工學碩士 學位論文

반복 부호의 저복잡도 및 고속 복호 알고리즘과 FPGA 구현에 관한 연구

A Study on Low Computational Complexity and High-Speed Algorithm of Iterative Codes and FPGA Implementation

指導敎授 鄭 智 元

2005年 **8**月

韓國海洋大學校 大學院

電 波 工 學 科

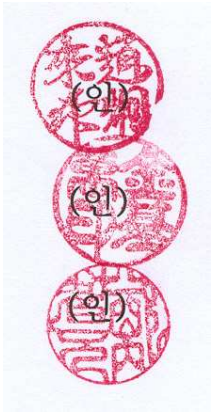
李 印 基

本 論文을 李印基의 工學碩士 學位論文으로 認准함.

委員長 : 工學博士 趙 炯 來

委 員 : 工學博士 金 基 萬

委 員 : 工學博士 鄭 智 元



2005年 8月

韓國海洋大學校 大學院

電 波 工 學 科

李 印 基

차 례

Abstract

Concatenate coding schemes are considered as being the best solution for powerful protection of digital information against nonlinear and fading noise channel. However, the performance of concatenate coding scheme is away from Shannon's limit. In 1993, Berrou and al introduced a new class of error correcting codes for digital transmission : the "turbo code". Turbo codes have been shown to perform near the capacity limit on the additive white Gaussian noise (AWGN) channels. As a powerful coding technique, turbo code offers great promise for improving the reliability of communication over wireless channels. Another interest area of channel coding scheme is LDPC(Low Density Parity Check) code. The high definition television(HDTV) satellite standard , known as the Digital Video Broadcasting (DVB-S2) transmission system employs a LDPC coding technique as a channel coding scheme. Unlike turbo codes, LDPC codes have easily parallelizable decoding algorithm which consists of simple operation such as addition, comparison and look-up table. Moreover the degree of parallelism is “adjustable” which makes it easy to trade-off throughput and complexity. However DVB-S2 system requires large block size and large number of iterations to near Shannon’s limit. The standard recommends that LDPC coded block size has 64800, and number of iteration is about 70 in the case of half coding rate. A large number of iterations for a large block size give rise to a large number of computation operations, mass power consumption, and decoding delay. It is necessary to reduce the iteration numbers and computation operations without performance degradation in order to implement with low power consumption.

This thesis proposes the two kinds of simplified complexity reduced algorithm. First, sequential decoding with partial group is proposed. It has same H/W complexity, and fewer number of iteration’s are required at same performance in comparison with conventional decoder algorithm. Secondly,

early detection method for reducing the computational complexity is proposed. Using a confidence criterion, some bit nodes and check node edges are detected early on during decoding. In this way, because early detected edges are not computed from following iterations, the computational complexity of further processing is reduced. However the encoder structure is very complicate. Therefore, it highly required channel coding scheme with simple encoder/decoder structure and good error performance in order to apply for wireless multimedia communications such as Ka-band satellite and wide-band mobile communication systems. Recently, there has been intensive focus on turbo product code(TPC) which has low latency and simple structures compare with turbo code. It achieve near-optimum performance at low signal-to-noise ratio. TPCs are two dimensional code constructed from small component codes. Different than original TPC decoder, which performs row and column decoding in a serial fashion, this thesis proposes a parallel decoder structure to reduce the latency. Furthermore, only one delay element is needed in contrast to two delay elements, i.e., decoding time is halved with this structure while maintaining the same performance level. Therefore, this thesis proposes the parallel TPC decoder and analyzes its performance. From the thesis, we describe the low latency and/or computational algorithm of three iterative codes.

Nomenclature

$\alpha(m)$: 수신 신호와 extrinsic 정보와의 표준편차

$\beta(m)$: 복호된 신호의 신뢰도

d_c : Parity Check matrix H 의 Column-weight

d_v : Parity Check matrix H 의 Row-weight

IT : 반복 회수

L_c : 채널 신뢰도

τ_{map} : MAP에서의 복호 시간

τ_{int} : 인터리버에서의 복호 시간

u_n : LDPC 복호기의 수신 신호

$v_{n\to k}$: bit nodes의 update 확률값

$w_{k\to n}$: Check nodes의 update 확률값

w_j : extrinsic 정보

$[w^{row}]$: 병렬 복호 시 row에 대한 extrinsic 정보

$[w^{col}]$: 병렬 복호 시 column에 대한 extrinsic 정보

제 1 장 서 론

연접부호는 높은 부호이득을 얻어낼 수 있으며, 무선통신 시스템에서 각광을 받고 있는 채널오류제어 기법이나 성능에 있어서 Shannon's Limit 와 다소 큰 격차를 보이고 있어, 이에 근접한 성능을 나타내는 Turbo 부호가 1993 년 Berrou 등에 의해 발표되었다[1]. 그러나 최근의 무선 통신 시스템은 고속 데이터 전송 및 동영상 등을 포함한 무선 멀티미디어 전송에 기반을 두고 있기 때문에, 고속 데이터 전송에 효율적이고 성능이 우수한 복호기 개발이 필수적이다. 현재의 Turbo 부호 응용 및 적용범위는 기존의 Viterbi 복호기와는 달리 처리할 데이터양과 연산작용이 매우 복잡해 저속 서비스에만 적용된다.

최근의 오류정정분야의 또 다른 연구는 LDPC(Low Density Parity Check)부호에 관심이 집중되고 있다. LDPC 부호는 터보 부호에 비해 복호화의 복잡도가 낮을 뿐 아니라 좋은 거리 특성으로 오류마류 현상이 나타나지 않고 완전 병렬 처리가 가능하다. 그러나 LDPC 는 복호화가 간단한 반면에 부호화 부분의 높은 복잡도가 LDPC 부호의 최대의 단점이다. Shannon 의 한계에 근접하기 위해서는 큰 블록 사이즈와 많은 반복횟수를 요구한다. 큰 블록 사이즈와 많은 반복 회수는 많은 계산량과 power 소모량을 요구하므로 성능 손실 없이

반복횟수를 줄일 수 있는 Subset 방법을 이용한 복호 알고리즘, 그리고 early stop 알고리즘에 대해 연구 하였고, 비트 노드 계산과 체크 노드 계산 시 일정한 신뢰도 값보다 크면 다음 반복 시 계산을 하지 않는 early detection 알고리즘에 대해 연구 하였다.

최근 작은 블록 크기를 가로 세로로 product 시킨 후 같은 복잡도로써 많은 블럭 크기의 효과를 얻을 수 있고 Turbo 부호에 비하여 복호기가 간단하여 고속 구현이 가능한 TPC 관심을 받고 있다. 그러나 TPC 복호기의 가장 큰 문제점은 두 개 복호기가 직렬로 연결된 구조에서는 복호기 두 개를 병렬로 처리할 수 없다는 것이 가장 큰 단점이다.

본 논문에서는 이러한 강력한 에러 정정 능력을 가진 반복부호의 효율적인 복호기법을 제안한다. 제 2 장에서는 블록부호를 기반으로 하는 반복부호의 효율적인 구현방안으로 LDPC 에서는 DVB-S2 규격안에의 성능 열화가 없는 저복잡도 알고리즘을, 그리고 TPC 에서는 고속 복호기를 제안하며 또한 VHDL 을 이용하여 구현하였다. 제 3 장에서는 트렐리스를 기반으로 하는 반복부호로써 Turbo 부호의 구현 방안과 기존의 방식이 가지는 문제점을 지적하여 고속 복호기로 구현할 수 있는 방안을 제시한다. 그리고 제 4 장의 결론으로 본 논문의 끝을 맺는다.

제 2 장 블록부호를 이용한 반복부호

2.1 Low Density Parity Check Codes

Turbo 부호의 경우 부호화 과정이 간단한 반면에 복호화 과정이 복잡하여 하드웨어 구현에 있어서 어려움을 겪고 있다. 반면에 LDPC 부호는 1962년 Gallager에 의해 발표된 이후, 1996년 Mackay 와 Neal, 그리고 Sipser 와 Spielman에 의해 우수성이 재발견 되어 현재 Turbo 부호와 함께, Shannon의 채널 용량 한계에 근접하는 성능을 가지는 반복 복호 방식의 부호로 각광 받고 있다[5]. LDPC 부호는 터보 부호에 비해 복호화의 복잡도가 낮을 뿐 아니라 좋은 거리 특성으로 오류마루 현상이 나타나지 않고, 완전 병렬 처리로 고속 처리가 가능한 장점이 있다. 반면에 부호화의 높은 복잡도가 LDPC 부호의 중요한 문제점이었으나 최근에 삼각행렬 분해법, Linear-congruence 방법을 사용하여 부호화기를 간단하게 하였다. LDPC 복호 과정을 위한 파라미터는 다음과 같다.

$$\begin{cases} K & : \text{message 벡터의 크기} \\ N & : \text{code word 벡터의 크기} \\ M & : \text{parity check 의 크기} \end{cases}$$

여기서, $M = N - K$ 이다.

$\mathbf{u}$ 를 message row vector ($\mathbf{u} = \{u_0, u_1, ..., u_{K-1}\}$)라 하고 $\mathbf{x}$ 는 code word vector ($\mathbf{x} = \{x_0, x_1, ..., x_{N-1}\}$)라 두면,

$\mathbf{x} = \mathbf{G}^T \mathbf{u}^T$ $(N \times K)(K \times 1)$ 이다.

