工學碩士 學位論文

병렬 Turbo Product 부호 복호기의 알고리즘 분석 및 VHDL설계

VHDL Design and Analysis of Decoder of Parallel Turbo Product Code

指導敎授 鄭 智 元

2004年 2月

韓國海洋大學校 大學院

電 波 工 學 科

李 泰 佶

本 論文을 李泰佶의 工學碩士 學位論文으로 認准함.

委員長 : 工學博士 趙 炯 來 (印)

委 員 : 工學博士 金 基 萬 (印)

委 員 : 工學博士 鄭 智 元 (印)

2003年 12月

韓國海洋大學校 大學院

電 波 工 學 科

李 泰 佶

차 례

Abstract

Recently, the trend of wireless communication is changed from the conventional narrow-band voice service to the wide-band multimedia service. In 1993, Berrou et *al* introduced a new class of error correcting codes for digital transmission : the "turbo code". Therefore, the important problems of high-speed applications of turbo decoder are decoding delay and computational complexity. Therefore, the real difficulty in the field of channel coding is essentially a problem of decoding complexity of powerful codes.

Another interest area of channel coding scheme is LDPC(Low Density Parity Check) code. But 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 TPC(Turbo Product Code) which has low latency and simple structures compare with turbo code and LDPC. 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, a parallel decoder structure to reduce the latency is proposed in this thesis. Proposed TPC decoder needs only one delay element in contrast to conventional algorithm, which needs two delay elements.

This thesis analyzes the parallel TPC decoder by mathematical theory and compares the performance between parallel algorithm and conventional algorithm by computer simulation. Also this thesis establishes parameters by fixed-point simulation for VHDL implementation. From results of computer simulation and VHDL implementation, this thesis shows that decoding time of parallel algorithm is halved with this structure while maintaining the same performance level.

Nomenclature

$\Lambda(y_i)$: 수신된 신호 y에 대한 log-likelihood ratio

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

$\beta(m)$: 복호된 신호의 신뢰도

w_j : extrinsic 정보

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

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

C^i : 가능한 모든 부호어의 집합

$c_l^{\min(+1)}$: j 번째 비트가 +1을 가지고 부호어 중 수신신호와의 해밍거리가 가장 작은 부호어의 l번째 비트

$c_l^{\min(-1)}$: j 번째 비트가 -1을 가지고 부호어 중 수신신호와의 해밍거리가 가장 작은 부호어의 l번째 비트

r_j : 복호기의 soft input 값

r_j' : chase 알고리즘에 의해 복호된 최적결정비트의 soft decision된 값

S_j^{+1} : $c_j^i = +1$을 가지는 codeword의 인덱스 $\{C^i\}$의 집합

S_j^{-1} : $c_j^i = -1$을 가지는 codeword의 인덱스 $\{C^i\}$의 집합

제 1 장 서론

연접부호는 높은 부호이득을 얻어낼 수 있으며, 무선통신시스템에서 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[2]. 그러나 LDPC 는 복호화가 간단한 반면에 부호화 부분의 높은 복잡도가 LDPC 부호의 최대의 단점이다. 1998 년 Pyndiah 에 의해 소개된 TPC[3]는 기존의 LDPC 부호의 단점인 부호화 시 구성 어려움, 그리고 성능 향상을 위한 많은 블록 크기를 요구한다는 것과 Turbo 부호의 많은 계산량과 고속 복호기 구성의 어려움 등의 단점을 보완한 작은 블록 크기를 가로 세로로 product 시킨 후 같은 복잡도로써 많은 블럭 크기의 효과를

얻을 수 있고 복호기가 간단하여 고속 구현이 가능하며, 높은 부호화율에서 Shannon's Limit 에 근접하는 새로운 차세대 오류정정 부호화 방식으로 무선 멀티미디어 통신을 요구하는 최근의 무선 통신시스템에 오류정정방식으로 적합하다. 그러나 TPC 복호기의 가장 큰 문제점은 두 개 복호기가 직렬로 연결된 구조에서는 복호기 두 개를 병렬로 처리할 수 없다는 것이 가장 큰 단점이다. 성능 향상을 위한 복호 반복을 위해서는 첫 번째 부호기에 해당하는 복호기를 이용하여 복호하고 난 뒤에 첫 번째 복호기에 나온 연판정 값을 이용하여 두 번째 복호기에서 데이터를 복호하고 이의 과정을 여러번 반복하면서 성능을 향상시킨다. 따라서 두 개의 복호기가 직렬로 연결되기 때문에 지연으로 인한 속도 저하를 가져올 수 있기 때문에 TPC 의 병렬화 구조제안은 필수적이다. 따라서 본 논문에서는 고속 복호가 가능한 병렬 TPC 구조를 제안하고 성능분석을 하였으며 구현을 위한 fixed point 시뮬레이션을 실행하여 최적 파라미터를 도출하고 아울러 VHDL 설계를 하였다.

제 2 장 Turbo Product Code의 구조분석

2.1 반복 복호와 외부 정보

1993년 Turbo 부호[1]가 소개되어진 이후에 블록부호에 대한 연판정 입출력(SISO : soft input soft output)에 대한 연구가 활발히 진행되고 있다. 채널의 'Soft values'로 의미 되어진 것을 더 명확히 정의 할 수 있고, soft value L(u)를 가지는 binary value u를 부호화 한다면 soft values L(x)와 코드화 된 비트 x를 만들 수 있다. 선형부호에서 쌍 대칭적인 채널(BSC) 또는 가우시안/페이딩 채널을 전송 후에 수신된 필터 출력 y에서 x의 log-likelihood 비율 L(x|y)는 식 (2.1)과 같다.

$$
\begin{aligned}
L(x|y) &= \log\frac{P(x=+1|y)}{P(x=-1|y)} = \log\frac{P(y|x=+1)P(x=+1)}{P(y|x=-1)P(x=-1)} \\
&= \log\frac{\exp\left(-\frac{E_S}{N_0}(y-a)^2\right)}{\exp\left(-\frac{E_S}{N_0}(y+a)^2\right)} + \log\frac{P(x=+1)}{P(x=-1)} \\
&= L_c y + L(x) \qquad\qquad L_c = 4E_S/N_o = 2/\sigma^2
\end{aligned}
\tag{2.1}
$$

2차원 부호에서 반복 복호의 원리는 그림 2.1과 같다. K1, K2 정보비트 u는 그림 2.1에서 보여진 것처럼, 직사각형의 매트릭스 이며 그것은 두 조직화 코드 C^{-}와 $C^{|}$의 패리트 비트 P^{-}와 $P^{|}$이다. 부합된 필터의 수신된 값은 y이며, $L_c \cdot y$ 는 모든 코드 비트들의 디코더로 이용하는 채널 매트릭스 값이다.

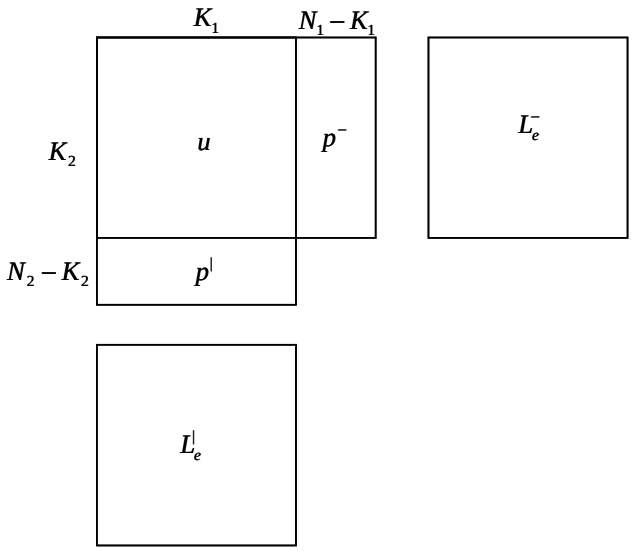


그림 2.1 이차원 부호에 관한 반복 복호

Fig. 2.1 Iterative decoding for two-dimensional codes.

간단한 예를 들어 복조하는 과정을 들어보면 그림 2.2 와 같다.

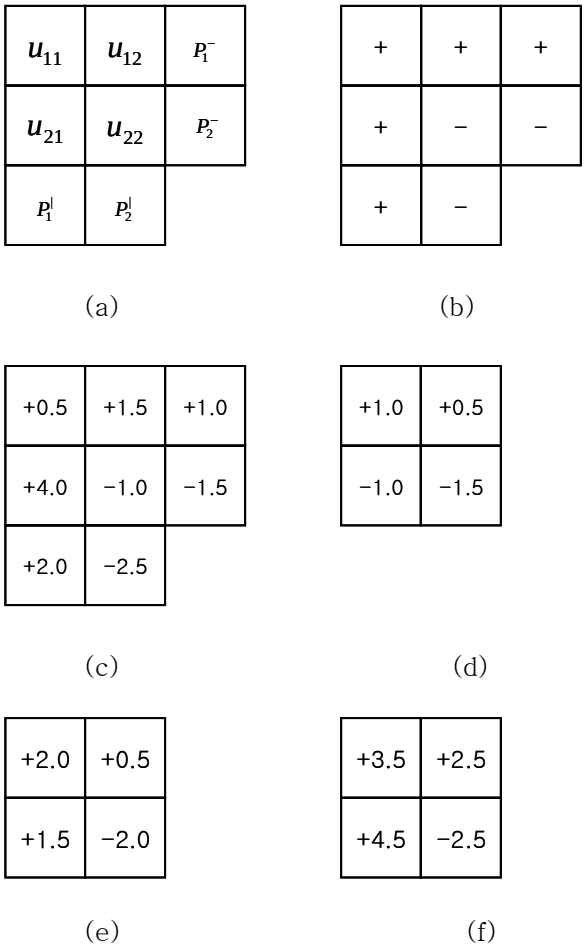


그림 2.2 반복 복호 예: a) 신호의 배치; b) 신호의 부호화; c)수신된 값 $L_c \cdot y$; d) 가로 복호 후의 Extrinsic 정보; e) 세로 복호 후의 Extrinsic 정보; f) 가로와 세로 복호 후 soft output.

