

32 비트 MCU에 적합한 MPEG 오디오 소프트웨어 복호화기

이근섭, 박영철*, 윤대희
연세대학교 전기·전자공학과, *연세대학교 신호처리연구센터

Optimized MPEG Audio Software Decoder for 32-bit MCU

Keun-Sup Lee, Young-Cheol Park* and Dae-Hee Youn

Dept. of Electrical & Electronic Eng., Yonsei Univ., *Center for Signal Processing Research, Yonsei Univ.

Email : taraji@cyclon.yonsei.ac.kr

요약문

본 논문에서는 32 비트 MCU RISC 프로세서를 사용하여 MPEG 오디오 복호화기를 소프트웨어로 구현하였다. 구현된 MPEG 오디오 복호화기는 MPEG-2 Layer-III (MP3)와 MPEG-2 AAC로 구성된다. 프로그래밍 가능한 소프트웨어로 구현하여 향후 성능 개선이나 새로운 기능을 추가할 수 있는 유연성을 극대화하였다. 복호화기 구현은 구현 시간과 비용을 고려하여 직접 어셈블리를 코딩하는 대신 최적화된 C 코드를 사용하여 컴파일하는 방법을 선택하였다. 이때 발생할 수 있는 성능 저하 요소들을 줄이기 위해 추가의 최적화 과정을 수행하여 성능을 개선하는 방법을 제시하였다. 구현된 복호화기의 출력 음질은 ISO 13818-4 compliance test 결과 Full compliance를 만족하였다. 또한 연산량 최적화 결과 MP3 와 AAC 테스트 비트열에 대해 모두 35 MHz 이하의 동작 주파수로 동작이 가능함을 확인하였다.

I. 서 론

최근 몇 년간 MP3 플레이어와 같은 휴대용 디지털 오디오 플레이어가 활발하게 개발됨에 따라 휴대용 CD 플레이어와 함께 디지털 오디오 시장을 주도하고 있다. 현재 상용화된 MP3 플레이어는 핵심 모듈인 MP3 복호화기에 대부분 DSP 프로세서를 사용하고 있다. 그러나 DSP의 많은 전력 소모로 인한 짧은 사용 시간이 최근에 커다란 단점으로 부각되고 있다. 이것은 DSP의 사용이 전력 소모나 구현 비용 등에 있어서 휴대용 디지털 오디오 기기에 적합하지 않다는 것을

보여준다. 따라서 구조가 간단하고 전력 소모도 적은 MCU (Micro-controller Unit) RISC 프로세서를 DSP 대신 사용할 수 있다면 휴대용 MP3 플레이어의 구현에 좀 더 유리하다. 그러나 MCU는 일반적으로 DSP에 비해 상대적으로 단순한 구조와 적은 자원을 가지고 있으므로 구현 방법 또한 기존의 DSP에 적용되는 것과는 달라져야 할 필요가 있다.

본 논문에서는 32 비트 MCU RISC 프로세서를 이용하여 휴대용 기기에 적합한 소프트웨어 MPEG 오디오 복호화기를 구현하였다. 구현된 복호화기는 현재 휴대용 디지털 오디오 플레이어에 많이 사용되는 MPEG-2 Layer-III (MP3)[1]와 MPEG-2 AAC (Advanced Audio Coding)[2]로 구성된다. 복호화기 구현은 구현 시간과 비용을 고려하여 직접 어셈블리를 코딩하는 대신 C 코드를 컴파일하여 어셈블리 코드를 얻는 방법을 선택하였다. 이러한 방법은 구현 시간은 절약할 수 있지만 직접 코딩하는 방법에 비해 많은 성능 저하 요소들을 포함하고 있으므로, 추가의 최적화 과정을 수행하여 성능을 개선하는 방법을 제시하였다.

II. MCU의 구조적인 특징

일반적으로 MCU는 DSP와 비교할 때 다음과 같은 구조적 특징을 갖는다[3][4].

