Implementation of a 16-Bit Fixed-Point MPEG-2/4 AAC Decoder for Mobile Audio Applications

Byoung Eul Kim* *Associate Member*, Sun-Young Hwang* *Regular Member*

ABSTRACT

An MPEG-2/4 AAC decoder on 16-bit fixed-point processor is presented in this paper. To meet audio quality criteria, despite small word length, special design methods for 16-bit fixed-point AAC decoder were devised. This paper presents particular algorithms for 16-bit AAC decoding. We have implemented an efficient AAC decoder using the proposed algorithms. Audio contents can be replayed in the decoder without quality degradation.

Key Words : AAC, Audio coding, MPEG audio, 16-bit fixed-point, IMDCT

Ⅰ. 서 론

During last two decades, digital audio has been widely accepted as a common medium for listening to the music because digital audio has unprecedented high fidelity and wide dynamic range. In particular, the market of mobile digital audio without mechanical parts has been growing rapidly since commercial success of MPEG-1/2 Layer III(MP3) standards. One of the reasons of the success of mobile digital audio products is the development of audio compression technology, which makes it possible to provide high quality audio at low bit rate such as compact disc. MPEG-2/4 AAC(Advanced Audio Coding) is one of the most advanced technologies among existing audio coding technologies in terms of compression rate and audio quality, and this is why AAC is broadly accepted in digital audio market[1].

Digital Signal Processor(or "DSP processor"). The DSP processors can be classified into fixed-point DSP processor and floating-point DSP processor. While a floating-point DSP processor has advantages of precision and range over fixed-point DSP processor, it tends to be more expensive and more power consuming than fixed-point DSP processor. Hence mobile devices are mostly designed on fixed-point DSP processor. Most commercialized DSP processors have 16-bit, 24-bit, or 32-bit word length. A DSP processor with a shorter word length usually has a lower arithmetic accuracy with less cost.

It has been reported that AAC decoder can fully meet audio quality criteria by employing the word length longer than 20 bits[2]. While AAC decoder on 16-bit fixed-point DSP processor such as TI's TMS320C54x is a cost-effective solution, due to limited dynamic range and scaling down of data, it is difficult to meet audio quality criteria when implementing the AAC decoder with 16-bit processors[3]. Most mobile audio devices support 16-bit resolution audio outputs. A critical limitation of 16-bit fixed-point AAC decoding is to get 16-bit audio resolution on 16-bit processing. It is difficult to obtain 16-bit audio accuracy on 16-bit processing since additional bits

※ This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) supported program supervised by the IITA (Institute for Information Technology Advancement) (IITA-2008-(C1090-0801-0012))

* The authors are with Department of Electronic Engineering, Sogang University, Seoul, Korea.

논문번호 : KICS2007-12-583, 접수일자 : 2007년 12월 22일, 최종논문접수일자 : 2008년 3월 13일

are needed to avoid overflows and to maintain data accuracy[4]. However it is possible to get satisfactory audio quality on 16-bit AAC decoder using particular decoding process for minimum accuracy loss. In this paper, we propose the algorithms for 16-bit fixed-point AAC decoding. We implemented an efficient 16-bit fixed-point AAC decoder using the proposed algorithms. The decoder can replay audio contents without quality degradation.

This paper is organized as follows. Section II and III describe the optimal number of guard bits of AAC spectral coefficients for 16-bit spectral processing and a novel fixed-point scaling algorithm of IMDCT, respectively. Section IV presents a performance test of the 16-bit fixed-point AAC decoder implemented using the proposed algorithms. Finally, results of implementation of the 16-bit fixed-point AAC decoder are discussed in section V.

