工學碩士 學位論文

Radix-4 방식 고속 터보 MAP 복호기의 구현에 관한 연구

A Study on Implementation of High-Speed Turbo MAP Decoder Using the Radix-4 Method

指導敎授 趙 炯 來

2002 年 **2** 月

韓國海洋大學校 大學院

電 波 工 學 科

金 相 勳

本 論文을 金相勳의 工學碩士 學位論文으로 認准함.

委員長 : 工學博士 金 東 一 (印)

委 員 : 工學博士 姜 仁 鎬 (印)

委 員 : 工學博士 趙 炯 來 (印)

2002 年 **2** 月

韓國海洋大學校 大學院

電 波 工 學 科

金 相 勳

차 례

ABSTRACT

For wireless communications, forward error correction(FEC) schemes are an important tool for improving communications reliability. Turbo codes have been shown to perform near the Shannon's 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.

Recently, the trend of wireless communication is changed from the conventional narrow-band voice service to the wide-band multimedia service. Therefore, it is highly required to develop the high-speed turbo decoder structure. An important problem in high-speed applications is decoding delay inherent to turbo decoding. Conventional Radix-2 MAP decoder has difficulty in applications of high-speed wireless communication mainly due to delay in interleaving/deinterleaving and iterative decoding.

To solve the problem with latency of turbo decoder, in this thesis a Radix-4 MAP decoding algorithm and many parameters were adopted such as branch metric, forward/backward state metric, etc. Simulation results over AWGN channel show that the BER performance of the Radix-4 MAP turbo decoding is almost the same as that of Radix-2 MAP turbo decoding.

Radix-4 turbo MAP decoder was compared with the conventional one in terms of decoding speed. The decoding speed of the Radix-4 MAP turbo decoder is faster by 2.4 times at least than conventional one in the case of one iteration of turbo decoding.

Nomenclature

C_k : k 시점에서의 전송 심볼

d_k : k 시점에서의 입력 정보 비트

I : iteration 반복 횟수

L_c : 채널 신뢰도

N : Radix-2 방식에서의 인터리버 크기

N_s : Radix-4 방식에서의 심볼인터리버 크기

p_k : k 시점에서의 I 채널 잡음 신호

q_k : k 시점에서의 Q 채널 잡음 신호

R_k : k 시점에서의 수신 심볼

x_k , y_k : k 시점에서의 출력 정보 비트

Z_k : k 시점에서의 estrinsic 정보

α_k^m : k 시점에서 상태노드(m=0,1,2,3)에 따른 순방향 state metric 값

β_k^m : k 시점에서 상태노드(m=0,1,2,3)에 따른 역방향 state metric 값

δ_k^m : k 시점에서 상태노드(m=0,1,2,3)에 따른 branch metric 값

λ_k^m : k 시점에서 상태노드(m=0,1,2,3)에 따른 사후확률 값

제 1 장 서 론

무선통신 시스템은 무선채널의 특성으로 비트 오류가 발생하기 쉬우며 이를 정정하기 위해 사용되는 채널부호는 무선통신시스템에서 매우 중요한 요소기술이다. 위성통신, 이동통신등에서 사용되는 채널부호는 일반적으로 연판정(soft decision)이 가능한 길쌈부호(convolutional code)와 연집오류특성에 강한 RS(Reed-Solomon)부호를 결합한 연접부호(concatnate code)를 사용한다[1]. 연접부호를 이용한 오류제어방식 또한 Shannons 한계와 다소 큰 격차를 보이고 있으며, Shannons 한계에 근접한 성능을 나타내는 Turbo Code 에 대한 연구가 집중되고 있다.

1993 년 Berrou 등에 의해 제안된 터보부호는 Eb/No 0.7dB, 부호율 1/2 에서 비트오류확률의 성능을 보였다[2]. 터보부호는 연판정 입/출력(soft-in/soft-out)이 가능하고, 정보신호에 대해서 서로 다른 인터리버에 의해 분리된 2 개이상의 구성코드(component code)들이 병렬연접(parallel concatenation)된 구성을 하고 있다. 터보부호의 기본 개념은 선행하는 구성코드의 복호기 soft decision ouput 을 다시 나머지 복호기에 입력하고 이러한 과정을 반복함으로써 향상된 성능을 가져온다. 터보부호의 복호기로는 SOVA(Soft Output Viterbi Algorithm), MAP(Maximum A Posteriori), Sub-MAP 복호기등이 있는데[3]-[5], 채널의 잡음분산평가가 필요하다는 단점이 있지만 일반적으로 성능이 우수한 MAP 을 사용한다[6],[7]. 이러한 MAP 기반의 터보부호는 MAP 복호기의 복잡성과 많은 연산량, 아주 큰 인터리버 블록크기로 인해 음성, 데이터, 동영상을 포함한 무선 멀티미디어 서비스를 요구하는 고속 무선 통신시스템의 채널부호로는 문제점을 가지고 있다. 따라서, 본 논문에서는 고속 터보 복호 알고리즘을 제시하고 VHDL 을 이용하여 Log-MAP 기반의 터보 복호기 설계를 한다.

터보 복호기를 고속화를 할 수 있는 방법은 고속뎃셈/곱셈 알고리즘을 적용하는 방법과 복호기구조를 한 클럭에 한비트를 복호하는 기존의

Radix-2 방식이 아니라 한클럭에 2 비트를 복호하는 Radix-4 방식의 적용이 필수적이다.

따라서, 본 논문에서는 Radix-4 방식을 적용하여 가우시안 채널 상에서 기존의 Radix-2 방식의 터보 MAP 복호기의 성능을 비교 하였다. 비교 결과 성능이 거의 유사함을 알 수 있었다. 또한 VHDL(Very high speed integrated circuit Hardware Description Language)을 이용하여 Radix-4 터보 MAP 복호기를 ALTERA 사의 FLEX10K100 CPLD(Complex Programmable Logic Device)에 구현 및 타이밍 시뮬레이션 한 결과 Radix-4 방식이 Radix-2 방식에 비해 한 번 iteration 시 약 2.4 배 고속화됨을 알 수 있었다.

제 2 장 Radix-4 방식의 고속 터보 MAP 복호 알고리즘 제안

기존 Radix-2 방식을 확장해 한번의 연산으로 2 비트를 동시에 복호 가능한 Radix-4 방식의 MAP 복호기를 제안하였다. 기존의 트렐리스 구조에서 2 개의 시점을 하나의 시점으로 간주하는 Radix-4 방식을 적용하기위해 branch metrics, forward recursive metrics, backward recursive metric 공식을 제안하였다.[8]

2.1 터보 코드의 부호기 및 복호기 구조

구성코드가 RSC 이고 부호율이 1/2 인 터보코드의 부호기 및 복호기 구조를 그림 2-1 과 그림 2-2 에 나타내었다.

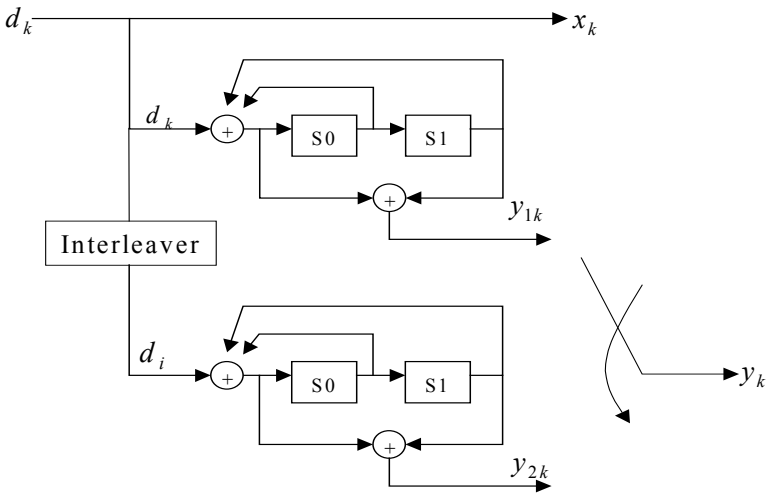


그림 2-1. Half-rate 병렬연접 RSC 부호기

Fig. 2-1. Half-rate Parallel concatenation of two RSC encoders.

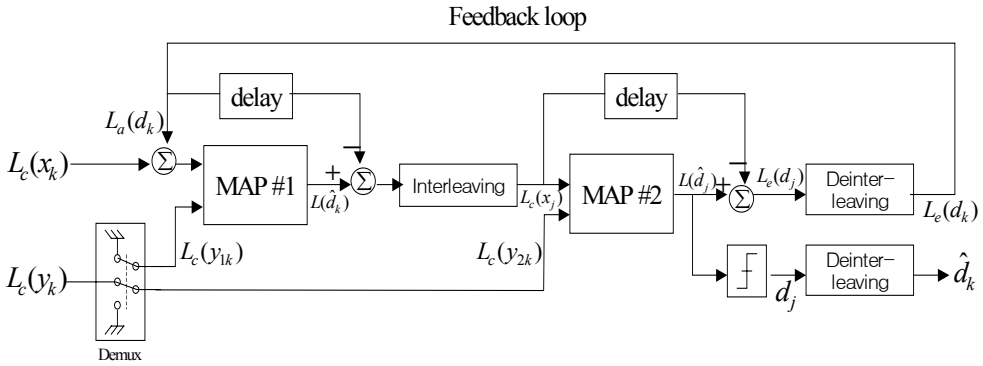


그림 2-2. Radix-2 기반의 터보 MAP 복호기 구조

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

1 비트 입력(d_k)에 2 비트가 출력(x_k, y_k)가 되는데 x_k는 d_k와 같고 y_k는 y_{1k}, y_{2k}를 교대로 puncturing 한 값으로서 패리티 정보이다. 출력신호를 QPSK 변조시키면, $+/-1$ 값을 가지는 전송신호 a_k와 b_k는 다음과 같이 나타낼 수 있다.

$$\begin{aligned} a_k &= 2 \times d_k - 1 \\ b_k &= 2 \times y_k - 1 \end{aligned} \tag{2.1}$$

시간 k에서 전송된 신호(a_k, b_k)들을 심볼 C_k라 정의하면, 가우시안 무기억 채널에 입력되는 전송신호열은 다음과 같이 주어진다.

$$C_1^N = (C_1, C_2 \ldots\ldots C_k \ldots\ldots C_N) \tag{2.2}$$

시간 k에서 분산 σ^2을 가지는 채널잡음 p_k, q_k과 더해진 수신신호 x_k, y_k는

$$x_k = a_k + p_k$$
$$y_k = b_k + q_k \tag{2.3}$$

으로 나타내지며, 수신 심볼 $R_k = (x_k, y_k)$ 은 다음과 같다.

$$R_1^N = (R_1, R_2 .....R_k .....R_N) \tag{2.4}$$

MAP 복호기는 다음 식(2.5)과 같이 정의되는 LLR(Log-Likelihood Ratio)를 이용하여 복호한다.

