Radix-4 방식의 고속 터보 MAP 복호기 설계

정회원 김 상 훈*, 정 지 원*, 고 성 찬**

The Design of High-Speed Turbo MAP Decoder using the Radix-4 method.

Sang-Hoon Kim*, Ji-Won Jung*, Sung-Chan Ko** *Regular Members*

요 약

본 논문에서는 radix-4 방식을 이용한 고속 터보 MAP 복호 알고리즘을 제안하고 이를 설계하기 위해 VHDL 모델링하였다. VHDL 시뮬레이션을 하기위해 radix-4 방식의 터보 MAP 복호기의 구조를 설계하였으며, 복호속도 효율성을 분석하기 위해 기존의 Radix-2 방식의 복호기도 VHDL 시뮬레이션 하였다. 구현 결과, 약 2.4배의 복호 속도 향상을 알 수 있었다.

ABSTRACT

In this paper, we proposed the high-speed turbo MAP decoding algorithm using the radix-4 method, and proposed decoder is modeled by VHDL for implementation. The structure of proposed scheme is designed for VHDL simulation, also conventional scheme is simulated in order to analyze the efficiency of proposed scheme in aspect to decoding speed. As shown in simulation results, it proved that the propose scheme is more fast than conventional scheme by 2.4 times.

Ⅰ. 서 론

1993년 Berrou등에 의해 제안된 터보부호는 Eb/No 0.7dB, 부호율 1/2에서 비트오류확률의 성능을 보였다[1]. 터보부호는 연판정 입/출력(soft-in/soft-out)이 가능하고, 정보신호에 대해서 서로 다른 인터리버에 의해 분리된 2개이상의 구성코드(component code)들이 병렬연접(parallel concatenation)된 구성을 하고 있다

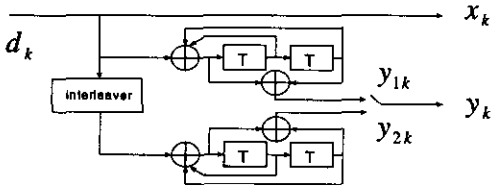
터보부호의 기본 개념은 선행하는 구성코드의 복호기 soft decision ouput을 다시 나머지 복호기에 입력하고 이러한 과정을 반복함으로써 향상된 성능을 가져온다. 터보부호의 복호기로는 SOVA(Soft Output Viterbi Algorithm), MAP(Maximum A Posteriori), Sub-MAP복호기등이 있는데[2][3], 채널의 잡음분산평가가 필요하다는 단점이 있지만 일반적으로 성능이 우수한 MAP을 사용한다. 이러한 MAP 기반의 터보부호는 MAP 복호기의 복잡성과 많은 연산량, 아주 큰 인터리버 블록크기로 인해 음성, 데이터, 동영상을 포함한 무선 멀티미디어 서비스를 요구하는 고속 무선 통신시스템의 채널부호로는 문제점을 가지고 있다. 따라서, 본 논문에서는 고속 터보 복호 알고리즘을 제시하고 VHDL을 이용하여 Log-MAP 기반의 터보 복호기 설계를 한다.

터보 복호기를 고속화를 할 수 있는 방법은 고속 덧셈/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 논문의 목표는 radix-4 복호 방식을 이용하여 기존의 터보 복호기의 복호속도를 2배이상으로 고속화시켜 MAP기반의 터보 복호기를 설계하는 것이다. 구현을 위한 설계는 VHDL 언어를 이용하였으며, timing 시뮬레이션에서 복호속도 효율성을 증명하였다.

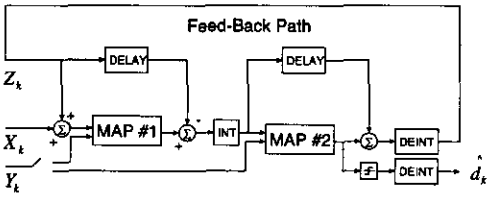
* 한국해양대학교 전파공학과 위성통신 연구실 ** 안동대학교 전자정보산업학부 디지털통신시스템 연구실
논문번호 : 00497-1230, 접수일자 : 2000년 12월 30일

Ⅱ. 터보코드의 기본 이론

구성코드가 RSC이고 부호율이 1/2인 터보코드의 부호기 및 복호기 구조를 그림 1에 나타내었다. 1비트 입력(d_k)에 2비트가 출력(x_k, y_k)되는데 x_k는 d_k와 같고 y_k는 y_{1k}, y_{2k}를 교대로 puncturing한 값으로서 패리티 정보이다. 출력신호는 디지털 변조되어 전송된 후 수신된다.



(a) 부호기 (g1/g2=7/5)



(b) MAP 방식의 터보 부호기

그림 1. 터보코드의 부·복호기

복조된 N(인터리버의 크기에 해당함)개의 신호열은 $R_1^N=(R_1,\cdots,R_k,\cdots R_N)$로 표현할 수 있다. 여기서 $R_k=(X_k, Y_k)$ 이며 $X_k=2(x_k-0.5)+N_k$, $Y_k=2(y_k-0.5)+N_k$이다. N_k는 평균이 0이고 분산이 σ^2인 가우시안 잡음이다. 복조된 N개의 신호열을 기초로 하여 k번째 송신한 비트를 추정한 LLR(Log Likelihood Ratio) λ_k는 다음과 같다.

$$\lambda_k=\frac{\log[\Pr(d_k=1\mid R_1^N)]}{\log[\Pr(d_k=0\mid R_1^N)]}=\frac{\sum_m \alpha_k^m \delta_k^{1,m}\beta_{k+1}^{f(1,m)}}{\sum_m \alpha_k^m \delta_k^{1,m}\beta_{k+1}^{f(1,m)}} \quad (1)$$

$$\alpha_k^m=\sum_{i=0}^{1}\alpha_{k-1}^{b(i,m)}\delta_{k-1}^{b(i,m)} \quad (2)$$

$$\beta_k^m=\sum_{i=0}^{1}\delta_{k-1}^{i,m}\beta_{k+1}^{f(i,m)} \quad (3)$$

m은 부호기의 상태값($2^{\nu-1}$, ν :부호기 메모리수)을 의미하고 b(i,m)은 비트 i가 입력되었을때 부호기의 현재 상태를 m으로 천이시킨 이전의 상태값을 의미하며, f(i,m)은 상태 m에서 비트 i가 입력되었을 경우 다음에 천이될 다음의 상태값을 의미한다. 그리고 $\delta_k^{i,m}$은 가지메트릭으로서 다음과 같이 표현된다.