Ⅱ. Optimal Number of Guard Bits for Spectral Processing

The MPEG-2/4 standard[5] defines several tools for different audio coding algorithms to establish optimal coding efficiency for a broad range of applications. Figure 1 shows an overview of the MPEG-4 AAC decoding process[5]. In Figure 1, quantized spectral data is noiselessly decoded from de-multiplexed AAC bitstream. Each coefficient is inversely quantized and re-scaled. Then spectral coefficients are completely recovered from spectral data by spectral processing which is marked in figure 1, before they are fed into IMDCT. IMDCT transforms spectral coefficients into time-domain samples. Finally, transformed time-domain samples are windowed, overlapped, and added for proper reconstruction. By this sequence, AAC bitstream is decoded into PCM samples.

In pre-spectral processing, spectral coefficients have to be inversely quantized and re-scaled after noiseless decoding. Pre-spectral processing needs dynamic range from $[0..9\times10^{16}]$ on AAC specifications[4]. However, this range cannot be covered by 16-bit processing. Thus, spectral coefficients should be limited to 16-bits by additional downscaling process. Extra gain factor G_e is defined for 16-bit limitation of spectral coefficients as in (1)[6].

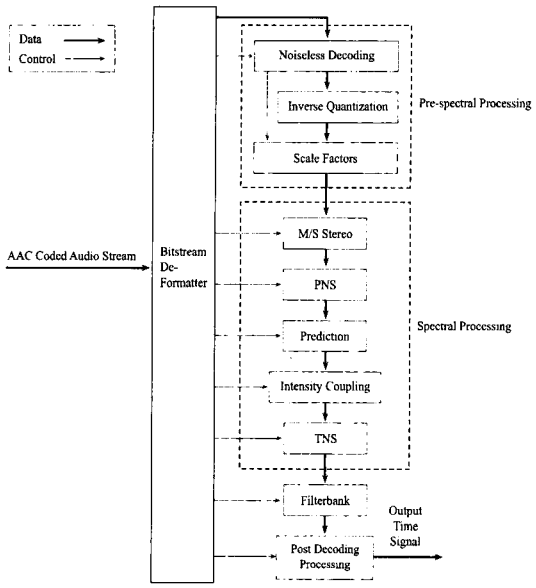


Fig. 1 MPEG-4 AAC decoder block diagram[5]

$$x_{\text{spec}}' = \left\lfloor \frac{1}{G_e} x_{\text{spec}} \right\rfloor \quad (1)$$

To prevent overflow during the 16-bit spectral processing, spectral coefficients limited by G_e have to be included proper guard bits into the left of the most significant bits. We select optimal number of guard bits of AAC spectral coefficients for 16-bit spectral processing on the remaining of this chapter. Assuming that the maximum absolute value of input spectral coefficients into spectral processing is x_{in} and the maximum absolute value of output spectral coefficients from spectral processing is x_{out}, bit growth of spectral coefficients can be obtained by (2).

$$\text{BitGrowth}_{\text{spec}} = \lceil \log_2 x_{\text{out}} \rceil - \lceil \log_2 x_{\text{in}} \rceil \quad (2)$$

Table 1 shows experimental results of frequency distribution of bit growth during the spec-

Table 1. Frequency distribution of bit growth during the spectral processing(1 frame = 1024 spectral coefficients)

AAC Files	Total number of frames in AAC file	Number of frames(Percentage)		
		0-bit growth	1-bit growth	Over 1-bit growth
1	6,909,301	4,435,771 (64.2%)	2,473,530 (35.8%)	0
2	25,953	19,065 (73.5%)	6,888 (26.5%)	0
3	292,002	159,654 (54.7%)	132,348 (45.3%)	0
4	245,385	142,803 (58.2%)	102,582 (41.8%)	0
5	191,142	105,288 (55.1%)	85,854 (44.9%)	0
6	1,242,300	900,606 (72.5%)	341,694 (27.5%)	0
7	1,027,419	593,352 (57.8%)	434,067 (42.2%)	0
8	385,482	203,811 (52.9%)	181,671 (47.1%)	0
9	542,184	293,109 (54.1%)	249,075 (45.9%)	0
10	1,014,012	680,190 (67.1%)	333,822 (32.9%)	0
11	224,967	128,904 (57.3%)	96,063 (42.7%)	0
12	92,619	52,152 (56.3%)	40,467 (43.7%)	0
13	61,008	33,210 (54.4%)	27,798 (45.6%)	0
14	92,127	50,184 (54.5%)	41,943 (45.5%)	0
15	2,879,799	1,980,054 (68.8%)	899,745 (31.2%)	0
16	465,678	272,814 (58.6%)	192,864 (41.4%)	0
17	411,435	222,876 (54.2%)	188,559 (45.8%)	0
18	2,656,308	1,939,710 (73.0%)	716,598 (27.0%)	0
19	1,039,965	699,255 (67.2%)	340,710 (32.8%)	0
20	7,569,174	4,753,950 (62.8%)	2,815,224 (37.2%)	0

