논문 2005-42SD-4-8

70MIPS 이내에서 동작하는 MPEG-2 AAC 부호화 칩 설계

(An MPEG-2 AAC Encoder Chip Design Operating under 70MIPS)

강 희 철*, 박 주 성*, 정 갑 주*, 박 종 인**, 최 병 갑***, 김 태 훈****, 김 승 우*****

(Hee-Chul Kang, Ju-Sung Park, Kab-Ju Jung, Jong-In Park, Byung-Gab Choi, Tae-Hoon Kim, and Sung-Woo Kim)

요 약

MPEG-2 오디오 압축방식인 AAC(Advanced Audio Coding) LC(Low Complexity) 스테레오 부호화기를 고속으로 구현할 수 있는 칩을 32비트 DSP 코어를 기반으로 설계하고 0.25um CMOS 기술을 이용하여 제작하였다. 계산량과 메모리 용량을 줄이기 위하여 알고리즘 구현방법 측면에서 최적화를 하였으며, FFT(Fast Fourier Transform)를 하드웨어로 구현하여 고속화 하였다. 제작된 칩의 크기는 7.20x 7.20 mm^2 이었으며 등가 게이트는 약 830,000 이었으며 70MIPS 이내에서 AAC 부호화를 할 수 있음을 확인하였다.

Abstract

A chip, which can fast encoder the audio data to AAC (Advanced Audio Coding) LC(Low Complexity) that is MPEG-2 audio standard, has been designed on the basis of a 32 bits DSP core and fabricated with 0.25um CMOS technology. At first, the various optimization methods for implementing the algorithm are devised to reduce the memory size and calculation cycles. FFT(Fast Fourier Transform) hardware block is added to the DSP core to get the more reduction of the calculation cycles. The chips has the size of 7.20x 7.20 mm^2 and about 830,000 equivalent gates, can carry out AAC encoding under 70MIPS(Million Instructions per Second).

Keywords: DSP, AAC Encoder, SoC, ASIC

I. 서 론

MP3 player와 같은 휴대디지털 오디오기기의 성능은 DSP (Digital Signal Processor)와 그 DSP를 이용하여 MP3 (MPEG-1 Layer 3), WMA(Window Media Audio), AAC(MPEG-2 Advanced Audio Coding) 등과 같은 알고리즘을 얼마나 효율적 부호화하고 복호화 하느냐에 달려있다. 하드와이어드 개념이 강조된 칩 설계는 특정한 알고리즘 구현에는 효율적일지는 모르지만 다양한 알고리즘을 구현하는 데는 적합하지 않다. 다양한 알고리즘을 구현하기 위해서는 성능이 좋은 DSP를 플래트폼으로 하여 어셈블리 언어수준의 알고리즘 최적화가 필요할 뿐만 아니라 경우에 따라 H/W S/W co-design 개념을 도입할 필요가 있다.

일반적으로 오디오 알고리즘을 구현함에 있어 부호화 알고리즘이 복호화 알고리즘보다 계산양이 많아 실시간 구현에 어려움을 겪게 된다. 특히 부호화 알고리즘 중에서도 AAC 부호화 알고리즘이 MP3 알고리즘보다 30% 이상의 계산양을 요구하기 때문에 실시간 구현에 어려움이 있다. Intel Pentium III로 만든 PC에서 AAC LC(Low Complexity) 부호화기를 48.6MIPS에서 구현하였다는 보고가 있다.[1] 이 연구는 PC상에서 알고리즘을 구현하였기 때문에 휴대오디오 기기에 직접 적용하는 데는 문제가 있다. AAC 엔코더를 구현한 연구결과를 정량적으로 발표한 문헌은 발견하기 힘들고, 텍사스 인스트루먼트사의 DSP를 사용하여 구현하였다는

* 정회원, 부산대학교 전자공학과
(Dept. of Electronic Engineering, Pusan National University)
** 정회원, (주)금영 부설연구소
(Kumyoung Technology Research Institute)
*** 정회원, 인천대학교 디지털정보전자공학과
(Dept. of Digital Information Electronics, Incheon City College)
**** 정회원, (주)보이소반도체
(Voiso Semiconductor Co.)
***** 정회원, 국방품질관리소
(Defense Quality Assurance Agency)
※ 본 연구는 IDEC, KOTEF, KIPA의 IT-SoC 사업단의 지원을 받음
접수일자: 2005년1월5일, 수정완료일: 2005년4월7일

사례는 많이 소개되고 있으나 기업의 영업적인 측면을 고려하여 정량적인 연구결과는 제시하지 않고 있다.[2][3] 연구결과 [3]을 통하여 데이터 프로세싱에 필요한 RAM을 DSP 외부에 두었을 경우 AAC 부호화기를 구현하는데 100MIPS 대의 계산량이 필요하다는 것을 유추할 수 있다.

