TMS320C6201을 이용한 MPEG-1 Layer Ⅲ 오디오 디코더의 실시간 구현

정회원 권 홍 석*, 김 시 호*, 배 건 성*

Real-Time Implementation of MPEG-1 Layer Ⅲ Audio Decoder Using TMS320C6201

Hong-seok Kwon*, Si-ho Kim*, Keun-sung Bae* ***Regular Members***

요 약

본 논문에서는 고정소수점 DSP인 TMS320C6201을 이용하여 MPEG-1 Layer III 오디오 디코더를 실시간으로 동작하도록 구현하였다. 음질의 손실 없이 부동소수점 연산을 고정소수점 연산으로 변환하였으며, 적은 메모리를 사용하여 동작하도록 소스 프로그램을 최적화하였다. 특히 연산의 정확성을 위해서 Descaling 모듈에서 중점적으로 부동소수점 연산을 고정소수점 연산으로 변환하였고, 연산량과 프로그램 크기를 줄이기 위해서 IMDCT 모듈과 Synthesis Polyphase Filter Bank 모듈에 대해서 최적화 작업을 수행하였다. 그 결과 구현된 디코더는 TMS320C6201 DSP가 수행할 수 있는 최대 연산량의 26%만으로 실시간 동작이 가능하였으며, 사용된 프로그램 ROM의 크기는 6.77 kWord, 데이타 ROM의 크기는 3.13 kWord, 데이타 RAM의 크기는 9.94 kWord이었다. 부동소수점 프로그램의 최종 출력 PCM값과 구현된 고정소수점 연산의 최종 출력 PCM값을 비교하여 60 dB 이상의 높은 SNR를 가짐을 확인함으로써 고정소수점 연산의 정확성을 검증하였다. 또한 EVM 보드에서 사운드 입출력과 호스트(PC) 통신을 이용하여 실시간으로 동작함을 확인하였다.

ABSTRACT

The goal of this research is the real-time implementation of MPEG-1 Layer Ⅲ audio decoder using the fixed-point digital signal processor of TMS320C6201. The main job for this work is twofold: one is to convert floating-point operation in the decoder into fixed-point operation while maintaining the high resolution, and the other is to optimize the program to make it run in real-time with memory size as small as possible. We, especially, devote much time to the descaling module in the decoder for conversion of floating-point operation into fixed-point operation with high accuracy. The inverse modified cosine transform(IMDCT) and synthesis polyphase filter bank modules are optimized in order to reduce the amount of computation and memory size. After the optimization process, in this paper, the implemented decoder uses about 26% of maximum computation capacity of TMS320C6201. The program memory, data ROM, data RAM used in the decoder are about 6.77 kWords, 3.13 kWords and 9.94 kWords, respectively. Comparing the PCM output of fixed-point computation with that of floating-point computation, we achieve the signal-to-noise ratio of more than 60 dB. A real-time operation is demonstrated on the PC using the sound I/O and host communication functions in the EVM board.

* 경북대학교 전자전기공학부(ksbae@ee.knu.ac.kr)
논문번호 00170-0512, 접수일자 : 2000년 5월 12일

Ⅰ. 서 론

디지털 오디오의 장점은 아날로그에 비해 음질이 훨씬 깨끗하고, 음질의 변화가 거의 없다는 것이다. 또한 디지털 오디오는 영화나 컴퓨터와 같은 다른 매체와의 호환성이 뛰어나서 그 응용범위도 확장되고 있다. 그러나 아날로그 오디오 신호를 샘플링해서 얻어지는 고음질의 디지털 오디오 신호는 데이타량이 아주 많아서 막대한 용량의 저장매체를 요구하며, 전송시 매우 넓은 주파수 대역을 차지하게 된다. 따라서 오디오 신호를 음질의 손상 없이 효과적으로 압축하는 기술이 필요하게 되었다. 이러한 요구를 충족시키는 오디오 압축기술 중에서 현재 세계적으로 가장 널리 사용되고 있는 것이 유럽의 MUSICAM 방식에 기반을 둔 MPEG이다[1]. 지금까지 상용화된 MPEG 오디오 압축 방식 중에서 가장 좋은 음질과 낮은 전송률을 가진 것이 Layer III이고 이는 CD 수준의 오디오 신호를 12:1 이상의 압축률로 재생가능하다[2-4].

