터보 부호에서 낮은 복잡도를 갖는 효율적인 반복복호 제어기법과 새로운 인터리버

정회원 김 순 영*, 장 진 수*, 성 락 주**, 이 문 호**

Efficient Iteration Control Method with low complexity and New Interleaver for Turbo Codes

Soon Young Kim*, Jin Su Chang*, Nark Joo Seong**, Moon Ho Lee** ***Regular Members***

요 약

본 논문에서는 새로운 터보 인터리버와 복호화 과정에서 낮은 복잡도를 갖는 효율적인 반복복호 제어기법을 제안한다. 터보코드는 반복 복호수와 인터리버 크기가 증가할수록 성능이 향상된다는 것은 잘 알려진 사실이다. 그러나 인터리버 크기가 증가하면 복호 과정에서 지연과 계산량이 증가하게 된다. 따라서 새로운 효율적인 인터리버를 제안한다. 제안 인터리버는 매직 매트릭스 특성을 이용함으로서 낮은 구현 복잡도를 가지면서 GF, Mother 등 기존의 IMT-2000에 제안된 터보 인터리버와 비슷한 성능을 보인다. 또한 터보 코드의 복호 과정에서 반복 복호수를 증가하면 성능이 향상되지만 어느 정도의 반복 복호수 이상에서는 성능 향상이 거의 나타나지 않는다. 따라서 본 논문에서는 최대반복 복호수를 수신 데이터를 이용, SNR(signal to noise ratio)을 추정하여 반복 복호수를 가변적으로 미리 설정(preset)함으로써 반복 복호수를 제어하는 기법을 제안한다. 제안구조는 터보 부호의 문제점 중 하나인 복호 계산량과 지연을 성능 저하 없이 효율적으로 감소시킬 수 있다.

ABSTRACT

In this paper, we propose a new turbo interleaver and an efficient iteration control method with low complexity for turbo decoding. Turbo codes has better performance as the number of iteration and the interleaver size increases. However, as the interleaver size is increased, it require much delay and computation for decoding. Thus we propose a new efficient turbo magic interleaver using the Magic square matrix. Simulation results show that the proposed interleaver realizes a good performance like GF, Mother interleaver proposed to IMT-2000. And as the decoding approaches the performance limit, any further iteration results in very little improvement. Therefore, we propose an efficient algorithm of decoding that can reduce the delay and computation. Just like the conventional stop criterion, it effectively stop the iteration process with very little performance degradation.

Ⅰ. 서 론

터보코드는 최근 ITU 등에서 IMT-2000 차세대 이동통신의 고속 데이터 전송용 채널 코드의 표준으로 채택되었다. 1993년 발표된 터보코드는 RSC (recursive systematic convolutional) 부호를 병렬로 연접하여 부호화하며 준 최적 복호 방법인 반복 복호를 통하여 복호 동작을 수행하게 된다. 또한 터보

* 전북대학교 컴퓨터공학과 (young@mdmc.chonbuk.ac.kr),
** 전북대학교 정보통신연구소 (moonho@moak.chonbuk.ac.kr)
논문번호 : 00057-0214, 접수일자 : 2000년 2월 14일

코드는 인터리버의 크기가 크고 반복 복호가 충분히 수행되었을 때 비트 에러율 관점에서 샤논 한계에 근접하는 우수한 성능을 보인다. 그러나 많은 연산량에 따른 복잡성의 증가, 그리고 인터리버와 반복복호수에 따른 지연과 실시간 처리의 어려움이라는 문제점을 안고 있다.

본 논문에서는 터보 코드의 큰 문제점 중 하나인 반복복호에 따른 계산량과 지연의 증가에 관심을 가지며 이를 해결하는 구조를 제시한다. 기존 구조는 터보 부호의 복호기 입력전에 미리 정해진 반복복호수에 따라 복호를 수행한다. 그러나 실제적으로 가변적 채널 환경에 따라 반복 복호수를 어느 정도 진행하면 그 이상의 반복 복호수에서는 부호이득이 매우 적게 되며, 임의의 반복 복호수에서 이미 요구하는 성능을 얻었다 하더라도 미리 정해진 반복복호 수만큼 복호를 수행함으로써 계산량과 지연이 크게 된다. 이러한 문제점을 해결하는 방법으로 복호기에서 미리 정해진 반복 복호수가 아닌 가변적으로 반복 복호를 멈추는 정지 기준(stop criterion) 구조가 발표되었다[1][5].

기존의 가변적 정지 기준 구조의 공통점은 복호기의 출력단에서 어떤 기준값(예를 들어 LLR값이나 외부정보)을 사용하여 반복복호를 멈추는 기준으로 이용하고 있다. 그러나 이러한 구조는 한 프레임에서 반복복호 때마다 기준 값을 측정해야하는 계산상의 복잡도를 가진다. 따라서 본 논문에서는 이러한 문제점을 해결하는 효율적인 기법을 제안하고, 기존 구조와 비교, 분석한다. 또한 터보코드에서 부호화한 비트들의 상관성이 증가할수록 복호화 동작을 위한 많은 정보를 얻을 수는 있겠지만 에러들에 대한 상관성이 증가하게 된다. 따라서 에러들 사이의 상관성을 제거하기 위해 인터리버를 사용하며 인터리버 크기에 따라 복잡성이나 지연의 문제도 수반하게 된다. 인터리버의 역할은 일반적인 채널 환경에서는 연집 에러를 랜덤 에러로 전환하여 주는 역할을 하게 되며 터보 코드에서는 일반적으로 같은 생성 다항식을 사용하므로 이러한 경우에 입력의 순서를 재배열하는 인터리빙 과정이 필요하다. 터보 코드의 성능을 향상시키기 위해서는 기존의 구조적 인터리빙 방식이 아닌 랜덤 인터리버를 사용해야 하는데 이러한 인터리빙 방식의 문제점은 각각의 전송율에 따라 모든 인터리빙 어드레스를 저장해야 한다는 점이다. 따라서 계산에 의한 어드레스를 생성하는 알고리즘이 필수적이다. 또한 인터리버의 설계 시 인터리빙 어드레스를 구성하기 위하여 가능한 적은 메모리를 가지고서 실현 가능해야 하며, 다양한 전송율에서도 쉽게 적용이 되어야 한다. 따라서 본 논문에서는 간단한 계산에 의하여 인터리빙 어드레스를 생성하며 다양한 전송율에서도 쉽게 적용이 가능한 매직 매트릭스를 이용한 인터리버를 제안한다.

먼저 Ⅱ장에서는 반복복호 제어기법에 관하여 기존의 방법을 분석하고, 낮은 복잡도를 갖는 제어기법을 제안하여 그 성능을 기존의 방법과 비교 분석한다. Ⅲ장에서는 터보코드의 내부 인터리버의 설계 개념을 설명하고 새로운 인터리버의 구성과정을 제시하며, 그의 성능을 컴퓨터 시뮬레이션을 통하여 기존의 인터리버와 비교 분석한다. 마지막으로 Ⅳ장에서 본 논문의 결론을 맺는다.