tral processing using 20 different AAC audio files. We obtained values of bit growth using (2). From table 1, 0-bit growth accounts for 52.9% ~ 73.4%, and 1-bit growth accounts for 26.5% ~ 47.1%. Over 1-bit growth has not been observed. Results of table 1 reveal that over 1-bit growth of spectral coefficients never occurs during the spectral processing. According to this result, **two guard bits,** which are included one margin bit, should be inserted into spectral coefficients during scaling process by G_e, in order to prevent overflow prior to IMDCT process. The proposed extra gain equation for selecting G_e including two guard bits is given as in (3), where x_{max} is the maximum absolute value of input spectral coefficients. By (3), spectral coefficients can be completely recovered during the spectral processing without overflow.

$$G_e = \begin{cases} 2^8, & x_{\max} \in [2^{21}, 2^{32}] \\ 2^7, & x_{\max} \in [2^{20}, 2^{21}) \\ 2^6, & x_{\max} \in [2^{19}, 2^{20}) \\ 2^5, & x_{\max} \in [2^{18}, 2^{19}) \\ 2^4, & x_{\max} \in [2^{17}, 2^{18}) \\ 2^3, & x_{\max} \in [2^{16}, 2^{17}) \\ 2^2, & x_{\max} \in [2^{15}, 2^{16}) \\ 2^1, & x_{\max} \in [2^{14}, 2^{15}) \\ 2^0, & x_{\max} \in [2^0, 2^{14}) \end{cases} \quad (3)$$

Ⅲ. Scaling Algorithm of IMDCT for 16-Bit Fixed-Point AAC Decoding

Spectral coefficients scaled by (3) and (1) are decoded into 16-bit PCM data by spectral processing and filter bank processing. Most errors are caused by IMDCT in this process, so it is important to implement IMDCT with a high precision. The IMDCT equation is shown by (4) [7][8].

$$x_i[n] = \frac{2}{N} \sum_{k=0}^{(N/2)-1} X_i[k] \cos\left[\frac{2\pi}{N}(n+n_0)(k+\frac{1}{2})\right] \quad (4)$$

In (4), n is sample index($0 \le n < N$), i is window index, k is spectral coefficient index, N is number of total samples which is 2048 or 256 according to the window type, and n_0 is $(N / 2 + 1) / 2$, respectively. IMDCT algorithm based on FFT is the most suitable for implementation on DSP processors[3]. Since N-point IMDCT can be expressed in terms of an IFFT computation of length $N / 4$, performing the IMDCT with N = 2,048 or N = 256 is similar to performing the FFT with N = 512 or N = 64[9]. Figure 3 shows a flow of IMDCT calculation based on FFT[9].

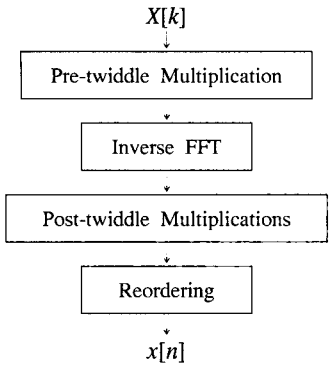


Fig. 2 Flow of IMDCT calculation based on FFT[9]