여기서, $P_r(d_k = i \mid R_1^N) = \sum_m \lambda_k^i(m)$ 이고, 부호기의 메모리가 v 일때, m 은 부호기의 상태번호이다. ($m = 0,1,...2^{v-1}$)

$$\begin{aligned} L(d_k) &= \log\left(\frac{\Pr(dk = 1 \mid observation)}{\Pr(dk = 0 \mid observation)}\right) \\ &= \frac{P_r(d_k = 1, S_k = m \mid R_1^N)}{P_r(d_k = 0, S_k = m \mid R_1^N)} \\ &= \log \frac{\sum_m \lambda_k^1(m)}{\sum_m \lambda_k^0(m)} \end{aligned} \tag{2.5}$$

이것은 MAP 복호기의 softoutput 이며, 마지막엔 아래와 같이 $L(d_k)$ 를 임계값 0 와 비교함으로써 복호기는 hard-decision 할 수 있다[7].

$$\begin{aligned} &if \quad L(d_k) \geq 0, \; the\ decoded \quad bit\ is\ 1 \\ &if \quad L(d_k) \leq 0, \; the\ decoded \quad bit\ is\ 0 \end{aligned}$$

식(2.5)에서 정의된 결합확률을 계산하기 위해 다음과 같이 확률함수를 정의한다.

$$\begin{aligned}\alpha_k^i(m) &= P_r(d_k = i, S_k = m, R_1^k) \\ &= \sum_{j=0}^{1} \alpha_{k-1}^{b(j,m)} \delta_{k-1}^{j,b(j,m)}\end{aligned} \tag{2.6}$$

$$\begin{aligned}\beta_k^i(m) &= P_r(R_{k+1}^N \mid d_k = i, S_k = m, R_1^k) \\ &= \sum_{j=0}^{1} \beta_{k+1}^{f(j,m)} \delta_k^{j,m}\end{aligned} \tag{2.7}$$

α_k^i 는 순방향 recursive function 이고, β_k^i 는 역방향 recursive function 이다. 그리고, $b(i,m)$ 은 격자도 상에서 입력이 i 이고 현재 상태를 m 으로 천이시킨 이전의 상태값을, $f(i,m)$ 은 상태 m 에서 bit i 가 입력되었을 때 다음에 천이될 상태값을 나타낸다. 식(2.6), 식(2.7)을 식(2.5)에 대입한 $L(d_k)$ 는 다음과 같다.

$$L(d_k) = \frac{\sum_m \alpha_k^m \beta_{k+1}^{f(0,m)} \delta_k^{0,m}}{\sum_m \alpha_k^m \beta_{k+1}^{f(1,m)} \delta_k^{1,m}} \tag{2.8}$$

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

$$\begin{aligned}\delta_k^m &= P_r(d_k = i, S_k = m, R_k) \\ &= \exp\left(\frac{2}{\sigma^2} \times (X_k i + Y_k C_k^{i,m})\right)\end{aligned} \tag{2.9}$$

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

2.2 Radix-4 방식의 터보 MAP 복호기 구조

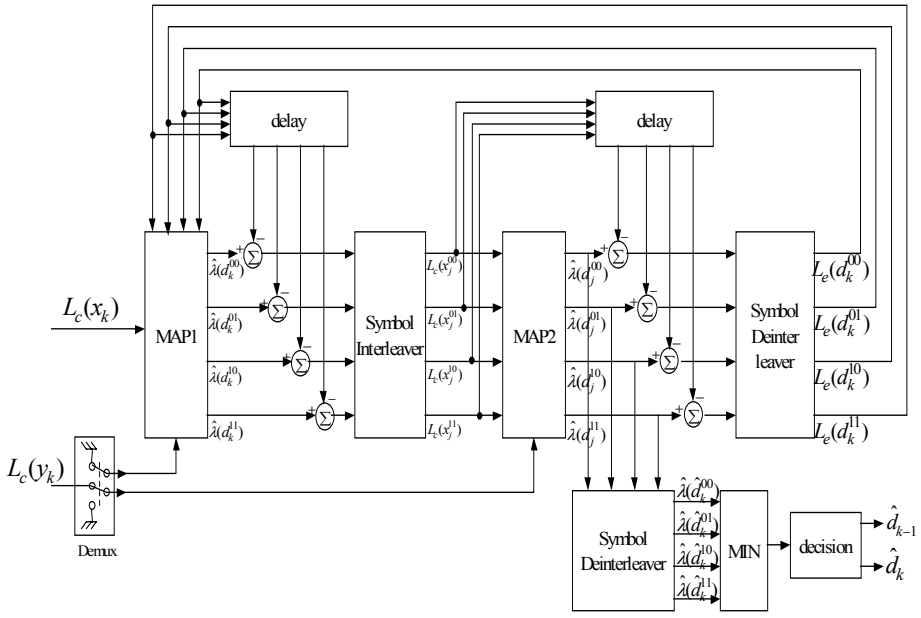
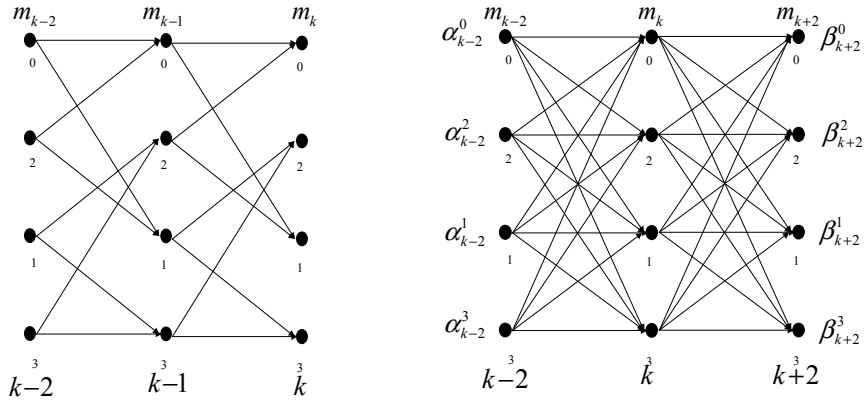


그림 2-3. Radix-4 기반의 터보 MAP 복호기 구조

Fig. 2-3. Block diagram of MAP turbo decoder based on radix-4.

MAP1 과 MAP2 가 직렬로 구성된 radix-4 기반의 터보 복호기 구조는 그림 2-3 과 같다. MAP 기반의 터보코드 복호시, Radix-2 방식은 임의의 k 시점에서 복호할 때 $k-1$ 시점에서의 순방향 state metric α_{k-1}^m, $k+1$ 시점에서 역방향 state metric β_{k+1}^m, 그리고 k 시점에서 branch metric δ_k^m 를 이용하여 k 시점에서 "0" 과 "1"에 대한 Log Likelihood Ratio 인 LLR 을 구하여 1 비트를 복호한다. 이에 반해 radix-4 방식은 두상태 이전인 $k-2$ 시점에서 β_{k+2}^m 를 구하여 k 시점에서 2 비트를 동시에 복호하기 때문에 Radix-2 방식보다 속도가 2 배 빠르며, 저장되는 메모리도 절반으로 감소할 수 있다. 과거 2 단의 수신비트를 입력 받아 한꺼번에 처리하는 radix-4 방식의 trellis 구조는 Radix-2 방식의 2 개 시점을 하나의 시점으로 간주하여 처리하며 Radix-2 방

식에서 radix-4 방식으로 trellis 구조 변경은 그림 2-4 와 같다.



(a) 4-state Radix-2 trellis (b) 4-state radix-4 trellis

그림 2-4. 4 상태 trellis 구조

Fig. 2-4. 4-state Trellis Structure.

Radix-4 trellis 상에 있어서 시간 $k-2$ 에서 k 까지의 α_k^m 연산과 k 에서 $k+2$ 까지의 β_k^m 연산은 다음 식(2.10)과 같이 표현되며

$$\begin{aligned} m_{k-2}(d_{k-1}, d_k, m_k) &= (d_{k-1} \oplus d_k \oplus m_{0,k}) \| (d_k \oplus m_{0,k} \oplus m_{1,k}) \\ m_{k+2}(d_{k+1}, d_{k+2}, m_k) &= (d_{k+1} \oplus d_{k+2} \oplus m_{1,k}) \| (d_{k+1} \oplus m_{0,k} \oplus m_{1,k}) \end{aligned} \tag{2.10}$$

d_k 는 k 시점에서의 uncoded data bit 이고, m_k 는 k 시점에서의 state 번호 그리고 $(a \| b)$ 는 a 와 b 의 연접(concatenation)을 뜻한다.

2.3 메트릭함수와 사후확률의 유도

평균이 0 이고 분산이 σ^2 인 AWGN 채널에서의 Radix-4 방식의 branch metric δ_k^m 은 식(2.11)과 같이 유도할 수 있다.

$$
\begin{aligned}
\delta_k^{p,m} &= P_r(D_k = p, S_k = m_1, R_k) \\
&= P_r(D_k = i_{k-1} \| i_k, S_k = m_k, R_k) \\
&= P_r(d_{k-1} = i_{k-1}, S_k = m_{k-1}, R_{k-1}) P_r(d_k = i_k, S_k = m_k, R_k) \\
&= P_r(R_{k-1} \mid d_{k-1} = i_{k-1}, S_{k-1} = m_{k-1})\ P_r(d_{k-1} = i_{k-1}, S_{k-1} = m_{k-1}) \\
&\quad \cdot P_r(R_k \mid d_k = i_1, S_1 = m_k)\ P_r(d_1 = i_1, S_k = m_1) \\
&= P_r(R_{k-1} \mid d_{k-1} = i_{k-1}, S_{k-1} = m_{k-1})\ P_r(S_{k-1} = m_{k-1} \mid d_{k-1} = i_{k-1}) P_r(d_{k-1} = i_{k-1}) \\
&\quad \cdot P_r(R_k \mid d_1 = i_1, S_1 = m_k)\ P_r(S_1 = m_k \mid d_k = i_k) P_r(d_k = i_k) \\
&= P_r(x_{k-1} \mid d_{k-1} = i_{k-1}, S_{k-1} = m_{k-1})\ P_r(y_{k-1} \mid d_{k-1} = i_{k-1}, S_{k-1} = m_{k-1}) \varsigma_{k-1}^{i_{k-1}} / 2^v \\
&\quad \cdot P_r(x_k \mid d_k = i_k, S_k = m_k)\ P_r(y_k \mid d_k = i_k, S_k = m_k) \varsigma_k^{i_k} / 2^v \\
&= \frac{\varsigma_{k-1}^{i_{k-1}}}{2^v \sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(x_{k-1} - (2d_{k-1} - 1))^2\right) dx_{k-1} \cdot \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(y_{k-1} - (2Y_{k-1} - 1))^2\right) dy_{k-1} \\
&\quad \cdot \frac{\varsigma_k^{i_1}}{2^v \sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(x_k - (2d_1 - 1))^2\right) dx_1 \cdot \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2}(y_1 - (2Y_1 - 1))^2\right) dy_1 \\
&= K_k \exp\left(\frac{2}{\sigma^2}(x_{k-1} d_{k-1} + y_{k-1} Y_{k-1} + x_k d_k + y_k Y_k)\right)
\end{aligned}
\tag{2.11}
$$

MAP 복호기의 brach metric $\delta_k^{p,m}$ 은 식(2.11)과 같이 전개되며 i_k 는 k 시점에서의 정보비트를 의미하며, $p = i_{k-1} \| i_k$ 이다. 따라서, 정보비트 $i = \{0,1\}$ 에 대한 복호비트열 $p = \{00,01,10,11\}$ 을 얻게 된다. K_k 는 상수, Y_k 는 부호기의 state 와 입력비트 d_k 에 대한 함수이다. k 시점에서의 순방향 state metric α_k^m 은 다음과 같이 표현된다.

$$
\begin{aligned}
\alpha_k^m &= P_r(R_1^{k-2} \mid D_k = p, S_k = m_k,\ R_k^N) \ = P_r(R_1^{k-2} \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(D_{k-2} = p, S_{k-2} = m', R_1^{k-2} \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(D_{k-1} = p, S_{k-2} = m', (R_1^{k-4}, R_{k-2}) \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(R_1^{k-4} \mid D_{k-2} = p, S_{k-2} = m', R_{k-2}, S_k = m_k) \\
&\quad \cdot P_r(D_{k-2} = p, S_{k-2} = m', R_{k-2} \mid S_k = m_k) \\
&= \sum_{p=0}^{3} P_r(R_1^{k-4} \mid S_{k-2} = b(p, m_k)) P_r(D_{k-2} = p, S_{k-2} = b(p, m_k), R_{k-2}) \\
&= \sum_{p=0}^{3} \alpha_{k-2}^{b(p,m_k)} \delta_{k-2}^{p,b(p,m_k)}
\end{aligned}
\tag{2.12}
$$

$b(p, m_k)$ 는 시점 k 이고 m state 에서 입력이 p 일때 시점 $k-2$ 만큼 뒤쪽으로 향하는 state 의 번호를 뜻하며, 식(2.10)에 의해 다음 식(2.13)와 같이 표현된다.

$$
b(p, m_k) = m_{k-2}(d_{k-1}, d_k, m_k) \tag{2.13}
$$

유사한 방법으로 역방향 state metric β_k^m 역시

$$
\begin{aligned}
\beta_k^m &= P_r(R_k^N \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(D_k = p, S_{k+2} = m', R_k^N \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(D_k = p, S_{k+2} = m', R_{k+2}^N, R_k \mid S_k = m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} P_r(R_{k+2}^N \mid S_k = m_k, D_k = p, S_{k+2} = m', R_k) \\
&\quad \cdot P_r(d_k = j, S_{k+1} = m', R_k \mid S_k = m_k) \\
&= \sum_{p=0}^{3} P_r(R_{k+2}^N \mid S_{k+2} = f(p, m_k)) P_r(D_k = p, S_k = m_k, R_k) \\
&= \sum_{p=0}^{3} \beta_{k+2}^{f(p,m_k)} \delta_k^{p,m_k}
\end{aligned}
\tag{2.14}
$$

와 같이 나타낼 수 있으며, $f(p,m_k)$ 는 입력이 p 이고 state m_k 일때 $k+2$ 시점만큼 앞으로 향하는 state 의 번호를 뜻한다. 마찬가지로, 식(2.10)에 의해서 다음과 같이 나타낼 수 있다.

$$f(p,m_k) = m_{k+2}(d_{k+1}, d_{k+2}, m_k) \tag{2-15}$$

식(2.12)와 (2.14)에 의해 사후확률(APP)는 식(2.16)과 같다.

