ISO/IEC JTC 1/SC 29/WG 1 **N 867**
Date: June 27, 1998

ISO/IEC JTC 1/SC 29/WG 1
(ITU-T SG8)

Coding of Still Pictures

JBIG	**JPEG**
Joint Bi-level Image Experts Group	Joint Photographic Experts Group

TITLE:	SBTLIB: A Flexible Computation Engine for Subband Transforms
SOURCE:	Michael D. Adams[†] and Faouzi Kossentini Dept. of Electrical and Computer Engineering University of British Columbia Vancouver, B.C., Canada V6T 1Z4 [†]`mailto:mdadams@ece.ubc.ca`
PROJECT:	JPEG 2000
STATUS:	Core Experiment Results
REQUESTED ACTION:	Discussion
DISTRIBUTION:	July 1998 Meeting of WG1

Contact:
ISO/IEC JTC 1/SC 29/WG 1 Convener—Dr. Daniel T. Lee
Hewlett-Packard Company, 11000 Wolfe Road, MS42U0, Cupertino, California 95014, USA
Tel: +1 408 447 4160, Fax: +1 408 447 2842, E-mail: Daniel_Lee@hp.com

Abstract

SBTLIB, a software library for the computation of subband transforms, is described. This library is implemented in the C programming language, and was developed in order to provide a flexible subband transform engine for use in several JPEG-2000 core experiments. Due to the many new functionalities offered by the SBTLIB software, the integration of this code into the official VM software distribution could prove beneficial to many.

1 Introduction

Subband transforms, such as wavelet and wavelet packet transforms, have proven to be an extremely useful tool in many digital signal processing applications. The SBTLIB software provides a flexible engine for calculating such transforms. This software is written in the C programming language, and is packaged in the form of a library.

Subband transforms can be efficiently computed using tree-structured filter banks (TSFBs) with quadrature mirror-image filter bank (QMFB) building blocks. The SBTLIB software uses such a computational structure where each QMFB is implemented using its lifting realization. The lifting framework was chosen due to its numerous advantages.

The SBTLIB software was developed to provide a flexible subband transform engine for use in several JPEG-2000 core experiments. Although the current VM has code for calculating subband transforms, this code is somewhat inflexible. By replacing the existing transform code in the VM with SBTLIB, many new functionalities are obtained:

- The TSFB underlying a particular transform can employ different QMFBs. That is, each of the QMFBs used to form a TSFB can be independently specified. This allows one, for example, to use different QMFBs for decomposition in the horizontal and vertical directions. With the original VM software, however, the same QMFB building block must be used everywhere in a TSFB.

- Reversible and nonreversible versions of transforms are handled together in a uniform manner. A transform is specified only once, but can be calculated in both a reversible or nonreversible manner. With the original VM code, however, reversible and nonreversible transforms are specified in completely different ways. This makes the original software more difficult to use.

- Transforms based on QMFBs with IIR synthesis filters are supported, such as the S+P family of transforms. The original VM code can only handle transforms based on FIR QMFBs.

- Any transform can be calculated using either fixed-point or floating-point arithmetic independent of whether the transform is calculated in a reversible manner. This cannot be done with the original VM software.

- The software does not assume that only 2-channel QMFBs are used, and is capable of supporting M-channel QMFBs. The current VM software, however, can only handle 2-channel QMFBs.

Due to the modular nature of the SBTLIB code, incorporating it into the VM software was a relatively simple exercise. Feedback from other people using the SBTLIB-enhanced version of the VM software has been quite positive. For this reason, it is felt that this software is worth adopting into the official VM software distribution.

In the remainder of this paper, the lifting framework used by SBTLIB to realize subband transforms is described at length. This discussion is then followed by a detailed description of the SBTLIB user interface. In particular, the format of the data files used to specify subband transforms is described.

2 Reversible Wavelet Transforms

2.1 Overview

In this section, we present the lifting-based structure used by SBTLIB to realize wavelet transforms. As will be seen, this computational structure readily facilitates both reversible and integer-to-integer transforms.

2.2 Wavelet Transforms

Although, in principle, wavelet transforms can be computed directly using matrix multiplication, such an approach is usually very inefficient. Instead, it is more practical to express the transform in terms of filtering, upsampling, and downsampling operations. This leads to an efficient computational structure for wavelet transforms based on multirate filter banks, or more specifically, tree-structured filter banks (TSFBs) with quadrature mirror-image filter bank (QMFB) building blocks.