Ⅱ. 제안한 반복 복호수 제어기법

2.1 기존의 반복복호 제어기법

터보 코드는 반복복호수를 증가하면 성능이 향상된다는 것은 잘 알려진 사실이다. 그러나 반복복호수가 증가할수록 복호를 위한 계산량과 지연이 증가하게 된다는 단점을 가지게 된다. 그러므로 복호기에서 미리 정해진 반복복호수가 아닌 가변적 반복 복호를 할 수 있다면 계산량과 지연을 줄일 수 있을 것이다. 반복 복호를 위해 교차 엔트로피를 사용한 효과적인 반복복호 정지기준 내용을 정리하면 다음과 같다[1]. m 번째 복호기의 출력에서 두 개의 연속적인 복호 동작의 사후확률(*a posteriori*) 분포가 있다고 가정하자. $P(\hat{u})$는 랜덤변수로서 복호기 2의 사후확률분포이고, $Q(\hat{u})$는 복호기 1의 사후확률분포라고 정의하여 다음 식 (1)과 같이 두 분포의 차의 측정치로서 정의 할 수 있다[1].

$$E_P\left\{\log\frac{P(\hat{u})}{Q(\hat{u})}\right\} \tag{1}$$

여기서 E_P는 분포 $P(\hat{u})$상에서 기대값 연산자(expectation operator)를 나타낸다. 통계적으로 독립을 가정하면 다음 식 (2)를 얻을 수 있다.

$$\log\frac{P(\hat{u})}{Q(\hat{u})} = \sum_k \log\frac{P(\hat{u}_k)}{Q(\hat{u}_k)} \tag{2}$$

복호기 1과 2로 구성되어 있는 첫 번째 반복에서 두 개의 연속적인 반복 복호수를 $(i-1)$과 (i)라하고 복호기의 soft 출력을 다음과 같이 정의하자.

$$L_Q^{(i)}(\hat{u}_k) = L_c \cdot y_k + L_{e2}^{(i-1)}(\hat{u}_k) + L_{e1}^{(i)}(\hat{u}_k)$$

$$L_P^{(i)}(\hat{u}_k) = L_c \cdot y_k + L_{e1}^{(i)}(\hat{u}_k) + L_{e2}^{(i)}(\hat{u}_k). \quad (3)$$

식(3)에서 $\hat{u}_k$는 추정된 k 번째 정보 비트이며, y_k는 k 번째 수신 비트, $L_{e1}^{(i)}(\hat{u}_k)$는 i 번째 반복 복호에서 첫 번째 복호기에서 얻은 k 번째 정보 비트에 대한 외부정보(extrinsic information)이며 다음 복호기의 사전확률(*a priori*) 값으로 사용된다. 또한 L_c는 채널 신뢰도를 나타낸다. 식 (3)에서 두 식의 차이는 식 (4)와 같이 되며

$$\begin{aligned} L_P^{(i)}(\hat{u}_k) - L_Q^{(i)}(\hat{u}_k) &= L_{e2}^{(i)}(\hat{u}_k) - L_{e2}^{(i-1)}(\hat{u}_k) \\ &= \Delta L_{e2}^{(i)}(\hat{u}_k) \end{aligned} \quad (4)$$

결정이 더 이상 변하지 않을 때 예를 들어 $sign\,(L_P^{(i)}(\hat{u}_k)) = sign\,(L_Q^{(i)}(\hat{u}_k)) = \hat{u}_k^{(i)}$이 면 서 $\Delta L_{e2}^{(i)}$가 $\hat{u}_k^{(i)}$와 같은 sign을 가지고 크기가 1보다 작을 때 식 (1)은 다음과 같이 간략화 될 수 있다.

$$E_P\left\{\log\frac{P(\hat{u})}{Q(\hat{u})}\right\} \approx \sum_k \frac{|\Delta L_{e2}^{(i)}(\hat{u}_k)|^2}{\exp(|L_Q^{(i)}(\hat{u}_k)|)} \quad (5)$$

비록 likelihood 값 사이의 통계적 독립의 가정은 약간의 반복 복호 뒤에는 정확하지 않지만 반복 복호를 위한 정지 기준으로 다음 식(6)을 사용할 수 있다.

$$T(i) = \sum_k \frac{|\Delta L_{e2}^{(i)}(\hat{u}_k)|^2}{\exp(|L_Q^{(i)}(\hat{u}_k)|)} < threshold \quad (6)$$

반복 복호를 중단하기 위한 적당한 기준 값은 $T(1)\cdot 10^{-3}$ 정도가 적당하다고 알려져 있다[1]. 교차 엔트로피를 사용한 정지 기준 구조는 복잡한 계산을 포함하고 있어 구현상의 어려움이 있게 된다. 이를 해결하는 구조로 두 가지 정지 기준 구조가 제안되었다. 첫 번째는 $L_{e2}^{(i)}(\hat{u}_k)$ 와 $L_{e2}^{(i-1)}(\hat{u}_k)$의 부호가 반복 복호에 따라 변하는 개수 $C(i)$를 구하고, $C(i) \le (0.005 \sim 0.03)N$ 이면 복호를 멈춘다. 이를 SCR (sign change ratio) 정지 기준이라 한다. 두 번째는 $L_{e2}^{(i)}(\hat{u}_k)$ 와 $L_{e2}^{(i-1)}(\hat{u}_k)$의 부호가 정보 블록의 모든 비트에 대해 같으면 복호를 중지하는 기준으로 하는데 이를 HDA (hard-decision -aided) 정지 기준이라 한다. 이 구조를 비교해 보면 같은 평균 반복복호수에서 교차 엔트로피 방식이 (5N-1) 실수 연산과 (N+2)의 실수 메모리가 필요하지만, SCR 방식은 (3N-1)정수 연산과 (N+2)의 정수 메모리만 필요하므로 계산량 및 하드웨어 복잡도가 감소하게 된다는 장점을 가진다. 또한 HDA 구조는 같은 비트오율에서 낮은 SNR에서 중간 정도의 SNR에서는 평균반복 복호수가 감소하는 장점이 있으나 높은 SNR에서는 교차 엔트로피 방식이나 SCR 방식보다 비효율적이다[5].

2.2 제안한 SNR 추정에 의한 반복 복호수 구조

본 논문에서는 터보 부호의 문제점 중 하나인 반복 복호수에 따른 계산량과 지연을 효과적으로 줄이는 방법으로 [4]에서 제안한 구조를 적용하여 SNR을 추정하고 가변적 정지기준을 접목한 반복복호수 preset 구조를 제안한다. 이를 위해 수신데이터 값을 이용하여 SNR을 추정하여 모의실험을 통해 이미 얻어진 결과를 저장한 look up table을 이용하여 요구하는 성능을 만족하는 반복복호수를 가변적으로 미리 설정함으로써 계산량과 지연을 효율적으로 감소시킨다.

