Wibro 시스템에서 중첩 행렬을 이용한 준 순환 LDPC 부호의 설계 및 계층 복호기

정회원 신 범 규*, 박 호 성*, 종신회원 김 상 효**, 노 종 선*

Quasi-Cyclic LDPC Codes using Superposition Matrices and Their Layered Decoders for Wibro Systems

Beomkyu Shin*, Hosung Park* *Regular Members,*
Sang-Hyo Kim**, Jong-Seon No* *Lifelong members*

요 약

Wibro를 포함한 많은 시스템에서 순환 치환 행렬(circulant)로 구성된 준 순환 LDPC(low-density parity-check) 부호를 사용하고 있다. 하지만 준 순환 부호의 기저 행렬 크기의 제약으로 인해 계층 복호(layered decoding)가 가능하고 일정 값 이상의 거스(girth)를 만족하면서 동시에 최적의 차수 분포를 갖도록 하는 것은 매우 힘들다. 본 논문에서는 이러한 문제점을 극복하기 위해 중첩 행렬(superposition matrix) 구조를 가지는 준 순환 LDPC 부호를 제안한다. 중첩 행렬을 이용할 경우에 특화된 거스 점검 조건들을 유도하고, 기존 행렬 구조와 중첩 행렬 구조 두 가지 모두에 대해 계층 복호를 수행할 수 있는 새로운 LDPC 복호기 구조를 제안한다. 모의실험을 통하여 중첩 행렬 구조를 가지는 LDPC 부호는 복호 시 수렴 속도가 개선되고 오류 정정율이 향상됨을 보인다.

Key Words : Error Correcting Codes, Low-Density Parity-Check(LDPC) Codes, Superposition Matrix, Layered Decoder, Wibro Systems, Girth Condition

ABSTRACT

Most communication systems including Wibro use quasi-cyclic LDPC codes composed of circulants. However, it is very difficult to design quasi-cyclic(QC) LDPC codes with optimal degree distribution satisfying conditions on layered decoding and girth due to the restriction of the size of its base matrix.

In this paper, we propose a good solution by introducing superposition matrices to QC LDPC codes. We derive the conditions on checking girth of QC LDPC codes with superposition matrices, and propose new decoder to support layered decoding both for original QC LDPC codes and their modifications with superposition matrices. Simulation results show considerable improvements to convergence speed and error-correcting performance of proposed scheme which adopts QC LDPC codes with superposition matrices.

Ⅰ. 서 론

1993년 Shannon의 한계에 근접하는 성능의 터보 부호(turbo code)[1]가 제안된 이후로 구현 가능한 연산량의 복호 방식을 가지면서 채널 용량에 근접하는 성능을 갖는 low-density parity-check(LDPC) 부호가 재발견 되었다[2]. LDPC 부호는 패리티 체

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(No.2010-0000867)
* 서울대학교 전기 · 컴퓨터공학부 부호및암호연구실 및 뉴미디어통신공동연구소 ({thechi,lovepark98}@ccl.snu.ac.kr, jsno@snu.ac.kr)
** 성균관대학교 정보통신공학부 통신및부호이론연구실 (iamshkim@skku.edu)
논문번호 : KICS2009-12-653, 접수일자 : 2009년 12월 30일, 최종논문접수일자 : 2010년 2월 8일

크 행렬의 원소 중 대부분이 0을 갖도록 설계되어 확률 값에 기반한 메시지를 이용한 반복 복호가 용이하다는 장점을 갖는다. 이러한 성능과 구현 측면의 많은 장점들로 인해 DVB-S2, Wibro 시스템을 비롯한 다양한 방송 및 무선 통신 표준의 오류정정 부호로 채택되었다[3]-[7].

이렇게 우수한 성능을 갖는 LDPC 부호는 기본적으로 무작위적인 생성 방법을 갖는데 좋은 성능을 달성하기 위해 패리티 체크 행렬의 크기가 커지게 되면 복호 연산이나 메모리 측면에서 구현이 어려워지기 때문에 대수적인 생성 방법[8]을 이용하거나 그래프 이론[9]을 이용하여 생성하려는 다양한 시도들이 있어왔다. 그 중 대표적인 것이 준 순환 LDPC 부호[8]라 할 수 있다.

무선 이동 통신의 대표적인 예라고 할 수 있는 Wibro 시스템의 경우에도 준 순환 LDPC 부호가 채택되었다. 특히 IEEE802.16e에 제시된 준 순환 LDPC 부호들은 계층 복호(layered decoding)를 적용, 병렬 처리(parallelism)을 지원하도록 설계되었는데 복호기의 처리율을 증가시키는 장점을 갖는다.

그러나 준 순환 LDPC 부호에 계층 복호를 적용하기 위해서는 LDPC 부호의 최대 변수 노드 차수($d_v^{\max}$), 또는 등가적으로 패리티 체크 행렬의 열 무게(column weight)가 행 블록 수의 1/2 이하여야 한다는 제약을 받게 된다. 반대로 성능에 영향을 주는 최대 차수를 늘리기 위해 부호의 길이가 정해진 상황에서 행 블록의 수를 늘리게 되면 패리티 체크 행렬 내의 개별 순환 행렬 크기가 줄어들게 되기 때문에 순환 이동 값의 선택이나 성능 상의 문제가 역시 발생하게 된다.