Radix-8 FFT algorithm is the most suitable choice to implement fast IMDCT for AAC decoders using 16-bit fixed-point DSP processors[3]. In case of radix-8 IFFT, the calculation given in (5) is performed at each stage. In (5), $X[k]$ is input samples, $X'[k]$ is output samples, and W_N^{nk} is twiddle factor($W_N = e^{(-j2\pi/N)}$), respectively.

$$\begin{bmatrix} X'(k) \\ X'(k+\frac{N}{8}) \\ X'(k+\frac{2N}{8}) \\ X'(k+\frac{3N}{8}) \\ X'(k+\frac{4N}{8}) \\ X'(k+\frac{5N}{8}) \\ X'(k+\frac{6N}{8}) \\ X'(k+\frac{7N}{8}) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{1+j}{\sqrt{2}} & j & \frac{-1+j}{\sqrt{2}} & -1 & \frac{-1-j}{\sqrt{2}} & -j & \frac{1-j}{\sqrt{2}} \\ 1 & j & -1 & -j & 1 & j & -1 & -j \\ 1 & \frac{-1+j}{\sqrt{2}} & -j & \frac{1+j}{\sqrt{2}} & -1 & \frac{1-j}{\sqrt{2}} & j & \frac{-1-j}{\sqrt{2}} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & \frac{-1-j}{\sqrt{2}} & j & \frac{1-j}{\sqrt{2}} & -1 & \frac{1+j}{\sqrt{2}} & -j & \frac{-1+j}{\sqrt{2}} \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & \frac{1-j}{\sqrt{2}} & -j & \frac{-1-j}{\sqrt{2}} & -1 & \frac{-1+j}{\sqrt{2}} & j & \frac{1+j}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} X(k) \\ X(k+\frac{N}{8})W_N^{n} \\ X(k+\frac{2N}{8})W_N^{2n} \\ X(k+\frac{3N}{8})W_N^{3n} \\ X(k+\frac{4N}{8})W_N^{4n} \\ X(k+\frac{5N}{8})W_N^{5n} \\ X(k+\frac{6N}{8})W_N^{6n} \\ X(k+\frac{7N}{8})W_N^{7n} \end{bmatrix} \quad (5)$$

To prevent addition overflow of fixed-point IFFT, proper guard bits are required. According to (6), eight add operations are performed per each sample on radix-8 IFFT and maximum magnitude growth factor is 10.899: input samples of each IFFT stage need four guard bits. However, four guard bits decrease precision of IFFT significantly since the number of effective bits is changed from 16 to 12. In this paper, we propose a novel scaling algorithm with minimum guard bits for fixed-point radix-8 IFFT. Figure 3 shows signal flow of radix-8 butterfly using the proposed scaling algorithm. Suggested flow includes three scaling stages to prevent addition overflows at each sub-stage. Scaling operation is performed at the end of each sub-stage. The proposed scaling algorithm is derived from radix-2/4/8 algorithm[10]. Cascade decomposition of radix-8 butterfly in [10] is originally derived for efficient VLSI implement. However, we prove that cascade decomposition of radix-8 butterfly can be used for efficient scaling of radix-8 IFFT calculation with minimum guard bits. Table 2 shows maximum number of add operations, maximum growth factor, and required guard bits at each sub-stage of the proposed IFFT butterfly(figure 5). In table 2, input samples of sub-stage 1 need one guard bit since one add operation is performed per each sample. Then, input samples of sub-stage 2 and sub-stage 3 need two

