 Bit-Parallel 곱셈기는 다항식 기저 곱셈기, 정규기저 곱셈기, 그리고 이중 기저(dual basis) 곱셈기로 분류되며 이들 곱셈기는 서로 다른 장단점을 가진다[8].

본 논문에서는 AOP에 의해 정의된 $GF(2^m)$ 상의 Bit-Parallel 곱셈기를 제안하며 이 $GF(2^m)$ 상의 임의의 원소 표현을 위해 정규기저를 사용한다. 제안된 곱셈기의 AND 및 XOR 게이트의 복잡도는 각각 m^2 및 m^2-1 이며, 시간 시간복잡도는 $T_A+(1+\lceil \log_2(m-1) \rceil)T_X$으로 이전에 제안된 같은 종류의 곱셈기 보다 복잡도가 낮거나 같다. 그리고 본 논문에서 제안한 Bit-Parallel 정규기저 곱셈기는 아키텍처가 규칙적(regular)이여 VLSI 구현에 적합한 장점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서 정규기저 표현을 이용한 $GF(2^m)$ 상의 곱셈연산과 AOP에 의해 정의되는 유한체 $GF(2^m)$ 상의 원소 표현에 대해서 설명한다. 제3장에서는 본 논문에서 제안하는 곱셈 연산 알고리즘 및 곱셈기의 아키텍처에 대해서 설명한다. 그런 다음, $m=10$인 경우를 예를 들어 본 논문에서 제안하는 곱셈기를 설명한다. 그리고 제4장에서 이전에 제안된 같은 종류의 곱셈기와 게이트 및 시간지연 면에서 복잡도를 서로 비교하며, 마지막으로 제5장에서 결론을 맺는다.

2. 예비 지식

2.1 정규기저 표현을 이용한 곱셈

유한체 $GF(2^m)$의 임의의 원소 A는 $GF(2)$ 상에서 정규기저(Normal Basis), $N=\{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\}$ ($\beta \in GF(2^m)$)를 사용해서 아래와 같이 표현할 수 있다.

$$A = a_0\beta + a_1\beta^2 + a_2\beta^{2^2} + \cdots + a_{m-1}\beta^{2^{m-1}}, \quad a_i \in GF(2). \tag{1}$$

또한 위 표현을 이용해서 A는 벡터, $(a_0, a_1, \cdots, a_{m-1})$으로도 표현할 수 있다.

A와 B를 $GF(2^m)$의 원소라고 하면, A와 B는 위의 정규기저 N를 사용해서 아래와 같이 표현된다.

$$A = (a_0, a_1, \cdots, a_{m-1}) = \sum_{i=0}^{m-1} a_i\beta^{2^i}, \tag{2}$$

$$B = (b_0, b_1, \cdots, b_{m-1}) = \sum_{j=0}^{m-1} b_j\beta^{2^j} \tag{3}$$

그리고 C를 A와 B의 곱이라 하면, C는 아래와 같이 계산된다[1].

$$C = AB = (\underline{a} \times \underline{\beta}^T) \times (\underline{\beta} \times \underline{b}^T) = \underline{a} \times \boldsymbol{M} \times \underline{b}^T, \tag{4}$$

여기서 $\underline{a}=[a_0, a_1, \cdots, a_{m-1}]$, $\underline{b}=[b_0, b_1, \cdots, b_{m-1}]$, $\underline{\beta}=[\beta, \beta^2, \cdots, \beta^{2^{m-1}}]$이고,

T는 vector transposition을 나타낸다. 그리고 $\boldsymbol{M}$은 아래와 같이 정의된다.

$$\boldsymbol{M} = \underline{\beta}^T \times \underline{\beta} = [\beta^{2^i+2^j}], \quad 0 \le i, j \le m-1$$

$$= \begin{bmatrix} \beta^{2^0+2^0} & \beta^{2^0+2^1} & \cdots & \beta^{2^0+2^{m-1}} \\ \beta^{2^1+2^0} & \beta^{2^1+2^1} & \cdots & \beta^{2^1+2^{m-1}} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \beta^{2^{m-1}+2^0} & \beta^{2^{m-1}+2^1} & \cdots & \beta^{2^{m-1}+2^{m-1}} \end{bmatrix} \tag{5}$$

$\boldsymbol{M}$의 각 성분은 정규기저 N를 사용해서 다시 표현될 수 있으므로 아래와 같이 전개될 수 있다[1].

$$\boldsymbol{M} = \boldsymbol{M}_0\beta + \boldsymbol{M}_1\beta^2 + \cdots + \boldsymbol{M}_{m-1}\beta^{2^{m-1}}, \tag{6}$$

여기서 $\boldsymbol{M}_i$는 각 성분이 $GF(2)$에 속하는 $m \times m$ 행렬이다.

(6)식을 (4)식에 대입하면, C의 각 계수 c_i는 아래와 같이 얻어진다[1].

$$c_i = \underline{a} \times \boldsymbol{M}_i \times \underline{b}^T = \underline{a}^{(i)} \times \boldsymbol{M}_0 \times \underline{b}^{(i)T}, \quad 0 \le i \le m-1, \tag{7}$$

여기서 $\underline{a}^{(i)}=[a_i, a_{i+1}, \cdots, a_{i-1}]$와 $\underline{b}^{(i)}=[b_i, b_{i+1}, \cdots, b_{i-1}]$은 각각 $\underline{a}$와 $\underline{b}$의 i-fold left cyclic shift이다.