본 논문에서 설계하는 칩은 DSP 코어와 ROM과 RAM을 모두 내장하고 있다. 주요 연구내용은 32비트 DSP 코어를 기반으로 하여 알고리즘 최적화, 어셈블리 언어 코딩, H/W S/W co-design을 통한 MDCT (Modify Discrete Cosign Transform)의 계산양 감소방법, 칩 제작 등 이다. 제Ⅱ장에서는 AAC 부호화 알고리즘의 전체적인 흐름을 다루고, 제Ⅲ장에서는 DSP를 이용하여 AAC 부호화기를 구현하기위한 고안한 다양한 최적화 방법을 소개 한다. 제Ⅳ장에서는 MDCT 블록을 하드웨어로 구현하는 과정을 다루고 제Ⅴ장에서는 설계된 칩의 검증과정, 칩 설계, 칩 테스트 방법과 테스트 결과에 대한 관한 내용을 다루고, 마지막으로 결론을 내린다.

Ⅱ. AAC 알고리즘

AAC는 main, low complexity(LC), scaleable sampling rate(SSR)와 같은 3종의 프로파일을 가진다.[4][5][6] 프로파일에 따라 다양한 옵션을 포함하고 있으며 휴대오디오 용으로는 LC가 가장 적당하다. 본 논문은 휴대 오디오를 목표로하는 LC 프로파일을 위한 부호화 칩 설계에 관한 내용을 다룬다. AAC Encoder 알고리즘의 전체적인 흐름을 그림 1. 에 나타내었다. 주요 단계에서 수행되는 내용을 살펴보면 다음과 같다.

1. 심리음향 모델

MP3와 마찬가지로 인간의 청각기능의 모델링 한 심리음향 모델을 사용하고 있다.[7][8] 이 블록의 출력은 인접한 주파수 성분에 의하여 마스킹되어 우리 귀에 들리지 않는 주파수성분을 찾아내어 masking threshold 값을 결정한다. 이 값은 나중에 iteration loop에서 비트 할당기준으로 사용된다. Masking threshold 값을 이용하여 심리음향 모델에서 우리 귀에 들리지 않은 주파수 성분을 제거할 수 있으므로 음질의 손상을 주지 않고 압축율을 높일 수 있는 기반을 제공한다.

2. MDCT

시간영역의 오디오 데이터를 주파수영역으로 변환함

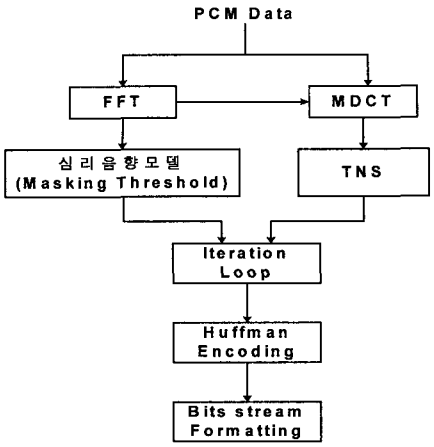

그림 1. AAC 부호화 과정도
Fig. 1. Flow diagram df AAC Encoder.

에 있어 압축율을 높이고 주파수 resolution을 높이기 위하여 AAC에서는 MDCT (Modified Discrete Cosine Transform) 방식을 채용한다. MDCT를 구현하기 위해서는 기본적으로 FFT(Fast Fourier Transform)를 수행하여야 한다. 이 단계에서는 시간영역에서 앨리아싱을 제거하기 위하여 TDAC (time domain aliasing cancellation) 기법을 사용한다.

3. TNS

TNS(Temporal Noise Shaping)는 AAC에서 새로이 도입된 개념으로, MP3나 MPEG-2 BC (Backward Compatibility) 방식이 과도구간의 부호화에서 겪고 있는 문제점을 해결하기 위한 방법이다. 이러한 문제는 마스킹 임계값과 양자화 잡음 사이의 시간적 불일치로 인한 프리 에코(pre-echo) 현상 때문에 지각적 부호화가 어렵기 때문에 발생하는 것이다. TNS 기술은 이러한 문제를 해결하기 위해 필터뱅크 윈도우 내에 있는 양자화 잡음의 시간적 미세 구조를 제어하고 있다.

4. Iteration loop

이 루프의 기본은 analysis by synthesis 형태로 진행되기 때문에 많은 연산이 필요한 부분이다. 심리음향 모델에서 계산된 masking threshold 값과 MDCT 과정을 거쳐 변환된 주파수영역의 데이터를 AAC 포맷에서 허용하는 정해진 비트율을 만족시키기 위하여 시행착오적인 방식으로 각 프레임의 데이터 비트를 정한다.

5. Huffman encoder 및 Formatting block

심리음향 모델을 이용하여 1차적으로 압축된 디지털 데이터를 허프만 엔코딩 기법을 이용하여 압축율을 더

욱 높인다. 최종적으로 압축된 데이터와 복원에 필요한 여러 정보를 조합하여 AAC 포맷에 맞게 데이터를 조합하는 최종 단계이다.

III. DSP를 이용한 알고리즘 구현

본장에서는 제II장에서 소개한 AAC 부호화 알고리즘을 Texas Instrument사의 32비트 floating point DSP(TMS320C30)를 이용하여 구현함에 있어 계산 사이클과 메모리 용량의 줄이기 위하여 고안한 방법들을 소개한다.