- 간단하고 단순한 구조
- 적은 수의 명령어
- 낮은 전력 소모
- 낮은 칩 가격

일반적으로 MCU는 단순한 구조로 인하여 사용 가능한 명령어의 수가 적다. 또한 매우 복잡한 구조를 갖는 연산기 중 하나인 곱셈기를 갖지 않거나 아주 간단한 형태의 곱셈기만을 갖는 것이 보통이다. 메모리 액세스 방법에 대해서도 액세스 시간이나 메모리 주소 발생 측면에 있어서 DSP와 같은 효율적인 구조를 갖지 못한다. 이러한 메모리 액세스에서의 단점을 보완하기 위해 비교적 많은 수의 레지스터를 가지고 있으며, 많은 레지스터를 사용함으로써 메모리 액세스 회수를 줄일 수 있다. 그러나 이처럼 많은 하드웨어 자원을 갖지 못한 만큼 전력 소모가 적으며 가격도 낮은 장점이 있다.

오디오 복호화기의 구현에 적합한 MCU를 선택하기 위해서는 프로세서의 연산 능력과 구현된 복호화기 출력의 음질을 주로 고려해야 한다. 오디오 복호화 알고리듬의 특성상 곱셈 연산이 많이 사용되므로 곱셈기를 가진 프로세서가 유리하다. 또한 고정소수점 연산 시에 발생하는 오차는 복호화기의 음질을 저하시키므로 가능한 한 긴 워드 길이를 갖는 ALU 구조가 필요하다. 따라서 본 논문에서는 단순한 형태의 곱셈기를 갖는 32비트의 MCU RISC 프로세서를 채택하였다.

III. MCU를 사용한 복호화기 구현

본 논문에서는 그림 1과 같은 과정을 거쳐 오디오 복호화기를 구현하였다. 오디오 복호화 알고리듬은 먼저 고정소수점 모의실험을 통하여 사용될 프로세서에서 동작할 수 있도록 C 언어를 통해 모델링 된다. 여기서 얻어진 C 코드는 컴파일러를 통하여 프로세서의 어셈블리 언어로 변환되어, 최적화 과정을 거쳐 최종적으로 소프트웨어 형태의 복호화기가 완성된다.

3.1. 고정소수점 모의실험

MCU는 대부분 단순한 형태의 연산만을 지원하므로 정수 형태의 데이터만을 처리한다. 따라서 복호화 알고리듬에서 사용되는 실수형 데이터의 연산을 위해서는 실수형 데이터를 고정소수점 연산으로 처리하기

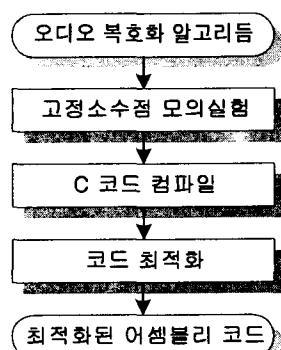


그림 1. MCU를 사용한 복호화기 구현 과정

위한 모의실험이 필요하다. 이때 고정소수점 연산에서 발생하는 양자화 오차는 최종 복호화기 출력의 음질에 직접적인 영향을 미치므로 이 오차를 최소화하기 위한 최적화 과정이 필요하다.

데이터를 표현하는 워드의 길이가 길어지면 데이터의 해상도를 높일 수 있으므로 긴 워드 길이를 갖는 프로세서를 사용하면 고정소수점 오차를 좀 더 줄여들 수 있다. 본 논문에서는 고정소수점 오차로 인한 음질 저하를 줄이고 또한 오차를 줄이는데 필요한 연산량을 최소화하기 위해 현재 많이 사용되고 있는 32비트의 MCU를 채택하였다.

그렇지만 높은 성능의 곱셈기를 갖지 못한 프로세서의 경우 '32비트 x 32비트' 곱셈 연산은 한번의 곱셈에도 많은 연산량이 필요하다. 따라서 본 논문에서는 대부분의 테이블 값을 16비트로 표현하고 '32비트 x 16비트' 곱셈을 사용함으로써 곱셈에 필요한 연산량을 줄이는 동시에 테이블에 사용되는 ROM의 사용량을 줄였다.

