논문 2011-48TC-7-2

높은 처리율을 갖는 고속 터보 복호 기법

(High Throughput Turbo Decoding Scheme)

최 재 성*, 신 준 영**, 이 정 우***

(Jaesung Choi, Joonyoung Shin, and Jeong Woo Lee)

요 약

본 논문에서는 높은 처리율을 가지는 다양한 터보 복호 방식을 소개하고 각각의 장점을 기반으로 한 새로운 고속 터보 복호 기법을 제안한다. 제안된 기법은 기본적으로 슬라이딩 윈도우, 복류 복호, 셔플 복호 방식을 사용하며 모의실험 결과, 제안된 기법은 기존의 방법에 비해서 적은 클록 사이클로도 동일한 BER 성능을 얻을 수 있음을 확인하였다. 게다가 슬라이딩 윈도우 크기를 적절하게 조정하면 메모리 사용량도 줄일 수 있음을 확인하였다. 따라서 본 논문에서 제안한 터보 복호 기법을 사용함으로써 저 전력, 저 면적의 고속 터보 복호기의 설계가 가능하다.

Abstract

In this paper, various kinds of high throughput turbo decoding schemes are introduced, and a new turbo decoding scheme using the advantages of each scheme is proposed. The proposed scheme uses the decoding structure of double flow scheme, sliding window scheme and shuffled turbo decoding scheme. Simulation results show that the proposed scheme offers a BER performance equivalent to those of existing turbo decoding schemes with less clock cycles. We also show that the required memory can be reduced by choosing proper size of sliding window. Consequently, we can design a high throughput turbo decoder requiring low power and low area.

Keywords : Turbo decoder, shuffled decoding, sliding window, double flow, high throughput decoder.

Ⅰ. 서 론

Berrou에 의해 1993년 제안된 터보 부호[1]는 강력한 오류정정성능 때문에 많은 관심을 받아왔다. 그러나 터보 부호의 복호에 사용되는 MAP 알고리즘[2]은 연산 복잡도가 매우 높아 복호기의 낮은 처리율과 긴 복호지연시간을 유발하므로 터보 부호가 고속의 데이터 통신 시스템에 사용되는 데에는 많은 문제가 존재한다. 또한 터보 복호기는 복호과정에서 순방향 및 역방향 메트릭, 외부 정보, 인터리버 등을 저장하기 위해 많은 양의 메모리를 필요로 하며 부호어 블록의 복호에 많은 클록 사이클을 사용한다. 따라서 터보 부호가 고속의 디지털 통신 시스템에 사용되기 위해서는 고속, 저 전력, 저 면적의 터보 복호기 설계가 반드시 필요하다. 특히 최근 들어 15Mb/s-35Mb/s의 데이터 전송속도를 지원하는 와이브로 시스템과 고속의 상태에서 100Mb/s-1Gb/s의 데이터 전송 속도를 지원할 것으로 예상되는 4세대 셀룰러 통신 시스템의 오류정정성능을 높이기 위해 터보 부호가 채택되어 사용되는 추세이므로 이와 같은 터보 복호기 설계가 중요한 이슈가 되고 있다.

터보 복호과정에 있어 높은 처리율을 얻고 시스템 메모리 사용량을 줄이기 위해서 많은 접근 방법들이 소개되었다[3~9]. 논문 [3~4]에서는 효과적인 방법으로 메모리 사용량을 줄이며 또한 복호 지연시간도 줄일 수 있는 방법인 슬라이딩 윈도우(sliding window) 방식이 소

* 정회원, *** 정회원-교신저자, 중앙대학교 전자전기공학부
(Chung-Ang University)
** 정회원, SK텔레콤 네트워크 기술원
(SK Telecom)
※ 본 연구는 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행하였음
(KRF-2008-313-D00695, 2009-0075215)
접수일자: 2011년3월21일, 수정완료일: 2011년7월13일

개되었다. 논문 [5]에서는 데이터 블록을 양쪽에서 복호해 나가는 복류(double flow) 방식이 소개되었다. 그리고 논문 [6]에서는 외부 정보를 효율적으로 이용하는 병렬 복호 방법인 셔플 복호(shuffled iterative decoding)방식이 소개되었다.

본 논문에서는 기존의 복호 방식들을 간략하게 소개한 후 이들의 장점을 기반으로 한 새로운 복호 기법을 제안한다. 각각의 장점을 혼합해서 적용하면 메시지 비트의 복호 순서가 바뀌게 되는데 이에 따라 외부 정보의 업데이트 규칙이 달라진다. 본 논문에서는 복합적인 복호 구조와 외부 정보의 업데이트 규칙을 제안하며, 기존의 방법과의 복호연산 소요 클록 사이클 수 및 BER 성능을 비교해본다.

본 논문은 다음과 같이 구성되어있다. Ⅱ장에서 기본적인 터보 복호 알고리즘에 대해 소개하고, Ⅲ장에서 높은 처리율을 갖는 터보 복호 기법들이 소개되며, Ⅳ장에서 새로운 터보 복호 기법을 소개한다. 성능 비교 및 모의실험 결과가 Ⅴ장에 소개되고 마지막 장에서 결론을 맺는다.

