# A Progressive to Lossless Embedded Audio Coder (PLEAC) with Reversible Modulated Lapped Transform

Jin Li
Microsoft Research, One Microsoft Way, Bld. 113, Redmond, WA 98052.
Tel. + 1 (425) 703-8451 Email: jinl@microsoft.com

## ABSTRACT

A progressive to lossless embedded audio coder (PLEAC) has been proposed. PLEAC is based purely on reversible transform, which is designed to mimic the non-reversible transform in a normal psychoacoustic audio coder as much as possible. Coupled with a high performance embedded entropy codec, this empowers PLEAC with both lossless capability and fine granular scalability. The PLEAC encoder generates a bitstream that if fully decoded, completely recovers the original audio waveform without loss. Moreover, it is possible to scale this bitstream in a very large bitrate range, with granularity down to a single byte. Extensive experimental results support the superior lossless performance and bitstream scalability of the PLEAC coder.

## Keywords

Audio compression, lossless, scalable, progressive to lossless, fine granular scalability, reversible rotation, multiple forms lifting, reversible MLT.

## 1. INTRODUCTION

High performance audio codec brings digital music into reality. Popular audio compression technologies, such as MP3, MPEG-4 audio, Real™ and Windows Media Audio (WMA™), are usually lossy in nature. The audio waveform is distorted in exchange for higher compression ratio. In quality critical applications such as a recording/editing studio, it is imperative to maintain the best sound quality possible, i.e., the audio should be compressed in a lossless fashion. Most lossless audio coding approaches, such as [1][2][3][6], simply build upon a lossy audio coder, and further encode the residue. The compression ratio of such approaches is often affected by the underlying lossy coder. Since the quantization noise in the lossy coder is difficult to model, the approaches usually lead to inefficiency in the lossless audio coding. Moreover, it is also more complex, as it requires a base coder and a residue coder. Some other approaches, e.g., [4], build the lossless audio coder directly through a predictive filter and then encode the prediction residue. The approaches may achieve good compression ratio, however, it is not compatible with existing lossy audio coding framework. Since the compression ratio of a lossless coder is rather limited, usually 2-3:1, the ability to scale a lossless bitstream is very useful. The bitstream generated by the predictive filter based lossless coder can not be scaled. A lossy/residue coder can generate a bitstream with two layers, a lossy base layer and a lossless enhancement layer. However, the scaling can not go beyond the lossy base layer. If further scaling in the lossless enhancement layer is required, it is necessary to match the design of the residue coder with that of the lossy coder, which causes a lot of complications.

In this work, a progressive to lossless embedded audio coder (PLEAC) is proposed. PLEAC is based purely on a reversible transform, which is designed to mimic the non-reversible transform in a normal psychoacoustic audio coder as much as possible. Coupled with a high performance embedded entropy codec, this empowers PLEAC with both lossless capability and fine granular scalability. If fully decoded, the compressed bitstream produced by PLEAC completely recovers the original audio waveform, and achieves lossless compression. Yet, if higher compression ratio is desired, the application may extract a subset of the compressed bitstream and forms a higher compression ratio bitstream of lossy nature. Such scaling can be performed in a very large bitrate range, with granularity down to a single byte. With the progressive to lossless functionality of PLEAC, the application can easily balance between the amount of compression required and the desired audio quality, from a fairly high compression ratio all the way to lossless.

The rest of the paper is organized as follows. The framework of the PLEAC encoder is outlined in Section 2. The reversible multiplexers and the reversible modulated lapped transform (MLT) are examined in Section 3. The entropy coder and the bitstream assembly module are discussed in Section 4. Experimental results are shown in Section 5.

## 2. FRAMEWORK

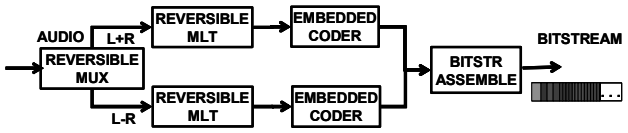


Figure 1 Framework

The encoder framework of the progressive to lossless embedded audio coder (PLEAC) can be shown in Figure 1. The PLEAC decoder is simply the reverse.

