IMT-2000에서 음성 전송을 위한 터보 코드 복호기 설계

강 태 환, 박 성 모
전남대학교 컴퓨터공학과
전화 : 062-530-0798 / 핸드폰 : 019-619-3617

Design of A Turbo-code Decoder for Speech Transmission in IMT-2000

Tae Hwan Kang, Seong Mo Park
Dept. of Computer Engineering, Chonnam National University
E-mail : thkang@cep03w.chonnam.ac.kr

Abstract

Recently, Turbo code has been considered for channel coding in IMT-2000(International Mobile Telecommunication-2000) system, because it offers better error correcting capability than the traditional convolution/viterbi coding.

In this paper, a turbo code decoder for speech transmission in IMT-2000 system with frame size 192 bits, constrait length K=3, generator polynomials G(5,7) and code rate R=1/3 is designed using SOVA(Soft Output Viterbi Algorithm) and block interleaver

I. 서론

통신에서 신뢰성 있는 정보를 주고받기 위해서는 채널 코딩은 필수적이다. 특히 IMT-2000에서는 무선 채널을 통하여 고품질의 음성, 인터넷, 영상 등 멀티미디어 통신이 가능한 서비스를 제공하기 때문에 강력한 채널코딩이 필요하다.

1993년 Berrou에 의해 처음 발표된 터보 코드(Turbo Code)는 Shannon limit에 거의 근접한 성능을 보여주었다[1]. 이 코드는 프레임 단위로 처리되며, 인터리버(interleaver) 크기가 클수록, 반복 복호를 많이 할수록 성능이 좋아진다. 그러나 프레임 크기가 클 경우 긴 지연이 발생하므로 IMT-2000에서 터보 코드는 데이터 전송 서비스를 위해 사용하며 음성에 대해서는 종래의 컨벌루션(Convolution)/비터비(Viterbi) 코드를 사용한다. 이러한 방식은 2개의 코드를 사용하기 때문에 비효율적이다. 최근 IMT-2000에서 고품질 음성 서비스를 전송하기 위하여 8Kbps, 32Kbps와 같은 저속 전송율을 사용하고 지연이 40ms 이내가 되어야 하며[2], BER(Bit Error Rate) < 10^{-4}을 요구되고있는 음성 전송처럼 작은 프레임을 갖는 터보 코드에 대해 많은 연구가 이루어지고 있으며[3], 특히 Peter Jung은 무선 채널에서 음성 전송을 위해서 프레임 크기가 200보다 작은 정보 비트들을 가지는 터보 코드를 발표하였다[4].

터보 코드는 복호기의 종류에 따라 MAP(Maximum A Posteriori)과 SOVA 복호기로 나누어진다[5]. MAP 복호기는 BER 성능이 SOVA보다 우수하지만 구현이 복잡하다. 따라서 본 논문에서는 BER이 10^{-4}에서 0.8dB 정도 낮지만 구현이 간단한 SOVA 복호기를 설계하였으며 설계된 복호기는 Synopsys 툴을 이용하여 합성하였으며, Synopsys VHDL 디버거를 이용하여 기능 검증을 하였다.

본 논문은 한국과학재단지정 전남대학교 고품질 전기전자부품 및 시스템 연구센터의 연구비 지원에 의해 연구되었음

Ⅱ. 터보코드 부호기 및 복호기

2.1 터보코드 부호기 구조

설계된 터보코드 부호기는 부호율(Code Rate)이 1/3 이며, 생성다항식이 G[5,7]인 2개의 RSC(Recursive Systematic Convolution) 부호기를 병렬로 연결한 구조이며 그림 1과 같다.

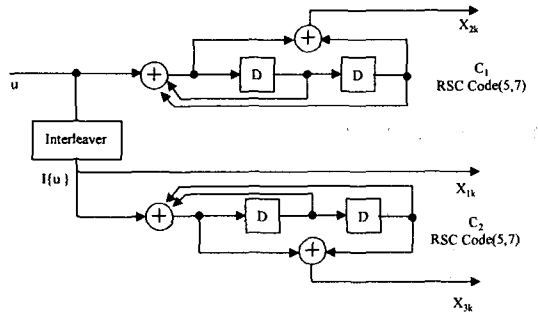

그림 1. 터보코드 부호기 구조

시간 k에서 부호기는 정보 비트 u를 첫 번째 구성 부호기에 입력한 후 PB(Parity Bit)인 X2k를 생성하며, 인터리빙(interleaving) 한 후 부호기를 거치지 않고 바로 X1k포트로 출력하며, 인터리빙 된 정보 비트 I{u}는 두 번째 구성 부호기를 통해서 PB인 X3k를 생성한다. 이러한 부호화 과정은 프레임 단위로 이루어지며 한 프레임의 길이는 인터리버의 길이와 같다. 인터리버는 연속되는 정보 비트의 순서를 바꾸어서 연집 오류(Burst Error)을 최소화한다. 인터리버의 종류로는 랜덤 인터리버, 블록 인터리버, 대각 인터리버 등이 있으며, 본 논문에서는 일반적으로 많이 사용하는 블록 인터리버를 사용하였다.

