Generalized Reversible Integer-to-Integer Transform Framework

Michael D. Adams
Dept. of Elec. and Comp. Engineering, University of Victoria, Victoria, BC, V8W 3P6, CANADA
E-Mail: mdadams@ece.uvic.ca

***Abstract*— The generalized reversible ITI transform (GRITIT) framework, a single unified framework for reversible ITI wavelet/block transforms, is proposed. The relationship between the GRITIT framework and several previously proposed frameworks is also examined.**

I. Introduction

To date, several frameworks have been proposed for constructing reversible integer-to-integer (ITI) wavelet transforms, including the S+P-transform [1], lifting [2], and overlapping-rounding-transform (ORT) [3] frameworks. Although these frameworks share a number of common ideas, each framework also has its own distinctive features. Thus, no single one of these frameworks can be used to describe all of the others. This state of affairs, however, is rather unfortunate, as a single unified framework for reversible ITI transforms would be highly beneficial. Having such a tool at our disposal, we could more easily analyze different frameworks and their interrelationships.

In this paper, we propose the generalized reversible ITI transform (GRITIT) framework, a single unified framework for reversible ITI wavelet/block transforms. This framework combines the ideas behind several previously developed frameworks, extending some of these ideas in the process. In our work, we explicitly consider the more general case of D-dimensional M-band wavelet transforms (where $D \geq 1$ and $M \geq 2$). As we shall see later, the GRITIT framework is a powerful tool, and can be used to construct reversible ITI wavelet and block transforms.

II. GRITIT Framework

With the GRITIT framework, a transform is constructed as a sequence of primitive reversible ITI operations. Six distinct types of primitive operations are employed: the split, join, displace, exchange, shift, and scale operations. Furthermore, each type of primitive operation has the property that it is inverted by another primitive operation. Thus, if we have a transform T given by

$$T = T_{N-1} \cdots T_1 T_0, \tag{1}$$

where the $\{T_i\}$ are primitive operations, then the corresponding inverse transform is given by

$$T^{-1} = T_0^{-1} T_1^{-1} \cdots T_{N-1}^{-1}. \tag{2}$$

where the $\{T_i^{-1}\}$ are also primitive operations. In what follows, we will now describe the six types of primitive operations in more detail.

1) Split Operation: The split operation is associated with the 1-input M-output network shown in Figure 1(a). This operation decomposes the input signal $x[\boldsymbol{n}]$ into M polyphase components $\{y_i[\boldsymbol{n}]\}_{i=0}^{M-1}$, where the polyphase decomposition is performed with respect to the the sampling matrix $\boldsymbol{M}$ and corresponding coset vectors $\{\boldsymbol{m}_i\}_{i=0}^{M-1}$, and $M = |\det \boldsymbol{M}|$. As a matter of notation, we denote an operation of this type as $\mathcal{P}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}])$.

The split operation is linear and shift variant. Since the downsampling and shift operations are ITI, the split operation is also ITI in nature. The inverse of a split operation is a join operation (to be defined next). (More specifically, in terms of notation yet to be defined, $\mathcal{P}^{-1}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}]) = \mathcal{J}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}])$).

2) Join Operation: The join operation is associated with the M-input 1-output network shown in Figure 1(b). This operation simply synthesizes a signal $y[\boldsymbol{n}]$ from its M polyphase components $\{x_i[\boldsymbol{n}]\}_{i=0}^{M-1}$. As a matter of notation, we denote an operation of this type as $\mathcal{J}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}])$.

The join operation is linear. Since the upsampling, shift, and addition operations are ITI, the join operation is also ITI in nature. The inverse of a join operation is a split operation. More specifically, $\mathcal{J}^{-1}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}]) = \mathcal{P}(\boldsymbol{M}, [\boldsymbol{m}_0\ \boldsymbol{m}_1\ \cdots\ \boldsymbol{m}_{M-1}])$.