$$\delta_k^{i,m}=\exp[2/\sigma^2\times(X_k i+Y_k C_k^{i,m})] \quad (4)$$

$C_k^{i,m}$은 상태 m에서 비트 i가 입력되었을때 부호기에서 생성되어 puncturing된 패리티 비트값이다. 그림 2.1의 복호기에서 첫번째 MAP 복호기는 총 3개의 입력 X_k, Y_k, Z_k 를 가지는데, Z_k는 두번째 MAP 복호기에서 산출한 extrinsic 정보이다. 이 extrinsic 정보는 첫번째 복호기에서 사용된 후 출력에서 제거되어 재 인터리버된다. 재 인터리버된 정보는 d_k에 대한 사전확률정보로서 두번째 복호기에 입력되며, 출력단에서는 출력값에서 입력값를 제거하고 이를 디인터리빙하여 첫 번째 MAP 복호기에 입력될 extrinsic 정보를 생성한다. 이러한 일련의 과정을 반복하여 BER 성능을 향상시키게 된다.

Ⅲ. Radix-4 방식의 고속 터보 MAP 복호 알고리즘 제안

기존 radix-2 방식을 확장해 한번의 연산으로 2비트를 동시에 복호 가능한 radix-4 방식의 MAP 복호기를 제안하였다. 기존의 트렐리스 구조에서 2개의 시점을 하나의 시점으로 간주하는 radix-4 방식을 적용하기위해 branch metrics, forward recursive metrics, backward recursive metric 공식을 제안하였다.[6]

3.1 Radix-4 방식의 터보 MAP 복호기 구조

MAP1과 MAP2가 직렬로 구성된 radix-4 기반의 터보 복호기 구조는 그림 2와 같다.

기존의 radix-2 구조와는 달리 각 상태에서 두 비트에 대한 LLR을 동시에 계산하므로 두 비트 심블 인터리버가 필요하며 네 개의 LLR, $\lambda(d_k^{00})$, $\lambda(d_k^{01})$, $\lambda(d_k^{10})$, $\lambda(d_k^{11})$ 이 각 MAP에서 구해지며 , $\lambda(d_k^{00})$은 입력비트 "00" 에 대한 LLR이다.

MAP 기반의 터보부호 복호시, radix-2 방식은 임의의 k 시점에서 복호할 때 $k-1$시점에서의 순방향 state metric α_{k-1}^m, $k+1$시점에서의 역방향

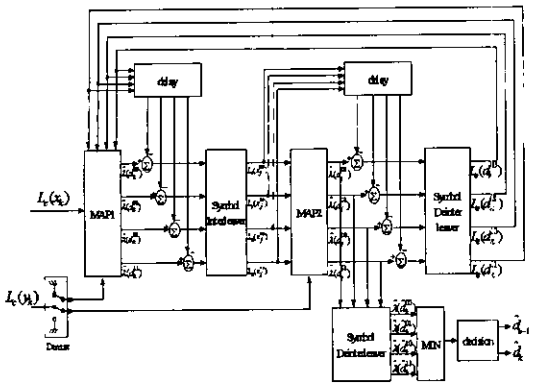
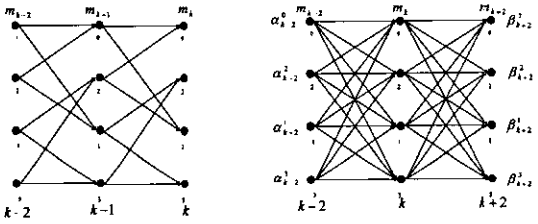


그림 2. Radix-4 기반의 터보 복호기 구조

state metric β_{k+1}^{m}, 그리고 k 시점에서 branch metric δ_k^m를 이용하여 k 시점에서 "0"과 "1"에 대한 Log Likelihood Ratio 인 LLR을 구하여 1 비트를 복호한다. 이에 반해 radix-4방식은 k 시점에서 $k-2$시점에서의 α_{k-2}^{m}, $k+2$시점에서의 β_{k+2}^{m}를 구하여 k시점에서 2비트를 동시에 복호하기 때문에 radix-2방식보다 속도가 2배 빠르며, 저장되는 인터리버 블록 크기도 절반으로 감소할 수 있다. 과거 2단의 수신비트를 입력 받아 한꺼번에 처리하는 radix-4방식의 trellis 구조는 radix-2방식의 2개 시점을 하나의 시점으로 간주하여 처리하며 radix-2방식에서 radix-4방식으로 trellis구조 변경은 그림 3과 같다.



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

그림 3. radix-4 방식의 trellis 구조

Radix-4 trellis 상에 있어서 시간 $k-2$에서 k까지의 α_{k-2}^{m} 연산과 k에서 $k+2$까지의 β_{k+2}^{m}을 구하기 위해 $k-2$시점에서 k시점으로 들어오는 상태 m_{k-2}, $k+2$시점에서 k시점으로 들어오는 상태 m_{k+2} 는 다음 식 (5)와 같이 표현된다.

$$m_{k-2}(d_{k-1}, d_k, m_k) = (d_{k-1} \oplus d_k \oplus m_{0,k}) \,\|\, (d_k \oplus m_{0,k} \oplus m_{1,k})$$

$$m_{k+2}(d_{k+1}, d_{k+2}, m_k) = (d_{k+1} \oplus d_{k+2} \oplus m_{1,k}) \,\|\, (d_{k+1} \oplus m_{0,k} \oplus m_{1,k}) \quad (5)$$

d_k는 k시점에서의 uncoded data bit 이고, m_k는 k시점에서의 state 번호 그리고 (a||b)는 a 와 b의 연접(concatenation)을 뜻한다.

