FAST TIME-FREQUENCY TRANSFORM ALGORITHMS AND THEIR APPLICATIONS TO REAL-TIME SOFTWARE IMPLEMENTATION OF AC-3 AUDIO CODEC

Yu-Chi Chen, Chien-Wu Tsai, and Ja-Ling Wu
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abtract—**AC-3 audio coding technology is a kind of Perceptual Audio Coder (PAC) developed by the Dolby Company. Up to 5 full-bandwidth channels and one subwoofer channel (cutoff at 120Hz) are available in AC-3 to provide multi-channel, low bit rate, and high perceptual quality of audio. This explains why AC-3 has become the audio standard of many international standards. In this paper, we focus on the real-time software implementation issues of AC-3. Two fast algorithms of the time-frequency transform, one for memory economization and the other is for time domain subsampling, are presented. Meanwhile, we will state the current performance status of our AC-3 decoder.**

I. INTRODUCTION

Along with the rapidly increasing computing power of PC, the diffusion of multimedia, so as the upspring of network and the internet, multimedia communication has played an important role in the so-called 3C market today. Moreover, since audio signals demand much lower bandwidth and cost than video signals, audio signal processing techniques have become fundamental and prerequisite.

The Dolby Company developed AC-3 audio coding scheme. Dolby Digital (the audio standard used in film industry, DVD, multimedia, HDTV), Dolby Surround Digital (the audio standard used in Home Theater System (HTS)), and Dolby Net (the audio standard used in the internet environment), all refer to the same kernel — the AC-3 audio coding technology. To feed these quite different demands of all these different areas, the target bit-rates of AC-3 ranges from 32k to 640k bps.

AC-3 is a perceptual audio coder (PAC), that is, AC-3 uses human psycho-acoustic features to "mask" the inaudible audio signals, so the bits are saved for representing really important signals. Up to 5 full-bandwidth channels and one subwoofer channel (cutoff at 120Hz) can be contained in an AC-3 bitstream (this is the so-called 5.1 channels). The time-frequency transform used in AC-3 is the Analysis/Synthesis Filter Bank with Time Domain Aliasing Cancellation[6]. A parametric bit allocation process is applied in AC-3, so flexibility and efficiency can be achieved at the same time. With sophisticated psycho-acoustic model and the decorrelation between neighboring frequencies and different channels, AC-3 provides low-bandwidth but high quality perceptual audio sound.

According to our implementation, we have proved that it is possible to decode AC-3 bitstreams by PC under the real-time constraint with software only. Furthermore, with the proposed time domain subsampling algorithm, audio quality and system execution time becomes a tradeoff. Sacrificing audio quality would increase decoding efficiency, so combination with other applications becomes possible, such as the integration with DVD video decoding software.

In this paper, the AC-3 audio coding scheme will be described in detail. In section II, we will address the psycho-acoustic model and the bit allocation process in AC-3. In section III, the Analysis/Synthesis Filter Bank is investigated; fast algorithms are also discussed in this section. All the codec process details are presented in section IV. Section V shows the system performance of our AC-3 player, and section VI gives the conclusion.

II. THE PSYCHO-ACOUSTIC MODEL USED IN AC-3

Digital audio signals are usually stored in the Pulse Coded Modulation (PCM) format (time domain). Traditionally, before these signals are further managed by a PAC, they are usually transformed to the Fourier frequency domain since the Human Auditory System (HAS) is understood in the Fourier transform frequency domain. These transformed data are then passed through a psycho-acoustic model to determine their importance to the HAS, and so their Signal to Noise Ratio (SNR).

AC-3 designed a parametric psycho-acoustic model such that flexibility, efficiency, and low bandwidth requirement can all be achieved in the bit allocation process. Fig. 1 is the block diagram of the bit allocation process used in AC-3. Six steps are contained in this process. All the operations needed are only addition, maximum, shift, and table-look-up in all these six steps.

Steps 1 and 2. Power Spectrum Mapping and Frequency Banding:

AC-3 uses Modified Discrete Cosine Transform (MDCT) as its time-frequency transform. Moreover, it divides the transformed coefficients to exponent part and mantissa part. Also, it is the power of critical band that is really considered in the bit allocation process. As a result,

Contributed Paper
Manuscript received March 16, 1998

two things must be done before the bit allocation process really starts. One is the mapping from exponents to the power spectrum, so the computation of the power spectrum can be simplified. The other is the critical band mapping from MDCT frequency domain to Fourier frequency domain, so the researches of the psycho-acoustics on Fourier frequency domain can be used.

Since the meaning of "frequency" in MDCT domain is very similar to that in Fourier transform domain, the original critical banding structure (BARK) has to be modified to work on the MDCT frequency domain. The banding structure of critical bands in AC-3 is called the AC-3 band.

The exponent of one coefficient can be seen as the approximated power of that coefficient. The power of AC-3 band is the summation of all the power of coefficients in that band. Since the power of coefficients is in logarithmic domain (i.e. in dB), this calculation of summations needs logarithmic additions. AC-3 uses a table-look-up method for speeding up the calculation of this power integration.

Step 3. Spreading Function:

According to the different masking behavior of AC-3 bands, AC-3 splits its banding structure to three parts. Since the spreading function of AC-3 band is skew, AC-3 only takes into account the upward spectral spreading function. The upward spectral spreading function is approximated with two line segments.

AC-3 also considers the possible audible quantization noise because of insufficient usage of bits. AC-3 again divides the AC-3 band to three parts. Along with calculating the masking curve of spreading function, the power relationship between the calculating band and the next band is considered. Under some circumstances, the masking curve of spreading function is lowered to prevent audible quantization distortion.

Step 4. Hearing Threshold:

The Absolute Threshold in Quiet, also called Just Noticeable Distortion (JND), is obtained by experiments. Fig. 2 is the JND curve of AC-3 band. The meaning of JND is the minimum power required to excite the human basilar membrane so the sound in that frequency can be heard. After the masking curve of spreading function is constructed, it is compared to JND, and the resulted masking curve is the maximum of the two in each band.

Step 5. Masking comparison:

AC-3 allows coder to change the masking curve so that different banding structures and different psycho-acoustic models can also be used. If a different psycho-acoustic model is used, both masking curves should be calculated. The AC-3 psycho-acoustic model's parameters and the differences between the two masking curves allows decoder to reconstruct the alternative masking curve, i.e., the psycho-acoustic model used by the encoder.

The coder compares the power of AC-3 band to a threshold (decided by the encoder) so that AC-3 band with higher pressure level has higher SNR. After that, the final masking curve is determined. A coarse SNR offset and a fine SNR offset are used in coder to increase the SNR since the masking curve is somewhat conservative. The coarse SNR offset adjusts the masking curve in 6 dB (or equivalently 1 bit) increments and is the same for every channel; the fine SNR offset adjusts in 3/8 dB increments and each channel has its own so that different channels can have different SNRs.

The difference between a coefficient and the masking curve represents the information that the coefficient carries. With a table-look-up, we can quantize the information and determine the SNR for that coefficient. Encoder then uses an iterative process to dynamically change the coarse and fine SNR offsets so that the desired bit rate can be maintained.

