논문 2005-42TC-2-8

실시간 COFDM시스템을 위한 효율적인 구조를 갖는 비터비 디코더 설계

(The viterbi decoder implementation with efficient structure for real-time Coded Orthogonal Frequency Division Multiplexing)

황 종 희*, 이 승 열*, 김 동 순**, 정 덕 진***

(Jong-Hee Hwang, Seung-Yerl Lee, Dong-Sun Kim, and Duck-Jin Chung)

요 약

디지털 멀티미디어 방송(DMB)은 대용량의 멀티미디어 정보를 무선환경의 이동체에 전송하기 위해 제안된 방식이다. 이러한 멀티미디어 서비스를 제공하기 위해 DMB시스템은 COFDM 변조방식을 사용하여 다중 경로 페이딩 현상을 극복하고, 동시에 강력한 채널오류 정정 능력을 필요로 한다. DMB 수신기를 위한 비터비 디코더(구속장 7, code rate 1/4)는 가변 부호화된 데이터의 복호화를 수행해야 하고, 방송시스템이므로 실시간으로 동작하기 위해서 효율적인 구조를 가져야 한다. 따라서 DMB 시스템을 위한 비터비 디코더를 구현하기 위해서는 복호화 과정을 고속으로 수행할 수 있는 별도의 전용 하드웨어 모듈을 설계하는 것이 바람직하다. 본 논문에서는 많은 연산량을 효율적으로 줄일 수 있는 결합된 Add-Compare-Select(ACS)와 Path Metric Normalization(PMN)구조를 새롭게 제안하고자 한다. PMN구조에서의 단점인 comparison tree에 의한 임계 경로(critical path)의 문제를 고정치(fixed value)에 의한 선택 알고리즘을 적용함으로써 고속 동작이 가능하게 하였고, ACS구조에서는 분할 기법(decomposition method)과 선계산(pre-computation)을 이용하여 덧셈기, 비교기, 표준화기의 복잡도를 줄일 수 있도록 하였다. 시뮬레이션 결과 펑처드 비터비 디코더는 일반적인 구조를 적용했을 때 보다 면적 3.78%, 전력소모 12.22%, 최대 게이트 지연 23.80%의 감소율을 보였다.

Abstract

Digital Multimedia Broadcasting(DMB) is a reliable multi-service system for reception by mobile and portable receivers. DMB system allows interference-free reception under the conditions of multipath propagation and transmission errors using COFDM modulation scheme, simultaneously, needs powerful channel error's correction ability. Viterbi Decoder for DMB receiver uses punctured convolutional code and needs lots of computations for real-time operation. So, it is desired to design a high speed and low-power hardware scheme for Viterbi decoder. This paper proposes a combined add-compare-select(ACS) and path metric normalization(PMN) unit for computation power. The proposed PMN architecture reduces the problem of the critical path by applying fixed value for selection algorithm due to the comparison tree which has a weak point from structure with the high-speed operation. The proposed ACS uses the decomposition and the pre-computation technique for reducing the complicated degree of the adder, the comparator and multiplexer. According to a simulation result, reduction of area 3.78%, power consumption 12.22%, maximum gate delay 23.80% occurred from punctured viterbi decoder for DMB system.

Keywords: Digital Multimedia Broadcasting(DMB),Viterbi Decoder(VD),Branch Metric(BM), Path Metric Normalization(PMN), Add-Compare-Select(ACS), Trace-Back(TB).

* 정회원, *** 종신회원, 인하대학교 정보통신공학과
(Dept. of Information and Communication Engineering, Inha University)
** 정회원, 전자부품연구원 선임연구원
(Korea Electronics Technology Institute)
※ 본 연구는 2004년도 프런티어 유비쿼터스 컴퓨팅 사업의 초소형 스마트 센싱 엔진 기술 개발 과제의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.
접수일자: 2004년10월22일, 수정완료일: 2005년1월14일

I. 서 론

현재 통신 시스템의 사용증가는 더 빠른 정보 전송과 그에 따른 데이터의 신뢰성, 그리고 효율적인 채널의 이용을 필요로 한다. 통신시스템의 사용증가에 따른 통신자원(resource)들의 부족, 전송 속도의 향상으로 인한 noise들 문제가 심각하게 제시되고, 이를 극복하기 위한 여러 가지 노력과 연구가 필요하다. 그런 연구의 일환으로써 1948년 Claude E. Shannon은 적절하게 정보를 부호화(encoding)하고 다시 복호화(decoding)함으로써 noisy 채널에 의해 생긴 에러를 제한된 정보 전송률의 조건에서 원하는 수준으로 줄일 수 있다는 주장을 했다. Shannon의 작업이 출현한 이후 많은 노력으로 부호화와 복호화 기술들을 발전시켰고, 뿐만 아니라 코드 조합 기술과 변조 기술은 bandwidth-limited 채널을 사용하는 통신의 성능을 향상시켰다. 이와 같이 디지털 통신에서 데이터 전송시에는 오류가 발생하는데 이를 수정하기 위해서 오류 정정 부호화(error correcting coding ; channel coding)가 필요하게 된다. 이 오류 정정 부호화에는 크게 블록 부호(block code)와 컨볼루션 부호(convolutional code)로 나뉘는데, 그림 1과 같이 블록 부호화는 데이터를 블록 단위로 부호화와 복호화를 수행하고, 반면에 컨볼루션 부호화는 그림 2와 같이 일정 길이의 메모리를 이용해 이전 데이터와 현재 데이터를 비교해 부호화를 수행한다. 이러한 컨볼루션 부호화의 수신측에서 사용되는 대표적인 방법이 비터비 알고리즘이다.

컨볼루션 인코더는 오류정정효율이 우수하여 채널부호에 많이 이용되지만 연접오류에는 취약하여 인터리버와 함께 사용된다. 컨볼루션 인코더를 표현하는데 세 가지 중요한 인자가 있다. 첫 번째로 구속장(K)은 인코더의 출력에 영향을 미치는 입력비트의 수를 나타내고, 두 번째로 Code Rate(R)는 비터비 인코더의 k비트 입력과, n비트의 출력이 나온다면, Code Rate는 n/k가 된다. 일반적인 값은 1/2, 1/3, 2/3이 사용되어지고, 에러정정능력과 하드웨어 복잡도는 k/n이 감소함에 따라 증가한다. 마지막으로 생성다항식은 인코더의 연결 상태를 구분지어 주는 인자로 사용되어 진다. 그림 2는 구속장 K=3, 생성다항식 g1=(1011), g2=(1111), Code Rate R=1/2의 컨볼루션 인코더를 나타내며, shifter register와 modulo-2 가산기의 연결 상태를 보여주고 있다. 최대의 최소자유거리를 갖는 컨볼루션 부호의 생성다항식은 이미 알려져 있다[1].

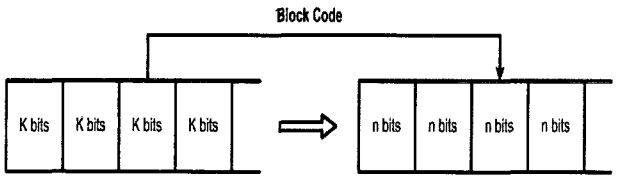


그림 1. 블록 부호
Fig. 1. Block code.

비터비 알고리즘은 구속장(K)의 크기에 따라 복잡도가 지수승으로 증가하게 된다[1]. 표 1의 ACS 단위블록에 대한 연산량 비교표에서 보는 거와 같이 K=3(R=1/2)인 경우와 K=7(R=1/2)인 경우의 복잡도를 비교해 보면 구속장에 따라 동일 ACS블록연산의 구조가 K=3에서는 2번, K=7에서는 32번이 반복되고, 가산기와 비교기의 입력비트수가 증가하게 되어 복잡도의 극명한 차이를 알 수 있다. 대략 $2^7/2^3$정도(16배)의 복잡도가 증가하게 된다. 구속장이 높은 비터비 디코더를

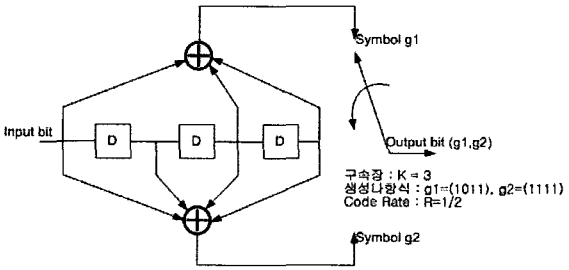


그림 2. 컨볼루션 인코더
Fig. 2. Convolution encoder.

표 1. K=3, K=7인 경우의 ACS 연산 비교
Table 1. ACS operation comparison in the case of K=3 and K=7.

K=3(R=1/2)인 경우의 ACS 블록 연산량		
연산	형태	butterfly 구조의 수
$PM_{i1}(a)=PM_{i-1}(a)+BM_i(a,0)$	4 to 2 bit addition	2
$PM_{i2}(a)=PM_{i-1}(c)+BM_i(c,0)$	4 to 2 bit addition	
$PM_{i1}(b)=PM_{i-1}(a)+BM_i(a,1)$	4 to 2 bit addition	
$PM_{i2}(b)=PM_{i-1}(c)+BM_i(c,1)$	4 to 2 bit addition	
$PM_i(a)=\min(PM_{i1}(a),PM_{i2}(a))$	4bit comparison and select	
$PM_i(b)=\min(PM_{i1}(b),PM_{i2}(b))$	4bit comparison and select	

K=7(R=1/2)인 경우의 ACS 블록 연산량		
연산	형태	butterfly 구조의 수
$PM_{i1}(a)=PM_{i-1}(a)+BM_i(a,0)$	9 to 5 bit addition	32
$PM_{i2}(a)=PM_{i-1}(c)+BM_i(c,0)$	9 to 5 bit addition	
$PM_{i1}(b)=PM_{i-1}(a)+BM_i(a,1)$	9 to 5 bit addition	
$PM_{i2}(b)=PM_{i-1}(c)+BM_i(c,1)$	9 to 5 bit addition	
$PM_i(a)=\min(PM_{i1}(a),PM_{i2}(a))$	9bit comparison and select	
$PM_i(b)=\min(PM_{i1}(b),PM_{i2}(b))$	9bit comparison and select	

구현시에 ACS블록에 대해 적절히 변화된 구조가 적용된다면 칩으로 구현시 면적, 전력, 데이터 처리량 측면에서 많은 이득을 얻을 수 있을 것이다.

