TMS320C541 DSP를 이용한 MP3 디코더 구현

Implementation of MP3 decoder with TMS320C541 DSP

윤 병 우*
Byungwoo Yoon

요 약

MPEG-1 오디오 표준은 고음질 디지털 오디오 신호의 압축 알고리즘이다. 이 표준은 인코더와 디코더의 기능을 규정하고 있고, 인코더와 디코더 알고리즘의 복잡도와 성능에 따라 세 가지 다른 계층으로 분류된다. 본 논문에서는 MPEG-1 오디오 계층3(MP3) 디코더를 고정소수점 DSP인 TMS320C541 칩으로 구현하였다. MP3 알고리즘은 인간의 청각구조의 심리음향 특성을 이용하는 알고리즘으로 인간의 귀에 들리지 않는 주파수의 성분은 미리 제거함으로써 데이터의 량을 줄이면서 음질의 손실을 최대한 줄이는 알고리즘이다. 이 알고리즘은 다이나믹 레인지가 매우 크기 때문에 고정 소수점으로 구현하기가 쉽지 않다. 본 연구에서는 가중 참조표를 적용하여 계산량을 줄이고 다이나믹 레인지 문제를 해결함으로써 고정 소수점 DSP칩을 이용하여 실시간 시스템을 구현하였다.

Abstract

MPEG-1 audio standard is the algorithm for the compression of high-quality digital audio signals. The standard dictates the functions of encoder and decoder pair, and includes three different layers as the complexity and the performance of the encoder and decoder. In this paper, we implemented the real-time system of MPEG-1 audio layer III decoder(MP3) with the TMS320C541 fixed point DSP chip. MP3 algorithm uses psycho-acoustic characteristic of human hearing system, and it reduces the amount of data with eliminating the signals hard to be heard to the hearing system of human being. It is difficult to implement MP3 decoder with fixed point DSP because of it's broad dynamic range. We implemented realtime system with fixed DSP chip by using weighted look-up tables to reduce the amount of calculation and solve the problem of broad dynamic range.

Key Words : MP3, Psychoacoustic, subband, IMDCT, Audio, Decoder

I. 서 론

MPEG1 오디오 표준은 인코딩 또는 디코딩 알고리즘의 성능 및 복잡도에 의해 분류된 3가지 레이어로 이루어져 있다. 이 레이어들은 MPEG1 오디오에서 가장 기본적인 오디오 코딩/디코딩 알고리즘으로 구성된 레이어 1과, 개선된 코딩 방식이나 추가적인 알고리즘으로 성능 및 복잡도가 증가된 레이어 2, 레이어 3으로 분류된다[1,4,6].

*경성대학교 전기 전자 컴퓨터 공학부
접수 일자 : 2003. 3. 05 수정 완료 : 2003. 7. 24
논문 번호 : **2003-2-4**
※본 논문은 2000년도 경성대학교 학술지원연구비에 의하여 연구되었음.

레이어 1은 테이프 또는 디스크에 디지털 데이터를 기록하는 목적으로 사용되며, MPEG1 오디오에서 가장 기본적인 알고리즘으로 구성되어 있다. 레이어 2는 레이어 1에서 사용된 기본적인 오디오 코딩 알고리즘을 기술적으로 보완하여 신호의 불필요한 성분을 더 많이 없애고, 보다 효율적인 방법으로 심리음향을 적용한다. 레이어 3은 레이어 2에 비해 여분의 신호들을 더욱더 억압하고, 필터를 이용하여 희미하게 들리는 가청주파수의 적출을 개선한 알고리즘이다.

MPEG 오디오 압축은 심리음향 코딩에 기반을 두고 있다[3]. 심리음향 코딩과정에서 코덱은 인코딩한 신호를 디코딩 했을 때 원래 신호와 동일한 신호를 유지하기보다는 인간의 귀를 통해 출력 신호를 들었을 때 원래신호와 최

대한 비슷하게 들리도록 하는 것을 목적으로 한다. 본 연구에서는 MPEG-1 알고리즘의 레이어 3인 MP3 알고리즘의 디코더를 TMS320C541[7,8]0 정수형 DSP를 이용하여 실시간 시스템을 구현하였다.

II. MP3의 오디오 압축 원리

MP3 알고리즘은 심리음향 모델을 이용함으로써 인간의 귀에 들리지 않는 신호를 없앤 후 코딩을 한 것으로 다음과 같은 몇 가지 특징을 이용한다.

MP3 알고리즘은 신호들 중에서 인간의 귀에 들리지 않는 신호들을 마스킹 함으로써 데이터의 량을 줄인다. 인간의 청각시스템에서 들리지 않는 신호를 억압하기 위해서는 인간의 귀와 유사한 특성을 갖는 시스템을 모델링해야 한다. 이를 위해 인코더에서는 심리음향 모델을 이용한다[3]. 심리음향 모델은 입력 신호를 몇 개의 연속적인 블록을 통해 해석하고 각각의 블록들에 대한 신호의 스펙트럼을 결정한다. 그리고 인간 청각시스템의 마스킹 특징을 모델하고 최소 가청 레벨을 추정한다.

