情報保護學會論文誌
第19卷 第2號, 2009. 4

삼항 기약다항식을 위한 효율적인 Shifted Polynomial Basis 비트-병렬 곱셈기*

장 남 수,[1†] 김 창 한,[2‡] 홍 석 희,[1] 박 영 호[3]
[1]고려대학교, [2]세명대학교, [3]세종사이버대학교

Efficient Bit-Parallel Shifted Polynomial Basis Multipliers for All Irreducible Trinomial*

Nam Su Chang,[1†] Chang Han Kim,[2‡] Seokhie Hong,[1] Young-Ho Park[3]
[1]Korea University, [2]Semyung University, [3]Sejong Cyber University

요 약

유한체 연산중에서 곱셈 연산은 중요한 연산중 하나이다. 또한, 최근에 Fan과 Dai는 이진체 곱셈기의 효율성을 개선하기 위하여 Shifted Polynomial Basis(SPB)와 이를 이용한 non-pipeline 비트-병렬 곱셈기를 제안하였다. 본 논문에서는 삼항 기약다항식 x^n+x^k+1에 의하여 정의된 F_{2^n} 위에서의 새로운 SPB 곱셈기 type I과 type II를 제안한다. 제안하는 type I 곱셈기는 기존의 SPB 곱셈기에 비하여 시간 및 공간 복잡도면에서 모두 효율적이다. 그리고 type II 곱셈기는 제안하는 type I 곱셈기를 포함하여 기존의 모든 결과보다 작은 공간 복잡도를 가진다. 그러나 type II 곱셈기의 시간 복잡도는 n과 k에 따라 최대 1 XOR time-delay 증가한다.

ABSTRACT

Finite Field multiplication operation is one of the most important operations in the finite field arithmetic. Recently, Fan and Dai introduced a Shifted Polynomial Basis(SPB) and construct a non-pipeline bit-parallel multiplier for F_{2^n}. In this paper, we propose a new bit-parallel shifted polynomial basis type I and type II multipliers for F_{2^n} defined by an irreducible trinomial x^n+x^k+1. The proposed type I multiplier has more efficient the space and time complexity than the previous ones. And, proposed type II multiplier have a smaller space complexity than all previously SPB multiplier(include our type I multiplier). However, the time complexity of proposed type II is increased by 1 XOR time-delay in the worst case.

Keywords: Shifted Polynomial Basis, Irreducible Trinomial, Mastrovito Multiplier, Bit-Parallel Multiplier

I. 서 론

유한체 연산은 부호이론, 암호알고리즘 등에서 매우 중요한 요소로 사용된다. 이러한 유한체 연산 중 곱셈 연산이 가장 중요한 비중을 차지하며 곱셈 연산의 효율성은 기저, 알고리즘, 기약다항식 등에 의하여 좌우된다. 효율적인 기약다항식을 선택하는 경우 All One Irreducible Polynomial(AOP)를 제외하고 대부분의 경우는 기약 다항식의 0이 아닌 항의 개수가 최소가 되는 경우가 가장 효율적이다. 따라서 일반적인 확장체에서는 이항 기약다항식(irreducible binomial)이 가장 효율적이지만 본 논문과 같이 이진체 환경에서는 이항 기약다항식이 존재하지 않으므로 삼항 기약다항식이 가장 효율적이다.

확장체 연산은 소수체 연산의 덧셈(뺄셈) 연산과 곱셈 연산에 의하여 계산된다. 본 논문에서는 이진체에

접수일(2008년 12월 23일), 게재확정일(2009년 3월 9일)
* 이 연구에 참여한 연구자(의 일부)는 '2단계 BK21사업'의 지원비를 받았음.
† 주저자, ns-chang@korea.ac.kr
‡ 교신저자, chkim@semyung.ac.kr

서의 곱셈 연산에 대하여 논하므로 F_2에서의 덧셈과 곱셈 연산은 비트 단위의 AND와 XOR 연산에 의하여 표현된다. 이는 하드웨어에서 2-입력 1-출력의 AND와 XOR 게이트(gate)이다. 또한 하드웨어에서 연산의 효율성은 시간 및 공간 복잡도에 의하여 정의되므로 본 논문에서는 공간 복잡도 계산을 위하여 AND와 XOR 게이트 수를 각각 #AND와 #XOR로 표기하고 시간 복잡도를 위하여 AND와 XOR 게이트 시간 지연(time delay)을 각각 T_A와 T_X로 표기한다.

기존에는 Polynomial Basis(PB)를 사용한 곱셈기가 대부분이었다[1-4]. Shifted Polynomial Basis(SPB)는 [5]에서 처음 제안했으며 이를 이용하여 파이프라인(pipeline)이 아닌 비트-병렬 곱셈기를 제안하였다. SPB는 PB를 변형한 형태로 둘 사이의 기저 변환이 매우 간단하게 수행된다. 따라서 기존의 SPB 비트-병렬 곱셈기[5-8]는 기존의 PB 비트-병렬 곱셈기와 많이 비교된다. [5]의 SPB 삼항 기약다항식을 위한 비트-병렬 곱셈기는 기존 PB 기반 보다 시간 복잡도면에서 효율적이며 공간 복잡도는 동일하다. 반면 [6]에서는 [5]의 곱셈기보다 공간 복잡도는 25%증가하지만 시간복잡도면에서 같거나 $1T_X$ 작은 곱셈기를 제안하였다. 마지막으로 [8]에서 [5]의 결과를 개선하여 공간 복잡도 증가 없이 [6]와 동일한 시간 복잡도를 가지는 곱셈기를 제안하였다.

본 논문에서는 Modified Shifted Polynomial Basis(MSPB)를 제안하고 이를 이용한 새로운 SPB 비트-병렬 곱셈 과 하드웨어 구조를 제안한다. 제안하는 곱셈기는 MSPB로의 기저 변환과 병렬 곱셈으로 구성된 두 과정이 병렬로 수행되며 입/출력은 기존 SPB에서 최적이라고 제안된 기저 $\alpha^{-k}\Gamma=\{\alpha^{i-k} \mid 0 \le i \le n-1\}$를 가정한다. 제안하는 비트-병렬 곱셈기는 두 가지로 type I 곱셈기는 시간 복잡도가 최소가 되도록 type II 곱셈기는 공간 복잡도가 최소가 되도록 구성하였다. 따라서 type I 곱셈기는 기존 [8]의 곱셈기에 비하여 시간 및 공간 복잡도 모두에서 효율적이고 type II 곱셈기는 공간 복잡도는 type I에 비하여 감소하나 n과 k에 따라 시간 복잡도의 효율성은 다양하게 나타난다. $100 \le n < 1{,}000$에서 $n \le 2k$를 만족하는 모든 삼항 기약다항식(1,335개)으로 시간 복잡도를 조사한 결과 type I의 경우 174개(13%)가 [8]의 결과보다 작고 1,161개(87%)가 같았으며, type II의 경우 33개(2.5%)가 [8]의 결과보다 작고 1,188개(89%)가 같았으며 114(8.5 %)가 기존 결과보다 $1T_X$ 증가하였다. 반면 AND 게이트와 XOR 게이트의 비용을 같다고 가정할 경우 공간 복잡도면에서는 [8]에 비하여 type I의 경우 최대 12.5%, type II의 경우 최대 25%의 게이트가 감소한다.

본 논문의 구성은 다음과 같다. 2절에서는 [5]에서 제안된 SPB와 SPB 비트-병렬 곱셈을 소개하고 이를 개선한 [8]의 삼항 기약다항식을 위한 SPB 비트-병렬 곱셈과 복잡도를 소개한다. 3절에서는 제안하는 MSPB와 이를 이용한 새로운 SPB 비트-병렬 곱셈과 하드웨어 구조 그리고 시간 및 공간 복잡도를 기술한다. 4절에서는 제안하는 방법의 결과와 기존 결과를 비교하고 결론을 내린다.

II. SPB 비트-병렬 곱셈

F_{2^n}는 w개의 항을 가지는 기약다항식 $f(x)=x^n+\sum_{i=0}^{w-2} x^{e_i}$에 의하여 생성되었고, $f(x)$의 근을 α라 정의하자. (이때 $0=e_0<e_1<\cdots<e_{w-2}<n$이다.) 그러면 F_{2^n}의 임의의 두 원소 $a(\alpha)$, $b(\alpha)$는

$$a(\alpha)=\sum_{i=0}^{n-1} a_i\alpha^i,\ b(\alpha)=\sum_{i=0}^{n-1} b_i\alpha^i$$

이며(이때 $a_i, b_i \in F_2$이다.), 두 원소의 곱 $c(\alpha)$ 는 다음과 같은 두 과정에 의하여 계산된다:

1. 다항식 곱셈: $s(\alpha)=a(\alpha)b(\alpha)=\sum_{i=0}^{2n-2} s_i\alpha^i$,
$$s_i = \sum_{\substack{u+v=i \\ 0 \le u,v \le n-1}} a_u b_v$$
2. $f(x)$에 의한 모듈러 감산 연산:
$$c(\alpha) \equiv s(\alpha) \bmod f(x) \equiv \sum_{i=0}^{n-1} c_i\alpha^i,\ c_i \in F_{2^n}.$$

따라서 곱셈 연산의 복잡도는 위의 두 과정에 의하여 결정되며, 모듈러 감산 연산 복잡도의 경우 기약다항식 $f(x)$의 항의 개수에 의하여 영향을 받는다.

본 논문에서는 삼항 기약다항식에 적합한 SPB 곱셈 방법을 제안하므로 이후 모든 기약다항식은 중간항이 k인 $f(x)=x^n+x^k+1$이다. 최근 [6]에서 F_{2^n}의 원소를 SPB로 표현하는 방법이 제안되었으며 이는 다음과 같이 정의된다.

정의 1. v를 임의의 정수라 하고 $\Gamma=\{\alpha^i \mid 0 \le i \le n-1\}$를 F_{2^n}의 다항식기저라 할 때, 순서집합 $\alpha^{-v}\Gamma=\{\alpha^{i-v} \mid 0 \le i \le n-1\}$를 Γ에 대한 Shifted

Polynomial Basis(SPB)라 한다.