여기서 $\mathbf{r}$ 은: received vector ($\mathbf{r} = \{r_0, r_1, ..., r_{N-1}\}$)이며, 잡음이 첨가된 수신 신호이다.

$\mathbf{r} = \mathbf{x} + \mathbf{e}$ ($\mathbf{e}$:noise vector) (2-1)

$\mathbf{H}$ 를 parity check matrix라 부르며, H 행렬 구성은 아래와 같다.

$$\mathbf{H} = \underbrace{\begin{bmatrix} \quad \end{bmatrix}}_{N} \Big\} M$$

$\mathbf{S}$ 를 Syndrome matrix 라 할 때, 수신측에서는 다음과 같은 식이 성립되어야 한다.

$\mathbf{S} = \mathbf{Hr} = \mathbf{H}(\mathbf{G}^T \mathbf{u}^T + \mathbf{e}) = \mathbf{He}$ $(\mathbf{HG}^T = 0)$ (2-2)

$\hat{\mathbf{x}} = \mathbf{r} + \hat{\mathbf{e}}$ ($\hat{\mathbf{e}}$: $\mathbf{S}$ 로 부터 estimate된 noise vector, $\hat{\mathbf{x}}$: decoded codeword vector)

그러므로 $\mathbf{S} = \mathbf{0}$ 이 될 때까지 계속 반복하면서 복호한다.

예들 들어 그림 2.1은 codeword 길이가 8, 부호화율이 1/2인 parity

check matrix이다. 여기서 $n_1, n_2, \cdots, n_8$ 등을 비트 노드(bit nodes), m_1, m_2, m_3, m_4 등을 체크 노드(check nodes)라 하며, 비트 노드의 개수를 column-weight, 체크 노드의 개수를 row-weight라 한다.

$$H = \begin{array}{c} \begin{matrix} n_1 & n_2 & n_3 & n_4 & n_5 & n_6 & n_7 & n_8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{matrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{matrix} \end{array}$$

그림 2.1 Parity check matrix의 예

Fig. 2.1 The example of the parity check matrix.

2.1.1 LDPC 복호 알고리즘

복호기의 목적은 전송된 비트들의 값을 결정하는 것이다. LDPC 복호기는 비트 노드와 체크 노드사이에서 각각의 확률 값을 구하며 각각에게 향상된 정보를 전달하는 방식이다. LDPC 복호의 순서는 총 세단계로 나눌 수 있다.

단계 1. 초기화(initialization)

복호 과정은 먼저 비트 노드의 값을 이용하여 체크 노드의 유도함으로써 시작한다. 그러기 위해 우선 비트노드의 값을 수신 채널 값으로 초기화 해야 한다. 다음 수식 (2-1)은 수신 심볼의 초기 채널 값이다.

$$u_n = -L_c \cdot r_n \quad (L_c = \frac{2}{\sigma^2}), \qquad n = (0,1,\ldots,N-1) \tag{2-3}$$

단계 2 : Check Node Update(CNU)

다음 그림2.2(a)는 비트 노드에서 row-weight의 수만큼의 비트 노드의 값을 이용하여 체크 노드 확률을 구하는 그림이다. n_{dc} 개의 row-weight를 가진다고 가정할 때, 각각의 체크 노드로 들어오는 비트들의 확률 $.w_{k\to n_1}, w_{k\to n_2}, .., .w_{k\to n_{dc}}$ 은 아래 수식(2-2)와 같이 유도된다.

$$w_{k\to n_1} = g(v_{n_1\to k}, v_{n_2\to k,}, \ldots, v_{n_{t-1}\to k,} v_{n_{t+1}\to k,} \ldots, v_{n_{dc}\to k}) \tag{2-4}$$

여기서, g함수의 정의는 수식(2-3)과 같다.

$$.g(a,b) = sign(a)\times \sin g(b)\times\{\min(|a|,|b|)\} + LUT_g(a,b) \tag{2-5}$$

$$LUT_g(a,b) = \log\left(1+e^{-|a+b|}\right) - \log\left(1+e^{-|a-b|}\right)$$

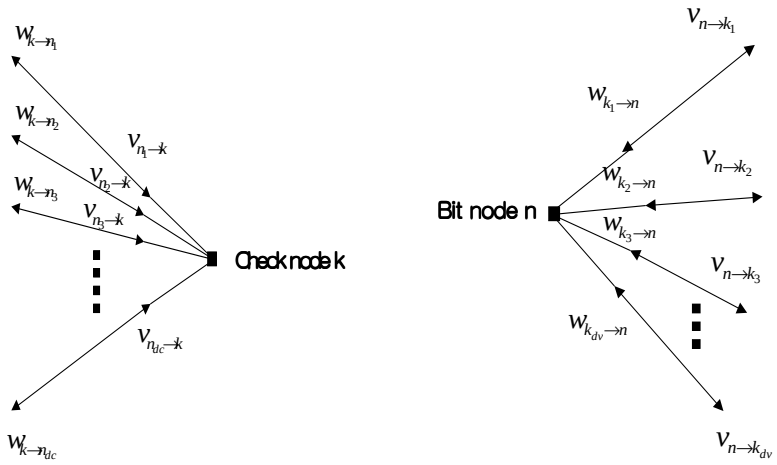
구현적인 측면에서 볼 때, $LUT_g(a,b)$ 는 ROM(read only memory)을 이용하여 간단히 구현될 수 있다.

단계 3. Bit Node Update(BNU)

다음 그림2.2(b)는 체크 노드에서 column-weight의 수만큼의 체크 노드의 값을 이용하여 비트 노드 확률을 구하는 그림이다. 각 비트

노드에 연결된 d_v 개의 확률, 즉, 각 컬럼에 해당하는 d_v 개의 비트 노드 $v_{n\to k_1}, v_{n\to k_2}, \ldots, v_{n\to k_{dv}}$ 의 확률을 다음 수식(2-4) 에서 구한다.

$$v_{n\to k_i} = u_n + \sum_{j\neq i} w_{k_i\to n} \tag{2-6}$$



(a) (b)

그림 2.2 비트 노드와 체크 노드에서의 데이터 연산 과정

(a) CNU 과정 (b) BNU 과정

Fig. 2.2 Message update at bit and check nodes.

(a) Message update at check nodes (b) Message update at bit nodes

2.1.1.1 성능분석

그림 2.3은 parity accumulator address 부호 방식을 이용하여 N=64800 일 때 iteration을 40회 했을 때의 E_s / N_0 에 따른 성능 곡선을 나타내었다. R=1/2 일 때 약 0.9[dB]에서 오류가 free 함을 알 수 있다.

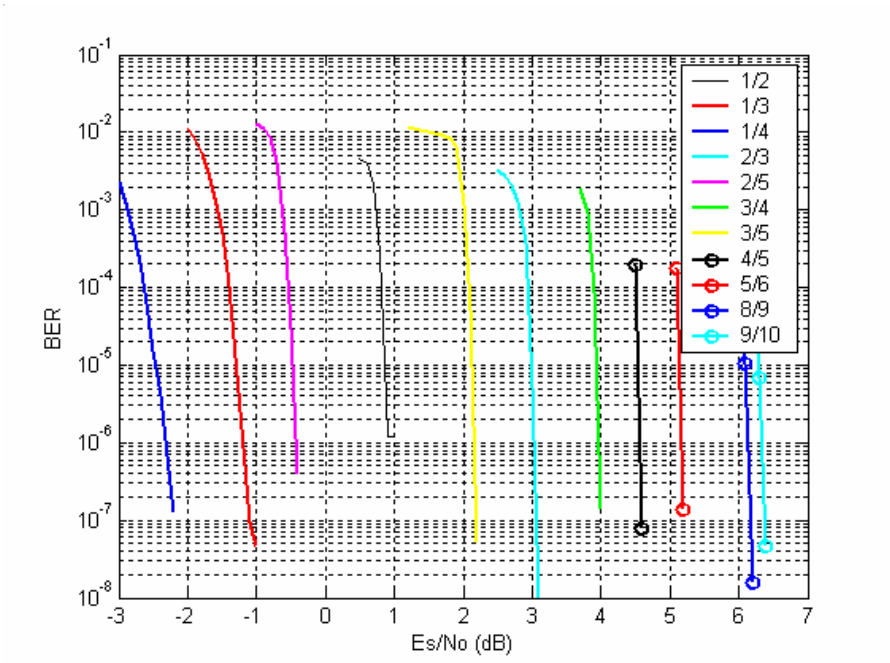


그림 2.3 부호율에 따른 성능 곡선(iteration=40)

Fig. 2.3 Simulation results accoding to coding rate.

2.1.2 Low Computational Complexity 알고리즘

DVB-S2 표준안에서 ㅊode word의 길이 N=64800이며, 대략 40~70회 반큼의 반복회수를 권고한다. 이는 많은 계산량과 큰 전력 손실을 요구한다. 따라서 본 절에서는 성능 선실 없이 반복회수를 줄일 수 있는 subset 방식을 이용한 복호 알고리즘, 그리고 early etop 알고리즘을 연구하였고, 비트 노드 계산과 체크 노드 계산 시 일정한 기준값 초과 시 계산을 하지 않는 early detection 알고리즘에 대해 연구하였다.

2.1.2.1 Sequential Decoding with partial Group

LDPC 의 장점은 복호 시 병렬화가 가능하여 복호 속도가 매우 빠르다는 것이다. 그러나 많은 비트 노드나 체크 노드 계산 시 병렬화 처리를 하면 하드웨어 크기나 power 소모량이 매우 클 수 있다는 단점을 가지고 있다. 그러므로 본 절에서는 check node 를 몇 개의 부분집합으로 나뉘어 복호하는 방식에 대해 논한다.

