가우시안 및 버스트성 잡음채널에서의 PSS 방식 Viterbi Algorithm 성능분석과 고속 설계

양 형 규† · 정 지 원††

요 약

본 논문에서는 계산량 및 전력소모를 줄일 수 있는 PSS 방식의 Viterbi 복호기를 성능 및 메모리 복잡도 관점에서 기존의 방식과 비교하여 Monte Carlo 모의실험 기법으로 분석하였다. 모의실험에서 채널모델을 가우시안 잡음만 존재하는 채널과 버스트성 잡음이 혼재하는 채널로 구성하였고, 버스트성 잡음이 혼재하는 채널에 convolutional 인터리빙을 적용한 경우 메모리 복잡도 및 전력 소모를 줄일 수 있었다. 또한 PSS 방식 Viterbi 복호기의 고속 설계 방안을 검토하여, VHDL tool을 이용하여 $r=1/2$, $k=3$인 PSS 방식 Viterbi 복호기를 구현하였으며, VHDL 구현 결과 복호됨을 확인하였다.

Performance Analysis and High-Speed Design of PSS-type Viterbi Algorithm in Gaussian and Burst Noise Channel

Hyung-Kyu Yang† · Ji-Won Jung††

ABSTRACT

In this paper, we analyze the performance of the PSS-type Viterbi decoder which can reduce calculations and power consumption using the Monte Carlo simulations. Gaussian and burst noise channels are considered in this simulation, and we achieve that convolutional interleaver can reduce complexity and power consumption in burst noise channel. In order to implement the high-speed PSS-type Viterbi decoder, the architectures of decoder are presented, and we implemented the PSS-type Viterbi decoder for $r=1/2$, $k=3$ using the VHDL tool, and prove the decoding process.

1. 서 론

Convolutional 부호는 우수한 성능 때문에 오늘날 디지털 통신시스템에서 널리 사용되고 있다 하지만, 그 복호 알고리즘인 Viterbi 알고리즘은 많은 계산량과 메모리 그리고 고속화에 대한 과제는 여전히 남아있다. 계산량 및 메모리 복잡도 문제를 해결하기 위해 여러 알고리즘이 제안되었는데, 대표적인 것이 PSS(Probability Selecting State) 방식 Viterbi 복호 알고리즘이다[1]

Viterbi 복호 알고리즘에서 복호기의 Trellis diagram의 상태수는 부호기의 구속장수 k에 지수함수적으로 증가하며, 각 상태마다 수신 신호와의 거리차를 계산하고 저장해야 한다. 따라서, 복호기의 복잡도 및 계산량을 줄이기 위해서는 이러한 Trellis diagram의 상태수를 줄여야 한다

PSS 방식의 Viterbi 알고리즘은 기존의 방식과 달리 수신신호를 직접 복호하지 않고 수신신호로 부터 채널 잡음을 분리하여 분리된 잡음만을 복호기에 입력하는데, 잡음신호의 분포는 가우시안 분포특성을 가진다 따라서 가우시안 분포특성에 따라, 높은 E_b/N_o인 때 성능저하를 초래하지 않으면서 상태수를 줄일 수 있

† 정 회 원 : 강남대학교 신전전공학부 교수
†† 정 회 원 : 한국해양대학교 전파공학과 교수
논문접수 1999년 11월 15일, 심사완료 : 2000년 5월 9일

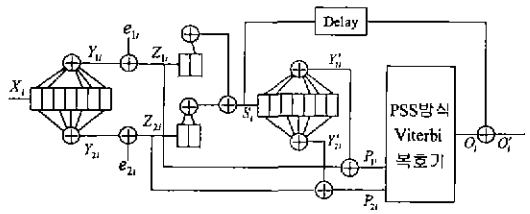
다. PSS 방식의 Viterbi 복호 알고리즘은 복호상태를 절반 가까이 제거 가능하게 함으로써 기존 알고리즘의 단점을 개선하였다. [1]에서, $r=1/2$, $k=7$인 시스템에 22개의 복호상태를 제거한 복호기의 성능이 $E_b/N_0=6$dB 근처에서 기존의 복호기의 성능에 근접한다는 것을 보여준다 그러나, 채널에 가우시안 잡음과 함께 버스트성 잡음이 혼재하는 경우 PSS 방식으로 복호할 경우 성능 저하를 가져오게 된다.

따라서, 본 논문에서는 $r=1/2$, $k=7$인 PSS 방식 Viterbi 복호시스템을 구성하고 MC 모의실험을 이용하여 버스트성 잡음이 성능에 미치는 영향과 인터리빙을 행하였을 때의 성능 및 메모리 복잡도의 향상을 분석하였다. 본 모의실험은 일정시간 동안 임펄스성 잡음이 지속시켜 버스트성 잡음을 모델링 하였으며, [1]에서와 같이 상태를 제거한 복호기를 사용하지 않고, 기존복호기에 PSS 방식을 적용하고 전체 상태들 중에서 사용되지 않은 상태의 개수를 측정하여 PSS 방식 복호시스템의 계산량을 분석하는 방식을 취하였다.