정의 1에 의하여 SPB로 표현된 F_{2^n}의 임의의 두 원소 $a(\alpha)$, $b(\alpha)$는 $a(\alpha)=\alpha^{-v}\sum_{i=0}^{n-1}a_i\alpha^i$, $b(\alpha)=\alpha^{-v}\sum_{i=0}^{n-1}b_i\alpha^i$로 표현되며, 두 원소의 곱은 $c(\alpha)\equiv a(\alpha)b(\alpha)\equiv\sum_{i=0}^{n-1}c_i\alpha^{i-v}$이며, $c(\alpha)$는 다음과 같이 계산된다.

$$
\begin{aligned}
s(\alpha) &= a(\alpha)b(\alpha) \\
&= \alpha^{-2v}\sum_{i=0}^{2n-2}s_i\alpha^i = \sum_{i=-2v}^{2n-2-2v}s_{t+2v}\alpha^i \\
&= r_- + r + r_+
\end{aligned}
$$

라 하면

$$
s_t = \sum_{\substack{i+j=t \\ 0\le i,j\le n-1}} a_ib_j \begin{cases} \sum_{m=0}^{t} a_{mb_{t-m}} & ,0\le t\le n-1 \\ \sum_{m=t+1-n}^{n-1} a_mb_{t-m} & ,n\le t\le 2n-2 \end{cases}
$$

이다. 이때 $s(\alpha)$는 $-2v$부터 $2n-2-2v$까지 항을 가지므로 $-2v \sim -v-1$사이의 항과 $n-v \sim 2n-2-2v$ 사이의 항은 모듈러 감산 연산되어야 하므로 이를 각각 r_-와 r_+로 표현하고 모듈러 감산이 필요 없는 항들을 r이라하면

$$
\begin{aligned}
r &= \sum_{i=-v}^{n-v-1}s_{i+2v}\alpha^i,\ r_- = \sum_{i=-2v}^{-v-1}s_{i+2v}\alpha^i,\ r_+ \\
&= \sum_{i=n-v}^{2n-2v-2}s_{i+2v}\alpha^i
\end{aligned}
$$

이고, $f(x)=x^n+x^k+1$로 r_-와 r_+를 모듈러 감산한 결과를 $\widetilde{r_-}$와 $\widetilde{r_+}$라 하자. 또한 [6]에서 v의 선택시 $v=k$ 또는 $v=k-1$인 경우가 최적임을 보였다. 따라서 $v=k$인 경우를 고려하면 $c(\alpha)\equiv s(\alpha)\ \mathrm{mod} f(x)$는 다음과 같다.

$$
\begin{aligned}
c(\alpha) &\equiv r+r_-+r_+ \quad \mathrm{mod} f(x) = r+\widetilde{r_-}+\widetilde{r_+} \\
&= \underbrace{\sum_{i=-v}^{n-2v-1}\left(\sum_{t=0}^{2v+i}a_tb_{2v+i-t}\right)\alpha^i}_{(a)} \\
&\quad + \underbrace{\sum_{i=n-2v}^{n-v-1}\left(\sum_{t=i+2v-n+1}^{n-1}a_tb_{2v+i-t}\right)\alpha^i}_{(b)} \\
&\quad + \underbrace{\sum_{i=n-2v}^{n-v-1}\left(\sum_{t=0}^{2v-n+i}a_tb_{2v-n+i-t}\right)\alpha^i}_{(c)} \\
&\quad + \underbrace{\sum_{i=-v}^{-1}\left(\sum_{t=0}^{v+i}a_tb_{v+i-t}\right)\alpha^i}_{(d)} \\
&\quad + \underbrace{\sum_{i=0}^{n-v-2}\left(\sum_{t=v+i+1}^{n-1}a_tb_{v+n+i-t}\right)\alpha^i}_{(e)} \\
&\quad + \underbrace{\sum_{i=-v}^{n-2v-2}\left(\sum_{t=2v+i+1}^{n-1}a_tb_{2v+n+i-t}\right)\alpha^i}_{(f)}.
\end{aligned} \tag{1}
$$

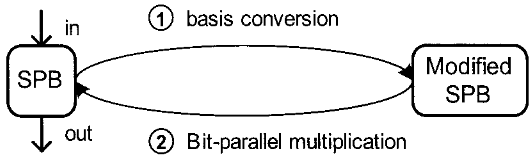
III. 제안하는 삼항 기약다항식 기반 SPB 곱셈

본 절에서는 제안하는 SPB 곱셈 방법에 대하여 기술한다. 제안하는 방법 또한 정의 1과 같이 SPB를 사용하지만 기존에 제안된[6,8] 방법과 다르게 곱셈을 수행한다. 제안하는 방법을 설명하기 위하여 삼항 기약다항식 $f(x)=x^n+x^k+1$의 환경에서 변형된 SPB를 정의하면 다음과 같다.

정의 2. v를 임의의 정수라 하고 $\Gamma=\{\alpha^i \mid 0\le i\le n-1\}$를 F_{2^n}의 다항식기저라 할 때, v와 δ가 각각

$$
v=\begin{cases} 2k, & 2k<n \\ k, & 2k=n, \\ n-k-1, & 2k>n \end{cases} \qquad \delta=\begin{cases} 2k, & 2k<n \\ k, & 2k=n \\ n, & 2k>n \end{cases}
$$

이면, 순서집합 $\alpha^{-v}\Gamma=\{\alpha^{i-v} \mid 0\le i\le k-1,\ \delta\le i\le \delta+n-k-1\}$를 Γ에 대한 변형된 MSPB(Mod ified Shifted Polynomial Basis)라 한다.



[그림 1] 제안하는 SPB 곱셈 방법

제안하는 곱셈 방법의 경우 입/출력은 기존의 SPB를 사용하지만 최적 v를 k 또는 $k-1$을 사용했던 기존의 결과와 달리 v값을

$$
v=\begin{cases} 2k, & 2k<n \\ k, & 2k=n \\ n-k-1, & 2k>n \end{cases}
$$

로 정의한다. 그리고 mastrovito 곱셈을 수행하기 전에 [그림 1]과 같이 MSPB로 기저를 변환하고 곱셈을 수행하며, 곱셈 결과는 다시 입력과 동일하게 처음의 SPB로 표현된다. 이때, [그림 1]의 기저 변환과 비트-병렬 곱셈은 병렬로 수행된다. 상세한 곱셈 방법 및 효율성은 정의 2에서와 같이 k에 따라 $n>2k$, $n=2k$ 또는 $n<2k$로 구분하여 설명한다. 이때 $n<2k$

인 경우는 $n>2k$인 경우의 곱셈 수행을 최저항부터 최고차항까지 서로 대칭이 되도록 수행하면 되므로 생략한다.

3.1 제안하는 SPB 곱셈 방법

본 절에서는 제안하는 곱셈 방법과 그에 따른 시간 및 공간 복잡도에 대하여 기술한다.

3.1.1 $n>2k$인 경우

$n>2k$인 경우는 정의 2에서와 같이 $v=2k$를 사용하며, $f(x)=x^n+x^k+1$에 의한 모듈러 감산 연산의 효율성을 최대화하기 위하여 기존의 방법과 다르게 다음과 같이 곱셈을 수행한다.

SPB로$(v=2k)$ 표현된 F_{2^n}의 임의의 두 원소 $a(\alpha)$, $b(\alpha)$는 다음과 같이 표현된다.

$$a(\alpha)=\alpha^{-2k}\left(\sum_{i=0}^{n-1}a_i\alpha^i\right),\quad b(\alpha)=\alpha^{-2k}\left(\sum_{i=0}^{n-1}b_i\alpha^i\right)$$

정의 2에 의하여 두 원소 $a(\alpha)$, $b(\alpha)$는 $f(x)$를 이용하여 다음과 같이 MSPB 원소로 변환한다.

$$\begin{aligned}
a(\alpha)&=\alpha^{-2k}\left(\sum_{i=0}^{n-1}a_i\alpha^i\right)\\
&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}a_i\alpha^i+\sum_{i=k}^{2k-1}a_i\alpha^i+\sum_{i=2k}^{n-1}a_i\alpha^i\right)\\
&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}a_i\alpha^i+\left(\sum_{i=0}^{k-1}a_{k+i}\alpha^i+\sum_{i=n}^{n+k-1}a_{i-n+k}\alpha^i\right)+\sum_{i=2k}^{n-1}a_i\alpha^i\right) \qquad (2)\\
&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}(a_i+a_{k+i})\alpha^i+\sum_{i=2k}^{n-1}a_i\alpha^i+\sum_{i=n}^{n+k-1}a_{i-n+k}\alpha^i\right),\\
b(\alpha)&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}(b_i+b_{k+i})\alpha^i+\sum_{i=2k}^{n-1}b_i\alpha^i+\sum_{i=n}^{n+k-1}b_{i-n+k}\alpha^i\right).
\end{aligned}$$

이때 $\hat{a}_{i,j}$와 $\overline{a_i}$를 각각

$$\widehat{a_{i,j}}=\begin{cases}a_i & ,i<n\\ a_{-n+i+j} & ,i\geq n\end{cases},\quad \overline{a_i}=a_i+a_{i+k}$$

라 하면 $a(\alpha)$, $b(\alpha)$는

$$\begin{aligned}
a(\alpha)&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}\overline{a_i}\alpha^i+\sum_{i=2k}^{n+k-1}\widehat{a_{i,k}}\alpha^i\right)\\
&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}\overline{a_i}\alpha^i+\left(\sum_{i=2k}^{n+k-1}\widehat{a_{i,k}}\alpha^{i-n}\right)\alpha^n\right),\\
b(\alpha)&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}\overline{b_i}\alpha^i+\sum_{i=2k}^{n+k-1}\widehat{b_{i,k}}\alpha^i\right)\\
&=\alpha^{-2k}\left(\sum_{i=0}^{k-1}\overline{b_i}\alpha^i+\left(\sum_{i=2k}^{n+k-1}\widehat{b_{i,k}}\alpha^{i-n}\right)\alpha^n\right)
\end{aligned}$$

