고속 전송을 위한 비터비 디코더 설계

학생회원 김 태 진*, 정회원 이 찬 호**

DESIGN OF A HIGH-THROUGHPUT VITERBI DECODER

Tae-Jin Kim*, Chanho Lee** *Regular Members*

요 약

본 논문에서는 trace-back 동작 없이 디코딩이 가능한 변형된 레지스터 교환(MRE) 방식을 블록 디코딩에 적용하여 전송 속도를 높이고 latency를 줄이는 비터비 디코딩 방식을 제안하였다. 변형된 레지스터 교환 방식을 블록 디코딩에 적용함으로써 디코딩 블록의 시작 상태를 결정하기 위해 필요한 동작 사이클을 줄여, 블록 디코딩을 사용하는 기존의 비터비 디코더보다 더 적은 latency를 가지게 되었다. 뿐만 아니라, 메모리를 더 효율적으로 사용할 수 있으면서 하드웨어의 구현에 있어서도 복잡도가 더 감소하게 된다. 또한 시작 상태를 결정하기 위해 필요한 trace-back 동작을 없애고 메모리를 줄여 이에 따른 전력 소모를 줄이는 저전력 동작이 가능하다. 제안된 방식은 같은 하드웨어 복잡도로도 메모리의 감소 또는 latency의 감소에 중점을 둔 설계가 가능하다. 또한, 몇 가지 디자인 파라미터를 변경하여 합성 단계에서 하드웨어 복잡도와 전송 속도를 trade-off 할 수 있도록 스케일러블한 구조로 설계하였다.

Key Words : Viterbi decoder, MRE, block decoding, error correction code.

ABSTRACT

A high performance Viterbi decoder is designed using modified register exchange scheme and block decoding method. The elimination of the trace-back operation reduces the operation cycles to determine the merging state and the amount of memory. The Viterbi decoder has low latency, efficient memory organization, and low hardware complexity compared with other Viterbi decoding methods in block decoding architectures. The elimination of trace-back also reduces the power consumption for finding the merging state and the access to the memory. The proposed decoder can be designed with emphasis on either efficient memory or low latency. Also, it has a scalable structure so that the complexity of the hardware and the throughput are adjusted by changing a few design parameters before synthesis.

Ⅰ. 서 론

요즘 시대에는 보다 효율적이고 신뢰도가 높은 디지털 데이터의 전송과 저장이 꾸준히 요구되고 있다. 이러한 요구를 만족시키기 위하여 데이터 전송 과정에서 발생하는 에러를 최소화 시키고자 하는 노력이 계속 되어 왔으며 그에 따라 채널 코딩은 통신 시스템에서 중요한 이슈가 되어 왔다. 채널 코딩은 데이터 전송에 있어서, 채널에서 발생되는 잡음에 의한 오류를 수신측이 검출 혹은 정정할 수 있도록 원래의 데이터에 추가로 리던던시(redundancy)를 덧붙이는 방법이다. 이때 신호 파워 증가 없

* 숭실대학교 전자공학과 (xowls@engineer.ssu.ac.kr), ** 숭실대학교 전자공학과 (chlee@ssu.ac.kr)
논문번호 : KICS2004-12-303, 접수일자 : 2004년 12월 6일
※ 본 연구는 숭실대학교 교내연구비 지원으로 수행되었습니다.

이 비트 에러율 (BER)을 줄일 수 있다. 이러한 코딩 기술은 GSTN(General Switched Telephone networks)과 같은 유한 파워 채널의 전송에 유용하다. 길쌈 부호(convolutional code)는 잡음 채널 환경에서 블록 디코딩 방법과 접목되어 사용되어져 왔다. 길쌈 부호는 현재 비트와 과거 비트 사이의 관계를 이용하여 부호화하는 방식이다.

