3GPP 규격 오류 정정 부호 기법의 성능 평가

Performance Analysis of Error Correction Codes for 3GPP Standard

신 나 나 · 이 창 우*

Na-Na Shin · Chang-Woo Lee*

요 약

3GPP 표준의 오류 정정 부호 기법 중의 하나로 채택된 turbo 부호는 그 성능이 Shannon이 제시하는 이론적 한계 값에 근사하기 때문에 많은 관심을 받고 있다. 그러나 계산상의 복잡함과 많은 메모리를 요구한다는 단점이 있고 이를 보완할 수 있는 Log-MAP, Max-Log-MAP, SOVA, sliding window 알고리즘 등이 제안되었다. 본 논문에서는 turbo 복호 알고리즘을 부동 소수점 연산과 고정 소수점 연산을 이용하여 구현하였을 때 성능을 해석하였다. 그리고 Log-MAP 알고리즘의 성능에 근사하는 효율적인 고정 소수점 구현 방법을 제안하였다. 이 방법을 Log-MAP과 sliding window 알고리즘에 적용하여 성능을 분석하였다.

Abstract

Turbo code has been adopted in the 3GPP standard, since its performance is very close to the Shannon limit. However, the turbo decoder requires a lot of computations and the amount of the memory increases as the block size of turbo codes becomes larger. In order to reduce the complexity of the turbo decoder, the Log-MAP, the Max-Log-MAP and the sliding window algorithm have been proposed. In this paper, the performance of turbo codes adopted in the 3GPP standard is analyzed by using the floating point and the fixed point implementation. The efficient decoding method is also proposed. It is shown that the BER performance of the proposed method is close to that of the Log-MAP algorithm.

Key words : Turbo Codes, Log-MAP, SOVA, Sliding Window Algorithm

Ⅰ. 서 론

이동 통신 채널에서 음성, 문자, 영상 등의 정보를 전송할 때 잡음, 간섭 혹은 다경로 전송에 따른 페이딩으로 인한 신호의 왜곡이 발생한다. 이러한 신호의 왜곡으로 인한 전송 오류를 줄이기 위해서는 오류 제어 기법을 도입하는 것이 필요하다. 채널의 특성에 따라 여러 가지 오류 제어 기법을 사용할 수 있는데 오류 정정 부호(error correcting code)를 사용하는 것도 하나의 유력한 방법이다[1].

3GPP 표준의 오류 정정 기법 중의 하나인 turbo 부호는 1993년 Berrou 등에 의해서 제안되었는데 Shannon의 이론적 한계 값에 근접하는 우수한 성능을 보이는 오류 정정 부호이다[2]. Turbo 부호는 두개의 RSC(recursive systematic convolutional code) 부호기가 인터리버를 사이에 두고 병렬로 연결된 형태를 갖는 부호기 부분과 반복적 복호 과정을 갖는 복호기 부분으로 구성되어 있다. Turbo 복호 알고리즘으로는 MAP(maximum a posteriori)에 근거한 알고리즘과 Viterbi 알고리즘의 출력을 연성 출력(soft-output) 형

「본 연구는 2002학년도 가톨릭대학교 교비 연구비 지원으로 수행되었음.」

가톨릭대학교 컴퓨터공학과 정보통신전공(Department of Computer Engineering, The Catholic University of Korea)

*가톨릭대학교 정보통신전자공학부(School of Information, Communications and Electronics Engineering, The Catholic University of Korea)

· 논 문 번 호 : 20031117-158

· 수정완료일자 : 2003년 12월 16일

태로 변경한 SOVA(soft output Viterbi algorithm)가 있다. 이중에서 MAP 알고리즘이 이론적 최적 알고리즘인데 다수의 곱셈과 exponential 연산으로 계산이 복잡하고 많은 메모리가 필요한 단점이 있다. 이러한 단점을 보완하기 위해서 Log-MAP, Max-Log-MAP, SOVA, sliding window 알고리즘 등이 제안되었다[3]. Log-MAP 알고리즘은 log 연산의 특성을 이용하여 곱셈을 덧셈으로 바꾸고 지수 연산을 크게 줄인 것이고 Max-Log-MAP 은 Log-MAP 알고리즘을 근사화한 준최적 알고리즘이다. 또한 sliding window 방법은 수신된 시퀀스를 제한된 범위로 나눠서 복호과정을 수행함으로써 MAP 알고리즘의 메모리의 크기를 줄이는 방법이다[4],[5].

