2003년도 대한전자공학회 하계종합학술대회 논문집 제26권 제1호

Modified Register Exchange 방식을 이용한 고성능 비터비 디코더 설계

한재선, 이찬호
숭실대학교 전자공학과

High performance Viterbi decoder using Modified Register Exchange methods

Jae-Sun Han, Chanho Lee
Department of Electronic Engineering, Soongsil University
E-mail : han3eb@engineer.ssu.ac.kr, chanho@e.ssu.ac.kr

Abstract

본 논문에서는 traceback 동작 없이 decoding 이 가능한 Modified Register Exchange 방식을 이용하여 이를 block decoding 에 적용하는 비터비 decoding 방식을 제안하였다. Modified Register Exchange 방식을 block decoding 에 적용함으로써 decision bit 들을 결정하기 위해 필요한 동작 사이클을 줄였고, block decoding 을 사용하는 기존의 비터비 디코더보다 더 적은 latency 가지게 되었다. 뿐만 아니라, 메모리를 더 효율적으로 사용할 수 있으면서 하드웨어의 구현에 있어서도 복잡도가 더 감소하게 된다. 제안된 방식은 같은 하드웨어 복잡도로도 메모리의 감소 또는 latency 의 감소에 중점을 둔 설계가 가능하다.

I. 서론

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 계속되었다. 채널 코딩은 데이터 전송에 있어서, 채널에서 발생되는 잡음에 의한 오류를 수신측이 검출 혹은 정정할 수 있도록 원래의 데이터에 새로운 데이터를 덧붙이는 방법이다. 비터비 알고리즘은 송신 단에서 길쌈부호화기를 사용할 때 수신 단에서 전송자료를 추정하기 위해 사용된다. 비터비 decoding 방식은 maximum likelihood decoding 으로써 L-bit 의 codeword 로부터 2^L개의 가능한 값 중에서 가장 유사한 값을 찾아내는 것이다.

비터비 디코더에 사용되는 일반적인 block decoding 방식으로 k-pointer 방식 [1,2], one pointer 방식 [3], hybrid traceback 방식 [4], Look-Ahead traceback 방식[5] 등이 있다. Look-Ahead traceback 방식을 제외한 나머지 방식들은 decoding 과정을 시작하기 위한 starting state 를 찾기 위해 merging 과정을 가지게 된다. Merging 과정에서는 merging 블록들에서 survival path 를 따라서 traceback 과정을 수행하면서 진행된 후 starting state 를 찾게 되며, decoding 과정에서는 decoding 블록에 대해 starting state 에서부터 traceback [6]이 진행되어 decoding 된 bit 들이 역순으로 출력이 된다. One pointer 방식과 hybrid traceback 방식은 k-pointer 방식보다 메모리를 보다 효율적으로 이용한다는 장점이 있다. 그러나 그 두 경우에 있어서는 traceback 과 decoding 동작이 Add-Compare-Select 연산보다 최소 2-3 배 더 빠르게 수행 되어야 한다[4].

그런데, 만약 merging 과정에서의 traceback 과정이 없어진다면 traceback 을 하면서 생겼던 latency 가 그만큼 줄어들 수 있을 것이다. 이는 merging 블록들에서 survival path 가 결정되자 마자 starting state 가 바로 결정이 된다면 가능하게 된다. Modified Register Exchange (MRE) 방식은 traceback 과정이 없이 decoding 이 가능하도록 한 구조로서 survival path 가 결정되면 바로 decoded data 가 결정되는 방식이다[7].

본 논문에서는 MRE 방식을 이용한 고성능의 비터비 디코더를 제안하였다. Starting state 는 MRE 방식을 사용해서 merging 과정에서의 traceback 과정을 제거함으로써 survival path 가 결정되자 마자 얻어지며, 따라서 decoding 된 data 의 출력도 traceback 을 없앤 만큼의 latency 를 줄인 효과를 보게 된다. 또한 traceback 과정이 없어지면서 필요한 메모리의 양도 감소한다. 이러한 과정은 hardware complexity 증가나 많은 메모리의 사용이 없이도 가능하다.