3) Displace Operation: The displace operation can be viewed as a generalization of the lifting operation employed in the lifting framework. The displace operation is associated with the M-input M-output network depicted in Figure 1(c). As a matter of notation, the network inputs and outputs are denoted as $\{x_i[\boldsymbol{n}]\}$ and $\{y_i[\boldsymbol{n}]\}$, respectively. In order to simplify the diagram, only the Kth output is shown. All of the other outputs are directly connected to their corresponding inputs (i.e., $y_i[\boldsymbol{n}] = x_i[\boldsymbol{n}]$ except for $i = K$). The Kth output (i.e., $y_K[\boldsymbol{n}]$) is generated by adding an adjustment value to the Kth input (i.e., $x_K[\boldsymbol{n}]$). This adjustment value is calculated by summing filtered versions of one or more inputs (i.e., $\{x_i[\boldsymbol{n}]\}_{i=0}^{M-1}$) and possibly the Kth output (i.e., $y_K[\boldsymbol{n}]$), and then rounding the result with the operator Q. If s is odd, the sign of the adjustment value is also inverted. Finally, the Kth output is formed by adding the adjustment value to the Kth input. Mathematically, we have

$$y_i[\boldsymbol{n}] = \begin{cases} x_i[\boldsymbol{n}] + (-1)^s Q\left(b[\boldsymbol{n}] * y_i[\boldsymbol{n}] + \sum_{l=0}^{M-1} a_l[\boldsymbol{n}] * x_l[\boldsymbol{n}]\right) & \text{for } i = K \\ x_i[\boldsymbol{n}] & \text{otherwise.} \end{cases}$$

As a matter of notation, we denote an operation of this type as

$$\mathcal{L}(K, Q, s, [A_0(\boldsymbol{z})\ A_1(\boldsymbol{z})\ \cdots\ A_{M-1}(\boldsymbol{z})], B(\boldsymbol{z})). \tag{3}$$

In order to avoid delay-free loops (which are physically unrealizable), $B(\boldsymbol{z})$ must have a constant term of zero. If no rounding operator is employed, Q is denoted as $\emptyset$ (e.g., as in $\mathcal{L}(1, \emptyset, 1, [0\ 1], 0)$). The inverse of a displace operation is another displace operation. More specifically, $\mathcal{L}^{-1}(K, Q, s, [A_0(\boldsymbol{z})\ A_1(\boldsymbol{z})\ \cdots\ A_{M-1}(\boldsymbol{z})], B(\boldsymbol{z})) = \mathcal{L}(K, Q, 1-s, [A_0'(\boldsymbol{z})\ A_1'(\boldsymbol{z})\ \cdots\ A_{M-1}'(\boldsymbol{z})], B'(\boldsymbol{z}))$ where

$$B'(\boldsymbol{z}) = A_K(\boldsymbol{z}) \quad \text{and} \quad A_i'(\boldsymbol{z}) = \begin{cases} A_i(\boldsymbol{z}) & \text{for } i \neq K \\ B(\boldsymbol{z}) & \text{for } i = K. \end{cases}$$

The displace operation is shift invariant, provided that Q is shift invariant. If Q is nonlinear (which is most frequently the case in practice), the displace operation is also nonlinear. One can easily confirm that, for any (reasonable) choice of Q, the displace operation is reversible. Since the rounding operator Q always yields an integer result, the displace operation is ITI in nature. If we neglect the effects of the rounding operator Q, the displace operation is linear, and therefore, its behavior can be characterized by a transfer matrix. This $M \times M$ matrix only has nonzero entries on the main diagonal and in the Kth row. Each of the entries along the main diagonal is one, except (possibly) for the entry in the Kth row. The entries along the Kth row are such that the (K,l)th entry has the form $\frac{1+(-1)^s A_l(\boldsymbol{z})}{1-B(\boldsymbol{z})}$ if $l = K$ and $\frac{(-1)^s A_l(\boldsymbol{z})}{1-B(\boldsymbol{z})}$ otherwise.