각 $\boldsymbol{M}_i$에서 1의 개수는 모두 동일하며, 이 1의 개수를 C_N으로 보통 표시한다. $\boldsymbol{M}_i$에서 1인 요소의 개수가 정규기저 곱셈기의 게이트 수를 결정하기 때문에, C_N을 정규기저의 복잡도라 한다[9].

C_N은 $C_N \ge 2m-1$이라고 증명되어 있으며, $C_N=2m-1$일 때의 정규기저를 최적정규기저(Optimal Normal Basis:ONB)라고 하며 이 최적정규기저에는 두 종류, 즉 type-I과 type-II가 존재한다[9]. 현재 이러한 최적정규기저를 이용한 곱셈 및 곱셈 역원 연산을 효율적으로 구현하는 알고리즘 및 아키텍처에 대한 연구가 많이 진행되고 있다[1-3,5-8].

2.2 All-One Polynomial에 의해 정의된 $GF(2^m)$ 상의 원소 표현

$GF(2)$ 상에서 다항식 $P(x)=p_0+p_1x+p_2x^2+\cdots+p_{m-1}x^{m-1}+p_mx^m$의 모든 계수가 1이면, 즉 $i=0, 1, 2, \cdots, m-1, m$에 대해서 $p_i=1$이면, 다항식 $P(x)$를 차수 m의 All-One Polynomial(AOP)이라 한다. AOP가 기약 다항식일 필요충분조건은 $m+1$이 소수이고 2가 법 $m+1$에 관하여 원시근이어야 한다. 100이하인 정수 m에 대하여, 차수 m의 AOP가 기약이 되는 m의 값은 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82, 100이다[2,3].

$P(x)=1+x+x^2+\cdots+x^{m-1}+x^m$을 차수 m의 기약 AOP라 하고 β를 $P(x)$의 근이라 하자. 그러면 $1+\beta+\beta^2+\cdots+\beta^{m-1}+\beta^m=0$ 이므로,

$$1=\beta+\beta^2+\cdots+\beta^{m-1}+\beta^m$$

이고, 또한 $\beta^m=1+\beta+\beta^2+\cdots+\beta^{m-1}$ 식의 양변에 β를 곱하면

$$\beta^{m+1}=\beta+\beta^2+\cdots+\beta^{m-1}+\beta^m$$
$$\therefore \beta^{m+1}=1 \quad (8)$$

이 된다. 그리고 계속해서 위 (8)식의 양변에 β를 곱해 가면,

$$\begin{aligned}\beta^{m+1}&=1\\ \beta^{m+2}&=\beta\\ \beta^{m+3}&=\beta^2\\ &\cdots\\ \beta^{m+i}&=\beta^{i-1}\end{aligned}$$

이 되므로, β의 지수부분은 mod $m+1$이 된다.

차수 m의 기약 AOP의 한 근을 β라고 하면, $\beta^{2^i}(i=1,\cdots,m-1)$도 또한 근이 된다. 그리고 차수 m의 기약 AOP에 의해 정의된 $GF(2^m)$의 정규기저는 이 기약 AOP의 근들의 집합인, $N=\{\beta,\beta^2,\cdots,\beta^{2^{m-1}}\}$에 의해 형성될 수 있으며, 이 정규기저는 type-I ONB이다[1].

기약 AOP의 차수 m에 대해서, 2는 법 $m+1$에 관해서 원시근이므로 $\{2^0,2^1,\cdots,2^{m-1}\}$은 법 $m+1$에 관해 $\{1,2,\cdots,m\}$과 동일하다. 따라서 기약 AOP에 의해 정의된 정규기저 $N=\{\beta,\beta^2,\beta^{2^2},\cdots,\beta^{2^{m-1}}\}$은

$$N=\{\beta,\beta^2,\beta^3,\cdots,\beta^m\} \quad (9)$$

과 동일하다. 집합 (9)는 또한 기저이므로 $GF(2^m)$의 임의의 원소를 표현할 수 있다. 그래서 정규기저 N로 표현된 $GF(2^m)$의 임의의 원소 $A=\sum_{i=0}^{m-1}a_i\beta^{2^i}$는 아래와 같이 주어진 치환 P를 사용해서

$$a'_{2^i \bmod m+1}=a_i,\quad i=0,1,\cdots,m-1 \quad (10)$$

다음과 같이 기저 N의 표현으로 쉽게 변환된다.

$$A=\sum_{i=0}^{m-1}a_i\beta^{2^i}=\sum_{i=1}^{m}a'_i\beta^i \quad (11)$$

다음 장에서 (9)의 기저를 이용한 새로운 low-complexity bit-parallel 곱셈기를 구축하는 알고리즘에 대해서 설명한다.

3. 새로운 Low-Complexity Bit-Parallel 정규기저 곱셈기

3.1 새로운 곱셈 알고리즘

기약 AOP에 의해 정의된 $GF(2^m)$ 상의 임의의 원소는 정규기저 N 대신에 (10)식의 치환 P를 사용하여 (9)의 N' 기저 표현으로 쉽게 변환될 수 있다. 본 절에서는 새로운 기저 N'를 이용하여 기약 AOP에 의해 정의된 유한체 $GF(2^m)$ 상에서 정규기저로 표현된 임의의 두 원소에 대해서 곱셈을 계산하는 새로운 알고리즘에 대해서 설명한다. 이후로 $GF(2^m)$은 기약 AOP에 의해 정의된 유한체이다.