데이터의 량을 줄이기 위해 MP3 포맷에서는 몇 가지의 방법을 사용한다. 인간의 귀는 각 주파수에 따라 서로 다른 가청한계를 갖는다. 이것을 최소 가청한계라고 하며, 이러한 한계 이하의 신호들은 인간의 귀가 인식을 못하기 때문에 인코딩 할 필요가 없다. 또한, 인간의 귀는 강한 소리들 속에 있는 약한 신호는 감지를 하지 못한다. 이와 같이 비록 최소 가청한계 이상의 세기를 갖는 신호이지만 주변의 소리 때문에 인간의 귀가 감지하지 못하는 신호는 인코딩 할 필요가 없다. 이것을 위해서 MP3 인코더는 인간 귀의 특성과 같은 심리음향 모델을 사용하는데 이 것을 마스킹효과라고 한다[1,3].

스테레오 신호에 대해서 MP3 포맷은 파일의 크기를 줄이기 위해 JS(joint stereo)라고 하는 몇 가지 기법을 이용한다. 일반적으로 hi-fi 세트에서는 한 개의 서브우프를 갖고 있다. 그러나 우리는 그 소리가 우퍼로부터 들린다기보다는 인접한 스피커들에서부터 들리는 것처럼 느낀다. 사실 인간의 귀는 주파수가 매우 낮거나 매우 높은 신호들에 대해서는 위치를 정확하게 인식하지 못한다. 따라서 MP3 포맷에서는 IS(intensity stereo)라는 트릭을 사용하여 이러한 신호들을 처리한다. 즉, 어떤 주파수 성분들을 모노포닉 신호로 녹음하고 공간적으로 복원하기 위해서 몇 가지 정보를 추가함으로써 데이터의 량을 줄인다. 또 다른 JS 기법은 M/S(mid/side) 스테레오로서 좌 우 채널이 매우 유사할 경우 좌 우 채널 대신 중간(L+R)과 사이드(L-R) 채널로 인코딩을 하여 사이드채널의 비트 수를 줄임으로써 최종 파일의 크기를 줄인다. 재생 시에는 중간과 사이드 채널로부터 MP3 디코더는 좌 우 채널을 복원한 후 재생을 한다[1].

MP3 알고리즘은 또한 허프만(Huffman)코딩 기법을 이용한다. 이 코딩 기법은 확률이 높은 심볼들의 코드 길이를 짧게 할당함으로써 전체 데이터의 량을 줄이는 방법이다. 큰 폴리포닉에서는 많은 음들이 마스킹 됨으로서 심리음향 코딩이 매우 효율적이지만 반복되는 정보가 적기 때문에 허프만 알고리즘이 효율적이지 못하다. 반면 단순한 음들에서는 마스킹효과들이 별로 없지만 양자화된 음에서 반복되는 바이트들이 많기 때문에 허프만 코드의 길이가 짧아지게 됨으로써 전체 코드의 길이가 짧게 된다. 따라서 심리음향 코딩 기법이 이상적인 상호 보완의 결과를 제공한다.

III. MP3 복호화 알고리즘 개요

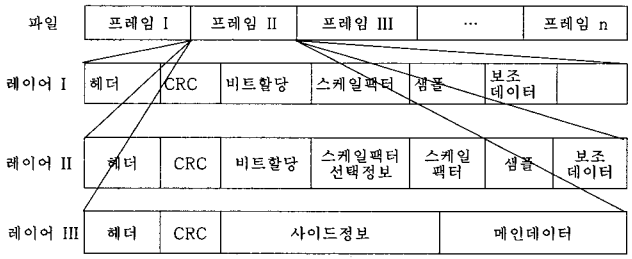

그림 1. MPEG 오디오의 레이어별 데이터 비트스트림 구조
Fig. 1. Bit stream structure of MPEG audio as the layers

MP3 비트스트림은 프레임의 연속으로 이루어져 있다. 한 프레임에는 32비트 헤더와 16비트 CRC, 사이드 정보, 메인 데이터로 구성되어 있다. 그림 1은 MPEG-1 오디오의 레이어별 비트스트림 구조의 차이를 나타내고 있다.

헤더는 32비트의 고정된 영역으로의 레이어의 종류와 샘플링 주파수, 모드(모드/스테레오)등의 정보를 가지고 있으며 이러한 정보들은 CRC와 프레임 정보, 메인 데이터의 크기를 계산하는 데 있어서 필수적인 정보를 제공한다. CRC는 오류 검출 코드로서 선택적이며 비트 스트림의 헤더 다음에 16비트 크기로 정의된다. 사이드 정보는 헤더에 정의된 모드에 따라서 17바이트 또는 32바이트의 크기를 가지며 압축된 오디오 데이터에 대하여 MP3 디코딩 알고리즘을 적용하는데 있어서 중요한 정보를 제공한다.

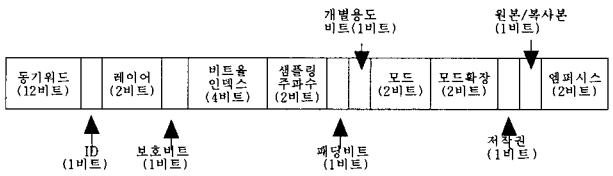

그림 2. MPEG 오디오 헤더의 구조
Fig 2. Head structure of MPEG/audio

그림 2는 MPEG 오디오 알고리즘의 헤더의 구조를 나타내고 있다. 이것은 MPEG-1 오디오 알고리즘의 모든

레이어에 공통적으로 적용된다. MP3 알고리즘의 프레임은 동기워드(12bit:Oxfff)에서부터 시작된다. MP3 알고리즘의 헤더의 구성을 요약하면 다음과 같다.