본 논문에서는 MPEG-1 Layer III 오디오 디코더를, ISO/IEC 11172-3[5]에서 제공하는 부동소수점 연산 프로그램을 이용하여 고정소수점 DSP인 TMS 320C6201 EVM 보드에서 실시간으로 동작하도록 구현하였다. MPEG-1 Layer III 디코더를 고정소수점으로 구현하는데 문제가 되는 부분은 Descaling 과정에서 발생되는 연산 결과의 정확성 저하와 허프만 디코딩된 값의 4/3승 연산을 위한 메모리의 낭비이다. 이를 해결하기 위하여 허프만 디코딩이 이루어진 값을 8로 나누고 선형 보간하는 방법과 부동소수점 연산 형식을 이용하는 방법을 적용하였다. 또, 수행 속도 증가를 위하여 IMDCT 모듈과 Synthesis Polyphase Filter Bank 모듈에 factorization 방법과 FDCT 방법을 사용하여 최적화하였다. 구현된 시스템의 프로그램 ROM의 크기는 6.77 kWord, 데이타 ROM의 크기는 3.13 kWord, 데이타 RAM의 크기는 9.94 kWord이고, 부동소수점 프로그램의 최종 출력 PCM값과 구현된 고정소수점 연산의 최종 출력 PCM값을 비교한 결과 60 dB 이상의 높은 SNR를 가짐을 확인하였다. 본 논문의 구성은 다음과 같다. 2장에서는 MPEG-1 Layer III 오디오 디코더의 디코딩 과정에 대해서 설명한다. 3장에서는 TMS320C6201의 기능 및 호스트와의 데이타 전송 방법과 사운드 제어 방법에 대하여 간략하게 설명한다. 4장에서는 부동소수점 연산을 고정

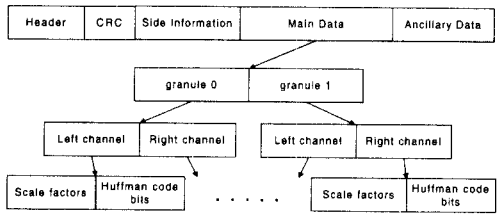


그림 1. MPEG-1 Layer Ⅲ의 프레임 구성

소수점 연산으로 변환하는 방법과 실시간 처리를 위한 최적화 방안에 대해서 구체적으로 설명한다. 5장에서는 구현된 고정소수점 MPEG-1 Layer III 오디오 디코더의 성능 및 결과를 제시하고 마지막으로 6장에서 결론을 맺는다.

Ⅱ. MPEG-1 Layer III 오디오 디코더

MPEG-1 Layer III 오디오 비트스트림의 각 프레임은 압축된 1152개의 PCM 샘플을 복원하는데 필요한 정보와 오디오 데이타를 담고 있으며 크게 5개 부분으로 나누어진다. 각 부분은 그림 1과 같이 헤더, CRC, 부가 정보(side information), 주 데이타(main data), 그리고 보조 데이타(ancillary data)로 나누어진다. 헤더 부분은 프레임 동기를 찾기 위한 12개의 비트와 layer, bit rate, sampling rate, 그리고 채널에 대한 정보를 담고 있다. 부가 정보는 인코딩 과정에서 사용된 block type, 허프만 데이타를 디코딩하는데 필요한 정보를 가지고 있다. 주 데이타 부분은 두 개의 그래뉼로 구분되며 각각은 인코딩된 오디오 데이타를 담고 있다.

그림 2는 전체적인 디코딩 과정을 나타내고 있으며 각 모듈별 기능은 다음과 같다. Synchronization

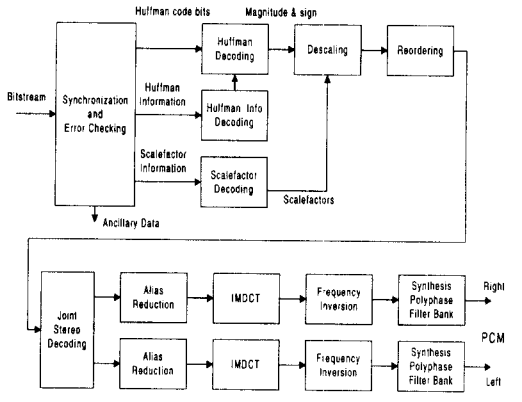


그림 2. MPEG-1 Layer III 오디오의 디코딩 과정

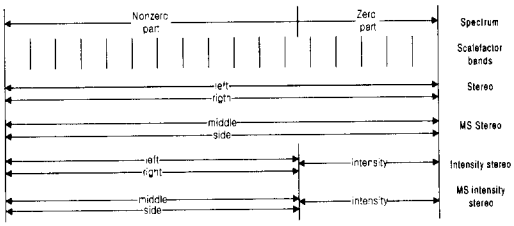


그림 3. 스테레오 신호에 대한 4가지 모드

and Error Checking 모듈은 입력 비트스트림을 받아서 프레임에 대한 동기를 찾고 한 프레임에 대한 비트스트림을 추출한다. Huffman Info Decoding 모듈은 Huffman Decoding 모듈에서 사용할 파라미터들을 추출하는 모듈로서 허프만 코드 비트의 특성에 대한 정보를 비트스트림의 부가 정보 부분에서 추출한다. Scalefactor Decoding 모듈은 주 데이타의 각 그래뉼, 각 채널마다 존재하는 scalefactor를 디코딩하는 단계이다. scalefactor는 Descaling 모듈에서 사용되는 값으로서 인코더의 MDCT(Modified Discrete Cosine Transform)에서 사용한 창의 형태에 따라 달라진다. Descaling 모듈은 인코더의 MDCT 과정에서 만들어진 frequency line을 복원하는 과정이다. Descaling은 Huffman Decoding 모듈에서 계산되는 frequency line과 Scalefactor Decoding 모듈에서 복원한 scalefactor를 이용하여 수행된다. 이때 각 그래뉼의 각 채널에 해당되는 frequency line은 long window인 경우에는 식 (1)을, short window인 경우에는 식 (2)를 이용하여 구해진다[5].

