Reversible Integer-to-Integer Wavelet Transforms With Improved Approximation Properties

Peter W. van Vugt and Michael D. Adams
Dept. of Elec. and Comp. Eng., University of Victoria, Victoria, BC, Canada

***Abstract*— Reversible integer-to-integer (ITI) transforms that approximate linear wavelet transforms have proven extremely useful in signal coding applications. The coding efficiency achievable with such reversible transforms, however, depends on how well they approximate their parent linear transforms. In this paper, a simple but effective method for constructing reversible ITI wavelet transforms which better approximate their parent linear transforms is proposed. Furthermore, the transforms obtained with this new method are shown to have superior performance for signal coding applications.**

I. Introduction

Reversible integer-to-integer (ITI) wavelet transforms have proven to be extremely useful in both lossless and lossy signal coding applications, finding use in many practical systems (e.g., the JPEG-2000 codec [1]). Such transforms are invertible in finite-precision arithmetic (i.e., reversible), map integers to integers (i.e., integer-to-integer) and approximate linear wavelet transforms. It has been suggested [2] that the approximation characteristics of reversible ITI transforms play an important role in determining the effectiveness of such transforms for signal coding applications. In practice, one usually starts with a linear wavelet transform having certain desirable properties from a signal coding perspective. Then, one proceeds to find a (nonlinear) reversible ITI approximation of this parent linear transform. In order for the desirable properties of the parent transform to be preserved in its reversible ITI version, this reversible version must closely approximate the parent transform. In this paper, we propose a method for constructing reversible ITI approximations of linear wavelet transforms with reduced approximation error. Moreover, we demonstrate that the transforms constructed using our new method offer superior performance in signal coding applications.

The remainder of this paper is structured as follows. To begin, Section II briefly introduces some of the notation used herein. The most common method for constructing reversible ITI wavelet transforms is introduced in Section III, along with its shortcomings. Then, Section IV proposes a new approach for constructing reversible ITI wavelet transforms that yields transforms with improved approximation properties. Next, some experimental results are presented in Section V that demonstrate the effectiveness of our proposed method. Finally, in Section VI, we conclude with a summary of our work and some closing remarks.

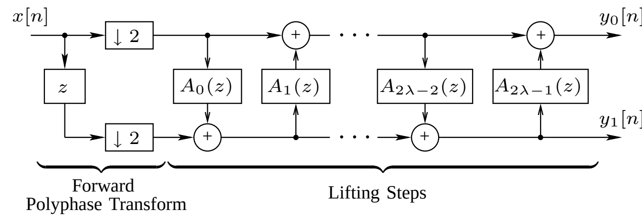

Fig. 1. Lifting realization of forward one-level wavelet transform.

II. Notation and Terminology

Before proceeding further, some brief comments are in order concerning the notation used herein. We denote the floor and ceiling of a real number x as $\lfloor x \rfloor$ and $\lceil x \rceil$, respectively. Matrices and vectors are denoted using uppercase and lowercase boldface letters, respectively. The $n \times n$ identity matrix is denoted $\boldsymbol{I}_n$. As a matter of terminology, a unit triangular matrix is a triangular matrix with all of its main-diagonal elements equal to one. A matrix $\boldsymbol{A}$ is said to be unimodular if $|\det \boldsymbol{A}| = 1$. For data with P-bits/sample, we define the peak-signal-to-noise ratio (PSNR) as $\text{PSNR} = 20 \log_{10}([2^P - 1]/\sqrt{\text{MSE}})$, where MSE denotes the mean squared error.

III. Reversible ITI Wavelet Transforms

To date, the most popular technique for constructing reversible ITI wavelet transforms is that proposed by Calderbank et al. in [3]. We shall, henceforth, refer to this approach as the conventional lifting (CL) method. With this method, one first finds a lifting realization [4] of the linear wavelet transform of interest. Then, rounding operations are carefully introduced in order to obtain a reversible ITI mapping.