ID(1)- MPEG 오디오
보호비트(1)- CRC 적용 여부
레이어(2비트) - 레이어 I, 레이어 II, 레이어 III를 나타냄
비트율 인덱스(4비트) - 모두 0인 경우는 자유포맷 조건이고 나머지는 32~448kbps를 표현 함
샘플링주파수(2비트) - 38kHz, 44.1kHz, 그리고 48kHz로 이루어짐
패딩비트(1)- 추가 프레임 슬롯 (44.1 kHz에 적용)
개별용도비트(1)- 사용되지 않음
모드(2비트) - 스테레오, 조인트스테레오, 듀얼 또는 싱글 채널
모드확장(2비트) - 모드 확장에 사용되는 비트
저작권(1)- 저작권 유무 설정
원본/복사본(1)- 비트 스트림에 대한 복사 유무
엠퍼시스(2비트) - 엠퍼시스 특성을 나타냄

메인데이터앤드	private bits	scfsi info	그래뉼

그림 3. 사이드 정보의 구조
Fig. 3. Structure of side information

MP3 알고리즘에서 사이드 정보 블록의 구조는 그림 3과 같다. 사이드 정보는 메인 데이터의 끝을 나타내는 메인 데이터 앤드, 개인적 용도로 사용되는 프라이비트 비트, 메인 데이터의 스케일 팩터들에 대한 정보들을 나타내는 scfsi info(scalefactor select information), 그리고 허프만 코딩된 데이터들에 대한 정보 및 디코딩에 필요한 나머지 정보들을 가지고 있는 그래뉼로 구성되어 있다.

메인데이터							
그래뉼 0				그래뉼 1			
Ch 0		Ch 1		Ch 0		Ch 1	
SCF	Huffman	SCF	Huffman	SCF	Huffman	SCF	Huffman

그림 4. 메인 데이터의 구조
Fig. 4. Structure of main data

MP3 비트스트림의 마지막 부분인 메인 데이터의 구조는 그림 4와 같다. 메인 데이터는 2개의 그래뉼로 구성되며 그래뉼 들은 각각 두 개의 채널(CH0/CH1)로 나누어진다. 각 채널은 양자화 잡음을 특징짓는 스케일 팩터(SCF)와 실제 데이터가 코딩된 허프만 코드로 구성된다. 허프만 데이터는 양자화된 값들 중 절대값 15보다 작은 값들이 쌍(x,y)으로 코드화 된 것이며, 15보다 크면 ESC 코드가 이 값의 플래그로 사용된다.

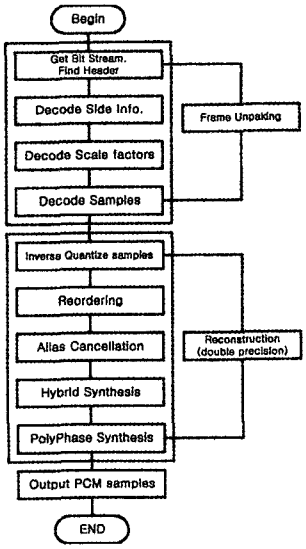

그림 5. MP3 디코딩 흐름도
Fig. 5. MP3 decoder flow chart

MP3의 디코딩 과정은 입력되는 비트 스트림으로부터 헤더, 사이드정보와 메인 데이터를 추출(unpacking)하는 과정으로 시작된다. 그림 5는 MP3 디코더의 알고리즘의 전체 흐름도이다.

디코딩 알고리즘은 먼저 비트 스트림으로부터 12비트의 동기워드를 찾음으로써 프레임의 시작을 알 수 있고, 사이드 인포메이션으로부터 디코딩 과정에 필요한 모든 정보들을 찾아내는데, 여기서 찾아낸 정보를 이용하여 메인 데이터에서 스케일 팩터와 허프만 샘플들을 추출하고, 이 후 나머지 정보들은 오디오 샘플을 재구성하는 단계에 이용된다.

재구성(reconstruction)과정에서는 추출된 각각의 정보와 데이터를 이용하여 역 양자화 과정을 통하여 32개의 서브밴드로 구성되는 주파수 영역의 오디오 데이터를 재구성한다. 이 때, 오디오 데이터의 극심한 변화가 예상되는, 상관 관계가 낮은 블록에 대한 재 정렬 후 블록들 간의 앨리아싱을 방지하고, IMDCT(Inverse Modified Discrete Cosine Transform)와 서브밴드 합성 및 필터링을 거쳐서 최종 오디오 데이터로 출력된다. MP3 알고리즘에서는 IMDCT를 이용하여 신호에 대한 추가적인 주파수 분해능을 높일 수 있는데[2,3], 오디오 데이터의 연속된 샘플링에 대하여 상관 관계가 낮은 부분에 대한 단블록(12샘플) IMDCT처리 과정은 장블록(36 샘플) 과정에 비해 압축효과는 다소 떨어지지만 피크 입력과 같은 극심한 변화가 예상되는 부분에서 음질저하 현상을 막을 수 있다.