비터비 복호 알고리즘은 길쌈 부호의 대표적인 디코딩 방법이며, 적은 에러율의 데이터 전송이 가능하여 통신과 신호 처리에 널리 쓰이고 있다. 비터비 복호 방법은 MLD (Maximum Likelihood Decoding) 알고리즘을 이용한다. MLD 알고리즘은 수신된 데이터로부터 확률적으로 가장 가까운 패턴을 찾는 방법이고, 가장 최적의 복호 방법으로 알려져 있다[1]. 무선랜 규격인 802.11a와 DMB (Digital Multimedia Broadcasting)에서는 K=7, R=1/2인 비터비 디코더가 사용된다.

전송속도를 증가시키기 위한 방법으로 블록 디코딩 방식이 많이 이용된다. 비터비 디코더에 사용되는 일반적인 블록 디코딩 방식으로 k-pointer 방식[2], one pointer 방식[2], hybrid trace-back 방식[2], Look-Ahead trace-back 방식[3] 등이 있다. Look-ahead trace-back방식을 제외한 나머지 방식들은 디코딩 과정을 시작하기 위한 시작 상태를 찾기 위해 머징(merging) 과정을 가지게 된다. 머징 과정에서는 머징 블록들에서 survival path를 따라서 trace-back 과정을 수행하면서 진행된 후 시작 상태를 찾게 되며, 디코딩 과정에서는 디코딩 블록에 대해 시작 상태에서부터 trace-back이 진행되어[4] 디코딩된 bit들이 역순으로 출력이 된다. One pointer 방식과 hybrid trace-back[2] 방식은 k-pointer 방식보다 메모리를 보다 효율적으로 이용한다는 장점이 있다. 그러나 그 두 경우, trace-back과 디코딩 동작이 ACS (Add-Compare-Select) 연산보다 최소 2-3배 더 빠르게 수행 되어야 한다[4].

그런데, 만약 머징 과정에서의 trace-back과정이 없어진다면 trace-back을 하면서 생겼던 latency가 그만큼 줄어들 수 있을 것이다. 이는 머징 블록들에서 생존 경로(survival path)가 결정되자마자 시작 상태가 바로 결정이 된다면 가능하게 된다.

본 논문에서는 변형된 레지스터 교환 (MRE) 방식[5]을 이용한 고성능의 비터비 디코더를 제안하였다. 시작상태는 MRE 방식을 사용해서 머징 과정에서의 trace-back 과정을 제거함으로써 생존 경로가 결정되자마자 얻어지며, 따라서 디코딩된 데이터의 출력도 trace-back을 없앤 만큼의 latency를 줄인 효과를 보게 된다. 또한 trace-back 과정이 없어지면서 필요한 메모리의 양도 감소한다. 이러한 과정은 하드웨어 복잡도의 증가나 많은 메모리의 사용이 없이도 가능하다. 또한, trace-back 동작을 없앰으로써 그 동작에 필요한 연산과 메모리 접근을 없애 전력 소모를 줄일 수 있다. 한편 Verilog HDL을 이용하여 몇 가지 디자인 파라미터를 변경하여 합성 단계에서 하드웨어 복잡도와 전송속도를 trade-off 할 수 있도록 스케일러블한 구조로 설계하였으며, 구속장 (constraint length)과 code rate도 디자인 파라미터로 조정 가능하도록 설계하였다.

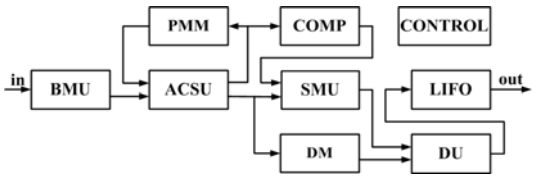
Ⅱ. Modified Register Exchange 방식을 적용한 Block Decoding 방식