DMB시스템은 무선 이동체 수신환경에서 라디오 수신방송의 음질을 떨어뜨리는 중요한 요인 중의 하나인 다중경로 페이딩 현상에 대처하기 위해서 OFDM(Orthogonal Frequency Division Multiplexing) 변조방식을 사용한다. 일반적으로 채널오류를 정정하기 위한 채널코드와 OFDM 신호를 합쳐서 COFDM(Coded OFDM)이라 부르는데, DMB 시스템은 COFDM을 사용하여 다중경로 페이딩 현상을 극복했으며 동시에 강력한 채널오류 정정능력을 제공한다. 이것은 효율적인 전력사용과 더불어 SFN (Single Frequency Network)구성을 가능하게하며 다양한 정보 전송을 가능케 하는 충분한 데이터 용량을 제공한다. DMB시스템에서는 전송되는 신호의 민감도에 따라서 에러 정정 코드의 protection 단계를 조절함으로써 전송채널의 사용을 극대화시키기 위해 RCPC (Rate-Compatible Punctured Convolutional) 코드를 채널코드로 사용하고, eureka-147에 의해 8/9~1/4의 코드율을 지원한다[2]. DMB 수신기를 위한 비터비 디코더는 실시간으로 동작하기 위해서 채널 디코더의 최대 출력 데이터율을 최대 1.824Mbps까지 지원 가능해야하며, 이것은 최대 64개의 채널을 통해서 오디오와 데이터 서비스를 모두 포함할 경우의 데이터율이다. 따라서 실시간 멀티미디어 방송의 비코더 디코더 구현을 위해서는 복호화 과정을 고속으로 수행할 수 있는 별도의 전용 하드웨어 모듈을 설계하는 것이 바람직하다. 이에 본 논문에서는 COFDM 시스템을 위한 효율적인 구조를 갖는 비터비 디코더를 검증하기 위해 DMB시스템에 맞는 고속으로 동작하면서도 저전력이 될 수 있는 전용 비터비 디코더를 설계하고자 한다. 비터비 디코더 구성 요소 중 ACS (add-compare-select)에서 분할 기법(decomposition method) 및 선계산(pre-processing)을 통한 전력 소모 감소 기법과 PMN(path metric normalization)에서는 기존의 comparison tree를 이용한 구조를 수행하는 대신 고정치(fixed value)에 의한 최소값을 결정하는 선택 알고리즘을 적용하여 maximum gate delay를 줄일 수 있는 구조를 제안하였다.

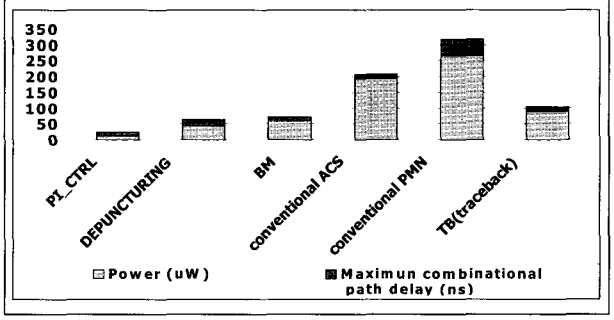
II. 본 론

일반적인 비터비 디코더의 구조일 경우 Synopsys사의 Design compiler를 이용한 전력, 크기 및 최대 지연시간은 구현상에 있어 아래 표2와 같은 특성이 있다. 표에서 확인할 수 있듯이 비터비 디코더의 구현시 RAM(Random Access Memory)의 사용이 필수적인 TB unit이 가장 많은 면적을 차지하게 되고 , ACS는 매 시간 간격마다 반복되어 덧셈, 비교, 선택을 수행해야하므로 많은 전력소모가 발생하게 된다. 또한 PMN 구조는 depth 8의 comparison tree를 사용하므로 전력소모가 크고, 많은 gate delay를 필요로 한다.

표 2. 일반적인 구조를 사용한 디펑처드 비터비 디코더의 성능분석

Table 2. Depunctured viterbi decoder's performance analysis for using the conventional architecture.

	Area	Power (uW)	Maximum combinational path delay (ns)
pi_ctrl	501.37651	13.449	10.483
depuncturing	154157.2485	43.857	19.297
bm	4629.34682	60.493	9.321
conventional acs	96023.67969	194.007	12.341
conventional pmn	79705.25781	266.799	50.656
tb(traceback)	624961.8183	87.129	17.305



따라서, 효과적인 비터비 디코더의 구현을 위해서는 ACS와 PMN 부분의 저전력 설계 방법과 PMN unit에서 gate delay를 최소화 하는 구조가 필수적이며, 본 논문에서는 이러한 부분을 줄이기 위한 새로운 구조들을 제안하여 최적화된 디펑처드 비터디 디코더를 구현하고자 한다.

1. DMB 시스템을 위한 디펑처드 비터비 디코더

비터비 알고리즘은 1967년에 컨볼루션 부호(convolutional code)를 복호할 목적으로 제안되었고, 다양한 통신시스템에서 오류정정의 방법으로 활용되었다. 비터비 알고리즘은 잡음이 섞인 수신된 데이터에 대하여 한정된 상태에서 프로세스의 연속 값에 대한 최적의 상태추적을 할 수 있다.

가. 펑처드 컨볼루션 인코더

고속의 데이터를 전송하기 위해서는 생성된 부호어의 규칙에 따라 특정 비트를 제거해 전송하는 펑처링 기법이 사용되어지는데, 펑처링 코드를 사용하는 또 다른 이유 중의 하나는 n/m, n≠1인 코드의 복호화를 쉽게 하기 위해서이다. 즉 n/m코드의 복호기가 복잡하므로 1/m인 mother 코드를 이용해 n/m을 만들고 디펑처링을 통하여 복호화를 하게 되면 비교적 간단한 1/m코드 복호기로 n/m코드를 복호화할 수 있기 때문이다. 펑처링 과정은 컨볼루션 인코더에서 생성된 코드워드의 일부를 일정한 규칙에 의해 버리고 전송하지 않는 방법으로 구현될 수 있다. 펑처링을 사용할 경우, 전송 데이터 자체에 오류가 내재되어 있기 때문에, 수신기의 성능을 저하시키는 특징이 있다[3,4]. DMB 시스템에서는 펑처링된 데이터를 Code Rate 1/4, Constraint length 7의 컨볼루션 부호화를 사용하여 송신측에서 데이터를 부호화시킨다.

나. 비터비 알고리즘 방법

비터비 알고리즘은 maximum-likelihood 복호화 과정으로서 제안되었다. 디코더는 송신단에서 전송된 sequence를 code 구조(code trellis), 수신된 sequence y, 채널 특성 등을 이용하여 추정하게 된다. 채널 입력이 x_m이면, 채널 출력y를 수신할 확률 및 이의 복호 metric은 아래와 같이 주어진다.

$$\Pr[\ y \mid x_m\ ] = \prod_{n=0}^{\infty} \Pr(\ y_n \mid x_{mn}\ )$$

$$\ln \Pr[\ y \mid x_m\ ] = \sum_{n=0}^{\infty} \ln[\Pr(\ y_n \mid x_{mn}\ )]$$

위 식을 이용하여, maximum likelihood 복호기는 아래의 추정된 값 x를 최대화시키는 x_m을 구하는 것으로 표현 할 수 있다[5].

$$\hat{x} = \max\ \ln \Pr[\ y \mid \hat{x}_m\ ]$$

$$where \quad \ln \Pr[\ y \mid x_m\ ] = \sum_{n=0}^{\infty} \ln[\Pr(\ y_n \mid x_{mn}\ )]$$

이 과정을 아래의 3단계로 구분하여 나타낼 수 있다.

1)수신된 데이터와 branch word사이의 hamming distance를 계산하여 trellis diagram의 weight를 결정한다.

2) 시간 t에서의 가장 짧은 경로를 이용해 재귀적인 방법으로 시간 t+1에서의 가장 짧은 경로를 계산한다. 어떤 한 상태로 들어오는 branch metric과 이전 상태에서의 path metric값의 합을 비교하여 적은 에러 metric 값을 짧은 경로로 선택하게 된다. 이 단계에서 결정된

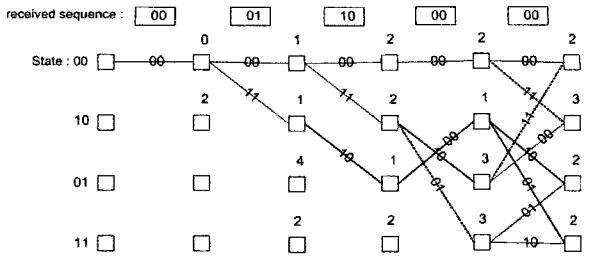


그림 3. 인코더의 4가지 상태에 대한 trellis diagram
Fig. 3. The trellis diagram for 4 states of the encoder.

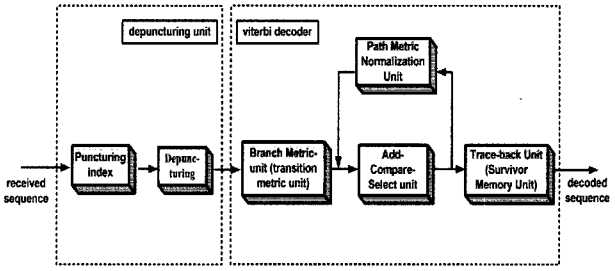


그림 4. DMB 시스템을 위한 디펑처드 비터비 디코더
Fig. 4. Depunctured viterbi decoder for DMB system.