허프만 인코딩 과정의 MDCT 모듈에서 long window가 사용되면 frequency line은 서브밴드, 주파수 순으로 배열된다. 그러나 short window가 사용될 경우에는 서브밴드, window, 주파수 순으로 배열된 frequency line을 허프만 인코딩의 효율을 높이기 위해 서브밴드, 주파수, window 순으로 재배열하게 된다. 따라서 디코딩 과정의 재배열(Reordering) 모듈에서는 short window가 사용되었을 경우에 대하여 원래 순서대로 바꾸는 과정을 수행한다.

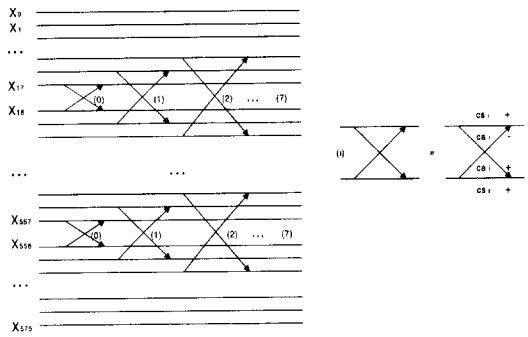


그림 4. Alias reduction 과정

MPEG-1 Layer III 알고리즘에서는 single 채널과 dual 채널을 지원할 뿐만 아니라, MS(Middle/Side) 스테레오와 intensity 스테레오도 지원한다. 이러한 방법들은 스테레오에서 양쪽 채널의 상관성 있는 정보를 제거함으로써 데이타량을 줄이기 위한 것이다. 스테레오 처리(Stereo Processing) 모듈에서는 인코딩된 스테레오 신호를 좌·우 두 채널의 원래 신호로 분리해 내는 일을 한다. 또 intensity 스테레오인 경우에는 상위 주파수 서브밴드를 하나의 합 신호로 나타내고 하위 주파수 서브밴드는 dual 채널이나 MS 스테레오를 이용하여 나타낸다. 따라서 MPEG-1 Layer III에서 가능한 스테레오의 구성은 그림 3과 같이 4가지 모드로 구성이 가능하다.

Alias Reduction 모듈에서는 그림 4와 같이 각 서브밴드마다 8개의 butterfly를 계산함으로써 수행된다. Alias Reduction 모듈로부터 얻은 frequency line은 IMDCT 모듈을 통하여 polyphase filter 서브밴드의 각 샘플로 변환된다[6]. 이때 사용되는 IMDCT의 변환 과정은 식 (3)과 같다. 여기서 X_k는 frequency line을 나타내고, short block에 대해서 n = 12, long block에 대해서 n = 36이다. 이렇게 구해진 각 샘플 값은 인코딩에서 사용된 창의 형태에 따라 windowing된 다음 이전 그래뉼의 값과 overlap-add를 함으로써 32개의 벡터를 얻을 수 있다.

$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos\left(\frac{\pi}{2n}(2i+1+\frac{n}{2})(2k+1)\right) \quad \text{for } i=0 \text{ to } n-1 \tag{3}$$

$$xr_i = sign(is_i)\cdot abs(is_i)^{4/3}\cdot \frac{2^{1/4(global\_gain[gr][ch]-210)}}{2^{(scalefac\_multiplier(scalefac\_l[gr][ch][sfb]+preflag[gr][ch]\cdot pretab[sfb]))}} \tag{1}$$

$$xr_i = sign(is_i)\cdot abs(is_i)^{4/3}\cdot \frac{2^{1/4(global\_gain[gr][ch]-210-8\cdot subblock\_gain[gr][ch][sfb][window])}}{2^{(scalefac\_multiplier\cdot scalefac\_s[gr][ch][sfb][window])}} \tag{2}$$

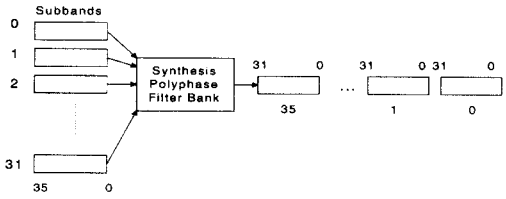


그림 5. Synthesis Polyphase Filter Bank 흐름도

Synthesis Polyphase Filter Bank에서의 frequency inversion을 보상하기 위하여 Frequency Inversion 모듈에서는 모든 홀수 서브밴드의 홀수 샘플에 대해 음의 값(-)을 곱하게 된다. 이 과정을 거친 32개 서브밴드의 각 샘플을 그림 5에서 보는 것처럼 Synthesis Polyphase Filter Bank를 통과 시키면 최종적으로 32개의 연속적인 오디오 샘플을 얻을 수 있다.