본 논문에서는 3GPP 표준 오류 정정 부호 기법 중의 하나인 turbo 부호의 복호기를 Log-MAP, Max-Log-MAP, sliding window Log-MAP, sliding window Max-Log-MAP, SOVA로 구현하고, 각각의 알고리즘에 대해 부동 소수점 연산(floating-point arithmetic)과 고정 소수점 연산(fixed-point arithmetic)[6]을 이용하여 AWGN(additive white Gaussian noise)과 페이딩 채널 환경에서의 성능을 분석하였다. 특히 Max-Log-MAP 알고리즘을 고정 소수점으로 구현하여 전체 비트수를 고정하고 최적의 성능을 가지는 소수점 이하 비트수를 구하였다. 또한 Log-MAP 알고리즘의 Jacobian logarithm 함수를 구현할 때 correction term 부분에 대해 LUT(look up table)를 이용하는 대신 고정 소수점 구현이 가능하고 계산량을 더욱 줄일 수 있는 새로운 함수를 제안하고 성능을 분석하였다.

본 논문의 구성은 다음과 같다. 제 2장에서는 터보 부호기와 터보 복호 알고리즘에 대해 소개하고, 제 3장에서는 부동 소수점을 이용한 turbo 복호 알고리즘에 대해 성능을 해석한다. 제 4장에서는 고정 소수점을 이용한 turbo 복호 알고리즘의 성능을 해석하고 Log-MAP 알고리즘의 계산량을 줄이기 위한 새로운 함수를 제안하며 마지막으로 제 5장에서 결론을 맺는다.

Ⅱ. Turbo 부호의 복호 알고리즘

2-1 Log-MAP 알고리즘[7]

3GPP 표준의 오류 정정 부호 기법 중의 하나로

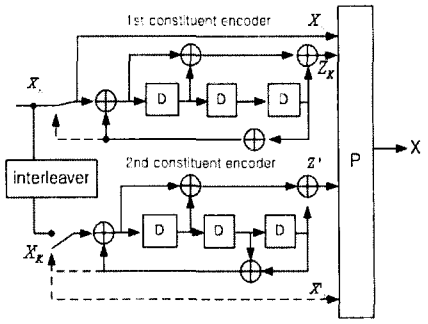

그림 1. 3GPP 표준 turbo 부호기
Fig. 1. Turbo encoder of 3GPP standard.

채택된 turbo 부호는 데이터 정보의 오류 정정을 위하여 주로 사용되는데 그림 1에 3GPP 표준 turbo 부호를 위한 부호기의 구조를 도시하였다[8].

Turbo 부호를 복호하기 위해서 1995년 Robertson 등에 의해 제안된 Log-MAP 알고리즘은 성능면에서는 MAP 알고리즘과 동일하나 log 연산의 특성을 이용하여 계산상의 복잡함을 감소시킨다. Log-MAP 알고리즘의 순방향 메트릭의 로그값 $A_k(s)$, 역방향 메트릭의 로그값 $B_{k-1}(s')$과 Log Likelihood Ratio (LLR) 값 Λ_k는 다음 식으로 계산할 수 있다. $\Gamma_k(s', s)$은 가지 메트릭의 로그값이고 h_α, h_β은 $A_k(s)$, $B_{k-1}(s')$값을 정규화하는 값이다.

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

$$B_{k-1}(s') = \max_s{}^*[B_k(s)+\Gamma_k(s',s)] + h_\beta \quad (2)$$

$$\Lambda_k = \max_{s1}{}^*[A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)] - \max_{s0}{}^*[A_{k-1}(s')+\Gamma_k(s',s)+B_k(s)] \quad (3)$$

이 때 max*( ) 부분은 다음과 같은 식에 의해서 계산되는 Jacobian logarithm이다.

$$\max{}^*(x,y) = \max(x,y) + \ln(1+\exp\{-|x-y|\}) = \max(x,y) + f_c(|x-y|) \quad (4)$$

식 (4)의 $f_c(\bullet)$ 함수를 계산에서 제외하여 복호과정을 단순화시킨 것이 Max-Log-MAP 알고리즘이다.

2-2 Sliding window 알고리즘[4]

Sliding window 알고리즘은 전체 블록을 제한된 크기의 부블록으로 나누고 각 부블록 내에서 순방향, 역방향, LLR을 계산함으로써 메모리의 크기를

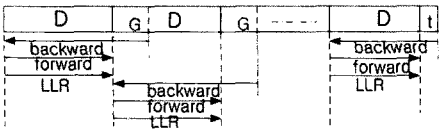

그림 2. Sliding window 알고리즘
Fig. 2. Sliding window algorithm.

