레터논문-08-13-3-13

휴대용 임베디드 프로세서에서의 MPEG-4 오디오의 실시간 재생을 위한 정수 디코딩 기법

차 경 애[a)‡]

MPEG-4 Audio Decoding Technique using Integer Operations for Real-time Playback on Embedded Processor

Kyung-Ae Cha[a)‡]

요 약

소형의 휴대용 단말기는 회로복잡도나 소비전력 등의 문제로 부동소수점 연산 프로세서를 탑재하지 않는 경우가 있는데, 이로 인해 오디오 데이터의 디코딩 시간이 길어져, 끊김이나 잡음이 발생한다. 본 논문에서는 이를 해결하기 위해서 MPEG-4 오디오 디코딩 시 수행되는 실수형 연산과정을 정수형 연산과정으로의 변환을 통하여 디코딩 속도를 향상 시킬 수 있는 알고리즘을 제안하고 실험결과를 통해서 효율성을 보인다.

Abstract

Some embedded microprocessors do not have an FPU(Floating Point Unit) due to a circuit complexity and power consumption. The performance speed of MPEG-4 AAC decoder on this hardware environment would be slower than corresponding speed for playing back of the decoded results. Therefore, irritating and high-pitched noises are interleaved in the original the audio data. So, in order to play MPEG-4 AAC file on such PDA, a new algorithm that transforms floating-point arithmetic to one with integers, is needed. We have developed a transformation algorithm from floating-point operation to integer operation and implemented the PDA's AAC Player. We also show the efficiency of our proposed method with the experimental results.

Keywords : MPEG Audio, MPEG-4 AAC, Embedded processor, IMDCT

I. 서 론

휴대용 단말기가 보편화됨에 따라 내장형이나 초소형 단말기의 인기가 높아지고 있다. 소형의 휴대단말기는 회로의 복잡도, 발열 문제와 소비 전력 문제 등으로 인하여 부동 소수점 연산 프로세서(Floating-point Processing Unit, FPU)가 없는 경우가 있다. 대표적으로 인텔사의 스트롱암(206MHz 32비트 인텔 StrongARM SA-1110) 프로세서를 들 수 있다[1].

한편 MPEG-4 AAC는 높은 음질과 압축률을 제공할 뿐만 아니라 임베디드 시스템에서의 오디오 데이터로써 활용도가 높다[2]. 그러나 FPU가 지원되지 않는 단말기에서 다량의 연속적인 실수 연산을 필요로 하는 MPEG-4 오디오 데이터의 디코딩은 연산 지연을 발생시켜, 재생 시 소리의

a) 대구대학교 정보통신공학부
School of Computer & Communication Engineering,, Daegu University
‡ 교신저자 : 차경애(chaka@daegu.ac.kr)
※이 논문은 2007학년도 대구대학교 학술연구비 지원에 의한 논문임

끊김 현상이나 잡음 현상을 초래한다.

본 논문은 임베디드 시스템을 탑재한 소형 기기에서 MPEG-4 오디오의 효과적인 디코딩을 위한 정수형 연산 변형 알고리즘을 제안한다. 즉, AAC 데이터 디코딩에서 실수 연산이 포함되는 IMDCT 과정의 세 단계인, Pre-twiddling, IFFT, Post-twiddling의 각 연산을 정수형을 사용하도록 변형하여 디코딩 지연을 방지하는 방법을 제안하고 실험결과를 통해서 효율성을 보인다.

제 2 장에서는 FPU가 없는 시스템에서 MPEG-4 AAC 데이터 디코딩 시 사용된 알고리즘들을 실수형 연산에서 정수형 연산으로 바꾸는 방법을 제시한다. 제 3 장에서는 구현 환경 및 개발 결과에 대해서 보이고, 연산 변형을 위해 제시한 방법들에 대한 오차 및 SNR 값을 측정해본다. 마지막으로 제 4 장에서는 결론 및 향후 연구방향을 제시한다.

II. MPEG-4 AAC의 정수형 연산 변형

MPEG-4 AAC 오디오 데이터의 디코딩 과정에서 실수연산이 필요한 IMDCT(Inverse Modified Discrete Cosine Transform) 연산식은 식 (1)과 같다[3,4,5].

$$\hat{x}(n)=\frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1}X(k)\cos\left[\frac{\pi}{2N}(2n+1+\frac{N}{2})(2k+1)\right] \quad (1)$$
$$n=0,1,...,N-1$$

IMDCT 과정을 효율적으로 수행하기 위해서, 실제 구현상에서 식 (1)의 각 항을 계산하는 과정은 Pre-twiddling, IFFT(Inverse Fast Fourie Transform), Post-twiddling의 세 단계로 나누어진다[3-7].