Ⅲ. TMS320C6201 EVM 보드 및 I/O 인터페이스

TMS320C6201 DSP(Digital Signal Processor)는 Texas Instrument사의 TMS320C62xx 고정소수점 DSP 제품군의 하나이다. TMS320C6201의 CPU는 A side와 B side로 분류되는 두 세트의 functional units를 가지고 있으며 하나의 클럭 사이클 당 최대 8개의 32 비트 명령을 동시에 수행할 수 있다. 따라서 CPU는 200MHz의 클럭 rate에서 최대 1600 MIPS(Million Instructions Per Second)의 성능을 발휘할 수가 있다[7,8]. CPU 내에는 DMA controller가 있어서 CPU의 개입 없이도 메모리 간의 데이타 이동이 가능해 데이타 이동으로 인한 CPU의 실행 속도 감소를 없앨 수 있다.

보드상의 MPEG-1 Layer III 디코더 프로그램은 입력 데이타로서 인코딩된 비트스트림을 필요로 하므로, PC의 호스트 프로그램이 하드디스크에서 비트스트림 파일을 읽어 보드로 데이타를 전송해 주어야 한다. 보드와 PC가 데이타를 주고받을 수 있는 방법에는 여러 가지가 있는데 그 중에서 대량의 데이타를 주고받을 수 있는 방법은 두 가지가 있다. 첫 번째가 보드의 PCI controller내 FIFO 레지스터를 통하는 방법이고 다른 하나는 HPI(Host Port Interface)를 통한 방법이다. HPI를 통한 방법은 호스트로부터 전송된 데이타가 DMA controller에 의해 보드의 지정된 메모리에 저장 또는 읽혀지는 형태로 수행된다. 이 방법의 장점은 보드 프로그램이 데이타 전송에 큰 관여를 하지 않으므로 CPU가 부하를 받지 않고 본래의 연산 작업에 충실할 수 있다는 점이다. 따라서 본 연구에서는 보드와 호스트 간의 통신을 위하여 HPI를 이용하였다[9]. EVM 보드에서 사운드 출력은 McBSP0(Multi-channel buffered serial port)를 통하여 이루어진다. EVM 보드에서 사운드의 실제 출력에 관여하는 장치는 stereo audio codec(CS4231A)으로서 16 비트 스테레오 채널에 5.5kHz에서 48kHz까지의 sampling rate를 제공한다. 따라서 EVM 보드는 32kHz, 44.1kHz, 48kHz의 sampling rate를 가지는 MPEG-1 Layer III 오디오를 실시간 구현하는데 충분한 sampling rate를 제공해 준다.

Ⅳ. TMS320C6201 EVM에서 실시간 처리를 위한 최적화 과정 및 고정소수점 변환

1. Descaling 과정

Descaling 과정은 2장에서 설명한대로 허프만 디코딩된 값과 Scalefactor Decoding 모듈에서 얻은 scalefactor 값을 이용하여 크기 변환된 frequency line을 구하는 부분이다. 실제로 전체 디코딩 과정에서 처음으로 부동소수점 값이 나타나는 부분이며 4/3승 연산을 포함하므로 고정소수점 연산을 하는데 문제가 되는 부분이다. 즉, TMS320C6201 DSP는 고정소수점 연산을 위한 지수 함수를 지원하지 않으므로 일반적으로 미리 표를 만들어 놓고 허프만 디코딩된 값에 따라 해당되는 값을 찾는 방식을 사용하게 된다. 하지만 이런 방법은 메모리 크기의 증가 및 낮은 Q-format을 야기시키므로 이를 해결하기 위하여 다음과 같은 방법을 이용하였다.

1) 부동소수점 형식 적용

Descaling 과정은 2장에서 설명한 것처럼 식 (1), (2)를 이용하게 된다. 이때 허프만 디코딩된 값의 크기는 그 범위가 최대 8191이고, 4/3승을 하면 결과적으로 최대 165140.3719이 된다. 따라서 이를 Q-format으로 나타내면 32 비트 레지스터를 이용하더라도 최대 Q-13 format밖에 사용할 수 없다. 이런 동적 범위의 크기를 줄이기 위하여 그림 6과 같은 형태로 표를 만들어 사용하였다. 가장 상위 비트는 계산의 편의를 위하여 0으로 값을 설정하고, 다음 상위 4 비트는 각 표 값의 2의 지수값에 해당되며, 나머지 27 비트는 밑수값으로서 Q-format을 이용하여 고정소수점으로 표현한 것이다.

$$
\begin{aligned}
xr_i &= sign(is_i)\cdot abs(is_i)^{\frac{4}{3}}\cdot \frac{2^{\frac{1}{4}(global\_gain[gr][ch]-210)}}{2^{(multiplier\cdot(scalefac\_l[gr][ch][sfb]+preflag[gr][ch]\cdot pretab[sfb]))}} \\
&= sign(is_i)\cdot abs(is_i)^{\frac{4}{3}}\cdot 2^{\frac{1}{4}(global\_gain[gr][ch]-210-4\cdot(multiplier\cdot(scalefac\_l[gr][ch][sfb]+preflag[gr][ch]\cdot pretab[sfb])))} \\
&= sign(is_i)\cdot abs(is_i)^{\frac{4}{3}}\cdot 2^{\frac{1}{4}(4\cdot a+b)} \\
&= (sign(is_i)\cdot abs(is_i)^{\frac{4}{3}}\cdot 2^{\frac{b}{4}})\ll a
\end{aligned}
\tag{4}
$$

