다양한 Design Issue에 대한 터보 디코더의 성능분석

박 태 근*, 김 기 환**

Performance Analysis on Various Design Issues of Turbo Decoder

Taegeun Park, Kiwhan Kim

요 약

Log-MAP 복호 알고리즘을 사용하는 터보 복호기는 뛰어난 복호 성능에도 불구하고, 반복적 연산으로 인하여 인터리버의 크기에 비례하는 많은 메모리와 높은 하드웨어 복잡도가 단점으로 지적된다. 이에 본 논문에서는 Log-MAP 복호 알고리즘 기반의 터보 복호기를 설계할 때 복호 성능 및 하드웨어 복잡도에 영향을 미칠 수 있는 다양한 설계 이슈들을 제시하고, 설계 이슈들의 변화에 따른 복호 성능을 모의실험을 통하여 비교 분석한다. 하드웨어 복잡도와 복호 성능간의 균형을 고려하여 수신정보, 사전정보, 상태 메트릭을 각각 5 비트, 6 비트 그리고 7 비트로 할당하여 부동 소수점 연산의 비트오율에 근접하는 성능을 확인하였다. Log-MAP 복호 알고리즘의 주연산인 MAX*에 대한 하드웨어 복잡도와 복호 성능을 비교 분석하였다. MAX* 연산 중 계산도가 큰 오류 보정 함수를 근사화된 조합회로로 구성하여 하드웨어 부담을 줄일 수 있는 방법을 제시하였고, 윈도우 블록 길이가 32인 슬라이딩 윈도우 기법을 적용하여 적은 복호 성능 저하로 상태 메트릭 저장에 필요한 메모리 공간을 감소할 수 있음을 확인하였다.

Key Words : Turbo Codes, Turbo Decoder, 3GPP, Log-MAP

ABSTRACT

Turbo decoder inherently requires large memory and intensive hardware complexity due to iterative decoding, despite of excellent decoding efficiency. To decrease the memory space and reduce hardware complexity, various design issues have to be discussed. In this paper, various design issues on Turbo decoder are investigated and the tradeoffs between the hardware complexity and the performance are analyzed. Through the various simulations on the fixed-length analysis, we decided 5-bits for the received data, 6-bits for a priori information, and 7-bits for the quantization state metric, so the performance gets close to that of infinite precision. The MAX operation which is the main function of Log-MAP decoding algorithm is analyzed and the error correction term for MAX* operation can be efficiently implemented with very small hardware overhead. The size of the sliding window was decided as 32 to reduce the state metric memory space and to achieve an acceptable BER.

Ⅰ. 서 론

오늘날의 무선 통신 시스템에서 음성, 문자 그리고 멀티미디어 데이터 등의 정보를 전송할 때 잡음, 간섭 그리고 페이딩으로 인한 오류의 발생은 불가피하다. 이러한 오류에 의한 시스템의 신뢰도를 높이기 위해서는 오류 제어 기법을 도입하는 것이 필요하다. 채널의 특성에 따라 정보의 중복성을 높여 오류를 정

* 가톨릭대학교 정보통신전자공학부 부교수 **가톨릭대학교 컴퓨터공학과 석사과정
논문번호 : #KICS2004-09-204, 접수일자 : 2004년 9월 21일

정함으로써 임의의 채널 오류에 의한 정보 손실을 최소화하는 오류 제어 기법은 여러 가지 형태가 있지만, 기본적으로 오류 정정 부호(Error Correcting Codes)를 사용하는 것이다.

3GPP(3rd Generation Partnership Project)에서 32Kbps 이상의 고속 데이터 전송용으로 채택된 터보 코드는 BCJR(Bahl Cocke Jelinek Raviv) 알고리즘을 개량하여 1993년 C. Berrou 등에 의해서 제안되었는데, Shannon의 이론적 채널 용량 한계에 근접하는 우수한 성능을 보인다[1][2].

터보 코드는 두 개의 RSC(Recursive Systematic Convolutional Codes) 부호기가 인터리버를 사이에 두고 병렬 연접된 형태를 갖는 부호기와 반복 복호를 수행하는 복호기로 구성된다. 초기 터보 코드는 연산기의 복잡성과 복호 시간의 지연으로 인해 실시간 처리에 많은 어려움이 따랐으나 많은 사람들에 의하여 연구가 진행되고 있다[3].

터보 복호 알고리즘은 대표적으로 MAP(Maximum A Posteriori) 기반의 복호 알고리즘을 들 수 있는데, 이는 Viterbi 알고리즘의 출력을 연성 출력 형태로 수정한 SOVA(Soft Output Viterbi Algorithm) 복호 알고리즘 보다 성능이 약 2배 정도 좋지만, 지수 연산으로 인한 계산상의 복잡성과 많은 메모리의 필요로 인하여 약 4배 정도 복잡한 구조를 갖고 있어 터보 복호기 구현의 연구대상이 되고 있다[4].