2.3 Quadrature Mirror-Image Filter Banks

Although many filter bank configurations exist, an extremely useful one is the so called M-channel quadrature mirror-image filter bank (QMFB). Such a system is more correctly referred to as a uniformly maximally decimated filter bank, but the term QMFB is often used for historical reasons. The general structure of such a system is shown in Figure 1. The input signal $x[n]$ is split into M channels, processed by the analysis filters H_k, downsampled by M, upsampled by M, transformed by the synthesis filters F_k, and then summed to produce the output signal $\hat{x}[n]$. Since each of the M channels is downsampled and upsampled by M, the filter bank is referred to as uniformly maximally decimated.

2.4 Perfect Reconstruction QMFBs

For the QMFB of Figure 1 to be of practical use, the output $\hat{x}[n]$ is usually required to be an accurate reproduction of the input $x[n]$. If the system is such that $\hat{x}[n] = x[n - n_0]$ for all $x[n]$ and for some integer n_0, the system is said to have the perfect reconstruction (PR) property. In other words, a PR system can reproduce the input signal exactly except for a time shift. Generally, there are three reasons that the reconstructed signal $\hat{x}[n]$ can differ

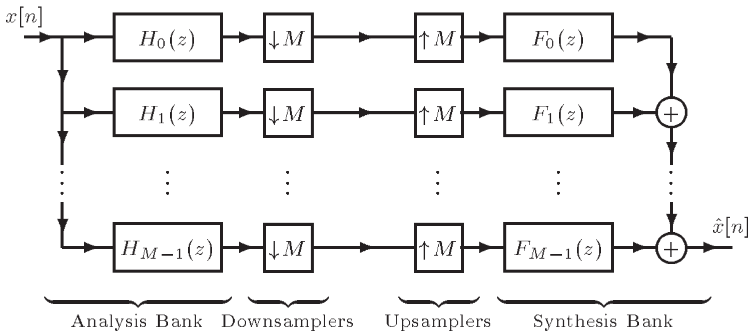

Figure 1: M-channel QMFB

from $x[n]$: aliasing distortion, amplitude distortion, and phase distortion. The analysis and synthesis filters can be designed in such a way as to eliminate some or all of these distortions depending on what is required by the application.

Since our interests lie with invertible transforms, we focus exclusively on QMFBs with the PR property in what follows. This said, however, it is important to understand that the notion of PR is often only a theoretical one. That is, when one states that a QMFB has the PR property, it is tacitly assumed that the filter bank performs all arithmetic calculations exactly (i.e., with infinite precision). In most computer applications, however, finite-precision arithmetic is employed. Such arithmetic is inexact due to roundoff error. Consequently, when a PR QMFB is implemented in finite-precision arithmetic, we usually obtain a physical realization that is not truly PR. In order to maintain the PR property in the presence of roundoff error, great care must be taken and special computational structures employed.

2.5 Polyphase Form of a QMFB

Although the structure for the M-channel QMFB shown in Figure 1 may be intuitively appealing, it is often not the most convenient structure with which to work. This leads us to the polyphase form of a QMFB. The polyphase representation has many advantages, but most importantly it simplifies many theoretical results and suggests an efficient means by which to implement QMFBs.

Many different variations on the polyphase form of a QMFB exist, but for our purposes we only need consider one here. The polyphase form of concern to us is shown in Figure 2. The polyphase representation, in effect, converts a QMFB from a SISO system to a MIMO system operating on the M polyphase components of the original input signal. As a matter of terminology, $\boldsymbol{E}(z)$ and $\boldsymbol{R}(z)$ are referred to as the analysis and synthesis polyphase matrices of the QMFB, respectively. The analysis and synthesis polyphase matrices are directly determined by the analysis and synthesis filters.