In PLEAC, the input audio waveform first goes through a reversible multiplexer (MUX). If the input audio is stereo, it is separated into L+R and L-R components, where L and R represent the waveform on the left and right audio channel, respectively. If the input audio is mono, the MUX simply passes through the audio. The waveform of each audio component is then transformed by a reversible modulated lapped transform (MLT) with switching windows. The window size can be either 2048 or 256 samples. After the reversible MLT transform, we group the MLT coefficients of a number of consecutive windows into a timeslot. In the current configuration, a timeslot consists of 16 long MLT windows or 128 short windows. A timeslot therefore consists of 32,768 samples, which is about 0.74 second if the input audio is sampled at 44.1kHz. The coefficients in the timeslot are then entropy encoded by a highly efficient sub-bitplane entropy coder, whose output bitstream can be truncated at any point later. Finally, a bitstream assemble module puts the bitstream of individual channels together, and forms the final compressed bitstream.

The framework of the PLEAC encoder is very similar to the embedded audio coder(EAC)[7][8], which is a high performance scalable lossy audio coder. In fact, we use the entropy coding module of EAC for PLEAC. It is the reversible MUX and reversi-

ble MLT modules that empower PLEAC to achieve the lossless functionality, and the bitstream assemble module that ensures the PLEAC compressed bitstream to be able to scaled from lossy to lossless, with fine granular scalability. Our discussion in the following is hence focused on the reversible MUX/MLT and the bitstream assemble module.

## 3. REVERSIBLE TRANSFORM

In this section, we discuss the reversible multiplexer (MUX) and the reversible modulated lapped transform (MLT) module. We notice (in Section 5) that the lossless compression efficiency is affected by the similarity between the reversible MLT and its non-reversible counter part. Therefore, we carefully design the reversible MLT so that its transform result mimics the result of a non-reversible MLT as much as possible.

### 3.1 Reversible multiplexer

Let $x$ and $y$ be the left and right channel, $x'$ and $y'$ be the multiplexed channel L+R and L-R, a reversible multiplexer can be implemented in lifting form as:

$$\begin{cases} \text{step } 0: y' = x - y \\ \text{step } 1: x' = x - \lfloor y'/2 \rfloor \end{cases}, \tag{1}$$

where $\lfloor . \rfloor$ denotes an integerize operation. The multiplexer (1) produces integer input from integer output and can be exactly reversed. Ignoring the nonlinearity in the integerize operation, the relationship between the input/output pair can be formulated through a linear transform as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2}$$

The determinant of the transform (2) is -1, which means the volume of the input/output space is equal.

### 3.2 Reversible rotation

To build a reversible MLT, we first build a reversible rotation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{3}$$

It is common knowledge that a rotation can be factored into lifting steps through:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \tag{4}$$

By using the integerize operation in each individual lifting step:

$$\begin{cases} \text{step } 0: z = x + \lfloor c_0 y \rfloor \\ \text{step } 1: x' = y + \lfloor c_1 z \rfloor \\ \text{step } 2: y' = z + \lfloor c_0 x' \rfloor \end{cases}, \tag{5}$$

where $c_0$=(cos$\theta$-1)/sin$\theta$ and $c_1$=sin$\theta$ are lifting parameters, the rotation becomes reversible. Existing researches on reversible DCT[5] and reversible MDCT[6] use the factorization in (4) as the basic operation for the reversible transform. It yields a reversible transform with compact data representation. However, with certain rotation angle, the quantization noise could be fairly large, and may lead to poor signal representation.

One contribution of this work is to factor the rotation operation with multiple forms. In addition to the factorization used in (4), we use three additional factorization forms:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & \frac{-\sin\theta - 1}{\cos\theta} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \cos\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{-\sin\theta - 1}{\cos\theta} \\ 0 & 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{\sin\theta - 1}{\cos\theta} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \cos\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{\sin\theta - 1}{\cos\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \frac{-\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{-\cos\theta - 1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{8}$$

We notice that the core of the factorization is still the three step lifting operation of (5). However, the pair of input/output variables may be swapped before (as in (7)) and after (as in (6)) the lifting operation. The sign of the input/output may be changed as well in certain cases. The additional forms of factorization lead to different lifting parameters $c_0$ and $c_1$ for the same rotation angle $\theta$, and let the reversible transform to mimic that of a linear non-reversible transform as closely as possible.

