REVERSIBLE FFT AND MDCT VIA MATRIX LIFTING

Jin Li

Microsoft Research, Communication, Collaboration and Signal Processing,
One Microsoft Way, Bld. 113, Redmond, WA, 98052.
Email: jinl@microsoft.com

ABSTRACT

Reversible transform not only converts integer input to integer output, but also reconstructs the exact input from the output. It is one of the key modules for lossless and progressive to lossless media codecs. We have established [1] that the closer the reversible transform is to its float counter part, the better the compression performance of the lossless and progressive to lossless codecs that utilize the transform. In this work, a new design of the reversible transform based on matrix lifting is proposed. With matrix lifting, we can design transform that is reversible, is close in transform value to its float counter part, and can be computed via fast algorithms. A progressive-to-lossless embedded audio codec employing the reversible MDCT with matrix lifting is implemented. Superior results on lossless and lossy audio compression are demonstrated.

1. INTRODUCTION

Reversible transform is a key module for lossless and progressive-to-lossless media codecs. To develop a progressive to lossless audio codec from a conventional audio codec, such as a MPEG-1 layer 3 (MP3) audio codec, the key is to swap the float modified discrete cosine transform (MDCT[1]) module with a reversible MDCT module, and then swap the entropy coding module with a lossless embedded entropy coding module. Both the reversible transform module and the lossless entropy coding module establish one-to-one correspondence between their input and output. Thus, from the compressed bitstream, the input audio can be exactly reconstructed.

An ideal reversible transform possesses the following characteristics. 1) It transforms integer input to integer output. 2) The integer input can be exactly reconstructed from the output. 3) The data volume of the output is the same as the input. 4) The forward and inverse transform can be computed efficiently, preferably with a fast transform structure. 5) The integer transform value should be as close as possible to the value of the float transform that it is referred. The first two characteristics ensure that the transform is indeed reversible. As long as the output is preserved in the entropy coding stage, the input can be exactly reconstructed from the inverse transform. The third character requires that the reversible transform be designed with reference to a normalized transform. It assures that the output be dense, which can be efficiently entropy encoded without waste. The forth character prefers a transform that can be implemented efficiently. The fifth character sets out that the output difference between the reversible transform and the float transform, which is defined as the quantization noise of the reversible transform, is as small as possible. The quantization noise is generated by the rounding operation used in various stages of the reversible transform, whose quantization noise has little correlation with the input and the output. We can thus consider the output of the reversible transform be the sum of a float transform plus a random, uncorrelated quantization noise. Since the random quantization noise cannot be compressed efficiently, the smaller the quantization noise, the fewer the bits are needed to encode the output (thus lead to better lossless performance). 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. It is shown in [1] that the reduction of quantization noise improves both the lossless and progressive-to-lossless compression performance.

Traditionally, a reversible transform is derived from its reference float transform by converting each and every basic module, often a rotation, into a corresponding reversible module, i.e., a reversible rotation. We notice that a float rotation can be implemented via a 3-step lifting operations as:

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

The rotation becomes reversible by using roundings in each lifting step:

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

where c_0=(cosθ-1)/sinθ and c_1=sinθ are lifting parameters, [x] is the rounding operation. An improved reversible rotation with smaller quantization noise has been proposed in [1], where three new factorizations of the rotation have been proposed:

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix}\begin{bmatrix}1 & \frac{-\sin\theta-1}{\cos\theta}\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0\\ \cos\theta & 1\end{bmatrix}\cdot\begin{bmatrix}1 & \frac{-\sin\theta-1}{\cos\theta}\\ 0 & 1\end{bmatrix} \quad (3)$$

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

$$\begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}=\begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & \frac{-\cos\theta-1}{\sin\theta}\\ 0 & 1\end{bmatrix}\cdot\begin{bmatrix}1 & 0\\ \sin\theta & 1\end{bmatrix}\cdot\begin{bmatrix}1 & \frac{-\cos\theta-1}{\sin\theta}\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad (5)$$

Equations (3)-(5) still use 3-step lifting, but with different lifting parameters c_0 and c_1. By using factorization forms (1), (3), (4) and (5) for rotation angles (-0.25π,0.25π), (-0.75π,-0.25π), (0.25π,0.75π) and (0.75π,1.25π), the quantization noise of the reversible rotation can be reduced.