It is important to note that with the polyphase form of a QMFB, analysis filtering is performed after downsampling and synthesis filtering is performed before upsampling. In other words, all filtering is performed in the downsampled domain (i.e., at the lower sampling

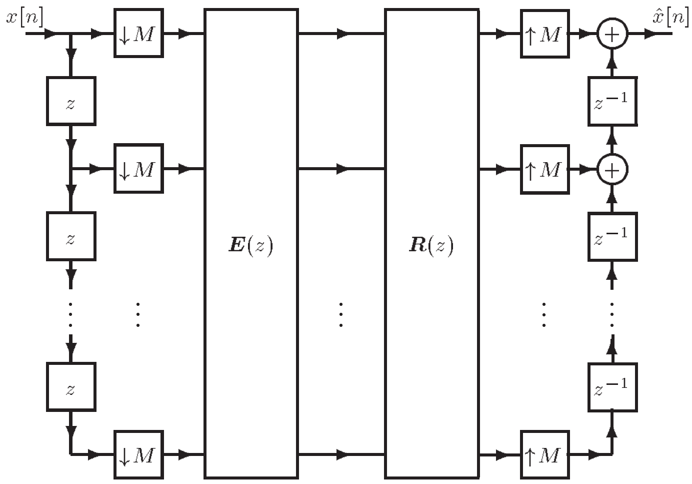

Figure 2: Polyphase form of an M-channel QMFB

rate). This leads to improved computational efficiency over a QMFB implemented in its canonical form. Consequently, most practical implementations of QMFBs make use of the polyphase form.

2.6 Lifting Realization of a QMFB

Although the lifting realization of a QMFB has become quite popular recently, the idea on which it is based has been around for at least several years. This realization strategy is based on a ladder network implementation of a QMFB's polyphase matrices. Such ladder networks were first proposed by Bruekers and Van Den Enden [2] and were also studied in some detail by Kalker and Shah [5] [6].

Until more recently, many of the benefits of the lifting realization were not fully appreciated. In the context of wavelet transforms, Sweldens first coined the term "lifting", and detailed the many advantages of this realization strategy [8] [9]. In his work with Daubechies [4], he also demonstrated that any 2-band wavelet transform associated with FIR PR QMFBs can be implemented using lifting. Not long afterwards, Calderbank et al. [3] described how the lifting realization provides a convenient framework for computing reversible integer-to-integer transforms.

In what follows, we discuss a generalization of the lifting realization. This generalization is presented briefly in [1] and was inspired in part by the S+P family of transforms [7]. Although the realization structure we describe is slightly different from the standard lifting realization, the basic idea remains the same. With our minor modifications, however, we are able to realize a slightly larger class of QMFBs—namely, some QMFBs with IIR synthesis filters.

With the lifting realization, a QMFB is implemented in its polyphase form. The distinguishing feature of this realization is the type of filtering network used to realize the polyphase matrices. The general structure of the lifting realization for an M-channel

QMFB is shown in Figure 3. As is evident from this diagram, the polyphase matrices are implemented by employing a number of cascaded filtering blocks. Four types of blocks are employed, corresponding to two types of operations and their inverses. The two types of operations are referred to as lifting and scaling operations. The lifting, inverse lifting, scaling, and inverse scaling operations are denoted L, L^{-1}, S, and S^{-1}, respectively. The four types of filtering blocks employed have the following structure:

- Lifting (L) block. In this block, all outputs except one are connected directly to their corresponding inputs. The remaining output is formed by adding an adjustment value to its corresponding input. The adjustment value is obtained by summing filtered versions of one or more inputs and then quantizing the result.

- Inverse lifting (L^{-1}) block. This block has the same structure as a lifting block except the adjustment value is subtracted instead of being added.

- Scaling (S) block. In this block, a single input is multiplied by a constant gain to produce its corresponding output. All other inputs are passed through to their corresponding outputs unchanged.

- Inverse scaling (S^{-1}) block. This block has the same general form as a scaling block.

The analysis polyphase matrix is implemented by one or more lifting operations followed by one or more scaling operations. Similarly, the synthesis polyphase matrix is realized by one or more inverse scaling operations followed by one or more inverse lifting operations. In fact, the synthesis side of the QMFB is nothing more than the stepwise inverse of the analysis side.

In the 2-channel case, the L and L^{-1} blocks have the general form shown in Figure 4 and the S and S^{-1} blocks have the form shown in Figure 5. The blocks labelled with the symbol Q are quantizers. In order for the system to be realizable, there are some constraints on the parameters of these blocks:

- In the case of lifting operations, the filter A_{i_0} must have a transfer function with strictly positive powers or strictly negative powers of z (or be identically zero). The transfer function must not have a constant term in order to avoid a delay-free loop which is physically unrealizable. In addition, the transfer function must not have both positive and negative powers of z to facilitate causal inversion.

- In the case of scaling operations, the scaling factor α cannot be zero. This condition is normally not of concern, however, since the lifting realization of any PR QMFB will always have nonzero scaling factors.