II. 비터비 디코더의 Block Decoding 방식

비터비 디코더의 block decoding 방식 중 k-pointer 방식, one pointer 방식, hybrid traceback 방식, Look-Ahead

traceback 방식들을 간단히 살펴보자.

k-pointer 방식은 여러 개의 메모리 블록을 이용하여 write 와 decode 가 병렬로 진행되는 방식이다. 즉, k 개의 포인터를 가지고 traceback 과정과 writing, decoding 과정을 다른 메모리 블록에서 동시에 수행을 하게 된다. 그림 1 은 k=3 인 3-pointer odd 방식을 예로서 설명한 것이다. 그림에서처럼 먼저 첫번째 메모리 블록에서부터 trace-forward 하면서 decision bit 를 저장하는 writing 과정(WR)이 계속해서 다음 블록으로 진행이 되고 블록 3 까지의 writing 과정이 완료가 되면 read 포인터 1 은 블록 3 에서부터 traceback 과정을 수행을 하기 시작하고 동시에 계속해서 블록 4 의 writing 과정을 수행하게 된다. Traceback 과정은 메모리 블록 1 을 decoding 하기 위한 이전 단계로써 survival path depth(T)만큼 이루어 져야 한다. 첫 번째 traceback 이 진행되면서 블록 4 의 writing 과정이 완료되면 read 포인터 2 에 의해 두 번째 traceback 이 메모리 블록 2 의 decoding 을 위해서 시작이 된다. 각 traceback 과정이 완료되자마자 decoding 될 블록에서는 decoding 과정의 시작점이 되는 starting state 가 결정이 되고 traceback 방식을 이용한 decoding 과정이 read 포인터 3 에 의해 수행이 된다. Traceback 방식이 decoding 과정에 적용이 되면 실질적으로 data 출력은 traceback 과정 중 LIFO(Last-In First-Out) 메모리에 저장이 되었다가 traceback 이 끝난 후 역순으로 출력이 된다[4]. LIFO 는 두개를 사용하여 각 decoding 블록이 번갈아서 사용하게 된다. k-pointer even 과 odd 의 차이는 decoding 블록과 writing 블록을 따로 두느냐, 아니면 decoding 과정 중 LIFO 에 저장 후 비는 공간을 이용하여 writing 블록과 decoding 블록을 같이 쓸 것이냐는 것만 차이가 난다. k-pointer even 방식은 크기가 T/(k-1)인 메모리블록을 2k 개 사용하게 되고, 이때 latency 는 2kT/(k-1)가 된다. k-pointer odd 방식은 크기가 T/(k-1)인 메모리블록을 2k-1 개를 사용하게 되고, latency 는 같다[4].

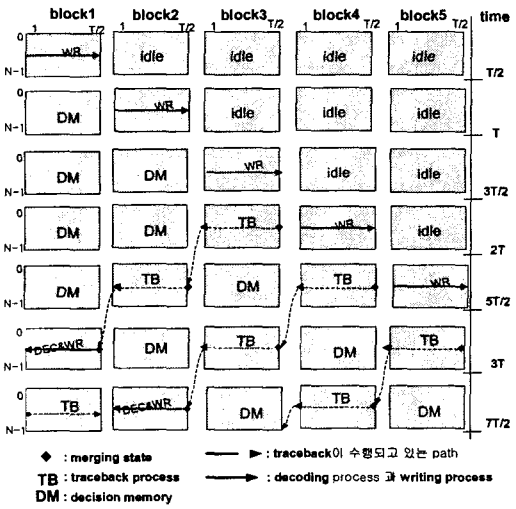


그림 1. 3-pointer odd 방식의 메모리 구조 및 동작

One pointer 방식은 read 포인터를 하나만 사용하는 대신 starting state 를 찾는 traceback 과정과 decoding 과정을 writing 과정보다 더 빠르게 처리한다. 이는 writing 과정이 traceback 동작의 reading 과정보다 시간이 더 걸리기 때문에 가능하다. Read 메모리 블록수가 3 인 경우 traceback 과정과 decoding 과정을 수행하는 속도는 writing 과정의 속도보다 3 배 빨라야 한다. 따라서 총 4 개의 메모리 블록 중 하나의 메모리 블록이 writing 과정을 수행해서 완료할 때까지, 나머지 3 개의 메모리 블록은 traceback 과정(또는 merging 과정)부터 decoding 과정까지 마치게 된다. k1 이 read 메모리 블록의 수라면 사용되는 메모리 블록은 T/(k1-1) 크기의 메모리가 k1+1 개 필요하고 latency 는 (k1+1)T/ (k1-1)가 되며 traceback 의 속도는 k1 배 빨라야 한다[4].

