論文2002-39SD-11-7

비터비 복호 알고리즘 처리를 위한 DSP 명령어 및 하드웨어 회로

(New DSP Instructions and their Hardware Architecture for the Viterbi Decoding Algorithm)

李 在 晟 *, 鮮 于 明 勳 **

(Jae Sung Lee and Myung Hoon Sunwoo)

요 약

본 논문은 비터비 복호(Decoding)를 DSP(Digital Signal Processor)에서 효율적이고 빠르게 구현 할 수 있는 명령어 집합 및 하드웨어 회로를 제안한다. 제안하는 하드웨어 구조는 기존의 DSP 칩에 비터비 복호 알고리즘의 연산 구조에 효율적인 명령어 및 이에 가장 적합한 연산 유닛의 배열과 데이터 패스 구조를 추가하여 비터비 복호뿐만 아니라 일반 신호 처리 알고리즘들을 구현 할 수 있다. 기존의 DSP 칩이 수십 Kbps 대의 전송률에서 비터비 복호를 수행하는 반면 본 구조는 100㎒ 동작 주파수를 갖는 DSP 칩에서 6.25 Mbps의 전송률의 비터비 복호를 수행할 수 있어 전용 비터비 프로세서에 근접한 성능을 갖는다. 따라서 본 구조는 IMT-2000의 요구 전송률인 2Mbps 환경에서도 사용 가능하다.

Abstract

This paper proposes new DSP instructions and their architecture which efficiently implements the Viterbi decoding algorithm. The proposed architecture, supporting typical signal processing functions as in existing DSP chips, consists of an array of operational units and data path structures adequate to the Viterbi algorithm. While existing DSP chips perform Viterbi decoding at the rate of about several dozen kbps, the proposed architecture can give the rate of 6.25 Mbps on 100 MHz DSP chips, which is nearly the same performance as that of custom-designed Viterbi processors. Therefore, the architecture can meet the standard of IMT-2000 having the 2Mbps data rate.

Key Words : Viterbi, DSP, 명령어, 하드웨어구조, 에러 정정

Ⅰ. 서 론

현재의 통신기술은 반도체 기술의 발달과 더불어 급속한 발전이 거듭되고 있다. 그러나, 최근 통신 시스템 개발에 있어 핵심 부품이었던 ASIC(Application-Specific IC) 기반의 칩들이 다양한 통신 표준에 대한 유연성(flexibility) 부재, 하드웨어 비용 및 개발 비용 상승, 개발 기간 장기화, 고 전력 소모 등의 문제를 나타내고 있다. 이러한 제약에 따라 여러 면에서 이점을 가질 수 있는 DSP(Digital Signal Processor)[1] 기반의 통신용 시스템 개발로 그 구현 방식이 전환되고 있다.[2]

고속 디지털 신호 처리를 위한 DSP 칩은 초기 음성 데이터의 변조, 압축 및 복원과 전파 또는 음파 탐지에 이용되었으나 최근 DSP 칩들은 디지털 가전, 초고속 정보통신 기기, 휴대 이동통신 등의 내장형 시스템

* 正會員, 韓國電子通信研究院
(ETRI)
** 正會員, 亞州大學校 電子工學部
(School of Electrical and Computer Eng., Ajou Univ.)

※ 본 연구는 과학기술부의 국가지정연구실(NRL) 사업과 반도체설계교육센터(IDEC)의 지원을 받아 이루어졌음.

接受日字:2002年4月9日, 수정완료일:2002年10月28日

(embedded system) 개발이 가속화되어 그 응용이 점차 확대되어 가고 있다.[3] 또한 시스템의 여러 기능을 하나의 칩으로 집적하는 기술인 SoC(System-on-Chip) 형태로 구현되어 ASIC의 문제점들을 상당 수준 극복하고 있다. 그러나 연산 처리에 있어 초고속 광대역 통신 알고리즘들은 방대한 연산량 및 특수한 형태의 연산을 처리할 수 있는 하드웨어 구조를 필요로 하고 있어, 기존의 단순한 연산 구조를 가지는 연산 유닛들을 대단위 병렬화하여 집적도를 높인 DSP 칩은 여러 가지 알고리즘에 범용으로 사용 할 수는 있지만 각각의 알고리즘에서 아직 전용 프로세서만큼의 성능을 얻는 것은 기대하기 힘들다.

통신용 시스템 응용에서 이러한 DSP 칩의 성능 부족 현상은 FEC(Forward Error Control) 알고리즘 처리의 경우 더욱 크게 나타난다. 현재 통신용 시스템에 널리 사용되는 오류 정정 알고리즘은 길쌈 부호와 비터비 복호 알고리즘, Reed-Solomon 부호 및 복호 알고리즘이 있다. 비터비 복호 알고리즘은 랜덤 오류(random error) 정정에 우수한 성능을 보이며 Reed-Solomon 복호 알고리즘은 연집 오류(burst error) 정정에 우수한 성능을 나타낸다. 현재까지 통신 시스템의 비터비 복호기 구현에 있어 전용 비터비 프로세서[4,5,6]를 사용하는 경우가 대부분이다. 그러나 응용 시스템마다 사양이 변하기 때문에 기존 ASIC 칩으로는 다양한 시스템 사양을 만족시키는데 한계가 있다. 따라서 본 논문에서는 여러 시스템에 이용 가능하도록 DSP 칩에서 비터비 복호를 효율적으로 구현할 수 있는 새로운 명령어 집합 및 하드웨어 회로를 제안한다.[7]