[1] Sometimes, modulated lapped transform (MLT) is used, which is essentially MDCT with alternative window and phase.

An alternative method is to factor a large component of the float transform into upper and lower unit triangular matrixes (UTM), which are triangular matrixes with diagonal entry 1 or -1. It is shown in [3] that an even sized real matrix **T** with determinant of norm 1 can be factored into:

$$\mathbf{T} = \mathbf{P}\mathbf{L}_1\mathbf{U}\mathbf{L}_2, \tag{6}$$

where **P** is a permutation matrix, $\mathbf{L_1}$ and $\mathbf{L_2}$ are lower UTMs, and **U** is an upper UTM. Matrixes $\mathbf{L_1}$, $\mathbf{L_2}$ and **U** can be reversibly implemented via lifting with *N* rounding operations, with *N* being the size of the matrix. The implementation of (6) leads to less rounding operations for large transform matrix, and thus smaller quantization error. Nevertheless, unlike a structured transform such as FFT, there is usually no structure in matrix $\mathbf{L_1}$, $\mathbf{L_2}$ and **U**, and thus there is no fast algorithm to compute multiplication by matrix $\mathbf{L_1}$, $\mathbf{L_2}$ and **U.** The computational complexity of such approach is thus high.

In this work, we propose a new approach of implementing the reversible transform that we called matrix lifting. In stead of using lifting on the scalar variables, matrix lifting applies a lifting step as a float transform followed by a vector rounding. We demonstrate that an entire transform, e.g., a fast Fourier transform (FFT) or a large part of MDCT, may be reversibly implemented via matrix lifting, with reduced rounding operations and efficient fast transform.

The rest of the paper is organized as follows. The basic idea of the matrix lifting is presented in Section 2. The implementation of the reversible FFT and MDCT with matrix lifting is shown in Section 3. Experimental results are shown in Section 4.

2. MATRIX LIFTING

Theory 1: Every non-singular even sized matrix $\mathbf{S_{2N}}$ (real or complex) of size 2Nx2N can be factored into :

$$\mathbf{S}_{2N} = \mathbf{P}_{2N}\begin{bmatrix} \mathbf{I}_N & \\ \mathbf{A}_N & \mathbf{I}_N \end{bmatrix}\begin{bmatrix} \mathbf{B}_N & \\ & \mathbf{I}_N \end{bmatrix}\begin{bmatrix} \mathbf{I}_N & \mathbf{C}_N \\ & \mathbf{I}_N \end{bmatrix}\begin{bmatrix} \mathbf{I}_N & \\ \mathbf{D}_N & \mathbf{I}_N \end{bmatrix}\mathbf{Q}_{2N}, \tag{7}$$

where $\mathbf{P_{2N}}$ and $\mathbf{Q_{2N}}$ are permutation matrixes of size 2Nx2N, $\mathbf{I_N}$ is the identity matrix, $\mathbf{A_N}$, $\mathbf{C_N}$ and $\mathbf{D_N}$ are NxN matrixes, and $\mathbf{B_N}$ is a non-singular NxN matrix.

Proof: Since $\mathbf{S_{2N}}$ is non-singular, there exists permutation matrixes $\mathbf{P}^{\mathbf{t}}_{2N}$ and $\mathbf{Q}^{\mathbf{t}}_{2N}$ so that

$$\mathbf{S}_{2N} = \mathbf{P}_{2N}\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\mathbf{Q}_{2N}, \tag{8}$$

with $\mathbf{S_{12}}$ being non-singular. Observing that:

$$\begin{bmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix}\begin{bmatrix} \mathbf{I}_N & \\ -\mathbf{S}_{12}^{-1}\mathbf{S}_{11} & \mathbf{I}_N \end{bmatrix} = \begin{bmatrix} & \mathbf{S}_{12} \\ -(\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21}) & \mathbf{S}_{22} \end{bmatrix}, \tag{9}$$