또한 본 논문에서는 Viterbi 복호기의 고속화를 위한 구조를 설계하였으며, 고속화를 위한 각 알고리즘을 적용하여 $r=1/2$, $k=3$인 PSS 방식 Viterbi 복호시스템의 VHDL(Very high speed Hardware Description Language)로 구현한 결과, 약 20Mbps급의 복호기를 구현하였으며, PSS 방식의 목적대로 높은 Eb/No일 때, 복호 상태수가 절반가량이 감소됨을 확인하였다. 향후 본 논문에서 제시한 고속화 알고리즘과 메모리를 줄일 수 있는 알고리즘을 결합하면 Viterbi 칩셋의 소요 면적과 고속화를 요구하는 시스템에 유용하게 적용 가능하리라 사료된다

2. PSS 방식의 Viterbi 복호 알고리즘

PSS 방식 Viterbi 복호시스템의 전체모델은 (그림 1)과 같다.



(그림 1) $r=1/2$, $k=7$인 PSS 방식 Viterbi 복호시스템 모형

송신단에서 정보비트열 X_i는 부호기에서 부호기의 생성다항식

$$C_1(D)=1+D+D^4+D^5+D^6$$
$$C_2(D)=1+D^2+D^3+D^4+D^6 \quad (1)$$

에 의해

$$Y_{1i}=X_i+X_{i-1}+X_{i-4}+X_{i-5}+X_{i-6}$$
$$Y_{2i}=X_i+X_{i-2}+X_{i-3}+X_{i-4}+X_{i-6} \quad (2)$$

와 같이 부호화 비트열이 된다. 채널을 통해 전송된 수신신호는 채널상의 잡음 e_{1i}, e_{2i}가 더해진 형태

$$Z_{1i}=Y_{1i}+e_{1i}=X_i+X_{i-1}+X_{i-4}+X_{i-5}+X_{i-6}+e_{1i}$$
$$Z_{2i}=Y_{2i}+e_{2i}=X_i+X_{i-2}+X_{i-3}+X_{i-4}+X_{i-6}+e_{2i} \quad (3)$$

와 같이 나타낼 수 있다.

여기서, 부호화 효과를 제거하기 위해 Syndrome predecoder를 통과시키면

$$\begin{aligned} S_i &= Z_{1i-1}+Z_{2i}+Z_{2i-1} \\ &= (Y_{1i-1}+e_{1i-1})+(Y_{2i}+e_{2i})+(Y_{2i-1}+e_{2i-1}) \\ &= X_i+e_{2i}+e_{2i-1}+e_{1i}=X_i+e_i' \end{aligned} \quad (4)$$

와 같이 정보신호 X_i와 잡음신호의 조합 $e_i'=e_{2i}+e_{2i-1}+e_{1i}$의 함으로 나타난다. 또한, Predecoder의 출력 S_i를 다시 부호기에 입력하면

$$Y_{1i}'=Y_{1i}+e_i'+e_{i-1}'+e_{i-4}'+e_{i-5}'+e_{i-6}'$$
$$Y_{2i}'=Y_{2i}+e_i'+e_{i-2}'+e_{i-3}'+e_{i-4}'+e_{i-6}' \quad (5)$$

이 되고, 이것을 다시 식 (3)에서의 수신신호 Z_{1i}, Z_{2i}와 XOR 연산하면

$$P_{1i}=e_i'+e_{i-1}'+e_{i-4}'+e_{i-5}'+e_{i-6}'+e_{1i}$$
$$P_{2i}=e_i'+e_{i-2}'+e_{i-3}'+e_{i-4}'+e_{i-6}'+e_{2i} \quad (6)$$

와 같이 정보신호 성분이 사라지고, 잡음성분의 조합만이 Viterbi 복호기에 입력된다.

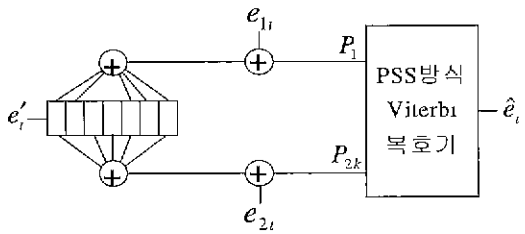
식 (6)은 (그림 2)와 같이 부호기의 입력정보비트가 e_i'이고 채널잡음신호 e_{1i}, e_{2i}인 부호시스템과 등가이다. 따라서 복호기 출력은

$$O_i=\hat{e}_i' \quad (7)$$

이며, O_t와 S_t를 XOR 연산을 하면

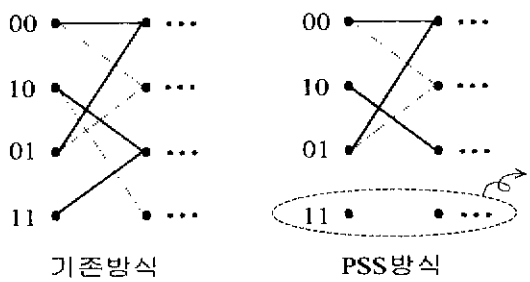
$$O_t = e_t^{'} + X_t + e_t^{'} = X_t \tag{8}$$

으로 정보신호를 복호하게 된다.



(그림 2) PSS 방식 Viterbi 복호기의 등가 부호시스템

(그림 2)에서 PSS 방식 시스템의 복호기입력은 잡음성분이며, 정보신호일 때와는 달리 균일하지 않는 가우시안 분포를 가지게 된다. 이는 신호 대 잡음전력비가 높다면 '1'이 연속되어 복호기에 입력되는 경우가 발생할 확률이 낮아지게 되고, 이 경우에 해당하는 Trellis diagram의 복호상태를 제거하고 복호해도 성능에 큰 영향을 주지 않게 된다는 것이다. (그림 3)에 구속장의 개수 $k=3$인 Trellis diagram의 4개의 상태중 "11"인 상태를 제거하여 복호하는 경우를 예를 보였다.