3.2. C 코드 컴파일 과정

일반적으로 C 코드를 컴파일해서 얻은 어셈블리 코드는 직접 코딩한 어셈블리 코드에 비해 비효율적인 요소를 포함하는 경우가 많다. 그러나 MCU 구조와 같이 단순한 명령어들만을 가지고 있는 경우에는 이러한 영향을 덜 받을 수 있다. 그림 2는 ARM7 코어의 ARM Development Suit 컴파일러를 사용하여 허프만 복호화 과정의 핵심 루틴을 컴파일한 결과를 나타낸 것이다. 제시된 허프만 복호화 루틴 예의 경우 컴파일된 결과가 직접 코딩한 어셈블리 코드와 다르지 않음을 보여준다. 이러한 결과는 모든 루틴에서 나타나는 현상은 아니지만 컴파일러를 통해 얻은 코드가 비교적 높은 효율성을 갖는다는 것을 보여주기엔 충분하다. 따라서 MCU 프로세서의 컴파일러를 사용하여 전체 코드를 컴파일한 후 일부 최적화가 필요한 루틴만을 Inline 어셈블리 기법을 사용하여 직접 코딩하면 빠른 시간 내에 최적화된 어셈블리 코드를 얻을 수 있다.

3.3. 코드 최적화 과정

코드 최적화 과정의 목적은 복호화에 필요한 연산량을 줄여서 실시간으로 복호화를 수행할 수 있도록 하는 것이다. 많은 연산량은 높은 동작 주파수를 요구하고 이것은 전력 소모량과도 관계가 있으므로 코드 최적화를 통하여 보다 가능한 한 많은 연산량을 줄이는 것이 유리하다.

일반적인 MCU는 간단한 구조라는 특성상 낮은 성능의 곱셈기로 인해 곱셈에 필요한 클럭 사이클 수가 많으며, 메모리 액세스에도 많은 연산이 필요하다. 또한 프로그램을 제어하는 함수 호출이나 루프 연산에서 DSP에 의해 효율이 낮다[3][4]. 따라서 본 논문에서는 연산량 최적화를 위해 많은 클럭 사이클을 소모하는

C-code	<pre> while((x = *val++) < 0) { if(Mask >= 0) val -= x; Mask <<= 1; num++; } </pre>
Hand Assembly Code	<pre> B ReadX Loop: CMP Rd, #0 SUBGE pHuff, pHuff, Re, LSL #2 MOV Rd, Rd, LSL #1 ADD Rb, Rb, #1 ReadX: LDR Re, [pHuff], #4 CMP Re, #0 BLT Loop </pre>
Cross Compiled Code	<pre> 0xf894:B 0xf8a8 0xf898:cmp r4, #0 0xf89c:subgr r12, r12, r0, lsr r3 0xf8a0:mov r4, r4, lsl #1 0xf8a4:add r14, r14, #1 0xf8a8:ldr r0, [r12], #4 0xf8ac:cmp r0, #0 0xf8b0:blt 0xf898 </pre>

그림 2. 컴파일러의 성능 비교

몇몇 특정 연산, 즉 곱셈, 메모리 액세스, 함수 호출 그리고 루프의 회수를 줄이는 방법을 사용하였다.

3.3.1. 고속 알고리듬의 사용

MPEG 오디오 복호화 알고리듬은 그 특성상 합성 필터 과정의 연산량이 가장 많으며 대부분 곱셈 연산으로 이루어져 있다[1][2]. 대부분의 MCU는 곱셈에 필요한 클럭 사이클 수가 많기 때문에 합성필터 과정을 처리하는데 많은 부담을 가진다. 따라서 합성필터 루틴에 곱셈 수가 적은 고속 알고리듬을 사용함으로써 많은 연산량을 줄일 수 있다.

본 논문에서는 합성필터에서 많은 비중을 차지하는 IMDCT 루틴에 대해 고속 알고리듬을 적용하였다. MPEG-2 AAC의 IMDCT 루틴은 2^n (n 은 정수)개에 해당하는 데이터를 처리하므로 FFT를 이용한 고속 알고리듬[5]을 사용하였고, MPEG-2 Layer-III는 9 포인트의 고속 Lee's DCT 알고리듬[6]을 사용하였다. 이러한 고속 알고리듬은 루틴 전체의 연산량도 줄어들지만, MCU 프로세서의 경우 곱셈에 필요한 연산량이 많으므로 곱셈의 회수가 줄어들면서 얻어지는 연산량 감소의 효과가 매우 크다.

