ARM926EJ-S 프로세서를 이용한 MPEG-4 BSAC 오디오 복호화기의 구현

전영택*, 박영철**

Implementation of MPEG-4 BSAC Audio Decoder using ARM926EJ-S Processors

Young-Taek Jeon*, Young-Cheol Park**

요 약

국내 지상파 DMB방송 표준에서는 2003년 말 국제 표준으로 제정한 MPEG-4 BSAC(Bit Sliced Arithmetic Coding) 오디오 복호화 방식를 표준으로 채택하였다. 본 논문에서는 MPEG-4 BSAC 오디오 복호화기의 주요 도구 및 모듈에 대해 32비트 고정소수점 연산으로 구현하고 ARM926EJ-S 프로세서에 인라인 어셈블리(Inline Assembly)를 적용하여 최적화 한다. 최적화에 대해 본 논문에서는 RISC프로세서인 ARM926EJ-S의 Core Cycle을 가장 높게 발생시키는 곱셈 및 MAC(Multiply And Accumulation)연산에 집중한다. 그리고 각 모듈 및 도구에서 빈번히 발생하는 곱셈 연산과 MAC연산의 처리를 효율적으로 하기 위하여 대상 프로세서인 ARM926EJ-S에서 사용 가능한 ARMv5용 어셈블리 명령어를 분석하여 사용한다. 최적화된 결과는 MIPS(Million Instruction Per Second)를 기준으로 평가한다. 구현 결과는 96kbps BSAC bitstream을 65MHz CPU clock에서 실시간으로 디코딩할 수 있음을 보여준다.

ABSTRACT

Domestic standard for Korean T-DMB includes MPEG-4 BSAC (Bit Sliced Arithmetic Coding) audio coding that has been established in 2003. This paper presents an implementation and optimization of MPEG-4 BSAC Audio Decoder on ARM926EJ-S processor. Tools and modules of the BSAC audio decoder were implemented with 32-bit fixed point operations. Further optimization was accomplished using ARM926EJ-S Inline Assembly. The optimization was based on the total number of multiplications and MAC (Multiply and Accumulation) operations causing most of core cycles of ARM926EJ-S, and also based on analysis of ARMv5 instructions. The result of optimization was evaluated on the basis of MIPS (Million Instruction per second). Implementation results show that BSAC bitstream at 96kbps can be decoded in real-time at 65MHz CPU clocks.

Key-words : 지상파 DMB, RISC 프로세서, ARM926EJ-S, MPEG-4 BSAC, 필터뱅크, MIPS, MAC

Ⅰ. 서 론

MPEG-4 BSAC은 현재 지상파 DMB방송에 사용되는 오디오 코덱 표준이다[1]. MPEG-4 BSAC은 MPEG-4 AAC의 허프만 복호화 기법에 대한 차선책중 하나로 미세 계층 스케일러빌리티를 제공하는 무손실 압축 도구이다.

MPEG-4 오디오 복호화기의 구현을 위해 다양한 하드웨어 플랫폼이 선택 되어질 수 있다.

ARM 코어와 같은 MCU를 이용한 소프트웨어 구현 방법의 경우 핸드 어셈블리 기술을 이용한다. 핸드 어셈블리는 하드웨어 자원의 효율성을 극대화 시킨 최적화 과정이 필요하다. 이러한 최적화 기법을 적용하여 구현된 오디오 복호화기는 저전력율을 보장하기 위해 최저 MIPS로 동작 하면서 복호화기의 PCM결과가 CD품질의 음질을 보장할 수 있어야한다[10]. ARM7TDMI 프로세서를 이용하여 MPEG-2 AAC 오디오 복호화기를

* (주)시그젠 연구원

** 연세대학교 컴퓨터정보통신기술학부 교수(young00@yonsei.ac.kr)

구현한 경우 최대 32.6MHz의 성능을 보인다고 알려져 있다[10].

DSP 코어 하드웨어 모듈을 이용한 오디오 복호화기 최적화의 디자인은 다음 세 가지 측면에 초점을 맞추어야 한다. 첫 번째는 고품질의 오디오 복호화를 위한 적합성이고, 두 번째는 소비 전력량이 낮아야 하며, 세 번째는 손쉽게 프로그래밍이 가능해야 한다. 16비트 고정 소수점 범용 DSP프로세서인 TeakLite DSP 시스템을 이용하여 MPEG 오디오 복호화기를 구현한 경우 약 43MIPS의 성능을 보인다고 알려져 있다[11].