따라서 처리율 및 속도에 영향을 주는 계층 복호를 지원하면서 동시에 LDPC 부호의 최대 차수를 증가시킬 수 있는 방안을 찾는 것은 중요한 문제이다. 본 논문에서는 중첩 행렬과 그에 따른 계층 복호기의 구조를 고안하여 제시하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 IEEE802.16e 표준에 제안된 준 순환 LDPC 부호를 중심으로 준 순환 부호의 구조와 문제점이 제시된다. 3장에서는 중첩 행렬을 이용한 패리티 체크 행렬의 확장과 이를 지원하기 위한 계층 복호기의 구조를 제시한다. 4장에서는 제안된 설계 기법의 효용성을 AWGN(additive white Gaussian noise) 채널에서 검증하고자 한다. 마지막으로 5장에서는 결론을 맺는다.

Ⅱ. 기존의 준 순환 LDPC 부호 설계

2.1 IEEE802.16e의 준 순환 LDPC 부호

본 장에서는 IEEE802.16e[3]에서 제시된 준 순환 LDPC 부호를 기반으로 동일한 표기를 사용할 것이다. 이를 위해 표준에 소개된 부호 기준으로 몇 가지 정의를 내리고 준 순환 부호에 대해서 살펴볼 것이다.

LDPC 부호들의 집합 내에 있는 각각의 LDPC 부호는 $m \times n$의 행렬 H로 정의되는데 다음과 같이 표현된다.

$$H = \begin{bmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,n_b-2} & P_{0,n_b-1} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,n_b-2} & P_{1,n_b-1} \\ P_{2,0} & P_{2,1} & \cdots & P_{2,n_b-2} & P_{2,n_b-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ P_{m_b-1,0} & P_{m_b-1,1} & \cdots & P_{m_b-1,n_b-2} & P_{m_b-1,n_b-1} \end{bmatrix}$$

여기서 $P_{i,j}$는 $z \times z$의 순환 행렬 또는 $z \times z$의 0 행렬이 될 수 있다. 행렬 H는 이동 값 z에 대해 $n = z \cdot n_b$와 $m = z \cdot m_b$인 $m_b \times n_b$의 기저 행렬 H_b로부터 확장된다. 기저 행렬 내의 0이 아닌 위치에는 $z \times z$의 순환 행렬을, 빈 공간에는 0행렬을 치환하면 확장된 패리티 체크 행렬을 얻을 수 있다. 0이 아닌 위치에 명시된 숫자는 사용된 이동 값들을 나타내는데 치환된 순환 행렬이 항등 행렬을 우측 방향으로 순환 치환해야 하는 자리 수를 나타낸다.

그림 1에서 기저 행렬의 예를 확인 할 수 있다. 0의 이동 값은 항등 행렬로 치환됨을 의미하며 이 부호의 경우 체계적인 부호 구조로 $m_b = 12$, $n_b = 24$의 값을 갖는다. z의 값을 24~96으로 변경함에 따라 576~2304 길이의 부호를 얻을 수 있다.

앞 서 언급된 바와 같이 계층 복호를 적용하기 위하여 IEEE802.16e에서는 상기와 같은 구조의 패

	94	73					55	83			7	0										
	27				22	79	9				12		0	0								
			24	22	81		33				0			0	0							
61		47					65	25							0	0						
		39				84			41	72							0	0				
				46	40		82				79	0						0	0			
		95	53						14	18									0	0		
	11	73				2			47												0	0
12				83	24		43				51											
					94		59			70	72											
		7	65					39	49													
43					66		41				26	7										

그림 1. IEEE802.16e의 부호율 0.5 LDPC부호의 기저행렬

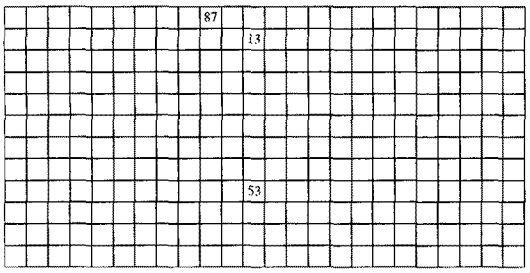


그림 2. 중첩 행렬의 예

리티 체크 행렬을 채택하였는데 첫 번째 행 블록과 세 번째 행 블록을 선택하는 경우, 다시 말해 첫 번째 행 블록에 해당하는 체크 노드들을 처리하는 경우와 세 번째 행 블록에 해당하는 체크 노드들을 처리하는 경우 서로 겹치지 않기 때문에 변수 노드의 LLR 메모리에 저장된 값을 갱신하는 경우 충돌이 일어나지 않게 된다. 반대로 생각해 보면 이러한 충돌을 회피하기 위해서 상기의 경우 패리티 체크 행렬이 가질 수 있는 최대 차수는 m_b의 1/2인 6으로 제한되는데 이는 준 순환 LDPC 부호의 설계 단계에서 최적의 차수 분포를 적용하는데 제한 요소로 작용하게 된다. 만약 오류 정정 성능을 개선하기 위해서 최대 차수를 늘리려면 계층 복호를 포기하거나 z값을 줄여야 하는데 그만큼 부호 설계의 자유도가 줄어들게 되어 좋은 성능의 부호를 찾는 것이 어려워진다.

