

Low Noise Reversible MDCT (RMDCT) and its Application in Progressive-to-lossless Embedded Audio Coding

Jin Li

ABSTRACT

A reversible transform converts an integer input to an integer output, while retaining the ability to reconstruct the exact input from the output sequence. It is one of the key components for lossless and progressive-to-lossless audio codecs. In this work, we investigate the desired characteristics of a high performance reversible transform. Specifically, we show that the smaller the quantization noise of the reversible modified discrete cosine transform (RMDCT), the better the compression performance of the lossless and progressive-to-lossless codec that utilizes the transform. Armed with this knowledge, we develop a number of RMDCT solutions. The first RMDCT solution is implemented by turning every rotation module of a float MDCT (FMDCT) into a reversible rotation, which uses multiple factorizations to further reduce the quantization noise. The second and third solutions use the matrix lifting to implement a reversible fast Fourier transform (FFT) and a reversible fractional-shifted FFT, respectively, which are further combined with the reversible rotations to form the RMDCT. With the matrix lifting, we can design the RMDCT that has less quantization noise, and can still be computed efficiently. A progressive-to-lossless embedded audio codec (PLEAC) employing the RMDCT is implemented with superior results for both lossless and lossy audio compression.

1. INTRODUCTION

High performance audio codecs bring digital music into practical reality. Popular audio compression technologies, such as MPEG-1 layer 3 (MP3), MPEG-4 audio, Real Audio and Windows Media Audio (WMA), are lossy in nature. The audio waveform is distorted in exchange for higher compression ratio. In applications where audio quality is critical, such as a professional recording and editing, the compromise of trading distortion for compression is not acceptable. These applications must preserve the original audio. Any audio compression should be performed in a lossless fashion. An especially attractive feature of the lossless audio codec is the progressive-to-lossless, where the audio is compressed into a lossless bitstream, which may be further truncated at arbitrary point to a lossy bitstream of lesser bitrate without re-encoding. The progressive-to-lossless media codec offers the greatest flexibility in compression. During initial encoding, the media may be compressed to lossless, which preserves all the information of the original media. Later, if the transmission bandwidth or the storage space is insufficient to accommodate the full lossless media, the compressed media bitstream may be effortlessly truncated to whatever bitrate desired. The state-of-the-art image compression algorithm, JPEG 2000[1], has the progres-

sive-to-lossless compression mode. No existing audio codec operates in the progressive-to-lossless mode.

Most lossless audio coding approaches, such as [5][6][7], are built upon a lossy audio coder. The audio is first encoded with an existing lossy codec, then the residue error between the original audio and the lossy coded audio is encoded. The resultant compressed bitstream has two rate points, the lossy base bitrate and the lossless bitrate. It may not be scaled at the other bitrate points. Since the quantization noise in the lossy coder is difficult to model, such approaches usually lead to a drop in the lossless compression efficiency. Moreover, it is also more complex, as it requires the implementation of a base coder and a residue coder. Some other approaches, e.g., [8], build the lossless audio coder directly through a predictive filter and then encode the prediction residue. The approaches may achieve good lossless compression performance. However, there is still no scalability of the resultant bitstream.

To develop a progressive-to-lossless embedded audio codec, there are two key modules: the reversible transform and the lossless embedded entropy coder. The reversible transform is usually derived from the linear transform of a traditional psychoacoustic audio coder. By splitting the linear transform into a number of modules, and implementing each module with a reversible transform module, we can construct a larger reversible transform module whose output resembles that of the linear transform, except for the rounding errors. The reversible transform establishes a one-to-one correspondence between its input and output, and converts the input audio to a set of integer coefficients. The lossless embedded entropy coder then encodes the resultant coefficients progressively all the way to lossless, often in a sub-bitplane by sub-bitplane fashion. By incorporating both modules in the audio codec, we can achieve progressive-to-lossless. If the entire compressed bitstream is delivered to the decoder, it may exactly reconstruct the original audio. If the bitstream is truncated at a certain bitrate, the decoder may reconstruct a high perceptual quality audio at that bitrate. In this paper, we focus on the design of the reversible transform. We refer to [12] for details of the embedded entropy coder.

For a transform to be reversible, it must convert the integer input to the integer output, and be able to exactly reconstruct the input from the output. These two properties are essential to guarantee reversibility. Nevertheless, there are other desired properties of the reversible transform. Low computational complexity is certainly one of them. Another desired property is the normalization. Consider the following two reversible transforms, which are the candidates of the stereo mixer used in the progressive-to-lossless audio codec (PLEAC):

$$\begin{cases} \text{step 1: } x' = x + y \\ \text{step 2: } y' = x - y \end{cases}, \quad (1)$$

$$\text{and } \begin{cases} \text{step 1: } y' = x - y \\ \text{step 2: } x' = x - [0.5y'] \end{cases}, \quad (2)$$

where $[\cdot]$ is a rounding to integer operation, x and y are integer inputs, and x' and y' are integer outputs. Both transforms are reversible, however, the output of transform (1) generates a sparse output set because all points with $x'+y'$ equal to odd are not occupied. In comparison, the output of transform (2) is dense.

We notice that if the rounding operation in (2) is removed, it will become a linear transform. In general, let a reversible transform be:

$$\mathbf{y} = \text{rev}(\mathbf{x}), \quad (3)$$

where \mathbf{x} is the input integer vector, \mathbf{y} is the output integer vector, $\text{rev}(\cdot)$ denotes the reversible transform operator. If omitting all the rounding operators in the transform, the transform can be converted into a linear transform represented with matrix multiplication:

$$\mathbf{y}' = \mathbf{M}\mathbf{x}, \quad (4)$$

we call the matrix \mathbf{M} as the characterization matrix of the reversible transform. The characterization matrices of the reversible transforms (1) and (2) are:

$$\mathbf{M}_0 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \text{ and } \mathbf{M}_1 = \begin{bmatrix} 0.5 & 0.5 \\ 1 & -1 \end{bmatrix}, \text{ respectively.} \quad (5)$$

If \mathbf{X} is the set of all possible input data points, the output of the reversible transform occupies a volume roughly determined by $|\det(\mathbf{M})| \cdot \|\mathbf{X}\|$, where $\|\mathbf{X}\|$ is the volume of the input data set, and $|\det(\mathbf{M})|$ is the absolute determinant of the matrix \mathbf{M} . A valid reversible transform cannot have a characterization matrix with absolute determinant smaller than 1, because such transform will compact the data, and cause multiple input integer vectors mapping to one output integer vector, which contradicts the reversibility. A reversible transform with absolute determinant $|\det(\mathbf{M})|$ greater than 1 expands the input data set, and creates holes in the output data. It is extremely difficult to design a lossless entropy coder to deal with the holes in the output dataset, particularly if the reversible transform is complex. As a result, a desired property of the reversible transform is that the absolute determinant of its characterization matrix $|\det(\mathbf{M})|$ is one, i.e., the reversible transform is normalized.