(그림 3) 기존방식과 PSS 방식의 Trellis diagram ($k=3$)

3. 모의실험

3.1 채널 모델

본 모의실험의 가우시안 잡음은 신호전력을 1로 정규화 했을 때 입력된 E_b/N_o[dB]로부터

$$\sigma = \sqrt{10^{-\frac{E_b/N_o}{10}}} \tag{9}$$

와 같이 계산된 분산을

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right] \tag{10}$$

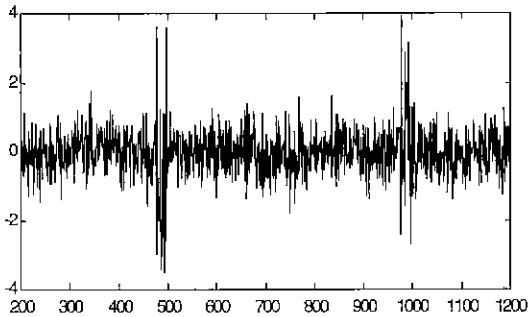
와 같은 가우시안 확률밀도함수에 적용시켜 생성하였다

임펄스성 잡음은 가우시안 잡음에서의 분산에

$$\sigma_I = \sqrt{\frac{\sigma^2}{\Gamma}} \tag{11}$$

와 같이 임펄스성 잡음 대 가우스성 잡음 전력비 Γ만큼 증가시켜 발생시킬 수 있다.

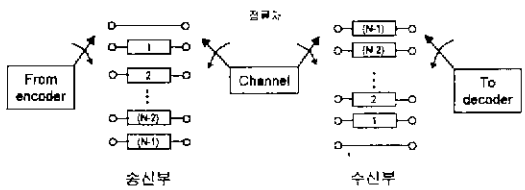
본 모의실험은 $\Gamma=0.1$인 임펄스성 잡음을 매 500개의 입력마다 발생하도록 하였고, 지속시간은 전체발생주기(500)의 0.05배로 하여 모델링 하여 버스트성 잡음효과를 가지게 하였으며, 이는 (그림 4)와 같다



(그림 4) 임펄스성 잡음으로 모델링한 버스트 잡음

3.2 Convolutional 인터리빙

블록 인터리빙과는 달리 연속적인 출력이 가능한 convolutional 인터리빙을 적용하였다. Convolutional 인터리빙은 (그림 5)에서와 같이 송신부와 수신부에 각각 N개 레지스터로 구성된다 송신부의 첫번째 레지스터는 입력을 즉시 전송하고 아래로 갈수록 지연을 증가시켜 마지막 레지스터의 출력은 $(N-1)$만큼 지연된 입력을 출력한다. 수신부의 구조는 송신부와 반대로 구성되어 있다. 송수신부 간의 입력 대 출력의 총 지연은



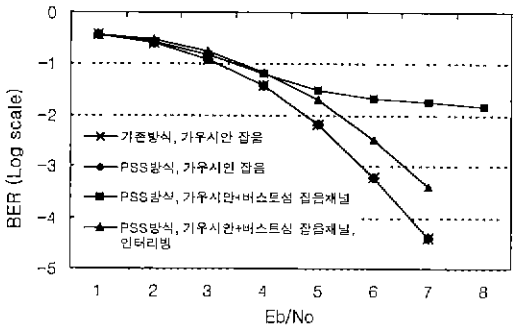
(그림 5) Convolutional Interleaver의 송·수신부 구조

$N(N-1)$이며, 메모리 요구량은 $N(N-1)/2$이 된다

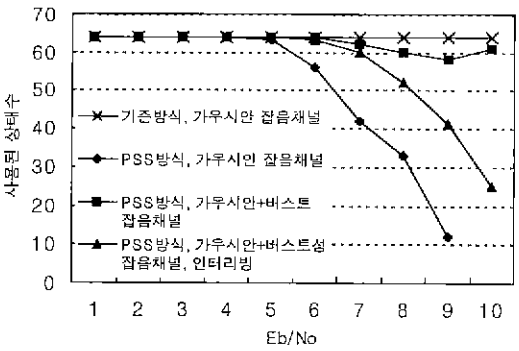
본 모의실험에서는 $N=60$인 convolutional 인터리빙을 사용하였다.

3.3 모의실험 결과 및 분석

(그림 6)은 총 10^5개의 데이터를 가지고 각 채널에 적용한 비트 오류 확률을 나타내었다. 기존방식과 PSS 방식 모두 같은 Viterbi 복호기를 사용하였기 때문에 가우시안 잡음채널에서 동일한 비트오류확률을 가짐을 알 수 있다. PSS 방식에 가우시안 잡음과 버스트성 잡음이 혼재하는 채널을 적용한 경우, 인터리빙을 하지 않은 경우는 비트 오류 확률이 개선되지 않으나, 인터리빙을 하였을 경우 가우시안 잡음 채널의 결과에 근접하고 있음을 알 수 있다.