2.2 터보코드 복호기 구조

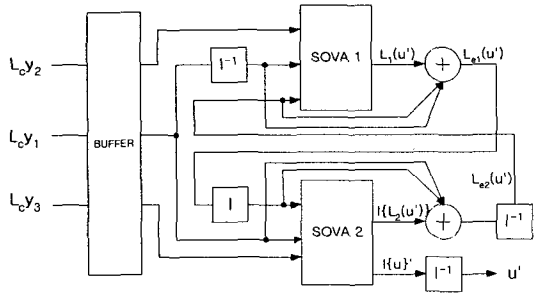

그림 2. 터보코드 복호기 구조

터보코드 복호기는 수신된 채널 비트들을 프레임 단위로 처리하며 N번의 반복 복호를 통해서 복호한다. 그림 2는 SOVA 알고리즘을 이용한 터보코드 복호기 구조를 보여주고 있으며, 2 개의 구성 복호기 SOVA 1, SOVA 2, 그리고 인터리버와 디인터리버를 이용해서 복호한다.

SOVA 1 복호기는 RSC 1 부호기로부터 생성된 PB인 X2와 SB(Systematic Bit)인 X1에 대한 수신된 채널 비트인 LcY2와 LcY1을 디인터리버(Deinterleaver)를 통과한 I-1{LcY1}과 SOVA 2 복호기의 외인성 정보(Extrinsic Information) Le2(u')를 받아들여 복호하며, 복호된 값은 LcY1을 디인터리빙 한 값과 SOVA 2의 외인성 정보 Le2(u') 값을 빼주어 SOVA 1의 외인성 정보 Le1(u')을 구하고 SOVA 2 복호기 입력으로 들어간다. 여기서 Le2(u')와 I-1{LcY1}을 빼주는 이유는 이 값들이 다음 SOVA 구성 복호기에 영향을 미치지 못하게 함으로서 SOVA 1 복호기에서 오류를 정정하지 못할 경우 SOVA 2 복호기에서 오류를 정정하기 위해서이다.

SOVA 2 복호기는 RSC 2 부호기로부터 생성된 PB인 X3와 SB인 X1에 대한 수신된 채널 비트인 LcY3, LcY1 그리고 SOVA 1 복호기의 외인성 정보 Le1(u')를 디인터리빙 한 후 받아들여 복호하며, 복호된 값은 SOVA 1 복호기와 마찬가지로 외인성 정보 Le2(u')를 구하여 SOVA 1 복호기 입력으로 보내며, N 번의 반복 복호가 끝나면 추정된(Estimated) 연속 비트인 I{u'}을 디인터리빙 한 후 복호한다.

Ⅲ SOVA 복호기 처리 과정

그림 3은 현재 시간 t와 t+1에 대해서 RSC 부호기의 상태천이를 하드웨어 설계를 위하여 변형한 상태천

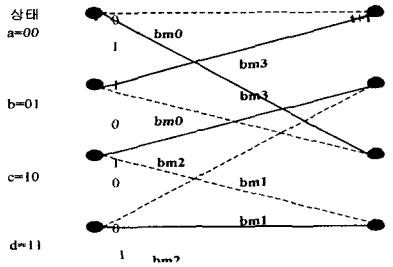

그림 3. 상태천이도

이도를 보여주고 있으며, 실선은 정보 비트 '1'에 해당하고 눈금 선은 정보 비트 '0'에 해당한다. 그림 1의 RSC 부호기의 메모리 크기가 2이므로 상태는 4 상태가 존재하며 각각의 경로 매트릭은 bm0, bm1, bm2,

bm3이다.

그림 3의 상태천이도부터 SOVA를 이용해서 복호되는 과정을 다음과 같이 설명할 수 있다.

단계 1)

시간 t=0에서 초기의 매트릭(Metric) 값을 모두 '0'으로 초기화한다.