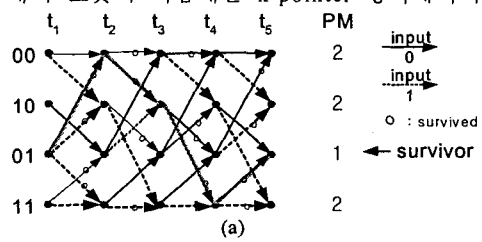
Hybrid traceback 방식은 k-pointer 방식과 one pointer 방식을 합쳐놓은 것으로써 k2 개의 read 포인터를 사용하면서 one pointer 처럼 k1 배만큼 traceback 하는 속도를 높여주게 된다. 이 때는 T/(k1k2-1) 크기의 memory 블록이 k2(k1+1)개(odd 의 경우 k2(k1+1)-1 개) 필요하게 되고, latency 는 k2(k1+1)T/(k1k2-1)가 된다[4].

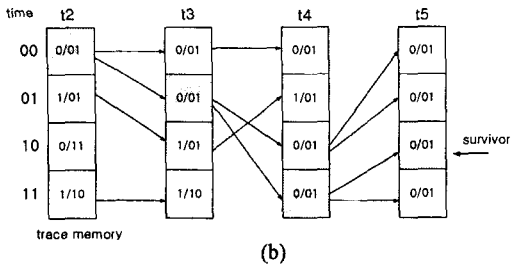
Look-ahead traceback 방식은 traceback 방식을 사용하지 않는 block decoding 방식으로서 path metric(PM)값을 survival path depth(T) 동안 매 stage 에 저장하면서 진행하므로 메모리의 소요가 크다. 하지만 그러한 PM 값들을 이용하여 traceback 없이 T 만큼만 look-ahead 방식을 사용하면 바로 데이터가 나오므로 latency 가 줄어들게 된다.

III. Modified Register Exchange 방식을 이용한 block decoding 방법

Traceback 동작이 없는 MRE 방식은 trace-forward 의 첫번째 stage 의 decision bit 와 state address 들을 저장한 후 다음 stage 에서 두 정보의 위치를 재정렬 한다. Survival path length(T) 만큼 trace-forward 가 진행된 후에는 곧바로 survival path 가 결정되어 그때 지정되는 decision bit 와 state address 가 얻고자 하는 decoded bit 와 starting address 가 된다[7]. 그림 2 는 이러한 MRE 방식을 보여준다. MRE 방식의 decoding 과정은 register exchange 방식[8]과 유사하지만, trace-forward 과정동안 중간 값들은 저장하지 않고 오직 첫번째 stage 의 decision bit 와 state address 만을 update 시켜 나간다는 것이 다르다[7]. 그림 2(b)에서 trace-forward 과정이 끝난 후 최소 PM 값을 가지는 메모리의 decision bit 와 state address 가 0 과 01 로 각각 정해지며, 이는 decoded bit 는 0 이고, starting state 는 01 임을 나타낸다.

MRE 방식을 사용해서 block decoding 하기 위한 블록 다이어그램이 그림 3 에 나타나 있다. 이 방식은 그림 4 에서 보듯이 처음에는 k-pointer 방식에서처럼





(b)