By taking determinant of $\mathbf{S_{2N}}$, and using the distributive property of the determinant, we have

$$\det(\mathbf{S}_{2N}) = \det(\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21})\det(\mathbf{S}_{12}). \tag{10}$$

The matrix $\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21}$ is thus non-singular as well. By assigning:

$$\mathbf{U}_N = (\mathbf{S}_{22}\mathbf{S}_{12}^{-1}\mathbf{S}_{11} - \mathbf{S}_{21})^{-1}, \tag{11}$$

$$\mathbf{A}_N = (-\mathbf{I}_N + \mathbf{S}_{22})\mathbf{S}_{12}^{-1}, \tag{12}$$

$$\mathbf{B}_N = \mathbf{S}_{12}\mathbf{U}_N^{-1}, \tag{13}$$

$$\mathbf{C}_N = \mathbf{U}_N, \tag{14}$$

$$\mathbf{D}_N = -\mathbf{U}_N^{-1} + \mathbf{S}_{12}^{-1}\mathbf{S}_{11}. \tag{15}$$

We can easily verify that (7) holds, and $\mathbf{B_N}$ is a non-singular.

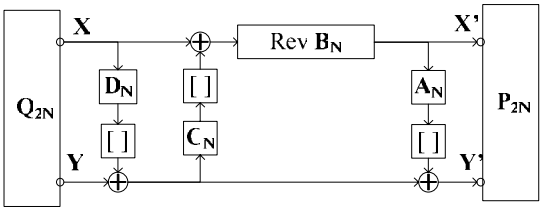

Figure 1 Forward reversible transform via matrix lifting.

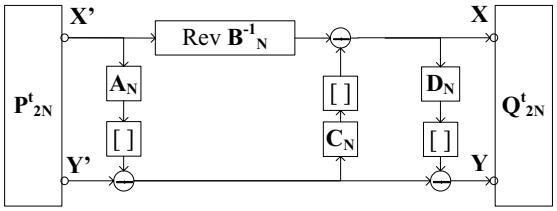

Figure 2 Inverse reversible transform via matrix lifting.

Using equation (7), we can derive a reversible transform shown in Figure 1. 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 **X** and **Y**, which are transformed through:

$$\begin{cases} \mathbf{Y}_1 = \mathbf{Y} + [\mathbf{D}_N\mathbf{X}] \\ \mathbf{X}_1 = \mathbf{X} + [\mathbf{C}_N\mathbf{Y}] \\ \mathbf{X'} = \mathbf{revB}_N(\mathbf{X}_1)' \\ \mathbf{Y'} = \mathbf{Y}_1 + [\mathbf{A}_N\mathbf{X}] \end{cases} \tag{16}$$

where $\mathbf{A_N}$, $\mathbf{C_N}$ and $\mathbf{D_N}$ are float transforms, [***x***] represents a vector rounding operation. In Cartesian coordinate, [***x***] can be implemented via rounding every element of ***x***. "Rev $\mathbf{B_N}$" is a reversible transform derived from the non-singular transform $\mathbf{B_N}$. Finally, another permutation $\mathbf{P_{2N}}$ is applied on the resultant integer vectors **X'** and **Y'**. Because each of the above operation can be exactly reversed, the entire transform is reversible with the inverse of the transform shown in Figure 2.

We call the operation in (16) as matrix lifting, because it bears similarity to the lifting used in (2), 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 [3], where matrix $\mathbf{S_{2N}}$ is factored into UTM matrices, each row of which is still calculated via scalar lifting.

Using different permutation matrixes $\mathbf{P_{2N}}$ and $\mathbf{Q_{2N}}$, we may factor the transform $\mathbf{S_{2N}}$ into reversible transforms with different lifting matrix $\mathbf{A_N}$, $\mathbf{C_N}$ and $\mathbf{D_N}$ and reversible core $\mathbf{B_N}$. The trick is to select the permutation matrixes $\mathbf{P_{2N}}$ and $\mathbf{Q_{2N}}$ so that:

a) The reversible core $\mathbf{B_N}$ is as simple as possible.
b) The computation complexity of transform $\mathbf{A_N}$, $\mathbf{C_N}$ and $\mathbf{D_N}$ is as low as possible.