In audio compression, we already know a good linear transform \mathbf{M} , the FMDCT, and need to design a RMDCT whose characterization matrix is the FMDCT. A common strategy of the reversible transform design is to factor the original linear transform into a series of simple linear transform modules,

$$\mathbf{M} = \prod_{i=1}^n \mathbf{M}_i, \quad (6)$$

It is easier to find the corresponding reversible transform for the simple module \mathbf{M}_i . We may then concatenate the reversible modules to form the reversible transform of \mathbf{M} . For such reversible transform design, another desired property concerns the quantization noise of the reversible transform, which is the deviation of the output of the reversible transform from that of the linear transform of its characterization matrix:

$$\mathbf{e} = \text{rev}(\mathbf{x}) - \mathbf{M}\mathbf{x}. \quad (7)$$

The quantization noise \mathbf{e} results from the rounding errors of various stages of the reversible transform. It is unavoidable, because it is the byproduct of reversibility, which forces the intermediate result and the output to be integers. The rounding error in each stage of the reversible transform can be considered as an independent random variable with no correlation with the input and output of that stage. Thus the aggregated quantization noise of the reversible transform also has likewise little correlation with the input and the output. Put in another way, the output of the reversible transform $\text{rev}(\mathbf{x})$ can be considered as the sum of the output of the linear transform $\mathbf{M}\mathbf{x}$ and a random quantization noise \mathbf{e} . It is preferable to design the reversible transform with as low quantization noise as possible. This is because the random quantization noise increases the entropy of the output, which reduces the lossless compression performance. Moreover, the quantization noise also creates a noise floor in the output of the reversible transform, which reduces the audio quality in the progressive-to-lossless stage as well. As a result, reduction of the quantization noise can improve the lossy compression performance. The correlation between the quantization noise level of the reversible transform and its lossless and lossy compression performance is confirmed by the experiments in Section 7.

The FMDCT can be factored into a series of rotations. One way to derive an RMDCT is thus to convert each and every rotation into a reversible rotation, as shown in [4]. It is common knowledge that a normalized rotation can be factored into a 3-step lifting operation via:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} = \begin{bmatrix} 1 & \frac{\cos\theta-1}{\sin\theta} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & \frac{\cos\theta-1}{\sin\theta} \\ 0 & 1 \end{bmatrix}. \quad (8)$$

By using rounding in each of the lifting steps, the rotation becomes reversible:

$$\begin{cases} \text{step 0: } z = x + [c_0 y] \\ \text{step 1: } x' = y + [c_1 z] \\ \text{step 2: } y' = z + [c_0 x'] \end{cases}, \quad (9)$$

where $c_0 = (\cos\theta - 1)/\sin\theta$ and $c_1 = \sin\theta$ are lifting parameters. Existing research on the reversible DCT[3] and the RMDCT[4] uses the factorization in (9) as the basic operation for the reversible transform. Though reversibility is achieved, the quantization noise of the approach (8) can be fairly large, and may lead to poor signal representation, and poor lossless and lossy compression performance.

An alternative method is to factor a large component of the linear transform \mathbf{M} into the upper and lower unit triangular matrices (UTM), which are triangular matrices with diagonal entries 1 or -1. It is shown in [9] that an even sized real matrix \mathbf{M} with absolute determinant of 1 can be factored into:

$$\mathbf{M} = \mathbf{P}\mathbf{L}_1\mathbf{U}\mathbf{L}_2, \quad (10)$$

where \mathbf{P} is a permutation matrix, \mathbf{L}_1 and \mathbf{L}_2 are lower UTMs, and \mathbf{U} is an upper UTM. Matrices \mathbf{L}_1 , \mathbf{L}_2 and \mathbf{U} can be reversibly implemented via lifting with N rounding operations, with N being the size of the matrix. The implementation of (10) leads to fewer rounding operations, and thus smaller quantization noise. Nevertheless, unlike a structured transform such as the FFT, there is usually no structure in matrix \mathbf{L}_1 , \mathbf{L}_2 and \mathbf{U} , and thus there is no fast algorithm to compute the mul-

complex matrix, while the other matrices in equation (16) are real matrices. This is interpreted by expanding every element of a complex matrix $\mathbf{C} = [c_{i,j}]_{i,j=0,1,L,N-1}$ into a 2x2 sub-matrix of the form:

$$\mathbf{C} = \begin{bmatrix} re_{i,j} & -im_{i,j} \\ im_{i,j} & re_{i,j} \end{bmatrix}_{i,j=0,1,L,N-1}, \quad (20)$$

where $re_{i,j}$ and $im_{i,j}$ are the real and imaginary part of the complex value $c_{i,j}$, respectively.

Like the FFT, the fractional-shifted FFT is an orthogonal transform. This can be easily verified as the Hermitian inner product of any two vectors of the fractional-shifted FFT is a delta function:

$$\frac{1}{N} \sum_j W_N^{-(i+\alpha)(j+\beta)} \cdot W_N^{+(k+\alpha)(j+\beta)} = \frac{1}{N} W_N^{-(k-i)\beta} \sum_j W_N^{(k-i)j} = \delta(k-i). \quad (21)$$

As a corollary, the inverse of the fractional-shifted FFT is:

$$\mathbf{F}_N^{-1}(\alpha, \beta) = \frac{1}{\sqrt{N}} [W_N^{-(i+\beta)(j+\alpha)}]_{i,j=0,1,L,N-1}. \quad (22)$$

The fractional-shifted FFT can be decomposed into a pre-rotation $\Lambda_N(\alpha, 0)$, FFT \mathbf{F}_N and a post-rotation $\Lambda_N(\beta, \alpha)$. The fractional-shifted FFT can thus be implemented via a standard FFT, as:

$$\mathbf{F}_N(\alpha, \beta) = \Lambda_N(\beta, \alpha) \mathbf{F}_N \Lambda_N(\alpha, 0), \quad (23)$$

where

$$\Lambda_N(\alpha, \beta) = \text{diag}\{W_N^{\alpha(j+\beta)}\}_{j=0,1,L,N-1}, \quad (24)$$

is a diagonal matrix of N rotations, and

$$\mathbf{F}_N = \frac{1}{\sqrt{N}} [W_N^{ij}]_{i,j=0,1,L,N-1}, \quad (25)$$

is the standard FFT.