Step 6. Quantizer selection:

Step 6 is the process that really quantizes the mantissas of coefficients. No quantization distortion due to insufficient bits should be heard. The SNR of coefficients ranges from 5/3 to 16 bits and is called bit allocation pointer (bap) in AC-3. Mantissas with different baps have different packing methods. The mantissa packing technique is described in section IV.

Both AC-3's encoder and decoder would calculate the six steps for the bit allocation information; encoder uses this information for packing mantissas and decoder for unpacking. In AC-3, only the parameters of the coarse offset, the fine offset for each channel, the spreading function (two line segment parameters), delta bit allocation, and the threshold are transmitted to the decoder, so bits are significantly reduced. One thing should be noted that once the coarse and fine SNR offsets are changed, the SNR of the whole channel is increased, not a specific band or mantissa. In our implementation, a "vertical" bit allocation is used to save memory and to speed up the execution.

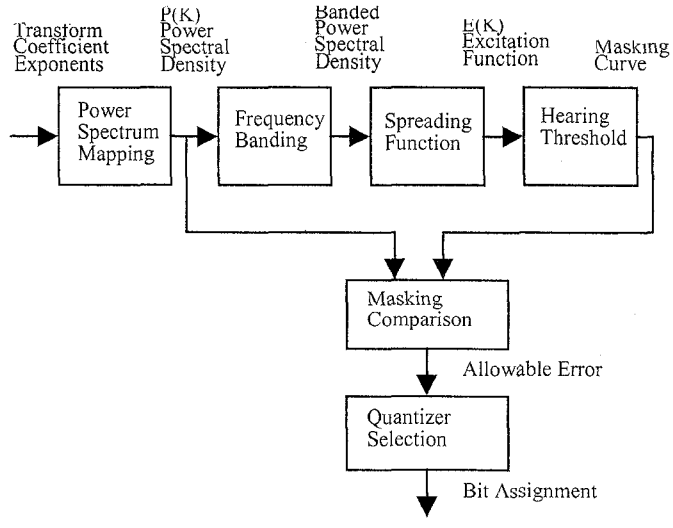

Fig. 1 The block diagram of AC-3 bit allocation process

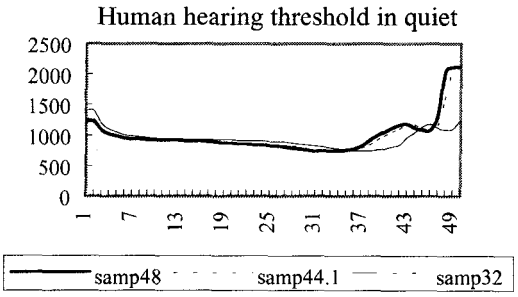

Fig. 2 The JND under AC-3 banding structure

III. THE SUBBAND FILTER BANKS IN AC-3

In PAC encoders, a "block" of audio signals must be windowed, and passed through an analysis filter. With the filtered data, the bit allocation process can understand the frequency behavior of this audio block and decide to what detail should one coefficient be coded. In decoder side, the frequency data would be reversely windowed, and passed through a synthesis filter. Then, the final overlap-and-add process would accomplish the reconstruction of the time-domain data.

In this analysis/synthesis filter bank structure, error occurs because of the filter is not perfect; sidelobes always introduce alias. Error also introduced because of quantization due to bit allocation. If quantization error is ignored, Perfect Reconstruction (PR) does really exist. Frequency Domain Aliasing Cancellation (FDAC) and Time Domain Aliasing Cancellation (TDAC) filter bank structures are the most commonly used. Since AC-3 use TDAC, FDAC will not be discussed in this paper.

A.. Time Domain Aliasing Cancellation

Since alias is inevitably produced in analysis/synthesis filter bank, and the cost to attenuate alias is very high, TDAC tries another way: to compensate the alias. That is, alias is introduced after the analysis filter, but the produced alias is eliminated by the overlap-and-add process after the synthesis filter. This alias elimination can be done because that the alias of the previous block and the current block are of different sign.

Eqn. 1 and Eqn. 2 are the formulas of forward and inverse transforms of the TDAC, respectively. K is the block size, $x(n)$ is the time domain sequence, $X_k(m)$ is the frequency data of band k in time block m, $\cos(m\pi/2)$ and $\sin(m\pi/2)$ are switches deciding which data path to go according to the block number, $\cos(\omega_k(n+n_0))$ and $\sin(\omega_k(n+n_0))$ are the analysis and synthesis filters, n_0 is the phase delay, $h(n)$ and $f(n)$ are the windows with size P, and symbols with '^' are the transmitted or reconstructed data.

$$X_k(m)=\cos(\frac{m\pi}{2})\sum_{n=-\infty}^{\infty}x(n)\cdot h(P-1+\frac{mK}{2}-n)\cdot\cos(\omega_k(n+n_0)) + \sin(\frac{m\pi}{2})\sum_{n=-\infty}^{\infty}x(n)\cdot h(P-1+\frac{mK}{2}-n)\cdot\sin(\omega_k(n+n_0)) \quad (1)$$

$$\hat{x}(n)=\frac{1}{K}\sum_{k=0}^{K-1}\{\cos(\omega_k(n+n_0))\sum_{m=-\infty}^{\infty}X_k(m)\cdot\cos(\frac{m\pi}{2})f(n-\frac{mK}{2}) + \sin(\omega_k(n+n_0))\sum_{m=-\infty}^{\infty}X_k(m)\sin(\frac{m\pi}{2})f(n-\frac{mK}{2})\} \quad (2)$$

Let the window size $P=K$; if $P<K$, and $h(n)=f(n)=0$ for $K>n>=P$. So, the time domain sequence $x(n)$ are divided into size K blocks with time index m, $x_m(n)$. Besides, the neighboring blocks are overlapped with $K/2$ components, i.e., $x_{m-1}(n+K/2) = x_m(n)$. The blocked data $x_m(n)$ is windowed and passed through the bandpass filters $\cos(\omega_k(n+n_0))$ and $\sin(\omega_k(n+n_0))$, respectively.

Two types of TDAC are designed: ω_k equals to $(2\pi k/K)$ is an evenly stacked TDAC; ω_k equals to $(2\pi\ (k+1/2)/K)$ is an oddly stacked TDAC. That's why there are only $M=K/2$ bands are unique, and the windowed, filtered data can be critical-sampled by M. Besides, even blocks will choose the upper path since $|\cos(m\pi/2)|=1$ and $|\sin(m\pi/2)|=0$ while odd blocks go the other way.