1. Pre-twiddling 과정

Pre-twiddling에서 수행되는 연산식은 식 (2)와 같다.

$$X'(k)=X(k)(\cos(\frac{2k\pi n_0}{N})+jsin(\frac{2k\pi n_0}{N})) \quad (2)$$
$$j=\sqrt{-1}, n_0=\frac{1}{2}+\frac{N}{4}$$

식 (2)에서 입력값은 $X(k)$ 이며 N은 윈도우 블록의 크기를 나타내며, 긴 윈도우 및 짧은 윈도우의 크기로 그 값이 결정된다.

인코딩된 데이터의 실수영역의 값과 허수영역의 값을 각각 $a(k)$ 와 $b(k)$ 라고 할 때 식 (3)과 같이 복소수로 표현된다.

$$X(k)=a(k)+jb(k) \quad (3)$$

위 식 (2)에 식 (3)을 대입하면 다음과 같다.

$$X'(k)=a(k)\cos(\frac{2k\pi n_0}{N})-b(k)\sin(\frac{2k\pi n_0}{N}) + j(a(k)\sin(\frac{2k\pi n_0}{N})+b(k)\cos(\frac{2k\pi n_0}{N})) \quad (4)$$

식 (4)의 실수부와 허수부는 삼각함수로 인해서 실수연산을 필요로 하게 된다. 하지만, $\cos(\frac{2k\pi n_0}{N})$와 $\sin(\frac{2k\pi n_0}{N})$에서 윈도우 크기 N이 결정되면 식 (2)에서와 같이 n_0의 값도 결정되기 때문에 이들의 값도 함께 결정된다. 이러한 삼각함수 값들을 최소한의 오차로 하는 정수로의 변환을 위해서는 정수범위에 해당하는 값 중 가장 큰 값을 곱해주어야 한다.

우선 식 (4)에서 우변의 실수부와 허수부 값이 존재해야 하는 범위를 고려하면 32비트 정수형일 경우 -2^{n-1}과 $2^{n-1}-1$ 사이이다. 이 조건을 항상 만족하도록 실수부와 허수부의 각 항의 범위가 $2^{n-2}-0.5$보다 작거나 같다고 가정한다.

예를 들어 허수부의 경우를 살펴보면 다음과 같이 그 범위를 표현할 수 있다.

$$-2^{n-1}\le a(k)\sin(\frac{2k\pi n_0}{N})+b(k)\cos(\frac{2k\pi n_0}{N}))\le 2^{n-1}-1$$

이 때 두 항인, $a(k)\sin(\frac{2k\pi n_0}{N})$ 와 $b(k)\cos(\frac{2k\pi n_0}{N}))$ 항이 각각 $2^{n-2}-0.5$보다 작다면 정수범위의 최대값을 초과하는 경우가 발생하지 않는다.

따라서 실수부와 허수부의 각 항의 삼각함수값을 정수로 변환하는 과정에서 오차를 최소로 하기 위해서 곱해야 할 값은 $2^{n-2}-0.5$가 된다. 여기서, 0.5의 값은 무시하도록 한다.

이제 식 (4)를 아래 식 (5)와 같이 변형한다.

$$a(k)\times\cos(\frac{2k\pi n_0}{N}) = (2^{n-2}\times\cos(\frac{2k\pi n_0}{N}))/(2^{n-2}/a(k)) \quad (5)$$

이미 언급한 바와 같이 $\cos(\frac{2k\pi n_0}{N})$의 값은 윈도우 크기 N에 의해 미리 알 수 있기 때문에 식 (5)에서 $2^{n-2}\times\cos(\frac{2k\pi n_0}{N})$의 값은 모든 k에 대해서 미리 계산할 수 있는 정수값이다. 또한 $a(k)$와 $b(k)$ 값은 인코딩 된 데이터로부터의 입력값이며, 정수형으로 표현 가능하다. 그러므로 식 $2^{n-2}/a(k)$는 정수 연산으로 수행된다. 이와 같이 식 (5)는 미리 계산된 분자의 정수값을 분모의 정수 연산 결과 값으로 나누는 연산식이 된다.

위와 같은 방법으로 식 (4)에서 보이는 $b(k)\sin(\frac{2k\pi n_0}{N})$, $a(k)\sin(\frac{2k\pi n_0}{N})$, $b(k)\cos(\frac{2k\pi n_0}{N})$ 값에 대해서도 정수 연산을 수행할 수 있다.

2. IFFT 과정

IFFT 과정은 Pre-twiddling의 결과값인 $X'(k)$를 이용하여 식 (6)과 같이 이루어진다.

$$s'(k) = \frac{1}{N}\sum_{k=0}^{N/2-1} X'(k)[\cos(\frac{2nk\pi}{N}) + jsin(\frac{2nk\pi}{N})] \quad (6)$$

여기에서 $X'(k)$의 값은 Pre-twiddling 과정에서 변환한 정수형 값이다. 또한 $\cos(\frac{2nk\pi}{N})$과 $\sin(\frac{2nk\pi}{N})$의 값 역시 Pre-twiddling 과정에서 보인 바와 동일한 방법으로 미리 계산되어 정수형으로 변환된 값을 실제 계산식에서 사용할 수 있다. 따라서 식 (6)은 정수연산으로 변형가능하다.

