情報保護學會論文誌
第20卷 第5號, 2010. 10

삼항 기약다항식 기반의 저면적 Shifted Polynomial Basis 비트-병렬 곱셈기*

장 남 수,[1†] 김 창 한[2‡]
[1]세종사이버대학교, [2]세명대학교

Low Space Complexity Bit-Parallel Shifted Polynomial Basis Multipliers using Irreducible Trinomials*

Nam Su Chang,[1†] Chang Han Kim[2‡]
[1]Sejong Cyber University, [2]Semyung University

요 약

최근 Fan과 Dai는 이진체 곱셈기의 효율성을 개선하기 위하여 Shifted Polynomial Basis(SPB)를 제안하고 이를 이용한 non-pipeline 비트-병렬 곱셈기를 제안하였다. SPB는 PB에 $\{1,\alpha,\cdots,\alpha^{n-1}\}$에 α^{-v}를 곱한 것으로, 이 둘 사이는 매우 적은 비용으로 쉽게 기저 변환이 된다. 이후 삼항 기약다항식 $f(x)=x^n+x^k+1$을 사용하여 Modified Shifted Polynomial Basis(MSPB) 기반의 SPB 비트-병렬 Mastrovito type I과 type II 곱셈기가 제안되었다. 본 논문에서는 SPB를 이용한 비트-병렬 곱셈기를 제안한다. $n\neq 2k$ 일 때 제안하는 곱셈기 구조는 기존의 모든 SPB 곱셈기와 비교하여 효율적인 공간 복잡도를 가진다. 또한, 기존의 가장 작은 공간 복잡도를 가지는 곱셈기와 비교하여 $1\leq k\leq (n+1)/3$ 인 경우 항상 효율적이다. 또한, $(n+2)/3\leq k<n/2$ 인 경우에도 일부 경우를 제외하고 기존 결과보다 항상 작은 공간 복잡도를 가진다.

ABSTRACT

Recently, Fan and Dai introduced a Shifted Polynomial Basis and construct a non-pipeline bit-parallel multiplier for F_{2^n}. As the name implies, the SPB is obtained by multiplying the polynomial basis $1,\alpha,\cdots,\alpha^{n-1}$ by α^{-v}. Therefore, it is easy to transform the elements PB and SPB representations. After, based on the Modified Shifted Polynomial Basis(MSPB), SPB bit-parallel Mastrovito type I and type II multipliers for all irreducible trinomials are presented. In this paper, we present a bit-parallel architecture to multiply in SPB. This multiplier have a space complexity efficient than all previously presented architecture when $n\neq 2k$. The proposed multiplier has more efficient space complexity than the best-result when $1\leq k\leq (n+1)/3$. Also, when $(n+2)/3\leq k<n/2$, the proposed multiplier has more efficient space complexity than the best-result except for some cases.

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

접수일: 2010년 4월 29일; 채택일: 2010년 8월 23일
* 본 연구는 2009년도 세명대학교 교내학술연구비 지원에 의해 수행된 연구임

† 주저자, nschang@sjcu.ac.kr
‡ 교신저자, chkim@semyung.ac.kr

I. 서 론

임베디드 관련 정보보호시스템 구축에 가장 중요한 요소 중 하나가 유한체의 연산 시스템 구현이다. 특히, 유한체 연산 중 곱셈 연산은 가장 높은 비중을 차지하며, 곱셈 연산의 효율성은 기저와 기약다항식 등에 의하여 달라진다. 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 본 논문과 같이 이진체 환경에서는 이항 기약다항식(Binomial)이 존재하지 않으므로 삼항 기약다항식(Trinomial)이 가장 효율적이다.

F_{2^n} 연산은 F_2 연산의 덧셈(뺄셈) 연산과 곱셈 연산에 의하여 계산되며, 덧셈과 곱셈 연산은 비트 단위의 AND와 XOR 연산에 의하여 표현된다. 이는 하드웨어에서 2-입력 1-출력의 AND와 XOR 게이트이다. 또한 하드웨어에서 연산의 효율성은 시간 및 공간 복잡도에 의하여 정의되므로 본 논문에서는 공간복잡도 계산을 위하여 AND와 XOR 게이트 수를 각각 N_A와 N_X로 표기하고, 시간복잡도를 위하여 AND와 XOR 게이트 시간지연(time delay)과 전체 시간지연을 각각 T_A, T_X와 T_C로 표기한다.

[2]에서 Shifted Polynomial Basis(SPB)를 처음 제안하고, 이를 이용하여 파이프라인(pipeline)이 아닌 비트-병렬 곱셈기를 제안하였다. SPB는 다항식 기저(PB:Polynomial Basis)를 변형한 형태로 둘 사이의 기저 변환이 매우 간단하게 수행된다. 따라서 기존의 SPB 비트-병렬 곱셈기[1,2,3,4,8]는 기존의 PB 비트-병렬 곱셈기[5,6,7,9,10]와 많이 비교되었다. 이후 [1]에서 Modified Shifted Polynomial Basis(MSPB)를 제안하였고, 이를 이용하여 기존 결과들보다 시간 및 공간 복잡도에서 효율적인 type I, II SPB 곱셈기를 제안하였다.

본 논문에서는 [1]의 결과와 같이 MSPB를 이용하여 새로운 SPB 비트-병렬 곱셈기를 제안한다. 삼항 기약다항식 기반에서 SPB를 사용하는 기존 결과에 비하여 제안하는 비트-병렬 Mastorvito 곱셈기는 작은 공간복잡도를 가진다. 기존의 가장 작은 공간복잡도를 가지는 [1]의 type II와 비교하여 $1 \le k \le (n+1)/3$인 경우 항상 AND 및 XOR 게이트가 작으며 $(n+2)/3 \le k < n/2$인 경우에도 k가 $n/2$에 매우 근사하는 경우를 제외하고 기존 결과보다 항상 작은 공간복잡도를 가진다. 따라서 제안하는 곱셈식의 공간복잡도가 비효율적인 경우 제안하는 방법은 [1]의 type II 구조를 사용한다. 시간복잡도의 경우 $n > 2k$를 만족하는 차수가 $1 \le n < 1{,}000$인 모든 삼항기약다항식(1,491개)에 대해 [1]의 type II와 비교하면, 942개(63.18%)가 같으며 549개(36.82 %)가 $1T_X$ 증가한다. [1]의 Type II 곱셈기의 공간복잡도가 낮아 제안하는 곱셈기를 사용하지 않고 [1]의 type II 곱셈기를 사용하는 경우까지 모두 고려하면 실제 399개(26.76%)만 $1T_X$ 증가한다.

본 논문의 구성은 다음과 같다. 2절에서는 [4]에서 제안된 SPB 비트-병렬 곱셈을 소개하고 이를 개선한 [1]의 삼항 기약다항식을 위한 SPB 비트-병렬 곱셈과 복잡도를 소개한다. 3절에서는 제안하는 MSPB 기반의 저면적 SPB 비트-병렬 곱셈기와 그 시간 및 공간 복잡도를 기술한다. 4절에서는 제안하는 곱셈기와 기존 결과의 복잡도를 비교하고 결론을 내린다.

II. 기존의 SPB 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 다항식 기저에 의하여 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:

- 다항식 곱셈:

$$s(\alpha) = a(\alpha)b(\alpha) = \sum_{i=0}^{2n-2} s_i \alpha^i, \; s_i = \sum_{\substack{u+v=i \\ 0 \le u,v \le n-1}} a_u b_v$$

- $f(x)$에 의한 모듈러 감산 연산:

$$c(\alpha) \equiv s(\alpha) \bmod f(x) \equiv \sum_{i=0}^{n-1} c_i \alpha^i, \; c_i \in F_2.$$

곱셈 연산의 복잡도는 위의 두 과정에 의하여 결정된다. 본 논문은 삼항 기약다항식 $f(x) = x^n + x^k + 1$에 대해서만 논하도록 한다.

2.1 [4]에서 제안된 SPB 곱셈

최근 [2]에서 처음으로 F_{2^n}의 원소를 SPB로 표현하는 방법이 제안되었다. SPB는 PB의 변형된 형태

로 다음과 같이 정의된다.

