2000년도 한국음향학회 학술발표대회 논문집 제 19 권 제 2(s) 호

Teaklite DSP에 기초한 MPEG-2 AAC decoder의 최적구현에 관한 연구

장봉근*, 정종훈*, 장태규*, 장흥엽**
중앙대학교 전자전기공학부*, 삼성전자 중앙연구소**

Development of a Teaklite DSP-based MEPG-2 AAC decoder

Bong-Keun Jang*, Jong-Hoon Jeong *, Tae-Gyu Chang*, Heung-Yeop Jang**
*School of Electrical and Electronics Engineering, Chung-Ang University
**Corporate R&D Center, Samsung Electronics
*tgchang@jupiter.cie.cau.ac.kr, **holo@hanmir.com

요약

본 논문에서는 MPEG-2 AAC 디코더를 고정소숫점 DSP 프로세서로 구현할 때 연산 구조에 따른 연산량 및 메모리 소요량 등의 측면에서의 최적 구현구조를 도출하고자 하였다. 이를 위하여 본 논문에서는 AAC의 주요 기능블록들인 inverse quantizer, predictor, TNS, IMDCT/Windowing등을 대상으로 연산 비트수 및 데이타 표현 구조에 따른 디코더의 성능 변화를 시뮬레이션한 후 이를 통해 얻어진 결과를 적용하여 16 비트 Teaklite DSP 프로세서 상에서 AAC 디코더를 구현하였다. 구현한 디코더는 일정수준의 음질을 유지하면서도 경제적인 메모리 소요를 보였으며 실시간으로 동작하는 것을 확인하였다.

1. 서 론

MPEG-2 AAC 오디오는 고성능의 디지털 오디오부터 인터넷 기반의 통합 멀티미디어 시스템까지의 적용을 목적으로 1997년에 MPEG에서 완성한 오디오 코덱이다. MPEG-1 오디오가 최대 128 kbps의 단일 채널 또는 스테레오 채널의 고음질을 지원하는 반면에 MPEG-2 AAC 오디오는 다섯 개의 full-bandwidth 채널에 대해 384 kbps 또는 그 이하의 데이터 속도에서 고성능 음질을 보장하는 코덱이다. 또한 MPEG-2 AAC에는 Main, Low-complexity(LC), Scaleable sampling rate(SSR)의 세 가지 프로파일이 있어서 여러 다양한 구현환경에서 최적의 음질을 확보할 수 있다.

MPEG-2 AAC 오디오 압축 알고리즘은 허프만 탐색, filter bank, prediction등을 포함한 수치적인 연산의 비중이 크기 때문에 인터넷 및 이동통신 등을 통한 멀티미디어 서비스 환경에서 고음질 저비트율 오디오 전송을 위해서는 수치적인 연산의 비중을 줄이기 위하여 DSP(digital signal processor)를 이용하여 최적 알고리즘을 구현하는 것이 효과적이다

이에 본 논문에서는 MPEG-2 AAC 오디오 코덱을 이루는 기능 블록들의 동작 원리에 대하여 살펴보고 , DSP Group의 16-bit TeakLite DSP를 이용한 오디오 디코더의 구현에 관하여 기술하였다.

2. MPEG-2 AAC에 기초한 디코더의 구성 및 동작원리

MPEG-2 AAC 오디오 디코더의 기능적인 블록도를 그림 1에서 나타내었다. MPEG-2 AAC에서는 일반적으로 양자화 과정을 통해서 오디오 데이터를 압축할 수 있다. 양자화 과정에서는 인간의 청각 특성인 심리음향 모델(phychoacoustic model)을 고려하여, scalefactor를 이용한 noise-shaping을 수행하며 이를 통하여 양자화 에러를 줄이고 충분한 SNR을 확보하게 된다. Scalefactor를 적용한 후 양자화된 각 오디오 채널 주파수의 redundancy를 줄이기 위해서 Noiseless coding 기법을 사용하며 Short-window의 경우 Grouping 및 Interleaving을 통해서 코딩 효율을 증가시킨다.