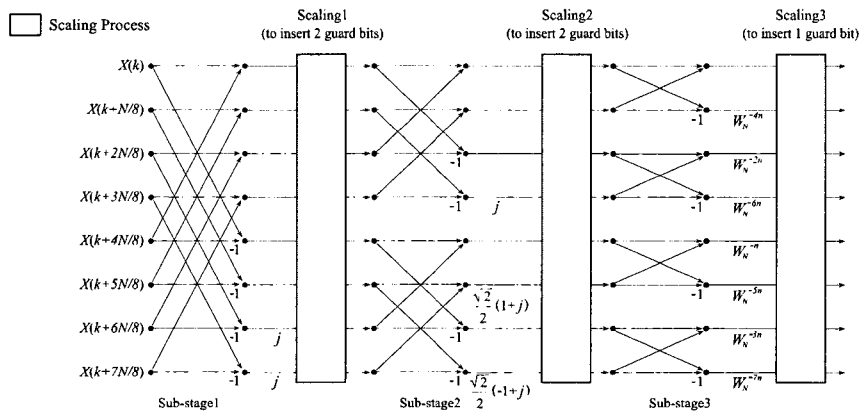


Fig. 3 Proposed signal flow structure of radix-8 butterfly. This structure includes three scaling stages to prevent addition overflows at each sub-stage

Table 2. Maximum number of guard bits at each sub-stage using proposed scaling algorithm

	Max. Number of Add Operations	Max. Growth Factor	Required Guard Bits
Sub-stage 1	1	2	1 bit
Sub-stage 2	2	2.828	2 bits
Sub-stage 3	2	2.828	2 bits

guard bits since at most two add operations are performed. This result reveals that guard bits of radix-8 IFFT butterfly can be decreased significantly using the proposed scaling algorithm.

Overflows can also occur at pre- and post-twiddle multiplication of IMDCT based on IFFT. The process of pre- and post-twiddle multiplication of IMDCT is given in (6) and (7) [9].

$$F[k]=\frac{2}{N}\left(X\left[\frac{N}{2}-1-2k\right]+jX[2k]\right)\cdot\exp\left(\frac{2\pi(8k+1)}{8N}\right) \quad (6)$$

$$g[n]=f[n]\cdot\exp\left(\frac{2\pi(8n+1)}{N}\right) \quad (7)$$

In (6) and (7), each input sample needs one guard bit since one add operation is performed per each sample. One guard bit is added to the input samples of both pre-twiddle multiplication by (3) and post-twiddle multiplication by the last stage of the proposed IFFT butterfly. Since initial input stage of the proposed IFFT has no scaling process, additional scaling process should be inserted after pre-twiddle multiplication to obtain one guard bit. Figure 4 shows proposed overall process of IMDCT calculation optimized for 16-bit fixed-point AAC decoding.

Ⅳ. Performance Test and Results

Low-cost 16-bit fixed-point AAC decoder has been implemented using proposed algorithms. A 16-bit fixed-point AAC decoder is designed to meet limited-accuracy criterion, not full-accuracy criterion[3][6][11]. In conformance standard[12], decoder with an accuracy level of '*k*-bit' is defined.

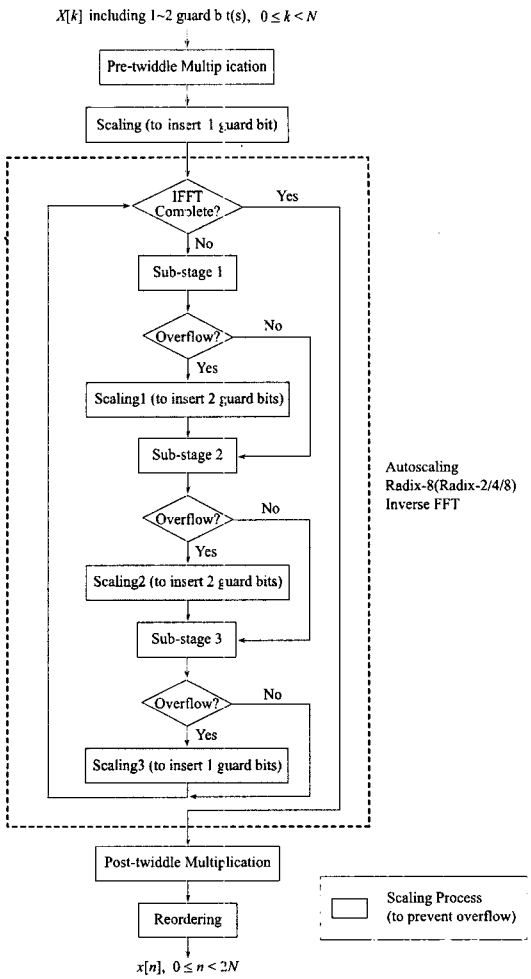


Fig. 4 Propose overall process of IMDCT calculation with minimum guard bits for 16-bit fixed-point AAC decoding.