단계 2)

시간 t를 하나 증가시키고 상태천이도 각각의 상태에 대해서 가지 매트릭(Branch Metric)을 계산한다.

단계 3)

각각의 상태에 대해서 전 시간의 경로 매트릭과 현재 시간의 가지 매트릭을 더하여, 노드에 들어오는 2개의 경로에 대해 크기를 비교하여 새로운 경로 매트릭과 생존경로에 대한 경판정 값과 델타(delta) 값을 구한다. 여기서 델타 값은 추정된 비트에 대한 신뢰도 또는 LLR(Log-Likely Rate)이다. 터보코드는 프레임 단위로 처리하기 때문에 한 프레임이 처리될 때까지 단계2와 단계3을 반복한다.

단계 4)

한 프레임이 끝나면 마지막 비트 위치에서 구한 모든 상태 매트릭의 크기를 비교하여 가장 큰 매트릭의 상태에서부터 한 프레임의 시작 위치까지 역추적(Traceback)한다. 역추적은 그림 3의 상태천이도로부터 현재상태와 경판정된 비트 값을 이용하여 전 상태를 찾는다. 시간 t-1에 대해서 역추적 할 때마다 상태, 비트 그리고 델타 값을 저장한다.

단계 5)

저장된 상태, 비트, 델타 값을 이용하여 한 프레임에 대해서 신뢰도 업데이트 과정을 앞에서부터 수행한다. 신뢰도 업데이트는 한 프레임 전체의 델타 값에 대해 업데이트를 처리해야 하지만, 이럴 경우 복호하는데 매우 긴 지연이 발생하므로, 본 논문에서는 신뢰도 업데이트 깊이를 15로 제안하였으며 그림 4는 첫 번째 델타 값에 대해서 업데이트되는 과정을 보여주고 있으며, 굵은 선은 역추적을 통해서 찾은 가장 유사한 경로(Most Likely Path)이다.

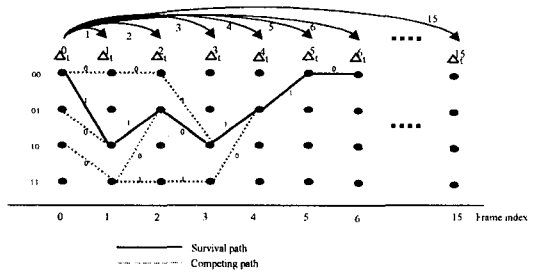

그림 4. 업데이트 처리 과정

업데이트 방법은 기준 델타와 업데이트 깊이에 위치한 델타의 크기를 비교 한 후 기준 델타가 클 경우 업데이트 깊이에 위치한 경로에서부터 경쟁 경로(Competing Path)를 기준 위치까지 찾은 후 기준에 해당하는 비트와 경쟁 경로의 마지막 비트를 비교하여 다르면 업데이트 깊이 위치에 해당하는 델타 값을 기준 델타에 저장하며, 같으면 업데이트를 하지 않고 깊이를 1 증가한 후 위의 과정을 반복한다. 여기서 기준은 업데이트할 위치의 델타이다. 이러한 과정은 한 프레임이 끝날 때까지 수행한다.

단계 6)

신뢰도 업데이트된 정보에서 SOVA 복호기의 입력으로 들어갔던 값들 중에서 채널로부터 수신된 비트 SB에 해당하는 값과 이전의 SOVA 구성 복호기의 외인성 정보를 빼주어 새로운 외인성 정보를 구한다.

단계 7)

반복(Iteration) 복호의 2배에 해당하는 값에 대해서 단계 2부터 단계 6을 반복 수행한다. 만약 반복 수행이 끝나면 단계 4에서 판정된 비트 값을 디인터리빙하여 출력한다.

Ⅳ 전체 회로 합성 및 기능 검정

4.1 SOVA를 이용한 전체 블록도

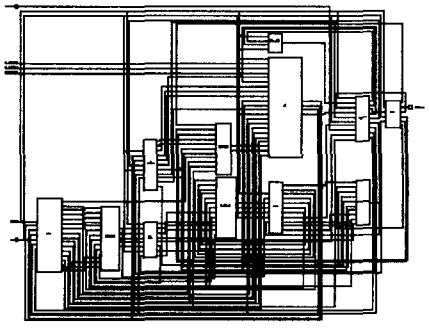

그림 5. SOVA를 이용한 복호기 전체 블록도

그림 5는 SOVA를 이용한 터보코드 복호기 전체 블록도를 보여주고 있으며 크게 TFU(Trace Forward Unit), TBU(Trace Backward Unit), RUU(Reliability Updating Unit), ECU(Extrinsic Calculation Unit), EBU(Estimate Bit Unit) 그리고 TOPCON(Top Control) 모듈로 나누어진다.