Let $\Delta x'$ and $\Delta y'$ be the quantization noise, which is the difference between the outcome of a reversible transform and that of a linear non-reversible transform. Our goal is to develop a reversible transform that minimizes the average energy of the quantization noise $E[\Delta x'^2] + E[\Delta y'^2]$. We notice that it is the integerize operation that introduces the quantization noise into the reversible transform. The coefficient swapping and sign changing operations in (6)-(8) do not introduce additional quantization noise. Let $\Delta$ be the quantization noise of a single integerize operation:

$$\lfloor x \rfloor = x + \Delta, \tag{9}$$

We may model the quantization noise in the reversible transform as:

$$\begin{bmatrix} \Delta x' \\ \Delta y' \end{bmatrix} = \begin{bmatrix} c_1\Delta_0 + \Delta_1 \\ (c_0c_1 + 1)\Delta_0 + c_0\Delta_1 + \Delta_2 \end{bmatrix} \tag{10}$$

where $\Delta_0$-$\Delta_2$ are the quantization noise at lifting steps 0-2. Let the quantization noise at each step be independent and identically distributed random variables, with $E[\Delta^2]$ be the average energy of the quantization noise of a single integerize operation. The average energy of the quantization noise can be calculated as:

$$E[\Delta x'^2] + E[\Delta y'^2] = \{(1 + c_0c_1)^2 + c_0^2 + c_1^2 + 2\}E[\Delta^2] \tag{11}$$

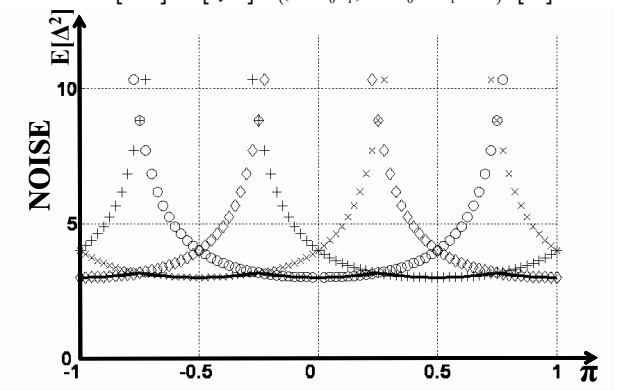


Figure 2 Quantization noise versus rotation angle of different factorization forms: the legends corresponding to the factorization forms are: o-(4), x-(6), +-(7) and ◊-(8). The solid line is the quantization noise by the combined factorization.

We plot the quantization noise versus rotation angles for different factorization forms (4),(6)-(8) in Figure 2. We observe that with any single factorization, the quantization noise can be large at certain rotation angle. By switching among different factorizations, more specifically, by using factorization forms (4), (6), (7) and (8) for rotation angles (-*0.25π*,0.25*π*), (-*0.75π*,-0.25*π*),

