# 슬라이딩 윈도우 방식의 터보 복호화기의 구조 및 성능

# The Structure and Performance of Turbo decoder using Sliding-window method

심 병 효* 구 창 설* 이 봉 운*
Shim, Byonghyo, Koo, Changsul, Lee, Bongwoon

**ABSTRACT**

Turbo codes are the most exciting and potentially important development in coding theory in recent years. They were introduced in 1993 by Berrou, Glavieux and Thitimajshima,[1] and claimed to achieve near Shannon-limit error correction performance with relatively simple component codes and large interleavers. A required Eb/N0 of 0.7dB was reported for BER of $10^{-5}$ and code rate of 1/2.[1] However, to implement the turbo code system, there are various important details that are necessary to reproduce these results such as AGC gain control, optimal wordlength determination, and metric rescaling. Further, the memory required to implement MAP-based turbo decoder is relatively considerable. In this paper, we confirmed the accuracy of these claims by computer simulation considering these points, and presented a optimal wordlength for Turbo code design. First, based on the analysis and simulation of the turbo decoder, we determined an optimal wordlength of Turbo decoder. Second, we suggested the MAP decoding algorithm based on sliding-window method which reduces the system memory significantly. By computer simulation, we could demonstrate that the suggested fixed-point Turbo decoder operates well with negligible performance loss.

주요기술용어 : Turbo Code (터보 코드), sliding window MAP decoder(슬라이딩 윈도우 방식의 최대사후확률의 복호화기)

## 1. 서 론

터보부호는 신호대 잡음비 측면에서 Shannon의 한계에 근접하는 놀라운 성능을 갖고 있는 부호이다. 1993년 Berrou[1]에 의하여 발표된 터보부호의 알고리듬은 부호이론 분야에 큰 반향을 불러 일으켰으며 이후 이에 대한 많은 연구가 이루어졌다.[2-7] 기본적으로 터보부호는 부호화시에 같은 정보원을 두 개이상의 부호기를 사용하여 부호화한다. 보통 부호기는 동일한 것을 사용하며 첫 번째 부호기는 원 신호열 순서대로 부호화를, 두 번째 부호기는 인터리버를 통과시켜서 흩뜨려진 순서로 부호화를 수행한다. 따라서 송수신기에서 동일한 인터리버를 사용해야 정확한 복호화가 이루어질 수 있다. 수신부에서는 연성출력(soft-output)이 가능한 복호화기를 사용한다. 대표적인 것으로 최대사후(MAP: Maximum A Posteriori) 방식의 복호화기와 연성출력 비터비 복호화기(SOVA:

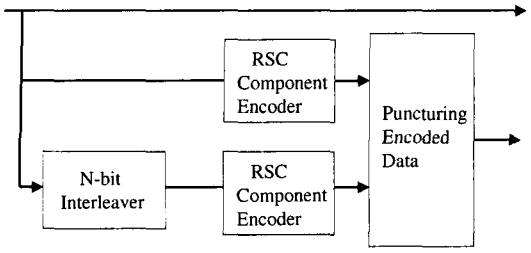
* 공군사관학교 전자공학과 교수

Soft Output Viterbi Algorithm)가 있다. SOVA방식은 복호화가 비교적 간단하나 MAP방식에 비해서 상대적으로 성능이 낮다. 이와 같은 연성 출력 복호화기를 두 개이상 병렬(또는 직렬)로 연접하여 하나의 복호화기의 출력을 다른 복호화기의 복호화를 돕기 위한 외부정보로 사용하고 이러한 과정을 반복적으로 수행하게 된다. 이와 같은 방식으로 복호화를 반복 수행함에 따라서 복호성능이 점차로 좋아지며 낮은 SNR에서도 뛰어난 복호성능을 갖는다. 이처럼 간섭이나 잡음이 많은 채널에서도 우수한 복호성능을 갖고 또한 자체적 암호화 기능을 내장하고 있어 부가적인 암호화가 필요 없이 인터리버의 주기적 변형만으로 제 3자의 도청을 방지할 수 있는 점은 군사통신에서도 유용하게 사용될 수 있음을 보여주고 있다. 터보 코드의 이와 같은 우수한 복호화 성능의 원인과 복호화 기법 자체에 대해 전세계적으로 활발한 연구가 진행되어 왔으며 근래 민간 통신뿐 아니라 여러 군사 통신시스템으로의 응용이 여러 연구기관에서 이루어지고 있다.[2-9] 이러한 노력의 결과로 터보코드는 차세대 이동통신의 표준인 IMT-2000에서 데이터 전송의 표준이 되었다. 그러나 이러한 우수한 성능에도 불구하고 터보코드를 실제 시스템으로 구현하기 위해서는 여러 가지 해결해야 할 문제들이 존재한다. 첫째로, 터보코드는 반복 복호화를 수행함에 따라서 성능이 좋아지는 특성이 있으므로 요구되는 BER에 적합한 회수만큼 반복 복호화를 한 결과를 최종적 복호화기의 출력으로 사용하게 된다. 따라서 복호화 지연이 상대적으로 다른 콘벌루션 부호나 블록 부호에 비해 큰 단점이 있다. 이는 전장상황에서 실시간 응용에 제한을 줄 수 있다. 두 번째로 전방향 메트릭과 후방향 메트릭을 사용하여 연성 출력(soft-output)을 생성하므로 복호화기 내부에 메트릭을 저장하기 위한 메모리가 필요하다. 게이트 사이즈의 증가는 전력소모의 증가를 일으키고 이동성이 크고, 장시간 충전 없이 사용되어야 하는 전장상황에서 제약점으로 작용한다. 세 번째로 각 요소 부호간에 서로 랜덤한 특성을 갖기 위해서는 비교적 큰 사이즈의 인터리버를 사용해야 한다. 따라서 큰 사이즈의 패킷을 사용해야 흩뜨려진 효과를 얻을 수 있고 이로 인한 지연시간이 발생하게 된다. 터보코드의 복호화 지연을 줄이기 위한 많은 연구가 있어 왔으며 Viterbi등에 의해 제시된 슬라이딩 윈도우 방식이 메모리 크기를 줄일 수 있는 대표적인 방식으로 알려져 있다.[9] 본 논문에서는 슬라이딩 윈도우 방식의 터보 복호화기를 VLSI로 구현하기 위한 터보코드의 하드웨어 구조를 제시하고 제안한 구조에 대한 기존 방식간의 성능을 모의실험을 통하여 비교하였다. 본 논문의 구성은 다음과 같다. 본 서론에 이어 2장에서는 슬라이딩 윈도우 방식의 터보코드에 대해서 살펴보고 3장에서는 실제 시스템 구현 시 고려사항을 다루었고 4장에서는 이에 대한 모의실험을 바탕으로 일반적인 터보 코드와 슬라이딩 윈도우 방식간의 성능을 비교 분석해 보았다. 5장에서는 결론을 맺는다.