3.3.2. 메모리 액세스 최소화

MPEG 오디오 알고리듬은 하나의 오디오 프레임 당 MP3는 1152 개, AAC는 1024 개의 데이터를 처리하

므로 매 루틴마다 메모리로부터 데이터를 읽고 쓰는 연산이 자주 사용된다. 특히 MCU는 메모리 액세스가 빠르지 않기 때문에 메모리의 데이터를 읽고 쓰는 회수가 줄어들수록 연산량 감소의 효과가 크다. 특히 대부분의 MCU들이 연속된 주소의 데이터를 한꺼번에 읽는 경우 필요한 클럭 사이클이 줄어들므로 이것을 적극 활용할 필요가 있다[3][4]. 본 논문에서는 서브밴드 필터뱅크 루틴에서 이러한 방법을 적용하였다. 기본적으로 매트릭싱 부분에서는 고속의 Lee's DCT 알고리듬을 사용하였고 루틴에서 사용되는 데이터가 연속적으로 배열되도록 식 (1)과 같이 변형된 알고리듬을 사용하였다[7].

$$out(i) = \sum_{m=0}^{15} V(32i+m) \cdot D(32i+m), \quad 0 \leq i \leq 31 \quad \dots \dots \quad (1)$$

where $out(i)$: 출력 PCM
 $V(\cdot)$: 매트릭싱 루틴 출력
 $D(\cdot)$: 윈도우 계수

ARM7 코어의 경우를 예로들면 메모리로부터 하나의 데이터를 읽을 때는 3 사이클이 필요하다. 하지만 연속된 주소에 있는 데이터를 한꺼번에 같이 읽어들이면 첫번째 이후의 데이터에는 1 사이클만에 읽을 수 있다. 이러한 방법을 사용하면 많은 수의 데이터를 한꺼번에 읽거나 쓸 경우 하나의 데이터당 약 1 사이클에 가까운 연산량으로 메모리를 액세스 할 수 있다. 따라서 루틴에서 사용되는 데이터를 연속적으로 배열하여 읽고 쓸 경우 메모리 액세스에 필요한 연산량을 줄일 수 있다.

3.3.3. 분기 연산 최소화

또한 MCU는 분기 명령어에도 많은 연산량을 요구한다. 복호화 알고리듬의 특성상 분기 명령어가 많이 사용되는 부분은 함수 호출과 루프이다. 이 두가지 경우는 모두 프로그램 상에서 반복되어 사용되는 부분을 재사용하기 위한 연산이므로 이 연산의 회수를 줄이게 되면 프로그램 메모리의 사용량이 늘어날 가능성이 높다.

함수 호출의 회수를 줄이기 위해 많이 사용되는 것이 바로 Inline 함수이다. Inline 함수를 사용하면 함수를 호출하는 대신 함수의 코드 전체가 포함되므로 분기 명령은 없어지지만 프로그램 메모리가 더 많이 필요하다. 따라서 자주 사용되지만 비교적 적은 연산량으로 구성된 함수에 적용하는 것이 유리하다. 본 논문에서는 비트열로부터 필요한 정보를 얻어내는 unpacking 루틴에 Inline 함수를 적용하였다. 또한 Inline 어셈블리를 사용하여 직접 어셈블리 코딩을 수행함으로써 함수에 필요한 연산량을 최소화 하였다.

루프에 사용되는 분기 명령은 언제 루프를 멈출 것인지를 결정해야하는 조건 분기 명령이므로 더 많은 클럭 사이클이 필요하다. 따라서 루프의 회수가 적은

회수로 징해져있고 루프 안에서 수행하는 연산이 간단한 경우에는 오히려 루프 대신 필요한 회수만큼 같은 루틴을 연속적으로 나열하는 것이 연산량 측면에서 좀 더 유리하다.

IV. 실험 및 결과

본 논문에서는 많은 상용 MCU RISC 프로세서들 중 ARM7TDMI 프로세서를 선택하여 디지털 오디오 복호화기를 구현하였다. ARM7TDMI 프로세서는 32비트 RISC 구조로서 간단한 곱셈기를 가지고 있으며 실제로 디지털 오디오 코덱 구현에 많이 사용되고 있다 [3][9]. 구현된 오디오 복호화기는 MPEG-2 Layer-III 와 MPEG-2 AAC이며 앞서 언급된 구현 방법과 최적화 방법이 모두 적용되었다.

먼저 구현된 복호화기들이 갖는 출력 PCM의 정확성을 평가하기 위하여 ISO 13818-4 compliance test를 수행하였다[8]. 표 1에 의하면 MPEG-2 Layer-III 와 MPEG-2 AAC 복호화기는 모두 테스트의 기준이 되는 최대값을 넘어서지 않고 있다. 따라서 두 복호화기의 출력은 모두 compliance test를 만족하는 높은 정확성을 갖는다고 할 수 있다.