Any appropriately normalized PR QMFB with FIR filters can be realized using the above strategy. (For a proof in the general M-channel case, see [1].) Moreover, some PR QMFBs with FIR analysis filters and IIR synthesis filters can also be implemented using this structure. The S+P family of transforms [7] is one such example. One can see that this slightly more general lifting realization degenerates into the standard lifting case as described by Sweldens when the $A_{i_0}(z)$ are all chosen to be zero.

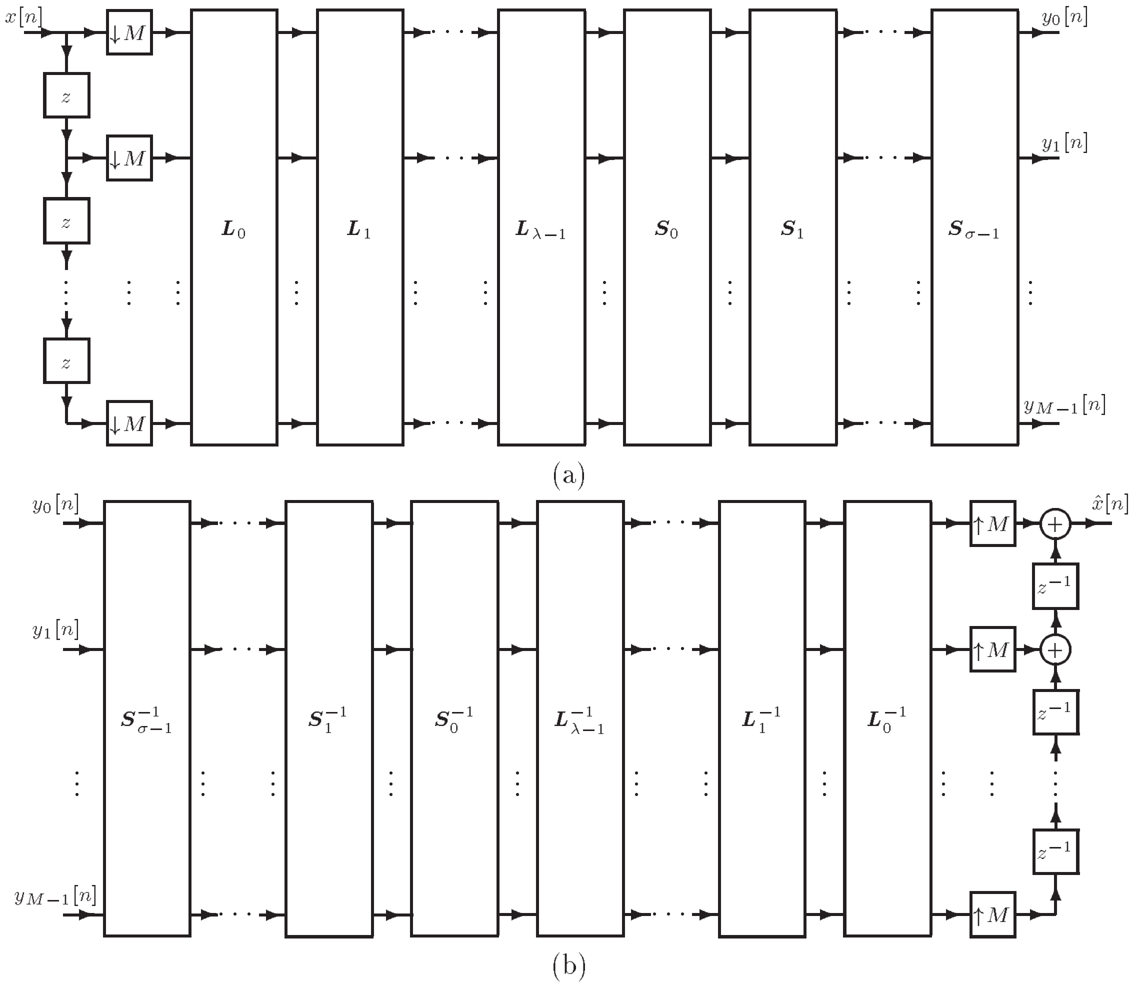

Figure 3: Lifting realization of M-channel QMFB. (a) Analysis side. (b) Synthesis side.