3. Post-twiddling 과정

IFFT의 결과인 $s'(k)$는 복소수형태이므로, 다음과 같이 나타낼 수 있다.

$$s'(k) = c(k) + jd(k) \quad (7)$$

Post-twiddling 과정은 $s'(k)$를 입력으로 하는 다음 식 (8)의 연산식으로 이루어진다.

$$s(n) = 2s'(k)[\cos\frac{\pi}{N}(n+n_0) + jsin\frac{\pi}{N}(n+n_0)] \quad (8)$$

여기에서 식 (7)을 식 (8)에 대입하면 식 (9)를 얻을 수 있다.

$$\begin{aligned} s(n) = 2[&c(k)\cos\frac{\pi}{N}(n+n_0) - d(k)\sin\frac{\pi}{N}(n+n_0) \\ &+ j(c(k)\sin\frac{\pi}{N}(n+n_0) + d(k)\cos\frac{\pi}{N}(n+n_0))] \end{aligned} \quad (9)$$

이는 Pre-twiddling 과정에서 보인 바와 같이 미리 계산되어 정수형으로 변환된 값을 실제 계산식에서 사용할 수 있다. 따라서 식 (9)은 정수연산으로 변형가능하다.

이렇게 생성된 결과값은 Half-Cycle sine 윈도우 알고리즘을 거치게 되는데, 이는 다시 위와 동일한 방법으로 정수형 변환을 하였다.

III. 구현 및 실험 결과

PDA용 MPEG-4 AAC 재생기의 구현을 위해 내장형 리눅스 기반의 샤프(Sharp)사의 자우루스(Zaurus) SL-5500을 이용하였다. 운영체제는 내장형 리눅스 커널을 사용하였으며, 탑재된 커널은 Kernel 2.4.6--rmk1-np2-embedix이다.

표 1은 실험에 사용한 전체 재생시간이 78초인 음악 데이터를 MPEG-4 AAC LC(Low Complexity)로 인코딩한 후 실수형 및 제안한 정수형 연산을 통해 디코딩한 결과를 보이고 있다.

표 1. 디코딩 시간 비교: 정수연산과 실수연산
Table 1. Decoding time Comparison : Integer operation and Float operation

원본 데이터(wav)		AAC 인코딩		디코딩속도(sec)		
재생시간 (sec)	샘플링율 (Hz)	비트율 (bps)	채널	PC 실수형 연산	PDA	
					실수형 연산	정수형 연산
78	44100	64000	스테레오	4.65초	1443.03	65.35
	8000	8000	모노	0.31초	350.14	11.57

PDA에서의 실수형 연산을 통한 디코딩 시간은 원곡의 재생 시간의 약 18.5배로서 재생이 불가능하였으나, 정수형 연산을 이용한 디코딩 시간은 원곡의 재생시간보다 작으므

로 끊김이 없이 재생되었다. 이 외에 다양한 장르의 음원에 대한 실험 결과도, 실수형 연산으로의 디코딩 시간은 재생 시간보다 훨씬 길었지만 제안한 정수형 연산으로의 디코딩 시간은 재생시간보다 짧다는 것을 알 수 있었다.

그림 1은 표 1의 실수형 연산과 정수형 연산의 결과값의 상대오차의 분포를 나타낸다. 그래프의 가로축은 두 연산 결과값의 차이를 실수형 연산 결과값으로 나눈 백분율에 해당한다. 세로축은 해당 범위에 대한 속하는 상대오차값을 가지는 샘플의 개수를 전체 샘플 개수로 나눈 값을 백분율로 계산한다. 예를 들면 0.5%의 상대오차를 가지는 데이터가 약 24%로 분포되며, 8% 이상의 상대오차를 가지는 데이터는 거의 나타지 않는다는 의미이다. 즉 대부분의 상대오차는 3%미만의 범위에 속하고 있다.