1. 함수 처리

Iteration 루프에서 양자화 과정을 수행하는 과정에서 식 (1)과 같은 함수를 계산을 해야 한다.

$$pow\_quant[i] = \mid i \mid^{\frac{3}{4}} \tag{1}$$

정수의 3/4승을 계산하기 위해서는 많은 계산사이클을 필요로 하기 때문에 계산량 감소를 위하여 look-up 테이블을 만들어 사용하였다. 엔코딩 과정에서 (식 1)에서 i 의 범위는 $0 \leq \mid i \mid \leq 8191$와 같다. 다양한 종류의 오디오 데이터에 대하여 실험해본 결과 i의 값이 128이내가 되는 경우가 99%이상 된다는 것을 확인하였다. 따라서 look-up 테이블을 저장하는 ROM 용량을 줄이기 위하여 $0 \leq \mid i \mid \leq 128$ 구간만 테이블화 하였다. 나머지 구간은 인터폴레이션 기법을 사용하여 값을 구하는 방식을 채택하였다. 8191 경우 중에서 1% 정도만 인터폴레이션을 사용하기 때문에 이에 따르는 계산양 증가는 많지 않으며 최대 오차가 0.001% 이하가 되게 하였다. 그리고 심리음향 모델링과정에서 자주 사용되는 로그연산과 지수연산, 삼각함수, 제곱근연산 등을 테일러급수와 look-up 테이블을 적절히 혼용하여 처리 하였다.

2. MDCT 계산양 축소

MDCT는 (식 2)와 같은 계산을 통하여 수행된다.[9]

$$X_{i,k} = 2 \cdot \sum_{n=0}^{N-1} x_{i,n} \cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right)$$
$$for\ 0 \leq k < \frac{N}{2} \tag{2}$$

(식 2)에서 하나의 주파수에 대한 결과값을 구하기 위해서는 $x_{i,n}$와 cos항을 곱하고 적어도 N/2번만큼 더해야 한다. 그러나 (식2)를 (식3)과 같이 odd-time, odd-frequency Discrete Fourier Transform(O^2DFT)을 이용하면 계산양을 획기적으로 줄일 수 있다.[10] AAC 부호화의 경우는 고속 MDCT를 이용하는 경우 계산 사이클을 1/36 정도로 줄일 수 있다.

$$Y(k) = \left[\sum_{n=0}^{N/4-1}\left\{\left(x(2n) - jx\left(\frac{N}{2}+2n\right)\right) e^{-j\frac{2\pi}{N}\left(n+\frac{1}{8}\right)}\right\} e^{-j\frac{2\pi}{N/4}kn}\right] \times e^{-j\frac{2\pi}{N}\left(k+\frac{1}{8}\right)} \tag{3}$$

3. Iteration 과정의 최적화

Iteration loop는 AAC 부호화기에서 가장 많은 계산 사이클을 차지하는 부분으로 inner iteration loop와 outer iteration loop로 구성되어 있다. 두 iteration 과정을 통하여 최적의 양자화 스텝사이즈를 찾는 곳이다. 이 과정에서는 계산 사이클 수를 줄이기 위하여 가능한 한 반복계산 횟수를 줄이는 것이 가장 중요하다.

가. Scale-factor 초기값 설정

Iteration 과정에서 inner loop를 처음 수행하면 최소한 3번의 common scale-factor 값이 변화하게 되는데, 이 값에 따라 루프 반복횟수가 결정되므로 초기값을 적절하게 설정해주는 것이 중요하다. 다양한 common scale-factor를 여러 장르의 오디오 데이터의 iteration loop 과정을 살펴본 결과 표 1.에 나타난 바와 같이 scale-factor를 64로 했을 때 루프를 수행하는 횟수가 가장 작아지는 것을 알았다. 계속적인 부호화 과정에서도 scale-factor를 직전 프레임의 마스킹에 사용된 값을 초기값으로 하여 반복계산을 수행하면 반복계산 횟수와 노이즈를 줄일 수 있었다.

표 1. Common Scale Factor의 초기값에 따른 iteration loop의 계산양 및 노이즈
Table 1. Calculation cycles and noise of iteration loop.

Common Scale factor 값	Inner loop 반복횟수	Outer loop 반복횟수	noise
주어진 식	4,919	725	1.01
10	10,578	725	1.01
40	4,698	725	1.01
64	2,316	739	1
80	588	196	1.44
이전프레임	1,981	695	1.04

표 1.은 몇 종류의 음악파일에 대하여 1,000프레임이상을 부호화하는 과정에서 얻은 데이터를 평균하여 정수 값을 취한 것이며, 노이즈는 가장 작은 경우의 값으로 표준화한 값이다. Common scale-factor가 64 보다 커지면 반복계산 횟수를 줄일 수 있지만 노이즈가 커지는 문제가 있어 음질의 손상을 가져오기 때문에 64를 초기값으로 선택하였다.