We notice that the matrices \mathbf{P}_N and $\mathbf{Q}_N^{\text{DST}}$ are permutation matrices and may be implemented as such in the reversible transform. To derive the RMDCT from the FMDCT above, we simply need to turn the window rotation $h(n)$ into the reversible rotation, and implement the fractional-shifted FFT $\mathbf{F}_N(\mathbf{0.25}, \mathbf{0.25})$ with a reversible fractional-shifted FFT.

An alternative implementation of the FMDCT is to first group the signal into pairs of $\{x(n), x(N+n)\}$ with $n=0, \dots, N-1$, rotate them according to an angle specified by $h(n)$, and then transform the middle section of the signal through a type-IV DCT with:

$$\text{DCTIV}_N = \left[\sqrt{\frac{2}{N}} \cos\left(\frac{\pi}{N}(i+0.5)(j+0.5)\right) \right]_{i,j=0,1,L,N-1}, \quad (26)$$

The implementation can be shown in Figure 1(b). It is easily verified that an N -point type-IV DCT can be converted to an $N/2$ -point inverse fractional-shifted FFT:

$$\text{DCTIV}_N = \mathbf{P}_N \mathbf{F}_{N/2}^{-1}(\mathbf{0.25}, \mathbf{0.25}) \mathbf{Q}_N^{\text{DCT}} \mathbf{P}_N, \quad (27)$$

$$\text{with } \mathbf{Q}_N^{\text{DCT}} = \begin{bmatrix} \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} & & & \\ & \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} & & \\ & & \circ & \\ & & & \begin{bmatrix} 1 & \\ & -1 \end{bmatrix} \end{bmatrix}. \quad (28)$$

With the FMDCT, the two implementations of Figure 1 lead to the same result. However, they lead to slightly different derived reversible transforms. The FMDCT transform has other alternative forms and implementations, with different phases and window functions. Some alternative FMDCTs, termed modulated lapped transform (MLT), are shown in [11]. Nevertheless, all FMDCT and alternative forms can be decomposed into the window rotations and the subsequent type-IV DST/DCT. In this work, we derive the RMDCT from the FMDCT in the form of Figure 1(a). Nevertheless, the result can be easily extended to the RMDCT derived from the other FMDCT forms. For example, if an alternative form FMDCT uses the type-IV DCT implementation, we only need to implement the inverse fractional-shifted FFT $\mathbf{F}_{N/2}^{-1}(\mathbf{0.25}, \mathbf{0.25})$, instead of the forward fractional-shifted FFT.

3. REVERSIBLE MDCT I – REVERSIBLE ROTATION THROUGH MULTIPLE FACTORIZATIONS

In Figure 1(a), we show that the FMDCT consists of the window rotation $h(n)$, the type-IV DST and the sign change. The type-IV DST can be implemented via the fractional-shifted FFT, which in turn consists of the pre-rotation, the FFT and the post-rotation. The FFT can be implemented via the butterfly operations; more specifically, the $2N$ -point FFT can be implemented via first apply the N -point FFT on the odd and even index signal, and then combine the output via the butterfly:

$$\mathbf{F}_{2N} = \mathbf{B}_{2N} \begin{bmatrix} \mathbf{F}_N & \\ & \mathbf{F}_N \end{bmatrix} \mathbf{O}\mathbf{E}_{2N}, \text{ with } \mathbf{B}_{2N} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \mathbf{I}_N & \\ & -\Lambda_N(0.5, 0) \end{bmatrix} \quad (29)$$

where \mathbf{F}_{2N} and \mathbf{F}_N are the $2N$ - and N -point FFT, $\mathbf{O}\mathbf{E}_{2N}$ is a permutation matrix that separates the $2N$ complex vector into the size- N vector of even indices and the size- N vector of odd indices, and \mathbf{B}_{2N} is the butterfly operator. Note in the standard FFT([2] Chapter 12), the butterfly operator is:

$$\mathbf{B}'_{2N} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_N & \\ & -\Lambda_N(0.5, 0) \end{bmatrix}, \quad (30)$$

and a normalizing operation of $1/\sqrt{N}$ is applied after the entire FFT has been completed. However, this is not feasible in the reversible FFT, as normalizing by $1/\sqrt{N}$ is not reversible. We thus need to adopt (29) as the basic butterfly. The matrix $\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$ can be implemented via the conjugated rotation of $-\pi/4$. The matrix $\Lambda_N(\mathbf{0.5}, \mathbf{0})$ are N complex rotations. As a result, the entire FMDCT can be implemented via a series of rotations. By implementing each and every rotation through the 3-step lifting operation of (8) and (9), we can derive one implementation of the RMDCT. The problem of such implementation is that the quantization noise of certain rotation angles could be fairly large, which leads to large quantization noise of the RMDCT.

where \mathbf{P}_{2N} and \mathbf{Q}_{2N} are permutation matrices of size $2N \times 2N$, \mathbf{I}_N is the identity matrix, \mathbf{A}_N , \mathbf{C}_N and \mathbf{D}_N are $N \times N$ matrices, and \mathbf{B}_N is a non-singular $N \times N$ matrix.

Proof: Since \mathbf{S}_{2N} is non-singular, there exist permutation matrices \mathbf{P}_{2N}^t and \mathbf{Q}_{2N}^t so that

$$\mathbf{S}_{2N} = \mathbf{P}_{2N} \begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \mathbf{Q}_{2N}, \quad (40)$$

with \mathbf{S}_{12} being non-singular. Observing that:

$$\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{I}_N & \mathbf{0} \\ -\mathbf{S}_{12}^{-1}\mathbf{S}_{11} & \mathbf{I}_N \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{S}_{12} \\ -(\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21}) & \mathbf{S}_{22} \end{bmatrix}, \quad (41)$$

by taking determinant of \mathbf{S}_{2N} , and using the distributive property of the determinant, we have

$$\det(\mathbf{S}_{2N}) = \det(\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21}) \det(\mathbf{S}_{12}). \quad (42)$$

The matrix $\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21}$ is thus non-singular as well. Let \mathbf{U}_N be the matrix:

$$\mathbf{U}_N = (\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21})^{-1}. \quad (43)$$

By assigning matrices \mathbf{A}_N , \mathbf{B}_N , \mathbf{C}_N and \mathbf{D}_N to be:

$$\begin{cases} \mathbf{A}_N = (-\mathbf{I}_N + \mathbf{S}_{22})\mathbf{S}_{12}^{-1}, \\ \mathbf{B}_N = \mathbf{S}_{12}\mathbf{U}_N^{-1}, \\ \mathbf{C}_N = \mathbf{U}_N, \\ \mathbf{D}_N = -\mathbf{U}_N^{-1} + \mathbf{S}_{12}^{-1}\mathbf{S}_{11}, \end{cases} \quad (44)$$

substituting (44) into (40), we may easily verify that the equation (39) holds.