The synthesis filter bank does the reverse work. Since AC-3 uses the oddly stacked TDAC, we will focus on the situation that ω_k equals to $(2\pi\ (k+1/2)/K)$. Since the window size is K, Eqn. 1 can be rewritten to Eqn. 3 with the variable replacement $-r=mK/2-n$ and some deductions. Since the first half part of the current block overlaps with the second half part of the previous block, the reconstructed time samples $\hat{x}_m(n)$ are got by adding the overlapping parts of both blocks. This is called overlap-and-add. This is represented in Eqn. 4, by substituting Eqn. 1 to Eqn. 2 and using the same trick in Eqn. 3.

$$X_k(m)=\cos(\pi mk)\sum_{r=0}^{K-1}x(r)\cdot h(K-1-r)\cdot\cos(\frac{2\pi}{K}(k+1/2)(r+n_0)) \quad (3)$$

$$\hat{x}(r)=f(r+\frac{K}{2})y_{m_0-1}(r+\frac{K}{2})+f(r)y_{m_0}(r) \quad (4-1)$$

$$y_m(r)=\frac{1}{K}\sum_{k=0}^{K-1}\sum_{p=0}^{K-1}X_m(p)h(K-1-p)\cdot\cos(\frac{2\pi}{K}(k+1/2)(p+n_0))\cdot \cos(\frac{2\pi}{K}(k+1/2)(r+n_0)) \quad (4-2)$$

Let the phase delay $n_0 = K/4 + 1/2$ and after some further deductions, we can see clearly how the alias cancellation is achieved (Eqn. 5). In Eqn. 5, x_{m0} are the original time signals of current block, x_{m0-1} are alias introduced by the previous block. If we want to achieve PR, $\hat{x}_{m0}(r)$ should be equals to $x_{m0}(r)$. Some constraints are put on the analysis and synthesis windows. First, the analysis and synthesis windows must be equivalent, $f(r) = h(r)$. Second, the windows must be symmetric, $f(r) = f(K-1-r)$. Third, $f^2(r) + f^2(r+K/2) = 2$. So, PR is reserved.

$$\hat{x}_{m_0}(r)=\frac{1}{2}x_{m_0}(r)\{f(r+\frac{K}{2})h(\frac{K}{2}-r-1)+f(r)h(K-1-r)\} + \frac{1}{2}x_{m_0-1}(K-1-r)\{f(r+\frac{K}{2})h(r)-f(r)h(\frac{K}{2}+r)\} \quad (5)$$

The details of TDAC can be found in [6,7]. Note that Eqn. 3 are in the form of MDCT. That means TDAC can be achieved by using MDCT. In the following section, we

will discuss MDCT and its fast algorithm.

B. Modified DCT and Its Fast Algorithm

When MDCT codes "attack" sounds, such as the sounds of triangles and castanets, the "pre-echo" phenomenon will occur. Long transform size would make this problem more apparent and audible. In AC-3, a "transient detect" process is used to detect if a segment of music sound is stationary or transient. If it is stationary, AC-3 uses a long transform size of 512 points for a higher frequency resolution; otherwise, AC-3 uses two short transforms of size 256 points to code transient sound. So the "pre-echo" will be inaudible.

And now we will show the process that TDAC can be accomplished with MDCT here. In Eqn. 3, since the factor $cos(\pi mk)$ simply results in odd channels being inverted in frequency domain, it is omitted. Then, let $(-2/K)x[r] = x(r)$ $h(K-1-r)$ and $n_0 = 1/2+K/4$, Eqn. 3 would become the forward transform of AC-3 (Eqn. 6); α is the phase factor for indicating different transform sizes. Eqn. 7 is the inverse transform of Eqn. 6. Note that since there are K/2 unique bands in Eqn. 6, only K/2 coefficients are transmitted.

$$X_k(m) = \frac{-2}{K}\sum_{r=0}^{K-1} x(r)\cos\left(\frac{2\pi}{4K}(2r+1)(2k+1)+\frac{\pi}{4}(2k+1)(1+\alpha)\right) \quad (6)$$

where α =-1 for the first short transform
0 for the long transform
1 for the second short transform

$$x(r) = -\sum_{k=0}^{\frac{K}{2}-1} X_k(m)\cos\left(\frac{2\pi}{4K}(2r+1)(2k+1)+\frac{\pi}{4}(2k+1)(1+\alpha)\right) \quad (7)$$

We have proved that MDCT (Eqn. 6 both for 512 and 256 points) can be accomplished by just a K/2 points DCT_{IV}, i.e., a 256 point DCT_{IV}. Similarly, IMDCT can be accomplished by a 256 point DCT_{IV}. The detail of the proof is given in section C.

Since MDCT and IMDCT can be accomplished by a 256-point DCT_{IV}, we can just use the fast algorithms published in the literature, such as [2], [8], [10], etc, to speed up the computation. In [2], DCT_{IV} with size N is accomplished by FFT of size N/2, accompanying with pre-twiddle factors, post-twiddle factors, and a rearrangement process. [8] presents a recursive algorithm to decompose DCT_{IV} with size N to one DCT_{III} and one DST_{III} with size N/2. The DCT_{III} and DST_{III} can be further decomposed into DCT_{IV} with size N/4. All these steps can be accomplished with some additional multiplications, additions, and permutations. In [10], DCT_{IV} can be accomplished by DCT_{II} or DCT_{III} with multiplying pre-twiddle factors. Other approaches are also available. In [9], MDCT is directly derived from FFT with size N/4. In all of these papers, the complexity does not differ very much, i.e., the variation of the number of addition and multiplication required is very small. But some need larger memory. We have derived a FFT fast algorithm in which the complexity is the same as [2] and [9], but less memory is required. We have also derived a DCT_{IV} fast algorithm that is accomplished with one DCT_{II} and one DST_{II} of the same size. Although this algorithm is slower than the algorithm using FFT; however, time domain subsampling can be achieved easily. The details of both algorithms will be illustrated in the next section.

C. Derived MDCT Fast Algorithms

In the previous section, we mentioned that the kernels of MDCT and IMDCT are the same. As a result, we will focus on IMDCT only (i.e. the decoder process only). In this section, we will first show how to implement MDCT with FFT using less memory. Then, we will introduce another MDCT fast algorithm in which time domain subsampling is possible. In our system profile, the MDCT module is one of the most time-consuming function. With time domain subsampling, we can greatly reduce the execution time of this module with degraded but acceptable quality.

C.1 The MDCT Fast Algorithm Using FFT

Eqn. 6 is the IMDCT used in AC-3. Since the deductions of the switched blocks ($\alpha=\pm1$), as shown in Eqn. 6, of the IMDCT used in AC-3 are similar to the non-switched blocks ($\alpha=0$), we will just focus on the non-switched blocks (Eqn. 9) and the scale factor is omitted.

For $n = 0 \sim 2N-1$,

$$x(n) = -\sum_{k=0}^{N-1} X(k)\cos\left(\frac{2\pi}{4K}(2n+1)(2k+1)+\frac{\pi}{4}(2k+1)(1+\alpha)\right) \quad (8)$$

And if $\alpha=0$

$$x(n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(N+2n+1)(2k+1)] \quad (9)$$

Because of the symmetric property of IMDCT, which means in the first half IMDCT, the odd part can be reconstructed from the even part with only sign-inverse (i.e., $x(2n)=x(N-1-2n)$); while, in the second half IMDCT, the odd part can be reconstructed directly from the even part (i.e., $x(N+2n)=x(2N-1-2n)$). As a result, we will focus only on the even part in the following deductions.