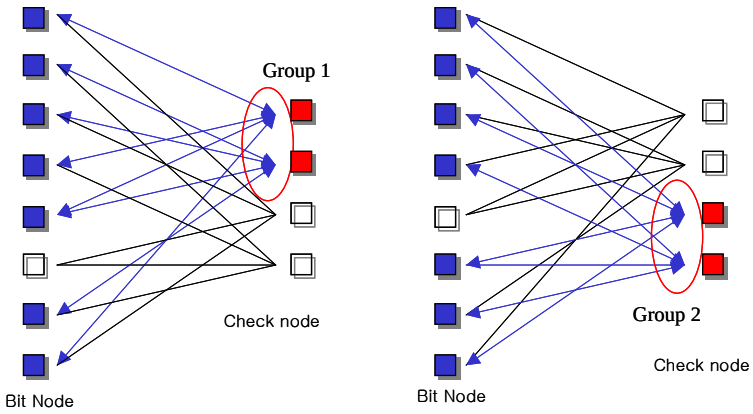


그림 2.4 sequential decoding 과정

(a) 첫번째 부분 집합의 업데이트 (a) 두번째 부분 집합의 업데이트

Fig. 2.4 The process of sequential decoding.

(a) The update of the first subset (b) The update of the second subset

앞 절에서 살펴보았듯이 일반적인 LDPC 복호기는 BNU block 과 CNU block 으로 구성된다. Bit to check block 블록은 동시에 병렬로 계산된다. 허나 그림 2.4 에서와 같이 sequential decoding algorithm 을 이용함으로써 동일한 성능에서 계산량의 감소를 가져 올 수 있다. 먼저 체크 노드를 p 개의 부분집합으로 나눈다. 각각의 부분집합은 n 개의 체크 노드를 가지고 있다. P=1 이면 부분집합으로 나누지 않은 기존의 복호 방식과 동일하다. 이 때 총 체크 노드의 수 $M = n \times p$ 이다. 복호 과정은 먼저 첫번째 부분 집합의 체크 노드를 업데이트 한다. 이후

업데이트된 체크 노드를 통해서 비트 노드를 업데이트(그림 2.4(a)) 하며 그 뒤 다른 부분집합에서의 체크 노드를 업데이트 하게 된다(그림 2.4(b)). 이때 부분집합을 나눌 때 각각의 check node 들과 bit node 들과의 edge 를 평균적으로 분포하는 것이 중요하다. 이에 따른 절차는 그림 2-5 와 같다.

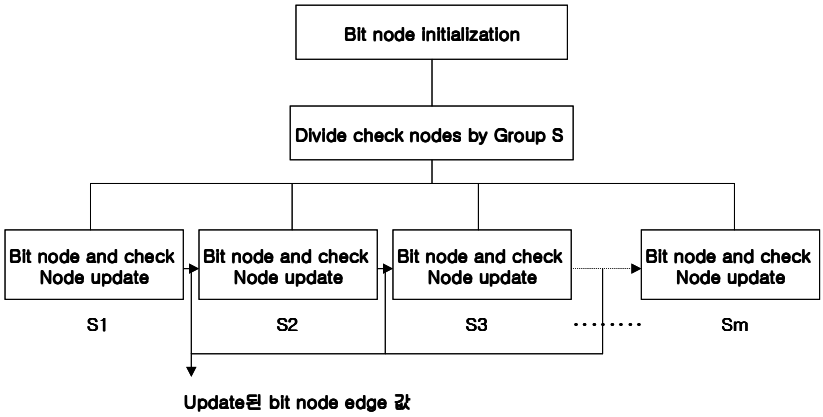


그림 2.5 sequential decoding 절차

Fig. 2.5 The procedure of sequential decoding.

이는 turbo 부호의 복호 시 복호기 두개가 직렬로 연결된 (serially concatenated)구조와 동일하다. 직렬로 연결된 구조에서 첫 번째 복호기가 복호를 한 후 외부정보(extrinsic information)를 두 번째 복호기에 전달되면서 복호 과정이 진행된다. 마찬가지로 sequential

decoding 과정 또한 p 개의 복호 모듈로 나뉘어 각각의 복호를 마친 후(bit node, check node), 다음 복호 모듈로 전달된다. 이는 이미 bit node update 결과 후 다음의 부분집합으로 전달하기 때문에 기존의 방식에 비해 성능열화 없이 같은 복호 시간에 $N_r/2$ 만큼의 반복회수만 필요하다.(N_r : 기존 방식의 반복횟수)

다음 그림 2-6 은 부분집합 (subset=2) 경우의 sequential decoder 와 일반적인 복호기와의 BER 성능 곡선이다. sequential decoding algorithm 을 사용하였을 경우 1/2 의 iteration 을 때 성능이 일반적인 복호기일 때와 일치함을 확인할 수 있다.

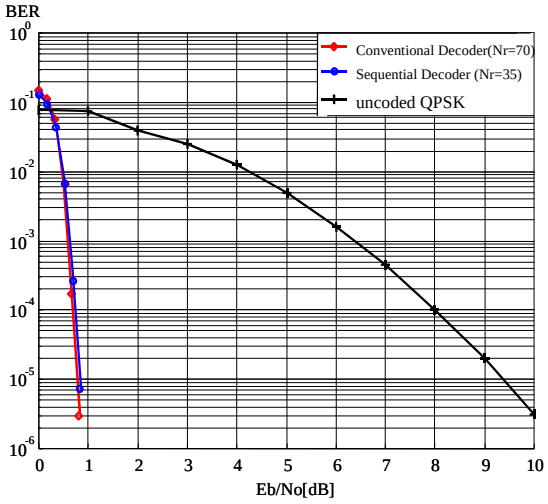


그림 2.6 sequential decoder 와 conventional decoder 의 성능곡선

Fig. 2.6 BER curve of sequential decoder and conventional decoder.

2.1.2.2 Early-stop algorithm

Shannon 의 한계에 근접하기 위해서는 많은 블록 사이즈와 많은 횟수의 반복 회수를 요구한다. 많은 횟수의 반복회수는 power 소모량을 증가시킬 뿐만 아니라 복호 속도 또한 느리게 한다. DVB-S2 권고안에서는 각 부호화율에 반복횟수를 정해 놓고 있으며, 약 40~70 회 설정하고 있다. LDPC 와 같은 반복 부호는 일정한 반복 후에는 더 이상 성능이 개선되지 않는 현상이 있으며, 이는 복호 속도 저하 및 power 소모량만 증가시키고는 결과만 맞고 있다. 따라서 적절한 early-stop algorithm 을 이용하여 더 이상의 성능 개선이 없으면 반복을 중지시키는 알고리즘을 그림 2.7 과 같이 구성해야 한다.

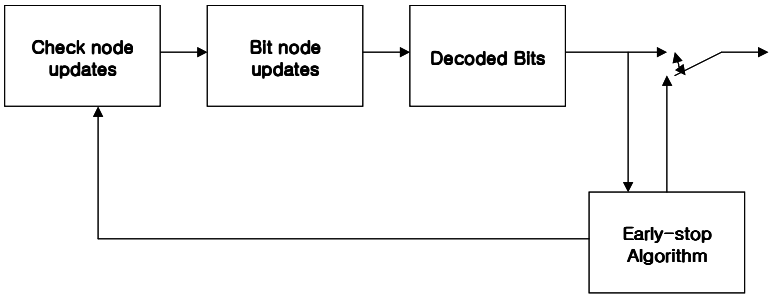


그림 2.7 Early-stop algorithm 블록도

Fig. 2.7 Block diagram of early-stop algorithm.

본 논문에서는 두 가지 알고리즘을 제시한다. 각 iteration 순간에 나온 LLR 값을 Hard decision 하여 비교하는 방식과 parity check equation 을 이용하는 방식이다.

A. Hard decision algorithm

각 부호화율에 따른 iteration 을 마치면 LLR 을 Hard decision 하여 decoded bit 를 판정한다. Iteration 을 거치면서 정정된 bit 가 있다면 LLR 의 비트에 변화가 있을 것이고 더 이상 정정할 것이 없다면 iteration 이 계속되더라도 더 이상의 변화는 없을 것이다. 따라서 각 iteration 일 때의 LLR 을 H·D 하여 그 다음 iteration 과 비교 더 이상의 변화가 없을 경우 decoding 을 종료하고 최종적으로 나오 LLR 을 이용하여 복호한다. 그림 2.8(a) 는 Hard decision algorithm 의 블록도 이다.

B. Parity check equation algorithm

비트 노드 업데이트 후 각 비트 노드에서 체크 노드로 가는 값과 체크 노드에서 그 비트 노드로 가는 값의 합을 hard decision 한다. 구해진 N 개의 값 중 정보비트 부분(K 개)을 다시 re-encoding 하면 새로운 parity bit (M 개)가 생성된다. 이 parity bit 와 hard decision 으로

구해진 parity bit 를 비교하여 같다면 iteration 을 중지한다. 그림 2.8(b) 는 Parity check equation algorithm 의 블록도 이다.