제안하는 명령어 집합 및 회로는 기존의 DSP 기능과 장점은 그대로 유지하면서 최소한의 회로 추가와 변환을 통해 기존 DSP 코아의 재설계 부담을 줄이고 비터비 복호 기능을 추가할 수 있는 특징을 가진다. 또한 연산 회로 부분을 IP로 구현할 경우 SoC 설계를 위한 라이브러리 형태로 다양한 통신용 DSP 칩 설계에 활용될 수 있다. 현재 프로그래머블 DSP 칩을 이용한 비터비 디코더를 구현해야 하는 필요성이 증대되고 있으며 반도체 공정 기술의 발달이 DSP 칩의 동작 속도를 빠르게 높이고 있어[8,9] 본 제안은 유용하게 활용될 수 있다.

본 논문은 다음과 같이 구성된다. 2 절에서는 비터비 복호 연산을 위한 기존 DSP 칩들의 구현 방법 및 하드웨어 구조를 조사하고, 3 절에서는 새로운 비터비 복호용 명령어 집합 및 하드웨어 구조를 제안하고 4 절에서 기존 DSP 들과의 성능 비교를 수행한다. 마지막으로 5 절에서 결론을 맺는다.

Ⅱ. 기존 DSP 상에서 비터비 알고리즘 구현

이 절에서는 비터비 복호기 구조 및 기존 DSP에서의 프로그램을 통한 구현 방법을 알아본다.

1. 비터비 복호기 구조

<그림 1>은 비터비 전용 ASIC 칩의 구조를 나타낸다. 비터비 복호기는 수신된 부호어와 인코더의 쉬프트 레지스터 상태에 따라 발생하는 부호어와의 차이를 계산하는 BMG(Branch Metric Generation) 블럭, BMG로부터 발생된 BM(Branch Metric) 값에 현재의 상태 값을 더한 뒤 두 값을 비교해 작은 값을 선택하는 ACS (Add- Compare-Select) 연산 블럭, 마지막으로 ACS로부터 선택된 최소거리를 갖는 경로들을 저장하고 디코딩 하는 생존자 경로 메모리 블럭의 3 블럭으로 크게 구성된다.[10]

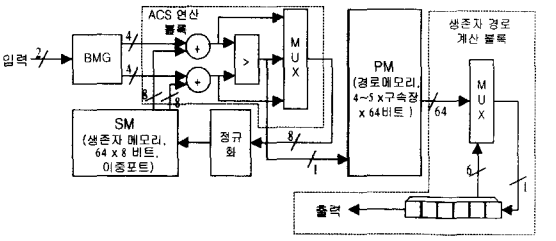


그림 1. ASIC 기반의 비터비 복호기 구조
Fig. 1. The Viterbi decoder architecture based on ASIC.

2. 기존 DSP에서의 비터비 복호기 구현

<그림 2>는 비터비 가속을 지원하지 않는 범용 DSP 칩에서 ACS 연산 수행 과정을 도시하고 있다. 일반적으로 ACS 연산은 한 쌍의 산술연산, 비교, 선택 및 저장의 과정을 거쳐야 한다. 즉, 과정 ①에서 두 쌍의 PM과 BM을 더하며,[11] 과정 ②에서 비교를 수행하여 플래그 레지스터에 그 신호를 저장한다. 과정 ③에서는 비교 결과 신호 1 비트를 메모리에 저장하고, ④에서는 비교시 결정된 최소 거리의 값을 쉬프트시켜 메모리에 저장한다.[12,13] 그러나, 이러한 일련의 과정들은 DSP 칩이 보유하고 있는 ALU 및 레지스터 파일의 사양에 따

라 각각 여러 클록 싸이클이 걸릴 수 있다. 또한 ③과 같은 경우는 비교 결과인 선택 신호를 메모리에 저장하기 전에 모든 상태에 대한 ACS 연산 결과의 선택 신호들을 쉬프트 하여 정렬시킨 후 워드 단위로 메모리에 저장하여야 하기 때문에 여러 싸이클이 걸릴수 있다.

DSP 칩 개발사들은 비터비 알고리즘 연산의 고속화 경향에 맞추어 새로운 명령어 구조들을 선보이고 있으나 비터비 알고리즘 중의 일부 연산을 수행하기 위한 국한된 명령어들만을 지원하고 있어 수행 속도를 크게 향상시키지는 못하고 있다. 비터비 디코딩의 가속을 지원하는 상용 DSP 칩들로는 DSP Group사의 OakDSP,[14] Texas Instrument 사의 TMS320C55x,[15] Motorola 사의 DSP56600 패밀리가[16] 있으며, 이들 모두 ACS 연산 부분만을 가속하고 있다.