$$\begin{aligned}
\lambda_k^{p,m} &= P_r(D_k = p, S_k = m \mid R_1^N) \\
&= P_r(D_k = p, S_k = m,\ R_1^N) / P_r(R_1^N) \\
&= P_r(D_k = p, S_k = m, R_1^{k-2}, R_k^N) / P_r(R_1^N) \\
&= P_r(R_1^{k-2} \mid D_k = p, S_k = m,\ R_k^N) \cdot P_r(D_k = p, S_k = m,\ R_k^N) / P_r(R_1^N) \\
&= P_r(R_1^{k-2} \mid D_k = p, S_k = m,\ R_k^N) \cdot P_r(D_k = p, S_k = m,\ R_k, R_{k+2}^N) / P_r(R_1^N) \\
&= P_r(R_1^{k-2} \mid D_k = p, S_k = m,\ R_k^N) P_r(R_{k+2}^N \mid D_k = p, S_k = m,\ R_k) \\
&\quad \cdot P_r(D_k = p, S_k = m,\ R_k) / P_r(R_1^N) \\
&= \alpha_k^m \beta_{k+2}^{f(p,m)} \delta_k^{p,m}
\end{aligned} \tag{2.16}$$

시점 k 이후의 사건은 시점 k 에서 관찰하는 부분에 영향을 주지 않으므로 R_1^k 와 R_{k+2}^N 은 서로 독립적이다. 마지막으로 radix-4 방식의 LLR 은 다음과 같다.

$$LLR(D_k) = \max\left\{\sum_m \lambda_k^{oo}(m), \sum_m \lambda_k^{01}(m), \sum_m \lambda_k^{10}(m), \sum_m \lambda_k^{11}(m)\right\} \tag{2.17}$$

$D_k = \{00,01,10,11\}$ 을 나타내므로 만약, $LLR(D_k) = \lambda_k^{01}$ 이면 $d_{k-1} = 0, d_k = 1$ 로 복호하여, 한 시점에서 2 비트를 동시에 복호할 수 있다.

그림 2-3 에서 터보 iteration 시 extrinsic 정보 $Le(d_k^p), p = \{00,01,10,11\}$ 을 첫 번째 MAP 에서의 branch metric $\delta_k^{p,m}$ 은 식(2.18)과 같이 나타낼 수 있다.

$$\delta_k^{p,m} = K_k \exp\left(-\frac{2}{\sigma^2}(x_{k-1}d_{k-1} + y_{k-1}Y_{k-1} + x_k d_k + y_k Y_k)\right) \times L_e(d_k^p) \tag{2.18}$$

2.4 시뮬레이션 결과

그림 2-5 는 인터리버 크기를 Radix-2 에서는 200 으로, 그림 2-6 은 Radix-4 에서는 100 으로 하여 컴퓨터 시뮬레이션 결과이다. 시뮬레이션 결과 성능이 일치함을 알 수 있다.

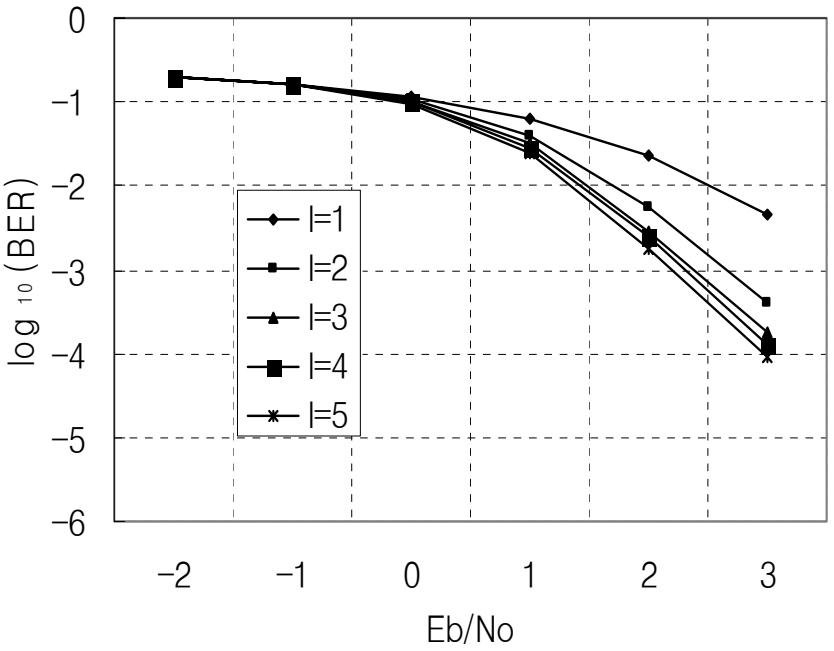


그림 2-5. Radix-2 방식(N=200)

Fig. 2-5. Performance of Radix-2 in case of N = 200.

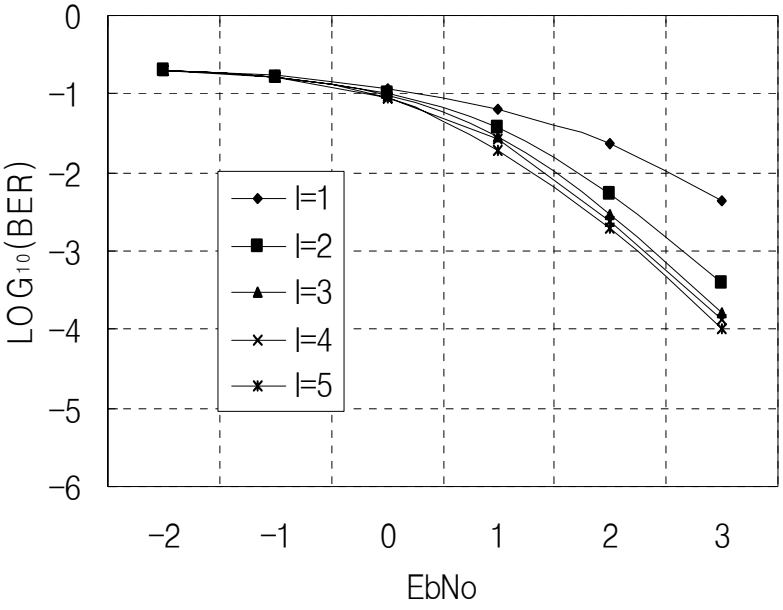


그림 2-6. Radix-4 방식(Ns=100)

Fig. 2-6. Performance of Radix-4 in case of Ns = 100.

제 3 장 Log-MAP 기반의 Radix-2 방식의 터보 복호기 설계

Turbo 복호기를 포함하는 수신 시스템의 개략적인 블록도를 그림 3-1 에 나타내었다. 본 논문에서는 디지털방식의 복조기를 고려한다.

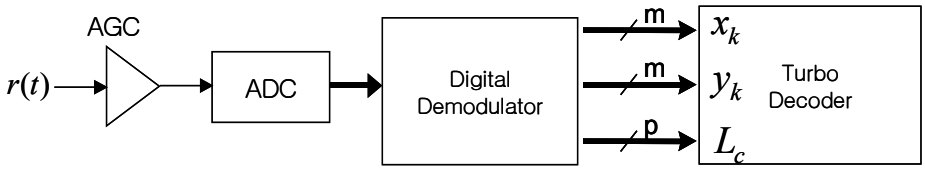


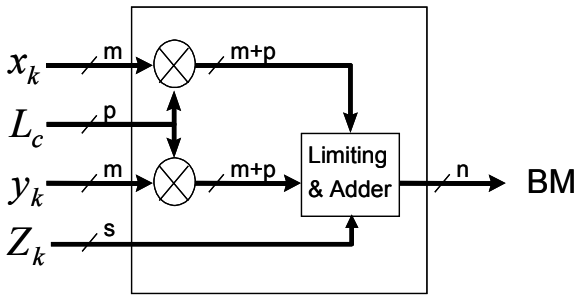
그림 3-1. Turbo 복호기 수신 시스템의 블록도

Fig. 3-1. A receiver that consists of digital demodulator and Turbo decoder.

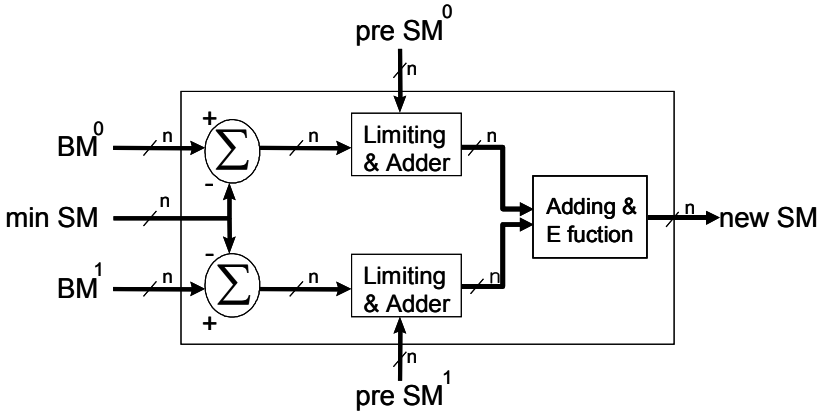
수신신호 $r(t)$ 는 AGC(Automatic Gain Control)를 거치고, 하향변환후 ADC(Analog to Digital Converter)에 의해 디지털 신호로 변환되거나 또는 subsampling 방식으로 직접 디지털 신호로 변환된다. Turbo 복호기에는 수신 신호에 대한 연판정값이 입력되어야 하므로 복조기는 복조를 수행하고난후 X_k, Y_k 에 대한 m 비트의 연판정 결과값 및 채널 신뢰도값 $L_c = 2/\sigma^2$ 에 대한 p 비트의 평가량을 출력한다. Turbo 복호에는 요구되는 연산량이 많고 특히, 메모리가 많이 요구되기 때문에 복호성능과 H/W 복잡성을 최적으로 절충하여 비트수 m 과 p 을 결정하여야 한다. Log-MAP 방식의 Turbo 복호에서는 X_k, Y_k 및 L_c 비트수에 의해서 다른 주요 구성요소들의 비트수가 결정되므로 사실은 복호성능과 H/W 복잡성이 m 과 p 의 값에 의해 전적으로 좌우된다[9].

3.1 Log-MAP 기반의 터보 복호기 모듈분석

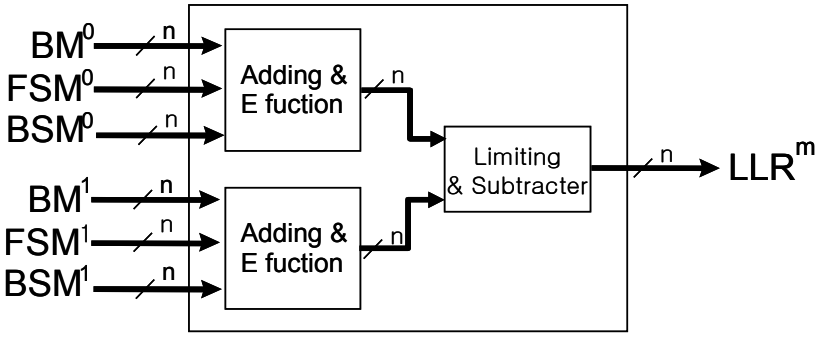
Log-MAP 기반의 Turbo 복호기의 주요 구성요소들은 가지메트릭을 계산하는 BMC (Branch Metric Calculator), 순방향 상태메트릭 FSM(Forward State Metric)과 역방향 상태메트릭 BSM(Backward State Metric)의 상태메트릭을 결정하는 SMC(State Metric Calculator), LLR 을 계산하는 LLRC (Log Likelihood Ratio Calculator)등을 들 수 있으며, 이들에 대해서는 그림 3-2 와 같이 구성할 수 있다.



(a) BMCU



(b) PMCU



(c) LLRC

그림 3-2. Log-MAP 기반 Turbo 복호기의 주요 블록에 대한 회로도

Fig. 3-2. Circuit diagram of a major constituting components of Log-MAP/Turbo decoder.