MAP 복호 알고리즘의 단점인 복잡한 연산을 보완하기 위하여 MAP 복호 알고리즘을 로그 영역으로 치환하여 복호 과정 연산의 복잡도를 감소한 Log-MAP 복호 알고리즘이 제안되었지만, 여전히 인터리버 크기에 비례하는 다량의 메모리를 필요로 한다. 슬라이딩 윈도우는 연산의 복잡도를 줄인 Log-MAP 복호 알고리즘의 단점인 높은 메모리 요구량을 감소하기 위해 제안된 기법으로 입력 프레임을 여러 개의 윈도우 블록으로 나누어 각각의 윈도우 블록 단위로 복호하고, 부수적인 연산 과정을 추가하여 Log-MAP 복호 알고리즘과 비슷한 성능을 나타낸다[5][6].

터보 코드에 관한 연구는 크게 인터리버와 효율적인 터보 복호기 설계로 나눌 수 있다. 인터리버에 관한 연구는 랜덤 인터리버의 성능보다 비슷하거나 우수한 인터리버를 설계할 수 있는가의 연구로서, 랜덤과 비랜덤 순열을 이용한 무게 분산 연구[7]를 들 수 있다. 효율적인 터보 복호기의 설계는 터보 코드 고유의 특징인 반복 복호에 있어서 성능 최적화를 위한 연구와 복호 알고리즘의 효율적인 하드웨어 구현에 관한 연구로서, Segmented Sliding Window 접근방법과 면적에 효율적인 고속 복호 개념에 관한 연구[8]와 Tile Graph 접근방법을 이용한 Hybrid Tiling을 제안하여 VLSI에 효율적인 구조에 관한 연구[9]가 있다. 또한 실제적인 터보 복호기의 설계에 적용할 수 있는 기본 연구로서 VLSI 구현에 대한 설계 이슈에 관한 연구[16]도 진행되었다.

본 논문에서는 Log-MAP 복호 알고리즘 기반의 터보 복호기의 성능 및 복잡도에 영향을 미칠 수 있는 다양한 설계 이슈들을 제시하고, 설계 이슈들의 변화에 따른 성능을 비교 분석하여 3GPP 규격의 터보 복호기 설계에 적용할 수 있는 설계 방법을 제시한다.

본 논문의 제 2장에서는 터보 코드의 기본적인 부/복호기 구조와 복호과정, 대표적인 복호 알고리즘에 관하여 설명한다. 제 3장에서는 터보 복호기의 설계 시 고려해야 할 다양한 설계 이슈들을 제시하고, 모의실험을 통하여 성능을 비교 분석한다. 마지막으로, 제 4장에서는 본 논문의 결론을 제시한다.

Ⅱ. 터보 코드

2.1 터보 부/복호기 구조

그림1은 3GPP 표준 터보 부호기[10]의 구조이다. 인터리버에 의해 구분된 두개의 RSC 부호기가 병렬로 연접한 형태를 갖으며 아래의 RSC 부호기는 입력 프레임의 비트들이 인터리버에 의해 재배열된 후 부호화를 수행한다. 따라서 터보 부호기는 입력 프레임 순서열에 대한 RSC 부호기의 출력과 인터리버에 의해 재배열 된 순서열에 대한 이중의 패리티 정보를 갖게 된다.

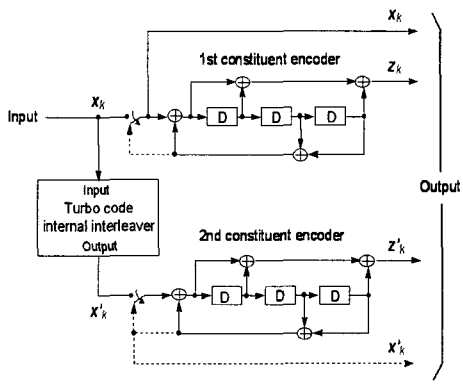

그림 1. 3GPP 표준 터보 부호기

그림2는 일반적인 터보 복호기의 구조를 나타낸다.

터보 복호기는 터보 부호기와는 달리 복호기 간의 정보를 서로 공유하여 보다 좋은 성능을 나타내기 위해 직렬 연접의 형태를 갖는다[11]. 터보 복호기는 채널을 통하여 수신된 정보와 다른편 SISO(Soft-In Soft-Out) 복호기에서 나온 사전정보(A Priori Information)를 이용하여 연성 출력을 발생하고, 다시 이전 SISO 복호기에서 나온 사전정보와 채널을 통하여 수신되고 인터리빙된 정보를 이용하여 반복적인 연산과정을 거쳐 최종적으로 경판정(hard decision)을 한다.

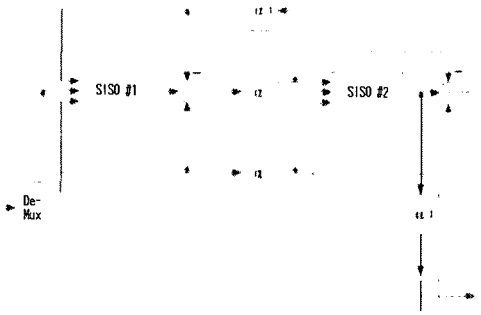

그림 2. 터보 복호기

2.2 Log-MAP

MAP 복호 알고리즘이 터보 코드에 사용되면서, 복호 성능에 영향을 미치지 않고 알고리즘의 복잡성을 감소시킬 수 있는 방법이 1995년 P. Robertson 등에 의해서 재귀적 연산을 로그 영역으로 치환함으로써 MAP 복호 알고리즘을 단순화 시키는 Log-MAP 복호 알고리즘이 제안되었다[12]. 이 Log-MAP 복호 알고리즘은 성능면에서 MAP 복호 알고리즘과 거의 동일하나 복잡성은 감소되는 특징을 갖는다.