Filter bank는 시간영역의 오디오 샘플을 주파수영역으로 전환하는 과정인 MDCT(modified discrete cosine transform)와, 블록 processing를 하기 위한 window-overlap-add process의 두 가지 복합기능을 가진다. 블록사이의 불연속성을 방지하기 위해 MDCT는 각 블록을 이전 블록과 50%씩 중복하여 수행한다. 이때 입력신호의 형태에 따라서 window 모양을 다르게 적용하는데 과도상태를 가진 신호인 경우에 2048 샘플을 가진 long-window를 사용하여 transform을 적용하게 되면 양자화 잡음이 퍼지는 pre-echo 현상이 일어

난다. 이 경우 Window 길이가 짧은 256샘플의 short-window를 사용하면 이러한 현상을 줄일 수 있다.

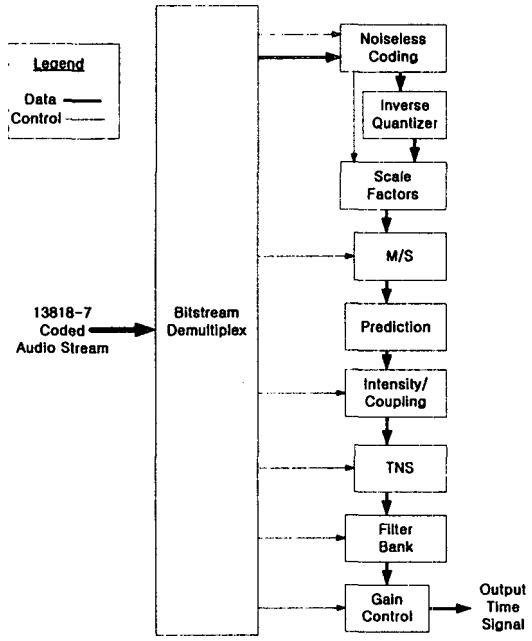

그림 1 MPEG-2 AAC decoder의 기능블록도

Prediction은 신호 중에서 stationary 특성을 보이는 부분들의 redundancy를 효율적으로 줄이는데 이용되며 long 윈도우들에 대하여 사용한다. Predictor parameter들은 LMS 알고리즘을 이용하여, 프레임 by 프레임 기반에 의한 현 신호의 통계에 따라 조절되며 최소한의 저장 메모리 공간 유지를 위해서 predictor 상태 변수들은 양자화하여 저장된다.

Gain control은 SSR profile에서만 사용되는 기능블록으로써 PQF(Polyphase Quadrature Filter) bank와 gain detector 그리고 gain modifier로 구성되어 있다. PQF bank에서는 시간 영역의 오디오 데이터를 동일한 넓이의 4개의 time-frequency band로 분할하여 각 band에 대하여 gain control을 적용한다. 그리고 encoder 에서의 샘플링 주파수에 따라서 band들을 가변적으로 선택함으로써 decoder의 complexity를 줄일 수 있다. Gain detector에서는 각 band별로 gain이 변하는 band의 수와 변경될 정보들을 검출하고, gain modifier에서는 이러한 정보를 바탕으로 time-frequency domain 오디오 데이터의 gain을 바꾼다.

3. 음성 코더의 DSP 구현

본 논절에서는 ISO/IEC 13818-7 MPEG-2 AAC Coding에 따른 음향 디코더를 TeakLite Development Kit를 이용하여 최적 구조로 구현한 결과 및 그 과정을 기술하였다.

3.1 AAC 디코더 전체 시스템 구성.

MPEG-2 AAC 디코더는 Host Data PC로부터 바이트 와이즈로 입력된 비트 스트림을 원형 입력 버퍼에 넣고 이를 demultiplexing한 후 Noiseless 디코딩하여 주파수 데이터 및 scalefactor 값들을 얻고 이를 바탕으로 역양자화 및 Joint Stereo 처리를 한다. 그리고 Inverse Prediction 및 TNS 디코딩을 한 후 Filterbank를 통과시켜 시간 축의 오디오 샘플값을 복원하게 된다. 구현한 디코더의 기본 프로그램 구조를 그림 2에 나타내었다.