## 2. 터보 부호

### 2.1 터보 부호기

일반적인 터보부호기의 모습이 그림 1에 나타나 있다. 터보코드의 부호화기 시스템은 두 개 또는 그 이상의 부호화기를 병렬 연접시켜서 구성한다. 그림 1에 보이는 것처럼 두 개의 부호화기를 가정할 경우에 부호화 시 원 정보와 두 개의 패리티 출력이 생성되게 된다. 부호화기는 동일한 것을 사용하며 첫 번째 부호화기는 원 정보입력 순서가, 두 번째 부호화기는 인터리버를 사용하여 흩뜨려진 순서로 부호화가 수행된다.

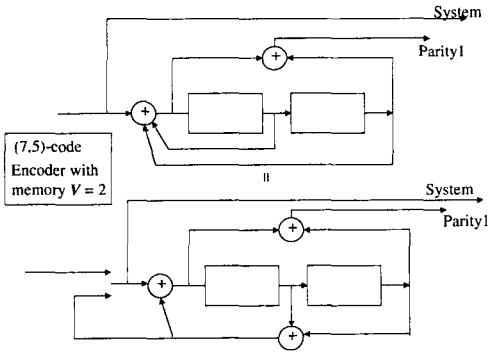



〔그림 1〕 터보 부호화기

코드률 1/2일 경우에 각 패리티는 펑쳐링(punctu-ring)이 수행되는 데 이때 변조기로 전송되는 패턴은 다음과 같다.

x[0] y1[0]
x[1] y2[1]
x[2] y1[2]
x[3] y2[3]
...

코드률 1/3일 경우에는 펑쳐링 없이 바로 시스템 출력, 패리티1, 패리티2를 전송하게 된다. 부호화기는 RSC(Recursive Systematic Code) 방식의 부호화기를 사용한다. 그림 2에 RSC방식의 부호화기가 나타나 있다. 부호기가 재귀적인 방식으로 구성되어 있기 때문에 비재귀적(Nonrecursive)부호에서와 같이 마지막 부분(tail part)에 메모리를 0으로 채우기 위하여 넣어주는 0은 의미가 없다.



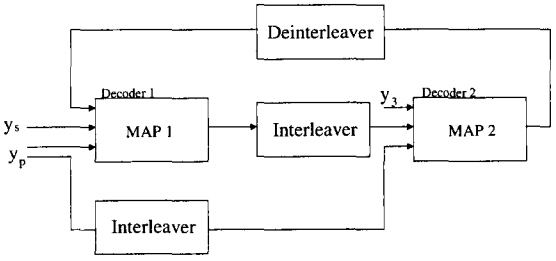

〔그림 2〕 RSC 부호화기

따라서 그림 2의 아래 부분에서와 같이 마지막 부분은 입력과 연결하지 않고 피드백 부분과 연결하여 0상태를 만들게 된다. 따라서 실제로 전송되는 패턴은 원 정보신호의 크기가 N, 코드률 1/2, 구속장길이 K=3일 경우 다음과 같다.

x[0] y1[0]
x[1] y2[1]
x[2] y1[2]
x[3] y2[3]
...
x[N-1] y2[N-1]
x[N] y1[N] <- 첫 부호화기의 tail
x[N+1] y1[N+1]
x'[N] y2[N] <- 두 번째 부호화기의
x'[N+1] y2[N+1] tail

## 2.2 터보 복호기

터보 복호기의 구조가 그림 3에 나타나 있다. 터보 복호기는 인터리버, 디인터리버와 두 개의 요소 복호화기를 사용하여 이루어진다. 첫 번째 복호화기는 시스템 입력과 첫 번째 부호화기의 패리티를 복호화한다. 두 번째 복호화기는 시스템 입력과 인터리버를 통과하여 흩뜨려진 패리티를 복호화한다. 흩뜨려진 순서로 복호화를 수행해야 하기 때문에 시스템 입력을 흩뜨리기 위하여 인터리버가 사용된다. 이 후 다시 첫 번째 복호




〔그림 3〕 터보 복호화기의 구조

화기에서 반복적 복호화를 수행 시에 다시 원 순서로 복호화를 수행하기 위해서 디인터리버가 사용된다.