Fig. 2.2 Example of Iteration decoding: a) Signal arrangement; b) Coded values; c) Received values $L_c \cdot y$; d) Extrinsic information L_e^- after first horizontal decoding; e) Extrinsic information $L_e^|$ after first horizontal decoding; f) soft out after the first horizontal and vertical decoding.

Component code로써 (3,2,2) single parity check code의 에서 4가지 정보 비트들을 그림 2.2(b)에서 보여진 것처럼 GF(2)에 (+1,-1)요소로 하여 두개의 (3,2,2) single parity check code 로 수식화 한다. 그림 2.2(c)에서 보여진 것처럼, 수신된 값이 $L_c \cdot y$ 를 가진다고 가정하고 우선 수평적 디코딩부터 시작하면 Bit u_{11} 의 정보는 직접적인 u_{11} 과 간접적인 $u_{12} \oplus P^-$ 두개로 나타낼 수 있다. u_{11} 과 P^- 는 정보 L-value를 위해 가지고 있는 통계상 독립적인 것으로 전송된다.

$$L(u_{12} \oplus p_1^-) = L(u_{12}) + L(p_1^-) = 1.5 + 1.0 \approx 1.0$$

그림 2.2(d)에서 볼 수 있는 u_{11} 에 대한 이 같은 간접적 정보를 extrinsic value라 부른다. u_{12} 로 $0.5+1.0 \approx 0.5$ 의 같은 수평적 extrinsic value를 얻는다. 수평의 extrinsic table이 채워질 때, 수직의 디코딩을 위한 priori value로써 L_e^- 를 사용하는 수직 디코딩을 시작한다. u_{11} 에대해 세 가지의 값을 가진다.

① 직접적으로 수신된 값 0.5 .

② 수평적 디코딩 +0.1로부터의 a priori value L_e^-.

③ 모두 사용 가능한 수직적인 extrinsic value

$L_e^- = u_{21} \oplus P_1^|$ $((4.0+(-1.0))+2.0 \approx 2.0)$. 수직의 extrinsic value은 그림 2.2(e)의 테이블 안에 저장된다.

$$u_{21} = (0.5+1.0)+2.0 \approx 1.5$$
$$u_{12} = (1.0+(-1.5)+(-2.5) \approx 0.5$$
$$u_{22} = (1.5+0.5)+(-2.5) \approx -2.0$$

만약 반복되는 것을 여기서 멈췄다면 수직 반복 후에 다음과 같은 soft output을 얻을 수 있다. (그림 2.2(f))

$$\mathrm{L}(\hat{\mathrm{u}}) = \mathrm{L_c} \cdot \mathrm{y} + \mathrm{L_e^-} + \mathrm{L_e^|} \qquad (2.2)$$

위에서 언급한 부호 블록도는 아래 그림 2.3과 같다.

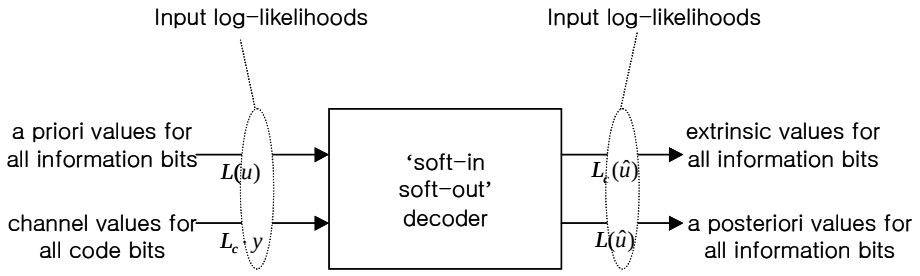


그림 2.3 'soft-in/soft-out' 복호기 블록도

Fig. 2.3 Block diagram of'soft-in/soft-out' decoder.

수평축과 수직축의 방향에서 k_1, k_2 의 정보비트를 부호화함으로써 조직블록코드의 결합을 사용할 수 있다. 예를 들어,

Horizontally : K_2 코드워드의 (K_1, N_1) 블록코드

C^- with rate $R_1 = K_1 / N_1$

Vertically : K_1 코드워드의 (K_2, N_2) 블록코드

$C^|$ with rate $R_2 = K_2 / N_2$

이며 2차원 코드의 총 부호화율은 아래식과 같다.

$$\mathrm{R} = \mathrm{R}_1 \times \mathrm{R}_2$$

각 열과 행의 정보 매트릭스는 정보열 u가 코드 워드 안에 부호화됨으로써 형성된다.

$$x = (x_1, x_2, \ldots\ldots, x) = (u_1 \ldots\ldots u_k, p_1 \ldots\ldots, p_{n-k})$$

$$\mathrm{x} \in \mathrm{C}^-, \mathrm{x} \in \mathrm{C}^| \qquad (2.3)$$

그림 2.3과 같이 'soft-in/soft-out' 부호기에서 Maximum a posteriori(map) decoder의 "symbol-by-symbol"의 출력은 정보열에서 보내는 +1,-1에 대한 posteriori log-likelihood ratio로

정의하며 식은 아래와 같다.

$$L(\hat{u}) = L(u \mid y) = \log \frac{P(u=+1 \mid y)}{p(u=-1 \mid y)} \tag{2.4}$$

이 같은 디코더는 모든 부호화된 비트 u에 대한 priori values, L(u)를 사용하며 정보 비트에 대한 연판정출력 $\mathrm{L}(\hat{\mathrm{u}})$ 와 부호열에서 다른 부호비트의 연판정 출력을 갖고있는 extrinsic 정보 $\mathrm{L_e(u)}$ 를 생성한다. 조직화된 코드에서 정보비트u에 대한 soft output은 세가지 정보의 합으로 다음식과 같이 나타낸다.

$$\mathrm{L}(\hat{\mathrm{u}}) = \mathrm{L_c} \cdot \mathrm{y} + \mathrm{L(u)} + \mathrm{L_e}(\hat{\mathrm{u}}) \tag{2.5}$$

처음 반복에서는 priori information $\mathrm{L(u)} = 0$ 이 된다. 수평적 코드 C^- 의 디코딩은 정보 부분과 수평적 패리티 부분에 대한 채널정보 $\mathrm{L_c} \cdot \mathrm{y}$ 를 사용함으로써 시작하며, 정보비트 u에 대한 수평적 코드 C^-의 extrinsic information의 $L_e(\hat{u})$는 아래 식과 같다.

$$\mathrm{L_e^-}(\hat{\mathrm{u}}) = \mathrm{L^-}(\hat{\mathrm{u}}) - \mathrm{L_c} \cdot \mathrm{y} \tag{2.6}$$

수평적 코드에서 추정한 $L_e^{\downarrow}(\hat{u})$는 수직적 디코딩 $C^{\downarrow}$에 대해 priori value 로서 사용된다.

$$L_e^{\downarrow}(\hat{u}) = L^{\downarrow}(\hat{u}) - (L_c \cdot y + L_e^{-}(\hat{u})) \tag{2.7}$$

이 수직적 extrinsic information은 다음 반복단계에서 code C^{-}의 디코딩의 새로운 priori value로서 사용될 것이다. 수직적이고 수평적 부호시 최초의 반복은 L-values들은 통계적으로 독립적이며, 그것들은 같은 정보를 간접적으로 사용할 것이며, 반복을 할수록 성능이 향상될 것이다. 물론, 마지막 수직적 반복 후에 비트를 결정하기 위해서는 마지막 두개의 extrinsic 정보를 결합한다.

$$L(\hat{u}) = L_c \cdot y + L_e^{-}(\hat{u}) + L_e^{\downarrow}(\hat{u}) \tag{2.8}$$

2.2 Turbo Product Code의 부호화기 구성

TPC 부호는 두개 혹은 그 이상의 짧은 길이의 블록 부호를

(C_1, C_2)를 이용하여 product 부호 $(P=C_1\times C_2)$를 만드는 것이 가장 효율 적이다. 그림 2.4은 product code의 구성도를 나타낸다.

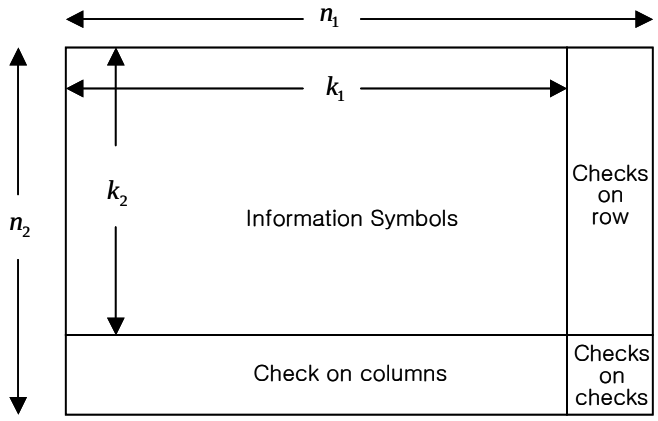


그림 2.4 TPC 부호화기 구성도 $(P=C_1\times C_2)$

Fig. 2.4 Encoder construction of turbo product code $(P=C_1\times C_2)$.