4) Exchange Operation: The exchange operation is associated with the M-input M-output network illustrated in Figure 1(d). As a matter of notation, the network inputs and outputs are, respectively, denoted as $\{x_i[\boldsymbol{n}]\}$ and $\{y_i[\boldsymbol{n}]\}$. In order to simplify the diagram, only

the Kth and Lth outputs and inputs are shown. All of the other outputs are directly connected to their corresponding inputs (i.e., $y_i[\boldsymbol{n}] = x_i[\boldsymbol{n}]$ except for $i \in \{K, L\}$). The exchange operation simply swaps the signals in the Kth and Lth channels. Mathematically, we have

$$y_i[\boldsymbol{n}] = \begin{cases} x_L[\boldsymbol{n}] & \text{for } i = K \\ x_K[\boldsymbol{n}] & \text{for } i = L \\ x_i[\boldsymbol{n}] & \text{otherwise.} \end{cases}$$

As a matter of notation, this type of operation is denoted as $\mathcal{E}(K, L)$.

The exchange operation is linear and shift invariant. Moreover, one can see that this operation is trivially ITI in nature. An exchange operation is also self inverting. That is, $\mathcal{E}^{-1}(K, L) = \mathcal{E}(K, L)$. Since the exchange operation (i.e., $\mathcal{E}(K, L)$) is linear, it can be characterized by a transfer matrix. This $M \times M$ matrix has all of its entries on the main diagonal equal to one, except the Kth and Lth entries which are zero. All of the off-diagonal entries are zero, except for the (K, L)th and (L, K)th entries which are one.

5) Shift Operation: The shift operation is associated with the M-input M-output network depicted in Figure 1(e). As a matter of notation, the network inputs and outputs are denoted as $\{x_i[\boldsymbol{n}]\}$ and $\{y_i[\boldsymbol{n}]\}$, respectively. In order to simplify the diagram, only the Kth input and Kth output are shown. Each of the remaining outputs is directly connected to its corresponding input (i.e., $y_i[\boldsymbol{n}] = x_i[\boldsymbol{n}]$ except for $i = K$). The shift operation forms the Kth output by translating the Kth input by the integer vector $\boldsymbol{m}$. Mathematically, we have

$$y_i[\boldsymbol{n}] = \begin{cases} x_i[\boldsymbol{n} - \boldsymbol{m}] & \text{for } i = K \\ x_i[\boldsymbol{n}] & \text{otherwise.} \end{cases}$$

As a matter of notation, we denote this type of operation as $\mathcal{T}(K, \boldsymbol{m})$.

The shift operation is linear and shift invariant. The inverse of a shift operation is another shift operation. More specifically, $\mathcal{T}^{-1}(K, \boldsymbol{m}) = \mathcal{T}(K, -\boldsymbol{m})$. Since the shift operation (i.e., $\mathcal{T}(K, \boldsymbol{m})$) is linear, it can be characterized by a transfer matrix. This $M \times M$ matrix is diagonal and has all of its diagonal entries equal to one, except for the Kth entry which is $\boldsymbol{z}^{-\boldsymbol{m}}$.

6) Scale Operation: The scale operation is associated with the M-input M-output network illustrated in Figure 1(f). The network inputs and outputs are, respectively, denoted as $\{x_i[\boldsymbol{n}]\}$ and $\{y_i[\boldsymbol{n}]\}$. To simplify the diagram, only the Kth output is shown. Each of the remaining outputs is directly connected to its corresponding input (i.e., $y_i[\boldsymbol{n}] = x_i[\boldsymbol{n}]$ except for $i = K$). Evidently, the scale operation forms the Kth output (i.e., $y_K[\boldsymbol{n}]$) by simply applying a multiplicative (scalar) gain s to the Kth input (i.e., $x_K[\boldsymbol{n}]$). Mathematically, we have

$$y_i[\boldsymbol{n}] = \begin{cases} s x_i[\boldsymbol{n}] & \text{for } i = K \\ x_i[\boldsymbol{n}] & \text{otherwise.} \end{cases}$$

As a matter of notation, we denote this type of operation as $\mathcal{S}(K, s)$.