In the following, we derive the reversible transforms for FFT and MDCT.

3. REVERSIBLE FFT AND MDCT

3.1. Reversible FFT via Matrix lifting

An N-point normalized fast Fourier transform (FFT) takes the form:

$$\mathbf{F_N} = \frac{1}{\sqrt{N}}\left[w_N^{ij}\right]_{i,j=0,1,\cdots,N-1}, w_N = e^{-j2\pi/N}. \quad (17)$$

In the following, we derive the reversible transform for a 2N-point FFT $\mathbf{F_{2N}}$. Inspired by the radix-2 FFT algorithm, we first apply permutation operation $\mathbf{OE_{2N}}$ on the input, which separates a 2N complex vector into a size-N complex vector of even indexes, and a size-N vector of odd indexes. We now have:

$$\mathbf{F_{2N}} = \begin{bmatrix} \frac{1}{\sqrt{2}}\mathbf{F_N} & \frac{1}{\sqrt{2}}\mathbf{\Lambda_N}(0.5,0)\mathbf{F_N} \\ \frac{1}{\sqrt{2}}\mathbf{F_N} & -\frac{1}{\sqrt{2}}\mathbf{\Lambda_N}(0.5,0)\mathbf{F_N} \end{bmatrix}\mathbf{OE_{2N}}, \quad (18)$$

where $\mathbf{F_N}$ is the N-point FFT, and

$$\mathbf{\Lambda_N}(\alpha,\beta) = diag\left\{w_N^{\alpha(j+\beta)}\right\}_{j=0,1,\cdots,N-1} \quad (19)$$

is a diagonal matrix of N rotations. Its inverse takes the form:

$$\mathbf{\Lambda_N^{-1}}(\alpha,\beta) = \mathbf{\Lambda_N}(-\alpha,\beta). \quad (20)$$

Setting $\mathbf{P_{2N}}=\mathbf{I_{2N}}$ and $\mathbf{Q_{2N}}=\mathbf{OE_{2N}}$, matrix $\mathbf{F_{2N}}$ can be factorized into the matrix lifting form of (7), with:

$$\begin{cases} \mathbf{A_N} = -\sqrt{2}\mathbf{F_N^T}\mathbf{\Lambda_N}(-0.5,0) - \mathbf{I_N}, \\ \mathbf{B_N} = -\mathbf{\Lambda_N}(0.5,0)\mathbf{F_N}\mathbf{F_N} = -\mathbf{\Lambda_N}(0.5,0)\mathbf{T_N}, \\ \mathbf{C_N} = -\frac{1}{\sqrt{2}}\mathbf{F_N^T}, \\ \mathbf{D_N} = \left(\sqrt{2}\mathbf{I_N} + \mathbf{F_N^T}\mathbf{\Lambda_N}(-0.5,0)\right)\mathbf{F_N}. \end{cases} \quad (21)$$

In (21), $\mathbf{F_N^t}$ is the inverse FFT, $\mathbf{T_N}$ is a permutation matrix in the form of:

$$\mathbf{T_N} = \begin{bmatrix} 1 & & & \\ & & & 1 \\ & & \iddots & \\ & 1 & & \end{bmatrix}. \quad (22)$$

The reversible core $\mathbf{B_N}$ is a permutation $\mathbf{T_N}$ followed by N rotations $\mathbf{\Lambda_N}(0.5,0)$, which can be implemented via the multiple factorization reversible rotation in (1)-(5). The float transform $\mathbf{A_N}$ consists of an inverse FFT, N rotations, and a vector addition[2]. 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 (21), with the computation complexity be four N/2-point complex FFT, N float rotations, and N/2 reversible rotations. It requires 4.5N roundings, with N roundings after each matrix lifting $\mathbf{A_N}$, $\mathbf{C_N}$ and $\mathbf{D_N}$, and 1.5N roundings for N/2 reversible rotations in $\mathbf{B_N}$.