Ⅲ. 중첩 행렬을 이용한 차수 분포 조정

전장에서 살펴보았듯이 준 순환 LDPC 부호가 갖는 많은 장점들에도 불구하고 제한적인 크기의 기저 행렬로 인해 병렬 복호를 지원하고 일정 크기 이상의 거스를 보장하면서 동시에 최적의 차수 분포를 갖도록 하는 것은 거의 불가능에 가깝다. 이 장에서는 효율적인 부호화 및 병렬 복호의 조건들을 모두 충족하면서도 차수 분포의 조절이 가능하며 설계 방법에 따라서는 거스도 증가시킬 수 있는 새로운 구조를 제시하도록 한다. 이러한 방법을 통해 성능의 개선을 이룰 수 있음을 다음 장에서 모의실험 결과를 통해 제시하기로 한다.

3.1 중첩 행렬

일반적으로 준순환 LDPC 부호는 이동 값을 내용으로 갖는 기저 행렬을 통해 각각 항등 행렬로부터 정해진 이동 값만큼 순환 회전된 블록으로 구성된다. 결과적으로 개별 블록들은 행과 열의 무게(weight)가 항등 행렬과 동일하게 각각 1의 값을 갖게 됨을 알 수 있다. 이제 기저 행렬과는 별개로 기저 행렬과 동일한 크기와 성질을 갖는 중첩 행렬(superposition matrix)를 생각했을 때 두 기저 행렬이 중첩된 상황을 고려해 볼 수 있다. 이 경우 중첩 행렬이 위치하는 블록에서는 무게가 1에서 2로 증가함을 알 수 있다.

이렇게 중첩이 된 패리티 검사 행렬 H_r은 기저 행렬과 중첩 행렬을 각각 H_b와 H_s라고 했을 때 다음과 같은 식으로 등가적으로 표현될 수 있다.

$$H_r = H_b + H_s$$

또 다음의 관계식을 만족하게 된다.

$$\underline{c}^T(H_b + H_s) = 0$$

LDPC 부호의 성능을 개선시키기 위해 최적의 차수 분포를 따르는 패리티 검사 행렬을 구성하고자 할 때 현재 주어진 준순환 LDPC 부호의 구조로는 준순환 LDPC의 많은 장점들을 포기할 수 밖에 없다. 그러나 중첩을 이용하여 행렬을 수정하는 경우 병렬 복호나 효율적인 부호화 등 많은 장점들을 고수하면서도 원하는 차수 분포로 조정이 가능하게 된다. 일반적으로 비정칙(irregular) LDPC 부호의 차수 분포를 보면 변수 노드의 경우 최고차 차수가 큰 값일수록, 또 차지하는 비중이 일정 비율 이상일수록 부호의 성능이 좋아지는 것으로 알려져 있기 때문에 이러한 차수 분포 조정에 자유도가 크다는 것은 설계 측면에서 큰 장점이 된다.

중첩 방법을 적용하는 경우 4개 이상의 블록의 중첩도 충분히 가능하나 이후의 전개에서는 복잡성 등을 고려하여 일단 고려하지 않는 것으로 한다. 또한 중첩 행렬을 표시하는 방법으로 다음과 같은 정의를 사용할 것이다.

정의

중첩 행렬은 0행렬을 제외한 모든 이동 값을 갖는 순환 행렬에 대해 다음을 열거하여 표현한다.

H_s: (행 좌표1, 열 좌표1; 이동 값1), (행 좌표2, 열 좌표2; 이동 값2)...

예시

그림 2의 중첩 행렬을 상기의 정의에 따라 표현해 보면 다음과 같다.

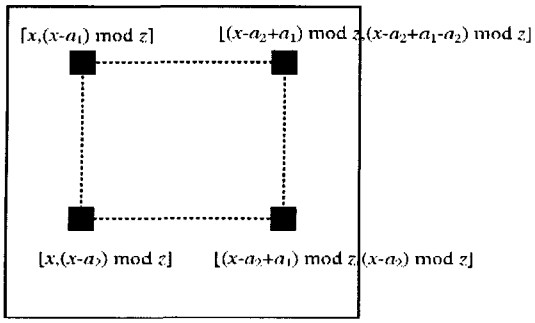


그림 3. 단일 블록 내에서 거스 4가 발생하는 경우 I

H_s: (0,9;87), (1,11;13), (8,11;53)

3.2 중첩 행렬 적용 시 거스 점검

중첩 행렬을 적용한 기저 행렬의 변형 시 Fossorier등[8]이 제시한 준 순환 LDPC 부호의 거스 점검 방법 외에도 추가적으로 고려해 줘야 하는 예외 경우들이 발생한다. 특히 거스 4가 발생하는 지 여부를 점검하는데 고려해야 할 대상으로는 다음과 같은 두 가지 내부 연결의 경우가 추가된다.