Ⅱ. 터보 복호 알고리즘

m_j를 터보 부호기로 들어가는 j번째 메시지 비트라 하고 부호기의 출력인 systematic 비트와 parity 비트를 각각 x_j^s와 x_j^p라 하자. 여기서 $j=1,2,...,N$ 이고, N은 메시지 블록의 길이를 나타낸다. y_j^s와 y_j^p를 각각 x_j^s와 x_j^p가 변조 후 채널을 통과한 뒤 수신단에서 수신된 신호라 하고 간결한 표기를 위해 $y_j=(y_j^s,y_j^p)$라고 하자. 터보 복호기는 그림 1에 나타내었듯이 MAP 알고리즘 기반으로 동작하는 두 개의 요소 복호기(component decoder) 간에 정보를 상호 교환하면서 반복 복호를 수행한다. MAP 알고리즘에서 얻어지는 j번째 메시지 비트에 대한 LLR(Log Likelihood Ratio)은

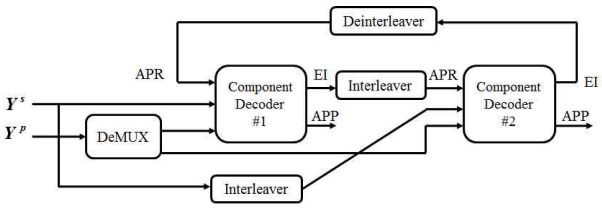

그림 1. 터보 복호기의 블록도
Fig. 1. Block diagram of Turbo decoder.

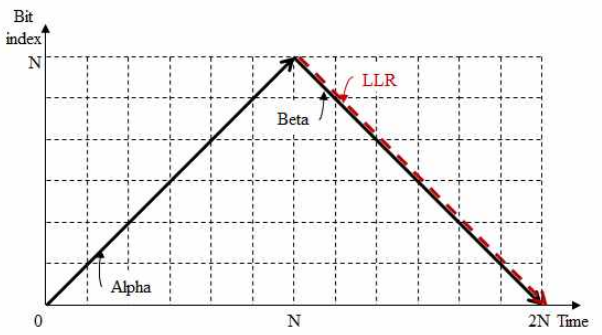

그림 2. 직렬 터보 복호방식의 타일 다이어그램
Fig. 2. Tile diagram of serial turbo decoding scheme.

$$L(m_j)=\log\left(\frac{\sum_{S_j}\sum_{S_{j-1}}\gamma_1(y_j,S_{j-1},S_j)\alpha_{j-1}(S_{j-1})\beta_j(S_j)}{\sum_{S_j}\sum_{S_{j-1}}\gamma_0(y_j,S_{j-1},S_j)\alpha_{j-1}(S_{j-1})\beta_j(S_j)}\right) \quad (1)$$

와 같다. 여기서 S_j는 j번째 시간의 트렐리스(trellis) 상태를 나타낸다. 순방향 상태 메트릭(α 메트릭)과 역방향 상태 메트릭(β 메트릭)은 각각

$$\alpha_j(S_j)=\frac{\sum_{S_{j-1}}\sum_{i=0}^{1}\gamma_i(y_j,S_{j-1},S_j)\alpha_{j-1}(S_{j-1})}{\sum_{S_j}\sum_{S_{j-1}}\sum_{i=0}^{1}\gamma_i(y_j,S_{j-1},S_j)\alpha_{j-1}(S_{j-1})} \quad (2)$$

$$\beta_j(S_j)=\frac{\sum_{S_{j+1}}\sum_{i=0}^{1}\gamma_i(y_{j+1},S_j,S_{j+1})\beta_{j+1}(S_{j+1})}{\sum_{S_j}\sum_{S_{j+1}}\sum_{i=0}^{1}\gamma_i(y_{j+1},S_j,S_{j+1})\alpha_j(S_j)} \quad (3)$$

와 같이 재귀적으로 얻어지는데, 식 (2)와 (3)의 분모는 값의 발산을 막기 위해 사용되며 저 복잡도 알고리즘에서는 생략가능하다. 상태 천이 메트릭(γ 메트릭)은

$$\gamma_i(y_j,S_{j-1},S_j)=p(y_j^s|m_j=i)p(y_j^p|m_j=i,S_j,S_{j-1}) \cdot p(m_j=i|S_j,S_{j-1})p(S_j|S_{j-1}); \quad i=0,1. \quad (4)$$

와 같고 여기서 $p(m_j=i|S_j,S_{j-1})$는 S_{j-1}에서 S_j로 천이될 때 비트 i에 따라 0 또는 1의 값을 갖는다. 그리고 식 (4)의 마지막 항은 m_j에 대한 사전확률(a priori probability)을 나타낸다. 두 개의 요소 복호기는 각각 번갈아 가며 외부 정보를 만들어내고 서로에게 보내서 사전확률로 사용한다. 이전 요소 복호기의 외부 정보(extrinsic information)는 다음 요소 복호기에서 사전확률로 사용되는데 반복 복호 횟수가 늘어날수록 Shannon의 이론적인 한계에 근접하는 오류정정성능을

나타낸다.