## 2.3 터보 복호기의 요소 복호기 (MAP복호화기)

터보 복호기의 요소 복호기로는 보통 MAP방식과 SOVA방식이 사용된다. SOVA방식은 구현이 비교적 간단한 장점이 있으나 성능이 MAP방식에 비해서 좋지 못한 단점이 있다. 이러한 이유로 MAP방식을 사용함에 있어서 하드웨어의 복잡도를 줄이기 위한 여러 방식들이 제시되었는데 log-MAP, sub-MAP등이 잘 알려져 있다. log-MAP방식은 원 MAP 알고리즘에 로그를 취해서 곱셈과 나눗셈을 덧셈과 뺄셈으로 바꾸었으며 log(exp( · ))함수를 min과 log뺄셈 항으로 바꾼 방식이다. sub-MAP방식은 log뺄셈 항을 생략하고 min함수로 근사화한 방식인데 log-MAP방식에 비해 성능이 떨어진다. log-MAP방식에서도 유도방식에 따라서 조금씩 다른 형태의 식으로 표현된다. 전체적으로는 수신신호로부터 브랜치 메트릭을 구하는 과정, 순방향 메트릭을 구하는 과정, 역방향 메트릭을 구하는 과정, 그리고 LLR(Log-likelihood Ratio)값을 구하는 과정으로 이루어져 있다. log-MAP알고리듬은 다음과 같이 표현된다.

(1) 수신된 신호와 외부 정보(extrinsic information)로부터 브랜치 메트릭을 구한다.

$$D_i(R_k, m) = \frac{2}{\sigma^2}(x_k i + y_k Y_k^i(m))$$

(2) 순방향 상태 메트릭값을 구한다.

$$A_k^i(m) = D_i(R_k, m) + E_{j=0}^{1}[A_{k-1}^{j}(S_b^j(m))]$$

(3) 역방향 상태 메트릭값을 구한다.

$$B_k^i(m) = E_{j=0}^{1}[B_{k+1}^{j}(S_f^i(m)) + D_j(R_{k+1}, S_f^i(m))]$$

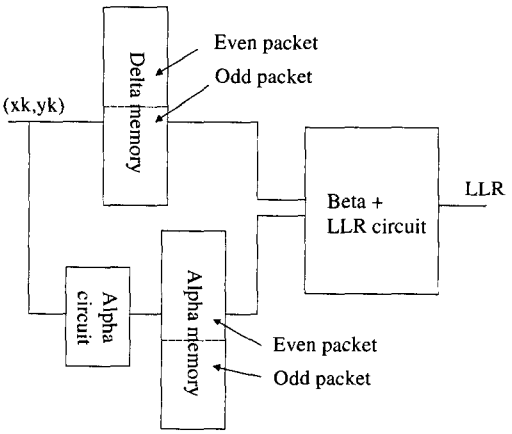
(4) LLR값을 구한다.

$$L(d_k) = E_m^{M-1}[A_k^1(m) + B_k^1(m)] - E_m^{M-1}[A_k^0(m) + B_k^0(m)]$$

## 2.4 슬라이딩 윈도우 방식의 MAP 복호화 기법

다 절에 나타난 식 (1)-(4)를 살펴보면 전방향 메트릭을 구하기 위해서는 패킷의 앞 부분부터 데이터가 들어와야 한다. 역방향 메트릭은 패킷의 맨 끝 부분부터 구해져야 하므로 한 패킷분량의 브랜치 메트릭 메모리가 저장되어야 한다. 또한 연속적인 MAP복호화를 수행하기 위해서는 다음 브랜치 메트릭을 저장하기 위하여 한 패킷 분량의 메모리가 더 필요하다.

LLR은 역방향 메트릭이 구해지는 동시에 계산된다고 할 때 역시 순방향 메트릭이 필요하므로 순방향 메트릭을 위한 메모리가 필요하다. 따라서 보통의 경우 패킷사이즈를 N으로 두었을 때 MAP복호화기 내에서 필요한 메모리는 아래 그림 4와 표 1을 참조하면 12N만큼이 필요하다. 구속장 K=3(4개의 상태)



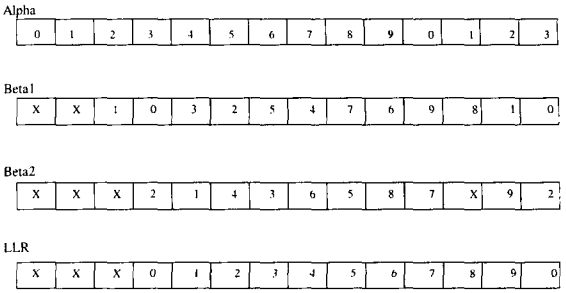

〔그림 4〕 일반적인 방식의 MAP복호화기의 하드웨어 구조

〔표 1〕 MAP복호화기에서 필요한 메모리

| 브랜치 메트릭을 위한 메모리 | 전방향 메트릭을 위한 메모리(K=3) | 총 메모리 사이즈 |
|---|---|---|
| 2 · 2N개 · 8bit/개 | 4 · 2N개 · 8bit/개 | 6 · 2N개 · 8bit/개 |
| 4N byte | 8N byte | 12N byte |

의 코드에서 패킷사이즈 N이 1k의 크기를 갖고 각 메트릭을 8비트로 저장하는 경우, 12kbyte만큼의 메모리가 필요하게 된다. 따라서 실제 시스템으로 구현시에 메모리 오버헤드가 큰 단점이 있다. 따라서 이를 해결할 필요가 있다.

