논문 2010-47SD-7-7

경로 메트릭 데이터의 효율적인 관리를 통한 고성능 비터비 디코더 회로 설계

(Design of High-performance Viterbi Decoder Circuit by Efficient Management of Path Metric Data)

김 수 진*, 조 경 순**

(Soojin Kim and Kyeongsoon Cho)

요 약

본 논문은 고성능 비터비 디코더 회로 구조를 제안한다. 제안하는 비터비 디코더는 가지 값의 특징을 이용하기 때문에 추가적인 메모리를 사용하지 않고 가지 메트릭을 계산할 수 있다. 또한 빠른 합-비교-선택 연산을 위해 경로 메트릭 데이터를 SRAM과 레지스터에 적절하게 재배열함으로써 디코더 전체의 속도를 75%까지 향상시킨다. 제안하는 비터비 디코더 회로를 Verilog HDL로 설계하였으며 130nm 표준 셀 라이브러리를 이용하여 게이트 수준 회로로 합성하였다. 제안하는 회로는 8,858개의 게이트로 구성되며 회로의 최대 동작 주파수는 130MHz이다.

Abstract

This paper proposes the architecture of high-performance Viterbi decoder circuit. The proposed circuit does not require additional memory to calculate the branch metrics because it uses the characteristics of the branch data. The speed of the Viterbi decoder circuit is increased up to 75% by rearranging the path metric data in SRAM and registers properly for fast add-compare-select operations. We described the proposed Viterbi decoder circuit in Verilog HDL and synthesized the gate-level circuit using 130nm standard cell library. The synthesized circuit consists of 8,858 gates and its maximum operating frequency is 130MHz.

Keywords: Viterbi 디코더, ACS 회로, BMU 회로, REU 회로, 데이터 재배열

I. 서 론

채널 부호화란 채널에서 일어나는 여러 가지 문제점들에 대한 대책으로, 송신 신호를 변형시켜 통신 성능을 향상시키기 위한 신호 변환 기술의 한 분야로서 오류 정정 부호화(ECC, Error Correction Code)라고도 한다[1]. 통신 채널을 통하여 신호를 전송할 경우 발생되는 오류를 정정하기 위한 방법으로 많은 디지털 시스템에서 순방향 오류 정정(FEC, Forward Error Correction) 방식을 도입하고 있다. FEC 방식은 보내고자 하는 정보 데이터에 패리티 비트를 부가하여 전송함으로써 수신 단에서 직접 오류를 검출 및 정정할 수 있는 기법[2]으로, 랜덤 오류에 강한 특성을 보이는 비터비 알고리즘(Viterbi algorithm)[3~4]이 널리 이용되고 있다.

비터비 디코더는 여러 통신 환경에서 사용되고 있기 때문에 회로의 크기와 성능을 향상시키기 위한 많은 방법들이 제안되어 왔다[5~11]. 회로의 크기 면에서 효

* 학생회원, ** 평생회원, 한국외국어대학교 전자공학과
(Department of Electronics Engineering, Hankuk University of Foreign Studies)

※ 본 연구는 지식경제부 및 한국산업기술평가관리원의 산업원천기술개발사업(정보통신)[KI002145, 차세대 광통신용 디지털 신호처리 기반 초고속 CMOS 회로 설계 기술]과 2010학년도 한국외국어대학교 교내학술연구비 지원의 일환으로 수행하였음.
접수일자: 2009년10월17일, 수정완료일: 2010년6월15일

율적인 비터비 디코더의 구조를 제안하는 연구[5]에서는 BMU(Branch Metric Unit) 회로의 크기를 줄이기 위하여 미리 계산된 가지 메트릭을 ROM(Read Only Memory)에 저장하여 참조하는 방식을 사용하였고, ACS(Add-Compare-Select) 연산의 특징을 이용한 연구[6~7]에서는 가산기의 수를 줄인 구조를 사용하였다. 또한 ACS 회로 4 개를 파이프라인 구조에 적용하여 비터비 디코더의 속도를 높이기 위한 방법도 제안되었다[8].

본 논문은 고성능 비터비 디코더 회로를 제안하고 있으며, 제안하는 회로는 가지 값의 특징을 이용하여 별도의 메모리를 사용하지 않고 가지 메트릭을 계산하는 BMU 회로, 효율적인 데이터 재배열과 메모리 사용을 통해 디코더 전체의 속도를 크게 향상시키는 ACS 회로 및 REU(Register-Exchange Unit) 회로로 구성된다. 제안하는 회로는 디코더 전체의 속도를 좌우하는 ACS 회로의 처리 속도를 크게 향상시키기 때문에 디코더 회로의 전체적인 속도를 향상시킬 수 있으며, 가지 값을 저장하기 위한 별도의 메모리를 사용하지 않기 때문에 회로의 크기 면에서도 효율적이다.