정규기저 N으로 표현된 $GF(2^m)$의 임의의 원소 A와 B의 곱셈, 즉 $C=AB$를 수행하기 위해, 먼저 (10)식의 치환 P를 사용하여 A와 B를 아래와 같이 기저 N'의 표현으로 변환한 후

$$A'=(a'_1,a'_2,\cdots,a'_m)=\sum_{i=1}^{m}a'_i\beta^i=a'_1\beta+a'_2\beta^2+\cdots+a'_m\beta^m \quad (12)$$

$$B'=(b'_1,b'_2,\cdots,b'_m)=\sum_{i=1}^{m}b'_i\beta^i=b'_1\beta+b'_2\beta^2+\cdots+b'_m\beta^m \quad (13)$$

다음과 같이 곱셈을 수행한다.

$$C'=A'B'=(\underline{a'}\times\underline{\beta}^T)\times(\underline{\beta}\times\underline{b'}^T)=\underline{a'}\times\boldsymbol{M'}\times\underline{b'}^T \quad (14)$$

여기서 $\underline{a'}=[a'_1,a'_2,\cdots,a'_m]$, $\underline{b'}=[b'_1,b'_2,\cdots,b'_m]$, $\underline{\beta}=[\beta,\beta^2,\cdots,\beta^m]$이고, $\boldsymbol{M'}$은 아래와 같이 정의된다.

$$\boldsymbol{M'}=\underline{\beta}^T\times\underline{\beta}=[\beta^{i+j}],\quad 1\le i,j\le m$$
$$=\begin{bmatrix}\beta^{1+1} & \beta^{1+2} & \cdots & \beta^{1+m}\\ \beta^{2+1} & \beta^{2+2} & \cdots & \beta^{2+m}\\ \cdot & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot\\ \cdot & \cdot & \cdot & \cdot\\ \beta^{m+1} & \beta^{m+2} & \cdots & \beta^{m+m}\end{bmatrix} \quad (15)$$

(15)식에서 β는 기약 AOP의 근이므로 β의 지수부분은 mod $m+1$이다. 그래서 $\boldsymbol{M'}$의 모든 성분의 지수부분에 mod $m+1$을 취하면 아래와 같이 다시 쓸 수 있다.

$$\boldsymbol{M'}=\begin{bmatrix}\beta^2 & \beta^3 & \beta^4 & \beta^5 & \cdots & \beta^{m-1} & \beta^m & 1\\ \beta^3 & \beta^4 & \beta^5 & \iddots & \iddots & \beta^m & 1 & \beta\\ \beta^4 & \beta^5 & \iddots & \iddots & \iddots & 1 & \beta & \beta^2\\ \beta^5 & \iddots & \iddots & \iddots & \iddots & \beta & \beta^2 & \beta^3\\ \vdots & \iddots & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots\\ \beta^{m-1} & \beta^m & \iddots & \iddots & \iddots & \iddots & \iddots & \vdots\\ \beta^m & 1 & \beta & \beta^2 & \beta^3 & \iddots & \iddots & \beta^{m-2}\\ 1 & \beta & \beta^2 & \beta^3 & \cdots & \cdots & \beta^{m-2} & \beta^{m-1}\end{bmatrix} \quad (16)$$

따라서 위 (16)식으로 변환된 M'을 (14)식에 대입하여 정리하면, C'는 아래와 같이 주어진다.

$$C' = \sum_{i=1}^{m} a'_i b'_{m+1-i} \cdot 1 + \sum_{i=1}^{m} \left(\sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j} \right) \beta^i \quad (17)$$

(17)식의 $\sum_{i=1}^{m} a'_i b'_{m+1-i} \cdot 1$에서 1은 $1 = \sum_{i=1}^{m} \beta^i$이다. 그래서 $\sum_{i=1}^{m} a'_i b'_{m+1-i}$을 δ로 표시하면 C'는 아래와 같이 다시 쓸 수 있다.

$$C' = \sum_{i=1}^{m} \left(\delta + \sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j} \right) \beta^i \quad (18)$$

(18)식에서 $\sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j} = U_i$라고 하면 C'은 아래와 같이 또다시 쓸 수 있다.

$$C' = \sum_{i=1}^{m} (\delta + U_i) \beta^i = (\delta + U_1)\beta + (\delta + U_2)\beta^2 + \cdots + (\delta + U_m)\beta^m \quad (19)$$

따라서 위 (19) 식에서 구한 C'에 대해서 각 좌표 c'_i를 나타내면 아래와 같이 된다.

$$c'_i = \delta + \sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j} = \delta + U_i, \quad 1 \le i \le m$$

(19)식과 같이 주어진 C'에 대해 P의 역치환을 수행하면 원래의 정규기저 N으로 표현된 $C=(c_0, c_1, \cdots, c_{m-1})$를 얻게 된다.

3.2 아키텍처

3.1절의 곱셈 알고리즘을 통해 얻어진 Bit-Parallel 정규기저 곱셈기의 구조를 그림 1에 나타낸다. 이 구조에서 P는 (10)식에 나타낸 치환을 나타낸다. 즉, 이 치환을 통해 정규기저 N으로 표현된 입력 A와 B는 (9)의 기저 N' 표현으로 변환된다. 그리고 각 $U_i(i=1, 2, \cdots, m)$는 $\sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j}$을 생성하고, δ는 $\sum_{i=1}^{m} a'_i b'_{m+1-i}$을 생성한다. (19)식에 의해 각 U_i의 출력 값은 δ의 출력 값과 더해진 다음, P의 역치환 P^{-1} 모듈에 입력된다. 이 모듈을 통해 원래의 정규기저 표현으로 변환되게 된다.