값으로 신호의 생존 경로(survival path)값을 재귀적으로 바꿔준다. 이 방법을 ACS(add-compare-select) 반복법이라고 알려져 있다.

3) 단계2)에서 결정되는 방법에 의해 각각의 격자도 상태에 도달하는 가장 짧은 경로를 재귀적으로 찾아 나간다. 여기서 가장 짧은 경로는 생존 경로라고 부르고, 그 상태와 프로세스는 생존 경로 디코드(survival path decode)처럼 이용된다. 마지막으로 모든 생존 경로들을 매 시간 마다 역추적(trace back)하면, 그들로 유일한 경로가 만들어지며 바로 그 경로가 우리가 찾으려는 신호의 경로와 가장 유사한 것이 된다.

2. 비터비 복호기의 설계

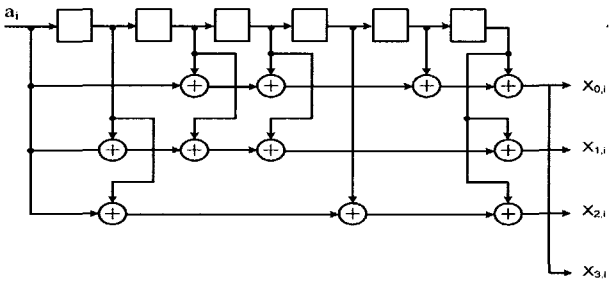
펑처링을 사용하는 비터비 복호기 설계과정에서 송신부는 펑처드 컨볼루션 인코더로 구현이 되고, 수신부의 하드웨어 구현은 그림 4와 같이 크게 디펑처링과 비터비 디코더 2개로 나눌 수 있으며, 세부 블록은 puncturing index, depuncturing, branch metric unit, add-compare-select unit, path metric normalization unit, trace-back unit의 기본적인 6가지 블록으로 구성된다.

가. 펑처링 및 컨볼루션 인코더의 구현

DMB에서는 CD수준의 오디오를 효율적으로 전송하기 위하여 PCM신호를 MPEG Audio Layer 2 방식을

이용하여 압축/전송하게 되는데, 비트 스트림은 실제 오디오 샘플이 압축되어 전송되는 body부분과 여러 가지 제어정보가 전송되는 header부분으로 나뉘게 된다. 하지만, header부분의 정보가 손상되면 복호화 자체가 불가능하게 되어 복호화된 신호의 음질을 크게 왜곡시키게 된다. 따라서, 이러한 경우에 신호의 민감도에 따라서 에러 정정 코드의 protection단계를 달리 적용할 필요가 있다. Eureka - 147에서는 채널의 잡음특성에 따라 24개의 코드율을 이용하여 부호율을 바꿀 수 있도록 하였다. 에러정정율을 유지하면서 부호율의 증가가 가능하므로 전송채널을 효과적으로 사용할 수 있게 한다. DMB에서는 그림 5에서 생성된 코드율 1/4인 컨볼루션 코드를 mother코드로 사용하고, 1 logical 프레임에 해당하는 I 비트 데이터 $(a_i)_{i=0}^{I-1}$는 컨볼루션 인코더의 입력으로 들어가 아래와 같이 4I+24 비트 코드워드 $\{(x_{0,i}, x_{1,i}, x_{2,i}, x_{3,i})\}_{i=0}^{I+5}$가 생성된다.

생성된 코드워드 중에서 최초 4I 비트는 L개의 128 비트 블록으로 나누어지고, 각각의 블록은 다시 4개의 32 비트 서브블록으로 나누어진다. 펑처링 과정은 하나의 128 비트 블록에 속하는 4개의 32 비트 서브블록들에 대해 같은 펑처리 벡터로 펑처링을 수행한다. 표 3에 펑처링 벡터테이블을 나타내었고, Code Rate는 {8/(8+PI)}과 동일하다. 이때, 0 상태 merging을 위해 추가된 6개의 0에 의한 24 비트 출력은 펑처링 벡터 V_T = (1100 1100 1100 1100 1100 1100)에 의해 펑처링되고 이때 생성된 12비트를 tail 비트로 부른다. 각 4개의 서브블록에 해당하는 128비트에서 생성되는 비트 수는 4×(8+PI)가 되고, 블록과 서브블록의 순서가 유지되어야 한다. DMB시스템에서는 한 프레임에 2개 이상의 펑처링 벡터를 사용하며, 사용한 벡터의 단계(PI)와 펑처링 벡터의 적용범위(L) 등을 protection profile을 통해 전송한다. 표 4는 제어 신호용 채널인 FIC의 protection profile로서, 전송모드에 따라 두 가지로 나뉘게 된다. 전송모드 I, II, IV에서는 energy dispersal scrambler의 출력 중 각 768비트가 컨볼루션 인코더의 입력으로 사용된다. 연속적인 mother코드 중 4I(3072)비트는 24개의 128비트의 서브블록으로 나눌 수 있다. 여기서 처음 21개의 서브블록은 PI=16에 의해서 펑처링되고 나머지 3개의 서브블록은 PI=15에 의해서 펑처리된다. padding 비트는 사용되지 않았고, Code Rate는 1/3을 유지한다. 마찬가지로, 전송모드 III에서는 1024비트가 사용되며, 컨볼루션 코드의 출력은 4I(4096)가 되고, 32개의 128비트의 서브블록으로 나뉘어 진다[2].



$$x_{0,i} = a_i \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-5} \oplus a_{i-6}$$
$$x_{1,i} = a_i \oplus a_{i-1} \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-6}$$
$$x_{2,i} = a_i \oplus a_{i-1} \oplus a_{i-4} \oplus a_{i-6}$$
$$x_{3,i} = a_i \oplus a_{i-2} \oplus a_{i-3} \oplus a_{i-5} \oplus a_{i-6}$$

그림 5. DMB 컨볼루션 인코더 (R=1/4, K=7)[2]
Fig. 5. DMB's convolution encoder (R=1/4, K=7)[2].

처음 29개의 서브블록은 PI=16에 의해서 펑처링되고 나머지 3개의 서브블록은 PI=15에 의해서 펑처링된다. 표 5는 주 서비스 채널인 MSC의 protection profile의

표 3. 펑처링 벡터 테이블
Table 3. Puncturing vector table.

PI(puncturing index)	($V_{PI,0}$, ... $V_{PI,31}$)
PI=1 (code rate:8/ 9)	1100 1000 1000 1000 1000 1000 1000 1000
PI=2 (code rate:8/10)	1100 1000 1000 1000 1100 1000 1000 1000
PI=3 (code rate:8/11)	1100 1000 1100 1000 1100 1000 1000 1000
PI=4 (code rate:8/12)	1100 1000 1100 1000 1100 1000 1100 1000
PI=5 (code rate:8/13)	1100 1100 1100 1000 1100 1000 1100 1000
PI=6 (code rate:8/14)	1100 1100 1100 1000 1100 1100 1100 1000
PI=7 (code rate:8/15)	1100 1100 1100 1100 1100 1100 1100 1000
PI=8 (code rate:8/16)	1100 1100 1100 1100 1100 1100 1100 1100
PI=9 (code rate:8/17)	1110 1100 1100 1100 1100 1100 1100 1100
PI=10 (code rate:8/18)	1110 1100 1100 1100 1110 1100 1100 1100
PI=11 (code rate:8/19)	1110 1100 1110 1100 1110 1100 1100 1100
PI=12 (code rate:8/20)	1110 1100 1110 1100 1110 1100 1110 1100
PI=13 (code rate:8/21)	1110 1110 1110 1100 1110 1100 1110 1100
PI=14 (code rate:8/22)	1110 1110 1110 1100 1110 1110 1110 1100
PI=15 (code rate:8/23)	1110 1110 1110 1110 1110 1110 1110 1100
PI=16 (code rate:8/24)	1110 1110 1110 1110 1110 1110 1110 1110
PI=17 (code rate:8/25)	1111 1110 1110 1110 1110 1110 1110 1110
PI=18 (code rate:8/26)	1111 1110 1110 1110 1111 1110 1110 1110
PI=19 (code rate:8/27)	1111 1110 1111 1110 1111 1110 1110 1110
PI=20 (code rate:8/28)	1111 1110 1111 1110 1111 1110 1111 1110
PI=21 (code rate:8/29)	1111 1111 1111 1110 1111 1110 1111 1110
PI=22 (code rate:8/30)	1111 1111 1111 1110 1111 1111 1111 1110
PI=23 (code rate:8/31)	1111 1111 1111 1111 1111 1111 1111 1110
PI=24 (code rate:8/32)	1111 1111 1111 1111 1111 1111 1111 1111

표 4. FIC Protection profile
Table 4. FIC Protection profile.

P (protection level)	전송 모드	L1	L2	PI1	PI2
-	I, II, IV	21	3	16	15
-	III	29	3	16	15

표 5. MSC Protection Profile (UEP : Unequal Error Protection)

Table 5. MSC Protection Profile (UEP : Unequal Error Protection).