나. Noiseless coding 구현방법

Noiseless coding block에서는 맨처음 계산중인 spectral section에서 어떤 허프만 코드북을 이용할 것인가를 결정하기 위하여 그 section 내의 계수들 중에서 가장 큰 절대값을 찾는다. 이 절대 값을 이용하여 엔코딩하는 spectral section의 다이나믹 영역을 계산한다. 다이나믹 영역을 바탕으로 엔코딩에 사용될 허프만북을 결정한다. AAC 부호화기에 사용되는 허프만북의 특징 중에 하나는 인근한 두개의 허프만 북은 동일한 최대 절대값을 가진다.

임의의 spectral section에서 최대절대 값이 같은 코드북인 경우에는 동일한 인덱스를 사용하여 코드길이와 코드워드를 구한다. 따라서 n 번째 코드 북을 가지고 엔코딩 했을 때와 n+1번째 코드북을 가지고 했을 때 동일한 인덱스를 사용하므로 이 인덱스를 중복해서 계산하면 계산양에서 많은 손해를 보게 된다. 인덱스를 한번만 계산해서 두개의 코드북에 적용하면 noiseless coding의 계산양을 데이터에 따라 30-40% 정도 줄일 수 있다.

지금까지 소개한 방법들을 이용하여 TMS320C30 DSP에 AAC LC 스테레오 부호화기를 구현해본 결과를 표 2.에 정리한 바와 같이 73.1MIPS가 필요하다. 여기서 한 프레임은 1024 샘플이고 샘플링 주파수는 44.1 KHz이다. 그리고 C 언어를 이용하여 floating point로 구현한 부호화 결과와 DSP를 이용하여 구현한 결과의 차이를 노이즈로 정의하였다. 30초 분량의 오디오 데이터를 이용하여 실험한 결과 프레임당 노이즈의 r.m.s S/N 비가 80dB 이상 이었으며, 엔코딩한 신호의 full scale 값을 1로 normalize 하였을 때 최대 노이즈가 $3.1x10^{-6}$ 이었다. 또한 부호화된 데이터를 PC상의 상용 프로그램(Winamp)의 AAC 복호화 프로그램을 통한 청음평가에서도 차이가 없음을 확인 하였다. 부호화 알고리즘을 수행하기 위하여 필요한 RAM은 90KB, ROM은 106 KB(프로그램 ROM 34 KB, 테이블 ROM 72 KB)가 필요하였다.

표 2.에서 보는 바와 같이 AAC 부호화기에서 iteration loop가 차지하는 비중이 가장 높은 것을 알 수 있다. 고속 엔코딩을 위해서는 iteration loop를 하드웨어로 구현하는 것이 계산양을 줄일 수 있지만, iteration loop의 경우는 규칙성이 없는 랜덤한 논리회로를 이용하여 구현해야 하기 때문에 하드웨어로 구현하는 데는 많은 어려움이 있다. 본 논문에서는 계산양이 많으면서도 규칙적인 FFT를 반복적으로 사용하여 MDCT를 수행하는 필터 뱅크 블록을 하드웨어와 소프트웨어로 구현하여 계산양 감소를 도모하였다.

표 2. TMS320C30을 이용한 AAC 부호화기 구현결과
Table 2. Calculation cycles on TMS320C30 DSP.

	Cycles per Frame	%	MIPS
심리음향	304,184	17.9	13.1
필터뱅크	273,998	16.2	11.8
TNS	134,677	7.9	5.8
Iteration Loop	873,078	51.4	37.6
기타	111,456	6.6	4.8
Total	1,697,393 (73.1MIPS)	100	73.1

Ⅳ. MDCT 알고리즘의 구현

MDCT를 포함하는 필터뱅크의 각 과정에서 소요되는 계산양은 표 3.과 같으며 FFT가 차지하는 비중이 70% 가까이 되는 것을 알 수 있다. 필터뱅크를 구현함

표 3. 필터뱅크의 계산량분석
Table 3. Calculation cycles of filter bank.

과정	Cycles per Frame	비율(%)
전처리 과정	19,727	7.2
FFT	190,977	69.7
후 처리	19,727	7.2
인터리빙	15,620	5.7
윈도우 중첩	27,947	10.2
합계	273,998	100

표 4. FFT의 기본연산 블록 성능비교
Table 4. Comparison of FFT blocks versus Radix.

알고리즘	게이트 수	cycles per frame for 512 points
Radix-2	5,980	20,736
Radix-4	18,492	8,448

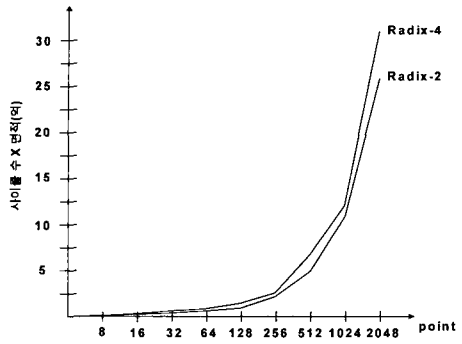

그림 2. Sample point에 따른 Radix-2와 Radix-4 FFT의 성능비교
Fig. 2. Performance comparison of Radix-2, 4.