그림 2.4와 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_1으로 부호화 시키고, 세로는 (n_2, k_2, δ_2) 를 가지는 C_2 로 부호화시켜 전송한다. 따라서 TPC부호 $P=C_1\times C_2$ 이므로, (n,k,δ) 를 가진다. 여기서 $n=n_1\times n_2$, $k=k_1\times k_2$, $\delta=\delta_1\times\delta_2$ 이고 부호화율은 $R=R_1\times R_2$ $(R_1=k_1/n_1,$

$R_2 = k_2 / n_2$) 이다. 따라서 두 부호어를 product 함으로써, 높은 부호화 율에서 최소 minimum Hamming distance의 증가에 의해서 오류 정정 능력은 향상된다. 그리고 오류를 산발시키는 효과가 있는 인터리버가 필요치 않으며, 복호시 가로 부분을 먼저 복호 한 후 이를 extrinsic 정보로 이용하여 세로 부분을 복호하면서 반복 복호를 한다. TPC에 적용되는 블록 부호 C_1, C_2 는 해밍부호, BCH부호, RS부호 등 다양한 블록 부호를 적용시킬 수 있다.

2.3 Turbo Product Code 복호기 구성

Pyndiah가 제안한 TPC 복호기 구조는 아래 그림 2.5와 같다.

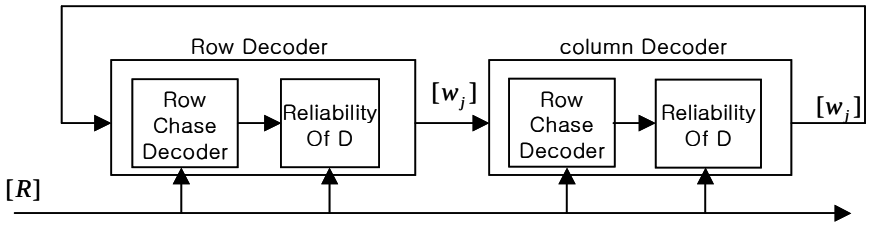


그림 2.5 TPC 복호기 구조

Fig. 2.5 Structure of TPC decoder.

원 신호 E 그리고 가우시안 잡음 신호 N 에 의해 수신신호 벡터 R 다음식과 같이 나타낼 수 있다.

$$R = E + N \tag{2.9}$$

여기에서 $R=(r_1,..,r_l,..,r_n)$, $E=(e_1,..,e_l,..,e_n)$, $N=(n_1,..,n_l,..,n_n)$ 이다. 최적 결정 비트 $D=(d_1,...,d_l,...,d_n)$ 는 식 (2.10)과 같이 maximum likelihood 방식에 의해 결정 된다.

$$\begin{aligned} &D=C^i \ \ if \ \ \Pr\{E=C^i|R\} > \Pr\{E=C^j|R\} \ \forall\, j\neq i \\ &D=C^i \ \ if \ \ |R-C^i|^2 > |R-C^j|^2 \ \ \forall \ \ j\neq i \\ &|R-C^i|^2=\sum_{l-1}^{n}(r_i-c_i^l)^2 \end{aligned} \tag{2.10}$$

$C^i=(c_1^i,c_2^i,...,c_l^i,...,c_n^i)$ 는 가능한 모든 부호어의 집합 C 의 i 번째 부호어 이다. 이 경우 n 값이 크면 계산량이 매우 많고 오래 걸리며 거의 불가능하다.

2.3.1 chase-Ⅱ 알고리즘

1972년 low complexity 알고리즘을 제안되었다[5]. 이는 높은 SNR에서 D는 Y 의 중심점에서 반경이 $\delta-1$의 구안에 포함 될 확률이

높다는 것을 이용한 알고리즘 이다. 여기서 $Y=(y_1,...,y_l,...,y_n)$, $y_i=0{,}5(1+\mathrm{sgn}(r_i))$ 이다. 후보 가능한 부호어 C 를 찾는 chase-Ⅱ 알고리즘은 다음과 같다.

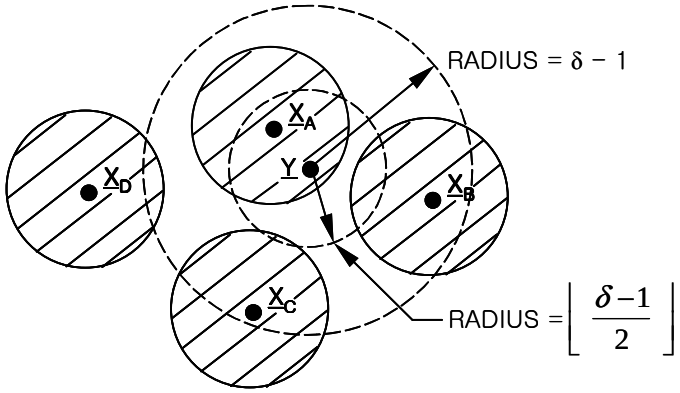


그림 2.6 수신 신호의 분포

Fig. 2.6 Geometric sketch for receive signal.

1단계: $p=\lfloor \delta/2 \rfloor$ 개의 신뢰성 없는 Y 의 비트 위치를 수신 백터 R 을 이용해서 결정한다. 신뢰성 없는 비트의 위치는 수신되는

$$\Lambda(y_j)=\ln\left(\frac{\Pr(e_j=+1/r_i)}{\Pr(e_i=-1/r_j)}\right)=\left(\frac{2}{\sigma^2}\right)r_j=|r_j|$$ 이다.

2단계: q개의 test pattern T^q 를 생성한다 ($q=2^p$) . T^q 의 생성

방법은 n개의 비트 위치 중 $\Lambda(y_i)$가 가장 적은 값에 해당하는 위치 j에 "1"을 위치시키고 나머지 비트위치는 "0"을 삽입하고, $\Lambda(y_i)$가 가장 적은 두개의 비트 위치에 "1"을 위치시키고 나머지 비트는 "0"으로 배치한다. 같은 방법을 계속해서 $\Lambda(y_i)$가 가장 작은 p개의 비트 위치에 "1"을 위치시키고 나머지 비트는 "0"을 삽입한다. 그리고 항상 all-zero pattern을 항상 포함시켜야 한다.

3단계 : q개의 T^q를 생성하고 난 뒤에, $Z^q=Y\oplus T^q$하여 오류 위치를 정정한 Z^q를 생성한다.

4단계: Z^q를 블록 복호하여 C^q를 생성한다.

각 column과 row에 Chase 알고리즘을 적용 시켜 에러 검출 가능한 범위 안의 product 부호에 가장 확률이 높은 부호어를 찾아 에러를 정정하는 알고리즘이다. 따라서 Chase 알고리즘은 최적의 수신신호를 찾는 것은 아니고 부최적인 알고리즘 이다. Chase 알고리즘은 그림 2.7과 같다.

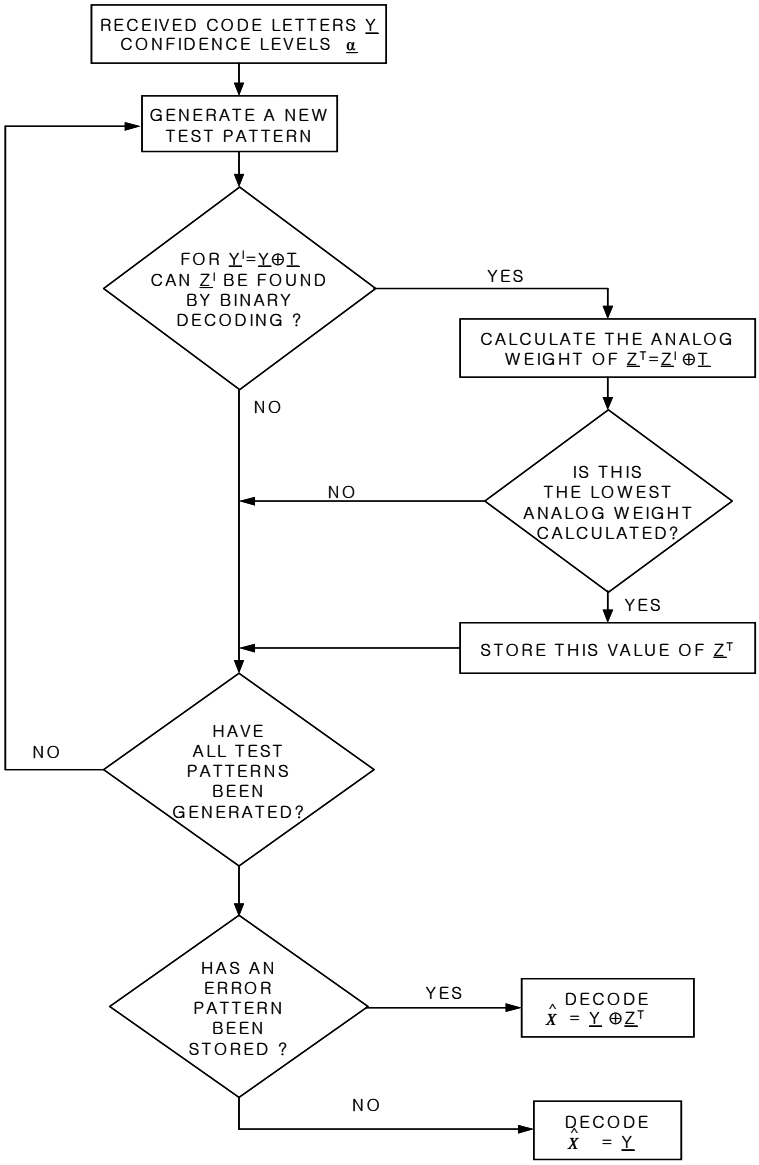


그림 2.7 chase 알고리즘의 플로차트

Fig. 2.7 Flow chart for chase algorithm.