슬라이딩 윈도우 방식은 콘벌루션 부호의 특성을 이용하여 아주 작은 메모리 사이즈로 MAP 복호화기를 구현할 수 있게 한 방식이다. 콘벌루션 부호에서는 복호화시 필요한 경로를 저장하고 있어야 하므로 시스템 구현시 경로 메모리의 비용이 복호화기의 총 비용 중 가장 큰 비중을 차지한다. 경로가 긴 경우 복호 지연시간과 메모리의 크기가 커지기 때문에 가급적 에러성능의 손실이 크지 않은 범위내에서 성능과 경로 메모리간의 절충이 요구된다. 일반적으로 히스토리의 길이가 5K정도 이상이 되면 성능이 수렴하는 것으로 알려져 있다.[10] 터보 부호화 시에 부호화기의 구속장은 K=3 또는 4의 것을 많이 사용하므로 5K를 가정할 때 실제 사용되기 위한 메트릭을 구하기 위한 경로의 크기를 20이상으로 두면 대개 경로가 수렴하게 된다. 이 경로를 MAP복호화기에서 윈도우의 길이 L으로 정의한다. 이때 그림 5의 상태 다이아그램에 나타난 순서대로 복호화를 수행한다. 한 패킷의 데이터가 10개의 윈도우 블록(0 , ..., 9)으로 쪼개진 것으로 가정하자. 이때 시간축에서 정방향으로 브랜치 메트릭과 순방향 메트릭이 계산된다. 첫 윈도우와 두 번째 윈도우가 채워지는 사이클에서는 역방향 메트릭은 계산되지 않는다. 두 번째 윈도우




〔그림 5〕 슬라이딩 윈도우 방식의 타이밍 다이아그램

(그림에서 1로 씌여진 윈도우)가 채워진 후 순방향 메트릭이 세 번째 윈도우를 채우기 시작하는 순간 첫 번째 역방향 메트릭 프로세서가 두 번째 윈도우의 마지막 부분부터 역방향 메트릭을 구하기 시작한다. 역방향 메트릭의 첫 상태는 알지 못하므로 모든 상태에 동일한 초기 메트릭을 할당하여 역방향 프로세서를 진행시킨다. 이 과정이 윈도우의 길이만큼 수행될 때 역방향 메트릭은 대개 수렴되게 된다.

따라서 역방향 메트릭 프로세서가 두 번째 윈도우를 다 통과하고 첫 번째 윈도우로 들어서는 순간부터는 이미 저장되어 있는 순방향 메트릭과 계산되는 역방향 메트릭이 LLR출력을 생성해내게 된다. 두 번째 역방향 메트릭 프로세서는 첫 번째 프로세서보다 한 윈도우 길이만큼의 시간을 더 기다린 후 순방향 메트릭이 세 번째 윈도우(그림에서 2로 표시된 윈도우)를 모두 수행한 시점부터 시작된다. 세 번째 윈도우에서부터 거꾸로 역방향 메트릭을 구해나가면서 하나의 윈도우만큼 수렴과정을 겪은 후 두 번째 윈도우를 수행할 때부터 저장되어 있는 순방향 메트릭과 함께 LLR 출력을 생성하게 된다. 따라서 일반적인 MAP 복호화기의 하드웨어처럼 한패킷(그림의 경우, 10개의 윈도우)이 지난 후 출력이 나오는 것이 아니라, 2개 윈도우의 지연 후 LLR 출력이 나오게 된다. 따라서 복호화 지연시간이 크게 줄어들게 된다. 또한 브랜치 메트릭과 순방향 메트릭을 각각 4개

〔표 2〕 MAP 복호화기에서 일반적 방식과 슬라이딩 윈도우 방식의 메모리 사이즈 비교

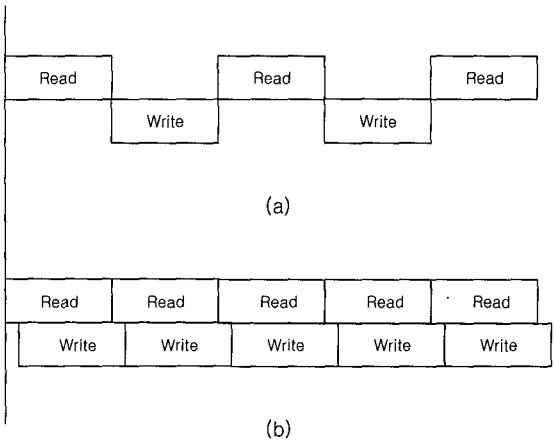
| 일반적 방식 | 슬라이딩 윈도우 방식 |
|---|---|
| 6 · 2N개 · 8bit/개 | 6 · 4L개 · 8bit/개 |
| 12N byte | 24L byte |
| 12 kbyte (N=1k) | 768 byte (L=32) |

윈도우 사이즈만큼 가지고 있으면 되므로 각각 두개의 패킷 사이즈만큼 가지고 있어야 하는 경우와 비교해 볼 때 메모리를 크게 절약할 수 있다. 또한 이 경우는 고정된 메모리로 구현될 수 있기 때문에 가변적인 패킷사이즈를 갖는 경우에도 시스템의 변경이 없이 쉽게 사용할 수 있는 장점이 있다. 윈도우 길이 L을 32로 설정한 경우, 패킷사이즈가 1k일 때 일반적 방식과 슬라이딩 윈도우 방식의 메모리는 표 2와 같다.