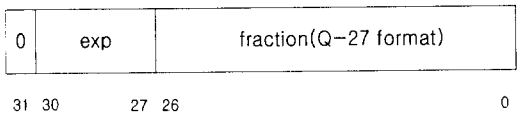


그림 6. Descaling 과정에서의 4/3승 표 구성

2) Descaling 과정의 수정

TMS320C6201 DSP에서는 Descaling 과정에서 필요한 2의 지수승 연산을 제공하지 않는다. 따라서 Descaling 과정은 식 (4)와 같이 수식을 수정함으로써 지수값의 소수부분이 단지 1/4승, 2/4승, 3/4승만으로 표현 가능하게 된다. 이렇게 함으로써 2의 지수승을 프로그램으로 구현할 때 필요한 배열의 크기를 줄일 수 있으며 계산도 간단하게 수행할 수 있다. 식 (4)는 long window에 대해서 나타낸 것이고 short window에 대해서도 식 (2)를 식 (4)처럼 적용함으로써 같은 효과를 얻을 수 있다.

3) 선형 보간법(Linear interpolation) 사용

허프만 디코딩이 이루어진 값은 0~8191의 값을 가지게 되므로 이 값을 위에서 언급한 방법으로 표현하더라도 최소한 Word32(4 bytes) 크기의 데이타가 8192개 필요하게 된다. 이는 표의 크기가 커져 메모리 낭비를 초래한다는 문제점이 있다. 따라서 이런 문제점을 해결하고자 식 (5)처럼 허프만 디코딩된 결과의 4/3승 부분을 8로 나눔으로써 메모리를 줄일 수 있으며, 그 오차를 선형 보간하여 보상한다[10].

$$
\begin{aligned}
&abs(is_i)^{\frac{4}{3}} = abs(\frac{is_i}{8}\times 8)^{\frac{4}{3}} = abs(is_i'\times 8)^{\frac{4}{3}} \\
&= abs(is_i')^{\frac{4}{3}}\times 16 = abs(is_i')^{\frac{4}{3}}\ll 4
\end{aligned}
\tag{5}
$$

2. 스테레오 처리 과정

MS 스테레오 모드인 경우에는 descaling된 각 채널의 값을 이용하여 식(6)에서 보는 것처럼 좌·우 채널값을 구하게 된다. 식 (6)에서 $\sqrt{2}$를 나누는 연산을, $1/\sqrt{2}$ 값을 Q-15 format으로 표현한 23170을 곱하는 연산으로 대치하여 사용하였다. Intensity 스테레오 모드는 intensity 스테레오 모드로 정의된 위치에서 scalefactor 주파수의 scalefactor 값으로 is_pos 값을 구하여 식 (7)에 적용하여 좌측 채널값과 우측 채널값을 구하게 된다. 식 (7)에서 1/(1+is_ratio)값을 표로 만들어 사용하면 우측 채널값은 표를 그대로 이용하여 구할 수 있고, 좌측 채널값은 입력된 좌측 채널값에서 우측 채널값을 빼는 연산으로 구할 수 있다.

$$
L_i = \frac{M_i+S_i}{\sqrt{2}},\ R_i = \frac{M_i-S_i}{\sqrt{2}} \tag{6}
$$

$$
\begin{aligned}
is''_{ratio_{sfb}} &= \tan(is''_{pos_{sfb}}\cdot\frac{\pi}{12}) \\
L_i &= L_i'\cdot\frac{is''_{ratio_{sfb}}}{1+is''_{ratio_{sfb}}}, \\
R_i &= L_i'\cdot\frac{1}{1+is''_{ratio_{sfb}}}
\end{aligned}
\tag{7}
$$

3. Antialias 과정

Antialias 과정에서는 식 (8)과 같이 정의되는 cs와 ca의 계산을 위하여 8개의 원소를 갖는 표가 2개 필요하다. 이 과정에서는 그림 4와 같은 butterfly를 각 서브밴드에 대하여 8번씩 수행하게 된다. 원래 제공되는 C 소스 프로그램은 하나의 서브밴드에 대하여 cs와 ca값을 8번 로딩하게 된다. 이것은 내부 반복문에서 수행되기 때문이며 수행속도의 저하를 초래한다. 이를 해결하기 위하여 하나의 butterfly를 모든 서브밴드에 대해 수행하도록 하여 로딩 회수를 줄였다.

$$
cs_i = \frac{1}{\sqrt{1+c_i^2}},\ ca_i = \frac{c_i}{\sqrt{1+c_i^2}} \tag{8}
$$

4. IMDCT 및 Synthesis Polyphase Filter Bank 과정

IMDCT 과정은 식 (3)을 이용하여 실제로 IMDCT를 수행하는 부분과 그 결과에 window를 곱하여 overlap-add하는 부분으로 나뉘어서 수행하