2.3.2 결정비트의 신뢰도와 모의 실험결과

4단계가 완료되면 검토 되어지는 $C^q=(C^1,C^2,C^3,...,C^q)$ 벡터가 생성되며, 그림 2.5의 부호기에서 연판정을 출력하기 위해 D 의 j 번째 비트 신뢰도 $(LLR)_j$ 는 아래 식 (2.11)와 같다.

$$(LLR)_j = \ln\left(\frac{\Pr(e_j=+1/R)}{\Pr(e_j=-1/R)}\right) \qquad (2.11)$$

여기서 $\Pr(e_j=+1/R) = \sum_{i\in S_j^{+1}} \Pr(E=C^i/R)$ 이며, S_j^{+1} 는 $c^i_j=+1$을 가지는 codeword의 인덱스 $\{C^i\}$의 집합이다.

$\Pr(e_j=-1/R) = \sum_{i\in S_j^{-1}} \Pr(E=C^i/R)$ 이며 S_j^{-1} 는 $c^i_j=-1$ 을 가지는 codeword의 인덱스 $\{C^i\}$ 의 집합이다. Bayes 이론을 적용시키고 서로 다른 부호어들이 균등하게 분포되어 있다고 가정하면 d_j 에 관한 LLR은 다음 식과 같다.

$$(LLR)_j = \ln\left(\frac{\sum_{i\in S_j^{+1}} p\{R/E=C^i\}P\left(E=C^i\right)}{\sum_{i\in S_j^{-1}} p\{R/E=C^i\}P\left(E=C^i\right)}\right) = \ln\left(\frac{\sum_{i\in S_j^{+1}} p\{R/E=C^i\}}{\sum_{i\in S_j^{-1}} p\{R/E=C^i\}}\right) \tag{2.12}$$

$$p\{R/E=C^i\} = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^n \exp\left(-\frac{\left|R-C^i\right|^2}{2\sigma^2}\right) \tag{2.13}$$

(2.12)와 (2.13)를 이용하여 (2.14)을 유도할 수 있다.

$$(LLR)_j = \ln\left(\frac{\sum_{i\in S_j^{+1}} \exp\left(-\frac{|R-C^j|^2}{2\delta^2}\right)}{\sum_{i\in S_j^{-1}} \exp\left(-\frac{|R-C^j|^2}{2\delta^2}\right)}\right) \tag{2.14}$$

식 (2.14)은 높은 부호율을 가지는 부호어에 대해서는 거의 계산 불가능하고 복잡하므로 Pyndiah 에 의해 식 (2.15)와 같이 간략하게 최적화 하였다.

$$(LLR)_j \approx \frac{2}{\delta^2}\left(r_j + \sum_{l=1, l\neq j}^{n} r_l\, c_l^{\min(+1)} P_l\right) \tag{2.15}$$

$$P_l = \begin{cases} 0 & if \quad c_l^{\min(+1)} = c_l^{\min(-1)} \\ 1 & if \quad c_l^{\min(+1)} = c_l^{\min(-1)} \end{cases}$$

$c_l^{\min(+1)}$ 과 $c_l^{\min(-1)}$ 은 j 번째 비트가 +1, -1을 가지고 부호어 중 수신신호와 해밍거리가 가장 작은 부호어의 l 번째 비트를 나타낸다. 따라서 식(2.15)는 식(2.16)와 같이 나타낼 수 있다.

$$r_j' = r_j + w_j$$

$$w_j = \sum_{l=1, l \neq j}^{n} r_l \, c_l^{\min(+1)} P_l \qquad (2.16)$$

r_j : soft input data , w_j : extrinsic information

이는 R 과 r_j ($l \neq j$) 의 최소 유클리언 거리의 함수 이다. 만일 $\mathbf{C}^{\min(+1)} = \mathbf{C}^{\min(-1)}$ 의 경우도 발생하는데 이 경우에는 least reliable bit와 test sequence의 수를 증가 시킨다. 식 (2.16)에서 반복 시 생성되는 r_j 의 soft decision 값 r_j' 는 입력 수신벡터의 r_j 와 extrinsic 정보 의 합으로서 표시될 수 있다. Extrinsic 정보 w_j 는

자기 신호 j 번째를 제외한 r_l 와 선택된 부호어 C_l 의 곱의 합으로 표시 될 수 있다. 즉 $r_j^{'}$ 는 chase 알고리즘에 의해 복호된 D 의 soft decision 값이다. $r_j^{'}$ 는 $r_j^{'}=\beta d_j$ $(\beta \geq 0)$ 와 같이 나타낼 수 있으며, β 는 신뢰도 factor 이다. 처음 반복 시에는 신뢰도가 낮으므로 낮은 값으로 반영하면서 반복횟수가 증가 할수록 높게 설정한다. Product code를 가로, 세로 복호 시 TPC의 복호 블록도는 아래 그림 2.8과 같다.

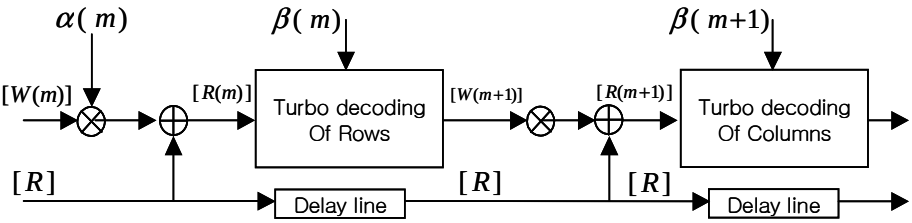


그림 2.8 Pyndiah가 제안한 TPC 복호기 구조

Fig. 2.8 TPC decoder structure that Pyndiah proposes.

$$R[2]=[R]+\alpha[2]\cdot[W[2]], \quad W[2]=R[2]-R[1]$$

α 는 스케일링(scaling) factor 이며, 이는 수신신호 R 과 W 에 있는

샘플들의 표준편차를 고려한 것이다. 따라서 대부분의 논문에서도

$$\alpha(m) = [0,\ 0.2,\ 0.3,\ 0.5,\ 0.7,\ 0.9,\ 1.0,\ 1.0],$$

$$\beta(m) = [0.2,\ 0.4,\ 0.6,\ 0.8,\ 1.0,\ 1.0,\ 1.0,\ 1.0]$$

으로 할당한다. 그림 2.9, 2.10는 위와 같은 방식으로 BCH(63.57.3)code, BCH(32.26.3)code 두개를 사용하여 TPC를 구성하여 각 반복 횟수 따른 성능을 나타낸 것이다.

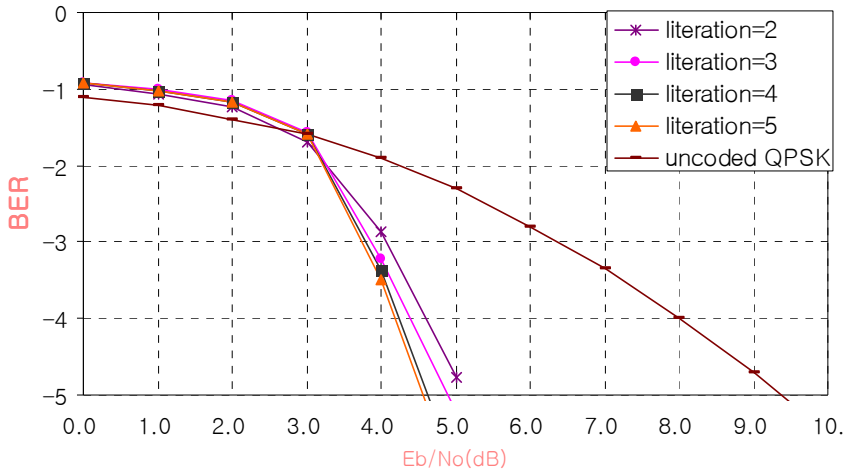


그림 2.9 BCH(63.57.3)TPC의 반복 횟수에 따른 성능

Fig. 2.9 Performance of BCH(63.57.3)TPC at each iteration number.