그림 2. (a) Trellis diagram (b) 각 trace-forward stage 에서의 메모리 update 상태

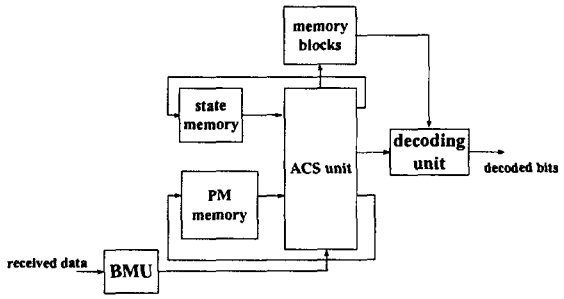


그림 3. MRE 방식을 이용한 비터비 디코더 블록도

writing 과정이 진행된다. 두 번째 writing 과정부터는 첫 번째 메모리 블록의 starting state(또는 merging state) 를 찾기 위해서 MRE 방식이 사용된다. 즉, 두 번째 블록에서 첫 번째 stage 의 state address 들이 state 메모리에 저장이 되고, 이 값들은 첫 번째 MRE 방식(M1)에 사용된다. Block decoding 에 MRE 방식이 적용될 때에는 기존의 MRE 방식과는 약간 다르게 첫번째 stage 의 decision bit 들은 사용하지 않고 state address 만 사용하므로 state 메모리에 저장할 필요가 없다. 세 번째 writing 과정이 시작되면 M1 과 M2 의 MRE 과정이 동시에 진행되고 첫번째와 두 번째 state 메모리에 있는 내용은 같은 방식으로 재정렬된다. 블록 1 의 decoding 과정을 위한 블록 2 의 starting state 는 M1 의 과정이 블록 5 까지 (survival path depth T 만큼) 진행되어서 마치게 되면 결정이 된다(①). Starting state 가 결정이 되면, decision 메모리 블록 1 의 traceback 방식이 starting state 를 시작점으로 해서 진행된다. Decoding 과정은 k-pointer odd 방식[2]과 동일하게 decoding unit 을 통해서 이루어진다. Decision 메모리 블록 1 의 decoding 과정이 끝난 후에는 블록 3 에 대한 starting state 가 결정이 되고(②), 블록 2 에 대한 decoding 과정이 계속해서 진행이 된다. 이와 동시에 LIFO 메모리에 있던 decoded data 들은 출력으로 나오게 된다. 이때 최종 출력이 나올 때까지 걸리는 latency 는 3T/2 = 3x36/2 = 48 clocks(constraint length(K)=7, T=36 때) 이 되고, 이 이후로는 매 clock 마다 decoded bit 들이 나오게 된다. Decision 메모리가 m 개의 블록들로 나누어지면 그 때 각 블록의 크기는 T/(m-1)가 되며, MRE 과정은 m-1 개의 bank 에 걸쳐서(survival path depth T 만큼) 진행된다.

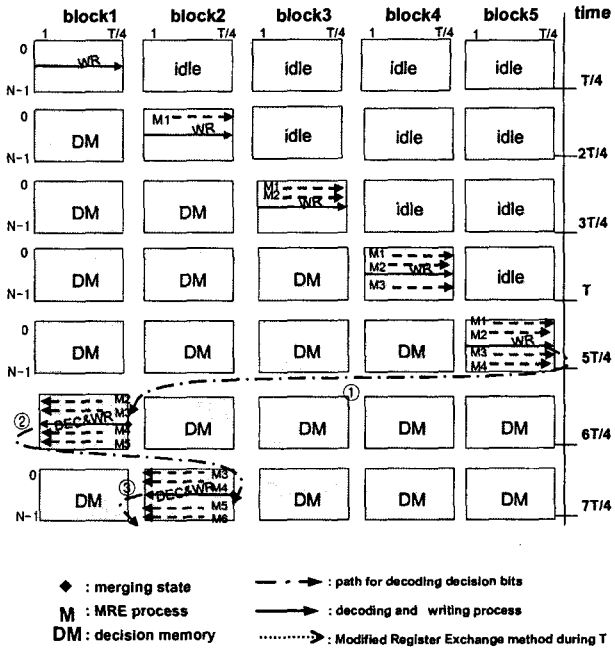


그림 4. MRE 를 이용한 5 메모리 block decoding 방식. 3-pointer 방식에 비해 메모리 블록의 크기가 절반이어서 전체 메모리 크기가 크게 감소한다.