For $n = 0 \sim \frac{N}{2} - 1$,

$$x(2n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(N+4n+1)(2k+1)], \quad (10)$$

$$x(N-1-2n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(3N-4n-1)(2k+1)], \quad (11-a)$$

$$= -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(4N-N-4n-1)(2k+1)], \quad (11-b)$$

$$= \sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(N+4n+1)(2k+1)], \quad (11-c)$$

$$x(N+2n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(3N+4n+1)(2k+1)], \quad (12)$$

$$x(2N-1-2n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(5N-4n-1)(2k+1)], \quad (13-a)$$

$$= -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(8N-3N-4n-1)(2k+1)], \quad (13-b)$$

$$= -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(3N+4n+1)(2k+1)], \quad (13-c)$$

Rewriting Eqn. 9, we can get Eqn. 14-b. Equally dividing *x(n)* into 4 parts, the even parts of IMDCT can be reduced to Equ. 15-c, Eqn. 16-b, Eqn. 17-b, and Eqn. 18-b.

For $n = 0 \sim 2N-1$,

$$x(n) = -\sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(N+2n+1)(2k+1)], \quad (14-a)$$

$$= -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(N+2n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\cos[\frac{2\pi}{8N}(N+2n+1)(2N-4k-1)], \quad (14-b)$$

For $n = 0 \sim \frac{N}{4}-1$,

$$x(2n) = -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(N+4n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\cos[\frac{2\pi}{8N}(N+4n+1)(2N-4k-1)], \quad (15-a)$$

$$= -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(N+4n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(N+4n+1)(4k+1)], \quad (15-b)$$

$$= -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(4(n+\frac{N}{4})+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(4(n+\frac{N}{4})+1)(4k+1)], \quad (15-c)$$

$$x(2n+N) = -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(2N+4(n+\frac{N}{4})+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(2N+4(n+\frac{N}{4})+1)(4k+1)], \quad (16-a)$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X(2k)\sin[\frac{2\pi}{8N}(4(n+\frac{N}{4})+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\cos[\frac{2\pi}{8N}(4(n+\frac{N}{4})+1)(4k+1)], \quad (16-b)$$

$$x(2n+\frac{N}{2}) = -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(2N+4n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(2N+4n+1)(4k+1)], \quad (17-a)$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X(2k)\sin[\frac{2\pi}{8N}(4n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\cos[\frac{2\pi}{8N}(4n+1)(4k+1)], \quad (17-b)$$

$$x(2n+\frac{3N}{2}) = -\sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(4N+4n+1)(4k+1)]$$

$$-\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(4N+4n+1)(4k+1)], \quad (18-a)$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X(2k)\cos[\frac{2\pi}{8N}(4n+1)(4k+1)]$$

$$+\sum_{k=0}^{\frac{N}{2}-1} X(N-1-2k)\sin[\frac{2\pi}{8N}(4n+1)(4k+1)], \quad (18-b)$$

Now, we introduce two new sequences, *Y(k)* and *y(n)*, which are defined as follows:

$$Y(k) = X(N-1-2k) + jX(2k), \qquad k = 0 \sim \frac{N}{2}-1 \quad (19-a)$$

$$y(n) = -x(2n+\frac{N}{2}) + jx(2n+\frac{3N}{2}), \qquad n = 0 \sim \frac{N}{4}-1$$

$$= -x(2n+N) - jx(2n), \qquad n = \frac{N}{4} \sim \frac{N}{2}-1 \quad (19-b)$$

Notice that *Y(k)* and *y(n)* relate to each other by:

$$y(n) = \sum_{k=0}^{\frac{N}{2}-1} Y(k) W_{8N}^{(4n+1)(4k+1)}, \qquad n = 0 \sim \frac{N}{2}-1, \quad (20)$$

where

$$W_N^{nk} = \cos(\frac{2\pi}{N}nk) + j\sin(\frac{2\pi}{N}nk)$$

Furthermore, the kernel ($W_{8N}^{(4n+1)(4k+1)}$) can be deducted to Eqn. 21 as:

$$W_{8N}^{(4n+1)(4k+1)} = W_{8N}^{16nk} \cdot W_{8N}^{4n+4k+1}$$

$$= W_{8N}^{16nk} \cdot W_{16N}^{8n+8k+2}$$

$$= W_{\frac{N}{2}}^{nk} \cdot W_{16N}^{8n+1} \cdot W_{16N}^{8k+1} \quad (21)$$

That is,

$$y(n) = \sum_{k=0}^{\frac{N}{2}-1} Y(k) W_{8N}^{(4n+1)(4k+1)}$$

$$= \sum_{k=0}^{\frac{N}{2}-1} Y(k) W_{\frac{N}{2}}^{nk} \cdot W_{16N}^{8n+1} \cdot W_{16N}^{8k+1}$$

$$= W_{16N}^{8n+1} \cdot \sum_{k=0}^{\frac{N}{2}-1} (Y(k) \cdot W_{16N}^{8k+1}) W_{\frac{N}{2}}^{nk}, \quad n = 0 \sim \frac{N}{2}-1 \quad (22)$$

As a result, an N/2-to-N/2-point FFT with the pre-twiddle and post-twiddle processes (c.f. Eqn. 22) can accomplish the N-to-2N-point IMDCT. Also notice that the pre-twiddle factors (W_{16N}^{8n+1}) equal to the post-twiddle (W_{16N}^{8k+1}) factors so only one table is needed.

C.2 The MDCT Fast Algorithm Using DCT and DST

With simple variable substitution (c.f. Eqn. 23), we can rewrite Eqn. 8 in the DCT_{IV} form but with longer length (N-to-2N) (c.f. Eqn. 24 and Eqn. 25). Using the symmetry properties of Eqns. 24 and 25 (as shown in Eqn. 26), IMDCT is reduced to DCT_{IV} (N-to-N) form (c.f. Eqn. 27).

Let

$$x'(n) = x(n+\frac{3N}{2}), \qquad n = 0 \sim \frac{N}{2}-1$$
$$= -x(n-\frac{N}{2}), \qquad n = \frac{N}{2} \sim 2N-1 \tag{23}$$

then

$$x'(n) = \sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(4N+2n+1)(2k+1)]$$
$$= \sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(2n+1)(2k+1)], \qquad n = 0 \sim \frac{N}{2}-1 \tag{24}$$

and

$$x'(n) = \sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(2n+1)(2k+1)], \quad n = \frac{N}{2} \sim 2N-1 \tag{25}$$

Since

$$x'(N+n) = -x'(N-1-n), \qquad n = 0 \sim \frac{N}{2}-1$$

and

$$x'(N+n) = x'(N-1-n), \qquad n = \frac{N}{2} \sim N-1 \tag{26}$$

we have

$$x'(n) = \sum_{k=0}^{N-1} X(k)\cos[\frac{2\pi}{8N}(2n+1)(2k+1)], \qquad n = 0 \sim N-1 \tag{27}$$

Eqn. 28 is the same as Eqn. 27 but with different variable names. Two new sequences *x(n)* and *y(n)* are deducted from *z(n)* (c.f. Eqn. 29). And inversely, given *x(n)* and *y(n)*, *z(n)* can be reconstructed (c.f. Eqn. 30).