정의 1. v를 임의의 정수라 하고 $\Gamma=\{\alpha^i \mid 0 \le i \le n-1\}$를 F_{2^n}의 다항식 기저라 할 때, 순서집합 $\alpha^{-v}\Gamma=\{\alpha^{i-v} \mid 0 \le i \le n-1\}$를 Γ에 대한 SPB (Shifted Polynomial Basis)라 한다.

비트-병렬 곱셈기에서 SPB를 사용할 경우, [4]에서 소개된 바와 같이 v는 k 또는 $k-1$에서 최적의 값을 가진다. SPB에 의하여 임의의 두 원소 $a(\alpha)$, $b(\alpha)$는

$$a(\alpha)=[a_0,a_1,\cdots,a_{n-1}]=\alpha^{-v}\sum_{i=0}^{n-1}a_i\alpha^i,$$

$$b(\alpha)=[b_0,b_1,\cdots,b_{n-1}]=\alpha^{-v}\sum_{i=0}^{n-1}b_i\alpha^i$$

이다. 따라서 곱 $c(\alpha)=a(\alpha)b(\alpha) \bmod f(x)$는 다음과 같다 [4]:

$$\begin{aligned}c(\alpha)=a(\alpha)b(\alpha) \bmod f(\alpha)&=\sum_{t=-v}^{n-v-1}c_{v+t}\alpha^t\\&=\sum_{t=-v}^{n-2v-1}\left(\sum_{i=0}^{2v+t}a_ib_{2v+t-i}\right)\alpha^t+\sum_{t=-v}^{-1}\left(\sum_{i=0}^{v+t}a_ib_{v+t-i}\right)\alpha^t\\&+\sum_{t=n-2v}^{n-v-1}\left(\sum_{i=2v+t-n+1}^{n-1}a_ib_{2v+t-i}\right)\alpha^t\\&+\sum_{t=n-2v}^{n-v-1}\left(\sum_{i=0}^{2v+t-n}a_ib_{2v+t-n-i}\right)\alpha^t\\&+\sum_{t=0}^{n-v-2}\left(\sum_{i=v+t+1}^{n-1}a_ib_{v+t+n-i}\right)\alpha^t\\&+\sum_{t=-v}^{n-2v-2}\left(\sum_{i=2v+t+1}^{n-1}a_ib_{2v+t+n-i}\right)\alpha^t.\end{aligned}\quad(1)$$

식 (1)에서 $\alpha^t(-v \le t \le n-v-1)$의 계수를 비교하여 연산량 감소를 위하여 공통 a_i를 이용하여 간소화 한다[4]. 예를 들어, $t=n-2v-1$인 경우

$$\begin{aligned}c_{n-v-1}&=\sum_{i=0}^{n-v-1}a_i(b_{n-v-i-1}+b_{n-i-1})\\&+\sum_{i=n-v}^{n-1}a_ib_{n-i-1}\end{aligned}$$

이다. 이때 병렬 계산을 위하여 $\lceil v/2 \rceil$ 개의 a_ib_{n-i-1} 항과 $n-v$ 개의 $a_i(b_{n-v-i-1}+b_{n-i-1})$ 항은 XOR 트리에 의하여 계산된다. 따라서 삼항 기약다항식 $f(x)=x^n+x^k+1$에 대하여 SPB 비트-병렬 곱셈기는 다음과 같은 복잡도를 가진다.

$$N_A=n^2,$$
$$N_X=\begin{cases}n^2-1, & n\neq 2k\\ n^2-n/2, & n=2k,\end{cases}$$

$$T_C=\begin{cases}T_A+\lceil \log_2(n+v) \rceil T_X, & n \le 2v\\ T_A+\lceil \log_2(2n-v-1) \rceil T_X, & n>2k.\end{cases}$$

이때 v는 k 또는 $k-1$이다.

2.2 [1]에서 제안된 SPB 곱셈

[1]에서 새로운 MSPB 기저와 v값을 이용하여 비트-병렬 SPB 곱셈기를 제안하였다. 제안된 곱셈기는 다음 정의를 기반으로 한다.

정의 2. v를 정수라 하고 $\alpha^{-v}\Gamma=\{\alpha^{i-v} \mid 0 \le i \le n-1\}$를 F_{2^n}의 SPB라 할 때, 만약

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

$$\delta_1=\begin{cases}0, & n \ge 2k\\ -n+k, & n<2k,\end{cases}\quad \delta_2=\begin{cases}2k, & n>2k\\ k, & n \le 2k,\end{cases}$$

이면, 순서집합(ordered set) $\Delta=\{\alpha^{i-v} \mid \delta_1 \le i \le \delta_1+k-1, \delta_2 \le i \le \delta_2+n-k-1\}$를 $\alpha^{-v}\Gamma$에 대한 삼항 기약다항식 $f(x)=x^n+x^k+1$를 위한 MSPB(Modified Shifted Polynomial Basis)라 한다.

[1]의 식 (6)을 다시 정리하면 다음과 같다.

$$\begin{aligned}c(\alpha)=&\underbrace{\sum_{t=-2k}^{-k-1}\left(\sum_{i=0}^{t+2k}(a_i+a_{i+k})(b_{t-i+2k}+b_{t-i+3k})\right)\alpha^t}_{(a1)}\\&+\underbrace{\sum_{t=-2k}^{-k-2}\left(\sum_{i=t+2k+1}^{k-1}(a_i+a_{i+k})(b_{t-i+3k}+b_{t-i+4k})\right)\alpha^t}_{(b)}\\&+\underbrace{\sum_{t=-2k}^{-k-2}\left(\sum_{i=t+n+2k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+2n+3k,k}\right)\alpha^t}_{(c1)}\\&+\underbrace{\sum_{t=-2k}^{-k-1}\left(\sum_{i=2k}^{t+n+2k}\tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0}\right)\alpha^t}_{(d)}\\&+\underbrace{\sum_{t=-k}^{-2}\left(\sum_{i=t+n+k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+2n+2k,k}\right)\alpha^t}_{(c2)}\\&+\underbrace{\sum_{t=-k}^{-1}\left(\sum_{i=2k}^{t+n+k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\right)\alpha^t}_{(e)}\\&+\underbrace{\sum_{t=-k}^{n-3k-2}\left(\sum_{i=t+3k+1}^{n+k-1}\tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0}\right)\alpha^t}_{(f)}\\&+\underbrace{\sum_{t=n-3k-1}^{n-2k-2}\left(\sum_{i=t+3k+1}^{n+k-1}\tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0}\right)\alpha^t}_{(g)}\\&+\underbrace{\sum_{t=0}^{n-2k-1}\left(\sum_{i=2k}^{t+2k}\tilde{a}_{i,k}\tilde{b}_{t-i+4k,k}\right)\alpha^t}_{(h)}\\&+\underbrace{\sum_{t=0}^{n-2k-1}\left(\sum_{i=t+2k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\right)\alpha^t}_{(i)}\\&+\underbrace{\sum_{t=n-3k}^{n-2k-1}\left(\sum_{i=0}^{t-n+3k}(a_i+a_{i+k})(b_{t-i-n+3k}+b_{t-i-n+4k})\right)\alpha^t}_{(a2)}\end{aligned}\quad(2)$$