3.2.1 단일 블록 내에서 거스 4가 발생하는 경우

기존의 준순환 LDPC 부호의 경우 단일 블록 내에서는 차수가 1로 균일하게 이루어져 있기 때문에 그래프의 거스를 고려할 때 블록간의 연결만이 관심사였다. 그러나 중첩의 방법을 적용하는 경우 블록 내의 차수가 2이상이 되므로 블록 내부에서 발생하는 사이클들과 차수 2 이상의 블록 간에 생성되는 사이클들에 주의할 필요가 있다.

먼저 블록 내의 사이클에 대해 살펴보면 블록에 할당된 이동 값이 각각 a_1과 a_2라고 했을 때 상기의 그림 3과 같은 형태의 사이클이 발생할 수 있다. 이 사이클에 의해 거스는 4로 결정이 되는데 이러한 사이클이 형성되는 조건을 살펴보면 다음과 같은 관계식이 성립함을 알 수 있다.

$$(x-a_1)\bmod z \equiv (x-a_2+a_1-a_2)\bmod z$$
$$(a_1-a_2+a_1-a_2)\bmod z \equiv 0$$
$$(2a_1-2a_2)\bmod z \equiv 0$$

여기서 z는 블록의 크기를 의미하며 그림에 표시되어 있는 값들의 쌍은 각 점 (1의 위치)의 블록 내에서의 좌표로 생각할 수 있다. 마지막 식으로부터 $a_1-a_2 \equiv 0 \bmod \frac{z}{2}$의 관계를 생각 해 볼 수 있는데 만약 블록 크기 z가 홀수이면 이동 값 a_1, a_2를 어떠한 값으로 잡아도 상관이 없지만 짝수인 경우 이동 값의 차이가 블록 크기 반값의 정수 배와 일치하게 할당하면 거스 4의 사이클들이 발생하게 됨을 확인 할 수 있다.

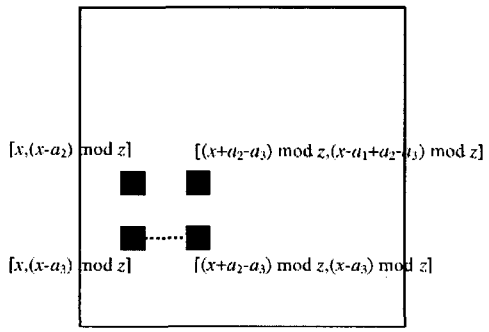


그림 4. 단일 블록 내에서 거스 4가 발생하는 경우 II

만약 3개의 블록이 중첩된 경우라면 블록 내에서 거스 4가 발생하는 것을 막기 위해서는 다음과 같은 경우도 고려해야만 한다. 블록 내의 이동 값들을 각각 크기 순서대로 a_1, a_2, a_3라 가정하자. 그러면 세 개의 이동 값들이 있는 경우 그림 4와 같은 형태와 같은 길이가 4인 사이클이 존재할 수 있다. 이동 값들의 관계를 정리해 보면,

$$(x-a_2)\bmod z \equiv (x-a_1+a_2-a_3)\bmod z$$
$$(a_1-2a_2+a_3)\bmod z \equiv 0$$
$$(a_1+a_3)\bmod z \equiv 2a_2 \bmod z$$

따라서 어느 두 이동 값의 합이 다른 이동 값의 2배가 되면 길이가 4인 사이클이 블록 내에 존재하게 된다.

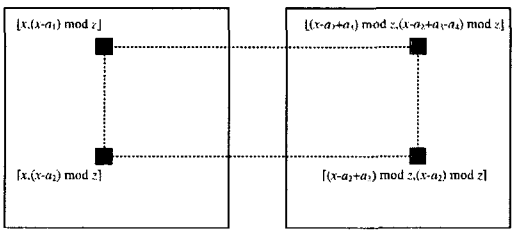


그림 5. 좌우 두 블록 간에 거스 4가 발생하는 경우

3.2.2 두 개 블록 간에 거스 4가 발생하는 경우

다음으로 중첩 블록 2개에 대해 거스 4가 발생하는 경우를 살펴보기로 한다. 첫 번째 블록에 할당된 이동 값이 a_1, a_2이고 두 번째 블록에 할당된 이동 값이 a_3, a_4라 하면 그림 5와 같은 상황을 생각해

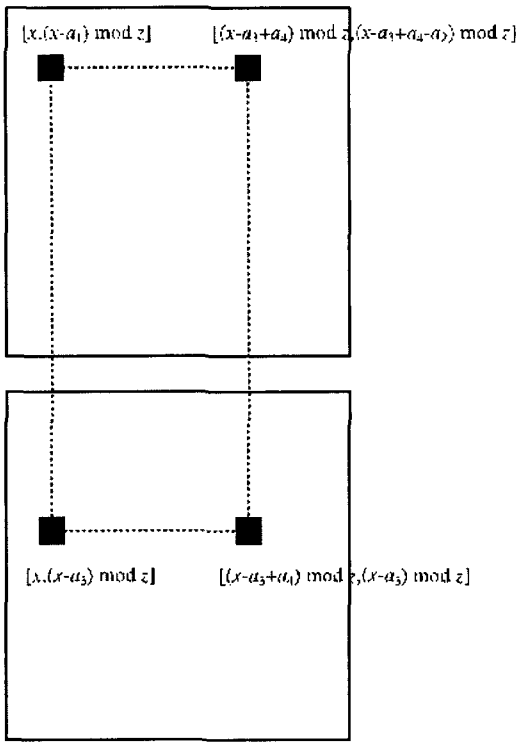


그림 6. 상하 두 블록 간에 거스 4가 발생하는 경우

볼 수 있다.