3.2 δ_k^m, α_k^m, β_k^m, λ_k^m의 계산

평균이 0이고 분산 σ^2인 AWGN 채널에서의 radix-4 방식의 branch metric δ_k^m은 식 (6)과 같이 유도 할 수 있다.

$$\begin{aligned}
\delta_k^{p,m} &= P_r(D_k=p,\ S_k=m_1,\ R_k) \\
&= P_r(D_k=i_{k-1}\|i_k,\ S_k=m_k,\ R_k) \\
&= P_r(d_{k-1}=i_{k-1},\ S_k=m_{k-1},\ R_{k-1}) \\
&\quad \times P_r(d_k=i_k,\ S_k=m_k,\ R_k) \\
&= P_r(R_{k-1} \mid d_{k-1}=i_{k-1},\ S_{k-1}=m_{k-1}) \\
&\quad \times P_r(d_{k-1}=i_{k-1},\ S_{k-1}=m_{k-1}) \\
&\quad \times P_r(R_k \mid d_k=i_1,\ S_1=m_k) \\
&\quad \times p_r(d_1=i_1,\ S_k=m_1 \\
&= K_k \exp\left(-\frac{2}{\sigma^2}(x_{k-1}\,d_{k-1} \right. \\
&\quad \left. + y_{k-1}\,Y_{k-1} + x_k\,d_k + y_k\,Y_k)\right)
\end{aligned} \quad (6)$$

MAP 복호기의 branch metric $\delta_k^{p,m}$은 식(6)와 같이 전개되며 i_k는 k시점에서의 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_k는 상수, Y_k는 부호기의 state 와 입력비트 d_k에 대한 함수이다. k시점에서의 순방향 state metric α_k^m은 다음과 같다.

$$\begin{aligned}
\alpha_k^m &= P_r(R_1^{k-2} \mid D_k=p,\ S_k=m_k,\ R_k^N) \\
&= P_r(R_1^{k-2} \mid S_k=m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} p_r(D_{k-2}=p,\ S_{k-2}=m',\ R_1^{k-2} \mid S_k=m_k) \\
&= \sum_{m'} \sum_{p=0}^{3} p_r(D_{k-1}=p,\ S_{k-2}=m',\ (R_1^{k-4}, R_{k-2}) \mid S_k=m_k) \\
&= \sum_{p=0}^{3} \alpha_{k-2}^{b(p,m_k)}\,\delta_{k-2}^{b(p,m_k)}
\end{aligned} \quad (7)$$

$b(p, m_k)$는 시점 k이고 m state에서 입력이 p일 때 S시점 k-2만큼 뒤쪽으로 향하는 state의 번호를 뜻하며, 식(5)에 의해 식(8)과 같이 표현된다.

$$b(p, m_k) = m_{k-2}(d_{k-1}, d_k, m_k) \quad (8)$$

유사한 방법으로 역방향 state metric β_k^m 역시

$$\begin{aligned}\beta_k^m &= P_r(R_N^k \mid S_k = m_k)\\ &= \sum_m \sum_{p=0}^{3} P_r(D_k = p,\ S_{k+2} = m',\ R_k^N \mid S_k = m_k)\\ &= \sum_m \sum_{p=0}^{3} P_r(D_k = p,\ S_{k+2} = m',\ R_{k+2}^N,\ R_k \mid S_k = m_k)\\ &= \sum_{p=0}^{3} \beta_{k+2}^{f(p,m_k)} \delta_k^{p,m_k} \end{aligned} \quad (9)$$

와 같이 나타낼 수 있으며, $f(p, m_k)$는 입력이 p이고 state m_k일 때 k+2 시점만큼 앞으로 향하는 state의 번호를 뜻한다. 마찬가지로 식(5)에 의해서 다음과 같이 나타낼 수 있다.

$$f(p, m_k) = m_{k+2}(d_{k+1}, d_{k+2}, m_k) \quad (10)$$

식(7)와 (9)에 의해 사후확률(APP)는 식(11)와 같다.

$$\begin{aligned}\lambda_k^{p,m} &= p_r(D_k = p,\ s_k = m \mid R_1^N)\\ &= p_r(D_k = p,\ s_k = m, R_1^N) / p_r(R_1^N)\\ &= p_r(D_k = p,\ s_k = m, R_1^{k-2}, R_k^N) / p_r(R_1^N)\\ &= p_r(R_1^{k-2} \mid D_k = p,\ s_k = m, R_k^N)\\ &\cdot p_r(D_k = p,\ s_k = m, R_k^N) / p_r(R_1^N)\\ &= \alpha_k^m \beta_{k+2}^{f(p,m)} \delta_k^{p,m} \end{aligned} \quad (11)$$

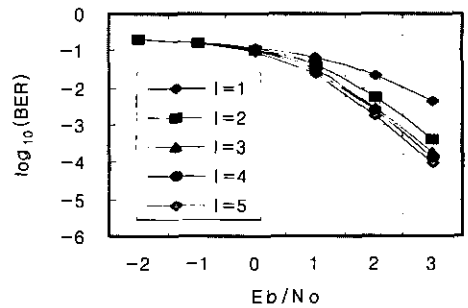
시점 k이후의 사건은 시점 k까지 관찰한 부분에 영향을 주지 않으므로 R_1^k와 R_{k+2}^N은 서로 독립적이다. 최종적으로 log-MAP일 경우 radix-4방식의 LLR은 다음과 같다.

$$LLR(D_k) = MIN\left\{\sum_m \lambda_k^{00}(m), \sum_m \lambda_k^{01}(m), \sum_m \lambda_k^{10}(m), \sum_m \lambda_k^{11}(m)\right\} \quad (12)$$

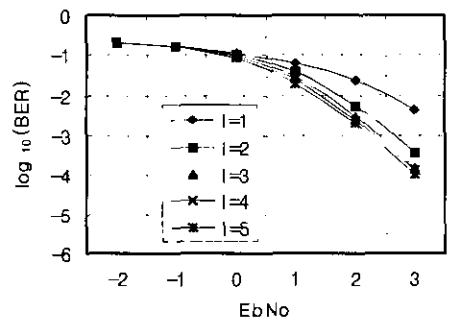
D_k={00,01,10,11}을 나타내므로 만약, $LLR(D_k)$ == λ_k^{01}이면 , d_{k-1}=0, d_k=1로 복호하여,한 시점에서 두비트를 동시에 복호할 수 있다. 그림 2에서 turbo iteration 시 extrinsic정보 Le (d_k^p), p={00,01, 10,11}을 첫 번째 MAP 에서의 branch metric $\delta_k^{p,m}$은 식 (13)과 같이 나타낼 수 있다.

$$\delta_k^{p,m} = K_k \exp\left(-\frac{2}{\sigma^2}(x_{k-1} d_{k-1} + y_{k-1} Y_{k-1} + x_k d_k + y_{Y_k})\right) \times L_e(d_k^p) \quad (13)$$

그림 4는 인터리버크기를 Radix-2에서는 200으로, Radix-4에서는 100으로하여 컴퓨터 시뮬레이션 결과이다. 시뮬레이션 결과, 성능이 일치함을 알 수 있다



(a) radix-2 방식 (N=200)



(b) radix-4 방식 (Ns=100)

그림 4. N = 200,Ns = 100 일 때 radix-2 방식과 radix-4방식 성능 비교

Ⅳ. 기존 및 제안한 방식의 터보 복호기 설계

터보 복호기를 포함하는 수신 시스템의 개략적인 블록도를 그림 5에 나타내었다. 본 논문에서는 디지털방식의 복조기를 고려한다.