게 된다. 이 과정은 576샘플에 대해서 32번을 수행하기 때문에 연산 속도를 증가시키기 위한 알고리즘이 실시간 구현을 위해서는 반드시 필요하다. 일반적으로 DCT 연산에 대한 고속화 알고리즘으로는 factorization 방법[11]과 Fast Discrete Cosine Transform(FDCT) 방법[12] 등이 있다. Short window의 경우에는 IMDCT의 수가 12이며 long window의 경우에는 36인데, 이것은 2의 지수승이 아니기 때문에 2의 지수승을 요구하는 알고리즘인 FDCT 방법만으로는 효과적으로 연산 속도를 증가시킬 수 없다. 따라서 factorization 방법과 FDCT 방법을 모두 적용하여 IMDCT 과정의 연산 속도를 증가시키도록 하였다. Short window인 경우에는 3-point에 대해서 factorization을 수행하고 그 결과를 가지고 FDCT에 적용하였으며, long window인 경우에는 9-point에 대해서 factorization을 수행하고 그 결과를 FDCT에 적용하였다.

Synthesis Polyphase Filter Bank는 전체 MPEG 오디오 디코딩 과정에서 가장 연산량이 많이 소요되는 모듈로서, 연산량을 최대한 줄임으로써 전체 MPEG-1 Layer III 오디오 디코딩 과정의 수행시간을 상당히 단축시킬 수 있다. 합성 필터는 IDCT 연산과 유사한 식 (9), (10)에 의해서 구해진다. 그림 7은 Synthesis Polyphase Filter Bank의 처리 과정을 보이고 있다.

$$N_{ik} = \cos[(16+i)(2k+1)\frac{\pi}{64}], \quad \text{for } 0 \le i \le 63, 0 \le k \le 31 \tag{9}$$

$$V[i] = \sum_{k=0}^{31} N_{ik} S_k \tag{10}$$

합성 과정에서 매 프레임당 식 (10)을 72번 반복 수행해야 하므로 실시간 구현을 위해서는 반드시 고속 알고리즘의 적용이 필요하다. 이를 위하여 앞에서 언급한 IMDCT 과정에서 수행한 것과 마찬가지로 FDCT 방법을 적용하여 수행속도를 증가시킬 수 있도록 하였다. 그리고 cosine 함수를 사용하고 있기 때문에 cosine 함수의 대칭성을 이용하여 64개의 V[i] 대신에 32개의 V[i] 값만으로 연산이 가능하게 하였다. 이때 사용되는 각 V[i]의 관계는 식 (11)에 나타나 있다.

$$\begin{aligned} V[i] &= -V[32-i] \quad i=0,1,\cdots,15 \\ V[i+32] &= V[64-i] \quad i=1,2,\cdots,15 \\ V[16] &= 0 \\ V[48] &= -\sum S_k \quad k=0,1,\cdots,31 \end{aligned} \tag{11}$$

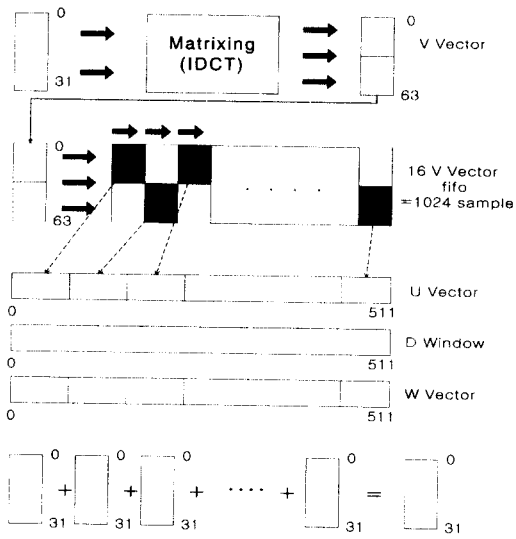


그림 7. Synthesis Polyphase Filter Bank의 처리 과정

V. MPEG-1 Layer Ⅲ 오디오 디코더의 실시간 DSP 구현 결과

이 장에서는 4장에서 설명한 방법으로 구현한 프로그램을 이용하여 고정소수점 연산 결과를 부동소수점 연산 결과와 비교하여 고정소수점 연산 결과의 정확성을 평가하고 최적화 전의 프로그램과 제시한 알고리즘을 적용한 프로그램간의 사이클 수를 측정하여 최적화 정도를 비교하였다. 고정소수점으로 연산한 결과에 대한 최종 출력 PCM값의 정확성을 확인하기 위하여 PC에서 부동소수점으로 연산한 결과와 비교해 보았다. 실험 비트스트림으로는 표 1에 주어진 것처럼 Fraunhofer IIS의 MPEG-1 Layer III 실험 비트스트림을 사용하였다.