마지막으로 ASIC과 같은 기술을 이용한 전용 프로세서로써의 구현 방법이 있다. ASIC기술은 목적 시스템에 따라 메모리와 MCU등의 모든 하드웨어 플랫폼을 사용자가 결정할 수 있는 장점이 있다. 그로인해 다른 구현 방법보다 시스템 친화적인 하드웨어의 구성이 가능하다[9].

본 논문의 구성은 다음과 같다. 2장에서는 MPEG-4 BSAC 오디오 복호화기의 구조와 도구들에 대해 분석한다. 3장에서는 구현 및 최적화에 대해 부동 소수점 연산 구현결과를 바탕으로 32비트 고정 소수점 연산의 구현결과를 평가한다. 또한 복호화기의 모듈들 중에 어셈블리를 적용한 모듈에 대해 적용한 방식과 성능에 대해 평가하고 최적화 방법에 대해 설명한다. 4장에서는 구현된 오디오 복호화기의 성능과 정확도에 대해 평가하고 기존의 구현된 오디오 복호화기의 성능과 비교한다. 그리고 마지막 5장에서 결론을 맺는다.

II. MPEG-4 BSAC 오디오 복호화기

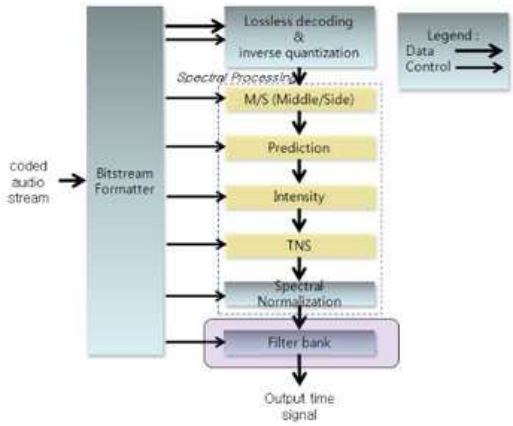

그림 1. MPEG-4 BSAC 복호화기 블록도
Fig. 1 Block diagram of MPEG-4 BSAC decoder

그림-1은 MPEG-4 BSAC 오디오 복호화기의 블록도이다. 블록도의 실제 모든 도구 및 모듈에 대해서 구현한다.

M/S 스테레오 부호화는 원 신호의 스테레오 음상에 대응하도록 부호화 잡음의 상을 제어하는 역할을 한다.

$$M=(L+R)/2 \qquad L=(M+S)/2$$
$$S=(L-R)/2 \qquad R=(M-S)/2$$

M/S 스테레오 부호화는 주파수 축에서 스케일 팩터 대역 단위 뿐만 아니라 블록별로도 선택하여 사용될 수 있다[3]. 음압 스테레오 부호화는 스테레오 신호에 대해 선택적으로 사용할 수 있는 부호화 방식으로 MPEG-1에서부터 사용되던 기술이다[7][8]. TNS는 AAC에서부터 사용되어 오던 지각 오디오 부호화로써 MPEG-4 BSAC 오디오 복호화기 에서도 동일하게 사용한다[4].

고음질 오디오 압축 복호화 시스템의 가장 기본적인 요소는 부호화기에서 시간영역의 신호를 내부적인 주파수 표현으로 바꾸는 것과 복호화기에서 이의 역 변환을 수행하는 것이다. AAC에서와 마찬가지로 MPEG-4 BSAC 오디오 복호화기에서 이 변환과정을 전 방향 IMDCT에 의해 수행한다. IMDCT에는 시간영역의 에일리어징 제거라 불리는 기법이 적용 된다[5][6].

IMDCT를 나타내는 수식은 다음과 같다.

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

여기서,

n = sample index

i = window index

k = spectral coefficient index

N = window length based on the window sequence value

$n_0 = (N/2+1)/2$

복호화기에서 이 과정의 실제 구현을 위해서는 주파수축의 계수들에 대해 적절한 블록을 취하고, 적절한 윈도우 함수에 의하여 그것들을 복조시킨 후 IMDCT를 수행한다. 입력 계수들의 각 블록은 이전 블록과 이후 블록이 50%씩 겹치게 되어 있다. 역 변환 블록 길이 N은 2048(Long Block)이나 256(Short Block)이 사용된다.