에 있어 곱셈기를 포함한 FFT의 기본연산블록은 하드웨어로 구현하고 나머지 루틴은 소프트웨어적으로 처리한다.

FFT연산을 위해서는 네 번의 실수 곱셈과 두 번의 실수 가감산을 수행해야한다. 칩의 면적을 줄이기 위해서는 곱셈기의 개수를 줄이는 것이 가장 좋은 방법이다. 사용하는 DSP의 데이터 비트가 32 비트이기 때문에 32 x 32 비트 곱셈기를 설계해야 하지만 곱셈연산 결과 64비트 중 상위 32 비트만 사용되기 때문에 17 x 17 곱셈기를 사용하고 결과 값 34비트 중에서 32비트만을 사용하는 형태로 곱셈기의 구조를 결정하였다.

가장 경제적인 FFT의 기본연산블록 설계를 위하여 Radix-2 기반의 FFT와 Radix-4 기반의 FFT 기본연산블록을 설계하여 게이트 복잡도와 사이클 수를 비교하였다. AAC 부호화기에 사용되는 512 포인터 FFT를 구현할 때 두 경우에 대한 결과를 표 4.에 정리하였다. Radix-4 방식은 Radix-2 경우보다 한번에 처리할 수 있는 데이터가 2배 많아 처리 속도가 빠르지만, 소요면적(게이트 수)에서는 불리하다. 동작속도와 면적을 동시에 비교하기 위하여 FFT 샘플 포인터에 따른 두 경우의 (사이클 수)X(게이트 수)의 결과를 그림 2.에 나타냈다. AAC 부호화기에서 사용하는 64와 512 포인터 및 다른 샘플 포인터에서 Radix-2 알고리즘이 Radix-4 알고리즘보다 약 15% 성능이 좋게 나왔다. 이를 바탕으로 본 연구에서는 Radix-2 기반의 FFT를 설계하기로 하였다.

기본연산블록은 그림 3.과 같이 DSP 코어와 독립적으로 FFT를 수행하기 위하여 FSM (Finite State Machine) 블록, FSM 블록에서 생성된 State를 바탕으로 제어신호를 생성하는 컨트롤 블록, ROM과 RAM으로부터 FFT 연산에 필요한 데이터를 가져오기 위한 어드레스 생성블록, 하나의 승산기와 버터플라이 연산기

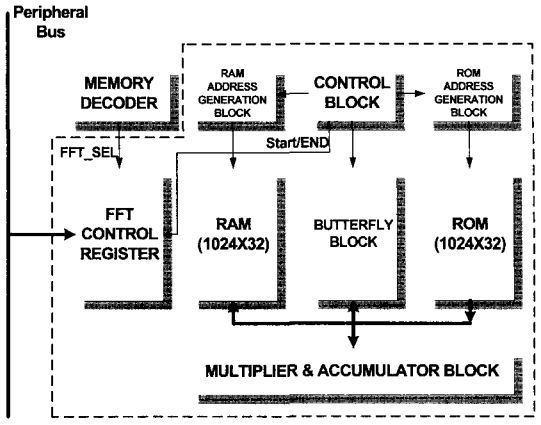

그림 3. FFT 기본블록의 구조
Fig. 3. Structure of FFT block.

로 구성되어있다. RAM 블록은 입력 값, FFT 결과 값 및 연산 처리 중의 값들이 저장되는 곳으로 최고 32-bit의 512개의 데이터를 저장 할 수 있어야 하므로, 데이터의 실수부와 허수부로 나누어 총 1024 × 32-bit의 RAM을 필요로 한다. ROM 블록에는 512 × 32-bit의 FFT 트위들 팩터를 저장한다. FFT 블록의 동작은 다음과 같다. DSP 코어에서 FFT를 수행할 데이터를 FFT블록의 RAM에 미리 저장해둔 다음 FFT를 시작하게 한다. FFT가 수행되고 RAM에 결과를 저장한 후에 DSP 코어에 수행완료를 알린다.

그림 3.과 같은 구조를 가지는 기본블록을 이용하여 설계된 기본연산블록을 이용하여 FFT를 수행하면 20,736사이클(0.89MIPS)이 소요되어 표 3.의 190,977사이클(8.82MIPS) 보다 줄어드는 것을 알 수 있다. FFT를 수행하는 동안에 DSP 코어는 심리음향 모델링을 포함한 다른 연산을 수행하도록 프로그램을 구성하여 FFT 수행시간이 전체 계산 사이클에서 포함되지 않게 하였다. 따라서 FFT 블록을 설계하여 추가함으로써 8.82MIPS가 줄어들어 AAC 부호화기를 64.3MIPS에 구현할 수 있다.