Ⅱ. 비터비 디코더 알고리즘

비터비 디코더 알고리즘은 채널을 통해 수신되는 데이터들을 여러 경로를 통하여 탐색한 후 그 중에서 가장 유사성이 높은 경로를 선택하고, 선택된 경로의 데이터를 디코딩하는 방식을 사용한다.

비터비 디코딩 과정은 크게 세 가지로 구분될 수 있다. 첫 번째는 수신된 데이터와 기존의 가지 값과의 차이를 계산하는 가지 메트릭 계산 과정이고, 두 번째는 하나의 상태로 입력되는 두 종류의 가지 메트릭과 경로 메트릭의 합을 비교하여 둘 중에 더 작은 값을 갖는 경로를 생존 경로로 선택하는 과정이다. 마지막 세 번째는 선택된 생존 경로에 대한 정보를 이용하여 최종 디코딩된 데이터를 출력하는 과정이다.

1. BMU 알고리즘

BMU는 수신된 데이터와 각 가지 값의 차이인 해밍 거리(Hamming distance)를 나타내는 가지 메트릭을 계산하는 역할을 한다. 그림 1은 수신된 데이터 중의 일부 2비트 데이터가 '00'일 때 가지 메트릭을 계산하는 과정을 예로 나타내고 있다. 그림 1에서는 각 시간 단

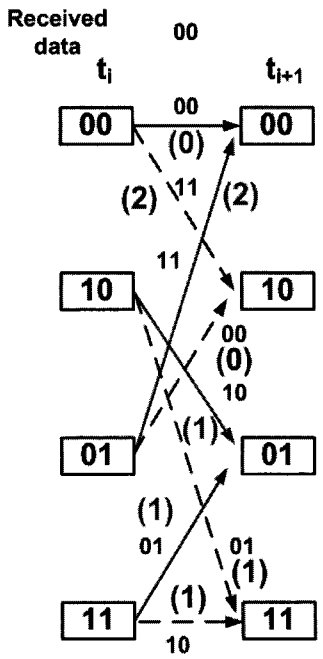


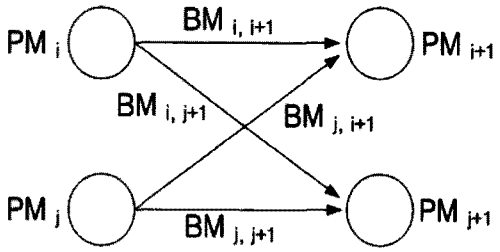
그림 1. 가지 메트릭 계산의 예
Fig. 1. Example of computing branch metrics.

위 t에서 4개의 상태가 존재하며, 이전 상태에서 다음 상태로의 상태 천이는 가지를 의미하는 화살표를 따라 이루어진다. 각 가지 옆의 데이터는 가지 값을 나타내고, 괄호 안의 값은 수신된 데이터와 가지 값과의 차이인 가지 메트릭을 나타낸다. 수신된 데이터가 '00'일 때 가지 값이 '00'인 경우에는 두 데이터 간의 차이가 없기 때문에 가지 메트릭은 '0'이 되고, 반대로 가지 값이 '11'인 경우에는 두 비트가 모두 차이나기 때문에 가지 메트릭은 '2'가 된다. 가지 값이 '01'이거나 '10'인 경우는 수신된 데이터와 한 비트가 차이나기 때문에 가지 메트릭은 '1'이 된다.

2. ACS 알고리즘

비터비 디코더 알고리즘의 두 번째 단계는 계산된 가지 메트릭을 이용하여 매 단마다 경로 메트릭을 갱신하고 생존 경로를 찾는 것이다. 경로 메트릭은 가지 메트릭을 누적시킨 값과 같으며, 하나의 상태로 들어오는 두 개의 누적된 가지 메트릭 중에서 작은 값을 선택하는 합-비교-선택 연산을 하는 것이 ACS 알고리즘의 과정이다. 이 과정에서 선택되지 않은 경로는 제거되고 선택된 경로는 생존 경로로 저장하게 된다. 생존 경로에 대한 정보는 비터비 디코더의 최종 디코딩된 데이터를 생성하기 위해 사용된다.

어느 한 시간 구간에서의 천이들은 서로 겹치지 않는



$$PM_{i+1} = \min(PM_i + BM_{i,i+1},\ PM_j + BM_{j,i+1})$$
$$PM_{j+1} = \min(PM_i + BM_{i,j+1},\ PM_j + BM_{j,j+1})$$

그림 2. 버터플라이 구조와 최소 경로 메트릭 연산식
Fig. 2. Butterfly architecture and minimum path matric equations.