Ⅲ. 구현 및 최적화

3.1 고정 소수점(Fixed-Point) 연산의 구현

본 논문에서는 IMDCT를 구현하기 위해 과도한 메모리와 연산량 필요를 감소시키기 위한 FFT알고리즘을 사용하였다[6]. N차의 FFT는 NlogN의 연산량을 필요로 한다. 이러한 알고리즘을 통해 long 블록의 경우 약 15*2048의 곱셈과 덧셈, 6*2048개의 메모리를 필요로 하며 short 블록의 경우 약 12*256의 곱셈과 덧셈, 6*256개의 메모리를 필요로 한고 연산량은 약 1/80, 메모리는 1/150 정도로 감소하게 된다. 이 복호화기 에서는 이를 더욱 향상시킨 알고리즘으로 N차의 FFT가 아닌 N/4차의 inverse FFT를 이용하는 알고리즘을 사용하였다[6]. Long 블록의 경우 17*512만큼의 곱셈과 덧셈, 4*512만큼의 메모리를 필요로 하며 short 블록의 경우 14*64만큼의 곱셈과 덧셈, 4*64만큼의 메모리를 필요로 한다. 따라서 short 블록을 사용하는 경우 8개의 IMDCT가 동작함을 고려하더라도 long 블록의 경우보다 더 적은 연산량을 소모함을 알 수 있다. 결과적으로 연산량은 약 1/200로 메모리는 1/1000 정도로 감소하게 된다. 단 IMDCT를 하기 전과 후에 재 정렬하는 과정이 필요하기 때문에 메모리 접근과정에서 추가적인 사이클을 소모하게 된다.

추가적인 연산량 감소를 위하여 FFT와 twiddle을 위한 상수들은 모두 16비트를 사용하였다. 고정소수점 연산의 정확도를 측정하기 위해 본 논문에서는 부동 소수점 연산 결과와 32비트 고정 소수점 연산결과의 차이를 이용하였다. 위 그림-2에서 보는 바와 같이 32비트 고정소수점 연산과 부동소수점 연산 결과의 차이는 1비트 이하의 차이를 보인다.

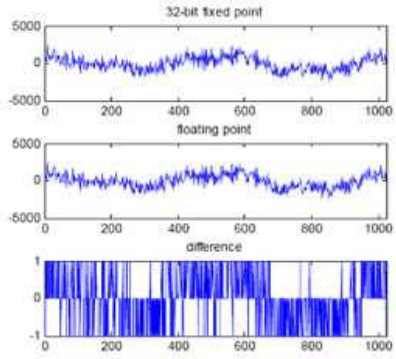

그림 2. 부동 소수점 연산과 32비트 고정소수점 연산의 차이
Fig. 2 Difference between floating-point processing and 32-bit fixed-point processing

3.2 어셈블리 코딩을 통한 저 전력 MPEG-4 BSAC 복호화기 개발

IMDCT 의 경우 많은 곱셈 및 MAC연산이 반복적으로 일어나기 때문에 MCU의 실행 사이클을 줄이는데 집중 하였다. 이를 위해 ARMv5에서 사용 가능한 곱셈 및 MAC연산을 위한 명령어를 신중히 선택할 필요가 있다.

SMULL, SMLAL, UMULL이나 MULAL과 같은 32비트 곱셈 명령어는 Table lookup 데이터로 unsigned short형을 사용하여 연산 결과의 1비트만큼의 해상도의 이득을 얻을 수 있다. 하지만 실행 사이클이 SMULWB, SMLAWB보다 많이 필요하게 되어 반복적으로 수행되는 루틴에 적합하지 않다. SMULWB, SMLAWB과 같은 16비트 곱셈 명

령어를 사용하면 한 사이클에 곱셈을 수행할 수 있다. 단 이 경우 16비트 상수는 signed short형만을 지원하기 때문에 unsigned short를 이용하던 루틴보다 1비트만큼의 해상도의 이득을 얻을 수 없어 RMS가 1dB정도 감소하였다.