(그림 6) 각 채널잡음에 대한 비트 오류 확률



(그림 7) 각 채널잡음에서 복호기가 사용한 상태수

(그림 7)에 4×10^3개의 데이터를 가지고 각 채널에 적용하였을 때, 복호기가 사용한 Trellis diagram 상태수를 나타내었다. 기존방식은 모든 경우에서 복호기가 모든 Trellis diagram 상태를 사용한 결과를 보였고, PSS 방식에 가우시안 잡음채널을 적용한 경우, E_b/N_o가 4dB 이상부터 복호기가 사용한 상태수가 감소하여 7dB일 때 상태수를 거의 절반으로 줄일 수 있었다. 가우시안 잡음과 버스트 잡음이 혼재하는 채널을 적용하고 인터리빙을 사용한 경우는 9dB에서 상태수를 절반으로 줄일 수 있음을 알 수 있다.

Viterbi 복호기에 요구되는 메모리는

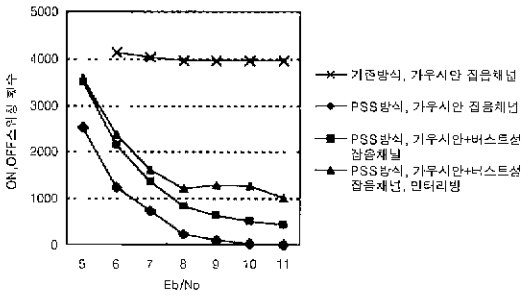
$$M = h\times 2^{k-1} \tag{12}$$

와 같이 나타낼 수 있으며 h는 복호기의 Decoding depth로서, 일반적으로 구속장 k의 4배 혹은 5배이다. (그림 7)의 결과를 식 (12)에 적용하여 메모리 요구량을 <표 1>에 나타내었다.

〈표 1〉 Eb/No에 따른 메모리 요구량 비교

방식 / Eb/No	기존방식	PSS 방식		
		가우시안 잡음채널	가우시안+버스트성 잡음채널	가우시안+버스트성 잡음채널, 인터리빙
4	$h\times64$	$h\times64$	$h\times64$	$h\times64$
5	$h\times64$	$h\times56$	$h\times64$	$h\times63$
6	$h\times64$	$h\times42$	$h\times62$	$h\times60$
7	$h\times64$	$h\times33$	$h\times60$	$h\times52$
8	$h\times64$	$h\times12$	$h\times58$	$h\times41$

(그림 8)에 4×10^3개의 데이터를 가지고 각 채널을 적용하였을 때, 복호기의 전력소모량을 나타내는 ON, OFF 스위칭 횟수를 보였다. 기존방식의 경우 전력소모량은 거의 일정하였으며, PSS 방식은 가우시안 잡음채널에서 가장 전력소모량이 적었고, 비스트성 잡음이 혼재하는 채널에서는 인터리빙을 하였을 경우 전력소모량이 감소함을 알 수 있다.

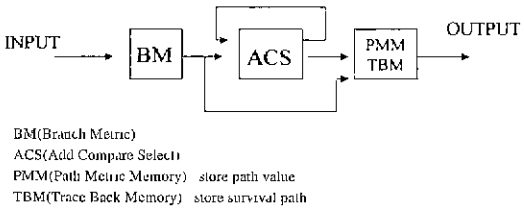


(그림 8) 각 채널잡음에서 복호기의 ON, OFF 스위칭 횟수

4. 고속화 구조 설계 및 VHDL 모델링

4.1 고속화 구조 설계

Viterbi 복호기는 (그림 9)와 같이 크게 3부분으로 나눌 수 있다 BM에서 입력신호와 Trellis diagram의 Branch word와의 비트차 즉, 헤밍거리를 구하고 ACS에서 과거 헤밍거리와 더한 다음, 가장 적은 헤밍거리 누적치를 갖는 생존경로를 선택한다. 이렇게 선택된 생존경로들을 PMM에 저장하고 TBM에서 PMM에 저장된 생존경로를 역추적하여 복호한다.



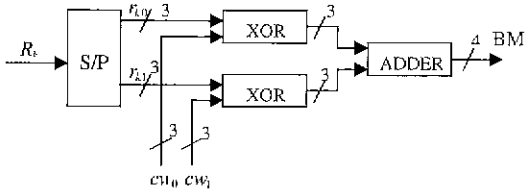
(그림 9) Viterbi 복호기의 구조

4.1.1 BM unit

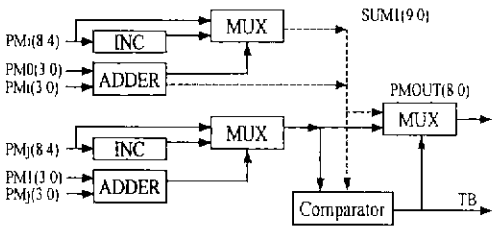
Viterbi 복호기의 트렐리스상의 부호화 비트인 BCW (Branch Coded Word) 값과 Viterbi 복호기의 입력 비트인 연판정된 복조기 출력 신호 $R_k=(r_{k1}, r_{k0})$를 XOR 함으로써, 수신신호와 부호화 비트간의 해밍거리를 구하는 부분이다 (그림 10)은 BM unit를 나타내고 있다. 수신신호가 3비트 연성판정 되었으므로 수신신호와 BCW(Branch Coded Word)와의 해밍거리를 구하기 위하여 cw_0 및 cw_1을 각각 3비트로 표현하여 BCW를 ("000", "000"), ("000", "111"), ("111", "000"), ("111", "111") 중의 하나가 된다. 예를 들어 수신신호를 ("100", "101")이라 하고 현재 BM을 계산하는 가지의 BCW가 ("111", "000")이라 하면 해밍거리는 $|7-4|+|0-5|=9$, "1001"이 된다. 즉, cw_i가 "000"일 때는 가산기 입력 A_i를 r_{ki}로 하고 cw_i가 "111"일 때는 가산기 입력 A_i를 not(r_{ki})로 취하여 가산한다 이 때 가산기의 출력이 BM값에 해당한다. BMU의 출력은 0~14 사이의 값을 가지므로 4비트로 표현할 수 있다.