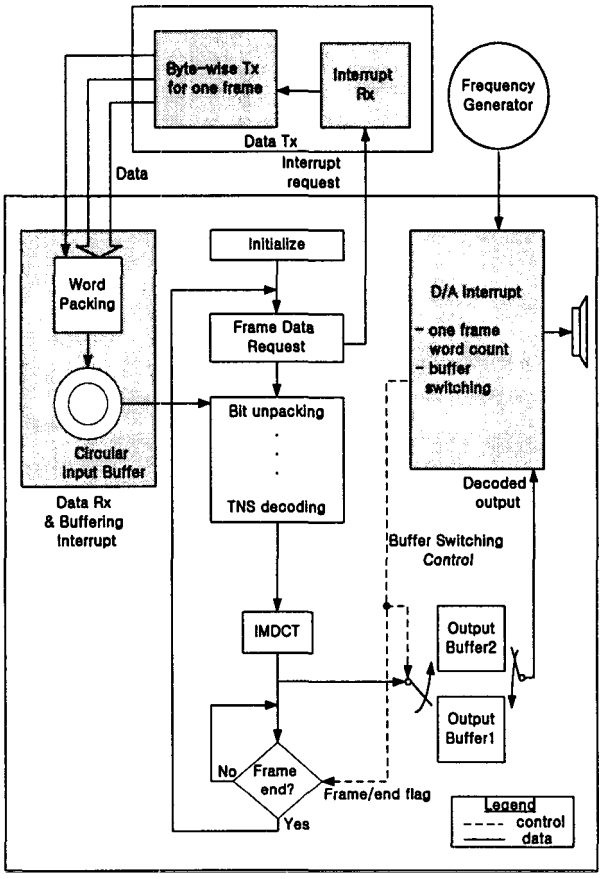

그림 2 AAC 디코더의 프로그램 구성도

3.2 Side Information decoding

입력 버퍼에 채워져 있는 비트 스트림으로부터 하나의 프레임을 처리하기 위한 정보를 얻는 부분이 프레임 데이터 디코딩의 처음 부분이다. 이를 위해 가장 먼저 싱크 비트를 탐색하여 프레임의 시작을 찾은 후 헤더 정보 및 section_data, scalefactors, pulse_data, tns_data, gain_control_data등의 side_information을 demultiplexing하여 할당된 변수에 저장한다.

Scalefactor는 허프만 디코딩의 방식으로 복원하며 Applying Scalefactor 그리고/또는 Intensity stereo에서 이 값을 사용한다. 첫 번째 scalefactor 값은 8 bits의 global_gain으로 전송되며 두 번째 scalefactor부터는 diffrentially 허프만 코딩된 값이 전송된다. Scalefactor를 디코딩 할 때에는 Standard Table A.1의 동일한

scalefactor 코드북을 사용하며 디코딩 된 scalefactor을 scalefactor band별로 각 band 내의 역양자화된 주파수 계수값들에 적용하여 스펙트럼 값들을 구한다.

3.3 Spectral Data decoding

Spectral Data를 디코딩하는 것은 Huffman table을 이용한 Noiseless 디코딩의 과정과, 역양자화의 과정으로 이루어지며 연산량 및 메모리 사용량을 최적화 하기 위하여 TeakLite의 16 bit 메모리 구조를 활용한 허프만 테이블을 구성하였다. 우선 Standard의 11개의 코드북에 대해서 각각 length 정보 및 data 정보를 나타내는 코드북과 그에 대응하는 codeword값을 나타내는 코드북를 만들고 11개의 두 종류 코드북 각각의 시작위치를 저장하는 포인터 테이블을 추가로 구성하였다. Length 정보 및 data 정보를 나타내는 코드북들은 그 구조에 따라 4-tuples와 2-tuples의 두 종류 코드북으로 구분한다.

Short_windows를 포함하는 sequences의 경우에 한해서는 위의 허프만 디코딩 후 스펙트럼 값들에 대해 부가적으로 deinterleaving을 적용한 후 역양자화를 하게 되는데 각 channel별로 디코딩 된 Huffman 값들을 *X_quant*, 역양자화된 스펙트럼 값을 *X_invquant*라 하면 *X_invquant*는 표준에 정의된 다음의 식 (1)을 통해서 얻을 수 있다

$$X\_invquant = Sign(X\_invquant) \times |X\_invquant|^{\frac{4}{3}} \quad (1)$$

실제 구현에 있어서는 (1)을 직접 계산하지 않고 최적의 연산량과 메모리 소요를 고려한 새로운 알고리즘으로 구현하였다. 즉 모든 주파수 스펙트럼 데이터 값에 대하여 128의 크기를 갖는 테이블로 이를 구현하였다. 이를 위해 Spectral 데이터의 절대값을 크게 세 범위로 나누고 위의 동일한 테이블을 적용하고 Scaling을 적용하여 주파수 데이터 값들을 찾아낸다.