Log-MAP 복호 알고리즘은 MAP 복호 알고리즘의 단순화를 위해, 식 (1) 과 같이 E 함수를 정의한다.

$$\begin{aligned} xEy &= \ln(e^{x}+e^{y}) \\ &= x + \ln(1+e^{x+y}) \\ &= y + \ln(1+e^{y-x}) \\ &= \max(x,y) + \ln(1+e^{|y-x|}) \end{aligned} \quad (1)$$

식(1)로부터 순방향 상태 메트릭 로그값 $A_k(s)$와 역방향 상태 메트릭 로그값 $B_k(s)$, 가지 메트릭 로그값 $\Gamma_k(s)$ 그리고 가능도비 $\Lambda_k(s)$는 다음과 같이 정의된다.

$$A_k(s) = \max_{s'}{}^{*}\,[A_{k-1}(s') + \Gamma_k(s', s)] \quad (2)$$

$$B_{k-1}(s') = \max_{s}{}^{*}\,[B_k(s) + \Gamma_k(s', s)] \quad (3)$$

$$\Gamma_k(s', s) = C + \frac{1}{2}u_k L(u_k) + \frac{L_c}{2}\sum_{l=1}^{n} y_{kl}x_{kl} \quad (4)$$

$$\Lambda_k(s) = \max_{s_1}{}^{*}\,[A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)] - \max_{s_0}{}^{*}\,[A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)] \quad (5)$$

여기에서 max*[ ]는 식(6)과 같은 정의에 의해 계산할 수 있다.

$$\max{}^{*}[x, y] = \max(x, y) + f_c(|y-x|) \quad (6)$$

2.3 슬라이딩 윈도우

MAP 복호 알고리즘을 이용하여 일련의 정보 열로 이루어진 입력 프레임을 복호하기 위해서는 일반적으로, 모든 정보에 대한 가지 메트릭을 계산하여 메모리에 저장한 후 역방향 상태 메트릭을 계산하면서, 메모리에 미리 계산하여 저장된 순방향 상태 메트릭을 이용하여 가능도비 값을 계산한다. 이 복호과정에서 가지 메트릭과 순방향 상태 메트릭을 저장해야 하기 때문에 인터리버의 크기에 비례하여 다량의 메모리가 필요하게 된다.

슬라이딩 윈도우 기법을 이용하여 MAP 복호 알고리즘에 적용하기 위해서는 순방향 상태 메트릭과 역방향 상태 메트릭을 윈도우 블록 단위로 계산해야 한다. 윈도우 블록을 순방향으로 수행한다면, 순방향 상태 메트릭은 윈도우 블록의 진행 방향과 같기 때문에, 각 윈도우 블록의 순방향 상태 메트릭 초기값은 이전 윈도우 블록의 최종 순방향 상태 메트릭 값을 통해서 구할 수 있다. 역방향 상태 메트릭은 윈도우 블록 진행 방향과 반대로 역방향으로 진행되기 때문에, 순방향 상태 메트릭 계산과는 달리 각 윈도우 블록의 초기값을 일정하게 설정해야 하는데 이는 복호기의 비트오율(Bit Error Rate: BER) 성능을 저하 시킨다. 이런 성능 저하를 해결하기 위해 각 윈도우 블록의 역방향 상태 메트릭의 초기값을 정해주는 부가적인 과정이 필요하다[8].

Ⅲ. 다양한 설계이슈에 대한 성능분석

3.1 실험환경

본 논문에서의 성능분석을 위한 모의실험 환경을 그림3과 같이 나타내었다. C 언어로 구현된 랜덤 발

생기로부터 발생된 정보를 이용하여 3GPP 표준 터보 부호화기로부터 부호화된 신호 ' 1' 과 ' 0' 을 각각 ' 1' 과 ' -1' 로 변조시키는 BPSK(Binary Phase Shift Keying) 변조 후 평균이 0이고 분산이 No/2 인 AWGN(Additive White Gaussian Noise) 채널을 통하여 정보가 전송된다고 가정한다. 또한 프레임의 크기는 10^7, 인터리버의 크기 N 은 1024, 부호율 R은 1/3, 반복 복호 횟수는 8, 그리고 Eb/No는 0.2dB 에서 1.2dB 까지 0.2dB의 변화를 가진 실험조건을 갖는다.