Bit Rate (kbit/s)	I	P	L	L1 L2 L3 L4	PI1 PI2 PI3 PI4	number of padding bits
32	768	5	24	3 4 17 0	5 3 2 -	0
32	768	4	24	3 3 18 0	11 6 5 -	0
32	768	3	24	3 4 14 3	15 9 6 8	0
32	768	2	24	3 4 14 3	22 13 8 13	0
32	768	1	24	3 5 13 3	24 17 12 17	4
48	1152	5	36	4 3 26 3	5 4 2 3	0
48	1152	4	36	3 4 26 3	9 6 4 6	0
48	1152	3	36	3 4 26 3	15 10 6 9	4
48	1152	2	36	3 4 26 3	24 14 8 15	0
48	1152	1	36	3 5 25 3	24 18 13 18	0
:	:	:	:	:	:	:
320	7680	5	240	11 26 200 3	8 5 2 6	4
320	7680	4	240	11 25 201 3	13 9 5 10	8
384	9216	5	288	11 27 247 3	8 6 2 7	0
384	9216	3	288	11 24 250 3	16 9 7 10	4
384	9216	1	288	12 28 245 3	24 20 14 23	8

예로서, DMB 시스템은 균등 부호화와 불균등 부호화의 두 가지 부호화 방법을 사용한다. MUSICAM (Masking pattern adapted Universal Subband Integrated Coding And Multiplexing) 프레임의 제어정보나 데이터 정보와 같이 에러가 나면 전체 성능에 큰 영향을 줄 부분과, 음성정보와 같이 수신단에서 보상이 가능하고 영향이 적은 부분에 대해서는 표 5와 같은 서로 다른 코드율의 오류 정정 코드를 사용하고, 스트림 또는 패킷 모드의 데이터 채널은 한 프레임에 같은 코드율의 오류 정정 코드를 사용한다. 다양한 비트 rate와 protection level에 따라 각 서브블록들이 취하는 펑처링 벡터의 단계를 보여주며, 컨볼루션 인코더의 출력에서 64비트의 정수배를 하나의 워드 길이로 나타내기위해 추가되는 padding 비트수를 나타내고 있다. protection 레벨 1(P=1)은 가장 높은 protection을 가하는 것으로, 상대적으로 펑처링 벡터의 단계(PI)가 높음을 알 수 있다. 이는 정확한 데이터 복구를 위해 펑처링되어 없어지는 비트수가 낮아진다는 것을 뜻한다.

나. Branch Metric Unit

비터비 디코더의 첫 번째 단계가 입력 데이터의 Branch Metric을 구하는 과정이다. Branch Metric은 입력 데이터가 전송할 때 생기는 오류의 크기를 나타내는 것으로, 그 계산은 다음과 같이 3가지 방법으로 계산할 수 있다.

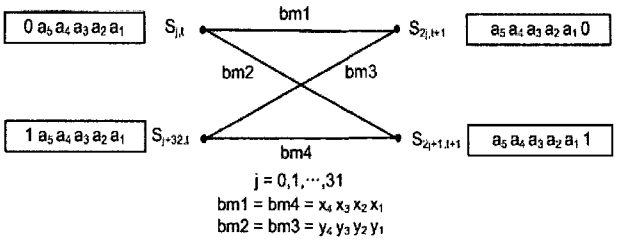


그림 6. DMB 컨볼루션 인코더의 트렐리스도 (R=1/4, K=7)[7]

Fig. 6. DMB's convolution encoder's trellis (R=1/4, K=7)[7].

$$ED = \sqrt{(r_i - v_i)^2} \quad \text{--- ①}$$

$$SED = (r_i - v_i)^2 \quad \text{--- ②}$$

$$AED = |r_i - v_i)| \quad \text{--- ③}$$

첫 번째 수식은 유클리디언 거리(euclidean distance)이며, 두 번째 수식은 자승 유클리디언 거리(squared euclidean distance), 그리고 마지막 세 번째 수식은 근사화 유클리디언 거리(approximated euclidean distance)라 한다[6]. 일반적으로 유클리디언 거리는 ROM(Read Only Memory)을 이용하여 구현되고, 자승 유클리디언 거리는 가산기, 곱셈기, 감산기 등을 이용하며, 마지막으로 근사화 유클리디언 거리는 인버터와 가산기를 이용하여 구할 수 있다. 유클리디언 거리는 곱셈기를 사용하므로, 비터비 디코더의 복잡도(complexity)를 상당히 증가시키므로, 저전력, 고속에 초점을 맞추고 있는 DMB 시스템을 위해서는 근사화 유클리디언 거리를 사용하게 된다.

DMB시스템에서는 컨볼루션 인코더의 구속장이 7이므로, 모두 64개의 상태를 가진다. 이것을 트렐리스로 나타내기에는 너무 복잡하고 직관적으로 얻을 수 있는 정보도 적으므로, 일반적으로 나비 모양의 상태쌍으로 나타낸다. 그림 6은 R=1/4, K=6 컨볼루션 코드에 대한 butterfly구조를 나타내는 것으로써, 비터비 디코더 설계에서 가장 널리 사용되는 구조이며 기본적인 processing단위이다. butterfly module은 4가지 상태 $S_{j,t}$, $S_{j+32,t}$, $S_{2j,t+1}$, $S_{2j+1,t+1}$을 포함하고, 이들 4가지 상태의 유일한 차이는 '0','1'의 차이에 있다. 현재 상태가 t시간 구간에서 $S_{j,t}$라 가정하고 입력 비트가 '0'이라하면, 다음 상태는 t+1시간구간에서 $S_{2j,t+1}$이 되고 출력 branch 심볼은 bm1 = (x_4 x_3 x_2 x_1)이 된다. 이와는 반대로 입력 비트가 '1'이라하면, 다음 상태는 $S_{2j+1,t+1}$이 되고 출력

branch 심볼은 bm2 = (y_4 y_3 y_2 y_1) 이 된다. butterfly 구조는 bm1 = bm4 = (y_4 x_3 x_2 x_1) 과 bm3 = (y_4 y_3 y_2 y_1) = ~(x_4 x_3 x_2 x_1)와 같은 특징을 가지고 있어, 하드웨어 구현을 단순화시킬 수 있다. 또한 다음 상태의 MSB(Most Significant Bit)는 traceback 동작을 위한 decision 비트로 사용된다. butterfly 구조를 바탕으로 수신된 데이터와 trellis diagram상의 codeword사이에서 branch metric을 계산할 수 있다. DMB시스템을 위한 컨볼루션 인코더의 구조적 특징 때문에 branch metric값은 8가지만이 존재하며, 그 값은 codeword와 수신된 데이터를 이용하여 근사화 유클리디언 거리로 bm_{0000}, bm_{0010}, bm_{0100}, bm_{0110}, bm_{1001}, bm_{1011}, bm_{1101}, bm_{1111}과 같이 나타내었다. 또한 이전 에러 metric값의 합인 path metrics $PM_t(S_{j,t})$과 $PM_t(S_{j+32,t})$은 각 시간구간에서 발생하는 $S_{j,t}$, $S_{j+32,t}$값을 가지고 구할 수 있다[7].

그림 7과 같이 구속장이 7이므로 64개의 상태수가 발생하며, 이를 butterfly구조로 변형하면 32개의 butterfly unit이 필요하게 된다. 채널을 거쳐 에러를 가진 데이터

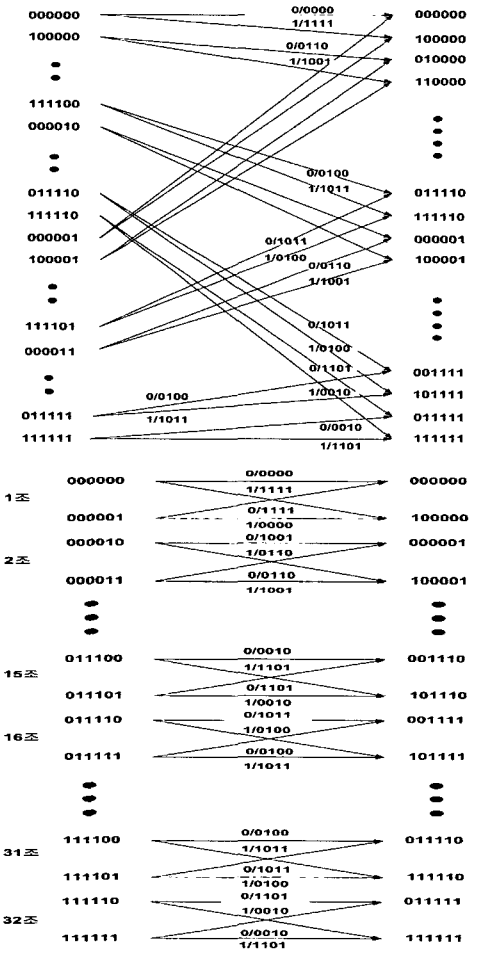


그림 7. 트렐리스 다이어그램과 버터플라이 구조
Fig. 7. Trellis diagram and butterfly structure.

를 높은 신뢰도로 복호화하기 위해서는 경판정(hard decision) 방식보다는 연판정(soft decision) 방식을 사용해야 하며, 본 논문에서는 4비트 연판정 방식을 이용하여 설계하였다. 4비트 연판정 방식을 사용하면, 경판정 방식에 비해 2.2dB의 프로세싱 이득을 얻을 수 있고, 또한 양자화 레벨이 무한대일 때와 비교해서 거의 성능 차이가 없는 장점이 있다.

다. Add-Compare-Select Unit

ACS블록은 비터비 디코더의 계산량 중에서 가장 많이 필요로 하는 부분이다. 하나의 ACS unit은 입력으로 각각 2개의 path metric과 branch metric값을 취하고 출력으로 새로운 2개의 path metric과 decision 비트를 내놓게 된다. 위의 경로를 택하게 되면, decision 비트 '0'을 아래 경로를 선택하면 '1'의 정보를 trace-back register set에 전달한다. 위의 과정이 수행된 후, 최소의 path metric값을 갖는 경로를 선택하고, 그 값을 path metric normalization으로 전달하여 overflow를 방지한다. 매 시간 구간마다 위의 연산과정이 반복되어 수행된다. 일반적인 구조에 적용된 과정을 수식과 그림을 이용하여 아래와 같이 나타내었다. N((N=상태수) ACS 연산은 아래와 같이 recurrent equation에 따라 반복 수행된다.

$$Min[\{BM_i(c,0)+PM_{i-1}(c)\},\{BM_i(a,0)+PM_{i-1}(a)\} \quad --④$$
$$Min[\{BM_i(a,1)+PM_{i-1}(a)\},\{BM_i(c,1)+PM_{i-1}(c)\} \quad --⑤$$

DMB 시스템을 위한 ACS블럭에서는 수신된 부호어에 대해 각 상태에서 결정된 2^M (M=인코더에 사용된 레지스터 수) 즉 64개의 생존 경로에 대한 path metric 값과 이에 해당하는 64개의 decision 비트가 생성된다.