4.1.2 ACS unit

ACS unit는 Viterbi 복호기가 처리할 수 있는 최대 데이터 속도를 결정하는 가장 중요한 블럭이다 복조기의 출력신호는 3비트 soft decision되어 복호기에 입력되는데, BM unit의 출력(PM0, PM1)과 과거 누적된 PM(Path Metric)값인 PMi를 합하여 각 상태 노드에 있는 두 경로의 PM값을 비교하여 PM값이 작은 경로를 선택하여, 선택된 PM의 경로정보를 trace back 복호부에 정보를 준다. ACSU의 구조는 (그림 11)과 같다.



(그림 10) BMU 구조



(그림 11) ACSU의 구조

현재의 BM값과 과거 누적된 PM값을 더하는 과정을 단순한 전가산기를 이용하면 carry 발생으로 인한 시간 지연이 있으므로 본 논문에서는 현재의 PM값을 구하기 위해 다음과 같은 step으로 구현한다.

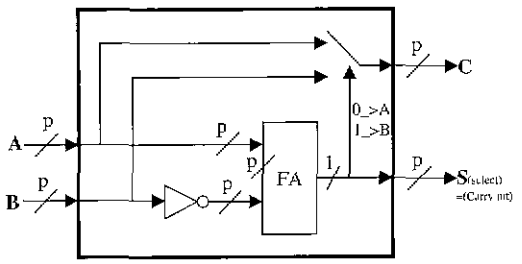
Step1. PM 비트를 5비트와 4비트로 분할한다

Step2. 하위 4비트와 BM 4비트를 더하는 동시에 상위 5비트를 1씩 증가시킨다.

Step3. 하위 4비트 합의 결과가 carry 발생하면 1씩 증가한 상위 5비트와 멀티플렉싱(multiplexing) 시킨다. carry가 발생하지 않으면 1씩 증가시키지 않은 원래의 상위 5비트와 멀티플렉싱한다.

Step1~Step3에서 알 수 있듯이 9비트의 PM을 4비트와 5비트로 분할하여 동시에 계산하면 9비트 덧셈보다 고속화로 구현 할 수 있다. 각 노드에서 최소 PM을 선택하는 comparator는 일반적으로 MUX와 뺄셈으로 구성되는데 이는 많은 시간을 요구한다. 따라서 고속화를 위해 Metric rescaling 방식으로 comparator를 구성하였다. comparator의 구조는 (그림 12)와 같다. Metric rescaling 방식은 뺄셈 대신에 1의 보수를 취하여 더하

는 방식이데 더한 결과의 MSB(Most Significanr Bit)가 경로정보인 동시에 MUX의 select단자의 입력이다. 경로정보는 ACSU로 입력되는 각 노드의 두개의 상태 중에서 위에서 입력되는 상태를 0, 아래서 입력되는 상태를 1로 선택하여 TBU로 보낸다.



p : 임의의 비트수

(그림 12) 비교기 구조

4.1.3 PM unit

다음 상태의 ACS 동작을 위해 현 상태까지의 누적된 해밍거리값을 나타내며 이는 PMM(PM Memory)에 저장한다. PM을 구현하는데 가장 큰 문제점은 한정되어 있는 PM 비트로 인하여 발생하는 overflow를 어떻게 처리하느냐에 있다. 여기서는 고속화를 위하여 Modulo Arithmetic 방법을 사용한다 Modulo Arithmetic 방법은 overflow를 무시하고 계속 더하는 방법으로 ACSU에 입력되는 각 상태노드에서의 두 PM값간의 차이만 비교할 수 있다면 *Viterbi* 알고리즘은 올바른 동작을 할 수 있고, 또한 임의의 시간에서 두 PM값의 차이에는 최대값이 존재하고 더 이상 커지지 않는다는 성질을 이용한 것이다. 이러한 성질을 이용하기 위해서는 PM의 비트수인 WL(Word Length) 정확히 규정하여야 한다.

$$WL \geq \lfloor \log_2(2nB) \rfloor + 1 \qquad (13)$$

여기서 $n = k-1$(k는 구속장 수), B는 BM의 상한치이다 따라서 본 논문에서는 PM에 할당된 비트 수를 $WL = \lfloor \log_2(2\times2\times14) \rfloor + 1 = 6$ bits로 할당하였다.