그림 3-2 의 (a) BMC 블록도에서 알 수 있듯이, 가지메트릭 BM 의 출력비트수 n 은 수신신호 X_k, Y_k 와 L_c 와의 곱셈연산후 m+p 로 증가하고 덧셈후 1 비트가 더 추가된다. 2 번째 iteration 부터는 첫번째 iteration 에서의 LLR 값 (즉, extrinsic 정보 Z_k) n 비트가 입력되어 더해지므로 limiting & adder 연산 (최소, 최대값이 -2^{n-1} 와 $2^{n-1}-1$ 이 되도록 제한하여 덧셈하는 연산)을 수행하지 않으면 iteration 수에 비례하여 n 값이 증가하게 된다. 따라서, limiting & adder 를 적용할 경우 BM 값의 표현 비트수 n 은 m 과 p 에 의해서만 결정된다. 그리고 FSM 과 BSM 을 구하는 그림 3-2 의 (b) SMC 에서는 먼저, 출력될 상태메트릭값이 발산하지 않도록하기 위해서 이전시점에서의 최소 상태메트릭값을 가지메트릭값에 빼준 다음, 이전시점의 상태메트릭값과의 합을 E 함수연산을 하여 새로운 상태메트릭값을 구한다. 이때, BMC 에서와 마찬가지로 상태메트릭값을 n 비트로 제한하기 위하여 내부의 덧셈연산시 limiting & adder 를 사용한다. 그림 3-2 의 (c) LLRC 에서는 입력비트 0 에 대한 각 메트릭과 1 에대한 각 메트릭값들을 더한 다음 E 함수연산을 행한후 뺄셈연산을 하여 한상태 m 에서의 LLR 값을 구한다. 역시, LLR 출

력 비트수를 n 비트로 제한하기 위하여 내부연산에서 각각 limiting adder 와 limiting subtracter 을 사용한다.

3.2 구현을 위한 각 메트릭의 최적 비트수 결정

앞절에서 설명한 바와 같이 터보 복호기 내부의 각 메트릭값 BM, SM(FSM,BSM), LLR 의 비트수 n 은 m 과 p 가 결정하게된다. 따라서 수신신호 X_k, Y_k 에 대한 적정 양자화 비트수의 결정과 채널 신뢰도 L_c 의 처리 과정은 log-MAP 방식의 터보복호기 하드웨어 구현에 있어서 매우 중요한 요소가 된다.

참고문헌 [10]에 의하면 수신신호의 양자화비트수는 6 비트로하고 L_c 는 2 비트로하는것이 최적의 할당비트인 것으로 설명하였다. 따라서, 각 메트릭 비트 n 은 9 비트가 된다. 이는 그림 3-3 의 컴퓨터 시뮬레이션을 통한 성능 분석으로도 확인할 수 있었다. 시뮬레이션 환경은 Log-MAP/Turbo 복호기를 구성하고 N=64 비트, iteration 횟수 I=3, 수신신호양자화 비트수 m 은 6 비트로, 그리고 L_c 는 2 비트로 하였다. 시뮬레이션결과 메트릭비트 n 을 8 비트에서 9 비트로 증가시켰을 경우 성능이 향상하였지만 9 비트 이상으로 하였을때는 성능의 향상이 없어서 n 을 9 비트로 하는 것이 최적임을 알 수 있었다. 또한, 같은 환경에서 수신신호의 양자화 비트수를 6 비트에서 8 비트로 확장하였을 경우에는 그림 3-4 와 같이 L_c 연산을 행하지 않은 경우가 오히려 성능이 우수한 것을 알 수 있었다. 이를 바탕으로한 Log-MAP/Turbo 복호기의 구현을 위한 최적 비트수 할당 내역을 표 3-1 에 정리하였다.

표 3-1. Log-MAP/Turbo 복호기의 각 메트릭의 최적 비트수 할당

Table 3-1. Optimum bit assignment of metrics at Log-MAP/Turbo decoder.

	일반적인 할당	최적 할당
L_c	2	제거
X_k, Y_k	6,6	8,8
BM	9	9
SM(FSM,BSM)	9	9
LLR	9	9
Z	9	9

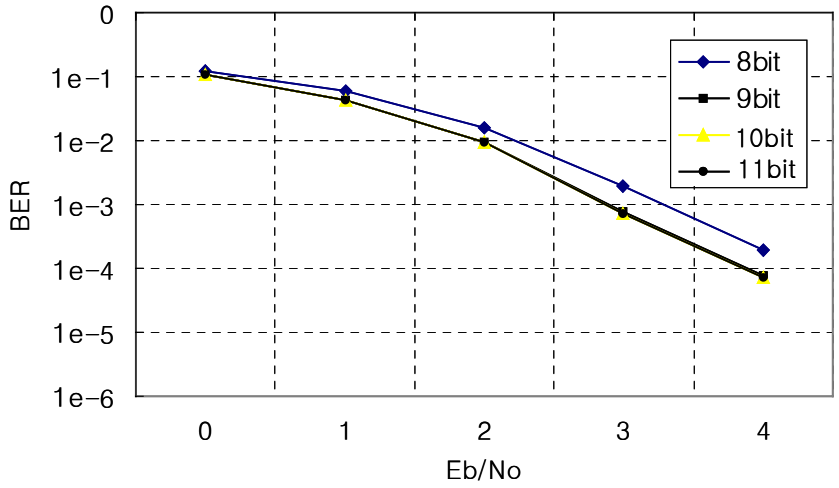


그림 3-3. 메트릭 비트 n 에 따른 Log-MAP/Turbo 복호기의 성능 (N=64, I=3)

Fig. 3-3. Performance of Log-MAP/Turbo decoder as a number of metric bit n(N=64, I=3)

3.3 Log-MAP/터보 복호기 CPLD 구현 및 타이밍 시뮬레이션

3.3.1 E-Table 의 설계

FSM, BSM, LLR 계산블록에서 E-함수 연산을 하게되는데, 하드웨어상으로는 계산값을 사전에 평가하여 이를 ROM 즉, LUT(Look-Up Table)에 저장하고 입력내용이 곧 바로 ROM 내의 주소를 지시하도록 설계한다. 본 논문에서 사용한 E-Table 은 입력 6 비트, 출력 4 비트로 하였다. 따라서 설계된 E-Table 의 크기는 2^6*4=256 비트 이다. E-Table 의 내용은 FLEX10K 디바이스 내부의 EAB(Embaded Array Block)의 ROM 영역에 할당하여 저장하였다.

3.3.2 Log-MAP 복호기에서의 메모리 할당

Log-MAP 복호기에 있어서 역방향 상태메트릭을 구하는 과정 때문에 수신신호 X_k, Y_k 를 복호블록 N 만큼 저장하여야 한다. 가지메트릭 BM 은 각 메트릭을 구할때마다 계산하지 않고, 저장된 수신신호를 읽여들여 N 블록만큼 모든 가지메트릭을 메모리에 저장하고 각 메트릭연산시 BM 값을 읽어들인다. 역방향 상태메트릭 BSM 은 구하는 순서는 역방향이지만 LLR 연산시 순방향으로 출력값이 입력되어야 하므로 역시 역방향으로 구한 BSM 을 메모리에 저장시켜야 한다. 이와 같이 Log-MAP 복호기의 설계에 요구되는 모든 메모리는 FLEX10K-100 디바이스의 내부램인 EAB 를 사용하였다. 그러나, 본 논문에서 사용한 FLEX10K-100GC503-4 디바이스에서 지원하는 EAB 의 한계용량 때문에 복호블록 N=64 비트로 제한하였다. 이때, 사용되는 메모리 용량을 계산하면 먼저 수신신호를 저장하는 메모리용량은 8 비트 양자화신호이므로 2*8*64=1024 비트, BM 메모리는 한 시점에서 존재하는 가지메트릭 BM00, BM01, BM10, BM11 을 복호블록 만큼 저장해야 하므로 요구되는 메모리는 4*9*64=2304 비트가 요구된다. 여기서, BMxx 는 가지부호어 xx 에 대한 가지메트릭이다. BSM 메모리 역시 한 시점에 있어서 4 개의 각 상태에 해당하는 메트릭 BSM0, BSM1, BSM2, BSM3 를 복호블록 만큼 저장해야 하므로 요구되는 메모리는 4*9*64=2304 비트 이다. 터보 복

호에 사용되는 인터리버는 랜덤 인터리버를 사용하였다. 인터리버의 주소 정보는 컴퓨터 시뮬레이션을 통해서 얻은 정보를 입력하였으며, 인터리버의 구현시 사용되는 메모리 역시 EAB 를 사용하였다. 디인터리버의 경우도 마찬가지 이며 복호블럭 N 의 크기와 인터리버의 크기는 일치하므로 인터리버와 디인터리버의 구현시 요구되는 메모리의 크기는 각각 9*64=576 비트가 된다. 따라서 Log-MAP/터보 복호기의 설계시 요구되는 총 메모리는 표 3-2 와 같다.

표 3-2. Log-MAP/터보 복호기의 메모리 할당

Table 3-2. Memory assignment of Log-MAP/Turbo decoder.

메모리 내용	메모리 크기
X_k, Y_k	2*8*63=1024 Bit
BM 메모리	4*9*64=2304Bit
BSM 메모리	4*9*64=2304Bit
E-Table	2^6*4=256 Bit
인터리버	9*64=576 Bit
디인터리버	9*64=576 Bit

3.3.3 Log-MAP 복호기의 VHDL 설계

Log-MAP 복호기를 전체적으로 구성한 것을 그림 3-4 에 나타내었다. 먼저, 수신신호를 m 비트 양자화한 신호를 각각 I_ch, Q_ch 에 입력하고 BMU(Branch Metric Unit)에서 순방향으로 각 수신신호에 대한 가지메트릭 BM00,BM01,BM10,BM11 을 구함과 동시에 이를 BM 메모리에 저장한다. 그 다음, 역방향으로 BM 메모리에 저장된 가지메트릭값을 불러들여 역방향 상태메트릭 BSM 을 구하고 마찬가지로 BSM 메모리에 저장한다. 다시, 순방향으로 BM 메모리로 부터 가지메트릭을 불러들여 순방향 상태메트릭 FSM 을 구하고 이 값과 BSM 메모리와 BM 메모리로 부터 불러들인 각 메트릭

값을 동시에 LLRU(Log Likelihood Ratio Unit)에 입력하여 LLR 을 구한다.

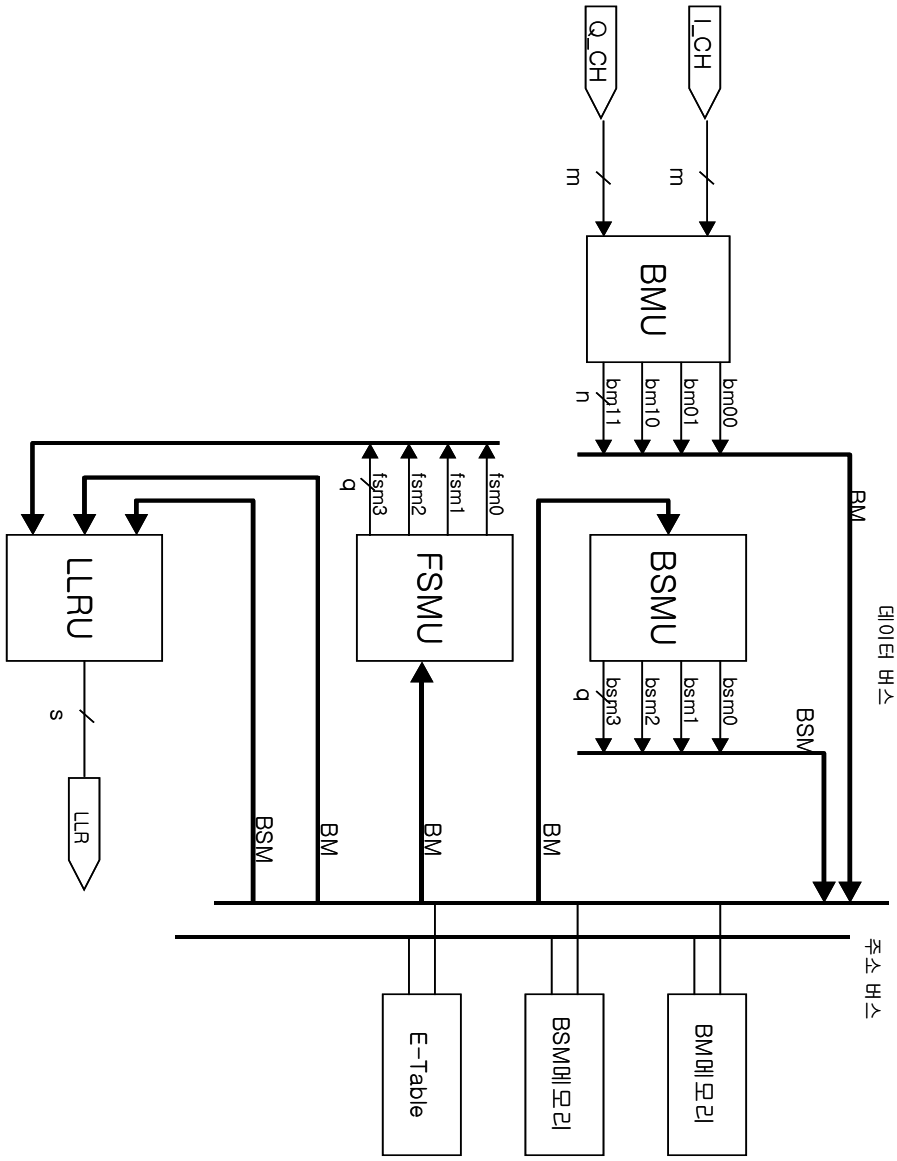


그림 3-4. Log-MAP 복호기의 하드웨어 구조

Fig. 3-4. Circuit of Log-MAP decoder.