MRE 방식은[5] trace-forward를 하는 과정에서 decision bit의 위치를 변경시켜 저장하여 trace-back의 필요성을 없애고 이를 위한 중간 값을 저장하지 않아도 trace-forward가 끝난 뒤 시작 상태를 즉시 알 수 있고 그에 따른 메모리도 줄일 수 있는 방식이다. 그러나 여전히 MRE 방식만으로는 충분한 디코딩 속도를 얻기 어려우므로 한번의 trace-forward에서 여러 bit를 한번에 디코딩할 수 있는 블록 디코딩 방식을 이용하여 디코딩 속도를 높일 필요가 있다. 그러나 블록 디코딩을 이용하여 한 클럭에 한 번씩 디코딩 bit을 얻기 위해서는 trace-forward 과정에서 얻은 모든 decision bit이 디코딩에 이용되므로 중간 값을 버리지 못하고 저장해야 한다. 따라서 블록 디코딩 방식을 이용할 때보다는 메모리의 크기를 줄일 수 있다는 이점은 크지 않다. 그러나 MRE 연산 유닛을 이용하여 여전히 디코딩을 위한 머징 과정 후 시작상태를 trace-back 과정 없이 얻을 수 있고 이에 따라 디코딩 블록의 크기를 좀더 세분하여 정할 수 있으므로 전체적으로 메모리 크기를 줄일 수 있다. 이에 따라 기존의 방식보다 적은 메모리를 사용하며 전력 소모도 더 줄일 수 있다.

그림 1은 MRE 방식을 이용한 비터비 디코더의 블록 다이어그램이다. 블록 디코딩 방식에 필요한 정보들은 decoding 과정 중간에 선택되어져 decision 메모리 (DM)에 저장된다. Decision 메모리는 그림 2와 같이 몇 개의 블록들로 구성된다. T는 생존 경로 길이(survival path depth)와 같고 구속장 (K)이 7일 경우에 T는 36이다. 첫 번째 기록 과정 (writing process) 동안에는 k-pointer 방법에서처럼

선택된 비트들이 decision 메모리 블록 1에 저장된다. 두 번째 기록 과정부터는 첫 번째 메모리 블록의 시작 상태(또는 머징 상태)를 찾기 위해서 MRE 방식이 사용된다. 즉, 두 번째 블록에서 첫 번째 단계의 시작 주소들이 상태(state) 메모리에 저장이 되고, 이 값들은 첫 번째 MRE 방식(M1)에 사용된다. 블록 디코딩에 MRE방식이 적용될 때에는 기존의 MRE 방식과는 약간 다르게 첫 번째 stage의 decision bit들은 사용하지 않고 상태 주소(state address)만 사용하므로 상태 메모리에 저장할 필요가 없다. 세 번째 기록 과정이 시작되면 M1과 M2의 MRE 과정이 동시에 진행되고 첫 번째와 두 번째 상태 메모리에 있는 내용은 같은 방식으로 재정렬된다. 블록1의 디코딩 과정을 위한 블록2의 시작 상태는 M1의 과정이 블록5까지(생존 경로 길이 T만큼) 진행되어서 마치게 되면 결정이 된다(그림 2에서 <1>로 표시된 부분). 시작 상태가 결정이 되면, decision 메모리 블록 1에 대해 시작 상태를 시작점으로 해서 trace-back을 진행하며 디코딩 bit을 얻게 된다. 이때 결정된 디코딩 bit은 LIFO(Last-In First-Out) 메모리에 저장된다. Decision 메모리 블록1의 디코딩 과정이 끝난 후에는 블록2에 대한 시작 상태가 결정이 되고(그림 2의 <2>), 블록2에 대한 디코딩 과정이 계속해서 진행이 된다. 이와 동시에 LIFO 메모리에 있던 디코딩 데이터는 출력으로 나오게 된다. 이때 최종 출력이 나올 때까지 소요되는 latency는 3T/2 = 3x36/2 = 48 클럭(구속장 K=7, T=36 인 경우)이 되고, 이 이후로는 매 클럭마다 디코딩 bit가 나오게 된다.

일반적으로, decision 메모리가 m개의 블록들로 나누어지면 그 때 각 블록의 크기는 T/(m-1)가 되며, MRE 과정은 m-1개의 블록에 걸쳐서(생존 경로 길이 T 만큼) 진행된다. 각 메모리 블록의 크기가 3-pointer 방식의 메모리 블록 크기의 절반이기 때문에 총 메모리 크기는 3-pointer 방식보다 상당히 줄어든다.