The lifting realization of a one-level wavelet transform is shown in Fig. 1. Here, only the structure for the forward transform is shown, as the inverse transform structure can be trivially deduced by symmetry. In short, the forward transform realization consists of a forward polyphase transform followed by a number of lifting (i.e., ladder) steps, where the filters $\{A_k\}$ are called lifting filters. In the case of a multi-level wavelet transform, the one-level structure (from above) is simply employed in a recursive manner.

With the CL method, a reversible ITI mapping is constructed from the lifting realization by inserting rounding operations at the output of each lifting filter, yielding lifting steps each of the form shown in Fig. 2, where Q denotes a rounding operator. In order to handle finite-length signals, symmetric extension [5] is usually employed. Since each of the lifting steps are

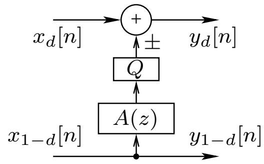

Fig. 2. Modified lifting step.

connected in series, the number of cascaded stages of rounding equals the number of lifting steps. (Here, by "cascaded", we mean that rounding operations are applied in succession, so that the rounding error from one calculation feeds into the next.) Thus, as the number of lifting steps increases, the approximation error introduced by rounding also increases. Since the number of lifting steps increases with the lengths of the associated analysis/synthesis filters as well as with the number of wavelet decomposition levels, approximation error grows as the transform becomes more "complex". It would be nice, however, if the rounding error could be made more or less independent of transform complexity. Fortunately, this goal is achievable, as we shall later demonstrate.

IV. Transforms With Improved Approximation Properties

Since any linear transform (in a N-dimensional space) can be characterized by an $N \times N$ matrix $\boldsymbol{A}$, we can describe the wavelet transform in terms of the equation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are N-dimensional vectors corresponding to the transform input and output, respectively. In what follows, we would like to examine the lifting realization of the CL method from this transform-domain (rather than polyphase-domain) perspective. As we shall see, this will allow us to gain some new and useful insights into the CL method, as well as develop a means by which to construct reversible ITI wavelet transforms with improved approximation properties.

Before proceeding further, a brief digression is needed in order to introduce three new types of matrices and some of their properties. First, we denote as $\mathcal{P}(n)$ the $n \times n$ permutation matrix that, when applied to the n-dimensional vector $\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n-1} \end{bmatrix}$, yields the vector $\begin{bmatrix} x_0 & x_2 & x_4 & \cdots & x_{2\lceil n/2 \rceil - 2} & x_1 & x_3 & x_5 & \cdots & x_{2\lfloor n/2 \rfloor - 1} \end{bmatrix}$ (i.e., the transformed vector has its elements sorted such that the even-indexed elements appear in ascending order first, followed by the odd-indexed elements in ascending order). Second, we denote as $\mathcal{L}(n, s, \boldsymbol{F})$, the $n \times n$ unit triangular matrix

$$\mathcal{L}(n, s, \boldsymbol{F}) = \begin{cases} \begin{bmatrix} \boldsymbol{I}_{\lceil n/2 \rceil} & \boldsymbol{F} \\ \boldsymbol{0} & \boldsymbol{I}_{\lfloor n/2 \rfloor} \end{bmatrix} & \text{for } s = 0 \\ \begin{bmatrix} \boldsymbol{I}_{\lceil n/2 \rceil} & \boldsymbol{0} \\ \boldsymbol{F} & \boldsymbol{I}_{\lfloor n/2 \rfloor} \end{bmatrix} & \text{for } s = 1, \end{cases}$$

where $\boldsymbol{F}$ is evidently $\lceil n/2 \rceil \times \lfloor n/2 \rfloor$ if $s = 0$ or $\lfloor n/2 \rfloor \times \lceil n/2 \rceil$ if $s = 1$. Third, we denote as $\mathcal{S}(n, k, \boldsymbol{v})$, the $n \times n$ matrix with all main-diagonal elements equal to one, and all off-diagonal elements equal to zero, except those in the kth row which are given (in order) by the elements of the $(n-1)$-dimensional vector $\boldsymbol{v}$. One can easily show that $\det \mathcal{L}(\cdot, \cdot, \cdot) = 1$. Also, we have the following result regarding the determinant of $\mathcal{P}(\cdot)$.