The scale operation is linear and shift invariant. The inverse of a scale operation is another scale operation. More specifically, $\mathcal{S}^{-1}(K, s) = \mathcal{S}(K, s^{-1})$. Clearly, a scale operation is only invertible if its associated gain s is nonzero. This type of operation is ITI in nature if the gain s is an integer. For this reason, any scale operations used in the computation of a forward transform must employ integer gain factors. Since the scale operation (i.e., $\mathcal{S}(K, s)$) is linear, it can be characterized by a transfer matrix. This $M \times M$ matrix is diagonal, and has all of its diagonal entries equal to one, except for the Kth entry which is s.

III. Reversible ITI Wavelet Transforms

Having introduced the primitive reversible ITI operations employed by the GRITIT framework, we are now in a position to explain how such operations can be utilized in order to construct reversible ITI wavelet transforms. The basic building block for wavelet transforms is the uniformly maximally-decimated (UMD) filter bank. Consequently, in order to create a reversible ITI wavelet transform, we simply need to construct a reversible ITI UMD filter bank.

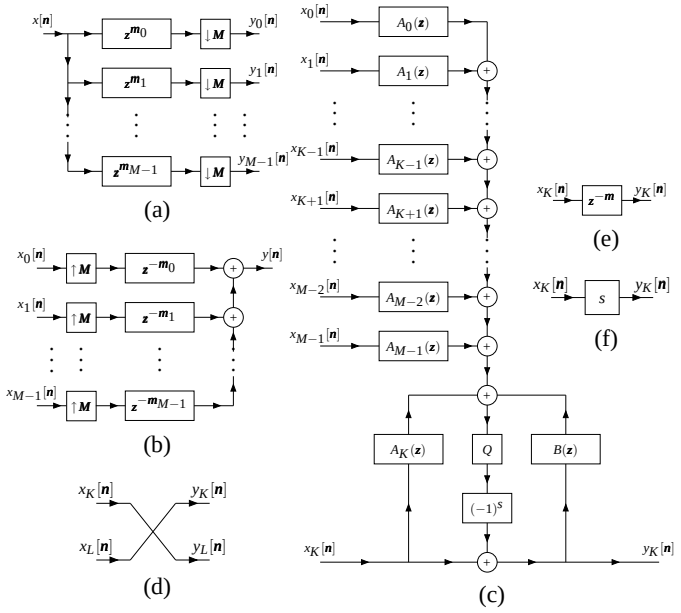

Fig. 1. The six types of operations used in the GRITIT framework. The (a) split, (b) join, (c) displace, (d) exchange, (e) shift, and (f) scale operations.

With the GRITIT framework, we can readily construct D-dimensional M-channel UMD filter banks based on polyphase techniques. Such filter banks are realized as follows. The analysis side of the filter bank, which corresponds to the forward wavelet transform, serves to decompose the input signal $x[\boldsymbol{n}]$ into M subband signals $\{y_i[\boldsymbol{n}]\}_{i=0}^{M-1}$. This transform is comprised of a split operation (i.e., $\mathcal{P}$) followed by one or more polyphase filtering operations (i.e., $\{T_i\}_{i=0}^{\eta-1}$). The polyphase filtering operations $\{T_i\}_{i=0}^{\eta-1}$ are chosen as displace, exchange, shift, and scale operations (as desired). The synthesis side of the filter bank, which corresponds to the inverse wavelet transform, generates the output signal $x[\boldsymbol{n}]$ from its corresponding subband signals (i.e., $\{y_i[\boldsymbol{n}]\}_{i=0}^{M-1}$). This is accomplished via the stepwise inversion of each operation from the analysis side. Since all of the operations employed (i.e., $\mathcal{P}$, $\mathcal{P}^{-1}$, $\{T_i\}$) are reversible, one can easily see that the overall transform is also reversible. Furthermore, since each operation on the analysis side maps integers to integers, the transform is also ITI.