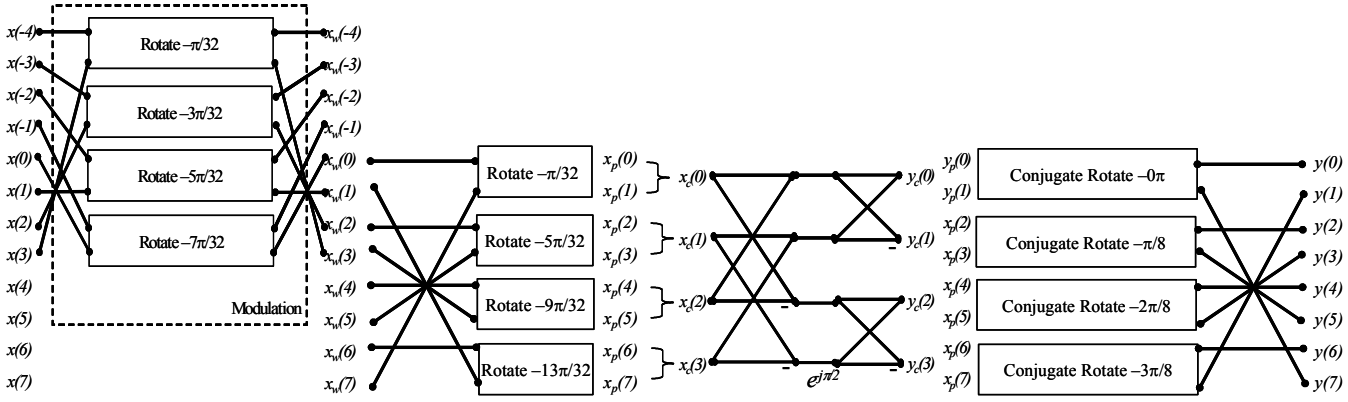


Figure 3 Four stages of the modulated lapped transform: (a) modulation, (b) pre-FFT rotation, (c) complex FFT, (d) post-FFT rotation.

($0.25\pi$,$0.75\pi$) and ($0.75\pi$,$1.25\pi$), respectively, we may control the quantization noise to be at most $3.2E[\Delta^2]$. To achieve the smallest possible quantization noise, we use the rounding towards the nearest integer as the integerize operation. Compared with truncation towards zero, this leads to significantly smaller quantization noise and better lossless compression performance, as shown in Section 5.

## 3.3 Reversible modulated lapped transform.

A modulated lapped transform (MLT) can be factored into a window modulation and a type IV DCT transform; the later of which can be further factored into a pre-FFT rotation, a complex FFT, and a post-FFT rotation operation. Using an 8-point MLT as an example, the four stages of the MLT can be illustrated in Figure 3(a)-(d), respectively. The modulation and pre-FFT rotation consist of only rotation operation (3), which can be reversibly implemented as in Section 3.1. The core of a fast complex FFT is the butterfly calculation:

$$\begin{bmatrix} y_c(i) \\ y_c(j) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & e^{-j\pi\omega} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_c(i) \\ x_c(j) \end{bmatrix}, \qquad (12)$$

where $x_c(i)$ and $y_c(i)$ are complex numbers. The first matrix is a complex rotation, which can be reversibly implemented according to the last section. The second matrix is a complex multiplexer. In this work, we implement it as a $0.25\pi$ rotation for both the real and imaginary part of $x_c(i)$ and $x_c(j)$. Note that with $0.25\pi$ rotation, there is a gain factor of $1/\sqrt{2}$, so the implemented reversible butterfly is:

$$\begin{bmatrix} y_c(i) \\ y_c(j) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ e^{-j\pi\omega} & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \cdot \begin{bmatrix} x_c(i) \\ x_c(j) \end{bmatrix}, \qquad (13)$$

In contrast to the butterfly (12) that has an absolute determinant of 2, the absolute determinant of butterfly (13) is 1, which does not expand the data and is thus more suitable for lossless compression purpose.

The core of the post-FFT rotation is the conjugate rotation operation, which can be implemented by changing the sign of the imaginary part after a normal rotation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ -\sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \qquad (14)$$

it can be reversibly implemented through Section 3.1 as well.

The reversible rotation has a determinant of 1, and the operations of swapping coefficients and changing sign have a determinant of -1. The absolute determinant of the entire reversible MLT is thus 1. Because a determinant 1 transform preserves the signal energy, the quantization noise of the reversible MLT is roughly proportional to the number of rotations in the transform, which is $O(N\log_2 N)E[\Delta^2]$ for an N-point MLT (note that there are two rotations per butterfly in the complex FFT). Such implementation is more favorable than factoring the reversible MLT (or type IV DCT) through an N-point LU transform, where the quantization noise that is caused by a long chain of lifting can be much larger.

# 4. ENTROPY CODING AND BITSTREAM ASSEMBLY

After the reversible transform, the MLT coefficients of multiple windows are grouped into a timeslot. The coefficients of each timeslot are then entropy encoded by a sub-bitplane entropy coder, which not only efficiently compresses the coefficients, but also renders the output bitstream with the embedding property, so that the bitstream of each channel can be truncated at any point. Our entropy coder derives the psychoacoustic masking from the partially coded coefficients during the embedded coding, hence the psychoacoustic masking (or the quantization step such as the scalefactor in MP3) need not to be sent to the decoder. Due to space limitation, the details of the entropy coder are not elaborated here. For the interested readers, we refer to [8].