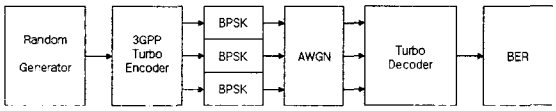

그림 3. 모의실험 시스템

3.2 고정 소수점 비트 할당

MAP 기반의 복호 알고리즘을 이용한 터보 복호기는 터보 부호화기로부터 부호화 되고, 채널을 거쳐 수신되는 수신정보(Received Data: RCV), 터보 복호기 내의 SISO 복호기로 부터의 연성 출력이 인터리버/디인터리버에 의해 재배열 되어 복호 과정에서 오류의 확률을 낮추는 사전정보(A Priori Information: API) 그리고 SISO 복호기 내의 이전 트렐리스 상태에서 다음 트렐리스 상태로 천이될 때의 가지 메트릭(Branch Metrics: BM)과 상태 메트릭(State Metrics: SM)을 저장할 수 있는 많은 양의 메모리가 필요하다. 이는 슬라이딩 윈도우 기법을 적용하여도 상태 메트릭 저장에 필요한 메모리 공간만을 줄일 수 있기 때문에, 기본적으로 터보 복호기 설계시 최소의 성능저하로 적절한 비트 할당을 고려해야 한다[13].

전체 소수점 이하의 비트 할당은 전체 비트 길이와 복호 성능에 영향을 주는 측면에서 중요성을 갖는다. 그림4는 전체 소수점 이하의 비트 길이 변화에 따른 비트 오율로서, 정수부 비트 길이에 제한을 두지 않은 상태에서 소수부에 대한 비트 길이를 변화하였을 때의 비트 오율 결과이다. 소수점 이하 비트 길이가 2와 3일 때는 부동 소수점 일 때의 비트 오율에 근접하기 때문에 소수점 이하의 비트 길이를 2로 제한하는 것이 적절하다.

터보 복호기의 구현에서 전체 비트 할당을 위해서는 소수점 이하 비트 길이를 2로 제한한 상태에서 수신정보, 사전정보 그리고 상태 메트릭의 순서로 비트 할당을 해야 하는데, 터보 복호기의 입력에서부터 덧셈/뺄셈이 주로 연산되는 과정에서 연산값의 증가가 이루어지기 때문이다.

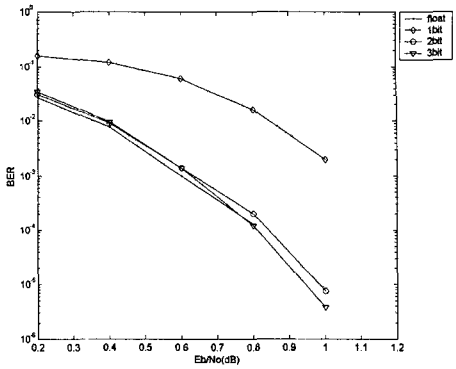

그림 4. 고정 소수점 비트할당에 대한 BER

채널을 통하여 수신된 신호는 정보 비트와 패리티 비트로 나눌 수 있다. 이 두 정보는 사전정보와 관련하여 가지 메트릭 값을 생성하고, 가지 메트릭 값은 SISO 복호기 내에서 상태 메트릭 값과 함께 연산하여 연성 출력 값에 이르기까지 영향을 미친다는 점에서 중요성을 갖고, 상태 메트릭은 SISO 복호기 내에서 상태 메트릭 연산기가 트렐리스 상태 당 2^{ν} 만큼 필요하기 때문에 상태 메트릭의 비트 길이는 상태 메트릭 연산기의 복잡도 및 필요 메모리, 가능도비 연산기에까지 영향을 미친다는 점에서 중요성을 갖는다. 여기에서 ν는 터보 복호기의 메모리 길이이다.

그림 5 는 주요 인자의 비트 할당에 대한 비트 오율을 나타내었다. 수신정보(RCV), 사전정보(API) 그리고 상태 메트릭(SM)의 순서로 비트 할당을 할수록 부동 소수점 일 때의 성능에 대해 차이를 보이지만 수신정보 5 비트, 사전정보 6 비트, 그리고 상태 메트릭 7 비트를 할당할 때 적은 성능저하로 적절한 비트 할당을 할 수 있다.

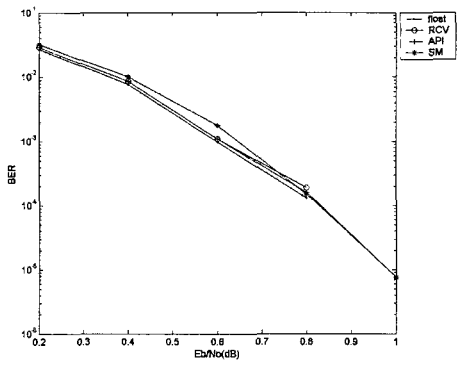

그림 5. 주요 인자의 비트 길이에 따른 BER

3.3 오류 보정 함수의 근사화

Log-MAP 복호 알고리즘은 MAP 복호 알고리즘의 계산상의 복잡도를 로그 영역으로 치환하여 감소하였지만 식(6)의 MAX* 연산에서 $f_c(x)$ 함수는 식(7)처럼 가장 큰 복잡도를 갖는다.

$$\ln(e^{x}+e^{y}) = MAX(X,Y) + \ln(1+e^{-|X-Y|}) = MAX^{*}(X,Y) \quad (7)$$

그림6의 실선은 식(7)에 따른 오류 보정 함수(Error Correcting Function) $f_c(x) = \ln(1+e^{-|x|})$를 나타낸 것이다. 이 함수 $f_c(x)$는 이상적인 값으로 실제 터보 복호기 구현에 사용될 수 없으므로 양자화된 대입 값이 필요하다.