Theorem 1. *For any positive integer n, the permutation matrix $\mathcal{P}(n)$ has the determinant given by* $\det \mathcal{P}(n) = (-1)^{\lfloor (n+1)/4 \rfloor}$.

Proof. See Appendix. □

With the above digression complete, we are now in a position to revisit the lifting realization employed by the CL method. To begin, we consider a one-level wavelet decomposition. In this case, the (forward) transform has the computational structure shown in Fig. 1. Let N denote the block size of the transform. Consider now, the general form of the linear operator (i.e., $N \times N$ matrix) associated with this transform. From the diagram, we can see that the first action of the transform is to apply a forward polyphase transform, which is equivalent to the permutation operator $\mathcal{P}(N)$. Next, a number of lifting steps are successively applied. Each lifting step, however, corresponds to a linear operator of the form $\mathcal{L}(N, \cdot, \cdot)$. This is true regardless of how the lifting filters are adjusted when filtering near signal boundaries (in order to accommodate finite-length signals). Thus, a one-level wavelet transform is characterized by the $N \times N$ matrix $\boldsymbol{B}$ given by

$$\boldsymbol{B} = \boldsymbol{L}_{2\lambda-1} \cdots \boldsymbol{L}_1 \boldsymbol{L}_0 \boldsymbol{P}, \tag{1}$$

where $\boldsymbol{P} = \mathcal{P}(N)$ and $\boldsymbol{L}_k \in \mathcal{L}(N, \cdot, \cdot)$ for $k = 0, 1, \ldots, 2\lambda - 1$. From Theorem 1, one can easily conclude that

$$\det \boldsymbol{B} = \det \mathcal{P}(n) = (-1)^{\lfloor (n+1)/4 \rfloor}. \tag{2}$$

Now, let us consider the case of an L-level wavelet decomposition (where $L \geq 1$). Since an L-level wavelet transform can be constructed by the recursive application of a one-level transform, we have that an L-level transform is characterized by the $N \times N$ matrix $\boldsymbol{A}$ given by

$$\boldsymbol{A} = \prod_{k=0}^{L-1} \begin{bmatrix} \boldsymbol{B}_k & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_{N-n_k} \end{bmatrix}, \tag{3}$$

where $\boldsymbol{B}_k$ is a $n_k \times n_k$ matrix of the form of (1), $n_0 = N$, and $n_k = \lceil n_{k-1}/2 \rceil$ for $k = 1, 2, \ldots, L-1$. Using (3) and (2), one can deduce that

$$\det \boldsymbol{A} = \prod_{k=0}^{L-1} \det \boldsymbol{B}_k = \prod_{k=0}^{L-1} \Delta(n_k), \tag{4}$$

where $\Delta(n) = (-1)^{\lfloor (n+1)/4 \rfloor}$ and n_k is as defined above. The above results are interesting as they formally show that the transform matrix $\boldsymbol{A}$ is always unimodular and furthermore provide a means to calculate the determinant of the matrix.

So far in this section, we have considered only the linear versions of wavelet transforms. Now, we consider how the process of creating reversible ITI mappings works when viewing the CL method from this new transform-domain perspective. Consider the N-input N-output network shown in Fig. 3, where the $\{s_k\}$ are amplifier gains and Q denotes a rounding operator. We shall refer to this network as a transform-domain-lifting (TDL) step. In the absence of Q, we simply have the linear system with transfer matrix $\mathcal{S}(N, j, \boldsymbol{s})$, where $\boldsymbol{s} = \begin{bmatrix} s_0 & \cdots & s_{j-1} & s_{j+1} & \cdots & s_{N-1} \end{bmatrix}$. When Q is included, however, we have a reversible ITI network. Now, consider (1). In this