이 경우 이동 값들 사이에 다음과 같은 관계가 성립하면 역시 사이클을 형성하여 거스 4가 발생하게 된다.

$$(x-a_1)\bmod z \equiv (x-a_2+a_3-a_4)\bmod z$$
$$a_1-a_2+a_3-a_4 \equiv 0 \bmod z$$

3개 이상의 블록이 중첩되어 있는 경우는 블록 내의 이동 값들의 쌍이 증가하는 것으로 생각할 수 있다. 즉 상기의 식을 보면,

$$a_1-a_2+a_3-a_4 \equiv 0 \bmod z$$
$$(a_1-a_2)\bmod z \equiv (a_4-a_3)\bmod z$$

가 되어 각 블록 내의 이동 값들의 차이를 비교하면 거스 4의 발생 여부를 점검할 수 있는데 2개의 블록이 중첩된 경우 이동 값의 차이는 하나의 값을 갖게 되지만 3개의 경우 3종류의 차이 값을 갖게 되므로 상기 식의 만족 여부를 점검하기 위한 비교 회수가 더 많아지게 되는 차이점이 있을 뿐이다.

중첩 블록이 상하에 위치하는 경우 그림 6에서

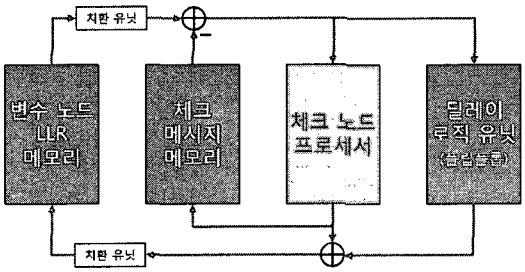


그림 7. 병렬 처리를 지원하는 기존 복호기의 구조

확인할 수 있듯이 다른 조건들은 모두 동일하고, 단지 이동 값들의 순서만 달라지는 것으로 이해 할 수 있다. 앞의 결과와 마찬가지로 위쪽의 블록에 할당된 이동 값을 a_1, a_2로 놓고, 아래 블록에 할당된 이동 값을 a_3, a_4로 놓으면 다음과 같은 관계식으로 정리된다.

$$(x-a_1)\bmod z \equiv (x-a_3+a_4-a_2)\bmod z$$
$$a_1-a_3+a_4-a_2 \equiv 0 \bmod z$$

3.3 병렬 처리를 지원하는 중첩 구조 복호기

3.3.1 기존의 복호기와 문제점

병렬 처리를 지원하기 위한 기존 복호기의 구조를 살펴보면 해당 LDPC 부호의 프로토그래프 상에서 동작하는 구조임을 확인 할 수 있다. 동일한 형태의 복호기 구조가 z 값 만큼 복제된 형태로 존재하여 크기 z의 벡터로 구성되는 메시지를 처리, 전달 및 저장하고 있는 것이다. 복호기의 구체적인 블록도는 그림 7에 나와 있다.

복호기의 동작을 살펴보기 위해 그림 8과 같은 프로토그래프의 일부를 생각해 보기로 하자. 이 때 메시지의 연산은 체크 노드 단위로 이루어지므로 그림의 체크 노드들 중 두 번째에 위치한 회색 음

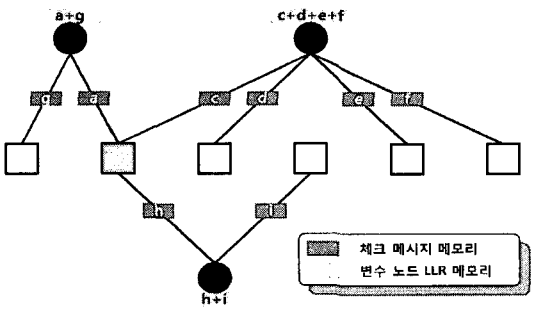


그림 8. 단일 연결만을 갖는 프로토그래프에서 체크 노드 연산 수행

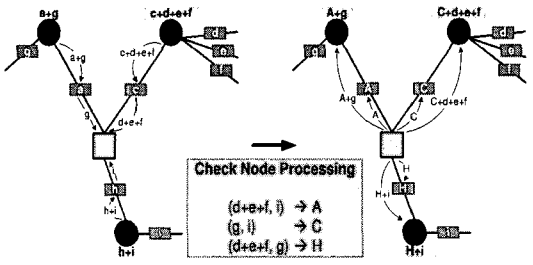


그림 9. 기존 복호기에서 메시지 갱신 과정

영 처리된 체크 노드에서 수행되는 연산을 독립적으로 생각 해 볼 수 있다.

체크 노드에서의 연산이 수행되기 전에 메모리에 저장되어 있는 메시지 값들이 그림 9의 좌측에 나와 있다. 이 때 LLR 메모리에는 개별적인 메시지 값이 저장되지 않고 각 체크 노드에서 수신한 메시지 벡터들이 합산되어 저장되어 있음을 확인 할 수 있다. 이 값으로부터 에지를 따라 연결된 체크 노드의 메시지 벡터들을 빼 주게 되면 해당 변수 노드로부터의 메시지 벡터로 볼 수 있게 된다.