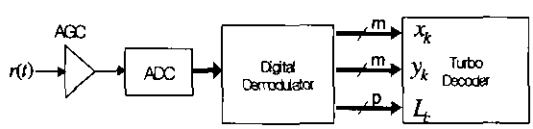


그림 5. 터보 복호기 수신 시스템의 블록도

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. 터보 복호기에는 수신신호에 대한 연판정값이 입력

되어야 하므로 복조기는 복조를 수행하고난후 X_k, Y_k에 대한 대한 m비트의 연판정 결과값 및 채널 신뢰도값 $L_c = 2/\sigma^2$에 대한 p 비트의 평가량을 출력한다. 터보 복호에는 요구되는 연산량이 많고 특히, 메모리가 많이 요구되기 때문에 복호성능과 H/W복잡성을 최적으로 절충하여 비트수 m과 p을 결정하여야 한다. Log-MAP 방식의 터보 복호에서는 X_k, Y_k L_c 비트수에 의해서 다른 주요 구성요소들의 비트수가 결정되므로 사실은 복호성능과 H/W 복잡성이 m과 p의 값에 의해 전적으로 좌우된다.

4.1 기존 터보 복호기 설계

4.1.1 Log-MAP기반의 터보 복호기 모듈분석

Log-MAP기반의 터보복호기의 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), LLR을 계산하는 LLRC (Log Likelihood Ratio Calculator)등을 들 수 있으며, 이들에 대해서는 그림 6과 같이 구성할 수 있다.

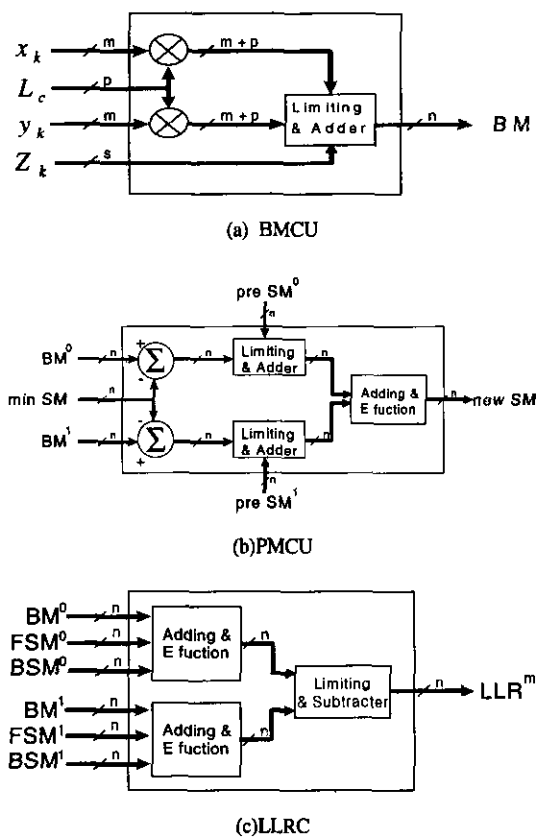


그림 6. Log-MAP 기반의 터보 복호기의 주요 블록에 대한 회로도

그림 6의 (a)BMC 블록도에서 알 수 있듯이, 가지메트릭 BM의 출력비트수 n은 수신신호 X_k, Y_k와 L_c와의 곱셈연산후 m+p로 증가하고 덧셈후 1 비트가 더추가된다. 2번째 iteration부터는 첫 번째 iteration 에서의 LLR값(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 iteration수에 비례하여 n값이 증가하게 된다. 따라서, limiting & adder를 적용할 경우 BM값의 표현 비트수 n은 m과 p에 의해서만 결정된다. 그리고 FSM과 BSM을 구하는 그림6의 (b) SMC에서는 먼저, 출력될 상태메트릭값이 발산하지 않도록하기 위해서 이전시점에서의 최소 상태메트릭값을 가지메트릭값에 빼준 다음, 이전시점의 상태메트릭값과의 합을 E함수연산을 하여 새로운 상태메트릭값을 구한다. 이때, BMC에서와 마찬가지로 상태메트릭값을 n비트로 제한하기 위하여 내부의 덧셈연산시 limiting & adder를 사용한다. 그림6의 (c) LLRC 에서는 입력비트 0에 대한 각 메트릭과 1에대한 각 메트릭값들을 더한 다음 E함수연산을 행한후 뺄셈연산을 하여 한상태 m에서의 LLR값을 구한다. 역시, LLR출력 비트수를 n비트로 제한하기 위하여 내부연산에서 각각 limiting adder와 limiting subtracter을 사용한다.

4.1.2 구현을 위한 각 메트릭의 최적 비트수 결정

앞절에서 설명한 바와 같이 터보 복호기 내부의 각 메트릭값 BM, SM(FSM,BSM), LLR의 비트수 n은 m과 p가 결정하게된다. 따라서 수신신호X_k, Y_k에 대한 적정 양자화 비트수의 결정과 채널 신뢰도 L_c의 처리과정은 log-MAP 방식의 터보복호기 하드웨어 구현에 있어서 매우 중요한 요소가 된다. 참고문헌 [5]에 의하면 수신신호의 양자화비트수는 6비트로하고 L_c는 2비트로하는 것이 최적의 할당비트인 것으로 설명하였다. 따라서, 각 메트릭비트 n은 9비트가 된다. 이는 그림 7의 컴퓨터 시뮬레이션을 통한 성능분석으로도 확인할 수 있었다. 시뮬레이션 환경은 Log-MAP/터보 복호기를 구성하고 N=64비트, iteration 횟수 I=3, 수신신호양자화 비트수 m은 6비트로, 그리고 L_c는 2비트로 하였다. 시

뮬레이션결과 메트릭비트 n을 8비트에서 9비트로 증가시켰을 경우 성능이 향상하였지만 9비트 이상으로 하였을때는 성능의 향상이 없어서 n을 9비트로 하는 것이 최적임을알 수 있었다. 또한, 같은 환경에서 수신신호의 양자화 비트수를 6비트에서 8비트로 확장하였을 경우에는 L_c 연산을 행하지 않은경우가 오히려 성능이 우수한 것을 알수 있었다. 이를 바탕으로한 Log-MAP/터보 복호기의 구현을 위한 최적 비트수 할당 내역을 표 1에 정리하였다.