V. 칩 설계 및 제작

1. DSP 코어 설계

칩 설계는 TMS320C30과 호환되는 DSP 코어를 기반으로 AAC 부호화기 칩을 설계하였다. DSP 코어의 주요 특징은 다음과 같다.[11]

. CPU: 40bit floating-point,
32bit integer multiply operation
. 파이프라인: 4단
. 명령어 수: 115종

. 버스: 10 개

. 어드레싱 모드: 8종

DSP 코어는 명령어별 검증, 다양한 응용알고리즘을 이용한 검증, 하드웨어 에뮬레이션을 통한 응용프로그램 수행 등을 통하여 호환성을 충분하게 검증하였다. [12][13]

2. DSP 코어와 FFT 블록 인터페이스

FFT 블록과 DSP 코어의 인터페이스를 위하여 그림 4.와 같은 구조를 가진 인터페이스 레지스터를 두었다. 인터페이스 레지스터에는 FFT 시작을 위한 start 비트, 64/512 포인터 FFT모드를 지정하기 위한 Long/Short 비트, FFT 완료를 알리는 End 비트로 구성되어 있다.

DSP 코어는 FFT를 시작하기 전에 FFT 블록의 RAM에 샘플데이터를 저장시킨 후 그림 5.에서 보여주는 바와 같이 peripheral 버스를 이용하여 인터페이스 레지스터에 FFT 블록제어에 필요한 정보를 쓴다. 이 정보에 따라 메모리 디코더가 FFT 블록제어에 필요한 신호를 만들어낸다. FFT 블록은 인터페이스 레지스터의 short/long 비트에 따라 64 포인터/512 포인터 FFT를 수행한 후 RAM에 결과를 저장한 후 인터페이스 레지스터 End 비트를 "1"로 세트한다.

FFT 수행에 필요한 사이클을 미리 알고 있기 때문에 적절한 시기에 End 비트를 조사하여 "1"인 경우에는 FFT 블록의 RAM에 저장된 데이터를 가져간다.

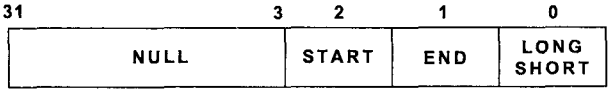

그림 4. 인터페이스 레지스터 구조

Fig. 4. Structure of the interface register.

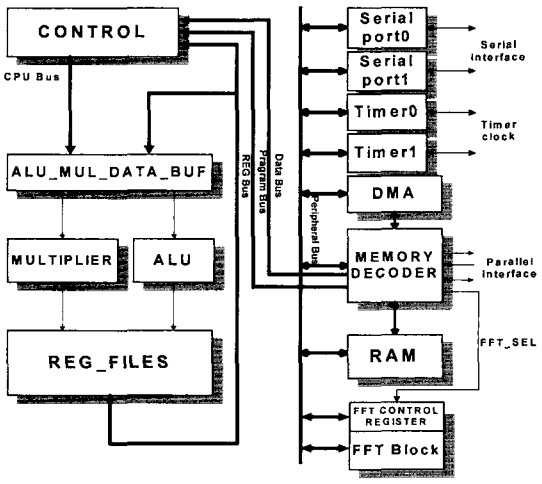

그림 5. FFT 기본연산 블록과 코어와 인터페이스

Fig. 5. Interface of DSP and FFT block.

3. 설계 검증 및 Back- End 설계

Verilog 언어를 이용하여 전체 칩을 설계하여 시뮬레이션을 통하여 대략적인 칩 동작을 확인하였다. HDL 논리시뮬레이션을 통하여 AAC 부호화기 동작전체를 완벽하게 검증하는 것은 현실적으로 불가능하다. 따라서 본 연구에서는 그림 6.에서 보여주는 바와 같이 하드웨어 에뮬레이션을 통하여 30초 분량의 PCM 데이터를 AAC 부화기로 압축하고 그 데이터를 C 언어를 이용하여 압축한 데이터와 비교해본 결과 제2장의 결과와 동일하였다. 압축한 데이터를 PC상의 상용 디코더 프로그램(Winamp)을 통하여 재생하여 주관적인 평가를 거쳐 모든 것이 정상적으로 동작한다는 것을 검증하였다.

하드웨어 에뮬레이션을 통하여 검증된 설계를 0.25um CMOS 표준 셀 라이브러리를 이용하여 블록단위로 합성하여 전체 칩을 레이아웃 하였다. 설계하는 칩은 내부에는 91KB RAM과 106.5KB ROM을 포함하고 있기 때문에 메모리가 차지하는 면적이 전체 칩의 85% 정도 되므로 메모리를 적절하게 배치하는데 많은 노력을 기울였다. 칩의 전체적인 배치는 그림 7.과 같으며 면적은 7.20 x 7.20mm^2 이다. 칩의 등가 게이트 수는 830,000 게이트이고 팩키지는 144핀 LQFP로 했다.

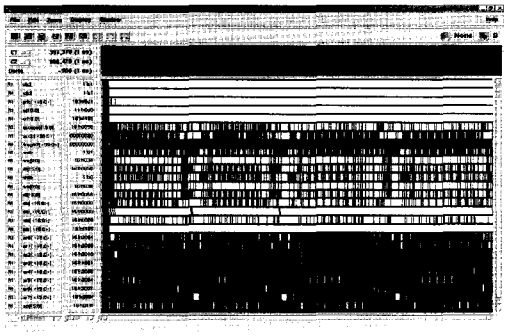

그림 6. Hardware Emulator를 이용한 AAC 엔코딩 결과화면

Fig. 6. Emulation result of AAC encoder.