윈도우의 크기를 L을 32로 둔 경우에 슬라이딩 윈도우 방식은 고정적으로 768 byte만큼의 메모리가 필요한 반면에 일반적 방식에서는 패킷에 비례하는 만큼의 메모리가 필요하다. 패킷의 크기 N이 1k인 경우에 12kbyte가 필요하므로 두 방식을 비교하면 필요한 메모리가 훨씬 줄어드는 것을 쉽게 파악할 수 있다. 또한 가변적 메모리를 사용하는 시스템에서도 부가적인 회로가 필요 없기 때문에 구현이 용이하다.

두 방식을 전체적으로 비교하면 일반적 방식에서는 순방향, 역방향, LLR 메트릭 프로세서가 각각 하나씩 사용되고 슬라이딩 윈도우 방식에서는 순방향, LLR 메트릭 프로세서는 각각 한 개, 역방향 메트릭 프로세서는 두 개가 필요하다.

그러나 메트릭 프로세서가 덧셈기 두 개와 E-함수 블럭 하나로 구성되므로 게이트 크기의 증감을 고려해볼 때 슬라이딩 윈도우 방식에서 메모리의 절약으로 인한 이득이 크다. 지연시간의 경우에도 일반적인 구성에서는 한 패킷을 다 읽은 후에야 LLR 출력이



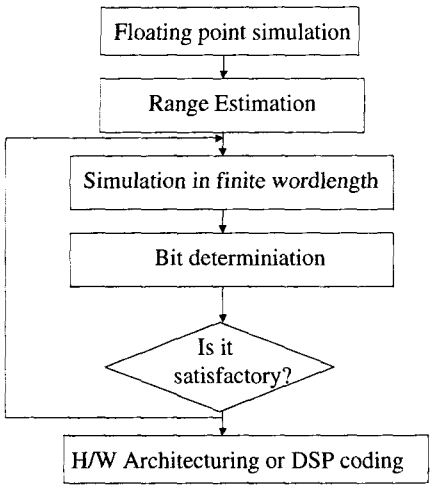

〔그림 6〕 기존의 방식과 슬라이딩 윈도우 방식의 복호화 타이밍

생성되기 시작해서 한 패킷구간동안 씌여지게 되므로 모두 마칠 때까지 기다려야 하는데 반해 슬라이딩 윈도우 방식에서는 2개의 윈도우만큼의 지연만이 이루어지므로 연속적인 복호화가 가능하다. 따라서 전체적인 터보코드의 입출력 속도가 대략 두 배정도 빨라지게 된다.

## 3. 터보 복호화기 시스템

### 3.1 터보 복호화기의 단어길이

터보 디코더를 시스템으로 구현하기 위해서는 성능과 구현의 비용을 고려하여 최적의 단어길이(optimal wordlength)를 설정해주어야 한다. 이를 위해서는 우선 각 내부 변수들의 범위를 모의실험을 통하여 알아낸 후 변수의 범위에 맞게 단어길이를 설정해주어야 한다. 그리고 이의 검증을 위해서 각 단어길이에 따른 모의실험을 통해서 BER등을 구한 후 주어진 조건에 부합할 수 있도록 최적의 비트 수를 결정해야 한다. 이 과정 후에 DSP, dedicated H/W등으로 설계를 수행하게 된다. 이 설계과정이 그림 7에 나타나 있다.




〔그림 7〕 고정소수점 시스템의 구현절차

아래 표에 각 변수들의 값의 범위가 나타나 있다. 실제 SNR에 따라서 변수들의 범위에 약간의 변동이 있지만 대략 비슷한 값의 범위를 갖는다.

표 3에서 나타난 바와 같이 MAP 모의실험의 결과에 나타난 변수들을 살펴보면 변수의 최대값이 대략 35, 최소값이 -21 정도가 된다. 따라서 정수부의 단어길이는 오버플로우를 방지하기 위해서 최대 7비트

〔표 3〕 각 변수들의 값의 범위

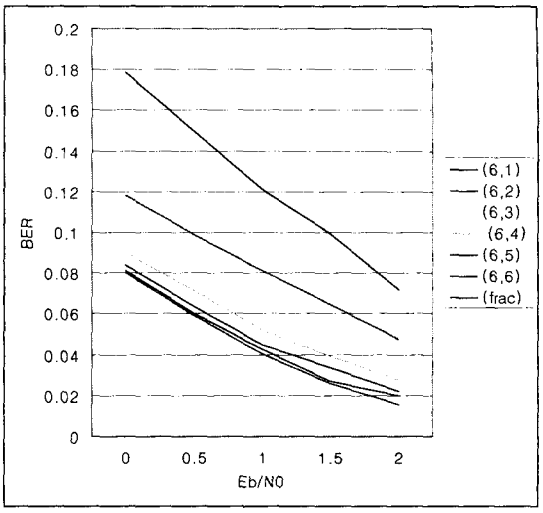
| 변수 | Max | Min | Avg | Stdev |
|---|---|---|---|---|
| 순방향 메트릭 | 23.47 | -16.21 | 1.43 | 9.36 |
| 역방향 메트릭 | 15.15 | -16.21 | 1.10 | 7.40 |
| 브랜치 메트릭 | 16.66 | -16.21 | 0.00 | 7.98 |