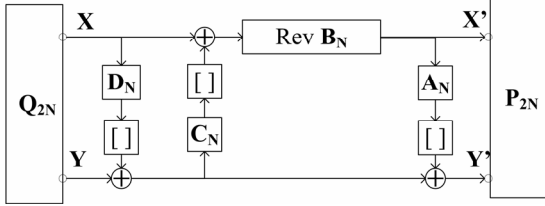


Figure 3 Forward reversible transform via the matrix lifting.

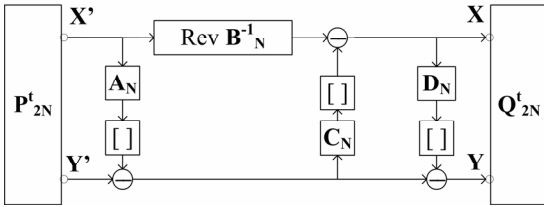


Figure 4 Inverse reversible transform via the matrix lifting.

Using the equation (39), we can derive a reversible transform from the linear transform of \mathbf{S}_{2N} . The operation flow of the resultant reversible transform can be shown in Figure 3. The input of the reversible transform is a size $2N$ (for real transform) or $4N$ (for complex transform, as each complex consists of an integer real part and an integer imaginary part) integer vectors. After the permutation operation \mathbf{Q}_{2N} , it is split into two size N (real) or size $2N$ (complex) integer vectors \mathbf{X} and \mathbf{Y} , which are transformed through:

$$\begin{cases} \mathbf{Y}_1 = \mathbf{Y} + [\mathbf{D}_N \mathbf{X}] \\ \mathbf{X}_1 = \mathbf{X} + [\mathbf{C}_N \mathbf{Y}_1] \\ \mathbf{X}' = \text{rev} \mathbf{B}_N(\mathbf{X}_1) \\ \mathbf{Y}' = \mathbf{Y}_1 + [\mathbf{A}_N \mathbf{X}'] \end{cases} \quad (45)$$

where \mathbf{A}_N , \mathbf{C}_N and \mathbf{D}_N are float transforms, $[\mathbf{x}]$ represents a vector rounding operation. Under the Cartesian coordinate, $[\mathbf{x}]$

can be implemented via rounding every element of \mathbf{x} . For a complex vector of \mathbf{x} , we may individually round the real and imaginary part of every element of \mathbf{x} . “Rev \mathbf{B}_N ” is a reversible transform to be derived from the linear non-singular transform \mathbf{B}_N . Finally, another permutation operation \mathbf{P}_{2N} is applied on the resultant integer vectors \mathbf{X}' and \mathbf{Y}' . Because each of the above operations can be exactly reversed, the entire transform is reversible. The inverse of the transform can be shown in Figure 4.

We call the operation in (45) matrix lifting, because it bears similarity to the lifting used in (9), except that the multiplication operation now is a matrix multiplication, and the rounding operation is a vector rounding. Note that our approach is different from the approach of [9], where the matrix \mathbf{S}_{2N} is factored into UTM matrices, each row of which is still calculated via scalar lifting.

Using different permutation matrices \mathbf{P}_{2N} and \mathbf{Q}_{2N} , we may derive different forms of the reversible transforms from the linear transform \mathbf{S}_{2N} with different lifting matrices \mathbf{A}_N , \mathbf{C}_N and \mathbf{D}_N and reversible core \mathbf{B}_N . The trick is to select the permutation matrices \mathbf{P}_{2N} and \mathbf{Q}_{2N} so that:

- The reversible core \mathbf{B}_N is as simple as possible. In the sub-optimal case, as in the reversible FFT, the reversible core \mathbf{B}_N consists of reversible rotations, which can be implemented via $3N$ lifting steps. In the best case scenarios, the reversible core \mathbf{B}_N consists of only permutations. In such scenarios, we may derive a reversible transform with only $3N$ (for real matrix \mathbf{S}_{2N}) or $6N$ (for complex matrix \mathbf{S}_{2N}) rounding operations. Compared to turning every small module, e.g., rotation, into the reversible rotation, which requires $O(N \log N)$ roundings, the matrix lifting may greatly reduce the number of rounding operations required in the reversible transform, and lower the quantization noise of the reversible transform.
- The computation complexity of the transforms \mathbf{A}_N , \mathbf{C}_N and \mathbf{D}_N is as low as possible.

4.2. The reversible FFT via the Matrix lifting

Let us derive the reversible FFT with the matrix lifting tool. Inspired by the radix-2 FFT of (29), a $2N$ -point FFT can be factored into:

$$\mathbf{F}_{2N} = \begin{bmatrix} \frac{1}{\sqrt{2}} \mathbf{F}_N & \frac{1}{\sqrt{2}} \mathbf{A}_N(0.5,0) \mathbf{F}_N \\ \frac{1}{\sqrt{2}} \mathbf{F}_N & -\frac{1}{\sqrt{2}} \mathbf{A}_N(0.5,0) \mathbf{F}_N \end{bmatrix} \mathbf{O} \mathbf{E}_{2N}, \quad (46)$$

Setting $\mathbf{P}_{2N} = \mathbf{I}_{2N}$, $\mathbf{Q}_{2N} = \mathbf{O} \mathbf{E}_{2N}$ and using (39), the matrix \mathbf{F}_{2N} can be factorized into the matrix lifting form of (39), with:

$$\begin{cases} \mathbf{A}_N = -\sqrt{2} \mathbf{F}_N^T \mathbf{A}_N(-0.5,0) - \mathbf{I}_N, \\ \mathbf{B}_N = -\mathbf{A}_N(0.5,0) \mathbf{F}_N \mathbf{F}_N = -\mathbf{A}_N(0.5,0) \mathbf{T}_N, \\ \mathbf{C}_N = -\frac{1}{\sqrt{2}} \mathbf{F}_N^T, \\ \mathbf{D}_N = (\sqrt{2} \mathbf{I}_N + \mathbf{F}_N^T \mathbf{A}_N(-0.5,0)) \mathbf{F}_N. \end{cases} \quad (47)$$

In (47), \mathbf{F}_N^t is the inverse FFT, \mathbf{T}_N is a permutation matrix in the form of:

$$\mathbf{T}_N = \begin{bmatrix} 1 & & & \\ & & & 1 \\ & & N & \\ & 1 & & \end{bmatrix}. \quad (48)$$