4.1 4 TB unit

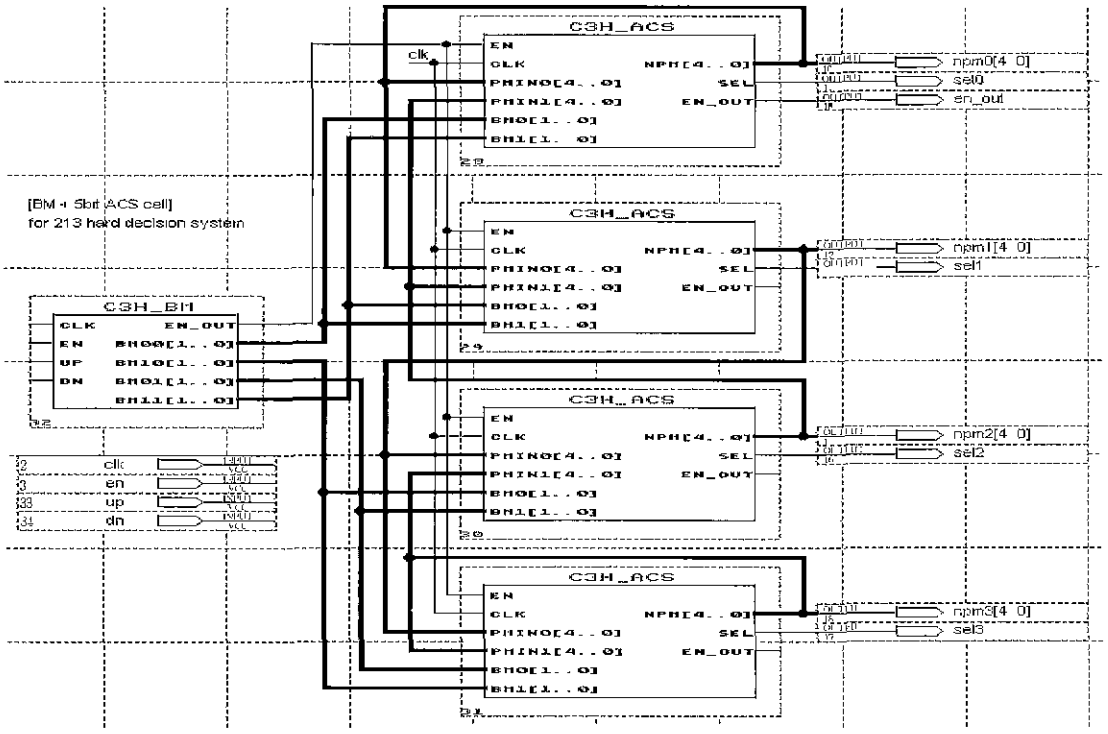
ACS에 의하여 결정되어진 경로정보가 저장된 메모리(TBM) 이용하여 복호하는 부분이다. TBM의 경로정보를 이용하여 복호하는 방식은 크게 REM(Register Exchange Method) 방식과 TB 방식으로 구분할 수 있는데, REM 방식은 VLSI로 구현할 때 쓰기 동작이 매우 복잡하고 전력소모가 크다는 단점이 있어, 대부분의 경우 TB 방식을 사용한다. TB 동작은 식 (14)를 이용하여 역방향으로 수행되며, 이 때 식 (14)에 의하여 송신비트를 복호해 낸다. 복호비트의 결정은 선택된 생존경로의 상태변화로 할 수 있다 예를 들어 (그림 3)과 같이 구속장 $k=3$인 Trellis diagram의 4개의 상태 00, 10, 01, 11의 입력을 살펴보면 00, 01로 들어오는 선은 실선 즉, 부호가 입력이 0일 때이고, 10, 11로 들어오는 선은 점선, 입력이 1인 것을 알 수 있다. 만약, 생존경로의 역추적 결과 상태변화가 $01 \rightarrow 11 \rightarrow 10 \rightarrow 00$이었다면, 복호비트는 상태의 상위비트가 되므로 0, 1, 1, 0이 된다 따라서, 본 VHDL 모의실험의 최종복호기 출력은 식 (14)와 같이 생존경로의 역추적 과정에서 발생하는 상태천이로 하였다.

$$\begin{aligned} X_{k-1}(1) &= X_k(0),\ X_{k-1}(0) = flag \\ d_k &= X_k(1) \end{aligned} \qquad (14)$$

여기서 $X_k(0)$, $X_k(1)$은 각각 t=k인 시점에서 노드 상태의 LSB(Least Significant Bit) 및 MSB(Most Significant Bit)를 의미하고 flag는 ACS에서 TBM에 저장한 경로정보이다 d_k는 복호된 비트로 상태정보의 MSB가 복호되는 비트이다 이러한 TB 동작을 구현하기 위해서는 복호시 TBM을 여러 개의 bank로 나누어 쓰는 동안 각 bank에서 병렬로 읽기 때문에 쓰기클럭과 동시에 읽기클럭이 수행되므로, 고속처리가 가능한 알고리즘인 k-pointer TB 알고리즘이 유용하다[5]. 본 논문에서는 k = 3 Point 알고리즘을 적용하여 총 6개의 bank로 구분하여 읽기클럭과 쓰기클럭에 대한 요구속도가 동일하므로 복호과정을 고속화 시켰다.