한편 MAP 알고리즘 및 Log-MAP 알고리즘은 실제 시스템에서 구현하기에는 연산량이 매우 높다. 따라서 실제 터보 복호기 하드웨어에서는 복호 성능의 열화를 감수하고 저 복잡도 알고리즘을 사용하는데 Max-Log-MAP이 대표적이다[7]. 이때 외부 정보에 스케일링 팩터(scaling factor)를 곱해 줌으로써 Max-Log-MAP의 성능 열화를 보상할 수 있다[8].

일반적으로 MAP, Log-MAP, Max-Log-MAP 복호 알고리즘은 모든 γ와 α, 그리고 β 메트릭 값을 순차적으로 구한 후에 LLR을 구할 수 있다. 본 논문에서는 γ 메트릭이 α와 β 메트릭을 구하기 전에 미리 계산되어 메모리에 저장되는 방식을 고려한다. α와 β 메트릭, LLR 연산에 관한 타일 다이어그램을 그림 2에 나타내었으며, 본 논문에서는 이러한 복호기법을 직렬 터보 복호(serial turbo decoding)라 칭한다. 그림에서 보듯이 직렬 터보 복호의 경우 LLR 계산을 위해 모든 α메트릭 값이 저장되어야한다. 이는 각 요소 복호기에 대해 N 만큼의 저장 공간을 필요로 하므로 복호기 하드웨어의 면적증가와 비용증가를 야기한다. 게다가 각각의 요소 복호기가 $2N$의 클록 사이클을 통해 LLR을 얻고 이를 인터리빙(interleaving) 또는 디인터리빙(de-interleaving)한 후 다음 요소 복호기로 전달해야 하므로 결과적으로 직렬 복호기는 1회 반복 당 $4N$이라는 클록 사이클을 필요로 한다. 이는 높은 복호 지연시간을 유발하여 고속의 데이터 전송에 사용하기 어려워진다.

Ⅲ. 높은 처리율을 갖는 터보 복호 기법

1. 슬라이딩 윈도우 기법

앞서 설명한 터보 복호기의 메모리 문제에 대한 해결책으로 슬라이딩 윈도우 기법이 제안되었다[3, 4]. 슬라이딩 윈도우 방식은 순방향 재귀는 그대로 진행하되 역방향재귀를 분할하여 진행하는 특징을 가진다. 역방향 재귀는 $k \cdot SW$ 번째 비트색인(bit index)부터 $2SW$ 클록 사이클 동안 수행되는데, 여기서 SW는 슬라이딩 윈도우의 크기를 나타내고 $k=2,3,...,N/SW$이다. $2SW$ 클록 사이클 중 처음 SW 클록 사이클 동안 더미 β 메트릭을 훈련(training) 해서 신뢰할 수 있는 β 메트릭 값을 얻은 뒤 그 값을 이용해 나머지 SW 클록 사이클 동안 재귀연산과 LLR 계산을 하는데, 이때 역방향 재귀의 시작점인 $k \cdot SW$ 지점에서의 트렐리스 초기상태는 동일한 값을 할당한다. 그리고 윈도우는 비트색인의 SW 만큼 위로 이동되어 연산을 반복해서 진행한다. 이 방식은 상태 메트릭 값을 저장하기 위해 단지 $2SW$ 만큼의 메모리만을 필요로 한다. 터보 복호기 하드웨어는 순방향 및 역방향 상태 메트릭과 상태천이 메트릭 저장에 가장 많은 메모리를 사용하는데 슬라이딩 윈도우 크기를 조절함으로써 사용되는 메모리양을 조절할 수 있다. 그러나 슬라이딩 윈도우의 크기는 복호 성능에 영향을 미치기 때문에 적정하게 결정되어야 한다. 그림 3은 슬라이딩 윈도우 방식의 타일 다이어그램을 나타내었다.

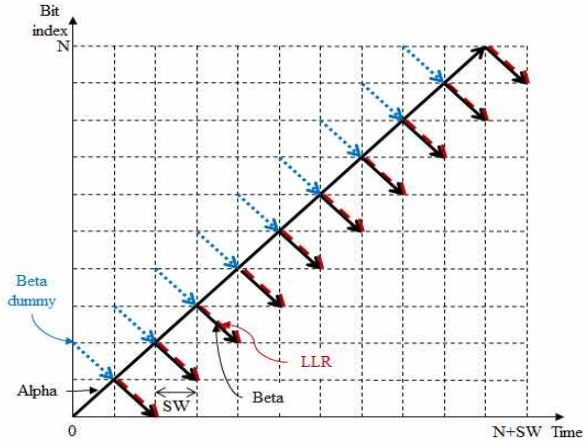

그림 3. 슬라이딩 윈도우방식의 타일 다이어그램
Fig. 3. Tile diagram of sliding window scheme.