(a)는 SNR=0dB, (b)는 SNR=2dB

(a)

| 변수 | Max | Min | Avg | Stdev |
|---|---|---|---|---|
| 순방향 메트릭 | 35.3 | -20.9 | 2.94 | 24.94 |
| 역방향 메트릭 | 23.79 | -20.9 | 2.83 | 20.43 |
| 브랜치 메트릭 | 19.23 | -19.01 | 0.00 | 19.91 |

(b)

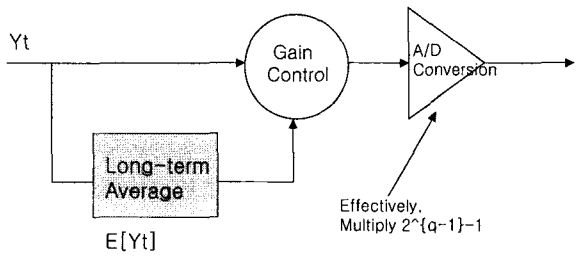



〔그림 8〕 정수부를 6비트로 고정시키고 소수부를 변화시켜 가면서 측정한 MAP복호화기의 성능
(그래프는 위로부터 표에 표시된 비트포멧 순임)

(정수부를 기준으로 63~-64)로 설정될 수 있다. 그러나 메트릭 값이 오버플로우가 발생할 정도로 아주 큰 경우는 강하게 1 또는 0을 의미하는 것이므로 포화되는 경우에도 성능에 큰 영향을 미치지 않는다. 한편, 소수부의 단어길이를 결정하기 위하여 정수부를 6비트로 고정시키고 소수부의 비트수를 변화시켜 가면서 MAP복호화 모의실험을 수행하였다. 이 모의실험의 결과가 그림 8에 나타나 있다. 결과를 살펴보면 소수부의 비트수가 증가함에 따라서 BER이 좋아지면서 소수부의 비트수가 3비트를 넘어서면 거의 성능에 차이가 발생하지 않게 된다. (6,4)비트포맷과 (6,5)비트포맷, 그리고 (6,6)비트포맷을 비교해보면 BER의 차이가 거의 없음을 볼 수 있다.

## 3.2 입력신호의 A/D변환

A/D변환기는 일반적으로 양자화 간격이 균일한 균등 양자화기(uniform quantizer)를 사용한다고 가정

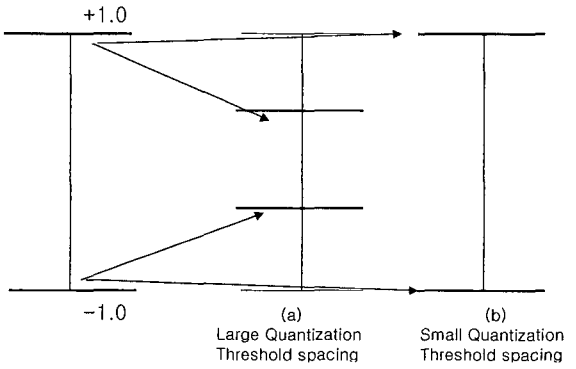



〔그림 9〕 간략화된 AGC와 A/D변환기의 모델

한다. 간략화된 AGC와 A/D변환기의 모델이 그림 9에 나타나 있다.

간략화된 AGC모델에 의하면 AGC는 수신 신호의 평균치를 1로 만들어 주는 역할을 수행한다. 이 값이 A/D변환이 되어 디지털화된 값으로 바뀌게 된다. 이 때 AGC의 이득조정에 따라서 원 전송신호가 실제 표현될 때는 그림 10의 (a), (b)에 보여지는 것과 같이 간격(spacing)이 바뀔 수 있다. 총 양자화 표현 단계수를 16단계로 가정하자. (a)의 경우에는 1.0~-1.0까지의 구간이 불과 4단계만으로 표현되어 있기 때문에 양자화 구간의 간격이 굉장히 넓은 경우로 볼 수 있다.

반면에 (b)의 경우에는 1.0~-1.0까지의 구간이 16단계로 이루어져 있으므로 구간간격이 (a)의 경우에 비해서 촘촘하다고 할 수 있다. 물론 양극단간의 절충이 필요하겠지만 (a)보다는 (b)와 같이 많은 샘플이



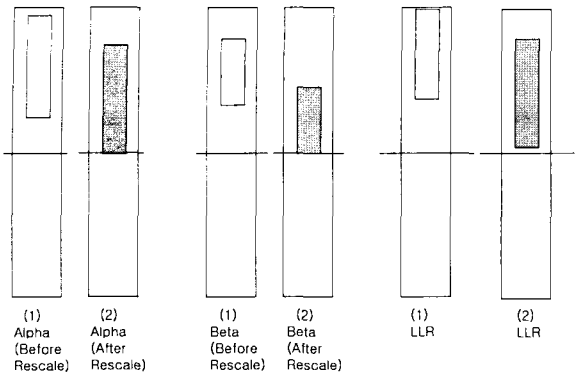

〔그림 10〕 AGC의 이득조정에 따른 변수 범위의 변환

분포하는 부분의 구간간격을 상대적으로 조밀하게 구성하는 것이 양자화 오차를 줄이는데 유리하다. 하지만 AGC, A/D모델뿐 아니라 앞절에서 프로세서 내부의 단어길이까지 고려하여 총체적인 비트결정을 수행하여야 한다. 앞절에서 송신신호를 +1, -1로 둘 경우에, 수신된 입력치의 구간은 -1X~1X까지이고 (이는 Lc값이 곱해지기 때문이다. Lc값은 SNR이 커질수록 커지며 Eb/N0 5dB에서 대략 6.3정도가 된다.). 순방향, 역방향 메트릭은 3X~-3X까지이며 LLR값은 2X~-2X까지의 값을 갖는다.