그림 1에서 모든 U_i는 Sum-Of-Product(곱항들의 합) 표현을 구현하는 것이므로 AND-XOR 구조로 구현되어지며, 또한 곱항의 개수가 모두 $m-1$로 같으므로 그림 2와 같은 동일한 구조를 가진다. 즉, 단지 입력과 출력 값만 다를뿐 모든 부분이 동일하다. 그리고 δ 또한 U_i처럼 Sum-Of-Product 표현을 구현하는 것이며, 곱항의 개수가 하나더 많은 m이라는 것만 제외하고는 그림 2와 거의 동일한 그림 3과 같은 구조를 가진다.

그림 2와 3에서 XOR Plane의 구조는 2-입력 XOR

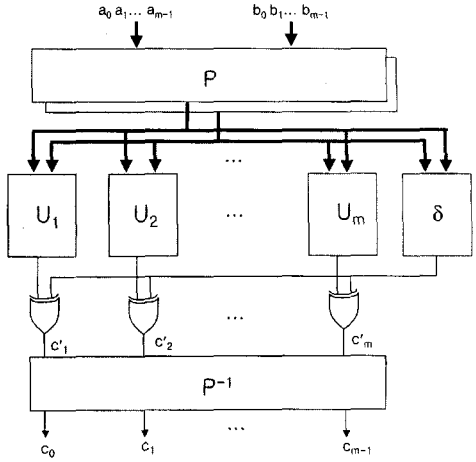

그림 1 제안한 Bit-Parallel 정규기저 곱셈기

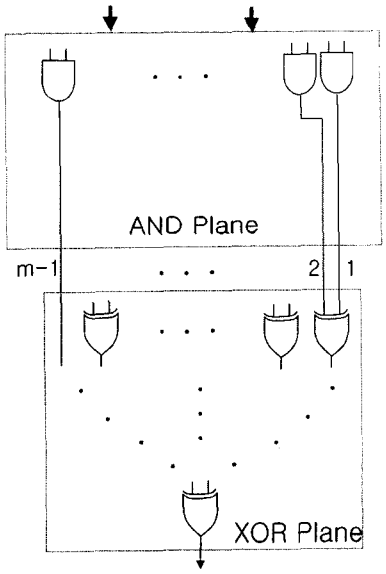

그림 2 U_i모듈의 구조

게이트만 사용한다고 가정하면 [1]에서 보이는 바와 같이 XOR 게이트의 2진 트리형태로 보통 구현된다

일반적으로 어떤 회로의 아키텍처가 간단하고, 가능한 한 동일한 모듈들로 구성되어 있으면 그 아키텍처가 규칙적이다고 하며, 이러한 아키텍처를 갖는 회로는 VLSI 구현에 적합하다. 따라서 그림 1의 곱셈기는 단순히 wiring에 의해 구현되는 P 및 P^{-1} 모듈, m개의 동일한 U_i 모듈과 이 모듈과 거의 유사한 δ 모듈 한 개, 그리고 추가적으로 m개의 XOR 게이트로 구성되어 있으며, 이들 모듈들이 간단하게 연결되어 있듯이, 본 곱셈기는 아키텍처가 규칙적이어서 VLSI 구현에 적합하다.

3.3 제안한 곱셈기의 예

본 절에서는 $m=10$인 경우에, 본 논문에서 제안하는 곱셈기를 구현하는 절차에 대해서 설명한다. $m=10$이면 $m+1$은 11인 소수이며 2는 법 11에 관하여 원시근이다. 그래서 AOP

$$P(x)=1+x+x^2+\cdots+x^9+x^{10} \tag{20}$$

은 기약다항식이 된다. 따라서 위 기약 AOP에 정의된 유한체 $GF(2^m)$의 정규기저는

$$N=\{\beta,\beta^2,\beta^{2^2},\cdots,\beta^{2^8},\beta^{2^9}\} \tag{21}$$

여기서 β는 기약 AOP (20)의 근이다.

이 되며 이 정규기저는 type-I ONB이다. 그리고 $\beta^{11}=1$이고 2는 법 11에 대해서 원시근이므로, 정규기저 N의 각 기저 원소는 아래와 같이 된다.

$$\begin{aligned}
\beta^{2^0}&=\beta^{1 \bmod 11}=\beta,\\
\beta^{2^1}&=\beta^{2 \bmod 11}=\beta^2,\\
\beta^{2^2}&=\beta^{4 \bmod 11}=\beta^4,\\
\beta^{2^3}&=\beta^{8 \bmod 11}=\beta^8,\\
\beta^{2^4}&=\beta^{16 \bmod 11}=\beta^5,\\
\beta^{2^5}&=\beta^{32 \bmod 11}=\beta^{10},\\
\beta^{2^6}&=\beta^{64 \bmod 11}=\beta^9,\\
\beta^{2^7}&=\beta^{128 \bmod 11}=\beta^7,\\
\beta^{2^8}&=\beta^{256 \bmod 11}=\beta^3,\\
\beta^{2^9}&=\beta^{512 \bmod 11}=\beta^6.
\end{aligned}$$

따라서 (21)의 정규기저 N은 새로운 기저

$$N=\{\beta,\beta^2,\beta^3,\cdots,\beta^9,\beta^{10}\} \tag{22}$$

과 동일하다. 그래서 아래와 같이 (21)의 정규기저 N으로 표현된

$$A=(a_0,a_1,\cdots,a_8,a_9)$$

$$=\sum_{i=0}^{9}a_i\beta^{2^i}=a_0\beta+a_1\beta^2+a_2\beta^{2^2}+\cdots+a_8\beta^{2^8}+a_9\beta^{2^9}$$