이다. 따라서 $c(\alpha)=a(\alpha)b(\alpha)$는 다음과 같다.

$$\begin{aligned}
c(\alpha)&=\alpha^{-2k}\left(\underbrace{\sum_{i=0}^{k-1}\overline{a_i}\alpha^i}_{A_0}+\left(\underbrace{\sum_{i=2k}^{n+k-1}\widehat{a_{i,k}}\alpha^{i-n}}_{A_1}\right)\alpha^n\right)\\
&\quad\cdot\alpha^{-2k}\left(\underbrace{\sum_{i=0}^{k-1}\overline{b_i}\alpha^i}_{B_0}+\left(\underbrace{\sum_{i=2k}^{n+k-1}\widehat{b_{i,k}}\alpha^{i-n}}_{B_1}\right)\alpha^n\right)\\
&=\alpha^{-4k}\big(A_0B_0+[(A_0+A_1)(B_0+B_1)+A_0B_0+A_1B_1]\alpha^n\\
&\qquad+A_1B_1\alpha^{2n}\big) \qquad (3)\\
&=\alpha^{-4k}\big([A_0B_0+A_0B_0\alpha^n]+[(A_0+A_1)(B_0+B_1)]\alpha^n\\
&\qquad+[A_1B_1\alpha^n+A_1B_1\alpha^{2n}]\big)\\
&=\alpha^{-4k}\big(A_0B_0\alpha^k+[(A_0+A_1)(B_0+B_1)]\alpha^n+A_1B_1\alpha^{n+k}\big).
\end{aligned}$$

이때, (A_0+A_1)는

$$\begin{aligned}
(A_0+A_1)&=\sum_{i=0}^{k-1}\overline{a_i}\alpha^i+\sum_{i=2k}^{n+k-1}\widehat{a_{i,k}}\alpha^{i-n}\\
&=\sum_{i=0}^{k-1}(a_i+a_{i+k})\alpha^i+\left(\sum_{i=2k}^{n-1}a_i\alpha^{i-n}+\sum_{i=0}^{k-1}a_{i+k}\alpha^i\right)\\
&=\sum_{i=2k}^{n-1}a_i\alpha^{i-n}+\sum_{i=n}^{n+k-1}a_{i-n}\alpha^{i-n}\\
&=\sum_{i=2k}^{n+k-1}\widehat{a_{i,0}}\alpha^{i-n}
\end{aligned}$$

이므로 (A_0+A_1)와 (B_0+B_1)은 실제 연산이 없으며, 기저 변환시 연산이 수행되는 기저원소와 그렇지 않은 원소가 독립적으로 곱셈을 수행하므로 기전 변환은 병렬연산이 가능하므로 식 (3)은 다음과 같다.

$$\begin{aligned}
c(\alpha)=&\ \alpha^{-4k}\Bigg[\left(\sum_{i=0}^{k-1}\overline{a_i}\alpha^i\right)\left(\sum_{i=0}^{k-1}\overline{b_i}\alpha^i\right)\alpha^k\\
&+\left(\sum_{i=2k}^{n+k-1}\widehat{a_{i,0}}\alpha^i\right)\left(\sum_{i=2k}^{n+k-1}\widehat{b_{i,0}}\alpha^i\right)\alpha^{-n}\\
&+\left(\sum_{i=2k}^{n+k-1}\widehat{a_{i,k}}\alpha^i\right)\left(\sum_{i=2k}^{n+k-1}\widehat{b_{i,k}}\alpha^i\right)\alpha^{-n+k}\Bigg]\\
=&\sum_{t=0}^{2k-2}\left(\sum_{\substack{i+j=t\\0\leq i,j\leq k-1}}\overline{a_i}\,\overline{b_j}\right)\alpha^{t-3k}\\
&+\sum_{t=4k}^{2n+2k-2}\left(\sum_{\substack{i+j=t\\2k\leq i,j\leq n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^{t-n-4k}\\
&+\sum_{t=4k}^{2n+2k-2}\left(\sum_{\substack{i+j=t\\2k\leq i,j\leq n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^{-n-4k+t}\\
&+\sum_{t=4k}^{2n+2k-2}\left(\sum_{\substack{i+j=t\\2k\leq i,j\leq n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^{-n-3k+t}\\
=&\underbrace{\sum_{t=-3k}^{-2k-1}\left(\sum_{\substack{i+j=t+3k\\0\leq i,j\leq k-1}}\overline{a_i}\,\overline{b_j}\right)\alpha^t}_{(ㄱ)}
\end{aligned}$$

$$
\begin{aligned}
&+\underbrace{\sum_{t=-2k}^{-k-2}\left(\sum_{\substack{i+j=t+3k\\0\le i,j\le k-1}}\overline{a_i b_j}\right)\alpha^t}_{\text{(ㄴ)}}\\
&+\underbrace{\sum_{t=-n}^{-2k-1}\left(\sum_{\substack{i+j=n+4k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t}_{\text{(ㄷ)}}\\
&+\underbrace{\sum_{t=-2k}^{n-2k-2}\left(\sum_{\substack{i+j=n+4k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t}_{\text{(ㄹ)}}\\
&+\underbrace{\sum_{t=-n+k}^{-k-1}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^t}_{\text{(ㅁ)}}\\
&+\underbrace{\sum_{t=-k}^{n-2k-1}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^t}_{\text{(ㅂ)}}\\
&+\underbrace{\sum_{t=n-2k}^{n-k-2}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^t.}_{\text{(ㅅ)}}
\end{aligned}
\tag{4}
$$

식 (4)에서 (ㄱ), (ㄷ), (ㅅ)는 모듈러 감산 되어야하며 $f(x)=x^n+x^k+1$에 의하여 (ㄷ)+(ㅁ)은

$$
\begin{aligned}
&\sum_{t=-n}^{-2k-1}\left(\sum_{\substack{i+j=n+4k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t+\sum_{t=-n+k}^{-k-1}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^t\\
&=\sum_{t=-n}^{-2k-1}\left(\sum_{\substack{i+j=n+4k+t\\2k\le i,j\le n+k-1}}a_i b_j\right)\alpha^t+\sum_{t=-n+k}^{-k-1}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}a_i b_j\right)\alpha^t\\
&\equiv\sum_{t=0}^{n-2k-1}\left(\sum_{\substack{i+j=4k+t\\2k\le i,j\le n+k-1}}a_i b_j\right)\alpha^t \mod f(x)
\end{aligned}
\tag{5}
$$

이므로 식 (5)과 (ㄱ), (ㅅ)을 모듈러 감산 연산하면 식 (4)는 다음과 같이 정리된다.

$$
\begin{aligned}
c(\alpha)=&\underbrace{\sum_{t=-2k}^{-k-1}\left(\sum_{\substack{i+j=2k+t\\0\le i,j\le k-1}}\overline{a_i b_j}\right)\alpha^t}_{(a2)}\\
&+\underbrace{\sum_{t=n-3k}^{n-2k-1}\left(\sum_{\substack{i+j=-n+3k+t\\0\le i,j\le k-1}}\overline{a_i b_j}\right)\alpha^t}_{(a1)}\\
&+\sum_{t=-2k}^{-k-2}\left(\sum_{\substack{i+j=3k+t\\0\le i,j\le k-1}}\overline{a_i b_j}\right)\alpha^t\\
&+\sum_{t=0}^{n-2k-1}\left(\sum_{\substack{i+j=4k+t\\2k\le i,j\le n+k-1}}a_i b_j\right)\alpha^t\\
&+\sum_{t=-2k}^{n-2k-2}\left(\sum_{\substack{i+j=n+4k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t\\
&+\sum_{t=-k}^{n-2k-1}\left(\sum_{\substack{i+j=n+3k+t\\2k\le i,j\le n+k-1}}\widehat{a_{i,k}}\widehat{b_{j,k}}\right)\alpha^t\\
&+\underbrace{\sum_{t=-2k}^{-k-2}\left(\sum_{\substack{i+j=5k+t\\k+1\le i,j\le 2k-1}}a_i b_j\right)\alpha^t}_{(b1)}\\
&+\underbrace{\sum_{t=-k}^{-2}\left(\sum_{\substack{i+j=4k+t\\k+1\le i,j\le 2k-1}}a_i b_j\right)\alpha^t}_{(b2)}
\end{aligned}
\tag{6}
$$

또한 식 (4)에서 $\hat{a}_{i,0}$과 $\hat{a}_{i,k}$는 i가 n보다 작은 경우 a_i로 같으므로 $-k\le t\le n-3k-2$에서 (ㄹ) 각항의 $\hat{a}_{t+3k+1,0}\hat{b}_{n+k-1,0}+\cdots+\hat{a}_{n-1,0}\hat{b}_{t+4k+1,0}$과 (ㅂ) 각항의 $\hat{a}_{t+3k+1,k}\hat{b}_{n-1,k}+\cdots+\hat{a}_{n-1,k}\hat{b}_{t+3k+1,k}$는

$$
\begin{aligned}
&a_{t+3k+1}(\widehat{b_{n+k-1}}+\widehat{b_{n-1}})+\cdots\\
&\qquad+a_{n-1}(\widehat{b_{t+4k+1}}+\widehat{b_{t+3k+1}})
\end{aligned}
\tag{7}
$$

로 간소화된다. 따라서 이를 적용하여 식 (6)의 각항을 k에 따른 XOR게이트 시간 지연으로 정리하면 〔표 1〕과 같다. 〔표 1〕에서 시간 지연을 계산할 때 k에 따라 다른 행들이 사용된다. 예를 들어 $k=1$인 경우 t의 범위는 $t=-k-1$, $t=-1$의 3행, $0\le t\le n-3k-2$, $t=n-3k-1$, $t=n-2k-1$이며 나머지 행들은 $k=1$에서는 존재하지 않는다. 이와 같은 비트-병렬 곱셈을 type I mastrovito 곱셈기라 하며 다음 정리를 만족한다.