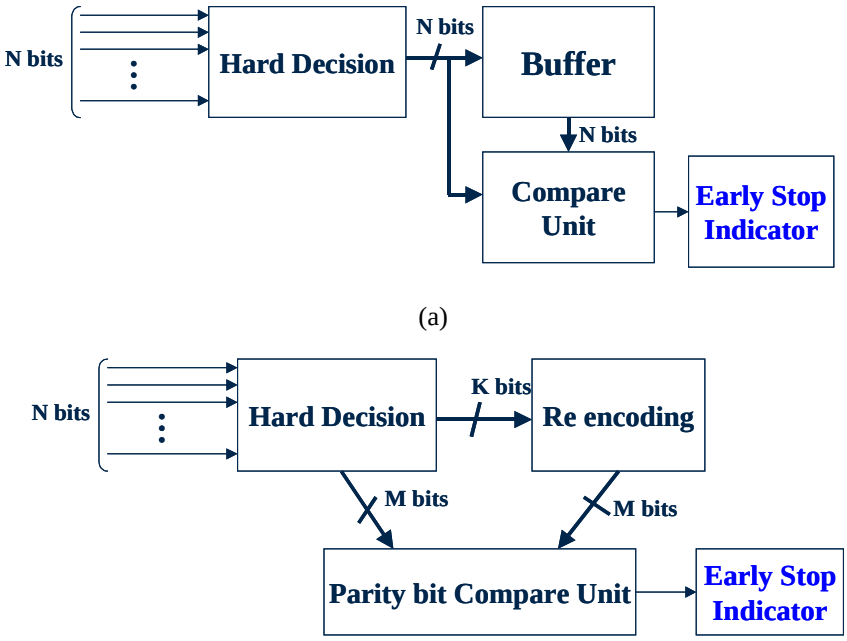


그림 2.8 두 가지 방식의 early-stop algorithm 블록도

(a) Hard decision algorithm (b) Parity check equation algorithm

Fig. 2.8 Block diagram of two types of early-stop algorithm.

(a) Hard decision algorithm (b) Parity check equation algorithm

표 2.1 Hard decision method 일 때의 E_b / N_0 에 따른
평균 정지된 iteration 횟수와 speed 증가율

Table 2.1 The average number of iteration according to E_b / N_0 and speed increment rate at the hard decision method.

E_b / N_0	Hard decision	
	Iteration(60 회)	Increment of decoder speed
0.8	49.52	17.46%
0.825	45.88	23.53%
0.85	43.11	28.15%
0.875	40.53	32.45%
0.9	38.53	35.78%

표 2.2. Parity check equation 일 때의 E_b / N_0 에 따른
평균 정지된 iteration 횟수와 speed 증가율

Table 2.2 The average number of iteration according to E_b / N_0 and speed increment rate at the parity check equation method.

E_b / N_0	Parity check equation	
	Iteration(60 회)	Increment of decoder speed
0.8	48.56	19.07%
0.825	45.04	24.93%
0.85	42.15	29.75%
0.875	39.86	33.57%
0.9	37.79	37.07%

위의 표 2.1 과 2.2 는 두 가지 방식의 early stop algorithm 을 적용하였을 시 iteration 수이며 그에 따른 속도의 증가량을 나타내고 있다. Parity check equation 방식을 사용하였을 때 약 1 회 iteration 을 적게 함을 알 수 있다. 그림 2.9 에서 보듯이 iteration 60 회 한 BER 성능과 위의 2 가지 알고리즘을 적용하였을 때의 성능 곡선이 거의 일치함을 알 수 있다.

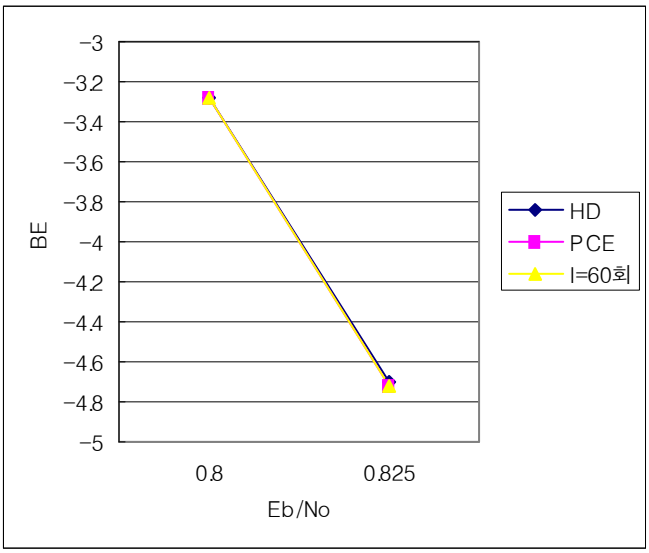


그림 2.9 early stop algorithm 의 BER 성능곡선

Fig. 2.9 BER curve of two types of early stop algorithm.

2.1.2.3 early Detection Method

복호 과정의 복잡한 계산량을 줄이는 또 다른 방법은 early detection 방식이다. 이 Early detection 방식은 높은 log likelihood 값을 지닌 비트 노드나 체크 노드는 신뢰할 수 있다는 개념을 기반으로 한다. 따라서 결정된 비트 노드나 체크 노드는 다음 iteration 에서 계산하지 않는다. LDPC 복호기에 사용되는 종전의 sum product algorithm 은 뛰어난 BER 성능을 가지나 많은 계산 과정이 필요하다. 따라서 계산량을 줄일 수 있는 방법이 필요하게 된다. 비트 노드나 체크 노드를 업데이트 할 때 일정한 기준 값을 넘게 되면 그 값은 더 이상 iteration 이후의 업데이트를 하지 않더라도 신뢰할 만한 값으로 사용 가능 하게 된다. Early detection 된 node 를 bipartite graph 로 보면 그림 2.10 과 같다.

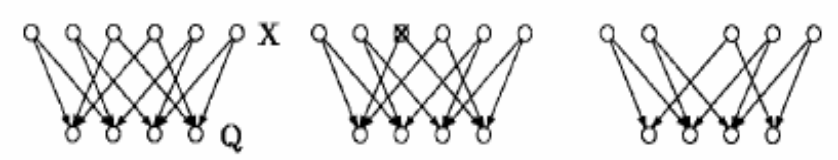


그림 2.10 Bipartite graph

Fig. 2.10 Bipartite graph.

LDPC 복호기에서 early detection 하기 위한 과정은 다음과 같다.

Step 1. initialization

모든 Detected Check node, bit node 를 0 으로 초기화 시킨다.

Step 2. Check node updates

체크 노드의 값이 일정한 기준 값(T_c) 이상 일 때 Detected check 를 1 로 설정하고 그렇지 않을 경우 check node update 공식에 따라 계산한다.

Step 3. Bit node updates

비트 노드의 값이 일정한 기준 값(T_b) 이상 일 때 Detected node 를 1 로 설정하고 그렇지 않을 경우 bit node update 공식에 따라 계산한다.

이 때 기준값(T_c)을 너무 높게 잡으면 계산량이 급격하게 줄어드나 성능이 떨어지며 기준값을 낮게 잡으면 반대로 성능열화는 없으나 계산량이 거의 줄어 들지 않으므로 이 기준값을 적당한 값으로 설정하는 것이 중요하다. 그림 2.11 에서 T_c =18 이상일 때 일반적인 LDPC 복호기와 성능이 거의 일치함을 알 수 있다.

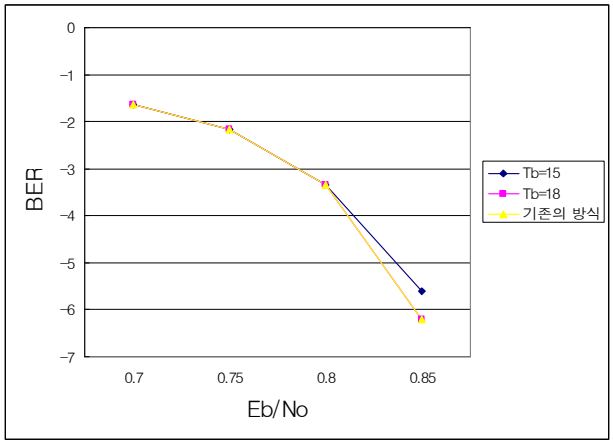


그림 2.11 T_c 의 변화에 따른 BER 성능곡선

Fig. 2.11 Simulation results according to the change of T_c .

표 2.3 는 LDPC decoder 에서 요구되는 계산량의 나열한 것이다.

표 2.3 LDPC decoder 에 필요한 계산량

Table 2.3 Decoding complexity for conventional and early detected method.

	Number of operations for conventional decoder	Number of operations for early detection method
Check nodes(g(a,b))	$\left(M\times\left(d_c-1\right)\right)\times N_r$	$\left(M\times\left(d_c-d_c^*-1\right)\right)\times N_r$
Bit nodes(additions)	$\left(N\times\left(d_v-1\right)\right)\times N_r$	$\left(\left(N-N^*\right)\times d_v\right)\times N_r$

여기서 d_c^*는 각 check node 당 early detected 한 edge 의 평균 숫자이고, N^* 는 bit node 를 early detected 한 평균 숫자이다. $T_b = 18, T_c = 10$으로 고정하였을 때 표 2.4 는 E_s / N_0가 1dB 일 때 각 iteration 횟수에 따른 d_c^*와 N^*의 숫자를 보여준다. 이 때 N=64800, K=32400, M=32400, 이며 d_c=7, d_v=13 이다. 표 2.4 에서 볼 수 있듯이 early detected 방식의 계산량은 기존의 방식과 비교했을 경우 체크노드 계산 시 50%, 비트 노드 계산 시 99% 정도의 감소를 가져온다.