표 1. Log-MAP/터보 복호기의 각 메트릭의 최적 비트수 할당

	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

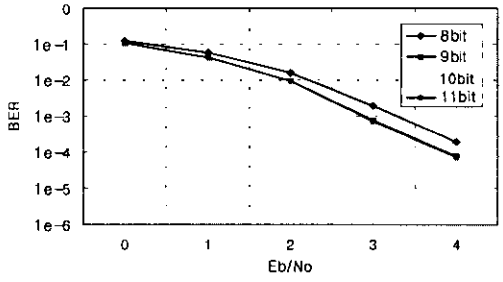


그림 7. 메트릭 비트 n에 따른 Log-MAP/터보복호기의 성능 (N=64, I=3)

4.1.3 E-Table의 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 ROM내의 주소를 지시하도록 설계한다. 본 논문에서 사용한 E-Table은 입력 6비트, 출력 4비트로 하였다. 따라서 설계된 E-Table의 크기는 26*4=256비트 이다. E-Table의 내용은 FLEX10K디바이스 내부의 EAB(Embaded Array Block)의 ROM영역에 할당하여 저장하였다.

4.1.4 Log-MAP 복호기에서의 메모리 할당

Log-MAP 복호기에 있어서 역방향 상태메트릭을 구하는 과정 때문에 수신신호 X_k, Y_k 를 복호블록 N만큼 저장하여야 한다. 가지메트릭 BM은 각 메트릭을 구할때마다 계산하지 않고, 저장된 수신신호를 읽어들여 N블록만큼 모든 가지메트릭을 메모리에 저장하고 각 메트릭연산시 BM값을 읽어들인다. 역방향 상태메트릭 BSM은 구하는 순서는 역방향이지만 LLR 연산시 순방향으로 출력값이 입력되어야 하므로 역시 역방향으로 구한 BSM을 메모리에 저장시켜야 한다. 이와 같이 Log-MAP복호기의 설계에 요구되는 모든 메모리는 FLEX10K-100디바이스의 내부램인 EAB를 사용하였다. 그러나, 본 논문에서 사용한 FLEX10K-100GC503-4 디바이스에서 지원하는 EAB의 한계용량 때문에 복호블록 N=64비트로 제한하였다. 이때, 사용되는 메모리 용량을 계산하면 먼저 수신신호를 저장하는 메모리용량은 8비트 양자화신호이므로 2*8*64=1024비트, BM 메모리는 한 시점에서 존재하는 가지메트릭 BM00, BM01, BM10, BM11을 복호블록 만큼 저장해야 하므로 요구되는 메모리는 4*9*64=2304비트가 요구된다. 여기서, BMxx는 가지부호어 xx에 대한 가지메트릭이다. BSM 메모리 역시 한 시점에 있어서 4개의 각 상태에 해당하는 메트릭 BSM0, BSM1, BSM2, BSM3를 복호블록 만큼 저장해야 하므로 요구되는 메모리는 4*9*64=2304비트이다.

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

표 2. Log-MAP/터보 복호기의 메모리 할당

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

4.1.5 Log-MAP 복호기의 VHDL설계

Log-MAP복호기를 전체적으로 구성한 것을 그림 8에 나타내었다. 먼저, 수신신호를 m비트 양자화한 신호를 각각 I_ch, Q_ch에 입력하고 BMU(Branch Metric Unit)에서 순방향으로 각 수신신호에 대한 가지메트릭 BM00, BM01, BM10, BM11을 구함과 동시에 이를 BM메모리에 저장한다. 그 다음, 역방향으로 BM메모리에 저장된 가지메트릭값을 불러들여 역방향 상태메트릭 BSM을 구하고 마찬가지로 BSM메모리에 저장한다. 다시, 순방향으로 BM메모리로 부터 가지메트릭을 불러들여 순방향 상태메트릭 FSM을 구하고 이 값과 BSM메모리와 BM메모리로 부터 불러들인 각 메트릭값을 동시에 LLRU에 입력하여 LLR을 구한다.

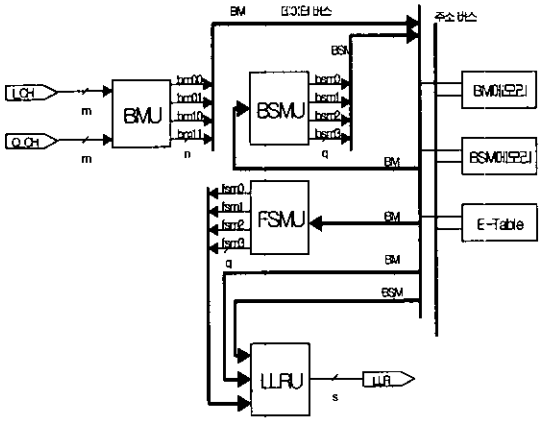


그림 8. Log_MAP 복호기의 하드웨어 구조

터보 복호시 필요한 2개의 Log-MAP복호기는 그 동작이 직렬동작 즉, 앞쪽의 복호기의 출력이 끝난 다음 그 출력을 입력받아 뒤쪽 복호기의 복호동작이 시작되므로, 본 설계에서는 Log-MAP복호기를 2개를 사용하지 않고 하나의 Log-MAP복호기를 "serial_sw" 라는 모듈에서 스위칭동작을 하여 동일한 MAP복호기로 각각 다른 복호동작이 가능하도록 설계하였다.

이러한 Log-MAP/터보 복호기를 VHDL로 설계한 Top레벨 회로도를 그림 9에 나타내었다. 여기서, 9비트 Log-MAP복호기를 하나의 모듈로 심볼화한 것이 "B9_MAP" 모듈이며 "SC_RAM"모듈에 컴퓨터 시뮬레이션을 통하여 얻어진 수신신호를 저장시켰다. 그리고, "INTLV"는 인터리버 "DE_INTLV"는 디인터리버이다.