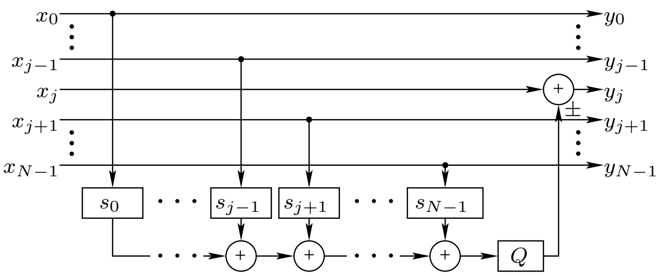

Fig. 3. Network for transform-domain lifting (TDL) step.

equation, each $\boldsymbol{L}_k$ can be trivially decomposed into a product of (approximately) $\frac{N}{2}$ matrices from $\mathcal{S}(N,\cdot,\cdot)$, where each factor corresponds to one of the rows of $\boldsymbol{L}_k$ having nonzero off-diagonal values. Since, as was explained above, each $\mathcal{S}(N,\cdot,\cdot)$ matrix is associated with a reversible ITI network (i.e., a TDL step), we can create a reversible ITI wavelet transform, by mapping each $\boldsymbol{L}_k$ matrix to a sequence of TDL steps. Consequently, a reversible ITI mapping derived in this way will employ (approximately) $\frac{NL\lambda}{2}$ rounding operations in total, or equivalently $\frac{L\lambda}{2}$ rounding operations per input sample.

A. Building Better Transforms

Suppose that we have a linear wavelet transform that can be converted to a reversible ITI mapping via the CL method. In the previous section, we showed that such a transform is represented by a matrix $\boldsymbol{A}$ of the form of (3), and furthermore that $\boldsymbol{A}$ is unimodular. In what follows, we consider two alternative factorizations of $\boldsymbol{A}$ that can be used to construct reversible ITI mappings. In particular, we consider the triangular elementary reversible matrix (TERM) and single-row elementary reversible matrix (SERM) factorizations proposed by Hao and Shi [6].

One can show [6] that any real unimodular $N \times N$ matrix $\boldsymbol{A}$ has a factorization of the form

$$\boldsymbol{A} = \boldsymbol{PLDUS}, \tag{5}$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{L}$ and $\boldsymbol{U}$ are lower and upper unit triangular matrices, respectively, $\boldsymbol{D} = \left[\begin{smallmatrix} \boldsymbol{I}_{N-1} & \boldsymbol{0} \\ \boldsymbol{0} & d \end{smallmatrix}\right]$, $d = (\det \boldsymbol{A})/(\det \boldsymbol{P})$, and $\boldsymbol{S} \in \mathcal{S}(N, N-1, \cdot)$. We refer to (5) as a TERM factorization of $\boldsymbol{A}$. Since $\boldsymbol{L}$ and $\boldsymbol{U}$ are both unit triangular, each of these matrices can be further decomposed into a product of $N-1$ matrices in $\mathcal{S}(N,\cdot,\cdot)$ (i.e., TDL steps). In this way, each of the $\boldsymbol{L}, \boldsymbol{U}$, and $\boldsymbol{S}$ matrices can be converted to TDL steps in order to obtain a reversible ITI mapping. Furthermore, a reversible ITI mapping derived in this way will have a total of $2N-1$ rounding operations, or equivalently $\frac{2N-1}{N}$ rounding operations per input sample (regardless of the complexity of the parent linear transform).

One can also show [6] that any real unimodular $N \times N$ matrix $\boldsymbol{A}$ has a factorization of the form

$$\boldsymbol{A} = \boldsymbol{P}\boldsymbol{S}_N\boldsymbol{S}_{N-1}\ldots\boldsymbol{S}_1\boldsymbol{S}_0, \tag{6}$$

where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{S}_k \in \mathcal{S}(N,\cdot,\cdot)$ for $k = 0, 1, \ldots, N$. We refer to (5) as a SERM factorization of $\boldsymbol{A}$. Since each $\boldsymbol{S}_k$ matrix can be mapped to a TDL step, a reversible ITI mapping derived from a SERM factorization will consist of a permutation and $N+1$ TDL steps. Consequently, a SERM-based ITI mapping will have a total of $N+1$ rounding operations, or equivalently $\frac{N+1}{N}$ rounding operations per input sample (regardless of the complexity of the parent linear transform).