표 2.4 early detected 된 수

Table 2.4 The number of early detected edges.

N_r	d_c^* / M	N^*
10	0.387	491.84
20	1.484	19048.59
30	2.619	62372.86
40	3.29	64334.73
50	3.6068	64335.46
60	3.852	64335.46

2.2 Turbo Product Codes

TPC는 두개 혹은 그 이상의 짧은 길이의 블록 부호로 구성된다. 그림 2.12은 product code의 구성도를 나타낸다.

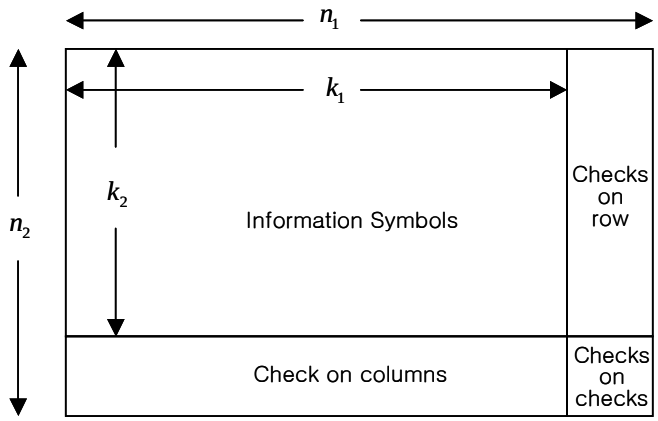


그림 2.12 TPC 부호화기 구성도 ($P=C_1\times C_2$)

Fig. 2.12 Encoder construction of turbo product code ($P=C_1\times C_2$) .

그림 2.12과 같이 두 개의 블록 부호를 적용할 경우, k_1 (또는 k_2)개의 정보 비트를 가로(또는 세로)로 배치한 후, 가로는 (n_1, k_1, δ_1) 을 가지는 블록 코드 C_1 으로 부호화 하고, 세로는 (n_2, k_2, δ_2) 를 가지는 C_2 로 부호화 한다. 따라서 TPC $P=C_1\times C_2$ 이므로, (n,k,δ) 를

가진다. 여기서 n은 cordword의 길이이며 k는 정보 심벌의 수, δ 은 minimum Hamming distance 를 나타낸다. TPC에 적용되는 블록 부호 C_1, C_2 는 해밍부호, BCH부호, RS부호 등 다양한 블록 부호를 적용시킬 수 있다.

2.2.1 고속 복호 알고리즘

기존의 TPC 복호기의 구조는 아래 그림 2.13와 같다.

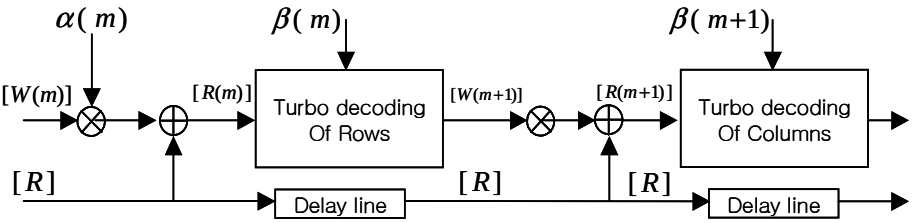


그림 2.13 기존의 TPC 복호기 구조

Fig. 2.13 Conventional TPC decoder structure.

여기서 W(m)은 m번째 iteration시의 외부 정보이며, [R]은 수신 심벌, α 는 reliability factor, β 는 scaling factor이다. 즉 TPC 복호기는 Row열의 블록 복호를 하여 update되는 외부 정보를 다음 column의

블록 복호 단에 주어서 복호를 하게 되며 여기 나온 외부정보가 다음 iteration시 Row에 입력되면서 에러를 정정한다.

2.2.1.1 Parallel Algorithm

앞에서 살펴본 그림 2.13과 같이 Pyndiah에 의해 제안된 복호기[14]는 세로(또는 가로)를 다음 복호하기 전에 반드시 가로(또는 세로)를 복호해야 하는데 반해, 지연을 감소 시키기 위한 본 논문에서 제안된 병렬 TPC복호기 구조는 그림 2.14와 같다. 그림 2.14에서와 같이 가로 세로 복호기에 대한 복호는 병렬로 이루어져지며, 복호 지연은 기존의 방식에 비해 절반 효과를 갖기 때문에 복호 속도는 2배 개선 시킨다. 메트릭 $[W^{row}]$ 와 $[W^{col}]$ 은 가로와 세로 부호에 대한 수신 신호의 extrinsic 정보 이며, 각 블록에 대한 복호를 마치고 block-by-block으로 같은 시간에 외부 정보의 교환으로 복호가 수행되어진다.

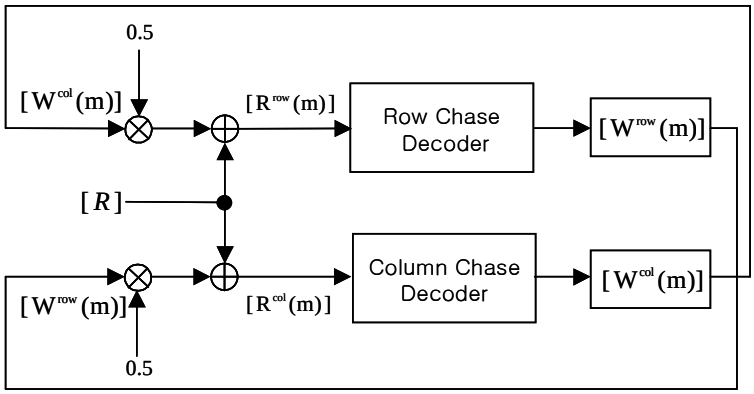


그림 2.14 병렬 복호기 구조 및 복호 과정

Fig. 2.14 Parallel decoder structure and decoder process.

2.2.1.2 Early Stop Algorithm

복호기는 여러 번의 반복회수에 의해 성능이 개선 되어 진다. 그러나 어느 일정한 반복회수가 지나면 에러율은 더 이상 개선되지 않는다(error floor 현상). 이런 경우 더 이상의 반복은 의미가 없으며 이는 복호 속도만 저하시키는 결과를 초래한다. 본 논문에서는 채널 추정된 결과 및 적정한 알고리즘을 바탕으로 더 이상 반복하지 않고 반복을 중지하는 알고리즘을 제안한다. 직렬 또는 병렬로 연결된 두개의 row decoder와 column decoder에서 출력된 soft decision값을 hard decision하여 복호된 비트 값과 비교 후 오류 개수를 체크한 후에 더 이상 오류를 정정하지 않으면 반복을 종료하는 알고리즘이다. 표 2.5

에서 알 수 있듯이 early-stop 알고리즘을 이용하였을 때 E_b / N_0 =4dB를 기준으로 약 65%의 복호 속도 향상을 가져옴을 확인 할 수 있었다.

표 2.5 Early-stop 알고리즘을 이용한 평균 반복회수

(반복회수 6 회, BCH(31,26,3)TPC)

Table 2.5 The average number of iteration using Early-stop.

(iteration 6, BCH(31,26,3)TPC)

E_b / N_0	평균 반복회수	복호속도 향상도(%)
1	6	0
2	5.986	0.2
3	4.82	19.7
4	2.049	65.85

본 논문에서 모의 실험 시 부호화율이 같은 두개의 BCH 부호를 사용하여 구성하였다. Chase 알고리즘을 위해서 가장 신뢰성이 없는 비트를 선택 시 p=4로 하였다. 그리고 반복 복호를 위한 scaling factor로써 $\alpha = 0.5$, $\beta = 1$ 로 고정하였다. β 는 모든 반복시 1로 고정하였기 때문에 Chase 복호기의 출력 decoder의 연판정 값을 구하기 위한 β 의 곱셈이 필요치 않으며, α 는 0.5로 고정하였기 때문에 가로

세로에 대한 복호기의 외부 정보의 절반에 해당하는 비트만큼 우측으로 shift하면 되기 때문에 H/W구현이 간단해 진다. 이러한 조건을 중심으로 반복회수를 4로 하였을 때 결과는 그림 2.15와 같다. $BER = 10^{-4}$ 을 기준으로 기존의 방식과 비교하였을 때 본 논문에서 제안한 방식은 0.2dB 이내의 성능감소가 발생된다. 반복회수가 동일 할 때 기존의 방식과 비교하여 성능의 감소가 일어나는 것은 첫번째 반복 시 기존의 방식은 row 복호 이후 발생된 외부 정보가 column 복호에 제공되지만 본 논문에서 제안한 방식은 column 복호에 제공되는 외부 정보가 0이기 때문이다.