수신데이터를 이용하여 SNR 추정 방법은 다음과 같다. 터보 부호는 AWGN 채널에서 최소 에러를 만들어 내는 채널상태 정보 (입력잡음분산)에 의존한다. 만약 수신된 데이터가 복호기 입력 전에 채널상태 값(입력잡음분산)을 정확히 알 수 있다면 복호기 입력 전에 요구하는 성능을 만족하도록 반복 복호수를 미리 설정하면 계산 복잡도를 줄일 수 있을 것이다. 그러나 대부분의 연구 결과들은 채널상태정보를 정확히 알고 있다는 가정하에 결과를 나타낸다. 채널상태정보를 추정하는 연구 결과들이 몇 가지 있다[2]-[4]. 그러나 이와 같은 연구 결과들은 독립적으로 이루어 졌으며 위에서 언급한 두 가지 방법(SNR 추정방법과 적응적 정지 기준 구조)을 접목한 연구결과들은 발표되지 않았다. 연구 결과 중 우리가 관심을 갖는 결과는 *Todd A. Summers*의 잘못 추정된 SNR의 영향과 정확한 잡음분산 추정을 위한 방법이다[4]. 본 논문에서 적용한 수신단에서 SNR을 추정하는 방법은 다음과 같다. 부호 비트를 전송하기 위해 AWGN 채널에서 BPSK (binary phase shift keying)변조를 가정하면 수신된 데이터는 식 (7)과 같이 표현할 수 있다.

$$r_n = \pm\sqrt{E_s} + n_n = \pm\mu + n_n \quad (7)$$

n_n은 평균은 0이며 분산은 $\sigma^2 = N_o/2$인 가우시안 잡음이다. 또한 식(7)에서 심볼 에너지와 비트당 에너지의 관계는 $E_s = E_b R$과 같으며, R은 부호율을 나타낸다. 식(7)에서 SNR을 추정하기 위해 식(7)의 제곱의 평균과 절대값의 평균을 고려하면 다음 식과 같다.

$$E(r_n^2) = E_s + \sigma^2 \tag{8}$$

$$E(|r_n|) = \sigma\sqrt{\frac{2}{\pi}}e^{-(E_s/2\sigma^2)} + \sqrt{E_s}erf\left(\sqrt{\frac{E_s}{2\sigma^2}}\right) \tag{9}$$

식(8)과 (9)의 관계를 다음과 같이 표현하면

$$\frac{E(r_n^2)}{[E(|r_n|)]^2} = \frac{1+\frac{E_s}{\sigma^2}}{\left\{\sqrt{\frac{2}{\pi}}e^{-(E_s/2\sigma^2)} + \sqrt{\frac{E_s}{\sigma^2}}erf\left(\sqrt{\frac{E_s}{2\sigma^2}}\right)\right\}^2}$$

$$= f\left(\frac{E_s}{\sigma^2}\right) = f(\beta) \tag{10}$$

식 (10)에서 $\beta = \mu^2/\sigma^2 = 2E_s/N_o$를 나타내며 식(10)의 평균값의 비를 변수 z라 정의하면 다음 식과 같다.

$$z = E(r_n^2)/[E(|r_n|)]^2 \tag{11}$$

즉 식 (10)과 (11)를 이용하여 β를 알 수 있으며 SNR을 추정할 수 있게 된다. 위 식에서 얻어진 z를 이용하여 다음식에서 β를 구한다. 다음식 (12)는 z와 β의 관계를 2차 방적식으로 풀어서 얻어진 결과식이다[4].

$$\beta \approx -34.0516z^2 + 65.9548z - 23.6184 \tag{12}$$

가변적으로 반복 복호수를 복호기 입력전에 미리 설정하기 위해서는 성능 감소가 거의 없는 최적의 반복 복호수를 설정하는 것이 필요하다. 이를 위해 본 논문에서는 첫 번째 MAP 복호기와 두 번째 MAP 복호기의 LLR 값에서 외부정보의 분산을 측정하고, 이를 이용하여 최적의 반복 복호수를 설정하고자 한다.

먼저, 수신측에서 터보 디코더는 수신 데이터 Y를 수신한 이후에는 수신 심볼에 기반한 가장 근사화한 입력 값 u_k을 결정하게 된다. 임의 시점 k에서 입력 심볼에 대한 우도 근사화 값(log-likelihood ratio)은 다음과 같이 표현된다.

$$L(\hat{u}_k) = \log\left[\frac{P(u_k = 1 \mid Y)}{P(u_k = 0 \mid Y)}\right] \tag{13}$$

터보코드에서 우도근사화값은 다음과 같이 3개의 성분으로 분할 할 수 있다.

$$L(\hat{u}_k) = L_{systematic} + L_{apriori} + L_{extrinsic} \tag{14}$$

윗 식에서 $L_{systematic}$ 은 시간 k에서 systematic 심볼에 대한 값을 나타내며 $L_{apriori}$는 입력 비트 u_k에 대한 APP(*a priori information*)이다. 그리고 $L_{extrinsic}$은 입력 비트 u_k와 관계하여 나타나는 새로운 정보인 외부 정보(*extrinsic information*)를 나타내며 시간 k에서 systematic 데이터를 제외한 모든 데이터와 패리티 비트들에 대한 정보를 가지고 있게 된다. 외부정보는 반복복호과정에서 성능 향상을 시키는 중요한 역할을 하게 된다.

i 번째 반복복호에서 $L_m^{(i)}(\hat{u}_k)$ 과 $Le_m^{(i)}(\hat{u}_k)$를 디코더 m에서의 정보 $\hat{u}_k$에 대한 각각의 LLR값, 외부정보 값이라고 하면 다음과 같이 윗식을 분리할 수있다.

$$L_1^{(i)}(\hat{u}_k) = L_c \cdot y_k + Le_2^{(i-1)}(\hat{u}_k) + Le_1^{(i)}(\hat{u}_k) \tag{15}$$

$$L_2^{(i)}(\hat{u}_k) = L_c \cdot y_k + Le_1^{(i)}(\hat{u}_k) + Le_2^{(i)}(\hat{u}_k) \tag{16}$$

여기서 $L_c = \frac{2}{\sigma^2}$ 로서 수신채널에 대한 정보값이다. 외부정보 $Le_m^{(i)}(\hat{u}_k)$를 구하면

$$Le_1^{(i)}(\hat{u}_k) = L_1^{(i)}(\hat{u}_k) - \{L_c \cdot y_k + Le_2^{(i-1)}(\hat{u}_k)\} \tag{17}$$

$$Le_2^{(i)}(\hat{u}_k) = L_2^{(i)}(\hat{u}_k) - \{L_c \cdot y_k + Le_1^{(i)}(\hat{u}_k)\} \tag{18}$$

본 논문에서 제안하는 터보코드의 반복복호 제어기법은 위 (17)(18)식에서 반복복호시에 성능 향상의 중요한 역할을 하게 되는 외부 정보값 $Le_m^{(i)}(\hat{u}_k)$에 대한 분산값을 이용한다. 따라서 각각의 분산값을 구하면 다음식과 같다.