BMU: branch metric unit,
ACSU: add-compare-select unit,
PMM: path metric memory, COMP: comparator,
SMU: state metric unit, DM: decision memory
LIFO: last-in last-out memory, DU: decision unit
그림 1. MRE 방식을 이용한 비터비 디코더의 블록 다이어그램

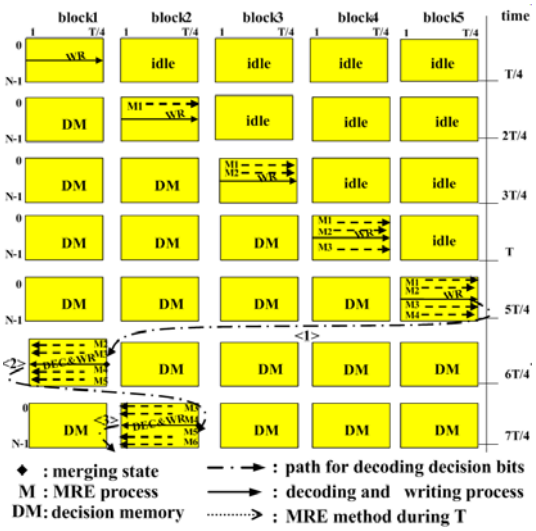


그림 2. MRE를 이용한 5 메모리 블록 디코딩 방식

Ⅲ. 설계와 구현

MRE 방식을 이용하여 블록을 하게 되면 머징 과정에서 trace-back을 사용하는 다른 방식에 비해서 더 적은 메모리 블록을 사용하게 된다. 그림 3은 MRE를 사용함으로써 메모리 블록의 수가 감소하고 latency가 줄어드는 것을 보여준다. k-pointer 방식의 경우[2] 머징 상태를 구하기 위해 trace-back을 진행하는 동안 계속 기록 과정을 진행하지 않으면 블록 3에 대해 디코딩을 진행한 후 디코딩 데이터의 출력이 잠시 중단된다. 따라서 블록 4,5가 필요하다. 그러나 MRE 방식을 이용하면 그럴 필요가 없으므로 블록 4,5가 필요 없다. 따라서 3개의 메모리 블록으로 충분하다. 이 때 메모리 블록 하나의 크기를 줄이고 메모리 블록의 수를 늘린다면 전체 메모리 크기가 조금 증가하는 대신 출력 latency를 줄일 수 있다. 본 연구에서는 latency를 줄이기 위해 5개의 메모리 블록을 이용하였다.

표 1은 구속장 K=7, Code Rate R=1/2, 생존 경로 길이 T=36인 경우 여러 블록 디코딩 방식과의 비교를 나타낸 것이다. 3-pointer 방식(even, odd)[2], one-pointer방식[2], hybrid even 방식[2], look-ahead 방식[3]을 5개의 메모리 블록을 가지고 MRE를 이용하는 방식과 비교하였다. 앞의 세 가지 구조들은 머징 상태를 결정하기 위하여 trace-back 과정을 필요

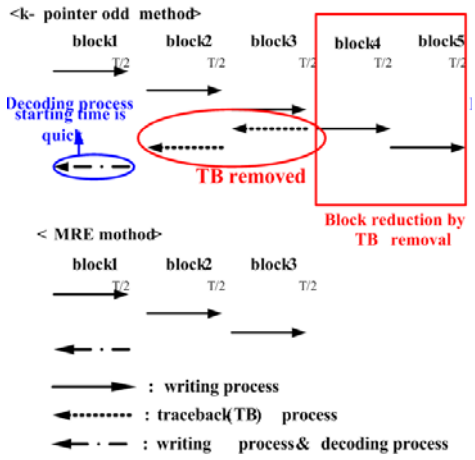


그림 3. MRE 방식과 3-pointer 방식의 차이점

표 1. 특성 비교: 메모리 크기, latency, 하드웨어 복잡도 (K=7, T=36)