The reversible core \mathbf{B}_N is a permutation \mathbf{T}_N followed by N rotations $\mathbf{A}_N(0.5,0)$, which can be implemented via the multiple factorization reversible rotation we have developed in Section 3. The float transform \mathbf{A}_N consists of an inverse FFT, N rotations, and a vector addition². The transform \mathbf{C}_N is an inverse FFT. The transform \mathbf{D}_N consists of a forward FFT, an inverse FFT and N rotations. An N -point reversible FFT (with $2N$ input integers, as each input is complex with real and imaginary parts) can thus be implemented via (47), with the computation complexity being four $N/2$ -point complex FFT, N float rotations, and $N/2$ reversible rotations. It requires $4.5N$ roundings, with N roundings being used after each matrix lifting \mathbf{A}_N , \mathbf{C}_N and \mathbf{D}_N , and $1.5N$ roundings being used in the $N/2$ reversible rotations in \mathbf{B}_N . Compared with using reversible rotation to directly implement the reversible FFT, which requires $O(N \log N)$ roundings, the matrix lifting approach greatly reduces the number of rounding operations.

4.3. The reversible fractional-shifted FFT via the Matrix lifting

We may implement the RMDCT with the reversible FFT developed above. Yet, there is an even simpler implementation of the RMDCT. Observing that the type-IV DST, which is the most important component of the FMDCT, is directly related to the fractional shifted FFT $\mathbf{F}_N(\alpha, \beta)$ with $\alpha = \beta = 0.25$ through (16), we may derive a reversible fractional-shifted FFT directly with the matrix lifting. We notice that the fractional shifted FFT with $\alpha = \beta$ has the following properties:

$$\mathbf{R}_N(\alpha) = \mathbf{F}_N(\alpha, \alpha) \mathbf{A}_N(-2\alpha, \alpha) \mathbf{F}_N(\alpha, \alpha), \quad (49)$$

where $\mathbf{R}_N(\alpha)$ is a permutation matrix with only the element $(0,0)$ and elements $(i, N-i)$, $i=1, \dots, N-1$ being non-zero.

The proof is rather straight forward. Let $\mathbf{R}_N(\alpha) = [r_N(i, k)]_{i, k=0, 1, \dots, N-1}$, the element of $\mathbf{R}_N(\alpha)$ may be calculated through the matrix rotation as:

$$r_N(i, k) = \frac{1}{N} \sum_{j=0}^{N-1} w_N^{(i+\alpha)(j+\alpha)} w_N^{(-2\alpha)(j+\alpha)} w_N^{(j+\alpha)(k+\alpha)} = \frac{1}{N} \sum_{j=0}^{N-1} w_N^{(i+k)(j+\alpha)} = w_N^{(i+k)\alpha} \delta((i+k) \bmod N). \quad (50)$$

We thus have:

$$\mathbf{R}_N(\alpha) = \begin{bmatrix} 1 & & & \\ & & & w_1^\alpha \\ & & N & \\ & w_1^\alpha & & \end{bmatrix} \quad (51)$$

To derive the reversible transform from the fractional-shifted FFT with $\alpha = \beta = 0.25$, we again use the radix-2 FFT structure. Using equation (50), we factor the fractional shifted FFT as follows:

$$\mathbf{F}_{2N}(\alpha, \beta) = \mathbf{K}_{2N} \mathbf{S}_{2N} \mathbf{O} \mathbf{E}_{2N}, \quad (52)$$

with:

$$\mathbf{K}_{2N} = \begin{bmatrix} \mathbf{A}_N((1+\beta)/2 - \alpha, \alpha) & \mathbf{0} \\ \mathbf{0} & w_2^\beta \mathbf{A}_N((1+\beta)/2 - \alpha, \alpha) \end{bmatrix}, \quad (53)$$

$$\text{and } \mathbf{S}_{2N} = \begin{bmatrix} \frac{1}{\sqrt{2}} \mathbf{A}_N(-1/2, \alpha) \mathbf{F}_N(\alpha, \alpha) & \frac{1}{\sqrt{2}} \mathbf{F}_N(\alpha, \alpha) \\ \frac{1}{\sqrt{2}} \mathbf{A}_N(-1/2, \alpha) \mathbf{F}_N(\alpha, \alpha) & -\frac{1}{\sqrt{2}} \mathbf{F}_N(\alpha, \alpha) \end{bmatrix}. \quad (54)$$

Substituting $\alpha=0.25$ and expanding the fractional-shifted FFT via (23), we factor the transform \mathbf{S}_{2N} into the matrix lifting form of (39), with:

$$\begin{cases} \mathbf{A}_N = -\sqrt{2} \mathbf{A}_N(-0.25, 0.25) \mathbf{F}_N^T \mathbf{A}_N(-0.25, 0) - \mathbf{I}_N, \\ \mathbf{B}_N = -\mathbf{R}_N(0.25) = \begin{bmatrix} -1 & & & \\ & & & j \\ & & N & \\ & & & j \end{bmatrix}, \\ \mathbf{C}_N = -\frac{1}{\sqrt{2}} \mathbf{A}_N(-0.25, 0.25) \mathbf{F}_N^T \mathbf{A}_N(0.25, 0.5), \\ \mathbf{D}_N = \mathbf{A}_N(-0.25, 0.25) (\sqrt{2} \mathbf{I}_N + \mathbf{F}_N^T \mathbf{A}_N(-0.5, 0.125)) \mathbf{F}_N \mathbf{A}_N(0.25, 0) \end{cases} \quad (55)$$

In (55), the reversible core \mathbf{B}_N is a permutation matrix, as multiplying by j just swaps the real and imaginary part and changes the sign of the imaginary part. The reversible fractional-shifted FFT of $\alpha = \beta = 0.25$ is thus the matrix lifting of (55) plus the post reversible rotations of \mathbf{K}_{2N} , which again can be implemented via the multiple factorization rotations described in Section 3. Using equation (55), an N -point RMDCT can be implemented via $N/2$ reversible window rotations of $h(n)$, a reversible matrix lifting of $\mathbf{S}_{N/2}$, and $N/2$ reversible rotations of $\mathbf{K}_{N/2}$ (noticing that the N -point FMDCT consists of an N -point type-IV DST, which can be further converted to an $N/2$ -point fractional-shifted FFT). The total computational complexity is the sum of N reversible rotations, four $N/4$ -point float FFTs, and $1.75N$ float rotations. The implementation complexity is about double of an FMDCT, which requires two $N/4$ -point FFTs and $1.5N$ float rotations. Altogether, the RMDCT requires $4.5N$ roundings, with three $0.5N$ roundings after each matrix lifting of (55), and $3N$ roundings for the N reversible rotations.