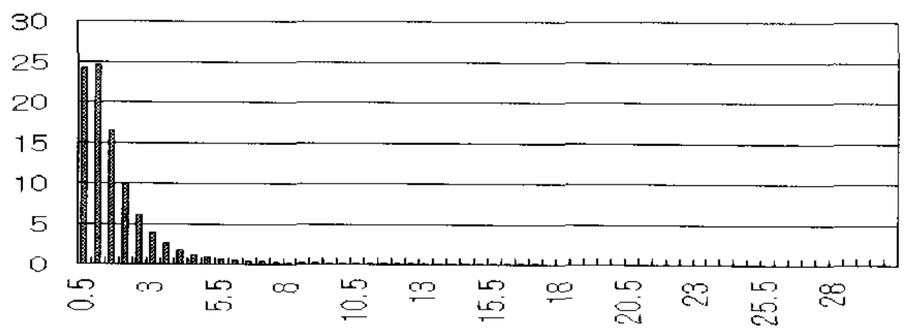

그림 1. 상대오차의 분포도

Fig. 1. The Distribution of Relative Error Rate

그림 2는 SNR(Singal-to-Noise Ratio, $20\log_{10}(S/N)$)값을 나타낸다. S는 실수형 연산 디코딩값이며 N은 정수형연산과의 절대오차값이다. 모든 실험은 긴 윈도우에 대해서 행하였고, 이에 대한 평균값으로 표현하였다. SNR의 평균값은 약 106dB이다. 이는 정수형 연산으로 복원된 데이터가 원 신호에 섞인 잡음은 재생 음질에 영향을 미치지 않을 정도임을 알 수 있다.

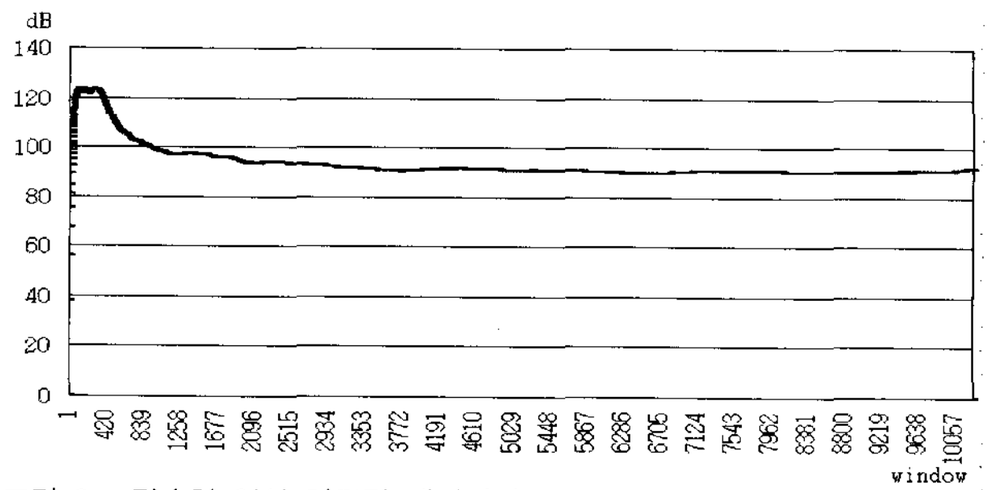

그림 2. 정수형 연산 디코딩 데이터의 SNR

Fig. 2. The SNR of Decoding Results from Integer operation

Ⅳ. 결론 및 향후 연구방향

본 논문은 FPU가 없는 PDA 환경에서 MPEG-4 AAC 데이터의 재생을 위해 디코딩 과정에서 필요한 실수형 연산을 정수형 연산으로의 변환에 관한 연구이다. 실험결과, 제안한 정수형 연산을 통해서 오디오 데이터의 끊김이나 지연현상이 없이 휴대용단말에서 재생이 가능함을 알 수 있었다. 또한 원래의 실수형 연산과의 디코딩 시 발생하는 오차는 재생 시 그 품질이 수용할 수 있는 범위임을 알 수 있다.

구현된 MPEG-4 AAC 재생기는 부동 소수점 연산 프로세서가 없는 인터넷 정보 가전 기기의 내장형 시스템에서도 적용될 수 있다.

참 고 문 헌

[1] http://www.arm.com

[2] ISO/IEC JTC1/SC29/WG11, "Coding of moving pictures and audio-MPEG-2 Advanced Audio Coding AAC," ISO/IEC 13818-7 International Standard, 1997.

[3] K.R. Rao and J.J. Hwang. Techniques and Standards for Digital Image/Video/Audio Coding. Prentice Hall, 1996.

[4] J. Cartinhour, Digital Signal Processing : An overview of basic principles, Prentice Hall, Upper Saddle River, N. J., 2000.

[5] Yoshikazu Yokotani and Soontorn Oraintara, "Lossless Audio compression using Integer Modified Discrete Cosine Transform," 2003 International Symposium on Intelligent Signal Processing and Communication System(ISPACS 2003), pp. 120- 126, December 7-10, 2003.

[6] http://www.hydrogenaudio.org/

[7] Y.C. Hou and S.D.You, "Implementation of IMDCT for MPEG2/4 AAC on 16 bit Fixed-point Digital Signal Processors," IEEE Asia-Pacific Conference on Circuit and Systems, December 6-9, 2004.