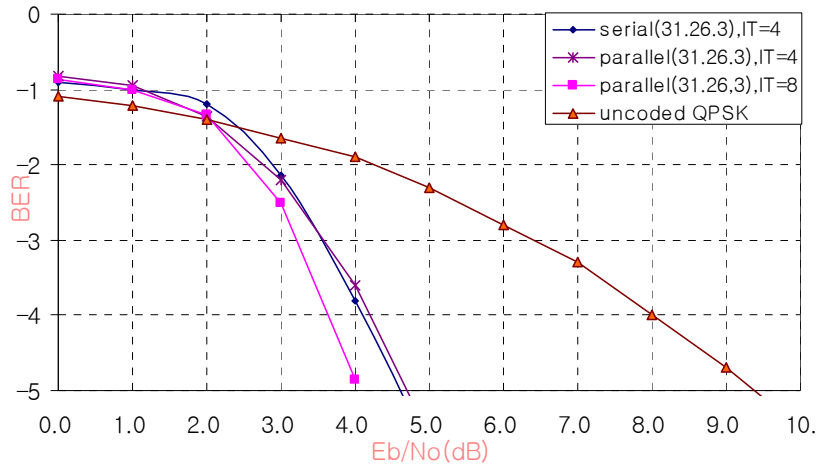


그림 2.15 기존방식과 병렬 복호방식의 성능비교(IT는 반복복호의 횟수)

Fig. 2.15 Performance of original and parallel decoded BCH TPCs on AWGN channel. (IT denotes number of iteration)

2.2.2 VHDL 모델링

본 논문에서는 고속화 방안들을 적용하여 VHDL(Very high speed integrated circuit Hardware Description Language)코드를 Altera사의 Design complier를 이용하여 시뮬레이션 하였다. 고속 복호기는 timing 시뮬레이션을 위해 APEX20KE(EP20K300EQC240-2)칩을 사용하였으며 기존의 직렬 복호기일 때와 본 논문에서 제안하는 병렬 고속 복호기를 비교 하였다. 다음 표 2.6에서 알 수 있듯이 decoding speed 는 기준으로 기존의 복호기보다 약 2배의 속도 향상을 가져 오고 요구되는 메모리(Logic cell)은 약 1.5배 정도 더 필요함을 알 수 있다.

표 2.6 복호속도와 메모리 비교

($BCH(15\times11)^2$, iteration=1, clock period = 10ns)

Table 2.6 Decoding Speed and Memory Comparision($BCH(15\times11)^2$, iteration=1, clock period = 10ns).

	Conventional algorithm(p=1)	Parallel algorithm
Memory(logic cell)	7.024×10^3	$10.1\times10_3$
Decoding speed(μs)	23.4	12.7

본 논문에서 제안하는 복호기의 타이밍 시뮬레이션 결과는 그림 2.15에 나타내었다. 클럭 주기는 15[ns]이고 복호 박식은 $BCH(15\times11)^2$ 을 적용하였다. 5비트로 양자화된 수신 비트를 가지고 TPC decoder를 통과한 후 1비트의 복호 비트가 출력이 됨을 알 수 있다.

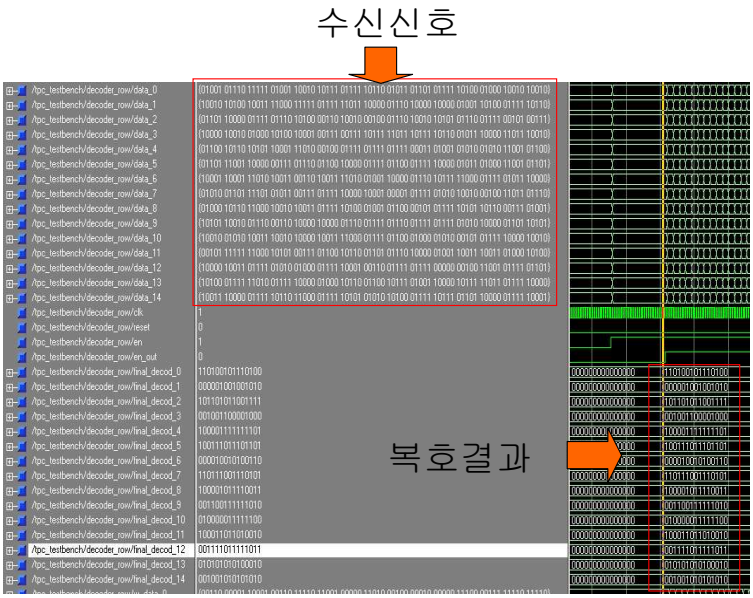


그림 2.16 TPC 복호기 타이밍 출력도

Fig. 2.16 The timing diagram of TPC Decoder.

위의 알고리즘을 바탕으로 VHDL을 통하여 직접 FPGA보드에 프로그래밍 하였으며 그림 2.17은 이때 사용한 FPGA 보드이다. 이 보드 상에서의 최종 복호 신호를 그림 2.18에서 처럼 디지털 오실로스코프를 이용하여 측정하였다.

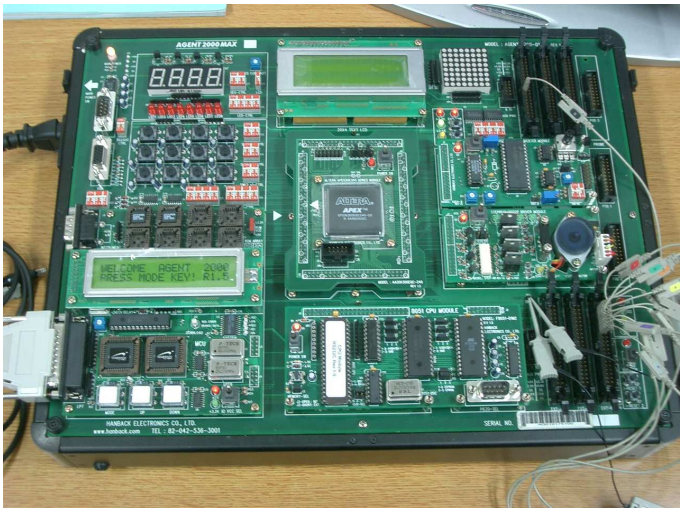


그림 2.17 FPGA 보드

Fig 2.17 The board of FPGA.

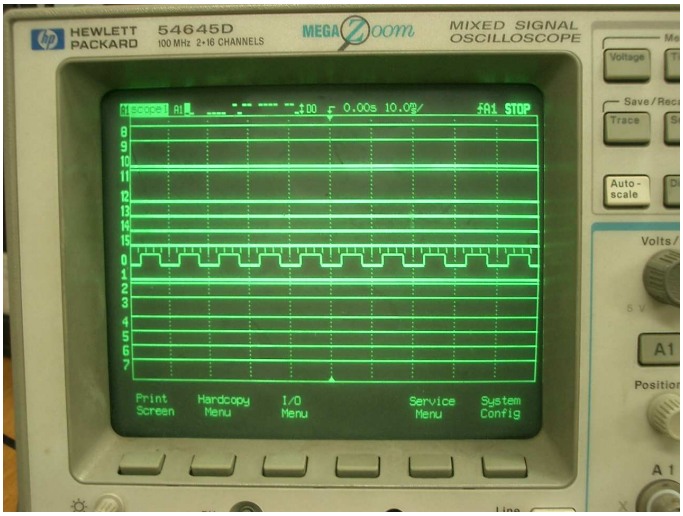


그림 2.18 디지털 오실로스코프에서의 측정 결과

Fig. 2.18 The result of measurement at the Digital Oscilloscope.

제 3 장 트렐리스 기반의 반복 부호

3.1 Turbo Codes

Turbo 부호는 Shannon's Limit에 근접하는 성능을 나태낼 수 있는데, E_b / N_0 0.7dB 부호율 1/2에서 비트오류확률 10^{-5} 성능을 보였다. 그 후 1~2년 간에 걸쳐 그 성능이 입증되었고, 현재는 적용분야에 대한 연구가 집중되고 있으며, 시스템 엔지니어링 차원에서 설계 기술이 축적 및 고도화되고 있는 단계이다. Turbo 복호기의 구성은 연판정 입/출력(soft-in/soft-out)이 가능하고, 정보신호에 대하여 인터리버에 의해 분리된 2개의 구성 코드들이 병렬 연접된 구성을 하고 있다. 이러한 Turbo 복호기의 구성 코드로는 SOVA(Soft Output Viterbi Algorithm), MAP(Maximum A Posteriori), Sub-MAP 복호기 등이 있는데 채널의 잡음분산 평가가 필요하다는 단점이 있지만 일반적으로 성능이 우수한 MAP을 사용한다. 하지만 MAP 기반의 Turbo부호는 채널의 잡음 분산량의 평가가 필요하고 복호 시 행해지는 반복동작으로 인해 긴 복호 지연을 가지며, 메모리 요구량과 연산량이 많아 고속 데이타, 동영상 등의 고속 서비스의 적용에는 아직 많은 개선점을 가지고 있다.

3.1.1 Turbo 부호의 알고리즘 분석

구성코드가 RSC이고 부호율이 1/2인 4상태 Turbo 코드의 부호기 구조와 트렐리스도를 그림 3.1와 그림 3.2에 나타내었다.

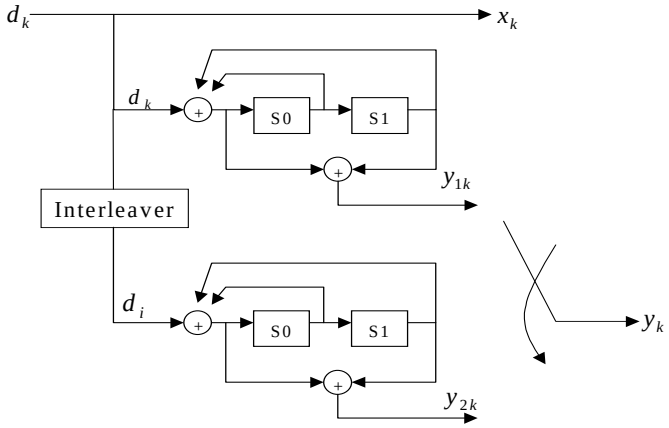


그림 3.1 Half-rate 병렬연접 RSC 부호기

Fig. 3.1 Half-rate Parallel concatenation of two RSC encoders.

