MPEG-1 Audio Decoder의 고정소수점 구현에 관한 연구

김선태°
한국전자통신연구원 인터넷정보가전연구부
stkim10@etri.re.kr

A Study on Fixed-point Implementation of MPEG-1 Audio Decoder

Seon-Tae Kim°
Dept. of Internet Appliance Technology, ETRI

요 약

디지털 신호처리 알고리즘의 구현은 속도나 메모리의 사용측면에서 고정 소수점 구현이 필요하다. 특히, 정수형 연산 프로세서에서는 소프트웨어에 의한 부동 소수점보다는 고정 소수점 구현이 훨씬 성능이 뛰어나다. 디지털 신호처리 알고리즘의 복잡함과 일반 프로세서의 처리능력의 부족으로 이제까지는 신호처리 알고리즘의 실시간 구현을 위하여 대개 전용 프로세서나 디지털 신호처리를 위한 전용 명령어가 하드웨어적으로 구현되어있는 프로세서를 사용하여왔다. 하지만 현재 범용 프로세서의 주파수 속도가 빨라짐에 따라 복잡한 디지털 신호처리 알고리즘을 실시간에 처리할 수 있게 되었다. 하지만 정수형 연산 프로세서에서의 부동 소수점 연산은 프로세서에서 실시간 처리에 많은 어려움을 주게 된다. 본 연구에서는 데이터 타입이 고정된 범용 정수형 연산 프로세서(ARM RISC 32bit CPU) 를 가지고 부동 소수점 연산 알고리즘을 고정 소수점 연산형으로 바꾸어서 속도측면과 메모리 측면의 성능을 비교해 보았다.

1. 서론

최근 몇 년 사이에 디지털 TV, 화상회의, ISDN, ATM과 같은 멀티미디어 서비스가 보편화되면서, 방대한 연산량을 요구하는 디지털 신호처리 알고리즘을 실시간으로 처리 할 수 있는 프로세서가 중요시 되고 있다. 과거에는 신호처리 알고리즘을 구현하는데 반복적이고 수치연산에서 월등한 프로그래머블 디지털 신호 프로세서를 많이 사용했지만, 최근에 프로세서의 동작 주파수가 빨라짐에 따라, 고급언어 컴파일러의 지원이 용이한 RISC 아키텍쳐 기반의 마이크로 콘트롤러를 이용한 프로세서의 구현이 점차 증가하고 있는 추세이다.

RISC 프로세서는 명령어가 간단하고, 어드레싱 모드가 단순하기 때문에 컨트롤러가 간단하여 구현하기 쉬울 뿐만 아니라, 레지스트가 풍부하고 레지스트의 길이가 길어 큰 메모리 공간을 사용할 수 있는 장점과 대부분의 명령어가 단일 사이클에 수행되고, 명령어 형식이 간단하여 컴파일러의 효율이 뛰어나 코드 밀도를 높일 수 있는 장점이 있다[1].

한편, 대부분의 RISC 프로세서는 정수형 연산이 하드웨어적으로 지원되고 칩의 복잡도를 줄이기 위해 부동 소수점 연산은 소프트웨어적으로 지원하는 경우가 많다. 부동소수점 연산이 소프트웨어적으로 지원되는 경우에는, 부동소수점을 고정 소수점으로 변환시킨 다음 정수형 연산이 일어나기 때문에 복잡한 디지털 신호처리 알고리즘이 실시간에 실행시키기는 많은 어려움이 있어 고정소수점의 구현이 필요하다. 본 연구에서는 데이터의 형이 고정된 범용 프로세서를 가지고 이 프로세서가 지원하는 데이터 형을 적절히 사용하여 음질의 저하를 최소로 하면서 실행 속도를 향상시키고 소요되는 메모리량을 최대로 줄이려고 했다.

본 논문의 구성은 다음과 같다. 2 장에서는 부동 소수점에서 고정 소수점으로의 변환에 대한 데이터 표현 방식과 MPEG-1 audio decoder 의 고정 소수점 구현 과정에 있어서 변수 및 상수의 정수단어길이를 결정하는 과정 및 최적화 과정을 살펴본다. 3 장에서는 구현 후의 두 알고리즘의 속도 및 메모리에 관련된 코드 사이즈의 성능 향상면에서 알아보고, 4 장에서는 본 논문의 결론과 앞으로의 해야 할 일을 언급하고 마친다.

2. 고정 소수점 구현 과정

2.1 데이터의 표현

데이터는 샘플링된 신호나 계수를 표현하는 변수와 상수의 값을 말한다. 고정 소수점 디지털 신호 처리기에서는 이런 데이터는 정수 부분과 소수점 이하의 수(fractional number)로 표현된다. 16-bit 데이터의 경우, 그림1에 나타난 것처럼 이진점(binary point)에 따라 데이터의 표현범위와 양자화 레벨이 결정된다. 예를 들어, 이진점이 부호비트(sign bit) 의 2 비트 아래에 위치한다면, 데이터는 -4~+4 까지 표현가능하며 정확도는 2^{-13}이 되며[2], 3 비트를 정수단어길이(integer word length)에 할당하면 -8~+8 까지의 표현범위를 넓힐 수 있지만 정확도는 2^{-12}로 떨어지게 되어 데이터를 보다 세밀하게 표현 못하게 된다. 즉, 이진점이 (가)위치에 존재하면 그림1의 데이터는 2.75 가 되고 (나)의 위치에 놓이게 되면 5.5 가 된다.