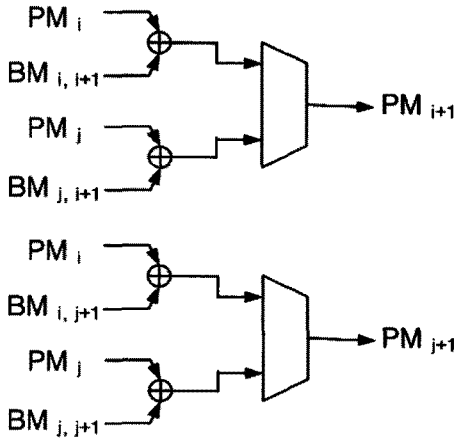


그림 3. 합-비교-선택 연산
Fig. 3. Add-compare-select operations.

셀들로 묶을 수 있는데, 이를 버터플라이 구조라고 한다. 그림 2는 비터비 디코더의 버터플라이 구조와 최소 경로 메트릭에 대한 연산식을 나타내고 있다. 하나의 버터플라이에는 2개의 이전 상태와 2개의 현재 상태가 있으며 총 4개의 가지가 존재한다. 비터비 디코더는 그림 2의 연산식을 통해 하나의 현재 상태로 들어오는 두 개의 이전 경로 메트릭(PM)과 가지 메트릭(BM)의 합 중 작은 값을 선택한다. 그림 3은 이러한 합-비교-선택 연산을 나타내고 있다. 비터비 디코더는 모든 상태에 대해 합-비교-선택 연산을 하여 경로 메트릭을 갱신하고 선택된 생존 경로에 대한 정보를 저장하며, 이러한 과정은 디코딩 과정이 끝날 때까지 반복된다.

3. REU 알고리즘

레지스터 교환 방식은 생존 경로에 대한 정보를 처리하는 방식 중 하나이며, 디코딩 단계가 진행됨에 따라 생존 경로에 대한 정보를 저장하고 있는 레지스터가 갱신되고 정보들이 서로 교환된다. 레지스터는 각 상태마다 존재하며 레지스터의 크기는 디코딩 길이에 따라 달라진다.

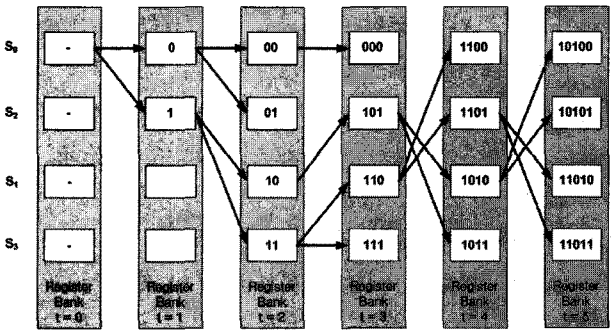


그림 4. 레지스터 교환 방식
Fig. 4. Register-exchange method.

그림 4는 레지스터 교환 방식의 예를 나타내고 있다. 네 개의 각 상태는 디코딩 정보를 저장하기 위한 레지스터가 있으며, 각 화살표는 생존 경로를 나타낸다. 현재 상태의 레지스터는 이전 상태의 레지스터가 저장하고 있던 데이터를 생존 경로를 따라 그대로 가져오며 LSB(Least Significant Bit)에 새로운 디코딩 정보를 한 비트 씩 덧붙인다. 디코딩 단계가 디코딩 길이에 도달했을 때 생존 경로에 대한 새로운 정보가 레지스터에 저장됨에 따라 MSB(Most Significant Bit)에 저장된 가장 오래된 비트가 최종 디코딩된 데이터로 출력된다.

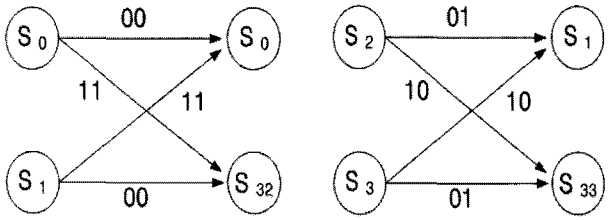
III. 제안하는 고성능 비터비 디코더 회로

비터비 디코더는 기능에 따라 크게 세 가지로 구분할 수 있는데, 그 중에서도 ACS 연산 과정의 처리 시간이 가장 길게 소요되며, 이는 비터비 디코더의 전체적인 속도를 좌우하게 된다. 따라서 고성능 비터비 복호기를 설계하기 위해서는 ACS 회로의 처리 속도를 높여야 한다. 본 절에서는 효율적인 데이터 재배열과 메모리 사용을 통해 ACS 회로의 처리 속도를 향상시키며, BMU 회로에서의 메모리의 사용을 줄이는 방법을 설명한다.