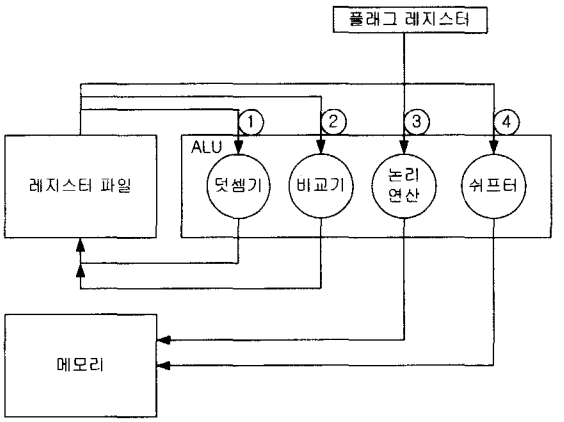


그림 2. 기존 DSP 칩에서 ACS 연산 수행 과정
Fig. 2. The ACS operation flow on the existing DSP chips.

비터비 복호화를 위해 특정 사양을 가지는 상용 DSP의 명령어는 단지 2쌍의 데이터를 동시 비교할 수 있는 비교문, 즉 OakDSP의 MIN, TMS320C55x의 MAXDIFF, DSP56600의 MAX만을 지원하고 있다.[14,15,17] 즉, 비교 이전의 산술 연산과 비교 연산 이후 최소 거리의 선택 및 저장 과정은 여러 개의 일반 DSP 명령어를 사용하여 처리함으로써 많은 클록 싸이클을 소모하고 있다. 따라서, <그림 2>의 과정 ②, ③에서 걸리는 싸이클만을 다소 감소시키고 있다.

또한, 생존자 경로 저장시 1 비트를 쉬프트 시켜서 저장하는 DSP56600의 VSL 명령어[16]는 비교를 수행하는 MAX 명령어와 파이프 라이닝하여 처리하면 데이터 의존성이 없어 한 클록 싸이클에 동시 수행하면 효율적이나 하드웨어 구조가 뒷받침되지 않아 각각 다른 싸이클에 계산을 하고 있어 VSL 명령어의 장점을 살리지 못하고 있다. 특히, 범용 DSP 칩에서 가장 처리하기 까다로운 생존자 경로 계산을 위한 특정 명령어는 전혀 지원되고 있지 않고 있다. 즉, ACS 연산을 빨리 수행한다 하더라도 데이터가 복원되어 나오는 비트율은 생존자 경로 계산을 얼마나 빠르게 처리하느냐에 달린 것인데 이를 위한 연산 구조는 지원되지 않아 매우 느린 복원 비트율을 가질 수 밖에 없었다. 따라서 앞서 살펴본 종래 DSP 칩들은 음성서비스와 같은 15Kbps 미만의 저속 데이터 통신에 국한되어 사용되어 왔던 것이다.

Ⅲ. 새로운 비터비 복호용 명령어 및 하드웨어 구조 제안

본 절에서는 비터비 디코딩 알고리즘 구현을 위한 2가지 명령어와 그에 따른 각각의 하드웨어 구조를 제안하며,[7] 명령어는 <그림 1>에 나타난 전용 비터비 복호기의 연산 구조와 유사한 방식으로 복호 알고리즘을 처리한다. 즉, 본 논문이 제안하는 DSP 칩용 명령어 집합은 ASIC으로 구현한 전용 비터비 디코더의 구조와 유사한 구조를 가지고 있어 대등한 수행 성능을 보일 수 있다. 제안하는 명령어들은 PACS(Packed Add-Compare-Select)와 SLBIT(Shift Left BIT) 이며 단지 2 개의 명령어들로 비터비 디코딩 알고리즘의 대부분을 구현할 수 있다. 즉, <그림 1>의 ACS 연산 블록과 생존자 경로 계산 블록을 위한 각각 하나씩의 명령어를 한 클록 싸이클에 동시 수행하도록 하여 전용 비터비 구조의 성능과 대등하도록 하였다.

참고로 BMG 연산 블록은 단순한 산술 연산을 수행하므로[18] 특별한 명령어나 하드웨어를 사용하지 않아도 기존 DSP 칩들의 일반적인 구조의 ALU를 사용하여 쉽게 구현이 가능하기 때문에 BMG 알고리즘 블록을 위한 명령어는 고려하지 않아도 된다.

1. PACS(Packed Add-Compare-Select) 명령어

PACS 명령어는 비터비 디코딩 알고리즘의 ACS 블록을 하나의 명령어로 처리하는 것으로써 <그림 3>에 도시하였다. <그림 3>은 32 비트 DSP 칩인 경우 1 개의 ALU 안에 있는 연산 유닛들을 이용하여 <그림 2>

에 도시한 PE 구조 2개를 구현한 것이다. 연산 유닛은 기존의 것들을 사용하여 구현할 수 있고, 단지 파이프라이닝에 의하여 덧셈과 비교를 연속적으로 수행할 수 있도록 데이터 패스 회로만 추가하여 구현 가능하다. <그림 3>은 레지스터 파일이 바이트 단위로 억세스 가능하다는 전제로 하는 구조이며 (최신 DSP 칩들은 모두 바이트 단위 억세스가 가능함), 2개의 32 비트 워드 데이터를 바이트 단위로 처리하여 2개의 ACS PE 역할을 수행하는 구조이다. 이러한 명령어 처리 구조는 2 싸이클마다 4개의 ACS 연산 결과를 출력할 수 있다. 비교 결과 선택된 값인 생존자는 SM(생존자 메모리)에 저장되며 생존자를 선택하는 선택 신호는 PM(경로 메모리)에 저장된다.

<그림 3>의 동작 설명은 다음과 같다.