정리 1.$(n>2k)$ 제안하는 type I mastrovito 곱셈기는 c_k 또는 c_0에서 최대 시간 지연을 가지며

$$
\text{Time Delay}=\begin{cases}1\mathrm{T_A}+\lceil\log_2(2n-2k-1)\rceil\mathrm{T_X}, & 3\mathrm{k}+1<\mathrm{n}\\ 1\mathrm{T_A}+\lceil\log_2(n+k)\rceil\mathrm{T_X}, & 3\mathrm{k}+1\ge\mathrm{n}\end{cases}
$$

이다.

증명. 〔표 1〕에서 각항의 시간 지연 계산 시 $\overline{a_i b_j}$ 또는 식 (7)의 $(\hat{b}_i+\hat{b}_j)$는 한 번의 덧셈을 수행한 후 곱셈과 덧셈을 순차적으로 수행해야한다. 따라서 $-2k\le t\le -k-2$인 c_{t+2k}는 $1\mathrm{T_A}+1\mathrm{T_X}$시간 이후 k개의 $\overline{a_i b_j}$와 $(n-k)/2$개의 $a_i b_j$ 합으로 계산되므로 $1\mathrm{T_A}+(1+\lceil\log_2(k+(n-k)/2)\rceil)\mathrm{T_X}=1\mathrm{T_A}+\lceil\log_2(n+k)\rceil\mathrm{T_X}$이다. 나머지 항들도 이와 같은 방법으로 계산하면 〔표 1〕의 XOR Delay와 같으며, k에 따라 $\lceil\log_2(n+k)\rceil$ 또는 $\lceil\log_2(2n-2k-1)\rceil$이 최대 시간 지연(Critic al Path Delay)이 되므로 $n>2k$ 일때 type I mastrovito 곱셈기 시간 복잡도는

〔표 1〕 $n>2k$일때 c_{t+2k} 계산에 따른 XOR 시간 지연

t의 범위	c_t의 계산	k의 조건	XOR Delays
$-2k \le t \le -k-2$	$\sum_{i=t+2k+1}^{k-1} \bar{a}_i \bar{b}_{t+3k-i} + \underbrace{\sum_{i=t+3k+1}^{2k-1} a_i b_{t+5k-i}}_{(b1)} + \underbrace{\sum_{i=0}^{t+2k} \bar{a}_i \bar{b}_{t+2k-i}}_{(a2)} + \sum_{i=2k}^{t+n+2k} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0}$	$2 \le k \le \frac{n-1}{2}$	$\lceil \log_2(n+k) \rceil$
$t=-k-1$	$\underbrace{\sum_{i=0}^{t+2k} \bar{a}_i \bar{b}_{t+2k-i}}_{(a2)} + \sum_{i=2k}^{t+n+2k} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0}$	$1 \le k \le \frac{n-1}{2}$	$\lceil \log_2(n+k) \rceil$
$-k \le t$ and $t \le \min\{n-3k-2, -2\}$	$\underbrace{\sum_{i=t+2k+1}^{2k-1} a_i b_{t+4k-i}}_{(b2)} + \sum_{i=t+3k+1}^{n-1} a_i \left(\hat{b}_{t+n+4k-i,0} + \hat{b}_{t+n+3k-i,k}\right) + \sum_{i=n}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+3k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k} + \sum_{i=n}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$2 \le k \le \frac{n-2}{2}$	$\lceil \log_2(2n-3k-t-1) \rceil$
$t=n-3k-1$	$\sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k} + \sum_{i=t+2k+1}^{2k-1} a_i b_{t+4k-i}$	$\frac{n+1}{3} \le k \le \frac{n-1}{2}$	$\lceil \log_2(2n-3k-t-1) \rceil$
$n-3k \le t \le -2$	$\underbrace{\sum_{i=0}^{t-n+3k} \bar{a}_i \bar{b}_{t-n+3k-i}}_{(a1)} + \sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k} + \sum_{i=t+2k+1}^{2k-1} a_i b_{t+4k-i}$	$\frac{n+2}{3} \le k \le \frac{n-1}{2}$	$\lceil \log_2(3k+t+1) \rceil$
$t=-1$	$\underbrace{\sum_{i=0}^{t-n+3k} \bar{a}_i \bar{b}_{t-n+3k-i}}_{(a1)} + \sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$\frac{n+1}{3} \le k \le \frac{n-1}{2}$	$\lceil \log_2(3k) \rceil$
	$\sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$k=\frac{n}{3}$	$\lceil \log_2(2n-3k) \rceil = \lceil \log_2(n) \rceil$
	$\sum_{i=t+3k+1}^{n-1} a_i \left(\hat{b}_{t+n+4k-i,0} + \hat{b}_{t+n+3k-i,k}\right) + \sum_{i=n}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=2k}^{t+3k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k} + \sum_{i=n}^{t+n+k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$1 \le k \le \frac{n-1}{3}$	$\lceil \log_2(2n-3k) \rceil$
$0 \le t \le n-3k-2$	$\sum_{i=2k}^{t+2k} a_i b_{t+k-i} + \sum_{i=t+3k+1}^{n-1} a_i \left(\hat{b}_{n+4k-1-i,0} + \hat{b}_{n+3k-1-i,k}\right) + \sum_{i=n}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=t+2k+1}^{t+3k} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k} + \sum_{i=n}^{n+k-1} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$1 \le k \le \frac{n-2}{3}$	$\lceil \log_2(2n-3k-t-1) \rceil$
$t=n-3k-1$	$\sum_{i=2k}^{t+2k} a_i b_{t+k-i} + \sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=t+2k+1}^{n+k-1} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$1 \le k \le \frac{n-1}{3}$	$\lceil \log_2(2n-3k-t-1) \rceil$
$n-3k \le t$ and $t \le n-2k-2$	$\underbrace{\sum_{i=0}^{t-n+3k} \bar{a}_i \bar{b}_{t-n+3k-i}}_{(a1)} + \sum_{i=2k}^{t+2k} a_i b_{t+k-i} + \sum_{i=t+3k+1}^{n+k-1} \hat{a}_{i,0} \hat{b}_{t+n+4k-i,0} + \sum_{i=t+2k+1}^{n+k-1} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$2 \le k \le \frac{n-2}{2}$	$\lceil \log_2(3k+t+1) \rceil$
$t=n-2k-1$	$\underbrace{\sum_{i=0}^{t-n+3k} \bar{a}_i \bar{b}_{t-n+3k-i}}_{(a1)} + \sum_{i=2k}^{t+2k} a_i b_{t+k-i} + \sum_{i=t+2k+1}^{n+k-1} \hat{a}_{i,k} \hat{b}_{t+n+3k-i,k}$	$1 \le k \le \frac{n-1}{2}$	$\lceil \log_2(n+k) \rceil$

$$\text{TimeDelay}=\begin{cases}1\mathrm{T_A}+\log_2(2n-2k-1)\mathrm{T_X} & ,3\mathrm{k}+1<\mathrm{n}\\ 1\mathrm{T_A}+\log_2(n+k)\mathrm{T_X} & ,3\mathrm{k}+1\geq \mathrm{n}\end{cases}$$

이다. □

정리 2.($n>2k$) Type I mastrovito 곱셈기의 공간 복잡도는

$$\#\mathrm{AND}=\mathrm{n}^2-\mathrm{k}^2,$$
$$\#\mathrm{XOR}=\mathrm{n}^2-1$$

이다.

증명. 식 (4)의 (ㄱ)+(ㄴ)에서 k^2개 AND 게이트가 소요되며 (ㄷ)+(ㄹ)와 (ㅁ)+(ㅂ)+(ㅅ)에서 각각 $(n-k)^2$개 AND 게이트가 필요하다. 또한 식 (5)에서 $(n-2k)\cdot(n-2k+1)$개 AND 게이트가 $(n-2k)(n-2k+1)/2$개 AND 게이트로 감소하고 식 (7)에서는 $(n-2k-1)(n-2k)$개 AND 게이트가 $(n-2k-1)(n-2k)/2$개 AND 게이트로 감소하므로 전체 $(n-2k)^2$개 AND 게이트가 감소한다. 따라서 전체 AND 게이트 소요량은 $k^2+2(n-k)^2-(n-2k)^2=n^2-k^2$개이다. XOR 게이트의 경우 〔표 1〕의 모든 c_{t+2k}의 계산은 정확히 n개의 a_ib_j, $a_i(b_j+b_m)$ 또는 $\overline{a_i}\,\overline{b_j}$를 더하여 계산되므로 $n(n-1)$개 XOR 게이트가 소요되며, $\overline{a_i}$와 $\overline{b_i}$의 계산에서 $2k$개 XOR 게이트와 식 (7)의 계산에서 $2n-2k-1$개 XOR 게이트가 소요되므로 전체 n^2-1개의 XOR 게이트가 소요된다. □

Type I mastrovito 곱셈기의 경우 시간 복잡도가 최소가 되도록 구성하였다. 따라서 공간 복잡도 입장에서는 식 (6)의 (a1)과 (a2), (b1)과 (b2)는 중복 연산이므로 독립 연산 후 다른 항들과 더해져야한다. 따라서 〔표 1〕의 $t=n-2k-1$인 경우를 고려하면 각각 k, $n-2k$, k개의 $\overline{a_i}\,\overline{b_j}$ 또는 a_ib_j를 더해야한다. 이 때 $\sum_{i=0}^{k-1}\overline{a_i}\,\overline{b}_{k-1-i}$가 독립 연산이므로 이를 더하는데 소요되는 XOR 시간 지연은 $(1+\lceil\log_2(k)\rceil)\mathrm{T_X}$이고 이 때 나머지 $n-k$개는 $(1+\lceil\log_2(k)\rceil)\mathrm{T_X}$ 시간동안 매번 반으로 감소하므로 $t=n-2k-1$일 때 전체 시간 복잡도는 다음과 같다.

$$1\mathrm{T_A}+\left\lceil \lceil\log_2(2k)\rceil+\log_2\!\left(\frac{n-k}{2^{\lceil\log_2(2k)\rceil}}+1\right)\right\rceil\mathrm{T_X}$$
$$=1\mathrm{T_A}+\left\lceil\log_2\!\left(n-k+2^{\lceil\log_2(2k)\rceil}\right)\right\rceil\mathrm{T_X}.$$

이와 같이 제안하는 방법에서 공간복잡도가 최소가 되도록 구성한 것을 type II mastrovito 곱셈기라 하고 다음 정리를 만족한다.