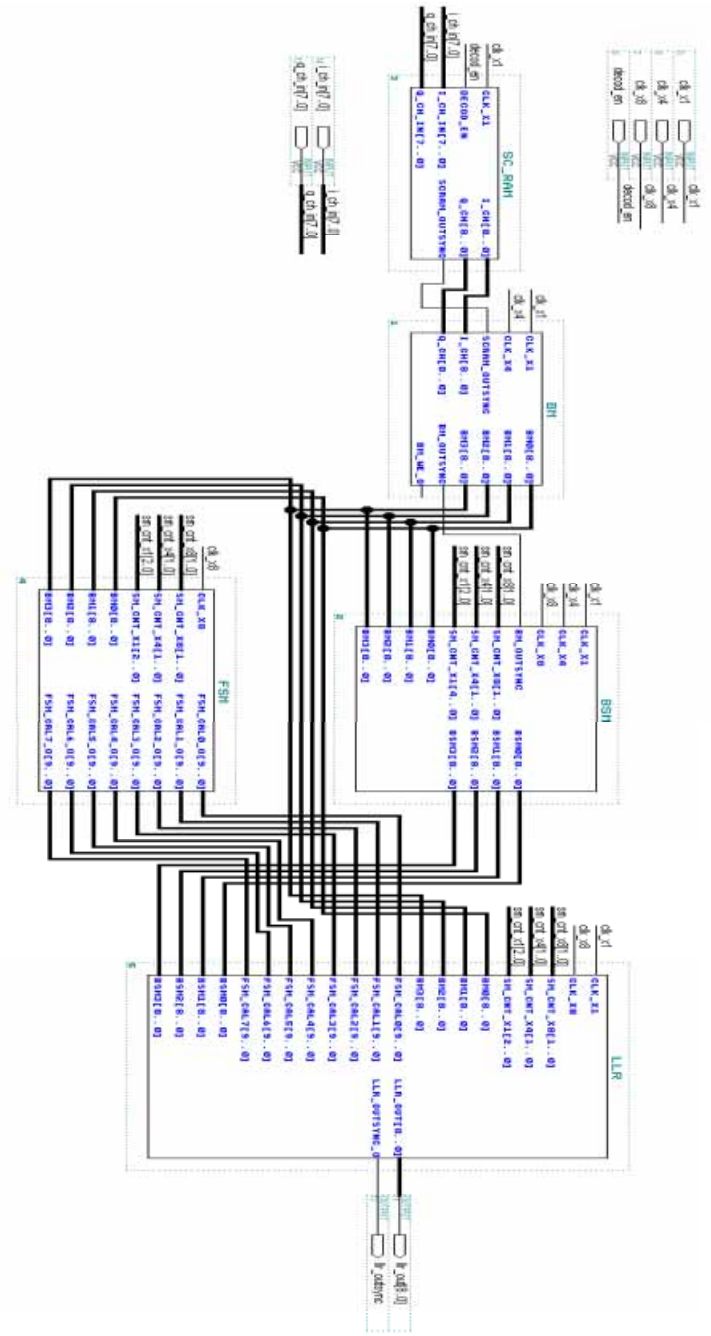
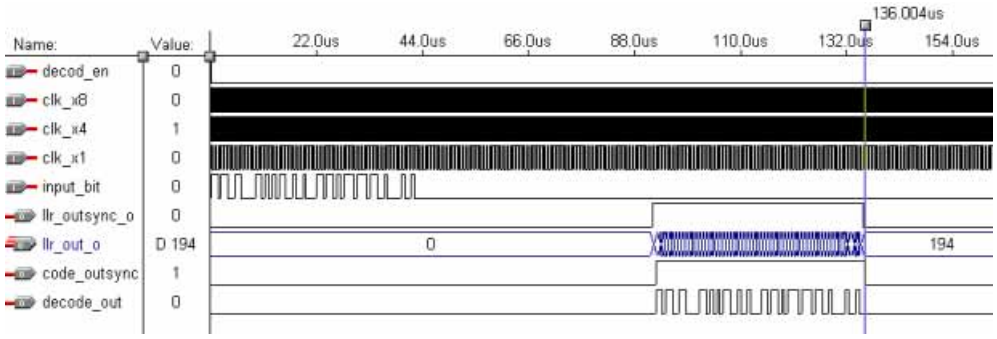
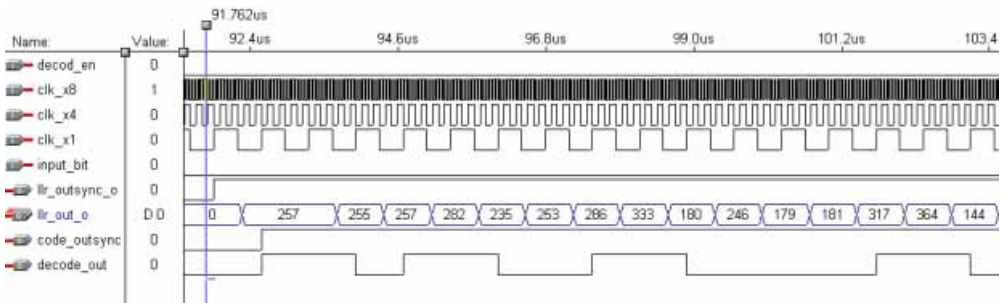


그림 3-5. VHDL 로 작성한 N=64 인 Log-MAP 복호기의 회로도

Fig. 3-5. Schematic diagram of Log-MAP decoder with N=64.



(a) Time Sequence 1



(b) Time Sequence 2

그림 3-6. Log-MAP 복호기의 타이밍 시뮬레이션 결과

Fig. 3-6. Timing simulation output of Log-MAP decoder.

VHDL 로 작성한 Log-MAP 복호기의 Top 레벨 회로도는 그림 3-5 과 같다. 그리고 이를 실행한 타이밍 시뮬레이션결과를 그림 3-6 에 나타내었다. 사용된 디바이스는 ALTERA 사의 FLEX10K100GC503-4 이며, 내부메모리 EAB 의 제약 때문에 복호블록 N=64 비트로 설정하였으며, 1 배수, 4 배수, 16 배수의 총 3 개의 클럭이 사용되었다. 1 배수 클럭은 수신신호를 읽어들이고 복호비트를 결정하는데 사용되었고 4 배수 클럭은 BM 메모리와 BSM 메모리에 각 4 개의 메트릭값(BM00, BM01, BM10, BM11 과 BSM0, BSM1, BSM2, BSM3)을 저장하기위해 사용되었으며 16 배수 클럭은 BSM, FSM, LLR 을 연산하기 위한 많은 작업 때문에 사용되었다.

타이밍 분석결과 16 배수 클럭의 요구 주기는 약 44ns 이었고, 64 비트를 복호하는데 소요된 총 클럭지연은 136us= 0.00735 Mbps 였다. 따라서, 복호기의 복호속도는 0.00735Mbps * 64 = 0.47 Mbps 이다.

3.3.4 Log-MAP/터보 복호기의 VHDL 설계

터보 복호시 필요한 2 개의 Log-MAP 복호기는 그 동작이 직렬동작 즉, 앞쪽의 복호기의 출력이 끝난 다음 그 출력을 입력받아 뒤쪽복호기의 복호동작이 시작되므로, 본 설계에서는 Log-MAP 복호기를 2 개를 사용하지 않고 하나의 Log-MAP 복호기를 “serial_sw” 라는 모듈에서 스위칭동작을 하여 동일한 MAP 복호기로 각각 다른 복호동작이 가능하도록 설계하였다.

이러한 Log-MAP/Turbo 복호기를 VHDL 로 설계한 Top 레벨 회로도를 그림 3-7 에 나타내었다. 여기서, "SC_RAM"모듈에 컴퓨터 시뮬레이션을 통하여 얻어진 수신신호를 저장시켰다. 그리고, "INTLV"는 인터리버 "DE_INTLV"는 디인터리버이다.

타이밍 시뮬레이션 결과를 그림 3-8 에 나타내었다. 복호블록 N=64 비트이고 타이밍 분석결과 제일 빠른 클럭인 16 배수 클럭의 요구 주기는 Log-MAP 복호기에서와 마찬가지로 약 44ns 이었다. 그림 3-8 의 (a) 에서와 같이 1 번 복호동작시 지연이 약 280us 였으며, 이는 터보복호기가 64 비트를 복호하는데 소요된 총 클럭지연이 280us= 0.00357 Mbps 이라는 의미이다. 따라서, 복호기의 복호속도는 0.00357Mbps * 64 = 0.23 Mbps 이다. 그리고, Iteration 이 증가 할수록 이러한 복호속도는 복호지연만큼 줄어들게 되는데, 예를 들어, Iteration=3 일 경우 총 복호동작은 3 번이며 복호속도는 0.23 / 3 = 0.076 Mbps 가 된다.

구현한 복호기의 복호동작을 알아보기 위하여 N=20 으로 하고 Iteration 에 따른 복호기의 출력을 그림 3-9 에 나타내었다. 분석 결과 첫번째 Iteration 시 5 개의 오류를 그리고, 두번째는 3 개의 오류를 가지는 출력을 하였으나 3 번째 Iteration 부터는 모든 오류를 정정한 출력을 하고 있는 것을 알수 있다.

사용된 FLEX10K100GC503-4 디바이스의 report 파일을 그림 3-10 에 나타내었다. 총 10 만 게이트중에서 62%를 사용하였다.