줄일 수 있는 복호 알고리즘이다. 그러나 역방향 메트릭의 초기 값을 구하는 과정에서 손실이 발생하여 성능이 저하될 수 있다. 이러한 손실을 줄이기 위해서 초기 역방향 메트릭 값을 구하는 역추적(trace back) 영역을 설정하여 역방향 메트릭 값의 손실을 줄인다. 그림 2에 sliding window 알고리즘을 나타내었는데 역추적 영역은 G, 복호 과정을 수행하는 영역은 D로 나타내었다. G 영역에서 역 추적해서 D 영역의 역방향 메트릭 값을 구하고, 그 이후 D 영역에서 순방향 메트릭을 계산하고 그 결과를 바탕으로 LLR 을 계산한다. 첫번째 D 영역에서 복호 과정이 끝나면 윈도우를 D만큼 이동하여 다시 복호 과정을 수행한다. Sliding window 알고리즘은 다음과 같이 수행된다.

i) 역방향 메트릭을 식 (5)와 같이 초기화하고 식 (2)를 사용해서 역방향 메트릭을 계산한다.

$$B_{(D+G)}(s) = \log[\tfrac{1}{8}], \qquad s = 0,1,.....7 \tag{5}$$

ii) 식 (1)을 사용해서 순방향 메트릭을 계산하면서 식 (3)을 사용해서 LLR을 계산한다.

이 때 첫번째 부블록의 경우 식 (7), 두번째 부블록부터는 식 (6)과 같이 초기화한다.

$$A_0(s) = A_D(s) \tag{6}$$

$$A_0(s) = \begin{cases} 0 & if \quad s = 0 \\ -\infty & if \quad s \neq 0 \end{cases} \tag{7}$$

2-3 SOVA(Soft-Output Viterbi Algorithm)[9]

SOVA는 기존의 Viterbi 알고리즘에 사전정보를 추가하여 스테이트 메트릭을 계산하고 복호된 비트들에 대해 연성 출력 형태의 사후 확률 (posteriori probabilities)을 계산한다. 스테이트 메트릭은 Max-Log-MAP 의 순방향 메트릭을 구하는 방법으로 계산할 수 있다. 그리고 사후 확률은 식 (8)을 이용하여

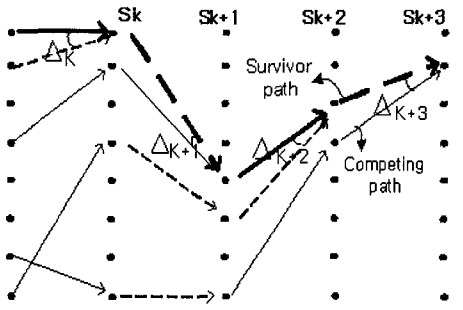

그림 3. SOVA 위한 트랠리스도
Fig. 3. Example of a trellis for SOVA.

계산할 수 있다. 이때 $\Delta_i(s)$는 각 스테이트의 차이 메트릭(difference metric)을 의미한다. 차이 메트릭은 스테이트 S에 도달하는 두 개의 가지의 가지 메트릭에 각각 이전 스테이트 메트릭을 더하고 그 차이를 계산하여 구할 수 있다. 이때 합이 더 큰 가지를 surviving path라고 하고, 다른 가지를 competing path 라고 한다. 그림 3에 SOVA 의 복호과정을 위한 트랠리스도를 도시하였다.

$$\Lambda_i \approx u_k \min_{\substack{i=k \ldots k+\delta \\ u_k \neq u_k^i}} \Delta_i(s) \tag{8}$$

식 (8)에서 u_k는 ML(maximum likelihood) 가지에 의해서 결정된 비트이고, u_k^i는 i 단계에서 선택되지 않으면서 ML 가지와 결합하는 가지에 대한 비트이다. δ는 역추적 깊이를 의미하며 길쌈부호의 구속장 (constraint length)의 다섯 배를 하여 결정된다. Λ_i는 ML가지를 $k+\delta$에서 k까지 역추적하여 u_k와 u_k^i의 비트가 다른 경우에 차이 메트릭을 갱신하여 계산할 수 있다.

Ⅲ. 부동 소수점을 이용한 turbo 복호기의 성능 분석

앞에서 설명한 turbo 복호 알고리즘의 성능을 해석하기 위해서 AWGN과 페이딩 채널 환경에서 모의실험을 수행하였다. 이때 turbo 부호기에 의해 부호화된 신호는 BPSK(binary phase shift keying) 변조기법을 이용하여 전송된다고 가정하면 다음과 같이 나타낼 수 있다.

$$y = a\,(2x - 1) + n \tag{9}$$

식 (9)에서 x, y는 각각 송신된 신호와 수신된 신호의 샘플링 값이고 a는 다경로 전송으로 인해서 발생하는 페이딩의 영향을 나타내는 진폭값을 나타내며 이 값의 확률 밀도 함수는 레일리(Rayleigh) 분포를 갖는다. 그리고 n은 평균이 0이고, 분산이 $N_0/2$인 가산 백색 잡음이다.