Recall that the number of rounding operations per input sample required in the case of CL-, TERM-, and SERM-based reversible ITI transforms is given by $\frac{L\lambda}{2}$, $\frac{2N-1}{N}$, and $\frac{N+1}{N}$, respectively, where N is the transform block size, L is the number of decomposition levels, and λ is the number of lifting steps (which is indirectly related to the complexity of the transform). Observe that the number of rounding operations in the TERM and SERM cases does not depend on the transform complexity (i.e., λ and L). Furthermore, in practice, we typically have $N \geq 128$, $L > 1$, and $\lambda \geq 2$. In such a scenario, however, the TERM- and SERM-based transforms require fewer rounding operations relative to the CL method. This leads us to suspect that TERM- and SERM-based transforms will have superior approximation properties. In fact, reversible ITI wavelet transforms constructed using our SERM/TERM-based method are superior, as we shall soon demonstrate.

V. Experimental Results

Above, we claimed that the reversible ITI transforms constructed using our SERM/TERM-based design method have superior performance. In what follows, we will substantiate this claim through experimental results. Although our method has potential application to many types of signals, we have elected to consider audio data herein. Throughout our experiments, six audio signals taken from the MATLAB Audio Toolbox were used as test data (i.e., chirp, gong, handel, laughter, splat, and train). The sample data was converted by scaling and rounding to 8-bit signed integer values, and the various audio sequences varied from approximately 10000 to 75000 samples in length. Since these sequences are relatively long, they are transformed in a blockwise manner in all of the subsequent experiments. In our experiments, we also often consider the 9/7 wavelet transform [1], as this transform is known to perform particularly well in signal coding applications.

A. Finding Good Factorizations

For a given $N \times N$ matrix $\boldsymbol{A}$, the factorizations defined by (5) and (6) are not unique, with the number of solutions growing explosively with increasing N. Since different factorizations yield distinct reversible ITI transforms and the approximation properties of these transforms can differ greatly, some means is needed for selecting good factorizations. In our work, we have computed SERM and TERM factorizations using the algorithms described in [6]. Of particular note, however, we found that the heuristic briefly mentioned at the end of Section VII in [6] is particularly effective at producing transforms with good approximation characteristics.

TABLE I

COMPARISON OF THE THEORETICAL UPPER BOUND ON PAE FOR THE 9/7 WAVELET TRANSFORM WITH A BLOCK SIZE OF 12 (TERM CASE)

Levels	Optimal	Heuristic
1	1.827	1.842
2	1.942	2.004

The effectiveness of the aforementioned heuristic was confirmed as follows. For a given transform matrix $\boldsymbol{A}$, we first computed a factorization using the heuristic. Then, we exhaustively searched all possible factorizations for the one with the best approximation goodness, where the goodness metric being employed is an upper bound on the theoretical PAE. We then compared the goodness metric of the transform obtained with the heuristic to the global optimum obtained from the exhaustive search. This process was repeated for numerous wavelet transforms, and the heuristic method was observed to consistently produce nearly optimal results. As an example, the results obtained in the cases of one- and two-level 9/7 wavelet transforms (with a block size of 12) are shown in Table I. In both of these cases, the goodness metric of the transform obtained using the heuristic was within 3.5% of the global optimum. Although such an experiment can only be performed for small N (in order to avoid computational intractability problems), it still provides some indication that this heuristic-driven factorization method is quite effective. Due to its effectiveness, we consistently employ the heuristic-based factorization method throughout the remainder of this paper.