$$Var[Le_1^{(i)}] = \frac{\sum_{k=0}^{N-1} Le_1^{(i)}(\hat{u}_k)^2}{N} - \left(\frac{\sum_{k=0}^{N-1} Le_1^{(i)}(\hat{u}_k)}{N}\right)^2 \tag{19}$$

$$Var[Le_2^{(i)}] = \frac{\sum_{k=0}^{N-1} Le_2^{(i)}(\hat{u}_k)^2}{N} - \left(\frac{\sum_{k=0}^{N-1} Le_2^{(i)}(\hat{u}_k)}{N}\right)^2 \tag{20}$$

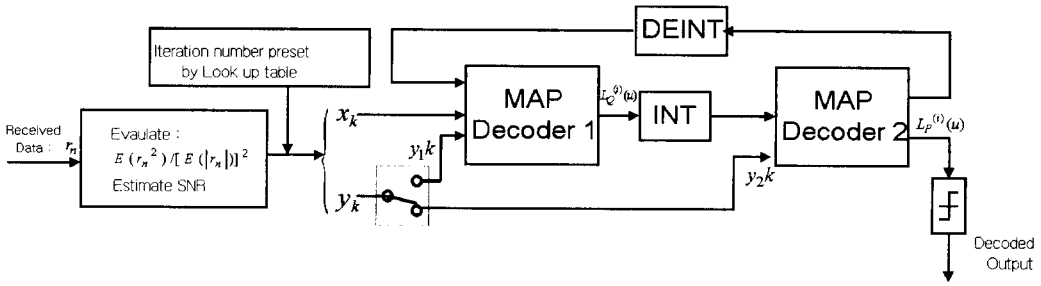

그림 1. 제안한 SNR 추정에 의한 반복 복호수 preset 구조.

먼저 수신 데이터를 이용하여 채널 상태정보를 추정한다. 그리고 이 값을 이용하여 최대 반복복호 횟수를 미리 결정하고 반복복호 과정을 시작한다. 본 논문에서는 첫 번째 MAP 디코더와 두 번째 MAP 디코더에서 외부 정보값의 분산값을 측정하고 이들값의 차, 즉 외부정보값의 증가량을 Delta variance라고 정의한다. 최대 반복복호 횟수는 이 Delta variance에 기반하여 이루어 지며, i 번째 반복복호에서 Delta variance는 다음 식으로 표현된다.

$$\Delta Var([L_e(i)]) = Var(L_{e2}^{(i)}) - Var(L_{e1}^{(i)}) \qquad (21)$$

그림 2와 3은 반복복호 횟수 10, 구속장 3, 코드율 1/2에서 반복복호 횟수에 따른 $\Delta Var[L_e(i)]$값과 각각의 비트에러율을 나타내고 있다.

$\Delta Var[L_e(i)]$ 값이 커짐에 따라 에러율은 작아진다. 즉 성능은 향상된다. 그러나 그림2에서 볼 수 있듯이 최대 delta variance값을 넘어서게 되면 에러마루 현상이 나타나서 반복복호를 계속 진행하여도 더 이상의 성능 향상은 미미하게 된다. 따라서 반복복호를 최대 delta variance값이 나타나는 시점에서 멈추자는 것이 본 논문에서 제안하는 임계치 결정 방안이다. 즉, 채널 값에 따른 $\Delta Var[L_e(i)]$의 최대값에 준하여 최대 반복복호 횟수를 미리 정해 놓고 반복복호를 진행한다

2.3 시뮬레이션 및 결과분석

제안한 SNR 추정에 의한 반복 복호수 preset 구조의 블록도를 그림 1에 나타내었다. 모의실험 조건은 AWGN 환경에서 부호율 1/2, 생성다항식 (7,5)인 구속장 (K)는 3이며, 프레임 크기는1024이다. 또한 PN 인터리버를 사용하였고, 복호 알고리즘은 MAP 알고리즘을 사용하였으며 최대 반복복호 수는 10회로 하였다. 모의 실험을 통해 반복 복호수에 따른 성능을 그림 4에 나타내었다. 그림에서 보듯이 제안한 반복복호 제어기법이 Cross entropy 기법에 비하여 약간 성능이 우세하며 반복복호 횟수면에서 표 1에서 보듯이 약간의 이득이 있다.

반복복호 제어기법을 적용하지않고 최대 반복복호횟수 10회를 수행할 경우가 성능이 가장 좋지만 반복횟수면에서 표 1에서 처럼 상당한 이득을 볼수 있다. 특히 2dB 이상에서는 거의 1/3로 반복복호 횟수를 줄일 수 있다.

또한 표 1에서는 실제 SNR과 추정된 SNR을 보여주고 있는데, 평균적으로 거의 근접한 추정을 보여주고 있으며 각 프레임에서 SNR이 잘못 추정되었다 하더라도 AWGN 채널에서 -3dB~3dB의 범위라면 성능에 큰 영향은 주지 않는 것으로 발표되었다[2][4]. 또한 교차 엔트로피 정지 기준 방식과 제안 방식의 평균 반복복호수를 나타내었다.

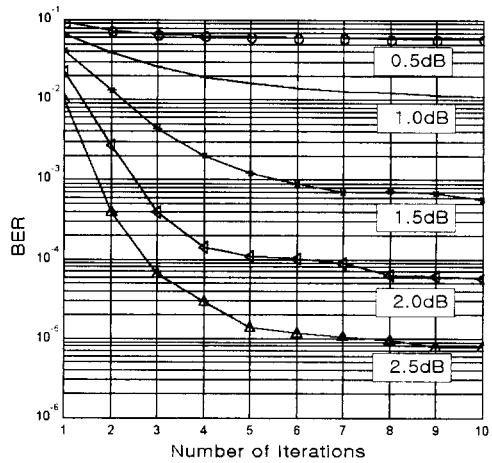

그림 2. 각 SNR에서 반복 복호수에 따른 Bit Error Probability.

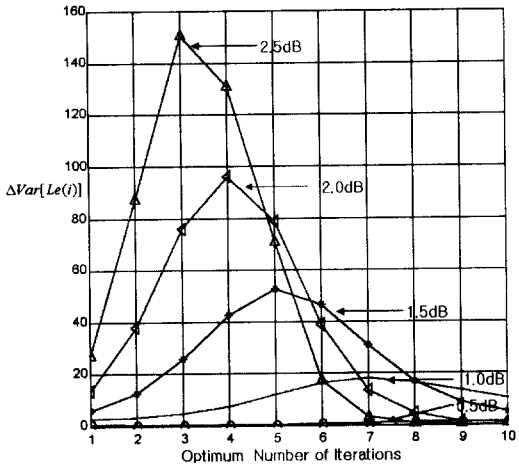

그림 3. 반복복호에 따른 $\Delta Var[L_e(i)]$.