The normalized root mean square error(RMSE) of the decoder output should be less than $2^{-(k-1)}/\sqrt{12}$ and the maximum absolute error(MAE) should be less than $2^{-(k-2)}$. According to [12], the 'limited-accuracy' MP3 decoder defined in [11] is equal to '12-bit' accuracy decoder in terms of RMSE accuracy. In this case, RMSE of the decoder should be less than 1.41×10^{4} and MAE should not exceed 9.77×10^{4}.

We utilize T−1 ~ T−8 series of audio sources and implement AAC decoders using two kind of decoding algorithms for performance test. Algorithms shown below are included. (i) the proposed algorithms(AGO1), (ii) the algorithm using

Table 3. Experimental results of performance test

Series	RMSE		MAE		Remark
	AGO1	AGO2	AGO1	AGO2	
Standard	$< \mathbf{1.41\times10^{-4}}$	$< 1.41\times10^{-4}$	$< \mathbf{9.77\times10^{-4}}$	$< 9.77\times10^{-4}$	Conformance Standard[12]
T-1	$\mathbf{3.38\times10^{-5}}$	6.45×10^{-4}	$\mathbf{7.90\times10^{-4}}$	3.42×10^{-2}	Symphony
T-2	$\mathbf{8.80\times10^{-6}}$	6.97×10^{-4}	$\mathbf{6.86\times10^{-4}}$	4.91×10^{-2}	Opera
T-3	$\mathbf{1.25\times10^{-5}}$	4.18×10^{-4}	$\mathbf{7.17\times10^{-4}}$	4.88×10^{-2}	Orchestral Work
T-4	$\mathbf{2.96\times10^{-5}}$	1.40×10^{-3}	$\mathbf{1.03\times10^{-3}}$	6.68×10^{-2}	Popular Music
T-5	$\mathbf{1.97\times10^{-5}}$	3.35×10^{-4}	$\mathbf{9.30\times10^{-4}}$	2.66×10^{-2}	Piano
T-6	$\mathbf{1.65\times10^{-3}}$	4.28×10^{-2}	$\mathbf{1.40\times10^{-3}}$	5.69×10^{-2}	Percussion
T-7	$\mathbf{1.10\times10^{-5}}$	2.77×10^{-4}	$\mathbf{2.90\times10^{-4}}$	2.23×10^{-2}	Vocal
T-8	$\mathbf{7.32\times10^{-5}}$	1.56×10^{-3}	$\mathbf{9.89\times10^{-4}}$	3.48×10^{-2}	Sweep Signal(20Hz ~ 10kHz)

previous 16-bit limitation method[3] and IMDCT based on conventional radix-2 IFFT(AGO2). Radix-2 IFFT in AGO2 uses the scaling strategy to divide the numbers by two after addition[13]. The reference series are outputs of 32-bit floating-point decoder implemented using the same decoding method at each algorithm. Results shown in the table 3 reveal that the implemented 16-bit fixed-point AAC decoder using proposed algorithms (AGO1) meets RMSE and MAE criterion of k = 12 accuracy level. The decoder provides sufficient audio quality to replay audio contents for mobile applications.

V. Conclusion

In this paper, we propose optimal number of guard bits of AAC spectral coefficients for spectral processing and a novel fixed-point scaling algorithm of IMDCT based on radix-8 IFFT for 16-bit AAC decoding, respectively. Using proposed algorithms, we implemented an efficient 16-bit fixed-point AAC decoder. Results of performance test show that audio contents can be replayed in the decoder without quality degradation.

References

[1] M. Watson and P. Buettner, "Design and Implementation of AAC Decoders", IEEE Trans. Consumer Electronics, Vol. 46, No. 3, pp. 819-824, Aug. 2000.