1. 별도의 메모리 사용이 없는 가지 메트릭 연산

BMU 회로는 수신된 데이터와 각 가지 값과의 차이를 계산하는 역할을 한다. 각 가지 값은 제한 길이(constraint length) K와 부호화율(coding rate) R에 따라 값이 정해져 있기 때문에 미리 계산된 가지 값들을 메모리에 저장하여 참조하는 방법이 주로 사용된다.

본 논문에서 제안하는 비터비 디코더의 BMU 회로는 각 버터플라이에 있는 가지 값의 특징을 이용하여 별도



(a) 첫 번째 버터플라이 (b) 두 번째 버터플라이

그림 5. K=7, R=1/2인 경우 버터플라이의 예
Fig. 5. Example of butterflies for K=7, R=1/2.
(a) The 1st butterfly (b) The 2nd butterfly

의 메모리 사용 없이 가지 메트릭을 계산한다. 그림 5는 제한 길이 K가 7이고 부호화율 R이 1/2인 경우에 하나의 단에 존재하는 32개의 버터플라이 중에서 첫 번째와 두 번째 버터플라이를 예로 나타내고 있다.

그림 5(a)는 첫 번째 버터플라이에서의 이전 상태 S_0와 S_1에서 현재 상태 S_0과 S_{32}로의 상태 천이를 나타낸다. 현재 상태가 S_0인 경우 이전 상태 S_0로부터 들어오는 가지 값은 '00'이고 이전 상태 S_1으로부터 들어오는 가지 값은 '11'이므로, 현재 상태 S_0에 대한 1쌍의 가지 값은 '0011'로 표현할 수 있다. 현재 상태가 S_{32}인 경우에는 이전 상태 S_0로부터 들어오는 가지 값은 '11'이고 이전 상태 S_1으로부터 들어오는 가지 값은 '00'이므로, 현재 상태 S_{32}에 대한 1쌍의 가지 값은 '1100'으로 표현할 수 있다. 이와 같이 하나의 현재 상태에 연결된 두 개의 가지는 서로의 비트를 반전한 것과 같으며, 하나의 버터플라이에 있는 두 개 상태에 대한 1쌍의 가지 값 또한 서로의 비트를 반전한 것과 같다. 이러한 특징은 그림 5(b)의 두 번째 버터플라이에서도 확인할 수 있으며 이는 모든 버터플라이에 적용된다.

제한 길이 K가 7이고 부호화율 R이 1/2인 경우 각 단의 32개 버터플라이마다 2비트의 가지가 4개 존재한다. 따라서 128개의 가지 값을 모두 저장하기 위해서는 256(128x2) 비트 크기의 메모리가 필요하다. 본 논문에서 제안하는 BMU 회로는 가지 메트릭을 계산하기 위해 모든 가지 값을 미리 저장하기 위한 별도의 메모리를 사용하지 않고 2 쌍의 가지 값을 저장하기 위한 8비트 내부 버퍼만을 사용한다.

2. 효율적인 데이터 재배열과 메모리의 사용

ACS 회로는 최대 유사(maximum likelihood) 복호 방식을 이용하여 가능성이 없는 경로를 미리 제거하기 위해 이전 상태로부터 전달되는 두 개의 경로 메트릭과 가지 메트릭의 합 중 작은 값을 선택하여 현재 상태의 경로 메트릭으로 저장한다. 하나의 경로 메트릭에 대한 ACS 연산은 메모리를 1개 사용할 경우 이전 상태의 경로 메트릭 데이터를 읽고 현재 상태의 경로 메트릭 데이터를 쓰는 시간을 포함하여 2 사이클이 소요된다. 따라서 ACS 회로는 N개의 상태가 있는 하나의 단을 처리하는데 2N 사이클이 소요된다[5].

가지 메트릭과 경로 메트릭의 합을 비교하여 생존 경로를 선택하는 ACS 회로는 메모리의 사용을 통한 연산 과정으로 인해 비터비 디코더 전체의 속도를 느리게 한다. 따라서 고성능 비터비 디코더의 설계를 위해서는 이 문제를 해결하여야 한다. 본 논문에서는 ACS 연산의 처리 속도를 N/2 사이클로 감소시키기 위하여 한번에 2개 상태의 데이터를 재배열하고, 각 단 사이에 지연을 없애기 위해 SRAM(Static Random Access Memory)과 함께 별도의 내부 레지스터를 사용하였다.