타이밍 시뮬레이션 결과를 그림 10에 나타내었다. 복호블록 N=64비트 이고 타이밍 분석결과 제일 빠른 클럭인 16배수 클럭의 요구 주기는 Log-MAP복호기에서와 마찬가지로 약 44ns 이었다. 그림 10의 (a) 에서와 같이 1번 복호동작시 지연이 약 136us 였으며, 이는 터보복호기가 64비트를 복호하는데 소요된 총 클럭지연이 136us= 0.00732 Mbps 이라는 의미이다. 따라서, 복호기의 복호속도는 0.00732Mbps * 64 = 0.47 Mbps 이다. 이는, 수십 Kbps의 전송속도를 가지는 IMT-2000시스템에 만족하는 속도이다. 그리고, Iteration이 증가 할수록 이러한 복호속도는 복호지연만큼 줄어들게 되는데, 예를 들어, Iteration=3일 경우 총 복호동작은 3번이며 복호속도는 0.47 / 3 = 0.156 Mbps 가 된다.

구현한 복호기의 복호동작을 알아보기 위하여 N=64으로 하고 Iteration에 따른 복호기의 출력을 그림 11에 나타내었다. 분석 결과 첫번째 Iteration 시 5개의 오류를 그리고, 두번째는 3개의 오류를 가지는 출력을 하였으나 3번째 Iteration부터는 모든 오류를 정정한 출력을 하고 있는 것을 알수 있다.

사용된 FLEX10K100GC503-4 디바이스의 report 파일을 그림 4.8에 나타내었다. 총 10만 게이트중에서 62%를 사용하였다.

사용된 디바이스는 ALTERA사의 FLEX10K 100GC503-4 이며, 내부메모리 EAB의 제약 때문에 복호블록 N=64 비트로 설정하였으며, 1배수, 4배수, 16배수의 총 3개의 클럭이 사용되었다

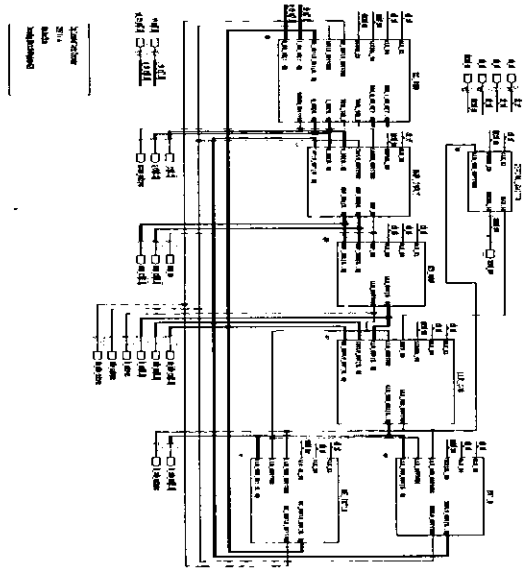
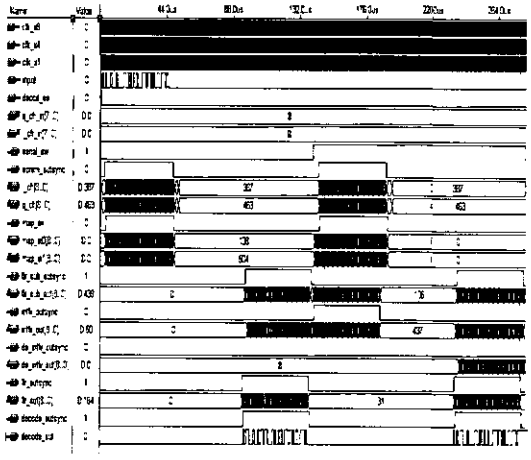
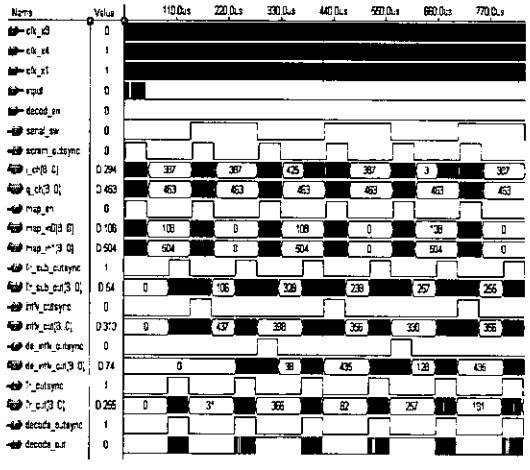