〔표 1〕 모든 경우에 대한 계수 c_{t+2k}의 식

구 분	t	c_{t+2k}
1) $n-3k \geq 2$ ⇒$n \geq 3k+2$인 경우	$-k \leq t \leq -2$ ($k=1$인 경우는 제외된다.)	(c2)+(e)+(f)
	$t=-1$	(e)+(f)
	$0 \leq t \leq n-3k-2$	(f)+(i)+(h)
	$t=n-3k-1$	(g)+(h)+(i)
	$n-3k \leq t \leq n-2k-2$ ($n=2k+1$인 경우는 제외된다.)	(a2)+(g)+(h)+(i)
2) $n-3k=1$ ⇒$n=3k+1$인 경우	$-k \leq t \leq -2$	(c2)+(e)+(f)
	$t=-1$	(e)+(f)
	$t=n-3k-1$	(g)+(h)+(i)
	$n-3k \leq t \leq n-2k-2$	(a2)+(g)+(h)+(i)
3) $n-3k=0$ ⇒$n=3k$인 경우	$-k \leq t \leq n-3k-2$	(c2)+(e)+(f)
	$t=-1$	(e)+(g)
	$0 \leq t \leq n-2k-2$	(a2)+(g)+(h)+(i)
4) $n-3k=-1$ ⇒$n=3k-1$인 경우	$-k \leq t \leq n-3k-2$	(c2)+(e)+(f)
	$t=n-3k-1$	(c2)+(e)+(g)
	$t=-1$	(a2)+(e)+(g)
	$0 \leq t \leq n-2k-2$	(a2)+(g)+(h)+(i)
5) $n-3k \leq -2$ ⇒$2k<n \leq 3k-2$인 경우	$-k \leq t \leq n-3k-2$ ($n=2k+1$인 경우는 제외된다.)	(c2)+(e)+(f)
	$t=n-3k-1$	(c2)+(e)+(g)
	$n-3k \leq t \leq -2$	(a2)+(c2)+(e)+(g)
	$t=-1$	(a2)+(e)+(g)
	$0 \leq t \leq n-2k-2$ ($n=2k+1$인 경우는 제외된다.)	(a2)+(g)+(h)+(i)

이때 $\tilde{a}_{i,j}$는

$$\tilde{a}_{i,j}=\begin{cases}a_{i+j}, & i<0\\ a_i, & 0 \leq i<n\\ a_{i+j-n}, & i \geq n\end{cases},\quad \tilde{b}_{i,j}=\begin{cases}b_{i+j}, & i<0\\ b_i, & 0 \leq i<n\\ b_{i+j-n}, & i \geq n\end{cases}$$

이다. 또한 [1]에서 식 (2)의 (f)+((e)+(i))는 $-k \leq t \leq n-3k-2$사이에서

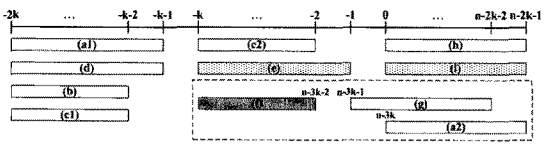
$$a_{t+3k+1}(\tilde{b}_{n+k-1}+\tilde{b}_{n-1})+\cdots+a_{n-1}(\tilde{b}_{t+4k+1}+\tilde{b}_{t+3k+1})$$

로 간소화됨을 보였다. 식 (2)에서 알 수 있듯이 $n-3k$ 값의 범위에 따라 α^t항의 계수값이 일부 달라지며 그림 1과 같다.

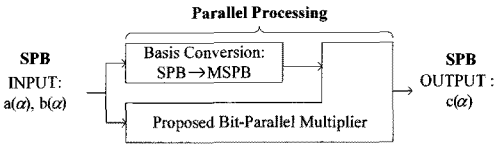
〔그림 1〕 에서와 같이 식 (2)에서 항 α^t의 계수들은 k의 조건에 따라 달라진다. t의 범위가

- $-2k \leq t \leq -k-2$:
  (a1)+(b)+(c1)+(d) ($k \neq 1$)
- $t=-k-1$: (a1)+(d)
- $t=n-2k-1$: (a2)+(h)+(i)

인 경우를 제외하고, $n>2k$인 경우의 k를 범위에 따라



〔그림 1〕 ($n>2k$) 식 (2)의 도식도



〔그림 2〕 [1]의 SPB 비트-병렬 곱셈기

항을 구분하여 계수를 정리하면 〔표 1〕과 같으며, 〔표 1〕에서 밑줄이 있는 부분은 간소화되는 부분이다.

구분된 항에 따라 계수 계산과 시간지연을 정리하면 [1]의 〔표 1〕과 같다. [1]의 SPB 곱셈기는 제안된 MSPB와 Karatsuba-Ofman(KO) 방법의 곱셈을 효율적으로 적용하여 KO 곱셈시 발생하는 시간복잡도 증가의 trade-off를 제거하였다. 또한 그림 2와 같이 비트-병렬 곱셈기는 기저변환, 다항식 곱셈, 모듈러 감산이 병렬로 계산된다. [1]에서는 시간복잡도를 최적화한 type I과 공간복잡도를 최적화한 type II를 제안하였다.

III. 제안하는 저면적 SPB 비트-병렬 곱셈기

본 절에서는 [1]에서 제안된 SPB 비트-병렬 곱셈기를 변형하여 [1]의 type II 보다 공간 복잡도가 효율적인 삼항 기약다항식 $f(x)=x^n+x^k+1$ 기반의 SPB 비트-병렬 곱셈기를 제안한다. 제안하는 방법은

n과 k의 값에 따라 각 항 α^t의 계수가 달라진다. 따라서 다음과 같이 세 가지 경우로 구분한다.

1. $n-3k \geq 0$인 경우
2. $n-3k=-1$인 경우
3. $n-3k \leq -2$인 경우

제안하는 비트-병렬 곱셈은 식 (2)에서 t가 $n-3k \leq t \leq n-2k-2$사이인 항에서 공간 복잡도를 효율적으로 줄인다. 만약 $k \neq 1$이고 $n > 2k$이면, (2)에

$$\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+3k+1}^{k-1}(a_i+a_{i+k})(b_{t-i-n+4k}+b_{t-i-n+5k})\right)\alpha^t \tag{3}$$

를 두 번 더하고, 이를 각각 (j1), (j2)라 하자.(F_{2^n} 이므로 (j1)+(j2)는 0이다.) 그러면 식 (2)는

$$c(\alpha)=(a1)+(b)+(c1)+(c2)+(d)+((f)+((e)+(i)))+(g)+(h)+(a2)+(j1)+(j2) \tag{4}$$

이다. 이때 (j2)를 풀어쓰면

$$\begin{aligned}
&\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+3k+1}^{k-1}(a_i+a_{i+k})(b_{t-i-n+4k}+b_{t-i-n+5k})\right)\alpha^t \\
&=\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+4k}\right. \\
&\quad+\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+5k}+\sum_{i=t-n+3k+1}^{k-1}a_{i+k}b_{t-i-n+4k} \\
&\quad\left.+\sum_{i=t-n+3k+1}^{k-1}a_{i+k}b_{t-i-n+5k}\right)\alpha^t \\
&=\sum_{t=n-3k}^{n-2k-2}\left(\underbrace{\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+4k}}_{(j2_1)}+\underbrace{\sum_{i=2k-1}^{t-n+4k+1}a_{t-i-n+5k}b_i}_{(j2_2)}\right. \\
&\quad\left.+\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_i b_{t-i-n+5k}}_{(j2_3)}+\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_i b_{t-i-n+6k}}_{(j2_4)}\right)\alpha^t
\end{aligned} \tag{5}$$

이며, $(j2_4)$는

$$\begin{aligned}
&\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+4k+1}^{2k-1}a_i b_{t-i-n+6k}\right)\alpha^t \\
&=\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+4k+1}^{2k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+4k,k}\right)\alpha^t
\end{aligned}$$

이므로 (a2)+(j1)+$(j2_4)$은

$$\begin{aligned}
&\underbrace{\sum_{t=n-3k}^{n-2k-1}\left(\sum_{i=0}^{t-n+3k}(a_i+a_{i+k})(b_{t-i-n+3k}+b_{t-i-n+4k})\right)\alpha^t}_{(a2)} \\
&+\underbrace{\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+3k+1}^{k-1}(a_i+a_{i+k})(b_{t-i-n+3k}+b_{t-i-n+4k})\right)\alpha^t}_{(j1)} \\
&+\underbrace{\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+4k+1}^{2k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+4k,k}\right)\alpha^t}_{(j2_4)} \\
&=\alpha^{n-k}\left[\sum_{t=-2k}^{-k-1}\left(\sum_{i=0}^{t+2k}(a_i+a_{i+k})(b_{t-i+2k}+b_{t-i+3k})\right)\alpha^t\right. \\
&\quad+\sum_{t=-2k}^{-k-2}\left(\sum_{i=t+2k+1}^{k-1}(a_i+a_{i+k})(b_{t-i+3k}+b_{t-i+4k})\right)\alpha^t \\
&\quad\left.+\sum_{t=-2k}^{-k-2}\left(\sum_{i=t+n+2k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+2n+3k,k}\right)\alpha^t\right]
\end{aligned} \tag{6}$$