3.2. Reversible MDCT via Matrix lifting

An N-point MDCT transform takes in a length-2N signal, and outputs N coefficients. It can be expressed as:

$$\mathbf{MDCT_{2N}}\mathbf{H_{2N}}, \quad (23)$$

where

$$\mathbf{MDCT_{2N}} = \left\{\sqrt{\frac{2}{N}}\cos\frac{\pi}{N}\left(i+\frac{1}{2}\right)\left(j+\frac{N+1}{2}\right)\right\}_{\substack{i=0,1,\cdots,N-1,\\ j=0,1,\cdots,2N-1.}}, \quad (24)$$

and $\mathbf{H_{2N}} = diag\{h(n)\}_{n=0,1,\cdots,2N-1.}$. (25)

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

$\mathbf{MDCT_{2N}}$ is the MDCT matrix, $h(n)$ is a window function. In MPEG audio, the window function $h(n)$ is:

$$h(n) = \sin\frac{\pi}{2N}(n+0.5). \quad (26)$$

According to [4], the MDCT transform in (23) can be calculated via a type-IV discrete sine transform (DST) shown in Figure 3[3].

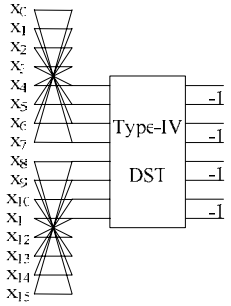

Figure 3 MDCT via type-IV DST.

The input signal is first grouped into pairs $x(n)$ and $x(N-n)$, $x(N+n)$ and $x(2N-n)$. Each pair is then treats as a complex number and rotates according to an angle specified by $h(n)$. We call this the window rotation. The middle section of the signal is then transformed by a type-IV DST with:

$$\mathbf{DSTIV_N} = \left[\sqrt{\frac{2}{N}}\sin\left(\frac{\pi}{N}(i+0.5)(j+0.5)\right)\right]_{i,j=0,1,\cdots,N-1}, \quad (27)$$

A 2N-point type-IV DST can be further converted to an N-point fractional-shifted FFT with $\alpha=\beta=0.25$, as:

$$\mathbf{DSTIV_{2N}} = \mathbf{P_{2N}}\mathbf{F_N}(\mathbf{0.25,0.25})\mathbf{Q_{2N}^{DST}}\mathbf{P_{2N}}, \quad (28)$$

where:

$$\mathbf{P_{2N}} = \left[p_{i,j}\right], \text{with } p_{i,j} = \begin{cases} 1 & i = j \text{ and } i \text{ is even} \\ 1 & i = 2N-j \text{ and } i \text{ is odd} \\ 0 & \text{otherwise} \end{cases}, \quad (29)$$

$$\mathbf{Q_{2N}^{DST}} = \begin{bmatrix} \begin{bmatrix} & 1 \\ 1 & \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} & 1 \\ 1 & \end{bmatrix} \end{bmatrix}, \quad (30)$$

After permutation operation $\mathbf{P_{2N}}$ and $\mathbf{Q_{2N}^{DST}}$, each pair of signal is converted to a complex number, and feed into the fractional-shifted FFT $\mathbf{F_N(\alpha,\beta)}$ in the form of:

$$\mathbf{F_N}(\alpha,\beta) = \frac{1}{\sqrt{N}}\left[w_N^{(i+\alpha)(j+\beta)}\right]_{i,j=0,1,\cdots,N-1}, \quad (31)$$

where α and β are shifting parameters. We may interpret each complex multiplication of the fractional-shifted FFT as a 2x2 rotation matrix of (1).

Because

$$\mathbf{F_N}(\alpha,\beta) = \mathbf{\Lambda_N}(\beta,\alpha)\mathbf{F_N}\mathbf{\Lambda_N}(\alpha,0), \quad (32)$$

the fractional-shifted FFT can be implemented via pre-rotation $\mathbf{\Lambda_N}(\alpha,0)$, FFT $\mathbf{F_N}$ and post-rotation $\mathbf{\Lambda_N}(\beta,\alpha)$. By implementing the pre- and post- rotations and window $h(n)$ rotations with reversible rotations (1)-(5), and implementing the reversible FFT through Section 3.1, we may implement a re-

[3] MDCT may be calculated via type-IV DCT, with alternative window folding and no sign change. Type-IV DCT may be calculated via an inverse fractional-shifted FFT of $\alpha=\beta=0.25$.

versible MDCT. Implementing an N-point MDCT this way requires 6.75N roundings.