- a ~ d : 8 비트 레지스터의 소스 오퍼런드 (BM 값들이 들어 있음)
- a′ ~ d′ : 또 다른 8 비트 레지스터의 소스 오퍼런드 (PM 값들이 들어 있음)
- 1 ~ 4 : Execution stage1(Ex1)에서 PM + BM 연산의 결과가 저장되는 파이프라인 레지스터,
- >/s : 비교 후 작은 값을 선택하고 선택 신호 1 비트 발생하는 기능 수행
  (선택결과와 선택신호를 출력으로 내보낸다.)
- 선택결과(SM으로 가는 값들) : 쉬프트1, 2에서 PM + BM 값이 9 비트이며 연속적인 덧셈에 의해 발생하는 오버플로우를 방지하기 위해 정규화 수행(오른쪽으로 1 비트 쉬프트)
- 선택신호(PM으로 가는 값들) : 하나의 큰 64 비트 쉬프트 레지스터에 저장하며, 이는 한 클록 싸이클마다 PE 개수 만큼씩 왼쪽으로쉬프트(<그림 3>의 경우는 2비트씩)

ACS 연산은 서로간에 데이터 의존성이 없으므로 이러한 구조는 여러 개의 ALU를 갖는 DSP 칩이나 32 비트, 64 비트, 128 비트 DSP 칩 등에서도 <그림 3>의 기본구조를 단순히 병렬적으로 확장하면 고속 병렬 처리가 가능하다. 예로써, 64 비트 칩인 경우는 ALU 하나로 ACS PE를 4 개, 128 비트 칩인 경우는 ACS PE 8 개를 구성할 수 있다. 또한, ALU가 2 개 이상이면 결국 ACS PE 수도 2 배 이상으로 늘어나는 것이다.

한 쌍의 ACS 버터플라이 연산 구조는 4개의 ACS PE를 필요로 한다. 따라서 <그림 2>에서 설명한 상용

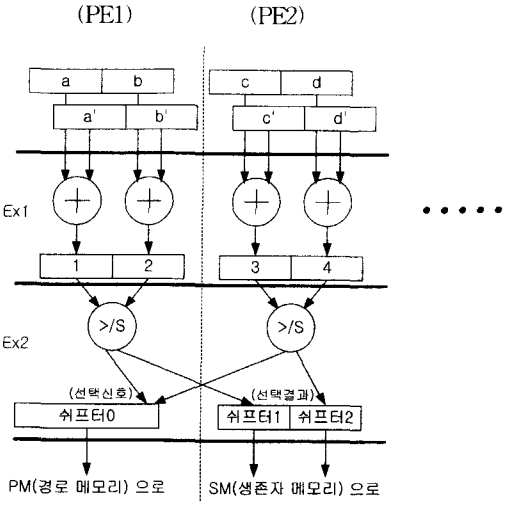


그림 3. PACS 명령어 수행 흐름도
Fig. 3. The operation flow diagram of the PACS instruction

DSP의 ACS 연산 명령어들을 PACS 명령어 두개로 처리할 수 있어 상용 DSP 보다 수행 싸이클 면에서 3~5 배 이상 빠르게 ACS 연산 블록을 처리할 수 있다. 앞의 <그림 2>에서 나타난 기존 DSP 칩의 ALU 구조 처리 방식은 ALU 안에 여러 가지 기능을 하는 연산 유닛들이 각기 연산을 취하는 방식이었으나 그 연산 유닛들간에 데이터 흐름을 위한 경로를 놓아주면 <그림 3>의 연산 흐름이 가능하다. 다시 말하면, 본 구조는 기존 DSP의 연산 유닛들에 연속 연산(한 쌍의 덧셈 연산 직후 비교 연산)이 가능하도록 하는 데이터 패스 구조와 2(K-1) 비트의 쉬프트 레지스터(쉬프터0 : 선택신호 저장) 만을 추가하여 구현 가능하므로 하드웨어 부담이 적고, 다른 DSP 알고리즘의 연산 수행도 원활히 수행하도록 할 수 있다.

2. SLBIT (Shift Left BIT) 명령어

SLBIT 명령어는 <그림 1>의 역추적 방식 생존자 경로 계산 회로를 위한 출력 레지스터를 그대로 구현한 것으로 <그림 4>에 동작도를 나타내고 있다. 즉, 구속장 K = 7 인 경우 ACS 연산 결과 생성된 선택 신호 1 비트들이 모여 2(7-1)= 64 비트 쉬프트 레지스터를 채우면 64비트의 데이터는 경로 메모리(PM) 역할을 하는 레지스터 파일에 저장되고 그 저장된 값들을 입력(소스 오퍼런드)으로 하여 <그림 4>의 연산을 수행한다. 데스티네이션 하위 6 비트의 조합만으로 26 = 64 비트를 비트 단위로 선택 가능하므로 소스 오퍼런드의

1 비트를 선택하고 그 1 비트가 데스티네이션의 LSB(Least Significant Bit)로 들어오면서 왼쪽으로 쉬프트되어 새로운 6비트를 구성하고 이 6 비트를 가지고 같은 방식의 연산을 반복 수행한다. 반복 연산은 PM의 처음(0 번지)부터 끝(K의 4~5 배 번지)까지 수행하여야 하며 이렇게 전체 PM을 트레이스 백하면 1 비트의 복호화 결과가 출력된다. 이것이 복호화된 데이터이다.