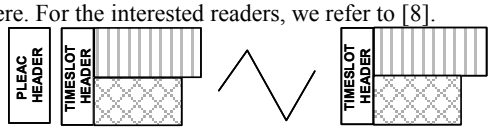


Figure 4 The PLEAC bitstream syntax.

Finally, a bitstream assembler puts together the embedded bitstream of the L+R and L-R channels, and forms the final PLEAC bitstream. The syntax of the PLEAC bitstream can be illustrated in Figure 4. There is a global header, which is followed by a number of timeslots. Each timeslot is again led by a header, which records the length of the compressed bitstream in the L+R and L-R channels, and is followed by the actual embedded bitstream of the channels. If exact waveform reconstruction is required, the entire bitstream will be decoded. In case higher compression ratio is called for, we extract a subset from the losslessly encoded bitstream to form a bitstream of higher compression ratio. Since this is achieved by truncating the embedded bitstream of the L+R and L-R channels of individual timeslot, the operation can be performed very fast. It is also possible to convert the compressed audio from stereo to mono by removing the compressed bitstream associated with the L-R channel.

## 5. EXPERIMENTAL RESULTS

To evaluate the performance of the progressive to lossless embedded audio coder (PLEAC), we benchmark it against the latest version of Monkey's Audio[4], which is the best lossless audio coders that the author is aware of. The test audio waveform is the MPEG-4 sound quality assessment materials (SQAM) downloaded from [9]. The original audio is in stereo and sampled at 44.1kHz. To check various design components, we test the lossless capability of PLEAC with a number of configurations:

a) PLEAC (described in the paper)
b) PLEAC with FFT butterfly implemented as (12)
c) PLEAC with factorization (4) only
d) PLEAC with truncation towards zero

The experimental results can be found in Table 1. For each clip, we list the lossless compression ratio (the higher, the better) for each tested algorithm/configuration. We notice that it pays to maintain smaller quantization noise, i.e., the reversible transform should mimic its non-reversible counterpart as closely as possible. Compare to using only factorization (4) for reversible rotation (configuration (c)) and using an integerize operation of truncation towards zero(configuration (d)), the scheme adopted in the paper improves the lossless compression performance by 48% and 5%, respectively. The improvement is very impressive considering that we have changed only a smaller component in the reversible transform module. Another important design aspect is that the absolute determinant of the reversible transform should be 1 (so that the volume of the coefficient data does not expand). If a reversible transform is used with a determinant larger than 1, as with butterfly (12) of configuration (b), there will be a degradation of lossless compression performance of 6%.

Table 1 Lossless compression ratio of different algorithms.

| Coder / Audio | Monkey's Audio3.97 | PLEAC (a) | PLEAC (b) | PLEAC (c) | PLEAC (d) |
|---|---|---|---|---|---|
| bass47_1 | 2.72 | 2.64 | 2.51 | 1.38 | 2.58 |
| frer07_1 | 6.43 | 6.25 | 5.83 | 2.47 | 5.54 |
| gspi35_1 | 4.07 | 4.37 | 4.13 | 1.59 | 4.06 |
| gspi35_2 | 2.96 | 3.35 | 3.15 | 1.40 | 3.19 |
| harp40_1 | 2.21 | 2.62 | 2.51 | 1.56 | 2.56 |
| horn23_2 | 3.99 | 3.88 | 3.69 | 1.40 | 3.71 |
| quar48_1 | 2.33 | 2.29 | 2.20 | 1.30 | 2.25 |
| sopr44_1 | 2.71 | 2.71 | 2.57 | 1.40 | 2.63 |
| spfe49_1 | 3.08 | 2.77 | 2.55 | 1.80 | 2.62 |
| spff51_1 | 2.57 | 2.02 | 1.89 | 1.50 | 1.97 |
| spfg53_1 | 3.07 | 2.83 | 2.62 | 1.87 | 2.69 |
| spme50_1 | 3.03 | 2.63 | 2.43 | 1.66 | 2.51 |
| spmf52_1 | 3.02 | 2.30 | 2.13 | 1.63 | 2.21 |
| spmg54_1 | 3.14 | 2.86 | 2.66 | 1.89 | 2.72 |
| trpt21_2 | 3.40 | 3.56 | 3.41 | 1.43 | 3.42 |
| vioo10_2 | 2.76 | 2.84 | 2.73 | 1.48 | 2.80 |
| Average | 3.22 | 3.12 | 2.94 | 1.61 | 2.97 |