이므로 식 (6)은 식 (2)의 (a1)+(b)+(c1)에 α^{n-k}를 곱한 값과 같다. 따라서 이 부분은 실제 계산되지 않는다. 이를 이용하여 식 (4)를 정리하면 다음과 같다.

$$\begin{aligned}
c(\alpha)&=(a1)+(b)+(c1)+(c2)+(d)+((f)+((e)+(i)))+(g)+(h)+(a2)+(j1)+(j2) \\
&=[(a1)+(b)+(c1)]+(c2)+(d)+((f)+((e)+(i)))+(g)(h)+[(a2)+(j1)+(j2_4)]+(j2_1)+(j2_2)+(j2_3) \\
&=[(a1)+(b)+(c1)]+[(a1)+(b)+(c1)]\alpha^{n-k}+(c2)+(d)+((f)+((e)+(i)))+(g)+(h)+(j2_1)+(j2_2)+(j2_3)
\end{aligned} \tag{7}$$

따라서 이후 소절에서 구분된 세 가지 경우에 따라 달라지는 항 α^t의 계수에 대하여 기술한다.

3.1 $n-3k \leq t \leq n-2k-2$차수 항의 곱셈식

본 소절에서는 식 (7)에서 t가 $n-3k \leq t \leq n-2k-2$사이 차수인 항 (g)+(h)+(i)+$(j2_1)$+$(j2_2)$+$(j2_3)$를 고려한다.

3.1.1 $n-3k \geq 0$인 경우

식 (2)에서 (g)는

$$\begin{aligned}
&\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t+3k+1}^{n+k-1}\tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0}\right)\alpha^t \\
&=\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+4k}\right)\alpha^t
\end{aligned} \tag{8}$$

이고, t가 $n-3k \leq t \leq n-2k-2$일 때 (i)를 풀어쓰면

$$\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t+2k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\right)\alpha^t$$
$$=\sum_{t=n-3k}^{n-2k-2}\left(\sum_{i=t+2k+1}^{n-1}\tilde{a}_{i,k}\tilde{b}_{t-i-n+3k,k}+\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}+\sum_{i=t+3k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\right)\alpha^t$$
$$=\sum_{t=n-3k}^{n-2k-2}\Bigg(\underbrace{\sum_{i=t-2n+3k+1}^{-n+k-1}\tilde{a}_{t-i-n+3k,k}\tilde{b}_{i,k}}_{(i_1)}+\underbrace{\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}}_{(i_2)}+\underbrace{\sum_{i=t+3k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}}_{(i_3)}\Bigg)\alpha^t \tag{9}$$

이다. 따라서 식 (5), (8)과 (9)로 부터 (g)+(i) + $(j2_1)+(j2_2)+(j2_3)$는 다음과 같다.

$$\sum_{t=n-3k}^{n-2k-2}\Bigg(\underbrace{\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+4k}}_{(g)}\Bigg)\alpha^t$$
$$+\sum_{t=n-3k}^{n-2k-2}\Bigg(\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_{t-i+4k}b_i}_{(i_1)}+\underbrace{\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}}_{(i_2)}+\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_i b_{t-i+4k}}_{(i_3)}\Bigg)\alpha^t$$
$$+\sum_{t=n-3k}^{n-2k-2}\Bigg(\underbrace{\sum_{i=t-n+3k+1}^{k-1}a_i b_{t-i-n+4k}}_{(j2_1)}+\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_{t-i-n+5k}b_i}_{(j2_2)}+\underbrace{\sum_{i=t-n+4k+1}^{2k-1}a_i b_{t-i-n+5k}}_{(j2_3)}\Bigg)\alpha^t$$
$$+\sum_{t=n-3k}^{n-2k-2}\Bigg(\sum_{i=t-n+4k+1}^{2k-1}(a_{t-i-n+5k}+a_{t-i+4k})b_i+\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}+\sum_{i=t-n+4k+1}^{2k-1}a_i(b_{t-i-n+5k}+b_{t-i+4k})\Bigg)\alpha^t \tag{10}$$

식 (10)의 결과를 (k)라 하자. 식 (10)에서 (g)와 $(j2_1)$은 같은 값으로 소거되고, (i_1)과 $(j2_2)$, (i_3)와 $(j2_3)$는 각각 b_i와 a_i에 의하여 간소화된다. 또한, [1]에서 소개된 바와 같이 ((f)+((e)+(i)))는 $(\tilde{b}_{t-i+n+3k,k}+\tilde{b}_{t-i+n+4k,0})$에 의하여 간소화된다. $t=-k$일 때 계산되는 값

$$(\tilde{b}_{n-1,k}+\tilde{b}_{n+k-1,0}),\cdots,(\tilde{b}_{n-k+1,k}+\tilde{b}_{n+1,0}),$$
$$(\tilde{b}_{n-k,k}+\tilde{b}_{n,0}),(\tilde{b}_{n-k-1,k}+\tilde{b}_{n-1,0}),\cdots,(\tilde{b}_{2k+1,k}+\tilde{b}_{3k+1,0}) \tag{11}$$

는 가정 $n\geq 3k$에 의하여

$$(b_{n-1}+b_{k-1}),\cdots,(b_{n-k+1}+b_1)$$
$$,(b_{n-k}+b_0),(b_{n-k-1}+b_{n-1}),\cdots,(b_{2k+1}+b_{3k+1}) \tag{12}$$

이다. 결국 $-k+1\leq t\leq n-3k-2$에서 계산되는 모든 $(\tilde{b}_{t-i+n+3k,k}+\tilde{b}_{t-i+n+4k,0})$는 $t=-k$일 때 모두 계산되므로 실제 연산은 $n-2k-1$번의 XOR 연산만 수행된다. 식 (k)의 $(a_{t-i-n+5k}+a_{t-i+4k})$와 $(b_{t-i-n+5k}+b_{t-i+4k})$도 마찬가지로 $t=n-3k$일 때 계산되는

$$(a_{n-1}+a_{k-1}),(a_{n-2}+a_{k-2}),\cdots,(a_{n-k+1}+a_1),$$
$$(b_{n-1}+b_{k-1}),(b_{n-2}+b_{k-2}),\cdots,(b_{n-k+1}+b_1) \tag{13}$$

에 의하여 $n-3k+1\leq t\leq n-2k-2$에서 계산되는 모든 $(a_{t-i-n+5k}+a_{t-i+4k})$와 $(b_{t-i-n+5k}+b_{t-i+4k})$는 계산되었으므로 실제 연산은 $2(k-1)$번의 XOR 연산만 수행된다. 그러나 식 (12)와 (13)의 $(\tilde{b}_{t-i+n+3k,k}+\tilde{b}_{t-i+n+4k,0})$와 $(b_{t-i-n+5k}+b_{t-i+4k})$는 일부 같은 값을 가진다. 가정에 의하여 $n+1\geq 3k+1\Rightarrow n\geq 3k$이므로, 식 (12)에서 (k)의 모든 $(b_{t-i-n+5k}+b_{t-i+4k})$이 계산되고, 식 (13)은 $k-1$번의 $(a_{t-i-n+5k}+a_{t-i+4k})$의 XOR 연산으로 계산된다. 따라서 $n-3k\geq 0$인 경우 추가적으로 소요되는 XOR 연산량은 $N_X=n-k-2$이다.

3.1.2 $n-3k=-1$또는 $n-3k\leq -2$인 경우

본 절의 서두에서 소개한 바와 같이 $n-3k=-1$ 또는 $n-3k\leq -2$인 경우에 t가 $n-3k\leq t\leq n-2k-2$ 사이일 때 다음 항이 고려된다.