Rewrite Eqn. 27 as

$$z(n) = \sum_{k=0}^{N-1} Z(k)\cos[\frac{2\pi}{8N}(2n+1)(2k+1)], \qquad n = 0 \sim N-1 \tag{28}$$

and let

$$x(n) = z(n) + z(n-1), \qquad y(n) = z(n) - z(n-1) \tag{29}$$

then

$$z(n) = \frac{x(n)+y(n)}{2} \tag{30}$$

With some further deduction, *x(n)* and *y(n)* can be reduced to Eqn. 31. Notice that they are in the form of DCT_{II} and DST_{II}, respectively (c.f. Eqn. 32). That means, IMDCT can be accomplished by one DCT_{II} and one DST_{II}.

Define

$$x(n) = \sum_{k=0}^{N-1}[Z(k)\cdot(2\cos\frac{2\pi}{8N}(2k+1))]\cdot\cos\frac{2\pi}{4N}n(2k+1), \quad n = 0 \sim N-1$$

and

$$y(n) = \sum_{k=0}^{N-1}[Z(k)\cdot(-2\sin\frac{2\pi}{8N}(2k+1))]\cdot\sin\frac{2\pi}{4N}n(2k+1), \tag{31}$$

And let

$$X(k) = 2Z(k)\cos\frac{2\pi}{8N}(2k+1), \qquad k = 0 \sim N-1$$
$$Y(k) = -2Z(k)\sin\frac{2\pi}{8N}(2k+1), \qquad k = 0 \sim N-1 \tag{32}$$

then we have

$$x(n) = \sum_{k=0}^{N-1} X(k)\cdot\cos\frac{2\pi}{4N}n(2k+1), \qquad n = 0 \sim N-1$$
$$y(n) = \sum_{k=0}^{N-1} Y(k)\cdot\sin\frac{2\pi}{4N}n(2k+1), \qquad n = 0 \sim N-1 \tag{33}$$

The algorithm proposed here is slower than the fast algorithm proposed in section C.1, because two N-to-N-point real transforms are needed here. However, time domain subsampling can be accomplished and the memory requirement is the same. A simple example is as follows: If we want to accomplish subsample-by-two, we choose the even points. According to the Nyquist Theorem[1], the upper half frequency coefficients should be set to zero to prevent alias. As a result, Eqn. 33 becomes Eqn. 34, and Eqn. 34 equals to the N/2-to-N/2-point DCT_{II} and DST_{II}. This is a recursive algorithm. According to our experiments, subsample-by-two is nearly indistinguishable, subsample-by-four can still have tolerable quality, and the distortion is apparent if the subsample factor is equal to or greater than eight. That is,

$$z(2n) = \frac{x(2n)+y(2n)}{2}, \qquad n = 0 \sim \frac{N}{2}-1$$
$$x(2n) = \sum_{k=0}^{\frac{N}{2}-1} X(k)\cdot\cos\frac{2\pi}{4N}2n(2k+1),$$
$$y(2n) = \sum_{k=0}^{\frac{N}{2}-1} Y(k)\cdot\sin\frac{2\pi}{4N}2n(2k+1), \tag{34}$$

IV. AC-3 AUDIO CODEC

In this section, we will discuss the functions of the whole AC-3 codec[5]. The bit allocation process and the forward and inverse transforms have been demonstrated in the previous sections, and will be skipped here. First, we describe the structure of the basic decodable element in AC-3. Secondly, we explain why and how AC-3 de-correlates the information between channels and successive blocks. Then, the exponent strategy and

mantissa strategy are discussed in section C. Finally, we illustrate the flow of the functions of AC-3 codec.

A. The Structure Of AC-3 Synchronization Frame

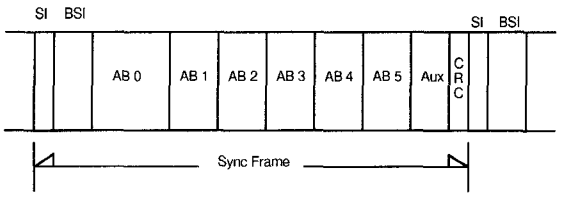

Fig. 3 The AC-3 bitstream format

An AC-3 serial coded audio bitstream is made up of a sequence of synchronization frames (c.f. Fig. 3). Each synchronization frame contains 6 audio blocks, each of which contains 256 new audio samples, as a total of 1536 new audio samples in a frame. Necessary parameters such as sampling rate, frame size, error detection code, and time stamp used for synchronization are of course contained in the bitstream. Other parameters such as channel remix parameters, dialogue parameter, and language code are also included in it. A synchronization frame can be divided to SI, BSI, AB_n AUX, and CRC parts. Some important elements are described below.

SI is the synchronization information header at the beginning of each frame, containing information needed to acquire and maintain synchronization, e.g. *syncword*, CRC check for the first 5/8 of the frame, sampling rate, and size of the frame. BSI (Bit Stream Information) contains the information associated with the whole bitstream, such as the type of service that the bitstream conveys, audio coding mode, mix level, etc.. AB_n is the n^{th} audio block. Audio block is the basic unit of MDCT. After MDCT, the frequency coefficients are divided to exponents and mantissas. At the end of synchronization frame is the AUX followed by the CRC. AUX is the auxiliary data field and is optional. The CRC word is used for error detection of the rest 3/8 frame.

B. Frequency Representation Quantization

An audio coder coding the differences of the perceptually significant spectral information is termed Differential Perceptual Audio Coder (DPAC). DPAC can achieve significant coding gains (up to 30%) with inaudible distortion and slightly higher complexity[4]. AC-3 achieves this high coding gain by coarsely quantizing the frequency domain representation of the audio signal. According to the experiments, music sounds exhibit some stationary of varying degree, raging from less than 100 ms to more than one second. Since there are 1536 samples in one sync frame in AC-3, ranging from 32ms to 48ms (from 48 kHz to 32 kHz), the correlation in spectra between successive audio block is generally high. AC-3 decorrelates this property in two ways. One is to group successive data; the other is to reuse the previously coded data. For example, in an audio block, successive exponents may be grouped together and only the difference between the current and previous group is coded. Some mantissas of the same SNR are also grouped together. Consecutive blocks within the same frame can reuse the previous block's data, such as exponent, bit allocation, and coupling, etc..