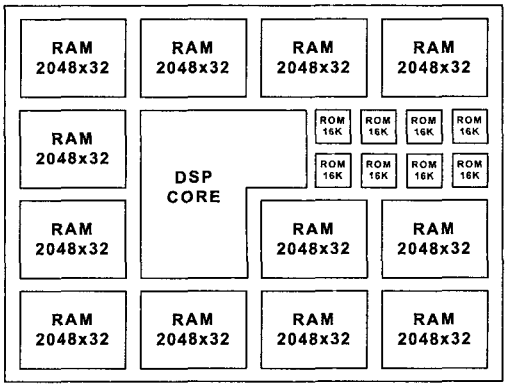

그림 7. 칩 레이아웃 구조도

Fig. 7. Chip layout.

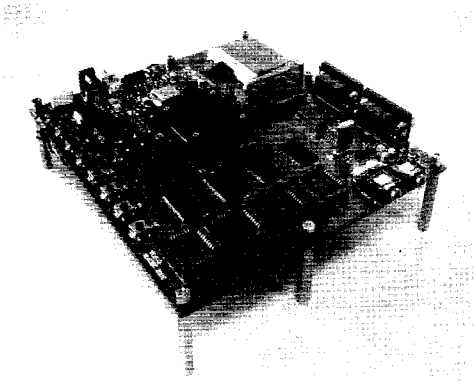

그림 8. AAC 부호화 칩 테스트 용 보드
Fig. 8. Test board of AAC encoder chip.

레이아웃 후의 시뮬레이션에서 설계된 칩이 75MHz 클럭 주파수에서 동작하므로 실시간으로 AAC 부호화기 구현이 가능하다는 것을 1차적으로 확인하였다.

4. 칩 테스트

제작된 칩을 테스트하기 위해서 그림 8.과 같은 테스트용 보드를 제작하였다. 보드의 구성은 보드제어에 필요한 프로그램용을 저장할 수 있는 EEPROM, 데이터를 저장 할 수 있는 SRAM를 준비했다. 여러 칩을 테스트 할 수 있게 하기 위해서 144 LQFP용 소켓을 PCB 보드에 장착했다. 그리고 PC와 편리하게 데이터를 주고받기 위한 UART, 데이터 저장을 위한 플래시 메모리, 오디오 데이터의 실시간 변환을 위한 오디오 코덱, DSP의 동작상태를 표시하는 LCD, 칩제어에 필요한 입력 키 등을 포함하고 있다.

칩 테스트 첫 단계로 DSP 코어의 동작상태를 테스트하는 프로그램을 수행시켜 DSP의 정상동작 여부를 확인 하였다. 다음 단계로 칩이 70MIPS에 동작하도록 클럭을 세팅하고 코덱을 통하여 44.1KHz로 샘플링되어 실시간으로 변환되는 PCM 음악데이터를 AAC 방식으로 부호화하여 SRAM에 저장한다. 저장된 데이터를 USB를 통하여 PC로 upload 하여 상용소프트웨어(Winamp)를 통하여 압축된 데이터를 재생시켜봄으로써 설계된 칩이 AAC 기본 포맷을 만족시키며 실시간 부호화가 가능하다는 것을 확인하였다.

정량적인 분석을 위하여 upload된 파일에서 압축된 데이터만 추출하여 C 언어로 압축된 데이터와 비교해 본 Ⅲ장의 결과와 동일하게 나오는 것을 확인하였다.

Ⅶ. 결 론

MPEG-2 AAC 부호화기를 70MIPS 이내에서 구현할 수 있는 칩을 제작하였다. 일차적으로 알고리즘을 효율적으로 구현하는 방법에 대하여 연구하였다. 다양한 오디오 데이터를 압축하는 과정에서 얻어진 실험적인 데이터를 활용하여, look-up 테이블과 인터폴레이션을 이용하여 ROM 용량을 줄였다. Odd-time, odd-frequency Discrete Fourier Transform(O^2DFT)을 이용하여 MDCT의 계산양을 1/36으로 줄이는 방법을 선택하였다. Iteration loop의 최적화를 위해서 heuristic한 방법으로 scale-factor 초기값을 구했으며, 허프만 코드북의 인덱스 공동사용을 통하여 허프만 엔코딩 과정의 계산양을 30-40% 정도 줄였다. 이러한 방법을 통하여 AAC 부호화기를 순수하게 S/W적으로 73.1MIPS에 구현할 수 있었다.