Turbo 복호기의 성능은 선택한 복호 알고리즘과 블록 크기, 부호율(code rate), 반복수(iteration) 등의 여러 요소에 의해서 결정된다. 먼저 블록 크기를 다양하게 하여 Log-MAP 복호 알고리즘의 성능 변화를 분석한다. 부동 소수점을 이용한 turbo 복호기의 성능 분석은 AWGN 채널환경에서 수행하였다. 신호대 잡음비가 0~1 dB까지는 블록의 크기 변화에 성능의 차이가 거의 없지만 1 dB 이후로는 성능의 차이가 크게 나타남을 그림 4에서 확인할 수 있다.

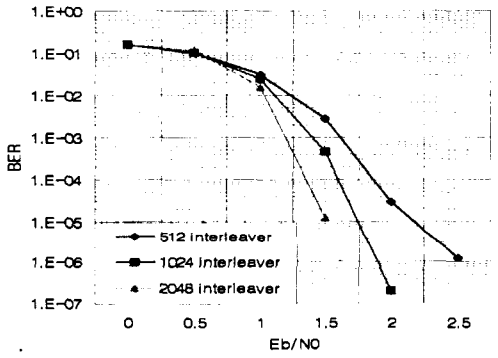

그림 4. Log-MAP 복호기의 프레임 크기에 따른 BER (부호율: 1/2, 반복수: 8)

Fig. 4. BER performance of Log-MAP decoder for various frame lengths(code rate: 1/2, iteration: 8).

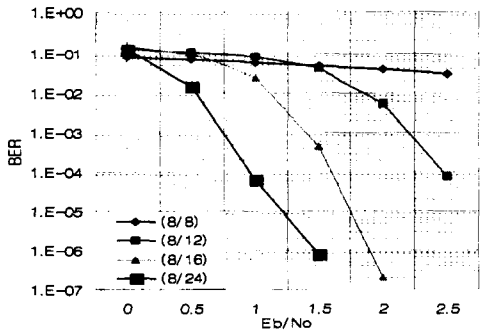

그림 5. Log-MAP 복호기의 부호율에 따른 BER(반복수: 8, 블록 크기: 1024)

Fig. 5. BER performance of Log-MAP decoder for various code rates(iteration: 8, block size: 1024).

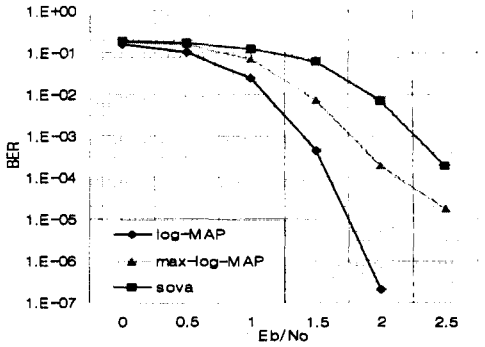

그림 6. 세 가지 복호 알고리즘에 대한 BER(부호율: 1/2, 반복수: 8, 블록 크기: 1024)

Fig. 6. BER performance of turbo codes for various decoding algorithms(code rate: 1/2, iteration: 8, block size: 1024).

다음으로 다양한 부호율에 대해 Log-MAP 복호 알고리즘의 성능 변화를 분석한다. Turbo 부호기에서는 패리티 비트를 펑쳐링하여 부호율을 조절할 수 있는데 다양한 부호율에 대한 성능 평가를 그림 5에 도시하였다. 비트 오율이 10^{-4}일 경우 부호율 8/24는 부호율 8/16보다 0.7 dB 정도 이득을 보며, 부호율 8/16은 부호율 8/12보다 0.8 dB 정도 이득을 보는 것을 확인할 수 있다.

다음에 나타낸 그림 6은 2장에서 설명한 turbo 부호기의 복호 알고리즘에 대한 성능을 보여준다. 이 때 Log-MAP 알고리즘은 식 (4)의 $f_c(\bullet)$을 이용하여 나타낸 결과이다. Max-Log-MAP과 SOVA가 Log-MAP과 비교해 성능이 저하됨을 그림 6에서 확인할 수 있다. 예를 들어 비트 오율(bit error probabilities)이 10^{-4}일 경우 Max-Log-MAP은 0.5 dB, SOVA는 1 dB 정도 성능이 저하됨을 알 수 있다.

Ⅳ. 고정 소수점을 이용한 turbo 복호기의 성능 분석

4-1 고정 소수점 구현

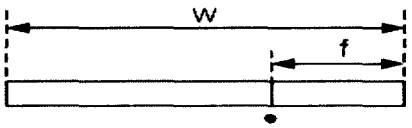

그림 7. 고정소수점 형식의 비트 수

Fig. 7. Fixed-point format.