이렇게 수집된 메시지들로부터 체크 노드 연산을 수행한 후 다시 체크 노드 메시지 메모리에 저장하게 되면 해당 메시지의 갱신은 종료가 되고 이전에 수신한 변수 노드로부터의 메시지에 이 값을 갱신시켜 주면 LLR 메모리에 저장되는 메시지들의 합산 값 역시 갱신이 이루어지게 된다. 이러한 과정을 체크 노드 단위로 수행하게 되면 병렬 처리가 지원되는 반복 복호의 수행이 가능하게 된다.

예를 들어 그림 9에서 'A'라는 체크 노드 연산 결과를 얻기 위해서는 우측 상단의 변수 노드에 저장된 'c+d+e+f'에서 이전 처리 과정에서 나가는 방향으로 전달된 메시지 'c'를 제외한 'd+e+f' 메시지를 얻어야 하고, 하단의 변수 노드로부터는 기 저장된 'h+i'에서 역시 이전 처리 과정에서 나가는 방향으로 전달된 메시지 'h'를 제외한 'i'를 얻은 후에 체크 노드 연산을 수행해야 한다.

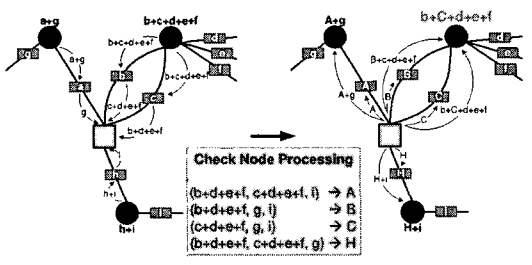


그림 10. 중첩 구조 적용 시 기존 복호기의 문제점

이러한 병렬 처리가 지원되는 복호기를 중첩 구조가 적용된 부호에 직접 적용하는 경우 어떠한 문제가 발생할 지 살펴보기로 하자.

그림 10에서 보면 초기 메시지로 'b'와 'c'값을 갖는 중첩된 에지를 확인 할 수 있다. 이러한 프로토그래프 구조에서 기존 방식의 복호기를 사용하게 되면 그림 우측에서 보는 바와 같이 중첩된 에지 중 나중에 갱신되는 쪽의 값을 LLR 메시지 메모리에 덮어 쓰게 되어 독립적인 메시지 갱신을 수행할 수 없게 된다.

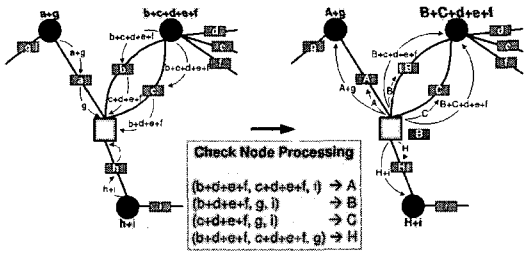


그림 11. 다중 중첩 구조를 지원하는 새로운 복호 방식

3.3.2 중첩 구조/표준 구조를 모두 지원하는 복호기

앞서 제기된 문제를 해결하기 위하여 해당 체크 노드에서 연산을 하는 과정에서 중첩 구조에 해당하는 에지의 메시지를 구하는 경우에는 별도의 처리 과정 및 저장 공간이 필요함을 그림 11을 통해 확인 할 수 있다.

그림에서 주어진 것과 같이, 동일한 연결 관계를 갖는 변수 노드의 LLR 메시지를 갱신하기 위한 연산이 에지 별로 독립적으로 수행되었었다. 그러나 다중 중첩 구조에서는 에지가 둘 이상 중첩되어 형

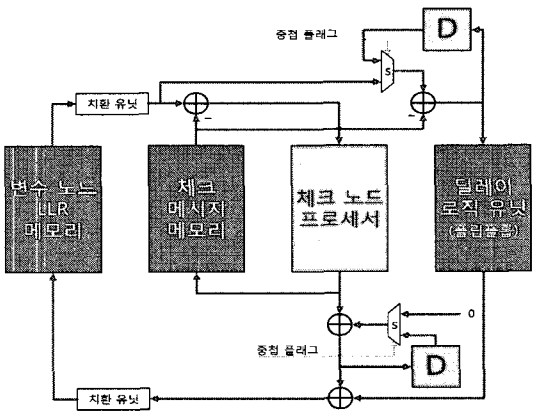


그림 12. 기본 구조와 중첩 구조를 동시에 지원하기 위하여 변경된 복호기 구조

성된 부분에서 서로간의 연산 결과를 알고 있어야 간섭이 없는 메시지의 갱신이 가능하다는 것을 알 수 있다. 이를 위해 메시지의 갱신에서 제외되어야 할 부분과 갱신이 필요한 부분을 구분하여 체크 노드 프로세서에서 처리해 주어야만 한다.