IMDCT 루틴 내에서 인라인 어셈블리의 적용시에 unsigned short의 해상도를 가지도록 의도된 변수 값에 대해 SMULWB가 signed short만을 지원하기 때문에 본래의 변수값의 1/2값을 사용하도록 look-up table을 수정하였다. 따라서 1/2만큼 나눈 값이 얻어지기 때문에 이를 보상하기 위하여 LSL 명령어를 이용하여 1비트를 올려주는 과정이 추가되었다.

unsigned short연산을 지원하는 명령어를 가지는 프로세서에서는 이러한 불필요한 과정을 제거할 수 있고 그에 따라 연산속도를 향상시킬 수 있으며 해상도에서 1비트의 이득을 얻을 수 있다.

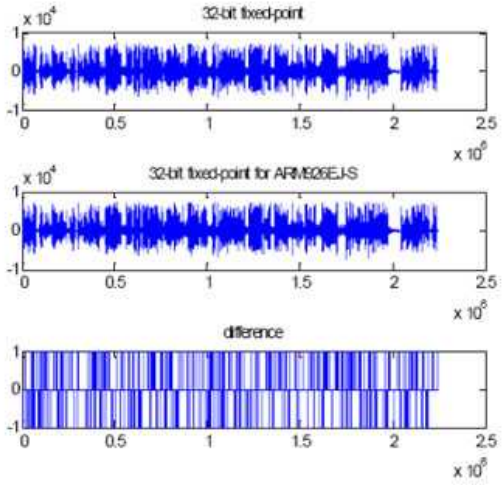

그림 3. 인라인 어셈블리연산과 32비트 고정소수점 연산의 차이
Fig. 3 Difference between Inline Assembly and 32-bit fixed-point processing.

인라인 어셈블리를 적용한 프로그램의 정확도를 측정하기 위해 본 논문에서는 고정소수점 연산에서와 마찬가지로 인라인 어셈블리 연산 결과와 32비트 고정소수점 연산 결과의 차이를 이용한다. 위 그림-3에서 보는바와 같이 인라인 어셈블리 연산 결과와 32비트 고정소수점 연산 결과의 차이는 1비트 이하의 차이를 보인다.

IV. 성능평가

4.1 필터뱅크(IMDCT)에 대한 성능평가

주요 모듈 중 하나인 필터 뱅크는 곱셈 연산과 MAC연산이 가장 빈번히 발생하는 모듈 중에 하나이다. 제 3장에서 설명한 바와 같이 look up table 데이터를 unsigned shot형을 사용 했을 경우와 signed short를 사용 했을 경우에 따라 각각 다른 성능을 보인다. 빈번히 발생하는 곱셈과 MAC 연산에 대해 signed short형을 사용하면 어셈블리 명령어의 선택에 따라 실행 사이클이 1사이클만 줄어들어도 큰 이득을 얻을 수 있는 반면 부호비트로 MSB를 사용하게 되어 전체적인 해상도는 1비트만큼 줄어들게 된다. 아래 표-1과 표-2에서 그 결과를 MIPS로 확인할 수 있다.

표-1 Unsigned Multiplication을 사용한 경우
Table 1 MIPS when unsigned multiplication was used

	계산량(MIPS)		
	M/S	TNS	IMDCT
Long Block	0.62	0.55	22.96
Short Block	0.61	0.56	29.33

표-2 Signed Multiplication을 사용한 경우
Table 2 MIPS when signed multiplication was used

	계산량(MIPS)		
	M/S	TNS	IMDCT
Long Block	0.62	0.55	22.96
Short Block	0.61	0.56	29.33

위 표에서 보는바와 같이 Signed Multiplication을 사용한 경우가 Unsigned Multiplication의 경우보다 IMDCT는 최소 7MIPS에서 17MIPS까지의 이득을 얻을 수 있다.

4.2 MPEG-4 BSAC 복호화기의 성능평가

MPEG-4 BSAC 오디오 복호화기의 성능을 평가하기 이전에 기존의 구현된 오디오 복호화기의 성능을 살펴볼 필요가 있다.

먼저 16비트 고정 소수점 범용 DSP 프로세서인 TeakLite시스템을 사용하여 MPEG-2 AAC 오디오 복호화기를 구현한 경우이다. 아래의 표-3은 TeakLite 시스템을 이용하여 구현한 MPEG-2 AAC 오디오 복호화기의 연산량을 나타낸다[11].