Ⅳ. 고정 소수점 방식의 MP3 디코더 모의실험

본 연구에서 구현한 MP3 디코더에 사용된 TMS320C541 DSP는 고정소수점 방식의 신호처리 전용칩으로 최대 64K 워드의 메모리 자원과 40MIPS의 연산 처리 속도를 가지고 있다[7,8]. 최종적인 실시간 MP3 디코더를 구현하기 앞서 연산 방식에 따른 알고리즘의 적용 가능성 여부와 성능 및 실시간 처리를 위하여 DSP에 대한 정수형 시뮬레이터의 구현을 통해 고정 소수점 연산방식에 대한 MP3 디코딩 알고리즘을 검증하였고, 최종적으로 TMS320C541 어셈블리 언어를 이용하여 알고리즘을 설계하여 시스템에 탑재하였다. 따라서, MP3 디코더의 정수형 시뮬레이터는 PC 윈도우 95 환경에서 C언어를 이용하여 구현하였고, 성능 평가는 MPEG 기구의 부동 소수점 연산 방식 디코더와의 비교를 통해 검증하였다. 또한 복잡한 알고리즘을 보다 간략화하기 위하여 각 부호화 블록별로 모듈화 시켜서 분류, 정리하여 성능 평가와 디버깅시 쉽게 접근할 수 있도록 하였다. 또한, 연산 속도를 높이기 위하여 고속 알고리즘의 적용 및 테이블화와 반복 루프를 풀어 전개함으로써 실시간 처리 속도 및 속도 개선 면에서 우수한 성능을 발휘하도록 하였다. TMS320C541의 연산은 기본적으로 16비트로 처리되기 때문에 보다 높은 해상도를 구현하기 위하여 2워드 즉, 32비트의 변수를 사용하였다. 그림 6은 앞에서 설명한 과정을 적용한 MP3 디코더의 정수형 시뮬레이터를 구현하는 과정을 도시한 것인다.

MPEG-1/audio 표준안에 제시된 MP3 디코더 신호처리 알고리즘들에 적용되는 테이블들은 허프만 테이블, 합성윈도우 계수, 스케일 팩터 밴드 등이 있다. 이 들 중에서 합성윈도우 계수는 정수형 시뮬레이션을 하기 위한 정수형 데이터로의 변환이 필요하다. 따라서, 모든 계수들에 대한 정규화 과정을 통해 32비트 배정도 연산에 필요한 정수형 데이터들로 형을 변환시키고, 이들 데이터들은 Q31포맷으로 소수점의 위치를 결정하였다.

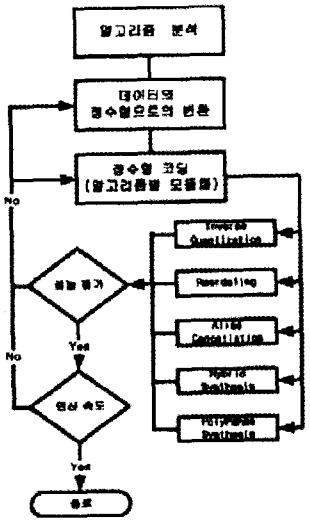

그림 6. 구현된 MP3 디코더 시뮬레이터의 흐름도
Fig. 6. Flow chart of implemented MP3 decoder simulator

4.1 구현한 알고리즘의 성능 평가

본 논문에서 구현된 정수형 시뮬레이터의 성능을 평가하기 위하여 ISO/MPEG 오디오 표준의 비트스트림을 시료로하여 부동소수점 연산방식의 MP3 디코더로부터 출력된 데이터를 비교해 보았다. 비교 평가에 사용된 비트스트림들은 레이어 3에서 지원 가능한 모든 파라미터들을 사용하여 합성한 신호들로서 다음에 열거된 신호들로서 구성되었다.

1) 헤더를 구성하는 파라미터
2) 각 샘플링 주파수(32, 44.1, 48kHz)의 모든 비트레이트
3) 지원 가능한 스테레오 모드 (mono/stereo /MS-stereo /Intensity-stereo)
4) 프레임 정보(side information)
5) 블록모드(long, start, short/mixed, stop)
6) 34가지(count1 테이블 포함) 허프만 코드
7) 1kHz 0dB의 Sine 신호
8) 10Hz - 10kHz/-20dB Sine Sweep 신호

시뮬레이터의 성능 평가 결과 부동 소수점 연산방식의 MP3 디코더와 본 논문에서 구현된 고정 소수점 연산방식 MP3 시뮬레이터의 성능차이는 스테레오 모드의 테스트에서 가장 낮은 SNR 88dB의 결과를 보였으나, 전체적으로 88~94dB의 SNR을 나타내었다.

전형적인 신호처리의 과정인 변환 및 필터링에서 반복되는 곱셈과 덧셈의 계산과정은 매우 많은 연산량을 차지한다[4-6]. 따라서, 이들의 복잡도 및 성능 개선의 여부에 따라 실시간 시스템으로서의 기능이 좌우된다고 할 수 있다. 표 1에서는 MPEG 오디오 표준안에 제시되어 있는 알고리즘 식들에 대하여 테이블 및 고속 알고리즘을 적용하여 구현된 실시간 디코더와 기존의 식을 그대로 사용하여 구현된 디코더와의 실행 속도를 나타내고 있다.

표 1. 알고리즘 구현방식에 따른 속도 비교
Table. 1. Comparison of excution time as implementation method of algorithms

분류 시스템	(1) 기존의 디코더 (sec)	(2) 구현된 디코더 (sec)
80586 - 166MHz	115	24
MMX - 166MHz	74	24
Pentium2 - 350MHz	33	24

Test bitstream : 64kbit, 44.1KHz, 1853 frames

표 1에서 나타난 바와 같이 MPEG 표준안에 제시된 알고리즘 식을 그대로 사용하여 구현된 디코더는 실시간 디코더에 대해 최대 4배까지의 프로세싱 속도 차이가 난다. 따라서, 일련의 과정들에서 MP3 알고리즘들에 대한 속도개선의 방법을 연구 분석하였고, 실시간 시스템 구현의 전 단계인 최적화 된 정수형 시뮬레이터를 구현하기 위하

여 고정 소수점 연산 방식에 대한 속도의 개선과 MP3 디코더의 안정된 성능을 결합하여 최적화 된 정수형 MP3 시뮬레이터를 구현하였다.