5. REVERSIBLE MDCT: INTEGER ARITHMETIC

Most operations of the reversible transform, e.g., the add/subtract operation in the lifting, the permutation, and the sign change operation, are integer operations. The only place that requires floating point operation is in the lifting, where the input integer value (vector) is multiplied by a float value (or transformed through a float matrix), and then rounded. The lifting operation can certainly be implemented via float arithmetic, e.g., with double precision, which provides high precision and large dynamic range. However, float arithmetic is inconsistent across machines, and therefore, reversibility can not be guaranteed across different platforms. Moreover, float arithmetic is also more complex. Since the float result is ultimately rounded after the lifting, high precision floating point operation is not essential in the reversible transform. Floating point operation of the reversible transform may thus be implemented with integer arithmetic. The key is to keep the calculation error caused by integer arithmetic negligible compared to the quantization noise of the rounding operation.

² Scale by $\sqrt{2}$ can be rolled into either the FFT or the rotation operations $\mathbf{A}_N(\alpha, \beta)$, with no additional complexity required.

To implement the lifting operation with integer arithmetic, each operand of floating point operation is interpreted as a fixed precision float number:

$$\pm b_n b_{n-1} \dots b_1 . a_1 a_2 \dots a_m, \quad (56)$$

where m is the number of bits of the fractional part, and n is the number of bits of the integer part. The representation in (56) requires a total of $n+m+1$ bits (with one bit for sign). We call n the dynamic range, and m the precision, as a fixed precision float in (56) may represent values with absolute magnitude up to $2^n \cdot 2^{-m}$, and with precision down to 2^{-m} . To perform a floating point operation:

$$y = w \cdot x, \quad (57)$$

where x is the input, y is the output, and w is the multiplication value, the following operations are performed. Assuming that the input and output x and y are represented with n_x bit dynamic range and m_x bit precision, and the transform coefficient/multiplication value w is represented with n_w bit dynamic range and m_w bit precision, we may treat x , y and w as integer values, perform the multiplication, and then right shift the result by m_w bits.

The only component left is the required dynamic range and bit precision of the input, the output, and the transform coefficient. In this paper, these parameters are derived empirically. First, we investigate the dynamic range and bit precision needed to represent the transform coefficient, which in the RMDCT, is mainly the rotation angle w_N^i . The rotation can be implemented either via a 2×2 matrix multiplication, where the values $\cos\theta$ and $\sin\theta$ are used, or via a 3-step multiple factorization lifting developed in Section 3, where the values $c_0 = (\cos\theta - 1)/\sin\theta$ and $c_1 = \sin\theta$ are used. In the multiple factorization reversible rotation, the absolute value of c_0 can reach 2.414, which needs $n_w=2$ bit dynamic range. Thus, if the transform coefficient is represented with a 32 bit integer, it can have a bit precision of at most $m_w=29$ bits.

To investigate the impact of the bit precision of the transform coefficient on the quantization noise level of the reversible transform, we measure the magnitude of the quantization noise of the RMDCT versus that of the FMDCT, under different bit precisions of the transform coefficients. The quantization noise is measured in terms of the mean square error (MSE), the mean absolute difference (MAD) and the peak absolute difference (PAD), where

$$MSE = \frac{1}{N} \sum_{i=0}^{N-1} (y'_i - y_i)^2, \quad (58)$$

$$MAD = \frac{1}{N} \sum_{i=0}^{N-1} |y'_i - y_i|, \quad (59)$$

$$\text{and } PAD = \max_i |y'_i - y_i|. \quad (60)$$

In the above, y_i is the FMDCT coefficient, and y'_i is the RMDCT coefficient. The test audio waveform is the concatenated MPEG-4 sound quality assessment materials (SQAM)[13]. The result is shown in Table 1. The RMDCT in use is derived via the fractional-shifted FFT of Section 4.3. We first show in the 2nd column of Table 4 the quantization noise level of the RMDCT implemented via float arithmetic. Then, we show in the following columns the quantization noise

level of the RMDCT implemented via integer arithmetic, with the bit precision of the transform coefficients m_w being 29, 20, 16 and 12 bits. We observe that with a bit precision above 16 bits, the RMDCT implemented via integer arithmetic has a quantization noise level very close to that of float arithmetic. Less bit precision significantly increases the quantization noise level of the reversible transform, as there is not enough accuracy to correctly represent the multiplicative value/transform coefficient. In the rest of the paper, we choose the bit precision for the transform coefficient m_w to be 29bits, as this still allows the transform coefficient to be represented with a 32 bit integer. For the remaining 3 bits, 2 bits are used for the dynamic range of the transform coefficients ($n_w=2$), and 1 bit is used for the sign.

Table 1 Bit precision of the transform coefficient m_w .

Precision	Float arithmetic	$m_w=29$	20	16	12
MSE	0.47	0.47	0.48	0.49	5.94
MAD	0.53	0.53	0.54	0.54	1.18
PAD	11.86	11.86	11.86	11.86	286.56

Next, we investigate the bit precision m_x required to represent the input and output of the matrix lifting operations. We again compare the quantization noise level of the RMDCT versus that of the FMDCT, with different bit precisions of the input and output. The result can be shown in Table 2. It is evident that the quantization noise level starts to increase with fewer than $m_x=5$ bits to represent the intermediate result of the float transform.

Precision	Float arithmetic	$m_x=9$	5	4	3	2	1	0
MSE	0.47	0.47	0.48	0.51	0.60	0.98	2.47	8.08
MAD	0.53	0.53	0.54	0.55	0.58	0.69	0.98	1.73
PAD	11.86	11.86	11.86	10.86	10.02	15.17	27.38	49.16

Table 2 Bit precision of the input/output of the matrix lifting m_x .

Finally, we investigate the dynamic range n_x needed to represent the input and output of the matrix lifting. We notice that all operations used in the RMDCT, whether the float rotation, the reversible rotation, or the FFT, are energy preserving operations. Thus, the maximum magnitude that a coefficient can reach is:

$$\max = 2^{\text{bitdepth}-1} \sqrt{2N}, \quad (61)$$

where bitdepth is the number of bits of the input audio, and N is the size of the MDCT transform. We need one bit³ to further guard against overflow in the RMDCT. The dynamic range n_x needed for the matrix lifting is thus:

$$n_x = \text{bitdepth} + \frac{1}{2} \log_2(2N), \quad (62)$$

³ Noticing $\sqrt{2}$ in (55).

Table 3 Maximum bit depth of the input audio that can be fed into an RMDCT with 32 bit integer.