그림 9. N=64 인 Log-MAP/터보 복호기의 회로도



(a)Iteration = 1



(b)Iteration = 3

그림 10. Log-MAP/터보 복호기의 타이밍 다이어그램

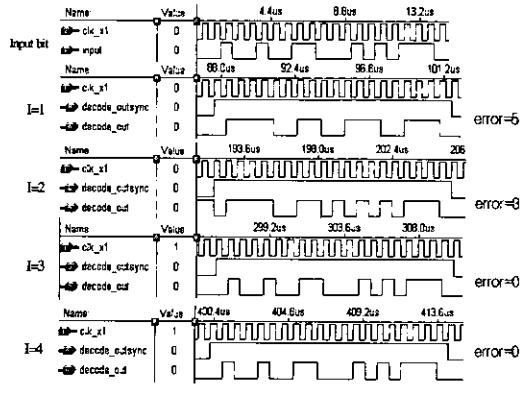


그림 11. Iteration에따른 복호과정 (Eb/No=1[dB])

```
** 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 12. 터보 복호기에 사용된 디바이스 report 파일

4.2 radix-4 방식 터보 복호기 설계

4.2.1 구현을 위한 구조 설계

본 논문에서 radix-4 방식의 터보 복호기을 구현할 경우, 구현을 위한 구조 설계도는 그림 13과 같다. 그림 13에서 알 수 있듯이, N개의 I_ch, Q_ch 수신신호는 각각 짝수 번째와 홀수 번째로 분리하여 수신신호 메모리에서 N/2 개의 4개의 신호를 출력한다. 수신신호 메모리에서 출력될 때 sync 제어 신호를 생성하여 BMC로 입력한다.

BMC에서는 수신신호의 메모리에서 출력된 sync 신호를 입력 받아 BM0000부터 BM1111까지의 Branch Metric을 구하고 이를 BM 메모리에 저장한다. 저장이 완료되면 각 상태에서의 BSM을 구하기 위해 BM 메모리의 역순으로 읽으면서 BSM을 계산하다. 계산된 BSM은 BSM 메모리에 저장되며, 저장이 완료되면 BSM_sync를 생성하여 BM 메모리로 입력한다.

BM 메모리에서는 BSM_sync 신호가 입력되며 각 상태에서의 FSM을 구하기 위해 BM 메모리의 BM0000부터 BM1111을 순방향으로 읽으면서 FSM을 구한다. FSM을 구하는 동시에 BSM 메모리에 저장된 정보를 역순으로 읽으면서 복호하고 타이밍

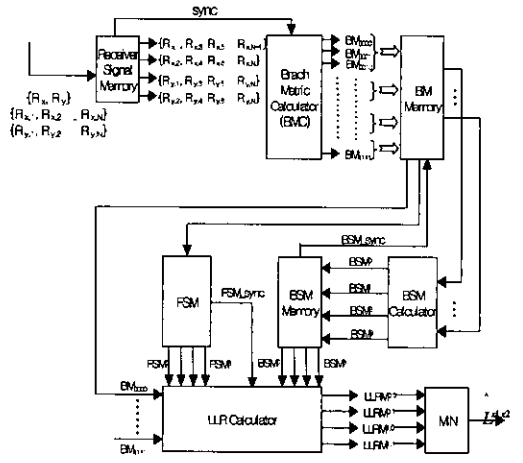


그림 13. radix-4 터보 MAP 회로도

분석도는 그림 14과 같다. BM, 계산된 BSM, FSM을 이용하여, LLR을 구한다. LLR을 구할 때 BSM, FSM, BM의 정확한 동기를 맞추기 위해, 그리고 FSM이 출력되는 순간 LLR을 구하기 위해 FSM_sync 신호를 LLRC에 보낸다.

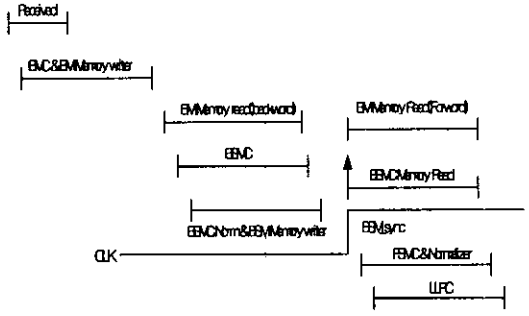


그림 14. 타이밍 분석도

4.2.2 최적의 비트수 결정

터보 복호기 내부의 각 메트릭값 BM, SM (FSM,BSM), LLR의 비트수는 수신신호 Xk, Yk에 대한 적정 양자화 비트수의 결정과 채널신뢰도 Lc의 비트수에 결정이 된다. 여기에서는 Lc연산은 제거 하여 8Bit로 할당하였다.

먼저 수신신호의 비트수를 결정하기 위해 100000개의 데이터를 입력하고 각 수신신호의 양자화 비트 수에 따른 성능을 시뮬레이션하였다. 그림 15에서 보는 바와 같이 8비트 이상에서는 성능의 차이가 거의 없으므로 수신신호의 최적의 양자화 비트수는 8비트임을 알 수 있다.

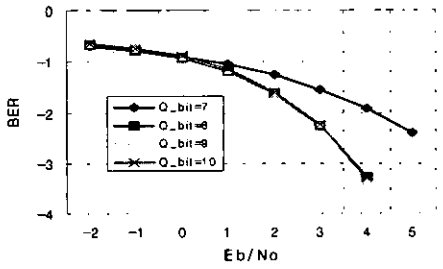
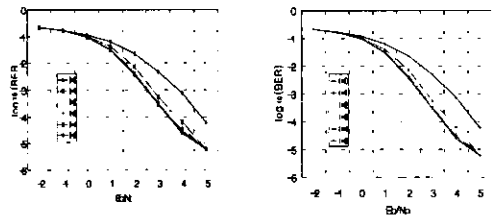


그림 15. 수신신호의 비트수에 따른 radix-4 Log-MAP/터보 복호기 성능(N=100, I=1)

다음은 각각의 메트릭 비트에 따른 성능을 평가하기 위하여 시뮬레이션을 하였다. 시뮬레이션 환경은 인터리버 사이즈 N=100, 데이터 500000개, 수신신호의 양자화 비트수는 8비트로 고정을 하고 각 메트릭 비트수를 변화시켰을때 성능분석은 그림 16와 같다. 이를 바탕으로 radix-4 Log-Map/터보 복호기의 구현을 위한 최적 비트수 할당 내역을 표 3에 정리 하였으며 기존의 Radix-2 방식과 비교하여 볼 때, bit수의 할당은 동일함을 알 수 있다.



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

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

그림 16. 각 메트릭비트에 의한 radix-4 Log-MAP/터보 복호기 성능

표 3. radix-4 Log-Map/터보 복호기의 각 메트릭의 최적 비트수 할당

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

4.2.3 VHDL 설계 결과 검토

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

시뮬레이션 환경은 기존 방식과 비교하기 위해 Chip을 동일하게 설정하였다. 최소 클럭은 약 58ns였으며, 기존의 방식이 44ns에 비해 약 12ns가 더 많이 요구되어 짐을 알 수 있다.

이유는 첫째로 기존의 방식은 BM을 구할 때 1개의 가산기만 요구되나, radix-4 방식은 4개의 수신신호를 더해야 하므로, 3개의 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 E함수가 기존의 방식은 2개가 필요하나 제안된 radix-4 방식에서는 3개가 요구되므로 클럭이 증가하였다. 그림 17에서 그림 20까지는 BM, BSM, FSM, LLR의 VHDL 시뮬레이션 결과를 나타낸 것이다.

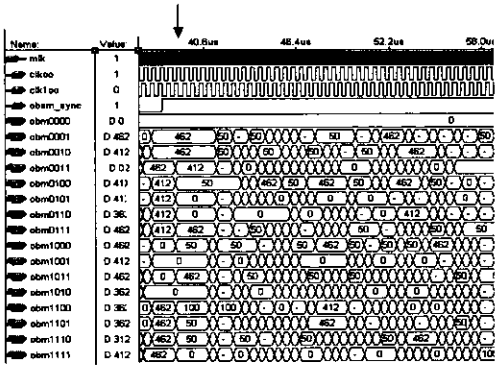


그림 17. BM VHDL 시뮬레이션 결과

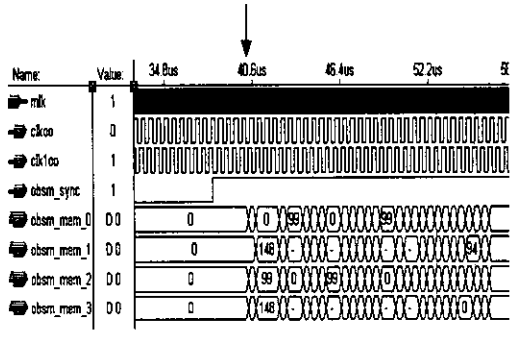