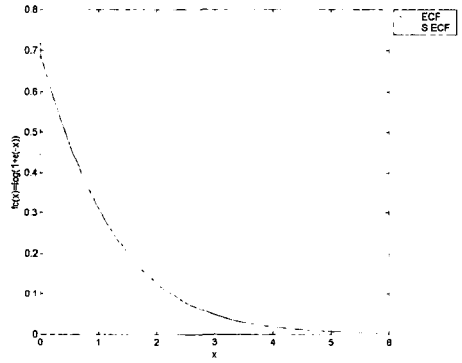

그림 6. 오류 보정 함수와 간략화된 오류 보정 함수의 특성

일반적으로 오류 보정 함수 $f_c(x)$ 는 0에서 2.0까지의 범위를 갖는 x에 대한 실제 값을 고정 소수점 이하 자리 수에 대응하는 근사화된 값으로 대체하여 ROM에 이식한 참조 테이블(LookUp Table: LUT) 방식을 이용한다. 하지만, ROM을 이용한 참조 테이블 방식을 이용할 경우 양자화된 값을 갖는 경우의 수와 트렐리스 상태의 수 그리고 양자화 비트 길이의 곱의 용량을 갖는 메모리가 필요하며, 양자화된 값을 갖는 경우의 수와 트렐리스 상태 수의 곱의 메모리 접근에 의한 부담이 발생한다.

이에 본 논문에서는 간략화된 MAX* 연산을 제안하였다. 앞서 고정 소수점 비트 할당에서의 정해진 전체 소수점 이하의 비트 길이에 따라 양자화된 값을 나타내는 표1을 참조하여 오류 보정 함수 $f_c(x)$를 설계하면 간략화된 오류 보정 함수는 그림6의 점선과 같은 오류 보정 특성을 갖고, 이는 그림7의 간단한 로직 게이트로 이루어진 조합회로로 구성할 수 있다.

표 1. 오류 보정 함수의 간략화

|x|	0.00	0.25~0.75	1.0~1.75	≥2.00
$\ln(1+e^{-x})$	0.75	0.50	0.25	0.00
allocated bit	11	10	01	00

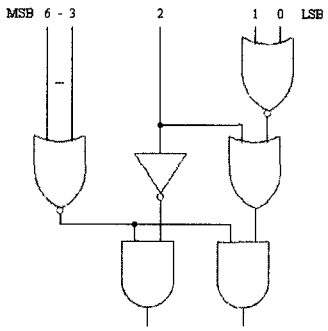

그림 7. 간략화된 오류 보정 함수 설계

고정 소수점 비트 할당을 고려한 오류 보정 함수 $f_c(x)$ 를 간략화 시켜 조합회로로 대체하여 비트 오율을 살펴 본 결과 그림8과 같이 기존의 부동 소수점일 때의 비트 오율에 근접하는 성능을 보인다. 또한 전체 부동 소수점 연산과 비교해 볼 때, 간략화된 오류 보정 함수 $f_c(x)$를 이용하면 빈번한 메모리 접근에 대한 시스템 자원의 부담을 감소시킬 수 있다.

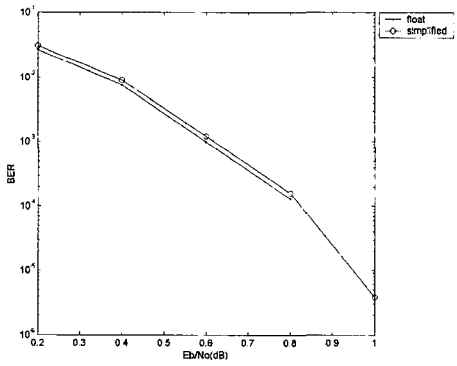

그림 8. 간략화된 오류 보정 함수에 대한 BER

3.4 MAX* 연산의 구조

Log-MAP 복호 알고리즘은 일반적으로 시간 k 에 대해 2^{v} 개의 순방향 상태 메트릭, 2^{v} 개의 역방향 상

태 메트릭 그리고 2×2^{v} 개의 가지 메트릭을 이용하여 하나의 가능도비를 입력 프레임 열에 대해 식(5)를 이용하여 계산하기 때문에 많은 계산량과 복잡도를 요구한다.

일반적으로 Log-MAP 복호 알고리즘은 MAX^{*} 연산의 반복으로 이루어지므로, MAX^{*} 연산 시간을 줄이거나 계산상의 복잡도를 줄이는 것이 Log-MAP 기반의 터보 복호기의 효율을 높일 수 있는 방법이 될 수 있다.

이에 본 논문에서는 시간 k 에서의 연산 시간이 가장 긴 가능도비 연산에 간소화된 MAX^{*} 연산을 적용하여 Log-MAP 복호 구조를 설계하고, 성능평가를 실시하였다.