Nevertheless, it is possible to implement the reversible MDCT with less roundings, and with about the same computation complexity. We observe that the fractional shifted FFT has the following properties:

Theory 2. Fractional-shifted FFT $\mathbf{F_N}(\alpha,\beta)$ is orthogonal.

Proof: The inner product of any two vectors of $\mathbf{F_N}(\alpha,\beta)$ is:

$$\frac{1}{N}\sum_j w_N^{-(i+\alpha)(j+\beta)} \cdot w_N^{+(k+\alpha)(j+\beta)} = \frac{1}{N} w_N^{(k-i)\beta} \sum_j w_N^{(k-i)j} = \delta(k-i). \tag{33}$$

Corollary 2. The inverse fractional-shifted FFT is

$$\mathbf{F}_N^{-1}(\alpha,\beta) = \frac{1}{\sqrt{N}} \left[w_N^{-(i+\beta)(j+\alpha)} \right]_{i,j=0,1,\cdots,N-1}. \tag{34}$$

Thoery 3:

$$\mathbf{R_N}(\alpha) = \mathbf{F_N}(\alpha,\alpha)\mathbf{\Lambda_N}(-2\alpha,\alpha)\mathbf{F_N}(\alpha,\alpha), \tag{35}$$

where $\mathbf{R_N}(\boldsymbol{\alpha})$ is a permutation matrix with only element (0,0) and elements (*i,N-i), i*=1,⋯,*N*-1 are non-zero.

Proof: Let $\mathbf{R_N}(\boldsymbol{\alpha}) = [r_N(i,k)]_{i,k=0,1,\cdots,N-1}$, then:

$$r_N(i,k) = \frac{1}{N}\sum_{j=0}^{N-1} W_N^{(i+\alpha)(j+\alpha)} W_N^{(2\alpha)(j+\alpha)} W_N^{(j+\alpha)(k+\alpha)} = W_N^{(i+k)\alpha}\delta(i+k). \tag{36}$$

To derive a reversible transform for the fractional-shifted FFT with α=β=0.25, we again use the radix-2 FFT structure. Noticing theory 3, we factor the fractional shifted FFT as follows:

$$\mathbf{F_{2N}}(\boldsymbol{\alpha},\boldsymbol{\beta}) = \mathbf{K_{2N}}\mathbf{S_{2N}}\mathbf{O}\mathbf{E_{2N}}, \tag{37}$$

with:

$$\mathbf{K_{2N}} = \begin{bmatrix} \mathbf{\Lambda_N}((1+\beta)/2-\alpha,\alpha) & \\ & W_2^{\beta}\mathbf{\Lambda_N}((1+\beta)/2-\alpha,\alpha) \end{bmatrix}, \tag{38}$$

$$\text{and } \mathbf{S_{2N}} = \begin{bmatrix} \frac{1}{\sqrt{2}}\mathbf{\Lambda_N}(-1/2,\alpha)\mathbf{F_N}(\alpha,\alpha) & \frac{1}{\sqrt{2}}\mathbf{F_N}(\alpha,\alpha) \\ \frac{1}{\sqrt{2}}\mathbf{\Lambda_N}(-1/2,\alpha)\mathbf{F_N}(\alpha,\alpha) & -\frac{1}{\sqrt{2}}\mathbf{F_N}(\alpha,\alpha) \end{bmatrix}. \tag{39}$$