정리 3.($n>2k$) Type II mastrovito 곱셈기의 시간 복잡도는

1. $k=1$인 경우, $\text{Time Delay}=1\mathrm{T_A}+(\log_2(2n-3))\mathrm{T_X}$
2. $k\neq 1$이고 $k=2^m+1$인 경우,

$$\text{TimeDelay}=\begin{cases}1\mathrm{T_A}+\log_2(n-k+2^{\lceil\log_2(k)\rceil})\mathrm{T_X} & ,5\mathrm{k}-3>\mathrm{n}\\ 1\mathrm{T_A}+\log_2(2n-3k+2^{\lceil\log_2(k-1)\rceil}) & ,5k-3\leq n\end{cases}$$

3. $k\neq 1$이고 $k\neq 2^m+1$인 경우,

$$\text{TimeDelay}=\begin{cases}1\mathrm{T_A}+\log_2(n+3\cdot 2^{\lceil\log_2(k-1)\rceil-1}+2^{\lceil\log_2(u)\rceil}-1)\mathrm{T_X} & ,2^{\lceil\log_2(k-1)\rceil-1}+2^{\lceil\log_2(u)\rceil}-1>n-3k\\ 1\mathrm{T_A}+\log_2(2n-3k+2^{\lceil\log_2(k-1)\rceil})\mathrm{T_X} & ,2^{\lceil\log_2(k-1)\rceil-1}+2^{\lceil\log_2(u)\rceil}-1\leq n-3k\end{cases}$$

이고 공간 복잡도는

$$\#\mathrm{AND}=\mathrm{n}^2-\mathrm{k}^2,$$
$$\#\mathrm{XOR}=\mathrm{n}^2-1-(\mathrm{k}-1)^2$$

이다.(이때,

$k-1=2^{\lceil\log_2(k-1)\rceil-1}+u\ (1\leq u\leq 2^{\lceil\log_2(k-1)\rceil-1})$

증명. 〔표 1〕에서 $k=1$인 경우는 중복인 독립 연산이 없으므로 $\lceil\log_2(2n-3)\rceil$의 XOR 시간지연을 가지며, $k\neq 1$인 경우 t가

$$-2k\leq t\leq -k-2,$$
$$t=-k-1$$
$$-k\leq t\leq \min\{\mathrm{n}-3\mathrm{k}-2,-2\}$$

일 때 중복인 독립 연산이 있으므로 위와 같은 방법으로 이들 각각에 대한 XOR 시간 지연을 계산하면

$$\begin{aligned}&\lceil\log_2(n-i+2^{\lceil\log_2(2i)\rceil}+2^{\lceil\log_2(k-i)\rceil})\rceil\mathrm{T_X}\Rightarrow(8.1),\\ &\lceil\log_2(n-k+2^{\lceil\log_2(k)\rceil})\rceil\mathrm{T_X}\quad\Rightarrow(8.2),\\ &\lceil\log_2(2n-3k+2^{\lceil\log_2(k-j)\rceil})\rceil\mathrm{T_X}\quad\Rightarrow(8.3)\end{aligned}\qquad(8)$$

이다.($1\leq i\leq k-1$이고 $1\leq j\leq\min\{\mathrm{n}-\mathrm{k}-1,\mathrm{k}-1\}$이다.) 임의의 k를 $k-1=2^m+u=2^{\lceil\log_2(k-1)\rceil-1}+u$, $(1\leq u\leq 2^m)$라 하면 (8.1)은 $2^{\lceil\log_2(2i)\rceil}$가 최대값을 가지는 가장 작은 i값에서 최대 시간 지연을 가지므로 $i=2^{\lceil\log_2(k-1)\rceil-1}+1$이고, (8.2)는 최소의 i값인 $i=1$에서 최대값을 가지므로 최대 시간 지연은

$$\log_2(n+3\cdot 2^{\lceil\log_2(k-1)\rceil-1}+2^{\lceil\log_2(u)\rceil}-1)\mathrm{T_X}\quad(8.1)`$$
$$\log_2(n-k+2^{\lceil\log_2(k)\rceil})\mathrm{T_X}\quad(8.2)`$$
$$\log_2(2n-3k+2^{\lceil\log_2(k-1)\rceil})\mathrm{T_X}\quad(8.3)`$$

이다. 우선 $k=2^m+1$일 때를 고려하면 (8.1)`〈

(8.2)`이므로 (8.2)`와 (8.3)`중 큰 값이 최대 시간 지연이 된다. 따라서

$$\begin{cases} \log_2(n-k+2^{\lceil \log_2(k) \rceil})\mathrm{T_X} & ,5\mathrm{k}-3>\mathrm{n} \\ \log_2(2n-3k+2^{\lceil \log_2(k-1) \rceil})\mathrm{T_X} & ,5\mathrm{k}-3 \le \mathrm{n} \end{cases}$$

이다. 다음으로 $k \neq 2^m+1$를 고려하면 (8.1)`〉(8.2)`이므로 (8.1)`와 (8.3)`중 큰 값이 최대 시간 지연이 된다. 따라서

$$\mathrm{TimeDelay} = \begin{cases} 1\mathrm{T_A} + \log_2(n+3\cdot 2^{\lceil \log_2(k-1) \rceil -1} + 2^{\lceil \log_2(u) \rceil} -1)\mathrm{T_X} \\ \qquad ,2^{\lceil \log_2(k-1) \rceil -1} + 2^{\lceil \log_2(u) \rceil} -1 > n-3k \\ 1\mathrm{T_A} + \log_2(2n-3k+2^{\lceil \log_2(k-1) \rceil})\mathrm{T_X} \\ \qquad ,2^{\lceil \log_2(k-1) \rceil -1} + 2^{\lceil \log_2(u) \rceil} -1 \le n-3k \end{cases}$$

이다. 공간 복잡도의 경우 〔표 1〕에서 (a2)와 (b2)의 계산시 소요되는 XOR 게이트를 빼면 되므로 (a2)의 계산에서 $k(k-1)/2$개 XOR 게이트가 중복이며 (b2)의 계산에서 $(k-1)(k-2)/2$개 XOR가 중복 연산이므로 전체 $(k-1)^2$개 XOR 게이트가 감소한다. 따라서 type II mastrovito 곱셈기의 공간 복잡도는

$$\#\mathrm{AND} = \mathrm{n}^2 - \mathrm{k}^2,$$
$$\#\mathrm{XOR} = \mathrm{n}^2 - 1 - (\mathrm{k}-1)^2$$

이다. □

3.1.2 $n=2k$인 경우

$n=2k$인 경우 기존 〔8〕에서와 같이 $v=k$이며 Karatsuba-Ofman 방법을 적용하기 위하여 $a(\alpha)$와 $b(\alpha)$는 다음과 같이 변형된다.

$$\begin{aligned} a(\alpha) &= \alpha^{-k}\left(\sum_{i=0}^{n-1} a_i\alpha^i\right) = \alpha^{-k}\left(\sum_{i=0}^{k-1} a_i\alpha^i + \sum_{i=k}^{n-1} a_i\alpha^i\right) \\ &= \alpha^{-k}\left(\sum_{i=0}^{k-1} a_i\alpha^i + \left(\sum_{i=0}^{k-1} a_{i+k}\alpha^i\right)\alpha^k\right), \\ b(\alpha) &= \alpha^{-k}\left(\sum_{i=0}^{n-1} b_i\alpha^i\right) = \alpha^{-k}\left(\sum_{i=0}^{k-1} b_i\alpha^i + \sum_{i=k}^{n-1} b_i\alpha^i\right) \\ &= \alpha^{-k}\left(\sum_{i=0}^{k-1} b_i\alpha^i + \left(\sum_{i=0}^{k-1} b_{i+k}\alpha^i\right)\alpha^k\right). \end{aligned}$$

이때, 두 원소의 곱 $c(\alpha) \equiv a(\alpha)b(\alpha)$는 다음과 같다.

$$\begin{aligned} c(\alpha) &\equiv \alpha^{-k}\left(\underbrace{\sum_{i=0}^{k-1} a_i\alpha^i}_{A_0} + \left(\underbrace{\sum_{i=0}^{k-1} a_{i+k}\alpha^i}_{A_1}\right)\alpha^k\right) \\ &\quad \cdot \alpha^{-k}\left(\underbrace{\sum_{i=0}^{k-1} b_i\alpha^i}_{B_0} + \left(\underbrace{\sum_{i=0}^{k-1} b_{i+k}\alpha^i}_{B_1}\right)\alpha^k\right) \end{aligned}$$

$$\begin{aligned} &= \alpha^{-2k}\left(A_0B_0 + [(A_0+A_1)(B_0+B_1) + A_0B_0 + A_1B_1]\alpha^k \right. \\ &\qquad \left. + A_1B_1\alpha^{2k}\right) \\ &= \alpha^{-2k}\left([A_0B_0 + A_0B_0\alpha^k] + [(A_0+A_1)(B_0+B_1)]\alpha^k \right. \\ &\qquad \left. + [A_1B_1\alpha^k + A_1B_1\alpha^n]\right) \\ &= \alpha^{-2k}\left(A_0B_0\alpha^n + [(A_0+A_1)(B_0+B_1)]\alpha^k + A_1B_1\right) \end{aligned} \tag{9}$$