	(1)	(2)	(3)	(3)	(5)
DM [bits]	6,912	4,608	4,608	2,304	2,880
PMM [bits]	640	640	640	23,040	640
SMM [bits]					1,536
LIFO [bits]	36	36	36	N/A	18
Total [bits]	7,588 (100%)	5,280 (69%)	5,280 (69%)	25,344 (334%)	5,074 (67%)
Latency	3T	2T	2T	T	3T/2
# of ACSU	64	64	64	64	64
# of TBU	2	N/A	1**	N/A	0
# of DU	1	1*	1**	N/A	1

(1) 3-pointer even, (2) One-pointer (kl=3)
(3) Hybrid even, (4) Lookahead trace-back
(5) 제안된 구조 (5 block),
* 동작 속도가 다른 unit의 2배
** 동작 속도가 다른 unit의 3배

로 한다. 또한 앞의 세 가지 방식은 latency를 줄이기 위해 하나의 메모리 블록의 크기를 줄이고 메모리 블록의 수를 늘릴 경우 메모리 블록을 한개 이상 필요로 하지만 MRE 구조는 메모리 블록 크기에 상관없이 오직 한 개의 메모리 블록만을 더 필요로 한다. 따라서 메모리 블록의 크기가 더 작아질수록, MRE 방식과 look-ahead trace-back 방식을 제외한 다른 방식들은 더 많은 메모리 블록과 더 많은 TBU(trace-back unit), 그리고 DU(decision unit)이 필요하게 되므로 이는 하드웨어 복잡도와 전력소모에 있어서 더 나쁜 결과를 가져온다. 예를 들어, 5-pointer odd 방식은 9x64bits 크기의 메모리 블록 9개와 5개의 TBU, 그리고 1개의 DU가 필요하다. 그러나 MRE를 사용하는 방식은 9x64bits 크기의 메모리 블록 5개와 오직 1개의 DU만이 필요하다. 뿐만 아니라, 디코딩 블록의 시작 상태가 trace-forward 과정이 끝나자마자 결정되므로 디코딩 bit가 출력될 때까지의 latency가 더 줄어들게 된다. K-pointer 방식은 메모리 블록의 수가 증가하게 되면 메모리와 latency가 줄어들지만, 포인터의 수가 늘어나므로 TBU가 증가하게 되는 단점이 있고, MRE를 사용하는 방식은 TBU의 증가 없이 latency를 줄이지만 state 메모리의 수가 늘어나므로 전체 메모리 수가 좀 더 증가된다. One pointer방식과 hybrid trace-back 방식은 TBU의 동작속도가 거의 메모리 블록의 수만큼 높여줘야 한다는 점에서 MRE방식과 큰 차이가 난다.

표 1에서 lookahead trace-back 방식[3]은 trace-back 과정이 필요 없으며 비교된 방식들 중에서 가장 적은 latency T를 가지지만, trace-forward 과정에서 생존 경로 길이(T)동안 매 단계에서 모든 PM 값을 저장하기 때문에 많은 메모리가 필요하게 된다. PM의 크기가 10bits 정도 된다고 하면 표 1에서 보듯이 다른 방식들에 비해서 엄청난 양의 메모리가 요구된다. MRE를 이용하는 방식이 latency면에서는 10% 정도 불리하지만, 메모리 측면에서는 1/4 정도의 훨씬 적은 양만을 필요로 한다.

따라서 MRE 방식을 이용하게 되면 메모리 블록의 수를 늘리고 줄임에 따라서 하드웨어 복잡도의 증가 없이 메모리 또는 latency에 중점을 두고 조절을 할 수 있다. 메모리의 크기를 줄이는 것에 중점을 두고 설계한다면 표1의 메모리 블록의 수를 3개로 줄이면 전체 메모리 크기는 4,900bit로 줄고 latency는 2T로 늘어난다. 반면에 latency에 중점을 두고 설계한다면 메모리 블록의 수를 10개로 늘려 메모리 크기는 6,664bit, latency는 11T/9가 되어 look-ahead trace-back 방식에 비해 1/4정도의 훨씬 적은 메모리 사용으로 가장 적은latency에 거의 근접하는 효과를 볼 수 있다.