또한, ASIC과 같은 시스템을 이용한 구현 방법으로써 VLSI시스템을 이용한 구현이다. 이 경우는 각 모듈에 적합하도록 하드웨어의 메모리 구조나 게이트 구조를 사용자가 직접 결정함으로써 시스템에 최적화된 구현을 그 목적으로 한다. 다음 표-4는 MPEG-4 오디오복호화기의 VLSI 구현의 성능을 나타낸다[9].

마지막으로 본 논문에서와 같이 32비트 MCU RISC 프로세서를 이용한 구현 방법이다. 대표적인 RISC 프로세서인 ARM계열의 프로세서로써 ARM7TDMI를 사용하여 구현한 경우 일반적인 고정 소수점 범용 DSP프로세서의 비해 저전력율이 보장될 수 있다는 장점이 있다. 다음 표-5는 ARM7TDMI를 이용하여 MPEG-오디오 복호화기를 구현한 성능을 나타낸다[10].

표-3 TeakLite 시스템을 이용한 AAC 오디오 복호화기의 성능[11]
Table 3 AAC audio decoder implemented using TeakLite

	Data Memory (Kword)	Program Memory (Kword)	연산량 (MIPS)
역양자화	0.1	0.04	2.5
M/S	-	0.13	0.8
TNS	-	0.44	-
Filter Bank	19.7	0.82	25

표-4 MPEG-4 오디오 복호화기의 VLSI 구현
Table 4 VLSI Implementation of MPEG-4 audio decoder

	게이트수	Cycles/frame	MIPS/frame
역양자화	794	2,048	0.096
TNS	1,815	8,130	0.38
Filter Bank	10,070	22.144	1.038

표-5 ARM7TDMI를 이용한 AAC복호화기의 구현
Table 5 Implementation of MPEG-4 audio decoder using ARM7TDMI

AAC	연산량(MIPS)
Bit stream	5.868
Decoding	1.956
TNS	7.172
IMDCT	17.604
Total	32.6

본 논문에서는 위 구현 결과를 바탕으로 MPEG-4 BSAC 오디오 복호화기의 전체 블록에 대한 구현 결과를 평가한다. 가장 좋은 비교 대상은 동일한 MCU RISC프로세서로써 하위 버전의 ARM7TDMI를 하드웨어 플랫폼으로 사용한 경우이다. 위 표-5에서 IMDCT모듈을 살펴보면 본 논문의 구현 결과와 비교하여 약 1MIPS정도 낮은 성능을 보이는 것을 알 수 있다.

본 논문에서 구현한 MPEG-4 BSAC 오디오 복호화기 블록도에 있는 전체 도구 및 모듈의 성능 평가에 사용된 입력 샘플은 부호화과정을 거친 비트 스트림으로써 총 다섯 개의 샘플을 사용하였다. 아래의 표-6은 다섯 가지 샘플에 대한 간략한 설명이다.

표-6 성능 평가를 위한 부호화된 MPEG-4 샘플
Table 6. MPEG-4 audio samples for performance evaluation

샘플파일명	Fs(kHz)	채널	샘플 설명
cast64.mp4	44.1	1	캐스터네츠를 연주하는 오디오 샘플
harp64.mp4	44.1	1	하프오르간 연주를 녹음한 오디오 샘플
KBSS.mp4	44.1	2	KBS 라디오 방송을 녹음한 오디오 샘플
MBC.mp4	48	2	MBC 라디오 방송을 녹음한 오디오 샘플
voice64.mp4	44.1	1	여성의 음성 을 녹음한 음성 샘플

위 다섯 가지 입력 샘플로 블록내의 도구 및 모듈에 대해 MIPS를 측정한 결과는 아래 표-7에 나타나 있다.

표-7 MPEG-4 BSAC 오디오 복호화기 전체 블록의 평균 MIPS
Table 7 Average MIPS of the total blocks of MPEG-4 audio decoder

샘플 파일명	계산량(MIPS)			
	BSAC 무손실 복호화	역 양자화	M/S	필터뱅크
cast64.mp4	19.11	1.42	-	7.70
harp64.mp4	40.49	1.75	-	7.71
KBSS.mp4	48.94	4.31	0.94	15.37
MBC.mp4	56.75	3.73	1.25	16.62
voice64.mp4	26.99	1.41	-	7.63
평균	38.46	2.52	1.09	11.01

BSAC 무손실 복호화 과정과 역양자화 그리고 필터뱅크를 제외한 그 이외의 도구들은 부호화과정을 마친 입력 비트 스트림의 신호 특성에 따라 선택적으로 수행된다. 특히 PNS, 음압 스테레오 부호화, TNS와 같은 도구는 본 논문에서 사용한 입력 비트 스트림에서 거의 사용되어지지 않아 평균 MIPS를 계산하는 것이 큰 의미를 가지지 않기 때문에 그 도구들을 제외한 나머지 모듈에 대해 평균 MIPS를 계산하였다.

