VHDL을 이용한 MPEG-2 AAC 복호화기 모듈의 구현

우광희(禹光熙), 김수현(金洙賢), 홍민철(洪民哲), 차형태(車亨泰)
숭실대학교 정보통신전자공학부
전화 : (02) 826-9063 / 팩스 : (02) 820-9063

Implementation of the MPEG-2 AAC Decoder Module using VHDL

Kwanghee Woo, Soohyun Kim, Mincheol Hong, Hyungtai Cha
Soongsil University, School of Electronic Engineering,
E-mail : com3com4@mmslab.ssu.ac.kr

요약

본 논문은 VHDL을 이용하여 1997년 국제 표준안으로 재정된 MPEG-2 AAC 복호화기의 각 모듈을 구현하였다. AAC 복호화기는 허프만 복호화, 역양자화, 고해상도 필터뱅크등의 도구들이 필수적으로 사용된다. AAC 복호화기의 실시간 구현을 위해 각 도구들의 알고리즘을 분석하고, 하드웨어 개발에 알맞게 최적화하여 고속화와 적은 메모리를 사용하여 효율적으로 구현하였다.

I. 서론

1994년, MPEG-2 오디오 표준화위원회는 멀티 채널을 지원하기 위한 고압축율을 지원하는 새로운 압축 알고리즘 표준화 작업에 들어갔는데, 기존의 MPEG-1 오디오와의 호환을 포기하고 MPEG-2 오디오 NBC(Non-Backward Compatible) 표준을 제안하였다[1]. 이 새로운 오디오 부호화 알고리즘은 MPEG-2 Advanced Audio Coding(MPEG-2 AAC)으로 표준화되었다[1]. MPEG-2 AAC는 1998년 12월에 표준화된 MPEG-4 오디오의 T/F에 TwinVQ, BSAC와 함께 채택되었다[3].

AAC 시스템을 디지털 방송 및 멀티미디어 시스템에 적용하기 위해서는 실시간 처리 가능한 부호화 및 복호화 시스템이 개발되어야 한다.

이러한 AAC의 실시간 복호화를 위하여 본 논문에서는 각 모듈의 연산량을 분석하고 알고리즘을 칩 설계에 적합하도록 최적화하여, VHDL(Very High Speed Integrated Circuit Hardware Description Language)을 이용하여 설계하였다.

본 논문의 구성은 II장에서 AAC 복호화기의 기본 알고리즘을 살펴보고, III장에서 최적화를 수행하여 구현하였고, 마지막으로 IV장에서 결론을 맺는다.

II. AAC 복호화기의 알고리즘

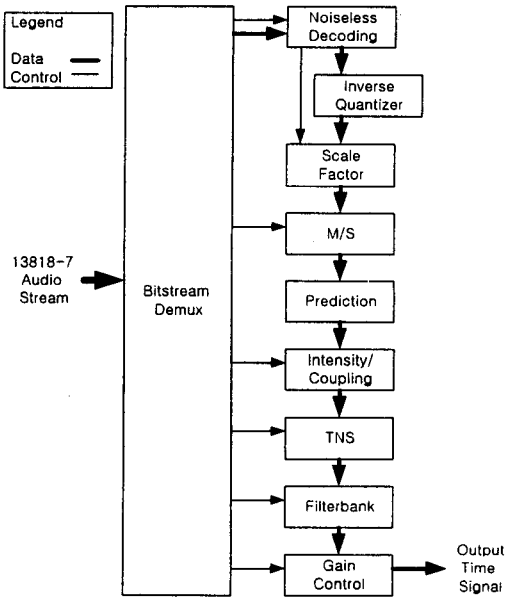

그림 1 . AAC 복호화기의 블럭도
Fig. 1 . Block Diagram of AAC Decoder

그림 1에 MPEG-2 AAC의 복호화 과정에 대한 블록도를 나타내었다[2].

복호화 과정은 AAC 부호화기에 의해 부호화된 비트열을 분석하여 스케일팩터와 스펙트럼 데이터를 허프만 복호화하여, 역양자화를 수행한다. 역 양자화된 스펙트럼 데이터를 허프만 복호화된 스케일팩터로 스케일링하고 M/S 스테레오를 적용한다. 예측기를 거쳐 인텐시티/커플링을 수행하고 시간영역잡음 변형을 수행한 후 필터뱅크를 거쳐 시간영역의 신호로 변환된다.

MPEG-2 AAC 복호화기의 필터뱅크는 주파수 영역의 신호를 시간영역의 신호로 변환하고, 윈도우와 오버랩애드(Overlap Add)를 수행한다.