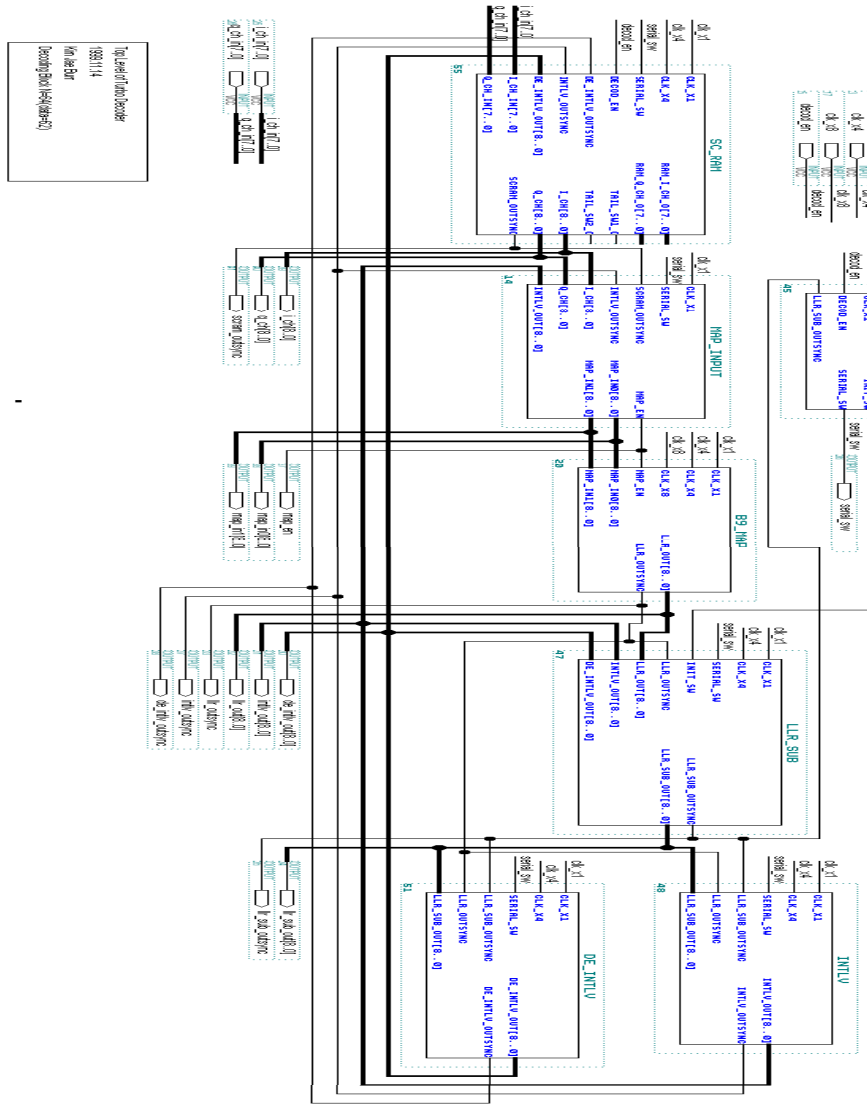
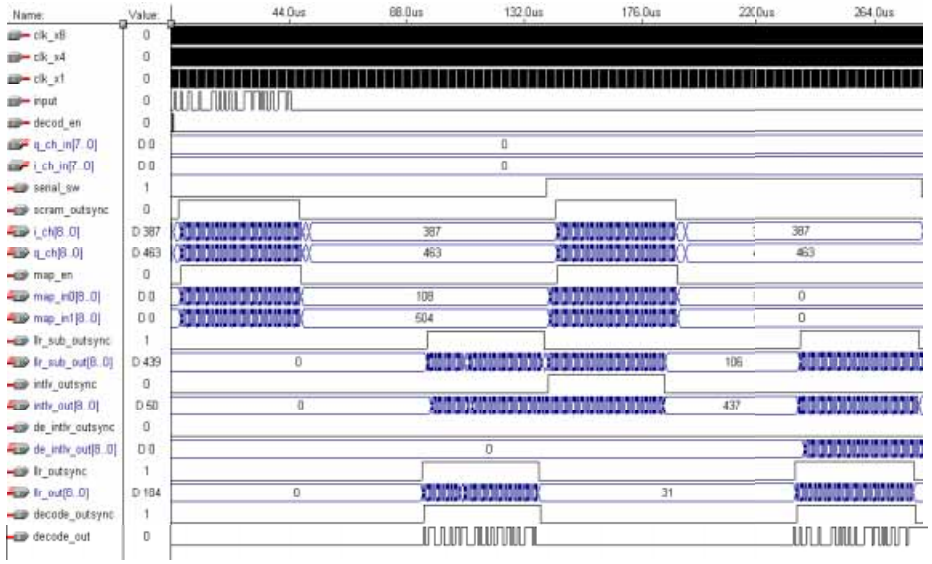
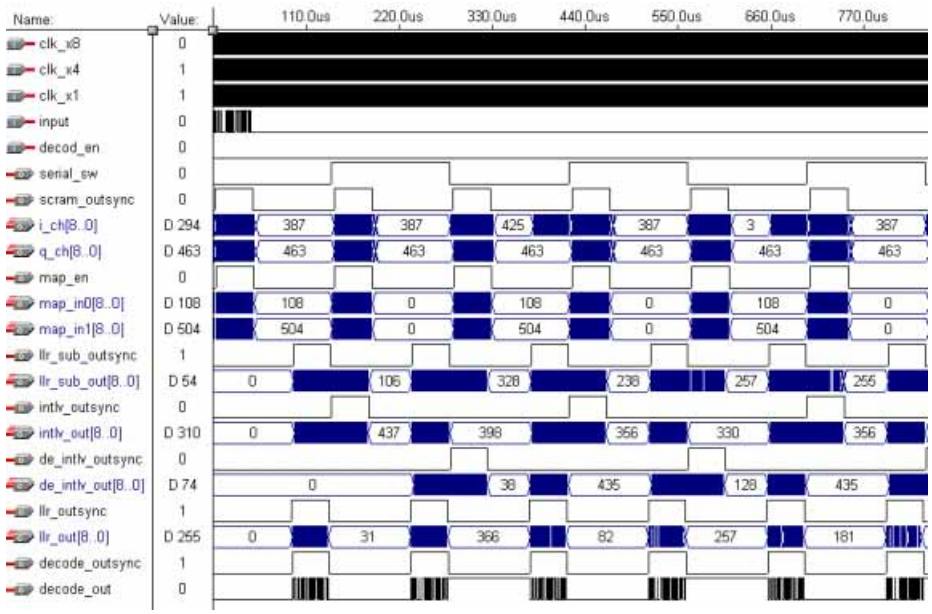


그림 3-7. N=64 인 Log-MAP/Turbo 복호기의 회로도

Fig. 3-7. Schamatic diagram of Log-MAP/Turbo decoder with N=64.



(a) Iteration = 1



(b) Iteration = 3

그림 3-8. Log-MAP/Turbo 복호기의 타이밍 다이어그램

Fig. 3-8. Timing simulation output of Log-MAP/Turbo decoder.

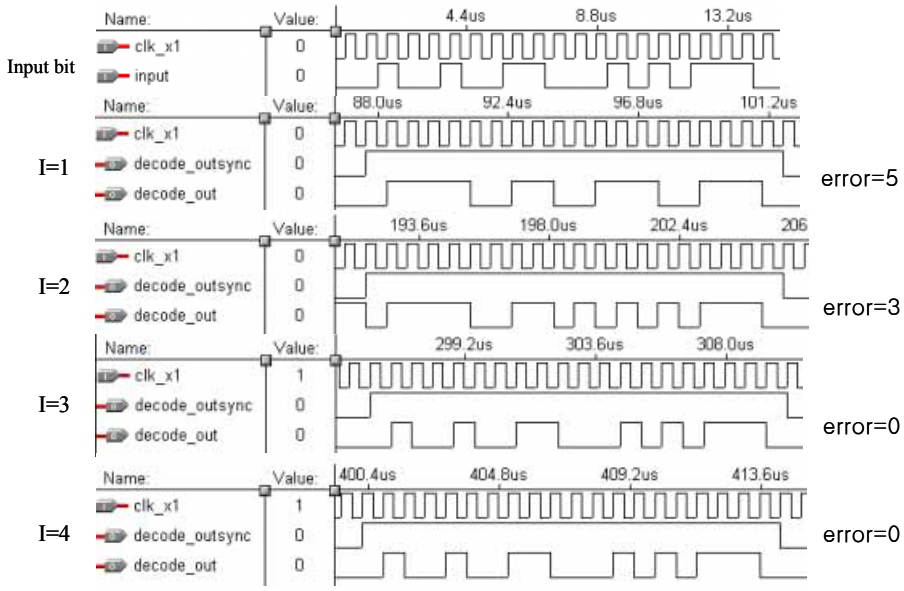


그림 3-9. Iteration 에 따른 복호과정 (Eb/No=1[dB])

Fig. 3-9. Decoding process of Log-MAP/Turbo decoder according to iteration (Eb/No=1[dB]).

```
** DEVICE SUMMARY **

Chip/                          Input Output Bidir  Memory  Memory            LCs
POF       Device               Pins  Pins   Pins   Bits % Utilized  LCs  % Utilized

tb_code   EPF10K100GC503-4 20    79     0    13312     54 %    3125     62 %

User Pins:                  20    79     0
```

그림 3-10. Turbo 복호기에 사용된 디바이스의 report file

Fig. 3-10. Device summary of Log-MAP/Turbo decoder.

사용된 디바이스는 ALTERA 사의 FLEX10K100GC503-4 이며, 내부메모리 EAB 의 제약 때문에 복호블록 N=64 비트로 설정하였으며, 1 배수, 4 배수, 16 배수의 총 3 개의 클럭이 사용되었다.

제 4 장 Log-MAP 기반의 Radix-4 방식의 터보 복호기 설계

4.1 구현을 위한 구조 설계

본 논문에서 Radix-4 방식의 터보 복호기를 구현할 경우, 구현을 위한 구조 설계도는 그림 4-1 과 같다.

그림 4-1 에서 알 수 있듯이, N 개의 I_ch, Q_ch 수신신호는 각각 짝수 번째와 홀수 번째로 분리하여 수신신호 메모리에서 N/2 개의 4 개의 신호를 출력한다. 수신신호 메모리에서 출력될 때 sync 제어신호를 생성하여 BMC 로 입력한다.

BMC 에서는 수신신호의 메모리에서 출력된 sync 신호를 입력 받아 BM0000 부터 BM1111 까지의 Branch Metric 을 구하고 이를 BM 메모리에 저장한다. 저장이 완료되면 각 상태에서의 BSM 을 구하기 위해 BM 메모리의 역순으로 읽으면서 BSM 을 계산하다. 계산된 BSM 은 BSM 메모리에 저장되며, 저장이 완료되면 BSM_sync 를 생성하여 BM 메모리로 입력한다.

BM 메모리에서는 BSM_sync 신호가 입력되며 각 상태에서의 FSM 을 구하기 위해 BM 메모리의 BM0000 부터 BM1111 을 순방향으로 읽으면서 FSM 을 구한다. FSM 을 구하는 동시에 BSM 메모리에 저장된 정보를 역순으로 읽으면서 복호하고 타이밍 분석도는 그림 4-2 과 같다. BM, 계산된 BSM, FSM 을 이용하여, LLR 을 구한다. LLR 을 구할 때 BSM, FSM, BM 의 정확한 동기를 맞추기 위해, 그리고 FSM 이 출력되는 순간 LLR 을 구하기 위해 FSM_sync 신호를 LLRC 에 보낸다.

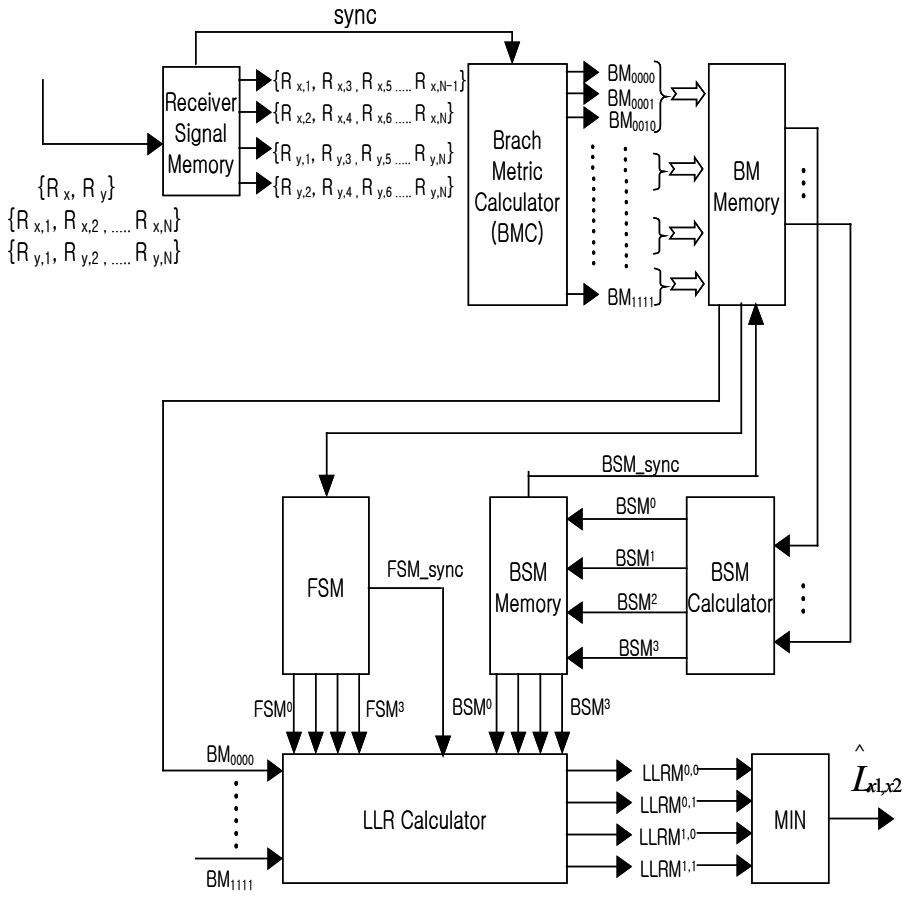


그림 4-1. Radix-4 터보 MAP 회로도

Fig.4-1. The designed schematic for the proposed Radix-4 turbo MAP decoder.

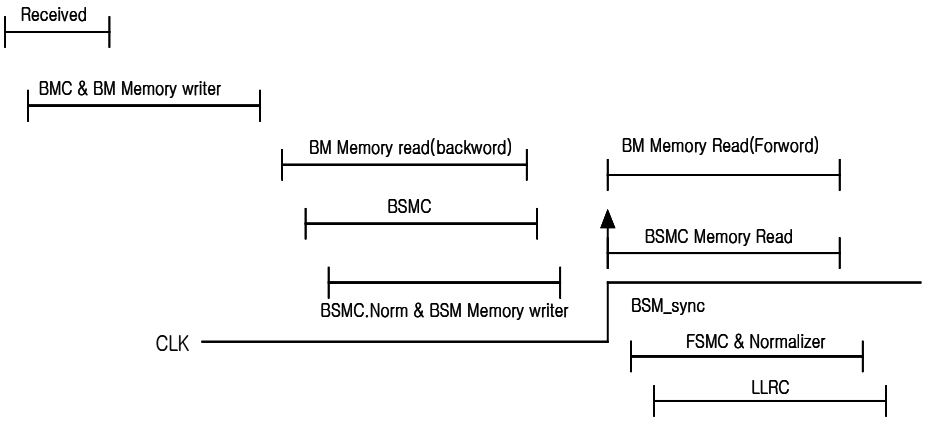


그림 4-2. 타이밍 분석도

Fig. 4-2. Analysis of timing.