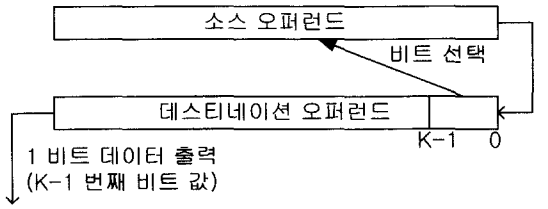


그림 4. SLBIT 명령어의 동작도
Fig. 4. The operation flow diagram of the SLBIT instruction.

<그림 4>의 동작설명은 다음과 같다.

- 데스티네이션 오퍼런드의 하위 K 비트가 가리키는 소스 오퍼런드의 비트를 선택
- 데스티네이션 오퍼런드의 원래 값을 1 비트 쉬프트
- 선택된 비트를 데스티네이션의 LSB로 복사

<그림 4>의 동작을 수행하기 위해서는 기존 DSP의 레지스터 파일에 단순히 2(K-1)×1 멀티플렉서(MUX)만 추가되면 구현 가능하다. 선택 신호들이 저장되는 레지스터 파일 안의 레지스터들은 2(K-1) 비트 이상 단위로 억세스 가능한 구조를 가져야 한다(K=7 인 경우 32 비트 칩에서 레지스터 2개를 연결하여 64 비트 단위로 억세스 가능하여야 함). 일반적으로 구속장 K가 7인 경우를 가장 많이 사용하나 7이 아닌 경우에도 사용이 용이하도록 K 값을 여유있게 선택하여 하드웨어를 구성하면 유리하다.

기존 DSP로는 쉬프트와 인덱스 어드레싱으로 데이터를 이동 시켜 약 5 싸이클에 거쳐 처리를 해야 하는 것을 1 싸이클로 줄일 수 있다. 그러나, 1 비트의 디코딩 데이터를 산출하기 위해서는 구속장 K가 7인 경우 일반적으로 4~5배 즉, 28~35개의 2(7-1) = 64 비트 PM 값들을 전부 역추적 해야 하기 때문에 앞서 아무리 ACS 연산을 빨리 수행한다 하더라도 고속 연산을 수행하기 어렵다. 기존 DSP에서는 PM 값들을 내부 메모리에 저장 시켜 놓고 임시 레지스터에 1 개씩 옮겨가면서 역추적 연산을 수행하고 메모리로부터 한번에 64 비트씩 레지스터로 이동하는 것 또한 불가능하기 때문에 많은 수행 싸이클이 소모된다. 예로써 메모리 억세스 싸이클이 3 이고 메모리 데이터의 비트 수가 8 비트인 경우에는 대략 8×3 + 5 = 29 싸이클이 걸린 셈이다. 따라서, PM 값들을 억세스 싸이클이 1인 레지스터 파일을 사용하여 처리해야 하며, 레지스터 파일에 64 비트 레지스터 개수가 28~35개 미만인 경우 내부 메모리와 레지스터 파일간의 데이터도 병렬 이동(parallel move)하는 구조를 반드시 가져야 한다.

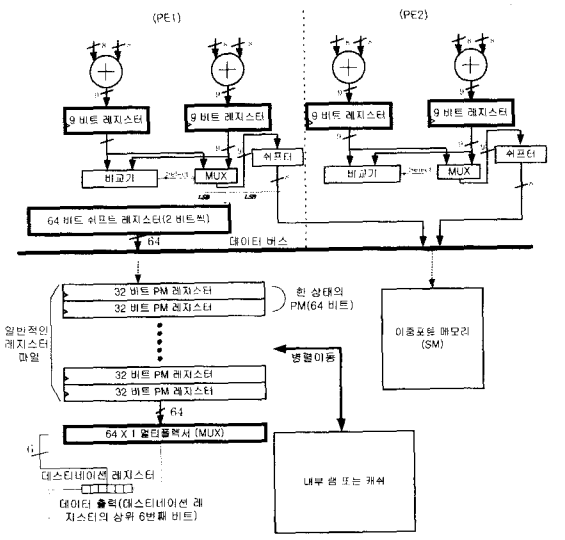
3. 비터비용 명령어 처리를 위한 하드웨어 회로 구조

<그림 5>에서 앞서 설명한 PACS와 SLBIT 명령어들을 효율적으로 처리하기 위한 하드웨어 회로 구조를 나타내고 있다. 본 그림의 데이터 버스의 상단부 구조는 PACS 명령어 처리를 위한 세부 회로이다. <그림 1>의 ACS 연산 구조를 사용한 것이며 2개의 PE를 가진다. 따라서 32 비트 DSP의 ALU 1 개로 구현 가능하다. 본 구조는 구속장 K가 7인 경우를 임의로 나타낸 것이며, ACS 연산부는 PACS 명령어에서 설명한 바와 같이 병렬로 확장한다면 더 빠른 ACS 연산이 가능하다. <그림 5>에서 굵은 박스로 나타낸 9비트 레지스터, 64비트 쉬프트레지스터, 64 x 1 멀티플렉서는 비터비 연산을 위해 기존 DSP 구조에 추가된 회로이다. 이들 블록을 제외한 나머지 연산 블록은 기존 DSP 구조를 그대로 사용 가능하므로 설계 재사용의 측면에서 장점을 지닌다.