그림 6은 제한 길이 K가 7인 비터비 디코더의 ACS 회로에서 단일 포트 메모리 2개를 번갈아 사용하는 경우를 나타내고 있다. 첫 번째 버터플라이에 대한 연산은 이전 상태 0, 1의 데이터를 입력받아 현재 상태 0, 32의 데이터를 출력한다. 두 번째 버터플라이에 대한 연산은 이전 상태 2, 3의 데이터를 입력받아 현재 상태

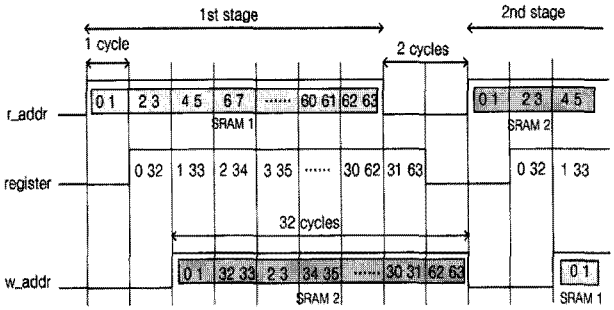


그림 6. 메모리를 사용한 데이터 재배열
Fig. 6. Data rearranging by using memory.

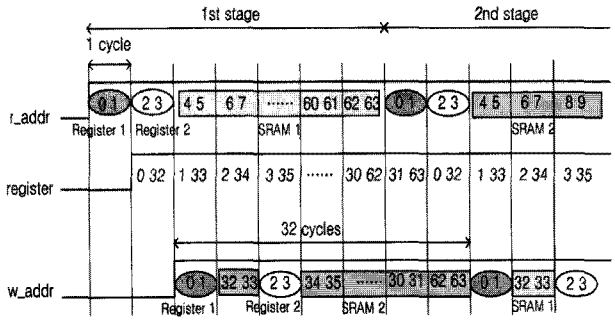


그림 7. 개선된 데이터 재배열
Fig. 7. Improved data rearranging.

1, 33의 데이터를 출력한다. 버터플라이는 입력과 출력 데이터의 순서가 다르기 때문에 1 사이클에 하나의 ACS 연산 결과를 메모리에 저장하기 위해서는 그림 6과 같이 별도의 내부 레지스터를 이용한 데이터 재배열이 필요하다.

데이터를 동시에 읽고 쓸 수 없는 메모리를 사용하는 경우에 K > 3이면 그림 6과 같이 매 단 사이에 2 사이클의 지연이 생긴다. 따라서 본 논문에서는 그림 7과 같이 일부 데이터를 별도의 내부 레지스터에 따로 저장하여 각 단 사이에 지연이 없도록 하였다. 제안하는 방식에서는 효율적인 데이터 재배열과 메모리 사용을 통해 하나의 단을 처리하는데 N/2 사이클이 소요되며 각 단을 연속적으로 처리할 수 있다. 제안하는 방식으로 설계한 비터비 디코더의 ACS 회로는 64개의 상태가 존재하는 하나의 단을 처리하는데 32사이클이 소요되며 그림 7과 같이 각 단 사이에 지연이 없다.

그림 8은 본 논문에서 ACS 회로의 처리 속도 향상을 위해 사용한 회로 구조이다. 제안하는 방식은 ACS 연산의 특징을 이용해 가산기의 수를 줄인 구조를 적용하였다[6~7]. 하나의 상태로 전달되는 2개의 경로 메트릭과 가지 메트릭의 합을 비교하는 대신, 각각의 경로 메트릭과 가지 메트릭의 값을 먼저 비교한 후 선택된 데이터들을 더하여 현재 상태의 경로 메트릭으로 결정하는 구조는 그림 3과 같은 기존의 회로 구조보다 가산기의 수를 줄일 수 있다. 또한 제안하는 회로는 속도 향상을 위해 단일 포트 메모리를 2개 사용하였다. 각 단을 지연 없이 연속적으로 처리하기 위해 내부 레지스터를 사용하였으며 효율적인 데이터 재배열을 통해 다른 연구[5]보다 처리 속도를 1/4로 감소시킬 수 있다.

경로 메트릭 데이터는 디코딩 길이가 42일 때 최대 7비트이고, 1개의 단일 포트 메모리는 30개의 버터플라

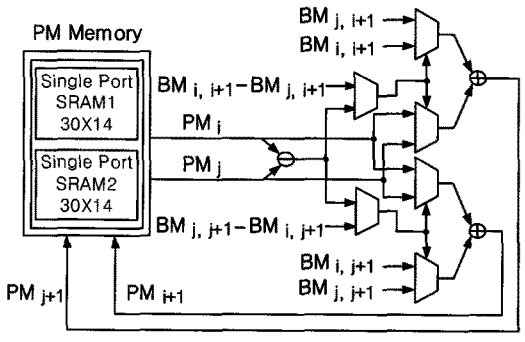


그림 8. 제안하는 ACS 회로 구조
Fig. 8. Architecture of the proposed ACS circuit.

이에 대한 경로 메트릭 데이터를 2개의 상태씩 동시에 저장하기 때문에 420(30x14)비트이다. 각 단을 연속적으로 처리하기 위해 사용되는 별도의 내부 레지스터는 2개 버터플라이에 대한 경로 메트릭을 저장하므로 28(14x2)비트이다.