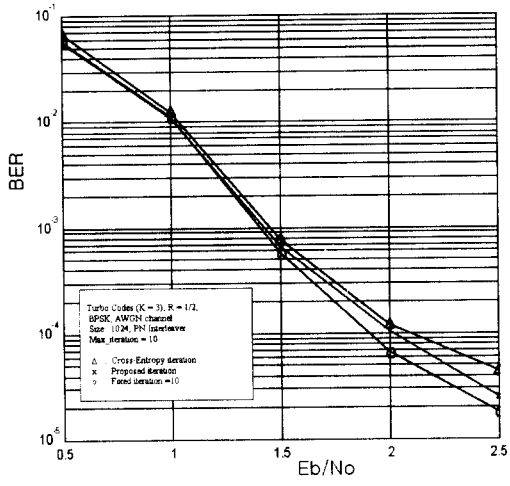

그림 4. SNR에 따른 제안구조의 BER 성능.

표 1. 실제 SNR과 추정된 SNR & 평균 반복복호수

True SNR	Estimated SNR	Cross-entropy 평균반복복호수	제안된 기법의 평균반복복호수
0.5	0.3952	9.00	8.72
1.0	0.8314	8.05	7.51
1.5	1.3865	5.28	5.14
2.0	1.9555	3.77	3.57
2.5	2.5301	3.22	3.12

Ⅲ. 터보 매직 인터리버

3.1 터보 코드와 인터리버

일반적인 채널 환경에서 인터리버의 역할은 연집 에러를 랜덤 에러로 전환하여 주는 역할을 하게 된다. 터보 코드에서 첫번째 구성부호기의 낮은 무게(low-weight)를 출력하는 입력열의 특정한 패턴이 인터리버를 통해서도 그대로 나타나게 되면, 두 번째 구성 부호기에서도 낮은 무게의 패리티 열이 출력되며 전체적으로 낮은 무게의 부호어가 생성되어 터보 부호의 성능이 열화 된다. 따라서 터보코드의 인터리버는 상관 관계가 있는 정보를 효과적으로 상관 관계가 없는 정보로 전환하기 위해서 정보 비트의 입력순서를 재배열하여 에러 패턴을 깨주는 역할을 하게 된다. 즉, 하나의 RSC 부호기의 입력 패턴이 작은 거리 특성을 주는 경우, 다른 하나의 RSC 부호기로의 입력 패턴이 큰 거리 특성을 줄 수 있도록 입력 패턴에 대한 치환(permutation)을 수행하는 것이다. 터보 부호어의 무게(weight)는 입력 정보열의 무게와 각각의 구성부호기 출력에서 나온 부호화된 열의 무게 합으로 정의되며, 자유거리(free distance)는 0 부호어를 제외한 가능한 모든 부호어 무게에서 최소 해밍 무게(Hamming weight)가 된다. 좋은 성능의 터보 부호는 자유거리를 크게 하여야 하며, 따라서 보다 큰 해밍무게를 갖는 입력 데이터 열이 인터리버 통과 후 동일한 데이터 열을 재 발생시킬 확률이 작아진다. 따라서 터보 인터리버의 설계 기준 중에 하나는 입력 시퀀스에 대한 부호어의 무게가 크게 되도록 하는 것이다[11]. 부호기 측면에서 터보 코드의 성능을 좌우하는 한가지 요소는 RSC 부호를 연결하는 인터리버로서 인터리버 크기가 클수록 성능은 좋아지며, 짧은 프레임 경우 랜덤 인터리버를 사용하는 것보다는 최적의 구조적 인터리버를 사용하는 것이 유리하다. 하지만 긴 프레임의 경우, 이러한 구조적 인터리버의 경우 고정된 구조 때문에 일정크기 이상의 최소 해밍거리를 가지는 것이 불가능하므로 랜덤 인터리버를 사용하는 것이 유리하다. 터보 코드의 성능을 향상시키기 위해서는 기존의 구조적 인터리빙 방식이 아닌 랜덤 인터리버를 사용해야 하는데, 이러한 인터리버 방식의 문제점은 각각의 전송율에 따라 모든 인터리버 어드레스를 저장해야 한다는 점이다. 따라서 최근의 추세는 인터리버의 랜덤성을 유지하면서 어드레스를 저장하지 않고 계산에 의해 만들

어 내는 방법(on-the-fly)들이 연구되고 있다. 이때 최적의 성능을 보장하도록 인터리버 어드레스의 계산 알고리즘으로 사용 할 수 있는 방법은 3GPP에서 표준으로 논의되었던 LCS (Linear Congruential Sequence), GF(Galois Field), MIL 인터리버(Multi stage Interleaving), Mother 인터리버 등이 있다[9][10][11]. 각 알고리즘은 탐색 과정을 통해 어드레스 계산에 필요한 파라미터들에 대한 최적화 과정을 수행하고 이렇게 얻어진 파라미터들을 ROM에 저장하여 어드레스 계산에 이용하는 방법들을 사용한다. 인터리버 어드레스 계산 시 일반적으로 2차원 배열을 이용하는 방식의 성능이 우수한 것으로 알려져 있다. 이때 2차원 인터리버 방식에서 사용하는 기법은 행 인덱스(row index)에 대해서는 비트 리버설(Bit Reversal) 방법을 사용하고, 각각의 행에 대하여 적절한 고유의 규칙을 이용하여 열 치환(column Permutation)을 수행하는 것이다. 인터리버의 설계 시 고려할 사항은 인터리버 어드레스를 구성하기 위하여 가능한 적은 메모리를 가지고서 실현 가능해야 하며, 다양한 전송율에서도 쉽게 적용이 되어야 한다.

3.2 제안하는 터보매직 인터리버

터보코드에서 복호과정은 반복적인 복호과정을 거치게 되므로 인터리버와 디인터리버는 반복횟수만큼 여러번 반복된다. 따라서 터보코드의 내부 인터리버는 복잡성을 줄여야하고 또한 성능에도 큰 영향을 미치므로 중요하게 취급되어야 한다. 본 논문에서 제안하는 인터리버는 기본적으로 매직 매트릭스를 이용하고 2차원배열 방식을 사용하여, 기존의 IMT-2000에 제안된 터보코드용 인터리버보다 어드레스 생성방법이 다소 간단하고, 보다 나은 성능을 갖도록 알고리즘을 설계하였다. 매직 매트릭스는 중국에서 낙서(洛書)라고 불리워지는 신비의 매트릭스로서 보통 마방진이라고 일컫는다. 이 마방진이 유럽에 건너가서 Magic square matrix로 통용되었으며 오늘날에는 수학 및 과학에서 연구가 계속 진행중이다. Magic(N) 은 정수로 구성되는 *N-by-N* 매트릭스이다. 1부터 N^2 까지 값을 가지며 각 행과 열 및 대각선 방향으로 각각의 합이 다음 식 (22)과 같이 동일하다. 또한 정수 N = 1, 3, 4, 5,… 에 대하여 매트릭스 구성이 가능하다.

$$S=\frac{N}{2}(N^2+1),\quad \text{N=1,3,4,5}\cdots, \tag{22}$$

Magic square 매트릭스의 생성 알고리즘은 여러 가지가 있으며 그림 5는 Matlab 5.1 버전에서 제공되는 구조로서 크기가 4×4와 5×5의 예이다. 간단한 단위 인터리빙 과정은4×4에서는 매트릭스 주소에 따라서 쓰여진 데이터를 열 방향(col by col)으로 읽으며, 5×5에서는 행 방향(row by row)으로 읽으면 된다. 이는 연속된 숫자열이 나타남을 방지하고자 함이다.