2. 복류 복호기법

복호 지연시간을 줄이기 위한 복류 복호기법은 [5]에서 제안되었다. 이와 같은 구조에서는 데이터 블록이

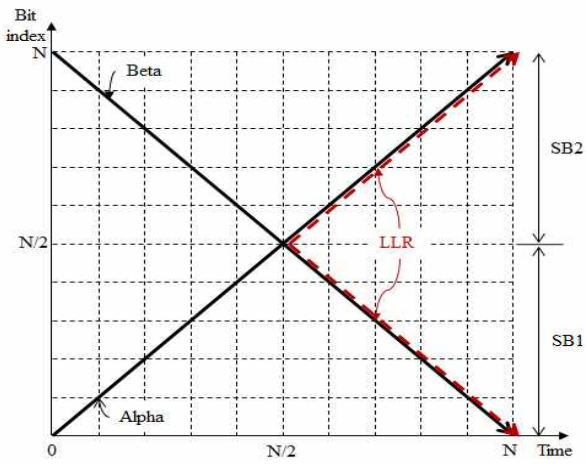

그림 4. 복류방식의 타일 다이어그램
Fig. 4. Tile diagram of double flow scheme.

두 개의 부 블록 ($SB1, SB2$)으로 나뉘어서 복호가 진행된다. $1 \sim N/2$ 클록 사이클 동안 $SB1$에서는 α 메트릭이 순방향으로 계산되는 동시에 $SB2$에서는 β 메트릭이 역방향으로 계산된다. 나머지 클록 사이클동안 $SB1$에서는 β 메트릭이 역방향으로 계산되는 동시에 $SB2$에서는 α 메트릭이 순방향으로 계산되며 $N/2$ 번째 클록 사이클 이후로는 한 클록 사이클 당 두 개의 메시지 비트에 대한 LLR을 동시에 구할 수 있다. 그림 4는 복류 복호 방식의 타일 다이어그램을 나타내었다. 복류 복호방식은 α 메트릭과 β 메트릭을 순차적으로 연산하는 직렬 복호 방식에 비해 복호 지연시간을 반으로 줄일 수 있는 장점을 가진다. 그러나 재귀 연산 유닛(recursive unit)이 직렬 복호 방식에 비해 두 배가 필요하게 된다.

3. 셔플 복호기법

병렬 터보 복호 기법(parallel turbo decoding)[9]은, 모든 시간에 있어서 두 개의 요소 복호기가 동시에 병렬로 동작한다. 매 반복이 끝날 때 마다 각각의 요소 복호기는 다른 요소 복호기로 외부 정보를 보내고 이를 다음 반복 시에 사전확률로 사용한다. 병렬 터보 복호기법의 개선된 형태인 셔플 복호 방식(shuffled iterative decoding)은 논문 [6]에서 제안되었다. 병렬 터보 복호기법은 직렬 복호 방식에 비해 복호 지연시간을 반으로 줄일 수 있지만, 외부 정보가 한 회의 복호 과정이 끝나기 전에는 사용될 수 없다. 반면 셔플 터보 복호기법의 경우에는 반복 복호 과정 중에도 다른 요소 복호기에서 업데이트 되어 이용 가능한 외부 정보를 사용할 수 있으므로 결과적으로 좀 더 빠른 복호 수렴 결과를 얻을 수 있다.

Ⅳ. 제안된 터보 복호 기법

본 논문에서는 앞서 소개한 세 가지 복호 방식의 장점을 이용하여 새로운 터보 복호 기법을 제안한다. 그림 5는 제안된 복호 기법의 요소 복호기당 타일 다이어그램을 나타내고 그림 6에서 요소 복호기의 복호과정을 블록 다이어그램으로 나타내었다. 제안된 복호기법의 기본적인 구조는 복류 복호, 슬라이딩 윈도우, 그리고 셔플 복호 방식의 혼합 형태이다. 복류 복호 방식에 의해 전체 데이터 블록을 $SB1$과 $SB2$로 나누어 동시에 복호를 진행하는데 이때 슬라이딩 윈도우기법을 사용하

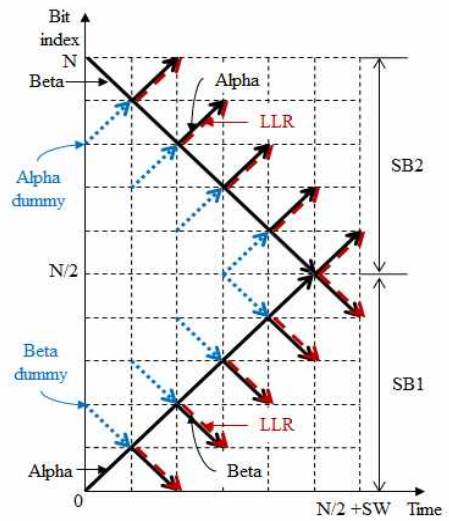

그림 5. 제안된 복호 기법의 타일다이어그램
Fig. 5. Tile diagram of proposed decoding scheme.