An another way to reduce the bit rate is the use of coupling. Coupling exploits properties of the human ear, which have to do with high frequency localization. The HAS follows individual waveform cycles of low frequency sounds and determine localization based on phase differences. But for high frequency sounds, the HAS can only follow the high frequency signal envelopes, not the detailed waveforms, due to physiological bandwidth limitations. Within narrow frequency bands, localization may use the inter-aural time delay of the high frequency signal envelope. So, the coupling technique in AC-3 maintains the high frequency signal envelope shape produced by each individual channel. However, the content of the signals within the envelope may contain information shared between channels. As a result, coupling is only used for high frequency signals. If coupling is used, channels included in the coupling process share a common encoded coupling channel above the coupling frequency. The coupling channel has its own exponents and mantissas. The fraction of the coupling channel and the coupled channels forms the coupling coordinates for each channel in coupling that are added to the bitstream for reproducing the original channel.

The simplest way to form a coupling channel is to sum up all the full-bandwidth channel's frequency data and divide it by 8 to avoid overflow. If the sign of each channel's data is altered to positive, phase cancellation in coupling can be avoided. The coupling channel is treated as an individual channel after it is constructed. So, it has its own exponent packing strategy, bit allocation information, etc. As mentioned earlier, the coupling coordinates of each coupled channel are also generated to compensate the differences between the coupled channel and the coupling channels. Not all of the frequency coefficients (bin) in couple have their own coupling coordinates. An equal banding structure is used here. Each equal band contains 12 bins and these bins share a common coupling coordinate.

When a decoder decodes a coupled channel to the frequencies above the starting coupling frequency, multiplying the correspondent coupling frequencies and relative coupling coordinates gets the frequencies.

C. Packing Exponents And Mantissas

A transformed coefficient is divided to exponent and mantissa. Since the coefficients are always less than 1, the exponents are negative and as a result, are sign inverted. The maximum value of exponents is 24. If an exponent is larger than 24, there may be leading zeros in the corresponding mantissa. Exponents can be shared across audio blocks of the same frame if they are very similar, i.e.,

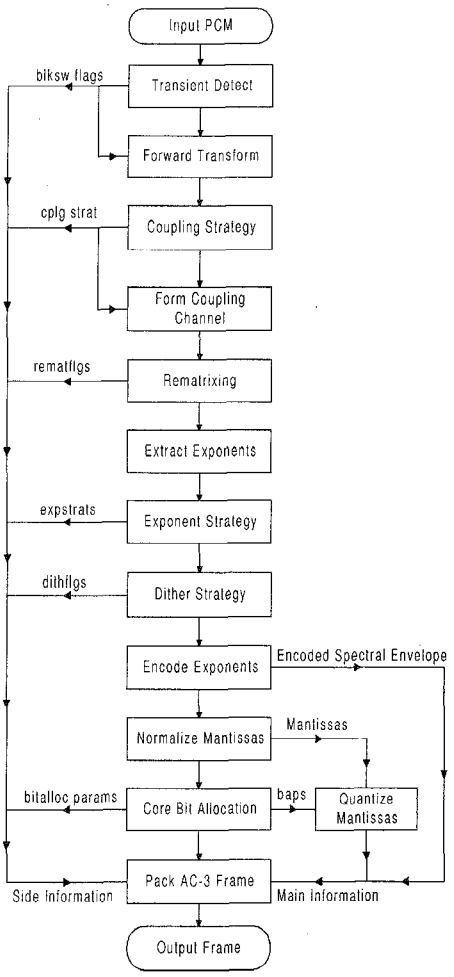

Fig. 4 The flowchart of an AC-3 encoder

audio blocks AB_1 to AB_5 can reuse the previous exponent data. Exponents of audio block AB_0 are always transmitted.

When coding the exponents of one block, the stationary of that block is first examined to determine the strategy used in its exponents. Three kinds of strategies can be used to code the exponents: D15, D25 and D45. 'D' stands for "differential", the middle '1,2,4' for the number of mantissas to share the same exponent, and the last '5' indicates 5 levels of quantization. If the exponents indicate a flat spectrum, an exponent strategy such as D25 or D45 may be used. If the spectrum is very tonal, a high spectral resolution exponent strategy such as D15 or D25 would be used. As a result, there is a tradeoff between fine frequency resolution, fine time resolution, and the number of bits required to send exponents.

Based on the selected exponent strategy, the exponents of each exponent set are preprocessed. D25 and D45 exponent strategies require that a single exponent be shared across 2 or 4 mantissas. The exponent sets are differentially coded for transmission in the bitstream. The dynamic range of the difference is confined to -2~2. When packing the exponent sets, the first exponent set is sent independently with 4 bits. The differences of the rest exponent sets are shift up to 0~5 and 3 successive exponent sets are grouped together and transmitted only with 7 bits (25*4+5*4+4=124). The differentially coded exponent sets are used to generate the spectral envelope for calculating bit allocation. A simple example is given below:

the original exponents: 4, 6, 5, 5, 7, 8, 6, 5, ...
the exponent strategy: D25
the exponent sets are 5, 5, 7, 5, ...
the first '5' is transmitted independently 0101
the rest three are differentially coded: 0, 2, -2, ...
after shift up, 2, 4, 0, ...
*after grouping, 25*2+5*4+0=70, 1000110*
the finally transmitted bitstream: 0101 000110 ...

After the spectral envelope is constructed, the bit allocation process discussed in section II is executed and the SNR (bap) of each mantissa is determined. Then, all mantissas are quantized to the fixed level of precision indicated by the corresponding bap. Symmetric and asymmetric quantizations are both used in AC-3. Mantissas quantized to 15 (bap=5) or fewer levels use symmetric quantization (i.e. equally quantized). Mantissas quantized to more than 15 levels use asymmetric quantization. Grouping is used for symmetric quantization baps of 1, 2, and 4. In an audio block (across channels), all the mantissas of bap 1, 2, and 4 through all the channels are grouped together. Every 3 mantissas of bap 1 (level=3) and 2 (level=5) are grouped together. Every 2 mantissas of bap 4 (level=11) are grouped together. If the number of mantissas of bap 1, 2, and 4 is not a multiple of 3 or 2, 0 is padded to produce a legal code. Symmetric quantization is to equally quantize the interval (-1, 1) according to the bap's level. Asymmetric quantization is the conventional 2's complement representation. A simple example is as follows:

the mantissa sequence: 0.1001, 0.1100101, -0.11, 0.101, 0.111001, ...
the corresponding bap: 1, 8, 1, 1, 5, ...
*the three bap=1 are group together: 9*2+3*0+2=20, 10100*
bap=8 are asymmetric quantized to: 1100101
bap=5 are symmetric quantized to: 1101
the final bitstream is 10100 1100101 1101 ...

D. The AC-3 Codec

Fig. 4 is the flowchart of an AC-3 encoder. We will introduce the basic processes of a typical encoder. Once the input PCM data of individual channel are in, they are first high-pass filtered to remove DC. A single pole filter at 3 Hz is used for a typical encoder. The low frequency effects (lfe) channel is low-pass filtered at 120 Hz. An 8^{th} order elliptic filter can be used here.