MP3 디코더에 적용되는 IMDCT[1]는 식(1)과 같다.

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

for $i=0 \ldots n-1$

이것의 역 변환에 사용되는 각 서브 밴드들의 입력 샘플들은 인코더에서 윈도우잉된 샘플 수에 따라 6 또는 18 포인트 IMDCT가 취해져 12×3 또는 36개의 시간 영역의 샘플들로 출력된다. 위 식에서 입력 샘플 X_k와 코사인 항목들에 대한 실제 MP3 알고리즘에 적용되는 IMDCT의 연산량을 계산해보면, 입력 18 샘플에 대한 36개의 출력 샘플이 만들어지기까지의 연산량은 곱셈 648(18×36), 덧셈 612(17×36)번의 연산이 필요하다. 따라서, 1개의 서브 밴드 블록을 출력하기 위해 사용된 많은 곱셈 및 덧셈 연산량을 줄이기 위하여 고속 알고리즘[5]을 적용하였고 그 결과 1개의 서브밴드인 18샘플 당 약 36번의 곱셈과 90번의 덧셈 연산량을 줄였다.

32개로 나누어진 서브밴드들에 대한 합성 필터의 설계에 있어서 MPEG 표준안에서는 연산량의 감소를 위하여 그림 7의 흐름과 같이 부분적 최적화를 시켰다. 그림 7로부터 각 서브밴드 당 1개의 샘플 즉, 32개 샘플의 합성에 필요한 연산량을 계산해보면 2,560(512 + 32×64)번의 곱셈과 2,464(31×64 + 15×32)번의 덧셈이 필요하다.

$$V[i] = \sum_{k=0}^{31} \cos\left(\frac{(2i+1)(k+16n)}{64}\right) S_k \quad (2)$$

for $i=0,\cdots,63$

이 때, 메트릭싱 블록에서는 식 (2)를 이용하여 32개의 입력 샘플에 대한 새로운 64개 샘플이 만들어지는데, 이 식을 이용하여 코사인 함수의 대칭성과 고속 알고리즘의 구현을 통하여 70% 이상의 곱셈과 덧셈의 연산량을 감소시킨 서브 밴드 합성 필터를 구현할 수 있었다.

다음의 식(3)은 MPEG 표준안에 제시된 역 양자화 알고리즘에 적용되는 계산식[1]이다.

$$xr(i) = is(i)^{4/3} \times 2^{0.25(global\_gain-64-8*subblock\_gain)} \times 2^{0.25*(-2*(1+scalefac\_scalefac-2*preflag*(1+scalefac\_scale)*pretab)} \quad (3)$$

is(i) : Huffman decoding output

xr(i) : Requantization output

역 양자학 샘플들을 구하기 위한 식(3)을 정수형 DSP로 구현하려면 매우 많은 계산량을 요구하기 때문에 실시간 구현이 어렵다. 이 문제를 해결하기 위해서 본 연구에서는 다음과 같은 방법을 이용하였다.

식(3) xr(i)의 복잡한 계산식은 식(4)와 같이 $is(i)^{4/3}$과 gain, 그리고 scale의 3부분으로 나눌 수 있고, 허프만 디코딩 된 입력 데이터와 프레임 정보들의 값에 따라 몇 가지 크기의 테이블을 작성할 수있다.

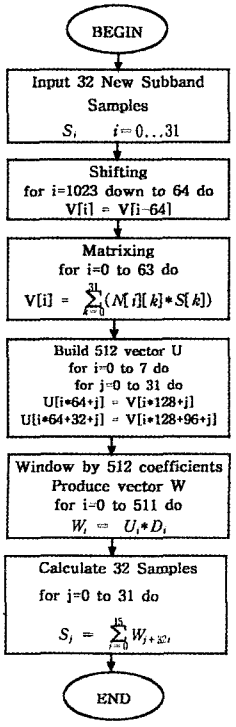

그림 7. 서브밴드 합성 필터의 흐름도

Fig. 7. Flow chart of subband synthesis filiter

$$xr(i) = is(i)^{4/3} \times gain \times scale \quad (4)$$

$$gain = 2^{0.25(global\_gain-64-8*subblock\_gain)}$$

$$scale = 2^{0.5*((1+scalefac\_scale)*(scalefac+preflag*pretab))}$$

테이블로 변환된 값들에 대한 정수형 데이터로 변환시 각 데이터들의 다이나믹 레인지에 대한 주의가 요구된다. 식(4)에서 gain의 경우 global_gain 이 0과 255의 최소/최대 값을 가질 때 gain의 값은 1.57×10e-16 ~ 2435.4961.. 의 값을 가진다. 이와 같이 매우 높은 다이나믹 레인지의 경우 32비트 배정도에 의한 정수형 데이터로의 변환은 사실상 불가능하다. 따라서, 이러한 경우 전체의 데이터를 몇 개의 블록으로 나누어 각 블록에 대하여 서로 다른 가중치를 적용하여 테이블을 만들어 사용함으로써 이 문제를 해결하였다. 또한 알고리즘의 구현 시 반복 구문을 사용하면 분기 명령어로 할당되는 시간으로 인하여 다른 명령에 비해 상대적으로 많은 처리시간을 필요로 한다. 따라서, 실시간 처리를 위한 방안으로 반복문을 최소화하여 처리시간을 최소화하였다.