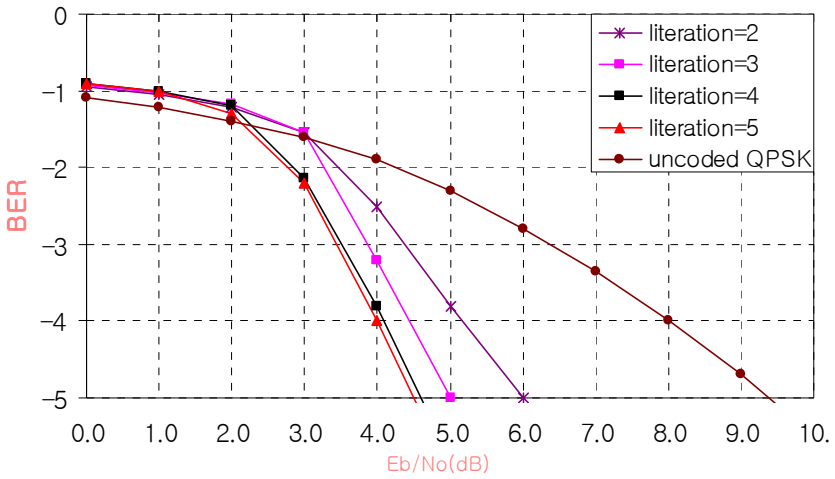


그림 2.10 BCH(31.26.3)TPC의 반복횟수에 따른 성능

Fig. 2.10 Performance of BCH(31.26.3)TPC at each iteration number.

제 3 장 병렬 Turbo Product Code 구조

3.1 부분 병렬 Turbo Product Code 분석 및 구조

앞에서 살펴본 그림 2.8과 같이 Pyndiah에 의해 제안된 복호기는 세로(또는 가로)를 복호하기 전에 반드시 가로(또는 세로)를 복호하여야 하기 때문에, 지연이 발생 된다. 따라서 지연을 감소 시키기 위해서 부분 병렬(P-parallel) TPC 복호기 구조가 제안 되었다[6]. 부분 병렬 복호기는 row(또는 column)복호기 내에 P개의 복호기를 사용하여 기존의 방식보다 P배 정도 지연을 감소 하고자 제안된 방식이다. 부분 복호기의 구성도와 복호과정은 그림 3.1과 그림 3.2와 같다.

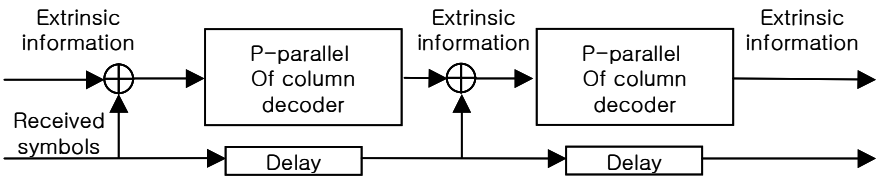


그림 3.1 부분 병렬 TPC 복호기

Fig. 3.1 P-parallel TPC decoder.

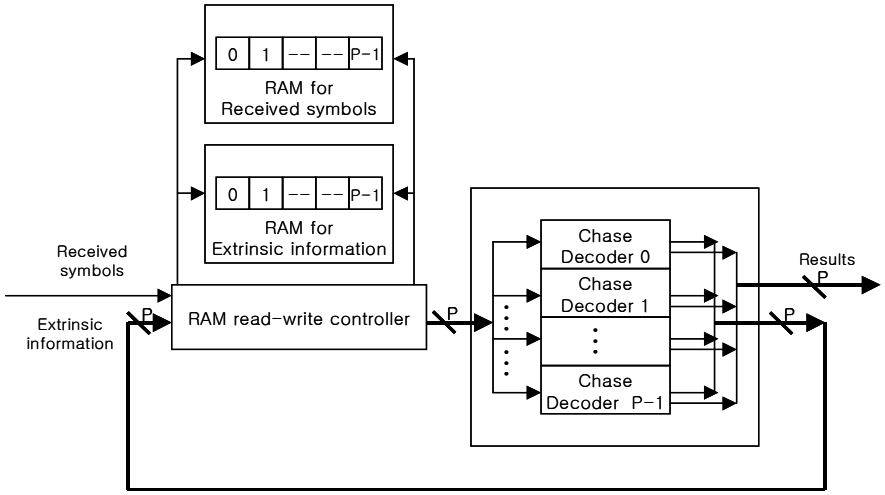


그림 3.2 부분 병렬 복호기의 복호 과정

Fig. 3.2 P-parallel procrssing TPC decoder

한번 반복을 기준으로 row(또는 Column)복호시 N/P만큼 시간이 기존 방식보다 지연이 줄어든다. 여기에서 N은 블록 크기를 말하고 P는 동시에 동작하는 복호기의 임의의 개수를 말한다. P개의 병렬 복호기를 실행 시킬 때 발생된 P개의 데이터는 동시에 0~(P-1)의 각각 다른 RAM에 저장된다. 예를 들어 64✕64 TPC부호가 있고 부분 병렬 복호기의 P가 8이라고 한다면 블록부호를 위한 8개의 복호기가 필요하고 이 복호기는 동시에 동작하여 기존의 방식보다 지연이 8배 감소하게 되고 따라서 복호속도는 8배 증가하게 된다. 그림 3.2에서 사용된 각각의 메모리는 8개가 존재하게 되고 각 메모리는 64✕8 bit로 구성되어

진다. P의 숫자가 증가 될수록 동시에 요구 되는 RAM의 수는 증가하고 동시에 저장하는 각각 RAM의 저장공간은 줄어든다. 따라서 이 방식은 구현하고자 하는 FPGA칩의 사양에 따라 요구되는 RAM에 따른 적절한 RAM수를 계산하여 P를 조절할 필요가 있다.

3.2 병렬 Turbo Product Code 분석 및 구조

앞에서 살펴본 부분 병렬 복호기는 지연은 감소 되나 동시에 P개의 복호기가 필요하므로 실제로 구현할 때 제작 비용이 기존방식보다 많이 들게 된다. 따라서 본 논문에서는 기존 방식 보다 지연이 작게 발생하고 부분 병렬 복호 방식보다 적은 비용이 사용되는 그림 3.3과 같은 병렬TPC 복호기를 제안한다. 그림 3.4에서와 같이 가로 세로 복호기에 대한 복호는 병렬로 이루어져지며, 복호 지연은 기존의 방식에 비해 절반 효과를 갖기 때문에 복호 속도는 2배 개선시킨다. 메트릭 $[W^{row}]$와 $[W^{col}]$은 가로와 세로 부호어에 대한 수신 신호의 extrinsic 정보 이며, 각 블록에 대한 복호를 마치고 block-by-block으로 같은 시간에 extrinsic 정보의 교환으로 복호가 수행되어진다.

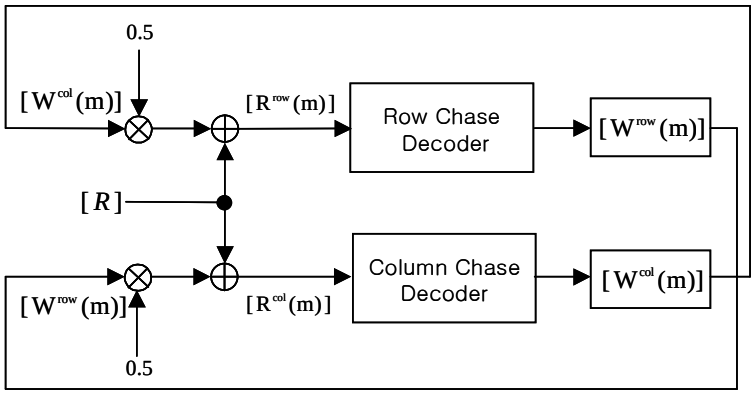


그림 3.3 병렬 복호기 구조 및 복호 과정

Fig. 3.3 Parallel decoder structure and decoder process.

첫 번째 반복 시, 즉 m=0, 일 시 $[R^{row}(0)] = [R^{col}(0)] = [R]$ 이라 두고, 가로 세로의 블록 길이는 n 이라 가정한다. 가로 세로에 대한 복호가 마친 후 업데이터 된 신호 벡터 $[R^{row}(m+1)]$, $[R^{col}(m+1)]$ 은 다음 반복을 위한 입력으로 전달 된다. 병렬적으로 업데이터 된 신호 벡터는 아래 식 (3.1)과 같다.

$$[R^{row}(m+1)]=[R]+\alpha(m)\cdot[W^{col}(m)]$$
$$[R^{col}(m+1)]=[R]+\alpha(m)\cdot[W^{row}(m)] \quad (3.1)$$

3.3 모의 실험 결과

본 논문에서 모의 실험 시 BCH 부호를 사용하여 구성 하였고 부호화율이 같은 두개의 BCH부호를 사용하였다. serial (n,k,δ) 는 두개의 BCH (n,k,δ) code를 각각 row와 column에 사용한 TPC를 말하고 같은 방식으로 parallel (n,k,δ) 은 두개의 BCH (n,k,δ) code를 각각 row와 column에 사용하고 병렬 복호를 한 TPC를 말한다. Chase 알고리즘을 위해서 가장 신뢰성이 없는 비트를 선택 시 $p=4$ 으로 하였다. 그리고 반복 복호를 위한 scaling factor로서