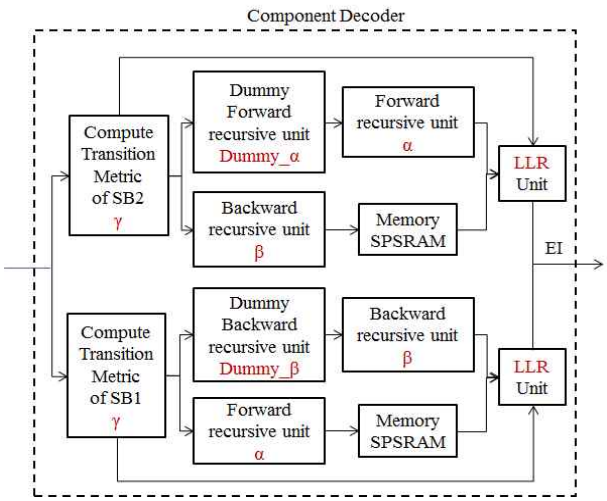

그림 6. 요소 복호기의 복호과정 블록 다이어그램
Fig. 6. Block diagram of decoding process at component decoder.

기 때문에 메시지비트의 LLR 계산이 비트색인에 따라 순차적으로 진행되지 않는다. $SB1$에서는 $k \cdot SW$ 번째 비트색인에서 시작해서 역방향으로 계산되고 SW 만큼 위로 이동되어 연산을 진행한다. $SB2$에서는 이와는 반대로 $(N/SW-k) \cdot SW$ 번째 비트색인에서 시작해서 순방향으로 계산되고 SW 만큼 아래로 이동되어 연산을 진행한다. 여기서 $k=1,2,...,N/2SW$이다. 복호 순서가 바뀌었기 때문에 이에 적용되는 외부 정보의 업데이트 규칙도 기존의 방법과는 달라져야 한다. 결과적으로 제안된 복호 기법은 그림 5에서 보는 것처럼 1회 반복에 필요한 클록 사이클이 $N/2+SW$로 줄어든다. 제안된 터보 복호 기법의 전체 블록 다이어그램은 그림 7에

나타나있다.

$\widetilde{\boldsymbol{m}}=(\widetilde{m}_1,\widetilde{m}_2,...,\widetilde{m}_N)$를 입력 정보 $\boldsymbol{m}=(m_1,m_2,...,m_N)$이 인터리빙 규칙 $\widetilde{m}_j=m_{\pi(j)}$에 의해 인터리빙된 시퀀스라고 하자. 여기서 $j=1,2,...,N$이다. 마찬가지로 디인터리빙 규칙은 $m_j=\widetilde{m}_{\pi^{-1}(j)}$와 같다. α_j^d, β_j^d, γ_j^d를 요소 복호기 d에 의해 연산된 m_j에 대한 메트릭으로 정의한다. 여기서 $d=1,2$이다. 그리고 i번째 반복에서 요소 복호기 d에 의해 얻어진 m_j의 외부 정보를 $L_e^{d,i}(m_j)$ 라고 정의한다. 제안된 방법에서는 두 개의 요소 복호기가 각각 두 개의 LLR 계산을 동시에 진행한다. 게다가 각각의 요소 복호기는 한 회의 복호 도중에도 이용 가능한 외부 정보를 다른 요소 복호기로 보낼 수 있다. T_j^d를 요소 복호기 d에서 j번째 메시지비트의 LLR 계산을 시작하는 시간이라 하자. 이때 $T_j^1=T_{N-j+1}^1=T_j^2=T_{N-j+1}^2$로 나타낼 수 있다.

i번째 반복 중 요소 복호기 1에서 복호과정을 고려해보면 시간 $T_j^1=T_{N-j+1}^1$ 이후 β_j^1와 α_{N-j+1}^1가 업데이트 되어야하고 이를 위해 γ_j^1와 γ_{N-j+1}^1값이 필요하다. 여기서 m_j와 m_{N-j+1}각각을 복호하는데 있어서 두 가지 경우가 발생된다. m_j에 대해 살펴보면, 요소 복호기 2의 메시지비트 m_j에 대한 외부 정보 $L_e^{2,i}(m_j)$가 복호기 1로 아직 전달되지 않은 경우와 이미 전달된 경우가 존재한다. 첫 번째 경우 $L_e^{1,i}(m_j)$ 계산을 위해 $i-1$번째 반복에서 업데이트 되고 저장된 γ_j^1이 사용되고 β_j^1를 업데이트한 후 LLR 연산을 통해 $L_e^{1,i}(m_j)$을 만들어내고, 두 번째 경우에는 $L_e^{2,i}(m_j)$가 바로 γ_j^1를 업데이트 하는데 사용되고 β_j^1를 업데이트한 후 LLR 연산을 통해 $L_e^{1,i}(m_j)$를 만들어낸다. 이때 j는 첫 번째 경우에는

$$\begin{aligned}&\left\{\left\lfloor\frac{j}{SW}\right\rfloor\cdot SW<\pi^{-1}(j)\le j\right\}\\ \text{or}\ &\left\{\left\lceil\frac{j}{SW}\right\rceil\cdot SW<\pi^{-1}(j)\le\left\lfloor\frac{N-j+1}{SW}\right\rfloor\cdot SW\right\}\\ \text{or}\ &\left\{N-j+1\le\pi^{-1}(j)\le\left\lceil\frac{N-j+1}{SW}\right\rceil\cdot SW\right\}\end{aligned}\quad(5)$$