0.25um CMOS공정기술에서 실시간으로 AAC 부호화기를 수행하게 하기위하여 17비트 x 17비트 승산기를 기반으로 한 radix-2 FFT를 설계하여 32비트 DSP 코어에 내장함으로써 64.3MIPS에 동작하는 AAC 부호화 칩을 설계하였다. 설계된 칩은 시뮬레이션, 하드웨어 에뮬레이션을 통하여 검증하였다. 검증된 설계는 0.25um CMOS 표준셀을 이용하여 레이아웃 하여 7.20x 7.20mm^2 크기의 칩을 얻었다. 칩의 등가 게이트 수는 약 830,000게이트 이며, 내장된 RAM은 90KB ROM은 106 KB이다.

칩을 70MIPS에 동작하게 클럭을 세팅하여 44.1KHz로 샘플링되어 실시간으로 들어오는 오디오 데이터를 AAC 포맷으로 부호화 하는 것을 확인하였다.

참 고 문 헌

[1] Yuichiro Takamizawa, Toshiyuki Nomura, and Masao Ikekawa, "High-Quality and Prosessor-Efficient Implementation of an MPEG-2 AAC Encoder," 0-7803-7041-4/01 2001 IEEE

[2] http://www.ti.com/

[3] http://www.iis.fraunhofer.de/amm/techinf/audio_dsp/ti.html

[4] ISO/IEC 13818-7, "Generic Coding of Moving Pictures and Associated Audio, Part 7: Advanced Audio Coding(AAC)," Mar. 1997.

[5] M. Bosi and et al., "ISO/IEC MPEG-2 Advanced Audio Coding," J. Audio Eng. Soc., Vol. 45, No. 10, pp. 789-814, Oct., 1997.

[6] K. Brandenburg and M. Bosi, "Overview of MPEG Audio : Current and Future Standards for Low-Bit-Rate Audio Coding," J. Audio Eng. Soc., Vol. 45, pp. 4-21, Jan./Feb., 1997.

[7] Brain C. J. Moore, "An introduction to the Psychology of Hearing, " Academic Press., 3rd

Ed., pp. 84-109, 1989.
[8] Cutis Roads, "The computer music tutorial," MIT Press Cambridge, Massachusetts, London, England, pp. 1051-1069, 1998.
[9] J. P. Princen, A. W. Johnson, and A. B. Bradley, "Subband/transform coding using filter bank designs based on time domain aliasing cancellation," in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing '87, Dallas, TX, pp. 2161-2164. Apr. 1987.
[10] R. Gluth, Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks, Proceedings of the IEEE ICASSP'91, Toronto, Canada, pp. 2205-2208, May 1991.
[11] Texas Instrument, "TMS320C3X User's Guide," july 1992.
[12] 우종식, 서진근, 임재영, 박주성 "32비트 부동소수점 호환 DSP의 설계 및 칩 구현에 관한 연구," 대한전자공학회논문지, 제37권, SD편 제11호, pp. 958-968, 2000. 11
[13] Jin Keun Seo, Jong Sik Woo, Ju Sung, Park, "Design & implementation of 32 bit floating point compatible DSP," IDEC Conference 2000, Cheonan Sangrok Resort, Aug. 17-18, 2000.

저 자 소 개

강 희 철(정회원)
1975년 부산대학교 전자공학과 학사 졸업
2000년 부산대학교 전자공학과 석사 졸업
2005년 부산대학교 전자공학과 박사 졸업 예정
<주요관심분야 : 신호처리, 반도체, 오디오>

박 종 인(정회원)
1982년 경북대학교 전자공학과 학사 졸업.
1999년 부산대학교 전자공학과 석사 졸업.
2002년 부산대학교 전자공학과 박사 수료.
2005년 현재 (주)금영 부설연구소 이사
<주관심분야 : 통신, 컴퓨터, 신호처리, 반도체>

정 갑 주(정회원)
2003년 경성대학교 전자공학과 학사 졸업.
2005년 부산대학교 전자공학과 석사 졸업.
<주관심분야 : 반도체, 오디오알고리즘>

최 병 갑(정회원)
1970년 부산대학교 전자공학과 학사 졸업.
1982년 숭전대학교 전자공학과 석사 졸업.
2000년 부산대학교 전자공학과 박사 수료.
<주관심분야 : 신호처리, 반도체>

박 주 성(정회원)
1976년 부산대학교 전자공학과 학사 졸업
1978년 KAIST 석사 졸업
1989년 Univ. of Florida 전자공학과 박사
1991년~현재 부산대학교 전자공학과 교수
1998년~현재 부산대 IDEC 센터장
<주관심분야 : DSP 설계, ASIC 설계, 음성/사운드 신호 처리 및 구현, SoC 설계>

김 승 우(정회원)
2001년 금오공대 전자공학과 학사 졸업.
2003년 부산대학교 전자공학과 석사 졸업.
2005년 현재 국방과학연구소 연구원
<주관심분야 : 사운드 압축 및 복원,, DSP 설계>

김 태 훈(정회원)
1995년 부산대학교 전자공학과 학사 졸업
1997년 부산대학교 전자공학과 석사 졸업
2002년 부산대학교 전자공학과 박사
2005년 현재 (주)보이소 반도체 S/W 팀장
<주관심분야: 사운드 압축 및 복원, DSP 설계>