MRE 방식을 이용하여 block decoding 을 하게 되면 merging 과정에서 traceback 을 사용하는 다른 방식에 비해서 더 적은 메모리 블록을 사용하게 된다. 그림 5 는 MRE 를 사용함으로써 메모리 블록의 수가 감소하고 latency 가 줄어드는 것을 보여준다.

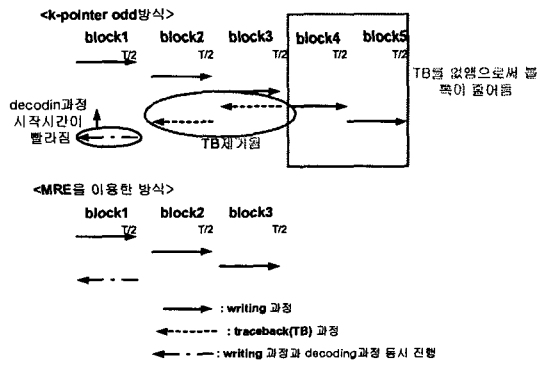


그림 5. MRE 방식을 사용함으로써 달라지는 과정

표 1 은 다른 비터비 block decoding 방식과의 비교를 나타낸 것이다. Constraint length(K) = 7, Code Rate(R) = 1/2, survival path depth(T) = 36 으로 놓았다. 3-pointer 방식 (even, odd), one pointer 방식, hybrid even 방식, Look-Ahead 방식을 제안된 방식, 5 개의 메모리 블록을 가지고 MRE 를 이용하는 방식과 비교하였다.

메모리 블록의 크기가 더 작아질수록, look-ahead

traceback 방식을 제외한 다른 방식들은 더 많은 메모리 블록과 더 많은 Traceback Unit(TBU)와 decoding unit 이 필요하게 되므로 이는 hardware complexity 와 power consumption 에 있어서 더 나빠지게 된다. 예로써, 5-pointer odd 방식은 9x64bits 크기의 메모리 블록 9 개와 5 개의 가 필요하다. 하지만, MRE 를 사용하는 방식은 9x64bits 크기의 메모리 블록 5 개와 오직 1 개의 TBU 만이 필요하다. 뿐만 아니라, starting state(또는 merging state)가 trace-forward 과정이 끝나자마자 결정되므로 decoded bit 가 출력될 때까지의 latency 가 더 줄어들게 된다. k-pointer 방식은 메모리 블록의 수가 증가하게 되면 메모리 와 latency 가 줄어들지만, 포인터의 수가 늘어나므로 TBU 가 증가하게 되는 단점이 있고, MRE 를 사용하는 방식은 TBU 의 증가 없이 latency 를 줄이지만 state 메모리의 수가 늘어나므로 전체 메모리 수가 좀 더 증가된다. One pointer 방식과 hybrid traceback 방식은 traceback 하는 속도를 높여줘야 하는 점에서 MRE 방식과 큰 차이가 난다.

표 1 에서 Look-Ahead traceback 방식은 비교된 방식들 중에서 가장 적은 latency T 를 가지지만, trace-forward 과정에서 매 stage 에서 모든 PM 값을 저장하기 때문에 많은 메모리가 필요하게 된다. PM 의 크기가 10bits 정도 된다고 하면 표 1 에서 보듯이 다른 방식들에 비해서 엄청난 양의 메모리가 요구된다. MRE 를 이용하는 방식이 latency 면에서는 10% 정도 불리하지만, 메모리 측면에서는 1/4 정도의 훨씬 적은 양만을 필요로 한다.

이러한 점들로 미루어 보아 MRE 방식을 이용하게 되면 메모리 블록의 수를 늘리고 줄임에 따라서 hardware complexity 증가 없이 메모리 또는 latency 에 중점을 두고 조절을 할 수 있다. 메모리의 크기를 줄이는 것에 중점을 두고 설계한다면 표 1 의 3 뱅크짜리 MRE 방식처럼 메모리 블록의 수를 줄일 경우, 비슷한 hardware complexity 와 latency 를 가지는 hybrid even 방식(k1=k2=2)보다도 전체 메모리의 양을 줄일 수 있고, latency 에 중점을 두고 설계한다면 표 1 의 10 뱅크짜리 MRE 방식처럼 메모리 블록을 늘릴 경우, 1/4 정도의 훨씬 적은 메모리 사용으로 latency 가 가장 적은 look-ahead traceback 방식의 latency 에 거의 근접하는 효과를 볼 수 있다.