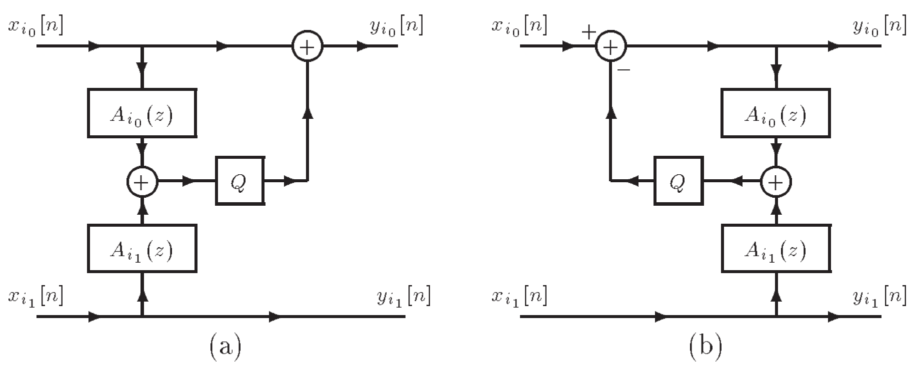

Figure 4: Filtering structure for lifting operation in the 2-channel case. (a) Lifting. (b) Inverse Lifting.

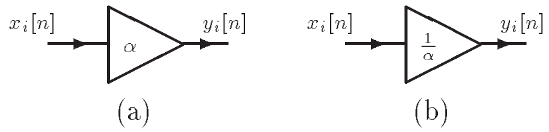

Figure 5: Filtering structure for scaling operation. (a) Scaling. (b) Inverse Scaling.

One remarkable property of the lifting realization is that provided the scaling factors on the analysis side of the QMFB are integers, one can maintain the PR property of the QMFB even when finite-precision arithmetic is used. Suppose now that the quantizers in all of the lifting operations round their input to some arbitrary, but fixed, precision. For example, the quantizers might round to the nearest multiple of 2^{-11}. By using quantization in this way, we can preserve the PR property even though inexact arithmetic is employed. As an extreme case, the quantizers could even round their input to an integer value. This scenario is particularly useful when QMFBs that map integers to integers are desired. In fact, it is this reversible integer-to-integer transform scenario that is of interest in the context of this paper.

In the discussion above, we assumed that the scaling factors on the analysis side of the QMFB were integers. From a practical viewpoint, however, it does not really matter if the scaling factors are integers. If the scaling factors are not integers, we can simply avoid performing the scaling operations. Doing this has the effect of scaling each of the subband signals by a constant factor. To compensate for this undesired scaling effect, one can simply apply weights to the transform coefficients. This achieves the same end result as if the original scaling operations in the lifting realization were still present. Yet, at the same time, the QMFB is still PR in finite-precision arithmetic.

2.7 Tree-Structured Filter Banks

As shown in Figure 6, the analysis side of a QMFB decomposes the input signal $x[n]$ into M subband signals $y_k[n]$. The synthesis side then recombines these subband signals to obtain $\hat{x}[n]$, the reconstructed version of the original signal. There is nothing, however, to prevent additional QMFBs from being employed to further decompose some or all of the subband signals $y_k[n]$. Of course, some or all of the resulting subband signals can again be decomposed with even more QMFBs. An approach like this generally leads to what is called a tree-structured filter bank (TSFB). If a TSFB is such that

- only the lowpass subband signal is decomposed at each level in the tree,
- the same basic QMFB building block is used for decomposition at all levels,
- and this basic block has PR and satisfies certain regularity conditions,

then the filter bank computes a wavelet transform. In the more general case, the filter bank is associated with a wavelet packet or subband transform.

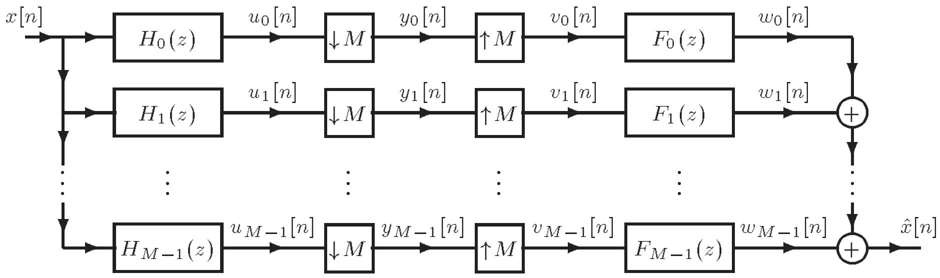

Figure 6: M-channel QMFB (revisited)

2.8 Finite-Length Signals

TSFBs are constructed from QMFBs by arranging them in a tree structure. The QMFBs are associated with nonexpansive transforms, and consequently, so are the TSFBs. Since the signals being transformed are finite in length, some scheme must be adopted to handle boundary conditions. Two different schemes are employed by this software. Symmetric extension is used when possible. Otherwise, a more general boundary filtering method is employed. One of the goals of this software was to avoid the need for the user to explicitly specify boundary filters.