3.4 Joint stereo decoding

Encoding시 듣기에 symmetric한 특성을 보이는 두 channel에 대해서는 M/S Stereo처리를 하여 encoding 시 보내는 스펙트럼 데이터의 비트 스트림 양을 줄일 수 있는데 디코딩 시 M/S Stereo부에서 이를 역으로 적용하여 left와 right 채널의 주파수 계수 값을 아래 식 (2)와 같이 복원한다.

$$\begin{aligned} l &= m - s \\ r &= m + s \end{aligned} \quad (2)$$

(m, s는 전송된 pair channel 신호, l과 r은 디코딩한 신호)

Intensity stereo decoding은, scalefactor로 전송 된 amplitude envelope의 차분값들을 이용하여 아래의 수식 (3)을 이용하여 intensity position 값을 구한다.

$$is\_position[g][sfb+1] = is\_position[g][sfb] + dpcm\_is\_position[g][sfb+1] \quad (3)$$

(g: group , sfb : scalefactor band)

그리고 위에서 구한 is_position값을 이용하여 아래의 식 (4)와(5)와 같이 right(또는 right surround) 채널의 주파수 값을 구하게 된다.

$$is\_position[g][sfb+1] = is\_position[g][sfb] + dpcm\_is\_position[g][sfb+1] \quad (4)$$

$$r_{quad}[g][b][sfb][i] = scale \times l_{quad}[g][b][sfb][i] \quad (5)$$

(g:group , b:window , sfb:scalefactor band)

3.5 TNS decoding

TNS tool에서는 양자화 잡음을 줄이기 위하여 시간영역에서의 noise shaping을 실시한다. 이 TNS tool은 크게 두개의 단계를 거치는데, 첫번째 단계는 필터에서 사용할 계수들을 생성하는 과정이고, 두번째 단계는 실제 데이타들을 필터링하는 과정이다. 먼저 필터 계수를 구하는 단계에서는 인코더에서 전달된 윈도우의 크기, 차수, 해상도 등에 의하여 인코더에서 사용한 필터의 계수값을 추출해 낸다. TNS 필터는 all-pole필터로서 프레임 윈도우의 크기에 따라서 main 프로파일의 경우에는 최대 20차, LC 프로파일의 경우 최대 12차의 필터가 적용 된다. TeakLite에서의 구현시에는 계수들을 16비트 수치로서 테이블화 하여 사용함으로써 계수를 구하는 과정에서 소요되는 연산량을 줄이도록 하였다.

3.6 Filter bank

Filter bank는 생성되어진 주파수 영역의 데이터들을 실제 우리가 들을 수 있는 시간영역의 데이터로 변환하는 과정을 실행한다. 이 filter bank는 크게 실제 변환을 실시하는 IMDCT단과, Windowing의 두 부분으로 나눌 수 있다. IMDCT단은 음의 특성에 따라 256개, 또는 2048개의 샘플을 이용하여 변환을 실시하며 이렇게 변환된 시간영역의 데이터는 각 프레임간의 연속성을 주기 위하여 overlap-add방법을 적용한다. 실제 구현에서는 연산에서 사용하는 삼각 함수 계수들을 미리 16비트 수치로서 테이블화 하여 사용하며, 이를 32비트 스펙트럼 데이터와 연산을 실시한다. Windowing단에서 윈도우의 계수는 16비트 값으로 사용하였다. 일반적인 32비트와 16비트의 곱 연산에 있어서는 32비트 데이터 출력을 가지도록 하였으나, 스피커로 출력 될 결과 값은 16비트 데이터만을 필요하므로 IMDCT변환후의 시간영역 데이터중 상위 16비트만을 이용함으로써 소요 메모리 공간의 절약과 함께 연산량의 감소를 가져오도록 하였다.

4. 동작 시험 및 결과

본 연구에서 수행한 MPEG-2 AAC 디코더의 동작을 시험하기 위하여 그림 3과 같은 실시간 시험 환경을 구현하였다.