IV. 결론

Traceback 과정이 없는 Modified Register Exchange 방하였다. MRE 방식은 trace-forward 과정이 끝나자마자 starting state 와 decoded decision bit 를 결정하므로 이를 이용하여 효율적인 block decoding 을 수행할 수 있다. 이 방식은 decoding 과정의 시작을 위한 starting state 를 구하는데 traceback 과정이 필요 없고 이에 따라 메모리 블록의 크기가 작아지는 장점이 있다. 다른 일반적인 block decoding 방식과 비교할 때 hardware complexity 의 증가 없이 더 적은 메모리의 크기와 decoding latency 만을 가지게 된다. 이는 같은 hardware complexity 로도 메모리 또는 latency 에 중점을 두는 설계가 가능하게 하며, 실제로 메모리 블록의 수를 줄일 경우 hybrid traceback 방식과 비교해서 hardware complexity 와 latency 가 더 우수하면서도 메모리의 크기를 줄일 수 있고, 메모리 블록의 수를 늘릴 경우 look-ahead traceback 방식과 비교해서 훨씬 더 적은 메모리로 비슷한 latency 를 가질 수 있다.

참고문헌

[1] R.Cypher and C.B.Shung, "Generalized traceback techniques for survivor memory management in the Viterbi algorithm," in *GLOBECOM*, Dec.1990, pp. 1318-1322.

[2] O.Collins and F.Pollara, "Memory management in traceback Viterbi decoders," TDA Prog.Rep. 42-99, Jet Prob. Lab., Pasadena, CA, Nov. 1989.

[3] G.Feygin, P.G.Gulak, "Survivor sequence memory management in Viterbi decoders," CSRI Tech. Rep 262, Univ. of Toronto, Jan. 1991.

[4] 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.

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

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

[7] 이찬호, 노승효,"변형된 레지스터 교환 방식의 비터비 디코더 설계", *대한전자공학회논문지* SD,2003.01 v.40, n.1, pp.36-44 1229-6368.

[8] Jens Sparso, Henrik N. Jorgensen, Erik Paaske, Steen Pendersen, and Thomas Rubner-Petersen, "An Area-Efficient Topology for VLSI Implementation of Viterbi Decoders and Other Shuffle-Exchange Type Structures", *IEEE J. Solid-State Circuit*, vol. 26, No.2, pp. 90-97, FEBRUARY 1991

표 1. 메모리 크기, latency, hardware complexity 측면에서의 비교(K=7, T=36 인 경우)

	3-pointer even	3-pointer odd	5-pointer odd	One pointer (k1=3)	Hybrid even (k1=k2=2)	Look-ahead traceback	MREM (3 banks)	MREM (5 banks)	MREM (10 banks)
Decision memory	6912 bit	5760 bit	5184 bit	4608 bit	4608 bit	2304 bit	3456 bit	2880 bit	2560 bit
PM memory	640 bit	640 bit	640 bit	640 bit	640 bit	23040 bit	640 bit	640 bit	640 bit
State memory							768 bit	1536 bit	3456 bit
LIFO	36 bit	36 bit	18 bit	36 bit	36 bit	N/A	36 bit	18 bit	8 bit
Total memory	7588 bit (100%)	6432 bit (84%)	5842 bit (77%)	5280 bit (69%)	5280 bit (69%)	25344 bit (334%)	4900 bit (65%)	5074 bit (67%)	6664 bit (88%)
Latency	3T	3T	5T/2	2T	2T	T	2T	3T/2	11T/9
ACSU	64 unit	64 unit	64 unit	64 unit	64 unit	64 unit	64 unit	64 unit	64 unit
TBU	2	2	4	N/A	1**	N/A	0	0	0
Decoding Unit	1	1	1	1*	1**	N/A	1	1	1

* : Decoding Unit 이 TBU 역할도 함, 3 배의 속도로 동작해야 함

** : 2 배의 속도로 동작해야 함