As we explained in section III, a "transient detection" process is needed to decide whether a long transform size

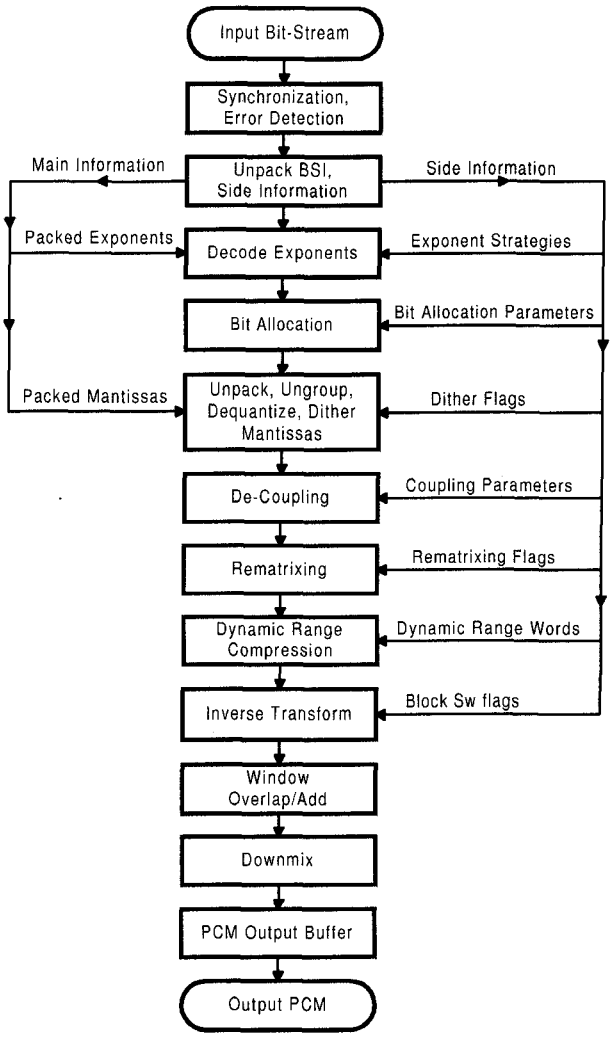

Fig. 5 The flowchart of an AC-3 decoder

is used or two short one instead. If short transforms are used, the 512 point block is split to two 256 point blocks and the outputs of the two transforms are interleaved after the forward transform.

After forward transform, the encoder decides to use coupling or not. If coupling is used, the coupling channel and the coupling coordinates for each coupled channel are formed. The corresponding flags are also set. Rematrix is mainly used for backward compatible with Dolby ProLogic. The whole spectrum is split to four segments. In each segment, the power of the original left and right channel is compared with the difference and the summation of them. If the maximum is one of the difference and summation channel, the original left and right in rematrix are replaced by the difference and the summation of them for higher coding gain. Rematrix is only used in audio coding mode 2/0.

The exponents and the mantissas are then extracted. According to the "shape" of the exponents, the exponent strategy is decided. Encoder also decides whether decoder should use dither or not. Dither is used to maintain approximately the same energy in the reproduced spectrum even if no bits are allocated to that mantissa. If the corresponding bap of a mantissa is 0, the use of the random value is conditional on the value of *dithflag*. Dither is applied after the individual channel is extracted from the coupling channel at the decoder.

After the exponents are packed, the encoded spectral envelope is used in the bit allocation process. The normalized mantissas are packed according to the baps. Then, the flags and the data decoder needed are packed to form a AC-3 frame and the CRC check word is also computed and filled in the AC-3 frame. AUX is the field for including auxiliary data.

Fig. 5 is the flowchart of an AC-3 decoder. Once the decoder receives a sync frame bitstream, *SI* (synchronization and error detection) and *BSI* (bitstream main information and side information) are first retrieved. The following steps should be executed for each audio block. Since there are six audio blocks in a frame, a loop is needed here. Audio block AB_0 contains all the information needed for data reconstruction, while $AB_{1\sim5}$ can reuse some of the previously decoded data. In fact, the following steps (except Downmix) are the inverse processes (exponent, mantissa, rematrix and inverse transform) or the same process (bit allocation) of the encoder. Downmix only happens in decoder to produce two-channel PCM source. Decoder can use different downmix methods according to *bsi* and if the surround information is kept. The final output PCM can be used for playback.

V. SYSTEM PERFORMANCE

All the data in this section are measured on the Pentium-133 PC, with VC++ 4.2 compiler. Fig. 6 is the measurement of the system performance without considering program I/O. The Y axis is the percentage of CPU time that the program takes, and the X axis is the measurement index.. It can be found that the program load is about 25% of CPU time. This means we can easily combine the AC-3 decoder with other I/O-bound applications or treat it as background audio player. In fact, we can both use Microsoft Word 7.0 and play an AC-3 audio sequence at the same time. Fig. 7 shows the same measurement but with time domain subsampling by two and four. About 18.8% (14.5%) improvement is achieved if the subsample factor is two (four).

Fig. 8 is the system module profile. Calc_mant() is the bit allocation process, and imdct() is the IMDCT function. We can see from the figure that calc_mant() is the most time consuming module, not the imdct()! The reason is demonstrated in Fig. 9 and Fig. 10. Fig. 9 is the profile of the calc_mant() and Fig. 10 is the profile of the imdct(). It is clear that in calc_mant() , even simple functions, such as min() and max(), are called many times and cost more than 7% of the CPU time (more than post_twiddle() and window_d() in imdct()). It is hard to reduce the cost since the AC-3 decoder has to recompute the bit allocation information for every coefficient. However, some

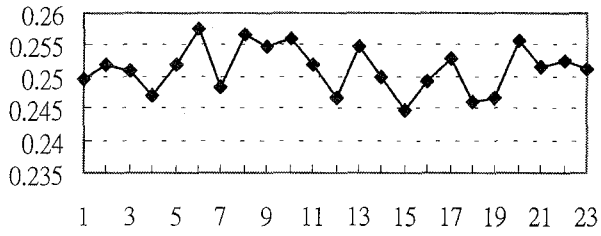

Fig. 6 The system performance (without I/O)