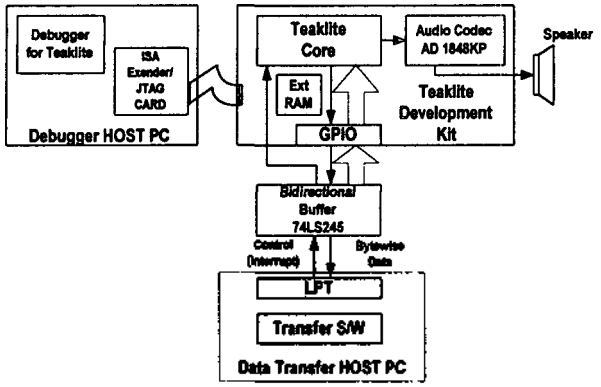

그림 3 AAC decoder의 하드웨어 구성도

디버거 호스트 PC로부터 ISA Extender/JTACK CARD를 통해서 TeakLite으로 프로그램을 다운로드 시키고 기타 디버거 명령과 프로그램 컴파일을 수행한다. 디버거 호스트 PC의 주요기능들은 TeakLite의 통합개발 툴인 TeakLite 디버거로 수행하게 된다. 데이터 전송 호스트 PC는 LPT 프린터 포트를 범용의 I/O 포트로 사용하여, AAC 오디오 비트 스트림을 TeakLite으로 전송하며 TeakLite으로부터의 제어신호를 받아들인다. TeakLite는 GPIO를 통해 오디오 데이터를 받아들이고 인터럽트 루틴을 통해 버퍼링한다. 또한 디코딩된 오디오 데이터는 오디오 코덱 AD 1848KP를 이용해 D/A converting하여 출력한다. TeakLite과 데이터 전송 호스트PC 사이에는 voltage source separation를 위해 양방향 버퍼 74LS245를 사용하여 데이터 및 제어신호를 주고 받는다.

자원 / 블록	Data memory (Kword)	Program memory (Kword)	연산량 (MIPS)
Demultiplex	10.9	2.49	1.8
Noiseless Coding	4.7	0.49	10.0
Inverse Quantizer	0.1	0.04	2.5
Scale Factors	12.0	0.2	1.7
M/S	-	0.13	0.8
Prediction	-	-	-
Intensity/coupling	0.5	0.15	1.2
TNS	-	0.44	-
Filterbank	19.7	0.82	25
Total	47.9	4.76	43.0

표 1 AAC decoder의 블록별 메모리 및 연산량 소요

TeakLite 시스템은 각기 최대 64 KWord의 프로그램 메모리와 데이터 메모리를 가지며 최대 100MIPS까지의 연산 성능을 가지고 있다. 이러한 성능 하에서 실시간 디코딩이 가능하면서 메모리 소요 및 연산 속도가 최적이 되도록 assembly 언어를 이용하여 AAC 디코더 알고리즘을 구현하였다. 그 결과 위의 표 1과 같은 부분별 연산 소요량과 메모리 소요량을 얻었다.

6. 결 론

본 연구에서는 16비트 고정 소숫점 범용 DSP인 TeakLite을 이용하여 MPEG-2 AAC 디코더를 구현하였다. 약 40MIPS의 연산량으로 실시간 MPEG-2 AAC 디코더를 구현할 수 있었고 다양한 음향 소스의 디코딩 결과에 대한 주관적인 평가에서도 만족할 만한 수준의 음질을 획득하였다.

향후 객관적인 음질 평가의 방법에 관한 연구 및 구현 알고리즘 최적화를 위한 연구를 수행할 계획이다.

7. 참고 문헌

[1] ISO/IEC JTC1/SC29/WG11 N1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding,AAC)"

[2] M.Bosi, K.Brandenburg, S.Quackenbush, Louis Fielder, K.Akagiri, H.Fuchs, M.Dietz, J.Herre, Y.Oikawa and G.Davison, " ISO/IEC MPEG-2 Advanced Audio Coding", presented at the 101st Convention of the Audio Engineering Society, J.Audio Eng.Soc. (Abstract), vol. 44, p. 1174 (1996 Dec.), preprint 4382

[3] K.Brandenburg, O.Kunz, A.Sugiyama, "MPEG-4 Natural Audio Coding", Image Communication Journal

[4 A.Gersho, "Advances in speech and audio compression," *proc. IEEE*, vol. 82, pp. 900-918, June 1994 Digital Speech:

[5] K.brandenburg and M. Bosi, :Overview of MPEG Audio: Current and Future Standards for Low-Bit-Rate Audio Coding," *Journal of Audio Engineering Society*, Vol. 45, No. 1/2, pp.4-21. Jan./Feb. 1997

[6] K. K. Parhi and T. Nishitani, *Digital Signal Processing for Multimedia Systems*, Chap. 3, Audio Compression, pp. 43-66, MARCEL DEKKER, INC., 1999.