위 결과에서 보는바와 같이 BSAC 무손실 복호화 과정은 입력 비트 스트림의 비트 율에 따라서 크게 차이가 나는 것을 볼 수 있다. 이는 BSAC이 FGS를 제공하기 때문에 얻을 수 있는 결과이다. 그에 반해 필터뱅크의 경우 블록 사이즈가 정해져 있어서 거의 모든 샘플에서 동일한 성능을 보인다. 하지만 필터뱅크는 내부적으로 FFT와 같은 복잡하고 반복적인 루프구조를 갖는 모듈이 포함되어 있어 상대적으로 다른 도구들에 비해 더 많은 계산량을 보인다.

4.3 적합성 시험(Conformance Test)

적합성 시험에서 사용되는 ISO 테스트 벡터를 복호화 하여 얻은 출력을 비교하여 RMS of difference와 Maximum peak Difference를 얻고 이것이 순응도 기준을 만족함을 확인한다. 다음 표-8은 MPEG-2 AAC의 Full Accuracy Requirement와 각각의 오디오 샘플에서 얻은 RMS of difference와 Maximum peak Difference를 보여준다.

표-8 적합성 시험
Table 8 Conformance Test

오디오 샘플	RMS of difference(dB)	Maximum peak Difference
Reference	-101.1008	6.1035 x 10-5
cast64.mp4	-108.1258	3.0518 x 10-5
harp64.mp4	-105.7412	3.0518 x 10-5
BSS.mp4	-106.3573	3.0518 x 10-5
MBC.mp4	-106.1894	3.0518 x 10-5
voice64.mp4	-106.1249	3.0518 x 10-5

RMS of difference(dB)와 Maximum peak difference 모두 모든 샘플에 대해 만족하는 것을 확인할 수 있다. Reference의 Maximum peak Difference인 6.1035 x 10-5이 나타내는 것은

2/32768로서 최대의 차이가 2가 되는 경우를 의미한다. 주어진 모든 샘플에서는 1의 차이만을 보이는 것을 확인할 수 있다.

4.4. 구현된 BSAC decoder의 복잡도 측정

연산량이 최대가 되는 시나리오인 48KHz stereo 오디오 샘플에 대해 64, 96, 128kbps의 세 가지 비트 율에 대해 측정하였다. 각각의 비트 율에 대해 각각 최대 43, 65, 87MHz의 연산량이 측정되었다.

표-9 전체 모듈의 연산량
Table 9 Complexity of overall modules

모듈	연산량(MHz)		
비트율 (kbps)	64	96	128
무손실 복호화	22	44	66
도구	4		
필터뱅크	Long Block : 17 Short Block : 13		
Total	43	65	87

각각의 모듈 별로 측정할 경우 64kbps의 경우 BSAC 무손실 복호화, 역 양자화 & 도구들, 필터뱅크 모듈에 대해 각각 22, 4, 17MHz의 연산량이 측정되었다. BSAC 무손실 복호화 과정의 경우 비트 율이 64, 96, 128kbps와 같이 증가함에 따라 연산량도 22, 44, 66과 같이 증가하였다. 이는 비트 율이 증가함에 따라 복호화 단에서 더 많은 향상층을 기록하게 되며 따라서 복호화 과정을 거치면서 그에 비례하는 반복문을 수행하기 때문이다. 역 양자화와 도구들의 경우 MS와 TNS만이 동작하는 시나리오에서 측정되었으며 각각 역 양자화가 3MHz, MS가 0.5MHz, TNS가 0.7MHz로 측정되었다. 이는 비트 율에 따라 변동되지 않으며 다만 TNS의 경우 특정 프레임에서만 동작하는 특징이 있다. 필터뱅크 모듈의 경우 역시 비트 율에 따른 연산량 변화는 없으며 Long 블록인 경우와 Short 블록인 경우 각각 연산량이 17, 13MHz로 측정되었다.