표 2는 구현된 복호화기의 성능을 나타내고 있다. MPEG-2 Layer-III (MP3)의 경우 2 채널의 128kbps, 44.1kHz로 부호화된 대중음악의 테스트 비트열에 대해 연산량을 구한 것이다. MPEG-2 AAC의 경우에는 역시 2 채널의 같은 조건을 갖는 백색 잡음의 테스트 비트열에 대해 연산량을 구하였다. MPEG-2 AAC의 경우 TNS (Temporal Noise Shaping) 루틴의 사용 여부에 따라 연산량이 많이 차이가 나므로 사용할 경우와 아닌 경우로 나누어서 나타내었다.

표 2에 의하면 MP3 와 AAC 모두 35 MHz 이하의 연산량을 가지고 있는 것을 알 수 있다. ARM7TDMI 코어의 경우 최대 130 MIPS의 성능을 가지므로 충분히 실시간 구현이 가능하다. ARM 사의 소프트웨어 라이브러리의 경우 MP3 는 약 29MHz, AAC는 22 MHz (without TNS) / 28 MHz (with TNS)의 연산량을 보인다[9]. 이 라이브러리의 성능은 C 언어와 어셈블리 코딩을 사용하여 충분히 최적화된 결과로 얻어진 것이며, 침파일러를 사용하여 C 언어로 최적화된 본 논문의 최적화

표 1. ISO 13818-4 compliance test 결과

	RMS of difference	Max. difference
Full Compliance	8.8097×10^{-6}	6.1035×10^{-5}
MP3	4.7483×10^{-6}	2.8729×10^{-5}
AAC	5.6012×10^{-6}	3.0518×10^{-5}

표 2. 구현된 MP3/AAC 복호화기의 연산량

	MP3	AAC	
		with TNS	without TNS
연산량	33.4 MHz	25.4 MHz	32.6 MHz

방법으로 얻은 결과와 비교했을 때 5MHz 이하의 비교적 적은 차이를 보인다.

V. 결론

본 논문에서는 32비트 MCU RISC 프로세서를 사용하여 MPEG-2 Layer-III, MPEG-2 AAC 오디오 복호화기를 소프트웨어로 구현하였다. MCU를 사용한 구현 방법으로서 구현 시간과 비용을 고려하여 직접 어셈블리를 코딩하는 대신 최적화된 C 코드를 사용하여 컴파일하는 방법을 선택하였다. 이때 발생할 수 있는 성능 저하 요소들을 줄이기 위해 추가의 최적화 과정을 수행하여 성능을 개선하는 방법을 제시하였다. 구현된 복호화기의 출력 음질은 ISO 13818-4 compliance test를 만족하였다. 또한 연산량 최적화 결과 MP3 와 AAC 테스트 비트열에 대해 모두 35 MHz 이하의 동작 주파수로 동작이 가능함을 확인하였다.

참고 문헌

- [1] ISO/IEC 13818-3 "Generic Coding of Moving Pictures and Associated Audio (Part 3: MPEG-Audio)", 2nd Edition, Feb., 1997
- [2] ISO/IEC JTC1/SC29/WG11 No.1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)", Apr., 1997.
- [3] ARM, *ARM7TDMI Data Sheet*, 1995
- [4] Motorola, *M•Core microRISC Engine Programmer's Reference Manual*, 1997
- [5] P. Duhamel, Y. Mahieux, and J. P. Petit, "A fast algorithm for the implementation of filter banks based on 'time domain aliasing cancellation'" *Proc. ICASSP*, May 1991, pp.2209-2212.
- [6] Byoung Gi Lee, "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Trans. on Acoustic, Speech and Signal Processing*, Vol. ASSP-32, No.6, pp.1243-1245, 1984
- [7] Tadashi Sakamoto, Maiko Taruki and Tomohiro Hase, "A Fast MPEG-Audio Layer-III Algorithm for a 32-bit MCU", *IEEE Trans. on Consumer Electronics*, Vol. 45, No. 3, Aug. 1999
- [8] ISO/IEC 13818-4 "Generic coding of moving pictures and associated audio information (Part 4: Compliance testing)"
- [9] ARM Ltd. Homepage (<http://www.arm.com>)