는 아래의 치환식

$$a'_{2^i \bmod 11}=a_i,\qquad i=0,1,\cdots,9 \tag{23}$$

을 이용해서 아래와 같이 (22)의 기저 N의 표현으로 변환시킬 수 있다.

$$\begin{aligned}
A'&=(a'_1,a'_2,\cdots,a'_9,a'_{10})\\
&=\sum_{i=1}^{10}a'_i\beta^i=a'_1\beta+a'_2\beta^2+\cdots+a'_9\beta^9+a'_{10}\beta^{10}
\end{aligned} \tag{24}$$

여기서, 각 a'_i는 아래와 같다.

$$a'_1=a_0,\quad a'_2=a_1,\quad a'_3=a_8,\quad a'_4=a_2,\quad a'_5=a_4,$$
$$a'_6=a_9,\quad a'_7=a_7,\quad a'_8=a_3,\quad a'_9=a_6,\quad a'_{10}=a_5$$

마찬가지로 $GF(2^{10})$의 임의의 원소 B도 아래와 같이 기저 N의 표현으로 변환될 수 있다.

$$\begin{aligned}
B'&=(b'_1,b'_2,\cdots,b'_9,b'_{10})\\
&=\sum_{i=1}^{10}b'_i\beta^i=b'_1\beta+b'_2\beta^2+\cdots+b'_9\beta^9+b'_{10}\beta^{10}
\end{aligned} \tag{25}$$

이제 위와 같이 (22)의 기저 N의 표현으로 변환된 A'와 B'에 대해서 곱셈 연산을 수행해 보자. A'와 B'의 곱을 C'이라고 하면 C'은 아래와 같이 계산된다.

$$C'=A'B'=(\underline{a'}\times\underline{\beta}^T)\times(\underline{\beta}\times\underline{b'}^T)=\underline{a'}\times\boldsymbol{M'}\times\underline{b'}^T \tag{26}$$

$$=\underline{a'}\times\begin{bmatrix}
\beta^2&\beta^3&\beta^4&\beta^5&\beta^6&\beta^7&\beta^8&\beta^9&\beta^{10}&1\\
\beta^3&\beta^4&\beta^5&\beta^6&\beta^7&\beta^8&\beta^9&\beta^{10}&1&\beta\\
\beta^4&\beta^5&\beta^6&\beta^7&\beta^8&\beta^9&\beta^{10}&1&\beta&\beta^2\\
\beta^5&\beta^6&\beta^7&\beta^8&\beta^9&\beta^{10}&1&\beta&\beta^2&\beta^3\\
\beta^6&\beta^7&\beta^8&\beta^9&\beta^{10}&1&\beta&\beta^2&\beta^3&\beta^4\\
\beta^7&\beta^8&\beta^9&\beta^{10}&1&\beta&\beta^2&\beta^3&\beta^4&\beta^5\\
\beta^8&\beta^9&\beta^{10}&1&\beta&\beta^2&\beta^3&\beta^4&\beta^5&\beta^6\\
\beta^9&\beta^{10}&1&\beta&\beta^2&\beta^3&\beta^4&\beta^5&\beta^6&\beta^7\\
\beta^{10}&1&\beta&\beta^2&\beta^3&\beta^4&\beta^5&\beta^6&\beta^7&\beta^8\\
1&\beta&\beta^2&\beta^3&\beta^4&\beta^5&\beta^6&\beta^7&\beta^8&\beta^9
\end{bmatrix}\times\underline{b'}^T \tag{27}$$

(27)식의 행렬을 계산하여 β^2를 포함하는 곱항만 정리하면 아래와 같이 된다.

$$\begin{aligned}
&a'_3b'_{10}\beta^2+a'_4b'_9\beta^2+a'_5b'_8\beta^2\\
&\quad+a'_6b'_7\beta^2+a'_7b'_6\beta^2+a'_8b'_5\beta^2\\
&\quad+a'_9b'_4\beta^2+a'_{10}b'_3\beta^2+a'_1b'_1\beta^2\\
&\quad=\left(\sum_{j=1}^{10-2}a'_{2+j}b'_{11-j}+\sum_{j=1}^{2-1}a'_jb'_{2-j}\right)\beta^2
\end{aligned} \tag{28}$$

(28)과 같이 각 β^i에 대해서, (27)식의 행렬을 계산하여 정리하면 C'는 아래와 같이 된다.

$$\begin{aligned}
C'&=\sum_{i=1}^{10}a'_ib'_{11-i}\cdot 1\\
&\quad+\sum_{i=1}^{10}\left(\sum_{j=1}^{10-i}a'_{i+j}b'_{11-j}+\sum_{j=1}^{i-1}a'_jb'_{i-j}\right)\beta^i
\end{aligned} \tag{29}$$

(29)식의 $\sum_{i=1}^{10}a'_ib'_{11-i}\cdot 1$에서 1은

$$1=\sum_{i=1}^{10}\beta^i=\beta+\beta^2+\beta^3+\beta^4+\beta^5+\beta^6+\beta^7+\beta^8+\beta^9+\beta^{10}$$

이므로, (29)식에서 $\sum_{i=1}^{10}a'_ib'_{11-i}$를 δ라고 하면 C'은 아래와 같이 다시 쓸 수 있다.

$$C'=\sum_{i=1}^{10}\left(\delta+\sum_{j=1}^{10-i}a'_{i+j}b'_{11-j}+\sum_{j=1}^{i-1}a'_jb'_{i-j}\right)\beta^i \tag{30}$$