를 만족하며 두 번째 경우에는

$$\begin{aligned}&\left\{1\le\pi^{-1}(j)\le\left\lfloor\frac{j}{SW}\right\rfloor\cdot SW\right\}\\ \text{or}\ &\left\{j<\pi^{-1}(j)\le\left\lceil\frac{j}{SW}\right\rceil\cdot SW\right\}\\ \text{or}\ &\left\{\left\lfloor\frac{N-j+1}{SW}\right\rfloor\cdot SW<\pi^{-1}(j)<N-j+1\right\}\\ \text{or}\ &\left\{\left\lceil\frac{N-j+1}{SW}\right\rceil\cdot SW<\pi^{-1}(j)\le N\right\}\end{aligned}\quad(6)$$

를 만족한다. 여기서 $\lfloor x\rfloor$는 x보다 크지 않은 가장 큰 정수를 나타내고 $\lceil x\rceil$는 x보다 작지 않은 가장 작은 정수를 나타낸다.

한편 m_{N-j+1}에 대해서도 위와 동일한 두 가지 경우가 존재하며 각 경우에 대해 위와 유사한 복호기 동작 규칙이 적용된다. 따라서 m_j와 m_{N-j+1}에 대한 γ_j^1와 γ_{N-j+1}^1값이 둘 다 i번째 반복과정 중에 업데이트 가능한 경우, 둘 중 하나만 업데이트 가능한 경우, 둘 다 업데이트 가능하지 않아서 $i-1$번째 반복에서 저장된 값을 사용하는 경우 이렇게 세 가지 경우가 존재하며, 각 경우의 복호기 동작규칙은 앞서 설명한 바와 같다. 요소 복호기 2에 대한 γ_j^2와 γ_{N-j+1}^2도 요소 복호기 1에서와 같은 규칙에 의해 동작한다.

그림 8에는 앞서 살펴본 규칙에 의해 동작하는 각각의 요소 복호기에 대한 연산과정과 비트색인을 세분화해서 예시하였다. 반복 복호 도중 외부 정보의 전달 여부는 각각의 시스템에서 사용되는 인터리빙, 디인터리

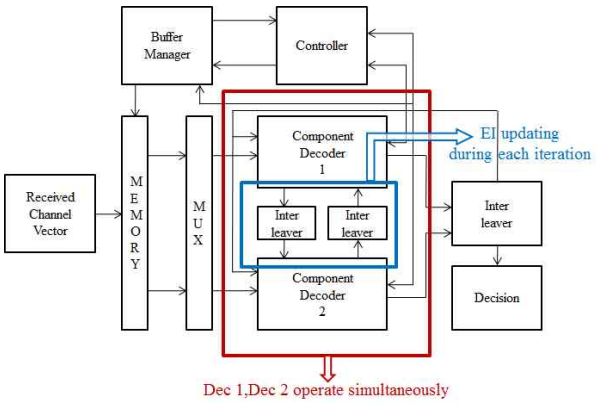

그림 7. 제안된 복호 기법의 복호기 블록 다이어그램
Fig. 7. Block diagram of proposed decoding scheme

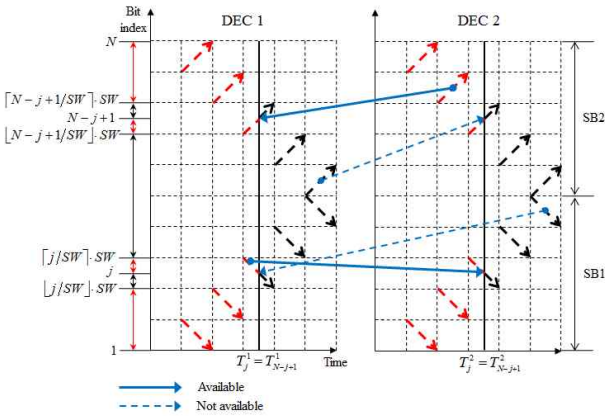

그림 8. 제안된 기법의 복호과정 예시
Fig. 8. Example of proposed decoding scheme.

빙 규칙과 앞서 살펴본 업데이트 규칙에 의해 좌우된다. 그림 8에 나타낸 예시는 다음과 같은 경우에 해당한다. 요소 복호기 1에서 j번째 비트에 해당하는 외부정보는 T_j^1까지 요소복호기 2에서 새롭게 업데이트 되지 못한 반면 $N-j+1$번째 비트에 해당하는 외부정보는 T_{N-j+1}^1이전에 요소복호기 2에서 이미 업데이트 되어있다. 또한 요소복호기 2에서는 j번째 비트에 해당하는 외부정보가 T_j^2 이전에 요소복호기 1에서 이미 업데이트 되었고 $N-j+1$번째 비트에 해당하는 외부정보는 T_{N-j+1}^2까지 새롭게 업데이트 되지 못했다. 단, 앞서 언급한 바와 같이 $T_j^1=T_{N-j+1}^1=T_j^2=T_{N-j+1}^2$이다. 이때 요소 복호기 1과 요소 복호기 2는 동시에 동작한다. 즉 그림 8의 두 타일다이어그램은 동일한 시간 축을 사용하고 있다. 한편 그림 8의 좌측에는 식(5)와 식(6)에 해당하는 비트색인을 표시하였다.