V. TMS320C541 DSP를 이용한 실시간 시스템 구현

TMS320C541은 한 개의 프로그램버스와 세 개의 데이터 버스를 가진 강화된 하바드구조(enhanced harvard archtecture)의 고성능 DSP 칩[7,8]으로 고도의 병렬 연산 기능을 갖고 있다. 이것은 데이터 메모리에 대하여 두 개의 읽기와 한 개의 쓰기를 단일 사이클에 수행할 수 있는 고성능 DSP이다.

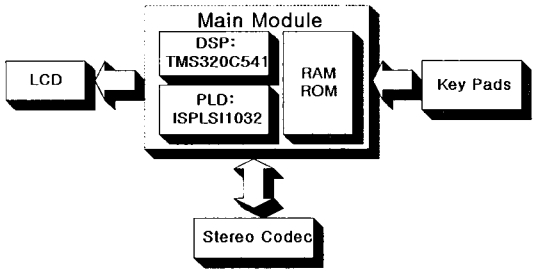

그림 8. 구현한 시스템의 블록도
Fig. 8. The block diagram of the implemented system.

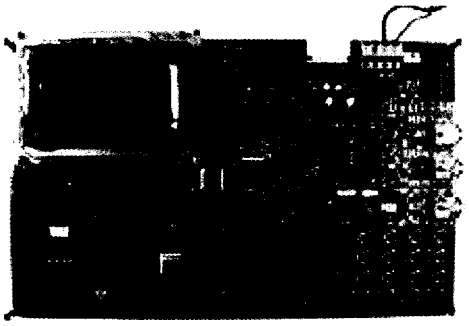

그림 9. 구현한 시스템의 사진
Fig. 9. The picture of the implemented system.

본 연구에서는 TI(Texas Instrument)사의 TMS320C541 DSP를 이용하여 MP3 디코더를 구현하였다. 그림 8은 본 연구에서 구현한 시스템의 블록도이고 그림 9는 구현한 시스템의 사진이다. 디코더의 알고리즘 구현을 위하여 먼저 앞 절에서 설명한 바와 같이 윈도우즈 환경 하의 PC에서 C언어를 이용하여 TMS320C541 DSP와 동일한 환경으로 정수형 시뮬레이션을 수행하였고, 이것을 바탕으로 TMS320C541 어셈블 프로그램을 구현하였다.

디버깅의 원활성을 위하여 EVM 보드를 PC에 장착하고 PC와 EVM보드의 인터페이싱 프로그램을 구현하였다. PC와 EVM보드 사이의 인터페이싱 프로그램을 구현함으로써 MP3에 적용되는 각 알고리즘 모듈들의 디버깅시 PC의 하드디스크에 있는 PCM 데이터를 EVM보드에 원활하게 전송하고, 또한 디코딩된 데이터를 PC에 저장하여 C를 이용한 정수형 시뮬레이션 결과와 비교를 해가면서 디버깅을 하였다.

TMS320C541 DSP의 어셈블리 언어는 산술 연산, 제어, I/O, 논리 연산, 읽기/쓰기, long-word 처리, 이동, 병렬 처리 명령어 집합으로 구성된다. 이들은 크게 192가지 단일 처리 명령어 집합과 13가지의 병렬처리 명령어 집합으로 나뉘어 지는데, MP3 디코딩 알고리즘은 192개의 단일 처리 명령어를 주로 하여 구현되었다.

MP3 디코딩 알고리즘의 연산에 있어서 오차를 최소화하고 데이터 값의 효율적인 해상도를 보장하기 위해서는 최소 32비트의 '배정도 연산이 요구된다. 따라서, TMS320C541 16비트 프로세서에 지원하는 배정도 값에 대한 가능한 몇 가지 long word 명령어와 기본적인 32 비트 소수점 값들에 대한 곱셈을 MACRO로 정의하여 MP3 알고리즘에 대한 연산을 수행하였다. 32비트 곱셈 연산이 알고리즘에 순수하게 적용될 때 최종 출력 값을 얻기 위한 시간은 최소 9사이클이 소요된다. 이 때, IMDCT 블록 또는 서브 밴드 합성 블록 등의 반복적인 덧셈 또는 곱셈 연산에서는 메모리 어드레싱을 변형하여 최소 6사이클로 줄일 수 있었다.

MP3 알고리즘에 있어서 디코더의 구조는 그림 5에서 도시하고 있는 바와 같이, 크게 프레임에 대한 정보를 추출하는 단계와 신호를 재구성하는 단계로 나눌 수 있다. 이 구조를 DSP로 구현할 때 배정도 연산의 적용 범위에 따라서도 그림 5와 같은 동일한 구조로 나눌 수 있다. 이렇게 나누어진 두 블록에 대해서 MP3 알고리즘을 구현하는 데 필요한 메모리 크기를 블록별로 계산할 수 있고, 계산된 메모리 블록, 즉 테이블의 크기에 대한 조정이 용이하게 된다. 따라서, 실시간 MP3 디코더의 구현에 있어서 프로그램 최적화 단계 중 테이블에 대한 부분 별 최적화를 가능하게 한다. 따라서, 블록화된 MP3 알고리즘의 구조에서 테이블의 크기들을 블록 별로 계산해 보면 다음과 같다.

프레임의 정보를 추출하는 구조는 비트 스트림으로부터 하나의 프레임을 구성하고 있는 헤더와 사이드 인포메이션, 메인 데이터를 추출하는 구조로 되어 있다. 이 때, 최종 PCM 오디오 데이터의 출력을 재구성하기 위해서는 프레임 정보를 이용하여 메인 데이터로부터 스케일 팩터와 허프만 코드로 압축된 샘플을 가져와야 한다.