$\bar{a}_i = a_i + a_{i+k}$라 하고 식 (9)를 정리하면

$$\begin{aligned} c(\alpha) &= \sum_{t=0}^{2k-2}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} a_{i+k}b_{j+k}\right)\alpha^{t-2k} \\ &\quad + \sum_{t=0}^{2k-2}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} \bar{a}_i\bar{b}_j\right)\alpha^{t-k} \\ &\quad + \sum_{t=0}^{2k-2}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} a_ib_j\right)\alpha^{t} \\ &= \sum_{t=-2k}^{-2}\left(\sum_{\substack{i+j=t+2k \\ 0\le i,j\le k-1}} a_{i+k}b_{j+k}\right)\alpha^{t} \\ &\quad + \sum_{t=-k}^{k-2}\left(\sum_{\substack{i+j=t+k \\ 0\le i,j\le k-1}} \bar{a}_i\bar{b}_j\right)\alpha^{t} \\ &\quad + \sum_{t=0}^{2k-2}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} a_ib_j\right)\alpha^{t} \\ &= \left(\sum_{t=-k}^{-2}\left(\sum_{\substack{i+j=t+2k \\ 0\le i,j\le k-1}} a_{i+k}b_{j+k}\right)\alpha^{t}\right. \\ &\quad + \sum_{t=-k}^{k-2}\left(\sum_{\substack{i+j=t+k \\ 0\le i,j\le k-1}} \bar{a}_i\bar{b}_j\right)\alpha^{t} \\ &\quad \left. + \sum_{t=0}^{k-1}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} a_ib_j\right)\alpha^{t}\right) \\ &\quad + \left(\sum_{t=-k}^{-1}\left(\sum_{\substack{i+j=t+k \\ 0\le i,j\le k-1}} a_{i+k}b_{j+k}\right)\alpha^{t}\right. \\ &\quad \left. + \sum_{t=-k}^{-2}\left(\sum_{\substack{i+j=t+2k \\ 0\le i,j\le k-1}} a_ib_j\right)\alpha^{t}\right) \Rightarrow (10.1) \\ &\quad + \left(\sum_{t=0}^{k-1}\left(\sum_{\substack{i+j=t \\ 0\le i,j\le k-1}} a_{i+k}b_{j+k}\right)\alpha^{t}\right. \\ &\quad \left. + \sum_{t=0}^{k-2}\left(\sum_{\substack{i+j=t+k \\ 0\le i,j\le k-1}} a_ib_j\right)\alpha^{t}\right) \Rightarrow (10.2) \end{aligned} \tag{10}$$

이다. 이를 이용하여 $n=2k$일때 제안하는 type I mastrovito 곱셈기는 다음 정리를 만족한다.

정리 4. $(n=2k)$ Type I mastrovito 곱셈기의 c_{-1+k}에서 최대 시간 지연을 가지며

$$\mathrm{Time\ \ Delay} = 1\mathrm{T_A} + (\log_2(n+k))\mathrm{T_X}$$

이다.

〔표 2〕 $n=2k$일 때 c_{t+k} 계산에 따른 XOR 시간 지연

t의 범위	c_t의 계산	XOR Delays
$-k \le t \le -2$	$\sum_{i=t+k+1}^{k-1} a_{i+k}b_{t+3k-i} + \sum_{i=0}^{t+k} \overline{a_i}\,\overline{b}_{t+k-i} + \underbrace{\sum_{i=0}^{t+k} a_{i+k}b_{t+2k-i}}_{(c1)} + \underbrace{\sum_{i=t+k+1}^{k-1} a_i b_{t+2k-i}}_{(d1)}$	$\lceil \log_2 (n+k+t+1) \rceil$
$t=-1$	$\sum_{i=0}^{k-1} \overline{a}_i \overline{b}_{k-1-i} + \underbrace{\sum_{i=0}^{k-1} a_{i+k}b_{2k-1-i}}_{(c1)}$	$\lceil \log_2 (n+k) \rceil$
$0 \le t \le k-2$	$\sum_{i=t+1}^{k-1} \overline{a_i}\,\overline{b}_{t+k-i} + \sum_{i=0}^{t} a_i b_{t-i} + \underbrace{\sum_{i=0}^{t} a_{i+k}b_{t+k-i}}_{(c2)} + \underbrace{\sum_{i=t+1}^{k-1} a_i b_{t+k-i}}_{(d2)}$	$\lceil \log_2 (n+k-t-1) \rceil$
$t=k-1$	$\sum_{i=0}^{k-1} a_i b_{k-1-i} + \underbrace{\sum_{i=0}^{k-1} a_{i+k}b_{2k-1-i}}_{(c2)}$	$\lceil \log_2 (n) \rceil$

증명. 식 (10)의 각항을 XOR 게이트의 시간 지연에 따라 정리하면 〔표 2〕와 같다. 〔표 2〕에서 각항의 시간 지연 계산 시 $\overline{a_i}\,\overline{b_j}$는 한 번의 덧셈을 수행한 후 곱셈과 덧셈을 순차적으로 수행해야한다. 따라서 $-k \le t \le -2$인 c_{t+k}는 $1T_A+1T_X$시간 이후 $t+k+1$개의 $\overline{a_i}\,\overline{b_j}$와 $(-2t+k-3)/2$개의 a_ib_j 합으로 계산되므로 $1T_A+(1+\lceil \log_2 (t+k+1+(-2t+k-3)/2)) \rceil T_X = 1T_A + \lceil \log_2 (n+k-1) \rceil T_X$이다. 나머지 항들도 이와 같은 방법으로 계산하면 〔표 2〕의 XOR Delay와 같다. 이중 $t=-1$일때 최대 시간 지연을 가지므로 type I mastrovito 곱셈기의 시간 복잡도는

$$\text{Time Delay} = 1T_A + \lceil \log_2 (n+k) \rceil T_X$$

이다. □

정리 5.($n=2k$) Type I mastrovito 곱셈기의 공간 복잡도는

$$\#AND = n^2-k^2 = (3/4)n^2,$$
$$\#XOR = (3/4)n^2+n-1$$

이다.

증명. A_0B_0, $(A_0+A_1)(B_0+B_1)$, A_1B_1은 각각 $k-1=n/2-1$차의 곱이므로 $(3/4)n^2$개의 AND 게이트가 필요하며, XOR 게이트의 경우 A_0B_0, $(A_0+A_1)(B_0+B_1)$, A_1B_1의 계산에서 $3(k-1)^2$개 필요하고 $\overline{a_i}$, $\overline{b_i}$의 계산에서 n개, A_0B_0, $(A_0+A_1)\cdot(B_0+B_1)$, A_1B_1 사이의 합에서 $n-2$개 식 (9)의 모듈러 감산에서 $2n-2$개가 필요하므로 총 $(3/4)\cdot n^2+n-1$개의 XOR 게이트가 필요하다. 따라서 공간 복잡도는

$$\#AND = n^2-k^2 = (3/4)n^2,$$
$$\#XOR = (3/4)n^2+n-1$$

이다. □

이전 절과 같이 $n=2k$인 경우도 중복 연산이 있으므로 각항에서 다른 덧셈들과 독립적으로 수행되어야 한다. 즉, 식 (10)에서 (10.1)과 (10.2)는 중복 연산이므로 (10.1)이 모두 연산된 후 각항에 더해져야 한다. 따라서 type I mastrovito 곱셈기 보다는 다소 시간 지연이 증가하나 공간 복잡도는 감소한다. 정리 4과 5에 의하여 type II mastrovito 곱셈기는 다음 정리를 만족한다.

정리 6.($n=2k$) Type II mastrovito 곱셈기의 시간 복잡도는

$$\text{Time Delay} = 1T_A + \left\lceil \log_2 (n+2^{\lceil \log_2(k) \rceil}) \right\rceil T_X$$

이고 공간 복잡도는

$$\#AND = n^2-k^2 = (3/4)n^2,$$
$$\#XOR = (1/2)n^2+3k-1$$

이다.

증명. 〔표 2〕에서 t를 $-k \le t \le -1$, $0 \le t \le k-1$로 구분하여 독립 연산을 고려하자. 우선 $-k \le t \le -1$일 때 (c1)+(d1)은 k개의 a_ib_j의 덧셈이고 나머지는 $(-t-1)+2\cdot(t+k+1)$개의 a_ib_j의 덧셈이다.(이때, $\overline{a_i}\,\overline{b_j}$는 $1T_X$ 이후에 계산되므로 2배로 가산한다.) 따라서 (c1)+(d1)의 k개를 모두 더하는데 $\lceil \log_2(k) \rceil T_X$의 시간이 소요되고 나머지는 $\lceil \log_2(k) \rceil T_X$ 시간지연 동안 계속 반씩 감소하므로 XOR 시간지연은 $\lceil \log_2(n+t+1+2^{\lceil \log_2(k) \rceil}) \rceil T_X$이다. t가 $0 \le t \le k-1$인 경우도 이와 같이 계산하면 $\lceil \log_2(n-t-1+$

$2^{\lceil \log_2(k) \rceil})\rceil T_X$ 이므로 $t=-1$일때 최대 시간 지연을 가지며 시간 복잡도는

$$\text{Time Delay} = 1T_A + \lceil \log_2(n + 2^{\lceil \log_2(k) \rceil})\rceil T_X$$