- $n-3k\leq t\leq -2$: (c2)+(e)+(g)+$(j2_1)$+$(j2_2)$+$(j2_3)$ ($n-3k=-1$인 경우 존재하지 않는다.)
- $t=-1$: (e)+(g)+$(j2_1)$+$(j2_2)$+$(j2_3)$
- $0\leq t\leq n-2k-2$: (g)+(h)+(i)+$(j2_1)$+$(j2_2)$+$(j2_3)$ ($n=2k+1$인 경우는 제외된다.)

이때, 이전 소절에서와 같이 (g)와 $(j2_1)$은 같은 값이므로 소거된다. t가 $0\leq t\leq n-2k-2$ 사이일 때의 항의 계수는 식 (10)에 의하여

$$\sum_{t=0}^{n-2k-2}\Bigg(\sum_{i=t-n+4k+1}^{2k-1}(a_{t-i-n+5k}+a_{t-i+4k})b_i + \sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k} + \sum_{i=t-n+4k+1}^{2k-1}a_i(b_{t-i-n+5k}+b_{t-i+4k})\Bigg)\alpha^t$$

이다. $n-3k \le t \le -2$인 항과 $t=-1$인 항을 고려하기 위하여 (e),(c2),($j2_2$)와 ($j2_3$)을 다시 정리하면 다음과 같다.

$$(e)=\sum_{t=n-3k}^{-1}\Bigg(\sum_{i=2k}^{t+n+k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\Bigg)\alpha^t$$
$$=\sum_{t=n-3k}^{-1}\Bigg(\sum_{i=2k}^{n-1}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}+\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}+\sum_{i=t+3k+1}^{t+n+k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}\Bigg)\alpha^t$$
$$=\sum_{t=n-3k}^{-1}\Bigg(\underbrace{\sum_{i=t-n+4k+1}^{t+2k}a_{t-i+4k}b_i}_{(e_1)}+\sum_{i=n}^{t+3k}\tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}+\underbrace{\sum_{i=t-n+4k+1}^{t+2k}a_ib_{t-i+4k}}_{(e_2)}\Bigg)\alpha^t,$$

$$(c2)=\sum_{t=n-3k}^{-2}\Bigg(\sum_{i=t+n+k+1}^{n+k-1}\tilde{a}_{i,k}\tilde{b}_{t-i+2n+2k,k}\Bigg)\alpha^t$$
$$=\sum_{t=n-3k}^{-2}\Bigg(\sum_{i=t+2k+1}^{2k-1}a_ib_{t-i+4k}\Bigg)\alpha^t,$$

$$(j2_2)=\sum_{t=n-3k}^{-1}\Bigg(\sum_{i=t-n+4k+1}^{2k-1}a_{t-i-n+5k}b_i\Bigg)\alpha^t$$
$$=\sum_{t=n-3k}^{-1}\Bigg(\underbrace{\sum_{i=t-n+4k+1}^{t+2k}a_{t-i-n+5k}b_i}_{(j2_{2-1})}+\underbrace{\sum_{i=t+2k+1}^{2k-1}a_{t-i-n+5k}b_i}_{(j2_{2-2})}\Bigg)\alpha^t,$$

$$(j2_3)=\sum_{t=n-3k}^{-1}\Bigg(\sum_{i=t-n+4k+1}^{2k-1}a_ib_{t-i-n+5k}\Bigg)\alpha^t$$
$$=\sum_{t=n-3k}^{-1}\Bigg(\underbrace{\sum_{i=t-n+4k+1}^{t+2k}a_ib_{t-i-n+5k}}_{(j2_{3-1})}+\underbrace{\sum_{i=t+2k+1}^{2k-1}a_ib_{t-i-n+5k}}_{(j2_{3-2})}\Bigg)\alpha^t. \quad (13)$$

식 (13)에서 $t=-1$일 때 ($j2_{2-2}$)와($j2_{3-2}$)는 존재하지 않으며, (e_1)과 ($j2_{2-1}$), (e_2)와 ($j2_{3-1}$), (c2)와 ($j2_{3-2}$)는 다음과 같이 간소화된다.

$$(e_1)+(j2_{2-1})=\sum_{i=t-n+4k+1}^{t+2k}(a_{t-i+4k}+a_{t-i-n+5k})b_i \quad (14)$$

$$(e_2)+(j2_{3-1})=\sum_{i=t-n+4k+1}^{t+2k}a_i(b_{t-i+4k}+b_{t-i-n+5k}) \quad (15)$$

$$(c2)+(j2_{3-2})=\sum_{i=t+2k+1}^{2k-1}a_i(b_{t-i+4k}+b_{t-i-n+5k}). \quad (16)$$

식 (16)은 $t=-1$인 항에서 존재하지 않는다. 식 (11)은 가정 $n-3k \le -1$에 의하여 $t=-k$일 때 계산되는 값

$$(b_{n-1}+b_{k-1}),(b_{n-2}+b_{k-2}),\cdots,(b_{2k+1}+b_{-n+3k+1})$$

에 의하여, 나머지 $-k+1 \le t \le n-3k-2$ 항에서 계산되는 모든 ($\tilde{b}_{t-i+n+3k,k}+\tilde{b}_{t-i+n+4k,0}$) 가 계산된다. 그리고 식 (10)의 ($a_{t-i-n+5k}+a_{t-i+4k}$)와 ($b_{t-i-n+5k}+b_{t-i+4k}$)도 마찬가지로 $t=0$일 때 계산되는

$$(a_{n-1}+a_{k-1}),(a_{n-2}+a_{k-2}),\cdots,(a_{2k+1}+a_{-n+3k+1}),$$
$$(b_{n-1}+b_{k-1}),(b_{n-2}+b_{k-2}),\cdots,(b_{2k+1}+b_{-n+3k+1})$$

에 의하여, $1 \le t \le n-2k-2$에서 계산되는 모든 ($a_{t-i-n+5k}+a_{t-i+4k}$)와 ($b_{t-i-n+5k}+b_{t-i+4k}$) 는 계산된다. 또한, 식 (14), (15), (16) 도 마찬가지로 $t=n-3k$일 때 계산되는 값

$$(b_{n-1}+b_{k-1}),(b_{n-2}+b_{k-2}),\cdots,(b_{2k}+b_{-n+3k}),$$
$$(a_{n-1}+a_{k-1}),(a_{n-2}+a_{k-2}),\cdots,(a_{2k}+a_{-n+3k}),$$
$$(b_{2k-1}+b_{-n+3k-1}),(b_{2k-2}+b_{-n+3k-2}),\cdots,(b_{n-k+1}+b_1),$$

에 의해 나머지 항의 모든 값이 계산된다.

$n-3k=-1$인 경우 위에서 기술한 바와 같이 (14)과 (15)의 계산값에 의하여 (k)와 식 (11)의 모든 값이 계산되며 (16)은 존재하지 않는다. $n-3k \le t \le -2$인 경우 또한 (14)과 (15)의 계산값에 의하여 (k)와 식 (11)의 모든 값이 계산되며 추가적으로 (16)이 계산된다. 따라서 $n-3k=-1$ 또는 $n-3k \le -2$ 경우의 XOR 연산량은 다음과 같다. $n-3k=-1$인 경우 (14)과 (15)에서 $2(n-k)=(n-2k+3k-1)=n-k-1$ 번의 XOR 연산이 소요되며, $n-3k \le -2$인 경우 (14)과 (15)에서 $2(n-k)$번의 XOR 연산과 (14.3)에서 $-n+3k-1$번의 XOR 연산이 소요되므로 전체 $2(n-k)+(-n+3k-1)=n-k-1$번의 XOR 연산이 소요된다. 즉, $n-3k \le -1$인 경우 추가적으로 소요되는 XOR 연산량은 $N_X=n-k-1$이다.