이 디코딩 과정에서 사용된 테이블들의 크기는 표 2와 같다. 허프만 디코더의 테이블은 색인(index)과 값(value)이 한 쌍으로 구성되며, 전체 5,610 워드(표 2.1 크기1)의 크기를 가진다. 하지만 색인과 값의 각각의 값들은 1바이트를 넘지 않으므로 1워드로 허프만 디코더 1쌍을 구성할 수 있고 그 결과 테이블의 크기는 반으로 줄일 수 있다(표 2.1 크기2). DSP 내부 자원의 한계로 실시간 구현을 위해서 테이블에 필요한 메모리의 수를 줄여야 하기 때문에 이와 같은 방법으로 필요한 메모리의 개수를 줄였다.

그림5에서 나타난 바와 같이 프레임 정보 추출 단계 과정 후 신호를 재구성하는 과정에서의 계산은 모두 배정도 연산으로 처리된다.

표 2. 프레임 정보 추출에 필요한 테이블 크기
Table 2. Table size for Frame Unpacking

테이블	크기1 (워드)	크기 2 (워드)
putmask	9	9
bitrate	16	16
joint stereo table	4	4
scalefactor length	32	32
scalefactor band index	111	111
huffman decoder table	5,610	2,805
계	5,782	2,977

표 3. 신호의 재구성에 필요한 테이블 크기
Table 3. Table size for reconstruct

테이블	크기1 (워드)	크기2 (워드)
$is^{4/3}$	16,384	16,384
gain	512	8
scale	76	4
is_ratio	64	64
cs	16	16
ca	16	16
inverse mdct 가중치	148	0
window 계수	288	288
LPF 계수	1,024	1,024
subband synthesis 가중치	62	0
계	18,590	17,804

재구성 단계에 있어서 한 과정인 샘플들의 역 양자화 과정에서 적용되는 식(3)은 식(4)로 변형하여 테이블의 크기를 다음과 같이 계산해 볼 수 있다. 식(4)에서 적용되는 윈도우 형태와 크리티컬밴드 영역에 따른 이득(gain)과 스케일(scale)의 각 테이블 크기는 식(5)와 식(6)의 분류와 같이 그 크기를 2가지 경우 로 나누어 볼수 있다

$$\text{long block, gain} = 2^{0.25*(\text{global\_gain}-64)} \quad (5)$$
$$\text{short block, gain} = 2^{0.25*(\text{global\_gain}-64-8*\text{subblock\_gain})}$$

식(5)에서 이득의 크기를 계산해보면 각각 global_gain이 8비트, subblock_gain이 1비트를 차지함으로 장 블록과 단 블록에서의 배정도 계산에 필요한 최대 테이블의 크기는 각각 512워드를 가져야 한다. 이때, 이득의 테이블 값들은 프레임 정보에 따라서 선형적인 변화를 가지므로 최소 테이블 크기를 4가지 값(8워드)으로 줄일 수 있다.

$$\text{long block, scale} = 2^{-0.5*((1+\text{scalefac\_scale})*(\text{scalefac}+\text{preflag}*\text{pretab}))}$$
$$\text{short block, scale} = 2^{-0.5*((1+\text{scalefac\_scale})*\text{scalefac})} \quad (6)$$

식(6)에 각각 1비트를 차지하는 scalefac_scale, preflag와 pretab 2비트, scalefac 4비트를 대입해서 배정도 계산에 필요한 장 블록과 단 블록 최대 테이블 크기를 계산해 보면 장 블록에서 76워드, 단 블록에서 64워드가 필요하다. 이때 이득에서의 경우와 같이 스케일에서의 테이블 값들은 프레임 정보에 따라 선형적인 변화를 가지므로 최소 테이블 크기를 2가지 값(4워드)으로 줄일 수 있다. 하지만 이득과 스케일의 최소 테이블 크기는 허프만 디코더 테이블을 2가지 경우로 할당한 것과 같이, 구현될 프로세서의 속도와 메모리 크기 문제에 의존하므로 최종적으로 고려하였다.

TMS320C541 DSP는 16비트 곱셈을 최소 1사이클에 처리할 수 있는 곱셈에 대하여 최적화 된 구조로 되어 있다. 16비트 유리수 나눗셈의 경우 최소 계산 시간은 대략 30사이클을 소비한다. 따라서, 나눗셈의 계산에 있어서 그 출력의 예상 값을 미리 계산하고 곱셈으로 바꾸어 계산할 경우 현격한 연산 속도의 개선이 이루어진다. IMDCT 계수와 서브밴드 합성 계수들은 룩업테이블로 구현하였다.

프레임 정보추출과 재구성에 필요한 테이블들의 할당 절차에 의해 MP3디코더의 실시간 처리를 위해서 DSP의 메모리에 적재된 테이블의 최종 크기는 약 21K워드이며 표4에 최종적으로 결정된 메모리의 크기가 나타나 있다.