표 1. 실험에 사용된 비트스트림

	bit rate	채널	sample rate
비트스트림 A (he_48khz.bit)	32kbps에서 320kbps로 가변	모노	48kHz
비트스트림 B (compl.bit)	64kbps	모노	48kHz
비트스트림 C (hecommon.bit)	128kbps	스테레오	44.1kHz

표 2는 3가지 실험 비트스트림에 대한 결과를 비교한 것이다. 이때 SNR은 식 (12)로 정의되며,

PCM_{fixed} 는 고정소수점 연산으로 계산된 최종 PCM 샘플값을 가리키고 PCM_{float} 는 부동소수점 연산으로 계산된 최종 PCM 샘플값을 나타낸다. 표 2에서 보듯이 SNR은 전체적으로 상당히 높게 나타나고 실제로 청취하더라도 부동 소수점 연산 결과와 차이점을 거의 느낄 수 없으며 파형에서도 그 차이를 볼 수 없다.

$$SNR[dB] = 20 \times \log_{10} \frac{|PCM_{fixed}|}{|PCM_{float} - PCM_{fixed}|} \quad (12)$$

표 2. 테스트 비트스트림에 대한 오차 및 SNR 비교

	최대 오차값	최대 오차 비트	SNR[dB]
비트스트림 A	2	2	66.5737
비트스트림 B	3	2	65.4235
비트스트림 C	L:4 R:12	L:3 R:4	L:61.4032 R:61.5354

구현된 디코더에 대한 성능 검증을 위해서 메모리 크기와 수행 속도 부문에 대하여 비교·분석하였다. 본 논문에서는 MPEG-1 Layer III 오디오 디코더의 성능만을 평가하기 위하여 사운드 입출력과 호스트 통신에 대한 과정은 제외시키고 측정하였다. 디코더의 성능을 평가하기 위하여 아래와 같이 정의한 3가지 경우에 대해 비교하였으며, 표 3은 사용된 메모리의 크기를 비교한 것이다.

- 디코더 A : 부동소수점 프로그램을 수정하여 고정소수점으로 구현한 디코더
- 디코더 B : 디코더 A에 최적화 알고리즘을 적용한 디코더
- 디코더 C : 디코더 B에 intrinsic을 사용하여 구현한 최종 디코더

표 3에서 보면 알 수 있듯이 디코더에 최적화 알고리즘과 intrinsic을 모두 적용함으로써 데이타 ROM의 크기와 데이타 RAM의 크기를 상당히 많이 줄일 수 있었다[13,14]. 디코더 B는 수행속도를 증가시키기 위하여 unrolling 과정을 첨가하여 다소 프로그램 ROM의 크기가 증가하였다. 그러나 디코더 A와 C를 비교해 보면 프로그램 ROM의 크기가 약간 늘어난 반면 데이타 메모리의 크기는 아주 많이 줄어듬을 알 수 있다.

표 3. 메모리 크기 비교 [kbyte(kWord)]

	디코더 A	디코더 B	디코더 C
프로그램 ROM	21.10 (5.27)	37.73 (9.43)	27.07 (6.77)
데이타 ROM	63.40 (15.85)	12.61 (3.15)	12.53 (3.13)
데이타 RAM (stack 포함)	107.73 (26.93)	45.92 (11.48)	39.77 (9.94)

수행 속도 면에서는 각 디코더에 대해서 프레임당 수행 사이클수를 측정함으로써 최적화의 정도를 확인하였다. 표 4는 비트스트림 A의 경우에 대해 하나의 그래뉼을 처리하는데 걸리는 수행 시간을 보여주는 것이다. 특히, MPEG-1 Layer III 오디오 디코더는 IMDCT와 Synthesis Polyphase Filter Bank의 성능이 전체 오디오 디코더의 성능을 좌우할 만큼 중요하기 때문에 두 모듈에 대해서 비교해 보았다. 표 4를 보면 디코더 A에 비해서 디코더 C의 성능이 아주 많이 개선되었음을 알 수가 있다.

표 5는 디코더 C로 single 채널인 비트스트림 A,

표 4. 그래뉼당 IMDCT와 Synthesis Polyphase Filter Bank 수행 시간 [cycles]

		디코더 A clock(msec)	디코더 B clock(msec)	디코더 C clock(μsec)	디코더 C /디코더 A
IMDCT	Avg.	66,491,880.4 (332)	776,269.3 (3.88)	32,908.0 (164.54)	0.05 %
	Max.	67,107,111 (336)	776,345 (3.88)	32,908 (164.54)	0.05 %
	Min.	65,524,679 (328)	776,262 (3.88)	32,908 (164.54)	0.05 %
Synthesis Polyphase Filter Bank	Avg.	148,802,948.0 (744)	2,885,735.8 (14.43)	192,763.5 (963.82)	0.13 %
	Max.	149,311,662 (747)	2,886,026 (14.43)	192,826 (964.13)	0.13 %
	Min.	146,747,610 (734)	2,885,707 (14.43)	192,763 (963.82)	0.13 %

표 5. 프레임당 수행 시간 비교 [cycles]

	비트스트림 A (msec)	비트스트림 B (msec)	비트스트림 C (msec)
Average	679,861.2 (3.40)	733,619.8 (3.67)	1,255,087 (6.28)
Maximum	819,268 (4.10)	835,045 (4.18)	1,257,393 (6.29)
Minimum	593,794 (2.97)	648,401 (3.24)	1,235,316 (6.18)

B와 스테레오 채널인 비트스트림 C에 대하여 한 프레임을 처리하는데 필요한 사이클 수와 이에 해당하는 시간을 나타내고 있다. 최대 sampling rate인 48 kHz에서 한 프레임의 시간은 24 msec이고 디코딩 연산에 걸리는 시간은 최대 6.29 msec이므로 한 프레임 수행 연산량은 TMS320C6201 DSP가 수행할 수 있는 최대 연산량의 26.21%에 해당된다.