IV. Reversible ITI Block Transforms

As suggested previously, the GRITIT framework is applicable not only to wavelet/subband transforms but also block transforms as well. Block transforms are associated with M-input M-output networks. In the case of block transforms, each of the inputs and outputs is comprised of a single sample and not a sequence (as in the wavelet/subband transform case). For this reason, some minor modifications are required to adapt the GRITIT framework to the block transform case. First, we discard the split, join, and shift operations, as such operations are only meaningful for sequences. This leaves us with the displace, exchange, and scale operations. Next, we redefine these remaining operations to operate on individual sample values, as opposed to sequences. This redefinition is trivial for the exchange and scale operations. In the case of the displace operation, we restrict all of its associated filters to have constant transfer functions. This, in effect, replaces the filters by simple

amplifiers.

With the aforementioned modifications in place, we can employ the GRITIT framework to construct reversible ITI block transforms. The forward transform is simply chosen as a sequence of displace, scale, and exchange operations (as desired). This maps the inputs $\{x_i\}$ to the outputs $\{y_i\}$. The inverse transform is then obtained through the stepwise inversion of each operation in the forward transform.

For example, by using the GRITIT framework, we can construct reversible ITI versions of linear block transforms such as the DCT and DFT. In fact, similar ideas have been recently proposed by a number of researchers [4]–[6].

V. GRITIT Realization of Wavelet/Block Transforms

Earlier, we introduced the GRITIT framework and explained how the primitive reversible ITI operations associated with this framework can be used to obtain general structures for reversible ITI wavelet/block transforms. We have yet to discuss, however, how one might choose the specific operations and corresponding parameters for a particular transform. That is, so far, we have not considered the transform design problem. We will now turn our attention to this matter.

The most common design technique is of an indirect nature. That is, we do not directly generate a reversible ITI transform. Instead, we first construct a linear wavelet/block transform with desirable properties. Then, we find a reversible ITI approximation of this transform. As we shall explain, this design approach is essentially equivalent to a matrix factorization problem. The particulars of the factorization problem differ, however, in the wavelet and block transform cases.

Wavelet Transforms: In the wavelet transform case, we first choose a specific polyphase decomposition for the corresponding filter bank. This determines the split operation to be used as well as the analysis polyphase matrix of the filter bank. Next, we must determine the polyphase filtering operations associated with the transform. To accomplish this, we simply need to decompose the analysis polyphase matrix into factors of the forms associated with the relevant GRITIT operations (i.e., the displace, exchange, shift, and scale operations). That is, we must decompose the analysis polyphase matrix $\boldsymbol{E}(\boldsymbol{z})$ as

$$\boldsymbol{E}(\boldsymbol{z}) = \boldsymbol{E}_{N-1}(\boldsymbol{z}) \cdots \boldsymbol{E}_1(\boldsymbol{z}) \boldsymbol{E}_0(\boldsymbol{z}) \tag{4}$$

where the $\{\boldsymbol{E}_i(\boldsymbol{z})\}$ are transfer matrices of the forms associated with the displace, exchange, shift, and scale operations. (The forms of such transfer matrices are specified in Section II.) Once we have determined the factorization in (4), the polyphase filtering operations in the forward transform are obtained by simply selecting the operations corresponding to each of the matrix factors (i.e., the $\{\boldsymbol{E}_i\}$). The inverse transform is trivially formed by the stepwise inversion of each of the operations in the forward transform.

Often, we are interested in wavelet transforms that are associated with FIR UMD filter banks. In this case, the corresponding analysis polyphase matrix $\boldsymbol{E}(\boldsymbol{z})$ has Laurent polynomial entries. Typically, in such a scenario, we want to obtain polyphase filtering operations that are associated with FIR filters. That is, in (4), we want to obtain a factorization with Laurent polynomial matrix factors. In the 1D case, in order to accomplish this, the factorization in (4) can be performed using a matrix Euclidean algorithm (e.g., as described in [7]). Provided that the transform is appropriately normalized, a solution to the factorization problem will always exist [7]. In the general D-dimensional case (where $D \geq 1$), this factorization problem is somewhat more complex, and a solution may not necessarily exist [8]. In some cases, however, a solution must exist as a consequence of a mathematical result known as Suslin's stability theorem [9]. To solve the factorization problem in the general D-dimensional case (assuming a solution exists), one can employ algorithms such as those proposed by Park and Woodburn [10] and Tolhuizen et al. [11].