이다. 공간 복잡도의 경우 식 (10.2)가 모두 중복연산이므로 $k(k-1)$개 XOR 게이트가 감소한다. 따라서 정리 5에서 이를 빼면 type II mastrovito 곱셈기의 공간 복잡도는

$$\#AND = n^2 - k^2 = (3/4)n^2,$$
$$\#XOR = (1/2)n^2 + 3k - 1$$

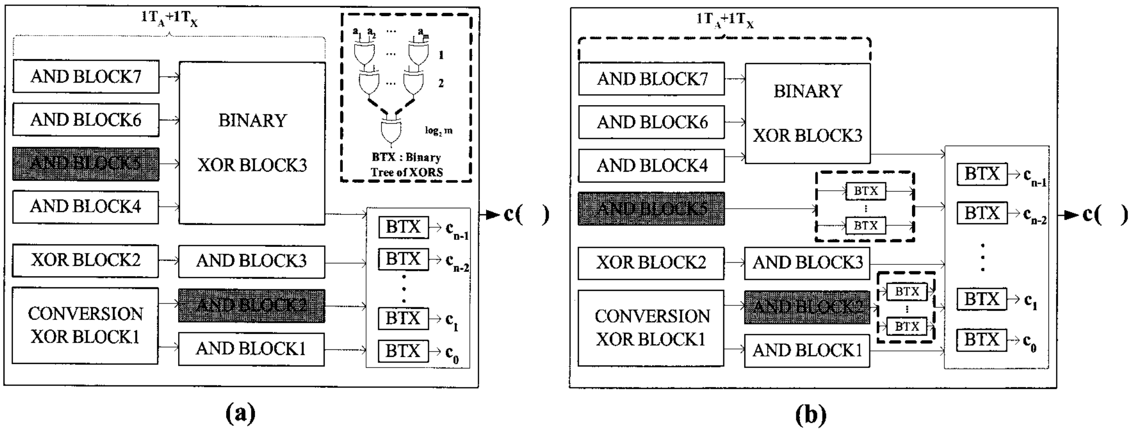
이다. □

IV. 제안하는 비트-병렬 SPB 곱셈기의 하드웨어 구조

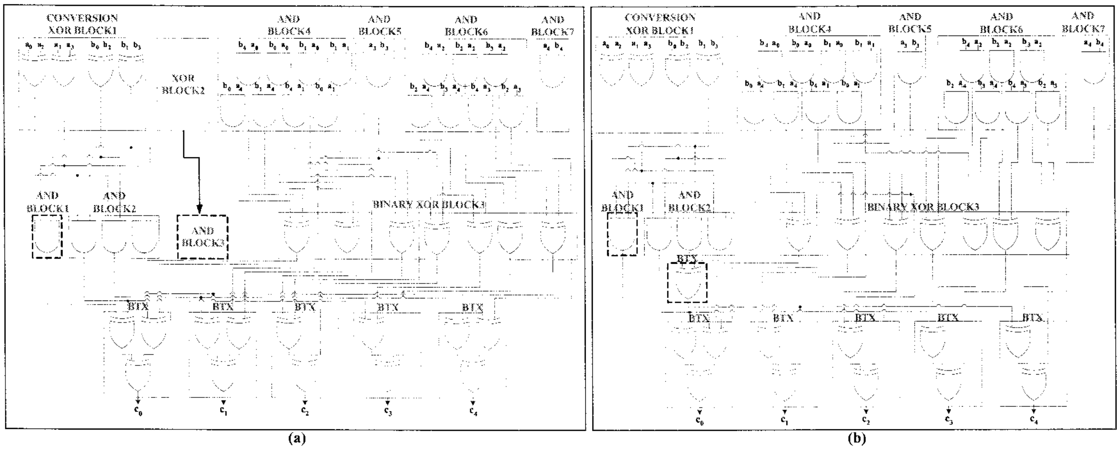
본 소절에서는 제안하는 SPB 곱셈기의 비트-병렬 하드웨어 구조에 대하여 기술한다. 식 (6)은 AND 또는 XOR 연산의 순서에 따라 구분되며, 이에 식 (7)의 간소화 표현까지 적용하여 정리하면 다음과 같다.

- CONVERSION XOR BLOCK1 :
  $\bar{a}_i = a_i + a_{k+i},\ \bar{b}_i = b_i + b_{k+i}$
- XOR BLOCK2 :
  $(\hat{b}_{n+k-1,0} + \hat{b}_{n-1,k}), \cdots, (\hat{b}_{3k+1,0} + \hat{b}_{2k+1,k})$
- BINARY XOR BLOCK3 :
  AND BLOCK3~AND BLOCK7까지 같은 차수의 연산 결과들 사이의 $1T_X$ 연산
- AND BLOCK1 : $\displaystyle\sum_{t=-2k}^{-k-2}\left(\sum_{\substack{i+j=t+3k \\ 0 \le i,j \le k-1}} \bar{a}_i \bar{b}_j\right)\alpha^t$
- AND BLOCK2 : $\displaystyle\sum_{t=-2k}^{-k-1}\left(\sum_{\substack{i+j=t+2k \\ 0 \le i,j \le k-1}} \bar{a}_i \bar{b}_j\right)\alpha^t$
- AND BLOCK3 :
  $$\sum_{t=-k}^{n-3k-2}\left\{\sum_{i=t+3k+1}^{n-1} a_i\left(\hat{b}_{t+n+4k-i,0} + \hat{b}_{t+n+3k-i,k}\right)\right\}\alpha^t$$
- AND BLOCK4 : $\displaystyle\sum_{t=-2k}^{-k-1}\left(\sum_{\substack{i+j=t+n+4k \\ 2k \le i,j \le n+k-1}} \widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t$
  $$+\sum_{t=-k}^{n-3k-2}\left(\sum_{i=n}^{n+k-1} \hat{a}_{j,0}\hat{b}_{t+n+4k-i,0}\right)\alpha^t$$
  $$+\sum_{t=n-3k-1}^{n-2k-2}\left(\sum_{\substack{i+j=t+n+4k \\ 2k \le i,j \le n+k-1}} \widehat{a_{i,0}}\widehat{b_{j,0}}\right)\alpha^t$$
- AND BLOCK5 : $\displaystyle\sum_{t=-2k}^{-k-2}\left(\sum_{\substack{i+j=t+5k \\ k+1 \le i,j \le 2k-1}} a_i b_j\right)\alpha^t$
- AND BLOCK6 :
  $$\sum_{t=n-3k-1}^{n-2k-1}\left(\sum_{\substack{i+j=t+n+3k \\ 2k \le i,j \le n+k-1}} \widehat{a_{jk}}\widehat{b_{j,k}}\right)\alpha^t$$
  $$+\sum_{t=-k}^{-1}\left(\sum_{i=2k}^{t+3k}\left(\hat{a}_{i,k}\hat{b}_{t+n+3k-i,k}\right) + \sum_{i=n}^{n+k+t}\left(\hat{a}_{i,k}\hat{b}_{t+n+3k-i,k}\right)\right)\alpha^t$$
  $$+\sum_{t=0}^{n-3k-2}\left(\sum_{i=t+2k+1}^{t+3k}\left(\hat{a}_{i,k}\hat{b}_{t+n+3k-i,k}\right) + \sum_{i=n}^{n+k-1}\left(\hat{a}_{i,k}\hat{b}_{t+n+3k-i,k}\right)\right)\alpha^t$$
- AND BLOCK7 : $\displaystyle\sum_{t=0}^{n-2k-1}\left(\sum_{\substack{i+j=t+4k \\ 2k \le i,j \le n+k-1}} a_i b_j\right)\alpha^t$

여기서 XOR BLOCK은 3가지로 $1T_X$ 시간 지연



[그림 2] 제안하는 비트-병렬 곱셈기 하드웨어 구조: (a) Type I mastrovito 곱셈기 (b) Type II mastrovito 곱셈기



〔그림 3〕 F_{2^5}에서 제안하는 비트-병렬 mastrovito 곱셈기 ($f(x)=x^5+x^2+1$) : (a) Type I (b) Type II

〔표 3〕 삼항 기약다항식을 위한 SPB 비트-병렬 곱셈기 복잡도 비교

<table>
<tr><td colspan="2">구 분</td><td colspan="2">SPB [8]</td><td colspan="2">제안하는 type I mastrovito 곱셈기</td><td colspan="3">제안하는 type II mastrovito 곱셈기</td></tr>
<tr><td rowspan="11">TIME COMPLEXITY</td><td rowspan="5">$2k<n$</td><td colspan="2" rowspan="5">$1T_A+\lceil \log_2(2n-k-1)\rceil T_X$</td><td rowspan="2">$3k+1\le n$</td><td rowspan="2">$1T_A+\lceil \log_2(2n-2k-1)\rceil T_X$</td><td colspan="2">$k=1$</td><td>$1T_A+\lceil \log_2(2n-2k-1)\rceil T_X$
$=1T_A+\lceil \log_2(2n-3)\rceil T_X$</td></tr>
<tr><td rowspan="2">$k=2^m+1$</td><td>$5k-3\le n$</td><td>$1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$</td></tr>
<tr><td rowspan="3">$3k+1>n$</td><td rowspan="3">$1T_A+\lceil \log_2(n+k)\rceil T_X$</td><td>$5k-3>n$</td><td>$1T_A+\lceil \log_2(n-k+2^{\lceil \log_2(k)\rceil})\rceil T_X$</td></tr>
<tr><td rowspan="2">$k\neq 2^m+1$</td><td>$\{2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1\}\le \mathrm{n}-3\mathrm{k}$</td><td>$1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$</td></tr>
<tr><td>$\{2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1\}>\mathrm{n}-3\mathrm{k}$</td><td>$T_A+\lceil \log_2(3\cdot 2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}+n-1)\rceil T_X$</td></tr>
<tr><td>$2k=n$</td><td colspan="2">$1T_A+\lceil \log_2(n+k)\rceil T_X$</td><td colspan="2">$1T_A+\lceil \log_2(n+k)\rceil T_X$</td><td colspan="3">$1T_A+\lceil \log_2(n+2^{\lceil \log_2(n-k)\rceil})\rceil T_X$</td></tr>
<tr><td rowspan="5">$2k>n$</td><td colspan="2" rowspan="5">$1T_A+\lceil \log_2(n+k)\rceil T_X$</td><td rowspan="2">$2n<3k-1$</td><td rowspan="2">$1T_A+\lceil \log_2(2k-1)\rceil T_X$</td><td colspan="2">$n-k=1$</td><td>$1T_A+\lceil \log_2(2n-3)\rceil T_X$</td></tr>
<tr><td rowspan="2">$n-k=2^m+1$</td><td>$5k+3\ge 4n$</td><td>$1T_A+\lceil \log_2(3k-n+2^{\lceil \log_2(n-k-1)\rceil})\rceil T_X$</td></tr>
<tr><td rowspan="3">$2n\ge 3k-1$</td><td rowspan="3">$1T_A+\lceil \log_2(2n-k)\rceil T_X$</td><td>$5k-3<n$</td><td>$1T_A+\lceil \log_2(k+2^{\lceil \log_2(n-k)\rceil})\rceil T_X$</td></tr>
<tr><td rowspan="2">$n-k\neq 2^m+1$</td><td>$\{2^{\lceil \log_2(n-k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1\}\le 3\mathrm{k}-2\mathrm{n}$</td><td>$1T_A+\lceil \log_2(3k-n+2^{\lceil \log_2(n-k-1)\rceil})\rceil T_X$</td></tr>
<tr><td>$2^{\lceil \log_2(n-k-1)\rceil-1}+$</td><td>$T_A+\lceil \log_2(n+2^{\lceil \log_2(u)\rceil}+3\cdot 2^{\lceil \log_2(n-k-1)\rceil-1}-1)\rceil T_X$</td></tr>
<tr><td rowspan="4">SPACE COMPLEXITY</td><td></td><td>#AND</td><td>#XOR</td><td>#AND</td><td>#XOR</td><td colspan="2">#AND</td><td>#XOR</td></tr>
<tr><td>$2k<n$</td><td>n^2</td><td>n^2-1</td><td>n^2-k^2</td><td>(n^2-1)</td><td colspan="2">n^2-k^2</td><td>$n^2-1-(k-1)^2$</td></tr>
<tr><td>$2k=n$</td><td>n^2</td><td>n^2-k</td><td>n^2-k^2</td><td>$n^2-1-(k^2-2k)$</td><td colspan="2">n^2-k^2</td><td>$n^2-1-(2k^2-3k)$</td></tr>
<tr><td>$2k<n$</td><td>n^2</td><td>n^2-1</td><td>$n^2-(n-k)^2$</td><td>(n^2-1)</td><td colspan="2">$n^2-(n-k)^2$</td><td>$n^2-1-(n-k-1)^2$</td></tr>
</table>

