ARM RISC 상에서의 MPEG-1 Audio decoder의 실시간 구현

김 선 태
한국전자통신연구원 인터넷정보가전연구부
전화 : 042-860-1082 / 핸드폰 : 011-9068-9585

Real-Time Implementation of MPEG-1 Audio decoder on ARM RISC

Seon-Tae Kim
Dept. of Internet Information Appliance , ETRI
E-mail : stkim10@etri.re.kr

Abstract

Recently, many complex DSP (Digital Signal Processing) algorithms have being realized on RISC CPU due to good compilation, low power consumption and large memory space. But, real-time implementation of multiple DSP algorithms on RISC requires the minimum and efficient memory usage and the lower occupancy of CPU. In this thesis, the original floating-point code of MPEG-1 audio decoder is converted to the fixed-point code and then optimized to the efficient assembly code in time-consuming function in accord with RISC feature. Finally, compared with floating-point and fixed-point, about 30 and 3 times speed enhancements are achieved respectively. And 3~4 times memory spaces are spared.

I. 서론

신호처리 알고리즘이 복잡해짐에 따라 이들 신호처리 알고리즘을 실시간 구현하고자 할 때는 전용 프로세서(ASIC)나 알고리즘의 성능을 향상시키기 위한 별도의 전용 명령어를 하드웨어로 갖춘 DSP(digital signal processor)를 애용해 왔다. 하지만, 프로세서의 동작 주파수가 빨라지고 신호처리 알고리즘이 범용 프로세서로 구현이 가능해짐에 따라 컴파일러의 효율이 뛰어난 RISC 프로세서를 사용하는 경향이 늘어나고 있다.

한편, 현대는 멀티미디어 알고리즘이 많은 분야에 이용되고 있다. 특히 휴대용 시스템에서 많이 활용 되어가고 있다. 하지만 휴대용 시스템에서 구현 요소 중 중요한 것이 전력문제이고 이를 해결하기 위한 하나의 방안이 RISC CPU를 사용한 시스템의 구현이며, 특히 ARM 계열의 프로세서는 전력 소비가 다른 프로세서에 비해 뛰어나다. 또한 ARM RISC 계열은 고급 언어에 대한 컴파일러의 지원이 용이해 다른 DSP(digital signal processor)에 비해서는 시간적 구현이 빠르고 구현이 쉽다는 장점이 있다.

본 논문은 DTV 셋톱에 들어가는 MPEG-1 오디오 디코더 알고리즘을 실시간으로 구현하는 것으로써, 다른 응용 프로그램과 함께 연동하여 실시간으로 구현되기 위해서는 보다 효율적인 구현이 필요했다. 즉, 비디오 알고리즘과 함께 연동하기 위해서는 보다 낮은 프로세서 점유율을 가져야 하며, 한정된 메모리를 사용하는 내장형 시스템에서 프로그램코드나 데이터는 보다 작아야하며 버퍼크기 역시 작아야 했다. 본 논문에서는 이를 달성하기 위해 부동 소수점으로 된 MPEG-1 오디오 디코더를 고정 소수점으로 변환하고 이를 다시 RISC 머신의 특성에 맞는 기계어로 변환하여 시스템의 속도 면과 메모리 측면에서 보다 효율적

으로 구현하려고 했다. 실험결과 ARM RISC계열중의 하나인 SA110-233Mhz에서 30% 이상의 CPU 점유율을 가진 알고리즘이 제안한 방법에 따른 구현 결과로 CPU 점유율을 15%정도 차지하면서 실시간이 이루어지도록 했다. 메모리는 부동소수점에 비해서는 4배 이상의 절감효과를 고정 소수점에 비해서는 30%의 메모리를 절약할 수 있었다.

본 논문의 구성은 우선 Ⅱ장에서는 MPEG-1 오디오 부호화 알고리즘에 대해서 살펴보고, Ⅲ장에서는 RISC에서의 효율적 구현 방법 몇 가지를 제시하고, Ⅳ장에서는 구현후의 결과를 살펴보고, Ⅴ장에서는 결론을 내리면서 논문을 맺는다.