3. 레지스터 교환방식을 이용한 디코딩

REU 회로는 ACS 회로에서 선택한 생존 경로에 대한 정보에 따라 각 상태에 해당하는 디코딩 정보들을 서로 교환하는 역할을 한다. 하나의 상태에 있는 2개의 가지 중 위쪽 가지가 생존 경로로 선택되면 디코딩 정보는 '0'이 되며, 아래쪽 가지가 선택되면 디코딩 정보는 '1'이 된다. 현재 상태에 대한 디코딩 정보는 선택된 이전 상태에 저장되어 있던 정보에 0 또는 1의 새로운 디코딩 정보를 LSB에 덧붙이고, 이러한 과정은 디코딩 과정이 끝날 때까지 반복된다.

REU 회로는 디코딩 단계가 디코딩 길이에 도달할 때부터 ACS 회로로부터 전달받은 가장 가능성이 큰 경로, 즉 최소거리에 대한 정보를 이용하여 매 단마다 디코딩된 데이터를 1 비트씩 출력한다. 그림 9는 본 논문에서 제안하는 REU 회로의 구조이며, ACS 회로와 마찬가지로 2개의 단일 포트 메모리를 사용하였다. 사용되는 단일 포트 메모리 1개의 크기는 30개 버터플라이에 대한 데이터를 2개의 상태씩 동시에 저장하기 때문에 2,520(30x84)비트이다.

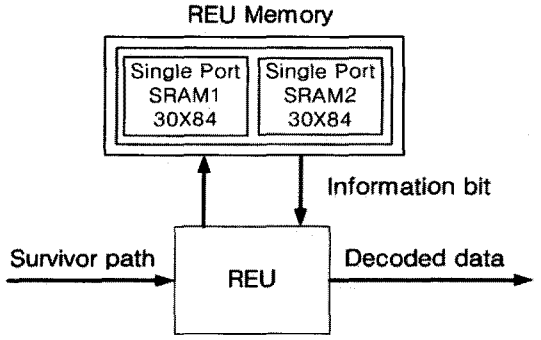


그림 9. 제안하는 REU 회로 구조
Fig. 9. Architecture of the proposed REU circuit.

Ⅳ. 실 험 결 과

본 논문에서 제안하는 고성능 비터비 디코더 회로는 Verilog HDL을 사용하여 RTL(Register Transfer Level)로 설계하였으며, Cadence 사의 NC-Verilog를

표 1. 제안하는 비터비 디코더 회로 합성 결과
Table 1. Synthesis results of the proposed Viterbi decoder circuit.

	Gate 개수	최대 지연시간(ns)
BMU	557	1.85
ACS	1,967	7.18
REU	6,331	3.17
비터비 디코더	8,858	7.67

사용하여 검증하였다. Synopsys 사의 Design-Complier를 통해 130nm 표준 셀 라이브러리 공정을 이용하여 게이트 수준의 회로로 합성하였다. 표 1은 제안하는 비터비 디코더 회로의 합성 결과를 나타내고 있다. K = 7일 때 ACS 회로를 한 개만 사용한 경우 비터비 디코더 회로 전체는 8,858 게이트 수를 가지며 최대 동작 주파수는 130MHz이다.

본 논문에서 제안하는 BMU 회로는 각 버터플라이에 있는 가지 값의 특성을 이용하여 별도의 외부 메모리 없이 가지 메트릭을 계산할 수 있다. 회로의 크기 면에서 효율적인 비터비 디코더의 구조를 제안하는 방식[5]에서는 BMU 회로의 크기를 줄이기 위하여 미리 계산된 가지 메트릭을 ROM에 저장하여 참조하는 방식을 사용하였지만, 본 논문에서 제안하는 BMU 회로는 256비트의 메모리를 사용하는 대신 더 작은 크기의 8비트 내부 버퍼를 사용하였기 때문에 회로의 크기 면에서 훨씬 더 효율적이다. 가지 메트릭을 저장하기 위해 ROM을 사용하는 경우 BMU 회로의 크기는 약 400 게이트 정도 증가한다.

비터비 디코더 전체의 속도를 좌우하는 ACS 회로의 속도를 향상시키기 위하여 본 논문에서는 효율적인 데이터 재배열과 메모리 사용을 통해 회로를 설계하였다. 제안하는 효율적인 데이터 재배열 방식은 한 번에 2개의 데이터를 동시에 처리할 수 있기 때문에, 메모리에 데이터를 읽고 쓰는 시간을 단축시킬 수 있으므로 기존의 방식보다 처리 시간을 1/4로 감소시킬 수 있다. 또한 데이터를 동시에 읽고 쓸 수 없는 메모리를 사용하는 경우에 발생하는 지연을 없애기 위해 별도의 내부 레지스터를 사용하였으므로, 모든 연산을 지연 없이 연속적으로 처리할 수 있다. 표 2와 3은 본 논문에서 제안하는 ACS 회로의 성능을 다른 연구들[5, 8]과 비교한 것이다.