Block Transforms: In the block transform case, we simply need to decompose the forward transform matrix into factors having the forms associated with the relevant GRITIT operations (i.e., the displace, exchange, and scale operations). Let us denote the forward transform matrix as $\boldsymbol{T}$. (We assume that $\boldsymbol{T}$ is a real matrix.) Then, we must decompose $\boldsymbol{T}$ as follows:

$$\boldsymbol{T} = \boldsymbol{T}_{N-1} \cdots \boldsymbol{T}_1 \boldsymbol{T}_0 \tag{5}$$

where the $\{\boldsymbol{T}_i\}$ are matrices of the forms associated with the displace, exchange, and scale operations. Once we have determined the factorization in (5), the operations in the forward transform are obtained by simply selecting the operations corresponding to each of the matrix factors (i.e., the $\{\boldsymbol{T}_i\}$). The inverse transform is trivially formed by the stepwise inversion of each of the operations in the forward transform.

The above factorization process can be performed by simple Gaussian elimination. In order to avoid an unnecessarily large number of factors, one might wish to employ a slightly more sophisticated technique like that proposed by Hao and Shi [5].

VI. Variations on the GRITIT Framework

The GRITIT framework, as described previously, constitutes a very powerful tool for the study and construction of reversible ITI wavelet/block transforms. In the interest of simplicity, we chose to describe this framework in the most basic form suitable to our needs herein. We would be remiss, however, if we failed to note that many variations on this framework are possible.

In this paper, we are interested in reversible ITI transforms that approximate linear transforms (e.g., wavelet/block transforms). For this reason, the displace operation is based on linear filtering operations. One could, however, just as easily employ other types of operations such as median filtering or morphological operators. To this end, one might exploit some of the ideas in [12]–[14].

Although adaptive transforms are not explicitly considered in this work, one can certainly construct such transforms using the GRITIT framework. That is, displace operations can certainly employ adaptive filters. For example, one might exploit adaptive filtering by using some of the related ideas in [15].

Another slight variation on the GRITIT framework can be obtained by changing the displace operations to employ modular arithmetic. By using modular arithmetic, one can construct transforms that avoid dynamic range growth. Such an idea has been proposed, for example, by Chao et al. [16] in the context of the lifting framework. In the opinion of this author, however, such an approach is of limited practical value in lossy signal coding applications. If modular arithmetic is employed, the transform behavior can become extremely nonlinear. That is, small perturbations in the transform coefficients can result in very large changes in the reconstructed signal. Obviously, such behavior is undesirable in the case of lossy coding, since the effect of transform coefficient quantization (on distortion) can become quite unpredictable.

When the GRITIT framework is used to construct reversible ITI wavelet transforms, the resulting computational structure is essentially a polyphase realization of a UMD filter bank. Such a structure is desirable from the standpoint of computational complexity, since analysis and synthesis filtering are performed in the downsampled domain (i.e., at the lower sampling density). We could, however, perform the analysis and synthesis filtering in the upsampled domain. An approach like this has been proposed by Komatsu and Sezaki [17],

[18]. From a computational standpoint, however, such an approach is less attractive (due to filtering operations being performed at the higher sampling density).

Although our focus herein is on ITI transforms, it is worth mentioning that one can also construct (reversible) real-to-real transforms using the GRITIT framework. This is accomplished by modifying the displace operation such that the rounding operator quantizes with some granularity finer than integers. For example, one might employ an operator that rounds results to the nearest integer multiple of 2^{-10}.

In passing, we note that the GRITIT framework can also be employed to build reversible ITI transmultiplexors (i.e., the "dual" of analysis-synthesis filter banks).