B. Forward Transform Approximation Goodness

Now, we shall demonstrate the effectiveness of our proposed SERM/TERM-based method for constructing reversible ITI transforms with reduced approximation error. In what follows, we consider three different reversible ITI versions of the one-level 9/7 wavelet transform (with a block size of 512), one generated with each of the CL-, TERM-, and SERM-based approaches. For each of the reversible ITI transforms, we applied the forward transform to an input (audio) sequence and measured the difference between the resulting transform coefficients and those obtained with the parent linear transform. The PAE and PSNR in each case are given in Table II(a). As an additional point of reference, in the column labelled "Round," we have indicated the error incurred if one simply rounds (to integer values) all of the coefficients of the linear transform. This provides an approximate bound on the best error performance that we can ever hope to achieve. For the sake of interest, we have also provided a (loose) theoretical upper bound on the PAE for each transform. The same process from above was then repeated, except for a five-level wavelet decomposition, yielding the results shown in Table II(b).

From the results in Table II, we can see that overall, in terms of both PAE and PSNR, the SERM-based transform has the best approximation error performance, followed by the TERM- and CL-based transforms (in that order). Furthermore, by comparing the numbers in Tables II(a) and II(b), we can observe that the difference in performance between the SERM/TERM-based transforms and the CL-based transform grows as the number of wavelet decomposition levels increases. Since, in practice, one typically employs several levels of wavelet decomposition, this further emphasizes the strength of our method.

TABLE II

COMPARISON OF THE APPROXIMATION ERROR FOR A (A) ONE-LEVEL AND (B) FIVE-LEVEL 9/7 WAVELET TRANSFORM WITH A BLOCK SIZE OF 512.

(a)

	PAE			PSNR			
Signal	CL	TERM	SERM	CL	TERM	SERM	Round
chirp	1.769	1.551	1.197	54.90	54.94	56.92	59.95
gong	2.012	1.528	1.180	53.21	54.73	56.63	58.94
handel	2.008	1.627	1.357	52.67	54.64	56.65	58.93
laughter	2.059	1.564	1.151	52.66	54.68	56.70	58.92
splat	1.673	1.563	1.117	56.38	56.08	57.39	60.70
train	1.955	1.571	1.146	52.58	54.55	56.69	58.93
Mean	1.913	1.567	1.191	53.73	54.94	56.83	59.40
Theoretical†	2.806	1.842	1.722				

†Theoretical upper bound on PAE

(b)

	PAE			PSNR			
Signal	CL	TERM	SERM	CL	TERM	SERM	Round
chirp	4.456	3.230	2.154	52.42	54.62	57.64	59.91
gong	5.958	2.949	2.497	50.37	54.54	56.58	58.91
handel	5.386	2.674	2.813	49.95	54.59	56.60	58.92
laughter	5.228	2.993	2.833	49.95	54.60	56.53	58.93
splat	4.043	2.034	2.216	53.23	55.76	58.34	60.61
train	4.089	2.366	2.352	49.87	54.61	56.66	58.89
Mean	4.860	2.708	2.478	50.97	54.79	57.06	59.36

C. Transform Lossy Coding Performance

As indicated previously, reversible ITI transforms with improved approximation characteristics have the potential to offer superior performance in signal coding applications. In what follows, we present some experimental results in order to quantify the achievable performance gain.

In this experiment, we considered the same three reversible ITI versions of the five-level 9/7 wavelet transform that were examined previously (in Section V-B). For each transform, we applied the forward transform to an audio signal, quantized the resulting data (using scalar quantization), dequantized, and then applied the corresponding inverse transform. In each case, the distortion (i.e., difference between the reconstructed and original signals) was measured using both PAE (i.e., peak absolute error) and PSNR. This process was repeated for each of the audio signals in our test set. In order to facilitate a meaningful comparison, the same set of quantizer step sizes was employed for all transforms.