회로의 크기 면에서 효율적인 비터비 디코더의 새로운 구조를 제안한 연구[5]에서는 일반적인 병렬 ACS 회로보다 더 적은 수의 ACS 회로를 사용하였다. 표 2는 ACS 회로를 1개만 사용한 경우에 한 단을 처리하는데 필요한 사이클 수를 제안하는 회로와 비교하여 나타낸 것이다. 다른 연구에서는 하나의 ACS 연산에서 2개의 경로 메트릭을 필요로 하기 때문에 듀얼 포트 메모리를 사용하였지만, 본 논문에서는 동일한 크기의 메모리를 단일 포트 메모리 2개로 나누어서 사용하였다. 결과적으로 동일한 크기의 메모리를 사용하면서도 본 논문에서 제안하는 ACS 회로의 처리 속도는 효율적인 데이터 재배열과 메모리 사용으로 인해 기존의 연구[5]의 4배로 향상되었다.

표 2. 제안하는 ACS 회로의 성능 비교(1)
Table 2. Performance comparison of the proposed ACS circuit(1).

		제한 길이(K)			
		3	4	5	7
사이클 수	연구[5]	8	16	32	128
	본 논문	2	4	8	32

표 3. 제안하는 ACS 회로의 성능 비교(2)
Table 3. Performance comparison of the proposed ACS circuit(2).

	연구[8]	본 논문
제한 길이(K)	7	7
ACS 회로 개수	4	1
메모리 개수	듀얼 포트 메모리 8개	단일 포트 메모리 2개
사이클 수	8	32

ACS 회로 4개를 파이프라인 구조에 적용하여 비터비 디코더의 속도를 높이기 위한 방법도 다른 연구에서 제안되었다[8]. 4개의 ACS 회로를 파이프라인 구조에 적용하기 위하여 듀얼 포트 메모리를 8개 사용하여 경로 메트릭을 저장하였으며, K = 7인 경우 64개 상태의 데이터를 각각 따로 처리하여 메모리에 저장하였다. 따라서 하나의 단을 처리하는데 8사이클이 소요된다. 표 3은 본 논문에서 제안하는 회로의 성능을 비교한 것이다.

다른 연구[8]에서는 ACS 회로를 4개 사용하였기 때문에 1개를 사용한 것보다 4배의 성능 향상을 얻는다. 하지만 본 논문에서 제안하는 방법으로 ACS 회로를 4개 사용하면 23 = 8배의 성능 향상을 가져올 수 있으므로, 하나의 단을 처리하는데 4사이클이 소요되고 이는 기존

표 4. 모듈별 메모리의 크기
Table 4. Memory size in each module.

	메모리 크기(bits)
BMU	-
ACS	840
REU	5,040
비터비 디코더	5,880

연구[8]의 처리 속도의 2배이다. 또한 본 논문에서 제안하는 방식에 다른 연구와 같이 듀얼 포트 메모리를 8개 사용하는 방식을 취해도 하나의 단을 처리하는데 4사이클이 소요되어 2배의 처리 속도를 보인다. 만일 다른 연구와 같이 ACS 회로를 4개 사용하고 듀얼 포트 메모리도 8개 사용하면 4배의 성능 향상을 가져올 수 있다. 따라서 본 논문에서 제안하는 ACS 회로는 사용되는 메모리 크기와 처리 속도 측면에서 다른 방법들보다 우수하다고 할 수 있다.

표 4는 본 논문에서 제안하는 고성능 비터비 디코더 회로에서 각 모듈별로 사용된 메모리의 크기를 나타내고 있다. BMU 회로는 다른 연구와 달리 별도의 메모리를 사용하지 않는다. ACS 회로에서 사용되는 메모리는 경로 메트릭을 저장하기 위해 사용된다. 경로 메트릭은 디코딩 길이가 42일 때 최대 7비트이며, 30개 버터플라이에 대한 데이터를 2개의 상태씩 동시에 저장하기 때문에 단일 포트 메모리 1개의 크기는 420(7x2x30)비트이다. 데이터를 빠른 속도로 연속적으로 처리하기 위해 단일 포트 메모리 2개를 번갈아서 사용하므로, ACS 회로에서 사용되는 메모리 전체의 크기는 840(420x2)비트이다.

REU 회로에서 사용되는 메모리는 생존 경로에 따른 디코딩 정보를 저장하기 위해 사용된다. 하나의 상태에 대해 저장해야 하는 정보는 디코딩 길이가 42이므로 42비트이다. REU 회로는 ACS 회로와 마찬가지로 30개 버터플라이에 대한 데이터를 한 번에 2개의 상태씩 동시에 처리한다. 데이터를 빠른 속도로 연속적으로 처리하기 위해 단일 포트 메모리를 2개 사용하였다. REU 회로에서 사용되는 단일 포트 메모리 1개의 크기는 2,520(42x2x30)비트이므로, 메모리 전체의 크기는 5,040(2,520x2)비트이다.