하지만, 위와 같은 구조는 매 시간 구간마다 동일한

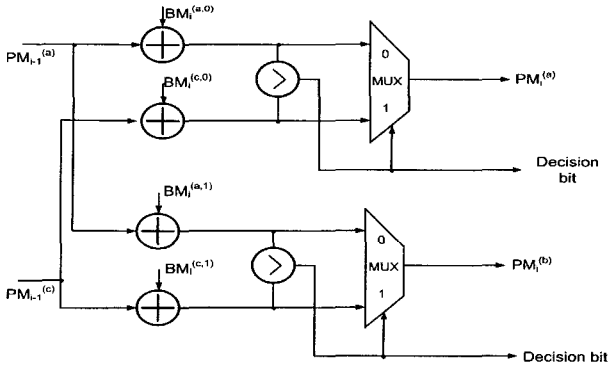


그림 8. 일반적인 ACS unit의 구조[8]
Fig. 8. Conventional ACS unit's architecture[8].

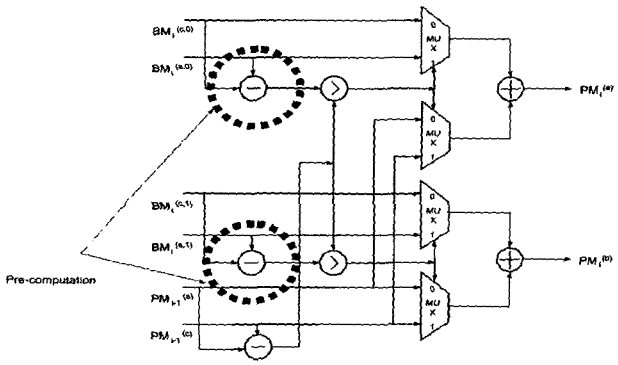


그림 9. 제안된 ACS unit의 구조
Fig. 9. Proposed ACS unit's architecture.

구조가 반복되므로 많은 복잡도를 가지고 있는 단점이 있다. 이에 많은 연산으로 인해 실시간 동작이 어려운 블록의 일부분을 앞단에서 계산해주는 선계산(pre-computation)기법 과 path metric값의 차이를 branch metric값들과 비교하는 분할기법(decomposition method)을 이용하여, 비교기 및 다중화기의 복잡도를 줄일 수 있는 구조를 제안하고자 한다. 이를 수식과 그림으로 표현하면, 아래와 같다.

$$BM_i(c,0)-BM_i(a,0)+PM_{i-1}(c)-PM_{i-1}(a) \geq or \leq 0 \quad --⑥$$
$$BM_i(c,1)-BM_i(a,1)+PM_{i-1}(c)-PM_{i-1}(a) \geq or \leq 0 \quad --⑦$$
$$BM_i(c,0)-BM_i(a,0) \geq or \leq PM_{i-1}(a)-PM_{i-1}(c) \quad --⑧$$
$$BM_i(c,1)-BM_i(a,1) \geq or \leq PM_{i-1}(a)-PM_{i-1}(c) \quad --⑨$$

⑥,⑦번의 수식에서와 같이 두 값의 최소값을 구하는 과정을 먼저 path metric값과 branch metric값의 차이값을 구한 다음 비교하여 그 값이 음인지 양인지를 판단할 수 있으며, 다시 변형을 가하여 ⑧,⑨번 수식과 같이 branch metric의 차이값과 path metric의 차이값 중 어느 값이 큰 지를 판단하여 최소값을 결정하는 식으로 바꿀 수 있다.

일반적인 구조를 분할기법(decomposition method)을 통해 $\min[\{BM_i(c,0) + PM_{i-1}(c)\}, \{BM_i(a,0) + PM_{i-1}(a)\}]$을 구하는 대신에 $\{BM_i(c,0) - BM_i(a,0)\}$ 값과 $\{PM_{i-1}(c) - PM_{i-1}(a)\}$ 값을 비교하여 최소의 값을 구할 수 있다.

라. Path Matric Normalization

eureka-147에 의해, 구속장 7을 가지므로, 매 시간구간마다 64개의 새로운 path metric이 계산되어야 한다. speed requirement때문에 매 단계마다 모든 path metric을 계산하기 위해서 parallel 구조를 사용하여 구

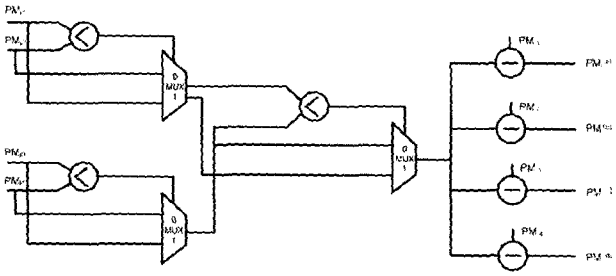


그림 10. 일반적인 PMN unit의 구조[9]
Fig. 10. Conventional PMN unit's architecture[9].

현되어진다. 하지만, path metric값이 크다면 system은 saturation이 된다. 항상 path metric값은 증가하게 되어 있어, 이를 제한해야하며, 또한 그 값들은 trace-back과정에 기여하지 못하므로 각 단계에서 적당한 path metric값을 할당해놓고 이를 정규화시키는 방법을 통해 구현을 하게 된다.

그림 10은 Rescaling approach 방법으로 metric vector($m_s(k) \mid s \in S$)로 부터의 일정한 값의 감산은 비터비 알고리즘에 영향을 끼치지 않는다는 성질을 이용한다. Rescaling functions R_k, minimum metric C_k라 하면 $R_k(x) = X-C_k$, $C_k = \min\{m_s(k) \mid s \in S\}$, $k \in N$가 된다. 여기서, S는 컨볼루션 인코더의 상태값, N는 시간 간격을 나타낸다. 따라서, Rescaling approach 구현에서 metric $m_s(k)$ 는 rescaled version인 $R_k(m_s(k))$로 바뀌어 진다[9].

$R_k(m_s(k)) = \min\{ R_{k-1}(m_s(k-1)) + b_{st}(k) \mid s \in S \}$ 이 되고, 최소의 path metric값을 결정하기 위한 comparison tree의 depth는 $\log_2(|S|)$에 비례한다. 일반적인 방법이 그림 10과 같이 최소의 path metric을 찾아 그 값을 이용하여 path metric값과 차이를 구하는 것이다. 최소값을 찾는 과정은 comparison tree를 이용하여 구현되기 때문에 비터비 디코더의 speed requirement를 만족시키지 못할 수 있다.

따라서, comparison tree를 단순화시킬 수 있도록 고정된 값을 이용하여 최소값을 찾는 구조와, ACS블록과의 연동하여 동작할 수 있도록 하는 새로운 cell을 아래와 같이 제안하고자 한다.

본 논문에서 제안하고자 하는 고정치(fixed value)를 이용한 PMN(path metric normalization)블록은 정규화 과정이 NOR gate logic을 이용하여 동작되어야 하는지 판단하는 것으로 구성함으로써, 고속의 데이터 처리가 가능하도록 하였다. DMB 시스템의 Code Rate=1/4, 구속장은 7이므로, path metric의 비트 수는 branch metric과 구속장을 이용하여 구할 수 있다. 구속장 K=7,

branch metric의 최대값과 최소값이 각각 BM_{MAX}=31, BM_{MIN}=0이므로 path metric 최대값과 최소값의 차이는 186이 된다. 그것은 path metric을 나타내는데 9비트가 필요하는 것을 의미한다. 그러나, 더 적은 비트로 path metric을 나타내는 것이 가능하다. 즉 BER성능의 감소없이 speed를 향상시킬 수 있도록 비트수를 줄일 수 있다. 여기서 고정치(fixed value)로 "0001000"을 선택하고, path metric값들의 상위 4비트를 이용하여 고정치(fixed value)보다 값이 큰지를 NOR gate로 비교하게 된다. 먼저 개개의 path metric값이 고정된 값보다 큰지 작은지를 NOR gate를 이용하여 판별하고, 그때의 NOR gate출력을 이용하여, 그 다음 모든 path metric 값이 고정된 값보다 큰지 작은지를 다시 NOR gate로 판별한다. 그 값이 고정치(fixed value)보다 작다면 NOR gate 출력이 0이 되어 정규화 과정이 수행되지 않고, NOR gate 출력이 1이 되면 정규화 과정이 수행된다.

그림 11의 우측부분이 제안된 PMN구조를 나타내며 butterfly구조 32개가 병렬처리 되므로, 제안된 cell이 매시간 간격마다 32개가 사용되어진다. rescaling approach에 비해 새로운 논리회로로 4비트 NOR gate 64개, 64비트 NOR gate가 사용되어지지만, comparison tree구조를 단순화시킴으로써 저전력의 효과도 가져올 수 있었다. 다시 정리하면, 그림 11의 좌측 사각형에서는 선계산(pre-computation)과 분할기법(decomposition method)을 사용하여 복잡도를 줄여, 저전력의 ACS를 구현하였고, 우측의 사각형블록은 ,새로운 PMN을 제안함으로써 고속의 데이터 처리가 가능하도록 하였다. 두 가지 unit을 결합시킨 구조를 유기적으로 적용할 수 있도록 하여, 전력소모, 면적, 데이터 처리량의 관계에서 상호보완관계를 가지고 있는 구조를 모두 만족시키는 최적화된 DMB시스템을 위한 전용 cell을 이용하여 에러정정코드를 구현하였다.