V. 성능 비교 및 모의실험 결과

이번 장에서는 제안된 터보 복호 기법과 다른 기법들의 성능을 비교한다. 표 1에는 각 복호기법의 반복복호당 소요되는 클록 사이클 수와 상태 메트릭 값을 저장하기 위해 요구되는 메모리를 나타냈다. 다음으로 컴퓨터 모의실험을 통해 각 복호 기법별 BER 성능을 얻었다. 실험에 사용한 터보부호는 IEEE802.16e 표준의 부호율 1/2인 $(15,13)_{(8)}$ 이중이진 터보 부호이고[10], 변조는 QPSK, 채널은 AWGN채널을 사용하였다. 이때 메시지 블록의 크기는 480비트로 하였으며 복호기에 사용한 슬라이딩 윈도우의 크기는 12, 24, 40으로 가변하며 모의실험을 수행하였다. 한편 요소 복호 알고리즘으로는 계산 복잡도가 낮은 Max-Log-MAP을 사용하였고,

표 1. 터보 복호 기법의 일반적인 성능 비교
Table 1. Comparison of complexities for various turbo decoding schemes.

	Clock cycles /iteration	Number of required memories for storing state metrics
Serial decoding	4N	N
Double flow	2N	N
Serial + Sliding window	2N+2SW	2SW
Double flow + Sliding window	N+2SW	4SW
Proposed decoding	N/2 +SW	8SW

이에 따른 성능저하를 보상하기 위한 스케일링 팩터는 0.75를 사용하였다. 그림 9에는 제안된 복호 기법의 반복 복호별 BER 성능을 나타내었고, 각 반복별로 슬라이딩 윈도우의 크기를 다르게 하여 성능을 비교하였다. 슬라이딩 윈도우 크기가 작을수록 성능이 안 좋게 나오는데 그 이유는 메트릭 값이 충분히 훈련되지 못해서 신뢰도가 낮은 값을 갖기 때문이다. 한편 IEEE802.16e 표준의 성능 요구조건에 따르면 메시지 블록길이 480비트, QPSK 변조, AWGN 채널의 경우 BER=10^{-6}를 만족하기 위해 필요로 하는 SNR이 E_b/N_0=2.9dB 이하이다[11]. 그림 10에는 IEEE802.16e 표준의 성능 요구조건을 만족하는 최소 반복복호를 통해 얻어지는 BER 성능을 도시하였다. 그림에서 볼 수 있는 것처럼, 제안된 복호 기법의 6회 반복일 때의 BER 성능은 직렬 복호기법의 4회 반복일 때의 성능과 거의 유사하다. 그러나 표 1에서 볼 수 있는 것처럼 직렬 복호기는 한회 반복 복호당 $4N$의 클록 사이클이 필요한 반면 제안된 방법은 $N/2+SW$ 만큼의 클록 사이클만 필요하다. 따라서 총 필요한 클록 사이클은 각각 $16N$과 $3N+6SW$이 된다. 복류복호를 사용함으로써 SW는 최대 $N/2$까지 될 수 있는데 이러한 경우에도 직렬 복호 보다 적은 $6N$의 클록 사이클이 필요하다. 이로서 동일한 조건에서 요구되는 BER 성능을 비교하였을 때 제안된 복호 방법이 기존의 복호 방법보다 낮은 복호 지연시간을 가지는 것을 볼 수 있다. 표 2에는 IEEE802.16e 표준의 성능을 만족하는 최소 반복복호 횟수와 총 소요되는 클록 사이클을

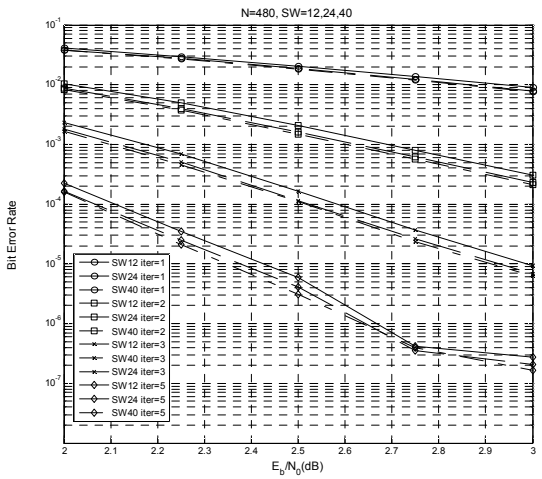

그림 9. 제안된 터보 복호기법의 슬라이딩 윈도우크기 및 반복 복호 횟수별 BER성능
Fig. 9. BER performance of proposed turbo decoding scheme.