또한, 위 (30)식에서 $\sum_{j=1}^{10-i}a'_{i+j}b'_{11-j}+\sum_{j=1}^{i-1}a'_jb'_{i-j}=U_i$라고 하면 C'은 아래와 같이 된다.

$$\begin{aligned}
C'&=\sum_{i=1}^{10}(\delta+U_i)\beta^i\\
&=(\delta+U_1)\beta+(\delta+U_2)\beta^2+\cdots+(\delta+U_{10})\beta^{10}
\end{aligned}$$

위와 같이 계산된 $C'=(c'_1,c'_2,\cdots,c'_{10})$은 아래의 역치환을 통해 (21)의 정규기저 N의 표현으로 변환된 $C=A\times B=(c_0,c_1,\cdots,c_9)$를 얻게 된다.

$c_0 = c'_1$, $c_1 = c'_2$, $c_2 = c'_4$, $c_3 = c'_8$, $c_4 = c'_5$,
$c_5 = c'_{10}$, $c_6 = c'_9$, $c_7 = c'_7$, $c_8 = c'_3$, $c_9 = c'_6$.

위와 같은 절차에 의해 얻어지는 $m=10$인 경우의 곱셈기의 아키텍처를 부록에 나타낸다. 부록에 있는 그림에서 볼 수 있듯이 모든 $U_i(i=1,\cdots,10)$ 모듈은 구조가 동일하여, δ 모듈 또한 U_i 모듈과 거의 동일함을 알 수 있다. 그래서 예로 든 본 절의 곱셈기에서도 알 수 있듯이 본 논문의 곱셈기는 그 아키텍처가 규칙적임을 알 수 있다.

4. 게이트 및 시간 복잡도

그림 1의 구조에서 치환 P와 역치환 P^{-1}은 단순히 wiring에 의해 구현되므로 어떠한 게이트 자원도 필요로 하지 않는다. 각 U_i 모듈은

$$\sum_{j=1}^{m-i} a'_{i+j} b'_{m+1-j} + \sum_{j=1}^{i-1} a'_j b'_{i-j} \qquad (31)$$

을 구현하는 것이므로, 이 식에서 곱셈과 덧셈의 개수가 바로 각 U_i의 AND 게이트와 XOR 게이트의 개수이다. 먼저 이 식에서 각 U_i의 곱셈의 개수는 $(m-i)+(i-1)$, 즉 $m-1$이며, 덧셈의 개수는 $m-2$이다. 그리고 δ 블록은 $\sum_{i=1}^{m} a'_i b'_{m+1-i}$을 구현하므로 곱셈 및 덧셈의 개수는 각각 m 및 $m-1$이다. 따라서 그림 1 곱셈기의 전체 AND 게이트 및 XOR 게이트의 개수는 아래와 같다.

- 전체 AND 게이트 개수
  = (각 U_i의 AND 게이트 개수) × (U_i의 개수)
  + (δ 의 AND 게이트 개수)
  = $(m-1)\times m + m$
  = m^2
- 전체 XOR 게이트 개수
  = (각 U_i의 XOR 게이트 개수) × (U_i의 개수)
  + (δ 의 XOR 게이트 개수) + m
  = $(m-2)m + m - 1 + m$
  = $m^2 - 1$

그 다음으로 그림 1의 곱셈기의 시간 지연(time delay)을 조사해 보자. 그림 1의 곱셈기에서 가장 긴 입력에서 출력까지 경로는 아래와 같다.

- 입력에서 출력까지의 경로
  = $P \rightarrow \delta \rightarrow$ XOR 게이트 $\rightarrow P^{-1}$

위 경로에서 P와 P^{-1}은 wiring에 의해 구현되므로 시간 지연에 포함되지 않는다. δ 모듈은 그림 3와 같은 구조로 구현된다. 그래서 δ 의 시간 지연은

$$T_A + \lceil \log_2 m \rceil T_X,$$

여기서 T_A= AND 게이트의 시간 지연,
T_X= XOR 게이트의 시간 지연

이다. 그리고 m이 짝수이면 $\lceil \log_2 m \rceil = \lceil \log_2 (m-1) \rceil$ 이다. 본 논문에서 m은 항상 짝수이므로, δ 블록의 시간 지연은

$$T_A + \lceil \log_2 (m-1) \rceil T_X$$

이 된다. 따라서 그림 1의 곱셈기의 전체 시간 지연은 아래와 같다.

전체 시간 지연 = $T_A + (1 + \lceil \log_2 (m-1) \rceil) T_X$

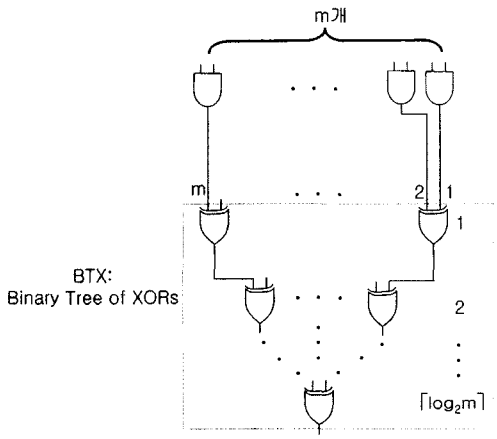

그림 3 δ 모듈 구조

위에서 계산된 그림 1의 병렬 곱셈기의 게이트 개수와 시간 지연에 대해서 기약 AOP에 의해 생성된 같은 종류의 다른 병렬 곱셈기와 비교한 결과를 표 1에 나타낸다. 표 1의 결과에 나타난 것처럼 본 논문의 곱셈기는 게이트 및 시간적인 면 모두에서 가장 효율적인 아키텍처에 속함을 알 수 있다.