그림 7에 도시한 것과 같이 w는 고정 소수점으로 나타내기 위한 전체 비트수이고, f는 소수점 이하 비트수라고 할 때 일반적으로 고정 소수점 정보를 (w, f)라고 나타내며 앞으로도 이 표현법을 사용한다. 복호기 내의 모든 데이터 값을 고정 소수점을 이용하여 구현할 때 비트수를 적게 할당하면 하드웨어상의 복잡도는 줄어들지만 복호화 성능은 저하된다. 반면 비트수를 많이 할당하게 되면 정확도가 높아져 복호기의 성능은 향상되지만 하드웨어의 복잡도는 커진다. 그러므로 고정 소수점을 이용하여 구현할 때 성능 저하를 줄이기 위해서는 데이터의 비트수를 적절하게 선택하는 것이 중요하다.

그림 8에 세 가지 신호 대 잡음비에 대해 Max-Log-MAP 복호 알고리즘의 성능 변화를 도시하였다. 먼저 소수점 이상의 비트 수를 고정시키고 소수점 이하 비트 수를 변경시킬 경우 전체 비트 수도 같이 변하게 하여 성능 분석을 하였다. 소수점 이하 비트 수가 커짐에 따라 성능이 향상됨을 확인할 수 있다. 소수점 이하 비트 수가 2일 때까지는 비트 오율의 값이 크게 차이를 보이며 향상되나, 2비트 이상부터는 비트 오율의 차이가 크지 않았다.

페이딩 채널 환경에서의 성능을 분석하기 위해서 앞에서 언급한 실험 조건을 가지고 Max-Log-MAP 복호 알고리즘에 대한 모의실험 결과를 그림 9에 도시하였다. 페이딩 채널 환경에서의 성능 변화는 가산 백색 잡음 채널 환경에서 실험한 그림 8의 성능 변화와 유사한 경향을 보이는 것을 확인할 수 있다.

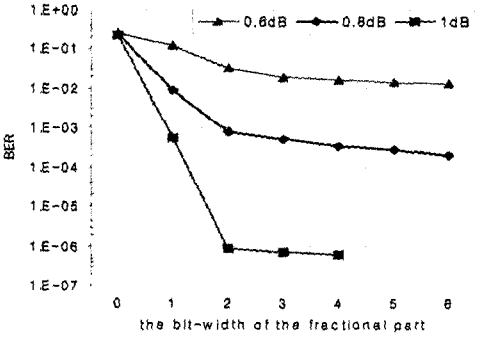

그림 8. 소수점 이하 비트 수 변화에 따른 BER(부호율은 1/3, 반복수: 8, 블록 크기는 4200)

Fig. 8. BER performance as a function of the bit width of the fractional part(code rate: 1/3, iteration: 8, block size: 4200).

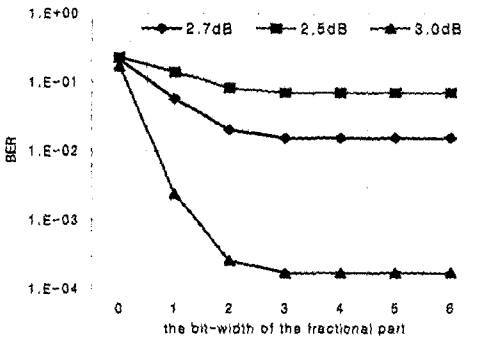

그림 9. 레일리 페이딩 채널 환경에서 소수점 이하 비트수 변화에 따른 BER

Fig. 9. BER performance as a function of the bit width of the fractional part over Rayleigh fading channels.

소수점 이하 비트 수가 2일 때까지는 비트 오율의 값이 크게 차이를 보이며 향상되나, 2비트 이상부터는 비트 오율의 차이가 크지 않았다.

데이터를 고정 소수점 정보로 바꿀 때 부호 비트를 포함하여 필요한 전체 비트수를 고정하고 소수점 이상의 비트수와 소수점 이하 비트 수를 변경하여 Max-Log-MAP 복호 알고리즘의 성능을 구하여 이를 그림 10에 도시하였다. 실험에 필요한 조건은 앞에서 언급한 값과 같다. 복호기 내의 순방향, 역방향 메트릭은 부호 비트를 포함하여 전체 비트수를 9비트, 가지 메트릭은 전체 비트수를 7비트로 할당하고 소수점 이하 자리 수는 입력 데이터의 소수점 이하 자리 수 변경과 같이 할당한다. 입력 데이터에 대해

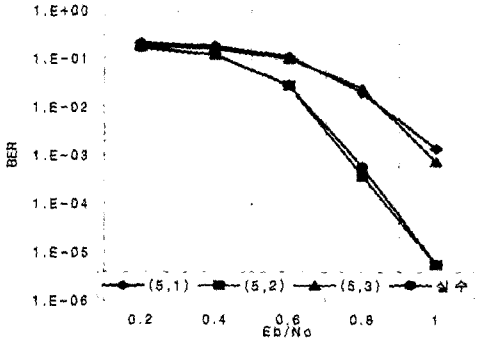

그림 10. 복호기 내의 모든 데이터의 전체 비트 수는 고정하고 소수점이하 자리 수 변화에 따른 BER

Fig. 10. BER performance for various bit width of the fractional part with fixing the whole bit width.