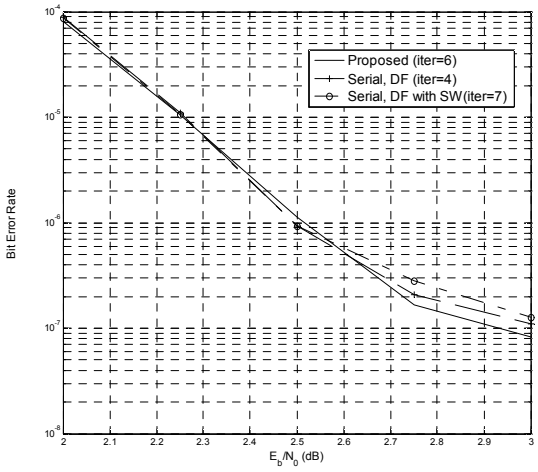

그림 10. 제안된 기법, 직렬 복호 기법, 복류 복호기법의 BER성능 비교

Fig. 10. BER performances of serial, double flow and proposed turbo decoding schemes

표 2. IEEE 802.16e 표준의 성능 요구조건을 만족하는 최소 반복 복호 횟수와 총 소요되는 클록 사이클 비교

Table 2. Comparison of the numbers of iterations and clock cycles needed to meet the performance requirement of IEEE802.16e turbo codes.

	Number of Required iterations	Total number of required clock cycles
Serial decoding	4	16N
Double flow	4	8N
Serial + Sliding window	7	14N+14SW
Double flow + Sliding window	7	7N+14SW
Proposed decoding	6	3N+6SW

비교했다.

제안된 터보 복호기법은 지연시간을 줄이기 위해 사용되지만, 사용되는 메모리의 양도 역시 줄일 수 있다. 표 1에서 정리된 것처럼 복호기법 별로 필요한 메모리 양을 나타내었는데 만일 슬라이딩 윈도우 크기가 전체 블록 N의 1/8보다 작다면 메모리 사용에 있어서도 기존의 직렬 복호 및 복류복호에 비해 장점을 보인다.

VI. 결 론

본 논문에서는 기존의 여러 복호 기법에 대해 간략하게 소개하고 새로운 터보 복호 기법을 제안하였다. 제안된 터보 복호기법은 기본적으로 복류 복호, 슬라이딩 윈도우, 셔플 복호 기법의 장점을 조합한 형태이다. 각 복호기법별로 연산에 소요되는 클록 사이클 수, 메모리 사용량 등을 BER 성능과 함께 비교, 분석한 결과 제안된 기법이 기존의 기법에 비해 고속의 데이터를 처리하는데 유리하며 적절한 크기의 슬라이딩 윈도우를 사용함으로써 메모리 사용량을 줄일 수 있음을 보였다. 본 논문에서 제안한 터보 복호기법은 저 전력, 저 면적의 고속 터보 복호기 하드웨어 설계에 유용하게 활용될 것으로 기대된다.

참 고 문 헌

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error-correction coding and decoding: turbo-codes,” in *Proc. ICC* ’93, vol. 2, pp. 1064-1070, May 1993.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inf. Theory,* pp. 284-287, Mar. 1974.

[3] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, “Soft output decoding algorithms for continuous decoding of parallel concatenated convolutional codes,” in *Proc. ICC* ’96, pp. 112-117, June 1996.

[4] Y. Zhang and K. Parhi, “Parallel turbo decoding,” in *Proc. ISCAS* ’04, vol. 2, pp. II-509-12, May 2004.

[5] C. Schurgers, F. Catthoor, and M. Engels, “Optimized MAP turbo decoder,” in *Proc. SiPS* ’00, pp. 245-254, Oct. 2000.

[6] J. Zhang and M. Fossorier, “Shuffled iterative decoding,” *IEEE Trans. Commun.,* vol. 53, no. 2, pp. 209-213, Feb. 2005.

[7] 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, June 1995.

[8] J. Vogt and A. Finger, “Improving the max-log-MAP turbo decoder,” *Electron. Lett.*, vol. 36, no. 23, pp. 1937-1939, Nov. 2000.

[9] D. Divsalar and F. Pollara, “Multiple turbo codes for deep-space communications,” *JPL TDA Prog. Rep.*, 71-78, May 1995.

[10] C. Berrou, C. Douillard, and M. Jezequel, “Multiple parallel concatenation of circular recursive systematic convolutional (CRSC) codes,” *Ann. Telecommun.*, vol. 54, pp. 166-172,

Mar.- Apr. 1999.
[11] WiMAX Forum Mobile System Profile: Release 1.0 Approved Specification, Revision 1.4.0, 2007.

저 자 소 개

최 재 성(정회원)
2010년 중앙대학교 전자전기 공학과 학사.
2010년～현재 중앙대학교 전자 전기공학과 석사과정.
<주관심분야 : 오류정정부호, 정보이론>

신 준 영(정회원)
2008년 중앙대학교 전자전기 공학과 학사.
2010년 중앙대학교 전자전기 공학과 석사.
2010년～현재 SK텔레콤 네트워크 기술원.
<주관심분야 : 오류정정부호, 정보이론>

이 정 우(정회원)-교신저자
1994년 서울대학교 전기공학과 학사.
1996년 서울대학교 전기공학과 석사.
2003년 University of Illinois at Urbana-Champaign, 전기공학과 박사.
2003년～2004년 University of Illinois, Research Associate.
2004년-현재 중앙대학교 전자전기공학부 교수.
<주관심분야 : 통신시스템, 오류정정부호, 정보이론, 무선통신, 신호처리>