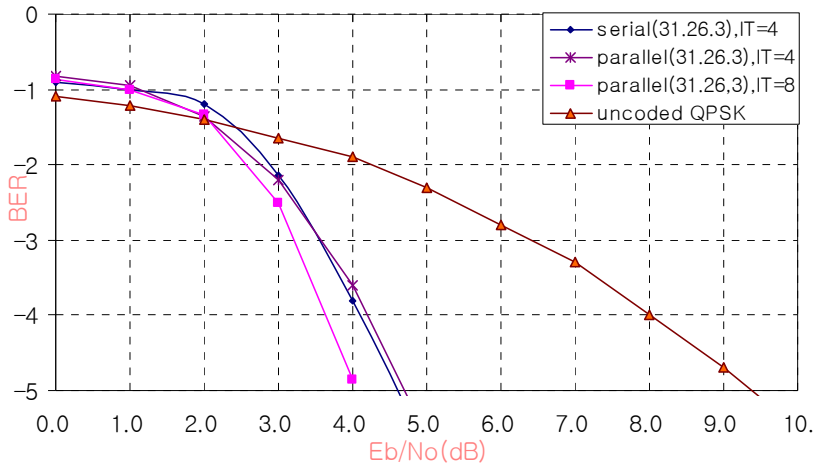
$\alpha(m)$= [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5],

$\beta(m)$= [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0],

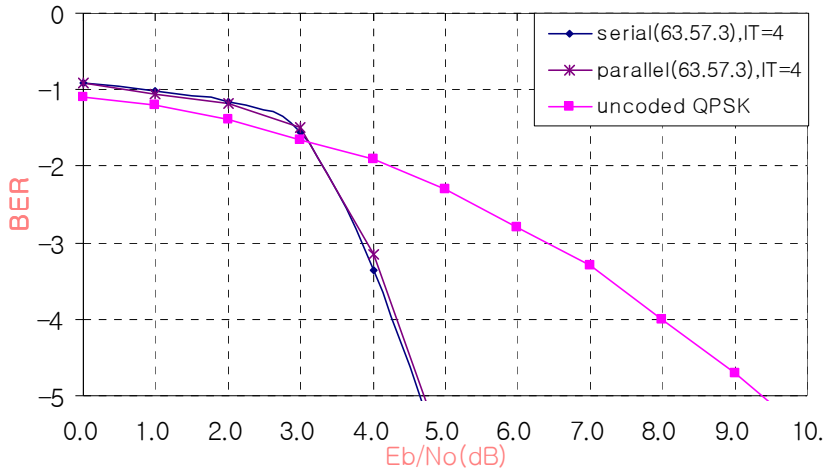
로 고정하였다.

β 는 반복시 1로 구성하였기 때문에 Chase 복호기의 출력 Decoder의 연판정 값을 구하기 위해서 전체 m에 대한 값이 1이기 때문에 곱셈이 필요치 않으며 α 는 0.5로 고정했기 때문에 가로 세로에 대한 복호기의 extrinsic 정보의 절반에 해당하는 bit 만큼 우측으로 shift 하면 되기 때문에 H/W구현이 간단해진다. 이러한 조건을 중심으로

반복 횟수를 4로 하였을 때 결과는 그림 3.4과 같다. BER=10^{-4}을 기준으로 Pyndiah가 제안한 방식과 비교하였을 때 본 논문에서 제안한 병렬 복호 방식과 거의 성능이 비슷함을 알 수 있다. 그러나 반복 횟수 8번을 같는 parallel(31.26.3)에서 반복횟수 4를 같는 serial (31.26.3)와 비교시 각각은 같은 지연을 가지지만 병렬방식이 성능이 0.4dB정도 좋아짐을 알 수 있다.



(a)



(b)

그림 3.4 기존방식과 병렬 복호방식의 성능비교(IT는 반복복호의 횟수) : a) parallel(31.26.3)의 성능; b) parallel(63.57.3)의 성능

Fig. 3.4 Performance of original and parallel decoded BCH TPCs on AWGN channel. (IT denotes number of iteration) : a) Performance of parallel(31.26.3); b) Performance of parallel(63.57.3).

제 4 장 최적 파라미터 설계

4.1 수신 신호의 양자화 범위에 따른 성능

수신 실수(float) 신호를 양자화를 하기 전에 양자화 하기 위한 신호의 범위를 결정하기 위한 모의 실험을 하였다. parallel(31.26.3)을 이용하여 송신된 신호를 수신하였을 때 모두 7bit 양자화를 하여 6개의 범위에 대한 모의 실험을 하였다. 모의 실험 결과 그림 4.1 과 같이 +1.3~-1.3에 대한 범위를 양자화 하였을 때 최적임을 알 수 있다.

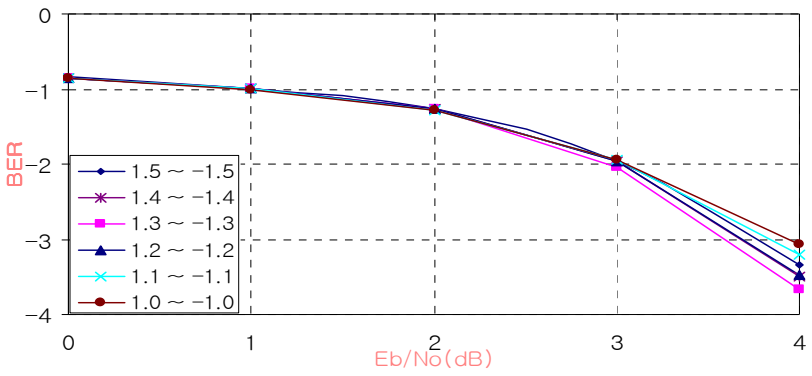


그림 4.1 양자화를 위한 수신신호의 최적 범위

Fig. 4.1 Receive signal optimum range for quantization.

4.2 수신 신호의 양자화 비트 수에 따른 성능

앞 절에서 수신된 BPSK 양자화 구간은 +1.3 ~ -1.3로 결정한 구간에서 양자화 비트 수에 따른 성능을 모의 실험 하였다. 4, 5, 6, 7, 8비트로 각각을 정규화하였고 parallel(31.26.3)을 사용하여 부호화 하였다. 그림 4.2 에서 알 수 있듯이 5비트가 최적의 비트 수임을 알 수 있다.

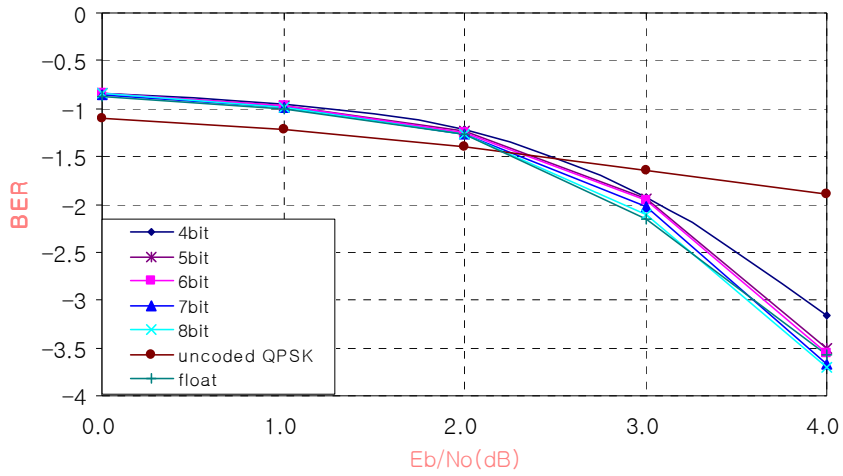


그림 4.2 수신신호 비트 수에 따른 TPC 복호기의 성능

Fig. 4.2 Performance of TPC decoder as a number of recovered signal bit.

제 5 장 병렬 Turbo Product Code VHDL 설계

수신 신호는 5bit로 양자화 하는 것이 4장에서 최적임을 알 수 있었다. 수신 신호가 5bit일 경우에 extrinsic 정보는 반복 복호가 진행될 경우 carry가 발생하므로 6bit로 설정하면 된다. 그리고 가로(또는 세로)복호가 끝난 후에 발생된 extrinsic 정보는 scaling factor에 의해서 절반에 해당되는 정보만큼 오른쪽으로 shift시켜서 다시 5bit로 만든 다음 수신신호와 합을 구해서 그 결과가 세로(또는 세로)복호기의 입력이 된다.

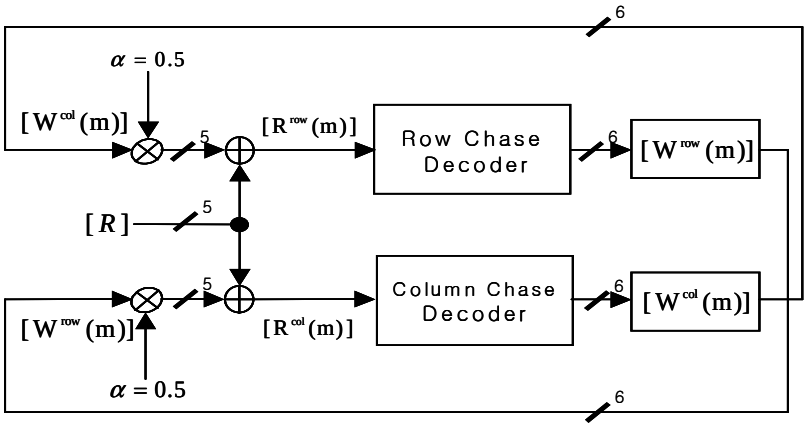


그림 5.1 TPC 복호기 수신 시스템의 블록도

Fig. 5.1 A Block diagram of TPC decoder.

본 논문에서 가로(또는 세로)복호기를 VHDL언어로 구성 할 때

hamming(15,11,3)decoder를 사용 하였으며 $p=4$ 하여 복호기 설계하였다. 그림 5.2 은 chase 복호기의 블록도이다.