III. AAC 복호화기의 최적화 및 구현

1998년 MPEG 오디오 서브그룹에서는 AAC의 각 프로필에 대한 각 툴의 연산량의 비교에 따른 소프트웨어와 하드웨어 구현을 위한 테스트를 수행하였다[4]. 표 1에서와 같이 필터뱅크와 허프만 복호화는 AAC 복호화기의 전체 연산량에서 가장 많은 비중을 차지한다.

표 1. LC 프로필의 연산량 비율[4]
Table 1. Instruction Complexity(LC profile)

	1 채널	5 채널	비율
허프만 복호화	13,657	68,285	32.5%
역양자화	1,708	8,540	4.1%
M/S		1,708	0.8%
커플링		11,546	5.5%
TNS	4,065	20,325	9.7%
필터뱅크	19,968	99,840	47.5%
합계	39,398	210,244	100.0%

1. 허프만 복호화기

MPEG-2 AAC 복호화기는 스펙트럼 데이터와 스케일팩터 데이터를 허프만 부호화 된 비트열로부터 복호화하는 과정부터 시작한다. AAC 허프만 복호화기는 11개의 스펙트럼 데이터 테이블과 1개의 스케일팩터 테이블이 있으며, 최대 289개의 심볼과 최대 19비트의 코드워드로 구성되어 있다[2].

본 논문에서는 허프만 복호화 과정의 고속화 및 적은 면적을 차지하고 메모리의 효율적 구성을 위하여 8진 트리 검색 방법으로 구현하였다.

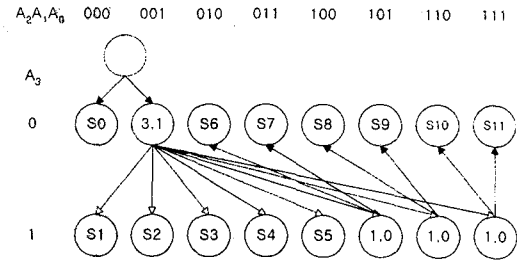

그림 2. 허프만 복호화기를 위한 8진 트리
Fig. 2. Octal-tree for Huffman decoder

그림 2에 8진 트리 검색 방법의 예를 나타내었다. 심볼이 적중하는 단말노드에서는 해당 심볼을 표시하고, 적중하지 않는 비단말 노드에서는 다음에 읽어올 비트수와 다음 노드가 위치한 메모리의 상위 주소 8비트를 표시하고 있다.

그림 3에 8진 트리검색 방식의 허프만 복호화기를 나타내었다.

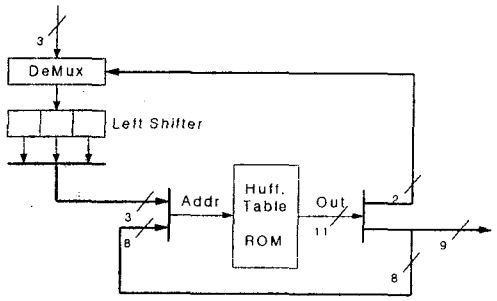

그림 3. AAC 허프만 복호화기의 블럭도
Fig. 3. Block Diagram of Huffman Decoder

파서부에서 비트열을 분석하여 각 밴드별로 11개의 허프만 테이블중 하나를 선택하여 허프만 부호화된 비트열에서 복호화한다. 11개의 허프만 테이블중 하나의 테이블을 선택하는 과정은 미리 마련된 코드북의 시작 주소를 선택함으로써 시작된다.

코드북이 선택되면 처음 한 비트를 입력받아 코드워드가 일치할 때까지 최대 3비트씩 읽어오고, 읽어올 코드워드가 2비트 이하일 경우 이전의 비트들을 왼쪽쉬프트를 이용하여 전체 3비트로 구성하여 어드레스의 하위 3비트로 사용한다. 또한, 코드워드가 일치하지 않을 경우 테이블의 값은 다음사이클에 더 읽어와야 할 비트수 2비트와 다음 사이클의 어드레스 상위 8비트를 포함하고 있다.

2. 역양자화기

허프만 복호화기에 의해 복호화된 스펙트럼 데이터는 식(1)의 연산에 의해 역양자화 된다[2].

$$Spec() = Sign(q) \cdot |q|^{4/3} \quad (1)$$

여기서, -8192 ≤q<8192의 정수이다.