〔표 2〕 모든 경우에 대한 c_{t+2k}의 식의 XOR 시간지연

t의 범위	k의 조건	c_t의 계산	AND Gates	XOR Delays
$-2k \le t \le -k-2$	$2 \le k \le \frac{n-1}{2}$	$\sum_{i=2k}^{t+n+2k} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=t+n+2k+1}^{n+k-1} \tilde{a}_{i,k}\tilde{b}_{t-i+2n+3k,k} + \sum_{i=0}^{t+2k}(a_i+a_{i+k})(b_{t-i+2k}+b_{t-i+3k}) + \sum_{i=t+2k+1}^{k-1}(a_i+a_{i+k})(b_{t-i+3k}+b_{t-i+4k})$	n	$\lceil \log_2(n+t+1 + 2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(-k-t-1) \rceil})\}$
$t=-k-1$	$1 \le k \le \frac{n-1}{2}$	$\sum_{i=0}^{t+2k}(a_i+a_{i+k})(b_{t-i+2k}+b_{t-i+3k}) + \sum_{i=2k}^{t+n+2k} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0}$	n	$\lceil \log_2(n-k + 2^{\lceil \log_2(2k) \rceil})\}$
$-k \le t \le \min\{n-3k-2,-2\}$	$2 \le k \le \frac{n-2}{2}$	$\left[\sum_{i=t+n+k+1}^{n+k-1} \tilde{a}_{i,k}\tilde{b}_{t-i+2n+2k,k}\right] + \sum_{i=t+3k+1}^{n-1} a_i(\tilde{b}_{t-i+n+4k,0}+\hat{b}_{t-i+n+3k,k}) + \sum_{i=n}^{n+k-1} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=2k}^{t+3k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k} + \sum_{i=n}^{t+n+k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}$	$n+t+1$	$\lceil \log_2(2n-3k + 2^{\lceil \log_2(-t-1) \rceil})\}$
$t=-1$	$\frac{n+1}{3} \le k \le \frac{n-1}{2}$ (②③의 경우)	$\left[(\sum_{i=0}^{-n+3k-1}(a_i+a_{i+k})(b_{-i-n+3k-1}+b_{-i-n+4k-1}) + \sum_{i=-n+3k}^{k-1}(a_i+a_{i+k})(b_{-i-n+4k-1}+b_{-i-n+5k-1}) + \sum_{i=3k}^{n+k-1} \tilde{a}_{i,k}\tilde{b}_{-i+n+4k-1,k}\right] + \sum_{i=n}^{3k-1} \tilde{a}_{i,k}\tilde{b}_{-i+n+3k-1,k} + \sum_{i=-n+4k}^{2k-1}(a_{-i+4k-1}+a_{-i-n+5k-1})b_i + \sum_{i=-n+4k}^{2k-1} a_i(b_{-i+4k-1}+b_{-i-n+5k-1})$	$n-k$	$\lceil \log_2(3n-5k + 2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(n-2k) \rceil})\}$
	$k=\frac{n}{3}$	$\sum_{i=t+3k+1}^{n+k-1} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=2k}^{t+n+k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}$	n	$\lceil \log_2(2n-3k) \rceil = \lceil \log_2(n) \rceil$
	$1 \le k \le \frac{n-1}{3}$	$\sum_{i=t+3k+1}^{n-1} a_i(\tilde{b}_{t-i+n+4k,0}+\tilde{b}_{t-i+n+3k,k}) + \sum_{i=n}^{n+k-1} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=2k}^{t+3k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k} + \sum_{i=n}^{t+n+k} \hat{a}_{i,k}\hat{b}_{t-i+n+3k,k}$	n	$\lceil \log_2(2n-3k) \rceil$
$0 \le t \le n-3k-2$	$1 \le k \le \frac{n-2}{3}$	$\sum_{i=t+3k+1}^{n-1} \tilde{a}_i(\tilde{b}_{n-i+n+4k,0}+\tilde{b}_{t-i+n+3k,k}) + \sum_{i=2k}^{t+2k} \tilde{a}_{i,k}\tilde{b}_{t-i+4k,k} + \sum_{i=n}^{n+k-1} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=t+2k+1}^{t+3k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k} + \sum_{i=n}^{n+k-1} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k}$	n	$\lceil \log_2(2n-3k-t-1) \rceil$
$t=n-3k-1$	$\frac{n+1}{3} \le k \le \frac{n-1}{2}$	$\sum_{i=t+3k+1}^{n+k-1} \tilde{a}_{i,0}\tilde{b}_{t-i+n+4k,0} + \sum_{i=2k}^{t+n+k} \tilde{a}_{i,k}\tilde{b}_{t-i+n+3k,k} + \sum_{i=t+n+k+1}^{n+k-1} \tilde{a}_{i,k}\tilde{b}_{t-i+2n+2k,k}$	$n+t+1$	$\lceil \log_2(2n-3k + 2^{\lceil \log_2(-n+3k) \rceil})\}$

3.2 제안하는 비트-병렬 곱셈기의 복잡도

본 소절에서는 이전 소절에서 기술한 새로운 SPB 비트-병렬 곱셈기의 구조와 시간 및 공간 복잡도에 대하여 논한다. 이전 절에서 보인바와 같이 식 (10)에서 항 α^t의 계수들은 k의 조건에 따라 달라진다. 이와 같이 구분된 k에 따른 계수 c_{t+2k}의 계산을 정리하면 [표 2]와 같다. [표 2]에서 AND 게이트 수는 중복된 연산은 제외한 결과이다([표 2]에서 [·]로 표현된 부분은 연산량에서 제외된다.). 즉, 식 (2)에서 (c2)와 식 (6)의 (a2)+(j1)+($j2_4$)의 AND 게이트 수는 제외된 결과이다. 제안하는 SPB 비트-병렬 곱셈 방법은 $n \neq 1$이고 $n > 2k$인 경우 [1]에서 제안된 식 (6)의 $n-3k \le t \le n-2k-2$인 경우의 항만 식을 변형된다. 따라서 구분된 k에 따라 제안하는 형태로 식이 변형되는 부분을 정리하면 다음과 같다.

①. $n \neq 2k+1$인 경우 :
$\max\{n-3k,0\} \le t \le n-2k-2$사이 항

②. $n-3k=-1$인 경우 : $t=-1$인 항,
$\max\{n-3k,0\} \le t \le n-2k-2$사이 항

③. $n-3k \le -2$인 경우 : $n-3k \le t \le -2$사이 항,
$t=-1$인 항, $\max\{n-3k,0\} \le t \le n-2k-2$사이 항

제안하는 비트-병렬 Mastrovito 곱셈기의 시간 및 공간 복잡도를 정리하면 다음 정리와 같다.

정리 1. $n > 2k$ 일 때 제안하는 비트-병렬 MSPB 곱셈기의 공간 복잡도는

$$T_A = \begin{cases} n^2-k^2-\dfrac{k(k-1)}{2}, & 1 \le k \le \dfrac{n+1}{3} \\ 2n^2-6nk+n+\dfrac{5k(3k-1)}{2}, & \dfrac{n+2}{3} \le k < \dfrac{n}{2} \end{cases},$$

$$T_X = \begin{cases} n^2-k^2-\dfrac{(k-1)(k-6)}{2}, & 1 \le k \le \dfrac{n}{3} \\ n^2-k^2+1-\dfrac{(k-1)(k-6)}{2}, & k = \dfrac{n+1}{3} \\ 2n^2-6nk+2n-1+\dfrac{5k(3k-1)}{2}, & \dfrac{n+2}{3} \le k < \dfrac{n}{2} \end{cases}$$

이다.