두 개의 인자를 갖는 $MAX^{*}(A,B)$ 연산을 이용한 $MAX^{*}(A,B,C,D)$ 연산을 식(8)로 나타낼 수 있고, 이로부터 식(9)를 유도할 수 있으며, 또한 $MAX^{*}(A,B,C,D,E,F,G,H)$로 유도할 수 있다.

$$\begin{aligned} &MAX^{*}(A,B,C,D) \\ &\quad = MAX^{*}(MAX^{*}(A,B),\ MAX^{*}(C,D)) \end{aligned} \tag{8}$$

$$\begin{aligned} &MAX^{*}(A,B,C,D) \\ &\quad = MAX^{*}(MAX(A,B)+\Delta_1,\ MAX(C,D)+\Delta_2) \\ &\quad \approx MAX(MAX(A,B),\ MAX(C,D))+\Delta_3 \end{aligned}$$

$$\begin{array}{lll} IF & S_1 \geq S_2 & ELSE \\ & A+\Delta_1+\Delta_3 & C+\Delta_2+\Delta_3 \\ & or & or \\ & B+\Delta_1+\Delta_3 & D+\Delta_2+\Delta_3 \end{array} \tag{9}$$

그림9는 식(8)을 바탕으로 일반적인 Log-MAP 복호 알고리즘을 이용하는 $MAX^{*}(A,B,C,D)$의 비교 선택기(Compare-Select: CS)의 구조이며 그림10은 간소화된 식(9)를 이용하여 설계한 $MAX^{*}(A,B,C,D)$의 비교 선택기의 구조이다.

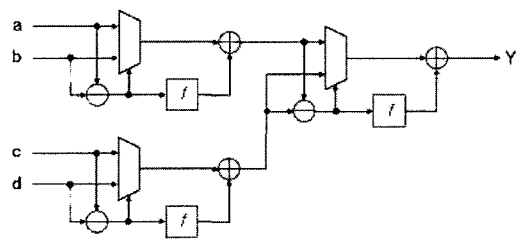

그림 9. 일반적인 비교 선택기의 구조

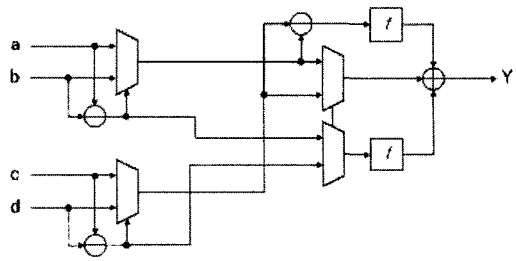

그림 10. 간소화된 비교 선택기의 구조

파이프 라인을 적용하지 않는다고 가정하면, 그림9와 같이 일반적인 비교 선택기로 $MAX^{*}(A,B,C,D)$ 연산을 하는 경우 네 번의 덧셈과 두 번의 테이블 참조 시간이 필요하지만, 그림10과 같이 간소화된 비교 선택기로 $MAX^{*}(A,B,C,D)$ 연산을 하면 세 번의 덧셈과 한번의 테이블 참조 시간이 필요하여 각각 한번의 덧셈과 테이블 참조시간을 감소할 수 있다.

표2는 8개의 트렐리스 상태에서 간소화된 비교 선택기에 따른 가능도비 연산기(LLR Computation Unit: LCU)의 복잡도를 나타내고, 그림11은 기존의 부동 소수점과 간소화된 비교 선택기의 변화에 따른 비트 오율을 나타낸다.

표 2. Log-MAP 복호 구조 변화에 따른 가능도비 연산기의 복잡도 비교

	MUX		ADDER		$f_c(x)$	MUX+ADDER
	개수	게이트 수	개수	게이트 수	개수	게이트 수
MAX2	14	945	28	4,326	14	5,271
MAX4	18	1,215	24	3,708	10	4,923
MAX8	22	1,485	20	3,090	6	4,575