The results of the above process are shown in Table III. As an additional point of reference, we provide, in the column labelled "Linear", the PAE and PSNR obtained by simply using the linear version of the transform in the above coding system. This quantity provides an approximate indication of the best performance that we can hope to achieve with reversible ITI transforms. From the results in Table III, we

TABLE III

COMPARISON OF THE LOSSY CODING PERFORMANCE FOR A FIVE-LEVEL 9/7 WAVELET TRANSFORM WITH A BLOCK SIZE OF 512.

	PAE				PSNR			
Signal	CL	TERM	SERM	Linear	CL	TERM	SERM	Linear
chirp	9	7	7	7	44.21	44.84	45.17	45.51
gong	11	9	9	8	40.84	41.93	42.10	42.47
handel	10	10	10	8	40.40	41.63	41.80	42.16
laughter	11	11	9	10	40.34	41.42	41.59	41.87
splat	9	7	8	8	44.26	45.00	45.18	45.49
train	9	7	7	7	40.86	42.11	42.26	42.53
Mean	9.83	8.50	8.33	8.00	41.82	42.82	43.02	43.34

can clearly see that, in terms of both PAE and PSNR, the SERM-based transform performs best, followed by the TERM- and CL-based transforms (in that order). In terms of PAE, the SERM- and TERM-based transforms offer an average improvement of about 1.5 and 1.33, respectively, over the CL-based transform. In terms of PSNR, the SERM- and TERM-based transforms yield an average improvement of about 1.2 dB and 1 dB, respectively, over the CL-based transform. From the above results, it is evident that the transforms obtained with our SERM/TERM-based method have superior performance for lossy coding purposes. Furthermore, of our SERM- and TERM-based approaches, the former is clearly the best.

VI. CONCLUSIONS

In this paper, we have proposed a new method for constructing reversible ITI versions of wavelet transforms based on SERM and TERM factorizations. In particular, we showed that, given any linear wavelet transform for which a reversible ITI version can be derived using the CL method, it is also possible to generate reversible ITI versions using SERM and TERM factorizations. We then proceeded to show that the resulting SERM- and TERM-based reversible ITI transforms have better approximation properties than those produced by the CL method. Furthermore, we showed that these improved approximation properties directly translate into superior performance for lossy audio coding. Of the SERM- and TERM-based approaches introduced herein, the SERM-based approach was shown to produce the best transforms. Clearly, the transforms obtained with our new method can offer significant benefits for audio and other signal coding applications.

APPENDIX

Proof of Theorem 1. Consider the quantity $\det \mathcal{P}(n)$. One can easily confirm that $\det \mathcal{P}(n) = 1$ for $n \in \{1, 2\}$. For $n \geq 3$, we implicitly construct the permutation $\mathcal{P}(n)$ by determining a sequence of exchange operations that, when applied to the n-dimensional vector $\boldsymbol{x} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{n-1} \end{bmatrix}$, yields the desired permuted vector $\boldsymbol{x}' = \begin{bmatrix} x_0 & x_2 & x_4 & \cdots & x_{2\lceil n/2 \rceil - 2} & x_1 & x_3 & x_5 & \cdots & x_{2\lfloor n/2 \rfloor - 1} \end{bmatrix}$.

We produce $\boldsymbol{x}'$ from $\boldsymbol{x}$ by way of an iterative algorithm as follows. We partition the elements of $\boldsymbol{x}$ into two subblocks, denoted as $\boldsymbol{s}$ and $\boldsymbol{u}$ (i.e., $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{s} & \boldsymbol{u} \end{bmatrix}$). The first subblock $\boldsymbol{s}$ contains elements that are correctly sorted relative to one another with respect to the desired permutation $\mathcal{P}(n)$, while the second subblock $\boldsymbol{u}$ contains elements yet to be sorted. We begin with $\boldsymbol{x}$ partitioned as shown in Fig. 4(a). Each iteration of the algorithm then moves the first element in $\boldsymbol{u}$ to its correct position amongst elements in $\boldsymbol{s}$ through a sequence of exchange operations. We have one of two cases, depending on whether the index i of the element to be moved is odd or even. These two cases are illustrated in Figs. 4(b) and 4(c), respectively, where $k = \left\lfloor \frac{i-1}{2} \right\rfloor$ and the boxed element is the one to be moved from $\boldsymbol{u}$ to its correct relative position in $\boldsymbol{s}$.