제안된 방식의 비터비 디코더를 Verilog-HDL을 이용하여 설계하고 Xilinx Virtex-2 FPGA를 이용하여 구현하였다. 2,939 슬라이스가 사용되고 최대 동작 주파수는 46MHz이다. 표 2는 기존에 발표된 비터비 디코더들과 합성 결과와 비교한 것이다. 제안된 비터비 디코더의 성능과 면적이 기존의 결과에 비해 우수함을 알 수 있다. 또한 무선랜 규격인 802.11a와 DMB에 적용할 수 있는 충분한 성능을

표 2. 제안된 비터비 디코더의 구현 결과와 기존 결과와의 비교

	Ref [6]	Ref. [7]	Ref. [8]	Proposed
R	1/2	1/2	1/2	1/2
K	9	7	5	7
T	78Kbps	32Mbps	3.4Mbps	46Mbps
Area	2,793 S	87,836 GC (2,630 S*)	86,787 GC (2,600 S*)	2,939 S
Tech	XC2V1000	XCV400	XCV400	XC2V1000

	Ref. [9]	Ref. [10]	Proposed
R	1/2	1/3	1/2
K	7	9	7
T	100Mbps	20Mbps	103Mbps
Area	50K GC 3.88 mm2	32.5K TR (8.2K GC*) 5.76 mm2	47.5K GC
Tech	0.25u	0.25u	0.35u

T: Throughputs S : slices
TR: transistor counts GC: gate counts
* Estimated value

표 3. ACSU(K=7, R=1/2)의 개수에 따른 비터비 디코더의 합성 결과 (0.35um CMOS technology)

# of ACSU	1	2	4
Area [GC]	33,113	33,314	33,621
Memory [bits]	2,889		
T [Mbps]	3.1	6.2	12.8
# of ACSU	8	16	32
Area [GC]	34,902	37,131	41,591
Memory [bits]	2,889		
T [Mbps]	25.7	51.3	102.6

GC : gate counts T: Throughputs

보여줌을 알 수 있다.

ACSU는 비터비 디코더에서 가장 많은 면적을 차지하므로 비터비 디코더의 크기에 가장 큰 영향을 미친다. 따라서 ACSU의 수를 줄일 수 있다면 디코딩 속도의 감소를 감수하고 비터비 디코더가 차지하는 면적을 줄일 수 있다. 제안된 비터비 디코더는 메모리 블록의 수와 ACSU의 수를 변경시켜 사용 목적에 맞도록 코딩되어 간단한 파라미터의 변경만으로 원하는 사양의 하드웨어를 합성할 있다. 표 3은 ACSU의 개수에 따른 비터비 디코더의 합성 결과를 보여준다. 0.35um CMOS 공정 셀 라이브러리를 이용하였으며 DM과 LIFO는 SRAM으로 합성되었다. 합성 결과를 보면 ACSU의 수가 32개에서 1개로 줄었을 때 크기의 감소폭이 기대보다 크지 않은 것을 알 수 있는데 이는 SMM(State Metric Memory)과 PMM이 레지스터로 구현되어 큰 면적을 차지하고 이들은 ACSU의 수와 무관하게 일정한 크기를 갖기 때문이다. 전력 소모면에서는 32개의 ACSU를 이용하고 동작 주파수를 낮추는 것이 유리하다.

그림 4는 FPGA kit을 이용하여 제안한 비터비 디코더를 구현한 후 실험한 결과이다. 비터비의 제어는 내장된 ARM processor를 이용하였다. 맨 위의 그림이 원래 그림이며, 중간의 그림이 잡음이 첨가된 그림이다. 마지막으로 맨 아래의 그림이 잡음이 첨가된 신호가 비터비 디코더를 통과하여 복호된 그림이다. 잡음 레벨은 Eb/No=3dB를 사용하였으며 성공적으로 복호가 된 것을 볼 수 있다.