증명. 우선 제안하는 비트-병렬 Mastrovito 곱셈기의 AND 게이트 복잡도를 고려해보자. [표 1]에서 소개한 바와 같이 α^t의 계수는 n과 k에 따라 달라진다. 따라서 [표 1]에서 구분된 1)~5)에 따라 복잡도를 기술한다. 1)의 경우 t에 따라 $-2k \le t \le -k-2$, $t=-k-1$, $-k \le t \le -2$, $t=-1$, $0 \le t \le n-3k-2$, $t=n-3k-1$, $n-3k \le t \le n-2k-2$와 $t=n-2k-1$로 나누어지므로 [표 2]의 AND 게이트 수에 의하여 각각의 α^t항은 n, n, $n+t+1$, n, n, n, $n-k$와 $n-k$개의 AND 연산으로 계산된다. 따라서 1)의 AND 게이트 수는

$$nk+(n-2k+1)(n-k)+\frac{(k-1)(2n-k)}{2}$$
$$=n^2-k^2-\frac{k(k-1)}{2}$$

이다. [표 1]에서 2)~4)의 경우도 이와 같은 방법으로 계산하면 $n^2-k^2-k(k-1)/2$개의 AND 게이트로 계산된다. 마지막으로 5)를 살펴보자. 5)의 경우 t에 따라 $-2k \le t \le -k-2$, $t=-k-1$, $-k \le t \le n-3k-2$, $t=n-3k-1$, $n-3k \le t \le -2$, $t=-1$, $0 \le t \le n-2k-2$와 $t=n-2k-1$로 나누어지므로 [표 2]의 AND 게이트 수에 의하여 각각의 α^t항은 n, n, $n+t+1$, $n+t+1$, $n-k-t-1$, $n-k$, $n-k$와 $n-k$개의 AND 연산으로 계산된다. 따라서 1)의 AND 게이트 수는 $nk+(3n-4k+1)/2+(n+k-1)(-n+3k)/2+(n-2k)(n-k)=2(n-(3/2)k)^2+n+k(6k-5)/2$이다. 다음으로 XOR 게이트 복잡도를 고려해보자. 먼저 [표 1]에서 구분된 1)~4)의 경우 XOR 게이트 복잡도를 살펴보자. 전체 AND 게이트 수에서 각각 항마다 1개씩 n개가 감소되고 (c2)와 ((a2)+(j1)+($j2_4$))의 덧셈에서 각각 $k-1$개와 k개의 XOR 게이트가 필요하다. 또한, 3.1.1과 3.1.2절에서 소개한 간소화 과정의 (b_u+b_v)들의 덧셈에서 $n-k-2$([표 1], [표 4])의 경우 $n-k-1$개)의 XOR 게이트가 소요되고, MSPB로의 기저변환에서 계산되는

$$(a_0+a_k),\cdots,(a_{k-1}+a_{2k-1}),(b_0+b_k),\cdots,(b_{k-1}+b_{2k-1})$$

에서 $2k$개의 XOR 게이트가 소요된다. 따라서 전체 XOR 게이트 수는

$$\left(n^2-k^2-\frac{k(k-1)}{2}\right)-n+(k-1)+k+(n-k-2)+2k$$
$$=\left(n^2-k^2-\frac{k(k-1)}{2}\right)+3k-3=n^2-k^2-\frac{(k-1)(k-6)}{2}$$

이다(단, 4)의 경우 $n^2-k^2+1-(k-1)(k-6)/2$이다.). 5)의 XOR 게이트 또한 이와 같이 계산되지만 Ⅲ.1.2절에서 소개한 바와 같이 $n-3k \le t \le -2$ 사이에서 (c2)의 일부가 간소화되므로 $k-1$ XOR 게이트가 아닌 $n-2k$ 개의 XOR 게이트가 소요된다. 그리고 Ⅲ.1.2절에서 소개한 간소화 과정 (b_u+b_v)들의 덧셈

에서 $n-k-1$ XOR 게이트가 소요된다. 따라서 전체 XOR 게이트 수는

$$\left(2n^2-6nk+n+\frac{5k(3k-1)}{2}\right)-n+(n-2k)+k$$
$$+(n-k-1)+2k=\left(2n^2-6nk+n+\frac{5k(3k-1)}{2}\right)+n-1$$
$$=2n^2-6nk+2n-1+\frac{5k(3k-1)}{2}$$

이다.

정리 2 .$n>2k$일 때 제안하는 비트-병렬 Mastrovito 곱셈기의 시간 복잡도는

$$T_C=\begin{cases}1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X \\ \qquad ,n>4k-2+2^{\lceil \log_2(2k)\rceil} \\ 1T_A+\lceil \log_2(n+k-2+2^{\lceil \log_2(2k)\rceil}+2^{\lceil \log_2(k-1)\rceil})\rceil T_X \\ \qquad ,n\le 4k-2+2^{\lceil \log_2(2k)\rceil}\end{cases}$$

이다.

증명. 제안하는 비트-병렬 Mastrovito 곱셈기의 c_{t+2k}의 t에 따른 시간 복잡도는 〔표 2〕와 같다. 〔표 2〕에서 t가 $-2k\le t\le -k-2$일 때의 경우를 제외하고 나머지 경우에서는 t가 가장 작은 값일 때 구간내의 최대 시간지연을 가진다. t가 $-2k\le t\le -k-2$인 경우의 시간지연은 $2^{\lceil \log_2(-k-t-1)\rceil}$가 최대값이 되는 경우의 값 $2^{\lceil \log_2(k-1)\rceil}$을 만족하는 t의 값 중 가장 큰 t값에서 최대 시간 지연을 가진다. $k-1=2^{\lceil \log_2(k-1)\rceil-1}$라 가정하자(이때, $1\le u\le 2^{\lceil \log_2(k-1)\rceil-1}$이다.). 그러면 $t=-2k+u-1$일 때 최대 시간지연을 가지므로

$$\lceil \log_2(n-2k+u+2^{\lceil \log_2(2k)\rceil}+2^{\lceil \log_2(k-u)\rceil})\rceil T_X$$

이다. 〔표 2〕로 부터 〔표 1〕에서 1)의 경우는 t가

〔표 3〕 SPB 곱셈기의 게이트 수 비교

구 분		AND 게이트 복잡도		XOR 게이트 복잡도		
		$1\le k\le \frac{n+1}{3}$	$\frac{n+2}{3}\le k<\frac{n}{2}$	$1\le k\le \frac{n}{3}$	$k=\frac{n+1}{3}$	$\frac{n+2}{3}\le k<\frac{n}{2}$
〔8〕		n^2		$n^2-n+\frac{k(k-1)}{2}+\frac{(n-k)(n-k-1)}{2}$		
〔4〕		n^2		n^2-1		
〔1〕	Type I	n^2-k^2		n^2-1		
	Type II	n^2-k^2		$n^2-(k-1)^2-1$		
제안하는 곱셈기		$n^2-k^2-\frac{k(k-1)}{2}$	$2n^2-6nk+n+\frac{5k(3k-1)}{2}$	$n^2-k^2-\frac{(k-1)(k-6)}{2}$	$n^2-k^2+1-\frac{(k-1)(k-6)}{2}$	$2n^2-6nk+2n-1+\frac{5k(3k-1)}{2}$

〔표 4〕 SPB 곱셈기의 시간지연 비교

구 분				시간 복잡도
〔8〕				$1T_A+(\lceil \log_2(2n-k)\rceil)T_X$
〔4〕				$1T_A+(\lceil \log_2(2n-k-1)\rceil)T_X$
〔1〕	Type I	$3k+1\le n$		$1T_A+\lceil \log_2(2n-2k-1)\rceil T_X$
		$3k+1>n$		$1T_A+\lceil \log_2(n+k)\rceil T_X$
	Type II	$k=1$		$1T_A+\lceil \log_2(2n-2k-1)\rceil T_X=1T_A+\lceil \log_2(2n-3)\rceil T_X$
		$k=2^m+1$	$5k-3\le n$	$1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$
			$5k-3>n$	$1T_A+\lceil \log_2(n-k+2^{\lceil \log_2(k)\rceil})\rceil T_X$
		$k\ne 2^m+1$	$\{2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1\}\le n-3k$	$1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$
			$\{2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1\}>n-3k$	$1T_A+\lceil \log_2(n+3\cdot 2^{\lceil \log_2(k-1)\rceil-1}+2^{\lceil \log_2(u)\rceil}-1)\rceil T_X$
제안하는 곱셈기		$n\le 4k-2+2^{\lceil \log_2(2k)\rceil}$		$1T_A+\lceil \log_2(n+k-2+2^{\lceil \log_2(2k)\rceil}+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$
		$n>4k-2+2^{\lceil \log_2(2k)\rceil}$		$1T_A+\lceil \log_2(2n-3k+2^{\lceil \log_2(k-1)\rceil})\rceil T_X$

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

$-k \le t \le -2$ 또는 $n-3k \le t \le n-2k-2$ 사이 일 때 최대시간지연을 가지므로, 만약 $n > 4k-2+2^{\lceil \log_2(2k) \rceil}$ 이면, $-2k \le t \le -k-2$에서 최대시간지연을 가지고, 그렇지 않으면 $n-3k \le t \le n-2k-2$에서 최대시간지연을 가진다. 또한 2)~3)의 경우는 모두 $n-3k \le t \le n-2k-2$에서 최대시간지연을 가지므로

$$1T_A + \lceil \log_2(n+k-2+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(k-1) \rceil}) \rceil T_X$$