일반적으로 고정소수점 연산방식의 하드웨어 구현은 계산될 지수연산 값을 ROM에 미리 저장하여 테이블로 구성한다. 그러나 식(1)에서 입력 q의 값의 범위가 상당히 크므로 실제 하드웨어 구현시 제한된 면적으로 구현하기 어렵다. 본 논문에서는 [8]의 자료를 참고하여 테이블의 크기를 최소화하였다.

$$\begin{aligned}
&|q|<127, \quad iq = q_1^{4/3} \quad \text{where, } q_1 = |q| < 127 \quad (2)\\
&|q|<1024, \quad iq = (q_2 \times 8)^{4/3} \quad \text{where, } q_2 = |q|/8\\
&\qquad = (q_2 \times 2^3)^{4/3} \qquad = |q| >> 3, \ |q_2| < 127\\
&\qquad = ([\text{int}]q_2)^{4/3} \times 2^4 + \alpha_2\\
&|q|<8192, \quad iq = (q_3 \times 64)^{4/3} \quad \text{where, } q_3 = |q|/64\\
&\qquad = (q_3 \times 2^6)^{4/3} \qquad = |q| >> 6, \ |q_3| < 127\\
&\qquad = ([\text{int}]q_3)^{4/3} \times 2^8 + \alpha_3
\end{aligned}$$

입력 q의 값을 양수로 바꾸고, 값의 크기에 따라 하위 3비트(혹은 6비트) 쉬프트 하여 128이내의 값으로 바꾸어 테이블 값을 이용한다. 여기서 잃어버린 3비트(혹은 6비트)로 선형 보간법을 이용하여 α 값을 보정해 주었다.

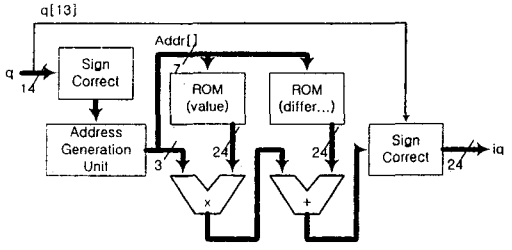

그림 4. 역양자화기의 하드웨어 블럭도
Fig. 4. Block Diagram of Inverse Quantizer

그림 4에 역양자화기의 구성을 나타내었다. 16k word (24 bits)의 테이블이 필요한 알고리즘을 선형보간법을 적용하여 128 word의 테이블 2개를 사용하여 구현하였다. 입력 값 q의 값의 범위에 따라 양수로 변환하여 128이내의 값으로 적용할 수 있도록 하였고, 계산결과를 출력하기 전에 부호를 고쳐주는 과정을 수행한다.

입력 값을 계산하여 출력하는데 3클럭이 소요되며, 다음 계산을 파이프라인으로 구성하여 전체 평균 1클럭 사이클이 소요되었다.

3. 필터뱅크

AAC의 가장 기본적인 요소인 필터뱅크는 시간 시간영역과 내부적인 시간 주파수 영역으로 상호 변환을 수행한다. 복호화기에서 사용하는 필터뱅크의 IMDCT의 수식은 식(3)과 같다[2].

$$x_{i,n} = \frac{2}{N}\sum_{k=0}^{N/2-1} X_{i,k} \cos\left(\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right), \ 0 \le n < N \quad (3)$$

여기서, N=2048 혹은 256이다.

식 (3)에서 시간영역의 한 샘플을 구하기 위해 X와 cos을 N/2번 곱하고 더하는 계산을 수행하여야 한다. 이러한 샘플을 N개 계산하는데는 $N^2/2$번의 계산이 수행되어야 한다.

이러한 IMDCT의 계산량을 줄이기 위해 식(4)와 같이 바꾸어 사용할 수 있다[5][6].

$$x'[n] = \left[\sum_{k=0}^{N/4-1}\left\{\tilde{X} e^{j\frac{2\pi}{N}\left(n+\frac{1}{8}\right)}\right\} e^{j\frac{2\pi}{N/4}nk}\right] e^{j\frac{2\pi}{N}\left(n+\frac{1}{8}\right)} \quad (4)$$

여기서, $\tilde{X} = X[N/2-2k-1] - jX[2k]$이다.

식(4)에서 $\sum_{k=0}^{N/4-1}\{\bullet\} e^{j\frac{2\pi}{N/4}nk}$ 는 N/4포인트 IFFT가 사용되었고, IFFT입력과 출력단에서 샘플의 순서를 바꾸어 복소수 곱셈을 하는 과정이 포함된다. IFFT에서 하나의 버터플라이를 수행하는데 8클럭이 소요되며, 다음 버터플라이는 5번째 클럭에서 시작하여 파이프라인으로 구성하여 평균 4클럭에 수행하였다[9].

IFFT의 출력의 주소 인덱스는 재배열 없이 후처리 과정의 입력 인덱스의 MSB와 LSB를 바꾸어 사용하였다. IFFT의 계수 값은 ROM에 미리 정의 해 두었으며 sin(·), cos(·)함수의 주기적인 특성으로 반주기만을 정의하여 사용하였다.