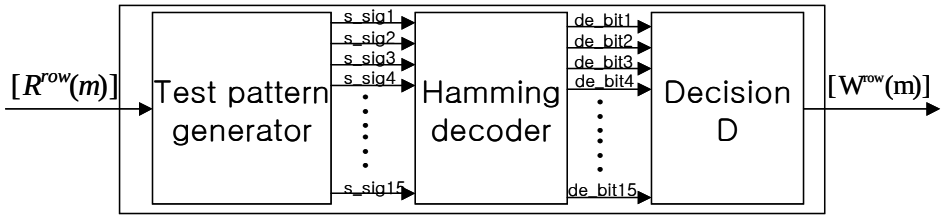


그림 5.2 chase 복호기의 블록도

Fig. 5.2 Block diagram of chase decoder.

여기에서 s_sig신호는 chase algorithm에 의해서 발생된 16개 신호와 수신신호의 절대값의 합이다.

5.1 Test pattern generator 설계

test pattern 발생시 $p=4$ 이기 때문에 총 16개의 test pattern을 발생 시켜야 한다. 2장에서 설명한 chase algorithm에 의해서 생성시킨다. 이 방법을 이용할 시 수신된 신호의 가장 신뢰도가 없는 비트에 1을 발생시키고 나머지 비트는 0을 배치하고, 두번째 신뢰가 없는 비트에 1을 발생시키고 나머지 비트에 0을 배치하면서 16개의

test pattern을 발생해야 한다. 그러나 가장 신뢰도가 작은 순서대로 비트를 찾아서 test pattern을 발생시키는 것이 회로가 복잡해지고 시간이 오래 걸리므로 신뢰도가 없는 비트가 발생하는 순서대로 test pattern을 발생 시켰다. 이것을 모의 실험을 통해서 본 성능은 그림 5.3과 같이 기존의 방식과 일치함을 알 수 있다.

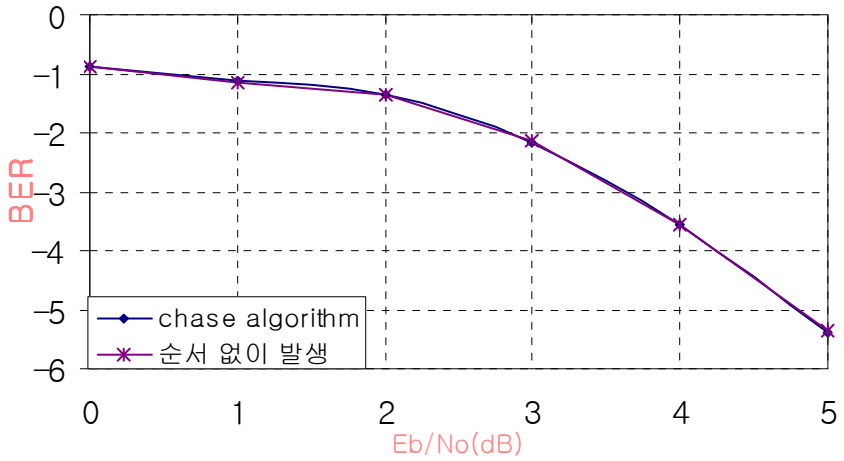


그림 5.3 기존 방식과 새로운 방식과의 비교

Fig. 5.3 Compare conventional method to modified method.

앞에서 설명한대로 test pattern을 발생시킨 것을 그림 5.4에서 확인할 수 있다.

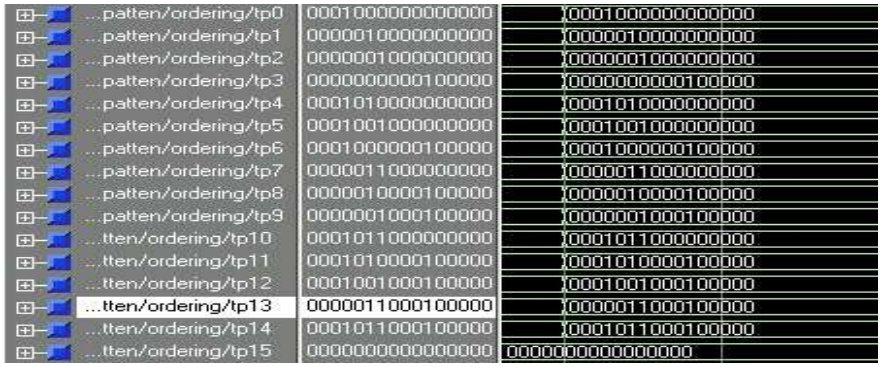


그림 5.4 test pattern 의 발생

Fig. 5.4 Generate test pattern.

test pattern과 수신 신호의 절대값과의 합인 s_sig는 그림 5.5와 같이 구해 진다.

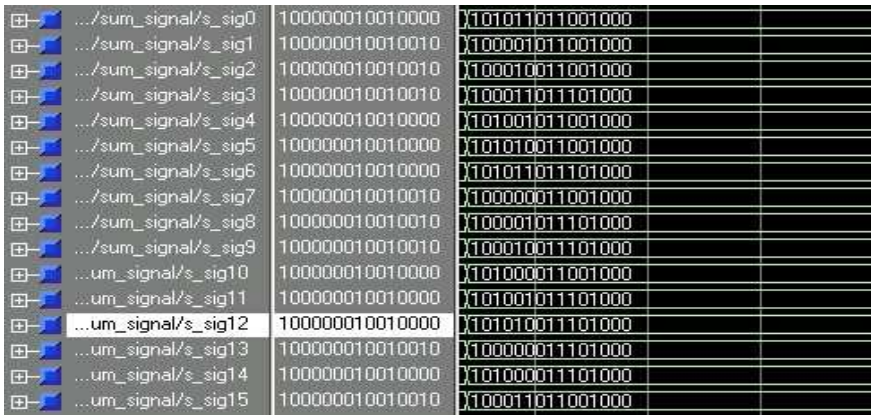


그림 5.5 수신신호의 절대치와 test pattern의 합

Fig. 5.5 Add test pattern to the absolute value of received signal.

5.2 hamming decoder 설계

다음 복호에 사용되는 extrinsic 정보를 얻기 위해서 현재 복호된 값과 현재 복호기로의 이전 입력 값이 필요하다. 그러기 위해서 복호된 값이 필요하다. 본 논문에서 VHDL로 설계시 hamming(15,11,3) decoder를 사용하였다. hamming decoder은 s_sig0 ~ s_sig15를 가지고 복호 한다. 이에 시뮬레이션 결과는 그림 5.6과 같다. 여기에서 de_bit는 s_sig를 복호한 비트이다.

그림 5.6 hamming decoder의 출력값

Fig. 5.6 Output of hamming decoder.

5.3 최적의 비트 결정 방법

복호된 비트들(de_bit~debit14)중에서 유클리드 거리를 이용해서 가장 작은 값을 선택해서 가로(또는 세로)의 복호 비트로 선택한다. 여기서 유클리드 거리를 구할 때 한 블록을 이루는 값인 15bit에 해당하는 각각의 수신신호와 그에 해당하는 de_bit의 값의 차를 제곱하고 더해서 최소값을 선택하는데 본 논문에서는 제곱을 하지 않고 두 값의 차를 절대치하여 값을 더하였다. 그림 5.7은 각 디코더에 대한 복호비트와 수신신호 간에 유클리드 거리를 나타내는 그림이다. 여기에서 sum_s는 각 블록에 대한 유클리드 거리를 나타내고 있다.

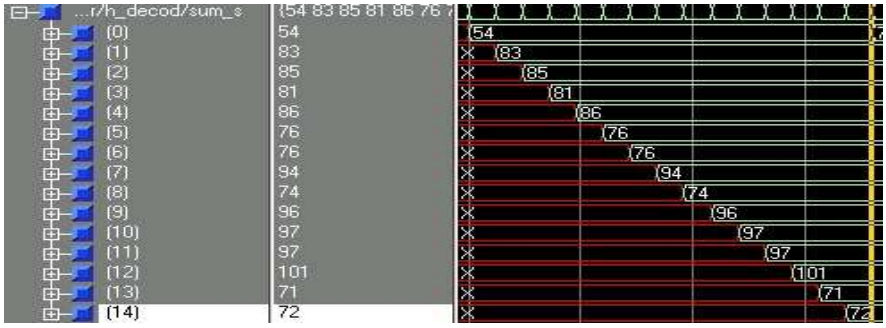


그림 5.7 복호된 비트의 유클리드 거리

Fig. 5.7 Euclidian distance of decode bit.

이렇게 나온 유클리드 거리중 가장 작은 값을 결정하는 블록도는 그림 5.8과 같다.