TFU에서는 모든 경로 매트릭에 대해 생존경로(Survival Pass) 매트릭을 구하며, TBU에서는 역추적을 수행하여 가장 근사한 경로를 찾으며, RUU에서는 신뢰도 업데이트를 수행하며, ECU에서는 외인성 정보를 계산하며, EBU에서는 TBU에서 복호된 비트를 디인터리빙하여 출력하며, TOPCON에서는 전체 복호기를 제어한다.

4.2 전체 회로 합성 및 기능 검증

설계된 복호기는 VHDL 언어를 이용하여 모델링하였으며 IDEC-C631 라이브러리와 Synopsys 툴의 Design_analyzer을 이용하여 회로를 합성하였고, VHDL 디버거를 이용하여 검증하였다. 표 1은 전체 SOVA 복호기의 게이트 수와 도착 시간을 보여주고 있으며, 게이트 수는 RUU 모듈이 가장 많았고 도착시간은 TFU 모듈에서 가장 긴 지연이 발생하였다.

표 1. 합성된 복호기의 게이트 수와 도착시간

블록	게이트(Gate) 수	도착시간
TFU	4840	25.14f
TBU	672	2.22f
RUU	9376	2.57f
RSLCU	997	2.94f
OBU	448	1.84f
TOPCON	236	1.38f
합계	16,569	

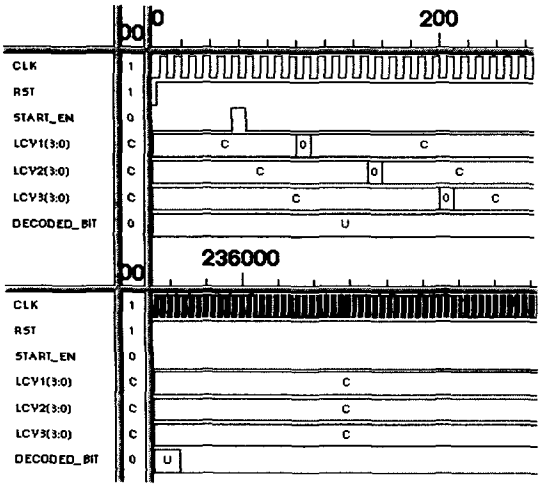

그림 6. 복호기 시뮬레이션 결과

연속적인 정보 비트 '0'을 부호화 한 후 AWGN 채널을 통해서 수신되었다고 가정할 때 몇 군데 오류가 발생한 비트들이 원래의 정보 비트 '0'으로 복호되는 시뮬레이션 결과의 일부분을 그림 6에 보였다.

V 결론

IMT-2000에서 음성 전송을 위한 채널코딩은 프레임 크기가 작고 완벽한 오류 정정이 필요하지 않는 한 컨벌루션/비터비 코드를 사용한다. 그러나 본 논문에서는 반복 복호를 2회로 고정된 터보 코드 복호기를 SOVA 알고리즘을 이용하여 상위수준에서 VHDL 언어와 Synopsys 툴을 이용하여 합성 및 기능 검증을 하였으며, 192 비트를 입력받아서 2번 반복 복호하는데 걸리는 총 클럭(Clock) 수는 46,872이며, 비트 당 224 클럭이 소요되었다.

참고문헌

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near shannon limit error correcting coding and decoding: turbo codes," Proceedings of ICC '93, pp. 1064-1070, Geneva, Switzerland, May 1993.

[2] FPLMTS/IMT-2000, Report of the Tenth Meeting of ITU-R Task Group 8/1, Mainz, April 1996.

[3] Min Wang, "The Performance of Turbo-codes is Asynchronous DS_CDMA," IEEE Global Telecommunications Conference, Vol. 3, pp 1548 -1551, 1997

[4] Peter Jung, "Comparision of turbo-code decoders applied to short frame transmission systems," IEEE Journal of Selected Areas in Communications, vol. 14, no.3, pp. 530-537, April 1996.

[5] Fossorier, M, Burkert, F, Shu Lin, Hagenauer, J. "On the equivalence between SOVA and max-log-MAP decodings," IEEE Communications Letters Vol. 2.5, pp. 137 -139 , May 1998.