Substitute α=0.25 and expanding fractional-shifted FFT via (32), we factor the transform $\mathbf{S_{2N}}$ into the matrix lifting form of (7), with:

$$\begin{cases} \mathbf{A_N} = -\sqrt{2}\mathbf{\Lambda_N}(-0.25,0.25)\mathbf{F_N^T}\mathbf{\Lambda_N}(-0.25,0) - \mathbf{I_N}, \\ \mathbf{B_N} = -\mathbf{R_N}(0.25) = \begin{bmatrix} -1 & & \\ & \ddots & j \\ & j & \end{bmatrix}, \\ \mathbf{C_N} = -\frac{1}{\sqrt{2}}\mathbf{\Lambda_N}(-0.25,0.25)\mathbf{F_N^T}\mathbf{\Lambda_N}(0.25,0.5), \\ \mathbf{D_N} = \mathbf{\Lambda_N}(-0.25,0.25)\left(\sqrt{2}+\mathbf{F_N^T}\mathbf{\Lambda_N}(-0.5,0.125)\right)\mathbf{F_N}\mathbf{\Lambda_N}(0.25,0). \end{cases} \tag{40}$$

In (40), the reversible core $\mathbf{B_N}$ is simply permutation, as multiplying by *j* is just swapping the real and imaginary part and changing the sign of the imaginary part. Using (40), an N-point MDCT can be implemented via N/2 reversible window rotations of *h*(*n*), a reversible matrix lifting of $\mathbf{S_{2N}}$, and N/2 reversible rotations of $\mathbf{K_{2N}}$. The total computational complexity is N reversible rotations, four N/4-point FFT, and 1.75N float rotations used in (40). The implementation complexity is about double of a float MDCT, which requires two N/4-point FFT, and 1.5N float rotations. Altogether, the reversible MDCT requires 4.5N roundings, with three 0.5N roundings after each matrix lifting of (40), and 3N roundings for the N reversible rotations.

4. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed reversible MDCT, we put the reversible MDCT module into a progressive to lossless embedded audio codec (PLEAC). We then compare the following reversible MDCT implementations:

a) With matrix lifting described in Section 3.2.

b) With scalar 3-step lifting of [1].

All the other modules of the PLEAC codec are the same, and are described in [1]. We first compare the output difference of the reversible MDCT module versus that of a float MDCT module, in terms of the mean square error (MSE), the mean absolute difference (MAD) and the peak absolute difference (PAD). We then compare the lossless compression ratio of the two codecs. Finally, we compare the progressive coding performance, by truncating the losslessly compressed bitstream to a lossy compressed audio bitstream of 64, 32 and 16kbps, 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 test audio waveform is the MPEG-4 sound quality assessment materials (SQAM). The aggregated comparison results are shown in Table 1.

Table 1 Comparison of different reversible MDCT modules.

Reversible MDCT	Matrix Lifting	Reversible Rotations
MSE	0.48	1.78
MAD	0.53	1.04
PAD	11.86	18.32
Lossless compression ratio	2.88:1	2.77:1
Lossy NMR (64kbps)	-2.18	-0.06
Lossy NMR (32kbps)	2.26	3.63
Lossy NMR (16kbps)	5.41	6.11

It is observed that with matrix lifting, the output of the reversible MDCT becomes much closer to the float MDCT, with MSE reduces by 73%. The improvement of the reversible MDCT also results in better lossless and lossy compression performance, as the lossless coding rate is reduced by 4%, and the lossy decoding NMR is increased by an average of 1.4dB. As a benchmark, the Monkey's Audio, the current state-of-the-art in lossless audio compression, compresses the test audio with a compression ratio of 2.93:1. PLEAC is thus within 2% of the lossless performance of the Monkey's Audio. Yet, the compressed bitstream of PLEAC can be flexibly and continuously scaled.

5. ACKNOWLEDGEMENTS

The author wishes to acknowledge H. S. Malvar and J. D. Johnston, for their insightful comments and discussions.

6. REFERENCES

[1] J. Li, "A progressive to lossless embedded audio coder (PLEAC) with reversible modulated lapped transform", in *Proc. of ICASSP '03*, Vol. 5, pp. 413-416, Hong Kong, China.

[2] R. Geiger, J. Herre, J. Koller, and K. Brandenburg, "IntMDCT - A link between perceptual and lossless audio coding," in *Proc. of ICASSP 2002*, Orlando, 2002.

[3] J. Wang, J. Sun and S. Yu, "1-D and 2-D transforms from integers to integers", in *Proc. of ICASSP'03, Hong Kong, China.*

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