3 User Interface

3.1 File Types

There are four types of data files employed by this software:

- QMFB Description File. This type of file specifies the structure of a particular QMFB.
- QMFB Catalog File. This type of file lists all of the known QMFBs and their associated QMFB description files.
- TSFB Description File. This type of file specifies the connection topology for a TSFB. That is, it specifies how various QMFBs are connected together in order to form a decomposition tree.
- TSFB Catalog File. This type of file lists all of the known TSFBs and their associated TSFB description files.

3.2 QMFB Description File

A QMFB description file is used to specify the structure of a QMFB. It is important to note that only the structure of the analysis side of the QMFB is specified in this file. The structure of the synthesis side follows immediately from the analysis side due to the symmetry of the lifting realization (and PR constraint). The QMFB description file consists of the following information in order:

- The number of channels in the QMFB. For the time being, only 2-channel QMFBs are supported.
- The number of lifting steps.
- For each lifting step, the following information is then provided:
  - The output channel.
  - The number of inputs.
  - For each input, the following information is then given:
    * The input channel.
    * The impulse response of the associated filter. The format of the impulse response data is described later in this section.
- The number of scaling steps.
- For each scaling step, the following information is provided:
  - The channel number.
  - The scaling factor.

The impulse response of a filter is specified using the following fields in order:

- The time-index of the first nonzero sample in the impulse response.
- The length of the impulse response sequence.
- The samples of the impulse response sequence (in order of increasing time).

All fields in the file are whitespace delimited. Comments begin with two consecutive slash characters and proceed until a newline character is encountered (as in C++). Boundary filters do not need to be specified explicitly. This software handles signal boundaries internally. The user does not need to specify explicit boundary filters.

3.3 QMFB Description File Example: The Haar Transform

For illustration purposes, the QMFB description file for the well-known Haar transform is included below. This example corresponds to the QMFB shown in Figure 8.

3.4 QMFB Description File Example: The S+P Transform

To further illustrate the format of the QMFB description file, another example is now provided. In this case, we consider a slightly more complicated example: the S+P transform with predictor B. The QMFB associated with this transform is shown in Figure 9. Notice that the third lifting step for the S+P transform has two inputs. This implies that the QMFB is associated with IIR filters.

```
// The number of channels.
2

// The number of lifting steps.
2

// The information for the first lifting step.
1 1
0 0 1 -1

// The information for the second lifting step.
0 1
1 0 1 0.5

// The number of scaling steps.
2

// The information for the first scaling step.
0 1.414213562373095

// The information for the second scaling step.
1 0.7071067811865475
```

Figure 7: QMFB description file for the Haar transform.

3.5 QMFB Catalog File

The QMFB catalog file simply lists all of the user-defined QMFBs. There is one entry in this file for each QMFB. Each entry consists of two whitespace separated fields that are in order:

- The name of the QMFB. This string is used to identify the QMFB and may contain letters, numbers, and the underscore character.
- The pathname of the description file for this QMFB. If the pathname is relative, it is interpreted relative to the location of the QMFB catalog file.

3.6 TSFB Description File

A TSFB is described in terms of a decomposition tree. Each node in the tree is associated with a pair of QMFBs, one being used for decomposition in the horizontal direction and one for the and vertical direction. The QMFBs used in each direction are specified separately and need not be the same. Furthermore, it is not necessary to perform a decomposition