첫번째 클럭 싸이클에서 한 쌍의 BM과 SM으로부터 한 쌍의 8비트의 데이터가 들어와 더해짐으로써 두 경로의 현재 스테이지까지의 PM을 계산한다. 두 번째 클럭 싸이클에서 두 쌍의 PM 값들의 크기를 비교하여 그 결과인 각각의 1 비트 신호를 64비트 레지스터의 LSB에 저장하고 작은 것으로 판명된 생존자를 정규화하기 위하여 1 비트 오른쪽으로 쉬프트하여 SM에 저장한다.

<그림 5>에서는 두 개의 ACS 연산을 독립적으로 처리하여 나온 두 개의 작은 값들을 데이터 버스를 통해 이중 포트 메모리에 저장되는 구조를 가지고 있으며, 이러한 이중 포트 메모리가 생존자를 저장하는 SM 역할을 수행한다. 선택 신호 2 비트가 저장되는 64 비트 쉬프트 레지스터는 매 클록마다 2 비트씩 왼쪽으로

쉬프트하여 64 비트를 채울 때까지 동작한다. <그림 5>와 같이 ACS PE가 2개인 경우 2 비트 단위로 쉬프트하여(ACS PE의 개수 단위로 쉬프트) 64 비트가 차게 되면 레지스터 파일의 32 비트 레지스터 두 개에 나누어 저장되어 진다. 저장된 PM은 그림의 왼쪽 하단부에 나타난 SLBIT 명령어의 하드웨어 구조를 통해 64 비트 데이터 중 1 비트가 64x1 멀티플렉서에서 선택되어 데스티네이션 레지스터에 LSB로 삽입되고 이 레지스터는 동시에 왼쪽으로 1 비트 쉬프트된다. 이러한 연산을 반복하여 PM의 마지막 데이터까지 처리되면 이때 데스티네이션 레지스터의 7-1= 6 번째 비트 값이 최종 디코딩된 1 비트 값이 된다.



* 굵은 선으로 표시된 박스는 기존 DSP구조에 추가되는 회로 블록

그림 5. PACS 및 SLBIT 명령어를 위한 하드웨어 구조(K=7 까지 고려한 경우)

Fig. 5. The hardware architecture for PACS and SLBIT instructions.

PACS 명령어 하드웨어 구조는 처리 데이터가 8 비트 단위로 억세스만 가능하면 DSP 칩의 비트 수에 상관없이 구현이 가능하고, DSP 칩의 처리 비트 단위가 클 경우 즉, 64 비트 머신이나 128 비트 머신인 경우 8 비트 PE의 수가 늘어나므로 특별한 회로의 추가 없이 그 만큼의 병렬 처리 효과를 가져올 수 있다. 그러나 반드시 생존자 경로 계산의 수행 싸이클과 조화를 이루어야지만 고속의 동작이 가능하다. 앞서 설명한 바와 같이 ACS가 아무리 빨리 처리된다 하더라도 생존자 경로 계산이 늦어지면 고속의 ACS 연산은 성능을 발휘할 수 없다.

따라서, 생존자 경로 계산은 레지스터 파일과 같은 1 싸이클 억세스가 가능한 저장매체가 필요하며, <그림 5>의 하단부에 나타난 바와 같이 레지스터 파일이 전체 PM 메모리 역할을 하기에 공간이 부족한 경우는 내부 램 또는 캐쉬 메모리와 레지스터 파일간의 병렬 이동이 용이한 구조를 가져야 병목 현상을 없앨 수 있다. 또한, <그림 5>의 맨 하단에 생존자 경로 계산을 하기 위한 6 비트의 쉬프트 레지스터는 일반 쉬프트 레지스터를 사용하여도 되며, 반드시 ACS 연산과 병렬적으로 처리되어야 고속 수행이 가능하다.

구속장 K가 7인 경우 64 개의 상태가 존재하므로 제안한 구조에서 ACS PE가 2개이므로 32 싸이클이 걸린다. 그리고, 동시에 선택 신호들은 2 비트씩 왼쪽으로 쉬프트 되어 64 비트 쉬프트 레지스터를 가득 채우게 된다. 가득 찬 64 비트 값을 레지스터 파일내에 있는 2 개의 32 비트 레지스터에 저장이 되고, PACS 연산은 같은 순서로 계속된다. 한편 레지스터 파일이 PM의 깊이인 K의 4~5배의 공간 즉, 28~35개의 레지스터를 갖거나 메모리와 병렬 데이터 이동이 가능한 구조를 가지면, SLBIT 연산은 28~35의 싸이클 만에 처리가 되고, 그 결과 1 비트의 데이터가 복호화되어 출력된다. 따라서, PACS와 SLBIT는 연산이 독립적이고 수행 싸이클이 거의 비슷하므로 TMS320C54x의 경우처럼 병렬 연산을 의미하는 심볼 '||' 을 사용하여 PACS || SLBIT의 형태로 병렬 사용이 가능하다.