※ $2k<n$일 때 $k-1=2^{\lceil \log_2(k-1)\rceil-1}+u$, $(1\le u\le 2^{\lceil \log_2(k-1)\rceil-1})$

$2k>n$일 때 $n-k-1=2^{\lceil \log_2(n-k-1)\rceil-1}+u$, $(1\le u\le 2^{\lceil \log_2(n-k-1)\rceil-1})$

이 일어나며 AND BLOCK은 7가지로 $1T_A$ 시간 지연이 일어나며, AND 연산만을 BLOCK으로 구분한다. 따라서 이와 같은 방법으로 이전 절에서 설명한 type I과 type II mastrovito 곱셈기를 설계하면 [그림 2]와 같다. [그림 2]에서 (a)는 시간복잡도에 최적화된 곱셈기이고 (b)는 공간복잡도에 최적화된 곱셈기이다. 또한 [그림 2]에서와 같이 상위 $1T_A+1T_X$ 시간 지연 이후의 시간 및 공간복잡도의 trade-off는 k에 의존한다. 이는 AND BLOCK2와 AND BLOCK5가 k에 따라 복잡도가 정의되기 때문이다. [그림 2]의 (a)와 (b)를 $f(x)=x^5+x^2+1$에 의하여 정의되는 F_{2^5}를 예로 설계하면 [그림 3]과 같다. 본 예제에서는 XOR BLOCK2가 존재하지 않는다. 이는 XOR BLOCK2의 XOR 연산은 총 $n-2k-1$개인데 예제에서는 0이 되기 때문이다. 따라서 이에 대응하는 AND BLOCK3도 존재하지 않는다. [그림 3]은 제안하는 type I mastrovito 곱셈기로 XOR BLOCK2와 AND BLOCK3이 없다. 기존의 가장 효율적인 결과인 [8]의 경우 본 예제를 적용하면 시간복잡도의 경우$1T_A+3T_X$이고 공간복잡도의 경우 25개 AND 게이트와 24개 XOR 게이트인 반면에 [그림 3(a)]의 제안하는 type I mastro vito 곱셈기의 경우 시간 복잡도는 1TA+3TX로 같으나 공간 복잡도는 21개 AND 게이트와 24개 XOR 게이트로 감소한다. 또한 [그림 3(b)]의 Type II mastrovito 곱셈기의 경우는 시간 복잡도는 $1T_A+3T_X$는 기존의 경우와 같으나 공간 복잡도의 경우 21개 AND 게이트와 24개 XOR 게이트로 기존 [8]의 결과보다도 작을 뿐만 아니라 제안하는 type I mastrovito 곱셈기보다도 작음을 알 수 있다.

V. 비교 및 결론

본 절에서는 k의 범위($2k<n$, $2k=n$, $2k>n$)에 따라 제안하는 type I과 II mastrovito 곱셈기의 효율성과 기존 결과를 비교하교 결론을 내린다. 삼항 기약다항식 기반에서 SPB를 사용하는 기존 결과 중에서 가장 효율적인 [8]의 결과와 제안하는 두 가지 비트-병렬 mastorvito 곱셈기와 시간 및 공간 복잡도를 비교하면 [표 3]과 같다. [표 3]의 결과에서 알 수 있듯이 공간복잡도의 경우 $n>2k$일 때 [8]에 비하여 type I은 k^2개 AND 게이트가 감소하고 type II는 k^2개 AND와 $(k-1)^2$개 XOR 게이트가 감소함을 알 수 있다. 또한 $n=2k$일 때는 type I은 k^2개 AND 게이트와 (k^2-3k+1)개 XOR 게이트가 감소하고 type II는 k^2개 AND 게이트와 $(2k^2-4k+1)$개 XOR 게이트가 감소함을 알 수 있다. 따라서 제안하는 두 곱셈기 모두 기존의 결과에 비하여 공간 복잡도가 감소한다. AND 게이트와 XOR 게이트의 비용을 같다고 가정할 경우 [8]에 비하여 type I의 경우 최대 12.5%, type II의 경우 최대 25%의 게이트가 감소한다.

Type I의 시간 복잡도는 [표 3]에서와 같이 항상 [8]의 결과보다 작거나 같지만 type II는 n과 k에 따라 달라진다. $100 \le n < 1000$에서 $n \leqq 2k$를 만족하는 모든 삼항 기약다항식(1,335개)으로 시간 복잡도를 조사한 결과 type I의 경우 174개(13%)가 [8]의 결과보다 작으며 1,161개(87%)가 같았으며, type II의 경우 33개(2.5%)가 [8]의 결과보다 작고 1,188개(89%)가 같았으며 114(8.5 %)가 기존 결과보다 $1T_X$ 증가하였다.

참 고 문 헌

[1] 이옥석, 장남수, 김창한, 홍석희, "Equally Spaced 기약다항식 기반의 효율적인 이진체 비트-병렬 곱셈기," 정보보호학회논문지, 18(2), pp. 3-10, 2008년 4월.

[2] 정석원, 이선옥, 김창한, "삼항 기약다항식을 이용한 효율적인 비트-병렬 구조의 곱셈기," 정보보호학회논문지, 13(5), pp. 179-187, 2003년 10월.

[3] 정석원, 윤중철, 이선옥, "$GF(2^n)$에서의 직렬-병렬 곱셈기 구조," 정보보호학회논문지, 13(3), pp. 27-34, 2003년 6월.

[4] A. Reyhani-Masoleh and M.A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^n)$," IEEE Trans. Computer s., vol. 53, no. 8, pp. 945-959, Aug. 2004.

[5] H. Fan and Y. Dai, "Fast bit parallel $GF(2^n)$ multiplier for all trinomials," IEEE Trans. Computers., vol. 54, no. 4, pp. 485-490, Apr. 2005.

[6] C. Negre, "Efficient parallel multiplier in shifted polynomial basis," Journal of Systems Architecture, vol. 53, no. 2-3, pp. 109 - 116, Feb. 2007.

[7] H. Fan and M.A. Hasan, "Relationshi p

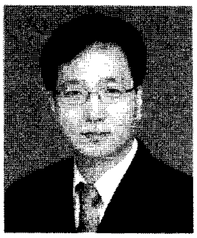
between $GF(2^n)$ Montgomery and shi fted polynomial basis multiplication al gorithms," IEEE Tran s. Computers., vol. 55, no. 9, pp. 1202-1206, Sep. 2006.

[8] H. Fan and M.A. Hasan, "Fast Bit Parallel Shifted Polynomial Basis Multipliers in $GF(2^n)$," IEEE Trans. Circuits & Systems-I, vol. 53, no. 12, pp. 2606-2615, Dec. 2006.

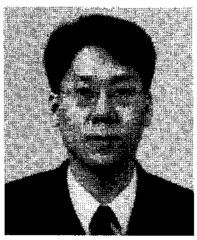
〈著者紹介〉

장 남 수 (Nam Su Chang) 학생회원
2002년 2월: 서울 시립대학교 수학과 이학사
2004년 8월: 고려대학교 정보보호 대학원 공학석사
2005년 2월~현재: 고려대학교 정보경영공학전문대학원 박사과정
〈관심분야〉 암호칩 설계 기술, 부채널 공격, 공개키 암호 알고리즘, 공개키 암호 암호분석

김 창 한 (Chang Han Kim) 정회원
1985년 2월: 고려대학교 수학과 학사
1987년 2월: 고려대학교 수학과 석사
1992년 2월: 고려대학교 수학과 박사
1992년 3월~현재: 세명대학교 정보통신학부 교수
〈주관심분야 : 정수론, 공개키암호, 암호프로토콜〉



홍 석 희 (Seokhie Hong) 정회원
1995년: 고려대학교 수학과 학사
1997년: 고려대학교 수학과 석사
2001년: 고려대학교 수학과 박사
1999년 8월~2004년 2월: (주)시큐리티 테크놀로지스 선임연구원
2003년 3월~2004년 2월: 고려대학교 시간강사
2004년 4월~2005년 2월: K.U. Leuven 박사후연구원
2005년 3월~현재: 고려대학교 정보경영전문대학원 부교수
〈관심분야〉 대칭키 암호 알고리즘, 공개키 암호 알고리즘, 포렌식



박 영 호 (Young-Ho Park) 정회원
1990년 2월: 고려대학교 수학과 학사
1993년 2월: 고려대학교 수학과 석사
1997년 2월: 고려대학교 수학과 박사
2002년 3월~현재: 세종 사이버대학교 부교수
〈주관심분야 : 정수론, 공개키암호, 암호프로토콜〉