PLEAC shows itself as a decent performance lossless audio coder, although it still trails behind the newest version of Monkey's audio for about 3%. Since PLEAC is based purely on the reversible transform, it differs from the lossy audio coder only by its transform module. It is thus easier for PLEAC to co-exist with an existing lossy audio coder. We notice that PLEAC outperforms Monkey's audio in certain instrumental music pieces, e.g., gspi35_1, gspi53_2 and harp40_1, while lags behind in speech coding, e.g., spfe49_1, spff51_1 and spmf52_1.

Another major advantage of PLEAC is that its bitstream can be scaled in a large bitrate range with granularity down to a single byte. We further evaluate the lossy compression performance of PLEAC. To measure the quality of the compressed audio, we use the noise-mask-ratios (NMR)[10], which measures the level (in dB) of audio coding noise above the just-noticeable-difference (JND) threshold. The lower the NMR, the less that the coding noise is audible; and thus the better the sound quality. Recognizing that NMR does not show every aspect of the audio defects, we also put the coded audio clips on the web[7] so that the readers may listen to the clips and evaluate themselves. Due to space limitation, we only list the average NMR value of the test audio clips. The average NMR of PLEAC with bitrate 256, 128 and 64kbps are shown in Table 2. We also test PLEAC configuration (c), i.e., with only factorization (4) in the reversible rotation. MP3 and Microsoft Windows Media Audio (WMA™) version 8.0 are used as benchmark for lossy audio compression in Table 2. Again it pays to design the reversible MLT as closely as possible to the non-reversible MLT. With just a change in the implementation of the reversible rotation, the lossy compression performance of PLEAC improves dramatically. PLEAC outperforms MP3, though it still lags behind the state-of-the-art WMA. With decent compression performance in both lossless and lossy mode, lossless capability, scalability with large bitrate range, fine granular scalability down to a single byte, PLEAC shows itself as a promising and versatile audio coder.

Table 2 Average NMR of PLEAC compressed audio.

| Coder / NMR | PLEAC 256kbps | PLEAC 128k | PLEAC 64k | PLEAC(c) 128k | MP3 128k | WMA 128k |
|---|---|---|---|---|---|---|
| Average(dB) | -5.26 | -0.40 | 3.23 | 9.63 | 8.15 | -2.95 |

## 6. REFERENCES


[1] M. Purat, T. Liebchen, P. Noll, "Lossless transform coding of Audio Signals", 102nd AES Convention, München, 1997.

[2] T. Moriya, A. Jin, T. Mori, K.Ikeda, T. Kaneko, "Lossless scalable audio coder and quality enhancement", Proc. *of the IEEE International Conf on Acoustics, Speech and Signal Processing,* Vol. 2, 2002, pp. 1829-1832.

[3] A. Wegener, "MUSICompress: lossless, low-MIPS audio compression in software and hardware", *Proc. International Conf on Signal Processing Applications and Technology,* San Diego, CA, 1997.

[4] Monkey's Audio, "A fast and powerful lossless audio compressor", http://www.monkeysaudio.com/ (Version 3.97).

[5] K. Komatsu and K. Sezaki, "Reversible discrete cosine transform", *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing,* Vol. 3, 1998, pp.1769-1772.

[6] R. Geiger, J. Herre, J. Koller and K. Brandenburg, "INTMDCT - a link between perceptual and lossless audio coding", *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing,* Vol. 2, 2002, pp.1813-1816.

[7] J. Li, "The embedded audio coder", http://research.microsoft.com/users/jinl/2001/eac/eac.htm.

[8] J. Li, "Embedded audio coding (EAC) with implicit auditory masking", ACM Multimedia 2002, Nice, France, Dec. 2002.

[9] "Sound quality assessment material recordings for subjective tests",http://www.tnt.uni-hannover.de/project/mpeg/audio/sqam/.

[10] B. Beaton, et. al, "Objective perceptual measurement of audio quality", in *Collected papers on digital audio bit-rate reduction* (N. Gilchrist and C. Crewin, eds.), pp.126-152, Audio Engineering Society, 1996.