4.2 최적의 비트수 결정

터보 복호기 내부의 각 메트릭값 BM, SM(FSM,BSM), LLR 의 비트수는 수신신호 Xk, Yk 에 대한 적정 양자화 비트수의 결정과 채널신뢰도 Lc 의 비트수에 결정이 된다. 여기에서는 Lc 연산은 제거 하여 8Bit 로 할당하였다.

먼저 수신신호의 비트수를 결정하기 위해 100000 개의 데이터를 입력하고 각 수신신호의 양자화 비트 수에 따른 성능을 시뮬레이션하였다. 그림 4-3 에서 보는 바와 같이 8 비트 이상에서는 성능의 차이가 거의 없으므로 수신신호의 최적의 양자화 비트 수는 8 비트임을 알 수 있다.

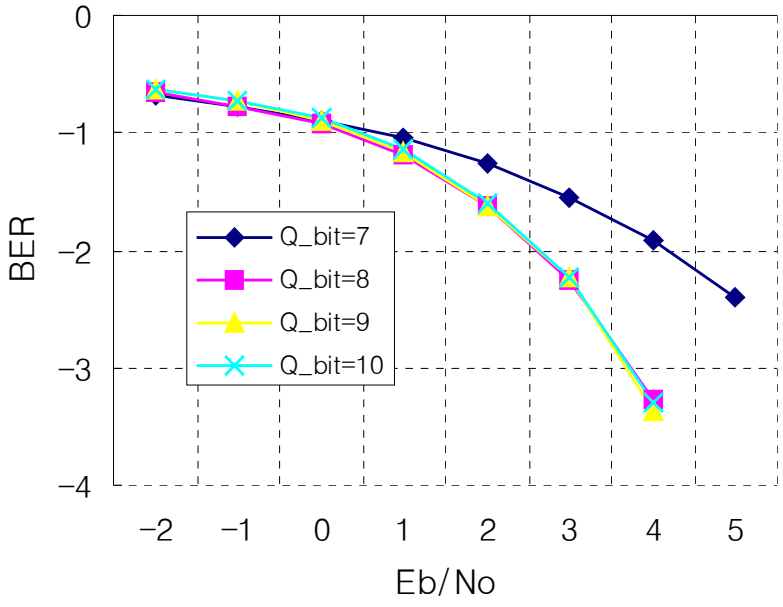
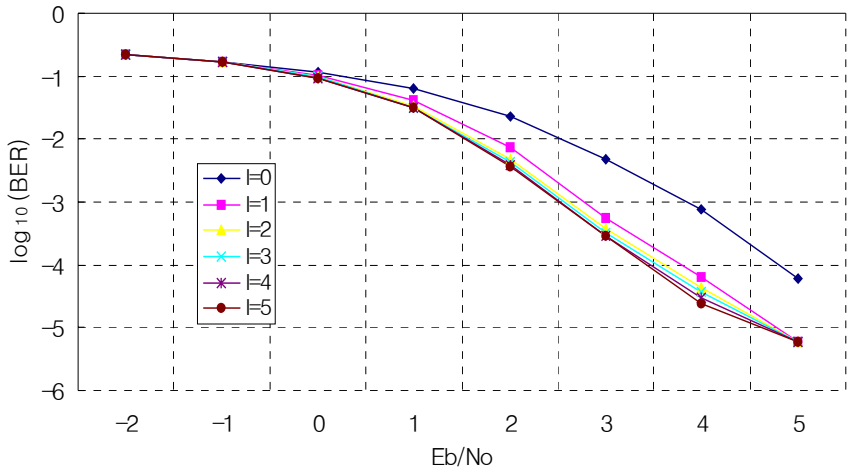


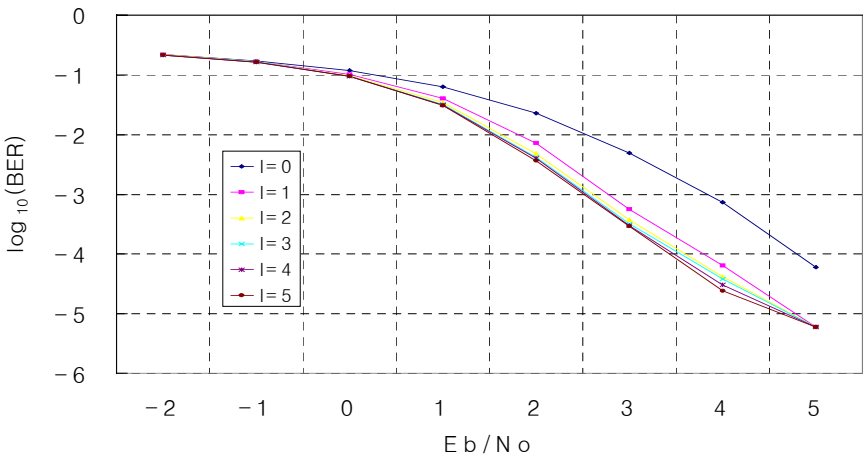
그림 4-3. 수신신호 비트 수에 따른 Radix-4 Log-MAP/터보 복호기 성능 (N=100, I=1)

Fig. 4-3. Performance of Radix-4 Log-MAP/Turbo decoder as a number of recivered signal bit(N=100, I=3)

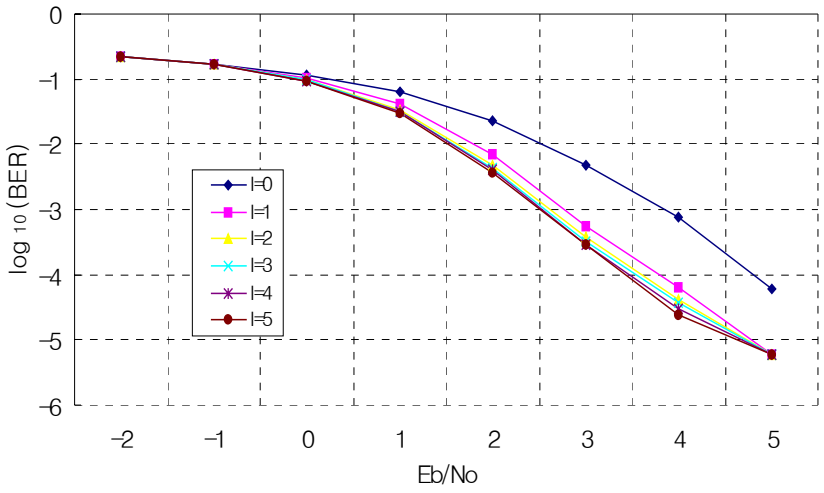
다음은 각각의 메트릭 비트에 따른 성능을 평가하기 위하여 시뮬레이션을 하였다. 시뮬레이션 환경은 인터리버 사이즈 N=100, 데이터 500000 개, 수신신호의 양자화 비트수는 8 비트로 고정을 하고 각 메트릭 비트수를 변화시켰을때 성능분석은 그림 4-4 와 같다. 이를 바탕으로 Radix-4 Log-Map/터보 복호기의 구현을 위한 최적 비트수 할당 내역을 표 4-1 에 정리 하였으며 기존의 Radix-2 방식과 비교하여 볼 때, bit 수의 할당은 동일함을 알 수 있다.



(a) BM=9bit SM=9bit LLR=9bit



(b) BM=9bit SM=10bit LLR=11bit



(c) BM=9bit SM=10bit LLR=10bit

그림 4-4. 메트릭비트 n 에 따른 Radix-4 Log-MAP/터보 복호기성능

Fig. 4-4. Performance of Radix-4 Log-MAP/Turbo decoder as a number of metric bit (N=64, I=3).

표 4-1. Radix-4 Log-MAP/터보 복호기의 각 메트릭의 최적 비트수 할당

Table 4-1. Optimum bit assignment of metrics at Radix-4 Log-MAP decoder.

	최적 할당
X_k, Y_k	8,8
BM	9
SM(FSM,BSM)	9
LLR	9
Z	9

4.3 Radix-4 방식의 Log-MAP 복호기 VHDL 설계 및 타이밍 시뮬레이션

4.3.1 E-Table 의 설계

앞 장에서 설명한 바와 같이 동일하다.

4.3.2 Log-MAP 복호기에서의 메모리 할당

Log-MAP 복호기에 있어서 역방향 상태메트릭을 구하는 과정 때문에 수신신호 X_k, Y_k 를 복호블록 N 만큼 저장하여야 한다. 여기서 N 은 Radix-2 의 경우는 앞서 설명한 바와 같이 64 이고, Radix-4 의 경우는 절반인 32 이다. 사용되는 메모리 용량을 계산하면 먼저 수신신호를 저장하는 메모리용량은 8 비트 양자화신호이므로 4*8*32=1024 비트, BM 메모리는 한 시점에서 존재하는 가지메트릭 BM0000, BM0001, BM0010, BM0011, BM0100, BM0101, BM0110, BM0111, BM1000, BM1001, BM1010, BM1011, BM1100, BM1101, BM1110, BM1111 을 복호블록 만큼 저장해야 하므로 요구되는 메모리는 16*9*32=4608 비트가 요구된다. 여기서, BMxxxx 는 가지부호어 xxxx 에 대한 가지메트릭이다. BSM 메모리 역시 한 시점에 있어서 4 개의 각 상태에 해당하는 메트릭 BSM0, BSM1, BSM2, BSM3 를 복호블록 만큼 저장해야 하므로 요구되는 메모리는 4*9*32=1152 비트 이다. 따라서 Log-MAP 복호기의 설계시 요구되는 총 메모리는 표 4-2 와 같다.

표 4-2. Radix-4 Log-MAP 복호기의 메모리 할당

Table 4-2. Memory assignment of Radix-4 Log-MAP decoder.

메모리 내용	메모리 크기
X_k, Y_k	2*8*63=1024 Bit
BM 메모리	16*9*32=4608Bit
BSM 메모리	4*9*32=1152Bit
E-Table	2^6*4=256 Bit

4.3.3 VHDL 설계 결과 검토

본 절에서는 앞절에서 제시한 구조 및 bit 수 할당을 기초로 VHDL 시뮬레이션을 하였다.

시뮬레이션 환경은 Radix-2 방식과 비교하기 위해 Chip 을 동일하게 설정하였다. 최소 클럭은 약 58ns 였으며, 기존의 방식이 44ns 에 비해 약 12ns 가 더 많이 요구되어 짐을 알 수 있다.

이유는 첫째로 기존의 방식은 BM 을 구할 때 1 개의 가산기만 요구되나, Radix-4 방식은 4 개의 수신신호를 더해야 하므로, 3 개의 가산기가 필요하며 연산 결과 다음 클럭에 BSM 및 FSM, LLR 계산을 하기 위해 1/2 클럭만에 출력해야 하므로 요구되는 클럭이 증가하였다.

둘째로, BSM, FSM, LLR 구할 시 E 함수가 기존의 방식은 2 개가 필요하나 제안된 Radix-4 방식에서는 3 개가 요구되므로 클럭이 증가하였다. 그림 4-5 에서 그림 4-8 까지는 BM, BSM, FSM, LLR 의 VHDL 시뮬레이션 결과를 나타낸 것이다.

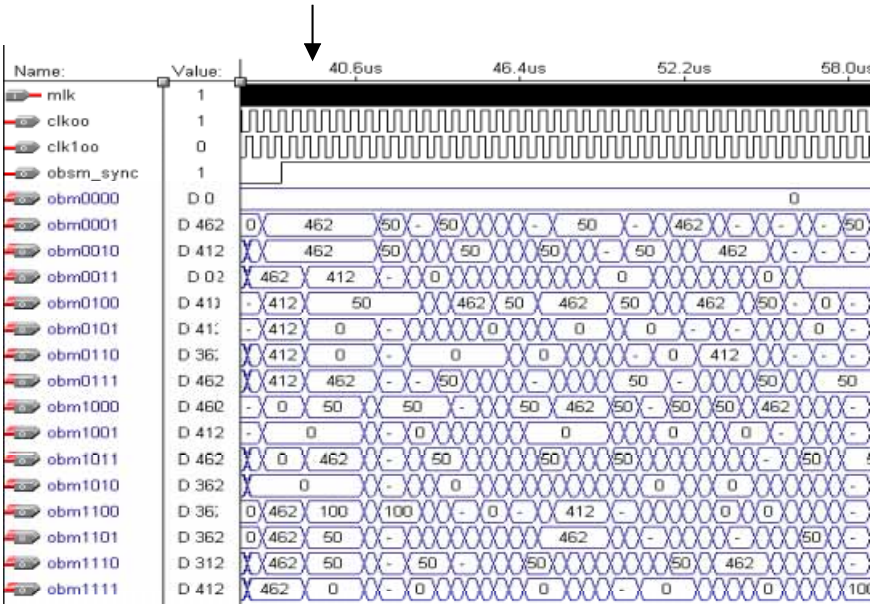


그림 4-5. Radix-4 Log-MAP/Turbo 복호기의 BM VHDL 결과

Fig. 4-5. BM VHDL simulation result of Radix-4 Log-MAP/Turbo decoder.