터보 복호화의 모의실험에서는 하드웨어의 비용을 고려하여 총 단어길이를 8비트로 설정한 상태에서 정수부를 위해서 할당해야 하는 최소한의 단어길이를 5비트로, 소수부를 위한 단어길이를 3비트로 설정하였다. 따라서 +1, -1의 구간을 8단계로(즉, 소수부 비트를 3비트로) 두고 순방향, 역방향, LLR메트릭을 위하여 5비트를 할당하였다. 따라서 아날로그로 볼 때 최대로 표현할 수 있는 값은 15.875(01111.111_binary)가 되고 최소의 값은 -16 (10000.000)이 된다. 그리고 MAP 복호화기를 거친 각 출력에서 7.875(00111.111_binary) 또는 -7.875보다 큰 값에 대해서는 레벨의 제한(level saturation)을 시켜서 다음 복호화시에 입력이 LLR값으로 인해서 saturation되지 않도록 LLR값을 조절하였다. 즉, 터보 복호화 프로세서 내부의 워드길이를 (5,3)-비트 포맷으로 데이터를 설정하여 수행하였다.

### 3.3 메트릭 리스케일링

순방향 메트릭 계산 회로와 역방향 메트릭 계산 회로에서 생성된 순방향 메트릭과 역방향 메트릭을 리스케일링 없이 바로 LLR을 바로 구하는 데 사용하는 경우에, LLR 계산부에서 메트릭이 더해질 때 범위 이상의 값이 생성되어 포화(saturation)가 발생하면 정밀도(precision)의 손실이 발생할 수가 있다.




〔그림 11〕 선스케일링과 후스케일링의 비교

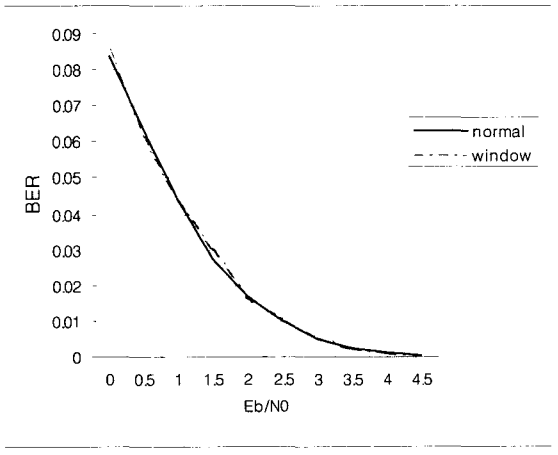
각 변수의 상대적 차이가 실제 값을 구하는데 있어서 중요한 요소이기 때문에 리스케일링이 수행되지 않은 채로 LLR을 계산할 경우에는 양 또는 음의 최대치에서 값이 포화되는 경우로 인해서 실제 메트릭간의 동적범위(dynamic range)가 줄어들고 이로 인하여 성능이 저하가 발생할 수 있다. 그림 11에 LLR 값을 구할 때 스케일링을 하는 경우와 하지 않은 경우의 예가 나타나 있다. 이와 같이 순방향 메트릭 회로 또는 역방향 메트릭 베타회로에서 미리 스케일링을 수행한 값을 LLR계산에서 사용하기 위하여 메트릭의 선스케일링(pre-rescaling)구조가 필요하다.

## 4. 모의실험

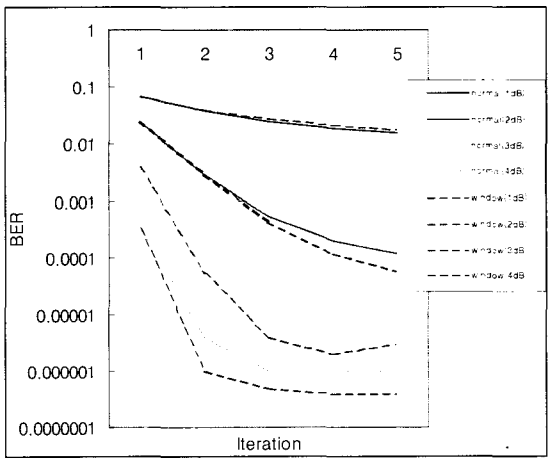
### 4.1 터보 복호화기를 위한 고려사항

그림 12에 일반적 방식과 슬라이딩 윈도우 방식의 MAP 복호화 모의실험의 결과가 나타나 있다. 그림을 살펴보면 일반적 방식과 슬라이딩 윈도우 방식간에 성능차이가 거의없음을 볼 수 있다.

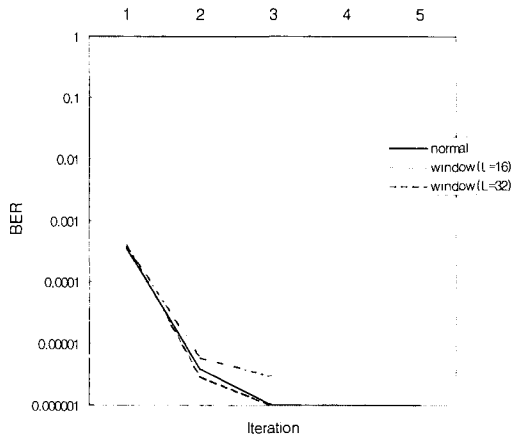
그림 13에는 일반적 MAP방식과 슬라이딩 윈도우 방식간의 반복 복호화에 따른 터보 복호화 성능을 도시하였다. 그림을 살펴보면 윈도우의 크기 L을 16으로 설정하였을 때는 반복 복호화 수행 시에 성능이 일반적인 방법에 비해서 떨어지는 것을 볼 수 있다. 그러나 윈도우의 크기 L이 32가 되는 경우에는 일반적 MAP복호화 알고리즘을 사용했을 때와의 성능의 차이가 거의 나타나지 않음을 볼 수 있다.