이러한 연산 절차를 반영하여 새롭게 복호기를 구성하면 그림 12와 같은 블록도를 얻을 수 있다. 그림에서 확인 할 수 있듯이 '중첩 플래그'가 설정되지 않은 일반적인 경우, 즉 단일 에지로 구성되는 경우에는 기존의 복호기 형태와 동일한 동작을 하게 된다. 체크 노드에서의 연산이 수행되는 중에 다중 에지가 발생하는 경우에는 부가적으로 앞선 에지에서의 연산 결과를 일시적으로 저장할 수 있는 공간이 필요하게 되는데 변수 노드 LLR 메모리에 저장되는 값은 갱신이 완료된 합산 값이기 때문에 누산기의 구조를 이용하여 2중, 3중으로 에지가 연결되어 있는 경우에도 상관없이 동작할 수 있도록 설계되어 있다.

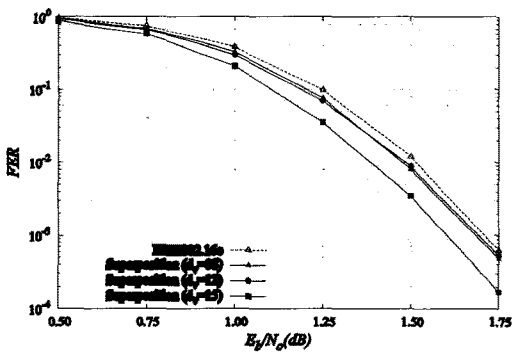


그림 13. 중첩 행렬을 이용하여 증가된 최대 차수에 따른 성능 비교

Ⅳ. 모의 실험 결과

4.1 개별 중첩 행렬 추가에 따른 성능 결과

IEEE802.16e에 제안된 부호율 1/2, 길이 2304인 LDPC 부호의 패리티 체크 행렬을 H라 하면, 앞서 정의된 중첩 행렬의 표기 방식을 이용하여 성능을 확인해 볼 중첩 행렬을 다음과 같이 제시한다.

$H_s^{d_v^{max}=8}$: (4,2;60),(6,2;0),(1,7;22),(2,7;81),
(0,9;20),(3,9;31),(5,11;40),(8,11;24)

$H_s^{d_v^{max}=12}$: (1,7;13),(2,7;43),(5,7;95),(8,7;61),
(9,7;79),(11,7;64),(0,9;25),(3,9;64),
(4,9;82),(6,9;56),(7,9;91),(10,9;0),
(1,11;49),(2,11;35),(5,11;16),
(8,11;81), (9,11;2),(11,11;50),

$H_{s_1}^{d_v^{max}=15}$: (0,9;11),(1,11;25),(2,11;33),
(3,9;45), (4,9;60), (5,11;81),
(6,9;30), (7,9;61), (8,11;55),
(9,11;78), (10,9;59), (11,11;34)

$H_{s_2}^{d_v^{max}=15}$: (2,11;61), (6,9;67), (7,9;3),
(9,11;14), (10,9;94), (11,11;70)

각각 8, 12, 15의 최대 차수를 갖도록 중첩 행렬을 설계하였으며 최대 차수 15의 경우 중첩 행렬을 두 개 사용하여 하나의 순환 행렬 자리에 이동 값을 3개 까지 갖도록 설정하였다.

하나의 부호어(codeword)가 하나의 프레임(frame)을 구성한다고 했을 때, 그림 13에서 확인할 수 있는 것처럼 최대 차수의 값이 커질수록 성능이 증가하는 경향을 보이고 있다. 이러한 성능 개선은 그림 12의 복호기를 적용하여 계층 복호를 수행하는 것이 가능하므로 처리율에서도 손해를 보지 않는다.

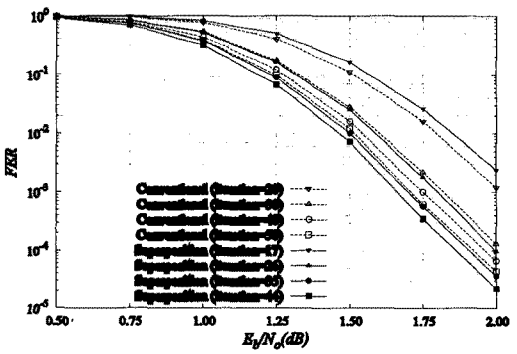


그림 14. 중첩 행렬 적용에 따른 수렴 속도 비교

4.2 중첩 행렬 적용에 따른 수렴 속도 비교

중첩구조를 적용하는 경우 에지의 수가 증가하기 때문에 연산량 역시 중첩 블록을 추가한 만큼 증가하게 된다. 기존 IEEE802.16e에 제시된 부호율 1/2 LDPC 부호의 경우 기저 행렬 내의 순환 행렬 수는 76개이고 수렴 속도를 비교하기 위해 사용한 부호의 경우 순환 행렬의 수는 86개이다. 연산량이

에지 수에 비례한다고 볼 수 있기 때문에 약 13.2%의 증가가 있는 셈이다. 공정한 비교를 위해 복호 과정에서 최대 반복 복호 수를 연산량 대비 감소시키게 되면 최대 50회의 반복 복호는 최대 44회로 줄여서 설정된 성능과 비교해 보면 된다.