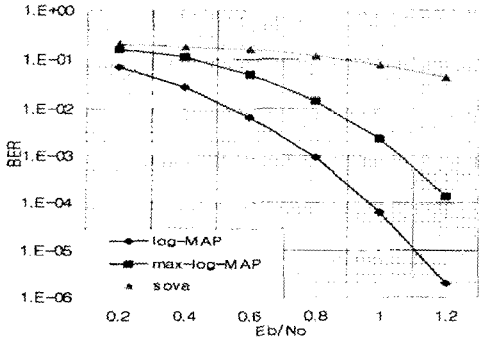

그림 11. 세 가지 복호 알고리즘에 대한 BER(부호율: 1/3, 반복수: 8, 블록 크기: 1024)

Fig. 11. BER performance of turbo codes for various decoding algorithms(code rate: 1/3, iteration: 8, block size: 1024).

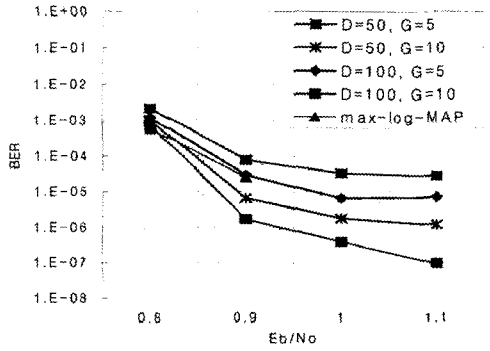

그림 12. Sliding window Max-Log-MAP의 BER(부호율: 1/3, 반복수: 8)

Fig. 12. BER performance of the sliding window Max-Log-MAP(code rate: 1/3, iteration: 8).

서는 (5,1), (5,2), (5,3)으로 비트수를 할당한다. 그림 10의 결과를 보면 소수점 이하 자리수가 2일 때 실수로 나타냈을 경우의 성능과 거의 유사함을 확인할 수 있다. (5,1)은 (5,2)보다 소수점 이하 비트 수가 적게 할당되어 데이터의 정확도가 떨어지기 때문에 0.2 dB 정도 SNR 손실이 있다. 그리고 (5,3)은 (5,2)보다 소수점 이상의 비트 수가 부족하여 오버 플로어가 발생할 수 있어 성능이 떨어진다. 앞의 두 실험을 통해 성능을 분석한 결과 고정 소수점 연산에서 전체 비트 수를 최적으로 고정시킬 경우 소수점 이하 2비트일 때의 결과가 실수 값을 이용한 결과와 근접함을 확인하였다.

그림 11은 2장에서 설명한 turbo 부호에 대한 여러 가지 복호 알고리즘을 고정 소수점 연산으로 구현했을 때의 성능을 보여준다. 복호기 내의 모든 데이터의 비트수는 위의 실험과 동일하다. 이때 Log-MAP 알고리즘은 식 (4)의 $f_c(\bullet)$을 이용하여 나타낸 결과이다. Max-Log-MAP 과 SOVA는 Log-MAP과 비교해 성능 저하가 크게 나타남을 그림 11에서 확인할 수 있다.

다음에 sliding window Max-Log-MAP 복호 알고리즘을 D와 G의 크기를 다양하게 하여 성능을 분석한다. 복호기 내의 모든 메트릭은 부호비트를 포함하여 전체 비트수를 16비트로 하여 성능을 분석한다. 복호기 내의 모든 메트릭은 부호비트를 포함하여 전체 비트수를 16비트로 할당하고, 입력으로 들어오는 계량값은 전체 비트수를 5비트로 할당한다. 그림 12에 프레임 크기는 5000, D 크기는 100과 50, G 크기를 다양하게 하여 비트 오율을 도시하였다. Sliding window Max-Log-MAP 알고리즘의 성능은 기존의 Max-Log-MAP 알고리즘을 고정 소수점을 이용하여 구현한 것과 비교해 성능이 저하됨을 확인할 수 있다. 그러나 D 크기와 G 크기를 크게 하면 성능이 향상됨을 그림 12에서 확인할 수 있다.