〔그림 12〕 일반적 방식과 슬라이딩 윈도우 방식 간의 MAP복호화 성능 (윈도우 크기 L=16)




〔그림 13〕 일반적 MAP방식과 슬라이딩 윈도우 방식 간의 터보 복호화 성능 (K=3, R=1/2 Eb/N0=4.0dB)




〔그림 14〕 고정소수점(fixed-point)방식을 사용한 터보 디코더의 모의실험 결과 (그래프는 위로부터 1dB부터 4dB 순이며 실선은 부동소수점 방식의 일반적 MAP알고리듬을 사용한 경우, 점선은 고정소수점 방식의 슬라이딩 윈도우 방식을 사용한 경우임)

즉, 윈도우의 크기를 5K보다는 조금 크게 설정해 주어야 함을 알 수 있다. 실제 구속장 K=3, 4에서 모두 사용하는 경우에는 L=32 정도가 적당하다.

그림 14에서는 윈도우 크기를 32로 설정한 경우에 각 SNR에서 반복회수에 따른 8-비트 고정소수점(fixed-point) 방식의 터보 디코더의 모의실험 결과가 나타나 있다.

그림을 살펴보면 부동소수점 방식과 고정소수점 방식간에 성능의 차이가 그리 크지 않을뿐더러 SNR이 큰 경우에는 고정소수점 방식이 오히려 부동소수점 방식에 비해서 성능이 근소하게 더 좋은 것을 볼 수 있다.

## 5. 맺음말

본 논문에서는 우수한 복호화 성능으로 많은 주목을 받고 있는 터보 복호화기를 실제 군용 통신 시스템에서 구현 시에 발생하는 여러 가지 고려할 사항을 분석하였고 또한 요소 복호화 기법인 MAP복호화기의 하드웨어 복잡도를 줄이기 위한 슬라이딩 윈도우 방식의 터보 복호화기의 구조를 제시하고 이를 모의실험을 통해서 성능을 평가하였다. 이를 실제 하드웨어로 구현하고자 할 경우에 고려되는 여러 가지 요소를 분석할 때에 하드웨어 비용과 성능간의 절충점을 찾기 위한 모의실험을 수행하였으며, 성능의 저하를 막기 위한 선-메트릭 리스케일링 구조를 제시하였다. 또한 변수의 범위뿐 아니라 복호화기 앞단의 AGC, ADC까지 고려한 단어길이를 설정하였다. MAP 디코더의 하드웨어 비용(메모리 크기)을 크게 줄일 수 있는 슬라이딩 윈도우 방식을 실제 요소 복호화 기법으로 사용하여 전체적인 터보 복호화기의 성능에 대한 모의실험을 수행하였다. 모의실험 결과 기존의 일반적 MAP복호화 방식을 사용한 결과와 비교하여 성능차이는 거의 무시할 정도였다. 또한 실제 8비트 고정소수점 방식의 모의실험에서도 부동소수점 방식으로 구현된 터보 복호화기의 성능과 거의 비슷하였으며 SNR이 큰 경우에는 근사적으로 성능이 우수한 경우도 있었다.

본 논문의 시스템 설계를 위한 여러 가지 고려사항을 적용한 슬라이딩 윈도우 기법을 사용한 터보코드는 군용 통신 시스템에서 데이터 서비스뿐 아니라 고속 실시간 응용을 위한 채널 부호로 유용하게 사용될 수 있을 것이다.

## 참 고 문 헌

1. C. Berrous, A.Glavieux, "Near Shannon limit error-correcting coding and decoding," in *Proc. ICC93*, May 1993, pp. 1064~1070.
2. C. Berrou, "Some clinical aspects of turbo codes,"

*Int. Symp. Turbo Codes*, Sept. 1997, pp. 26~31.

3. D. Divsalar and F. Pollara, "Turbo Codes for PCS Application," *Proc. of IEEE ICC'95*, June 1995.
4. P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operationg in the log domain," *Proc. ICC'95*, June 1995, pp. 1009~1013.
5. L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284 ~ 287, Mar. 1974.
6. P. Robertson, "Illuminating the structure of code and decoder for parallel concatenated recursive systematic (turbo) codes," *Proc. GLOBECOM'94*, Dec. 1994, pp. 1298~1303.
7. D. Divsalar and R. J. McEliece, "The Effective Free Distance of Turbo Codes," *IEE Electronic Letters*, vol. 32, no. 5, pp. 445~446, Feb. 29, 1996.
8. J. Hagenauer, E. Offer, and L. papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Information Theory*, vol. 42, no. 2, pp. 429~445, March 1996.
9. A. Viterbi "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, 1998, Vol, 16, no. 2, pp. 260~264.
10. 심병효, 이봉운, 구창설, "비터비 디코더의 양자화 효과분석," 공사논문집 제 43집, 1999.