이다. 4) 경우는 $t=-1$인 최대시간지연을 가지므로

$$\begin{aligned} & 1T_A + \lceil \log_2(3n-5k+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(n-2k) \rceil}) \rceil T_X \\ = & 1T_A + \lceil \log_2(n+k-2+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(k-1) \rceil}) \rceil T_X \end{aligned}$$

이다(4)의 경우 $n=3k-1$). 마지막 5)의 경우 $n-3k \le t \le -2$에서 최대시간지연을 가지므로

$$1T_A + \lceil \log_2(n+k-2+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(k-1) \rceil}) \rceil T_X$$

이다. 이를 모두 정리하면 제안하는 비트-병렬

Mastrovito 곱셈기의 시간 복잡도는 $n-3k \le t \le -2$에서 최대시간지연을 가지므로

$$1T_A + \lceil \log_2(n+k-2+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(k-1) \rceil}) \rceil T_X$$

이다. 따라서 이를 모두 정리하면 제안하는 비트-병렬 Mastrovito 곱셈기의 시간 복잡도는

$$T_C = \begin{cases} 1T_A + \lceil \log_2(2n-3k+2^{\lceil \log_2(k-1) \rceil}) \rceil T_X \\ \qquad , n > 4k-2+2^{\lceil \log_2(2k) \rceil} \\ 1T_A + \lceil \log_2(n+k-2+2^{\lceil \log_2(2k) \rceil} + 2^{\lceil \log_2(k-1) \rceil}) \rceil T_X \\ \qquad , n \le 4k-2+2^{\lceil \log_2(2k) \rceil} \end{cases}$$

이다.

본 논문에서는 $n < 2k$인 경우는 [1]의 결과와 같이 k 대신에 $n-k$를 사용하여 전개하면 동일하므로 생략한다. 정리 1에서 알 수 있듯이 1)~4) 경우는

모두 [1]의 결과보다 AND 게이트와 XOR 게이트가 모두 감소한다. 그러나 5)의 경우 k에 따라 효율성이 달라진다. 따라서 [1]의 type II의 XOR 게이트 수 $n^2-1-(k-1)^2$보다 $2n^2-6nk+2n-1+5k(3k-1)/2$가 큰 경우는 제안하는 방법을 적용하지 않고 [1]의 type II 곱셈기 구조를 사용한다. 이는 k가 $n/2$에 매우 근사한 경우 식 (13)의 $(j2_{2-2})$가 매우 커지기 때문이다. 따라서 제안하는 방법이 [1]의 Type II 보다 효율성이 같거나 작은 경우는 다음과 같다.

- $n > 2k$인 경우
  - $k=1$
  - $\frac{n+2}{3} \le k < \frac{n}{2}$이면서 [1]의 type II의 XOR 게이트 수 보다 많은 경우
- $n = 2k$인 경우
- $n < 2k$인 경우
  - $n = k+1$
  - $\frac{n}{2} < k \le \frac{2n+2}{3}$이면서 [1]의 type II의 XOR 게이트 수 보다 많은 경우

IV. 비교 및 결론

본 소절에서는 SPB 기반의 기존의 결과와 제안하는 SPB 비트-병렬 Mastrovito 곱셈기의 효율성을 비교하고 결론을 내린다. 삼항 기약다항식 기반에서 SPB를 사용하는 기존 결과와 제안하는 비트-병렬 Mastorvito 곱셈기와 공간 및 시간 복잡도를 비교하면 각각 [표 3], [표 4]와 같다. [표 3]에서 알 수 있듯이 제안하는 곱셈기는 기존의 가장 작은 공간 복잡도를 가지는 [1]의 type II와 비교하여 $1 \le k \le (n+1)/3$인 경우 항상 공간복잡도가 작으며 $(n+2)/3 \le k < n/2$인 경우에도 k가 $n/2$에 매우 근사하는 경우를 제외하고 기존 결과보다 항상 작은 공간복잡도를 가진다. 따라서 제안하는 곱셈식의 공간복잡도가 비효율적인 경우 제안하는 방법은 [1]의 type II 구조를 사용한다. $n > 2k$를 만족하는 차수가 $1 \le n < 1{,}000$인 모든 삼항 기약다항식 1,491개 중 [1]의 type II 곱셈기 보다 더 작은 공간 복잡도를 가지는 경우는 1314개(88.13%) 존재한다. 기약다항식을 선택할 때 주어진 차수에서 가장 작은 k를 선택한다면 이러한 기약다항식을 사용할 확률은 매우 낮다.

다음으로 [표 4]의 결과를 이용하여 시간 복잡도를 고려하여 보자. $n > 2k$를 만족하는 차수가 $1 \le n < 1{,}000$인 모든 삼항 기약다항식 1,491개에 대하여 [1]의 type I, II와 비교하면 다음과 같다. [1]의 type I 곱셈기와 비교하여 2개(0.14%)가 $1T_X$ 감소하고, 969개(68.29%)가 같으며 343개(24.17%)가 $1T_X$ 증가한다. 또한 [1]의 Type II 곱셈기와 비교하여 942개(63.18%)가 같으며 549개(36.82%)가 $1T_X$ 증가한다. [1]의 Type II 곱셈기의 공간복잡도가 낮아 제안하는 곱셈기를 사용하지 않고 [1]의 Type II 곱셈기를 사용하는 경우까지 모두 고려하면 실제 399

개(26.76%)만 $1T_X$ 증가한다.

제안하는 곱셈기는 차수가 1,000 이하인 모든 삼항 기약 다항식을 고려할 때, 그 중 대략 30%정도만 $1T_X$ 증가하지만 기존의 곱셈기와 비교하여 더 작은 공간 복잡도를 가지므로 저면적이 요구되는 환경에 효율적으로 적용가능하다.

참 고 문 헌

[1] 장남수, 김창한, 홍석희, 박영호, "삼항 기약다항식을 위한 효율적인 Shifted polynomial Basis 비트-병렬 곱셈기," 정보보호학회논문지, 19(2), pp. 49-61, 2009년 4월.

[2] 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.

[3] H. Fan and M.A. Hasan, "Relationship between $GF(2^n)$ Montgomery and shifted polynomial basis multiplication algorithms," *IEEE Trans. Computers.*, vol. 55, no. 9, pp. 1202-1206, Sep. 2006.

[4] 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.

[5] A. Halbutogullari and M.A. Hasan, "Mastorovito Multiplier for General Irreducible Polynomials," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 503-518, May. 2000.

[6] E.D. Mastrovito, "VLSI designs for multiplication over finite fields $GF(2^m)$," *In Proceedings of the Sixth Symposium on Applied Algebra, Algebraic Algorithms, and Error Correcting Codes*, vol. 6, pp. 297-309, July. 1988.

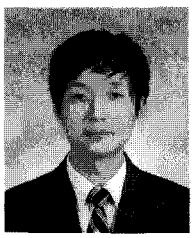
[7] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Linkoping University Dept. Electr. Eng., Linkoping, Sweden, 1991.

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

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

[10] H. Shen and Y. Jin, "Low complexity bit parallel multiplier for $GF(2^n)$ generated by equally-spaced trinomials," *Information Processing Letters.*, vol. 107, pp. 211-215, Aug. 2008.

〈著者紹介〉



장 남 수 (Nam Su Chang) 정회원
2002년 2월: 서울 시립대학교 수학과 이학사
2004년 8월: 고려대학교 정보보호 대학원 공학석사
2010년 2월: 고려대학교 정보경영공학전문대학원 공학박사
2010년 2월~6월: 고려대학교 정보경영공학전문대학원 연구교수
2010년 7월~현재: 세종사이버대학교 정보보호시스템학과 전임강사
〈관심분야〉 암호칩 설계 기술, 부채널 공격, 공개키 암호 알고리즘, 공개키 암호 암호분석

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