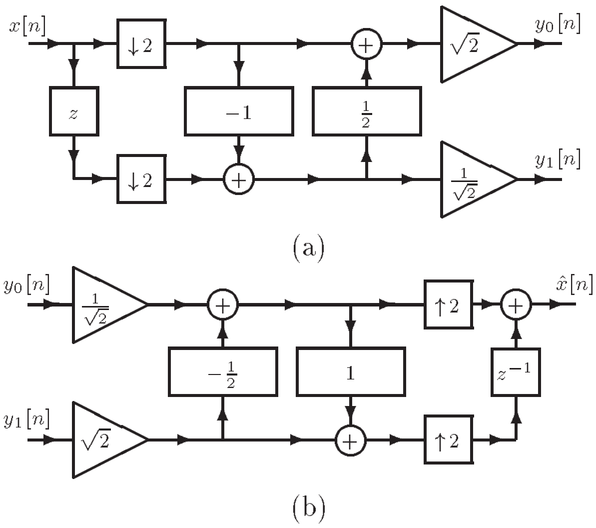

Figure 8: QMFB for the Haar Transform. (a) Analysis side (forward transform). (b) Synthesis side (inverse transform).

in both directions. This provides additional flexibility. For example, it allows the code to support 1D transforms as a special case. The maximum number of children for a particular node is equal to the number of bands associated with that node. If only decomposition in the horizontal or vertical direction is performed, the number of bands is simply the number of channels in the horizontal or vertical QMFB, respectively. Otherwise, the maximum number of bands is simply the product of the number of channels for the two QMFBs.

A TSFB description file is used to specify the structure of a TSFB. Such a file consists of pairs of QMFB names which specify a particular decomposition tree. To generate the TSFB description file for a particular tree, one traverses the tree one level at a time (left to right within a level), and lists the horizontal and vertical decomposition QMFBs for each node as it is visited. Note that this is not a depth-first traversal of the tree!

Comments begin with two consecutive slash characters and proceed until the first newline character is encountered. When the file is processed, any occurrence of the keyword "BASENAME" is replaced with the basename of the file containing the TSFB description. This is useful in cases where one wants to use the same tree decomposition with different QMFB building blocks. In this way, by using symbolic links, one can easily repeat the same decomposition structure without the need for multiple files.

3.7 TSFB Description File Example: The 3-Level Haar Wavelet Transform

Assuming that the QMFB for the Haar transform has been named "haar", the TSFB description file for a 3-level 2D Haar wavelet transform would be as illustrated in Figure 11:

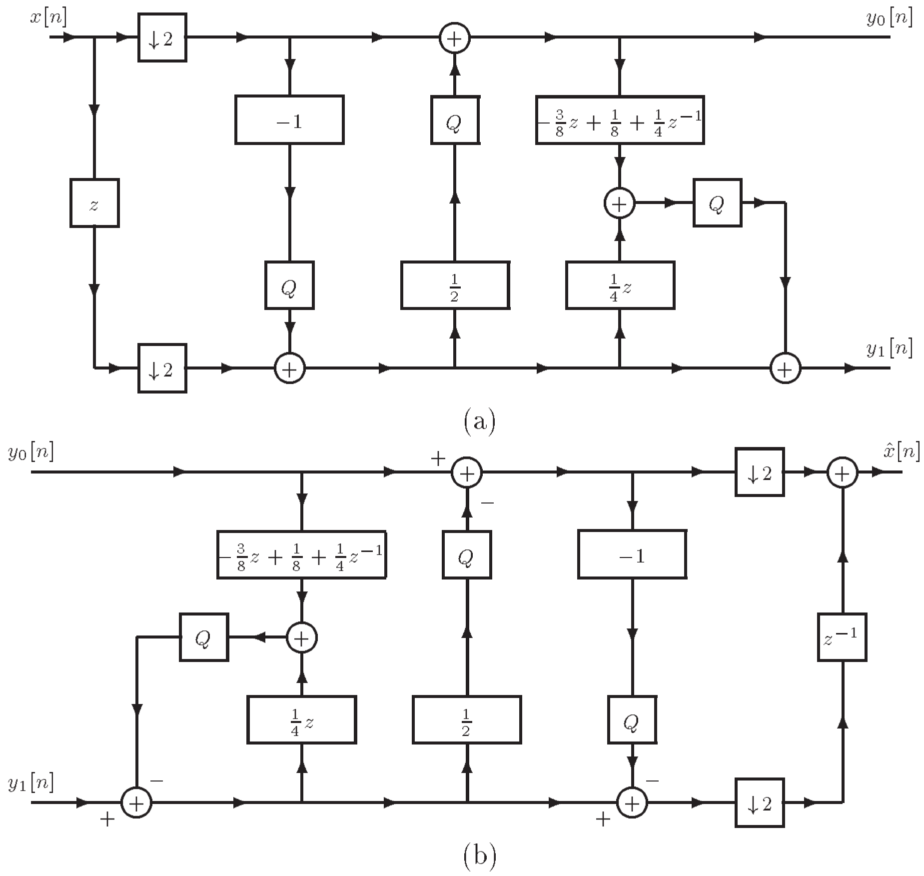

Figure 9: S+P transform with predictor B. (a) Analysis side (forward transform). (b) Synthesis side (inverse transform).