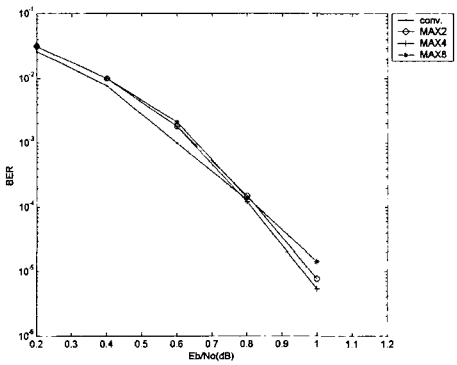

그림 11. Log-MAP 복호 구조의 변화에 따른 BER

$MAX^{*}(A,B)$를 이용하여 가능도비를 연산할 경우

시간 k 에 대해 두개의 인자(A, B)를 연산하는 MAX2 비교 선택기를 기준으로 $2^v+\frac{1}{2}2^v+\frac{1}{4}2^v$ 번의 비교 선택이 필요하다. 네 개의 인자(A, B, C, D)를 연산하는 간소화된 MAX4와 여덟 개의 인자(A, B, C, D, E, F, G, H)를 연산하는 간소화된 MAX8을 이용할 경우 시간 k 에 대해 비교 선택의 복잡도는 표3과 같이 MUX의 개수는 늘어나지만, 상대적으로 부담이 적은 덧셈기의 개수와 오류 보정 함수의 참조횟수가 줄어든다. 게이트 수는 2-input NAND gate를 기준으로 9bit MUX와 9bit ADDER를 합성하였을 때의 결과로서 각각 67.5unit과 154.5 unit이다. 이때, ADDER는 Ripple Carry Adder이다. 비트 오율 측면에서는 그림11과 같이 1dB 이후에서 동일하게 10^{-6} 이하의 성능을 갖지만, 파이프 라인의 적용을 고려해 볼 때, MAX2가 터보 복호기 구현의 측면에서 규칙적인 특성을 보이므로 파이프 라인 구성에 적합함을 보인다.

3.5 슬라이딩 윈도우 블록의 길이

슬라이딩 윈도우 기법을 역방향 상태 메트릭에 적용할 경우 각 윈도우 블록에서 역방향 상태 메트릭의 초기 값을 구하는 과정에서 손실의 발생으로 성능이 저하될 수 있기 때문에 역추적 과정을 설정하여 역방향 상태 메트릭의 손실을 줄일 수 있다. SW-Log-MAP 복호 알고리즘은 상태 메트릭의 계산 과정에서 Log-MAP 복호 알고리즘과 구별된다. Log-MAP 복호 알고리즘이 입력 프레임의 끝에서부터 상태 메트릭을 계산하는 것에 반해 SW-Log-MAP 복호 알고리즘은 입력 프레임을 윈도우 블록으로 나누어 각 윈도우 블록에서부터 더미 연산 과정을 거쳐 상태 메트릭을 계산한다. 따라서, SW-Log-MAP 복호 알고리즘의 구현에는 윈도우 블록의 크기와 더미 연산 블록 크기의 설정이 필요하다[14][15].

일반적으로 SW-Log-MAP 복호기는 더미 연산 블록 크기와 같은 윈도우 블록 크기를 설정하여 상태 메트릭 연산기의 활용도를 최대로 한다. 더미 연산 블록 크기가 윈도우 블록 크기보다 크면 실제 값을 계산할 상태 메트릭 연산기는 더미 연산을 위한 상태 메트릭 연산기가 동작하는 차이만큼 기다려야 한다. 또한, 윈도우 블록 크기가 더미 연산 블록 크기보다 크다면 더미 연산을 위한 상태 메트릭 연산기는 그 차이만큼 동작하지 않으므로 하드웨어 자원의 낭비를 초래하게 되어, 윈도우 블록 크기와 더미 연산 블록 크기는 같게 한다.

그림12는 부동 소수점과 고정 소수점 비트 할당한 Log-MAP, 슬라이딩 윈도우 블록 크기 변화에 대한 비트 오율 성능의 변화를 나타낸다.

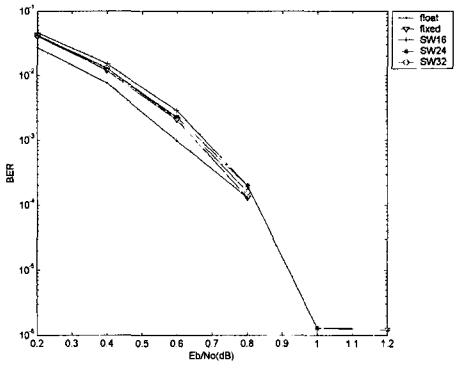

그림 12. 슬라이딩 윈도우 크기 변화에 대한 BER

0.8dB 까지 윈도우 블록의 크기가 입력 프레임의 길이와 같은 고정 소수점 비트 할당된 Log-MAP 복호기와 윈도우 블록의 크기가 변화된 SW-Log-MAP 복호기의 성능은 큰 차이를 보이지 않지만, 1dB 이상에서는 윈도우 블록의 크기가 32(SW32)인 경우 Log-MAP 복호기의 성능에 근접함을 보인다.

Ⅳ. 결 론

Log-MAP 복호 알고리즘은 오류 정정 성능을 높이기 위하여 반복 연산의 과정을 거치기 때문에 인터리버의 크기에 비례하는 많은 메모리의 필요가 치명적인 단점이 된다. 이는 적은 하드웨어 자원과 저전력으로 동작해야 하는 이동통신 기기에의 적용에 큰 장애물로 귀착될 수 있다. Log-MAP 터보 복호기의 설계시 적은 성능 저하로 하드웨어 복잡도와 필요 메모리 부담을 줄이기 위하여 본 논문에서는 다양한 설계 이슈들을 제시하였고, 모의실험을 통하여 비교 분석하였다.

하드웨어 복잡도와 복호 성능간의 균형을 고려하여 소수점 이하 2 비트, 수신정보 5 비트, 사전정보 6 비트 그리고 상태 메트릭 7 비트로 비트를 할당하면 부동 소수점 연산의 비트오율에 근접하는 성능을 얻을 수 있고, 적은 필요 메모리 공간과 적은 하드웨어의 부담으로 Log-MAP 복호 알고리즘 기반의 터보 복호기를 설계할 수 있다.