[2] K. Bang, J. Kim, N. Jeong, Y. Park, and D. Youn, "Design Optimization of MPEG-2 AAC Decoder", IEEE Trans. Consumer Electronics, Vol. 47, No. 4, pp. 895-903, Nov. 2001.

[3] S. You and Y. Hou, "Implementation of IMDCT for MPEG2/4 AAC on 16-bit Fixed-Point Digital Signal Processors", Proc. 2004 IEEE Asia-Pacific Conf. on Circuits and Systems, Vol. 2, pp. 813-816, Dec. 2004.

[4] C. Burgel, R. Bartholomaus, W. Fiesel, J. Hilpert, A. Hoelzer, and K. Linzmeier, "Beyond CD-Quality: Acvanced Audio Coding(AAC) for High Resolution Audio with 24-Bit Resolution and 96-kHz Sampling Frequency", AES 111th Convention, 2001.

[6] H. Wang, W. Xu, X. Dong, C. Li, and W. Yu, "Implementation of MPEG-2 AAC on 16-bit Fixed-Point DSP", Proc. IEEE Asia-Pacific Conf. on Circuits and Systems 2006, pp. 1903-1906, Dec. 2006.

[5] ISO/IEC 14496-3 : Information Technology - Coding of Audio-Visual Objects - Part 3: Audio, 1999.

[7] J. Princen and A. Bradley, "Analysis / Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 34, No 5, pp. 1153-1161, Oct. 1986.

[8] J. Princen, A. Johnson, and A. Bradley, "Subband / Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation", IEEE International Conf. on Acoustics, Speech, and Signal Processing '87, Vol. 12, pp. 2161-2164, Apr. 1987.

[9] P. Duhamel, Y. Mahieux, and J. Petit, "A Fast Algorithm for the Implementation of Filter Banks Based on 'Time Domain Aliasing Cancellation'", IEEE International Conf. on Acoustics, Speech, and Signal Processing '91, Vol. 3, pp.2209-2212, May 1991.

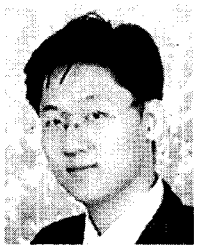
[10] L. Jia, Y. Gao, and H. Tenhunen, "A New VLSI-Oriented FFT Algorithm and Implementation", Proc. 11th Annual IEEE International ASIC Conf., pp. 337-341, Sep. 1998.

[11] ISO/IEC 11172-3, Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5Mbit/s - Part 4: Compliance Testing, 1993.

[12] ISO/IEC 13818-4 Amd. 1/Cor. 1, Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Part 4: Conformance Testing, Amendment I: Advanced Audio Coding(AAC) Conformance Testing, Technical Corrigendum 1, 2003.

[11] A. V. Oppenhiem and R. W. Schafer, "*Discrete-Time Signal Processing*", Ch. 9, Prentice-Hall, Englewood Cliffs, N. J, 1989.

Byoung Eul Kim Associate Member



He received the B.S. degree in electronic engineering from Sogang University, Seoul, Korea, in 2006. He is currently working towards the M.S. degree in electrical engineering at Sogang University. His current research interests include digital video/audio compression, real-time DSP systems, and CAD systems.

Sun-Young Hwang Regular Member

He received the B.S. degree in electronic engineering from Seoul National University, Seoul, Korea, in 1976, the M.S. degree from Korea Advanced Institute of Science in 1978, and the Ph.D. degree in electrical engineering from Stanford University, California, U.S.A., in 1986. Since 1986, he has been with the Center for Integrated Systems at Stanford University, working on design of a high-level synthesis and simulation system. In 1986 and 1987, he held a consulting position at Palo Alto Research Center of Fairchild Semiconductor Corporation. In 1989, he joined the Department of Electronic Engineering at Sogang University, where he is now a professor. His current research interests include hardware/software co-design, DSP/VLSI systems design, and embedded systems design.