VII. Relationship Between Frameworks

As suggested earlier, the proposal of the GRITIT framework was partially motivated out of the desire to have a single unified view of frameworks for reversible ITI wavelet/block transforms. In what follows, we briefly examine the relationship between the GRITIT framework and each of a number of other frameworks, including the S+P transform, lifting, and ORT frameworks.

S+P Transform Framework: Let us consider the relationship between the GRITIT and S+P transform frameworks. In the case of the S+P transform framework [1], a forward transform can be expressed in operator notation as

$$\mathcal{L}\left(1,\text{bfloor},1,\left[\begin{smallmatrix}A(z) & -B(z)\end{smallmatrix}\right],0\right)\mathcal{L}\left(0,\lfloor\cdot\rfloor,0,\left[\begin{smallmatrix}0 & \frac{1}{2}\end{smallmatrix}\right],0\right)$$
$$\mathcal{L}\left(1,\emptyset,0,\left[\begin{smallmatrix}-1 & 0\end{smallmatrix}\right],0\right)\mathcal{E}(0,1)\mathcal{P}\left(2,\left[\begin{smallmatrix}0 & 1\end{smallmatrix}\right]\right). \quad (6)$$

The inverse transform is simply obtained through the stepwise inversion of the operations in the forward transform. From (6), we can see that the forward and inverse transforms are each comprised of a split/join, exchange, and three displace operations.

Lifting Framework: Next, we consider the relationship between the GRITIT and lifting frameworks. In the case of the lifting framework [2], we can express a forward transform in operator notation as

$$S_{\rho-1}\cdots S_1 S_0 L_{\lambda-1}\cdots L_1 L_0 P \quad (7)$$

where P is a split operation, the $\{S_i\}$ are scale operations, and the $\{L_i\}$ are displace operations. The scale operations $\{S_i\}$ are constrained to have nonzero integer scaling factors, in order for the resulting transform to be both reversible and ITI. The displace operations $\{L_i\}$ are constrained such that the $A_K(\boldsymbol{z})$ and $B(\boldsymbol{z})$ parameters in (3) are zero. From (7), we can see that the forward and inverse transforms are each comprised of one split/join operation, λ displace operations, and ρ scale operations.

ORT Framework: Now, we consider the relationship between the GRITIT and ORT frameworks. In the case of the ORT framework [3], a forward transform can be expressed in operator notation as

$$T_{\lambda-1}\cdots T_1 T_0 P \quad (8)$$

where P is a split operation, the $\{T_i\}$ are either shift or exchange operations or operations of the following forms:

$$\mathcal{S}(1,-s)\mathcal{L}(0,\lfloor\cdot\rfloor,0,\left[\begin{smallmatrix}0 & 1-H(z)\end{smallmatrix}\right],0)\mathcal{L}(1,\emptyset,0,\left[\begin{smallmatrix}-1 & 0\end{smallmatrix}\right],0), \quad (9a)$$
$$\mathcal{S}(1,\pm 1)\mathcal{L}(1,\emptyset,0,\left[\begin{smallmatrix}\pm 1 & 0\end{smallmatrix}\right],0)\mathcal{L}(0,\lfloor\cdot\rfloor,0,\left[\begin{smallmatrix}0 & H(z)\end{smallmatrix}\right],0), \quad \text{and} \quad (9b)$$
$$\mathcal{S}(1,s)\mathcal{L}(0,\lfloor\cdot\rfloor,0,\left[\begin{smallmatrix}0 & H(z)\end{smallmatrix}\right],0). \quad (9c)$$

The inverse transform can be trivially deduced through the stepwise inversion of each operation in the forward transform. Evidently, the forward and inverse transforms are each comprised of one split/join operation and several displace, shift, scale, and exchange operations. By comparing (8) and (7) and using the identities discussed in [19], one can show that the ORT framework is essentially a special case of the lifting framework with only trivial extensions. This observation was made possible, in part, through the insight provided by the GRITIT framework.

VIII. Conclusions

In this paper, we proposed the generalized reversible ITI transform (GRITIT) framework, a single unified framework for reversible ITI wavelet/block transforms. This new framework was then used to parameterize several previously developed frameworks. Having a unified framework at our disposal is quite beneficial as this allows interrelationships between previously proposed frameworks to be more easily studied and new insights to be obtained.