5. 결 론

본 논문에서는 AOP에 의해 정의된 $GF(2^m)$ 상의 Bit-Parallel 정규기저 곱셈기를 제안하였다. 본 곱셈기의 게이트 및 시간 복잡도는 이전에 제안된 같은 종류의 곱셈기 보다 낮거나 동일하다. 표 1의 결과에서 볼 수 있듯이 지금까지 제안된 같은 종류의 곱셈기들 중에서 가장 효율적인 아키텍처에 속하며, 이러한 종류의 곱셈기를 구현하는데 있어 또 하나의 디자인이 될 수 있다. 그리고 그림 1에서 볼 수 있듯이 본 곱셈기의 아키텍처는 m개의 동일한 모듈 U_i와 이 모듈과 거의 동일한 δ 모듈 1개 그리고 추가적으로 m개의 XOR 게이트로 규칙적으로 구성되어 있어 VLSI 구현에 적합하다.

표 1 기약 AOP에 의해 정의된 $GF(2^m)$ 상의 Bit-Parallel 곱셈기의 비교

곱셈기	기저	#AND	#XOR	시간 지연
Itoh와 Tsujii[4]	Polynomial	$(m+1)^2$	m^2+2m	$T_A+(\lceil \log_2 m \rceil + \lceil \log_2(m+2) \rceil)T_X$
Hasan 등[5]	Polynomial	m^2	m^2+m-2	$T_A+(m+\lceil \log_2(m-1) \rceil)T_X$
Koc과 Sunar[2]	Polynomial	m^2	m^2-1	$T_A+(2+\lceil \log_2(m-1) \rceil)T_X$
Wu-Hasan[8]	Weakly dual	m^2	m^2-1	$T_A+(1+\lceil \log_2(m-1) \rceil)T_X$
Wang 등(MO)[7]	Normal	m^2	$2m^2-2m$	$T_A+(1+\lceil \log_2(m-1) \rceil)T_X$
Koc와 Sunar[2]	Normal	m^2	m^2-1	$T_A+(2+\lceil \log_2(m-1) \rceil)T_X$
Hasan 등[6]	Normal	m^2	m^2-1	$T_A+(1+\lceil \log_2(m-1) \rceil)T_X$
Reyhani-Masoleh 등[1]	Normal	m^2	m^2-1	$T_A+(1+\lceil \log_2(m-1) \rceil)T_X$
본 논문	Normal	m^2	m^2-1	$T_A+(1+\lceil \log_2(m-1) \rceil)T_X$

참 고 문 헌

[1] A. Reyhani-Masoleh and M.A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$," *IEEE Trans. Computers*, vol. 51, no. 5, pp. 511~520, May 2002.

[2] C.K. Koc and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finites Fields," *IEEE Trans. Computers*, vol. 47, no. 3, pp. 353~356, Mar. 1998.

[3] C.Y. Lee, E.H. Lu, and J.Y. Lee, "Bit-Parallel Systolic Multipliers for $GF(2^m)$ Fields Defined by All-One and Equally Spaced Polynomials," *IEEE Trans. Computers*, vol. 50, no. 5, pp. 385~393, May 2001.

[4] T. Itoh and S. Tsujii, "Structure of Parallel Multiplier for a Class of Fields $GF(2^m)$," *Information and Computation*, vol. 83, pp. 21~40, 1989.

[5] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields $GF(2^m)$," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 962~971, Aug. 1992.

[6] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1278~1280, Oct. 1993.

[7] C.C. Wang, T.K. Truong, H.M. Shar, L.J. Deutsch, J.K. Omura, and I.S. Reed, "VLSI Architecture for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE Trans. Computers*, vol. 34, no. 8, pp. 709~716, Aug. 1985.

[8] H. Wu and M.A. Hasan, "Low Complexity Bit-Parallel Multipliers for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 47, no. 8, pp. 883~887, Aug. 1998.

[9] R.C. Mullin, I.M. Onyszchuk, S. A. Vanstone, and R.M. Wilson, "Optimal Normal Bases in $GF(p^n)$," *Discrete Applied Mathematics*, vol. 22, pp. 149~161, 1988/89.

〈부 록〉

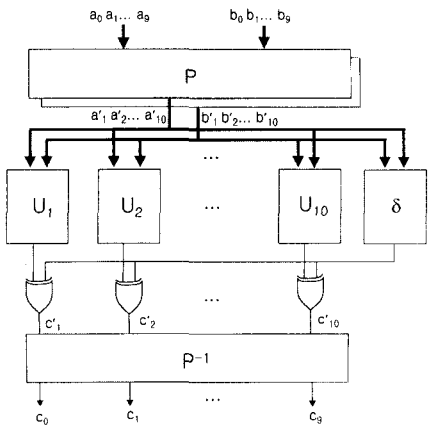

그림 4 $m=10$인 경우의 곱셈기

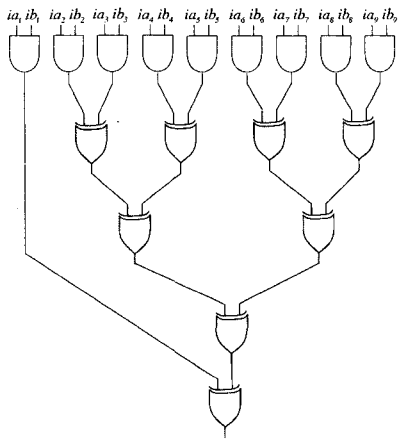

그림 5 그림 4의 U_1-U_{10} 모듈(각 모듈의 AND 게이트 입력 신호는 표 2에)