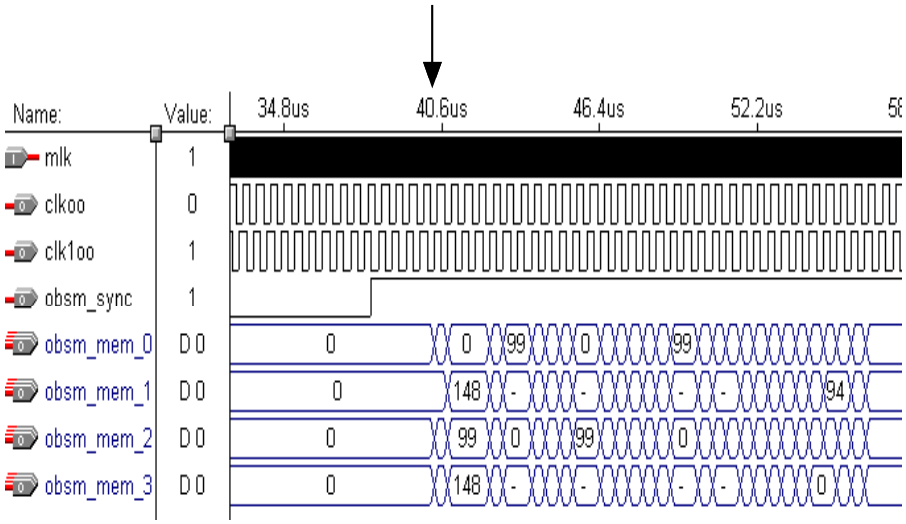


그림 4-6. Radix-4 Log-MAP/Turbo 복호기의 BSM VHDL 결과

Fig. 4-6. BSM VHDL simulation result of Radix-4 Log-MAP/Turbo decoder.

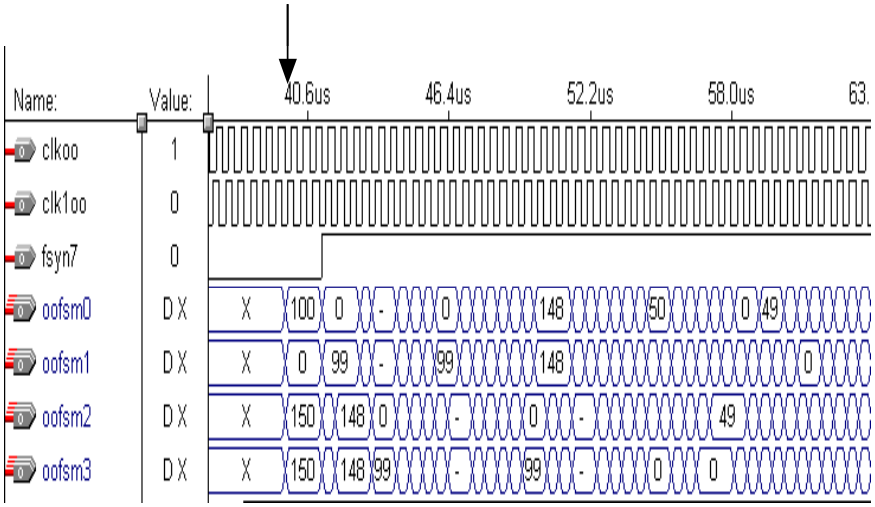


그림 4-7. Radix-4 Log-MAP/Turbo 복호기의 FSM VHDL 결과

Fig. 4-7. FSM VHDL simulation result of Radix-4 Log-MAP/Turbo decoder.

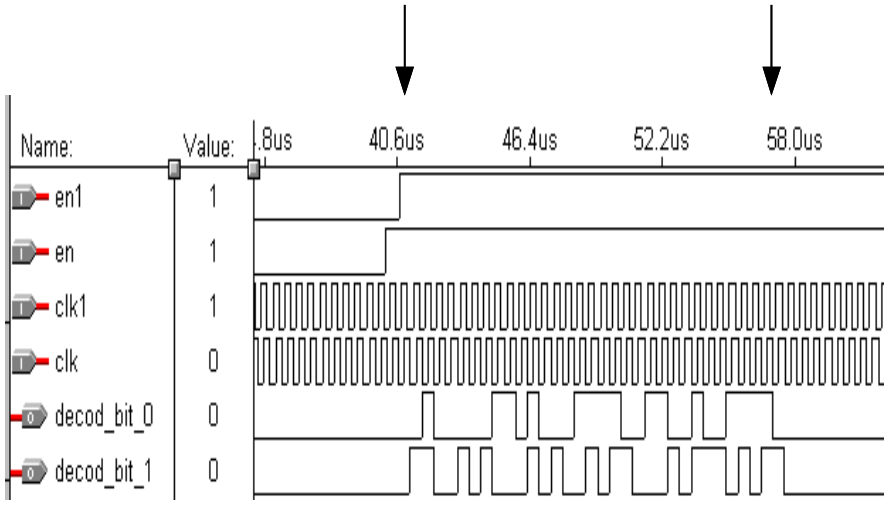


그림 4-8. Radix-4 Log-MAP/Turbo 복호기의 LLR VHDL 결과

Fig. 4-8. LLR VHDL simulation result of Radix-4 Log-MAP/Turbo decoder.

4.4 Radix-2 방식과 Radix-4 방식의 결과 및 검토

본 논문의 처리 속도 및 지연에 대한 구현 결과 비교는 표 4-3 과 같다. 표 4-3 의 세부항목들은 그림 4-5 에서 그림 4-8 의 화살표 표시된 부분의 시점이다.

표 4-3. Radix-2 방식과 Radix-4 방식의 처리 속도 비교
Table. 4-3. Comparison of processing speed between Radix-2 method and Radix-4 method.

방식 / 항목	Radix-2	Radix-4
Main clock	44ns	58ns
BM delay (FSM 을 구하기 위한 BM 의 순방향 read 초기 time)	66us	38us
FSm delay (FSM 계산 초기 time)	67.6us	40us
BSM delay (LLR 을 구하기 위한 BSM 역방향 read 초기 time)	67us	39us
LLR (LLR 출력 end time)	136us	57us
복호속도	0.47[Mbps]	1.12[Mbps]

표 4-3 에서 알 수 있듯이 전체 지연 시간은 LLR 의 출력이 끝나는 시간이므로 Radix-2 방식에 비해 약 2.4 배 정도 지연을 감소 시켰으며, 복호 속도는 Radix-2 방식에 비해 약 2.4 배 향상시켰음을 알 수 있다.

제 5 장 결 론

현재의 터보코드는 기존의 Viterbi 복호기와는 달리 처리할 데이터양과 연산작용이 매우 복잡해 처리속도가 저속이어서 음성, 데이터 등의 저속 서비스에만 적용되고 있으며, 고속데이타, 동영상 등의 고속 서비스에의 적용은 불가능한 실정이다. 터보 복호기를 고속화를 할 수 있는 방법은 복호기 구조를 한 클럭에 한 비트를 복호하는 기존의 Radix-2 방식이 아니라 한 클럭에 2 비트를 복호하는 Radix-4 방식의 적용이 필수적이다. 따라서 본 논문에서는 Radix-4 복호 방식을 이용하여 기존의 터보코드 복호속도를 2 배 이상 고속화시켜 MAP 기반의 터보 복호기를 FPGA(Field Programmable Gate Array)화 하는 것을 목표로 하고 있다.

이를 위하여 Radix-4 방식을 적용한 Log-MAP 기반의 터보 복호 알고리즘을 제안 하고, 제안한 알고리즘을 토대로 구현을 위한 bit 수를 설정하여 VHDL 시뮬레이션 하여, FPGA test board 에 구현하기 위한 타이밍 시뮬레이션을 하였다. 이 방식이 지연 및 속도 면에서 효율적임을 증명하기 위하여 기존의 Radix-2 방식의 Log-MAP 기반의 터보 복호기를 타이밍 시뮬레이션하였다.

시뮬레이션 결과, 제안한 방식과 기존 방식을 성능 면에서 에러정정능력이 일치하며, 구현 결과 복호 속도는 기존의 방식에 비해 약 2.4 배 고속화됨을 확인하였다. 따라서 제안한 방식은 차세대 이동통신인 IMT-2000 등과 같은 단말기의 실시간 복호기에 적용이 가능하리라 기대된다.

참고문헌

[1] G. D. Forney, Jr., *Concatenated Codes*, Cambridge, MA: MIT. Press, 1996.

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

[3] L. Papke, P. Robertson, and E. Villebrun, "Improved decoding with the SOVA in a parallel concatenated (turbo-code) scheme," in *Proc., IEEE Int. Conf. on Commun*., 1996.

[4] L. Lin and R. S. Cheng, "Improvements in SOVA- based decoding for turbo codes," in *Proc., IEEE Int. Conf. on Commun*., 1997.

[5] P. Robertson, E. Villebrun, and P. Hoeher, “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain,” in *Proc. ICC’95*, pp.1009-1013, Jun. 1996.

[6] D. Divsalar and F. Pollara, “Serial and Hybrid Concatenated Codes with Applications,” *Proceedings of the International Symposium on Turbo Codes & Related Topics*, pp. 80-87, Sep. 1997.

[7] S. S. Pietrobon, "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder," in *Proc., IEEE Int. Symp. on Inform. Theory*, pp. 471-489, 1995.

[8] 정지원, 성진숙, 김명섭, 오덕길, “Radix-4 방식의 터보 MAP 복호 알고리즘”, 25 권 4A 호 한국통신학회, 2000. 4.

[9] 손만익, 고성찬, “Log-MAP 기반 터보복호기의 VHDL 설계”, 한국통신학회 하계학술대회 논문집(上), pp. 799-803, 1999.

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

감사의 글

대학원 2 년 이라는 시간 동안 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 지금은 캐나다에서 열심히 연구를 하고 계신 정지원 교수님께 깊은 감사를 드립니다. 연구과정에서의 힘겨웠던 순간들과 연구결과에 대한 확신이 부족하던 때에 교수님의 독려가 없었다면 지금의 이 논문은 결코 완성될 수 없었을 것입니다. 그리고 잠시 학교를 비우신 정지원 교수님 대신 6 개월동안 논문지도를 해 주신 조형래 교수님께도 감사의 마음을 전합니다. 논문의 미비점을 보완하여 보다 충실한 내용이 될 수 있도록 논문 심사를 맡아주신 김동일 교수님, 강인호 교수님께 감사드리며, 항상 새로운 가르침을 주시고, 조언을 아끼지 않으셨던 김기만 교수님, 지금은 일본에 계신 민경식 교수님께도 감사드립니다.

졸업은 했지만 물심양면으로 많은 도움을 준 재범이와 상우, 상명이 그리고 항상 옆에서 힘이 되어준 원철이형, 졸업논문을 쓰기 위해 연구실에 들어와서 힘든 사항에서도 군소리 없이 묵묵히 따라와 준 학부생 태길, 승현, 승민, 건희, 민정에게 고마움을 전하며, DSP 연구실의 인식, 승수, 외형, 용주, 승용이를 비롯한 연구실에서 대학원 생활을 같이 한 대학원생들에게 감사의 마음을 전합니다. 또한 나를 걱정하고 격려해 준 93 학번 동기들과 94, 95 학번 후배들에게도 감사함과 앞으로의 무궁한 발전을 기대합니다. 비록, 지금 나와는 다른 처지에 있지만 나에 대한 변함없고 한결 같은 믿음으로 응원해준 고향 친구들에게 감사와 더불어 따뜻한 우정의 마음을 전합니다. 그리고 학부 4 학년 때 만나서 지금까지 항상 옆에서 나를 아껴주고 마음의 위로가 되어준 나의 짝지 주연이에게도 감사의 마음을 전합니다.

마지막으로, 항상 든든한 후원자이며 사랑하는 부모님께 지면으로나마 사랑한다는 말을 전하고, 서울에서 자기 일 열심히 하고 있는 하나뿐인 동생 혁이에게도 감사의 마음을 전합니다.

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