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

(a) 4×4 Magic matrix

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

(b) 5×5 Magic matrix

그림 5. Magic square 메트릭스의 예.

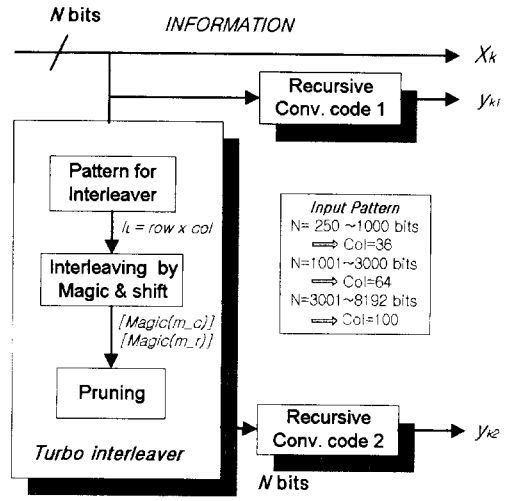

그림 6. 매직 인터리버가 적용된 터보 인코더 구조.

그림 6은 터보 코드의 인코딩 과정에서 매직 인터리버가 적용되는 과정을 도식화한 것이다. 그림에서 첫 번째 출력단에서는 입력된 정보비트가 원 정보로 그대로 출력되고, 두 번째 출력단은 컨벌루션 코드로 부호화된 데이터가 출력되며, 세 번째 출력단에서는 인터리빙이 적용된 데이터를 컨벌루션 코드로 부호화하여 데이터를 출력한다. 즉 여기서 인터리빙 과정에 제안하는 매직 매트릭스를 이용한

터보 매직 인터리빙 기법을 적용하는 것이다. 터보 코드에 적용되는 내부 인터리버는 다음과 같이 4단계로 구성된다. 첫 번째 단계에서는 입력 비트열을 열 단위로 쓰고, 두 번째 단계에서는 행 내부(intra-row) 단위로 데이터를 섞고 세 번째로 천이과정을 거친다. 마지막 네 번째 단계에서는 각 행 간(inter-row) 단위로 데이터를 섞고 열 방향으로 읽어서 인터리빙 어드레스를 출력시키면 된다.

1 단계 : 각 행 단위로 입력 데이터를 쓴다. 먼저 입력되는 프레임 길이를 N 이라고 하고 다음과 같은 규칙에 의하여 행과 열의 개수를 구한다. 크기가 N인 인터리버를 구성하기 위하여 $I_L = row \times col$ 인 패턴을 결정하고 최종적으로 N 보다 큰 값은 버린다. 행과 열의 수를 미리 결정하고 입력 비트열을 행 단위(row by row)로 쓴다. 다음의 규칙에 의해 열(column)의 개수를 먼저 결정한다.

Case 1

(N = 250 ~ 1000 bits) ⇒ $col = 36$

Case 2

(N = 1001 ~ 3000 bits) ⇒ $col = 64$

Case 3

(N = 3001 ~ 8192 bits) ⇒ $col = 100$

행(row)의 개수는 다음식에 의하여 구한다.

$$row = \lceil \frac{N}{col} \rceil \tag{23}$$

여기서 $\lceil A \rceil$ 는 A 보다 큰 최소 정수를 의미한다. 인터리버의 입력 비트열은 $I_L = row \times col$ 매트릭스에 행 단위로 입력된다.

2 단계 : 행 내부 단위로 치환 (Intra_row Permutation)을 한다. 즉, 각각의 행 내부의 데이터들을 섞는다. 각 행의 데이터를 크기가 $m_c \times m_c$ 인 매직 매트릭스를 이용하여 치환시킨다. 여기서 $m_c = \sqrt{col}$ 이며, i 번째 행의 각 열들에 대한 주소값 $A(i)$는 다음 식과 같이 구할 수 있다.

$$A(i) = \{[\ Magic(m_c)\ ] + col \cdot (i-1)\}_{\ll(i)} \quad \text{if } m_c = 4\text{의 배수} \tag{24}$$

$$A(i) = \{[\ Magic(m_c)\ ]^T + col \cdot (i-1)\}_{\ll(i)} \quad \text{others}$$

위의 식에서 먼저 m_c 값에 의한 매직 매트릭스를 이용하여 I_{col} 인터리버를 구성하는데, 여기서 m_c 가 4의 배수일 경우는 입력값을 $m_c \times m_c$ 매직 매트릭스로 채운다음 열 단위 방향으로 읽어오며, 그 외에는 입력된 데이터 매트릭스를 전치(Transpose)시켜서 열 단위 방향으로 읽어온다. 이것은 매직 매트릭스 특성상 나타나는 연속되는 주소를 제거시키기 위한 것이다.

3 단계 : 행 내부 단위로 왼쪽으로 천이 (Intra_row Left Shift)하는 과정이다. 각 행별로 읽어온 데이터를 왼쪽으로 천이(shift) 시킨다. 위의 식 (24)에서 $\ll(i)$은 왼쪽으로 i 만큼 천이(shift) 시킴을 의미한다. 즉 i 번째 행(row)에 대하여 i 값만큼 왼쪽으로 천이 시킨다. 이것은 행 별로 같은 인터리빙 패턴을 방지하기 위함이다.

4 단계 : 행간의 치환(Inter_row Permutation)이다. 전체 행에 대하여 행 값을 치환시키기 위하여 먼저 m_r 값을 구한다. 크기가 $m_r \times m_r$ 인 매직 매트릭스를 이용하여 I_{row} 인터리버를 구성하고 이를 이용하여 열 단위로 인터리빙된 데이터를 읽어온다. 여기서 $m_r = \sqrt{row}$ 이고, 만일 m_r 값이 정수가 아니면 행 보다 큰 최소 제곱수를 구하여 그의 근으로 정한다. 그리고 행 보다 큰 수는 삭제(pruning)시킨다. 행간 치환을 위한 어드레스 패턴은 다음 식과 같이 된다.

$$f(i) = [Magic(m_r)] \quad i = 1, 2, \cdots row \tag{25}$$

위 식은 Magic(m_r)의 매트릭스를 구하고 이를 일련의 어드레스 값으로 읽어들인 값이다. i 번째 행은 $f(i)$를 이용하여 얻어진 어드레스 값으로 바뀌게 되고, 따라서 행 간(inter-row)에 치환이 이루어지게 되다. 최종적으로 행 단위로 데이터를 읽어온다. 마지막으로 $I_L = row \times col$ 로 구성된 인터리빙 어드레스 값에 대하여 입력 프레임의 길이 N 보다 큰 값에 대한 삭제(pruning)과정을 거친다. 삭제 값 S는 다음과 같다.

$$S = N - (row \times col) \tag{26}$$

지금까지 인터리빙 과정을 프레임 크기가 250인 경우에 대하여 예로서 그림 7에 도식적으로 나타내었다. 그림 8은 인터리빙 과정의 흐름도이다.