쉬프터는 일반 산술연산유닛에 존재하는 쉬프터에 데이터 버스로 가는 경로를 추가하여 구현할 수 있다. 멀티플랙서는 6비트의 선택신호를 받아 경로 메모리의 데이터를 가지고 있는 PM 레지스터의 64 비트 데이터 중 1 비트를 선택해 이를 출력으로 내보낸다. 그러면, 아래의 6 비트 레지스터의 하위 비트로 삽입되고 그 레지스터의 데이터가 전체적으로 왼쪽으로 쉬프트를 하게 되면서 역추적 과정을 수행하게 되는 것이다. 즉, 약 32 싸이클마다 1 비트 데이터를 출력하므로 100 Mhz의 동작 주파수를 가지는 DSP 칩(100MIPS로 가정)인 경우라도 100,000,000/32 = 3.125 Mbps의 복호화 성능을 낼 수 있으며, 부호율이 1/2이므로 6.25 Mbps의 고속 전송률 갖는 통신 환경에서도 사용 가능하다.

이 구조는 상용 DSP의 일반적인 명령어들을 처리할 수 있도록 하는 연산 유닛의 배열을 유지하면서 빠른

비터비 복호를 가능케 하는 구조를 갖는다는데 더 큰 이점이 있다.

Ⅳ. 성능 평가

<표 1>은 구속장 K = 7 인 경우의 기존 DSP 칩과의 비터비 디코딩 수행 싸이클 비교표이다. ACS 연산의 경우 상용 DSP들의 벤치마크 프로그램에서 ACS 버터 플라이 연산 프로그램과 수행 싸이클만 설명하고 있다. 따라서 실제 64/4 = 16쌍의 버터플라이 연산을 반복 수행하면서 루프문 동작시 결과 데이터 저장 및 분기 등으로 소모되는 데이터 지연 싸이클 수는 α로 표기하였으며, 대략 버터플라이 연산 수행 싸이클의 1/3정도를 더 소모한다.

표 1. 구속장 K=7 인 경우의 수행 싸이클 비교 (생존자 메모리 깊이는 K의 5 배로 계산)

Table 1. The comparision of DSP performances at K=7.

구조 / 연산 블록	OakDSP[1,4]	TMS320C55x	DSP56600	제안한 구조
ACS 연산 블록	약 152+α	약 96+α	약 128+α	32
생존자 경로 계산	약 910 (= 26×7×5)			35
전체	약 1062+α	약 1006+α	약 1038+α	35

또한, 기존 DSP 칩에서는 ACS 연산과 생존자 경로 계산을 따로 수행하였으나 본 논문에서는 ACS 와 생존자 경로 계산의 동시 수행이 가능하므로 전체적으로 약 30 배 이상의 성능 향상을 보일 수 있다. 제안한 비터비 전용 DSP 명령어 하드웨어 구조는 <그림 5>에서 굵은 박스로 나타낸 9 비트 레지스터, 2(K-1) 비트 쉬프트 레지스터, 2(K-1)×1 멀티플렉서들만 제외하면 기존 DSP 에서 널리 사용되는 연산 유닛들로 구성되어 있다. 따라서, 칩 제작시 기존 연산 모듈들 대부분이 재사용이 가능하고 연속적인 연산이 가능하도록 하는 파이프라이닝 데이터 패스 회로들만 추가하면 되므로 설계 비용 측면에서도 경제적이다.

또한, ACS 연산 블록의 경우 같은 구조를 단순히 반복하여 쉽게 병렬로 확장이 가능하며 생존자 경로 계산의 경우 K 값을 넉넉히 고려할 경우 단순히 2(K-1)×1 멀티플렉서 비트 수만 좀더 커지고, 한 상태의 PM 값을 저장하기 위해 서로 결합되는 레지스터들의 수(32 비트 칩에서 K = 7인 경우 2개, K = 8 인 경우 4 개, K = 9 인 경우 8 개)만 늘어나면 되는 매우 유연한 구조이다.

Ⅴ. 결 론

본 논문에서는 통신용 에러 정정을 위해 가장 많이 쓰이는 알고리즘 중에 하나인 비터비 복호를 DSP (Digital Signal Processor)에서 효율적으로 구현할 수 있도록 하는 새로운 명령어 집합 및 하드웨어 회로를 제안하였다. 제안하는 하드웨어 구조는 기존 DSP의 범용성을 유지하면서 비터비 복호 알고리즘의 연산 구조에 가장 적합한 연산 유닛의 배열과 데이터 패스 구조를 제공하여 전용 비터비 프로세서에 근접한 성능을 가지도록 하였다. 기존 DSP 기술에서 15 Kbps 대의 전송률을 보이던 것을 본 논문이 제안하는 구조를 가지고 100 ㎒ 동작 주파수를 갖는 DSP 칩에서 동작한다면 6.25 Mbps의 전송률을 제공할 수 있어 IMT-2000과 같은 무선 멀티미디어 서비스의 요구 전송률인 2 Mbps 환경에서도 사용 가능하다. 실제로 최근 DSP 공정 기술은 수백 ㎒ 이상의 동작을 가능하게 하고 있으며, 레지스터 파일의 집적도도 매우 커지고 있어, 앞으로 반도체 공정 기술이 더욱 발달한다면 처리 가능 전송률은 더욱 높아질 수 있다. 또한 본 논문에서 제안한 구조들은 병렬적으로 확장이 가능하므로 수 ~ 수십 Mbps까지 가능할 것이다. 따라서, 현재 음성 데이터 전송에 국한된 DSP를 이용한 비터비 디코더 기술을 LMDS(Local Multipoint Distribution Systems) 서비스나 MCNS(Multimedia Cable Network System) 등의 표준안을 만족하는 CATV나 VOD 등의 멀티미디어 통신 서비스의 에러 정정용으로 사용할 수 있다.