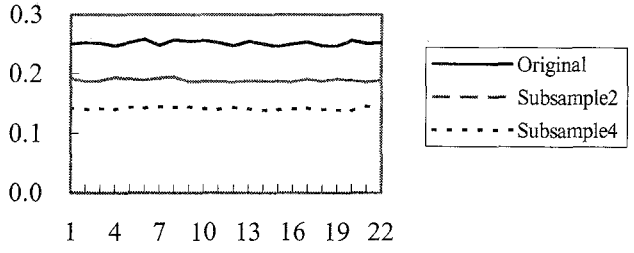

Fig. 7 The system performance with subsampling

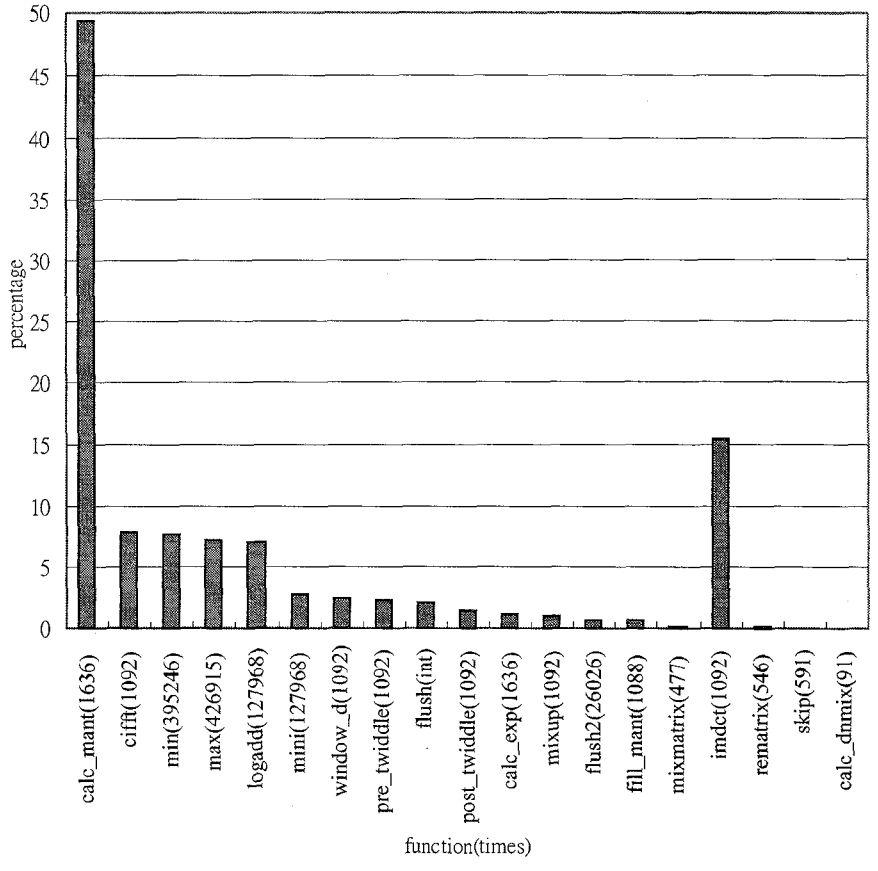

Fig. 8 The profile of the AC-3 decoder modules

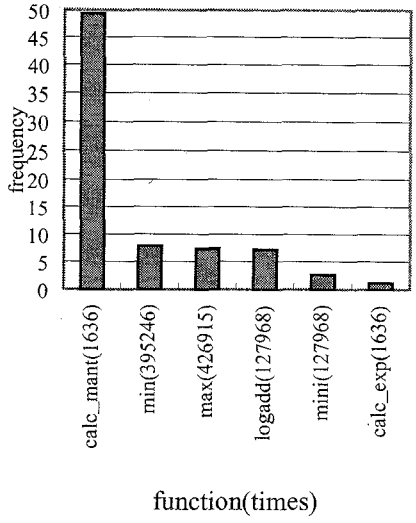

Fig. 9 The profile of calc_mant() module

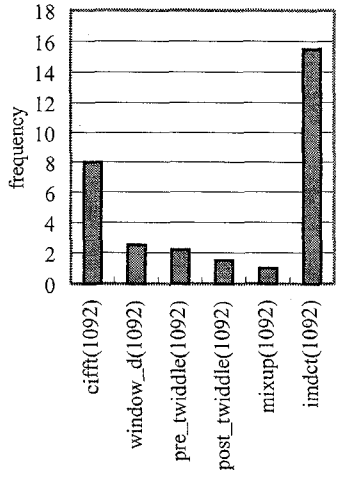

Fig. 10 The profile of *imdct()* module

improvement can still be achieved. Since audio blocks of the same frame may reuse the previously decoded information, we can record the previously decoded information and skip the bit allocation execution for that block. Another way is to rewrite the assembly code of this module. Moreover, we can use the "horizontal" bit allocation calculation instead of the "vertical" one that we adopted in our decoder. The reason is that the horizontal one is more suitable for parallel processing and MMX instructions can be used in some sub-processes. The drawback of the first suggestion is that the memory consumption overhead: we have to record all the information needed for every channel and some flag detections are needed. The drawback of the second one is hardware dependent, more complex, and not so effective.

In our system, we provide functions to turn-on or turn-off channels, and flexible downmix function is also provided. There are two down-mix modes (suggested in [5]): SREREO and STEREO_S. SREREO provides the traditional two channel downmix, and SREREO_S provides the two channel downmix with surround information. If different bitstreams modes and downmix factors are given, such as KARAOKE, we can also provide different downmix factors. We also use early downmix to reduce the IMDCT overhead since there are only two channels in PC and IMDCT is a linear transform.

VI. CONCLUSION

In this paper, we have presented the overall structure of AC-3 audio standard, and proposed two main approaches to accelerate our AC-3 decoder: bit allocation process speed up and IMDCT fast algorithm speed up. In the IMDCT fast algorithm speed up, two fast algorithms are provided, one for memory economization and the other for time domain subsampling. Our purpose is to develop a real-time AC-3 decoder and further reduce the workload of CPU.

Due to the excellent audio quality of AC-3, it is very suitable for combining with other standards, especially with video standards. For example, the combination with DVD video decoding software, such as MPEG-2, would produce a DVD software player. And this will be our future work.

REFERENCES

[1] N.S. Jayant, Peter Noll, "Digital Coding of Waveforms – Principles and Applications to Speech and Video", Prentice Hall.

[2] Henrique S. Malvar, "Signal Processing with Lapped Transforms", Artech House.

[3] Ken C. Pohlmann, "Principles of Digital Audio", McGraw-Hill, Inc.

[4] M. Paraskevas, J. Mourjopoulos, "A Differential Perceptual Audio Coding Method with Reduced Bitrate Requirements", IEEE trans. on Speech and Audio Processing, Vol. 3. No. 6, pp.490-503, Nov 1995.

[5] Advanced Television Systems Committee, "Digital Audio Compression (AC-3) Standard", Doc. A/52, Nov, 92.

[6] John P. Princen, Alan Bernard Bradley, "Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation", IEEE trans. on ASSP, Vol. 34, No. 5, pp.1153-1161, Oct. 1986.

[7] J. P. Pricen, A. W. Johnson, A. B. Bradley, "Subband/Transform Coding Using Filter Bank Designs Based on Time Domain Aliasing Cancellation", IEEE Proc. ICASSP'87, pp.2161-2164.

[8] Zhongde Wang, "On Computing the Discrete Fourier and Cosine Transforms", IEEE trans. on ASSP, Vol. ASSP-33, NO. 4, pp.387-392, Oct. 1985.

[9] P. Duhamel, Y. Mahieux, J.P. Petit, "A Fast Algorithm for the Implementation of Filter Banks Based on "Time Domain Aliasing Cancellation"", IEEE Proc. ICASSP'91, pp. 2209-2212.

[10] D.-Y. Chan, J, -F. Yang, S.-Y, Chen, "Regular Implementation Algorithms of Time Domain Aliasing Cancellation", IEE Proc. Image Signal Process., Vol. 143, No 6, pp.1341-1344, Dec 1996.