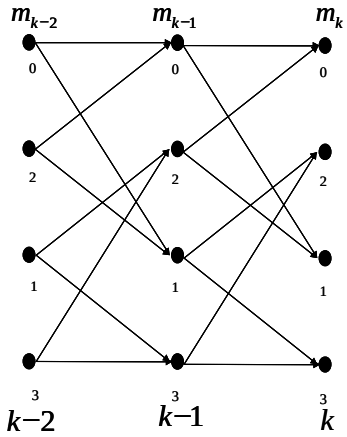


그림 3.2 4상태 트렐리스도

Fig. 3.2 Trellis Diagram.

Radix-2방식의 MAP 복호 알고리즘 구현을 위해 다음의 유도된 수식을 이용한다.

$$\begin{aligned}\alpha_k^i(m) &= P_r(d_k = i, S_k = m, R_1^k) \\ &= \sum_{j=0}^{1} \alpha_{k-1}^{b(j,m)} \delta_{k-1}^{j,b(j,m)}\end{aligned} \tag{3.1}$$

$$\begin{aligned}\beta_k^i(m) &= P_r(R_{k+1}^N \mid d_k = i, S_k = m, R_1^k) \\ &= \sum_{j=0}^{1} \beta_{k+1}^{f(j,m)} \delta_k^{j,m}\end{aligned} \tag{3.2}$$

α_k^i는 순방향 recursive function 이고, β_k^i는 역방향 recursive function이다. 그리고, $b(i,m)$은 격자도 상에서 입력이 i이고 현재 상태를 m으로 천이시킨 이전의 상태값을, $f(i,m)$은 상태 m에서 bit i가 입력되었을 때 다음에 천이될 상태값을 나타낸다. 식(3.1), 식(3.2)
을 이용하여 다음 수식 (3.3)을 얻을 수 있다.

$$L(d_k) = \frac{\sum_m \alpha_k^m \beta_{k+1}^{f(0,m)} \delta_k^{0,m}}{\sum_m \alpha_k^m \beta_{k+1}^{f(1,m)} \delta_k^{1,m}} \tag{3.3}$$

$\delta_k^{i,m}$은 branch metrics로서 다음과 같이 표현된다.

$$\begin{aligned}\delta_k^m &= P_r(d_k = i, S_k = m, R_k) \\ &= \exp(\frac{2}{\sigma^2} \times (X_k i + Y_k C_k^{i,m})\end{aligned} \tag{3.4}$$

여기서, $C_k^{i,m}$ 은 부호기의 상태가 m 이고 입력이 i 일 때 부호기 출력값이다.

위의 수식을 이용하여 Radix –2 기반의 Turbo 복호기 구조는 다음 그림 3.2와 같다.

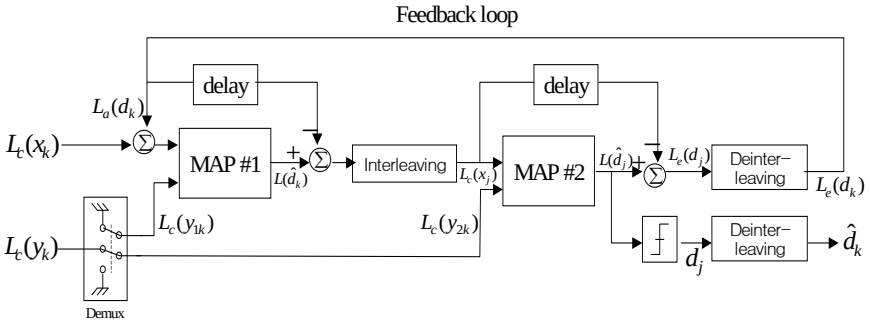


그림 3.2 Radix-2 기반의 터보 MAP 복호기 구조

Fig. 3.2 Block diagram of MAP turbo decoder based on Radix-2.

3.1.2 Turbo 부호 복호기 구현

앞 절에서의 복호 알고리즘을 이용하여 Radix-2기반의 복호기를 구현해 보았다. 구현 스팩은 다음 표3.1과 같다.

표 3.1 구현 스팩

Table 3.1 Implementing Specification.

방식	Radix-2
State	8
프레임 길이	212
Tail bit	3
인터리버 종류	Golden-interleaver
반복 회수	8

8비트로 양자화된 수신 비트는 먼저 길이 256 □ 8 의 fifo(first in first out)에 저장된다. Fifo에서 나온 출력은 MAP1의 입력으로 연결되며 MAP1에서는 BM과 FSM, BSM 연산 과정을 통해 최종적으로 LLR 을 출력한다. 이 LLR은 인터리버의 입력으로 연결되며, 이때 인터리버의 index는 ROM(256 □ 8)에 저장되어 있으며 이를 이용하여 데이터를 출력한다. 인터리버의 출력값은 MAP2의 입력으로 연결되며 MAP2에서의 MAP1과 동일한 과정을 거쳐 출력을 Deinterleaver로 입력하고, deinterleaver에서는 저장된 index의 역순을 이용하여 데이터를 출력, 이를 외부 정보로써 MAP1의 입력으로 첨가 된다. MAP1,MAP2,interleaver,deinterleaver,fifo의 동작은 FSM(Finite State

Machine)으로 동작하는 controller에 의해 제어 된다. 다음 그림 3.3은 이를 바탕으로 설계된 Turbo 복호기의 하드웨어 구조이다.

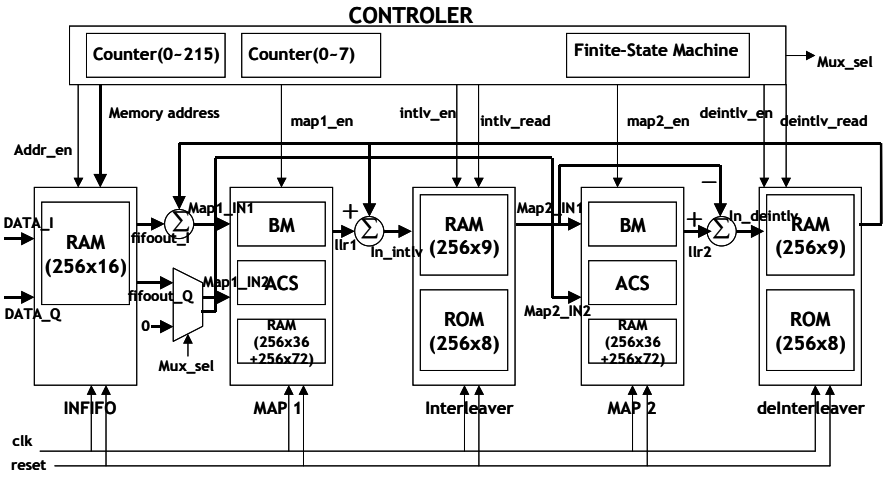


그림 3.3 Turbo 복호기의 하드웨어 구조

Fig. 3.3 Hardware schematic of Turbo decoder.

다음 그림 3.4 는 그림 3.3의 회로를 바탕으로 VHDL을 이용하여 Model-SIM에서 타이밍 시뮬레이션 한 wave 파형이다. 메인 클럭의 주기는 60ns 이고, 8회의 iteration 이후 428 μs 에서 복호 비트가 출력되었다.

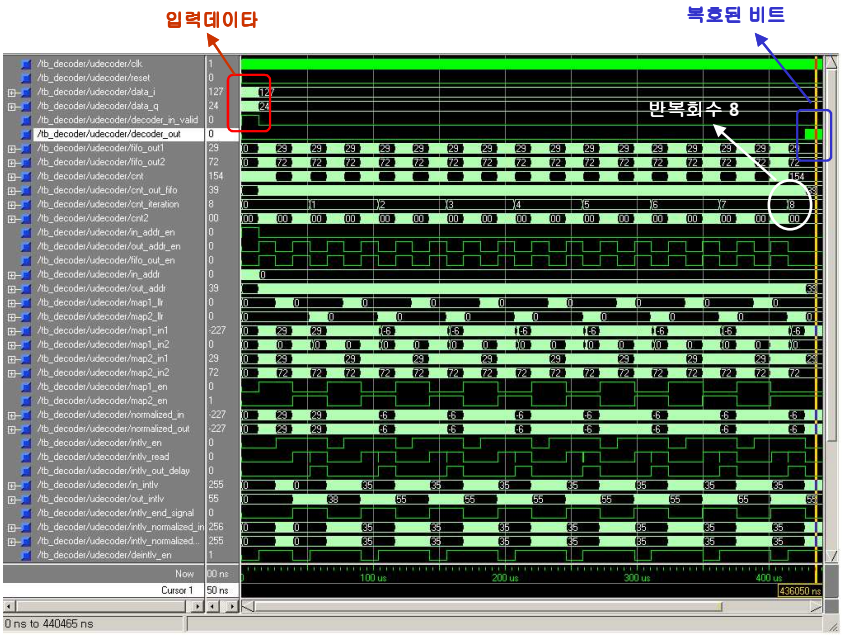


그림 3.4 타이밍 시뮬레이션 결과

Fig. 3.4 Timing Simulation results.