4-2 고정 소수점 구현에 용이한 Log-MAP 알고리즘

Log-MAP 알고리즘을 구현할 때 식 (4)에서 $f_c(\bullet)$ 함수 부분의 계산상의 복잡함을 줄이기 위해서 사용할 수 있는 방법으로 LUT를 이용하는 방법이 있다. 본 논문에서는 LUT를 사용하는 대신 고정 소수점 구현이 가능하고 계산량을 더욱 줄일 수 있는 방법을 제안한다. 고정 소수점 연산을 고려하고 성능의 손실을 최소화하는 방향으로 최적화를 수행한 결과 쉬프트 (shift) 연산만으로 구현 가능한 알고리즘을 제안하였다. 제안된 함수 $y(k)$는 다음과 같이 정의한다.

$$y(k) = \begin{cases} 3.25 - k/2, & 0 \le k < 1 \\ 2.75 - k/4, & 1 \le k < 4 \\ 2.0 - k/8, & 4 \le k < 9 \\ 0 & otherwise \end{cases} \quad (10$$

식 (10)에서 $y(k)$는 식 (4)의 $f_c(|x-y|) = \ln(+e^{-|x-y|})$를 고정 소수점으로 구현한 결과에 근한 값을 가지는 함수이다. 여기서 k는 $f_c(\bullet)$ 함수의

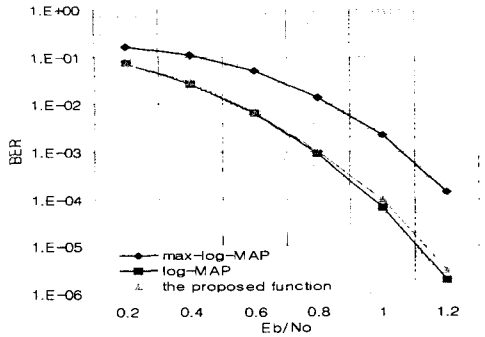

그림 13. 제안하는 알고리즘에 대한 BER(부호율은 1/3, 반복수는 8, 블록 크기는 1024)

Fig. 13. BER performance of the proposed function (code rate: 1/3, iteration: 8, block size: 1024).

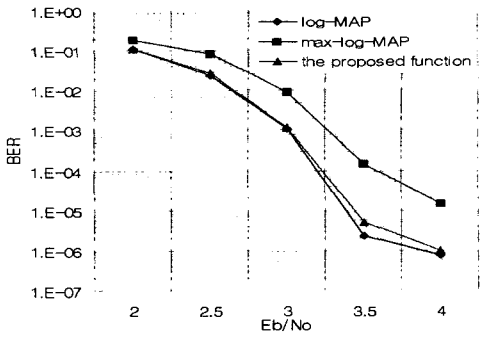

그림 14. 레일리 페이딩 채널 환경에서 제안하는 알고리즘에 대한 BER

Fig. 14. BER performance of the proposed function over Rayleigh fading channels.

자인 $|x-y|$를 그림 7에 나타낸 형식의 고정 소수점으로 구현하였을 때 생성되는 정수 값을 나타낸다. 즉 k는 고정 소수점으로 구현된 복호 과정에서 정수 값으로 나타나는 순방향, 역방향 그리고 사후 메트릭 각각의 차의 값에 절대 값을 취한 값이다. 제안하는 함수는 소수점 이하 2비트를 사용하여 구현한 결과이다.

제안된 기법의 성능을 분석하기 위해서 기존의 Log-MAP, Max-log-MAP, 고정 소수점 구현이 용이한 제안하는 알고리즘에 대한 모의실험 결과를 그림 13에 나타내었다. 복호기 내의 순방향, 역방향 메트릭은 (9,2), 가지 메트릭은 (7,2), 입력으로 들어오는 메트릭은 (5,2)로 할당한다. 그림 13을 살펴보면, 제안한 함수를 사용한 결과는 기존의 Log-MAP의 결과와 거의 근접한 성능을 보이고 Max-Log-MAP보다는 우수한 성능을 보이는 것을 확인할 수 있다.

그림 14에 기존의 Log-MAP, Max-Log-MAP, 고정 소수점 구현이 용이한 제안하는 알고리즘에 대하여 페이딩 채널 환경에서 수행한 모의실험 결과를 나타내었다. 실험 조건은 앞의 실험과 동일하다. 페이딩 채널 환경에서도 제안한 함수를 사용한 결과는 기존의 Log-MAP과 근접한 성능을 보이고 있음을 확인할 수 있다.

제안한 함수를 sliding window Log-MAP 알고리즘에 적용하여 고정 소수점 연산으로 구현한 결과의 비트 오율을 그림 15에 도시하였다. 복호기 내의 데이터 값에 대한 비트 수는 앞의 실험과 동일하다. 블록 크기는 4200, D의 크기는 100, G의 크기는 10으로

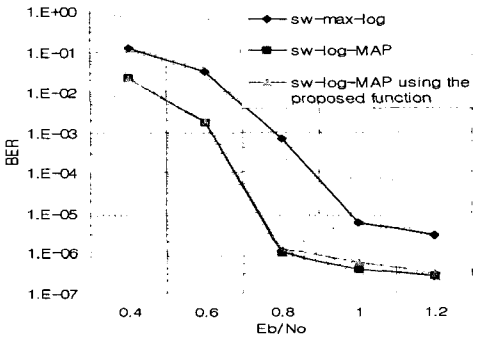

그림 15. 제안한 함수를 적용한 sliding window Log-MAP의 BER(부호율: 1/3, 반복수: 8)

Fig. 15. BER performance of sliding window Log-MAP using proposed function(code rate: 1/3, iteration: 8).