표 2 각 U_i 모듈의 AND 게이트 입력 신호

모듈 \ 입력	ia_1	ib_1	ia_2	ib_2	ia_3	ib_3	ia_4	ib_4	ia_5	ib_5	ia_6	ib_6	ia_7	ib_7	ia_8	ib_8	ia_9	ib_9
U_1	a'_2	b'_{10}	a'_3	b'_9	a'_4	b'_8	a'_5	b'_7	a'_6	b'_6	a'_7	b'_5	a'_8	b'_4	a'_9	b'_3	a'_{10}	b'_2
U_2	a'_3	b'_{10}	a'_4	b'_9	a'_5	b'_8	a'_6	b'_7	a'_7	b'_6	a'_8	b'_5	a'_9	b'_4	a'_{10}	b'_3	a'_1	b'_1
U_3	a'_4	b'_{10}	a'_5	b'_9	a'_6	b'_8	a'_7	b'_7	a'_8	b'_6	a'_9	b'_5	a'_{10}	b'_4	a'_1	b'_2	a'_2	b'_1
U_4	a'_5	b'_{10}	a'_6	b'_9	a'_7	b'_8	a'_8	b'_7	a'_9	b'_6	a'_{10}	b'_5	a'_1	b'_3	a'_2	b'_2	a'_3	b'_1
U_5	a'_6	b'_{10}	a'_7	b'_9	a'_8	b'_8	a'_9	b'_7	a'_{10}	b'_6	a'_1	b'_4	a'_2	b'_3	a'_3	b'_2	a'_4	b'_1
U_6	a'_7	b'_{10}	a'_8	b'_9	a'_9	b'_8	a'_{10}	b'_7	a'_1	b'_5	a'_2	b'_4	a'_3	b'_3	a'_4	b'_2	a'_5	b'_1
U_7	a'_8	b'_{10}	a'_9	b'_9	a'_{10}	b'_8	a'_1	b'_6	a'_2	b'_5	a'_3	b'_4	a'_4	b'_3	a'_5	b'_2	a'_6	b'_1
U_8	a'_9	b'_{10}	a'_{10}	b'_9	a'_1	b'_7	a'_2	b'_6	a'_3	b'_5	a'_4	b'_4	a'_5	b'_3	a'_6	b'_2	a'_7	b'_1
U_9	a'_{10}	b'_{10}	a'_1	b'_8	a'_2	b'_7	a'_3	b'_6	a'_4	b'_5	a'_5	b'_4	a'_6	b'_3	a'_7	b'_2	a'_8	b'_1
U_{10}	a'_1	b'_9	a'_2	b'_8	a'_3	b'_7	a'_4	b'_6	a'_5	b'_5	a'_6	b'_4	a'_7	b'_3	a'_8	b'_2	a'_9	b'_1

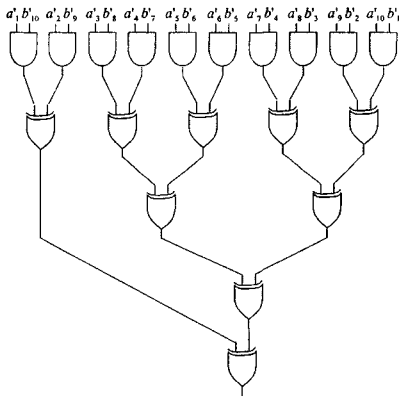

그림 6 그림 4의 δ 모듈

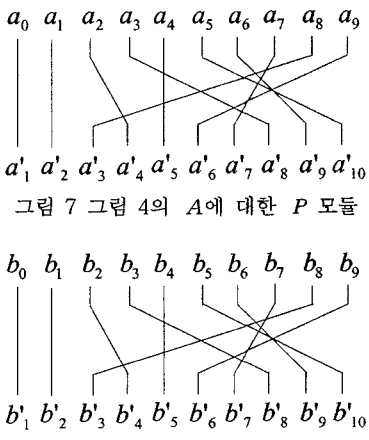

그림 7 그림 4의 A에 대한 P 모듈

그림 8 그림 4의 B에 대한 P 모듈

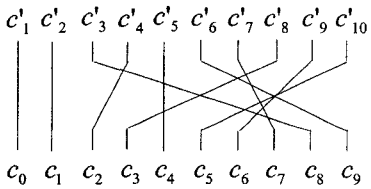

그림 9 그림 4의 P^{-1} 모듈

장 용 희

1996년 2월 한국항공대학교 항공통신정보공학과 졸업. 1998년 2월 한국항공대학교 정보통신공학과 대학원 졸업(공학석사). 1998년 3월~현재 한국항공대학교 정보통신공학과 대학원 박사과정 재학중. 관심분야는 논리회로 설계 및 합성, 정보보호, 암호이론

권 용 진

1986년 2월 한국항공대학교 항공전자공학과 졸업. 1990년 3월 일본쿄토대학 정보공학과 대학원 졸업(공학석사). 1994년 3월 일본쿄토대학 정보공학과 대학원 졸업(공학박사). 1994년 3월~현재 한국항공대학교 전자·정보통신·컴퓨터공학부 부교수. 관심분야는 정보보호, 논리설계 및 합성, 정보검색