3.2 고속화 방안

터보 복호기의 전체 소요시간은 다음 수식(3.5)에 의해 계산 할수 있다.

$$\tau = 2\times(\tau_{map} + \tau_{\text{int}}) \tag{3.5}$$

여기서 τ_{map} 은 MAP1에서 복호를 위해 소모된 시간을 말하며, τ_{int} 는 인터리버에서 걸리는 시간을 말한다. 이 때 τ_{map} 는 다음의 수식으로

정의 할 수 있다.

$$\tau_{map} = 2 \times l_{flame} \times IT \times T_{clk} \quad (3.5)$$

여기서, l_{flame} 는 codeword의 길이에 tail bit를 더한 값이고, IT는 반복 회수를 의미하며, T_{clk} 는 클럭 주기이며 이 수식에서 2가 곱해지는 것은 BSM과 FSM을 각각 계산하기 때문이다.

수식 (3.5)에서 2가 곱해지는 이유는 기존의 방식일 경우 두개의 MAP은 서로 직렬로 연결되어 있기 때문이다. 따라서 고속의 Turbo 복호기 구현을 위해서는 다음과 같은 지연 요인을 극복할 필요가 있으며, 표 3.2에 고속화 방안을 나타내었다. 이는 향후 연구할 과제로 남아 있다.

첫째, 2개의 MAP에서의 직렬 연결

둘째, 반복회수의 수

셋째, 프레임 길이

넷째, MAP에서의 복호 방식(BSM,FSM)

표 3.2 고속화 방안

Table 3.2 The method of high speed decoder.

①연산 방식	기존의 덧셈 및 곱셈 방식(Full Adder)인한 연산 속도 저하	oCarry Save Adder oCarry Select Adder oCarry Lookahead Adder oOverlap Bit Scanning Multiplier 의 적용으로 인한 연산속도 향상
②복호지연 관점	터보코드는 수신단에서 N 개의 수신비트를 저장한 후 복호를 시작하므로 항상 N 비트 만큼의 복호 지연을 가진다	**Center-to-Top 방식**으로 복호지연시간을 N/2 만큼 단축한다
③반복중지 알고리즘	반복횟수가 고정되어 있다	**반복 중지 알고리즘**으로 어느 반복이 어느 정도 지나면 더 이상 개선 여지가 없으면 반복을 중지
④복호속도 관점	한 클럭에 1 비트를 복호하는 구조를 가진다	**Radix-4 방식**으로 한 클럭에 2 비트의 정보를 복호하므로 복호 속도를 2 배 향상시킨다
⑤병렬처리	두개의 복호기를 직렬 모드로 구성	**병렬 구조를 이용하여 두개의 복호기를 동시에 사용함으로 지연 감소**

제 4 장 결 론

본 논문에서는 세가지 반복 부호에 대한 기존의 복호기를 살펴 보고 기존의 방식에서 문제시 되는 많은 연산량 및 시간 지연을 극복할 수 있는 방안을 제안하였다.

DVB-S2 표준안에서 제시된 LDPC의 경우 큰 codeword 사이즈와 많은 반복회수로 인해 발생하는 전력 소모와 시간지연을 극복할 3가지 방식을 제안한다. 성능의 열화 없이 N의 길이를 극복하는 방안으로 Sequential Decoding방식과 Early Detected 방식을 제안 하였으며, 반복회수를 줄이기 위해 두가지의 Early-stop방식을 살펴 보았다.

기존의 TPC복호기는 row 복호 후 column 복호를 하기 때문에 지연으로 인한 복호 속도 저하가 생겼으나, 본 논문에서는 고속 데이터 통신 전송 및 동영상 등을 포함한 무선 멀티미디어 전송을 실현 하기 위해서 병렬 TPC 복호기 및 Early-stop 방식을 제안하였다. 고속화 방안들을 적용하여 VHDL 코드를 Altera사의 Design complier를 이용하여 시뮬레이션 하였다. 고속 복호기는 timing 시뮬레이션을 위해 APEX20KE(EP20K300EQC240-2)칩을 사용하였으며 기존의 직렬 복호기일 때와 본 논문에서 제안하는 병렬 고속 복호기를 비교 하였다. 그 결과 decoding speed 는 기준으로 기존의 decoder보다 약 2배의 속도 향상을 가져 오고 요구되는 메모리(Logic cell)은 약 1.5배 정도 더 필요함을 알 수 있었다.

마지막으로 트렐리스를 기반으로 하는 Turbo 부호를 설계하였으며,

설계 결과 Turbo 부호에서 복호 지연을 수식으로 정리하여 이를 바탕으로 고속 복호기 구현을 위해 개선되어야 할 4가지 요인을 제시하였다. 제시된 요인을 극복할 알고리즘을 살펴보고 실제 구현함에 있어서 속도와 H/W적인 복잡도 면에서 최적의 알고리즘을 연구하는 것이 향후 과제이다.

참고 문헌

[1] C. Berrou, A. Glavieux, and P.Thitimajshima, “Near Shannon Limit Error-Correcting Code and Decoding : Turbo Codes”, in *Proc. Of ICC’93*, 1993.

[2] “Satellite Broadcasting System of Integrated Service Digital Broadcasting”, ITU-R BO.1227-2

[3] “Digital Video Broadcasting(DVB): Framing Structure, Channel Coding and Modulation for Digital Satellite News Gathering(DSNG) and Other Contribution Applications by Satellite”, *ETSI EN 301 210 :European Standard*

[4] Carden Frank, "A Quantized Euclideans soft Decision Maximum Likelihood Sequence Decoder: A Concept for Spectrally Efficient TM Systems", *Proceedings of the International Telemetering Conference, Vol. XXIV, pp. 375-384, OCT., 1988.*

[5] R.G.Gallager, ”Low-Density Parity-Check Codes”, *IRE Trans. Information Theory*, vol.8, pp.21-28,1962

[6] D.J.C. Mackay and R. M. Neal, “Near Shannon Limit Performance of Low-Density Parity-Check Codes,” *Electron. Letter*, Vol.32, pp.1645-1646, Aug.1996

[7] C.Berrou, A.Glavieus, and P.Thitimajshima, “Near Shanon Limit Error-Correcting Coding and Decoding: Turbo-Codes,” *in Proc. ICC93*, pp.1064-1070, May 1993.

[8] P.Robertson, E.Villebrun, and P.Hoeher, “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain,” ICC95, pp.1009-1013. 1995.

[9] S.S. Pietrobon, "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder," *in Proc. IEEE Int. Symposium on Information Theory*, pp. 471-480, 1995.

[10] S.S.Pietrobon, "Implementation and Performance of a Turbo/MAP Decoder," International Journal of Satellite Communications, vol.16, pp.23-46, 1998.

[11] D.J.C. Mackay and R.M.Neal, “Near Shannon Limit Performance of Low-Density Parity-check Codes,” *Electron. Letter,* Vol.32 ,pp 1710-1722, Aug. 1996.

[12] R.M. Pyndiah, “Near-optimum decoding of product codes: Block turbo codes,” *IEEE Trans. Commun*., vol. 46, pp 1003-1010, Aug. 1998.

[13] D.Chase, “A class of algorithms for decoding block codes with channel measurement information,” *IEEE Trans.Inform. Theory*, vol. IT-18, pp.170-182, Jan. 1972.

[14] O.Aitsad and R.Pyndiah, “Performance of Reed-Solomon block turbo codes,” in *Proc. IEEE GLOBECOM’96 Conf*., vol.1/3, London. UK, Nov., pp.121-125, 1996.

[15] 이태길, 정지원, “병렬 구조를 이용한 Turbo Product Code의 성능 분석”, 한국통신학회 하계종합 학술대회, vol 27, pp37, 2003. 7.

감사의 글

사람의 일생은 흔히 사계(四季)로 비유되기도 합니다. 나는 지금 내 생의 어디쯤 와 있는가... 이렇게 되돌아보며 지난 2년간의 석사과정을 정리하려 합니다. 돌이켜 생각해 보면 고마운 사람들이 너무나도 많습니다...

먼저, 논문이 완성될 때까지 미흡한 저를 이끌어 주신 제 인생의 스승 정지원 교수님께 깊은 감사를 드립니다. 석사과정동안 교수님의 사랑과 관심이 없었다면 지금의 저는 결코 없었을 것입니다. 그리고 논문의 미비점을 보완하여 보다 충실한 내용이 될 수 있도록 맡아주신 조형래 교수님, 김기만 교수님께 감사드립니다. 또한 학부와 석사과정동안 무수한 가르침을 주시고 조언을 아끼지 않으셨던 김동일 교수님, 강인호 교수님, 민경식 교수님 그리고 석사과정때 들어오신 윤영 교수님 감사드립니다.

학부때 연구실에 들어와서 교수님 버금가게 가르쳐주시고 조언을 아끼지 않았던 태길이형과 성준이형에게도 감사의 마음을 전합니다. 또한 같이 동고동락하면서 논문에 많은 도움을 준 덕군이와 진희, 그리고 학부생 민혁이에게 고마움을 전하며, 대학원 생활에 웃음을 준 동기 동환이, 경식, 세영이 그리고 충렬이형, 영배형에게도 감사의 마음을 전합니다. 대학원 생활에 많은 조언을 준 동식이형, 종호형, 수홍이형, 그리고 후배 형준이에게도 감사의 마음을 전합니다. 또한 언제나 웃음을 잃지 않도록 힘을 되어준 동아리 HAM의 선후배, 동기들에게도 감사의 마음을 전합니다.

마지막으로, 한없는 신뢰와 사랑으로 이 자리까지 밀어주신 사랑하는 어머니와 누나, 그리고 하늘에 계신 아버지에게 이 논문을 바칩니다.