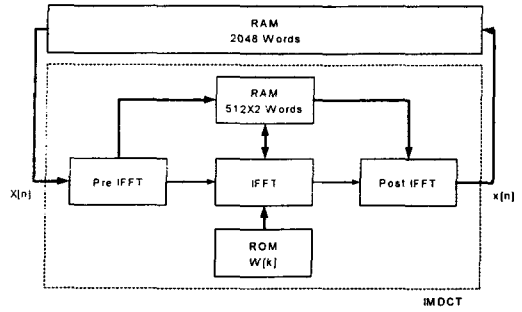

그림 5. AAC IMDCT 하드웨어 블록도
Fig. 5. H/W Block Diagram of AAC IMDCT

그림 5에 IMDCT의 블록도를 나타내었고 그림 6에 IMDCT의 출력에 윈도우와 오버랩애드를 수행하는 전체 필터뱅크의 블록도를 나타내었다.

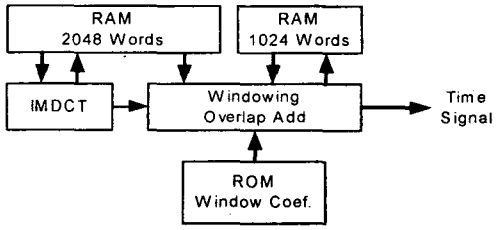

그림 6. 필터뱅크의 하드웨어 블록도
Fig. 6. H/W Block Diagram of Filterbank

IV. 결론

본 논문은 MPEG-2 AAC의 연산량의 대부분을 차지하는[4] 허프만 복호화기, 역양자화기, 필터뱅크의 모듈을 VHDL을 이용하여 구현하였다. 실제 합성 가능한 명령만을 사용하여 synopsys 상에서 검증하였다. 표 2는 하드웨어로 구현된 각 도구들의 클럭 점유율을 나타내었다.

표 2. AAC 도구들의 수행 속도
Table 2. Processing speed of AAC tools

	허프만 복호화	역 양자화	필터 뱅크	계
Clk	약 4,000	1,027	15,380	약 20,407
비율	20%	5%	75%	100%

허프만 복호화기는 400게이트 면적으로 설계하였으며 2k word(11bit)의 ROM으로 12개의 허프만 테이블을 구성하였다. 최대 19비트로 구성된 코드워드를 복호화하기 위해 8클럭 사이클이 소요된다. 역양자화기는 16k word의 테이블을 사용하지 않고 2개의 128 word의 테이블과 선형보간법을 이용하여 1클럭 사이클에 연산할 수 있도록 구현하였다. 필터뱅크는 연산량을 줄이기 위하여 전처리 후처리 과정이 포함된 IFFT를 사용하여 IMDCT를 구현하고 데이터의 재배열과정 없이 윈도우와 오버랩애드를 수행하여 연산속도를 향상시켰다.

참고문헌

[1] M. Bosi, "Overview of MPEG Audio : Current and Future Standards for Low-Bit-Rate Audio Coding," J. AES, Vol. 45, No. 1/2, pp. 4-21, Jan/Feb. 1997

[2] ISO/IEC 13818-7, "Generic Coding of Moving Pictures and Associated Audio Information - Part 7 : Advanced Audio Coding," 1997

[3] ISO/IEC 14496-3, "Information Technology - Coding of Audiovisual Objects - Part 3 : Audio, Subpart 4 : T/F Coding," 1998

[4] ISO/IEC JTC1/SC29/WG11 N2005, "Revised Report on Complexity of MPEG-2 AAC Tools," Feb. 1998

[5] Rolf Gluth, "Regular FFT-Related Transform Kernels for DCT/DST-Based Polyphase Filter banks," ICASSP, vol.3, pp. 2205-8, 1991

[6] ATSC, "Digital Audio Compression Standard (AC-3)," Dec. 1995

[7] Vikram Iyengar, Krishnendu Chakrabarty, "An efficient finite-state machine implementation of Huffman decoders," Information Processing Letters, V.64 N.6, 271-275, Dec. 1997

[8] 김진원, 정남훈, 김준석, 이근섭, 이충용, "MPEG-1 계층 III 오디오 복호화기의 VLSI 설계," 신호처리합동학술대회 논문집 제 12권 1호, pp 847-50, 1999

[9] 우광희, 차형태, "VHDL을 이용한 MPEG-2 AAC 복호화기 필터뱅크의 구현," 대한전자공학회 하계종합학술대회 논문집, 제 23권 제 1호, pp 178-181, Jun, 2000