참 고 문 헌

[1] J. Glossner, J. Moreno, M. Moudgill, J. Derby, E. Hokenek, D. Meltzer, U. Shvadron, and M. Ware, "Trends in compilable DSP Architecture," in *Proc. Workshop on SiGNAL*

Processing Systems (SiPS), 2000, pp. 181~199.

[2] J. Eyre and J. Bier, "DSP processors hit the mainstream," *IEEE Trans. Comput.*, Vol. 31, pp. 51~59, Aug. 1998.

[3] A. Gupte, M. Mehendale, R. Ramamritham, and D. Nair, "Performance considerations in embedded DSP based system-on-a-chip designs," in *Proc. 17th Int. Conf. VLSI Design*, 2001, pp. 36~41.

[4] Y. Chang, H. Suzuki, and K. Parhi, "A 2-Mb/s 256-state 10-mW rate-1/3 Viterbi decoder," *IEEE J. Solid-State Circuits*, Vol. 35, pp. 826~834, June 2000.

[5] M. Hara, T. Yoshinaka, Y. Sugizaki, and S. Ohura, "A high speed viterbi decoder using path limited PRML method and its application to 1/2 inch HD full bit rate digital VCR," *IEEE Trans. Consumer Electron.*, Vol. 47, pp. 80~86, Feb. 2001.

[6] L. Inkyu and J. Sonntag, "A new architecture for the fast Viterbi algorithm," in *Proc. Global Telecommun. Conf. (Globecomm)*, Vol. 3, 2000, pp. 1664~1668.

[7] 선우 명훈, 이 재성, "프로그래머블 프로세서에서의 비터비 디코딩 연산방법 및 그 연산방법을 실행하기 위한 연산회로," 출원번호 제 10-2001-0043712호.

[8] K. Guixia and Z. Ping, "The implementation of Viterbi decoder on TMS320C6201 DSP in W-CDMA system," in *Proc. Int. Conf. Commun. Tech. (ICCT)*, Vol. 2, 2000, pp. 1693~1696.

[9] K. S. Ahmad, M. M. Saqib, and S. Ahmed, "Parallel Viterbi algorithm for a VLIW DSP," in *Proc. Int. Conf. Acoust., Speech and Signal Processing (ICASSP)*, Vol. 3, 2000, pp. 1555~1558.

[10] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Engle wood Cliffs, NJ: Prentice-Hall, 1995.

[11] C. B. Shung, H.-D. Lin, R. Cypher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for the Viterbi algorithm. I. Theory," *IEEE Trans. Commun.*, Vol. 41, pp. 636~643, Apr. 1993.

[12] H. D. Lin and D. G. Messerschmitt, "Parallel Viterbi decoding methods for uncontrollable and controllable sources," *IEEE Trans. Commun.*, Vol. 41, pp. 62~69, Jan. 1993.

[13] M. Boo, F. Arguello, J. D. Bruguera, R. Doallo, and E. L. Zapata, "High-performance VLSI architecture for the Viterbi algorithm," *IEEE Trans. Commun.*, Vol. 45, pp. 168~176, Feb. 1997.

[14] O. B. Sheva, W. Gideon, and B. Eran. (1999). The OakDSPCore's Viterbi accelerator speeds up digital communications. Smartcores Articles. [Online]. Available: http://www.dspg.com.

[15] M. A. Chishtie, "Viterbi implementation on the TMS320C5x for V.32 modems," Texas Instruments Inc., Dallas, TX, Rep. SPRA099, Apr. 1996.

[16] D. Taipale, "Implementing Viterbi decoders using the VSL instruction on DSP families DSP56300 and DSP56600," Motorola Inc., Denver, CO, Tech. Doc. APR40/D, May 1998.

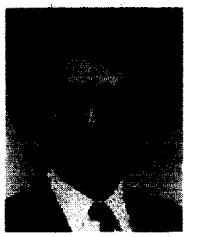
[17] Texas Instruments Inc., *TMS320C55 DSP Programmer's Guide*, 2000.

[18] S. Kubota, S. Kato, and T. Ishitani, "Novel Viterbi decoder VLSI implementation and its performance," *IEEE Trans. Commun.*, Vol. 41, pp. 1170~1178, Aug. 1993.

저 자 소 개

李 在 晟(正會員)

1999년 2월 아주대학교 전자공학부 학사. 2001년 2월 아주대학교 전자공학과 석사. 2001년~현재 한국전자통신연구소(ETRI) 연구원. <주관심분야 : VLSI Architecture, 병렬처리 프로세서 및 DSP 칩 설계, Protocol Processing>



鮮于明勳(正會員)

1980년 서강대학교 전자공학 학사. 1982년 한국과학기술원 전기 및 전자공학 석사. 1982년~1985년 한국전자통신연구소(ETRI) 연구원. 1985년~1990년 Univ. of Texas at Austin 전기 및 컴퓨터 공학 박사. 1990년~1992년 미국 Motorola, DSP Chip Division. 2001년~현재 IEEE Senior Member. 1992년~현재 아주대학교 전자공학부 교수. <주관심분야 : VLSI 및 SoC Architecture, 멀티미디어 통신용 DSP 칩 및 ASIC 설계>