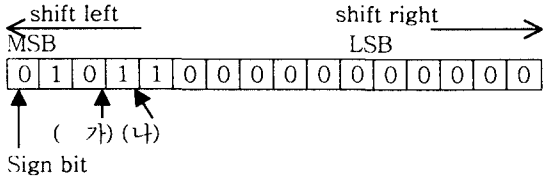

그림 1. 고정 소수점에서의 데이터 표현

2.2 변수 및 상수의 정수단어길이의 결정

변수의 정수 단어 길이는 시뮬레이션을 통한 통계적 성질과 변수의 최고치 중 큰 값을 가지고 결정하게 된다. 통계적 성질은 입력 샘플에 대해 변수의 평균과 표준편차를 구해서 원하는 오차의 범위에 벗어나지 않도록 최대의 정수단어길이를 결정하게 되는데, 식(1)과 같이 한다.

$$R(x) = |\mu(x)| + n * \sigma(x) \qquad (1)$$

여기서 n 은 대개 4 에서 16 까지를 사용하게 되며[2], n 이 클수록 데이터의 표현 범위가 넓어지고 오버플로(overflow)의 가능성이 낮아진다. 본 논문에서는 n 값을 4로 놓고 정수 단어길이를 결정했다. 변수에 대한 정수 단어길이는 다음 식(2)와 같이 계산된다[2][3].

$$W_I(x) = \lceil \log_2 R(x) \rceil \qquad (2)$$

여기서 $\lceil x \rceil$는 x 보다 크거나 같은 가장 작은 정수를 의미한다.

테이블이나 계수의 경우에는 상수로 주어지는데, 이들 상수들은 오버플로가 일어나지 않으면서도 이들간의 양자화 에러를 줄여 음질의 저하를 막기위해 소수점 이하에 많은 비트를 할당하도록 했다.

2.3 고정 소수점 구현의 최적화 과정

• ARM 프로세서에서 디지털 신호처리 알고리즘 d 연산에 많이 사용되는 주요 명령어에 대해서 부동 소수점과 고정 소수점 사이의 소요된 사이클을 시뮬레이션으로 비교해 보았는데, 가산/감산의 경우에는 120 배의 사이클이, 승산에 대해서는 약 60 배의 사이클이 부동 소수점연산에서 더 소요되는 것을 볼 수 있었다. 한편 메모리 접근 속도(load/store) 를 살펴보면, 각각의 데이터 형식을 그대로 이동시키기 때문에 부동 소수점과 고정 소수점 사이에 사이클의 차이가 없었다. 하지만 구현에 필요한 메모리 측면에서 보면, 64 비트 혹은 32 비트의 부동 소수점 데이터를 오버플로를 일으키지 않은 데이터에 대해서 16 비트로 구현 가능한 고정 소수점의 변환으로 메모리를 최적화 시켰다. 또한, 구조체의 구성멤버(members) 의 배열을 적절히 재배치하여 메모리의 불필요한 여백이 생기지 않도록 했다.

• ARM 프로세서에서는 long long이라는 64 비트의 데이터형을 제공하며, 32*8 의 승산 연산기가 있다. 승산연산은 최대 32*32 까지 가능하며 이때의 결과값은 32비트 혹은 64 비트로 저장 가능하다. 따라서 64 비트 경우, 부동 소수점과 같이 정밀하게 데이터를 표현할 수 있고, 이 데이터형을 적절히 사용하면 양자화 에러를 최소로 줄일 수 있다. 하지만, 이 데이터형은 32 비트 프로세서에서 두 개의 레지스터를 사용하고 불필요한 연산이 추가되어 수행속도에 영향을 미치므로 고도의 정밀도가 필요로 한 곳에만 사용되었다.

• 정수단어길이가 다른 두 변수나 상수사이에 가산/감산을 하거나, 데이터를 메모리로 이동을 할 때, 이진점을 위치를 같게 하려면 쉬프트 연산이 뒤따르는데 이 쉬프트 연산이 많아지면 시스템의 성능이 현저히 저하된다. 따라서 음질의 저하를 떨어뜨리지 않고 쉬프트 연산을 줄이도록 변수간의 정수단어길이 조정이 필요하다[3]. 즉, 두 변수 a,b가 정수단어길이는 다르지만 쉬프트 연산이 필요한 가산/감산 연산이 많을 때에는 두 변수 a,b의 정수단어길이를 같게 함으로써 쉬프트 연산을 감소시켜 시스템의 성능을 향상시켰다.