표 4. MP3 디코더에 구현된 테이블 크기
Table 4. Implemented Table size on real-time MP3 decoder

테이블	크기 (워드)	테이블	크기(워드)
putmask	9	scale	76
bitrate	16	is_ratio	64
joint stereo table	4	cs	26
scalefactor length	32	ca	16
scalefactor band index	111	inverse mdct 계수	148
huffman decoder table	2,805	window 계수	288
is	16,384	LPF 계수	1,024
gain	512	subband synthesis 계수	62
계	21,567		

MP3가 처리할수 있는 PCM오디오 샘플은 1프레임 당 모노일 경우 576샘플, 스테레오 모드일 경우 1152샘플이다. 이러한 샘플들을 실시간으로 출력하기 위해서는 샘플들의 표본화 된 시간과 샘플 수, 구현 될 프로세서의 시스템 동작시간을 정확히 계산하여야 하고, 이러한 디코딩 시간에 따라 입력되는 비트 스트림과 디코딩 후의 출력 샘플을 저장하고 있는 버퍼의 크기를 적절히 조정하는 과정이 필요하다. 따라서, MP3 디코더가 실시간으로 안정하게 동작하기 위한 보장 시간과 적절한 버퍼의 크기를 계산해 보면 다음과 같다.

출력 샘플 수 (MAX) : 1152
샘플링 주파수 (MAX) : 48kHz
비트레이트 (MAX) : 320 Kbit/s
TMS320C541 연산속도 (MAX): 40 MIPS

출력 샘플 수와 샘플링 주파수의 최대값들을 이용하여 1프레임의 출력 샘플 1152개를 실시간으로 디코딩하기 위해서는 12 ms/프레임(MAX)의 시간 내에 디코딩 작업이 이루어져야한다. 따라서, TMS320C541을 실시간 MP3디코더로 동작시키기 해서는 약 0.96 MIPS/프레임(MAX)의 시간이 필요하고, 안정된 디코더로 동작하기 위해서는 약 0.72 MIPS/프레임의 명령 처리 시간과 0.24MIPS/프레임의 마진이 요구되어진다.

표 5. TMS320C541 디버그를 이용한 MP3 디코더의 벤치 마킹
Table 5. The result of bench marking test for implemented MP3 decoder using 'C541 Debugger

알고리즘	시간 (cycle/frame)	알고리즘	시간 (cycle/frame)
find Header	1132	Stereo	86736
get Side Information	7206	Reordering	156
get Scale factors	8767	Antialiasing	57512
get Huffmande	79978	Hybrid Synthesis	197244
Requantization	50454	Subband Synthesis	323104
합계	812,289		

인코딩된 비트스트림 데이터를 저장하는 입력 데이터와 PCM 오디오 데이터의 출력 버퍼는 환형 버퍼로 구현되었다. 이 때, MP3에서 필요한 입력 버퍼는 최소 960 바이트/프레임(nSlots = 144×비트프레임/샘플링 주파수)의 크기가 필요하고, 오디오 데이터의 출력 버퍼의 경우 최소 1152 워드가 필요하다. 하지만, 이러한 최소 사양의 버퍼 크기는 버퍼의 오버플로 또는 언더플로를 유발할 수 있다. 따라서 1 프레임 당 디코딩 시간은 최대 30K 사이클/프레임 정도에 차이가 있었으므로, 오디오 데이터의 출력에 있어서 21.3ms시간 지연을 두었고 최종적으로 각 버퍼의 크기는 2K 워드로 설정하여 충분한 여유를 주었다 표 5에서는 TMS320C541 디버그를 이용하여 MP3 디코더의 실시간 동작을 검증한 벤치 마킹 테스트 결과를 나타내고 있다. 본 연구에서 구현된 디코더에서는 실시간으로 동작하기 위한 최대 시간으로부터 16%의 시간적 여유를 보이고 있다.

Ⅵ. 결 론

본 논문에서는 MPEG-1 오디오 레이어 3의 실시간 디코더를 고정 소수점 연상방식의 프로세서인 TMS320C541 DSP를 이용하여 구현하였다. MP3 알고리즘을 실시간 시스템으로 구현하기에 앞서 DSP에 대한 정수형 시뮬레이션을 통하여 고정소수점 연산방식에 대한 알고리즘의 구현 가능성을 검증하였고, 알고리즘 연산속도를 개선하기 위하여 고속 알고리즘을 적용하고 룩업테이블 등을 적용하였다. 최종적으로 실시간 시스템의 구현 단계에서는 선정된 프로세서의 처리속도를 고려하여 상위 레벨 언어로부터 어셈블 코드로의 효과적 코딩 방식을 사용하였다.

참고문헌

[1] ISO/IEC International Standard IS 11172-3 "Informaition TRechnology -Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s - part 3 : Audio".

[2] K. R. Rao, J. J. Hwang, "Techniques & Standards for Image/Video & Audio Coiding," 1996, Prentice Hall.

[3] Ted Painter, Andreas Spanias , "A Review of Algorithms for Perceptual Coding of Digital Audio Signals".

[4] Davis Pan, "A Tutorial on MPEG/Audio Compression," IEEE Multimedia Journal, Summer, 1996 issue.

[5] Byeong Gi Lee, " A New Algorithm to Compute the Discrete Cosine Transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-32 No. 6, December 1984.

[6] Charles D. Murphy and K. Anandakumar, "Real-Time MPEG-1 Audio Coding and Decoding on a DSP Chip," IEEE Transactions on Consumer Electronics, Vol. 43, No. 1, February 1997.

[7] TMS320C54x CPU and Peripherals, Texas Instruments.

[8] TMS320C54x Users Guide, Texas Instruments.

VLSI 설계 등

윤 병 우(Byung-Woo Yoon)
正會員
1987년 부산대학교 전자공학과 공학사
1989년 부산대학교 전자공학과 공학석사
1992년 부산대학교 전자공학과 공학박사
1993년~1995년 한국전자통신연구원 선임연구원
1995년~현재 경성대학교 전기전자·컴퓨터공학부 부교수
관심분야 : 배열안테나, 적응신호처리, 음성신호처리, VLSI 설계 등