3.8 TSFB Catalog File

The TSFB catalog file simply lists all of the user-defined TSFBs. There is one entry in this file for each TSFB. Each entry consists of three whitespace separated fields which are in order:

- The ID for the TSFB. This is a nonnegative integer that uniquely identifies the TSFB.
- The name of the TSFB. This string is also used to identify the TSFB and may contain letters, numbers, and the underscore character.
- The pathname of the description file for this TSFB. If the pathname is relative, it is interpreted relative to the location of the TSFB catalog file.

```
// The number of channels.
2

// The number of lifting steps.
3

// The information for the first lifting step.
1 1
0 0 1 -1

// The information for the second lifting step.
0 1
1 0 1 0.5

// The information for the third lifting step.
1 2
0 -1 3 -0.375 0.125 0.25
1 -1 1 0.25

// The number of scaling steps.
2

// The information for the first scaling step.
0 1.414213562373095

// The information for the second scaling step.
1 0.7071067811865475
```

Figure 10: QMFB description file for the S+P transform.

4 Summary

The SBTLIB software was described. This code has been successfully incorporated into the VM software (Release 0.0), and the new functionality which results has proven to be extremely beneficial. For this reason, we would like to offer the SBTLIB code for integration into the official VM software distribution. It is our belief that incorporating SBTLIB into the baseline coder will allow future core experiments relating to subband transforms to be performed much more easily.

```
// The first level of the tree.
haar haar

// The second level of the tree.
haar haar null null null null null null

// The third level of the tree.
haar haar null null null null null null

// The fourth level of the tree.
null null null null null null null null
```

Figure 11: TSFB description file for 3-level Haar wavelet transform.

References

[1] M. D. Adams. Reversible wavelet transforms and their application to embedded image compression. M.A.Sc. thesis, Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada, January 1998. Available from `http://www.ece.ubc.ca/~mdadams/papers/mascthesis.ps.Z`.

[2] F. A. M. L. Bruekers and A. W. M. van den Enden. New networks for perfect inversion and perfect reconstruction. *IEEE Journal on Selected Areas in Communications*, 10(1):130–137, January 1992.

[3] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. Technical report, Department of Mathematics, Princeton University, August 1996. Available from `http://cm.bell-labs.com/who/wim/papers/integer.ps`.

[4] I. Daubechies and W. Sweldens. Factoring wavelet transforms into lifting steps. Technical report, Bell Laboratories, Lucent Technologies, 1996. Available from `http://cm.bell-labs.com/who/wim/papers/factor.ps`.

[5] A. A. C. Kalker and I. A. Shah. Ladder structures for multidimensional linear phase perfect reconstruction filter banks and wavelets. In *Proceedings of SPIE: Visual Communications and Image Processing*, volume 1818, pages 12–20, Boston, MA, November 1992.

[6] T. A. C. M. Kalker and I. A. Shah. On ladder structures and linear phase conditions for multidimensional bi-orthogonal filter banks. Preprint, June 1993.

[7] A. Said and W. A. Pearlman. An image multiresolution representation for lossless and lossy compression. *IEEE Trans. on Image Processing*, 5(9):1303–1310, September 1996.

[8] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In *Proceedings of SPIE Wavelet Applications in Signal and Image Processing III*, pages 68–79, 1995. Available from `ftp://ftp.math.sc.edu/pub/wavelet/papers/varia/spie95.ps.gz`.

[9] W. Sweldens. The lifting scheme: A custom-design construction of biorthogonal wavelets. *Applied and Computational Harmonic Analysis*, 3(2):186–200, 1996. Available from `ftp://ftp.math.sc.edu/pub/imi_94/imi94_7.ps`.

A List of Abbreviations

FIR	Finite-Duration Impulse Response
IIR	Infinite-Duration Impulse Response
MIMO	Multi-Input Multi-Output
PR	Perfect Reconstruction
QMFB	Quadrature Mirror-Image Filter Bank
TSFB	Tree-Structured Filter Bank
SISO	Single-Input Single-Output