Ⅵ. 결 론

본 논문에서는 TMS320C6201 EVM 보드에서 실시간으로 동작하는 고정소수점 MPEG-1 Layer III 오디오 디코더를 구현하였다. ISO/IEC에서 제공하는 부동소수점 C 프로그램을 TMS320C6201 DSP에서 동작하도록 고정소수점 연산 프로그램으로 변환하고 실시간 동작을 위하여 최적화 작업을 수행하였다. 그리고 호스트 통신을 통하여 비트스트림을 호스트로부터 받도록 하였으며 DMA를 이용하여 사운드를 출력함으로써 CPU의 부담을 줄이면서 디코딩 결과를 출력하도록 하였다. 부동소수점 연산을 수행하는 PC 프로그램의 최종 출력 PCM값과 구현된 고정소수점 연산의 최종 출력 PCM값을 비교하여 60 dB 이상의 높은 SNR을 가짐을 확인함으로써 고정소수점 연산의 정확성을 검증하였다. 그리고 오디오 디코더의 연산량과 메모리 크기를 줄이기 위하여 IMDCT 모듈과 Synthesis Polyphase Filter Bank 모듈 등에 중점적으로 최적화 알고리즘을 적용한 결과 TMS320C6201 DSP가 수행할 수 있는 최대 연산량의 26.21%만으로 충분히 실시간으로 동작함을 확인하였다.

참 고 문 헌

[1] T. Painter, A. Spanial, "A Review of Algorithms for Perceptual Coding of Digital Audio Signals," *IEEE 13th International Conference on Digital Signal Processing Proceedings*, DSP 97., Vol. 1, pp. 179 - 208, 1997

[2] P. Noll, "MPEG Digital Audio Coding," *IEEE Signal Processing Magazine*, Sep., 1997

[3] D. Pan, "A Tutorial on MPEG/Audio Compression," *IEEE Trans. on multimedia*, Vol. 2, No. 2, pp. 60-74, 1995

[4] S. Shlien, "Guide to MPEG-1 Audio Standard," *IEEE Trans. on Broadcasting*, Vol. 40, No. 4, pp. 206 - 218, 1994

[5] ISO/IEC 11172-3 "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 3 : Audio," First edition, 1993

[6] J. Princen, A. Bradley, "Analysis/ Synthesis Filterbank Design Based on Time Domain Aliasing Cancellation," *IEEE Trans. on Acoust. Speech, and Signal Processing*, Vol. ASSP-34, pp. 1153-1161, 1986

[7] Texas Instruments, *TMS320C6201 : Fixed Point Digital Signal Processor*

[8] Texas Instruemtns, *TMS320C6201 CPU and Instruction Set Reference Guide*

[9] Texas Instruments, *TMS320C6201/ C6701 Evaluation Module Technical Reference Guide*

[10] 김진원, 정남훈, 김준석, 이근섭, 이충용, "MPEG-1 계층 III 오디오 복호화기의 VLSI 설계," *제 12회 신호처리합동학술대회*, pp. 847-850, 1999

[11] http://www.iro.umontreal.ca/~boyerf /IMDCT/

[12] Byeong Gi Lee, "A New Algorithm to Compute the Discrete Cosine Transform," *IEEE Trans. on Acoust, Speech and Signal Processing*, Vol. ASSP-32, No. 6, pp. 1243 - 1245, 1984

[13] Texas Instruments, *TMS320C62xx Programmer's Guide*, 1997

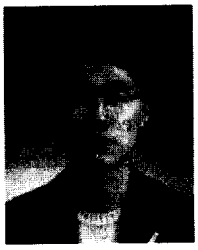
[14] Texas Instruments, *TMS320C6000 Assembly Language Tools User's Guide*

권 홍 석(Hong-seok Kwon)

1997년 2월 : 경북대학교
전자공학과 졸업
1999년 2월 : 경북대학교
전자공학과 석사
1999년 3월~현재 : 경북대학교
전자공학과 박사과정

<주관심 분야> 음성신호처리, 디지털 신호처리,
오디오 코딩 등

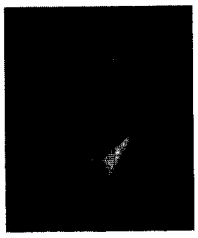
김 시 호(Si-ho Kim)



1998년 2월 : 경북대학교
전자공학과 졸업
1998년 3월~현재 : 경북대학교
전자공학과 석사과정

<주관심 분야> 신호처리, 디지털 통신, 오디오 코딩

배 건 성(Keun-sung Bae)



1977년 2월 : 서울대학교
전자공학과 졸업
1979년 2월 : 한국과학기술원
전기및전자공학과 석사
1989년 5월 : University of
Florida 공학박사

1979 3월~현재 : 경북대학교 전자·전기공학부 교수
<주관심 분야> 음성분석 및 인식, 디지털 신호처리,
디지털 통신 음성 부호화, 웨이브렛 이론
등