필터뱅크 모듈은 BSAC 무손실 복호화 모듈을 제외하고 가장 많은 연산량을 필요로 한다. 그에 따라 Long 블록과 Short 블록에 대해 성능평가가 필요하다. 필터뱅크 모듈의 성능은 ARM 명령어의 선택에 따라 크게 차이를 보였다.

V. 결 론

본 논문에서는 MPEG-4 BSAC 오디오 복호화기 블록도에 있는 모든 도구 및 모듈에 대해 고정소수점 연산의 구현 및 최적화를 하는 것을 목적으로 하였다. 도구 및 모듈에 대해서 곱셈 및 MAC연산에 집중하여 구현하였으며 최적화를 위해 인라인 어셈블리를 이용하였다. 구현된 복호화기의 성능은 어셈블리 명령어 선택에 따라 최소 7MIPS이상의 이득을 얻을 수 있었다. 또한 필터뱅크 모듈의 경우 본 논문의 구현 결과는 기존 연구 결과와 비교해서 약 1MIPS정도 성능의 이득을 얻을 수 있었다.

본 논문에서의 구현을 바탕으로 향후 다양한 부호화 과정을 거친 MPEG-4 오디오 파일을 이용하여 성능을 검증하고 구현되지 않은 BSAC 무손실 복호화 과정을 구현하고 성능을 최적화 할 필요가 있다.

참고문헌

[1] ISO/IEC 14496-3, ""Information technology-Codi ng of audio-visual objects, Part 3: Audio,"" Dec. 2001

[2] ARM Ltd. Homepage (http://www.arm.com)

[3] J.Johnston and A. Ferreira, "Sum-Difference Stereo Transform Coding," in Proc. IEEE ICASSP, pp. 569-571, 1992.

[4] J. Herre and Johnston, "Enhancing the Performance of Perceptual Audio Coders by Using Temporal Noise Shaping (TNS)," presented at the 101st convention of the Audio Engineering Society, preprint 4384

[5] Kyoung Ho Bang, Joon Seok Kim, Nam Hun Jeong, Young Cheol Park, and Dae Hee Youn, "Design Optimization of MPEG-2 AAC Decoder," in Proc. Intl. Conf. Consumer Electronics, Los Angeles, California, USA, June 17-21, 2001.

[6] P. Duhamel, Y. Mahieux and J. P. Petit, "A Fast Algorithm for the Implementation of Filter Banks Based on Time Domain Aliasing Cancellation," In Proc. IEEE Int. Conf, Acoust., Speech, Signal Processing, pp. 2209-2212, 1991.

[7] ISO/IEC JTCI/SC29/WG11 No.71 "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s - CD 11172-3(Part3.MPEG-Audio)"

[8] J. Herre and K. Brandenburg, "Intensity Stereo Coding." Audio Engineering Society 96th Convention, preprint 3799, 1994.

[9] S. Hashimoto, et, al.,"VLSI implementation of portable MPEG-4 audio decoder," Proc. 13th Annual IEEE Intl. ASIC/SOC Conf., pp. 80~84, Sept. 2000.

[10] Keun-Sup Lee Young-Cheol Park and Dae Hee Youn, "Software Optimization of the MPEG-Audio decoder Using a 32-bit MCU Risc Processor," IEEE Transactions on Consumer Electronics, Vol. 48 No. 3 pp.671-676, Aug. 2002.

[11] 장봉근, 정종훈, 장태규, 장홍엽, "TeakLite DSP에 기초한 MPEG-2 AAC decoder의 최적 구현에 관한 연구", 한국음향학회, 2000년도 학술발표대회 논문집, 제 19권, 제 2(s)호, pp.119-122

저자약력

전 영 택 (Young-Taek Jeon)

2002년 연세대학교 전산학과 이학사
2008년 연세대학교 전산학과 이학석사
2008년 현재 (주)시그젠 연구원

<관심분야> 디지털 신호처리, 음성/오디오 신호처리, 적응 필터

박 영 철 (Young-Cheol Park)

1986년 연세대학교 전자공학과 공학사
1988년 연세대학교 전자공학과 공학석사
1993년 연세대학교 전자공학과 공학박사
2002년-현재 연세대학교 컴퓨터정보통신 기술학부 교수

<관심분야> 디지털 신호처리, 음성/오디오 신호처리, 적응 필터