제안된 중첩 행렬을 이용한 구조와 기존 IEEE802.16e에 제시된 LDPC 부호의 수렴 속도를 비교하기 위하여 기존 알고리즘을 40회, 30회, 20회의 반복 복호한 결과와 중첩 구조에서 동일한 연산량을 갖도록 감소시킨 35, 26, 17회의 반복 복호에 대한 성능 곡선을 그림 14에 도시하였다.

성능 결과를 보면 알 수 있듯이 중첩 행렬을 사용하는 경우 복호 초반에는 상대적으로 좋지 않은 성능을 보이지만 20회 후반부터 추월하여 복호 회수가 늘어날수록 성능 차이가 발생한다. 특히 기존 부호를 40회 반복 복호한 부호와 동일한 연산량을 갖도록 설정된 35회 복호 결과를 봐도 이미 기존 부호의 50회 반복 복호 성능을 넘어서는 결과를 보여 주고 있다. 따라서 동일한 연산량으로 고려하더라도 반복 복호 중반 이후에는 더 빠른 수렴 속도를 갖는 것을 알 수 있고, 연산량을 고려하지 않고 절대적인 수렴 속도만을 비교한다면 복호 초반에도 우월한 성능을 갖는다고 할 수 있다.

V. 결 론

제시된 중첩 알고리듬을 통해 효율적인 부호화와 간결한 부호 기술 등 준 순환 LDPC 부호로서의 장점뿐 아니라 계층 복호가 가능한 구조를 유지하면서도 원하는 차수 분포에 가깝게 부호를 설계하는 것이 가능하게 되었다. 이렇게 중첩 행렬을 이용하여 설계된 준 순환 LDPC 부호는 복호의 수렴 속도나 부호 성능의 개선이 향상되었는데, 본 논문에서 제안된 계층 복호기를 적용할 경우 기존의 표준에 제시된 부호를 지원함은 물론 중첩 행렬을 적용하여 새롭게 제안된 부호에 모두 적용 가능한 장점을 갖는다.

참 고 문 헌

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. IEEE Int. Conf. Commun.*, 1993, pp.1064-1070.

[2] D. Mackay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Elec. Lett.*, Vol.32, No.18, pp.1645-1646, 1996.

[3] IEEE Std. 802.16., "IEEE Standard for local and metropolitan area network part 16: Air interface for fixed and mobile broadband access systems," 2004.

[4] 3GPP, "Technical specification group radio access network multiplexing and channel coding (FDD)," 3GPP TS25.212, 1999.

[5] 3GPP2, "Physical layer standard for cdma2000 spread spectrum systems," Rev. D, C.S0002-D, Feb. 2004.

[6] IEEE Std. 802.11n/D2.00, "Draft IEEE Standard for Local Metropolitan Networks-Specific Requirements. Part 11: Wireless LAN MAC and PHY Specifications: Enhancements for Higher Throughput," Feb. 2007.

[7] European Telecommunications Standards Institute (ETSI). Digital Video Broadcasting (DVB) Second generation framing structure for broadband satellite applications; EN 302 307 V1.1.1.

[8] M. P. C. Fossorier, "Quasi-Cyclic Low-Density Parity-Check Codes From Circulant Permutation Matrices," IEEE Trans. on Information Theory, Vol.50, No.8, August 2004.

[9] J. Thorpe, "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs," IPN Progress Report 42-154

신 범 규 (Beomkyu Shin) 정회원

1999년 2월 서울대학교 전기공학부 공학사

1999년 3월~2002년 1월 (주) Locus, 연구원

2002년 1월~2003년 6월 (주) 휴맥스, 전임 연구원

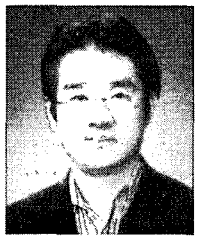
2004년 3월~현재 서울대학교 전기·컴퓨터공학부 석사 박사통합과정

<관심분야> LDPC 부호, 반복 복호 알고리듬, 통신 시스템

박 호 성 (Hosung Park) 정회원

2007년 2월 서울대학교 전기공학부 공학사
2009년 2월 서울대학교 전기·컴퓨터공학부 공학석사
2009년 3월~현재 서울대학교 전기·컴퓨터공학부 박사과정
<관심분야> 압축 센싱, LDPC 부호, 통신 시스템

김 상 효 (Sang-Hyo Kim) 종신회원



1998년 2월 서울대학교 전기공학부 공학사
2000년 2월 서울대학교 전기공학부 공학석사
2004년 2월 서울대학교 전기공학부 공학박사
2004년 3월~2006년 7월 삼성전자, 책임연구원
2006년 8월~2007년 8월 박사 후 연구원 (USC)
2007년 9월~현재 성균관대학교 정보통신공학부 조교수
<관심분야> LDPC 부호, 시공간 부호, MIMO, 릴레이 통신, 협력 통신

노 종 선 (Jong-Seon No) 종신회원

1981년 2월 서울대학교 전자공학과 공학사
1984년 2월 서울대학교 전자공학과 공학석사
1988년 2월 USC, 전기공학과 공학박사
1990년 9월~1999년 7월 건국대학교 전자공학과 부교수
1999년 8월~현재 서울대학교 전기·컴퓨터공학부 교수
<관심분야> 시퀀스, 시공간 부호, LDPC 부호, 암호학