MAX^* 연산에서 복잡도가 제일 큰 오류 보정 함수 $f_c(x)$는 근사화한 $f_c(x)$ 함수를 이용하여 조합회로

로 설계하면 참조 테이블을 이용한 방법에 비하여 연산 속도와 ROM의 이용에 수반되는 하드웨어 부담을 줄일 수 있다. 또한, Log-MAP 복호 알고리즘의 주연산인 MAX^* 연산을 근사화하여 가능도비 연산에 이용할 수 있는 여러 구조를 제시하고 분석하였다. 한 번에 계산할 수 있는 인자의 수가 많아질수록 복호 연산 시간과 하드웨어 복잡도를 낮출 수 있지만, 복호 성능의 저하를 야기하며, 파이프라인 구성의 어려움이 따르므로 규칙성을 갖는 $MAX^*(A,B)$ 연산이 파이프라인 구성에 적합하다.

최종적으로 슬라이딩 윈도우 기법의 도입은 상태 메트릭 저장 공간을 윈도우 블록 크기로의 획기적인 절감을 얻을 수 있지만, 윈도우 블록 크기가 복호 성능을 좌우 하므로 이를 비교 분석한 결과 윈도우 블록 크기가 32 일 때 부동 소수점 연산의 복호 성능에 근접하는 터보 복호기를 설계할 수 있다.

감사의 글

저자들은 본 연구를 위하여 설계 환경을 제공하여 준 IDEC(IC Design Education Center)에 감사드린다.

참 고 문 헌

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo-codes(1)," *in Proc. IEEE ICC'93*, 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. Information Theory*, vol. IT-20, pp. 248-287, Mar. 1974.

[3] 김수영, 이수인, "터보코드(Turbo Codes) 개발 동향," *주간 기술 동향*, 한국전자통신연구원, 888호, pp. 1-12, 1999년.

[4] S. Barbulescu and S. Piebrobon, "Turbo codes: A tutorial on a new class of powerful error correcting coding schemes, part 2: Decoder design and performance," *IEEE Journal of Electrical and Electronics Engineering*, Australia, vol. 19, no. 3, pp. 143-152, Sept. 1999.

[5] H. Dawid and H. Meyr, "Real-time algorithms and VLSI architectures for soft output MAP convolutional decoding," *in Proc. Personal, Indoor, and Mobile Radio Communications*, PIMRC'95. Wireless:Merging onto the Information Superhighway, vol. 1, pp. 193-197, 1995.

[6] S. Benedetto et al., "Soft-output decoding algorithms in iterative decoding of Turbo codes," *JPL*, TDA Progress Report 42-124, Feb. 1996.

[7] S. Dolinar and D. Divsalar, "Weight distributions for Turbo codes using random and nonrandom permutations," TDA Progress report 42-122, pp. 56-65, Aug. 1995.

[8] Z. Wang, Z. Chi and K. K. Parhi, "Area-efficient high-speed decoding schemes for Turbo decoders," *IEEE Trans. VLSI Systems*, vol. 10, no. 6, Dec. 2002.

[9] M. M. Mansour and N. R. Shanbhag, "VLSI architectures for SISO-APP decoders," *IEEE Trans. VLSI Systems*, vol. 11, no. 4, Aug. 2003.

[10] 3GPP TS 25.212, 3GPP technical specification group radio access network, multiplexing and channel coding(FDD), (Release 4).

[11] F. Huang, "Evaluation of soft output decoding for Turbo codes," Virginia Tech, 1997.

[12] P. Robertson, E.Villebrun and P.Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *in Proc. Int. Conf. Communications*, pp. 1009-1013, June 1995.

[13] 김기환, 박태근, "터보 디코더 비트할당에 대한 성능분석," *SoC 설계연구회 학술발표회 논문집*, p. 35-39, May 2004.

[14] G. Masera, G. Piccinini, M. Rock, and M. Zamboni, "VLSI architectures for Turbo codes," *IEEE Trans. VLSI Systems*, vol. 7, pp. 369-79, Sept. 1999.

[15] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 260-264, Feb. 1998.

[16] Z. Wang, H. Suzuki and K. K. Parhi, "VLSI implementation issues of Turbo decoder design for wireless applications," *IEEE Workshop on*

Signal Processing Systems, pp. 503-512, Oct. 1999.

박 태 근 (Taegeun Park)

1985년 연세대학교 전자공학 학사.
1988년 Syracuse Univ. Computer 공학석사.
1993년 Syracuse Univ. Computer 공학박사.
1991년 - 1993년 Coherent Research Inc. VLSI 설계 엔지니어.
1994년 - 1998년 현대전자 System IC 연구소 책임연구원.
1998년 - 현재 가톨릭대학교 정보통신전자공학부 부교수.
<관심 분야> VLSI 설계, CAD, 병렬처리 등임.

김 기 환 (Kiwhan Kim)

2003년 가톨릭대학교 정보통신공학 학사.
2004년 현재 가톨릭대학교 컴퓨터공학과 석사과정.
<관심 분야> VLSI 설계, 이동통신 시스템 등임.