First, let us suppose that i is odd (i.e., $i = 2k+1$), which corresponds to the scenario shown in Fig. 4(b). In this case, the ith element is already in its correct position. Thus, no exchanges are required. Now, let us suppose that i is even (i.e., $i = 2k+2$), corresponding to the scenario in Fig. 4(c). In this case, we can move the ith element into its correct position by successively exchanging it with each of the odd-indexed elements to its left, which requires $k+1 = \left\lfloor \frac{i+1}{2} \right\rfloor$ exchanges. Since this algorithm iterates for $i = 1, 2, \ldots, n-1$, the total number of exchanges required is $c = \sum_{i=1, i \text{ even}}^{n-1} \left\lfloor \frac{i+1}{2} \right\rfloor = \frac{1}{2} \left\lceil \frac{n}{2} \right\rceil (\left\lceil \frac{n}{2} \right\rceil - 1)$. Since $\mathcal{P}(n)$ is the product of c exchange matrices, we have that $\det \mathcal{P}(n) = (-1)^c$ which can be simplified to $\det \mathcal{P}(n) = (-1)^{\lfloor (n+1)/4 \rfloor}$. $\square$

$$\big[\ \overbrace{x_0}^{\boldsymbol{s}}\ \overbrace{\boxed{x_1}\ \ x_2\ \ \cdots\ \ x_{n-1}}^{\boldsymbol{u}}\ \big]$$

(a)

$$\big[\overbrace{\underbrace{x_0\ \ x_2\ \ \cdots\ \ x_{2k}}_{(k+1)\text{ elements}}\ \underbrace{x_1\ \ x_3\ \ \cdots\ \ x_{2k-1}}_{k\text{ elements}}}^{\boldsymbol{s}}\ \overbrace{\boxed{x_{2k+1}}\ \ x_{2k+2}\ \ \cdots\ \ x_{n-1}}^{\boldsymbol{u}}\big]$$

(b)

$$\big[\overbrace{\underbrace{x_0\ \ x_2\ \ \cdots\ \ x_{2k}}_{(k+1)\text{ elements}}\ \underbrace{x_1\ \ x_3\ \ \cdots\ \ x_{2k+1}}_{(k+1)\text{ elements}}}^{\boldsymbol{s}}\ \overbrace{\boxed{x_{2k+2}}\ \ x_{2k+3}\ \ \cdots\ \ x_{n-1}}^{\boldsymbol{u}}\big]$$

(c)

Fig. 4. Incremental sorting process. (a) Initial state. The case of handling of an (b) odd-indexed and (c) even-indexed element.

REFERENCES

[1] *ISO/IEC 15444-1: Information technology—JPEG 2000 image coding system—Part 1: Core coding system*, 2000.

[2] M. D. Adams and F. Kossentini, “Reversible integer-to-integer wavelet transforms for image compression: Performance evaluation and analysis,” vol. 9, no. 6, pp. 1010–1024, June 2000.

[3] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, “Wavelet transforms that map integers to integers,” *Applied and Computational Harmonic Analysis*, vol. 5, no. 3, pp. 332–369, July 1998.

[4] I. Daubechies and W. Sweldens, “Factoring wavelet transforms into lifting steps,” *Journal of Fourier Analysis and Applications*, vol. 4, pp. 247–269, 1998.

[5] M. D. Adams and R. K. Ward, “Symmetric-extension-compatible reversible integer-to-integer wavelet transforms,” *IEEE Trans. on Signal Processing*, vol. 51, no. 10, pp. 2624–2636, Oct. 2003.

[6] P. Hao and Q. Shi, “Matrix factorizations for reversible integer mapping,” *IEEE Trans. on Signal Processing*, vol. 49, no. 10, pp. 2314–2324, Oct. 2001.