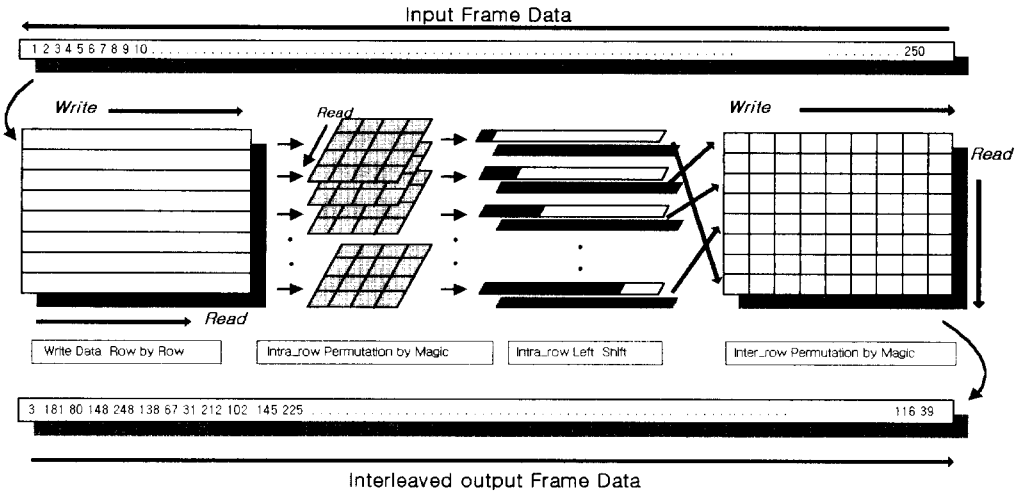

그림 7. 터보 매직(Turbo Magic) 인터리빙 과정

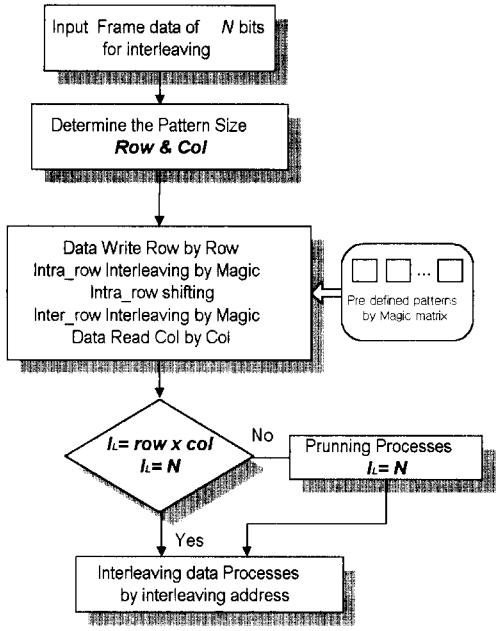

그림 8. 인터리빙 과정 흐름도

3.3 시뮬레이션 결과 및 분석

그림 9는 제안된 인터리버의 성능 분석을 위한 전체 시뮬레이션 블록도 이다. 성능 분석의 환경은 AWGN 및 레일리 페이딩 채널에서 이루어졌으며, 터보 코드의 구속장 (K) = 4이며 부호율(R)은 1/3로 고정하였고, 반복 복호 수는 4회이다. 레일리 페이딩 채널의 도플러 주파수는 50 Hz이고, 다중 경로 수는 6이다. 그림 10,11은 AWGN 채널 환경에서 인터리버의 종류에 따른 비트 에러율(BER) 성능을 나타낸다. 입력되는 프레임크기는 N=1024와 3200 비트이다. 그림 12,13는 레일리 페이딩 채널 환경에서 인터리버의 종류에 따른 비트 에러율(BER) 성능을 나타내며 입력프레임의 크기는 1024, 3200 비트이다. 각각의 그림에서 블록 인터리버의 성능이 가장 떨어지고, 제안된 매직 인터리버가 우수함을 알 수 있으며, 그 사이에서 랜덤 인터리버가 약간의 우열관계를 보이고 있다. HNS사의 GF 인터리버와 Mother 인터리버의 성능이 제안된 인터리버와 비슷하게 나타난다. 블록 인터리버와 순수한 랜덤 인터리버의 성능은 에러 마루현상이 생기나 제안 인터리버와 GF 및 Mother 인터리버는 신호잡음비(Eb/No)가 증가함에 따라 성능이 향상됨을 알 수 있다. 성능이 우수하게 나타나는 이유는 해밍무게 분포특성이 다른 인터리버에 비해 우수하고, 한 프레임내에서 상관 관계가 있는 입력정보를 효과적으로 상관 관계가 없는 정보로 변환해주기 때문이다. 표 2.는 인터리버 종류별 자유거리(Free distance)를 터보부호기에서 컴퓨터를 이용하여 구한 값이다. 터보 부호어의 무게(weight)는 입력 정보열의 무게와 각각의 구성부호기 출력에서 나온 부호화된 열의 무게 합으로 정의되며, 자유거리(free distance)는 0 부호어를 제외한 가능한 모든 부호어 무게에서 최소 해밍 무게(Hamming weight)가 된다. 본 논문에서는 무게(weight)는 2이고 구속장 4, 생성다항식 g=[1011,1101] 이다. 표에서 나타난 바와 같이 랜덤

인터리버가 다소 떨어지고 HNS사의 GF 인터리버와 Mother 인터리버의 자유거리 값이 제안된 인터리버와 비슷하게 나타난다. 본 논문에서 사용한 매직 매트릭스는 Matlab 5.1 버전에서 제공되는 구조를 사용하였으며, 프레임 크기가 1024인 경우는 Row 값이 16, Col 값이 64 이므로 4×4 와 8×8 크기의 매직 매트릭스를 사용하였다.

표 2. 인터리버에 따른 d_{free}

프레임 크기	GF 인터리버	Magic 인터리버	Random 인터리버	Mother 인터리버
445	18	18	13	22
1024	30	26	17	25
1158	22	30	14	21
1280	25	31	17	27
3200	31	31	14	26