References

[1] A. Said and W. A. Pearlman, "An image multiresolution representation for lossless and lossy compression," *IEEE Trans. on Image Processing*, vol. 5, no. 9, pp. 1303–1310, Sept. 1996.

[2] W. Sweldens, "The lifting scheme: A new philosophy in biorthogonal wavelet constructions," in *Proc. of SPIE*, vol. 2569, San Diego, CA, USA, Sept. 1995, pp. 68–79.

[3] H.-Y. Jung and R. Prost, "Lossless subband coding system based on rounding transform," *IEEE Trans. on Signal Processing*, vol. 46, no. 9, pp. 2535–2540, Sept. 1998.

[4] S. Oraintara, Y. J. Chen, and T. Q. Nguyen, "Integer fast Fourier transform," *IEEE Trans. on Signal Processing*, vol. 50, no. 3, pp. 607–618, 2002.

[5] 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.

[6] Y. Zeng, L. Cheng, G. Bi, and A. C. Kot, "Integer DCTs and fast algorithms," *IEEE Trans. on Signal Processing*, vol. 49, no. 11, pp. 2774–2782, 2001.

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

[8] A. A. C. Kalker and I. A. Shah, "Ladder structures for multidimensional linear phase perfect reconstruction filter banks and wavelets," in *Proc. of SPIE*, vol. 1818, Boston, MA, USA, Nov. 1992, pp. 12–20.

[9] A. Suslin, "On the structure of the special linear group over polynomial rings," *Mathematics of the USSR Izvestija*, vol. 11, pp. 221–238, 1977.

[10] H. Park and C. Woodburn, "An algorithmic proof of Suslin's stability theorem for polynomial rings," *Journal of Algebra*, vol. 178, no. 1, pp. 277–298, Nov. 1995.

[11] L. Tolhuizen, H. Hollmann, and T. A. C. M. Kalker, "On the realizability of biorthogonal *m*-dimensional two-band filter banks," *IEEE Trans. on Signal Processing*, vol. 43, no. 3, pp. 640–648, Mar. 1995.

[12] H. J. A. M. Heijmans and J. Goutsias, "Nonlinear multiresolution signal decomposition schemes—part ii: Morphological wavelets," *IEEE Trans. on Image Processing*, vol. 9, no. 11, pp. 1897–1913, Nov. 2000.

[13] O. Egger, W. Li, and M. Kunt, "High compression image coding using an adaptive morphological subband decomposition," *Proc. of IEEE*, vol. 83, no. 2, pp. 272–287, Feb. 1995.

[14] H. Cha and L. F. Chaparro, "Adaptive morphological representation of signals: polynomial and wavelet methods," *Multidimensional Systems and Signal Processing*, vol. 8, no. 3, pp. 249–271, July 1997.

[15] O. N. Gerek and A. E. Cetin, "Adaptive polyphase subband decomposition structures for image compression," *IEEE Trans. on Image Processing*, vol. 9, no. 10, pp. 1649–1660, Oct. 2000.

[16] H. Chao, P. Fisher, and Z. Hua, "An approach to integer wavelet transforms for lossless for image compression," in *Proc. of International Symposium on Computational Mathematics*, Guangzhou, China, Aug. 1997, pp. 19–38.

[17] K. Komatsu and K. Sezaki, "Design of lossless block transforms and filter banks for image coding," *IEICE Trans. Fundamentals*, vol. E82-A, no. 8, pp. 1656–1664, Aug. 1999.

[18] ——, "Lossless filter banks based on two point transform and interpolative prediction," in *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, Phoenix, AZ, USA, Mar. 1999, pp. 1469–1472.

[19] M. D. Adams and F. Kossentini, "On the relationship between the overlapping rounding transform and lifting frameworks for reversible subband transforms," *IEEE Trans. on Signal Processing*, vol. 48, no. 1, pp. 261–266, Jan. 2000.