Ⅱ. MPEG-1 오디오 부호화 알고리즘

MPEG-1 오디오 알고리즘은 고품질, 고능률 스테레오 부호화를 위한 ISO/IEC의 표준방식으로써, 압축은 32 밴드에 기초한 서브밴드 코딩(대역분할 부호화)과 MDCT(modified discrete cosine transform)를 사용하

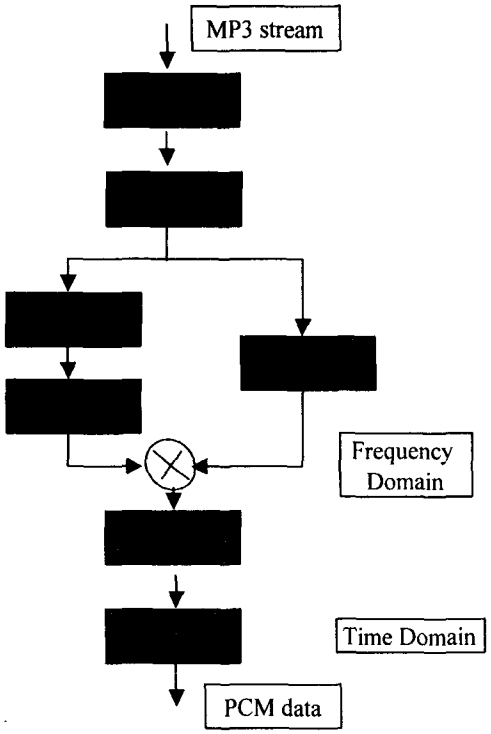

그림1. MPEG-1 오디오 디코더 구조도

는데, 청각심리적 특성을 이용해서 고능률의 압축이 실현되고 있다. 다른 알고리즘의 부호화에 비해 뛰어난 압축율을 가지므로 복호화 하는데도 그 만큼의 복잡도를 가진다.

MPEG-1 오디오는 계층 I,II,III이라는 세 가지의 모드를 가지고 있다. 높은 계층일수록 고품질과 고압축율이 실현되는 반면 프로그램의 코드가 커지고 하드웨어 규모가 커진다. 계층 I,II는 압축 알고리즘으로 주파수 대역을 32개의 밴드로 세분하여 서브밴드 부호화를 행한다. 반면, 계층 III은 그림1에서와 같이 계층 I,II의 32밴드보다 세밀하게 주파수대역을 분할하는 MDCT를 사용하며 이 MDCT에 의해서 얻어진 계수를 허프만 부호화를 한다.

본 연구에서 구현되는 MPEG-1 오디오 부호화는 MPEG-2 디지털 방송과 MPEG-1 미디어 재생을 위해 계층 2가 사용되었으며, 계층 3은 MP3 재생용으로 사용되었다.

Ⅲ. 구현 과정 및 방법

3.1 구현 과정

구현과정은 그림2와 같이 Strong Arm(SA-110) 233Mhz에서도 실시간의 3배 이상의 시간이 요구되는 MPEG-1 오디오 디코더를 먼저 구현할 ARM CPU 연산형에 맞게 고정소수점으로 알고리즘을 수정 가했다.

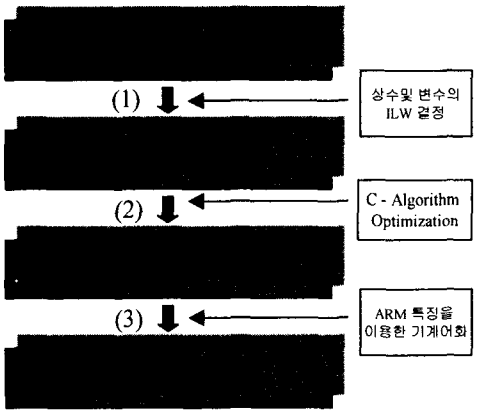

그림2. 최적화 과정

(1)단계에서는 단지 변수의 오버플로가 일어나지 않도록 데이터의 변수 타입을 CPU가 지원하는 최대의 데이터 타입(integer:4bytes)으로 사용하여 변환했다. 그리고 속도의 향상보다는 음질의 저하를 막기 위해 승산연산의 결과에서는 8바이트의 데이터형(long long)을

사용했다. (2)단계에서는 오버플로가 일어나지 않은 범위 내에서 메모리 사용을 최소화하도록 데이터의 형을 줄이도록 했다. 즉, 1바이트 혹은 2바이트로 표현 가능한 데이터가 4바이트로 표현된다면 메모리는 4배 내지 2배가 더 사용되는 셈이므로 데이터 형을 변환하여 메모리의 사용을 줄였다. 한편, 여기 단계에서는 형 변환이 가능하더라도 속도의 향상을 고려해서 그대로 사용한 경우도 있다. (3)단계에서는 최적화된 고급언어를 캐쉬라는 저장공간을 효율적으로 이용하고 다른 응용 프로그램과의 연동을 위한 프로세서의 시간 점유율을 낮추고 메모리 절약을 위해서 고급언어의 기계화를 주요 함수에 대해서 실행했다.