그림 18. BSM VHDL 시뮬레이션 결과

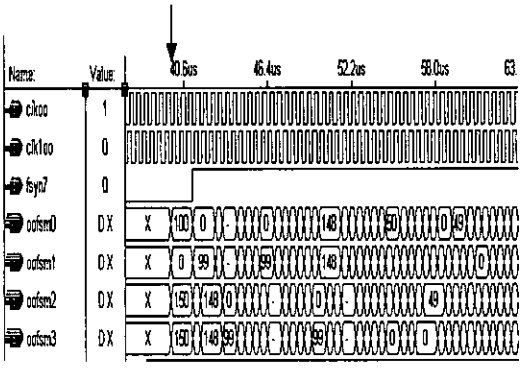


그림 19. FSM VHDL 시뮬레이션 결과

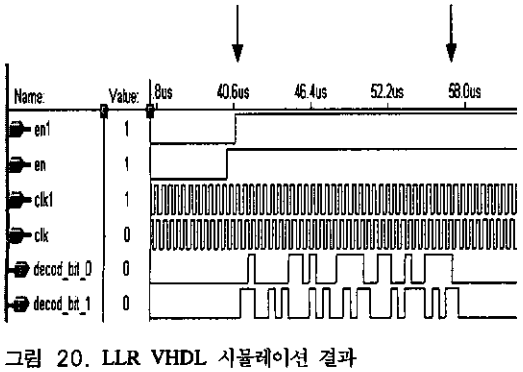


그림 20. LLR VHDL 시뮬레이션 결과

4.3 구현결과 비교 검토

본 논문의 처리 속도 및 지연에 대한 구현 결과 비교는 표 4와 같다. 표 4의 세부항목들은 그림 17에서 그림 20의 화살표 표시된 부분의 시점이다.

표 4. 제안한 방식과 기존의 방식의 처리 속도 비교

항목 \ 방식	기존의 방식 (radix-2)	제안한 방식 (radix-4)
main clock	44ns	58ns
BM delay (FSM을 구하기 위한 BM의 순방향 read 초기 time)	66us	38us
FSM delay (FSM 계산 초기 time)	67.6us	40us
BSM delay (LLR을 구하기 위한 BSM 역방향 read 초기 time)	67us	39us
LLR 출력 (LLR 출력 end time)	136us	57us
복호 속도	0.47[Mbps]	1.12[Mbps]

표 4에서 알 수 있듯이 전체 지연 시간은 LLR의 출력이 끝나는 시간이므로 기존 방식에 비해 약 2.4배 정도 지연을 감소 시켰으며, 복호 속도는 기존 방식에 비해 약 2.4배 향상 시켰음을 알 수 있다.

V. 결 론

현재의 터보코드 응용 및 적용범위는 기존의 Viterbi 복호기와는 달리 처리할 데이터양과 연산작용이 매우 복잡해 처리속도가 저속이어서 음성, 데이터등의 저속 서비스에만 적용될 예정이며, 고속데이타, 동영상등의 고속 서비스로의 적용은 불가능한 실정이다.

터보 복호기를 고속화를 할 수 있는 방법은 복호기 구조를 한 클럭에 한 비트를 복호하는 기존의 radix-2 방식이 아니라 한 클럭에 2비트를 복호가는 radix-4방식의 적용이 필수적이다. 따라서 본 논문 목표는 radix-4 복호 방식을 이용하여 기존의 터보 코드 복호속도를 2배이상으로 고속화시켜 MAP 기반의 터보 복호기를 FPGA(Field Programmable Gate Array)화 하는 것이다.

본 논문에서는 radix-4 방식을 적용한 Log-MAP 기반의 터보 복호 알고리즘을 제안 하였고, 제안한 알고리즘을 토대로 구현을 위한 bit 수를 설정하여 VHDL 시뮬레이션 하여, FPGA test board에 구현 하였다.

또한 제안한 방식의 지연 및 속도 면에서 효율적임을 증명하기 위해 기존의 radix-2 방식의 Log-MAP 기반의 터보 복호기도 아울러 구현하였다. 시뮬레이션 결과 제안한 방식과 기존 방식과의 성능 면에서 비교해 볼 때, 성능이 일치함을 알 수 있었고 구현 결과 지연은 기존의 방식에 비해 약 2.4배 감소 시켰고 복호 속도는 기존의 방식에 비해 약 2.4배 향상됨을 확인 할 수 있었다. 따라서 차세대 이동통신인 IMT-2000등과 같은 단말기에 터보 부호의 실시간 적용이 가능하리라 사료된다.

참 고 문 헌

[1] C.Berrou, A.Glavieux, and P.thitimajshima, "Near Shanon Limit Error-Correcting Coding and Decoding : Turbo-Codes," in Proc. ICC93,1993.

[2] 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.

[3] 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, June 1995.

[4] 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, 1995.

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

[6] 정지원 외 4인, "Radix-4 방식의 터보 MAP복호 알고리즘", 25권 4A호 한국통신학회, 2000. 4.

김 상 훈(Sang-Hoon Kim) 준회원

2000년 2월 : 한국해양대학교 전파공학과 학사

2000년 3월~현재 : 한국해양대학교 전파공학과 석사

<주관심 분야> 이동통신, 채널코딩, FPGA 기술 등

정 지 원(Ji-Won Jung) 정회원

1989년 2월 : 성균관대학교 전자공학 학사

1991년 2월 : 성균관대학교 전자공학 석사

1995년 2월 : 성균관대학교 전자공학 박사

1991년 1월~1992년 2월 : LG 정보통신 연구소 연구원

1995년 9월~1996년 8월 : 한국통신 위성통신연구실 선임연구원

1997년 3월~1998년12월 : 한국전자통신연구원 초빙연구원

1996년 9월~현재 : 한국해양대학교 전파공학과 조교수

<주관심 분야> 위성통신, 이동통신, 변.복조기술, 채널코딩, FPGA 기술 등

고 성 찬(Sung-Chan Ko) 정회원

1987년 2월 : 성균관대학교 전자공학 학사

1989년 2월 : 성균관대학교 전자공학 석사

1993년 2월 : 성균관대학교 전자공학 박사

1993년 3월~1996년 2월 : 한국전자통신연구소 위성통신연구단 선임연구원

1996년 3월~현재 : 안동대학교 전자정보산업학부 조교수

1997년 2월~1998년 12월 : 한국전자통신연구원 초빙연구원

<주관심 분야> 위성통신, 이동통신, 변.복조기술, 채널코딩, VHDL 등