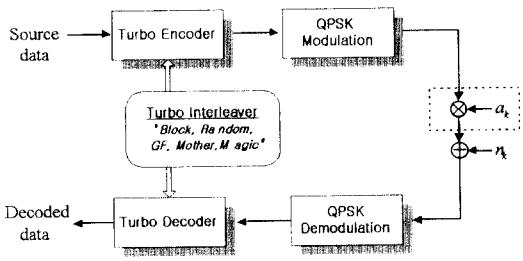

그림 9. 성능분석을 위한 블록도

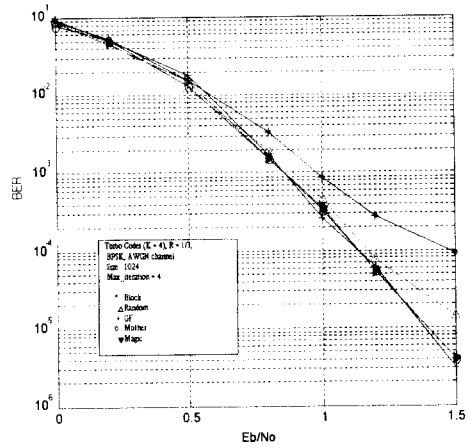

그림 10. 인터리버의 크기가 1024인 경우 AWGN 환경에서 인터리버 종류별 BER 성능비교

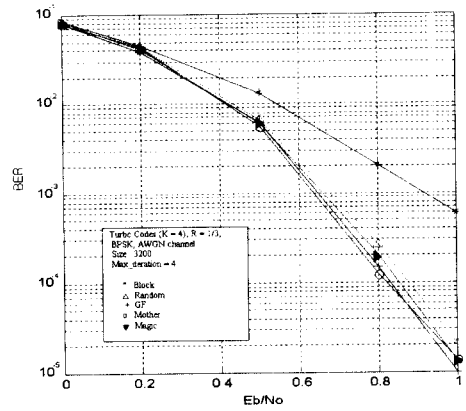

그림 11. 인터리버의 크기가 3200인 경우 AWGN 환경에서 인터리버 종류별 BER 성능비교

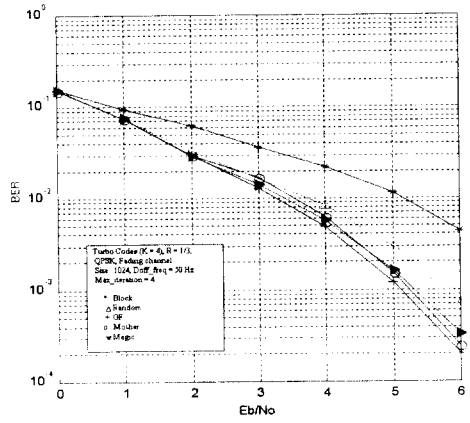

그림 12. 인터리버 크기가 1024인 경우 레일리 페이딩 환경에서 인터리버 종류별 BER 성능비교

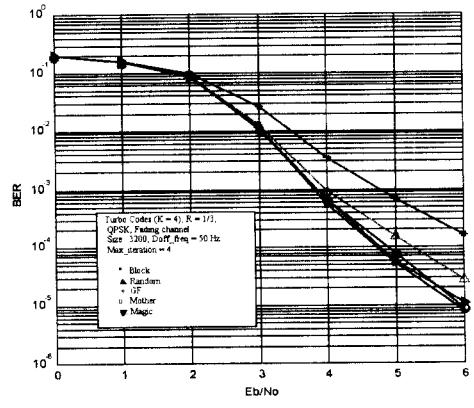

그림 13. 인터리버 크기가 3200인 경우 레일리 페이딩 환경에서 인터리버 종류별 BER 성능비교

Ⅳ. 결 론

터보 코드는 반복복호 횟수를 증가하면 성능이 향상되지만 반복 복호수가 증가할수록 복호를 위한 계산량과 지연이 증가하게 된다는 단점을 가지게 된다. 따라서 본 논문에서는 복호기 입력전의 수신 데이터를 이용하여 SNR를 추정하고, SNR에 따라 요구하는 성능을 만족하는 반복 복호수를 적응적으로 복호기 앞단에서 미리 설정한다. 최대 반복 복호를 수행했을 때와 비교하여 성능 감소 없이 평균 반복 복호수를 줄일 수 있었으며 기존에 제안된 정지기준 구조의 문제점인 각 프레임에서 반복 복호할 때마다 어떤 기준값을 계산해야 하는 복잡도와 지연을 프레임마다 한번만 계산함으로서 효과적으로 줄일 수 있다. 또한 터보코드에서 복호과정은 반복적인 복호과정을 거치게 되므로 인터리빙 과정은 반복횟수만큼 반복된다. 따라서 터보코드의 내부 인터리버는 복잡성을 줄여야하고 성능에도 큰 영향을 미치므로 중요하게 취급되어야 한다. 본 논문에서 제안하는 인터리버는 기본적으로 매직 매트릭스를 이용하고 2차원배열 방식을 사용하였으며, 기존의 IMT-2000에 제안된 터보코드용 인터리버와 비교하여 어드레스 생성방법이 간단하며 우수한 성능을 보인다.

참 고 문 헌

[1] J. Hagenauer, E.Offer and Lutz Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol.42. no.2, March 1996.

[2] Michael A Jordan and Robert A. Nichols,"The Effects of Channel Characteristics on Turbo Code Performance," *MILCOM'96*, Virginia, pp 17-21 Oct 21-24, 1996.

[3] M. C. Reed, and J. A. Asenstorfer, "A Novel Variance Estimator for Turbo-Code Decoding" *International Conference on Telecommuniation.* Australia, pp 173-178, 2-5 April, 1997.

[4] Todd A. Summers and Stephen Wilson, "SNR Mismatch and Online Estimation in Turbo Decoding," *IEEE Transactions on Comm.* Vol. 46, No. 4, pp 421-423, April 1998.

[5] Rose Y. Shao, Shu Lin, and Marc P.C Fossorier, "Two Simple Stopping Criteria For TurboDecoding," *IEEE Transaction on Comm.* Vol. 47, No. 8, pp 1117-1120, August 1999.

[6] S. Dolinar Divsalar, Weight Distributions for Turbo Codes Using Random and Nonrandom Permutations. Aug 1995, JPL TDA.

[7] Qualcomm, "A New Low-Complexity Turbo Code Interleaver", TR45.5.3.1 1999.

[8] J. Hagenauer, E.Offer Lutz Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol.42. no.2, March 1996.

[9] Akira, Hirohito, Fumiyuki, "Complexity Reduction of Turbo Decoding", *IEEE VTC'99*, pp1570~1574, 1999

[10] Chris Heegard, Stephen B.Wicker, *Turbo Coding*, Kluwer Publishers, pp35~63, 1999

[11] 윤 영우 외 3인 "터보 복호기 설계시의 고려사항", *JCCI'99*, pp 40-45, 1999.4

[12] 이문호, *실용정보이론 - 오류정정이론의 기초* - 복두, 1999

김 순 영(Soon-young Kim) 정회원
통신학회 논문지 제23권 1호 참조
현재 : 한국통신
<주관심 분야> IMT-2000, 채널코딩

장 진 수(Jin Su Chang) 준회원
통신학회 논문지 제24권 4호 참조
현재 : 전북대학교 컴퓨터공학과 박사과정
<주관심 분야> IMT-2000, 채널코딩

성 락 주(Nark-Joo Seong) 준회원

1998년 : 원광대학교 전자공학과 졸업
1999년 3월~현재 : 전북대학교 정보통신학과 석사과정
<주관심 분야> IMT-2000, 채널코딩

이 문 호(Moon Ho Lee) 정회원
통신학회 논문지 제24권 4호 참조
현재 : 전북대학교 교수, 정보통신연구소 소장
<주관심 분야> IMT-2000, 영상신호처리, 이동통신