Precision m_x used	9	8	7	6	5	4
N=256	17	18	19	20	21	22
N=1024	16	17	18	19	20	21
N=4096	15	16	17	18	19	20

In Table 3, we list the maximum bitdepth of the input audio that can be fed into the RMDCT with 32 bit integer arithmetic implementation, with different bit precision m_x and RMDCT size. We have verified this table by feeding a pure sine wave of maximum magnitude into the RMDCT module, and by making sure that there is no overflow in the RMDCT module. With the RMDCT block size N being 1024, the RMDCT with 32 bit integer arithmetic may accommodate a 20-bitdepth input audio with $m_x=5$ bits left to represent the fractional part of the lifting.

6. PROGRESSIVE-TO-LOSSLESS EMBEDDED AUDIO CODEC (PLEAC) WITH THE RMDCT

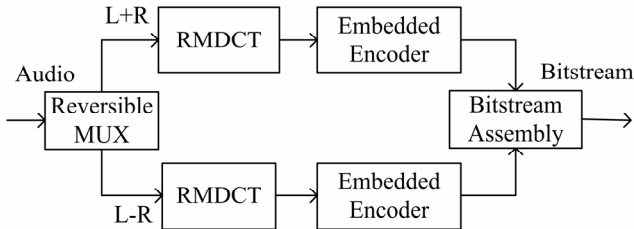


Figure 5 PLEAC encoder framework.

Using the RMDCT and the lossless embedded entropy coder developed in [12], we develop a progressive-to-lossless embedded audio coder (PLEAC). The encoder framework of PLEAC can be shown in Figure 5. If the input audio is stereo, the audio waveform first goes through a reversible multiplexer (MUX), whose formulation can be shown with equation (2), and separates into the L+R and L-R components, where L and R represent the audio on the left and right 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 an RMDCT module with switching windows. The RMDCT window size switched between 2048 and 256 samples. After the RMDCT transform, we group the RMDCT coefficients of a number of consecutive windows into a timeslot. In the current configuration, a timeslot consists of 16 long windows or 128 short windows, which in either case include 32,768 samples. The coefficients in the timeslot are then entropy encoded by a highly efficient psychoacoustic embedded entropy coder. The entropy encoder generates a bitstream that if delivered in its entirety to the decoder, may losslessly decode the input coefficients. Yet, the bitstream can be truncated at any point with graceful degradation. Finally, a bitstream assembly module puts the bitstreams of both channels together, and forms the final compressed bitstream. For the details of the sub-bitplane entropy coder and the bitstream assembly module, we refer the readers to [12].

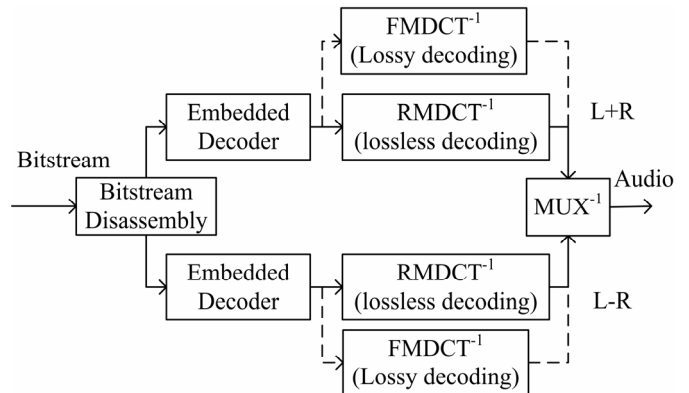


Figure 6 PLEAC decoder framework.

The framework of the PLEAC decoder can be shown in Figure 6. The received bitstream is first split into individual channel bitstreams by a disassembly module. Then, it is decoded by the embedded entropy decoder. If the decoder finds the received bitstream to be lossless or be close to lossless, i.e., the coefficients can be decoded to the last bitplane, the decoded coefficients are transformed by an inverse RMDCT module. Otherwise, an inverse FMDCT module is used. The reason to apply the inverse FMDCT for the lossy bitstream is that the inverse FMDCT not only has lower computational complexity, but also generates no additional quantization noise. In comparison, the inverse RMDCT generates quantization noise, which when decoded to lossless, serves to cancel the quantization noise of the encoding stage, but when decoded to lossy is just additional noise, which degrades the audio playback quality. After the inverse MDCT transform, the individual audio channels are then demultiplexed to form the decoded audio waveform.

Notice that the RMDCT module is always used in the PLEAC encoder, but only used in the lossless decoding mode in the PLEAC decoder. The computational penalty of the RMDCT thus only resides with the PLEAC encoder and lossless PLEAC decoder.

7. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed RMDCT, we put the RMDCT modules into the progressive-to-lossless embedded audio codec (PLEAC). We then compare the following RMDCT configurations:

- Through the reversible fractional-shifted FFT via the matrix lifting described in Section 4.3.
- Same as (a), except that the reversible rotation is implemented with only the factorization form of (8).
- Same as (a), except that the rounding operation is implemented as truncation towards zero.
- Through the reversible FFT via the matrix lifting described in Section 4.2.
- With only multiple factorization reversible rotations described in Section 3.

All the other modules of the PLEAC codec are the same. We first compare the output difference of the RMDCT modules versus that of the FMDCT module, in terms of the mean square error (MSE), the mean absolute difference (MAD) and the peak absolute difference (PAD) calculated in equations (58)-(60). We then compare the lossless compression ratio of

the PLEAC codecs using the specific RMDCT configuration with the state-of-the-art lossless audio compressor, Monkey’s Audio[8]. Finally, we compare the lossy coding performance. The lossy compressed bitstream is derived by truncating the losslessly compressed bitstream to the bitrate of 64, 32 and 16kbps. We then decode the lossy bitstream, and measure the decoding noise-mask-ratio (NMR) versus that of the original audio waveform (the smaller the NMR, the better the quality of the decoded audio). The NMR results are then compared with those of the lossy EAC codec, which is the PLEAC with the FMDCT module in both the encoder and the decoder. The test audio waveform is formed by concatenating the MPEG-4 sound quality assessment materials (SQAM)[13]. The aggregated comparison results are shown in Table 4 and Table 5.

Table 4 Quantization noise levels of different RMDCT modules.

PLEAC w/ RMDCT versus FMDCT	a	b	c	d	e
MSE	0.48	2.18	3.03	0.81	1.78
MAD	0.54	0.68	1.14	0.69	1.04
PAD	11.86	466.48	34.22	11.10	18.32

Table 5 Lossless & lossy compression performance of

Audio codec	PLEAC w/ RMDCT a	PLEAC w/ RMDCT b	PLEAC w/ RMDCT c	PLEAC w/ RMDCT d	PLEAC w/ RMDCT e	Monkey’s Audio (lossless)	EAC w/ FMDCT (lossy)
Lossless compression ratio	2.88:1	2.73:1	2.85:1	2.85:1	2.77:1	2.93:1	
Lossy NMR (64kbps)	-2.18	4.13	-1.60	-1.73	-0.06		-3.80
Lossy NMR (32kbps)	2.26	6.69	2.64	2.48	3.63		1.54
Lossy NMR (16kbps)	5.41	8.23	5.63	5.49	6.11		5.37

PLEAC.