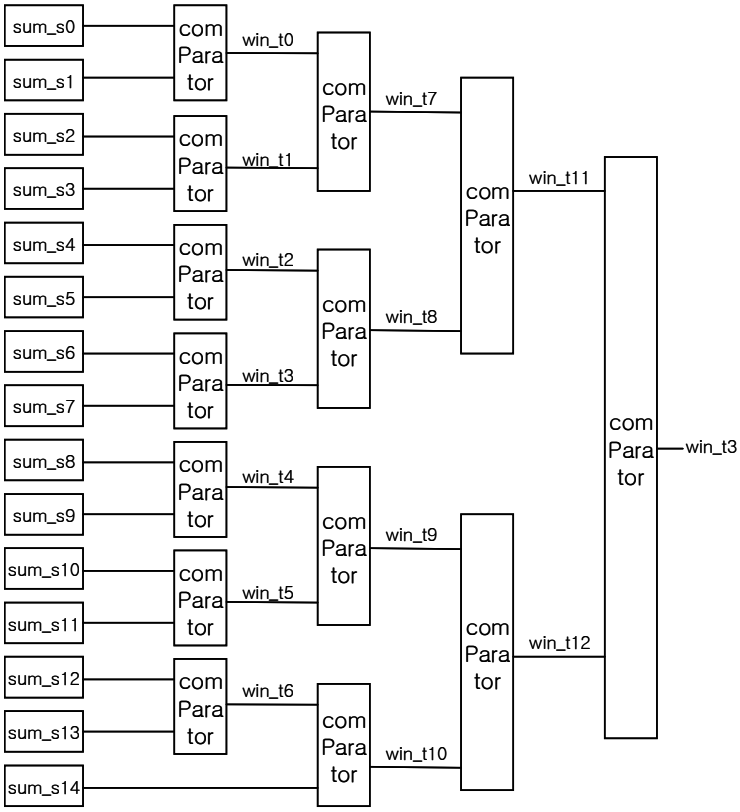


그림 5.8 최소 거리 결정 블록도

Fig. 5.8 Block diagram for minimum distance decision.

비교기는 두개의 입력 중에서 위쪽 입력의 유클리드 거리가 작으면 0을 아래쪽에서 작으면 1을 출력한다. 이 출력 값은 win_t에 입력이

되고 최종적으로 win_t13이 나왔을 때 가장 작은 유클리드 거리가 결정된다. 결정 방법은 예를 들어 win_t0=0, win_t7=0, wint_t11=0, wint_t13=0 이면 가장 작은 유클리드 값은 sum_s0가 되고 첫 번째 디코드에서 나온 비트가 최종 결정 비트가 된다. 그림 5.9는 위에서 언급한 모든 것들을 통합하여 시뮬레이션한 파형이다. 입력신호인 data1과 복호된 신호인 decodebit2가 일치하므로 프로그램의 유효성을 증명할 수 있다.

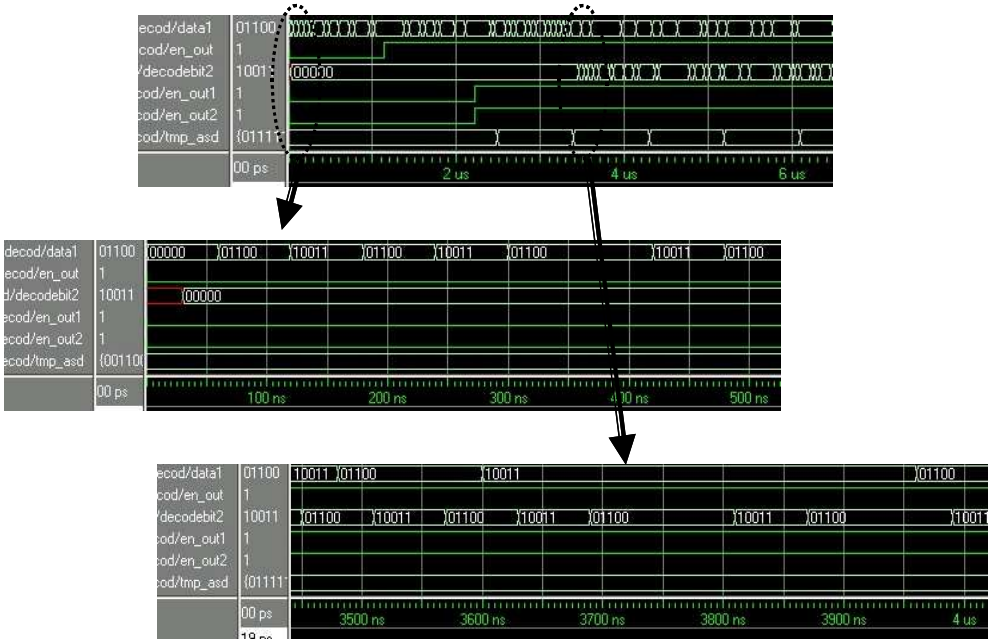


그림 5.9 복호된 신호

Fig. 5.9 Decoded bit.

제 6 장 결론

기존의 TPC복호기는 row복호 후 column 복호를 하기 때문에 지연으로 인한 복호 속도 저하가 생겼다. 본 논문에서는 고속 데이터 통신 전송 및 동영상 등을 포함한 무선 멀티미디어 전송을 실현 하기 위해서 병렬 TPC 복호기를 제안하였다.

병렬로 동작하는 복호기는 row와 column이 부분에서 동시에 복호하고 동시에 extrinsic 정보를 넘겨주기 때문에 기존의 방식보다 지연시간이 작다. 그리고 scaling factor와 Reliability Factor가 0.5와 1로 고정 되기 때문에 하드웨어로 구현 시 회로가 간단히 설계 될 수 있다. 병렬 TPC복호기는 BER=10^{-4} 을 중심으로 살펴 볼 때 기존의 방식에 0.08dB 성능 열화가 있다. 구현 상의 메모리의 요구량은 늘어난다. 하지만 기존 방식의 한번 반복을 기준으로 볼 때 같은 수의 메모리가 사용되고 성능이 0.28dB정도 향상됨을 알 수 있다.

문헌[6]에서 제안된 방식은 row나 column 데이터를 복호하기 위해 P개의 내부적인 복호기가 필요며, 이로 지연은 본 논문에서 제안한 방식보다 줄일 수 있으나, 부호화 데이터 N이 증가 할수록 H/W가 복잡해지고 power 소모량, 사이즈 면에서 효율성을 가지지 못한다.

H/W 복잡성을 절충하면서 Xiujun Zhang가 제시한 방법과 본 논문에서 제안한 방식을 결합하기 위해 속도면이나 H/W 복잡도면에서 최적의 파라미터를 절충하여 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] 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.

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

[4] S.Sjoholm and L.Lindh, *VHDL FOR DESIGNERS* . Prentice-Hall, 1997.

[5] 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.

[6] Xiujun Zhang, Ming Zhao, Shidong Zhou and Jing Wang, “Parallel Decoding of Turbo Product Codes for High Data Rate Communication”, *IEEE VTC 2003-Spring conference*, April 2003.

[7] R.. Pyndiah, A. Glavieux, A.Picart, and S. Jacq “Near opimum decoding of product codes” *in Proc.IEEE GLOBECOM’94 Conf.,* San Francisco, CA, vol.1/3, , pp.339-343, Nov.-Dec. 1994.

[8] N. Yn, Y.kim, and P.Lee, “Iterative decoding of product codes composed of extended Hamming codes,” *Proc. Fifth IEEE Symposium on computer and communications*, pp.732-737, 2000.

[9] 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.

[10] S.Lin and D.J.Costello, *Error Control coding : Fundamentals and Applications*. Prentice-Hall, 1983.

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

감사의 글

대학원 2년 이라는 시간 동안 못난 저를 여기에 있기 까지 주변의 많은 분들이 저의 부족한 점을 깨우쳐 주셨습니다. 그 분들과 함께 했었던 시간들은 저에게는 참으로 많은 것을 배울 수 있었던 행운의 시간이었습니다. 먼저, 본 논문이 완성되기까지 시종일관 세심한 지도와 따뜻한 격려를 해주시고 미흡한 저를 지금까지 이끌어주신 지도교수 정지원 교수님께 깊은 감사를 드립니다. 늦은 밤 작은 문제로 상담을 해주셨을 때 웃으시면서 밤새 조언을 해주셨던 교수님의 은혜는 소중히 간직하면서 영원히 잊지 않겠습니다. 그리고 논문의 미비점을 보완하여 보다 충실한 내용이 될 수 있도록 논문 심사를 맡아주신 조형래 교수님, 김기만 교수님께 감사드리며, 항상 새로운 가르침을 주시고, 조언을 아끼지 않으셨던 김동일 교수님, 민경식 교수님, 강인호 교수님께도 감사드립니다.

항상 옆에서 힘이 되어준 성준이 선배, 같은 실험실 후배로서 나를 도와준 상진이, 인기, 연구실에 들어와서 힘든 사항에서도 군소리 없이 묵묵히 따라온 학부생 덕군에게 고마움을 전하며, DSP 연구실의 외형이형 윤준이형 같이 대학원 생활을 한 95학번 동기 도연, 동식, 진산, 재교를 비롯한 5년 동안 수업을 같이한 순영, 은정, 그리고 대학원 생활을 같이 한 대학원생들에게 감사의 마음을 전합니다. 또한 서울에서 열심히 일하면서 나를 걱정하고 격려해 준 여러 동기들과 선후배님들, 그리고, 익수, 상조, 정광, 승민, 승현, 세진이에게 감사함과 앞으로의 무궁한 발전을 기대합니다. 지금 까지 항상 옆에서 나를 아껴 주고 마음의 위로가 되어준 나의 짝 현정이에게도 감사의 마음을 전합니다.

마지막으로, 항상 절 믿고 응원하고 계신 아버님과 부족한 절 위해 항상 희생하시는 어머님께 지면으로나마 사랑한다는 말을 전하고, 하나 뿐인 동생 정은이에게 감사의 마음을 전합니다.

이렇게 모든 분들의 그 은혜에 보답하기 위해 안이한 생각을 하지 않고 최선을 다해서 항상 발전하는 모습으로 보답하겠습니다.