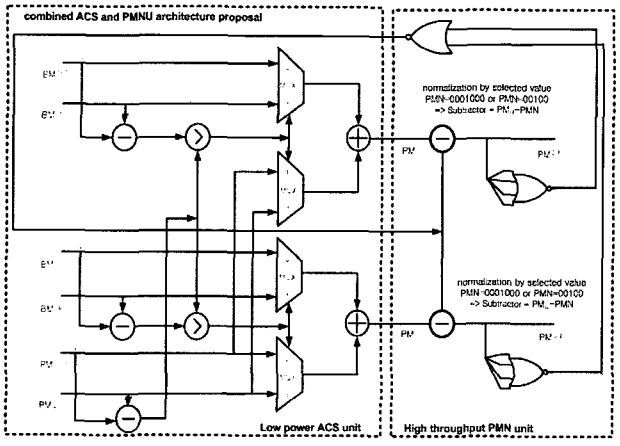


그림 11. ACS와 PMN이 결합된 새로운 cell 제안
Fig. 11. The combined ACS and PMN unit cell proposal.

마. Trace-Back Unit

일반적으로 survival path 저장과 복호를 위한 알고리즘은 register exchange와 trace-back 두 가지로 나뉜다. 전자는 개념적으로 가장 간단하고 일반적으로 사용되었던 방식이며, 매 event마다 decision 비트를 옮겨가야하므로 과다한 전력 소비와 칩 면적이 증가하는 단점을 가지고 있다. 구속장이 길고 고성능을 갖는 디코더에서는 순환 포인터 배열 형태로 decision 비트를 저장하는 trace-back방법이 사용된다. Trace-back구조에서는 복잡도가 낮은 반면 출력을 내놓은 속도가 낮은 특성을 갖고 있어, 고속 통신 시스템에서는 대체적으로 trace-back구조를 선택하여 복잡도를 줄이는 경향이 있다. 이에 본 논문에서는 trace-back구조를 선택하여 데이터를 복호화하였다. 그림 12에서와 같이 trace-back unit에서 역추적 방법을 이용한 trace-back 메모리는 개념적으로 쓰기 영역과 읽기 영역으로 나뉘어진다. 각각의 복호화 단계 동안 읽기 영역(RD)에서는 생존자 패스를 역추적 및 복호화 하는 일을, 쓰기 영역(WR)에서는 생존자 열을 쓰는 일을 한다. 이때, 읽기 동작을 수행하는 시간과 쓰기 동작을 수행하는 시간은 같아야 한다. 읽기 영역은 세부적으로 역추적 읽기(TB-RD)와 디코드 읽기(TB-DC)블록으로 나뉘어지는데, 역추적 읽기의 블록은 디코드 블록 시작 부분의 초기 상태를 추정하기 위하여 사용된다. 0~63은 상태수, TB-RD에 있는 화살표는 traceback 포인터의 패스, TB-DC는 디코더 포인터의 패스를 각각 나타낸다. WR에서는 새로운 decision 비트들이 메모리에 쓰이고 있는 것을 보여주

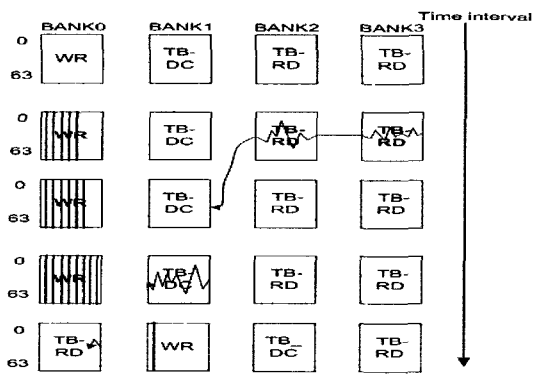


그림 12. One pointer 알고리즘의 trace-back 메모리 갱신
Fig. 12. Trace-back memory reformation of the one-pointer algorithm.

고 있다. 이런 방식을 이용하면 연속적으로 메모리에 데이터를 읽고 쓸 수 있다[10].

Trace-back depth는 비터비 디코더의 BER성능과 밀접한 관련이 있으므로 선택시 주의하여야 한다. 일반적으로 Forney에 의하면, 구속장의 5 ~ 6배 정도의 길이를 선택하면, 모든 생존자 패스는 한 곳으로 수렴하며 신뢰할 수 있는 복호 데이터를 얻을 수 있다고 한다[11] 이를 이용하여 역추적 깊이 T의 값을 정하도록 하였다.

Ⅲ. 구현 결과 및 검증

수신된 데이터는 그림 13과 같이 처음에 디펑처링을 과정을 거치게 된다. 디펑처링을 거친 데이터는 BM에서 매 시간간격마다 8가지의 hamming distance를 구하고, ACS에서는 최소값을 갖는 상태 정보를 TB에 전달하게 되고, 한 상태에 들어오는 두 가지 에러 metric값 중 최소값을 MS에 전달하여 normalization과정을 거쳐 그 값을 ACS에 되돌려준다. TB에서는 ACS와 MS에서 전달된 정보를 이용하여 데이터를 역추적하게 되고, 복호화된 값은 시간적으로 순서가 바뀌어 있으므로, LIFO에 저장하였다가 매 32 클럭마다 출력으로 내보내어 복호화된 정보를 얻게 된다.

가. 디펑처링 구조

디펑처링은 컨벌루션 인코더에서 생성된 데이터 중 펑처링 과정을 통해 전송되지 않은 비트열을 임의의 비트열로 복원하는 과정이다. 디펑처링 과정은 전송된 프레임의 protection profile 정보를 참조하여 이루어지는데, 표 4, 5에서 보듯이 DMB 시스템에서 protection profile의 형태는 전송 프레임의 채널 및 코드율 적용방식에 따라 다르게 나타난다. 따라서 전송 프레임의 채널 및 코드율 적용방식에 관계없이 디펑처링 과정을 수행하기 위해서는 공통적으로 적용할 수 있는 규약이 필요하다. 본 논문은 표 6, 7과 같이 채널에 관계없이 모든 Protection Profile을 (L1, L2, L3, L4, L5), (PI1, PI2, PI3, PI4, PI5)로 정하였다. 이때 L1, L2, L3, L4의 값은 표 3, 4의 L1, L2, L3, L4에 32를 곱한 값으로 하였다. 여기에서 32는 비터비 디코더의 입력으로 컨볼루션 부호화된 4개의 심볼이 들어가기 때문에 실제 디펑처링 과정에서는 4 비트 펑처링 벡터가 사용되고, 구간 L 하나에 4번 적용되는 32 비트 펑처링 벡터를 8번 나누어 사용하기 때문에 정해진 값이다. 따라서, FIC와 MSC의 EEP 코딩에서 L3, L4는 0 의 값을 가지며, 0

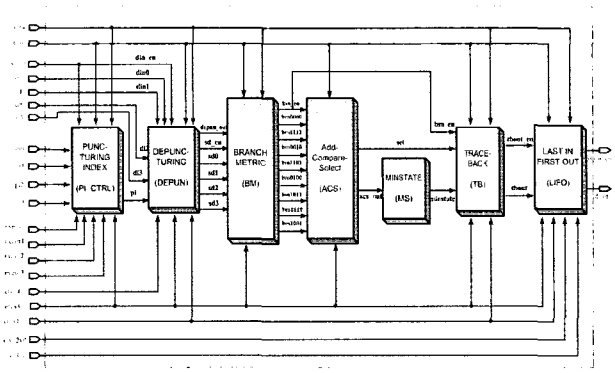


그림 13. 비터비 디코더의 구성도
Fig. 13. Viterbi decoder's top view.

표 6. 변형된 FIC protection profile
Table 6. Modified FIC protection profile.

P (protection level)	전송 모드	L1	L2	L3	L4	L5	PI1	PI2	PI3	PI4	PI5
-	Ⅰ,Ⅱ,Ⅳ	672	96	0	0	6	16	15	-	-	8
-	Ⅲ	928	96	0	0	6	16	15	-	-	8

표 7. 변형된 MSC Protection Profile
Table 7. Modified MSC Protection Profile.

Bit Rate	P	L1	L2	L3	L4	L5	PI1	PI2	PI3	PI4	PI5	padding bits
32	5	96	128	544	0	6	5	3	2	-	8	0
32	4	96	96	576	0	6	11	6	5	-	8	0
32	3	96	128	448	96	6	15	9	6	8	8	0
32	2	96	128	448	96	6	22	13	8	13	8	0
32	1	96	160	416	96	6	24	17	12	17	8	4
48	5	128	96	832	96	6	5	4	2	3	8	0
48	4	96	128	832	96	6	9	6	4	6	8	0
48	3	96	128	832	96	6	15	10	6	9	8	4
48	2	96	128	832	96	6	24	14	8	15	8	0
48	1	96	160	800	96	6	24	18	13	18	8	0
·	·	·					·					·
320	5	352	832	6400	96	6	8	5	2	6	8	4
320	4	352	800	6432	96	6	13	9	5	10	8	8
384	5	352	864	7904	96	6	8	6	2	7	8	0
384	3	352	768	8000	96	6	16	9	7	10	8	4
384	1	384	896	7840	96	6	24	20	14	23	8	8

상태 merging을 위해 추가된 6개의 0 는 24 비트 펑처링 벡터 V_T에 의해 펑처링 되므로, 이에 대한 블록 크기 및 protection 단계를 나타내는 L5, PI5는 FIC와 MSC에 관계없이 각각 6, 8의 값을 갖는다.

나. ACS 성능분석

ACS블록은 하위블록인 ACS_CELL로 구성되어있다. ACS_CELL은 각 상태로 들어오는 두개의 branch metric값과 하나의 path metric값을 각각 더하여 최소값을 구하는 블록이다. 여기서 path metric의 데이터는 9비트이고, branch metric의 데이터는 5비트를 받게 되

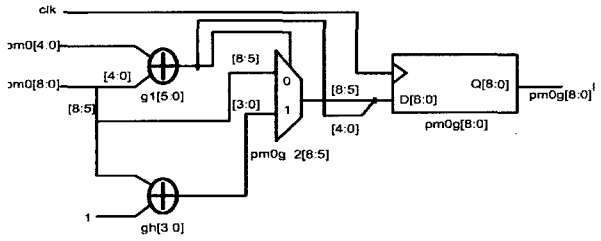


그림 14. Carry를 이용한 9비트 가산기
Fig. 14. 9bit adder using the carry.