하였다. 그림 13의 경우와 유사하게, sliding window Log-MAP 알고리즘에도 제안하는 함수를 사용한 경우, 기존의 방법을 사용한 결과와 근접한 비트 오율을 달성함을 확인할 수 있다.

V. 결 론

본 논문에서는 3GPP 표준 오류 정정 부호 기법 중의 하나인 turbo 부호의 복호기를 Log-MAP, Max-Log-MAP, sliding window Log-MAP, sliding window Max-Log-MAP, SOVA로 구현하고, 각각의 알고리즘에 대해 부동 소수점 연산(floating-point arithm etic)과 고정 소수점 연산(fixed-point arithmetic)을 이용하여 AWGN(additive white Gaussian noise)과 페이딩 채널 환경에서의 성능을 해석하였다. 고정 소수점

연산을 이용하여 구현하였을 경우 소수점 이하 비트 수가 2 이상일 때 성능이 부동 소수점 연산에 근접함을 입증하였다. 그리고 Log-MAP 알고리즘의 계산상의 복잡함을 줄이기 위해서 사용할 수 있는 방법으로 LUT를 사용하는 방법 대신 고정 소수점 연산을 이용하여 간략하게 구현할 수 있는 방법을 제안하였다. 제안한 함수는 고정 소수점 연산을 고려하고 성능의 손실을 최소화하는 방향으로 최적화를 수행한 결과 쉬프트(shift) 연산만으로 구현 가능하다. 이 방법은 기존의 Log-MAP 알고리즘의 성능과 근접하는 성능을 보임을 입증하였다. 또한 이 방법을 메모리 효율을 높인 sliding window Log-MAP 알고리즘에도 적용하여 해석한 결과, 기존의 Log-MAP을 사용한 결과와 근접한 비트 오율을 나타냄을 모의실험을 통하여 확인하였다.

참 고 문 헌

[1] N. Farber, B. Girod and J. Villasenor, "Extension of ITU-T recommendation H.324 for error-resilient video transmission", *IEEE Commun. Mag.*, vol. 36, pp. 120-128, Jun. 1998.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shnnon Limit Error-Correcting coding and decoding :Turbo codes (1)", *Proc. ICC*, Geneva, Switzerland, May 1993.

[3] Sangho Yoon, "Turbo decoder implementation", *Vehicular Technology Conf.*, vol. 3, pp. 7-11, Oct. 2001.

[4] M. Marandian, J. Fridman, Z. Zvonar and M. Salehi, "Performance analysis of turbo decoder for 3GPP standard using the sliding window algorithm", *Personal, Indoor and Mobile Radio Communication, 2001 12th IEEE International Symposium on*, vol. 2, Oct. 2001.

[5] S. Benedetto, D. Divsalar, G. Montorsi and F. Pollara, "Soft-Output Decoding Algorithm in Iterative Decoding of Turbo Codes", *TDA Progress Report*, 1996.

[6] G. Montorsi, S. Benedetto, "Design of Fixed-Point Iterative Decoders for Concate nated Codes with Interleavers", *IEEE J. on Select. Areas in Commun.*, vol. 19, no. 5, May 2001.

[7] M. C. Valenti, "Ierative detection and decoding for wireless communications", Ph. P thesis, Blacksbug, Virsinia, Sep. 1998.

[8] 3GPP TS 25.212, 3rd Generation Partnership Project Technical Specification Group Radio Access Network, Multiplexing and channel coding(FDD), (Release 4).

[9] L. Hanzo, T. H. Liew and B. L. Yeap, *Turbo coding, Turbo equalization and space-time coding for transmission over fading channels*, John Wiley & Sons, pp. 107-171, 2003.

신 나 나

2003년 2월: 가톨릭대학교 컴퓨터·전자공학부 (공학사)

2003년 3월~현재: 가톨릭대학교 컴퓨터공학과 정보통신전공 석사과정

[주 관심분야] 영상통신, turbo code 및 space-time code

이 창 우

1988년 2월: 서울대학교 제어계측공학과 (공학사)

1990년 2월: 서울대학교 제어계측공학과 대학원 (공학석사)

1996년 2월: 서울대학교 제어계측공학과 대학원 (공학박사)

1996년 3월~1997년 8월: 삼성전자 신호처리연구소 선임연구원

1997년 9월~2001년 8월: 가톨릭대학교 컴퓨터전자공학부 조교수

2001년 9월~현재: 가톨릭대학교 정보통신전자공학부 부교수

[주 관심분야] 영상통신, 영상압축, turbo code