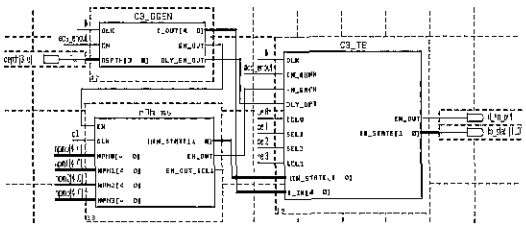
4.2 VHDL 구현

(그림 13)에 BM 유닛과 ACS가 $k=3$인 Trellis diagram의 구조에 따라 매치된 VHDL 도면을 보였고, (그림 14)에 PMM과 연결된 TB 유닛의 도면을 보였다. $k=7$인 Viterbi 복호기는 64개의 상태를 가지므로 구현 결과를 나타내기에는 매우 복잡하여 본 논문에서는 4개의 상태를 가지는 $k=3$인 경우의 예를 들었으며, 복호기 입력 비트는 복호가 정상적으로 동작하는지에 대한 검증하는 단계에서 잡음이 없는 비트를 입력

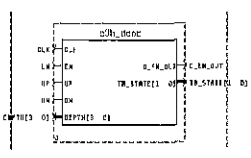


(그림 13) BM 유닛과 ACS 도면

하였다. (그림 13), (그림 14)의 두 도면을 결합하여 (그림 15)와 같이 하나의 Viterbi 복호기 모듈을 만들고, (그림 1)에서와 같이 부호기와 Syndrome Predecoder 등을 첨가하여 (그림 16)에 PSS 방식 Viterbi 복호시스템의 전체 도면을 구성하였다



(그림 14) PMM과 TB유닛



(그림 15) Viterbi 복호기 모듈

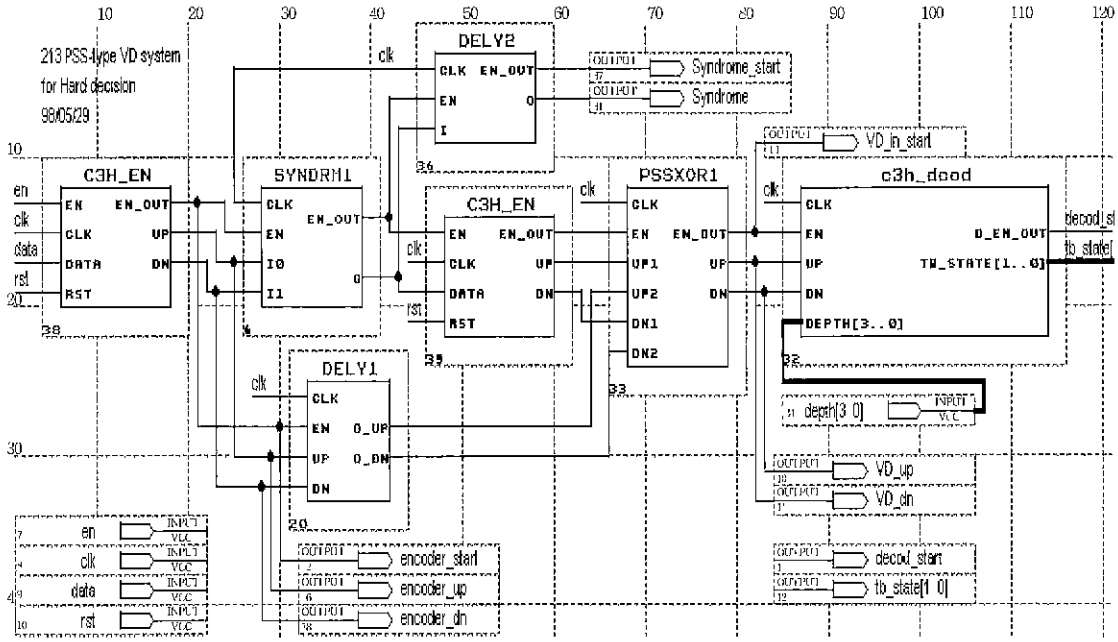
(그림 1)과 비교하여 (그림 16)의 각 출력포트는 <표 2>와 같이 대응된다.

<표 2> PSS 방식 블록도에 대응되는 블록에 대응하는 VHDL 출력 ports

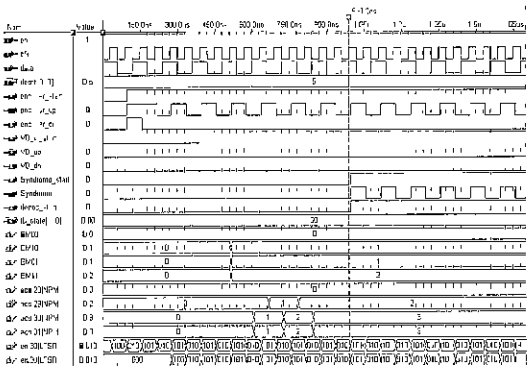
	PSS 방식 블록도 (그림 1)	VHDL 출력 ports (그림 16)
송신부 부호기출력	Y_{1i} , Y_{2i}	Encoder_up, encoder_dn
수신신호	Z_{1i} , Z_{2i}	〃
Syndrome Predecoder 출력	S_i	Syndrome
Viterbi 복호기 입력	P_{1i} , P_{2i}	VD_up, VD_dn
Viterbi 복호기 출력	O_i	Tb_state[1]

VHDL 모의실험은 실제 동작여부를 확인하는 차원에서 채널상의 잡음을 고려하지 않고 출력한 것을 (그림 17)에 보였다 클럭 주기는 60ns이며, 입력에 대한 출력 지연은 약 $1\mu s$ 였다.