표 8. 일반적인 ACS와 제안된 ACS구조의 연산량
Table 8. Number of operations for conventional ACSU and proposed ACSU.

일반적인 ACS구조	
operation	type
$PM_{i1}(a) = PM_{i-1}(a)+BM_i(a,0)$	9 to 5 bit addition
$PM_{i2}(a) = PM_{i-1}(c)+BM_i(c,0)$	9 to 5 bit addition
$PM_{i1}(b) = PM_{i-1}(a)+BM_i(a,1)$	9 to 5 bit addition
$PM_{i2}(b) = PM_{i-1}(c)+BM_i(c,1)$	9 to 5 bit addition
$PM_i(a) = \min(PM_{i1}(a),PM_{i2}(a))$	9bit comparison and select
$PM_i(b) = \min(PM_{i1}(b),PM_{i2}(b))$	9bit comparison and select

변형된 ACS구조	
operation	type
$PM = PM_{i-1}(a)-PM_{i-1}(c)$	9 bit subtration
comp $(PM,BM_i(c,0)-BM_i(a,0))$	9 to 5bit comparison and select
comp(PM,BMi(c,1)-BMi(a,1))	9 to 5bit comparison and select
$PM_i(a) = PM_{i-1}(n)+BM_i(n,0)$	9 to 5 bit addition
$PM_i(b) = PM_{i-1}(n)+BM_i(n,1)$	9 to 5 bit addition

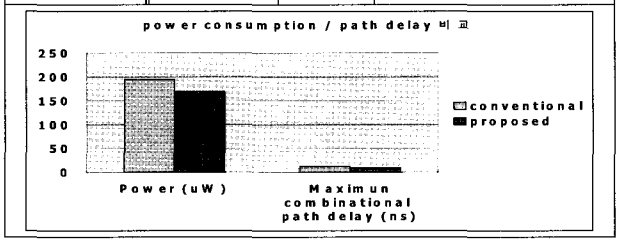
어 있다. 일반적인 구조에서는 9비트 가산기를 사용하고 있으나, 실시간동안 동작하는 비터비 복호기의 speed를 향상시키기 위해서 그림 14와 같이 하위 5비트에 대한 가산기를 수행한 후 발생하는 carry에 의해 상위비트 가산기의 수행유무를 판단할 수 있는 변형된 구조를 사용함으로써, 구조의 최적화를 수행하였다.

구속장이 7이므로 64가지의 상태가 발생하고, 각 상태에서 2개의 9비트 가산기를 사용함으로 각 시간구간마다 128개의 9비트 가산기를 사용함으로, 상당한 switching activity를 줄일 수 있다. 그림 14와 같이 복잡도는 약간 증가를 했지만, 5-비트 가산기의 carry out의 출력에 따라 나머지 4-비트 가산기의 동작유무를 판단함으로써, 데이터 복호과정에서 switching activity를 줄일 수 있었다. 제안된 구조와 일반적인 구조사이에 butterfly unit을 사용하였을 때 정성적으로 연산량을 비교해 보면 아래와 같이 정리할 수 있다.

위의 분석표에서와 같이 비교기의 입력비트수가 줄어들고, 가산기의 수가 줄어들게 되어 복잡도와 전력소모가 일반적인 구조보다 향상됨을 확인할 수 있다. 정성적인 결과를 토대로 실제로 저전력을 목적으로 한 ACS블록을 synopsys design compiler를 이용하여 전력소모, 면적 등의 성능을 측정하기 위해 3.3V 공급전압과 10MHz의 동작 주파수로 설정하고, target library는 교육용으로 제공되는 Hynix 0.35um(2 metal 3.3V)을 사용하였다.

표 9. 일반적인 ACS와 제안된 ACS구조의 전력과 면적
Table 9. Power and area for conventional ACSU and proposed ACSU.

ACSU	Area	Power (uW)	Maximum combinational path delay(ns)
conventional	96023.67969	194.007	12.341
proposed	83328.24244	168.493	8.754
% reduction	13.2	13.1	29.1



다. PMN 성능분석

본론에서 언급한 일반적인 구조와 제안된 PMN구조를 verilog HDL(Hardware Description Language)로 설계를 완료하고, synplify사의 합성툴을 이용하여 gate level에서 설계검증을 수행하였다. 합성툴에서 제공하는 합성레포트를 이용하여 gate delay를 비교 분석하여 데이터 처리량에 대한 인자를 고려하였다. 일반적인 PMN 구조는 9비트 comparison tree를 이용하여 최소값을 구하고 있으며, 구해진 최소값을 이용하여 path metric값을 정규화 시키고 있다. 반면에 제안된 구조는 고정된 값을 이용하여 PMN unit의 동작상태를 판단하여 구현된 구조를 보여주고 있는 것으로 각각 임계 경로(critical path)를 아래와 같이 분석할 수 있다.

합성 리포트에서의 gate delay내용을 정리하면, 일반적인 구조에서는 IBUF(input buffer delay)와 OBUF(out buffer delay)는 4.859[ns]의 gate delay와 1.520[ns]의 net delay로 이루어진다. 사용된 combinational logic들의 delay는 13.137 [ns]의 gate delay와 31.040 [ns]의 net delay로 구성된다. 위 수치를 합산하면 maximum combinational gate delay는 50.656 [ns]가 된다. logic delay는 전체 delay중에 35.7%를 차지하고, route delay

는 64.3%가 되므로, net delay에서 상당한 부분이 소요됨을 알 수 있다.

제안된 구조에서는 고정치(fixed value)에 의해 비교기 연산을 NOR gate연산으로 대치함으로써, 전체적인 gate delay를 줄이고, speed requirement를 향상시키고자 하는데 있다. 최종 NOR gate출력이 '1'이 되면 고정치(fixed value)와 정규화 과정이 진행되고, 그렇지 않으면 구해진 path metric값이 그대로 전달되는 과정을 거치게 된다. 표 10과 같이 path delay는 gate delay와 net delay로 이루어지며, top module에 사용되는 cell들의 delay를 합산하여 구할 수 있다. target device로는 Xilinx사의 XCV200E-6-PQ240을 사용하여 gate delay를 구하였다.

(1) 제안된 구조의 maximum combinational path delay : 2.317(IBUF)+17.679(LOGIC)+4.062(OBUF)=24.048[ns]

(2) logic delay : 9.143 [ns], (3) route delay : 14.915 [ns]로서, logic delay는 전체 delay중에 38%, route delay는 62%가 되므로, net delay에서 상당한 부분이 소요됨을 알 수 있다.

제안된 PMN구조를 사용했을 경우 표 11에서와 같이 gate delay는 52.5%정도 줄일 수 있었으며, 면적과 전력면에서도 우수한 걸로 나타났다.

라. Trace-back 구조

trace back unit은 크게 LIFO(Last In First Out)와 trace-back으로 구성되었으면, 비터비 디코더의 출력을 얻기 위해 32 클럭이 주어져 있기 때문에 한번의 쓰기 동작이 일어나는 동안 31번의 읽기 동작을 실행할 수 있다. 그림 15에서와 같이 각각의 tb_men은 64×32비트의 SRAM으로 구현되었고, one pointer 알고리즘을 이용하여 설계한 trace-back 메모리가 동작하는 과정은 다음과 같다. t0, t2, t4, … , t62와 같은 홀수 번째 32 사이클 동안에는 31번의 TB-RD과정과 1번의 WR과정을, t1, t3, t5, … , t63과 같은 짝수 번째 32 클럭 동안에는 29번의 TB-RD과정과 2번의 TB-DC과정 그리고 1번의 WR과정을 수행한다. 여기에서 짝수 번째 32 클럭 동안 두 번의 TB-DC과정을 거쳐 복호화된 값은 시간적으로 순서가 바뀌어 있으므로, LIFO에 저장하였다가 매 32 클럭마다 출력으로 내보낸다.

마. 디펑처드 비터비 디코더 시스템의 성능분석

위에서 제안된 구조를 사용하여 디펑처드 비터비 디코더를 구현하였을 경우 표 12와 같은 성능을 얻을 수 있었으며, 디펑처드 비터비 디코더 시스템 전체에서는 표 12를 표 2와 비교했을 때 면적 3.78%, 전력소모 12.22%, 최대 게이트 지연 23.80%의 감소율을 보였다.

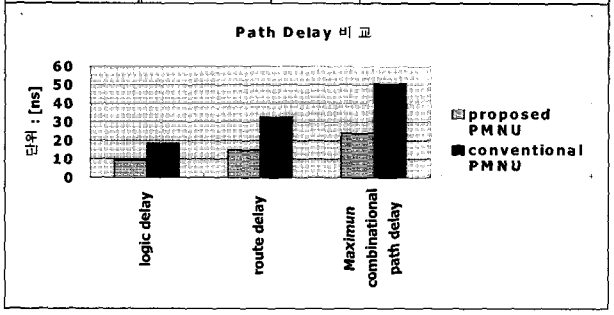
표 10. 제안된 PMN구조의 최대 경로 지연 시간
Table 10. Proposed PMNU's maximum combinational path delay.

```
Timing constraint: Default path analysis
Delay:              24.058ns (Levels of Logic = 10)
  Source:           input56<3> (PAD)
  Destination:      m_in13<7> (PAD)

  Data Path: input56<3> to m_in13<7>
                                    Gate     Net
    Cell:in->out      fanout        Delay    Delay    Logical Name (Net Name)
    ----------------------------------------------------------------------------
    IBUF:I->O             4         0.797    1.520    input56_3_IBUF (input56_3_IBUF)
    LUT4:I0->O            1         0.468    0.920    pnu721 (CHOICE588)
    LUT2:I1->O            1         0.468    0.920    pnu722 (CHOICE589)
    LUT4:I3->O            1         0.468    0.920    pnu742_SW0 (N29227)
    LUT4:I3->O            1         0.468    0.920    pnu742 (CHOICE591)
    LUT2:I0->O            1         0.468    0.920    pnu852 (CHOICE623)
    LUT4:I3->O           10         0.468    2.250    pnu876_SW0 (N29219)
    LUT4:I3->O           93         0.468    4.825    pnu876_3 (pnu876_3)
    LUT3:I0->O            5         0.468    1.720    Mmux_m_in49_Result<4>1 (m_in49_4_OBUF)
    OBUF:I->O                       4.602             m_in49_4_OBUF (m_in49<4>)
    ----------------------------------------------------------------------------
    Total                          24.058ns (9.143ns logic, 14.915ns route)
                                            (38.0% logic, 62.0% route)
```

표 11. PMN구조에 대한 area / power / path delay 비교
Table 11. Area / power / path delay comparison for PMN architecture.