Because the configuration (b) only differs from the configuration (a) in the implementation of the reversible rotations, their difference in Table 4 demonstrates the effectiveness of the multiple factorization reversible rotations. By intelligently selecting the proper factorization form under different rotation angles, multiple factorization greatly reduces the quantization noise in the rotations. The MSE of the quantization noise is reduced by 78%. There is also a noticeable improvement in the lossless (5% less bitrate) and lossy (on average 4.5dB better) coding performance of the PLEAC codec by replacing the single factorization reversible rotation with the multiple factorization reversible rotations.

The configuration (c) only differs from the configuration (a) in the implementation of the rounding operation. The configuration (a) uses rounding towards the nearest integer, while the configuration (c) uses the rounding towards zero. Though the two schemes only differ slightly in implementation, rounding

towards the nearest integer is proven to be a better choice for rounding. As shown in Table 4 and 5, the configuration (a) reduces the MSE by 84%, with slightly better lossless (1% less bitrate) and lossy (on average 0.4dB better) compression performance.

From the configuration (e) to (d) to (a), the matrix lifting is used to implement an ever-larger chunk of the FMDCT into a reversible module. Comparing the configuration (e) (only reversible rotations) with the configuration (a), which uses the matrix lifting on the fractional-shifted FFT, the quantization noise of the MSE is reduced by 73%, while there is a reduction of 4% of lossless coding bitrate. We also observe that the NMR of lossy decoded audio improves by an average of 1.4dB.

Overall, the RMDCT configuration with lower quantization noise leads to better lossless and lossy audio compression performance. The anomaly lies in the configuration (c). Though using truncation towards zero results in a big increase in the quantization noise in term of MSE, MAD and PAD, it does not incur as much penalty in the lossless and lossy compression performance as compared with the configurations (b) and (e). This anomaly may be explained by the fact that the truncation towards zero reduces the absolute value of the transform coefficients. Most entropy coders including the sub-bitplane entropy coder employed by PLEAC generate a shorter compressed bitstream with smaller transform coefficients. This

partly mitigates the rising quantization noise caused by using the truncation towards zero. Nevertheless, we notice that rounding towards the nearest integer still leads to superior performance in lossless and lossy compression.

It is observed that with the matrix lifting, the output of the RMDCT becomes very close to the FMDCT. The best RMDCT configuration (a), which is implemented via the reversible fractional-shifted FFT with the matrix lifting and the multiple-factorization reversible rotations, results in the MSE of the quantization

noise of only 0.48. Therefore, a large number of RMDCT coefficients are just the same as the FMDCT coefficients after rounding. Incorporating the RMDCT module (a), the PLEAC codec achieves a lossless compression ratio of 2.88:1, while the state-of-the-art lossless audio compressor, Monkey’s Audio[8], achieves a lossless compression ratio of 2.93:1 of the same audio waveform. PLEAC with the RMDCT is thus within 2% of the state-of-the-art lossless audio codec. Moreover, the PLEAC compressed bitstream can be scaled, from lossless all the way to very low bitrate, whereas such feature is non-existing in Monkey’s Audio. Comparing with the EAC codec, which uses the FMDCT in both the encoder and decoder, PLEAC only results in an NMR loss of 0.8dB. PLEAC is thus a fantastic all-around scalable codec from lossy all the way to lossless.

8. CONCLUSIONS

A low noise reversible modified discrete cosine transform (RMDCT) is proposed in this paper. With the matrix lifting and the multiple factorization reversible rotation, we greatly reduce the quantization noise of the RMDCT. We demonstrate that the RMDCT can be implemented with integer arithmetic, and thus be ported across platforms. A progressive-to-lossless embedded audio codec (PLEAC) incorporating the RMDCT module is implemented, with its compressed bitstream capable of being scaled from the lossless to any desired bitrate. PLEAC has decent lossless and lossy audio compression performance, with the lossless coding bitrate of PLEAC within 2% of the state-of-the-art of the lossless audio codec.

9. ACKNOWLEDGEMENTS

The author wishes to acknowledge H. S. Malvar and J. D. Johnston for their insightful comments and discussions; A. Colburn for proofreading the paper; and the associate editor and the anonymous reviewers for their close reading and thorough critique of the paper.

10. REFERENCES

- [1] D. S. Taubman and M. W. Marcellin, *JPEG 2000: image compression fundamentals, standards and practice*, Kluwer Academic Publishers.
- [2] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical recipes in C++*, Cambridge University Press.
- [3] K. Komatsu and K. Sezaki, "Reversible discrete cosine transform", *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing*, Vol. 3, May 1998, pp.1769-1772, Seattle, WA.
- [4] R. Geiger, J. Herre, J. Koller, and K. Brandenburg, "IntMDCT - A link between perceptual and lossless audio coding," in *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing, 2002*, Vol. 2, May 2002, pp.1813-1816, Orlando, FL.
- [5] M. Purat, T. Liebchen, P. Noll, "Lossless transform coding of Audio Signals", 102nd AES Convention, München, 1997.
- [6] T. Moriya, A. Jin, T. Mori, K. Ikeda, T. Kaneko, "Lossless scalable audio coder and quality enhancement", in *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing, 2002*, Vol. 2, May 2002, pp.1829-1832, Orlando, FL.
- [7] 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.
- [8] Monkey's Audio, "A fast and powerful lossless audio compressor", <http://www.monkeysaudio.com/> (Version 3.97).
- [9] J. Wang, J. Sun and S. Yu, "1-D and 2-D transforms from integers to integers", in *Proc of the IEEE International Conf on Acoustics, Speech and Signal Processing, 2003*, Vol. 2, Apr. 2003, pp.549-552, Hongkong, China.
- [10] R. Geiger, Y. Yokotani, G. Schuller, "Improved integer transforms for lossless audio coding", in *Proc of the 37th Asilomar Conf on Signals, Systems and Computers, 2003*, Nov. 9-12, 2003, pp. 2119-2123.

[11] H. S. Malvar, "Lapped transform for efficient transform/subband coding", *IEEE Trans. On ASSP*, vol. 38, no. 6, Jun. 1990, pp. 969-978.

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

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