Viterbi 복호기에 입력되는 신호를 나타내는 'VD_up', 'VD_dn' 포트를 보면, 잡음을 넣지 않아 항상 Low인 것을 알 수 있으며, Viterbi 복호기의 출력 즉, 생존경로의



(그림 16) VHDL로 구현한 PSS 방식 Viterbi 복호 시스템의 전체 도면



(그림 17) VHDL 모의실험 결과

상태천이를 나타내는 'tb_state[1..0]' 출력포트 역시 00의 상태를 유지하였다. 그리고, 복호기 출력시점을 나타내는 'decod_start' 포트가 High로 변하는 시점과 맞추어 Syndrome Predecoder의 출력 'syndrome' 포트가 'data' 포트와 동일한 파형을 출력하고 있다 따라서, (그림 1)에 의해 전체출력 O_i'는 복호기 출력 'tb_state[1]'과 'syndrome' 포트의 신호와 XOR 연산을 행하여 'data' 포트의 신호가 출력됨을 알 수 있다.

5. 맺음말

본 논문에서는 $r=1/2$, $k=7$인 PSS 방식 Viterbi 복호 알고리즘의 성능, 메모리 요구량 및 전력 소모량을 가우시안 잡음채널 및 버스트성 잡음채널에서 비교하였으며, 인터리빙을 행하였을 때 이느 정도 향상이 있는지 분석하였다 그리고, $r=1/2$, $k=3$인 시스템을 VHDL tool을 이용하여 구성하고 기본적인 출력을 확인하였다.

가우시안 및 버스트성 잡음이 혼재하는 채널에서 예상했던 바와 같이 비트오류 확률이 낮아지고, 사용된 복호상태의 개수가 증가하여 재산량이 증가하였다. 같은 환경의 시스템에 $N=60$ 단의 레지스터를 갖는 convolutional 인터리빙을 행하였을 경우, 사용된 복호상태 수 및 비트오류확률이 가우시안잡음 채널만 존재한 결과에 근접하는 성능향상을 얻을 수 있었다 사용된 복호상태수는 가우시안잡음 채널에서의 결과에 약 2dB의 E_b/N_0 차이가 있었고, 비트오류확률은 약 1dB의 차이를 보이다.

또한 본 논문에서는 Viterbi 복호기의 고속화를 위한 구조를 설계하였으며, 고속화를 위한 각 알고리즘을

적용하여 $r=1/2$, $k=3$인 PSS 방식 Viterbi 복호 시스템의 VHDL 모델링 한 결과, 약 20Mbps급의 복호기를 구현하였으며, 복호 상태수가 높은 Eb/No일 때, PSS 방식의 목적대로 절반가량이 감소됨을 확인하였다. 향후 본 논문에서 제시한 고속화 알고리즘과 메모리를 줄일 수 있는 알고리즘을 결합하면 Viterbi 칩셋의 소요 면적과 고속화를 요구하는 시스템에 유용하게 적용 가능하리라 사료된다.

참 고 문 헌

[1] SUN PING, YAO YAN AND CHONGXI FENG, "An Effective Simplifying Scheme for Viterbi Decoder." IEEE Trans. Commun., Vol.39, No.1, January 1991.

[2] S. Kubota, K. Ohtani, and S. Kato. "High-Speed and High-Coding-Gain Viterbi Decoder with Low Power Consumption Employing SST scheme," Electron. Lett., Vol.22, No.9, pp.491, Apr 1986.

[3] Coates, "Monte Calro Simulation" IEEE Journal on Select Vol.6, pp.58-66, 1988.

[4] Sklar, B. *Digital Communications : Fundamentals and Applications*, Prentice Hall, pp.333-428, 1988.

[5] T.K. Truong, Ming-Tang Shih and E.H. Satorius, "A VLSI Design for a Trace-Back Viterbi Decoder" IEEE Trans. Commun., Vol.40, No.30, March 1992.

[6] TSUNWHACHI, "A Scarce-State-Transition Viterbi-Decoder VLSI for Bit Error Correction," IEEE Journal of Solid State Circuit, Vol.sc-22, No.4, AUG., pp.575-581, 1987.

[7] 한국전자통신연구원, *155Mbps급 위성 ATM 전송 변복조 핵심기술 개발*, 연구보고서, 1997.

[8] 한국전자통신연구원, *B-WLL용 오류정정 부복호화기 구조설계 및 성능분석*, 연구보고서, 1999.

[9] F. J. MacWilliams, and N. J. Sloane, *The Theory of Error-Correction Codes, North Holland*, New York, 1997.

[10] 정지원외 4인, "20Mbps급의 64 state Viterbi 복호기 구조 설계 및 CPLD 구현", 한국통신학회 하계 학술논문집, pp.843-846, 1999

양 형 규

e-mail : hkyang@kns.kangnam.ac.kr

1983년 성균관대학교 전자공학과 (학사)

1995년 성균관대학교 전자공학과 (석사)

1994년 성균관대학교 정보공학과 (박사)

1982년~1990년 삼성전자 컴퓨터 부문 선임연구원

1995년~현재 강남대학교 이공대학 산진전공학부 전자계산전공 조교수

관심분야 : 암호 이론 및 응용, 네트워크 보안, 전자화폐, 정보 은닉

정 지 원

e-mail : jwjung@hanara.kmaritime.ac.kr

1989년 성균관대학교 전자공학 (학사)

1991년 성균관대학교 전자공학 (석사)

1995년 성균관대학교 정보공학 (박사)

1991년~1992년 LG 정보통신연구소 연구원

1995년~1996년 한국통신 위성통신연구실 선임연구원

1997년~1998년 한국전자통신연구원 초빙 연구원

1996년~현재 한국해양대학교 전파공학과 조교수

관심분야 : 위성통신, 이동통신, 변·복조기술, 채널코딩, FPGA 기술