• 부동소수점에서 가능한 연산이 고정소수점에서는 구현이 불가능하거나 많은 사이클이 요구되는 경우에는 테이블 형식으로 변환하여 속도향상을 가져왔으며, 반대로 메모리를 많이 차지하고 실행속도에 영향을 크게 미치지 않으면서 사칙연산으로 대체 가능한 부분에 대해서는 테이블을 가능한 연산으로 대체하여 메모리 절감 효과를 가져왔다. 또한 많은 레지스터를 충분히 사용할 수 있도록 알고리즘을 수정하여 메모리 접근을 최소화시켜 속도향상을 얻었다.

• 시간 점유율이 많은 합성 필터 함수(synthesis filter function) 의 경우에는 배열의 크기를

다소 크게 늘려 메모리량이 다소 증가했지만, 분기(branch)를 포함하는 조건문을 없앰으로써 파이프라인(pipeline)의 지연시간을 줄여 시간 점유율을 줄였다[4].

3. 구현 결과

실험은 각 계층(layer)에 대해서 모노와 스테레오 모두를 테스트했다. 샘플은 www.mpeg.org/mpeg 웹 사이트에서 다운로드 받았으며, 샘플링 주파수는 44.1khz 에 비트율은 모노와 스테레오에 대해 각각 160kbps 와 224kbps이다. 실험 환경은 32bit RISC 프로세서인 Strong ARM(sa-110) 을 사용했는데, 메모리 접근시간은 하나의 사이클이 소요된다고 가정하고 시뮬레이션했다.

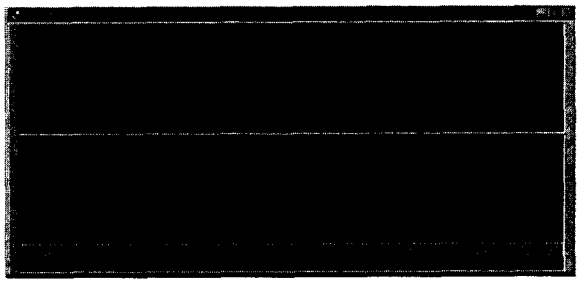

그림2. 부동소수점과 고정소수점 사이의 복호화된 두 신호의 차이 신호

그림2은 하나의 스테레오 샘플을 가지고 코드 변환 전후의 복호화된 두 신호를 빼서 2 배 곱한 값이다. 변환 전후의 양자화 에러나 음질의 변동이 거의 없음을 알 수 있다. 표2는 부동 소수점과 고정 소수점에서 요구되는 프로그램 코드 크기와 데이터 저장을 위한 메모리의 크기를 보여주는데 약 3 배의 메모리 절감 효과를 얻을 수 있었다. 표3은 코드 변환 전후의 한 프레임당 MPEG-1 audio decoder 알고리즘의 수행 명령어와 사이클을 보여주는데, 각 계층마다 현저한 명령어의 감소가 있었고, 사이클면에서 보면 layer1,2 은 약 18 배의 속도 향상이 있었고, layer3 은 약 12 배의 속도 향상을 가져왔다.

	Program Code Size		Data Size	
	Object	Library	Table Const	Array
Floating Point	62k	24k	124k	55k
Fixed Point	21k	8k	40k	20k

표2. 부동소수점과 고정소수점을 구현하는데 필요한 각각의 메모리량

		Floating Point		Fixed Point	
		Instructions	Cycles	Instructions	Cycles
Layer I	M	2002824	1460868	134746	103320
	S	3947670	2902091	225903	172977
Layer II	M	6183963	4522982	356751	279072
	S	11974940	8817016	621954	488905
Layer III	M	7688204	5596447	663029	519845
	S	15194060	11042860	1213109	945764

여기서 M은 모노, S는 스테레오를 의미한다.

표3. 한 프레임당 부동 소수점과 고정 소수점 사이의 명령어 수와 사이클 수

4. 결론 및 향후 과제

MPEG-1 audio decoder 알고리즘에 대한 부동 소수점에서 고정 소수점으로의 변환으로 음질의 저하를 느끼지 못 하면서 약 3 배가량의 메모리를 절약할 수 있었으며, 속도측면에서도 12~18 배의 시스템 성능 향상을 가져왔다. 따라서, 32bit RISC CPU 에서 실시간으로 동작이 어려운 알고리즘을 고급언어(C 언어)만 가지고 계층 1,2 에 대해서는 39Mhz, 계층 3 에 대해서는 70Mhz 정도의 동작 주파수에서도 실시간으로 동작이 가능하도록 했다.

앞으로 ARM RISC CPU 의 특성을 적절히 이용해서 주요 부분을 기계어로 대체하여, MPEG-1 audio decoder 알고리즘이 25Mhz 정도의 동작 주파수에서도 실시간으로 동작 가능하도록 할 예정이다. 또한 SRAM,DRAM과 캐쉬의 다양한 메모리 모델을 가지고 시스템의 성능분석을 하려고 한다.

참고 문헌

[1] 김선태, " RISC CPU 를 위한 디지털 신호처리 처리 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] T.Imai,J.Shiokawa and H.Chiba etc. " MPEG-1 audio real-time encoding system, " IEEE Trans. on consumer electronics, vol44, no.3, aug.1998.