V. 결 론

비터비 디코더 알고리즘은 통신 채널을 통해 데이터를 전송할 때 발생되는 오류를 정정하기 위한 방법으로 널리 사용되고 있으며, 본 논문에서는 효율적인 데이터 재배열과 메모리 사용을 통한 고성능 비터비 디코더 회로를 제안하고 있다. 제안하는 비터비 디코더 회로는 기능에 따라 가지 메트릭을 계산하는 BMU 회로, 생존 경로를 찾는 ACS 회로, 선택된 생존 경로를 따라 최종 디코딩된 데이터를 출력하는 REU 회로로 나누어진다.

본 논문에서 제안하는 BMU 회로에서는 가지 값의 특징을 이용하였기 때문에 가지 값을 저장하기 위한 별도의 메모리를 사용하지 않는다. 또한 제안하는 ACS 회로는 효율적인 데이터 재배열과 메모리 사용을 통해 비터비 디코더 전체의 속도를 좌우하는 ACS 연산의 처리 속도를 75%까지 향상시킨다. 따라서 본 논문에서 제안하는 비터비 디코더 회로는 회로의 속도와 크기 면에서 효율적이다. 제안하는 회로는 130nm 표준 셀 라이브러리 공정을 이용하여 합성한 결과 8,858의 게이트 수를 가지며 최대 동작주파수는 130MHz이다.

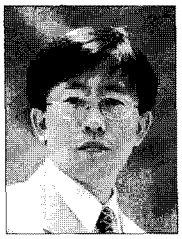
참 고 문 헌

[1] Bernard Sklar, *Digital Communications*, 2nd edition, 2006.
[2] P. Sweeney, *Error Control Coding*, Prentice-hall, 1991.
[3] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. on Information Theory, vol.13, issue 2, pp.260-269, April 1967.
[4] A. Viterbi, "Convolutional Codes and Their Performance in communication Systems," IEEE Trans. On Communication Technology, vol.19, issue 5, pp.751-772, Oct. 1971.
[5] S. Saleem and S. A. Khan, "Design and Tradeoff Analysis of an Area Efficient Viterbi Decoder," IEEE International Multitoplic Conference, pp. 1-5, Dec. 2005.
[6] C. Tsui, R. S-K. Cheng and C. Ling, "Low Power ACS Unit Design for the Viterbi Decoder," IEEE International Symposium on Circuits and Systems, vol. 1, pp.137-140, June 1999.
[7] F. Ghanipour and A. R. Nabaavi, "Design of a Low-power Viterbi Decoder for Wireless

Communications," IEEE International Conference on Electronics, Circuits and Systems, vol.1, pp.304-307, Dec. 2003.
[8] Y. Zhu and M. Benaissa, "Reconfigurable Viterbi Decoding Using a New ACS Pipelining Technique," IEEE International Conference on Application-Specific Systems, Architectures, and Processors, pp. 360-368, June 2003.
[9] Seongjoo Lee, "An Efficient Implementation Method of Parallel Processing Viterbi Decoders for UWB Systems," IEEE International Conference on Electrical Engineering/Electronic, Computer, Telecommunications and Information Technology, pp. 512-515, May 2009.
[10] C. Cheng and K.K. Parhi, "Hardware Efficient Low-latency Architecture for High Throughput Rate Viterbi Decoders," IEEE Trans. on Circuits and Systems, pp. 1254-1258, Dec. 2008.
[11] Qing Li, Xuan-zhong Li, Han-hong Jiang and Wen-hao He, "A High-speed Viterbi Decoder," IEEE Fourth International Conference on Natural Computation, pp. 313-316, Oct. 2008.

저 자 소 개

김 수 진(학생회원)
2007년 2월 한국외국어대학교 전자공학과 학사 졸업.
2009년 2월 한국외국어대학교 전자공학과 석사 졸업.
2009년 2월~현재 한국외국어대학교 전자공학과 박사과정.
<주관심분야 : SoC 설계>



조 경 순(평생회원)
1982년 2월 서울대학교 전자공학과 학사 졸업.
1984년 2월 서울대학교 전자공학과 석사 졸업.
1988년 12월 미국 Carnegie Mellon University 전기 및 컴퓨터 공학과 박사 졸업.
1988년11월~1994년 8월 삼성전자(주) 반도체 총괄 선임, 수석 연구원.
1994년 8월~현재 한국외국어대학교 전자공학과 조교수, 부교수, 정교수.
<주관심분야 : SoC 설계>