3.2 구현 방법

RISC에서의 구현은 다음의 세 가지 방법과 ARM CPU의 특성을 이용한 방법을 통해 최적화 시키려고 했다. 특히 SA-110은 캐쉬 메모리를 가진 코어를 되어 있어서 캐쉬에 대한 고려를 충분히 반영하여 최적화 시키려고 했다.

1. 루프 합성: 여러 개의 루프에서 루프의 횟수가 동일하고 이들의 루프를 하나로 통합이 가능하다면, 루프에 횟수에 따른 분기수가 감소시킬 수 있으므로 시스템 성능에 상당한 효과를 볼 수 있다. 캐쉬가 있는 구조에서는 시간적 지역성을 없애주므로 캐쉬 참조 실패에 의한 시스템의 성능을 개선시킬 수 있다.
2. 루프 전개: 이 방법은 시간 점유율이 많은 부분에 대해서는 프로그램 코드 크기의 증가를 감수하고서 루프의 횟수를 감소시켜 시스템의 속도 성능을 향상시킬 수 있다. 특히, ARM 명령어에서는 8비트, 16비트나 32비트의 메모리 접근이 동일한 사이클을 필요하므로, 8비트나 16비트의 접근을 32비트로 대체 한다면 4번 혹은 2번의 메모리 접근을 동일 사이클을 소요하면서 단 한번의 메모리 접근으로 구현이 가능해진다.
3. 배열 병합: 함수의 루프 중에 배열의 수가 동일하고 동시에 같은 인덱스를 참조하는 부분이 있다면 배열 서로의 간섭으로 인해 캐쉬의 참조 실패가 일어날 가능성이 있다. 이런 위험성을 없애기 위해서는 두 개 혹은 그 이상의 배열을 인터리버시켜 하나의 배열로 통합하여 캐쉬의 참조 실패가 최소화되도록 했다.
4.. 구조체 및 데이터의 배열 재배치: 위의 세 가지 방법이 시스템의 속도 향상의 문제라면 이 방법은 메모리의 여백을 없애 쓸데없는 메모리 크기를 줄이기 위한 방법이다. 즉 데이터는 1바이트, 2 바이트, 4바이트, 8바이트로 이루어져 있으므로 데이터의 배치를 적절히 하여 메모리 사이의 공백이 생기지 않도록 했다.
5. 시프트를 이용한 상수의 승산: ARM CPU에서는 32*8의 승산기를 제공하며, 연산자 뒤의 승산 타입의 바이트가 커짐에 따라 연산에 필요한 사이클이 늘어난다. 일 예로, 16*16의 승산은 3 사이클이 소요된다. 따라서 상수의 연산을 수행할 경우 상수를 레지스터에 넣는 명령어와 승산하는 과정이 필요하다. 하지만 배럴 쉬프트를 이용해서 승산을 수행하면 보다 작은 사이클로 원하는 승산이 가능하다.
6. 알고리즘 측면에서의 최적화: 합성 필터는 프로세서의 시간 점유율을 많이 차지하는데, 코드의 크기를 줄이고 다중 메모리 접근 명령어 을 이용하여 최적화하기 위해서 여기서는 오히려 16비트의 메모리 접근보다는 32비트 메모리 접근을 하도록 했다. 또한, 메모리의 접근 (저장)을 줄이기 위해서 순환 어드레싱보다는 버퍼를 많이 할당하여 함수가 끝날 때 단순히 메모리 내용을 이동시켜 주는 방법을 사용했다. 다른 메모리 접근에 대해서는 우선 32비트로 로드하여 적당하게 쉬프트를 해서 원하는 두 개의 값을 얻어 연산을 수행하였다. 역시 저장의 경우에도 같은 방법을 사용하였다.
7. 테이블과 연산의 상호관계: 삼각함수나 제곱근함수 등의 부동소수점에서는 연산으로 가능한 수식에 대해서 고정소수점에서는 많은 사이클이 소요될 때, 이 수식을 테이블로 수정하여 연산의 효율을 높였다. 반대로 나눗셈에서의 몫이나 나머지 계산의 연산을 수행하는 메모리를 많이 차지하는 테이블 상수에 대해서는 간단한 연산이나 함수로 바꿔 수행시간에 영향을 주지 않으면서 메모리 사용량을 줄였다.