PMNU	Area	Power (uW)	Maximum combinational path delay(ns)
conventional	79705.25781	266.799	50.656
proposed	56069.44922	206.337	24.048
% reduction	29.7	22.7	52.5



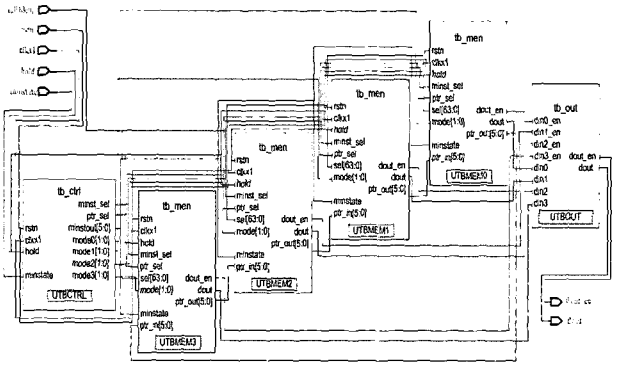
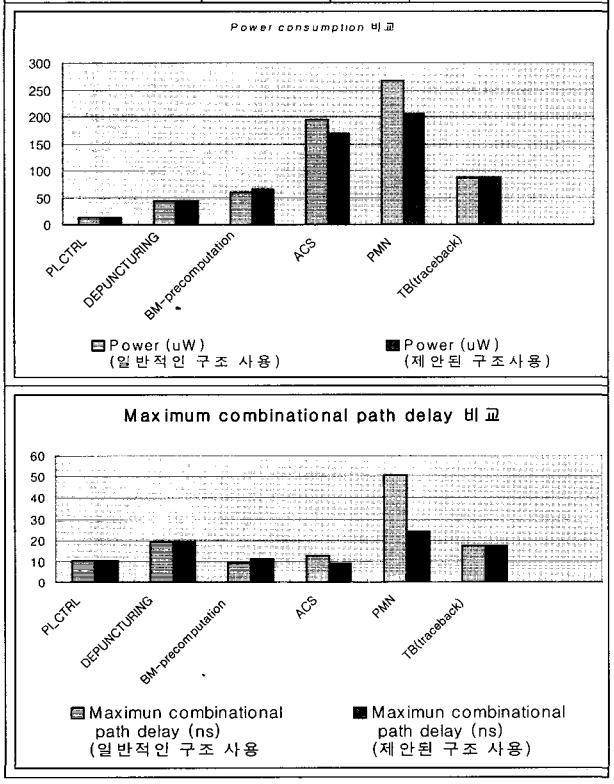


그림 15. Trace-back 구성도
Fig. 15. Trace-back top view.

표 12. 제안된 구조를 사용한 디펑처드 비터비 디코더의 성능분석

Table 12. Depunctured viterbi decoder's performance analysis using the proposed architecture.

	Area	Power (uW)	Maximum combina- tional path delay(ns)
pi_ctrl	501.37651	13.449	10.483
depuncturing	154157.2485	43.857	19.297
bm (pre-computation)	4733.67833	65.126	11.103
proposed acs	83328.24244	168.493	8.754
proposed pmn	56069.44922	206.337	24.048
tb(traceback)	624961.8183	87.129	17.305



Ⅳ. 결 론

본 연구에서는 실시간 COFDM시스템을 위한 효율적인 구조를 갖는 비터비 디코더를 설계하기 위하여 DMB시스템을 선정하여 제안된 구조를 검증하였다. 본문에서 제안된 세 가지 구조에 대해서 구속장 7, Code Rate 1/4의 디펑처드 비터비 디코더를 verilog HDL (Hardware Description Language)를 이용하여 설계하였고, modelsim에 의해 functional simulation을 수행하였다. PMN unit에서 필요한 고정된 값 선정 기준은 MATLAB을 이용하여 BER performance 확인검증을 거쳐 선택하였으며, synopsys design compiler와 FPGA(Field Programmable Gate Array) 테스트용으로 synplify를 이용하여 각각 합성을 통해 gate level 구조를 확인하였다. 여기서 생성된 합성 리포트를 분석하면 일반적인 구조와 제안된 구조의 gate count수와 maximum combinational gate delay을 분석할 수 있었다. 또한 synopsys를 이용하여 전력과 면적에 대한 인자도 고려할 수 있었다. area 정보는 일반적으로 2-input nand gate를 1로 보고 다른 cell들을 비교 수치로 나타내거나, 실제 면적인 square microns를 정보를 이용하지만, 여기서는 target library가 제공하는 전자의 방법을 선택하였다. 저전력을 목적으로 한 제안된 ACS 구조는 일반적인 구조에 비해 전력면에서 13.1%의 감소와 면적에서는 13.2%의 감소결과를 보였으며, gate delay을 줄임으로써 전체적으로 throughput을 높이는 것을 목적으로 제안된 PMN구조는 일반적인 구조에 비해서 gate delay를 52.5%를 줄일 수 있었다. 따라서, 성능분석결과를 토대로 실시간 COFDM시스템을 위한 비터비 복호기를 구현할 때 결합된 ACS와 PMN구조를 사용한다면, 일반적인 구조를 사용했을 때에 비해 switching activity와 logic연산을 줄일 수 있고, 고속의 데이터 처리가 가능하여 최적화된 디펑처드 비터비 디코더를 구현 할 수 있을 것이다.

참 고 문 헌

[1] Bernard Sklar, "Digital Communications Fundamentals and Applications," Prentice-Hall, pp. 381-429, 1989.

[2] ETS 300 401, "Radio Broadcasting System ; Digital Audio Broadcasting to mobile portable and fixed receiver," 2nd Ed., ETSI, May 1997.

[3] Y. Yasuda, K. Kashiki, and Y. Hirata, "High-rate punctured convolutional codes for soft decision Viterbi decoding," IEEE Trans. Commun., vol. COM-32, pp. 315-319, Mar 1984.

[4] J. Hagenauer, "Rate-compatible punctured convolutional codes (RCPC codes) and their applications," IEEE Trans. Commun., vol. COM-36, pp. 389-400, Apr 1988.

[5] Jon Feldman Ibrahim Abou-Faycal, "A Fast Maximum-Likelihood Decoder for Convolutional Codes," IEEE trans. pp. 7803-7467 , Mar 2002.

[7] Inyup Kang, Alan N. Willson Jr. "Low-Power Viterbi Decoder for CDMA Mobile Terminals," IEEE Journal of Solid-State Circuits, vol. 33, no. 3, Mar 1998.

[8] 김재석 외 3명 공저, "이동 통신용 모뎀의 VLSI

설계 (CDMA/OFDM/MC-CDMA모뎀).", 대영사, pp. 235-305, 2001.
[9] Inkyu Lee and Jeff L. Sonntag, "A New Architecture for the Fast Viterbi Algorithm," IEEE Trans. Commun, vol. 51, pp. 1624-1628, Oct 2003.
[10] Andries P. Hekstra, "An Alternative to Metric Rescaling in Viterbi Decoders," IEEE Trans. Commun, vol. 37, pp. 1220-1222, Nov 1989.
[11] G. Feygin, P. G. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders", IEEE Trans. on Commun., Vol. 41, No. 3, pp. 425-429, Mar 1993.
[12] G. D. Forney Jr., "The Viterbi Algorithm," proc. IEEE, vol.61, pp. 268-278, Mar 1973.
[13] G. Fettweis and H. Meyr, "Parallel Viterbi algorithm implementation: Breaking the ACS bottleneck," IEEE Trans. Commun., vol. 37, pp. 785-790, Aug 1989.
[14] Jung-Gi Baek, Sang-Hun Yoon and Jong-Wha Chong "Memory Efficient Pipelined Viterbi Decoder with Look-ahead Trace Back," The 8th IEEE International Conference, vol. 2, pp. 769-772, Sept 2001.

저 자 소 개

황 종 희(정회원)
2001년 인하대학교 반도체공학과 학사 졸업.
2005년 인하대학교 정보통신 공학과 석사 졸업.
2005년~현재 LG. Philips LCD 회로설계팀 재직.
<주관심분야 : 통신, LCD구동회로, 신호처리, VLSI 및 SoC설계>

김 동 순(정회원)
1997년 인하대학교 전자재료 공학과 학사 졸업.
1999년 인하대학교 전자재료 공학과 석사 졸업.
1999년~현재 전자부품연구원 DMB개발사업단 선임연구원
<주관심분야 : 통신, DMB, VLSI 및 SoC설계>

이 승 열(정회원)
2003년 인하대학교 전자전기 공학과 학사 졸업.
2005년 인하대학교 정보통신 공학과 석사 졸업.
2005년~인하대학교 정보통신 공학과 박사 과정.
<주관심분야 : 통신, WPAN,. FFT, VLSI 및 SoC 설계>

정 덕 진(종신회원)
1970년 서울대학교 전기공학과 학사 졸업.
1984년 미국 Utah State University (석사)(Electrical Eng.).
1988년 미국 University of Utah State(공학박사) (Electrical Eng.).
1989년~현재 인하대학교 정보통신공학과 교수
<주관심분야 : VLSI 및 SoC설계, 컴퓨터 구조 및 Embedded System 설계>