그림 4. FPGA로 비터비 디코더를 구현하여 이미지 파일 데이터에 잡음을 첨가 후 디코더를 통과시켜 잡음이 제거된 것을 보여주는 영상 이미지

Ⅳ. 결 론

Trace-back 과정이 없는 MRE 방식을 이용해서 블록 디코딩에 적용하여 적은 메모리 크기로 고속 디코딩이 가능한 비터비 디코더 구조를 제안하였다. MRE방식은 trace-forward 과정이 끝나자마자 시작 상태와 디코딩 bit을 결정하므로 이를 이용하여 효율적인 블록 디코딩을 수행할 수 있다. 기존의 블록 디코딩 방식과 비교하여 하드웨어 복잡도의 증가 없이 더 적은 메모리의 크기와 디코딩latency를 가지게 된다. 또한 메모리 블록의 개수와 크기를 조정함으로써 메모리 크기와 latency를 조정할 수 있다. 또한 몇 개의 디자인 파라미터를 조정하여 하드웨어 복잡도와 디코딩 속도를 조절 가능한 구조로 설계하였다. Xilinx Virtex2 FPGA를 이용하여 구현된 비터비 디코더는 2,939 슬라이스를 사용하고 최대

디코딩 속도는 46Mbps이다.

참 고 문 헌

[1] G. Forney, "Convolutional codes II. Maximum-likelihood decoding." *Information and Control*, 25(3), pp 222-266, July 1974.

[2] Gennady Feygin and P.G. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. On Commun.*, vol.41, no.3, pp.425-429, March 1993.

[3] Jung-Gi Baek, Sang-Hun Yoon, Jong-Wha Chong, "Memory efficient pipelined Viterbi decoder with Look-Ahead trace-back," *the 8th IEEE international Conference on Circuit and Systems*. vol.2, pp.769-772, 2001.

[4] Shu Lin, Daniel J. Costello, Jr, *Error Control Coding : Fundamentals and Applications*, Prentice Hall, 1983.

[5] Jae-Sun Han, Tae-Jin Kim, Chanho Lee, "HIGH PERFORMANCE VITERBI DECODER USING MODIFIED REGISTER EXCHANGE MEHODS", *IEEE International Symposium on Circuits And Systems 2004*, pp. DSP-P4.5, 2004.

[6] Man Guo, Omair Ahmad, M., Swamy, M.N.S., Chunyan Wang, "A low-power systolic array-based adaptive Viterbi decoder and its FPGA implementation," *Proceedings of the 2003 International Symposium on Circuits and Systems,* Vol. 2 ,pp. 25-28 May 2003,

[7] Zhu, Y.Benaissa, M., "A novel ACS scheme for area-efficient Viterbi decoders", *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, vol.2, pp. II-264-II-267, 2003

[8] Yiyan Tang, Yingtao Jiang, Yuke Wang, and M. N. S. Swamy, "A Trace-Back-Free Viterbi Decoder Using A New Survival Path Management Algorithm", *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Scottsdale, Arizona, pp. I-261 - I-264, vol.1, May 2002.

[9] Yu-xin You, Jin-xiang Wang, Feng-chang Lai, Yi-zheng Ye, "VLSI design and implementation of high-speed Viterbi decoder", *Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on*,. 1,pp. 64 - 68, 29 June-1 July 2002

[10] Xun Liu, Papaefthymiou, M.C., "Design of a high-throughput low-power IS95 Viterbi decoder" *Design Automation Conference, 2002. Proceedings. 39th*,pp. 263 268, 10-14 June 2002

김 태 진(Tae-Jin Kim) 학생회원

2003년 2월 숭실대학교 정보통신 전자 공학과 졸업

2003년 3월~현재 숭실대학교 전자 공학과 석사

<관심분야> 채널코딩

이 찬 호(Chanho Lee) 정회원

1987년 2월 서울대학교 전자 공학과 졸업

1989년 2월 서울대학교 전자 공학과 석사

1994년 6월 UCLA, 전자공학과 박사

1994년 8월~1995년 2월 삼성전자 반도체연구소 선임연구원

1995년 3월~현재 숭실대학교 정보통신전자공학부 부교수

<관심분야> 채널코덱의 구현, SoC on-chip-bus, SoC 설계방법론, 저전력 프로세서 설계