표1은 위의 세 가지 방법을 적용하여 각각의 성능 향상을 살펴본 것이다. 메모리의 접근 시 많은 사이클을 소요하는 DRAM에서 보다 큰 향상을 볼 수 있다.
한편, 표에 나타난 명령어나 사이클수는 각 함수 하나에 대한 사이클을 구한 것이기 때문에 캐쉬에 대한 고려는 없다. 캐쉬에 대한 고려가 들어가면 캐쉬 참조에 의한 실패의 이득을 얻을 수 있다.

방법 \ 사이클		명령어수	사이클 수	
			SRAM	DRAM
루프합성	전	487	538	1576
	후	150	167	436
루프전개	전	360	382	1076
	후	90	109	446
배열병합	전	150	181	536
	후	140	162	446

표1. 각각의 방법이 적용되었을 때의 명령어수 및 메모리의 종류에 따른 수행 사이클.

Ⅳ. 구현 결과

실험에는 프로세서로 ARM CPU계열의 SA-110을 사용했으며, 이 코어에는 명령어와 데이터 캐쉬가 각각 16kbytes가 존재한다. 또한 메모리의 경우에는 모든 메모리는 내부 메모리를 사용한다고 가정했다. 즉 메모리의 접근은 오직 하나의 사이클이 소요되는 것이다. 그리고 코어 내부와 외부의 주파수는 각각 235.5Mhz와 118Mhz이다.

입력 샘플은 www.mpeg.org/mpeg에서 다운로드 받았으며, 모노/스테레오에 대해서 샘플 주파수는 44.1Khz이고 비트율은 각각 160kbps와 224kbps이다.

RISC 프로세서에서 자주 사용하는 위의 세 가지 방법 및 ARM RISC의 특징적인 명령어를 적절히 사용하여 MPEG-1 오디오 디코더를 최적화 한 결과, 고정소수점으로 구현된 것보다 약 3배의 시스템 속도 향상으로 프로세서의 약 10%에서 실시간으로 구현할 수 있었으며, 메모리 측면에서도 30%의 절감효과를 가져왔다. 특히, 내장형 시스템에서의 메모리는 한정되게 마련이며 내부 메모리 대신 외부 메모리를 사용하게 되면 훨씬 더 큰 시스템의 성능 향상을 볼 수 있다.

Ⅴ. 결론 및 향후과제

부동소수점으로 구현된 알고리즘을 고정소수점으로 구현한 결과, 계층 1,2에 대해서는 18배, 계층 3에서는 12가량의 속도 향상이 있었고, 메모리 측면에서는 약 3배의 절감효과를 얻었다. 나아가, 고정소수점에서 기계어로의 최적화 변환으로 속도는 3배 가량의 향상이 있었고, 메모리는 30%를 절약할 수 있었다. 따라서, 부호화하는데 사용한 프로세서의 300% 이상의 시간이 필요한 알고리즘을 프로세서의 약 10%정도 사용하면서 실시간이 되도록 구현하였다. 요구된 메모리 측면에서도 많은 이득을 얻었다. 프로세서에 맞는 연산으로 구현한 다음 명령어의 적절하고 효율적인 이용으로 프로그램을 최적화시켜 원하는 시스템의 성능을 얻을 수 있었다.

앞으로의 과제는 MPEG-1 오디오 디코더를 이용하여 본 과제에서 구현하는 RTOS (Q+) 셋탑박스에서 안정성있는 MPEG-1 미디어 재생기를 완성하는 것과 데이터 방송을 위한 MPEG-2와 MP3 파일을 실시간에 구현하는 것이다.

참고문헌

[1] 김선태, "RISC CPU를 위한 디지털 신호처리 알고리즘의 효율적 구현," 석사학위 논문, 11.1999.

[2] S.Kim and W.Sung, A floating-point to fixed-point assembly program translator for the TMS 320C25, IEEE Trans. Circuit and System-II: Analog and Digital Signal Processing, vol.41, no.11,pp.730-739,nov.1994.

[3] Ki-Il Kum, Jiyang Kang and Wonyong Sung, "A Floating-point to Fixed-point C Converter for Fixed-point Digital Signal Processors," in Proc. of the Second SUIF Compiler Workshop, Aug. 1997.

[4] 정제창, "그림으로 보는 최신 MPEG," 교보문고, pp199~219, 1995

[5] 김선태, "MPEG-1 오디오 디코더의 고정소수점 구현에 관한 연구," 한국정보과학회 추계학술대회, 27권 2호, pp213-215, 10.2000

[6] J.L.Hennessey and D.A.Patterson, Computer Architecture: A Quantitative Approach, 2nd edition, Morgan Kaufmann, San Fransisco,1996